

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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**.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.**

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.
-

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.
-

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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.

WSS · MÉTODO DE DESENVOLVIMENTO DE SITES

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