Documentação de Arquitetura da API de Vitivinicultura Embrapa

Apresentado por Rodrigo Viannini e Leonardo Pena

Arthur Tavari | rm356537

Elyas Carvalho | rm357117

Jean Alves | rm357143

Leticia Pires | rm357721

Osman Lira | rm357904

SUMÁRIO

1. INTRODUÇÃO

Este documento descreve a arquitetura da API desenvolvida em Python que interage com os dados disponibilizados pelo site da Embrapa nas seguintes abas: Produção, Processamento, Comercialização, Importação e Exportação. O objetivo principal desta API é extrair e transformar os dados provenientes dessas fontes para

gerar artefatos JSON, os quais serão posteriormente analisados utilizando um modelo de Machine Learning.

2. CONTEXTO

A Embrapa (Empresa Brasileira de Pesquisa Agropecuária) é uma referência em pesquisa agrícola no Brasil e disponibiliza uma vasta quantidade de dados essenciais para estudos e análises no setor agropecuário. Para aproveitar esses dados de forma eficiente, a API foi projetada para automatizar a coleta, processamento e transformação dos mesmos.

3. OBJETIVOS

Os objetivos principais desta API são:

1. Consultar dados nas abas de Produção, Processamento, Comercialização, Importação e Exportação do site da Embrapa.
2. Gerar artefatos JSON estruturados e limpos para análise guardados em uma estrutura S3.
3. Garantir a segurança dos dados e das transações utilizando autenticação baseada em JWT (JSON Web Token) com a biblioteca `pyjwt`.

4. PROJETO

- Endereço código Fonte
  https://github.com/osmanlirajr/fiap_machine_learning/tree/main
- Gravação de explicação do projeto

- ML https://drive.google.com/drive/folders/1B2kzwFWS3uP8RxxejitWAAVqZZib9v-S?usp=drive_link
- Projeto código
- **https://1drv.ms/v/s!AhRCVDnjMV47iaJKv43PleM7cM1Rsg?e=AVX9gq**

5. VISÃO GERAL DA ARQUITETURA DA API

A API descrita no repositório fiap_machine_learning é projetada para consultar dados do site da Embrapa nas abas de Produção, Processamento, Comercialização, Importação e Exportação, transformando esses dados em artefatos JSON para análise com modelos de Machine Learning. A arquitetura da API é composta por vários componentes essenciais que juntos garantem sua funcionalidade, segurança e escalabilidade.

4.1. COMPONENTES PRINCIPAIS

- **FastAPI:**
  - **Descrição:** Framework web utilizado para criar a API. É conhecido por seu alto desempenho e por suportar a criação de aplicações com rotas bem definidas e documentação automática.
  - **Função:** Gerenciar as rotas, middlewares e toda a lógica de negócio da aplicação.
- **JWT Autenticação (pyjwt):**
  - **Descrição:** Biblioteca utilizada para implementar autenticação baseada em JSON Web Tokens (JWT).
  - **Função:** Garantir que apenas usuários autenticados possam acessar e manipular os dados da API.
- **Consultas Web (Web Scraping):**
  - **Descrição:** Métodos e scripts para realizar web scraping das páginas da Embrapa.

    - **Função:** Extrair dados das abas de Produção, Processamento, Comercialização, Importação e Exportação do site da Embrapa.
- **Pandas:**
    - **Descrição:** Biblioteca Python para manipulação e análise de dados.
    - **Função:** Processar, limpar e transformar os dados extraídos em estruturas tabulares (DataFrames), facilitando a manipulação e análise dos dados antes de serem convertidos para JSON.
- **JSON:**
    - **Descrição:** Formato de dados utilizado para intercâmbio de informações.
    - **Função:** Converter os dados processados e estruturados em DataFrames para o formato JSON, que será utilizado na análise por modelos de Machine Learning e na resposta da API.

