



DuocUC[®] INFORMÁTICA Y
TELECOMUNICACIONES

■ SPRING BOOT: ESTRUCTURA DEL PROYECTO

DESARROLLO FULLSTACK I
DSY1103

En el capítulo anterior...

¿Qué son las **@anotaciones** en Spring?



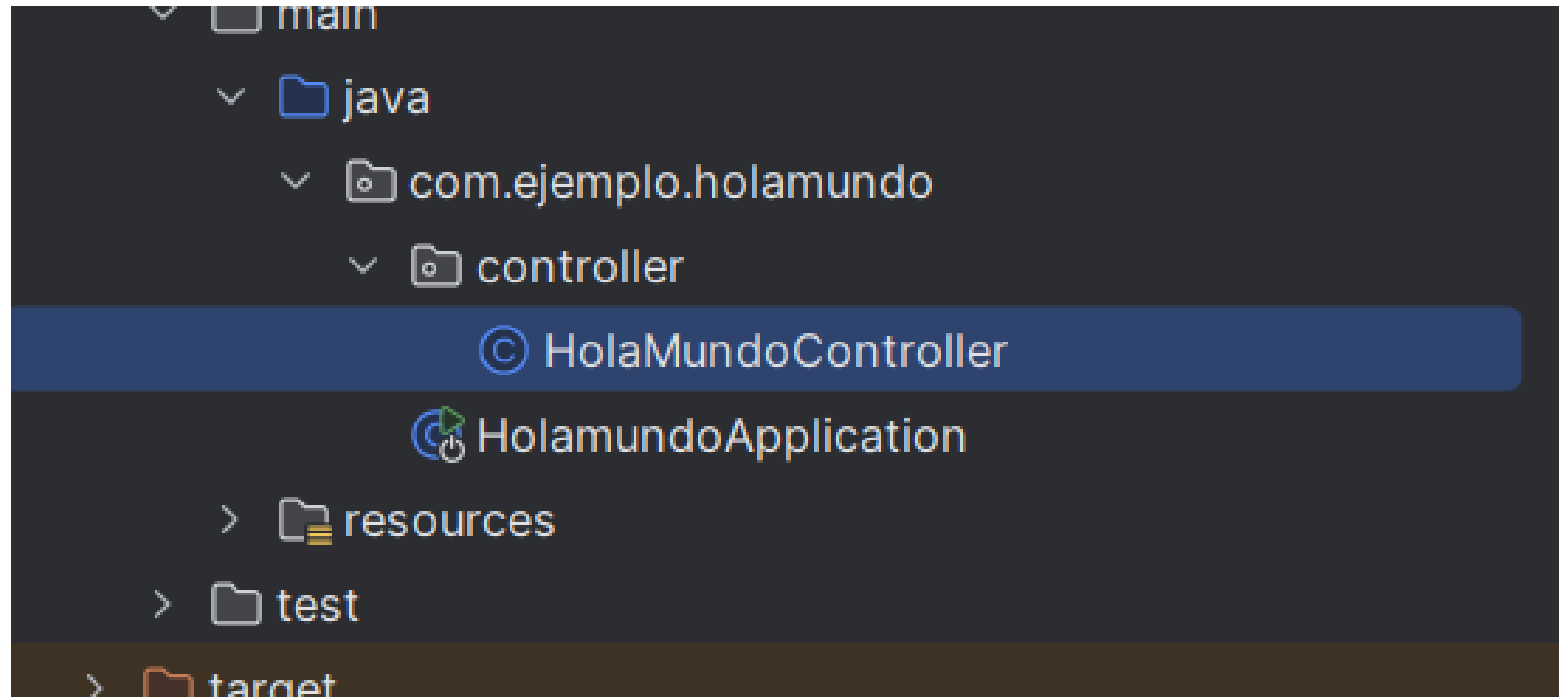
En el capítulo anterior...

¿Qué son las **@anotaciones** en Spring?

*"En Spring Boot tenemos algo llamado **anotaciones** y su característica principal es usar el simbolo "@" al frente de estas palabras claves. Las anotaciones realizan ciertos procesos internos que a su vez Spring Boot detecta y con uso ya sabe qué tipo de clase es. Si estas anotaciones no existieran entonces nosotros como **developers** tendríamos que especificar qué tipo de clase es, siendo esto mucho más complicado..."*

En el capítulo anterior...

¿Qué hace el package **controller**?



• Contenidos

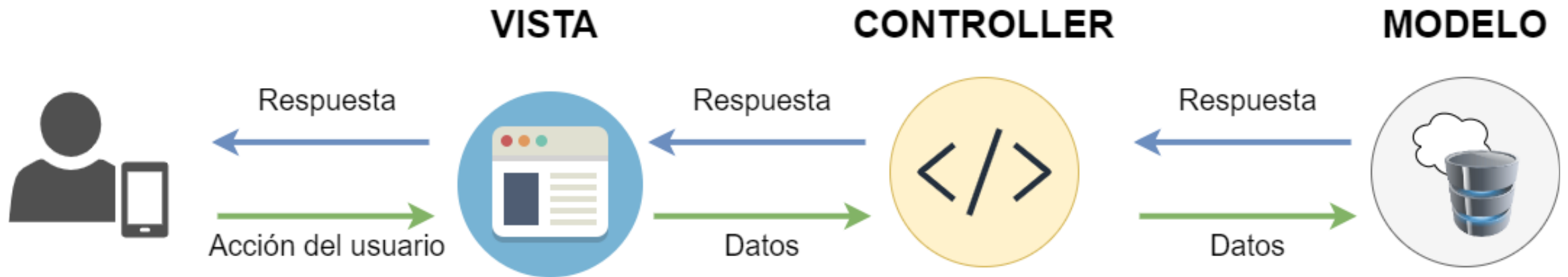
- Arquitectura MVC.
- Arquitectura en Spring Boot.
- Definición de cada uno de los componentes @Controller, @Service, @Repository y Modelo.
- Orden carpeta en proyecto nuevo.
- Dependencias en Maven.
- Realizar guías prácticas 2 y 3.
- Realizar actividad.
- Reflexión.

A black and white photograph of a man in a suit standing in a modern office, holding a tablet and smiling. The office has glass partitions and modern lighting. In the foreground, there is a desk with a coffee cup and some papers.

01

Arquitectura MVC

¿Qué es **MVC**?



• MVC

MVC (Model-View-Controller) es un patrón de arquitectura de software que separa una aplicación en tres componentes principales: el **modelo**, la **vista** y el **controlador**. Esta separación permite una mejor organización del código, facilita el mantenimiento y la escalabilidad de la aplicación.

