WEB STAR STUDIO

COLEÇÃO DOCUMENTAL CONSOLIDADA

MÉTODO DE DESENVOLVIMENTO DE SITES

8 Manuais · 12 SOPs
Índice Oficial · Termo de Atualização

VERSÃO
v1.1 FINAL

DATA
11/09/2026

STATUS
COLEÇÃO CANÔNICA VIGENTE

WEB STAR STUDIO

GOVERNANÇA DOCUMENTAL

MÉTODO DE DESENVOLVIMENTO DE SITES

ÍNDICE OFICIAL
Índice Oficial da Coleção

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — COLEÇÃO CANÔNICA VIGENTE

GOVERNANÇA DOCUMENTAL

ÍNDICE OFICIAL

Índice Oficial da Coleção

ARQUIVO	WSSMetodoDesenvolvimentoSitesIndiceColecaov1.1_FINAL.md
VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — COLEÇÃO CANÔNICA VIGENTE
ESCOPO	porta de entrada oficial da documentação humana do Método de Desenvolvimento de Sites WSS

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 5

2. Escopo da coleção v1.1 5

3. Estado documental 5

PARTE I — MANUAIS CANÔNICOS 7

4. Manual 1 — Visão Geral e Toolchain 8

5. Manual 2 — Entrada e Contexto do Projeto 8

6. Manual 3 — Construção Visual 8

7. Manual 4 — Projeto-Base, Admin e Funcionalidades 8

8. Manual 5 — Desenvolvimento com Codex, Lovable e GitHub 8

9. Manual 6 — Testes, QA, Debugging, Segurança e SEO 8

10. Manual 7 — Publicação, Entrega e Pós-Publicação 8

11. Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais 9

PARTE II — SOPS CANÔNICOS 10

12. SOP-001 — Contextualizar um Novo Projeto WSS 11

13. SOP-002 — Iniciar Tecnicamente um Projeto WSS: Base → Lovable → GitHub → Ambiente Local 11

14. SOP-003 — Construir e Refinar a Home a partir das Referências 11

15. SOP-004 — Construir Páginas Internas Mantendo Consistência Visual 11

16. SOP-005 — Configurar Admin e Módulos Aplicáveis 11

17. SOP-006 — Configurar e Validar Chatbot e Base de Conhecimento 11

18. SOP-007 — Portar Capacidade da Base para Projeto Legado 12

19. SOP-008 — Executar QA Visual, Funcional e Responsivo 12

20. SOP-009 — Diagnosticar e Corrigir Bug por Camada 12

21. SOP-010 — Tratar Auditoria SEO, AEO e GEO 12

22. SOP-011 — Processar Rodada de Feedback via ajustes.md 12

23. SOP-012 — Preparar, Publicar, Validar Produção e Entregar Admin 12

PARTE III — MAPA DE PROPRIEDADE 13

24. Manual proprietário → SOPs relacionados 14

PARTE IV — FLUXO PONTA A PONTA 15

25. Mapa macro 16

PARTE V — PROCESSO SITES WSS × MÉTODO DE DESENVOLVIMENTO 17

26. Processo Sites WSS 18

27. Método de Desenvolvimento de Sites WSS 18

PARTE VI — HIERARQUIA DOCUMENTAL 19

28. Hierarquia de uso 20

PARTE VII — AUTORIDADE SEO/AEO/GEO 21

29. Norma canônica vigente 22

30. AS-IS × norma 22

PARTE VIII — SKILLS 23

31. Camada posterior e adicional 24

PARTE IX — DECISÕES ESTRATÉGICAS ABERTAS 25

32. Itens não resolvidos por esta coleção 26

PARTE X — HISTÓRICO E STATUS 27

33. Histórico mínimo 28

34. Estado final da documentação humana v1.1 28

1. Finalidade

Este documento é a **porta de entrada oficial da coleção canônica do Método de Desenvolvimento de Sites WSS**.

Sua função é:

- orientar a navegação entre Manuais e SOPs;
- indicar a propriedade de cada procedimento;
- mostrar o fluxo macro de execução;
- preservar a relação entre Método, Processo Sites WSS e normas técnicas;
- registrar o estado documental da coleção.

Este arquivo **não é um Manual**, não substitui os SOPs e não redefine o conteúdo já ratificado.

O antigo `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT` permanece como artefato da etapa de descoberta e **não é substituído nem atualizado por este documento**.

---

2. Escopo da coleção v1.1

A coleção documenta como a WSS transforma os insumos consolidados de um projeto de site em uma implementação desenvolvida, validada, publicada e entregue.

Ela cobre, no nível apropriado:

- entrada e contextualização;
- toolchain;
- preparação técnica;
- construção visual;
- Base Admin e módulos;
- desenvolvimento com Lovable, GitHub, Codex e Claude;
- QA e debugging;
- SEO/AEO/GEO no método;
- publicação e entrega;
- boas práticas, anti-patterns, troubleshooting e casos reais;
- procedimentos operacionais associados.

A coleção não redefine o Processo Sites WSS e não substitui normas técnicas especializadas.

---

3. Estado documental

A documentação humana principal do Método WSS permanece materialmente encerrada e passa a ter como autoridade documental vigente a coleção `v1.1 FINAL`.

Coleção canônica

- **8 Manuais** `v1.1 FINAL`
- **12 SOPs** `v1.1 FINAL`
- **1 Índice Oficial da Coleção** `v1.1 FINAL`

Os Manuais e SOPs foram auditados, formalizados e ratificados pela Coordenação. A `v1.1 FINAL` incorpora exclusivamente o Pacote Corretivo Pós-Entrega decorrente da validação do Dev, além da normalização referencial administrativa necessária ao fechamento. A coleção `v1.0 FINAL` permanece preservada como baseline histórico.

Não fazem parte do fechamento documental v1.1:

- Skills;
- novos SOPs;
- novos Manuais;
- redesenho TO-BE.

Registro de atualização documental

A transição de autoridade entre as coleções está formalizada em:

`WSS_Termo_Atualizacao_Substituicao_Documental_Metodo_v1.1_FINAL.md`

Esse termo preserva a coleção `v1.0 FINAL` como baseline histórico, sem edição em lugar, e registra a coleção `v1.1 FINAL` como autoridade documental vigente.

PARTE I — MANUAIS CANÔNICOS

4. Manual 1 — Visão Geral e Toolchain

Arquivo: `WSS_Manual_01_Visao_Geral_Toolchain_v1.1_FINAL.md`

Apresenta o Método WSS, sua fronteira com o Processo Sites WSS, o fluxo macro e o papel das principais ferramentas.

5. Manual 2 — Entrada e Contexto do Projeto

Arquivo: `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`

Documenta o handoff WSS → Dev, contextualização, prompt master, fontes de projeto, pendências e preservação de contexto.

6. Manual 3 — Construção Visual

Arquivo: `WSS_Manual_03_Construcao_Visual_v1.1_FINAL.md`

Documenta referências, Home, páginas internas, consistência, reuso, assets, efeitos, responsividade, performance e acessibilidade no recorte do desenvolvimento visual.

7. Manual 4 — Projeto-Base, Admin e Funcionalidades

Arquivo: `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`

Documenta a Base Admin WSS, módulos operacionais, Blog, Leads, formulários, Chat/CTA, Chatbot, perfis e portabilidade para legados.

8. Manual 5 — Desenvolvimento com Codex, Lovable e GitHub

Arquivo: `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`

Documenta o fluxo híbrido entre Lovable, GitHub, ambiente local, Codex e Claude, incluindo sincronização, contexto, conflitos e rollback AS-IS.

9. Manual 6 — Testes, QA, Debugging, Segurança e SEO

Arquivo: `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`

Documenta QA, testes funcionais, debugging, regressão, segurança e tratamento de SEO/AEO/GEO, preservando a separação entre prática atual e norma vigente.

10. Manual 7 — Publicação, Entrega e Pós-Publicação

Arquivo: `WSS_Manual_07_Publicacao_Entrega_Pos_Publicacao_v1.1_FINAL.md`

Documenta publicação temporária, domínio, produção, QA pós-publicação, contingência, entrega do Admin e continuidade operacional.

11. Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais

Arquivo: `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`

Concentra conhecimento acumulado, troubleshooting, casos reais, práticas recomendadas, práticas depreciadas e limites de generalização.

PARTE II — SOPs CANÔNICOS

12. SOP-001 — Contextualizar um Novo Projeto WSS

Arquivo: `WSS_SOP_001_Contextualizar_Novo_Projeto_WSS_v1.1_FINAL.md`

Transforma o handoff documental em entendimento operacional e prompt master suficientes para início técnico.

13. SOP-002 — Iniciar Tecnicamente um Projeto WSS: Base → Lovable → GitHub → Ambiente Local

Arquivo: `WSS_SOP_002_Iniciar_Tecnicamente_Projeto_WSS_v1.1_FINAL.md`

Prepara um novo projeto para o fluxo híbrido de desenvolvimento a partir da Base Admin.

14. SOP-003 — Construir e Refinar a Home a partir das Referências

Arquivo: `WSS_SOP_003_Construir_Refinar_Home_Referencias_v1.1_FINAL.md`

Consolida a Home como âncora visual do projeto por refinamento orientado por referências.

15. SOP-004 — Construir Páginas Internas Mantendo Consistência Visual

Arquivo: `WSS_SOP_004_Paginas_Internas_Consistencia_Visual_v1.1_FINAL.md`

Desenvolve as páginas internas preservando coerência com a linguagem visual estabelecida pela Home.

16. SOP-005 — Configurar Admin e Módulos Aplicáveis

Arquivo: `WSS_SOP_005_Configurar_Admin_Modulos_Aplicaveis_v1.1_FINAL.md`

Adapta a camada operacional da Base ao escopo e às necessidades do cliente.

17. SOP-006 — Configurar e Validar Chatbot e Base de Conhecimento

Arquivo: `WSS_SOP_006_Configurar_Validar_Chatbot_Base_Conhecimento_v1.1_FINAL.md`

Configura conhecimento, comportamento, integração com Leads/Admin e validação do Chatbot quando aplicável.

18. SOP-007 — Portar Capacidade da Base para Projeto Legado

Arquivo: `WSS_SOP_007_Portar_Capacidade_Base_Projeto_Legado_v1.1_FINAL.md`

Porta seletivamente capacidades atuais para projetos existentes, preservando o que o destino já faz corretamente.

19. SOP-008 — Executar QA Visual, Funcional e Responsivo

Arquivo: `WSS_SOP_008_QA_Visual_Funcional_Responsivo_v1.1_FINAL.md`

Executa a principal validação visual, responsiva e funcional do site e da camada operacional aplicável.

20. SOP-009 — Diagnosticar e Corrigir Bug por Camada

Arquivo: `WSS_SOP_009_Diagnosticar_Corrigir_Bug_Por_Camada_v1.1_FINAL.md`

Organiza debugging por sintoma, evidência, camada, hipótese, correção e regressão proporcional.

21. SOP-010 — Tratar Auditoria SEO, AEO e GEO

Arquivo: `WSS_SOP_010_Tratar_Auditoria_SEO_AEO_GEO_v1.1_FINAL.md`

Confronta auditorias com o estado atual, separa achado de delta real e aplica a camada normativa vigente.

22. SOP-011 — Processar Rodada de Feedback via `ajustes.md`

Arquivo: `WSS_SOP_011_Processar_Rodada_Feedback_Ajustes_md_v1.1_FINAL.md`

Consolida, executa, valida e encerra rodadas de feedback mantendo pendências e mudanças de escopo explicitamente controladas.

23. SOP-012 — Preparar, Publicar, Validar Produção e Entregar Admin

Arquivo: `WSS_SOP_012_Preparar_Publicar_Validar_Producao_Entregar_Admin_v1.1_FINAL.md`

Conduz o fechamento técnico da entrega, da prontidão de publicação ao QA em produção e acesso do cliente ao Admin.

PARTE III — MAPA DE PROPRIEDADE

24. Manual proprietário → SOPs relacionados

Manual proprietário	SOPs
Manual 1 — Visão Geral e Toolchain	Nenhum SOP proprietário; orientação transversal.
Manual 2 — Entrada e Contexto do Projeto	SOP-001
Manual 3 — Construção Visual	SOP-003, SOP-004
Manual 4 — Projeto-Base, Admin e Funcionalidades	SOP-005, SOP-006, SOP-007
Manual 5 — Desenvolvimento com Codex, Lovable e GitHub	SOP-002
Manual 6 — Testes, QA, Debugging, Segurança e SEO	SOP-008, SOP-009, SOP-010, SOP-011
Manual 7 — Publicação, Entrega e Pós-Publicação	SOP-012
Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais	Nenhum SOP proprietário; referência transversal de lições, troubleshooting e casos.

PARTE IV — FLUXO PONTA A PONTA

25. Mapa macro

```
Handoff documental WSS
        ↓
SOP-001 — Contextualizar projeto
        ↓
SOP-002 — Iniciar tecnicamente
        ↓
SOP-003 — Construir/refinar Home
        ↓
SOP-004 — Construir páginas internas
        ↓
SOP-005 — Configurar Admin/módulos
        ↓
SOP-006 — Chatbot/KB, quando aplicável
        ↓
SOP-008 — QA visual/funcional/responsivo
        ↓
SOP-009 — Debugging, quando houver bug
        ↓
SOP-010 — Auditoria SEO/AEO/GEO, quando aplicável
        ↓
SOP-011 — Feedback/ajustes, quando houver rodada
        ↓
SOP-012 — Publicar, validar produção e entregar Admin
        ↓
Manutenção / recorrência conforme escopo
```

Fluxo alternativo para projeto legado

```
Projeto existente
        ↓
SOP-007 — Portar capacidade da Base para legado
        ↓
QA / debugging / auditorias / publicação conforme necessidade
```

O fluxo é **macro e navegacional**, não uma sequência rígida. Há paralelismo real, especialmente entre revisão do institucional, finalização do Admin, feedback e validações específicas.

SOP-006, SOP-007, SOP-009, SOP-010 e SOP-011 são condicionais conforme projeto e estágio.

PARTE V — PROCESSO SITES WSS × MÉTODO DE DESENVOLVIMENTO

26. Processo Sites WSS

O **Processo Sites WSS** organiza marcos, status, condução, entregáveis, checklists, pendências, homologação, publicação e acompanhamento.

Ele responde principalmente:

> **o que precisa acontecer no projeto e em que estágio ele está?**

27. Método de Desenvolvimento de Sites WSS

O **Método de Desenvolvimento de Sites WSS** documenta como o Dev executa tecnicamente o trabalho, com quais ferramentas e procedimentos, da contextualização à publicação e entrega.

Ele responde principalmente:

> **como executar tecnicamente o desenvolvimento e a entrega do site?**

Regra de fronteira

O Método não substitui Roadmap, Checklists do Processo, status do Flow, critérios formais de marco ou homologações. O Processo, por sua vez, não substitui os procedimentos técnicos desta coleção.

PARTE VI — HIERARQUIA DOCUMENTAL

28. Hierarquia de uso

Nível 1 — Índice da Coleção

Entrada, navegação e localização do documento correto. Não redefine conteúdo.

Nível 2 — Normas técnicas vigentes

Estabelecem requisitos normativos especializados. Quando mais exigentes que uma prática histórica, prevalecem como requisito, sem apagar o registro AS-IS.

Nível 3 — Manuais `v1.1 FINAL`

Explicam contexto, princípios, fronteiras, decisões humanas, variantes, AS-IS, ausências de padronização e relação com normas.

Nível 4 — SOPs `v1.1 FINAL`

Executam procedimentos reproduzíveis derivados dos Manuais. O SOP não deve contradizer seu Manual proprietário.

Nível 5 — Baseline/evidências de descoberta

Incluem AS-IS consolidado, Matriz de Cobertura, Handoff, VALs, evidências de projetos e o Índice Mestre v0.1 DRAFT.

Servem à proveniência, rastreabilidade e histórico da consolidação; não são a porta de entrada operacional da coleção final.

PARTE VII — AUTORIDADE SEO/AEO/GEO

29. Norma canônica vigente

A autoridade técnica-operacional vigente para SEO, AEO e GEO é:

`WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`

O antigo `WSS_Padrao_Tecnico_Arquitetura_SEO_v0.1` permanece apenas como baseline histórico/proveniência e não concorre com a autoridade atual.

Os Manuais/SOPs podem apontar para anexos e SOPs especializados do padrão vigente quando o detalhe pertence àquela camada normativa.

30. AS-IS × norma

A coleção preserva quatro camadas:

- **AS-IS** — prática declarada/observada/comprovada;
- **AUSÊNCIA ATUAL DE PADRONIZAÇÃO** — ausência real de regra universal no AS-IS;
- **NORMA TÉCNICA VIGENTE** — requisito formal aprovado;
- **OPORTUNIDADE FUTURA — FORA DO ESCOPO** — evolução ainda não incorporada.

> **Norma vigente pode ser mais exigente que o AS-IS histórico, mas não reescreve retroativamente a história do método.**

PARTE VIII — SKILLS

31. Camada posterior e adicional

Skills **não integram o fechamento documental v1.1**.

A sequência aprovada é:

```
documentação humana
→ Manuais FINAL
→ SOPs FINAL
→ coleção canônica encerrada
→ eventual camada futura de Skills
```

As Skills poderão futuramente transformar partes estabilizadas do método em execução assistida por agentes de IA.

Elas são adicionais, derivadas da documentação final e não necessárias para considerar a documentação humana concluída.

A coleção v1.1 não presume quantidade de Skills, equivalência `1 SOP = 1 Skill` ou automação de decisões que continuam humanas.

PARTE IX — DECISÕES ESTRATÉGICAS ABERTAS

32. Itens não resolvidos por esta coleção

Permanecem abertas e fora da autoridade deste Índice:

1. **política padrão para crawlers de treinamento**, incluindo decisões sobre `GPTBot`, `Google-Extended` e equivalentes;
2. **seleção definitiva de ferramenta paga** para SEO/AEO/GEO/monitoramento;
3. **eventual obrigatoriedade universal futura do GA4**;
4. **fronteira entre entrega padrão × serviço recorrente**.

Esses pontos não devem receber default silencioso em Manuais, SOPs, projetos ou futuras Skills.

PARTE X — HISTÓRICO E STATUS

33. Histórico mínimo

Versão	Data	Estado	Registro
`v0.1 DRAFT` — Índice Mestre	03/09/2026	Descoberta	Mapa de cobertura da etapa de levantamento; permanece preservado como artefato histórico.
`v1.0 FINAL` — Índice da Coleção	10/09/2026	**FINAL — BASELINE HISTÓRICO**	Emitido após ratificação dos 8 Manuais `v1.0 FINAL` e dos 12 SOPs `v1.0 FINAL` ; preservado integralmente após a atualização v1.1.
`v1.1 FINAL` — Índice da Coleção	11/09/2026	**FINAL — CANÔNICO VIGENTE**	Incorpora o Pacote Corretivo Pós-Entrega decorrente da validação do Dev, sua normalização referencial de fechamento e a promoção dos 8 Manuais e 12 SOPs para `v1.1 FINAL`.

34. Estado final da documentação humana v1.1

```
8 Manuais v1.1 FINAL
+
12 SOPs v1.1 FINAL
+
Índice da Coleção v1.1 FINAL
=
Documentação humana principal do Método de Desenvolvimento de Sites WSS
materialmente encerrada e canônica
```

Status: `v1.1 FINAL — COLEÇÃO CANÔNICA VIGENTE`

WEB STAR STUDIO

MANUAL

MÉTODO DE DESENVOLVIMENTO DE SITES

MANUAL 1
Visão Geral e Toolchain

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

MANUAL

MANUAL 1

Visão Geral e Toolchain

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
ESCOPO DOCUMENTAL	primeira documentação ampliada — P0 + P1 + P2 relevante
PROJETO	WSS — Método de Desenvolvimento de Sites
FORMALIZAÇÃO	Pacote Corretivo Pós-Entrega v1.1 aprovado pela Coordenação em 11/09/2026; versão canônica vigente, com v1.0 FINAL preservado como baseline histórico.

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Objetivo 5

2. Escopo e fronteira 5

3. Como ler esta documentação 6

4. Processo Sites WSS × Método de Desenvolvimento de Sites WSS 7

5. Onde o trabalho do desenvolvedor começa 8

6. Onde o trabalho do desenvolvedor termina 9

7. Entradas principais 9

8. Saídas principais 10

9. Fluxo macro AS-IS 10

10. Papéis no fluxo 12

11. Tipos de projeto relevantes ao método 13

PARTE II — TOOLCHAIN WSS **15**

12. Visão geral do toolchain 16

13. ChatGPT pessoal 16

14. Lovable 17

15. GitHub 18

16. Codex 18

17. Claude 19

18. Supabase / Lovable Cloud 19

19. Navegador e DevTools 20

20. Ferramentas de referência visual 21

21. SEOmator 21

22. Flow e ferramentas operacionais adjacentes 22

PARTE III — CRITÉRIOS DE ESCOLHA DAS FERRAMENTAS **23**

23. Regra prática atual 24

24. Critérios observados de roteamento 24

25. Trabalho manual do Dev como camada do método 25

PARTE IV — GOVERNANÇA E LIMITES 27

26. Fontes de conhecimento do método 28

27. Decisões humanas que permanecem centrais 28

28. NORMA TÉCNICA VIGENTE 29

29. AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO relevantes para esta visão geral 29

30. Práticas atuais que explicam o desenho do toolchain 29

PARTE V — EXEMPLO REAL E NAVEGAÇÃO DA COLEÇÃO 31

31. Exemplo real resumido — Dr. Guilherme Mendonça 32

32. Relação com os demais Manuais 32

33. SOPs relacionados 33

34. Critério de conclusão deste Manual 33

35. Fontes principais desta versão 34

1. Objetivo

Este Manual apresenta a visão geral do **Método de Desenvolvimento de Sites WSS** e o ecossistema de ferramentas utilizado no fluxo atual.

Sua finalidade é permitir que um novo desenvolvedor, colaborador técnico ou agente de IA compreenda, antes de executar qualquer procedimento:

- o que o Método de Desenvolvimento de Sites WSS documenta;
- onde ele começa e termina;
- como ele se relaciona com o Processo Sites WSS;
- quais são as entradas e saídas principais;
- quem participa do fluxo;
- como o trabalho ocorre de ponta a ponta;
- para que cada ferramenta é usada hoje;
- como as ferramentas se relacionam;
- quais decisões permanecem humanas;
- onde não existe padronização formal no AS-IS.

Este Manual é deliberadamente **orientador**. Ele não substitui os Manuais específicos de entrada/contexto, construção visual, Base Admin, desenvolvimento com Codex/Lovable/GitHub, QA, publicação ou conhecimento acumulado.

---

2. Escopo e fronteira

2.1 O que entra neste Manual

Este Manual cobre:

1. conceito do Método de Desenvolvimento de Sites WSS;
2. fronteira entre Processo Sites WSS e Método de Desenvolvimento;
3. visão ponta a ponta do fluxo atual;
4. principais entradas e saídas;
5. papéis envolvidos;
6. tipos de projeto relevantes ao método;
7. visão funcional do toolchain:
    - ChatGPT pessoal;
    - Lovable;
    - GitHub;
    - ambiente local;
    - Codex;
    - Claude;

    - Supabase / Lovable Cloud;
    - navegador e DevTools;
    - Awwwards e fontes visuais auxiliares;
    - SEOmator;
    - ferramentas complementares;

8. critérios práticos de escolha entre ferramentas;
9. papel do julgamento humano;
10. ausências atuais de padronização relevantes para a visão geral.

2.2 O que não entra neste Manual

Ficam deliberadamente fora:

- procedimento detalhado de contextualização e criação do prompt master;
- comandos e rotinas de Git;
- clone, pull, commit e push em passo a passo;
- construção detalhada de Home e páginas internas;
- componentização e Design System em profundidade;
- configuração do Admin e de seus módulos;
- arquitetura detalhada do Chatbot;
- execução de QA;
- debugging;
- auditorias de SEO/AEO/GEO;
- validações de segurança;
- publicação, domínio e DNS em passo a passo;
- catálogo completo de troubleshooting;
- estudos de caso completos.

Esses assuntos pertencem aos demais Manuais e SOPs da coleção.

---

3. Como ler esta documentação

A documentação do Método WSS separa quatro camadas.

3.1 AS-IS

Descreve o que é efetivamente praticado hoje, declarado pelos responsáveis, observado em projetos ou tecnicamente comprovado.

É a camada principal deste Manual.

3.2 AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Registra situações em que hoje não existe regra, checklist, threshold, ritual ou procedimento formal único.

A ausência é um achado válido. Ela não deve ser silenciosamente preenchida com uma prática ideal.

3.3 NORMA TÉCNICA VIGENTE

Registra exigências aprovadas em documentos normativos próprios.

Uma norma vigente pode complementar o método, mas não deve ser apresentada como se sempre tivesse feito parte do AS-IS do desenvolvedor.

3.4 OPORTUNIDADE FUTURA — FORA DO ESCOPO

Registra, quando necessário, uma possível melhoria ou formalização futura.

Esta camada não deve ser usada para desenhar o TO-BE durante a documentação do método atual.

3.5 Convenções terminológicas da coleção

Para reduzir variações desnecessárias, esta coleção adota as seguintes convenções editoriais:

- **Base Admin WSS** — nome preferencial da base reutilizável; **Base** pode ser usada como forma curta depois de apresentada;
- **Admin** — forma curta de Admin do Site/painel administrativo quando o contexto já estiver claro;
- **Web Star Studio (WSS)** — nome institucional; `admin_webstar` e `admin_equipe` permanecem nomes técnicos quando a implementação exigir precisão;
- **perfil Equipe/cliente** — forma funcional para o perfil operacional do cliente; quando a implementação concreta for relevante, usar `admin_equipe`;
- **Chat/CTA** — capacidade persistente de interação/captação; WhatsApp permanece canal condicional, não sinônimo obrigatório de Chat;
- **Claude** — referência à família/ferramenta no fluxo; **Claude Code** deve ser usado quando a referência for especificamente ao ambiente de código/terminal.

Os nomes técnicos existentes em código, configuração ou interface não devem ser renomeados apenas para uniformização editorial.

---

4. Processo Sites WSS × Método de Desenvolvimento de Sites WSS

Os dois conceitos são relacionados, mas não são equivalentes.

4.1 Processo Sites WSS

O **Processo Sites WSS** organiza o projeto em termos de:

- marcos;
- condução;
- decisões;
- entregáveis;
- validações;
- pendências;
- acompanhamento interno;
- acompanhamento do cliente.

Ele responde principalmente:

> **o que precisa acontecer no projeto e em que estágio o projeto se encontra?**

4.2 Método de Desenvolvimento de Sites WSS

O **Método de Desenvolvimento de Sites WSS** documenta como o desenvolvedor transforma os insumos já consolidados em um site efetivamente implementado.

Ele responde principalmente:

> **como o desenvolvedor executa tecnicamente o trabalho, com quais ferramentas, quais decisões, quais verificações e quais caminhos alternativos?**

Na prática, o Método de Desenvolvimento se conecta principalmente aos marcos de execução, configuração, homologação e publicação do Processo Sites WSS.

O Processo não deve ser reescrito neste Manual. Quando um conceito de marco, entregável ou condução for necessário, a documentação do Processo Sites WSS deve ser tratada como referência própria.

---

5. Onde o trabalho do desenvolvedor começa

No fluxo atual, o Dev recebe da WSS um handoff documental que contém, como núcleo:

1. **Contexto de Desenvolvimento Consolidado**;
2. **Sitemap**;
3. **Briefing Final**;
4. arquivos `.md`, referências visuais e outros materiais do cliente, quando existirem.

Na intenção operacional da WSS, o Contexto de Desenvolvimento Consolidado é a principal fonte para execução.

Na prática atual do Dev, os documentos recebidos são consultados conjuntamente durante a contextualização inicial.

A partir desse handoff, a preparação técnica e a implementação passam a ser conduzidas pelo Dev.

O procedimento detalhado de leitura, organização, prompt master, `contexto.md` e `design.md` pertence ao **Manual 2 — Entrada e Contexto do Projeto**.

---

6. Onde o trabalho do desenvolvedor termina

A entrega técnica inicial é considerada concluída, no AS-IS, quando:

- os ajustes do cliente foram implementados;
- o site foi publicado/entregue;
- o Admin está funcional, validado e entregue.

Antes da entrega do Admin, os perfis internos já foram testados e os dados de teste são limpos.

Depois da entrega inicial, o projeto pode permanecer com o Dev para acompanhamento e manutenção pós-entrega.

Portanto, o método não termina necessariamente no primeiro publish. Ele inclui validação pós-publicação e entrega operacional do Admin.

---

7. Entradas principais

No nível macro, as entradas do método são:

- Contexto de Desenvolvimento Consolidado;
- Sitemap;
- Briefing Final;
- referências visuais validadas;
- identidade do cliente, quando definida;
- materiais e assets existentes;
- conteúdo e informações disponíveis;
- módulos/capacidades aplicáveis;
- decisões de kickoff;
- pendências ainda abertas;
- acessos e configurações necessários conforme o projeto.

Nem toda pendência precisa estar resolvida para toda atividade começar. A fronteira exata entre o que é obrigatório e o que pode permanecer pendente é tratada no Manual 2 e nos Manuais específicos.

---

8. Saídas principais

A execução pode produzir, conforme o escopo do projeto:

- site institucional implementado;
- Home e páginas internas;
- componentes e efeitos visuais;
- assets adaptados;
- Admin configurado;
- Blog configurado;
- formulários e captura de Leads;
- Chat/CTA;
- Chatbot, quando aplicável;
- perfis e acessos;
- ajustes decorrentes da revisão interna e do cliente;
- validações técnicas;
- site publicado;
- QA pós-publicação;
- acesso do cliente ao Admin;
- continuidade para manutenção ou recorrência.

Este Manual não transforma essa lista em checklist universal de conclusão. A aplicabilidade depende do projeto e do escopo.

---

9. Fluxo macro AS-IS

O fluxo consolidado atual pode ser representado assim:

```
Handoff documental da WSS
        ↓
Contextualização inicial pelo Dev
        ↓
ChatGPT pessoal / organização do contexto
        ↓
Prompt master
        ↓
Remix da Base Admin no Lovable
        ↓
Primeiros prompts e primeiros protótipos
        ↓
Conexão ao GitHub
        ↓
Clone para ambiente local
        ↓
Codex + Claude + Lovable
        ↓
Home
        ↓
Páginas internas
        ↓
Componentização / consistência / assets / conteúdo
        ↓
Efeitos / responsividade / performance
        ↓
QA interno do institucional
        ↓
Publicação temporária para validação do cliente
        ↓
Admin finalizado/configurado em paralelo
        ↓
Blog / Leads / Chat / Chatbot / perfis
        ↓
Feedback do cliente
        ↓
ajustes.md / rodadas de correção
        ↓
Auditorias e validações técnicas aplicáveis
        ↓
Validação final
        ↓
Domínio / produção
        ↓
QA pós-publicação
        ↓
Entrega do Admin
        ↓
Manutenção pós-entrega
```

Esse fluxo representa a jornada macro observada. Ele não deve ser interpretado como uma sequência rígida em que toda atividade só começa quando a anterior termina.

Há paralelismo real. Por exemplo, o institucional pode ser enviado para revisão do cliente enquanto o Dev continua configurando o Admin.

Para evitar ambiguidade, a coleção distingue três momentos de validação que podem se sobrepor conforme o projeto:

1. **QA do institucional** — validação visual, responsiva e funcional do site público antes ou durante a primeira revisão externa;

2. **QA geral/operacional** — revalidação do site junto com Admin e módulos aplicáveis à medida que a camada operacional amadurece;
3. **QA pós-publicação** — validação do comportamento no domínio/ambiente final.

A numeração dos Manuais e dos SOPs é **editorial e de navegação**, não uma imposição de execução totalmente linear.

---

10. Papéis no fluxo

10.1 Direção / Operações WSS

Atua principalmente em:

- consolidação e encaminhamento dos insumos;
- decisões operacionais;
- esclarecimento de escopo;
- resolução de dúvidas;
- validação de possíveis mudanças de escopo;
- coordenação do projeto.

Quando o Dev identifica uma dúvida relevante, uma contradição ou uma solicitação que pode representar nova funcionalidade, a gestão pode ser acionada antes da execução.

10.2 Desenvolvedor

É o principal responsável pela execução técnica.

No AS-IS, o Dev:

- contextualiza o projeto;
- prepara o caminho inicial;
- inicia o projeto no Lovable;
- conecta GitHub e ambiente local;
- usa Codex e Claude;
- constrói e refina o site;
- configura capacidades operacionais;
- testa;
- investiga falhas;
- executa ajustes;
- participa das auditorias técnicas;
- valida produção;
- entrega o Admin.

10.3 Analice e Felipe/CEO

No fluxo atual, concentram o acesso à GoDaddy e a etapa de domínio definitivo quando essa conta é necessária.

O Dev normalmente não executa essa configuração porque não possui esse acesso.

10.4 Cliente

Participa especialmente de:

- validação de referências;
- decisões de kickoff;
- revisão do site;
- envio de feedback;
- validação factual final de conteúdos sensíveis ou específicos;
- criação do próprio acesso ao Admin.

10.5 Ferramentas de IA

Codex, Claude, Lovable e ChatGPT são ferramentas de execução e apoio.

Elas não substituem a responsabilidade humana por:

- escopo;
- fidelidade;
- decisão visual;
- risco;
- validação;
- interpretação de auditoria;
- confirmação de fatos do cliente.

---

11. Tipos de projeto relevantes ao método

11.1 Projeto novo no fluxo atual

O fluxo corrente parte normalmente do **remix da Base Admin no Lovable**.

A Base fornece capacidades reutilizáveis, enquanto o institucional é adaptado ao cliente.

Iniciar do zero sem a Base Admin não representa a prática corrente preferencial.

11.2 Projeto antigo / legado

Projetos antigos podem possuir arquitetura, Admin ou capacidades diferentes da geração atual.

Nesses casos, a evolução não ocorre por substituição integral automática.

O padrão observado é de **portabilidade incremental e adaptativa de capacidades**, preservando o que já funciona melhor no destino.

O procedimento pertence ao Manual 4 e ao SOP específico de portabilidade.

11.3 Projeto já publicado

Após a entrega inicial, o projeto pode continuar sob acompanhamento/manutenção.

Correções, atualizações de capacidades, conteúdo e outras evoluções passam a ocorrer sobre um projeto que já possui estado funcional a preservar.

PARTE II — TOOLCHAIN WSS

12. Visão geral do toolchain

O método atual não depende de uma única ferramenta.

O fluxo combina ferramentas com papéis diferentes:

Ferramenta / ambiente	Papel predominante no AS-IS
ChatGPT pessoal	contextualização inicial e preparação do prompt master
Lovable	início do projeto, remix da Base Admin, primeiros protótipos e criação inicial da estrutura herdada de backend/banco/auth; no pós-criação, intervenções integradas podem ser executadas diretamente ou conduzidas pelo Codex
GitHub	ponte de versionamento e sincronização entre Lovable e ambiente local
Ambiente local	execução do projeto e base para trabalho direto com Codex/Claude
Codex	desenvolvimento direto, fidelidade visual, investigação, implementação pesada, testes e operação do Lovable em alguns casos
Claude	refinamento visual, tipografia, design, efeitos e acabamento
Supabase / Lovable Cloud	backend, banco, autenticação e recursos integrados conforme o projeto
Navegador / DevTools	validação visual/funcional e diagnóstico técnico
Awwwards	principal fonte de referências visuais
Fontes visuais auxiliares	componentes, efeitos, navegação e inspiração
SEOmator	auditoria externa SEO/GEO em casos aplicáveis
Flow	registro e contexto operacional adjacente ao desenvolvimento

A divisão não é absoluta. Uma mesma tarefa pode ser executada por mais de uma ferramenta.

---

13. ChatGPT pessoal

Uso atual

Antes de criar ou alterar o projeto, o Dev usa seu ChatGPT pessoal para:

- carregar documentos;
- resumir o projeto;
- organizar informações;
- compreender o caminho geral;
- avaliar por onde começar;

- preparar a instrução inicial.

Dessa etapa nasce o **prompt master**.

O Claude em terminal pode ser usado excepcionalmente nessa preparação quando o resultado obtido com ChatGPT não satisfaz.

Fronteira

Este Manual registra apenas a função do ChatGPT pessoal.

O procedimento de contextualização pertence ao Manual 2.

---

14. Lovable

Uso atual

O Lovable é o ponto normal de início do projeto.

É usado com frequência para:

- remix da Base Admin;
- primeiros prompts;
- primeiros protótipos;
- criação inicial da estrutura herdada de backend, banco e autenticação durante o remix da Base;
- tarefas pós-criação de backend, banco ou Supabase quando a integração da plataforma facilita, inclusive quando conduzidas pelo Codex;
- bugs funcionais;
- pequenos ajustes finais;
- operação pelo próprio Dev;
- operação indireta pelo Codex via navegador/chat em alguns casos.

Limite observado

O Dev tende a evitar o Lovable como ferramenta principal para reproduzir referências visuais complexas.

A prática antiga de desenvolver o projeto praticamente inteiro dentro do Lovable foi abandonada no fluxo atual.

Relação com outras ferramentas

O Lovable não funciona isoladamente no método atual.

Depois dos primeiros protótipos, o projeto é conectado ao GitHub e o desenvolvimento passa a ocorrer também no ambiente local.

---

15. GitHub

O GitHub funciona como ponte técnica entre o Lovable e o trabalho local.

Fluxo macro:

```
Lovable
   ↓
GitHub
   ↓
Projeto local
   ↓
Codex / Claude
   ↓
GitHub
   ↓
Lovable
```

No fluxo atual, o repositório é criado após os primeiros prompts/protótipos e clonado para a máquina do Dev.

A operação detalhada de branch, pull, commit, push, conflito e rollback pertence ao **Manual 5 — Desenvolvimento com Codex, Lovable e GitHub**.

---

16. Codex

Uso atual

O Codex ocupa posição central no fluxo corrente.

É usado com frequência para:

- reprodução visual com alta fidelidade;
- implementações mais pesadas;
- desenvolvimento direto no repositório;
- investigação de problemas;
- leitura e compreensão de projetos;
- análise de referências;
- análise e catalogação de imagens;
- geração contextualizada de imagens;
- testes amplos solicitados pelo Dev;
- operação do Lovable via navegador/chat em alguns casos.

Se o Dev precisasse escolher apenas uma ferramenta no fluxo atual, sua preferência declarada tenderia ao Codex.

Codex operando o Lovable

Quando o contexto já está concentrado no Codex e a tarefa faz mais sentido no Lovable, o Codex pode:

- abrir o Lovable;
- acessar o projeto;
- conversar no chat;
- acompanhar resposta e preview;
- conduzir a interação.

Essa modalidade não elimina a intervenção humana.

Pontos sensíveis como:

- credenciais;
- chaves de Supabase;
- tabelas;
- mudanças de risco;

continuam exigindo atenção e decisão humana.

---

17. Claude

O Claude é usado principalmente como ferramenta de refinamento.

Usos declarados:

- tipografia;
- design;
- efeitos;
- adaptação de interfaces;
- acabamento visual;
- versões mobile de composições e efeitos;
- refinamento de bases criadas pelo Codex.

O Claude não substitui o papel central do Lovable no início do projeto nem o papel de ponte do GitHub.

---

18. Supabase / Lovable Cloud

No fluxo atual, backend, banco, autenticação e capacidades operacionais podem envolver a infraestrutura integrada ao Lovable e ao Supabase. A criação inicial dessa estrutura herdada acontece no Lovable durante o remix da Base.

No pós-criação, tarefas de:

- backend;
- banco;

- autenticação;
- configuração relacionada;

podem ser investigadas/conduzidas pelo Codex e encaminhadas ou operadas no Lovable/Lovable Cloud quando a integração facilita ou é necessária. O uso direto do Lovable pelo Dev continua válido.

O ecossistema também aparece em diagnóstico: falhas de variável, key, conexão ou ambiente podem ser identificadas no console antes de qualquer tentativa de reescrever a funcionalidade afetada.

A arquitetura detalhada de cada módulo não pertence a este Manual.

---

19. Navegador e DevTools

Navegador

O navegador faz parte do trabalho manual e assistido por IA.

É usado para:

- navegar no site;
- comparar resultado e referência;
- testar interações;
- testar responsividade;
- verificar Admin;
- testar fluxos completos;
- validar produção.

DevTools / console

Não existe uma inspeção técnica rígida em DevTools para toda atividade.

O Dev recorre a console e ferramentas técnicas principalmente quando o sintoma sugere:

- configuração;
- key;
- ambiente;
- frontend;
- requests;
- erro de runtime.

O caso Mirela/Supabase é exemplo de diagnóstico em que o console ajudou a localizar a camada correta do problema.

---

20. Ferramentas de referência visual

20.1 Awwwards

É a principal fonte declarada de referências visuais.

A equipe normalmente seleciona entre **3 e 5 referências** modernas para o projeto, considerando UI, UX, design e desempenho percebido.

A referência principal e características de referências secundárias podem ser validadas com o cliente no kickoff.

20.2 Fontes auxiliares

Também aparecem no fluxo:

- Navbar Gallery;
- React Bits;
- 21st.dev;
- Framer;
- CodePen;
- X/Twitter;
- Instagram.

Essas fontes são usadas para encontrar:

- layouts;
- navegação;
- componentes;
- efeitos;
- animações;
- ideias visuais.

Elas não formam um catálogo obrigatório que precise ser usado em todos os projetos.

---

21. SEOmator

O SEOmator foi confirmado como ferramenta externa utilizada em auditoria SEO/GEO.

No fluxo observado, seu resultado é tratado como **entrada para investigação**, não como ordem automática de alteração.

Uma auditoria externa pode:

- apontar problema real;
- refletir limitação do crawler;
- estar baseada em versão anterior;
- exigir interpretação antes de qualquer edição.

O procedimento de triagem e tratamento da auditoria pertence ao Manual 6 e ao SOP correspondente.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não há evidência de uso obrigatório do SEOmator em toda entrega nem de score mínimo universal que funcione como gate WSS.

Heurística pessoal declarada pelo Dev — não normativa

Na prática pessoal do Dev, a faixa verde é buscada como referência; um score em torno de 90 é considerado bom e 95+ é uma meta desejada quando viável. Essa heurística não é regra WSS, não é requisito de entrega e não substitui a análise técnica dos achados.

---

22. Flow e ferramentas operacionais adjacentes

O Flow não é o ambiente principal de desenvolvimento do código.

Ele aparece como camada operacional adjacente, por exemplo em:

- registro de projeto;
- tarefas e acompanhamento interno;
- registros operacionais;
- armazenamento de credenciais internas conforme o fluxo atual.

A documentação completa do Processo Sites WSS/Flow permanece fora do escopo deste Manual.

PARTE III — CRITÉRIOS DE ESCOLHA DAS FERRAMENTAS

23. Regra prática atual

Não existe hoje uma matriz formal e obrigatória que determine ferramenta por tipo de tarefa.

A escolha é pragmática e depende de:

- natureza da tarefa;
- qualidade do resultado anterior;
- integração disponível;
- nível de fidelidade visual desejado;
- necessidade de trabalhar diretamente no código;
- necessidade de investigar;
- camada afetada;
- contexto já carregado em determinada IA;
- risco da alteração.

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe uma regra formal do tipo:

> “toda tarefa da categoria X deve obrigatoriamente ser executada pela ferramenta Y”.

O método atual depende de julgamento do Dev.

---

24. Critérios observados de roteamento

Preferência por Lovable

O Lovable tende a ser escolhido quando:

- o projeto está começando e a Base precisa ser remixada;
- a criação inicial da estrutura herdada de backend/banco/auth acontece no projeto remixado;
- no pós-criação, a integração com backend/banco/Supabase torna a execução no Lovable/Lovable Cloud mais adequada;
- o problema é funcional;
- há pequenos ajustes finais;
- a tarefa já faz mais sentido dentro do contexto do Lovable.

Preferência por Codex

O Codex tende a ser escolhido quando:

- é necessária maior fidelidade visual;
- a implementação é mais pesada;
- é necessário trabalhar diretamente no código;
- há necessidade de investigação;
- no pós-criação, é útil investigar ou conduzir uma intervenção de backend/banco/auth e, quando necessário, operar/orquestrar o Lovable;
- é preciso analisar uma referência;
- a tarefa exige testes ou inspeções mais amplas;
- o contexto já está concentrado no Codex.

Preferência por Claude

O Claude tende a ser escolhido quando:

- o problema é de refinamento visual;
- tipografia, composição ou acabamento precisam melhorar;
- um efeito precisa de adaptação;
- uma solução desktop precisa de leitura específica para mobile.

Preferência por abordagem manual/humana

A intervenção humana permanece central quando é necessário:

- julgar se ficou bom;
- comparar referência e resultado;
- avaliar fidelidade;
- decidir se uma adaptação faz sentido;
- proteger escopo;
- decidir diante de contradição;
- validar risco;
- impedir que IA altere áreas já aprovadas;
- confirmar informação sensível ou factual.

---

25. Trabalho manual do Dev como camada do método

O método atual não é um fluxo “IA executa e humano aceita”.

A camada manual é composta principalmente por:

- julgamento;
- navegação;
- comparação;
- cliques;

- testes;
- decisão de fidelidade;
- validação de comportamento;
- correção do entendimento da IA;
- controle de escopo;
- decisão de risco.

Essa camada é especialmente importante porque várias partes do método atual não possuem critérios formais completamente automatizáveis.

Exemplos:

- qualidade visual;
- percepção de performance;
- decisão de preservar ou descartar um efeito;
- escolha de qual ferramenta usar;
- definição de quando um resultado já está bom o suficiente para avançar.

PARTE IV — GOVERNANÇA E LIMITES

26. Fontes de conhecimento do método

A documentação atual foi consolidada a partir de diferentes tipos de fonte:

- Direção/Operações;
- Dev;
- demonstração prática;
- projetos reais;
- código;
- GitHub;
- Lovable;
- artefatos técnicos;
- validações dirigidas.

Uma peculiaridade encontrada em um projeto não deve ser convertida automaticamente em regra geral.

Quando necessário, os achados podem ser classificados como:

- **CONFIRMADO**;
- **OBSERVADO**;
- **INFERIDO**;
- **DEPRECIADO**;
- **PENDENTE**.

Essa classificação serve para governar a documentação, não para transformar o Manual em relatório arqueológico.

---

27. Decisões humanas que permanecem centrais

Mesmo com forte uso de IA, permanecem humanas decisões como:

- escopo;
- prioridade;
- mudança de escopo;
- direção visual;
- fidelidade;
- adequação da referência;
- risco de alteração;
- validação factual;
- interpretação de auditorias;
- aprovação para publicação;
- contingência diante de falha.

Nenhuma ferramenta deve ser tratada como autoridade absoluta apenas por produzir uma análise, um diff ou uma sugestão.

---

28. NORMA TÉCNICA VIGENTE

A autoridade canônica vigente para SEO, AEO e GEO é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md` — **padrão canônico vigente**.

O antigo `WSS_Padrao_Tecnico_Arquitetura_SEO_v0.1.md` está superado como autoridade autônoma e permanece apenas como baseline histórico/proveniência quando necessário.

As normas devem ser respeitadas quando aplicáveis independentemente de a implementação ocorrer por Lovable, Codex, Claude Code ou outra ferramenta.

Entretanto:

> a norma técnica não deve ser usada para reescrever retroativamente o AS-IS documentado.

Nos Manuais seguintes, sempre que uma norma complementar a prática atual, as duas camadas devem permanecer explicitamente diferenciadas.

29. AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO relevantes para esta visão geral

No nível macro, o AS-IS não possui:

- matriz formal obrigatória de escolha de ferramenta;
- automação que determine sozinha Lovable × Codex × Claude;
- ritual universal de trabalho idêntico para todos os tipos de projeto;
- uso obrigatório de todas as ferramentas auxiliares;
- score SEO/performance externo único como gate geral;
- eliminação completa do julgamento humano.

Essas ausências não impedem a reprodução do método, desde que o executor compreenda quais decisões exigem julgamento.

---

30. Práticas atuais que explicam o desenho do toolchain

Duas mudanças históricas ajudam a explicar o fluxo atual:

1. **desenvolver praticamente tudo apenas dentro do Lovable** deixou de ser a abordagem predominante;

2. **iniciar todo projeto do zero** foi substituído, no fluxo corrente, pelo uso da Base Admin como ponto de partida.

Por isso o toolchain atual é híbrido:

```
Lovable
+ GitHub
+ ambiente local
+ Codex
+ Claude
+ validação humana
```

O detalhamento das práticas depreciadas pertence ao Manual 8.

PARTE V — EXEMPLO REAL E NAVEGAÇÃO DA COLEÇÃO

31. Exemplo real resumido — Dr. Guilherme Mendonça

O projeto do Dr. Guilherme Mendonça é usado como principal caso representativo do fluxo atual.

Ele exemplifica:

- início por remix da Base Admin;
- primeiros passos no Lovable;
- integração com GitHub;
- trabalho local;
- uso combinado de Codex e Claude;
- construção visual;
- configuração do Admin;
- QA;
- auditorias de SEO/AEO/GEO;
- publicação e entrega.

Exclusão explícita

Em determinado momento houve um esboço antecipado antes de todas as definições do cliente estarem concluídas.

Essa ocorrência foi tratada como exceção operacional e **não integra o Método WSS**.

O estudo completo do caso pertence ao Manual 8.

---

32. Relação com os demais Manuais

Manual 2 — Entrada e Contexto do Projeto

Explica como transformar o handoff em contexto utilizável, prompt master e referências persistentes.

Manual 3 — Construção Visual

Explica referências, Home, páginas internas, componentização, Design System, conteúdo, assets, efeitos, responsividade, performance e acessibilidade AS-IS.

Manual 4 — Projeto-Base, Admin e Funcionalidades

Explica Base Admin, módulos, Blog, Leads, Chat/CTA, Chatbot, autenticação, perfis e portabilidade para projetos existentes.

Manual 5 — Desenvolvimento com Codex, Lovable e GitHub

Explica operacionalmente repositório, trabalho local, sincronização, modelos de uso do Codex/Lovable, conflitos e rollback.

Manual 6 — Testes, QA, Debugging, Segurança e SEO

Explica validação, regressão, troubleshooting, auditorias, SEO/AEO/GEO e segurança.

Manual 7 — Publicação, Entrega e Pós-Publicação

Explica domínio temporário, produção, QA pós-publicação, entrega do Admin e continuidade.

Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais

Concentra conhecimento acumulado, problemas conhecidos e estudos de caso.

---

33. SOPs relacionados

Este Manual não gera SOP próprio.

Ele funciona como pré-leitura e mapa para os SOPs da coleção, principalmente:

- SOP-001 — Contextualizar um Novo Projeto WSS;
- SOP-002 — Iniciar Tecnicamente um Projeto WSS: Base → Lovable → GitHub → Ambiente Local;
- SOP-003 — Construir e Refinar a Home a partir das Referências;
- SOP-005 — Configurar Admin e Módulos Aplicáveis;
- SOP-008 — Executar QA Visual, Funcional e Responsivo;
- SOP-012 — Preparar, Publicar, Validar Produção e Entregar Admin.

---

34. Critério de conclusão deste Manual

Este Manual cumpre seu objetivo quando o leitor consegue responder:

1. o que é o Método de Desenvolvimento de Sites WSS;
2. como ele se distingue do Processo Sites WSS;
3. onde o trabalho do Dev começa e termina;
4. qual é o fluxo macro atual;
5. quais são as principais ferramentas;
6. para que cada ferramenta tende a ser usada;
7. como elas se conectam;
8. quais decisões permanecem humanas;
9. quais ausências de padronização não devem ser preenchidas com um processo ideal;
10. em qual Manual/SOP procurar o procedimento detalhado.

---

35. Fontes principais desta versão

Baseline canônico:

1. `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
2. `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
3. `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
4. `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
5. `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

Contexto adicional utilizado apenas na fronteira do Processo:

- `Roadmap_Sites_WSS_Modelo_Padrao_v0.2_FINAL.md`.

Normas técnicas são referências próprias e devem permanecer separadas do relato AS-IS.

Autoridade normativa SEO/AEO/GEO vigente:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Status: FINAL — APROVADO

WEB STAR STUDIO

MANUAL

MÉTODO DE DESENVOLVIMENTO DE SITES

MANUAL 2
Entrada e Contexto do Projeto

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

MANUAL

MANUAL 2

Entrada e Contexto do Projeto

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
ESCOPO DOCUMENTAL	primeira documentação ampliada — P0 + P1 + P2 relevante
PROJETO	WSS — Método de Desenvolvimento de Sites
FORMALIZAÇÃO	Pacote Corretivo Pós-Entrega v1.1 aprovado pela Coordenação em 11/09/2026; versão canônica vigente, com v1.0 FINAL preservado como baseline histórico.

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Objetivo 6

2. Escopo e fronteira 6

3. Posição deste Manual no fluxo WSS 7

PARTE I — HANDOFF WSS → DEV **9**

4. Momento de entrada do desenvolvedor 10

5. Contexto de Desenvolvimento Consolidado 10

6. Sitemap 11

7. Briefing Final 11

8. Referências visuais e materiais adicionais 12

PARTE II — HIERARQUIA REAL DAS FONTES **13**

9. Três comportamentos coexistentes no AS-IS 14

10. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — hierarquia documental universal 14

11. OPORTUNIDADE FUTURA — FORA DO ESCOPO 15

PARTE III — CONTEXTUALIZAÇÃO INICIAL **16**

12. ChatGPT pessoal como primeira camada de organização 17

13. Resultado esperado da contextualização inicial 17

14. Prompt master 18

15. Do prompt master para o Lovable 18

PARTE IV — CONTEXTO PERSISTENTE E CONTEXTO DA TAREFA **19**

16. Duas camadas práticas de contexto 20

17. contexto.md 20

18. design.md 21

19. Commits e histórico como apoio de contexto 21

PARTE V — TROCA DE CHAT OU FERRAMENTA **22**

20. Entrada em nova sessão 23

21. Codex 23

22. Claude 23

23. Lovable 23

PARTE VI — COMO O DEV ESTRUTURA PROMPTS DEPOIS DO PROMPT MASTER 25

24. Regra prática principal: limitar o escopo 26

25. Criar ou alterar uma seção 26

26. Corrigir um problema visual 26

27. Corrigir um bug funcional 27

28. Investigar antes de editar 27

29. Executar testes 27

30. Comparar com referência 27

PARTE VII — PROTEÇÃO DE ESCOPO 29

31. Preservar áreas já validadas 30

32. Anti-pattern de prompting 30

PARTE VIII — VALIDAÇÃO DE ENTENDIMENTO 31

33. Resumo antes de continuar 32

34. Sinais de que a IA saiu do contexto 32

35. O que o Dev faz quando há desvio 32

36. Indicador de effort/info 33

PARTE IX — PENDÊNCIAS E INFORMAÇÃO INCOMPLETA 34

37. Nem toda pendência bloqueia o início 35

38. Placeholders 35

39. Arquivo .md de pendências 35

40. Informações factuais e sensíveis 35

PARTE X — EXEMPLO REAL DE CONTEXTO DE DESENVOLVIMENTO 37

41. Dr. Guilherme Mendonça 38

PARTE XI — NORMA, AUSÊNCIAS E OPORTUNIDADES 40

42. NORMA TÉCNICA VIGENTE 41

43. AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 41

44. OPORTUNIDADES FUTURAS — FORA DO ESCOPO 41

PARTE XII — RELAÇÃO COM OS OUTROS MANUAIS E SOP 43

45. Relação com o Manual 3 — Construção Visual 44

46. Relação com o Manual 4 — Projeto-Base, Admin e Funcionalidades 44

47. Relação com o Manual 5 — Codex, Lovable e GitHub 44
48. Relação com o Manual 6 — QA, Debugging, Segurança e SEO 44
49. SOP relacionado 45
50. Critério de conclusão deste Manual 45
51. Fontes principais desta versão 46

1. Objetivo

Este Manual documenta como o projeto passa da WSS para o desenvolvedor e como os materiais recebidos são transformados em **contexto operacional utilizável pelas ferramentas de IA e pelo desenvolvimento**.

Seu foco é a passagem:

```
WSS
→ handoff documental
→ leitura e organização pelo Dev
→ contextualização
→ prompt master
→ contexto persistente no projeto
→ validação de entendimento
→ início técnico do projeto
```

Ao final deste Manual, o leitor deve compreender:

- quais materiais chegam ao Dev;
- como eles são usados hoje;
- qual é a relação real entre Contexto de Desenvolvimento Consolidado, Sitemap e Briefing Final;
- como o Dev organiza o projeto inicialmente no ChatGPT pessoal;
- qual é o papel do prompt master;
- como `contexto.md` e `design.md` funcionam;
- como o contexto é preservado entre Lovable, Codex e Claude;
- como o Dev limita o escopo dos prompts;
- como identifica que uma IA perdeu o contexto ou inventou requisitos;
- como lida com informações ainda pendentes;
- quais decisões e validações continuam humanas.

Este Manual registra o **AS-IS**. Ele não cria um novo modelo de handoff, uma nova hierarquia documental ou um template ideal de prompt.

---

2. Escopo e fronteira

2.1 O que entra neste Manual

Este Manual cobre:

1. momento de entrada do Dev;
2. materiais recebidos da WSS;
3. Contexto de Desenvolvimento Consolidado;
4. Sitemap;
5. Briefing Final;

6. referências visuais e materiais complementares;
7. relação prática entre essas fontes;
8. contextualização inicial com ChatGPT pessoal;
9. preparação do prompt master;
10. envio de contexto ao Lovable;
11. contexto persistente no projeto;
12. `contexto.md`;
13. `design.md`;
14. reconstrução de contexto em Codex e Claude;
15. escopo de prompts;
16. sinais de perda de contexto;
17. tratamento de requisitos inventados;
18. informações pendentes;
19. placeholders e arquivos `.md`;
20. validação factual e informações sensíveis.

2.2 O que não entra neste Manual

Ficam fora:

- remix da Base Admin em passo a passo;
- criação e conexão do repositório GitHub;
- clone e comandos Git;
- execução técnica de Codex/Lovable/Claude;
- construção detalhada da Home;
- análise detalhada de referências;
- configuração de Admin e módulos;
- QA;
- SEO/AEO/GEO;
- publicação.

Esses assuntos pertencem aos Manuais seguintes.

---

3. Posição deste Manual no fluxo WSS

Este Manual cobre o intervalo entre:

```
decisões do projeto já consolidadas pela WSS
                ↓
        handoff ao Dev
                ↓
     entendimento operacional
                ↓
          prompt master
                ↓
 contexto suficiente para começar
```

O próximo estágio técnico é tratado principalmente no:

Manual 5 — Desenvolvimento com Codex, Lovable e GitHub

A construção visual decorrente desse contexto pertence ao:

Manual 3 — Construção Visual

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE I — HANDOFF WSS → DEV

4. Momento de entrada do desenvolvedor

No fluxo atual, o Dev entra após receber da WSS o conjunto de informações necessário para iniciar a preparação técnica.

O handoff atual inclui, como núcleo:

1. **Contexto de Desenvolvimento Consolidado**;
2. **Sitemap**;
3. **Briefing Final**;
4. arquivos `.md` com informações do cliente, quando existirem;
5. referências visuais;
6. materiais complementares disponíveis.

A partir desse envio, o Dev assume a preparação técnica e começa a organizar o contexto para as ferramentas que usará no desenvolvimento.

---

5. Contexto de Desenvolvimento Consolidado

Na intenção operacional atual da WSS, o **Contexto de Desenvolvimento Consolidado** é o principal artefato de entrada para o desenvolvimento.

Sua finalidade é concentrar em uma visão operacional informações que já foram decididas ou organizadas anteriormente no Processo Sites WSS.

Ele pode reunir, conforme o projeto:

- objetivo do site;
- posicionamento;
- público;
- direção visual;
- referências;
- identidade;
- sitemap;
- conteúdos;
- módulos aplicáveis;
- CTAs;
- restrições;
- pendências;
- diretrizes técnicas relacionadas.

O Contexto Consolidado é concebido para reduzir a necessidade de o Dev reconstruir todo o histórico do projeto a partir dos documentos do Flow.

> **Importante**
>
> No AS-IS, o Contexto Consolidado **não é a única fonte consultada pelo Dev**.

Ele recebe e consulta também Sitemap, Briefing Final e demais materiais.

---

6. Sitemap

O Sitemap chega ao Dev como fonte própria de consulta e validação.

Ele orienta principalmente:

- cobertura de páginas;
- hierarquia pública;
- páginas prioritárias;
- páginas individuais;
- organização geral de navegação;
- escopo estrutural que deverá aparecer no site.

Na construção posterior, o Sitemap continua funcionando como guia de cobertura das páginas internas.

Este Manual não transforma o Sitemap em especificação técnica de rotas. A implementação pertence aos Manuais de construção e desenvolvimento.

---

7. Briefing Final

O Briefing Final também é enviado ao Dev como fonte de consulta.

Ele sustenta especialmente:

- posicionamento;
- informações do cliente;
- escopo;
- tom;
- objetivos;
- decisões de negócio;
- conteúdos e fatos validados;
- restrições;
- pontos levantados no kickoff.

Na prática atual, o Dev não ignora o Briefing apenas porque existe um Contexto Consolidado.

---

8. Referências visuais e materiais adicionais

O Dev também pode receber:

- referência visual principal;
- referências secundárias;
- screenshots;
- arquivos `.md`;
- fotos;
- links;
- documentos adicionais;
- conteúdos do cliente;
- outras evidências úteis para o projeto.

A direção visual já tende a chegar consolidada quando o desenvolvimento começa.

O processo de analisar e traduzir essas referências pertence ao Manual 3.

PARTE II — HIERARQUIA REAL DAS FONTES

9. Três comportamentos coexistentes no AS-IS

A descoberta encontrou três formas relacionadas, mas não idênticas, de tratar autoridade documental.

9.1 Intenção operacional da WSS

A WSS concebe o **Contexto de Desenvolvimento Consolidado** como principal artefato de execução.

A intenção é que ele seja suficientemente completo para orientar o desenvolvimento sem exigir que o Dev reconstrua todo o histórico do Processo Sites WSS.

9.2 Prática declarada pelo Dev

Na prática, o Dev carrega e consulta conjuntamente:

- Contexto Consolidado;
- Sitemap;
- Briefing Final;
- arquivos `.md`;
- referências;
- demais materiais recebidos.

Portanto, a execução atual não se baseia em leitura exclusiva de uma única fonte.

9.3 Caso real — Dr. Guilherme Mendonça

O arquivo:

`Dr_Guilherme_Mendonca_Contexto_Desenvolvimento_v1.0_FINAL.md`

foi criado como documento operacional de entrada, mas registra explicitamente que **não substitui as fontes canônicas do projeto**.

Nesse caso específico, o próprio documento define uma ordem de precedência que começa pelo Briefing Curado Pós-Kickoff, seguido pelo Sitemap Validado e por outros documentos.

Isso demonstra que um Contexto Consolidado pode carregar **uma regra de precedência própria do projeto**.

---

10. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — hierarquia documental universal

Não existe hoje, no Método de Desenvolvimento documentado, uma regra universal única e formalmente estabilizada dizendo:

```
Contexto Consolidado sempre vence Sitemap
```

ou:

```
Briefing sempre vence Contexto Consolidado
```

O que está sustentado é:

- a WSS pretende que o Contexto Consolidado seja o principal artefato de entrada;
- o Dev consulta as fontes em conjunto;
- um projeto real pode possuir ordem de precedência explicitamente definida dentro do próprio pacote.

Consequência documental

Este Manual **não escolhe** uma nova hierarquia.

Quando um projeto ou artefato trouxer uma ordem de precedência explícita, essa regra deve ser respeitada como parte do contexto daquele projeto.

Quando não houver regra explícita e surgir uma contradição material, a divergência deve ser tratada como decisão humana, não resolvida por invenção da IA.

---

11. OPORTUNIDADE FUTURA — FORA DO ESCOPO

O baseline de descoberta já registrou como oportunidade futura:

> alinhar formalmente a hierarquia entre Contexto Consolidado, Sitemap e Briefing Final.

Este Manual apenas preserva esse achado.

Não é objetivo desta versão desenhar essa hierarquia futura.

PARTE III — CONTEXTUALIZAÇÃO INICIAL

12. ChatGPT pessoal como primeira camada de organização

Antes de criar ou alterar o projeto, o Dev usa normalmente seu **ChatGPT pessoal**.

Nessa etapa ele:

- carrega documentos e referências;
- resume o projeto;
- organiza informações;
- procura compreender o caminho geral;
- avalia por onde começar;
- prepara a instrução inicial do projeto.

O objetivo não é desenvolver o site dentro dessa conversa.

O ChatGPT pessoal funciona como uma camada de **organização e preparação do contexto**.

O Claude no terminal pode ser usado excepcionalmente nessa fase quando o resultado obtido com ChatGPT não satisfaz.

---

13. Resultado esperado da contextualização inicial

A contextualização inicial deve deixar o Dev com entendimento suficiente sobre:

- quem é o cliente;
- objetivo do projeto;
- posicionamento;
- público;
- sitemap;
- direção visual;
- referências;
- conteúdos relevantes;
- restrições;
- módulos/capacidades aplicáveis;
- pendências;
- riscos ou pontos de atenção;
- caminho inicial de desenvolvimento.

No AS-IS, esse entendimento é consolidado pelo Dev com apoio da IA antes da implementação principal.

---

14. Prompt master

Dessa etapa nasce o **prompt master**.

No fluxo atual, ele funciona como a instrução inicial que leva o contexto do projeto para o início da implementação, principalmente no Lovable.

O prompt master é preparado depois que o Dev:

- leu os materiais;
- organizou as informações;
- formou uma visão geral;
- decidiu por onde começar.

O que este Manual não deve inventar

A descoberta não consolidou um **template canônico único** de prompt master para todos os projetos.

Portanto, esta documentação não cria uma estrutura fixa artificial.

O que está suficientemente sustentado é sua função:

> transformar o entendimento acumulado do projeto em uma instrução inicial suficientemente contextualizada para começar o desenvolvimento.

---

15. Do prompt master para o Lovable

No fluxo atual, o projeto começa no Lovable.

O Dev normalmente:

1. remixa a Base Admin;
2. envia o prompt master;
3. fornece documentos e referências;
4. usa os primeiros prompts para contextualizar o projeto;
5. gera a primeira base do institucional;
6. chega ao primeiro ou segundo protótipo.

A partir daí o projeto é conectado ao GitHub e segue também para o ambiente local.

O procedimento técnico dessa transição pertence ao Manual 5.

PARTE IV — CONTEXTO PERSISTENTE E CONTEXTO DA TAREFA

16. Duas camadas práticas de contexto

No fluxo atual é útil distinguir duas camadas observadas.

16.1 Contexto persistente do projeto

Fica principalmente no próprio projeto e em seus artefatos.

Exemplos:

- código existente;
- `contexto.md`;
- `design.md`;
- outros arquivos `.md`;
- commits;
- histórico disponível;
- documentos do projeto.

Essa camada permite que uma nova sessão reconstrua a história e a lógica geral.

16.2 Contexto específico da tarefa

É reforçado no prompt atual.

Pode incluir:

- tela;
- seção;
- módulo;
- bug;
- breakpoint;
- screenshot;
- link de referência;
- restrições;
- áreas que não devem ser alteradas;
- objetivo específico daquela rodada.

O contexto persistente reduz a necessidade de repetir tudo; o contexto da tarefa limita o escopo da execução atual.

---

17. `contexto.md`

O Dev usa `contexto.md` como arquivo persistente de consolidação do desenvolvimento.

Ele pode ajudar ferramentas futuras a entender:

- projeto;

- decisões;
- estado;
- contexto acumulado.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe hoje uma regra de atualizar `contexto.md` em toda troca de ferramenta, a cada prompt ou após cada alteração.

Ele funciona como referência persistente e é reforçado quando necessário.

Este Manual não cria uma cadência obrigatória nova.

---

18. `design.md`

O `design.md` é uma referência persistente especialmente importante para a direção visual.

O Dev declarou que Codex e Claude usam esse arquivo para preservar, entre páginas e chats:

- tipografia;
- espaçamentos;
- cores;
- referências;
- linguagem visual;
- padrões gerais de design.

Quando a ferramenta começa a sair da direção visual, o Dev reforça novamente o `design.md`.

O detalhamento de como essa direção se materializa no código pertence ao Manual 3.

---

19. Commits e histórico como apoio de contexto

O Dev também usa o próprio estado do projeto, commits e histórico disponível como sinais de continuidade.

Ao trocar de ferramenta, a IA pode reconstruir parte do contexto a partir de:

- repositório;
- arquivos;
- commits;
- histórico acessível no ambiente.

Neste Manual, esses elementos são tratados apenas como **fontes de contexto**.

A operação Git pertence ao Manual 5.

PARTE V — TROCA DE CHAT OU FERRAMENTA

20. Entrada em nova sessão

Ao iniciar uma nova sessão com Codex ou Claude, o Dev costuma pedir que a ferramenta:

- leia o projeto;
- entenda a estrutura;
- leia ou releia os documentos;
- entenda quem é WSS;
- entenda quem é o Dev;
- entenda quem é o cliente;
- apresente um resumo antes de continuar.

Quando um arquivo não está acessível à ferramenta, ele é enviado novamente.

Essa prática reduz a chance de começar uma alteração com uma compreensão incompleta do projeto.

---

21. Codex

Depois do clone, o Codex consegue ler o repositório e os documentos disponíveis.

No fluxo do Dev, ele também pode reconstruir contexto a partir do que foi trabalhado anteriormente no projeto e, em algumas situações, operar o Lovable pelo navegador.

O uso operacional detalhado pertence ao Manual 5.

---

22. Claude

Ao passar do Codex para o Claude, o Dev espera que o Claude reconstrua o contexto por:

- projeto;
- arquivos `.md`;
- commits;
- histórico disponível no ambiente.

O Claude é usado com frequência para refinamento visual, o que torna o `design.md` particularmente relevante nessa transição.

---

23. Lovable

O Lovable é a ferramenta em que o Dev percebe maior chance de perda de contexto quando retorna depois de trabalhar fora dele.

Mesmo assim, essa perda não foi descrita como problema recorrente ou bloqueante no fluxo atual.

Quando necessário, o Dev reforça novamente:

- documentos;
- regras;
- objetivo;
- contexto do projeto;
- escopo da tarefa.

PARTE VI — COMO O DEV ESTRUTURA PROMPTS DEPOIS DO PROMPT MASTER

24. Regra prática principal: limitar o escopo

O Dev procura cercar a tarefa no próprio prompt.

Em vez de pedir alterações muito amplas, normalmente delimita:

- uma tela;
- uma seção;
- um módulo;
- um problema;
- um breakpoint;
- uma funcionalidade.

Essa limitação é uma das práticas atuais mais importantes para reduzir desvios da IA.

---

25. Criar ou alterar uma seção

Quando trabalha apenas uma parte da interface, o Dev especifica o recorte exato.

Exemplo de comportamento atual:

- se o foco é mobile, declara que desktop já está validado e não deve mudar;
- se o foco é desktop, evita que a ferramenta altere mobile sem necessidade;
- reforça `design.md` quando a direção visual precisa permanecer intacta.

O conteúdo visual detalhado desses prompts pertence ao Manual 3.

---

26. Corrigir um problema visual

O Dev tende a fornecer evidência concreta:

- print;
- link;
- referência;
- efeito específico.

Quando existe comportamento complexo, pode pedir à ferramenta que **estude a referência antes de editar**.

A abordagem genérica de apenas mandar um site e pedir “copie com fidelidade” é considerada fraca no AS-IS.

---

27. Corrigir um bug funcional

O Dev delimita o problema e pede investigação focada.

Para bugs funcionais, usa com frequência:

- Lovable;
- Codex.

Em problemas difíceis, pode comparar as investigações antes de seguir.

O método de debugging pertence ao Manual 6.

---

28. Investigar antes de editar

Quando o resultado depende de referência, efeito ou solução não trivial, o Dev pode pedir primeiro:

- pesquisa;
- leitura;
- comparação;
- avaliação de alternativa;
- estudo do efeito.

Isso evita transformar a primeira hipótese da IA em implementação imediata.

---

29. Executar testes

Não há um template rígido de prompt de testes no AS-IS.

O Dev delimita:

- área;
- funcionalidade;
- fluxo;

e procura impedir que a ferramenta expanda o trabalho para partes já aprovadas.

A estratégia completa de testes pertence ao Manual 6.

---

30. Comparar com referência

A prática visual atual trabalha por partes.

Em vez de pedir fidelidade ao site inteiro de uma vez:

1. escolhe seção;

2. fornece print;
3. fornece link;
4. pede estudo da referência;
5. compara resultado;
6. refina;
7. depois adapta ao projeto.

Esse procedimento pertence principalmente ao Manual 3 e aos SOPs de Home/páginas.

PARTE VII — PROTEÇÃO DE ESCOPO

31. Preservar áreas já validadas

O Dev usa instruções explícitas para reduzir alterações colaterais.

Exemplos declarados:

- mobile não deve alterar desktop já validado;
- desktop não deve quebrar mobile validado;
- frontend não deve provocar alteração desnecessária no backend;
- backend não deve gerar mudança de interface sem necessidade;
- trabalho no Admin não deve alterar o institucional;
- um módulo do Admin não deve quebrar módulos já validados.

Formulação AS-IS

A proteção do escopo depende fortemente da **clareza do prompt e da validação humana posterior**.

Não existe uma trava técnica universal que impeça a IA de tocar outra área.

---

32. Anti-pattern de prompting

Um padrão declarado como problemático é:

> fornecer apenas o link de uma referência e pedir para a ferramenta ser “extremamente fiel”.

O problema é o excesso de liberdade de interpretação.

A abordagem que tende a funcionar melhor é:

- reduzir escopo;
- dar evidência concreta;
- pedir estudo;
- trabalhar em etapas;
- preservar áreas já aprovadas.

O catálogo completo de anti-patterns pertence ao Manual 8.

PARTE VIII — VALIDAÇÃO DE ENTENDIMENTO

33. Resumo antes de continuar

Uma prática usada pelo Dev ao entrar em nova sessão é pedir que a ferramenta **resuma o projeto antes de executar**.

Esse resumo funciona como verificação prática de que ela:

- leu os arquivos;
- entendeu o cliente;
- reconheceu o contexto;
- sabe a área que será trabalhada.

Não existe um formulário formal universal de aceite do resumo.

A validação é feita pelo julgamento do Dev.

---

34. Sinais de que a IA saiu do contexto

O Dev identifica desvio quando a ferramenta:

- deixa de seguir o design explicado;
- adiciona textos aleatórios sem relação com o projeto;
- cria requisito ou elemento que nunca foi pedido;
- começa a alterar áreas fora do escopo;
- ignora regras já explicadas;
- quebra uma área previamente validada.

Esses sinais valem mais, no AS-IS, do que uma checagem documental automática formal.

---

35. O que o Dev faz quando há desvio

Quando percebe perda de contexto ou invenção:

1. interrompe a execução;
2. explica que a ferramenta saiu do contexto;
3. reforça documentos e regras relevantes;
4. reduz novamente o escopo;
5. divide a tarefa em partes menores;
6. avança gradualmente.

Em vez de tentar corrigir todo o desvio em um único prompt, o Dev prefere reconduzir a ferramenta em etapas.

---

36. Indicador de effort/info

O Dev também declarou observar o indicador de `effort/info` da ferramenta como um sinal prático.

Na sua experiência:

- effort baixo pode coincidir com maior perda de contexto;
- effort mais alto tende a preservar melhor o entendimento.

> **Limite**
>
> Isso é uma **heurística pessoal declarada**, não uma regra formal do Método WSS nem uma relação técnica comprovada.

Não deve ser transformado em threshold obrigatório.

PARTE IX — PENDÊNCIAS E INFORMAÇÃO INCOMPLETA

37. Nem toda pendência bloqueia o início

O AS-IS permite que certas informações permaneçam pendentes enquanto o desenvolvimento avança.

Um caso real documentado, Dr. Guilherme Mendonça, instrui que pendências do cliente sejam tratadas como **não bloqueantes**, salvo quando forem indispensáveis para publicar determinado conteúdo factual.

Essa formulação pertence ao caso e não deve ser automaticamente universalizada para todos os projetos.

No método geral, a decisão depende do tipo de pendência e do impacto da informação ausente.

---

38. Placeholders

Quando falta uma informação operacional simples, o Dev pode trabalhar com placeholder.

Exemplo declarado:

- telefone pendente → número provisório no formato `(81) 9 9999-9999`.

O uso de placeholder não significa que a informação foi validada.

Antes da entrega/publicação aplicável, o dado precisa ser substituído ou tratado conforme o caso.

---

39. Arquivo `.md` de pendências

Para outros conteúdos ainda não recebidos, o Dev registra pendências em arquivo `.md`.

Esse arquivo funciona como memória operacional.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> O próprio Dev reconhece que esse controle depende bastante de acompanhamento humano e existe risco de um item ser esquecido.

Não há hoje, no AS-IS, mecanismo formal universal que garanta automaticamente a resolução de toda pendência registrada em `.md`.

---

40. Informações factuais e sensíveis

O Dev evita inventar ou modificar com IA, sem validação:

- formação;
- títulos;
- procedimentos;

- resultados;
- especialidades;
- afirmações médicas;
- outros fatos sensíveis do cliente.

Quando o conteúdo é criado ou adaptado por IA, a validação factual final fica principalmente com o cliente.

