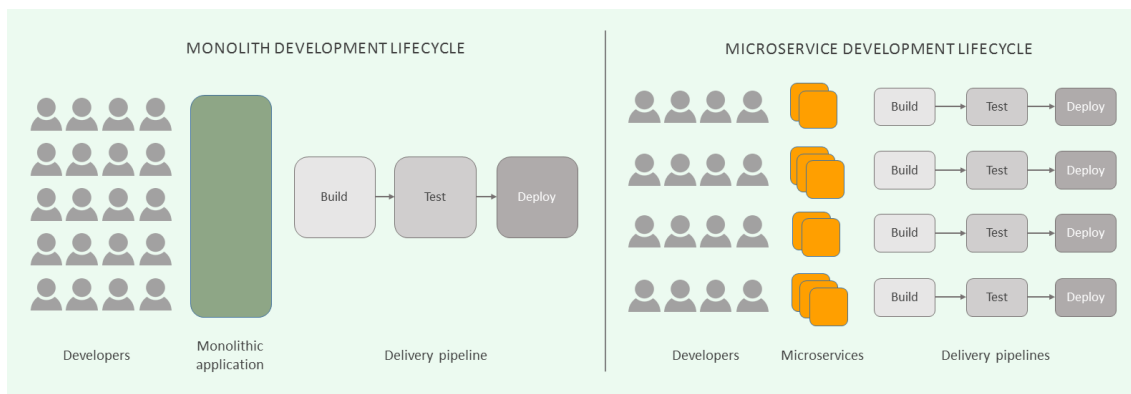
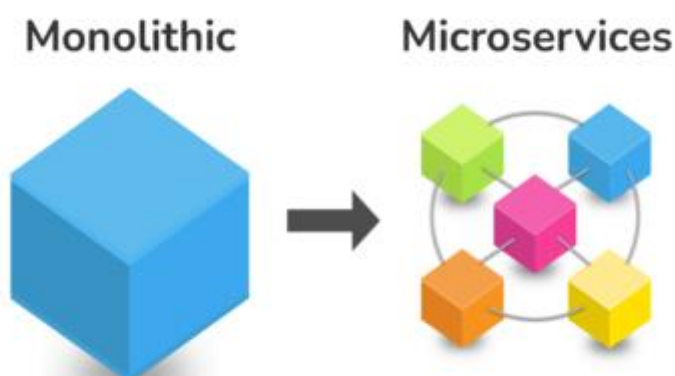


## O que são Microserviços?

**Microserviços** são uma abordagem arquitetônica e organizacional do desenvolvimento de software na qual o software consiste em pequenos serviços independentes que se comunicam usando APIs bem definidas.

Esses serviços pertencem a pequenas equipes autossuficientes. As arquiteturas de microserviços facilitam a escalabilidade e agilizam o desenvolvimento de aplicativos, habilitando a inovação e acelerando o tempo de introdução de novos recursos no mercado.



O **Spring Boot** é um framework que torna fácil a criação de aplicações Spring autossuficientes e robustas, possibilitando a execução imediata. Contudo isso só é possível por conta da abordagem opinativa sobre a plataforma Spring e bibliotecas de terceiros, que permite ao desenvolvedor gastar o mínimo de tempo possível configurando o projeto, e sim codificando suas regras de negócio.

Para cumprir com esse propósito, o framework se baseia em quatro princípios centrais:

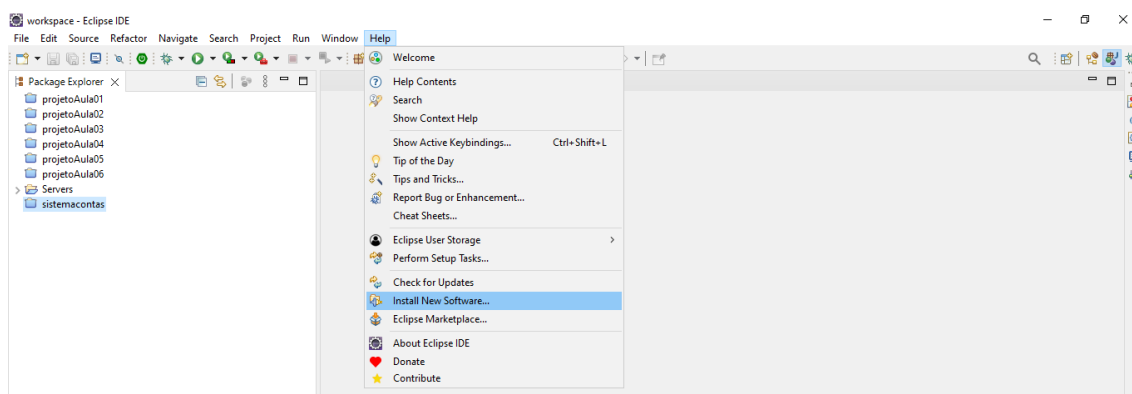
1. Oferecer uma experiência de início de projeto rápida e direta;
2. Apresentar uma visão opinativa e flexível sobre o modo como os projetos Spring devem ser configurados;

3. Fornecer requisitos não funcionais pré-configurados;
4. Não prover geração de código e zerar a necessidade de arquivos XML.

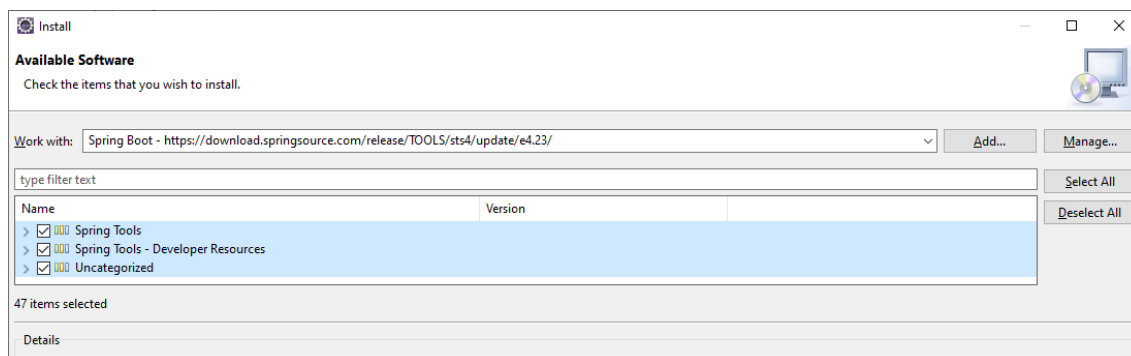
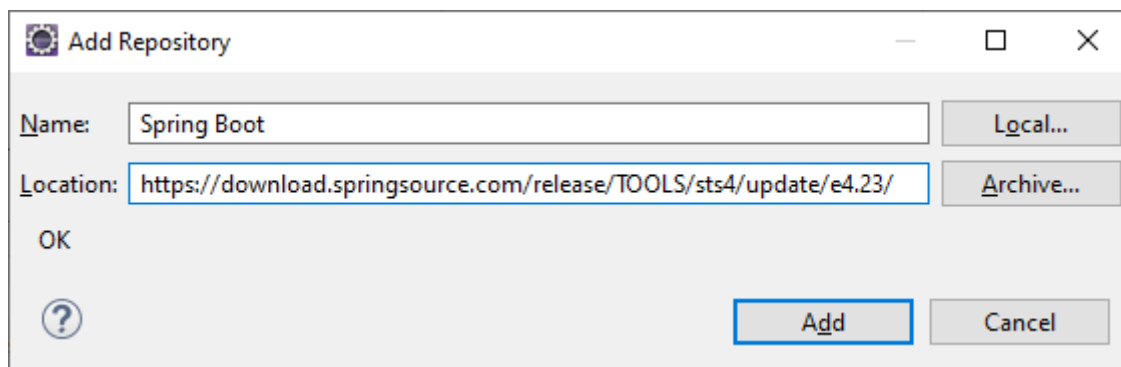


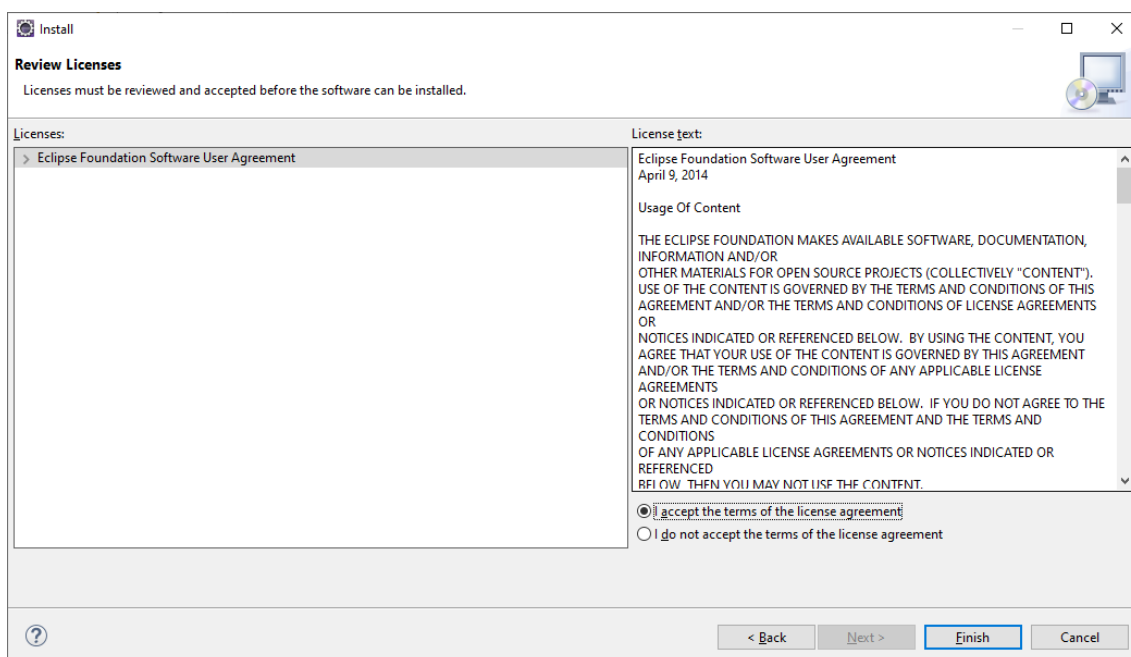
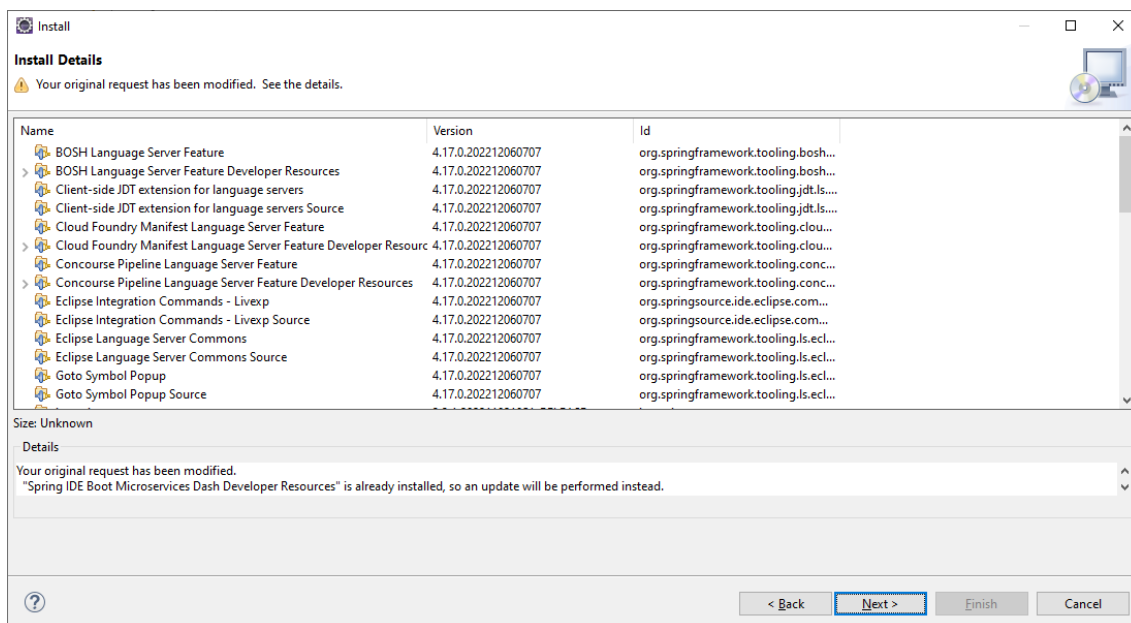
## Instalando um plugin no Eclipse para criação de projetos baseados em Spring Boot:

