ASTROFY

Guia do usuário

Do pedido da página à implementação validada

Setup, entrevista, especificação, plano técnico,
execução e checkup em projetos Astro.

Edição v0.13.0
Gerado em 21 de outubro de 2026
Fonte editorial: docs/user/

Sumário

Astrofy: guia do usuário 5
- Comece aqui 5
- Consulte pela tarefa 5
- Formatos 6

Visão geral 7
- Por que o percurso de página é dividido em três etapas 7
- Um percurso completo, do pedido ao arquivo 8
- Artefatos do projeto 8

Instalação e setup 11
- Requisitos 11
- Instale o pacote e as skills 11
- Diga o que quer fazer 11
- O comando sem argumentos abre o painel 12
- Resultado da instalação 12

Planejamento de uma página 13
- Por que sete especialistas, e não uma entrevista genérica 13
- Pontos de revisão, em ordem 14
- O que sai dessa etapa, e o que ainda não existe 14

Entrevista e conteúdo 15
- O que a entrevista cobre, e por que cada item importa 15
- Como responder 15
- Conduza a entrevista na conversa 16
- Alternativa pelo CLI 16
- Exemplo curto 17

Plano de implementação 18
- O que o plano registra 18
- Por que isso virou tarefas em `.sitespec/`, e não um arquivo de plano 18
- As três tarefas mecânicas 19
- Reexecutar o plano não perde trabalho 19
- Aprovação 20

Execução e validação 21
- Conferência mínima 22
- O que fica para revisão manual, e como registrar o resultado 22

Exemplos por tipo de página ... 24
Landing page de vendas ... 24
Página de produto ... 24
Página de serviço ... 24
Página inicial ... 25
Página Sobre ... 25
Página de contato ... 25
Página de preços ... 25
Resultado comum ... 26
Execução retomável ... 27
Por que existir um estado por tarefa, e não só “feito”/“não feito” ... 27
Estados da tarefa ... 27
Passagem prática, do plano à conclusão ... 28
Retomada em outra sessão ... 29
Tarefas do site com sitespec ... 30
Por que um contrato próprio, em vez de uma issue qualquer ... 30
Criar uma tarefa ... 30
Checklist e identidade do item ... 31
Acompanhar o progresso ... 31
Executar a tarefa ... 32
Dependências entre tarefas ... 32
Retomada e manutenção ... 33
Quando o conteúdo muda depois que o plano já existe ... 33
Atualização do Astrofy ... 33
Manutenção contínua ... 34
Solução de problemas ... 35
O projeto não é reconhecido ... 35
O setup encontrou JSON inválido ... 35
O tipo de página ficou ambíguo ... 35
A implementação divergiu do conteúdo aprovado ... 35
O plano ficou desatualizado ... 36
Uma tarefa não sai de `blocked` ... 36
O Chromium não está disponível ... 36
A migração pendente deve falhar no CI ... 36
O ebook não confere ... 37

`astrofy setup` está lento ... 37
Astrofy Design na prática ... 38
Escolher entre examinar e alterar ... 38
Criar uma página ou um componente ... 38
Estudar uma referência visual ... 39
Manter a identidade entre páginas ... 39
Conferir a entrega ... 40
Referência de uso do Astrofy Design ... 41
Disponibilizar a skill no projeto ... 41
Build: criar uma interface ... 41
Study: analisar uma referência ... 42
Audit: revisar sem editar ... 43
Redesign: mudar a composição existente ... 44
Arquivos de consulta e arquivos alterados ... 45
Tokens e verificações técnicas ... 45

Astrofy: guia do usuário

Use o Astrofy em uma conversa com Codex ou Claude. Depois de instalar o pacote e disponibilizar as skills no projeto, descreva o trabalho com suas próprias palavras — sem precisar nomear um comando, uma skill ou um formato de arquivo. O agente examina o site, seleciona as skills necessárias conforme o pedido e solicita só as informações que faltam para continuar, em vez de seguir um roteiro fixo igual para todo caso.

Você pode consultar este manual diretamente em duas situações: quando você quer entender o que o agente está fazendo e por quê, antes de aprovar algo; e quando você quer conduzir uma etapa manualmente pelo terminal, sem o agente — por exemplo, numa automação ou num script de CI.

Comece aqui

1. Instale o pacote e as skills.
2. Veja como a conversa conduz o trabalho — este capítulo explica por que o percurso de página é dividido em etapas e o que cada etapa produz.
3. Descreva ao agente a página ou a mudança desejada, em linguagem natural.

Os demais capítulos explicam cada etapa em profundidade: o porquê de cada decisão de design do Astrofy, não só o passo a seguir. Volte a eles quando quiser acompanhar o que o agente está fazendo, entender por que ele pediu algo específico, ou operar o CLI manualmente.

Consulte pela tarefa

- **Definir uma página:** planejamento e entrevista de conteúdo — do pedido inicial até o conteúdo aprovado.
- **Revisar a implementação:** plano técnico e execução e validação — como o conteúdo aprovado se transforma em tarefas técnicas e código.
- **Consultar exemplos:** sete tipos de página — um percurso resumido para cada tipo de entrevistadora.
- **Continuar depois:** execução retomável e retomada e manutenção — estados, transições e como voltar a um projeto depois de um tempo.
- **Registrar uma tarefa pontual do site:** sitespec — para qualquer trabalho que não é uma página nova, e para as próprias tarefas técnicas derivadas de uma página.
- **Criar ou revisar a composição visual:** Astrofy Design na prática explica os modos com exemplos comentados. A referência de uso detalha entradas, arquivos, limites e exemplos para consulta.
- **Resolver uma falha:** solução de problemas.

O manual do CLI descreve, comando por comando, todas as opções disponíveis para instalação de skills, consulta de estado, checks, revisão manual e entrevistas pelo terminal — use-o como referência de sintaxe quando já souber qual comando quer executar; use este manual quando quiser entender o fluxo por trás do comando.

Formatos

Os capítulos deste diretório são a fonte canônica — eles não são gerados a partir de outro lugar. O projeto compila a mesma edição em PDF e EPUB dentro de `ebook/`, para leitura offline ou distribuição fora do repositório. A versão do manual (e do ebook) acompanha a versão do pacote Astrofy publicada no npm; os dois nunca divergem porque o próprio processo de release recusa publicar uma versão nova sem recompilar e validar o ebook contra os capítulos atuais.

Visão geral

Você descreve o trabalho ao agente na conversa, em linguagem natural. O Astrofy não espera um comando nem um formato: ele examina o projeto, decide quais das suas skills resolvem o pedido e pergunta só o que falta para continuar. Para uma página inteira, isso significa aprovar o conteúdo antes da edição de qualquer arquivo técnico. para um ajuste pontual, significa registrar uma tarefa e executá-la sem reabrir uma entrevista completa.

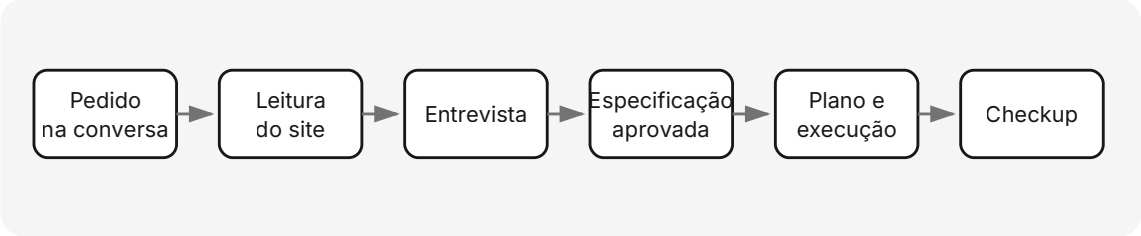
Por que o percurso de página é dividido em três etapas

Uma página nova mistura duas decisões de natureza muito diferente: o que ela diz (conteúdo, tom, oferta, prova) e como ela é construída (rota, componentes, SEO, acessibilidade, testes). Misturar as duas no mesmo passo tende a produzir um dos dois problemas: ou a conversa editorial trava em detalhe técnico antes de a mensagem estar certa, ou a implementação começa sobre conteúdo que ainda vai mudar. O Astrofy separa essas decisões em três etapas, cada uma com uma orquestradora e um artefato próprio:

1. **`astrofy-page-planner`** identifica o tipo de página (venda, produto, serviço, Home, Sobre, contato ou preços), escolhe a entrevistadora especializada nesse tipo e conduz a conversa até você aprovar o conteúdo. O resultado é `page-spec.json` — um contrato de conteúdo, sem nenhuma decisão de implementação ainda.
2. **`astrofy-implementation-planner`** só entra depois dessa aprovação. Ele relê o `page-spec.json`, cruza com a arquitetura real do projeto (rotas existentes, componentes disponíveis, configuração atual) e produz o trabalho técnico como tarefas rastreáveis — hoje, sitespecs ligadas à página (capítulo 4).
3. As skills técnicas selecionadas por esse plano implementam e conferem cada tarefa aprovada, uma a uma, registrando o que mudou.

Essa ordem não é burocracia: ela existe para que uma mudança de conteúdo (um título diferente, uma oferta reformulada) não exija refazer trabalho técnico que já estava correto, e para que uma decisão técnica (trocar um componente, ajustar uma integração) não force reabrir a conversa sobre o que a página diz.

`astrofy-setup` fica fora desse percurso de página: ele serve para a adoção ampla do framework num projeto existente, ou para revisar a cobertura das 54 skills contra o site atual. Se você pedir uma página específica, o agente avalia se precisa executar o setup antes — você não precisa pedir isso separadamente.



Etapas de uma página Astro

Um percurso completo, do pedido ao arquivo

Veja como isso se encadeia na prática, com um pedido real:

> “Quero uma página para o nosso produto Atlas, focada em time de atendimento.”

