MVPFY

MVPFy — Guia do usuário · v0.5.0

Defina o primeiro MVP SaaS

Percurso do usuário para transformar uma ideia em um plano de MVP SaaS.

Guia do usuário
Edição 0.5.0
Gerado em 19/08/2026
Fonte editorial: docs/user/

Sumário

Guia do usuário do MVPFy

Você não precisa chegar com um plano pronto. Basta descrever o que imagina criar, mesmo que a ideia ainda misture público, funcionalidades, preço e tecnologia. O MVPFy lê esse primeiro relato, aproveita o que já está claro e conduz a conversa até uma versão inicial que possa ser construída e testada.

Durante a entrevista, você conversa com a skill `mvpfy`. Ela faz uma pergunta por vez, salva a resposta antes de seguir e chama especialistas para tratar do problema, do público, do produto, do preço, da tecnologia e da entrada no mercado.

No fim, você recebe um arquivo `MVP.md`. Ele dá às equipes uma base comum para criar o website, orientar a marca, construir a versão 1.0, vender o serviço e acompanhar os primeiros clientes.

O MVPFy não cria `spec.md`, não executa o método do Specsfy e não altera o repositório `specsfy`. A semelhança entre os guias existe apenas na organização da documentação e no padrão visual do ebook.

Escolha como começar

Você pode ler este percurso online ou levar o conteúdo para um leitor digital. Esta edição portátil é a **v0.4.7**:

- PDF, para leitura e impressão.
- EPUB, para leitores digitais.
- manifesto, com a versão e os hashes do build.

O ebook contém somente o guia do usuário. A ordem de compilação está em reading-order.txt. A documentação técnica continua disponível online para instalação, manutenção e contribuição com o framework.

Para qual situação o MVPFy serve

No começo de uma empresa, é comum pensar no painel antes de entender o problema, escolher a tecnologia antes de conhecer o volume e imaginar o preço sem saber qual empresa ou pessoa fará o pagamento. O MVPFy organiza essas escolhas em uma conversa simples.

O guia foi feito para este cenário:

- uma empresa nova ou uma operação ainda em formação.
- um primeiro produto de software.
- uma oferta contínua de software como serviço.
- uma versão 1.0 para aprender com os primeiros clientes.

Se a ideia for uma agência que cria sites sob encomenda, o MVPFy verifica se existe também um produto SaaS repetível. Sem esse produto, a proposta não se encaixa no percurso principal.

Um percurso curto e progressivo

1. Leia A proposta para saber o que o MVPFy prepara.
2. Siga Instalação para preparar o projeto que receberá o plano.
3. Acompanhe Primeiro MVP com um exemplo do início ao fim.
4. Consulte Entrevista para entender a conversa e a retomada.
5. Abra MVP.md para conhecer o documento entregue.
6. Use Pesquisa e preço e Tecnologia e operação quando essas partes fizerem sentido para sua ideia.
7. Consulte Progresso para acompanhar as áreas do MVP pela CLI ou TUI.
8. Consulte o catálogo de skills se quiser saber qual especialista trata cada assunto.

Você pode continuar depois

Você pode parar quando quiser. O MVPFy salva as respostas no projeto que você preparou e retoma a conversa depois. Ao dizer “continuar”, ele lê o que já foi registrado, encontra o próximo ponto relevante e mostra somente uma pergunta.

Uma mensagem longa também é bem-vinda. Se você explicar que agências pequenas perdem leads recebidos por WhatsApp, formulário e Instagram, o MVPFy pode usar essa informação para descrever o problema, o público, os canais e a alternativa atual. Você não precisa repetir a mesma resposta em perguntas diferentes.

Próximo passo

Para acompanhar um exemplo concreto, abra Primeiro MVP. Se você já instalou o framework e precisa consultar uma regra, use o índice de skills. O guia de desenvolvimento fica separado para instalação e manutenção da implementação.

O que o MVPFy prepara

O MVPFy foi criado para o primeiro momento de uma empresa. Você tem uma ideia de software, ainda precisa descobrir o recorte comercial e quer chegar a uma versão que possa ser construída, vendida e aprendida com clientes reais.

O plano que sai da conversa

Ao terminar a entrevista, o MVPFy reúne o trabalho em um único arquivo chamado `MVP.md`. O arquivo serve como ponto de encontro para produto, desenvolvimento, website, marca, marketing, vendas e operação.

Você não recebe uma cópia bruta da conversa. O plano separa o que você descreveu, o que uma especialista recomendou, o que ainda precisa ser testado e o que continua sem resposta. Assim, uma equipe consegue ler o contexto sem depender do histórico do chat.

O plano cobre as perguntas que mais afetam a primeira versão:

- a empresa, o contexto e a forma de oferecer o SaaS.
- o problema, a alternativa usada hoje e a hipótese de valor.
- o público, o cliente pagador, o pagador, as pessoas usuárias e as personas.
- a promessa, a jornada central e o limite da versão 1.0.
- o espaço, o onboarding, a ativação, a assinatura, o suporte e o cancelamento.
- o posicionamento inicial, o nome, a marca e o slogan.
- concorrentes, preços observados e alternativas manuais.
- preço, custo operacional, receita recorrente e margem estimada.
- Laravel, VPS, banco, arquivos, e-mail, IA e rotina de operação.
- website, conteúdo, captação, vendas, lançamento, métricas e próximos passos.

O que fica fora deste trabalho

O MVPFy não constrói o software, publica o website ou entrega um logo final. Ele prepara o contexto para que outros agentes e equipes executem essas atividades com menos suposições.

O framework também não cria `spec.md`, não altera o Specsfy e não converte o `MVP.md` em uma especificação do Specsfy. O MVPFy possui seu próprio arquivo, seu próprio estado e seus próprios scripts.

Por que o plano trata o serviço inteiro

Uma lista de funcionalidades não explica como um SaaS funciona. O plano precisa mostrar qual pessoa cria o espaço, qual pessoa usa, qual pessoa paga, como a primeira pessoa chega ao valor e o que acontece com o acesso depois.

Durante a validação, algumas tarefas podem continuar manuais. Uma pessoa pode liberar um espaço, acompanhar a implantação e conferir um pagamento enquanto a equipe aprende com os primeiros clientes. O `MVP.md` registra essa rotina, seu responsável e o momento previsto para automatizá-la.

O limite da versão 1.0

O plano procura manter quatro pontos ligados: um público prioritário, um problema principal, uma promessa central e uma jornada capaz de entregar essa promessa do começo ao fim. Um recurso pode ser útil e ainda assim ficar para depois se não for necessário para testar essa combinação.

Recomendação não é obrigação

O MVPFy pode sugerir Laravel, VPS, Supabase, armazenamento de objetos e um provedor de IA. Essas escolhas partem do padrão da Promovaweb e do contexto fornecido por você. O plano também registra a razão da recomendação e o sinal que justificaria uma revisão.

Instalação do MVPFy

Para usar o MVPFy, tenha Node.js 22 ou superior instalado.

