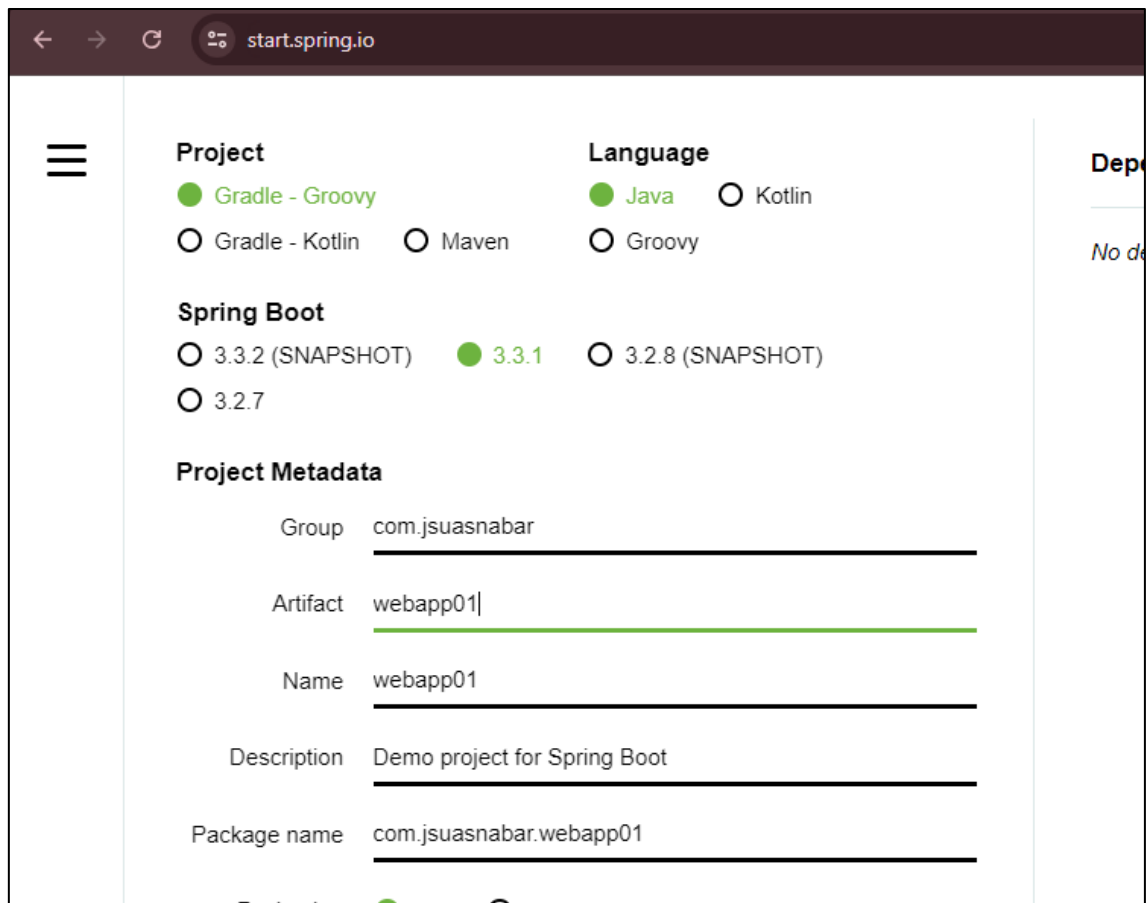


## SEMANA 15

SPRING <https://start.spring.io/>

Configuración de aspectos generales



The screenshot shows the Spring Start web application interface. The browser address bar displays 'start.spring.io'. The main content area is divided into sections for project configuration:

- Project:** Includes radio buttons for 'Gradle - Groovy' (selected), 'Gradle - Kotlin', and 'Maven'.
- Language:** Includes radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'.
- Spring Boot:** Includes radio buttons for '3.3.2 (SNAPSHOT)', '3.3.1' (selected), '3.2.8 (SNAPSHOT)', and '3.2.7'.
- Project Metadata:** A form with the following fields:
  - Group: com.jsuasnabar
  - Artifact: webapp01|
  - Name: webapp01
  - Description: Demo project for Spring Boot
  - Package name: com.jsuasnabar.webapp01

Name: nombre del proyecto

Artifact : donde encontraremos el proyecto

Spring Boot

☐ 3.3.2 (SNAPSHOT)

☒ 3.3.1

☐ 3.2.8 (SNAPSHOT)

☐ 3.2.7

Project Metadata

Group

com.jsuasnabar

Artifact

webapp01

Name

webapp01

Description

Primera aplicacion web con spring

Package name

com.jsuasnabar.webapp01

Packaging

☒ Jar

☐ War

Java

☒ 22

☐ 21

☐ 17

Click

Project

☒ Gradle - Groovy

☐ Gradle - Kotlin

☐ Maven

Language

☒ Java

☐ Kotlin

☐ Groovy

Dependencies

ADD DEPENDENCIES... CTRL + B

No dependency selected

Spring Boot

Project

☐ Gradle - Groovy

☐ Gradle - Kotlin

☒ Maven

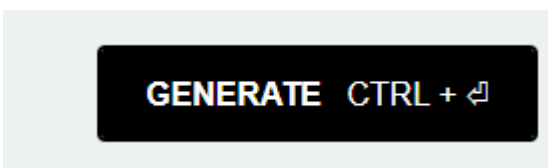
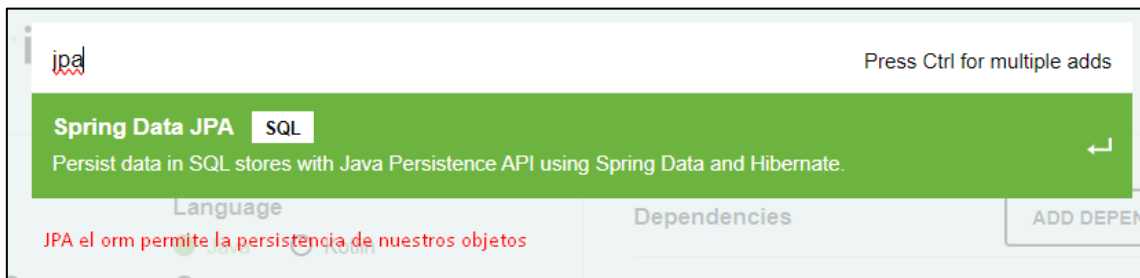
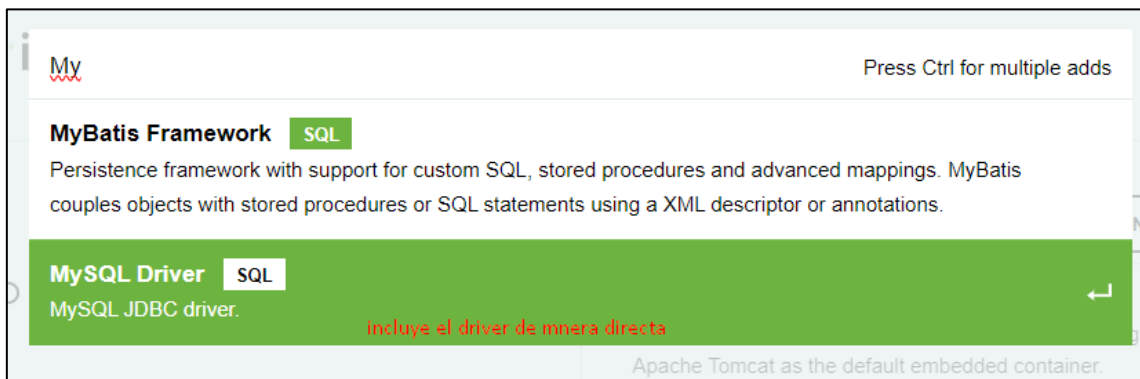
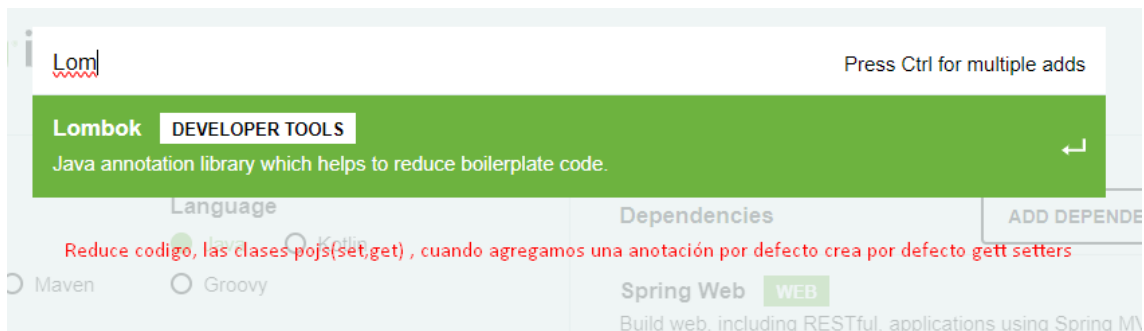
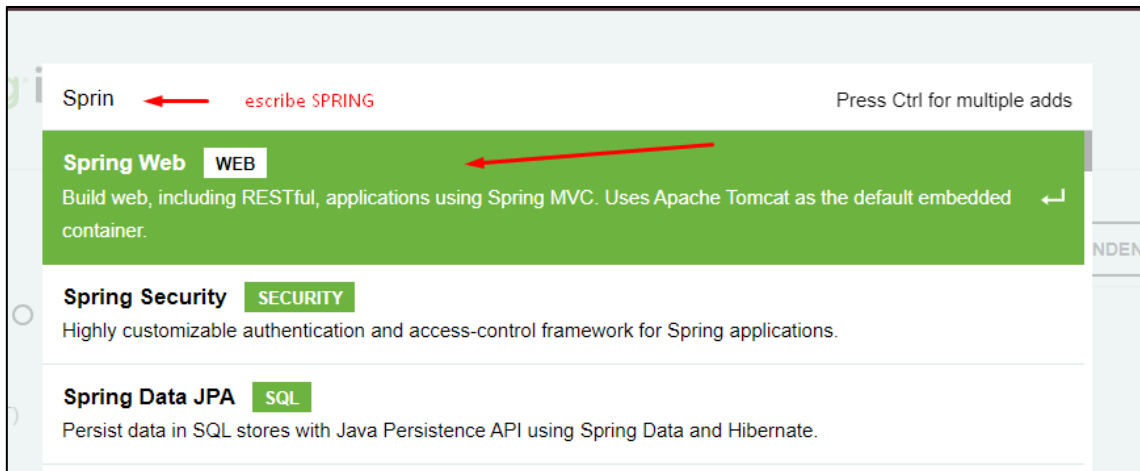
Language