Essa prática não equivale a uma política formal universal de revisão clínica/editorial em todos os projetos.

PARTE X — EXEMPLO REAL DE CONTEXTO DE DESENVOLVIMENTO

41. Dr. Guilherme Mendonça

O arquivo:

`Dr_Guilherme_Mendonca_Contexto_Desenvolvimento_v1.0_FINAL.md`

é um exemplo concreto de documento de entrada para desenvolvimento.

Ele reúne em um único artefato operacional, entre outros elementos:

- fontes canônicas e precedência;
- resumo do projeto;
- público;
- princípios de UX;
- direção visual;
- referências;
- paleta;
- posicionamento;
- linguagem;
- arquitetura pública;
- estrutura das páginas;
- Blog;
- SEO/AEO/GEO;
- animações/performance;
- captação e CTAs;
- temas específicos;
- conteúdo provisório;
- pendências.

41.1 O que este caso demonstra

O caso mostra que um Contexto de Desenvolvimento pode funcionar como **ponte operacional** entre decisões consolidadas e implementação.

Também mostra que o próprio contexto pode carregar:

- limites;
- regras;
- fontes;
- pendências;
- placeholders;
- referências normativas.

41.2 O que não deve ser generalizado

A estrutura exata do arquivo do Dr. Guilherme não deve ser tratada automaticamente como template universal do Método WSS.

O caso é evidência real de um handoff, não definição de um novo padrão documental.

PARTE XI — NORMA, AUSÊNCIAS E OPORTUNIDADES

42. NORMA TÉCNICA VIGENTE

Documentos normativos técnicos podem chegar ao desenvolvimento ou ser referenciados pelo Contexto Consolidado.

Para SEO/AEO/GEO, a autoridade atual é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md` — **padrão canônico vigente**.

O antigo `WSS_Padrao_Tecnico_Arquitetura_SEO_v0.1.md` permanece apenas como baseline histórico/proveniência, não como autoridade concorrente.

Quando uma norma estiver explicitamente aplicável, ela deve ser preservada durante o desenvolvimento.

Entretanto:

> uma norma técnica não deve ser apresentada como descrição do que o Dev sempre praticou historicamente.

A camada normativa permanece distinta do AS-IS.

43. AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

Este Manual registra como ausências reais:

- não existe hierarquia universal única estabilizada entre Contexto Consolidado, Sitemap e Briefing Final;
- não existe obrigação de atualizar `contexto.md` a cada troca de ferramenta;
- não existe obrigação de atualizar `design.md` a cada troca de ferramenta;
- não existe validação formal automatizada de que a IA “entendeu” o projeto;
- a validação de entendimento depende do resumo e julgamento do Dev;
- não há mecanismo automático universal para impedir alterações fora do escopo;
- controle de pendências em `.md` depende de acompanhamento humano;
- não foi consolidado um template canônico único de prompt master para todos os projetos.

Essas ausências não devem ser silenciosamente preenchidas nesta documentação.

---

44. OPORTUNIDADES FUTURAS — FORA DO ESCOPO

Já identificadas nas fontes:

- alinhar formalmente a hierarquia entre Contexto Consolidado, Sitemap e Briefing Final;
- eventual evolução do controle de pendências;
- eventual padronização adicional de artefatos de entrada.

Nenhuma dessas oportunidades é desenvolvida neste Manual.

PARTE XII — RELAÇÃO COM OS OUTROS MANUAIS E SOP

45. Relação com o Manual 3 — Construção Visual

O Manual 2 entrega:

- contexto compreendido;
- direção;
- referências;
- restrições;
- `design.md`.

O Manual 3 explica como isso é transformado em Home, páginas, componentes, conteúdo, efeitos e consistência visual.

---

46. Relação com o Manual 4 — Projeto-Base, Admin e Funcionalidades

O Manual 2 identifica:

- módulos/capacidades aplicáveis;
- restrições do cliente;
- contexto do projeto.

O Manual 4 explica como a Base Admin e suas capacidades são configuradas/adaptadas.

---

47. Relação com o Manual 5 — Codex, Lovable e GitHub

O Manual 2 explica **como o contexto é criado e preservado**.

O Manual 5 explica **como as ferramentas e o repositório são operados**.

Não se deve duplicar, no Manual 2, clone, Git, branch, commit ou rollback.

---

48. Relação com o Manual 6 — QA, Debugging, Segurança e SEO

Este Manual trata sinais de que a IA entendeu ou não entendeu o projeto.

O Manual 6 trata a validação do **produto implementado** e a investigação de falhas técnicas.

---

49. SOP relacionado

Este Manual é a base principal para:

SOP-001 — Contextualizar um Novo Projeto WSS

O SOP formaliza este conhecimento em sequência executável, incluindo:

- gatilho;
- inputs;
- leitura;
- organização;
- prompt master;
- validação de entendimento;
- stop condition.

O SOP já foi produzido e aprovado materialmente; sua formalização versionada ocorre em gate posterior ao fechamento dos Manuais.

---

50. Critério de conclusão deste Manual

Este Manual cumpre seu objetivo quando o leitor consegue responder:

1. quais documentos chegam ao Dev;
2. como eles são utilizados na prática;
3. qual é a diferença entre intenção da WSS e prática do Dev na hierarquia de fontes;
4. como o ChatGPT pessoal entra no fluxo;
5. para que serve o prompt master;
6. o que é contexto persistente;
7. o papel de `contexto.md`;
8. o papel de `design.md`;
9. como o contexto é recuperado ao trocar de ferramenta;
10. como o Dev limita o escopo dos prompts;
11. como percebe perda de contexto;
12. o que faz quando a IA inventa requisitos;
13. como lida com informações pendentes;
14. quais fatos não devem ser inventados;
15. quais pontos continuam sem padronização formal.

---

51. Fontes principais desta versão

Baseline canônico

1. `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
2. `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
3. `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
4. `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
5. `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

Evidência humana consultada para nuance

- `questionario_metodo_desenvolvimento_AS-IS_perguntas_1_a_70.docx`;
- `respostas p1 p2.txt`.

Exemplo real

- `Dr_Guilherme_Mendonca_Contexto_Desenvolvimento_v1.0_FINAL.md`.

Norma técnica vigente relacionada

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Versões anteriores/superadas de AS-IS, Matriz, Handoff e Plano não foram usadas como baseline ativo.

Status: FINAL — APROVADO

WEB STAR STUDIO

MANUAL

MÉTODO DE DESENVOLVIMENTO DE SITES

MANUAL 3
Construção Visual

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

MANUAL

MANUAL 3

Construção Visual

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
ESCOPO DOCUMENTAL	primeira documentação ampliada — P0 + P1 + P2 relevante
PROJETO	WSS — Método de Desenvolvimento de Sites
FORMALIZAÇÃO	Pacote Corretivo Pós-Entrega v1.1 aprovado pela Coordenação em 11/09/2026; versão canônica vigente, com v1.0 FINAL preservado como baseline histórico.

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Objetivo 7

2. Escopo e fronteira 7

3. Posição deste Manual no fluxo 9

PARTE I — REFERÊNCIAS VISUAIS 10

4. Fonte principal de referências 11

5. Quem participa da escolha 11

6. Referências não precisam ser do mesmo setor 11

7. Fontes auxiliares 12

PARTE II — ANÁLISE DE REFERÊNCIA 13

8. A análise não é apenas “olhar o site” 14

9. O que tende a ser observado 14

10. DevTools não é etapa obrigatória de toda referência 14

11. Screenshots como evidência visual 15

PARTE III — TRADUÇÃO DA REFERÊNCIA PARA O PROJETO 16

12. Preservar essência, não fazer cópia cega 17

13. O que pode ser reaproveitado conceitualmente 17

14. Combinação de referências 17

15. Prompting visual que tende a funcionar 18

16. Anti-pattern de prompting visual 18

PARTE IV — HOME COMO ÂNCORA VISUAL 19

17. A Home é o principal ponto de partida 20

18. Primeira versão 20

19. O que é refinado na Home 20

20. Fidelidade e adequação 21

21. Quando avançar para páginas internas 21

PARTE V — PÁGINAS INTERNAS 22

22. Sitemap como guia de cobertura 23
23. Ordem usual 23
24. Paralelismo 23
25. Primeira geração de página interna 23
26. Elementos recorrentes 24
27. Elementos que podem variar 24
28. Peso visual diferente entre páginas 24

PARTE VI — CRITÉRIO PRÁTICO DE PÁGINA PRONTA 25

29. Três verificações principais 26
30. Não confundir “pronta visualmente” com “homologada” 26
31. AUSÊNCIA ATUAL DE PADRONIZAÇÃO 26

PARTE VII — COMPONENTIZAÇÃO E REUSO 27

32. Não existe regra humana formal de componentização 28
33. Reuso declarado 28
34. Reuso técnico confirmado 28
35. Formulação AS-IS segura 28
36. Componentização não deve apagar particularidade 29

PARTE VIII — DESIGN SYSTEM E CONSISTÊNCIA VISUAL 30

37. design.md como âncora operacional 31
38. O código também materializa consistência 31
39. Consistência híbrida 31
40. Não existe Design System único universal 32
41. O que a consistência busca preservar 32

PARTE IX — ASSETS E IMAGENS 33

42. Entrada de imagens do cliente 34
43. Catalogação com Codex 34
44. Quando falta material 34
45. Oportunidade futura sobre origem/autorização 34
46. Compressão, formato e naming 35
47. Imagens e performance 35
48. Imagens e responsividade 35

PARTE X — CONTEÚDO 36

49. IA participa fortemente da redação 37
50. Conteúdo pode chegar parcialmente pronto 37
51. Fatos sensíveis não devem ser inventados 37
52. Informação faltante 38
53. Validação factual 38
54. AUSÊNCIA ATUAL DE PADRONIZAÇÃO 38

PARTE XI — EFEITOS E ANIMAÇÕES **39**
55. Efeitos fazem parte da linguagem visual 40
56. Tecnologias e técnicas observadas 40
57. Daniel P como evidência de sofisticação 41
58. Como investigar um efeito 41
59. Quando adaptar, simplificar, substituir ou remover um efeito 41
60. Problemas reais com efeitos 42

PARTE XII — RESPONSIVIDADE **43**
61. Responsividade entra durante a construção 44
62. Tamanhos/dispositivos relevantes 44
63. Desktop e mobile podem exigir soluções diferentes 44
64. Preservar breakpoint já validado 44
65. AUSÊNCIA ATUAL DE PADRONIZAÇÃO 45

PARTE XIII — PERFORMANCE **46**
66. Performance é tratada de forma pragmática 47
67. Primeiro foco quando há peso/lentidão 47
68. Sofisticação com contenção técnica 47
69. Gate de performance atual 48
70. Performance × referência 48

PARTE XIV — ACESSIBILIDADE AS-IS **49**
71. Existe atenção consciente 50
72. Problemas reais detectados 50
73. Acessibilidade não é auditoria formal 50
74. Limite deste Manual 50

PARTE XV — RELAÇÃO COM NORMA TÉCNICA **52**
75. NORMA TÉCNICA VIGENTE 53

76. Não reescrever o AS-IS com a norma 53

PARTE XVI — CASOS REAIS **55**

77. Guilherme Mendonça 56

78. Daniel P 56

79. O que não generalizar a partir dos casos 56

PARTE XVII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO **57**

80. Lista consolidada deste Manual 58

PARTE XVIII — OPORTUNIDADES FUTURAS — FORA DO ESCOPO **59**

81. Oportunidades já identificadas 60

PARTE XIX — RELAÇÃO COM OS OUTROS MANUAIS **61**

82. Relação com o Manual 2 62

83. Relação com o Manual 4 62

84. Relação com o Manual 5 62

85. Relação com o Manual 6 62

86. Relação com o Manual 8 63

PARTE XX — SOPS DERIVADOS **64**

87. SOP-003 — Construir e Refinar a Home a partir das Referências 65

88. SOP-004 — Construir Páginas Internas Mantendo Consistência Visual 65

89. Efeitos não viram SOP próprio nesta primeira versão 65

PARTE XXI — CRITÉRIO DE CONCLUSÃO **67**

90. Critério de conclusão deste Manual 68

91. Fontes principais desta versão 68

1. Objetivo

Este Manual documenta como a WSS transforma referências, contexto, Sitemap e conteúdo em uma experiência visual implementada e coerente.

Seu foco é a construção visual do site público:

```
referências
→ análise
→ tradução para o cliente
→ Home
→ páginas internas
→ reuso / consistência
→ assets / conteúdo
→ efeitos
→ responsividade
→ performance
→ atenção à acessibilidade
```

Ao final, o leitor deve compreender:

- como as referências são escolhidas e utilizadas;
- como a Home estabelece a linguagem visual;
- como páginas internas derivam dessa linguagem;
- como o Dev trabalha fidelidade e adaptação;
- como reuso e componentização aparecem no AS-IS;
- como `design.md` se relaciona com tokens/componentes no código;
- como imagens e conteúdo entram na composição;
- como efeitos/animações são escolhidos ou descartados;
- como desktop, notebook e mobile influenciam a construção;
- como performance é tratada de forma pragmática;
- qual é a atenção atual à acessibilidade;
- o que ainda não possui padronização formal.

Este Manual registra o **AS-IS**. Ele não cria um Design System formal universal, não cria política nova de performance, não institui auditoria WCAG e não transforma um projeto visualmente sofisticado em padrão obrigatório para todos os clientes.

---

2. Escopo e fronteira

2.1 O que entra neste Manual

Este Manual cobre:

1. pesquisa de referências;
2. escolha da referência principal e referências secundárias;

3. análise visual;
4. screenshots e recortes;
5. investigação de comportamento/efeito;
6. tradução de referência para o cliente;
7. Home;
8. páginas internas;
9. ordem prática das páginas;
10. reuso;
11. componentização;
12. Design System híbrido;
13. `design.md`;
14. tokens/variáveis/componentes;
15. assets e imagens;
16. conteúdo textual;
17. efeitos e animações;
18. responsividade;
19. performance;
20. acessibilidade AS-IS;
21. critérios práticos de “pronto” na construção;
22. casos reais úteis para esta camada.

2.2 O que não entra neste Manual

Ficam fora:

- preparação inicial do contexto e prompt master em profundidade;
- Git/GitHub;
- clone, pull, commit e push;
- operação detalhada Codex/Lovable/Claude;
- configuração do Admin;
- Chatbot/RAG;
- QA final e regressão;
- debugging;
- SEO/AEO/GEO;
- segurança;
- publicação.

O M3 é proprietário de **como construir e adaptar visualmente**.

O M6 é proprietário de **como validar tecnicamente o que foi construído**.

3. Posição deste Manual no fluxo

Este Manual começa quando:

- o projeto já foi contextualizado;
- as referências já foram encaminhadas/validadas;
- existe Sitemap;
- existe direção do cliente;
- o projeto já possui primeira base no Lovable/local.

Seu recorte principal é:

```
primeira base do institucional
→ comparar com referência
→ refinar Home
→ consolidar linguagem visual
→ desenvolver páginas internas
→ tratar conteúdo/assets/reuso
→ adaptar efeitos
→ ajustar responsividade/performance
→ considerar página visualmente pronta
```

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE I — REFERÊNCIAS VISUAIS

4. Fonte principal de referências

A principal fonte declarada é:

Awwwards

A equipe busca referências modernas com qualidade de:

- UI;
- UX;
- design;
- performance percebida;
- composição;
- navegação;
- efeitos.

Normalmente são selecionadas **3 a 5 referências**.

---

5. Quem participa da escolha

Participam da busca de referências:

- Felipe;
- Dev;
- Analice;
- Felipe/CEO.

No kickoff, o cliente pode validar:

- referência principal;
- características específicas de referências secundárias.

Quando o desenvolvimento começa, a direção já tende a estar registrada no Contexto de Desenvolvimento Consolidado.

---

6. Referências não precisam ser do mesmo setor

A referência pode vir de outro segmento.

O critério não é apenas semelhança setorial.

A equipe pode aproveitar:

- estrutura;
- ritmo;

- navegação;
- sofisticação;
- uso de tipografia;
- Hero;
- animações;
- composição de seções;
- experiência.

A adaptação ao cliente é feita depois.

---

7. Fontes auxiliares

Outras fontes observadas:

- Navbar Gallery;
- React Bits;
- 21st.dev;
- Framer;
- CodePen;
- X/Twitter;
- Instagram.

Elas aparecem principalmente para:

- navbars;
- componentes;
- animações;
- efeitos;
- microinterações;
- ideias visuais.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe catálogo obrigatório de referências que toda equipe precise consultar.

Awwwards é a principal fonte declarada, e as demais são auxiliares.

PARTE II — ANÁLISE DE REFERÊNCIA

8. A análise não é apenas “olhar o site”

O Dev combina diferentes formas de leitura:

- navegação visual;
- observação de layout;
- observação de comportamento;
- screenshots;
- comparação de proporções;
- investigação com Codex quando o comportamento é complexo.

---

9. O que tende a ser observado

Na prática, a leitura pode considerar:

- Hero;
- proporções;
- hierarquia;
- tipografia;
- paleta;
- imagem;
- espaçamento;
- largura de conteúdo;
- cards;
- CTAs;
- transições;
- scroll;
- componentes;
- comportamento responsivo;
- ritmo visual entre seções.

Não existe checklist universal formal para essa análise.

---

10. DevTools não é etapa obrigatória de toda referência

Não há inspeção técnica fixa via DevTools para qualquer referência.

Quando necessário, o Dev ou Codex pode aprofundar:

- comportamento;
- animação;

- efeito;
- interação;
- estrutura aparente.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> A profundidade da inspeção varia com a complexidade do que se quer reproduzir.

---

11. Screenshots como evidência visual

Screenshots são usados para reduzir ambiguidade.

Eles ajudam a indicar:

- qual estado;
- qual seção;
- qual enquadramento;
- qual detalhe;
- qual breakpoint.

Essa prática é especialmente importante no prompting visual.

PARTE III — TRADUÇÃO DA REFERÊNCIA PARA O PROJETO

12. Preservar essência, não fazer cópia cega

A prática observada é buscar fidelidade visual, mas adaptar quando necessário.

A referência pode ser modificada quando:

- entra em conflito com o briefing;
- não combina com o cliente;
- prejudica usabilidade;
- não funciona bem em mobile;
- exigiria cópia literal inadequada;
- existe necessidade funcional diferente.

Formulação AS-IS

> preservar a essência visual e comportamental útil, adaptando-a ao cliente e ao contexto real do projeto.

---

13. O que pode ser reaproveitado conceitualmente

Pode-se aproveitar, por exemplo:

- composição;
- ritmo;
- proporções;
- modelo de Hero;
- comportamento de seção;
- hierarquia;
- interação;
- transição;
- direção tipográfica;
- lógica de cards.

Isso não implica duplicação literal de conteúdo, identidade ou marca.

---

14. Combinação de referências

Um projeto pode ter:

- uma referência principal;
- características selecionadas de referências secundárias.

O resultado pode combinar elementos.

Não há exigência de reproduzir um único site de referência de ponta a ponta.

---

15. Prompting visual que tende a funcionar

A prática corrente favorece:

1. escolher uma seção;
2. fornecer screenshot;
3. fornecer link;
4. pedir estudo da referência;
5. limitar escopo;
6. buscar fidelidade;
7. adaptar depois ao cliente.

Esse processo reduz o grau de interpretação livre da IA.

---

16. Anti-pattern de prompting visual

Abordagem considerada fraca:

> enviar apenas o link do site e pedir que a IA seja “extremamente fiel”.

O problema é a amplitude excessiva.

A IA pode:

- escolher detalhes errados;
- simplificar a composição;
- interpretar de forma genérica;
- alterar áreas não pedidas.

O catálogo formal de anti-patterns pertence ao Manual 8.

PARTE IV — HOME COMO ÂNCORA VISUAL

17. A Home é o principal ponto de partida

A Home funciona como primeira grande materialização da direção visual.

Ela recebe mais refinamento inicial porque estabelece:

- identidade;
- ritmo;
- linguagem;
- proporções;
- atmosfera;
- padrões de seções;
- comportamento;
- referência para as páginas internas.

---

18. Primeira versão

A primeira versão normalmente nasce no Lovable a partir do prompt master.

Depois, o Dev compara com a referência e inicia refinamento.

Quase nunca a primeira geração é considerada suficiente em fidelidade.

---

19. O que é refinado na Home

O Dev trabalha principalmente:

- estrutura;
- proporções;
- tipografia;
- imagens;
- espaçamentos;
- alinhamentos;
- comportamento;
- efeitos;
- responsividade.

O processo é iterativo.

---

20. Fidelidade e adequação

A Home precisa equilibrar:

- referência visual;
- briefing;
- identidade do cliente;
- conteúdo real;
- conversão/CTA;
- usabilidade;
- mobile.

Quando esses elementos entram em conflito, não há algoritmo formal de decisão.

O julgamento permanece humano.

---

21. Quando avançar para páginas internas

Quando a Home já estabelece bem a linguagem visual, o Dev avança.

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe score objetivo universal que diga:

> “Home pronta = X pontos”.

A decisão é predominantemente visual e funcional.

PARTE V — PÁGINAS INTERNAS

22. Sitemap como guia de cobertura

O Sitemap continua sendo referência para saber:

- quais páginas precisam existir;
- qual estrutura pública deve ser coberta;
- quais páginas são prioritárias.

Ele não determina sozinho o grau de sofisticação visual de cada página.

---

23. Ordem usual

Prática predominante observada:

```
Home
→ Contato
→ Blog
```

Contato e Blog costumam vir logo após a Home por serem, em geral, páginas mais simples e rápidas de estabilizar. Depois, as demais páginas avançam conforme:

- complexidade;
- referência;
- ideia visual;
- importância estratégica.

Essa sequência não é mandatória e pode mudar por prioridade estratégica, conteúdo, referência ou dependência técnica.

---

24. Paralelismo

O trabalho pode ser paralelo.

Se surge uma boa ideia para uma seção de outra página, o Dev pode trabalhá-la antes de terminar a página atual.

Isso é compatível com a natureza visual do processo.

---

25. Primeira geração de página interna

A IA pode receber instrução semelhante a:

> com base na Home e no Sitemap, crie a página X.

A primeira versão é avaliada pelo Dev e normalmente refinada.

---

26. Elementos recorrentes

Podem ser reutilizados:

- Footer;
- CTA final;
- FAQ;
- identidade geral;
- linguagem;
- padrões visuais.

---

27. Elementos que podem variar

Podem mudar conforme a página:

- Hero;
- depoimentos;
- estrutura de seções;
- efeitos;
- grau de sofisticação;
- composição.

A consistência não significa que todas as páginas precisam ter o mesmo layout.

---

28. Peso visual diferente entre páginas

Prática observada:

- Sobre tende a receber bastante atenção;
- página geral de Especialidades também;
- páginas individuais podem ser mais diretas;
- página prioritária para o negócio pode receber refinamento adicional.

Isso é um critério prático, não regra formal universal.

PARTE VI — CRITÉRIO PRÁTICO DE PÁGINA PRONTA

29. Três verificações principais

Na fase de construção, o Dev considera principalmente:

1. todas as seções previstas no Sitemap estão presentes;
2. a qualidade visual está satisfatória e não parece simples demais;
3. botões e ações principais funcionam.

---

30. Não confundir “pronta visualmente” com “homologada”

Esses critérios não substituem:

- QA;
- testes completos;
- regressão;
- SEO;
- segurança;
- produção.

Essas etapas pertencem ao Manual 6 e Manual 7.

---

31. AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe checklist formal detalhado de “página pronta” para a construção visual.

O método atual preserva julgamento humano.

PARTE VII — COMPONENTIZAÇÃO E REUSO

32. Não existe regra humana formal de componentização

O Dev não declarou uma disciplina formal do tipo:

> “se um elemento aparece duas vezes, deve virar componente”.

A prática é pragmática.

---

33. Reuso declarado

O Dev tende a:

- copiar/adaptar elementos entre páginas;
- preservar padrão;
- alterar localmente quando a página precisa de solução própria;
- evitar repetir em excesso efeitos muito elaborados.

---

34. Reuso técnico confirmado

A VAL-001 confirmou reuso real no código do Daniel P.

Exemplos:

- `FaqSection` compartilhado;
- `PillarCards` parametrizado;
- `SpecialtyPageLayout` orientado por dados;
- componentes específicos quando a solução é local.

Isso demonstra que o AS-IS não é apenas copiar e colar visualmente.

---

35. Formulação AS-IS segura

> **Reuso pragmático e híbrido: componentes compartilhados quando há ganho claro, layouts parametrizados quando páginas possuem estrutura semelhante e componentes locais quando a solução é específica.**

---

36. Componentização não deve apagar particularidade

O fato de existir um componente compartilhado não significa que toda página deva ser forçada ao mesmo layout.

Quando a página exige solução própria, o projeto pode manter componente específico.

PARTE VIII — DESIGN SYSTEM E CONSISTÊNCIA VISUAL

37. `design.md` como âncora operacional

O `design.md` é a principal referência persistente declarada pelo Dev para:

- tipografia;
- paleta;
- espaçamento;
- direção visual;
- referências;
- comportamento esperado.

Codex e Claude podem reler esse arquivo entre páginas/chats.

---

38. O código também materializa consistência

A VAL-001 confirmou no Daniel P:

- tokens CSS;
- variáveis globais;
- temas;
- componentes compartilhados;
- layouts parametrizados.

Portanto, a consistência não depende apenas do `design.md`.

---

39. Consistência híbrida

Formulação segura:

> **`design.md` orienta intenção e continuidade; o código pode materializar tokens, utilitários, componentes e variantes, coexistindo com decisões locais.**

---

40. Não existe Design System único universal

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não há evidência de:

- um Design System único;
- completamente formal;
- centralizado;
- idêntico em todos os projetos WSS.

Também foram encontradas constantes locais específicas de subsistemas/páginas.

---

41. O que a consistência busca preservar

No AS-IS, a consistência procura evitar:

- tipografias desconectadas;
- espaçamentos arbitrários;
- cores sem relação com a direção;
- componentes visualmente incoerentes;
- linguagem diferente em páginas que deveriam pertencer ao mesmo sistema.

Mas isso não significa uniformidade total.

PARTE IX — ASSETS E IMAGENS

42. Entrada de imagens do cliente

Quando o cliente envia imagens, normalmente chegam por Google Drive.

O Dev baixa o material.

---

43. Catalogação com Codex

O Dev pode pedir ao Codex:

- analisar foto por foto;
- catalogar;
- sugerir em qual página/seção cada imagem funciona melhor.

Isso transforma o conjunto bruto de imagens em mapa de uso.

---

44. Quando falta material

O Dev pode:

- procurar imagens públicas do profissional;
- consultar Doctoralia;
- consultar Google;
- gerar imagens com IA;
- usar Codex para geração contextualizada.

Canva aparece pontualmente.

---

45. Oportunidade futura sobre origem/autorização

OPORTUNIDADE FUTURA — FORA DO ESCOPO

Foi identificada necessidade futura de revisar:

- uso de imagens públicas;
- autorização;
- origem;
- geração de imagens a partir do cliente.

Este Manual não cria política nova.

---

46. Compressão, formato e naming

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Compressão, formato e naming:

- são tratados quando necessário;
- mas não constituem hoje uma etapa formal separada do método.

Isso deve ser documentado como ausência, não substituído por pipeline ideal.

---

47. Imagens e performance

Quando existe peso/lentidão, imagens são um dos primeiros pontos de atenção.

O Dev relatou uso de:

- otimização;
- WebP;
- ajustes de assets.

Mas não existe pipeline universal formal de imagem para todos os projetos.

---

48. Imagens e responsividade

O enquadramento pode variar por breakpoint.

O Dev pode:

- reposicionar foco;
- ajustar crop;
- tratar mobile separadamente;

sem alterar o desktop já validado.

PARTE X — CONTEÚDO

49. IA participa fortemente da redação

O Dev estimou que cerca de **90% dos textos** são criados ou adaptados por:

- Claude;
- Codex;
- Lovable.

Isso inclui:

- estruturação;
- adaptação;
- complementação;
- ajustes de linguagem.

---

50. Conteúdo pode chegar parcialmente pronto

Copys e direcionamentos podem chegar do:

- briefing;
- contexto;
- decisões do projeto;
- conteúdo fornecido pelo cliente.

A IA não cria obrigatoriamente tudo do zero.

---

51. Fatos sensíveis não devem ser inventados

O Dev evita inventar ou alterar sem validação:

- formação;
- títulos;
- credenciais;
- procedimentos;
- resultados;
- especialidades;
- locais;
- números;
- afirmações médicas;
- outros fatos sensíveis.

---

52. Informação faltante

Quando falta dado confiável:

- pode ser usado placeholder em casos simples;
- pode ser registrada pendência em `.md`;
- o conteúdo pode avançar sem transformar o placeholder em fato validado.

---

53. Validação factual

O cliente funciona como principal validação factual final.

Limite documental

Isso não equivale a:

- revisão editorial formal universal;
- medical review formal obrigatório;
- workflow de aprovação de conteúdo padronizado.

---

54. AUSÊNCIA ATUAL DE PADRONIZAÇÃO

O controle de pendências em `.md` depende de acompanhamento humano.

O Dev reconheceu risco de esquecimento.

Esse é um achado AS-IS.

PARTE XI — EFEITOS E ANIMAÇÕES

55. Efeitos fazem parte da linguagem visual

Efeitos podem ser usados quando:

- sustentam a referência;
- combinam com o cliente;
- aumentam sofisticação;
- não prejudicam o uso.

---

56. Tecnologias e técnicas observadas

Nos projetos principais aparecem:

- Framer Motion / Motion;
- GSAP;
- ScrollTrigger;
- SplitText;
- Lenis;
- anime.js;
- AOS;
- Three.js;
- React Three Fiber;
- Drei;
- Lottie;
- Swiper;
- CSS animations;
- CSS transitions;
- keyframes;
- IntersectionObserver;
- requestAnimationFrame;
- parallax;
- SVG filters;
- masks;
- clip-path;
- WebGL;
- shaders;
- smooth scroll;
- text reveal.

A lista demonstra variedade, não obrigação.

---

57. Daniel P como evidência de sofisticação

A VAL-001 confirmou no projeto Daniel P:

- Framer Motion;
- GSAP;
- ScrollTrigger;
- SplitText;
- Three;
- React Three Fiber;
- Drei;
- efeitos de retrato;
- scroll customizado;
- motion em componentes.

Esse caso sustenta que a WSS consegue operar com soluções visuais sofisticadas.

---

58. Como investigar um efeito

A prática pode incluir:

1. identificar a seção/efeito;
2. fornecer screenshot;
3. fornecer link;
4. pedir estudo do comportamento;
5. investigar técnica/biblioteca;
6. reproduzir de forma isolada;
7. adaptar ao projeto;
8. validar desktop/mobile.

Esse procedimento ficará incorporado ao Manual e aos SOPs de Home/página, não como SOP próprio nesta primeira versão.

---

59. Quando adaptar, simplificar, substituir ou remover um efeito

Quando um efeito apresenta problema, a preferência prática é preservar seu valor visual sempre que possível:

1. adaptar;
2. simplificar ou substituir;
3. remover apenas quando a solução continuar inviável, instável ou prejudicial.

Essa ordem é aplicada quando o efeito:

- está quebrado;
- não combina com o cliente;
- conflita com briefing;
- prejudica usabilidade;
- prejudica mobile;
- gera instabilidade;
- pesa demais.

---

60. Problemas reais com efeitos

Foram relatados:

- objeto 3D instável;
- borda de Hero quebrando em notebook;
- scroll horizontal prejudicando usabilidade.

Esses exemplos mostram que sofisticação não é objetivo absoluto.

PARTE XII — RESPONSIVIDADE

61. Responsividade entra durante a construção

O Dev não espera necessariamente a QA final para pensar em mobile.

Responsividade faz parte do refinamento visual.

---

62. Tamanhos/dispositivos relevantes

No fluxo atual aparecem como referências práticas:

- desktop/monitor;
- notebook/MacBook;
- mobile físico.

O QA final desses tamanhos pertence ao Manual 6.

---

63. Desktop e mobile podem exigir soluções diferentes

Um efeito ou composição aprovado no desktop não precisa ser reproduzido de forma idêntica no mobile.

O Dev pode:

- reposicionar/adaptar;
- simplificar;
- substituir quando necessário;
- remover o efeito apenas quando a adaptação/simplificação não resolver de forma aceitável;
- mudar crop;
- alterar composição;
- adaptar navegação.

---

64. Preservar breakpoint já validado

No prompting, o Dev explicita quando:

- desktop já está validado;
- mobile é o único foco;
- ou o contrário.

Isso reduz regressão lateral.

---

65. AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe matriz extensa obrigatória de todos os breakpoints intermediários.

O fluxo atual prioriza tamanhos/dispositivos reais e de maior impacto.

PARTE XIII — PERFORMANCE

66. Performance é tratada de forma pragmática

O Dev considera:

- tempo percebido;
- fluidez;
- peso visual;
- efeitos;
- imagens;
- 3D.

A meta é preservar qualidade visual sem tornar a experiência lenta ou instável.

---

67. Primeiro foco quando há peso/lentidão

Práticas relatadas:

- otimizar imagens;
- usar WebP;
- revisar efeitos;
- revisar animações;
- revisar 3D;
- pedir ao Codex otimização sem perder a proposta visual.

---

68. Sofisticação com contenção técnica

No Daniel P foram observados:

- lazy loading;
- chunking;
- separação de `framer-motion`.

Isso demonstra preocupação técnica real em conter o custo de uma interface sofisticada.

Não deve ser transformado em receita universal de bundling.

---

69. Gate de performance atual

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe score numérico universal de aprovação.

O gate declarado é principalmente perceptivo:

- carregamento aceitável;
- fluidez;
- experiência sem travamentos relevantes.

Auditorias técnicas adicionais podem ocorrer, mas não substituem esse achado AS-IS.

---

70. Performance × referência

Quando o efeito desejado prejudica a experiência, o Dev prioriza:

1. otimizar/adaptar;
2. simplificar ou substituir;
3. remover somente se a solução continuar inadequada.

Fidelidade não é prioridade absoluta diante de quebra/usabilidade/performance.

PARTE XIV — ACESSIBILIDADE AS-IS

71. Existe atenção consciente

O AS-IS inclui atenção a:

- contraste;
- tamanho de fonte;
- legibilidade;
- mobile;
- público idoso;
- alt text.

---

72. Problemas reais detectados

No Daniel P foram encontrados, por exemplo:

- texto de 7px em mobile;
- botão-ícone sem nome acessível.

Esses itens foram tratados como correções específicas.

---

73. Acessibilidade não é auditoria formal

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe hoje auditoria formal de acessibilidade obrigatória em todos os projetos.

Também não existe checklist WCAG universal incorporado ao método atual.

Esse fato precisa permanecer explícito.

---

74. Limite deste Manual

Este Manual registra o que é praticado hoje.

Ele não introduz:

- certificação;
- score;
- nível WCAG;
- auditoria formal;

- gate de acessibilidade novo.

PARTE XV — RELAÇÃO COM NORMA TÉCNICA

75. NORMA TÉCNICA VIGENTE

A construção visual permanece descrita acima como **AS-IS**. Em paralelo, o padrão canônico vigente de SEO/AEO/GEO é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Quando aplicável à superfície pública, essa norma adiciona requisitos e referências que **não substituem** o julgamento visual do Dev nem reescrevem a prática histórica.

Performance normativa

O padrão define como metas **recomendadas**, no percentil 75 quando houver dados adequados:

- LCP ≤ 2,5 s;
- INP ≤ 200 ms;
- CLS ≤ 0,1.

Dados de campo/CrUX devem ser priorizados quando disponíveis; PageSpeed Insights e Lighthouse permanecem instrumentos de diagnóstico e regressão. Essas metas não são promessa de ranking e **não viram o gate visual AS-IS deste Manual**.

Terceiros, Chat e CTA

Scripts, widgets, Chat/chatbot/CTA e outros terceiros não devem bloquear a renderização nem causar degradação desproporcional da experiência. A funcionalidade central da página deve continuar disponível diante de falha de terceiro.

Imagens

Na camada normativa:

- `alt` útil/contextual é obrigatório para imagens informativas;
- imagens responsivas/eficientes são recomendadas;
- mídia não deve ser a única portadora de informação essencial quando isso comprometer compreensão/rastreabilidade.

Semântica e acessibilidade

HTML semântico e acessibilidade possuem valor próprio e podem melhorar robustez para tecnologias assistivas e agentes. O padrão **não** autoriza afirmar que ARIA ou acessibilidade geram ranking/GEO direto.

Portanto, permanecem simultaneamente verdadeiras:

- **AS-IS:** não existe auditoria WCAG formal universal incorporada hoje ao método do Dev;
- **NORMA:** requisitos técnicos específicos de semântica, imagens, experiência e acessibilidade devem ser respeitados quando classificados como obrigatórios/aplicáveis pelo padrão vigente.

76. Não reescrever o AS-IS com a norma

A norma pode exigir mais formalização do que o Dev historicamente executa.

Este Manual deve preservar as duas camadas:

AS-IS

Como o Dev constrói hoje.

NORMA

O que o padrão técnico exige.

O detalhamento de SEO/AEO/GEO fica no Manual 6.

PARTE XVI — CASOS REAIS

77. Guilherme Mendonça

Uso:

- principal exemplo do fluxo atual;
- Home como âncora;
- desenvolvimento a partir de referências;
- evolução para páginas internas;
- adaptação visual do institucional.

Exclusão

O esboço antecipado antes de todas as definições do cliente não integra o método.

---

78. Daniel P

Uso:

- caso visualmente sofisticado;
- componentização;
- tokens;
- motion;
- 3D;
- lazy loading;
- chunking.

Esse caso demonstra amplitude técnica, não padrão mínimo obrigatório.

---

79. O que não generalizar a partir dos casos

Não afirmar que:

- todo site precisa de 3D;
- todo projeto deve usar GSAP;
- todo projeto precisa do mesmo nível de motion;
- todo projeto deve seguir os mesmos tokens;
- toda página interna deve usar layout parametrizado;
- toda Home deve ter a mesma sofisticação.

O caso demonstra capacidade e práticas possíveis.

PARTE XVII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

80. Lista consolidada deste Manual

O AS-IS não possui:

- checklist formal universal de análise de referência;
- regra humana formal de componentização;
- Design System único completamente centralizado;
- checklist detalhado universal de “página pronta”;
- pipeline formal separado de compressão/formato/naming de assets;
- revisão editorial/medical review formal universal;
- mecanismo automático para garantir resolução de todas as pendências de conteúdo;
- matriz obrigatória de efeitos por tipo de projeto;
- matriz extensa obrigatória de breakpoints intermediários;
- score numérico universal de performance;
- auditoria formal obrigatória de acessibilidade.

Essas ausências são parte da documentação.

PARTE XVIII — OPORTUNIDADES FUTURAS — FORA DO ESCOPO

81. Oportunidades já identificadas

Sem desenvolver solução nesta fase:

- formalizar políticas adicionais de componentização/design system;
- formalizar auditoria de acessibilidade;
- revisar política de imagens públicas/geradas;
- formalizar tratamento de assets;
- aperfeiçoar controle de pendências de conteúdo.

Nenhuma dessas oportunidades altera retroativamente o método atual.

PARTE XIX — RELAÇÃO COM OS OUTROS MANUAIS

82. Relação com o Manual 2

O Manual 2 entrega:

- contexto;
- referências;
- `design.md`;
- restrições.

O Manual 3 transforma isso em experiência visual.

---

83. Relação com o Manual 4

O Manual 4 é proprietário das capacidades do Admin.

Quando Blog, Chat ou outro módulo precisa de adaptação visual, o M3 fornece os princípios visuais; a configuração funcional permanece no M4.

---

84. Relação com o Manual 5

O Manual 5 explica como:

- Codex;
- Claude;
- Lovable;
- GitHub;

são operados.

O Manual 3 explica **o que o Dev busca construir visualmente com essas ferramentas**.

---

85. Relação com o Manual 6

O Manual 3 trata:

- responsividade durante construção;
- performance durante construção;
- atenção básica à acessibilidade.

O Manual 6 trata:

- QA;
- regressão;

- testes;
- validação final;
- auditorias.

---

86. Relação com o Manual 8

O Manual 8 concentrará:

- caso Daniel P em profundidade;
- anti-patterns visuais;
- problemas conhecidos;
- práticas depreciadas.

PARTE XX — SOPs DERIVADOS

87. SOP-003 — Construir e Refinar a Home a partir das Referências

O SOP formaliza deste Manual:

- inputs;
- leitura da referência;
- recorte;
- primeira versão;
- refinamento;
- conteúdo;
- assets;
- efeitos;
- responsividade;
- critério de conclusão.

---

88. SOP-004 — Construir Páginas Internas Mantendo Consistência Visual

O SOP formaliza:

- Sitemap;
- linguagem da Home;
- reuso;
- variação;
- conteúdo;
- consistência;
- breakpoint;
- critério de pronto.

---

89. Efeitos não viram SOP próprio nesta primeira versão

O procedimento de:

- investigar;
- reproduzir;
- adaptar;

fica incorporado a M3/SOP-003/SOP-004.

Isso evita fragmentação.

PARTE XXI — CRITÉRIO DE CONCLUSÃO

90. Critério de conclusão deste Manual

Este Manual cumpre seu objetivo quando o leitor consegue responder:

1. onde buscar referências;
2. como uma referência é analisada;
3. como ela é adaptada ao cliente;
4. por que a Home é a âncora visual;
5. quando avançar para páginas internas;
6. como o Sitemap orienta cobertura;
7. como reuso e variação coexistem;
8. como a componentização aparece no AS-IS;
9. o que significa Design System híbrido;
10. qual é o papel de `design.md`;
11. como imagens são recebidas/catalogadas;
12. o que acontece quando faltam assets;
13. como IA participa do conteúdo;
14. quais fatos não devem ser inventados;
15. como efeitos são escolhidos;
16. quando um efeito deve ser descartado;
17. como mobile pode divergir do desktop;
18. como performance é tratada;
19. qual é a atenção atual à acessibilidade;
20. quais ausências não devem ser preenchidas com um TO-BE.

---

91. Fontes principais desta versão

Baseline canônico

1. `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
2. `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
3. `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
4. `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
5. `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

Evidência técnica complementar

- `WSS_VAL_001_DanielP_Componentizacao_DesignSystem_Visual_v0.1.md`.

Manuais já estabilizados usados para preservar fronteiras

- `WSS_Manual_01_Visao_Geral_Toolchain_v1.1_FINAL.md`;

- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`;
- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`;
- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`.

Norma técnica vigente relacionada

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Versões anteriores/superadas de AS-IS, Matriz, Handoff e Plano não foram usadas como baseline ativo.

Status: FINAL — APROVADO

WEB STAR STUDIO

MANUAL

MÉTODO DE DESENVOLVIMENTO DE SITES

MANUAL 4
Projeto-Base, Admin e Funcionalidades

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

MANUAL

MANUAL 4

Projeto-Base, Admin e Funcionalidades

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
ESCOPO DOCUMENTAL	primeira documentação ampliada — P0 + P1 + P2 relevante
PROJETO	WSS — Método de Desenvolvimento de Sites
FORMALIZAÇÃO	Pacote Corretivo Pós-Entrega v1.1 aprovado pela Coordenação em 11/09/2026; versão canônica vigente, com v1.0 FINAL preservado como baseline histórico.

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Objetivo 6
2. Escopo e fronteira 6
3. Posição deste Manual no fluxo 7

PARTE I — O PAPEL DA BASE ADMIN 8

4. O que é a Base Admin no método atual 9
5. Projeto novo 9
6. Base ≠ produto final 9
7. O que vem da Base e o que muda por cliente 10
8. Módulos não aplicáveis 10

PARTE II — MOMENTO E ORGANIZAÇÃO DO ADMIN 12

9. Admin em paralelo com a revisão do institucional 13
10. Configuração módulo por módulo 13
11. Personalização sem reconstrução integral 13

PARTE III — BLOG 14

12. Papel do Blog 15
13. Blog público 15
14. Blog no Admin 15
15. Relação Blog × Chatbot 15
16. NORMA TÉCNICA VIGENTE relacionada ao Blog 16
17. Finalidade 16
18. Teste ponta a ponta de Lead 16
19. Integração externa no fluxo de Leads 17
20. Dados de teste 17

PARTE V — CHAT / CTA / WHATSAPP 18

21. Chat/CTA persistente 19
22. Validação do Chat 19
23. WhatsApp 19

24. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — canal universal 19

PARTE VI — CHATBOT COM IA 20

25. O Chatbot é uma capacidade operacional real 21

26. Arquitetura AS-IS do conhecimento 21

27. Origens de conhecimento 22

28. Base de Conhecimento 22

29. Limites do assistente 22

30. Captura de dados e Lead 23

31. Atendimento humano 23

32. Fallbacks 23

33. Configuração do Chatbot por cliente 23

34. Benchmark 24

35. Modelo de IA 25

PARTE VII — AUTENTICAÇÃO, PERFIS E CAPABILITIES 26

36. Duas contas internas de teste 27

37. Objetivo do teste de perfis 27

38. Presets observados 27

39. Base de Conhecimento como capacidade Webstar 27

40. Criação da conta do cliente 28

41. Manual de uso do Admin 28

PARTE VIII — INTEGRAÇÕES E ANALYTICS 29

42. Integrações no AS-IS 30

43. Search Console 30

44. Analytics e Tag Manager 30

45. NORMA TÉCNICA VIGENTE 31

PARTE IX — LIMITES WEBSTAR × EQUIPE DO CLIENTE 32

46. Separação funcional 33

47. Não confundir UI com autorização 33

PARTE X — PORTABILIDADE DA BASE PARA PROJETOS EXISTENTES 34

48. Não existe sincronização automática contínua 35

49. Portabilidade incremental e adaptativa 35

50. Princípio comparativo 35

51. Equivalentes, deltas e funcionalidade nova 36
52. Dependências primeiro 37
53. Preservar o que o destino faz melhor 37
54. Caso real — Dr. Guilherme Maia 37

PARTE XI — MIGRATIONS, DADOS E ROLLBACK **39**
55. AUSÊNCIA ATUAL DE PADRONIZAÇÃO 40
56. O que este Manual pode afirmar 40
57. Relação com troubleshooting 40

PARTE XII — NORMA, AUSÊNCIAS E OPORTUNIDADES **41**
58. NORMA TÉCNICA VIGENTE 42
59. AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 42
60. OPORTUNIDADE FUTURA — FORA DO ESCOPO 42

PARTE XIII — RELAÇÃO COM OS OUTROS MANUAIS **44**
61. Relação com o Manual 2 45
62. Relação com o Manual 3 45
63. Relação com o Manual 5 45
64. Relação com o Manual 6 45
65. Relação com o Manual 7 46
66. Relação com o Manual 8 46

PARTE XIV — SOPS DERIVADOS **47**
67. SOP-005 — Configurar Admin e Módulos Aplicáveis 48
68. SOP-006 — Configurar e Validar Chatbot e Base de Conhecimento 48
69. SOP-007 — Portar Capacidade da Base para Projeto Legado 48

PARTE XV — CRITÉRIO DE CONCLUSÃO **50**
70. Critério de conclusão deste Manual 51
71. Fontes principais desta versão 51

1. Objetivo

Este Manual documenta o papel da **Base Admin WSS** no método atual de desenvolvimento de sites e como suas capacidades são reutilizadas, configuradas, adaptadas e, quando necessário, portadas para projetos existentes.

Seu objetivo é permitir que outro desenvolvedor ou agente compreenda:

- o que a Base representa no fluxo atual;
- o que normalmente vem pronto da Base;
- o que precisa ser configurado por cliente;
- como o Admin é finalizado em paralelo ao institucional;
- como funcionam, no nível operacional, Blog, Leads, formulários, Chat/CTA, WhatsApp, Chatbot, autenticação e perfis;
- quais integrações são relevantes hoje;
- quais capacidades permanecem sob responsabilidade da Web Star Studio;
- como a Base evolui para projetos antigos sem substituir indiscriminadamente o que já existe;
- quais ausências atuais de padronização não devem ser preenchidas com um processo ideal.

Este Manual registra o **AS-IS**. Ele não redesenha a Base, não cria uma arquitetura TO-BE e não transforma detalhes de um único projeto em regra universal.

---

2. Escopo e fronteira

2.1 O que entra neste Manual

Este Manual cobre:

1. objetivo e papel da Base Admin;
2. uso da Base em projeto novo;
3. relação entre Base e institucional;
4. configuração do Admin por cliente;
5. módulos aplicáveis e não aplicáveis;
6. Blog;
7. formulários e Leads;
8. Chat/CTA;
9. WhatsApp;
10. Chatbot com IA;
11. Base de Conhecimento;
12. handoff humano;
13. autenticação;
14. perfis;

15. capabilities relevantes;
16. limites Webstar × equipe do cliente;
17. integrações e Analytics no AS-IS;
18. portabilidade incremental/adaptativa para projetos antigos;
19. ausências atuais de padronização de migrations, preservação de dados e rollback.

2.2 O que não entra neste Manual

Ficam fora:

- clone, pull, commit, push e branches;
- operação detalhada de GitHub;
- uso detalhado de Codex/Lovable/Claude;
- construção visual do institucional;
- QA completo;
- auditoria de segurança;
- RLS;
- secrets;
- autorização server-side;
- SEO/AEO/GEO em profundidade;
- publicação e domínio;
- troubleshooting completo.

Esses assuntos pertencem aos Manuais 3, 5, 6, 7 e 8.

---

3. Posição deste Manual no fluxo

Este Manual cobre principalmente a camada:

```
Projeto contextualizado
→ remix/uso da Base Admin
→ institucional em desenvolvimento
→ Admin configurado em paralelo
→ módulos adaptados ao cliente
→ capacidades testadas
→ Admin pronto para entrega
```

No fluxo atual, essa camada não é totalmente sequencial.

O institucional pode ser enviado para a primeira revisão do cliente **antes** de o Admin estar concluído.

Enquanto o cliente revisa o site público, o Dev continua configurando o Admin módulo por módulo.

PARTE I — O PAPEL DA BASE ADMIN

4. O que é a Base Admin no método atual

A Base Admin funciona como **referência funcional reutilizável** para novos projetos WSS.

Ela fornece estruturas e capacidades que não precisam ser recriadas integralmente a cada site.

No fluxo atual, iniciar todo projeto do zero foi substituído pelo uso da Base como ponto de partida preferencial.

A Base não deve ser entendida apenas como um template visual.

Ela reúne capacidades operacionais como:

- Admin;
- Blog;
- Leads;
- formulários;
- Chat/CTA;
- Chatbot, quando aplicável;
- autenticação;
- perfis;
- configurações;
- recursos de suporte a SEO/conteúdo;
- outras capacidades reutilizáveis presentes na geração corrente.

---

5. Projeto novo

No fluxo atual, o projeto normalmente começa por **remix da Base Admin no Lovable**.

A partir daí:

1. o projeto recebe o contexto do cliente;
2. o institucional começa a ser construído;
3. a Base continua como fundação funcional;
4. o Admin é adaptado ao cliente;
5. módulos aplicáveis são configurados;
6. módulos não aplicáveis podem ser ocultados/desativados.

O procedimento técnico de criar/remixar, conectar GitHub e preparar o ambiente local pertence ao Manual 5 e ao SOP-002.

---

6. Base ≠ produto final

A Base não é entregue ao cliente sem adaptação.

O projeto do cliente precisa receber:

- dados reais;
- conteúdo real;
- configurações próprias;
- identidade;
- WhatsApp/destinos;
- perguntas;
- perfis;
- Base de Conhecimento, quando aplicável;
- conteúdos de Blog/FAQ;
- ajustes visuais;
- capacidades compatíveis com o escopo.

Conteúdo demonstrativo ou genérico da Base não deve ser tratado como conteúdo final do cliente.

---

7. O que vem da Base e o que muda por cliente

No AS-IS, a estrutura parte da Base, mas o trabalho no cliente é principalmente de configuração/adaptação.

Itens mencionados como normalmente ajustados incluem:

- Google Search;
- Performance;
- avaliações do Google;
- depoimentos;
- FAQ;
- Base de Conhecimento;
- visual do Blog;
- configurações de SEO/conteúdo do Blog;
- WhatsApp;
- textos;
- perguntas;
- identidade visual;
- dados capturados no Chat/CTA.

A lista varia conforme escopo e capacidades presentes no projeto.

---

8. Módulos não aplicáveis

Módulos que não se aplicam ao cliente tendem a ser:

- ocultados;

- desativados;
- removidos da experiência administrativa do projeto;

sem que isso implique necessariamente apagar o código da Base.

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não foi identificado um procedimento universal único para remover fisicamente código de módulos não aplicáveis.

A prática observada privilegia personalização/ocultação/desativação.

Este Manual não transforma isso em política técnica de dead-code elimination.

PARTE II — MOMENTO E ORGANIZAÇÃO DO ADMIN

9. Admin em paralelo com a revisão do institucional

O Admin não precisa estar totalmente concluído para o site público ser enviado à primeira revisão do cliente.

Fluxo observado:

```
institucional suficientemente pronto
→ publicação temporária / revisão do cliente
→ Dev continua configuração do Admin
→ feedback do cliente ocorre em paralelo
```

Isso permite aproveitar o tempo de revisão externa para concluir a camada operacional.

---

10. Configuração módulo por módulo

O Dev trabalha no Admin de forma modular.

Ele tende a:

1. abrir o módulo;
2. ajustar o que é específico do cliente;
3. conferir dados/configurações;
4. testar a funcionalidade relevante;
5. seguir para o próximo módulo.

O QA completo do Admin pertence ao Manual 6, mas este Manual registra o comportamento esperado de cada capacidade.

---

11. Personalização sem reconstrução integral

A lógica corrente é aproveitar a capacidade já existente e adaptar:

- dados;
- conteúdo;
- identidade;
- destinos;
- permissões;
- configurações.

Não há evidência de que todo módulo deva ser refeito por cliente.

PARTE III — BLOG

12. Papel do Blog

O Blog faz parte das capacidades recorrentes da Base.

Há duas superfícies distintas:

1. Blog público;
2. editor/gestão no Admin.

---

13. Blog público

No AS-IS, a adaptação do Blog público tende a ser principalmente visual.

Itens citados:

- navbar;
- tipografia;
- tamanho de texto;
- cores;
- espaçamentos.

A intenção é manter o Blog coerente com a identidade do site institucional.

---

14. Blog no Admin

O editor tende a permanecer estruturalmente como na Base.

O que pode mudar por projeto inclui:

- configurações;
- cálculos ligados a SEO;
- conteúdo;
- tema/especialidade;
- integrações com Base de Conhecimento, quando aplicável.

Este Manual não afirma que todos os projetos possuem exatamente o mesmo editor ou a mesma configuração histórica.

---

15. Relação Blog × Chatbot

Quando o Chatbot está ativo, posts publicados podem ser sincronizados para a Base de Conhecimento.

No fluxo observado:

```
Blog publicado
→ conteúdo selecionado/sincronizado
→ knowledge_base
→ knowledge_chunks
→ recuperação pelo chatbot
```

O Blog não é lido integralmente em tempo real pelo chatbot a cada pergunta.

Há uma etapa de materialização/ingestão na Base de Conhecimento.

---

16. NORMA TÉCNICA VIGENTE relacionada ao Blog

A autoridade atual de SEO/AEO/GEO é `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Para páginas públicas indexáveis, o padrão preserva arquitetura server-first/SSR e determina, entre outros requisitos aplicáveis ao Blog:

- listagem e artigos com conteúdo principal disponível no HTML inicial;
- metadados/canonical/status por URL;
- `Article` ou `BlogPosting` coerente em artigos individuais;
- breadcrumb em artigos individuais;
- publicação pelo Admin sem exigir rebuild manual para cada artigo;
- áreas privadas/Admin fora do sitemap e com `noindex` quando aplicável.

Essas exigências pertencem à **camada normativa**. O restante deste Manual registra o papel operacional do Blog no AS-IS.

17. Finalidade

Formulários, CTAs e fluxos de agendamento/captação alimentam a camada de Leads do Admin.

O objetivo funcional é garantir que uma ação pública relevante resulte em registro administrativo verificável.

---

18. Teste ponta a ponta de Lead

O Dev realiza o seguinte fluxo:

1. aciona formulário/CTA/agendamento;
2. preenche os dados;
3. confere mensagem de sucesso ou erro;
4. entra no Admin;
5. verifica se o Lead foi capturado;
6. confere nome;
7. confere origem/trajeto;

8. confere especialidade/opção escolhida.

Esse teste é importante porque não valida apenas o frontend: valida a chegada do dado ao Admin.

---

19. Integração externa no fluxo de Leads

Não foi relatada integração externa adicional obrigatória para o fluxo padrão de captação além do registro no Admin.

Isso não significa que nenhuma integração possa existir em um projeto específico.

Significa apenas que o AS-IS consolidado não sustenta tratar CRM externo, automação de marketing ou outro serviço como requisito universal da captação.

---

20. Dados de teste

O Dev cria Leads e outros registros falsos durante validações.

Esses dados devem ser limpos antes da entrega do Admin.

O procedimento final de limpeza e entrega pertence ao Manual 7/SOP-012.

PARTE V — CHAT / CTA / WHATSAPP

21. Chat/CTA persistente

O Chat/CTA é uma capacidade adaptada ao projeto.

Por cliente, o Dev ajusta principalmente:

- WhatsApp/destino;
- textos;
- perguntas;
- identidade visual;
- dados capturados.

---

22. Validação do Chat

O Dev inicia uma conversa e verifica se ela aparece corretamente no módulo de atendimento do Admin.

Quando o fluxo envolve Lead, também confere a ligação entre conversa e dados capturados.

---

23. WhatsApp

O WhatsApp aparece como:

- destino de CTA;
- canal de encaminhamento;
- fallback do Chatbot quando configurado.

Ele não é obrigatório em toda execução da capacidade.

No Chatbot observado, a ausência de número válido não impede o funcionamento do assistente; apenas impede que ele ofereça aquele canal.

---

24. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — canal universal

Não há evidência de que todo cliente precise usar o mesmo canal ou número, nem de que todo fluxo de atendimento deva terminar em WhatsApp.

A configuração é específica do projeto.

PARTE VI — CHATBOT COM IA

25. O Chatbot é uma capacidade operacional real

A VAL-002 confirmou que o Chatbot herdado da Base Admin não é apenas um componente visual.

Ele combina:

- chat público;
- captura de visitante;
- integração com Lead;
- Base de Conhecimento;
- recuperação de contexto;
- IA;
- regras de resposta;
- atendimento humano;
- WhatsApp condicional;
- feedback;
- benchmark técnico;
- acompanhamento pelo Admin.

---

26. Arquitetura AS-IS do conhecimento

O fluxo observado é:

```
conteúdo do site
+ FAQ
+ Blog
+ entradas manuais
+ feedback convertido em conhecimento
        ↓
knowledge_base
        ↓
knowledge_chunks
        ↓
busca de contexto
        ↓
system prompt
        ↓
Lovable AI Gateway
        ↓
resposta
```

O site **não é crawleado integralmente em tempo real a cada pergunta**.

O conteúdo relevante é materializado/ingerido na Base de Conhecimento.

---

27. Origens de conhecimento

Foram observadas quatro origens principais no Admin:

- Manual;
- Blog sincronizado;
- FAQ sincronizada;
- feedback convertido em conhecimento.

Além disso, conteúdo real das páginas pode ser materializado em entradas relacionadas ao site.

---

28. Base de Conhecimento

A Base de Conhecimento mantém entradas editáveis e trechos derivados utilizados na recuperação.

No nível necessário ao método:

- `knowledge_base` guarda entradas;
- `knowledge_chunks` guarda trechos derivados;
- a busca recupera os trechos relevantes;
- o prompt recebe o contexto recuperado;
- a IA responde com base nesse contexto.

Não é necessário, neste Manual, mapear cada arquivo ou função da implementação.

---

29. Limites do assistente

Na implementação observada, o prompt impõe regras como:

- responder com base no contexto recuperado;
- não completar lacunas com conhecimento próprio;
- não diagnosticar;
- não indicar ou ajustar medicação;
- não interpretar exames;
- não inventar preços;
- não inventar horários;
- não inventar convênios;
- não inventar endereços;
- não inventar profissionais;
- encaminhar dúvida clínica para avaliação profissional;
- oferecer atendimento humano quando a informação não estiver confirmada.

Esses limites observados refletem o projeto/base inspecionados.

Eles não devem ser tratados como lista universal imutável para todo setor ou tipo de cliente.

---

30. Captura de dados e Lead

Para iniciar uma conversa no fluxo observado, são exigidos:

- nome;
- telefone válido.

A conversa cria ou vincula um **Lead real**.

Depois, pode ser enriquecida com:

- interesse;
- notas.

A integração Chatbot → Lead é parte funcional da capacidade.

---

31. Atendimento humano

Quando um atendente assume a conversa:

- a IA deixa de responder;
- o fluxo passa para atendimento humano.

O visitante também pode solicitar atendimento humano.

O Admin permite assumir e encerrar a conversa.

---

32. Fallbacks

Foram observados fallbacks para situações como:

- rate limit;
- falta de créditos;
- falha técnica.

Em falha técnica, o Chatbot pode oferecer encaminhamento para atendimento humano.

Este Manual não converte esses detalhes em SLA ou política universal.

---

33. Configuração do Chatbot por cliente

O conteúdo demonstrativo da Base não deve ser entregue como conteúdo real do cliente.

O setup observado inclui, conforme aplicável:

1. revisar/substituir Base de Conhecimento;
2. incluir conteúdo real das páginas;
3. revisar FAQ;
4. sincronizar FAQ quando aplicável;
5. revisar Blog;
6. sincronizar Blog quando aplicável;
7. configurar WhatsApp/destino;
8. ajustar textos;
9. ajustar perguntas;
10. ajustar identidade;
11. confirmar secret necessário ao serviço de IA;
12. testar captura de dados;
13. testar criação do Lead;
14. testar conversa;
15. testar ausência de resposta confirmada;
16. testar handoff;
17. conferir conversa no Admin.

O procedimento executável será formalizado no SOP-006.

---

34. Benchmark

A Base observada possui benchmark técnico do Chatbot.

Ele avalia perguntas de referência por critérios como:

- coerência;
- completude;
- compliance;
- score geral;
- flags;
- latência.

Limite documental

A existência do benchmark **não substitui** o teste manual do Dev.

Também não deve ser tratada como prova de que todos os projetos históricos executaram o benchmark da mesma forma.

---

35. Modelo de IA

Na versão inspecionada, havia modelo específico configurado no Lovable AI Gateway.

Esse detalhe é de implementação atual e pode evoluir.

Regra documental

O nome do modelo não é requisito imutável do Método WSS.

O que pertence ao método é a capacidade e sua validação, não a permanência de uma versão específica do modelo.

PARTE VII — AUTENTICAÇÃO, PERFIS E CAPABILITIES

36. Duas contas internas de teste

O Dev cria duas contas internas de teste:

- perfil/tag de cliente/equipe;
- perfil/tag Webstar.

As credenciais internas ficam registradas no Flow.

---

37. Objetivo do teste de perfis

O Dev verifica se cada perfil acessa somente o que deveria.

No AS-IS funcional:

- perfil Webstar possui capacidades administrativas mais amplas;
- perfil Equipe/cliente possui capacidades operacionais limitadas ao que deve utilizar.

A validação técnica de autorização server-side, RLS e escalonamento pertence ao Manual 6.

---

38. Presets observados

A validação técnica identificou presets como:

- `admin_webstar`;
- `admin_equipe`.

Também foram testados:

- autenticado sem papel;
- `anon`.

Neste Manual, o interesse é funcional: diferenciar responsabilidades e superfícies.

Os detalhes de hardening pertencem ao Manual 6.

---

39. Base de Conhecimento como capacidade Webstar

Na Base inspecionada, a capability de gestão da Base de Conhecimento/IA faz parte do perfil Webstar e não do perfil Equipe.

Assim:

- Webstar administra a camada de conhecimento/IA;
- equipe do cliente mantém capacidades operacionais permitidas;

- o cliente não deve ser presumido como administrador da Base de Conhecimento apenas porque possui acesso ao Admin.

---

40. Criação da conta do cliente

Na entrega:

1. perfis internos já foram testados;
2. o Dev gera um link de criação de conta;
3. o cliente cria seu próprio acesso.

O link pode ser enviado pelo Dev ou por Felipe.

A entrega do Admin pertence ao Manual 7.

---

41. Manual de uso do Admin

Capacidade da geração atual

A geração atual do Admin pode incluir um módulo de **Manual**, com capítulos organizados por perfil. Essa capacidade é posterior a alguns projetos já entregues/apresentados e não deve ser projetada retroativamente como se sempre tivesse existido.

A demonstração do funcionamento em reunião continua fazendo parte da prática observada.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO — obrigatoriedade universal**
>
> A existência do módulo de Manual na geração atual não comprova que um manual formal seja obrigatório e idêntico em toda entrega, inclusive em projetos históricos.

Este Manual não cria obrigação universal retroativa.

PARTE VIII — INTEGRAÇÕES E ANALYTICS

42. Integrações no AS-IS

A descoberta humana não identificou um conjunto amplo de integrações externas obrigatórias além das capacidades já citadas.

Integrações relevantes observadas incluem, conforme aplicável:

- Supabase/Lovable Cloud;
- Lovable AI Gateway;
- WhatsApp;
- Search Console;
- serviços relacionados às capacidades da Base.

---

43. Search Console

AS-IS da responsabilidade do Dev

O Dev participa da configuração do Search Console quando aplicável, com apoio de Codex/Lovable em alguns casos.

Isso descreve **quem executa no fluxo observado**, não a obrigatoriedade normativa do recurso.

> **NORMA TÉCNICA VIGENTE**
>
> O `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md` estabelece **Google Search Console configurado na publicação** como requisito WSS.

Consequentemente:

- o requisito deve ser cumprido no go-live;
- a pessoa responsável pela execução pode variar conforme o projeto;
- não se deve transformar o requisito WSS em atribuição pessoal universal do Dev.

44. Analytics e Tag Manager

No relato do Dev:

- Analytics;
- Google Tag Manager;

não são atribuição normal dele no fluxo descrito.

Isso não significa que a WSS não os utilize em nenhum projeto.

Significa apenas que não devem ser documentados como responsabilidade universal do Dev neste método AS-IS.

45. NORMA TÉCNICA VIGENTE

O padrão canônico vigente relacionado a arquitetura, SEO, AEO e GEO é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Ele pode impor requisitos específicos de indexação, rastreamento, Blog, áreas privadas, Search Console, performance e arquitetura. GA4 permanece **recomendado, não obrigatório universal**; ferramentas pagas permanecem sem seleção definitiva.

Este Manual não substitui essa norma e não a utiliza para reescrever o AS-IS da responsabilidade do Dev.

PARTE IX — LIMITES WEBSTAR × EQUIPE DO CLIENTE

46. Separação funcional

A Base atual sustenta uma separação entre:

- capacidades Webstar;
- capacidades da equipe do cliente.

Isso é coerente com a existência de perfis distintos.

Exemplo confirmado:

- gestão da Base de Conhecimento/IA → Webstar;
- Leads/atendimento/conteúdo permitido → equipe, conforme configuração.

---

47. Não confundir UI com autorização

Este Manual documenta a separação funcional.

Mas esconder uma opção da interface não é prova suficiente de segurança.

A validação de:

- autorização server-side;
- RLS;
- policies;
- escalonamento de privilégio;
- acesso de `anon`;
- exposição de PII;

pertence ao Manual 6.

PARTE X — PORTABILIDADE DA BASE PARA PROJETOS EXISTENTES

48. Não existe sincronização automática contínua

O recorte AS-IS ficou suficientemente esclarecido:

> não existe sincronização automática contínua Base ↔ projetos/sites existentes.

Também não existe simples substituição integral do Admin antigo pelo atual.

---

49. Portabilidade incremental e adaptativa

O comportamento observado na portabilidade Base → projeto é:

```
capacidade evolui na Base
→ comparar origem e destino
→ identificar delta funcional relevante
→ identificar dependências
→ portar dependências necessárias
→ adaptar capacidade ao destino
→ preservar particularidades locais
→ estabilizar
→ validar
```

Também pode ocorrer o fluxo inverso de forma **controlada**:

```
melhoria/correção validada em um projeto
→ avaliar se é reutilizável
→ remover dados e regras específicas do cliente
→ comparar com a Base vigente
→ avaliar risco/regressão
→ incorporar à Base quando aprovado
→ validar a Base
```

Isso não constitui sincronização automática. A Base e os projetos podem alimentar a evolução um do outro apenas após avaliação e validação humana.

A Base funciona como referência funcional reutilizável, não como autoridade absoluta para sobrescrever cada linha do destino.

---

50. Princípio comparativo

Em projeto antigo, a pergunta correta não é:

> “como substituir tudo pela Base nova?”

A lógica observada é:

> “qual capacidade falta, quais dependências ela exige e o que o destino já faz melhor e precisa ser preservado?”

---

51. Equivalentes, deltas e funcionalidade nova

Uma comparação real Base × Guilherme Maia demonstrou três categorias úteis:

51.1 Já existe

Capacidade já presente no destino.

Exemplo observado em Leads:

- CRUD;
- busca/filtros;
- score/urgência;
- ficha do Lead;
- timeline;
- exportação.

Esses pontos não precisam ser refeitos apenas para “igualar a Base”.

51.2 Delta útil

Capacidade existente na Base que pode representar ganho real no destino.

Exemplos observados:

- KPIs;
- gráficos;
- ações em lote.

51.3 Funcionalidade nova

Ideia que não é simples paridade entre Base e destino.

Exemplo real: determinada “Jornada do paciente” foi explicitamente deixada fora porque representaria feature nova.

Regra AS-IS

Não confundir portabilidade de capacidade com criação de funcionalidade nova.

Quando a comparação revela escopo novo, a decisão volta para gestão/escopo.

52. Dependências primeiro

Uma capacidade pode depender de:

- banco;
- tabelas;
- migrations;
- autenticação;
- perfis;
- dados;
- componentes;
- serviços;
- configurações.

A prática observada é identificar e portar/adaptar dependências antes de considerar a capacidade concluída.

53. Preservar o que o destino faz melhor

A Base não deve apagar automaticamente melhorias locais.

Quando o projeto antigo já possui uma solução:

- funcional;
- superior;
- específica;
- compatível com o cliente;

ela pode ser preservada.

Esse princípio é central para evitar regressão por “sincronização” indiscriminada.

54. Caso real — Dr. Guilherme Maia

O projeto do Dr. Guilherme Maia é a principal evidência de portabilidade evolutiva.

Ele representa uma geração anterior baseada em stack diferente dos projetos atuais e com Admin próprio já desenvolvido.

O caso mostra que atualização de legado exige:

- comparação;
- seleção;
- compatibilidade;

- preservação.

O estudo de caso completo pertence ao Manual 8.

PARTE XI — MIGRATIONS, DADOS E ROLLBACK

55. AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe hoje um procedimento recorrente, universal e separado para:

- migrations em toda atualização de Base;
- preservação de dados em todo cenário;
- rollback por categoria de mudança.

Isso é um achado do AS-IS.

---

56. O que este Manual pode afirmar

Quando uma capacidade depende de alteração estrutural:

- a dependência deve ser identificada;
- o impacto precisa ser considerado;
- o destino deve ser estabilizado e validado.

Mas este Manual **não inventa** um ritual universal de migration/backup/rollback que não foi observado como prática padronizada.

---

57. Relação com troubleshooting

O risco típico é:

> atualização da Base ameaça quebrar comportamento do legado.

O catálogo de troubleshooting correspondente pertence ao Manual 8.

O método de investigação e regressão pertence ao Manual 6.

PARTE XII — NORMA, AUSÊNCIAS E OPORTUNIDADES

58. NORMA TÉCNICA VIGENTE

Algumas capacidades da Base estão sujeitas a normas técnicas específicas, especialmente arquitetura pública/Blog, autenticação, SEO, segurança, privacidade e indexação.

Para SEO/AEO/GEO, vale o `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Legado e princípio de não regressão

A portabilidade incremental/adaptativa descrita neste Manual permanece válida, mas deve ser lida junto ao princípio normativo atual:

- projeto existente é medido contra o padrão vigente;
- divergência AS-IS do legado **não reduz a norma**;
- retrofit integral não é automático;
- risco, impacto, custo, contrato, prioridade e ciclo governam a estratégia/momento de remediação — não a classificação técnica da divergência.

Portanto, preservar o que o legado faz melhor não significa transformar uma divergência normativa em nova regra WSS.

A norma não deve reescrever retroativamente o AS-IS aqui descrito.

59. AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

Este Manual registra como ausências reais:

- não existe sincronização automática contínua Base ↔ projetos/sites;
- não existe substituição integral padrão do Admin em legado;
- não existe procedimento universal de migrations em toda portabilidade;
- não existe ritual universal de preservação de dados/rollback;
- não existe manual formal obrigatório do Admin em toda entrega;
- nem todo módulo aplicável/não aplicável é tratado por exclusão física de código;
- não existe conjunto amplo de integrações externas obrigatórias além das capacidades efetivamente observadas.

---

60. OPORTUNIDADE FUTURA — FORA DO ESCOPO

Podem existir oportunidades futuras de:

- formalizar versionamento/evolução da Base;
- formalizar matriz de compatibilidade;
- formalizar migrations/rollback;
- ampliar documentação de entrega do Admin;
- automatizar sincronização seletiva.

Nenhuma dessas soluções é desenhada neste Manual.

PARTE XIII — RELAÇÃO COM OS OUTROS MANUAIS

61. Relação com o Manual 2

O Manual 2 entrega:

- contexto;
- escopo;
- módulos/capacidades aplicáveis;
- restrições do cliente.

O Manual 4 explica como essas capacidades são materializadas/configuradas na Base/Admin.

---

62. Relação com o Manual 3

O Manual 3 é proprietário da construção visual.

Quando Blog, Chat ou outros módulos precisam de adaptação visual, o princípio visual vem do Manual 3.

Este Manual é proprietário da função/configuração do módulo.

---

63. Relação com o Manual 5

O Manual 4 explica **o que a Base e os módulos fazem**.

O Manual 5 explica **como operar Lovable, GitHub, ambiente local e Codex para implementar alterações**.

Não duplicar:

- clone;
- pull;
- commit;
- push;
- branch;
- conflito;
- rollback Git.

---

64. Relação com o Manual 6

Este Manual registra comportamentos esperados de:

- Leads;
- Chat;
- Chatbot;

- perfis;
- módulos.

O Manual 6 explica como testar, diagnosticar e validar tecnicamente essas capacidades.

Segurança detalhada também pertence ao Manual 6.

---

65. Relação com o Manual 7

Este Manual explica configuração e funcionamento do Admin.

O Manual 7 é proprietário de:

- limpeza final de dados;
- criação/envio do acesso;
- entrega;
- pós-publicação.

---

66. Relação com o Manual 8

O Manual 8 concentra:

- caso Guilherme Maia;
- troubleshooting de portabilidade;
- práticas depreciadas;
- anti-patterns;
- lições de Base/Admin.

PARTE XIV — SOPs DERIVADOS

67. SOP-005 — Configurar Admin e Módulos Aplicáveis

Este SOP consolida:

- identificação dos módulos aplicáveis;
- ajustes por cliente;
- configuração;
- testes funcionais básicos;
- perfis;
- evidências esperadas.

Leads/formulários ficam como subfluxo desse SOP, evitando fragmentação excessiva.

---

68. SOP-006 — Configurar e Validar Chatbot e Base de Conhecimento

O Chatbot merece SOP próprio por possuir:

- Base de Conhecimento;
- ingestão;
- captura de dados;
- Lead;
- regras de resposta;
- handoff;
- WhatsApp;
- benchmark;
- validações específicas.

---

69. SOP-007 — Portar Capacidade da Base para Projeto Legado

Este SOP transforma o princípio adaptativo em sequência executável:

```
inventariar
→ comparar
→ identificar equivalentes
→ identificar deltas
→ separar feature nova
→ mapear dependências
→ portar/adaptar
→ preservar local
→ validar
```

Ele não cria política universal de migration ou rollback além do que o AS-IS sustenta.

PARTE XV — CRITÉRIO DE CONCLUSÃO

70. Critério de conclusão deste Manual

Este Manual cumpre seu objetivo quando o leitor consegue responder:

1. o que é a Base Admin no método atual;
2. por que ela é usada;
3. o que tende a vir pronto;
4. o que muda por cliente;
5. quando o Admin é configurado;
6. como Blog, Leads, Chat/CTA e WhatsApp entram no fluxo;
7. como o Chatbot usa Base de Conhecimento;
8. como Chatbot e Lead se relacionam;
9. como funciona o handoff humano;
10. quem administra a camada de conhecimento;
11. como perfis Webstar e Equipe se diferenciam funcionalmente;
12. quais integrações são realmente relevantes no AS-IS;
13. por que legado não recebe substituição integral automática;
14. o que significa portabilidade incremental/adaptativa;
15. como distinguir equivalente, delta e feature nova;
16. quais ausências de padronização ainda existem;
17. quais assuntos devem ser buscados nos Manuais 5, 6, 7 e 8.

---

71. Fontes principais desta versão

Baseline canônico

1. `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
2. `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
3. `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
4. `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
5. `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

Evidência técnica complementar

- `WSS_VAL_002_BaseAdmin_Chatbot_v0.1.md`;
- `WSS_VAL_004_Troubleshooting_Estudos_de_Caso_v0.1.md`.

