PAGESPEEDFY

Guia do usuário

Da medição no PageSpeed Insights à correção no site Astro

API key, medição, leitura do relatório, skill de otimização e referência completa do CLI.

Edição v0.2.1
Gerado em 5 de novembro de 2026
Fonte editorial: docs/user/

Sumário

Pagespeedfy: guia do usuário 4
- Instalação rápida 4
- Comece aqui 4
- Consulte pela tarefa 4

Visão geral 5
- O CLI mede e organiza 5
- A skill planeja e corrige 5
- Uma página por vez 6
- Por que a URL precisa ser pública 6

Instalação 7
- Requisitos 7
- Instalar o CLI 7
- Instalar a skill no projeto 7
- Ignorar os relatórios no Git 8
- Atualizar 8

API key e credencial 9
- Criar a chave 9
- Gravar a chave 9
- Onde a chave fica 9
- Conferir e remover 10
- Usar em CI 10
- Cuidados 10

Primeira medição 11
- Medir 11
- Ler o resumo 11
- Medir mais rápido enquanto ajusta 11
- Comparar 12
- Variação entre medições 12

Otimizar uma página com a skill 13
- Instalação 13
- Acionamento 13
- O que a skill faz 13
- O plano de correções 14

O que a skill não faz 14
Referências da skill 14
Ler o relatório 15
Pasta de saída 15
Arquivos 15
`report.json` (formatVersion 1) 16
Regras da normalização 18
Compatibilidade 18
Referência do CLI 19
Opções globais 19
Códigos de saída 19
Precedência da API key 19
`pagespeedfy auth login` 19
`pagespeedfy auth status` 20
`pagespeedfy auth logout` 21
`pagespeedfy run <url>` 22
`pagespeedfy report <caminho>` 24
`pagespeedfy compare <antes> <depois>` 25
Solução de problemas 27
`Nenhuma API key configurada` 27
`A API key foi recusada` 27
`A cota diária da API acabou` 27
`A API recusou a consulta (HTTP 400)` 27
`O PageSpeed Insights não alcança localhost` 27
`a API não respondeu em 180 s` 27
`runtimeError` preenchido no relatório 28
Notas muito diferentes entre duas medições 28
O relatório não mostra todos os itens 28

Pagespeedfy: guia do usuário

O Pagespeedfy mede uma página no PageSpeed Insights e transforma o resultado em correções no código de um site Astro. Você informa a URL. O CLI consulta a API do Google e guarda um relatório completo. A skill lê esse relatório, encontra os arquivos responsáveis e aplica as mudanças.

Este guia acompanha o percurso inteiro, da criação da API key à comparação entre a medição de antes e a de depois.

Instalação rápida

```
npm install -g @promovaweb/pagespeedfy
npx skills add https://github.com/promovaweb/pagespeedfy
pagespeedfy auth login
```

O primeiro comando instala o CLI. O segundo, rodado na raiz do projeto Astro, instala a skill a partir do repositório promovaweb/pagespeedfy. O terceiro grava a API key. Os detalhes estão em Instalação.

Comece aqui

1. Entenda como as peças se encaixam.
2. Instale o CLI pelo npm e a skill pelo `npx skills add`.
3. Crie e grave a API key.
4. Faça a primeira medição.
5. Peça ao agente para otimizar a página.

Consulte pela tarefa

- **Entender um campo do relatório:** Ler o relatório.
- **Conferir uma opção do terminal:** Referência do CLI.
- **Resolver um erro:** Solução de problemas.

Visão geral

O PageSpeed Insights avalia uma página em quatro categorias: Performance, Acessibilidade, Boas práticas e SEO. Ele roda o Lighthouse nos servidores do Google, em um celular simulado e em um desktop simulado, e junta os dados de usuários reais do Chrome UX Report quando a página tem tráfego suficiente.

O resultado chega como um JSON de vários megabytes. Ler esse JSON à mão para descobrir qual componente atrasa o carregamento toma tempo. O Pagespeedfy divide esse trabalho em duas partes.

O CLI mede e organiza

O comando `pagespeedfy run` consulta a API para uma URL, nas duas estratégias, e salva três coisas numa pasta datada:

- o JSON bruto de cada estratégia, sem nenhuma alteração;
- um `report.json` com as notas, as métricas, os dados de campo, as auditorias reprovadas ordenadas por impacto, as verificações manuais e a lista das auditorias aprovadas;
- um `report.md` com o mesmo conteúdo em formato de leitura.