Help / Install new Software

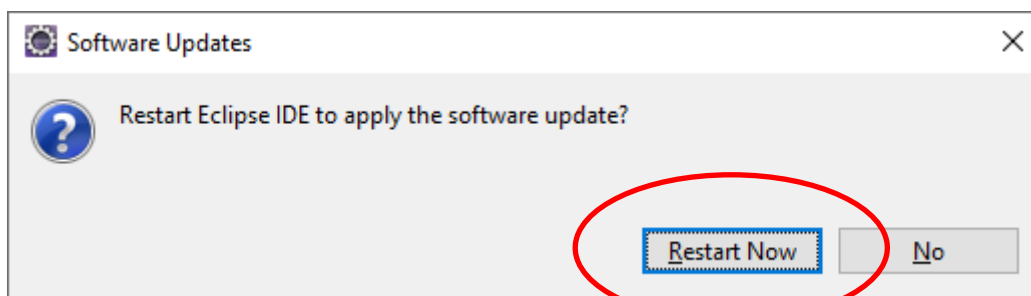


<https://download.springsource.com/release/TOOLS/sts4/update/e4.23/>





Ao final, reinicie o eclipse:

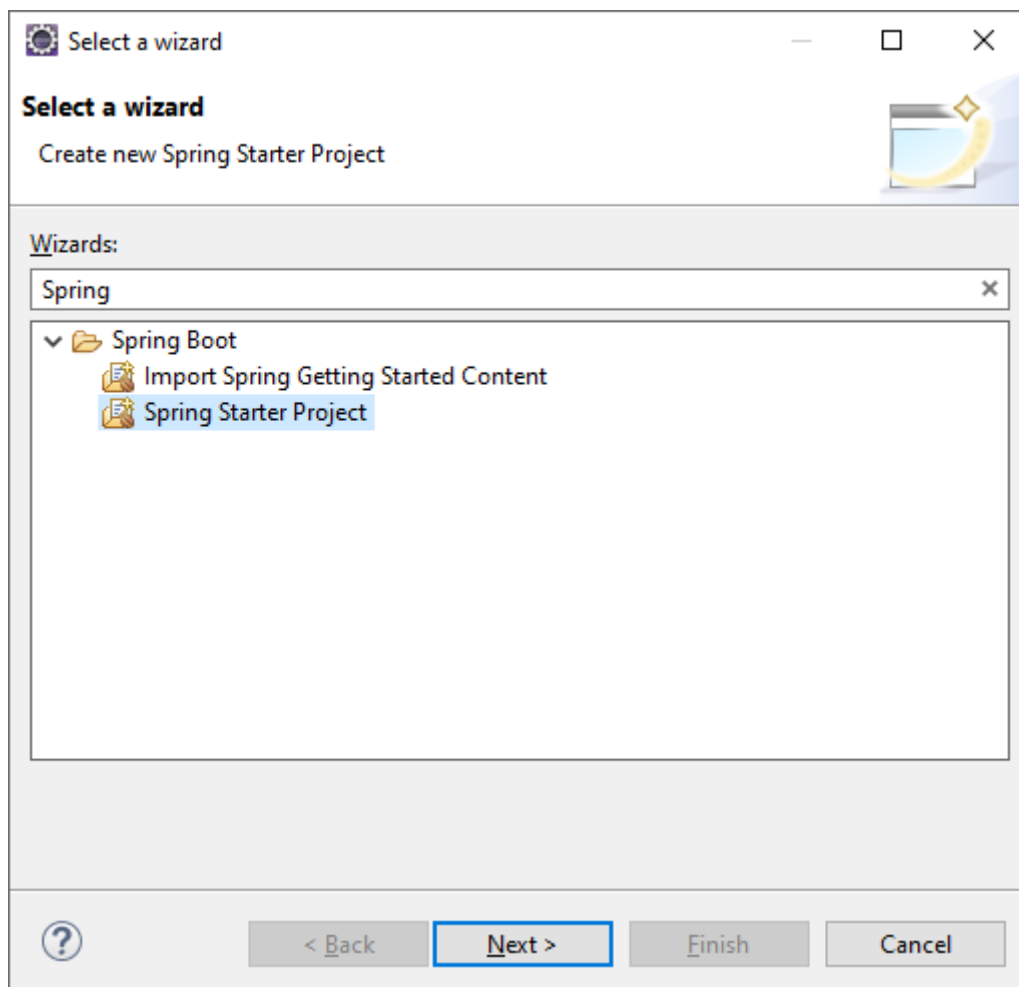
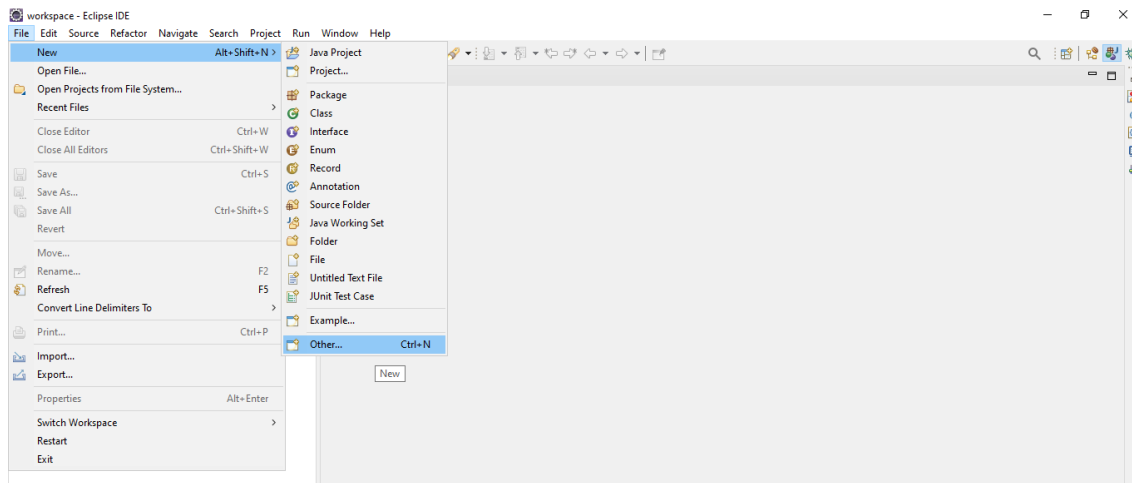


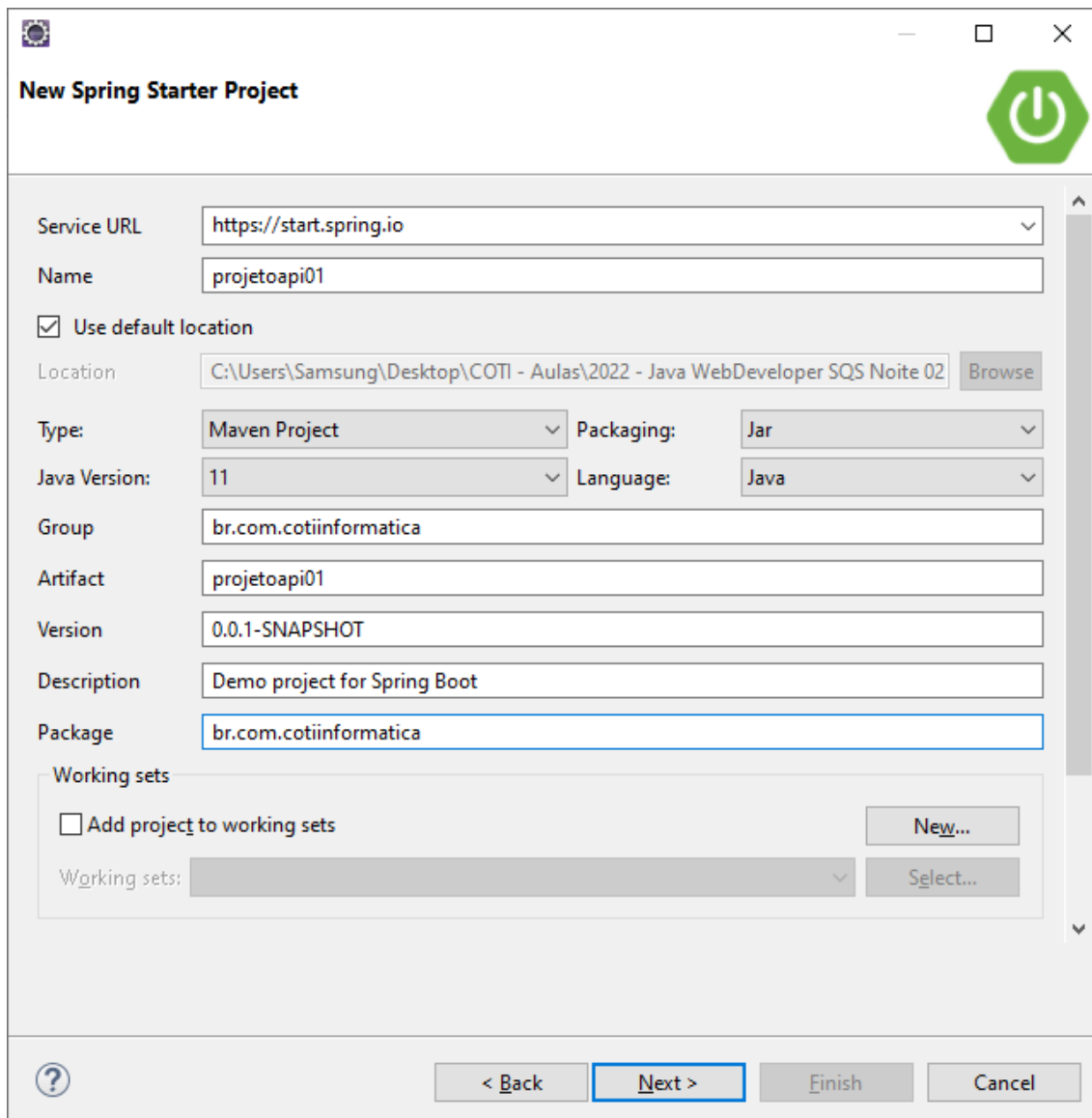
## Criando um projeto com Spring Boot:

Desenvolvendo uma API para controle de produtos.