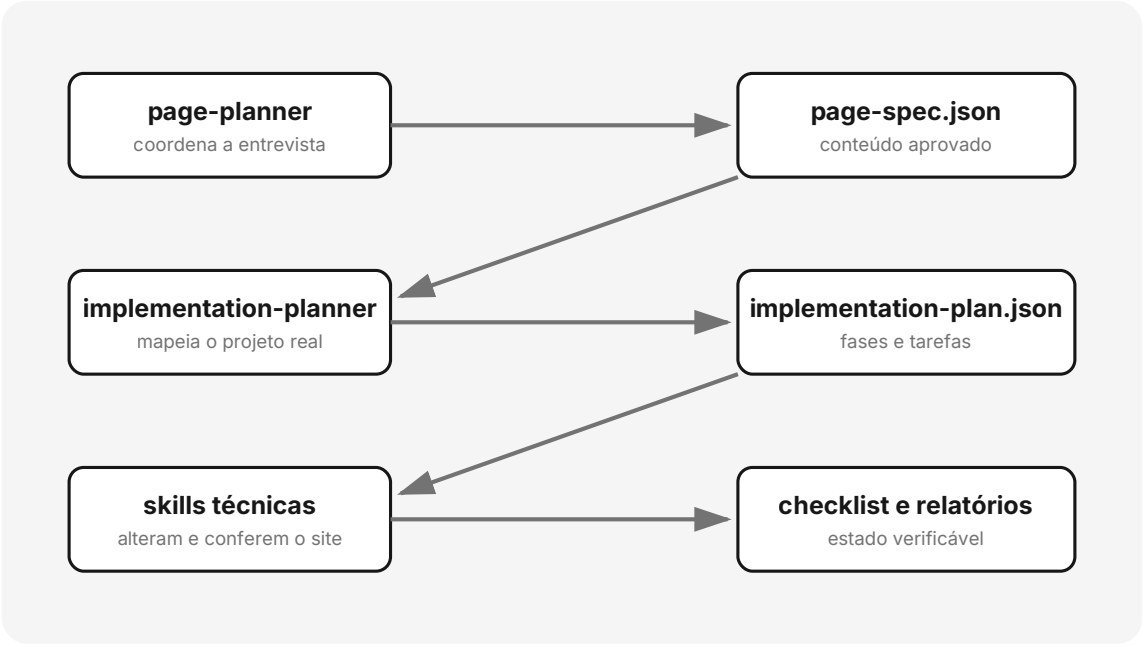
1. O agente identifica que isso é uma página de **produto** e aciona `astrofy-plan-product-page` através de `astrofy-page-planner`.
2. A entrevistadora pergunta o essencial: ação principal (demonstração? teste? compra?), público, capacidades a destacar, integrações reais. Você responde por etapas; o agente propõe três arquiteturas quando precisa da sua escolha e recomenda uma.
3. Quando o conteúdo está completo, o agente grava `.astrofy/pages/atlas/page-spec.json` (o contrato estruturado) e `.astrofy/pages/atlas/page.md` (a mesma informação em texto corrido, para você revisar sem abrir JSON). Nada de código ainda existe.
4. Você aprova. Só então `astrofy-implementation-planner` entra: ele lê o tipo de página, decide a skill de composição adequada (por exemplo `astrofy-page-design` para produto) e grava três tarefas em `.sitespec/`: criar a rota, aplicar o conteúdo e validar a página publicada. Cada uma aponta de volta para `atlas` pelos campos `page` e `task` do front matter (capítulo 4 detalha isso).
5. As skills técnicas executam essas tarefas em ordem (a segunda depende da primeira, a terceira da segunda) e registram o progresso. Você pode acompanhar com `astrofy sitespec progress` ou pela aba Specs da TUI em qualquer momento, mesmo em outra sessão.

Essas cinco etapas não publicam o site automaticamente: aprovação de conteúdo autoriza conteúdo, aprovação de plano autoriza o trabalho técnico descrito, e publicar o site é uma decisão sua, feita fora do Astrofy.

Artefatos do projeto

A tabela abaixo lista onde cada decisão fica registrada e por quê. Esses arquivos existem para que o trabalho seja retomável — uma conversa interrompida não perde o que já foi decidido, porque a decisão está no disco, não só no histórico da conversa.

Caminho	Conteúdo	Por que existe
`.astrofy/config/project.json`	Configuração criada durante a adoção do Astrofy	Registra convenções do projeto (caminhos, recursos habilitados) uma única vez, para que todo comando futuro as reconheça sem perguntar de novo
`.astrofy/pages/<slug>/page-spec.json`	Especificação estruturada da página aprovada	É o contrato de conteúdo; o plano técnico lê exatamente este arquivo, nunca a conversa em si
`.astrofy/pages/<slug>/page.md`	O mesmo conteúdo, em texto corrido	Existe só para leitura humana — revisar um JSON extenso é mais lento que ler prosa
`astrofy.checklist.json`	Estado das verificações executadas no site	Conserva histórico de aprovação/falha por regra e por rota, para que `astrofy check` não precise reavaliar o que não mudou
`.sitespec/<id>-<slug>/SITEPEC.md`	Tarefa pontual de desenvolvimento	Cobre trabalhos definidos fora de uma entrevista de página — correção, ajuste, pendência de SEO
`.sitespec/<id>-<slug>-<task>/SITEPEC.md`	Tarefa técnica da página (`route` , `content` , `check`)	É o que `astrofy plan` produz a partir do `page-spec.json` ; `page` e `task` no front matter ligam a tarefa de volta à página



Relação entre orquestradoras, artefatos e skills

Dois arquivos que existiam em versões anteriores do Astrofy desapareceram de propósito: `.astrofy/plans/<slug>/implementation-plan.json` e `.md`. O plano técnico de uma página hoje é um conjunto de sitespecs, não um arquivo próprio — isso unifica o progresso de uma página com o progresso de qualquer outra tarefa do site, numa única consulta (capítulos 4, 7 e 8).

Instalação e setup

Requisitos

- Node.js 22.12.0 ou posterior — o CLI usa recursos recentes do runtime (como `node:timers/promises` com suporte a `AbortSignal`) e não testa versões anteriores.
- Um projeto Astro existente com `package.json` declarando `astro` em `dependencies` ou `devDependencies`. O Astrofy não cria um site do zero: ele opera sobre um projeto que já existe, mesmo que ainda esteja no início.
- Permissão de escrita no diretório do projeto. Toda operação que grava algo usa escrita atômica (arquivo temporário, sincronização e troca por `rename`) e um lock em `.astrofy/cache/write.lock`, então duas execuções simultâneas não corrompem o estado — mas ambas precisam conseguir escrever no disco.

Instale o pacote e as skills

```
npm install --global @promovaweb/astrofy
cd /caminho/do/site
astrofy skills install --agent codex
```

Troque `codex` por `claude` se esse for o agente usado no projeto — o destino muda (`.agents/skills/` para Codex, `.claude/skills/` para Claude), mas o conteúdo instalado é o mesmo catálogo de 54 skills. Para conferir o que seria escrito antes de gravar, acrescente `--dry-run`; o comando mostra a lista de arquivos sem gravar no disco.

A instalação é local ao projeto, nunca global: o Astrofy não escreve em diretórios fora do site aberto. Ela também é conservadora com personalização: se você já editou o `SKILL.md` de uma skill instalada, uma nova instalação recusa sobrescrevê-lo e pede para conciliar manualmente — isso existe para que um ajuste seu nunca seja apagado silenciosamente por uma atualização de versão.

Diga o que quer fazer

Depois de instalar, abra uma conversa no projeto e diga o que quer, sem se preocupar com qual comando ou skill usar — essa escolha é do agente. Por exemplo:

> Quero criar uma página de produto. Use o Astrofy para entender o projeto, conversar comigo sobre o conteúdo e mostrar o plano antes de implementar.

O agente lê o pedido e decide o caminho:

- **Pedido amplo** (“organize o projeto”, “revise o que já existe”, “prepare o site para o Astrofy”) encaminha para `astrofy-setup`. Essa skill reconhece convenções e integrações já presentes no site, inicializa os contratos que ainda faltam (`.astrofy/config/`, `astrofy.checklist.json`) e registra o estado em `.astrofy/setup-state.json` para que uma nova execução saiba o que já foi feito e o que mudou desde então.
- **Pedido de página específica** (“quero uma página para X”) encaminha direto para `astrofy-page-planner` (capítulo 2), sem passar por setup — a menos que o projeto ainda não tenha os contratos básicos, caso em que o agente prepara isso primeiro, sem perguntar.
- **Tarefa pontual** (“ajuste o contraste do botão”, “corrija esse link quebrado”) encaminha para `astrofy-sitespec-create` (capítulo 8).

Em nenhum desses três caminhos você precisa saber de antemão qual comando do CLI corresponde à etapa. O agente executa `astrofy inspect`, `astrofy init`, `astrofy page create`, `astrofy page answer`, `astrofy plan` e outros comandos conforme a conversa avança — você só vê os resultados. Use esses comandos diretamente apenas quando quiser conduzir a mesma etapa pelo terminal, sem o agente, por exemplo em CI ou num script de automação. O manual do CLI descreve todos eles com exemplos.

O comando sem argumentos abre o painel

Depois de instalado, rodar `astrofy` sem nenhum comando abre a TUI — o painel interativo de consulta descrito no capítulo 7, que também cobre as tarefas de `.sitespec/`. Isso é deliberado: para uma pessoa num terminal real, abrir o painel é mais útil do que um resumo de texto. Em um script ou pipeline de CI, sem terminal interativo (ou com `--json` / `--ci` explícitos), o mesmo comando sem argumentos devolve exatamente a mesma consulta que `astrofy status` devolveria — então nenhuma automação existente quebra por esse comportamento.

Resultado da instalação

Depois da instalação das skills, elas ficam disponíveis no projeto, mas nada muda no código ainda — instalar skills não é o mesmo que adotar o framework. A adoção (via `astrofy-setup` ou pela primeira tarefa que precisar desses contratos) cria `.astrofy/` com a configuração do projeto e `astrofy.checklist.json` com o catálogo de verificações aplicáveis.

Antes de aprovar qualquer implementação, confira os arquivos alterados — `git status` e `git diff` continuam sendo a fonte de verdade sobre o que realmente mudou no disco, independente do que a conversa descreveu.

Planejamento de uma página

Descreva a página ao agente na conversa, com suas próprias palavras — você não precisa nomear um tipo de página nem conhecer o catálogo de especialistas. `astrofy-page-planner` examina a estrutura atual do site (rotas existentes, convenções já em uso) e escolhe qual das sete entrevistadoras especializadas conduz a conversa. Se o projeto ainda não tiver os contratos básicos do Astrofy, o agente prepara isso primeiro, sem perguntar — esse é o mesmo encaminhamento automático descrito no capítulo 1.

