

Documentação do Projeto Integrador: ArtAtiva

The image shows a dark-themed web form for configuring a new Spring project. It includes sections for Project type, Language, Spring Boot version, and Project Metadata. The 'Project' section has 'Maven Project' selected. The 'Language' section has 'Java' selected. The 'Spring Boot' section has '2.7.1' selected. The 'Project Metadata' section includes fields for Group, Artifact, Name, Description, and Package name, all filled with project-specific information. The 'Packaging' section has 'Jar' selected, and the 'Java' version section has '11' selected.

Project		Language	
<input checked="" type="radio"/> Maven Project		<input checked="" type="radio"/> Java	<input type="radio"/> Kotlin
<input type="radio"/> Gradle Project		<input type="radio"/> Groovy	

Spring Boot

☐ 3.0.0 (SNAPSHOT) ☐ 3.0.0 (M3) ☐ 2.7.2 (SNAPSHOT)
☒ 2.7.1 ☐ 2.6.10 (SNAPSHOT) ☐ 2.6.9

Project Metadata

Group

Artifact

Name

Description

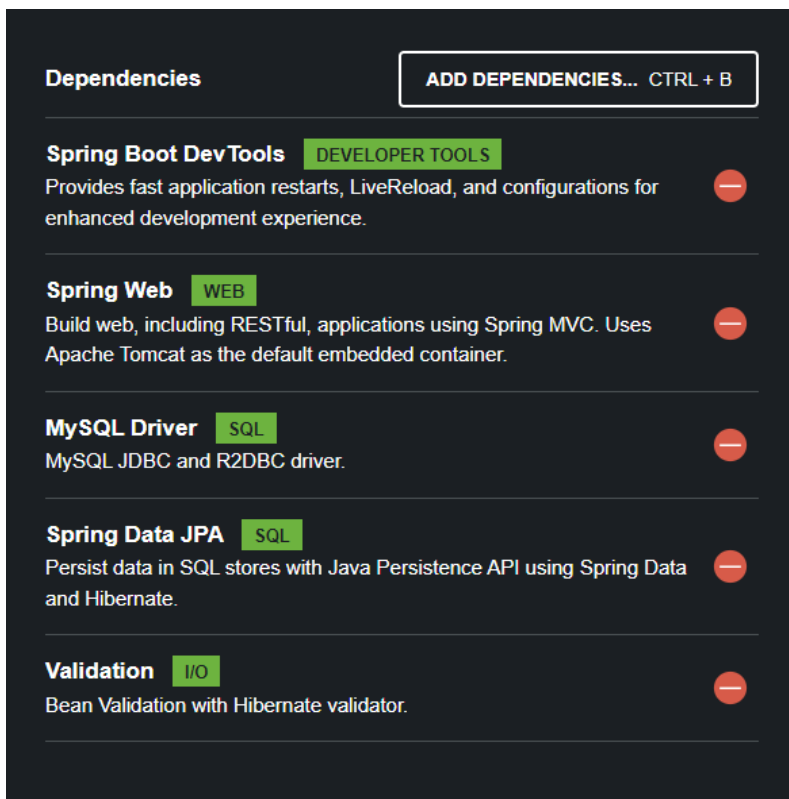
Package name

Packaging ☒ Jar ☐ War

Java ☐ 18 ☐ 17 ☒ 11 ☐ 8

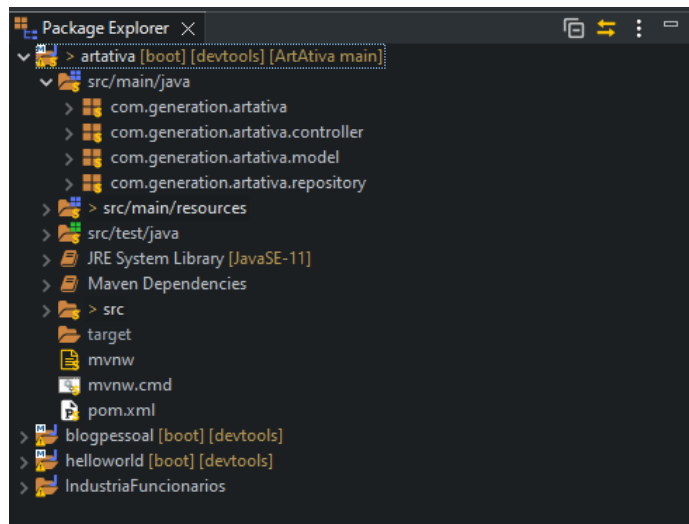
Em primeiro lugar, é necessário criar as configurações iniciais que darão base a nossa API Rest. É possível efetuar as configurações iniciais do projeto Spring dentro da própria ferramenta Eclipse/STS, no entanto, nos exemplos iremos utilizar o site Spring Initializer.