Evidência técnica relacionada, usada apenas para delimitar perfis/segurança

- `WSS_VAL_003_Auditorias_SEO_AEO_GEO_Seguranca_v0.1.md`.

Norma técnica vigente relacionada

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Versões anteriores/superadas de AS-IS, Matriz, Handoff e Plano não foram usadas como baseline ativo.

Status: FINAL — APROVADO

WEB STAR STUDIO

MANUAL

MÉTODO DE DESENVOLVIMENTO DE SITES

MANUAL 5
Desenvolvimento com Codex, Lovable e GitHub

VERSÃO
v1.1 FINAL

DATA
11/09/2026

STATUS
FINAL — APROVADO

MANUAL

MANUAL 5

Desenvolvimento com Codex, Lovable e GitHub

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
ESCOPO DOCUMENTAL	primeira documentação ampliada — P0 + P1 + P2 relevante
PROJETO	WSS — Método de Desenvolvimento de Sites
FORMALIZAÇÃO	Pacote Corretivo Pós-Entrega v1.1 aprovado pela Coordenação em 11/09/2026; versão canônica vigente, com v1.0 FINAL preservado como baseline histórico.

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Objetivo 7

2. Escopo e fronteira 7

3. Posição deste Manual no fluxo 8

PARTE I — INÍCIO TÉCNICO DO PROJETO 10

4. O projeto começa no Lovable 11

5. Por que o fluxo não começa localmente 11

6. Primeiros protótipos antes do GitHub/local 11

PARTE II — GITHUB E AMBIENTE LOCAL 13

7. Conexão ao GitHub 14

8. Clone local 14

9. Abrir o projeto nas ferramentas 15

PARTE III — BRANCH E ESTADO DO REPOSITÓRIO 16

10. main como prática predominante 17

11. Branch separada 17

12. Pull Requests e tags 18

PARTE IV — PULL, COMMIT E PUSH 19

13. Pull antes da rodada local 20

14. Rodada local 20

15. Validação antes de subir 20

16. Commit 21

17. Push 21

18. Ciclo resumido de uma rodada local 22

PARTE V — CODEX COMO DESENVOLVEDOR DIRETO 23

19. Papel do Codex 24

20. Fluxo Codex direto 24

21. Quando o Codex tende a ser escolhido 24

22. Codex para investigação 25

23. Codex e referências visuais 25

PARTE VI — CODEX CONDUZINDO O LOVABLE 26

24. Segunda modalidade do Codex 27

25. Quando esse modo faz sentido 27

26. O Codex como condutor, não como substituto do julgamento 27

27. Limites do modo Codex → Lovable 28

PARTE VII — LOVABLE COMO AMBIENTE DE IMPLEMENTAÇÃO 29

28. Uso corrente do Lovable 30

29. Limite visual do Lovable 30

PARTE VIII — CLAUDE 31

30. Papel do Claude 32

31. Claude no fluxo local 32

32. Codex × Claude não é divisão rígida 32

PARTE IX — MODELO HÍBRIDO 34

33. O método atual é híbrido 35

34. Três modelos operacionais 35

35. Critério prático de escolha 36

PARTE X — CONTINUIDADE DE CONTEXTO 37

36. Contexto persistente 38

37. Nova sessão 38

38. Arquivo inacessível 38

39. Contexto da tarefa 38

PARTE XI — LIMITAÇÃO DE ESCOPO 40

40. Regra prática 41

41. Por que isso importa no fluxo híbrido 41

42. Não existe trava universal 41

PARTE XII — EDIÇÃO SIMULTÂNEA, DIVERGÊNCIA E CONFLITO 42

43. Não há bloqueio formal 43

44. Prevenção prática 43

45. Quando ocorre conflito 43

46. “Origem da verdade” no AS-IS 43

47. Conflito não é motivo para reconstrução total 44

PARTE XIII — ROLLBACK E RECUPERAÇÃO 45

48. Formas observadas de rollback 46

49. Decisão de rollback 46

50. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — ritual de rollback 46

51. O que não deve ser inferido 46

PARTE XIV — CODEX PARA INSPEÇÃO E ARQUEOLOGIA TÉCNICA 48

52. Uso transferível 49

53. O que não deve virar etapa universal 49

PARTE XV — PONTOS SENSÍVEIS E CONTROLE HUMANO 50

54. Credenciais e secrets 51

55. Banco e tabelas 51

56. Mudança ampla 51

57. Áreas já aprovadas 51

PARTE XVI — PRÁTICAS DEPRECIADAS RELACIONADAS AO FLUXO TÉCNICO 53

58. Desenvolver praticamente tudo no Lovable 54

59. Começar fora do Lovable e integrar depois 54

PARTE XVII — NORMA, AUSÊNCIAS E OPORTUNIDADES 55

60. NORMA TÉCNICA VIGENTE 56

61. AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 56

62. OPORTUNIDADE FUTURA — FORA DO ESCOPO 57

PARTE XVIII — RELAÇÃO COM OS OUTROS MANUAIS 58

63. Relação com o Manual 1 59

64. Relação com o Manual 2 59

65. Relação com o Manual 3 59

66. Relação com o Manual 4 59

67. Relação com o Manual 6 60

68. Relação com o Manual 7 60

69. Relação com o Manual 8 60

PARTE XIX — SOP DERIVADO 61

70. SOP-002 — Iniciar Tecnicamente um Projeto WSS: Base → Lovable → GitHub → Ambiente Local 62

71. Procedimentos que permanecem incorporados ao Manual 62

PARTE XX — CASO REAL 63

72. Dr. Guilherme Mendonça 64

PARTE XXI — CRITÉRIO DE CONCLUSÃO 65

73. Critério de conclusão deste Manual 66

74. Fontes principais desta versão 66

1. Objetivo

Este Manual documenta a mecânica atual de desenvolvimento entre **Lovable, GitHub, ambiente local, Codex e Claude**.

Seu objetivo é permitir que outro desenvolvedor ou agente compreenda:

- como um projeto contextualizado passa do Lovable para GitHub e ambiente local;
- por que o fluxo atual é híbrido;
- quando o Dev trabalha diretamente no código;
- quando usa o Codex como operador/condutor do Lovable;
- como Claude entra no refinamento;
- como `main`, pull, commit e push são usados hoje;
- como as mudanças locais retornam ao Lovable;
- como o Dev lida com divergência e conflito;
- como o rollback ocorre no AS-IS;
- como contexto e escopo são preservados entre ferramentas;
- quais decisões continuam humanas;
- quais práticas ainda não possuem padronização formal.

Este Manual descreve **como as ferramentas são operadas e combinadas**.

Ele não ensina construção visual, configuração funcional do Admin, QA completo ou publicação.

---

2. Escopo e fronteira

2.1 O que entra neste Manual

Este Manual cobre:

1. início técnico do projeto após contextualização;
2. remix da Base no Lovable apenas como gatilho técnico;
3. primeiros protótipos;
4. conexão ao GitHub;
5. clone para ambiente local;
6. abertura e leitura do projeto por Codex/Claude;
7. branch `main` no AS-IS;
8. uso excepcional de branch separada;
9. pull;
10. commit;
11. push;
12. retorno da alteração ao Lovable;
13. trabalho local;

14. Codex como desenvolvedor direto;
15. Codex como condutor do Lovable;
16. Claude;
17. modelo híbrido;
18. escolha entre ferramentas;
19. continuidade de contexto;
20. limitação de escopo;
21. edição simultânea;
22. divergência/conflito;
23. rollback;
24. inspeção técnica com Codex;
25. limites humanos e ausências de padronização.

2.2 O que não entra neste Manual

Ficam fora:

- preparação do prompt master;
- hierarquia Contexto/Sitemap/Briefing;
- construção detalhada da Home;
- análise visual de referências;
- componentização e Design System;
- configuração detalhada da Base Admin;
- Chatbot/RAG;
- estratégia completa de testes;
- debugging em profundidade;
- segurança;
- SEO/AEO/GEO;
- domínio;
- publicação.

Esses assuntos pertencem aos outros Manuais.

---

3. Posição deste Manual no fluxo

Este Manual começa quando o projeto já possui:

- contexto suficientemente compreendido;
- prompt master;
- direção inicial;
- Base Admin escolhida como ponto de partida.

O recorte principal é:

```
Contexto pronto
→ Lovable
→ primeiros protótipos
→ GitHub
→ ambiente local
→ Codex / Claude
→ alterações
→ validação da rodada
→ commit / push
→ Lovable atualizado
→ nova rodada
```

O fluxo se repete durante o desenvolvimento.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE I — INÍCIO TÉCNICO DO PROJETO

4. O projeto começa no Lovable

No fluxo atual, um novo projeto normalmente começa no **Lovable**.

O Dev:

1. remixa a Base Admin;
2. envia o prompt master;
3. fornece documentos e referências;
4. usa os primeiros prompts para contextualizar o projeto;
5. produz a primeira base do institucional;
6. chega ao primeiro ou segundo protótipo.

A função e as capacidades da Base pertencem ao Manual 4.

Aqui interessa apenas sua função como **origem técnica do projeto corrente**.

---

5. Por que o fluxo não começa localmente

O Dev já experimentou iniciar fora do Lovable e trazer o projeto posteriormente.

Essa abordagem gerou dificuldades e não representa o fluxo normal atual.

Prática corrente

```
Lovable primeiro
→ GitHub depois
→ local em seguida
```

Portanto, o ambiente local é incorporado cedo, mas **não substitui o Lovable como origem normal do projeto**.

---

6. Primeiros protótipos antes do GitHub/local

O GitHub entra normalmente depois do primeiro ou segundo protótipo.

Isso significa que o Dev primeiro permite que o Lovable:

- materialize a Base;
- receba o contexto;
- estabeleça uma primeira versão.

Depois disso, o projeto passa para o fluxo híbrido.

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe um número rígido universal de prompts ou uma condição técnica automatizada para determinar o momento exato de conectar o GitHub.

A referência corrente é prática:

> depois do primeiro ou segundo protótipo, quando o projeto já possui base suficiente para seguir localmente.

PARTE II — GITHUB E AMBIENTE LOCAL

7. Conexão ao GitHub

Depois da primeira base:

1. o projeto é conectado ao GitHub;
2. o repositório é criado;
3. o projeto é clonado para a máquina do Dev.

O GitHub passa a funcionar como ponte entre:

- Lovable;
- ambiente local;
- Codex/Claude.

Fluxo:

```
Lovable
   ↓
GitHub
   ↓
Projeto local
   ↓
Codex / Claude
   ↓
GitHub
   ↓
Lovable
```

---

8. Clone local

O Dev clona o projeto logo após a criação/conexão do repositório.

A partir daí, o repositório local se torna a base para:

- leitura de arquitetura;
- implementação direta;
- refinamento;
- investigação;
- testes;
- trabalho com Codex;
- trabalho com Claude.

Fronteira

Este Manual documenta a sequência operacional.

Ele não prescreve um comando Git específico ou uma estrutura de pastas universal que não esteja consolidada nas fontes.

9. Abrir o projeto nas ferramentas

Depois do clone, Codex e Claude podem ler:

- código;
- arquivos `.md`;
- estrutura;
- configurações;
- commits;
- demais artefatos acessíveis.

Ao iniciar uma nova sessão, o Dev costuma pedir que a ferramenta:

- leia o projeto;
- entenda a estrutura;
- leia os documentos;
- entenda WSS, Dev e cliente;
- apresente um resumo antes de continuar.

A preparação do contexto pertence ao Manual 2.

Aqui, essa prática funciona como **gate operacional antes de editar**.

PARTE III — BRANCH E ESTADO DO REPOSITÓRIO

10. `main` como prática predominante

No AS-IS, a prática predominante é trabalhar na:

`main`

O motivo declarado é a compatibilidade com o Lovable.

Regra documental

Isso deve ser registrado como:

AS-IS — prática atual

e não como:

NORMA UNIVERSAL DE GIT DA WSS

Este Manual não cria uma política normativa de branching.

---

11. Branch separada

O Dev pode usar branch separada quando existe uma mudança radical.

Exemplos declarados:

- alteração ampla de uma página;
- mudança relevante de fluxo;
- intervenção que justifica isolamento.

Depois, a mudança é integrada.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe hoje regra formal universal determinando:

- quando exatamente abrir branch;
- nomenclatura;
- duração;
- critério de merge;
- reviewer;
- Pull Request obrigatório.

A decisão é contextual.

---

12. Pull Requests e tags

Não foi relatado uso recorrente de:

- Pull Requests;
- tags;

como parte normal do fluxo corrente.

Quando uma rodada usa branch ou Pull Request, o Codex costuma criar esses artefatos sob instrução/supervisão do Dev; o Dev não mantém a abertura manual de PR como etapa fixa.

Isso não significa que branch ou PR sejam obrigatórios nem estabelece convenção de nomenclatura.

Significa apenas que **não pertencem ao caminho AS-IS predominante documentado como requisito fixo**.

PARTE IV — PULL, COMMIT E PUSH

13. Pull antes da rodada local

Antes de iniciar uma rodada de trabalho local, o Dev faz pull para trazer a versão mais recente disponível.

Objetivo:

- evitar começar sobre uma cópia desatualizada;
- incorporar mudanças feitas pelo Lovable ou já enviadas ao GitHub;
- reduzir risco de conflito.

Prática atual

O Dev declarou que não inicia nova rodada local sem antes atualizar a cópia local.

---

14. Rodada local

Depois do pull, o Dev pode trabalhar com:

- Codex;
- Claude;
- editor/terminal;
- navegador;
- outras ferramentas necessárias.

A rodada pode conter:

- implementação;
- refinamento;
- investigação;
- correção;
- adaptação;
- testes relacionados à alteração.

O QA completo do projeto pertence ao Manual 6.

---

15. Validação antes de subir

As alterações locais são commitadas e enviadas depois que o Dev entende que podem subir.

Isso pressupõe algum nível de validação da rodada.

> **Importante**
>
> Este Manual não converte essa validação em uma suíte universal.

O tipo de teste depende do que foi alterado.

A estratégia completa pertence ao Manual 6.

---

16. Commit

O commit registra a mudança local no histórico do projeto.

No AS-IS, ele ocorre como parte natural do fluxo de sincronização.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> As fontes não sustentam uma convenção universal obrigatória para:

- mensagem;
- granularidade;
- prefixo;
- tamanho do commit;
- número de commits por rodada.

Este Manual não cria essa convenção.

---

17. Push

Após a validação local, o Dev faz push.

Depois do push:

- a mudança aparece no histórico do Lovable;
- o preview é atualizado pouco depois.

Assim, o GitHub funciona de fato como ponte de retorno ao Lovable.

---

18. Ciclo resumido de uma rodada local

```
confirmar contexto
↓
pull
↓
implementar / investigar
↓
validar a alteração
↓
commit
↓
push
↓
Lovable recebe atualização
↓
preview reflete a mudança
↓
nova conferência quando necessário
```

Esse ciclo é uma das unidades operacionais centrais do modelo híbrido.

PARTE V — CODEX COMO DESENVOLVEDOR DIRETO

19. Papel do Codex

O Codex é uma das ferramentas centrais do método atual.

Usos recorrentes:

- desenvolvimento direto no repositório;
- reprodução visual com alta fidelidade;
- implementação pesada;
- investigação;
- leitura da arquitetura;
- análise de referências;
- análise/catalogação de assets;
- geração contextualizada de imagens;
- testes solicitados pelo Dev;
- inspeções técnicas.

Se o Dev precisasse escolher apenas uma ferramenta no fluxo atual, tenderia ao Codex.

---

20. Fluxo Codex direto

Modelo:

```
GitHub
→ projeto local
→ Codex lê contexto/código
→ Codex investiga/implementa
→ validação
→ commit/push
→ GitHub
→ Lovable
```

Nesse modo, o Codex atua **diretamente na base de código**.

---

21. Quando o Codex tende a ser escolhido

O Dev tende a preferir Codex quando:

- busca maior fidelidade a uma referência;
- a implementação é mais pesada;
- precisa trabalhar diretamente no código;
- precisa investigar;
- precisa entender uma dependência;

- quer uma análise técnica;
- precisa testar de forma mais ampla;
- o contexto já está concentrado no Codex.

---

22. Codex para investigação

O Codex também pode ser usado antes de editar.

Ele pode:

- ler código;
- localizar arquivos;
- comparar implementações;
- entender dependências;
- investigar efeito;
- identificar causa provável;
- mapear impacto.

Essa capacidade é importante porque o Dev não costuma iniciar uma investigação lendo manualmente linha por linha.

A IA pode fazer a primeira varredura técnica, enquanto o Dev julga os resultados.

---

23. Codex e referências visuais

Quando uma referência é complexa, o Codex pode:

- estudar estrutura;
- investigar comportamento;
- comparar screenshots;
- ajudar a reproduzir efeito;
- analisar código disponível/publicamente observável quando aplicável.

A estratégia visual completa pertence ao Manual 3.

PARTE VI — CODEX CONDUZINDO O LOVABLE

Esta parte trata do **pós-criação**. A origem técnica normal do projeto permanece o remix da Base Admin no Lovable, que traz a estrutura inicial do projeto, inclusive backend/banco/auth herdados da Base.

24. Segunda modalidade do Codex

Depois dessa criação inicial, o Codex não atua apenas no código local.

Em alguns casos, ele também pode **investigar/conduzir a tarefa e operar/orquestrar o Lovable pelo navegador/chat**.

Modelo:

```
Codex
→ navegador
→ Lovable
→ projeto
→ chat
→ prompt
→ resposta/implementação
→ preview
→ nova orientação
```

---

25. Quando esse modo faz sentido

Essa modalidade tende a fazer sentido, no pós-criação, quando:

- o contexto já está no Codex;
- o Codex funciona como primeira camada de investigação/condução;
- a tarefa é mais adequada ao Lovable;
- backend/banco/Supabase exigem ou se beneficiam da integração do Lovable/Lovable Cloud;
- o Dev quer manter a condução concentrada no Codex;
- o Codex precisa acompanhar resposta e preview do Lovable.

---

26. O Codex como condutor, não como substituto do julgamento

Nesse modo, o Codex pode:

- abrir o Lovable;
- acessar o projeto;
- conversar no chat;
- enviar instruções;
- acompanhar respostas;
- observar preview;

- orientar correções.

Mas o Dev continua relevante em pontos como:

- credenciais;
- chaves;
- tabelas;
- risco;
- escopo;
- decisão de aceitar uma mudança;
- validação do resultado.

---

27. Limites do modo Codex → Lovable

O fato de o Codex conseguir operar o navegador não significa que qualquer tarefa deva ser delegada sem supervisão.

O risco aumenta quando a tarefa envolve:

- banco;
- autenticação;
- migrations;
- secrets;
- mudanças amplas;
- áreas já aprovadas.

A decisão de executar e os controles de risco continuam humanos.

PARTE VII — LOVABLE COMO AMBIENTE DE IMPLEMENTAÇÃO

28. Uso corrente do Lovable

Mesmo depois do clone, o Lovable continua fazendo parte do desenvolvimento.

Usos recorrentes:

- backend/banco/Supabase;
- bugs funcionais;
- pequenas alterações finais;
- recursos que a integração nativa torna mais simples;
- implementação conduzida pelo próprio Dev;
- implementação conduzida pelo Codex via chat.

---

29. Limite visual do Lovable

O Dev tende a evitar o Lovable como principal ferramenta para reprodução visual complexa.

Isso não significa incapacidade técnica da plataforma.

É uma preferência AS-IS baseada na experiência do Dev com:

- fidelidade;
- complexidade;
- refinamento visual.

Nesses casos, Codex e Claude tendem a assumir papel maior.

PARTE VIII — CLAUDE

30. Papel do Claude

Claude é usado principalmente para:

- refinamento visual;
- tipografia;
- design;
- efeitos;
- adaptação de interface;
- acabamento;
- versão mobile;
- refinamento de bases produzidas por Codex.

---

31. Claude no fluxo local

Modelo típico:

```
projeto local
→ Claude lê código/contexto
→ refinamento
→ validação
→ commit/push
```

Claude pode trabalhar sobre uma base:

- iniciada pelo Lovable;
- desenvolvida pelo Codex;
- já existente no projeto.

---

32. Codex × Claude não é divisão rígida

Não existe uma fronteira absoluta.

A divisão declarada é pragmática:

- Codex tende a ser preferido para fidelidade, investigação e implementação pesada;
- Claude tende a ser preferido para refinamento visual e acabamento.

Mas ambos podem atuar em tarefas sobrepostas.

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe matriz obrigatória:

```
tarefa X = sempre Codex
tarefa Y = sempre Claude
```

A escolha depende do resultado, contexto e necessidade.

PARTE IX — MODELO HÍBRIDO

33. O método atual é híbrido

O modelo atual pode combinar, no mesmo projeto:

- Lovable;
- Codex direto;
- Codex operando Lovable;
- Claude;
- intervenção manual do Dev.

Não há expectativa de que toda implementação passe por um único ambiente.

---

34. Três modelos operacionais

Modelo A — Lovable primeiro, Codex direto depois

```
Lovable
→ GitHub/local
→ Codex
→ GitHub
→ Lovable
```

Modelo B — Codex conduz Lovable

```
Codex
→ navegador
→ Lovable
→ implementação
→ preview
→ correção
```

Modelo C — Híbrido dentro da mesma rodada

```
Codex investiga
→ Lovable resolve backend
→ Codex/Claude refinam frontend
→ GitHub sincroniza
→ navegador valida
```

O modelo escolhido depende do tipo de tarefa.

---

35. Critério prático de escolha

Os fatores observados incluem:

- natureza da tarefa;
- integração necessária;
- complexidade;
- fidelidade desejada;
- onde o contexto já está carregado;
- camada afetada;
- risco;
- qualidade da resposta anterior;
- facilidade de validação.

Essa decisão permanece parcialmente tácita.

PARTE X — CONTINUIDADE DE CONTEXTO

36. Contexto persistente

O projeto mantém contexto através de:

- código;
- documentos;
- `contexto.md`;
- `design.md`;
- commits;
- histórico disponível;
- outros `.md`.

O papel de cada artefato foi documentado no Manual 2.

---

37. Nova sessão

Ao iniciar nova sessão, o Dev costuma pedir:

1. leitura do projeto;
2. leitura dos documentos;
3. entendimento de WSS, Dev e cliente;
4. resumo antes de continuar.

Essa etapa funciona como gate de entendimento.

---

38. Arquivo inacessível

Quando a ferramenta não possui acesso a um documento necessário, o Dev reenvia o arquivo.

O método atual não pressupõe que toda IA sempre terá acesso a todo contexto histórico.

---

39. Contexto da tarefa

Além do contexto persistente, o prompt atual delimita:

- módulo;
- página;
- seção;
- bug;
- breakpoint;
- referência;

- área que não deve mudar.

Esse recorte reduz risco de uma IA usar o contexto global para alterar partes não relacionadas.

PARTE XI — LIMITAÇÃO DE ESCOPO

40. Regra prática

O Dev procura limitar cada pedido ao menor recorte razoável.

Exemplos:

- alterar apenas mobile;
- não tocar desktop já validado;
- alterar apenas uma seção;
- não mexer no backend;
- corrigir somente um módulo do Admin;
- preservar restante do institucional.

---

41. Por que isso importa no fluxo híbrido

Como várias ferramentas podem editar o mesmo projeto, escopo excessivamente aberto aumenta risco de:

- mudança colateral;
- perda de fidelidade;
- regressão;
- conflito;
- dificuldade de identificar causa.

A limitação de escopo é, portanto, uma prática operacional importante.

---

42. Não existe trava universal

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe mecanismo técnico universal que impeça Lovable, Codex ou Claude de tocar em áreas fora do pedido.

A proteção depende de:

- instrução;
- contexto;
- revisão;
- testes.

PARTE XII — EDIÇÃO SIMULTÂNEA, DIVERGÊNCIA E CONFLITO

43. Não há bloqueio formal

O fluxo atual não possui mecanismo formal que impeça:

- edição no Lovable;
- edição local;

ao mesmo tempo.

Portanto, existe risco de divergência.

---

44. Prevenção prática

A prevenção observada passa principalmente por:

- pull antes da rodada local;
- atenção ao estado mais recente;
- evitar começar sobre versão desatualizada;
- validar antes de subir;
- entender o que mudou em cada lado.

Não existe protocolo formal de lock.

---

45. Quando ocorre conflito

O Dev procura entender:

- qual mudança ocorreu;
- qual versão está funcional;
- qual versão causa menos dano;
- qual estado precisa ser preservado.

Frequentemente, quando existe uma versão publicada/Lovable claramente funcional, ela é tratada como referência estável.

---

46. “Origem da verdade” no AS-IS

As fontes não sustentam uma única regra abstrata como:

```
GitHub sempre vence
```

ou:

```
Lovable sempre vence
```

O comportamento observado é pragmático:

> preservar o estado funcional e minimizar dano.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe hoje uma política universal formal de “source of truth” para todo cenário de conflito.

---

47. Conflito não é motivo para reconstrução total

Quando existe divergência, a resposta atual é investigar e recuperar o estado correto.

Não se deve concluir automaticamente que:

- a Base deve sobrescrever;
- o local deve sobrescrever;
- o Lovable deve sobrescrever tudo.

O objetivo é preservar trabalho válido.

PARTE XIII — ROLLBACK E RECUPERAÇÃO

48. Formas observadas de rollback

O retorno a uma versão anterior pode ocorrer por:

- histórico do Lovable;
- commit anterior;
- build anterior;
- instrução à IA local para restaurar uma versão anterior.

---

49. Decisão de rollback

A decisão depende do problema.

Pode haver:

- correção localizada;
- retorno parcial;
- retorno a versão funcional anterior.

A contingência de produção pertence ao Manual 7.

Aqui interessa o rollback durante desenvolvimento.

---

50. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — ritual de rollback

Não existe hoje um procedimento formal universal por categoria de mudança.

Não há, nas fontes, tabela como:

```
frontend → rollback A
backend → rollback B
migration → rollback C
```

Este Manual não cria esse ritual.

---

51. O que não deve ser inferido

Não afirmar que o método atual exige sempre:

- revert Git específico;
- tag prévia;

- branch de recuperação;
- backup formal por alteração;
- Pull Request;
- aprovação em code review.

Essas práticas podem ser úteis em outros contextos, mas não foram consolidadas como AS-IS universal.

PARTE XIV — CODEX PARA INSPEÇÃO E ARQUEOLOGIA TÉCNICA

52. Uso transferível

O Codex pode ser usado para inspeção técnica de um projeto existente.

Exemplos:

- localizar capacidade;
- mapear dependências;
- entender arquitetura;
- comparar origem/destino;
- investigar implementação;
- levantar riscos.

Isso é útil em desenvolvimento e manutenção.

---

53. O que não deve virar etapa universal

As ARQ-001 a ARQ-006 foram criadas para **descobrir e documentar o próprio Método WSS**.

Essa metodologia de arqueologia documental não deve ser tratada como etapa obrigatória de todo projeto cliente.

Formulação correta

> Codex pode fazer arqueologia/inspeção técnica quando a tarefa exige entender um projeto existente.

Não:

> todo site precisa passar por ARQ formal antes de qualquer mudança.

PARTE XV — PONTOS SENSÍVEIS E CONTROLE HUMANO

54. Credenciais e secrets

O Dev intervém especialmente quando a tarefa envolve:

- credenciais;
- chaves de Supabase;
- secrets;
- acessos.

O fato de a IA conseguir operar uma ferramenta não transfere automaticamente autoridade sobre esses itens.

---

55. Banco e tabelas

Mudanças em:

- banco;
- tabelas;
- migrations;
- dados;

são tratadas como pontos mais sensíveis.

Lovable pode ser preferido por integração, mas o controle humano permanece importante.

A validação de segurança pertence ao Manual 6.

---

56. Mudança ampla

Quando uma mudança tem potencial de afetar grande parte do projeto, o Dev tende a:

- isolar mais;
- investigar antes;
- eventualmente usar branch;
- validar com maior cuidado.

Não existe threshold numérico universal de “mudança ampla”.

---

57. Áreas já aprovadas

A IA deve receber instrução para preservar áreas que já foram validadas.

Essa regra se torna ainda mais importante quando:

- troca de ferramenta;
- entra em nova sessão;
- altera breakpoint;
- muda apenas backend;
- muda apenas Admin.

PARTE XVI — PRÁTICAS DEPRECIADAS RELACIONADAS AO FLUXO TÉCNICO

58. Desenvolver praticamente tudo no Lovable

A abordagem anterior de manter quase todo o desenvolvimento dentro do Lovable deixou de representar o fluxo atual.

O método evoluiu para uso combinado com:

- GitHub;
- ambiente local;
- Codex;
- Claude.

O catálogo completo de práticas depreciadas pertence ao Manual 8.

---

59. Começar fora do Lovable e integrar depois

Também não representa o fluxo atual normal.

A experiência anterior trouxe dificuldades.

O ponto de partida corrente é:

Lovable/Base primeiro.

PARTE XVII — NORMA, AUSÊNCIAS E OPORTUNIDADES

60. NORMA TÉCNICA VIGENTE

Normas técnicas próprias da WSS permanecem válidas independentemente da ferramenta utilizada.

Para SEO/AEO/GEO, a autoridade canônica atual é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Guardrail entre ferramentas

Lovable, Codex, Claude Code ou outro agente não podem, por conveniência de implementação:

- converter conteúdo público indexável para uma solução que deixe de cumprir o resultado server-first/SSR exigido;
- mover conteúdo principal público para dependência exclusiva de JavaScript cliente;
- transformar preview/staging em origem canônica de produção;
- remover requisitos normativos de metadata, status, sitemap, robots, structured data ou áreas privadas.

> **Regra**

> ferramenta não muda norma.

Mas a norma também não deve ser usada para reescrever retroativamente o AS-IS do fluxo Git/IA.

O detalhamento de implementação e QA do padrão pertence ao documento normativo e seus anexos; este Manual apenas preserva o guardrail no fluxo de desenvolvimento.

61. AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

Este Manual registra como ausências reais:

- não existe política formal universal de branching;
- não existe Pull Request obrigatório no fluxo atual;
- não existe tag obrigatória por entrega;
- não existe convenção universal consolidada de commit;
- não existe lock Lovable/local;
- não existe regra abstrata única de source of truth para todo conflito;
- não existe ritual universal de rollback;
- não existe matriz obrigatória Codex × Claude × Lovable;
- não existe threshold formal de mudança ampla;
- não existe automação que impeça alteração fora de escopo.

62. OPORTUNIDADE FUTURA — FORA DO ESCOPO

Possíveis evoluções futuras podem incluir:

- política formal de branching;
- estratégia de lock/coordenação;
- convenções de commit;
- rollback formal;
- maior automação de gates.

Nenhuma dessas soluções é desenhada neste Manual.

PARTE XVIII — RELAÇÃO COM OS OUTROS MANUAIS

63. Relação com o Manual 1

O Manual 1 explica **para que serve cada ferramenta**.

O Manual 5 explica **como elas são combinadas operacionalmente**.

---

64. Relação com o Manual 2

O Manual 2 explica:

- criação do contexto;
- prompt master;
- `contexto.md`;
- `design.md`;
- validação de entendimento.

O Manual 5 usa esse contexto durante a execução, mas não o recria.

---

65. Relação com o Manual 3

O Manual 3 explica:

- referência;
- Home;
- páginas;
- efeitos;
- responsividade;
- performance;
- consistência visual.

O Manual 5 explica como Codex/Claude/Lovable entram tecnicamente nesse trabalho.

---

66. Relação com o Manual 4

O Manual 4 é proprietário das capacidades da Base/Admin.

O Manual 5 é proprietário da mecânica usada para alterar essas capacidades.

---

67. Relação com o Manual 6

O Manual 5 registra que uma alteração deve ser validada antes de subir.

O Manual 6 define:

- como testar;
- como executar QA;
- como investigar bug;
- como revalidar regressões;
- como tratar segurança e SEO.

---

68. Relação com o Manual 7

O Manual 5 trata rollback e sincronização durante desenvolvimento.

O Manual 7 trata:

- publicação;
- produção;
- contingência de produção;
- rollback diante de falha publicada.

---

69. Relação com o Manual 8

O Manual 8 concentra:

- anti-patterns;
- práticas depreciadas;
- troubleshooting;
- casos reais ligados a conflitos, ambiente e recuperação.

PARTE XIX — SOP DERIVADO

70. SOP-002 — Iniciar Tecnicamente um Projeto WSS: Base → Lovable → GitHub → Ambiente Local

Este Manual é a principal base para o:

SOP-002 — Iniciar Tecnicamente um Projeto WSS: Base → Lovable → GitHub → Ambiente Local

O SOP formaliza em sequência executável:

```
contexto pronto
→ remix Base
→ prompt master no Lovable
→ primeiros protótipos
→ conectar GitHub
→ clonar
→ abrir ambiente local
→ ler projeto/contexto
→ validar entendimento
→ preparar primeira rodada local
```

A arquitetura editorial aprovada optou por **não criar um SOP separado apenas para conectar Lovable ↔ GitHub ↔ local**, porque isso faz parte da mesma sequência de inicialização técnica.

---

71. Procedimentos que permanecem incorporados ao Manual

Não viram SOP independente nesta primeira versão:

- conflito Lovable/local;
- recuperação de contexto;
- rollback durante desenvolvimento;
- decisão Codex × Claude × Lovable;
- uso de branch para mudança radical.

São caminhos condicionais do fluxo principal.

PARTE XX — CASO REAL

72. Dr. Guilherme Mendonça

O projeto do Dr. Guilherme Mendonça é o principal exemplo do fluxo corrente:

```
Base Admin
→ Lovable
→ primeiros protótipos
→ GitHub
→ ambiente local
→ Codex/Claude
→ sincronização
→ validação
```

O caso demonstra o uso combinado das ferramentas.

Exclusão

O esboço antecipado feito antes do encerramento de todas as definições do cliente permanece explicitamente fora do método.

O estudo completo pertence ao Manual 8.

PARTE XXI — CRITÉRIO DE CONCLUSÃO

73. Critério de conclusão deste Manual

Este Manual cumpre seu objetivo quando o leitor consegue responder:

1. onde o projeto técnico começa;
2. quando GitHub entra;
3. quando o clone ocorre;
4. por que `main` é usada hoje;
5. quando branch separada pode aparecer;
6. como pull, commit e push entram na rotina;
7. como a alteração local retorna ao Lovable;
8. quando Codex atua direto no código;
9. quando Codex pode conduzir o Lovable;
10. onde Claude entra;
11. o que significa modelo híbrido;
12. como o contexto é retomado;
13. como escopo é limitado;
14. como o Dev lida com edição simultânea;
15. como decide diante de conflito;
16. quais formas de rollback existem;
17. quais pontos ainda não possuem padronização formal;
18. quais assuntos devem ser buscados nos Manuais 3, 4, 6 e 7.

---

74. Fontes principais desta versão

Baseline canônico

1. `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
2. `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
3. `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
4. `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
5. `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

Manuais já estabilizados usados para preservar fronteiras

- `WSS_Manual_01_Visao_Geral_Toolchain_v1.1_FINAL.md`;
- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`;
- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`.

Norma técnica vigente relacionada

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Versões anteriores/superadas de AS-IS, Matriz, Handoff e Plano não foram usadas como baseline ativo.

Status: FINAL — APROVADO

WEB STAR STUDIO

MANUAL

MÉTODO DE DESENVOLVIMENTO DE SITES

MANUAL 6
Testes, QA, Debugging, Segurança e SEO

VERSÃO
v1.1 FINAL

DATA
11/09/2026

STATUS
FINAL — APROVADO

MANUAL

MANUAL 6

Testes, QA, Debugging, Segurança e SEO

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
ESCOPO DOCUMENTAL	primeira documentação ampliada — P0 + P1 + P2 relevante
PROJETO	WSS — Método de Desenvolvimento de Sites
FORMALIZAÇÃO	Pacote Corretivo Pós-Entrega v1.1 aprovado pela Coordenação em 11/09/2026; versão canônica vigente, com v1.0 FINAL preservado como baseline histórico.

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Objetivo 8

2. Escopo e fronteira 9

3. Posição deste Manual no fluxo 10

PARTE I — ESTRATÉGIA DE TESTES AS-IS **11**

4. Modelo atual 12

5. QA não significa apenas teste automatizado 12

6. Codex em testes 12

7. Build e typecheck 13

8. Testes automatizados 13

PARTE II — MOMENTO PRINCIPAL DE QA **14**

9. Quando ocorre 15

10. Relação com revisão do cliente 15

PARTE III — QA VISUAL E RESPONSIVO **16**

11. Três cenários principais 17

12. Aprovação visual prática 17

13. O que o Dev observa 17

14. Responsividade 18

15. Viewports intermediários 18

PARTE IV — QA FUNCIONAL DO SITE PÚBLICO **19**

16. Percurso página por página 20

17. Itens principais 20

18. Links e rotas 20

19. Formulários e Leads 20

20. Chat 21

21. Chatbot 21

PARTE V — QA DO ADMIN **22**

22. Módulo por módulo 23
23. Itens típicos 23
24. Perfis 23

PARTE VI — ESTADOS DE ERRO E CASOS NÃO SISTEMÁTICOS 24
25. Estados vazios 25
26. Mensagens de erro 25
27. 404 25

PARTE VII — DADOS DE TESTE 26
28. Uso de dados falsos 27
29. Limpeza 27

PARTE VIII — DEBUGGING 28
30. Sequência prática 29
31. Começar pelo sintoma 29
32. Console e DevTools 29
33. Frontend/visual 29
34. Backend, banco e autenticação 30
35. Problemas difíceis 30
36. Causa confirmada no AS-IS 31

PARTE IX — CASO REAL DE DEBUGGING: MIRELA 32
37. Sintoma 33
38. Evidência 33
39. Diagnóstico 33
40. Correção 33
41. Lição 33

PARTE X — REGRESSÃO 35
42. Não existe checklist universal 36
43. Alteração visual 36
44. Alteração funcional 36
45. Admin 36
46. Autenticação 36
47. Base/portabilidade 37
48. Regressão não óbvia 37

PARTE XI — TROUBLESHOOTING COMO MÉTODO 38
49. Padrão transversal observado 39
50. Troubleshooting ≠ catálogo neste Manual 39
51. Casos que alimentam o método 39

PARTE XII — SMOKE BUILD DE PRODUÇÃO 41
52. Caso Guilherme Mendonça 42
53. Resposta técnica observada 42
54. Rotas observadas no caso 42
55. O que o caso prova 42
56. Limite documental 43

PARTE XIII — SEO TÉCNICO AS-IS 44
57. Como o Dev trabalha SEO hoje 45
58. Auditoria não é ordem automática 45
59. Classificação de achados 45
60. Itens SEO observados 45
61. Ajustar somente o necessário 46
62. Validação pós-ajuste SEO 46

PARTE XIV — SEOMATOR, AEO E GEO 47
63. SEOmator 48
64. Regra de uso 48
65. Falso positivo ou limitação de coleta 48
66. Dimensões AEO/GEO observadas externamente 48
67. Checklist AEO/GEO real em projeto 49
68. Limite do checklist Daniel 50

PARTE XV — NORMA TÉCNICA VIGENTE DE SEO/AEO/GEO 51
69. Camada normativa separada 52
70. Regra obrigatória de separação 53

PARTE XVI — SEGURANÇA AS-IS 54
71. Como segurança aparece hoje 55
72. Hardening real da Base Admin 55
73. Papéis 55
74. UI não basta 56

75. RLS 56
76. PII 56
77. Superfície pública legítima 56
78. Secrets 57
79. Exemplo real de hardening 57
80. Padrão de segurança observado 58

PARTE XVII — VALIDAÇÃO DE SEGURANÇA PROPORCIONAL 59
81. Procedimento reproduzível sustentado 60
82. Por que não vira SOP próprio agora 60

PARTE XVIII — FEEDBACK E RODADAS DE CORREÇÃO 61
83. Canal predominante 62
84. Formatos 62
85. Consolidação 62
86. Quando voltar para gestão 62
87. ajustes.md 63
88. Antes/depois 63
89. Relação com regressão 63

PARTE XIX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 64
90. Lista consolidada deste Manual 65

PARTE XX — OPORTUNIDADES FUTURAS — FORA DO ESCOPO 66
91. Oportunidades identificáveis 67

PARTE XXI — RELAÇÃO COM OS OUTROS MANUAIS 68
92. Relação com o Manual 3 69
93. Relação com o Manual 4 69
94. Relação com o Manual 5 69
95. Relação com o Manual 7 69
96. Relação com o Manual 8 70

PARTE XXII — SOPS DERIVADOS 71
97. SOP-008 — Executar QA Visual, Funcional e Responsivo 72
98. SOP-009 — Diagnosticar e Corrigir Bug por Camada 72
99. SOP-010 — Tratar Auditoria SEO, AEO e GEO 72
100. SOP-011 — Processar Rodada de Feedback via ajustes.md 73

101. Procedimentos incorporados, sem SOP próprio 73

PARTE XXIII — CASOS REAIS UTILIZADOS **74**

102. Guilherme Mendonça 75

103. Daniel P 75

104. Mirela Albuquerque 75

105. Guilherme Maia 75

PARTE XXIV — CRITÉRIO DE CONCLUSÃO **76**

106. Critério de conclusão deste Manual 77

107. Fontes principais desta versão 77

1. Objetivo

Este Manual documenta como a WSS verifica, investiga e corrige a qualidade técnica e funcional dos sites durante o desenvolvimento.

Seu foco é a passagem:

```
implementação
→ QA
→ teste funcional
→ detecção de problema
→ debugging
→ correção
→ regressão
→ auditorias técnicas
→ ajustes
→ nova validação
```

Ao final deste Manual, o leitor deve compreender:

- quando ocorre o principal QA;
- como o Dev combina validação manual e Codex;
- quais dispositivos/tamanhos são priorizados;
- como o site público é percorrido;
- como o Admin é testado;
- como Leads, formulários, Chat, Chatbot e perfis entram no QA;
- o que não é testado sistematicamente hoje;
- como o Dev inicia um debugging;
- quando usa console/DevTools, Codex, Claude ou Lovable;
- como considera uma hipótese suficientemente confirmada;
- como revalida regressões;
- por que “dev funciona” não prova saúde do build de produção;
- como auditorias SEO/AEO/GEO são interpretadas;
- como SEOmator é usado como input e não como autoridade automática;
- como segurança aparece nas auditorias reais;
- como RLS, papéis, secrets, PII e autorização server-side aparecem no AS-IS;
- como feedback é convertido em `ajustes.md`;
- quais gates formais não existem hoje e não devem ser inventados.

Este Manual registra o **AS-IS** e mantém normas técnicas vigentes em camada própria.

2. Escopo e fronteira

2.1 O que entra neste Manual

Este Manual cobre:

1. momento principal de QA;
2. teste manual;
3. Codex em testes amplos;
4. QA visual;
5. QA responsivo;
6. QA funcional;
7. site público;
8. Admin;
9. Leads/formulários;
10. Chat;
11. Chatbot;
12. autenticação/perfis;
13. dados de teste;
14. debugging;
15. console/DevTools;
16. decisão de ferramenta por camada;
17. regressão;
18. troubleshooting como método;
19. smoke build como procedimento condicional;
20. SEO técnico AS-IS;
21. AEO/GEO AS-IS;
22. SEOmator;
23. segurança AS-IS;
24. RLS;
25. papéis;
26. secrets;
27. PII;
28. autorização server-side;
29. feedback;
30. `ajustes.md`;
31. revalidação das correções.

2.2 O que não entra neste Manual

Ficam fora:

- construção visual detalhada;
- operação Git;

- configuração funcional da Base Admin;
- domínio/DNS;
- publicação em produção;
- contingência de produção;
- catálogo completo de troubleshooting;
- estudos de caso completos.

O catálogo canônico de problemas conhecidos pertence ao Manual 8.

A publicação e o QA em ambiente final pertencem ao Manual 7.

---

3. Posição deste Manual no fluxo

O principal recorte é:

```
páginas suficientemente desenvolvidas
→ QA/revisão do Dev + revisão interna adicional podem se sobrepor
→ correções
→ revisão do cliente quando aplicável
→ novas correções
→ auditorias técnicas aplicáveis
→ validação final
```

No AS-IS, a validação principal começa depois de as páginas estarem suficientemente desenvolvidas. Quando o Dev sinaliza uma versão ou área revisável, outras pessoas podem iniciar a revisão interna em paralelo à própria revisão do Dev.

Isso não significa que testes só aconteçam nesse momento.

Durante a construção, o Dev já valida:

- visual;
- mobile;
- ações principais;
- módulos específicos.

O M6 documenta a camada **sistemática/principal de validação e investigação**.

PARTE I — ESTRATÉGIA DE TESTES AS-IS

4. Modelo atual

A estratégia atual combina:

- validação manual do Dev;
- pedidos amplos de teste ao Codex;
- testes técnicos específicos quando necessários;
- build/typecheck em algumas tarefas/projetos;
- inspeção de navegador/console;
- testes funcionais de fluxos completos.

Não existe uma suíte formal obrigatória e idêntica em todos os projetos.

---

5. QA não significa apenas teste automatizado

No método atual, QA inclui:

- olhar;
- navegar;
- clicar;
- preencher;
- comparar;
- observar;
- reproduzir;
- testar fluxos reais.

A validação humana é central.

---

6. Codex em testes

O Dev pode pedir ao Codex uma rodada ampla de testes.

O Codex pode contribuir com:

- navegação;
- inspeção;
- testes técnicos;
- build;
- typecheck;
- leitura de erros;
- comparação de comportamento.

Limite

A existência do Codex não elimina a validação manual do Dev.

---

7. Build e typecheck

Build e typecheck aparecem em trabalhos reais.

Mas:

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não formam uma suíte formal obrigatória universal que seja executada de forma idêntica em todo projeto e toda rodada.

Este Manual não inventa uma pipeline obrigatória.

---

8. Testes automatizados

Projetos podem conter testes ou scripts específicos.

Exemplos reais incluem:

- benchmark do Chatbot;
- smoke build de produção;
- scripts/verificações dirigidas.

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe suíte automatizada universal obrigatória para todos os sites WSS.

PARTE II — MOMENTO PRINCIPAL DE QA

9. Quando ocorre

O principal QA ocorre depois de as páginas estarem suficientemente desenvolvidas. A revisão interna adicional não precisa esperar o Dev concluir toda a sua própria revisão; ela pode começar em paralelo quando houver versão revisável.

O Dev combina:

1. pedido de testes ao Codex;
2. validação manual;
3. correções;
4. revalidação.

---

10. Relação com revisão do cliente

O institucional pode ser publicado temporariamente para revisão do cliente enquanto o Admin continua sendo finalizado.

Portanto:

- QA do institucional;
- configuração do Admin;
- revisão do cliente;

podem ter sobreposição temporal.

O fluxo não é completamente linear.

PARTE III — QA VISUAL E RESPONSIVO

11. Três cenários principais

O Dev prioriza três cenários:

Desktop

MacBook conectado a monitor.

Notebook

O próprio MacBook.

Mobile

Aparelho físico acessando o projeto local pela rede/Wi-Fi.

---

12. Aprovação visual prática

O visual é considerado adequado quando:

- desktop;
- notebook;
- mobile;

estão corretos no julgamento do Dev.

---

13. O que o Dev observa

O QA visual verifica, entre outros pontos:

- fidelidade;
- tipografia;
- espaçamento;
- alinhamento;
- caixas/cards;
- imagens;
- crop;
- overflow;
- quebras;
- animações;
- scroll;
- navegação;
- consistência;

- CTAs.

---

14. Responsividade

A validação responsiva observa:

- layout;
- textos;
- imagens;
- botões;
- navegação;
- efeitos;
- scroll horizontal;
- comportamento em telas menores.

O Dev pode aceitar solução diferente entre desktop e mobile.

---

15. Viewports intermediários

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe teste sistemático de uma grande matriz de viewports intermediários.

A limitação de tempo faz o Dev priorizar:

- desktop;
- notebook;
- mobile físico.

Isso não significa que tamanhos intermediários nunca sejam testados.

Significa que não há matriz formal universal.

PARTE IV — QA FUNCIONAL DO SITE PÚBLICO

16. Percurso página por página

O site público é percorrido:

- página por página;
- seção por seção;
- do Hero ao Footer.

---

17. Itens principais

O Dev verifica:

- navegação;
- links;
- CTAs;
- formulários;
- interações;
- botões;
- fluxos de contato;
- ações principais.

---

18. Links e rotas

Links precisam levar ao destino esperado.

Caso real Daniel P:

- CTA apontava para `/sobre-mim`;
- a rota existente era `/sobre`;
- resultado era 404.

A correção foi localizada no path.

Lição

Sofisticação visual não substitui QA básico de navegação.

---

19. Formulários e Leads

O teste ponta a ponta inclui:

1. acionar formulário/CTA/agendamento;

2. preencher dados;
3. conferir sucesso/erro;
4. abrir Admin;
5. localizar Lead;
6. conferir nome;
7. conferir origem/trajeto;
8. conferir especialidade/opção.

Isso valida frontend e chegada do dado ao Admin.

---

20. Chat

O Dev:

1. inicia conversa;
2. verifica comportamento;
3. abre Admin;
4. confirma que a conversa aparece.

---

21. Chatbot

Quando aplicável, o Dev testa como:

- usuário;
- administrador.

Pode verificar:

- captura de nome/telefone;
- criação do Lead;
- pergunta coberta;
- pergunta sem resposta confirmada;
- handoff humano;
- WhatsApp quando configurado;
- aparição da conversa no Admin.

A Base possui benchmark técnico, mas ele não substitui o teste manual.

PARTE V — QA DO ADMIN

22. Módulo por módulo

O Dev testa o Admin:

1. abre o módulo;
2. clica;
3. executa os fluxos relevantes;
4. observa o resultado.

---

23. Itens típicos

Conforme o projeto:

- Blog;
- Leads;
- Chat;
- Chatbot;
- FAQ;
- depoimentos;
- perfis;
- configurações;
- outras capacidades contratadas.

---

24. Perfis

O Dev cria duas contas internas:

- Equipe/cliente;
- Webstar.

Ele verifica se cada perfil acessa somente o que deveria.

A validação técnica de autorização server-side aparece mais adiante neste Manual.

PARTE VI — ESTADOS DE ERRO E CASOS NÃO SISTEMÁTICOS

25. Estados vazios

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Estados vazios não são testados sistematicamente em todos os projetos.

Entram quando:

- o fluxo exige;
- há risco percebido;
- surge problema;
- a auditoria aponta.

---

26. Mensagens de erro

Também não existe uma matriz universal de mensagens de erro testadas em toda entrega.

O Dev testa principalmente os fluxos essenciais.

---

27. 404

URLs inexistentes/404 não são testadas sistematicamente no AS-IS de todos os projetos.

Quando surgem em QA/auditoria, são tratadas.

Importante

Se existir **NORMA TÉCNICA VIGENTE** exigindo 404 real, isso pertence à camada normativa e deve ser respeitado quando aplicável.

PARTE VII — DADOS DE TESTE

28. Uso de dados falsos

O Dev cria dados de teste:

- Leads;
- conversas;
- registros necessários.

Isso permite validar fluxos ponta a ponta.

---

29. Limpeza

Antes da entrega do Admin, os dados de teste devem ser limpos/resetados.

A execução final da limpeza pertence ao Manual 7/SOP-012.

PARTE VIII — DEBUGGING

30. Sequência prática

O debugging atual é pragmático e não rígido.

Sequência típica:

1. reproduzir no navegador;
2. observar o sintoma;
3. identificar a camada provável;
4. usar ferramenta adequada;
5. aplicar correção ligada à hipótese;
6. reproduzir novamente.

---

31. Começar pelo sintoma

O Dev não começa necessariamente lendo código linha por linha.

Primeiro procura:

- reproduzir;
- ver o erro;
- entender onde quebra.

---

32. Console e DevTools

Quando o sintoma sugere:

- configuração;
- key;
- ambiente;
- erro frontend;
- runtime;

o Dev abre console/DevTools.

---

33. Frontend/visual

Problemas de:

- visual;
- layout;

- comportamento;
- efeito;
- responsividade;

tendem a ser tratados com:

- Codex;
- Claude.

---

34. Backend, banco e autenticação

Esta seção trata de **investigação/correção pós-criação**. A criação inicial do projeto e da estrutura herdada de backend/banco/auth continua ocorrendo no Lovable durante o remix da Base Admin.

Em problemas pós-criação ligados a:

- backend;
- banco;
- autenticação;
- configuração integrada;

o Codex pode ser a primeira camada de investigação/condução e, conforme a necessidade, executar diretamente o que couber ou operar/orquestrar:

- Lovable;
- Lovable Cloud.

O uso direto do Lovable pelo Dev continua válido quando a integração da plataforma torna esse caminho mais adequado.

---

35. Problemas difíceis

Quando um problema é mais complexo, o Dev pode usar:

- Lovable;
- Codex;

em paralelo para comparar investigações.

---

36. Causa confirmada no AS-IS

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe metodologia formal universal de root cause analysis.

O Dev considera a causa suficientemente confirmada quando:

- a hipótese levou a uma correção;
- o comportamento foi restaurado;
- o erro deixou de reproduzir.

Essa é uma confirmação pragmática.

PARTE IX — CASO REAL DE DEBUGGING: MIRELA

37. Sintoma

No projeto Mirela Albuquerque:

- ao retornar ao Lovable, a configuração/key do Supabase estava ausente/incorreta;
- o Blog deixou de funcionar corretamente.

---

38. Evidência

O erro apareceu no console.

A implementação atual corrobora que ausência de:

- URL;
- publishable key;

pode impedir fluxos dependentes do Supabase.

---

39. Diagnóstico

A causa estava na camada de:

- configuração;
- ambiente;
- conexão Supabase/Lovable.

Não no componente visual do Blog.

---

40. Correção

A configuração foi corrigida via Lovable/ambiente.

---

41. Lição

> Quando um módulo inteiro quebra após mudança de ambiente/conexão, verificar env/secrets/conexões antes de reescrever código funcional.

O caso completo pertence ao Manual 8.

PARTE X — REGRESSÃO

42. Não existe checklist universal

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não há padrão único e completo de regressão para todas as mudanças.

A regressão é contextual.

---

43. Alteração visual

Depois de mudança visual:

- revalidar desktop;
- revalidar mobile;
- verificar a área relacionada.

---

44. Alteração funcional

Depois de mudança funcional:

- refazer o fluxo afetado;
- testar entradas/saídas;
- verificar área relacionada.

---

45. Admin

Depois de mudança no Admin:

- testar o módulo alterado;
- verificar fluxos dependentes quando óbvios.

---

46. Autenticação

Depois de alteração de acesso/perfil:

- testar perfis;
- confirmar limites de acesso.

---

47. Base/portabilidade

Depois de adaptar capacidade da Base:

- testar capacidade portada;
- testar dependências relevantes;
- preservar particularidades locais.

---

48. Regressão não óbvia

Regressões não óbvias podem aparecer somente em:

- QA posterior;
- revisão interna;
- revisão do cliente;
- produção.

Esse risco faz parte do AS-IS atual.

PARTE XI — TROUBLESHOOTING COMO MÉTODO

49. Padrão transversal observado

A VAL-004 permitiu sintetizar:

```
sintoma
→ reproduzir
→ localizar camada
→ formular hipótese
→ confirmar proporcionalmente
→ corrigir de forma localizada
→ revalidar
```

---

50. Troubleshooting ≠ catálogo neste Manual

Este Manual ensina **como investigar**.

O Manual 8 mantém o catálogo canônico de casos:

- TS-001;
- TS-002;
- etc.

Isso evita duplicação.

---

51. Casos que alimentam o método

Dev funciona / produção 500

Possível diferença entre dev e artefato real.

Auditoria externa contradiz UI

Investigar representação/crawler.

Blog quebra após ambiente

Investigar env/secrets/conexão.

CTA 404

Comparar destino com rotas.

Base ameaça legado

Comparar equivalentes/deltas/dependências.

Relatório antigo pede correção já resolvida

Confrontar achado com estado atual.

PARTE XII — SMOKE BUILD DE PRODUÇÃO

52. Caso Guilherme Mendonça

O projeto registrou incidente em que:

- modo dev funcionava;
- build de produção apresentava falha;
- rotas respondiam HTTP 500.

A causa estava ligada ao pacote de produção/bundling.

---

53. Resposta técnica observada

Foi criado smoke test que:

1. gera o build real;
2. sobe localmente o artefato/worker;
3. testa rotas públicas;
4. aceita redirect esperado no Admin;
5. falha em status inesperado;
6. orienta não publicar quando falha.

---

54. Rotas observadas no caso

O smoke atual cobre:

- `/`;
- `/blog`;
- `/contato`;
- `/sobre`;
- `/especialidades`;
- `/auth`;
- `/admin`.

---

55. O que o caso prova

> Dev saudável não garante artefato de produção saudável.

Mudanças em:

- Vite;
- Nitro;
- TanStack Start;
- bundler;
- dependências de build;

podem justificar smoke do pacote real.

---

56. Limite documental

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> O smoke build acima não foi comprovado como gate universal histórico de todos os sites WSS.

Portanto:

- é procedimento técnico real e valioso;
- entra neste Manual;
- não vira SOP independente nesta primeira versão;
- não deve ser apresentado como obrigação AS-IS universal.

Se norma futura o tornar obrigatório, isso será outra camada.

PARTE XIII — SEO TÉCNICO AS-IS

57. Como o Dev trabalha SEO hoje

O Dev não executa manualmente uma lista técnica completa item por item em todo projeto.

Ele tende a pedir auditoria ao:

- Lovable;
- Codex;

e trabalhar os achados.

---

58. Auditoria não é ordem automática

Princípio observado:

```
auditoria/relatório
→ confrontar com estado atual
→ classificar achado
→ corrigir somente delta real
→ validar
```

---

59. Classificação de achados

No caso Guilherme, os achados foram tratados como:

- presente;
- parcialmente resolvido;
- já resolvido;
- não reproduzido.

Essa classificação evita refazer algo já corrigido.

---

60. Itens SEO observados

Auditorias reais verificaram:

- title;
- meta description;
- canonical;
- Open Graph;
- Twitter Cards;

- imagem social;
- dados estruturados;
- breadcrumbs;
- sitemap;
- `robots.txt`;
- links internos;
- `noindex/nofollow`;
- domínio oficial versus preview;
- HTML SSR.

---

61. Ajustar somente o necessário

Exemplo real:

- canonical já resolvido → preservado;
- Open Graph parcial → completado;
- sitemap sem problema descrito → preservado;
- `robots.txt` incompleto → corrigido;
- schemas parciais → completados.

---

62. Validação pós-ajuste SEO

Foram observadas validações como:

- lint direcionado;
- diff check;
- build;
- HTML SSR;
- navegação;
- links internos;
- canonical;
- Open Graph;
- breadcrumbs;
- sitemap;
- robots.

Isso demonstra que auditoria não termina na edição.

PARTE XIV — SEOmator, AEO E GEO

63. SEOmator

SEOmator é ferramenta externa confirmada no AS-IS.

No Daniel P há auditoria GEO real.

---

64. Regra de uso

Resultado do SEOmator deve ser tratado como:

1. achado;
2. interpretação;
3. plano sugerido;
4. implementação comprovada ou não.

Não pular direto de “score” para “alteração”.

---

65. Falso positivo ou limitação de coleta

Caso Daniel:

- SEOmator reportou H1/imagens/schema ausentes;
- a UI mostrava os elementos;
- o crawler não via corretamente conteúdo criado client-side.

Lição

Antes de corrigir:

- entender o que o crawler lê;
- inspecionar HTML bruto;
- distinguir ausência real de invisibilidade ao crawler.

---

66. Dimensões AEO/GEO observadas externamente

A auditoria considerou:

- AI Citability;
- Brand Authority;
- Platform Readiness;
- conteúdo visível sem JS;
- H1;

- imagens/alt;
- schemas;
- SSR/pré-renderização;
- robots;
- crawlers;
- `llms.txt`;
- E-E-A-T;
- entidade/`sameAs`;
- fontes reconhecidas;
- citabilidade factual.

Nem todos os itens são regra universal.

---

67. Checklist AEO/GEO real em projeto

Daniel P possui checklist interno agrupado em:

- AEO;
- GEO;
- SEO;
- E-E-A-T.

Exemplos:

AEO

- FAQ;
- bloco “Em resumo”;
- TL;DR;
- headings interrogativos.

GEO

- entidades;
- variações semânticas;
- bloco “Quem deve ler/agir”;
- links internos.

SEO

- tamanho de title;
- meta description;
- keyword;
- contagem de palavras;
- headings.

E-E-A-T

- revisor;
- data de revisão.

---

68. Limite do checklist Daniel

REGRA DE CONTENÇÃO

Esse checklist é uma implementação real de um projeto.

Ele não deve ser transformado automaticamente em checklist universal WSS nesta documentação AS-IS.

PARTE XV — NORMA TÉCNICA VIGENTE DE SEO/AEO/GEO

69. Camada normativa separada

A autoridade canônica vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md` — **padrão canônico vigente**.

O antigo `WSS_Padrao_Tecnico_Arquitetura_SEO_v0.1.md` está superado como autoridade autônoma e permanece somente como baseline histórico/proveniência.

O checklist AEO/GEO do Daniel P continua sendo um **caso real AS-IS**, não o checklist universal WSS. A partir do Gate 7, porém, a WSS possui matriz e checklists normativos próprios no caderno de anexos do v0.2 FINAL.

Referências operacionais principais para este Manual:

- `WSS_SEO_AEO_GEO_Matriz_Requisitos_Tecnicos_Parte2_v0.3`;
- `WSS_SEO_AEO_GEO_Matriz_Schema_por_Tipo_Pagina_v0.3`;
- `WSS_SEO_AEO_GEO_Matriz_Crawlers_Diretivas_v0.2`;
- `WSS_SEO_AEO_GEO_Checklist_PrePublicacao_v0.3`.

Esses anexos têm propriedade sobre o detalhe técnico; este Manual não os duplica.

Requisitos normativos que afetam QA

Entre os requisitos/guardrails vigentes estão, conforme aplicabilidade:

- arquitetura SSR-first/híbrida e conteúdo principal no HTML inicial para páginas públicas indexáveis;
- metadata/head específico por rota;
- canonical/origem oficial, redirects/status reais e 404 real;
- sitemap XML e `robots.txt`;
- staging/preview não indexável e fora do sitemap oficial;
- structured data coerente e validado;
- áreas privadas fora do sitemap e com `noindex` quando aplicável;
- Google Search Console configurado na publicação.

Performance normativa

Metas de Core Web Vitals são **recomendadas**, no p75 quando houver dados adequados:

- LCP ≤ 2,5 s;
- INP ≤ 200 ms;
- CLS ≤ 0,1.

Priorizar dados de campo/CrUX quando disponíveis; Lighthouse/PSI são diagnósticos e não garantia de ranking.

Decisões preservadas

- GA4: **recomendado, não obrigatório universal**;
- Bing Webmaster Tools: recomendado;

- ferramentas pagas: sem seleção definitiva;
- `llms.txt`: não é requisito universal;
- Search crawling e treinamento devem ser tratados separadamente, mas o default WSS para crawlers de treinamento permanece pendente;
- entrega padrão × recorrência permanece decisão estratégica/comercial pendente.

---

70. Regra obrigatória de separação

Este Manual deve mostrar:

AS-IS

O que o Dev e os projetos realmente fazem hoje.

> **NORMA TÉCNICA VIGENTE**
>
> O que deve ser cumprido por exigência aprovada.

A norma pode ser mais exigente que a rotina histórica. Ela não deve ser usada para afirmar que o Dev sempre executou historicamente o mesmo checklist.

PARTE XVI — SEGURANÇA AS-IS

71. Como segurança aparece hoje

O Dev não declarou executar um checklist manual isolado de segurança antes de toda publicação.

Segurança aparece principalmente em:

- auditorias gerais;
- Lovable;
- Codex;
- hardening específico;
- regressões.

---

72. Hardening real da Base Admin

A VAL-003 confirmou verificações reais de:

- credenciais fixas;
- endpoints privilegiados;
- papéis;
- usuário sem papel;
- `anon`;
- escalonamento;
- RLS;
- PII;
- payloads;
- autorização server-side;
- secrets;
- superfície pública;
- resíduos de QA;
- build/typecheck.

---

73. Papéis

Foram explicitamente testados:

- `admin_webstar`;
- `admin_equipe`;
- autenticado sem papel;
- `anon`.

Objetivos:

- Webstar mantém acesso previsto;
- Equipe não ganha acesso Webstar-only;
- sem papel não ganha acesso administrativo;
- `anon` não ganha acesso administrativo;
- escalonamento é bloqueado.

---

74. UI não basta

A auditoria verificou autorização **server-side**, não apenas ocultação visual.

Portanto:

> botão escondido não é prova suficiente de controle de acesso.

---

75. RLS

Houve testes negativos de RLS em tabelas reais.

O objetivo era distinguir:

- leitura pública legítima;
- dados administrativos/PII que não devem ser públicos.

---

76. PII

Exemplos de superfícies que não devem ser expostas a `anon` incluem:

- Leads;
- configurações administrativas;
- auditorias;
- knowledge administrativa;
- dados sensíveis/caches externos conforme o caso.

---

77. Superfície pública legítima

Segurança não significa bloquear tudo.

Foram preservados/validados:

- tracking;
- chat;
- FAQ;
- formulários;
- páginas públicas;
- robots;
- sitemap.

---

78. Secrets

Em projeto real foram registrados como server-only:

- `SUPABASE_URL`;
- `SUPABASE_PUBLISHABLE_KEY`;
- `SUPABASE_SERVICE_ROLE_KEY`;
- `INBOUND_WEBHOOK_SECRET`;
- `LOVABLE_API_KEY`.

> **Limite**
>
> A lista é exemplo real, não inventário universal de todos os projetos.

---

79. Exemplo real de hardening

Foi encontrado endpoint histórico de seed de administradores com:

- credenciais fixas;
- acesso privilegiado.

Antes da remoção:

1. investigou-se uso/dependência;
2. confirmou-se ausência de consumidor produtivo;
3. endpoint foi neutralizado;
4. credenciais/acesso foram removidos;
5. regressões foram testadas.

---

80. Padrão de segurança observado

```
achado potencial
→ provar uso/dependências
→ medir risco
→ corrigir somente se seguro
→ testar regressão
```

PARTE XVII — VALIDAÇÃO DE SEGURANÇA PROPORCIONAL

81. Procedimento reproduzível sustentado

Quando uma validação de segurança é necessária, o AS-IS técnico sustenta:

```
inventariar papéis/rotas/endpoints/server functions/tabelas
→ procurar credenciais e superfícies privilegiadas
→ provar dependências antes de remover
→ testar papéis
→ testar sem papel
→ testar anon
→ testar escalonamento
→ testar RLS negativa
→ confirmar superfície pública legítima
→ verificar PII
→ typecheck/build quando aplicável
→ revalidar regressões
```

---

82. Por que não vira SOP próprio agora

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não há evidência de que essa sequência seja um **gate universal obrigatório antes de toda publicação**.

Ela é uma prática técnica real/proporcional sustentada por auditorias.

Para evitar TO-BE:

- fica incorporada ao M6;
- não vira SOP independente nesta primeira versão.

PARTE XVIII — FEEDBACK E RODADAS DE CORREÇÃO

83. Canal predominante

O feedback chega predominantemente por:

WhatsApp

O Flow existe para tarefas/registros internos, mas o WhatsApp foi identificado como principal canal de feedback no AS-IS.

---

84. Formatos

Feedback pode chegar como:

- texto;
- áudio;
- mensagem;
- documento;
- prints;
- conversa.

Quando vem organizado em documento/prints, o Dev considera o processo mais simples.

---

85. Consolidação

O Dev costuma ler e iniciar correções.

Felipe/gestão podem consolidar quando necessário.

---

86. Quando voltar para gestão

Se a solicitação:

- está pouco clara;
- é contraditória;
- representa possível mudança de escopo;
- sugere nova funcionalidade;

o Dev valida com Felipe/gestão antes de executar.

---

87. `ajustes.md`

O arquivo centraliza a rodada.

Fluxo:

1. registrar pedidos;
2. trabalhar item por item;
3. marcar concluídos;
4. pedir à IA revisão da lista completa;
5. revalidar áreas afetadas;
6. encerrar quando todos os itens foram tratados.

---

88. Antes/depois

Quando necessário, o Dev pode gerar documento com:

- prints anteriores;
- prints posteriores;
- comprovação das alterações.

Não é etapa universal obrigatória.

---

89. Relação com regressão

Cada item resolvido pode afetar outra área.

Por isso:

- não basta marcar como concluído;
- é necessário revalidar área afetada;
- a revisão final da lista ajuda a detectar resíduo.

PARTE XIX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

90. Lista consolidada deste Manual

O AS-IS não possui:

- suíte automatizada universal obrigatória;
- mesma bateria de testes em todos os projetos;
- método formal universal de root cause analysis;
- matriz sistemática de todos os viewports intermediários;
- checklist sistemático de estados vazios em todos os projetos;
- checklist sistemático de 404 em todos os projetos;
- smoke build universal comprovado para todos os sites;
- score SEOmator universal;
- uso obrigatório de SEOmator em toda entrega;
- mesmos thresholds AEO/GEO em todos os sites;
- checklist manual de segurança isolado antes de toda publicação;
- mesma matriz de RLS em todos os projetos;
- SOP universal de segurança antes de publicação;
- processo formal único de regressão;
- canal único obrigatório para feedback.

Essas ausências são achados e devem permanecer visíveis.

PARTE XX — OPORTUNIDADES FUTURAS — FORA DO ESCOPO

91. Oportunidades identificáveis

Sem desenhar solução:

- ampliar automação de testes;
- formalizar gates de build;
- padronizar regressão;
- formalizar segurança por criticidade;
- estruturar feedback prioritariamente em Flow;
- ampliar cobertura de estados de erro;
- formalizar acessibilidade.

Nenhuma dessas oportunidades deve reescrever o AS-IS.

PARTE XXI — RELAÇÃO COM OS OUTROS MANUAIS

92. Relação com o Manual 3

M3 constrói:

- visual;
- responsividade;
- performance percebida.

M6 valida:

- resultado;
- regressão;
- comportamento.

---

93. Relação com o Manual 4

M4 explica capacidades.

M6 testa:

- Leads;
- Chat;
- Chatbot;
- perfis;
- Admin.

---

94. Relação com o Manual 5

M5 explica execução e sincronização.

M6 explica como validar alterações e diagnosticar quando algo falha.

---

95. Relação com o Manual 7

M6 é proprietário do método geral de QA.

M7 é proprietário do QA aplicado em produção/pós-publicação.

---

96. Relação com o Manual 8

M6 ensina:

> como investigar.

M8 registra:

> quais problemas conhecidos já ocorreram.

PARTE XXII — SOPs DERIVADOS

97. SOP-008 — Executar QA Visual, Funcional e Responsivo

Este SOP formaliza:

- momento;
- preparação;
- desktop/notebook/mobile;
- páginas;
- Admin;
- Leads;
- Chat;
- perfis;
- evidências;
- stop condition.

---

98. SOP-009 — Diagnosticar e Corrigir Bug por Camada

Este SOP formaliza:

```
reproduzir
→ localizar camada
→ coletar evidência
→ formular hipótese
→ corrigir
→ revalidar
```

Sem inventar método formal de causa raiz.

---

99. SOP-010 — Tratar Auditoria SEO, AEO e GEO

Este SOP formaliza:

```
receber auditoria
→ comparar com estado atual
→ classificar achado
→ identificar falso positivo/limitação
→ corrigir delta real
→ validar
```

---

100. SOP-011 — Processar Rodada de Feedback via `ajustes.md`

Este SOP formaliza:

- receber;
- consolidar;
- resolver ambiguidade;
- registrar;
- executar;
- marcar;
- revisar lista;
- revalidar.

---

101. Procedimentos incorporados, sem SOP próprio

Nesta primeira versão ficam incorporados ao M6:

- smoke build;
- validação de segurança proporcional.

Motivo:

- são práticas/procedimentos reais;
- mas não foram comprovados como gate universal AS-IS.

PARTE XXIII — CASOS REAIS UTILIZADOS

102. Guilherme Mendonça

Uso no M6:

- QA corrente;
- SEO/AEO/GEO;
- build de produção;
- smoke test;
- validação de metadados.

---

103. Daniel P

Uso:

- CTA 404;
- SEOmator;
- falso positivo/visibilidade CSR;
- acessibilidade pontual.

---

104. Mirela Albuquerque

Uso:

- debugging por ambiente;
- Supabase;
- console;
- não reescrever Blog antes de verificar configuração.

---

105. Guilherme Maia

Uso:

- regressão em portabilidade;
- preservar capacidade local;
- validar delta adaptado.

PARTE XXIV — CRITÉRIO DE CONCLUSÃO

106. Critério de conclusão deste Manual

Este Manual cumpre seu objetivo quando o leitor consegue responder:

1. quando ocorre o principal QA;
2. como Dev e Codex se dividem;
3. quais dispositivos são priorizados;
4. como o site público é percorrido;
5. como o Admin é testado;
6. como Leads/Chat/Chatbot entram no QA;
7. o que não é testado sistematicamente;
8. como o debugging começa;
9. quando usar console;
10. como escolher Codex/Claude/Lovable por camada;
11. como a causa é considerada confirmada no AS-IS;
12. como executar regressão contextual;
13. por que dev não garante build de produção;
14. como smoke build entra sem virar gate universal;
15. como uma auditoria SEO é triada;
16. como usar SEOmator sem obedecer cegamente;
17. quais dimensões AEO/GEO aparecem em casos reais;
18. como segurança é validada em auditorias reais;
19. como papéis/RLS/secrets/PII entram;
20. por que a validação de segurança permanece proporcional;
21. como feedback vira `ajustes.md`;
22. quais ausências atuais não devem ser silenciosamente formalizadas.

---

107. Fontes principais desta versão

Baseline canônico

1. `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
2. `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
3. `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
4. `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
5. `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

Evidência técnica complementar

- `WSS_VAL_003_Auditorias_SEO_AEO_GEO_Seguranca_v0.1.md`;
- `WSS_VAL_004_Troubleshooting_Estudos_de_Caso_v0.1.md`.

Manuais já estabilizados usados para preservar fronteiras

- `WSS_Manual_01_Visao_Geral_Toolchain_v1.1_FINAL.md`;
- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`;
- `WSS_Manual_03_Construcao_Visual_v1.1_FINAL.md`;
- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`;
- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`.

Norma técnica relacionada

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md` — **padrão canônico vigente**;
- `WSS_Padrao_Tecnico_Arquitetura_SEO_v0.1.md` — baseline histórico/proveniência; **não é autoridade autônoma vigente**;
- `WSS_SEO_AEO_GEO_Matriz_Requisitos_Tecnicos_Parte2_v0.3`;
- `WSS_SEO_AEO_GEO_Matriz_Schema_por_Tipo_Pagina_v0.3`;
- `WSS_SEO_AEO_GEO_Matriz_Crawlers_Diretivas_v0.2`;
- `WSS_SEO_AEO_GEO_Checklist_PrePublicacao_v0.3`.

Versões anteriores/superadas de AS-IS, Matriz, Handoff e Plano não foram usadas como baseline ativo.

Status: FINAL — APROVADO

WEB STAR STUDIO

MANUAL

MÉTODO DE DESENVOLVIMENTO DE SITES

MANUAL 7
Publicação, Entrega e Pós-Publicação

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

MANUAL

MANUAL 7

Publicação, Entrega e Pós-Publicação

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
ESCOPO DOCUMENTAL	primeira documentação ampliada — P0 + P1 + P2 relevante
PROJETO	WSS — Método de Desenvolvimento de Sites
FORMALIZAÇÃO	Pacote Corretivo Pós-Entrega v1.1 aprovado pela Coordenação em 11/09/2026; versão canônica vigente, com v1.0 FINAL preservado como baseline histórico.

