



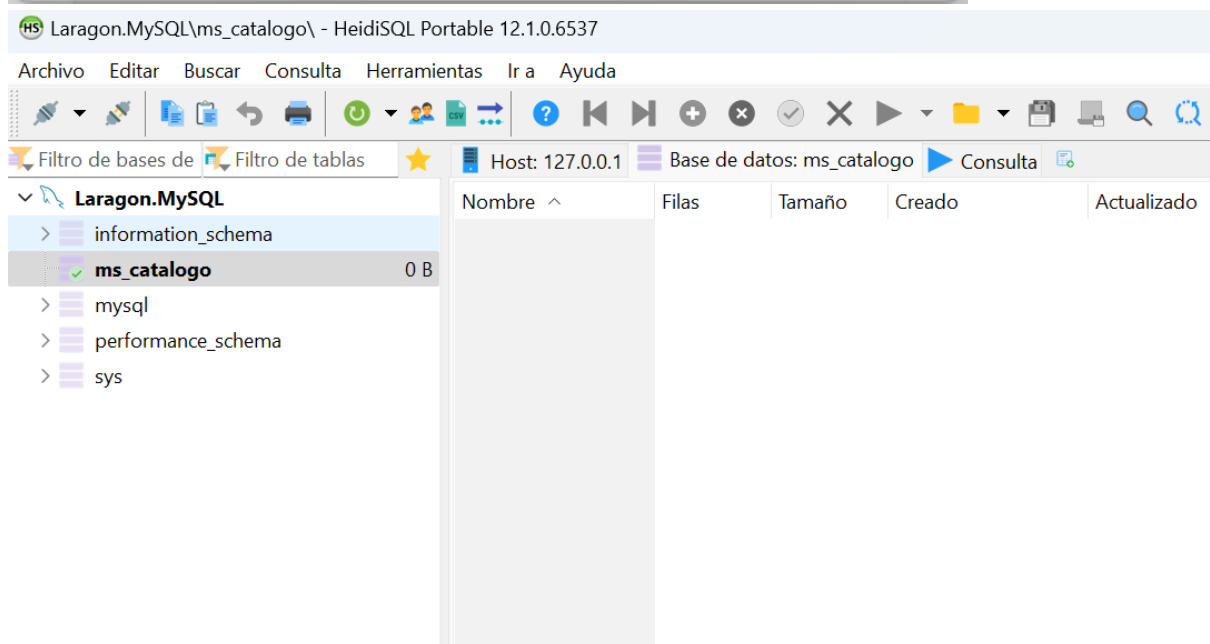
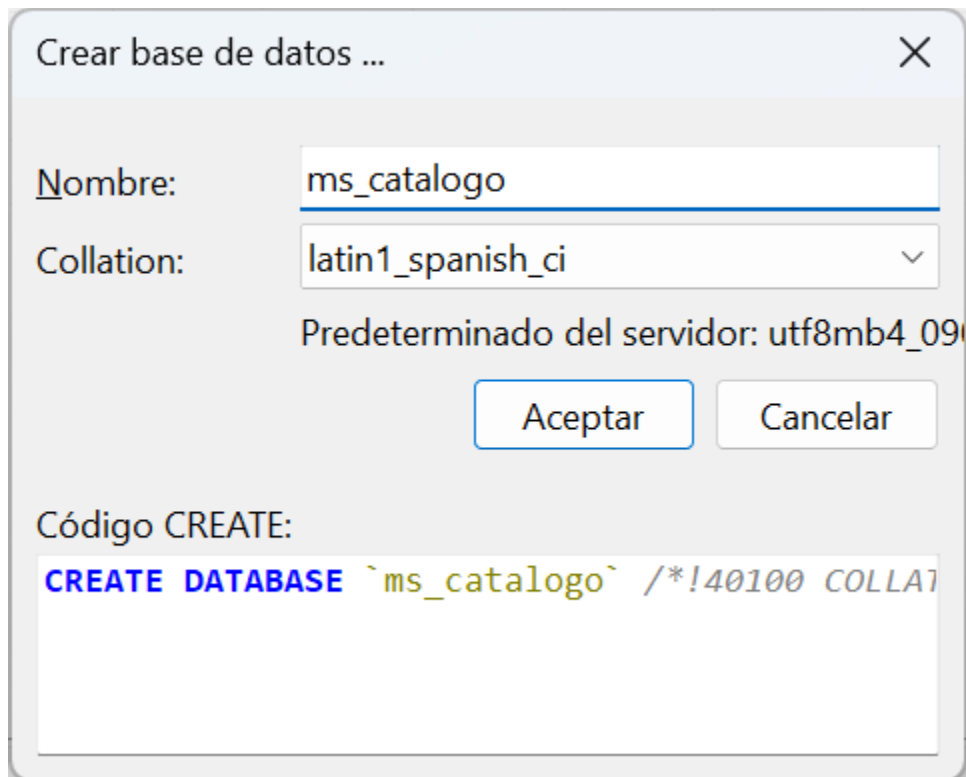
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS
CURSO: DESARROLLO DE APLICACIONES DISTRIBUIDAS