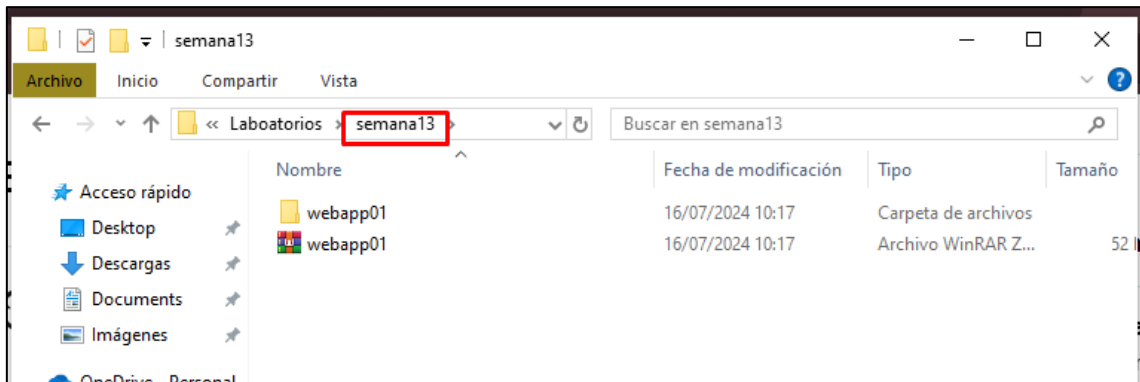
☒ Java

☐ Kotlin

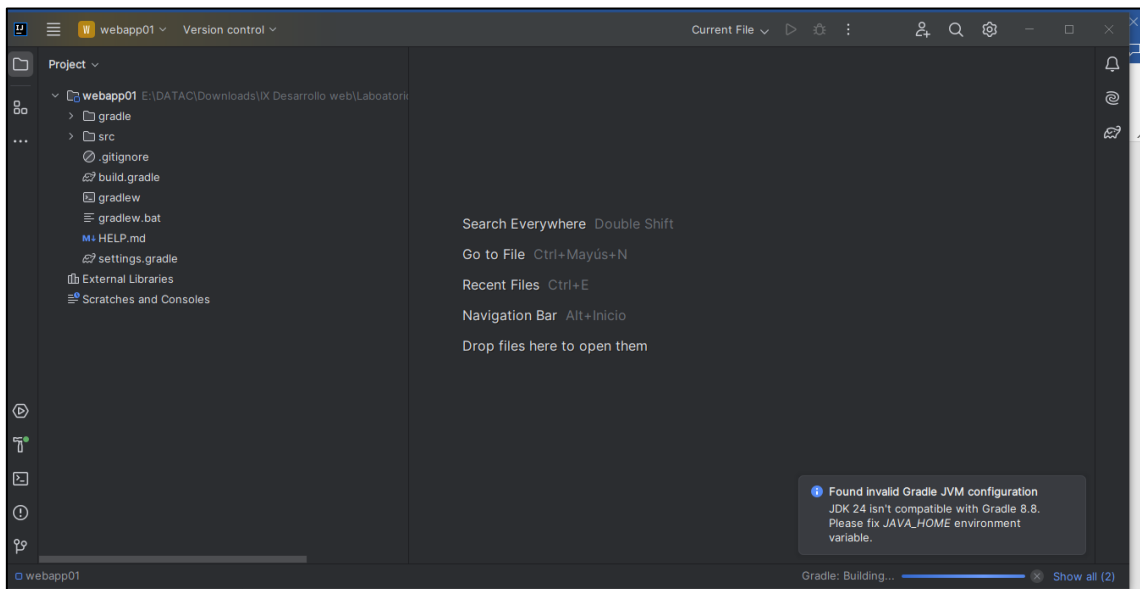
☐ Groovy

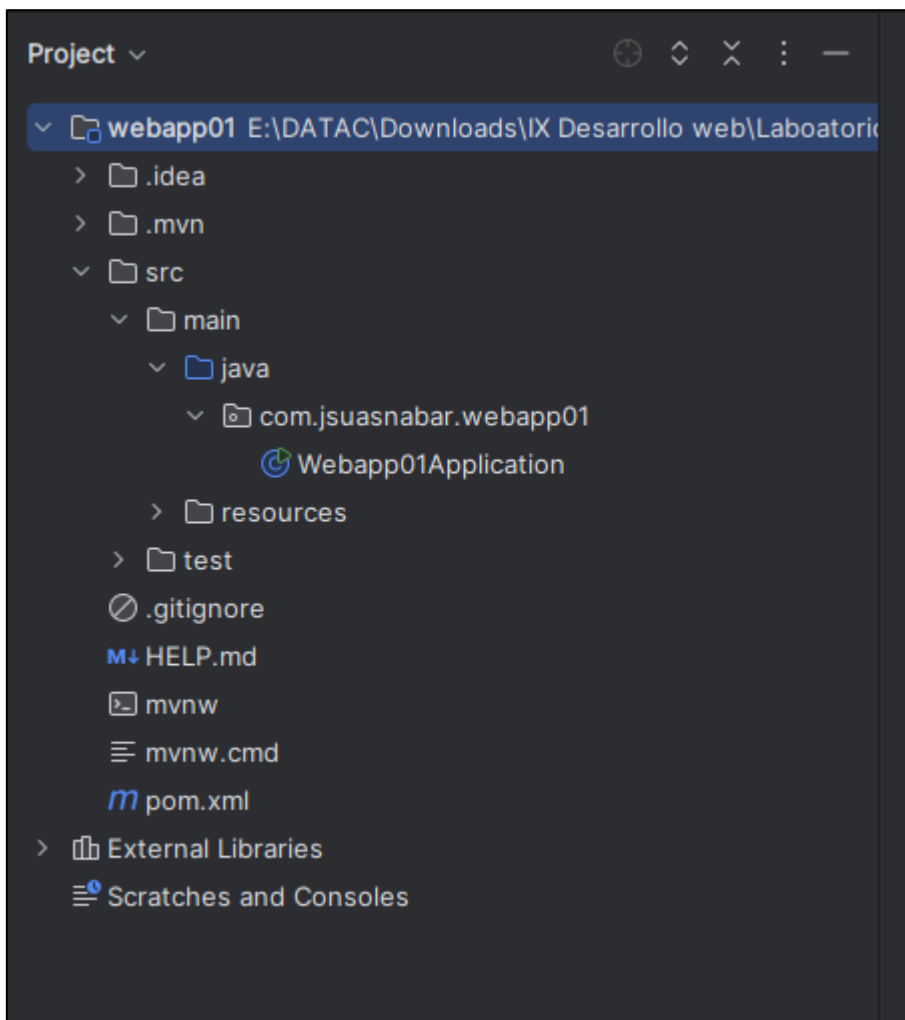
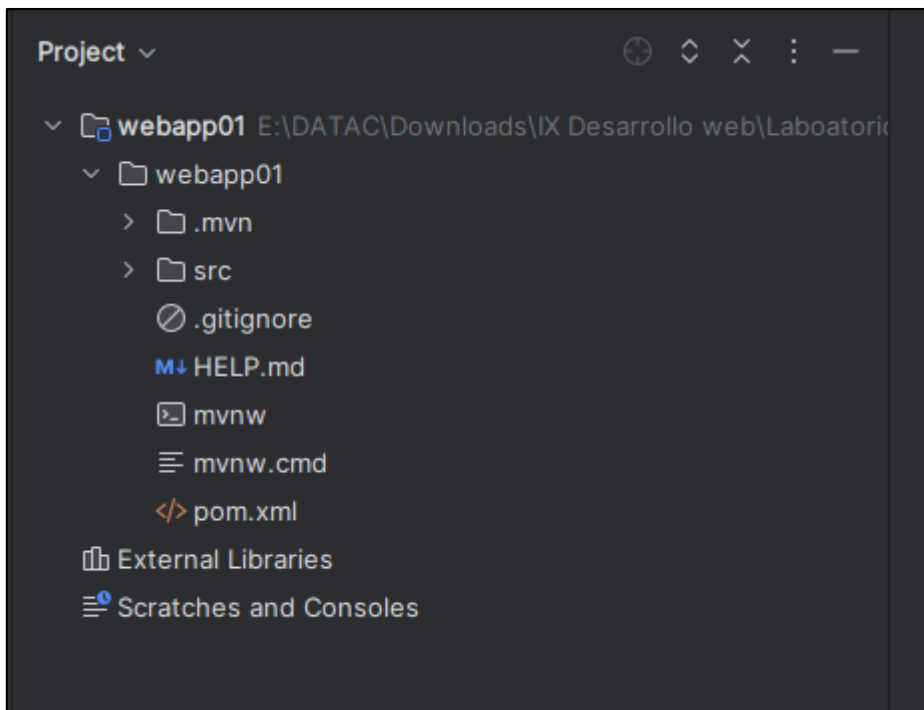
Spring Boot

