1. Configurando o ambiente de desenvolvimento

A IDE Eclipse IDE for Enterprise Java and Web Developers Ferramentas para desenvolvedores que trabalham com aplicativos Java e Web, incluindo um IDE Java, ferramentas para JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA e Data Tools, Maven e Gradle, Git e muito mais. Este pacote inclui:

• Plataforma de ferramentas de dados

• Integração Git para Eclipse

• Ferramentas de Desenvolvimento Java Eclipse

• Ferramentas do desenvolvedor Eclipse Java EE

• Integração Maven para Eclipse

• Lista de Tarefas Mylyn

• Ambiente de Desenvolvimento de Plug-in Eclipse

* 1. Procedimento para Configurar o ambiente de desenvolvimento

1. link: <https://www.eclipse.org/downloads/packages/release/2021-12/r/eclipse-ide-enterprise-java-and-web-developers>
2. Fazer o download do ***Eclipse IDE for Enterprise Java and Web Developers*** para Windows:

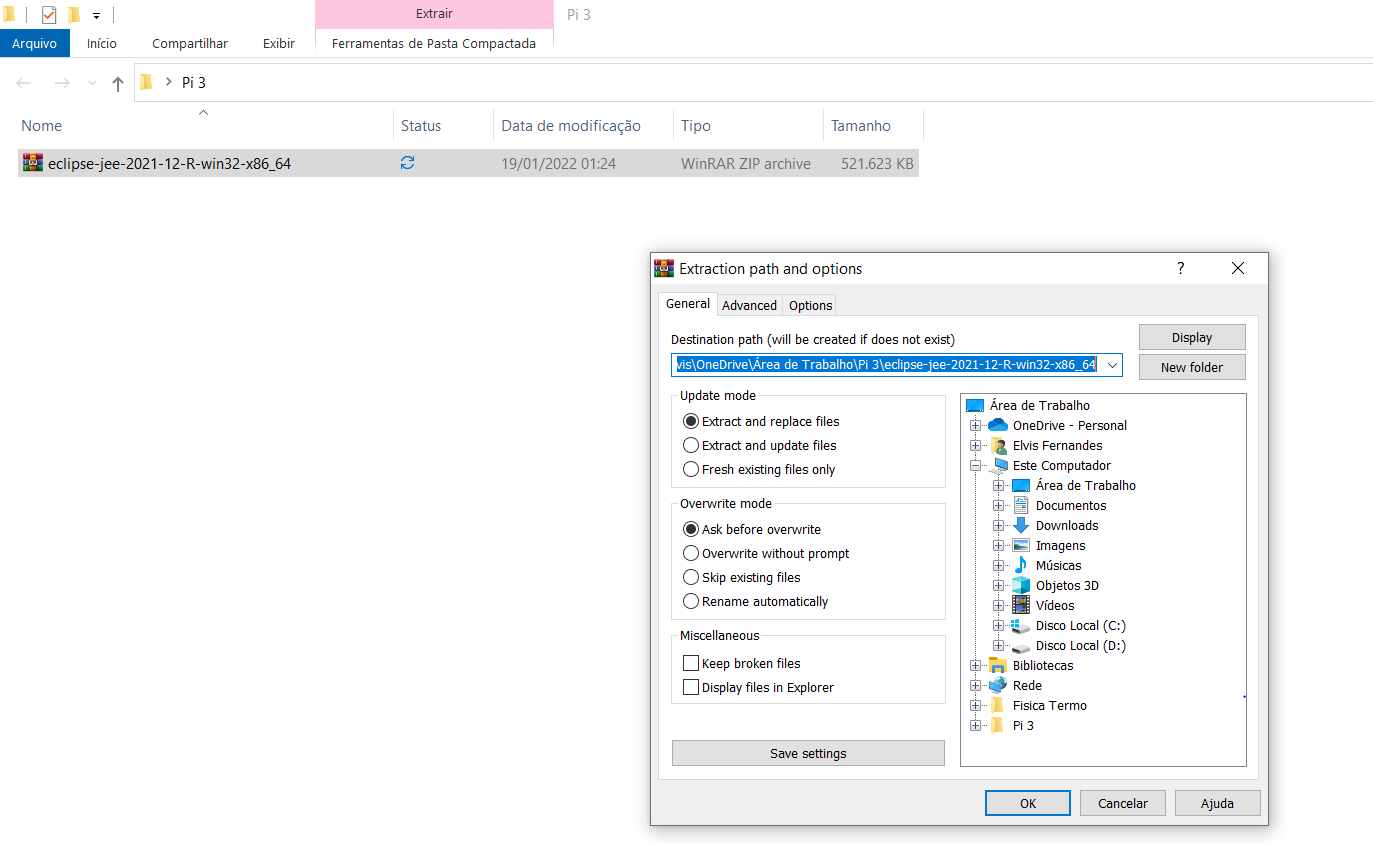


1. Clicar no Botão Download

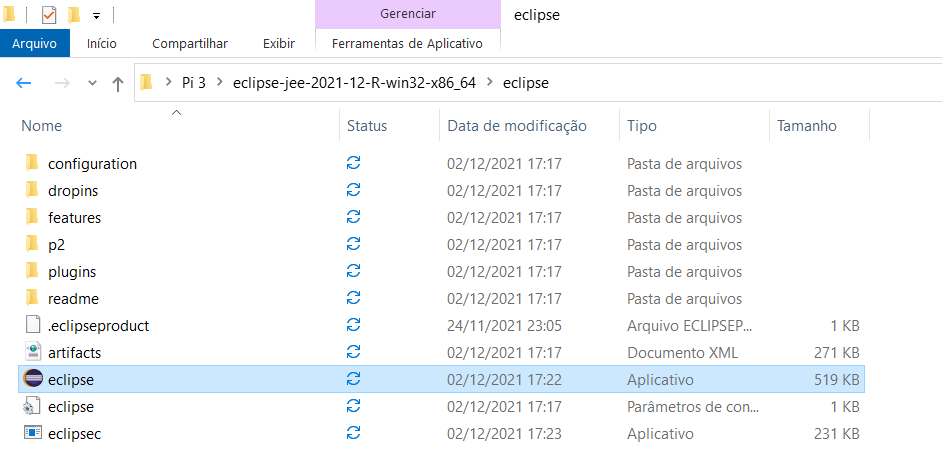




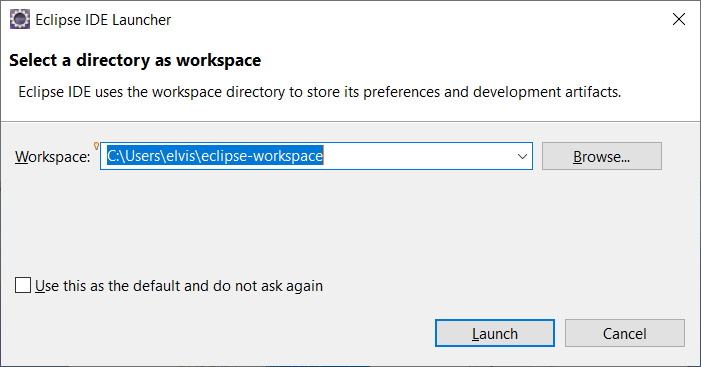
1. Instalar o Software
   1. Extrair os arquivos zipados



* 1. Clicar no aplicativo Eclipse



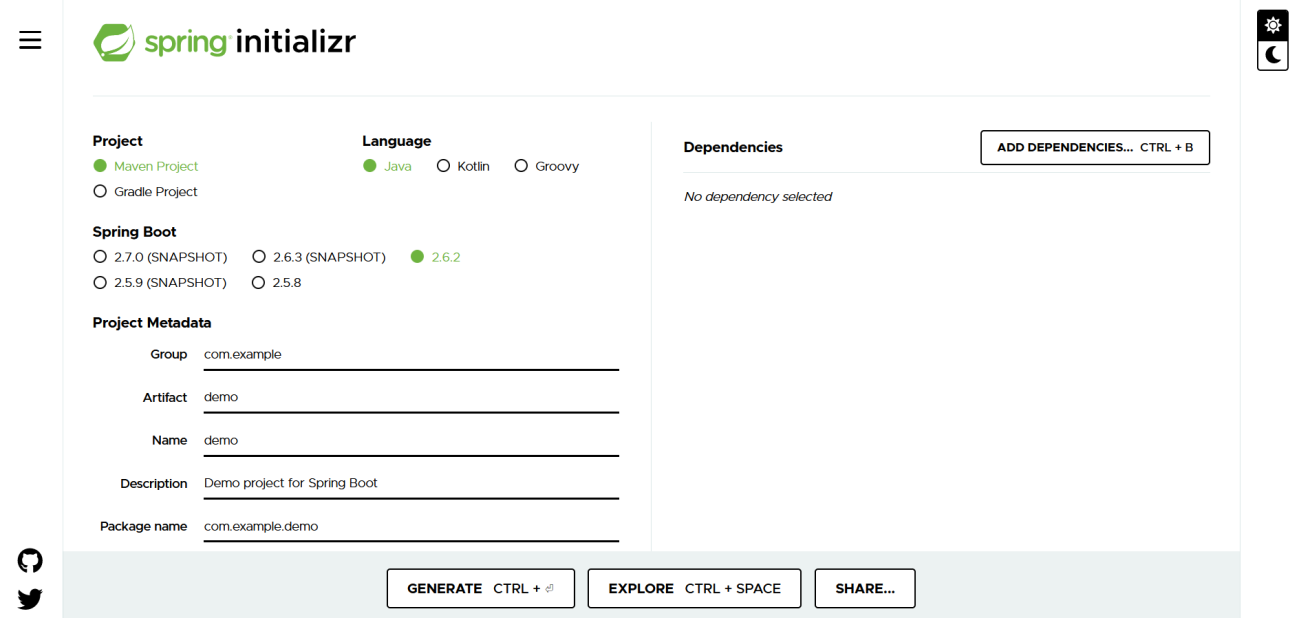
* 1. Configurar ***Workspace*** (Clicar no botão ***Launch***)



* 1. Validação da Configuração do ambiente de desenvolvimento



1. Criação do Projeto
   1. Acessar o link: [**https://start.spring.io/**](https://start.spring.io/)

****

* 1. Requisitos de Projeto:
* **Padrão: Maven Project**
* **Linguagem: Java**
* **Spring Boot: 2.62**
* **Project Metada**

