Spring Boot

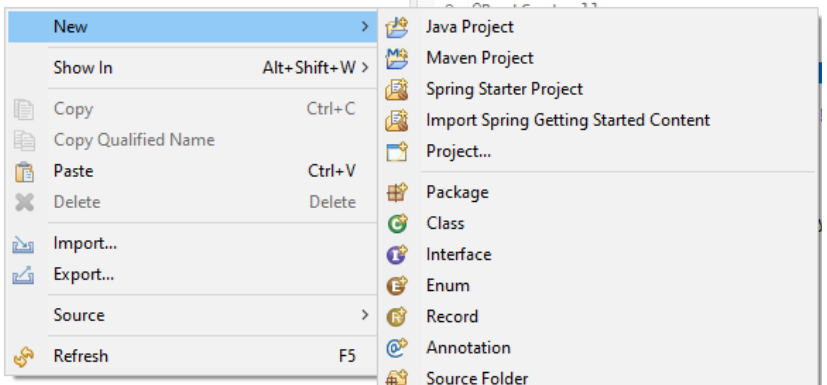
Criando uma aplicação que acessa banco de dados

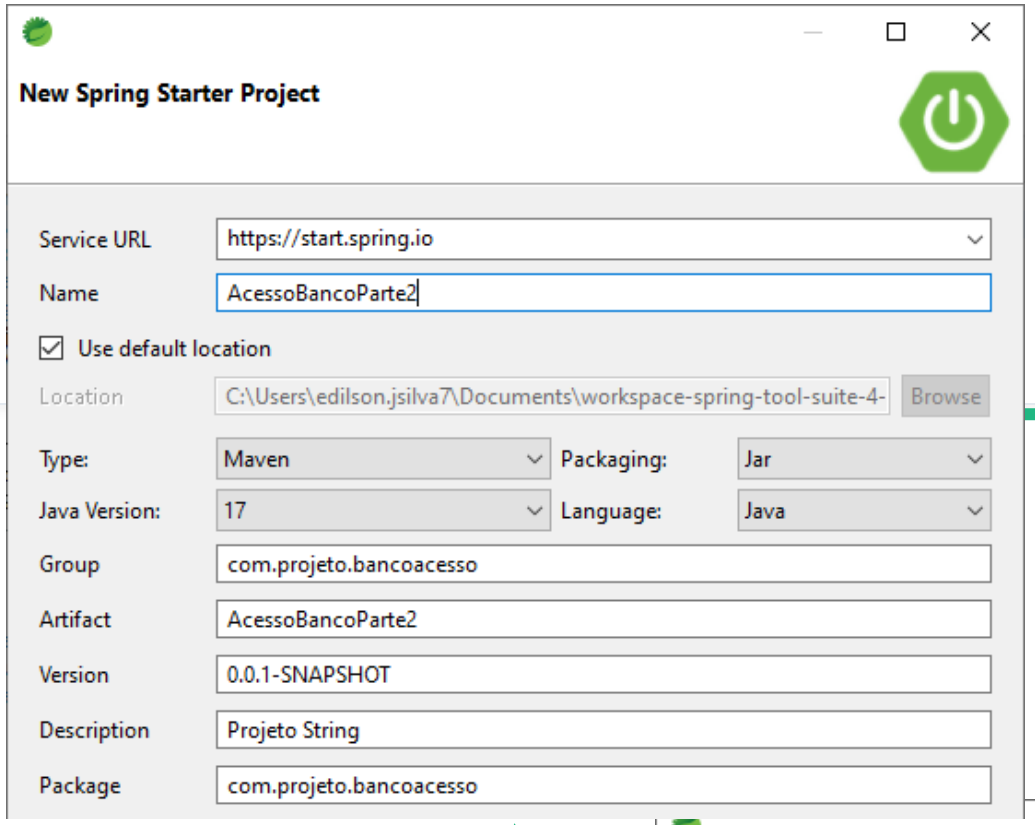
Spring Boot

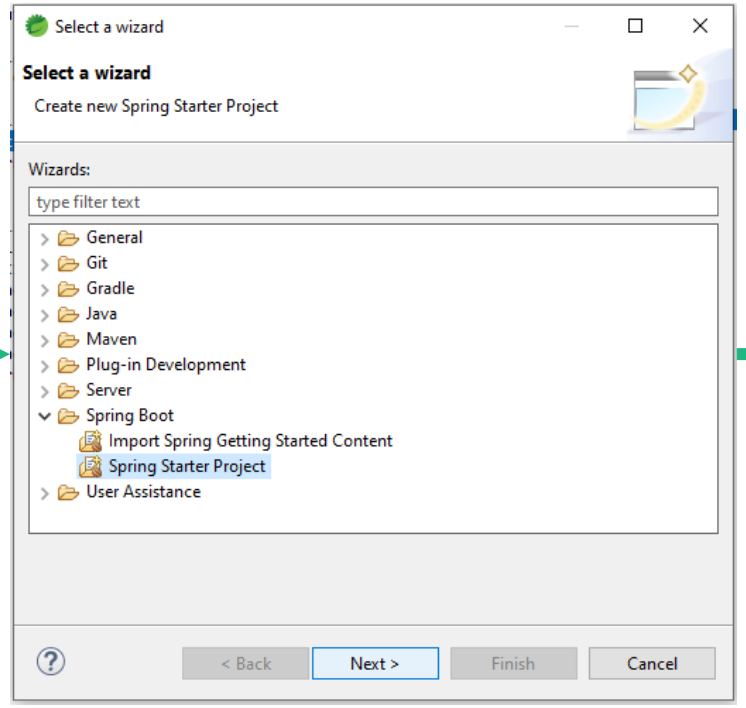
Crie um novo projeto no STS (Spring Tools Suite) com o nome de AcessoBanco. Adicione as seguintes dependências:

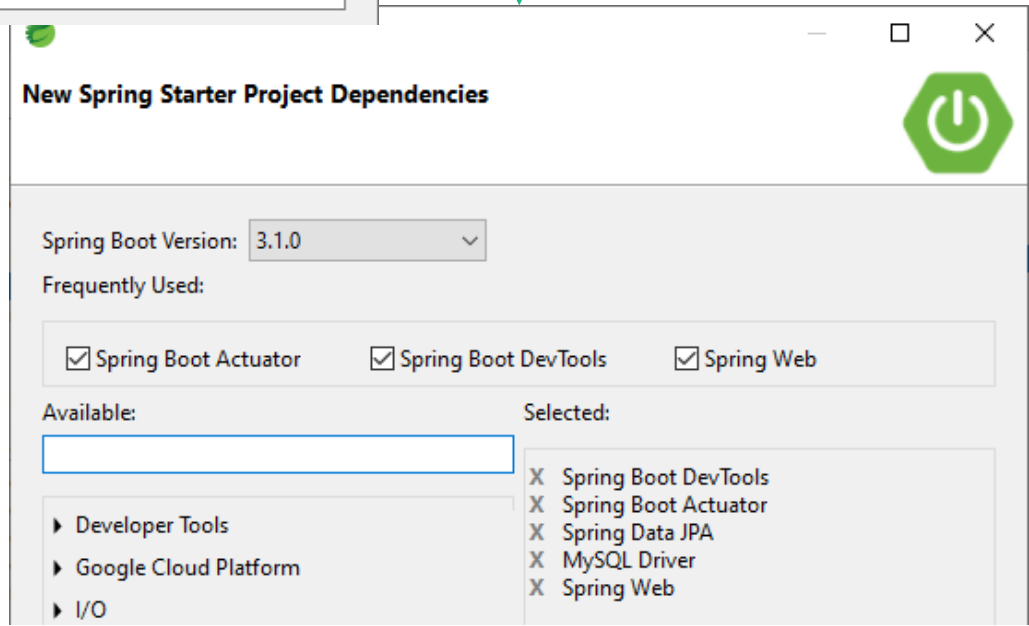
- Spring Web;
- Spring Boot Dev Tools;
- Spring Boot Actuator;
- Spring Data JPA;
- MySQL Driver

Veja no próximo slide


New
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Import...
Export...
Source
Refresh F5
Java Project
Maven Project
Spring Starter Project
Import Spring Getting Started Content
Project...
Package
Class
Interface
Enum
Record
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Other... Ctrl+N

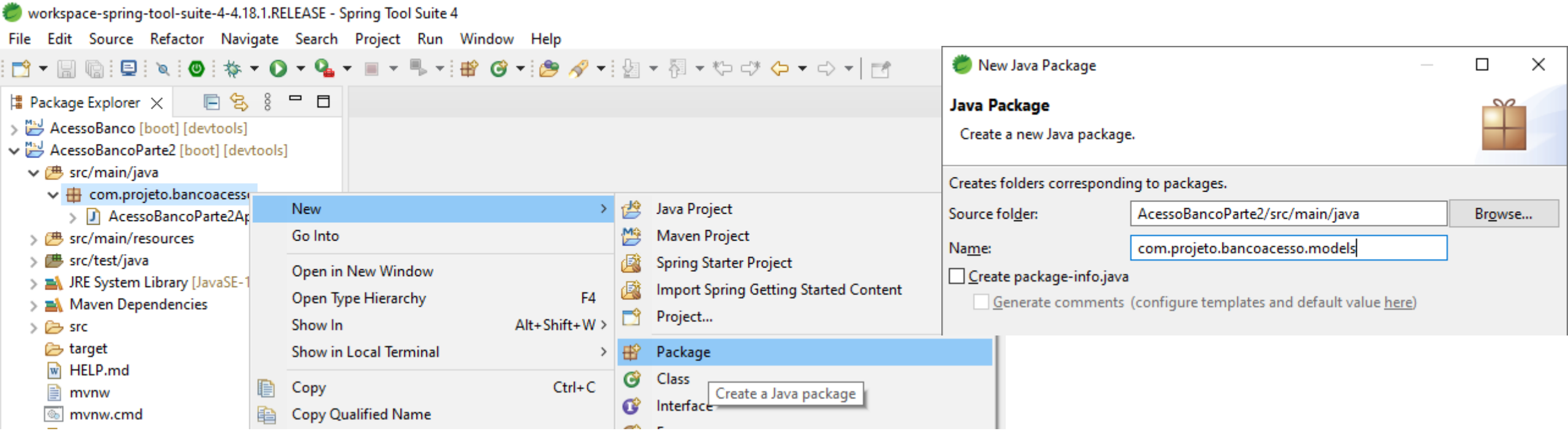

New Spring Starter Project
Service URL https://start.spring.io
Name AcessoBancoParte2
Use default location
Location C:\Users\edilson.jsilva7\Documents\workspace-spring-tool-suite-4-
Browse
Type: Maven
Packaging: Jar
Java Version: 17
Language: Java
Group com.projeto.bancoacesso
Artifact AcessoBancoParte2
Version 0.0.1-SNAPSHOT
Description Projeto String
Package com.projeto.bancoacesso


Select a wizard
Select a wizard
Create new Spring Starter Project
Wizards:
type filter text
General
Git
Gradle
Java
Maven
Plug-in Development
Server
Spring Boot
Import Spring Getting Started Content
Spring Starter Project
User Assistance
< Back
Next >
Finish
Cancel


New Spring Starter Project Dependencies
Spring Boot Version: 3.1.0
Frequently Used:
Spring Boot Actuator
Spring Boot DevTools
Spring Web
Available:
Selected:
Developer Tools
Google Cloud Platform
I/O
X Spring Boot DevTools
X Spring Boot Actuator
X Spring Data JPA
X MySQL Driver
X Spring Web