Prepare o projeto

Na raiz do projeto consumidor, execute um único comando. Ele instala as skills e prepara os arquivos locais:

```
npx --yes @promovaweb/mvpfy install --project .
```

O comando cria `MVP.md` e `.mvpfy/` para guardar o plano e o estado necessário para retomar a entrevista, sem alterar o código da aplicação ou os arquivos do Specsfy.

Confira o resultado com os dois comandos abaixo. Eles mostram se o projeto e o planejamento já estão prontos:

```
mvpfy doctor --project .
mvpfy progress --project .
```

Com o projeto preparado, carregue `$mvpfy` no agente e peça para começar. A primeira mensagem livre será registrada antes da primeira pergunta fechada.

Atualize as skills

Na mesma raiz do projeto, execute o comando de atualização. Ele mantém as skills disponíveis para a próxima sessão:

```
mvpfy update --project .
```

Quando já existir um `MVP.md` de uma versão anterior, a atualização compara o template e chama `mvpfy-migrate` antes de continuar. As respostas registradas permanecem disponíveis para a próxima lacuna da entrevista.

Se algo não aparecer no projeto, consulte Solução de problemas. Essa página reúne as verificações para corrigir a instalação.

Primeiro MVP: do relato ao plano

Este exemplo acompanha uma ideia fictícia. Ele não descreve um produto real do MVPFy. A proposta é uma ferramenta para pequenas agências acompanharem leads recebidos por WhatsApp, formulário e Instagram.

Conte a ideia do jeito que ela vier

Você pode começar com uma frase curta:

```
Quero criar um sistema para pequenas agências que perdem leads dos clientes.
```

Também pode enviar tudo o que já imaginou em uma única mensagem:

```
Quero um SaaS para pequenas agências acompanharem leads dos clientes. A
agência paga, o cliente acompanha apenas os próprios contatos e o MVP deve
receber o lead, atribuí-lo a alguém e mostrar um resumo. Já imagino CRM,
integração com WhatsApp e um painel. Quero usar Laravel em uma VPS e cobrar
mensalidade.
```

O agente lê a mensagem inteira antes de perguntar. CRM, WhatsApp e painel entram como ideias para examinar. Problema, público, jornada, tecnologia e cobrança podem receber informações desde o primeiro relato.

Você também pode mandar outra mensagem antes de escolher `4. Avançar`. O MVPFy acolhe esse complemento e só começa a entrevista fechada depois que você indicar que terminou de apresentar a ideia.

O ponto de partida não é transformar cada recurso citado em módulo. Primeiro, o MVPFy tenta entender a situação que faz alguém procurar o produto.

Veja como uma pergunta aparece

Uma conversa possível:

```
MVPFy: Em qual situação o problema aparece com mais frequência?
1. A equipe recebe contatos em vários canais e perde o acompanhamento.
2. A equipe demora para responder ao primeiro contato.
3. A equipe não consegue mostrar resultados para o cliente.
4. Avançar
5. Conversar mais sobre este tema

Você: 1

MVPFy: resposta salva. Qual pessoa usaria o produto no trabalho diário?
1. A equipe da agência.
2. O cliente da agência.
3. A equipe e o cliente, com acessos diferentes.
4. Avançar
5. Conversar mais sobre este tema
```

O texto “resposta salva” representa uma etapa interna obrigatória. A próxima pergunta só aparece depois do registro no `answers.jsonl` e da atualização do estado do projeto.

Se você escolher 5, pode explicar um caso, perguntar o significado de uma opção ou enviar uma resposta complexa. O MVPFy continua conversando sobre o mesmo assunto, salva cada turno e mostra no máximo uma pergunta principal.

Envie mais contexto quando precisar

Você não precisa escolher apenas um número. Esta resposta também é válida:

```
A agência paga. Cada cliente deve acompanhar somente os próprios leads, e a
equipe da agência precisa administrar tudo em um espaço.
```

Essa frase já informa a empresa cliente, a pessoa responsável pelo pagamento, o titular do espaço e as pessoas usuárias, permissões e separação entre clientes. O MVPFy aproveita essas informações e não pergunta novamente qual empresa fará o pagamento.

Transforme a ideia em uma jornada

Depois de entender o problema, a conversa procura um caminho completo para o primeiro valor. No exemplo, o fluxo poderia ser:

1. A agência cria um espaço.
2. A equipe cadastra um cliente.
3. O sistema recebe ou registra um lead.

4. A equipe atribui o lead a alguém.
5. O cliente acompanha o andamento.
6. A agência mostra o resultado em um resumo simples.

O MVPFy pergunta quais dessas etapas precisam funcionar na primeira versão. Um aplicativo móvel, um marketplace e automações avançadas podem ficar para depois se não ajudarem a provar esse caminho.

Explique como o serviço começa

O plano registra a agência como espaço pagadora e limita o acesso de cada cliente aos próprios leads. Também descreve o primeiro valor: registrar um lead, atribuí-lo e acompanhar seu andamento. A conversa ainda trata do teste, da implantação assistida e do sinal que confirma a ativação.

Compare alternativas e monte uma faixa de preço

Se você fornecer URLs de concorrentes, a especialista de mercado consulta os planos, limites e preços publicados e registra a fonte e a data. Sem uma referência direta, a conversa olha para o que a agência gasta hoje com planilhas, horas de acompanhamento e leads perdidos.

A especialista de preço transforma esses dados em uma faixa inicial. Isso é uma hipótese comercial, não uma promessa. O documento pode comparar um cenário mínimo, um cenário base e um cenário de crescimento.

Gere o `MVP.md`

Quando você pedir “gerar o documento”, o MVPFy reúne as áreas já trabalhadas. Uma primeira versão pode ter estado `preliminary` e indicar `Pendente` onde faltarem informações. Depois, com problema, público, jornada, escopo, modelo SaaS, preço, tecnologia, aquisição, métricas e comprovações suficientes, o documento pode receber o estado `ready`.

Use o plano com as equipes

O `MVP.md` permite que as equipes partam do mesmo entendimento:

- produto encontra o escopo e as condições de aceite.
- desenvolvimento encontra arquitetura, espaços e integrações.
- website encontra público, promessa e oferta.
- marca encontra posicionamento, nomes e tom.
- marketing encontra canais, conteúdo e captação.
- vendas encontra preço, processo e objeções.
- operação encontra onboarding, suporte e métricas.

A entrevista, uma pergunta por vez

A entrevista é o lugar onde a ideia ganha forma. Você escreve como falaria com uma pessoa da equipe, e a orquestradora escolhe o próximo ponto que mais ajuda a montar o plano.

Primeiro, conte a ideia inteira

Antes da primeira pergunta com opções, descreva livremente a ideia do SaaS e do MVP. Conte o que o produto pode fazer, qual público ele atenderia e quais módulos, recursos ou integrações já apareceram na sua cabeça. O MVPFy salva esse relato e separa os itens citados para examiná-los depois.