- Esta API deverá permitir que produtos sejam cadastrados, editados, excluídos e também fornecer consultas de produtos.

### Criando o projeto:





**New Spring Starter Project**

Service URL:

Name:

☒ Use default location

Location:

Type:  Packaging:

Java Version:  Language:

Group:

Artifact:

Version:

Description:

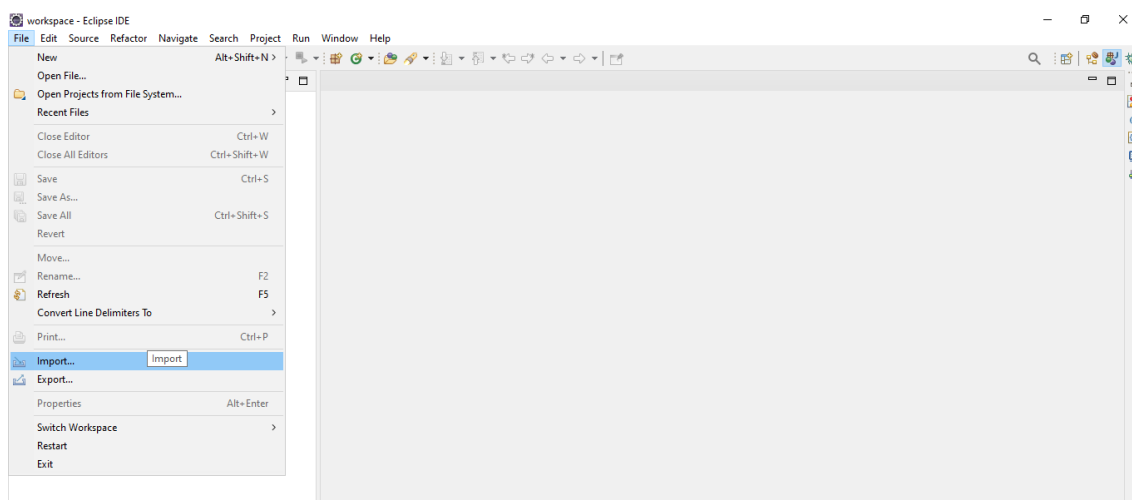
Package:

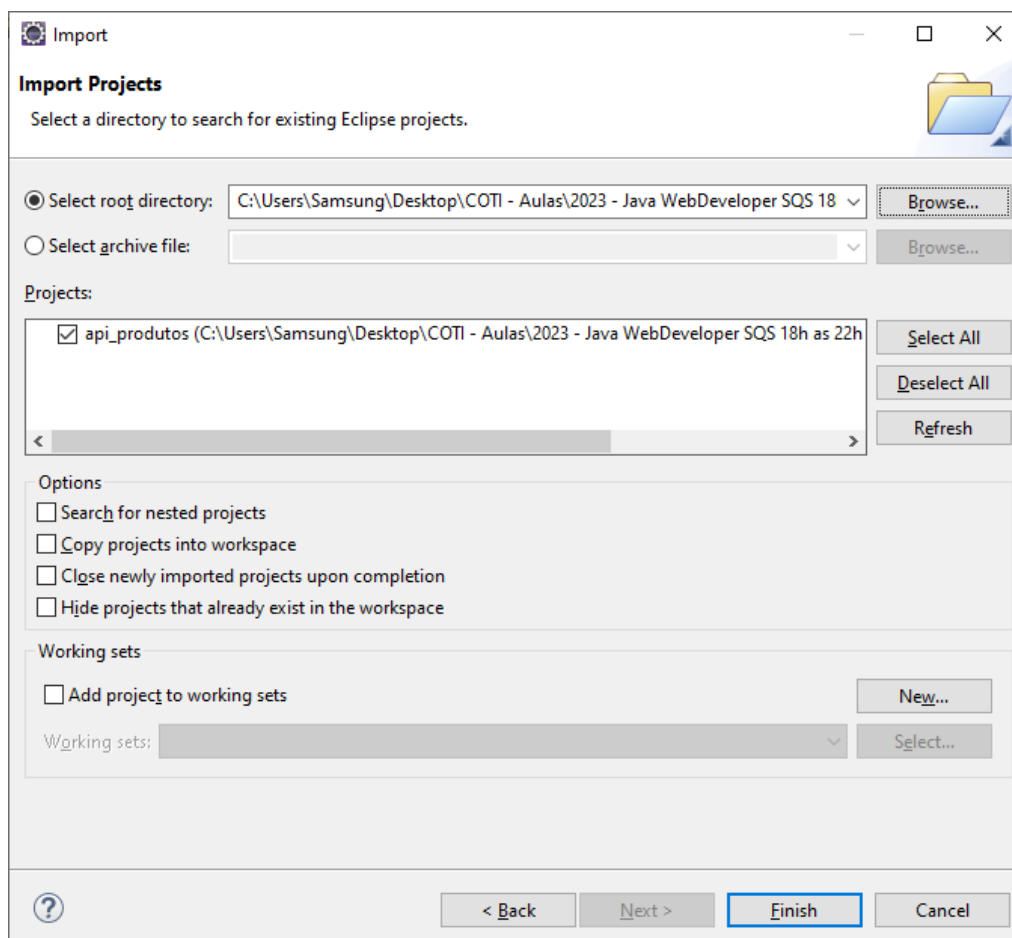
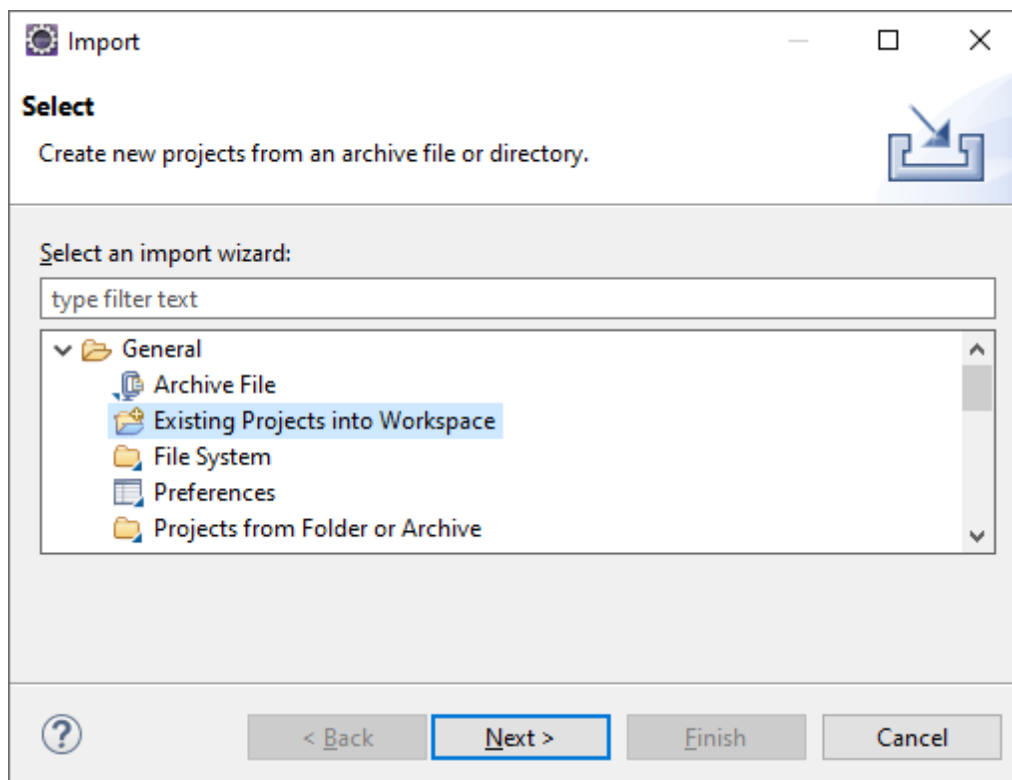
Working sets

☐ Add project to working sets

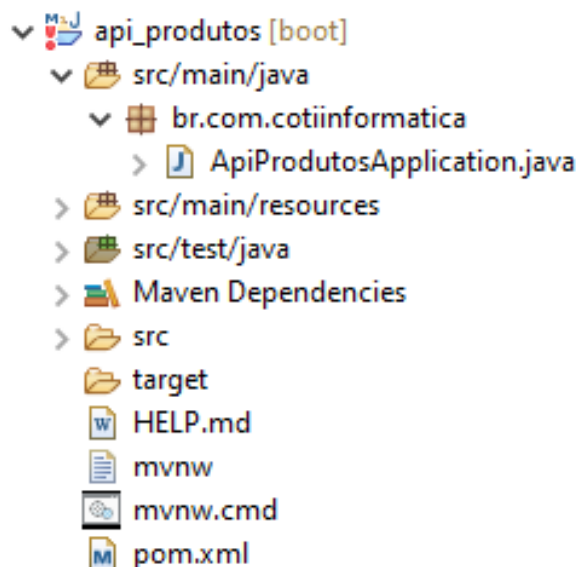
Working sets:

## Abrindo um projeto já existente:





### Estrutura do projeto:



### /pom.xml

Arquivo de configuração do Maven utilizado para instalarmos as bibliotecas e dependencias do projeto:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.2</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>br.com.cotiinformatica</groupId>
  <artifactId>api_produtos</artifactId>
  <version>1.0</version>
  <name>api_produtos</name>
  <description>API Produtos</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>