A API key fica em `~/.pagespeedfy/credentials`, fora de qualquer projeto.

A skill planeja e corrige

A skill `pagespeedfy-otimizar-pagina` roda dentro de um agente de programação aberto no projeto Astro. Ela executa o CLI, lê todos os relatórios da página (o `report.json`, os JSON brutos e as medições anteriores) e segue cada item reprovado até o arquivo de origem: a rota em `src/pages/`, o layout, o componente, a imagem ou o script.

Com isso ela escreve um plano de correções, mostra o plano na tela e espera você autorizar. Só depois altera o código.

Cada auditoria recebe um escopo:

Escopo	Exemplo	O que a skill faz
Página	Imagem de capa sem `<Image>`	Propõe a correção na rota
Componente	Botão do header sem nome acessível	Propõe e lista as páginas que mudam junto
Global	`build.inlineStylesheets` no `astro.config.mjs`	Propõe com o diff resumido

Escopo	Exemplo	O que a skill faz
Fora do código	Cabeçalho de cache da CDN	Descreve o ajuste para a hospedagem

Depois das correções autorizadas, a skill roda o build. Quando existe uma URL de preview com as mudanças, ela mede de novo e compara.

Uma página por vez

O Pagespeedfy trabalha com uma URL por execução. Cada relatório fica ligado a uma rota, e a comparação de antes e depois mede o efeito das mudanças nessa rota. Para otimizar a home e o blog, rode a skill duas vezes.

Por que a URL precisa ser pública

O Lighthouse do PageSpeed Insights roda no Google, não na sua máquina. Ele não alcança `localhost` nem a rede interna. Meça a produção ou um deploy de preview. O CLI recusa URLs locais antes de gastar a cota.

Instalação

Requisitos

- Node.js 22.12 ou superior.
- Git.
- Um projeto Astro para otimizar.
- Um agente de programação com suporte a skills, como Codex ou Claude Code.

Instalar o CLI

O pacote está no npm como `@promovaweb/pagespeedfy`:

```
npm install -g @promovaweb/pagespeedfy
```

Para rodar sem instalar:

```
npx @promovaweb/pagespeedfy --version
```

Confira:

```
pagespeedfy --version
```

O comando imprime a versão, por exemplo `0.2.0`.

Instalar a skill no projeto

A skill fica na pasta `skills/` do repositório promovaweb/pagespeedfy, no formato lido pelo `npx skills`. Na raiz do projeto Astro, rode:

```
npx skills add https://github.com/promovaweb/pagespeedfy
```

O comando baixa a skill `pagespeedfy-otimizar-pagina` do GitHub, pergunta para quais agentes instalar (Codex, Claude Code e outros) e registra a origem no `skills-lock.json` do projeto. Para conferir:

```
npx skills list
```

Ignorar os relatórios no Git

Os relatórios trazem URLs e trechos de HTML que não pertencem ao repositório. No projeto Astro:

```
echo "pagespeedfy-reports/" >> .gitignore
```

A skill faz essa conferência sozinha na primeira execução.

Atualizar

```
npm install -g @promovaweb/pagespeedfy@latest
```

Para atualizar a skill no projeto, rode de novo:

```
npx skills add https://github.com/promovaweb/pagespeedfy
```

API key e credencial

A API do PageSpeed Insights aceita consultas sem chave, mas a cota anônima é compartilhada entre todos os usuários e costuma acabar cedo. Por isso o Pagespeedfy exige uma chave própria: a cota passa a ser do seu projeto no Google Cloud.

Criar a chave

1. Abra o Google Cloud Console e escolha ou crie um projeto.
2. Em **APIs e serviços** → **Biblioteca**, procure **PageSpeed Insights API** e clique em **Ativar**.
3. Em **APIs e serviços** → **Credenciais**, clique em **Criar credenciais** → **Chave de API**.
4. Edite a chave criada e, em **Restrições de API**, selecione só a PageSpeed Insights API.

A chave começa com `AIza` e tem 39 caracteres.