Conforme as boas práticas sugerem, o nome do pacote se inicializa sempre com o mesmo padrão “.com.generation.nomedoprojeto”. Cabe ressaltar ainda que a versão mais indicada do Spring Boot é a 2.7.1 e a versão 11 do Java.



Para que o projeto se torne executável é importante acrescentar as dependências que serão utilizadas. Selecionamos as principais: Spring Boot Dev Tools, Spring Web, MySQL Driver, Spring Data JPA e Validation.

Após finalizado as configurações e efetuado o download do projeto, ao importar o mesmo via IDE o resultado obtido foi esse:



Um dos arquivos mais importantes a ser configurado antes de iniciarmos o desenvolvimento das classes model é o application properties. Com ele fazemos a conexão do projeto com o banco de dados do MySQL.

```
*application.properties X
1 spring.jpa.hibernate.ddl-auto=update
2 spring.jpa.database=mysql
3 spring.datasource.url=jdbc:mysql://localhost/db_artativa?createDatabaseIfNotExist=true&serve
4 spring.datasource.username=root
5 spring.datasource.password=root
6
7 spring.jpa.show-sql=true
8
9 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL8Dialect
10
11 spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
12 spring.jackson.time-zone=Brazil/East
13
```

Os atributos escolhidos para a classe “produto” são: id, nome, descrição, preço, foto, quantidade, data, categoria_id e usuario_id . O id é utilizado como chave primária para identificar o produto, o nome é utilizado para registrar o produto, a descrição serve para registrar uma breve descrição, a foto será utilizada para exibir uma imagem, o preço é utilizado para registrar o valor, usuario_id serve para identificar o usuário que registrou o produto e a categoria_id é utilizada para registrar em qual categoria o produto pertence.

```

application.properties  J Produto.java X
18 @Entity
19 @Table(name = "tb_produto")
20 public class Produto {
21
22     @Id
23     @GeneratedValue(strategy = GenerationType.IDENTITY)
24     private long id;
25
26     @NotNull
27     @Size(min = 3, max = 255)
28     private String nome;
29
30     @NotNull
31     @Size(min = 3, max = 600)
32     private String descricao;
33
34     @NotNull
35     private double preco;
36
37     @NotNull
38     private int quantidade;
39
40     private String foto;
41
42     @UpdateTimeStamp
43     private LocalDateTime data;
44
45     @ManyToOne
46     @JsonIgnoreProperties("produto")
47     private Usuario usuario;
48
49     @ManyToOne
50     @JsonIgnoreProperties("produto")
51     private Categoria categoria;
52

```

Os atributos da tabela usuário foram escolhidos com base na necessidade de armazenar informações básicas sobre os usuários do sistema. O atributo id é utilizado para identificar de forma única cada usuário, o atributo nome armazena o nome completo do usuário, o atributo usuario armazena o nome de usuário utilizado para login no sistema e o atributo senha armazena a senha utilizada para login no sistema. Por fim, o atributo foto armazena a foto do usuário, que pode ser utilizada para identificação visual.

```

17 @Entity
18 @Table (name = "tb_usuario")
19 public class Usuario {
20
21     @Id
22     @GeneratedValue(strategy = GenerationType.IDENTITY)
23     private long id;
24
25     @NotNull
26     @Size(min = 3, max = 255)
27     private String nome;
28
29     @NotNull
30     @Size(min = 3, max = 255)
31     private String usuario;
32
33     @NotNull
34     @Size(min = 3, max = 255)
35     private String senha;
36
37     private String foto;
38
39     @OneToMany(mappedBy = "usuario", cascade = CascadeType.ALL)
40     @JsonIgnoreProperties("usuario")
41     private List<Produto> produto;

```

Na última classe de categoria, os atributos restantes que foram usados são: id, utilizado como chave primária, nome, para ser usado como identificador

da categoria e uma descrição para fornecer mais detalhes sobre ela.

```
*application.properties  Produto.java  Usuario.java  Categoria.java ×
1 package com.generation.artativa.model;
2
3 import java.util.List;
16
17
18 @Entity
19 @Table(name = "tb_categoria")
20 public class Categoria {
21
22
23     @Id
24     @GeneratedValue(strategy = GenerationType.IDENTITY)
25     private long id;
26
27     @NotNull
28     @Size(min = 3, max = 255)
29     private String nome;
30
31     @NotNull
32     @Size(min = 3, max = 255)
33     private String descricao;
34
35     @OneToMany(mappedBy = "categoria", cascade = CascadeType.ALL)
36     @JsonIgnoreProperties
37     private List<Produto> produto;
38
```