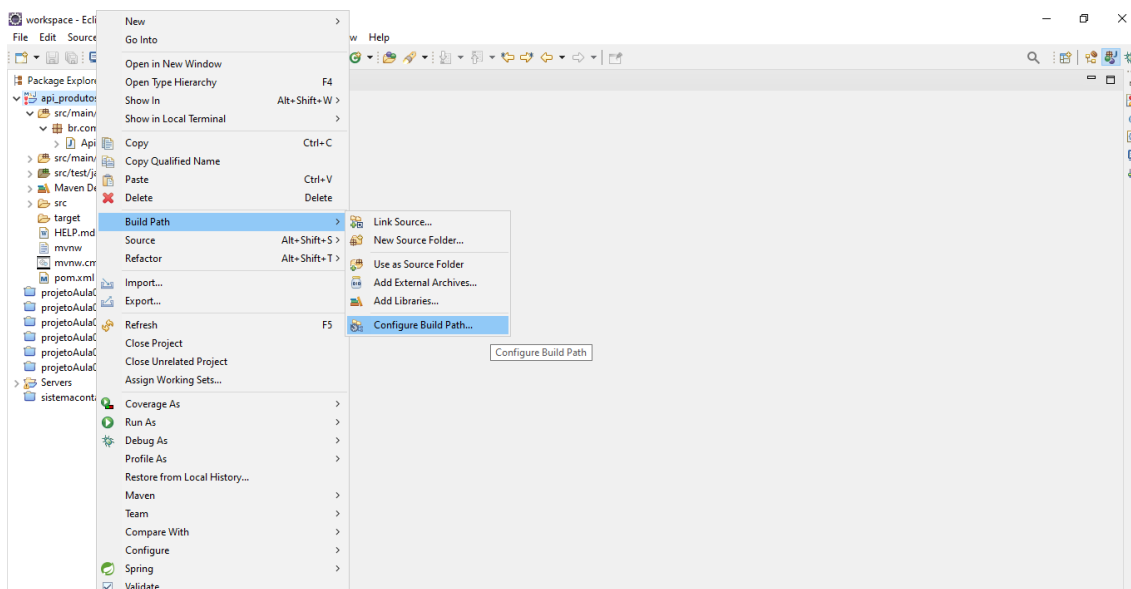
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

```
</dependencies>

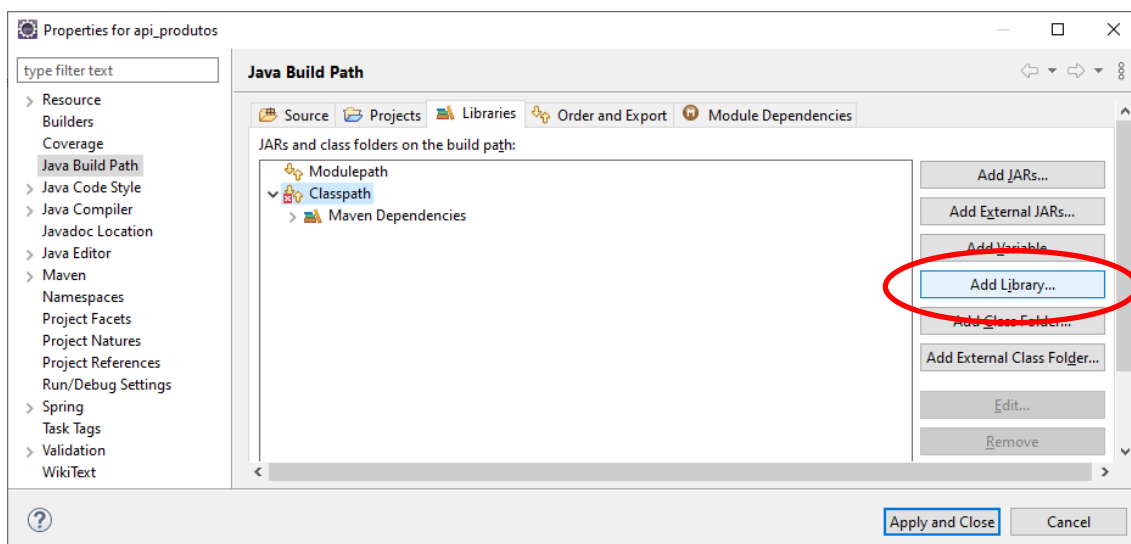
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

</project>
```

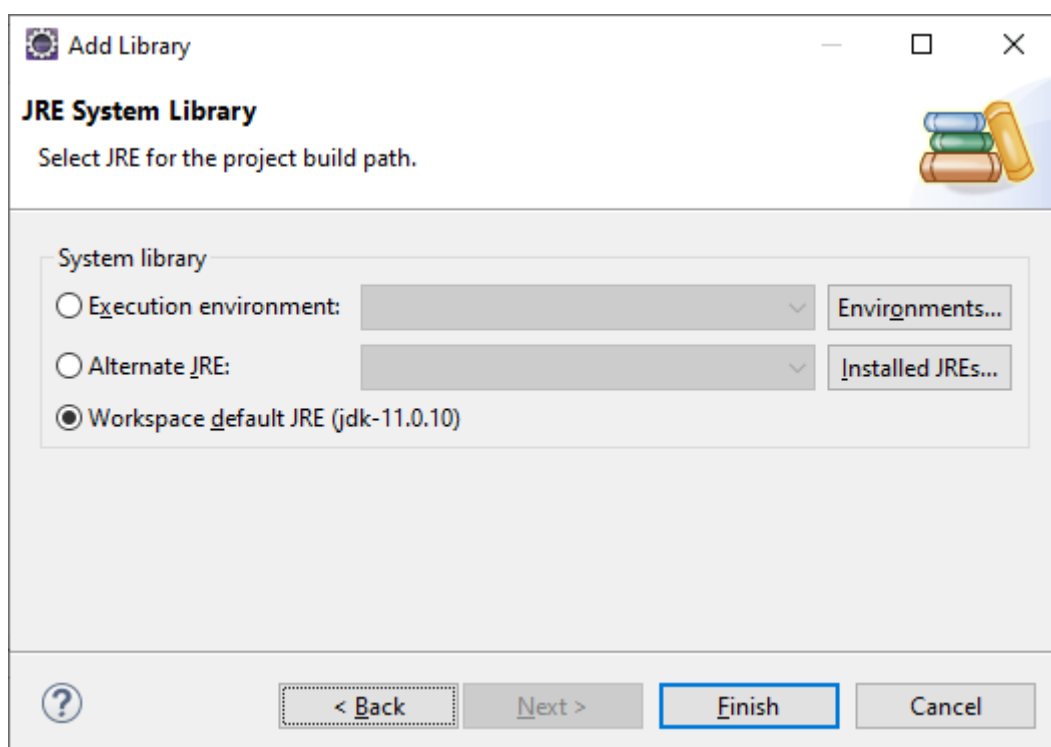
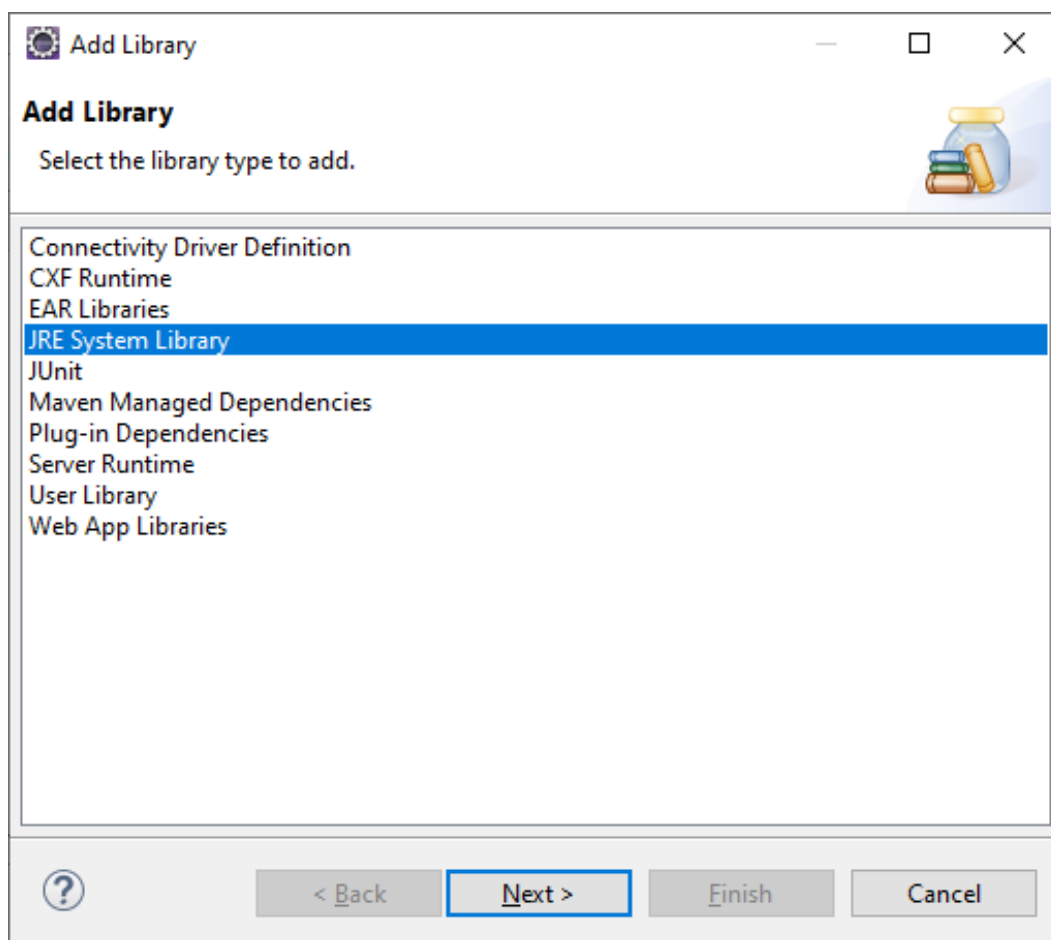
### Corrigindo a versão do Java utilizada no projeto:

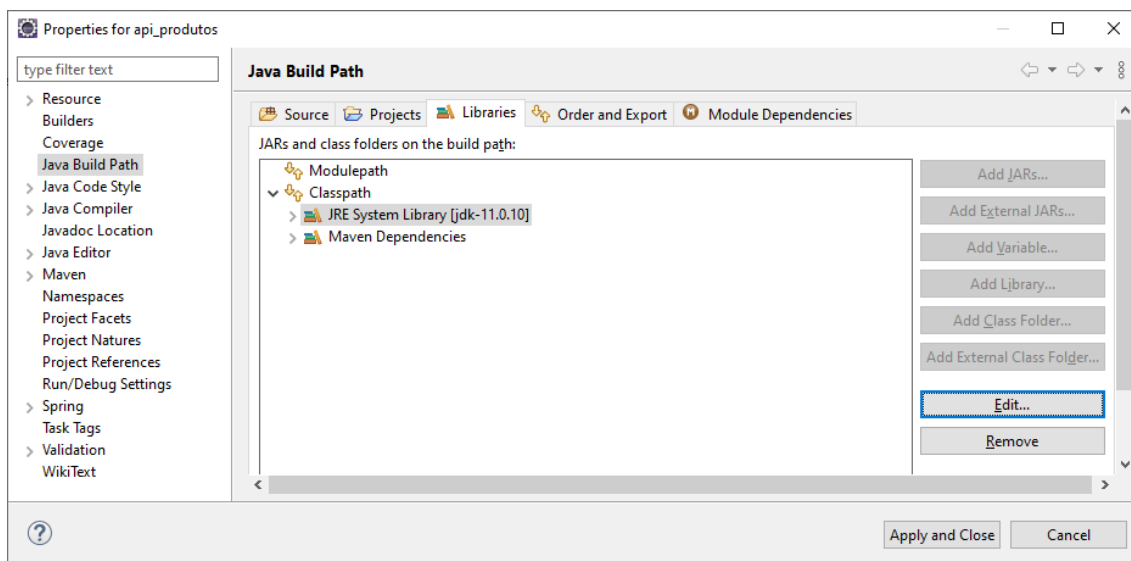


- Remova a opção do JDK17
- Em seguida, clique em: **Add Library**

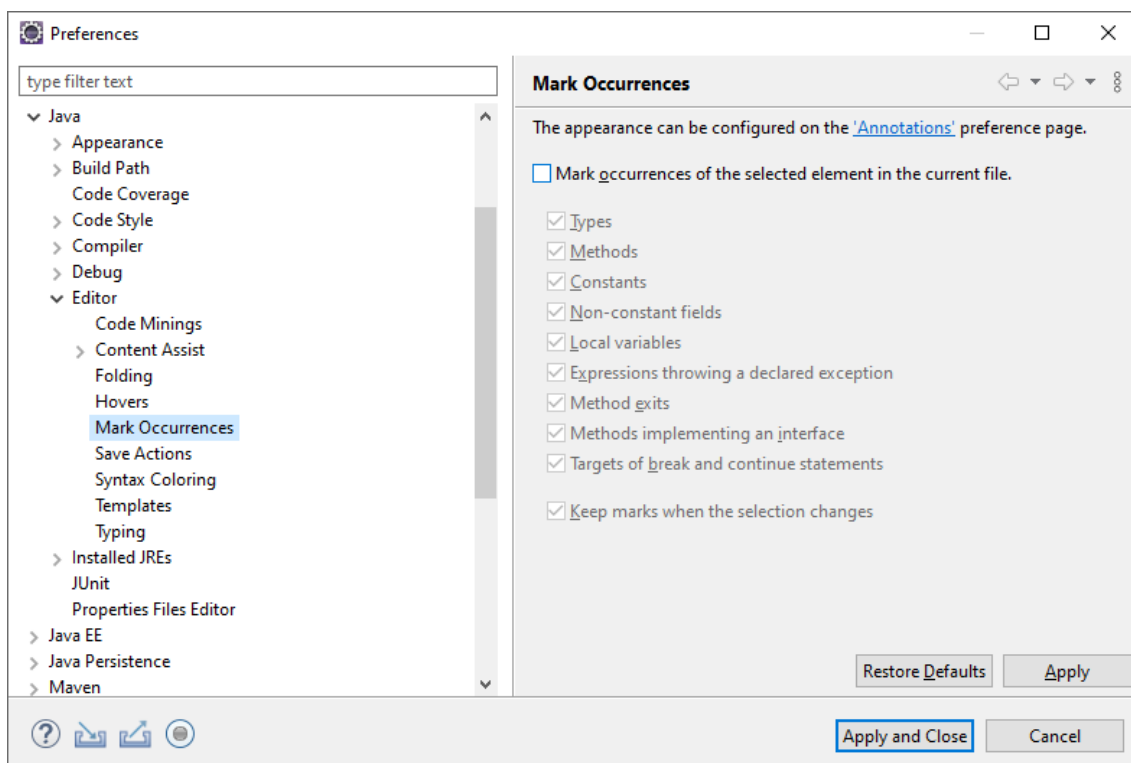
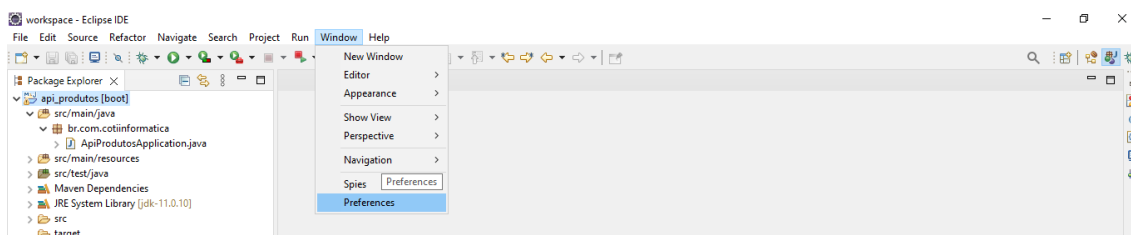








**Desmarque a opção de geração de ocorrências do editor do Java no Eclipse:**



## Configurando a versão do Spring Boot para 2.7.5

E adicionando bibliotecas para começar o desenvolvimento web:

### /pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.5</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>br.com.cotiinformatica</groupId>
  <artifactId>api_produtos</artifactId>
  <version>1.0</version>
  <name>api_produtos</name>
  <description>API Produtos</description>
  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

```
</dependencies>
```

```
<build>
```

```
<plugins>
```

```
<plugin>
```

```
<groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-maven-plugin</artifactId>
```

```
</plugin>
```

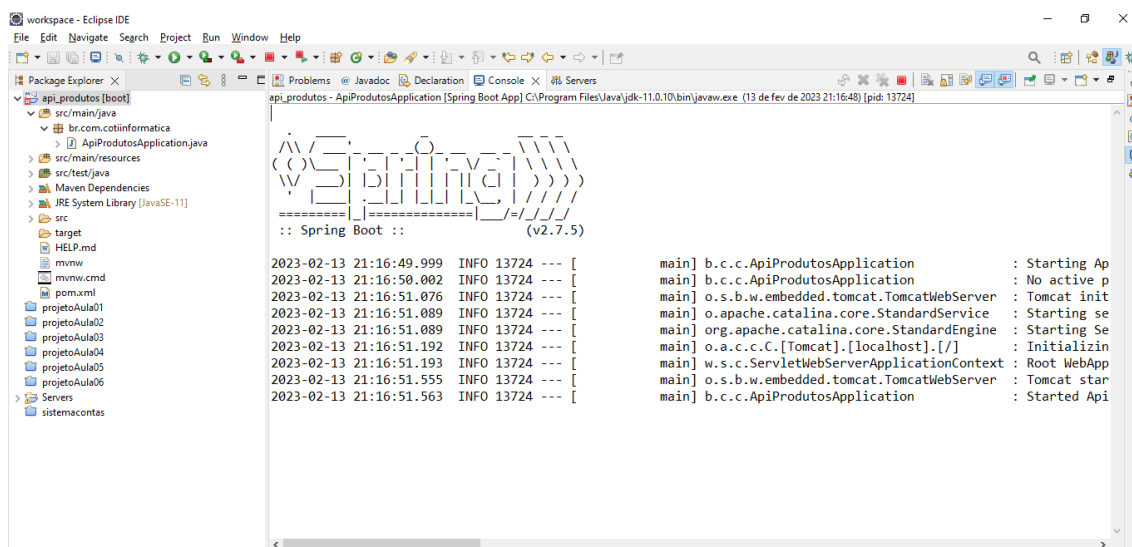
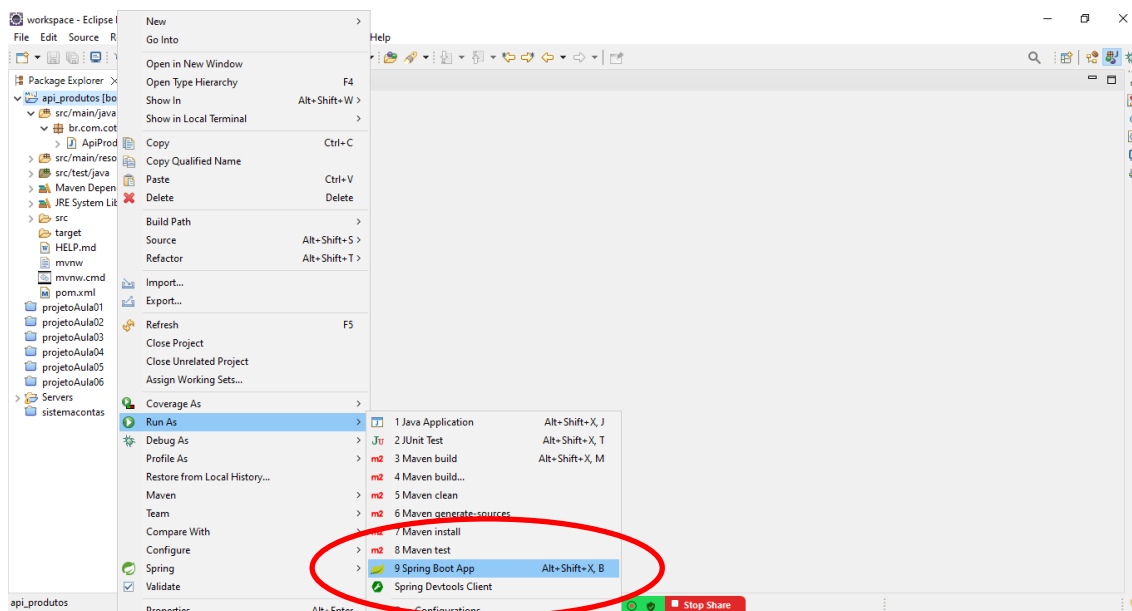
```
</plugins>
```

```
</build>
```

