



Spring Boot

Criação do Projeto e sincronização com o Github - aula 02

PROF. ME. JEFFERSON PASSERINI



Spring Boot

Criando Projeto Spring Boot

- Vamos utilizar o IntelliJ para desenvolvimento, na sua versão Community não temos o gerador de projetos Spring na própria IDE então utilizaremos o Spring Initializr.



Project
☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ **Maven**

Language
☒ **Java** ☐ Kotlin ☐ Groovy

Spring Boot
☐ 3.4.0 (SNAPSHOT) ☐ 3.4.0 (M2) ☐ 3.3.4 (SNAPSHOT) ☒ **3.3.3**
☐ 3.2.10 (SNAPSHOT) ☐ 3.2.9

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ **Jar** ☐ War

Java ☐ 22 ☐ 21 ☒ **17**

Dependencies ADD DEPENDENCIES... CTRL + B

No dependency selected

GENERATE CTRL + ⌘

EXPLORE CTRL + SPACE

SHARE...

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin☒ Maven

Language

☒ Java ☐ Kotlin ☐ Groovy

1

Spring Boot

☐ 3.4.0 (SNAPSHOT) ☐ 3.4.0 (M2) ☐ 3.3.4 (SNAPSHOT) ☒ 3.3.3☐ 3.2.10 (SNAPSHOT) ☐ 3.2.9

Project Metadata

Group com.curso

Artifact suporteos

Name suporteos

Description Projeto Curso de Java Spring Boot

Package name com.curso.suporteos

Packaging ☒ Jar ☐ WarJava ☐ 22 ☐ 21 ☒ 17

3

Dependencies

ADD DEPENDENCIES... CTRL + B

No dependency selected

- 1 - Especificar a linguagem: **Java**, o gerenciador de pacotes: **Maven**, e a versão do Spring Boot: **3.3.3**
- 2 - Defina o Project Metadata como na imagem.
- 3 - Defina o empacotamento do build do projeto: **Jar** e a versão do Java: **17**

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin

☒ Maven

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 3.4.0 (SNAPSHOT) ☐ 3.4.0 (M2) ☐ 3.3.4 (SNAPSHOT) ☒ 3.3.3

☐ 3.2.10 (SNAPSHOT) ☐ 3.2.9

Project

- 4 - Adicione as dependências de projeto, conforme a imagem.
- 5 - Gere seu projeto Spring através do botão Generate.

Packaging

☒ Jar ☐ War

Java ☐ 22 ☐ 21 ☒ 17

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot DevTools DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

H2 Database SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

PostgreSQL Driver SQL

A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Validation I/O

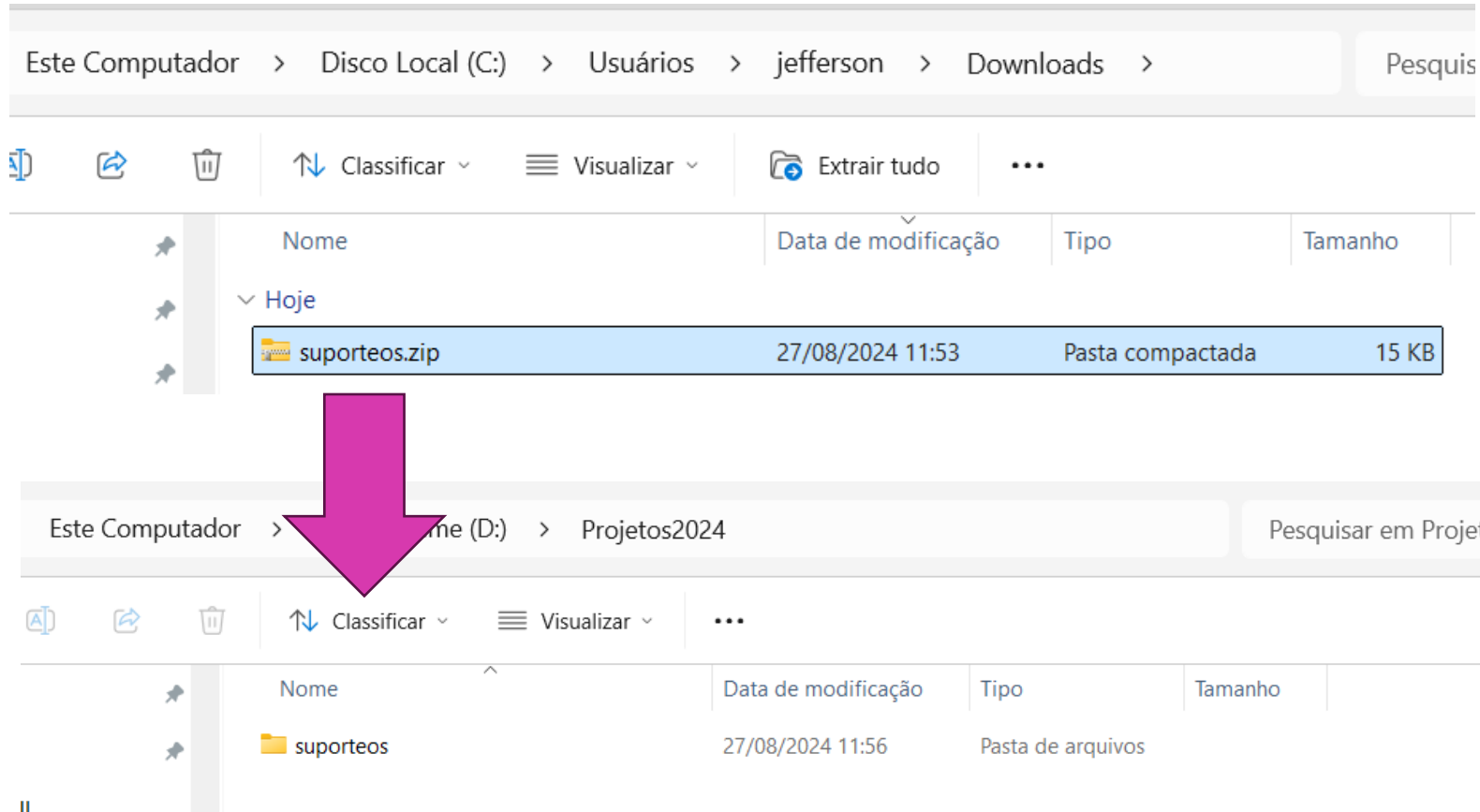
Bean Validation with Hibernate validator.

5 GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

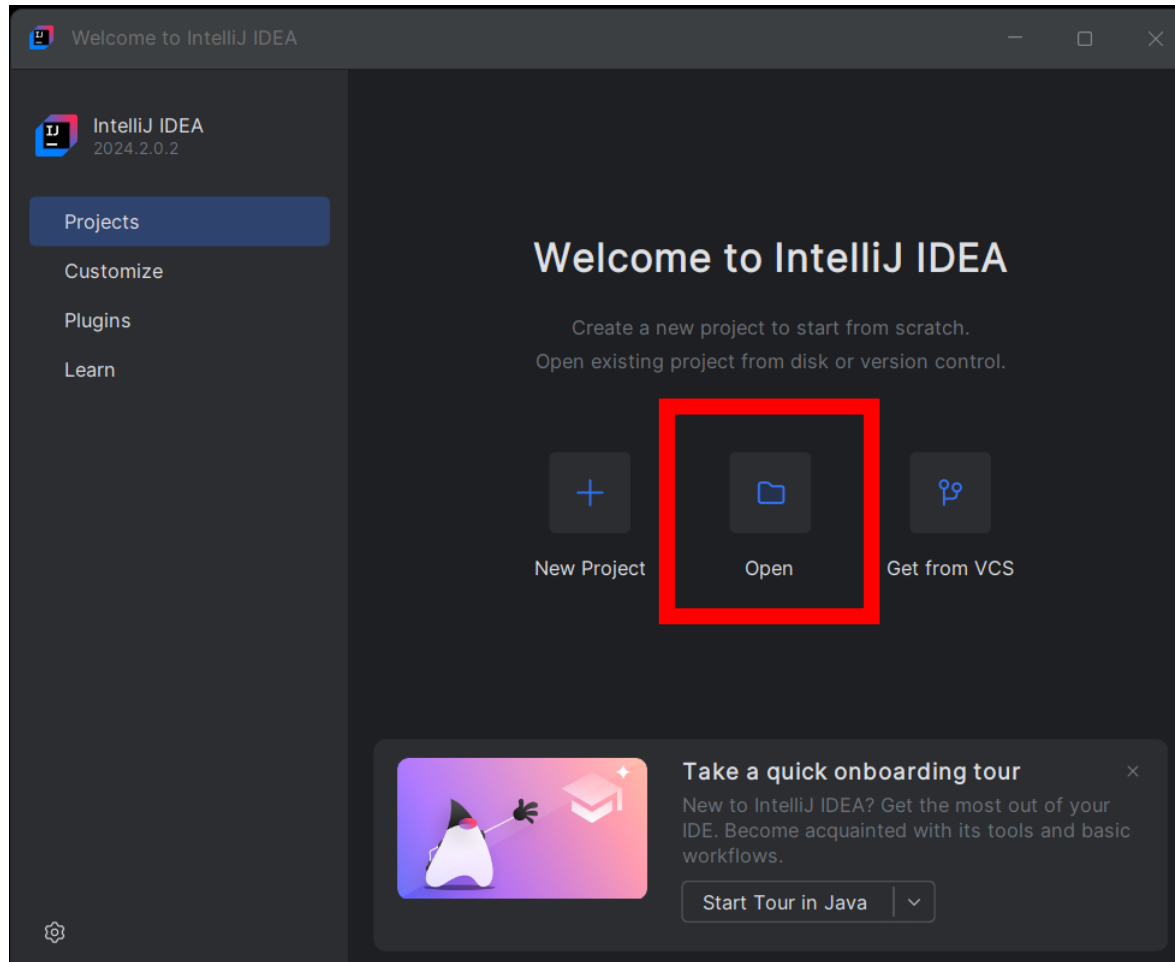
- **O Spring Initializr irá realizar o download do seu projeto em formato ZIP, descompacte o arquivo em sua pasta de projetos.**



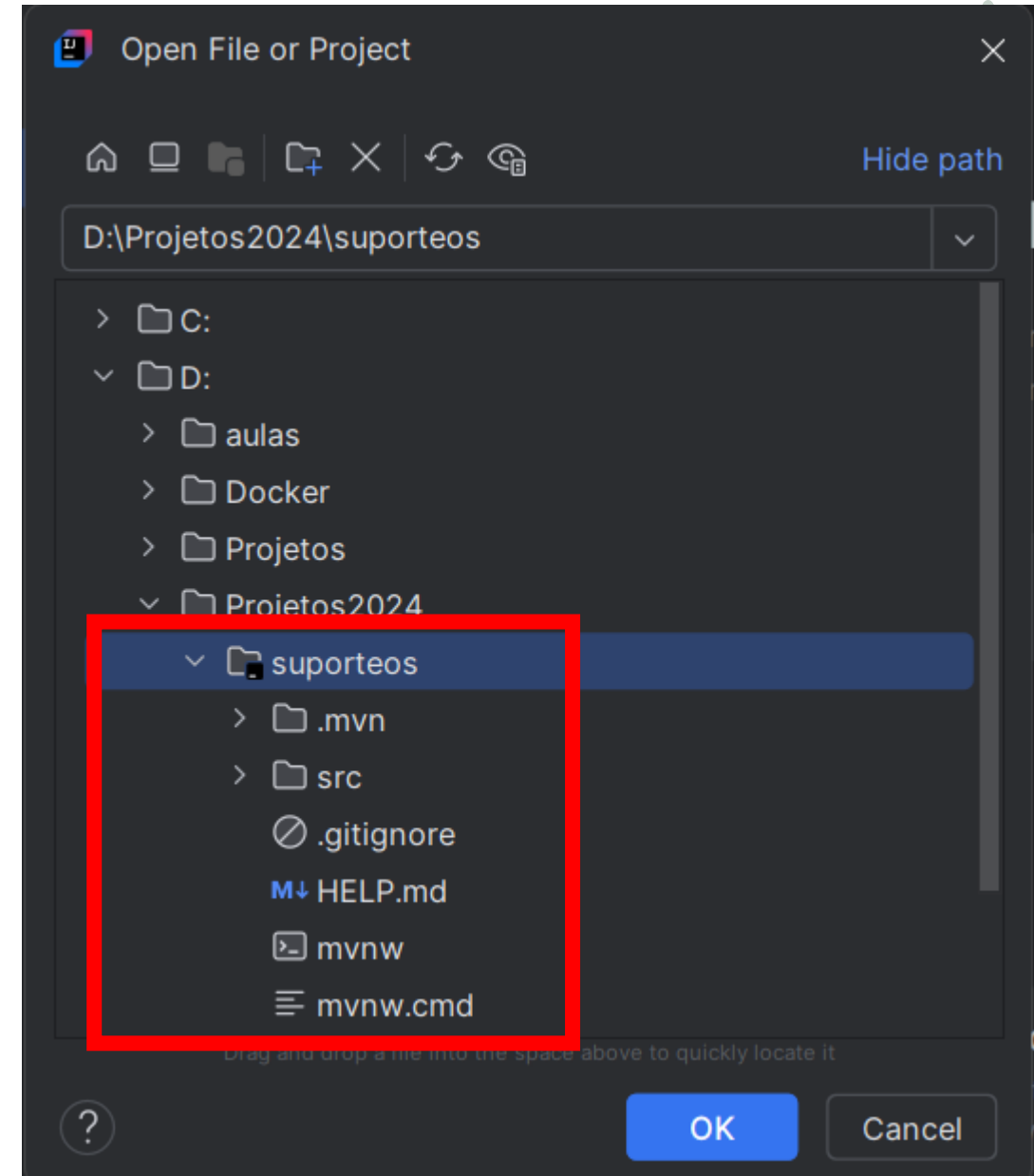
- Descompacte o arquivo zipado para sua pasta de projeto.
- **A pasta descompactada tem o projeto gerado pelo Spring Initializr**

