Lista 24 –Spring DATA

O exemplo da aula do dia 10.02 e a apresentação 'aula 50 – spring data' foram atualizados a fim de realizar consultadas *case insensitive*. Por favor baixe as versões atualizadas destes arquivos.

1. Utilizando como base o sistema desenvolvido nas aulas dos dias 09.02 e 10.20.2023, elabore do zero um sistema semelhante, utilizando o SGBD H2, que realize o cadastro de alunos, da seguinte forma:
    a. A tabela se chamará `aluno`, com os seguintes campos:
        - `id: LONG AUTO_INCREMENT`
        - `nome: VARCHAR(100)`
        - `salario: NUMERIC(10, 2)`
        - `nascimento: DATE`
    b. Crie o *script* de criação da tabela no arquivo $schema.sql$ e popule a tabela com alguns registros no arquivo $data.sql$
    c. A entidade deverá ser uma classe (e não um *record*) chamada `Aluno`, com os seguintes atributos:
        - `id: Long`
        - `nome: String`
        - `salario: BigDecimal`
        - `nascimento: Date`
    d. Crie uma interface chamada `AlunoRepository` que estende de `CrudRepository`.
        - Declare um método personalizado em `AlunoRepository` sem utilizar `@Query` e nem JPQL que busque os alunos por uma parte do nome sem levar em consideração as letras maiúsculas / minúsculas.
        - Crie uma consulta personalizada com `@Query` / JPQL chamada `porSalario()` que busque por alunos que ganhem um salário maior ou igual ao informado no parâmetro de busca.
        - Crie uma consulta personalizada com `@Query` / JPQL de forma a buscar alunos por uma parte do nome, utilizando `like` e que seja *case insensitive*.
    e. Implemente na classe principal o método `commandLineRunner()` e faça alguns testes com o repositório, como por exemplo com os métodos `save()`, `delete()`, `findById()` e `findAll()`, para se certificar que os dados estão sendo gravados na base.
    f. Crie uma classe do tipo REST *controller* chamada `AlunoController` om 3 métodos de listagem, `listarComFiltro()`, cujo *end-point* será "`/filtro`", `listarTudo()` cujo *end-point* será "`/`" e `listarPorSalario()`, com *end-point* "`/salario`". O repositório deverá ser instanciado através do recurso de injeção de dependências do Spring. O método `listarTudo()` irá retornar os dados por meio do método `findAll()` do repositório. O método `listarComFiltro()` irá chamar a consulta personalizada com JQPL. O método `listarPorSalario()` irá retornar os dados por meio do método `porSalario()` do repositório.
    g. Com base na classe `ProdutoRestController` do 'roteiro spring rest - 02.02 (back-end) e 03.02 (front-end).pdf' que trabalhou com um CRUD somente em memória e com o conteúdo visto nesta semana, crie as funcionalidades de inclusão, alteração e exclusão de alunos usando o SGBD H2 na classe `AlunoController`. Para isso basta utilizar os métodos já disponibilizados no repositório.
    h. Integre no sistema o *front-end* feito na lista 23, adaptando-o para funcionar com a tabela de alunos.
    i. Utilize os recursos de localização do JavaScript para formatar os campos de salário e data de nascimento corretamente nas *interfaces web*. Na trilha da aula 50 há exemplos disso no projeto 'lista 22 – front-end'.

2. Utilize o Postman para todos os métodos da API Rest implementados na classe `AlunoController`. Faça um *print* das telas do Postman em um arquivo e exporte-o em PDF para posterior entrega.

3. Após finalizar o sistema desenvolvido no item 1 acima e se certificar que ele está funcionando corretamente, crie um repositório no GitHub para armazená-lo na nuvem. Utilize o GitBash para criar um repositório local e conectá-lo ao repositório no GitHub. Faça o *commit* com a versão atual do sistema e então realize o *push* para enviar o projeto para o repositório remoto.