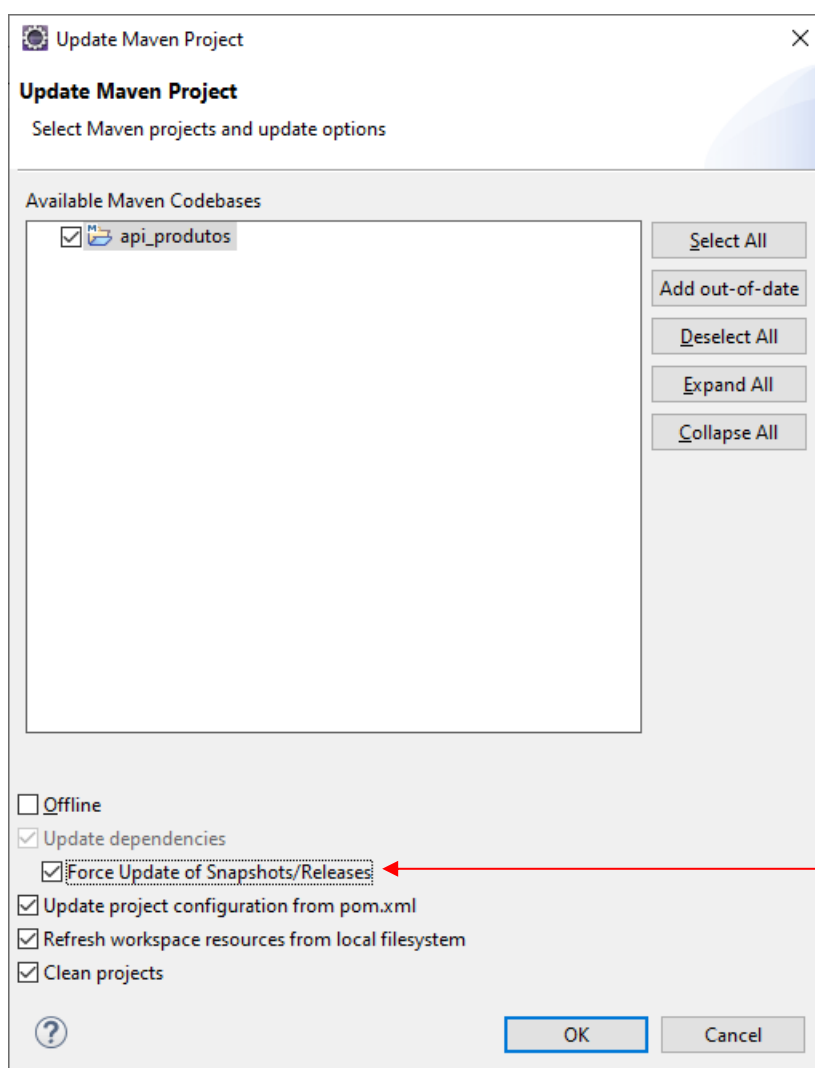
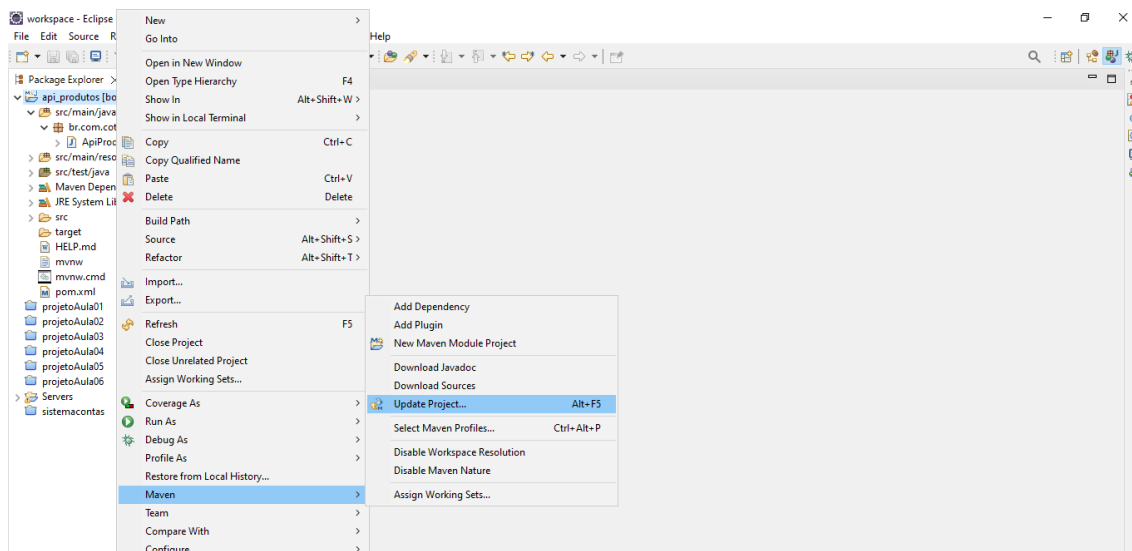
```
</project>
```

**Os projetos criados com SpringBoot já possuem um servidor TOMCAT "embutido" no projeto.**

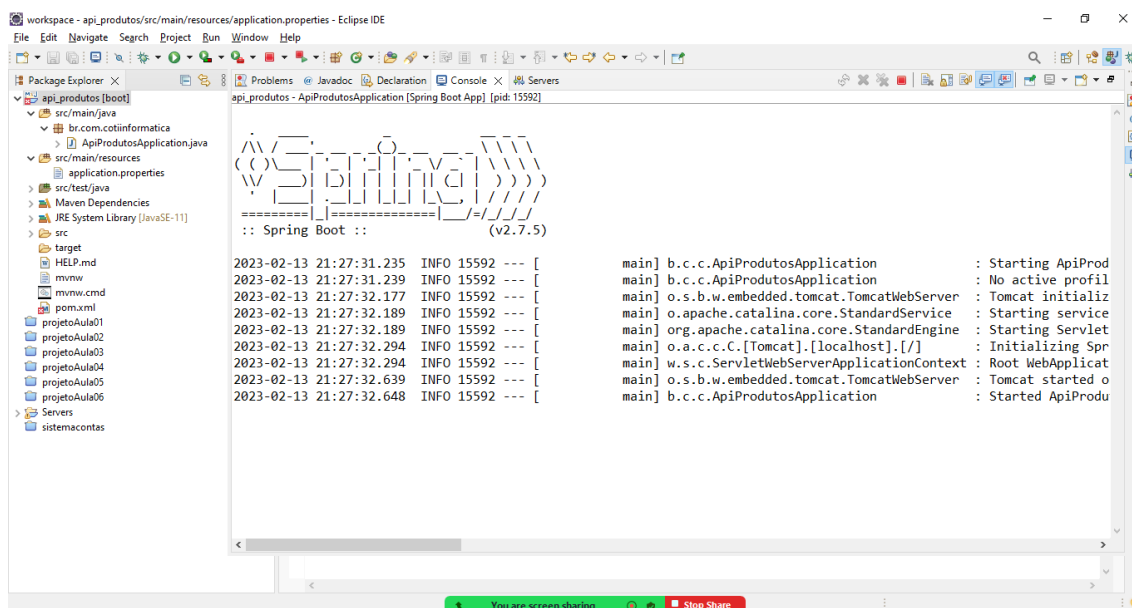
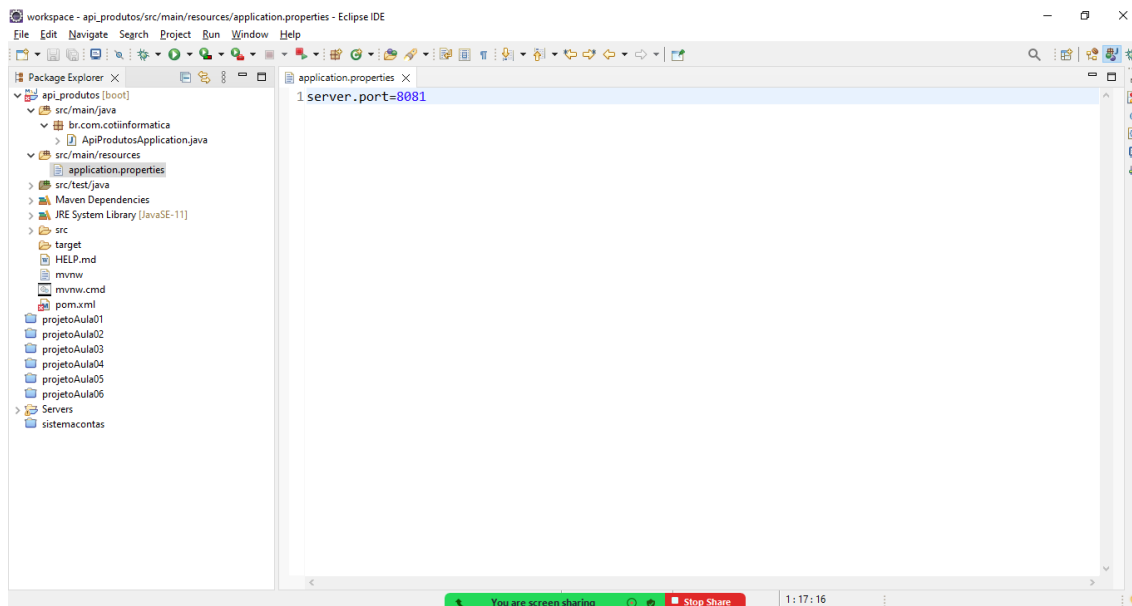
Basta rodar o projeto, que ele será executado neste servidor tomcat que já esta embarcado na aplicação.



## Recurso para forçar o MAVEN a reinstalar todas as bibliotecas do projeto:



## Modificando a porta de execução do TOMCAT: /src/main/resources/application.properties



```

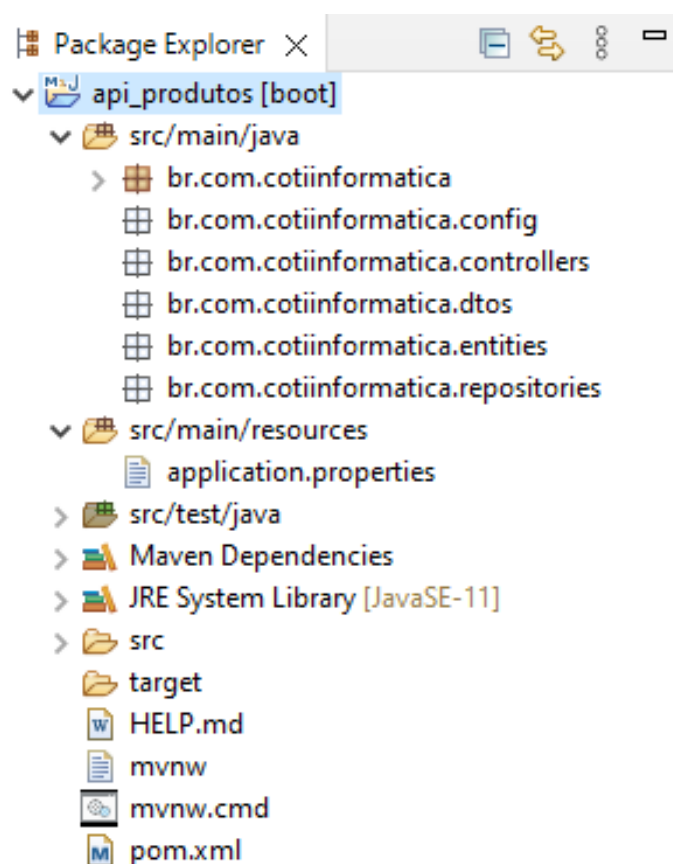
:: Spring Boot :: (v2.7.5)

2023-02-13 21:28:19.145 INFO 26532 --- [main] b.c.c.ApiProdutosApplication : Starting
ApiProdutosApplication using Java 11.0.10 on DESKTOP-P9F6D9F with PID 26532 (C:\Users\Samsung\Desktop\COTI -
Aulas\2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)\workspace\api_produtos\target\classes started by
Samsung in C:\Users\Samsung\Desktop\COTI - Aulas\2023 - Java WebDeveloper SQS 18h as 22h (Início em
09.01)\workspace\api_produtos)
2023-02-13 21:28:19.148 INFO 26532 --- [main] b.c.c.ApiProdutosApplication : No active
profile set, falling back to 1 default profile: "default"
2023-02-13 21:28:20.009 INFO 26532 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
initialized with port(s): 8081 (http)

