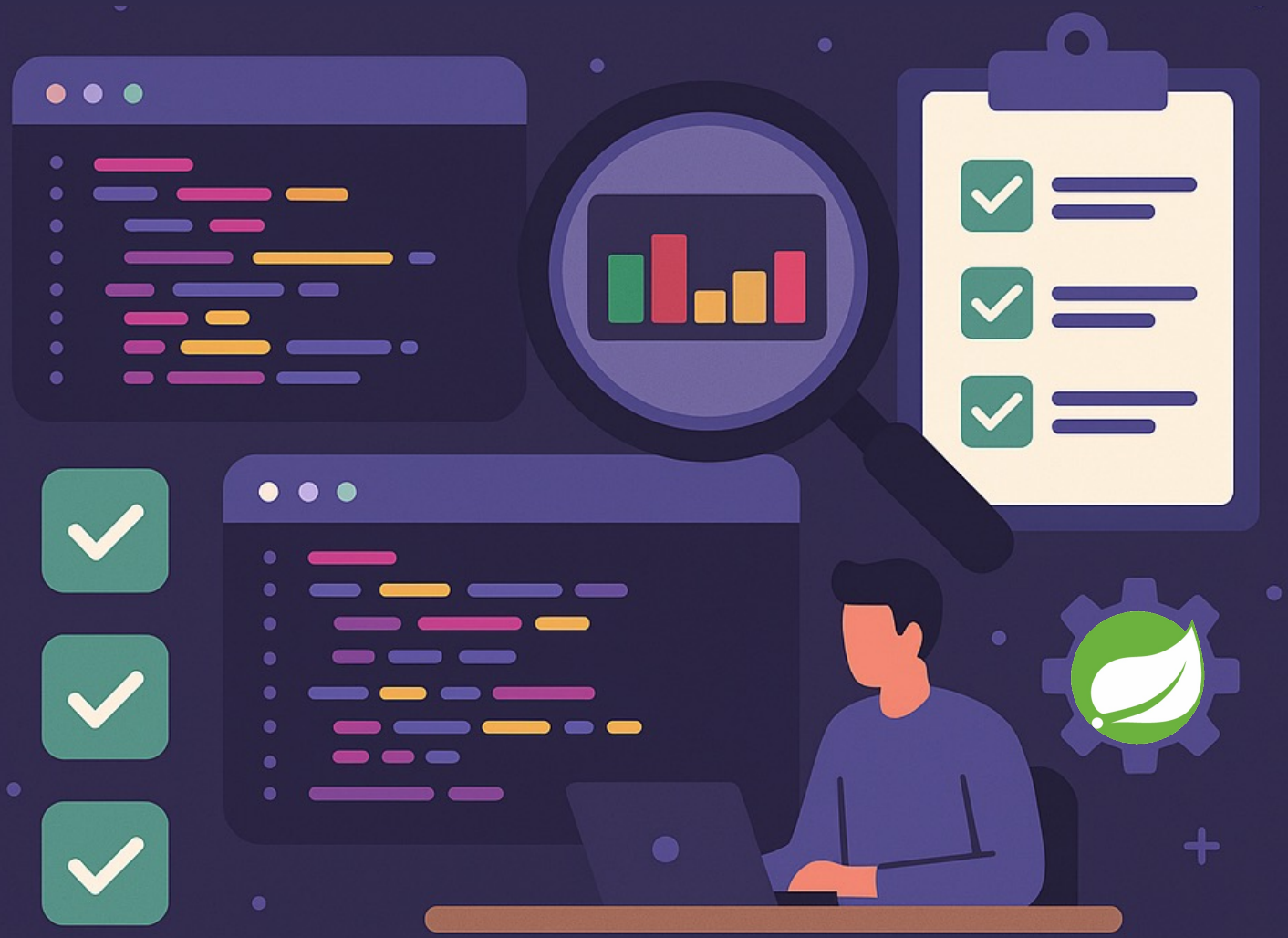
Qualidade - Cobertura de Código & Testes



Alberto Yano

Introdução

O que vamos ver neste e-book?

Em um ambiente corporativo é exigido no desenvolvimento de software qualidade de código, documentação, testabilidade, cobertura de testes, segurança, manutenibilidade, modularização, dentre outros.

Assim, é comumente utilizado o software Sonar para a verificação de alguns desses itens.

Dessa forma, para agilizar o processo de desenvolvimento é recomendável utilizar o Sonar para verificação do código antes de subí-lo para um repositório.

Então vamos ver como isso funciona!?

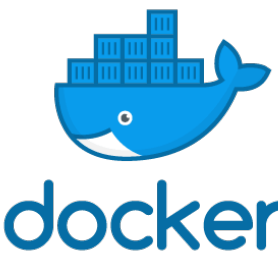
Introdução

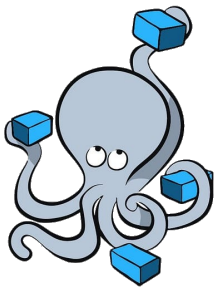
Escopo

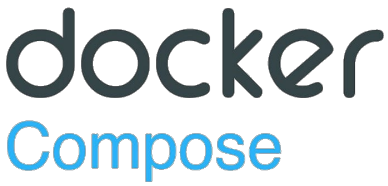
Utilizaremos o Docker e o Docker Compose para rodar o Sonar configurando-o para utilizar o mesmo profile usado em uma empresa.

Criaremos um projeto no Sonar com uma pequena API REST Java utilizando Spring Boot.

E por fim, faremos um teste unitário e veremos o que o Sonar nos informa do projeto.







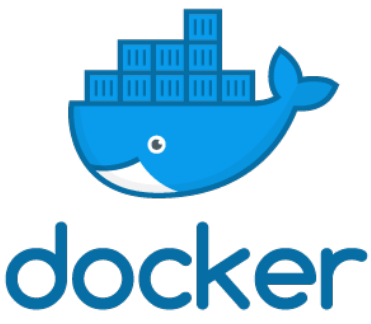
Docker e Docker Compose

Docker e Docker Compose

O que é Docker?

Docker é uma plataforma *open source* que permite a criação, execução e o gerenciamento de **aplicações em containers**. Os Containers são unidades leves, portáteis e isoladas que empacotam uma aplicação com todas as suas **dependências, bibliotecas e configurações**, garantindo que ela funcione de maneira consistente em qualquer ambiente.

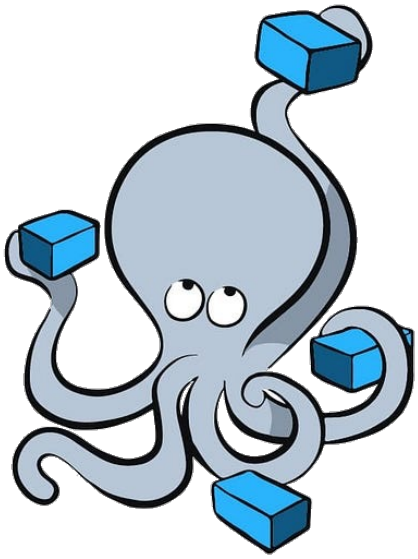
O **Docker** simplifica o processo de desenvolvimento e implantação, evitando o clássico problema do "na minha máquina funciona".



Docker e Docker Compose

O que é Docker Compose?

Docker Compose é uma ferramenta complementar ao Docker que permite definir e gerenciar **múltiplos containers** de forma coordenada, utilizando um arquivo de configuração no formato YAML (docker-compose.yml).



docker
Compose

Docker e Docker Compose

Instalação

Não é o objetivo deste e-book ensinar a instalação do Docker e do Docker Compose, mas eles serão necessários para a utilização do Sonar.

Então seguem os links para a instalação:

(Windows)
https://docs.docker.com/desktop/setup/install/windows-install/

(Linux)
https://docs.docker.com/desktop/setup/install/linux/

(Mac)
https://docs.docker.com/desktop/setup/install/mac-install/

02

Sonar ou SonarQube

Sonar

O que é o Sonar ou SonarQube?

O **SonarQube** é uma plataforma open source projetada para a **inspeção contínua da qualidade de código** em diversos tipos de linguagens de programação, como Java, JavaScript, C#, Python, entre outras. Ele automatiza a análise estática do código-fonte, identificando problemas relacionados à **manutenibilidade, segurança, confiabilidade e complexidade**.

Sonar

Objetivo do Sonar

Seu principal objetivo é fornecer um **feedback rápido e contínuo** aos desenvolvedores, ajudando a prevenir a introdução de bugs, vulnerabilidades e más práticas de codificação ao longo do ciclo de desenvolvimento.

Além disso, o SonarQube promove a **melhoria contínua** ao permitir que equipes definam **padrões de qualidade** e acompanhem a sua evolução por meio de **painéis** e **relatórios detalhados**.

03

Instalando o Sonar

Sonar

Instalação do Sonar

A forma mais fácil para a **instalação do Sonar** é baixar o arquivo docker-compose.yml no seguinte link:

https://github.com/SonarSource/docker-sonarqube/tree/master/example-compose-files/sq-with-postgres

E no diretório onde está o arquivo executar a seguinte linha de comando:

prompt

```
docker-compose up -d
[+] Running 9/9
 ✔ Network sonar_ipv4                        Created
 ✔ Volume "sonar_sonarqube_temp"             Created
 ✔ Volume "sonar_postgresql"                 Created
 ✔ Volume "sonar_postgresql_data"            Created
 ✔ Volume "sonar_sonarqube_data"             Created
 ✔ Volume "sonar_sonarqube_extensions"       Created
 ✔ Volume "sonar_sonarqube_logs"             Created
 ✔ Container postgresql                      Healthy
 ✔ Container sonarqube                       Started
```

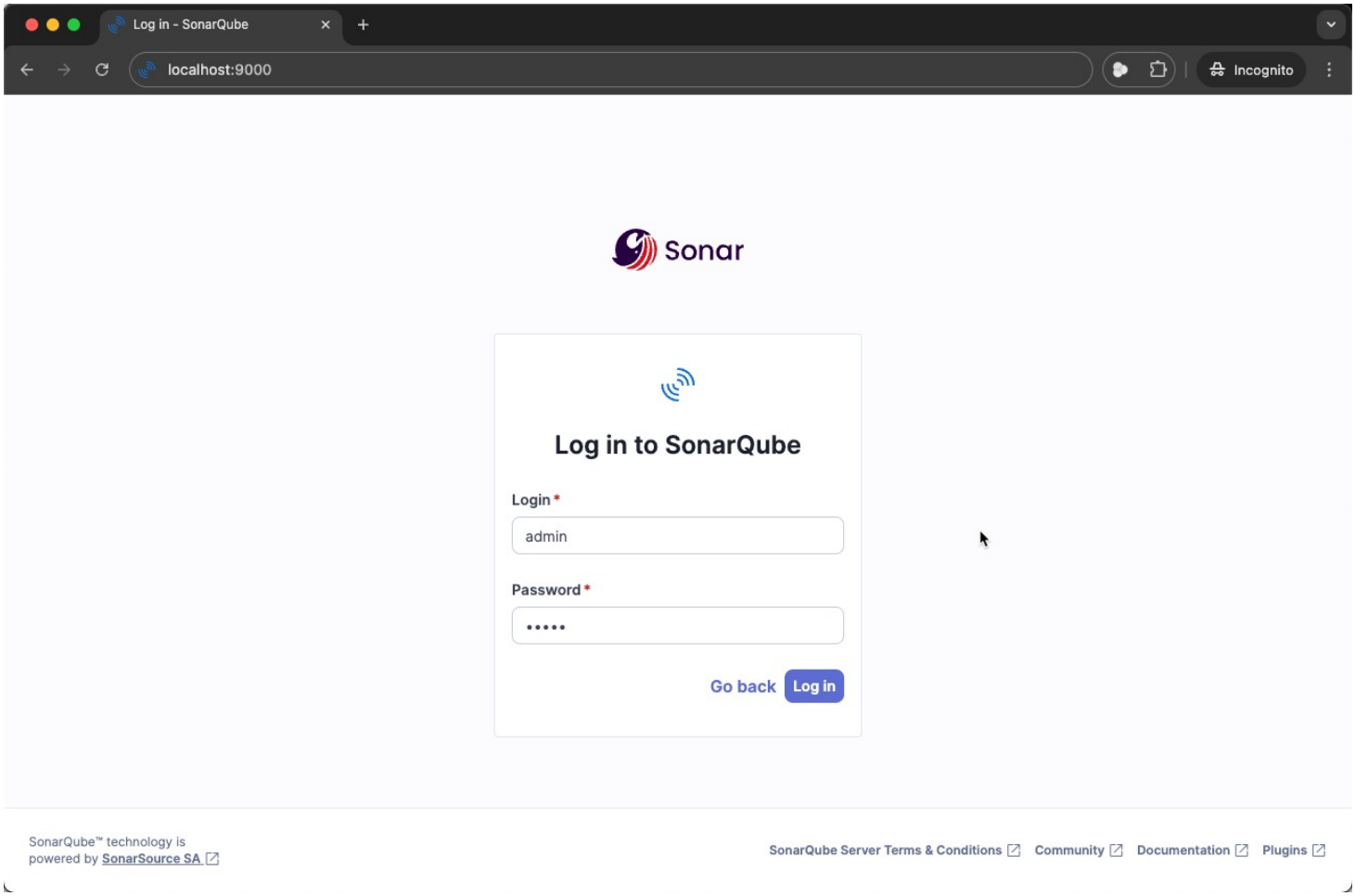
Sonar

Acessando o Sonar

Utilize a seguinte URL no seu navegador:

http://localhost:9000/

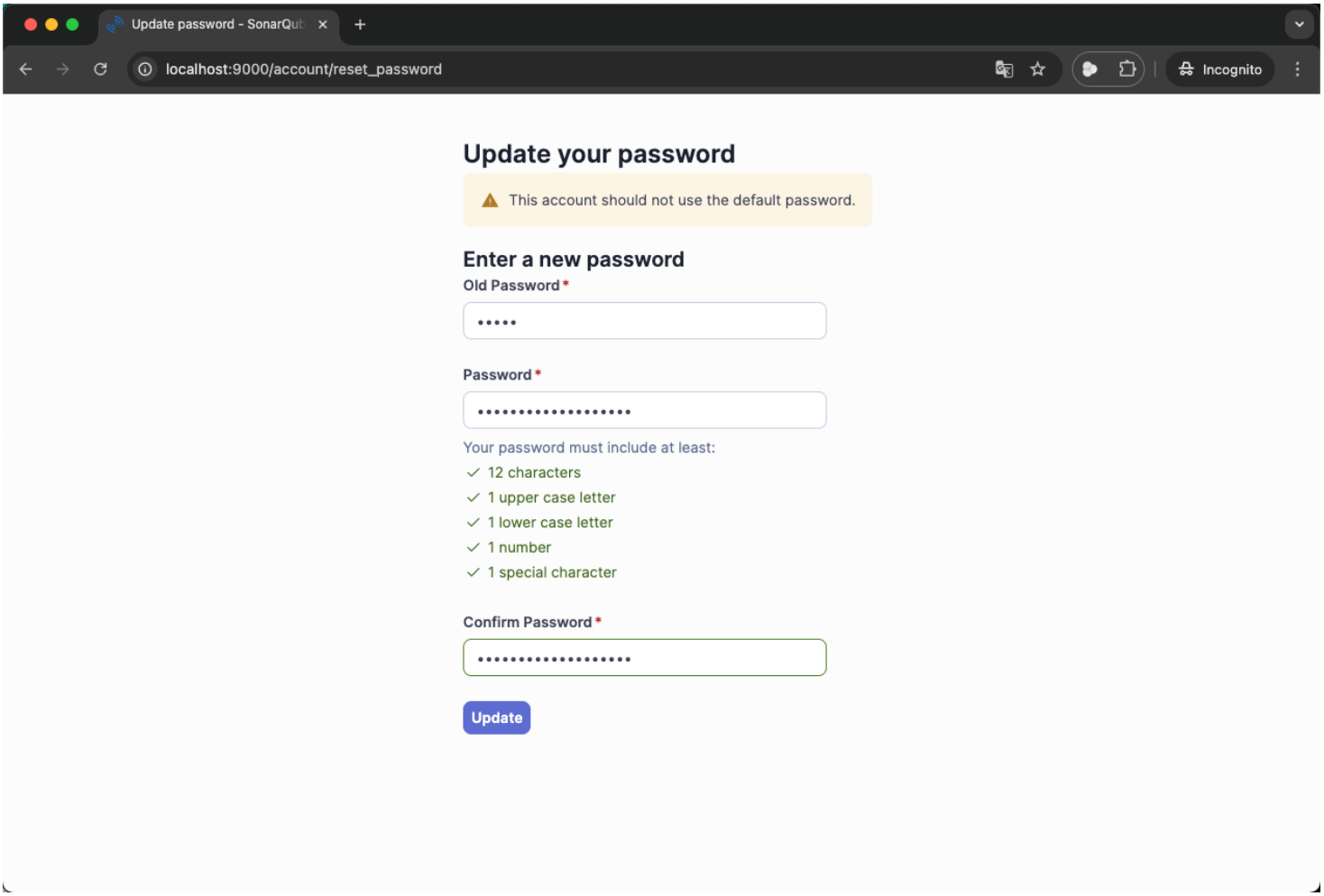
Utilize o usuário e senha “admin”



Sonar

Acessando o Sonar

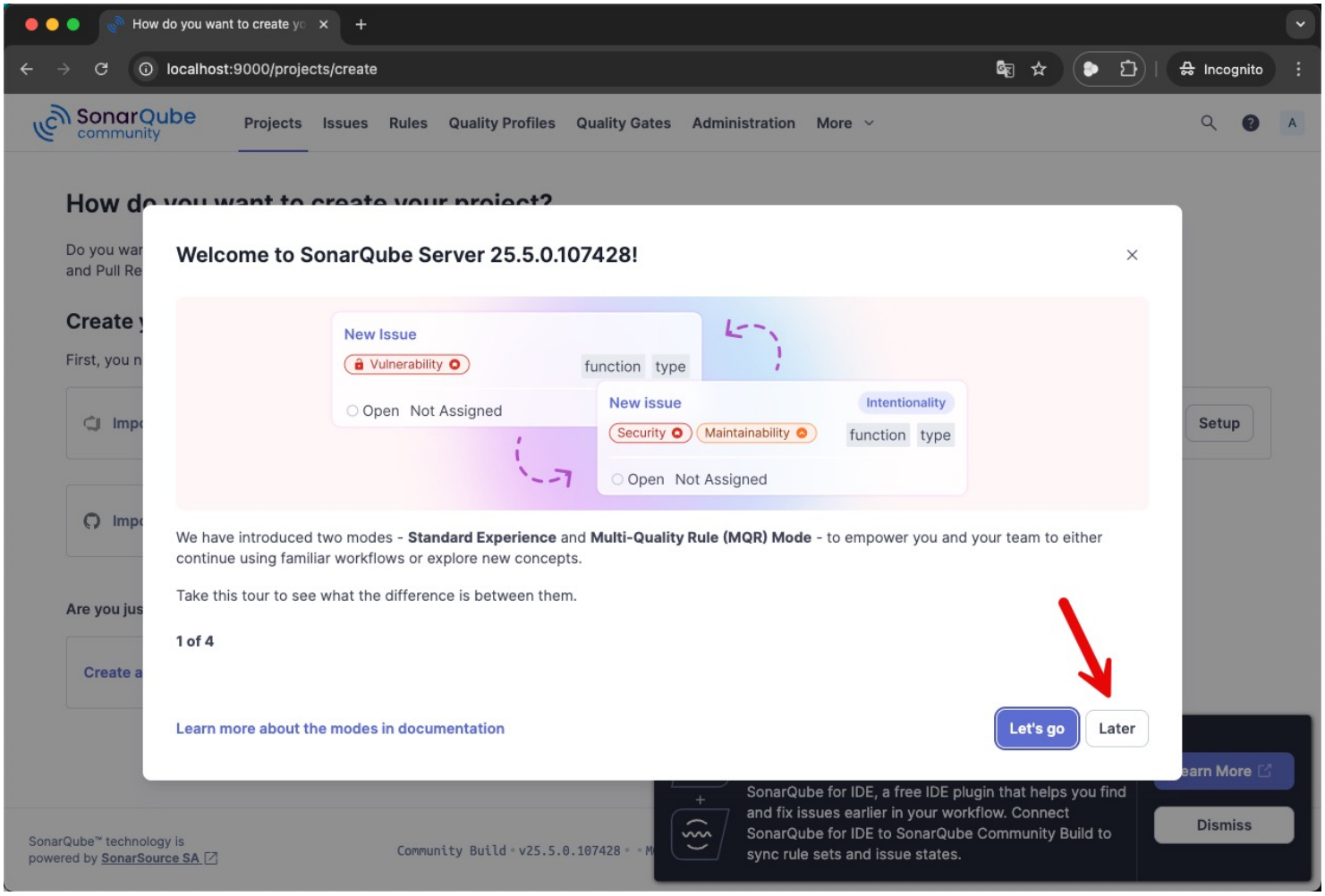
Será solicitada a troca da senha, que precisará atender às exigências de uma senha forte.