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Objetivo 7
2. Escopo e fronteira 8
3. Posição no fluxo 9
PARTE I — PREPARAÇÃO PARA PUBLICAÇÃO 10
4. Critério prático do Dev 11
5. O que esse gate pressupõe 11
6. Não existe “reconfiguração final” universal 11
7. Build de produção e smoke 12
PARTE II — PUBLICAÇÃO TEMPORÁRIA PARA VALIDAÇÃO 13
8. Finalidade 14
9. Exemplo real 14
10. Quem pode publicar temporariamente 14
11. Publicação temporária não significa entrega final 14
11.1 NORMA TÉCNICA VIGENTE — staging/preview 15
12. Revisão do cliente 15
PARTE III — DA REVISÃO À VALIDAÇÃO FINAL 16
13. Rodada de ajustes 17
14. Questões de escopo 17
15. Validação final 17
PARTE IV — PROCESSO SITES WSS × MÉTODO DE PUBLICAÇÃO 18
16. Marco 9 como contexto operacional 19
17. Não reescrever o Processo 19
PARTE V — DOMÍNIO DEFINITIVO 20
18. Responsáveis atuais 21
19. Dois cenários recorrentes 21
20. Papel do cliente 21

PARTE VI — DNS 22

21. DNS varia por caso 23

22. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — sequência DNS única 23

23. O que pode ser documentado com segurança 23

PARTE VII — PUBLICAÇÃO EM PRODUÇÃO 24

24. Mudança de ambiente 25

25. O site publicado deve preservar o que já funcionava 25

PARTE VIII — QA PÓS-PUBLICAÇÃO 26

26. O Dev retorna ao projeto 27

27. Páginas e rotas 27

28. Refresh direto 27

29. Formulários e Leads 27

30. Chat/CTA 27

31. Blog 28

32. Login e Admin 28

33. Mobile 28

34. Domínio e HTTPS 28

35. Analytics 28

36. Pontos de SEO 29

PARTE IX — NORMA TÉCNICA VIGENTE NA PUBLICAÇÃO 30

37. Camada normativa 31

38. Separação obrigatória 31

PARTE X — FALHA EM PRODUÇÃO 32

39. Primeira classificação prática 33

40. Falha localizada 33

41. Falha geral 33

42. Prioridade 33

43. Origem da versão de rollback 34

44. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — rollback de produção 34

PARTE XI — ENTREGA DO ADMIN 35

45. Pré-condições 36

46. Limpeza de dados 36

47. O que não deve ser apagado automaticamente 36
48. Perfis internos 36
49. Geração do link de criação de conta 37
50. Quem envia o link 37
51. Credenciais internas 37

PARTE XII — ORIENTAÇÃO DE USO DO ADMIN 38
52. Demonstração 39
53. Manual na geração atual × obrigatoriedade universal 39
54. O que este Manual não deve fazer 39

PARTE XIII — CRITÉRIO DE ENTREGA INICIAL CONCLUÍDA 40
55. Três condições principais 41
56. Publicar site não encerra sozinho o trabalho 41
57. Entrega inicial × operação recorrente 41

PARTE XIV — HANDOFF PARA MANUTENÇÃO/RECORRÊNCIA 42
58. Continuidade 43
59. Serviços recorrentes possíveis 43
60. Relação com o Processo Sites WSS 43

PARTE XV — ANALYTICS, SEARCH CONSOLE E ITENS CONEXOS 45
61. Analytics no Processo 46
62. Analytics no AS-IS do Dev 46
63. Search Console 46

PARTE XVI — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 47
64. Lista consolidada deste Manual 48

PARTE XVII — OPORTUNIDADES FUTURAS — FORA DO ESCOPO 49
65. Oportunidades possíveis 50

PARTE XVIII — BOAS PRÁTICAS RELEVANTES 51
66. Validar produção de verdade 52
67. Não entregar dados de teste 52
68. Não entregar acesso interno como conta do cliente 52
69. Restaurar antes de investigar quando produção está amplamente quebrada 52

PARTE XIX — ANTI-PATTERNS RELACIONADOS 53

70. Considerar publish bem-sucedido como QA suficiente 54

71. Inventar uma receita única de DNS 54

72. Corrigir produção amplamente quebrada sem considerar rollback 54

73. Entregar Admin com dados de teste 54

PARTE XX — CASO REAL **55**

74. Guilherme Mendonça 56

75. O que não generalizar 56

PARTE XXI — RELAÇÃO COM OS OUTROS MANUAIS **57**

76. Relação com o Manual 4 58

77. Relação com o Manual 5 58

78. Relação com o Manual 6 58

79. Relação com o Manual 8 58

PARTE XXII — SOP DERIVADO **59**

80. SOP-012 — Preparar, Publicar, Validar Produção e Entregar Admin 60

81. Por que não separar entrega do Admin em outro SOP 60

PARTE XXIII — CRITÉRIO DE CONCLUSÃO **61**

82. Critério de conclusão deste Manual 62

83. Fontes principais desta versão 62

1. Objetivo

Este Manual documenta como um site WSS passa da validação interna para:

- publicação temporária;
- revisão do cliente;
- domínio definitivo;
- produção;
- validação pós-publicação;
- entrega do Admin;
- continuidade em manutenção ou recorrência.

Seu foco é a transição:

```
site validado
→ publicação temporária
→ revisão do cliente
→ ajustes
→ validação final
→ domínio/produção
→ QA pós-publicação
→ entrega do Admin
→ manutenção/recorrência
```

Ao final, o leitor deve compreender:

- quando o site é considerado pronto para publicação pelo Dev;
- para que serve o domínio temporário `webstarstudio.site`;
- quem pode executar a publicação temporária;
- quem concentra hoje domínio/GoDaddy;
- quais são os dois cenários recorrentes de domínio;
- por que não existe uma única sequência DNS no AS-IS;
- o que é revalidado depois da publicação;
- como agir diante de falha localizada ou falha geral;
- como o Admin é entregue;
- como dados de teste são tratados;
- como o cliente cria seu próprio acesso;
- quando a entrega inicial é considerada concluída;
- como o projeto segue para manutenção/recorrência;
- quais pontos pertencem ao método atual e quais pertencem ao Processo Sites WSS ou a normas técnicas próprias.

Este Manual registra o **AS-IS**. Ele não cria uma nova política de deploy, DNS, rollback, handoff ou operação recorrente.

2. Escopo e fronteira

2.1 O que entra neste Manual

Este Manual cobre:

1. gate prático de pré-publicação;
2. publicação temporária para validação do cliente;
3. domínio `webstarstudio.site`;
4. revisão do cliente;
5. validação final;
6. domínio definitivo;
7. GoDaddy;
8. cenários de domínio existente/novo;
9. DNS no nível AS-IS;
10. papéis de Dev, Analice, Felipe/CEO e cliente;
11. publicação em produção;
12. QA pós-publicação;
13. páginas/rotas;
14. refresh direto;
15. formulários/Leads;
16. Chat/CTA;
17. Blog;
18. login/Admin;
19. mobile;
20. domínio/HTTPS;
21. Analytics quando aplicável;
22. pontos de SEO;
23. falha localizada;
24. rollback por falha geral;
25. limpeza de dados de teste;
26. perfis internos;
27. criação do acesso do cliente;
28. demonstração do Admin;
29. encerramento da entrega inicial;
30. manutenção e recorrência.

2.2 O que não entra neste Manual

Ficam fora:

- desenvolvimento visual;
- configuração funcional detalhada do Admin;
- Git/GitHub;

- debugging geral;
- QA completo pré-publicação;
- auditoria SEO/AEO/GEO em profundidade;
- segurança em profundidade;
- tutorial universal de DNS;
- manual de uso detalhado do Admin;
- processo comercial de recorrência.

O M6 é proprietário do método geral de QA.

O M7 aplica esse método ao ambiente publicado.

---

3. Posição no fluxo

O recorte principal começa quando:

- páginas estão validadas;
- fluxos essenciais já foram testados;
- revisão interna foi concluída;
- ajustes necessários já foram tratados ou estão no estágio final.

O fluxo macro é:

```
QA interno
↓
publicação temporária
↓
revisão do cliente
↓
rodada de ajustes
↓
validação final
↓
domínio definitivo
↓
produção
↓
QA pós-publicação
↓
entrega do Admin
↓
manutenção/recorrência
```

PARTE I — PREPARAÇÃO PARA PUBLICAÇÃO

4. Critério prático do Dev

No AS-IS, o Dev considera o site pronto para publicação quando:

- as páginas estão validadas;
- desktop funciona;
- notebook funciona;
- mobile funciona;
- validação interna foi concluída;
- fluxos técnicos já testados continuam funcionando.

Esse é o gate prático do Dev.

---

5. O que esse gate pressupõe

Antes da publicação, já devem ter sido tratados, conforme aplicável:

- navegação;
- formulários;
- Leads;
- Chat/CTA;
- Blog;
- login/Admin;
- perfis;
- ajustes do cliente;
- auditorias técnicas necessárias;
- correções de regressão.

O detalhamento pertence aos Manuais 4 e 6.

---

6. Não existe “reconfiguração final” universal

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe hoje uma etapa separada e universal imediatamente antes do publish para reconfigurar:

- variáveis;
- callbacks;
- integrações;
- secrets;

- destinos.

A expectativa do Dev é que esses itens **já tenham sido validados antes**.

Isso não significa que nunca seja necessário revisar configuração.

Significa apenas que não existe um ritual adicional único no AS-IS.

---

7. Build de produção e smoke

O caso Guilherme Mendonça comprovou que:

- dev pode estar saudável;
- o artefato de produção pode falhar.

O smoke build é uma prática técnica real naquele projeto e pode ser relevante quando houver risco de build/bundler.

> **Limite**
>
> O M7 não transforma o smoke em gate universal.

O procedimento e sua classificação pertencem ao Manual 6.

PARTE II — PUBLICAÇÃO TEMPORÁRIA PARA VALIDAÇÃO

8. Finalidade

Antes do domínio definitivo, a WSS pode publicar o site em domínio temporário:

`webstarstudio.site`

Objetivo:

- permitir navegação externa;
- permitir revisão do cliente;
- validar a experiência em ambiente publicado;
- coletar ajustes antes do domínio definitivo.

---

9. Exemplo real

Caso Guilherme Mendonça:

`https://drguilhermemendonca.webstarstudio.site`

O endereço foi usado como ambiente de validação antes da produção definitiva.

---

10. Quem pode publicar temporariamente

No fluxo atual:

> qualquer integrante autorizado da equipe pode publicar via Lovable no domínio temporário.

Não há evidência de que essa ação seja exclusiva do Dev.

---

11. Publicação temporária não significa entrega final

A publicação temporária pode ocorrer quando:

- o institucional está suficientemente validado;
- o cliente já pode revisar;
- o Admin ainda está sendo finalizado em paralelo.

Portanto:

```
site público em revisão
+
Admin em configuração
```

podem coexistir.

11.1 NORMA TÉCNICA VIGENTE — staging/preview

A publicação temporária em `webstarstudio.site` é uma prática AS-IS válida para revisão, mas o `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md` estabelece que staging/preview deve:

- permanecer **não indexável**;
- ficar fora do sitemap oficial;
- não assumir a própria origem como canonical de produção por fallback;
- nunca se tornar a autoridade canônica do domínio final.

Essa regra normativa complementa, sem eliminar, o uso operacional do domínio temporário.

---

12. Revisão do cliente

Depois de receber o ambiente temporário, o cliente pode:

- navegar;
- identificar correções;
- enviar feedback;
- solicitar ajustes;
- validar conteúdo e experiência.

O processamento de feedback pertence ao Manual 6/SOP-011.

PARTE III — DA REVISÃO À VALIDAÇÃO FINAL

13. Rodada de ajustes

Os pedidos do cliente são consolidados em:

`ajustes.md`

quando aplicável.

Fluxo já documentado:

1. registrar;
2. executar;
3. marcar concluído;
4. revisar a lista inteira;
5. revalidar áreas afetadas.

---

14. Questões de escopo

Se o pedido do cliente:

- é ambíguo;
- contradiz decisões anteriores;
- representa possível nova funcionalidade;
- altera escopo;

o Dev retorna para Felipe/gestão antes de executar.

---

15. Validação final

Depois dos ajustes:

- o Dev revalida;
- a equipe realiza as validações internas necessárias;
- o projeto avança para domínio definitivo/produção.

> **Importante**
>
> O M7 não cria uma cadeia universal rígida de aprovadores além do que está sustentado.

O Processo Sites WSS pode possuir validações formais próprias em seus marcos.

PARTE IV — PROCESSO SITES WSS × MÉTODO DE PUBLICAÇÃO

16. Marco 9 como contexto operacional

No Roadmap Sites WSS, o Marco 9 — Publicação e Pós-Entrega representa:

- domínio/DNS;
- site publicado;
- Admin acessível;
- convite enviado;
- testes pós-publicação;
- entrega;
- transição para recorrência.

Esse Roadmap organiza **o que precisa existir**.

Este Manual documenta **como a etapa é executada no AS-IS do desenvolvimento**.

---

17. Não reescrever o Processo

Este Manual não substitui:

- critérios formais de marco;
- status do Flow;
- checklists do Marco 9;
- registro formal de encerramento.

Esses elementos pertencem ao Processo Sites WSS.

PARTE V — DOMÍNIO DEFINITIVO

18. Responsáveis atuais

No AS-IS:

- **Analice**;
- **Felipe/CEO**;

concentram o acesso à GoDaddy.

O Dev normalmente não executa a configuração de domínio por não possuir esse acesso.

---

19. Dois cenários recorrentes

Cenário A — cliente já possui domínio

O cliente fornece:

- dados;
- acesso;
- informações necessárias.

A WSS publica/configura o site no domínio existente.

Cenário B — cliente não possui domínio

A WSS:

- cria/registra o domínio;
- utiliza a GoDaddy;
- conecta o site.

---

20. Papel do cliente

Conforme o cenário, o cliente pode precisar:

- fornecer acesso ao provedor;
- autorizar alteração;
- informar dados de domínio;
- colaborar com transferência/configuração.

O método não presume que a WSS sempre controle previamente o domínio.

PARTE VI — DNS

21. DNS varia por caso

O procedimento de DNS depende de:

- provedor;
- domínio existente ou novo;
- configuração atual;
- acesso disponível;
- tipo de apontamento;
- estado anterior do domínio.

---

22. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — sequência DNS única

Não existe, no AS-IS consolidado, uma sequência técnica universal registrada como:

```
sempre alterar registro X
→ sempre remover Y
→ sempre aguardar Z
```

Este Manual **não inventa** esse tutorial.

---

23. O que pode ser documentado com segurança

No nível atual:

1. identificar cenário;
2. garantir acesso/responsável;
3. conectar domínio ao site;
4. aguardar/confirmar propagação quando aplicável;
5. verificar HTTPS;
6. testar produção.

A implementação específica depende do provedor/caso.

PARTE VII — PUBLICAÇÃO EM PRODUÇÃO

24. Mudança de ambiente

A publicação definitiva leva o projeto para o domínio oficial.

Essa mudança exige revalidação porque:

- origem muda;
- domínio muda;
- HTTPS entra na verificação;
- rotas diretas precisam funcionar;
- integrações podem depender de origem;
- metadados/SEO podem depender do domínio oficial.

---

25. O site publicado deve preservar o que já funcionava

O princípio do Dev é prático:

> navegar e comprovar que produção preserva os fluxos previamente validados.

Não basta o publish terminar sem erro aparente.

PARTE VIII — QA PÓS-PUBLICAÇÃO

26. O Dev retorna ao projeto

Depois da publicação, o Dev volta ao site e realiza validação prática.

---

27. Páginas e rotas

Verifica:

- Home;
- páginas internas;
- rotas relevantes;
- navegação;
- URLs.

---

28. Refresh direto

O Dev testa refresh direto nas rotas.

Isso ajuda a identificar:

- problema de roteamento;
- comportamento diferente entre navegação SPA e acesso direto;
- falha de servidor/configuração.

---

29. Formulários e Leads

Reexecuta, quando aplicável:

- envio de formulário;
- CTA;
- captação;
- confirmação no Admin.

---

30. Chat/CTA

Verifica:

- presença;
- abertura;

- destino;
- conversa;
- captação, quando aplicável.

---

31. Blog

Verifica:

- listagem;
- artigos;
- navegação;
- funcionamento em produção.

---

32. Login e Admin

Quando aplicável:

- login;
- acesso;
- Admin;
- módulos essenciais.

---

33. Mobile

O Dev também volta a verificar a experiência mobile em produção.

---

34. Domínio e HTTPS

Confere:

- domínio correto;
- site acessível;
- HTTPS funcionando.

---

35. Analytics

O Dev verifica Analytics **quando aplicável**.

Limite AS-IS

O próprio Dev declarou que Analytics/GTM não são atribuição normal dele no fluxo descrito.

Portanto, este item não deve ser transformado em obrigação pessoal universal do Dev.

---

36. Pontos de SEO

Depois da publicação, o Dev verifica pontos de SEO relevantes.

Isso pode envolver:

- domínio oficial;
- canonical;
- metadados;
- sitemap;
- robots;
- indexabilidade;
- outros itens conforme auditoria/norma.

O procedimento detalhado pertence ao Manual 6.

PARTE IX — NORMA TÉCNICA VIGENTE NA PUBLICAÇÃO

37. Camada normativa

A autoridade canônica vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Na publicação, o padrão exige/estabelece, conforme aplicabilidade:

- uma única origem oficial de produção para canonical, URLs absolutas de Open Graph, sitemap, `robots.txt` e structured data;
- metadata/head específico por rota pública indexável;
- sitemap/robots/status/redirects/404 coerentes;
- staging/preview não indexável e fora do sitemap oficial;
- áreas privadas fora do sitemap e com `noindex` quando aplicável;
- Google Search Console configurado na publicação;
- preservação do conteúdo principal público em arquitetura server-first/SSR conforme o padrão.

A ausência de configuração crítica da origem oficial em produção não deve ser mascarada por fallback silencioso para host de preview/staging.

Para o detalhamento técnico, consultar o padrão e, especialmente:

- `WSS_SEO_AEO_GEO_Checklist_PrePublicacao_v0.3`;
- `WSS_SEO_AEO_GEO_Checklist_PosPublicacao_v0.2`.

38. Separação obrigatória

Este Manual preserva:

AS-IS

O que o Dev realmente verifica e executa hoje.

> **NORMA TÉCNICA VIGENTE**
>
> O que uma norma própria exige.

Uma exigência normativa pode ser mais ampla que a rotina histórica do Dev.

PARTE X — FALHA EM PRODUÇÃO

39. Primeira classificação prática

Quando surge falha publicada, o Dev diferencia:

Falha localizada

- uma área específica quebra;
- o restante do site permanece utilizável.

Falha geral

- a publicação compromete o funcionamento global;
- site deixa de estar adequadamente utilizável.

---

40. Falha localizada

No AS-IS:

> correção emergencial localizada.

O objetivo é restaurar a área afetada sem derrubar o restante.

---

41. Falha geral

Quando o funcionamento geral quebra:

1. retornar para build/versão anterior funcional;
2. restaurar o site;
3. investigar a causa depois.

---

42. Prioridade

O princípio é:

> restabelecer funcionamento antes de aprofundar a investigação quando a produção está amplamente quebrada.

O debugging posterior pertence ao Manual 6.

---

43. Origem da versão de rollback

A versão anterior funcional pode ser recuperada por mecanismos já documentados no M5, como:

- histórico do Lovable;
- commit anterior;
- build/versão anterior.

---

44. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — rollback de produção

Não existe ritual universal formalizado por categoria de falha.

Este Manual não inventa:

- tabela de rollback;
- SLA;
- branch obrigatória;
- tag obrigatória;
- playbook de incidente.

PARTE XI — ENTREGA DO ADMIN

45. Pré-condições

Antes da entrega do Admin:

- perfis internos já foram testados;
- capacidade relevante já foi validada;
- dados de teste precisam ser limpos.

---

46. Limpeza de dados

O Dev pede ao:

- Lovable;
- Codex;

para limpar/resetar dados de teste no banco antes da entrega.

Podem incluir:

- Leads falsos;
- conversas;
- outros registros de QA.

---

47. O que não deve ser apagado automaticamente

A limpeza deve atingir **dados de teste**, não dados reais.

Controle humano

A identificação do que é teste versus produção permanece uma responsabilidade sensível.

Este Manual não cria rotina automática de purge universal.

---

48. Perfis internos

Antes de entregar:

- perfil Equipe/cliente foi testado;
- perfil Webstar foi testado;
- restrições funcionais foram verificadas.

Os detalhes de segurança pertencem ao Manual 6.

---

49. Geração do link de criação de conta

O Dev gera no Admin um link para o cliente criar sua própria conta.

Isso evita entregar ao cliente uma credencial interna compartilhada.

---

50. Quem envia o link

O link pode ser enviado por:

- Dev;
- Felipe.

Não existe exclusividade registrada.

---

51. Credenciais internas

Credenciais internas de teste/equipe ficam registradas no Flow.

Elas não substituem a conta própria do cliente.

PARTE XII — ORIENTAÇÃO DE USO DO ADMIN

52. Demonstração

No fluxo atual, o Admin costuma ser apresentado ao cliente em reunião.

A equipe pode explicar:

- acesso;
- módulos;
- principais ações;
- funcionamento relevante.

A demonstração continua válida mesmo quando o Admin da geração corrente possui módulo próprio de Manual.

---

53. Manual na geração atual × obrigatoriedade universal

A geração atual pode incluir um módulo de **Manual** dentro do Admin, com capítulos por perfil. Esse recurso é posterior a alguns projetos históricos.

Sua existência não transforma o manual em requisito universal e retroativo para toda entrega WSS. Não existe evidência de uma obrigação única e idêntica para todos os projetos.

---

54. O que este Manual não deve fazer

Este Manual não cria:

- apostila universal do Admin;
- treinamento obrigatório padronizado;
- duração mínima de reunião;
- certificação de usuário.

Essas seriam decisões futuras.

PARTE XIII — CRITÉRIO DE ENTREGA INICIAL CONCLUÍDA

55. Três condições principais

O Dev considera a entrega inicial concluída quando:

1. ajustes do cliente foram implementados;
2. site está publicado/entregue;
3. Admin está funcional, validado e entregue.

---

56. Publicar site não encerra sozinho o trabalho

O primeiro publish não representa automaticamente fim da entrega.

Ainda podem existir:

- QA pós-publicação;
- limpeza;
- acesso do cliente;
- demonstração do Admin;
- ajustes finais.

---

57. Entrega inicial × operação recorrente

Depois do encerramento inicial, o projeto pode continuar com o Dev para:

- acompanhamento;
- manutenção;
- ajustes posteriores;
- conteúdo;
- Blog;
- evolução de capacidades;
- atividades recorrentes contratadas.

PARTE XIV — HANDOFF PARA MANUTENÇÃO/RECORRÊNCIA

58. Continuidade

No AS-IS, o projeto continua com o Dev para acompanhamento/manutenção pós-entrega.

A continuidade específica depende:

- contrato;
- escopo;
- serviços recorrentes;
- necessidades do cliente.

---

59. Serviços recorrentes possíveis

Conforme o projeto, podem existir:

- manutenção;
- Blog;
- SEO/AEO/GEO;
- conteúdo;
- outras evoluções.

> **Limite**
>
> Este Manual não define o pacote comercial nem a frequência de cada serviço.

Quando SEO/AEO/GEO recorrente estiver contratado/aplicável, o encaminhamento operacional deve considerar:

- `WSS_SEO_AEO_GEO_SOP_Monitoramento_Recorrente_v0.2`;
- `WSS_SEO_AEO_GEO_SOP_Otimizacao_Artigos_v0.2`;
- `WSS_SEO_AEO_GEO_Matriz_Indicadores_Monitoramento_v0.2`.

A fronteira entre **entrega padrão × serviço recorrente permanece decisão estratégica/comercial aberta** e não é definida por este Manual.

---

60. Relação com o Processo Sites WSS

O Roadmap prevê transição do projeto para:

- operação recorrente;
- uso pelo cliente;
- encerramento.

A formalização de status e registros pertence ao Processo Sites WSS/Flow.

PARTE XV — ANALYTICS, SEARCH CONSOLE E ITENS CONEXOS

61. Analytics no Processo

O Roadmap prevê Analytics “quando aplicável” no Marco 9.

---

62. Analytics no AS-IS do Dev

O Dev declarou que:

- Analytics;
- Tag Manager;

não são normalmente configurados por ele.

Regra de separação

Não usar o Roadmap para afirmar retroativamente que o Dev executa esses itens como rotina pessoal.

Norma atual

O padrão SEO/AEO/GEO v0.2 FINAL mantém **GA4 recomendado, não obrigatório universal**. Analytics/logs first-party podem complementar a medição conforme arquitetura, privacidade e escopo. Essa classificação não transforma GA4 em responsabilidade pessoal automática do Dev.

63. Search Console

AS-IS da responsabilidade do Dev

O Dev participa da configuração do Search Console quando aplicável, podendo usar Codex/Lovable como apoio.

> **NORMA TÉCNICA VIGENTE**
>
> O `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md` estabelece **Google Search Console configurado na publicação** como requisito WSS.

Portanto, “quando aplicável” descreve a participação pessoal do Dev no fluxo observado, **não** a obrigatoriedade do requisito no go-live. O responsável concreto pode variar.

Bing Webmaster Tools é recomendado, não obrigatório universal.

PARTE XVI — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

64. Lista consolidada deste Manual

No AS-IS não existe:

- sequência DNS universal para todo cliente;
- responsabilidade do Dev por GoDaddy/domínio em todos os casos;
- etapa universal separada de “reconfigurar tudo” imediatamente antes do publish;
- smoke build universal obrigatório em todos os sites;
- checklist pós-publicação formal idêntico para todos os projetos;
- manual formal obrigatório do Admin;
- script universal de limpeza de dados;
- playbook universal de incidente de produção;
- ritual formal de rollback por categoria;
- atribuição do Analytics/GTM ao Dev como regra;
- processo comercial único de recorrência dentro do método técnico.

PARTE XVII — OPORTUNIDADES FUTURAS — FORA DO ESCOPO

65. Oportunidades possíveis

Sem desenhar solução:

- formalizar, se futuramente necessário, um checklist geral pré-publicação do Método, distinto dos checklists normativos SEO/AEO/GEO já existentes;
- formalizar, se futuramente necessário, um checklist geral pós-publicação do Método, distinto dos checklists normativos SEO/AEO/GEO já existentes;
- formalizar DNS por provedor;
- formalizar contingência/rollback;
- formalizar limpeza de dados;
- formalizar material de treinamento do Admin;
- definir handoff técnico para recorrência.

Nenhuma dessas oportunidades modifica o AS-IS atual.

PARTE XVIII — BOAS PRÁTICAS RELEVANTES

66. Validar produção de verdade

Depois do publish:

> abrir, navegar, clicar e comprovar.

Não presumir que o ambiente publicado preservou automaticamente o comportamento do preview/dev.

---

67. Não entregar dados de teste

Leads/conversas criados para QA não devem permanecer misturados ao ambiente entregue ao cliente.

---

68. Não entregar acesso interno como conta do cliente

O cliente deve criar o próprio acesso pelo link gerado.

---

69. Restaurar antes de investigar quando produção está amplamente quebrada

Se a publicação derruba o site de forma geral:

> recuperar versão funcional primeiro.

PARTE XIX — ANTI-PATTERNS RELACIONADOS

70. Considerar publish bem-sucedido como QA suficiente

Anti-pattern:

> “publicou, então está funcionando”.

O AS-IS exige revalidação prática em produção.

---

71. Inventar uma receita única de DNS

Não transformar um caso específico em tutorial universal.

---

72. Corrigir produção amplamente quebrada sem considerar rollback

Quando o site inteiro fica comprometido, insistir em correções sucessivas no estado quebrado pode aumentar risco.

O comportamento atual admite retorno à versão funcional.

---

73. Entregar Admin com dados de teste

Deve ser evitado.

PARTE XX — CASO REAL

74. Guilherme Mendonça

O projeto é o principal exemplo corrente da jornada:

```
QA
→ domínio temporário webstarstudio.site
→ revisão do cliente
→ ajustes
→ domínio definitivo
→ produção
→ QA pós-publicação
→ Admin
```

O caso demonstra o uso de domínio temporário antes da entrega final.

---

75. O que não generalizar

Não afirmar a partir desse caso que:

- todos os domínios usam o mesmo apontamento;
- todos os clientes usam GoDaddy como provedor original;
- toda publicação segue os mesmos minutos/horários de propagação;
- todo projeto possui exatamente a mesma sequência de validação.

PARTE XXI — RELAÇÃO COM OS OUTROS MANUAIS

76. Relação com o Manual 4

M4 explica:

- Admin;
- módulos;
- perfis;
- configuração.

M7 explica:

- limpeza;
- criação de acesso;
- entrega ao cliente.

---

77. Relação com o Manual 5

M5 explica rollback/sincronização durante desenvolvimento.

M7 explica contingência quando a falha já está em produção.

---

78. Relação com o Manual 6

M6 é proprietário do método geral de QA/debugging/SEO.

M7 aplica esse método ao ambiente publicado.

---

79. Relação com o Manual 8

M8 concentrará:

- casos de falha;
- troubleshooting;
- anti-patterns;
- lições de publicação.

PARTE XXII — SOP DERIVADO

80. SOP-012 — Preparar, Publicar, Validar Produção e Entregar Admin

Este Manual é a base principal do SOP-012.

O SOP formaliza em sequência executável:

```
confirmar prontidão
→ publicar temporário quando aplicável
→ coletar revisão
→ concluir ajustes
→ definir cenário de domínio
→ coordenar responsável/acesso
→ publicar em produção
→ executar QA pós-publicação
→ tratar falha se houver
→ limpar dados de teste
→ gerar acesso do cliente
→ orientar Admin
→ confirmar entrega
```

---

81. Por que não separar entrega do Admin em outro SOP

A arquitetura editorial aprovada optou por não criar:

- SOP separado de limpeza;
- SOP separado de acesso;
- SOP separado de entrega do Admin.

Essas etapas fazem parte do mesmo fechamento técnico da entrega.

PARTE XXIII — CRITÉRIO DE CONCLUSÃO

82. Critério de conclusão deste Manual

Este Manual cumpre seu objetivo quando o leitor consegue responder:

1. quando o Dev considera o site pronto para publicar;
2. para que serve `webstarstudio.site`;
3. quem pode publicar temporariamente;
4. por que o Admin pode continuar em paralelo;
5. como revisão do cliente entra;
6. quem concentra GoDaddy;
7. quais são os dois cenários de domínio;
8. por que não existe sequência DNS única;
9. o que muda ao passar para produção;
10. o que o Dev verifica depois do publish;
11. por que refresh direto importa;
12. como formulários/Leads/Chat/Blog/Admin são revalidados;
13. como domínio/HTTPS entram;
14. como Analytics aparece sem virar responsabilidade universal do Dev;
15. como SEO aparece na publicação;
16. como diferenciar falha localizada de falha geral;
17. quando rollback é usado;
18. como dados de teste são limpos;
19. como o cliente recebe acesso;
20. como o Admin é normalmente apresentado;
21. quando a entrega inicial termina;
22. como o projeto segue para manutenção/recorrência;
23. quais ausências atuais não devem ser preenchidas com TO-BE.

---

83. Fontes principais desta versão

Baseline canônico

1. `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
2. `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
3. `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
4. `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
5. `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

Contexto operacional

- `Roadmap_Sites_WSS_Modelo_Padrao_v0.2_FINAL.md`.

Manuais já estabilizados usados para preservar fronteiras

- `WSS_Manual_01_Visao_Geral_Toolchain_v1.1_FINAL.md`;
- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`;
- `WSS_Manual_03_Construcao_Visual_v1.1_FINAL.md`;
- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`;
- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`;
- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`.

Normas técnicas aplicáveis à publicação permanecem camada documental separada do AS-IS.

Status: FINAL — APROVADO

WEB STAR STUDIO

MANUAL

MÉTODO DE DESENVOLVIMENTO DE SITES

MANUAL 8
Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais

VERSÃO
v1.1 FINAL

DATA
11/09/2026

STATUS
FINAL — APROVADO

MANUAL

MANUAL 8

Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
ESCOPO DOCUMENTAL	primeira documentação ampliada — P0 + P1 + P2 relevante
PROJETO	WSS — Método de Desenvolvimento de Sites
FORMALIZAÇÃO	Pacote Corretivo Pós-Entrega v1.1 aprovado pela Coordenação em 11/09/2026; versão canônica vigente, com v1.0 FINAL preservado como baseline histórico.

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Objetivo 8

2. Escopo e fronteira 8

3. Regra editorial deste Manual 9

PARTE I — BOAS PRÁTICAS WSS 10

4. Validar desktop, notebook e mobile 11

5. Trabalhar referências por seção 11

6. Preservar áreas já validadas 12

7. Pedir que a IA leia e resuma antes de editar 12

8. Investigar antes de alterar 13

9. Usar a ferramenta conforme a tarefa 13

10. Centralizar rodadas de correção em ajustes.md 13

11. Comparar Base e legado antes de portar 14

12. Não obedecer auditoria externa automaticamente 14

13. Restaurar produção antes de aprofundar incidente geral 15

14. Validar o fluxo completo, não só a UI 15

15. Validar fatos sensíveis com o cliente 15

PARTE II — ANTI-PATTERNS 16

16. Anti-pattern × ausência de padronização 17

17. Prompt “copie este site com extrema fidelidade” 17

18. Aceitar a primeira implementação de IA 17

19. Levar automaticamente efeito desktop para mobile 18

20. Considerar visual sofisticado como substituto de QA 18

21. Corrigir auditoria sem confirmar estado atual 18

22. Interpretar crawler como representação completa da UI 19

23. Reescrever módulo funcional antes de verificar ambiente 19

24. Sobrescrever legado pela Base atual 19

25. Confundir feature nova com paridade 20

26. Considerar publish bem-sucedido como QA suficiente 20
27. Entregar Admin com dados de teste 20
28. Usar conta interna como conta definitiva do cliente 21

PARTE III — PRÁTICAS DEPRECIADAS 22
29. Desenvolver praticamente tudo apenas no Lovable 23
30. Começar todo projeto do zero 23
31. Começar fora do Lovable e trazer depois 24
32. Controlar correções apenas em mensagens salvas do WhatsApp 24

PARTE IV — AUSÊNCIAS QUE NÃO SÃO ANTI-PATTERNS 25
33. Não existe PR obrigatório 26
34. Não existe Design System único universal 26
35. Não existe score universal de performance no AS-IS 26
36. Não existe auditoria formal obrigatória de acessibilidade no AS-IS 26
37. Não existe sequência DNS única 27
38. Não existe root cause framework formal 27

PARTE V — CATÁLOGO DE TROUBLESHOOTING 28
39. Como usar este catálogo 29
TS-001 — Dev funciona, produção retorna HTTP 500 29
TS-002 — Auditoria diz que H1/imagens/schema não existem, mas UI mostra 30
TS-003 — Blog/Admin quebra depois de mudança/retorno de ambiente 30
TS-004 — CTA termina em 404 31
TS-005 — Atualização da Base ameaça quebrar projeto legado 32
TS-006 — Relatório antigo pede correção que já não existe 33
TS-007 — IA começa a alterar áreas não solicitadas 33
TS-008 — Efeito visual quebra notebook/mobile 34
TS-009 — Conflito entre Lovable e trabalho local 35

PARTE VI — ESTUDOS DE CASO 36
40. Estrutura dos casos 37

CASO CR-001 — DR. GUILHERME MENDONÇA 38
41. Papel do caso 39
42. Contexto operacional 39
43. Construção visual 39

44. Execução técnica 40
45. Admin 40
46. QA e auditorias 40
47. Incidente de produção 40
48. Lição transferível 41
49. O que NÃO generalizar 41

CASO CR-002 — DR. GUILHERME MAIA 42
50. Papel do caso 43
51. Contexto técnico 43
52. Problema operacional 43
53. Princípio aplicado 43
54. Exemplo real — Leads 44
55. Escopo novo separado 44
56. Lição transferível 44
57. O que NÃO generalizar 44

CASO CR-003 — DANIEL P 45
58. Papel do caso 46
59. Componentização 46
60. Sofisticação visual 46
61. Lição de arquitetura visual 47
62. Incidente CTA 404 47
63. Auditoria SEOmator 47
64. Acessibilidade pontual 47
65. O que NÃO generalizar 48

CASO CR-004 — MIRELA ALBUQUERQUE 49
66. Papel do caso 50
67. Sintoma 50
68. Investigação 50
69. Causa 50
70. Correção 50
71. Validação 50
72. Lição transferível 51
73. O que NÃO generalizar 51

PARTE VII — LIÇÕES TRANSVERSAIS DOS QUATRO CASOS 52

74. IA precisa de escopo e contexto 53

75. A Base acelera, mas não elimina adaptação 53

76. Visual e técnico são inseparáveis 53

77. Ambiente faz parte do produto 53

78. Auditoria precisa de interpretação 54

PARTE VIII — SINAIS DE ALERTA OPERACIONAIS 55

79. IA começa a inventar texto/requisito 56

80. Efeito pesa ou quebra mobile 56

81. Auditoria contradiz o navegador 56

82. Mudança em Base exige muitas alterações no legado 56

83. Dev saudável + erro em produção 56

84. Módulo inteiro quebra depois de mudança de ambiente 57

PARTE IX — RELAÇÃO COM SOPS 58

85. SOPs que operacionalizam as lições 59

PARTE X — FUTURAS SKILLS 60

86. Este Manual não cria Skills 61

87. Exemplos de conhecimento que poderá virar regra de Skill 61

PARTE XI — AUSÊNCIAS ATUAIS CONSOLIDADAS 62

88. Ausências relevantes já documentadas 63

PARTE XII — OPORTUNIDADES FUTURAS — FORA DO ESCOPO 64

89. Possíveis evoluções 65

PARTE XIII — NAVEGAÇÃO DA COLEÇÃO 66

90. Para entender o método inteiro 67

91. Para contexto e prompting 67

92. Para construção visual 67

93. Para Base/Admin 67

94. Para Codex/Lovable/GitHub 67

95. Para QA/debugging/SEO/segurança 67

96. Para produção/entrega 68

PARTE XIV — CRITÉRIO DE CONCLUSÃO 69

97. Critério de conclusão deste Manual 70

98. Fontes principais desta versão 70

1. Objetivo

Este Manual concentra o **conhecimento acumulado** do Método de Desenvolvimento de Sites WSS.

Sua função não é repetir os procedimentos dos Manuais 1–7.

Ele organiza:

- boas práticas reconhecidas no AS-IS;
- anti-patterns;
- práticas depreciadas;
- diferenças entre “não padronizado” e “não fazer”;
- catálogo inicial de troubleshooting;
- estudos de caso reais;
- lições transferíveis;
- limites de generalização.

Seu papel é responder perguntas como:

- o que costuma funcionar bem;
- o que costuma dar errado;
- o que a WSS deixou de fazer;
- que sinais merecem atenção;
- quais incidentes reais já ocorreram;
- como foram diagnosticados;
- qual Manual/SOP deve ser consultado para executar o procedimento correto.

Este Manual permanece **AS-IS**. Ele não transforma lições em um novo processo TO-BE.

---

2. Escopo e fronteira

2.1 O que entra

Este Manual cobre:

1. boas práticas;
2. anti-patterns;
3. práticas depreciadas;
4. ausências de padronização que não são anti-patterns;
5. troubleshooting;
6. casos reais;
7. lições;
8. sinais de alerta;
9. referências cruzadas para Manuais/SOPs.

2.2 O que não entra

Ficam fora:

- passo a passo completo de contextualização;
- passo a passo de Git;
- construção detalhada de Home;
- configuração completa do Admin;
- execução completa de QA;
- procedimento completo de publicação;
- desenho de Skills;
- novos padrões TO-BE.

Quando um item exigir execução, este Manual referencia o proprietário correto.

---

3. Regra editorial deste Manual

Para evitar duplicação:

> **M8 registra a lição; o Manual proprietário explica o método; o SOP executa a rotina.**

Exemplo:

```
M8:
“Auditoria externa pode contradizer a UI.”

↓ referência

M6:
explica como investigar.

↓ referência

SOP-010:
transforma em sequência executável.
```

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE I — BOAS PRÁTICAS WSS

4. Validar desktop, notebook e mobile

Uma das práticas mais recorrentes é não considerar o site pronto apenas com uma visualização.

O Dev valida:

- desktop;
- notebook;
- mobile físico.

Por que funciona

Ajuda a detectar:

- quebras;
- crop inadequado;
- overflow;
- efeitos instáveis;
- textos grandes/pequenos;
- diferenças de comportamento.

Ver

- Manual 3 — Construção Visual;
- Manual 6 — Testes, QA, Debugging, Segurança e SEO.

---

5. Trabalhar referências por seção

Em vez de pedir à IA que reproduza um site inteiro:

1. escolher seção;
2. fornecer print;
3. fornecer link;
4. pedir estudo;
5. limitar escopo;
6. comparar;
7. refinar.

Por que funciona

Reduz:

- ambiguidade;
- simplificação;
- mudança colateral;
- perda de fidelidade.

Ver

- Manual 3;
- SOP-003;
- SOP-004.

---

6. Preservar áreas já validadas

Quando uma alteração é localizada, o Dev explicita:

- o que pode mudar;
- o que não pode mudar.

Exemplos:

- mobile sem alterar desktop;
- Admin sem mexer no institucional;
- backend sem mudar UI;
- módulo sem quebrar demais módulos.

Lição

Escopo aberto demais aumenta regressão.

---

7. Pedir que a IA leia e resuma antes de editar

Em nova sessão, o Dev costuma pedir:

- leitura do projeto;
- leitura dos documentos;
- entendimento de WSS/cliente;
- resumo antes de seguir.

Lição

Resumo funciona como teste de contexto.

Ver

- Manual 2;
- Manual 5.

---

8. Investigar antes de alterar

Quando há:

- efeito complexo;
- bug difícil;
- auditoria contraditória;
- mudança de Base;
- comportamento desconhecido;

o Dev pode primeiro pedir investigação.

Lição

Não transformar primeira hipótese da IA em implementação automática.

---

9. Usar a ferramenta conforme a tarefa

Prática atual:

- Lovable: início do projeto e criação inicial da estrutura herdada de backend/banco/auth no remix da Base; no pós-criação, integração e bugs funcionais quando esse ambiente é adequado;
- Codex: código, investigação, fidelidade, implementação pesada e, no pós-criação, condução de intervenções que podem operar/orquestrar o Lovable;
- Claude: refinamento visual/acabamento;
- humano: julgamento, risco, escopo.

Lição

Não existe benefício em forçar toda tarefa para a mesma ferramenta.

---

10. Centralizar rodadas de correção em `ajustes.md`

Fluxo:

- registrar;
- executar;
- marcar;
- revisar lista inteira;
- revalidar áreas afetadas.

Lição

Uma lista persistente reduz perda de pedido e conflito entre ajustes.

---

11. Comparar Base e legado antes de portar

Em projeto antigo:

- inventariar;
- identificar equivalentes;
- identificar deltas;
- separar feature nova;
- preservar o que já funciona melhor.

Lição

Base atual é referência funcional, não ordem para sobrescrever o destino.

Ver

- Manual 4;
- SOP-007.

---

12. Não obedecer auditoria externa automaticamente

Fluxo observado:

```
auditoria
→ estado atual
→ classificação
→ delta real
→ correção
→ validação
```

Lição

Crawler, relatório ou IA podem:

- estar desatualizados;
- não enxergar CSR;
- reportar falso positivo;
- pedir algo já resolvido.

Norma atual

O checklist de um caso real continua não universal, mas a WSS possui agora o `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md` e anexos canônicos próprios. A lição "não obedecer auditoria automaticamente" continua válida: a verificação deve comparar o achado com o estado atual e com a norma vigente.

---

13. Restaurar produção antes de aprofundar incidente geral

Quando o site inteiro fica amplamente quebrado:

- recuperar versão funcional;
- investigar depois.

Lição

Disponibilidade tem prioridade sobre análise extensa enquanto o site está fora.

---

14. Validar o fluxo completo, não só a UI

Exemplo Leads:

```
formulário
→ sucesso
→ Admin
→ Lead
→ origem
→ dados
```

Lição

Um frontend aparentemente correto pode não estar persistindo o resultado.

---

15. Validar fatos sensíveis com o cliente

IA pode criar/adaptar grande parte do conteúdo.

Não deve inventar:

- títulos;
- formação;
- locais;
- procedimentos;
- resultados;
- afirmações médicas;
- dados factuais sensíveis.

Lição

Fluência textual não equivale a factualidade.

PARTE II — ANTI-PATTERNS

16. Anti-pattern × ausência de padronização

É importante diferenciar.

Anti-pattern

Há evidência de que a prática tende a causar problema ou deve ser evitada.

Ausência de padronização

Não existe regra formal hoje.

Uma ausência não é automaticamente erro.

Exemplo:

- não ter PR obrigatório → ausência;
- aceitar primeira implementação de IA sem validar → anti-pattern.

---

17. Prompt “copie este site com extrema fidelidade”

Problema

Link isolado + pedido amplo.

Riscos

- baixa fidelidade;
- simplificação;
- interpretação errada;
- alteração excessiva.

Alternativa AS-IS

Trabalhar por seção com screenshot + link + estudo.

---

18. Aceitar a primeira implementação de IA

Problema

A primeira versão pode:

- estar visualmente fraca;
- inventar conteúdo;
- quebrar área validada;
- não compreender referência.

Regra prática

IA entrega → humano valida.

---

19. Levar automaticamente efeito desktop para mobile

Problema

Efeito pode:

- pesar;
- quebrar;
- reduzir legibilidade;
- criar scroll horizontal.

Prática correta

Priorizar adaptar; se necessário, simplificar ou substituir; remover apenas quando a solução continuar inviável ou prejudicial.

---

20. Considerar visual sofisticado como substituto de QA

Caso Daniel P mostrou CTA quebrado em projeto visualmente sofisticado.

Lição

Motion/3D não elimina necessidade de testar:

- link;
- rota;
- CTA;
- formulário.

---

21. Corrigir auditoria sem confirmar estado atual

Problema

Pode refazer item já resolvido.

Prática correta

Classificar:

- presente;
- parcial;

- resolvido;
- não reproduzido.

---

22. Interpretar crawler como representação completa da UI

Caso Daniel:

- crawler reportou H1/imagens ausentes;
- UI possuía esses elementos;
- conteúdo client-side não estava visível à coleta.

Lição

Perguntar:

> qual representação a ferramenta está enxergando?

---

23. Reescrever módulo funcional antes de verificar ambiente

Caso Mirela:

- Blog quebrado;
- causa estava em configuração/key Supabase.

Anti-pattern

Reescrever Blog antes de checar:

- env;
- key;
- conexão;
- ambiente.

---

24. Sobrescrever legado pela Base atual

Problema

Pode perder:

- funcionalidade local;

- dados;
- comportamento superior;
- particularidade do cliente.

Prática correta

Portabilidade incremental/adaptativa.

---

25. Confundir feature nova com paridade

Caso Guilherme Maia:

"Jornada do paciente" não era simples atualização da Base.

Lição

Delta da Base ≠ autorização para criar escopo novo.

---

26. Considerar publish bem-sucedido como QA suficiente

Problema

Produção pode apresentar:

- problema de rota;
- binding;
- domínio;
- HTTPS;
- build;
- integração.

Prática correta

QA pós-publicação.

---

27. Entregar Admin com dados de teste

Problema

Mistura:

- Leads falsos;
- conversas de QA;
- registros demonstrativos;

com operação real.

Prática correta

Limpar dados de teste antes da entrega.

---

28. Usar conta interna como conta definitiva do cliente

Prática correta

Gerar link para criação do acesso próprio do cliente.

PARTE III — PRÁTICAS DEPRECIADAS

29. Desenvolver praticamente tudo apenas no Lovable

Status

DEPRECIADA

Antes

Lovable concentrava praticamente todo o trabalho.

Hoje

Fluxo híbrido:

```
Lovable
+ GitHub
+ local
+ Codex
+ Claude
```

Motivo

Maior flexibilidade para:

- fidelidade;
- implementação pesada;
- investigação;
- refinamento.

---

30. Começar todo projeto do zero

Status

DEPRECIADA

Hoje

Projeto novo parte normalmente da Base Admin.

Motivo

Reutilizar:

- Admin;
- Blog;
- Leads;
- Chat/CTA;

- demais capacidades.

---

31. Começar fora do Lovable e trazer depois

Status

DEPRECIADA como fluxo normal

A experiência gerou dificuldades.

Hoje

Lovable/Base primeiro → GitHub/local depois.

---

32. Controlar correções apenas em mensagens salvas do WhatsApp

Status

DEPRECIADA como prática preferencial

Evolução

Hoje o Dev prefere:

- `ajustes.md`;
- Flow quando possível.

> **Importante**
>
> WhatsApp continua sendo canal real de feedback.

A prática depreciada é o **controle informal exclusivo**, não o canal em si.

PARTE IV — AUSÊNCIAS QUE NÃO SÃO ANTI-PATTERNS

33. Não existe PR obrigatório

Isso é ausência de padronização.

Não documentar como “erro”.

---

34. Não existe Design System único universal

É ausência.

Projetos podem possuir:

- tokens;
- componentes;
- layouts;
- decisões locais.

---

35. Não existe score universal de performance no AS-IS

É ausência da prática histórica documentada.

Performance é hoje pragmática/perceptiva no método do Dev.

Camada normativa atual

O `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md` possui metas **recomendadas** de Core Web Vitals e prioriza dados de campo quando disponíveis. Isso não transforma um score de laboratório em gate histórico nem em garantia de ranking.

---

36. Não existe auditoria formal obrigatória de acessibilidade no AS-IS

É ausência da prática histórica documentada.

Há atenção consciente, mas não auditoria WCAG universal.

Camada normativa atual

O padrão v0.2 FINAL possui requisitos/classificações específicos de semântica e mídia, incluindo `alt` útil/contextual para imagens informativas. Isso não deve ser reescrito como se uma auditoria formal de acessibilidade sempre tivesse feito parte do fluxo.

---

37. Não existe sequência DNS única

É ausência condicionada pelos cenários/provedores.

38. Não existe root cause framework formal

Debugging atual é pragmático.

Isso não deve ser rotulado automaticamente como anti-pattern.

PARTE V — CATÁLOGO DE TROUBLESHOOTING

39. Como usar este catálogo

As entradas abaixo servem para:

- reconhecer sintomas;
- orientar a investigação inicial;
- apontar o Manual/SOP correto.

Elas não substituem diagnóstico.

Formato:

```
Sintoma
→ evidência
→ investigação
→ causa conhecida
→ correção
→ validação
```

TS-001 — Dev funciona, produção retorna HTTP 500

Caso

Guilherme Mendonça.

Sintoma

- servidor dev saudável;
- build/publicação falha;
- rotas retornam 500.

Investigação

- gerar artefato de produção;
- executar pacote real;
- testar rotas;
- separar falta de binding local de falha do bundle.

Causa real do caso

Problema de chunking/bundling no stack de produção.

Correção/contingência

- smoke de produção;
- bloquear publish quando falha;
- corrigir/contornar versão problemática.

Validação

Rotas reais retornam status esperado no artefato de produção.

Ver

- Manual 6.

TS-002 — Auditoria diz que H1/imagens/schema não existem, mas UI mostra

Caso

Daniel P / SEOmator.

Sintoma

Relatório externo contradiz navegador.

Investigação

1. verificar HTML bruto;
2. verificar SSR/CSR;
3. descobrir o que o crawler consegue ver;
4. comparar UI × resposta inicial.

Causa real do caso

Parte do conteúdo era materializada client-side e não estava visível ao crawler.

Correção

Não “adicionar H1” cegamente.

Corrigir a causa real quando necessária ou classificar limitação/falso positivo.

Ver

- Manual 6;
- SOP-010.

TS-003 — Blog/Admin quebra depois de mudança/ retorno de ambiente

Caso

Mirela Albuquerque.

Sintoma

Funcionalidade antes estável deixa de operar.

Investigação

- console;
- env;
- URL Supabase;
- publishable key;
- conexão Lovable Cloud.

Causa real

Configuração/key Supabase ausente/incorreta.

Correção

Restaurar configuração correta.

Validação

Fluxo volta a funcionar sem reescrever o módulo.

Ver

- Manual 6;
- SOP-009.

TS-004 — CTA termina em 404

Caso

Daniel P.

Sintoma

Clique navega para rota inexistente.

Investigação

- identificar `href/path`;
- comparar com tabela real de rotas.

Causa real

CTA usava `/sobre-mim`; rota era `/sobre`.

Correção

Ajustar path.

Validação

Reexecutar clique.

Ver

- Manual 6.

TS-005 — Atualização da Base ameaça quebrar projeto legado

Caso

Guilherme Maia.

Sintoma

Diferenças extensas entre Base e destino.

Investigação

1. inventariar origem/destino;
2. equivalentes;
3. deltas;
4. features novas;
5. dependências;
6. particularidades locais.

Causa

Não necessariamente existe “bug”.

O problema é tratar evolução como substituição integral.

Ação

Portar apenas delta aprovado.

Validação

Capacidade funciona e particularidades locais permanecem.

Ver

- Manual 4;
- SOP-007.

TS-006 — Relatório antigo pede correção que já não existe

Sintoma

Auditoria/plano descreve problema incompatível com estado atual.

Investigação

- comparar relatório com `main`;
- reproduzir;
- classificar.

Possíveis estados

- presente;
- parcial;
- já resolvido;
- não reproduzido.

Ação

Alterar apenas o delta real.

Ver

- Manual 6;
- SOP-010.

TS-007 — IA começa a alterar áreas não solicitadas

Sintoma

- desktop muda em tarefa mobile;
- outra seção é reescrita;
- design se afasta;
- requisito novo aparece.

Investigação

Verificar:

- contexto;
- escopo do prompt;
- arquivos lidos;
- tamanho da tarefa.

Ação

1. interromper;
2. reforçar contexto;
3. limitar escopo;
4. dividir tarefa;
5. revalidar área preservada.

Ver

- Manual 2;
- Manual 5.

TS-008 — Efeito visual quebra notebook/mobile

Sintoma

- overflow;
- scroll horizontal;
- borda quebrada;
- 3D instável.

Investigação

- breakpoint;
- dimensões;
- efeito;
- peso;
- comportamento por dispositivo.

Ação

1. adaptar;
2. simplificar ou substituir/criar solução específica;
3. remover somente se a solução continuar inviável ou prejudicial.

Ver

- Manual 3;
- SOP-003/004.

TS-009 — Conflito entre Lovable e trabalho local

Sintoma

Versões divergem ou mudança é sobrescrita.

Investigação

- identificar alterações dos dois lados;
- verificar último estado funcional;
- localizar versão estável.

Ação

Preservar:

- versão funcional;
- menor dano;
- trabalho válido.

> ### Limite
>
> Não existe source-of-truth formal universal.

Ver

- Manual 5.

PARTE VI — ESTUDOS DE CASO

40. Estrutura dos casos

Cada caso registra:

1. papel do caso;
2. contexto;
3. prática/problema;
4. evidência;
5. decisão;
6. resultado;
7. lição;
8. o que não generalizar.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

CASO CR-001 — DR. GUILHERME MENDONÇA

41. Papel do caso

Principal caso representativo do fluxo corrente.

Exemplifica:

```
handoff
→ contexto
→ prompt master
→ Base Admin/Lovable
→ GitHub/local
→ Codex/Claude
→ Home
→ páginas
→ Admin
→ QA
→ cliente
→ ajustes
→ auditorias
→ produção
→ entrega
```

---

42. Contexto operacional

O projeto foi usado como referência principal da descoberta atual porque representa:

- stack/fluxo atual;
- Base Admin;
- uso híbrido de ferramentas;
- publicação temporária;
- Admin em paralelo;
- auditorias atuais.

---

43. Construção visual

O caso ilustra:

- referência principal;
- referências secundárias;
- Home como âncora;
- páginas internas;
- adaptação visual ao cliente.

---

44. Execução técnica

Também representa:

- Lovable primeiro;
- GitHub depois;
- clone local;
- Codex/Claude;
- retorno ao Lovable via GitHub.

---

45. Admin

Mostra:

- capacidades herdadas da Base;
- Blog;
- Leads;
- Chat/CTA;
- Chatbot;
- perfis.

---

46. QA e auditorias

O projeto possui evidências de:

- SEO técnico;
- AEO/GEO;
- metadados;
- schema;
- robots;
- sitemap;
- build/SSR;
- smoke de produção.

---

47. Incidente de produção

Houve falha:

- dev saudável;
- build real com 500.

O projeto ganhou smoke test de produção.

---

48. Lição transferível

> A integração híbrida acelera desenvolvimento, mas produção precisa ser validada como ambiente próprio.

---

49. O que NÃO generalizar

Não usar como método:

- esboço antecipado ocorrido antes do encerramento de todas as definições do cliente.

Foi exceção operacional.

Também não afirmar que:

- todo projeto precisa do mesmo smoke;
- todo projeto usa exatamente a mesma stack;
- todo projeto recebe o mesmo nível de auditoria.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

CASO CR-002 — DR. GUILHERME MAIA

50. Papel do caso

Principal caso de:

- projeto antigo;
- evolução da Base;
- portabilidade de capacidades.

---

51. Contexto técnico

Projeto de geração anterior:

- React;
- Vite;
- React Router;
- Supabase;
- Admin já desenvolvido.

---

52. Problema operacional

A Base atual evoluiu.

O projeto antigo possui capacidades próprias.

Pergunta:

> como atualizar sem perder o que já funciona?

---

53. Princípio aplicado

Portabilidade incremental/adaptativa:

```
comparar
→ equivalentes
→ deltas
→ dependências
→ preservar local
→ portar
→ validar
```

---

54. Exemplo real — Leads

A análise identificou capacidades já existentes:

- CRUD;
- filtros;
- score;
- timeline;
- exportação.

Não foram propostas para serem refeitas.

Deltas úteis incluíam:

- KPIs;
- gráficos;
- ações em lote.

---

55. Escopo novo separado

“Jornada do paciente” foi deixada fora por representar:

funcionalidade nova

e não simples paridade.

---

56. Lição transferível

> Base é referência para evolução, não autoridade para substituição integral.

---

57. O que NÃO generalizar

Não afirmar que:

- todo projeto antigo possui o mesmo stack;
- toda migração tem os mesmos módulos;
- existe migration universal;
- existe rollback universal.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

CASO CR-003 — DANIEL P

58. Papel do caso

Principal caso de:

- sofisticação visual;
- componentização;
- Design System híbrido;
- motion;
- performance;
- QA visual;
- SEOmator.

---

59. Componentização

Foram confirmados:

- componentes compartilhados;
- props/variantes;
- layouts orientados por dados;
- tokens CSS;
- decisões locais.

---

60. Sofisticação visual

Foram confirmados:

- Framer Motion;
- GSAP;
- ScrollTrigger;
- SplitText;
- Three;
- React Three Fiber;
- Drei;
- lazy loading;
- chunking.

---

61. Lição de arquitetura visual

> Sofisticação pode coexistir com reuso e contenção de performance.

Mas não implica Design System completamente formal.

---

62. Incidente CTA 404

CTA apontava para rota inexistente.

Lição:

> QA básico continua necessário em UI sofisticada.

---

63. Auditoria SEOmator

O relatório indicava ausências que não correspondiam à UI.

Diagnóstico:

- crawler via HTML inicial;
- conteúdo client-side invisível.

Lição:

> interpretar a forma de coleta antes de editar.

---

64. Acessibilidade pontual

Foram encontrados:

- texto de 7px em mobile;
- botão-ícone sem nome acessível.

Isso demonstra atenção e correção contextual.

Não prova auditoria WCAG formal.

---

65. O que NÃO generalizar

Não afirmar que todo site deve usar:

- Three;
- GSAP;
- 3D;
- SplitText;
- mesma arquitetura de componentes;
- checklist AEO/GEO do Daniel.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

CASO CR-004 — MIRELA ALBUQUERQUE

66. Papel do caso

Principal caso de:

- problema/recuperação;
- configuração de ambiente;
- Supabase/Lovable;
- debugging pragmático.

---

67. Sintoma

Ao retornar o projeto ao Lovable:

- Blog deixou de funcionar.

---

68. Investigação

Console indicou problema relacionado a:

- Supabase;
- configuração;
- key.

---

69. Causa

Configuração/key ausente ou incorreta.

---

70. Correção

Ajuste do ambiente/Lovable.

---

71. Validação

O Blog voltou a funcionar.

---

72. Lição transferível

> Antes de reescrever código funcional depois de mudança de ambiente, verificar configuração, env e conexão.

---

73. O que NÃO generalizar

Não afirmar que todo problema de Blog é Supabase.

O caso serve como padrão de diagnóstico:

sintoma → camada → evidência → correção.

PARTE VII — LIÇÕES TRANSVERSAIS DOS QUATRO CASOS

74. IA precisa de escopo e contexto

Guilherme/Daniel demonstram:

- ferramentas são potentes;
- resultado depende de contexto;
- escopo excessivo aumenta desvio.

---

75. A Base acelera, mas não elimina adaptação

Guilherme Mendonça:

- Base reduz reconstrução.

Guilherme Maia:

- Base não pode simplesmente sobrescrever legado.

---

76. Visual e técnico são inseparáveis

Daniel:

- motion/3D;
- performance;
- links;
- acessibilidade;
- SEO.

Uma página visualmente excelente ainda pode falhar tecnicamente.

---

77. Ambiente faz parte do produto

Mirela:

- configuração pode quebrar funcionalidade.

Guilherme:

- build de produção pode quebrar apesar do dev.

---

78. Auditoria precisa de interpretação

Daniel/Guilherme:

- relatórios podem estar antigos;
- crawler pode não enxergar tudo;
- implementação real precisa ser confirmada.

PARTE VIII — SINAIS DE ALERTA OPERACIONAIS

79. IA começa a inventar texto/requisito

Sinal de perda de contexto.

Ação:

- interromper;
- reler;
- resumir;
- reduzir escopo.

---

80. Efeito pesa ou quebra mobile

Sinal de que fidelidade não deve ser preservada literalmente.

---

81. Auditoria contradiz o navegador

Sinal para investigar:

- HTML;
- SSR/CSR;
- crawler;
- versão analisada.

---

82. Mudança em Base exige muitas alterações no legado

Sinal para separar:

- paridade;
- delta;
- feature nova.

---

83. Dev saudável + erro em produção

Sinal para:

- build real;
- bindings;
- runtime;

- smoke.

---

84. Módulo inteiro quebra depois de mudança de ambiente

Sinal para verificar:

- env;
- keys;
- service connection;
- configuração.

PARTE IX — RELAÇÃO COM SOPs

85. SOPs que operacionalizam as lições

Este Manual referencia:

- SOP-001 — Contextualizar um Novo Projeto WSS;
- SOP-002 — Iniciar Tecnicamente um Projeto WSS: Base → Lovable → GitHub → Ambiente Local;
- SOP-003 — Construir e Refinar a Home a partir das Referências;
- SOP-004 — Construir Páginas Internas Mantendo Consistência Visual;
- SOP-005 — Configurar Admin e Módulos Aplicáveis;
- SOP-006 — Configurar e Validar Chatbot e Base de Conhecimento;
- SOP-007 — Portar Capacidade da Base para Projeto Legado;
- SOP-008 — Executar QA Visual, Funcional e Responsivo;
- SOP-009 — Diagnosticar e Corrigir Bug por Camada;
- SOP-010 — Tratar Auditoria SEO, AEO e GEO;
- SOP-011 — Processar Rodada de Feedback via `ajustes.md`;
- SOP-012 — Preparar, Publicar, Validar Produção e Entregar Admin.

Os 12 SOPs já foram produzidos e aprovados materialmente. Sua formalização versionada ocorre em gate posterior ao fechamento dos Manuais.

PARTE X — FUTURAS SKILLS

86. Este Manual não cria Skills

Skills serão derivadas depois.

O M8 poderá contribuir especialmente com:

- erros conhecidos;
- anti-patterns;
- stop conditions;
- contingências;
- casos de teste;
- sinais de escalonamento humano.

---

87. Exemplos de conhecimento que poderá virar regra de Skill

Uma futura Skill poderá incorporar:

- não alterar fora do escopo;
- pedir resumo antes de editar;
- não obedecer auditoria cegamente;
- não inventar fatos;
- retornar à gestão diante de escopo novo;
- restaurar produção antes de investigação longa;
- preservar capacidades melhores no legado.

Nenhuma Skill é criada nesta versão.

PARTE XI — AUSÊNCIAS ATUAIS CONSOLIDADAS

88. Ausências relevantes já documentadas

Entre outras:

- sem matriz formal universal de ferramentas;
- sem hierarquia documental universal;
- sem template único de prompt master;
- sem política formal universal de branching;
- sem PR obrigatório;
- sem source-of-truth universal em conflito;
- sem rollback universal;
- sem Design System único;
- sem score universal de performance;
- sem auditoria formal obrigatória de acessibilidade;
- sem suíte universal de testes;
- sem root cause framework formal;
- sem checklist universal de segurança;
- sem DNS universal;
- sem manual Admin obrigatório.

> **Importante**
>
> Esses itens **não devem ser automaticamente rotulados como defeitos**.

São descrições do AS-IS.

PARTE XII — OPORTUNIDADES FUTURAS — FORA DO ESCOPO

89. Possíveis evoluções

Já apareceram como oportunidades:

- centralizar demandas no Flow;
- permitir anexos de imagem/áudio nas Tasks;
- segunda camada de validação humana;
- formalizar assets;
- formalizar acessibilidade;
- formalizar rollback;
- formalizar branching;
- formalizar DNS;
- formalizar handoff;
- evoluir controle de pendências;
- criar Skills.

Nenhuma delas é desenhada neste Manual.

PARTE XIII — NAVEGAÇÃO DA COLEÇÃO

90. Para entender o método inteiro

Consultar:

Manual 1 — Visão Geral e Toolchain

91. Para contexto e prompting

Consultar:

Manual 2 — Entrada e Contexto do Projeto

92. Para construção visual

Consultar:

Manual 3 — Construção Visual

93. Para Base/Admin

Consultar:

Manual 4 — Projeto-Base, Admin e Funcionalidades

94. Para Codex/Lovable/GitHub

Consultar:

Manual 5 — Desenvolvimento com Codex, Lovable e GitHub

95. Para QA/debugging/SEO/segurança

Consultar:

Manual 6 — Testes, QA, Debugging, Segurança e SEO

96. Para produção/entrega

Consultar:

Manual 7 — Publicação, Entrega e Pós-Publicação

PARTE XIV — CRITÉRIO DE CONCLUSÃO

97. Critério de conclusão deste Manual

Este Manual cumpre seu objetivo quando o leitor consegue:

1. distinguir boa prática de anti-pattern;
2. distinguir ausência de padronização de anti-pattern;
3. reconhecer práticas depreciadas;
4. reconhecer os principais sintomas de troubleshooting;
5. localizar o Manual/SOP proprietário;
6. compreender os quatro casos;
7. extrair lições transferíveis;
8. entender o que não deve ser generalizado;
9. reconhecer sinais que exigem investigação;
10. entender como esses achados poderão alimentar Skills futuras.

---

98. Fontes principais desta versão

Baseline canônico

1. `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
2. `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
3. `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
4. `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
5. `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

Evidências técnicas complementares

- `WSS_VAL_001_DanielP_Componentizacao_DesignSystem_Visual_v0.1.md`;
- `WSS_VAL_002_BaseAdmin_Chatbot_v0.1.md`;
- `WSS_VAL_003_Auditorias_SEO_AEO_GEO_Seguranca_v0.1.md`;
- `WSS_VAL_004_Troubleshooting_Estudos_de_Caso_v0.1.md`.

Manuais estabilizados usados como referência cruzada

- Manual 1 — Visão Geral e Toolchain;
- Manual 2 — Entrada e Contexto do Projeto;
- Manual 3 — Construção Visual;
- Manual 4 — Projeto-Base, Admin e Funcionalidades;
- Manual 5 — Desenvolvimento com Codex, Lovable e GitHub;
- Manual 6 — Testes, QA, Debugging, Segurança e SEO;
- Manual 7 — Publicação, Entrega e Pós-Publicação.

Norma técnica vigente relacionada

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL.md`.

Versões anteriores/superadas de AS-IS, Matriz, Handoff e Plano não foram usadas como baseline ativo.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-001
Contextualizar um Novo Projeto WSS

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-001

Contextualizar um Novo Projeto WSS

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 2 — Entrada e Contexto do Projeto
ESCOPO	AS-IS — contextualização do handoff WSS antes do início técnico do projeto

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 5

2. Gatilho — quando usar 5

3. Quando NÃO usar 6

4. Papéis 6

5. Pré-condições 7

6. Inputs 7

7. Ferramentas e acessos 8

PARTE I — PROCEDIMENTO 9

8. Passo 1 — Reunir o pacote de handoff 10

9. Passo 2 — Verificar se existe regra explícita de precedência 10

10. Passo 3 — Carregar os materiais na IA de contextualização 11

11. Passo 4 — Consolidar o entendimento do projeto 11

12. Passo 5 — Classificar pendências 12

13. Passo 6 — Tratar placeholders com cautela 13

14. Passo 7 — Produzir um resumo de entendimento 14

15. Passo 8 — Corrigir desvios antes de gerar o prompt master 14

16. Passo 9 — Preparar o prompt master 15

17. Passo 10 — Delimitar o que o prompt master NÃO pode fazer 16

18. Passo 11 — Identificar contexto persistente relevante 16

PARTE II — DECISÕES CONDICIONAIS 17

19. Contradição entre documentos 18

20. Informação faltante 18

21. IA não compreendeu o projeto 18

22. IA inventou fato sensível 18

23. Arquivo não está acessível 19

PARTE III — VALIDAÇÃO E EVIDÊNCIAS 20

24. Evidências mínimas de conclusão 21

25. Checklist de validação do entendimento 21

PARTE IV — STOP CONDITION **22**

26. Condição de parada 23

PARTE V — CONTINGÊNCIA **24**

27. Se o contexto continuar insuficiente 25

PARTE VI — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO **26**

28. Não inventar no SOP 27

PARTE VII — NORMA TÉCNICA VIGENTE **28**

29. Tratamento de normas 29

PARTE VIII — ERROS CONHECIDOS / ANTI-PATTERNS **30**

30. Iniciar com leitura parcial do handoff 31

31. Escolher sozinho qual documento “vence” 31

32. Pedir à IA para “entender tudo” sem verificar resumo 31

33. Gerar prompt master antes de resolver desvio evidente 31

34. Esperar todas as pendências para começar 31

35. Inventar dado sensível para “completar” a experiência 31

PARTE IX — OUTPUT **32**

36. Saída esperada 33

PARTE X — PRÓXIMO PASSO **34**

37. Encaminhamento 35

38. Referências 35

1. Finalidade

Transformar o handoff documental recebido da WSS em **entendimento operacional suficiente para iniciar tecnicamente o projeto**, produzindo o prompt master e deixando explícitos:

- objetivo;
- cliente;
- posicionamento;
- público;
- Sitemap;
- direção visual;
- referências;
- conteúdo relevante;
- módulos/capacidades;
- restrições;
- pendências;
- riscos/pontos de atenção;
- caminho inicial de desenvolvimento.

Este SOP não cria um novo modelo de briefing, não redefine a hierarquia documental e não inicia o desenvolvimento técnico.

---

2. Gatilho — quando usar

Usar quando:

- um novo projeto de site foi formalmente entregue ao Dev pela WSS;
- existe handoff documental suficiente para começar a preparação técnica;
- o projeto ainda precisa ser compreendido e organizado antes do primeiro trabalho no Lovable.

Fluxo:

```
handoff WSS
→ SOP-001
→ contexto compreendido
→ prompt master
→ SOP-002 / início técnico
```

---

3. Quando NÃO usar

Não usar como substituto de:

- Briefing Curado;
- Sitemap;
- Contexto de Desenvolvimento Consolidado;
- kickoff;
- definição de escopo;
- investigação de bug;
- QA;
- construção visual;
- configuração do Admin;
- trabalho Git/local.

Também não usar para reconstruir todo o histórico do Processo Sites WSS quando o pacote de handoff já contém informação suficiente.

---

4. Papéis

4.1 WSS / Direção / Operações

Responsável por disponibilizar o handoff e resolver, quando necessário:

- contradições materiais;
- dúvidas de escopo;
- decisões não resolvidas;
- possíveis mudanças de escopo.

4.2 Desenvolvedor

Responsável por:

- ler/organizar os materiais;
- consolidar entendimento;
- identificar lacunas;
- distinguir bloqueios de pendências;
- preparar o prompt master;
- validar o entendimento antes de iniciar.

4.3 IA de contextualização

No AS-IS, normalmente:

- ChatGPT pessoal.

Excepcionalmente, Claude pode auxiliar quando o resultado inicial não satisfaz.

A IA organiza e resume; não decide autonomamente escopo, fatos sensíveis ou contradições materiais.

---

5. Pré-condições

Antes de iniciar, deve existir um pacote de handoff com o núcleo disponível.

Núcleo esperado

1. **Contexto de Desenvolvimento Consolidado**;
2. **Sitemap**;
3. **Briefing Final**.

Materiais complementares, quando existirem

- arquivos `.md`;
- referência visual principal;
- referências secundárias;
- screenshots;
- fotos;
- links;
- conteúdos;
- documentos adicionais;
- materiais do cliente.

Se faltar um dos três documentos centrais

Não inventar seu conteúdo.

Avaliar se o material disponível ainda permite entendimento suficiente.

Se a ausência impedir compreensão de escopo, arquitetura pública ou decisões centrais:

> **DECISÃO HUMANA NECESSÁRIA — solicitar o material/clareza correspondente.**

---

6. Inputs

O executor deve reunir em uma única sessão de contextualização, conforme disponibilidade:

- Contexto de Desenvolvimento Consolidado;
- Sitemap;
- Briefing Final;
- referências visuais;
- arquivos `.md`;

- materiais do cliente;
- conteúdos já aprovados;
- pendências registradas;
- restrições normativas explicitamente indicadas no pacote.

Não presumir que a IA possui acesso a arquivos apenas porque eles existem em outro ambiente.

---

7. Ferramentas e acessos

Ferramenta normal

- ChatGPT pessoal do Dev.

Ferramenta complementar

- Claude, quando necessário.

Não necessário nesta etapa

- GitHub;
- Codex no repositório;
- Lovable;
- DevTools;
- banco/Supabase.

Essas ferramentas entram em etapas posteriores.

PARTE I — PROCEDIMENTO

8. Passo 1 — Reunir o pacote de handoff

Confirmar quais materiais foram recebidos.

Registrar mentalmente ou em nota de trabalho:

- presentes;
- ausentes;
- inacessíveis.

Não fazer

- preencher documento ausente por inferência;
- assumir que Sitemap está contido integralmente no Briefing;
- assumir que Contexto Consolidado substitui automaticamente as demais fontes.

---

9. Passo 2 — Verificar se existe regra explícita de precedência

Antes de reconciliar documentos, procurar se:

- o Contexto Consolidado;
- outro documento do projeto;
- o próprio pacote;

define uma ordem de autoridade.

Se existir regra explícita

Respeitá-la naquele projeto.

Se NÃO existir

Aplicar o AS-IS:

- Contexto Consolidado é concebido pela WSS como principal artefato de entrada;
- Dev consulta Contexto, Sitemap, Briefing e complementos em conjunto;
- nenhuma dessas fontes “vence” automaticamente por regra universal.

Se houver contradição material

Não decidir por conta própria.

> **DECISÃO HUMANA NECESSÁRIA.**

Material = divergência que afeta, por exemplo:

- escopo;
- página;
- direção visual;
- conteúdo factual;
- funcionalidade;
- módulo;
- decisão de cliente;
- publicação futura.

---

10. Passo 3 — Carregar os materiais na IA de contextualização

Fornecer os documentos e referências à sessão utilizada.

Orientar a IA a:

1. ler todos os materiais;
2. organizar o projeto;
3. identificar pontos centrais;
4. separar decisão confirmada de pendência;
5. não inventar requisitos;
6. apontar contradições relevantes;
7. produzir um resumo antes de qualquer implementação.

> **Limite**
>
> O objetivo da sessão é **entender e preparar**.

Não desenvolver o site dentro desta etapa.

---

11. Passo 4 — Consolidar o entendimento do projeto

A contextualização precisa permitir responder, no mínimo:

Cliente e negócio

- quem é o cliente;
- qual o objetivo do site;
- qual posicionamento;
- qual público;

- qual tom/linguagem.

Estrutura

- qual Sitemap;
- quais páginas principais;
- quais áreas prioritárias.

Visual

- qual direção visual;
- referência principal;
- referências secundárias;
- identidade/paleta quando já definidas;
- restrições visuais.

Conteúdo

- quais conteúdos já existem;
- quais fatos estão confirmados;
- quais ainda estão pendentes.

Operacional

- módulos/capacidades aplicáveis;
- CTAs;
- canais;
- integrações explicitamente indicadas;
- restrições relevantes.

Risco/contexto

- pendências;
- decisões que ainda dependem de humano;
- pontos que podem bloquear uma etapa;
- pontos que podem avançar provisoriamente.

---

12. Passo 5 — Classificar pendências

Não assumir que toda pendência bloqueia o desenvolvimento.

Para cada ausência relevante, perguntar:

> **essa informação é indispensável para iniciar a construção ou apenas para concluir/publicar uma parte específica?**

Se bloqueia entendimento estrutural

Exemplos:

- Sitemap contraditório;
- escopo não definido;
- direção central conflitante;
- módulo essencial indefinido.

Aguardar decisão humana correspondente antes de iniciar a parte afetada.

Se não bloqueia o início

Registrar como pendência operacional.

Exemplos possíveis:

- telefone;
- foto;
- dado factual específico;
- informação complementar.

> **Atenção**
>
> O caso Dr. Guilherme tratou suas pendências como não bloqueantes salvo quando indispensáveis a conteúdo factual.

Essa formulação **não é automaticamente universal**.

A classificação depende do impacto concreto.

---

13. Passo 6 — Tratar placeholders com cautela

Quando houver dado operacional simples ausente, o AS-IS admite placeholder em alguns casos.

Exemplo declarado:

- telefone provisório no formato `(81) 9 9999-9999`.

> **Regra**
>
> Placeholder:

- deve permanecer reconhecível como provisório;
- não vira informação validada;
- não deve mascarar fato sensível.

Nunca inventar como placeholder factual

- formação;
- título;
- procedimento;
- resultado;
- especialidade;
- afirmação médica;
- outro fato sensível do cliente.

---

14. Passo 7 — Produzir um resumo de entendimento

Pedir à IA um resumo do projeto.

O resumo deve permitir ao Dev validar que a IA compreendeu:

- cliente;
- objetivo;
- estrutura;
- direção;
- restrições;
- módulos;
- pendências;
- caminho inicial.

Validação

Não existe formulário automático de aceite.

O Dev revisa o resumo por julgamento humano.