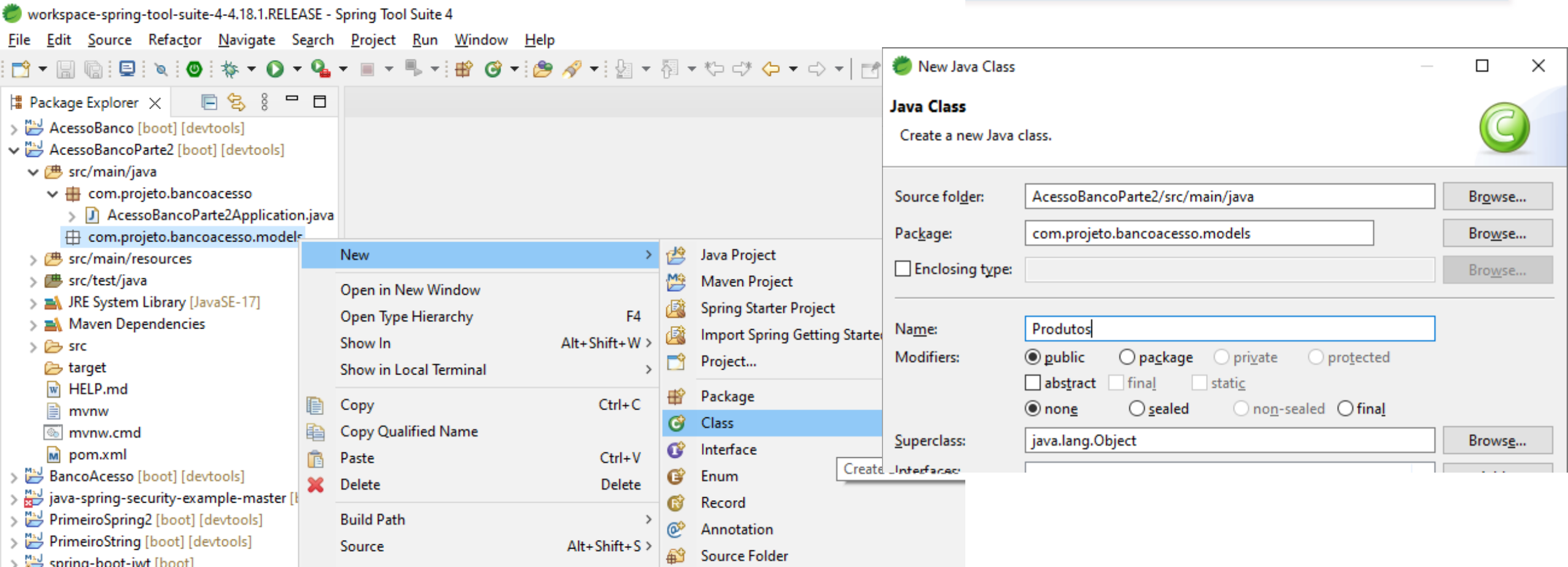
Criação do package models

- Depois de criar o projeto, crie um novo package chamados models no package principal. Veja abaixo



Criação do package models

- No package criado, models, crie uma classe com o nome de Produtos



Criação da classe Produtos

- Adicione o código referente a classe produtos, que deve representar a tabela cliente do banco de dados na camada da aplicação. Note que a classe possui anotações que qualifica cada propriedade desta classe

Produtos.java

```java
package com.projeto.bancoacesso.models;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Produtos {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer idproduto;

    @Column
    private String nomeproduto;

    @Column
    private String descricao;

    @Column
    private String categoria;

    @Column
    private String lote;

    @Column
    private String datafabricacao;

    @Column
    private String datavalidade;

    @Column
    private Double preco;

    @Column
    private String imagemproduto;
```

Criação da classe Produtos

- Adicione o código referente a classe produtos, que deve representar a tabela produtos do banco de dados na camada da aplicação. Note que a classe possui anotações que qualifica cada propriedade desta classe

```
39      public Produtos() {
40      }
41
42      public Produtos(Integer idproduto, String nomeproduto, String descricao, String categoria, String lote,
43              String datafabricacao, String datavalidade, Double preco, String imagemproduto) {
44          this.idproduto = idproduto;
45          this.nomeproduto = nomeproduto;
46          this.descricao = descricao;
47          this.categoria = categoria;
48          this.lote = lote;
49          this.datafabricacao = datafabricacao;
50          this.datavalidade = datavalidade;
51          this.preco = preco;
52          this.imagemproduto = imagemproduto;
53      }
```

Criação da classe Produtos

- Adicione o código referente a classe produtos, que deve representar a tabela produtos do banco de dados na camada da aplicação. Note que a classe possui anotações que qualifica cada propriedade desta classe

```
55    public Integer getIdproduto() {
56        return idproduto;
57    }
58
59    public void setIdproduto(Integer idproduto) {
60        this.idproduto = idproduto;
61    }
62
63    public String getNomeproduto() {
64        return nomeproduto;
65    }
66
67    public void setNomeproduto(String nomeproduto) {
68        this.nomeproduto = nomeproduto;
69    }
70
71    public String getDescricao() {
72        return descricao;
73    }
74
75    public void setDescricao(String descricao) {
76        this.descricao = descricao;
77    }
78
79    public String getCategoria() {
80        return categoria;
81    }
82
83    public void setCategoria(String categoria) {
84        this.categoria = categoria;
85    }
86
87    public String getLote() {
88        return lote;
89    }
```