MODELO



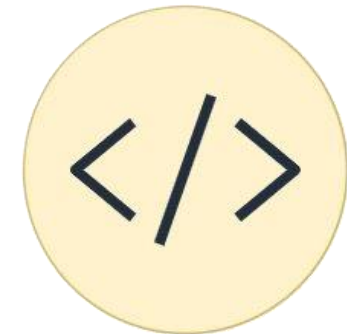
Representa los datos y la lógica de negocio de la aplicación.

VISTA



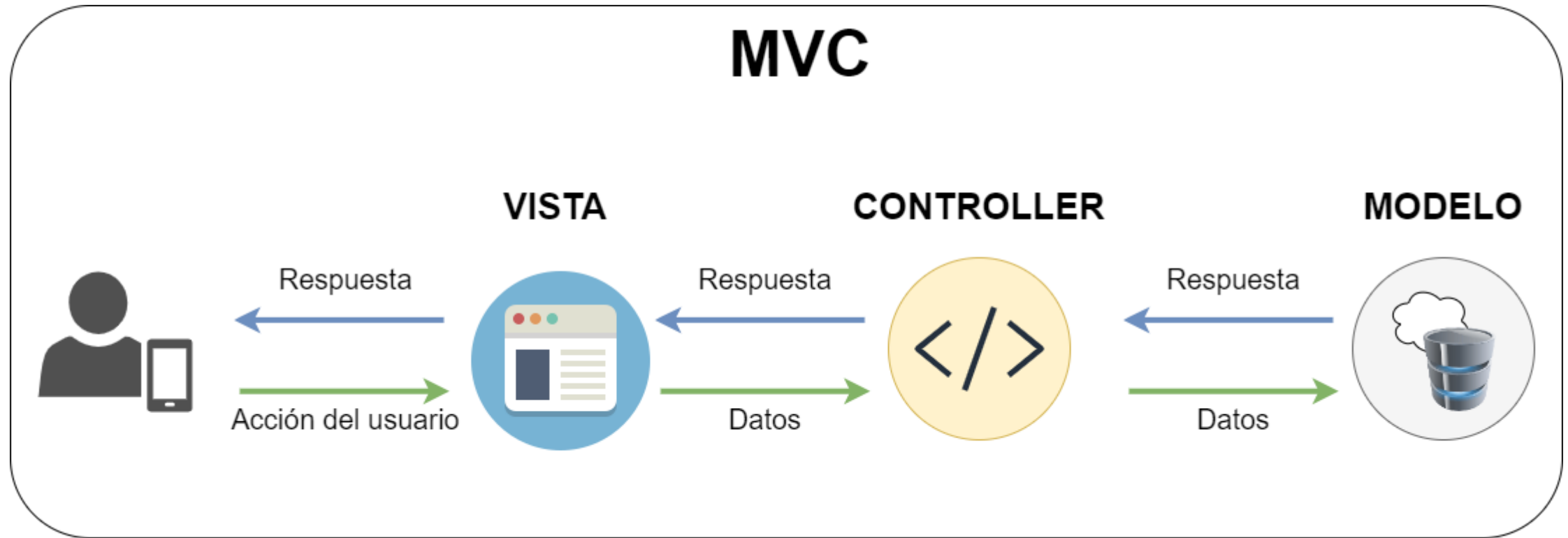
Presenta los datos al usuario y maneja la interacción del usuario.

CONTROLLER



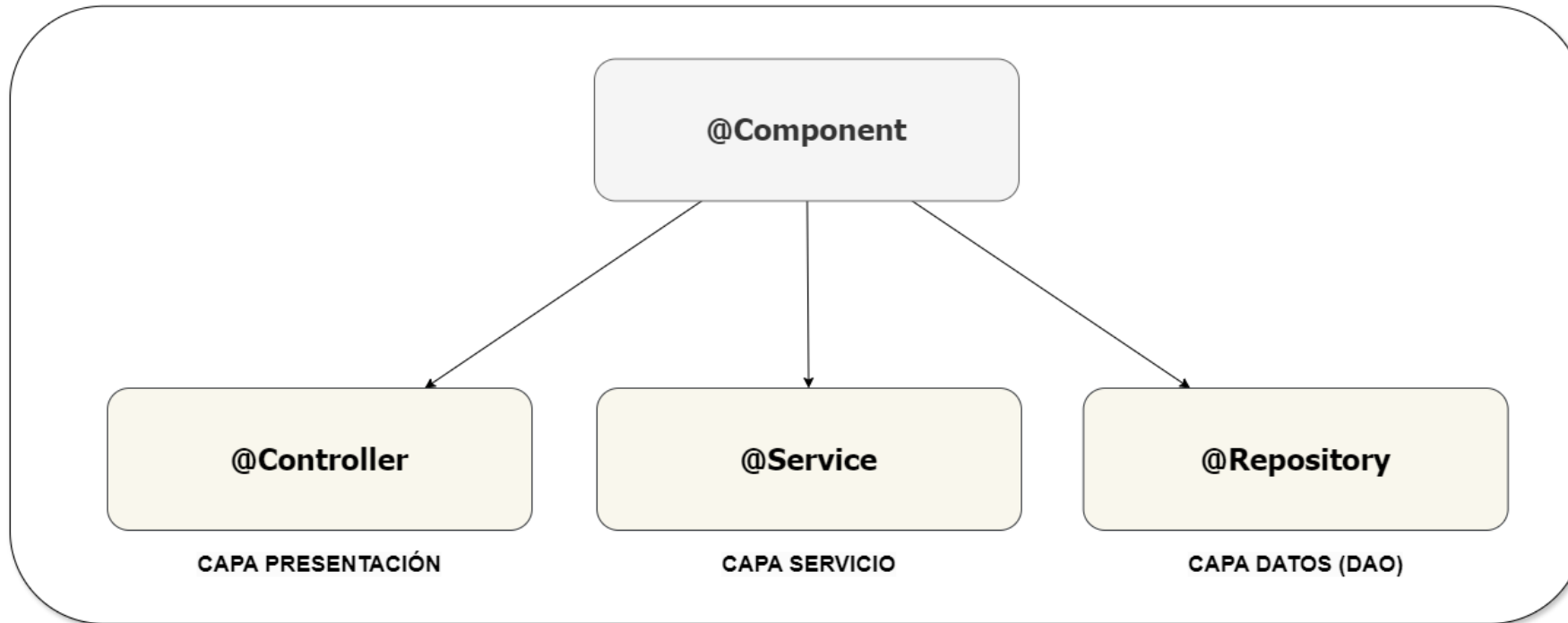
Actúa como intermediario entre el modelo y la vista.