- Abrindo o Projeto

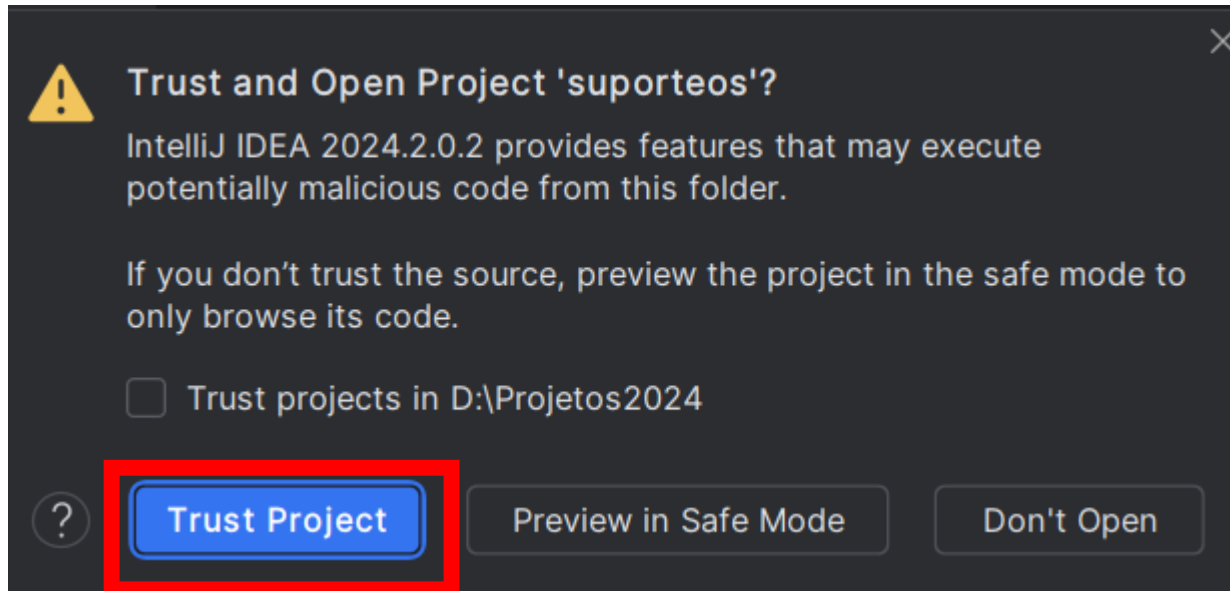
Criando Projeto Spring Boot



- **Abra o IntelliJ e abra sua pasta de projeto: suporteos**

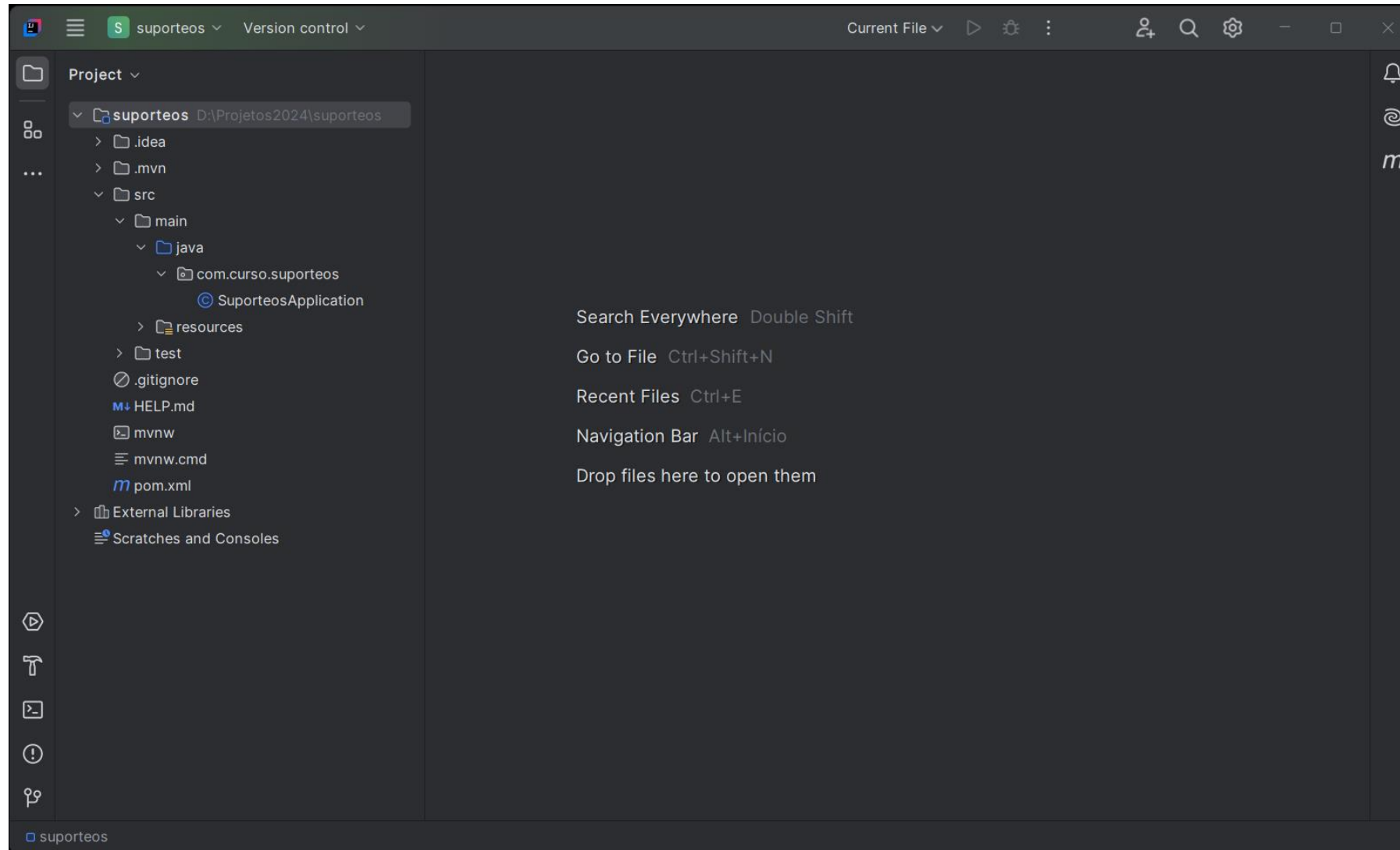


- Abrindo o Projeto



- Clique em **Trust Project**.
- Se sua pasta de projetos for sempre a mesma, marque também a opção **Trust projects in**

- Abrindo o Projeto

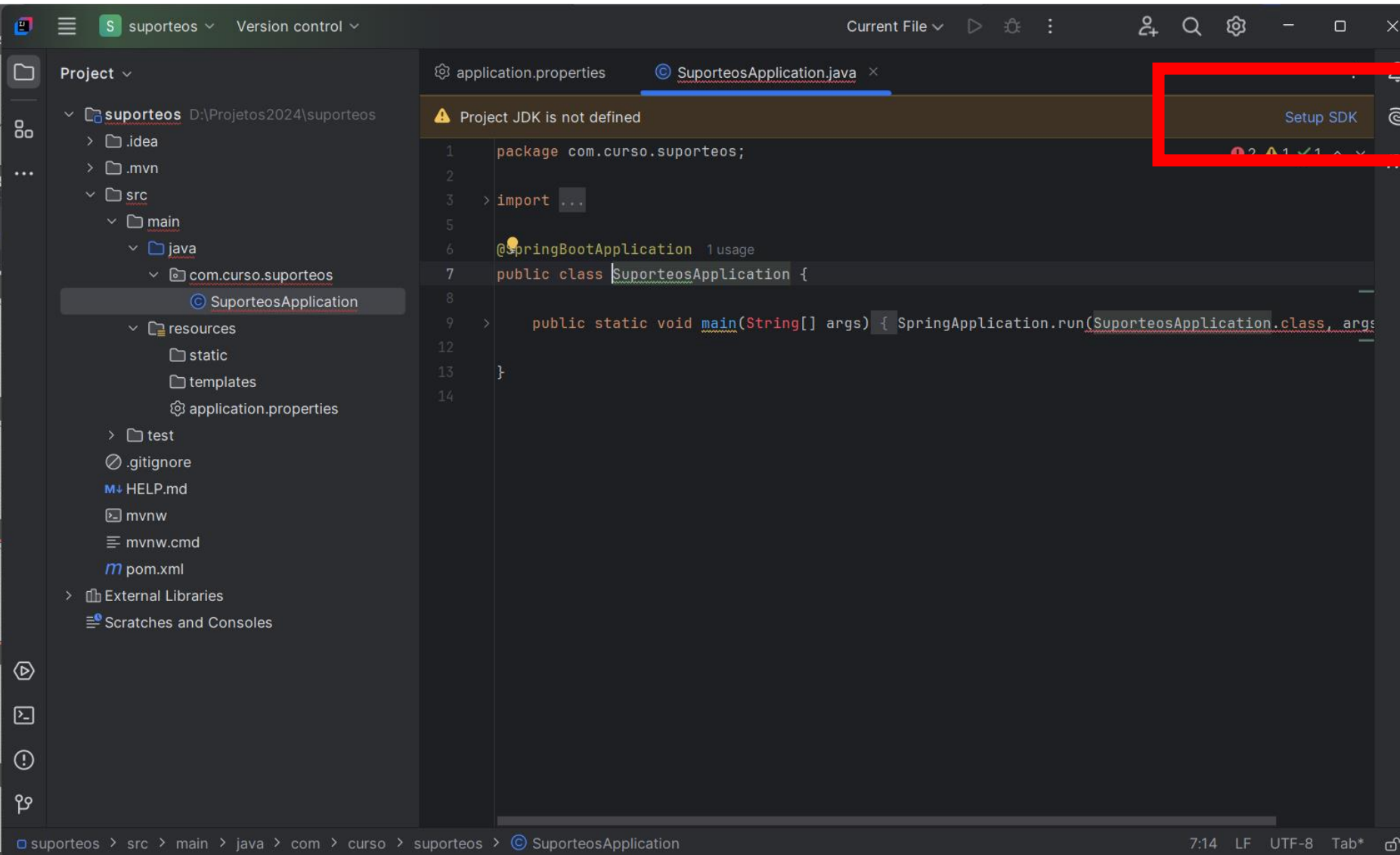


- Aqui temos nosso projeto gerado.

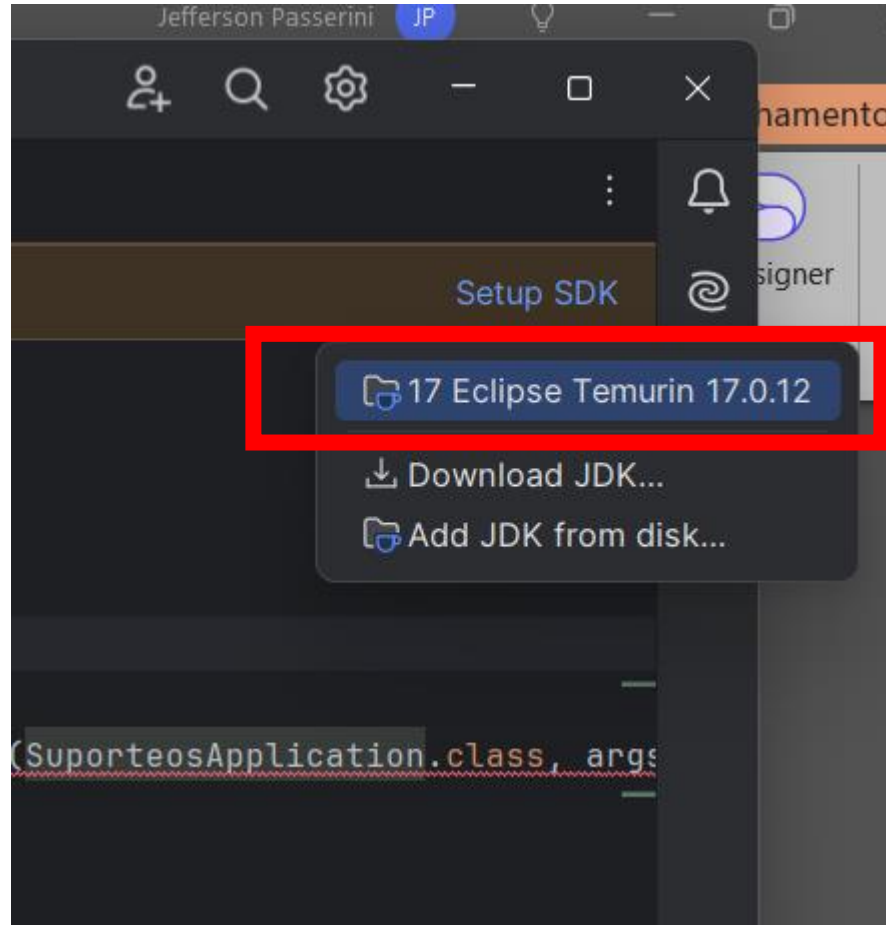
- Abrindo o Projeto

Criando Projeto Spring Boot

- Abra a classe principal do seu projeto.
- **Observe que provavelmente sua Java JDK não foi identificada.**
- **Clique em Setup JDK**



- Abrindo o Projeto



- Escolha o Java JDK que instalamos anteriormente.
- **Importante: como geramos no Spring Initializr o projeto com Java 17 temos que possuir a JDK do Java 17 instalado em nossa máquina.**

Abrindo o Projeto

Criando Projeto Spring Boot

- Execute o projeto gerado.