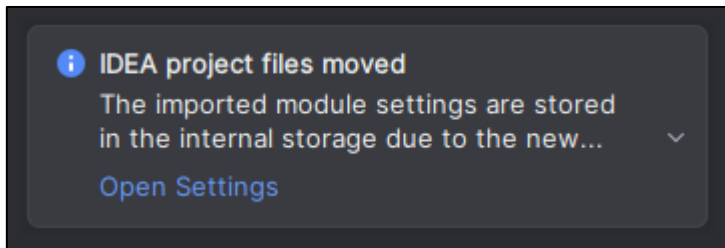




Ahora nos dirigimos en IntelliJ y abrimos





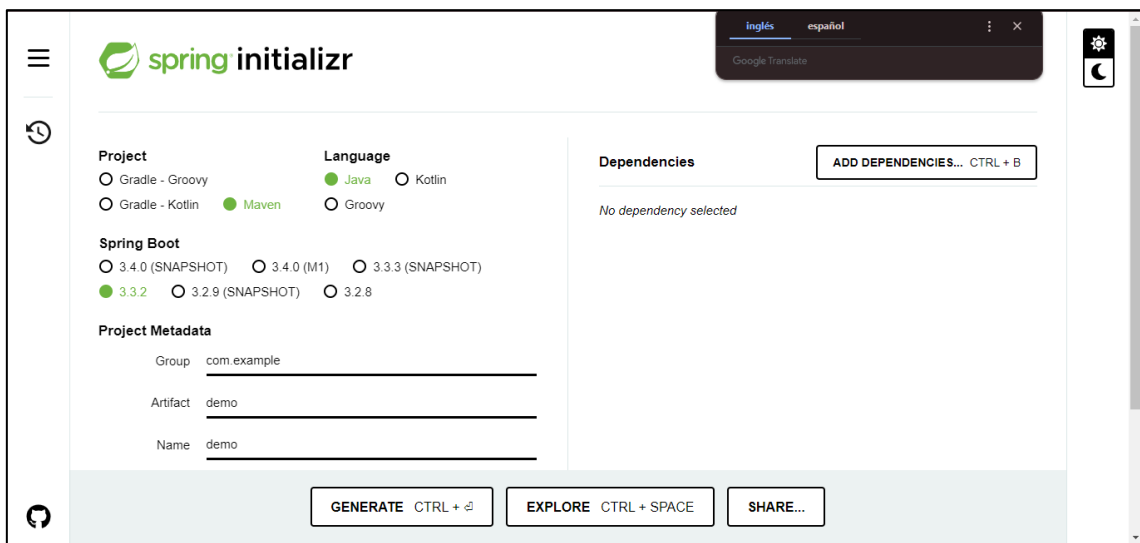


Tenemos que cambiar todo a jdk 22

SPRING BOOT definición

## VIERNES

Ingresar a Sprint boot inicializa



### Project

☐ Gradle - Groovy

☐ Gradle - Kotlin

☒ Maven

### Language

☒ Java

☐ Kotlin

☐ Groovy

### Spring Boot

☐ 3.4.0 (SNAPSHOT)

☐ 3.4.0 (M1)

☐ 3.3.3 (SNAPSHOT)

☒ 3.3.2

☐ 3.2.9 (SNAPSHOT)

