# Criação de um Banco de Dados com SQLite após Web Scraping de Livros

Allyson de Almeida Sirvano

20 de maio de 2025

## 1 Introdução

Este trabalho tem como objetivo apresentar o processo completo de extração de dados do site *Books to Scrape* de forma automatizada utilizando a biblioteca Playwright. Os dados obtidos são armazenados em um banco de dados local SQLite, o que permite análises posteriores por meio de bibliotecas gráficas como Matplotlib e Seaborn.

## 2 Bibliotecas Utilizadas

### 2.1 Playwright

Responsável por controlar o navegador de forma programática, o Playwright permite navegar entre páginas, capturar elementos e simular interações humanas como cliques e rolagens.

Listing 1: Importação do Playwright

```
1 from playwright.sync_api import sync_playwright
```

O método `sync_playwright()` inicia uma instância sincronizada do navegador, permitindo o uso das funções do Playwright em modo não assíncrono, facilitando o uso em scripts simples.

### 2.2 SQLite3

Utilizado para a criação de um banco de dados leve, local e sem dependências externas. `SQLite3` permite a persistência dos dados extraídos.

Listing 2: Importação do SQLite

```
1 import sqlite3
```

A biblioteca `sqlite3` provê uma interface para interagir com bancos de dados SQLite por meio de comandos SQL embutidos em Python.

### 2.3 Matplotlib e Seaborn

Estas bibliotecas são usadas para visualização de dados. O Matplotlib oferece maior controle sobre os gráficos, enquanto o Seaborn simplifica a criação de visualizações estatísticas.

Listing 3: Importação para visualização

```
import matplotlib.pyplot as plt
import seaborn as sns
```

## 3 Processo de Extração de Dados

A navegação é feita de forma automatizada, capturando os links de todos os livros presentes na página inicial. Posteriormente, cada link é acessado individualmente para coletar informações detalhadas.

Listing 4: Extração dos links dos livros

```
livros_links = page.eval_on_selector_all(
    '.product_pod h3 a',
    '(elements) => elements.map(element => ({ title: element.
       getAttribute("title"), href: element.href }))'
)
```

A função `eval_on_selector_all` executa um código JavaScript dentro do contexto da página, coletando todos os elementos que correspondem ao seletor CSS informado. Neste caso, obtém-se os títulos e URLs dos livros exibidos na página.

## 4 Tratamento dos Dados

Os dados obtidos são transformados para os tipos corretos (números, texto) e validados. Registros com informações incorretas ou ausentes são descartados para garantir a integridade do banco.

Listing 5: Validação dos dados

```
if not titulo or preco <= 0 or quantidade < 0 or avaliacao not in
   range(1, 6):
    print(f"Dados inv\'alidos encontrados: {livro}")
    continue
```

A condição verifica se o título é vazio, se o preço é menor ou igual a zero, se a quantidade é negativa ou se a avaliação não está entre 1 e 5 estrelas. Dados que falharem essas validações são ignorados.

## 5 Criação do Banco de Dados

Duas tabelas principais foram criadas: uma para categorias e outra para livros. Cada livro está vinculado à sua categoria por meio de uma chave estrangeira.

Listing 6: Criação das tabelas

```
CREATE TABLE IF NOT EXISTS categorias (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL UNIQUE
);
CREATE TABLE IF NOT EXISTS livros (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    titulo TEXT NOT NULL UNIQUE,
    preco REAL NOT NULL,
    quantidade INTEGER NOT NULL,
    avaliacao INTEGER NOT NULL,
    categoria_id INTEGER,
    FOREIGN KEY (categoria_id) REFERENCES categorias(id)
);
```

`CREATE TABLE IF NOT EXISTS` cria as tabelas apenas se elas ainda não existirem. Campos como `AUTOINCREMENT` garantem que o ID seja gerado automaticamente.

# 6 Inserção dos Dados no Banco

A inserção é feita evitando duplicidades. Cada categoria é verificada previamente e, se ainda não existir, adicionada ao banco. Em seguida, os livros são inseridos com referência à categoria correspondente.

Listing 7: Inserção de livros com categoria associada

```
INSERT INTO livros (titulo, preco, quantidade, avaliacao,
   categoria_id)
VALUES (?, ?, ?, ?, ?)
```

A sintaxe com `?` permite uso de parâmetros seguros com o SQLite, evitando injeção de SQL.

# 7 Reorganização das Categorias

Para manter os IDs das categorias organizados, foi criada uma tabela temporária com nova sequência. Os registros da tabela de livros foram atualizados para refletir os novos IDs.

Listing 8: Atualização dos IDs das categorias

```
UPDATE livros
SET categoria_id = (
    SELECT id FROM categorias_temp
    WHERE categorias_temp.nome = (
        SELECT nome FROM categorias
        WHERE categorias.id = livros.categoria_id
    )
)
```

Este comando substitui os IDs antigos pelos novos baseando-se no nome da categoria. Utiliza subqueries aninhadas para buscar a equivalência entre as tabelas.

# 8 Indicadores de Performance

Foram geradas consultas para identificar padrões relevantes, como livros com alta avaliação, estoques baixos e preços médios.

Listing 9: Consultas para indicadores

```
SELECT COUNT(*) FROM livros WHERE avaliacao >= 4
SELECT COUNT(*) FROM livros WHERE quantidade <= 5
SELECT AVG(preco) FROM livros WHERE avaliacao >= 4
```

Cada comando SQL realiza uma análise específica: contagem de livros bem avaliados, livros com estoque baixo e média de preços dos livros de boa avaliação.

# 9 Visualização de Dados

Os dados foram visualizados por meio de gráficos de barras e pizza. O gráfico de barras exibe a quantidade de livros por nota de avaliação.

Listing 10: Gráfico de barras com Seaborn

```
sns.barplot(
    x=distribuicao.keys(),
    y=distribuicao.values(),
    palette="muted",
    ax=axes[0]
)
```

A função `barplot()` do Seaborn recebe as chaves (notas) e valores (quantidades), e renderiza um gráfico de barras com paleta de cores especificada.

O gráfico de pizza apresenta a proporção percentual de livros em cada faixa de avaliação.

Listing 11: Gráfico de pizza com Matplotlib

```
axes[1].pie(
    percentual,
    labels=[f'{p} estrelas ({v:.1f}%)' for p, v in zip(distribuicao
        .keys(), percentual)],
    startangle=140,
    colors=sns.color_palette("pastel", len(distribuicao))
)
```

O método `pie()` do Matplotlib gera um gráfico circular usando os percentuais e rótulos personalizados. O parâmetro `startangle` gira o início do gráfico e as cores são definidas com a paleta "pastel"do Seaborn.