- Abra no navegador:

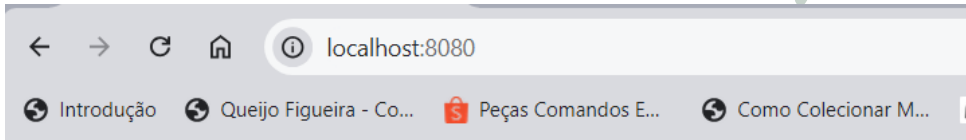
- **http://localhost:8080**

The screenshot shows an IDE with the project 'suporteos' open. The file 'SuporteosApplication.java' is selected, showing the following code:

```
1 package com.curso.suporteos;
2
3 import ...
4
5
6 @SpringBootApplication
7 public class SuporteosApplication {
8
9     public static void main(String[] args) { SpringApplication.run(SuporteosApplication.class, args);
10 }
11
12
13
14
```

The Run console shows the following output:

```
2024-08-27T13:00:38.503-03:00 INFO 22392 --- [suporteos] [ restartedMain] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at
2024-08-27T13:00:38.722-03:00 INFO 22392 --- [suporteos] [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing Pe
2024-08-27T13:00:38.823-03:00 INFO 22392 --- [suporteos] [ restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM
2024-08-27T13:00:38.870-03:00 INFO 22392 --- [suporteos] [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level
2024-08-27T13:00:39.292-03:00 INFO 22392 --- [suporteos] [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup:
2024-08-27T13:00:39.714-03:00 INFO 22392 --- [suporteos] [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platfo
2024-08-27T13:00:39.730-03:00 INFO 22392 --- [suporteos] [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityMa
2024-08-27T13:00:39.792-03:00 WARN 22392 --- [suporteos] [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view
2024-08-27T13:00:40.198-03:00 INFO 22392 --- [suporteos] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is run
2024-08-27T13:00:40.230-03:00 INFO 22392 --- [suporteos] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8
2024-08-27T13:00:40.230-03:00 INFO 22392 --- [suporteos] [ restartedMain] c.curso.suporteos.SuporteosApplication : Started SuporteosApplica
```



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Aug 27 13:02:27 BRT 2024

There was an unexpected error (type=Not Found, status=404).

No static resource .

org.springframework.web.servlet.resource.NoResourceFoundException: No static resource .

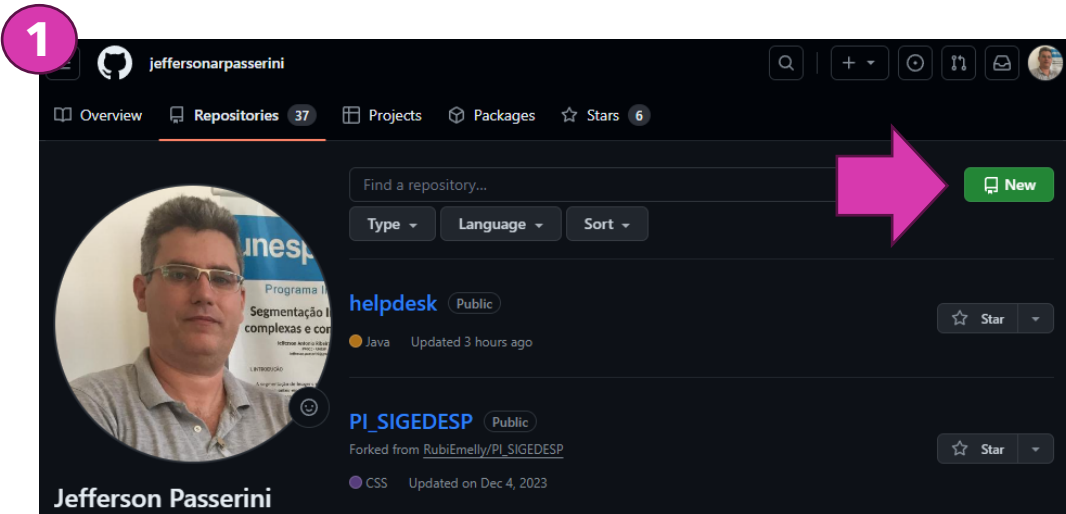
at org.springframework.web.servlet.resource.ResourceHttpRequestHandler.handleRequest(ResourceHttpRequestHandler.java:116)
at org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter.handle(HttpRequestHandlerAdapter.java:53)
at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1089)
at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:979)
at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1014)
at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:903)
at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:564)
at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:885)
at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:658)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:195)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
at org.apache.catalina.core.WebsocketServerFilter.doFilter(WebsocketServerFilter.java:51)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:164)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
at org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.java:100)
at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:164)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
at org.springframework.web.filter.FormContentFilter.doFilterInternal(FormContentFilter.java:93)
at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:116)



Spring Boot

Criando Repositório no Github

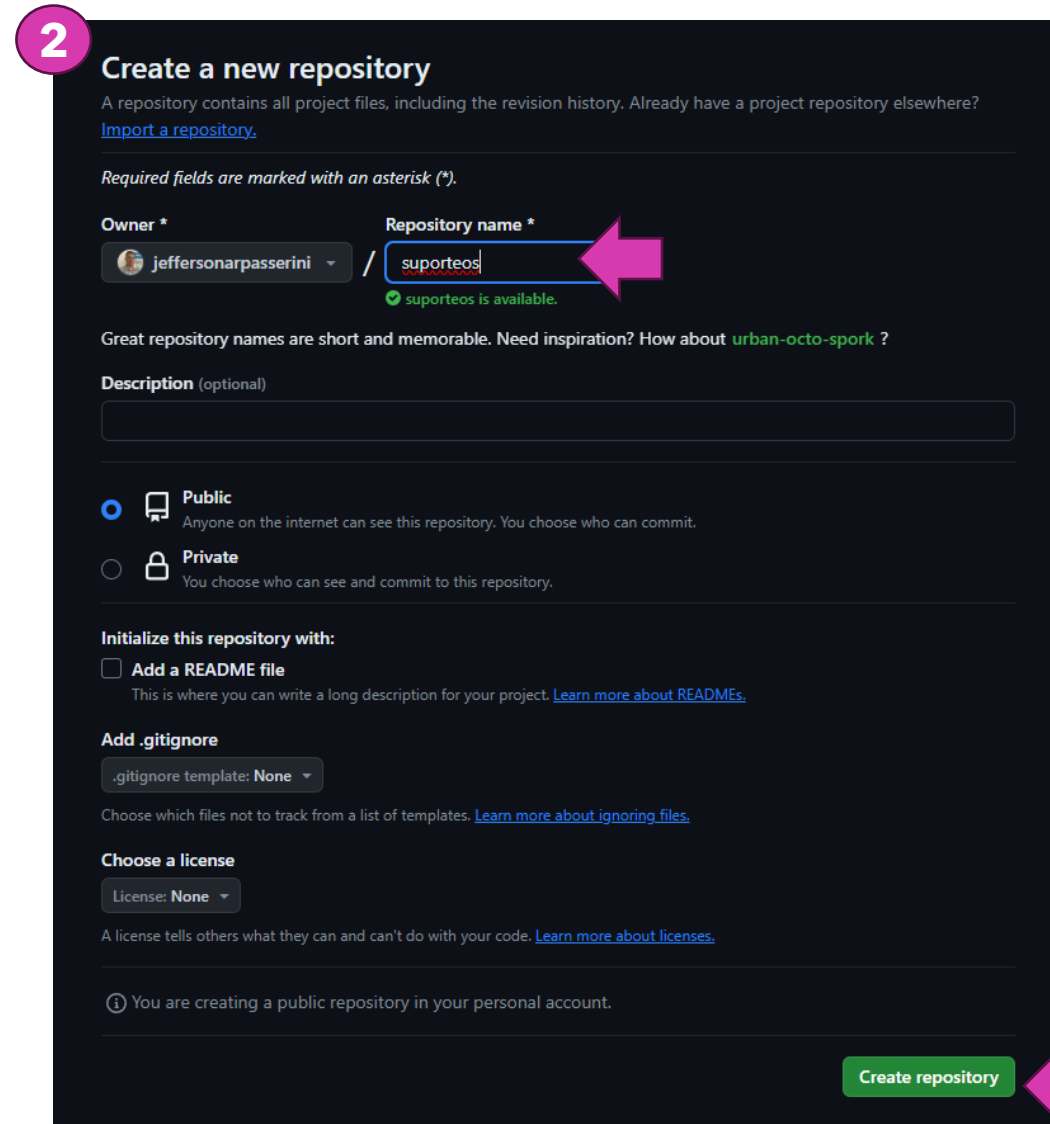
Criando Projeto Spring Boot – Utilizando Git/Github





1 - Criar um repositório no Github para nosso projeto


2 - Informe o nome do repositório (utilizo o mesmo nome do projeto)


3 - Selecione "Create Repository"




suporteosPublic

Pin

Unwatch1

Fork0

Star0



Set up GitHub Copilot

Use GitHub's AI pair programmer to autocomplete suggestions as you code.

Get started with GitHub Copilot



Add collaborators to this repository

Search for people using their GitHub username or email address.

Criando Projeto Spring Boot

Quick setup — if you've done this kind of thing before

Set up in Desktop

 or

HTTPS

SSH

git@github.com:jeffersonarpasserini/suporteos.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# suporteos" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:jeffersonarpasserini/suporteos.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:jeffersonarpasserini/suporteos.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

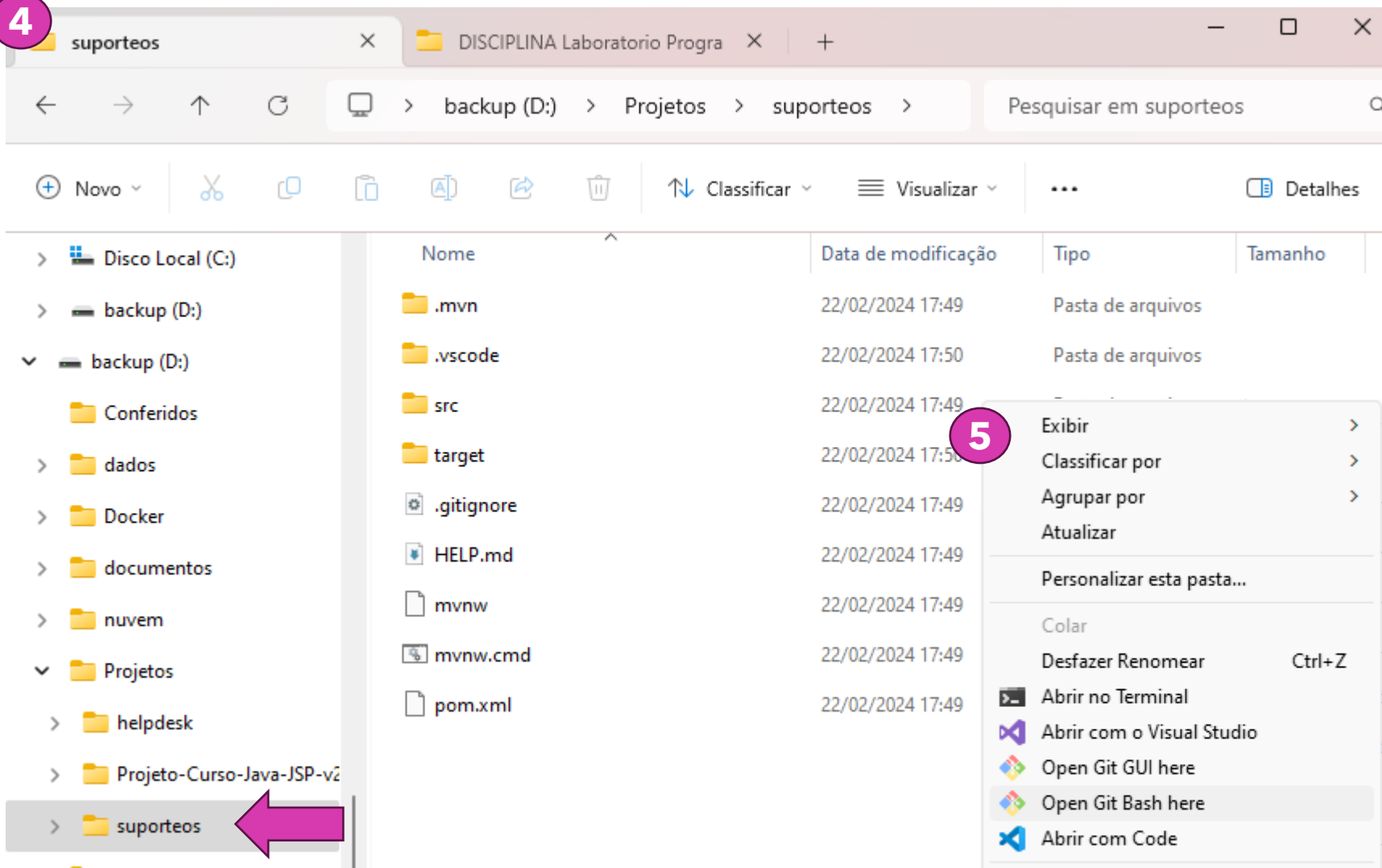
Import code

Criando Projeto Spring Boot – Utilizando Git/Github

3

Instruções para criarmos nosso repositório Git em nosso projeto e realizarmos a primeira sincronização

4



5

1 - Na pasta do projeto que criamos clique com o botão direito do seu mouse para abrir o menu suspenso.