**Group: com.temperatura**

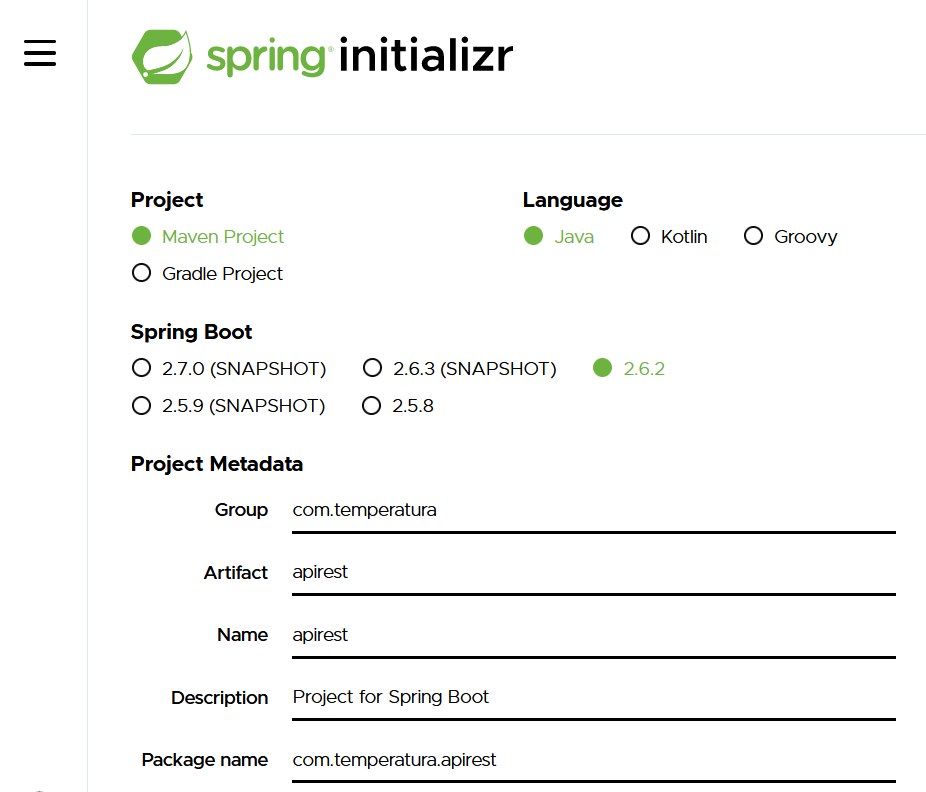
**Artifact: apirest**

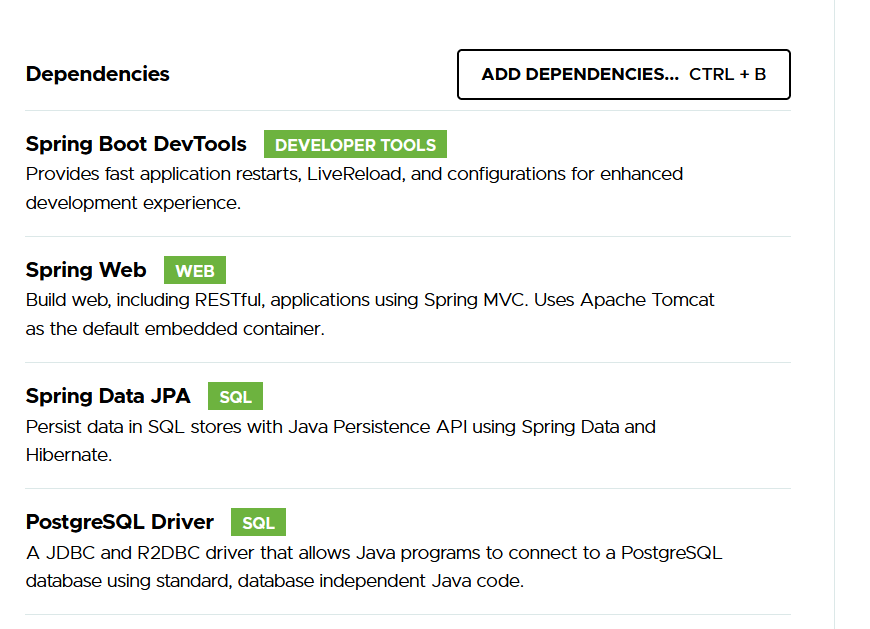
**Name: apirest**

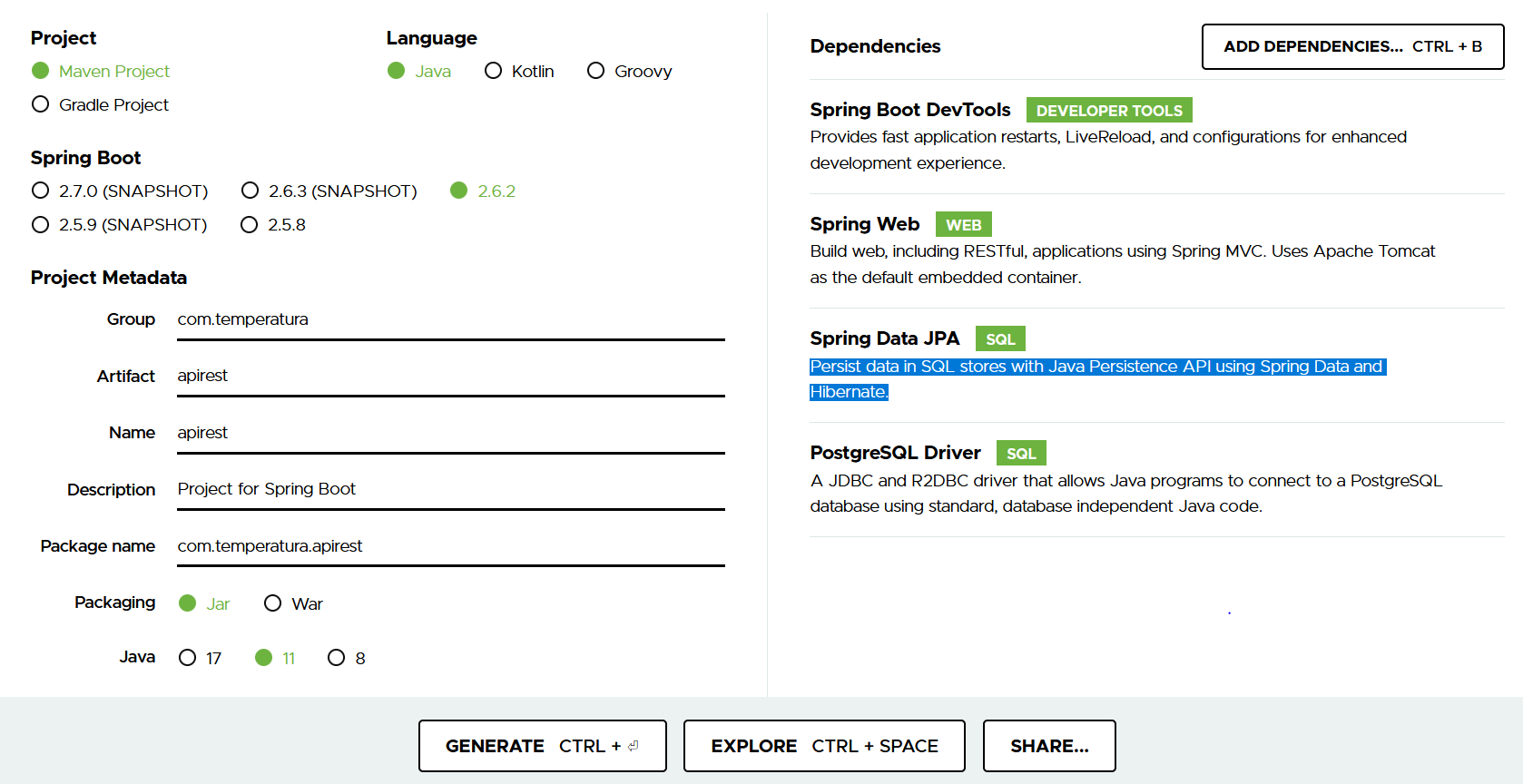
**Description: Project for Spring Boot**

**Package name: com.temperatura.apirest**

* **Packging: Jar**
* **Java: 11**
  1. Dependências:
* ***Spring Boot DevTools***: Fornece reinicializações rápidas de aplicativos, LiveReload e configurações para uma experiência de desenvolvimento aprimorada.
* ***Spring Web WebBuild web:*** Criar aplicativos web, incluindo RESTful, usando Spring MVC. Usa o Apache Tomcat como o contêiner incorporado padrão.
* ***Spring Data JPA***: Persistir dados em armazenamentos SQL com Java Persistence API usando Spring Data e Hibernate.
* ***PostgreSQL Driver***: Um driver JDBC e R2DBC que permite que programas Java se conectem a um banco de dados PostgreSQL usando código Java padrão independente de banco de dados..

****

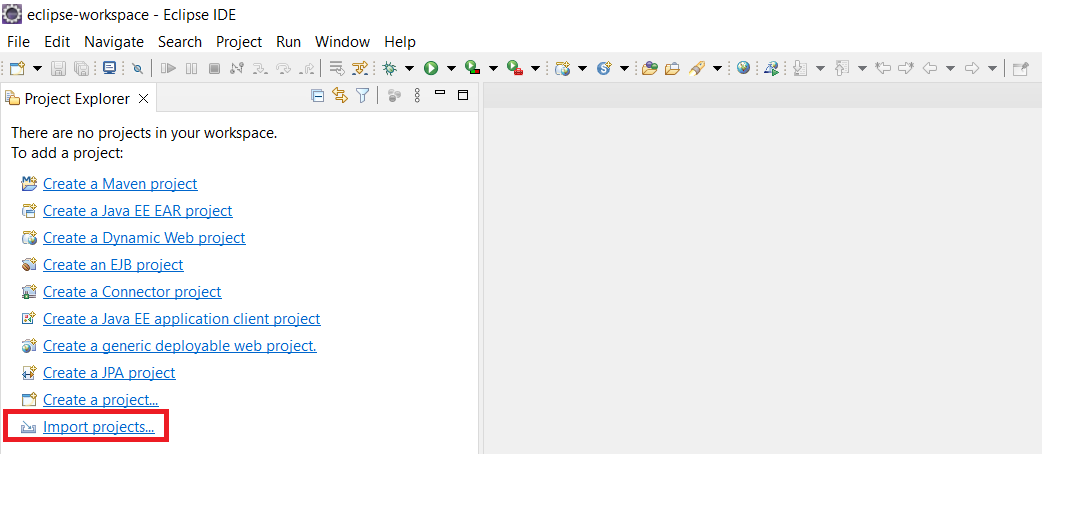
****



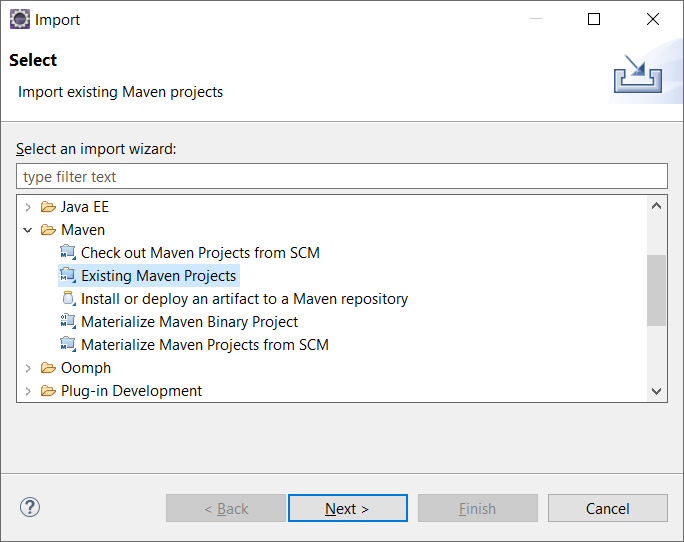
* 1. Gerar o arquivo ***Maven*** (Clicar no botão ***GENERATE***)



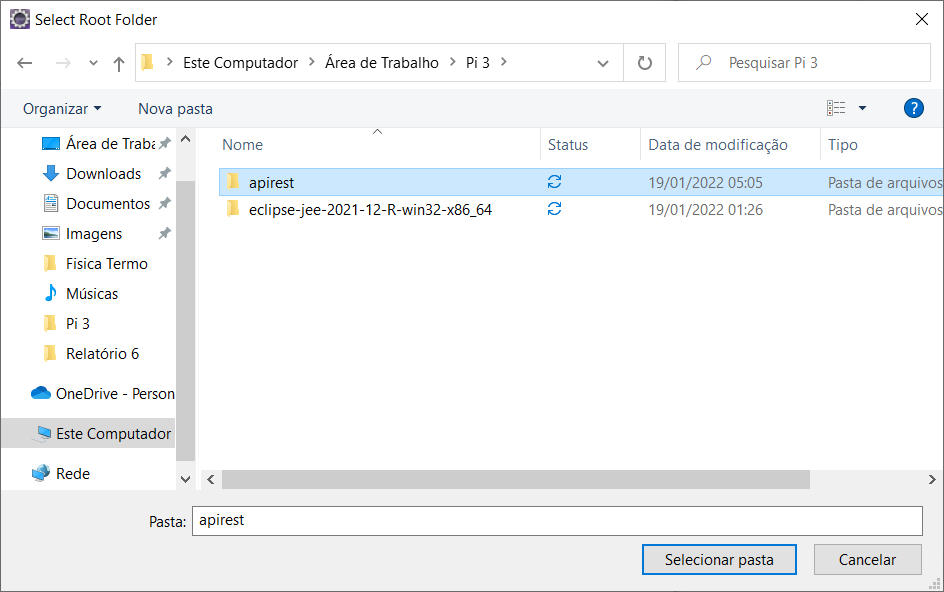
* 1. Importar o arquivo ***Maven*** gerado (apirest)
* Clicar em ***Import projects***



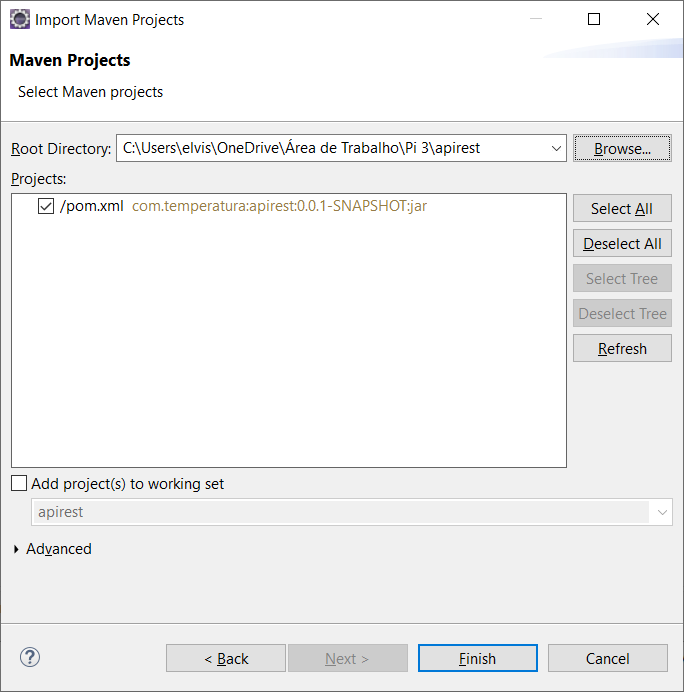
* Selecionar ***Maven > Existing Maven Projects***



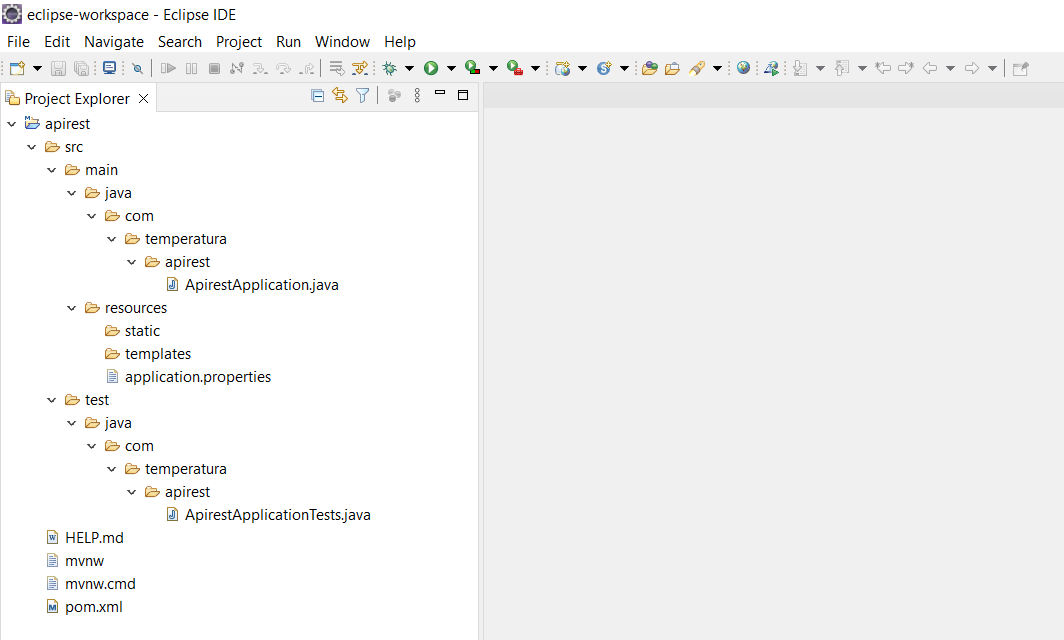
* Clicar em ***Next***
* Selecionar a pasta do o arquivo ***Maven*** (gerado e extraído)

