

Guía práctica HOSPITAL V&M

PARTE 1

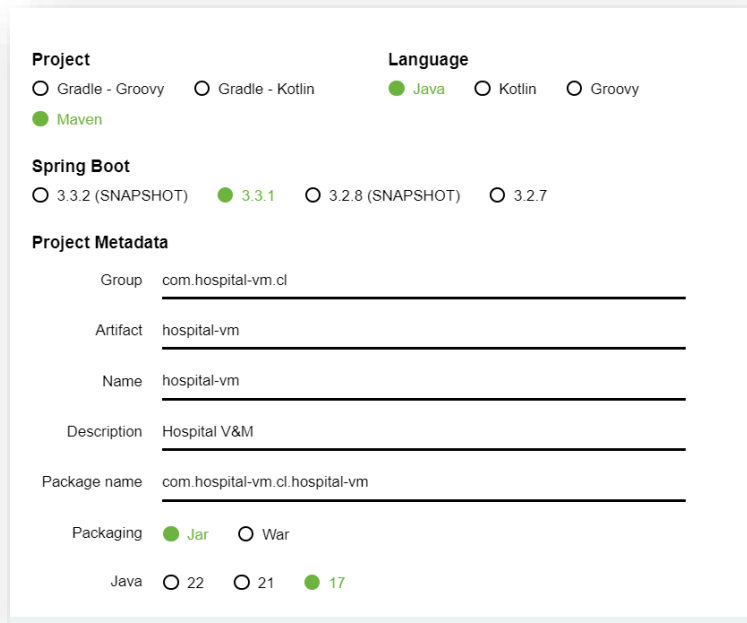
Configurar MYSQL



En esta guía, detallaremos paso a paso el desarrollo de un proyecto en Spring llamado "Hospital V&M". A continuación, se describen los pasos para crear el proyecto en Spring Boot agregando dependencias de base de datos MYSQL.

Paso 1: Ir a Spring Initializr

- ❑ Abre tu navegador web y ve a la siguiente página: Spring Initializr. (<https://start.spring.io/>)



The screenshot shows the Spring Initializr web form with the following configuration:

- Project:** ☐ Gradle - Groovy, ☐ Gradle - Kotlin, ☒ **Java**, ☐ Kotlin, ☐ Groovy
- ☒ **Maven**
- Spring Boot:** ☐ 3.3.2 (SNAPSHOT), ☒ **3.3.1**, ☐ 3.2.8 (SNAPSHOT), ☐ 3.2.7
- Project Metadata:**
 - Group:
 - Artifact:
 - Name:
 - Description:
 - Package name:
- Packaging:** ☒ **Jar**, ☐ War
- Java:** ☐ 22, ☐ 21, ☒ **17**

Paso 2: Añadir Dependencias

- ❑ Haz clic en el botón "Add Dependencies" y añade la siguiente dependencia:
 - **Spring Web:** Para crear aplicaciones web, incluyendo RESTful.
 - **MySQL Driver:** Para conectarse a Mysql.
 - **Lombok:** Reducir el código boilerplate en las clases Java.

- **Spring Data JPA:** Simplificar la persistencia de datos en bases de datos relacionales.

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.	—
Lombok DEVELOPER TOOLS	Java annotation library which helps to reduce boilerplate code.	—
Spring Data JPA SQL	Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.	—

Definiciones:**Spring Web**

Propósito: Crear aplicaciones web, incluidas API RESTful.

Uso:

- Define controladores (@RestController) para manejar solicitudes HTTP (GET, POST, PUT, DELETE).
- Mapea URLs a métodos en los controladores (@GetMapping, @PostMapping, etc.).

MySQL Driver

Propósito: Conectar la aplicación a una base de datos MySQL.

Uso:

- Proporciona la biblioteca JDBC necesaria para interactuar con MySQL.
- Configura la conexión a la base de datos mediante propiedades en **application.properties**.

Lombok

Propósito: Reducir el código boilerplate en las clases Java.

Uso:

- Genera automáticamente getters, setters, constructores, toString, equals, y hashCode usando anotaciones como
 - @Getter
 - @Setter
 - @NoArgsConstructor
 - @AllArgsConstructor
 - @Data

Spring Data JPA

Propósito: Simplificar la persistencia de datos en bases de datos relacionales.