Gravar a chave

```
pagespeedfy auth login
```

O terminal pede a chave e não mostra o que você digita. Para conferir se ela funciona antes de gravar:

```
pagespeedfy auth login --validar
```

Essa opção faz uma consulta real de SEO em `https://example.com/` e leva alguns segundos.

Onde a chave fica

```
~/.pagespeedfy/
└── credentials
```

O arquivo é um JSON com a chave e as datas de criação e atualização:

```
{
  "version": 1,
  "apiKey": "AIza...",
  "createdAt": "2026-11-05T13:00:00.000Z",
  "updatedAt": "2026-11-05T13:00:00.000Z"
}
```

A pasta tem permissão `0700` e o arquivo `0600`: só o seu usuário lê.

Conferir e remover

```
pagespeedfy auth status   # mostra a chave mascarada e a origem em uso
pagespeedfy auth logout   # apaga o arquivo
```

Usar em CI

Em automação, defina a variável `PAGESPEEDFY_API_KEY`. Ela tem prioridade sobre o arquivo e evita gravar a chave no disco do runner.

```
PAGESPEEDFY_API_KEY="$PSI_KEY" pagespeedfy run https://promovaweb.com/
```

Cuidados

- Não passe a chave como argumento de comando. Ela ficaria no histórico do shell.
- Não grave a chave no projeto Astro nem em `.env` versionado.
- Não envie a chave no chat do agente. A skill pede que você rode `pagespeedfy auth login` no seu terminal.

Primeira medição

Medir

Na pasta do projeto Astro:

```
pagespeedfy run https://promovaweb.com/
```

O CLI consulta mobile e depois desktop. Cada consulta leva de 20 a 90 segundos. O progresso aparece no stderr:

```
Consultando mobile: https://promovaweb.com/
Consultando desktop: https://promovaweb.com/
mobile: performance 71 · accessibility 96 · best-practices 100 · seo 100 · 6
auditorias pendentes
desktop: performance 94 · accessibility 96 · best-practices 100 · seo 100 · 3
auditorias pendentes
Relatório: pagespeedfy-reports/promovaweb.com/home/20261105-103012/report.json
Resumo: pagespeedfy-reports/promovaweb.com/home/20261105-103012/report.md
```

Ler o resumo

```
pagespeedfy report pagespeedfy-reports/promovaweb.com/home/20261105-103012
```

O resumo mostra, por estratégia:

1. a nota de cada categoria;
2. as métricas de laboratório (LCP, CLS, TBT e outras);
3. os dados de campo do CrUX, quando existem;
4. as auditorias reprovadas, com a orientação do Lighthouse e os primeiros itens (URLs, seletores, economia estimada);
5. as verificações manuais.

Medir mais rápido enquanto ajusta

Para repetir só o que interessa:

```
pagespeedfy run https://promovaweb.com/ -s mobile -c performance
```

Comparar

Depois de publicar uma mudança, meça de novo e compare as duas pastas:

```
pagespeedfy compare \
  pagespeedfy-reports/promovaweb.com/home/20261105-103012 \
  pagespeedfy-reports/promovaweb.com/home/20261106-091544
```

A comparação mostra a diferença de notas e métricas e lista as auditorias resolvidas, as novas e as que continuam.

Variação entre medições

O Lighthouse simula rede e CPU, e duas medições seguidas podem diferir alguns pontos em Performance. Olhe as métricas e a lista de auditorias resolvidas antes de concluir que uma mudança funcionou. Os dados de campo refletem 28 dias de usuários reais e só mostram o efeito semanas depois.

Otimizar uma página com a skill

A skill leva um agente de programação da medição à correção de uma página Astro. Ela usa o CLI para medir e trabalha no código do projeto aberto no agente.

Instalação

Na raiz do projeto Astro, instale a skill direto do repositório no GitHub:

```
npx skills add https://github.com/promovaweb/pagespeedfy
```

O `npx skills` lê a pasta `skills/` do repositório e instala a skill nos agentes escolhidos. A skill precisa do CLI instalado e de uma chave gravada com `pagespeedfy auth login`.

Acionamento

```
Use $pagespeedfy-otimizar-pagina para otimizar https://promovaweb.com/blog/
```

Informe uma URL. Com várias, a skill pede que você escolha uma e lista as outras para depois.

