Desafio Técnico – Desenvolvedor Full Stack

Objetivo: Avaliar raciocínio, boas práticas, estruturação do código e autonomia.

1. Contexto do Desafio

Pretende-se desenvolver uma **aplicação simples de gestão de tarefas com RBAC** (Role-Based Access Control) — ex: "Workplace Tasks" — com backend em .NET, base de dados PostgreSQL e frontend em Angular. O foco é a organização do código, clareza e capacidade de raciocínio técnico, não o design.

A aplicação deverá implementar controle de acesso baseado em 3 roles: **admin**, **manager** e **member**. As permissões de cada role estão definidas na secção "RBAC — Roles e Permissões".

2. Backend (.NET / C#)

Requisitos:

- Criar uma API REST com os endpoints: GET /tasks, POST /tasks, PUT /tasks/{id}, DELETE /tasks/{id}.
- Cada tarefa deve ter: id (UUID ou auto-increment), title (string), description (string), status (enum: "Pending", "InProgress", "Done"), createdAt, updatedAt (timestamps).
- Armazenar dados em PostgreSQL, usando Entity Framework Core.
- Implementar validação de dados e tratamento de erros estruturado (retornar códigos HTTP adequados).
- Implementar autenticação (por exemplo JWT) e autorização baseada em roles (RBAC).

RBAC — Roles e Permissões (Backend)

admin

- Acesso total a todos os endpoints e recursos.
- Pode criar, ler, atualizar e eliminar qualquer tarefa.
- Pode gerir usuários e atribuir roles.

manager

- Pode criar tarefas (`POST /tasks`).
- Pode ler todas as tarefas (`GET /tasks`).
- Pode atualizar **qualquer** tarefa (`PUT /tasks/{id}`).

- Pode eliminar apenas tarefas que criou (`DELETE /tasks/{id}`)

member

- Pode criar tarefas (`POST /tasks`).
- Pode ler tarefas (`GET /tasks`).
- Pode atualizar **apenas** tarefas que criou (ou apenas o `status` das tarefas atribuídas a si, se modelar atribuição de owner/assignee).
- Pode eliminar apenas tarefas que criou (`DELETE /tasks/{id}`)

Bónus:

- Implementar paginação e filtros por status.
- Utilizar injeção de dependências corretamente.
- Adicionar Scalar ou Swagger UI para documentação.

3. Frontend (Angular)

Requisitos:

- Criar um ecrã simples com: lista de tarefas (tabela ou cards), formulário para criar tarefa e botões para editar e eliminar tarefas.
- Conectar-se à API criada.
- Usar boas práticas de estrutura de projeto Angular (services, components, models, environments).

RBAC — Roles e Permissões (Backend)

admin

- Verá botões de editar e eliminar em todas as tarefas.
- Terá acesso a páginas/controles de administração (se implementados).

manager

- Verá botões de editar em todas as tarefas (ou conforme restrição definida no backend).
- Poderá eliminar tarefas se for o criador.

member

- Verá botão de editar apenas nas tarefas que criou (ou apenas para alterar `status` das tarefas atribuídas a si).
- Poderá eliminar tarefas se for o criador.

Bónus:

- Usar Reactive Forms.
- Implementar loading indicators e tratamento de erros.
- Usar Angular Material (opcional).

4. Entrega

O candidato deve entregar o código em um repositório Git público (GitHub, GitLab, etc.) e incluir um README.md com:

- Passos para executar o backend e frontend
- Decisões técnicas
- Pontos de melhoria, se houvesse mais tempo
- **(Importante)** Instruções para testar RBAC: contas de teste com roles (ex.: admin@example.com / senha), endpoints para criação de tokens ou instruções para gerar JWTs de teste.

5. O que valorizamos

Critério	Peso	Descrição
Estrutura e clareza do código	25%	Organização, legibilidade, consistência
Boas práticas (REST, EF Core), e uso de princípios como DRY e KISS	20%	Qualidade técnica e padrões de projeto
Funcionamento correto e completude	20%	Se todos os requisitos funcionam corretamente
Capacidade de documentação / README	10%	Clareza das instruções e explicações
Tratamento de erros e validações	10%	Robustez e atenção a detalhes
Qualidade e integração do Frontend (Angular)	15%	Estrutura, usabilidade e ligação com API

6. Etapa Complementar – Entrevista Técnica (30 min)

Durante a revisão:
• Pedir que explique uma decisão técnica do backend e outra do frontend.
• Perguntar o que melhoraria se tivesse mais tempo.
• Pedir para simular uma pequena alteração (ex: adicionar campo prioridade) e observar raciocínio.