Por que sete especialistas, e não uma entrevista genérica

Uma página de vendas e uma página de contato precisam de informação completamente diferente: a primeira vive de oferta, prova e objeção; a segunda vive de campos de formulário, consentimento e prazo de resposta. Uma entrevista única e genérica para os dois casos acaba perguntando coisa irrelevante para um tipo e deixando de perguntar o essencial para o outro. Por isso o Astrofy mantém uma entrevistadora por tipo, cada uma com as perguntas certas para aquele formato:

Pedido	Especialista	O que ela garante que não falte
Landing page de vendas	`astrofy-plan-sales-page`	Oferta, prova, preço, objeções, urgência
Página de produto	`astrofy-plan-product-page`	Capacidades, casos de uso, integrações, prova
Página de serviço	`astrofy-plan-service-page`	Escopo, entregas, processo, qualificação do cliente
Página inicial	`astrofy-plan-homepage`	Públicos, destinos principais, proposta central
Página Sobre	`astrofy-plan-about-page`	História verificável, pessoas autorizadas, princípios
Página de contato	`astrofy-plan-contact-page`	Canais, campos mínimos, consentimento, prazo
Página de preços	`astrofy-plan-pricing-page`	Planos, cobrança, comparação, limites

Todas as sete entregam o mesmo formato de saída (`page-spec.json` validado pelo mesmo schema), então o restante do percurso — plano técnico, execução, validação — não precisa saber qual entrevistadora rodou. A diferença fica contida na conversa, não se propaga para o código.

Se a solicitação combinar dois formatos — uma página de produto com tabela de planos, por exemplo — a orquestradora define um tipo principal (aqui, produto) e registra as necessidades do segundo formato (preços) como requisitos complementares dentro da mesma especificação. Ela não cria duas páginas nem duas entrevistas separadas para um pedido que é, na prática, uma única página.

Pontos de revisão, em ordem

A entrevista não é uma lista de perguntas despejada de uma vez. Ela segue uma ordem deliberada, porque cada resposta restringe a próxima decisão:

1. **Finalidade e ação principal.** Sem saber o que a página precisa fazer (vender, capturar lead, explicar, comparar), nenhuma decisão de arquitetura tem base.
2. **Arquitetura.** Com a finalidade clara, a especialista propõe três estruturas de seções diferentes e recomenda uma. Você escolhe ou pede ajuste — nunca é forçado a aceitar a primeira proposta.
3. **Conteúdo e mídia.** Dentro da arquitetura aprovada, cada seção recebe texto, imagens e referências (capítulo 3 detalha como responder a essa parte).
4. **Comportamentos.** Estados de interface, validações de formulário, condições de exibição — o que a página faz, não só o que ela mostra.

Você recebe a página completa (em `page.md`, texto corrido) para revisar ao final, não só pedaço por pedaço. Isso existe porque uma seção isolada pode parecer certa e, lida em sequência com as outras, revelar repetição, contradição ou lacuna que só aparece no conjunto.

O que sai dessa etapa, e o que ainda não existe

A saída estruturada segue `packages/schemas/page-spec.schema.json` — um contrato fechado que valida tipo, rota, ações, seções, SEO, acessibilidade e os testes esperados da página. Ele fica em `.astrofy/pages/<slug>/page-spec.json`, com a cópia legível em `page.md` no mesmo diretório.

A implementação ainda não começa neste ponto. Nenhum componente é criado, nenhuma rota existe no projeto, nenhum commit é sugerido. Primeiro você revisa `page.md` e aprova a especificação — só essa aprovação explícita libera `astrofy-implementation-planner` para o capítulo 4. Separar essas duas aprovações (conteúdo, depois plano técnico) é o que permite ajustar um texto sem reabrir uma tarefa técnica já concluída, e vice-versa.

Entrevista e conteúdo

A especialista escolhida no capítulo 2 pergunta somente o necessário para o tipo de página em questão — ela não aplica um questionário fixo. Ela aceita três formas de resposta igualmente válidas: texto pronto para publicar, notas incompletas que ela organiza, ou autorização explícita para propor alternativas quando você não tem uma resposta formada ainda.

O que a entrevista cobre, e por que cada item importa

- **Finalidade e ação principal:** indique se a página deve levar ao cadastro, à compra, ao contato ou a outro destino específico.
- **Público e origem do acesso:** a oferta pode precisar de explicações diferentes para visitas provenientes de anúncios ou de buscas específicas.
- **Áreas da página e ordem de leitura** — a arquitetura aprovada no capítulo 2 ganha conteúdo real aqui, área por área.
- **Títulos, textos, rótulos e mensagens de estado** — inclui texto que é fácil esquecer, como a mensagem de erro de um formulário ou o rótulo de um botão desabilitado.
- **Imagens, vídeos, ilustrações e suas fontes** — a especialista registra a origem para a skill de implementação saber se precisa buscar, gerar ou receber o arquivo.
- **Links, formulários e integrações** — destinos reais, não placeholders; um link `#` ou um endpoint fictício se propaga para o plano técnico como se fosse real.
- **Amostra de texto aprovada, tom desejado e expressões a preservar ou evitar** — é o que faz a copy escrita por uma skill técnica soar como a marca, em vez de genérica.
- **Fatos que precisam de fonte** — números, afirmações institucionais e qualquer coisa verificável entra marcada, para que a revisão saiba o que confirmar antes de publicar.
- **Conteúdo ainda pendente** — uma lacuna reconhecida e registrada é diferente de uma lacuna que ninguém percebeu.

Como responder

Você pode responder uma etapa por vez, na ordem que a especialista apresenta, ou tudo de uma vez se já tiver as respostas prontas. Para cada proposta dela, três reações são válidas: **aprovar**, **ajustar** ou **manter pendente**. Não existe uma quarta opção de "ignorar" — uma área sem resposta vira pendência registrada, não um vazio silencioso.

Quando você fornece um texto final, diga explicitamente que ele deve ser preservado. Copy fornecida e aprovada não é reescrita só para caber no layout — se o texto não couber, a solução é ajustar o layout, a skill de implementação não tem autorização para encurtar ou reescrever conteúdo que você já aprovou. Quando você quiser ajuda editorial em vez de escrever o texto você mesmo, informe os fatos disponíveis e o tom desejado; o agente propõe a copy como rascunho, não como final.

Você também pode fornecer uma amostra de texto já aprovada em outro lugar (um post, um email, uma página existente) ou simplesmente descrever como quer que a página soe (“direto, sem jargão”, “mais técnico que o resto do site”). O Astrofy registra essa orientação de voz para que as skills que escrevem ou revisam copy no plano técnico (capítulo 4) e na execução (capítulo 5) a usem — isso evita que o tom combinado na entrevista se perca entre uma etapa e outra.

Propostas escritas pelo agente permanecem como rascunho até sua aprovação explícita — nada que você não aprovou vira parte da especificação final. Áreas opcionais podem ficar pendentes deliberadamente: a aprovação final registra esse estado, então uma lacuna conhecida não é preenchida silenciosamente durante a implementação por esquecimento do que ficou pendente.

Conduza a entrevista na conversa

No caminho comum, você simplesmente responde às perguntas do agente na ordem apresentada. Diante de cada proposta, aprove, ajuste ou deixe uma área opcional pendente, usando linguagem natural — não há sintaxe especial a aprender. O agente registra cada resposta e monta a especificação conforme a conversa avança. Você pode consultar os arquivos gerados depois de responder.

Alternativa pelo CLI

Use estes comandos se você quiser conduzir a mesma entrevista pelo terminal, sem o agente — por exemplo, para integrar a uma automação ou para registrar respostas de um formulário externo. Eles não são necessários no uso conversacional comum.

Inicie a entrevista informando o slug e o tipo da página:

```
astrofy page create --slug atlas --type product
astrofy page status --slug atlas
```

`page status` devolve até três itens em `nextQuestions` — o mesmo limite que a conversa usa, para você responder a uma lista curta por vez. Registre cada resposta com o identificador de campo recebido:

```
astrofy page answer --slug atlas --field title --value "Atlas"
astrofy page answer --slug atlas --field route --value /atlas/
```

O arquivo `interview.json` conserva respostas e lacunas entre execuções — rodar `page answer` em sessões diferentes continua a mesma entrevista, não recomeça do zero. Quando todas as entradas obrigatórias existem, o próprio comando gera `page-spec.json` com conteúdo, SEO, integrações, acessibilidade e a lista de testes esperados, sem uma etapa separada.

O schema atual do CLI não tem um campo próprio para amostras de voz. Quando conduzir a entrevista pelo terminal, registre a direção de voz confirmada na seção “Voz e redação” de `page.md`, manualmente, durante a revisão da especificação — no uso conversacional, o agente já faz esse registro por você.

Exemplo curto

```
Usuário: Quero uma página do produto Atlas.
Astrofy: A ação principal deve iniciar teste, solicitar demonstração ou comprar?
Usuário: Solicitar demonstração.
Astrofy: Vou propor três arquiteturas para uma página de produto orientada à demo.
```

A partir daqui a conversa continua área por área — capacidades a destacar, integrações reais, provas disponíveis — até a especificação completa estar pronta para sua revisão final.

Plano de implementação

Depois da aprovação da página (capítulo 3), o agente chama `astrofy-implementation-planner`. Essa skill relê o projeto — rotas existentes, componentes disponíveis, configuração atual — e relaciona cada área da especificação aprovada com o trabalho técnico real necessário para publicá-la. Ela produz trabalho, não conteúdo: nenhuma palavra do texto aprovado muda aqui.

O que o plano registra

- rota e modo de renderização;
- direção de voz para tarefas que exibem texto, herdada da entrevista;
- layouts, componentes e conteúdo que podem ser reutilizados em vez de recriados;
- arquivos que serão criados ou alterados, por caminho real do projeto;
- integrações e variáveis de ambiente necessárias;
- acessibilidade, SEO e dados estruturados aplicáveis ao tipo de página;
- testes automáticos e pontos que exigem revisão manual;
- revisão editorial, quando a tarefa cria ou altera texto visível;
- documentação afetada pela mudança;
- dependências, ordem de execução e estimativa de cada tarefa.

Por que isso virou tarefas em `.sitespec/`, e não um arquivo de plano

Antes, esse plano vivia sozinho em `.astrofy/plans/<slug>/implementation-plan.json` — um formato próprio, sem relação com qualquer outra tarefa do site. Isso criava um problema real: uma correção pontual (ajustar um contraste, corrigir um link) e uma tarefa de implementação de página eram rastreadas em dois lugares completamente separados, sem nenhuma visão única de progresso.