☐ 3.2.8

### Project Metadata

Group

Artifact

Name

Description

Description

Package name

Packaging ☒ Jar ☐ War

Java ☒ 22 ☐ 21 ☐ 17

Java 11 no puede utilizar con este spring

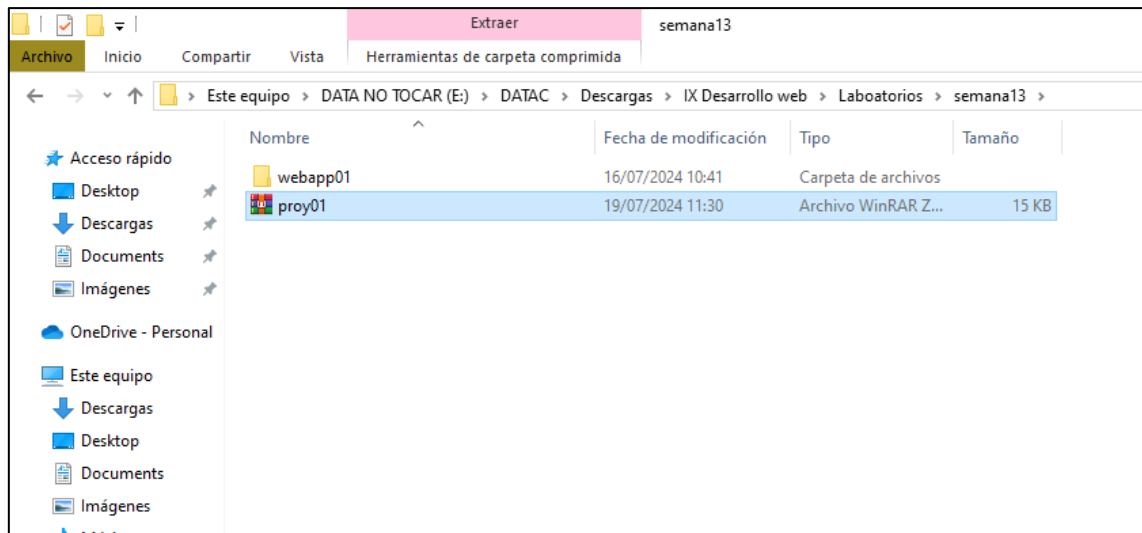
### Dependencies

**ADD DEPENDENCIES... CTRL + B**

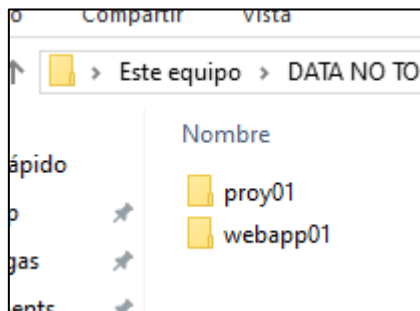
### Spring Web **WEB**

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

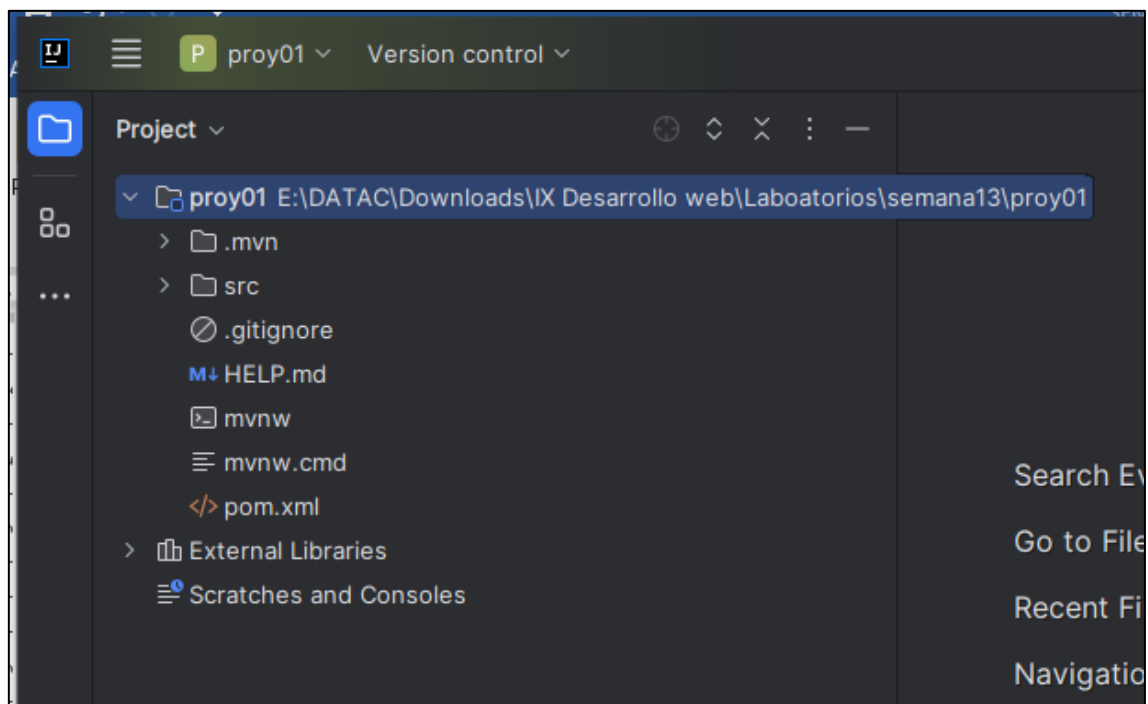
## DESCARGAR



## COPIAR EN SEMANA 13 Y EXTRAER

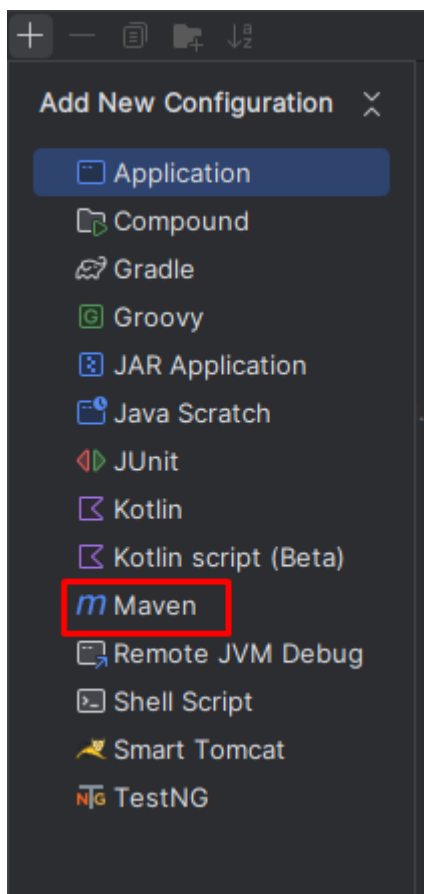
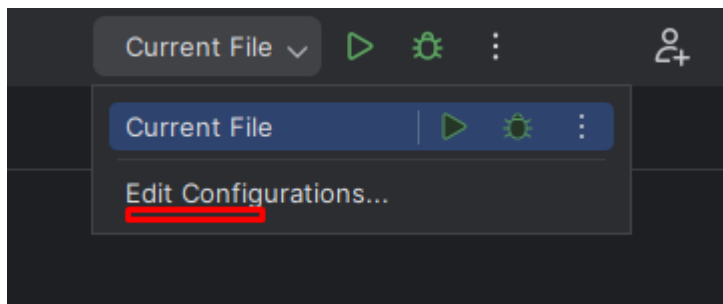
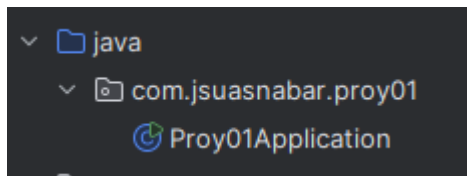
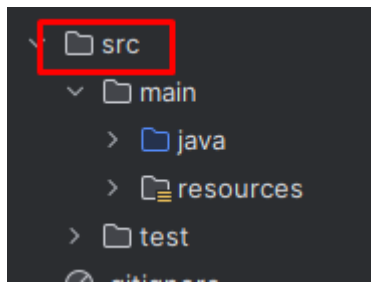


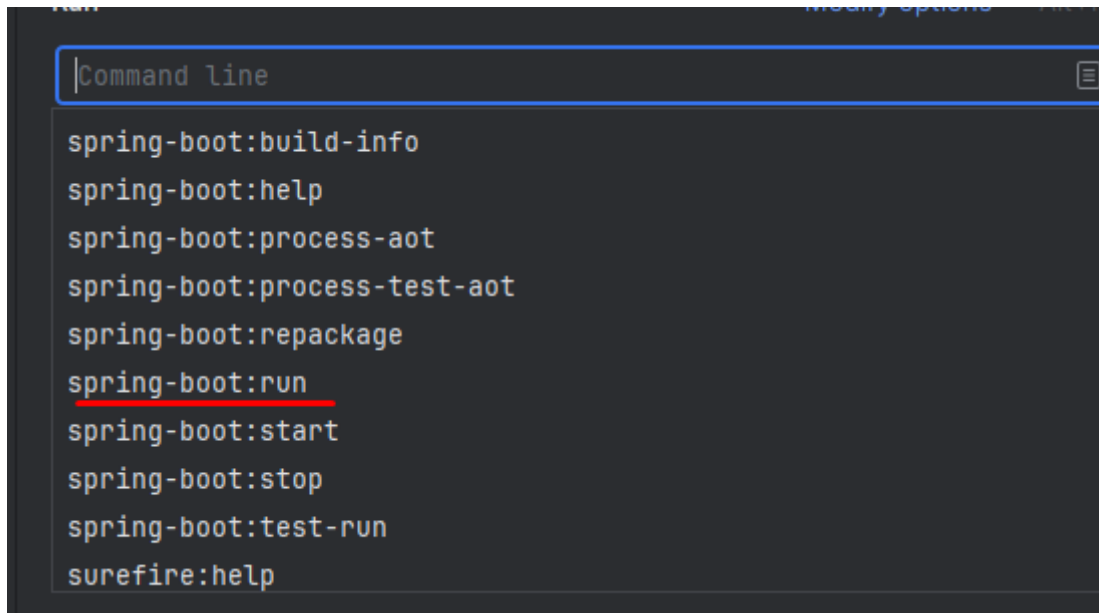
## Ingresar al proyecto por inteligen



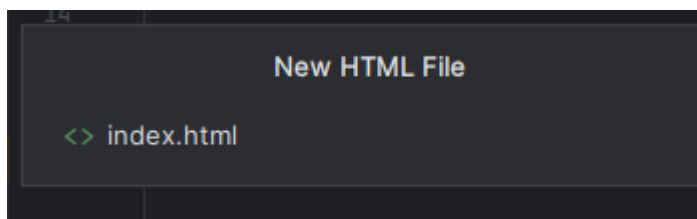
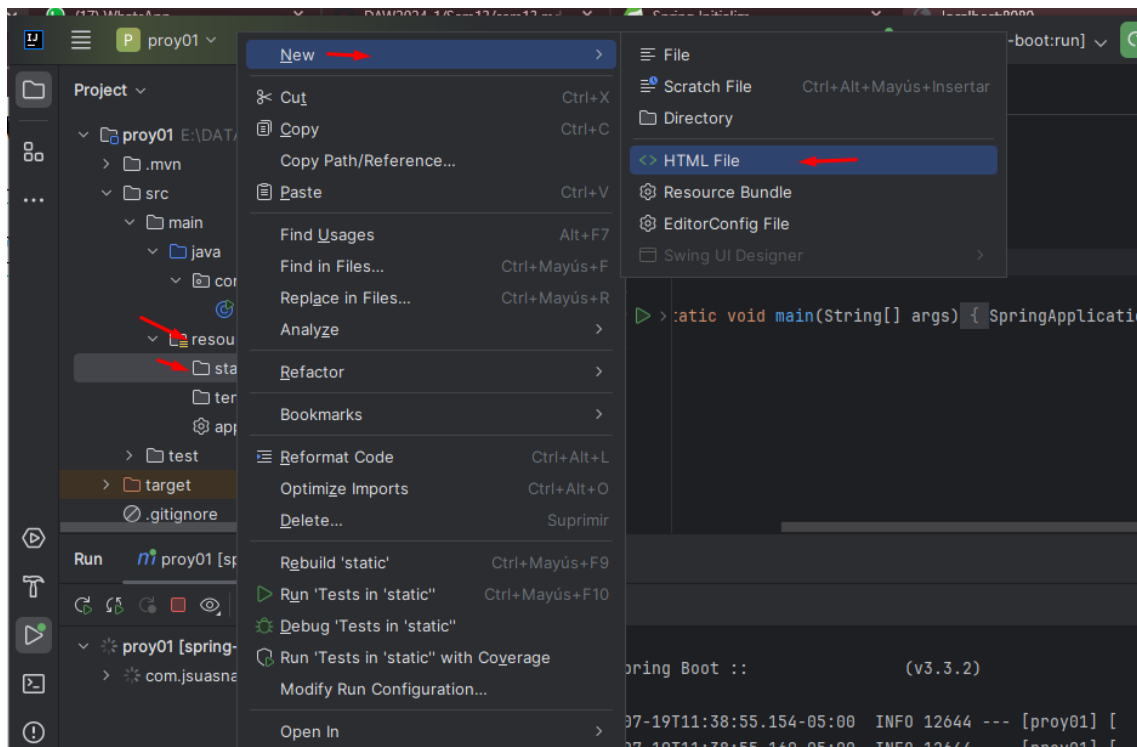
todo dentro de src trabajaremos







okey

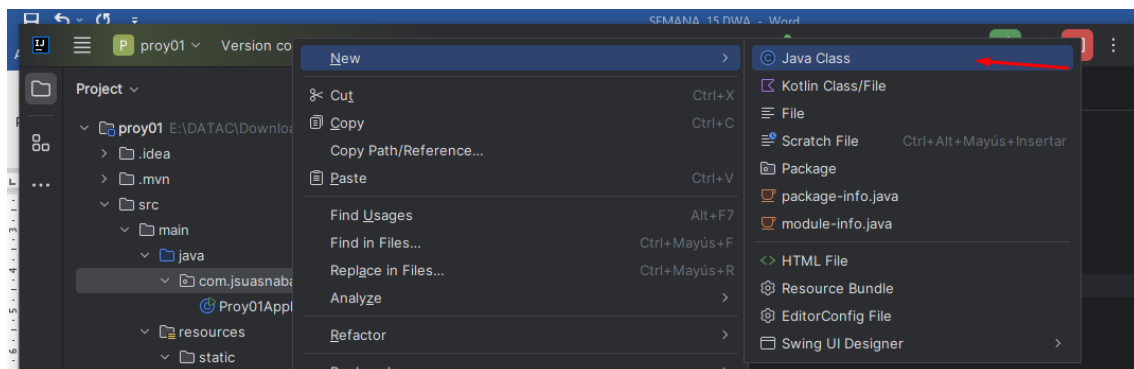
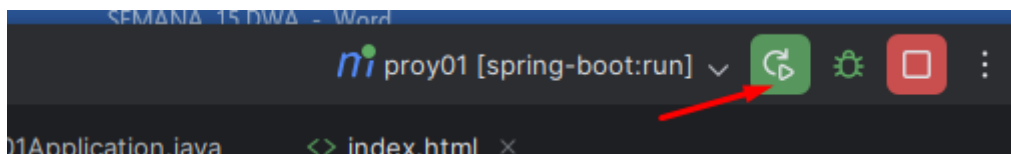