Depois, começa a entrevista fechada. Cada turno tem uma única pergunta e cinco opções. Um item citado não entra automaticamente na versão 1.0. A conversa vai mostrar se ele apoia a jornada principal, se deve esperar ou se não serve ao recorte escolhido.

Você também pode enviar tudo em uma mensagem. Por exemplo:

```
Quero um SaaS para pequenas agências acompanharem leads de seus clientes.
Hoje elas usam planilhas e WhatsApp. A agência paga, cada cliente vê apenas
seus próprios leads e o MVP precisa cadastrar o lead, atribuí-lo a alguém e
mostrar um resumo. Quero começar com Laravel em uma VPS e cobrar mensalidade.
```

Nesse caso, o MVPFy já pode registrar problema, público, cliente pagador, permissões, jornada, tecnologia e cobrança. Ele não precisa perguntar de novo qual empresa ou pessoa fará o pagamento ou repetir etapas que você descreveu. A próxima pergunta trata apenas do ponto importante que ainda falta.

Se quiser acrescentar informações, envie outra mensagem. Todas são acolhidas e salvas. Quando terminar, escolha `4. Avançar`.

O encerramento da entrada inicial pode aparecer assim:

```
MVPFy: Entendi a ideia e já registrei o problema, o público, a jornada e os
itens candidatos. Você quer acrescentar algo antes de começar?
1. Acrescentar outro módulo ou recurso.
2. Explicar melhor o público ou o problema.
3. Informar tecnologia, preço ou canal de venda.
4. Continuar para as perguntas.
5. Conversar mais sobre este tema.
```

As opções 1, 2, 3 e 5 permitem continuar enviando conteúdo. A opção 4 salva o estado da entrada e libera a próxima pergunta fechada.

A primeira pergunta fechada

Depois de salvar a ideia inicial, o MVPFy confirma o modelo de atendimento antes de investigar problema, preço ou tecnologia:

```
O sistema atenderá várias empresas ou equipes separadas dentro da mesma aplicação?
1. Sim. Vários clientes usarão a mesma aplicação, com dados separados.
2. Não. Cada cliente terá uma instalação ou ambiente próprio.
3. Ainda não sei. Quero comparar os dois modelos.
4. Avançar
5. Conversar mais sobre este tema
```

Se você escolher a primeira opção, a conversa pode fazer uma única pergunta curta sobre a unidade do tenant e a pessoa administradora. Os detalhes de membros, papéis, acesso entre espaços, separação de dados, banco e criação de novos tenants viram recomendações técnicas. Assim, o MVP não vira um cadastro interminável de regras de acesso.

Se ainda não souber, a opção 5 permite conversar sobre a diferença antes de registrar a escolha. A pergunta continua sendo uma por vez.

Quantas perguntas serão feitas

A entrevista fechada tem no máximo oito perguntas. Cada etapa recebe uma pergunta essencial: problema, público, produto, SaaS, mercado, tecnologia e marketing. A etapa SaaS pode receber uma segunda pergunta curta. Se a ideia inicial já respondeu uma etapa, ela é pulada. O que permanecer será registrado como recomendação ou hipótese no `MVP.md`.

Depois da oitava resposta, o MVPFy encerra a entrevista normal e monta o documento. Você pode pedir uma revisão específica depois, mas o fluxo inicial não continua perguntando sem fim.

O que acontece depois da sua resposta

1. Você responde com número, texto, combinação de opções ou “não sei”.
2. O MVPFy guarda sua mensagem original.
3. A resposta recebe uma leitura organizada, sem mudança de sentido.
4. O sistema registra escolhas, hipóteses e áreas já cobertas.
5. `state.json` e `MVP.md` são atualizados quando fizer sentido.
6. A orquestradora revê o que ainda falta.
7. Uma única pergunta seguinte aparece.

Se a gravação falhar, o fluxo não avança. A mesma pergunta volta para que você possa tentar novamente.

O formato de cada turno

Cada pergunta aparece com cinco opções:

```
Qual situação descreve melhor o problema?
1. A equipe perde o acompanhamento.
2. A equipe demora para responder.
3. A equipe não consegue mostrar o resultado.
4. Avançar
5. Conversar mais sobre este tema
```

As três primeiras opções respondem à mesma pergunta. A opção 4 aceita o entendimento atual e leva à próxima etapa. A opção 5 abre uma conversa sobre o mesmo tema. Você pode explicar um caso, tirar uma dúvida ou enviar uma resposta complexa. O MVPFy pode continuar essa conversa por mais de um turno, sempre salvando cada mensagem e mostrando no máximo uma pergunta principal.

Nunca aparecem duas perguntas principais no mesmo turno.

Você pode responder do seu jeito

Você pode responder:

- `1`, `2` ou `3`.
- `4` para avançar.
- `5` para conversar mais sobre a pergunta atual.
- “2, mas também atendo consultores”.
- uma explicação livre.
- “ainda não sei”.
- uma correção de resposta anterior.
- “pausar”, “continuar”, “revisar preço” ou “gerar documento”.

“Ainda não sei” também ajuda. O MVPFy registra a lacuna e pode sugerir uma hipótese provisória, deixando claro que ela ainda precisa de comprovação.

Peça uma ação quando precisar

PEDIDO	RESULTADO
“Começar”	Cria ou abre um projeto e faz a primeira pergunta útil.
“Continuar”	Lê o estado e retoma pela pendência prioritária.

PEDIDO	RESULTADO
“Pausar”	Salva o ponto atual sem apagar respostas.
“Revisar preço”	Trabalha somente em modelo, faixa e premissas de preço.
“Mudar o público”	Registra a correção e revisa áreas dependentes.
“O que falta?”	Mostra um resumo das pendências sem abrir formulário.
“Gerar documento”	Consolida a melhor versão, mesmo que preliminar.

Como o fluxo aproveita o que você já disse

Cada resposta recebe uma lista de áreas cobertas. Uma mensagem como “a agência paga e o cliente acompanha” já descreve papéis comerciais e de produto. A orquestradora usa essa informação para não colocar perguntas repetidas na fila.

Ela também confere mudanças de direção. Se antes você disser que o cliente pagador é uma clínica e depois informar que venderá para agências, a nova mensagem fica registrada como correção. Personas, preço, jornada e canais ligados ao público anterior voltam para revisão.

Quando a ideia ficou grande demais

Quando aparecem várias jornadas, muitos públicos ou módulos que parecem produtos separados, o MVPFy pede um recorte. A pergunta passa a ser:

```
Qual resultado precisa funcionar do começo ao fim na primeira versão?
1. Registrar e acompanhar o lead.
2. Gerar o relatório para o cliente.
3. Automatizar a distribuição do lead.
```

As outras ideias são preservadas em “Fora do MVP e evolução futura”.

Acompanhe o progresso do MVP

O arquivo `MVP.md` cresce enquanto a entrevista registra respostas e recomendações. A skill `mvpfy-progress` apresenta esse estado sem abrir uma pergunta nova. Ela pode ser usada quando você quiser saber o que já está claro e qual assunto ainda precisa de atenção.