Hoje cada tarefa do plano **é uma sitespec comum** — o mesmo contrato descrito em `docs/sitespec.md` e usado pelo capítulo 8 para tarefas pontuais, com o mesmo schema (`packages/schemas/sitespec.schema.json`). A única diferença é que uma tarefa de página carrega dois campos extras no front matter: `page` (o slug da página) e `task` (o papel da tarefa dentro do plano daquela página — `route`, `content` ou `check`). Isso significa que `astrofy sitespec list`, `astrofy sitespec progress` e a aba Specs da TUI (capítulos 7 e 8) mostram as tarefas de página e as tarefas pontuais juntas, no mesmo resumo — você não precisa consultar dois lugares para saber o que falta no site inteiro.

As três tarefas mecânicas

`astrofy plan --slug <slug>` grava sempre três tarefas, nessa ordem de dependência:

task	O que cobre	Depende de
`route`	Criar a rota com o layout atual	nada
`content`	Aplicar o conteúdo e as seções aprovadas	`route`
`check`	Executar `npm run check` e `astrofy check --browser`	`content`

Cada uma vira um diretório próprio em `.sitespec/<id>-<slug>-<task>/`, por exemplo `.sitespec/0007-atlas-route/`. A dependência entre elas é real, não decorativa: `content` só perde a entrada em `blockedBy` depois que `route` chega a `status: done`; `check` só depois de `content`. `astrofy-implementation-planner` pode acrescentar tarefas além dessas três quando a página for grande o suficiente para exigir mais decomposição — sempre com o mesmo `page` e um `task` próprio, que não colida com `route`, `content` ou `check` (veja `skills/astrofy-implementation-planner/references/technical.md` para os orientações para dividir o trabalho em mais tarefas).

```
astrofy plan --slug atlas
```

As tarefas começam em `status: draft` — nenhuma delas é marcada como "pronta" separadamente; a leitura (`astrofy sitespec list` / `progress`) calcula `blockedBy` a partir de `dependsOn` a cada consulta. Uma tarefa `draft` sem `blockedBy` já pode começar; uma tarefa `draft` com `blockedBy` ainda espera a predecessora.

Reexecutar o plano não perde trabalho

Se a especificação da página mudar depois que o plano já foi gerado — uma seção nova, um texto revisado — rode `astrofy plan --slug <slug>` de novo. O comando reconcilia, não recria do zero:

- o `id` de cada tarefa (o prefixo numérico do diretório) é preservado;
- o `status` atual é preservado — uma tarefa já `done` ou `in_progress` não volta para `draft` só porque o plano foi regenerado;
- os itens de checklist já marcados `[x]` permanecem marcados quando o texto do passo não mudou; um passo com texto idêntico ao anterior mantém o mesmo `{#id}` de item também;
- o que de fato é atualizado é o texto da tarefa, a skill responsável e as dependências, a partir do `page-spec.json` atual.

Isso é o que permite ajustar conteúdo sem que o progresso técnico já feito seja descartado — reexecutar o plano depois de uma mudança pequena na especificação não é um passo destrutivo.

Aprovação

Confira se cada tarefa aponta para um resultado observável e uma forma de conferência — não para uma descrição vaga como “deixar responsivo”. Ajuste o plano antes de executar quando o caminho proposto não combinar com a arquitetura atual do projeto (por exemplo, se a skill de composição escolhida não for a certa para o tipo de página). A aprovação do plano libera as skills técnicas para começar; ela não autoriza push, publicação ou qualquer alteração fora do que está descrito nas tarefas.

Execução e validação

O plano (capítulo 4) indica a skill principal de cada tarefa. O agente executa, uma tarefa por vez, as skills que se aplicam ao escopo aprovado — ele não dispara todas de uma vez nem decide sozinho pular uma etapa que a tarefa pede. Uma sequência possível de aplicação usa:

1. `astrofy-architecture`, `astrofy-routing` e `astrofy-config` para a base — rota, estrutura de pastas e configuração pública.
2. `astrofy-components`, `astrofy-sections` e `astrofy-page-design` para a interface — o que a página mostra e como se organiza.
3. `astrofy-images`, `astrofy-forms` ou `astrofy-react` conforme a página exigir — mídia, formulário ou interação que precise de hidratação.
4. `astrofy-accessibility`, `astrofy-seo`, `astrofy-open-graph` e `astrofy-structured-data` para qualidade pública — o que torna a página utilizável e descobrível, não só visualmente correta.
5. `astrofy-editorial-review` quando uma tarefa criar ou alterar texto visível. Essa skill pode auditar sem editar, ou corrigir a copy dentro do escopo aprovado na entrevista — ela não tem autorização para mudar o que foi decidido no capítulo 3, só para corrigir linguagem, voz e alegações dentro desse limite.
6. `astrofy-testing`, `astrofy-documentation` e `astrofy-checkup` para fechar o trabalho — confirmar que o comportamento esperado está coberto por teste e que a documentação do projeto reflete o que mudou.

Essa ordem não é arbitrária: cada camada depende da anterior existir primeiro. Revisar acessibilidade antes de a interface existir, ou revisar SEO antes do conteúdo final estar aplicado, produziria uma avaliação sobre algo que ainda vai mudar.

Cada skill lê a tarefa (hoje, uma sitespec — capítulos 4 e 8), confere a implementação atual contra o que a tarefa descreve, registra o que alterou e atualiza o estado. Se você preferir acompanhar isso pelo terminal em vez de deixar o agente conduzir, use `apply` para marcar o início e a conclusão de cada tarefa:

```
astrofy apply --slug atlas
astrofy apply --slug atlas --task route --status completed
```

O primeiro comando, sem `--task`, seleciona automaticamente a tarefa a iniciar — a que já está `in_progress`, ou a primeira `draft` sem dependência pendente (`blockedBy` vazio). O segundo marca `route` como concluída: isso grava `status: done` na sitespec e marca cada item da checklist dela como `[x]`, e libera `content` (que dependia de `route`) para começar. O capítulo 7 detalha todos os estados e o que cada transição exige.

Conferência mínima

```
npm run check
npm run build
astrofy check --root .
astrofy check --root . --browser
```

`npm run check` e `npm run build` são do próprio projeto Astro, não do Astrofy — rode-os sempre, independente do que o Astrofy reporta, porque eles detectam erro de tipo e falha de build que nenhuma skill substitui.

O primeiro `astrofy check` não abre navegador: ele avalia regras estáticas (estrutura, configuração, metadados presentes no HTML renderizado). A opção `--browser` inclui Chromium e confere comportamento que só existe em runtime — tema persistente, overflow em larguras diferentes, responsividade real, erros de JavaScript durante o carregamento. Rodar sem `--browser` primeiro é mais rápido para iterar; rodar com `--browser` antes de considerar a tarefa concluída é o que confirma que o comportamento, não só a estrutura, está correto.

O que fica para revisão manual, e como registrar o resultado

Nem toda verificação pode ser automática. O agente executa os checks aplicáveis e informa explicitamente o que ainda precisa de revisão manual — ele não declara uma tarefa concluída com uma pendência dessas em aberto. Revise a página no navegador em larguras de celular e desktop, não só numa. Formulários, menus, foco de teclado, links e mensagens de erro exigem interação real: um check estático pode confirmar que um atributo existe, mas não que o formulário de fato funciona quando você preenche e envia.

Depois de confirmar visualmente, registre o resultado — não deixe a confirmação só na sua cabeça ou na conversa, porque isso se perde entre sessões. Use `astrofy tui` (tecla `m`, dentro da aba Itens) quando estiver num terminal interativo, ou `astrofy review` quando não estiver — por exemplo, num script, ou quando você (ou um agente de IA sem terminal interativo) está registrando o resultado de uma verificação feita em outro lugar:

```
astrofy review --rule accessibility.alternatives --page /produtos/atlas/ --status
passed --reviewer "QA" --reason "Alt text confere com a imagem publicada."
```

`--rule` identifica a regra avaliada; `--page` ou `--component` desambiguam quando a mesma regra se aplica a mais de uma rota ou componente — uma regra de escopo `project`, que vale para o site inteiro, não precisa de nenhum dos dois. `--reviewer` e `--reason` são obrigatórios e não podem ficar vazios: o registro existe justamente para que a pessoa indicada em `--reviewer` explique a conferência realizada.

Uma regra `automatic` não aceita esse registro manual — ela só é aprovada pelo próprio verificador, dentro de `astrofy check`. Isso evita que uma verificação que deveria ser objetiva seja aprovada por julgamento humano só porque é mais rápido.

Exemplos por tipo de página

Os exemplos abaixo mostram, para cada um dos sete tipos de página do capítulo 2, o pedido inicial, as respostas que de fato mudam a arquitetura, a arquitetura aprovada e o foco do plano técnico que sai disso. Nomes e textos são ilustrativos, mas a estrutura de cada seção — o que a entrevistadora pergunta, o que ela produz — é real. Use estes exemplos para prever, antes de começar sua própria conversa, que tipo de pergunta esperar e o que vai sair no fim de cada etapa.

Landing page de vendas

Pedido: "Quero vender a formação Operação DevOps."

Respostas centrais: profissionais de desenvolvimento; inscrição como ação principal; turma com data e vagas; depoimentos autorizados; pagamento externo.

Arquitetura: hero com oferta, transformação esperada, programa, formato, instrutor, depoimentos, preço, perguntas frequentes e chamada final.

Plano técnico: rota da campanha, componentes de prova e preço, integração do checkout, eventos da ação principal, metadados e testes do percurso.

Página de produto

Pedido: "Quero criar uma página para o Atlas, nosso produto de atendimento."

Respostas centrais: equipes de suporte; demonstração como ação principal; capturas reais do produto; integrações confirmadas pela documentação.

Arquitetura: hero, problemas atendidos, casos de uso, recursos, demonstração visual, integrações, perguntas frequentes e chamada final.

Plano técnico: rota `/atlas/`, seções reutilizáveis, formulário de demonstração, imagens responsivas, `SoftwareApplication` em JSON-LD e testes.

Página de serviço

Pedido: "Preciso apresentar nossa consultoria de modernização."

Respostas centrais: empresas com aplicações legadas; diagnóstico como ação; escopo por etapas; entregas e responsabilidades explícitas; contato comercial.

Arquitetura: cenário atendido, resultados, entregas, processo, requisitos do cliente, equipe, casos relacionados, perguntas e contato.

Plano técnico: composição com componentes existentes, formulário com qualificação, confirmação de envio, metadados `Service` e teste dos estados.

Página inicial

Pedido: “Quero reorganizar a Home para explicar a empresa.”

Respostas centrais: três públicos; produto como destino principal; serviços e conteúdos como rotas secundárias; mensagem institucional já aprovada.

