Exercício de Programação Web Java com Spring Boot: Sistema de Gerenciamento de Tarefas

Descrição: Desenvolver o backend de uma API Rest para gerenciamento de tarefas utilizando Java com Spring Boot. O usuário poderá cadastrar, visualizar, editar e excluir tarefas. As tarefas deverão ser armazenadas em um banco de dados relacional.

Link do projeto desenvolvido em aula: https://github.com/douglasfkln/ms-java-springboot

Requisitos:

1. Utilize o Spring Initializer ([https://start.spring.io/]) para criar um novo projeto Spring Boot com as seguintes configurações:
    a. Projeto: Manven
    b. Linguagem: Java na versão 22
    c. Spring Boot: 3.2.5
    d. Defina o nome do grupo e do artefato do seu projeto.
2. Adicione as dependências:
    a. Spring Web
    b. Spring Data JPA
    c. H2 Database (ou se preferir utilizar outro banco de dados de sua preferência)
3. Baixe o projeto, abra no IntelliJ e deixe o Maven fazer o download das dependências;
4. Confirmar se o IntelliJ possui uma JDK do Java na versão 22. Você pode verificar em *File > Project Structure;*
5. Crie a estrutura de pacotes a seguir:
    a. *entites*
    b. repositories
    c. controllers
    d. services
6. Crie uma classe de entidade para `Tarefa` com os atributos: `id`, `descricao`, `dataCriacao`, `dataLimite` e `finalizada`.
7. Anote a classe com `@Entity` para indicar que ela será mapeada para uma tabela no banco de dados e faça as demais anotações necessárias nos atributos.
8. Criar o Repositório:
    a. Crie uma interface `TarefaRepository` que extends `JpaRepository<Tarefa, Long>`.
    b. Essa interface vai fornecer métodos prontos para realizar operações CRUD (Criar, Ler, Atualizar, Deletar) com a entidade `Tarefa`.
9. Implementar o Controlador:
    a. Crie uma classe `TarefaController` anotada com `@RestController e @RequestMapping(value = "/tasks")`.
    b. Criar métodos GetAll, GetById, POST, PUT e DELETE.
    c. Usar as anotações necessárias em cada endpoint.
10. Crie uma classe Service para encapsular a lógica de negócios relacionada às operações CRUD das tarefas.

Endpoints:

1. Defina métodos para cada operação CRUD:
    a. `@PostMapping("/tasks")`: Cadastrar uma nova tarefa.
    b. `@GetMapping("/tasks")`: Buscar todas as tarefas.
    c. `@GetMapping("/tasks/{id}")`: Buscar uma tarefa específica por ID.
    d. `@PutMapping("/tasks/{id}")`: Atualizar uma tarefa.
    e. `@DeleteMapping("/tasks/{id}")`: Deletar uma tarefa.
2. Implemente um novo endpoint que faça a atualização de status da tarefa para `finalizada=true`.
3. Utilize ferramentas como Postman ou Insônia para testar os endpoints da API REST.
4. Verifique se as operações CRUD estão funcionando corretamente.
5. Siga as melhores práticas de desenvolvimento de software, como o uso de padrões de design e documentação adequada.
6. Estudos extra classe
    a. Implementar testes unitários pelo menos na classe de Service para garantir o bom funcionamento do seu código. A lib de sua preferência.
    b. Implementar uma documentação com Swagger. Pode ser utilizado o https://editor.swagger.io/ para auxílio.

Entrega do Trabalho:

O trabalho deverá ser versionado no github e entregue apenas o link do repositório no sistema. O repositório precisa ter um arquivo README com instruções de como rodar a API e o arquivo yml da documentação do swagger. O exercício pode ser feito em grupo de até 3 pessoas.

Recursos Adicionais:

Documentação do Spring Boot:
https://docs.spring.io/spring-framework/reference/index.html
Tutoriais Spring Boot: https://www.baeldung.com/spring-boot
Exemplos de CRUD com Spring Boot: https://github.com/topics/spring-boot-crud