- **MVC**



• Arquitectura en Spring Boot

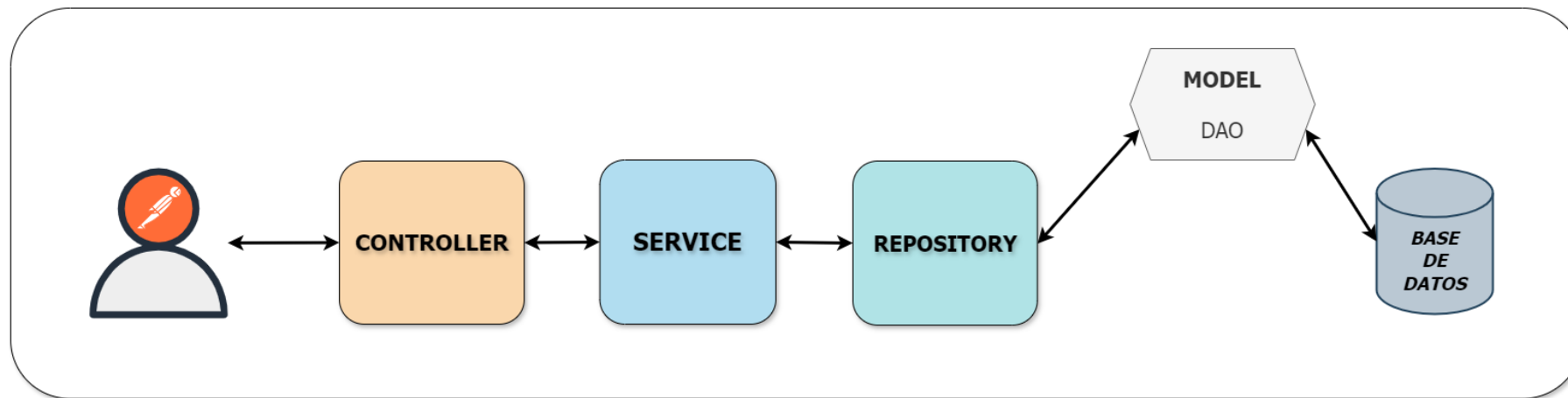
La arquitectura **Controller-Service-Repository (CSR)** en **Spring Boot** se utiliza para separar las responsabilidades en diferentes capas, facilitando la mantenibilidad y la escalabilidad de la aplicación.



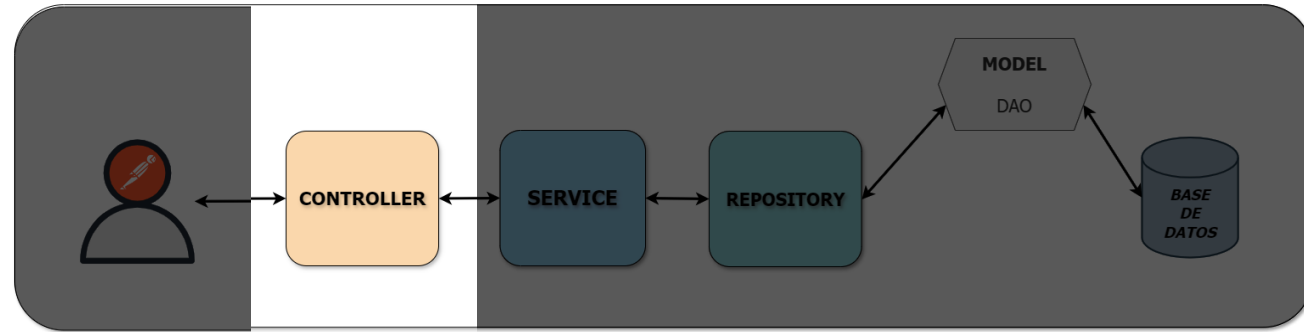
• Arquitectura en Spring Boot

Es una variación de **MVC** que enfatiza una clara separación entre la **lógica de la aplicación** (Controller), la **lógica de negocio** (Service) y el **acceso a datos** (Repository).

CSR separa explícitamente la lógica de negocio en una capa de servicio independiente, mientras que **MVC** combina la lógica de negocio y de aplicación en el controlador.



• Controller



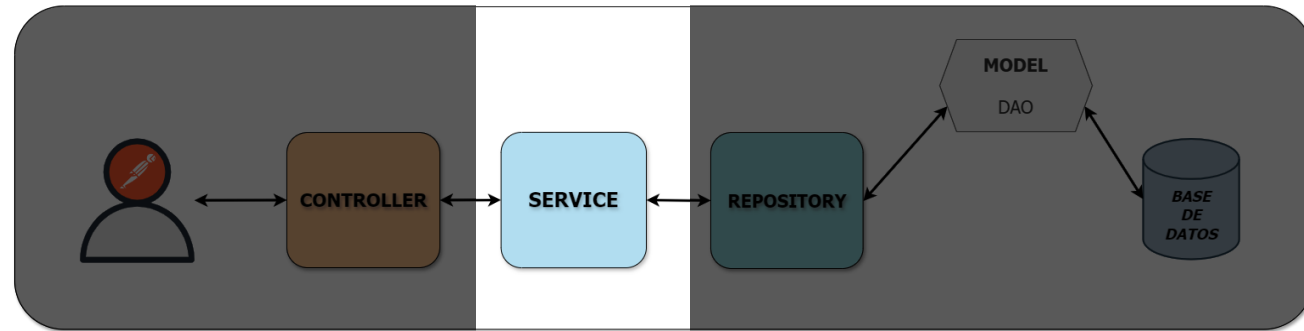
Responsabilidad

- Maneja las solicitudes **HTTP** entrantes y devuelve las **respuestas HTTP**.
- Actúa como intermediario entre la vista (front-end) y la lógica de negocio (servicio).

Características:

- Anotado con **@RestController** (para APIs RESTful) o **@Controller** (para controladores MVC tradicionales).
- Define los endpoints de la API utilizando anotaciones como **@GetMapping**, **@PostMapping**, **@PutMapping**, **@DeleteMapping**.
- Procesa datos de las solicitudes HTTP y envía datos de respuesta a los clientes.

• Service



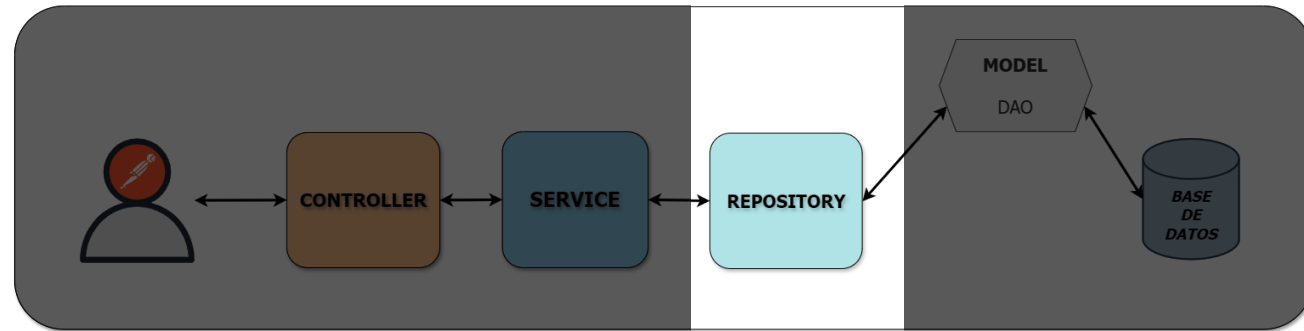
Responsabilidad:

- Contiene la lógica de negocio de la aplicación.
- Orquesta las operaciones entre el controlador y el repositorio.
- Proporciona una capa intermedia para separar las preocupaciones.