```

```
2023-02-13 21:28:20.018 INFO 26532 --- [main] o.apache.catalina.core.StandardService : Starting
service [Tomcat]
2023-02-13 21:28:20.019 INFO 26532 --- [main] org.apache.catalina.core.StandardEngine : Starting
Servlet engine: [Apache Tomcat/9.0.68]
2023-02-13 21:28:20.112 INFO 26532 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] :
Initializing Spring embedded WebApplicationContext
2023-02-13 21:28:20.112 INFO 26532 --- [main] w.s.c.ServletWebServerApplicationContext : Root
WebApplicationContext: initialization completed in 917 ms
2023-02-13 21:28:20.450 INFO 26532 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat
started on port(s): 8081 (http) with context path ''
2023-02-13 21:28:20.462 INFO 26532 --- [main] b.c.c.ApiProdutosApplication : Started
ApiProdutosApplication in 1.656 seconds (JVM running for 2.571)
```

## Criando a estrutura do projeto:



## Swagger

Gerando documentações online para as nossas APIs.

O **Swagger** é um framework composto por diversas ferramentas que, independente da linguagem, auxilia a descrição, consumo e visualização de serviços de uma API REST. A **especificação** da API consiste em determinar os modelos de dados que serão entendidos pela API e as funcionalidades presentes na mesma. Para cada funcionalidade, é preciso especificar o seu nome, os parâmetros que devem ser passados no momento de sua invocação e os valores que irão ser retornados aos usuários da API.

Com o Swagger UI, a partir da especificação da API, podemos criar documentações elegantes e acessíveis ao usuário, permitindo assim uma

compreensão maior da API, pois além de poder ver os endpoints e modelos das entidades com seus atributos e respectivos tipos, o módulo de UI possibilita que os usuários da API interajam intuitivamente com a API

## /pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.5</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>br.com.cotiinformatica</groupId>
  <artifactId>api_produtos</artifactId>
  <version>1.0</version>
  <name>api_produtos</name>
  <description>API Produtos</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-swagger-ui</artifactId>
      <version>3.0.0</version>
    </dependency>

  </dependencies>
</project>
```



```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-boot-starter</artifactId>
  <version>3.0.0</version>
</dependency>

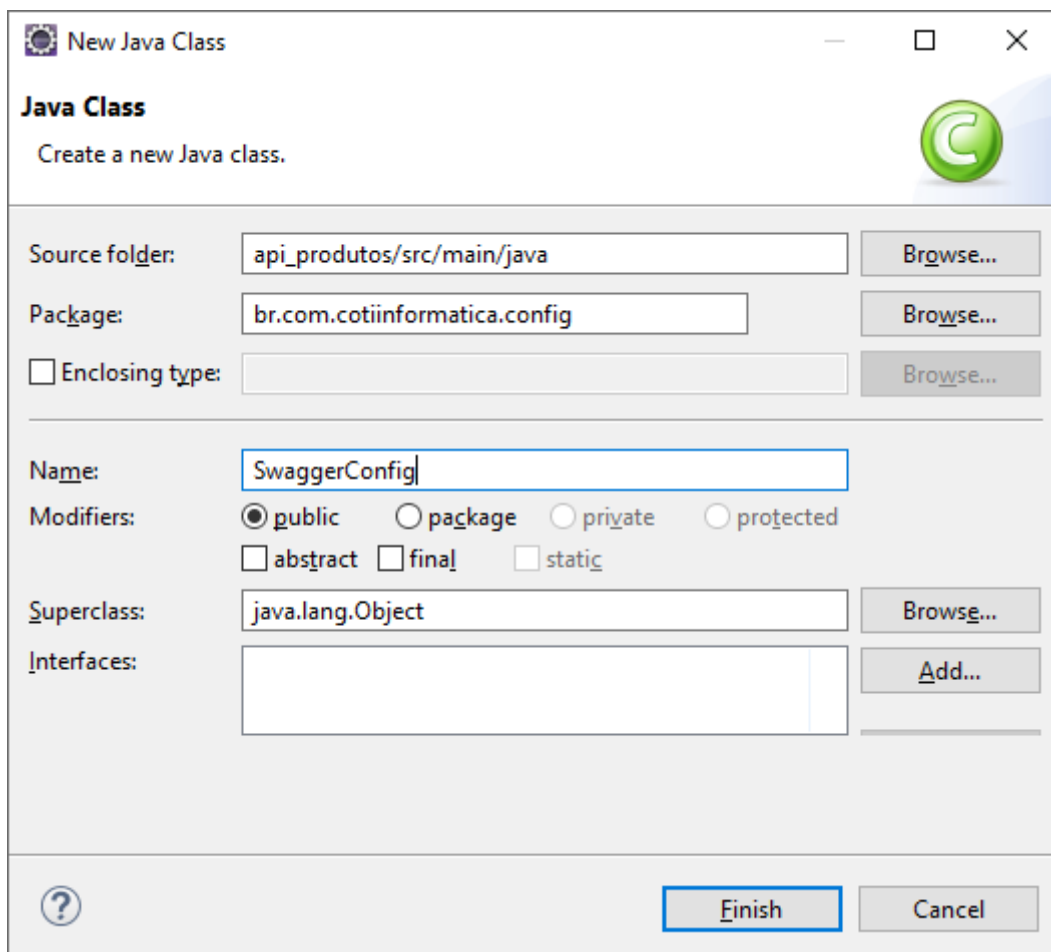
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

</project>
```

## /config/**SwaggerConfig.java**

Criando uma classe para configurar a geração da documentação do Swagger.



**New Java Class**

Java Class  
Create a new Java class.

Source folder:  [Browse...](#)

Package:  [Browse...](#)

☐ Enclosing type:  [Browse...](#)

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:  [Browse...](#)

Interfaces:  [Add...](#)

[Finish](#) [Cancel](#)

```
package br.com.cotiinformatica.config;

import java.util.Collections;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

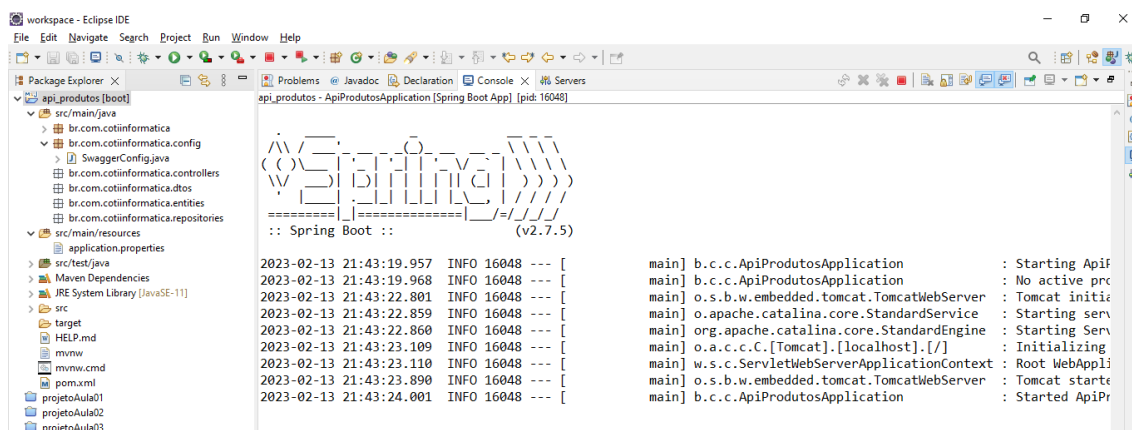
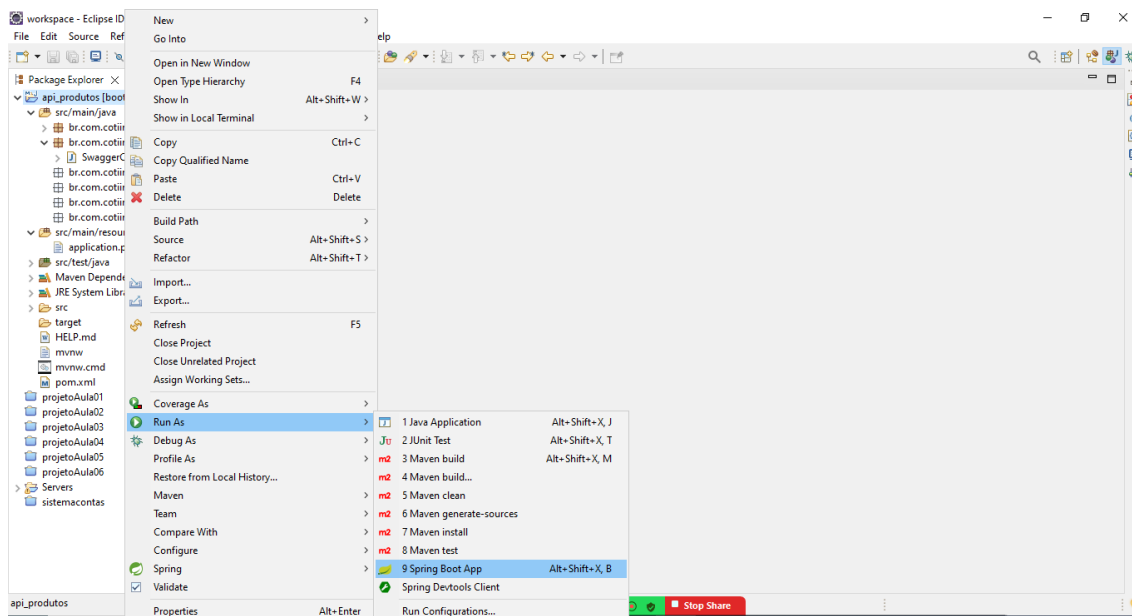
@Configuration
@EnableWebMvc
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket api() {