DESARROLLA UN CRUD CON SPRING BOOT

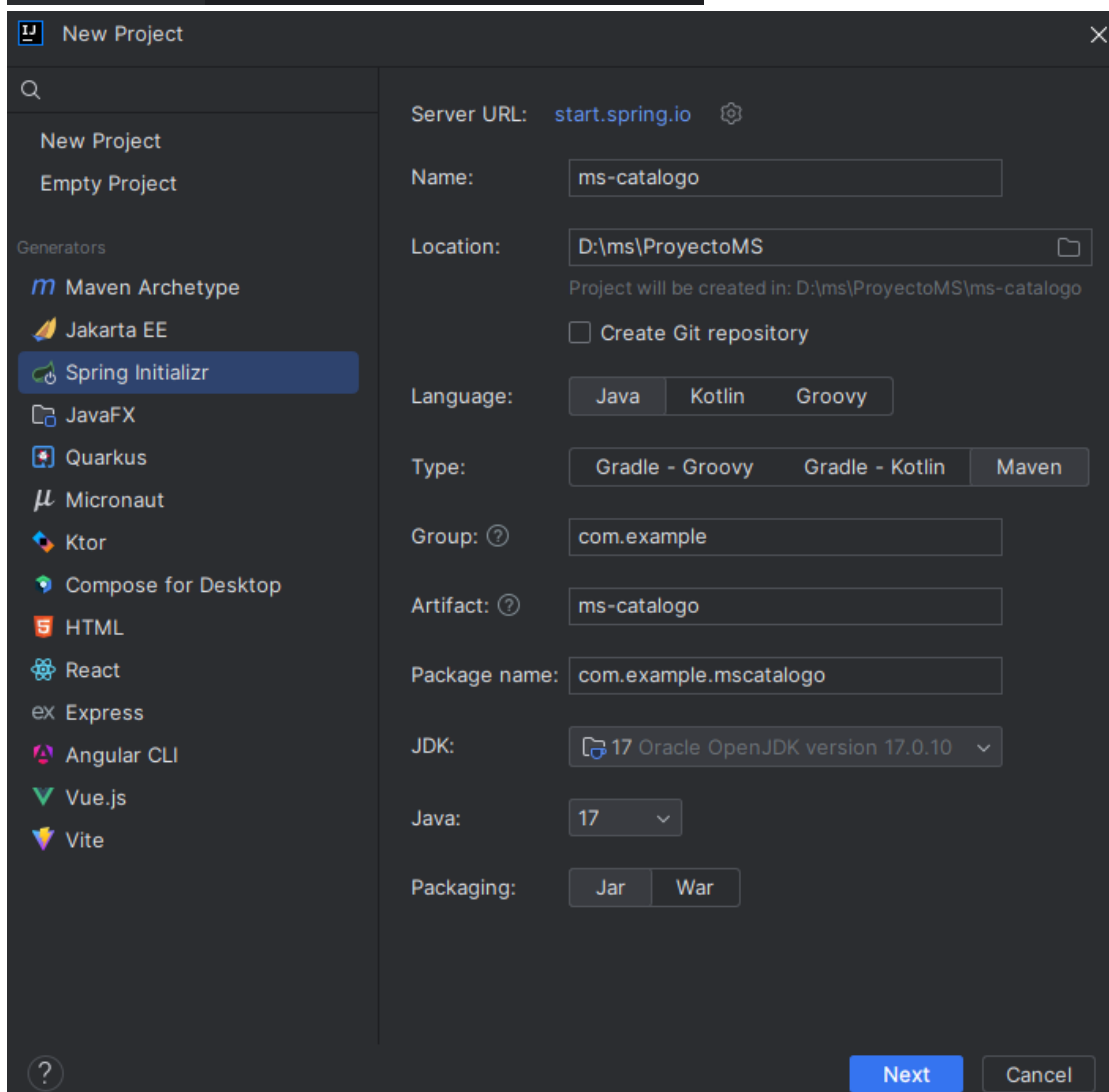
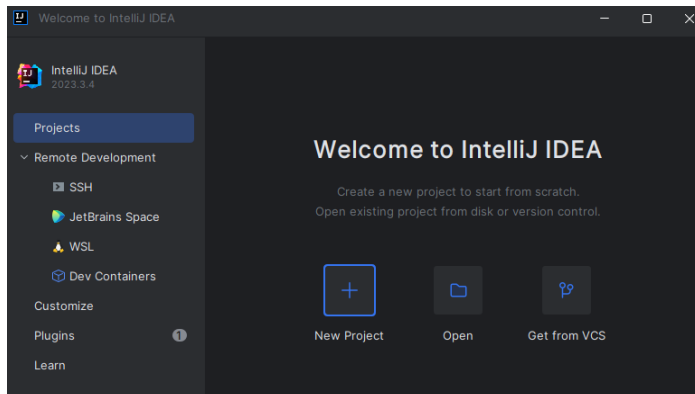
ESTUDIANTE:
ANTHONY KELMAN MAMANI VARGAS