Arquitetura: proposta central, caminhos por público, produto em destaque, serviços, recursos recentes, comprovação institucional e contato.

Plano técnico: revisão da rota inicial, navegação, seções compartilhadas, ordem responsiva, links internos, metadados da organização e regressão visual.

Página Sobre

Pedido: “Quero uma página Sobre que conte nossa trajetória.”

Respostas centrais: fatos documentados; marcos com datas; pessoas com autorização; princípios descritos por ações; contato para imprensa.

Arquitetura: atuação atual, história, marcos, pessoas, princípios, presença pública e canais institucionais.

Plano técnico: conteúdo estruturado, imagens com texto alternativo, schema `Organization`, links para fontes públicas e revisão de todas as afirmações.

Página de contato

Pedido: “Quero separar suporte de oportunidades comerciais.”

Respostas centrais: dois destinos; campos mínimos por assunto; consentimento; prazo de retorno informado; email como canal alternativo.

Arquitetura: escolha do assunto, formulário adaptado, orientações de privacidade, mensagens de envio, prazo de retorno e canais alternativos.

Plano técnico: validação no cliente e servidor, proteção contra abuso, integrações por ambiente, foco após envio, testes de falha e documentação.

Página de preços

Pedido: “Quero comparar três planos do produto.”

Respostas centrais: cobrança mensal e anual; limites por plano; impostos e moeda explícitos; teste gratuito; contato para volume maior.

Arquitetura: contexto da cobrança, seletor de período, cartões dos planos, tabela comparativa, condições comerciais, perguntas e ação por plano.

Plano técnico: fonte única para preços, seletor acessível, URLs do checkout, dados estruturados compatíveis com o conteúdo e testes das combinações.

Resultado comum

Independente do tipo, os sete exemplos terminam no mesmo conjunto de arquivos, porque o formato de saída não varia por tipo de página — só o conteúdo da entrevista varia:

- `.astrofy/pages/<slug>/page-spec.json` e `page.md`, com o conteúdo aprovado (capítulos 2 e 3);
- três sitespecs em `.sitespec/` — `route`, `content` e `check` —, cada uma apontando de volta para `<slug>` pelos campos `page` e `task` (capítulo 4);
- o progresso dessas três tarefas visível em `astrofy sitespec progress` e na aba Specs da TUI, junto de qualquer outra tarefa do site.

A implementação somente usa fatos e textos aprovados na entrevista — uma skill técnica não inventa um número de prova, um nome de integração ou uma frase de oferta que não apareceu na conversa. Se um exemplo acima parece faltar detalhe (um preço, uma integração específica), é porque a entrevista real cobriria isso explicitamente antes do plano ser gerado; os exemplos aqui são resumidos para mostrar o formato, não o conteúdo completo de uma conversa real.

Execução retomável

Cada tarefa da página é uma sitespec com `page` e `task` no front matter (contrato completo em `docs/sitespec.md`, capítulo 8 para o uso geral de sitespecs). O agente atualiza esse registro enquanto trabalha — não ao final de tudo, mas a cada transição real — para que você, ou outra sessão de conversa, consiga retomar exatamente de onde o trabalho parou, consultando o histórico do trabalho realizado. Este capítulo mostra os comandos do CLI para consultar e atualizar essas tarefas pelo terminal, sem precisar do agente.

Por que existir um estado por tarefa, e não só "feito"/"não feito"

Uma tarefa técnica real passa por mais situações do que só concluída ou pendente. Ela pode estar esperando outra terminar, pode estar em andamento numa sessão anterior que foi interrompida, pode ter sido tentada e falhado numa validação, ou pode ter sido deliberadamente descartada. Cada uma dessas situações exige de você uma ação diferente ao retomar o trabalho — continuar, investigar a falha, ou simplesmente seguir para a próxima tarefa. Por isso o Astrofy registra um estado específico, não um booleano.

Estados da tarefa

`astrofy apply --status` usa o vocabulário abaixo — o mesmo desde antes da unificação com sitespec, para que nenhum script existente precise mudar. Internamente, cada valor grava um `status` de sitespec diferente; essa tradução é feita pelo próprio comando, você não escreve o `status` interno diretamente.

`--status` de `apply`	`status` da sitespec	Significado	Próxima ação permitida
(tarefa recém-criada)	`draft`	Ainda não começou	Conferir `blockedBy` e iniciar
`in_progress`	`in_progress`	Há trabalho em andamento	Continuar os passos registrados na checklist
`completed`	`done`	Checklist inteira marcada	Libera as tarefas dependentes
`failed`	`blocked`	Uma validação falhou	Corrigir e tentar de novo
`skipped`	`archived`	Saiu do escopo com justificativa	Manter a justificativa registrada

Versões anteriores do Astrofy usavam um par `pending` / `ready` separado do `status` em si — uma tarefa era `pending` enquanto dependia de outra, e virava `ready` quando a dependência terminava, só então podendo passar para `in_progress`. Isso não existe mais como campo gravado: a mesma informação é calculada a cada leitura, não armazenada, o que evita que ela

fique desatualizada após uma edição manual do arquivo. `astrofy sitespec list` / `progress` calculam `blockedBy` a partir de `dependsOn` toda vez que leem — uma tarefa `draft` sem `blockedBy` pode começar agora; uma tarefa `draft` com `blockedBy` ainda depende de outra terminar primeiro.

Passagem prática, do plano à conclusão

`astrofy plan --slug atlas` (capítulo 4) grava a tarefa `route` como uma sitespec nova, em `draft`, sem nenhuma dependência:

```
status: draft
dependsOn: []
relatedFiles:
  - src/pages/produtos/atlas.astro
```

Ao iniciar o trabalho, a skill relê a rota esperada, confirma os arquivos envolvidos e muda o status para `in_progress` — isso é registrado antes de qualquer edição de código, não depois, para que uma interrupção no meio do trabalho deixe rastro de que algo estava em andamento:

```
astrofy apply --slug atlas --task route --status in_progress
```

Depois de editar o arquivo, a skill executa as validações listadas na checklist da tarefa e faz a revisão manual que ela indicar. Só então registra a conclusão — e esse comando faz duas coisas de uma vez: marca `status: done` **e** marca cada item da checklist dessa tarefa como `[x]`. Isso existe porque o contrato de sitespec exige checklist completa para um item estar `done` (capítulo 8); `apply --status completed` não deixaria essa regra satisfeita por acidente.

```
astrofy apply --slug atlas --task route --status completed
```

Se uma validação falhar, registre `failed` em vez de `completed` — não force a conclusão para "destravar" a dependente. A tarefa vai para `status: blocked`, com um motivo registrado automaticamente (`blockedReason`), e a tarefa dependente continua com `blockedBy` não vazio, porque a predecessora ainda não está `done`. Uma nova tentativa depois da correção volta à tarefa com falha antes de avançar — o Astrofy não deixa uma dependente começar enquanto a predecessora registrada como falha não for corrigida ou explicitamente contornada:

```
astrofy apply --slug atlas --task route --status failed
astrofy apply --slug atlas --task route --status completed
```

Ao concluir uma tarefa, a dependente perde a entrada correspondente em `blockedBy` automaticamente, na próxima leitura — isso não é um campo que você precisa limpar manualmente, pois o comando o recalcula. Sem `--task`, `astrofy apply --slug atlas` seleciona

a tarefa automaticamente: primeiro procura uma tarefa já `in_progress` (para continuar o que estava em andamento antes de começar algo novo); se não houver, escolhe a primeira `draft` sem `blockedBy`.

Retomada em outra sessão

Quando você volta a uma conversa ou abre o terminal depois de um tempo, siga esta sequência — ela existe para reconstruir o contexto antes de agir, não para agir primeiro e descobrir o contexto depois:

1. Confira `git status` e os arquivos alterados — o disco é a fonte de verdade, não a memória da conversa anterior.
2. Execute `astrofy page status --slug atlas` e leia `page-spec.json` para confirmar que o conteúdo aprovado não mudou enquanto você estava fora.
3. Execute `astrofy sitespec list --json` (ou `sitespec progress`) e filtre as tarefas com `page: atlas` para localizar a próxima com trabalho pendente.
4. Localize tarefas `in_progress` ou `blocked` — são as que precisam de atenção imediata, diferente de uma `draft` ainda esperando dependência.
5. Confirme se os arquivos no disco ainda correspondem ao que o registro da tarefa descreve. Uma edição feita fora do fluxo do Astrofy pode ter alterado a implementação nesse intervalo.
6. Continue a tarefa e repita as validações da checklist — não assuma que uma validação passada antes continua válida sem reexecutar.
7. Atualize o estado com `astrofy apply` e confirme que a dependência saiu de `blockedBy` da próxima tarefa, antes de considerar a página pronta para a próxima etapa.

Tarefas do site com sitespec

Nem toda mudança no site é uma página nova. Um ajuste de componente, uma correção de contraste, uma pendência de SEO, um link quebrado — nada disso precisa de uma entrevista de página inteira para ser registrado e acompanhado. Use `astrofy-page-planner` (capítulo 2) quando o resultado é uma página completa, com conteúdo a ser aprovado; use sitespec para qualquer outro trabalho de desenvolvimento do site, incluindo as próprias tarefas técnicas derivadas de uma página (capítulo 4 explica como essas duas coisas convivem no mesmo lugar).

Por que um contrato próprio, em vez de uma issue qualquer

Uma tarefa registrada apenas em texto livre (uma anotação, um comentário de código, uma mensagem) não tem skill responsável definida, não tem condição verificável de conclusão e não aparece em nenhuma consulta automatizada de progresso. O sitespec resolve isso com um contrato mínimo, mas fechado: todo arquivo `SITEPEC.md` tem uma skill que executa, um estado de um conjunto fixo e uma checklist com itens identificados para `astrofy sitespec list` / `progress` somarem o progresso real do site inteiro a partir de tarefas com a mesma estrutura.