Sonar

Acessando o Sonar

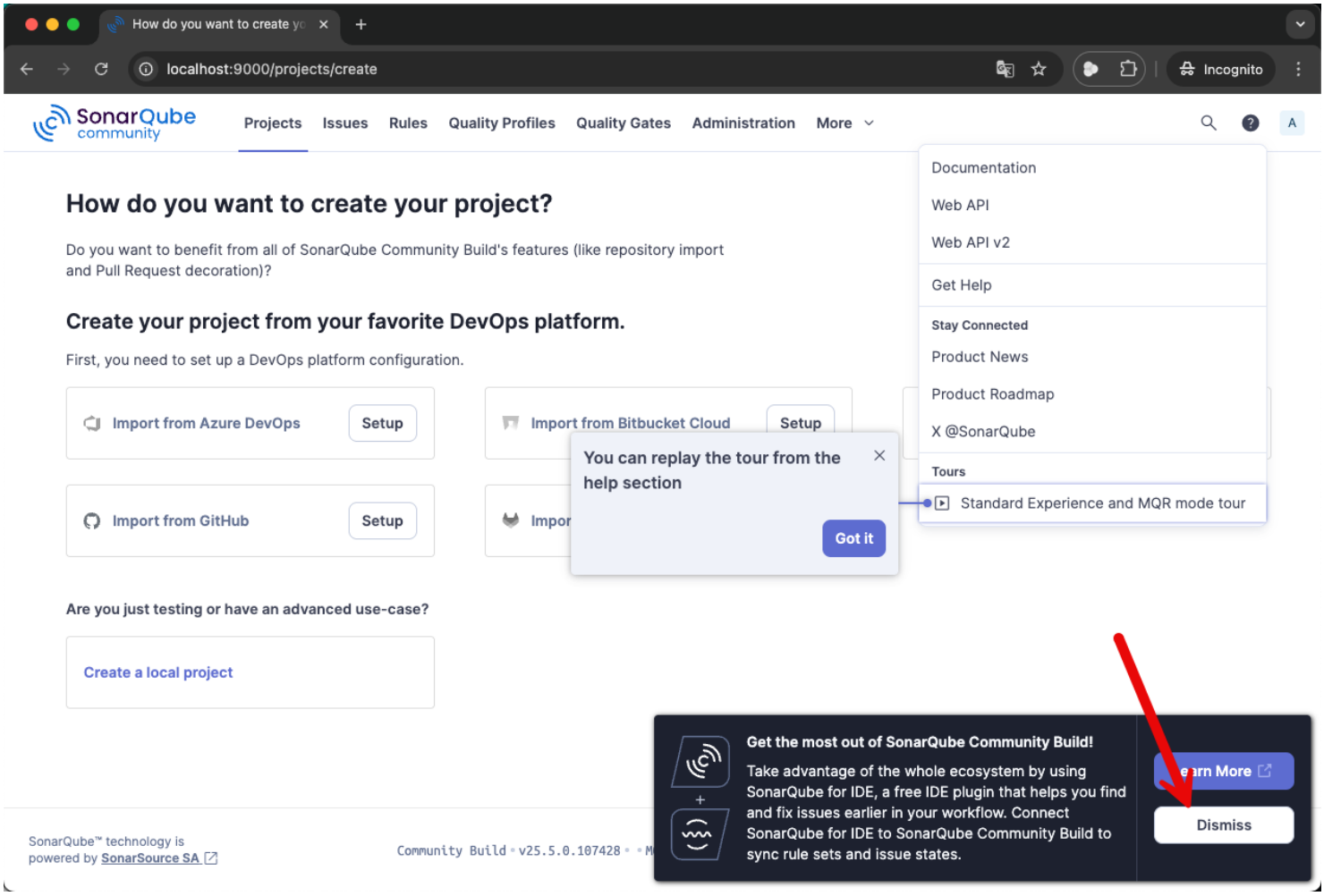
Clique em “*Later*”



Sonar

Acessando o Sonar

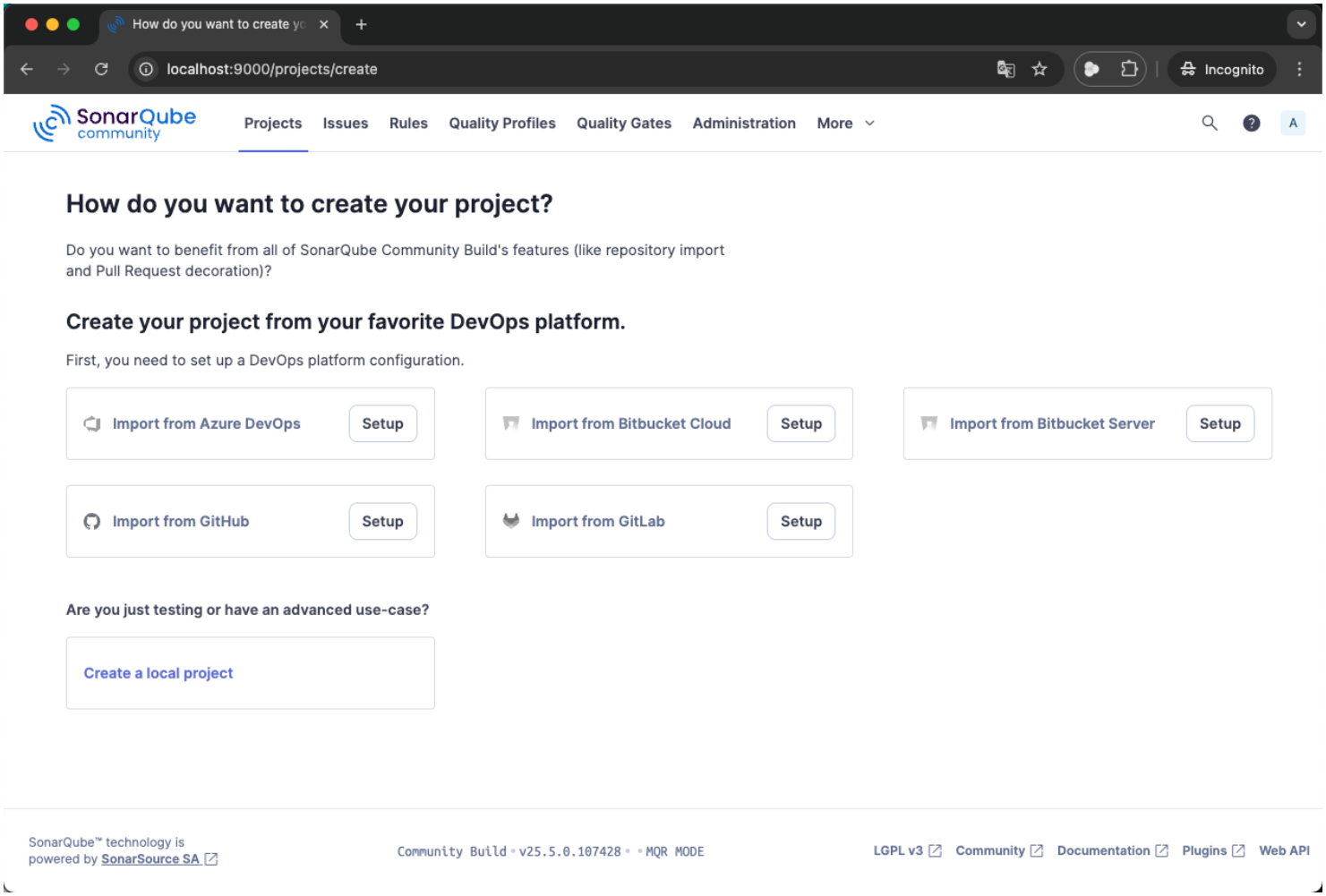
Clique em “*Dismiss*”



Sonar

Acessando o Sonar

Pronto seu Sonar estará instalado ...!!!



04

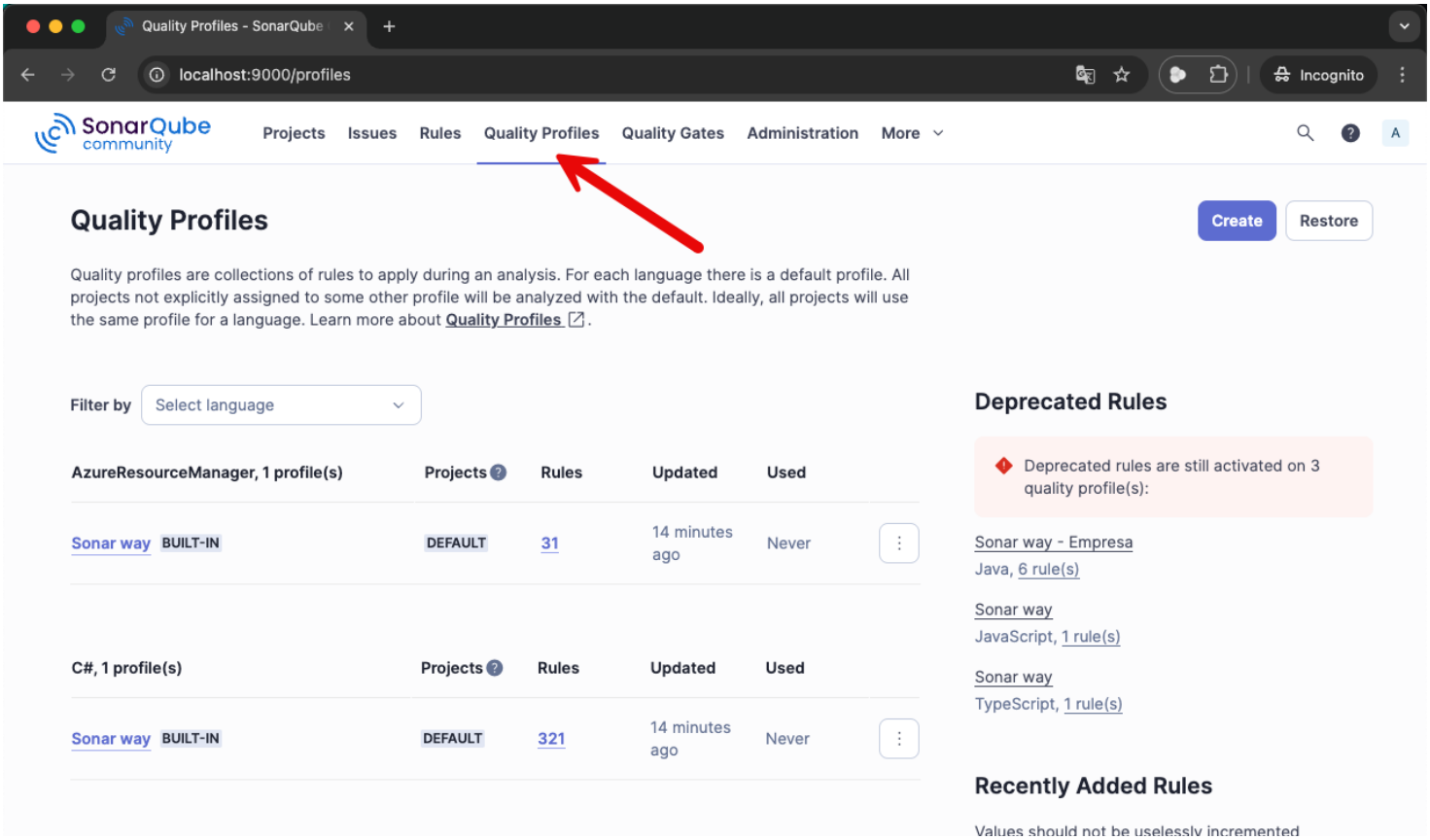
Configurando o Sonar

Sonar

Configurando o Sonar

Muito provavelmente sua empresa tem um sonar configurado com as políticas de verificações, então precisaremos importá-las.

Acesse o Sonar da sua empresa e siga os seguintes passos:

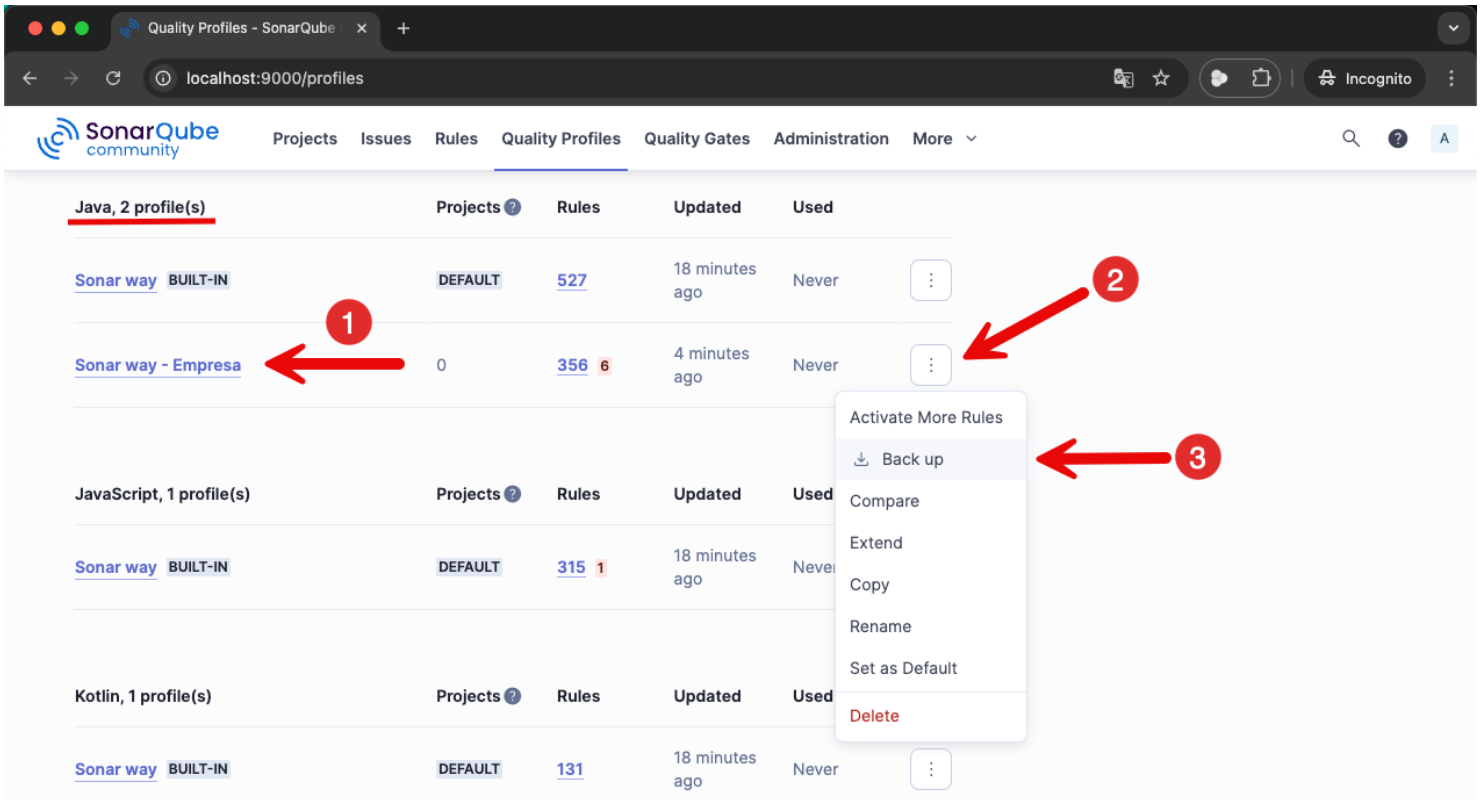


Sonar

Configurando o Sonar

Neste exemplo, exportaremos o *profile* Java da Empresa.

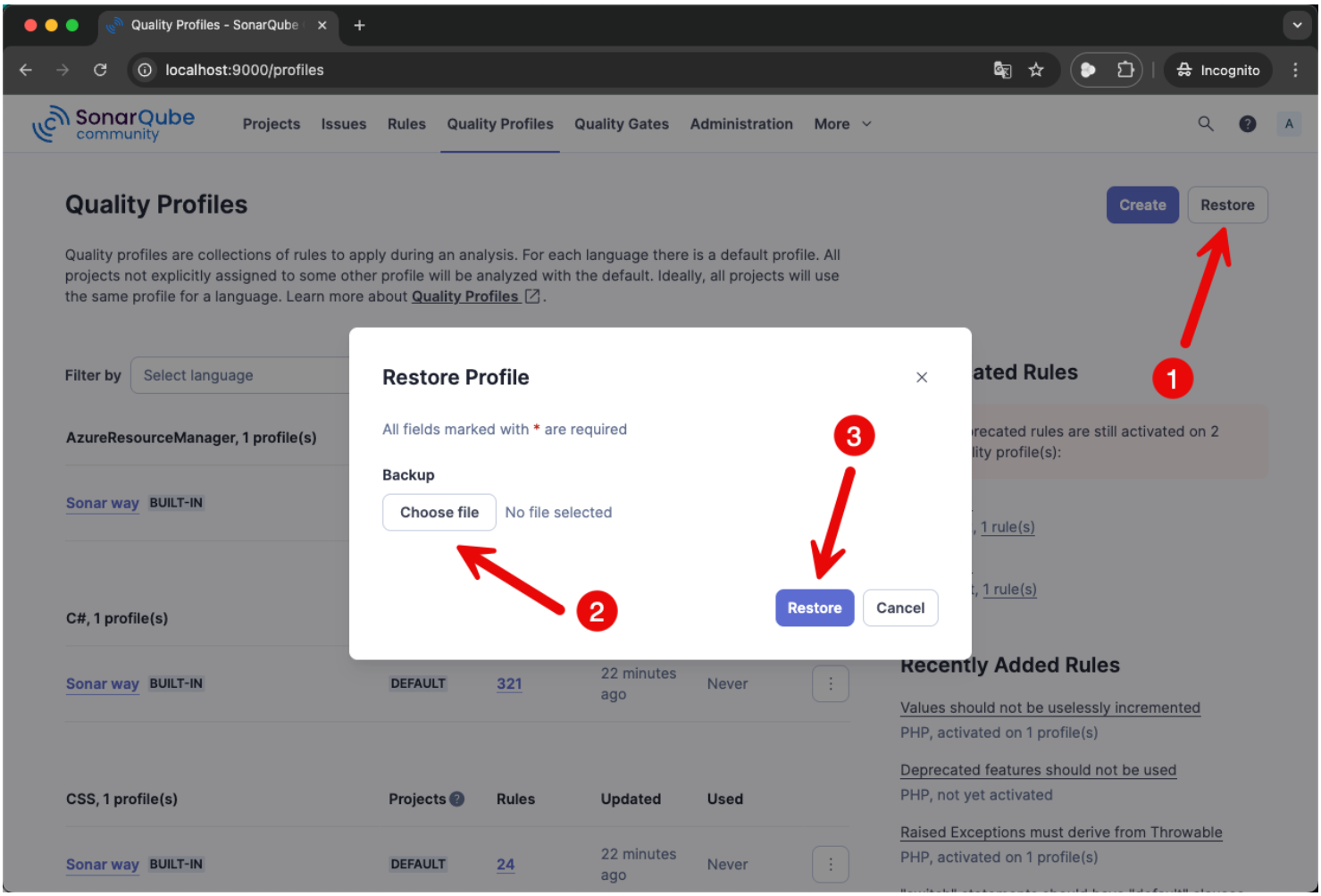
Salve em um diretório do seu computador.