DOCENTE:
NOE WILBER TIPO MAMANI

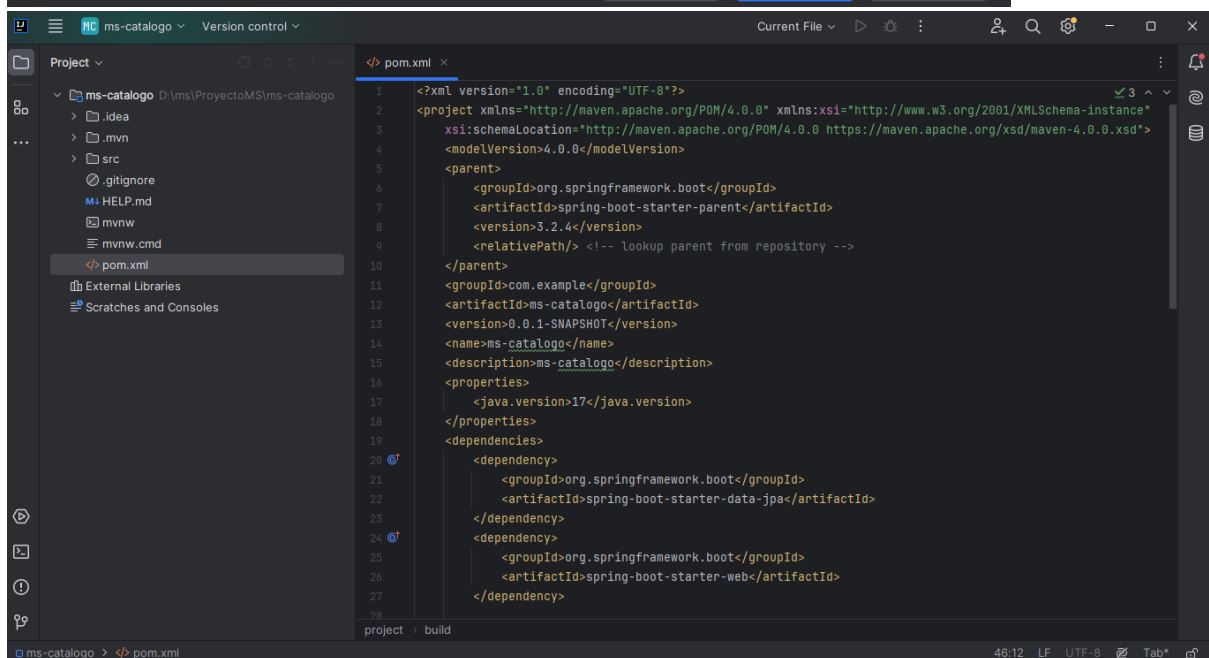
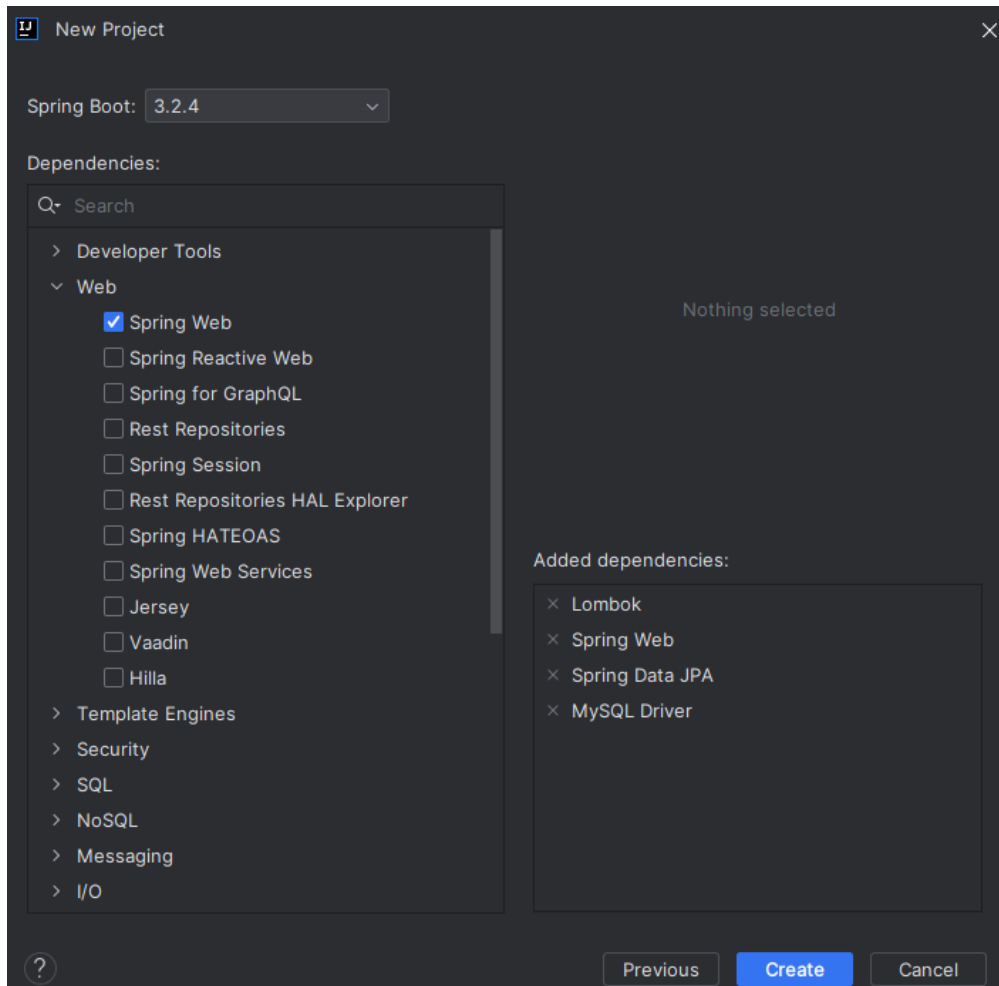
Crear una base de datos en MySQL








