Desafio Técnico Full Stack - PonteTech (VERSÃO SIMPLIFICADA - 7 DIAS)

🎯 Objetivo

Desenvolver um **sistema de gerenciamento de tarefas** simples, focando em **qualidade** ao invés de quantidade.

Prazo: 7 dias corridos **Foco**: Backend sólido, Frontend funcional, Código limpo

---

🚀 O Que Você Vai Construir (SIMPLIFICADO)

Funcionalidades OBRIGATÓRIAS (Núcleo do sistema)

1. Autenticação Simples

Cadastro

- Nome (obrigatório, mínimo 3 caracteres)
- Email (obrigatório, único, formato válido)
- Senha (obrigatória, mínimo 8 caracteres)

Login

- Email e senha
- Retorna token JWT (válido por 7 dias)

Sem níveis de permissão complexos - todos os usuários têm acesso igual por enquanto.

---

2. Gerenciamento de Tarefas (CORE)

Atributos da Tarefa

- Título (obrigatório, 3-100 caracteres)
- Descrição (obrigatório, 10-500 caracteres)
- Status: Pendente, Em Progresso, Concluída (padrão: Pendente)

- Prioridade: Baixa, Média, Alta (padrão: Média)
- Criador (usuário logado)
- Data de criação (automática)

Validações

- Título não pode ser vazio
- Status segue fluxo: Pendente → Em Progresso → Concluída
- Status Concluída não pode voltar

Operações

- ✅ Criar tarefa
- ✅ Listar todas as tarefas do usuário
- ✅ Ver detalhes de uma tarefa
- ✅ Atualizar status da tarefa
- ✅ Deletar tarefa (apenas criador)

---

3. Funcionalidades Básicas

Dashboard Simples

- Contador de tarefas por status
- Lista das minhas tarefas

Filtros Básicos

- Filtrar por status
- Filtrar por prioridade
- Ordenar por data de criação

Funcionalidades OPCIONAIS (Se sobrar tempo)

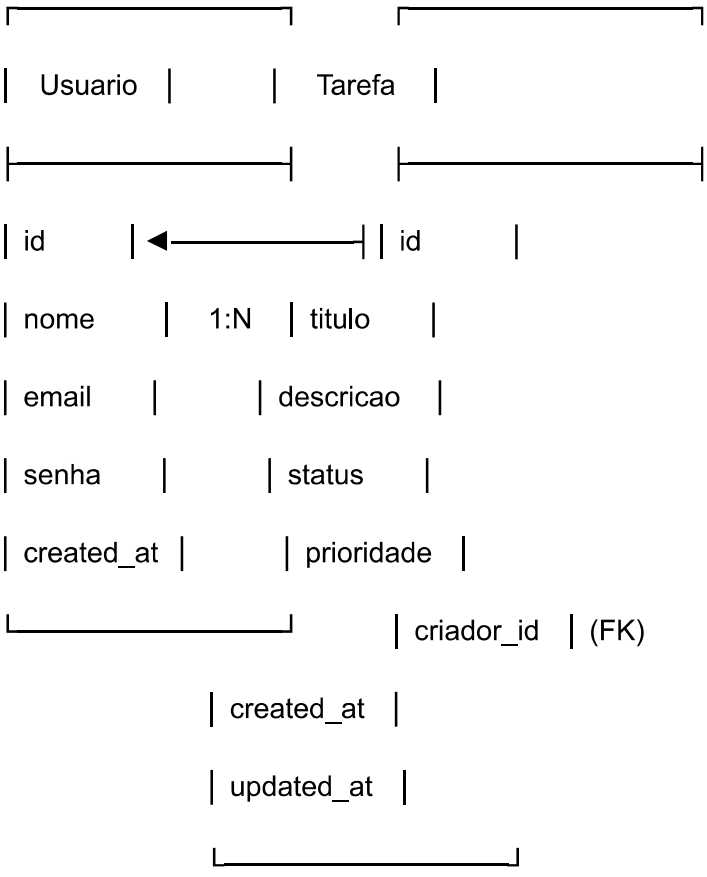
Escolha **no máximo 2** destas para implementar:

- ☐ Busca por título
- ☐ Atribuir tarefa a outro usuário
- ☐ Prazo/deadline para tarefas
- ☐ Comentários simples em tarefas