2 - Escolha a opção Open Git Bash here

MINGW64:/d/Projetos/suporteos

```
jefferson@MarsNotebook MINGW64 /d/Projetos/suporteos (main)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bu
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultBranch=main
user.email=jefferson.passerini@gmail.com
user.name=jeffersonarpasserini
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=git@github.com:jeffersonarpasserini/suporteos.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.main.remote=origin
branch.main.merge=refs/heads/main
branch.main.vscode-merge-base=origin/main
```

6 - Defina as configurações de seu usuário do Github no Git local de sua máquina

Digite os comandos:

git config --global user.name NomedoUsuário

git config --global user.email email@email.com

O NomedoUsuario e o Email são os mesmos que da sua conta online no Github.

Depois verifique:

git config --list

```
jeffe@CanisMajoris MINGW64 /d/Projetos/suporteos
$ git init
Initialized empty Git repository in D:/Projetos/suporteos/.git/

jeffe@CanisMajoris MINGW64 /d/Projetos/suporteos (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        .mvn/
        mvnw
        mvnw.cmd
        pom.xml
        src/

nothing added to commit but untracked files present (use "git add"
to track)

jeffe@CanisMajoris MINGW64 /d/Projetos/suporteos (main)
$ |
```

1 - Inicializa o Repositório Git Local

2 - Verifique o status de seu repositório. Observe-se que existem arquivos que ainda não estão gerenciados pelo Git



A terminal window titled 'MINGW64:/d/Projetos/suporteos' with standard window controls. The prompt is 'jefferson@MarsNotebook MINGW64 /d/Projetos/suporteos (main)'. The text '(main)' is highlighted with a red rectangle. A red arrow points from this rectangle to a separate code block below.

```
jefferson@MarsNotebook MINGW64 /d/Projetos/suporteos (main)
$ |
```

```
/d/Projetos/suporteos (main)
```

Verifique se a Branch criada é "(main)" assim não teremos problemas na sincronização com o github.

Se for outra Branch digite o seguinte comando:

git Branch -M main

```
jeffe@CanisMajoris MINGW64 /d/Projetos/suporteos (main)  
$ git add .
```

3 - Adiciona os arquivos para serem gerenciados pelo Git.

```
jeffe@CanisMajoris MINGW64 /d/Projetos/suporteos (main)  
$ git status  
On branch main  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   .gitignore  
    new file:   .mvn/wrapper/maven-wrapper.jar  
    new file:   .mvn/wrapper/maven-wrapper.properties  
    new file:   mvnw  
    new file:   mvnw.cmd  
    new file:   pom.xml  
    new file:   src/main/java/com/curso/suporteos/SuporteosApplica  
tion.java  
    new file:   src/main/resources/application.properties  
    new file:   src/test/java/com/curso/suporteos/SuporteosApplica  
tionTests.java
```

4 - Verifique o status de seu repositório novamente. Observa-se que os arquivos estão gerenciados pelo Git mas não houve nenhum commit.

```
jeffe@CanisMajoris MINGW64 /d/Projetos/suporteos (main)
$ git commit -m "inicialização do projeto"
[main (root-commit) 62056e6] inicialização do projeto
9 files changed, 640 insertions(+)
create mode 100644 .gitignore
create mode 100644 .mvn/wrapper/maven-wrapper.jar
create mode 100644 .mvn/wrapper/maven-wrapper.properties
create mode 100644 mvnw
create mode 100644 mvnw.cmd
create mode 100644 pom.xml
create mode 100644 src/main/java/com/curso/suporteos/SuporteosApp1
lication.java
create mode 100644 src/main/resources/application.properties
create mode 100644 src/test/java/com/curso/suporteos/SuporteosApp1
licationTests.java
```

1 - Faça o commit nos arquivos gerenciados.

Atenção !

É muito importante a mensagem que é colocada no git commit pois ela vai identificar facilmente a alteração que foi realizada no pacote.

Criando Projeto Spring Boot – Utilizando Git/Github

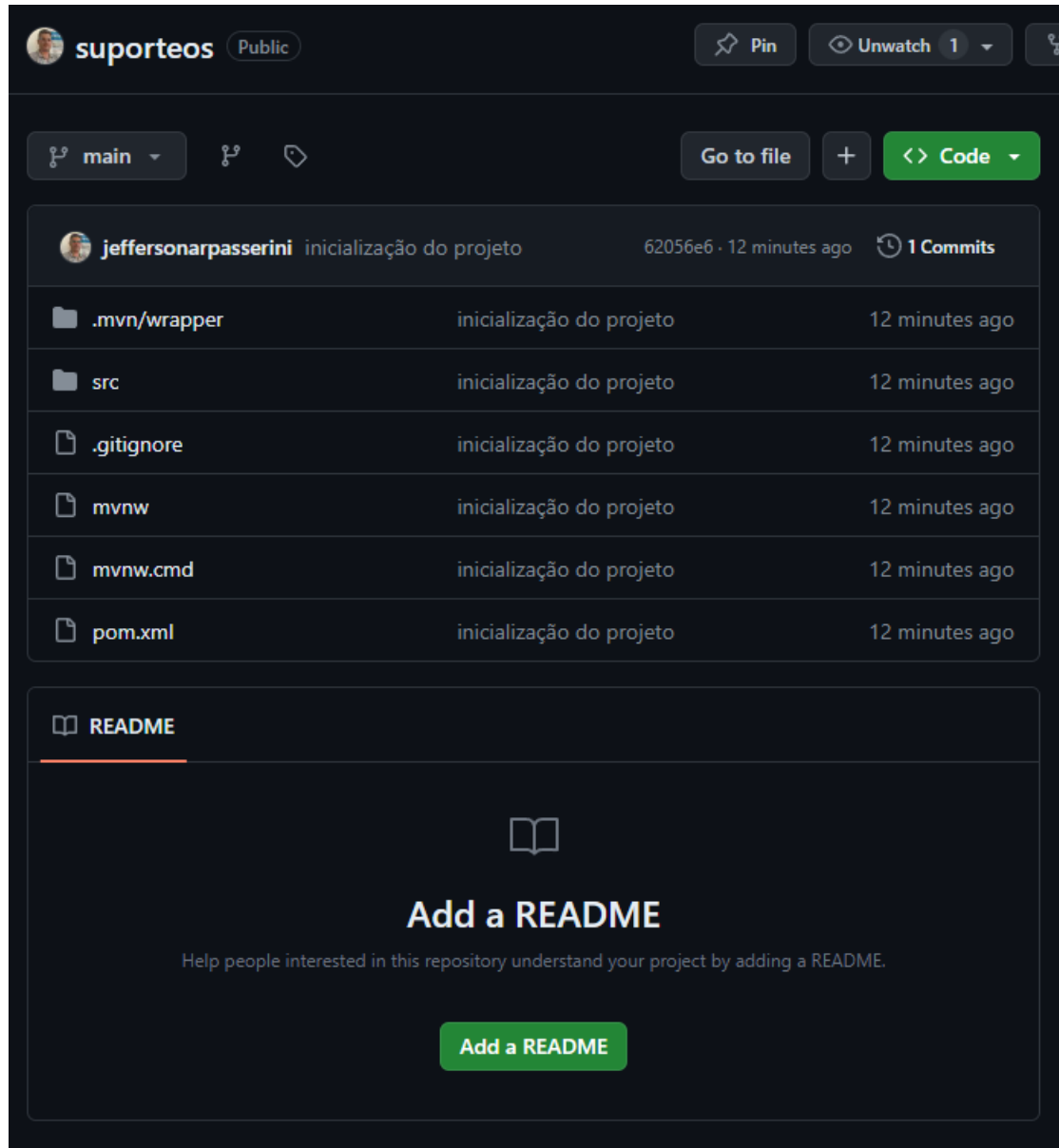
```
jeffe@CanisMajoris MINGW64 /d/Projetos/suporteos (main)  
$ git remote add origin git@github.com:jeffersonarpasserini/suporteos.git
```

Associe o seu repositório remoto no github ao seu repositório local.

```
jeffe@CanisMajoris MINGW64 /d/Projetos/suporteos (main)  
$ git push -u origin main  
Enumerating objects: 25, done.  
Counting objects: 100% (25/25), done.  
Delta compression using up to 16 threads  
Compressing objects: 100% (16/16), done.  
Writing objects: 100% (25/25), 62.97 KiB | 454.00 KiB/s, done.  
Total 25 (delta 0), reused 0 (delta 0), pack-reused 0  
To github.com:jeffersonarpasserini/suporteos.git  
 * [new branch]      main -> main  
branch 'main' set up to track 'origin/main'.
```

E faça a sincronização do projeto para o repositório remoto.

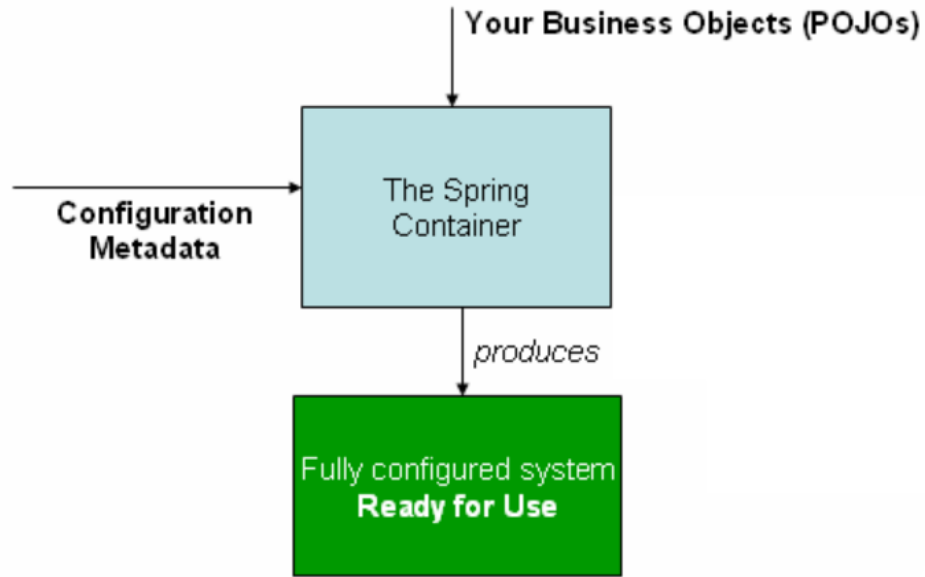
Criando Projeto Spring Boot – Utilizando Git/Github



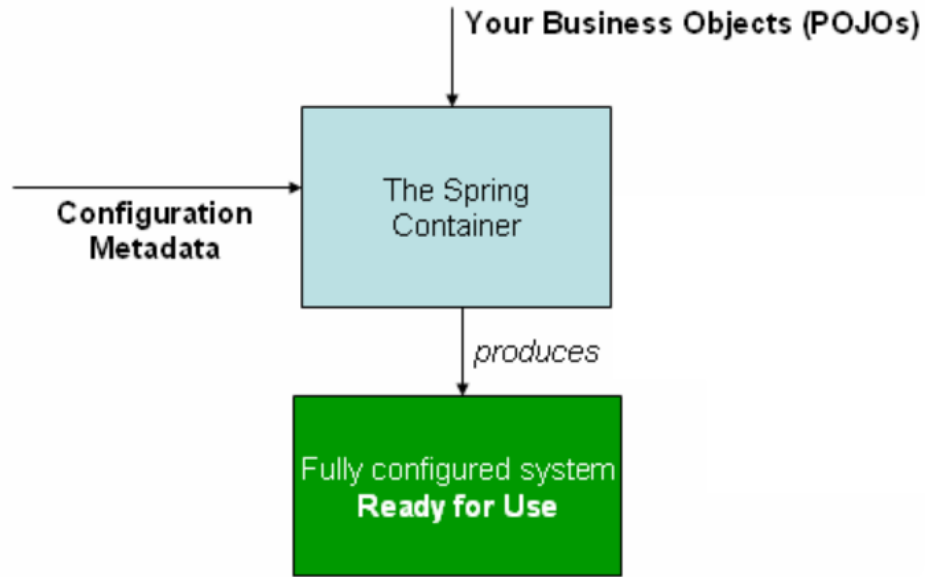
Agora nosso projeto está sincronizado com o repositório remoto.



