Ecommerce con Spring Boot

Creacion de la BD: La base de datos se creara en MySQL Workbench y se llamara Ecommerce Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Creacion del Proyecto : la configuración inicial del proyecto Java para un sistema de ecommerce utilizando Spring Initializr, una herramienta oficial para generar proyectos Spring Boot preconfigurados.

**Project**

* **Maven**: Se ha seleccionado *Maven* como herramienta de construcción. Esta permite gestionar dependencias, compilar el proyecto y empaquetarlo.

**Language**

* **Java**: Se ha elegido Java como lenguaje principal del proyecto, en lugar de Kotlin o Groovy.

**Spring Boot Version**

* **3.5.4**: Es la versión estable seleccionada de Spring Boot, compatible con Java 17 y con mejoras respecto a versiones anteriores.

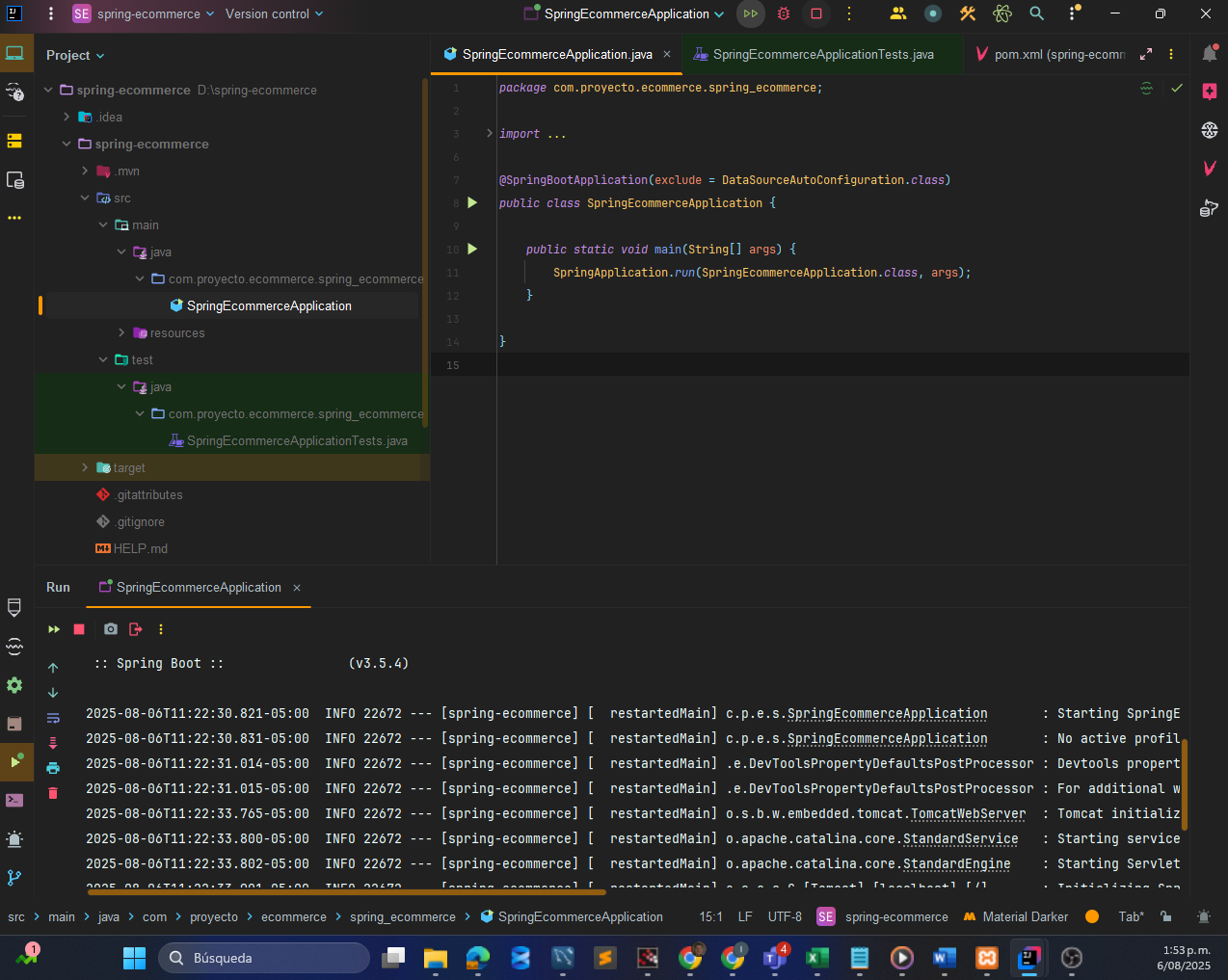
**Dependencias Incluidas**

Las siguientes librerías han sido seleccionadas al momento de generar el proyecto:

1. **Lombok (Developer Tools)**  
   Reduce el código repetitivo (boilerplate) mediante anotaciones como @Getter, @Setter, @Builder, etc.
2. **Spring Web (Web)**  
   Habilita la creación de aplicaciones web, especialmente RESTful APIs, usando Spring MVC y Tomcat como servidor embebido.
3. **Thymeleaf (Template Engines)**  
   Motor de plantillas que permite generar contenido HTML dinámico desde el servidor.
4. **Spring Data JPA (SQL)**  
   Facilita el acceso a bases de datos mediante Java Persistence API (JPA) y Hibernate.
5. **MySQL Driver (SQL)**  
   Controlador JDBC necesario para conectar la aplicación con bases de datos MySQL.
6. **Spring Boot DevTools (Developer Tools)**  
   Herramientas de desarrollo que permiten reinicios automáticos, recarga en caliente (LiveReload) y una experiencia de desarrollo más ágil.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.



**Estructurar el proyecto y crear las clases para el modelo 02**

Para la estructura del proyecto se necesita una **estructura típica de paquetes (carpetas) organizada por capas según la arquitectura por capas o arquitectura de tipo MVC (Modelo - Vista - Controlador).**

**Paquetes que aparecen:**

**1. controller**

**Responsabilidad:** manejar las peticiones HTTP que vienen del cliente (navegador, Postman, etc.).

* Contiene las clases con anotaciones como @RestController o @Controller.
* Cada método se encarga de atender una URL y delega la lógica al servicio.

**2. model**

**Responsabilidad:** definir las entidades del dominio (por ejemplo, productos, usuarios, órdenes, etc.).

* Contiene clases anotadas con @Entity si usas JPA.
* Representan las tablas de tu base de datos.

**3. repository**

**Responsabilidad:** acceder a la base de datos.

* Contiene interfaces que extienden JpaRepository, CrudRepository, etc.
* Spring genera la implementación automáticamente.

**4. service**

**Responsabilidad:** contener la lógica de negocio.

* Aquí se hace el procesamiento de datos, validaciones, llamadas a los repositorios, etc.
* Usualmente están anotadas con @Service.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Dentro de la carpeta model creamos 4 clases de entidad que representan las tablas en la base de datos estas clases son Usuario, Producto. Orden, DetalleOrden y contienen la siguiente estructura

**Anotaciones de Lombok**

Estas anotaciones son parte de la librería Project Lombok que ayuda a reducir código repetitivo:

| **Anotación** | **¿Qué hace?** |
| --- | --- |
| @Getter | Genera automáticamente los métodos get para todos los atributos. |
| @Setter | Genera automáticamente los métodos set. |
| @AllArgsConstructor | Genera un constructor con todos los campos como parámetros. |
| @NoArgsConstructor | Genera un constructor vacío (sin argumentos). |

**Atributos de la clase**

La clase tiene los siguientes atributos:

Esto representa un usuario típico con:

* id: Identificador único.
* nombre: Nombre real.
* username: Nombre de usuario.
* mail: Correo electrónico.
* direccion: Dirección física.
* telefono: Número de contacto.
* tipo: Posiblemente el rol del usuario (ej: admin, cliente...).
* password: Contraseña (💡 no deberías mostrarla en toString por seguridad).

**🔁 Método toString()**

Este método sobrescrito (@Override) sirve para obtener una representación en texto del objeto, útil para logs o debugging.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

**Cómo versionar el proyecto usando Git y GitHub**

**Conceptos**

@SpringBootApplication(exclude = DataSourceAutoConfiguration.*class*):

**¿Qué hace @SpringBootApplication?**

Esa anotación es un **atajo** que combina varias anotaciones de Spring Boot:

@Configuration

@EnableAutoConfiguration

@ComponentScan

Es decir: le dice a Spring Boot:

“Configura automáticamente mi aplicación, escanea los componentes y usa esta clase como configuración principal.”

**¿Y qué significa exclude = DataSourceAutoConfiguration.class?**

Spring Boot tiene una característica llamada **"auto-configuración"**, donde **detecta automáticamente** que tienes una base de datos y configura la conexión por ti.

Pero si no tienes una base de datos configurada (como es tu caso), **esa auto-configuración falla**.

Al hacer esto:

@SpringBootApplication(exclude = DataSourceAutoConfiguration.class)

Le estás diciendo a Spring Boot:

“❌ No intentes configurar una base de datos automáticamente.”