🧪 Avaliação Técnica – Desenvolvedor(a) Backend Pleno (Java + Spring)

📝 Objetivo

Criar uma API REST para gerenciamento de releases de sistemas. A API deverá ser desenvolvida utilizando **Java** com **Spring Boot**, seguindo as especificações abaixo.

🗃️ Modelo de Dados

Crie uma tabela chamada `release` utilizando o banco de dados H2, com os seguintes campos:

Campo	Tipo	Observações
id	autoIncrement	Chave primária
system	String (255)	Trim automático, obrigatório
version	String (30)	Obrigatório
commits	Json/Array	Lista de commits (strings)
notes	Text	Opcional
user	String (100)	Obrigatório
userUpdate	String (100)	Deve ser extraído do token
releasedAt	DateTime	Data de criação automática
deletedAt	DateTime	Usado para "soft delete"

📍 Endpoints

🔹 Create Release

`POST /releases`

Request Body:

```
{
  "system": "string",
```

```
  "version": "string",
  "commits": ["string", "string"],
  "notes": "string",
  "user": "string"
}
```

Response:

```
{
  "id": 1,
  "message": "Release criado com sucesso."
}
```

🔹 **Get Release by ID**

```
GET /releases/{id}
```

Response:

```
{
  "message": "Release listado com sucesso.",
  "id": 1,
  "system": "string",
  "version": "string",
  "commits": ["string", "string"],
  "notes": "string",
  "user": "string",
  "userUpdate": "string",
  "releasedAt": "2025-05-26T14:00:00Z"
}
```

🔹 **Update Release Notes**

```
PUT /releases/{id}
```

Request Body:

```
{
  "notes": "string"
}
```

Response:

```
{
  "message": "Release atualizado com sucesso."
}
```

🔹 **Delete Release (Soft Delete)**

```
DELETE /releases/{id}
```

Response:

```
{
  "message": "Release deletado com sucesso."
}
```

✅ Requisitos Técnicos

- Java 17+ com Spring Boot
- Persistência com JPA
- Commit da prova em repositório Git público ou zip
- Autenticação JWT (mockada ou simples)
- Tratamento de erros e validações
- Swagger ou documentação de endpoints

🕐 Tempo Sugerido

Até **3 dias corridos** para entrega após o recebimento do desafio.

🎯 Critérios de Avaliação

- Organização do código
- Boas práticas de Java e Spring

- Tratamento de erros e validações
- Arquitetura da aplicação
- Testes unitários e/ou de integração (desejável)
- Clareza na documentação e no uso da API

🔧 Pontos Bônus (não obrigatórios)

- Criar paginação e filtros de pesquisa nas APIs de listagem.
- Criar um dockerfile para a aplicação, que permita a fácil construção de uma imagem Docker da aplicação.