**Tech Challenge – Fase 1 | Machine Learning Engineering**

---

Tech Challenge é o projeto da fase que englobará os conhecimentos obtidos em todas as disciplinas da fase. Esta é uma atividade que, a princípio, deve ser desenvolvida em grupo. Importante atentar-se ao prazo de entrega, pois trata-se de uma atividade obrigatória, uma vez que sua pontuação se refere a 60% da nota final.

## O problema

Você foi contratado(a) para criar **uma API pública** capaz de consultar, em tempo real, os dados de vitivinicultura disponibilizados pela Embrapa (Produção, Processamento, Comercialização, Importação e Exportação). Não é necessário armazenar, tratar ou modelar as informações neste momento.

---

## Entregáveis obrigatórios

1. **Repositório GitHub**
   • Código organizado em módulos, com README detalhado (descrição do projeto, instruções de instalação, execução e testes).

2. **Link público do deploy da API**
   • Heroku, Vercel, Render, Fly.io ou qualquer plataforma similar — desde que permita o acesso para testes externos.

3. **Diagrama/Imagem da Arquitetura**
   • Fluxograma de alto nível que mostre: raspagem → API → (fallback para arquivo local, se aplicável).

4. **Vídeo de apresentação (3 – 12 min)**
   • Demonstre a arquitetura, execute chamadas reais à API e comente boas práticas utilizadas.

---

## Objetivo técnico

- **Construir a API** em Python usando Flask, FastAPI, Django REST ou framework equivalente.

- Trazer os dados diretamente do site da Embrapa. (SUGESTÃO: Implementar **web-scraping** (BeautifulSoup, Selenium, Scrapy, Playwright ou similar))
- Criar **rotas** que retornem as tabelas raspadas em formato JSON.
- **Persistência de dados não é obrigatória**; entretanto, por instabilidade ocasional do site, recomenda-se:
    1. Tentar a raspagem ao vivo.
    2. Caso falhe, servir um arquivo CSV/JSON local previamente baixado.
- Atenção: o endpoint deve tentar o site primeiro, para que possamos validar a raspagem durante a correção.

---

## Liberdade de escolha & boas práticas

- **Frameworks e bibliotecas**: escolha livre desde que atendam aos requisitos (API, scraping e deploy).
- **Deploy**: qualquer nuvem/plataforma que forneça URL pública.
- **Versionamento & CI/CD**: use Git, mantenha commits claros; pipelines de teste são bem-vindos.
- **Documentação**: Swagger e README descrevendo todas as rotas.
- **Autenticação**: opcional, mas altamente valorizada (ex.: JWT).
- **Código limpo**: separação de responsabilidades, tratamento de exceções, logs adequados.

## Opcionais (extras bem-vistos)

• **Autenticação avançada** (JWT, OAuth2, refresh tokens).

• **Persistência em banco de dados** (SQL ou NoSQL) para cache ou histórico.

• **Containerização com Docker** e arquivo `docker-compose.yml`.

• **Pipeline CI/CD** automatizando testes e deploy.

• **Frontend leve** (Streamlit, React, Next.js) consumindo a API.

• **Dashboard interativo** ou notebooks exploratórios dos dados.

• **Protótipo de modelo de ML** usando a API como fonte.

• **Observabilidade**: logs estruturados, métricas e monitoramento.

> Estes itens **não são obrigatórios**, mas agregam valor, demonstram profundidade técnica e podem destacar seu projeto.

---

> **Importante**: este projeto deve equilibrar desafio e acessibilidade. Para quem já tem experiência, é uma chance de refinar boas práticas; para quem está começando, concentre-se no essencial: raspagem + endpoints funcionais + deploy.

---

Lembre-se de que você poderá apresentar o desenvolvimento do seu projeto chamando diretamente um dos professores(as). Essa é uma boa oportunidade para discutir sobre as dificuldades encontradas e pegar dicas valiosas com docentes especialistas e colegas de turma.

Vamos pra cima!