DESIGNFY

# Guia do Usuário

Design system, componentes e telas que saem desenhadas.

O método e as peças do Designfy pelo terminal, pelas skills e pelo MCP.

Edição v0.2.1
Gerado em 25 de novembro de 2026
Fonte editorial: docs/user/

# Sumário


Designfy · Guia do Usuário ..... 6
- Percurso ..... 6
- Documentação técnica ..... 7
- Onde encontrar o projeto ..... 7

Capítulo 1: Introdução e instalação ..... 8
- O que o Designfy faz ..... 8
- Três modos no mesmo executável ..... 8
- Requisitos ..... 9
- Instalar com o npm ..... 9
- Instalar pelo binário da release ..... 9
- Instalar a skill no agente ..... 10
- Do código-fonte ..... 10
- Conferir a instalação ..... 10
- Ajuda embutida ..... 12
- Conferir ..... 12

Capítulo 2: Preparar o projeto ..... 14
- Rodar o init ..... 14
- Ler o estado ..... 16
- Projetos em Tailwind 3 ..... 17
- Rodar a partir de outra pasta ..... 17
- Bibliotecas sugeridas ..... 18
- Conferir o que falta ..... 19
- Instalar ..... 19
- Adotar um projeto que já tem design ..... 20
- Conferir ..... 21

Capítulo 3: Skills ..... 22
- As onze skills internas ..... 22
- O que cada skill faz ..... 23
- Estudos de páginas e de telas ..... 25
- O fluxo da skill `designfy` ..... 25
- Três pedidos de exemplo ..... 26
- As skills externas ..... 26
- Instalar e listar ..... 27
- Conferir ..... 27

Capítulo 4: Viewspec ..... 28
Dois níveis de arquivo ..... 28
O guia do projeto ..... 28
Regras com origem e identificador ..... 29
Aprendizados ..... 30
O viewspec de uma tela ..... 30
Projeto com Astrofy ou Specsfy ..... 32
Modelos e exemplos ..... 33
A checklist ..... 34
O ciclo do status ..... 34
Ler e validar ..... 34
Conferir ..... 36
Capítulo 5: Tema e paleta ..... 37
O arquivo de tokens ..... 37
Ler o tema do projeto ..... 38
Conferir contraste ..... 40
A roda de cores ..... 41
Gerar a paleta ..... 42
Medir um par ..... 43
Claro e escuro ..... 44
Conferir ..... 44
Capítulo 6: Catálogo e componentes ..... 45
O catálogo de interface ..... 45
O registro de componentes ..... 47
Conferir um item ..... 49
Copiar para o projeto ..... 50
shadcn e ReUI ..... 51
A regra de tema único ..... 51
Conferir ..... 52
Capítulo 7: Fundos e movimento ..... 53
Três níveis de fundo ..... 53
O componente Background ..... 53
Escolher um preset ..... 54
Movimento é o padrão ..... 55
Movimento com o Motion ..... 56
Conferir ..... 57

Capítulo 8: Capturas e revisão . . . . . 58
Instalar o Playwright . . . . . 58
Capturar . . . . . 58
Conjunto mínimo de uma revisão . . . . . 60
O manifesto . . . . . 60
Listar execuções . . . . . 61
Como a revisão usa as capturas . . . . . 62
Revisões, gravidade e veredito . . . . . 62
Fechar a tela . . . . . 63
Captura em CI . . . . . 63
Conferir . . . . . 64
Capítulo 9: TUI . . . . . 65
Abrir . . . . . 65
Cabeçalho e Home . . . . . 65
Quadro da Home . . . . . 66
Os módulos . . . . . 68
Teclas globais . . . . . 70
Listas, filtros e planos . . . . . 70
Tema, ícones e mouse . . . . . 71
Tamanhos . . . . . 71
Conferir . . . . . 71
Capítulo 10: Agentes e MCP . . . . . 72
O servidor . . . . . 72
As tools . . . . . 73
Modo de leitura . . . . . 74
Resources e prompts . . . . . 75
Configurar os clientes . . . . . 75
Testar . . . . . 76
Conferir . . . . . 77
Capítulo 11: Referência . . . . . 78
Preferências . . . . . 78
O designfy.json . . . . . 79
Arquivos e pastas . . . . . 80
Exit codes . . . . . 81
Variáveis de ambiente . . . . . 81
Solução de problemas . . . . . 82

Onde encontrar o projeto . . . . . 83
Conferir . . . . . 83

# Designfy · Guia do Usuário



Logo do Designfy

O Designfy é um framework visual para projetos web. Ele reúne o método e as peças para uma interface sair desenhada: uma skill orquestradora que recebe o pedido, o viewspec que especifica cada tela, um catálogo de fragmentos, um registro de componentes React e Astro sobre Tailwind, shadcn e Motion, a paleta pela roda de cores e a captura das telas em celular e desktop para a revisão. O mesmo binário atende a pessoa no terminal, os scripts de CI e o agente por MCP.

Este guia acompanha a versão 0.2.0. Os exemplos usam comandos reais do binário. Cada option citada existe no `designfy --help` da mesma versão.

## Percurso

Leia na ordem na primeira vez. Depois, cada capítulo funciona como consulta:


1. Introdução e instalação
2. Preparar o projeto
3. Skills
4. Viewspec
5. Tema e paleta
6. Catálogo e componentes
7. Fundos e movimento
8. Capturas e revisão
9. TUI
10. Agentes e MCP
11. Referência


Os capítulos 1 e 2 levam você da instalação ao projeto preparado, com as bibliotecas conferidas, inclusive num projeto que já tem design. O 3 apresenta as skills, os estudos de páginas e de telas e como chamar a orquestradora. O 4 explica o guia da raiz, o viewspec de cada tela e o vínculo com o Astrofy e o Specsfy. O 5 e o 6 cuidam do tema, da paleta, do catálogo e dos componentes.

O 7 cobre fundos e movimento. O 8 mostra a captura com Playwright, a página inteira e a revisão. O 9 percorre a Home em menu e os módulos da TUI. O 10 liga o binário ao agente por MCP. O 11 reúne preferências, arquivos, exit codes e solução de problemas.

## Documentação técnica

A pasta docs/ guarda a documentação que viaja dentro do executável, com o contrato do viewspec, a referência do CLI, os módulos da TUI e as tools MCP. O `designfy --skill` imprime essa documentação em Markdown.

## Onde encontrar o projeto

O código, os binários e as notas de cada versão ficam no repositório público promovaweb/designfy. O pacote do npm é `@promovaweb/designfy`. A versão online deste guia e a documentação técnica ficam em promovaweb.com/docs/designfy; a apresentação do projeto, em promovaweb.com/designfy.

# Capítulo 1: Introdução e instalação

Você abre o projeto Astro que acabou de clonar e roda `designfy status`. A resposta cabe numa tela: framework `astro`, Tailwind 4, `npm` como gerenciador, nenhuma skill de projeto instalada, `0001` como próximo viewspec e o caminho do CSS do tema. A partir dali, cada pedido de tela ganha um arquivo de especificação, um registro de componentes e uma captura para conferir o resultado.

## O que o Designfy faz

Um site feito por agente costuma sair com a mesma cara: uma paleta improvisada, tipografia padrão, um componente novo a cada página, nada se move e o celular fica para depois. O Designfy reúne o método e as peças para a interface sair desenhada. Ele não desenha sozinho. Ele prepara o projeto, entrega o catálogo de fragmentos, gera a paleta, copia componentes que leem o tema e captura as telas para a revisão.

O trabalho acontece em quatro frentes:

| FRENTE | O QUE ENTREGA |
|---|---|
| Projeto | `designfy.json` com framework, gerenciador de pacotes, Tailwind, shadcn e caminhos |
| Especificação | o guia `VIEWSPEC.md` da raiz e um viewspec por tela, com checklist e estado |
| Peças | registro de componentes React e Astro, catálogo de interface, paleta e fundos |
| Conferência | `theme check`, `deps check`, `viewspec validate` e capturas com Playwright |

A skill orquestradora `designfy` recebe o pedido, lê o projeto, monta o viewspec e chama as especialistas na ordem. O binário é a mão dela no terminal: detecta o projeto, copia arquivos, mede contraste e abre o navegador. A mesma base atende a pessoa no terminal, os scripts de CI e o agente por MCP.

## Três modos no mesmo executável

O `designfy` tem três formas de uso, com a mesma configuração em `~/.designfy/`:

| MODO | COMO ENTRAR | PÚBLICO |
|---|---|---|
| TUI | `designfy` sem argumentos, num terminal interativo | pessoas |

| MODO | COMO ENTRAR | PÚBLICO |
| --- | --- | --- |
| CLI | `designfy <command> [options]`, com texto ou `--json` | pessoas, scripts e CI |
| MCP | `designfy mcp`, servidor Model Context Protocol via stdio | agentes de IA |

O CLI devolve dados em stdout, avisos em stderr e um exit code fixo por família de falha. Com `--json`, a resposta vem no envelope `{ ok, command, data }`, que um script lê sem interpretar texto.

## Requisitos

- **Node.js 18.17 ou superior** para três trabalhos: baixar o binário pelo pacote do npm, instalar as skills com `npx skills` e executar o Playwright num processo Node do projeto durante a captura.
- **Bun 1.4.4 ou superior** só para rodar o código-fonte ou compilar o binário. A versão mínima está em `package.json` e o `designfy doctor` a confere. O executável da release já leva o runtime.
- **Acesso ao GitHub** para o `npx skills add` buscar as skills em `promovaweb/designfy`, quando você instalar a skill à mão.

## Instalar com o npm

```
npm i -g @promovaweb/designfy
designfy --version
```

O pacote do npm é um invólucro fino. Ele baixa o binário da release da mesma versão para `~/.designfy/bin/v<versão>/`, confere o SHA-256 e o mantém em cache; as execuções seguintes usam o arquivo guardado. O `postinstall` adianta o download e a primeira execução o repete se o cache estiver vazio.

## Instalar pelo binário da release

Baixe o executável da sua plataforma em Releases: `designfy-linux-x64`, `designfy-linux-arm64`, `designfy-darwin-x64`, `designfy-darwin-arm64` ou `designfy-windows-x64.exe`. Confira o arquivo com o `SHA256SUMS` da mesma release, dê permissão de execução e coloque numa pasta do `PATH`.

## Instalar a skill no agente

```
npx skills add promovaweb/designfy
```

O comando instala as skills que o agente lê. Dentro de um projeto, o caminho completo é o `designfy skills install`, do capítulo 3, que escolhe as skills compatíveis com o framework detectado.

## Do código-fonte

Para desenvolver o Designfy ou gerar o binário da sua plataforma:

```
git clone https://github.com/promovaweb/designfy.git
cd designfy
bun install
bun run build
./dist/designfy --version
```

O build gera `dist/designfy` para a plataforma da máquina. Para os cinco alvos de release, use `bun run build:all`. O ambiente de desenvolvimento, os testes e a validação estão no CONTRIBUTING.md.

## Conferir a instalação

Três comandos confirmam que o binário responde e que a pasta de dados pode ser criada:

```
designfy --version
designfy about
designfy doctor
```

O `--version` imprime a versão:

```
designfy 0.2.0
```

O `about` mostra o executável, o runtime, a plataforma, a pasta de dados, a licença e os créditos:

```
Designfy v0.2.0
Visual framework for web projects: design system, React and Astro components with
Tailwind, shadcn and motion, skills, dependency and MCP setup, and Playwright screen
captures.

Binary

  Executable  ~/.bun/bin/bun
  Runtime     Bun 1.4.4
  Platform    linux-x64
  PID         3904428
  Data        ~/.designfy
  License     AGPL-3.0-only

Project

  Author      Promovaweb
  Site        https://promovaweb.com
  Repository  https://github.com/promovaweb/designfy

Credits

  Bun                 runtime, bundler and binary compiler https://bun.sh
  OpenTUI             terminal user interface rendering https://opentui.com
  MCP TypeScript SDK  Model Context Protocol server over stdio https://
modelcontextprotocol.io
  OpenCLI             standard description of the command-line interface https://
opencli.org
  Zod                 schema validation for MCP tools https://zod.dev
```

O `doctor` confere o ambiente sem gravar arquivo. Numa máquina nova, a pasta `~/.designfy/` ainda não existe e o resultado mostra `!` nas verificações que dependem dela:

```
✓ bun: 1.4.4; minimum 1.4.4
! stdin: Not a TTY
! stdout: Not a TTY
! directory: Not created
! settings: Not created
! master-key: none
! credentials: Not created
✓ node: /usr/bin/node
✓ npx: ~/.npm-global/bin/npx
✓ designfy.json: Project file exists
! VIEWSPEC.md: Not created
✓ tailwind-theme: /tmp/opencode/designfy-demo/src/styles/app.css
✓ binary: designfy 0.2.0
✓ platform: linux-x64
```

O `✓` é uma verificação aprovada, o `!` é um aviso e o `✗` reprova o comando com exit code 3. Os avisos de `~/.designfy/` somem quando você grava a primeira preferência. `VIEWSPEC.md` passa a `✓` quando o guia é criado; num projeto sem `designfy.json` ou sem CSS do Tailwind, essas duas linhas também aparecem com `!` até o capítulo 2.

## Ajuda embutida

Todo comando aceita `--help`, com arguments, options, exemplos e os exit codes que ele pode devolver:

```
designfy capture run --help
designfy components add --help
```

O binário também carrega a documentação escrita para agentes. O `designfy --skill` imprime tudo em Markdown; `designfy --skill cli` imprime só a seção do CLI, com a referência gerada do registry. Os tópicos são `cli`, `tui`, `mcp`, `config`, `exit-codes` e `reference`. A documentação viaja dentro do executável, então o agente lê a versão exata instalada na máquina.

A pasta `docs/` do repositório é a fonte dessa documentação. Este guia é outro texto: ele explica o uso desde o primeiro contato com o produto, com cenas de resultado, enquanto `docs/` é a referência técnica do binário.

## Conferir

- `designfy --version` responde com a versão do binário.
- `designfy about` mostra runtime, plataforma, pasta de dados e licença.
- `designfy doctor` termina com exit code 0 e sem `✗`.

- `designfy --help` lista os commands e as options globais.
- `designfy --skill cli` imprime a documentação embutida do CLI.

# Capítulo 2: Preparar o projeto

O `init` roda na raiz de um projeto Astro e grava um arquivo pequeno:

```
Wrote /tmp/opencode/designfy-demo/designfy.json (astro).
```

A partir desse arquivo, o binário sabe qual framework ler, onde copiar componentes, onde procurar o CSS do tema e qual alias usar nos imports. O `status` mostra a mesma detecção em forma de painel, com o que já está instalado e o que falta. Sem esses dois comandos, cada comando seguinte adivinharia caminhos e o agente repetiria a mesma leitura a cada pedido.

## Rodar o init

```
designfy init
```

O `init` lê o `package.json` e detecta:

| O QUE | COMO |
|---|---|
| Framework | `astro`, `next`, `react-vite`, `react` ou `unknown` |
| Tailwind | a versão maior de `tailwindcss`, ou `null` |
| shadcn | a presença de `components.json` |
| Gerenciador | o lockfile, na ordem `bun.lock`, `bun.lockb`, `pnpm-lock.yaml`, `yarn.lock`, `package-lock.json`; sem lockfile, `npm` |
| TypeScript | o `tsconfig.json` ou a dependência `typescript` |

O arquivo gravado tem os campos abaixo.

| CAMPO | VALOR NO EXEMPLO |
|---|---|
| `schemaVersion` | `1` |
| `framework` | `astro` |
| `typescript` | `false` |
| `packageManager` | `npm` |
| `tailwind` | `4` |
| `shadcn` | `false` |
| `paths.components` | `src/components/designfy` |

| CAMPO | VALOR NO EXEMPLO |
| --- | --- |
| `paths.styles` | `src/styles` |
| `paths.viewspec` | `viewspec` |
| `aliases.components` | `@/components/designfy` |

O `designfy.json` do projeto de exemplo fica assim:

```
{
  "framework": "astro",
  "typescript": false,
  "packageManager": "npm",
  "tailwind": 4,
  "shadcn": false,
  "schemaVersion": 1,
  "paths": {
    "components": "src/components/designfy",
    "styles": "src/styles",
    "viewspec": "viewspec"
  },
  "aliases": {
    "components": "@/components/designfy"
  }
}
```

Três options corrigem a detecção ou o destino:

```
designfy init --framework next --components-path src/ui --styles-path app/styles
```

O `--dry-run` imprime a proposta sem gravar. Um `designfy.json` existente faz o comando parar com exit code 2:

```
error designfy.json already exists. Use --force to overwrite it. [2 USAGE]
Use "designfy init --help".
```

Use `--force` só quando quiser substituir o arquivo inteiro. Sem `package.json` na pasta, a resposta é `error No package.json found in the current directory. [6 NOT_FOUND]`. Rode o comando na raiz do site ou da aplicação, não numa pasta interna.

