

Projeto eBook com IA:

Construí a estrutura do prompt em markdown:

Estrutura Técnica do eBook

Pág 1: O Fim do Boilerplate

- Introdução sobre a **mudança** de paradigma: de
 - “escrevedor de código” para “revisor de arquitetura”
- Como a IA elimina o trabalho repetitivo

Pág 2-3: IA no Ciclo de Vida do Software (SDLC)

- **Tópico 1**
 - Técnicas avançadas de **Prompt Engineering** para **Refactoring** e **Unit Testing**
- **Tópico 2**
 - Uso de IA para **documentação técnica e diagramação**
 - Exemplo: **Mermaid.js** via LLMs

Case Prático

- Exemplo: tarefa de migração de legado ou integração de API
 - Antes: 3 dias
 - Depois: 4 horas
- Depois: 4 horas

Pág 4: Ética, Segurança e Limitações

- **Guia rápido** sobre o que “não enviar para a IA” (segurança de dados)
- Como validar “**alucinações**” em lógicas complexas

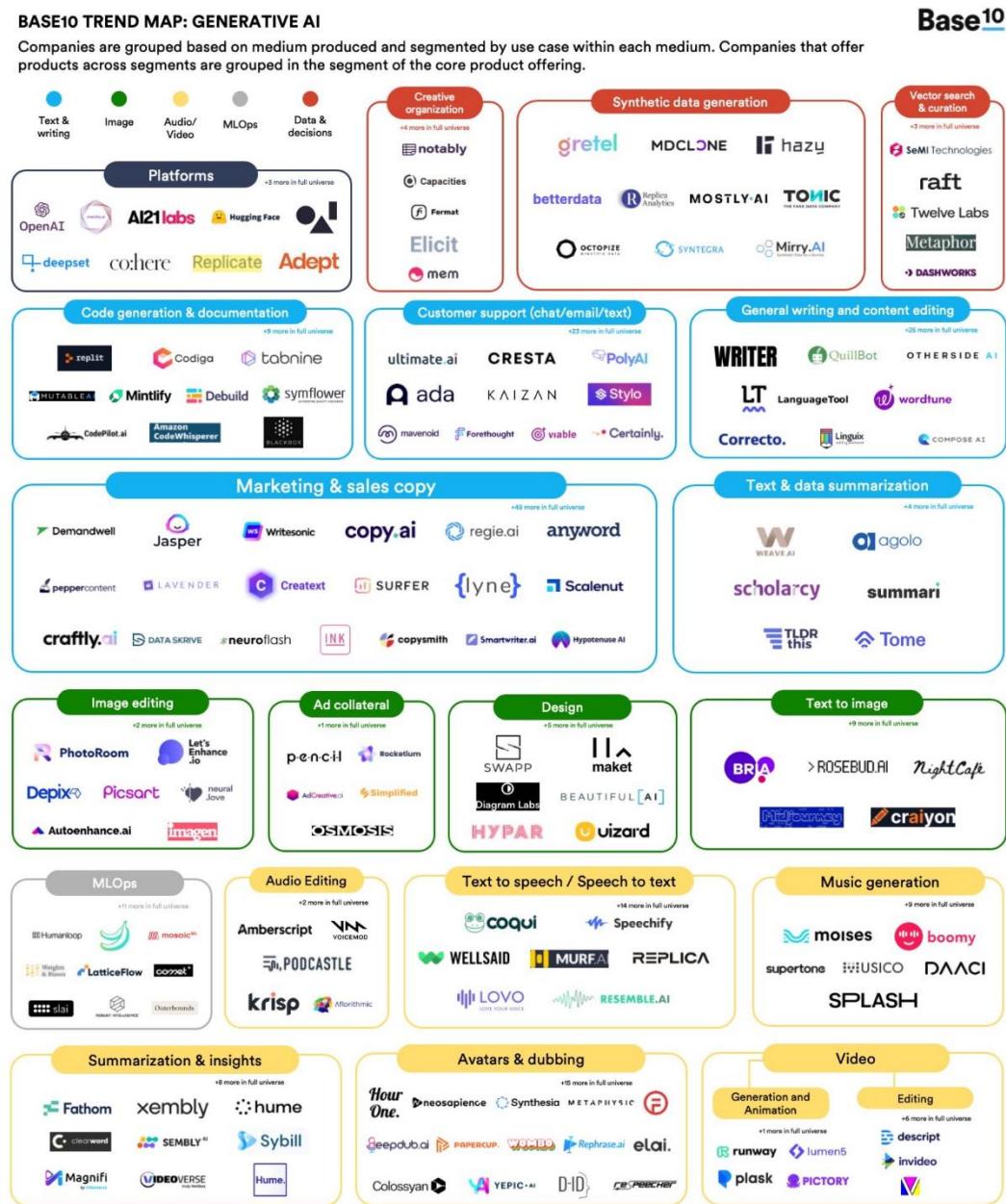
Pág 5: Toolbox & Hands-on

- **Checklist**: Configuração ideal de extensões (Copilot, Cursor, etc.)
- **Desafio**: Problema de algoritmos para o leitor resolver
- **Bonus**: “**Cheat Sheet**” com comandos de contexto para terminal e um homem só”

Diretrizes de Estilo Técnico

- **Linguagem**
 - Use termos como: **refactoring**, **technical debt**, **boilerplate**, **context window**...
- **Formatação**
 - Blocox de código
 - Tabelas comparativas de **ferramentas**
- **O que mudou nesta versão?**
 - **Autoridade Técnica**: De ‘especialista ninfoprodutss’ pra **Staff Engineer**
 - **Foco em Utilidade**: Programadores odeiam textos longos sem aplicação --

Anexei uma imagem de referência do desafio:



Utilizei o ChatGPT baseado no GPT-5.2:

Excelente base. O material já está **bem acima da média** para devs — técnico, pragmático e sem hype. Abaixo vai uma **análise crítica + sugestões de lapidação** para transformar isso em um eBook **vendável, respeitado e reutilizável como ativo de autoridade** (LinkedIn, palestras, cursos internos).