Uso:

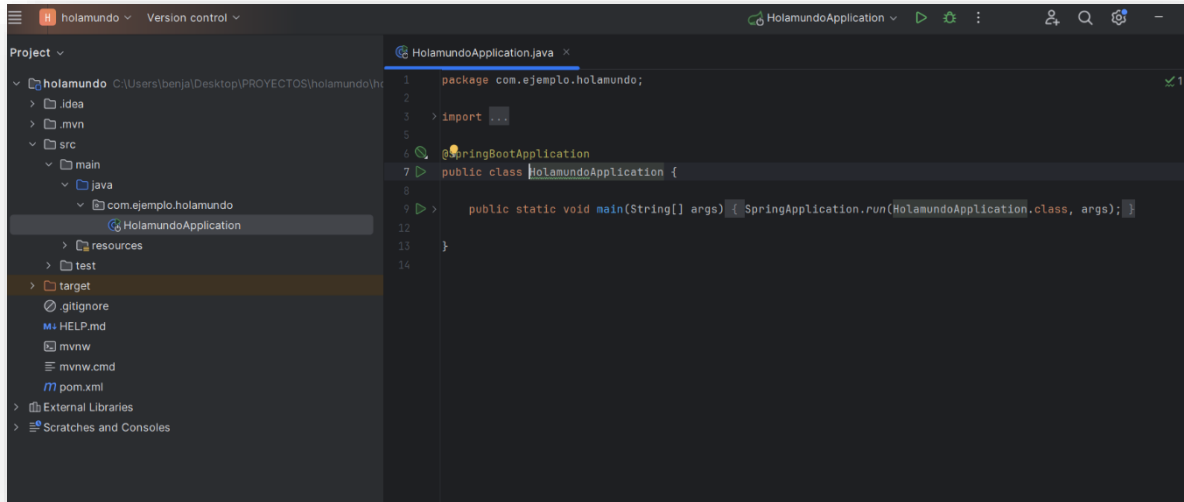
- Define entidades (@Entity) que representan tablas en la base de datos.
- Utiliza repositorios (JpaRepository) para realizar operaciones CRUD sin necesidad de escribir SQL manualmente.

Paso 3: Generar el Proyecto

- ☐ Una vez que hayas configurado todo, haz clic en el botón "**GENERATE**" para descargar un archivo ZIP con tu proyecto Spring Boot.

Paso 4: Importar el Proyecto en tu IDE

- ❑ Descomprime el archivo ZIP descargado y abre tu IDE preferido (VS Code o IntelliJ IDEA).
 - Navega hasta la carpeta descomprimida y selecciona el proyecto para importarlo.