Depois do primeiro `init`, edite o `designfy.json` à mão quando o projeto mudar de pasta de componentes ou de alias. O arquivo é a fonte dos caminhos para o `components add`, para o `theme show` e para o `viewspec`.

Quando a busca automática não encontra o CSS certo, o `--theme-path` grava o caminho no arquivo:

```
designfy init --theme-path src/styles/tokens.css
```

O valor entra em `paths.theme` e o `theme show` e o `theme check` passam a ler esse arquivo antes de procurar. O capítulo 5 mostra a ordem completa de escolha.

## Ler o estado

```
designfy status
```

A saída do projeto de exemplo:

```
  Framework             astro
  Detected framework    astro
  TypeScript            no
  Package manager       npm
  Tailwind major        4
  shadcn                no
  Skills installed      1
  Dependencies present  3
  Components copied     0
  Viewspecs             0
    draft               0
    in_progress         0
    blocked             0
    done                0
    archived            0
  Next viewspec ID      0001
  Viewspec guide        not found
  Tailwind theme        not found
```

O painel separa a detecção atual ( `Detected framework` ) da configuração em vigor, que vem do `designfy.json` . Quando os dois divergem, o arquivo vale. `Skills installed` mostra o total de skills do catálogo já instaladas no projeto ou no escopo global. `Components copied` mostra quantos arquivos do registro já estão na pasta do projeto. `Viewspec guide` diz se o `VIEWSPEC.md` da raiz existe. `Tailwind theme` mostra `not found` até o tema ser copiado; depois do `components add designfy-theme` , ela aponta o CSS de entrada.

O `status --json` entrega o mesmo conteúdo em `data` , com os objetos `detected` e `project` completos. Um agente usa essa forma para não reler o `designfy.json` :

```
designfy status --json
```

O `status` não grava nada e não precisa de rede. Rode depois de cada instalação para confirmar o efeito.

## Projetos em Tailwind 3

O Designfy lê o tema de um projeto que ainda está no Tailwind 3. A busca aceita o CSS com `@tailwind base` ou `@import "tailwindcss/base"` e completa os tokens com `tailwind.config.*` na raiz: cores aninhadas, fontes, raios, sombras, telas, curvas e keyframes, inclusive valores como `rgb(var(--x) / <alpha-value>)` e `hsl(var(--x))`.

```
designfy theme show
designfy status
designfy doctor
```

Os três informam a versão do Tailwind e os arquivos lidos. Um aviso aparece quando a configuração não carrega e o resultado traz só os tokens do CSS. O comando executa o módulo de configuração, então use em projeto cujo código você confia.

O registro de componentes usa a sintaxe do Tailwind 4. Por isso o `components add` recusa um projeto Tailwind 3 com exit code 10 e explica a migração. O `--allow-tailwind3` copia os arquivos mesmo assim, com aviso:

```
designfy components add site-header --allow-tailwind3
```

Não rode `designfy deps install` sem grupo num projeto que já tem Tailwind: o grupo `core` traz o Tailwind 4. No Tailwind 3, use o item do registro como referência de estrutura, de estados e de movimento ao escrever o componente no padrão do projeto.

## Rodar a partir de outra pasta

A option global `--project` ( `-C` ) executa qualquer command com a raiz do site indicada, sem trocar de pasta antes:

```
designfy -C ./website status
designfy --project ../app capture run http://localhost:4321/ --viewport mobile
```

A variável `DESIGNFY_PROJECT` define o padrão quando a option não aparece. A option vence a variável. O command usa a pasta atual quando as duas faltam. Caminhos relativos passados aos commands, como `theme show --file`, partem da raiz escolhida. Uma pasta inexistente sai com exit code 6. No servidor MCP, a mesma escolha vale por chamada, pelo argumento `project`.

## Bibliotecas sugeridas

O catálogo de dependências agrupa os pacotes por assunto. Cada grupo tem um identificador, um título e a lista de pacotes com a faixa de versão, o motivo e a URL oficial.

```
designfy deps list
```

Cada linha traz o grupo, o pacote, a faixa do catálogo, o estado no `package.json` e a versão instalada. Uma linha aprovada do exemplo é `core / tailwindcss ^4.3.5 ok ^4.3.5`. Os estados são `ok`, `missing` e `outdated`. `outdated` aparece quando a versão instalada é de uma versão maior anterior. O `--group <id>` restringe a lista:

```
designfy deps list --group motion
```

Os grupos desta versão são:

| GRUPO | ASSUNTO | PADRÃO |
|---|---|---|
| `core` | Tailwind 4, `cn`, variantes e ícones | sim |
| `vite` | plugin do Tailwind para Astro e React com Vite | sim |
| `motion` | Motion para React e vanilla | sim |
| `typography` | classes de prosa e as fontes Inter e Manrope | sim |
| `shadcn` | o CLI do shadcn e as primitivas Radix | sim |
| `base-ui` | primitivas do Base UI, alternativa ao Radix | não |
| `astro` | ilhas React e MDX dentro do Astro | sim |
| `forms` | estado de formulário, validação e campos especiais | não |
| `tables` | tabelas com ordenação, filtro e virtualização | não |
| `charts` | gráficos e contadores animados | não |
| `feedback` | toasts, paleta de comandos e drawers | não |
| `media` | embeds de vídeo e carrossel | não |

| GRUPO | ASSUNTO | PADRÃO |
| --- | --- | --- |
| `astro-embed` | componentes de embed para Astro e MDX | não |
| `content` | destaque de código e renderização de Markdown | não |
| `state` | estado de URL, estado de servidor e painéis | não |
| `scroll` | rolagem suave para páginas longas | não |
| `color` | conversão de cor no código do projeto | não |
| `testing` | Playwright e auditoria de acessibilidade | não |

## Conferir o que falta

```
designfy deps check
```

O `deps check` compara o `package.json` com os grupos padrão compatíveis com o framework e com os pacotes exigidos pelos componentes já copiados. Quando falta algo, a saída resume a lista e termina com exit code 10:

```
error Missing: tw-animate-css, class-variance-authority, lucide-react, @tailwindcss/vite, motion, @tailwindcss/typography, @fontsource-variable/inter, @fontsource-variable/manrope, shadcn, radix-ui, @astrojs/react, @astrojs/mdx, react, react-dom. [10 CHECK_FAILED]
```

Um `deps check` com exit code 0 significa que os grupos conferidos estão completos. Use `--group <id>` para conferir um grupo só e `--json` para ler `data` num script.

## Instalar

```
designfy deps install
```

Sem argumentos, o comando instala o que falta nos grupos padrão compatíveis com o framework. Com argumentos, aceita identificadores de grupo ou nomes de pacote:

```
designfy deps install motion
designfy deps install tables feedback
```

O gerenciador vem do `designfy.json`, com `--pm` para forçar outro:

```
designfy deps install --pm bun
```

O plano aparece antes de qualquer download:

```
designfy deps install --dry-run
```

```
npm install 'tw-animate-css@^1.4.0' 'class-variance-authority@^0.7.1' 'lucide-
react@^1.79.0' '@tailwindcss/vite@^4.3.5' 'motion@^14.14.0' '@tailwindcss/
typography@^0.5.20' '@fontsource-variable/inter@^5.3.0' '@fontsource-variable/
manrope@^5.3.0' 'radix-ui@^1.8.1' '@astrojs/react@^7.1.9' '@astrojs/mdx@^8.1.6'
'react@^19.3.1' 'react-dom@^19.3.1'
npm install -D 'shadcn@^4.23.0'
```

O comando separa dependências de produção das de desenvolvimento, como mostra a linha com `-D`. Uma falha do gerenciador sai com exit code 5 e devolve o comando executado em `details`.

## Adotar um projeto que já tem design

Quando o projeto já tem telas publicadas, tema próprio ou regras de design escritas, o primeiro resultado não é tela nova. É o guia `VIEWSPEC.md` descrevendo o projeto como ele é, com a fonte de cada regra. Nada de tema novo, Tailwind novo, componente copiado ou dependência instalada nesse momento.

Os sinais de que o caminho de adoção vale: páginas ou telas no ar, CSS de tema ou `tailwind.config.*`, documento de marca ou de componentes, regras de interface no `AGENTS.md`, no `CLAUDE.md` ou no `README.md` e monorepo com a pasta do projeto dentro de outro repositório que tem instruções próprias.

A sequência:

1. **Ler as instruções.** Abra o `AGENTS.md`, o `CLAUDE.md` e o `README.md` do projeto e, em monorepo, também os da raiz do repositório pai.
2. **Encontrar os documentos de design.** Procure por arquivos com `design`, `brand`, `ui` ou `componentes` no nome e leia as seções de cores, tipografia, layout, movimento, componentes, páginas e acessibilidade.
3. **Ler o tema real.** Rode `designfy theme show --json` e `designfy status --json` e anote a versão do Tailwind, o arquivo de tokens e como o modo escuro é ligado. Projeto antigo usa

escala própria ( `primary-500` ). Nesse caso, o guia descreve a escala do projeto, sem trocar pelos papéis do shadcn.

4. **Ler as telas.** Liste os componentes e leia duas ou três telas de cada tipo. Capture antes de mudar, para ter a linha de base: `designfy capture run http://localhost:4321/ --viewport mobile,desktop`.
5. **Medir a compatibilidade com o registro.** Com Tailwind 4 e papéis do shadcn, o `components add` funciona direto. Com escala própria, copie e troque as classes ou mapeie os papéis no `@theme inline`. No Tailwind 3, o item serve de referência.
6. **Escrever o guia.** Chame a `designfy-viewspec` no modo guia com o que foi levantado. O `tokensFile` aponta para o CSS que o projeto já usa.
7. **Configurar sem mexer no build.** Rode `designfy init` e ajuste `components`, `styles` e `viewspec` no `designfy.json`.

Regra do projeto prevalece sobre o padrão de uma especialista: siga o projeto e registre a diferença em `## Aprendizados`, citando a skill. O processo do projeto também continua, com sitespec, spec, issue, validações e convenções de commit. Quando uma fonte já responde, não pergunte de novo.

## Conferir

- `designfy.json` existe na raiz e `designfy status` mostra o framework certo.
- `designfy status --json` traz `detected` e `project` em `data`.
- `designfy deps list` mostra `ok` para os pacotes centrais que o projeto já tem.
- `designfy deps check` termina com exit code 0 ou lista exatamente o que falta.
- `designfy deps install --dry-run` imprime os comandos antes de instalar.
- `designfy -C <pasta> status` responde pelo projeto indicado.
- Num projeto em Tailwind 3, `theme show` informa a versão e os arquivos lidos.

# Capítulo 3: Skills

O pedido chega ao agente com uma frase: “cria a página de preços do produto”. O agente abre a skill `designfy`, lê o projeto e o guia, escreve o viewspec da tela e chama as especialistas na ordem. No fim, a página existe, a checklist está marcada e as capturas foram revisadas. Nenhuma etapa mora no binário: todas moram nas skills que o agente lê.

## As onze skills internas

O catálogo tem onze skills `designfy*`, todas com origem em `promovaweb/designfy` e marcadas como padrão.

| SKILL | GRUPO | QUANDO ENTRA | ENTREGA |
|---|---|---|---|
| `designfy` | core | todo pedido de interface | viewspec, interface pronta, revisão e relato |
| `designfy-viewspec` | core | guia, tela nova, retomada ou validação | os dois níveis do viewspec e o estado |
| `designfy-design-system` | foundation | tema ausente, paleta, tipo, fundo e forma | arquivo de tokens e contraste medido |
| `designfy-components` | foundation | escolher, copiar, compor ou criar peça | componentes no projeto, com estados |
| `designfy-motion` | foundation | entrada, rolagem, hover e troca de estado | movimento com versão parada |
| `designfy-websites` | surfaces | site, landing page e página pública | página montada como argumento |
| `designfy-content` | surfaces | blog, artigo e documentação | leitura confortável e páginas ao redor |
| `designfy-webapps` | surfaces | dashboard, tabela, formulário e CRUD | tela de trabalho com todos os estados |
| `designfy-responsive` | quality | celular, tablet e tela larga | plano por faixa e estouro corrigido |
| `designfy-accessibility` | quality | WCAG 2.2 nível AA | auditoria, correções e relatório |
| `designfy-review` | quality | fechamento de um viewspec | capturas, achados e veredito |

Até a versão 0.1.4, a skill `designfy-webapps` se chamava `designfy-systems`. O nome antigo saiu do catálogo; use o novo nas chamadas, no `skills-lock.json` e nas referências do guia.

## O que cada skill faz

### `designfy`

É a porta de entrada. Lê o projeto e o guia, classifica o pedido em superfície, tipo de tela, rota e ação principal, escolhe os fragmentos e monta o viewspec. Lê as referências de leitura do pedido, roteamento, fluxo, preparo, entrega e uso do binário. Entrega a interface implementada, a checklist executada, as capturas revisadas e o relato final.

### `designfy-viewspec`

Escreve e mantém o guia `VIEWSPEC.md` da raiz e o viewspec de cada tela em `viewspec/<id>-<slug>/VIEWSPEC.md`, com as nove seções e a checklist. É chamada para criar o guia, especificar uma tela, retomar um trabalho aberto ou validar arquivos. Lê o contrato, o guia do projeto, o viewspec da tela, as checklists por tipo, a execução e os erros de validação. Também usa os modelos e exemplos da skill. Entrega o arquivo válido, com o progresso e as notas.

### `designfy-design-system`

Cuida do arquivo de tokens que todos os componentes leem: cor, tipografia, raios, espaçamento, fundos, sombras e temas. Entra quando o projeto não tem tema, quando `designfy theme check` reprova ou quando o pedido muda a identidade visual. Lê tokens, a roda de cores, contraste, tipografia, espaçamento, fundos e o catálogo de padrões, formas e modo escuro. Entrega o tema ligado ao build, a paleta com a harmonia registrada, os pares medidos e o guia atualizado.

### `designfy-components`

Resolve de onde vem cada peça da tela e como ela se comporta. Entra quando a região precisa de componente, quando é hora de instalar shadcn ou ReUI e quando um componente novo precisa seguir as convenções do registro. Lê o catálogo de interface, fontes e instalação, o registro, shadcn e ReUI, componentização, estados, Tailwind, React, Astro e bibliotecas sugeridas. Entrega os arquivos no projeto, as dependências resolvidas, os estados cobertos e os componentes novos documentados.

### `designfy-motion`

Define o movimento de cada parte da tela. Entra em toda tela, porque movimento é o padrão. Dedica mais atenção a telas com entrada, revelação ao rolar, laço ou troca de estado. Lê princípios, movimento por tipo de tela, utilidades do Tailwind e do Designfy, CSS moderno, `motion/react`, receitas, movimento reduzido e desempenho. Entrega a lista do que se move com gatilho, duração e curva, a implementação na ferramenta mais leve e a captura parada.

### `designfy-websites`

Monta páginas públicas como um argumento, uma ideia por tela. Entra em landing page, home, página de produto, preços, sobre, contato, hero, cabeçalho ou rodapé. Lê a arquitetura da página, o hero, as seções, a navegação e o rodapé, a conversão, o texto, as imagens e os vícios de página gerada. Entrega a ordem das seções com a pergunta que cada uma responde, o fragmento e a fonte de cada uma, as lacunas de conteúdo e o que fica para as outras especialistas.

### `designfy-content`

Cuida da leitura em blogs, artigos, guias e documentação. Entra em artigo, listagem, categoria, documentação e embed de mídia. Lê a diagramação do texto, a página de artigo, a listagem, os elementos do texto, imagens e embeds, documentação e Markdown ou MDX. Entrega a medida e a escala, os elementos tratados, a listagem no ar e o comportamento de tabela e código no celular.

### `designfy-webapps`

Desenha telas de trabalho em React com shadcn, ReUI e Tailwind. Entra em dashboard, tabela, formulário, CRUD, detalhe, configurações e autenticação. Lê o shell e a navegação, as tabelas, os formulários, os dashboards e contadores, o CRUD e o detalhe, as configurações, a autenticação, os estados e avisos e a densidade do texto. Entrega a especificação do dado, a tela composta, todos os estados vistos e o estado de tela na URL.

### `designfy-responsive`

Planeja cada largura em vez de encolher o desktop. Entra em pedido de versão mobile, correção de rolagem horizontal, tabela no celular, menu mobile e ajuste de tela larga. Lê as faixas e pontos de quebra, os padrões de adaptação, a fluidez e as consultas de container, o toque e o celular, o estouro e a conferência com capturas. Entrega o plano por região, as capturas antes e depois e os padrões levados ao guia.