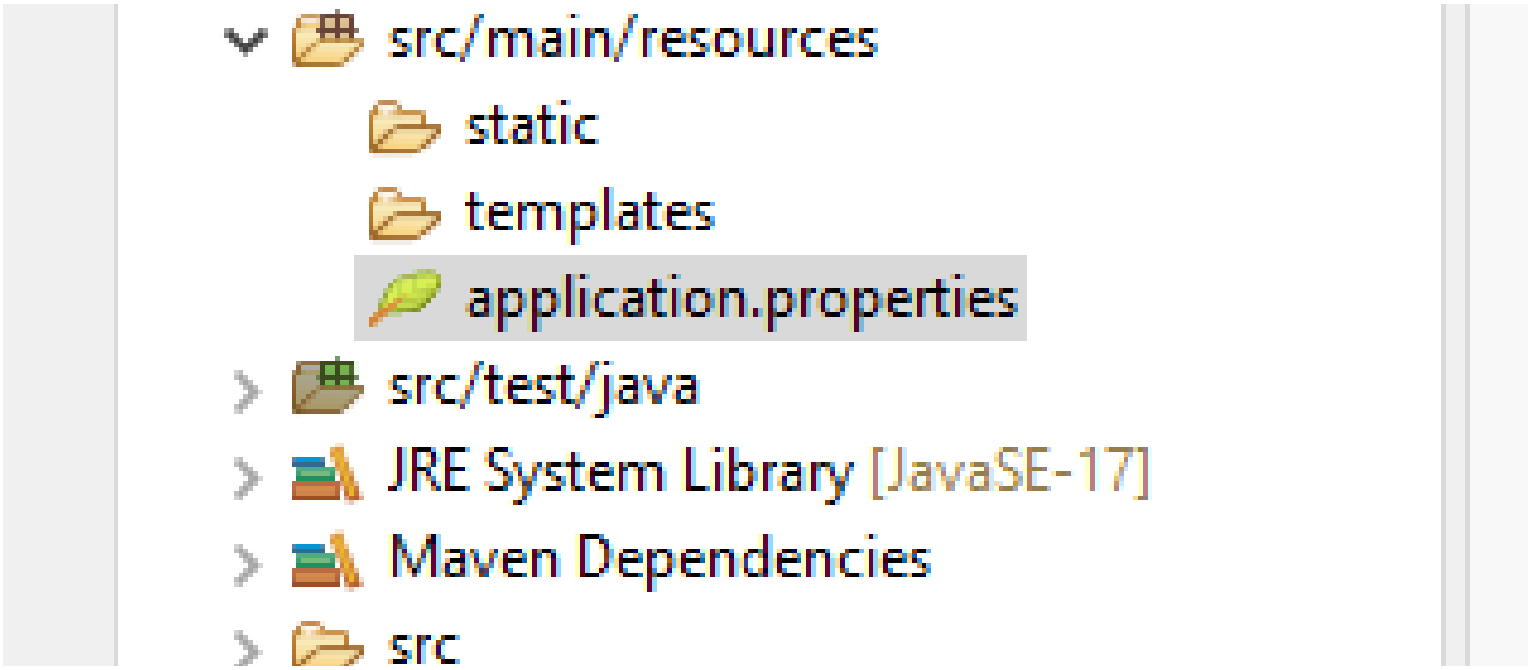
Criação da classe Produtos

- Adicione o código referente a classe produtos, que deve representar a tabela produtos do banco de dados na camada da aplicação. Note que a classe possui anotações que qualifica cada propriedade desta classe

```
86
87      public String getLote() {
88          return lote;
89      }
90
91      public void setLote(String lote) {
92          this.lote = lote;
93      }
94
95      public String getDatafabricacao() {
96          return datafabricacao;
97      }
98
99      public void setDatafabricacao(String datafabricacao) {
100         this.datafabricacao = datafabricacao;
101     }
102
103     public String getDatavalidade() {
104         return datavalidade;
105     }
106
107     public void setDatavalidade(String datavalidade) {
108         this.datavalidade = datavalidade;
109     }
110
111     public Double getPreco() {
112         return preco;
113     }
114
115     public void setPreco(Double preco) {
116         this.preco = preco;
117     }
118
119     public String getImagemproduto() {
120         return imagemproduto;
121     }
122
123     public void setImagemproduto(String imagemproduto) {
124         this.imagemproduto = imagemproduto;
125     }
126 }
```

Adição de configuração

- Agora, vamos configurar os parâmetros da aplicação para acessar o banco de dados.
- Abra o arquivo applicaton.properties



Adição de configuração

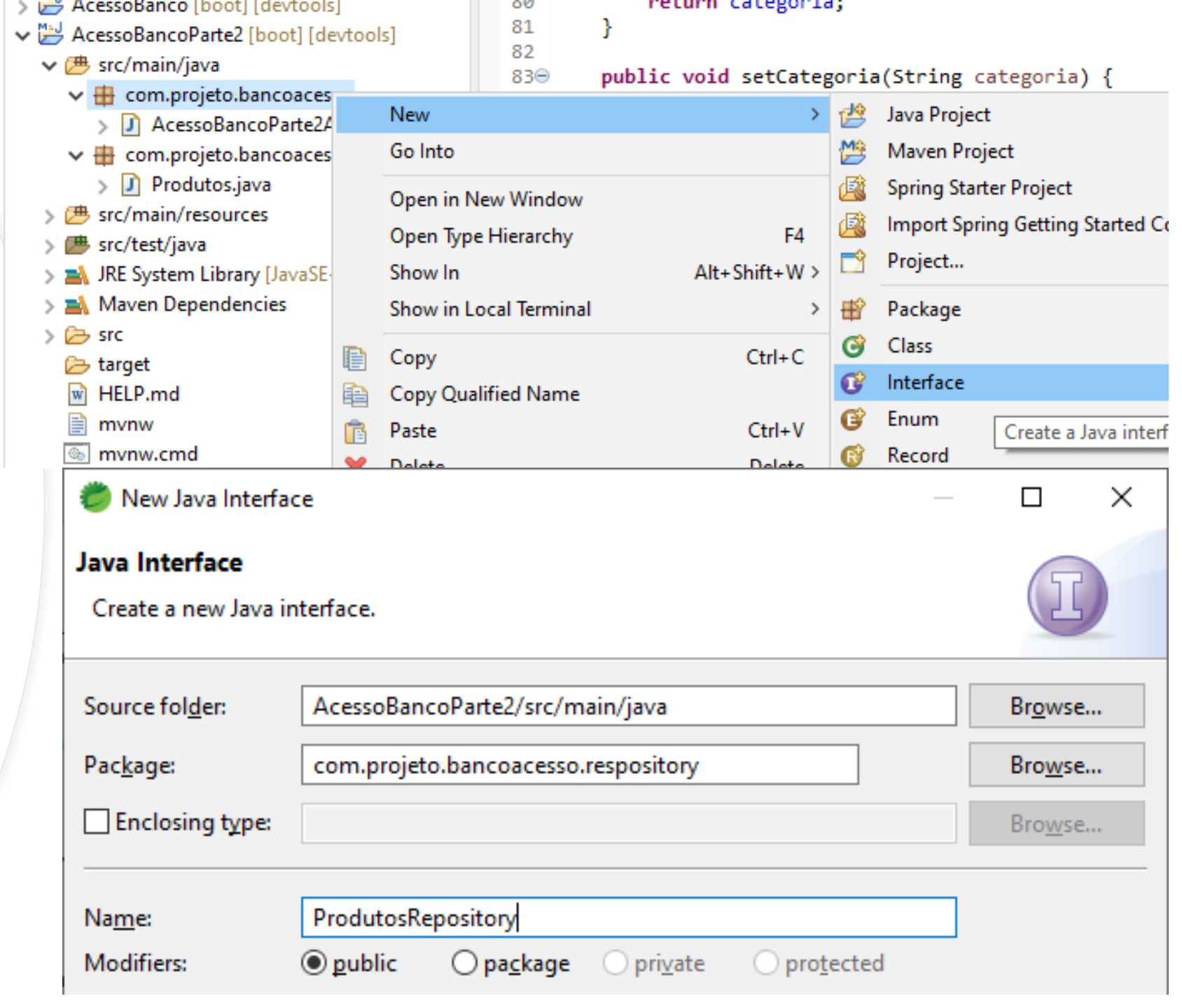
- Adicione as configurações

```
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/produtosdb?usessl=false
spring.datasource.username=root
spring.datasource.password=
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update

#porta do servidor spring
server.port=8080
```