Pelo terminal

Na raiz do projeto consumidor:

```
mvpfy progress --project .
```

O resultado agrupa o plano em Problema, Público, Produto, SaaS, Mercado e preço, Tecnologia, Marketing e Validação. O percentual é uma indicação de preenchimento das seções encontradas no `MVP.md`; ele não substitui a leitura das hipóteses e dos pontos ainda abertos.

Para abrir o painel:

```
mvpfy --project .
```

Na aba **Home**, você vê o resumo geral, as áreas e o próximo ponto. Na aba **Áreas**, pode navegar por cada grupo e ver quais seções estão completas. Na aba **MVP.md**, o documento é renderizado com cores e pode ser percorrido com as setas do terminal. Essa leitura não altera o arquivo.

Instalação e atualização no painel

A aba **Skills** oferece duas ações:

- `I` instala ou reconcilia as skills do MVPFy;
- `R` atualiza as skills já instaladas.

As ações executam `npx skills add promovaweb/mvpfy --skill '*' --agent claude-code codex --copy --yes` e `npx skills update --project --yes`. A instalação é feita somente para Claude Code e Codex. O MVPFy não substitui o CLI oficial nem cria uma cópia paralela da instalação.

Como interpretar o resumo

```
Progresso do MVP: 48%
✓ Problema: 100% (2/2)
◖ Produto: 60% (3/5)
○ Marketing: 0% (0/3)
Próximo ponto: Preencher a seção marketing
```

Uma área concluída tem todas as seções encontradas preenchidas. Uma área em andamento possui parte do conteúdo definido. Uma área pendente ainda não tem informação suficiente no documento. O modelo de atendimento SaaS aparece à parte porque a escolha entre multitenante compartilhado e instalação separada afeta várias áreas.

Sem TUI

Para agentes e automações, use JSON:

```
mvpfy progress --project . --json
```

Esse formato inclui as áreas, seções pendentes, estado da entrevista, modelo SaaS e próximo ponto. O comando é somente de leitura.

O plano que você recebe: `MVP.md`

`MVP.md` é a entrega central do MVPFy. Ele transforma a conversa em um plano que pode circular entre produto, desenvolvimento, marketing, vendas e operação sem depender do histórico do chat.

Como ler o estado do plano

ESTADO	SIGNIFICADO
`preliminary`	Há conteúdo aproveitável, mas faltam escolhas importantes.
`ready`	Os campos mínimos do primeiro SaaS estão descritos e coerentes.
`validated`	As hipóteses principais já foram conferidas fora da entrevista.

O estado não funciona como uma nota. Ele mostra quanto do plano já tem base suficiente para orientar o próximo trabalho.

O que o plano reúne

O template atual reúne 35 áreas. Elas acompanham o caminho natural de uma empresa que sai da ideia e chega à primeira operação SaaS.

Primeiro, o plano explica o ponto de partida:

- resumo executivo.
- empresa e contexto.
- modelo da empresa SaaS.
- problema.
- comprovação, alternativas e hipóteses.
- público-alvo.
- pessoa usuária, cliente pagador e pagador.
- personas.
- proposta de valor e posicionamento.
- nome, marca e slogan.

Depois, ele descreve o produto e a relação com cada cliente:

- jornada principal.
- modelo multitenante, espaço, acesso e separação de dados.
- onboarding e primeiro valor.
- escopo funcional da versão 1.0.

- módulos e funcionalidades.
- perfis e permissões.
- assinatura e cobrança.
- suporte, retenção e cancelamento.
- processos manuais.
- itens fora do MVP.

Por fim, o plano mostra como a empresa pode chegar ao mercado e operar:

- concorrentes e alternativas.
- modelo comercial e preço.
- custos e economia unitária.
- tecnologia e arquitetura.
- infraestrutura e operação.
- uso de IA.
- website.
- marketing.
- vendas e lançamento.
- métricas.
- validações e dependências.
- execução.
- escolhas confirmadas.
- hipóteses e pendências.
- fontes.

Cada seção possui um comentário estável, como `<!-- mvpfy:section:problem-definition -->`. Se o nome visível mudar, o renderer ainda encontra o conteúdo pelo identificador.

O que está confirmado e o que ainda falta

O documento usa quatro rótulos para que uma sugestão não pareça uma certeza:

- **Confirmado:** informação declarada por você ou respaldada por uma fonte.
- **Recomendado:** sugestão de uma especialista, baseada no contexto disponível.
- **Hipótese:** afirmação ainda não conferida com pessoas, mercado ou uso.
- **Pendente:** informação necessária que ainda não foi fornecida.

Essa separação impede que uma estimativa de preço pareça uma pesquisa concluída ou que uma sugestão de Laravel pareça uma exigência do produto.

Como o plano evolui

Quando o template evolui, `mvpfy-migrate` insere as seções novas, conserva o texto existente e marca os campos que precisam de resposta. Depois da atualização, a orquestradora continua pela primeira pergunta relevante, sempre uma por vez.

Mercado, preço e custo

O MVPFy pesquisa o mercado quando você informa concorrentes, envia URLs ou solicita uma referência atual. A pesquisa ajuda a montar uma hipótese de preço, mas não substitui a conversa com potenciais clientes.

Quando você tem concorrentes para comparar

Informe uma URL ou um nome. A skill `mvpfy-market` procura, quando os dados estão publicados:

- o público e a promessa.
- os recursos úteis para a comparação.
- o preço mensal, anual ou sob consulta.
- a moeda, os limites e as condições do plano.
- a diferença entre concorrente direto, indireto e alternativa manual.

O `MVP.md` registra fonte e data. Se o preço não estiver publicado, o plano escreve "não publicado". O MVPFy não inventa uma faixa nem apresenta uma promoção como preço permanente.

Quando não há concorrente direto

Nesse caso, a conversa olha para o trabalho feito hoje. As perguntas continuam aparecendo uma por vez, com três respostas prontas, `Avançar` e `Conversar mais sobre este tema`. A investigação pode tratar de:

- tempo gasto por mês.
- ferramentas combinadas.
- erros, perdas ou atrasos.
- resultado que traria benefício suficiente para justificar o pagamento.

Essas respostas ajudam a aproximar o valor percebido e escolher uma unidade de cobrança que seja fácil de explicar.

Como surge a faixa de preço

`mvpfy-pricing` considera cliente pagador, unidade de valor, frequência de uso, benefício, esforço comercial, custo variável, suporte e estágio da validação. Com esses dados, pode recomendar cobrança por espaço, por pessoa, por volume, por uso ou um modelo combinado.

Quando houver dados suficientes, o plano compara três cenários:

CENÁRIO	USO
Mínimo	Poucos clientes, infraestrutura mínima e operação acompanhada de perto.
Base	Volume inicial esperado e custos recorrentes conhecidos.
Crescimento	Aumento de uso sem acrescentar complexidade antes de existir necessidade.

