Headers em Webscraping

Definir corretamente os `headers` em web scraping é uma das práticas mais importantes para:

- **Evitar bloqueios**
- **Simular um navegador real**
- **Acessar conteúdo dinâmico**
- **Evitar redirecionamentos indesejados**
- **Receber a resposta correta da página (HTML esperado)**

☑ Principais `headers` utilizados em requisições HTTP

Aqui estão os principais headers que você pode configurar, com explicações e exemplos de quando usar:

◇ `User-Agent` ☑ (essencial)

- Simula um navegador real.
- Sem ele, muitos sites retornam uma página diferente ou bloqueiam.

```
'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)...'
```

◇ `Accept` ☑

- Indica os tipos de resposta que o cliente aceita.
- Padrão: aceitar qualquer tipo (*/*)

```
'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'
```

◇ Accept-Language ☑

- Idioma preferido. Pode influenciar o conteúdo da página (ex: Walmart em inglês ou português).

```
'Accept-Language': 'en-US,en;q=0.9,pt;q=0.8'
```

◇ Accept-Encoding

- Tipos de compressão aceitos. Usado para economizar dados.

```
'Accept-Encoding': 'gzip, deflate, br'
```

◇ Connection

- Indica se a conexão deve ser mantida aberta.

```
'Connection': 'keep-alive'
```

◇ Referer (ou Referrer) ◇

- Indica a página anterior. Útil para sites que esperam que você venha de uma busca, por exemplo.

```
'Referer': 'https://www.google.com/'
```

◇ Origin

- Semelhante ao `Referer`, mas mais específico para chamadas POST e CORS.

```
'Origin': 'https://www.walmart.com'
```

◇ Host

- Indica o domínio da requisição (em geral, não é necessário definir manualmente).

```
'Host': 'www.walmart.com'
```

◇ Upgrade-Insecure-Requests

- Indica se o cliente aceita redirecionamento de HTTP para HTTPS.

```
'Upgrade-Insecure-Requests': '1'
```

◇ `Content-Type` 🟢 (em POSTs)

- Especifica o tipo de dados enviados em uma requisição `POST`.

```
'Content-Type': 'application/json'
```

◇ `Cookie` 🍪

- Usado para simular sessões autenticadas ou comportamento de usuários reais.
- Pode ser essencial para acessar conteúdos que exigem login.

```
'Cookie': 'sessionid=abc123; other_cookie=value;'
```

🧪 Como saber quais headers usar?

1. Ferramentas de inspeção do navegador

- No Chrome: `F12 > Network > clicar em uma requisição > Headers`
- Copie os headers da aba **Request Headers**

2. Análise por tentativa e erro

- Comece com os headers mais comuns (`User-Agent`, `Accept`, `Accept-Language`)
- Se a resposta for diferente do esperado ou vazia, adicione `Referer`, `Origin`, `Cookie`, etc.

3. Automação com bibliotecas

- Você pode usar bibliotecas como `fake_useragent` para gerar User-Agents reais aleatórios.

```
from fake_useragent import UserAgent
HEADERS = {'User-Agent': UserAgent().random}
```

📋 Exemplo completo e profissional de headers

```
HEADERS = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)...',
    'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;
q=0.8',
    'Accept-Encoding': 'gzip, deflate, br',
    'Accept-Language': 'en-US,en;q=0.9',
    'Connection': 'keep-alive',
    'Referer': 'https://www.google.com/',
    'Upgrade-Insecure-Requests': '1',
}
```

⚠️ Dica final

- **Menos é mais**: adicione apenas os headers que você **realmente precisa**. Headers desnecessários podem levantar suspeitas em sistemas anti-bot.
- Use `time.sleep()` ou `random.uniform()` entre requisições para parecer mais humano.
- Em páginas muito dinâmicas (JS pesado), considere usar **Selenium** ou **Playwright** com navegação real.