

# MANUAL PRUEBA SPRING

Fernando López-Nuño Riera

## Índice

1.- Creación del proyecto Spring.....	2
2.- Creación de modelos.....	2
3.- Creación de Repositories.....	3
4.- Creación de controllers y DTOs.....	3
5.- Conexión con BBDD y testeo básico.....	4
6.- Front.....	6
7.- Tests.....	6
8.- Ejecución de la app.....	6

# 1.- Creación del proyecto Spring

NOTA: para ejecutar la app ver punto 8.

En primer lugar vamos a proceder a crear nuestro proyecto Spring en VSC. Para ello nos dirigiremos a la web de generación de proyectos Spring:

[start.spring.io](https://start.spring.io)

Escogeremos Maven, Java como lenguaje y la última versión estable de Spring Boot, en este caso la 3.5.3, y Java 21.

A continuación añadimos las dependencias que usaremos en nuestro proyecto: Spring Web, para crear aplicaciones REST, y Spring data JPA & MySQL driver para la base de datos, que será MySQL:

The screenshot shows the Spring Boot project generator interface. It is divided into three main sections: Project, Language, and Dependencies.

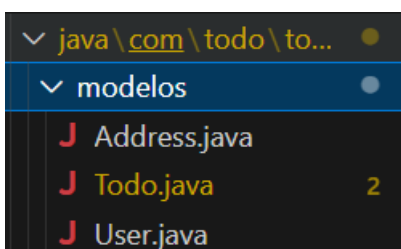
- Project:** Includes options for Gradle (selected) and Maven, and Groovy and Kotlin languages. The Spring Boot version is set to 3.5.3 (selected).
- Project Metadata:** Fields for Group (com.todo), Artifact (todo), Name (todo), Description (Todo list en spring boot), and Package name (com.todo.todo). Packaging is set to Jar (selected) and War. Java version is set to 21 (selected).
- Dependencies:** A list of dependencies to be added to the project. The dependencies shown are Spring Web (WEB), Spring Data JPA (SQL), and MySQL Driver (SQL). Each dependency has a red circle next to it, indicating it is selected.

Clicamos en "generate" y se nos descargará un ZIP con el proyecto vacío ya creado.

## 2.- Creación de modelos

Para nuestra base de datos usaremos MySQL y HeidiSQL; previamente tenemos que crear los modelos para mapear nuestras tablas en la BD, y para ello crearemos 2 modelos: user y todo.

Dentro de src/main/java/com/todo/todo creamos la carpeta "modelos" y en ella creamos 3 ficheros .java: Address.java, Todo.java y User.java. Address es un objeto anidado en la clase User.



En el modelo de Todo incluiremos una validación para que el title no pueda ser mayor de 200 caracteres.

### 3.- Creación de Repositories

A continuación crearemos los Repositories, que son interfaces que comunican la BD con el controller que crearemos más adelante y nos permiten realizar las operaciones CRUD.

Los crearemos en la carpeta .../todo/repositorios, importando en ellos los respectivos modelos. Ambos tendrán este aspecto:

```
package com.todo.todo.repositorios;

//importamos el model del "todo":
import com.todo.todo.modelos.Todo;
//importamos la interfaz JpaRepository de Spring al proyecto:
import org.springframework.data.jpa.repository.JpaRepository;

//JpaRepository ya trae los métodos crud estándar; esto es para añadir consultas personalizadas, cosa que aquí no haremos:
public interface TodoRepository extends JpaRepository<Todo, Long> {

}
```

### 4.- Creación de controllers y DTOs

Ahora vamos a crear los controladores y los DTOs para nuestra app. Su ubicación, como los ficheros anteriores, será .../todo/, y crearemos un controlador y un DTO para cada entidad (user y todo), además de uno extra para cuando se crea el usuario en el que se incluye la password (en el DTO empleado para mostrar datos no se incluye por seguridad).

Dentro de los mismos implementaremos las operaciones CRUD habituales: get, post, etc.

Ver comentarios del controlador UserController.java y de UserDTO.java para más info y detalles.