Criar uma tarefa

Descreva o pedido ao agente, em linguagem natural, do mesmo jeito que você descreveria uma página. `astrofy-sitespec-create` confere primeiro se já existe uma tarefa equivalente — ela não cria duplicata para um pedido que já está registrado sob outro texto. Em seguida, ela mapeia as skills do catálogo relacionadas ao pedido e só pergunta o que sobrar sem resposta depois disso: a maioria dos pedidos já é específica o suficiente (qual arquivo, qual comportamento, qual resultado confirma a correção) para não gerar pergunta nenhuma. Quando o pedido é genuinamente ambíguo em algo que mudaria a tarefa — por exemplo, não está claro se o problema é de copy, de SEO ou de layout —, a skill faz no máximo três perguntas objetivas, cada uma com uma opção recomendada, antes de escrever qualquer arquivo. Isso é deliberado: o objetivo é não ser burocrático, então o padrão é zero perguntas, não um roteiro fixo repetido para todo pedido.

```
Pedido: "o botão primário está com contraste baixo no tema claro"
→ sem pergunta: skill, arquivo e condição de conclusão já estão no pedido
→ cria .sitespec/0003-corrigir-contraste-botao/SITEPEC.md
```

A tarefa criada fica em `status: draft`, com `primarySkill` (a skill que vai efetivamente executar a correção) e `reviewSkills` (as skills cujo conhecimento moldou os itens da checklist — por exemplo, `astrofy-accessibility` para um item de contraste, mesmo que a correção em si seja feita por `astrofy-design-system`).

Checklist e identidade do item

Cada item da checklist tem um identificador curto e fixo, `{#id}` de quatro caracteres, e pode marcar a skill responsável por aquele item específico, quando ela diverge da skill principal da tarefa:

```
- [ ] Ajustar o token de cor do botão `#astrofy-design-system` {#a3f2}
- [x] Medir o contraste atual `#astrofy-accessibility` {#a001}
```

O `{#id}` não muda depois de criado, mesmo que o item seja reordenado dentro da checklist — isso existe para que uma automação ou um comentário externo possa apontar para um item pelo identificador, mesmo após uma edição manual mudar sua posição na lista. Um item sem `{#id}`, ou com um id repetido na mesma checklist, invalida a tarefa inteira até a correção, do mesmo jeito que um campo obrigatório ausente no front matter invalidaria — o contrato completo, com todos os campos e suas regras, está em `docs/sitespec.md`.

Acompanhar o progresso

```
astrofy sitespec list --json
```

Devolve todas as tarefas existentes — pontuais e de página — com progresso (`done` / `total` da checklist), o próximo id de quatro dígitos livre para uma tarefa nova, e, para cada tarefa, `blockedBy` (ids de dependências ainda não concluídas) e `missingDependencies` (ids declarados em `dependsOn` que não correspondem a nenhuma tarefa existente — um sinal de referência a uma tarefa ausente).

Quando você quer um resumo agregado, em vez de cada tarefa em detalhe, use `astrofy sitespec progress`. Sem `--watch`, é uma única consulta; com `--watch`, o comando fica rodando e reimprime um novo resumo cada vez que algo muda em `.sitespec/`, até você interromper com `Ctrl+C`:

```
astrofy sitespec progress
astrofy sitespec progress --watch --interval 10
```

Pelo terminal interativo, `astrofy tui` (sem argumentos, já que esse é o comando padrão — capítulo 1) abre a aba **Specs** (`Ctrl+S`): ela lista as tarefas com estado, prioridade e uma barra de progresso textual, e `Enter` abre o detalhe completo de uma tarefa selecionada — descrição, checklist com a skill de cada item, arquivos relacionados e notas. A aba é só consulta: nada que você faz nela grava um arquivo, inclusive porque isso permite usá-la livremente para explorar o estado do projeto sem alterar arquivos por engano.

Executar a tarefa

`astrofy-sitespec-implement` lê a tarefa, muda `status` para `in_progress` antes de editar os arquivos da implementação, executa cada item da checklist com a skill marcada nele (ou a skill principal, quando o item não tem uma marcação própria) e marca `[x]` pelo `{#id}` correspondente ao terminar cada um — não em lote ao final, mas item por item, para que uma interrupção no meio do trabalho deixe o progresso real registrado. Ao final, três resultados são possíveis:

Resultado	`status`	O que fica registrado
Toda a checklist concluída	`done`	Nada além da checklist marcada — a conclusão é objetiva
Impedimento externo à tarefa	`blocked`	`blockedReason`, com a causa específica
Decisão de não terminar	`archived`	`archivedReason`, com a justificativa

Decisões e desvios durante a execução — por que um item foi feito diferente do esperado, qual causa impediu a continuação da tarefa — ficam registrados em `## Notas`, datados. Isso é o que permite que você retome ou revise a tarefa depois com acesso à explicação do trabalho realizado.

Dependências entre tarefas

Uma tarefa pode declarar `dependsOn` com o id de outra que precisa terminar primeiro — por exemplo, uma tarefa de conteúdo que depende de uma tarefa de rota já concluída. `astrofy sitespec list --json` resolve isso automaticamente, a cada leitura:

```
astrofy sitespec list --json
```

A tarefa dependente aparece com `blockedBy: ["0001"]` enquanto a tarefa `0001` não estiver `done`. Isso é puramente informativo — a leitura não muda `status` para `blocked` sozinha nem impede a edição da tarefa dependente mesmo assim; a pessoa ou a skill que está executando decide se uma dependência pendente de fato impede avançar naquele caso específico, ou se faz sentido seguir mesmo assim.

Retomada e manutenção

Os arquivos em `.astrofy/` e `.sitespec/` existem justamente para que o trabalho não dependa de uma conversa contínua — eles permitem continuar em outra sessão, dias depois, sem reconstituir o contexto de memória. Abra uma conversa no projeto e peça ao agente para retomar a página pelo slug, ou simplesmente descreva o que você quer continuar. O agente confere a especificação aprovada, as tarefas de `.sitespec/` ligadas a essa página, o estado do Git e continua pela próxima tarefa cujas dependências já estejam concluídas — exatamente a mesma lógica descrita no capítulo 7, só que iniciada pelo agente em vez de por você digitando comandos. Use `astrofy page status` e `astrofy apply` quando quiser conduzir essa retomada diretamente pelo terminal.

Quando o conteúdo muda depois que o plano já existe

Se a especificação de conteúdo mudar antes da implementação ter começado — você decide trocar a oferta, ajustar o público, reescrever uma seção — volte à especialista da página (capítulo 2 ou 3) para registrar a mudança primeiro, e só depois gere novamente as partes afetadas do plano com `astrofy plan --slug <slug>`. O comando reconcilia em vez de recriar do zero (capítulo 4 detalha exatamente o que é preservado e o que é atualizado nesse processo).

Se a mudança surgir depois que a execução já começou — uma tarefa já está `in_progress` ou até `done` quando você percebe que o conteúdo precisa mudar — não edite o código silenciosamente para acomodar a mudança. Registre o impacto primeiro: qual tarefa precisa ser revisitada, o que exatamente mudou na especificação, e só então edite. Isso existe porque uma tarefa marcada `done` é, por contrato, uma tarefa com checklist inteira cumprida — reabrir o trabalho sem reabrir o registro deixa o estado do projeto mentindo sobre o que de fato está pronto.

Atualização do Astrofy

```
npm update --save-dev @promovaweb/astrofy
npx astrofy --version
npx astrofy inspect --root .
```

Leia o `CHANGELOG.md` entre a versão instalada e a nova edição antes de assumir que nada mudou de comportamento — o Astrofy já passou por pelo menos uma mudança de ruptura documentada (a remoção de `.astrofy/plans/<slug>/implementation-plan.json` em favor de sitespecs ligadas à página, capítulo 4), e futuras mudanças de contrato seguem o mesmo padrão: documentadas no changelog, nunca silenciosas. Depois de atualizar, execute as validações do projeto (`astrofy`

`check`, os testes do próprio site) antes de considerar a atualização concluída — uma versão nova do CLI pode exigir que uma tarefa antiga seja replanejada ou revisitada, e é melhor descobrir isso por uma verificação explícita do que por uma falha inesperada depois.

Manutenção contínua

Fora de uma tarefa específica, duas consultas cobrem a maior parte do que vale verificar periodicamente num projeto que já usa o Astrofy:

- `astrofy status` (ou simplesmente `astrofy`, que abre o mesmo estado na TUI — capítulo 1) para conferir o que está pendente de verificação no checklist técnico do site.
- `astrofy sitespec progress` para conferir o que está pendente de execução entre as tarefas registradas, de página ou pontuais.

As duas consultas preservam os arquivos do projeto. O valor delas é justamente permitir voltar ao projeto depois de um tempo e saber, antes de decidir o que fazer, exatamente onde o trabalho estava.

Solução de problemas

O projeto não é reconhecido

Execute `astrofy inspect --root <diretório>` apontando para a pasta que contém o `package.json` do site. Confira se `astro` aparece em `dependencies` ou `devDependencies` desse arquivo — essa é a única condição que o Astrofy usa para decidir que um diretório é um projeto Astro válido. Se o comando for executado dentro de um subdiretório do projeto (como `src/pages/`), a descoberta sobe pelos diretórios pais até encontrar esse manifesto; `--root` explícito pula essa busca e aponta direto para a raiz.

O setup encontrou JSON inválido

Abra o caminho informado na mensagem, corrija a sintaxe e repita `astrofy init --dry-run` para confirmar a correção antes de gravar. O comando preserva arquivos existentes deliberadamente e interrompe a gravação quando não consegue interpretá-los — ele nunca sobrescreve uma configuração corrompida com um padrão novo, porque isso poderia apagar uma personalização que só está temporariamente malformada.

O tipo de página ficou ambíguo

Informe a finalidade principal e a ação esperada primeiro — essas duas respostas são o que a orquestradora (capítulo 2) usa para escolher a especialista certa entre as sete disponíveis. Necessidades secundárias (uma tabela de preços dentro de uma página de produto, por exemplo) não precisam de ambiguidade: registre-as como requisito complementar depois que o tipo principal estiver definido, em vez de tentar encaixar o pedido em dois tipos ao mesmo tempo.

A implementação divergiu do conteúdo aprovado

Pare a tarefa antes de continuar — não siga implementando sobre uma divergência já percebida. Compare a página atual com `page.md` e `page-spec.json` da página (capítulo 2) linha por linha, não só por impressão geral, e corrija a implementação para voltar a corresponder ao que foi aprovado. Se a divergência na verdade revela que o conteúdo aprovado estava errado ou incompleto, a correção certa é voltar à etapa de conteúdo (capítulo 3) e aprovar a mudança lá — nunca ajustar o texto só na implementação, porque isso deixa `page-spec.json` e o código dizendo coisas diferentes.