As espaços conceituais são:

```
custo mensal = fixos + variáveis + IA + comunicação + suporte + taxas
margem por cliente = receita líquida - custo variável por cliente
clientes para equilíbrio = custos fixos / margem por cliente
MRR = soma da receita recorrente mensal dos clientes ativos
ARPA = MRR / espaços pagantes
```

O resultado é uma faixa acompanhada das suas premissas. Não é previsão contábil e não garante receita.

Tecnologia e operação para começar

O ponto de partida da Promovaweb para os projetos atendidos pelo MVPFy é Laravel em uma VPS. Essa combinação mantém a aplicação simples, torna o custo mais previsível e encurta o caminho até a primeira validação.

Componentes de referência

NECESSIDADE	PADRÃO INICIAL
Aplicação	Laravel.
Serviço complementar	Node.js apenas quando uma necessidade concreta justificar seu uso.
Banco	PostgreSQL, com Supabase quando fizer sentido para o projeto.
Servidor	VPS com ambientes e cópias de segurança definidos.
Arquivos	Object storage compatível com S3.
Mensagens	Provedor de e-mail transacional.
IA	Provedor externo apenas quando houver uma função clara no produto.
Operação	Logs, cópias de segurança, filas ou tarefas agendadas conforme a jornada.

Outra opção pode fazer sentido quando o contexto exigir. Nesse caso, a especialista explica o motivo e mostra o efeito no custo e na operação.

O que precisa estar claro na primeira versão

O plano não precisa desenhar cada classe do sistema. Ele precisa responder às perguntas que afetam a jornada e a operação:

- como o espaço é criado e identificado.
- se o produto é multitenante ou usa instalação separada por cliente.
- como os dados de clientes ficam separados.
- quais pessoas acessam cada parte.
- quais integrações são indispensáveis.
- quais tarefas podem rodar fora da tela.
- onde os arquivos ficam guardados.
- como funcionam cópias de segurança, logs e suporte.
- qual custo mensal aparece em cada cenário.

Para a maioria dos primeiros SaaS, o MVPFy prefere uma aplicação compartilhada com separação lógica segura entre espaços. Uma instância dedicada só entra quando contrato, segurança, desempenho ou posicionamento exigirem esse custo.

Quando Laravel for escolhido para um SaaS multitenante, o plano avalia o suporte de Teams da solução adotada para representar equipes, membros, convites e papéis. Essa base organiza a associação das pessoas, mas cada consulta ainda precisa respeitar o tenant ativo e sua separação de dados.

Comece com operação acompanhada

Durante a validação, uma pessoa pode liberar espaços, acompanhar o onboarding e conferir assinaturas manualmente. O `MVP.md` registra essa rotina, seu responsável e o sinal que indicará a hora de automatizar o trabalho.

Quando algo não funcionar

O estado não aparece depois da instalação

Confirme que o comando foi executado na raiz do projeto que receberá o plano:

```
npx --yes @promovaweb/mvpfy install --project .
```

Depois, confira se existem `MVP.md` e `.mvpfy/state.json`. O comando deve rodar no projeto consumidor. Executá-lo na raiz do repositório `mvpfy` tenta preparar o próprio pacote, não o projeto da entrevista.

A pergunta repetiu algo que você já informou

Confira se a mensagem anterior foi salva no `answers.jsonl` e se as áreas extraídas chegaram ao `state.json`. Uma resposta livre pode precisar de uma leitura adicional. A ausência de um número não torna a mensagem inválida.

O documento continua como `preliminary`

Abra a seção “Hipóteses e pendências”. Esse estado informa que ainda falta um item necessário para descrever o primeiro SaaS. Peça “continuar” ou “o que falta?” para voltar ao próximo ponto relevante.

Você mudou uma escolha anterior

Diga a alteração com clareza, por exemplo: “o público agora são clínicas, não agências”. O MVPFy conserva a mensagem anterior no histórico, registra a nova e revisa as áreas afetadas.

O ebook não mostra a versão atual

No repositório `mvpfy`, execute:

```
npm run ebook
npm run ebook:verify
```

Se uma página não aparecer, confira o caminho correspondente em `docs/user/reading-order.txt`. Essa lista deve conter apenas o guia do usuário.

As especialistas que trabalham no MVPFy

Você conversa apenas com `$mvpfy`. A orquestradora lê o que já foi registrado, escolhe a especialista adequada e incorpora o retorno ao plano. Você não precisa chamar cada skill manualmente. Conhecer o papel de cada uma ajuda a entender por que determinada pergunta apareceu.

Ordem típica

```
contexto existente
      ↓
problema → público → produto → SaaS
      ↓          ↓         ↓       ↓
mercado → preço → tecnologia → marketing
      ↓
marca → MVP.md → migração quando o template mudar
```

Essa ordem é apenas um caminho comum. Um documento já existente pode preencher uma área inteira, e uma resposta nova pode levar a conversa de volta para um assunto anterior. A orquestradora acompanha o conteúdo disponível, não uma lista fixa de perguntas.

Skills

SKILL	QUANDO ENTRA	O QUE ACRESCENTA AO PLANO
`mvpfy`	Sempre que você inicia ou retoma	Próxima pergunta e estado integrado
`mvpfy-context`	Setup e início de cada conversa	Specs, código, stack e lacunas disponíveis
`mvpfy-problem`	O problema ainda está genérico	Declaração do problema e hipótese
`mvpfy-audience`	Público ou papéis estão misturados	Segmento, ICP e personas
`mvpfy-product`	A solução precisa virar escopo	Jornada, módulos e versão 1.0
`mvpfy-saas`	É preciso explicar o serviço recorrente	Espaço do cliente, onboarding e ciclo do cliente
`mvpfy-brand`	Produto e público já têm direção	Nome, posicionamento e slogan
`mvpfy-market`	Há concorrentes ou URLs	Comparação de mercado e fontes
`mvpfy-pricing`	Valor e cobrança precisam de hipótese	Faixas, planos e cenários

SKILL	QUANDO ENTRA	O QUE ACRESCENTA AO PLANO
`mvpfy-technology`	A jornada permite desenhar a base técnica	Arquitetura Laravel e custos
`mvpfy-marketing`	Oferta e público precisam chegar ao mercado	Aquisição, conteúdo e venda
`mvpfy-document`	Você pede o plano ou há dados suficientes	`MVP.md` renderizado e validado
`mvpfy-migrate`	O template mudou	Documento atualizado sem perda
`mvpfy-progress`	Você quer ver o andamento sem abrir uma pergunta	Resumo por áreas e próxima lacuna

Cada página explica o que a especialista recebe, como trabalha, o que entrega e onde termina sua responsabilidade. Os exemplos usam uma ideia de SaaS para mostrar a passagem de uma etapa para outra.

O catálogo descreve o MVPFy. Ele não altera a biblioteca Specsfy.

A orquestradora: `mvpfy`

`mvpfy` é a orquestradora e o único ponto de conversa. Antes de escolher a primeira pergunta, ela chama `mvpfy-context` para verificar se o projeto já tem material aproveitável. Depois ela não tenta saber tudo sobre produto, preço ou tecnologia. Em vez disso, lê o contexto existente, chama a especialista certa, controla a entrevista e reúne o resultado no `MVP.md`.