Características:

- Anotado con **@Service**.
- Puede contener métodos transaccionales utilizando la anotación **@Transactional**.
- Se enfoca en la lógica de negocio y reglas de negocio.

• Repository



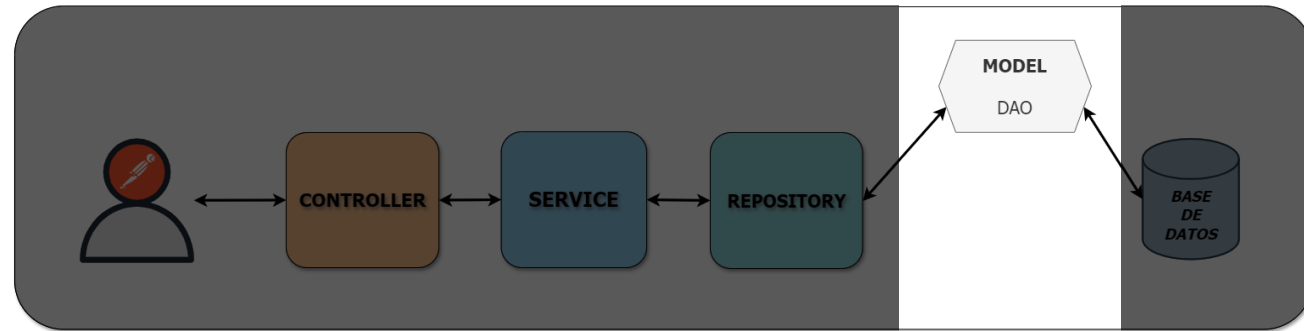
Responsabilidad:

- Interactúa directamente con la base de datos para realizar operaciones **CRUD**.
(Crear, Leer, Actualizar, Eliminar)
- Proporciona una **abstracción** sobre las operaciones de persistencia y recuperación de datos.

Características:

- Anotado con **@Repository** o extendiendo interfaces proporcionadas por Spring Data JPA, como JpaRepository.
- Define métodos personalizados para consultas específicas.
- Utiliza anotaciones de JPA para mapear las entidades de Java a las tablas de la base de datos.

• Modelo/DAO



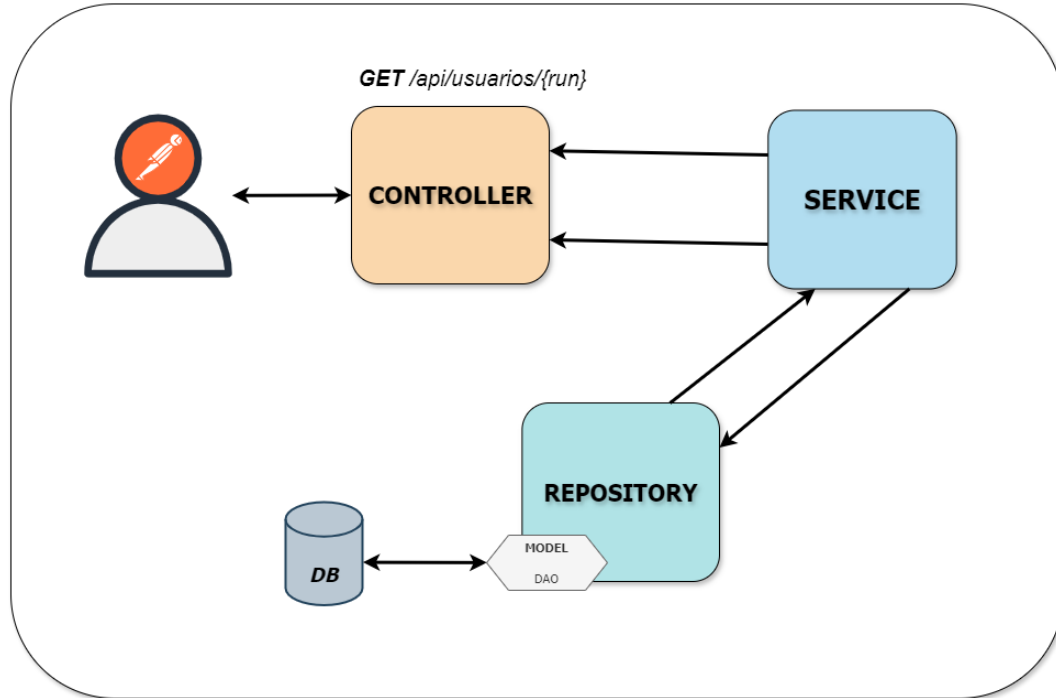
Responsabilidad:

- Representa los **datos de la aplicación** y las reglas de negocio asociadas a esos datos.
- Define la estructura de los datos y cómo se almacenan en la base de datos.

Características:


- Utiliza anotaciones de **JPA (Java Persistence API)** para mapear clases a tablas de la base de datos.
- Contiene atributos que corresponden a columnas en la base de datos.
- Incluye métodos **getters** y **setters** para acceder y modificar los datos.

• ¿Cómo funciona?



1. **Solicitud del Cliente:** El cliente envía una solicitud HTTP (por ejemplo, `GET /usuarios/{email}`) al servidor.
2. **Controlador:** El controlador recibe la solicitud y la dirige al servicio adecuado.
3. **Servicio:** El servicio contiene la lógica de negocio y llama al repositorio para interactuar con la base de datos.
4. **Repositorio/DAO:** El repositorio ejecuta las operaciones de persistencia (por ejemplo, buscar un usuario por email) y devuelve los datos al servicio.
5. **Servicio:** El servicio puede realizar operaciones adicionales en los datos si es necesario y luego devolver los resultados al controlador.
6. **Controlador:** El controlador devuelve la respuesta HTTP al cliente con los datos solicitados.

• Orden de carpetas en el proyecto

 controller model repository service

- **Controller** maneja las solicitudes y respuestas HTTP.
- **Service** contiene la lógica de negocio.
- **DAO/Repository** interactúa con la base de datos.
- **Model** representa la estructura de los datos y cómo se almacenan en la base de datos.

