

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

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE II — ANTI-PATTERNS

16. Anti-pattern x 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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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_SE0_AE0_GE0_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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.
-

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

PARTE VII — 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.
-

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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