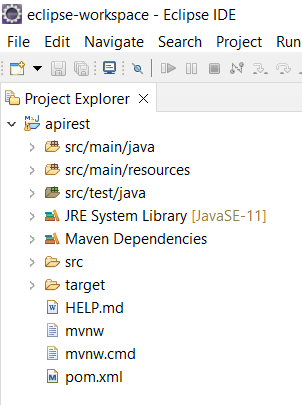


* Clicar em ***Finish***



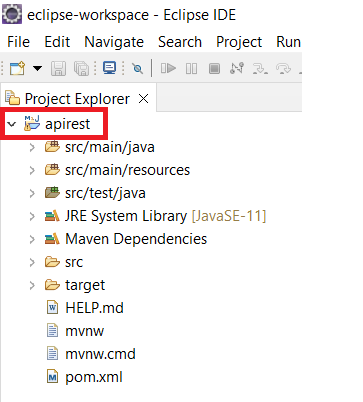
2.6 Estrutura do arquivo ***Maven*** Base



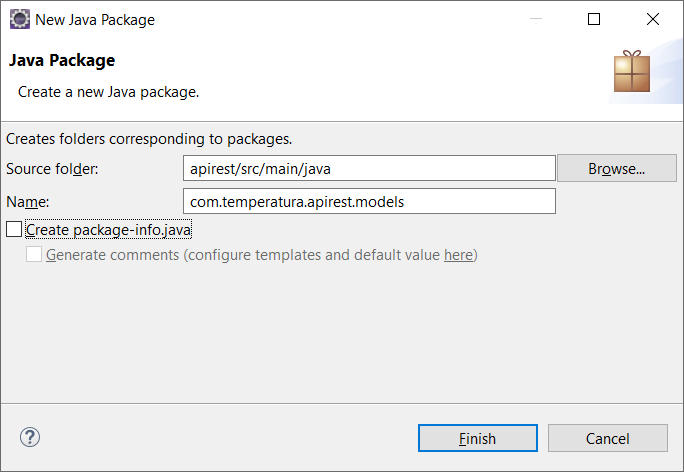


1. Desenvolvimento do Projeto
   1. Criação do Pacote Models

* Clicar com o botão direito do mouse em ***apirest > New > Package***



* Source folder: **apirest/src/main/Java**
* Name: **com.temperatura.apirest.models**
* Clicar em ***Finish***

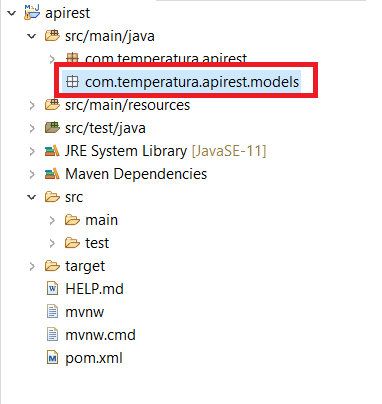


* Validação da criação do Models

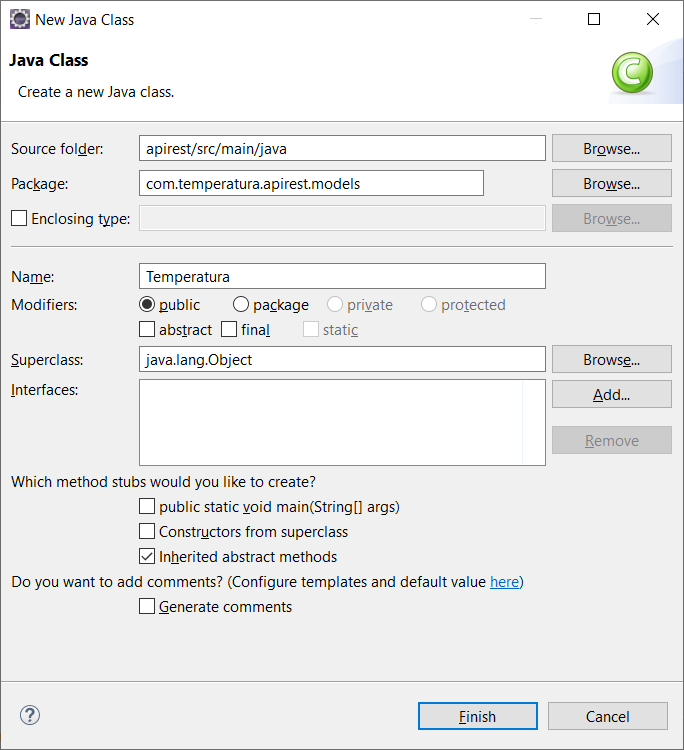


* 1. Criação da classe Temperatura.java dentro do pacote.models
* Clicar com o botão direito do mouse em:

***com.temperatura.apirest.models > New > Class***



* Name: **Temperatura**
* Clicar em ***Finish***



* Validação da criação do pacote Models

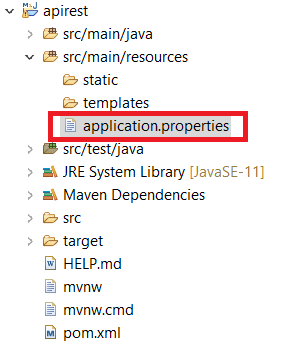


* 1. Desenvolvimento do Código da Classe Temperatura

|  |
| --- |
| **package** com.temperatura.apirest.models;  **import** java.io.Serializable;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.Table;  @Entity  @Table(name = "TB\_TEMPERATURA")  **public** **class** Temperatura **implements** Serializable {  /\*\*  \*  \*/  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  @Id  @GeneratedValue(strategy = GenerationType.***AUTO***)  **private** **long** idDispositivo;  **private** String valorSensor;  **public** **long** getIdDispositivo() {  **return** idDispositivo;  }  **public** **void** setIdDispositivo(**long** idDispositivo) {  **this**.idDispositivo = idDispositivo;  }  **public** String getValorSensor() {  **return** valorSensor;  }  **public** **void** setValorSensor(String valorSensor) {  **this**.valorSensor = valorSensor;  }  } |

4) **Configurar o banco de dados no arquivo *application.properties***

* Abrir o arquivo ***application.properties***

****

* 1. Desenvolvimento do Código do Arquivo de Banco de Dados

|  |
| --- |
| spring.jpa.properties.hibernate.jdbc.lob.non\_contextual\_creation=true  #Banco local - Elvis  spring.datasource.url= jdbc:postgresql://localhost:5432/temperatura\_apirest  spring.datasource.username=postgres  spring.datasource.password=banco123  spring.jpa.hibernate.ddl-auto=update |

Em que:

* Nome do banco de dados: ***temperatura\_apirest***
* Usuário do banco de dados: ***postgres***
* Senha do banco de dados: ***banco123***

**5) Instalação do Banco de Dados PostgresSQL**

* **Acessar o link:** [**https://www.postgresql.org/**](https://www.postgresql.org/)
* **Clicar em Download**



* Clicar em ***Windows***



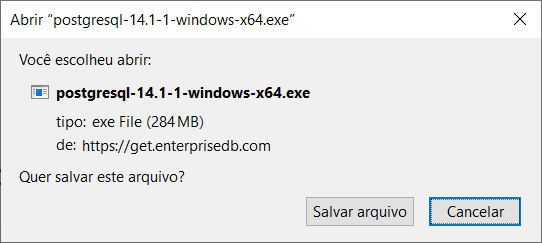
* Clicar em ***Download the installer***

****

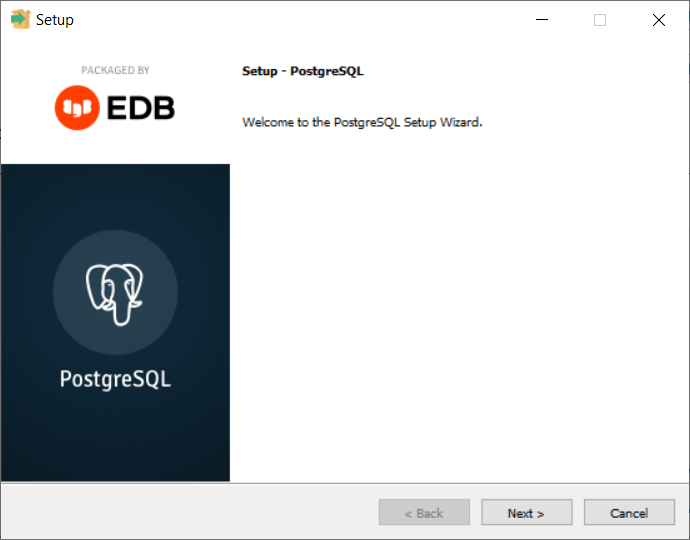
* Selecionar a versão ***14.1*** para ***Windows 64 bits***

****

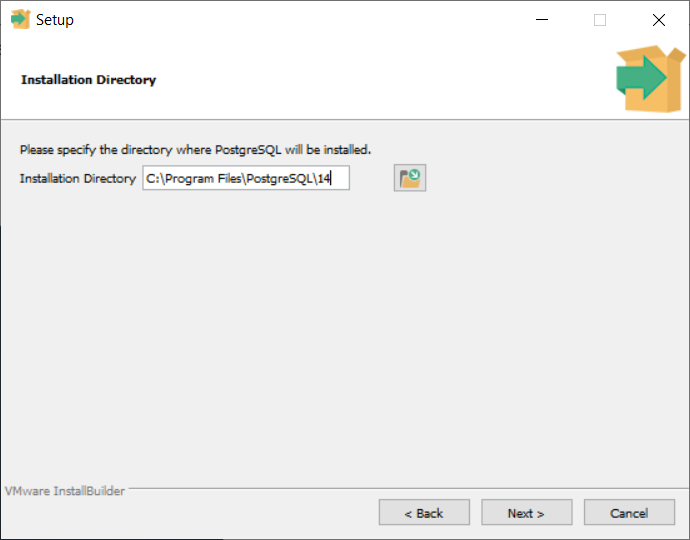
* Clicar em ***Salvar arquivo***



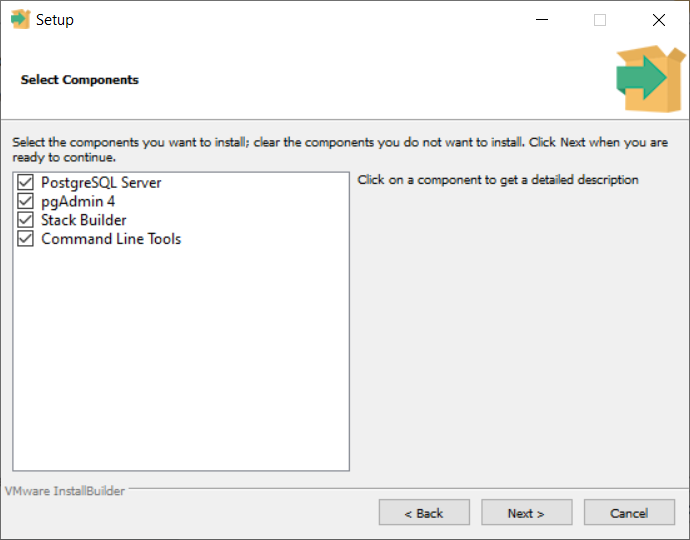
* Abrir o aplicativo **postgresql-14.1-1-windows-x64**
* **Clicar em Next**



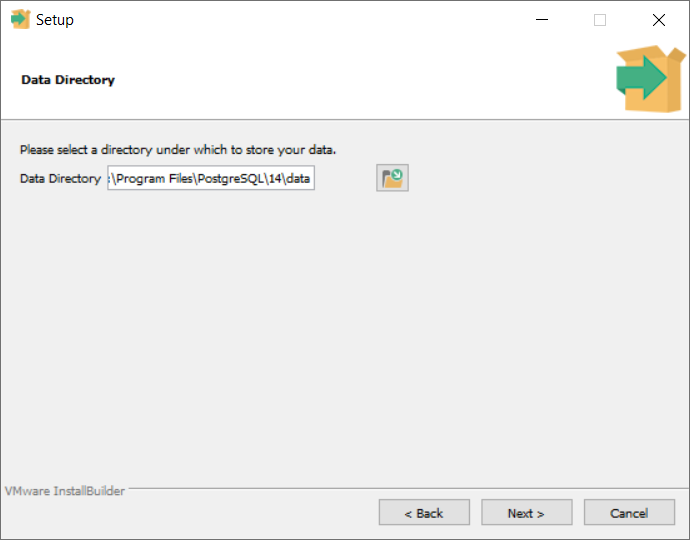
* Escolher o diretório de Instalação
* Clicar em ***Next***



* Selecionar os componentes
* Clicar em *Next*



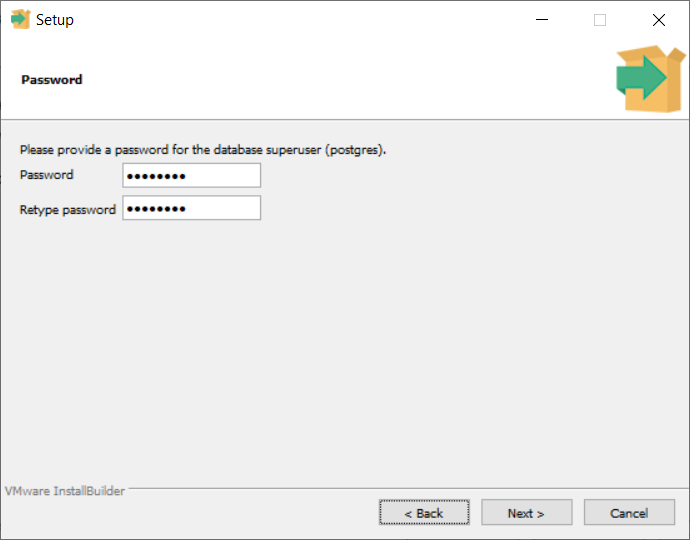
* Escolher o diretório de dados
* Clicar em ***Next***