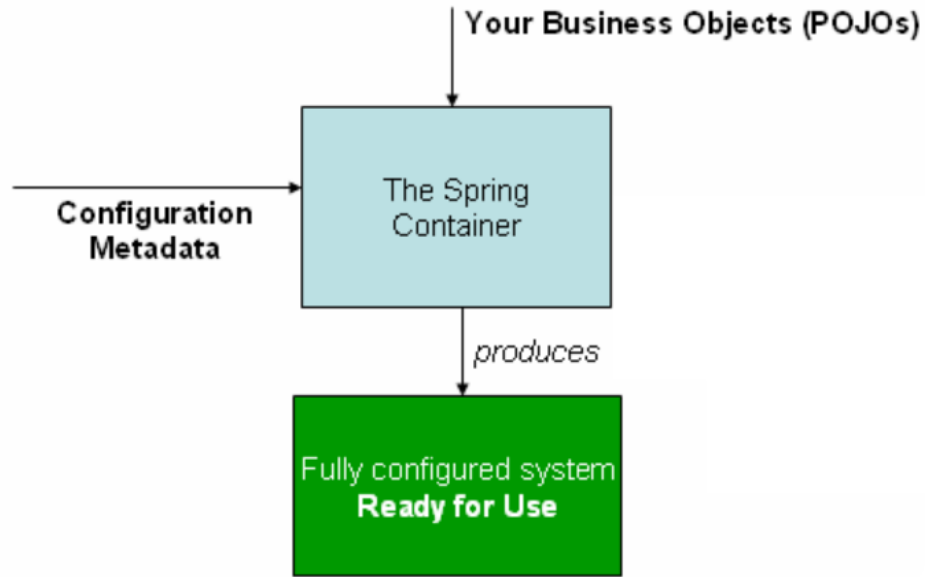
Spring Boot Container IOC



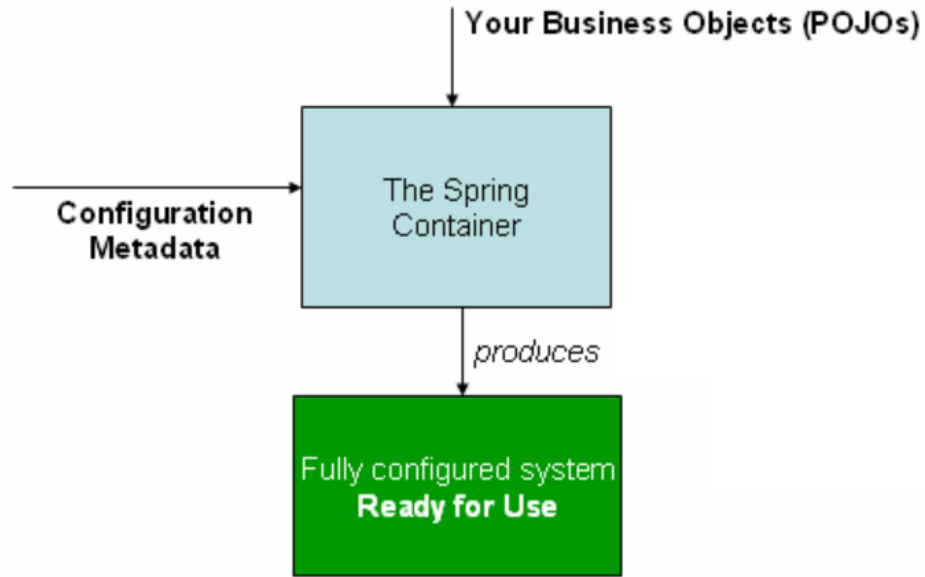
- Inversion of Control (IOC) → é o processo pelo qual o Spring Boot gerencia as injeções de dependência através de um container de injeção, ou seja, o framework fica responsável por gerenciar as instâncias dos objetos definidos em seu projeto.
- A definição das responsabilidades para a injeção de dependência no Spring é definida através de annotations que define os componentes chave: @Entity, @Repository, @Controllers, @Services.



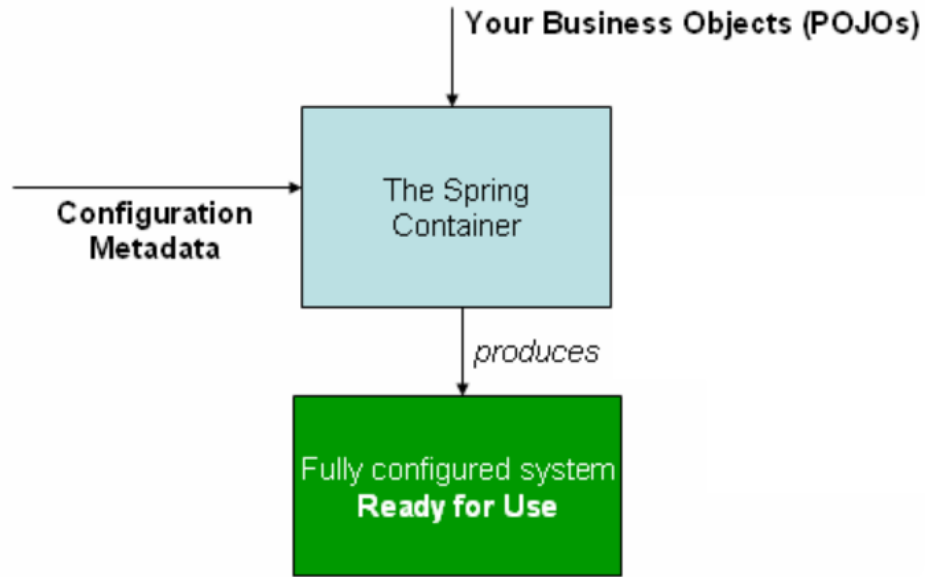
- A injeção de dependência permite que os componentes sejam desacoplados e facilita a reutilização de código, testes unitários e a manutenção do código.
- Ela remove a responsabilidade de criar instâncias e gerenciar dependências das classes individuais, deixando essa tarefa para o contêiner de injeção de dependência.



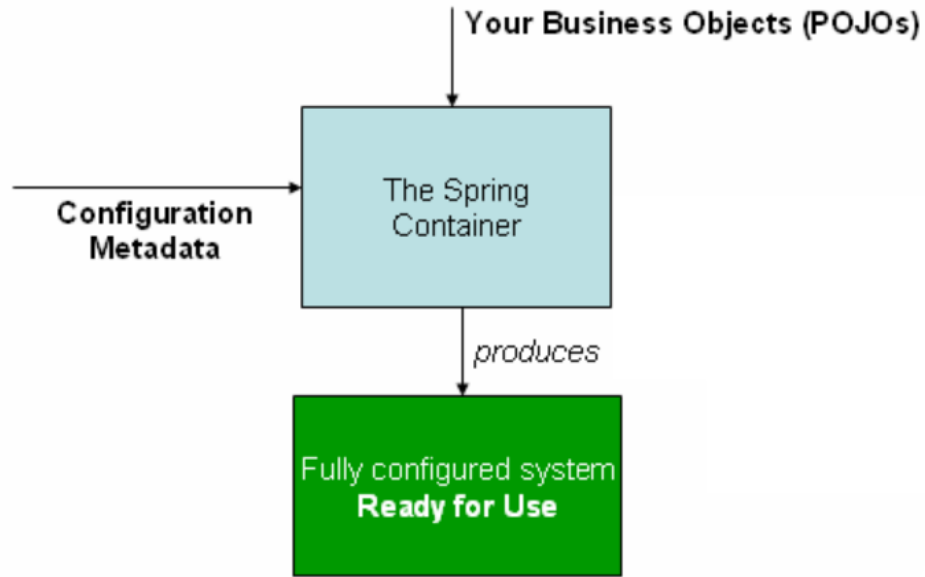
- Em uma API REST desenvolvida com Spring Boot, o próprio Spring Boot e o framework Spring IOC (inversion of control) são responsáveis por gerenciar as dependências e realizar a injeção de dependência automaticamente, desde que as anotações corretas sejam utilizadas (por exemplo **@Autowired**).
- Dessa forma, você não precisa se preocupar em criar manualmente as instâncias de dependências nos seus componentes.



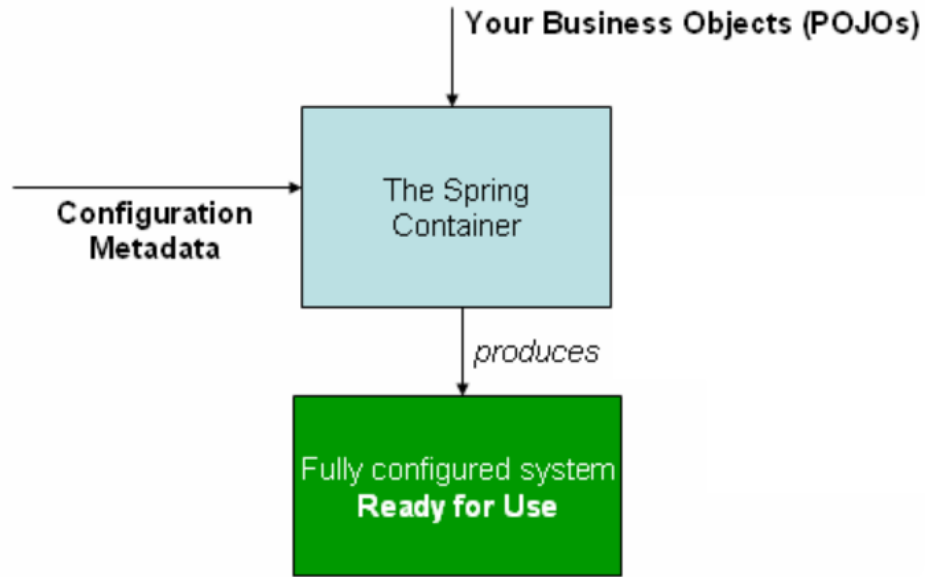
- **@Entity** → faz parte do framework de persistência do Java chamada JPA (Java Persistence API), é utilizada para indicar que uma classe representa uma entidade persistente, ou seja, uma classe que será mapeada para uma tabela em um banco de dados relacional.



- **@Repository** → são responsáveis por acessar o banco de dados ou qualquer outro meio de armazenamento de dados. Eles fornecem métodos para realizar operações CRUD nos objetos de domínio. A injeção de dependência é usada aqui para fornecer a conexão com o banco de dados ou qualquer outro componente necessário para acessar os dados.



- **@Controllers** → Os controladores são responsáveis por receber as requisições HTTP e direcioná-las para o código apropriado. Elas definem os endpoints da API e mapeiam as requisições para métodos específicos.
- A injeção de dependência é comumente usada para fornecer serviços para os controladores.
- Por exemplo, você pode injetar um serviço de Produto para que ele possa acessar os métodos desse serviço e executar a lógica necessária.



