Ecossistema Spring

O Spring Boot é um dos projetos que fazem parte desse ecossistema.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Spring Security: usado para implementação de autorização e autenticação.

Spring Cloud: permite a construção de arquiteturas de microserviços completas.

Spring Data: permite trabalhar com diferentes modelos de base de dados.

Spring Web: Utilizado para a construção da API.

Interface gráfica do usuário

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Preparação do ambiente:

- Instalar o Java 17.

- Nesse curso será usado o spring initializr [Spring Initializr](https://start.spring.io/)

- Definir o gerenciador de dependências, nesse caso será o Maven

- Sempre estará selecionada automaticamente a versão mais atualizada do spring boot.

Interface gráfica do usuário

Descrição gerada automaticamente

- Selecionar em add dependences as dependências essências para o projeto.

No caso:

- Spring Web (construções de aplicações web, incluindo RESTful)

- Spring Data JPA (visto que será usado um banco de dados SQL)

- PostgreSQL Driver

- Validation (incluir algumas validações iniciais)

- Após clicar em generate, será gerado um arquivo .zip que precisa ser descompactado e importado na IDE.

- Ao importar, esperar um pouco para o MAVEN fazer o download e baixar todas as dependências iniciais.

- No caminho src/main/java/resources/application.properties, realizar a conexão com a base de dados. No caso, criei uma base de teste no pgadmin (foi necessário baixar o pgadmin e o postgresql)

Texto

Descrição gerada automaticamente

- Foi criado um pacote chamado model clicando com o botão direito em -> com.example.springboot

Interface gráfica do usuário

Descrição gerada automaticamente

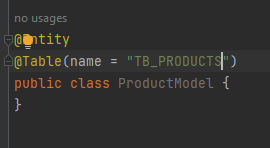
- Criada nova class clicando com o botão direito em -> model

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Pois será uma API de produtos RESTful seguindo o modelo de maturidade definido por Leonard Richardson, que considera 4 níveis para que uma API atinja e seja considerada RESTful.

Para que essa classe seja uma entidade da base de dados, para que se faça o mapeamento de uma classe java para uma entidade no banco, é necessário utilizar algumas anotações que facilitam todo esse processo, como por exemplo: @Entity e @Table (necessário informar qual será o nome da tabela).



Implements Serializable -> interface onde se mostra para a JVM que essa é uma classe habilitada a passar por serializações. É uma interface apenas de marcação, não tem nenhum método para implementar.

Criados os getters e setters dos atributos.

Criado novo pacote onde serão definidos os repositórios.

Interface gráfica do usuário

Descrição gerada automaticamente

Criada uma interface dentro desse pacote.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Criado um novo pacote:

Texto

Descrição gerada automaticamente

Criada uma classe dentro:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Criado novo pacote:

Interface gráfica do usuário, Texto, Site

Descrição gerada automaticamente

Criado um record dentro:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

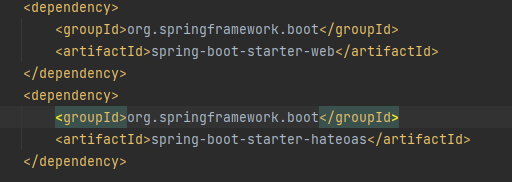
4 Níveis para que uma API seja considerada REST.

0º) Utilização do protocolo HTTP.

1º) Possua recursos bem definidos e correto uso das URIs, ou seja, quando for definir cada uma das URIs desses recursos, utilizar de substantivos para que fique dentro dos padrões e que contemple cada nível proposto dentro do modelo de maturidade.

2º) Utilização dos métodos HTTP de forma semântica, ou seja, quando for utilizar um método para salvar, utilizar o POST. Se quiser apenas listar a leitura desses recursos, utilizar o GET. Se quiser deletar um recurso, usar o DELETE e para atualizar, utilizar o PUT.

3º) Deve apresentar hipermídias (navegabilidade entre os recursos) HATEOAS. É necessário inserir mais uma dependência no POM.xml.



Clicar em maven para ele fazer os downloads necessários.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Realizar um extendes para RepresentationModel<ProductModel>

Texto

Descrição gerada automaticamente

Explicando o @PostMapping

Interface gráfica do usuário, Texto

Descrição gerada automaticamente



Inicialização do método. O retorno do método será o ResponseEntity do tipo ProductModel. O método se chama saveProduct que vai receber como corpo da solicitação HTTP via POST, o ProductRecordDto (Por isso precisa colocar a anotação RequestBody). Para que as validações @NotNull e @NotBlank entrem em vigor , é necessário incluir no início do método a anotação Valid, porque se não nenhuma anotação será feita.

Tela de celular com aplicativo aberto

Descrição gerada automaticamente