

PLAN GENIE

# Blueprint de Produto

Guia Tático de Implementação e Autonomia

Projeto Demonstração

Plan Genie Framework v2.0 - Product Blueprint

# 00 O Fim da Fórmula Mágica

---

Vivemos em uma era de ruído. Prometem a você diariamente que a IA "fará tudo sozinha", que você pode criar o próximo unicórnio enquanto dorme, e que o código é uma commodity que não requer mais supervisão. **Isso é uma meia-verdade perigosa.**

A tecnologia evoluiu, sim. Hoje, uma pessoa sozinha tem o poder de foda de uma equipe de 20 engenheiros de dez anos atrás. Mas poder sem controle não cria produtos; cria caos. O Plan Genie não está aqui para te vender um sonho impossível. Estamos aqui para te dar um **Método**.

A IA é uma ferramenta de alavancagem infinita. Mas uma alavanca precisa de um ponto de apoio e de alguém para aplicar a força na direção certa. **Você é esse alguém.** Sua responsabilidade é deixar de ser apenas um "usuário" de ferramentas e tornar-se um "Arquiteto de Processos".

*"Os processos são a espinha dorsal de todo projeto bem-sucedido. Embora as ferramentas de IA sejam incríveis para executar esses processos com eficiência, o design do processo em si é o que realmente importa. Ótimos processos + IA = resultados poderosos. Processos ruins + IA = caos executado com eficiência. Projete primeiro, automatize depois."*

## O que isso significa na prática?

Imagine que a IA é uma Ferrari. Se você não souber o caminho (o processo) e não souber dirigir (os conceitos básicos), acelerar fundo só fará você bater no muro mais rápido. Neste documento, nós não te damos apenas o carro; nós ensinamos a traçar

a rota, verificar o motor e pilotar com segurança até o destino. As ferramentas que citamos (Claude, ClickUp, Trae) são apenas sugestões atuais; o **Processo** é eterno.

# 01 Glossário & Fundamentos

---

Para comandar essa nova força de trabalho digital, você precisa aprender o idioma dela. Não vamos falar "tech-ês" complicado, vamos usar analogias para solidificar o entendimento.

## LLM (Large Language Model)

É o "cérebro" incorpóreo. Pense nele como um gênio numa lâmpada muito inteligente, mas que não tem mãos. Ele sabe tudo o que está na internet até a data de seu treinamento, mas não consegue "tocar" no seu computador.

*Exemplos: Claude 3.5 Sonnet, GPT-4o, Gemini 1.5 Pro.*

## IDE (Integrated Development Environment)

É o "chão de fábrica" ou o "ateliê". É o programa instalado no seu computador onde os arquivos de código realmente existem. Sem uma IDE, o código é apenas texto num chat; na IDE, ele vira software real.

*Exemplos Modernos (com IA): Trae, Antigravity, Cursor, VS Code (com plugins).*

## Agente

Se a LLM é o cérebro, o Agente é o "funcionário". Ele tem permissão de usar o cérebro (LLM) para pensar e "mãos" (ferramentas) para agir. Ele pode criar arquivos, ler pastas e rodar comandos.

*Analogia: Uma LLM te diz como fazer um bolo. Um Agente vai na cozinha, pega os ovos e faz o bolo para você.*

## MCP (Model Context Protocol)

São os "super-poderes" ou "ferramentas especializadas" que damos aos Agentes. Por padrão, uma IA não sabe ver suas tarefas no ClickUp ou seu calendário. O MCP é um padrão universal que conecta a IA a serviços externos.

*Exemplo: O "MCP do ClickUp" permite que seu Agente na IDE leia: "Tarefa 1: Criar Tela de Login" diretamente do seu gerenciador, sem você precisar copiar e colar nada.*

---

## **Context Window (Janela de Contexto)**

A "memória de curto prazo" da IA. Imagine que a IA tem uma prancheta. Se você passar 5 horas conversando, a prancheta enche e ela começa a apagar o início para escrever o fim. Isso causa o esquecimento de regras iniciais (alucinação).

*Solução: Nós "resetamos" a memória trocando de chats ou agentes a cada nova fase do projeto.*

---

## 02 A Metodologia em 4 Atos

---

Por que dividir o trabalho? Por que não pedir para a IA "fazer tudo" de uma vez? Porque complexidade gera erro. Em engenharia de software, dividimos problemas grandes em problemas pequenos. Nossa metodologia aplica esse princípio ao uso de IA, criando "silos de contexto" para evitar alucinações.