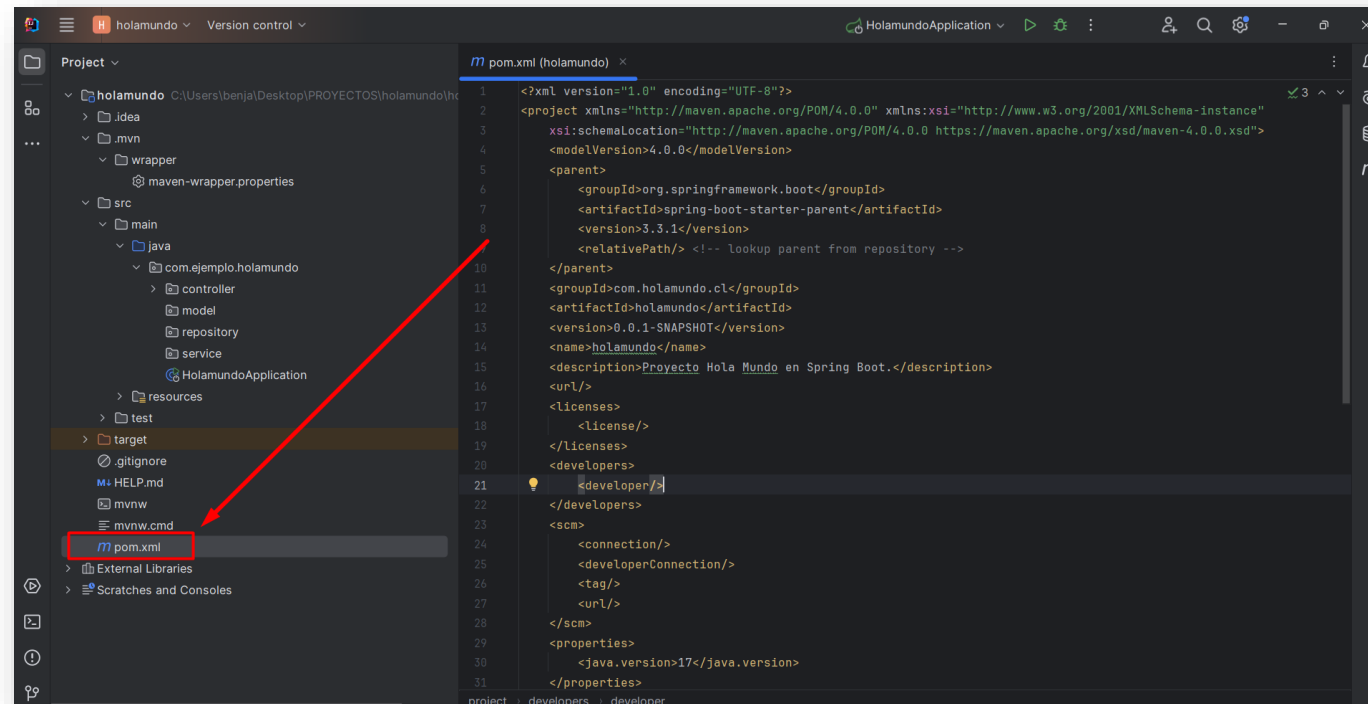
A black and white photograph of a man and a woman in a modern office setting. The man, on the left, is wearing a light-colored button-down shirt and dark trousers. The woman, on the right, is wearing a dark blazer over a light-colored turtleneck. They are both looking at a tablet held by the man. In the background, there are office desks, a computer monitor displaying a complex diagram, and another person working at a desk. The overall atmosphere is professional and collaborative.

02

SPRINT: Dependencias

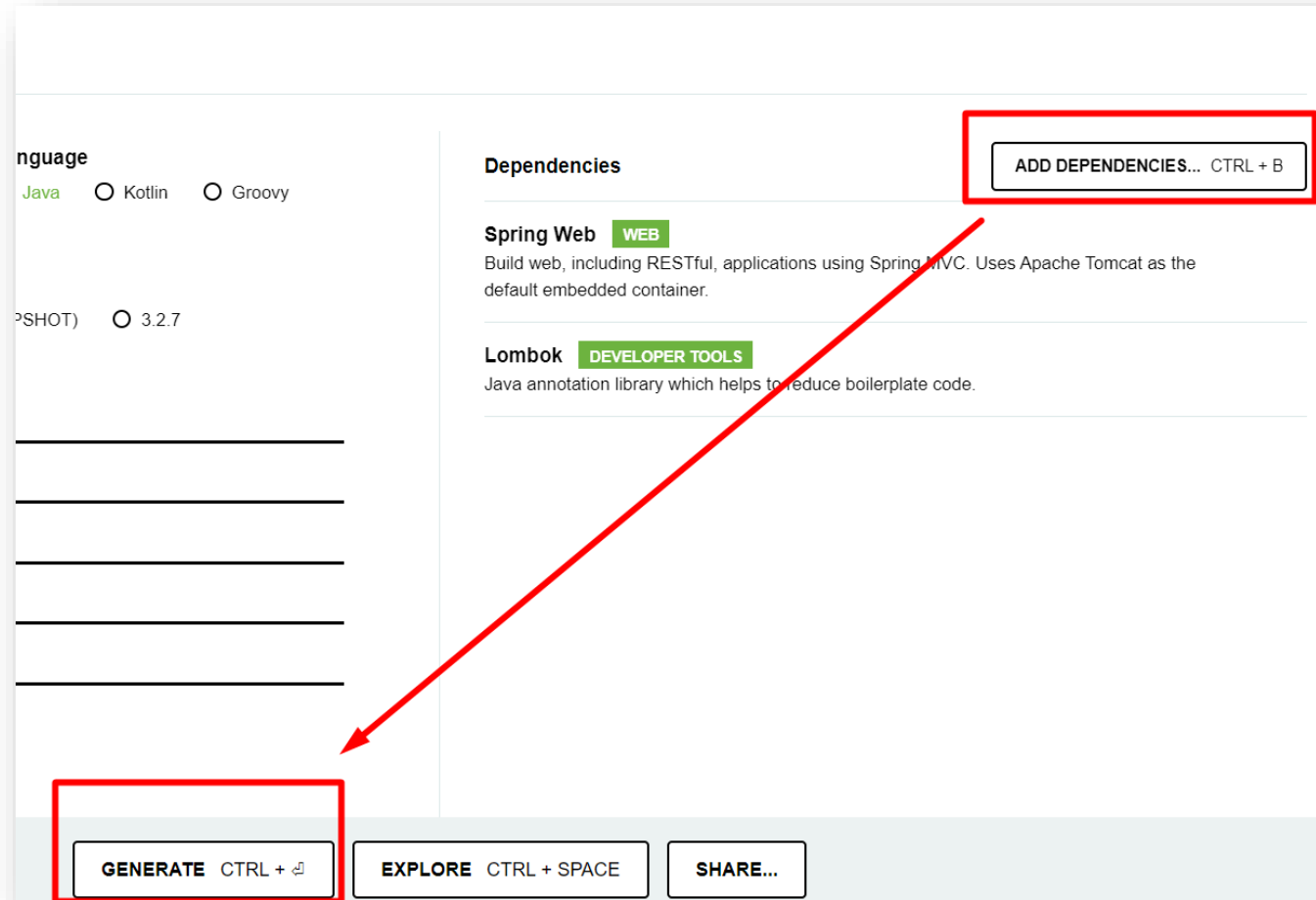
• Dependencias en Maven (pom.xml)

Las **dependencias** se manejan a través de un archivo de configuración, generalmente **pom.xml** para proyectos que usan Maven o build.gradle para proyectos que usan Gradle. Estas dependencias especifican las bibliotecas y módulos que tu proyecto necesita para funcionar.



