

# Desafio Prático: Desenvolvedor(a) Full-Stack

## Objetivo:

Bem-vindo(a) ao nosso desafio técnico! O objetivo deste teste é avaliar sua habilidade em desenvolver uma solução *full-stack* completa, desde a modelagem do banco de dados até a interface do usuário.

Queremos observar não apenas seu conhecimento técnico nas ferramentas listadas, mas também sua capacidade de estruturar um projeto, aplicar boas práticas de desenvolvimento, escrever código limpo e testável.

## O Desafio: API e Interface de um "Mini-Kanban"

Você deverá construir uma aplicação *web* simples para gerenciamento de tarefas no estilo Kanban. A aplicação deve ser composta por um **Back-end (API RESTful)** e um **Front-end (SPA)**.

## Contexto de Domínio (DDD)

A aplicação gira em torno de três entidades principais:

1. **Quadro (Board)**: Um contêiner de alto nível para um projeto ou contexto.
  - *Atributos*: `id`, `name` (ex: "Project Alpha", "Week Tasks").
2. **Coluna (Column)**: Representa uma etapa do fluxo de trabalho e pertence a um **Quadro**.
  - *Atributos*: `id`, `name` (ex: "To Do", "In Progress", "Done"), `board_id`.
  - *Regra de Negócio*: A ordem das colunas dentro de um quadro é importante.
3. **Cartão (Task/Card)**: Representa a tarefa em si e pertence a uma **Coluna**.
  - *Atributos*: `id`, `title`, `description` (opcional), `column_id`.
  - *Regra de Negócio*: Um cartão deve sempre pertencer a uma coluna.

O **Agregado** principal aqui seria o **Quadro**, que "controla" o ciclo de vida de suas **Colunas** e, indiretamente, dos **Cartões** contidos nelas.

## Requisitos Funcionais (RF)

### 1. Back-end (API)

O back-end deve expor *endpoints* RESTful para gerenciar as entidades:

- **Quadros (Boards)**:
  - `POST /boards`: Cria um novo quadro (ex: `{ "name" : "My Project" }`).

- `GET /boards`: Lista todos os quadros.
  - `GET /boards/{id}`: Retorna um quadro específico com **todas as suas colunas e cartões aninhados**. (Importante para o front-end).
- **Colunas (Columns)**:
  - `POST /boards/{id}/columns`: Cria uma nova coluna dentro de um quadro (ex: `{ "name": "To Do" }`).
  - (*Opcional/Simplificação*): A gestão de colunas (CRUD) pode ser simplificada, talvez criando-as junto com o quadro. O foco é nos cartões.
- **Cartões (Tasks/Cards)**:
  - `POST /columns/{id}/cards`: Cria um novo cartão em uma coluna (ex: `{ "title": "Develop API" }`).
  - `PUT /cards/{id}`: Atualiza um cartão (ex: mudar título ou descrição).
  - `DELETE /cards/{id}`: Exclui um cartão.
- **Ação de Domínio Principal**:
  - `PATCH /cards/{id}/move`: Endpoint específico para mover um cartão para uma nova coluna.
    - *Request Body*: `{ "newColumnId": "uuid-of-destination-column" }`
    - Este endpoint deve conter a lógica de domínio para validar a movimentação.

## 2. Front-end (React)

A interface deve consumir a API criada e permitir ao usuário:

- Listar os quadros existentes e selecionar um.
- Ao selecionar um quadro, visualizar todas as suas colunas e os cartões dentro delas (layout Kanban).
- Adicionar novos cartões a qualquer coluna.
- Excluir cartões.
- **(Requisito Chave)** Mover cartões entre colunas.
  - *Idealmente*: Com funcionalidade de *drag-and-drop* (arrastar e soltar).
  - *Alternativa simples*: Um botão no cartão que abre um menu/modal para selecionar a coluna de destino.

## Requisitos Técnicos (RT)

1. **Back-end**: Utilize **uma** das seguintes linguagens:
  - Node.js (Express, NestJS, etc.)
  - Python (Django, Flask, FastAPI, etc.)
  - Java (Spring Boot, etc.)
2. **Front-end**: Utilize **React**.
3. **Banco de Dados**: Utilize **um** dos seguintes:
  - PostgreSQL ou MySQL (Relacional)
  - MongoDB (NoSQL)
4. **Testes**:

- O back-end **deve** conter testes unitários e/ou de integração, especialmente para os *endpoints* e a lógica de negócio (como mover um cartão).
- Testes no front-end são um bônus.

#### 5. Controle de Versão e Entrega:

- O projeto final deve ser hospedado em um repositório Git (preferencialmente GitHub).
- O repositório **deve estar público** para que os recrutadores possam avaliar o código.
- O link do repositório é o principal item de entrega e deve ser compartilhado com os recrutadores.
- O histórico de *commits* (com mensagens claras) também será avaliado.

#### 6. Documentação:

- Um **README .md** é obrigatório, explicando:
  - Como configurar o ambiente (variáveis de ambiente, etc.).
  - Como instalar as dependências (back-end e front-end).
  - Como rodar o projeto localmente.
  - Como rodar os testes.

## Critérios de Avaliação

Não esperamos uma aplicação 100% pronta para produção, mas sim que a estrutura e a lógica demonstrem seu conhecimento. Focaremos em:

- **Arquitetura e Design (SOLID & DDD):**
  - O código do back-end está bem estruturado? (Ex: separação em camadas como Controllers, Services, Repositories).
  - Os princípios SOLID são aplicados? (Especialmente o 'S' de Responsabilidade Única).
  - Como a lógica de domínio (regras de negócio) foi implementada? Está separada da lógica de infraestrutura (framework web, acesso a banco)?
- **Qualidade do Código:**
  - Clareza, legibilidade e manutenibilidade.
  - Uso correto de convenções da linguagem escolhida.
- **Testes:**
  - A qualidade e a cobertura dos testes automatizados no back-end.
- **Banco de Dados:**
  - A modelagem do banco de dados (relacional ou documental) está correta e eficiente para o problema?
- **Front-end (React):**
  - Estrutura dos componentes, gerenciamento de estado (pode ser estado local, Context API, etc.) e comunicação com a API.
- **Boas Práticas Gerais:**
  - Uso do Git (histórico de *commits*).
  - Qualidade do **README .md**.
  - Tratamento de erros (na API e no front-end).

## Entrega: Vídeo Explicativo + Repositório no GitHub

Para entendermos melhor suas decisões e seu processo de raciocínio, gostaríamos que você gravasse um vídeo curto (10-20 minutos) apresentando sua solução.

- **O que incluir:**
  - Uma rápida demonstração da aplicação funcionando (criando um quadro, colunas e movendo cartões).
  - Uma explicação da arquitetura escolhida (pastas do projeto, divisão de responsabilidades no back-end).
  - Destacar uma parte do código que você considera interessante ou um desafio que superou.
- **Como entregar:**
  - Suba o vídeo para uma plataforma (YouTube - não listado, Google Drive, Loom, etc.) e adicione o link no [README .md](#) do seu repositório.

**Boa sorte!**