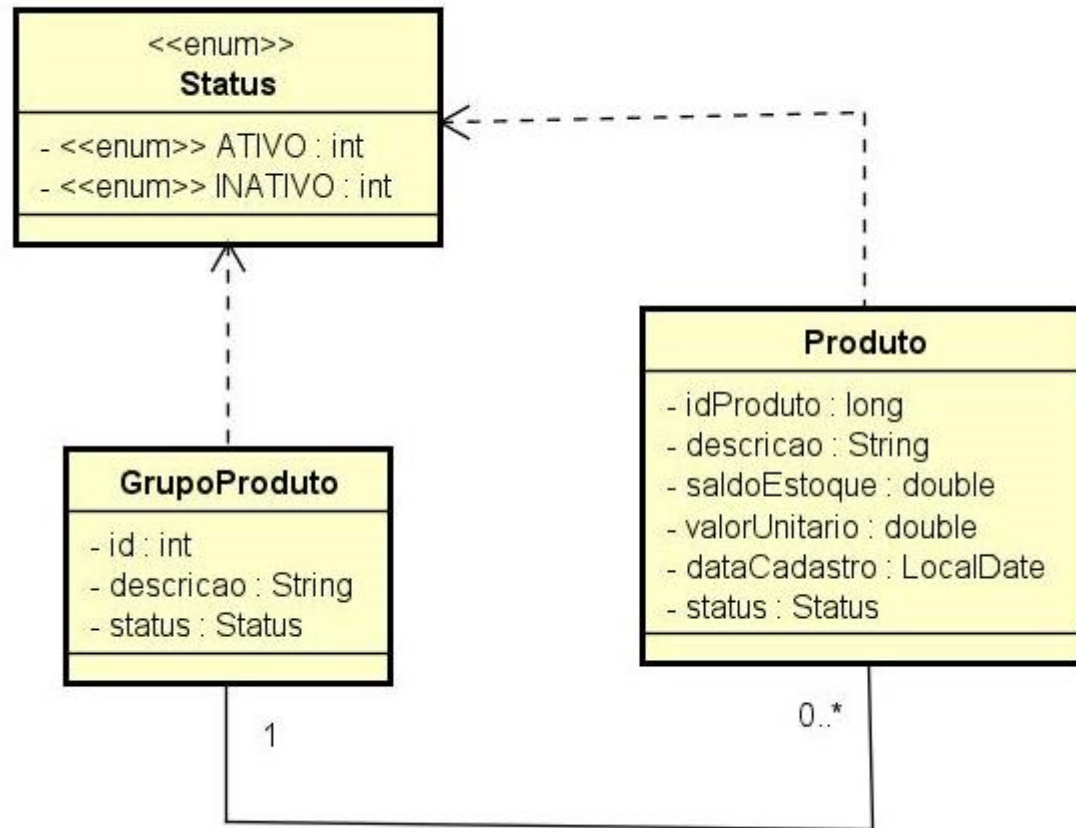
- **@Services** → Os serviços são responsáveis por executar a lógica de negócio da aplicação.
- Eles encapsulam ações mais complexas que podem ser necessárias nos controladores.
- A injeção de dependência é útil aqui para fornecer outras dependências necessárias aos serviços.



Spring Boot Diagrama de Classe do Projeto

Criando Projeto Spring Boot

- Modelo do projeto (Fase 1) que será desenvolvido durante as aulas.

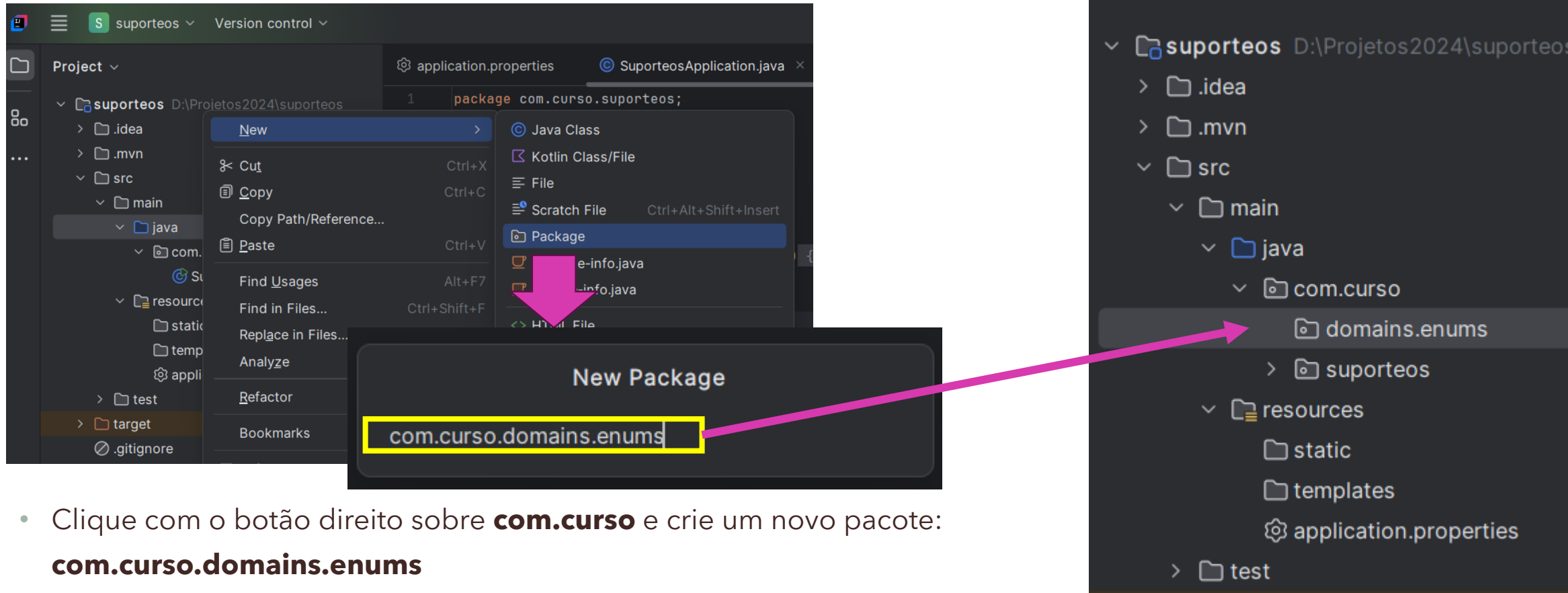




Spring Boot

Criando os Tipos Enumerados - enums

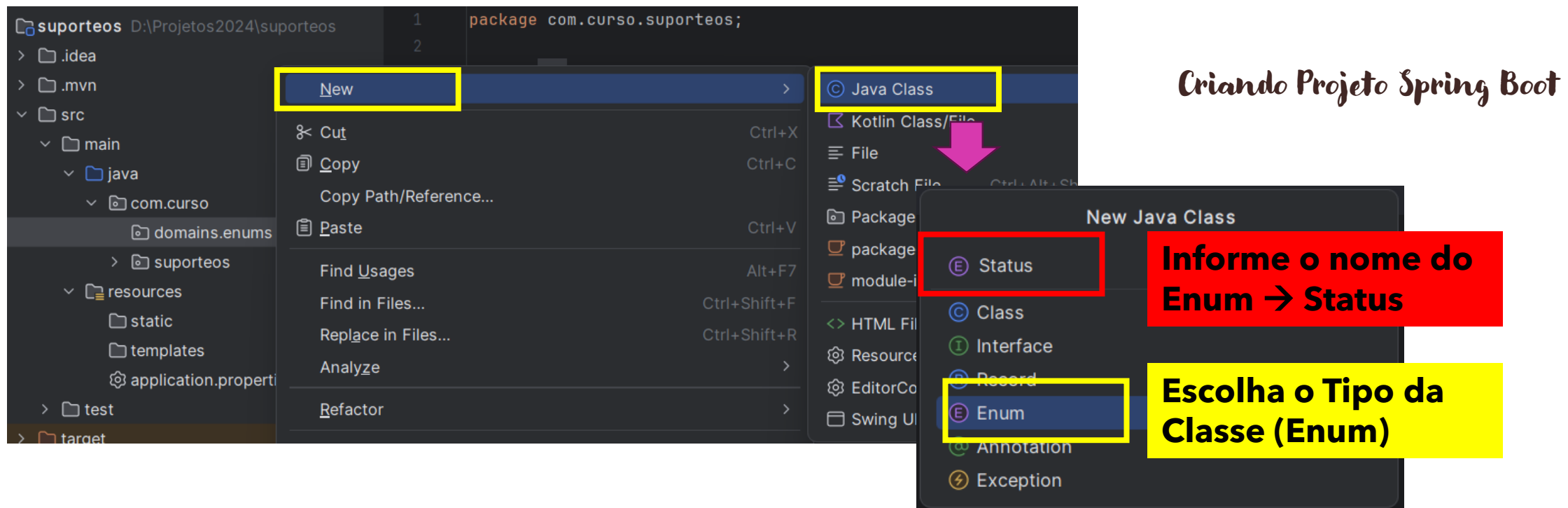
- Continuando nosso projeto vamos criar os pacotes para armazenar nossas models e enums



- Clique com o botão direito sobre **com.curso** e crie um novo pacote: **com.curso.domains.enums**
- Como enums é um subpacote de domains, domains também será criado.
- Atenção! Estamos colocando os nomes no plural devido ao problema de conflitos com palavras reservadas da linguagem Java e do framework Spring.
- Deixe a estrutura do seu projeto como demonstrado na figura.

- **Enums**
- Uma classe do tipo enum (abreviação de "enumeration") é uma estrutura em linguagens de programação que define um conjunto fixo e nomeado de constantes.
- Cada constante dentro de um enum é considerada um valor distinto do tipo enum.
- Isso é útil quando você deseja trabalhar com valores que representam um conjunto limitado de opções, evitando erros de atribuição ou a necessidade de usar valores literais, como inteiros ou strings, que podem ser inconsistentes ou suscetíveis a erros de digitação.

```
enum Cores {  
    Vermelho,  
    Azul,  
    Verde  
}
```



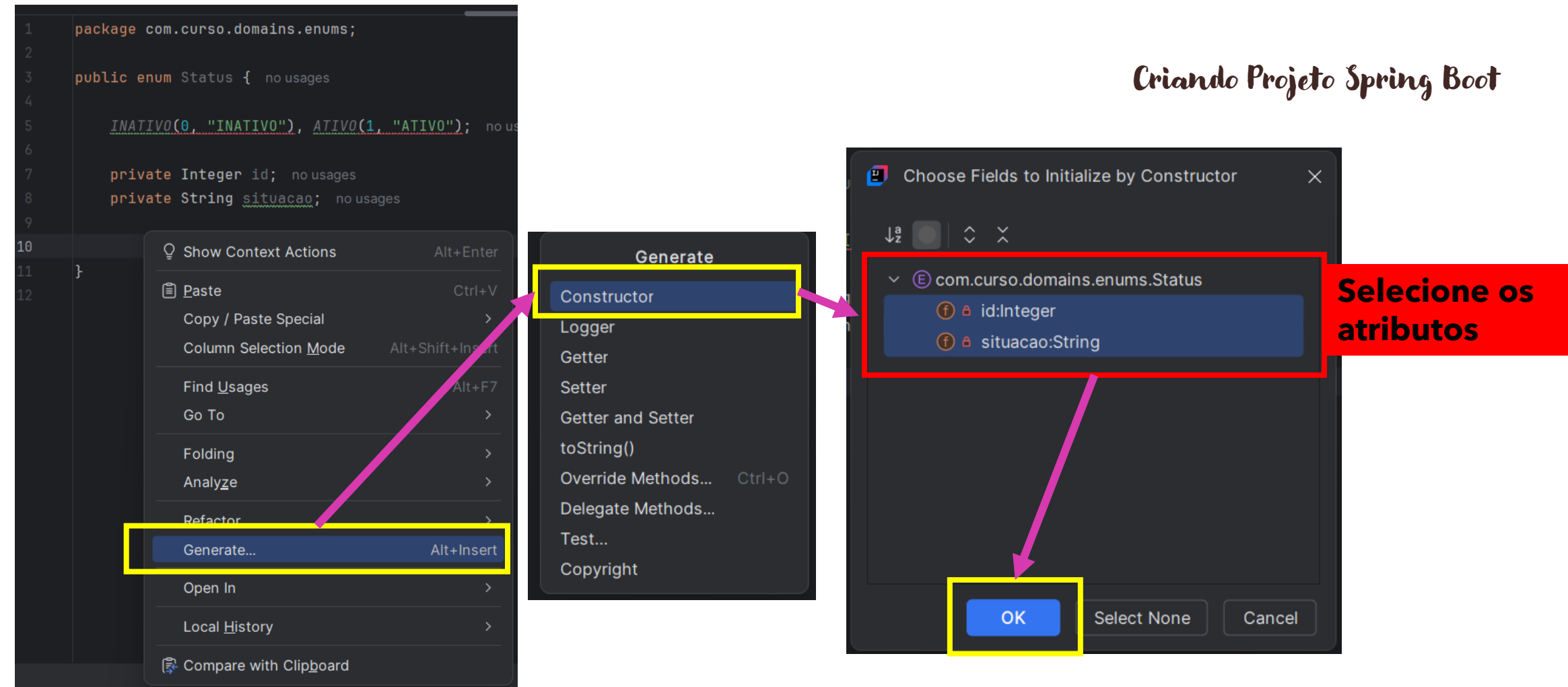
- 1 - Agora clique com o botão direito sobre a pasta enum e escolha as opções New Java File → Enum
- 2 - Informe o nome do arquivo (**Status**)
- O enum Status irá identificar se o cadastro está Ativo ou Inativo.

Criando Projeto Spring Boot

```
1 package com.curso.domains.enums;  
2  
3 public enum Status { no usages  
4  
5     1 INATIVO(0, "INATIVO"), ATIVO(1, "ATIVO"); no usages  
6  
7     2 private Integer id; no usages  
8     private String situacao; no usages  
9  
10 }
```

- 1 - Defina os tipos enumerados que irão representar os status possíveis, onde Inativo terá o valor 0 e o Ativo terá valor 1;
- 2 - Criar atributos para representar as informações dos tipos enumerados.