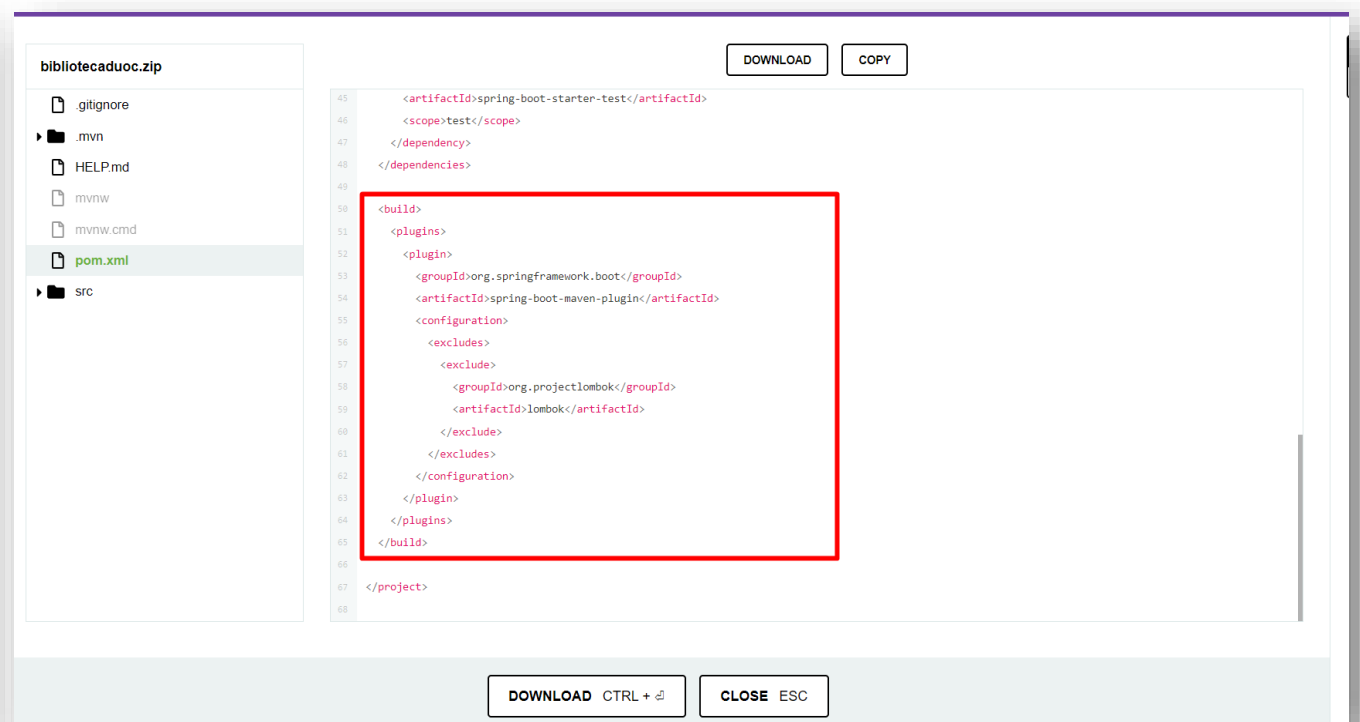
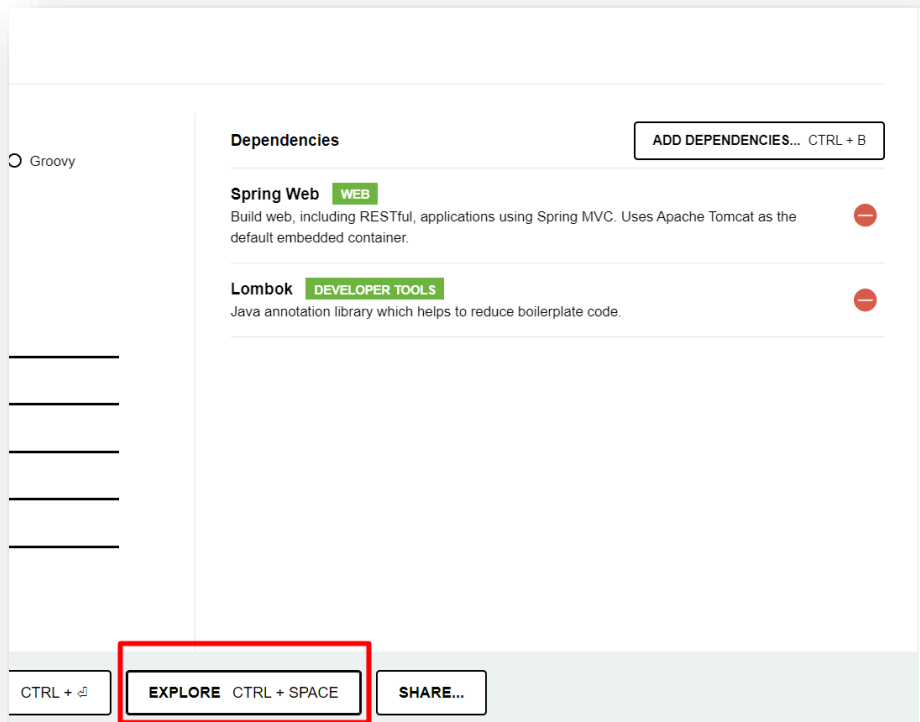
• Añadir dependencias

Cuando creamos un nuevo proyecto añadimos dependencia a la base.