****

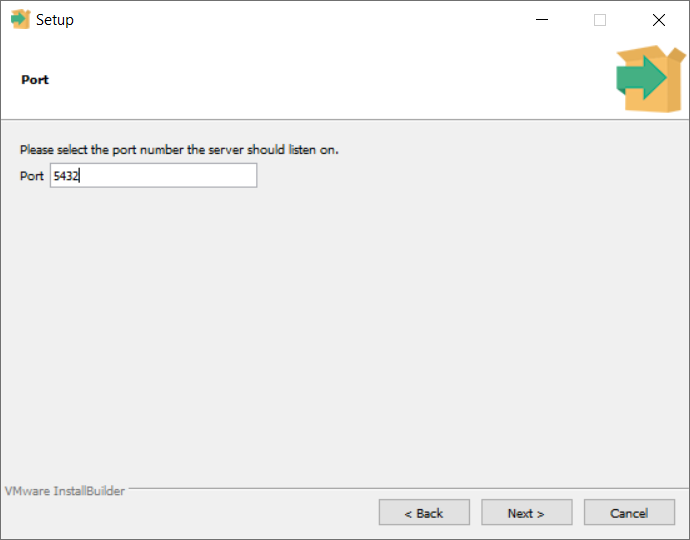
* **Digitar uma senha para o superusuário postgres**

****

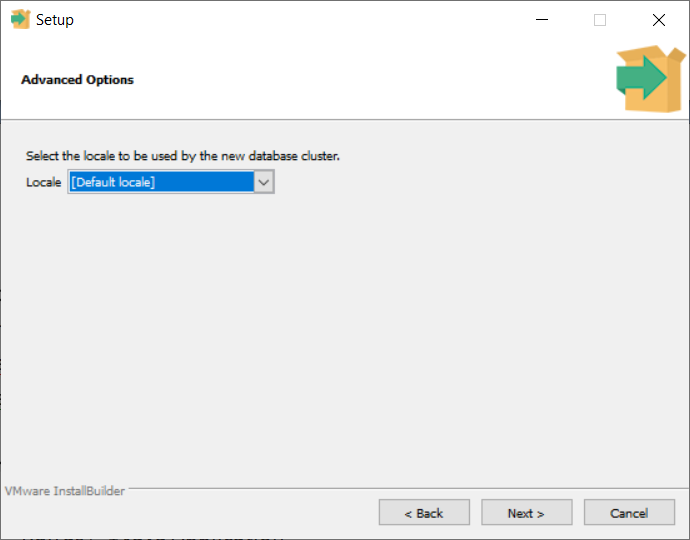
* **Senha escolhida: banco123**
* Clicar no botão ***Next***

****

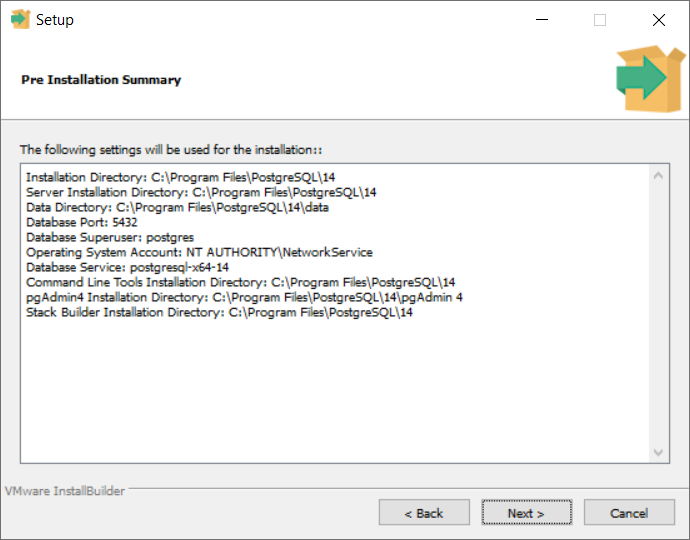
* Selecionar o número da Porta
* Clicar em ***Next***

****

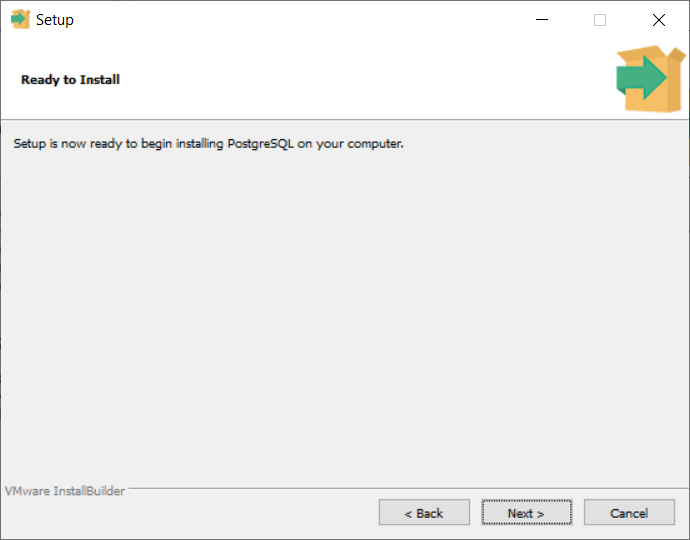
* Selecionar o local do banco de dados (**cluster**)
* ***Default Locale***
* Clicar em ***Next***

****

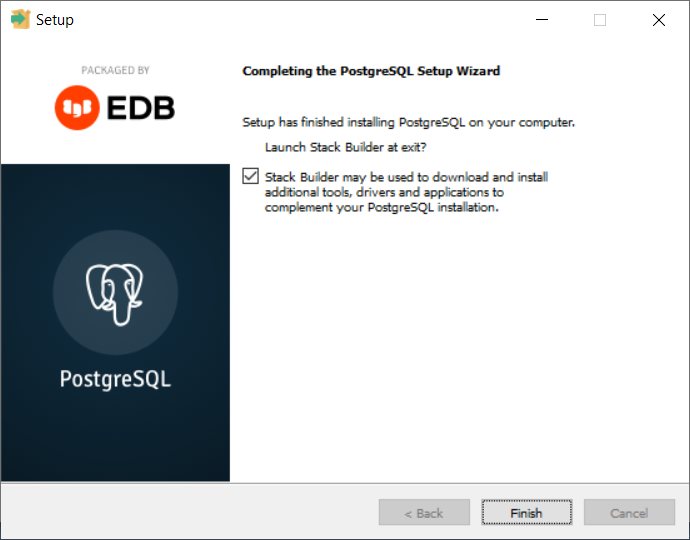
* Clicar em ***Next***

****

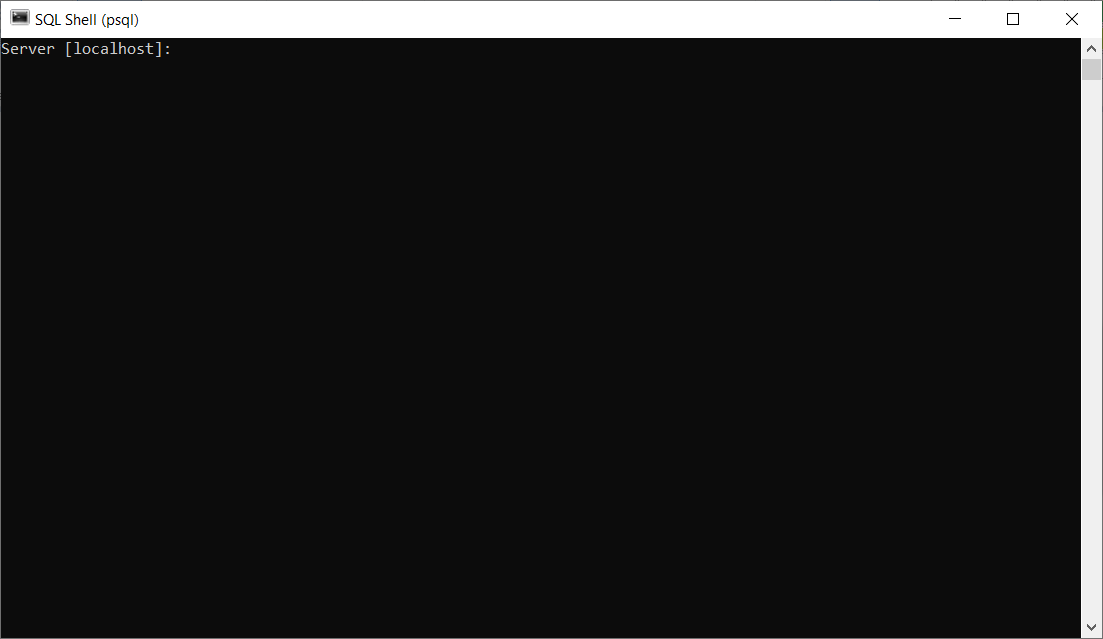
* Clicar em ***Next***

****

* Clicar em ***Finish***



* 1. **Criação do Banco de Dados Local**
* Abrir o SQL Shell (psql)



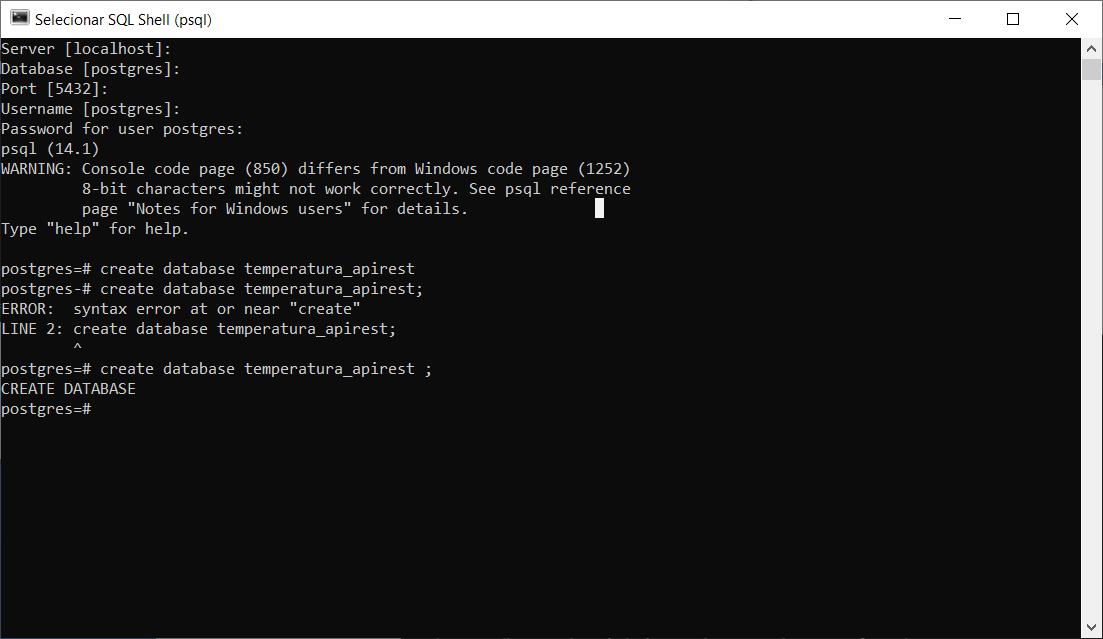
* Configurar

1. Apertar Enter (no teclado)
2. Apertar Enter (no teclado)
3. Apertar Enter (no teclado)
4. Apertar Enter (no teclado)
5. Digitar a senha banco123
6. Apertar Enter (no teclado)

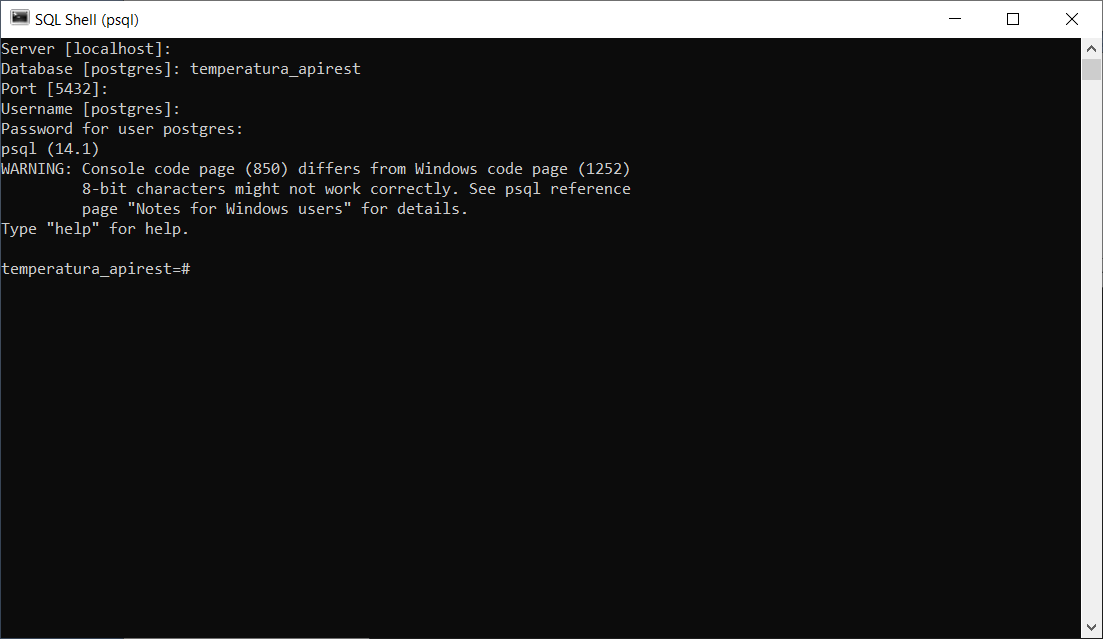


* Criar o banco de dados com o seguinte comando:

***create database temperatura\_apirest;***

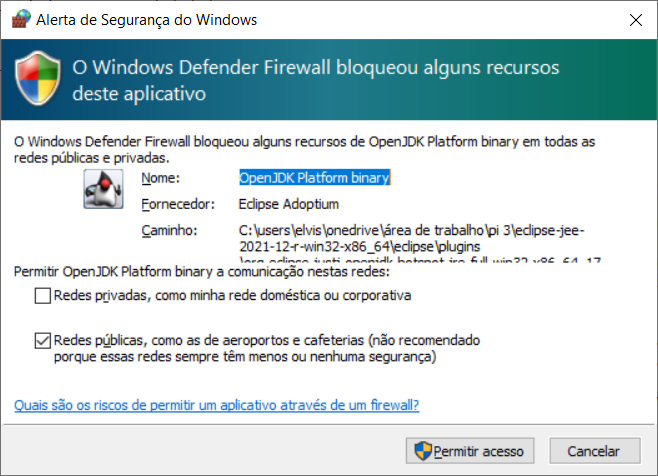
******

* Acessar o banco de dados temperatura\_apirest

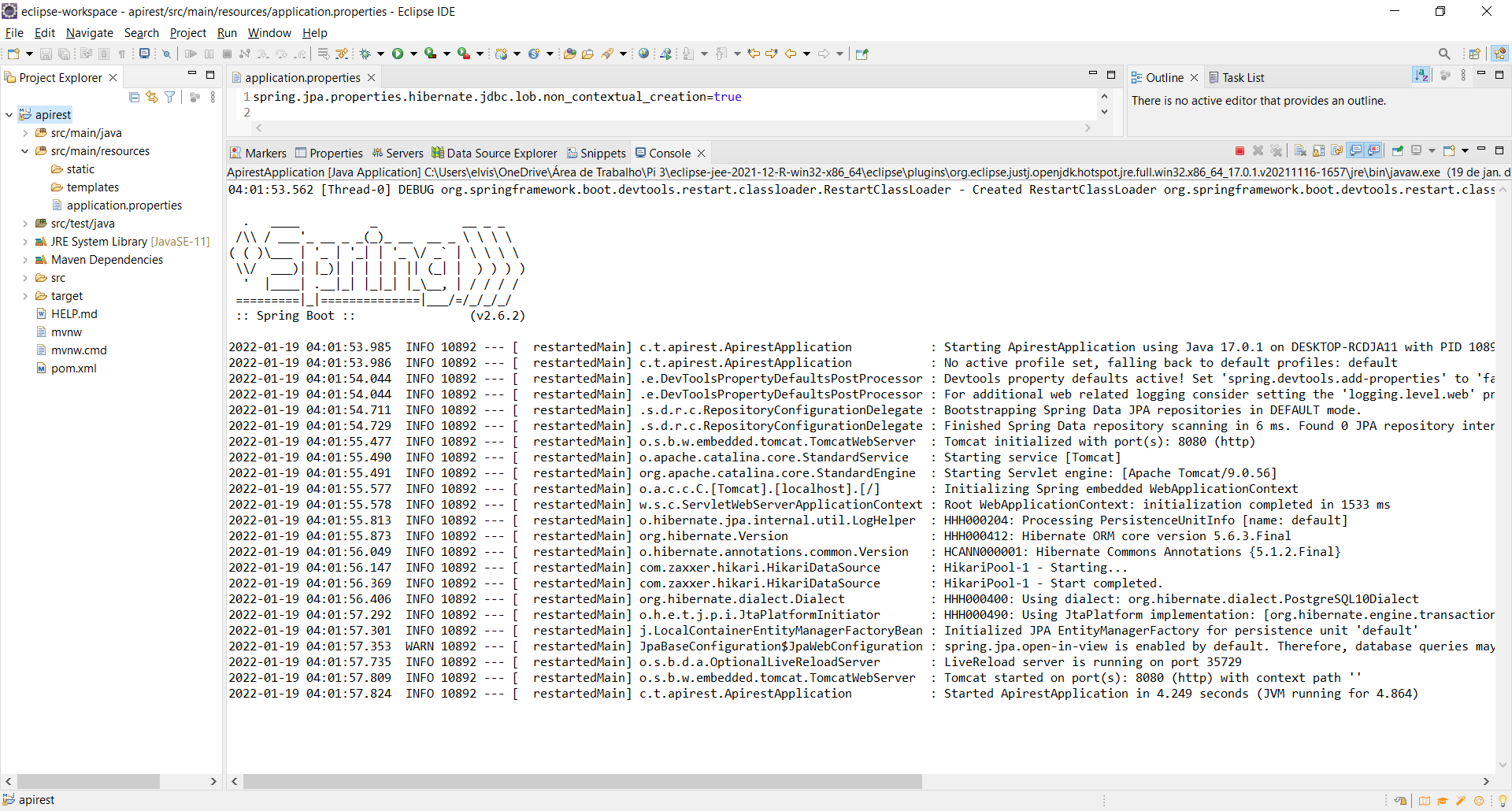
****

1. **Testando a conexão da aplicação com o banco de dados**

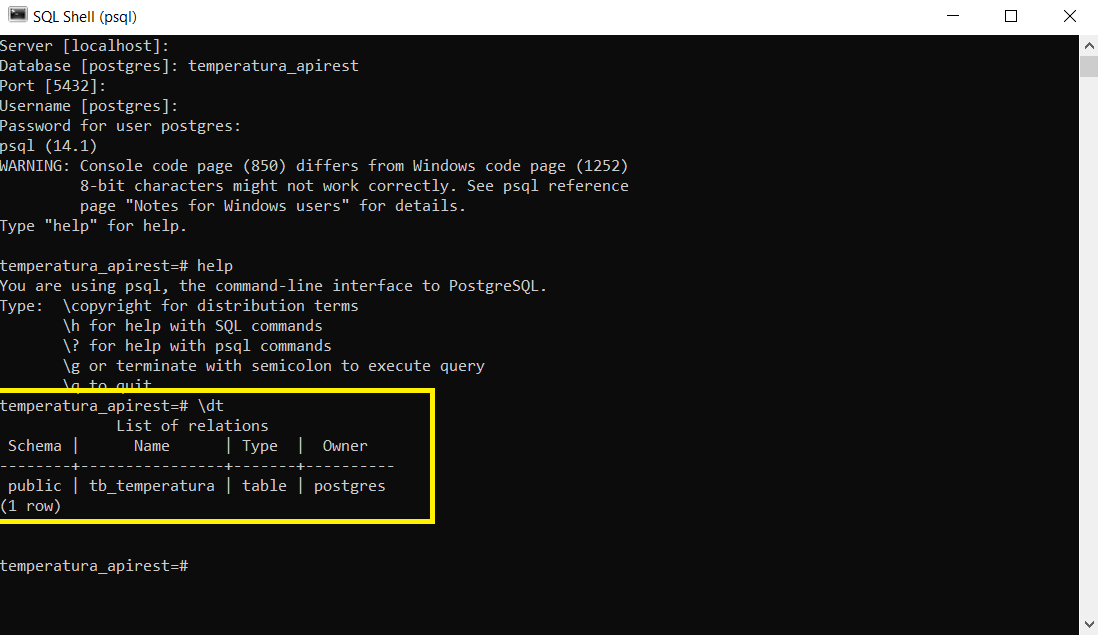
* Botão direito em ApiRestApplication.java
* Run as > 1 Java Application
* Clicar em **Permitir Acesso**



* Tela de Conexão Estabelecida

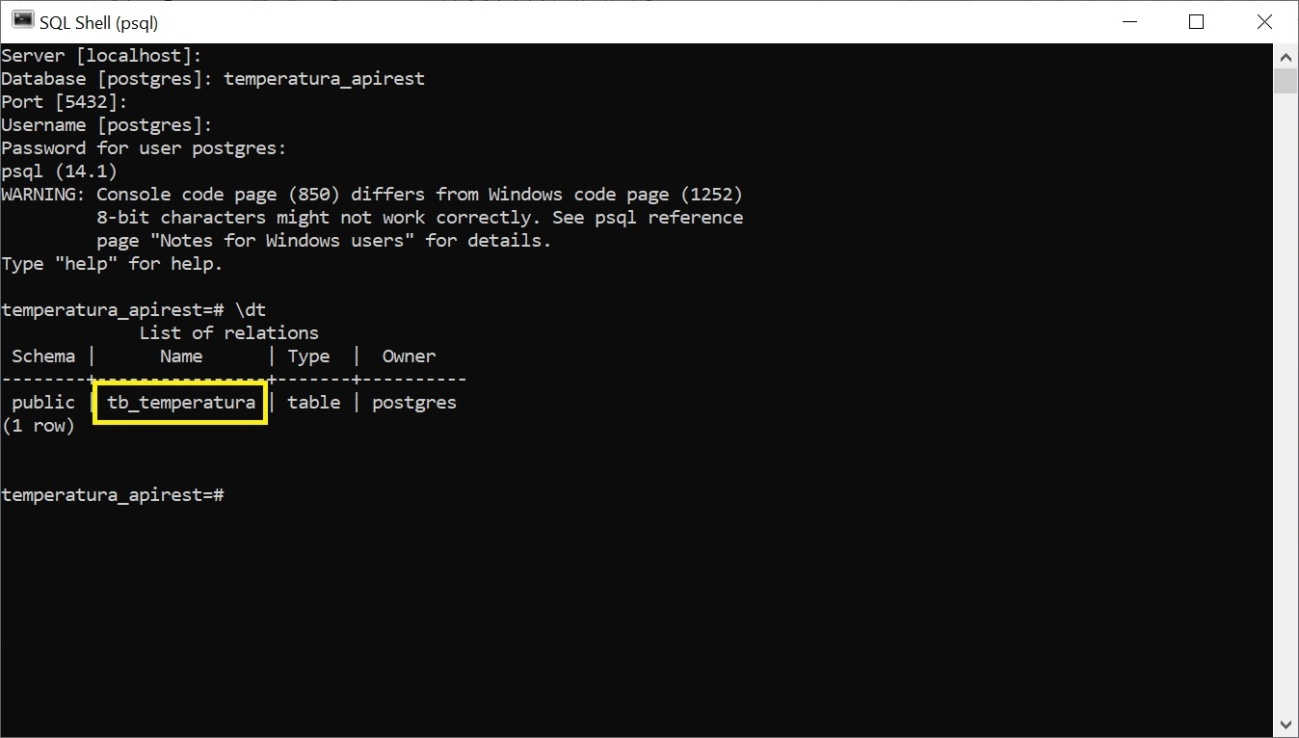


* Verificando a criação das tabelas geradas no banco de dados ***PostgresSQL***
* Digitar o comando: ***\dt***

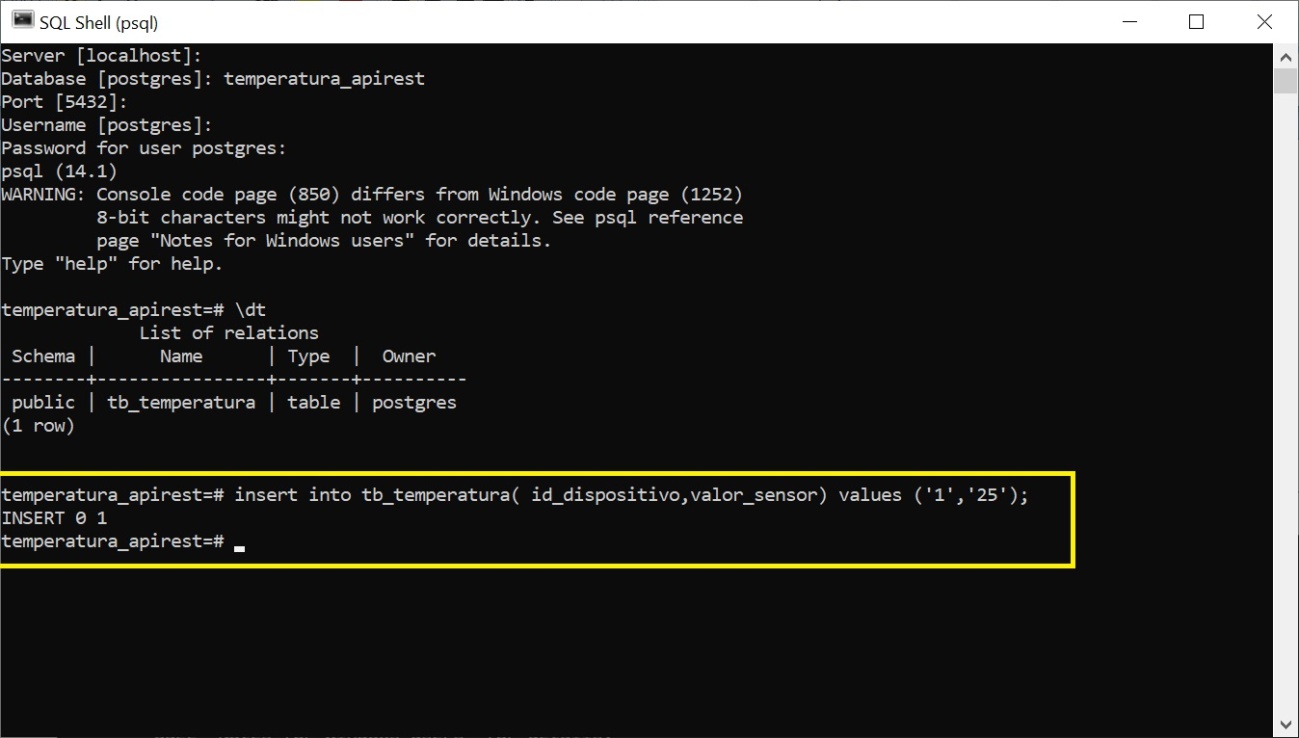


1. **Adicionando valores ao banco de dados**

* **Dados do Banco de dados**
* Nome do banco de dados: **temperatura\_apirest**
* Porta: **5432**
* Senha: **banco123**

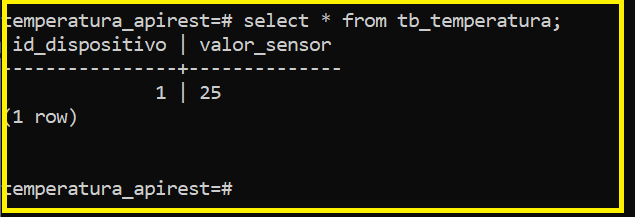


* **Comando para adicionar temperatura manualmente no banco de dados**
* insert into tb\_temperatura( id\_dispositivo,valor\_sensor) values ('1','25');



* **Comando para verificar o conteúdo do banco de dados**
* select \* from tb\_temperatura;