---

15. Passo 8 — Corrigir desvios antes de gerar o prompt master

Sinais de desvio:

- textos aleatórios;
- requisito não pedido;
- direção visual incompatível;
- página inexistente no escopo;
- contradição ignorada;
- invenção factual.

Se ocorrer:

1. interromper;
2. indicar o desvio;
3. reforçar documentos relevantes;
4. reduzir escopo;
5. pedir novo resumo;
6. só avançar após entendimento aceitável.

---

16. Passo 9 — Preparar o prompt master

Depois de o Dev considerar o contexto suficientemente compreendido, produzir o prompt master.

Função do prompt master

Levar o entendimento acumulado para o início técnico do projeto, principalmente no Lovable.

Cobertura mínima esperada

Sem impor template fixo, o prompt master deve carregar contexto suficiente sobre:

- cliente;
- objetivo;
- público;
- posicionamento;
- Sitemap;
- direção visual;
- referências;
- identidade/diretrizes existentes;
- conteúdo/restrições;
- módulos aplicáveis;
- CTAs;
- pendências relevantes;
- limites importantes;
- caminho inicial esperado.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe template canônico único de prompt master.

Não criar seções artificiais apenas para “cumprir formato”.

O critério é suficiência de contexto.

---

17. Passo 10 — Delimitar o que o prompt master NÃO pode fazer

Reforçar quando necessário:

- não inventar fatos;
- não criar funcionalidade fora do escopo;
- não resolver contradição por conta própria;
- não substituir documento canônico por inferência;
- não ignorar pendências;
- não transformar referência visual em cópia literal sem adaptação.

---

18. Passo 11 — Identificar contexto persistente relevante

Antes de encerrar, identificar o que deverá permanecer disponível ao longo do desenvolvimento.

Exemplos observados:

- `contexto.md`;
- `design.md`;
- demais `.md`;
- documentos do projeto.

> **Limite**
>
> Este SOP não cria obrigação de:

- gerar `contexto.md` antes do Lovable;
- atualizar `contexto.md` a cada prompt;
- atualizar `design.md` a cada troca de ferramenta.

A cadência não é padronizada no AS-IS.

O objetivo aqui é garantir que o executor saiba **quais referências precisarão permanecer acessíveis**.

PARTE II — DECISÕES CONDICIONAIS

19. Contradição entre documentos

Existe regra explícita de precedência?

Sim → seguir a regra do projeto.

Não → não escolher vencedor automaticamente.

Se material:

> retornar à decisão humana.

---

20. Informação faltante

Bloqueia início/parte imediata?

Sim → obter decisão/dado antes daquela execução.

Não → registrar pendência e seguir.

---

21. IA não compreendeu o projeto

Não gerar prompt master definitivo.

Executar novamente:

```
corrigir contexto
→ reduzir escopo
→ reler documentos
→ novo resumo
```

---

22. IA inventou fato sensível

Rejeitar o conteúdo inventado.

Voltar à fonte humana/documental.

Não “ajustar por plausibilidade”.

---

23. Arquivo não está acessível

Reenviar o arquivo necessário.

Não presumir memória ou acesso da ferramenta.

PARTE III — VALIDAÇÃO E EVIDÊNCIAS

24. Evidências mínimas de conclusão

Ao encerrar o SOP, devem existir:

1. materiais de handoff lidos;
2. resumo de entendimento revisado pelo Dev;
3. contradições materiais resolvidas ou explicitamente escaladas;
4. pendências não bloqueantes registradas;
5. fatos sensíveis sem invenção;
6. prompt master pronto para uso;
7. referências que precisarão persistir identificadas.

---

25. Checklist de validação do entendimento

Antes de encerrar, o Dev deve conseguir responder “sim” às perguntas:

- sei quem é o cliente?
- sei qual é o objetivo do site?
- sei qual é o público/posicionamento?
- sei qual Sitemap devo seguir?
- sei qual direção visual/referências devo usar?
- sei quais módulos/capacidades se aplicam?
- sei quais restrições precisam ser preservadas?
- sei quais informações continuam pendentes?
- sei quais pendências bloqueiam e quais não?
- há algum conflito documental material ainda não resolvido?
- consigo explicar por onde o desenvolvimento deve começar?
- o prompt master reflete esse entendimento?

Se a resposta material for “não”, não encerrar.

PARTE IV — STOP CONDITION

26. Condição de parada

O SOP está concluído quando:

> **o Dev possui entendimento operacional suficiente para iniciar tecnicamente o projeto, o prompt master está pronto, não há contradição material não tratada e as pendências não bloqueantes estão explícitas.**

Não exigir para encerrar

- todos os dados finais do cliente;
- todas as imagens;
- todos os fatos futuros de publicação;
- ausência total de pendências;
- criação de repositório;
- remix da Base;
- implementação de página.

Esses itens pertencem a outras etapas.

PARTE V — CONTINGÊNCIA

27. Se o contexto continuar insuficiente

Se, mesmo após releitura, o executor não consegue determinar:

- escopo;
- estrutura;
- direção;
- módulos;
- caminho inicial;

não iniciar tecnicamente por inferência.

Escalar somente a lacuna concreta para:

- Direção/Operações;
- responsável pelo projeto;
- Dev/gestão conforme natureza.

Regra de contenção

Não reabrir investigação histórica ampla.

Perguntar/validar apenas o que impede reprodução.

PARTE VI — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

28. Não inventar no SOP

O AS-IS não possui:

- hierarquia universal Contexto × Sitemap × Briefing;
- template canônico único de prompt master;
- formulário automatizado de aceite do entendimento;
- regra de atualização de `contexto.md` por interação;
- regra de atualização de `design.md` por interação;
- mecanismo automático que impeça invenção da IA;
- mecanismo automático universal para resolver pendências.

Essas ausências permanecem válidas.

PARTE VII — NORMA TÉCNICA VIGENTE

29. Tratamento de normas

Se o pacote referenciar uma norma técnica vigente:

- identificar a norma;
- preservar seus requisitos como camada própria;
- não reescrever o AS-IS;
- não ignorar exigência explicitamente aplicável.

Quando a norma envolver SEO/AEO/GEO, a autoridade canônica vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Este SOP não é o local para reinterpretar ou redesenhar a norma.

PARTE VIII — ERROS CONHECIDOS / ANTI-PATTERNS

30. Iniciar com leitura parcial do handoff

Risco:

- perda de requisito;
- direção errada;
- retrabalho.

---

31. Escolher sozinho qual documento “vence”

Risco:

- inventar hierarquia inexistente.

---

32. Pedir à IA para “entender tudo” sem verificar resumo

Risco:

- falsa sensação de contexto.

---

33. Gerar prompt master antes de resolver desvio evidente

Risco:

- propagar erro para Lovable e demais ferramentas.

---

34. Esperar todas as pendências para começar

Pode criar bloqueio artificial.

Classificar impacto primeiro.

---

35. Inventar dado sensível para “completar” a experiência

Proibido pelo AS-IS documentado.

PARTE IX — OUTPUT

36. Saída esperada

Ao final, entregar:

Obrigatório

- **Resumo de entendimento do projeto** validado pelo Dev;
- **Prompt master** pronto para uso no início técnico;
- **Lista de pendências relevantes**, com indicação prática de bloqueio quando necessário;
- **Lista de contradições materiais**, se existirem, com status de decisão.

Referências persistentes identificadas

- Contexto Consolidado;
- Sitemap;
- Briefing;
- referências;
- `contexto.md` / `design.md` quando existentes ou adotados no projeto;
- demais arquivos necessários.

PARTE X — PRÓXIMO PASSO

37. Encaminhamento

Com o SOP-001 concluído, seguir para:

SOP-002 — Iniciar tecnicamente um projeto: Base → Lovable → GitHub → local

O SOP-002 começa no ponto em que:

- contexto está suficientemente compreendido;
- prompt master está pronto.

---

38. Referências

Manual proprietário

- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`

Manuais relacionados

- Manual 1 — Visão Geral e Toolchain;
- Manual 3 — Construção Visual;
- Manual 5 — Desenvolvimento com Codex, Lovable e GitHub;
- Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais.

Baseline canônico

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
- `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-002
Iniciar Tecnicamente um Projeto WSS: Base → Lovable → GitHub → Ambiente Local

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-002

Iniciar Tecnicamente um Projeto WSS: Base → Lovable → GitHub → Ambiente Local

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 5 — Desenvolvimento com Codex, Lovable e GitHub
MANUAIS RELACIONADOS	Manual 2 — Entrada e Contexto do Projeto; Manual 4 — Projeto-Base, Admin e Funcionalidades
ESCOPO	AS-IS — preparação técnica de um novo projeto WSS até a primeira rodada local

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 5

2. Gatilho — quando usar 5

3. Quando NÃO usar 5

4. Papéis 6

5. Pré-condições 6

6. Inputs 7

7. Ferramentas 7

PARTE I — PROCEDIMENTO 9

8. Passo 1 — Remixar a Base Admin no Lovable 10

9. Passo 2 — Aplicar o prompt master e os materiais 10

10. Passo 3 — Produzir a primeira base do institucional 10

11. Passo 4 — Chegar ao primeiro ou segundo protótipo 11

12. Passo 5 — Verificar se o protótipo corresponde ao contexto 11

13. Passo 6 — Conectar o projeto ao GitHub 12

14. Passo 7 — Clonar o repositório para a máquina 12

15. Passo 8 — Abrir o projeto na ferramenta local 13

16. Passo 9 — Validar entendimento local 13

17. Passo 10 — Confirmar branch e estado atual 13

18. Passo 11 — Atualizar o ambiente local antes da primeira rodada 14

PARTE II — VALIDAÇÃO DO SETUP 15

19. Checklist de prontidão técnica 16

20. Evidência mínima de conclusão 16

PARTE III — CONDICIONAIS 17

21. O Lovable gerou uma primeira versão fraca 18

22. O Lovable gerou requisito fora do escopo 18

23. O GitHub ainda não deve entrar 18

24. A tarefa exige mudança radical já no início 18

25. Codex e Claude divergem sobre o projeto 19

PARTE IV — STOP CONDITION **20**

26. Condição de parada 21

27. O que NÃO é necessário para encerrar 21

PARTE V — CONTINGÊNCIA **22**

28. Falha na conexão Lovable ↔ GitHub 23

29. Clone local desatualizado 23

30. Contexto não está acessível localmente 23

PARTE VI — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO **24**

31. Não inventar 25

PARTE VII — NORMA TÉCNICA VIGENTE **26**

32. Tratamento 27

PARTE VIII — ERROS CONHECIDOS / ANTI-PATTERNS **28**

33. Começar fora do Lovable e tentar trazer depois 29

34. Permanecer todo o projeto apenas no Lovable 29

35. Clonar e editar sem atualizar o repositório 29

36. Editar localmente sem validar contexto 29

37. Criar branch/PR/tag como obrigação documental 29

PARTE IX — OUTPUT **30**

38. Saída esperada 31

PARTE X — PRÓXIMO PASSO **32**

39. Encaminhamento 33

40. Relação com o fluxo contínuo depois da inicialização 33

41. Referências 33

1. Finalidade

Transformar um projeto já contextualizado em um ambiente técnico pronto para o fluxo híbrido atual da WSS:

```
contexto pronto
→ Base Admin
→ Lovable
→ primeiros protótipos
→ GitHub
→ clone local
→ Codex / Claude
→ primeira rodada local pronta
```

O SOP termina quando o projeto:

- já existe como remix da Base no Lovable;
- já recebeu o contexto/prompt master;
- já possui primeira base/protótipo;
- já está conectado ao GitHub;
- já está clonado localmente;
- já pode ser lido pelas ferramentas locais;
- já está atualizado para iniciar trabalho local.

---

2. Gatilho — quando usar

Usar quando o **SOP-001 — Contextualizar um Novo Projeto WSS** estiver concluído.

Pré-condição conceitual:

```
contexto compreendido
+ prompt master pronto
→ iniciar SOP-002
```

---

3. Quando NÃO usar

Não usar:

- para projeto legado que já possui arquitetura própria e precisa receber capacidade da Base;
- para investigar bug em projeto existente;
- para reconstruir histórico técnico;
- para fazer QA;

- para publicar;
- para configurar todos os módulos do Admin;
- para construir a Home em profundidade.

Projetos antigos/legados devem seguir o procedimento de portabilidade correspondente.

---

4. Papéis

4.1 Desenvolvedor

Responsável por:

- remixar a Base;
- aplicar contexto;
- gerar primeira versão;
- conectar GitHub;
- clonar;
- abrir localmente;
- validar que o projeto está pronto para a primeira rodada técnica.

4.2 IA / ferramentas

Podem participar:

- Lovable;
- Codex;
- Claude.

Nenhuma ferramenta substitui o julgamento humano sobre:

- escopo;
- risco;
- versão correta;
- contexto;
- necessidade de mudança estrutural.

---

5. Pré-condições

Antes de iniciar:

1. SOP-001 concluído;
2. prompt master pronto;
3. materiais principais disponíveis;
4. Base Admin atual identificada como ponto de partida;

5. acesso ao Lovable;
6. acesso ao GitHub;
7. máquina/ambiente local disponível para clone;
8. ferramenta local escolhida disponível, normalmente Codex e/ou Claude.

---

6. Inputs

Obrigatórios

- prompt master;
- Contexto de Desenvolvimento Consolidado;
- Sitemap;
- Briefing Final;
- referências visuais;
- demais arquivos essenciais do projeto.

Quando existirem

- `contexto.md`;
- `design.md`;
- outros `.md`;
- assets;
- materiais do cliente.

---

7. Ferramentas

Lovable

Usado para:

- remix da Base;
- primeiros prompts;
- primeira versão/protótipos;
- origem normal do projeto.

GitHub

Usado como ponte entre:

- Lovable;
- ambiente local;
- Codex/Claude.

Ambiente local

Usado para:

- leitura;
- desenvolvimento;
- refinamento;
- investigação.

Codex / Claude

Entram depois do clone conforme a tarefa.

PARTE I — PROCEDIMENTO

8. Passo 1 — Remixar a Base Admin no Lovable

Criar o novo projeto a partir da **Base Admin WSS atual** usando remix.

Objetivo

Começar com:

- estrutura funcional reutilizável;
- Admin;
- Blog;
- módulos/capacidades já disponíveis na Base.

> **Importante**
>
> O projeto deve começar como novo projeto, sem depender do histórico de chat do projeto-base.

A Base é ponto de partida técnico, não produto final do cliente.

---

9. Passo 2 — Aplicar o prompt master e os materiais

No projeto remixado:

1. enviar o prompt master;
2. disponibilizar documentos e referências necessárias;
3. reforçar restrições importantes;
4. orientar a primeira geração do institucional.

Não fazer

- depender de memória implícita do Lovable;
- assumir que o remix trouxe contexto do cliente;
- tratar conteúdo demo da Base como conteúdo final;
- inventar fatos ausentes.

---

10. Passo 3 — Produzir a primeira base do institucional

Usar os primeiros prompts para gerar uma primeira versão utilizável.

O objetivo aqui não é concluir o visual.

É criar uma base suficiente para:

- materializar o projeto;

- confirmar que o contexto foi aplicado;
- permitir o próximo estágio híbrido.

---

11. Passo 4 — Chegar ao primeiro ou segundo protótipo

No AS-IS, o GitHub entra normalmente depois do primeiro ou segundo protótipo.

Critério prático

O projeto deve possuir base suficiente para seguir localmente.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe:

- número rígido de prompts;
- score;
- checklist automatizado;
- momento exato universal.

A decisão é prática e humana.

---

12. Passo 5 — Verificar se o protótipo corresponde ao contexto

Antes de conectar o GitHub, confirmar que o projeto não saiu claramente do escopo.

Verificar, no mínimo:

- cliente correto;
- direção inicial coerente;
- Sitemap/estrutura reconhecível;
- ausência de requisito inventado evidente;
- Base correta.

Se houver desvio forte

Corrigir contexto antes de avançar.

Não propagar um projeto claramente mal contextualizado para o fluxo local.

---

13. Passo 6 — Conectar o projeto ao GitHub

Depois da primeira base:

1. conectar o projeto Lovable ao GitHub;
2. criar/confirmar o repositório correspondente;
3. verificar que o projeto está representado no repositório.

Objetivo

Estabelecer a ponte:

```
Lovable
↔ GitHub
↔ local
```

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Este SOP não define:

- convenção universal de nome;
- organização obrigatória de repositórios;
- PR obrigatório;
- tags obrigatórias.

---

14. Passo 7 — Clonar o repositório para a máquina

Após a conexão, clonar o repositório para o ambiente local.

O Dev faz isso cedo no fluxo.

Resultado esperado

O projeto local deve conter:

- código;
- documentos;
- configurações versionadas;
- demais artefatos presentes no repositório.

> **Limite**
>
> Este SOP não prescreve comando Git específico.

---

15. Passo 8 — Abrir o projeto na ferramenta local

Abrir o repositório em:

- Codex;
- Claude;
- ou ambos, conforme necessidade.

Pedir à ferramenta

- ler o projeto;
- entender a estrutura;
- ler os `.md`;
- identificar contexto do cliente;
- resumir antes de editar.

Se um arquivo necessário não estiver acessível

Reenviar ou disponibilizar o arquivo.

Não presumir acesso.

---

16. Passo 9 — Validar entendimento local

Antes da primeira alteração local, confirmar que a ferramenta consegue explicar:

- o que é o projeto;
- quem é o cliente;
- qual é a direção;
- qual é a estrutura;
- quais áreas já existem;
- quais restrições principais devem ser preservadas.

Se houver perda de contexto

1. interromper;
2. reforçar documentos;
3. pedir nova leitura;
4. pedir novo resumo;
5. só então iniciar alteração.

---

17. Passo 10 — Confirmar branch e estado atual

No AS-IS, a prática predominante é trabalhar na:

`main`

> **Regra**
>
> Registrar isso como prática atual, não como norma universal de Git.

Branch separada

Pode ser usada em mudança radical.

Quando uma rodada usa branch ou Pull Request, o Codex costuma criar esses artefatos sob instrução/supervisão do Dev. Isso não torna branch/PR obrigatórios, não cria etapa manual fixa para o Dev e não estabelece convenção de nomenclatura.

Não é necessária para a inicialização normal.

---

18. Passo 11 — Atualizar o ambiente local antes da primeira rodada

Antes de iniciar a primeira rodada local:

- trazer a versão mais recente disponível do repositório.

No AS-IS, o Dev faz pull antes de iniciar trabalho local.

Objetivo

Evitar começar sobre cópia desatualizada.

PARTE II — VALIDAÇÃO DO SETUP

19. Checklist de prontidão técnica

Antes de encerrar o SOP, confirmar:

- [ ] projeto foi remixado da Base Admin atual;
- [ ] prompt master foi aplicado;
- [ ] documentos/referências essenciais foram fornecidos;
- [ ] primeira base/protótipo existe;
- [ ] contexto não apresenta desvio material evidente;
- [ ] projeto está conectado ao GitHub;
- [ ] repositório foi clonado;
- [ ] ferramenta local consegue ler o projeto;
- [ ] contexto local foi resumido/validado;
- [ ] branch/estado atual foram entendidos;
- [ ] ambiente local foi atualizado antes da primeira rodada.

---

20. Evidência mínima de conclusão

O executor deve conseguir mostrar ou confirmar:

1. projeto Lovable existente;
2. repositório GitHub correspondente;
3. cópia local do repositório;
4. primeira versão/protótipo;
5. ferramenta local com compreensão suficiente;
6. ambiente pronto para alteração.

PARTE III — CONDICIONAIS

21. O Lovable gerou uma primeira versão fraca

Não é motivo automático para abandonar o fluxo.

Perguntar:

- o contexto está correto?
- a base foi aplicada?
- o protótipo é suficiente para seguir localmente?

Se sim:

> seguir para GitHub/local e refinar depois.

O visual final pertence ao Manual 3 / SOP-003.

---

22. O Lovable gerou requisito fora do escopo

Corrigir antes de seguir.

Não conectar ao fluxo local uma base que já carregue erro estrutural evidente sem tratá-lo.

---

23. O GitHub ainda não deve entrar

Se o projeto ainda não possui base minimamente utilizável:

- continuar os primeiros prompts no Lovable.

> **Limite**
>
> Não prolongar indefinidamente.

O fluxo atual leva GitHub/local cedo.

---

24. A tarefa exige mudança radical já no início

Branch separada pode ser considerada.

DECISÃO HUMANA NECESSÁRIA

Não existe threshold formal que obrigue branch.

---

25. Codex e Claude divergem sobre o projeto

Não escolher automaticamente pelo nome da ferramenta.

Voltar às fontes:

- código;
- contexto;
- documentos;
- estado real do projeto.

Preservar a leitura sustentada pelo projeto.

PARTE IV — STOP CONDITION

26. Condição de parada

O SOP está concluído quando:

> **o projeto já existe no Lovable a partir da Base, recebeu o contexto, possui primeira base/protótipo, está conectado ao GitHub, foi clonado localmente, a ferramenta local compreende o projeto e o ambiente está atualizado para iniciar a primeira rodada local.**

---

27. O que NÃO é necessário para encerrar

Não exigir nesta etapa:

- Home final;
- páginas internas concluídas;
- Admin totalmente configurado;
- módulos testados;
- QA completo;
- build de produção;
- SEO;
- segurança;
- publicação;
- domínio.

PARTE V — CONTINGÊNCIA

28. Falha na conexão Lovable ↔ GitHub

Se a conexão não se estabelece:

- não criar fluxos paralelos improvisados;
- verificar a conexão/configuração correspondente;
- resolver o bloqueio técnico antes de tratar o projeto local como integrado.

> **Limite**
>
> Este SOP não inventa procedimento técnico específico de recuperação que não esteja documentado nas fontes.

---

29. Clone local desatualizado

Antes de editar:

- atualizar.

Não começar deliberadamente sobre estado antigo.

---

30. Contexto não está acessível localmente

Disponibilizar os arquivos necessários.

Não assumir que o código sozinho substitui:

- Briefing;
- Sitemap;
- Contexto;
- `design.md`.

PARTE VI — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

31. Não inventar

O AS-IS não possui:

- nomenclatura universal de repositório;
- número rígido de prompts antes do GitHub;
- comando Git obrigatório documentado;
- política formal universal de branching;
- PR obrigatório;
- tags obrigatórias;
- convenção universal de commit;
- build obrigatório como parte do setup;
- uso obrigatório simultâneo de Codex e Claude;
- checklist automático para decidir “hora de clonar”.

PARTE VII — NORMA TÉCNICA VIGENTE

32. Tratamento

Se o projeto estiver sujeito a norma técnica vigente:

- ela continua válida independentemente da ferramenta;
- não deve ser removida pelo remix;
- não deve ser ignorada no primeiro protótipo;
- deve permanecer acessível ao desenvolvimento posterior.

Para SEO/AEO/GEO, a autoridade vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Lovable, Codex, Claude ou outra ferramenta não devem degradar, por conveniência de implementação, os requisitos públicos aplicáveis definidos nessa norma.

Este SOP não redefine a norma.

PARTE VIII — ERROS CONHECIDOS / ANTI-PATTERNS

33. Começar fora do Lovable e tentar trazer depois

Prática depreciada como fluxo normal.

O AS-IS atual começa no Lovable/Base.

---

34. Permanecer todo o projeto apenas no Lovable

Prática depreciada.

O fluxo atual incorpora GitHub/local cedo.

---

35. Clonar e editar sem atualizar o repositório

Risco:

- divergência;
- sobrescrita;
- conflito.

---

36. Editar localmente sem validar contexto

Risco:

- mudança tecnicamente correta para o projeto errado;
- perda de direção;
- requisito inventado.

---

37. Criar branch/PR/tag como obrigação documental

Não pertence ao AS-IS universal.

Pode existir caso a caso.

PARTE IX — OUTPUT

38. Saída esperada

Ao final, devem existir:

No Lovable

- projeto remixado da Base;
- contexto aplicado;
- primeira base/protótipo.

No GitHub

- repositório conectado ao projeto.

Localmente

- clone do repositório;
- ferramenta local com acesso;
- contexto compreendido;
- estado atualizado.

Estado operacional

- pronto para primeira rodada de desenvolvimento local.

PARTE X — PRÓXIMO PASSO

39. Encaminhamento

Depois do SOP-002:

Construção visual

Seguir para:

SOP-003 — Construir e Refinar a Home a partir das Referências

quando o foco for Home/direção visual.

Configuração operacional

O Admin pode evoluir em paralelo conforme:

- Manual 4;
- SOP-005 — Configurar Admin e Módulos Aplicáveis.

---

40. Relação com o fluxo contínuo depois da inicialização

Depois deste SOP, o ciclo normal passa a ser:

```
confirmar contexto
→ pull
→ implementar/investigar
→ validar
→ commit
→ push
→ Lovable atualiza
→ nova rodada
```

Esse ciclo pertence ao Manual 5 e se repete durante o desenvolvimento.

---

41. Referências

Manual proprietário

- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`;
- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

SOP anterior

- `WSS_SOP_001_Contextualizar_Novo_Projeto_WSS_v1.1_FINAL.md`

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
- `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-003
Construir e Refinar a Home a partir das Referências

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-003

Construir e Refinar a Home a partir das Referências

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 3 — Construção Visual
MANUAIS RELACIONADOS	Manual 2 — Entrada e Contexto do Projeto; Manual 5 — Desenvolvimento com Codex, Lovable e GitHub; Manual 6 — Testes, QA, Debugging, Segurança e SEO
ESCOPO	AS-IS — construção e refinamento da Home como âncora visual do projeto

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 6

2. Gatilho — quando usar 6

3. Quando NÃO usar 6

4. Papéis 7

5. Pré-condições 8

6. Inputs 8

PARTE I — PREPARAÇÃO DA RODADA VISUAL 9

7. Passo 1 — Confirmar o estado atual da Home 10

8. Passo 2 — Relembrar a direção visual 10

9. Passo 3 — Separar referência de cópia literal 10

PARTE II — COMPARAÇÃO E REFINAMENTO POR SEÇÃO 12

10. Passo 4 — Escolher uma seção por vez 13

11. Passo 5 — Fornecer evidência concreta 13

12. Passo 6 — Pedir estudo antes de editar quando necessário 13

13. Passo 7 — Delimitar o escopo da alteração 14

14. Passo 8 — Buscar fidelidade estrutural e visual 14

15. Passo 9 — Adaptar para o cliente 15

PARTE III — HERO E ÁREAS DE MAIOR PESO VISUAL 16

16. Passo 10 — Validar o Hero e priorizar o esforço por valor 17

17. Passo 11 — Manter consistência entre seções 17

PARTE IV — SUBFLUXO DE EFEITOS E ANIMAÇÕES 18

18. Quando aplicar 19

19. Passo E1 — Identificar o efeito 19

20. Passo E2 — Fornecer screenshot e link 19

21. Passo E3 — Investigar técnica/biblioteca 19

22. Passo E4 — Reproduzir isoladamente 20

23. Passo E5 — Adaptar ao projeto 20

24. Passo E6 — Validar desktop/notebook/mobile 20

25. Quando adaptar, simplificar, substituir ou remover o efeito 21

PARTE V — ASSETS E CONTEÚDO NA HOME **22**

26. Passo 12 — Aplicar assets disponíveis 23

27. Passo 13 — Tratar conteúdo com fidelidade factual 23

PARTE VI — RESPONSIVIDADE DURANTE A CONSTRUÇÃO **24**

28. Passo 14 — Não esperar o QA final para olhar mobile 25

29. Passo 15 — Preservar breakpoints já aprovados 25

30. Solução diferente por dispositivo 25

PARTE VII — PERFORMANCE E ACESSIBILIDADE NO NÍVEL DA CONSTRUÇÃO **26**

31. Passo 16 — Observar peso percebido 27

32. Passo 17 — Corrigir problemas óbvios de acessibilidade 27

PARTE VIII — CICLO ITERATIVO **28**

33. Passo 18 — Comparar novamente com a referência 29

34. Passo 19 — Refinar apenas o delta relevante 29

35. Passo 20 — Validar manualmente 29

PARTE IX — CRITÉRIO DE PRONTO **30**

36. Condição de pronto da Home 31

37. Relação com o Marco 5 do Processo Sites WSS 31

38. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — score visual 31

PARTE X — STOP CONDITION **33**

39. Condição de parada 34

40. O que NÃO é necessário para encerrar 34

PARTE XI — CONTINGÊNCIAS **35**

41. A referência não funciona bem para o cliente 36

42. A IA perde a direção visual 36

43. Um efeito quebra mobile/notebook 36

44. A primeira versão do Lovable está muito simples 36

PARTE XII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO **37**

45. Não inventar 38

PARTE XIII — NORMA TÉCNICA VIGENTE **39**

46. Preservar normas aplicáveis 40

PARTE XIV — ANTI-PATTERNS **41**

47. Enviar apenas um link e pedir “extrema fidelidade” 42

48. Aceitar a primeira geração como final 42

49. Alterar várias seções quando o problema é local 42

50. Levar automaticamente efeito desktop para mobile 42

51. Preservar efeito quebrado em nome da referência 42

PARTE XV — OUTPUT **43**

52. Saída esperada 44

PARTE XVI — PRÓXIMO PASSO **45**

53. Encaminhamento 46

54. Referências 46

1. Finalidade

Transformar a primeira base do institucional em uma **Home suficientemente refinada para estabelecer a linguagem visual do projeto**.

Fluxo principal:

```
primeira versão
→ comparar com referência
→ trabalhar por seção
→ refinar
→ adaptar ao cliente
→ tratar efeitos quando aplicável
→ ajustar responsividade
→ validar visualmente
→ Home estabelece linguagem do projeto
```

A Home funciona como principal âncora para as páginas internas.

---

2. Gatilho — quando usar

Usar quando:

- o SOP-002 já deixou o projeto tecnicamente pronto;
- existe primeira versão/protótipo;
- a referência principal está definida;
- referências secundárias relevantes estão disponíveis;
- o contexto do cliente está compreendido;
- o objetivo é consolidar a direção visual da Home.

---

3. Quando NÃO usar

Não usar:

- para definir briefing ou direção visual ainda não validada;
- para selecionar referências do zero sem contexto do projeto;
- para construir página interna;
- para corrigir bug técnico não visual;
- para executar QA completo do site;
- para configurar Admin;
- para publicar.

---

4. Papéis

4.1 Desenvolvedor

Responsável por:

- julgar fidelidade;
- julgar adequação ao cliente;
- escolher o grau de adaptação;
- limitar escopo;
- validar desktop/notebook/mobile;
- decidir quando a Home já estabelece linguagem suficiente.

4.2 Codex

Uso recorrente quando a prioridade é:

- fidelidade à referência;
- investigação técnica;
- implementação pesada;
- análise de comportamento/efeito.

4.3 Claude

Uso recorrente para:

- refinamento visual;
- tipografia;
- acabamento;
- adaptação de composição;
- efeitos;
- mobile.

4.4 Lovable

Pode permanecer no fluxo para:

- primeira base;
- pequenos ajustes;
- tarefas funcionais;
- intervenções em que sua integração seja mais conveniente.

A divisão entre ferramentas não é rígida.

5. Pré-condições

Antes de iniciar:

1. SOP-001 concluído;
2. SOP-002 concluído;
3. Home inicial existente;
4. referência principal identificada;
5. características de referências secundárias conhecidas, quando existirem;
6. Contexto/Sitemap/Briefing acessíveis;
7. `design.md` disponível quando adotado no projeto;
8. ambiente local atualizado;
9. ferramenta utilizada compreende o projeto antes de editar.

---

6. Inputs

Obrigatórios

- Home atual;
- referência principal;
- contexto do cliente;
- direção visual;
- Sitemap;
- conteúdo disponível.

Quando existirem

- screenshots;
- referências secundárias;
- `design.md`;
- identidade visual;
- paleta;
- assets;
- fotos;
- vídeos;
- exemplos de efeitos;
- links de Awwwards ou outras fontes.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE I — PREPARAÇÃO DA RODADA VISUAL

7. Passo 1 — Confirmar o estado atual da Home

Antes de editar:

1. abrir a Home existente;
2. observar a versão atual;
3. confirmar que o projeto está atualizado;
4. identificar o que já está aceitável;
5. identificar o que ainda está distante da direção visual.

Objetivo

Não partir da referência ignorando o que já existe no projeto.

---

8. Passo 2 — Relembrar a direção visual

Revisar, conforme disponíveis:

- referência principal;
- características aprovadas de referências secundárias;
- `design.md`;
- Contexto de Desenvolvimento;
- identidade;
- público;
- linguagem;
- briefing.

Pergunta de controle

> O que desta referência pertence à linguagem do projeto e o que é específico da marca original?

---

9. Passo 3 — Separar referência de cópia literal

A meta AS-IS é:

> preservar a essência visual e comportamental útil, adaptando-a ao cliente.

Pode ser aproveitado:

- composição;

- ritmo;
- proporções;
- Hero;
- hierarquia;
- direção tipográfica;
- cards;
- interações;
- transições;
- comportamento de seções.

Não copiar automaticamente:

- conteúdo;
- marca;
- identidade;
- elementos que conflitam com briefing;
- solução inadequada a mobile/usabilidade.

PARTE II — COMPARAÇÃO E REFINAMENTO POR SEÇÃO

10. Passo 4 — Escolher uma seção por vez

Trabalhar a Home por partes.

Exemplos:

- Header;
- Hero;
- seção institucional;
- especialidades/serviços;
- prova social;
- CTA;
- FAQ;
- Footer.

Regra prática

Evitar pedir para a IA “refazer a Home inteira com extrema fidelidade” em um único prompt.

---

11. Passo 5 — Fornecer evidência concreta

Para a seção escolhida, fornecer quando disponível:

- screenshot;
- link;
- referência;
- estado visual desejado;
- breakpoint relevante.

Screenshots reduzem ambiguidade sobre:

- enquadramento;
- proporção;
- estado;
- detalhe;
- comportamento.

---

12. Passo 6 — Pedir estudo antes de editar quando necessário

Se a seção for simples, a implementação pode seguir diretamente.

Se houver:

- comportamento complexo;
- animação;
- interação;
- efeito;
- composição pouco óbvia;

pedir primeiro que a ferramenta:

1. observe;
2. descreva;
3. identifique o comportamento;
4. avalie a técnica provável;
5. só então implemente.

Objetivo

Evitar que a primeira hipótese vire código imediatamente.

---

13. Passo 7 — Delimitar o escopo da alteração

O prompt deve dizer claramente:

- qual seção pode mudar;
- qual seção não pode mudar;
- se desktop está preservado;
- se mobile está preservado;
- se backend/Admin não deve ser alterado;
- quais componentes validados devem permanecer.

Exemplo de lógica AS-IS

```
trabalhar Hero mobile
→ preservar Hero desktop já validado
→ não alterar demais seções
```

---

14. Passo 8 — Buscar fidelidade estrutural e visual

Durante o refinamento, comparar principalmente:

- estrutura;
- proporções;

- tipografia;
- imagens;
- espaçamentos;
- alinhamentos;
- largura;
- ritmo;
- comportamento;
- efeitos.

A primeira geração raramente é suficiente.

O trabalho é iterativo.

---

15. Passo 9 — Adaptar para o cliente

Depois de compreender a referência, adaptar:

- conteúdo;
- identidade;
- imagens;
- CTA;
- linguagem;
- necessidades de conversão;
- contexto do cliente.

Se houver conflito

Priorizar julgamento humano considerando:

- briefing;
- identidade;
- usabilidade;
- mobile;
- funcionalidade.

Não existe algoritmo formal universal de decisão.

PARTE III — HERO E ÁREAS DE MAIOR PESO VISUAL

16. Passo 10 — Validar o Hero e priorizar o esforço por valor

O Hero é uma área crítica de primeira leitura e precisa estar coerente com a linguagem da Home. Isso não significa que deva receber sempre a maior parcela de esforço: outra seção de maior valor estratégico para o cliente, maior complexidade ou maior impacto pode demandar mais refinamento.

Revisar:

- composição;
- proporção;
- tipografia;
- imagem;
- CTA;
- movimento;
- enquadramento;
- leitura inicial.

Se o Hero usa efeito complexo

Aplicar o subfluxo de efeitos da Parte IV.

---

17. Passo 11 — Manter consistência entre seções

Depois de uma seção ser refinada, verificar se a Home mantém coerência em:

- tipografia;
- espaçamento;
- paleta;
- raio/bordas;
- densidade;
- ritmo;
- linguagem de animação.

> **Importante**
>
> Consistência não significa repetição mecânica.

PARTE IV — SUBFLUXO DE EFEITOS E ANIMAÇÕES

18. Quando aplicar

Aplicar apenas quando o efeito:

- faz parte da referência;
- combina com o cliente;
- agrega à experiência;
- é tecnicamente viável.

---

19. Passo E1 — Identificar o efeito

Registrar:

- seção;
- comportamento;
- gatilho;
- direção;
- relação com scroll;
- interação;
- estado inicial/final.

---

20. Passo E2 — Fornecer screenshot e link

Sempre que possível:

- mostrar a seção;
- fornecer a referência real;
- indicar exatamente o comportamento a estudar.

---

21. Passo E3 — Investigar técnica/biblioteca

A ferramenta pode investigar se a solução depende de:

- CSS;
- Framer Motion;
- GSAP;
- ScrollTrigger;
- Three;
- R3F;

- Lottie;
- Swiper;
- IntersectionObserver;
- outra técnica existente no projeto.

> **Regra**
>
> A biblioteca é meio, não objetivo.

Não adicionar tecnologia apenas por preferência da IA.

---

22. Passo E4 — Reproduzir isoladamente

Quando o risco for relevante:

- construir/testar a lógica do efeito na seção delimitada;
- evitar alterar outras áreas simultaneamente.

---

23. Passo E5 — Adaptar ao projeto

A referência não precisa ser reproduzida literalmente.

Adaptar:

- cores;
- assets;
- intensidade;
- velocidade;
- dimensões;
- conteúdo;
- comportamento.

---

24. Passo E6 — Validar desktop/notebook/mobile

Um efeito pode funcionar no desktop e falhar em:

- notebook;
- mobile;
- telas estreitas.

Verificar:

- overflow;
- scroll horizontal;
- legibilidade;
- estabilidade;
- peso percebido.

---

25. Quando adaptar, simplificar, substituir ou remover o efeito

Quando um efeito apresenta problema, seguir a preferência prática:

1. adaptar;
2. simplificar ou substituir;
3. remover apenas em caso extremo, quando a solução continuar instável, prejudicial ou incompatível.

Aplicar essa ordem quando o efeito:

- quebra;
- fica instável;
- pesa demais;
- não combina com o cliente;
- prejudica mobile;
- prejudica usabilidade;
- conflita com briefing.

Lições reais

Problemas já observados incluem:

- 3D instável;
- borda de Hero quebrando em notebook;
- scroll horizontal inadequado.

PARTE V — ASSETS E CONTEÚDO NA HOME

26. Passo 12 — Aplicar assets disponíveis

Usar, conforme projeto:

- fotos do cliente;
- assets de identidade;
- materiais enviados;
- imagens selecionadas;
- imagens contextuais geradas quando necessário.

Controle humano

Validar:

- enquadramento;
- crop;
- foco;
- adequação visual;
- comportamento por breakpoint.

---

27. Passo 13 — Tratar conteúdo com fidelidade factual

A IA pode criar/adaptar texto.

Não inventar fatos sensíveis.

Especialmente:

- formação;
- títulos;
- especialidades;
- procedimentos;
- resultados;
- afirmações médicas.

Pendência pode ser registrada ou receber placeholder simples quando suportado pelo contexto, sem transformá-lo em fato validado.

PARTE VI — RESPONSIVIDADE DURANTE A CONSTRUÇÃO

28. Passo 14 — Não esperar o QA final para olhar mobile

Responsividade entra durante o refinamento visual.

A cada seção relevante, observar:

- desktop;
- notebook;
- mobile.

---

29. Passo 15 — Preservar breakpoints já aprovados

Se a correção é específica:

- mobile → preservar desktop;
- desktop → preservar mobile.

A instrução deve ser explícita.

---

30. Solução diferente por dispositivo

É aceitável que mobile não reproduza literalmente desktop.

Pode haver:

- simplificação;
- reposicionamento;
- remoção de efeito;
- mudança de composição.

O objetivo é preservar linguagem e usabilidade.

PARTE VII — PERFORMANCE E ACESSIBILIDADE NO NÍVEL DA CONSTRUÇÃO

31. Passo 16 — Observar peso percebido

Durante refinamento, observar se:

- scroll perde fluidez;
- efeito trava;
- 3D pesa;
- imagem demora;
- composição fica excessivamente pesada.

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe score numérico universal de performance para essa etapa.

---

32. Passo 17 — Corrigir problemas óbvios de acessibilidade

No AS-IS há atenção consciente a:

- contraste;
- tamanho de texto;
- alt;
- nome acessível;
- legibilidade mobile.

Limite

Não existe auditoria WCAG formal obrigatória dentro deste SOP.

PARTE VIII — CICLO ITERATIVO

33. Passo 18 — Comparar novamente com a referência

Depois de cada rodada:

1. abrir a referência;
2. abrir o resultado;
3. comparar;
4. identificar diferença relevante;
5. decidir se deve ser corrigida ou adaptada.

---

34. Passo 19 — Refinar apenas o delta relevante

Não recomeçar a seção inteira quando o problema está localizado.

Exemplos:

- proporção;
- espaço;
- tipografia;
- imagem;
- timing;
- posição.

---

35. Passo 20 — Validar manualmente

O Dev:

- navega;
- observa;
- compara;
- testa interação;
- verifica breakpoints.

A validação humana é parte obrigatória do AS-IS.

PARTE IX — CRITÉRIO DE PRONTO

36. Condição de pronto da Home

A Home pode avançar quando:

> **já estabelece de forma suficientemente clara a linguagem visual do projeto e está adequada para servir de referência às páginas internas.**

Isso inclui, por julgamento humano:

- todas as seções previstas para a Home no Sitemap/contexto estão presentes;
- pendências e placeholders remanescentes estão identificados como tais;
- direção visual coerente;
- estrutura principal adequada;
- tipografia coerente;
- imagens adequadas;
- espaçamentos/alinhamentos satisfatórios;
- comportamento aceitável;
- CTAs coerentes;
- desktop/notebook/mobile suficientemente corretos;
- efeitos aceitáveis quando aplicáveis.

---

37. Relação com o Marco 5 do Processo Sites WSS

No Processo Sites WSS, o Marco 5 busca uma Home que represente:

- direção visual;
- posicionamento;
- estrutura;
- conversão esperada.

Este SOP descreve **como o Dev chega visualmente a essa Home**.

Não substitui o critério formal do Marco 5.

---

38. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — score visual

Não existe:

- score de fidelidade;
- percentual de equivalência;
- checklist numérico universal;

- threshold automático de aprovação.

A decisão é humana.

PARTE X — STOP CONDITION

39. Condição de parada

O SOP está concluído quando:

> **a Home já estabelece a linguagem visual do projeto, está suficientemente refinada e responsiva para servir de base às páginas internas e não possui problema visual evidente que impeça esse avanço.**

---

40. O que NÃO é necessário para encerrar

Não exigir:

- QA completo do site;
- todas as páginas internas;
- Admin concluído;
- auditoria SEO;
- auditoria de segurança;
- publicação;
- aprovação final do cliente.

PARTE XI — CONTINGÊNCIAS

41. A referência não funciona bem para o cliente

Não forçar.

Adaptar ou descartar elementos incompatíveis.

> **DECISÃO HUMANA NECESSÁRIA**

---

42. A IA perde a direção visual

1. interromper;
2. reforçar `design.md` /contexto;
3. mostrar screenshot;
4. limitar seção;
5. pedir novo estudo;
6. retomar em partes menores.

---

43. Um efeito quebra mobile/notebook

- não preservar por fidelidade cega;
- simplificar;
- substituir;
- remover;
- criar solução específica por breakpoint.

---

44. A primeira versão do Lovable está muito simples

Isso é comum.

A Home é refinada posteriormente com o fluxo híbrido.

Não significa reiniciar o projeto.

PARTE XII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

45. Não inventar

O AS-IS não possui:

- checklist universal de análise de referência;
- obrigação de usar DevTools em toda referência;
- regra fixa de quantas rodadas de refinamento;
- score visual;
- biblioteca obrigatória de animação;
- threshold numérico de performance;
- auditoria WCAG obrigatória;
- regra universal de componentização por seção;
- obrigação de reproduzir desktop e mobile de forma idêntica.

PARTE XIII — NORMA TÉCNICA VIGENTE

46. Preservar normas aplicáveis

Quando existir norma técnica vigente para páginas públicas:

- ela deve ser respeitada durante a construção;
- a busca por fidelidade visual não pode anulá-la;
- conteúdo público não deve ser transformado em arquitetura incompatível com a norma por preferência visual.

Para SEO/AEO/GEO, a autoridade vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Na camada normativa, Core Web Vitals funcionam como metas recomendadas, e `alt` contextual/semântica aplicam-se quando exigidos pela classificação técnica correspondente.

Isso não transforma CWV, acessibilidade ou SEO em novo gate visual deste SOP.

Este SOP não redefine a norma.

PARTE XIV — ANTI-PATTERNS

47. Enviar apenas um link e pedir “extrema fidelidade”

Evitar.

Melhor:

```
seção
+ screenshot
+ link
+ estudo
+ escopo limitado
+ comparação
```

48. Aceitar a primeira geração como final

A primeira geração normalmente precisa de refinamento.

49. Alterar várias seções quando o problema é local

Aumenta risco de regressão e perda de contexto.

50. Levar automaticamente efeito desktop para mobile

Pode prejudicar:

- desempenho;
- legibilidade;
- navegação;
- estabilidade.

51. Preservar efeito quebrado em nome da referência

Sofisticação não é objetivo absoluto.

PARTE XV — OUTPUT

52. Saída esperada

Ao final:

Home

- linguagem visual consolidada;
- estrutura refinada;
- Hero coerente;
- tipografia/imagens/espaçamentos ajustados;
- CTAs integrados;
- efeitos adequados quando aplicáveis;
- responsividade suficiente para avançar.

Contexto visual

A Home passa a funcionar como referência para:

- páginas internas;
- padrões recorrentes;
- decisões de composição;
- refinamento posterior.

PARTE XVI — PRÓXIMO PASSO

53. Encaminhamento

Depois do SOP-003:

SOP-004 — Construir Páginas Internas Mantendo Consistência Visual

O SOP-004 deve usar a Home como principal referência interna do próprio projeto, junto com:

- Sitemap;
- conteúdo;
- contexto;
- referências específicas da página quando existirem.

---

54. Referências

Manual proprietário

- `WSS_Manual_03_Construcao_Visual_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`;
- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`;
- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

SOP anterior

- `WSS_SOP_002_Iniciar_Tecnicamente_Projeto_WSS_v1.1_FINAL.md`

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
- `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-004
Construir Páginas Internas Mantendo Consistência Visual

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-004

Construir Páginas Internas Mantendo Consistência Visual

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 3 — Construção Visual
MANUAIS RELACIONADOS	Manual 2 — Entrada e Contexto do Projeto; Manual 5 — Desenvolvimento com Codex, Lovable e GitHub; Manual 6 — Testes, QA, Debugging, Segurança e SEO
ESCOPO	AS-IS — desenvolvimento das páginas internas após estabilização da linguagem visual da Home

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 7

2. Gatilho — quando usar 7

3. Quando NÃO usar 7

4. Papéis 8

5. Pré-condições 9

6. Inputs 9

PARTE I — DEFINIR QUAL PÁGINA DESENVOLVER 10

7. Passo 1 — Consultar o Sitemap 11

8. Passo 2 — Priorizar a próxima página 11

9. Passo 3 — Permitir paralelismo quando útil 12

PARTE II — PREPARAR A PÁGINA 13

10. Passo 4 — Identificar a finalidade da página 14

11. Passo 5 — Confirmar as seções previstas 14

12. Passo 6 — Identificar conteúdo disponível e pendente 14

PARTE III — USAR A HOME COMO REFERÊNCIA INTERNA 16

13. Passo 7 — Ler a Home como sistema visual 17

14. Passo 8 — Identificar o que deve permanecer 17

15. Passo 9 — Identificar o que pode variar 18

PARTE IV — PRIMEIRA GERAÇÃO 19

16. Passo 10 — Gerar a primeira versão 20

17. Passo 11 — Limitar o escopo 20

18. Passo 12 — Avaliar a primeira versão 20

PARTE V — REUSO E COMPONENTIZAÇÃO 21

19. Princípio AS-IS 22

20. Passo 13 — Verificar se já existe solução reutilizável 22

21. Passo 14 — Reutilizar quando fizer sentido 22

22. Passo 15 — Não generalizar solução específica sem necessidade 22

23. Passo 16 — Preservar o Design System híbrido do projeto 23

PARTE VI — REFINAMENTO VISUAL **24**

24. Passo 17 — Refinar seção por seção 25

25. Passo 18 — Comparar com a Home 25

26. Passo 19 — Evitar cópia literal da Home 25

27. Passo 20 — Ajustar o peso visual à importância da página 26

PARTE VII — REFERÊNCIA ESPECÍFICA E EFEITOS **27**

28. Quando uma página possui referência própria 28

29. Subfluxo de efeito 28

30. Regra de contenção 28

PARTE VIII — ASSETS **29**

31. Passo 21 — Aplicar imagens disponíveis 30

32. Passo 22 — Validar enquadramento por breakpoint 30

33. Assets ausentes 30

34. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — pipeline de assets 30

PARTE IX — CONTEÚDO **32**

35. Passo 23 — Aplicar ou adaptar conteúdo 33

36. Passo 24 — Preservar fatos sensíveis 33

37. Conteúdo provisório 33

PARTE X — CTAS, NAVEGAÇÃO E CONEXÕES ENTRE PÁGINAS **34**

38. Passo 25 — Aplicar CTAs relevantes 35

39. Passo 26 — Conferir navegação principal da página 35

40. Caso real de atenção 35

PARTE XI — RESPONSIVIDADE **36**

41. Passo 27 — Revisar desktop 37

42. Passo 28 — Revisar notebook 37

43. Passo 29 — Revisar mobile 37

44. Passo 30 — Adaptar quando necessário 37

PARTE XII — PERFORMANCE E ACESSIBILIDADE NO NÍVEL DE CONSTRUÇÃO **39**

45. Passo 31 — Observar custo percebido 40

46. Passo 32 — Corrigir problemas óbvios de acessibilidade 40

PARTE XIII — VALIDAÇÃO DA PÁGINA 41

47. Três verificações AS-IS principais 42

48. Verificação adicional de consistência 42

49. Não confundir página pronta com homologação 42

PARTE XIV — CICLO ENTRE PÁGINAS 43

50. Passo 33 — Registrar pendências relevantes 44

51. Passo 34 — Preservar componentes/padrões úteis 44

52. Passo 35 — Seguir para a próxima página 44

PARTE XV — COBERTURA DO SITEMAP 45

53. Quando o conjunto de páginas internas pode avançar 46

54. Relação com o Marco 6 do Processo Sites WSS 46

PARTE XVI — STOP CONDITION 47

55. Condição de parada por página 48

56. Condição de parada do SOP 48

57. O que NÃO é necessário para encerrar 48

PARTE XVII — CONTINGÊNCIAS 49

58. A página ficou igual demais à Home 50

59. A página ficou desconectada da Home 50

60. A página ficou simples demais 50

61. Falta conteúdo 50

62. Falta imagem 50

63. A IA altera outra página/Home 51

64. Um componente compartilhado quebra outra página 51

PARTE XVIII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 52

65. Não inventar 53

PARTE XIX — NORMA TÉCNICA VIGENTE 54

66. Preservar normas aplicáveis 55

PARTE XX — ANTI-PATTERNS 56

67. Criar cada página como se fosse um projeto novo 57

68. Repetir exatamente a Home em todas as páginas 57

69. Criar componente genérico para qualquer bloco específico 57

70. Ignorar componente reutilizável já existente 57

71. Trabalhar página inteira com prompt excessivamente amplo 57

72. Aceitar página visualmente bonita com ações quebradas 57

73. Inventar conteúdo para preencher seção 58

PARTE XXI — OUTPUT **59**

74. Saída esperada por página 60

75. Saída esperada do conjunto 60

PARTE XXII — PRÓXIMO PASSO **61**

76. Encaminhamento 62

77. Referências 62

1. Finalidade

Transformar a linguagem visual já consolidada na Home em páginas internas:

- coerentes;
- navegáveis;
- visualmente compatíveis com o projeto;
- adequadas ao conteúdo específico de cada página;
- reaproveitando componentes quando fizer sentido;
- sem transformar consistência em repetição mecânica.

Fluxo principal:

```
Home estabilizada
→ Sitemap
→ escolher página
→ definir conteúdo/estrutura
→ gerar primeira versão
→ reutilizar o que faz sentido
→ variar o que precisa variar
→ refinar
→ aplicar assets/conteúdo
→ adaptar efeitos quando aplicável
→ revisar responsividade
→ validar página
→ seguir para próxima
```

---

2. Gatilho — quando usar

Usar quando:

- o SOP-003 foi concluído;
- a Home já estabelece suficientemente a linguagem visual do projeto;
- o Sitemap está disponível;
- existe pelo menos uma página interna a desenvolver.

---

3. Quando NÃO usar

Não usar:

- antes de a Home estabelecer a direção visual;
- para redesenhar a Home;
- para definir Sitemap;
- para criar funcionalidade nova fora do escopo;

- para configurar Admin;
- para executar QA completo;
- para publicar;
- para portar Base para projeto legado.

---

4. Papéis

4.1 Desenvolvedor

Responsável por:

- priorizar páginas;
- interpretar a Home como referência;
- decidir nível de sofisticação;
- julgar reuso × variação;
- validar qualidade visual;
- preservar o escopo;
- decidir quando a página está pronta para avançar.

4.2 Codex

Pode ser usado para:

- construção direta;
- fidelidade visual;
- investigação;
- implementação pesada;
- reprodução de comportamento;
- reuso/componentização.

4.3 Claude

Pode ser usado para:

- refinamento visual;
- tipografia;
- acabamento;
- efeitos;
- mobile;
- adaptação de interface.

4.4 Lovable

Pode ser usado conforme a tarefa, principalmente quando:

- a alteração faz mais sentido em sua integração;
- há questão funcional/backend;

- pequenos ajustes são convenientes ali.

A divisão entre ferramentas permanece contextual.

---

5. Pré-condições

Antes de iniciar:

1. SOP-003 concluído;
2. Home suficientemente estabilizada;
3. Sitemap validado disponível;
4. contexto do projeto acessível;
5. conteúdo disponível ou pendências identificadas;
6. assets disponíveis ou necessidade de substituição/provisório conhecida;
7. ambiente atualizado antes da rodada local;
8. ferramenta utilizada compreende o projeto.

---

6. Inputs

Obrigatórios

- Home vigente;
- Sitemap;
- Contexto de Desenvolvimento;
- Briefing Final quando necessário;
- conteúdo disponível da página.

Quando existirem

- `design.md`;
- referência visual específica da página;
- screenshots;
- assets;
- fotos;
- material de cliente;
- componentes já existentes;
- layouts compartilhados;
- pendências de conteúdo.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE I — DEFINIR QUAL PÁGINA DESENVOLVER

7. Passo 1 — Consultar o Sitemap

Usar o Sitemap para responder:

- quais páginas precisam existir;
- qual é a função de cada página;
- quais páginas são prioritárias;
- quais páginas podem ser mais diretas;
- quais páginas merecem maior refinamento.

O Sitemap orienta **cobertura**.

Ele não determina sozinho o nível visual de cada página.

---

8. Passo 2 — Priorizar a próxima página

No AS-IS, não existe ordem universal rígida.

A sequência predominante observada após a Home é:

```
Home
→ Contato
→ Blog
```

Contato e Blog costumam vir cedo por serem, em geral, páginas mais simples. A ordem pode mudar por prioridade estratégica, conteúdo, referência, complexidade ou dependência técnica.

Outras práticas observadas:

- Sobre tende a receber atenção alta;
- página geral de Especialidades pode receber refinamento relevante;
- páginas individuais podem ser mais diretas;
- página estrategicamente importante pode receber mais atenção.

Critérios práticos de priorização

- complexidade;
- existência de referência;
- ideia visual disponível;
- importância estratégica;
- velocidade de execução;
- dependência de conteúdo.

DECISÃO HUMANA NECESSÁRIA

Não existe algoritmo de priorização.

---

9. Passo 3 — Permitir paralelismo quando útil

Se surgir uma solução clara para uma seção de outra página:

- ela pode ser trabalhada antes de a página atual terminar.

Limite

Paralelismo não deve provocar:

- perda de contexto;
- abandono de páginas;
- inconsistência;
- escopo fora de controle.

PARTE II — PREPARAR A PÁGINA

10. Passo 4 — Identificar a finalidade da página

Antes de gerar a primeira versão, entender:

- o que o usuário precisa encontrar;
- qual conteúdo precisa estar presente;
- qual ação principal é esperada;
- qual relação ela tem com a Home e outras páginas.

---

11. Passo 5 — Confirmar as seções previstas

Usar Sitemap/contexto/conteúdo para listar as seções esperadas.

> **Regra**
>
> Não inventar novas seções estratégicas apenas porque “ficariam bonitas”.

Se uma nova seção implicar:

- escopo novo;
- informação nova;
- funcionalidade nova;
- decisão do cliente;

> **DECISÃO HUMANA NECESSÁRIA.**

---

12. Passo 6 — Identificar conteúdo disponível e pendente

Classificar:

- conteúdo disponível;
- conteúdo a adaptar;
- conteúdo provisório;
- conteúdo sensível;
- conteúdo ainda pendente.

> **Importante**
>
> Uma pendência não bloqueia automaticamente a construção.

Avaliar impacto concreto.

PARTE III — USAR A HOME COMO REFERÊNCIA INTERNA

13. Passo 7 — Ler a Home como sistema visual

Observar:

- tipografia;
- cores;
- espaçamentos;
- largura;
- botões;
- cards;
- bordas;
- raios;
- linguagem de imagem;
- ritmo;
- motion;
- Header;
- Footer;
- CTA final;
- padrões recorrentes.

---

14. Passo 8 — Identificar o que deve permanecer

Elementos frequentemente reaproveitados:

- Header;
- Footer;
- CTA final;
- FAQ quando aplicável;
- botões;
- identidade;
- tipografia;
- cores;
- padrões de espaçamento;
- linguagem geral.

Objetivo

Fazer a página parecer parte do mesmo produto.

---

15. Passo 9 — Identificar o que pode variar

Podem variar:

- Hero;
- estrutura de seções;
- depoimentos;
- composições;
- efeitos;
- densidade;
- grau de sofisticação.

> **Princípio**

> consistência visual não significa páginas idênticas.

PARTE IV — PRIMEIRA GERAÇÃO

16. Passo 10 — Gerar a primeira versão

A instrução pode seguir a lógica:

```
com base na Home
+ Sitemap
+ contexto
+ conteúdo da página
→ criar página X
```

Se houver referência específica

Adicionar:

- screenshot;
- link;
- seção relevante;
- comportamento desejado.

---

17. Passo 11 — Limitar o escopo

Especificar:

- página que pode mudar;
- componentes compartilhados que podem ser usados;
- áreas que não devem ser alteradas;
- Home já validada que deve ser preservada;
- Admin/backend fora de escopo quando não relacionados.

---

18. Passo 12 — Avaliar a primeira versão

Verificar:

- se as seções previstas aparecem;
- se a página parece pertencer ao mesmo site;
- se está simples demais;
- se há excesso de repetição da Home;
- se o conteúdo encaixa bem;
- se a hierarquia faz sentido.

A primeira versão normalmente recebe refinamento.

PARTE V — REUSO E COMPONENTIZAÇÃO

19. Princípio AS-IS

A componentização é pragmática.

Não existe regra formal como:

> “se apareceu duas vezes, obrigatoriamente criar componente”.

---

20. Passo 13 — Verificar se já existe solução reutilizável

Antes de criar algo novo, procurar:

- componente compartilhado;
- padrão existente;
- layout parametrizado;
- seção adaptável;
- token/variável existente.

---

21. Passo 14 — Reutilizar quando fizer sentido

Reuso pode preservar:

- consistência;
- velocidade;
- manutenção;
- comportamento.

Exemplos reais de projetos WSS demonstram:

- FAQ compartilhado;
- cards parametrizados;
- layouts orientados por dados.

---

22. Passo 15 — Não generalizar solução específica sem necessidade

Se a seção:

- é única;
- tem comportamento próprio;

- possui composição muito específica;

ela pode permanecer local.

Objetivo

Evitar overengineering.

---

23. Passo 16 — Preservar o Design System híbrido do projeto

Quando existirem:

- `design.md`;
- tokens;
- variáveis;
- componentes;
- utilitários;

usar como referências complementares.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe um Design System único formal universal para todos os sites WSS.

PARTE VI — REFINAMENTO VISUAL

24. Passo 17 — Refinar seção por seção

Trabalhar:

- Hero;
- seções informativas;
- cards;
- imagens;
- CTAs;
- FAQs;
- provas sociais;
- blocos especiais.

---

25. Passo 18 — Comparar com a Home

A cada rodada, verificar:

- tipografia;
- proporções;
- espaçamentos;
- linguagem;
- ritmo;
- contraste;
- acabamento.

Pergunta de controle:

> esta página pertence visualmente ao mesmo projeto?

---

26. Passo 19 — Evitar cópia literal da Home

Não repetir mecanicamente:

- mesmos blocos;
- mesma ordem;
- mesmo Hero;
- mesmos efeitos;

se o conteúdo da página pede outra solução.

---

27. Passo 20 — Ajustar o peso visual à importância da página

Página prioritária pode receber:

- composição mais elaborada;
- refinamento extra;
- referência própria;
- efeito específico.

Página individual simples pode ser mais direta.

> **DECISÃO HUMANA NECESSÁRIA**
>
> Não existe classificação universal de sofisticação.

PARTE VII — REFERÊNCIA ESPECÍFICA E EFEITOS

28. Quando uma página possui referência própria

Se existir referência específica:

1. identificar a seção;
2. fornecer screenshot;
3. fornecer link;
4. pedir estudo;
5. limitar escopo;
6. adaptar à linguagem já consolidada da Home.

---

29. Subfluxo de efeito

Quando houver efeito relevante, usar a mesma lógica do SOP-003:

```
identificar
→ estudar
→ reproduzir isoladamente
→ adaptar
→ validar
→ simplificar/substituir se necessário
→ remover apenas se a solução continuar inviável
```

---

30. Regra de contenção

Não usar efeito apenas para “deixar todas as páginas sofisticadas”.

O efeito precisa:

- fazer sentido;
- combinar com a página;
- preservar usabilidade;
- funcionar em mobile.

PARTE VIII — ASSETS

31. Passo 21 — Aplicar imagens disponíveis

Fontes podem incluir:

- material do cliente;
- Drive;
- assets do projeto;
- imagens selecionadas;
- imagens contextuais geradas por IA quando necessário.

---

32. Passo 22 — Validar enquadramento por breakpoint

Verificar:

- foco;
- crop;
- proporção;
- posição;
- legibilidade sobre imagem.

Mobile pode precisar de posicionamento próprio.

---

33. Assets ausentes

Se faltar material:

- registrar pendência;
- utilizar solução provisória quando suportada;
- não transformar provisório em definitivo sem validação.

---

34. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — pipeline de assets

Não existe etapa formal universal separada de:

- compressão;
- formato;
- naming.

Não inventar pipeline neste SOP.

PARTE IX — CONTEÚDO

35. Passo 23 — Aplicar ou adaptar conteúdo

A IA pode ajudar a:

- criar;
- organizar;
- adaptar;
- resumir;
- reescrever.

---

36. Passo 24 — Preservar fatos sensíveis

Não inventar:

- formação;
- títulos;
- especialidades;
- procedimentos;
- resultados;
- afirmações médicas;
- fatos sensíveis equivalentes.

---

37. Conteúdo provisório

Quando aplicável:

- marcar como provisório;
- manter pendência visível;
- substituir quando o conteúdo final chegar.

PARTE X — CTAs, NAVEGAÇÃO E CONEXÕES ENTRE PÁGINAS

38. Passo 25 — Aplicar CTAs relevantes

Verificar se a página possui:

- ação principal;
- CTA coerente;
- destino adequado.

---

39. Passo 26 — Conferir navegação principal da página

No nível de construção, testar:

- botões principais;
- links principais;
- CTA;
- retorno/navegação esperada.

> **Limite**
>
> QA completo pertence ao SOP-008.

---

40. Caso real de atenção

Um projeto visualmente sofisticado apresentou CTA apontando para rota inexistente.

Lição:

> qualidade visual não substitui conferência de ação principal.

PARTE XI — RESPONSIVIDADE

41. Passo 27 — Revisar desktop

Observar:

- estrutura;
- largura;
- espaçamento;
- imagens;
- efeitos;
- CTA.

---

42. Passo 28 — Revisar notebook

Notebook pode revelar problemas que não aparecem no monitor maior:

- proporções;
- bordas;
- overflow;
- efeitos.

---

43. Passo 29 — Revisar mobile

Verificar:

- leitura;
- stacking;
- espaçamento;
- imagens;
- CTAs;
- efeitos;
- scroll horizontal.

---

44. Passo 30 — Adaptar quando necessário

Mobile pode ter:

- composição diferente;
- efeito simplificado;
- imagem reposicionada;

- elementos removidos.

Não exigir equivalência literal ao desktop.

PARTE XII — PERFORMANCE E ACESSIBILIDADE NO NÍVEL DE CONSTRUÇÃO

45. Passo 31 — Observar custo percebido

Se a página fica pesada:

- identificar efeito/asset/componente causador;
- simplificar ou otimizar quando necessário.

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Sem score universal obrigatório nesta etapa.

---

46. Passo 32 — Corrigir problemas óbvios de acessibilidade

Observar, quando perceptível:

- contraste;
- tamanho de texto;
- nomes acessíveis;
- alt;
- legibilidade.

Limite

Não existe auditoria WCAG formal obrigatória neste SOP.

PARTE XIII — VALIDAÇÃO DA PÁGINA

47. Três verificações AS-IS principais

Antes de considerar a página pronta na fase de construção:

1. **todas as seções previstas no Sitemap estão presentes;**
2. **a qualidade visual está satisfatória e não parece simples demais;**
3. **botões e ações principais funcionam.**

Essas são as três verificações práticas consolidadas.

---

48. Verificação adicional de consistência

Confirmar também:

- identidade preservada;
- linguagem compatível com a Home;
- conteúdo coerente;
- ausência de desvio evidente.

Essa verificação não cria um novo checklist formal; apenas explicita o objetivo do SOP.

---

49. Não confundir página pronta com homologação

“Pronta” nesta etapa não significa:

- QA completo;
- regressão concluída;
- SEO concluído;
- segurança concluída;
- publicação pronta;
- cliente aprovou.

Essas etapas pertencem aos SOPs/Manuais posteriores.

PARTE XIV — CICLO ENTRE PÁGINAS

50. Passo 33 — Registrar pendências relevantes

Antes de seguir:

- conteúdo faltante;
- imagem faltante;
- decisão humana;
- problema não bloqueante.

---

51. Passo 34 — Preservar componentes/padrões úteis

Se a página criou solução reutilizável que faz sentido para outras:

- mantê-la acessível;
- reaproveitar posteriormente quando apropriado.

---

52. Passo 35 — Seguir para a próxima página

Voltar ao Sitemap e repetir:

```
priorizar
→ construir
→ refinar
→ validar
→ registrar pendências
→ próxima
```

PARTE XV — COBERTURA DO SITEMAP

53. Quando o conjunto de páginas internas pode avançar

O conjunto pode avançar para QA principal quando:

- páginas previstas no recorte de desenvolvimento foram implementadas ou possuem pendência/justificativa conhecida;
- navegação principal está utilizável;
- conteúdo-base está aplicado;
- páginas estão visualmente coerentes;
- problemas bloqueantes evidentes foram tratados.

---

54. Relação com o Marco 6 do Processo Sites WSS

No Processo Sites WSS, o Marco 6 trata:

- páginas internas;
- conteúdo-base;
- implementação do Sitemap;
- navegação.

Este SOP documenta **como o Dev constrói essas páginas**.

Ele não substitui:

- checklist formal do Marco;
- status;
- evidências;
- decisão formal de conclusão.

PARTE XVI — STOP CONDITION

55. Condição de parada por página

Uma página está concluída neste SOP quando:

> **possui as seções previstas, apresenta qualidade visual satisfatória e coerente com a Home, e suas ações principais funcionam no nível necessário para seguir à próxima etapa.**

---

56. Condição de parada do SOP

O SOP-004 está concluído quando:

> **as páginas internas necessárias ao estágio atual foram desenvolvidas com coerência visual suficiente para entrar no QA principal, com pendências remanescentes explicitamente registradas.**

---

57. O que NÃO é necessário para encerrar

Não exigir:

- Admin concluído;
- QA completo;
- auditoria SEO;
- auditoria de segurança;
- publicação;
- aprovação final do cliente;
- ausência absoluta de pendências não bloqueantes.

PARTE XVII — CONTINGÊNCIAS

58. A página ficou igual demais à Home

- revisar finalidade;
- variar composição;
- preservar identidade sem repetir estrutura.

---

59. A página ficou desconectada da Home

- reler `design.md`;
- comparar tokens/padrões;
- reutilizar elementos recorrentes;
- revisar tipografia/espaçamento/CTA.

---

60. A página ficou simples demais

- revisar hierarquia;
- composição;
- imagens;
- ritmo;
- referência específica;
- importância estratégica.

Não adicionar complexidade sem função.

---

61. Falta conteúdo

Avaliar:

- bloqueia a estrutura?
- pode usar provisório?
- precisa de decisão humana?

Registrar a pendência.

---

62. Falta imagem

Usar solução provisória adequada ou registrar pendência.

Não inventar material factual.

---

63. A IA altera outra página/Home

1. interromper;
2. restaurar/preservar área validada;
3. limitar escopo;
4. reforçar página específica;
5. retomar.

---

64. Um componente compartilhado quebra outra página

Tratar como regressão.

Não aceitar a mudança apenas porque melhorou a página atual.

Ver Manual 6 / SOP-008.

PARTE XVIII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

65. Não inventar

O AS-IS não possui:

- sequência rígida universal de páginas;
- score de qualidade visual;
- regra formal universal de componentização;
- obrigação de usar template compartilhado;
- Design System universal formal;
- número fixo de rodadas;
- nível obrigatório de sofisticação por tipo de página;
- matriz universal de breakpoints;
- score universal de performance;
- auditoria WCAG obrigatória;
- pipeline formal universal de assets.

PARTE XIX — NORMA TÉCNICA VIGENTE

66. Preservar normas aplicáveis

Quando o projeto estiver sujeito a norma técnica vigente:

- preservar arquitetura pública exigida;
- não sacrificar requisito técnico por conveniência visual;
- manter regras aplicáveis de URL, renderização, indexação e conteúdo;
- tratar a norma como camada própria.

Para SEO/AEO/GEO, a autoridade vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Conteúdo público indexável continua sujeito ao resultado arquitetural exigido por essa norma. Core Web Vitals, `alt` contextual e semântica permanecem requisitos/recomendações da camada normativa quando aplicáveis, sem transformar este SOP visual em procedimento de SEO.

Este SOP não redefine a norma.

PARTE XX — ANTI-PATTERNS

67. Criar cada página como se fosse um projeto novo

Isso quebra consistência.

A Home é a referência interna principal.

68. Repetir exatamente a Home em todas as páginas

Consistência não significa duplicação.

69. Criar componente genérico para qualquer bloco específico

Pode gerar overengineering.

70. Ignorar componente reutilizável já existente

Pode aumentar duplicação e divergência.

71. Trabalhar página inteira com prompt excessivamente amplo

Preferir:

- página delimitada;
- seção;
- objetivo;
- referências concretas.

72. Aceitar página visualmente bonita com ações quebradas

Botões e ações principais fazem parte do critério prático de pronto.

73. Inventar conteúdo para preencher seção

Especialmente inadequado em fatos sensíveis.

PARTE XXI — OUTPUT

74. Saída esperada por página

Cada página deve sair com:

- estrutura prevista implementada;
- linguagem coerente com a Home;
- conteúdo-base aplicado;
- assets aplicados/provisórios identificados;
- CTAs principais;
- ações principais funcionais;
- responsividade suficiente;
- pendências registradas.

---

75. Saída esperada do conjunto

Ao final do SOP:

- Sitemap materializado no estágio necessário;
- páginas internas visualmente coerentes;
- Home preservada como âncora;
- reuso aplicado de forma pragmática;
- variações próprias das páginas preservadas;
- conjunto pronto para QA principal.

PARTE XXII — PRÓXIMO PASSO

76. Encaminhamento

Depois do SOP-004, a execução pode seguir em paralelo conforme o projeto para:

- SOP-005 — Configurar Admin e Módulos Aplicáveis;
- SOP-008 — Executar QA Visual, Funcional e Responsivo.

A sequência real pode ter sobreposição, pois o Admin pode continuar em paralelo ao institucional.

---

77. Referências

Manual proprietário

- `WSS_Manual_03_Construcao_Visual_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`;
- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`;
- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

SOP anterior

- `WSS_SOP_003_Construir_Refinar_Home_Referencias_v1.1_FINAL.md`

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
- `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`.

Evidência técnica complementar

- `WSS_VAL_001_DanielP_Componentizacao_DesignSystem_Visual_v0.1.md`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-005
Configurar Admin e Módulos Aplicáveis

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-005

Configurar Admin e Módulos Aplicáveis

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 4 — Projeto-Base, Admin e Funcionalidades
MANUAIS RELACIONADOS	Manual 2 — Entrada e Contexto do Projeto; Manual 3 — Construção Visual; Manual 6 — Testes, QA, Debugging, Segurança e SEO; Manual 7 — Publicação, Entrega e Pós-Publicação
ESCOPO	AS-IS — adaptação e configuração da camada operacional herdada da Base Admin para um projeto cliente

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 7

2. Gatilho — quando usar 7

3. Quando NÃO usar 7

4. Papéis 8

5. Pré-condições 9

6. Inputs 9

7. Ferramentas 10

PARTE I — DETERMINAR APLICABILIDADE 11

8. Passo 1 — Inventariar os módulos presentes 12

9. Passo 2 — Classificar cada capacidade 12

10. Passo 3 — Tratar módulo não aplicável 13

11. Passo 4 — Escalar dúvida de escopo 13

PARTE II — PREPARAR O ADMIN 14

12. Passo 5 — Confirmar acesso ao Admin 15

13. Passo 6 — Remover conteúdo demonstrativo da experiência final 15

14. Passo 7 — Aplicar configuração específica do cliente 15

PARTE III — BLOG 17

15. Quando o Blog se aplica 18

16. Passo 8 — Conferir o Blog público 18

17. Passo 9 — Conferir o Blog no Admin 18

18. Passo 10 — Confirmar reflexo público básico 19

PARTE IV — FORMULÁRIOS, CTAS E LEADS 20

19. Finalidade 21

20. Passo 11 — Inventariar pontos de captação aplicáveis 21

21. Passo 12 — Configurar campos e destinos 21

22. Passo 13 — Executar teste ponta a ponta de Lead 21

23. Dados de teste 22

24. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — integração externa 22

PARTE V — CHAT / CTA / WHATSAPP 23

25. Passo 14 — Configurar Chat/CTA 24

26. Passo 15 — Configurar WhatsApp quando aplicável 24

27. Passo 16 — Testar o Chat 24

28. Chat sem WhatsApp 25

PARTE VI — FAQ, DEPOIMENTOS E OUTROS MÓDULOS 26

29. Passo 17 — Configurar capacidades complementares aplicáveis 27

30. Não inventar comportamento ausente 27

PARTE VII — CHATBOT — GATE PARA SOP-006 28

31. Passo 18 — Determinar se Chatbot se aplica 29

32. O que o SOP-005 deve confirmar antes do SOP-006 29

33. Não duplicar o SOP-006 29

PARTE VIII — AUTENTICAÇÃO E PERFIS 30

34. Passo 19 — Preparar contas internas de teste 31

35. Passo 20 — Conferir perfil Equipe/cliente 31

36. Passo 21 — Conferir perfil Webstar 31

37. Base de Conhecimento como exemplo de limite 31

38. Fronteira de segurança 31

PARTE IX — INTEGRAÇÕES E CONFIGURAÇÕES COMPLEMENTARES 33

39. Passo 22 — Conferir integrações aplicáveis 34

40. Analytics 34

PARTE X — CONFIGURAÇÃO MÓDULO POR MÓDULO 35

41. Ciclo operacional recomendado pelo AS-IS 36

42. Limitar alteração ao módulo 36

43. Quando o módulo exige backend/banco 36

PARTE XI — VALIDAÇÃO FUNCIONAL DA CONFIGURAÇÃO 37

44. O que validar dentro deste SOP 38

45. O que NÃO validar completamente aqui 38

PARTE XII — PENDÊNCIAS E BLOQUEIOS 40

46. Passo 23 — Registrar pendências por módulo 41

47. Distinguir bloqueante de não bloqueante 41

48. Mudança de escopo 41

PARTE XIII — STOP CONDITION 42

49. Condição de parada 43

50. Condições mínimas 43

51. O que NÃO é necessário para encerrar 43

PARTE XIV — CONTINGÊNCIAS 44

52. Módulo herdado está quebrado 45

53. Módulo existe na Base, mas não deveria aparecer 45

54. Conteúdo demo aparece no cliente 45

55. Lead não chega ao Admin 45

56. Chat funciona visualmente, mas conversa não aparece no Admin 45

57. Perfil acessa módulo indevido 45

58. Chatbot contratado não está configurado 46

PARTE XV — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 47

59. Não inventar 48

PARTE XVI — NORMA TÉCNICA VIGENTE 49

60. Preservar normas aplicáveis 50

PARTE XVII — PROCESSO SITES WSS / MARCO 7 51

61. Relação com o Marco 7 — Configuração Operacional 52

62. Não absorver o Crosswalk do Flow como AS-IS do Dev 52

PARTE XVIII — ANTI-PATTERNS 53

63. Entregar a Base como veio 54

64. Recriar módulo que já funciona 54

65. Manter módulo não contratado apenas porque está no código 54

66. Apagar código de módulo não aplicável sem necessidade 54

67. Validar formulário apenas pela mensagem “enviado” 54

68. Validar Chat apenas porque o widget abre 54

69. Criar permissão extra apenas para facilitar operação 54

70. Confundir configuração com feature nova 55

PARTE XIX — OUTPUT 56

71. Saída esperada 57

PARTE XX — PRÓXIMO PASSO **58**

72. Encaminhamento 59

73. Referências 59

1. Finalidade

Transformar a camada operacional herdada da Base Admin em uma configuração específica do cliente, aproveitando as capacidades já existentes e adaptando apenas o que é necessário.