Sonar

Configurando o Sonar

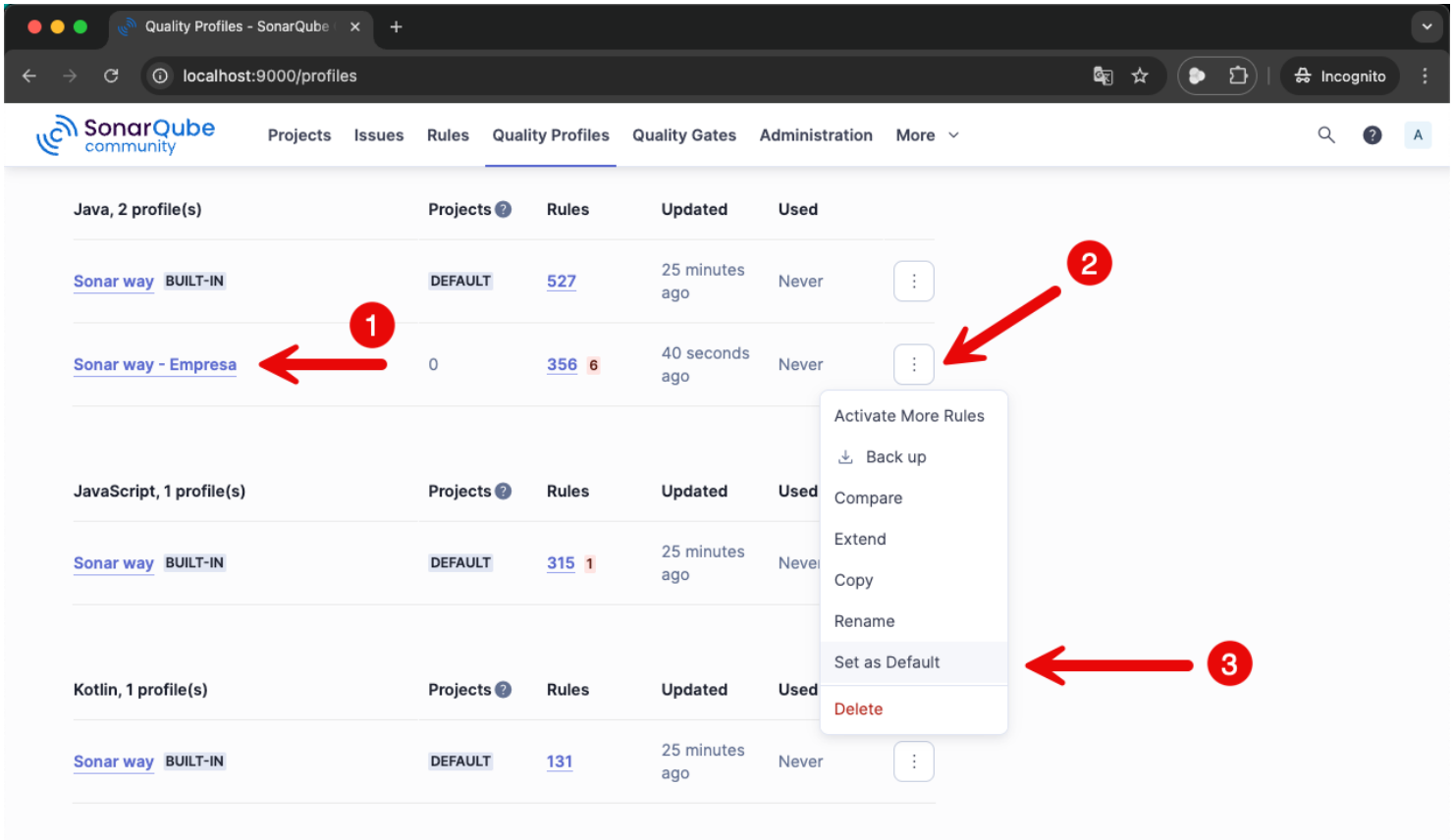
Volte ao Sonar do seu computador e siga os seguintes passos selecionando o arquivo salvo.



Sonar

Configurando o Sonar

Depois deixe o *profile* como sendo o *default*, para que ao enviar os dados para o Sonar ele utilize o *profile* da empresa.



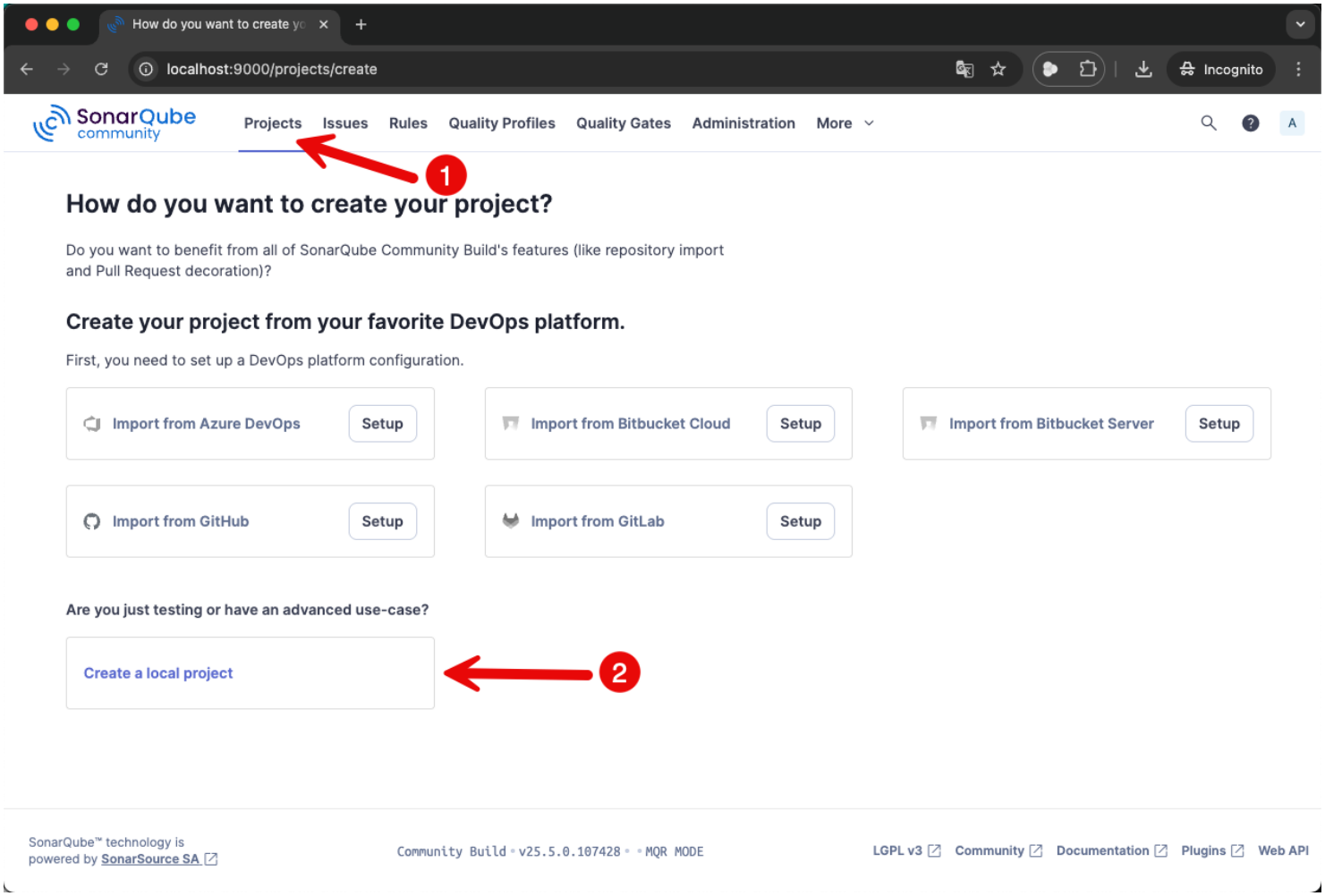
05

Criando um Projeto no Sonar

Sonar

Criando um Projeto no Sonar

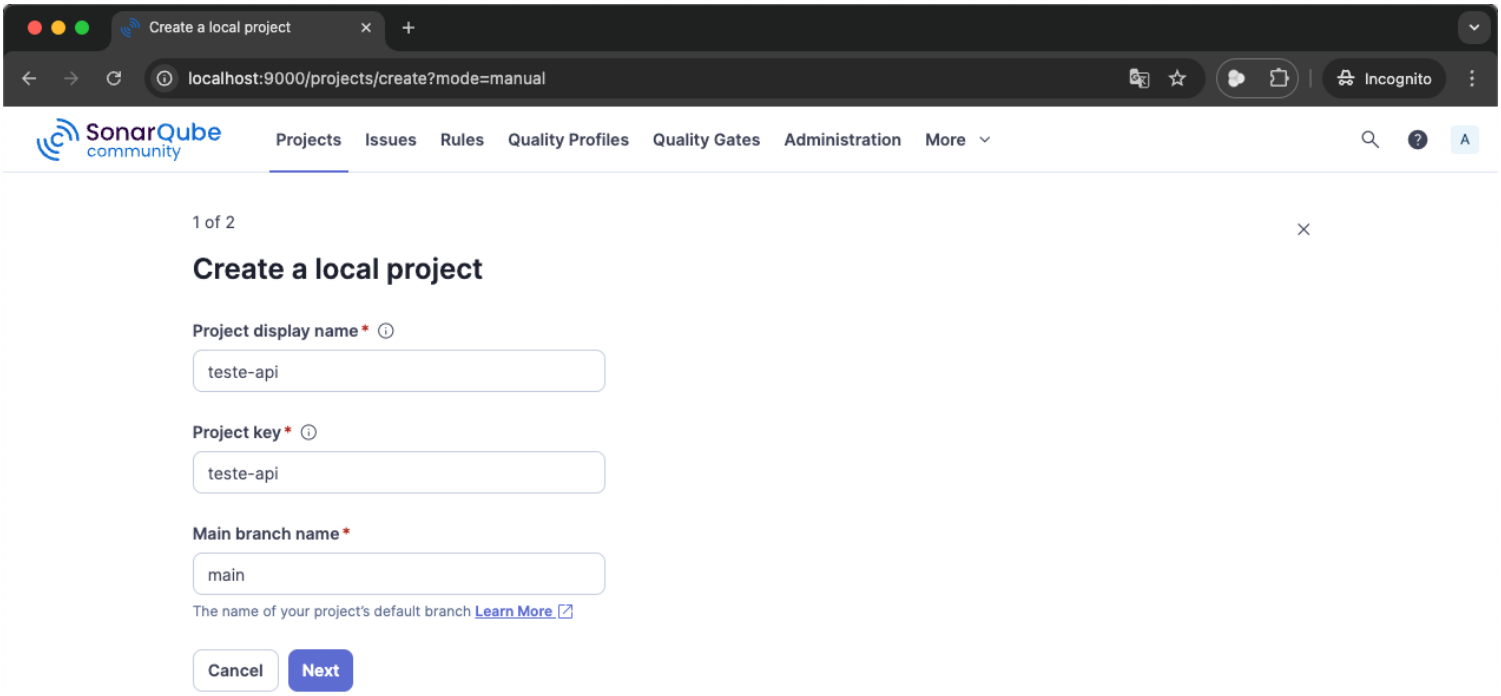
Clique em “*Projects*” e depois em “*Create a local project*”.



Sonar

Criando um Projeto no Sonar

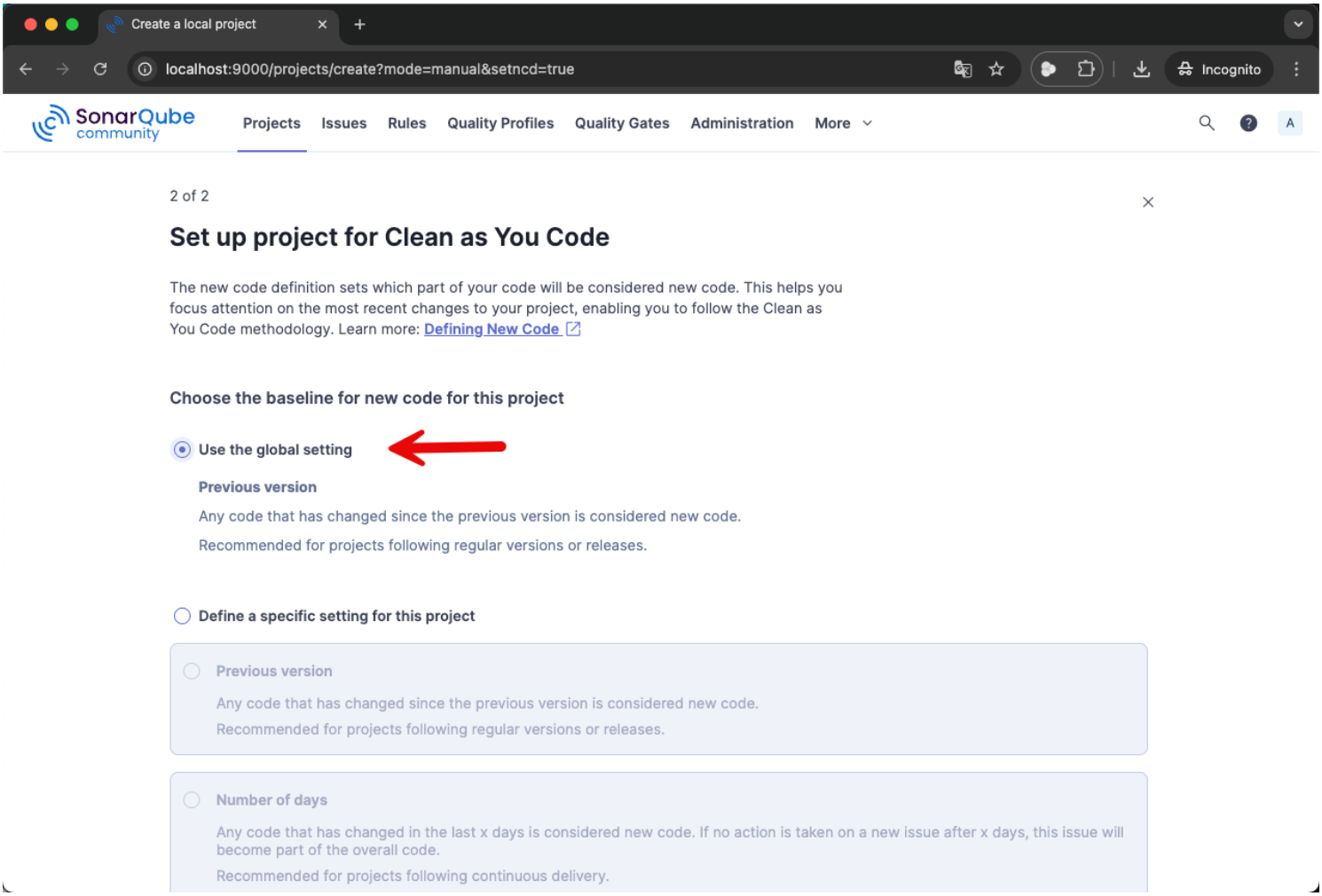
Coloque o “*Project Display name*” e o “*Project key*”. Preencha com todas as letras minúsculas e sem espaço.



Sonar

Criando um Projeto no Sonar

Selecione “*Use the global setting*” e clique em “*Create project*”.

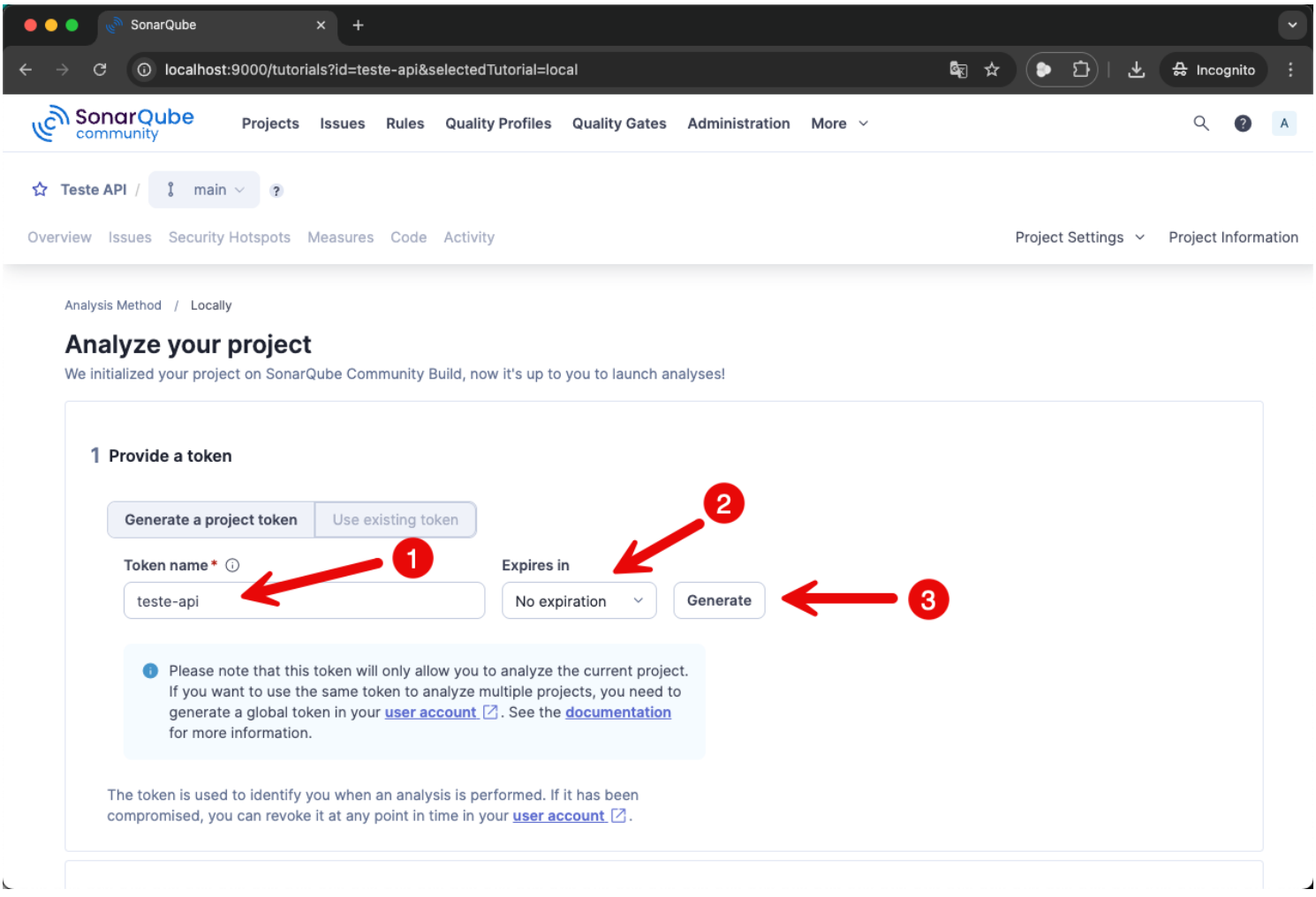


Sonar

Criando um Projeto no Sonar

Coloque o “*Token name*” igual ao “*Project key*” tudo em minúsculo e sem espaço.