Llegados a este punto, la estructura de carpetas de nuestro proyecto será la siguiente:



## 5.- Conexión con BBDD y testeo básico

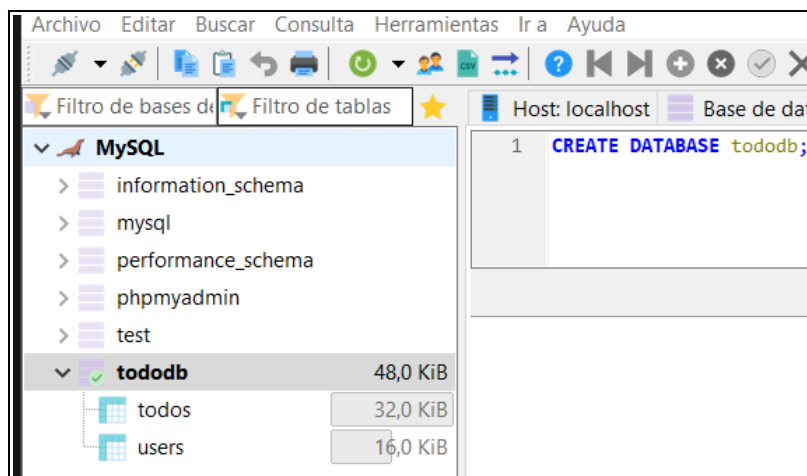
5.1.- Abrimos el archivo build.gradle, en la raíz del proyecto, y dentro de 'dependencies' añadimos el siguiente código para configurar la conexión con MySQL y para poder realizar tests en el futuro:

```
runtimeOnly 'mysql:mysql-connector-java'
testImplementation 'org.springframework.boot:spring-boot-starter-test'
```

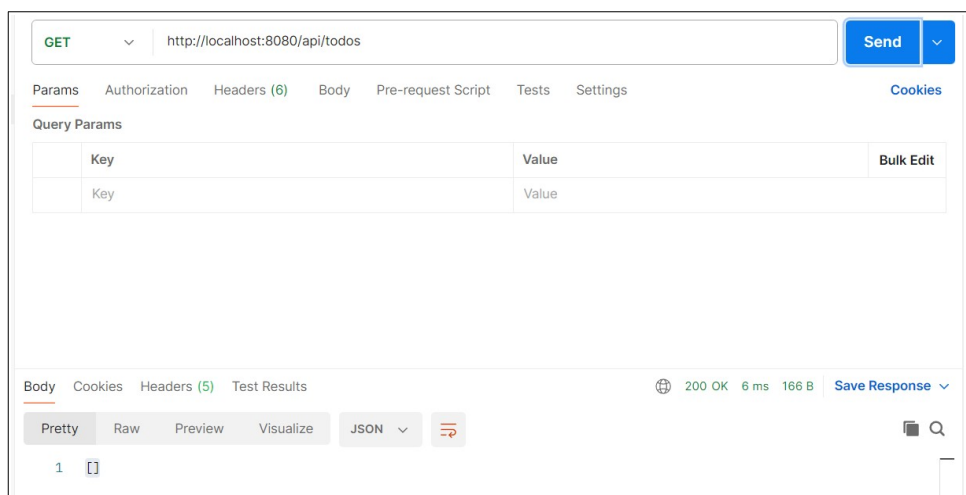
5.2.- Ahora editamos el fichero application.properties indicando la url de la conexión a mysql, nuestro usuario y la password.

5.3.- Ejecutamos el servidor local MySQL (en mi caso usaré XAMPP & HeidiSQL) y creamos la BD en HeidiSQL con el comando CREATE DATABASE tododb; dicho nombre, 'tododb', deberemos indicarlo en la URL en el archivo mencionado en el apartado 5.2.