Quando usar

Use `$mvpfy` para começar, continuar, pausar, corrigir uma escolha, revisar uma área ou gerar o `MVP.md`.

De onde ela parte

- `.mvpfy/existing-project.json`, renovado no setup e no início da conversa.
- `.mvpfy/config.yaml`.
- `.mvpfy/state.json`.
- `.mvpfy/answers.jsonl`.
- `MVP.md`.
- template atual.
- specs, backlogs, briefs, docs, tickets e planos já existentes no projeto consumidor, sempre em modo somente leitura, resumidos pela `mvpfy-context` e consultados quando necessário.

Esses arquivos servem como referência. A orquestradora pode usar uma informação já registrada para evitar uma pergunta, mas não escreve nem corrige esses documentos.

Quando o relatório encontra manifestos e código, ele pode sugerir a stack e confirmar que existe uma base técnica. Quando encontra uma spec, ele aponta o arquivo e um trecho curto para orientar a leitura. O código sozinho não define o problema, o público, o pagador ou o recorte do MVP.

Como ela conduz a conversa

1. Confere se o projeto é um primeiro SaaS.
2. Reaproveita o material já disponível.
3. Verifica se o template precisa de atualização.
4. Escolhe o ponto que mais afeta o plano.
5. Exibe exatamente uma pergunta principal.
6. Registra a resposta antes de prosseguir.

7. Recalcula as áreas cobertas e as mudanças de direção.
8. Chama a especialista adequada.

Regra rígida de saída

Uma resposta pode conter confirmação, contexto curto e opções, mas somente uma pergunta que pede resposta. Ela sempre termina com três opções prontas, `4. Avançar` e `5. Conversar mais sobre este tema`. Nunca apresente um formulário, duas perguntas encadeadas ou "responda A e B". Se duas áreas estiverem pendentes, escolha a mais importante e deixe a outra para o próximo turno.

`Avançar` encerra a etapa com o entendimento atual. `Conversar mais sobre este tema` abre texto livre sobre a mesma pergunta. A conversa pode continuar por mais de uma mensagem, mas nunca apresenta uma segunda pergunta principal no mesmo turno.

Limite sobre fontes externas ao MVPFy

Specs, backlogs e documentos do projeto podem ser lidos como contexto. O MVPFy não cria, altera, renomeia, remove ou migra esses arquivos. O único documento final que ele escreve é `MVP.md`, além do estado em `.mvpfy/`.

Regra para a documentação e o ebook

O guia do usuário e a referência técnica são percursos diferentes. A orquestradora pode consultar a documentação técnica para entender o produto, mas o ebook do usuário usa somente as páginas listadas em `docs/user/reading-order.txt`.

Antes de publicar uma nova edição, leia as páginas completas. Confirme que os exemplos ajudam uma pessoa leiga, que a prosa explica a razão das orientações e que listas e tabelas não substituem o raciocínio. O padrão visual dos projetos do Hub serve como referência de organização, não como texto para copiar.

O contexto existente: `mvpfy-context`

`mvpfy-context` procura material que já existe no projeto antes de a conversa começar. Assim, um projeto vazio segue pelo relato livre da ideia, enquanto uma aplicação com specs e código oferece um ponto de partida concreto.

Quando ela entra

A análise acontece em dois momentos:

- durante `mvpfy install` ou `setup-project.mjs`;
- no início de cada conversa iniciada ou retomada com `$mvpfy`.

Para executar a leitura manualmente, use:

```
node skills/mvpfy-context/scripts/analyze-existing-project.mjs --project .
```

O arquivo `.mvpfy/existing-project.json` recebe o relatório atual. O `state.json` guarda apenas o resumo necessário para a orquestradora localizar o relatório sem aumentar o histórico da conversa.

O que ela procura

O scanner percorre documentos com nomes e pastas usados com frequência em projetos, como `spec.md`, `specs/`, `backlog/`, `briefs/`, `plans/`, `docs/`, `product/` e `decisions/`. Também lê manifestos como `package.json`, `composer.json`, `pyproject.toml`, `go.mod` e `Cargo.toml`.

Os arquivos de programação entram no relatório com caminho, linguagem e número de linhas. Dependências e scripts dos manifestos ajudam a reconhecer a stack. Pastas geradas, dependências instaladas, caches, cobertura, builds e o repositório Git ficam fora da leitura.

Como a conversa aproveita o resultado

Imagine um projeto que já tem `specs/checkout.md`, `composer.json` com Laravel e uma pasta `app/`. A análise pode sugerir que existe uma base Laravel e apontar a spec do checkout. A orquestradora usa esse material para evitar uma pergunta repetida sobre a stack e começa a conversa perguntando pelo problema, pelo público ou pelo recorte que ainda não aparece nos arquivos.

O relatório separa sugestão técnica de escolha confirmada. Uma rota chamada `/customers` mostra que existe uma implementação relacionada a clientes, mas não confirma quem paga, qual dor levou ao projeto nem quais recursos entram na versão 1.0. Esses pontos continuam na conversa e aparecem como pendências até que a pessoa os confirme.

Proteção do projeto analisado

O MVPFy não altera as specs, o código, os arquivos do Specsfy nem os documentos encontrados. A única escrita acontece em `.mvpfy/`, para guardar o relatório e seu resumo. O conteúdo do relatório fica local ao projeto e não é enviado para um serviço externo pelo script.

Setup do MVPFy: `$mvpfy-setup`

Use `$mvpfy-setup` para instalar as skills, criar os arquivos do planejamento e conferir se o projeto pode iniciar ou retomar a entrevista.

Um comando para começar

Na raiz do projeto consumidor, execute:

```
npx --yes @promovaweb/mvpfy install --project .
```

O comando instala as skills e prepara `MVP.md` e `.mvpfy/`. Se esses arquivos já existirem, ele preserva as respostas e a estrutura atual.

Conferência e atualização

Use os comandos abaixo para verificar o resultado ou atualizar as skills:

```
mvpfy doctor --project .
mvpfy progress --project .
mvpfy update --project .
```

O setup não cria nem modifica arquivos do Specsfy. O resultado do MVPFy fica somente em `MVP.md` e `.mvpfy/`.

O problema: `mvpfy-problem`

Esta especialista ajuda a tirar a conversa do nome da solução. “Preciso de um aplicativo” pode ser o começo do relato, mas o plano precisa explicar o que acontece hoje, qual grupo enfrenta a situação e qual consequência torna a mudança valiosa.

O que ela procura

- situação e frequência.
- pessoa afetada.
- tarefa que ela tenta realizar.
- impacto atual.
- alternativa usada hoje.
- motivo da insuficiência.
- comprovações já disponíveis.
- hipótese que o MVP deve testar.

O que entra no `MVP.md`

> `Para [público], que precisa [tarefa], o problema é [dificuldade], causando [impacto]. Hoje usa [alternativa], que falha porque [limitação]. O MVP deve testar [hipótese].`

Um exemplo

“Agências perdem leads” ainda não explica a frequência, a consequência nem a alternativa atual. Se o contexto já disser que os contatos chegam por três canais e terminam em planilhas, a especialista aproveita esses dados. Ela pergunta somente o ponto que ainda muda a leitura do problema.