O plano ficou desatualizado

Execute `astrofy plan --slug <slug>` depois de reler o projeto e a especificação atual. O comando reconcilia as tarefas existentes em vez de recriá-las do zero — `status`, itens de checklist já marcados e o `id` de cada tarefa são preservados; o que é atualizado é o texto, a skill responsável e as dependências, a partir do `page-spec.json` atual (capítulo 4 detalha exatamente o que muda e o que não muda nessa reconciliação). Não exclua manualmente os arquivos em `.sitespec/` antes de reexecutar: isso descartaria progresso que o comando preservaria automaticamente. Se você precisar consultar o histórico de tarefas concluídas numa versão anterior do plano, use o histórico do Git sobre esses arquivos — eles são texto versionável, não um estado opaco.

Uma tarefa não sai de `blocked`

Confira o `blockedReason` registrado no front matter da sitespec — ele descreve a causa específica da interrupção, não um estado genérico. Se a causa já foi resolvida, registre `astrofy apply --slug <slug> --task <task> --status completed` (ou `in_progress`, se o trabalho ainda não tiver terminado) para seguir adiante; o comando aceita a transição de `blocked` para `in_progress` especificamente para permitir essa nova tentativa. Se a causa for uma dependência de outra tarefa ainda não concluída, confira `blockedBy` em vez de `blockedReason` — são conceitos diferentes: `blockedReason` é um impedimento que a própria tarefa registrou; `blockedBy` é calculado automaticamente a partir de `dependsOn` e não precisa de nenhuma ação manual além de concluir a predecessora.

O Chromium não está disponível

Consulte a instalação controlada pelo Astrofy, que mantém o binário fora do processo de `npm install` do próprio pacote para que você possa escolher se precisa do download para executar os checks de navegador:

```
astrofy browser status
astrofy browser install
```

Depois execute `astrofy check --browser`. No CI, mantenha o cache indicado no workflow distribuído para evitar baixar o Chromium a cada execução — o download sozinho pode custar mais tempo que o restante da suíte de verificação.

A migração pendente deve falhar no CI

Use o modo de conferência, que não grava nada e serve justamente para decidir se um pipeline deve interromper antes de continuar:

```
astrofy migrate --check
```

O comando retorna código 1 quando o contrato do projeto ainda está numa versão anterior e precisa de migração, e código 0 quando a configuração já usa a versão atual. `astrofy migrate --apply` é a única forma de de fato aplicar a migração — `--check` nunca escreve um arquivo.

O ebook não confere

Na raiz do Astrofy (não dentro de um projeto que consome o pacote), execute:

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

O primeiro comando recompila PDF, EPUB e o manifesto `build.json` a partir dos capítulos atuais de `docs/user/`. O segundo compara a edição já gerada com todos os capítulos e arquivos do pipeline, byte a byte — ele identifica capítulos editados após a última geração do ebook, antes da publicação de uma nova versão.

`astrofy setup` está lento

Se a execução parece demorar mais do que o esperado num projeto grande, confira se `.astrofy/cache/file-hashes.json` existe e está sendo atualizado — esse cache evita reler e recalcular o hash de todo o projeto a cada execução, reaproveitando o hash de um arquivo cujo `mtime` e tamanho não mudaram desde a última leitura. Um cache ausente (primeira execução, ou depois de limpar `.astrofy/cache/`) explica uma primeira execução mais lenta; execuções seguintes no mesmo projeto devem ser bem mais rápidas. O cache é descartável por natureza: se ele estiver corrompido ou vier de outra máquina, a próxima execução só o reconstrói, sem interromper o setup.

Astrofy Design na prática

Uma página pode ter o conteúdo aprovado e ainda precisar de ajustes na composição. O título ocupa espaço demais no celular, as seções parecem iguais ou o botão principal perde destaque entre os outros elementos. Você pode pedir ao Astrofy Design que examine essa página, explique os problemas ou implemente uma nova composição, conforme o trabalho que pretende fazer.

A skill `astrofy-design` orienta o agente na organização dos elementos, na tipografia, nas cores e nas interações. Ela consulta os componentes e os tokens existentes antes de propor mudanças. Num site com várias páginas, a identidade compartilhada continua orientando o resultado, mesmo quando uma seção recebe uma composição diferente.

Escolher entre examinar e alterar

Para entender o que precisa melhorar numa página existente, comece por `audit`. O agente examina os arquivos indicados e apresenta os problemas localizados, com sugestões de correção. Esse modo entrega o diagnóstico na conversa e preserva os arquivos do projeto.

Você pode pedir: “Use astrofy-design audit na página de produto. Confira a hierarquia dos títulos, o destaque da ação principal e o comportamento no celular. Explique o que encontrou sem editar os arquivos.” O caminho da página ajuda o agente a delimitar a análise. Se a conferência depender de abrir o navegador e isso não for possível, a entrega deve indicar o que ficou pendente.

Quando você já pretende mudar a aparência, use `redesign` e descreva o que quer preservar. Um pedido como “Reorganize a página de produto mantendo os textos, o formulário e as cores da marca” autoriza uma mudança visual dentro desse escopo. O agente precisa considerar os componentes que a página usa para evitar que o ajuste altere outras rotas sem necessidade.

Os nomes `audit` e `redesign` fazem parte da conversa com o agente. Eles não são comandos que você executa no terminal. O CLI instala a skill e oferece verificações técnicas, enquanto o agente conduz a análise ou a implementação solicitada.

Criar uma página ou um componente

Use `build` quando precisar implementar uma interface nova no projeto Astro. Informe a finalidade da página, o conteúdo aprovado e o destino esperado para a ação principal. O agente consulta os componentes disponíveis e apresenta a composição antes de escrever o código, quando ainda houver escolhas abertas.

Por exemplo, você pode pedir: “Use astrofy-design build para criar uma página de produto com este conteúdo aprovado. Reaproveite o cabeçalho, os botões e os tokens do site. A ação principal deve abrir o formulário de demonstração.” Esse pedido relaciona a composição com elementos reais do projeto. Se o formulário ainda não existir, o agente precisa informar isso antes de tratar a ação como funcional.

O mesmo modo atende a um componente isolado. Ao pedir uma tabela de preços, você pode delimitar a mudança à tabela e aos estados de seus botões. Isso permite avaliar a leitura dos planos sem redesenhar o cabeçalho, o rodapé e as outras seções da página. Confira também onde esse componente é usado, pois uma alteração compartilhada pode aparecer em mais de uma rota.

A skill não completa informações comerciais ausentes. Preços, depoimentos, quantidades de clientes e resultados precisam vir do conteúdo aprovado. Quando falta uma imagem, você pode fornecer um arquivo ou solicitar sua produção. Os caminhos ilustrativos da biblioteca não são imagens prontas para entrar no site.

Estudar uma referência visual

O modo `study` recebe uma URL ou uma imagem e descreve como os elementos se organizam. Você pode usá-lo para entender por que uma composição funciona antes de pedir algo semelhante para seu próprio conteúdo. O estudo examina hierarquia, distribuição do espaço, papéis tipográficos e cores, sem alterar a página do projeto.

Um pedido possível é: “Use astrofy-design study nesta captura. Explique como o título, a imagem e a ação principal se relacionam e o que muda numa tela estreita.” Uma captura registra apenas o estado visível. Ela não comprova como um menu abre, como a página reage ao teclado ou qual fonte foi usada. Essas informações precisam permanecer identificadas como não verificadas.

Uma URL pode permitir a leitura do HTML e do CSS, dependendo do acesso e das ferramentas disponíveis. Se o endereço exigir autenticação ou devolver apenas uma estrutura vazia que depende de JavaScript, o agente deve explicar a limitação. O estudo não deve apresentar uma descrição visual inventada para compensar a falta da página renderizada.

Depois da análise, você pode encerrar o trabalho, pedir uma composição para seu conteúdo ou solicitar um `design.md` portátil. Esse arquivo registra o sistema visual que será usado no seu projeto. Estudar uma página externa não transfere seus textos, logos, fotografias ou outros ativos para o seu site.

Manter a identidade entre páginas

Num redesign de várias rotas, o agente começa pelo contrato visual comum. A página de produto e a página de preços podem distribuir o conteúdo de formas diferentes, mas precisam continuar compartilhando as fontes, as cores e os estados dos componentes definidos para a marca. Trocar o tema em cada rota prejudicaria essa continuidade.

Você pode pedir: “Use astrofy-design redesign nas páginas de produto e preços. Mantenha a identidade atual e aplique o mesmo padrão de botões e títulos nas duas. Preserve os formulários, as URLs e os textos aprovados.” O agente deve apresentar os arquivos envolvidos e conferir os componentes compartilhados antes de implementar as mudanças.

Os tokens concentram valores reutilizados, como cores e espaçamentos. No Astrofy, os caminhos do arquivo de design e do CSS gerado vêm da configuração do projeto. A mudança começa na fonte dos tokens e passa pelo gerador. Editar somente o CSS gerado faria o ajuste desaparecer na próxima geração.

A skill conserva um histórico das composições em `.astrofy/design/log.json`. Esse registro ajuda a consultar trabalhos anteriores e mantém até vinte entradas recentes. Ele não substitui o Git nem registra automaticamente uma aprovação. Os modos de estudo e auditoria não criam esse histórico.

Conferir a entrega

Abra a página em larguras de celular e desktop. Confira se o título cabe, se os elementos mantêm a ordem de leitura e se a ação principal continua acessível. Uma composição pode parecer correta numa captura larga e ainda produzir rolagem horizontal ou esconder parte do formulário numa tela estreita.

A conferência precisa incluir teclado, foco visível e os estados que o componente oferece. Se houver animação, verifique a alternativa para redução de movimento. Num site com temas claro e escuro, confira as cores em ambos. O relatório deve distinguir o que foi observado no navegador do que foi conferido apenas pela leitura dos arquivos.

Execute os scripts de validação e build declarados pelo projeto. Em uma instalação inicializada do Astrofy, os comandos de tokens ajudam a conferir se a fonte e o CSS gerado continuam sincronizados. Esses resultados precisam acompanhar a inspeção visual, porque a validade do arquivo não comprova que o botão funciona ou que a página mantém uma leitura confortável.

Para consultar as entradas de cada modo, os arquivos envolvidos e mais exemplos de pedidos, use a referência do Astrofy Design. O capítulo de execução e validação explica como registrar as verificações do projeto depois da implementação.

Referência de uso do Astrofy Design

Os quatro modos do Astrofy Design recebem instruções na conversa com o agente. Esta referência descreve o que informar em cada pedido, qual resultado esperar e quais arquivos podem mudar. Para acompanhar exemplos comentados de uso, consulte o guia prático.