entre

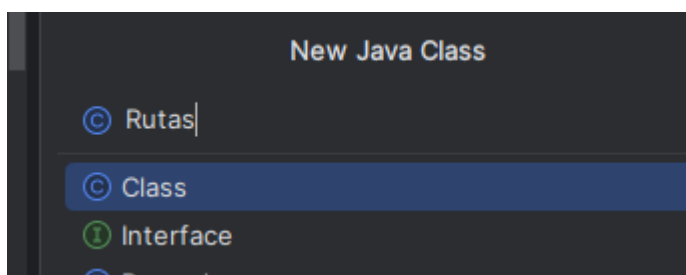
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Title</title>
6 </head>
7 <body>
8   <h1>Jakacrta sping</h1>
9
10 </body>
11 </html>
```

html > body > h1

Localhost 8080 volver hacer run



Enter



Debemos manejar protocolos http

**Que es:** se ejecuta en la capa de aplicación

Métodos get, post(por el form), put , delete

Status http:

- 1xx códigos informativos
- 2xx códigos de éxito
- 3xx códigos de redirección
- 4xx códigos de error de cliente
- 5xx códigos de error de servicio

## SERVICIO DE API Y REST API

**SERVICIO DE API:** conjunto de protocolos y definiciones que pueden permitir que una aplicación se comunique

## ENDPOINTS + API REST

**CRUD:** ( crear , read update y delete )

**ENDPOINT = METHOD HTTP + URL SEGMENT + QUERY PARAMS**

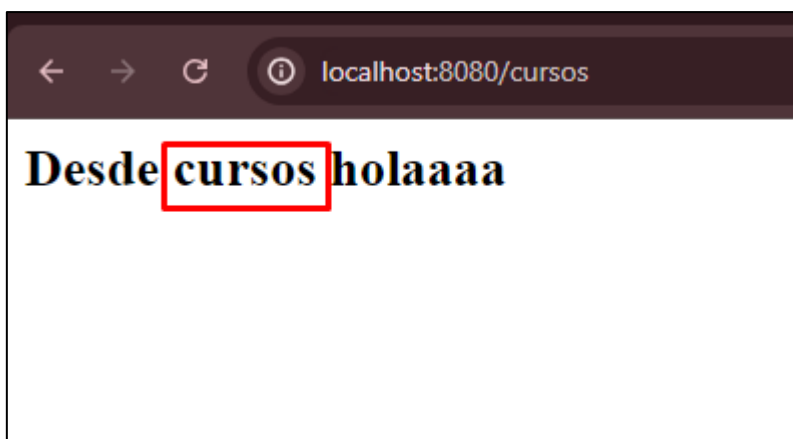
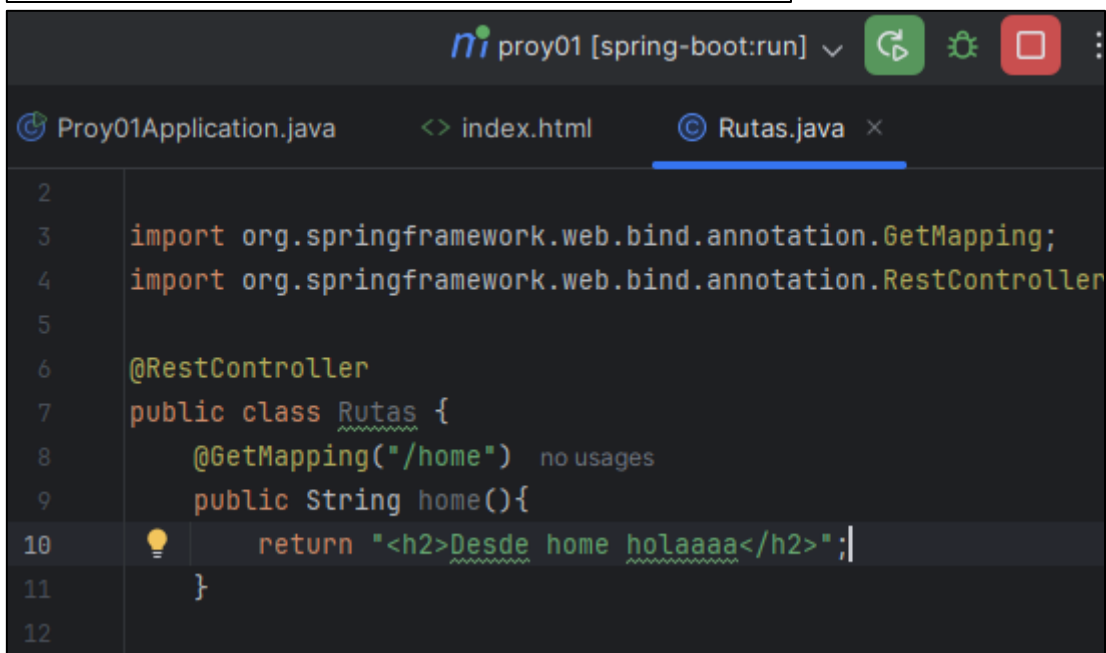
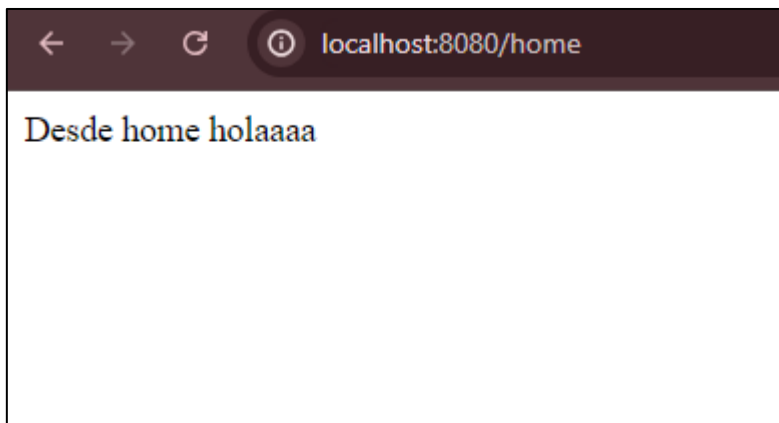
## DESARROLLO DE API REST- JAKARTA SPRING

Controlador , service ,model

**Controladores:** usaremos anotaciones: @GetMapping, y @RequestMapping  
inyección de dependencias.



```
Proy01Application.java  index.html  Rutas.java x
2
3  import org.springframework.web.bind.annotation.GetMapping;
4  import org.springframework.web.bind.annotation.RestController;
5
6  @RestController  no usages
7  public class Rutas {
8      @GetMapping("/home")  no usages
9      public String home(){
10         return "Desde home holaaaaa";
11     }
12
13 }
14
```



```

@GetMapping(path="/estu/{id}") no usages
public String estu(){
    return "<h2>Desde estu holaaaa</h2>";
}
}

```

el parametro se recibe ahii

```

@GetMapping(path="/estu/{id}") no usages
public String estu(@PathVariable int id){
    return "<h2>Desde estu holaaaa</h2>";
}
}

```

spring inyecta

```

@GetMapping("/estu/{id}") no usages
public String estu(@PathVariable int id){
    return "<h2>Desde estu holaaaa: </h2>"+id;
}

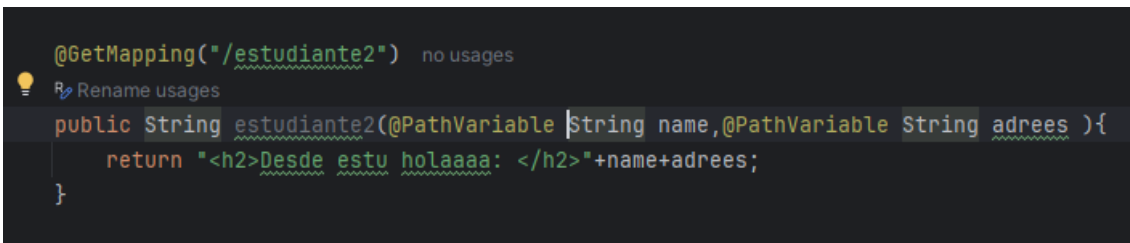
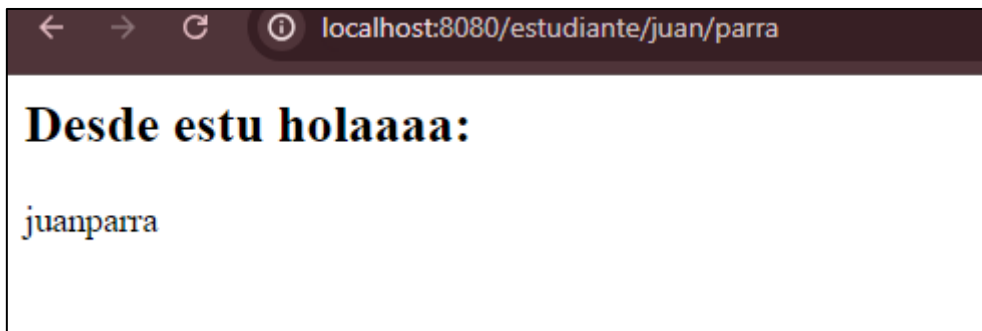
```