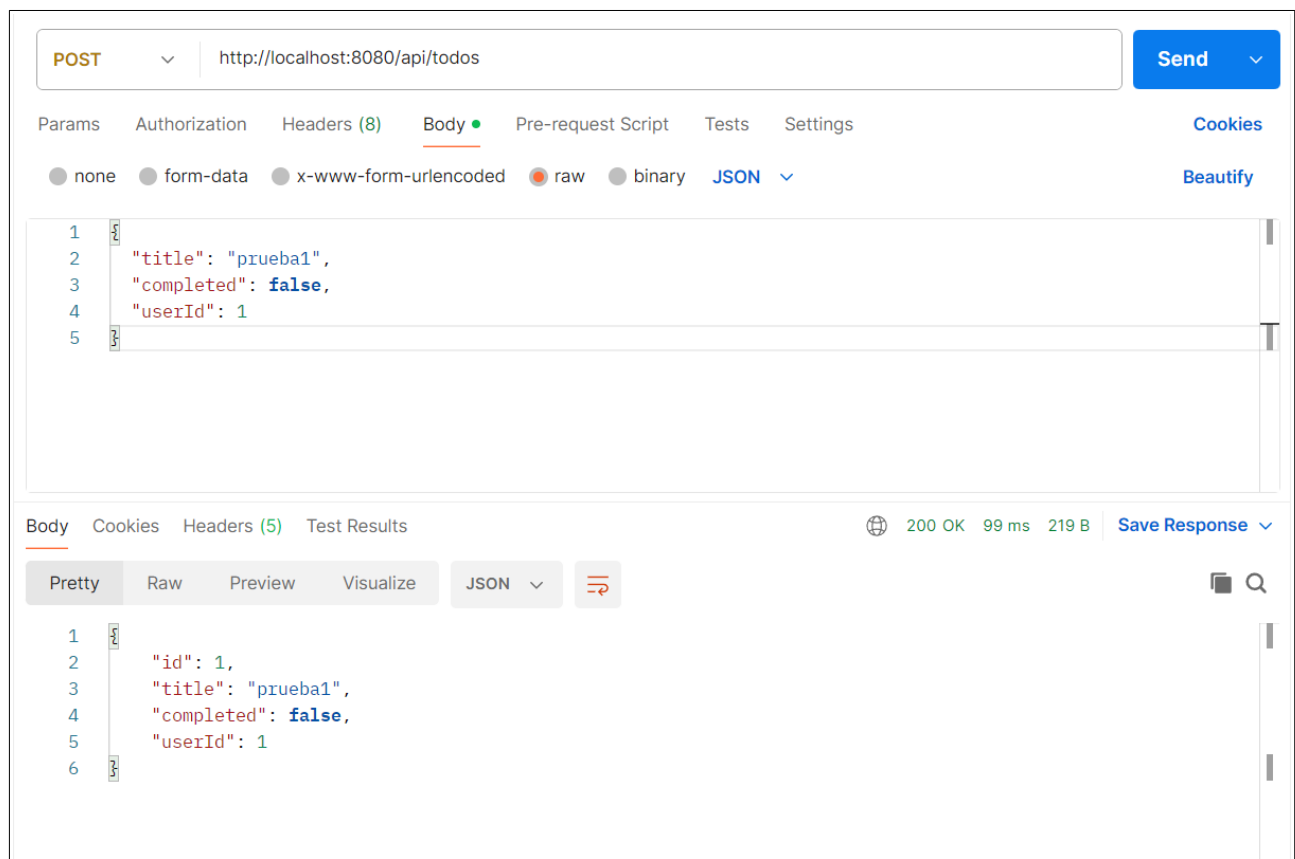
5.4.- Una vez creada la BD es hora de ejecutar el programa: nos dirigimos a TodoApplication.java, click derecho y "run java"; veremos como en la terminal nos aparece el mensaje de éxito, y en HeidiSQL veremos como, efectivamente, se ha rellenado la BD con las tablas users y todos:



5.5.- Ahora podemos realizar unos pequeños tests con Postman para confirmar que la app funciona correctamente, una vez ejecutada ésta y con el servidor local mySQL corriendo. Como podemos ver, se devuelve correctamente el GET con la lista vacía, pues todavía no tenemos elementos en ella:



Para probar los otros métodos previamente crearemos un usuario de prueba en Heidi, ya que es necesario para poder asignarlo a un Todo una vez lo creamos. Vemos que el POST también funciona correctamente, y lo mismo pasa con el delete y el PUT.