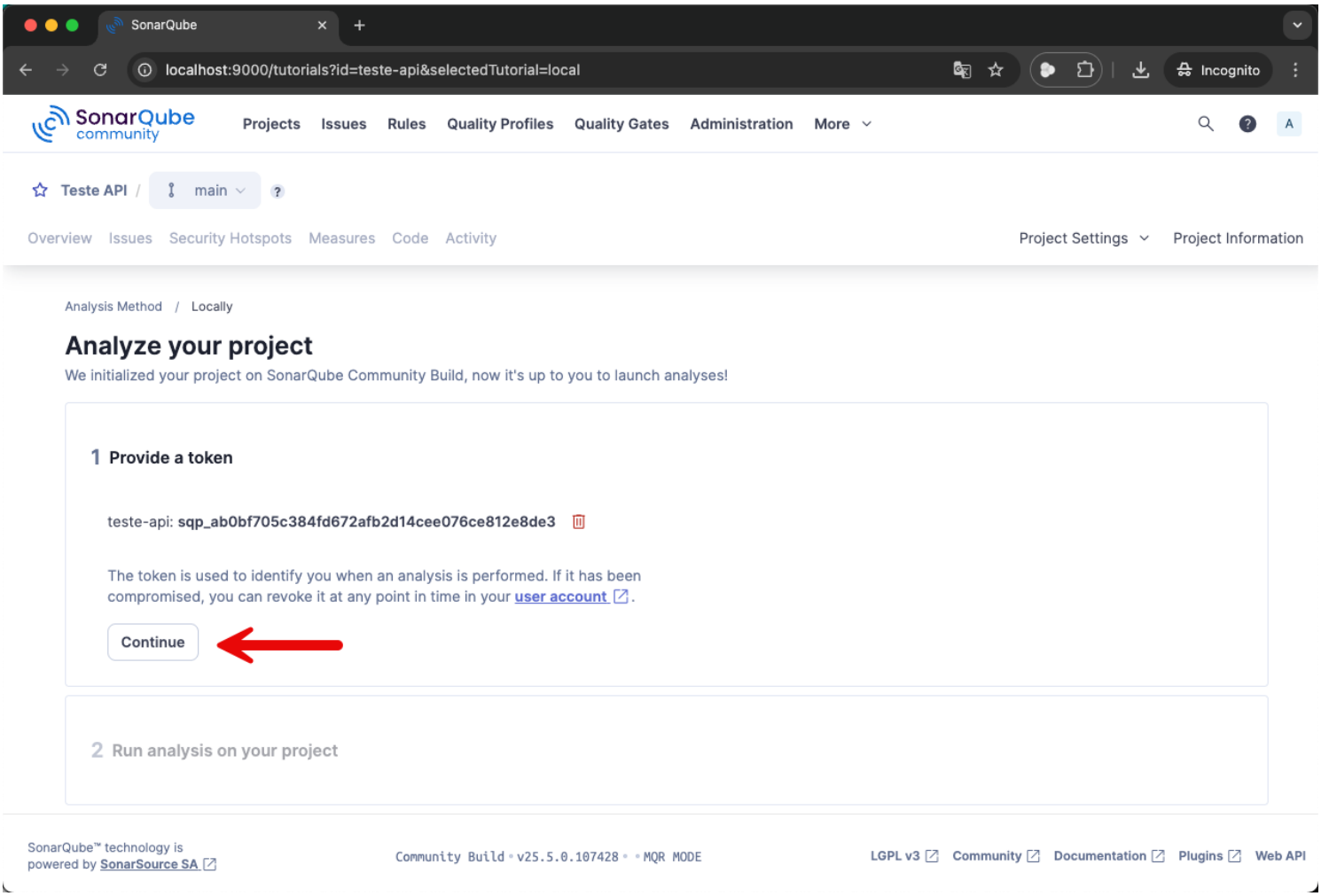
Por fim, selecione “*No expiration*” e clique em “*Generate*”.



Sonar

Criando um Projeto no Sonar

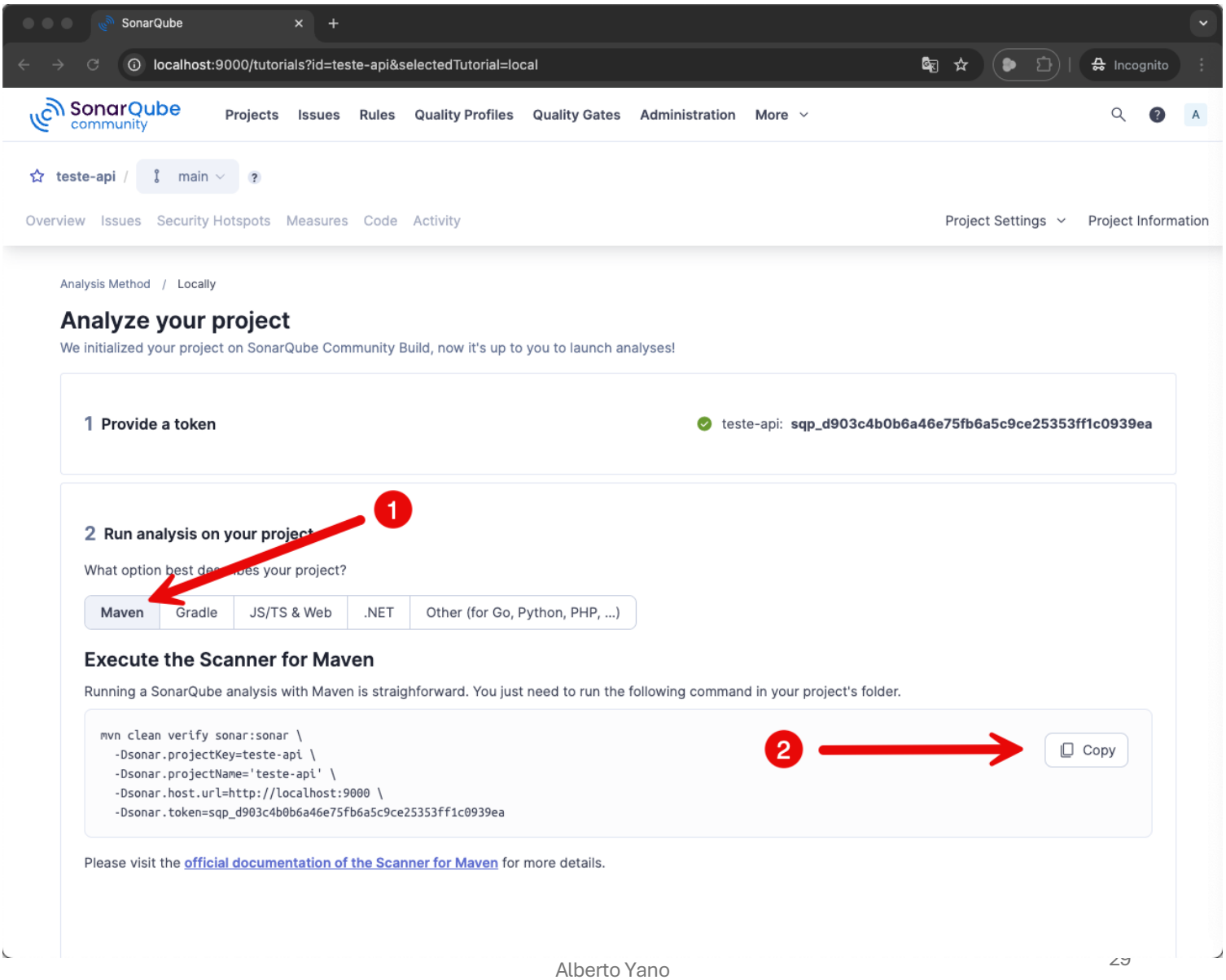
Clique em “*Continue*”.



Sonar

Criando um Projeto no Sonar

Clique em “*Maven*” e copie o conteúdo.



Sonar

Criando um Projeto no Sonar

Cole o conteúdo em um editor de texto e retire o “*mvn*” os “\“ e deixe tudo em uma única linha.

Ficará da seguinte forma:

```
clean verify sonar:sonar -Dsonar.projectKey=teste-api -
Dsonar.projectName='teste-api' -
Dsonar.host.url=http://localhost:9000 -
Dsonar.token=sqp_d903c4b0b6a46e75fb6a5c9ce2535
3ff1c0939ea
```

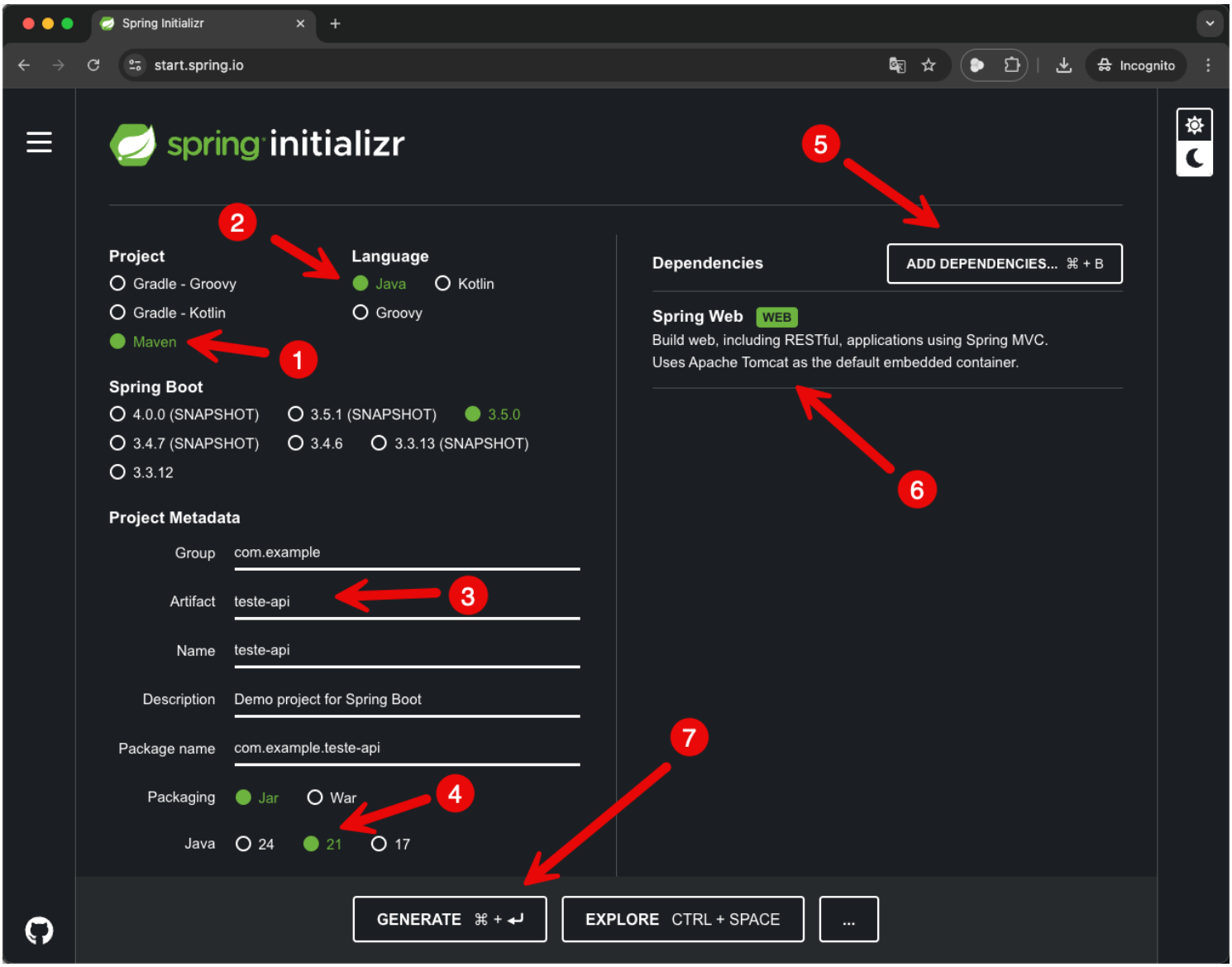
06

Spring Boot
Aplicação REST API

Spring Boot

Criando uma Aplicação REST API

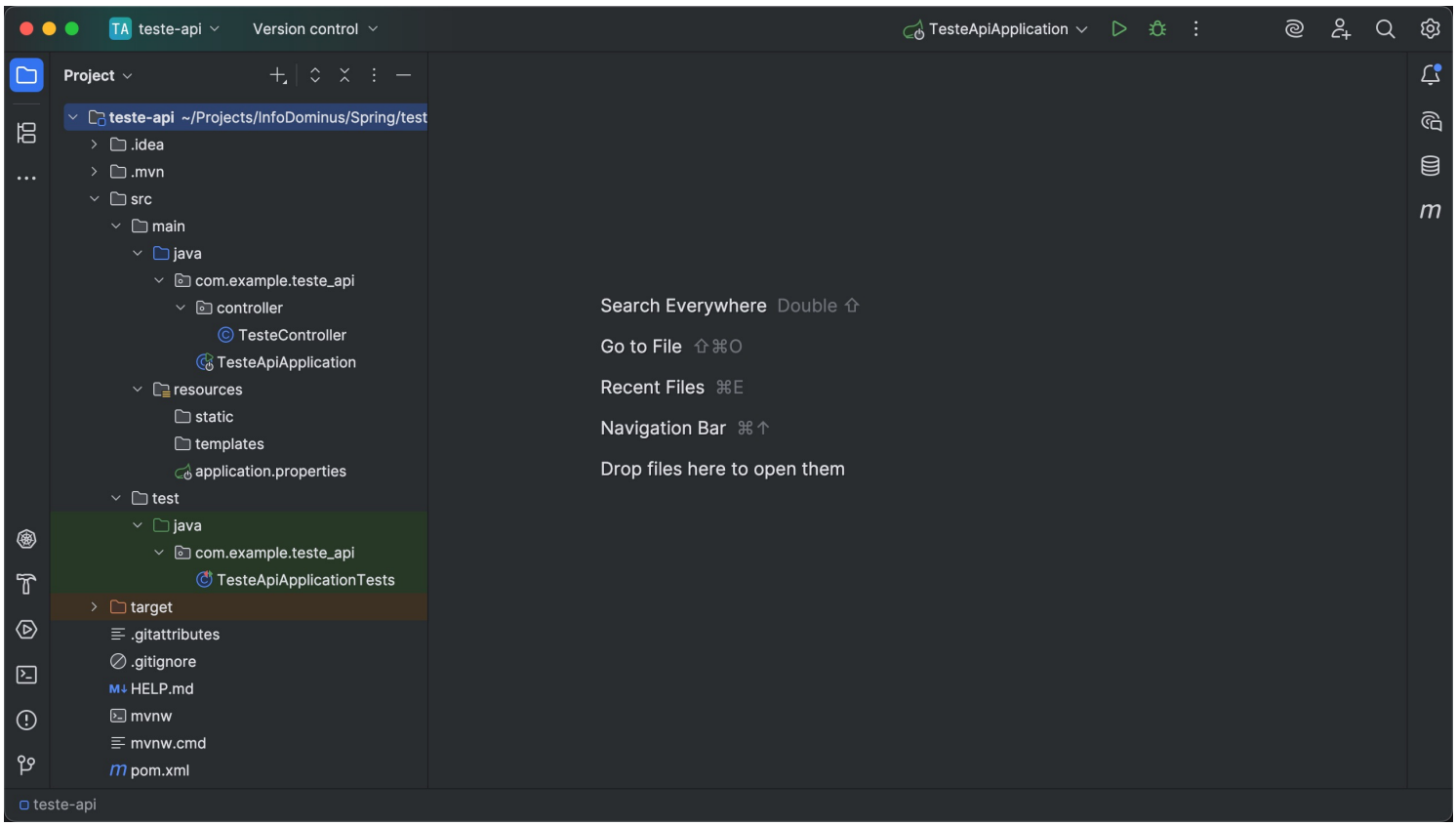
Acesse o seguinte link: https://start.spring.io/
e selecione preenchendo da forma abaixo:



Spring Boot

Criando uma Aplicação REST API

Descompacte o arquivo baixado e abra em sua IDE preferida.