Criando o arquivo de manipulação de dados

- Vamos criar o arquivo chamado ProdutosRepository que será o responsável pela persistência dos dados em banco. Este arquivo é uma interface e deverá ser criado dentro de um package chamado repository



Criando o arquivo de manipulação de dados

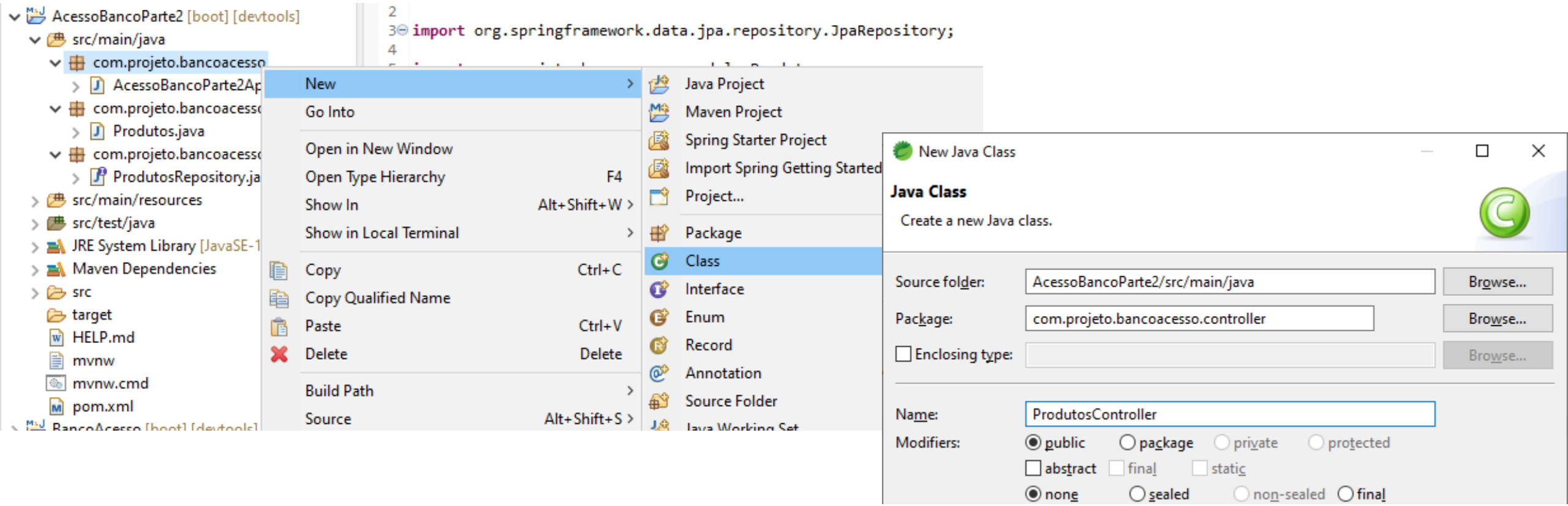
- Adicione o seguinte código a interface criada.

ProdutosRepository.java

```
1  package com.projeto.bancoacesso.respository;
2
3  import org.springframework.data.jpa.repository.JpaRepository;
4
5  import com.projeto.bancoacesso.models.Produtos;
6
7  public interface ProdutosRepository extends JpaRepository<Produtos,Integer>{
8
9  }
10
```

Agora, vamos criar o controlador de dados

- Crie um novo package chamado controller e adicione uma nova classe chamada ProdutosController.