6. ESTRUTURA DO PROJETO

- ***main.py***: Arquivo principal onde a aplicação FastAPI é configurada e iniciada.
- ***modelos/***: Contém os modelos de dados utilizados pela API.
- ***services/:*** Contém Lógica de negócios e funções de serviço.
- ***auth/:*** Contém as funções que provê a segurança da API com JWT.

7. DEPLOY DA APLICAÇÃO

6.1 AWS CLOUD

O diagrama mostra que a aplicação está sendo implantada na nuvem AWS, utilizando diversos serviços da AWS para garantir escalabilidade, disponibilidade e desempenho.

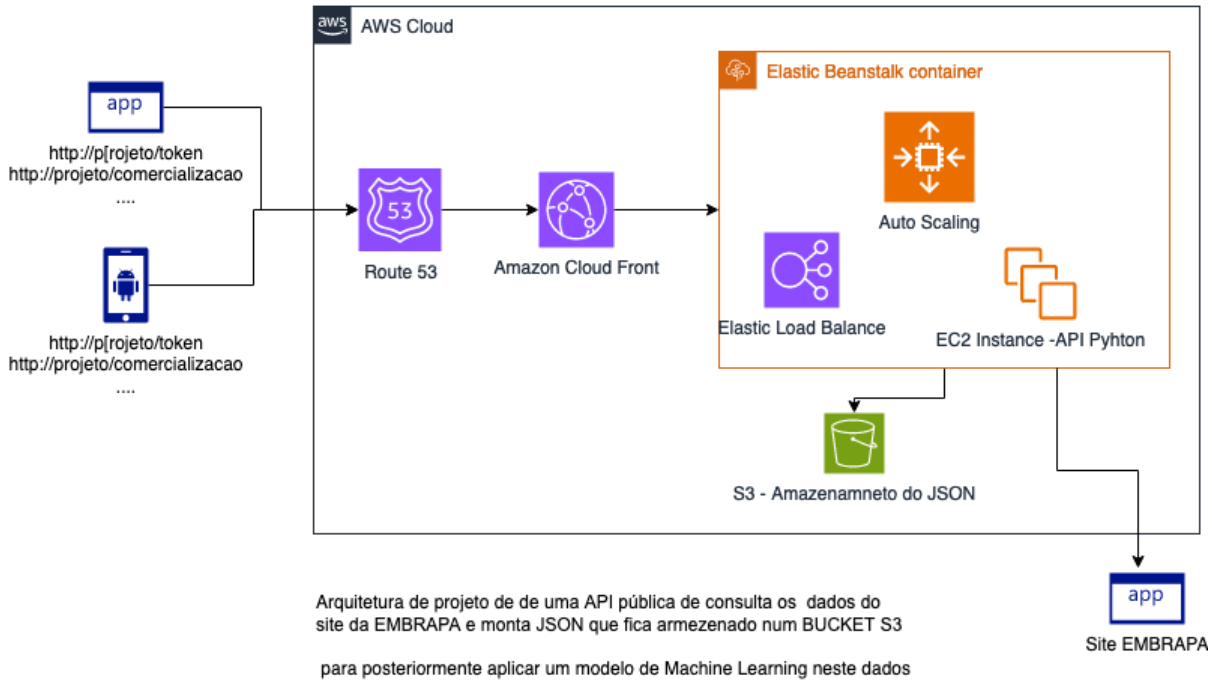

Figura 1: Arquitetura de Deploy.

- **Route 53**:
    - **DNS Management**: O Amazon Route 53 é usado para gerenciar o DNS da aplicação, direcionando o tráfego dos domínios da web para os serviços apropriados dentro da infraestrutura da AWS.
- **Amazon CloudFront**:
    - **Content Delivery Network (CDN)**: O Amazon CloudFront é utilizado para distribuir o conteúdo da aplicação, como arquivos estáticos, globalmente com baixa latência. Ele melhora a experiência do usuário ao entregar conteúdo através de uma rede global de servidores de borda.
- **Elastic Beanstalk**:
    - **PaaS para Deploy**: O AWS Elastic Beanstalk gerencia a implantação da aplicação, fornecendo um contêiner que suporta auto escalonamento e balanceamento de carga.