Crear el proyecto: ms-catalogo

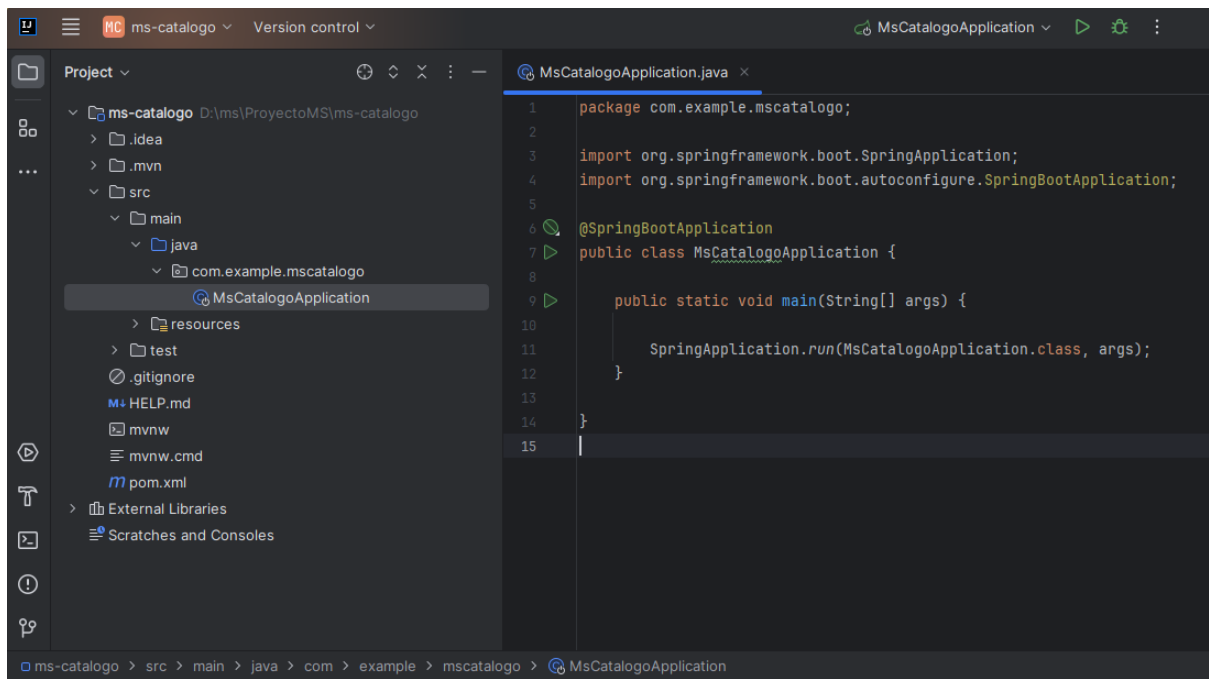


Name: ms-catalogo



Recordar estas dependencias iniciales

```
</> pom.xml x
28
29  <dependency>
30     <groupId>com.mysql</groupId>
31     <artifactId>mysql-connector-j</artifactId>
32     <scope>runtime</scope>
33 </dependency>
34  <dependency>
35     <groupId>org.projectlombok</groupId>
36     <artifactId>lombok</artifactId>
37     <optional>true</optional>
38 </dependency>
39  <dependency>
40     <groupId>org.springframework.boot</groupId>
41     <artifactId>spring-boot-starter-test</artifactId>
42     <scope>test</scope>
43  </dependency>
44 </dependencies>
45
46 <build>
47     <plugins>
48         <plugin>
49             <groupId>org.springframework.boot</groupId>
50  <artifactId>spring-boot-maven-plugin</artifactId>
51             <configuration>
52                 <excludes>
53                     <exclude>
54                         <groupId>org.projectlombok</groupId>
55                         <artifactId>lombok</artifactId>
56                     </exclude>
57                 </excludes>
58             </configuration>
59         </plugin>
60     </plugins>
61 </build>
62
63 </project>
64
```