### `designfy-accessibility`

Confere a tela contra a WCAG 2.2 nível AA e corrige o que encontra. Use a skill ao revisar acessibilidade ou auditar contraste, teclado, leitor de tela e ARIA. Ela também participa da revisão final de cada tela. Lê a WCAG na prática, teclado e foco, semântica e nomes, cor e contraste, formulários, componentes interativos, mídia e movimento e a auditoria. Entrega a auditoria automática, o roteiro manual, as correções e o relatório.

### `designfy-review`

Olha a tela pronta com olhos novos. Entra no fechamento de um viewspec, depois da primeira dobra ou antes de publicar. Lê o ciclo de revisão, as capturas, a leitura das imagens, a otimização por largura, a gravidade e o encaminhamento, a revisão por tipo e a lista de acabamento. Entrega as execuções de captura, os achados por gravidade com a dona de cada um, o veredito e o que vai para o guia.

## Estudos de páginas e de telas

Duas skills guardam estudos de interfaces reais, lidas por inteiro: a ordem das seções, a anatomia das peças, os estados, o movimento e o comportamento no celular. `skills/designfy-websites/references/estudos/` traz páginas de site e `skills/designfy-webapps/references/estudos/` traz telas de sistema, no mesmo formato.

O primeiro estudo é a landing de oferta com prazo e turma limitada do Plano Founders, em `landing-de-oferta-founders.md`. Ele lê as 21 seções na ordem, com a pergunta que cada uma responde, o palco de pixels do hero, o relógio regressivo do prazo, o carrossel de depoimentos e a tabela de preço de um plano só. Foi dele que saíram os onze itens novos do registro, no capítulo 6.

A pasta de telas de sistema já traz o formato e ainda não tem estudo. O primeiro entra quando houver uma tela real, em uso, para ler do começo ao fim; até lá, as referências da skill e os apps de `examples/react-system` cobrem as telas comuns.

O estudo serve para o argumento e a composição, não para copiar dados, nomes de clientes ou a marca do produto estudado. Cada estudo termina apontando os itens do registro criados a partir dele.

## O fluxo da skill `designfy`

A orquestração tem oito estágios: ler o projeto, ler o pedido, garantir o guia, escolher os fragmentos, montar o viewspec, executar com as especialistas, revisar com capturas e fechar. O trabalho para em três pontos apenas:

1. **Esclarecimento do pedido.** Uma mensagem, no máximo três perguntas, com a opção recomendada primeiro.
2. **Plano da tela.** Só quando existe mais de uma estrutura plausível. O plano mostra as regiões em ordem, o fragmento de cada uma e o que muda.
3. **Achado que é escolha de produto.** Por exemplo, decidir quais colunas da tabela ficam no celular.

Fora deles, o trabalho corre e você recebe marcos curtos: o que ficou pronto e como conferir.

## Três pedidos de exemplo

**Página de preços.** "Cria a página de preços em `/precos/` com os planos de `content/precos.md`; a ação principal é falar com vendas." A `designfy` classifica como `site-page`, superfície `site`, cria `viewspec/0001-pagina-de-precos/VIEWSPEC.md` e chama `designfy-websites`, `designfy-components`, `designfy-motion`, `designfy-responsive`, `designfy-accessibility` e `designfy-review`.

**Dashboard.** "Redesenha o dashboard de vendas em `/app/vendas`; os dados vêm de `GET /api/sales/summary`; a ação principal é abrir os pedidos em atraso." A classificação é `dashboard` e a superfície é `webapp`. O agente cria o viewspec em `viewspec/0002-dashboard-de-vendas/VIEWSPEC.md`. Rodam `designfy-design-system` se faltar papel de cor, `designfy-webapps`, `designfy-components`, `designfy-motion`, `designfy-responsive`, `designfy-accessibility` e `designfy-review`.

**Artigo do blog.** "Monta a página de artigo com o texto de `content/blog/primeiro-post.md`, índice lateral e embeds de vídeo." A classificação é `blog-post`, superfície `site`, com `viewspec/0003-artigo-do-primeiro-post/VIEWSPEC.md`. Rodam `designfy-content`, `designfy-components`, `designfy-motion`, `designfy-responsive`, `designfy-accessibility` e `designfy-review`.

## As skills externas

O catálogo também instala seis skills de fora:

| SKILL | ORIGEM | QUANDO ENTRA |
|---|---|---|
| `hallmark` | nutlope/hallmark | desenhar página nova com direção de arte e slop test |
| `shadcn` | shadcn/ui | procurar, instalar, corrigir e compor componentes |
| `migrate-radix-to-base` | shadcn/ui | migrar Radix UI para Base UI, só com pedido explícito |
| `astrofy` | promovaweb/astrofy | projeto Astro: arquitetura, páginas, SEO e deploy |
| `specsfy` | promovaweb/specsfy | projeto com specs: a spec da fatia como fonte da interface |
| `brandfy` | promovaweb/brandfy | marca nova: estratégia, logo, paleta e manual |

O `shadcn`, o `migrate-radix-to-base` e o `astrofy` só entram em projeto compatível. O `astrofy`, o `specsfy` e o `brandfy` instalam a biblioteca inteira da origem.

## Instalar e listar

```
designfy skills list
designfy skills install --dry-run
```

O `list` mostra `project`, `global` ou `no` para cada skill. O `install` agrupa as skills por origem e roda um `npx -y skills add` por grupo. O `update` e o `remove` seguem o mesmo caminho e gravam o `skills-lock.json` do projeto.

## Conferir

- `designfy skills list` mostra as onze internas e as seis externas.
- `skills-lock.json` está versionado com o projeto.
- `designfy skills install --dry-run` imprime um comando por origem.
- O relato final cita o viewspec criado e a pasta da captura.

# Capítulo 4: Viewspec

Você abre o projeto pela manhã e roda `designfy viewspec progress`. A tela responde que existe um viewspec em `in_progress` com `2/4` itens marcados e que o próximo `id` livre é `0002`. O agente que retoma o trabalho lê o `VIEWSPEC.md` da tela, vê a checklist e continua do item seguinte. Esse é o efeito do viewspec, a especificação de interface do Designfy.

## Dois níveis de arquivo

O viewspec tem dois níveis, os dois em Markdown e os dois versionados com o código.

| ARQUIVO | ALCANCE | O QUE GUARDA |
|---|---|---|
| `VIEWSPEC.md`, na raiz | o site ou sistema inteiro | o guia de interface: preferências, diretrizes, padrões e aprendizados |
| `viewspec/<id>-<slug>/VIEWSPEC.md` | uma página ou tela | a especificação daquela interface e a checklist da execução |

Toda tela é escrita e revisada contra o guia da raiz. A skill `designfy` lê o guia antes de montar o da tela, a `designfy-viewspec` mantém os dois e o binário só lê e valida. Nenhum command grava viewspec.

A pasta `viewspec/` fica na raiz e o caminho vem de `paths.viewspec`. O nome do diretório é `<id>-<slug>`, com o `id` de quatro dígitos e o `slug` em kebab-case repetidos no front matter.

## O guia do projeto

O guia segue a ideia do formato aberto DESIGN.md: um arquivo de texto com o sistema visual para pessoas e agentes. O Designfy acrescenta padrões de tela, um identificador por regra e o histórico do que foi aprendido.

O front matter declara o escopo, o nome, as superfícies, os frameworks, o arquivo de tokens e os temas:

```
---
schemaVersion: "1.0.0"
scope: project
name: Designfy Demo
description: Landing page de um produto de exemplo usada nos testes do guia.
surfaces: [site]
frameworks: [astro]
tokensFile: src/styles/global.css
themes: [light, dark]
createdAt: 2026-11-24T09:00:00-03:00
updatedAt: 2026-11-24T09:00:00-03:00
---
```

Depois vêm catorze seções fixas, nesta ordem: Visão geral, Cores, Tipografia, Layout, Fundos e profundidade, Formas, Movimento, Voz e conteúdo, Componentes, Padrões de tela, Responsivo, Acessibilidade, Faça e não faça e Aprendizados. Uma seção sem conteúdo aplicável fica com `Não se aplica.` e o motivo.

## Regras com origem e identificador

Uma regra é um item de lista que cabe numa conferência: o que fazer, por quê e de onde veio. O identificador e a origem ficam no fim da linha:

```
- Texto e fundo com contraste de 4.5:1 ou mais, conferido com `designfy theme
check`. `@agente` {#c002}
- A página abre com a promessa em uma frase, sem slogan acima do título. `@pessoa`
{#v001}
```

O `{#id}` é obrigatório e único no arquivo. Uma letra e três dígitos, com a letra seguindo a seção:

| LETRA | SEÇÃO |
|---|---|
| `v` | Visão geral |
| `c` | Cores |
| `y` | Tipografia |
| `l` | Layout |
| `f` | Fundos e profundidade |
| `s` | Formas |
| `m` | Movimento |

| LETRA | SEÇÃO |
|---|---|
| `w` | Voz e conteúdo |
| `k` | Componentes |
| `t` | Padrões de tela |
| `r` | Responsivo |
| `a` | Acessibilidade |
| `d` | Faça e não faça |

A origem é `@pessoa` (indicada pelo responsável pelo projeto), `@agente` (escolhida pelo agente e ainda não contestada) ou `@revisão` (achado em captura ou auditoria). Parágrafo ou item sem `{#id}` é contexto, não regra. Uma regra `@pessoa` só muda com o aceite dela; uma `@agente` pode ser revista, com entrada nova em Aprendizados.

Uma regra boa cabe numa conferência de captura ou de código e diz o que fazer, por qual motivo e com qual origem. “Um só `h1` por tela” é regra; “usar boa hierarquia” não é. “Tabela com mais de quatro colunas vira cartões abaixo de 640 px” é regra; “tabelas responsivas” não é.

Todo guia novo começa com um conjunto pequeno de regras `@agente`: papéis do tema sem cor literal, contraste de 4.5:1 conferido, um `h1` por tela, container com teto, movimento com versão parada, uma ação principal por tela, carregando, vazio e erro em toda região de dado, nenhuma rolagem horizontal entre 320 e 1920 px, alvo de toque de 44 px, foco visível e nada inventado.

## Aprendizados

Aprendizados guarda o histórico com data, origem, o que foi aprendido e o destino: o `{#id}` da regra criada ou alterada, `proposta` enquanto a pessoa não confirmou, ou `descartado` com o motivo.

```
- 2026-11-24 `@pessoa` A página abre com a promessa em uma frase. → {#v001}
- 2026-11-24 `@revisão` O cartão de preço estourava a largura em 390 px. → {#r001}
```

Preserve cada entrada e descreva a mudança em outra que cite a anterior. Quando um aprendizado vira padrão, ele sobe para a seção do assunto e recebe o identificador.

## O viewspec de uma tela

O arquivo de uma tela abre com front matter e nove seções fixas. O exemplo:

```
---
schemaVersion: "1.0.0"
scope: view
id: "0001"
slug: landing-page-do-produto
title: Landing page do produto
summary: Página única de venda que leva o visitante a pedir uma demonstração.
status: in_progress
kind: landing-page
surface: site
framework: astro
route: /produto/
createdAt: 2026-11-24T10:00:00-03:00
updatedAt: 2026-11-24T11:30:00-03:00
skills: [designfy-websites, designfy-components, designfy-motion, designfy-
responsive, designfy-accessibility, designfy-review]
components: [hero-split, feature-grid, cta-band]
relatedFiles:
  - src/pages/produto.astro
guideRules: [v001, c002, m001, r001]
priority: high
---
```

`kind` é o tipo de interface: `landing-page`, `home`, `site-page`, `blog-index`, `blog-post`, `docs-page`, `dashboard`, `table`, `form`, `crud`, `detail`, `settings`, `auth` ou `other`. `surface` é `site` ou `webapp`; o valor antigo `system` continua aceito e vira `webapp` na leitura. `skills` lista as especialistas na ordem de execução. `guideRules` registra as regras do guia que a tela aplica de forma direta e `deviations` registra as que ela não segue, com o motivo em Notas.

Os campos opcionais são `priority` (`low`, `medium` ou `high`), `dependsOn` (IDs que precisam estar `done`), `source`, `blockedReason`, `archivedReason` e `captures` (execuções da revisão).

O `trackedBy` liga o viewspec ao registro da tarefa em outro sistema. Cada vínculo informa `system`, `id` e `path`. O caminho parte da raiz do projeto e aponta para um arquivo que existe:

```
trackedBy:
  - system: specsfy
    id: "0007"
    path: specs/planned/0007-cadastro-de-clientes/spec.md
```

O `system` aceita `astrofy`, `specsfy` ou `other`. O `designfy viewspec validate` confere o arquivo de cada vínculo e mostra o caminho novo quando a spec muda de pasta de estado. A listagem e a consulta individual mostram os vínculos, na CLI e nas tools do MCP.

Numa entrega com várias telas, crie todos os viewspecs antes da primeira implementação e marque a ordem em `dependsOn`: o shell antes das telas, a listagem antes do detalhe.

As nove seções do corpo são Objetivo (o público, o que ele precisa conseguir e a ação principal), Conteúdo (o que aparece e a origem de cada texto, dado e mídia), Estrutura (as regiões de cima para baixo, com o fragmento de cada uma), Estados (vazio, carregando, erro, sucesso, sem permissão, parcial e lotado), Responsivo (o comportamento em 390, 820 e 1440 px), Movimento (o que se move, quando e por quê, com a versão parada), Acessibilidade (marcos, títulos, foco, nomes e contraste), Checklist (as tarefas, com dona e identificador) e Notas (o registro cronológico das escolhas e dos impedimentos).

## Projeto com Astrofy ou Specsfy

O Designfy convive com dois sistemas de especificação da Promovaweb. O `status` e o `doctor` mostram o bloco **Integrations**, com a presença de cada um e os caminhos encontrados:

```
Integrations

  astrofy  present  sitespecDir=.sitespec  astrofyDir=.astrofy
dependency=@promovaweb/astrofy
  specsfy  absent  specsDir=specs
```

O `status --json` entrega o mesmo em `integrations.astrofy` e `integrations.specsfy`. A tool `project_status` expõe os mesmos dados. Caminho ausente vem `null`.

### Astrofy

O projeto usa Astrofy quando tem `.sitespec/` com tarefas `<id>-<slug>/SITEPEC.md`, `.astrofy/` ou a dependência `@promovaweb/astrofy`.

| ASSUNTO | FICA COM | ARQUIVO |
|---|---|---|
| Tarefa técnica do site, dependências e comprovação | Astrofy | `.sitespec/<id>-<slug>/SITEPEC.md` |
| Conteúdo aprovado de uma página planejada | Astrofy | `.astrofy/pages/<slug>/page-spec.json` |
| Guia de interface do projeto | Designfy | `VIEWSPEC.md` |
| Especificação da tela e checklist das especialistas | Designfy | `viewspec/<id>-<slug>/VIEWSPEC.md` |
| Rotas, SEO, dados estruturados, imagens, build e deploy | skills `astrofy-*` | a sitespec |

| ASSUNTO | FICA COM | ARQUIVO |
| --- | --- | --- |
| Tokens gerados pelo Astrofy | `astrofy-design-system` | `.astrofy/design/design-system.json` |

O Designfy não grava sitespec e o Astrofy não grava viewspec. Cada um cita o outro: a sitespec aponta o viewspec pelo caminho relativo e o `trackedBy` do viewspec aponta a sitespec com `system: astrofy`. O CSS gerado pelo Astrofy é a fonte dos tokens. O `tokensFile` do guia aponta para ele. O `designfy theme check` mede o contraste desse CSS e a correção volta ao JSON.

### Specsfy

O projeto usa Specsfy quando tem `specs/` com as pastas de estado, `.specsfy/`, `DESIGNSYSTEM.MD` ou `INTERFACE.md`.

| ASSUNTO | FICA COM | ARQUIVO |
| --- | --- | --- |
| O que a fatia entrega, regras, permissões, tarefas e gates | Specsfy | `specs/<estado>/<NNNN>-<slug>/spec.md` |
| Interface da fatia, na seção Interface da spec | Specsfy | a spec |
| Orientações gerais de interface do produto | Specsfy | `DESIGNSYSTEM.MD` |
| Registro de componentes, blocos e telas | Specsfy | `INTERFACE.md` |
| Guia do Designfy, citando o `DESIGNSYSTEM.MD` | Designfy | `VIEWSPEC.md` |
| Execução da interface, checklist e capturas | Designfy | `viewspec/<id>-<slug>/VIEWSPEC.md` |

A spec manda. Quando o viewspec e a spec divergem, siga a spec. Registre nela o que faltar antes de acrescentar o conteúdo ao viewspec. Spec com `Interface para pessoas: Não` não ganha viewspec. Antes de copiar um item do registro, procure no `INTERFACE.md` o componente que já faz a função. Registre um componente novo no `INTERFACE.md` na mesma entrega. Os itens da checklist citam o ID da tarefa `[CODE]` da spec quando executam uma delas.