        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.basePackage("br.com.cotiinformatica"))
            .paths(PathSelectors.ant("/**"))
            .build()
            .apiInfo(apiInfo());
    }

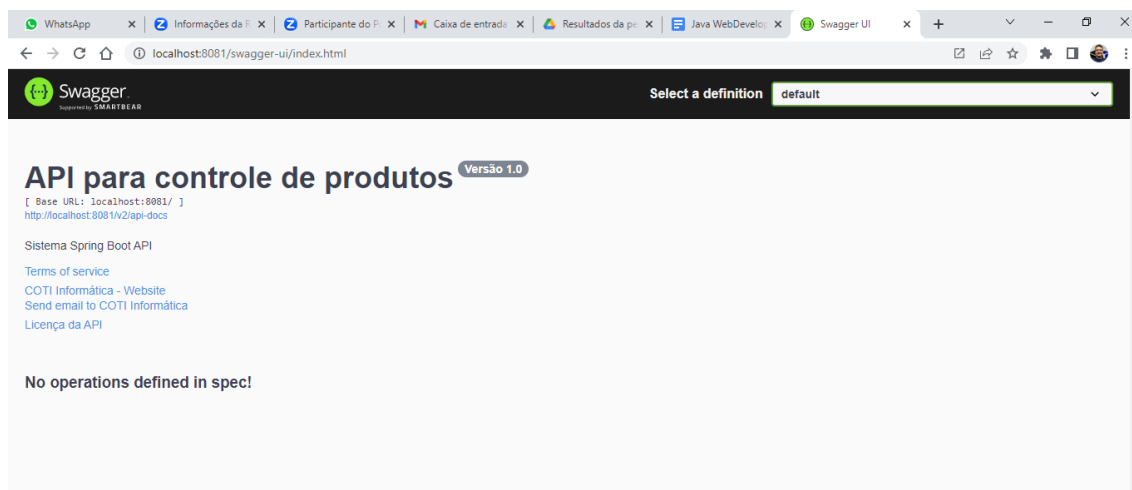
    private ApiInfo apiInfo() {
        return new ApiInfo(
            "API para controle de produtos",
            "Sistema Spring Boot API",
            "Versão 1.0",
            "http://www.cotiinformatica.com.br",
            new Contact("COTI Informática",
                "http://www.cotiinformatica.com.br",
                "contato@cotiinformatica.com.br"),
            "Licença da API",
            "http://www.cotiinformatica.com.br",
            Collections.emptyList()
        );
    }
}
```

## Executando o projeto e visualizando a sua documentação:

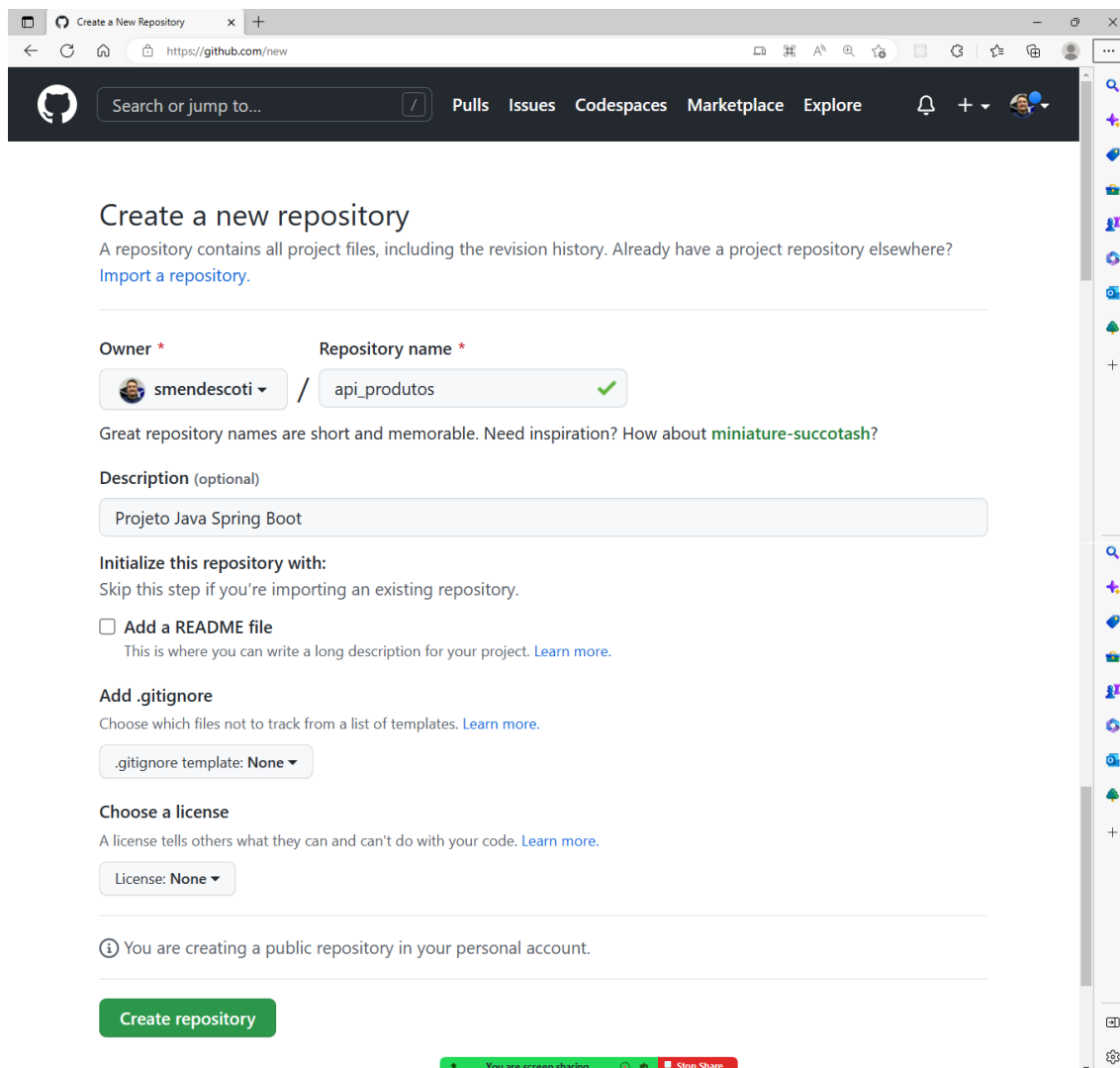


## Acessando a página de documentação do Swagger:

<http://localhost:8081/swagger-ui/index.html>



## Publicando o projeto no GITHUB: Criando um repositório:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \* smendescoti / Repository name \* api\_produtos ✓

Great repository names are short and memorable. Need inspiration? How about [miniature-succotash](#)?

Description (optional)  
Projeto Java Spring Boot

Initialize this repository with:  
Skip this step if you're importing an existing repository.

☒ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)  
.gitignore template: None

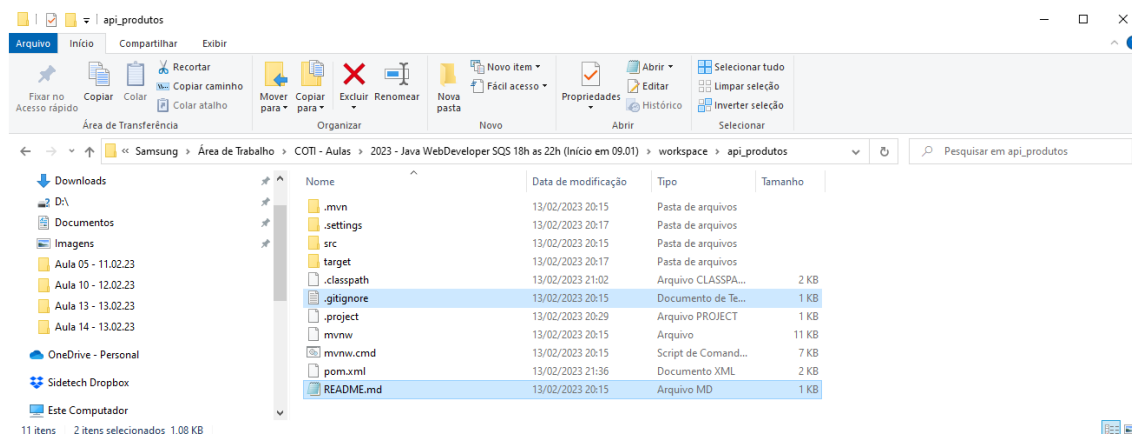
Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)  
License: None

*i* You are creating a public repository in your personal account.

[Create repository](#)

You are screen sharing [Stop Share](#)

O projeto criado no Spring BOOT já possui  
os arquivos **.gitignore** e **README.md**



```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 -  
Java webDeveloper SQS 18h as 22h (Início em  
09.01)/workspace/api_produtos  
$ git init
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 -  
Java webDeveloper SQS 18h as 22h (Início em  
09.01)/workspace/api_produtos (master)  
$ git branch -m main
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 -  
Java webDeveloper SQS 18h as 22h (Início em  
09.01)/workspace/api_produtos (main)  
$ git add .
```

### Fazendo o COMMIT e PUSH:

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 -  
Java webDeveloper SQS 18h as 22h (Início em  
09.01)/workspace/api_produtos (main)  
$ git commit -m 'first commit'
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java webDeveloper SQS 18h as 22h (Início em 09.01)/workspace  
/api_produtos (main)  
$ git commit -m 'first commit'  
[main (root-commit) 4d717e1] first commit  
11 files changed, 691 insertions(+)  
create mode 100644 .gitignore  
create mode 100644 .mvn/wrapper/maven-wrapper.jar  
create mode 100644 .mvn/wrapper/maven-wrapper.properties  
create mode 100644 README.md  
create mode 100644 mvnw  
create mode 100644 mvnw.cmd  
create mode 100644 pom.xml  
create mode 100644 src/main/java/br/com/cotiinformatica/ApiProdutosApplication.java  
create mode 100644 src/main/java/br/com/cotiinformatica/config/SwaggerConfig.java  
create mode 100644 src/main/resources/application.properties  
create mode 100644 src/test/java/br/com/cotiinformatica/ApiProdutosApplicationTests.java
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 -  
Java webDeveloper SQS 18h as 22h (Início em  
09.01)/workspace/api_produtos (main)  
$ git remote add origin  
https://github.com/smendescoti/api_produtos.git
```

```
Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 -  
Java webDeveloper SQS 18h as 22h (Início em  
09.01)/workspace/api_produtos (main)  
$ git push -u origin main
```

Continua...