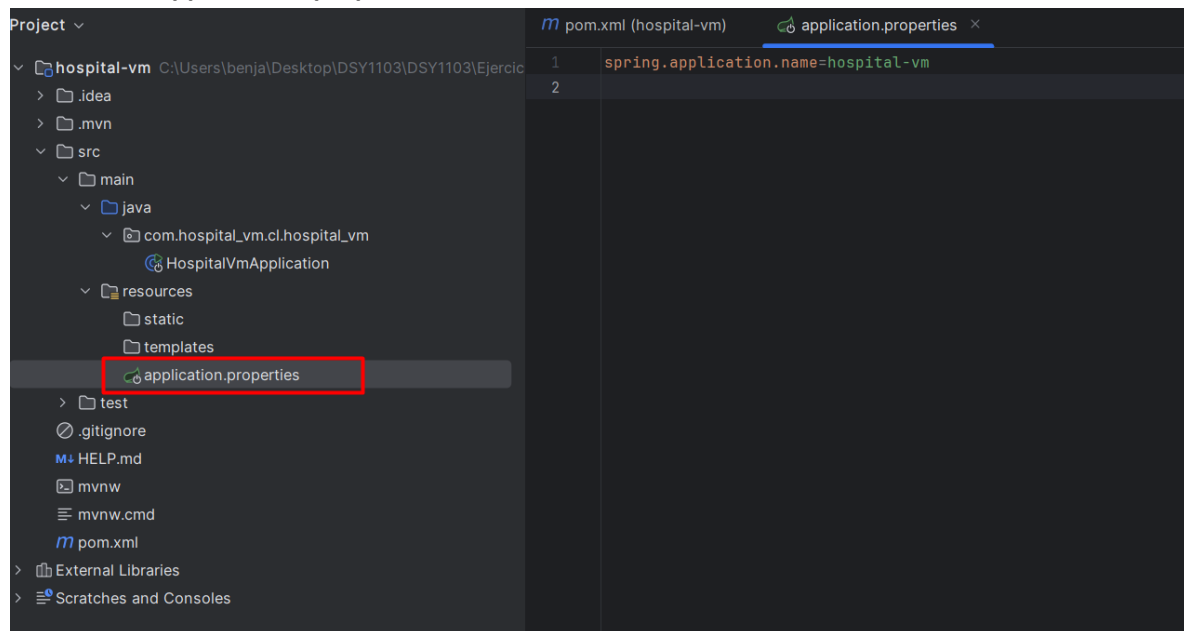
Paso 5: Configuración base de datos

Base de datos MYSQL

Ir a documento de base de datos en MYSQL

Configurar el proyecto con la base de datos de MYSQL

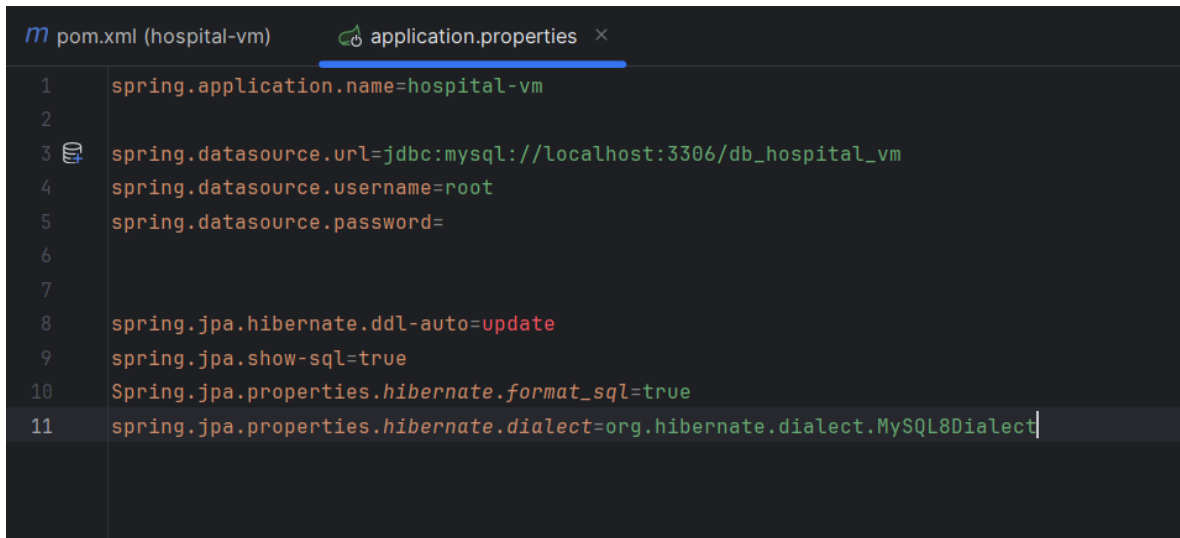
Ir al archivo **application.properties**



Añadir lo siguiente

```
spring.datasource.url=jdbc:mysql://localhost:3306/db_hospital_vm
spring.datasource.username=root
spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```



```
1  spring.application.name=hospital-vm
2
3  spring.datasource.url=jdbc:mysql://localhost:3306/db_hospital_vm
4  spring.datasource.username=root
5  spring.datasource.password=
6
7
8  spring.jpa.hibernate.ddl-auto=update
9  spring.jpa.show-sql=true
10 Spring.jpa.properties.hibernate.format_sql=true
11 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

- ☐ **spring.datasource.url**: URL de conexión a la base de datos.
- ☐ **spring.datasource.username**: Usuario de la base de datos.
- ☐ **spring.datasource.password**: Contraseña del usuario.

- ☐ **spring.jpa.hibernate.ddl-auto**: Estrategia de gestión del esquema de la base de datos (creación y actualización).
- ☐ **spring.jpa.show-sql**: Muestra las consultas SQL generadas en la consola.
- ☐ **spring.jpa.properties.hibernate.format_sql**: Formatear las consultas SQL generadas por Hibernate para que sean más legibles cuando se imprimen en la consola.
- ☐ **spring.jpa.properties.hibernate.dialect**: Dialecto específico de la base de datos utilizado por Hibernate.