- **Auto Scaling**: O Auto Scaling é configurado para garantir que a aplicação pode aumentar ou diminuir a quantidade de instâncias EC2 conforme a demanda, mantendo a performance e otimização de custos.
- **Elastic Load Balancer (ELB)**: O ELB distribui automaticamente o tráfego de entrada entre várias instâncias EC2, aumentando a tolerância a falhas da aplicação.
- **EC2 Instances**: A aplicação em si é executada em instâncias EC2 dentro do contêiner do Elastic Beanstalk. O diagrama especifica que estas instâncias estão rodando uma API Python.
- **Amazon S3**:
    - **Armazenamento de Dados**: O Amazon S3 é utilizado para armazenar arquivos JSON, que podem ser utilizados pela API ou outros componentes da aplicação para armazenar e recuperar dados de forma escalável e durável.

6.2. FLUXO DE DADOS

1. **Usuários e Aplicativos**:
    - Os usuários e aplicativos móveis enviam requisições para o domínio gerenciado pelo Route 53.
2. **Entrega de Conteúdo**:
    - O CloudFront serve conteúdo estático diretamente dos servidores de borda mais próximos dos usuários, reduzindo a latência e melhorando a performance.
3. **Roteamento e Processamento**:
    - O ELB recebe o tráfego das requisições dinâmicas e distribui para as instâncias EC2 que estão executando a API Python dentro do contêiner do Elastic Beanstalk.
    - O Auto Scaling ajusta o número de instâncias conforme a necessidade.
4. **Armazenamento de Dados**:

- A aplicação interage com o Amazon S3 para armazenar e recuperar dados JSON conforme necessário.

Esta arquitetura oferece uma solução robusta para uma aplicação escalável, distribuída globalmente e com alta disponibilidade, utilizando os serviços da AWS para gerenciamento eficiente e otimização de recursos.

6.3 IMPLANTAÇÃO NA AWS

6.3.1.Configuração do Route 53

1. **Registrar Domínio**: Se ainda não tiver um domínio, registre um através do Route 53 ou transfira um existente.
2. **Configurar Zona Hospedada**: Crie uma zona hospedada no Route 53 para gerenciar o DNS do seu domínio.
3. **Configurar Registros DNS**: Adicione registros DNS (A, CNAME, etc.) que apontem para o CloudFront e outros recursos.

6.3.2. Configuração do Amazon S3

1. **Criar Bucket S3**: Crie um bucket no S3 para armazenar os arquivos JSON.
2. **Configurar Políticas de Acesso**: Defina políticas de acesso adequadas para garantir que apenas sua aplicação tenha acesso ao bucket.
3. **Fazer Upload dos Arquivos**: Faça upload dos arquivos JSON necessários para o bucket.

6.3.3. Configuração do Elastic Beanstalk

1. **Criar Aplicação Elastic Beanstalk**:

- Acesse o console do Elastic Beanstalk e crie uma nova aplicação.
- Escolha a plataforma Python para a sua aplicação.

2. **Configurar Ambiente Elastic Beanstalk**:
   - Crie um novo ambiente dentro da aplicação Elastic Beanstalk.
   - Selecione o ambiente de Web Server.
   - Configure as instâncias EC2 e outros recursos necessários (tamanho da instância, chave SSH, etc.).