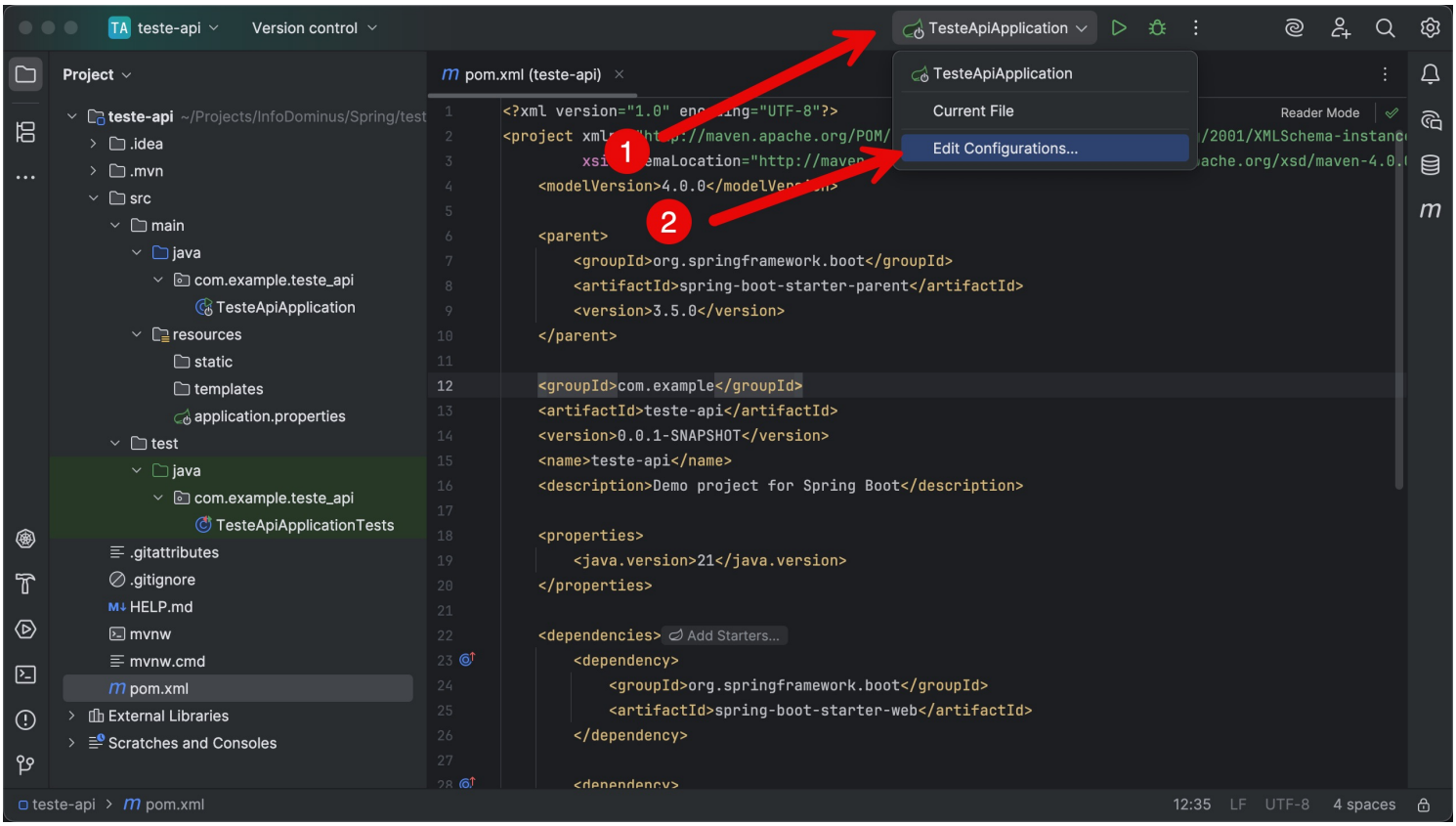


Spring Boot

Criando uma Aplicação REST API

Crie em sua IDE uma execução Maven com o comando da página 30 acrescentando ao final o seguinte código: -Dsonar.scm.disabled=true

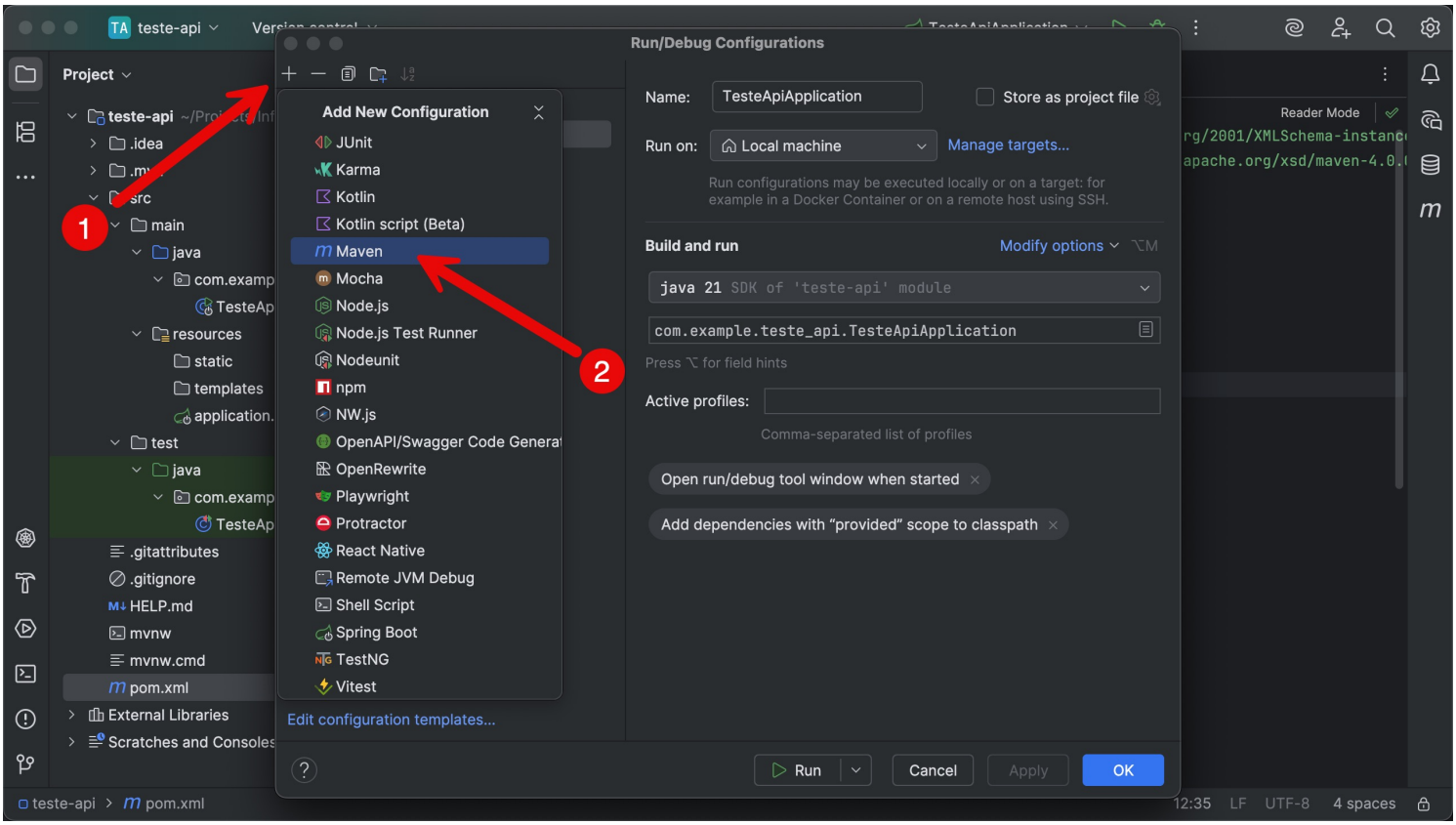
Usaremos aqui o IntelliJ:



Spring Boot

Criando uma Aplicação REST API

No IntelliJ, clique no “+” e selecione *Maven* para criar uma execução *Maven* para rodar o projeto com teste para o Sonar.

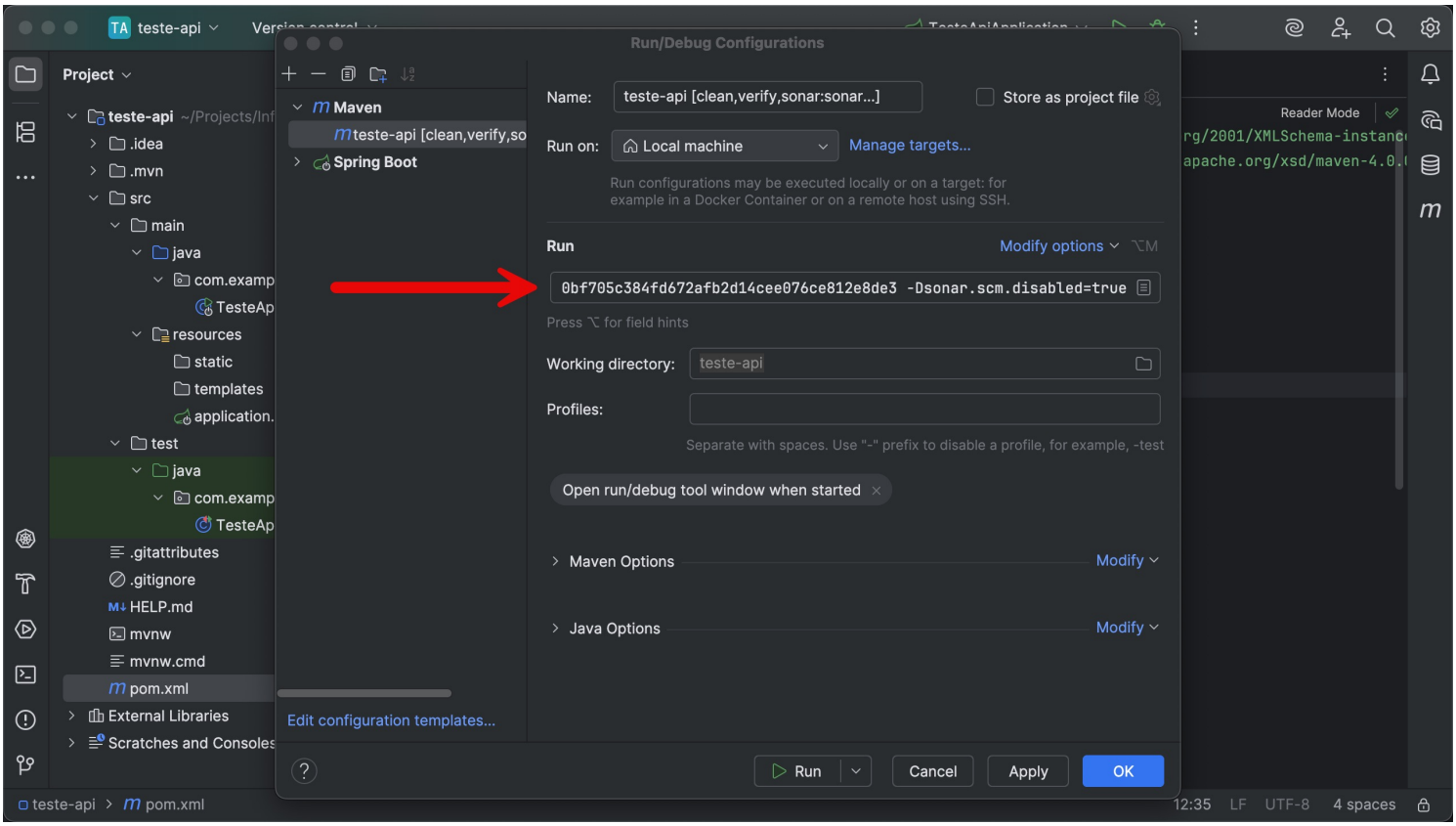


Spring Boot

Criando uma Aplicação REST API

Cole o commando completo no espaço “Run” e clique em “OK”:

clean verify sonar:sonar -Dsonar.projectKey=teste-api -Dsonar.projectName='teste-api' -Dsonar.host.url=http://localhost:9000 -Dsonar.token=sqp_d903c4b0b6a46e75fb6a5c9ce25353ff1c0939ea -Dsonar.scm.disabled=true



Spring Boot

Criando uma Aplicação REST API

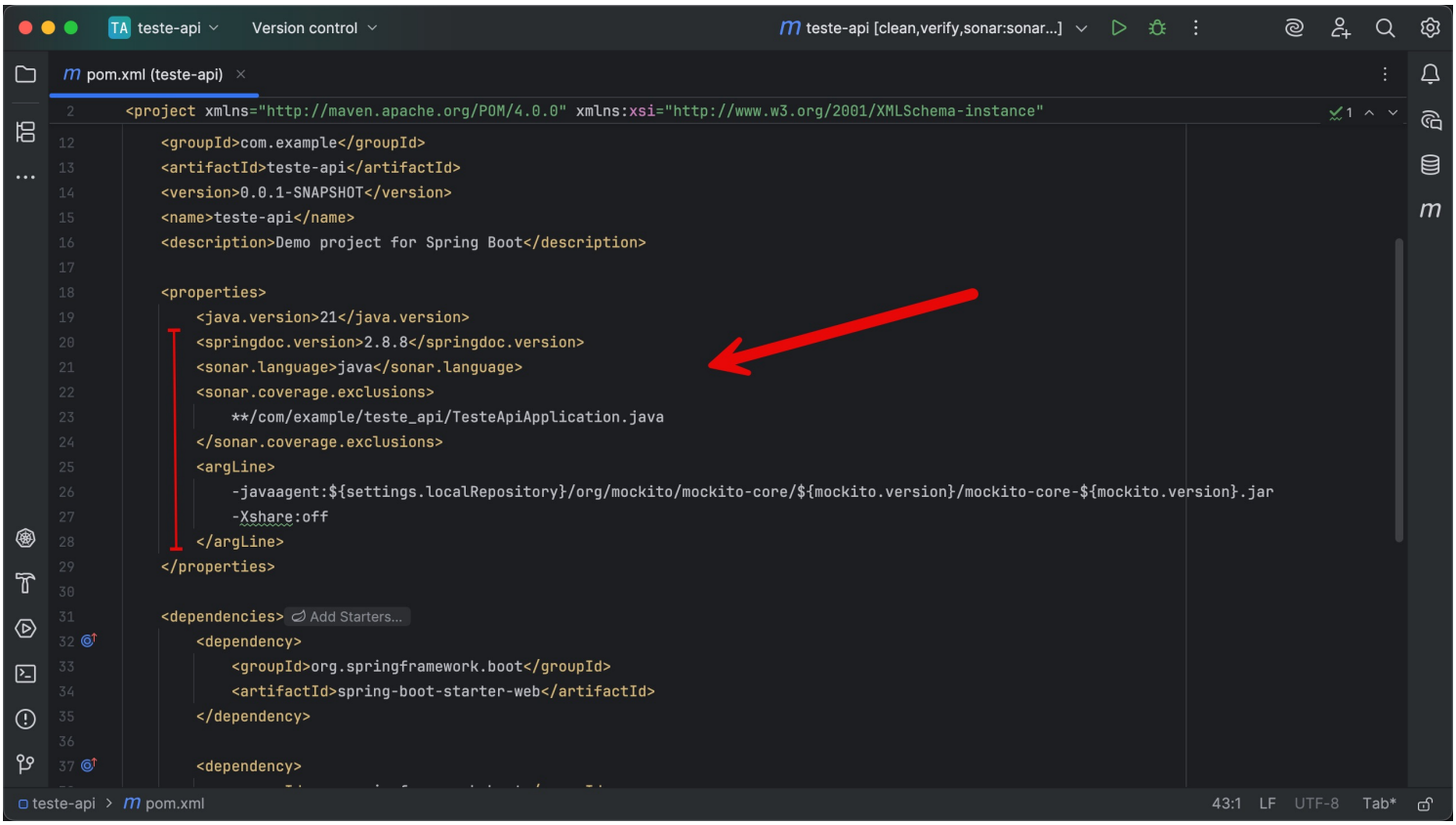
No POM.XML vamos acrescentar algumas tags dentro de <properties> dessa forma coloque as seguintes tags:

```
<springdoc.version>2.8.8</springdoc.version>
<sonar.language>java</sonar.language>
<sonar.coverage.exclusions>
**/com/example/teste_api/TesteApiApplication.java
</sonar.coverage.exclusions>
<argLine>
-
javaagent:${settings.localRepository}/org/mockito/mockit
o-core/${mockito.version}/mockito-core-
${mockito.version}.jar
-Xshare:off
</argLine>
```

Spring Boot

Criando uma Aplicação REST API

O <properties> do POM.XML ficará da seguinte forma:



Spring Boot

Criando uma Aplicação REST API

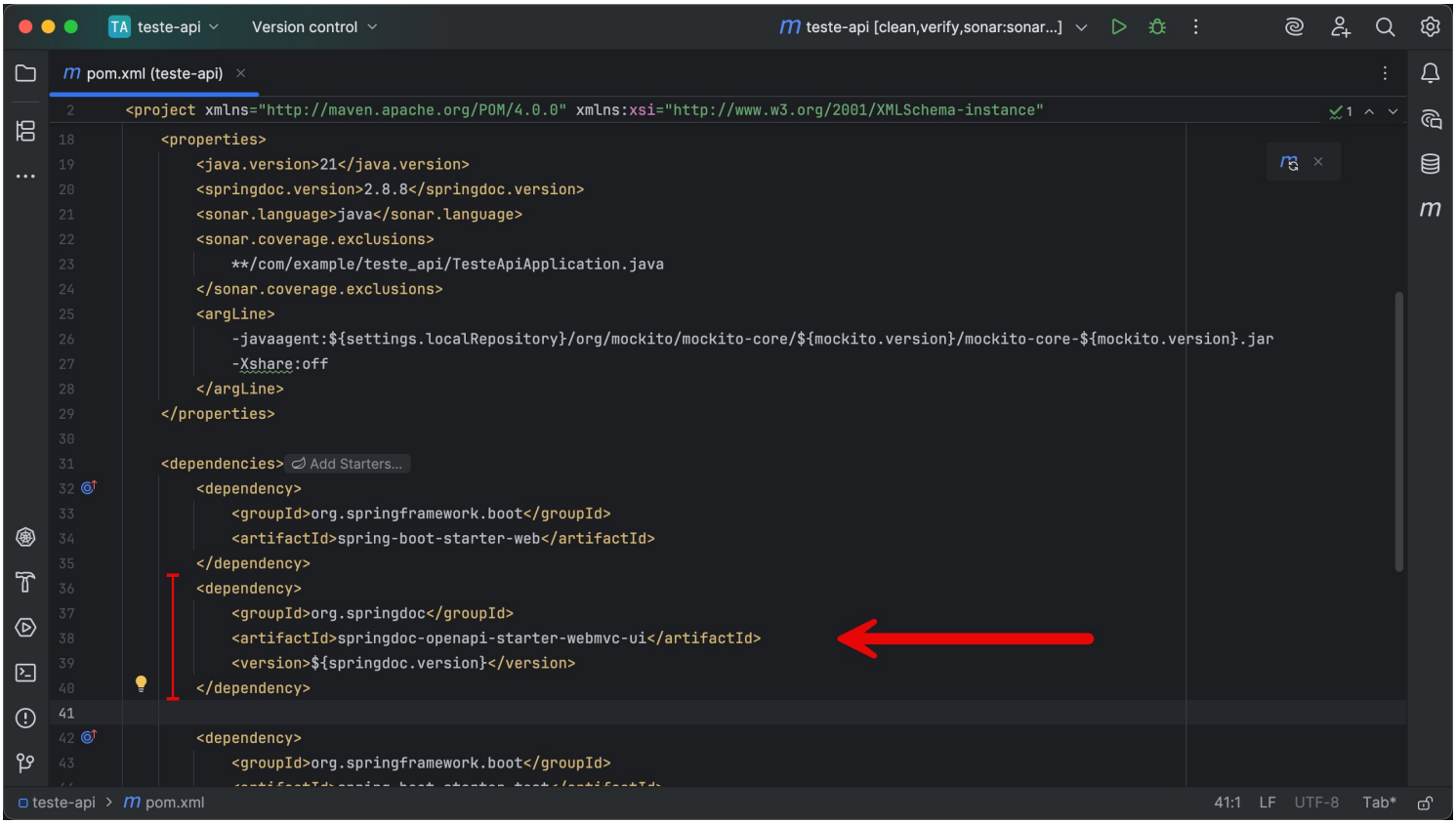
No POM.XML vamos acrescentar uma dependência para a documentação e acesso à API:

```
<dependency>
   <groupId>org.springdoc</groupId>
   <artifactId>springdoc-openapi-starter-webmvc-
ui</artifactId>
   <version>${springdoc.version}</version>
</dependency>
```

Spring Boot

Criando uma Aplicação REST API

O <dependencies> do POM.XML ficará da seguinte forma:



Spring Boot

Criando uma Aplicação REST API

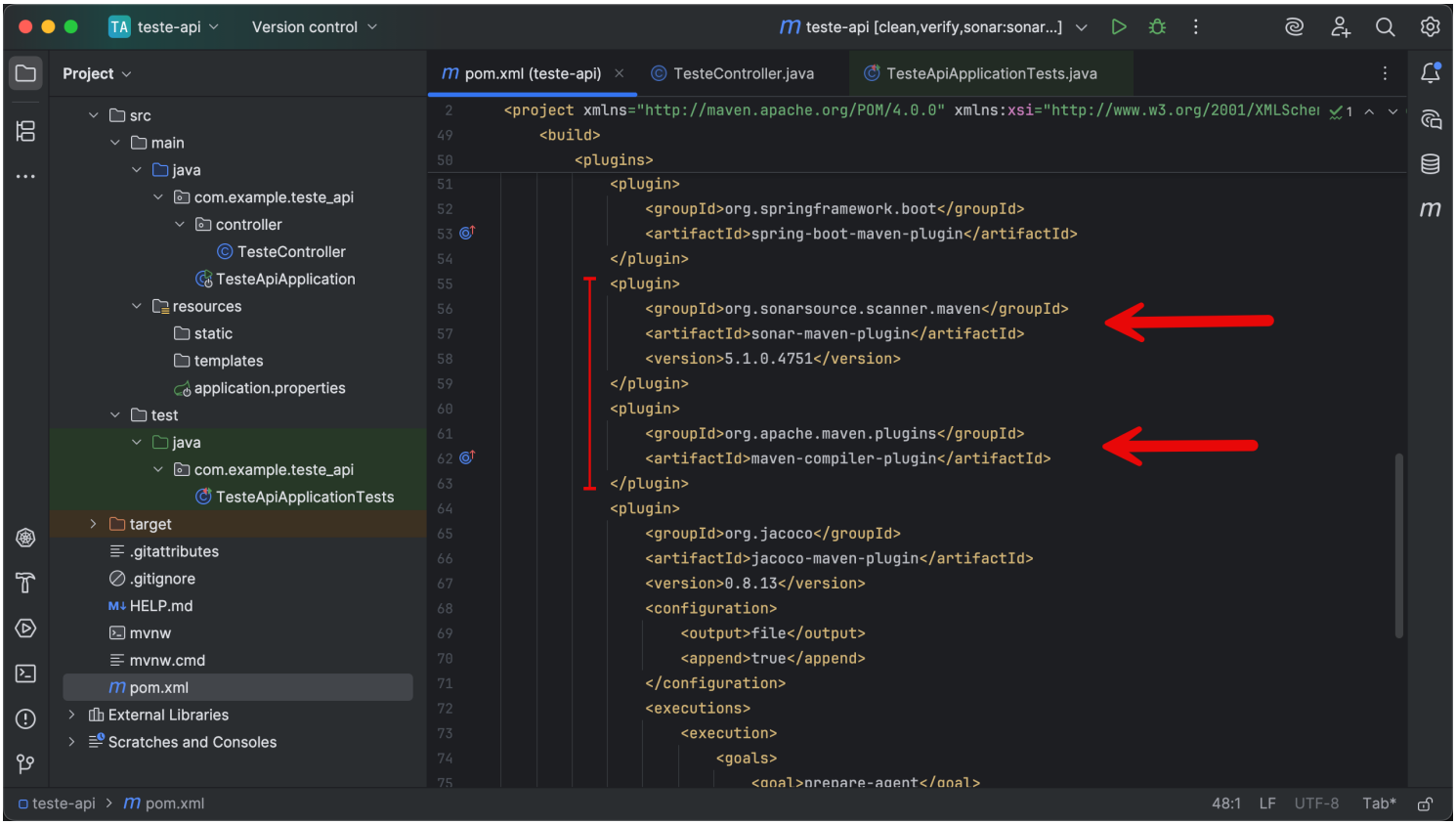
No POM.XML vamos acrescentar ainda alguns plugins:

```
<plugin>
	<groupId>org.sonarsource.scanner.maven</groupId>
	<artifactId>sonar-maven-plugin</artifactId>
	<version>5.1.0.4751</version>
</plugin>
<plugin>
	<groupId>org.apache.maven.plugins</groupId>
	<artifactId>maven-compiler-plugin</artifactId>
</plugin>
<plugin>
```

Spring Boot

Criando uma Aplicação REST API

O <plugins> do POM.XML ficará da seguinte forma:



Spring Boot

Criando uma Aplicação REST API

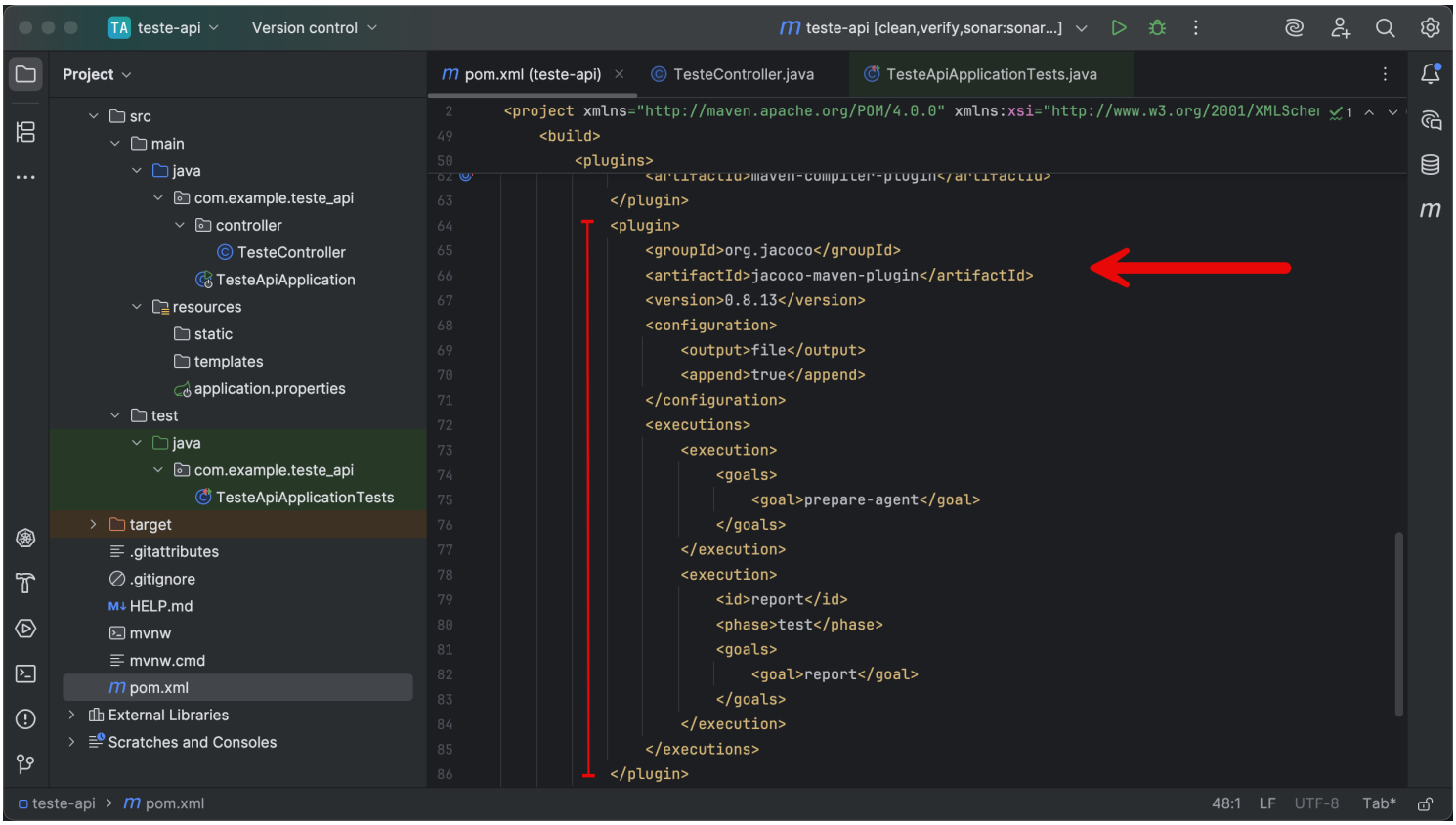
No POM.XML vamos acrescentar um *plugin* para cobertura de código:

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.13</version>
  <configuration>
    <output>file</output>
    <append>true</append>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <execution>
      <id>report</id>
      <phase>test</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Spring Boot

Criando uma Aplicação REST API

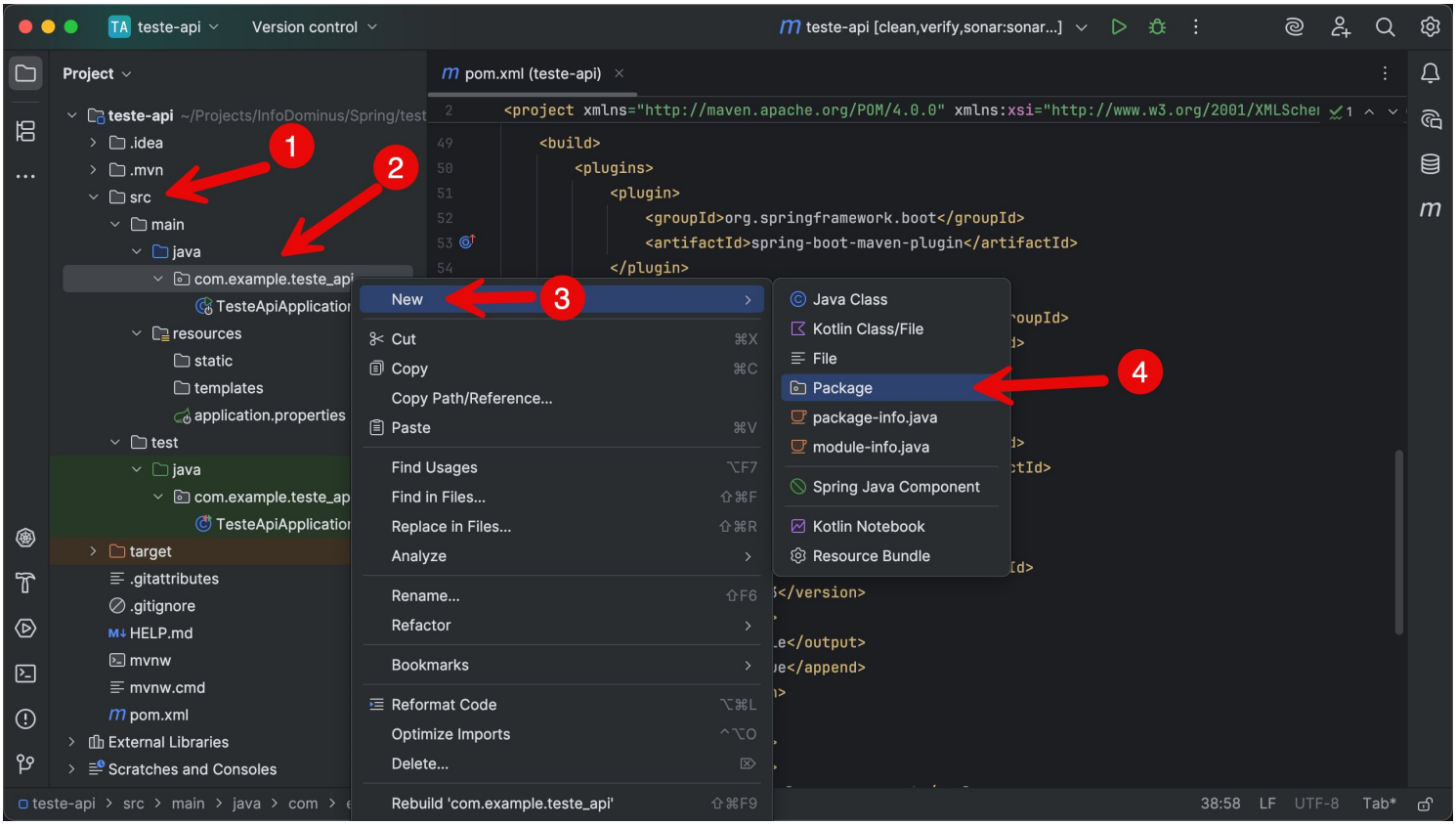
O <plugins> do POM.XML ficará da seguinte forma:



Spring Boot

Criando uma Aplicação REST API

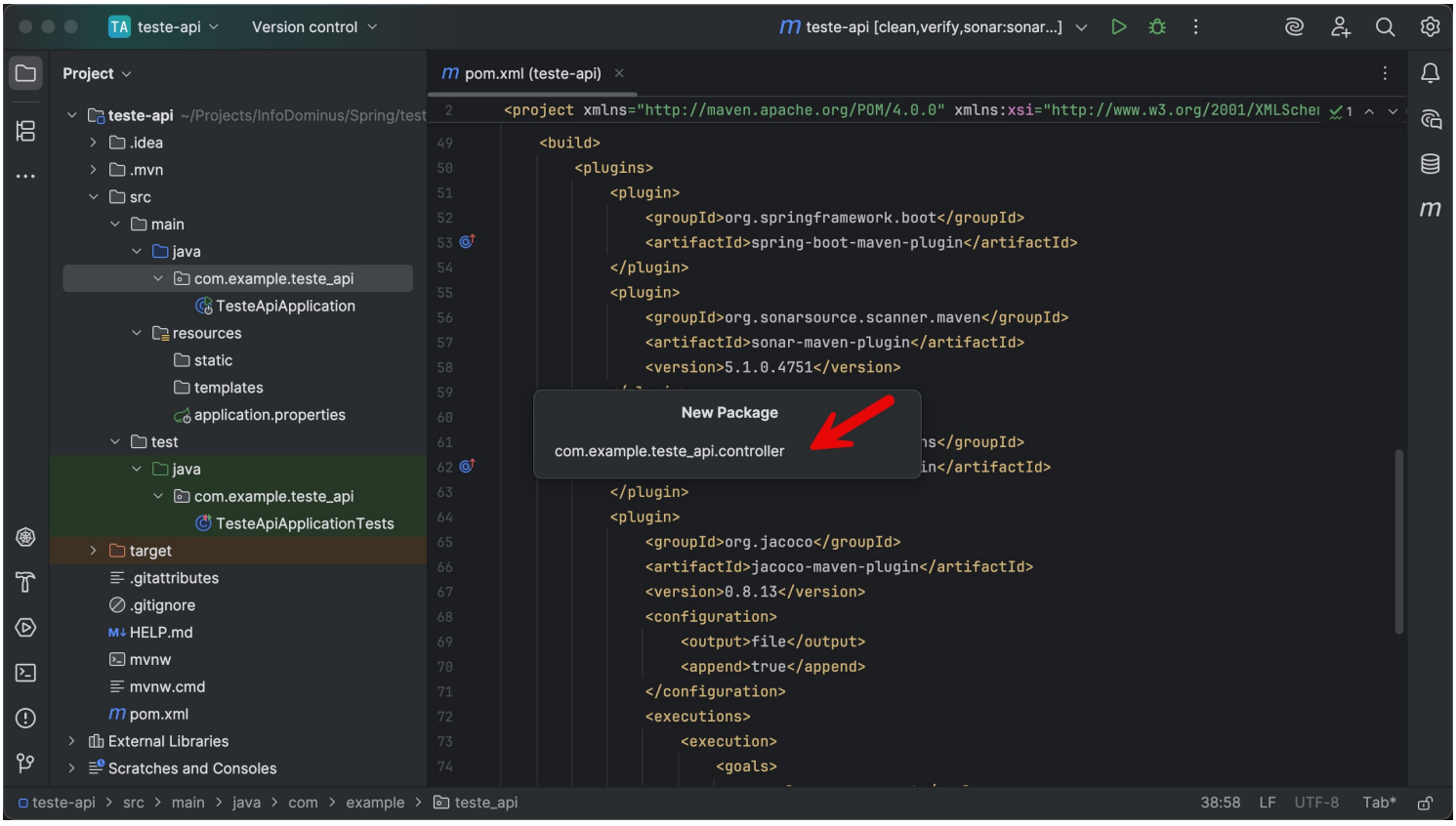
Vamos criar uma package “*controller*” onde ficará a nossa classe que receberá as requisições:



Spring Boot

Criando uma Aplicação REST API

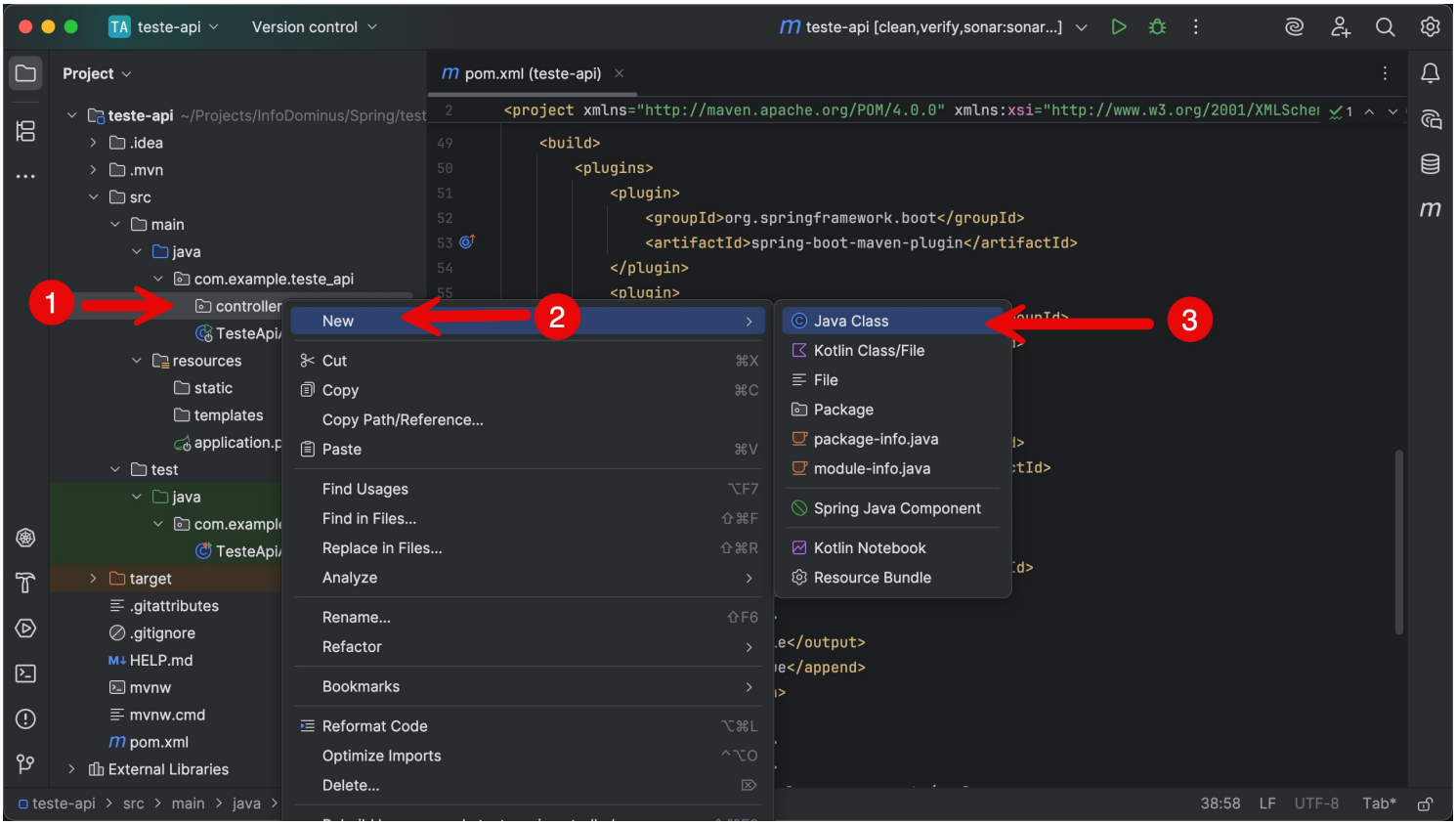
Colocar no nome da package “*controller*”.