Onde termina este trabalho

Esta especialista não escolhe framework, cria módulos ou define preço. Quando o grupo afetado estiver claro, ela entrega o contexto para `mvpfy-audience`.

O público: `mvpfy-audience`

Esta especialista separa o mercado amplo do primeiro grupo que vale atender. Também diferencia a pessoa que usa o produto, a pessoa que compra, a pessoa responsável pelo pagamento, o contato que influencia a compra e o grupo que recebe o benefício. Essa distinção evita que "qualquer empresa" vire o público inteiro do MVP.

O que ela procura

- frequência e intensidade do problema por segmento.
- acesso ao público.
- capacidade de pagar.
- comportamento atual.
- objeções e gatilhos.
- canais de comunicação.
- diferenças entre uso e compra.

O que entra no `MVP.md`

O `MVP.md` recebe um segmento prioritário e, no máximo, três personas úteis: produto, compra e marketing quando forem diferentes. Cada uma descreve contexto, objetivo, dor, alternativa, objeção, gatilho e canal. A intenção não é criar biografias, e sim dar à equipe elementos para construir e vender.

Exemplo

Na ferramenta para agências, a agência pode ser compradora e pagadora, a pessoa de atendimento pode ser usuária diária e o cliente da agência pode ser um usuário com permissão limitada. Essa distinção afeta produto, SaaS, preço e marketing.

Onde termina este trabalho

Esta especialista não inventa biografias para preencher espaço nem mantém três públicos prioritários ao mesmo tempo. Quando houver um segmento e um resultado central, ela entrega o contexto para `mvpfy-product`.

O produto: `mvpfy-product`

Esta especialista transforma o problema e o público em uma primeira versão que alguém consegue usar e comprar. Ela começa pela jornada principal. A lista de recursos só entra depois, quando fica claro o trabalho que cada item apoia.

O que ela procura

- promessa central.
- entrada, processamento e saída.
- jornada de ponta a ponta.
- evento de ativação.
- módulos essenciais.
- perfis e permissões.
- regras de negócio.
- integrações indispensáveis.
- itens posteriores.

Como separa o escopo

CLASSE	USO
Essencial	Sem o item, a promessa não é entregue.
Necessária	Apoia operação, segurança ou cobrança inicial.
Posterior	Pode esperar aprendizado.
Descartada	Não atende ao problema prioritário.

Um exemplo

No caso dos leads, registrar um contato, atribuí-lo e mostrar seu andamento podem formar a jornada principal. Um aplicativo móvel e um marketplace podem ser úteis no futuro, mas não entram apenas por parecerem interessantes.

Onde termina este trabalho

Esta especialista não define sozinha qual pessoa paga ou como o espaço começa. Ela encaminha essas perguntas para `mvpfy-saas` e depois ajusta o escopo com as restrições recebidas.

O serviço recorrente: `mvpfy-saas`

Esta especialista explica o que acontece ao redor do software. Um SaaS não termina quando a pessoa entra na tela principal. Uma pessoa precisa criar o espaço, usar o serviço, pagar, receber suporte e encerrar o acesso quando necessário.

Ela começa confirmando se várias empresas ou equipes usarão a mesma aplicação. Essa pergunta vem logo depois da ideia inicial. Se a resposta for multitenante, a especialista faz no máximo uma pergunta curta sobre a unidade do espaço e sua administração antes de tratar preço ou arquitetura.

O que ela procura

- B2B, B2C ou profissionais independentes.
- titular do espaço e pessoas usuárias.
- separação dos dados entre espaços.
- convite, demonstração ou teste.
- onboarding e primeiro valor.
- ativação.
- unidade de valor.
- assinatura, limites e excedentes.
- cancelamento, inadimplência e suporte.
- processos manuais aceitáveis durante a validação.

Com essa resposta, ela recomenda:

- unidade do tenant.
- titularidade e administração.
- convites, membros e papéis.
- participação de uma pessoa em vários tenants.
- limite dos dados e banco.
- criação e configuração de novos tenants.

Esses itens não abrem uma fila adicional. Eles aparecem como recomendações para o MVP, usando o suporte de Teams do Laravel quando fizer sentido.

Um exemplo

Uma agência pode criar o espaço, adicionar seus clientes e manter a cobrança no próprio contrato. Cada cliente acompanha seus leads, sem acesso aos dados de outros espaços. Esse arranjo muda permissões, preço e onboarding.

Quando Laravel for escolhido, o plano avalia o suporte de Teams da solução adotada para equipes, membros, convites e papéis. Teams ajuda na associação de pessoas ao tenant, mas não substitui as regras de isolamento e autorização.

Ponto de partida técnico

Para muitos primeiros produtos, uma aplicação compartilhada com separação lógica segura atende bem. Uma instância dedicada só deve aparecer quando segurança, contrato, desempenho ou posicionamento justificarem esse desenho.

Onde termina este trabalho

Esta especialista não presume que a cobrança precisa ser totalmente automática no primeiro dia. Se uma operação acompanhada ajudar a validar a oferta, o plano pode registrá-la como parte provisória do serviço.

A marca: `mvpfy-brand`

Esta especialista prepara a direção da marca depois que problema, público e promessa já têm uma base. O trabalho ajuda a escolher uma linguagem para o produto e orientar naming e comunicação. Disponibilidade de domínio ou registro de marca só entra como fato depois de uma pesquisa própria.

O que entra no plano

- categoria.
- posicionamento em uma frase.
- benefício central.
- diferenciais.
- personalidade e tom.
- palavras a usar e evitar.
- regras para o nome.
- sugestões de nome quando solicitadas.
- slogans curtos.
- direção visual conceitual.

Um exemplo

Se o valor for dar visibilidade aos leads para agências pequenas, o posicionamento deve falar de acompanhamento e resultado. A marca não precisa depender de uma promessa genérica de inteligência artificial.

Onde termina este trabalho

Esta especialista não cria o logo final, não confirma disponibilidade sem pesquisa e não usa o nome para encobrir um problema ainda indefinido.

O mercado: `mvpfy-market`

Esta especialista pesquisa concorrentes e alternativas quando você fornece referências ou pede dados atuais. Ela registra o que encontrou com fonte e data, separando informação publicada de interpretação.

O que ela compara

- concorrente direto, indireto e alternativa manual.
- público e promessa.
- recursos comparáveis.
- preço, moeda, periodicidade e limites.
- lacuna percebida.
- condições promocionais ou preço sob consulta.

O que entra no `MVP.md`

Uma tabela no `MVP.md` com referência, tipo, público, promessa, recursos, preço, lacuna, link e data da consulta.

Um exemplo

Se uma página mostra “US$ 49 por mês para cinco usuários”, o plano registra o preço, a moeda e o limite. Se a página só diz “fale com vendas”, o documento registra preço não publicado. A falta do preço também é um dado sobre a forma de venda.