Fluxo principal:

```
Base Admin herdada
→ identificar módulos aplicáveis
→ ocultar/desativar os não aplicáveis
→ configurar Admin
→ adaptar Blog
→ configurar captação / Leads
→ configurar Chat / CTA / WhatsApp
→ preparar perfis e acessos
→ encaminhar Chatbot ao SOP-006 quando aplicável
→ validar funcionalmente módulo por módulo
→ registrar pendências
→ camada operacional pronta para QA geral
```

A finalidade deste SOP é **configurar e fazer uma validação funcional mínima das capacidades**, não executar a homologação completa do site.

---

2. Gatilho — quando usar

Usar quando:

- o projeto já foi criado/remixado a partir da Base Admin;
- o contexto do cliente está compreendido;
- módulos/capacidades aplicáveis são conhecidos em nível suficiente;
- a camada institucional já possui estrutura suficiente para conectar os fluxos operacionais;
- o Dev inicia ou retoma a finalização do Admin.

No fluxo atual, este SOP pode acontecer **em paralelo à revisão do institucional pelo cliente**.

---

3. Quando NÃO usar

Não usar:

- para criar um novo projeto do zero;
- para substituir integralmente o Admin de um projeto legado;
- para portar evolução da Base para site antigo;
- para construir Home/páginas internas;
- para executar auditoria de segurança;

- para executar QA geral;
- para publicar;
- para entregar o acesso definitivo do cliente.

Procedimentos relacionados

- projeto novo/setup técnico → SOP-002;
- Chatbot/RAG → SOP-006;
- Base → legado → SOP-007;
- QA geral → SOP-008;
- entrega/limpeza final → SOP-012.

---

4. Papéis

4.1 Desenvolvedor

Responsável por:

- identificar a configuração necessária;
- adaptar os módulos;
- preservar capacidades da Base;
- configurar dados, destinos e conteúdos;
- testar funcionalidade básica;
- registrar pendências;
- escalar mudança de escopo.

4.2 WSS / Direção / Operações

Responsável por esclarecer, quando necessário:

- módulo contratado;
- capacidade aplicável;
- destino de CTA;
- regra comercial;
- acesso futuro;
- informação faltante;
- pedido que represente nova funcionalidade.

4.3 Cliente

Pode fornecer:

- dados;
- canais;
- conteúdo;
- acessos;

- informações operacionais.

O cliente não é responsável por definir tecnicamente a arquitetura da Base.

---

5. Pré-condições

Antes de iniciar, deve existir:

1. projeto cliente derivado da Base ou equivalente atual;
2. Admin acessível ao Dev;
3. escopo/capacidades conhecidos;
4. contexto e materiais do cliente disponíveis;
5. páginas/formulários/CTAs relevantes implementados ou suficientemente definidos;
6. destinos/canais conhecidos quando necessários;
7. ambiente atualizado para a rodada de configuração.

Se uma capacidade estiver indefinida

Não presumir que está contratada.

> **DECISÃO HUMANA NECESSÁRIA.**

---

6. Inputs

Conforme o projeto:

- capacidades/módulos aplicáveis;
- Contexto de Desenvolvimento;
- Briefing/Sitemap quando necessários;
- identidade visual;
- conteúdo;
- dados do cliente;
- WhatsApp/destinos;
- perguntas;
- campos de captação;
- perfis esperados;
- conteúdo de Blog/FAQ;
- configurações existentes;
- pendências operacionais.

---

7. Ferramentas

Podem participar:

- Lovable;
- ambiente local;
- Codex;
- Claude, quando houver adaptação visual;
- Supabase/Lovable Cloud quando a capacidade depender dessa camada;
- navegador/Admin.

Fronteira

A mecânica Git pertence ao Manual 5.

Este SOP descreve **o que configurar e verificar**, não os comandos de implementação.

PARTE I — DETERMINAR APLICABILIDADE

8. Passo 1 — Inventariar os módulos presentes

Abrir o projeto/Admin e reconhecer as capacidades herdadas da Base.

Exemplos recorrentes:

- Admin;
- Blog;
- Leads;
- formulários;
- Chat/CTA;
- WhatsApp;
- Chatbot;
- FAQ;
- depoimentos;
- avaliações;
- configurações;
- Search/Performance;
- autenticação/perfis;
- outras capacidades presentes na geração corrente.

> **Importante**
>
> A lista concreta depende da versão do projeto.

Não presumir que toda geração histórica contém o mesmo conjunto.

---

9. Passo 2 — Classificar cada capacidade

Para cada módulo, decidir:

- aplicável;
- não aplicável;
- pendente de decisão.

Fonte principal

Usar o escopo/capacidades do projeto.

Não fazer

- manter módulo demo apenas porque veio da Base;
- ativar feature nova por conveniência;

- tratar presença no código como prova de contratação.

---

10. Passo 3 — Tratar módulo não aplicável

No AS-IS, módulos não aplicáveis tendem a ser:

- ocultados;
- desativados;
- retirados da experiência administrativa do projeto.

> **Limite**
>
> Isso **não exige apagar fisicamente o código**.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe procedimento universal de dead-code elimination para módulo não utilizado.

---

11. Passo 4 — Escalar dúvida de escopo

Se a configuração exige funcionalidade não prevista ou transforma significativamente a capacidade herdada:

> **DECISÃO HUMANA NECESSÁRIA — validar escopo antes de implementar.**

PARTE II — PREPARAR O ADMIN

12. Passo 5 — Confirmar acesso ao Admin

Antes de configurar módulos:

- abrir Admin;
- confirmar que o ambiente correto está sendo usado;
- confirmar navegação básica;
- identificar módulos disponíveis.

Se o Admin não abre

Não seguir configurando por inferência.

Tratar como problema técnico e encaminhar ao fluxo de debugging.

---

13. Passo 6 — Remover conteúdo demonstrativo da experiência final

A Base é ponto de partida, não produto final.

Revisar módulos que possam conter:

- conteúdo demo;
- dados genéricos;
- textos genéricos;
- identidade anterior;
- registros de exemplo.

> **Regra**
>
> Conteúdo demonstrativo não pode ser confundido com conteúdo real do cliente.

---

14. Passo 7 — Aplicar configuração específica do cliente

Conforme a capacidade, adaptar:

- dados;
- conteúdo;
- identidade;
- destinos;
- permissões;
- configurações;

- campos;
- textos.

Não reconstruir integralmente um módulo que já resolve a necessidade.

PARTE III — BLOG

15. Quando o Blog se aplica

Se Blog fizer parte do projeto:

- manter o módulo disponível no Admin;
- adaptar a experiência pública à identidade do site;
- revisar configurações específicas;
- revisar conteúdo necessário.

Se não se aplica:

- não expor o módulo como capacidade operacional do cliente.

---

16. Passo 8 — Conferir o Blog público

No AS-IS, a adaptação pública tende a se concentrar em:

- navbar;
- tipografia;
- tamanho de texto;
- cores;
- espaçamentos;
- coerência com o institucional.

A construção visual detalhada pertence ao Manual 3.

---

17. Passo 9 — Conferir o Blog no Admin

O editor tende a preservar a estrutura herdada da Base.

Revisar, conforme aplicável:

- configuração;
- conteúdo;
- categorias;
- imagem;
- campos editoriais;
- recursos relacionados a SEO;
- integrações com Base de Conhecimento.

Não presumir

Nem toda geração do projeto possui exatamente as mesmas funções editoriais.

18. Passo 10 — Confirmar reflexo público básico

Realizar ao menos uma verificação funcional suficiente para confirmar que:

- conteúdo válido criado/publicado pelo fluxo previsto aparece onde deveria;
- a experiência não continua usando conteúdo demo indevido.

> **Limite**
>
> A auditoria completa do Blog pertence ao SOP-008/SOP-010 conforme o tipo de validação.

PARTE IV — FORMULÁRIOS, CTAs E LEADS

19. Finalidade

A camada de captação deve transformar uma ação pública relevante em registro verificável no Admin.

---

20. Passo 11 — Inventariar pontos de captação aplicáveis

Identificar:

- formulários;
- CTAs;
- agendamento;
- Chat/CTA;
- outros pontos que deveriam gerar Lead.

---

21. Passo 12 — Configurar campos e destinos

Conforme o projeto, revisar:

- campos coletados;
- interesse;
- especialidade/opção;
- origem;
- destino;
- textos;
- mensagem de sucesso/erro.

> **Limite**
>
> Não adicionar coleta nova sem necessidade/escopo.

---

22. Passo 13 — Executar teste ponta a ponta de Lead

Para cada fluxo essencial:

1. acionar formulário/CTA/agendamento;
2. preencher com dados fictícios;
3. enviar;
4. conferir sucesso ou erro;

5. abrir Admin;
6. localizar o Lead;
7. conferir nome;
8. conferir origem/trajeto;
9. conferir especialidade/opção quando aplicável.

Critério funcional

O frontend sozinho não basta.

O registro precisa chegar ao Admin de forma coerente.

---

23. Dados de teste

Usar dados claramente fictícios.

Os registros podem permanecer temporariamente durante QA.

> **Importante**
>
> A limpeza definitiva dos dados de teste ocorre antes da entrega do Admin no SOP-012.

---

24. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — integração externa

Não existe integração externa adicional universal obrigatória para Leads no AS-IS consolidado.

Não transformar:

- CRM externo;
- automação de marketing;
- ferramenta de terceiros;

em requisito universal sem escopo específico.

PARTE V — CHAT / CTA / WHATSAPP

25. Passo 14 — Configurar Chat/CTA

Revisar, conforme o projeto:

- posição/experiência;
- identidade visual;
- textos;
- perguntas;
- campos;
- dados capturados;
- destino.

---

26. Passo 15 — Configurar WhatsApp quando aplicável

O WhatsApp pode funcionar como:

- destino de CTA;
- encaminhamento;
- canal disponível no Chat;
- fallback do Chatbot.

Validar:

- número;
- mensagem;
- destino.

> **Importante**
>
> WhatsApp não é universalmente obrigatório.

---

27. Passo 16 — Testar o Chat

Executar uma conversa de teste.

Depois:

- abrir o Admin;
- confirmar que a conversa aparece;
- conferir dados relacionados;
- conferir vínculo com Lead quando o fluxo o prevê.

28. Chat sem WhatsApp

Se o Chat funciona sem WhatsApp e o projeto não exige esse canal:

- não tratar a ausência do número como falha automática.

A configuração depende do projeto.

PARTE VI — FAQ, DEPOIMENTOS E OUTROS MÓDULOS

29. Passo 17 — Configurar capacidades complementares aplicáveis

Módulos como:

- FAQ;
- depoimentos;
- avaliações;
- Search;
- Performance;
- configurações complementares;

devem ser tratados pelo mesmo padrão:

```
abrir
→ substituir/adaptar conteúdo do cliente
→ conferir configuração
→ executar funcionalidade essencial
→ registrar pendência
→ seguir
```

---

30. Não inventar comportamento ausente

Se as fontes não sustentam exatamente como uma capacidade deve ser configurada:

- seguir a implementação real do projeto;
- não criar regra universal no SOP.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE VII — CHATBOT — GATE PARA SOP-006

31. Passo 18 — Determinar se Chatbot se aplica

Se Chatbot não foi contratado/não se aplica:

- ocultar/desativar conforme a prática do projeto;
- não executar SOP-006.

Se Chatbot se aplica:

> executar **SOP-006 — Configurar e Validar Chatbot e Base de Conhecimento**.

---

32. O que o SOP-005 deve confirmar antes do SOP-006

No mínimo:

- módulo identificado como aplicável;
- área administrativa correspondente disponível;
- conteúdo demo não será tratado como conhecimento final;
- dados/contexto necessários do cliente foram identificados;
- canais de encaminhamento conhecidos quando aplicáveis.

---

33. Não duplicar o SOP-006

Ficam fora deste SOP:

- ingestão detalhada da Base de Conhecimento;
- `knowledge_base`;
- `knowledge_chunks`;
- RAG;
- system prompt;
- limites clínicos em profundidade;
- benchmark;
- perguntas de referência;
- handoff detalhado.

PARTE VIII — AUTENTICAÇÃO E PERFIS

34. Passo 19 — Preparar contas internas de teste

No AS-IS, o Dev cria duas contas internas:

- perfil Equipe/cliente;
- perfil Webstar.

As credenciais internas são mantidas no local operacional apropriado da WSS, não incorporadas indiscriminadamente à documentação pública do projeto.

---

35. Passo 20 — Conferir perfil Equipe/cliente

Verificar funcionalmente se:

- consegue acessar o que deveria;
- não recebe superfícies Webstar-only evidentes;
- consegue operar os módulos previstos.

---

36. Passo 21 — Conferir perfil Webstar

Verificar se:

- possui as capacidades administrativas esperadas;
- consegue acessar módulos exclusivos quando aplicável.

---

37. Base de Conhecimento como exemplo de limite

Na Base inspecionada, a gestão da Base de Conhecimento/IA é capacidade Webstar e não é presumida como disponível ao perfil Equipe.

> **Regra**
>
> Não ampliar permissão apenas para facilitar teste.

---

38. Fronteira de segurança

Este SOP verifica o comportamento funcional dos perfis.

Não substitui:

- RLS;
- teste negativo de autorização;
- escalonamento de privilégio;
- secrets;
- auditoria server-side.

Esses itens pertencem ao Manual 6 e às validações de segurança aplicáveis.

PARTE IX — INTEGRAÇÕES E CONFIGURAÇÕES COMPLEMENTARES

39. Passo 22 — Conferir integrações aplicáveis

Tratar somente integrações realmente previstas.

Exemplos observados, conforme o projeto:

- Supabase/Lovable Cloud;
- WhatsApp;
- Search Console;
- Lovable AI Gateway;
- serviços relacionados à Base.

---

40. Analytics

Quando aplicável, conferir a configuração prevista para o projeto.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> O AS-IS não sustenta um conjunto universal amplo de integrações/Analytics idêntico para todo site.

PARTE X — CONFIGURAÇÃO MÓDULO POR MÓDULO

41. Ciclo operacional recomendado pelo AS-IS

Para cada módulo aplicável:

```
abrir
→ entender estado herdado
→ ajustar dados/configuração do cliente
→ remover conteúdo demo inadequado
→ testar função essencial
→ corrigir problema localizado
→ registrar pendência
→ seguir para o próximo
```

---

42. Limitar alteração ao módulo

Ao trabalhar em uma capacidade:

- preservar institucional já validado;
- preservar outros módulos quando não relacionados;
- evitar alteração ampla desnecessária.

---

43. Quando o módulo exige backend/banco

Usar a ferramenta adequada, frequentemente Lovable quando sua integração facilita a tarefa.

Controle humano

Pontos sensíveis como:

- credenciais;
- chaves;
- tabelas;
- permissões;

exigem atenção humana.

PARTE XI — VALIDAÇÃO FUNCIONAL DA CONFIGURAÇÃO

44. O que validar dentro deste SOP

No nível mínimo necessário antes do QA geral:

Admin

- abre;
- navegação básica funciona;
- módulos aplicáveis aparecem;
- módulos não aplicáveis não são apresentados indevidamente.

Blog, quando aplicável

- módulo existe;
- configuração básica está correta;
- conteúdo pode seguir o fluxo previsto.

Leads/formulários

- ação pública gera resultado;
- Lead aparece no Admin;
- origem/dados essenciais estão coerentes.

Chat

- conversa inicia;
- aparece no Admin.

WhatsApp

- destino/número estão corretos quando aplicável.

Perfis

- Equipe e Webstar operam as superfícies esperadas no teste funcional.

Chatbot

- SOP-006 concluído ou encaminhado como pendência bloqueante quando aplicável.

---

45. O que NÃO validar completamente aqui

Não executar como parte deste SOP:

- QA página por página completo;
- matriz extensa de erros;
- auditoria formal de segurança;

- SEO/AEO/GEO;
- smoke de produção;
- QA pós-publicação.

PARTE XII — PENDÊNCIAS E BLOQUEIOS

46. Passo 23 — Registrar pendências por módulo

Exemplos:

- dado do cliente ausente;
- WhatsApp ainda não confirmado;
- conteúdo de Blog pendente;
- acesso indisponível;
- módulo contratado ainda não configurado;
- problema funcional aguardando correção.

---

47. Distinguir bloqueante de não bloqueante

Perguntar:

> esta pendência impede que a capacidade funcione/possa ser validada ou apenas impede acabamento/entrega final?

Se bloqueante

Não considerar o módulo pronto.

Se não bloqueante

Registrar e seguir, quando seguro.

---

48. Mudança de escopo

Se o pedido passa de configuração para funcionalidade nova:

- parar;
- registrar;
- retornar à gestão.

Não mascarar feature nova como “ajuste do Admin”.

PARTE XIII — STOP CONDITION

49. Condição de parada

O SOP está concluído quando:

> **todos os módulos aplicáveis ao projeto foram configurados no nível necessário para funcionar, os principais fluxos operacionais foram verificados, módulos não aplicáveis não são apresentados indevidamente e as pendências remanescentes estão explicitamente registradas, deixando a camada operacional pronta para o QA geral.**

---

50. Condições mínimas

Antes de encerrar, confirmar:

- [ ] Admin acessível;
- [ ] aplicabilidade dos módulos definida;
- [ ] conteúdo demo inadequado tratado;
- [ ] módulos não aplicáveis ocultados/desativados quando necessário;
- [ ] Blog configurado quando aplicável;
- [ ] captação/Leads testados ponta a ponta;
- [ ] Chat testado quando aplicável;
- [ ] WhatsApp conferido quando aplicável;
- [ ] perfis internos preparados e verificados funcionalmente;
- [ ] Chatbot encaminhado/concluído pelo SOP-006 quando aplicável;
- [ ] pendências registradas.

---

51. O que NÃO é necessário para encerrar

Não exigir:

- entrega do acesso do cliente;
- limpeza final de todos os dados de teste;
- QA geral concluído;
- auditoria de segurança completa;
- SEO concluído;
- domínio definitivo;
- publicação.

PARTE XIV — CONTINGÊNCIAS

52. Módulo herdado está quebrado

Não reescrever imediatamente.

Primeiro:

1. reproduzir;
2. identificar ambiente/camada;
3. verificar configuração;
4. seguir SOP-009 quando o problema exigir diagnóstico.

---

53. Módulo existe na Base, mas não deveria aparecer

- confirmar aplicabilidade;
- ocultar/desativar;
- não apagar código sem necessidade.

---

54. Conteúdo demo aparece no cliente

Corrigir antes de considerar configuração pronta.

---

55. Lead não chega ao Admin

O fluxo não está funcional.

Encaminhar ao debugging e não considerar a capacidade pronta.

---

56. Chat funciona visualmente, mas conversa não aparece no Admin

Não considerar concluído.

O fluxo precisa ser verificado ponta a ponta.

---

57. Perfil acessa módulo indevido

Tratar como falha relevante.

Não corrigir apenas escondendo UI se houver suspeita de problema de autorização.

Encaminhar à validação técnica de segurança quando necessário.

---

58. Chatbot contratado não está configurado

SOP-005 permanece incompleto para essa capacidade até o SOP-006 atingir sua stop condition ou haver decisão formal que altere a aplicabilidade.

PARTE XV — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

59. Não inventar

O AS-IS não possui:

- reconstrução completa do Admin por cliente;
- exclusão física obrigatória de todo módulo não aplicável;
- conjunto universal de integrações externas;
- CRM externo obrigatório para Leads;
- WhatsApp obrigatório em todo projeto;
- manual formal obrigatório do Admin nesta etapa;
- suíte universal completa de testes dentro da configuração;
- auditoria de segurança completa embutida neste SOP;
- procedimento universal de migrations/rollback para qualquer alteração de módulo.

PARTE XVI — NORMA TÉCNICA VIGENTE

60. Preservar normas aplicáveis

Capacidades da Base podem estar sujeitas a normas específicas de:

- arquitetura pública;
- Blog;
- autenticação;
- segurança;
- privacidade;
- SEO/indexação.

Quando houver norma vigente:

- respeitá-la;
- não confundi-la com prática histórica;
- não remover requisito normativo por conveniência da Base.

Para SEO/AEO/GEO, a autoridade vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Isso inclui, quando aplicável, requisitos normativos do Blog público e de áreas privadas.

O padrão estabelece Google Search Console configurado na publicação como requisito WSS. Isso não transforma Search Console em tarefa universal deste SOP nem em responsabilidade pessoal obrigatória do Dev.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE XVII — PROCESSO SITES WSS / MARCO 7

61. Relação com o Marco 7 — Configuração Operacional

O Processo Sites WSS define o Marco 7 como a preparação da camada operacional para revisão geral.

No Roadmap, ele prevê, conforme aplicabilidade:

- Admin configurado;
- Blog no Admin;
- Chat/CTA;
- captação validada;
- Leads de teste;
- Chatbot quando contratado;
- pendências operacionais registradas.

Este SOP descreve **como o Dev executa a configuração técnica/operacional**.

---

62. Não absorver o Crosswalk do Flow como AS-IS do Dev

O Processo Sites WSS possui sua própria estrutura de:

- capacidades;
- módulos;
- validações;
- evidências;
- checklist;
- histórico.

Esses registros formais não devem ser confundidos com o comportamento técnico descrito neste SOP.

Regra

```
Processo/Marco 7
= o que precisa estar controlado/concluído

SOP-005
= como o Dev configura a camada operacional
```

PARTE XVIII — ANTI-PATTERNS

63. Entregar a Base como veio

A Base contém estrutura reutilizável, não configuração final do cliente.

64. Recriar módulo que já funciona

Primeiro adaptar a capacidade existente.

65. Manter módulo não contratado apenas porque está no código

Aplicabilidade vem do projeto, não da presença técnica.

66. Apagar código de módulo não aplicável sem necessidade

Ocultar/desativar é prática corrente; remoção física não é regra universal.

67. Validar formulário apenas pela mensagem “enviado”

É necessário conferir o Lead no Admin.

68. Validar Chat apenas porque o widget abre

É necessário conferir a conversa no Admin.

69. Criar permissão extra apenas para facilitar operação

Perfis devem preservar a separação prevista.

70. Confundir configuração com feature nova

Nova funcionalidade exige decisão de escopo.

PARTE XIX — OUTPUT

71. Saída esperada

Ao final:

Admin

- acessível;
- adaptado ao cliente;
- sem conteúdo demo indevido relevante;
- módulos aplicáveis configurados;
- módulos não aplicáveis tratados.

Operação

- Blog configurado quando aplicável;
- captação chegando ao Admin;
- Leads de teste identificáveis;
- Chat funcional quando aplicável;
- WhatsApp correto quando aplicável;
- perfis internos preparados;
- Chatbot tratado pelo SOP-006 quando aplicável.

Controle

- pendências conhecidas;
- bloqueios explícitos;
- camada operacional pronta para QA geral.

PARTE XX — PRÓXIMO PASSO

72. Encaminhamento

Se Chatbot se aplica:

SOP-006 — Configurar e Validar Chatbot e Base de Conhecimento

Depois, conforme o fluxo:

SOP-008 — Executar QA Visual, Funcional e Responsivo

O Admin pode ser configurado em paralelo à revisão do institucional; portanto, a sequência operacional não precisa ser completamente linear.

---

73. Referências

Manual proprietário

- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`;
- `WSS_Manual_03_Construcao_Visual_v1.1_FINAL.md`;
- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`;
- `WSS_Manual_07_Publicacao_Entrega_Pos_Publicacao_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
- `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`.

Processo — fronteira

- `Roadmap_Sites_WSS_Modelo_Padrao_v0.2_FINAL.md`;
- `WSS_Crosswalk_Operacional_Curado_Marco_7_Configuracao_Operacional_v0.1_FINAL.docx`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-006
Configurar e Validar Chatbot e Base de Conhecimento

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-006

Configurar e Validar Chatbot e Base de Conhecimento

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 4 — Projeto-Base, Admin e Funcionalidades
MANUAIS RELACIONADOS	Manual 6 — Testes, QA, Debugging, Segurança e SEO; Manual 7 — Publicação, Entrega e Pós-Publicação
ESCOPO	AS-IS — configuração e validação do Chatbot herdado da Base Admin para um projeto cliente

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 7
2. Gatilho — quando usar 7
3. Quando NÃO usar 7
4. Papéis 8
5. Pré-condições 9
6. Inputs 9

PARTE I — COMPREENDER A CAPACIDADE EXISTENTE 10

7. Arquitetura AS-IS 11
8. O site não é crawleado em tempo real a cada pergunta 11
9. Componentes lógicos da Base de Conhecimento 11
10. Origens observadas de conhecimento 12

PARTE II — LIMPEZA E PREPARAÇÃO DO CONHECIMENTO 13

11. Passo 1 — Abrir a Base de Conhecimento 14
12. Passo 2 — Identificar conteúdo demonstrativo 14
13. Passo 3 — Substituir/revisar conteúdo demonstrativo 14
14. Passo 4 — Materializar conteúdo real das páginas 14
15. Passo 5 — Separar conteúdo público de informação não confirmada 15

PARTE III — FAQ E BLOG 16

16. Passo 6 — Revisar FAQ 17
17. Passo 7 — Revisar Blog 17
18. Conteúdo despublicado/inativo 17

PARTE IV — CONFIGURAÇÃO DO ASSISTENTE 18

19. Passo 8 — Configurar identidade e textos 19
20. Passo 9 — Configurar WhatsApp quando aplicável 19
21. Passo 10 — Confirmar dependência de IA 19

PARTE V — LIMITES DE RESPOSTA 21

22. Passo 11 — Revisar os limites configurados 22
23. Não universalizar a lista clínica 22
24. WhatsApp dentro do prompt 22

PARTE VI — CAPTURA DE DADOS E LEADS **23**
25. Passo 12 — Iniciar conversa como usuário 24
26. Passo 13 — Confirmar criação/vínculo do Lead 24
27. Enriquecimento 24

PARTE VII — TESTES DE RESPOSTA **25**
28. Passo 14 — Testar pergunta coberta 26
29. Passo 15 — Testar pergunta fora da cobertura 26
30. Passo 16 — Testar fato sensível 26
31. Passo 17 — Testar WhatsApp quando configurado 26

PARTE VIII — HANDOFF HUMANO **27**
32. Passo 18 — Solicitar atendimento humano 28
33. Passo 19 — Assumir conversa no Admin 28
34. Passo 20 — Encerrar atendimento quando aplicável 28

PARTE IX — FALLBACKS TÉCNICOS **29**
35. Comportamentos observados 30
36. Este SOP não exige simular toda falha externa 30

PARTE X — BASE DE CONHECIMENTO COMO CAPACIDADE WEBSTAR **31**
37. Passo 21 — Confirmar segregação funcional 32
38. Não corrigir autorização apenas por UI 32

PARTE XI — BENCHMARK **33**
39. Quando usar 34
40. Passo 22 — Revisar perguntas de referência 34
41. Passo 23 — Executar/revisar benchmark 34
42. Passo 24 — Interpretar resultado 35
43. Feedback negativo 35

PARTE XII — VALIDAÇÃO MANUAL ADMIN + USUÁRIO **36**
44. Passo 25 — Repetir o fluxo como usuário 37
45. Passo 26 — Repetir o fluxo como Admin 37
46. Por que as duas visões são necessárias 37

PARTE XIII — PENDÊNCIAS E BLOQUEIOS **38**

47. Passo 27 — Registrar pendências 39

48. Classificar bloqueio 39

PARTE XIV — STOP CONDITION **40**

49. Condição de parada 41

50. Checklist mínimo 41

51. O que NÃO é necessário para encerrar 41

PARTE XV — CONTINGÊNCIAS **43**

52. O Chatbot responde algo que não está na Base 44

53. Pergunta conhecida não é respondida 44

54. Conversa existe, mas Lead não 44

55. Lead existe, mas conversa não aparece no Admin 44

56. Handoff não silencia a IA 45

57. WhatsApp aparece sem estar configurado 45

58. Perfil Equipe consegue gerenciar conhecimento indevidamente 45

59. Benchmark ruim 45

PARTE XVI — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO **46**

60. Não inventar 47

PARTE XVII — NORMA TÉCNICA VIGENTE **48**

61. Segurança e privacidade 49

62. Conteúdo público e arquitetura 49

PARTE XVIII — ANTI-PATTERNS **50**

63. Entregar conhecimento demo 51

64. “O site é a Base de Conhecimento” entendido como crawler automático 51

65. Alimentar a IA com fato não confirmado 51

66. Validar só pela resposta visual 51

67. Validar só pelo benchmark 51

68. Dar acesso de conhecimento ao cliente por conveniência 51

69. Fixar modelo atual como padrão permanente 52

PARTE XIX — OUTPUT **53**

70. Saída esperada 54

PARTE XX — PRÓXIMO PASSO **55**

71. Encaminhamento 56

72. Referências 56

1. Finalidade

Configurar a capacidade de Chatbot da Base Admin para um cliente real, substituindo conteúdo demonstrativo, preparando a Base de Conhecimento, validando integração com Leads/Admin e verificando o comportamento do assistente como usuário e como administrador.

Fluxo principal:

```
Chatbot aplicável
→ substituir conteúdo demo
→ materializar conteúdo real do cliente
→ revisar FAQ/Blog
→ sincronizar conhecimento aplicável
→ configurar textos / identidade / WhatsApp
→ confirmar dependência de IA
→ testar identificação do visitante
→ confirmar Lead
→ testar perguntas cobertas
→ testar pergunta sem resposta confirmada
→ testar handoff humano
→ conferir conversa no Admin
→ revisar benchmark quando aplicável
→ registrar pendências
→ Chatbot pronto para QA geral
```

---

2. Gatilho — quando usar

Usar quando:

- o SOP-005 identificou o Chatbot como capacidade aplicável ao projeto;
- a infraestrutura herdada da Base está presente;
- existe conteúdo real suficiente para iniciar a configuração;
- o Dev precisa adaptar e validar a capacidade antes da revisão geral.

---

3. Quando NÃO usar

Não usar:

- se Chatbot não faz parte do escopo;
- para criar um Chatbot completamente novo fora da Base existente;
- para redesenhar toda a arquitetura de IA;
- para executar auditoria geral de segurança;
- para substituir o SOP-008 de QA;
- para entregar o acesso final do cliente;
- para definir política universal de IA para todos os setores.

Procedimentos relacionados

- Admin e módulos → SOP-005;
- QA geral → SOP-008;
- debugging → SOP-009;
- entrega/limpeza final → SOP-012.

---

4. Papéis

4.1 Desenvolvedor / Webstar

Responsável por:

- configurar conteúdo;
- revisar a Base de Conhecimento;
- adaptar identidade/textos;
- configurar canais;
- testar como usuário;
- testar como administrador;
- verificar Lead/conversa;
- revisar benchmark quando aplicável;
- registrar pendências.

4.2 Perfil Webstar

Na Base inspecionada, a gestão da Base de Conhecimento/IA pertence ao perfil/capability Webstar.

4.3 Perfil Equipe/cliente

Pode operar capacidades permitidas de atendimento e gestão, mas **não deve ser presumido como administrador da Base de Conhecimento**.

4.4 Cliente / responsável pelo conteúdo

Pode precisar validar:

- fatos;
- serviços;
- informações institucionais;
- conteúdo médico/factual;
- canais;
- dados de atendimento.

---

5. Pré-condições

Antes de iniciar:

1. Chatbot identificado como aplicável;
2. módulo disponível no projeto;
3. acesso Webstar ao Admin;
4. conteúdo do cliente disponível em nível suficiente;
5. FAQ e Blog conhecidos quando aplicáveis;
6. dados das páginas públicas acessíveis;
7. WhatsApp/destino conhecido quando aplicável;
8. infraestrutura da Base funcional;
9. dependência/secret da IA disponível quando necessária.

Se o conteúdo real ainda não existe

Não usar conhecimento demo como conteúdo final.

Registrar a pendência e limitar o avanço.

---

6. Inputs

Conforme o projeto:

- conteúdo das páginas públicas;
- serviços/especialidades;
- FAQ;
- Blog;
- informações institucionais;
- conteúdos manuais;
- perguntas frequentes;
- textos do Chatbot;
- identidade;
- WhatsApp;
- dados de atendimento;
- restrições de resposta;
- perguntas de referência do benchmark, quando existirem.

PARTE I — COMPREENDER A CAPACIDADE EXISTENTE

7. Arquitetura AS-IS

A capacidade observada segue a lógica:

```
conteúdo do site
+ FAQ
+ Blog
+ entradas manuais
+ feedback convertido em conhecimento
        ↓
knowledge_base
        ↓
knowledge_chunks
        ↓
busca de contexto
        ↓
system prompt
        ↓
Lovable AI Gateway
        ↓
resposta
```

---

8. O site não é crawleado em tempo real a cada pergunta

O Chatbot não deve ser configurado com a expectativa de que ele "leia todo o site ao vivo" em cada interação.

O conteúdo relevante é:

- materializado;
- ingerido;
- sincronizado;

na Base de Conhecimento.

---

9. Componentes lógicos da Base de Conhecimento

No nível necessário a este SOP:

`knowledge_base`

Armazena entradas editáveis.

`knowledge_chunks`

Armazena trechos derivados utilizados na recuperação.

Busca de contexto

Recupera trechos relevantes para a pergunta.

Prompt

Recebe o contexto recuperado e aplica regras de resposta.

---

10. Origens observadas de conhecimento

A Base possui origens como:

- Manual;
- Blog sincronizado;
- FAQ sincronizada;
- Feedback convertido em conhecimento.

Conteúdo das páginas do site também pode ser materializado em entradas relacionadas à própria rota/conteúdo.

PARTE II — LIMPEZA E PREPARAÇÃO DO CONHECIMENTO

11. Passo 1 — Abrir a Base de Conhecimento

No Admin, acessar a área de conhecimento como perfil autorizado.

Confirmar:

- entradas existentes;
- origens;
- categorias;
- status ativo/inativo;
- presença de conteúdo demo.

---

12. Passo 2 — Identificar conteúdo demonstrativo

Revisar qualquer entrada genérica ou de exemplo.

> **Regra**
>
> Conteúdo demo não pode ser entregue como conhecimento real do cliente.

---

13. Passo 3 — Substituir/revisar conteúdo demonstrativo

Para cada entrada:

- manter apenas se for realmente aplicável e validada;
- adaptar quando fizer sentido;
- substituir quando for genérica;
- desativar quando não se aplica.

> **Importante**
>
> A Base observada desativa conteúdos que deixam de ser aplicáveis em alguns fluxos de sincronização, em vez de depender de exclusão automática universal.

---

14. Passo 4 — Materializar conteúdo real das páginas

Identificar páginas relevantes:

- Home;

- Sobre;
- serviços;
- especialidades;
- contato;
- outras páginas com informação útil.

Transformar as informações necessárias em conhecimento utilizável pela Base.

Objetivo

Permitir que o Chatbot responda com base em dados reais do projeto.

---

15. Passo 5 — Separar conteúdo público de informação não confirmada

Não ingerir como fato:

- informação provisória;
- placeholder;
- dado não validado;
- promessa futura;
- conteúdo contraditório.

> **DECISÃO HUMANA NECESSÁRIA**
>
> Quando houver conflito factual.

PARTE III — FAQ E BLOG

16. Passo 6 — Revisar FAQ

Quando FAQ fizer parte do projeto:

- revisar perguntas/respostas;
- confirmar que são atuais;
- remover conteúdo demo;
- sincronizar entradas aplicáveis à Base de Conhecimento.

---

17. Passo 7 — Revisar Blog

Quando posts do Blog forem utilizados como fonte de conhecimento:

- revisar conteúdos publicados;
- confirmar aplicabilidade;
- sincronizar os posts relevantes.

---

18. Conteúdo despublicado/inativo

Não presumir que conteúdo antigo continuará válido só porque foi ingerido uma vez.

Revisar status e desativar conhecimento que não deva mais ser usado.

PARTE IV — CONFIGURAÇÃO DO ASSISTENTE

19. Passo 8 — Configurar identidade e textos

Adaptar ao cliente:

- identidade visual;
- textos;
- perguntas iniciais;
- mensagens;
- apresentação do assistente.

> **Limite**
>
> Não alterar a lógica central sem necessidade.

---

20. Passo 9 — Configurar WhatsApp quando aplicável

Quando houver canal WhatsApp:

- configurar número/destino;
- validar se o canal está disponível;
- conferir comportamento do encaminhamento.

Na Base inspecionada, a configuração usa `VITE_WHATSAPP_NUMBER`.

> **Importante**
>
> Sem número válido, o Chatbot pode continuar funcionando sem oferecer WhatsApp.

---

21. Passo 10 — Confirmar dependência de IA

A versão inspecionada utiliza:

- Lovable AI Gateway;
- secret `LOVABLE_API_KEY`.

Confirmar que a dependência necessária está disponível no ambiente.

Regra documental

O **nome do modelo de IA não é requisito imutável do Método WSS**.

A implementação pode evoluir.

PARTE V — LIMITES DE RESPOSTA

22. Passo 11 — Revisar os limites configurados

O system prompt observado impõe limites como:

- responder com base no contexto recuperado;
- não completar lacunas com conhecimento próprio;
- não inventar fatos;
- encaminhar quando a informação não está confirmada.

Em contexto médico observado, também existem restrições como:

- não diagnosticar;
- não ajustar medicação;
- não interpretar exames;
- não estimar gravidade;
- não inventar preços;
- não inventar horários;
- não inventar convênios;
- não inventar endereços;
- não inventar profissionais;
- orientar avaliação profissional em dúvidas clínicas;
- oferecer atendimento humano quando necessário.

---

23. Não universalizar a lista clínica

Os limites acima são **confirmados na Base/projetos médicos inspecionados**.

Para outro setor:

- preservar a lógica de responder apenas com contexto confirmado;
- não inventar política setorial nova neste SOP;
- aplicar limites específicos somente quando sustentados pelo projeto/norma correspondente.

---

24. WhatsApp dentro do prompt

Quando WhatsApp está configurado, a implementação observada utiliza marcador controlado para gerar o botão correspondente.

Quando não está configurado:

- o assistente não deve inventar esse canal.

PARTE VI — CAPTURA DE DADOS E LEADS

25. Passo 12 — Iniciar conversa como usuário

Abrir o Chatbot no site público e iniciar uma conversa real de teste.

No fluxo observado, são exigidos:

- nome;
- telefone brasileiro válido.

Também podem existir identificadores técnicos do visitante/origem.

---

26. Passo 13 — Confirmar criação/vínculo do Lead

Depois de iniciar:

1. abrir o Admin;
2. localizar a conversa;
3. localizar o Lead;
4. confirmar vínculo;
5. conferir nome/telefone;
6. conferir origem quando disponível.

Critério

Se a conversa aparece mas o Lead obrigatório não foi criado/vinculado:

> a capacidade não está pronta.

---

27. Enriquecimento

O fluxo pode acrescentar:

- interesse;
- notas.

Quando aplicável, conferir que esse enriquecimento chega ao Lead/conversa corretamente.

PARTE VII — TESTES DE RESPOSTA

28. Passo 14 — Testar pergunta coberta

Perguntar algo claramente presente na Base de Conhecimento.

Verificar:

- resposta coerente;
- uso do conteúdo correto;
- ausência de invenção;
- linguagem adequada.

---

29. Passo 15 — Testar pergunta fora da cobertura

Perguntar algo que não esteja confirmado na Base.

Esperado:

- não inventar;
- reconhecer ausência de confirmação;
- oferecer atendimento humano quando previsto;
- encaminhar adequadamente.

---

30. Passo 16 — Testar fato sensível

Em projeto médico ou outro domínio sensível, usar pergunta que pressione o limite do assistente.

Objetivo:

- verificar se o comportamento de segurança do prompt é preservado.

Não utilizar dado real desnecessário do cliente/paciente no teste.

---

31. Passo 17 — Testar WhatsApp quando configurado

Verificar:

- oferta do canal apenas quando aplicável;
- botão/encaminhamento correto;
- destino correto.

PARTE VIII — HANDOFF HUMANO

32. Passo 18 — Solicitar atendimento humano

Como usuário:

- solicitar atendimento humano.

Confirmar:

- mudança de estado da conversa;
- ausência de resposta automática indevida depois do handoff;
- visibilidade no Admin.

---

33. Passo 19 — Assumir conversa no Admin

Como operador autorizado:

- assumir a conversa.

Confirmar que:

- a IA fica em silêncio;
- o atendimento passa ao humano.

---

34. Passo 20 — Encerrar atendimento quando aplicável

Verificar o fluxo de encerramento previsto pelo Admin.

PARTE IX — FALLBACKS TÉCNICOS

35. Comportamentos observados

A Base possui tratamento para:

- rate limit;
- falta de créditos;
- falha técnica.

Em falha técnica, o fluxo pode oferecer atendimento humano.

---

36. Este SOP não exige simular toda falha externa

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe evidência de que toda configuração de cliente precise forçar artificialmente:

- rate limit;
- indisponibilidade;
- falta de créditos;

antes de poder avançar.

Regra

Validar fallback específico quando:

- houve alteração nessa lógica;
- há suspeita de problema;
- QA dirigido exige a simulação.

PARTE X — BASE DE CONHECIMENTO COMO CAPACIDADE WEBSTAR

37. Passo 21 — Confirmar segregação funcional

Na Base inspecionada:

- `admin_webstar` possui a capability de gestão de conhecimento/IA;
- `admin_equipe` não recebe essa capability.

Verificar funcionalmente que essa separação foi preservada.

---

38. Não corrigir autorização apenas por UI

Se o perfil Equipe acessa algo indevido:

- não confiar apenas em esconder botão;
- encaminhar para validação técnica conforme Manual 6/SOP-009.

PARTE XI — BENCHMARK

39. Quando usar

A Base possui benchmark técnico real.

Executar/revisar quando:

- o projeto possui perguntas de referência;
- a configuração do Chatbot está sendo homologada;
- houve mudança relevante no conhecimento/prompt/modelo;
- o fluxo atual do projeto prevê essa validação.

> **Importante**
>
> O benchmark não substitui o teste manual.

---

40. Passo 22 — Revisar perguntas de referência

Confirmar se as perguntas ativas são:

- pertinentes;
- atuais;
- representativas;
- baseadas no conteúdo real.

Na implementação inspecionada existe limite atual de até 12 perguntas ativas por execução.

Regra documental

Esse número é detalhe da implementação atual, não requisito permanente do método.

---

41. Passo 23 — Executar/revisar benchmark

A implementação observada avalia:

- coerência;
- completude;
- compliance;
- score geral;
- flags;
- latência.

O resultado é persistido para comparação.

42. Passo 24 — Interpretar resultado

Não tratar score isolado como autorização automática.

Revisar:

- perguntas que falharam;
- causa provável;
- falta de cobertura;
- regra de prompt;
- erro factual;
- necessidade de novo conhecimento.

43. Feedback negativo

A Base também registra feedback positivo/negativo.

Perguntas negativas podem ajudar a identificar:

- lacunas;
- conhecimento insuficiente;
- respostas inadequadas.

> **Limite**
>
> Não transformar todo feedback em conhecimento automaticamente sem revisão humana.

PARTE XII — VALIDAÇÃO MANUAL ADMIN + USUÁRIO

44. Passo 25 — Repetir o fluxo como usuário

Executar um percurso real:

```
abrir Chatbot
→ identificar-se
→ iniciar conversa
→ perguntar
→ receber resposta
→ pedir humano/WhatsApp quando aplicável
```

---

45. Passo 26 — Repetir o fluxo como Admin

No Admin:

- localizar conversa;
- localizar Lead;
- conferir dados;
- assumir atendimento;
- conferir Base de Conhecimento;
- revisar benchmark quando aplicável.

---

46. Por que as duas visões são necessárias

A UI pública pode parecer correta enquanto:

- Lead não foi criado;
- conversa não chegou ao Admin;
- conhecimento não foi recuperado;
- permissão está errada;
- handoff não funciona.

PARTE XIII — PENDÊNCIAS E BLOQUEIOS

47. Passo 27 — Registrar pendências

Exemplos:

- conteúdo ainda não validado;
- FAQ pendente;
- WhatsApp ausente;
- Blog ainda não sincronizado;
- pergunta de referência faltante;
- problema de autorização;
- falha de Lead;
- ausência do secret/dependência de IA.

---

48. Classificar bloqueio

Bloqueante

Impede:

- conversa;
- resposta;
- Lead;
- handoff;
- acesso necessário;
- uso seguro do conhecimento.

Não bloqueante

Não impede o núcleo funcional, mas exige resolução antes de entrega ou operação final.

PARTE XIV — STOP CONDITION

49. Condição de parada

O SOP está concluído quando:

> **o Chatbot aplicável está alimentado com conhecimento real do cliente, sem conteúdo demo indevido, identifica o visitante, cria/vincula Lead, responde adequadamente a pergunta coberta, não inventa resposta quando não há confirmação, permite handoff humano, aparece corretamente no Admin e teve sua validação técnica complementar revisada quando aplicável.**

---

50. Checklist mínimo

Antes de encerrar:

- [ ] conteúdo demo tratado;
- [ ] conteúdo real do site materializado;
- [ ] FAQ revisada/sincronizada quando aplicável;
- [ ] Blog revisado/sincronizado quando aplicável;
- [ ] textos/identidade configurados;
- [ ] WhatsApp configurado quando aplicável;
- [ ] dependência de IA disponível;
- [ ] nome/telefone capturados;
- [ ] Lead criado/vinculado;
- [ ] conversa aparece no Admin;
- [ ] pergunta coberta testada;
- [ ] pergunta sem cobertura testada;
- [ ] limites do assistente preservados;
- [ ] handoff humano testado;
- [ ] perfil Webstar/Equipe respeitado funcionalmente;
- [ ] benchmark revisado quando aplicável;
- [ ] pendências registradas.

---

51. O que NÃO é necessário para encerrar

Não exigir:

- auditoria geral de segurança;
- QA completo do site;
- simulação forçada de toda falha externa;

- limpeza final de todos os dados de teste;
- entrega do acesso definitivo do cliente;
- publicação em domínio final.

PARTE XV — CONTINGÊNCIAS

52. O Chatbot responde algo que não está na Base

Tratar como falha.

Investigar:

- conhecimento recuperado;
- prompt;
- ingestão;
- conteúdo antigo;
- comportamento da IA.

Não corrigir adicionando conteúdo inventado.

---

53. Pergunta conhecida não é respondida

Verificar:

1. conteúdo existe em `knowledge_base`;
2. está ativo;
3. foi ingerido/chunked;
4. a busca recupera o conteúdo;
5. o prompt recebe contexto adequado.

Se necessário, seguir SOP-009.

---

54. Conversa existe, mas Lead não

Não considerar funcional.

A integração Chatbot → Lead faz parte da capacidade observada.

---

55. Lead existe, mas conversa não aparece no Admin

Não considerar funcional.

Investigar persistência/consulta/estado do Chat.

---

56. Handoff não silencia a IA

Falha funcional relevante.

Não entregar até corrigir.

---

57. WhatsApp aparece sem estar configurado

Falha de configuração.

O assistente não deve inventar canal inexistente.

---

58. Perfil Equipe consegue gerenciar conhecimento indevidamente

Tratar como problema de autorização/permite.

Não resolver apenas escondendo a tela.

---

59. Benchmark ruim

Não corrigir diretamente pelo número.

Identificar:

- pergunta problemática;
- falta de conhecimento;
- regra de prompt;
- resposta inadequada;
- cobertura insuficiente.

PARTE XVI — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

60. Não inventar

O AS-IS não sustenta afirmar que:

- todo conteúdo público é crawleado automaticamente em tempo real;
- o cliente administra a Base de Conhecimento;
- todo projeto precisa de WhatsApp;
- existe integração externa obrigatória além da infraestrutura da Base;
- benchmark substitui teste manual;
- um modelo de IA específico é permanente;
- todo projeto histórico executou benchmark;
- todos os projetos precisam usar exatamente as mesmas perguntas;
- o limite atual de perguntas do benchmark é regra do método.

PARTE XVII — NORMA TÉCNICA VIGENTE

61. Segurança e privacidade

Quando normas vigentes exigirem:

- proteção de dados;
- autorização server-side;
- não exposição de áreas privadas;
- uso adequado de secrets;

essas exigências permanecem válidas.

Para SEO/AEO/GEO, a autoridade vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Admin e áreas privadas permanecem sujeitos às regras aplicáveis de segurança, autorização e `noindex`.

Este SOP não substitui a auditoria de segurança.

---

62. Conteúdo público e arquitetura

O Chatbot pode operar em CSR como interação.

Ele não deve substituir o conteúdo principal indexável das páginas públicas nem exigir que informação essencial exista apenas dentro do Chat.

O modelo de IA e os secrets citados neste SOP permanecem detalhes da implementação atual, não requisitos normativos imutáveis.

PARTE XVIII — ANTI-PATTERNS

63. Entregar conhecimento demo

A Base demonstrativa não representa o cliente.

64. “O site é a Base de Conhecimento” entendido como crawler automático

Formulação incorreta.

O conteúdo precisa ser materializado/ingerido.

65. Alimentar a IA com fato não confirmado

Fluência não equivale a verdade.

66. Validar só pela resposta visual

É necessário conferir:

- Lead;
- conversa;
- Admin;
- handoff.

67. Validar só pelo benchmark

Benchmark não substitui uso real.

68. Dar acesso de conhecimento ao cliente por conveniência

Preservar a separação de capacidades.

69. Fixar modelo atual como padrão permanente

O modelo é detalhe de implementação.

PARTE XIX — OUTPUT

70. Saída esperada

Ao final:

Conhecimento

- conteúdo demo removido/desativado;
- conteúdo real disponível;
- FAQ/Blog integrados quando aplicáveis.

Chatbot

- identidade/textos adequados;
- resposta baseada no contexto;
- ausência de invenção quando não houver confirmação;
- WhatsApp condicional;
- handoff humano funcional.

Integração

- visitante identificado;
- Lead criado/vinculado;
- conversa visível no Admin.

Administração

- gestão de conhecimento preservada para o perfil autorizado;
- benchmark/revisão técnica utilizados quando aplicável.

Controle

- pendências conhecidas;
- capacidade pronta para QA geral.

PARTE XX — PRÓXIMO PASSO

71. Encaminhamento

Depois do SOP-006:

- retornar ao SOP-005, se houver outros módulos pendentes;
- seguir para **SOP-008 — Executar QA Visual, Funcional e Responsivo** quando a camada operacional estiver pronta.

---

72. Referências

Manual proprietário

- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`;
- `WSS_Manual_07_Publicacao_Entrega_Pos_Publicacao_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`.

Evidência técnica principal

- `WSS_VAL_002_BaseAdmin_Chatbot_v0.1.md`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-007
Portar Capacidade da Base para Projeto Legado

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-007

Portar Capacidade da Base para Projeto Legado

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 4 — Projeto-Base, Admin e Funcionalidades
MANUAIS RELACIONADOS	Manual 5 — Desenvolvimento com Codex, Lovable e GitHub; Manual 6 — Testes, QA, Debugging, Segurança e SEO; Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais
ESCOPO	AS-IS — portabilidade incremental e adaptativa de capacidades da Base Admin para projetos existentes/legados

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 7

2. Gatilho — quando usar 7

3. Quando NÃO usar 8

4. Papéis 8

5. Pré-condições 9

6. Inputs 9

7. Ferramentas 10

PARTE I — DEFINIR O RECORTE **11**

8. Passo 1 — Nomear a capacidade-alvo 12

9. Passo 2 — Descrever o resultado funcional esperado 12

10. Passo 3 — Delimitar o que está fora do recorte 12

PARTE II — INVENTARIAR O DESTINO **14**

11. Passo 4 — Ler o projeto de destino antes de editar 15

12. Passo 5 — Identificar a geração técnica 15

13. Passo 6 — Inventariar o que o destino já possui 15

PARTE III — INVENTARIAR A ORIGEM **17**

14. Passo 7 — Identificar a implementação correspondente na Base 18

15. Passo 8 — Separar capacidade de implementação 18

PARTE IV — MATRIZ ORIGEM × DESTINO **19**

16. Passo 9 — Classificar equivalências 20

17. Equivalente 20

18. Parcialmente equivalente 20

19. Ausente no destino 20

20. Melhor no destino 20

21. Incompatível 21

22. Nova funcionalidade 21

PARTE V — SEPARAR PARIDADE DE FEATURE NOVA 22

23. Passo 10 — Perguntar “isso é delta ou produto novo?” 23

24. Caso real — Leads do Guilherme Maia 23

25. Caso real — “Jornada do paciente” 23

PARTE VI — MAPEAR DEPENDÊNCIAS 24

26. Passo 11 — Mapear dependências da capacidade 25

27. Passo 12 — Classificar dependências 25

28. Regra AS-IS 26

PARTE VII — BANCO, MIGRATIONS E DADOS 27

29. Passo 13 — Verificar impacto em banco/dados 28

30. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — migrations 28

31. Dados existentes 28

32. Passo 14 — Separar migration de alteração frontend-only 29

PARTE VIII — AUTENTICAÇÃO, ROLES E PERMISSÕES 30

33. Passo 15 — Verificar se a capacidade depende de autorização 31

34. Não portar apenas a UI de permissão 31

35. Diferenças entre gerações 31

PARTE IX — DEFINIR O DELTA APROVADO 32

36. Passo 16 — Consolidar o delta 33

37. Stop intermediário 33

PARTE X — IMPLEMENTAR DEPENDÊNCIAS 34

38. Passo 17 — Portar/adaptar dependências necessárias 35

39. Regra de contenção 35

40. Validar dependência antes da camada seguinte 35

PARTE XI — IMPLEMENTAR A CAPACIDADE 36

41. Passo 18 — Adaptar a capacidade ao destino 37

42. Não perseguir identidade de código 37

43. Preservar comportamento local melhor 37

PARTE XII — ALTERAÇÕES FRONTEND-ONLY 38

44. Quando o delta é apenas frontend 39

45. Não criar migration sem necessidade 39

PARTE XIII — ALTERAÇÕES COM BACKEND/BANCO 40

46. Quando o delta exige backend 41

47. Quando exige banco 41

PARTE XIV — ESTABILIZAÇÃO 42

48. Passo 19 — Executar a capacidade no destino 43

49. Passo 20 — Corrigir incompatibilidades 43

50. Não “resolver” regressão apagando comportamento local 43

PARTE XV — VALIDAÇÃO FUNCIONAL 44

51. Passo 21 — Testar a capacidade portada 45

52. Passo 22 — Testar o que foi preservado 45

53. Passo 23 — Testar dependências 45

PARTE XVI — REGRESSÃO 46

54. Regra AS-IS 47

55. Passo 24 — Comparar antes × depois 47

56. QA completo 47

PARTE XVII — ROLLBACK E CONTINGÊNCIA 49

57. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — rollback específico de portabilidade 50

58. Recursos observados no método geral 50

59. Quando parar 50

PARTE XVIII — PROJETOS COM ARQUITETURA ANTIGA 51

60. Legado não precisa virar projeto atual inteiro 52

61. Norma técnica vigente 52

PARTE XIX — CASO GUILHERME MAIA 53

62. Papel do caso 54

63. O que o caso ensina 54

64. Leads como exemplo 54

65. Blog como exemplo 55

66. O que NÃO generalizar do caso Maia 55

PARTE XX — STOP CONDITION 56

67. Condição de parada 57

68. Checklist mínimo 57

69. O que NÃO é necessário para encerrar 57

PARTE XXI — CONTINGÊNCIAS 59

70. A Base tem algo que o destino já faz melhor 60

71. O destino já tem a capacidade 60

72. A capacidade depende de algo ausente 60

73. A dependência exigiria mudança ampla 60

74. A comparação revela feature nova 60

75. A migration ameaça dados existentes 60

76. A nova capacidade quebra comportamento local 61

77. A UI funciona, mas backend não 61

78. A portabilidade exige trocar framework 61

PARTE XXII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 62

79. Não inventar 63

PARTE XXIII — NORMA TÉCNICA VIGENTE 64

80. Normas continuam aplicáveis em camada própria 65

PARTE XXIV — ANTI-PATTERNS 66

81. “Copiar o Admin novo por cima” 67

82. Comparar arquivos sem comparar função 67

83. Refazer capacidade equivalente 67

84. Portar UI sem dependência 67

85. Chamar feature nova de “paridade” 67

86. Apagar particularidade local porque não existe na Base 67

87. Assumir migration idêntica 67

88. Transformar atualização seletiva em migração de stack 68

PARTE XXV — OUTPUT 69

89. Saída esperada 70

PARTE XXVI — PRÓXIMO PASSO 71

90. Encaminhamento 72

91. Referências 72

1. Finalidade

Portar para um projeto existente somente a capacidade realmente necessária da Base Admin atual, preservando o que o destino já possui e evitando substituição indiscriminada do painel ou da arquitetura do projeto.

Fluxo principal:

```
necessidade de evolução no legado
→ definir capacidade-alvo
→ comparar Base atual × destino
→ inventariar equivalências
→ identificar delta real
→ separar paridade de feature nova
→ identificar dependências
→ avaliar impacto em dados/auth/backend
→ portar dependências necessárias
→ adaptar a capacidade ao destino
→ estabilizar
→ validar capacidade
→ revalidar dependências/regressões
→ concluir delta
```

Princípio central:

> **A Base é referência funcional reutilizável, não autoridade absoluta para sobrescrever o projeto existente.**

---

2. Gatilho — quando usar

Usar quando um site/projeto existente precisa receber:

- uma capacidade que evoluiu na Base;
- parte de um módulo atual;
- melhoria funcional já existente na Base;
- comportamento equivalente ao padrão atual;
- atualização seletiva de Admin/Blog/Leads/Chat/CTA ou outra capacidade reutilizável.

O projeto de destino já existe e possui implementação própria.

---

3. Quando NÃO usar

Não usar:

- para criar projeto novo;
- para remixar a Base em novo cliente;
- para substituir automaticamente todo o Admin;
- para redesenhar a arquitetura completa do legado;
- para incluir funcionalidade nova apenas porque “seria melhor”;
- para executar modernização tecnológica geral;
- para converter todo projeto antigo para a stack mais recente;
- para executar QA geral sem uma portabilidade específica.

Procedimentos relacionados

- projeto novo → SOP-002;
- configuração normal da Base em cliente novo → SOP-005;
- debugging → SOP-009;
- QA → SOP-008.

---

4. Papéis

4.1 Desenvolvedor

Responsável por:

- comparar origem e destino;
- inventariar capacidade existente;
- identificar delta;
- identificar dependências;
- adaptar implementação;
- preservar comportamento local melhor;
- testar a capacidade;
- revalidar impacto.

4.2 WSS / Direção / Operações

Responsável por decidir quando houver:

- dúvida de escopo;
- funcionalidade nova;
- mudança comercial;
- risco relevante;
- necessidade de alteração ampla;
- conflito entre manter legado e evoluir capacidade.

4.3 IA / Codex / Lovable / Claude

Podem auxiliar em:

- inspeção;
- comparação;
- implementação;
- diagnóstico;
- testes.

Não decidem autonomamente:

- escopo;
- perda aceitável de funcionalidade;
- migração de dados;
- remoção de comportamento local;
- troca de arquitetura.

---

5. Pré-condições

Antes de iniciar:

1. projeto de destino identificado;
2. Base atual identificada;
3. capacidade-alvo conhecida;
4. acesso ao código/configuração necessária;
5. estado atual do destino compreendido;
6. origem e destino comparáveis no nível da capacidade;
7. responsável humano disponível para decisões de escopo quando necessário.

Não é pré-condição

O destino não precisa:

- usar a mesma stack;
- ter a mesma estrutura;
- ter o mesmo Admin;
- estar na geração técnica atual.

---

6. Inputs

Obrigatórios

- capacidade-alvo na Base;
- implementação atual do destino;

- requisito/necessidade que motivou a portabilidade.

Conforme o caso

- componentes;
- hooks;
- rotas;
- tabelas;
- migrations;
- funções;
- policies;
- autenticação;
- roles/capabilities;
- configurações;
- variáveis;
- documentação;
- testes;
- dados existentes;
- comportamento público/Admin.

---

7. Ferramentas

Podem ser usadas:

- GitHub;
- Codex;
- ambiente local;
- Lovable;
- Supabase/Lovable Cloud;
- navegador;
- DevTools;
- Claude quando houver impacto visual.

> **Regra**
>
> A ferramenta é escolhida pela camada do problema.

A mecânica Git pertence ao Manual 5.

PARTE I — DEFINIR O RECORTE

8. Passo 1 — Nomear a capacidade-alvo

Antes de comparar repositórios inteiros, formular o objetivo em termos de capacidade.

Exemplos:

- melhorar KPIs de Leads;
- trazer ações em lote;
- portar uma melhoria do Blog Admin;
- incorporar comportamento atual de Chat;
- adicionar configuração presente na Base.

Evitar formulação ampla

> "Atualizar o site antigo para ficar igual à Base."

Essa formulação é grande demais e contradiz o padrão AS-IS.

---

9. Passo 2 — Descrever o resultado funcional esperado

Responder:

- o que a capacidade deve permitir;
- quem usa;
- onde aparece;
- qual problema resolve;
- qual comportamento atual precisa permanecer.

Objetivo

Comparar **função**, não apenas arquivos.

---

10. Passo 3 — Delimitar o que está fora do recorte

Registrar explicitamente o que não deve ser alterado.

Exemplos:

- institucional;
- módulos não relacionados;
- dados existentes;
- autenticação que já funciona;

- fluxos locais superiores;
- arquitetura global.

PARTE II — INVENTARIAR O DESTINO

11. Passo 4 — Ler o projeto de destino antes de editar

Entender:

- stack;
- estrutura;
- Admin;
- backend;
- banco;
- autenticação;
- capacidade existente;
- particularidades locais.

> **Regra**
>
> Não assumir que “legado” significa “inferior”.

---

12. Passo 5 — Identificar a geração técnica

O projeto pode pertencer a geração diferente da Base atual.

Caso real Guilherme Maia:

- React;
- Vite;
- React Router;
- Supabase;
- Admin próprio já desenvolvido.

Projetos atuais podem utilizar stack diferente.

Consequência

Não copiar solução tecnicamente incompatível sem adaptação.

---

13. Passo 6 — Inventariar o que o destino já possui

Para a capacidade-alvo, listar:

- telas;
- fluxos;
- dados;

- operações;
- filtros;
- ações;
- integrações;
- permissões;
- particularidades.

Objetivo

Evitar refazer algo que já funciona.

PARTE III — INVENTARIAR A ORIGEM

14. Passo 7 — Identificar a implementação correspondente na Base

Na Base atual, localizar:

- superfície;
- fluxo;
- componentes;
- backend;
- dados;
- permissões;
- dependências.

---

15. Passo 8 — Separar capacidade de implementação

Pergunta central:

> o que é comportamento funcional reutilizável e o que é apenas detalhe da arquitetura atual da Base?

Exemplo conceitual

Uma capacidade pode existir em ambas as stacks, mas exigir implementações diferentes.

PARTE IV — MATRIZ ORIGEM × DESTINO

16. Passo 9 — Classificar equivalências

Para cada parte da capacidade, marcar como:

- equivalente;
- parcialmente equivalente;
- ausente no destino;
- existente e melhor no destino;
- incompatível;
- nova funcionalidade.

---

17. Equivalente

Se origem e destino já resolvem a mesma necessidade:

> **não refazer por padrão.**

---

18. Parcialmente equivalente

Identificar exatamente o delta.

Evitar trocar o módulo inteiro.

---

19. Ausente no destino

Pode ser candidato à portabilidade.

Ainda precisa passar por:

- dependências;
- compatibilidade;
- escopo;
- risco.

---

20. Melhor no destino

Preservar a solução local, salvo decisão humana contrária.

> **Princípio**

> evolução não significa regressão para igualar a Base.

---

21. Incompatível

Se a implementação da Base depende de arquitetura inexistente no destino:

- não copiar diretamente;
- identificar o comportamento desejado;
- adaptar à arquitetura local;
- avaliar custo/risco.

---

22. Nova funcionalidade

Se algo existe na Base ou surge durante comparação, mas não corresponde ao objetivo aprovado:

> **não classificar como paridade.**

Exige decisão de escopo.

PARTE V — SEPARAR PARIDADE DE FEATURE NOVA

23. Passo 10 — Perguntar “isso é delta ou produto novo?”

Para cada melhoria:

- já existe na Base?
- resolve capacidade já prevista?
- é necessária para equivalência?
- adiciona jornada ou regra nova?

---

24. Caso real — Leads do Guilherme Maia

O destino já possuía:

- CRUD;
- busca/filtros;
- score;
- urgência;
- ficha/drawer;
- timeline;
- exportação.

Esses itens não deveriam ser refeitos.

Deltas identificados incluíam, por exemplo:

- KPIs;
- gráficos;
- ações em lote.

---

25. Caso real — “Jornada do paciente”

Foi explicitamente separada da portabilidade porque representava:

> **funcionalidade nova, não paridade.**

> **Regra**
>
> Não aproveitar uma portabilidade para ampliar silenciosamente o produto.

PARTE VI — MAPEAR DEPENDÊNCIAS

26. Passo 11 — Mapear dependências da capacidade

Antes de portar UI ou comportamento, investigar se depende de:

- tabela;
- coluna;
- função;
- RPC;
- policy;
- role;
- capability;
- endpoint;
- hook;
- componente;
- configuração;
- variável;
- package;
- migration.

---

27. Passo 12 — Classificar dependências

Separar:

Já existe no destino

Reutilizar quando compatível.

Existe parcialmente

Adaptar.

Não existe

Avaliar necessidade de portabilidade.

É específica da Base

Não transportar automaticamente.

---

28. Regra AS-IS

> **Dependências estruturais necessárias devem ser tratadas antes da capacidade que depende delas.**

PARTE VII — BANCO, MIGRATIONS E DADOS

29. Passo 13 — Verificar impacto em banco/dados

Se a capacidade exige mudança persistente, identificar:

- tabela envolvida;
- estrutura atual;
- dados existentes;
- compatibilidade;
- necessidade de migration.

---

30. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — migrations

Não existe hoje um procedimento universal recorrente e formal para toda portabilidade contendo:

- modelo único de migration;
- ordem única;
- ferramenta única;
- checklist único.

Consequência

Não inventar sequência universal.

A migration deve ser tratada conforme:

- projeto;
- stack;
- banco;
- risco;
- dados existentes.

---

31. Dados existentes

A descoberta sustenta o princípio de **preservar o que o destino já possui**.

Por isso:

- não apagar dados por conveniência;
- não substituir estrutura útil apenas para igualar a Base;
- não presumir que migration destrutiva é aceitável.

DECISÃO HUMANA NECESSÁRIA

Quando houver risco real de perda/transformação de dados.

---

32. Passo 14 — Separar migration de alteração frontend-only

Comparações reais já distinguiram:

- fases apenas frontend;
- fases que exigiam migration;
- integrações que exigiriam desenho adicional.

Essa distinção deve ser feita antes de implementar.

PARTE VIII — AUTENTICAÇÃO, ROLES E PERMISSÕES

33. Passo 15 — Verificar se a capacidade depende de autorização

Identificar:

- quem deve acessar;
- quem não deve acessar;
- role/capability;
- regra server-side;
- RLS quando aplicável.

---

34. Não portar apenas a UI de permissão

Se a capacidade depende de autorização:

- preservar regra efetiva;
- não considerar ocultação visual como autorização suficiente.

Ver

Manual 6 / validação de segurança.

---

35. Diferenças entre gerações

Destino e Base podem ter modelos de permissão diferentes.

Não substituir a estrutura local sem análise.

Adaptar o requisito funcional ao modelo compatível.

PARTE IX — DEFINIR O DELTA APROVADO

36. Passo 16 — Consolidar o delta

Antes de editar, produzir uma lista enxuta:

Preservar

O que o destino já faz.

Portar

O que falta e está aprovado.

Adaptar

O que precisa mudar para ser compatível.

Excluir do recorte

Feature nova ou item sem ganho.

Decidir

Questões que dependem de humano.

---

37. Stop intermediário

Se o delta ainda não consegue ser descrito claramente:

> **não iniciar portabilidade.**

Voltar à comparação.

PARTE X — IMPLEMENTAR DEPENDÊNCIAS

38. Passo 17 — Portar/adaptar dependências necessárias

Tratar primeiro o suporte requerido.

Pode incluir, conforme caso:

- banco;
- funções;
- permissões;
- componentes base;
- packages;
- configuração.

---

39. Regra de contenção

Portar apenas dependência necessária.

Não importar subsistema inteiro apenas porque veio junto na Base.

---

40. Validar dependência antes da camada seguinte

Quando possível:

- confirmar que a dependência existe;
- confirmar compatibilidade;
- confirmar que não quebrou comportamento existente.

PARTE XI — IMPLEMENTAR A CAPACIDADE

41. Passo 18 — Adaptar a capacidade ao destino

Implementar o delta aprovado respeitando:

- stack do destino;
- componentes locais;
- autenticação local;
- dados existentes;
- identidade;
- fluxos existentes.

---

42. Não perseguir identidade de código

O objetivo é:

> **equivalência/ganho funcional compatível.**

Não é:

> arquivos idênticos à Base.

---

43. Preservar comportamento local melhor

Se o destino resolve uma parte melhor:

- manter;
- integrar o novo delta ao redor;
- evitar regressão funcional.

PARTE XII — ALTERAÇÕES FRONTEND-ONLY

44. Quando o delta é apenas frontend

Se a capacidade não exige mudança de dados/backend:

- limitar a alteração ao frontend;
- preservar backend;
- validar interação e regressão.

---

45. Não criar migration sem necessidade

A simples existência de migration na Base não significa que o destino precise da mesma migration.

Comparar estrutura real.

PARTE XIII — ALTERAÇÕES COM BACKEND/BANCO

46. Quando o delta exige backend

Verificar:

- compatibilidade de função;
- contrato;
- payload;
- autenticação;
- retorno;
- dependências.

---

47. Quando exige banco

Tratar projeto-especificamente:

- estrutura;
- dados;
- migration;
- autorização;
- regressão.

Ausência

Não existe receita única formal do AS-IS.

PARTE XIV — ESTABILIZAÇÃO

48. Passo 19 — Executar a capacidade no destino

Depois da implementação:

- abrir;
- navegar;
- executar;
- observar;
- corrigir falhas locais.

---

49. Passo 20 — Corrigir incompatibilidades

Problemas podem surgir por:

- stack diferente;
- dependência diferente;
- dado diferente;
- autenticação;
- componente local;
- expectativa herdada da Base.

Corrigir preservando o destino.

---

50. Não “resolver” regressão apagando comportamento local

Se o conflito decorre de uma particularidade importante do projeto:

> reavaliar a adaptação.

PARTE XV — VALIDAÇÃO FUNCIONAL

51. Passo 21 — Testar a capacidade portada

Executar o fluxo principal da capacidade.

Exemplo genérico:

```
abrir módulo
→ executar ação
→ persistir/consultar quando aplicável
→ confirmar resultado
```

---

52. Passo 22 — Testar o que foi preservado

Revalidar as funções que já existiam no destino e não deveriam ter sido alteradas.

---

53. Passo 23 — Testar dependências

Se houve alteração em:

- banco;
- autenticação;
- API;
- componente compartilhado;

revalidar as áreas dependentes.

PARTE XVI — REGRESSÃO

54. Regra AS-IS

Depois da portabilidade, o destino passa por:

- estabilização;
- correções;
- QA.

Não basta “a nova tela abriu”.

---

55. Passo 24 — Comparar antes × depois

Perguntar:

- o delta apareceu?
- o que já funcionava continua funcionando?
- dados foram preservados?
- permissões permanecem corretas?
- particularidades locais foram mantidas?

---

56. QA completo

O QA geral pertence ao SOP-008.

Este SOP exige a validação dirigida da:

- capacidade;
- dependências;
- áreas diretamente afetadas.

56.1 Retorno controlado de melhoria/correção à Base

Se a portabilidade ou a execução real no projeto revelar uma melhoria/correção potencialmente reutilizável, avaliar o fluxo inverso antes de encerrar a rodada:

```
melhoria/correção validada no projeto
→ verificar se é reutilizável
→ remover dados e regras específicas do cliente
→ comparar com a Base vigente
→ avaliar risco/regressão
→ incorporar à Base somente quando aprovado
→ validar a Base
```

Esse retorno é controlado e humano. Não constitui sincronização automática Base ↔ projetos e não autoriza incorporar à Base toda particularidade do cliente.

PARTE XVII — ROLLBACK E CONTINGÊNCIA

57. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — rollback específico de portabilidade

Não existe hoje um ritual universal formal para rollback de toda evolução Base → legado.

---

58. Recursos observados no método geral

Conforme o projeto, recuperação pode usar:

- histórico Git;
- commit anterior;
- histórico/plataforma quando disponível;
- restauração orientada da versão funcional.

> **Limite**
>
> Este SOP não cria uma política nova de rollback.

---

59. Quando parar

Se a portabilidade:

- ameaça dados;
- quebra funções importantes;
- exige mudança arquitetural ampla não aprovada;
- virou feature nova;
- ultrapassou o delta definido;

parar e retornar à decisão humana.

PARTE XVIII — PROJETOS COM ARQUITETURA ANTIGA

60. Legado não precisa virar projeto atual inteiro

O fato de a Base atual usar geração tecnológica diferente não implica:

- refatorar tudo;
- migrar framework;
- converter toda renderização;
- substituir roteamento;
- reconstruir Admin.

---

61. Norma técnica vigente

Pode existir norma atual aplicável a novos projetos ou a determinadas superfícies.

> **Regra**
>
> Não usar este SOP para declarar automaticamente que todo legado precisa ser convertido integralmente à arquitetura normativa atual.

Quando norma vigente exigir intervenção no legado:

> tratar como decisão/projeto próprio, com escopo explícito.

PARTE XIX — CASO GUILHERME MAIA

62. Papel do caso

`maia-uro-shine` é a principal evidência real de projeto antigo usado para comparação com a Base.

---

63. O que o caso ensina

O projeto já possuía Admin desenvolvido.

As comparações não partiram de:

> “substituir pelo Admin novo”.

Partiram de:

> “o que falta aqui que vale portar?”

---

64. Leads como exemplo

Preservar:

- CRUD;
- busca/filtros;
- score;
- urgência;
- ficha;
- timeline;
- exportação.

Avaliar como delta:

- KPIs;
- gráficos;
- ações em lote.

Excluir do recorte quando não aprovado:

- nova jornada funcional.

---

65. Blog como exemplo

Comparações reais distinguiram:

- capacidades existentes;
- lacunas;
- frontend-only;
- migration;
- integrações que exigiriam desenho adicional.

Lição

O delta pode ter naturezas técnicas diferentes dentro do mesmo módulo.

---

66. O que NÃO generalizar do caso Maia

Não generalizar:

- stack como definição de todo legado;
- módulos específicos como obrigatórios;
- deltas de Leads como padrão para todo projeto;
- migrations específicas;
- estrutura concreta do Admin.

Generalizar apenas o princípio de portabilidade incremental/adaptativa.

PARTE XX — STOP CONDITION

67. Condição de parada

O SOP está concluído quando:

> **o delta funcional aprovado foi portado/adaptado ao projeto de destino, as dependências necessárias foram tratadas, as capacidades locais que deveriam ser preservadas continuam funcionando e a alteração passou por validação dirigida e regressão proporcional nas áreas afetadas.**

---

68. Checklist mínimo

Antes de encerrar:

- [ ] capacidade-alvo definida;
- [ ] destino inventariado;
- [ ] Base inventariada no recorte;
- [ ] equivalências identificadas;
- [ ] delta real identificado;
- [ ] feature nova separada;
- [ ] dependências mapeadas;
- [ ] impacto em banco/dados identificado;
- [ ] impacto em autenticação/permissões identificado;
- [ ] itens a preservar registrados;
- [ ] delta aprovado implementado;
- [ ] dependências tratadas;
- [ ] capacidade testada;
- [ ] comportamento local preservado testado;
- [ ] regressão dirigida executada;
- [ ] pendências registradas.

---

69. O que NÃO é necessário para encerrar

Não exigir:

- igualdade integral com a Base;
- mesma stack;
- mesma arquitetura;
- substituição de todo Admin;
- migração completa do projeto;

- Design System idêntico;
- eliminação de toda divergência histórica;
- modernização tecnológica geral.