6.3.4. Fazer Deploy da Aplicação:

1. **Acessar o Console Elastic Beanstalk**:
   - Acesse a Console do Elastic Beanstalk na AWS.
2. **Criar uma Nova Aplicação**:
   - Clique em "Create Application".
   - Preencha o nome da aplicação e, opcionalmente, uma descrição.
3. **Configurar a Plataforma**:
   - Selecione a plataforma "Python".
   - Escolha a versão apropriada do Python conforme necessário.
4. **Fazer Upload do Código da Aplicação**:
   - Prepare seu código para deployment. Crie um arquivo ZIP contendo seu código fonte da aplicação, incluindo o `requirements.txt`.
   - Na seção "Application code", selecione "Upload your code".
   - Clique em "Choose file" e faça o upload do arquivo ZIP da sua aplicação (`my-app.zip`).
5. **Configurar o Ambiente**:
   - Escolha um nome para o ambiente.
   - Escolha um domínio para o ambiente (opcional).
   - Configure a capacidade do ambiente conforme necessário, ajustando as configurações de instâncias EC2, Auto Scaling, e Load Balancer.
6. **Configurar Opções Adicionais (opcional)**:
   - Você pode ajustar configurações adicionais como variáveis de ambiente, opções de banco de dados, monitoramento e logs, se necessário.

8. MODELO DE MACHINE LEARNING NA AWS

A criação de uma API para coletar dados diretamente da página de Uvas e Vinhos da Embrapa pode trazer muitos benefícios, especialmente para a modelagem de Machine Learning. Essa abordagem proporciona uma base sólida e atualizada de informações para o desenvolvimento de modelos de machine learning, garantindo que as previsões e análises estejam sempre alinhadas com as condições atuais do mercado e da produção vinícola.

A utilização de uma API de web scraping permite a coleta contínua e automatizada de dados atualizados da página de Uvas e Vinhos da Embrapa. Isso elimina a necessidade de intervenção manual, economizando tempo e recursos. Além disso, a consistência e a integridade dos dados coletados automaticamente são significativamente melhoradas, reduzindo a possibilidade de erros e garantindo a qualidade dos dados utilizados nos modelos de machine learning.

Além disso, pode-se integrar isso com o Amazon SageMaker, dessa forma, pode-se aproveitar a capacidade de treinamento escalável, a implantação fácil e o gerenciamento de ciclo de vida de modelos que a plataforma oferece. Isso resulta em uma solução completa e eficiente para a criação e gestão de modelos de machine learning na AWS.

7.1 EXEMPLOS DE APLICAÇÕES COM OS ENDPOINTS

- ***Importação:***
    - Classificar os países em diferentes categorias (principais importadores, importadores moderados, etc.). **Features**: quantidade, ano, valor. **Algoritmo**: k-means.

    - Prever a quantidade de vinhos de mesa, espumantes e uvas frescas que serão importados em anos futuros. **Features**: ano, país, quantidade, valor. **Algoritmo**: Regressão linear.
- ***Produção:***
    - Prever a quantidade de produção de vinhos, sucos e derivados para anos futuros. **Features**: produto, ano. **Target**: quantidade. **Algoritmo**: Regressão Linear.
    - Identificar tendências na produção ao longo dos anos para diferentes produtos. **Features:** produto, ano. **Target:** quantidade. **Algoritmo:** Análise de séries temporais (ARIMA).
- ***Processamento:***
    - Prever a quantidade de uvas processadas para diferentes categorias e cultivares nos anos futuros. **Features:** ano, categoria (viníferas, americanas e híbridas, uvas de mesa, sem classificação), cor. **Target:** quantidade. **Algoritmo:** Regressão linear (necessário converter a tabela com coluna de categoria).

7.2 UTILIZAÇÃO DO AMAZON SAGEMAKER

O Amazon SageMaker é uma plataforma completa para construção, treinamento e implantação de modelos de machine learning na AWS. Ele oferece várias vantagens que facilitam o processo de desenvolvimento e gerenciamento de modelos:

- Ambiente Integrado: SageMaker oferece notebooks Jupyter integrados para desenvolvimento e experimentação, permitindo que os cientistas de dados escrevam e testem código de forma eficiente.
- Treinamento Escalável: A plataforma permite o treinamento de modelos em larga escala, utilizando instâncias de alta performance e distribuindo a carga de trabalho conforme necessário.