- [ ] Marcar tarefa como favorita
- [ ] Paginação nas listagens

📊 Modelagem de Dados SIMPLIFICADA

🗄️ Banco de Dados: 2 Tabelas



1️⃣ Tabela: **usuario**

Campo	Tipo	Restrições	Validação	Exemplo
id	UUID ou Integer	PRIMARY KEY, AUTO INCREMENT	-	1

Campo	Tipo	Restrições	Validação	Exemplo
nome	VARCHAR(100)	NOT NULL	Min: 3 caracteres	"João Silva"
email	VARCHAR(255)	NOT NULL, UNIQUE	Formato email válido	"joao@email.com"
senha	VARCHAR(255)	NOT NULL	Hash bcrypt, min 8 chars na original	"$2b$10$..."
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW	Automático	"2025-01-10 10:30:00"

Índices:

- PRIMARY KEY (id)
- UNIQUE INDEX (email)

Regras de Validação:

- ✅ Nome: mínimo 3 caracteres, não pode ser vazio
- ✅ Email: formato válido (regex), único no sistema
- ✅ Senha: mínimo 8 caracteres antes do hash, deve conter ao menos 1 letra e 1 número

Seeds Recomendadas:

-- 3 usuários de teste

INSERT INTO usuario (nome, email, senha) VALUES

('Admin Teste', 'admin@teste.com', '$2b$10$hashedpassword1'),

('João Silva', 'joao@teste.com', '$2b$10$hashedpassword2'),

('Maria Santos', 'maria@teste.com', '$2b$10$hashedpassword3');

-- Senha para todos: "Teste@123"

---

2 Tabela: **tarefa**

Campo	Tipo	Restrições	Validação	Exemplo
id	UUID ou Integer	PRIMARY KEY, AUTO INCREMENT	-	1
titulo	VARCHAR(100)	NOT NULL	Min: 3, Max: 100	"Implementar login"
descricao	TEXT	NOT NULL	Min: 10, Max: 500	"Criar tela de login com JWT"
status	ENUM	NOT NULL, DEFAULT 'Pendente'	'Pendente', 'Em Progresso', 'Concluída'	"Pendente"
prioridade	ENUM	NOT NULL, DEFAULT 'Média'	'Baixa', 'Média', 'Alta'	"Alta"
criador_id	UUID ou Integer	FOREIGN KEY, NOT NULL	Deve existir em usuario	1
created_at	TIMESTAMP	NOT NULL, DEFAULT NOW	Automático	"2025-01-10 10:30:00"
updated_at	TIMESTAMP	ON UPDATE NOW	Atualiza automaticamente	"2025-01-11 15:45:00"

Relacionamentos:

- `criador_id` → `usuario.id` (Foreign Key)
- Cada tarefa pertence a UM usuário
- Um usuário pode ter MUITAS tarefas

Índices:

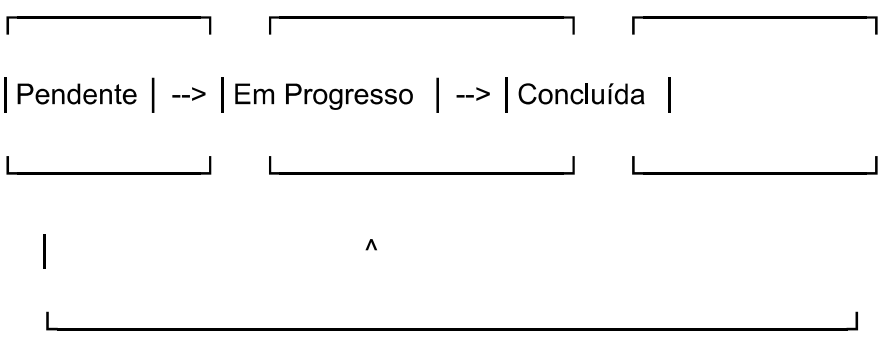
- PRIMARY KEY (id)
- INDEX (criador_id)
- INDEX (status)

- INDEX (prioridade)

Regras de Validação:

- ✅ Título: 3-100 caracteres, não pode ser vazio
- ✅ Descrição: 10-500 caracteres, não pode ser vazia
- ✅ Status: apenas valores válidos (Pendente, Em Progresso, Concluída)
- ✅ Prioridade: apenas valores válidos (Baixa, Média, Alta)
- ✅ Criador: deve existir na tabela usuario

Fluxo de Status (Importante!):



Regras:

- Pendente → pode ir para Em Progresso ou Concluída
- Em Progresso → pode ir apenas para Concluída
- Concluída → NÃO pode voltar (status final)