PARTE XXI — CONTINGÊNCIAS

70. A Base tem algo que o destino já faz melhor

Preservar o destino.

Portar somente o ganho real.

---

71. O destino já tem a capacidade

Não refazer.

Verificar se existe delta útil.

---

72. A capacidade depende de algo ausente

Portar/adaptar a dependência necessária antes.

Não implementar só a camada superficial.

---

73. A dependência exigiria mudança ampla

Parar e avaliar escopo/risco.

---

74. A comparação revela feature nova

Não implementar automaticamente.

Retornar à gestão.

---

75. A migration ameaça dados existentes

Não prosseguir por inferência.

> **DECISÃO HUMANA NECESSÁRIA.**

---

76. A nova capacidade quebra comportamento local

Reverter/corrigir o delta.

Não considerar a portabilidade concluída.

---

77. A UI funciona, mas backend não

Capacidade incompleta.

Seguir debugging por camada.

---

78. A portabilidade exige trocar framework

Isso deixou de ser uma portabilidade incremental simples.

Retornar à decisão de escopo.

PARTE XXII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

79. Não inventar

O AS-IS não possui:

- sincronização automática contínua Base ↔ projetos/sites;
- substituição integral obrigatória do Admin;
- algoritmo automático de delta;
- matriz universal de compatibilidade;
- regra única de migrations;
- procedimento único de preservação de dados;
- ritual único de rollback;
- obrigação de atualizar stack do legado;
- regra universal de branch para portabilidade;
- threshold numérico de risco;
- obrigação de portar toda melhoria nova da Base.

PARTE XXIII — NORMA TÉCNICA VIGENTE

80. Normas continuam aplicáveis em camada própria

Quando uma alteração toca área sujeita a norma vigente:

- segurança;
- SEO;
- arquitetura pública;
- privacidade;
- autenticação;

a norma deve ser considerada.

Para SEO/AEO/GEO, a autoridade vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Princípio de não regressão

- divergência do legado não reduz a norma vigente;
- retrofit integral do projeto legado não é automático;
- risco, impacto, custo, contrato, prioridade e ciclo podem orientar a estratégia e o momento de remediação.

Isso não altera o princípio central deste SOP: a portabilidade permanece incremental e adaptativa.

> **Limite**
>
> Norma atual não deve ser usada para reescrever retroativamente o AS-IS do projeto legado sem decisão explícita.

PARTE XXIV — ANTI-PATTERNS

81. “Copiar o Admin novo por cima”

Contradiz o padrão observado.

82. Comparar arquivos sem comparar função

Pode produzir falsa divergência.

83. Refazer capacidade equivalente

Gera custo e risco sem ganho.

84. Portar UI sem dependência

Cria módulo visualmente presente, mas funcionalmente incompleto.

85. Chamar feature nova de “paridade”

Mascaramento de escopo.

86. Apagar particularidade local porque não existe na Base

A Base não é autoridade absoluta sobre o destino.

87. Assumir migration idêntica

Estrutura/dados podem ser diferentes.

88. Transformar atualização seletiva em migração de stack

Escopo diferente.

PARTE XXV — OUTPUT

89. Saída esperada

Ao final:

Comparação

- origem e destino compreendidos;
- equivalências conhecidas;
- delta delimitado.

Preservação

- capacidades locais úteis mantidas;
- dados relevantes preservados;
- particularidades necessárias preservadas.

Portabilidade

- dependências necessárias tratadas;
- capacidade adaptada ao destino;
- feature nova excluída ou decidida separadamente.

Validação

- capacidade funciona;
- dependências funcionam;
- regressão dirigida concluída;
- pendências registradas.

PARTE XXVI — PRÓXIMO PASSO

90. Encaminhamento

Depois do SOP-007:

- executar **SOP-008 — Executar QA Visual, Funcional e Responsivo** quando a alteração integrar uma rodada maior;
- usar **SOP-009 — Diagnosticar e Corrigir Bug por Camada** se houver falha durante estabilização.

---

91. Referências

Manual proprietário

- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`;
- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`.

Evidência técnica complementar

- `WSS_VAL_004_Troubleshooting_Estudos_de_Caso_v0.1.md`.

Caso real principal

- Dr. Guilherme Maia — `maia-uro-shine`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-008
Executar QA Visual, Funcional e Responsivo

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-008

Executar QA Visual, Funcional e Responsivo

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 6 — Testes, QA, Debugging, Segurança e SEO
MANUAIS RELACIONADOS	Manual 3 — Construção Visual; Manual 4 — Projeto-Base, Admin e Funcionalidades; Manual 7 — Publicação, Entrega e Pós-Publicação; Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais
ESCOPO	AS-IS — validação visual, responsiva e funcional antes da revisão final/ publicação

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 7

2. Gatilho — quando usar 7

3. Quando NÃO usar 7

4. Papéis 8

5. Pré-condições 9

6. Inputs 9

7. Ferramentas e ambientes 10

PARTE I — PREPARAR A RODADA 11

8. Passo 1 — Confirmar versão e ambiente 12

9. Passo 2 — Definir escopo do QA 12

10. Passo 3 — Revisar Sitemap e módulos aplicáveis 12

PARTE II — QA VISUAL 14

11. Passo 4 — Percorrer página por página 15

12. Passo 5 — Observar fidelidade visual 15

13. Passo 6 — Verificar tipografia 15

14. Passo 7 — Verificar espaçamentos e alinhamentos 16

15. Passo 8 — Verificar imagens e assets 16

16. Passo 9 — Verificar efeitos/animações 16

PARTE III — QA RESPONSIVO 18

17. Três cenários principais do AS-IS 19

18. Passo 10 — Validar desktop 19

19. Passo 11 — Validar notebook 19

20. Passo 12 — Validar mobile físico 20

21. Passo 13 — Procurar scroll horizontal 20

22. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — viewports intermediários 20

PARTE IV — NAVEGAÇÃO E ROTAS 22

23. Passo 14 — Testar navegação principal 23

24. Passo 15 — Testar CTAs 23

25. Passo 16 — Testar rotas públicas principais 23

26. 404 24

PARTE V — QA DE CTAS, FORMULÁRIOS E LEADS **25**

27. Passo 17 — Identificar fluxos de captação 26

28. Passo 18 — Executar formulário ponta a ponta 26

29. Passo 19 — Conferir Lead 26

30. Passo 20 — Testar erros evidentes do formulário 27

PARTE VI — QA DE CHAT / CTA / WHATSAPP **28**

31. Passo 21 — Testar Chat 29

32. Passo 22 — Testar WhatsApp 29

PARTE VII — QA DE CHATBOT **30**

33. Quando Chatbot se aplica 31

34. Passo 23 — Revalidar fluxo básico do Chatbot 31

35. Teste dirigido adicional 31

PARTE VIII — QA DO ADMIN **32**

36. Passo 24 — Abrir Admin 33

37. Passo 25 — Percorrer módulo por módulo 33

38. Passo 26 — Executar função essencial de cada módulo 33

PARTE IX — QA DE PERFIS E ACESSO **35**

39. Passo 27 — Testar perfil Equipe/cliente 36

40. Passo 28 — Testar perfil Webstar 36

41. Limite 36

PARTE X — DADOS DE TESTE **37**

42. Passo 29 — Usar dados claramente fictícios 38

43. Não limpar cedo demais 38

PARTE XI — ERROS, EMPTY STATES E CASOS NÃO COBERTOS **39**

44. Empty states 40

45. Estados de erro 40

PARTE XII — USO DO CODEX NO QA **41**

46. Passo 30 — Solicitar rodada ampla ao Codex quando útil 42

47. Não delegar decisão final 42

48. Build e typecheck 42

PARTE XIII — REGISTRAR PROBLEMAS **43**

49. Passo 31 — Registrar cada falha reproduzível 44

50. Classificar por camada 44

PARTE XIV — CORREÇÃO E REVALIDAÇÃO **45**

51. Passo 32 — Corrigir problemas localizados 46

52. Passo 33 — Encaminhar problema não trivial 46

53. Passo 34 — Revalidar exatamente o fluxo que falhou 46

PARTE XV — REGRESSÃO **47**

54. Passo 35 — Revalidar áreas impactadas 48

55. Regressão visual 48

56. Regressão funcional 48

57. Regressão de perfil 48

PARTE XVI — REVISÃO HUMANA ADICIONAL **49**

58. Revisão adicional em paralelo 50

PARTE XVII — PROCESSO SITES WSS / MARCO 8 **51**

59. Relação com o Marco 8 52

60. Processo ≠ SOP 52

PARTE XVIII — STOP CONDITION **53**

61. Condição de parada 54

62. Checklist mínimo 54

63. O que NÃO é necessário para encerrar 54

PARTE XIX — CONTINGÊNCIAS **56**

64. Visual correto em desktop e quebrado em mobile 57

65. CTA termina em 404 57

66. Formulário mostra sucesso mas Lead não aparece 57

67. Chat abre mas conversa não chega ao Admin 57

68. Admin abre para Webstar, mas não para Equipe 57

69. Efeito quebra somente em mobile 58

70. Codex aponta problema que o Dev não reproduz 58

PARTE XX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 59

71. Não inventar 60

PARTE XXI — NORMA TÉCNICA VIGENTE 61

72. Normas complementam o QA 62

PARTE XXII — ANTI-PATTERNS 63

73. Validar só a Home 64

74. Validar só desktop 64

75. Validar formulário apenas visualmente 64

76. Validar Chat apenas pelo widget 64

77. Confiar em primeira saída do Codex 64

78. Corrigir sem revalidar 64

79. Testar só área corrigida quando componente é compartilhado 64

PARTE XXIII — OUTPUT 65

80. Saída esperada 66

PARTE XXIV — PRÓXIMO PASSO 67

81. Encaminhamento 68

82. Referências 68

1. Finalidade

Verificar se o site e sua camada operacional estão visualmente corretos, navegáveis e funcionalmente utilizáveis antes de avançar para revisão final, homologação e publicação.

Fluxo principal:

```
páginas suficientemente desenvolvidas
→ atualizar/abrir ambiente
→ teste visual
→ teste responsivo
→ teste funcional do site público
→ teste de formulários/Leads
→ teste de Chat/Chatbot quando aplicável
→ teste do Admin
→ teste de perfis quando aplicável
→ registrar problemas
→ corrigir/revalidar
→ executar regressão dirigida
→ QA concluído
```

Este SOP não cria uma suíte automatizada universal.

Ele organiza o QA praticado hoje em uma sequência reproduzível.

---

2. Gatilho — quando usar

Usar quando:

- as páginas públicas estão suficientemente desenvolvidas;
- a Home e as páginas internas já passaram pela fase principal de construção;
- módulos aplicáveis do Admin estão configurados ou suficientemente maduros para teste;
- o projeto está entrando na revisão interna;
- uma rodada ampla de alterações exige revalidação.

Também pode ser usado de forma parcial durante o desenvolvimento para validar uma área específica.

---

3. Quando NÃO usar

Não usar como substituto de:

- construção visual;
- debugging detalhado;
- auditoria SEO/AEO/GEO;
- auditoria de segurança;

- publicação;
- smoke de produção específico;
- entrega do Admin.

Procedimentos relacionados

- bug identificado → SOP-009;
- SEO/AEO/GEO → SOP-010;
- feedback/ajustes → SOP-011;
- publicação/entrega → SOP-012.

---

4. Papéis

4.1 Desenvolvedor

Responsável por:

- executar validação manual;
- definir o percurso de teste;
- julgar qualidade visual;
- reproduzir problemas;
- corrigir ou encaminhar bugs;
- decidir quando a rodada está suficientemente validada.

4.2 Codex

Pode ser usado para:

- rodada ampla de testes;
- navegação automatizada;
- build/typecheck quando aplicável;
- inspeção de erros;
- comparação;
- validação dirigida;
- regressão.

4.3 Lovable

Pode apoiar em:

- bugs funcionais;
- backend;
- banco;
- autenticação;
- Supabase;
- correções específicas.

4.4 Claude

Pode apoiar em:

- problemas visuais;
- refinamento;
- responsividade;
- efeitos.

4.5 WSS / revisão humana

Pode realizar segunda revisão interna.

A revisão humana não é substituída pela automação.

---

5. Pré-condições

Antes de iniciar:

1. projeto acessível;
2. versão atual conhecida;
3. páginas principais implementadas;
4. Sitemap disponível para conferir cobertura;
5. módulos aplicáveis identificados;
6. dados fictícios de teste disponíveis;
7. perfis de teste disponíveis quando aplicável;
8. ambiente suficientemente estável para navegação.

Se a versão estiver desatualizada

Atualizar antes do QA.

Não validar uma versão sabidamente superada.

---

6. Inputs

Conforme o projeto:

- Sitemap;
- Contexto de Desenvolvimento;
- Home e páginas internas;
- `design.md`;
- referências visuais;
- lista de módulos aplicáveis;
- perfis de teste;

- dados fictícios;
- `ajustes.md`, se a rodada for pós-feedback;
- resultado de rodada anterior;
- erros conhecidos.

---

7. Ferramentas e ambientes

Podem ser usados:

- navegador;
- DevTools;
- desktop/monitor;
- notebook;
- mobile físico;
- Codex;
- Lovable;
- Claude;
- Admin;
- ambiente local;
- preview publicado quando aplicável.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe matriz universal de:

- navegadores;
- sistemas operacionais;
- resoluções;
- dispositivos.

O AS-IS prioriza cenários reais de uso do Dev.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE I — PREPARAR A RODADA

8. Passo 1 — Confirmar versão e ambiente

Antes de testar:

- confirmar que o site é a versão que deveria ser validada;
- confirmar que não há alteração pendente conhecida;
- confirmar ambiente correto.

Objetivo

Evitar falso bug ou validação inválida por ambiente incorreto.

---

9. Passo 2 — Definir escopo do QA

Classificar a rodada como:

QA completo

Quando:

- site está entrando em revisão interna;
- houve muitas alterações;
- publicação se aproxima.

QA dirigido

Quando:

- mudança localizada;
- correção específica;
- módulo isolado;
- regressão pós-fix.

> **Regra**
>
> O escopo da validação deve acompanhar o risco da alteração.

---

10. Passo 3 — Revisar Sitemap e módulos aplicáveis

Antes de navegar, lembrar:

- quais páginas deveriam existir;
- quais funcionalidades se aplicam;

- quais módulos são opcionais.

Não fazer

Testar ausência de módulo não contratado como erro.

PARTE II — QA VISUAL

11. Passo 4 — Percorrer página por página

Percorrer o site público:

```
Home
→ páginas internas
→ Blog quando aplicável
→ demais rotas públicas
```

Em cada página:

- iniciar no Hero;
- seguir seção por seção;
- terminar no Footer.

---

12. Passo 5 — Observar fidelidade visual

Verificar:

- direção visual;
- consistência;
- tipografia;
- paleta;
- proporções;
- imagens;
- ritmo;
- cards;
- CTAs;
- comportamento.

> **Importante**
>
> O QA não exige que toda página seja idêntica à Home.

Exige coerência com o sistema visual do projeto.

---

13. Passo 6 — Verificar tipografia

Observar:

- fonte correta;

- peso;
- hierarquia;
- tamanho;
- quebra de linha;
- legibilidade;
- headings;
- textos excessivamente comprimidos/longos.

---

14. Passo 7 — Verificar espaçamentos e alinhamentos

Observar:

- margens;
- paddings;
- gaps;
- alinhamento;
- largura de conteúdo;
- consistência entre seções.

---

15. Passo 8 — Verificar imagens e assets

Conferir:

- imagem correta;
- crop;
- foco;
- proporção;
- qualidade;
- ausência de stretching;
- ausência de imagem demo indevida.

---

16. Passo 9 — Verificar efeitos/animações

Observar:

- entrada;
- scroll;
- hover;
- transição;
- parallax;

- motion;
- efeito 3D quando aplicável.

Falhas típicas

- trava;
- atraso excessivo;
- pisca;
- elemento some;
- sobreposição;
- scroll horizontal;
- efeito inadequado em mobile.

PARTE III — QA RESPONSIVO

17. Três cenários principais do AS-IS

O Dev prioriza:

1. **Desktop** — MacBook em monitor;
2. **Notebook** — MacBook;
3. **Mobile físico** — aparelho real via rede/local quando aplicável.

---

18. Passo 10 — Validar desktop

Percorrer:

- Home;
- páginas principais;
- Header;
- menus;
- CTAs;
- formulários;
- efeitos;
- Footer.

Observar:

- largura;
- densidade;
- alinhamentos;
- proporções;
- comportamento de hover/motion.

---

19. Passo 11 — Validar notebook

Repetir percurso relevante.

Objetivo:

- detectar layout que funciona em monitor maior mas quebra em largura menor;
- observar textos;
- cards;
- imagem;
- menus.

---

20. Passo 12 — Validar mobile físico

Abrir em aparelho real.

Observar:

- menu;
- Hero;
- imagens;
- textos;
- botões;
- formulários;
- Chat/CTA;
- efeitos;
- scroll;
- toque;
- Footer.

> **Atenção**
>
> O mobile pode usar solução diferente do desktop.

Não exigir replicação literal de efeitos.

---

21. Passo 13 — Procurar scroll horizontal

Em mobile:

- percorrer a página;
- observar elementos fora da viewport;
- testar componentes com movimento;
- conferir carrosséis/cards.

Scroll horizontal inesperado deve ser tratado.

---

22. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — viewports intermediários

Não existe hoje uma matriz grande e obrigatória de breakpoints testados manualmente em todo projeto.

Consequência

Não documentar uma matriz fictícia.

Viewports adicionais podem ser testados quando:

- existe suspeita;
- componente é sensível;
- alteração afeta breakpoint;
- QA dirigido exige.

PARTE IV — NAVEGAÇÃO E ROTAS

23. Passo 14 — Testar navegação principal

Verificar:

- logo;
- navbar;
- menu mobile;
- links do Header;
- Footer;
- links contextuais.

---

24. Passo 15 — Testar CTAs

Clicar nos CTAs principais.

Confirmar:

- destino;
- rota;
- ação;
- comportamento.

Caso real Daniel P

CTA apontava para:

`/sobre-mim`

mas a rota existente era:

`/sobre`.

O resultado era 404.

Lição

Todo CTA relevante precisa ser clicado.

---

25. Passo 16 — Testar rotas públicas principais

Abrir diretamente páginas importantes.

Verificar:

- renderização;
- conteúdo;

- navegação;
- ausência de erro.

---

26. 404

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> O AS-IS não possui uma rotina sistemática universal de testar toda classe de 404 em todo projeto.

Aplicação

Testar quando:

- rota foi alterada;
- auditoria apontou problema;
- link foi corrigido;
- projeto possui comportamento crítico de rota.

Normas técnicas podem exigir 404 real em páginas públicas.

PARTE V — QA DE CTAs, FORMULÁRIOS E LEADS

27. Passo 17 — Identificar fluxos de captação

Mapear:

- formulário;
- CTA;
- agendamento;
- Chat;
- WhatsApp;
- outras entradas de Lead.

---

28. Passo 18 — Executar formulário ponta a ponta

Fluxo:

1. abrir;
2. preencher com dados fictícios;
3. enviar;
4. observar mensagem de sucesso/erro;
5. abrir Admin;
6. localizar Lead;
7. conferir dados.

---

29. Passo 19 — Conferir Lead

Verificar:

- nome;
- origem;
- interesse;
- especialidade/opção quando aplicável;
- trajetória/source quando disponível.

Critério

Sucesso visual sem persistência no Admin não é suficiente.

---

30. Passo 20 — Testar erros evidentes do formulário

Quando aplicável:

- campo obrigatório vazio;
- telefone inválido;
- envio incompleto.

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe matriz universal e exaustiva de validação negativa para todo formulário.

PARTE VI — QA DE CHAT / CTA / WHATSAPP

31. Passo 21 — Testar Chat

Executar conversa.

Confirmar:

- widget abre;
- interação funciona;
- dados são coletados quando aplicável;
- conversa aparece no Admin.

---

32. Passo 22 — Testar WhatsApp

Quando aplicável:

- clicar no destino;
- conferir número;
- conferir mensagem;
- confirmar canal correto.

Se não se aplica:

- ausência não é erro.

PARTE VII — QA DE CHATBOT

33. Quando Chatbot se aplica

O QA detalhado de configuração pertence ao SOP-006.

Neste SOP, revalidar apenas que a capacidade continua funcionando dentro do site completo.

---

34. Passo 23 — Revalidar fluxo básico do Chatbot

Verificar:

- abre;
- captura dados;
- inicia conversa;
- responde;
- conversa aparece no Admin;
- Lead aparece quando obrigatório;
- handoff continua possível.

---

35. Teste dirigido adicional

Se houve alteração em Chatbot/IA/Base de Conhecimento:

- usar SOP-006;
- executar benchmark quando aplicável;
- aumentar a profundidade do QA.

PARTE VIII — QA DO ADMIN

36. Passo 24 — Abrir Admin

Confirmar:

- login;
- carregamento;
- navegação;
- módulos aplicáveis.

---

37. Passo 25 — Percorrer módulo por módulo

Conforme projeto:

- Leads;
- Blog;
- Chat;
- Chatbot;
- FAQ;
- depoimentos;
- configurações;
- Search;
- Performance;
- outros.

Não testar módulo não aplicável como falha.

---

38. Passo 26 — Executar função essencial de cada módulo

Exemplos:

Leads

- abrir;
- buscar;
- conferir registro de teste.

Blog

- abrir gestão;
- conferir conteúdo/configuração;

- validar reflexo público quando necessário.

Chat

- localizar conversa;
- assumir/encerrar quando aplicável.

Configurações

- conferir dados relevantes.

PARTE IX — QA DE PERFIS E ACESSO

39. Passo 27 — Testar perfil Equipe/cliente

Confirmar funcionalmente:

- acessa o que deve;
- não vê superfície Webstar-only evidente;
- opera módulos previstos.

---

40. Passo 28 — Testar perfil Webstar

Confirmar:

- acesso ampliado esperado;
- módulos administrativos apropriados;
- funções Webstar-only quando aplicáveis.

---

41. Limite

Este SOP testa o comportamento funcional visível.

Não substitui auditoria de:

- RLS;
- server-side authorization;
- privilege escalation;
- secrets.

Esses temas pertencem ao Manual 6 e às validações técnicas aplicáveis.

PARTE X — DADOS DE TESTE

42. Passo 29 — Usar dados claramente fictícios

Durante QA:

- Leads;
- conversas;
- posts;
- outros registros;

devem ser identificáveis como teste.

---

43. Não limpar cedo demais

Dados de teste podem ser úteis durante:

- QA;
- correção;
- regressão.

A limpeza final pertence ao SOP-012.

PARTE XI — ERROS, EMPTY STATES E CASOS NÃO COBERTOS

44. Empty states

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe teste sistemático universal de todos os empty states.

Testar quando

- módulo depende fortemente de estado vazio;
- alteração tocou essa área;
- problema foi reportado.

---

45. Estados de erro

Não existe matriz exaustiva universal.

Testar de forma dirigida quando:

- fluxo sofreu alteração;
- existe risco;
- bug anterior;
- integração externa relevante.

PARTE XII — USO DO CODEX NO QA

46. Passo 30 — Solicitar rodada ampla ao Codex quando útil

O Dev pode pedir ao Codex:

- testar páginas;
- navegar;
- clicar;
- identificar erros;
- verificar build;
- executar typecheck;
- inspecionar console;
- relatar inconsistências.

---

47. Não delegar decisão final

O relatório do Codex é input.

O Dev ainda precisa:

- interpretar;
- reproduzir;
- confirmar;
- priorizar.

---

48. Build e typecheck

Podem ser executados quando:

- a alteração é técnica;
- o projeto usa esses checks;
- há risco relevante.

AUSÊNCIA

Não são gate universal obrigatório para todo QA.

PARTE XIII — REGISTRAR PROBLEMAS

49. Passo 31 — Registrar cada falha reproduzível

Para cada problema:

- página/módulo;
- ação;
- sintoma;
- ambiente;
- dispositivo;
- evidência;
- impacto.

---

50. Classificar por camada

Quando possível:

- visual;
- responsivo;
- navegação;
- frontend;
- backend;
- banco;
- autenticação;
- configuração;
- conteúdo.

Objetivo

Facilitar encaminhamento ao SOP-009.

PARTE XIV — CORREÇÃO E REVALIDAÇÃO

51. Passo 32 — Corrigir problemas localizados

Quando a causa é simples/óbvia e a correção segura:

- corrigir;
- reexecutar o mesmo teste.

---

52. Passo 33 — Encaminhar problema não trivial

Se exige investigação:

> usar SOP-009 — Diagnosticar e Corrigir Bug por Camada.

---

53. Passo 34 — Revalidar exatamente o fluxo que falhou

Depois da correção:

- repetir ação;
- confirmar desaparecimento do sintoma;
- confirmar resultado esperado.

PARTE XV — REGRESSÃO

54. Passo 35 — Revalidar áreas impactadas

Depois de corrigir ou alterar:

- testar área corrigida;
- testar componentes compartilhados relacionados;
- testar fluxo conectado.

---

55. Regressão visual

Quando alteração visual:

- desktop;
- notebook;
- mobile;
- áreas que reutilizam componente.

---

56. Regressão funcional

Quando alteração funcional:

- ação principal;
- persistência;
- Admin;
- integrações relacionadas.

---

57. Regressão de perfil

Quando alteração de autorização/configuração:

- Webstar;
- Equipe;
- usuário sem papel quando aplicável.

Auditoria profunda pertence à camada de segurança.

PARTE XVI — REVISÃO HUMANA ADICIONAL

58. Revisão adicional em paralelo

O fluxo atual reconhece valor em outras pessoas revisarem o site a partir do momento em que o Dev sinaliza uma versão ou área revisável. Sempre que possível, essa revisão acontece **em paralelo** à própria revisão do Dev, e não apenas depois de ele encerrar toda a sua validação.

Razão

Quem passa muitas horas na mesma interface pode deixar detalhes passarem; perspectivas simultâneas ajudam a encontrar problemas mais cedo.

Limite

Não existe cadeia universal rígida de revisores formalizada neste SOP.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE XVII — PROCESSO SITES WSS / MARCO 8

59. Relação com o Marco 8

No Roadmap Sites WSS, o Marco 8 — Revisão Geral e Homologação inclui:

- revisão visual;
- revisão responsiva;
- revisão de textos;
- links;
- pontos de captação;
- Admin;
- Blog;
- SEO básico;
- ajustes finais;
- revisão do cliente quando aplicável.

Este SOP fornece o método técnico de QA que sustenta parte dessa homologação.

---

60. Processo ≠ SOP

O Marco 8 organiza:

> o que precisa estar validado.

O SOP-008 organiza:

> como o Dev executa o QA visual, responsivo e funcional.

Não substituir:

- checklist do Flow;
- evidências formais;
- aprovação do cliente;
- decisão de homologação.

PARTE XVIII — STOP CONDITION

61. Condição de parada

O SOP está concluído quando:

> **as páginas relevantes foram percorridas nos cenários principais de desktop, notebook e mobile; navegação, CTAs e fluxos essenciais foram testados; módulos operacionais aplicáveis foram verificados; problemas identificados foram corrigidos ou explicitamente encaminhados; e as áreas afetadas passaram por revalidação/regressão proporcional.**

---

62. Checklist mínimo

Antes de encerrar:

- [ ] versão/ambiente corretos;
- [ ] páginas principais percorridas;
- [ ] visual revisado;
- [ ] desktop validado;
- [ ] notebook validado;
- [ ] mobile físico validado;
- [ ] navegação testada;
- [ ] CTAs principais testados;
- [ ] formulários/Leads testados;
- [ ] Chat/WhatsApp testados quando aplicáveis;
- [ ] Chatbot revalidado quando aplicável;
- [ ] Admin percorrido;
- [ ] módulos aplicáveis verificados;
- [ ] perfis testados funcionalmente quando aplicável;
- [ ] problemas registrados;
- [ ] bugs complexos encaminhados ao SOP-009;
- [ ] correções revalidadas;
- [ ] regressão dirigida executada.

---

63. O que NÃO é necessário para encerrar

Não exigir universalmente:

- suíte automatizada completa;
- cobertura percentual;
- matriz ampla de viewports;

- teste de todo empty state;
- teste de toda falha externa;
- auditoria profunda de segurança;
- auditoria completa de SEO/AEO/GEO;
- publicação final.

PARTE XIX — CONTINGÊNCIAS

64. Visual correto em desktop e quebrado em mobile

Corrigir a versão mobile sem alterar desktop validado, quando possível.

Depois:

- revalidar mobile;
- revalidar desktop para regressão.

---

65. CTA termina em 404

1. reproduzir;
2. conferir URL;
3. comparar com rota real;
4. corrigir path;
5. testar novamente.

---

66. Formulário mostra sucesso mas Lead não aparece

Considerar fluxo incompleto.

Encaminhar ao SOP-009.

---

67. Chat abre mas conversa não chega ao Admin

Considerar falha funcional.

Não aprovar a capacidade.

---

68. Admin abre para Webstar, mas não para Equipe

Verificar se:

- comportamento é esperado;
- capability é restrita;
- existe erro real.

Não ampliar permissão automaticamente.

---

69. Efeito quebra somente em mobile

Priorizar adaptar; se necessário, simplificar ou substituir. Remover apenas quando a solução continuar inviável, instável ou prejudicial.

Não forçar efeito desktop em mobile.

---

70. Codex aponta problema que o Dev não reproduz

Não alterar automaticamente.

Reproduzir/confirmar estado atual antes.

PARTE XX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

71. Não inventar

O AS-IS não possui:

- suíte automatizada universal;
- cobertura mínima obrigatória;
- matriz fixa de browsers;
- matriz fixa de viewports;
- teste sistemático universal de empty states;
- teste universal de todos os 404;
- checklist universal de erro por formulário;
- build/typecheck obrigatório em toda rodada;
- gate numérico de performance;
- auditoria WCAG formal obrigatória;
- aprovação automática pelo Codex.

PARTE XXI — NORMA TÉCNICA VIGENTE

72. Normas complementam o QA

Quando norma vigente exigir, por exemplo:

- SSR;
- HTML inicial;
- 404 real;
- metadados;
- segurança;
- noindex;
- acessibilidade específica;

essas verificações devem ser aplicadas na camada correta.

Para SEO/AEO/GEO, a autoridade vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

O recorte técnico pré-publicação possui checklist normativo próprio:

- `WSS_SEO_AEO_GEO_Checklist_PrePublicacao_v0.3`.

A ausência AS-IS de teste sistemático de toda classe de 404 não torna **404 real** opcional onde a norma o exige.

Este SOP não incorpora nem substitui o checklist SEO/AEO/GEO completo.

> **Importante**
>
> Não reclassificar uma norma atual como prática histórica do Dev.

PARTE XXII — ANTI-PATTERNS

73. Validar só a Home

Páginas internas podem conter falhas próprias.

74. Validar só desktop

Mobile real é parte do AS-IS.

75. Validar formulário apenas visualmente

É necessário conferir persistência no Admin.

76. Validar Chat apenas pelo widget

É necessário conferir conversa/Admin.

77. Confiar em primeira saída do Codex

Resultado automatizado deve ser interpretado.

78. Corrigir sem revalidar

Fix não é conclusão.

79. Testar só área corrigida quando componente é compartilhado

Pode haver regressão em outras páginas.

PARTE XXIII — OUTPUT

80. Saída esperada

Ao final:

Site público

- páginas navegadas;
- visual coerente;
- principais cenários responsivos validados;
- CTAs/links funcionais.

Captação

- formulários testados;
- Leads chegando ao Admin;
- Chat/WhatsApp funcionando quando aplicáveis.

Operação

- Admin acessível;
- módulos aplicáveis testados;
- perfis verificados funcionalmente.

Controle

- problemas corrigidos ou encaminhados;
- regressão dirigida concluída;
- projeto pronto para próxima camada de validação.

PARTE XXIV — PRÓXIMO PASSO

81. Encaminhamento

Depois do SOP-008:

- bugs complexos → **SOP-009 — Diagnosticar e Corrigir Bug por Camada**;
- SEO/AEO/GEO → **SOP-010 — Tratar Auditoria SEO, AEO e GEO**;
- feedback → **SOP-011 — Processar Rodada de Feedback via `ajustes.md`**;
- publicação → **SOP-012 — Preparar, Publicar, Validar Produção e Entregar Admin**.

---

82. Referências

Manual proprietário

- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_03_Construcao_Visual_v1.1_FINAL.md`;
- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`;
- `WSS_Manual_07_Publicacao_Entrega_Pos_Publicacao_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`.

Evidências técnicas complementares

- `WSS_VAL_004_Troubleshooting_Estudos_de_Caso_v0.1.md`;
- `WSS_VAL_003_Auditorias_SEO_AEO_GEO_Seguranca_v0.1.md`.

Contexto de Processo

- `Roadmap_Sites_WSS_Modelo_Padrao_v0.2_FINAL.md`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-009
Diagnosticar e Corrigir Bug por Camada

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-009

Diagnosticar e Corrigir Bug por Camada

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 6 — Testes, QA, Debugging, Segurança e SEO
MANUAIS RELACIONADOS	Manual 4 — Projeto-Base, Admin e Funcionalidades; Manual 5 — Desenvolvimento com Codex, Lovable e GitHub; Manual 7 — Publicação, Entrega e Pós-Publicação; Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais
ESCOPO	AS-IS — diagnóstico pragmático e correção localizada de bugs por camada

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 7

2. Gatilho — quando usar 7

3. Quando NÃO usar 7

4. Papéis 8

5. Pré-condições 9

6. Inputs 10

PARTE I — COMEÇAR PELO SINTOMA 11

7. Passo 1 — Reproduzir no navegador 12

8. Passo 2 — Registrar o sintoma observável 12

9. Passo 3 — Confirmar ambiente 12

10. Passo 4 — Confirmar se o problema ainda existe 13

PARTE II — LOCALIZAR A CAMADA PROVÁVEL 14

11. Passo 5 — Classificar o sintoma por camada 15

12. Bug visual/layout 15

13. Bug responsivo 15

14. Bug de rota/navegação 16

15. Bug frontend/lógica 16

16. Bug de configuração/env 17

17. Bug backend/banco 17

18. Bug de autenticação/permissão 17

19. Bug de integração externa 18

20. Bug de build/produção 18

PARTE III — COLETAR EVIDÊNCIA 19

21. Passo 6 — Abrir console/DevTools quando apropriado 20

22. Passo 7 — Ler erro antes de reescrever 20

23. Passo 8 — Conferir Network quando necessário 20

24. Passo 9 — Conferir código/configuração relacionada 21

25. Passo 10 — Comparar com versão funcional quando útil 21

PARTE IV — FORMULAR HIPÓTESE **22**

26. Passo 11 — Formular hipótese simples e testável 23

27. Passo 12 — Ligar hipótese à evidência 23

28. Passo 13 — Evitar múltiplas alterações simultâneas 23

PARTE V — ESCOLHER A FERRAMENTA PELA CAMADA **24**

29. Frontend/visual → Codex/Claude 25

30. Backend/banco/auth/configuração — condução pós-criação 25

31. Problema difícil → investigação paralela 25

32. DECISÃO HUMANA NECESSÁRIA 26

PARTE VI — CORRIGIR **27**

33. Passo 14 — Aplicar correção ligada à hipótese 28

34. Não reescrever o módulo inteiro por padrão 28

35. Passo 15 — Preservar áreas não relacionadas 28

PARTE VII — REPRODUZIR NOVAMENTE **29**

36. Passo 16 — Refazer exatamente o fluxo 30

37. Passo 17 — Confirmar restauração 30

38. Causa suficientemente confirmada 30

PARTE VIII — REGRESSÃO APÓS A CORREÇÃO **31**

39. Passo 18 — Revalidar área relacionada 32

40. Regressão proporcional 32

PARTE IX — CASO REAL — MIRELA ALBUQUERQUE **33**

41. Sintoma 34

42. Evidência 34

43. Camada correta 34

44. Correção 34

45. Validação 34

46. Lição transferível 34

47. O que NÃO generalizar 35

PARTE X — CASO REAL — DANIEL P / CTA 404 **36**

48. Sintoma 37

49. Investigação 37

50. Causa 37
51. Correção 37
52. Validação 37
53. Lição 37

PARTE XI — CASO REAL — GUILHERME MENDONÇA / PRODUÇÃO 500 **38**
54. Sintoma 39
55. Camada correta 39
56. Investigação 39
57. Causa observada 39
58. Lição 39
59. Smoke não vira regra universal 39

PARTE XII — BUGS RELACIONADOS A AUDITORIA **41**
60. Relatório pode estar errado/desatualizado 42
61. Se não reproduzir 42

PARTE XIII — BUGS EM LEADS/FORMULÁRIOS **43**
62. Sintoma — sucesso visual sem Lead 44
63. Camadas a investigar 44
64. Critério de correção 44

PARTE XIV — BUGS EM CHAT/CHATBOT **45**
65. Chat abre, conversa não aparece 46
66. Chatbot responde, mas Lead não é criado 46
67. Handoff não silencia IA 46

PARTE XV — BUGS DE AUTORIZAÇÃO **47**
68. Perfil vê módulo indevido 48
69. Escalonamento de privilégio 48

PARTE XVI — BUGS VISUAIS / RESPONSIVOS **49**
70. Bug só em mobile 50
71. Preservar desktop validado 50
72. Efeito instável 50

PARTE XVII — BUGS DE CONTEXTO/IA **51**
73. IA altera área fora do escopo 52
74. Correção operacional 52

PARTE XVIII — STOP CONDITION 53

75. Condição de parada 54

76. Checklist mínimo 54

77. O que NÃO é necessário para encerrar 54

PARTE XIX — QUANDO NÃO ENCERRAR 55

78. O erro desapareceu sem saber por quê 56

79. A correção criou regressão 56

80. A correção exige mudança estrutural não aprovada 56

81. Produção continua quebrada 56

PARTE XX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 57

82. Não inventar 58

PARTE XXI — NORMA TÉCNICA VIGENTE 59

83. Normas podem aumentar a exigência 60

PARTE XXII — ANTI-PATTERNS 61

84. Corrigir sem reproduzir 62

85. Reescrever módulo antes de verificar configuração 62

86. Corrigir a camada errada 62

87. Fazer várias mudanças ao mesmo tempo 62

88. Esconder problema de autorização apenas na UI 62

89. Aceitar hipótese da IA sem evidência 62

90. Corrigir e não revalidar 62

PARTE XXIII — OUTPUT 63

91. Saída esperada 64

PARTE XXIV — PRÓXIMO PASSO 65

92. Encaminhamento 66

93. Referências 66

1. Finalidade

Diagnosticar um problema reproduzível sem alterar aleatoriamente o projeto, identificando a camada provável, coletando evidência proporcional e aplicando uma correção ligada à hipótese observada.

Fluxo principal:

```
sintoma
→ reproduzir
→ localizar camada
→ coletar evidência
→ formular hipótese
→ corrigir
→ reproduzir novamente
→ revalidar áreas relacionadas
→ encerrar
```

Este SOP organiza o debugging AS-IS da WSS.

Ele **não cria um método formal universal de root cause analysis**.

---

2. Gatilho — quando usar

Usar quando:

- QA identificou falha não trivial;
- cliente/equipe reportou bug;
- módulo deixou de funcionar;
- rota quebra;
- comportamento muda após alteração;
- console/runtime apresenta erro;
- ambiente local, preview ou produção divergem;
- uma integração aparentemente funcional deixa de responder;
- regressão aparece após implementação.

---

3. Quando NÃO usar

Não usar como substituto de:

- QA completo → SOP-008;
- auditoria SEO/AEO/GEO → SOP-010;
- tratamento de feedback amplo → SOP-011;
- publicação/contingência de produção → SOP-012;

- portabilidade Base → legado → SOP-007.

> **Observação**
>
> Um bug pode surgir durante qualquer desses procedimentos e então ser encaminhado a este SOP.

---

4. Papéis

4.1 Desenvolvedor

Responsável por:

- reproduzir;
- interpretar o sintoma;
- escolher a camada provável;
- decidir qual ferramenta usar;
- avaliar hipótese;
- validar correção;
- limitar alteração.

4.2 Codex

Usado com frequência para:

- investigação;
- leitura de código;
- frontend;
- lógica;
- comparação;
- reprodução técnica;
- testes;
- análise de build.

4.3 Claude

Usado principalmente quando o bug envolve:

- visual;
- layout;
- efeitos;
- responsividade;
- acabamento.

4.4 Lovable / Lovable Cloud

Usado com frequência quando a causa provável envolve:

- backend;
- banco;
- autenticação;
- Supabase;
- configuração integrada;
- variáveis/ambiente da plataforma.

4.5 Humano responsável

Necessário quando a correção deixa de ser localizada e passa a envolver:

- mudança de escopo;
- risco de dados;
- arquitetura ampla;
- permissão sensível;
- decisão de rollback;
- produção comprometida.

---

5. Pré-condições

Antes de alterar:

1. projeto correto identificado;
2. versão/ambiente conhecidos;
3. sintoma descrito;
4. acesso às ferramentas necessárias;
5. capacidade de reproduzir ou coletar evidência mínima.

Se não houver reprodução

Não corrigir por suposição sem evidência suficiente.

Pode ser necessário:

- pedir passos;
- confirmar ambiente;
- obter screenshot/erro;
- conferir versão.

---

6. Inputs

Conforme o bug:

- descrição do sintoma;
- URL/rota;
- página/módulo;
- ambiente;
- dispositivo;
- horário aproximado quando relevante;
- console;
- network/request;
- erro de runtime;
- screenshot;
- versão/commit atual;
- comportamento esperado;
- comportamento observado.

PARTE I — COMEÇAR PELO SINTOMA

7. Passo 1 — Reproduzir no navegador

Tentar executar exatamente o fluxo relatado.

Exemplos:

- clicar CTA;
- abrir Blog;
- enviar formulário;
- autenticar;
- abrir Admin;
- carregar rota;
- iniciar Chat;
- publicar conteúdo;
- navegar em mobile.

Objetivo

Confirmar que existe um bug real no estado atual.

---

8. Passo 2 — Registrar o sintoma observável

Descrever sem antecipar causa.

Bom exemplo

> Ao clicar no CTA “Saiba mais”, a aplicação navega para uma rota 404.

Mau exemplo

> O React Router está quebrado.

O primeiro é evidência.

O segundo já é hipótese.

---

9. Passo 3 — Confirmar ambiente

Identificar:

- local;

- Lovable preview;
- domínio temporário;
- produção;
- Admin;
- ambiente específico.

> **Importante**
>
> Um erro pode existir apenas em um ambiente.

---

10. Passo 4 — Confirmar se o problema ainda existe

Quando o relato/auditoria é antigo:

- testar a versão atual;
- não assumir que permanece válido.

Se não reproduzir:

- classificar como não reproduzido;
- não alterar automaticamente.

PARTE II — LOCALIZAR A CAMADA PROVÁVEL

11. Passo 5 — Classificar o sintoma por camada

Categorias práticas:

- visual/layout;
- responsividade;
- rota/navegação;
- frontend/lógica;
- runtime/browser;
- configuração/env;
- backend;
- banco;
- autenticação/permissão;
- integração externa;
- build/produção;
- conteúdo/dado.

---

12. Bug visual/layout

Sinais:

- desalinhamento;
- seção quebrada;
- componente sobreposto;
- tipografia errada;
- efeito incorreto;
- elemento desaparece.

Ferramentas recorrentes:

- navegador;
- Codex;
- Claude.

---

13. Bug responsivo

Sinais:

- desktop correto/mobile quebrado;
- overflow;
- scroll horizontal;

- crop inadequado;
- menu ruim;
- efeito instável.

Ferramentas:

- navegador/mobile;
- Codex;
- Claude.

---

14. Bug de rota/navegação

Sinais:

- CTA leva a 404;
- link não navega;
- rota incorreta;
- refresh falha.

Primeiro verificar:

- path;
- tabela real de rotas;
- destino configurado.

---

15. Bug frontend/lógica

Sinais:

- botão não executa;
- estado não atualiza;
- formulário não muda;
- interação falha;
- componente trava.

Ferramentas:

- navegador;
- console;
- Codex.

---

16. Bug de configuração/env

Sinais:

- módulo inteiro funcionava e parou após mudança de ambiente;
- key ausente;
- URL ausente;
- integração desconectada;
- runtime reclama de variável.

Primeiro verificar:

- console;
- env;
- keys;
- conexão;
- configuração de serviço.

---

17. Bug backend/banco

Sinais:

- frontend mostra sucesso mas dado não persiste;
- Lead não aparece;
- query falha;
- CRUD não funciona;
- endpoint retorna erro.

Ferramentas recorrentes:

- Lovable;
- Lovable Cloud;
- Supabase;
- Codex para inspeção quando necessário.

---

18. Bug de autenticação/permissão

Sinais:

- login falha;
- usuário vê área indevida;
- perfil correto perde acesso;
- autorização diverge da UI.

Cuidado

Ocultar botão não prova correção de autorização.

Problemas de segurança podem exigir validação técnica própria.

---

19. Bug de integração externa

Sinais:

- WhatsApp/destino incorreto;
- API não responde;
- gateway falha;
- serviço conectado não funciona.

Primeiro distinguir:

- configuração local;
- payload;
- credencial;
- disponibilidade externa.

---

20. Bug de build/produção

Sinais:

- dev funciona;
- produção/artefato falha;
- rota publicada retorna 500;
- comportamento aparece apenas após build.

Primeiro investigar:

- build real;
- bundler;
- runtime de produção;
- bindings;
- artefato gerado.

PARTE III — COLETAR EVIDÊNCIA

21. Passo 6 — Abrir console/DevTools quando apropriado

Usar principalmente quando há indício de:

- frontend;
- runtime;
- env;
- key;
- request;
- integração;
- erro JS.

---

22. Passo 7 — Ler erro antes de reescrever

Registrar:

- mensagem;
- stack;
- request;
- status;
- variável;
- recurso faltante.

> **Regra**
>
> Erro explícito deve orientar a próxima hipótese.

---

23. Passo 8 — Conferir Network quando necessário

Quando o bug envolve comunicação:

- request sai?
- status?
- payload?
- resposta?
- autenticação?
- endpoint?

Não é obrigatório em todo bug.

---

24. Passo 9 — Conferir código/configuração relacionada

Depois de localizar a camada:

- abrir apenas arquivos relevantes;
- evitar leitura indiscriminada do projeto inteiro;
- comparar comportamento esperado e implementação atual.

---

25. Passo 10 — Comparar com versão funcional quando útil

Se o bug é regressão:

- comparar antes/depois;
- identificar mudança relacionada;
- não desfazer alterações não relacionadas.

PARTE IV — FORMULAR HIPÓTESE

26. Passo 11 — Formular hipótese simples e testável

Exemplo:

> O Blog falha porque a URL/key do Supabase não está disponível neste ambiente.

Não:

> O Blog inteiro precisa ser refeito.

---

27. Passo 12 — Ligar hipótese à evidência

Perguntar:

- qual erro sustenta isso?
- qual comportamento é compatível?
- existe mudança de ambiente?
- a camada afetada depende disso?

---

28. Passo 13 — Evitar múltiplas alterações simultâneas

Para testar hipótese:

- alterar o mínimo necessário;
- evitar “limpar” várias coisas de uma vez.

Objetivo

Saber qual correção restaurou o comportamento.

PARTE V — ESCOLHER A FERRAMENTA PELA CAMADA

29. Frontend/visual → Codex/Claude

Usar conforme natureza:

Codex

- lógica;
- código;
- investigação;
- estrutura;
- comportamento.

Claude

- visual;
- layout;
- efeito;
- responsividade;
- refinamento.

---

30. Backend/banco/auth/configuração — condução pós-criação

A criação inicial da estrutura herdada de backend/banco/auth permanece no Lovable durante o remix da Base Admin. Este passo trata apenas de **bugs e intervenções pós-criação**.

Nessa fase, o Codex pode ser a primeira camada de investigação/condução. Quando a integração da plataforma facilita ou é necessária, o Codex pode operar/orquestrar Lovable/Lovable Cloud, ou o Dev pode usá-los diretamente, para tratar:

- banco;
- Supabase;
- auth;
- configuração;
- backend.

---

31. Problema difícil → investigação paralela

O Dev pode usar:

- Lovable;
- Codex;

em paralelo.

> **Regra**
>
> Comparar hipóteses.

Não aplicar duas correções conflitantes simultaneamente.

---

32. DECISÃO HUMANA NECESSÁRIA

Quando o bug exige:

- migration destrutiva;
- alteração de dados reais;
- mudança ampla de arquitetura;
- permissão sensível;
- remoção de capacidade;
- feature nova;
- rollback relevante.

PARTE VI — CORRIGIR

33. Passo 14 — Aplicar correção ligada à hipótese

A correção deve ser:

- localizada;
- proporcional;
- compatível com o projeto;
- limitada ao problema.

---

34. Não reescrever o módulo inteiro por padrão

Antes de reescrever:

- confirmar que configuração não é a causa;
- confirmar que dependência não é a causa;
- confirmar que path não é a causa;
- confirmar que ambiente não é a causa.

---

35. Passo 15 — Preservar áreas não relacionadas

Explicitamente proteger:

- páginas já validadas;
- outros breakpoints;
- Admin quando não relacionado;
- backend quando não relacionado;
- dados existentes.

PARTE VII — REPRODUZIR NOVAMENTE

36. Passo 16 — Refazer exatamente o fluxo

Depois da correção:

- repetir os mesmos passos;
- no mesmo ambiente;
- com condição equivalente.

---

37. Passo 17 — Confirmar restauração

O comportamento deve:

- voltar ao esperado;
- deixar de reproduzir o erro.

No AS-IS, isso é a principal confirmação prática da hipótese.

---

38. Causa suficientemente confirmada

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe método formal universal de causa raiz.

O Dev considera a causa suficientemente confirmada quando:

1. a hipótese orientou a correção;
2. a correção restaurou o comportamento;
3. o erro deixou de reproduzir.

PARTE VIII — REGRESSÃO APÓS A CORREÇÃO

39. Passo 18 — Revalidar área relacionada

Conforme a correção:

Visual

- desktop;
- mobile;
- componente compartilhado.

Funcional

- fluxo completo;
- entrada;
- saída/persistência.

Admin

- módulo alterado;
- ação relacionada.

Auth

- perfis relevantes.

Base/legado

- capacidade preservada;
- dependências.

---

40. Regressão proporcional

Não existe checklist universal completo.

Testar:

- relações óbvias;
- dependências diretas;
- áreas compartilhadas.

PARTE IX — CASO REAL — MIRELA ALBUQUERQUE

41. Sintoma

Após retorno/mudança no ambiente Lovable:

- Blog deixou de funcionar corretamente.

---

42. Evidência

O console indicou problema relacionado a:

- Supabase;
- configuração;
- key.

---

43. Camada correta

A causa estava em:

- ambiente;
- configuração;
- conexão Supabase/Lovable.

Não no visual do Blog.

---

44. Correção

A configuração foi corrigida.

---

45. Validação

O Blog voltou a funcionar.

---

46. Lição transferível

> **Quando um módulo inteiro quebra após mudança de ambiente/conexão, verificar env, keys e integrações antes de reescrever código funcional.**

47. O que NÃO generalizar

Não afirmar:

> todo bug de Blog é Supabase.

Generalizar apenas o padrão:

```
sintoma amplo
→ localizar camada
→ confirmar configuração/ambiente
→ corrigir causa real
```

PARTE X — CASO REAL — DANIEL P / CTA 404

48. Sintoma

Clique em CTA terminava em 404.

---

49. Investigação

Comparação:

- path configurado;
- rota real do projeto.

---

50. Causa

CTA usava:

`/sobre-mim`

Rota real:

`/sobre`

---

51. Correção

Ajustar apenas o path.

---

52. Validação

Reexecutar clique.

---

53. Lição

Problema localizado de rota não exige reconstruir componente/página.

PARTE XI — CASO REAL — GUILHERME MENDONÇA / PRODUÇÃO 500

54. Sintoma

- modo dev saudável;
- artefato de produção falhava;
- rotas retornavam HTTP 500.

55. Camada correta

Build/bundler/runtime de produção.

56. Investigação

O projeto passou a:

- gerar build real;
- subir worker;
- testar rotas;
- separar falta de binding local de falha real do bundle.

57. Causa observada

Problema de chunking/bundling em faixa problemática do stack.

58. Lição

> **Dev funcionando não prova produção saudável.**

59. Smoke não vira regra universal

O projeto possui smoke específico.

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe evidência de que todo site execute o mesmo smoke.

O procedimento permanece incorporado ao Manual 6.

PARTE XII — BUGS RELACIONADOS A AUDITORIA

60. Relatório pode estar errado/desatualizado

Quando bug vem de:

- Codex;
- auditoria;
- SEOmator;
- relatório antigo;

primeiro reproduzir.

---

61. Se não reproduzir

Não corrigir automaticamente.

Classificar:

- ainda presente;
- parcial;
- resolvido;
- não reproduzido.

O tratamento completo de auditoria pertence ao SOP-010.

PARTE XIII — BUGS EM LEADS/ FORMULÁRIOS

62. Sintoma — sucesso visual sem Lead

Frontend informa sucesso, mas Admin não recebe registro.

63. Camadas a investigar

- frontend;
- request;
- backend;
- banco;
- autenticação;
- payload.

64. Critério de correção

O fluxo precisa voltar a funcionar ponta a ponta:

```
envio
→ sucesso
→ persistência
→ Lead visível no Admin
```

PARTE XIV — BUGS EM CHAT/CHATBOT

65. Chat abre, conversa não aparece

Investigar:

- persistência;
- backend;
- conversa;
- consulta Admin.

---

66. Chatbot responde, mas Lead não é criado

Capacidade incompleta.

Investigar vínculo:

```
conversation
↔ lead
```

---

67. Handoff não silencia IA

Tratar como bug funcional relevante.

Usar também SOP-006 para revalidar o fluxo completo.

PARTE XV — BUGS DE AUTORIZAÇÃO

68. Perfil vê módulo indevido

Não assumir que esconder UI resolve.

Investigar:

- capability;
- role;
- autorização server-side;
- RLS quando aplicável.

---

69. Escalonamento de privilégio

Se existe suspeita de privilégio indevido:

- interromper correção superficial;
- executar validação técnica proporcional;
- não aumentar permissão para “fazer funcionar”.

PARTE XVI — BUGS VISUAIS / RESPONSIVOS

70. Bug só em mobile

Investigar:

- breakpoint;
- largura;
- overflow;
- efeito;
- imagem;
- componente.

---

71. Preservar desktop validado

Se a falha é mobile:

- limitar correção ao mobile quando possível;
- revalidar desktop depois.

---

72. Efeito instável

Pode ser necessário:

- adaptar;
- simplificar;
- remover.

A fidelidade à referência não justifica manter bug.

PARTE XVII — BUGS DE CONTEXTO/IA

73. IA altera área fora do escopo

Sintomas:

- página não solicitada muda;
- desktop muda em tarefa mobile;
- requisito novo aparece.

---

74. Correção operacional

1. interromper;
2. reforçar contexto;
3. reduzir escopo;
4. restaurar área afetada;
5. retomar em partes menores.

Esse tipo de problema também é tratado no Manual 2/5 e no Manual 8.

PARTE XVIII — STOP CONDITION

75. Condição de parada

O SOP está concluído quando:

> **o bug foi reproduzido, sua camada provável foi sustentada por evidência suficiente, uma correção proporcional ligada à hipótese foi aplicada, o comportamento esperado foi restaurado, o erro deixou de reproduzir e as áreas diretamente relacionadas passaram por regressão proporcional.**

---

76. Checklist mínimo

Antes de encerrar:

- [ ] bug reproduzido;
- [ ] ambiente confirmado;
- [ ] sintoma descrito;
- [ ] camada provável identificada;
- [ ] evidência coletada;
- [ ] hipótese formulada;
- [ ] ferramenta escolhida pela camada;
- [ ] correção limitada ao problema;
- [ ] mesmo fluxo reexecutado;
- [ ] comportamento restaurado;
- [ ] erro não reproduz mais;
- [ ] regressão relacionada executada;
- [ ] pendência/risco residual registrado quando existir.

---

77. O que NÃO é necessário para encerrar

Não exigir universalmente:

- documento formal de RCA;
- diagrama de causa;
- análise forense completa;
- prova matemática da causa;
- teste de todas as áreas do site;
- suite automatizada completa;
- post-mortem formal.

PARTE XIX — QUANDO NÃO ENCERRAR

78. O erro desapareceu sem saber por quê

Se a mudança não está ligada a hipótese clara e o erro simplesmente “sumiu”:

- aumentar cautela;
- revalidar;
- não declarar causa confirmada com excesso de certeza.

---

79. A correção criou regressão

Não encerrar.

Corrigir ou restaurar.

---

80. A correção exige mudança estrutural não aprovada

Não encerrar por execução unilateral.

Retornar à decisão humana.

---

81. Produção continua quebrada

Se falha é ampla:

- aplicar contingência/rollback conforme Manual 7/SOP-012;
- não prolongar debugging com produção amplamente indisponível.

PARTE XX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

82. Não inventar

O AS-IS não possui:

- método formal universal de root cause analysis;
- template obrigatório de ticket de bug;
- severidade numérica universal;
- SLA técnico universal dentro deste método;
- ferramenta única de debugging;
- ordem rígida de DevTools → código → backend;
- checklist universal de regressão;
- post-mortem obrigatório;
- smoke universal;
- procedimento único para todo bug.

PARTE XXI — NORMA TÉCNICA VIGENTE

83. Normas podem aumentar a exigência

Quando o bug toca uma área normativa:

- segurança;
- SSR;
- SEO;
- privacidade;
- autenticação;
- 404;
- dados;

a correção deve respeitar a norma aplicável.

Quando o bug tocar SEO, indexação ou áreas privadas, consultar:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

> **Regra**
>
> Não usar o AS-IS pragmático como justificativa para descumprir norma vigente.

PARTE XXII — ANTI-PATTERNS

84. Corrigir sem reproduzir

Risco de alterar problema inexistente ou já resolvido.

85. Reescrever módulo antes de verificar configuração

Caso Mirela demonstra o risco.

86. Corrigir a camada errada

Exemplos:

- CTA 404 → não é redesign;
- produção 500 → não é necessariamente componente;
- Supabase ausente → não é Blog visual.

87. Fazer várias mudanças ao mesmo tempo

Dificulta identificar o que resolveu.

88. Esconder problema de autorização apenas na UI

Não garante segurança.

89. Aceitar hipótese da IA sem evidência

IA é ferramenta de investigação, não prova.

90. Corrigir e não revalidar

Correção não comprovada não encerra bug.

PARTE XXIII — OUTPUT

91. Saída esperada

Ao final:

Diagnóstico

- sintoma reproduzido;
- camada identificada;
- evidência suficiente;
- hipótese registrada implicitamente ou explicitamente.

Correção

- mudança proporcional;
- escopo limitado;
- comportamento restaurado.

Validação

- erro não reproduz;
- fluxo esperado funciona;
- regressão relacionada conferida.

Controle

- riscos/pendências residuais conhecidos;
- decisão humana acionada quando necessária.

PARTE XXIV — PRÓXIMO PASSO

92. Encaminhamento

Depois do SOP-009:

- retornar ao **SOP-008** para QA/regressão quando o bug surgiu durante revisão;
- seguir para **SOP-010** quando o problema deriva de auditoria SEO/AEO/GEO;
- seguir para **SOP-011** se a correção pertence a rodada de feedback;
- seguir para **SOP-012** quando a correção ocorreu em publicação/produção.

---

93. Referências

Manual proprietário

- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`;
- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`;
- `WSS_Manual_07_Publicacao_Entrega_Pos_Publicacao_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`.

Evidência técnica complementar

- `WSS_VAL_004_Troubleshooting_Estudos_de_Caso_v0.1.md`.

Casos reais utilizados

- Mirela Albuquerque;
- Daniel P;
- Dr. Guilherme Mendonça.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-010
Tratar Auditoria SEO, AEO e GEO

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-010

Tratar Auditoria SEO, AEO e GEO

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 6 — Testes, QA, Debugging, Segurança e SEO
MANUAIS RELACIONADOS	Manual 3 — Construção Visual; Manual 5 — Desenvolvimento com Codex, Lovable e GitHub; Manual 7 — Publicação, Entrega e Pós-Publicação; Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais
ESCOPO	AS-IS — interpretação, correção e validação de auditorias SEO/AEO/GEO no fluxo atual WSS
CAMADA NORMATIVA RELACIONADA	WSSPadraoTecnicoOperacionalSEOAEOGEOv0.2FINAL

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 7

2. Gatilho — quando usar 7

3. Quando NÃO usar 7

4. Papéis 8

5. Pré-condições 9

6. Inputs 9

PARTE I — NÃO TRATAR A AUDITORIA COMO VERDADE AUTOMÁTICA 11

7. Princípio central 12

8. Passo 1 — Identificar origem e data do relatório 12

9. Passo 2 — Confirmar o estado atual 12

10. Passo 3 — Reproduzir cada achado relevante 13

PARTE II — CLASSIFICAR O ACHADO 14

11. Passo 4 — Classificar cada item 15

12. Registrar classificação antes de editar 15

13. Relatório baseado em build anterior 15

PARTE III — ENTENDER O QUE A FERRAMENTA ESTÁ ENXERGANDO 16

14. Passo 5 — Determinar a representação analisada 17

15. Caso real Daniel P 17

16. Passo 6 — Comparar UI × HTML inicial 17

17. Falso positivo × problema arquitetural 17

PARTE IV — SEO TÉCNICO AS-IS 19

18. Passo 7 — Revisar metadados quando o achado os envolve 20

19. Passo 8 — Revisar domínio/origem 20

20. Passo 9 — Revisar dados estruturados 20

21. Passo 10 — Revisar breadcrumbs e links internos 21

22. Passo 11 — Revisar sitemap 21

23. Passo 12 — Revisar robots.txt 21

24. Passo 13 — Revisar noindex/nofollow 21

PARTE V — SSR / HTML INICIAL **23**

25. Passo 14 — Confirmar conteúdo público no HTML inicial quando aplicável 24

26. NORMA TÉCNICA VIGENTE 24

27. AS-IS × NORMA 25

PARTE VI — BLOG **26**

28. Passo 15 — Revisar listagem do Blog 27

29. Passo 16 — Revisar artigo individual 27

30. Norma de Blog 27

PARTE VII — AEO / GEO NO AS-IS **28**

31. AEO/GEO nas auditorias atuais 29

32. Não universalizar checklist Daniel P 29

PARTE VIII — SEOMATOR **31**

33. Passo 17 — Ler o score como indicador, não como gate 32

34. Não criar threshold universal 32

35. Passo 18 — Decompor o relatório 32

PARTE IX — CONTEÚDO E VALIDAÇÃO HUMANA **33**

36. Passo 19 — Identificar itens que dependem do cliente 34

37. Não declarar revisão humana inexistente 34

38. Conteúdo YMYL/sensível 34

PARTE X — DEFINIR O DELTA REAL **35**

39. Passo 20 — Consolidar itens que precisam de alteração 36

40. Passo 21 — Ordenar correções por dependência 36

PARTE XI — IMPLEMENTAR **37**

41. Passo 22 — Corrigir somente o delta 38

42. Preservar o que já está correto 38

43. Mudança localizada 38

PARTE XII — VALIDAR A CORREÇÃO **39**

44. Passo 23 — Executar validação dirigida 40

45. Não transformar a lista em suíte universal 40

46. Passo 24 — Validar metadados 40

47. Passo 25 — Validar HTML inicial 40

48. Passo 26 — Validar sitemap/robots 41

49. Passo 27 — Validar links/breadcrumbs 41

PARTE XIII — REGRESSÃO 42

50. Passo 28 — Revalidar áreas relacionadas 43

51. Não quebrar o site para subir score 43

PARTE XIV — ACHADOS DE SEGURANÇA EM AUDITORIAS MISTAS 44

52. SEO/GEO pode trazer item de segurança 45

PARTE XV — STATUS DA AUDITORIA 46

53. Passo 29 — Consolidar resultado 47

54. Não usar “100%” sem critério sustentado 47

PARTE XVI — STOP CONDITION 48

55. Condição de parada 49

56. Checklist mínimo 49

57. O que NÃO é necessário para encerrar 49

PARTE XVII — CONTINGÊNCIAS 51

58. Auditoria diz que elemento está ausente, mas UI mostra 52

59. Relatório pede item já resolvido 52

60. Auditoria usa domínio preview 52

61. Score baixo por conteúdo não validado 52

62. Ferramenta sugere llms.txt 52

63. Crawler não vê conteúdo CSR 53

64. Correção SEO quebra componente/rota 53

PARTE XVIII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 54

65. Não inventar 55

PARTE XIX — NORMA TÉCNICA VIGENTE 56

66. Arquitetura SSR-first / híbrida 57

67. CSR permitido para interação 57

68. Blog normativo 57

69. Áreas privadas 58

PARTE XX — ANTI-PATTERNS 59

70. Executar relatório antigo sem conferir main 60
71. Corrigir H1 que já existe sem entender o crawler 60
72. Transformar score em meta absoluta 60
73. Universalizar checklist de um projeto 60
74. Inventar revisão clínica 60
75. Adicionar schema sem correspondência real 60
76. Melhorar SEO sacrificando arquitetura/UX 60

PARTE XXI — OUTPUT **61**
77. Saída esperada 62

PARTE XXII — PRÓXIMO PASSO **63**
78. Encaminhamento 64
79. Referências 64

1. Finalidade

Tratar uma auditoria de SEO, AEO ou GEO como **fonte de evidência a ser verificada**, e não como ordem automática de alteração.

Fluxo principal:

```
relatório / auditoria
→ confirmar versão analisada
→ reproduzir no estado atual
→ classificar achado
→ separar limitação da ferramenta de delta real
→ identificar camada
→ corrigir somente o necessário
→ validar tecnicamente
→ registrar pendências humanas/normativas
→ encerrar
```

Este SOP documenta o **AS-IS de tratamento de auditorias**.

Ele não cria um novo padrão SEO/AEO/GEO.

---

2. Gatilho — quando usar

Usar quando houver:

- relatório de SEOmator;
- auditoria produzida por Lovable;
- auditoria produzida por Codex;
- checklist interno;
- relatório SEO/AEO/GEO;
- achado técnico de indexação;
- pedido de correção baseado em build/versão anterior;
- revisão técnica antes de publicação.

---

3. Quando NÃO usar

Não usar como substituto de:

- construção visual;
- QA geral → SOP-008;
- debugging geral → SOP-009;
- segurança;

- publicação → SOP-012;
- definição de estratégia editorial;
- criação de nova norma SEO/AEO/GEO.

> **Regra**
>
> Se a auditoria revelar bug técnico localizado:

> usar SOP-009 para o diagnóstico detalhado.

---

4. Papéis

4.1 Desenvolvedor

Responsável por:

- interpretar o relatório;
- comparar com o estado atual;
- reproduzir achados;
- separar delta real de falso positivo/limitação;
- corrigir;
- validar;
- registrar dependências humanas.

4.2 Codex

Pode apoiar:

- inspeção do código;
- HTML SSR;
- build;
- links;
- schemas;
- metadados;
- sitemap;
- robots;
- validações dirigidas.

4.3 Lovable

Pode apoiar:

- análise;

- configuração;
- alterações integradas;
- backend/SSR quando aplicável.

4.4 SEOmator

É ferramenta externa de auditoria utilizada no AS-IS.

Seu resultado é:

> **input de investigação**

e não autoridade automática.

4.5 Cliente / WSS

Pode ser necessário para validar:

- conteúdo factual;
- revisão clínica;
- autoria;
- fontes;
- dados institucionais;
- decisões editoriais.

---

5. Pré-condições

Antes de alterar:

1. relatório/auditoria disponível;
2. projeto/versão atual acessível;
3. branch/estado atual conhecidos;
4. domínio/origem relevantes conhecidos quando aplicável;
5. capacidade de reproduzir o achado;
6. norma técnica aplicável identificada quando necessário.

---

6. Inputs

Podem incluir:

- relatório SEOmator;
- plano Lovable/Codex;
- checklist AEO/GEO;

- `robots.txt`;
- sitemap;
- HTML inicial;
- metadados;
- schemas;
- rotas;
- Blog;
- Search Console quando aplicável;
- domínio oficial;
- norma técnica vigente;
- versão/commit analisado.

PARTE I — NÃO TRATAR A AUDITORIA COMO VERDADE AUTOMÁTICA

7. Princípio central

No AS-IS observado:

```
auditoria
→ estado atual
→ classificação
→ delta real
→ correção
→ validação
```

Não:

```
auditoria
→ executar tudo cegamente
```

---

8. Passo 1 — Identificar origem e data do relatório

Registrar:

- ferramenta;
- projeto;
- versão/build;
- momento aproximado;
- ambiente analisado.

Objetivo

Saber se o relatório ainda representa o projeto atual.

---

9. Passo 2 — Confirmar o estado atual

Abrir:

- código atual;
- `main` ou estado vigente;
- preview/ambiente correto.

Regra

Antes de corrigir:

comparar o pedido com a implementação atual.

---

10. Passo 3 — Reproduzir cada achado relevante

Para cada item:

- verificar se existe;
- localizar evidência;
- confirmar impacto.

Não confiar apenas no texto do relatório.

PARTE II — CLASSIFICAR O ACHADO

11. Passo 4 — Classificar cada item

Usar as categorias AS-IS observadas:

A. Ainda presente

O problema existe no estado atual.

B. Parcialmente resolvido

Existe implementação, mas incompleta.

C. Já resolvido

O relatório descreve algo que já foi corrigido.

D. Não reproduzido

A ferramenta apontou algo que não é confirmável no estado atual.

---

12. Registrar classificação antes de editar

Essa classificação evita:

- retrabalho;
- regressão;
- edição baseada em histórico antigo.

---

13. Relatório baseado em build anterior

Caso conhecido:

- relatório pede correção;
- `main` já contém solução;
- alteração seria redundante.

Ação

Classificar como:

já resolvido

e não refazer.

PARTE III — ENTENDER O QUE A FERRAMENTA ESTÁ ENXERGANDO

14. Passo 5 — Determinar a representação analisada

Perguntar:

- crawler lê HTML inicial?
- executa JavaScript?
- analisa DOM após hidratação?
- usa URL preview ou domínio oficial?
- acessa conteúdo autenticado?
- recebeu resposta SSR?

---

15. Caso real Daniel P

SEOmator relatou, entre outros:

- H1 ausente;
- imagens ausentes;
- schema ausente.

A UI mostrava elementos que o crawler não estava enxergando da mesma forma.

Lição

> **UI visível no navegador não prova que o conteúdo está presente no HTML entregue ao crawler.**

---

16. Passo 6 — Comparar UI × HTML inicial

Quando houver contradição:

1. abrir página;
2. confirmar elemento visual;
3. inspecionar HTML inicial/SSR;
4. comparar o que o crawler recebe.

---

17. Falso positivo × problema arquitetural

Se a ferramenta não enxerga algo:

Pode ser

- limitação da ferramenta;
- configuração de crawl;
- ambiente incorreto;
- JavaScript client-side;
- problema real de renderização/indexabilidade.

Não decidir pela aparência

Investigar a camada real.

PARTE IV — SEO TÉCNICO AS-IS

18. Passo 7 — Revisar metadados quando o achado os envolve

Itens observados em auditorias reais:

- `title`;
- meta description;
- canonical;
- `og:title`;
- `og:description`;
- `og:url`;
- `og:image`;
- Twitter Cards.

---

19. Passo 8 — Revisar domínio/origem

Confirmar:

- domínio oficial;
- preview;
- URLs absolutas;
- canonical;
- sitemap;
- OG.

Problema conhecido

Canonical editorial antigo pode apontar para preview.

Não presumir que a origem atual está correta sem verificar.

---

20. Passo 9 — Revisar dados estruturados

Auditorias reais verificaram:

- `Physician`;
- `WebSite`;
- `Blog`;
- `BlogPosting`;
- `BreadcrumbList`;
- outros schemas aplicáveis.

> **Regra**
>
> Schema precisa corresponder ao conteúdo real.

Não adicionar tipo apenas para elevar score.

---

21. Passo 10 — Revisar breadcrumbs e links internos

Quando aplicável:

- breadcrumbs;
- navegação contextual;
- links internos no Blog;
- links entre conteúdos relacionados.

---

22. Passo 11 — Revisar sitemap

Verificar:

- existência;
- URLs esperadas;
- domínio correto;
- ausência de rotas indevidas;
- artigos publicados quando a arquitetura prevê.

---

23. Passo 12 — Revisar `robots.txt`

Verificar:

- regras atuais;
- sitemap declarado quando aplicável;
- áreas privadas;
- bloqueios não intencionais.

---

24. Passo 13 — Revisar `noindex/nofollow`

Quando houver rotas que não devem indexar:

- login;

- Admin;
- dashboards;
- rotas conceituais internas;
- outras áreas privadas.

> **Importante**
>
> A aplicabilidade depende da rota/projeto.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE V — SSR / HTML INICIAL

25. Passo 14 — Confirmar conteúdo público no HTML inicial quando aplicável

Para páginas públicas indexáveis, verificar se o conteúdo principal aparece no HTML entregue inicialmente.

Itens importantes:

- H1;
- conteúdo principal;
- links;
- metadados;
- schema quando aplicável.

---

26. NORMA TÉCNICA VIGENTE

A autoridade canônica atual para SEO, AEO e GEO é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

O antigo `WSS_Padrao_Tecnico_Arquitetura_SEO_v0.1` permanece apenas como baseline histórico/proveniência, não como autoridade autônoma vigente.

O padrão vigente mantém abordagem:

SSR-first / híbrida

para páginas públicas indexáveis.

Classificações normativas

Os requisitos do padrão podem ser classificados como:

- Obrigatória;
- Recomendada;
- Condicional;
- Experimental;
- Não adotada;
- Sem decisão.

A classificação deve ser preservada; recomendação não deve ser silenciosamente convertida em obrigação.

Regra normativa

Páginas públicas devem preservar, conforme a classificação aplicável:

- HTML completo por URL;
- conteúdo principal;

- headings;
- links;
- metadados;
- dados estruturados aplicáveis;
- status HTTP correto;

sem depender do cliente para materializar o conteúdo principal.

Também devem ser considerados, no recorte aplicável:

- staging/preview não indexável e fora do sitemap oficial;
- origem oficial de produção para canonical e demais URLs absolutas;
- Google Search Console configurado na publicação;
- Bing Webmaster Tools como recomendado;
- GA4 como recomendado, não obrigatório universal;
- `llms.txt` como não obrigatório;
- política padrão de crawlers de treinamento ainda sem decisão final;
- ferramentas pagas sem escolha definitiva.

---

27. AS-IS × NORMA

Não afirmar que todo projeto histórico já cumpre essa arquitetura.

> **Regra**

```
AS-IS
= o que o projeto realmente faz

NORMA
= o que a arquitetura vigente exige
```

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE VI — BLOG

28. Passo 15 — Revisar listagem do Blog

Quando o Blog entra na auditoria:

- conteúdo publicado;
- cards/títulos;
- links;
- metadados;
- canonical;
- HTML inicial;
- domínio.

---

29. Passo 16 — Revisar artigo individual

Verificar:

- slug;
- H1;
- corpo;
- metadados;
- canonical;
- Open Graph;
- schema;
- breadcrumbs;
- links internos;
- comportamento para slug inexistente.

---

30. Norma de Blog

A norma técnica vigente define Blog SSR para novos projetos públicos.

Isso não deve ser apresentado como se fosse prática histórica universal de todo legado.

PARTE VII — AEO / GEO NO AS-IS

31. AEO/GEO nas auditorias atuais

Podem envolver:

Visibilidade técnica

- HTML acessível;
- headings;
- structured data;
- robots;
- sitemap;
- crawlers;
- canonical.

Estrutura de conteúdo

- FAQ;
- resumo;
- TL;DR;
- headings interrogativos;
- entidades;
- variações semânticas;
- links internos.

Autoridade

- autoria/revisão;
- E-E-A-T;
- fontes;
- entidade/schema;
- `sameAs`.

---

32. Não universalizar checklist Daniel P

O projeto Daniel P possui checklist interno real com grupos:

- AEO;
- GEO;
- SEO;
- E-E-A-T.

Exemplos observados:

AEO

- FAQ;
- “Em resumo”;
- TL;DR;
- headings em forma de pergunta.

GEO

- entidades relacionadas;
- variações semânticas;
- “Quem deve ler/agir”;
- links internos.

SEO

- tamanho de título;
- meta description;
- palavra-chave;
- estrutura;
- contagem.

E-E-A-T

- revisor médico;
- última revisão.

> **Regra**
>
> Isso prova uma implementação real.

Não prova um checklist universal WSS.

PARTE VIII — SEOmator

33. Passo 17 — Ler o score como indicador, não como gate

Caso real Daniel P:

- auditoria GEO SEOmator;
- score 82/100.

Dimensões incluíam:

- AI Citability;
- Brand Authority;
- Platform Readiness.

---

34. Não criar threshold universal

AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Não existe score mínimo SEOmator universal aprovado para todo projeto WSS no AS-IS.

Heurística pessoal declarada pelo Dev — não normativa

O Dev busca, como referência pessoal, chegar à faixa verde; considera 90 um resultado bom e 95+ uma meta desejada quando viável. Isso não é gate WSS, não é requisito de publicação/entrega e não autoriza corrigir cegamente o relatório para elevar score.

---

35. Passo 18 — Decompor o relatório

Separar:

1. achado;
2. interpretação;
3. recomendação;
4. implementação comprovada.