- Implantação Fácil: SageMaker facilita a implantação de modelos em produção com endpoints escaláveis e gerenciados, permitindo inferências em tempo real ou em batch.
- Gerenciamento de Ciclo de Vida: Ferramentas integradas para monitoramento, tuning de hiperparâmetros e gerenciamento de versões de modelos garantem que todo o ciclo de vida dos modelos de machine learning seja bem controlado.
- Integração com Outros Serviços AWS: SageMaker se integra perfeitamente com outros serviços AWS, como S3 para armazenamento de dados, Lambda para automação e API Gateway para criação de APIs.

7.3 ARQUITETURA DE ML NO SAGEMAKER

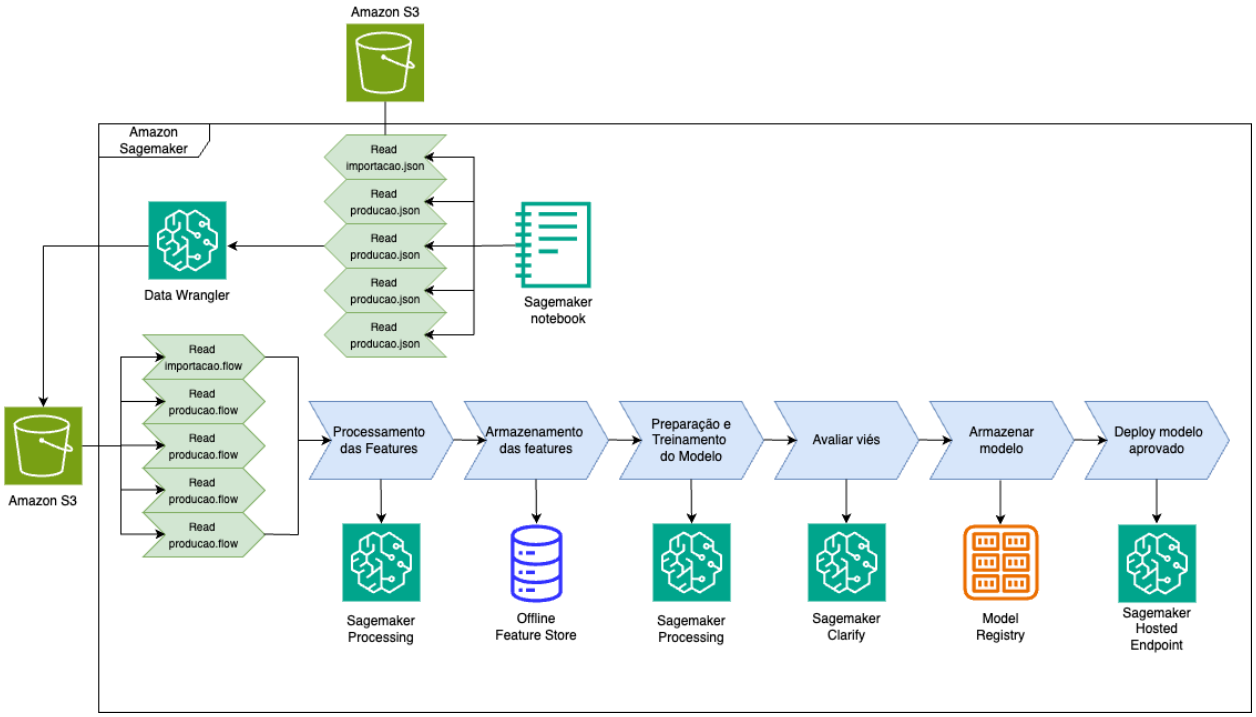

Figura 2: Arquitetura de Machine Learning no Sagemaker.

1. **SageMaker Notebook:** O fluxo começa com o SageMaker Notebook, onde os dados são lidos do Amazon S3. Os arquivos da API são carregados no notebook para exploração e análise inicial de dados.