Spring Boot

Criando uma Aplicação REST API

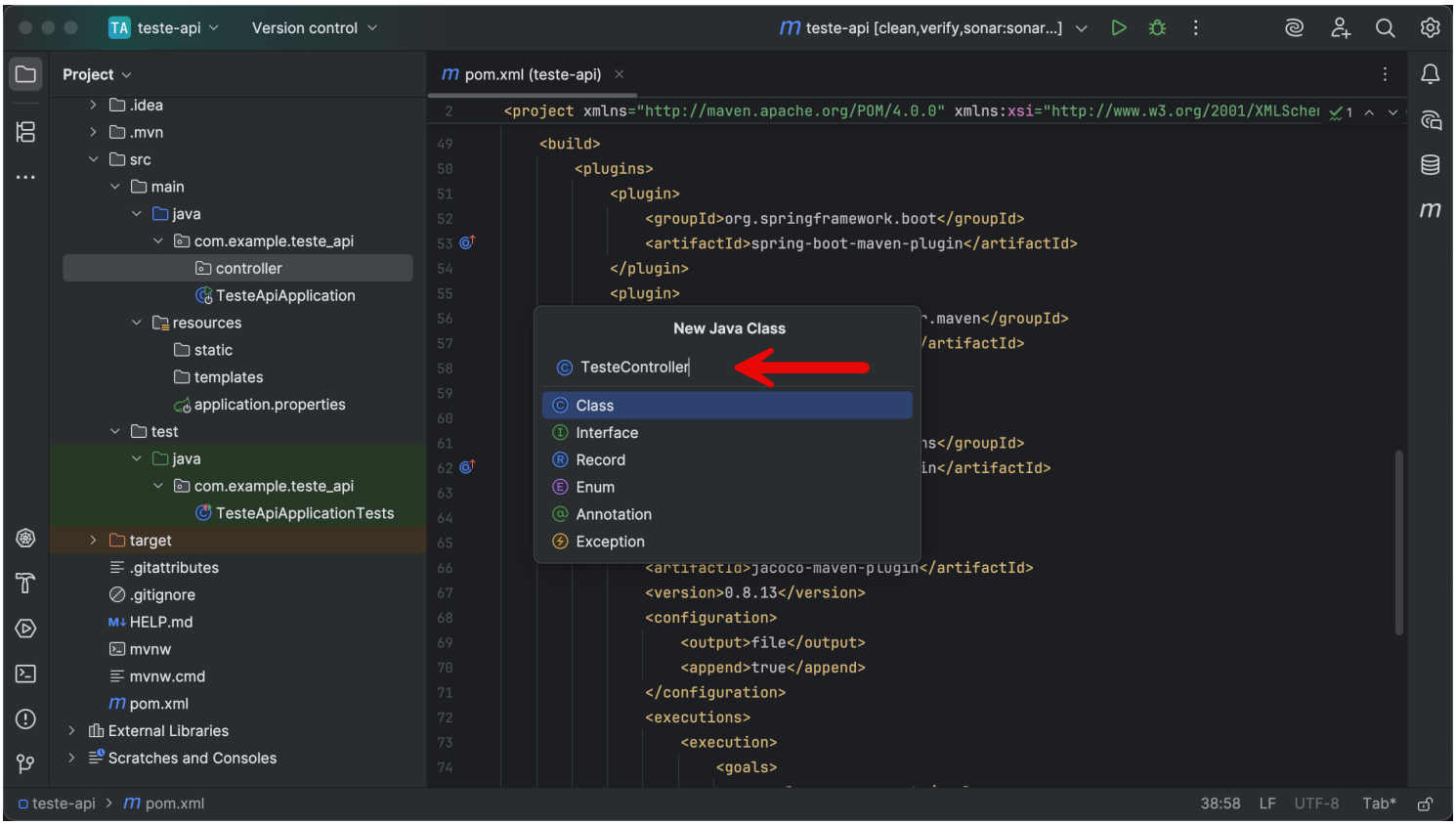
Vamos criar a nossa classe *Controller* dentro da package "*controller*".



Spring Boot

Criando uma Aplicação REST API

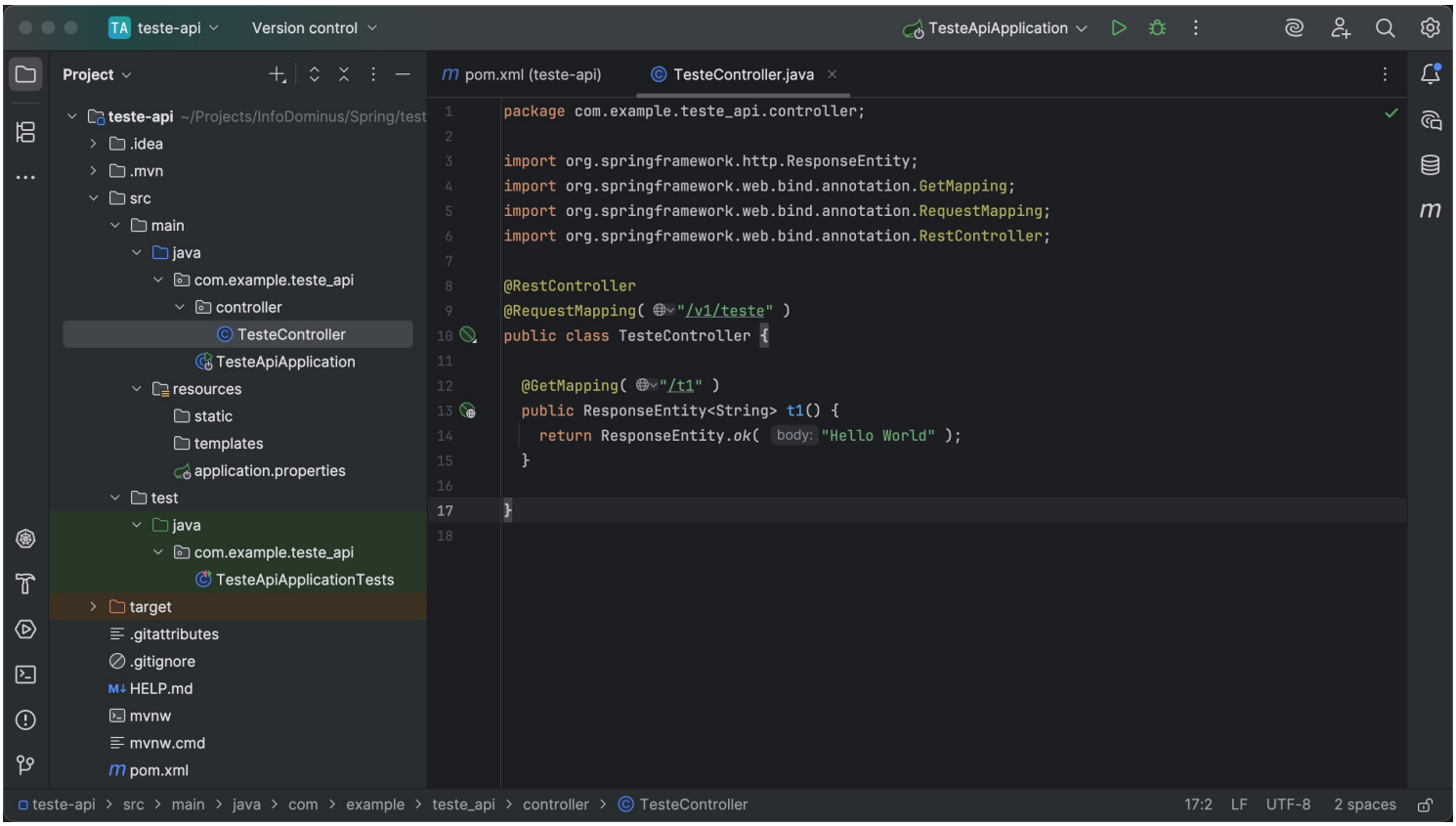
Colocar o nome da Classe como *TesteController*.



Spring Boot

Criando uma Aplicação REST API

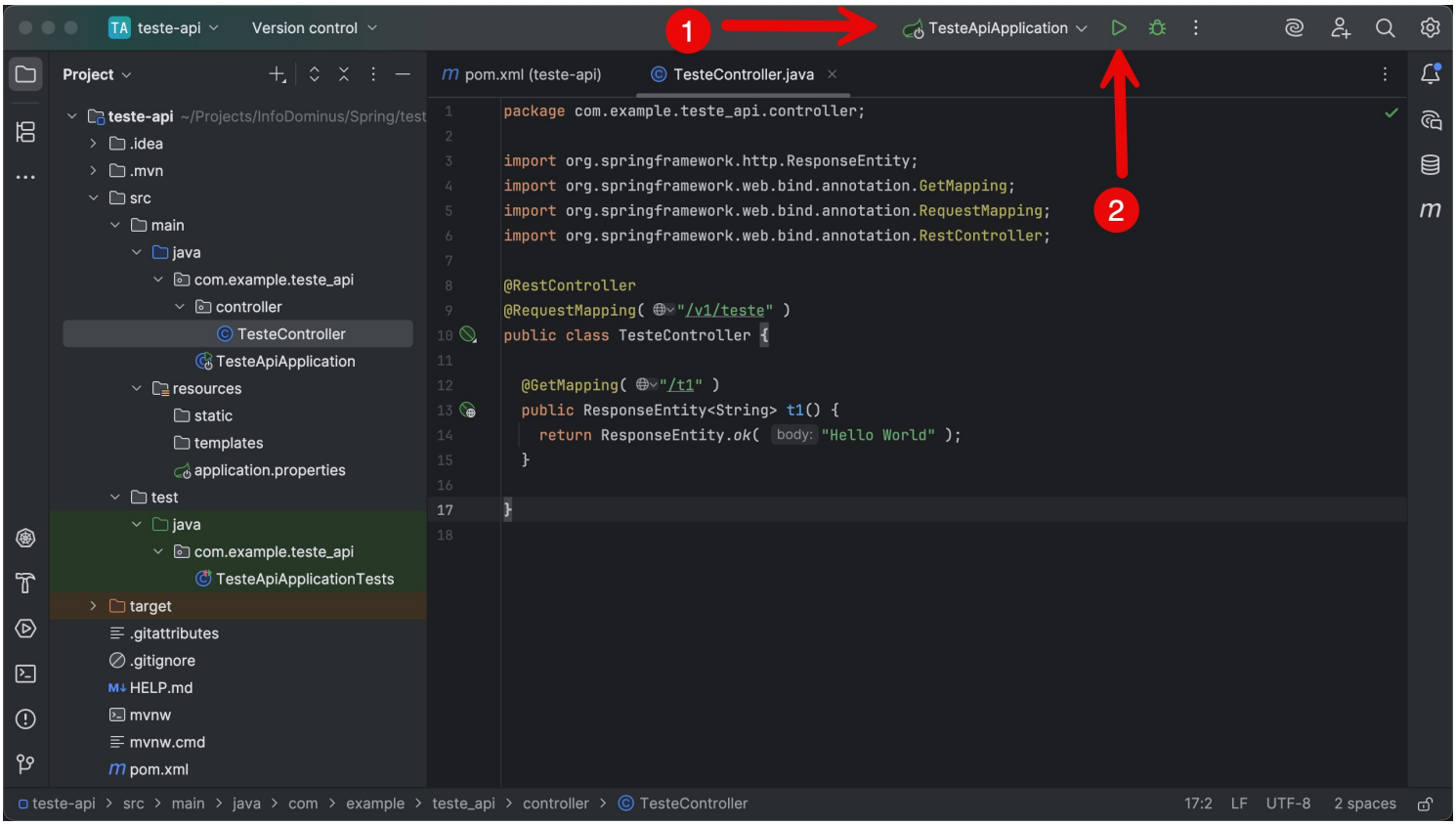
Vamos anotar a classe com as tags: *@RestController* e *@RequestMapping* e criar um endpoint t1 que retorna o Texto "*Hello World*".



Spring Boot

Criando uma Aplicação REST API

Com isso já podemos rodar o projeto e acessar o *Swagger*.

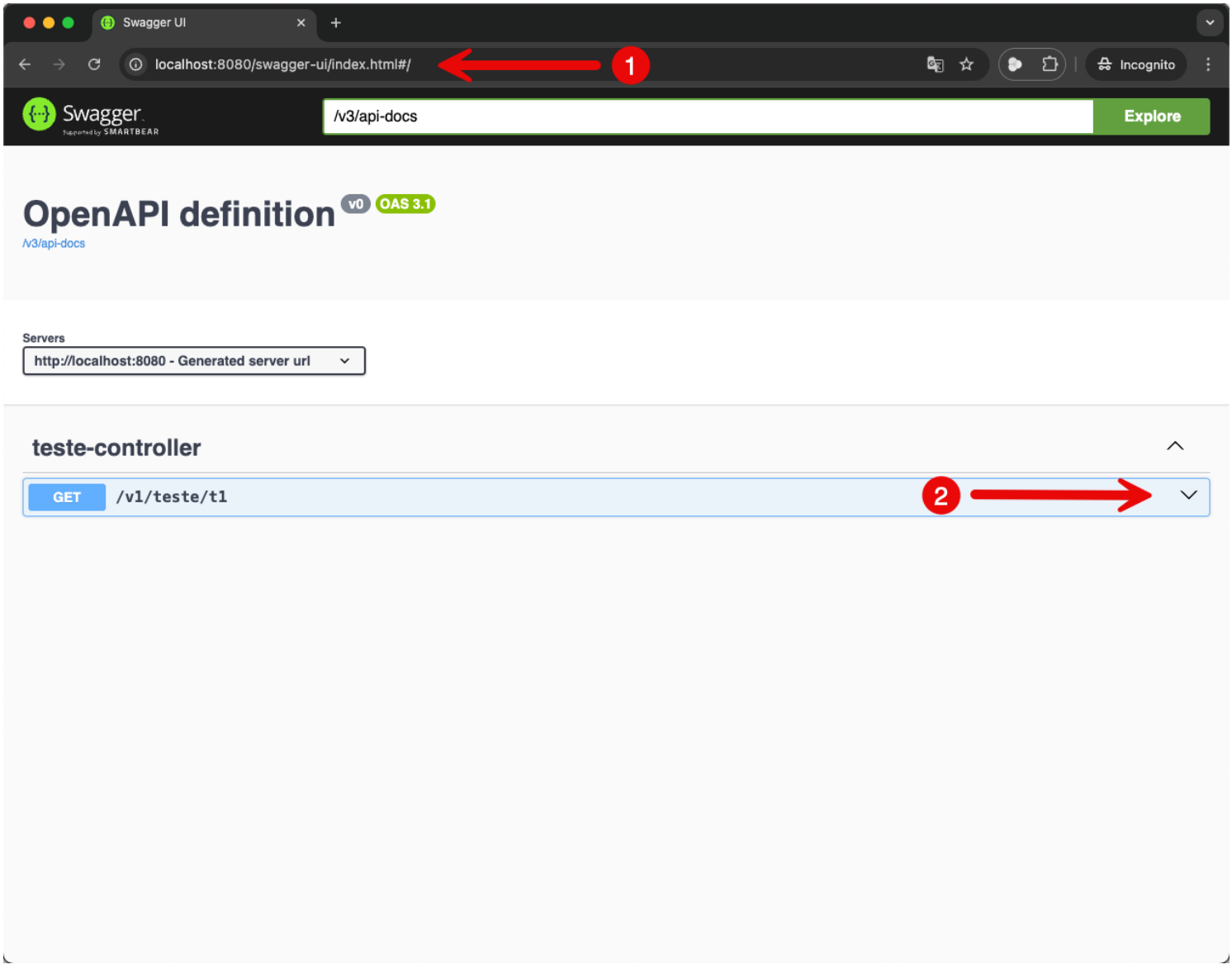


Spring Boot

Criando uma Aplicação REST API

No browser acesse a URL:

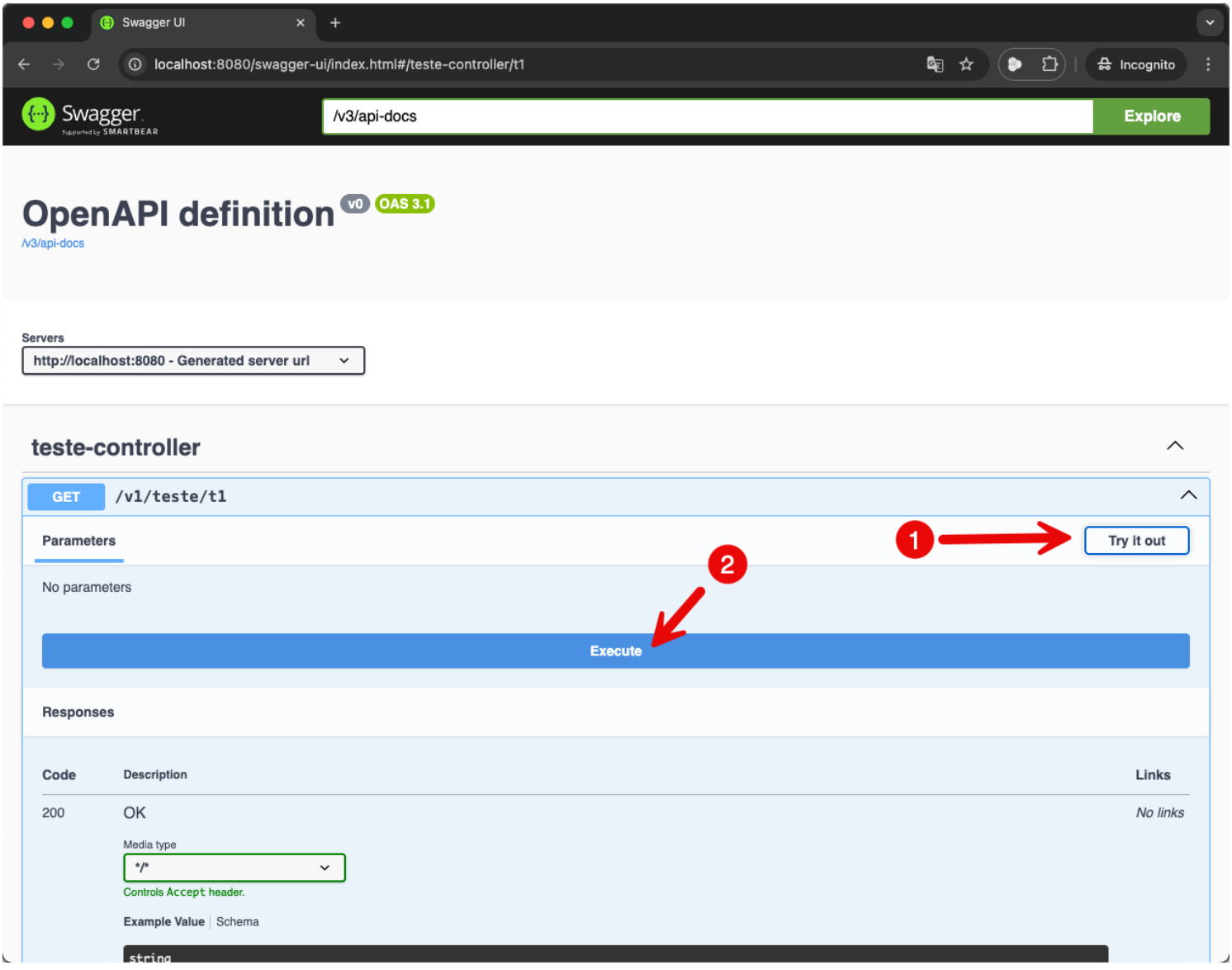
http://localhost:8080/swagger-ui/index.html



Spring Boot

Criando uma Aplicação REST API

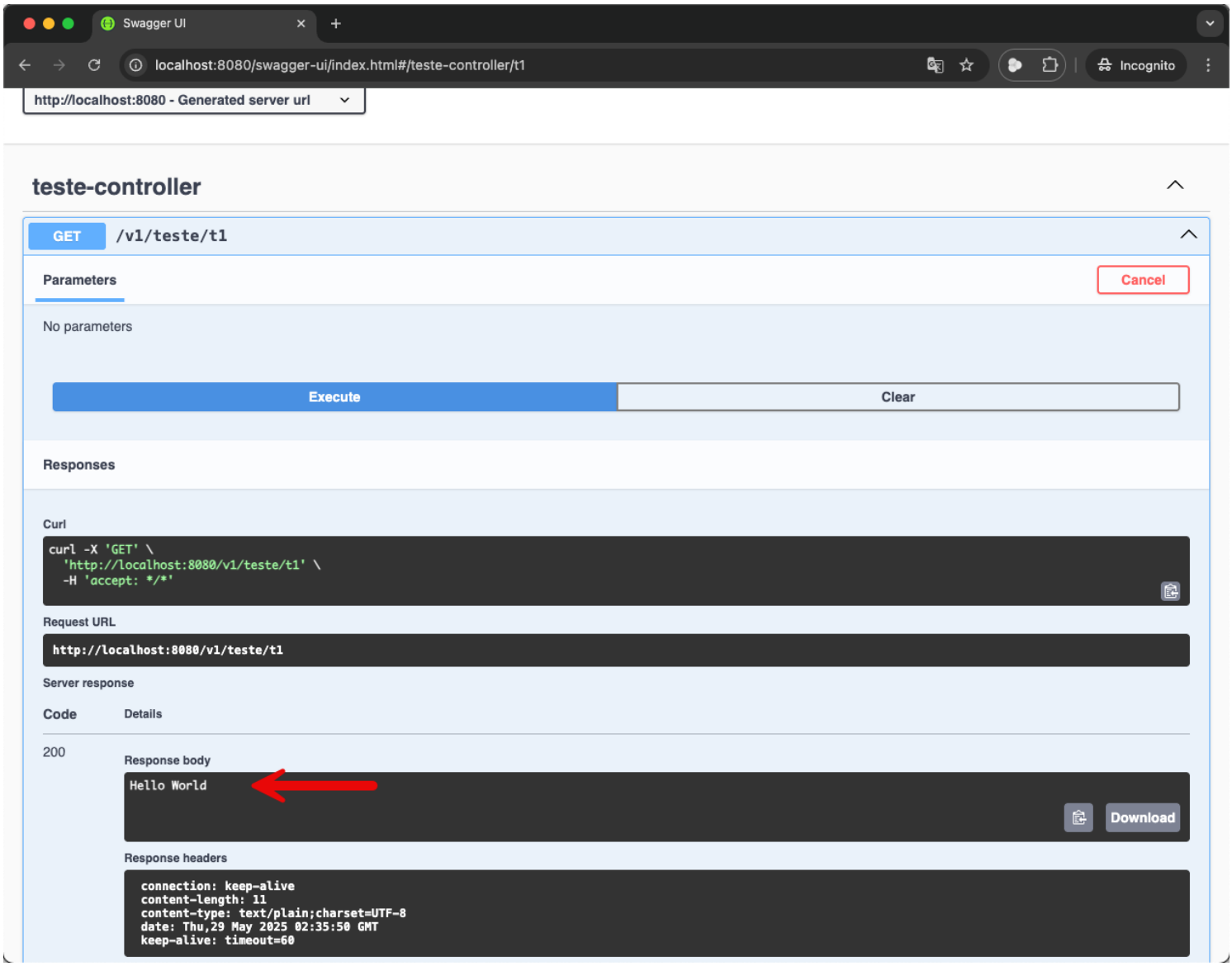
Clique em "*Try it out*" e depois em "*Execute*" para realizar a chamada.