### 1. O Estrategista (Business Analyst)

Ferramenta: Plan Genie (Nós)

Entrega: O Mapa do Tesouro (PDF de Mercado)



### 2. O Tradutor (Product Owner)

Ferramenta: Claude (Web) + Extended Thinking

Entrega: A Planta Baixa (Manual Funcional)



### 3. O Arquiteto (Tech Lead 1)

Ferramenta: Claude (Web) + MCP ClickUp

Entrega: O Canteiro de Obras (Tasks Organizadas)



#### 4. A Fábrica (Execução & Review)

Ferramenta: IDE (Trae/Cursor)

Ciclo Contínuo: Dev (Agente) ↔ Revisor (Agente)

#### ⚠ O PERIGO DA "CONVERSA INFINITA"

Um erro comum é tentar fazer tudo isso em um único chat (uma única thread). No começo, a IA é brilhante. Na centésima mensagem, ela começa a esquecer nomes de variáveis, inventar bibliotecas e ignorar suas ordens.

**Nossa Regra de Ouro:** Cada caixinha acima é um Chat (ou Agente) NOVO. Contexto limpo = Inteligência máxima.

## 03 O Tradutor (PO)

---

**O Problema:** Desenvolvedores (humanos ou IAs) odeiam ambiguidade. Se você disser "quero um app tipo Uber", existem 10.000 maneiras de interpretar isso.

**A Solução:** O Product Owner (PO) é o papel que traduz "necessidades de negócio" (identificadas no PDF de Mercado) em "requisitos funcionais". Ele não se preocupa com código, mas sim com comportamento. "Quando o usuário clica aqui, acontece X".

### Passo A: Preparando o Terreno

1. Abra sua LLM favorita (Recomendamos Claude 3.5 Sonnet pela capacidade de raciocínio).
2. Ative o modo **"Extended Thinking"** (se disponível). Isso força a IA a gastar "segundos de silêncio" planejando antes de escrever.
3. Anexe o PDF de Estudo de Mercado que o Plan Genie gerou.

### Passo B: O Prompt de Definição

**CONTEXTO:**

Você é um Product Owner Sênior especializado em tradução de negócios para tech.

Você recebeu o Estudo de Mercado em anexo.

**OBJETIVO:**

Criar a DOCUMENTAÇÃO FUNCIONAL para o MVP (Minimum Viable Product).

INSTRUÇÕES:

1. Ignore tecnologias por enquanto (não decida se é React ou Python). Foque no PRODUTO.
2. Liste os Eixos Principais (Épicos) baseados nos Gaps de Mercado identificados.
3. Para cada Épico, crie Histórias de Usuário detalhadas ("Como usuário, eu quero... para que...").
4. Adicione Critérios de Aceite para cada história (Isso é crucial para testarmos depois).

SAÍDA:

Gere um documento estruturado. Se possível, estruture em Markdown para fácil leitura.

*Guarde o resultado. Esse texto gerado é a "Bíblia" do seu produto. Se algo não está escrito aqui, não existirá no software.*

## 04 O Arquiteto (Tech Lead 1)

---

Agora temos a Planta Baixa (Funcional). Precisamos de um Engenheiro Civil para decidir os materiais e organizar a equipe de construção.

Nesta etapa, o Tech Lead (ainda no Claude/Navegador) vai configurar seu sistema de gestão. **Não pule isso.** Tentar codar sem tarefas definidas é a receita para o "Spaghetti Code".

### Passo C: Configuração e MCP

1. Inicie um **NOVO CHAT** (Contexto Limpo).
2. Conecte o **MCP do ClickUp** (ou Trello/Jira).
3. Anexe a "Bíblia Funcional" que o PO criou no passo anterior.

### Passo D: O Prompt de Arquitetura

ROLE:

Você é um Tech Lead e Arquiteto de Software Sênior.

INPUT:

Documentação Funcional em anexo.

AÇÃO 1 - ARQUITETURA:

Defina a Stack Tecnológica ideal para este projeto. (Ex: React + Node ou Next.js + Supabase?). Justifique suas escolhas pensando em velocidade de MVP e escalabilidade futura.

AÇÃO 2 - GESTÃO (Via MCP):

1. Verifique se tem acesso ao meu ClickUp.
2. Crie uma nova Lista/Projeto chamado "Projeto MVP".
3. Crie as colunas de status OBRIGATÓRIAS: "Backlog", "In Progress", "To Review", "To Fix", "Done".
4. Traduza as Histórias de Usuário do anexo em TASKS TÉCNICAS no ClickUp. Cada task deve ser pequena e clara.

