1DEVinHouse

Módulo 2 - Projeto Avaliativo 2

SUMÁRIO

1 INTRODUÇÃO

A **DEVinPharmacy LTDA**, uma renomada empresa do ramo farmacêutico, está expandindo a sua rede de lojas. Por conta da expansão, o time de gestão necessita da criação de um sistema online, intitulado **PharmacyManagement**, para gerenciamento de medicamentos e farmácias. O seu perfil chamou a atenção dos gestores para a criação do **back-end** do sistema que deverá ser codificado em **Java**, utilizando o framework **Spring Boot**.

2 REQUISITOS DA APLICAÇÃO

A aplicação que deverá ser realizada **individualmente** deve contemplar os seguintes requisitos:

1. A aplicação deverá ser codificada utilizando o framework **Spring Boot** juntamente com a linguagem de programação **Java**, e o banco de dados **PostgreSQL**;
2. Deve retornar todos os dados no formato **JSON**;
3. Deve seguir as **boas práticas de programação** Java e Spring Boot;
4. Deve ser criado um **readme.md** completo, com todas as informações necessárias sobre o projeto e como utilizar;
5. Deve ser utilizado o **GitHub**, seguido o padrão GitFlow apresentado na primeira aula da semana 2 do módulo 1;
6. Deve seguir o **roteiro da aplicação**.

3 ROTEIRO DA APLICAÇÃO

O software a ser desenvolvido será o back-end do Projeto 2 do Módulo 1. Sendo assim ele deve conter os seguintes endpoints:

1. Um endpoint /usuario que deverá ter 2 caminhos:
    a. O endpoint /usuario/login que será usado para o recebimento das informações de login do usuário e deverá retornar o id do usuário.
    b. O endpoint /usuario/cadastro que deverá receber um novo login de usuário e deverá cadastrar ele no banco de dados.

2. Um endpoint /farmacia que deverá conter as seguintes funcionalidades
    a. Buscar todas as farmácias e endereço
    b. Buscar uma farmácia por ID
    c. Adicionar uma nova farmácia e endereço
    d. Atualizar uma farmácia por ID
    e. Deletar uma farmácia por ID

3. Um endpoint /medicamentos
    a. Buscar todos os medicamentos
    b. Buscar um medicamento por ID
    c. Adicionar um novo medicamento
    d. Atualizar um medicamento por ID
    e. Deletar um medicamento por ID

Cada funcionalidade acima deve utilizar o método HTTP corresponde ou mais próximo da funcionalidade, e cada resposta deve retornar com o status correto para o tipo de operação realizada.
Onde temos o ID envolvido devemos utilizar o padrão de o ID ser parte do caminho da url, sendo assim teríamos por exemplo ***/farmacia/1***, esse se refere a farmácia com ID 1 e o método usado nesse endpoint irá definir a operação realizada.

Por exemplo:

- Criação de itens: POST, retorno 201 (CREATED)
- Procura de iten(s): GET, retorno 200 (OK)
- Update de itens: PUT, retorno 200
- Deleção de itens: DELETE, retorno 202

As respostas devem ter a seguinte estrutura padrão de mensagem:

- Status code
- Mensagem
- Dados
    - Deve possuir tanto classes quanto listas com os dados da resposta

3.1 TRATAMENTO DE ERROS

O software deve conter tratamentos para os erros que ocorrerem no programa. Os erros devem ser retornados com o status HTTP correto.

As mensagens de erro devem seguir a seguinte estrutura padrão de erro:

- Status code
- Mensagem
- Erro
- Causa

3.2 BANCO DE DADOS

O software deve ter uma conexão com o banco de dados externo, e deve armazenar todos os dados tratados pelo programa.

O banco de dados deve ser um instância PostgreSQL e deve ser gerado um script de criação de cada entidade. Esse script deve ser adicionado a raíz do projeto na pasta **script**.

As entidades do banco de dados devem seguir a seguinte estrutura:

- Usuário
    - Identificador (obrigatório, chave primária)
    - Email (obrigatório)
    - Senha (obrigatório)

- Farmácia
    - Identificador (obrigatório, chave primária)
    - Razão social (obrigatório)
    - CNPJ (obrigatório)
    - Nome Fantasia (obrigatório)
    - E-mail (obrigatório)
    - Telefone (opcional)

    - Celular (obrigatório)
    - Identificador Endereço (obrigatório)

- Endereço
    - Identificador (obrigatório, chave primária)
    - CEP (obrigatório)
    - Logradouro/Endereço (obrigatório)
    - Número (obrigatório)
    - Bairro (obrigatório)
    - Cidade (obrigatório)
    - Estado (obrigatório)
    - Complemento (opcional)
    - Latitude (obrigatório)
    - Longitude (obrigatório)

- Medicamento
    - Identificador (obrigatório, chave primária)
    - Nome do medicamento (obrigatório)
    - Nome do laboratório (obrigatório)
    - Dosagem do medicamento (obrigatório)
    - Descrição do medicamento (opcional)
    - Preço unitário do medicamento (obrigatório)
    - Tipo do medicamento (obrigatório)

3.3 SERVIÇO EXTERNO

Adicione uma consulta do **CEP** pela **API** ViaCEP do IBGE Brasil. Essa chamada deve ser feita utilizando uma das seguintes tecnologias:

- RestTemplate
- Feign
- Outra tecnologia para chamada de API no Java/Spring de sua escolha.

3.4 DESAFIO EXTRA

Utilizando fetch ou axios, realize a integração do back-end criado neste projeto, com o front-end criado no projeto anterior (M1P2).