Seeds Recomendadas:

-- 8 tarefas de exemplo

INSERT INTO tarefa (titulo, descricao, status, prioridade, criador_id) VALUES

('Configurar projeto', 'Criar estrutura inicial do projeto', 'Concluída', 'Alta', 1),

('Implementar autenticação', 'Criar sistema de login e registro', 'Em Progresso', 'Alta', 1),

('Criar dashboard', 'Página principal com lista de tarefas', 'Pendente', 'Média', 1),

('Adicionar filtros', 'Filtrar tarefas por status e prioridade', 'Pendente', 'Baixa', 2),

('Escrever testes', 'Testes unitários do backend', 'Pendente', 'Alta', 2),

('Dockerizar aplicação', 'Criar Docker Compose', 'Pendente', 'Média', 2),

('Documentar API', 'Escrever README completo', 'Pendente', 'Alta', 3),

('Corrigir bugs', 'Revisar e corrigir problemas encontrados', 'Pendente', 'Baixa', 3);

📋 Exemplo de Queries SQL

Listar todas as tarefas de um usuário:

SELECT * FROM tarefa

WHERE criador_id = 1

ORDER BY created_at DESC;

Filtrar por status:

SELECT * FROM tarefa

WHERE criador_id = 1 AND status = 'Pendente'

ORDER BY prioridade DESC;

Contar tarefas por status (Dashboard):

SELECT status, COUNT(*) as quantidade

FROM tarefa

WHERE criador_id = 1

GROUP BY status;

Atualizar status de uma tarefa:

UPDATE tarefa

```
SET status = 'Em Progresso', updated_at = NOW()

WHERE id = 1 AND criador_id = 1;
```

Deletar tarefa (apenas do criador):

```
DELETE FROM tarefa

WHERE id = 1 AND criador_id = 1;
```

🔐 Regras de Negócio Importantes

Autenticação:

- ✅ Todo endpoint (exceto register/login) requer JWT
- ✅ JWT deve conter: id do usuário, email
- ✅ Token válido por 7 dias

Tarefas:

- ✅ Usuário só pode criar tarefas para si mesmo
- ✅ Usuário só pode ver suas próprias tarefas
- ✅ Usuário só pode editar/deletar suas próprias tarefas
- ✅ Ao editar tarefa, validar fluxo de status
- ✅ Campo updated_at deve atualizar automaticamente

Validações no Backend:

```
// Exemplo de validação (pseudocódigo)

function validarTarefa(dados) {

  if (!dados.titulo || dados.titulo.length < 3 || dados.titulo.length > 100) {

    throw new Error("Título deve ter entre 3 e 100 caracteres");

  }

  if (!dados.descricao || dados.descricao.length < 10 || dados.descricao.length > 500) {

    throw new Error("Descrição deve ter entre 10 e 500 caracteres");
```

```
  }

  const statusValidos = ['Pendente', 'Em Progresso', 'Concluída'];

  if (!statusValidos.includes(dados.status)) {

    throw new Error("Status inválido");

  }

  const prioridadeValida = ['Baixa', 'Média', 'Alta'];

  if (!prioridadeValida.includes(dados.prioridade)) {

    throw new Error("Prioridade inválida");

  }

  return true;

}
```

🛠️ Requisitos Técnicos

Backend

Stack: Livre escolha (Node.js, Python, Java, Go, etc.)

Obrigatório:

- ✅ API REST (6-8 endpoints no mínimo)
- ✅ Autenticação JWT
- ✅ Validação de dados
- ✅ Tratamento de erros
- ✅ Testes unitários (cobertura mínima 40%)
- ✅ README com instruções

Endpoints Mínimos:

POST /api/auth/register

POST /api/auth/login

GET /api/tasks

POST /api/tasks

GET /api/tasks/:id

PUT /api/tasks/:id

DELETE /api/tasks/:id

GET /api/dashboard

Frontend

Stack: Livre escolha (React, Vue, Angular, etc.)