## Modelos e exemplos

A skill traz dois modelos em `assets/templates/`, `VIEWSPEC.guia.md` e `VIEWSPEC.tela.md`. Os cinco exemplos ficam em `assets/examples/`: `guia-de-site.md`, `guia-de-sistema.md`, `tela-landing-page.md`, `tela-dashboard.md` e `tela-artigo.md`. Os exemplos mostram o detalhe esperado em cada seção; use como forma, não como texto para copiar.

## A checklist

Cada item é uma tarefa GFM com a especialista e um identificador de quatro caracteres:

```
- [x] Hero com conteúdo real `#designfy-websites` {#a1b2}
- [x] Benefícios em três colunas `#designfy-components` {#c3d4}
- [ ] Cascata de entrada com versão parada `#designfy-motion` {#e5f6}
- [ ] Capturas em celular e desktop revisadas e achados resolvidos `#designfy-review` {#z9y8}
```

`[ ]` é pendente e `[x]` é concluído. A anotação `` `#designfy-<skill>` `` diz qual especialista executa o item; sem anotação, o item pertence à primeira skill de `skills`. O `{#id}` é criado uma vez e mantido. Item sem `{#id}` ou com `{#id}` repetido recusa o arquivo.

## O ciclo do status

| ESTADO | SIGNIFICADO |
|---|---|
| `draft` | viewspec escrito; a execução não começou |
| `in_progress` | execução em andamento |
| `blocked` | parado por algo externo; exige `blockedReason` |
| `done` | checklist completa e revisão com capturas registrada em `captures` |
| `archived` | encerrado sem conclusão; exige `archivedReason` |

Você altera o estado ao editar o arquivo. O leitor usa o estado explícito. Item pendente recusa `done`.

## Ler e validar

```
designfy viewspec guide
designfy viewspec list
designfy viewspec show 0001
designfy viewspec progress
designfy viewspec validate
```

O `guide` mostra o front matter, as seções presentes, o total de regras por seção e por origem e as propostas pendentes:

```
  Visão geral            1
  Cores                  3
  Tipografia             2
  ...
Rules by origin

  @pessoa   2
  @agente   18
  @revisão  0

Pending proposals

No pending proposals.
```

O `list` mostra uma linha por tela, com `id`, título, `kind`, `status` e progresso:

```
  0001  Landing page do produto  |  landing-page  |  in_progress  |  2/4
```

O `progress` resume por estado e devolve o próximo `id`:

```
  draft        0
  in_progress  1
  blocked      0
  done         0
  archived     0
  Checklist    2/4
  Next id      0002
```

O `show <id>` imprime o front matter e o corpo inteiro. O `validate` confere o contrato dos dois níveis e, com tudo certo, responde `Viewspec validation passed.` com exit code 0. Um arquivo inválido aparece com caminho e linha:

```
/tmp/opencode/designfy-demo/VIEWSPEC.md:80 [Aprendizados] Use YYYY-MM-DD, a valid
origin, a description and → followed by the learning destination.
```

Sem `VIEWSPEC.md` na raiz, o `validate` mostra `guide: missing` e continua com exit code 0; o `viewspec guide` para com exit code 6. A qualidade da interface fica com a `designfy-review`, com capturas.

## Conferir

- O `VIEWSPEC.md` da raiz tem as catorze seções e regras com `{#id}`.
- `designfy viewspec validate` termina com exit code 0.
- `designfy viewspec progress` devolve o próximo `id` livre.
- O viewspec da tela tem as nove seções e a checklist com dona e `{#id}`.
- O `status` da tela reflete o trabalho real, com `captures` quando fecha.
- Cada `trackedBy` aponta um arquivo que existe e passa no `viewspec validate`.
- O bloco **Integrations** do `status` mostra o Astrofy e o Specsfy do projeto.

# Capítulo 5: Tema e paleta

O `designfy theme check` responde com catorze pares medidos e a frase `Theme check passed.`:

```
Theme check passed.

  light foreground on background               16.89:1 pass
  light primary-foreground on primary          6.27:1 pass
  light muted-foreground on muted              5.92:1 pass
  light muted-foreground on background         6.46:1 pass
  light card-foreground on card                17.61:1 pass
  light accent-foreground on accent            10.46:1 pass
  light destructive-foreground on destructive  5.13:1 pass
  dark foreground on background                16.61:1 pass
  dark primary-foreground on primary           10.59:1 pass
  dark muted-foreground on muted               6.48:1 pass
  dark muted-foreground on background          7.78:1 pass
  dark card-foreground on card                 15.29:1 pass
  dark accent-foreground on accent              8.94:1 pass
  dark destructive-foreground on destructive    6.59:1 pass
```

Nenhum componente do registro carrega cor. Todos leem os papéis do tema definidos no mesmo arquivo CSS. Este capítulo mostra como ler esse arquivo, gerar a paleta e conferir o contraste antes de a tela existir.

## O arquivo de tokens

O Designfy usa o tema do Tailwind 4 no formato do shadcn. Um arquivo CSS com `@import "tailwindcss"`, um bloco `@theme inline` que liga as variáveis aos nomes do Tailwind e os blocos `:root` e `.dark` com os valores.

Para criar o arquivo completo no projeto:

```
designfy components add designfy-theme designfy-motion
```

O comando grava `designfy-theme.css` e `designfy-motion.css` em `paths.styles`. Ligue os dois no CSS de entrada, nesta ordem:

```
@import "./designfy-theme.css";
@import "tw-animate-css";
@import "./designfy-motion.css";
```

O `designfy-theme.css` já traz `@import "tailwindcss"`. Os papéis seguem os nomes do shadcn e cobrem superfície, texto, marca, estado, linha, foco, série de dados e shell:

| GRUPO | PAPÉIS |
|---|---|
| Superfície e texto | `background`, `foreground`, `card`, `popover`, `muted`, `muted-foreground` |
| Marca e ação | `primary`, `primary-foreground`, `secondary`, `secondary-foreground`, `accent`, `accent-foreground` |
| Estado | `destructive`, `success`, `warning`, `info`, cada um com o `-foreground` |
| Linha e foco | `border`, `input`, `ring` |
| Dados | `chart-1` a `chart-5` |
| Shell | `sidebar`, `sidebar-foreground`, `sidebar-primary`, `sidebar-accent`, `sidebar-border`, `sidebar-ring` |
| Forma, tipo e movimento | `radius`, `df-font-sans`, `df-font-heading`, `duration-*`, `ease-*` |

Nenhum componente usa `bg-indigo-500` nem `text-[#123]`. O código usa `bg-background`, `text-foreground`, `bg-primary`, `text-muted-foreground`, `border-border`, `font-heading`, `font-sans` e `rounded-lg`. A utilidade lê o papel. Assim, a troca de tema muda a tela inteira num lugar só.

## Ler o tema do projeto

```
designfy theme show
```

O comando segue uma ordem para escolher o arquivo:

1. `--file <path>`, quando você indica.
2. `paths.theme` do `designfy.json`, gravado por `init --theme-path`.
3. O `tokensFile` do `VIEWSPEC.md` da raiz, quando aponta para um arquivo que existe.
4. A entrada convencional em `paths.styles`, `src/styles` e `src/app`, com `global.css`, `app.css` ou `main.css` que importem o Tailwind.
5. A busca por CSS do Tailwind nesses mesmos diretórios, com `src/index.css` e `app/globals.css` na lista.

A busca segue os imports locais e extrai `@theme`, `@theme inline`, `:root` e `.dark`. Áreas como `docs/`, `docs-theme.css`, `admin/` e `email/` ficam no fim da fila. A resposta começa pelo arquivo lido e agrupa os tokens:

```
/tmp/opencode/designfy-demo/src/styles/app.css

colors
  --color-background          light var(--background) → #f7fbfb | dark var(--
background) → #061114
  --color-foreground          light var(--foreground) → #0d1b1e | dark var(--
foreground) → #e7f1f0
  --color-primary             light var(--primary) → #006966 | dark var(--primary) →
#4cd7c8
  ...

radius
  --radius-sm   light calc(var(--radius) - 4px) | dark calc(var(--radius) - 4px)
  --radius-lg   light var(--radius) | dark var(--radius)

motion
  --duration-base  light 250ms | dark 250ms
  --ease-out-expo  light cubic-bezier(0.16, 1, 0.3, 1) | dark cubic-bezier(0.16, 1,
0.3, 1)
```

Cada cor aparece em dois formatos quando o valor é uma variável: a declaração ( `var(--background)` ) e o valor resolvido em hex para claro e escuro. Use `--group` para ver um assunto só e `--file` para escolher outro CSS:

```
designfy theme show --group colors
designfy theme show --group motion
designfy theme show --file src/styles/app.css
```

Os grupos são `colors` , `fonts` , `text` , `radius` , `spacing` , `shadows` , `breakpoints` e `motion` . Sem CSS do Tailwind no projeto, o comando para com exit code 6 e a dica:

```
error No Tailwind stylesheet found. Run "designfy components add designfy-theme". [6
NOT_FOUND]
```

O `--json` informa qual regra escolheu o arquivo: `selection` vale `--file` , `paths.theme` , `tokensFile` , `conventional entry` ou `Tailwind search` . O campo `entryFile` aponta o CSS de entrada que importa o arquivo escolhido, quando ele existe.

### Tailwind 3

Num projeto que ainda está no Tailwind 3, a busca aceita `@tailwind base` ou `@import "tailwindcss/base"` e completa os tokens com `tailwind.config.*` na raiz. A resposta informa `tailwind: 3`, o `configFile` e a lista de arquivos lidos. Se a configuração não carregar, um aviso aparece e o resultado traz só os tokens do CSS. O comando executa o módulo de configuração, então rode apenas em projeto cujo código você confia.

## Conferir contraste

```
designfy theme check
```

O comando mede os pares conhecidos nos dois temas: `foreground` em `background`, `primary-foreground` em `primary`, `muted-foreground` em `muted` e em `background`, `card-foreground` em `card`, `accent-foreground` em `accent` e `destructive-foreground` em `destructive`. Também confere se os papéis que o registro usa existem no arquivo.

Um par abaixo de 4.5 reprova o comando com exit code 10:

```
error Theme check failed. light primary-foreground on primary (2.26:1). [10
CHECK_FAILED]
```

Pares opcionais com token ausente são pulados e aparecem no resultado. O `--file` escolhe o CSS. Quando o tema está dividido em vários arquivos, aponte para o CSS de entrada que importa todos, porque o comando segue os imports locais:

```
designfy theme check --file src/styles/app.css
```

Sem o `--file`, o comando usa a mesma ordem de escolha do `theme show`: `paths.theme`, o `tokensFile` do guia e a busca no projeto. Se ele acusar um token de movimento ausente, falta o `designfy-motion.css` no projeto ou no CSS de entrada.

### Modo custom

Um projeto com escalas próprias não precisa dos papéis do shadcn para medir contraste. Nesse caso o comando entra no modo `custom` e mede os pares que você declarar em `theme.pairs`, no `designfy.json`:

```
{
  "theme": {
    "pairs": [
      { "fg": "--text-body", "bg": "--bg-page", "label": "Texto corrido" },
      { "fg": "primary-950", "bg": "white", "label": "Destaque", "min": 7 }
    ]
  }
}
```

`fg` e `bg` aceitam variável CSS com o prefixo `--` e nome de cor do tema do Tailwind, como `primary-950` e `white`. `label` dá nome ao par e `min` aceita razão de 1 a 21, com 4,5 como padrão. O par se repete no tema escuro quando algum dos tokens tem declaração escura ou muda de valor. Sem pares configurados, o comando tenta `--text-body`, `--text-heading` e `--accent-text` sobre `--bg-page`, além de `--foreground` sobre `--background`, quando existirem.

```
Theme check passed.

Mode: custom

project uses its own color scales; shadcn roles not checked

  light --text-body on --bg-page (Texto corrido)  7.73:1 pass (min 4.5:1)
  light primary-950 on white (Destaque)           18.53:1 pass (min 7:1)
  dark --text-body on --bg-page (Texto corrido)   16.86:1 pass (min 4.5:1)
```

O modo `shadcn` continua para o tema com os papéis do registro. Nesse modo, o comando também confere se cada papel obrigatório existe.

## A roda de cores

```
designfy palette wheel "#2AD5BE"
```

A resposta traz a base e uma linha por harmonia, com hex e OKLCH:

```
  Base                #2ad5be oklch(0.78624324 0.13467707 180.88504461)
  complementary       #ff93b4 oklch(0.78624324 0.13426665 0.88504461)
  analogous           #73d28c oklch(0.78624324 0.13467707 150.88504461), #11d0ea
oklch(0.78624324 0.13467707 210.88504461)
  triadic             #c7a5ff oklch(0.78624324 0.12900255 300.88504461), #f7a458
oklch(0.78624324 0.13467707 60.88504461)
  split-complementary  #eb98e1 oklch(0.78624324 0.13467707 330.88504461), #ff9a88
oklch(0.78624324 0.12475465 30.88504461)
  tetradic            #6cc3ff oklch(0.78624324 0.12019084 240.88504461), #ff93b4
oklch(0.78624324 0.13426665 0.88504461), #f7a458 oklch(0.78624324 0.13467707
60.88504461)
  monochromatic       #00453c oklch(0.35 0.06313226 180.88504461), #a3cfc6
oklch(0.82 0.04713697 180.88504461)
```

As harmonias são `complementary`, `analogous`, `triadic`, `split-complementary`, `tetradic` e `monochromatic`. Escolha pelo tipo de produto: complementar para contraste entre ação e destaque, análoga para um conjunto calmo, tríade para três papéis com a mesma força, monocromática para uma interface discreta.

## Gerar a paleta

```
designfy palette generate --base "#2AD5BE" --harmony triadic --format css
```

O comando monta a escala `50` a `950` em OKLCH para `primary`, uma escala `neutral` com o matiz da base e croma baixo e, com `--harmony`, as escalas de acento. Também gera os tokens semânticos claros e escuros com o contraste conferido:

```
:root {
  --background: oklch(0.985 0.00216 180.88504461);
  --foreground: oklch(0.2 0.0045 180.88504461);
  --card: oklch(0.955 0.00468 180.88504461);
  --muted: oklch(0.9 0.00774 180.88504461);
  --border: oklch(0.475 0.0144 180.88504461);
  --primary: oklch(0.475 0.08567937 180.88504461);
  --primary-foreground: oklch(0.985 0.00216 180.88504461);
  --accent: oklch(0.9 0.0554711 300.88504461);
  --ring: oklch(0.475 0.10320204 300.88504461);
}

.dark {
  --background: oklch(0.2 0.0045 180.88504461);
  --foreground: oklch(0.985 0.00216 180.88504461);
  --primary: oklch(0.655 0.11814727 180.88504461);
}
```

`--format` aceita `css` (padrão), `json` e `tailwind`. O `css` imprime variáveis no formato do shadcn com `@theme inline`; o `tailwind` imprime só o bloco `@theme`; o `json` devolve o objeto para um script. `--out <file>` grava no arquivo. Substitua os blocos `:root` e `.dark` do arquivo de tokens pelos papéis gerados e mantenha os tokens de tipo, forma, layout e movimento.

Cores entram como hex ( `#RGB`, `#RRGGBB` ) ou `oklch(L C H)`. Use aspas no shell.

## Medir um par

```
designfy palette contrast "#00161E" "#C6F4ED"
```

```
  Foreground    #00161e
  Background    #c6f4ed
  WCAG ratio    15.50:1
  APCA          Lc 93.0
  Normal text   AA pass, AAA pass
  Large text    AA pass, AAA pass
  UI component  AA pass, AAA N/A
```

O veredito separa texto normal (4.5:1), texto grande (3:1) e elemento de interface (3:1). Se você usa a métrica mais recente, consulte o Lc com sinal do APCA. Por padrão, o comando sai com 0. A opção `--min <ratio>` faz sair com 10 abaixo do valor escolhido, o que serve para uma conferência em CI:

```
designfy palette contrast "#00161E" "#C6F4ED" --min 7
```

## Claro e escuro

Cada papel tem uma definição para o modo claro e outra para o escuro. O escuro não é uma inversão automática: o `primary` muda de luminosidade, o `muted` ganha outra superfície e o `ring` acompanha o acento. Por isso um papel sem valor no `.dark` aparece como ausente na leitura do tema.

O arquivo de tema declara `@custom-variant dark (&:where(.dark, .dark *))` e cada papel muda sozinho quando a classe `dark` entra no `html`. Componente não usa `dark:` para cor. O `dark:` fica só para diferença que não é de cor, como a troca de uma imagem.