O eBook pronto:



Avaliação Geral (Visão de Staff Engineer)

Força principal:

Você não está ensinando *ferramentas*, está ensinando **como pensar e trabalhar melhor com IA no SDLC**. Isso posiciona o material como:

- *Guia operacional* (não opinativo)
- *Acelerador de carreira*
- Conteúdo que **times de engenharia comprariam**, não só indivíduos

Risco atual:

Algumas partes ainda estão muito “conceituais” para devs mais seniores. Eles querem:

- Exemplos **concretos**
- Checklists **copiáveis**
- Anti-padrões explícitos

Vou atacar exatamente isso abaixo.



Sugestões por Página (com upgrades práticos)

Pág 1 — O Fim do Boilerplate (ótima abertura, falta um “choque”)

O que já funciona

- Mudança de identidade: *escrevedor* → *arquiteto*
- Eliminação de trabalho repetitivo (dor real)

Sugestão de upgrade

Adicionar um **mini-manifesto** (3 bullets) para gerar autoridade imediata:

O novo contrato do desenvolvedor

- Você não é pago para digitar código
- Você é pago para **reduzir risco, organizar complexidade e entregar decisões corretas**
- IA é o novo estagiário que escreve rápido — você valida

📌 Isso cria alinhamento mental imediato.

🔄 Págs 2–3 — IA no SDLC (parte mais forte do eBook)

◆ Tópico 1 — Prompt Engineering para Refactoring & Tests

👉 Sugestão essencial: **Prompt ≠ texto bonito**

Explique como **estruturar contexto**, por exemplo:

Context:

- Language: Python 3.11
- Framework: FastAPI
- Constraint: backward compatibility
- Goal: Reduce cyclomatic complexity

Task:

Refactor the function below and generate unit tests using pytest.

📌 Inclua uma **tabela curta**:

Objetivo	Anti-Padrão	Prompt Correto
Refactor	“Melhore esse código”	“Reduza complexidade mantendo contrato público”
Testes	“Crie testes”	“Cubra edge cases e falhas silenciosas”

◆ Tópico 2 — Documentação & Diagramas

Excelente escolha citar **Mermaid.js**.

💡 Sugestão: deixar explícito o ganho cognitivo:

“Diagramas não são para stakeholders.
São para **você não quebrar algo às 23h numa sexta-feira.**”

Exemplo simples já resolve:

```
sequenceDiagram
Client->>API: Request
API->>Service: Validate
Service->>DB: Query
DB-->>Service: Result
Service-->>API: Response
```

📦 Case Prático (muito bom, mas pode ficar memorável)

Você já tem:

- Antes: 3 dias
- Depois: 4 horas

Sugestão: **explicitar o porquê:**

O tempo não caiu porque a IA “programa melhor”

Caiu porque:

- Context switching foi reduzido
- Documentação foi gerada em paralelo
- Testes nasceram junto com o código

👉 Isso educa, não só impressiona.

Pág 4 — Ética, Segurança e Limitações (parte que dá credibilidade)

Essa seção é **obrigatória** e te diferencia de 90% dos infoprodutos.

Sugestão prática

Inclua uma **regra simples de ouro**:

✗ Nunca envie para a IA nada que você não colocaria em um GitHub público

E um **checklist anti-alucinação**:

- Conferir contratos públicos (API, schemas)
- Executar exemplos gerados
- Validar lógica condicional complexa manualmente

👉 Isso fala direto com quem já se queimou com LLM.

Pág 5 — Toolbox & Hands-on (onde o eBook vira produto)

Checklist de Extensões

Excelente. Sugestão: classificar por **função**, não por marca.

Categoria	Ferramenta
Pair Programming	Copilot, Cursor
Navegação de código	Sourcegraph + LLM
Documentação	Markdown + Mermaid

Desafio Hands-on (ponto alto)

Ótima ideia. Sugestão:

- Problema **simples, mas traiçoeiro**
- Ex: parsing, validação, edge cases

E uma regra clara:

“Você só pode aceitar código depois de explicar **por que ele funciona.**”

 Isso reforça o papel do humano.

Marketing — Está no tom certo (Low Hype ✓)

Títulos

Todos bons. O mais forte para devs seniores:

O Programador Aumentado
IA como alavanca, não como muleta

Copy (ajuste fino)

Essa frase é ouro — eu só ajustaria levemente:

*Menos tempo brigando com Regex e StackOverflow.
Mais tempo em arquitetura, decisões e impacto real.*

Posicionamento Final (importante)

Este eBook **não é para iniciantes** — e isso é bom.

Você está se posicionando como:

- Staff / Principal Engineer
- Tech Lead que já passou por legado

- Alguém que **usa IA com responsabilidade**

👉 Isso permite:

- Preço mais alto
 - Leads mais qualificados
 - Convites para talks, consultoria, treinamento interno
-

Próximo Passo (recomendado)

Se quiser, no próximo passo posso:

1.  Escrever **1 página completa** (ex: SDLC + Refactoring) no tom final
2.  Criar o **Cheat Sheet de Prompts para Devs**
3.  Ajustar a copy para **landing page / LinkedIn**

Diga qual você quer atacar primeiro.