Desafio Técnico — Estagiário Python/Django :: 2026.1

Funcionalidades obrigatórias

1. Cadastro de Alunos

- Nome
- E-mail
- CPF
- Data de ingresso

2. Cadastro de Cursos

- Nome
- Carga horária
- Valor da inscrição
- Status (ativo/inativo)

3. Matrículas

- Matricular aluno em um curso
- Registrar data
- Status: pago / pendente

4. Financeiro

- Listar matrículas com status
- Total devido por aluno
- Total pago pelo aluno

5. Relatórios (HTML)

a) Histórico do aluno

- Cursos matriculados
- Status
- Totais

b) Dashboard geral

- Total de alunos
- Cursos ativos
- Matrículas pagas vs pendentes

PS.: Não precisa necessariamente ser um dashboard, mas somente listagens.

---

6. API — Endpoints com Django Rest Framework (OBRIGATÓRIO)

Criar endpoints para:

Alunos

- Listar / Criar / Atualizar / Remover

Cursos

- Listar / Criar / Atualizar / Remover

Matrículas

- Criar matrícula
- Listar matrículas de um aluno
- Marcar como paga

Relatórios via API

Pelo menos 1 relatório via JSON, por exemplo:

- total de matrículas por curso
- total devido por aluno
- total de pagamentos pendentes

---

7. Avaliação SQL (OBRIGATÓRIO)

Arquivo `meu_database.sql`

Crie o SQL com:

- `CREATE TABLE`
- Tipos adequados
- PK / FK

Não precisa replicar a migration do Django, mas deve representar o modelo.

Consulta usando SQL bruto

Crie ao menos **uma rota ou endpoint API** usando `cursor()` ou `Model.objects.raw()` contendo:

- JOIN
- Aggregation (SUM, COUNT, GROUP BY, etc.)

Pode ser exibida em:

- página HTML
- endpoint JSON

---

8. Docker (OBRIGATÓRIO)

Você deve entregar a aplicação rodando via Docker, incluindo:

`Dockerfile` — build da aplicação

`docker-compose.yml` — orquestração

Com no mínimo:

- Serviço `web` (Django)
- Serviço `db` (SQLite opcional / recomendado: PostgreSQL)

Scripts de inicialização

No `docker-compose`, a aplicação deve subir executando:

1. Instalação de dependências
2. `python manage.py migrate`
3. `python manage.py runserver`

Como rodar (no README):

```
docker-compose up --build
```

Após isso, o avaliador deve acessar:

- http://localhost:8000 (frontend)
- http://localhost:8000/api/... (endpoints DRF)

Regras gerais

- Usar Python 3 + Django + Django Rest Framework
- Usar banco relacional (SQLite ou Postgres)
- Criar **alguns templates HTML, mas também pode utilizar o ambiente Django Admin.**
- Criar APIs com DRF
- Entregar tudo via GitHub
- Na raiz incluir:
  - `README.md`

- `meu_database.sql`
- `Dockerfile`
- `docker-compose.yml`

Considerações

Não se prenda tanto a design. Foque nas funcionalidades requeridas, versionamento do código e no funcionamento adequado da aplicação, estas serão avaliadas conforme nossos critérios do seletivo.

Além disso, seu código **não precisa estar 100% completo**.

Queremos avaliar:

- clareza
- organização
- entendimento dos conceitos

Sobre o envio do desafio

- Realize o envio do repositório do GitHub para o e-mail **joao.lucas@tecnotech.org** com assunto “DESAFIO TÉCNICO - VAGA ESTÁGIO PYTHON 2026.1” até o dia 06/12/2025 às 23:59.
- Obrigatório que o readme do repositório contenha as instruções para execução do projeto.