• Añadir dependencias

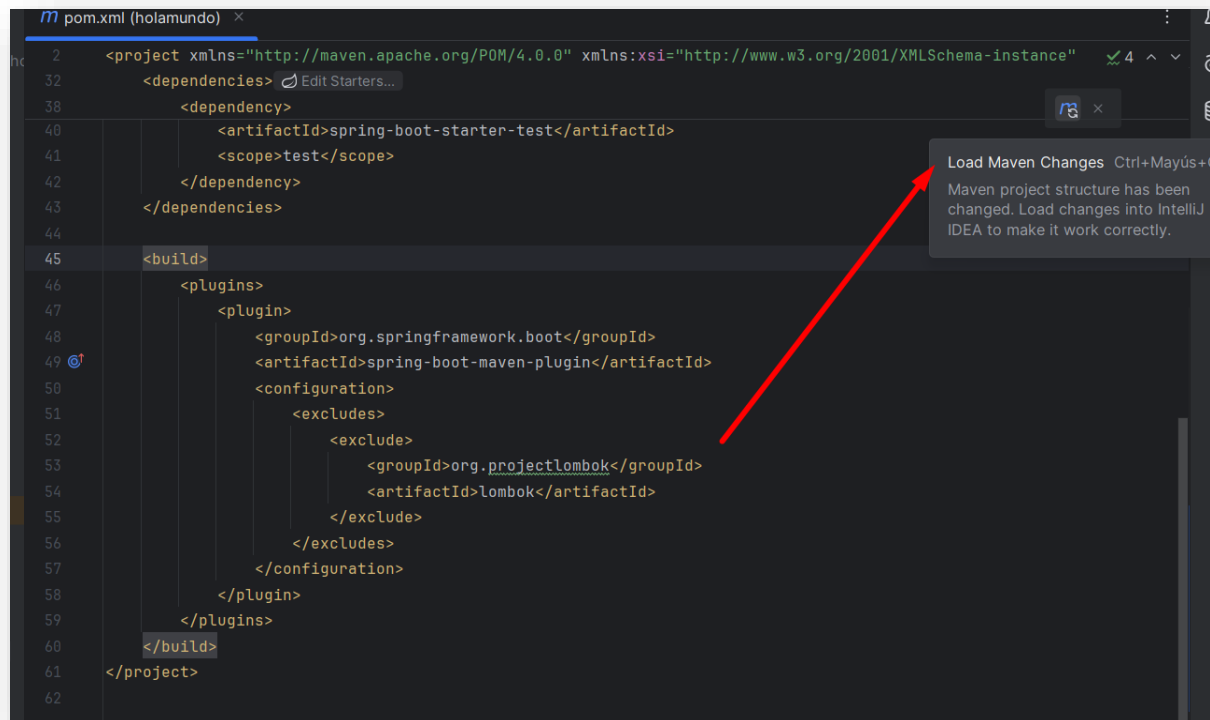
Podemos añadirla de forma manual las dependencias.



• Añadir dependencias

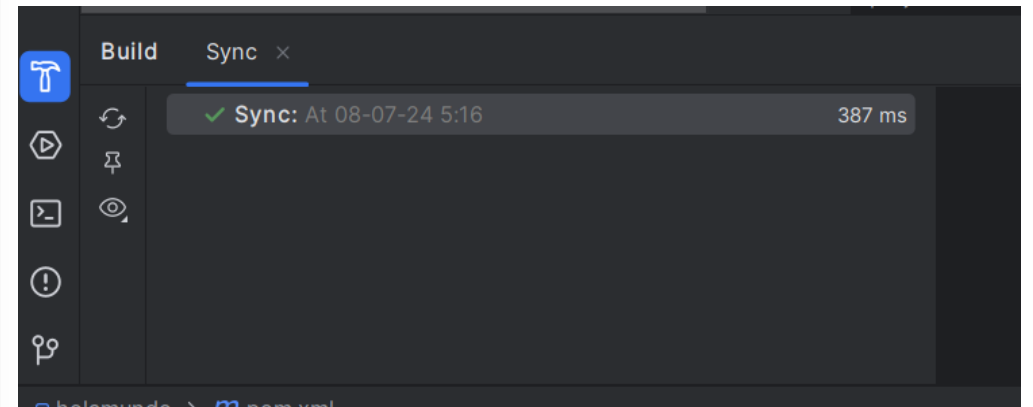
Copiando el código XML y añadiéndolo a pom.xml.

Dar a sincronizar y listo.



```
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
32 <dependencies>
38 <dependency>
40 <artifactId>spring-boot-starter-test</artifactId>
41 <scope>test</scope>
42 </dependency>
43 </dependencies>
44
45 <build>
46 <plugins>
47 <plugin>
48 <groupId>org.springframework.boot</groupId>
49 <artifactId>spring-boot-maven-plugin</artifactId>
50 <configuration>
51 <excludes>
52 <exclude>
53 <groupId>org.projectlombok</groupId>
54 <artifactId>lombok</artifactId>
55 </exclude>
56 </excludes>
57 </configuration>
58 </plugin>
59 </plugins>
60 </build>
61 </project>
```

Load Maven Changes Ctrl+Mayús+O
Maven project structure has been changed. Load changes into IntelliJ IDEA to make it work correctly.



