Apresentação – Teste Técnico (Java 8 + Spring Boot)

Título

- Projeto: API de Usuários e Cartões (Java 8 + Spring Boot)
- Autora: Leticia Rodrigues
- Repositório: https://github.com/letrodrigueszz/Teste-Tecnico
- Site (front): https://letrodrigueszz.github.io/Teste-Tecnico/

Visão Geral

- Objetivo: CRUD de usuários e cartões; vínculo 1:N (user → cards)
- Backend: Spring Boot 2.7, JPA/H2, Flyway, Security (Basic), Swagger
- Frontend: Angular 17 (opcional), Bootstrap 5
- Camadas: controller → service → repository → model (+ DTO e validação)

Requisitos Atendidos

- Obrigatórios: Java 8, Spring Boot, Maven, JPA (H2), camadas, SQL relacional, Angular, GitHub
- Diferenciais: Login (Basic), perfil (ADMIN), Flyway, Swagger, nativeQuery, DTO, responsivo

> *H2 foi escolhido para desenvolvimento rápido; facilmente trocável por Postgres/MySQL.*

Arquitetura Lógica

- Controller: REST + mapeamento de rotas
- Service: regras, validações, hash de senha (BCrypt)
- Repository: Spring Data JPA + queries nativas
- Model: entidades JPA; DTO para contratos de API
- Security: HTTP Basic; Swagger/H2 liberados

Modelo de Dados (JPA)

- User (`users`): id, nome, email (unique, not null), senha (BCrypt); `@OneToMany` cards
- Card (`cards`): id, numeroCartao, nome, status, tipoCartao (ENUM), user (`@ManyToOne`)
- FK com ON DELETE CASCADE (via Flyway)

Migrações (Flyway)

- `src/main/resources/db/migration/V1__init.sql`

```
CREATE TABLE users (
  id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  nome VARCHAR(255),
  email VARCHAR(255) NOT NULL UNIQUE,
  senha VARCHAR(255)
);

CREATE TABLE cards (
```

```
  id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  numero_cartao BIGINT,
  nome VARCHAR(255),
  status BOOLEAN,
  tipo_cartao VARCHAR(50),
  user_id BIGINT,
  CONSTRAINT fk_cards_user FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Endpoints (REST)

- Usuários: GET/POST/PUT/DELETE /usuarios; GET /usuarios/{id}
- Cartões por usuário: POST /usuarios/{id}/cartoes; DELETE /usuarios/{id}/cartoes/{cartaoId}
- Cartões: GET/POST/DELETE /cartoes; PUT /cartoes/{id}/status?status=true|false

> *Erros de validação (400) retornam `{ campo: mensagem }`. 404 para recurso inexistente.*

Contratos (DTO)

- UserDTO

```
{
  "id": number?,
  "nome": string,
  "email": string,
  "senha": string?,
  "cartoes": CardDTO[]?
}
```

- CardDTO

```
{
  "id": number?,
  "numeroCartao": number,
  "nome": string,
  "status": boolean?,
  "tipoCartao": "COMUM" | "ESTUDANTE" | "TRABALHADOR"
}
```

Regras de Negócio

- Senha: hash com BCrypt no `UserService`
- Remoção de usuário remove cartões (cascade/orphanRemoval)
- Status de cartão alterado por endpoint dedicado

Repositórios e nativeQuery

- `UserRepository#findByEmailNative`: `SELECT * FROM users u WHERE u.email = :email`
- `CardRepository#findAllByUserIdNative`: `SELECT * FROM cards c WHERE c.user_id = ?1`

Segurança

- HTTP Basic (in-memory): `admin/admin123`
- Público: `/v3/api-docs/**`, `/swagger-ui.html`, `/swagger-ui/**`, `/h2-console/**`
- CORS liberado para `http://localhost:4200`

Observabilidade e Dev

- Swagger para contratos
- H2 console para dados
- `spring.jpa.show-sql=true` em dev

Como Rodar

- Backend

```
mvn clean package
mvn spring-boot:run
```

- Frontend (opcional)

```
cd frontend
npm install
npm start
```

Decisões de Design

- H2: setup ágil
- Flyway: versionamento do schema
- DTO + Validation + ControllerAdvice: contratos limpos e respostas consistentes
- BCrypt: segurança de senhas; queries nativas: exemplo realista

---

Roadmap / Evoluções

- Usuários/roles persistidos (Security + JPA)
- Postgres/MySQL + Docker Compose
- Testes unitários e integração (Testcontainers)
- Paginação, cache, métricas
- CRUD completo no Angular

Encerramento

- Repositório: `github.com/letrodrigueszz/Teste-Tecnico`
- Swagger: `/swagger-ui.html` • H2: `/h2-console`
- Obrigada!