POST http://localhost:8080/api/todos

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

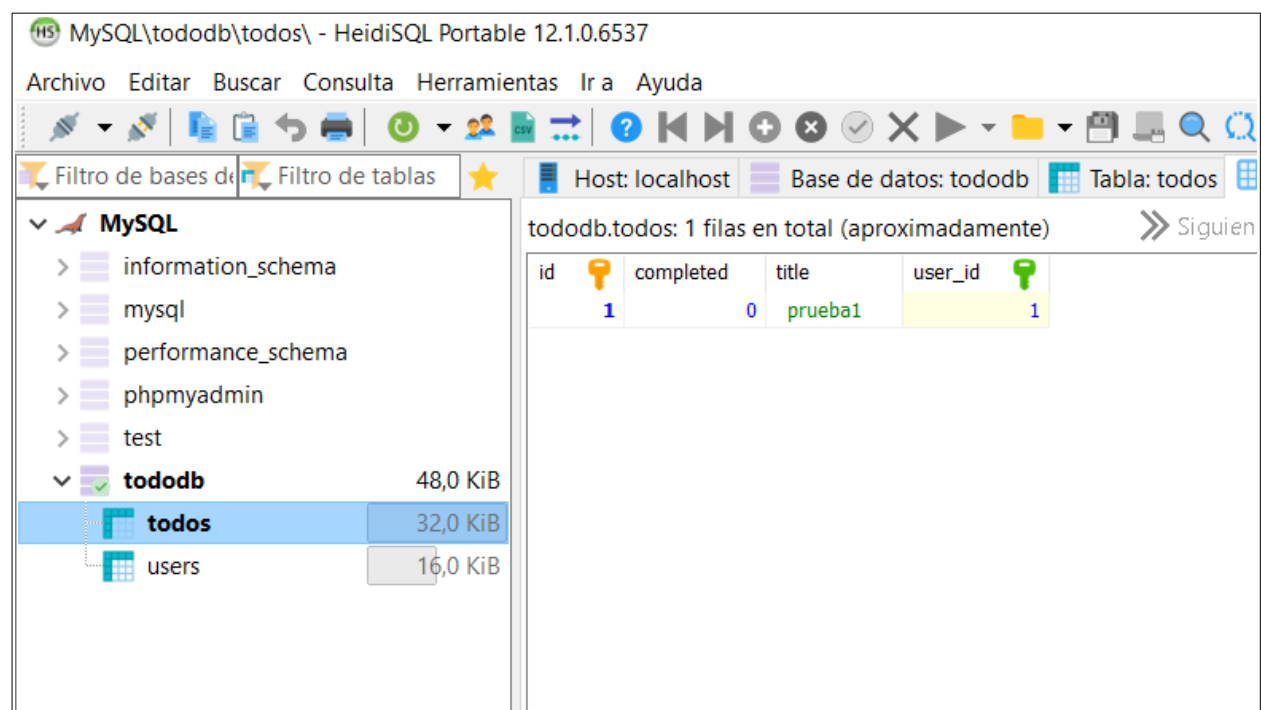
none form-data x-www-form-urlencoded raw binary JSON

```
1 {
2   "title": "prueba1",
3   "completed": false,
4   "userId": 1
5 }
```

Body Cookies Headers (5) Test Results 200 OK 99 ms 219 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "title": "prueba1",
4   "completed": false,
5   "userId": 1
6 }
```



MySQL\tododb\todos\ - HeidiSQL Portable 12.1.0.6537

Archivo Editar Buscar Consulta Herramientas Ir a Ayuda

Filtro de bases de datos Filtro de tablas Host: localhost Base de datos: tododb Tabla: todos

MySQL

- information\_schema
- mysql
- performance\_schema
- phpmyadmin
- test
- tododb** 48,0 KiB
  - todos** 32,0 KiB
  - users 16,0 KiB

tododb.todos: 1 filas en total (aproximadamente)

id	completed	title	user_id
1	0	prueba1	1

Así pues, una vez comprobado que el programa funciona y responde a las llamadas, procederemos a crear un front sencillo con Thymeleaf para crear las vistas de listado, edición y

adición de nuevos elementos.

## 6.- Front

Ahora procedemos a crear las vistas para nuestro programa. Habrá 2:

- vista principal en la que se verá un listado con todos los TODOs.
- vista de creación y edición de TODOs con un formulario simple.

Las mismas las crearemos con Thymeleaf, siguiendo las instrucciones proporcionadas, y con un css sencillo para centrarlo todo y que sea presentable. En el formulario establecemos un select para seleccionar el usuario al que se le asigna el TODO y un checkbox para indicar si la tarea está completada o no.

En la página principal del listado hemos hecho que cada columna sea ordenable alfabéticamente con esta línea en cada campo:

```
<a th:href="@{/todos?page=0&sortBy=id&sortDir=${sortBy == 'id' && sortDir == 'asc' ? 'desc' : 'asc'}}">
```

Para enrutar las vistas con nuestro backend usaremos un nuevo controlador: `TodoViewController`, donde estableceremos las rutas de nuestra app.

## 7.- Tests

Nuestro test unitario de ejemplo, para el modelo `Todo`, lo crearemos en `src/test/java/com/todo/todo/modelos`.

Introducimos en consola: `./gradlew test` para ejecutar nuestro nuevo test.

Veremos el resultado en la propia consola:

```
TodoApplicationTests > contextLoads() PASSED

TodoTest > testTodoConUsuario() STANDARD_OUT
Test exitoso

TodoTest > testTodoConUsuario() PASSED

BUILD SUCCESSFUL in 5s
4 actionable tasks: 2 executed, 2 up-to-date
PS C:\Users\obipp\Desktop\todo>
```

Y también podemos chequear los resultados en la carpeta `.../build/reports/tests/test/index.html`.

## 8.- Ejecución de la app

Para crear esta app he usado VSC y windows 10. El enlace de GitHub:

<https://github.com/FERLOPEZNUNO/Todo-list-Spring>

Una vez descargado y abierto el repo, para iniciar la app introducimos en la consola:

```
./gradlew clean build  
./gradlew bootRun
```

Y en nuestro navegador la URL correspondiente:

<http://localhost:8080/todos>