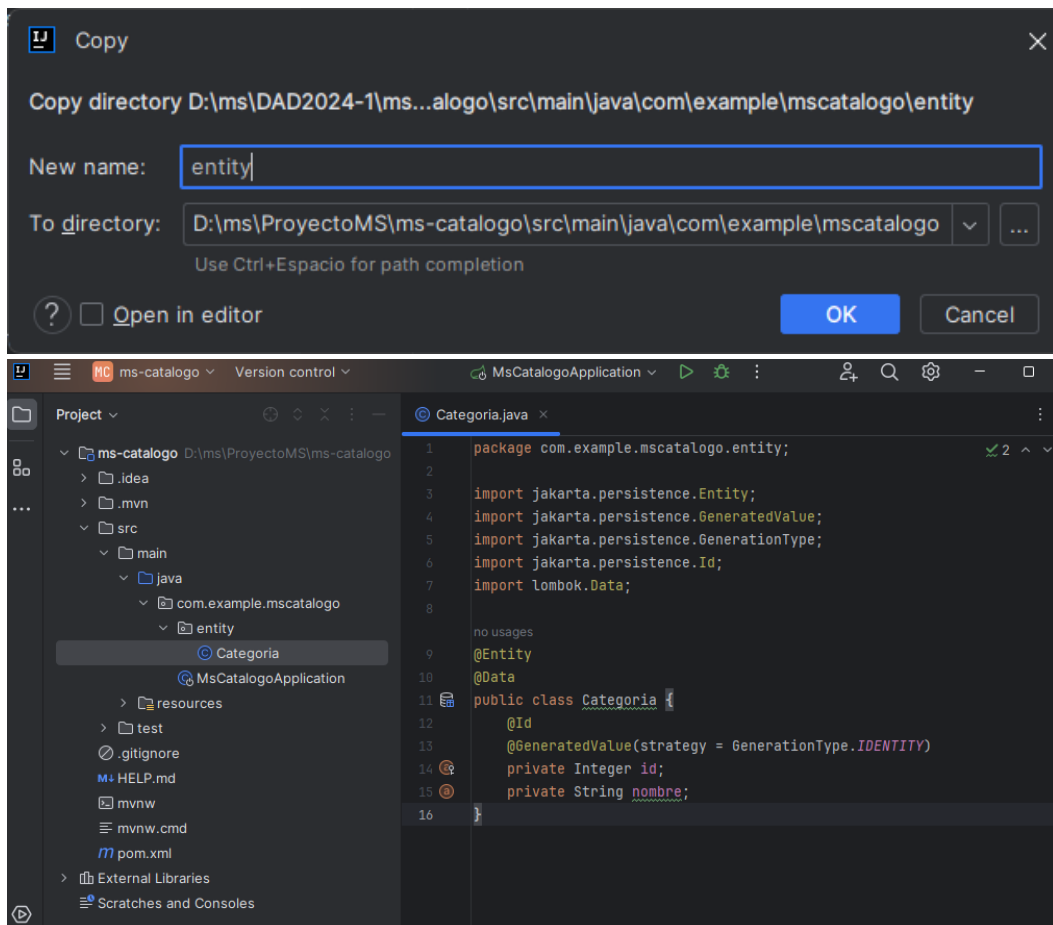
Entity

Una entidad (Entity) es una clase Java que representa una tabla en una base de datos relacional. Esta clase se mapea directamente a una tabla específica en la base de datos, y cada instancia de la clase representa una fila en esa tabla.

```
package com.example.mscatalogo.entity;

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

@Entity
@Data
public class Categoria {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String nombre;
}
```