Neutros levam um pouco do matiz da base, com croma entre 0.004 e 0.02. Cinza puro, preto absoluto e branco absoluto ficam fora das superfícies.

## Conferir

- O CSS de entrada importa o tema e o movimento. O build resolve os dois.
- `designfy theme show --group colors` mostra claro e escuro para cada papel.
- `designfy theme check` termina com exit code 0 nos dois temas.
- Num projeto com escalas próprias, o modo `custom` mede os pares do `theme.pairs`.
- `designfy palette wheel` lista as seis harmonias para a cor da marca.
- `designfy palette contrast` aprova os pares que você usa em texto e botão.

# Capítulo 6: Catálogo e componentes

O agente precisa de um cartão de indicador para o topo do dashboard. Em vez de inventar um, ele roda `designfy catalog show stat-card` e recebe a anatomia, os estados, o movimento, os papéis de cor e a origem da implementação:

```
Stat card
  ID              stat-card
  Surface         webapp
  Category        data-display
  Purpose         Shows one indicator with its change and context.
  Use when        Dashboards and record summaries.
  Avoid when
  Anatomy         label; value; change against the previous period; sparkline or
icon (optional); link to detail
  Variants        plain; with sparkline; with progress toward a goal
  States          default; loading; error; no data
  Responsive      Four in a row, then two, then one or a horizontal snap row.
  Motion          The value counts from the previous number; the change badge fades;
the sparkline draws once.
  Accessibility   Change has a sign and text, not only color or arrow.; Label and
value are associated.
  Tokens          card; card-foreground; border; radius; muted-foreground; success;
destructive
  Viewspec kinds  dashboard; detail
  Related         counter; chart-card
  Sources         designfy:stat-card
```

A última linha diz de onde vem a peça. `designfy:stat-card` copia com o próprio binário; `shadcn:<item>` e `reui:<item>` instalam pelo CLI do shadcn; `custom` avisa que aquela função ainda não tem implementação pronta. O catálogo decide a forma; o registro entrega o arquivo.

## O catálogo de interface

O catálogo tem 91 fragmentos comuns de site e de sistema, em 15 categorias e três superfícies: 43 entradas de `site`, 22 de `webapp` e 26 de `both`. O valor antigo `system` continua aceito e é lido como `webapp`. Cada entrada declara propósito, quando usar, quando evitar, anatomia, variações, estados, responsividade, movimento, acessibilidade, papéis de tema, itens relacionados e as fontes de implementação.

```
designfy catalog list
designfy catalog list --surface webapp
designfy catalog list --kind dashboard
designfy catalog search table
```

O `--kind` usa o mesmo tipo de tela do viewspec. Para um dashboard, a lista começa assim:

```
  button             Button (both, actions)
  link               Text link (both, actions)
  card               Card (both, containers)
  badge              Badge (both, data-display)
  avatar             Avatar (both, data-display)
  tabs               Tabs (both, navigation)
  dropdown-menu      Dropdown menu (both, actions)
  tooltip            Tooltip (both, feedback)
  dialog             Dialog (both, feedback)
  drawer             Drawer (both, feedback)
  counter            Animated counter (both, motion)
  theme-toggle       Theme toggle (both, actions)
  app-shell          App shell (webapp, shell)
  ...
```

A busca não diferencia caixa nem acento:

```
  dropdown-menu      Dropdown menu (both, actions)
  pricing-table      Pricing table (site, site-sections)
  comparison-table   Comparison table (site, site-sections)
  prose              Prose (site, content)
  table-of-contents  Table of contents (site, content)
  figure             Figure (site, content)
  chart-card         Chart card (webapp, data-display)
  data-table         Data table (webapp, data-display)
  filter-bar         Filter bar (webapp, data-display)
  bulk-actions       Bulk action bar (webapp, data-display)
  settings-layout    Settings layout (webapp, settings)
```

Id desconhecido sai com exit code 6. A skill `designfy-components` exige que `useWhen` valha e `avoidWhen` não valha antes de escolher a peça.

## O registro de componentes

O registro tem dois manifestos no formato do shadcn, `registry/react/registry.json` e `registry/astro/registry.json`. O registro inclui também os arquivos de origem. O binário leva o registro embutido, gerado por `bun run registry:embed`, então `components list` funciona sem o repositório.

```
designfy components list
designfy components list --framework react
designfy components list --surface webapp
```

O registro React tem 69 itens e o Astro tem 57. Sem `--framework`, o comando usa o framework do `designfy.json`; `next` e `react-vite` leem o registro React. A lista desta versão:

| CATEGORIA | ITENS |
|---|---|
| `layout` | `background`, `background-styles`, `pixel-stage`, `section` |
| `utilities` | `cn`, `designfy-motion-runtime` |
| `styles` | `designfy-motion`, `designfy-theme`, `designfy-prose` |
| `hero` | `hero-split`, `hero-centered`, `hero-media`, `hero-pixel` |
| `motion` | `reveal`, `counter` |
| `site-navigation` | `announcement-bar`, `site-footer`, `site-header` |
| `site-sections` | `audience-fit`, `bento-grid`, `comparison-table`, `cta-band`, `detail-card`, `faq`, `feature-grid`, `feature-rows`, `feature-split`, `logo-marquee`, `marquee`, `newsletter-form`, `numbered-checklist`, `offer-card`, `outcome-grid`, `milestone-timeline`, `pricing-table`, `profile-split`, `statement-stats`, `stats-band`, `steps`, `testimonial` |
| `containers` | `card` |
| `actions` | `theme-toggle` |
| `content` | `article-header`, `author-box`, `callout`, `code-block`, `figure`, `post-card`, `post-list`, `prose`, `reading-progress`, `table-of-contents`, `video-embed` |

| CATEGORIA | ITENS |
| --- | --- |
| `data-display` | `activity-feed`, `description-list` no Astro; mais `chart-card`, `data-table`, `filter-bar`, `stat-card` no React |
| `shell` | `app-shell`, `page-header`, só no React |
| `auth` | `auth-card`, só no React |
| `feedback` | `empty-state`, `error-state` |
| `forms` | `file-upload`, `form-layout`, `stepper`, só no React |
| `settings` | `settings-layout`, `settings-section`, só no React |

A base do registro está nas três primeiras categorias da tabela: `designfy-theme` traz os tokens, `designfy-motion` e `designfy-motion-runtime` trazem o movimento e o script, `cn` combina classes, `designfy-prose` liga a tipografia de leitura, `background` e `background-styles` trazem os padrões de fundo e `section` fecha o ritmo das seções. Copie esses arquivos de estilo e de biblioteca uma vez por projeto.

### Os onze itens do estudo Founders

Onze itens vêm da landing de oferta lida no estudo do capítulo 3, em React e em Astro, com o mesmo nome, as mesmas props e a mesma estrutura:

| ITEM | O QUE ENTREGA |
| --- | --- |
| `pixel-stage` | mosaico de pixels em canvas, que lê o tema, acende com o ponteiro, pausa por teclado e tem versão parada |
| `hero-pixel` | abertura de landing sobre o palco de pixels, com resumo lateral opcional |
| `feature-rows` | vantagens numeradas em linhas alinhadas, com introdução acima ou lateral fixa |
| `milestone-timeline` | eventos datados ou etapas em ordem, com marcador do momento atual e resultados alinhados |
| `profile-split` | apresenta uma pessoa com retrato, credenciais e links |
| `offer-card` | oferta com prazo, vagas opcionais e ação principal, nas variantes `tinted` e `stage` |
| `audience-fit` | duas listas, com perfis indicados e contraindicados, com ou sem nota |

| ITEM | O QUE ENTREGA |
| --- | --- |
| `numbered-checklist` | inclusões com chips numerados e destaques opcionais, em uma ou duas colunas |
| `outcome-grid` | resultados curtos numa grade responsiva de blocos com ícone |
| `detail-card` | explica um recurso com ícone, parágrafos e itens de conferência; inclui o `DetailCardGrid` |
| `statement-stats` | frase em dois tons ao lado de estatísticas empilhadas, para compor com a linha do tempo |

Cinco itens que já existiam ganharam variante:

| ITEM | VARIANTE NOVA |
| --- | --- |
| `faq` | `split` com `stickyIntro`, o título fixo à esquerda enquanto as perguntas rolam |
| `cta-band` | `secondaryAction`, a segunda ação ao lado da principal |
| `testimonial` | `carousel`, um depoimento por vez no celular e vários no desktop |
| `pricing-table` | `single`, um plano com oferta e lista de inclusões no lugar da grade |
| `marquee` | `accent`, com separadores em losango e a cor de destaque do tema |

## Conferir um item

```
designfy components show hero-split
```

```
Split hero
Pairs a headline and actions with a framed visual in a responsive, animated layout.

  Category               hero
  Surface                site
  Motion                 Yes
  Files                  hero-split/HeroSplit.astro
  Dependencies           None
  Dev dependencies       None
  Registry dependencies  designfy-theme, designfy-motion
```

`--source` inclui o código do arquivo na resposta. Nome desconhecido sai com exit code 6.

## Copiar para o projeto

```
designfy components add hero-split
```

O comando resolve os itens exigidos do Designfy em ordem, grava cada arquivo no destino e relata o que fez:

```
  Written                /tmp/opencode/designfy-demo/src/components/designfy/hero-
split/HeroSplit.astro
  Unchanged              /tmp/opencode/designfy-demo/src/styles/designfy-theme.css
                         /tmp/opencode/designfy-demo/src/components/designfy/lib/
designfy-motion.ts
                         /tmp/opencode/designfy-demo/src/styles/designfy-motion.css
  Skipped                None
  Missing npm packages   None
  Missing dev packages   None
  Install shadcn items   None
```

As categorias da resposta separam o resultado:

| CATEGORIA | SIGNIFICADO |
|---|---|
| `Written` | arquivo novo gravado |
| `Unchanged` | o arquivo já estava igual |
| `Skipped` | o arquivo existe com conteúdo diferente e mantém a versão local |

| CATEGORIA | SIGNIFICADO |
| --- | --- |
| `Missing npm packages` | pacotes de produção que o arquivo importa e o projeto não tem |
| `Missing dev packages` | pacotes de desenvolvimento que faltam |
| `Install shadcn items` | itens `shadcn:` que precisam do CLI do shadcn |

O destino segue o tipo de arquivo. Componente vai para `paths.components`; estilo, para `paths.styles`. O marcador `@designfy/components` é trocado pelo alias de `designfy.json`, então o import funciona com o alias do projeto. Um arquivo existente e diferente não é sobrescrito sem `--overwrite`. Depois de copiado, o arquivo pertence ao projeto e pode ser editado.

O `--dry-run` mostra o plano sem gravar, inclusive as dependências pendentes. `--path <dir>` troca a pasta de destino e `--framework <framework>` escolhe o manifesto. Sem `designfy.json`, o comando usa os padrões e avisa.

## shadcn e ReUI

O Designfy não substitui o CLI do shadcn. Os primitivos acessíveis continuam saindo do shadcn:

```
npx shadcn@latest add button card dialog
```

O ReUI entra pelo registro `@reui` do mesmo CLI:

```
npx shadcn@latest add @reui/data-grid
```

Os três usam o mesmo tema: papéis do shadcn sobre o Tailwind 4. O registro do Designfy traz composições de página, conteúdo e sistema, com movimento e com os estados descritos no catálogo. O shadcn traz as primitivas sem estilo. O ReUI traz blocos avançados, como grade de dados, filtros e kanban.

## A regra de tema único

Todo item do registro obedece à mesma regra: cor, fonte, raio e movimento vêm dos papéis do tema. O item não leva cor literal, classe de paleta fixa (`bg-indigo-500`, `text-gray-900`), cor arbitrária (`bg-[#0af]`) nem `font-family` literal. Transparência sai do papel, como `bg-primary/10` ou `color-mix(in oklab, var(--primary) 18%, transparent)`.

`dark:` não serve para cor, porque os papéis já mudam com o tema. `dark:` fica para diferença que não é de cor, como trocar uma imagem. O `bun run registry:check` recusa o item fora da regra.

O registro também exige conteúdo por propriedade. Nenhum texto com cara de real dentro do componente; texto, link, imagem e número chegam por prop ou slot. Região opcional sem conteúdo não renderiza nada. A peça é semântica, tem alvo de toque de 44 px no celular, mantém o foco visível e usa uma biblioteca de ícones só.

## Conferir

- `designfy catalog list --kind <tipo>` devolve os fragmentos do tipo de tela.
- `designfy catalog show <id>` mostra a origem da implementação.
- `designfy components list --framework react` mostra os 69 itens.
- `designfy components list --framework astro` mostra os 57 itens.
- `designfy components add <item> --dry-run` mostra os arquivos antes da cópia.
- `designfy deps check` termina com exit code 0 depois da cópia.
- Nenhum arquivo copiado tem cor literal ou `font-family`.

# Capítulo 7: Fundos e movimento

A landing page abre com o hero sobre um padrão de grade discreto, com um desvanecimento radial que segura o texto legível. As seções seguem na superfície, uma faixa tingida marca a virada de assunto e a chamada final repete o padrão do hero. Enquanto você rola, os blocos entram em cascata; no fim, tudo continua legível com o movimento reduzido ligado no sistema. Esse é o padrão que o Designfy entrega para fundo e movimento.

## Três níveis de fundo

O fundo dá ritmo à página e separa seções sem precisar de caixa.

| NÍVEL | O QUE É | CLASSE OU COMPONENTE |
|---|---|---|
| Superfície | a cor da página | `bg-background` |
| Faixa | seção inteira tingida | `bg-muted/60`, `bg-card`, ou `bg-primary text-primary-foreground` numa chamada |
| Padrão | desenho discreto atrás do conteúdo | componente `Background` com um preset |

Numa página de site, a maior parte das seções fica na superfície. Uma faixa tingida a cada duas ou três seções marca mudança de assunto. O padrão aparece no hero, na chamada final e, no máximo, em mais uma seção. Em sistema, o fundo é superfície e cartão; padrão só em tela de entrada, estado vazio e página de erro.

## O componente Background

```
designfy components add background
```

O comando copia o componente e o `designfy-backgrounds.css`. Importe o CSS uma vez no CSS de entrada.

```
<Background preset="signal-grid" fade="radial">
  <HeroCentered … />
</Background>
```

```
<Background preset="studio-soft-blueprint" fade="bottom">
  <slot />
</Background>
```

As propriedades são poucas:

| PROP | VALORES | EFEITO |
|---|---|---|
| `preset` | o slug de um padrão | escolhe o desenho |
| `fade` | `none`, `bottom`, `edges`, `radial` | aplica máscara na camada do padrão |
| `className` ou `class` | classes do projeto | ajuste local do container |

O componente renderiza a camada do padrão com `aria-hidden="true"` e mantém o conteúdo numa camada acima. Todos os presets leem os papéis do tema e mudam com a paleta e com o modo escuro. Nenhum preset tem cor literal.

## Escolher um preset

O catálogo desta versão tem 334 presets, com 257 desenhos distintos, em quatro categorias:

| CATEGORIA | PRESETS | O QUE TRAZ | COMBINA COM |
|---|---|---|---|
| `geometric` | 125 | grades, pontos, linhas, hachuras | produto técnico, ferramenta, documentação, sistema |
| `gradients` | 64 | lavagens e brilhos radiais | página de venda, marca suave, chamada final |
| `decorative` | 66 | texturas e composições | marca com personalidade, página editorial |
| `effects` | 79 | holofotes, auroras, brilhos animados | hero de lançamento; um por página |

Cada preset declara a legibilidade em `presets.json`:

| VALOR | PRESETS | COMO USAR |
|---|---|---|
| `text-safe` | 160 | pode ficar atrás de título e texto curto |
| `needs-fade` | 114 | use com `fade="radial"` ou `fade="bottom"` e meça o contraste |
| `decorative-only` | 60 | só em área sem texto: lateral do hero, painel de marca, rodapé |

A classificação é uma estimativa do gerador. A conferência final é a medida de contraste no ponto mais difícil atrás do texto e a captura nos dois temas.

Nove presets se movem, com brilho que pulsa, feixe que varre ou grade com foco. Use um por página, em seção curta. Todos param com `prefers-reduced-motion` e quando saem da tela. Padrão geométrico sem função vira enfeite: ele funciona quando sugere a matéria do produto, como grade para ferramenta de layout e pontos para cartografia e estatísticas.

Os presets vêm de duas fontes. O ReactBD BG distribui padrões do Pattern Craft de Megh Bari sob a licença MIT. O GradientDeck convida a copiar os padrões e não publica licença para eles. O Designfy converteu as cores em papéis do tema e manteve a geometria.