Onde termina este trabalho

Esta especialista não inventa preço, não trata uma página antiga como atual e não coloca um recurso no MVP apenas porque um concorrente o oferece.

O preço: `mvpfy-pricing`

Esta especialista transforma valor, custo e comportamento de compra em uma hipótese comercial que possa ser testada. Ela procura uma cobrança fácil de explicar e ligada ao valor percebido, sem exigir medições que o primeiro produto ainda não precisa fazer.

O que ela procura

- cliente pagador e pagador.
- unidade de valor.
- ticket imaginado.
- preço de alternativas.
- benefício financeiro ou operacional.
- custo fixo e variável.
- suporte e esforço de venda.
- implantação, teste e inadimplência.
- MRR, ARPA e clientes para equilíbrio.

O que entra no plano

O plano apresenta faixa de preço, premissas, modelo de cobrança e cenários mínimo, base e crescimento. Quando os dados são insuficientes, a especialista marca a faixa como hipótese.

Um exemplo

Se o valor surge da gestão de vários clientes, cobrar por espaço da agência pode ser mais simples do que cobrar por cada lead. Essa é uma sugestão para testar com potenciais clientes, não uma regra universal.

Onde termina este trabalho

Esta especialista não oferece precisão contábil, não promete margem e não fixa o preço sem considerar custo operacional e disposição de pagamento.

A tecnologia: `mvpfy-technology`

Esta especialista traduz a jornada do MVP em uma arquitetura pequena, operável e compatível com o momento da empresa. A tecnologia deve apoiar o primeiro aprendizado sem criar uma operação maior do que o produto precisa.

Ponto de partida recomendado

- Laravel como aplicação principal.
- Node.js somente com justificativa concreta.
- PostgreSQL, com Supabase quando fizer sentido.
- VPS.
- armazenamento compatível com S3.
- e-mail transacional.
- IA externa apenas com função de produto clara.
- backups, logs e tarefas agendadas conforme necessidade.

O que entra no `MVP.md`

O `MVP.md` recebe arquitetura, dados conceituais, autenticação, papéis, espaços, isolamento, integrações, filas, arquivos, segurança básica, observabilidade e custo mensal por cenário.

Um exemplo

Para a ferramenta de leads, a arquitetura precisa separar agências e clientes, registrar mudanças de status e enviar notificações. Microsserviços e Kubernetes podem esperar até existir uma necessidade real para esse custo.

Onde termina este trabalho

Esta especialista não implementa o software, não compra a VPS e não transforma a stack padrão em requisito mais importante do que o problema do produto.

A entrada no mercado: `mvpfy-marketing`

Esta especialista prepara a chegada do primeiro SaaS ao mercado. Ela conecta promessa, público, oferta, conteúdo, captação e venda em poucos canais que a empresa consiga manter na rotina.

O que entra no plano

- mensagem principal.
- objeções e respostas.
- oferta inicial.
- website mínimo.
- canal prioritário.
- conteúdo por etapa.
- lead magnet quando fizer sentido.
- e-mail marketing.
- grupos e redes sociais.
- demonstração ou venda assistida.
- métricas de aquisição, ativação, conversão e retenção.

Um exemplo

Para agências, o primeiro conteúdo pode mostrar o caminho de um lead desde o contato até o retorno ao cliente. O canal só entra no plano se as agências estiverem presentes nele e a equipe conseguir manter essa rotina.

Onde termina este trabalho

Esta especialista não recomenda todos os canais, não cria uma campanha final e não define a mensagem sem usar o público e a proposta registrados antes.

O documento final: `mvpfy-document`

Esta skill monta e confere o arquivo final. Ela não cria o conteúdo dos domínios. Recebe fatos, escolhas, recomendações, hipóteses e pendências da orquestradora e os organiza no template do `MVP.md`.

Arquivos que ela usa

- `assets/MVP.template.md`.
- `scripts/render-company.mjs`.
- `scripts/validate-company.mjs`.
- `references/document-rules.md`.

Como ela é executada

```
node skills/mvpfy-document/scripts/render-company.mjs --project .
node skills/mvpfy-document/scripts/validate-company.mjs --project .
```

O renderer localiza os identificadores estáveis, mantém o conteúdo existente e usa “Pendente” quando ainda não há informação. O validator confere se as áreas necessárias para um primeiro SaaS aparecem no arquivo.

Quando o `MVP.md` alimentar um ebook, a ordem deve incluir somente o guia do usuário. A especificação e a referência de desenvolvimento ficam fora do PDF e do EPUB. Depois do build, leia o resultado completo para conferir voz, exemplos, ritmo e separação de públicos. Os validadores confirmam estrutura e integridade do arquivo, mas não substituem a leitura editorial.

O PDF, o EPUB e o manifesto usam a mesma versão do framework. `VERSION` é a fonte canônica e `ebooks/VERSION` funciona apenas como espelho de conferência.

Onde termina este trabalho

Esta skill não cria `spec.md`, não edita backlog e não apaga uma resposta para colocar um texto genérico no lugar.

A atualização do template: `mvpfy-migrate`

Esta skill cuida da atualização do `MVP.md` quando o template evolui. Ela usa IDs de seção e a versão do schema para acrescentar o que falta sem apagar o plano que você já construiu.

Como a atualização acontece

1. Ler a versão do documento.
2. Localizar os identificadores existentes.
3. Comparar o arquivo com o template atual.
4. Inserir as seções ausentes.
5. Preservar o conteúdo preenchido.
6. Marcar campos novos como pendentes.
7. Atualizar a versão depois de uma gravação válida.
8. Entregar à orquestradora uma única pergunta nova relevante.

Um exemplo

Se o template ganhar “Plano de onboarding” e o documento já descrever convite e primeiro valor em outra seção, o texto antigo permanece. A nova seção recebe uma síntese segura ou “Pendente”. O MVPFy pergunta somente o detalhe que não puder ser entendido a partir do material existente.

Onde termina este trabalho

Esta skill não remove histórico, não duplica seções, não migra arquivos do Specsfy e não cria uma série de perguntas em lote.

`mvpfy-progress`

Esta skill lê o estado e o `MVP.md` para mostrar o andamento do planejamento. Ela não conduz a entrevista, não salva resposta e não altera arquivos do Specsfy.

O que ela apresenta

- percentual geral do plano;
- andamento por área;
- respostas salvas e último ponto da entrevista;
- modelo multitenante ou instalação separada;
- primeira seção ainda pendente.

O percentual é uma referência de preenchimento. Uma seção pode conter uma recomendação e ainda precisar de confirmação na entrevista.

Exemplo

```
Progresso do MVP: 48%
Concluído: Problema e Público.
Em andamento: Produto e SaaS.
Pendente: Marketing e Validação.
Próximo ponto: definir o primeiro canal de aquisição.
```

Para abrir a visão completa, use `mvpfy progress` ou a aba **Progresso** da TUI. Para ler o documento inteiro com cores, use a aba **MVP.md**.