Tech challenge: API para Consulta de Livros

Produzido: Augusto Santos Bettin
RM: rm367357
Linkedin: link

Construindo a Fundação: O Início do Sistema de Recomendação

Contexto

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

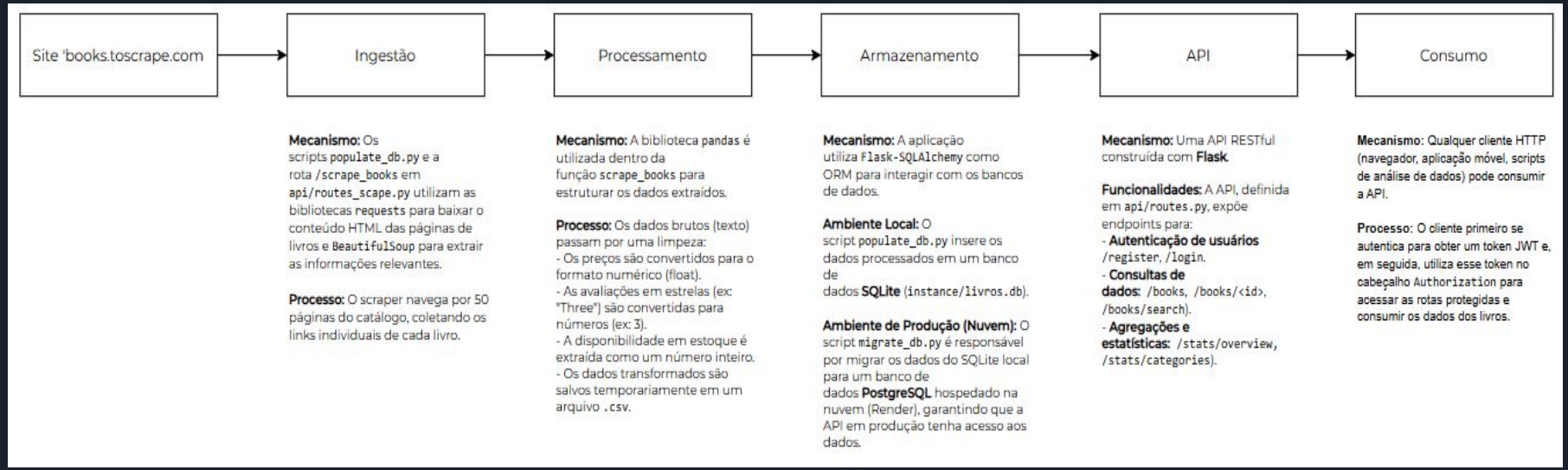
Desafio

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

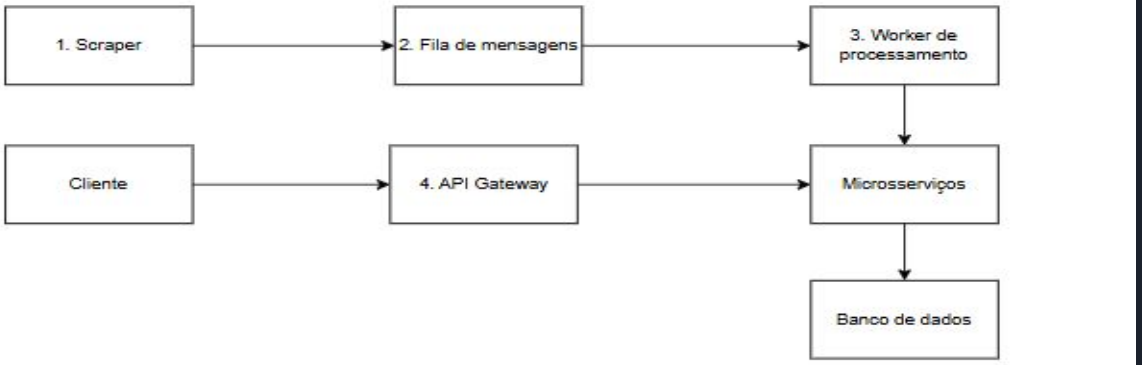
Solução

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

Arquitetura atual



Proposta de arquitetura futura



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

Benefício: Evita sobrecarregar a API com tarefas pesadas e permite a coleta contínua de dados sem intervenção manual.

2. 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.

Benefício: Cria uma arquitetura assíncrona 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.

Benefício: 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).

Benefício: 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:
  - 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.