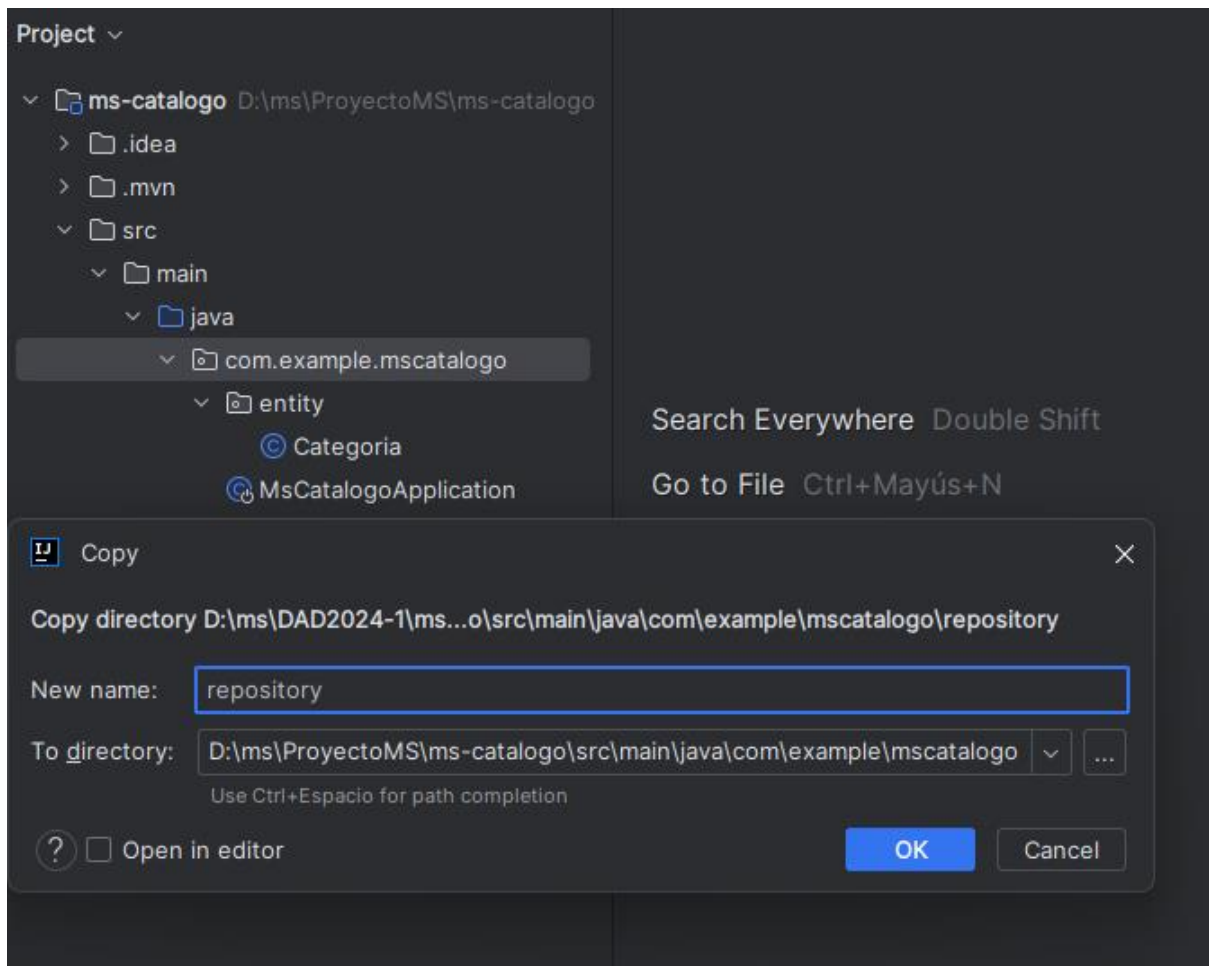


Repository

```
package com.example.mscatalogo.repository;

import com.example.mscatalogo.entity.Categoria;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CategoriaRepository extends JpaRepository<Categoria, Integer> {
}
```

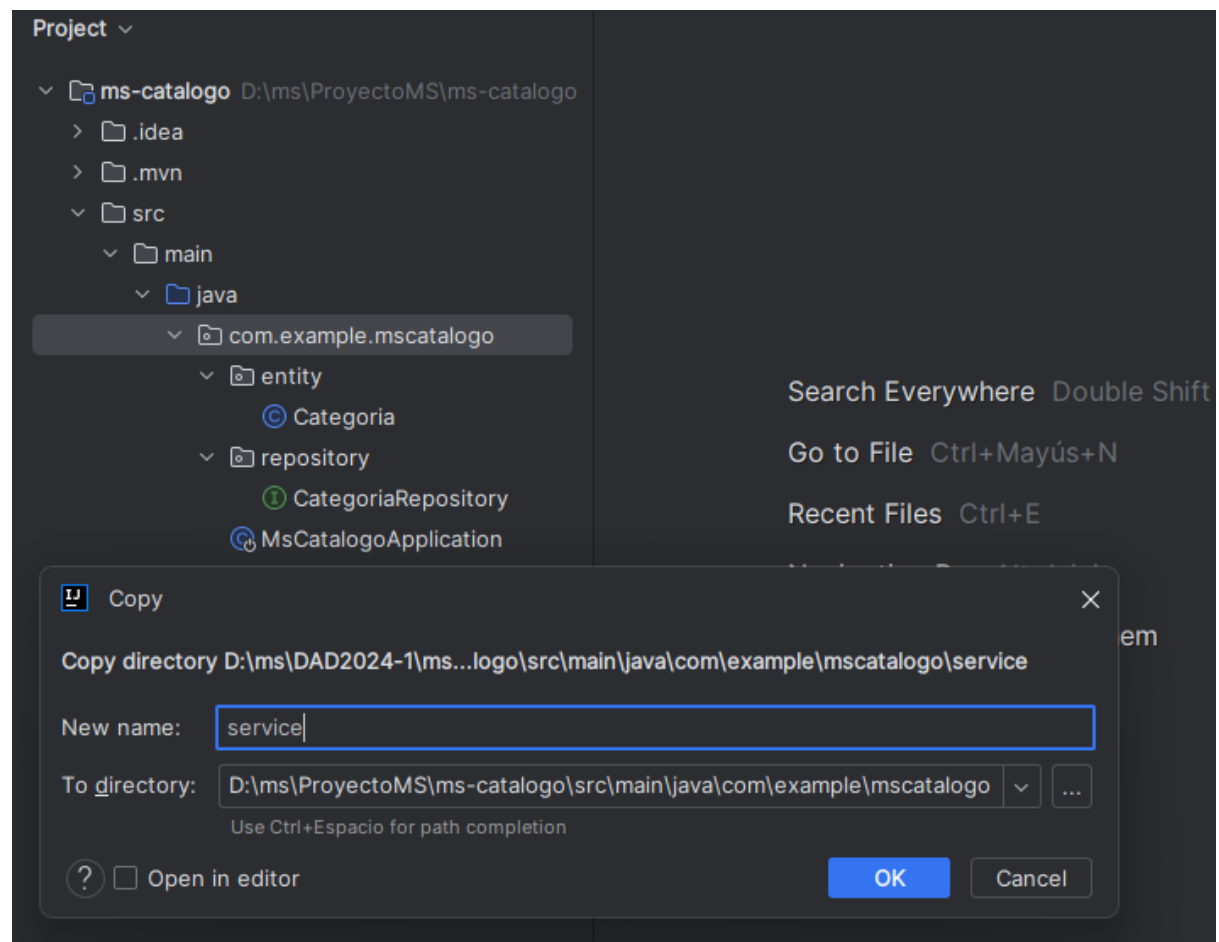
Service

```
package com.example.mscatalogo.service;

import com.example.mscatalogo.entity.Categoria;

import java.util.List;
import java.util.Optional;

public interface CategoriaService {
    public List<Categoria> listar();
    public Categoria guardar(Categoria categoria);
    public Categoria actualizar(Categoria categoria);
    public Optional<Categoria> listarPorId(Integer id);
    public void eliminarPorId(Integer id);
}
```