O que a skill faz

1. Confere o CLI, a chave, o projeto Astro e o estado do Git.
2. Confere se `pagespeedfy-reports/` está no `.gitignore`. Se faltar, a linha entra no plano de correções.
3. Mede a página com `pagespeedfy run <url> --print path`.
4. Lê todos os relatórios: o `report.json`, os dois JSON brutos e as medições anteriores da mesma rota, para separar problemas recorrentes dos novos.
5. Acha a rota em `src/pages/`, o layout, os componentes e os arquivos de origem de cada item reprovado, sem alterar nada.
6. Cria o plano de correções em `plano.md`, na pasta da medição, com uma linha por auditoria: escopo, arquivo, correção proposta e efeito esperado.
7. **Mostra o plano na tela e espera você dizer que pode corrigir.**
8. Aplica só as linhas autorizadas e atualiza o status de cada uma no plano.
9. Roda o build e os testes do projeto.
10. Mede de novo quando houver URL de preview e compara com a linha de base.
11. Entrega o relatório final.

O plano de correções

O plano é o ponto de controle da skill. Nenhum arquivo do projeto muda antes da sua resposta. Você pode:

- autorizar o plano inteiro;
- autorizar só algumas linhas, por número, ou só alguns escopos;
- pedir ajustes. A skill atualiza o `plano.md`, mostra a nova versão e pergunta de novo.

A resposta fica registrada na seção "Autorização" do `plano.md`. Se durante a correção aparecer uma mudança que não estava no plano, a skill acrescenta a linha e pede autorização outra vez.

Cada linha recebe um de quatro escopos:

Escopo	Exemplo
Página	Imagem de capa sem `<Image>` na própria rota
Componente	Botão do header sem nome acessível, usado em várias páginas
Global	`build.inlineStylesheets` no `astro.config.mjs`
Fora do código	Cabeçalho de cache da CDN

O que a skill não faz

- Não mede mais de uma página por execução.
- Não pede nem mostra a API key.
- Não altera o projeto antes de você autorizar o plano.
- Não faz deploy, commit nem push.
- Não altera hospedagem, CDN ou DNS.
- Não remove conteúdo, funcionalidade ou analytics para ganhar nota sem perguntar.

Referências da skill

- `references/leitura-relatorio.md`: todos os campos do `report.json` e como consultar o JSON bruto.
- `references/astro-correcoes.md`: correção recomendada para cada ID de auditoria de Performance, Acessibilidade, Boas práticas e SEO.
- `references/modelo-plano.md`: estrutura do `plano.md`.

Ler o relatório

Cada `pagespeedfy run` cria uma pasta própria. Nada é sobrescrito: duas medições da mesma página ficam lado a lado e podem ser comparadas com `pagespeedfy compare`. Se duas execuções começarem no mesmo segundo, a segunda pasta recebe o sufixo `-2`.

Pasta de saída

```
<output>/<host>/<rota>/<AAAAMMDD-HHMMSS>/
├── mobile.raw.json
├── desktop.raw.json
├── report.json
└── report.md
```

Parte	Regra
`<output>`	`--output`, padrão `pagespeedfy-reports` no diretório atual
`<host>`	Host da URL; `:` da porta vira `_`
`<rota>`	Segmentos do caminho unidos por `--`, sem acentos; `/` vira `home`; query string vira sufixo `--q-...`
`<AAAAMMDD-HHMMSS>`	Data e hora locais do início da execução

Arquivos

- **`<estratégia>.raw.json`**: resposta da API sem nenhuma alteração. Contém o `lighthouseResult` completo, incluindo auditorias aprovadas, telas de carregamento em base64 e o mapa de entidades.
- **`report.json`**: relatório normalizado, descrito abaixo.
- **`report.md`**: o mesmo relatório em Markdown, com até cinco itens por auditoria. Pode ser gerado de novo com `pagespeedfy report`.