Spring Boot

Criando uma Aplicação REST API

O Resultado aparecerá embaixo, e então nossa API REST estará funcionando.



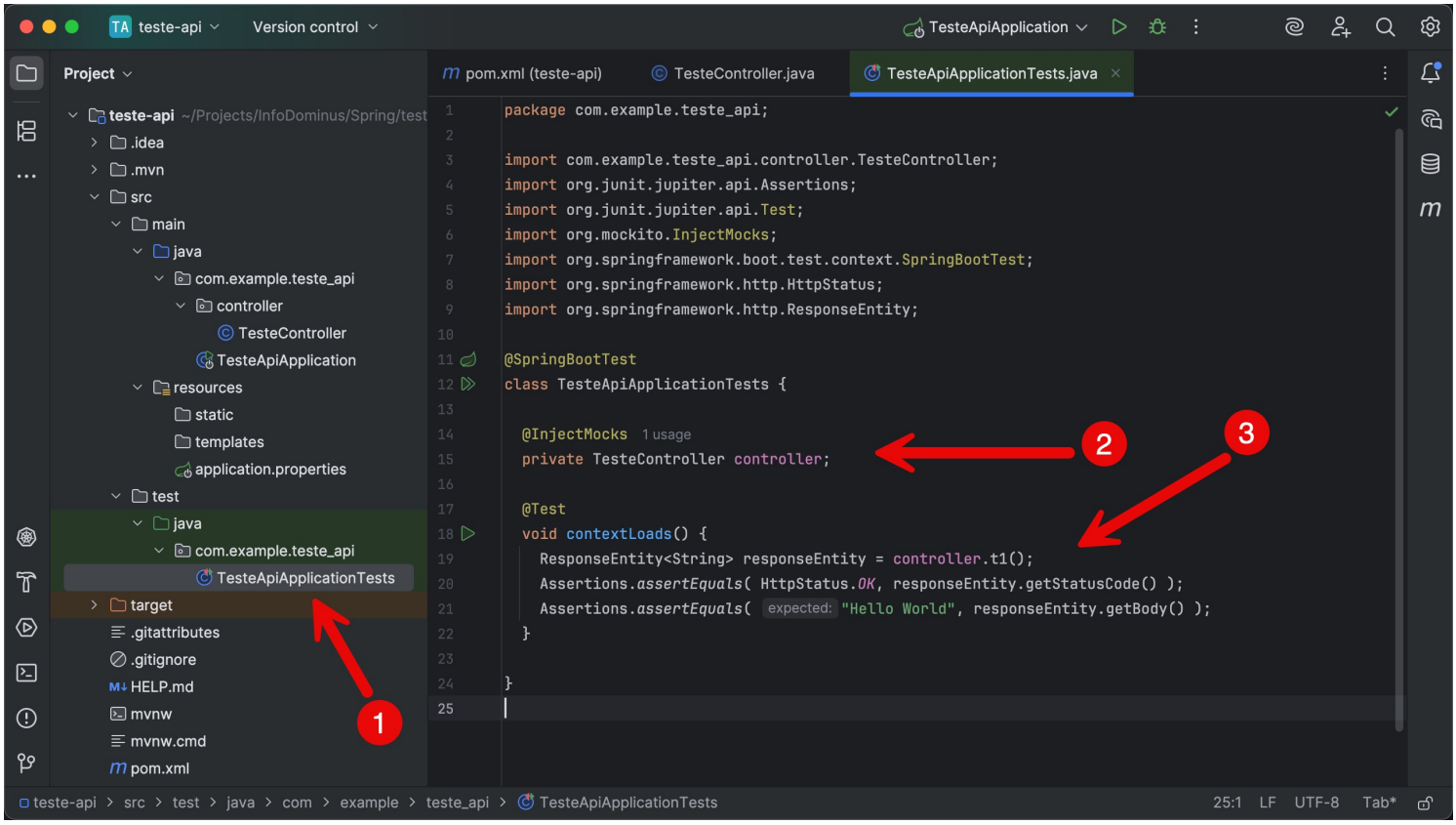
07

Spring Boot
Teste Unitário

Spring Boot

Criando o Teste Unitário

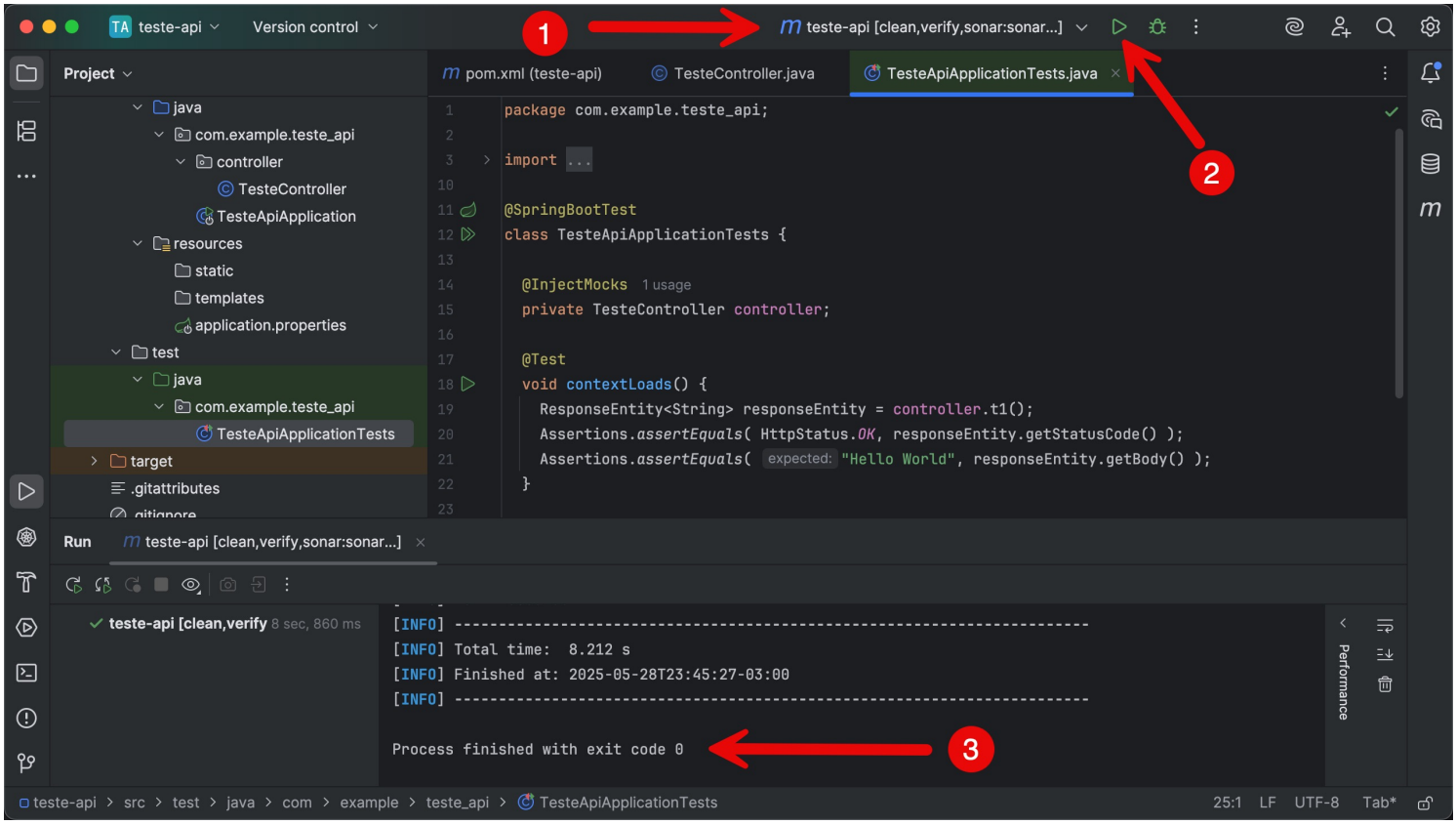
Para simplificar vamos utilizar a classe de teste criada automaticamente pelo *Spring Initializr* devendo só complementar com o código abaixo (OBS: Não esqueça dos imports).



Spring Boot

Criando o Teste Unitário

Agora vamos selecionar o runner do *Maven* que criamos anteriormente e executar. A execução deve ocorrer sem erros.



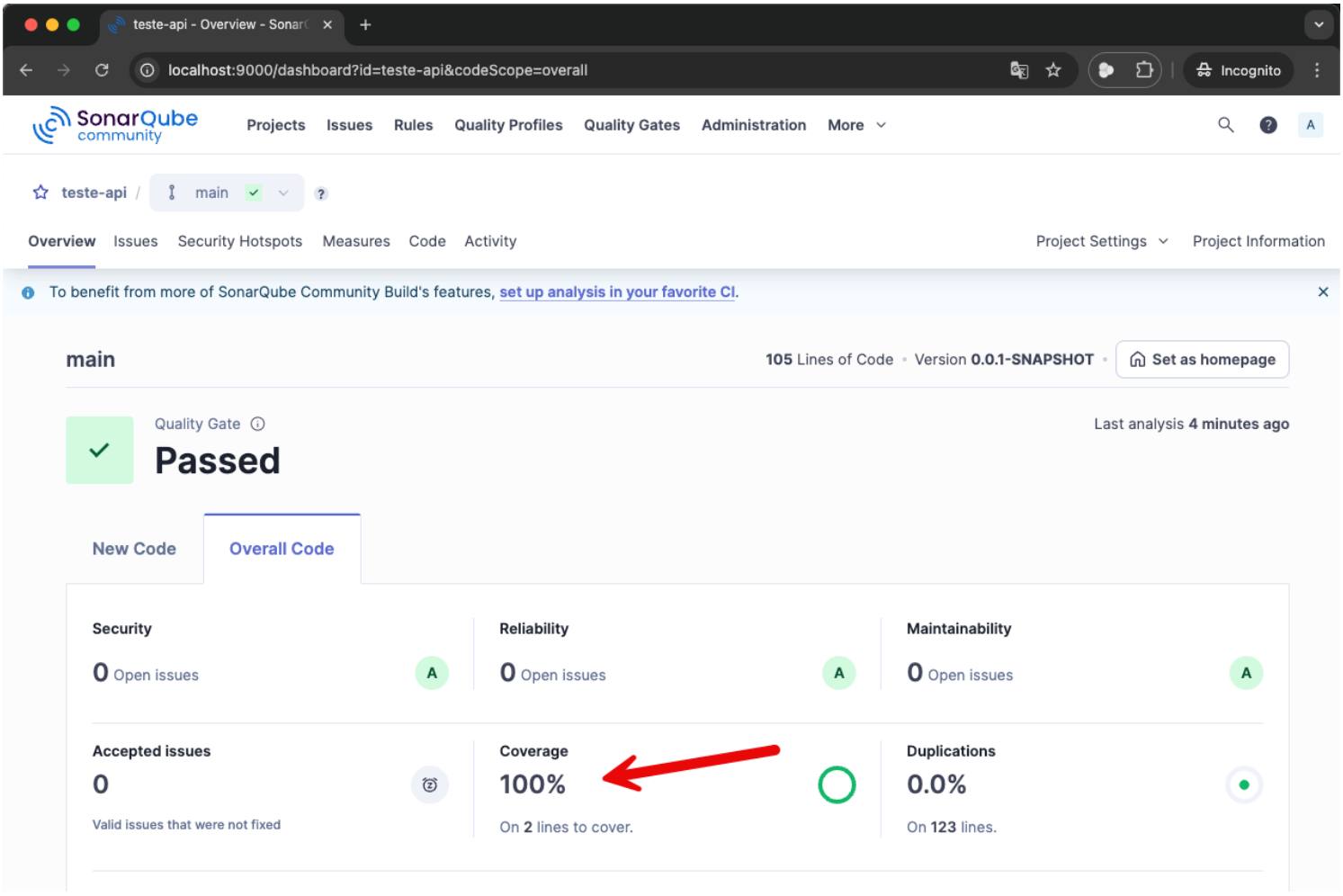
Spring Boot

Criando o Teste Unitário

Acessando o Sonar na URL:

http://localhost:9000/

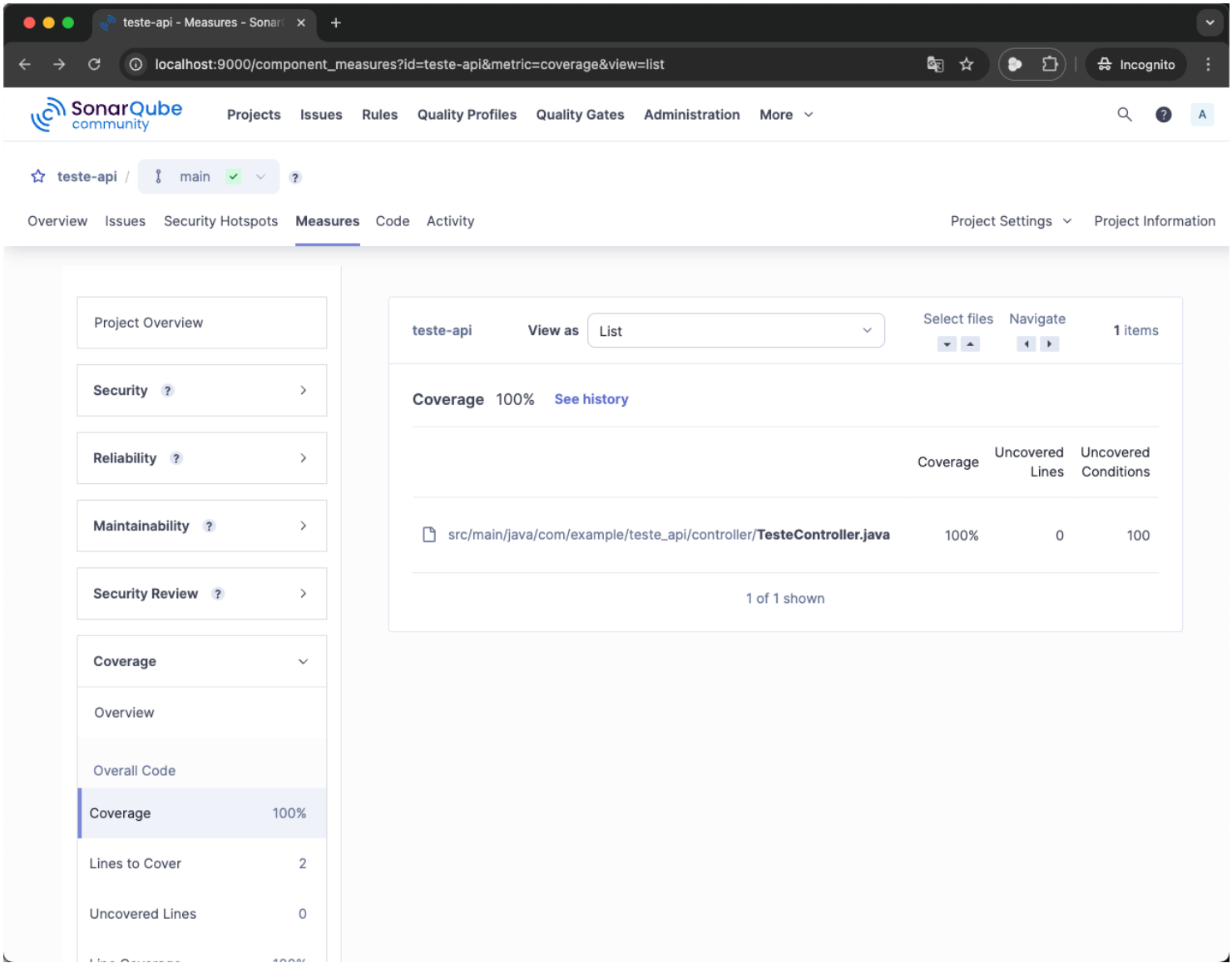
Veremos que tivemos 100% de cobertura.



Spring Boot

Criando o Teste Unitário

Ao clicar no “100%” mostrará as classes e suas coberturas.



09

Esclarecimentos Finais

Esclarecimentos Finais

Espero que este e-Book lhe ajude a melhorar e aperfeiçoar o seu código de forma mais rápida e eficiente.

Contudo alertamos que pode ocorrer de tentar colocar as dicas desse e-Book em um projeto já existente e não funcionar a cobertura de Código.

E isso ocorre, por exemplo, devido aos conflitos de bibliotecas e plugins. A dica é tentar retirar tudo do seu antigo projeto e colocar somente o que foi mostrado aqui, e ir incrementando aos poucos para tentar descobrir qual biblioteca ou plugin está conflitando e que não esteja deixando realizar os testes e a cobertura de código.

Esclarecimentos Finais

O projeto mínimo mostrado aqui no e-book ficará disponível no github no seguinte link:

https://github.com/alberto-yano/dio-lab-ebook

E o vídeo de apresentação no link:
https://youtu.be/86Zb8l8n6Mc

Seguem os meus contatos para o caso de dúvidas e sugestões:

https://www.linkedin.com/in/albertoyano/

https://github.com/alberto-yano

https://gitlab.com/alberto_yano