2. **Data Wrangler:** Os arquivos são lidos para realizar a preparação e transformação dos dados para o processamento.
3. **Processamento das Features:** Utilização do SageMaker Processing para realizar o processamento de features. As features são transformadas e preparadas para o armazenamento e treinamento do modelo.
4. **Armazenamento das Features:** As features processadas são armazenadas na Offline Feature Store, e permite acesso rápido e eficiente às features processadas para diferentes tarefas de machine learning.
5. **Treinamento do modelo:** Sagemaker Training é usado para treinar o modelo e fazer ajustes de hiperparâmetros.
6. **Avaliação de viés:** Sagemaker Clarify é utilizado para mitigar vieses nos dados e no modelo. Ele garante que o modelo seja justo e equitativo, identificando potenciais fontes de discriminação.
7. **Deploy do modelo aprovado:** o modelo aprovado é implantado em um endpoint hospedado no Sagemaker. Ele disponibiliza o modelo para inferências em tempo real ou em batch, permitindo sua utilização em aplicações de produção.

7.4 EXEMPLO DE MODELAGEM

Nesta seção, será apresentado um exemplo prático e simples de como realizar a modelagem de dados utilizando um dataset de importação de vinhos. O objetivo é demonstrar as etapas necessárias para preparar os dados, realizar a análise descritiva, dividir os dados em conjuntos de treino e teste, treinar um modelo de regressão linear e avaliar os resultados obtidos.

É importante ressaltar que este exemplo é bastante simplificado. Em cenários reais de machine learning, frequentemente são necessários ajustes nas features, refinamentos nos dados ou até mesmo a utilização de abordagens e algoritmos diferentes para alcançar melhores resultados.

7.4.1. Carregamento dos dados

Inicialmente, os dados são carregados a partir de um arquivo JSON contendo informações sobre a importação de vinhos por país, quantidade, tipo de importação e ano. Utiliza-se a biblioteca ***pandas*** para manipulação e análise dos dados. A seguir, são visualizadas as primeiras linhas do dataframe carregado:

```
import pandas as pd

df = pd.read_json('/content/importacao_ordenado.json')
df.head()
```

	pais	quantidade	tipo_importacao	ano
0	Afeganistão	0.0	VINHOS DE MESA	1970
1	Afeganistão	0.0	VINHOS DE MESA	1971
2	Afeganistão	0.0	VINHOS DE MESA	1972
3	Afeganistão	0.0	VINHOS DE MESA	1973
4	Afeganistão	0.0	VINHOS DE MESA	1974

7.4.2. Análise descritiva

É realizada uma análise descritiva dos dados para entender melhor as suas características. Essa etapa inclui a verificação de estatísticas básicas, como média, desvio padrão, valores mínimos e máximos para cada variável.

```
print(data.describe())
```

7.4.3. Preparação dos dados

Para preparar os dados para a modelagem, utiliza-se a técnica de codificação ***one-hot encoding*** nas variáveis categóricas, como o país e o tipo de importação. Em seguida, são separadas as features (X) da variável alvo (y), que neste caso é a quantidade importada.

```
data = pd.get_dummies(data, columns=['pais', 'tipo_importacao'])
X = data.drop(columns=['quantidade'])
y = data['quantidade']
```

7.4.4. Divisão dos dados

Os dados são divididos em conjuntos de treino e teste para avaliar a performance do modelo. Utiliza-se 80% dos dados para o treino e 20% para o teste.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

7.4.5. Treinamento do modelo

Treina-se um modelo de Regressão Linear utilizando o conjunto de treino.

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

7.4.6. Previsões e avaliação

Com o modelo treinado, são realizadas previsões no conjunto de teste e a performance do modelo é avaliada utilizando o erro absoluto médio (MAE - Mean Absolute Error).

```python
from sklearn.metrics import mean_absolute_error

# Previsões
y_pred = model.predict(X_test)

# Avaliação do Modelo
mae = mean_absolute_error(y_test, y_pred)
print(f'Mean Absolute Error: {mae}')
```