ATIVIDADE – Spring

Atividade prática – Desenvolvendo o Backend com Spring

Instruções gerais:

1. Utilize o Spring Tool Suite (STS) para desenvolver os projetos Spring.
2. Crie um repositório no Github para cada projeto Spring
3. Ao concluir uma nova etapa do projeto, envie todas as alterações para o Repositório do projeto criado na sua conta pessoal do Github (a pasta inteira)
4. Envie o link do repositório no Github através da Plataforma da Generation na data indicada
5. Caso seja solicitado, adicione os links individuais dos arquivos .JAVA indicados, no item: **Adicione um dos links da sua entrega**, localizada depois do link do Repositório, na tela de entrega da atividade na plataforma, para validação da atividade.

Mantenha as entregas das Atividades em dia na Plataforma da Generation

EXERCÍCIOS

Boas práticas:

1) Na Camada Model, crie a segunda Classe Model e o Relacionamento entre as 2 Classes.
2) Execute a sua aplicação no STS e verifique se a tabela e o relacionamento foram criados no **MySQL Workbench**.
3) Na Camada Repository, crie a segunda Interface Repository.
4) Na nova Interface Repository, crie um Método de Busca Específica
5) Na Camada Controller, crie a segunda Classe Controller.
6) Na Classe Controller, crie todes os Métodos do **CRUD.**
7) Teste todes os métodos do CRUD através do **Insomnia**.
8) Envie as atualizações do projeto para o repositório remoto.

ATIVIDADE CRUD BACKEND BLOG-PESSOAL – PARTE 03

Na terceira parte do Projeto Blog Pessoal iremos adicionar as seguintes características:

01 – Criar o Recurso Tema

1) Na Camada Model será criada a Classe **Tema** com os seus respectivos atributos.
2) A Classe Tema criará a Tabela **tb_temas** no Banco de dados **db_blogpessoal**.
3) Na Camada Repository será criada a Interface **TemaRepository** (com a capacidade de se comunicar com o banco de dados MySQL).
4) Na Interface **TemaRepository** vamos adicionar o Método de busca específica:
    - **findAllByDescricaoContainingIgnoreCase()** com a função de trazer todes os Temas cuja descrição possua a palavra pesquisada.
5) Na Camada Controller será criada a Classe **TemaController.**
6) Na Classe **TemaController** serão criados os seguintes endpoints:
    - **getAll()** com a capacidade de listar todes os Temas.
    - **getById()** com a função de trazer um único Tema identificado pelo id.
    - **getByDescricao()** com a função de trazer todes os Temas cuja descrição possua a palavra pesquisada.
    - **post()** com a função de gravar (persistir) um novo Tema no Banco de dados.
    - **put()** com a função de atualizar os dados de um Tema.
    - **delete()** com a função de apagar um Tema no banco de dados.

02 – Criar o Relacionamento entre as Classes Tema e Postagem

1) Na Camada Model, na Classe **Tema**, crie a Relação **OneToMany** com a Classe **Postagem**. Não esqueça de adicionar todas as **Anotações** e **Métodos Get e Set** necessários.
2) Na Camada Model, na Classe **Postagem**, crie a Relação **ManyToOne** com a Classe **Tema**. Não esqueça de adicionar todas as **Anotações** e **Métodos Get e Set** necessários.
3) Em especial, não esqueça da anotação **@JsonIgnoreProperties** para inibir a Recursividade no relacionamento.
4) Faça os ajustes necessários no **Insomnia** para Inserir e Atualizar os dados do Recurso Postagem, habilitando o Relacionamento.
5) Teste todes os Métodos dos Recursos Tema e Postagem no **Insomnia**.
6) Verifique nos Testes se ao listar as Postagens, o respectivo Tema associado é exibido, como mostra a figura abaixo:

7) Verifique nos Testes se ao listar os Temas, as Postagens associadas serão exibidas, como mostra a figura abaixo: