Tech challenge: API para Consulta de Livros

Produzido: Augusto Santos Bettin

RM: rm367357 Linkedin: <u>link</u>

Construindo a Fundação: O Início do Sistema de Recomendação



A empresa no início do desenvolvimento de um projeto de recomendação de livros e necessita acessar as informações de seus livros para gerar o modelo

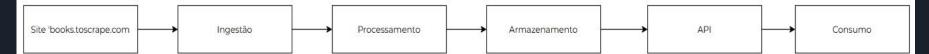


Para o primeiro passo para o desenvolvimento será necessário gerar um fluxo de dados do zero



Desenvolver um pipeline completo para extrair e transformar dados, disponibilizando-os através de uma API, com foco em escalabilidade e reuso futuro.

Arquitetura atual



Mecanismo: Os scripts populate_db, py e a rota /scrape_books em api/routes_scape, py utilizam as bibliotecas requests para baixar o conteúdo HTML das páginas de livros e BeautifulSoup para extrair as informacões relevantes.

Processo: O scraper navega por 50 páginas do catálogo, coletando os links individuais de cada livro. Mecanismo: A biblioteca pandas é utilizada dentro da função scrape_books para estruturar os dados extraídos.

Processo: Os dados brutos (texto)

passam por uma limpeza:

- Os preços são convertidos para o formato numérico (float). - As avaliações em estrelas (ex: "Three") são convertidas para números (ex: 3). - A disponibilidade em estoque é extraída como um número inteiro. - Os dados transformados são salvos temporariamente em um arquivo .csv.

Mecanismo: A aplicação utiliza Flask-SQLAlchemy como ORM para interagir com os bancos de dados.

Ambiente Local: O script populate_db.py insere os dados processados em um banco de dados SQLite (instance/livros.db).

Ambiente de Produção (Nuvem): O script migrate_db. py é responsável por migrar os dados do SQLite local para um banco de dados PostgreSQL hospedado na nuvem (Render), garantindo que a API em produção tenha acesso aos dados.

Mecanismo: Uma API RESTful construída com Flask.

em api/routes.py, expõe endpoints para: - Autenticação de usuários /register, /login.

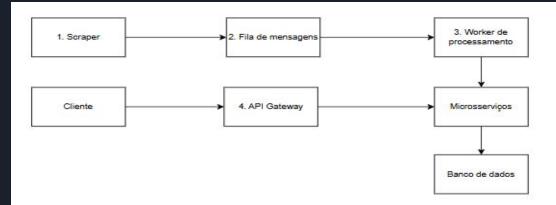
Funcionalidades: A API, definida

- Consultas de dados: /books, /books/<id>, /books/search).

 Agregações e estatísticas: /stats/overview, /stats/categories). Mecanismo: Qualquer cliente HTTP (navegador, aplicação móvel, scripts de análise de dados) pode consumir a API.

Processo: O cliente primeiro se autentica para obter um token JWT e, em seguida, utiliza esse token no cabeçalho Authorization para acessar as rotas protegidas e consumir os dados dos livros.

Proposta de arquitetura futura



 Scraper agendado: O processo de scraping é desacoplado da aplicação principal e se toma um serviço independente que roda em intervalos agendados.

Beneficio: Evita

sobrecarregar a API com tarefas pesadas e permite a coleta continua de dados sem intervenção manual. Fila de Mensagens: Após o scraping, em vez de processar os dados imediatamente, o Scraper envia os links ou dados brutos para uma fila de mensagens.

Beneficio: Cria uma arquitetura assincrona e resiliente. Se o serviço de processamento falhar, as mensagens permanecem na fila para serem processadas mais tarde, evitando a perda de dados. 3. Workers de

Processamento: Um ou mais serviços "workers" consomem as mensagens da fila, realizam o processamento e a limpeza dos dados, e finalmente os inserem no banco de dados.

Beneficio: A carga de processamento pode ser escalada horizontalmente de forma independente, adicionando mais workers conforme a necessidade. 4. API Gateway: Serve como um ponto de entrada único para todas as requisições dos clientes. Ele é responsável por rotear as chamadas para o microsserviço apropriado (ex: serviço de livros, serviço de autenticação).

Beneficio: Centraliza a gestão de autenticação, rate limiting e logging, simplificando os microsserviços.

Cenário de uso

Extração de Dados:

- O cientista de dados utiliza um script em Python com a biblioteca requests para interagir com a API.
- Primeiro, ele se autentica na rota /login para obter um token de acesso.
- Em seguida, ele faz uma chamada para a rota /books para obter o dataset completo de todos os livros, convertendo a resposta JSON em um DataFrame do pandas.

Insights iniciais (Pré - EDA):

- Com os dados em um DataFrame, o cientista explora as informações. Ele utiliza as rotas /stats/overview e /stats/categories para obter um resumo rápido sobre a distribuição de preços, avaliações e a contagem de livros por categoria.
- Ele analisa a correlação entre o preço e a avaliação, a popularidade das categorias e a distribuição das notas (estrelas).

Treinamento do Modelo:

- Com base na análise, ele decide criar um modelo de recomendação baseado em conteúdo.
- Ele utiliza técnicas de Processamento de Linguagem Natural (PLN) para vetorizar os títulos dos
- livros e combiná-los com outras features, como categoria e preço.

 Ele treina um modelo que, ao receber um livro como entrada, calcula a similaridade com os outros livros e retorna os mais parecidos como recomendação.

Plano de Integração com Modelos de ML

Criação de um Microsserviço de ML:

- Um novo serviço (ex: ML-Recomendacao-Service) é criado, também em Flask ou FastAPI, e containerizado.
- Este serviço carrega o modelo de recomendação treinado e expõe um endpoint interno (ex: /predict) que recebe o ID de um livro e retorna uma lista de IDs de livros recomendados.

Novo Endpoint na API de Livros:

• Uma nova rota, como GET /books/<int:book_id>/recommendations, é adicionada ao serviço de livros.

Orquestração da Chamada:

Quando um cliente chama GET /books/123/recommendations: a. O API Gateway encaminha a requisição para o Serviço de Livros. b. O Serviço de Livros faz uma chamada interna para o Serviço de ML, enviando as informações do livro de ID 123. c. O Serviço de ML processa a requisição, usa o modelo para gerar as recomendações e retorna uma lista de IDs (ex: [45, 210, 503]). d. O Serviço de Livros usa essa lista de IDs para buscar os dados completos dos livros recomendados no banco de dados. e. Finalmente, o Serviço de Livros retorna a lista completa de objetos de livros recomendados para o cliente.

Pipeline de Retreinamento:

- Um pipeline automatizado de MLOps (GitHub Actions) é criado.
- Este pipeline é executado periodicamente (ex: semanalmente) para:
 - o Buscar os dados mais recentes do banco de dados.
 - Retreinar o modelo de recomendação.
 - Avaliar a performance do novo modelo e, se for superior ao antigo, implantá-lo automaticamente no Serviço de ML sem interromper o serviço.