EXECUTE AGORA.

Acompanhe a IA criando task por task no seu gerenciador. Quando ela terminar, você terá um plano de batalha pronto.

## 05 A Fábrica (IDE)

---

Chegamos ao coração da operação. Saímos do navegador e entramos na IDE (Trae, Cursor, etc). Aqui a mágica acontece, mas ela precisa de regras rígidas.

Você vai configurar um Agente na sua IDE. Diferente do chat do navegador, este agente tem acesso aos seus arquivos locais E ao ClickUp.

### Passo E: O Prompt do "Operário Padrão" (Dev)

Cole isso nas "Rules" ou "System Prompt" do seu Agente na IDE. Este prompt ensina a ele ética de trabalho e priorização.

#### IDENTITY:

Você é o Desenvolvedor Full-Stack Sênior deste projeto.

#### FERRAMENTAS:

Você tem acesso ao Filesystem (para codar) e ao ClickUp (para saber o que fazer).

#### ALGORITMO DE TRABALHO (Loop Infinito):

- CHECK PRIORIDADE 'CRÍTICA' (Status: To Fix):
  - Vá ao ClickUp. Existe algo em 'To Fix'?
  - SIM: PARE TUDO. Isso é um bug ou reprovação. Leia o comentário do Tech Lead. Corrija o arquivo imediatamente. Mova de volta para 'To Review' quando corrigir.
- CHECK PRIORIDADE 'CONTINUIDADE' (Status: In Progress):
  - Não há nada em 'To Fix'. Existe algo em 'In Progress'?
  - SIM: Significa que você começou e não terminou. Continue essa task.
- CHECK PRIORIDADE 'NOVA' (Status: Backlog):
  - Não há nada com pendência. Pegue a task DO TOPO do Backlog.
  - Mova ela para 'In Progress' IMEDIATAMENTE (para sinalizar que está

```
trabalhando).  
- Leia a descrição. Crie/Edite os arquivos necessários.
```

```
DEFINIÇÃO DE PRONTO (DoD):
```

- O código roda?
- Seguiu os requisitos?
- Mova a task para 'To Review'.
- Comente: "Pronto para análise".
- NÃO FAÇA COMMIT NA MAIN AINDA.

Com esse prompt, você só precisa dizer "Trabalhe" no chat da IDE. O Agente vai automaticamente buscar a task, executar e pedir revisão.

## 06 O Guardião (Tech Lead 2)

---

Aqui está o segredo que separa amadores de profissionais. Amadores confiam cegamente no código da IA. Profissionais revisam.

Como você (talvez) não saiba ler código, usaremos a própria IA para revisar a si mesma, mas num papel diferente. Isso quebra o viés de confirmação.

### Passo F: O Ciclo de Revisão

Quando o Dev disser "Pronto para análise" e mover a task para **To Review**, você muda de "Persona" no chat da IDE (ou abre um novo chat na IDE) e roda este prompt:

```
ROLE:
Você agora é o Tech Lead Revisor (Conhecido como "O Chato").

AÇÃO:
1. Olhe o ClickUp, coluna 'To Review'.
2. Para cada task lá:
  - Leia o código que foi alterado nos arquivos.
  - Compare com o que a Task pedia.
  - ANÁLISE DE SEGURANÇA: Há senhas expostas? Há loops infinitos? O código está sujo?

VEREDITO:
- Se estiver RUIM: Mova para 'To Fix'. Escreva um comentário no ClickUp explicando EXATAMENTE o que o Dev deve corrigir. (O Dev vai pegar isso no próximo loop).
- Se estiver BOM: Mova para 'Done'. (Opcional: Faça o commit no git:
```

```
"feat: task X concluída").
```

## **Conclusão: A Autonomia Real**

Percebeu o que fizemos? Montamos uma empresa inteira dentro do seu computador.

Você tem o Visionário (Você/BA), o Tradutor (PO), o Arquiteto, o Operário (Dev) e o Auditor (Revisor). Sua função deixou de ser "escrever código" e passou a ser "gerenciar o fluxo".

Se o Dev erra, o Revisor pega. Se o Arquiteto planeja mal, o Dev trava e te avisa. Este sistema se auto-corrige. É assim que se constrói software sólido no século 21.