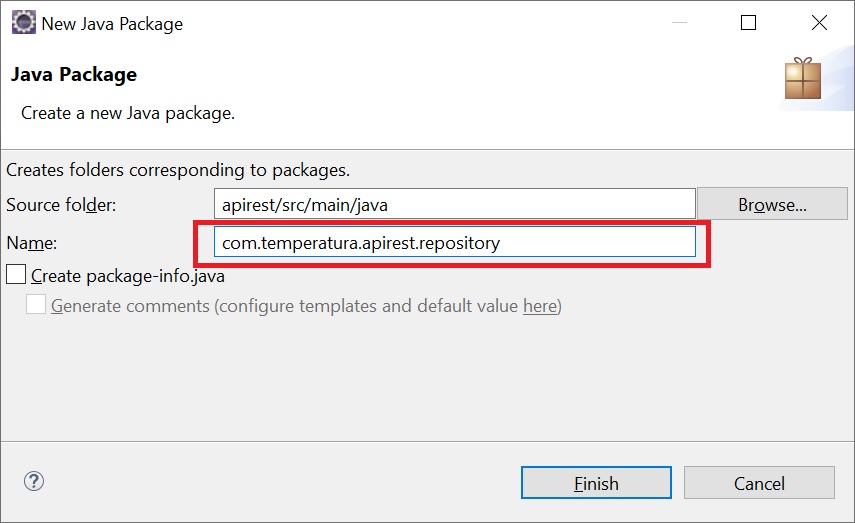
Validação de teste:



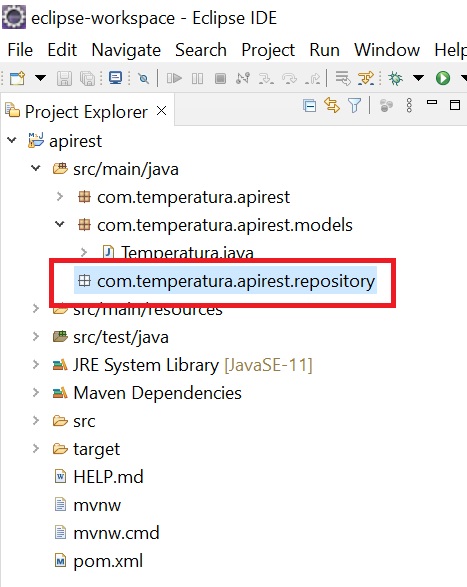
1. **Criar Repositório para a classe Temperatura.java**

* Criar **pacote** repository

Name: com.temperatura.apirest.repository

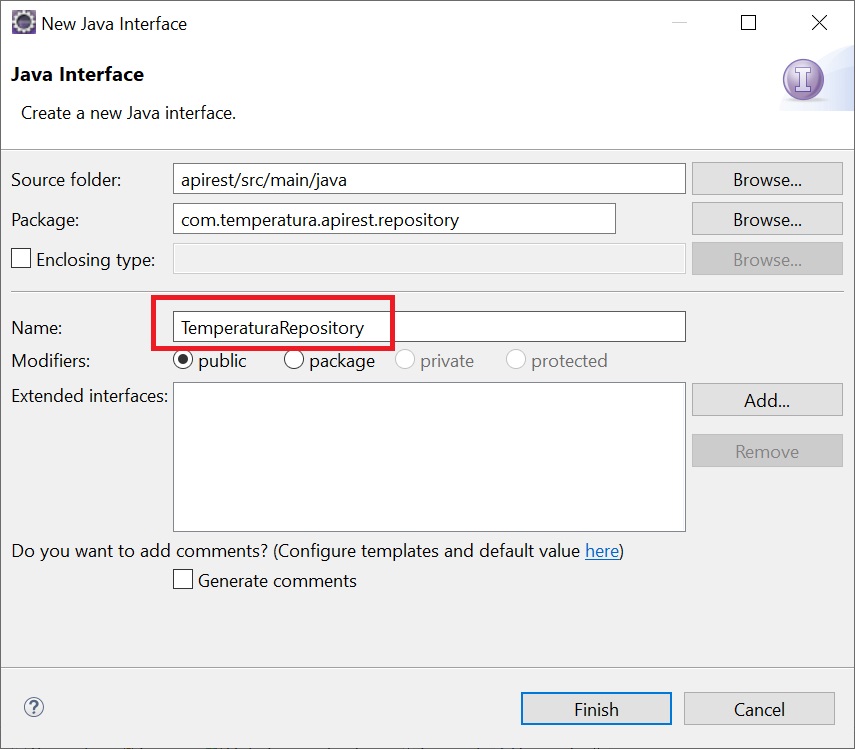


Validação de teste:

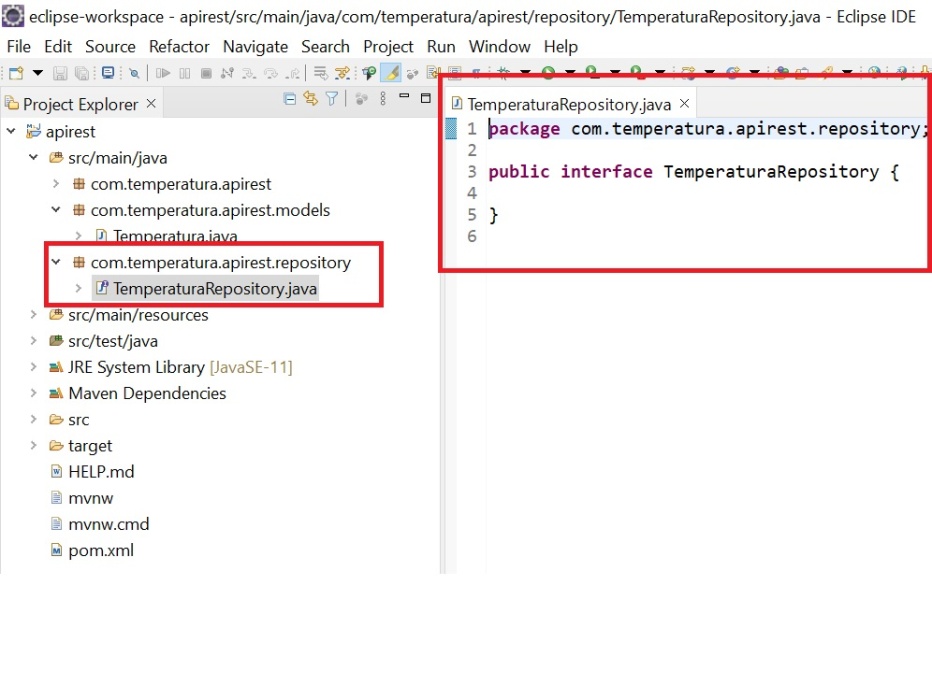


* Criar **interface** TemperaturaRepository

Name: **TemperaturaRepository**



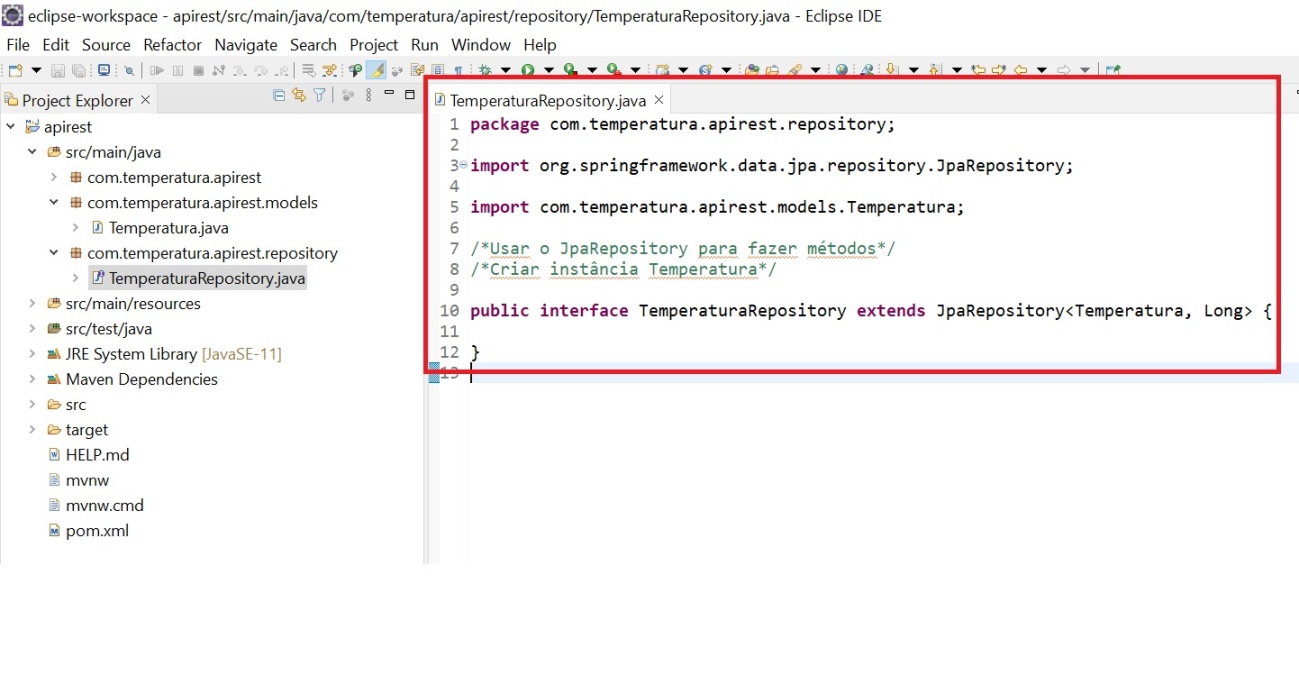
Validação de teste:



Código:

|  |
| --- |
| **package** com.temperatura.apirest.repository;  **import** org.springframework.data.jpa.repository.JpaRepository;  **import** com.temperatura.apirest.models.Temperatura;  /\*Usar o JpaRepository para fazer métodos\*/  /\*Criar instância Temperatura\*/  **public** **interface** TemperaturaRepository **extends** JpaRepository<Temperatura, Long> {  } |

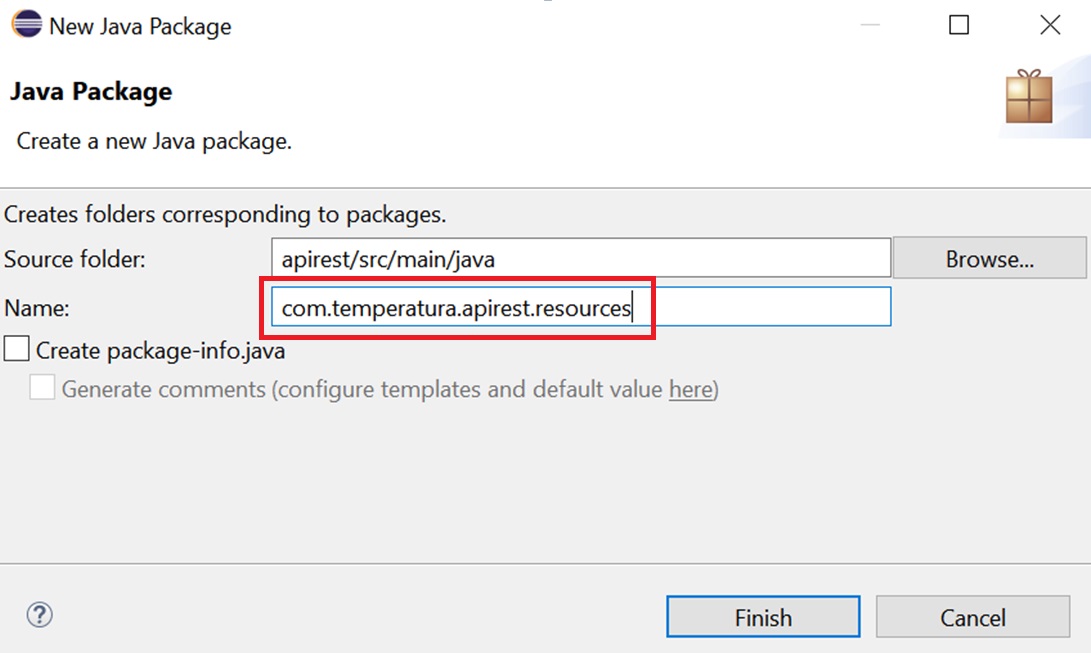
Validação de teste:



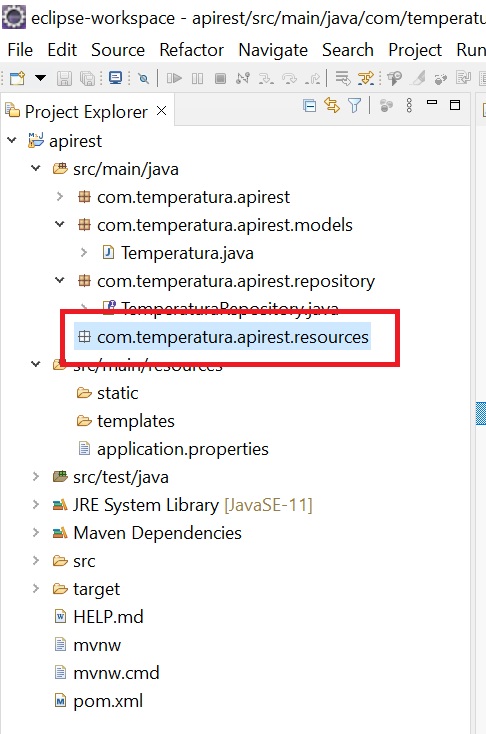
1. **Criar Métodos com persistência para banco de dados para a classe Temperatura.java**

