

# 2º Etapa - Desafio Técnico

Vaga: Desenvolvedor Backend

### 1. Objetivo

Desenvolver uma API RESTful que simule parte do backend de uma plataforma centralizada, focando em:

- Ingestão e gerenciamento de documentos
- Autenticação de usuários
- Registro de buscas simuladas via IA
- Persistência em banco de dados relacional

# 2. Tecnologias Obrigatórias

- Node.js com Express
- PostgreSQL com Prisma ORM
- Autenticação via JWT
- Upload de arquivos com Multer ou similiar
- Docker (Dockerfile + docker-compose.yml)
- Documentação com Swagger UI

### 3. Diferenciais

- Fluxograma visual explicativo incluído no README
- Endpoint /me para retorno de informações do usuário autenticado
- Interface simples (web ou Swagger) para consumo dos endpoints
- Testes automatizados (Jest ou Vitest)
- Integração opcional com ferramenta externa de IA Text Generation API (ex: Hugging Face Inference API)

#### 4. Tarefas

#### 4.1 Autenticação e Controle de Acesso

- Endpoints: /auth/register, /auth/login
- Geração e validação de JWT
- Middleware de proteção de rotas
- Endpoint /me para retorno do usuário autenticado



#### 4.2 Upload e Ingestão de Dados

- Endpoint: /datasets/upload
- Arquivos aceitos: .csv e .pdf
- Armazenamento local ou em base64
- Registro de metadados: nome, tamanho, data, usuário
- Tabela datasets para origem dos arquivos
- Tabela records com conteúdo em campo JSON

#### 4.3 Consulta e Listagem

- GET /datasets listar datasets do usuário
- GET /datasets/:id/records listar registros do dataset
- GET /records/search?query=... busca textual por palavra-chave no JSON

#### 4.4 Registro de Buscas com Mock IA

- Endpoint: POST /queries
- Corpo: pergunta + datasetId
- Simulação de resposta com base em palavras-chave ou texto aleatório

```
function mockIAResponse(pergunta) {
  if (pergunta.includes("contrato")) {
    return "Este documento trata de cláusulas contratuais.";
  }
  return "A IA identificou informações relevantes.";
}
```

Registro em tabela queries: pergunta, resposta, usuário, data

#### 4.5 Histórico de Consultas

• GET /queries — listar perguntas e respostas anteriores

## 5. Modelagem de Dados Sugerida

- users(id, nome, email, senha\_hash)
- datasets(id, nome, usuario\_id, criado\_em)
- records(id, dataset\_id, dados\_json, criado\_em)
- queries(id, usuario\_id, pergunta, resposta, criado\_em)



# 6. Entrega

Entrega via repositório público no GitHub contendo:

- Código-fonte completo e funcional
- Dockerfile e docker-compose.yml
- README com instruções de execução
- Documentação Swagger
- (Opcional) Interface web ou frontend simples
- (Opcional) Fluxograma visual incluído no README

# 7. Avaliação

Critério	Avaliação
Organização do código	Estrutura e boas práticas
Aderência ao escopo	Implementação correta dos requisitos
Segurança	JWT e proteção das rotas
Documentação	Swagger + README claro
Diferenciais	Testes, interface visual, integração com IA