Agora, vamos criar o controlador de dados

- Adicionar o código do controlador do ProdutosController

ProdutosController.java

```java
package com.projeto.bancoacesso.controller;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.projeto.bancoacesso.models.Produtos;
import com.projeto.bancoacesso.respository.ProdutosRepository;



@RestController
public class ProdutosController {


    @Autowired
    private ProdutosRepository produtosRepo;

    @GetMapping("/produtos/listar")
    public List<Produtos> listar(){
        return produtosRepo.findAll();
    }

    @PostMapping("/produtos/cadastrar")
    public String cadastrar(@RequestBody Produtos produtos) {

        produtosRepo.save(produtos);
        return "Cadastrou";

    }
```

Agora, vamos criar o controlador de dados

- Adicionar o código do controlador do ProdutosController

```
39      @PutMapping("/produtos/atualizar/{id}")
40      public String atualizar(@PathVariable Integer id, @RequestBody Produtos produtos) {
41          String msg = "";
42          Optional<Produtos> c = produtosRepo.findById(id);
43
44          if(c.isPresent()) {
45              produtos.setIdproduto(id);
46              produtosRepo.save(produtos);
47              msg = "Produto atualizado";
48          }
49          else {
50              msg = "Produto não encontrado";
51          }
52          return msg;
53      }
54      @DeleteMapping("/produtos/apagar/{id}")
55      public String apagar(@PathVariable Integer id) {
56          String msg = "";
57          Optional<Produtos> c = produtosRepo.findById(id);
58
59          if(c.isPresent()) {
60              produtosRepo.deleteById(id);
61              msg = "Produto apagado";
62          }
63          else {
64              msg = "Produto não localizado";
65          }
66          return msg;
67
68      }
69  }
```

Construindo o banco de dados

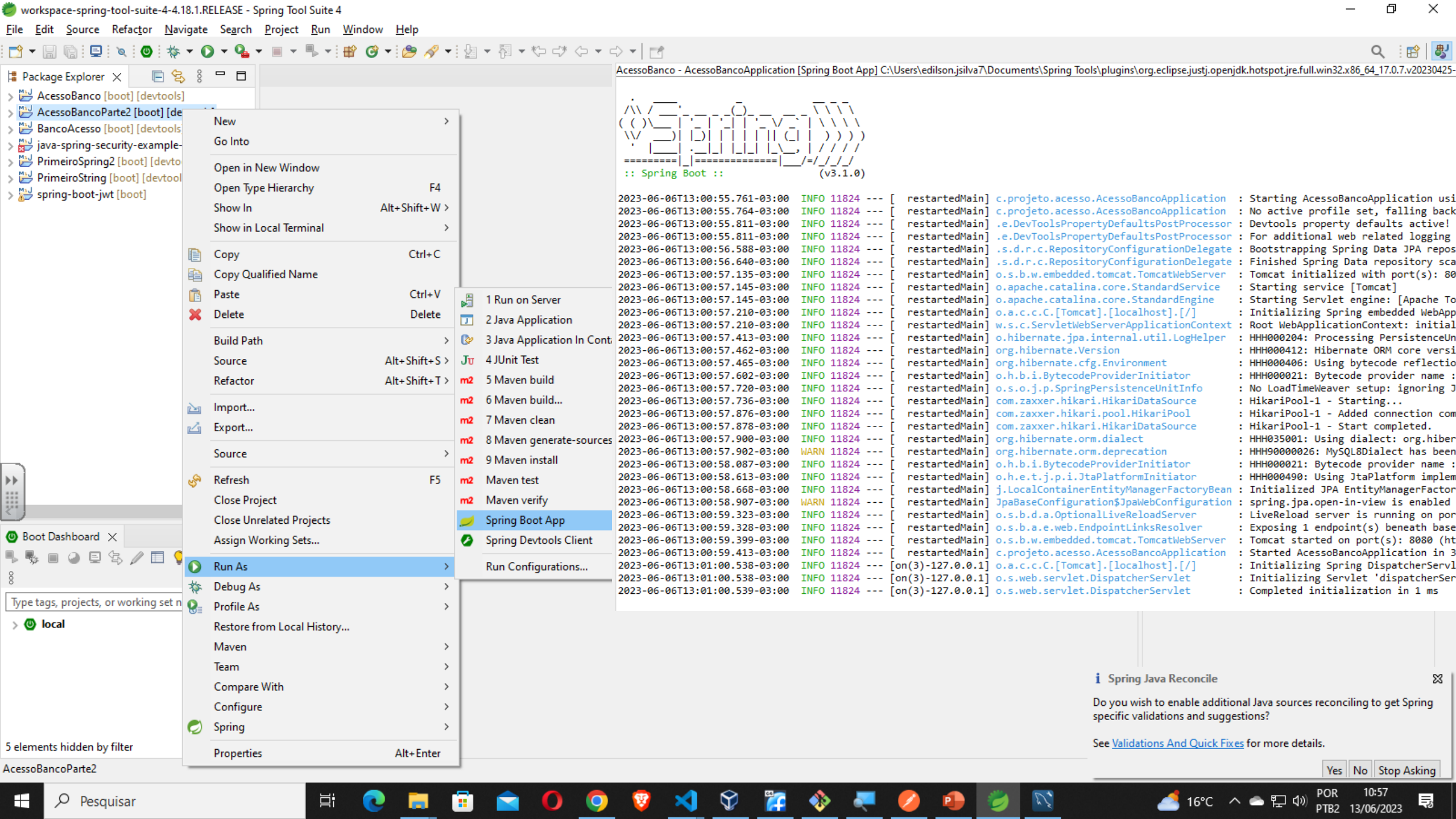
- Agora, vamos criar o banco de dados. Crie conforme ao lado

```
▷ Execute
CREATE DATABASE produtosdb
    DEFAULT CHARACTER SET = 'utf8mb4';

▷ Execute
use produtosdb;
▷ Execute
create table produtos(
    idproduto INT AUTO_INCREMENT PRIMARY KEY,
    nomeproduto VARCHAR(100),
    descricao TEXT,
    categoria VARCHAR(50),
    lote VARCHAR(20),
    datafabricacao VARCHAR(20),
    datavalidade VARCHAR(20),
    preco DECIMAL(10,2),
    imagemproduto VARCHAR(200)
);
```

