Avaliação Java / Spring: API Rest para Cadastro de Pessoas

Descrição do Projeto

O desafio consiste em criar uma aplicação API Rest para gerenciar um sistema de cadastro de Pessoas e seus respectivos Endereços, onde cada Pessoa pode ter apenas um endereço. O principal objetivo é permitir que operações CRUD (Criar, Ler, Atualizar, Deletar) sejam realizadas na estrutura de Pessoas e Endereços.

Requisitos Técnicos

- A aplicação deve ser criada utilizando Java com Spring Boot (Versão 3.1.10).
- Utilize JPA/Hibernate para persistência de dados, com banco de dados H2 como implementado em aula/exemplo:
    - https://github.com/eduardohen1/Java2024BugLovers/tree/main/AppProdutos
- Implemente tratamento dos dados de entrada e validações necessárias.

Funcionalidades Necessárias

1. ***CRUD de Pessoas:***
    a. Criar Pessoa
    b. Obter Pessoa por ID
    c. Obter Pessoa por ID para mala direta
    d. Listar todas as Pessoas
    e. Atualizar Pessoa por ID
    f. Deletar Pessoa por ID
2. ***CRUD de Endereços:***
    a. Adicionar um novo Endereço a uma Pessoa
    b. Obter Endereço por ID
    c. Atualizar Endereço por ID
    d. Deletar Endereço por ID

Modelagem Sugerida

- ***Pessoa***: deve conter, pelo menos, os seguintes campos:
    - ID (único, não pode ser nulo)
    - Nome (não pode ser nulo)
    - Documento (não pode ser nulo)
- ***Endereço:*** deve conter, pelo menos, os seguintes campos:
    - ID (único, não pode ser nulo)
    - Tipo logradouro (não pode ser nulo)
    - Logradouro (não pode ser nulo)
    - CEP (pode ser nulo)
    - Cidade (pode ser nulo)
    - UF (pode ser nulo)
    - Relacionamento com a entidade Pessoa [1 para 1] (não pode ser nulo)

***Endpoints* Necessários**

- ***Pessoa*:**
    - POST /api/pessoas (cria uma nova Pessoa)
    - GET /api/pessoas/{id} (retorna os dados de uma Pessoa por ID)
    - GET /api/pessoas/maladireta/{id} (retorna os dados de uma Pessoa por ID para mala direta)
        - No *endpoint* de mala direta, utilizar o conceito de DTO. Este conceito cria uma classe diferente da classe Pessoa, com apenas os dados que

precisamos (pesquisar!). Dê preferência para a criação de Records (Java 17+).
        - Utilizar os campos para o DTO: ID; Nome; Concatenação do Endereço, CEP, Cidade, UF
            - Exemplo:

```
{
 “ID”: 1,
 “Nome”: “Fulano”,
 “MalaDireta”: “Rua A, 1 – CEP: 11111-000 – Cidade/UF”
}
```

    - GET /api/pessoas (lista todas as Pessoas)
    - PUT /api/pessoas/{id} (atualiza uma Pessoa existente)
    - DELETE /api/pessoas/{id} (remove uma Pessoa por ID)
- ***Contato***:
    - POST /api/ endereco (adiciona um novo Endereço a uma Pessoa)
    - GET /api/endereco/{id} (retorna os dados de um Endereço por ID)
    - PUT /api/endereco/{id} (atualiza um Endereço existente)
    - DELETE /api/ endereco /{id} (remove um Endereço por ID)

Critérios de Avaliação

- Cumprimento dos requisitos funcionais.
- Organização do código, incluindo estrutura, clareza e limpeza.
- Implementação correta do relacionamento entre entidades (Pessoas e Endereços).
- Uso adequado de padrões de design e melhores práticas do Spring.
- Estratégias de tratamento e validações dos dados de entrada.
- Configuração adequada do ambiente de persistência do JPA/Hibernate.

Instruções de Entrega

- O código deve ser entregue via arquivo tipo ZIP ou 7Z na atividade disponível no **Google Classroom**:
    - https://classroom.google.com/c/NjU2MzI3MjExODkx?cjc=qwagvdz
- Incluir um README com instruções sobre como executar a aplicação.

Prazo de Entrega

- ***11/04/2024***

Bons estudos e bom trabalho!