## Movimento é o padrão

Toda tela tem algum movimento com função: entrada, resposta ao toque, troca de estado ou rolagem. Tela parada só quando o guia do projeto disser. Todo movimento tem versão parada para `prefers-reduced-motion: reduce`.

```
designfy components add designfy-motion
```

O arquivo `designfy-motion.css` declara as animações, as utilidades e o runtime `initDesignfyMotion`, que liga a classe `js` no `html`, marca as revelações e pausa laços fora da tela. Use CSS para a maior parte dos efeitos e recorra ao Motion quando a animação exigir mais recursos.

### Utilidades de CSS

| UTILIDADE | O QUE FAZ |
|---|---|
| `data-df-reveal` | revela o elemento uma vez ao aparecer; valores `up`, `down`, `left`, `right`, `scale`, `fade` e `mask` |
| `data-df-stagger` no pai | numera os filhos e escalona a revelação |
| `style="--df-i: 3"` | posição na sequência, quando não há pai com `data-df-stagger` |
| `data-df-count="1250"` | anima o número até o valor final ao aparecer, com `data-df-count-from` e `data-df-count-duration` |
| `data-df-loop` | marca um laço para pausar fora da tela |
| `animate-df-*` | animações prontas: `fade-in`, `fade-up`, `fade-down`, `slide-in-left`, `slide-in-right`, `scale-in`, `mask-up`, `pop`, `shake`, `draw`, `check`, `marquee`, `shimmer`, `pulse-soft`, `float`, `pan`, `spin-slow`, `border-beam`, `collapse-down` e `collapse-up` |

| UTILIDADE | O QUE FAZ |
| --- | --- |
| `df-lift` | elevação de 2 px e borda mais forte no hover, com efeito de botão apertado no clique |
| `df-press` | escala 0.98 enquanto o botão estiver apertado, com transição de cor |
| `df-underline` | sublinhado que cresce da esquerda no hover e no foco |
| `df-collapse` com `data-df-open` | painel que abre por `grid-template-rows` |
| `df-marquee` | faixa rolante, com `--df-marquee-duration`, `--df-marquee-gap` e `data-df-direction="right"` |
| `df-shimmer` | brilho de esqueleto com os papéis do tema |
| `df-keep-motion` | mantém a animação com movimento reduzido, para indicador de progresso |

A cascata usa `data-df-stagger` no pai. Os filhos recebem o índice em `--df-i` e o atraso sai de `--stagger`. A revelação só esconde o conteúdo quando o script adiciona `js` ao `html`; sem JavaScript e com animação por rolagem do navegador, o conteúdo aparece.

As durações e curvas vêm do tema: `--duration-instant` (100 ms), `--duration-fast` (150 ms), `--duration-base` (250 ms), `--duration-slow` (500 ms) e `--duration-slower` (800 ms), com `--ease-out-expo`, `--ease-out-quart` e `--ease-spring` entre as curvas.

### O que se move

- Só `transform`, `opacity`, `clip-path`, `mask` e `background-position` animam. Exclua da animação `transition-all`, `width`, `height`, `top`, `left`, `margin` ou `padding`.
- Um sinal de hover por elemento, aplicado ao container para abranger também a área ao redor da imagem.
- Laço pausa no hover, no foco e fora da tela. Com movimento reduzido, o laço para.
- Fundo animado fica em seção curta. Atrás de texto longo, prefira o estático e ponha o movimento no conteúdo.
- O componente React lê `useReducedMotion()` e renderiza o estado final.

## Movimento com o Motion

Quando o CSS não resolve, a skill `designfy-motion` usa `motion/react` para animação de layout, entrada e saída de elemento, gesto e número com mola. Use `motion/react` para a importação. Mantenha `framer-motion` fora das importações e mantenha a versão parada.

```
import { motion, useReducedMotion } from "motion/react";

export function Stat({ value }: { value: number }) {
  const reduced = useReducedMotion();
  return (
    <motion.span
      initial={reduced ? false : { opacity: 0, y: 8 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.25 }}
    >
      {value}
    </motion.span>
  );
}
```

O CSS continua sendo a primeira escolha: custa menos, funciona sem JavaScript e não depende de hidratação.

## Conferir

- O CSS de entrada importa `designfy-backgrounds.css` e `designfy-motion.css`.
- `designfy theme check` aprova o contraste do texto sobre o padrão escolhido.
- Cada seção declara um nível de fundo: superfície, faixa ou padrão.
- Todo preset usado atrás de texto é `text-safe` ou leva o `fade` indicado.
- Todo movimento tem versão parada em `prefers-reduced-motion`.
- Laços pausam no hover, no foco e fora da tela.
- Nenhum arquivo usa `transition-all` nem anima altura ou largura.

# Capítulo 8: Capturas e revisão

A skill `designfy-review` abre o `manifest.json` da última execução e encontra uma linha com `overflowX: true` na captura de 390 px. A tabela de pedidos estoura a largura do celular. Ela abre o PNG, confirma o corte, devolve o achado para a `designfy-responsive` e pede uma captura nova depois da correção. Sem a captura, o problema teria passado.

## Instalar o Playwright

A captura roda o Playwright num processo Node, a partir do projeto. O binário não leva navegador. Instale o pacote pelo catálogo e baixe o Chromium:

```
designfy deps install testing
npx playwright install chromium
```

Sem o pacote, o comando para com exit code 10 e a dica:

```
error Playwright is unavailable. Run designfy deps install testing. [10
CHECK_FAILED]
```

O pacote é de desenvolvimento e traz também o `@axe-core/playwright`, usado nas auditorias de acessibilidade. Rode a instalação do navegador uma vez por máquina ou no passo de instalação da CI.

## Capturar

```
designfy capture run http://localhost:4321/produto/
```

O comando abre cada URL nos viewports escolhidos e grava um PNG por combinação. O padrão vem de `capture.viewports`, cuja configuração inicial é `mobile,desktop`.

```
designfy capture run http://localhost:4321/ http://localhost:4321/precos/ --
viewport mobile,tablet,desktop --theme dark
designfy capture run http://localhost:4321/ --viewport mobile,desktop --theme
light,dark
```

Os presets são:

| NOME | DIMENSÃO | TOQUE E ESCALA |
|---|---|---|
| `mobile` | 390 × 844 | toque, escala 3 |

| NOME | DIMENSÃO | TOQUE E ESCALA |
|---|---|---|
| `tablet` | 820 × 1180 | toque, escala 2 |
| `desktop` | 1440 × 900 | sem toque, escala 1 |
| `wide` | 1920 × 1080 | sem toque, escala 1 |

`LARGURAxALTURA` cria outra dimensão, como `1024x768`. `--viewport` se repete ou aceita lista por vírgula.

As options cobrem o que a revisão precisa:

| OPTION | EFEITO |
|---|---|
| `--theme light\|dark` | define o `colorScheme` do navegador; se repete ou aceita lista, como `--theme light,dark` |
| `--full-page` | captura a página inteira, o padrão |
| `--fold` | captura só a primeira dobra, sem preparar a página |
| `--no-scroll` | pula a rolagem e a espera por recursos antes da captura da página inteira |
| `--scroll-budget <ms>` | limita a rolagem e a espera por recursos, 15000 ms por padrão |
| `--hide <css>` | oculta os elementos do seletor; se repete, como `--hide '#chat' --hide '.floating-button'` |
| `--show-dev-overlays` | mantém visíveis os overlays de desenvolvimento |
| `--reduced-motion` | emula a preferência por movimento reduzido |
| `--wait <ms>` | espera depois do carregamento |
| `--selector <css>` | captura só o elemento que casa com o seletor |
| `--out <dir>` | troca a pasta raiz das capturas |
| `--name <run>` | nomeia a pasta da execução |

Com dois ou mais temas, cada combinação de URL, viewport e tema produz um PNG cujo nome termina em `--<viewport>--<theme>.png`. Com um tema, termina em `--<viewport>.png`.

### A página inteira

Antes do print da página inteira, o navegador percorre a página em passos da altura do viewport, aguarda a rede ficar ociosa e as imagens e fontes terminarem de carregar e volta ao topo. Isso dispara as revelações por rolagem e as imagens preguiçosas: sem a preparação, o PNG mostra só o que já estava na primeira dobra.

- Iframes de outra origem abaixo da dobra são fotografados durante a preparação, para preservar o conteúdo visível no PNG.
- Os overlays conhecidos de Astro, Vite e Next.js ficam ocultos por padrão. `--show-dev-overlays` mantém todos. A opção `--hide` esconde qualquer outro seletor.
- `--scroll-budget` limita o tempo da preparação. Quando o limite chega, a captura segue e o manifesto marca `scrollBudgetExhausted`.
- `--no-scroll` desliga a preparação. É útil em página curta, com animação infinita ou servidor lento.
- `--fold` e `--selector` também pulam a preparação, porque capturam só uma área.
- Acima de 64 Mi pixels físicos, a captura de página inteira usa a escala CSS para reduzir o uso de memória do Chromium.

Para revisar um componente isolado:

```
designfy capture run http://localhost:4321/componentes/hero-split --selector
"#preview" --fold
```

O `--fold` é útil para o hero; o `--full-page` mostra a página inteira e revela o que só aparece no fim.

## Conjunto mínimo de uma revisão

Quatro execuções cobrem a revisão: a página inteira em `mobile,tablet,desktop,wide`; a dobra em `320x640,mobile,1280x800,desktop` com `--fold`; o tema escuro em `mobile,desktop` com `--theme dark`; e o movimento reduzido em `mobile,desktop` com `--reduced-motion`. Em projeto com um tema só, a terceira não existe; com dois temas, uma execução com `--theme light,dark` cobre os dois de uma vez. Carregando, vazio e erro também são tela: capture com um parâmetro de desenvolvimento ou um login de teste, como descreve a referência de capturas.

## O manifesto

Cada execução grava uma pasta em `.designfy/captures/<run>/`, com os PNGs e um `manifest.json`. O manifesto registra:

| CAMPO | CONTEÚDO |
| --- | --- |
| `name`, `createdAt`, `outDir` | nome, horário e pasta da execução |
| `urls` | as URLs visitadas |
| `viewports` | nome, largura, altura, escala, toque e modo móvel de cada viewport |
| `themes` | os temas em ordem; `theme` guarda o primeiro |
| `fullPage`, `scroll`, `scrollBudgetMs`, `showDevOverlays`, `hide`, `reducedMotion`, `waitMs`, `selector` | as options usadas |
| `shots` | uma entrada por captura |
| `errors` | falhas de captura, com URL e viewport |

Cada entrada de `shots` traz `url`, `viewport`, `theme`, `file` e o que a revisão lê: `title`, `documentWidth`, `documentHeight`, `viewportWidth`, `overflowX`, `consoleErrors`, `pageErrors`, `scrollSteps`, `incompleteImages`, `scrollBudgetExhausted` e `error` quando a captura falha. `scrollSteps` informa o total de passos da preparação; `incompleteImages` lista as imagens que ainda não tinham carregado; `scrollBudgetExhausted` avisa que o tempo de preparação acabou antes do fim da página. O `overflowX` é o primeiro sinal que a revisão lê. Ele indica conteúdo mais largo que a tela, o defeito mais comum no celular. A correção costuma ser uma tabela que rola dentro da própria caixa, uma imagem sem `max-width` ou uma grade com largura mínima fixa.

O manifesto é gravado mesmo quando parte das capturas falha. Uma URL que não responde aparece em `errors` e o comando sai com exit code 5.

## Listar execuções

```
designfy capture list
```

A lista vem da mais recente para a mais antiga, com as URLs, os viewports e os arquivos de cada execução. Numa pasta vazia:

```
No capture runs found.
```

`--out <dir>` aponta para outra pasta. A TUI mostra as mesmas execuções no módulo Captures, com os arquivos, os tamanhos e o `overflowX` de cada captura.

## Como a revisão usa as capturas

A skill `designfy-review` segue um caminho fixo:

1. Lê o `manifest.json` e separa as capturas com `overflowX`, erro de console e erro de página.
2. Abre cada PNG, começando pelo celular, depois desktop, tablet, tela larga, tema escuro e movimento reduzido.
3. Compara com o guia e com o viewspec: a ação principal aparece, o contraste segura, o texto não corta, o foco aparece, o movimento tem função.
4. Classifica cada achado por gravidade e devolve para a especialista dona.
5. Pede uma captura nova depois das correções e repete a leitura.

A revisão não compara pixel a pixel. A skill examina a imagem junto com o manifesto e registra os achados.

## Revisões, gravidade e veredito

Cada revisão tem um nome: `<slug>-r1` para achar tudo, `<slug>-r2` para conferir as correções e o que elas trouxeram e `<slug>-r3` para fechar o que restou. Em cada nova revisão, confira os achados anteriores primeiro (resolvido, parcial ou aberto) e depois olhe a tela inteira de novo. O limite é de três revisões.

A gravidade orienta a correção:

| GRAVIDADE | O QUE SIGNIFICA | O QUE ACONTECE |
|---|---|---|
| alta | a tela falha no que promete ou fere uma regra sem exceção: rolagem horizontal, ação principal fora da dobra, texto ilegível, conteúdo inventado, erro de script, controle sem foco ou sem nome, contraste abaixo do mínimo | corrige antes de fechar |
| média | a tela funciona, mas parece genérica ou descuidada: tudo centralizado, seções iguais, estado vazio sem ação, movimento ausente | corrige ou registra o motivo em Notas |
| baixa | detalhe de ofício: palavra sozinha na última linha, aspas retas, sombra forte, alinhamento óptico | responde item a item |

Cada achado vai à especialista dona: `designfy-design-system` para cor, tipo, espaço e tema; `designfy-components` para componente, estado e dependência; `designfy-motion` para animação; `designfy-websites`, `designfy-content` e `designfy-webapps` para a composição do tipo de tela; `designfy-responsive` para estouro, empilhamento e toque; `designfy-accessibility` para foco, teclado, nomes e semântica; `designfy-viewspec` para o arquivo ou uma regra do guia. O que falta de conteúdo vai para você, pela `designfy`.

No fim, o veredito é um dos três:

| VEREDITO | CONDIÇÃO |
|---|---|
| aprovada | nenhum achado de gravidade alta ou média aberto; os baixos respondidos |
| aprovada com ressalvas | nenhum de gravidade alta; médios abertos com o motivo em Notas |
| reprovada | algum de gravidade alta aberto |

Só `aprovada` e `aprovada com ressalvas` permitem o estado `done` no viewspec.

## Fechar a tela

Quando a checklist está completa e a revisão terminou, o viewspec registra a execução usada em `captures` e muda o status para `done`:

```
captures:
  - 2026-11-24T14-10-00-000Z
```

Um item pendente recusa `done`. A captura que documenta o estado fica na pasta da execução e o caminho aparece no relato da skill.

## Captura em CI

O mesmo comando serve para o build:

```
designfy deps install testing
npx playwright install --with-deps chromium
designfy capture run http://localhost:4321/ --viewport mobile,desktop --
out .designfy/captures
designfy capture list --json
```

O passo precisa do servidor de desenvolvimento ou da prévia do build no ar. O `--json` entrega o manifesto em `data`, que o script publica como artefato do job. O `overflowX: true` vira uma falha do job quando o script confere o campo.

O `--reduced-motion` gera uma segunda captura com a preferência ligada e serve para conferir que a tela não depende de animação para aparecer.

## Conferir

- `designfy deps install testing` e `npx playwright install chromium` rodaram.
- `designfy capture run` grava a pasta e o `manifest.json`.
- `designfy capture list` mostra a execução mais recente.
- `designfy capture run --theme light,dark` grava um PNG por viewport em cada tema.
- O manifesto registra `scrollSteps` e `scrollBudgetExhausted` da preparação.
- O manifesto tem `overflowX: false` nas capturas de celular.
- Cada achado tem gravidade e responsável. Você conferiu a correção em `r2`.
- O veredito é `aprovada` ou `aprovada com ressalvas` antes do `done`.
- O viewspec em `done` cita a execução em `captures`.

# Capítulo 9: TUI

Você digita `designfy` no terminal e a Home abre em forma de menu: a faixa do próximo passo, um cartão numerado por módulo com o resumo vivo do projeto e a linha de totais. Não existe barra de abas. Você escolhe o módulo pelo número, pelas setas ou pelo clique e volta com o chip `← Back`.

## Abrir

A TUI exige stdin e stdout num terminal interativo. Num pipe, encerra com exit code 8:

```
designfy
designfy tui --tab theme
```

A tela inicial vem de `tui.initialTab`, com `overview` como padrão. O `--tab` aceita qualquer ID de módulo e tem prioridade sobre a preferência: `overview`, `theme`, `catalog`, `components`, `skills`, `deps`, `mcp`, `viewspecs`, `captures`, `project`, `settings`, `credentials`, `docs` e `about`.