`report.json` (formatVersion 1)

```
{
  "formatVersion": 1,
  "tool": "pagespeedfy",
  "toolVersion": "0.2.0",
  "generatedAt": "2026-11-05T13:30:12.000Z",
  "url": "https://promovaweb.com/",
  "locale": "pt-BR",
  "categories": ["performance", "accessibility", "best-practices", "seo"],
  "maxItems": 50,
  "runs": [
    {
      "strategy": "mobile",
      "requestedUrl": "https://promovaweb.com/",
      "finalUrl": "https://promovaweb.com/",
      "fetchTime": "2026-11-05T13:30:40.000Z",
      "lighthouseVersion": "13.0.0",
      "userAgent": "Mozilla/5.0 ...",
      "runWarnings": [],
      "runtimeError": null,
      "scores": { "performance": 71, "accessibility": 96, "best-practices": 100,
"seo": 100 },
      "labMetrics": {
        "largest-contentful-paint": { "title": "Largest Contentful Paint",
"displayValue": "3,4 s", "numericValue": 3412, "score": 0.66 }
      },
      "field": {
        "page": { "overallCategory": "AVERAGE", "metrics": {
"LARGEST_CONTENTFUL_PAINT_MS": { "percentile": 2900, "category": "AVERAGE" } } },
        "origin": null
      },
      "failedAudits": [
        {
          "id": "image-delivery-insight",
          "categories": ["performance"],
          "group": "insights",
          "weight": 0,
          "title": "Melhorar a entrega de imagens",
          "description": "...",
          "score": 0.5,
          "scoreDisplayMode": "metricSavings",
          "displayValue": "Economia estimada de 180 KiB",
          "numericValue": null,
```

```
          "metricSavings": { "LCP": 300 },
          "detailsType": "table",
          "headings": [{ "key": "url", "label": "URL" }],
          "items": [{ "url": "https://promovaweb.com/cover.png", "wastedBytes":
184320, "node": { "selector": "main > img", "snippet": "<img ...>" } }],
          "totalItems": 1,
          "truncated": false
        }
      ],
      "manualAudits": [{ "id": "focus-traps", "category": "accessibility", "title":
"..." }],
      "passedAuditIds": ["document-title", "image-alt"],
      "rawFile": "mobile.raw.json"
    }
  ]
}
```


Regras da normalização

- **Notas:** `score` da categoria multiplicado por 100 e arredondado. Categoria sem nota fica `null`.
- **Métricas de laboratório:** FCP, LCP, TBT, CLS, Speed Index e TTI. Elas ficam fora de `failedAudits` mesmo quando reprovadas.
- **Auditoria reprovada:** pertence a uma categoria consultada, tem `scoreDisplayMode` igual a `binary`, `numeric` ou `metricSavings` e nota menor que 0,9 (ou nula).
- **Ignoradas:** `notApplicable`, `informative` e `error` não entram em nenhuma lista. Continuam no JSON bruto.
- **Manuais:** `scoreDisplayMode` igual a `manual` vai para `manualAudits`.
- **Ordem:** peso na categoria (maior primeiro), soma de `metricSavings` (maior primeiro), nota (menor primeiro) e ID.
- **Itens:** os primeiros `maxItems`, sem as chaves internas `boundingRect`, `lhId` e `path`. `totalItems` guarda a contagem original.
- **Dados de campo:** `loadingExperience` vira `field.page` e `originLoadingExperience` vira `field.origin`. Bloco sem métricas vira `null`.

Compatibilidade

Um campo novo pode entrar em `report.json` sem mudar `formatVersion`. Remover um campo ou mudar o tipo dele exige `formatVersion` novo e registro no CHANGELOG.

Referência do CLI

Esta referência cobre todos os comandos do `pagespeedfy` a partir da versão 0.1.0. A fonte é `src/cli.ts`. O teste `tests/docs.test.ts` confere que cada comando implementado tem uma seção aqui, com pelo menos cinco exemplos.

Opções globais

Opção	Tipo	Padrão	Efeito
`--json`	flag	desligada	Imprime a saída em JSON, para scripts e agentes
`-V`, `--version`	flag		Imprime a versão
`-h`, `--help`	flag		Mostra a ajuda do comando

`--json` vem antes do comando: `pagespeedfy --json auth status`.

Códigos de saída

Código	Significado
`0`	Sucesso
`1`	Falha de operação: sem chave, API recusou, arquivo inválido
`2`	Uso incorreto: URL inválida ou local, opção com valor inválido

Precedência da API key

1. Variável `PAGESPEEDFY_API_KEY`.
2. Arquivo `~/.pagespeedfy/credentials` (ou `$PAGESPEEDFY_HOME/credentials`).

Sem nenhuma das duas, os comandos que consultam a API falham com código `1`.