```

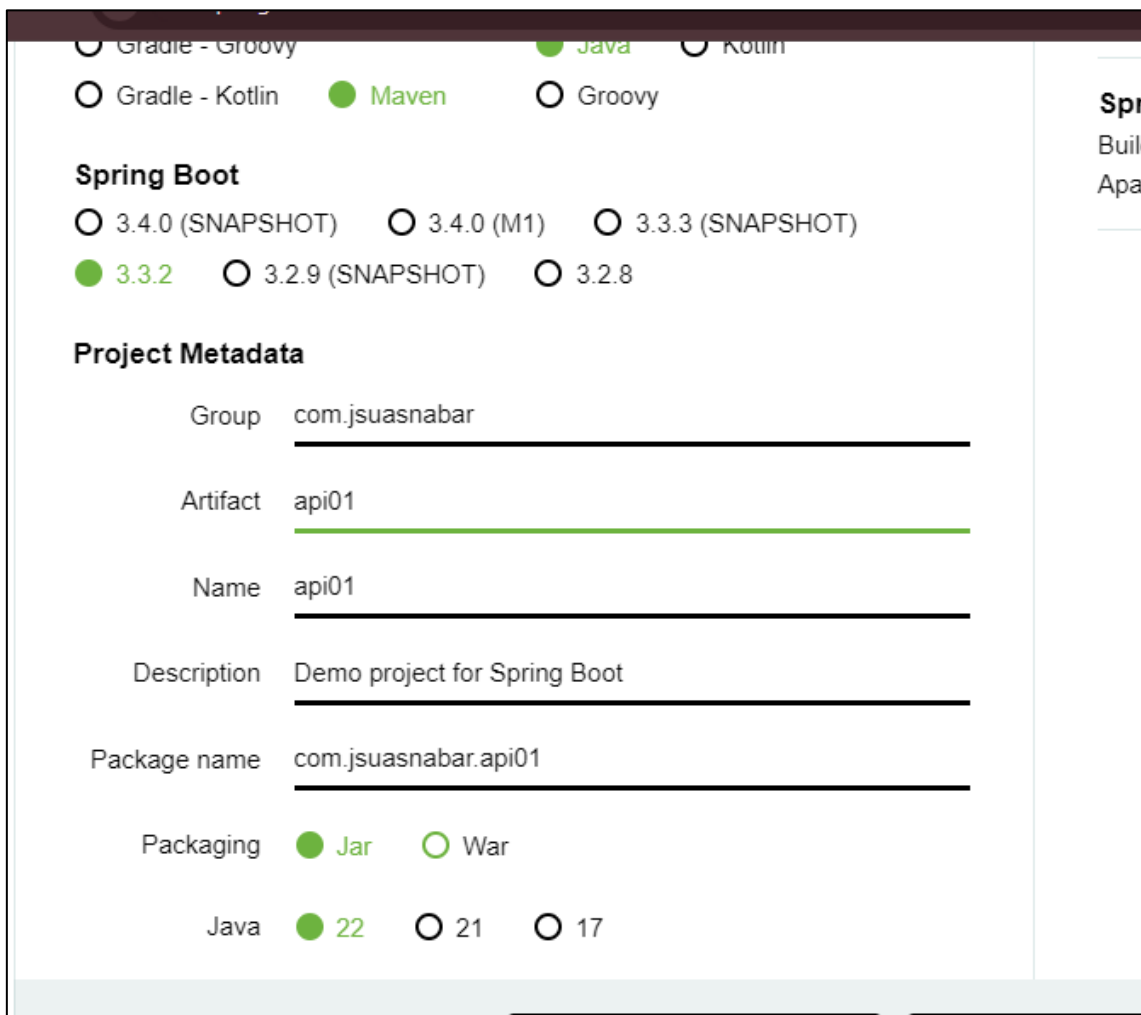
@GetMapping("/estudiante/{fname}/{lname}") no usages
public String estudiante(@PathVariable String fname,@PathVariable String lname ){
    return "<h2>Desde estu holaaaa: </h2>"+fname+lname;
}

```



Request body: recibe parámetros ocultos dentro de un formulario

BACKEND vamos aprender cómo funciona JPA



Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web

WEB

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

—

MySQL Driver

SQL

MySQL JDBC driver.