## Cabeçalho e Home

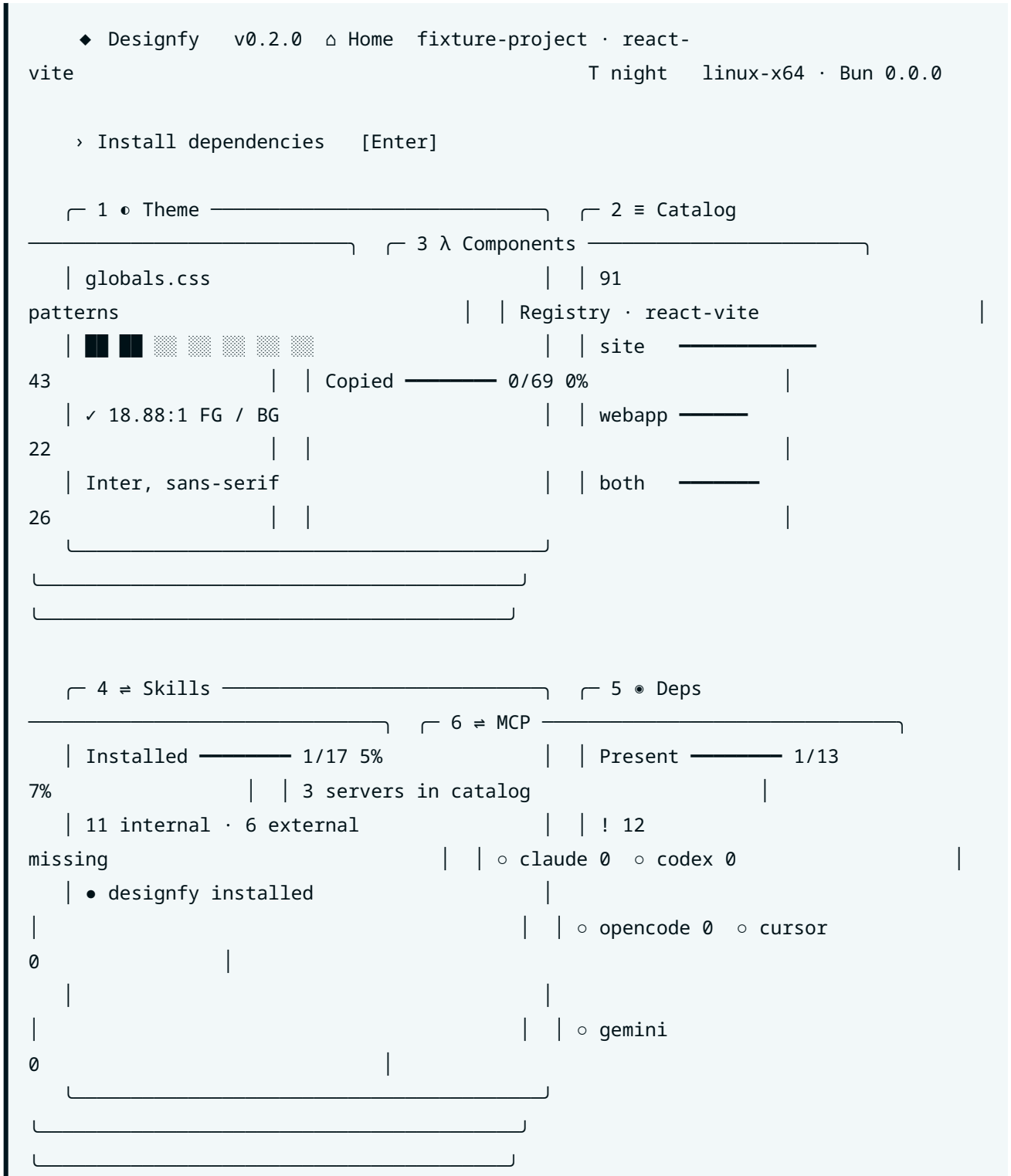
O cabeçalho mostra a marca e a versão, a trilha da tela atual, a pasta do projeto, o framework, o chip do tema, a plataforma e a versão do Bun. Clique na marca para voltar à Home; o chip `T` troca a paleta. Quando a largura aperta, a pasta encurta e a plataforma some.

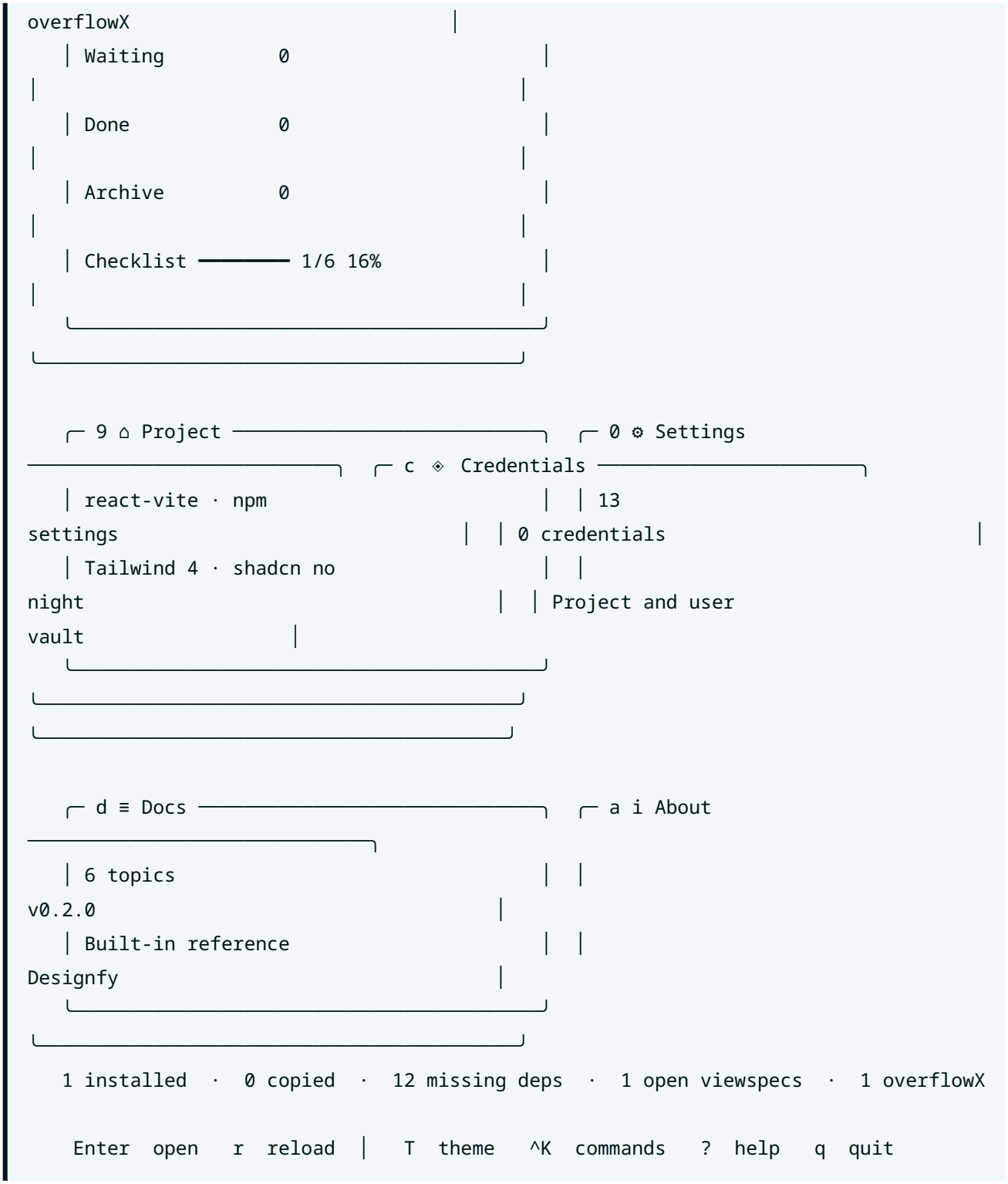
A Home responde “o que faço agora?” com a faixa de próximo passo. A ordem é: inicializar o projeto, adicionar o tema, criar o guia, instalar dependências e continuar um viewspec aberto. A faixa exibe `All set` quando nada falta. Pressione `Enter` ou clique nela para abrir o módulo indicado.

Abaixo fica a grade de cartões. Cada um traz o resumo vivo do módulo: os tokens e o contraste no Theme, as superfícies no Catalog, os componentes marcados no Components, as skills instaladas no Skills, os pacotes no Deps, os clientes no MCP, o progresso das checklists no Viewspecs, as capturas e o `overflowX` no Captures. A linha de totais destaca o que pede atenção.

Mova o foco com as setas ou com `h`, `j`, `k` e `l`. Pressione `Enter` ou `Space` para abrir o cartão, ou clique nele. `r` relê o projeto. `c`, `d` e `a` abrem Credentials, Docs e About.

# Quadro da Home


◆ Designfy v0.2.0 ⌂ Home fixture-project · react-vite
T night linux-x64 · Bun 0.0.0
› Install dependencies [Enter]
1 ◐ Theme
globals.css
Inter, sans-serif
2 ≡ Catalog
91 patterns
Registry · react-vite
site
43
Copied 0/69 0%
webapp
22
both
26
3 λ Components
✓ 18.88:1 FG / BG
4 ⇌ Skills
Installed 1/17 5%
11 internal · 6 external
● designfy installed
5 ◉ Deps
6 ⇌ MCP
Present 1/13 7%
3 servers in catalog
! 12 missing
○ claude 0 ○ codex 0
○ opencode 0 ○ cursor 0
○ gemini 0
7 ≡ Viewspecs
✓ Guide · 33 rules
Draft 0
Active 1
8 ❯ Captures
2026-11-24-test
2026-11-24 · 1 shots
✗ 1




## Os módulos

Cada módulo tem um painel de lista e um de detalhe, com o título dentro da borda. O painel em foco destaca o título e a borda.

| TECLA | MÓDULO | PAINÉIS | TECLAS DO MÓDULO |
|---|---|---|---|
| | Home | próximo passo, cartões e totais | `Enter` abre, `r` relê, `c`, `d`, `a` abrem Credentials, Docs e About |
| `1` | Theme | tokens com amostra truecolor e detalhe da cor com contraste WCAG | `t` claro e escuro, `g` grupo, `Enter` cor, `/` filtro, `r` relê |
| `2` | Catalog | padrões por categoria e detalhe com uso, anatomia, estados, responsivo, movimento, acessibilidade, tokens e fontes | `s` superfície, `k` tipo, `Enter` detalhe, `/` filtro, `r` recarrega |
| `3` | Components | registro por categoria e detalhe com arquivos e dependências | `f` React e Astro, `Space` marca, `a` lista, `n` limpa, `i` copia, `Enter` detalhe |
| `4` | Skills | skills por tipo, com origem, escopo e estado, e detalhe | `g` tipo, `Space` marca, `a` lista, `n` limpa, `i` instala, `u` atualiza, `d` remove |
| `5` | Deps | grupos e detalhe das versões declarada e instalada | `Space` marca, `a` lista, `n` limpa, `c` confere, `i` instala, `Enter` detalhe |
| `6` | MCP | servidores do catálogo e estado por cliente | `Space` marca, `a` lista, `n` limpa, `c` cliente, `f` substitui, `i` grava |
| `7` | Viewspecs | guia, telas, estados e medidores de checklist | `v` guia e telas, `s` estado, `Enter` lê o arquivo, `Tab` volta |
| `8` | Captures | execuções, viewports, imagens, tamanhos e erros | `Enter` imagem, `o` copia o caminho, `/` filtro, `r` recarrega |
| `9` | Project | ambiente, estado do Designfy e próximos passos | `r` recarrega, `c` copia a configuração MCP |

| TECLA | MÓDULO | PAINÉIS | TECLAS DO MÓDULO |
|---|---|---|---|
| `0` | Settings | key, valor e padrão, com detalhe | `Enter` ou `e` edita, `x` ou `Delete` restaura, `r` recarrega |
| `c` na Home | Credentials | nomes, origem da master key e total do cofre | `a` adiciona, `Enter` ou `e` edita, `d` ou `Delete` remove |
| `d` na Home | Docs | índice de tópicos e o documento em Markdown | `t` escolhe o tópico e as teclas de rolagem |
| `a` na Home | About | binário, licença, links e créditos | `c` copia o link do projeto |

## Teclas globais

| TECLA | AÇÃO |
|---|---|
| `1` a `9` e `0` | abre o módulo numerado, de qualquer tela |
| `Tab` e `Shift+Tab` | move o foco entre os painéis |
| setas, `PgUp`, `PgDn`, `Home` e `End` | move a seleção ou rola a prévia |
| `Esc` ou `Backspace` | volta à Home; na Home, `Esc` sai |
| `Ctrl+K` ou `:` | abre a paleta de comandos |
| `T` | alterna o tema |
| `?` | abre ou fecha o help |
| `q` ou `Ctrl+C` | sai |

Para procurar um módulo ou uma ação na paleta, digite as letras em sequência. No pé, a barra de teclas mostra os atalhos do módulo, um divisor e os globais; clique num chip para executar.

## Listas, filtros e planos

Use `/` nas listas de Theme, Catalog, Components, Skills, Deps, MCP, Viewspecs e Captures. A busca ignora acentos e caixa e o título mostra a quantidade visível, o total e a busca. Pressione `Enter` para manter o filtro ou `Esc` para restaurar a busca anterior.

Nas listas de seleção múltipla, `Space` marca o item, `a` marca todos os visíveis e `n` limpa.

Toda ação de escrita segue o mesmo caminho: você usa o atalho para ver o plano com os comandos ou arquivos. Use o atalho de novo para iniciar a execução. Acompanhe o medidor, o tempo e os passos recentes até ler o resultado. Outra tecla cancela o plano.

## Tema, ícones e mouse

`T` ou o chip do cabeçalho alterna entre `terminal`, `night` e `day`. A mudança mantém o módulo aberto e grava `tui.theme`. `terminal` herda as cores do terminal. `night` e `day` usam petróleo, menta e âmbar da marca Designfy e ficam iguais em qualquer terminal.

`tui.icons` aceita `auto`, `nerd`, `unicode` ou `ascii`. O `auto` usa Nerd Font no Ghostty, no Kitty, no WezTerm e no Omarchy, ASCII no console Linux e Unicode nos demais terminais. `tui.motion=false` desliga as animações curtas de diálogos e toasts. `tui.mouse=false` devolve a seleção de texto ao terminal; com o mouse ligado, use `Shift` e arraste para selecionar.

## Tamanhos

A Home usa três colunas a partir de 110 células, duas entre 80 e 109 e uma abaixo de 80. Com menos de 32 linhas, a TUI compacta o conteúdo e mantém `? help` e `q quit` na barra. Abaixo de 60 por 16, aparece a mensagem de terminal pequeno até você ampliar.

## Conferir

- `designfy` abre o menu da Home sem barra de abas. Use `q` para fechar.
- A faixa de próximo passo abre o módulo que resolve o que falta.
- `1` a `9` e `0` abrem os módulos numerados de qualquer tela.
- O chip `T` e a tecla `T` alternam `terminal`, `night` e `day`.
- `Ctrl+K` encontra um módulo pelo nome.
- Skills, Deps e MCP mostram o plano antes de instalar ou gravar.
- Em 60 por 16 a TUI mostra a mensagem de terminal pequeno.

# Capítulo 10: Agentes e MCP

O agente abre a conversa e chama a tool `project_status`. Recebe o framework, o tema, as skills instaladas e o próximo viewspec livre. Em seguida chama `catalog_list` com `kind: "dashboard"`, escolhe os fragmentos e chama `components_add` com `dryRun: true` para ver os arquivos antes de escrever. Nada disso passa pelo terminal: o agente fala com o binário por MCP.

## O servidor

```
designfy mcp
```

O comando transforma o processo num servidor Model Context Protocol sobre stdio. Stdout fica reservado para JSON-RPC; diagnósticos vão para stderr. O cliente inicia o processo e conversa com ele.

A configuração no cliente:

```
{
  "mcpServers": {
    "designfy": { "command": "designfy", "args": ["mcp"] }
  }
}
```

No Claude Code:

```
claude mcp add designfy -- designfy mcp
```

Para desenvolver sem compilar:

```
{
  "mcpServers": {
    "designfy": { "command": "bun", "args": ["run", "src/main.ts", "mcp"] }
  }
}
```

As tools de projeto usam a pasta de trabalho do servidor. Abra o cliente na raiz do site ou da aplicação, ou defina a pasta na linha de comando:

```
designfy mcp --project ./website
```

A option global `--project` ( `-C` ) vale para o servidor. A variável `DESIGNFY_PROJECT` define o padrão quando ela falta. Nas tools que aceitam o argumento `project`, o caminho pode ser absoluto ou relativo à pasta do servidor e vale só para aquela chamada. Os outros caminhos da mesma chamada partem do projeto selecionado.