`pagespeedfy auth login`

Grava a API key do PageSpeed Insights em `~/.pagespeedfy/credentials`.

Opção	Tipo	Obrigatória	Padrão	Efeito
`--stdin`	flag	não	desligada	Lê a chave do stdin em vez de perguntar
`--validar`	flag	não	desligada	Consulta `https://example.com/` (mobile, SEO) com a chave antes de gravar

Saída: linha com a chave mascarada e o caminho. Com `--json`: `{ path, apiKey, validated, updatedAt }`.

Efeito persistente: cria `~/.pagespeedfy/` com permissão `0700` e grava `credentials` com `0600`, de forma atômica. Ao regravar, preserva `createdAt`.

Permissões: escrita na pasta pessoal. Nenhuma alteração no projeto.

Recusas: sem terminal interativo e sem `--stdin` (código 2); chave vazia (código 2); com `--validar`, chave recusada pela API (código 1, nada é gravado).

Limites: uma chave por máquina. A chave fora do formato `AIza` + 35 caracteres gera aviso, mas é gravada.

Exemplos:

```
# 1. Pergunta a chave no terminal, sem eco
pagespeedfy auth login

# 2. Valida a chave com uma consulta real antes de gravar
pagespeedfy auth login --validar

# 3. Lê a chave de um gerenciador de senhas
op read "op://Promovaweb/PageSpeed/credential" | pagespeedfy auth login --stdin

# 4. Grava em outra pasta base, para CI ou testes
PAGESPEEDFY_HOME=/tmp/psf pagespeedfy auth login --stdin < chave.txt

# 5. Saída em JSON para um script de setup
pagespeedfy --json auth login --stdin < chave.txt
```

`pagespeedfy auth status`

Mostra qual chave será usada e de onde ela vem.

Sem opções próprias.

Saída: caminho do arquivo, chave mascarada do arquivo e da variável e a origem em uso. Com `--json`: `{ path, file, env, active }`. O campo `active` vale `"env"`, `"arquivo"` ou `null`.

Efeito persistente: nenhum.

Recusas: arquivo corrompido (código 1). Sem chave configurada, imprime a orientação e sai com código 1.

Limites: não testa a chave na API. Use `auth login --validar` para isso.

Exemplos:

```
# 1. Conferência rápida
pagespeedfy auth status

# 2. Uso em script: falha se não houver chave
pagespeedfy auth status > /dev/null || echo "configure a chave"

# 3. Origem em JSON
pagespeedfy --json auth status | jq -r .active

# 4. Confirma que a variável tem prioridade
PAGESPEEDFY_API_KEY=AIza... pagespeedfy auth status

# 5. Inspeciona outra pasta base
PAGESPEEDFY_HOME=/tmp/psf pagespeedfy auth status
```

`pagespeedfy auth logout`

Remove o arquivo de credencial.

Sem opções próprias.

Saída: `Removido: <caminho>.` ou `Nenhuma credencial gravada.` Com `--json`: `{ path, removed }`.

Efeito persistente: apaga `~/.pagespeedfy/credentials`. A pasta e os relatórios ficam.

Recusas: nenhuma; sem arquivo, sai com código 0.

Limites: não altera `PAGESPEEDFY_API_KEY` e não revoga a chave no Google Cloud.

Exemplos:

```
# 1. Remove a chave da máquina
pagespeedfy auth logout

# 2. Troca de chave
pagespeedfy auth logout && pagespeedfy auth login

# 3. Resultado em JSON
pagespeedfy --json auth logout

# 4. Limpeza de pasta de teste
PAGESPEEDFY_HOME=/tmp/psf pagespeedfy auth logout

# 5. Confirma a remoção
pagespeedfy auth logout && pagespeedfy auth status
```

`pagespeedfy run <url>`

Consulta o PageSpeed Insights para **uma** página e salva o relatório.

