

# Desafio Técnico: Plataforma Multi-tenant Simplificada

## Desafio Técnico: Plataforma Multi-tenant Simplificada

**Posição:** Pessoa Desenvolvedora Fullstack (Laravel + Angular) – Júnior

### Visão Geral

Neste desafio, você vai implementar uma versão simplificada da **Plataforma de Colaboração Multi-tenant**, focando apenas em **3 entidades** e avaliando sua capacidade de:

- Modelar relacionamentos básicos no Laravel (BelongsTo, HasMany)
- Criar endpoints RESTful que exponham esses relacionamentos
- Consumir e renderizar dados relacionados no Angular

Não há autenticação nem pacotes de multi-tenant: tudo rodará no schema `public` do PostgreSQL, e o isolamento entre “tenants” será feito manualmente via campo `tenant_id`.

## Objetivos de Avaliação

### 1. Migrations & Eloquent

- Criação de tabelas e uso correto de BelongsTo/HasMany

### 2. API RESTful

- Endpoints que retornem dados com relacionamentos aninhados

### 3. Frontend Angular

- Consumo dos endpoints e exibição de dados relacionados em componentes

### 4. Docker Compose

- Orquestração simples de Laravel + PostgreSQL

## Modelo de Dados (schema `public`)

```
-- 1) Tabela tenants: representa as organizações (tenants) no sistema
CREATE TABLE IF NOT EXISTS public.tenants (
  id SERIAL PRIMARY KEY,           -- Identificador único (chave primária)
  :contentReference[oaicite:0]{index=0}
  name TEXT NOT NULL,              -- Nome da empresa/organização
```

```

    created_at TIMESTAMP DEFAULT now() -- Data de criação (padrão: timestamp atual)
);

-- 2) Tabela workspaces: espaços de trabalho associados a um tenant
CREATE TABLE IF NOT EXISTS public.workspaces (
    id SERIAL PRIMARY KEY,                -- Identificador único do workspace
    tenant_id INTEGER NOT NULL             -- Referência ao tenant (chave estrangeira)
:contentReference[oaicite:1]{index=1}
    REFERENCES public.tenants(id)
    ON DELETE CASCADE,                   -- Se o tenant for removido, apaga workspaces relacionados
    name TEXT NOT NULL,                  -- Nome do workspace
    created_at TIMESTAMP DEFAULT now()    -- Data de criação
);

-- 3) Tabela projects: projetos dentro de um workspace
CREATE TABLE IF NOT EXISTS public.projects (
    id SERIAL PRIMARY KEY,                -- Identificador único do projeto
    workspace_id INTEGER NOT NULL          -- Referência ao workspace (chave estrangeira)
:contentReference[oaicite:2]{index=2}
    REFERENCES public.workspaces(id)
    ON DELETE CASCADE,                   -- Se o workspace for removido, apaga projetos relacionados
    title TEXT NOT NULL,                  -- Título do projeto
    description TEXT,                     -- Descrição opcional do projeto
    created_at TIMESTAMP DEFAULT now()    -- Data de criação
);

```

- **Tenants *hasMany* Workspaces**
- **Workspace *belongsTo* Tenant**
- **Workspace *hasMany* Projects**
- **Project *belongsTo* Workspace**

## Tarefas

### 1. Preparar o Repositório

- Crie um repositório público (GitHub/GitLab).
- Adicione um `docker-compose.yml` com serviços:
  - `app` : Laravel (PHP-FPM, Composer)
  - `db` : PostgreSQL (schema `public` )

### 2. Backend (Laravel)

#### 1. Migrations & Models

- Crie migrations conforme o modelo acima.
- Defina os Models com `tenant()`, `workspaces()`, `workspace()` e `projects()`, usando `BelongsTo` e `HasMany`.

## 2. Seeder Inicial

- Insira ao menos 2 tenants, cada um com 1–2 workspaces e 2–3 projects.

## 3. Controllers & Rotas

- **GET** `/api/tenants` → lista todos os tenants.
- **GET** `/api/tenants/{id}/workspaces` → lista workspaces de um tenant.
- **GET** `/api/workspaces/{id}/projects` → lista projects de um workspace.
- **POST** `/api/workspaces` → cria novo workspace (informar `tenant_id`, `name`).
- **POST** `/api/projects` → cria novo project (informar `workspace_id`, `title`, `description`).

## 4. Scope de Tenant

- Apesar de não haver login, implemente um **Global Scope** no Model de Workspace que filtre por um `tenant_id` fixo (ex.: `1`) para demonstrar capacidade de usar scopes.

# 3. Frontend (Angular)

## 1. Estrutura do Projeto

- Inicialize com `ng new`.

## 2. Services

- `TenantService`: busca `/api/tenants`.
- `WorkspaceService`: busca `/api/tenants/{tenantId}/workspaces`.
- `ProjectService`: busca `/api/workspaces/{workspaceId}/projects`.

## 3. Componentes

- **TenantListComponent**: exibe lista de tenants; ao clicar, carrega workspaces.
- **WorkspaceListComponent**: exibe workspaces selecionados; ao clicar, carrega projects.
- **ProjectListComponent**: exibe projetos do workspace corrente.

## 4. Templates & Relacionamentos

- Em cada lista, exiba também o nome do “pai” (ex.: em Workspaces, mostre o `tenant.name`; em Projects, mostre `workspace.name`).

- Use *nested property binding* (e.g. `workspace.tenant.name` ) para demonstrar consumo dos relacionamentos aninhados que a API retornará.


## 4. Documentação

No `README.md` , inclua:

- Como rodar `docker-compose up` e inicializar Laravel + Postgres.
- Comandos para `php artisan migrate --seed` .
- Endpoints disponíveis e exemplos de resposta JSON.
- Instruções para iniciar e usar a aplicação Angular.

## Critérios de Avaliação

Critério	Peso
<b>Correção das migrations e relacionamentos</b>	25%
<b>Implementação dos Models &amp; Scopes</b>	20%
<b>Qualidade e clareza dos Endpoints</b>	20%
<b>Consumo correto no Angular (nested data)</b>	20%
<b>Estrutura do Docker Compose e README</b>	15%

 **Boa sorte!** Esse desafio permite que você demonstre domínio de relacionamentos básicos no Laravel, criação de APIs e consumo de dados aninhados em Angular, com configuração mínima de containers.