Configuración de la base de datos

Spring.datasource.url

```
spring.datasource.url=jdbc:mysql://localhost:3306/hospital_v_m
```

Descripción: Especifica la URL de la base de datos a la que se conectará la aplicación.

Detalles:

- ☐ jdbc:mysql://: Protocolo JDBC para MySQL.
- ☐ localhost: El host donde se encuentra la base de datos. En este caso, es el mismo servidor donde se ejecuta la aplicación (localhost).
- ☐ 3306: El puerto por defecto para MySQL.
- ☐ db_hospital_v_m: El nombre de la base de datos a la que se conectará.

Spring.datasource.username

```
spring.datasource.username=tu_usuario
```

Descripción: Especifica el nombre de usuario que la aplicación usará para conectarse a la base de datos.

spring.datasource.password

```
spring.datasource.password=tu_pass
```

Descripción: Especifica la contraseña correspondiente al usuario que la aplicación usará para conectarse a la base de datos.

Configuración de JPA e Hiberante

spring.jpa.hibernate.ddl-auto

`spring.jpa.hibernate.ddl-auto=update`

Descripción: Configura la estrategia de Hibernate para la gestión del esquema de la base de datos.

Valores posibles:

- **validate:** Valida el esquema existente, pero no hace cambios.
- **update:** Actualiza el esquema de la base de datos según las entidades JPA. Si la tabla no existe, la crea. Si hay cambios en la entidad, los aplica.
- **create:** Crea el esquema de la base de datos cada vez que se inicia la aplicación. Borra los datos existentes.
- **create-drop:** Crea el esquema al iniciar la aplicación y lo elimina al cerrarla.
- **none:** No hace nada con el esquema de la base de datos.

***Nota:** update es útil en desarrollo, pero en producción es más seguro usar validate o none.*

spring.jpa.show-sql

`spring.jpa.show-sql=true`

Descripción: Indica si se deben mostrar las consultas SQL generadas por Hibernate en la consola.

Valores posibles: true para mostrar las consultas, false para no mostrarlas.

spring.jpa.properties.hibernate.format_sql

`spring.jpa.properties.hibernate.format_sql=true`

Descripción: Formatear las consultas SQL generadas por Hibernate para que sean más legibles cuando se imprimen en la consola.

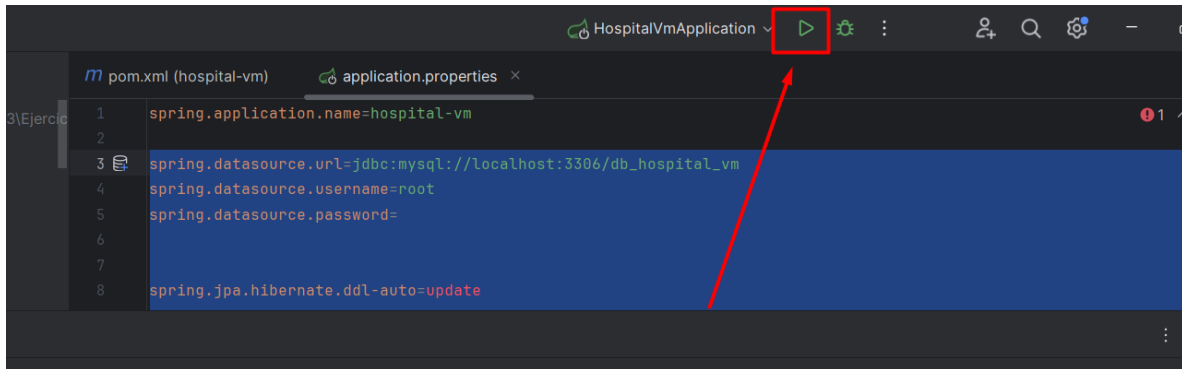
Spring.jpa.properties.hibernate.dialect

`spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect`

Descripción: Especifica el dialecto de Hibernate que se debe utilizar. El dialecto es una configuración específica de la base de datos que Hibernate necesita para generar el SQL adecuado.

Valor: `org.hibernate.dialect.MySQL5Dialect` indica que se está utilizando MySQL como base de datos. Si se usa una versión más reciente de MySQL, puedes optar por `MySQL8Dialect`.

Paso 6: Ejecutar aplicación



Verificar que todo funciona correctamente