Obrigatório:

- ✅ 4-5 páginas básicas:
  - Login
  - Cadastro
  - Dashboard/Lista de tarefas
  - Criar/Editar tarefa
- ✅ Responsivo (funciona em mobile)
- ✅ Validação de formulários
- ✅ Feedback de loading e erros

NÃO é obrigatório:

- Design sofisticado (use Bootstrap/Tailwind)
- Animações complexas
- Gerenciamento de estado complexo

Banco de Dados

Stack: Livre escolha (PostgreSQL, MySQL, SQLite, MongoDB)

Obrigatório:

- ✅ 2 tabelas (Usuario, Tarefa)

- ✅ Seeds com dados de teste

DevOps

Obrigatório:

- ✅ Docker Compose (backend + banco)
- ✅ README claro
- ✅ 1 comando para rodar

NÃO é obrigatório:

- CI/CD
- Deploy em produção

🔒 Segurança (Básica)

Obrigatório:

- ✅ Senhas hasheadas (bcrypt)
- ✅ JWT para autenticação
- ✅ Validação de inputs
- ✅ CORS básico

NÃO é obrigatório:

- Rate limiting
- Refresh tokens
- 2FA

📊 Como Você Será Avaliado (100 pontos)

Backend (40 pontos)

- API funcional: 15 pontos
- Validações: 10 pontos
- Autenticação: 10 pontos
- Testes: 5 pontos

Frontend (30 pontos)

- Funcional: 15 pontos
- Responsivo: 8 pontos
- Validações: 7 pontos

Arquitetura (15 pontos)

- Organização do código: 10 pontos
- Separação de responsabilidades: 5 pontos

Documentação (15 pontos)

- README: 8 pontos
- Setup funciona: 7 pontos

Aprovação: 60+ pontos (reduzido de 70)

⏰ Estimativa de Tempo

Tarefa	**Tempo Estimado**
Setup do projeto	1-2h
Backend (API + Auth)	8-12h
Frontend básico	8-12h
Integração	2-4h
Testes	3-5h
Docker + README	2-3h
Total	**24-38h**

Em 7 dias, isso significa **3-5 horas por dia** - muito mais realista!

💡 Estratégia Recomendada

Dia 1-2: Backend

- Setup do projeto
- Models e migrations
- Autenticação (register/login)
- CRUD de tarefas

Dia 3-4: Frontend

- Setup do projeto
- Páginas de login/registro
- Lista de tarefas
- Criar/editar tarefa

Dia 5: Integração

- Conectar frontend com backend
- Testar fluxos completos
- Ajustar bugs

Dia 6: Testes e Documentação

- Testes unitários principais
- README completo
- Docker Compose

Dia 7: Revisão e Polimento

- Revisar código
- Testar setup do zero
- Ajustes finais

📋 Checklist de Entrega

Mínimo Aceitável (60+ pontos):

- ☐ Login e cadastro funcionam
- ☐ Posso criar, listar, editar e deletar tarefas
- ☐ Validações básicas funcionam

- [ ] Backend e frontend rodam com Docker Compose
- [ ] README com instruções claras
- [ ] Alguns testes (20-40% cobertura)

Bom (70+ pontos):

- [ ] Tudo acima +
- [ ] Filtros por status/prioridade funcionam
- [ ] Frontend responsivo
- [ ] Tratamento de erros adequado
- [ ] 40-60% de cobertura de testes
- [ ] Código organizado

Excelente (85+ pontos):

- [ ] Tudo acima +
- [ ] 1-2 features opcionais
- [ ] 60%+ cobertura de testes
- [ ] Código muito bem organizado
- [ ] Documentação detalhada

🚫 O Que NÃO Fazer

❌ Tentar implementar tudo ❌ Usar tecnologias que você não conhece ❌ Deixar testes para o último dia ❌ Ignorar o README ❌ Complicar demais a arquitetura

✅ Fazer menos, mas fazer bem ✅ Usar tecnologias que você domina ✅ Testar enquanto desenvolve ✅ Documentar desde o início ✅ Manter simples e funcional

---

📧 Entrega

1. Repositório Git público
2. README com:
    - Como rodar (Docker Compose)
    - Tecnologias usadas
    - O que foi implementado
    - O que não foi implementado e por quê
    - Melhorias futuras

3. Enviar link para: rafael.leite@ponte-tech.com.br

💬 Dúvidas

rafael.leite@ponte-tech.com.br Resposta em até 24h úteis

🎯 Lembre-se

> "É melhor entregar 50% muito bem feito do que 100% mal feito."

Foque em:

- ✅ Código limpo
- ✅ Validações corretas
- ✅ Funcionalidades básicas funcionando
- ✅ Bom README

Não se preocupe com:

- ❌ Design perfeito
- ❌ Features avançadas
- ❌ Otimizações prematuras

Boa sorte! 🚀

PonteTech | rafael.leite@ponte-tech.com.br