Disponibilizar a skill no projeto

A instalação completa das skills inclui `astrofy-design` quando ela está presente no catálogo da edição instalada. Confira o catálogo antes de tentar selecionar apenas essa skill:

```
astrofy skills list --json
astrofy skills install --agent codex --skill astrofy-design --dry-run
astrofy skills install --agent codex --skill astrofy-design
```

Execute os comandos na raiz do site ou informe `--root /caminho/do/site`. Troque `codex` por `claude` para instalar no Claude. O `--dry-run` lista os arquivos previstos sem gravá-los. Se o nome não aparecer no catálogo, use uma edição que contenha a skill ou siga a instalação pelo checkout descrita na referência de implementação.

O destino do Codex é `.agents/skills/astrofy-design/`. O destino do Claude é `.claude/skills/astrofy-design/`. A instalação inclui as instruções, as referências, as paletas de consulta e a licença. Ela não importa o CSS no site nem executa um redesign. Arquivos personalizados precisam ser conciliados antes de uma atualização que substituiria seu conteúdo.

Build: criar uma interface

O modo `build` atende a páginas e componentes novos dentro do projeto Astro. Ele também é o padrão quando o pedido de criação não contém um verbo explícito. Informe o conteúdo aprovado, a finalidade e o escopo da interface. O caminho de destino pode ser proposto pelo agente depois da leitura das rotas existentes.

Entrada	Tipo e obrigatoriedade	Uso
Finalidade e escopo	Texto necessário	Delimita a página ou o componente
Conteúdo	Texto ou arquivo necessário para a composição final	Define o que a interface apresenta
Destino	Caminho opcional	Indica a rota ou o componente a criar

Entrada	Tipo e obrigatoriedade	Uso
Identidade	Arquivos ou descrição, quando disponíveis	Orienta fontes, cores e componentes
Preferência visual	Texto opcional, sem valor padrão obrigatório	Orienta a proposta dentro da identidade existente

A entrega inclui a implementação, os arquivos alterados e as verificações executadas. O agente pode criar ou editar componentes, estilos e tokens dentro do escopo autorizado. Uma implementação também pode atualizar o histórico visual. Conteúdo ausente, imagem indisponível ou verificação não executada precisa aparecer na entrega.

Exemplos de build

1. **Página de produto:** “Use astrofy-design build para criar a página deste produto com o conteúdo aprovado em produto.md e os componentes atuais.”
2. **Tabela de preços:** “Use astrofy-design build para uma tabela com estes três planos. Preserve os preços e explique os estados dos botões.”
3. **Página de evento:** “Use astrofy-design build para esta programação. Organize horários, palestrantes e inscrição usando os ativos fornecidos.”
4. **Componente de depoimento:** “Use astrofy-design build para apresentar este depoimento autorizado, com nome e cargo exatamente como fornecidos.”
5. **Estado vazio:** “Use astrofy-design build para o estado vazio da busca, mantendo o campo de pesquisa e uma ação para remover os filtros.”

Study: analisar uma referência

O modo `study` recebe uma URL ou uma imagem. Seu resultado descreve a composição observada e os limites da análise. Uma URL permite consultar HTML e CSS quando acessíveis. Uma imagem permite analisar apenas o que aparece na captura, sem confirmar fontes exatas ou comportamentos interativos.

Entrada	Tipo e obrigatoriedade	Uso
Referência	URL ou imagem necessária	Fornece o objeto do estudo
Foco da análise	Texto opcional	Delimita tipografia, composição ou outra parte
Conteúdo próprio	Texto opcional	Permite discutir uma aplicação posterior
Exportação	Pedido explícito, ausente por padrão	Solicita um `design.md` portátil

O estudo não altera o site nem cria histórico. Um relatório em arquivo exige pedido específico. Uma URL inacessível não produz um diagnóstico visual completo. A skill recusa a reprodução de templates pagos e a cópia de páginas por pixels. A exportação baseada em URL precisa confirmar que a referência é própria ou pública para orientar a marca do projeto.

Exemplos de study

1. **Hierarquia:** "Use astrofy-design study nesta captura e explique a relação entre o título, a imagem principal e o botão."
2. **Página própria:** "Use astrofy-design study na URL da nossa página de produto e descreva a composição que existe hoje."
3. **Tipografia:** "Use astrofy-design study nesta imagem e descreva os papéis tipográficos, sem presumir os nomes das fontes."
4. **Comparação de telas:** "Use astrofy-design study nas capturas desktop e mobile fornecidas e compare a distribuição do conteúdo."
5. **Especificação portátil:** "Use astrofy-design study nesta página nossa e registre o sistema visual em design.md para reutilização no projeto."

Audit: revisar sem editar

O modo `audit` examina uma página, um componente ou um conjunto de arquivos. Informe o alvo e, se necessário, o aspecto a conferir. O agente consulta o contrato visual disponível e relaciona cada problema ao arquivo ou ao elemento afetado, com uma sugestão de correção.

Entrada	Tipo e obrigatoriedade	Uso
Alvo	Caminho ou conjunto de caminhos necessário	Delimita a revisão
Foco	Texto opcional	Prioriza uma parte da interface
Contrato visual	Arquivo existente, quando disponível	Permite comparar a implementação com a identidade
Relatório em disco	Caminho solicitado explicitamente	Grava o diagnóstico no destino indicado

A entrega padrão acontece na conversa. O modo não edita a implementação, não cria `design.md` e não atualiza o histórico visual. O relatório pode classificar problemas por gravidade, mas um total de achados ou uma nota não aprova a página. Quando a análise não incluir navegador, essa limitação deve aparecer.

Exemplos de audit

1. **Página inicial:** “Use astrofy-design audit em src/pages/index.astro e confira a hierarquia e a repetição de seções.”
2. **Cabeçalho:** “Use astrofy-design audit em src/components/Header.astro e examine a navegação, o foco e a leitura no celular.”
3. **Tokens:** “Use astrofy-design audit nestas duas páginas e compare as cores e as fontes com o design.md existente.”
4. **Formulário:** “Use astrofy-design audit no formulário de contato e confira os estados de erro, envio e sucesso sem editar os arquivos.”
5. **Relatório:** “Use astrofy-design audit na página de preços e salve o diagnóstico em docs/revisao-visual.md, preservando a implementação.”

Redesign: mudar a composição existente

O modo `redesign` recebe uma interface existente e a mudança visual desejada. Ele preserva textos aprovados, rotas, integrações e funcionalidades dentro do escopo solicitado. Num conjunto de páginas, o contrato visual comum orienta todas as mudanças.

Entrada	Tipo e obrigatoriedade	Uso
Alvo	Caminho ou conjunto de caminhos necessário	Define a interface existente
Mudança desejada	Texto necessário para delimitar a intervenção	Explica o resultado visual esperado
Elementos a preservar	Texto ou contrato existente	Explicita as restrições da mudança
`--mood`	Texto opcional na conversa	Sugere um tom visual, sem substituir a marca

A notação `--mood` pertence ao pedido dirigido ao agente, não ao parser do CLI. Na ausência dessa preferência, o agente usa a direção já confirmada ou pergunta somente o que falta para propor a composição. Uma escolha anterior compatível não precisa ser confirmada novamente.

A entrega inclui os arquivos alterados, a explicação da composição e a conferência do resultado. O trabalho pode atualizar o histórico visual. Uma revisão de várias páginas pode estabelecer `design.md` quando esse contrato fizer parte do escopo aprovado. O agente preserva o documento existente e não remove rotas ou diretórios apenas para simplificar a implementação.

Exemplos de redesign

1. **Produto:** “Use astrofy-design redesign na página de produto para dar mais espaço à demonstração, preservando textos e formulário.”

2. **Cabeçalho:** “Use astrofy-design redesign no cabeçalho mobile, mantendo os links, os destinos e o comportamento de autenticação.”
3. **Duas rotas:** “Use astrofy-design redesign nas páginas de produto e preços com o mesmo padrão de títulos, botões e espaçamento.”
4. **Preferência visual:** “Use astrofy-design redesign na página do evento com –mood editorial, respeitando as cores e fontes da marca.”
5. **Redução de movimento:** “Use astrofy-design redesign nesta seção para reduzir animações e manter a leitura com prefers-reduced-motion.”

Arquivos de consulta e arquivos alterados

A instalação da skill e a execução de um trabalho visual têm efeitos diferentes. A tabela separa os arquivos de instrução daqueles que pertencem à interface ou ao histórico do projeto.

Caminho	Função	Quando muda
`.agents/skills/astrofy-design/`	Biblioteca instalada no Codex	Instalação ou atualização das skills
`.claude/skills/astrofy-design/`	Biblioteca instalada no Claude	Instalação ou atualização das skills
`.astrofy/config/paths.json`	Caminhos de design e CSS do projeto	Configuração explícita do projeto
`.astrofy/design/log.json`	Até vinte composições recentes	Implementação com build ou redesign
`design.md` ou `DESIGN.md`	Contrato visual portátil	Exportação solicitada ou escopo aprovado de várias páginas
JSON de design e CSS gerado	Tokens e resultado do gerador	Mudança de tokens dentro do escopo
Rotas e componentes indicados	Interface do site	Build ou redesign autorizado

Cada entrada do histórico registra `date`, `macrostructure`, `theme`, `enrichment` e `brief`. Temas personalizados acrescentam `theme_axes` e podem informar `vibe`. Uma execução sobre várias páginas pode usar `scope: app`. Se o JSON estiver inválido, o agente deve preservá-lo e informar o problema. Esse histórico é escrito pelo agente conforme a skill, não por um comando automático de design do CLI.

Tokens e verificações técnicas

Em um projeto inicializado, valide a fonte dos tokens antes de gerar CSS. A simulação permite conferir a operação antes da escrita:

```
astrofy tokens validate
astrofy tokens build --dry-run
astrofy tokens build
astrofy tokens check
astrofy check --rule design.schema
```

Os comandos usam os caminhos configurados pelo projeto. O Astrofy espera um envelope próprio para os tokens, por isso uma exportação DTCG genérica precisa ser adaptada antes de ser usada como entrada. O arquivo CSS gerado não deve receber correções manuais que desapareceriam na geração seguinte.

O build do site e a inspeção no navegador completam essa conferência. A skill prevê larguras de 320, 375, 414 e 768 pixels, além de desktop. Confira teclado, foco, contraste, zoom, tema e redução de movimento conforme os componentes alterados. Uma validação não executada precisa permanecer como pendência.