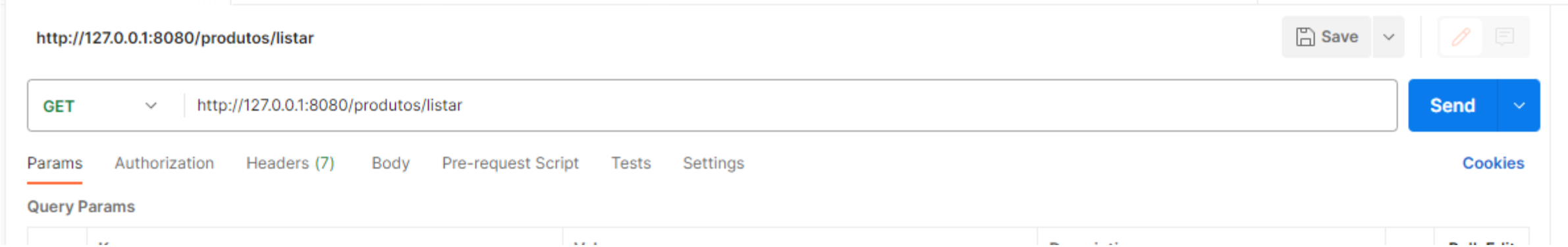
Cadastre alguns Produtos

- Selecione a tabela clicando sobre ela e selecionando a opção Select Rows


workspace-spring-tool-suite-4-4.18.1.RELEASE - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
AcessoBanco [boot] [devtools]
AcessoBancoParte2 [boot] [de
BancoAcesso [boot] [devtools
java-spring-security-example-
PrimeiroSpring2 [boot] [devto
PrimeiroString [boot] [devtool
spring-boot-jwt [boot]
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Source
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Run As
Debug As
Profile As
Restore from Local History...
Maven
Team
Compare With
Configure
Spring
Properties Alt+Enter
1 Run on Server
2 Java Application
3 Java Application In Cont
4 JUnit Test
5 Maven build
6 Maven build...
7 Maven clean
8 Maven generate-sources
9 Maven install
Maven test
Maven verify
Spring Boot App
Spring Devtools Client
Run Configurations...
Boot Dashboard
Type tags, projects, or working set n
local
5 elements hidden by filter
AcessoBancoParte2
AcessoBanco - AcessoBancoApplication [Spring Boot App] C:\Users\edilson.jsilva7\Documents\Spring Tools\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.7.v20230425-
:: Spring Boot :: (v3.1.0)
2023-06-06T13:00:55.761-03:00 INFO 11824 --- [restartedMain] c.projeto.acesso.AcessoBancoApplication : Starting AcessoBancoApplication usi
2023-06-06T13:00:55.764-03:00 INFO 11824 --- [restartedMain] c.projeto.acesso.AcessoBancoApplication : No active profile set, falling back
2023-06-06T13:00:55.811-03:00 INFO 11824 --- [restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active!
2023-06-06T13:00:55.811-03:00 INFO 11824 --- [restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging
2023-06-06T13:00:56.588-03:00 INFO 11824 --- [restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repos
2023-06-06T13:00:56.640-03:00 INFO 11824 --- [restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository sca
2023-06-06T13:00:57.135-03:00 INFO 11824 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 80
2023-06-06T13:00:57.145-03:00 INFO 11824 --- [restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-06-06T13:00:57.145-03:00 INFO 11824 --- [restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache To
2023-06-06T13:00:57.210-03:00 INFO 11824 --- [restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApp
2023-06-06T13:00:57.210-03:00 INFO 11824 --- [restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initial
2023-06-06T13:00:57.413-03:00 INFO 11824 --- [restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUn
2023-06-06T13:00:57.462-03:00 INFO 11824 --- [restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core versi
2023-06-06T13:00:57.465-03:00 INFO 11824 --- [restartedMain] org.hibernate.cfg.Environment : HHH000406: Using bytecode reflectio
2023-06-06T13:00:57.602-03:00 INFO 11824 --- [restartedMain] o.h.b.i.BytecodeProviderInitiator : HHH000021: Bytecode provider name :
2023-06-06T13:00:57.720-03:00 INFO 11824 --- [restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring J
2023-06-06T13:00:57.736-03:00 INFO 11824 --- [restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-06-06T13:00:57.876-03:00 INFO 11824 --- [restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com
2023-06-06T13:00:57.878-03:00 INFO 11824 --- [restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-06-06T13:00:57.900-03:00 INFO 11824 --- [restartedMain] org.hibernate.orm.dialect : HHH035001: Using dialect: org.hiber
2023-06-06T13:00:57.902-03:00 WARN 11824 --- [restartedMain] org.hibernate.orm.deprecation : HHH90000026: MySQL8Dialect has been
2023-06-06T13:00:58.087-03:00 INFO 11824 --- [restartedMain] o.h.b.i.BytecodeProviderInitiator : HHH000021: Bytecode provider name :
2023-06-06T13:00:58.613-03:00 INFO 11824 --- [restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implem
2023-06-06T13:00:58.668-03:00 INFO 11824 --- [restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactor
2023-06-06T13:00:58.907-03:00 WARN 11824 --- [restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled
2023-06-06T13:00:59.323-03:00 INFO 11824 --- [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on por
2023-06-06T13:00:59.328-03:00 INFO 11824 --- [restartedMain] o.s.b.a.e.web.EndpointLinksResolver : Exposing 1 endpoint(s) beneath base
2023-06-06T13:00:59.399-03:00 INFO 11824 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (ht
2023-06-06T13:00:59.413-03:00 INFO 11824 --- [restartedMain] c.projeto.acesso.AcessoBancoApplication : Started AcessoBancoApplication in 3
2023-06-06T13:01:00.538-03:00 INFO 11824 --- [on(3)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServl
2023-06-06T13:01:00.538-03:00 INFO 11824 --- [on(3)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherSer
2023-06-06T13:01:00.539-03:00 INFO 11824 --- [on(3)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
Spring Java Reconcile
Do you wish to enable additional Java sources reconciling to get Spring specific validations and suggestions?
See Validations And Quick Fixes for more details.
Yes No Stop Asking
Pesquisar
16°C
POR PTB2
10:57 13/06/2023