Criando Projeto Spring Boot

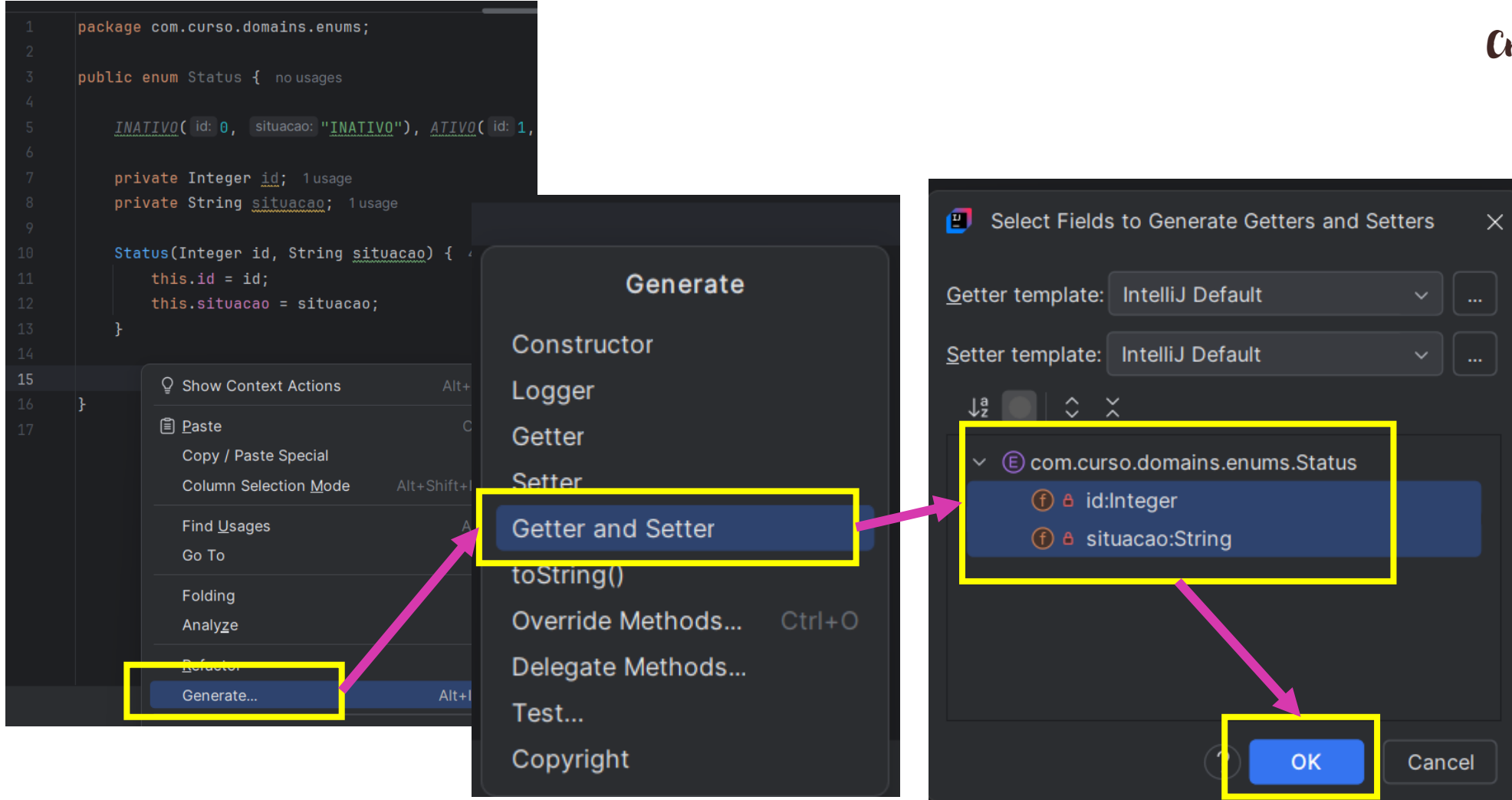


- 1 - Clique com botão direito no seu enum e escolha a opção Generate → Constructor e selecione os atributos e clique no botão OK.

Criando Projeto Spring Boot

```
1 package com.curso.domains.enums;
2
3 public enum Status { no usages
4
5     INATIVO(id: 0, situacao: "INATIVO"), ATIVO(id: 1, situacao: "ATIVO"); n
6
7     private Integer id; 1 usage
8     private String situacao; 1 usage
9
10    Status(Integer id, String situacao) { 4 usa
11        this.id = id;
12        this.situacao = situacao;
13    }
14
15
16 }
```

- Código gerado para o construtor.



- 1 - Clique com botão direito no seu enum e escolha a Generate → Getters and Setters.
- 2 - Selecione os atributos a serem gerados e clique em ok.

ⓔ Status.java ×

```
1 package com.curso.domains.enums;
2
3 public enum Status { no usages  ⚙ jeffersonarpasserini
4
5     INATIVO(id: 0, situacao: "INATIVO"), ATIVO(id: 1, situacao: "ATIVO"); no usag
6
7     private Integer id; 3 usages
8     private String situacao; 3 usages
9
10    Status(Integer id, String situacao) { 4 usages  ⚙ jeffersonarpasserini
11        this.id = id;
12        this.situacao = situacao;
13    }
14
15    > public Integer getId() { return id; }
16
17
18
19    > public void setId(Integer id) { this.id = id; }
20
21
22
23    > public String getSituacao() { return situacao; }
24
25
26
27    > public void setSituacao(String situacao) { this.situacao = situacao; }
28
29
30 }
```

Criando Projeto Spring Boot

- **Código do enum Status até agora.**

```
1 package com.curso.domains.enums;
2
3 public enum Status { 3 usages  jeffersonarpasserini *
4
5     INATIVO(id: 0, situacao: "INATIVO"), ATIVO(id: 1, situacao: "ATIVO"); no usag
6
7     private Integer id; 3 usages
8     private String situacao; 3 usages
9
10    Status(Integer id, String situacao) { 4 usages  jeffersonarpasserini
11        this.id = id;
12        this.situacao = situacao;
13    }
14
15    > public Integer getId() { return id; }
16
17
18
19    > public void setId(Integer id) { this.id = id; }
20
21
22
23    > public String getSituacao() { return situacao; }
24
25
26
27    > public void setSituacao(String situacao) { this.situacao = situacao; }
28
29
30
31    public static Status toEnum(Integer id){ no usages new *
32        if(id==null) return null;
33        for(Status x : Status.values()){
34            if(id.equals(x.getId())){
35                return x;
36            }
37        }
38        throw new IllegalArgumentException("Status invalido");
39    }
40 }
```

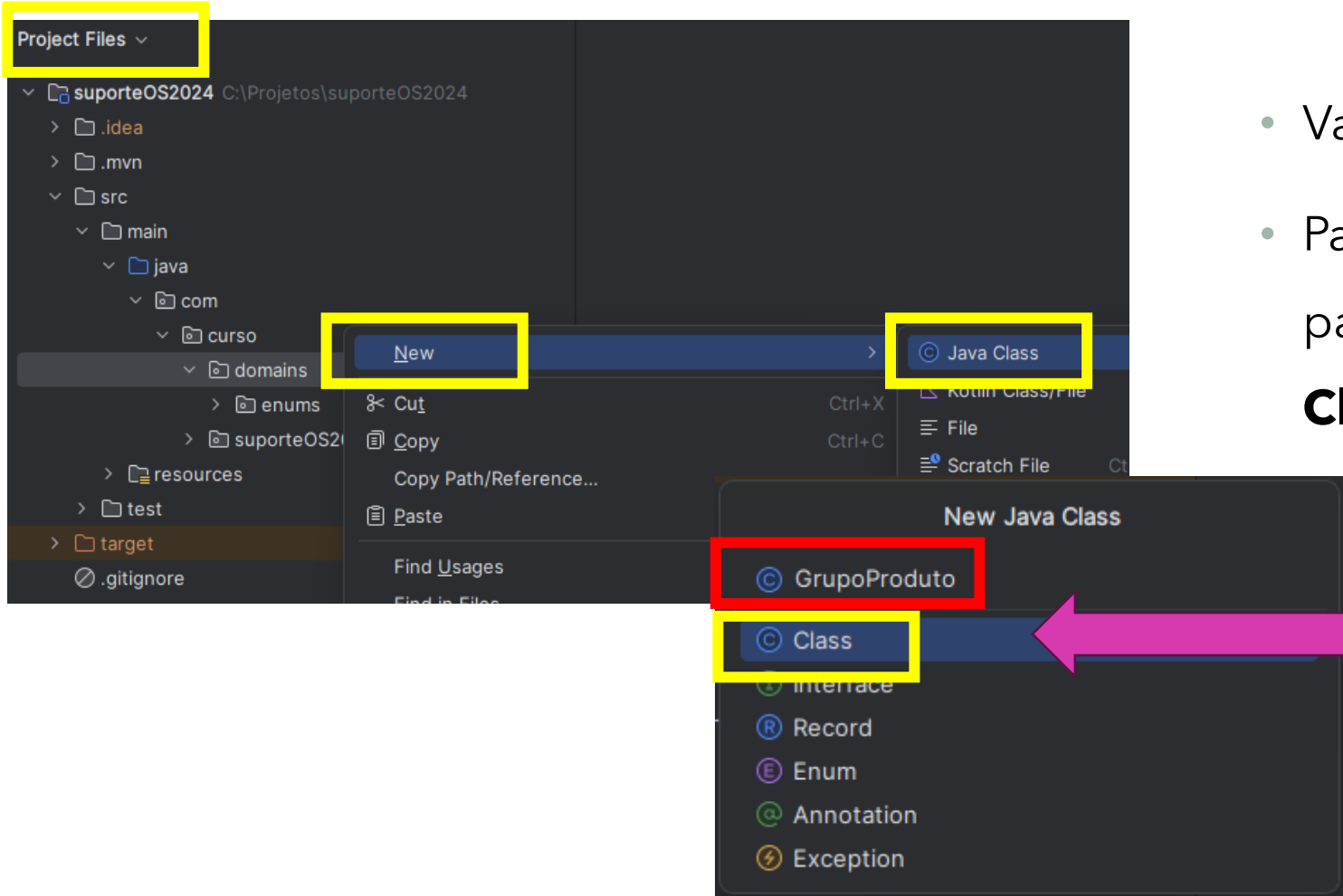
- Para finalizar nosso Enum:
- Crie um método do tipo **static** para que não seja necessário instanciar o enum para utilizá-lo.



Spring Boot

Criando as Entidades (Entities)

GrupoProduto



- Vamos criar nossas entidades do projeto.
- Para isso clique com o botão direito no pacote **domains** e escolha **New → Java Class**
- Na janela "New Java Class" escolha a **opção Class** e informe o nome da classe a ser criada: **GrupoProduto**.


```
1 package com.curso.domains;
2
3 import com.curso.domains.enums.Status;
4
5 import java.util.Objects;
6
7 public class GrupoProduto { 2 usages new *
8
9     private int id; 6 usages
10    private String descricao; 6 usages
11    private Status status; 3 usages
12 }
```

- Na classe **GrupoProduto** gerada insira os atributos da classe;
- Para o atributo status devemos fazer a importação do pacote **enums**.

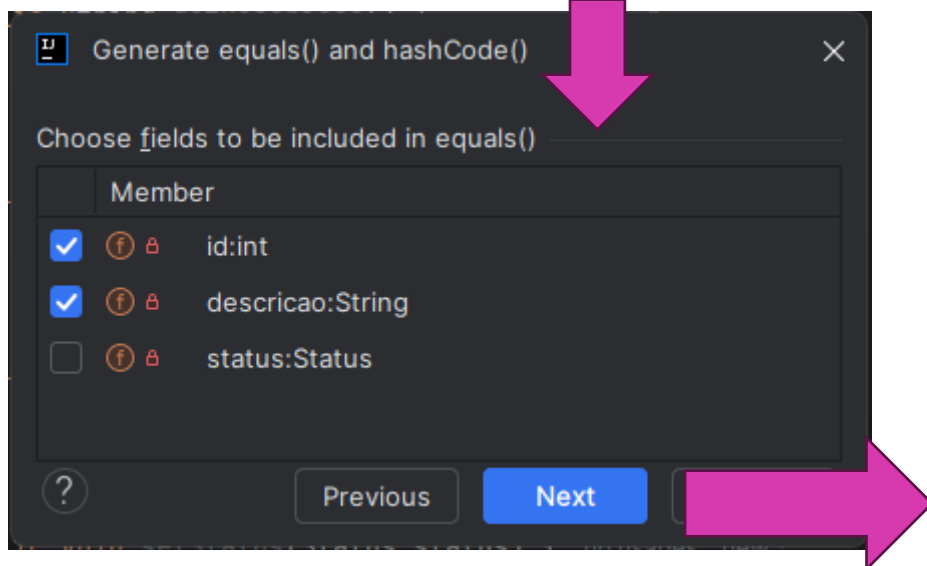
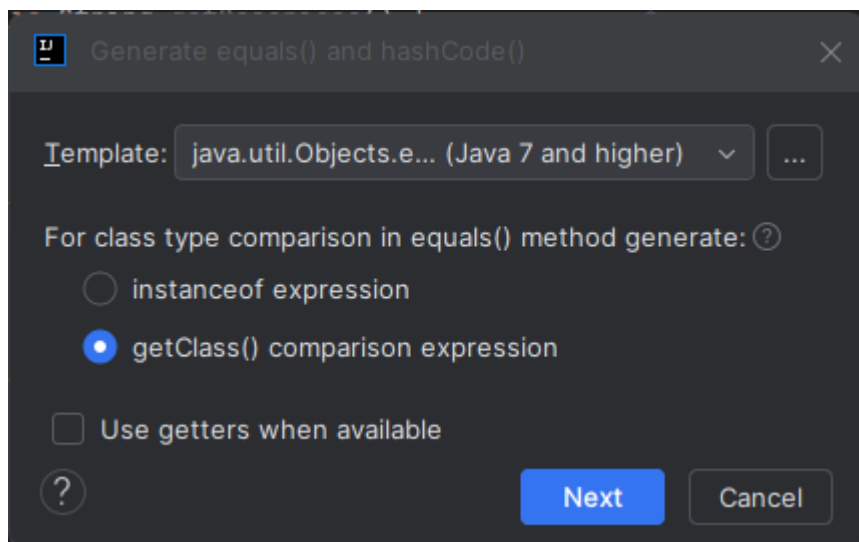
```
1 package com.curso.domains;
2
3 import com.curso.domains.enums.Status;
4
5 import java.util.Objects;
6
7 public class GrupoProduto { new *
8
9     private int id; 4 usages
10    private String descricao; 4 usages
11    private Status status; 3 usages
12
13    public GrupoProduto() { no usages new *
14    }
15
16    public GrupoProduto(int id, String descricao, Status status) {
17        this.id = id;
18        this.descricao = descricao;
19        this.status = status;
20    }
21
22    public int getId() { no usages new *
23        return id;
24    }
25
26    public void setId(int id) { no usages new *
27        this.id = id;
28    }
29
30    public String getDescricao() { no usages new *
31        return descricao;
32    }
33 }
```