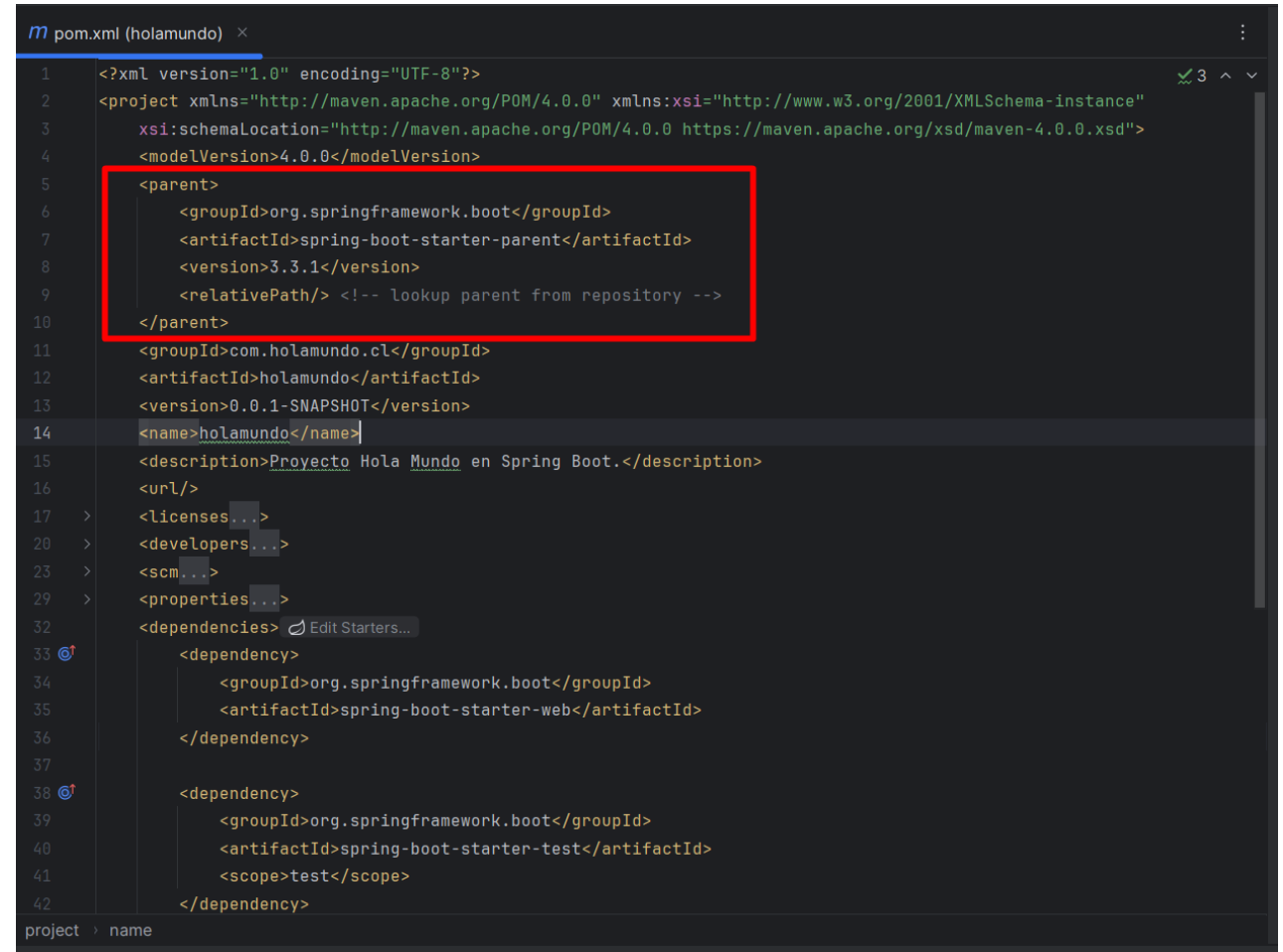
• Dependencias más comunes

spring-boot-starter-web:

Incluye Spring MVC y Tomcat (por defecto), necesarios para desarrollar aplicaciones web RESTful.

spring-boot-starter-data-jpa:

Incluye Spring Data JPA, Hibernate, y otras dependencias necesarias para trabajar con JPA.



```
m pom.xml (holamundo) x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>3.3.1</version>
9     <relativePath/> <!-- lookup parent from repository -->
10  </parent>
11  <groupId>com.holamundo.cl</groupId>
12  <artifactId>holamundo</artifactId>
13  <version>0.0.1-SNAPSHOT</version>
14  <name>holamundo</name>
15  <description>Proyecto Hola Mundo en Spring Boot.</description>
16  <url/>
17  <licenses...>
20  <developers...>
23  <scm...>
29  <properties...>
32  <dependencies> Edit Starters...
33    <dependency>
34      <groupId>org.springframework.boot</groupId>
35      <artifactId>spring-boot-starter-web</artifactId>
36    </dependency>
37
38    <dependency>
39      <groupId>org.springframework.boot</groupId>
40      <artifactId>spring-boot-starter-test</artifactId>
41      <scope>test</scope>
42    </dependency>
project > name
```


• Dependencias más comunes

- **lombok:**

- Incluye la biblioteca Lombok para reducir el código boilerplate.

- **com.h2database:h2:**

- Incluye H2, una base de datos en memoria útil para pruebas y desarrollo.

- **spring-boot-starter-test:**

- Incluye dependencias para pruebas unitarias y de integración, como JUnit, Hamcrest y Mockito.



03

Dependencia Lombok

• Lombok

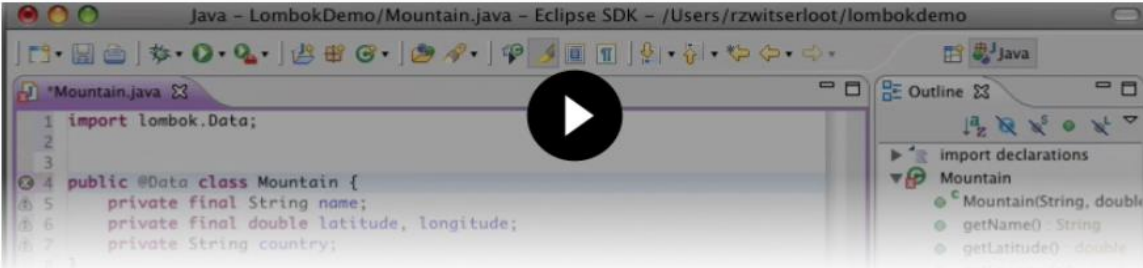
Es una **biblioteca de Java** que se utiliza para reducir el **código repetitivo** y **redundante mediante** el uso de anotaciones.

Simplifica el desarrollo al generar automáticamente el código boilerplate, como **getters**, **setters**, **constructores**, **métodos toString**, **equals**, **hashCode**, entre otros, durante la compilación. Esto mejora la legibilidad y mantenibilidad del código.

[IR A PROYECTO LOMBOK](#)


Project Lombok

Project Lombok is a java library that automatically plugs into your editor and build tools, spicing up your java. Never write another getter or equals method again, with one annotation your class has a fully featured builder, Automate your logging variables, and much more.



Show me a text and images based explanation and tutorial instead!

Project Lombok is **powered by:**

PRISMA. Michael Hashe  **subshell**

TIDELIFT

[I want to support Project Lombok too!](#)

• GUÍA PRÁCTICA



Ingresa al AVA de esta actividad
y desarrolla la

**Guía práctica - Proyecto
Biblioteca en Spring Boot**

• GUÍA PRÁCTICA



Ingresa al AVA de esta actividad
y desarrolla la

**Guía práctica - Proyecto
Biblioteca en Spring
Boot Reporte**

• ACTIVIDAD

En el proyecto de la Biblioteca se requiere crear un sistema de **préstamos de libros**.

Ingresa al AVA de esta actividad y desarrolla la actividad para de **Actividad - Proyecto Biblioteca en Spring Boot - Solicitud de libros**

Comparte tus resultados con tus compañeros.



Reflexionemos



- ¿Cuál es la diferencia entre `@Controller` y `@RestController` en Spring Boot?
- Explica la estructura de carpetas en un proyecto típico de Spring Boot.
- Describe el flujo de trabajo típico en una aplicación Spring Boot que sigue el patrón CSR (Controller-Service-Repository).
- ¿Cuáles son las dependencias?
- ¿Cómo se añaden manualmente las dependencias en un proyecto Spring Boot utilizando Maven?

04

¡Muchas gracias!

DuocUC[®]

CERCANÍA. LIDERAZGO. FUTURO.

duoc.cl

7 AÑOS
ACREDITADO



DESDE AGOSTO 2017 HASTA AGOSTO 2024.
DOCENCIA DE PREGRADO. GESTIÓN
INSTITUCIONAL. VINCULACIÓN CON EL MEDIO.