## As tools

As tools leem o mesmo domínio que o CLI. A tabela abaixo segue os grupos do `mcp --list`.

| TOOL | ENTRADA | DEVOLVE |
|---|---|---|
| `project_status` | nenhuma | detecção, configuração, totais, guia e CSS do tema |
| `skills_list` | `internal?` , `external?` | skills com estado de instalação |
| `deps_list` , `deps_check` | `group?` | grupos, faixas e estado; a conferência reprovada vem com `isError` |
| `catalog_list` , `catalog_show` , `catalog_search` | filtros, `id` , `text` | fragmentos de interface e a entrada inteira |
| `components_list` , `components_show` | filtros, `name` , `framework?` , `source?` | metadados, arquivos e conteúdo |
| `components_add` | `names` , `framework?` , `path?` , `overwrite?` , `dryRun?` | arquivos gravados e dependências |
| `theme_show` , `theme_check` | `file?` , `group?` | tokens por grupo; contraste em claro e escuro |
| `palette_generate` , `palette_wheel` , `palette_contrast` | `base` , uma cor, duas cores | escalas, harmonias e vereditos |
| `viewspec_guide` , `viewspec_list` , `viewspec_show` | filtros, `id` | guia, lista e o arquivo da tela |
| `viewspec_progress` , `viewspec_validate` | `id?` | total e próximo `id` ; diagnósticos por arquivo e linha |
| `capture_run` , `capture_list` | `urls` , opções; `outDir?` | manifesto e imagens |
| `mcp_servers` | nenhuma | servidores sugeridos e estado por cliente |
| `doctor` | nenhuma | verificações de ambiente, sem gravar arquivo |

| TOOL | ENTRADA | DEVOLVE |
|---|---|---|
| `about`, `exit_codes`, `skill`, `config_list`, `config_get`, `credentials_status` | conforme a tool | identidade, tabela, documentação, settings e metadados do cofre |

As tools que escrevem no projeto são `components_add` e `capture_run`. Todas as outras só leem. Nenhuma tool devolve valor de credencial; o cofre expõe apenas metadados.

## Modo de leitura

Para um agente que só consulta, inicie o servidor sem as tools que gravam:

```
designfy mcp --read-only
```

O modo retira `components_add` e `capture_run` e mantém todas as outras. Use numa revisão, numa auditoria ou num agente que não deve alterar arquivos.

Para ver a superfície sem iniciar o servidor:

```
designfy mcp --list
designfy mcp --list --read-only
designfy mcp --list --json
```

A listagem mostra cada tool com a descrição, os resources e os prompts:

```
MCP server designfy v0.2.0

Tools:
  project_status  Use to inspect the current web project. Returns detection,
effective project settings, installation counts, viewspec progress, guide counts and
the Tailwind stylesheet.
  catalog_list  List candidate interface fragments for a site or webapp. Returns
entries filtered by surface, category, and viewspec kind.
  ...
Resources:
  designfy://about  Binary identity and runtime as JSON.
  designfy://exit-codes  Exit-code table as JSON.
  designfy://skill  Full documentation in Markdown.
  designfy://skill/{topic}  One documentation section. Topics: cli, tui, mcp,
config, exit-codes, reference.
Prompts:
  operate  Guides the agent to read the documentation and run a task through the
CLI.
```

## Resources e prompts

| RESOURCE | CONTEÚDO |
|---|---|
| `designfy://about` | identidade do binário em JSON |
| `designfy://exit-codes` | tabela de exit codes em JSON |
| `designfy://skill` | documentação completa em Markdown |
| `designfy://skill/{topic}` | uma seção da documentação |

O prompt `operate` recebe `task` e instrui o agente a ler `skill cli` e executar a tarefa pelo CLI com `--json`. É o caminho para um agente que prefere o terminal ao MCP.

## Configurar os clientes

O catálogo de servidores traz três entradas: o próprio Designfy, o shadcn e o Playwright. O Designfy e o shadcn são recomendados; o Playwright entra com pedido.

```
designfy mcp servers
```

```
  designfy (Designfy)      claude: absent, codex: absent, opencode: absent, cursor:
absent, gemini: absent
  shadcn (shadcn/ui)       claude: absent, codex: absent, opencode: absent, cursor:
absent, gemini: absent
  playwright (Playwright)  claude: absent, codex: absent, opencode: absent, cursor:
absent, gemini: absent
```

O `mcp setup` grava as entradas nos arquivos de projeto dos clientes:

| CLIENTE | ARQUIVO |
|---|---|
| `claude` | `.mcp.json` |
| `codex` | `.codex/config.toml` |
| `opencode` | `opencode.json` |
| `cursor` | `.cursor/mcp.json` |
| `gemini` | `.gemini/settings.json` |

```
designfy mcp setup --dry-run
```

```
  claude / designfy  planned add · /tmp/opencode/designfy-demo/.mcp.json
  claude / shadcn    planned add · /tmp/opencode/designfy-demo/.mcp.json
```

Sem `--client`, o comando usa os clientes que já têm arquivo no projeto e, quando nenhum tem, o `claude`. Para escolher:

```
designfy mcp setup designfy shadcn --client codex,opencode
```

O comando preserva as outras chaves e entradas. Uma entrada existente e diferente exige `--force`. `--dry-run` imprime o plano. Nenhuma entrada leva segredo: o servidor do Designfy não precisa de credencial para ler o projeto e as tools de catálogo não chamam API externa.

## Testar

O Inspector do MCP abre uma página local para chamar tools, ler resources e inspecionar as mensagens JSON-RPC:

```
npx @modelcontextprotocol/inspector designfy mcp
```

Para testar sem interface, um cliente MCP mínimo inicia o processo, pede a listagem de tools e chama `project_status`. O retorno precisa trazer o projeto da pasta do servidor.

## Conferir

- `designfy mcp --list` mostra as tools, resources e prompts.
- `designfy mcp --list --read-only` não lista `components_add` nem `capture_run`.
- `designfy mcp setup --dry-run` mostra os arquivos de cada cliente.
- O cliente reiniciado enxerga o servidor `designfy` e responde a `project_status`.
- Nenhuma resposta de tool traz valor de credencial.

# Capítulo 11: Referência

O agente errou o caminho e o comando responde com exit code 6. O script de CI reprovou o build com exit code 10. A TUI fechou com exit code 8 porque rodou sem terminal. Cada família de falha tem um número fixo. Consulte neste capítulo os números, as preferências, os arquivos e as mensagens que aparecem no dia a dia. Use como consulta depois de ler os capítulos anteriores.

## Preferências

As preferências ficam em `~/.designfy/settings.json`, com permissão 0600.

| KEY | TIPO | PADRÃO | FUNÇÃO |
|---|---|---|---|
| `output.format` | `text` ou `json` | `text` | formato padrão da saída do CLI |
| `output.colors` | boolean | `true` | cores ANSI na saída de texto |
| `tui.icons` | `auto`, `nerd`, `unicode`, `ascii` | `auto` | conjunto de ícones da TUI |
| `tui.initialTab` | ID de módulo | `overview` | tela aberta ao iniciar; `overview` abre a Home |
| `tui.motion` | boolean | `true` | animações curtas de diálogos e toasts |
| `tui.mouse` | boolean | `true` | clique, hover e roda na TUI |
| `tui.theme` | `terminal`, `night`, `day` | `terminal` | paleta da TUI |
| `skills.source` | string | `promovaweb/designfy` | repositório das skills internas |
| `skills.agents` | string | `universal,claude-code,codex,opencode` | agentes de destino separados por vírgula |
| `deps.packageManager` | `auto`, `npm`, `pnpm`, `bun`, `yarn` | `auto` | gerenciador da instalação |
| `capture.viewports` | string | `mobile,desktop` | viewports separados por vírgula |
| `capture.outDir` | string | `.designfy/captures` | pasta das capturas |
| `telemetry.enabled` | boolean | `false` | reservado para projetos derivados |

```
designfy config list
designfy config get tui.theme
designfy config set output.format json
designfy config unset telemetry.enabled
designfy config path
designfy config schema
```

O `config list` mostra o arquivo em uso e o valor de cada key:

```
File: ~/.designfy/settings.json (not created yet; defaults)

  capture.outDir       ".designfy/captures"
  capture.viewports    "mobile,desktop"
  deps.packageManager  "auto"
  output.colors        true
  output.format        "text"
  skills.agents        "universal,claude-code,codex,opencode"
  skills.source        "promovaweb/designfy"
  telemetry.enabled    false
  tui.icons            "auto"
  tui.initialTab       "overview"
  tui.motion           true
  tui.mouse            true
  tui.theme            "terminal"
```

`config path` imprime a pasta e os arquivos; `config schema`, o tipo, o padrão e a descrição de cada key. Uma key fora do schema é aceita como texto.

## O designfy.json

| CAMPO | VALOR |
|---|---|
| `schemaVersion` | `1` |
| `framework` | `astro`, `next`, `react-vite`, `react` ou `unknown` |
| `typescript` | booleano |
| `packageManager` | `bun`, `pnpm`, `yarn` ou `npm` |
| `tailwind` | versão maior ou `null` |
| `shadcn` | booleano |
| `paths.components` | pasta dos componentes copiados |

| CAMPO | VALOR |
| --- | --- |
| `paths.styles` | pasta dos estilos copiados e do CSS do tema |
| `paths.theme` | caminho opcional do CSS do tema, relativo à raiz ou absoluto; também aceito por `init --theme-path` |
| `paths.viewspec` | pasta dos viewspecs |
| `aliases.components` | alias que substitui `@designfy/components` |
| `theme.pairs` | pares de contraste do modo `custom`, com `fg`, `bg`, `label` e `min` |

Quando o arquivo existe, os campos definidos nele prevalecem sobre a detecção. A option global `--project` (`-C`) escolhe a pasta do projeto por execução. A variável `DESIGNFY_PROJECT` define o padrão.

## Arquivos e pastas

| CAMINHO | CONTEÚDO |
| --- | --- |
| `designfy.json` | projeto detectado e caminhos |
| `VIEWSPEC.md` | guia de interface do projeto |
| `viewspec/<id>-<slug>/VIEWSPEC.md` | especificação e checklist de uma tela |
| `.designfy/captures/<run>/` | capturas e `manifest.json` |
| `~/.designfy/settings.json` | preferências |
| `~/.designfy/credentials.json` | cofre cifrado com AES-256-GCM |
| `~/.designfy/master.key` | master key em hexadecimal |
| `skills-lock.json` | versões das skills instaladas |
| `.claude/skills/`, `.agents/skills/` | skills instaladas para os agentes |
| `src/components/designfy/` | componentes copiados, por padrão |
| `src/styles/` | estilos copiados, por padrão |

A pasta `~/.designfy/` é criada com permissão 0700. Os arquivos recebem permissão 0600. A master key vem de `DESIGNFY_MASTER_KEY` quando a variável existe e, sem ela, de `master.key`.

## Exit codes

| CÓDIGO | NOME | QUANDO APARECE |
|---|---|---|
| 0 | `SUCCESS` | operação concluída |
| 1 | `GENERIC_ERROR` | falha inesperada sem classificação |
| 2 | `USAGE` | argument, option ou subcommand inválido; binário sem argumentos fora de um TTY |
| 3 | `CONFIG` | settings.json ausente, inválido ou key desconhecida |
| 4 | `CREDENTIALS` | cofre ilegível, master key ausente ou credencial não encontrada |
| 5 | `NETWORK_OR_API` | rede, API fora do esperado ou processo externo que falhou |
| 6 | `NOT_FOUND` | recurso, componente, arquivo ou guia inexistente |
| 7 | `PERMISSION` | permissão negada no sistema de arquivos |
| 8 | `NO_TTY` | a TUI exige terminal interativo |
| 9 | `MCP` | servidor MCP não iniciou ou perdeu o transporte |
| 10 | `CHECK_FAILED` | conferência reprovada: dependência, contraste ou viewspec |
| 130 | `INTERRUPTED` | entrada cancelada ou processo interrompido |

Com `--json`, o mesmo número aparece em `error.code`. Algumas conferências entregam dados com exit code diferente de zero: confira `data.ok` no `doctor`, `data.valid` no `viewspec validate` e `data.errors` nas consultas de viewspecs.

Um tratamento em código confere `data.ok`, `data.valid` e `data.errors` antes de seguir.

## Variáveis de ambiente

| VARIÁVEL | EFEITO |
|---|---|
| `DESIGNFY_HOME` | troca `~/.designfy/` por outra pasta |

| VARIÁVEL | EFEITO |
| --- | --- |
| `DESIGNFY_PROJECT` | pasta do projeto por padrão no CLI e no servidor MCP; a option `--project` tem precedência |
| `DESIGNFY_MASTER_KEY` | master key em 64 caracteres hexadecimais; dispensa `master.key` |
| `DESIGNFY_CATALOG_DIR` | pasta com os JSONs de catálogo no lugar dos embutidos |
| `DESIGNFY_REGISTRY_FILE` | JSON com os registros embutidos de React e Astro |
| `DESIGNFY_E2E` | `1` habilita o teste de captura com Playwright |
| `NO_COLOR` | desliga as cores da saída de texto |

## Solução de problemas

| SINTOMA | CAUSA PROVÁVEL | O QUE FAZER |
| --- | --- | --- |
| `designfy.json already exists` com exit 2 | o arquivo já está na raiz | use `--force` só para substituir, ou edite o arquivo |
| `No package.json found in the current directory` com exit 6 | comando rodou fora da raiz | entre na raiz do site ou da aplicação |
| `No Tailwind stylesheet found` com exit 6 | o projeto não tem CSS com `@import "tailwindcss"` | rode `designfy components add designfy-theme` e ligue no CSS de entrada |
| `Theme check failed` com exit 10 | par abaixo de 4.5 ou papel obrigatório ausente | meça com `designfy palette contrast` e corrija o papel no arquivo de tokens |
| `Theme check failed` no modo `custom` | par do `theme.pairs` abaixo do `min` | ajuste os valores no `designfy.json` ou o `min` do par |
| `components add` recusa com exit 10 num projeto Tailwind 3 | o registro usa a sintaxe do Tailwind 4 | use o item como referência, migre o projeto ou copie com `--allow-tailwind3` |
| `trackedBy` aponta arquivo ausente | a spec ou a sitespec mudou de pasta de estado | atualize o `path` no front matter do viewspec |
| `missing animate-df-marquee` com exit 10 | o CSS de movimento ficou fora do arquivo lido | importe `designfy-motion.css` e use `--file` no CSS de entrada |
| `Missing: ...` com exit 10 | pacotes ausentes no `package.json` | rode `designfy deps install` |

| SINTOMA | CAUSA PROVÁVEL | O QUE FAZER |
|---|---|---|
| `Unknown component` com exit 6 | nome errado ou framework errado | confira `designfy components list --framework <framework>` |
| `Playwright is unavailable` com exit 10 | pacote ou navegador ausente | `designfy deps install testing` e `npx playwright install chromium` |
| `One or more screenshots failed` com exit 5 | URL fora do ar ou seletor sem resultado | suba o servidor, confira a URL e o `--selector` |
| `Project guide VIEWSPEC.md is missing` com exit 6 | o projeto ainda não tem guia | rode a skill `designfy-viewspec` no modo guia |
| `Viewspec validation passed.` com `guide: missing` | o aviso não reprova | crie o guia na próxima etapa |
| A TUI encerra com exit 8 | o comando rodou sem terminal interativo | use o CLI em script e a TUI só no terminal |
| `npx skills add` falha com exit 5 | sem rede ou agente inválido | confira a conexão, o nome do repositório e o nome do agente |
| `Shadcn items` na resposta do `components add` | o item exige primitivas do shadcn | rode o comando `npx shadcn@latest add` que a resposta indica |

Quando o sintoma não está na tabela, `designfy doctor` confere o ambiente sem alterar arquivo: runtime, terminal, pasta de dados, settings, master key, Node, npx, `designfy.json`, CSS do Tailwind e guia. Um `✗` reprova com exit code 3; os avisos `!` não reprovam.

## Onde encontrar o projeto

O código, os binários e as notas de versão ficam em promovaweb/designfy, um repositório público sob a AGPL 3.0. O pacote do npm é `@promovaweb/designfy`. As skills vêm da mesma origem. O guia do método usado para construir o binário fica no Termfy. A versão online deste guia e a documentação técnica ficam em promovaweb.com/docs/designfy. Você encontra a apresentação do projeto em promovaweb.com/designfy.

## Conferir

- `designfy config list` mostra as keys com o valor em vigor.
- `designfy config path` aponta a pasta e os arquivos de dados.
- A tabela de exit codes deste capítulo bate com `designfy --help`.
- `designfy doctor` termina com exit code 0 e sem `✗`.
- O sintoma encontrado aparece na tabela com a ação correspondente.