* Criar **pacote** resources

Name: com.temperatura.apirest.resources

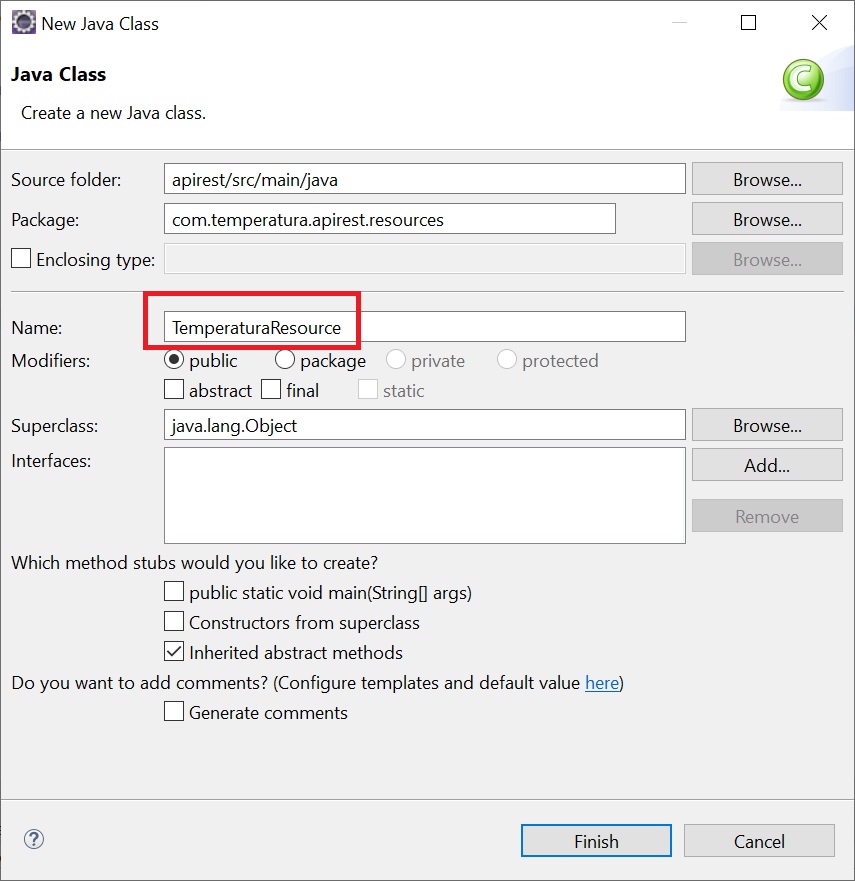


Validação de Teste:

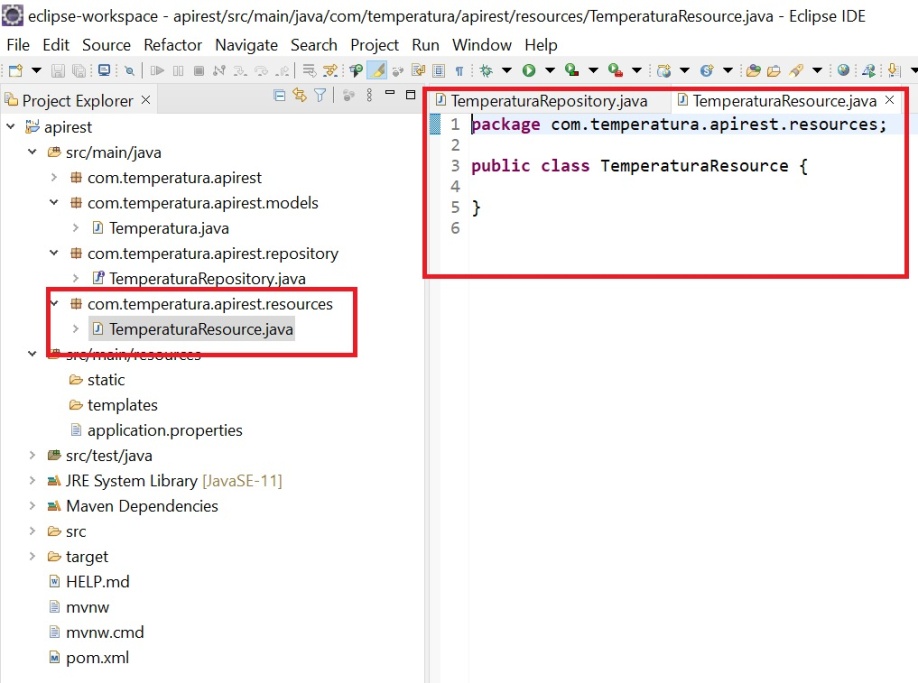


* Criar **classe** TemperaturaResource

Name: **TemperaturaResource**



Validação de Teste:

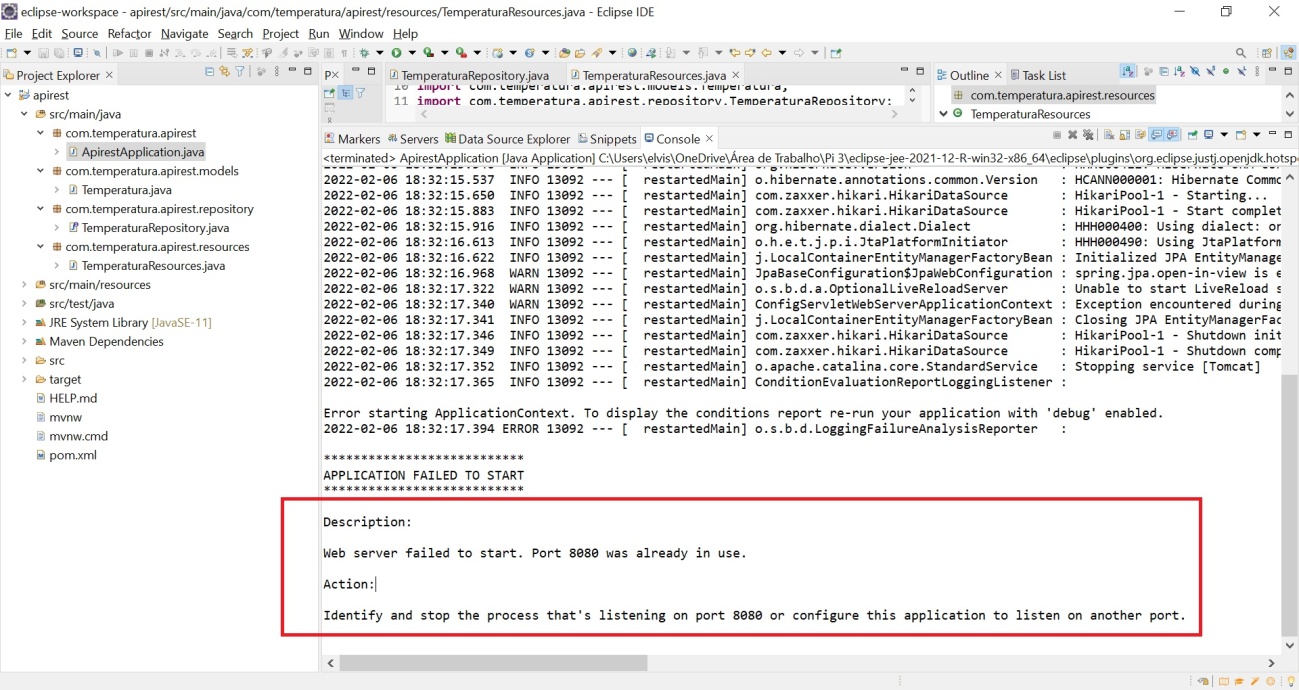


**Código:**

|  |
| --- |
| **package com.temperatura.apirest.resources;**  **import java.util.List;**  **import org.springframework.beans.factory.annotation.Autowired;**  **import org.springframework.web.bind.annotation.GetMapping;**  **import org.springframework.web.bind.annotation.RequestMapping;**  **import org.springframework.web.bind.annotation.RestController;**  **import com.temperatura.apirest.models.Temperatura;**  **import com.temperatura.apirest.repository.TemperaturaRepository;**  **@RestController**  **@RequestMapping(value = "/api")**  **public class TemperaturaResources {**  **@Autowired**  **TemperaturaRepository temperaturaRepository;**  **@GetMapping("/temperatura")**  **public List <Temperatura> listaTemperaturas() {**  **return temperaturaRepository.findAll();**  **}**  **}** |

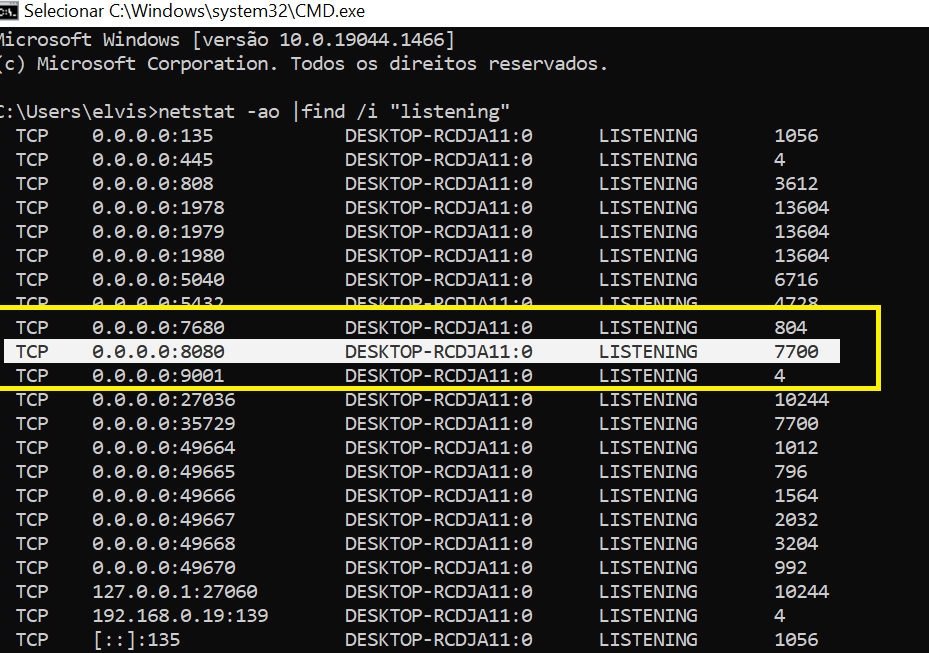
1. Liberando a porta de comunicação

* Erro ao subir a aplicação:



* Abrir o prompt de comando e vericar quais os processos estão sendo executados nas portas disponíveis

Comando: **netstat -ao |find /i "listening"**



* Parar o processo em execução no seu número de porta (neste caso, é 8080 e o ID do processo é **7700**)

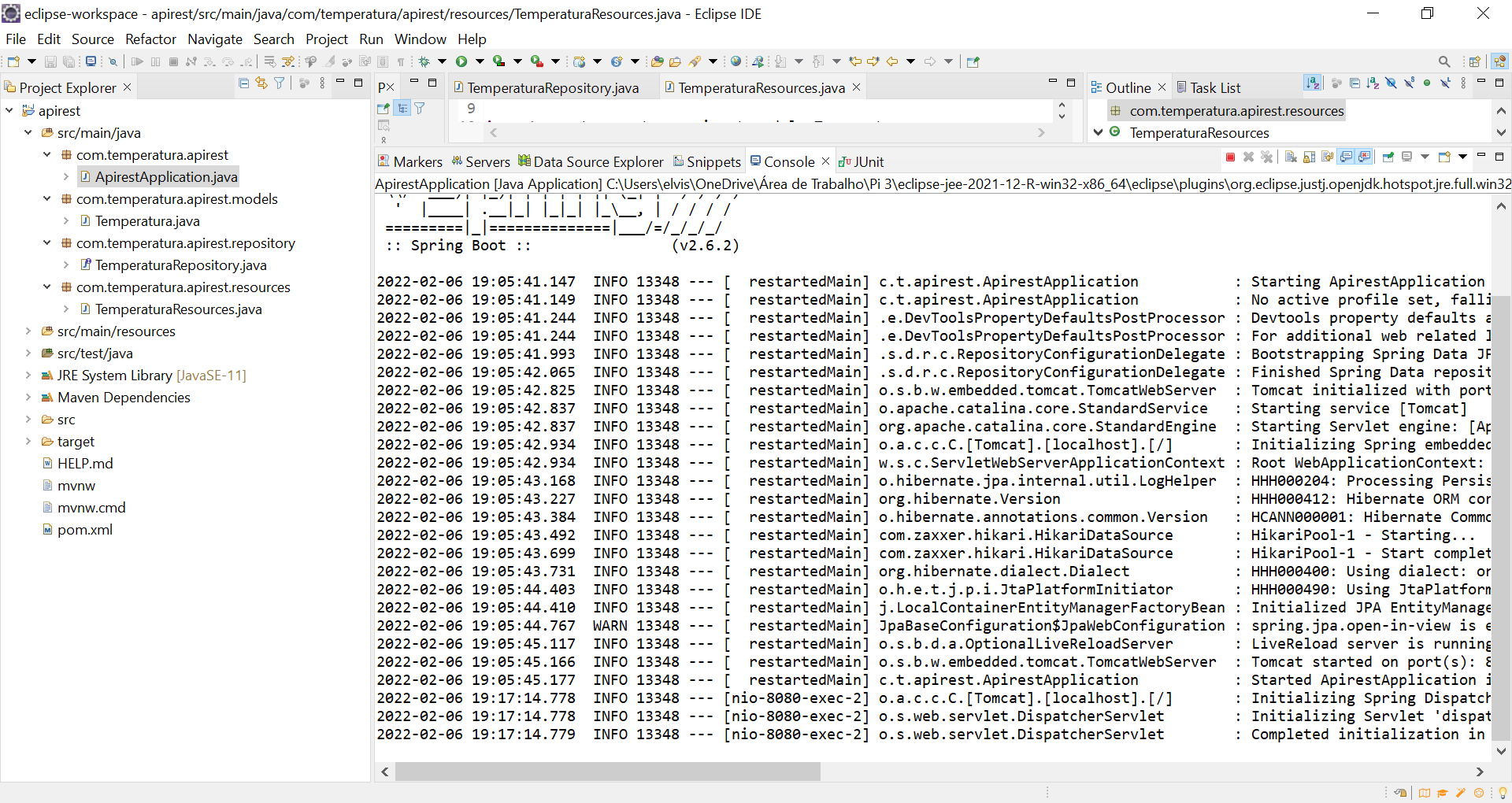
Comando: **Taskkill /F /IM 7700**

1. **Iniciando a aplicação**

* Botão direito em ApiRestApplication.java
* Run as > 1 Java Application

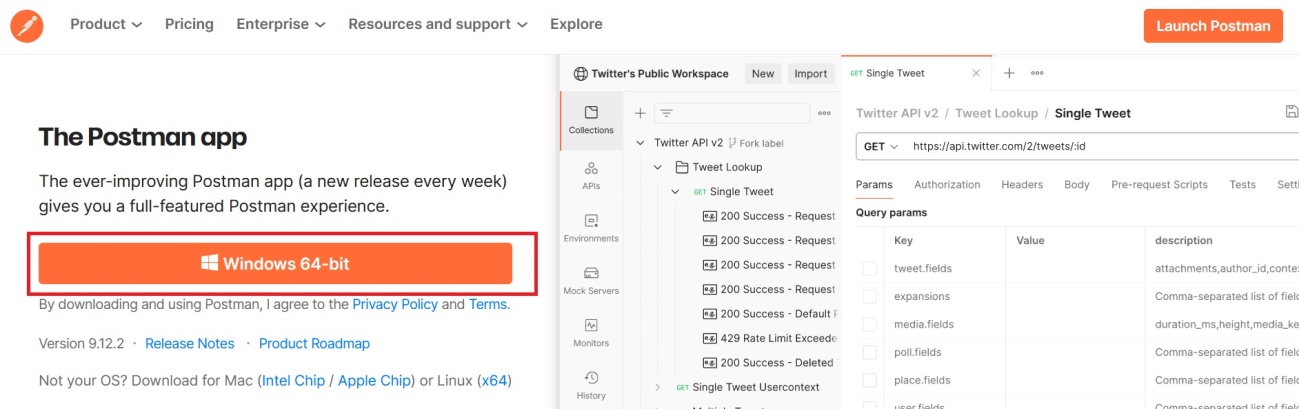


Validação de Teste:



1. **Testes**

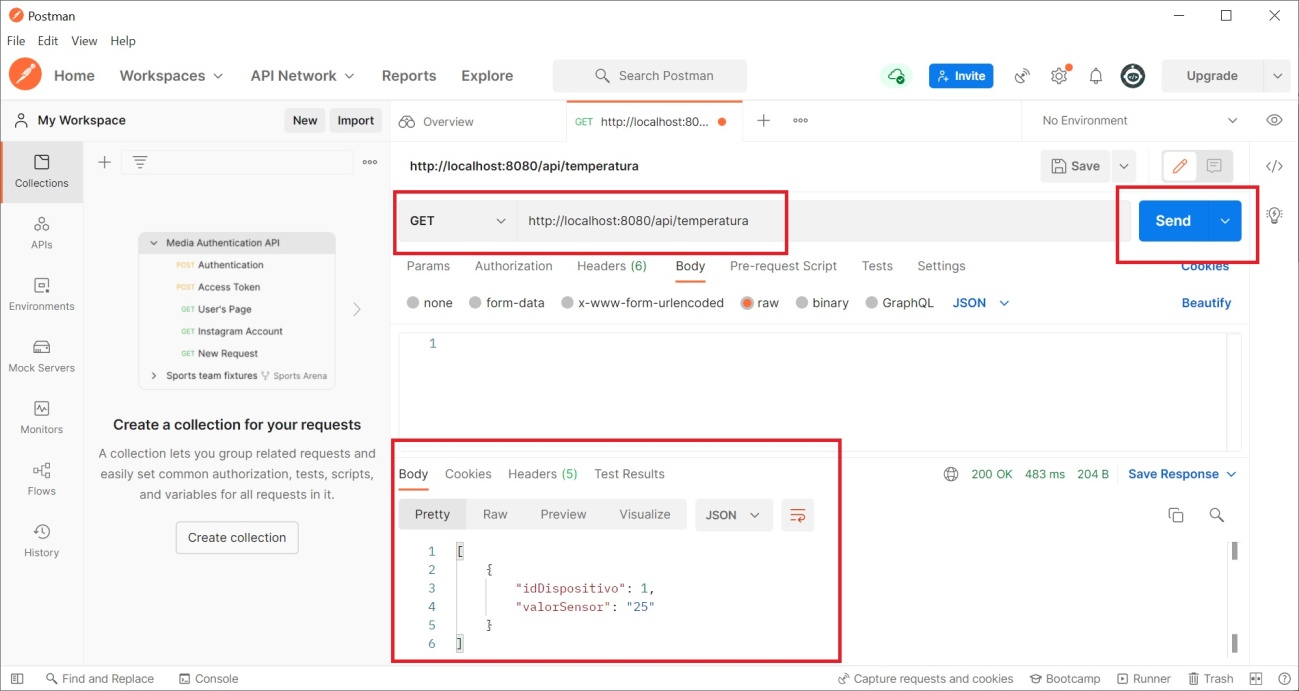
* Acessar o link: <https://www.postman.com/downloads/>
* Clicar em Windows 64-bit



* Baixar e instalar o *software* Postman;
  1. **Testando a aplicação utilizando o *software* Postman**
* Para listar as temperaturas utilizando o método GET

**Requisição HTTP:** [**http://localhost:8080/api/temperatura**](http://localhost:8080/api/temperatura)

**Validação de Teste:**



* 1. **Método para Listar Temperatura Específica**
* Para listar a temperatura pelo ID utilizando o método GET

**Requisição:** [**http://localhost:8080/api/temperatura/listar/1**](http://localhost:8080/api/temperatura/listar/1)

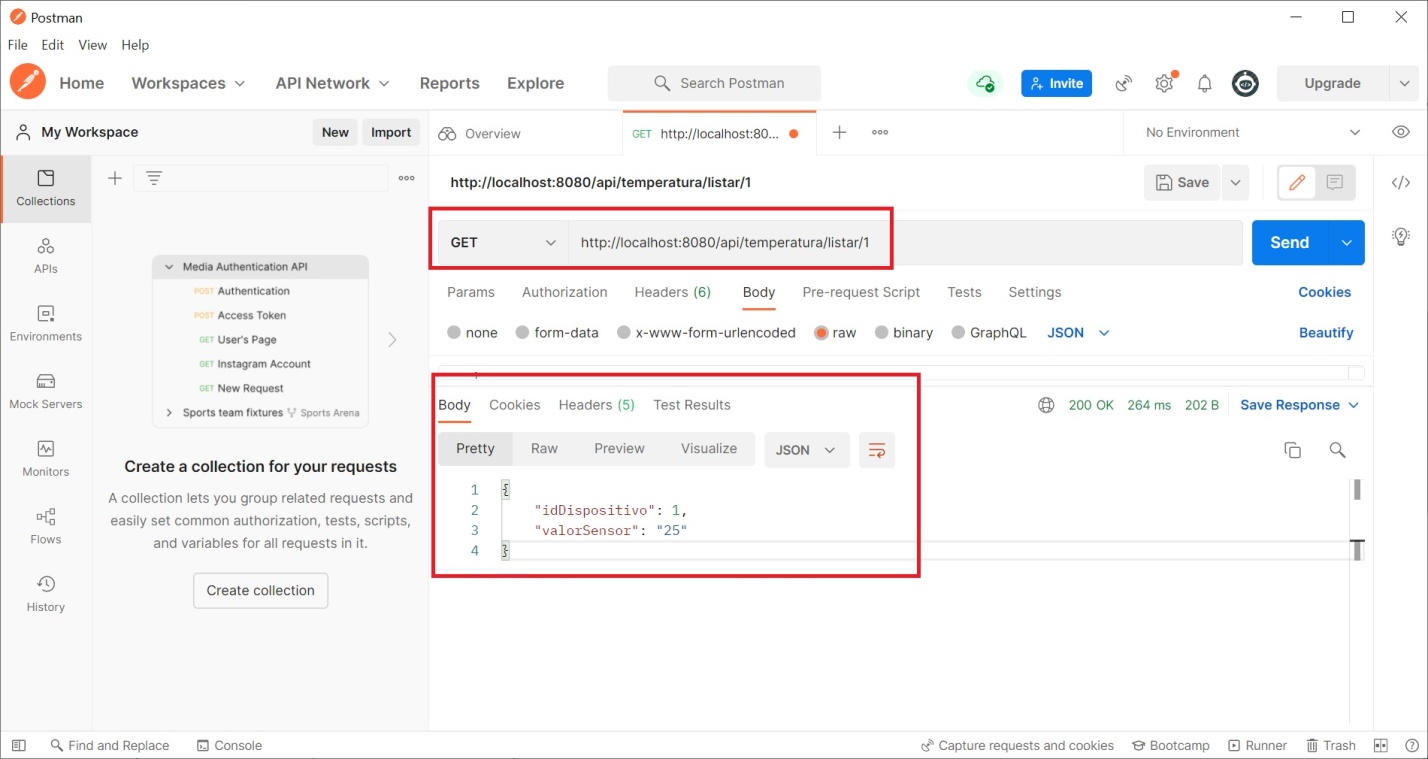
**Imports:**

|  |
| --- |
| **import** org.springframework.web.bind.annotation.PathVariable;  **import** org.springframework.beans.factory.annotation.Autowired; |

**Código:**

|  |
| --- |
| @GetMapping("/temperatura/listar/{id}")  public Temperatura listaTemperaturaUnica(@PathVariable(value = "id") long id) {  return temperaturaRepository.findById(id);  } |

**Validação de Teste:**



* 1. **Método para Salvar Temperatura Específica**
* Para adicionar uma temperatura ao banco de dados utilizando o método POST:
* No postman selecionar Raw;
* Exemplo de envio:

{

"valorSensor": "35"

}

**Requisição:** [**http://localhost:8080/api/temperatura/salvar**](http://localhost:8080/api/temperatura/salvar)

**Código:**

|  |
| --- |
| @PostMapping("/temperatura/salvar")  public Temperatura salvaProduto(@RequestBody Temperatura temperatura) {  return temperaturaRepository.save(temperatura);  } |

**Validação de Teste:**

|  |
| --- |
| **C:\Users\elvis\OneDrive\Área de Trabalho\Pi 3\Imagens\28- postman post.jpg** |

* 1. **Método para Deletar Temperatura Específica**
* Para apagar uma temperatura ao banco de dados utilizando o método DELETE;
* No postman selecionar Raw;

|  |
| --- |
| {  "idDispositivo": 2,  "valorSensor": "35"  } |

* No header:

|  |
| --- |
| Content-type: Application/json |

**Requisição:** [**http://localhost:8080/api/temperatura/deletar**](http://localhost:8080/api/temperatura/deletar)

**Código:**

|  |
| --- |
| @DeleteMapping("/temperatura/deletar")  **public** **void** deletaTemperatura(@RequestBody Temperatura temperatura) {  temperaturaRepository.delete(temperatura);  } |

**Validação de Teste:**

|  |
| --- |
| **C:\Users\elvis\OneDrive\Área de Trabalho\Pi 3\Imagens\28- postman delete parte 1.jpg**  **C:\Users\elvis\OneDrive\Área de Trabalho\Pi 3\Imagens\28- postman delete parte 2.jpg** |

* 1. **Método para Atualizar Temperatura Específica**
* Para atualizar uma temperatura ao banco de dados utilizando o método PUT:
* No postman selecionar Raw;

|  |
| --- |
| {  "idDispositivo": 1,  "valorSensor": "35"  } |

* No header:

|  |
| --- |
| Content-type: Application/json |

**Requisição:** [**http://localhost:8080/api/temperatura/atualizar**](http://localhost:8080/api/temperatura/atualizar)

**Código:**

|  |
| --- |
| @PutMapping("/temperatura/atualizar")  **public** Temperatura atualizaTemperatura(@RequestBody Temperatura temperatura) {  **return** temperaturaRepository.save(temperatura);  } |

**Validação de Teste:**

|  |
| --- |
| **C:\Users\elvis\OneDrive\Área de Trabalho\Pi 3\Imagens\29- postman PUT parte 1.jpg**  **C:\Users\elvis\OneDrive\Área de Trabalho\Pi 3\Imagens\30- postman PUT parte 2.jpg** |

1. **Resumo**

|  |  |  |
| --- | --- | --- |
| **Requisição** | **Tipo de Método** | **Resultado Esperado** |
| [**http://localhost:8080/api/temperatura**](http://localhost:8080/api/temperatura) | Get | [  {  "idDispositivo": 1,  "valorSensor": "25"  }  ] |
| [**http://localhost:8080/api/temperatura/listar/1**](http://localhost:8080/api/temperatura/listar/1) | Get | {  "idDispositivo": 1,  "valorSensor": "25"  } |
| [**http://localhost:8080/api/temperatura/salvar**](http://localhost:8080/api/temperatura/salvar) | POST | {  "idDispositivo": 2,  "valorSensor": "35"  } |
| [**http://localhost:8080/api/temperatura/deletar**](http://localhost:8080/api/temperatura/deletar) | DELETE |  |
| [**http://localhost:8080/api/temperatura/atualizar**](http://localhost:8080/api/temperatura/atualizar) | PUT | {  "idDispositivo": 1,  "valorSensor": "35"  } |

Código da classe até aqui

|  |
| --- |
| package com.temperatura.apirest.resources;  import java.util.List;  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.RequestMapping;  import org.springframework.web.bind.annotation.RestController;  import com.temperatura.apirest.models.Temperatura;  import com.temperatura.apirest.repository.TemperaturaRepository;  @RestController  @RequestMapping(value = "/api")  public class TemperaturaResources {  @Autowired  TemperaturaRepository temperaturaRepository;  @GetMapping("/temperatura")  public List <Temperatura> listaTemperaturas() {  return temperaturaRepository.findAll();  }    @GetMapping("/temperatura/listar/{id}")  public Object listaTemperaturaUnica(@PathVariable(value = "id") long id) {  return temperaturaRepository.findById(id);  }    @PostMapping("/temperatura/salvar")  public Temperatura salvaProduto(@RequestBody Temperatura temperatura) {  return temperaturaRepository.save(temperatura);  }    @DeleteMapping("/temperatura/deletar")  public void deletaTemperatura(@RequestBody Temperatura temperatura) {  temperaturaRepository.delete(temperatura);  }    @PutMapping("/temperatura/atualizar")  public Temperatura atualizaTemperatura(@RequestBody Temperatura temperatura) {  return temperaturaRepository.save(temperatura);  }    } |