Argumento ou opção	Tipo	Obrigatório	Padrão	Efeito
`<url>`	texto	sim		URL pública e completa, `http` ou `https`
`-s`, `--strategy`	`mobile`, `desktop`, `all`	não	`all`	Estratégias consultadas, em sequência
`-c`, `--category`	lista	não	as quatro	`performance`, `accessibility`, `best-practices`, `seo`. Repita a opção ou separe por vírgula
`-l`, `--locale`	texto	não	`pt-BR`	Idioma dos títulos e descrições das auditorias
`-o`, `--output`	caminho	não	`pagespeedfy-reports`	Pasta base dos relatórios, relativa ao diretório atual

Argumento ou opção	Tipo	Obrigatório	Padrão	Efeito
`--max-items`	inteiro > 0	não	`50`	Itens de detalhe preservados por auditoria no `report.json`
`--timeout`	segundos inteiros > 0	não	`180`	Tempo máximo de cada consulta
`--retries`	inteiro > 0	não	`3`	Tentativas por estratégia em HTTP 429 e 5xx
`--print`	`summary`, `markdown`, `path`	não	`summary`	O que imprimir ao fim

Saída: com `summary`, uma linha de notas por estratégia e os caminhos do `report.json` e do `report.md`. Com `markdown`, o relatório inteiro. Com `path`, só o caminho absoluto do `report.json`. Com `--json`: `{ dir, report, markdown, runs: [{ strategy, scores, failedAudits }] }`. O progresso vai para o stderr.

Efeito persistente: cria `<output>/<host>/<rota>/<AAAAMMDD-HHMMSS>/` com `<estratégia>.raw.json`, `report.json` e `report.md`. A rota `/` vira `home`; `/blog/post/` vira `blog--post`. Detalhes em Ler o relatório.

Permissões: leitura da chave; escrita na pasta de saída; acesso HTTPS a `www.googleapis.com`.

Recusas:

- URL sem protocolo, com protocolo diferente de http/https, `localhost`, IP privado (`10.*`, `172.16-31.*`, `192.168.*`, `127.*`) ou `.local`: código 2.
- Sem API key: código 1.
- HTTP 400 (URL inacessível para o Google) e 403 (chave recusada ou API desativada): código 1, sem nova tentativa.
- HTTP 429 de cota diária esgotada: código 1, sem nova tentativa.
- 429 por minuto, 5xx e tempo esgotado: repete com espera de 2 s, 4 s e assim por diante, e falha após `--retries` tentativas.

Limites: uma URL por execução. A API pode levar de 20 a 90 segundos por estratégia. A chave nunca aparece em mensagens: o CLI troca o valor por `***`.

Exemplos:

```
# 1. Medição completa, mobile e desktop
pagespeedfy run https://promovaweb.com/

# 2. Só mobile e só Performance, para iterar rápido
pagespeedfy run https://promovaweb.com/blog/ -s mobile -c performance

# 3. Acessibilidade e SEO em inglês
pagespeedfy run https://promovaweb.com/planos/ -c accessibility,seo -l en

# 4. Caminho do report.json para encadear com outro comando
REPORT=$(pagespeedfy run https://promovaweb.com/ --print path)

# 5. Pasta própria, mais itens por auditoria e JSON de resumo
pagespeedfy --json run https://preview.promovaweb.com/ -o relatorios --max-items 200
```

`pagespeedfy report <caminho>`

Reimprime um relatório salvo.

Argumento	Tipo	Obrigatório	Efeito
`<caminho>`	caminho	sim	Pasta da execução ou arquivo `report.json`

Saída: o Markdown do relatório, gerado de novo a partir do `report.json`. Com `--json`, o `report.json` inteiro.

Efeito persistente: nenhum.

Recusas: caminho sem `report.json` ou JSON que não foi gerado pelo Pagespeedfy (código 1).

Limites: não consulta a API. Mostra até cinco itens por auditoria no Markdown.

Exemplos:

```
# 1. Pela pasta da execução
pagespeedfy report pagespeedfy-reports/promovaweb.com/home/20261105-103012

# 2. Pelo arquivo
pagespeedfy report pagespeedfy-reports/promovaweb.com/home/20261105-103012/
report.json

# 3. Lista os IDs reprovados no mobile
pagespeedfy --json report "$REPORT" | jq -r '.runs[] | select(.strategy=="mobile")
| .failedAudits[].id'

# 4. Salva o Markdown em outro lugar
pagespeedfy report "$REPORT" > auditoria-home.md

# 5. Última execução de uma rota
pagespeedfy report "$(ls -d pagespeedfy-reports/promovaweb.com/blog/* | tail -1)"
```

`pagespeedfy compare <antes> <depois>`

Compara duas execuções da mesma página.