Teste a rota no postman

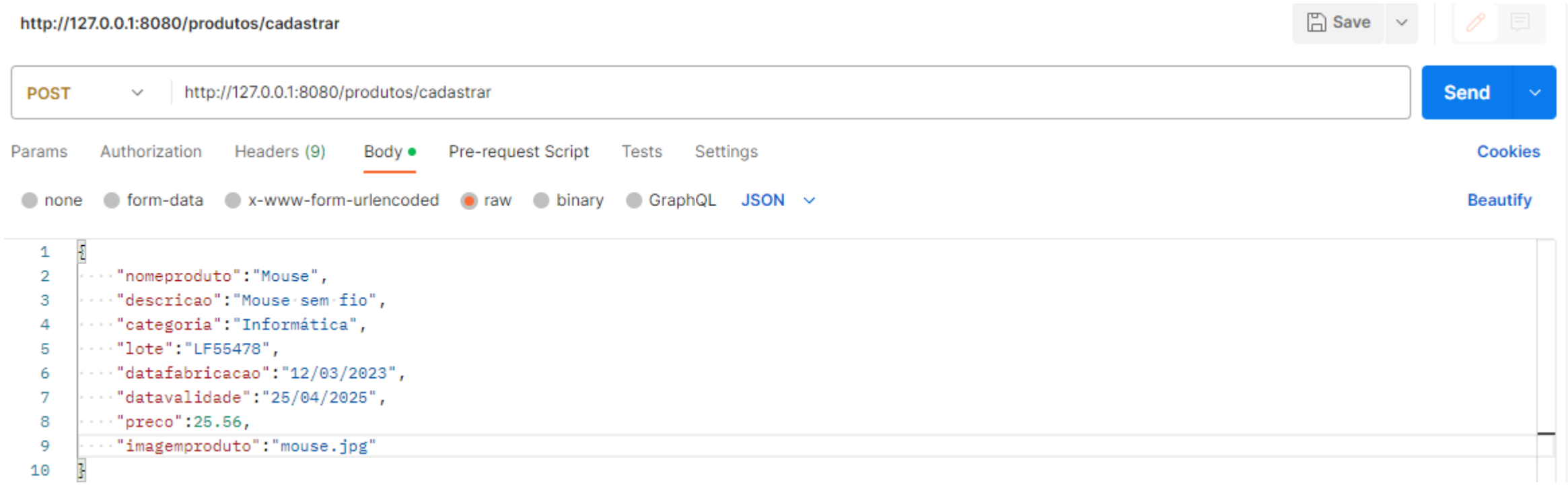
- Vamos acessar o postman e com o verbo de solicitação GET digite a seguinte url:
- http://127.0.0.1:8080/produtos/listar
- E Clique no botão Send



Teste a rota no postman

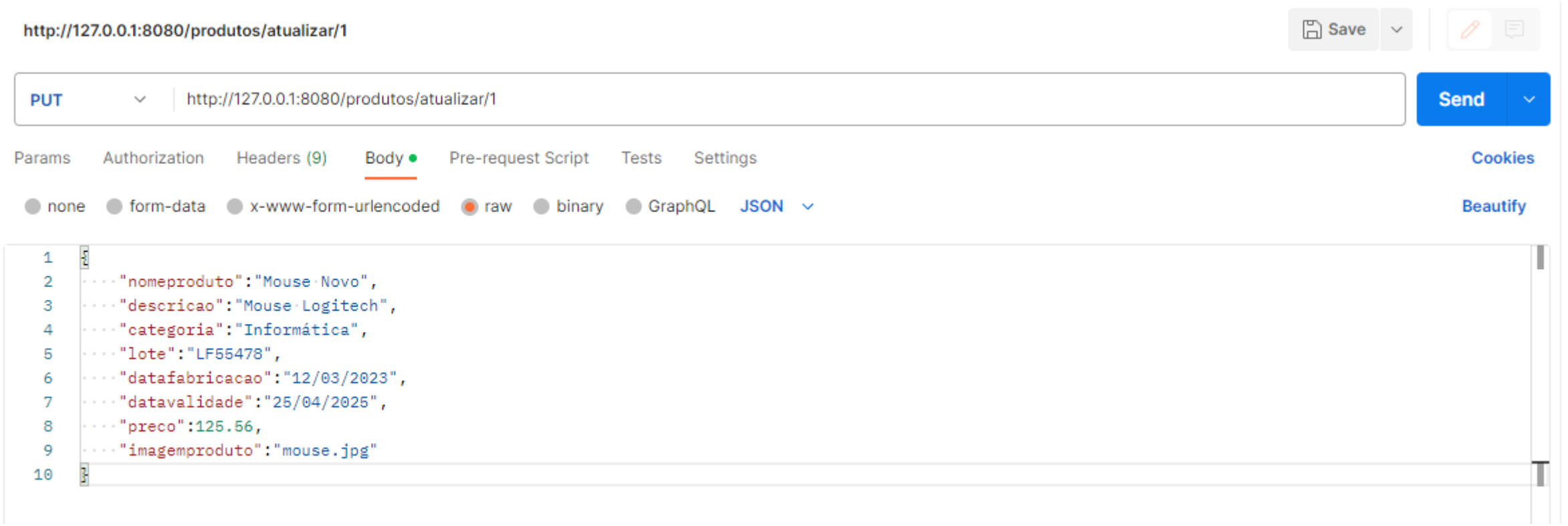
- Vamos acessar o postman e com o verbo de solicitação POST digite a seguinte url:
- http://127.0.0.1:8080/produtos/cadastrar

- Passe os dados no corpo como segue e clique em Send



Teste a rota no postman

- Vamos acessar o postman e com o verbo de solicitação POST digite a seguinte url:
- http://127.0.0.1:8080/produtos/atualizar
- Passe os dados no corpo como segue e clique em Send



Enviar

- Publique o projeto no git e envie a url para o email
- edilsonbackend@gmail.com