Regra

Plano sugerido pela ferramenta não equivale a implementação.

PARTE IX — CONTEÚDO E VALIDAÇÃO HUMANA

36. Passo 19 — Identificar itens que dependem do cliente

Exemplos:

- revisão clínica;
- título/formação;
- autoria;
- fatos;
- fontes;
- data de revisão;
- informação institucional.

---

37. Não declarar revisão humana inexistente

Caso real:

- revisão clínica dependia do cliente;
- auditoria não declarou falsamente como concluída.

> **Regra**
>
> Automação técnica não substitui validação factual.

---

38. Conteúdo YMYL/sensível

Quando o conteúdo é médico ou factual sensível:

- não inventar;
- não completar lacunas;
- preservar decisão humana.

PARTE X — DEFINIR O DELTA REAL

39. Passo 20 — Consolidar itens que precisam de alteração

Depois da triagem:

Corrigir

- ainda presente;
- parcial.

Não tocar

- já resolvido;
- não reproduzido sem evidência.

Escalar

- conteúdo/decisão humana;
- mudança normativa/estratégica;
- escopo novo.

---

40. Passo 21 — Ordenar correções por dependência

Quando uma correção depende de outra:

- tratar suporte primeiro.

Exemplo:

- domínio/origem;
- depois canonical/OG absoluto.

PARTE XI — IMPLEMENTAR

41. Passo 22 — Corrigir somente o delta

Aplicar mudanças dirigidas.

Evitar:

- reescrever SEO inteiro;
- substituir implementação saudável;
- adicionar requisito só porque apareceu em relatório externo.

---

42. Preservar o que já está correto

Caso real Guilherme:

- canonical já resolvido → preservado;
- sitemap não tinha problema antigo → preservado.

---

43. Mudança localizada

Quando problema é:

- robots sem sitemap;
- OG parcial;
- schema parcial;

corrigir o item correspondente.

PARTE XII — VALIDAR A CORREÇÃO

44. Passo 23 — Executar validação dirigida

Auditorias reais utilizaram, conforme caso:

- ESLint direcionado;
- `git diff --check`;
- build;
- inspeção de HTML SSR;
- origem oficial simulada;
- navegação;
- teste de links.

---

45. Não transformar a lista em suíte universal

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Essas validações são exemplos reais.

Não existe uma combinação única obrigatória para toda auditoria.

---

46. Passo 24 — Validar metadados

Conferir, conforme alteração:

- title;
- description;
- canonical;
- OG;
- Twitter.

---

47. Passo 25 — Validar HTML inicial

Quando a correção envolve renderização/indexabilidade:

- inspecionar resposta inicial;
- confirmar conteúdo/metadados/schema.

---

48. Passo 26 — Validar sitemap/robots

Quando alterados:

- abrir;
- conferir conteúdo;
- conferir domínio;
- conferir referência entre arquivos.

---

49. Passo 27 — Validar links/breadcrumbs

Quando alterados:

- navegar;
- clicar;
- confirmar destino.

PARTE XIII — REGRESSÃO

50. Passo 28 — Revalidar áreas relacionadas

Exemplos:

Metadados compartilhados

Revisar mais de uma rota.

Helper de origem

Revisar canonical/OG/sitemap.

Blog

Revisar listagem + artigo.

Robots/noindex

Revisar áreas públicas e privadas relacionadas.

---

51. Não quebrar o site para subir score

Uma correção SEO/AEO/GEO não pode:

- prejudicar navegação;
- quebrar SSR;
- quebrar Blog;
- expor área privada;
- alterar conteúdo factual indevidamente.

PARTE XIV — ACHADOS DE SEGURANÇA EM AUDITORIAS MISTAS

52. SEO/GEO pode trazer item de segurança

Exemplos:

- HTTPS;
- security headers;
- exposição;
- crawler access.

> **Limite**
>
> Este SOP não executa auditoria completa de segurança.

Se houver achado relevante:

- encaminhar à validação de segurança proporcional;
- não misturar correção superficial com autorização/PII.

PARTE XV — STATUS DA AUDITORIA

53. Passo 29 — Consolidar resultado

Para cada item, registrar uma situação:

- corrigido;
- preservado porque já correto;
- não reproduzido;
- pendente de cliente;
- pendente de decisão;
- fora do escopo.

---

54. Não usar “100%” sem critério sustentado

Não declarar auditoria perfeita apenas porque:

- ferramentas não mostram mais alertas;
- score subiu.

O estado precisa refletir o que foi realmente validado.

PARTE XVI — STOP CONDITION

55. Condição de parada

O SOP está concluído quando:

> **os achados relevantes da auditoria foram confrontados com o estado atual, classificados, os deltas reais foram corrigidos, as correções foram tecnicamente revalidadas no nível proporcional e os itens dependentes de cliente, norma ou decisão humana estão explicitamente registrados.**

---

56. Checklist mínimo

Antes de encerrar:

- [ ] origem/data/versão da auditoria identificadas;
- [ ] estado atual conferido;
- [ ] achados relevantes reproduzidos;
- [ ] cada item classificado;
- [ ] UI × HTML inicial comparados quando necessário;
- [ ] domínio/origem conferidos quando aplicável;
- [ ] metadados conferidos quando aplicável;
- [ ] schemas conferidos quando aplicável;
- [ ] sitemap/robots conferidos quando aplicável;
- [ ] Blog conferido quando aplicável;
- [ ] delta real implementado;
- [ ] validação técnica executada;
- [ ] regressão dirigida executada;
- [ ] dependências humanas registradas;
- [ ] norma técnica mantida em camada separada.

---

57. O que NÃO é necessário para encerrar

Não exigir universalmente:

- score SEOmator específico;
- mesma checklist AEO/GEO em todo projeto;
- `llms.txt`;
- mesma quantidade de FAQ;
- mesma contagem de palavras;
- mesmo conjunto de schemas;

- auditoria externa paga;
- implementação de toda recomendação da ferramenta.

PARTE XVII — CONTINGÊNCIAS

58. Auditoria diz que elemento está ausente, mas UI mostra

1. confirmar UI;
2. conferir HTML inicial;
3. determinar modo de coleta;
4. classificar limitação × problema real;
5. corrigir somente se houver delta.

---

59. Relatório pede item já resolvido

Classificar:

já resolvido

e preservar.

---

60. Auditoria usa domínio preview

Reavaliar resultados dependentes de:

- canonical;
- OG;
- sitemap;
- origem.

---

61. Score baixo por conteúdo não validado

Não inventar conteúdo.

Registrar dependência humana.

---

62. Ferramenta sugere `llms.txt`

Não tornar obrigatório apenas porque apareceu na auditoria.

Avaliar à luz da norma vigente e decisão WSS.

---

63. Crawler não vê conteúdo CSR

Investigar se:

- é conteúdo principal/indexável;
- deveria estar no HTML inicial conforme norma;
- é interação legítima CSR.

---

64. Correção SEO quebra componente/rota

Tratar como regressão.

Usar SOP-009.

PARTE XVIII — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

65. Não inventar

O AS-IS não possui:

- score mínimo SEOmator universal;
- ferramenta externa única obrigatória;
- checklist AEO/GEO único para todos os projetos;
- `llms.txt` universal obrigatório;
- número universal de FAQs;
- contagem universal de palavras;
- conjunto fixo de schemas para todo site;
- pipeline única de validação;
- obrigação universal de usar auditoria paga;
- regra de implementar toda recomendação de crawler.

PARTE XIX — NORMA TÉCNICA VIGENTE

66. Arquitetura SSR-first / híbrida

O documento normativo vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Ele estabelece, para páginas públicas indexáveis e conforme a classificação aplicável:

- SSR como padrão;
- HTML completo e específico por URL;
- conteúdo principal no HTML inicial;
- metadados;
- canonical;
- dados estruturados aplicáveis;
- links rastreáveis;
- status HTTP adequado.

Para verificação de conformidade, consultar sem duplicar:

- `WSS_SEO_AEO_GEO_Matriz_Requisitos_Tecnicos_Parte2_v0.3`;
- `WSS_SEO_AEO_GEO_Matriz_Schema_por_Tipo_Pagina_v0.3`;
- `WSS_SEO_AEO_GEO_Checklist_PrePublicacao_v0.3`.

---

67. CSR permitido para interação

A norma admite CSR em:

- busca;
- filtros;
- formulários;
- Chat;
- Chatbot;
- CTA;
- áreas privadas.

Não deve substituir o conteúdo principal público.

---

68. Blog normativo

A norma vigente define:

- listagem SSR;
- artigo SSR;

- metadados próprios;
- schema;
- breadcrumbs;
- 404 real;
- publicação via Admin sem depender de rebuild para cada artigo.

---

69. Áreas privadas

Normativamente:

- noindex;
- fora do sitemap;
- autenticação adequada;
- autorização não baseada apenas em ocultação visual.

Decisões preservadas

Este SOP não decide:

- política padrão para crawlers de treinamento;
- seleção definitiva de ferramenta paga;
- obrigatoriedade universal futura do GA4;
- fronteira entre entrega padrão e serviço recorrente.

PARTE XX — ANTI-PATTERNS

70. Executar relatório antigo sem conferir `main`

Pode reintroduzir problema.

71. Corrigir H1 que já existe sem entender o crawler

Pode atacar o sintoma errado.

72. Transformar score em meta absoluta

Ferramenta não substitui julgamento técnico.

73. Universalizar checklist de um projeto

Daniel P não define sozinho o padrão WSS.

74. Inventar revisão clínica

Revisão humana precisa existir de fato.

75. Adicionar schema sem correspondência real

Structured data não deve inventar entidade/conteúdo.

76. Melhorar SEO sacrificando arquitetura/UX

Correção não pode quebrar o produto.

PARTE XXI — OUTPUT

77. Saída esperada

Ao final:

Triagem

- auditoria comparada ao estado atual;
- itens classificados.

Correção

- somente deltas reais alterados;
- itens corretos preservados.

Validação

- metadados/HTML/schema/sitemap/robots/links revalidados conforme aplicabilidade;
- regressão proporcional realizada.

Governança

- pendências humanas explícitas;
- recomendações não obrigatórias separadas;
- norma técnica identificada em camada própria.

PARTE XXII — PRÓXIMO PASSO

78. Encaminhamento

Depois do SOP-010:

- bug/regressão → **SOP-009**;
- feedback do cliente → **SOP-011**;
- publicação final → **SOP-012**.

---

79. Referências

Manual proprietário

- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_03_Construcao_Visual_v1.1_FINAL.md`;
- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`;
- `WSS_Manual_07_Publicacao_Entrega_Pos_Publicacao_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`.

Evidência técnica principal

- `WSS_VAL_003_Auditorias_SEO_AEO_GEO_Seguranca_v0.1.md`.

Evidência de troubleshooting

- `WSS_VAL_004_Troubleshooting_Estudos_de_Caso_v0.1.md`.

Norma técnica relacionada

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`;
- `WSS_Padrao_Tecnico_Arquitetura_SEO_v0.1` — baseline histórico/proveniência;
- `WSS_SEO_AEO_GEO_Matriz_Requisitos_Tecnicos_Parte2_v0.3`;
- `WSS_SEO_AEO_GEO_Matriz_Schema_por_Tipo_Pagina_v0.3`;
- `WSS_SEO_AEO_GEO_Matriz_Crawlers_Diretivas_v0.2`;
- `WSS_SEO_AEO_GEO_Checklist_PrePublicacao_v0.3`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-011
Processar Rodada de Feedback via ajustes.md

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-011

Processar Rodada de Feedback via ajustes.md

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 6 — Testes, QA, Debugging, Segurança e SEO
MANUAIS RELACIONADOS	Manual 2 — Entrada e Contexto do Projeto; Manual 3 — Construção Visual; Manual 5 — Desenvolvimento com Codex, Lovable e GitHub; Manual 7 — Publicação, Entrega e Pós-Publicação; Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais
ESCOPO	AS-IS — recebimento, consolidação, execução e encerramento de rodadas de feedback/correção

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 7

2. Gatilho — quando usar 7

3. Quando NÃO usar 7

4. Papéis 8

5. Pré-condições 9

6. Inputs 9

PARTE I — RECEBER O FEEDBACK 11

7. Canal predominante AS-IS 12

8. Passo 1 — Reunir todas as mensagens da rodada 12

9. Passo 2 — Identificar a versão do site à qual o feedback se refere 12

PARTE II — CONSOLIDAR OS PEDIDOS 14

10. Passo 3 — Transformar mensagens em itens claros 15

11. Passo 4 — Agrupar mensagens que tratam do mesmo problema 15

12. Passo 5 — Separar pedidos independentes 15

PARTE III — CLASSIFICAR O PEDIDO 17

13. Passo 6 — Classificar cada item 18

14. Ajuste claro 18

15. Pedido ambíguo 18

16. Contradição 19

17. Possível mudança de escopo 19

18. Bug 19

19. Conteúdo/fato pendente 19

PARTE IV — REGISTRAR EM AJUSTES.MD 21

20. Passo 7 — Atualizar ajustes.md 22

21. Estrutura mínima do item 22

22. Não inventar campos obrigatórios 22

23. Passo 8 — Manter separados os itens bloqueados 23

PARTE V — PREPARAR A EXECUÇÃO 24

24. Passo 9 — Confirmar contexto antes de alterar 25

25. Passo 10 — Definir escopo da alteração para a IA 25

26. Regra de contenção 25

PARTE VI — EXECUTAR ITEM POR ITEM 26

27. Passo 11 — Selecionar o próximo item executável 27

28. Passo 12 — Implementar a alteração 27

29. Passo 13 — Validar o item imediatamente 27

30. Passo 14 — Marcar item concluído 28

PARTE VII — ORDEM DA RODADA 29

31. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — prioridade formal 30

32. Critério prático 30

PARTE VIII — EVITAR REGRESSÃO 31

33. Passo 15 — Preservar o que já foi aprovado 32

34. Passo 16 — Revalidar componente compartilhado quando alterado 32

35. Passo 17 — Revalidar breakpoint relacionado 32

36. Passo 18 — Revalidar fluxo funcional quando necessário 32

PARTE IX — IA CONFERINDO A LISTA 34

37. Passo 19 — Pedir revisão da lista inteira 35

38. Limite da revisão por IA 35

PARTE X — REVISÃO HUMANA FINAL DA RODADA 36

39. Passo 20 — Percorrer os itens concluídos 37

40. Passo 21 — Conferir itens ainda abertos 37

41. Passo 22 — Revalidar áreas afetadas 37

PARTE XI — EVIDÊNCIAS DE ANTES/DEPOIS 38

42. Uso opcional 39

43. AUSÊNCIA ATUAL DE PADRONIZAÇÃO 39

PARTE XII — NOVO FEEDBACK DURANTE A RODADA 40

44. Passo 23 — Registrar pedidos novos 41

45. Evitar rodada infinita silenciosa 41

PARTE XIII — FEEDBACK APÓS PUBLICAÇÃO TEMPORÁRIA 42

46. Contexto 43

47. Não confundir publicação temporária com entrega final 43

PARTE XIV — RELAÇÃO COM QA **44**

48. Feedback não substitui QA 45

49. QA não substitui feedback 45

PARTE XV — RELAÇÃO COM DEBUGGING **46**

50. Quando um pedido revela bug 47

PARTE XVI — RELAÇÃO COM ESCOPO **48**

51. Sinais de possível mudança de escopo 49

52. Passo 24 — Interromper execução automática 49

53. Não mascarar feature nova como “ajuste” 49

PARTE XVII — CONTEÚDO E FATOS **50**

54. Passo 25 — Separar preferência textual de fato 51

55. IA não completa lacuna factual 51

PARTE XVIII — STOP CONDITION **52**

56. Condição de parada da rodada 53

57. Checklist mínimo 53

58. O que NÃO é necessário para encerrar 53

PARTE XIX — QUANDO NÃO ENCERRAR **55**

59. Item marcado sem validação 56

60. Item ambíguo executado por inferência 56

61. Mudança de escopo executada sem decisão 56

62. Regressão encontrada 56

63. Cliente ainda precisa fornecer fato essencial 56

PARTE XX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO **57**

64. Não inventar 58

PARTE XXI — NORMA TÉCNICA VIGENTE **59**

65. Feedback não pode remover requisito normativo silenciosamente 60

66. Norma ≠ preferência 60

PARTE XXII — ANTI-PATTERNS **61**

67. Corrigir diretamente da memória do WhatsApp 62

68. Executar todos os pedidos em um único prompt amplo 62
69. Marcar item assim que o código muda 62
70. Deixar itens ambíguos escondidos no meio da lista 62
71. Tratar feature nova como correção simples 62
72. Confiar apenas na conferência da IA 62
73. Fechar rodada sem reler a lista completa 62

PARTE XXIII — OUTPUT **64**

74. Saída esperada 65

PARTE XXIV — PRÓXIMO PASSO **66**

75. Encaminhamento 67
76. Referências 67

1. Finalidade

Transformar feedback recebido em uma rodada controlada de correções, reduzindo perda de pedidos, interpretação ambígua, regressão e encerramento prematuro.

Fluxo principal:

```
feedback recebido
→ reunir pedidos
→ consolidar quando necessário
→ separar ajuste de dúvida/escopo
→ registrar em ajustes.md
→ executar item por item
→ marcar concluído
→ revisar lista completa
→ revalidar áreas afetadas
→ confirmar pendências
→ encerrar rodada
```

Este SOP documenta o fluxo AS-IS atualmente praticado.

Ele não cria um novo sistema de tickets, não substitui o Flow e não transforma WhatsApp em ferramenta formal única de gestão.

---

2. Gatilho — quando usar

Usar quando houver uma rodada de feedback sobre o projeto, especialmente após:

- revisão interna;
- publicação temporária;
- revisão do cliente;
- reunião;
- troca de mensagens;
- rodada de QA;
- validação de conteúdo;
- entrega parcial.

Pode ser usado tanto para uma lista pequena quanto para uma rodada ampla.

---

3. Quando NÃO usar

Não usar como substituto de:

- definição de escopo;
- briefing;

- Sitemap;
- debugging técnico profundo;
- QA completo;
- auditoria SEO/AEO/GEO;
- publicação.

Encaminhamentos

- bug não trivial → SOP-009;
- QA geral → SOP-008;
- auditoria SEO/AEO/GEO → SOP-010;
- produção/entrega → SOP-012.

---

4. Papéis

4.1 Cliente

Pode:

- apontar correções;
- sugerir melhorias;
- validar textos/imagens;
- solicitar alterações;
- indicar problemas percebidos.

O cliente não decide sozinho, por meio de uma mensagem ambígua, se algo passa a fazer parte do escopo técnico.

4.2 Desenvolvedor

No AS-IS, costuma:

- ler o feedback;
- compreender o pedido;
- registrar;
- começar correções;
- marcar itens;
- revisar a rodada;
- revalidar o site.

4.3 Felipe / Gestão WSS

Pode:

- consolidar pedidos;
- resolver ambiguidade;
- interpretar contradições;
- decidir possível mudança de escopo;
- orientar prioridade;
- validar dúvida comercial/operacional.

> **Regra**
>
> Se o pedido:

- é pouco claro;
- contradiz decisão anterior;
- sugere feature nova;
- muda escopo;

> **DECISÃO HUMANA NECESSÁRIA — retornar para Felipe/gestão antes de executar.**

---

5. Pré-condições

Antes de iniciar:

1. feedback foi recebido;
2. projeto/versão correta estão identificados;
3. Dev consegue acessar os pedidos;
4. contexto do projeto continua disponível;
5. existe local para manter `ajustes.md`.

> **Importante**
>
> Não começar a executar uma rodada extensa apenas a partir de memória das mensagens.

---

6. Inputs

Podem incluir:

- mensagens de WhatsApp;

- documento do cliente;
- comentários de reunião;
- prints;
- vídeos;
- áudios transcritos/interpretados;
- mensagens internas;
- notas do Flow;
- feedback de QA;
- lista de correções anterior;
- `ajustes.md` já existente.

PARTE I — RECEBER O FEEDBACK

7. Canal predominante AS-IS

O principal canal de feedback atualmente identificado é:

WhatsApp

O Flow existe para:

- Tasks;
- registros internos;
- organização operacional.

> **Importante**
>
> Isso não significa:

> “todo feedback deve obrigatoriamente chegar por WhatsApp”.

É apenas o canal predominante no AS-IS.

---

8. Passo 1 — Reunir todas as mensagens da rodada

Antes de executar amplamente:

- localizar mensagens;
- reunir pedidos relacionados;
- identificar anexos;
- identificar prints;
- identificar áudios;
- identificar contexto adicional.

Objetivo

Evitar corrigir apenas a última mensagem e esquecer pedidos anteriores da mesma rodada.

---

9. Passo 2 — Identificar a versão do site à qual o feedback se refere

Confirmar se o cliente avaliou:

- preview;

- domínio temporário;
- versão atual;
- versão anterior.

Se houver dúvida

Comparar com o estado atual antes de alterar.

PARTE II — CONSOLIDAR OS PEDIDOS

10. Passo 3 — Transformar mensagens em itens claros

Cada solicitação deve virar um item que permita responder:

- onde;
- o que mudar;
- qual resultado esperado.

Exemplo

Mensagem:

> “Essa parte aqui poderia ficar mais forte.”

Item ainda insuficiente.

É necessário entender:

- qual seção;
- o que significa “mais forte”;
- tipografia?
- contraste?
- composição?
- conteúdo?

---

11. Passo 4 — Agrupar mensagens que tratam do mesmo problema

Se várias mensagens descrevem a mesma alteração:

- consolidar em um único item;
- preservar nuances importantes.

Objetivo

Evitar executar duas vezes a mesma correção com formulações diferentes.

---

12. Passo 5 — Separar pedidos independentes

Não transformar uma rodada inteira em uma única instrução ampla.

Separar por:

- página;
- seção;
- módulo;
- tipo de alteração.

PARTE III — CLASSIFICAR O PEDIDO

13. Passo 6 — Classificar cada item

Categorias práticas:

A. Ajuste claro

Pedido compreensível e dentro do escopo.

B. Pedido ambíguo

Não há informação suficiente para executar com segurança.

C. Contradição

Conflita com:

- briefing;
- Sitemap;
- decisão anterior;
- outro pedido do mesmo cliente.

D. Possível mudança de escopo

Pode incluir:

- nova página;
- nova funcionalidade;
- novo módulo;
- integração não prevista;
- mudança estrutural significativa.

E. Bug

Comportamento não funciona conforme esperado.

F. Conteúdo/fato pendente

Exige informação do cliente.

---

14. Ajuste claro

Pode seguir para `ajustes.md`.

---

15. Pedido ambíguo

Não inferir silenciosamente.

Ação

- esclarecer;
- ou retornar à gestão quando necessário.

---

16. Contradição

Não escolher automaticamente qual mensagem “vence”.

Ação

> **DECISÃO HUMANA NECESSÁRIA.**

---

17. Possível mudança de escopo

Não implementar como se fosse ajuste normal.

Ação

- registrar;
- separar da rodada executável;
- retornar a Felipe/gestão.

---

18. Bug

Se simples e evidente:

- pode ser corrigido na rodada.

Se não trivial:

- encaminhar ao SOP-009.

---

19. Conteúdo/fato pendente

Se a alteração depende de informação:

- não inventar;
- registrar a pendência;
- manter o item aberto ou bloqueado.

Especial atenção a:

- formação;
- títulos;
- especialidades;
- procedimentos;
- resultados;
- informações médicas;
- endereços;
- contatos;
- outros fatos sensíveis.

PARTE IV — REGISTRAR EM `ajustes.md`

20. Passo 7 — Atualizar `ajustes.md`

Centralizar os pedidos executáveis em:

`ajustes.md`

Função

O arquivo funciona como:

- memória da rodada;
- checklist humano;
- contexto para IA;
- proteção contra esquecimento.

---

21. Estrutura mínima do item

O AS-IS não define template universal.

Mas o item precisa ser claro o suficiente para execução.

Exemplo funcional:

```
- [ ] Home — Hero: trocar a foto principal pela imagem X enviada pelo cliente.
```

Ou:

```
- [ ] Especialidade Y — remover a frase Z e manter o restante da seção.
```

---

22. Não inventar campos obrigatórios

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe template formal obrigatório com:

- prioridade numérica;
- severidade;
- responsável;
- prazo;
- story points;

- ID;
- categoria;
- evidência.

Esses campos podem existir quando úteis, mas não fazem parte do AS-IS universal.

---

23. Passo 8 — Manter separados os itens bloqueados

Itens que dependem de decisão podem permanecer registrados como:

- pendente;
- bloqueado;
- aguardando retorno.

> **Regra**
>
> Não marcá-los como concluídos apenas porque não serão executados naquela rodada.

PARTE V — PREPARAR A EXECUÇÃO

24. Passo 9 — Confirmar contexto antes de alterar

Antes da rodada:

- reler `ajustes.md`;
- reler contexto relevante;
- confirmar página/módulo;
- preservar áreas já validadas.

---

25. Passo 10 — Definir escopo da alteração para a IA

Ao trabalhar com Codex, Claude ou Lovable, informar:

- item atual;
- página/seção;
- o que pode mudar;
- o que deve permanecer.

Exemplo

```
Corrija apenas o item 4 do ajustes.md.
Trabalhe somente a seção Hero mobile.
Preserve o desktop já aprovado.
Não altere outras páginas nem o Admin.
```

---

26. Regra de contenção

Não enviar toda a lista para implementação ampla quando:

- há itens muito diferentes;
- existem decisões pendentes;
- o risco de regressão é alto.

Trabalhar item por item ou em grupos coerentes.

PARTE VI — EXECUTAR ITEM POR ITEM

27. Passo 11 — Selecionar o próximo item executável

Escolher item que:

- está claro;
- não está bloqueado;
- não exige decisão de escopo.

---

28. Passo 12 — Implementar a alteração

Usar a ferramenta apropriada:

Visual

- Codex;
- Claude.

Backend/banco/configuração

No pós-criação, o Codex pode ser a primeira camada de investigação/condução. Conforme a tarefa:

- Codex pode executar diretamente o que couber;
- Lovable/Lovable Cloud podem ser usados diretamente pelo Dev ou operados/orquestrados pelo Codex quando a integração for necessária.

A criação inicial da estrutura herdada de backend/banco/auth continua pertencendo ao remix da Base no Lovable.

Conteúdo

- ferramenta de IA + validação humana/factual.

Bug

- SOP-009 se não trivial.

---

29. Passo 13 — Validar o item imediatamente

Depois da alteração:

- abrir área;
- conferir resultado;
- comparar com o pedido.

Não marcar como concluído antes de verificar.

---

30. Passo 14 — Marcar item concluído

Quando validado:

```
- [x] ...
```

ou mecanismo equivalente já usado no projeto.

> **Importante**
>
> “Código alterado” não equivale a “item concluído”.

PARTE VII — ORDEM DA RODADA

31. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — prioridade formal

Não existe regra universal de:

- primeiro visual;
- depois conteúdo;
- depois funcional;
- ordem por página;
- prioridade numérica.

---

32. Critério prático

O Dev pode ordenar considerando:

- dependência;
- facilidade;
- risco;
- página;
- coerência da alteração.

> **DECISÃO HUMANA NECESSÁRIA**
>
> Quando ordem impactar:

- escopo;
- entrega;
- prazo;
- prioridade comercial.

PARTE VIII — EVITAR REGRESSÃO

33. Passo 15 — Preservar o que já foi aprovado

Cada prompt/correção deve limitar:

- página;
- seção;
- breakpoint;
- módulo.

---

34. Passo 16 — Revalidar componente compartilhado quando alterado

Se o item toca:

- Header;
- Footer;
- CTA compartilhado;
- componente reutilizado;
- layout genérico;

verificar outras páginas relevantes.

---

35. Passo 17 — Revalidar breakpoint relacionado

Se ajuste é:

- mobile → conferir mobile e depois regressão básica desktop;
- desktop → conferir desktop e comportamento mobile relacionado.

---

36. Passo 18 — Revalidar fluxo funcional quando necessário

Exemplos:

- CTA;
- formulário;
- Lead;
- Chat;
- Admin.

Se a correção toca o fluxo:

> executar novamente o caminho relevante.

PARTE IX — IA CONFERINDO A LISTA

37. Passo 19 — Pedir revisão da lista inteira

Depois de executar os itens:

- pedir à IA para reler `ajustes.md`;
- comparar itens marcados;
- apontar algum item não tratado ou incompleto.

Objetivo

Usar a IA como segunda leitura da rodada.

---

38. Limite da revisão por IA

A IA não decide sozinha se:

- pedido subjetivo foi atendido;
- cliente ficará satisfeito;
- mudança de escopo foi aprovada;
- fato sensível está correto.

A revisão automatizada é auxiliar.

PARTE X — REVISÃO HUMANA FINAL DA RODADA

39. Passo 20 — Percorrer os itens concluídos

O Dev deve:

- revisar a lista;
- abrir áreas alteradas;
- confirmar resultado.

---

40. Passo 21 — Conferir itens ainda abertos

Classificar cada restante como:

- ainda não executado;
- bloqueado;
- aguardando cliente;
- aguardando gestão;
- fora do escopo;
- novo bug.

---

41. Passo 22 — Revalidar áreas afetadas

A profundidade depende da alteração.

Visual

- página;
- desktop/notebook/mobile relevantes.

Funcional

- fluxo;
- persistência;
- Admin.

Compartilhado

- demais superfícies que reutilizam o componente.

PARTE XI — EVIDÊNCIAS DE ANTES/DEPOIS

42. Uso opcional

Quando necessário, o Dev pode gerar:

- prints de antes/depois;
- documento demonstrando alterações.

Finalidade

- comprovar correção;
- facilitar comunicação com o cliente;
- esclarecer alteração visual.

---

43. AUSÊNCIA ATUAL DE PADRONIZAÇÃO

Prints antes/depois:

- não são obrigatórios em toda rodada;
- não possuem formato universal.

São usados quando ajudam.

PARTE XII — NOVO FEEDBACK DURANTE A RODADA

44. Passo 23 — Registrar pedidos novos

Se chegam novos pedidos durante a execução:

- não depender da memória;
- adicionar à lista;
- classificar.

---

45. Evitar rodada infinita silenciosa

Se a lista muda continuamente:

- manter claro o que pertence à rodada original;
- identificar novos pedidos;
- escalar quando houver impacto de escopo/prioridade.

PARTE XIII — FEEDBACK APÓS PUBLICAÇÃO TEMPORÁRIA

46. Contexto

O site pode estar publicado em:

`webstarstudio.site`

para revisão do cliente.

Nesse momento:

- feedback chega;
- Admin pode continuar sendo configurado em paralelo;
- `ajustes.md` centraliza a rodada.

---

47. Não confundir publicação temporária com entrega final

A existência de URL pública não impede:

- ajustes;
- refinamento;
- correções;
- validação final.

PARTE XIV — RELAÇÃO COM QA

48. Feedback não substitui QA

O cliente pode não detectar:

- rota quebrada;
- persistência de Lead;
- problema de perfil;
- regressão técnica;
- SEO;
- erro de produção.

O SOP-008 continua necessário conforme o estágio.

---

49. QA não substitui feedback

Uma experiência tecnicamente correta pode ainda precisar de:

- conteúdo;
- imagem;
- tom;
- ajuste visual;
- preferência do cliente.

As duas camadas são complementares.

PARTE XV — RELAÇÃO COM DEBUGGING

50. Quando um pedido revela bug

Exemplo:

> “O formulário diz que enviou, mas não recebemos nada.”

Isso não deve ser tratado apenas como alteração visual.

Ação

- reproduzir;
- usar SOP-009;
- depois voltar ao item em `ajustes.md`.

PARTE XVI — RELAÇÃO COM ESCOPO

51. Sinais de possível mudança de escopo

Exemplos:

- nova página não prevista;
- nova integração;
- novo módulo;
- nova jornada;
- nova funcionalidade;
- alteração estrutural ampla.

---

52. Passo 24 — Interromper execução automática

Quando houver sinal de escopo:

1. registrar o pedido;
2. não implementar;
3. retornar a Felipe/gestão;
4. aguardar decisão.

---

53. Não mascarar feature nova como “ajuste”

Essa separação protege:

- contrato;
- prazo;
- arquitetura;
- expectativa do cliente.

PARTE XVII — CONTEÚDO E FATOS

54. Passo 25 — Separar preferência textual de fato

Preferência

Exemplo:

- “quero um texto mais direto”.

Pode ser reescrito.

Fato

Exemplo:

- formação;
- título;
- cidade de atendimento;
- procedimento realizado.

Precisa ser confirmado.

---

55. IA não completa lacuna factual

Se o feedback pede adicionar dado inexistente:

- solicitar fonte;
- usar material oficial;
- não inferir.

PARTE XVIII — STOP CONDITION

56. Condição de parada da rodada

O SOP está concluído quando:

> **todos os pedidos recebidos para a rodada foram registrados e classificados; os itens executáveis foram implementados e marcados; a lista completa foi revisada; as áreas afetadas foram revalidadas; e qualquer item remanescente está explicitamente identificado como pendente, bloqueado, aguardando decisão ou fora do escopo.**

---

57. Checklist mínimo

Antes de encerrar:

- [ ] feedback da rodada reunido;
- [ ] versão avaliada identificada;
- [ ] pedidos consolidados;
- [ ] itens ambíguos separados;
- [ ] contradições identificadas;
- [ ] possíveis mudanças de escopo escaladas;
- [ ] itens executáveis registrados em `ajustes.md`;
- [ ] cada item executado foi validado;
- [ ] itens concluídos marcados;
- [ ] IA realizou conferência da lista completa quando útil;
- [ ] Dev revisou a lista;
- [ ] áreas afetadas revalidadas;
- [ ] componentes compartilhados revalidados quando necessário;
- [ ] pendências remanescentes classificadas;
- [ ] fatos sensíveis não foram inventados.

---

58. O que NÃO é necessário para encerrar

Não exigir universalmente:

- cliente responder novamente a cada item;
- documento formal de aprovação por item;
- print antes/depois em toda correção;
- ticket no Flow para cada ajuste;
- prioridade/severidade numérica;
- template rígido de `ajustes.md`;

- reunião após toda rodada.

PARTE XIX — QUANDO NÃO ENCERRAR

59. Item marcado sem validação

Não encerrar.

Reabrir o item.

---

60. Item ambíguo executado por inferência

Revisar com humano.

---

61. Mudança de escopo executada sem decisão

Não considerar a rodada corretamente concluída.

Registrar e retornar à gestão.

---

62. Regressão encontrada

Encaminhar:

- SOP-008 para QA;
- SOP-009 para bug;

conforme o caso.

---

63. Cliente ainda precisa fornecer fato essencial

Manter pendência explícita.

Não inventar conclusão.

PARTE XX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

64. Não inventar

O AS-IS não possui:

- canal único obrigatório de feedback;
- template universal de `ajustes.md`;
- IDs obrigatórios;
- severidade numérica;
- prioridade universal;
- workflow formal de aprovação por item;
- SLA universal para correções;
- obrigação de registrar todo pedido no Flow;
- documento antes/depois obrigatório;
- ordem rígida de execução;
- aceite formal do cliente item por item.

PARTE XXI — NORMA TÉCNICA VIGENTE

65. Feedback não pode remover requisito normativo silenciosamente

Se um pedido do cliente contradiz:

- segurança;
- arquitetura pública;
- SEO;
- autenticação;
- privacidade;
- outra norma vigente;

não executar automaticamente.

Quando o conflito envolver SEO/AEO/GEO, consultar:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Ação

> **DECISÃO HUMANA NECESSÁRIA**, preservando a camada normativa aplicável.

---

66. Norma ≠ preferência

É possível adaptar:

- visual;
- linguagem;
- composição.

Não é aceitável tratar requisito técnico aprovado como simples preferência visual.

PARTE XXII — ANTI-PATTERNS

67. Corrigir diretamente da memória do WhatsApp

Risco:

- esquecer pedido;
- misturar rodadas;
- interpretar errado.

---

68. Executar todos os pedidos em um único prompt amplo

Risco:

- regressão;
- alteração fora de escopo;
- dificuldade de validar item a item.

---

69. Marcar item assim que o código muda

O item só termina após validação.

---

70. Deixar itens ambíguos escondidos no meio da lista

Separar para decisão.

---

71. Tratar feature nova como correção simples

Risco de escopo.

---

72. Confiar apenas na conferência da IA

A revisão humana continua necessária.

---

73. Fechar rodada sem reler a lista completa

Esse é exatamente o risco que `ajustes.md` busca reduzir.

PARTE XXIII — OUTPUT

74. Saída esperada

Ao final:

Registro

- `ajustes.md` atualizado;
- itens concluídos marcados;
- pendências explicitadas.

Execução

- correções implementadas;
- mudanças fora de escopo não executadas silenciosamente.

Validação

- áreas afetadas revalidadas;
- regressões tratadas/encaminhadas.

Governança

- dúvidas/contradições/escalonamentos resolvidos ou registrados;
- fatos pendentes preservados como pendências.

PARTE XXIV — PRÓXIMO PASSO

75. Encaminhamento

Depois do SOP-011:

Se ainda houver QA

→ **SOP-008 — Executar QA Visual, Funcional e Responsivo**

Se houver bug

→ **SOP-009 — Diagnosticar e Corrigir Bug por Camada**

Se houver auditoria técnica

→ **SOP-010 — Tratar Auditoria SEO, AEO e GEO**

Se a rodada está encerrada e o projeto está pronto

→ **SOP-012 — Preparar, Publicar, Validar Produção e Entregar Admin**

---

76. Referências

Manual proprietário

- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`

Manuais relacionados

- `WSS_Manual_02_Entrada_Contexto_Projeto_v1.1_FINAL.md`;
- `WSS_Manual_03_Construcao_Visual_v1.1_FINAL.md`;
- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`;
- `WSS_Manual_07_Publicacao_Entrega_Pos_Publicacao_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
- `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
- `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

Contexto de Processo

- `Roadmap_Sites_WSS_Modelo_Padrao_v0.2_FINAL.md`.

Status: FINAL — APROVADO

WEB STAR STUDIO

PROCEDIMENTO OPERACIONAL

MÉTODO DE DESENVOLVIMENTO DE SITES

SOP-012
Preparar, Publicar, Validar Produção e Entregar Admin

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — APROVADO

PROCEDIMENTO OPERACIONAL

SOP-012

Preparar, Publicar, Validar Produção e Entregar Admin

VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — APROVADO
MANUAL PROPRIETÁRIO	Manual 7 — Publicação, Entrega e Pós-Publicação
MANUAIS RELACIONADOS	Manual 4 — Projeto-Base, Admin e Funcionalidades; Manual 5 — Desenvolvimento com Codex, Lovable e GitHub; Manual 6 — Testes, QA, Debugging, Segurança e SEO; Manual 8 — Boas Práticas, Anti-Patterns, Práticas Depreciadas e Casos Reais
ESCOPO	AS-IS — fechamento técnico da entrega, da prontidão de publicação à entrega do Admin e transição pós-entrega

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 8

2. Gatilho — quando usar 8

3. Quando NÃO usar 9

4. Papéis 9

5. Pré-condições 10

6. Inputs 11

PARTE I — CONFIRMAR PRONTIDÃO 12

7. Passo 1 — Confirmar o gate prático de pré-publicação 13

8. Passo 2 — Conferir fluxos essenciais 13

9. Passo 3 — Conferir ajustes do cliente 13

10. Passo 4 — Conferir auditorias técnicas aplicáveis 14

PARTE II — PUBLICAÇÃO TEMPORÁRIA 15

11. Quando usar 16

12. Finalidade 16

13. Passo 5 — Publicar temporariamente quando aplicável 16

14. Admin pode continuar em paralelo 17

15. Passo 6 — Enviar ambiente para revisão 17

PARTE III — PROCESSAR A REVISÃO 18

16. Passo 7 — Receber feedback 19

17. Passo 8 — Consolidar em ajustes.md 19

18. Passo 9 — Concluir a rodada 19

PARTE IV — VALIDAÇÃO FINAL PRÉ-PRODUÇÃO 20

19. Passo 10 — Executar última revalidação proporcional 21

20. Não existe “reconfiguração final universal” 21

21. Build/smoke quando relevante 21

PARTE V — DEFINIR O CENÁRIO DE DOMÍNIO 23

22. Passo 11 — Identificar o cenário 24
23. Cenário B — cliente não possui domínio 24
24. Passo 12 — Confirmar responsável/acesso 24
25. Cliente pode participar 24

PARTE VI — DNS **25**
26. Passo 13 — Identificar a configuração específica 26
27. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — receita única 26
28. Sequência segura no nível AS-IS 26

PARTE VII — PUBLICAÇÃO EM PRODUÇÃO **27**
29. Passo 14 — Publicar no domínio oficial 28
30. Passo 15 — Confirmar acesso ao domínio 28
31. Mudança de origem 28

PARTE VIII — QA PÓS-PUBLICAÇÃO **29**
32. Passo 16 — Abrir Home 30
33. Passo 17 — Percorrer páginas principais 30
34. Passo 18 — Testar refresh direto 30
35. Passo 19 — Testar formulários/Leads 30
36. Passo 20 — Testar Chat/CTA 31
37. Passo 21 — Testar Chatbot 31
38. Passo 22 — Testar Blog 31
39. Passo 23 — Testar login/Admin 32
40. Passo 24 — Testar mobile em produção 32
41. Passo 25 — Conferir domínio/HTTPS 32
42. Passo 26 — Conferir Analytics quando aplicável 32
43. Passo 27 — Conferir pontos de SEO 33

PARTE IX — NORMA TÉCNICA VIGENTE **34**
44. Requisitos normativos aplicáveis à publicação 35
45. AS-IS × NORMA 35

PARTE X — FALHA EM PRODUÇÃO **36**
46. Passo 28 — Classificar a falha 37
47. Falha localizada 37
48. Falha geral 37

49. Passo 29 — Restaurar versão funcional 37

50. Prioridade 38

51. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — rollback 38

PARTE XI — CONFIRMAR ESTABILIDADE DE PRODUÇÃO 39

52. Passo 30 — Reexecutar fluxo pós-fix 40

53. Não considerar publish técnico como validação 40

PARTE XII — LIMPEZA DE DADOS DE TESTE 41

54. Momento 42

55. Passo 31 — Inventariar dados de teste 42

56. Passo 32 — Diferenciar teste de dado real 42

57. Passo 33 — Limpar/resetar dados de teste 42

58. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — script universal 43

PARTE XIII — PERFIS INTERNOS 44

59. Passo 34 — Confirmar perfis testados 45

60. Não entregar credencial interna ao cliente 45

PARTE XIV — GERAR ACESSO DO CLIENTE 46

61. Passo 35 — Gerar link no Admin 47

62. Passo 36 — Enviar o link 47

63. Passo 37 — Cliente cria a própria conta 47

PARTE XV — ORIENTAR O USO DO ADMIN 48

64. Passo 38 — Demonstrar o Admin 49

65. Manual da geração atual × obrigatoriedade universal 49

66. Não inventar treinamento formal 49

PARTE XVI — CONFIRMAR ENTREGA INICIAL 50

67. Passo 39 — Confirmar três condições principais 51

68. Publicação não basta 51

69. Passo 40 — Registrar pendências pós-entrega 51

PARTE XVII — HANDOFF PARA MANUTENÇÃO/RECORRÊNCIA 52

70. Passo 41 — Identificar continuidade aplicável 53

71. Limite 53

72. Relação com o Processo Sites WSS 53

PARTE XVIII — STOP CONDITION 54

73. Condição de parada 55

74. Checklist mínimo 55

75. O que NÃO é necessário para encerrar universalmente 56

PARTE XIX — QUANDO NÃO ENCERRAR 57

76. Produção está amplamente quebrada 58

77. Lead/Chat/Blog crítico não funciona em produção 58

78. Dados de teste estão misturados ao ambiente entregue 58

79. Cliente ainda não consegue criar/acessar conta 58

80. Domínio/HTTPS não está funcional 58

PARTE XX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO 59

81. Não inventar 60

PARTE XXI — NORMA TÉCNICA VIGENTE 61

82. Publicação deve preservar requisitos técnicos aplicáveis 62

83. Norma não reescreve o AS-IS histórico 62

PARTE XXII — ANTI-PATTERNS 63

84. “Publicou, terminou” 64

85. Inventar DNS padrão a partir de um único cliente 64

86. Corrigir produção amplamente quebrada sem restaurar 64

87. Entregar dados de teste 64

88. Entregar credencial interna ao cliente 64

89. Limpar banco sem diferenciar teste de dado real 64

90. Encerrar sem validar mobile em produção 64

91. Fechar entrega com Admin sem acesso do cliente 65

PARTE XXIII — OUTPUT 66

92. Saída esperada 67

PARTE XXIV — RELAÇÃO COM O PROCESSO SITES WSS 68

93. Marco 9 — Publicação e Pós-Entrega 69

94. Processo ≠ SOP 69

PARTE XXV — REFERÊNCIAS 70

95. Manual proprietário 71

96. Manuais relacionados 71

97. SOPs relacionados 71
98. Baseline 71
99. Contexto de Processo 71
100. Norma técnica relacionada 71

1. Finalidade

Conduzir o projeto da validação final até a entrega técnica inicial, incluindo:

- publicação temporária quando aplicável;
- revisão do cliente;
- conclusão dos ajustes;
- definição do cenário de domínio;
- publicação em produção;
- QA pós-publicação;
- tratamento de falhas;
- limpeza dos dados de teste;
- criação do acesso do cliente ao Admin;
- orientação de uso;
- confirmação da entrega inicial.

Fluxo principal:

```
QA/validação final
→ publicação temporária quando aplicável
→ revisão do cliente
→ ajustes
→ validação final
→ definir domínio
→ publicar em produção
→ QA pós-publicação
→ corrigir/rollback se necessário
→ limpar dados de teste
→ gerar acesso do cliente
→ orientar Admin
→ confirmar entrega
→ manutenção/recorrência
```

---

2. Gatilho — quando usar

Usar quando:

- páginas públicas já passaram pelo QA necessário;
- fluxos essenciais foram testados;
- ajustes do cliente foram concluídos ou estão no estágio final;
- Admin está funcional ou próximo da conclusão;
- o projeto está pronto para avançar ao ambiente publicado/final.

---

3. Quando NÃO usar

Não usar:

- durante construção inicial;
- para configurar módulos do Admin;
- para executar QA geral pela primeira vez;
- para investigar bug complexo;
- para tratar auditoria SEO/AEO/GEO;
- para definir escopo comercial de recorrência.

Procedimentos relacionados

- QA → SOP-008;
- bug → SOP-009;
- SEO/AEO/GEO → SOP-010;
- feedback → SOP-011.

---

4. Papéis

4.1 Desenvolvedor

Responsável, no AS-IS, por:

- confirmar prontidão técnica;
- participar da publicação;
- validar produção;
- corrigir falhas localizadas;
- acionar/restaurar versão funcional quando necessário;
- limpar dados de teste;
- gerar link de criação da conta do cliente;
- apoiar orientação do Admin.

4.2 Analice

No fluxo atual, possui acesso à GoDaddy e participa das etapas de domínio quando necessário.

4.3 Felipe / CEO

No fluxo atual:

- possui acesso à GoDaddy;
- pode conduzir etapas de domínio;
- pode enviar o link de criação da conta do cliente;

- participa de decisões quando necessário.

4.4 Cliente

Pode precisar:

- revisar o ambiente temporário;
- fornecer feedback;
- fornecer acesso ao provedor do domínio;
- autorizar alteração;
- informar dados;
- criar sua própria conta do Admin pelo link recebido.

4.5 Equipe WSS autorizada

Na publicação temporária:

> qualquer integrante autorizado da equipe pode publicar via Lovable no domínio `webstarstudio.site`.

A ação não é exclusiva do Dev.

---

5. Pré-condições

Antes de avançar para publicação final, confirmar que:

1. páginas relevantes foram validadas;
2. desktop funciona;
3. notebook funciona;
4. mobile funciona;
5. validação interna foi concluída;
6. fluxos essenciais continuam funcionando;
7. ajustes do cliente necessários foram tratados;
8. módulos aplicáveis estão suficientemente validados;
9. pendências bloqueantes estão resolvidas;
10. responsável pelo domínio está definido.

---

6. Inputs

Podem incluir:

- versão final candidata;
- Sitemap;
- domínio temporário;
- domínio definitivo;
- acesso ao provedor;
- credenciais internas;
- perfis de teste;
- dados de QA;
- `ajustes.md`;
- auditorias técnicas relevantes;
- configurações de produção;
- informações do cliente.

PARTE I — CONFIRMAR PRONTIDÃO

7. Passo 1 — Confirmar o gate prático de pré-publicação

O Dev considera o site pronto quando:

- páginas estão validadas;
- desktop está correto;
- notebook está correto;
- mobile está correto;
- validação interna foi concluída;
- fluxos técnicos previamente testados continuam funcionando.

---

8. Passo 2 — Conferir fluxos essenciais

Conforme o projeto:

- navegação;
- CTAs;
- formulários;
- Leads;
- Chat;
- Chatbot;
- Blog;
- login;
- Admin;
- perfis;
- integrações aplicáveis.

> **Regra**
>
> Não pressupor que “funcionou semana passada” ainda basta.

A versão candidata deve preservar esses fluxos.

---

9. Passo 3 — Conferir ajustes do cliente

Se houve rodada de feedback:

- abrir `ajustes.md`;
- confirmar itens concluídos;
- confirmar pendências;

- garantir que nenhum item bloqueante permanece escondido.

Se necessário:

> voltar ao SOP-011.

---

10. Passo 4 — Conferir auditorias técnicas aplicáveis

Quando o projeto exige:

- SEO/AEO/GEO;
- segurança;
- outra validação técnica;

confirmar que o estado necessário para publicação foi atingido.

> **Limite**
>
> Não repetir auditoria inteira apenas por ritual.

PARTE II — PUBLICAÇÃO TEMPORÁRIA

11. Quando usar

A publicação temporária pode ser usada antes do domínio definitivo.

Domínio padrão observado:

`webstarstudio.site`

---

12. Finalidade

Permitir:

- navegação externa;
- revisão do cliente;
- validação em ambiente publicado;
- coleta de ajustes.

---

13. Passo 5 — Publicar temporariamente quando aplicável

Publicar via Lovable no domínio temporário.

> **Importante**
>
> A publicação temporária não significa:

- entrega final;
- domínio definitivo;
- Admin concluído.

> **NORMA TÉCNICA VIGENTE — staging/preview**
>
> Quando usado como staging/preview, o ambiente temporário deve permanecer:

- não indexável;
- fora do sitemap oficial;
- sem assumir a própria origem como canonical de produção.

A autoridade normativa é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

---

14. Admin pode continuar em paralelo

É aceitável:

```
site público em revisão
+
Admin ainda em finalização
```

Essa sobreposição faz parte do AS-IS.

---

15. Passo 6 — Enviar ambiente para revisão

Disponibilizar o endereço ao cliente/equipe responsável.

Orientar a revisão conforme o projeto.

PARTE III — PROCESSAR A REVISÃO

16. Passo 7 — Receber feedback

Coletar:

- correções;
- observações;
- sugestões;
- validações;
- conteúdos pendentes.

---

17. Passo 8 — Consolidar em `ajustes.md`

Usar SOP-011.

Não executar mudanças ambíguas ou de escopo automaticamente.

---

18. Passo 9 — Concluir a rodada

Depois das correções:

- revalidar;
- revisar a lista completa;
- confirmar ausência de bloqueios.

PARTE IV — VALIDAÇÃO FINAL PRÉ-PRODUÇÃO

19. Passo 10 — Executar última revalidação proporcional

Antes do domínio final:

- abrir páginas;
- testar principais ações;
- confirmar módulos críticos;
- verificar regressão.

> **Regra**
>
> A profundidade depende das mudanças ocorridas desde o QA anterior.

---

20. Não existe “reconfiguração final universal”

> **AUSÊNCIA ATUAL DE PADRONIZAÇÃO**
>
> Não existe etapa formal obrigatória de:

```
reconfigurar todas as variáveis
→ callbacks
→ secrets
→ integrações
```

imediatamente antes do publish.

A expectativa é que já estejam corretos.

Porém

Se a mudança de ambiente exige valor específico:

- conferir.

---

21. Build/smoke quando relevante

Se existe risco associado a:

- build;
- bundler;
- runtime;

- stack;

pode ser necessário validar artefato de produção.

> **Limite**
>
> O smoke do Guilherme Mendonça é real, mas **não é gate universal**.

PARTE V — DEFINIR O CENÁRIO DE DOMÍNIO

22. Passo 11 — Identificar o cenário

Cenário A — cliente já possui domínio

Cliente fornece:

- dados;
- acesso;
- informações necessárias.

A WSS conecta/publica nesse domínio.

---

23. Cenário B — cliente não possui domínio

A WSS:

- cria/registra o domínio;
- utiliza GoDaddy;
- conecta o site.

---

24. Passo 12 — Confirmar responsável/acesso

No AS-IS, GoDaddy é concentrada em:

- Analice;
- Felipe/CEO.

O Dev normalmente não realiza essa etapa diretamente por não possuir acesso.

---

25. Cliente pode participar

Quando o domínio está em provedor do cliente, pode ser necessário:

- acesso;
- autorização;
- informação;
- transferência/configuração.

PARTE VI — DNS

26. Passo 13 — Identificar a configuração específica

Antes de alterar DNS, entender:

- provedor;
- registros atuais;
- estado do domínio;
- tipo de conexão necessário;
- acesso disponível.

---

27. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — receita única

Não existe sequência universal como:

```
sempre excluir registro X
→ sempre criar registro Y
→ sempre aguardar N horas
```

Este SOP não inventa essa receita.

---

28. Sequência segura no nível AS-IS

O procedimento geral é:

1. identificar cenário;
2. garantir responsável/acesso;
3. conectar domínio;
4. aguardar/confirmar propagação quando aplicável;
5. verificar HTTPS;
6. testar produção.

A implementação exata depende do caso.

PARTE VII — PUBLICAÇÃO EM PRODUÇÃO

29. Passo 14 — Publicar no domínio oficial

Executar a publicação/conexão conforme a infraestrutura do projeto.

---

30. Passo 15 — Confirmar acesso ao domínio

Verificar:

- domínio abre;
- domínio correto;
- HTTPS;
- ausência de erro geral.

---

31. Mudança de origem

Ao passar para domínio final, podem mudar:

- canonical;
- OG;
- sitemap;
- origem de requests;
- cookies;
- callbacks;
- integrações.

> **Regra**
>
> Revalidar o que depende de origem.

PARTE VIII — QA PÓS-PUBLICAÇÃO

32. Passo 16 — Abrir Home

Confirmar:

- carregamento;
- visual;
- navegação;
- domínio.

---

33. Passo 17 — Percorrer páginas principais

Verificar:

- páginas internas;
- rotas relevantes;
- navegação;
- links.

---

34. Passo 18 — Testar refresh direto

Em rotas relevantes:

- abrir URL diretamente;
- atualizar navegador.

Objetivo

Detectar:

- problema de roteamento;
- comportamento diferente do preview;
- falha de servidor/configuração.

---

35. Passo 19 — Testar formulários/Leads

Quando aplicável:

```
formulário
→ enviar
→ mensagem
→ Admin
→ Lead
```

---

36. Passo 20 — Testar Chat/CTA

Verificar:

- botão/widget;
- abertura;
- destino;
- conversa;
- integração.

---

37. Passo 21 — Testar Chatbot

Quando aplicável:

- iniciar;
- responder;
- confirmar Lead/conversa;
- verificar handoff básico.

Se houve alteração relevante:

> usar SOP-006.

---

38. Passo 22 — Testar Blog

Verificar:

- listagem;
- artigo;
- navegação;
- slug;
- refresh direto quando relevante.

---

39. Passo 23 — Testar login/Admin

Quando aplicável:

- login;
- Admin;
- módulo essencial;
- perfil.

---

40. Passo 24 — Testar mobile em produção

Abrir no mobile.

Verificar:

- menu;
- layout;
- CTA;
- formulário;
- Chat;
- navegação.

---

41. Passo 25 — Conferir domínio/HTTPS

Confirmar:

- domínio oficial;
- certificado;
- ausência de alerta.

---

42. Passo 26 — Conferir Analytics quando aplicável

O Roadmap prevê Analytics quando aplicável.

AS-IS

Analytics/GTM não são responsabilidade pessoal universal do Dev.

Verificar conforme responsabilidade/projeto.

> **NORMA TÉCNICA VIGENTE**
>
> GA4 é recomendado, não obrigatório universal.

Essa classificação não transforma Analytics/GTM em responsabilidade pessoal automática do Dev.

---

43. Passo 27 — Conferir pontos de SEO

Quando aplicável:

- canonical;
- domínio oficial;
- metadata;
- sitemap;
- robots;
- indexabilidade;
- noindex de áreas privadas;
- Search Console.

O procedimento detalhado pertence ao SOP-010.

PARTE IX — NORMA TÉCNICA VIGENTE

44. Requisitos normativos aplicáveis à publicação

A autoridade vigente é:

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`.

Conforme a classificação aplicável, a publicação deve preservar:

- origem oficial de produção;
- domínio oficial em canonical;
- metadados corretos;
- sitemap;
- `robots.txt`;
- Google Search Console configurado na publicação;
- SSR/HTML inicial;
- status HTTP;
- noindex de áreas privadas;
- segurança/autorização.

O requisito de Search Console é da WSS no go-live e não implica responsabilidade pessoal universal do Dev.

Para o recorte técnico de publicação, consultar:

- `WSS_SEO_AEO_GEO_Checklist_PrePublicacao_v0.3`;
- `WSS_SEO_AEO_GEO_Checklist_PosPublicacao_v0.2`.

---

45. AS-IS × NORMA

AS-IS

Descreve a rotina real de publicação e validação.

NORMA

Pode estabelecer requisitos adicionais.

Não misturar as duas camadas.

PARTE X — FALHA EM PRODUÇÃO

46. Passo 28 — Classificar a falha

Falha localizada

Uma parte falha, mas o site continua utilizável.

Falha geral

O site ou parte essencial fica amplamente indisponível/quebrada.

---

47. Falha localizada

Aplicar:

> correção emergencial localizada.

Depois:

- revalidar área;
- executar regressão proporcional.

Se exigir investigação:

> SOP-009.

---

48. Falha geral

Quando produção está amplamente quebrada:

1. recuperar versão funcional;
2. restabelecer o site;
3. investigar depois.

---

49. Passo 29 — Restaurar versão funcional

Pode utilizar, conforme projeto:

- histórico do Lovable;
- commit anterior;

- build/versão anterior funcional.

---

50. Prioridade

> **disponibilidade primeiro quando produção está amplamente quebrada.**

Não manter o cliente em versão geral quebrada apenas para investigar “ao vivo”.

---

51. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — rollback

Não existe ritual universal formal por categoria.

Este SOP não cria:

- SLA;
- playbook completo;
- tag obrigatória;
- branch obrigatória;
- tabela de severidade.

PARTE XI — CONFIRMAR ESTABILIDADE DE PRODUÇÃO

52. Passo 30 — Reexecutar fluxo pós-fix

Depois de correção/rollback:

- abrir domínio;
- navegar;
- refazer fluxo afetado;
- revalidar áreas relacionadas.

---

53. Não considerar publish técnico como validação

Anti-pattern:

> “o deploy terminou, então o site está bom.”

O ambiente publicado precisa ser testado.

PARTE XII — LIMPEZA DE DADOS DE TESTE

54. Momento

Depois de produção estar validada e antes de entregar o Admin ao cliente.

---

55. Passo 31 — Inventariar dados de teste

Conforme o projeto:

- Leads falsos;
- conversas;
- posts de QA;
- registros temporários;
- outros dados criados para teste.

---

56. Passo 32 — Diferenciar teste de dado real

Controle humano obrigatório

Não apagar por padrão tudo que “parece recente”.

Confirmar:

- origem;
- nome;
- contexto;
- finalidade.

---

57. Passo 33 — Limpar/resetar dados de teste

No AS-IS, o Dev pode pedir:

- Lovable;
- Codex;

para realizar a limpeza.

> **Regra**
>
> A atuação deve atingir apenas dados de teste confirmados.

---

58. AUSÊNCIA ATUAL DE PADRONIZAÇÃO — script universal

Não existe script universal de purge.

Não inventar automação genérica.

PARTE XIII — PERFIS INTERNOS

59. Passo 34 — Confirmar perfis testados

Antes da entrega:

- perfil Equipe/cliente;
- perfil Webstar.

Já devem ter sido testados no desenvolvimento/QA.

---

60. Não entregar credencial interna ao cliente

Credenciais internas:

- permanecem sob controle WSS;
- ficam registradas no Flow.

O cliente deve criar conta própria.

PARTE XIV — GERAR ACESSO DO CLIENTE

61. Passo 35 — Gerar link no Admin

O Dev gera link para criação da conta do cliente.

---

62. Passo 36 — Enviar o link

O link pode ser enviado por:

- Dev;
- Felipe.

Não existe exclusividade formal.

---

63. Passo 37 — Cliente cria a própria conta

O cliente utiliza o link para configurar seu acesso.

> **Regra**
>
> Não usar a conta interna da equipe como conta definitiva do cliente.

PARTE XV — ORIENTAR O USO DO ADMIN

64. Passo 38 — Demonstrar o Admin

No fluxo atual, costuma haver reunião/demonstração.

Pode incluir:

- login;
- navegação;
- módulos principais;
- ações relevantes.

A demonstração permanece aplicável mesmo quando a geração corrente do Admin possui módulo próprio de Manual.

---

65. Manual da geração atual × obrigatoriedade universal

A geração atual pode incluir um módulo de **Manual** dentro do Admin, com capítulos por perfil. Esse recurso não estava disponível em todos os projetos históricos.

A existência dessa capacidade não cria obrigação universal retroativa: não existe requisito de que todo projeto, de toda geração, entregue um manual idêntico.

---

66. Não inventar treinamento formal

Este SOP não exige:

- duração mínima;
- certificado;
- roteiro universal;
- apostila padrão.

PARTE XVI — CONFIRMAR ENTREGA INICIAL

67. Passo 39 — Confirmar três condições principais

A entrega inicial é considerada concluída quando:

1. ajustes do cliente foram implementados;
2. site está publicado/entregue;
3. Admin está funcional, validado e entregue.

---

68. Publicação não basta

Ainda podem existir antes do fechamento:

- QA de produção;
- limpeza;
- acesso do cliente;
- orientação do Admin.

---

69. Passo 40 — Registrar pendências pós-entrega

Se houver item não bloqueante que permanece:

- registrar;
- encaminhar corretamente;
- não ocultar.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE XVII — HANDOFF PARA MANUTENÇÃO/RECORRÊNCIA

70. Passo 41 — Identificar continuidade aplicável

Depois da entrega, o projeto pode seguir para:

- manutenção;
- Blog;
- conteúdo;
- SEO/AEO/GEO;
- evolução;
- outros serviços contratados.

Quando SEO/AEO/GEO recorrente estiver contratado/aplicável, encaminhar para:

- `WSS_SEO_AEO_GEO_SOP_Monitoramento_Recorrente_v0.2`;
- `WSS_SEO_AEO_GEO_SOP_Otimizacao_Artigos_v0.2`;
- `WSS_SEO_AEO_GEO_Matriz_Indicadores_Monitoramento_v0.2`.

---

71. Limite

Este SOP não define:

- pacote comercial;
- preço;
- frequência;
- SLA de manutenção.

A fronteira entre **entrega padrão × serviço recorrente permanece decisão estratégica/comercial aberta** e não é definida por este SOP.

---

72. Relação com o Processo Sites WSS

O Processo/Flow é responsável por:

- status;
- encerramento formal;
- recorrência;
- registros do Marco 9.

O SOP documenta apenas a execução técnica.

PARTE XVIII — STOP CONDITION

73. Condição de parada

O SOP está concluído quando:

> **o site está em produção no domínio apropriado, passou pela validação pós-publicação necessária, qualquer falha crítica foi tratada, dados de teste foram limpos com segurança, o cliente recebeu/criou seu acesso próprio ao Admin, o Admin foi orientado no nível aplicável e a entrega técnica inicial foi confirmada, com eventuais pendências e continuidade registradas.**

---

74. Checklist mínimo

Antes de encerrar:

- [ ] gate de prontidão confirmado;
- [ ] feedback/ajustes encerrados ou explicitados;
- [ ] cenário de domínio definido;
- [ ] responsável/acesso de domínio confirmado;
- [ ] publicação em produção concluída;
- [ ] domínio/HTTPS verificados;
- [ ] páginas/rotas verificadas;
- [ ] refresh direto verificado em rotas relevantes;
- [ ] formulários/Leads revalidados quando aplicáveis;
- [ ] Chat/CTA revalidado quando aplicável;
- [ ] Blog revalidado quando aplicável;
- [ ] login/Admin revalidado quando aplicável;
- [ ] mobile revalidado;
- [ ] SEO/domain-origin conferido quando aplicável;
- [ ] falhas críticas tratadas;
- [ ] dados de teste identificados e limpos;
- [ ] perfis internos confirmados;
- [ ] link do cliente gerado;
- [ ] cliente recebeu acesso;
- [ ] orientação do Admin realizada quando aplicável;
- [ ] pendências pós-entrega registradas;
- [ ] continuidade/manutenção encaminhada quando aplicável.

---

75. O que NÃO é necessário para encerrar universalmente

Não exigir:

- sequência DNS única;
- smoke obrigatório em todo site;
- manual formal do Admin;
- Analytics/GTM configurado pessoalmente pelo Dev;
- script universal de limpeza;
- zero pendências não bloqueantes;
- recorrência comercial contratada;
- checklist técnico idêntico para todos os provedores.

PARTE XIX — QUANDO NÃO ENCERRAR

76. Produção está amplamente quebrada

Não encerrar.

Restaurar versão funcional.

77. Lead/Chat/Blog crítico não funciona em produção

Não encerrar enquanto a capacidade contratada relevante permanecer quebrada.

78. Dados de teste estão misturados ao ambiente entregue

Não encerrar.

Limpar com segurança.

79. Cliente ainda não consegue criar/acessar conta

Não encerrar a entrega do Admin.

80. Domínio/HTTPS não está funcional

Não considerar publicação definitiva concluída.

PARTE XX — AUSÊNCIAS ATUAIS DE PADRONIZAÇÃO

81. Não inventar

O AS-IS não possui:

- receita DNS universal;
- responsabilidade universal do Dev por domínio/GoDaddy;
- reconfiguração final universal;
- smoke universal;
- checklist pós-publicação idêntico para todos;
- script universal de limpeza;
- manual obrigatório do Admin;
- treinamento obrigatório padronizado;
- playbook universal de incidente;
- rollback formal por categoria;
- responsabilidade universal do Dev por Analytics/GTM;
- pacote único de manutenção/recorrência.

PARTE XXI — NORMA TÉCNICA VIGENTE

82. Publicação deve preservar requisitos técnicos aplicáveis

Conforme `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`:

- canonical deve usar a origem oficial de produção;
- sitemap/robots devem estar coerentes;
- staging/preview deve permanecer não indexável e fora do sitemap oficial;
- páginas públicas devem preservar arquitetura indexável;
- áreas privadas não devem indexar;
- status/404 devem estar corretos;
- Google Search Console deve estar configurado na publicação;
- autenticação/autorização devem permanecer seguras.

GA4 permanece recomendado, não obrigatório universal.

Este SOP não decide:

- política padrão de crawlers de treinamento;
- seleção definitiva de ferramenta paga;
- obrigatoriedade universal futura do GA4;
- fronteira entre entrega padrão e serviço recorrente.

---

83. Norma não reescreve o AS-IS histórico

Este SOP mantém clara a diferença entre:

- rotina de publicação real;
- exigência normativa aprovada.

PARTE XXII — ANTI-PATTERNS

84. “Publicou, terminou”

Errado.

É necessário QA pós-publicação.

85. Inventar DNS padrão a partir de um único cliente

Cada caso pode variar.

86. Corrigir produção amplamente quebrada sem restaurar

Pode prolongar indisponibilidade.

87. Entregar dados de teste

Não deve ocorrer.

88. Entregar credencial interna ao cliente

O cliente deve usar conta própria.

89. Limpar banco sem diferenciar teste de dado real

Risco de perda.

90. Encerrar sem validar mobile em produção

Mobile faz parte da validação prática do Dev.

91. Fechar entrega com Admin sem acesso do cliente

Publicação pública não substitui entrega operacional.

PARTE XXIII — OUTPUT

92. Saída esperada

Ao final:

Produção

- domínio oficial funcional;
- HTTPS funcional;
- site validado;
- fluxos essenciais preservados.

Admin

- funcional;
- dados de teste limpos;
- perfis internos preservados;
- cliente com acesso próprio.

Entrega

- orientação realizada quando aplicável;
- entrega inicial confirmada.

Continuidade

- pendências registradas;
- manutenção/recorrência encaminhada quando aplicável.

PARTE XXIV — RELAÇÃO COM O PROCESSO SITES WSS

93. Marco 9 — Publicação e Pós-Entrega

No Roadmap Sites WSS, o Marco 9 representa:

- domínio/DNS;
- site publicado;
- Admin acessível;
- convite enviado;
- testes pós-publicação;
- registro de entrega;
- transição para operação/recorrência.

---

94. Processo ≠ SOP

```
Marco 9 / Flow
= o que precisa estar formalmente controlado/concluído

SOP-012
= como a execução técnica atual conduz publicação e entrega
```

Este SOP não substitui:

- status;
- checklists;
- evidências;
- encerramento formal do Flow.

PARTE XXV — REFERÊNCIAS

95. Manual proprietário

- `WSS_Manual_07_Publicacao_Entrega_Pos_Publicacao_v1.1_FINAL.md`

---

96. Manuais relacionados

- `WSS_Manual_04_Projeto_Base_Admin_Funcionalidades_v1.1_FINAL.md`;
- `WSS_Manual_05_Desenvolvimento_Codex_Lovable_GitHub_v1.1_FINAL.md`;
- `WSS_Manual_06_Testes_QA_Debugging_Seguranca_SEO_v1.1_FINAL.md`;
- `WSS_Manual_08_Boas_Praticas_AntiPatterns_Depreciadas_Casos_Reais_v1.1_FINAL.md`.

---

97. SOPs relacionados

- SOP-008 — Executar QA Visual, Funcional e Responsivo;
- SOP-009 — Diagnosticar e Corrigir Bug por Camada;
- SOP-010 — Tratar Auditoria SEO, AEO e GEO;
- SOP-011 — Processar Rodada de Feedback via `ajustes.md`.

---

98. Baseline

- `WSS_Arqueologia_Tecnica_ASIS_v0.3.md`;
- `WSS_Matriz_Cobertura_ASIS_P0_P1_P2_v0.3_FINAL.md`;
- `WSS_Handoff_Consolidacao_Manuais_SOPs_Skills_v0.2.md`;
- `WSS_Metodo_Desenvolvimento_Sites_Plano_Conducao_v0.3.md`;
- `WSS_Metodo_Desenvolvimento_Sites_Indice_Mestre_v0.1_DRAFT.md`.

---

99. Contexto de Processo

- `Roadmap_Sites_WSS_Modelo_Padrao_v0.2_FINAL.md`.

---

100. Norma técnica relacionada

- `WSS_Padrao_Tecnico_Operacional_SEO_AEO_GEO_v0.2_FINAL`;
- `WSS_SEO_AEO_GEO_Checklist_PrePublicacao_v0.3`;
- `WSS_SEO_AEO_GEO_Checklist_PosPublicacao_v0.2`.

Status: FINAL — APROVADO

WEB STAR STUDIO

GOVERNANÇA DOCUMENTAL

MÉTODO DE DESENVOLVIMENTO DE SITES

TERMO DE ATUALIZAÇÃO
Termo de Atualização e Substituição Documental

VERSÃO	DATA	STATUS
v1.1 FINAL	11/09/2026	FINAL — RATIFICAÇÃO DOCUMENTAL DA COLEÇÃO v1.1

GOVERNANÇA DOCUMENTAL

TERMO DE ATUALIZAÇÃO

Termo de Atualização e Substituição Documental

ARQUIVO	WSSTermoAtualizacaoSubstituicaoDocumentalMetodov1.1_FINAL.md
VERSÃO	v1.1 FINAL
DATA	11/09/2026
STATUS	FINAL — RATIFICAÇÃO DOCUMENTAL DA COLEÇÃO v1.1
ESCOPO	atualização de autoridade documental da coleção humana do Método de Desenvolvimento de Sites WSS

NAVEGAÇÃO DO DOCUMENTO

Sumário

Estrutura do documento canônico v1.1 FINAL.

1. Finalidade 4
2. Autoridade documental vigente 4
3. Preservação da coleção v1.0 4
4. Termo de Encerramento v1.0 4
5. Natureza da atualização v1.1 5
6. Fronteiras preservadas 5
7. Skills 5
8. Regra de consulta a partir da v1.1 5
9. Efeito de substituição 6
10. Estado documental resultante 6

1. Finalidade

Este Termo formaliza a atualização controlada da coleção documental do **Método de Desenvolvimento de Sites WSS**, da coleção `v1.0 FINAL` para a coleção `v1.1 FINAL`.

A atualização decorre da validação pós-entrega realizada com o desenvolvedor de referência, consolidada em `WSS_Relatorio_Validacao_PosEntrega_Dev_Manuais_SOPs_v0.2.md`, seguida do Pacote Corretivo Pós-Entrega v1.1 aprovado pela Coordenação e de sua normalização referencial de fechamento.

Este Termo não reabre a etapa de descoberta e não cria novo escopo metodológico.

---

2. Autoridade documental vigente

A partir desta emissão, passam a constituir a **autoridade documental vigente** do Método de Desenvolvimento de Sites WSS:

- 8 Manuais `v1.1 FINAL`;
- 12 SOPs `v1.1 FINAL`;
- `WSS_Metodo_Desenvolvimento_Sites_Indice_Colecao_v1.1_FINAL.md` como porta de entrada oficial da coleção.

Os 20 documentos canônicos são os listados nominalmente no Índice da Coleção `v1.1 FINAL`.

---

3. Preservação da coleção v1.0

A coleção `v1.0 FINAL` permanece preservada integralmente como **baseline histórico** da documentação humana originalmente encerrada.

A substituição documental significa mudança de autoridade vigente; não significa exclusão, sobrescrita ou edição retrospectiva da coleção anterior.

Portanto:

- nenhum Manual `v1.0 FINAL` é editado em lugar;
- nenhum SOP `v1.0 FINAL` é editado em lugar;
- o Índice da Coleção `v1.0 FINAL` permanece preservado como registro histórico;
- artefatos históricos de descoberta, AS-IS, VALs, Handoff e demais evidências mantêm suas versões originais.

---

4. Termo de Encerramento v1.0

O **Termo de Encerramento v1.0 original não é alterado, reemitido nem substituído em lugar**.

A emissão da coleção `v1.1 FINAL` representa uma atualização documental pós-entrega controlada. Ela não invalida o fato de que a fase documental foi formalmente encerrada na v1.0.

---

5. Natureza da atualização v1.1

A `v1.1 FINAL` incorpora exclusivamente:

1. os deltas materiais aprovados no Pacote Corretivo Pós-Entrega decorrente da validação do Dev;
2. as correções de consistência cruzada necessárias para que os documentos afetados não se contradigam;
3. a normalização administrativa das referências internas para a coleção `v1.1 FINAL`;
4. a promoção administrativa de SOP-005 e SOP-006, sem mudança semântica, para manter toda a coleção em uma única versão canônica.

A atualização preserva integralmente os pontos deliberadamente classificados como `NÃO ALTERAR`, inclusive O-03, O-13, O-14 e O-15.

---

6. Fronteiras preservadas

Esta atualização:

- não reabre descoberta;
- não reabre ARQs ou VALs;
- não redesenha TO-BE;
- não altera o Processo Sites WSS fora dos deltas já aprovados;
- não altera normas técnicas externas vigentes;
- não cria novos Manuais ou novos SOPs de conteúdo;
- não converte preferências pessoais do Dev em regra WSS além do que foi expressamente aprovado.

---

7. Skills

As Skills continuam como camada **posterior** à estabilização da documentação humana.

A promoção da coleção para `v1.1 FINAL` não inicia Skills, não define sua quantidade e não cria equivalência automática entre SOP e Skill.

---

8. Regra de consulta a partir da v1.1

Para uso operacional corrente:

1. consultar o `WSS_Metodo_Desenvolvimento_Sites_Indice_Colecao_v1.1_FINAL.md`;

2. utilizar os Manuais e SOPs `v1.1 FINAL` como autoridade documental vigente;
3. recorrer à coleção `v1.0 FINAL` e aos demais artefatos históricos apenas para proveniência, auditoria, comparação e rastreabilidade;
4. preservar normas técnicas especializadas em suas próprias versões vigentes.

---

9. Efeito de substituição

Com a ratificação deste Termo:

```
Coleção v1.0 FINAL
→ preservada como baseline histórico

Coleção v1.1 FINAL
→ autoridade documental vigente
```

A atualização não desfaz o encerramento documental anterior; apenas estabelece a versão vigente após o Pacote Corretivo Pós-Entrega aprovado.

---

10. Estado documental resultante

```
8 Manuais v1.1 FINAL
+
12 SOPs v1.1 FINAL
+
Índice da Coleção v1.1 FINAL
+
Termo de Atualização e Substituição Documental v1.1 FINAL
=
Coleção humana vigente do Método de Desenvolvimento de Sites WSS
com baseline v1.0 preservado
```

Status final: `v1.1 FINAL — AUTORIDADE DOCUMENTAL VIGENTE`