—

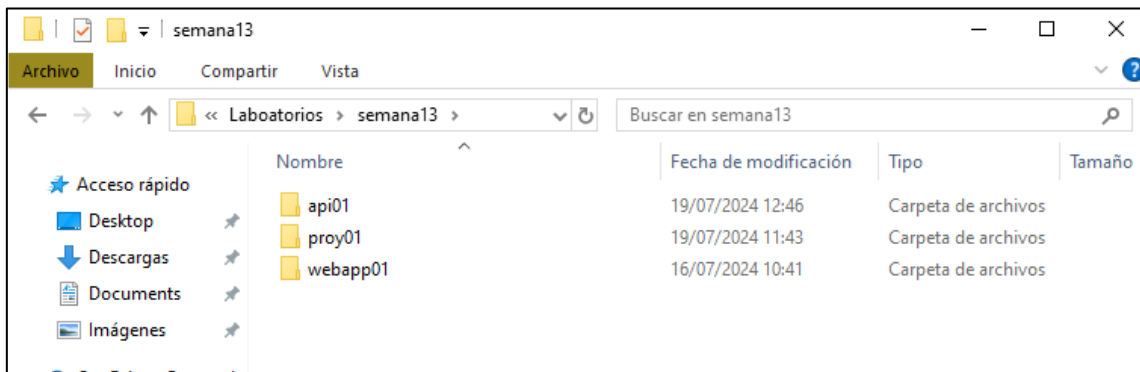
Spring Data JPA

SQL

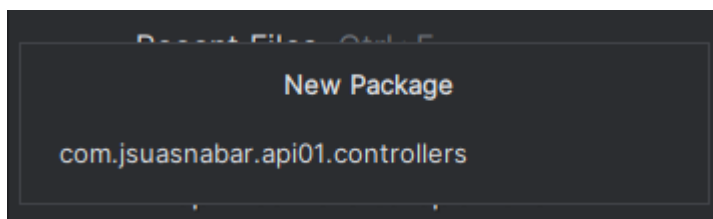
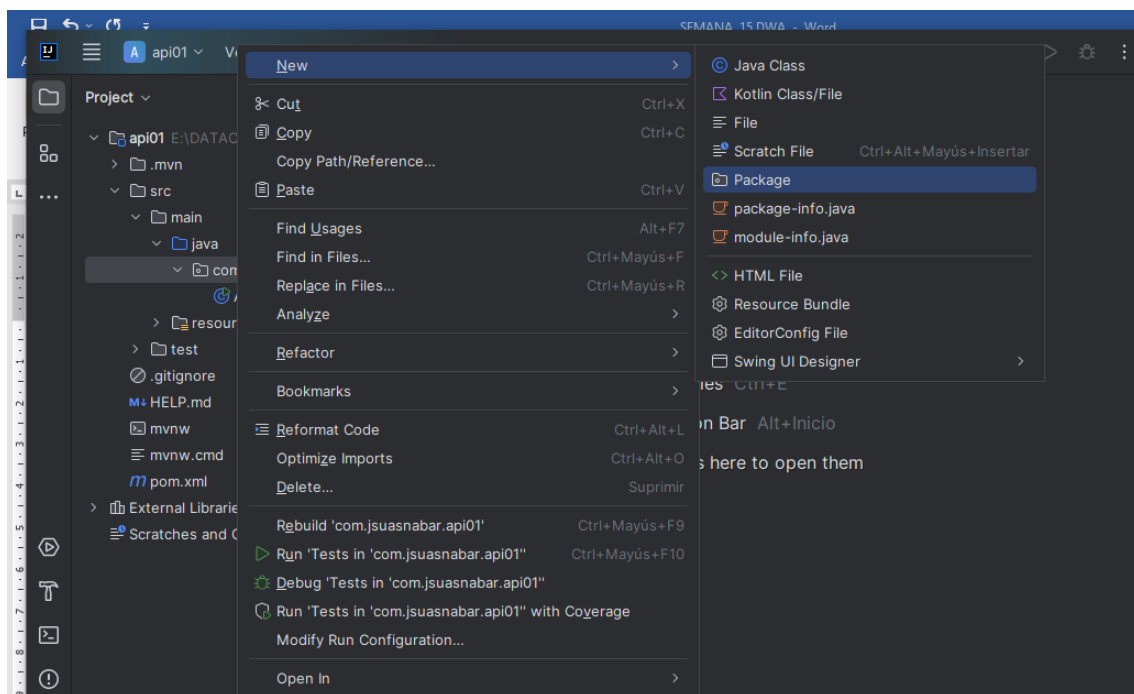
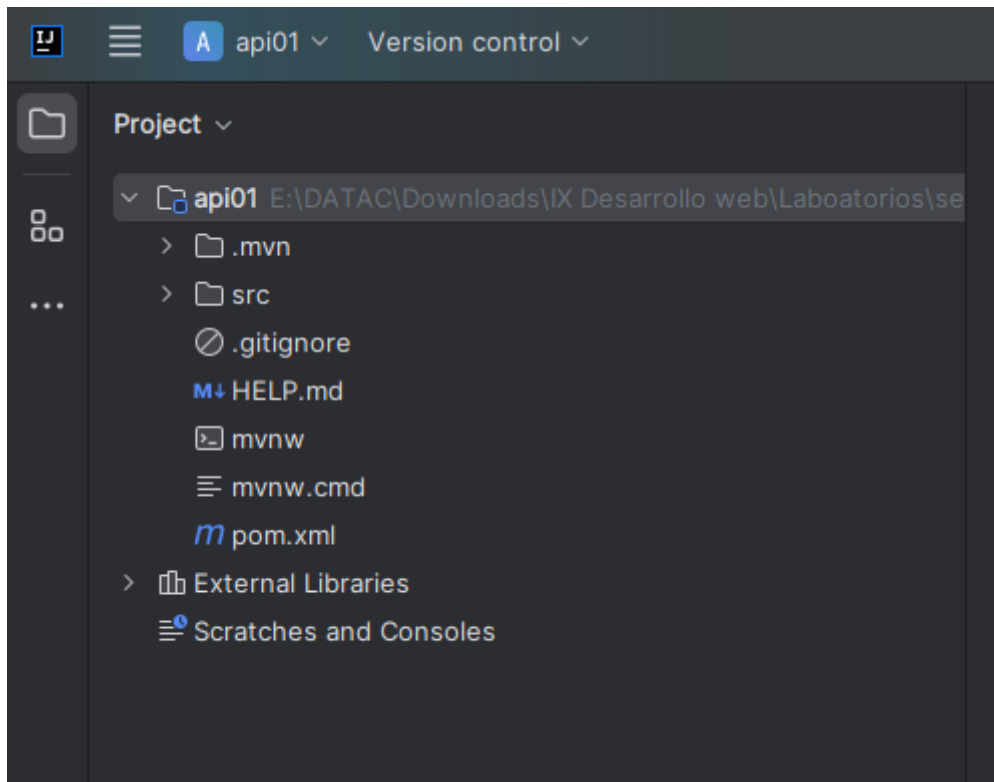
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

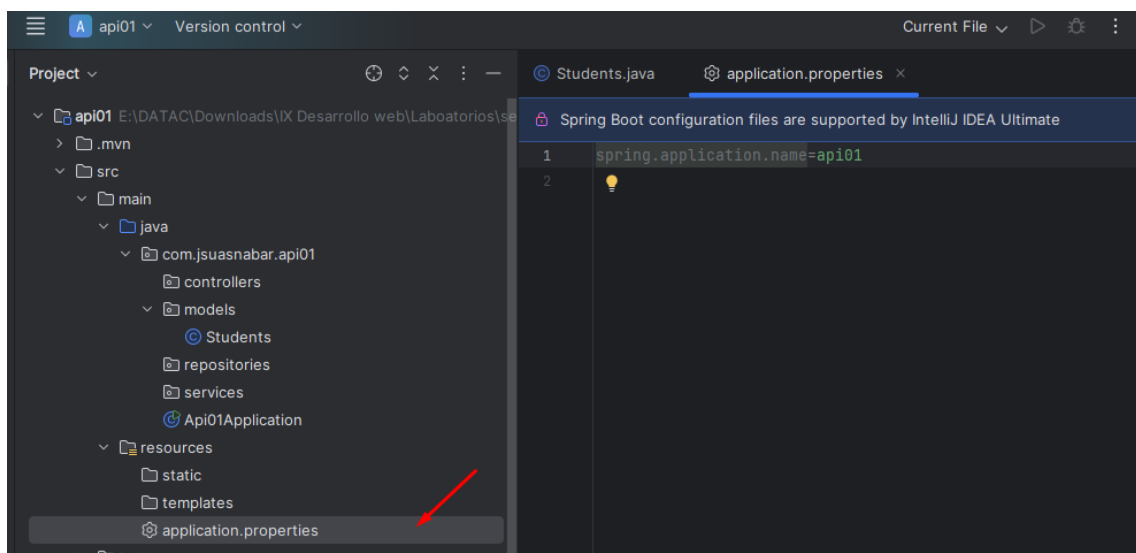
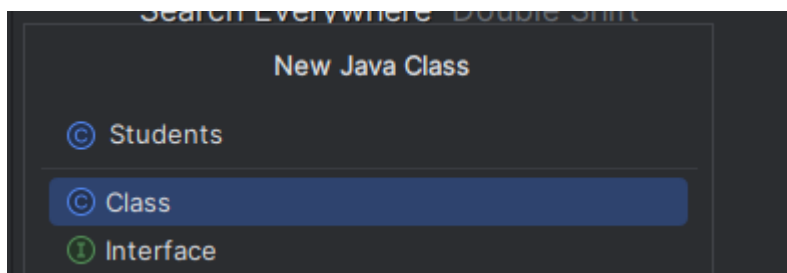
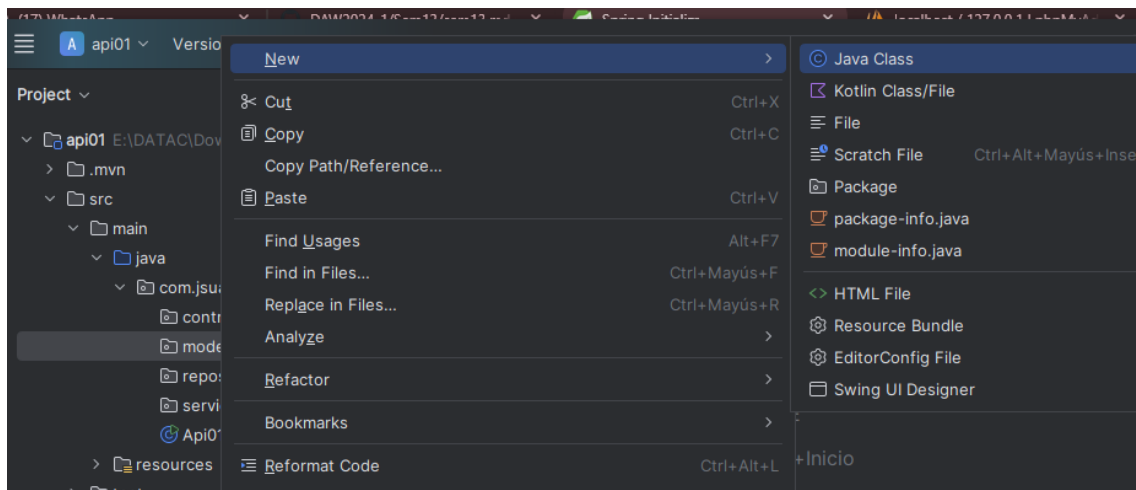
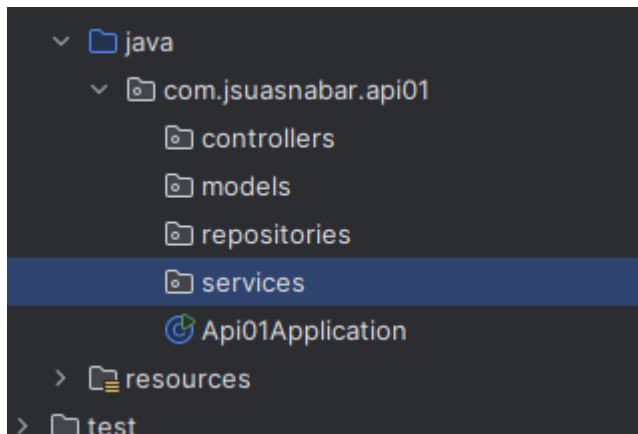
—

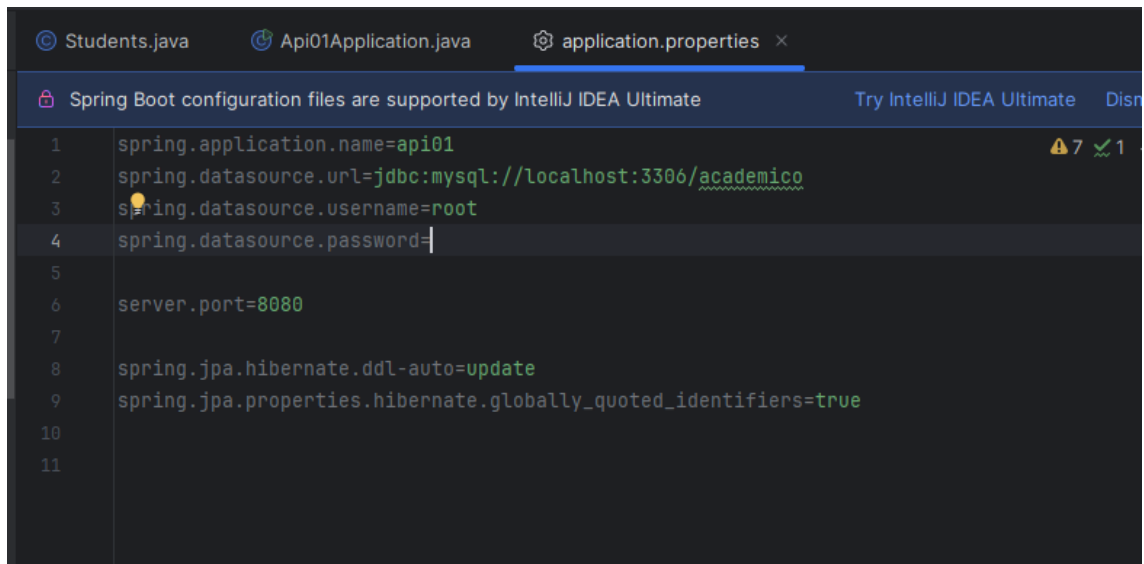
Generate



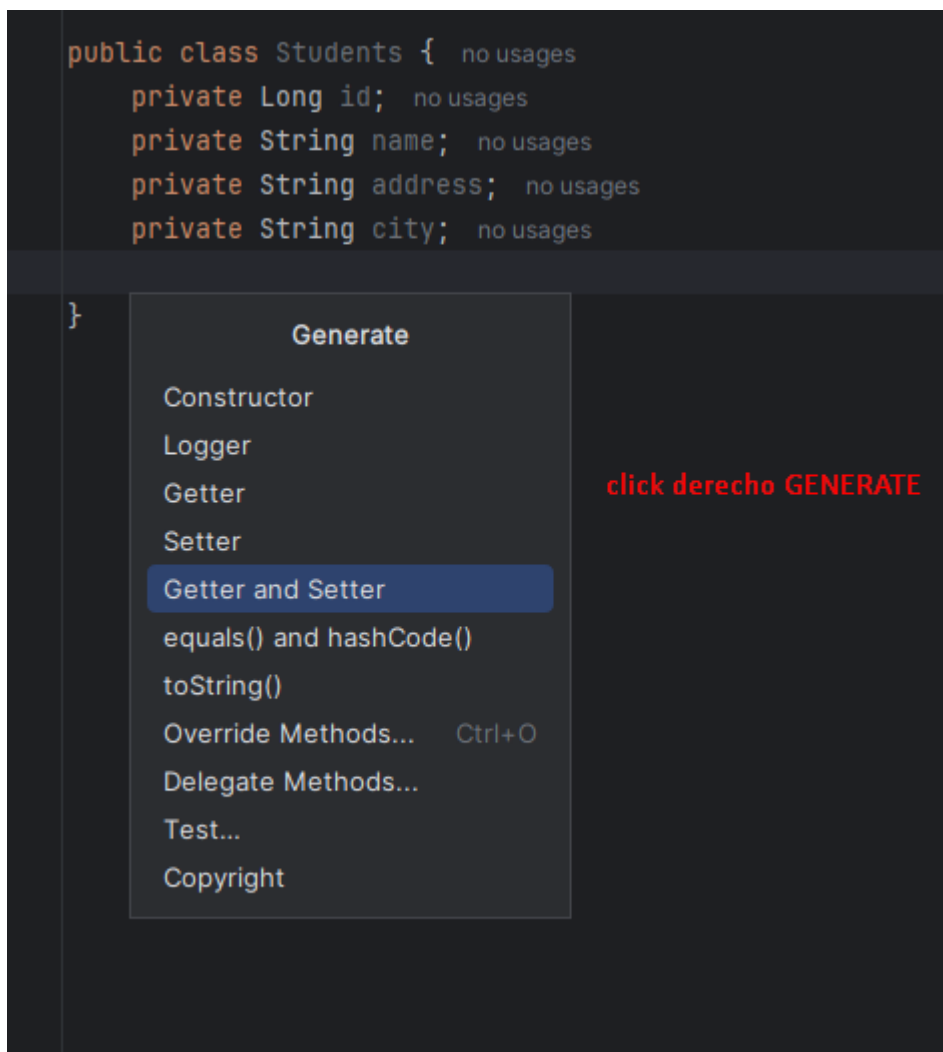