Argumento	Tipo	Obrigatório	Efeito
`<antes>`	caminho	sim	Pasta ou `report.json` da execução anterior
`<depois>`	caminho	sim	Pasta ou `report.json` da execução posterior

Saída: por estratégia comum aos dois relatórios, tabela de notas e de métricas de laboratório com a diferença e três listas: auditorias resolvidas, novas e mantidas. Com `--json`: `{ before, after, sameUrl, strategies: [{ strategy, scores, labMetrics, resolvedAudits, newAudits, remainingAudits }] }`.

Efeito persistente: nenhum.

Recusas: qualquer dos caminhos inválido (código 1).

Limites: URLs diferentes geram aviso, não erro, para permitir comparar produção com preview. Estratégias presentes em só um relatório ficam de fora.

Exemplos:

```
# 1. Antes e depois de uma correção
pagespeedfy compare pagespeedfy-reports/promovaweb.com/home/20261105-103012 \
  pagespeedfy-reports/promovaweb.com/home/20261106-091544

# 2. Produção contra preview
pagespeedfy compare "$REPORT_PRODUCAO" "$REPORT_PREVIEW"

# 3. Só a diferença de Performance no mobile
pagespeedfy --json compare "$A" "$B" | jq '.strategies[] |
select(.strategy=="mobile") | .scores.performance'

# 4. Falha num script de CI quando surgem auditorias novas
test "$(pagespeedfy --json compare "$A" "$B" | jq '[.strategies[].newAudits[]] |
length')" -eq 0

# 5. As duas últimas execuções de uma rota
set -- $(ls -d pagespeedfy-reports/promovaweb.com/home/* | tail -2)
pagespeedfy compare "$1" "$2"
```

Solução de problemas

`Nenhuma API key configurada`

Nenhuma chave no arquivo nem na variável. Rode `pagespeedfy auth login` ou defina `PAGESPEEDFY_API_KEY`.

`A API key foi recusada`

A API devolveu HTTP 403. Confira, no Google Cloud Console:

- se a PageSpeed Insights API está ativada no projeto da chave;
- se as restrições da chave incluem essa API;
- se a chave não foi apagada ou trocada.

Depois, grave de novo com `pagespeedfy auth login --validar`.

`A cota diária da API acabou`

A cota do projeto do Google Cloud ligado à chave em uso chegou ao limite do dia. O CLI não repete, porque a resposta não muda até a renovação. Peça mais cota no Google Cloud Console, use a chave de outro projeto ou espere o dia seguinte no fuso do Pacífico.

`A API recusou a consulta (HTTP 400)`

O Google não conseguiu carregar a página. Causas comuns:

- a URL exige login ou está atrás de autenticação básica;
- a página responde 404, 500 ou redireciona em laço;
- o firewall ou a proteção contra bots bloqueia o Lighthouse.

Abra a URL numa janela anônima. Se ela não carregar ali, o Google também não carrega.

`O PageSpeed Insights não alcança localhost`

O CLI recusa `localhost`, IPs privados e `.local`. Publique um preview e meça a URL pública.

`a API não respondeu em 180 s`

Páginas pesadas podem passar do tempo padrão. Aumente com `--timeout 300` ou meça uma estratégia por vez com `-s mobile`.

`runtimeError` preenchido no relatório

O Lighthouse rodou, mas falhou ao medir. A mensagem fica no topo da seção da estratégia no `report.md`. Exemplos: `NO_FCP` (a página não pintou nada) e `ERRORED_DOCUMENT_REQUEST`. Corrija o carregamento da página e meça de novo.

Notas muito diferentes entre duas medições

Variação de até 5 pontos em Performance é comum. Rode duas ou três medições e compare as métricas. Mudança grande sem alteração no código costuma vir de terceiros (anúncios, chat, analytics) ou de cache frio na CDN.

O relatório não mostra todos os itens

O `report.json` guarda até 50 itens por auditoria (`--max-items`). Quando `truncated` é `true`, os demais estão no `mobile.raw.json` ou `desktop.raw.json` da mesma pasta.