4 CRITÉRIOS DE AVALIAÇÃO

A tabela abaixo apresenta os critérios que serão avaliados durante a correção do projeto. O mesmo possui variação de nota de 0 (zero) a 10 (dez) como nota mínima e máxima, e possui peso de **40% sobre a avaliação do módulo**.

Serão **desconsiderados e atribuída a nota 0 (zero)** os projetos que apresentarem plágio de soluções encontradas na internet ou de outros colegas. Lembre-se: Você está livre para utilizar outras soluções como base, mas **não é permitida** a cópia.

Nº	Critério de Avaliação	0	-	0,50
1	O aluno criou um readme.md completo, com todas as informações necessárias sobre o projeto e como utilizar?	O aluno **não** criou um redme.md completo com todas as informações necessárias sobre o projeto e como utilizá-lo.	-	O aluno criou um redme.md completo com todas as informações necessárias sobre o projeto e como utilizá-lo.
2	O aluno seguiu o padrão de versionamento de código GitFlow?	O aluno **não** seguiu o padrão de versionamento GitFlow.	-	O aluno seguiu o padrão de versionamento GitFlow.
Nº	**Critério de Avaliação**	**0**	**0,50 a 1,00**	**2,00**
3	O aluno realizou a implementação dos Endpoints com os métodos e respostas corretos?	O aluno **não** realizou a implementação dos Endpoints com os métodos e respostas corretos	O aluno realizou a implementação dos Endpoints de forma parcial, ou com respostas ou métodos errados	O aluno realizou a implementação dos Endpoints com os métodos e respostas corretos de forma completa
4	O aluno realizou o tratamento de Erro com os métodos e respostas corretos?	O aluno **não** realizou o tratamento de erro com os métodos e respostas corretos	O aluno realizou o tratamento de Erro com os métodos e respostas corretos	O aluno realizou o tratamento de Erro com os métodos e respostas corretos de forma completa
Nº	**Critério de Avaliação**	**0**	**0,25 a 0,50**	**1,00**
5	O aluno desenvolveu as Entidades do Banco de Dados?	O aluno **não** desenvolveu as Entidades do Banco de Dados	O aluno desenvolveu Entidades do Banco de Dados parcialmente ou com falhas	O aluno desenvolveu Entidades do Banco de Dados corretamente
6	O aluno implementou a conexão com o Banco de Dados PostgreSQL?	O aluno **não** implementou a conexão com o	O aluno implementou a conexão com um	O aluno implementou a conexão com o

		Banco de Dados PostgreSQL	banco de dados em memória	Banco de Dados PostgreSQL
7	O aluno desenvolveu o script de Criação de tabelas do Banco de Dados?	O aluno **não** desenvolveu o script de Criação de tabelas do Banco de Dados	O aluno desenvolveu o script de Criação de tabelas do Banco de Dados, porém houve algum erro na execução do script	O aluno desenvolveu o script de Criação de tabelas do Banco de Dados e foi executado sem problemas
8	O aluno programou as resposta do endpoints de forma correta?	O aluno **não** programou as resposta do endpoints de forma correta	O aluno programou as resposta do endpoints de forma parcial, ou com algum item faltando	O aluno programou as resposta do endpoints de forma correta
9	A chamada da API ViaCEP foi realizada com sucesso?	O aluno **não** realizou a chamada da API ViaCEP.	O aluno realizou a chamada da API ViaCEP, porém houve erros na implementação ou execução	O aluno realizou a chamada da API ViaCEP de forma bem sucedida
Nº	**Critério Extra**	**0**	**-**	**1,00**
10	O aluno realizou a integração com a aplicação front-end criada no projeto anterior?	O aluno **não** realizou a integração com a aplicação front-end criada no projeto anterior.		O aluno realizou a integração com a aplicação front-end criada no projeto anterior, e forneceu o repositório da aplicação anterior atualizada.

5 ENTREGA

O código desenvolvido deverá ser submetido no **GitHub**, e o link deverá ser disponibilizado na tarefa **Módulo 2 - Projeto Avaliativo 2**, presente na semana 12 do AVA até o dia **15/01/2023** às **23h55**.

O repositório deverá ser **privado**, com as seguintes pessoas adicionadas:

- André Santana Nunes- **andresantnunes**
- Operação DEVinHouse - **devinhouse-operacao**

Importante:

1. Não serão aceitos projetos submetidos **após a data limite da atividade**, e, ou **alterados** depois de entregues.
2. Será considerado como data final de entrega a **última atualização** no repositório do projeto no GitHub. Lembre-se de não modificar o código até receber sua nota.
3. Não esqueça de **submeter o link no AVA**. Não serão aceitos projetos em que os links não tenham sido submetidos.

6 PLANO DE PROJETO

Ao construir a aplicação proposta, o aluno estará colocando em prática os aprendizados em:

- **Java:**
  - Introdução ao Java (instalação, sintaxe básica, compilação e execução)
  - Estruturas de Controle de Fluxo (condicional, repetição)
  - Arrays e ArrayList
  - Documentação de Código
  - Conceitos de POO: classes, objetos, métodos, atributos, construtores
  - Modificadores (private, public, static, final), encapsulamento e sobrecarga
  - Herança e Polimorfismo (extends, implements, protected, sobrescrita)
  - Tratamento de Exceções
- **SQL**:
  - Configuração do ambiente PostgreSQL e comandos DDL (CREATE, ALTER, DROP)
  - Modelo relacional e comandos DML (INSERT, UPDATE, DELETE, SELECT)
- **Spring**:
  - Java Servlets, Spring Boot, Spring Core (injeção de dependências)
  - Spring Data
  - CRUD REST API: GET, POST, PUT, DELETE
- **API**:
  - Criação de APIs funcionais e de qualidade.