```
1 spring.application.name=api01
2 spring.datasource.url=jdbc:mysql://localhost:3306/academico
3 spring.datasource.username=root
4 spring.datasource.password=
5
6 server.port=8080
7
8 spring.jpa.hibernate.ddl-auto=update
9 spring.jpa.properties.hibernate.globally_quoted_identifiers=true
10
11
```



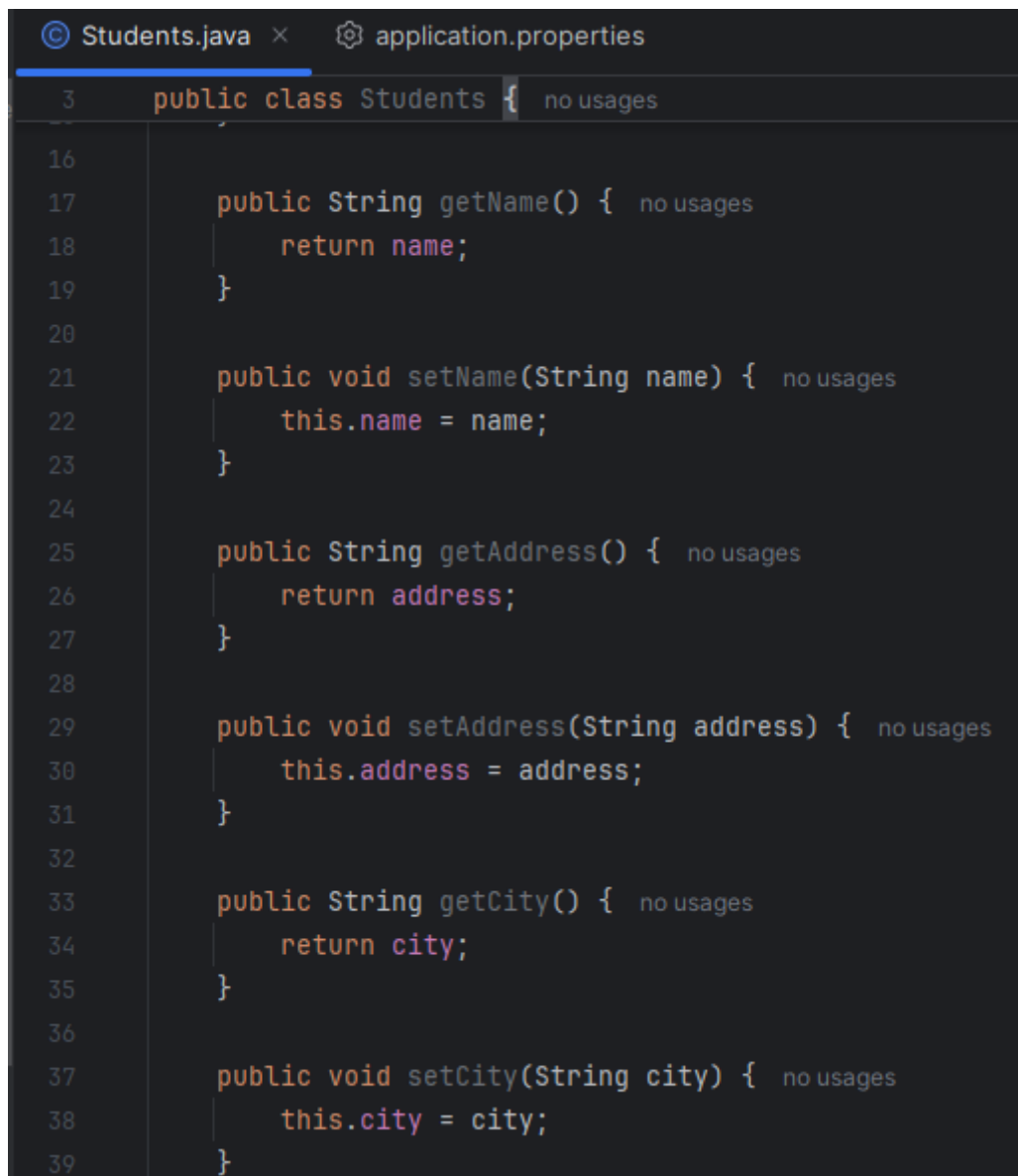
```
public class Students {
    private Long id;
    private String name;
    private String address;
    private String city;
}
```

**Generate**

- Constructor
- Logger
- Getter
- Setter
- Getter and Setter**
- equals() and hashCode()
- toString()
- Override Methods... Ctrl+O
- Delegate Methods...
- Test...
- Copyright

click derecho GENERATE

Clase poo



```
© Students.java × application.properties
3 public class Students { no usages
16
17     public String getName() { no usages
18         return name;
19     }
20
21     public void setName(String name) { no usages
22         this.name = name;
23     }
24
25     public String getAddress() { no usages
26         return address;
27     }
28
29     public void setAddress(String address) { no usages
30         this.address = address;
31     }
32
33     public String getCity() { no usages
34         return city;
35     }
36
37     public void setCity(String city) { no usages
38         this.city = city;
39     }
```

Con esto ya debe estar la bd agregando las 3 flechas

```

package com.jsuasnabar.api01.models;

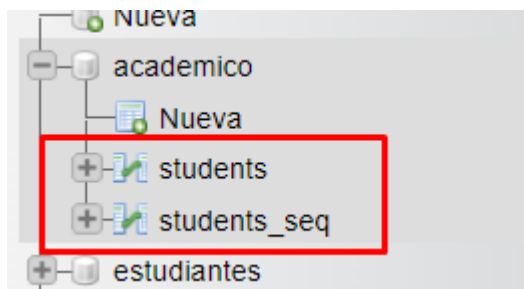
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Students {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String name;
    private String address;
    private String city;

    public Long getId() {
        return id;
    }
}

```

Ejecutar y automaticamente sin haber creado nada en la bd se crea lo siguiente



Académico es una bd que debemos crear sin ninguna tabla eso se agrega al ejecutar