- Como fizemos anteriormente criar através do **Generate**, os métodos **construtores** e os métodos **Getter e Setter**.

```
34     public void setDescricao(String descricao) { no usage
35         this.descricao = descricao;
36     }
37
38     public Status getStatus() { no usages new *
39         return status;
40     }
41
42     public void setStatus(Status status) { no usages new
43         this.status = status;
44     }
45
46 }
```

- Como fizemos anteriormente criar através do **Generate**, os métodos **construtores** e os métodos **Getter e Setter**.

- Gere também os métodos Equals e Hashcode através do método Generate.



```
@Override new *
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    GrupoProduto that = (GrupoProduto) o;
    return id == that.id && Objects.equals(descricao, that.descricao);
}

@Override new *
public int hashCode() {
    return Objects.hash(id, descricao);
}
```



Spring Boot

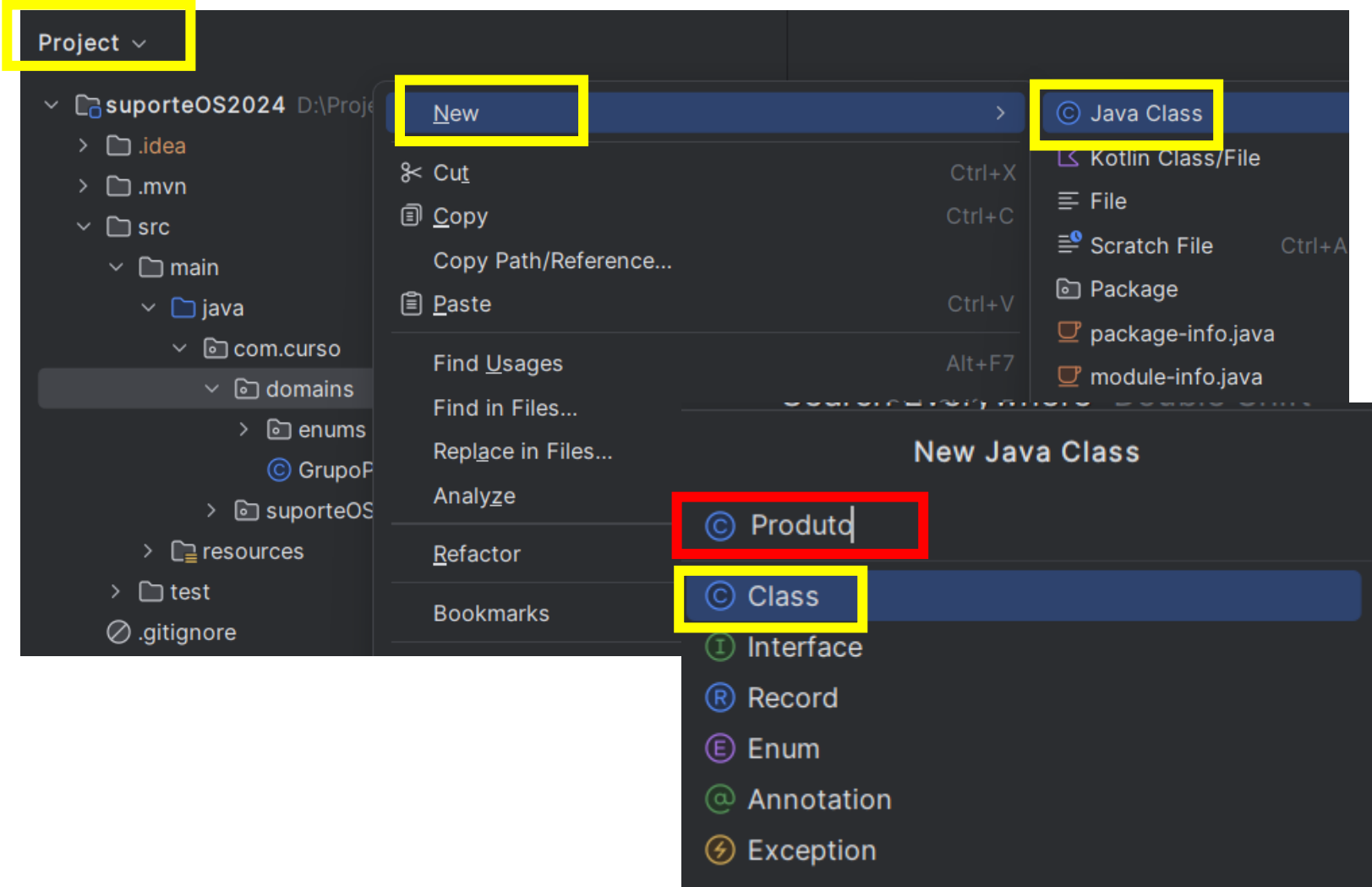
Criando as Entidades (Entities)

Produto

Criando Projeto Spring Boot

- Vamos criar nossas entidades do projeto.
- Para isso clique com o botão direito no pacote **domains** e escolha **New → Java Class**

Na janela "New Java Class" escolha a **opção Class** e informe o nome da classe a ser criada: **Produto**.



© Produto.java x

```
1 package com.curso.domains;
2
3 import com.curso.domains.enums.Status;
4
5 import java.time.LocalDate;
6
7 public class Produto { no usages new *
8
9     private long idProduto; no usages
10    private String descricao; no usages
11    private double saldoEstoque; no usages
12    private double valorUnitario; no usages
13    private LocalDate dataCadastro; no usages
14    private Status status; no usages
15
16 }
17
```

- Na classe Produto gerada insira os atributos da classe;
- Para o atributo status devemos fazer a importação do pacote enums.

```
1 package com.curso.domains;
2
3 import com.curso.domains.enums.Status;
4
5 import java.time.LocalDate;
6
7 public class Produto { no usages new *
8
9     private long idProduto; 3 usages
10    private String descricao; 3 usages
11    private double saldoEstoque; 3 usages
12    private double valorUnitario; 3 usages
13    private LocalDate dataCadastro; 3 usages
14    private Status status; 3 usages
15
16    public Produto() { no usages new *
17    }
18
19    public Produto(long idProduto, String descricao, double saldoEstoque, no usages
20                   double valorUnitario, LocalDate dataCadastro, Status status) {
21        this.idProduto = idProduto;
22        this.descricao = descricao;
23        this.saldoEstoque = saldoEstoque;
24        this.valorUnitario = valorUnitario;
25        this.dataCadastro = dataCadastro;
26        this.status = status;
27    }
28
```

Como fizemos anteriormente criar
através do **Generate**, os métodos
Constructores e os métodos **Getter**
Setter.

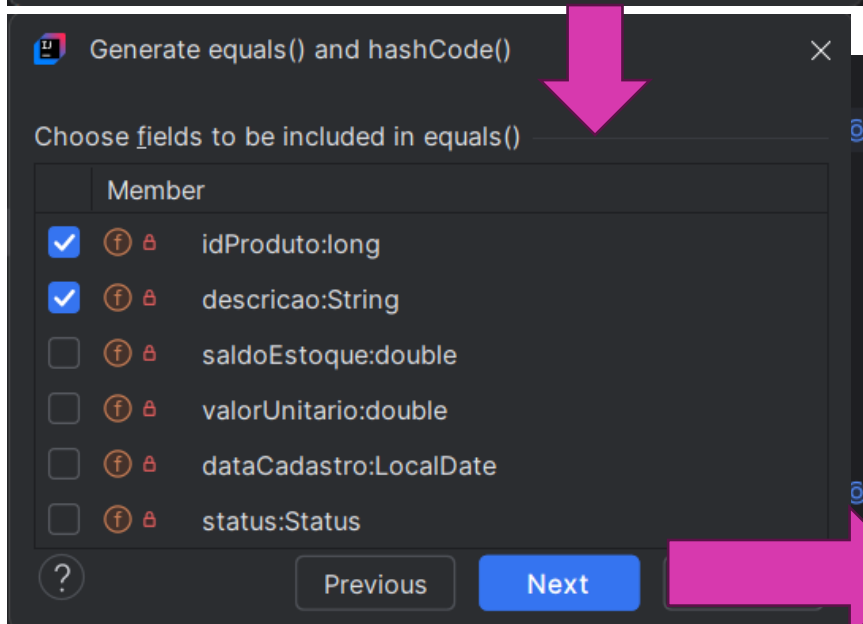
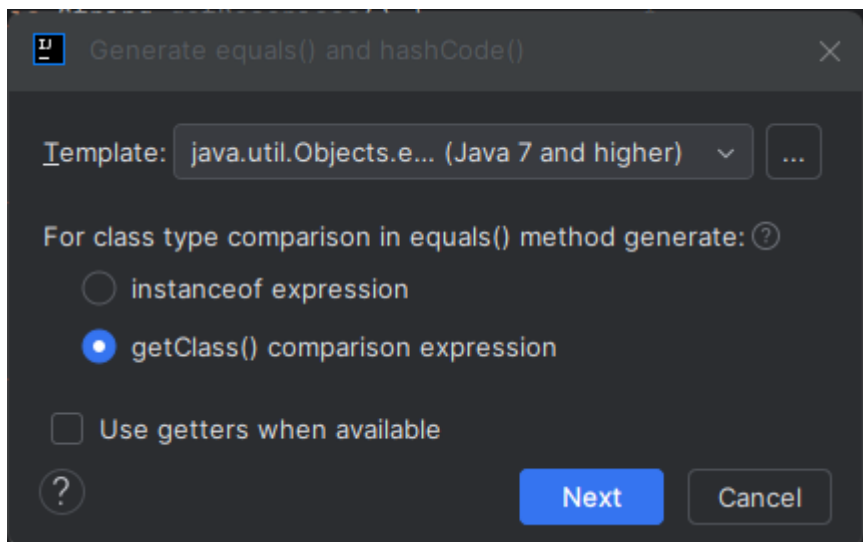
```
29 public long getIdProduto() { no usages new *
30     return idProduto;
31 }
32
33 public void setIdProduto(long idProduto) { no usages new *
34     this.idProduto = idProduto;
35 }
36
37 public String getDescricao() { no usages new *
38     return descricao;
39 }
40
41 public void setDescricao(String descricao) { no usages new *
42     this.descricao = descricao;
43 }
44
45 public double getSaldoEstoque() { no usages new *
46     return saldoEstoque;
47 }
48
49 public void setSaldoEstoque(double saldoEstoque) { no usages new *
50     this.saldoEstoque = saldoEstoque;
51 }
52
53 public double getValorUnitario() { no usages new *
54     return valorUnitario;
55 }
56
```

- Como fizemos anteriormente criamos através do **Generate**, os métodos **construtores** e os métodos **Getter e Setter**.

```
57 public void setValorUnitario(double valorUnitario) { no us
58     this.valorUnitario = valorUnitario;
59 }
60
61 public LocalDate getDataCadastro() { no usages new *
62     return dataCadastro;
63 }
64
65 public void setDataCadastro(LocalDate dataCadastro) { no u
66     this.dataCadastro = dataCadastro;
67 }
68
69 public Status getStatus() { no usages new *
70     return status;
71 }
72
73 public void setStatus(Status status) { no usages new *
74     this.status = status;
75 }
76 }
```

- Como fizemos anteriormente criar através do **Generate**, os métodos **construtores** e os métodos **Getter e Setter**.

- Gere também os métodos Equals e Hashcode através do método Generate.



```
@Override new *
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Produto produto = (Produto) o;
    return idProduto == produto.idProduto && Objects.equals(descricao, produto.descricao);
}

@Override new *
public int hashCode() {
    return Objects.hash(idProduto, descricao);
}
```