ServiceImpl

```
package com.example.mscatalogo.service.impl;

import com.example.mscatalogo.entity.Categoria;
import com.example.mscatalogo.repository.CategoriaRepository;
import com.example.mscatalogo.service.CategoriaService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class CategoriaServiceImpl implements CategoriaService {
    @Autowired
    private CategoriaRepository categoriaRepository;

    @Override
    public List<Categoria> listar() {
        return categoriaRepository.findAll();
    }

    @Override
    public Categoria guardar(Categoria categoria) {
        return categoriaRepository.save(categoria);
    }

    @Override
    public Categoria actualizar(Categoria categoria) {
        return categoriaRepository.save(categoria);
    }

    @Override
    public Optional<Categoria> listarPorId(Integer id) {
        return categoriaRepository.findById(id);
    }

    @Override
    public void eliminarPorId(Integer id) {
        categoriaRepository.deleteById(id);
    }
}
```

Controller

```
package com.example.mscatalogo.controller;

import com.example.mscatalogo.entity.Categoria;
import com.example.mscatalogo.service.CategoriaService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/categoria")
public class CategoriaController {

    @Autowired
    private CategoriaService categoriaService;

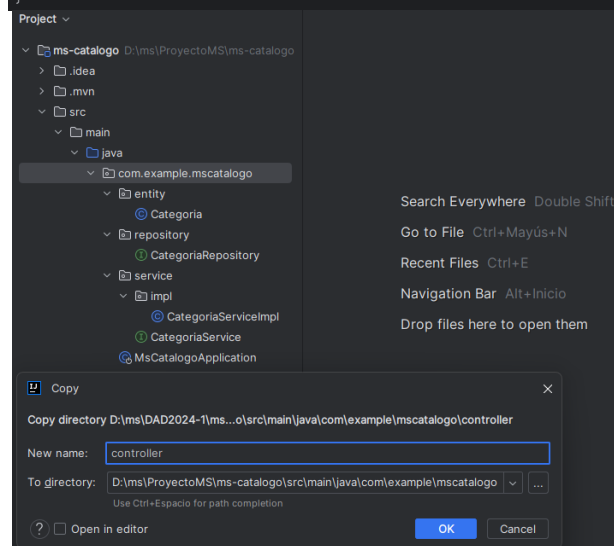
    @GetMapping()
    public ResponseEntity<List<Categoria>> list() {
        return ResponseEntity.ok().body(categoriaService.listar());
    }

    @PostMapping()
    public ResponseEntity<Categoria> save(@RequestBody Categoria categoria){
        return ResponseEntity.ok(categoriaService.guardar(categoria));
    }

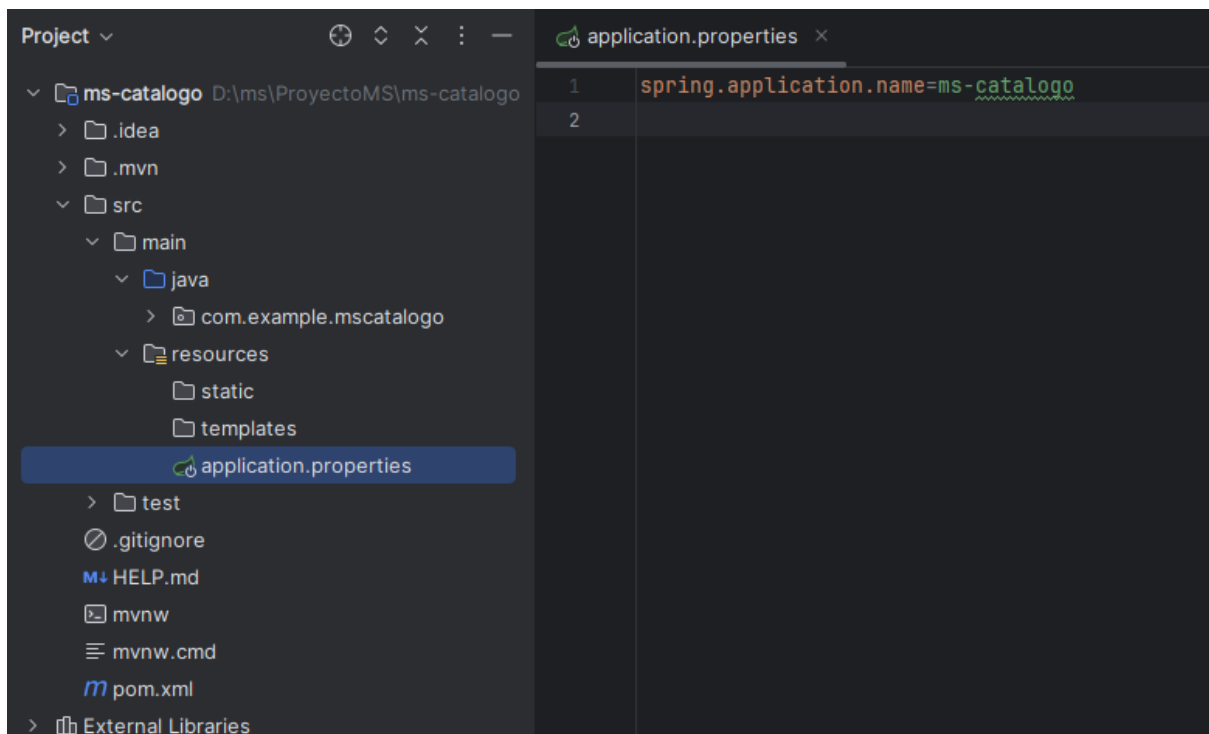
    @PutMapping()
    public ResponseEntity<Categoria> update(@RequestBody Categoria categoria){
        return ResponseEntity.ok(categoriaService.actualizar(categoria));
    }

    @GetMapping("/{id}")
    public ResponseEntity<Categoria> listById(@PathVariable(required = true) Integer id){
        return ResponseEntity.ok().body(categoriaService.listarPorId(id).get());
    }

    @DeleteMapping("/{id}")
    public String deleteById(@PathVariable(required = true) Integer id){
        categoriaService.eliminarPorId(id);
        return "Eliminacion Correcta";
    }
}
```

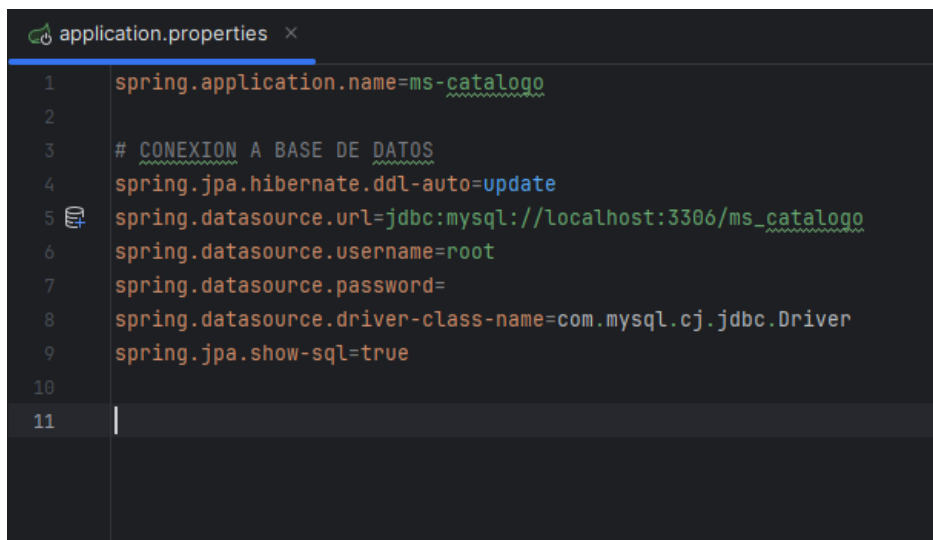


Conexión a base datos

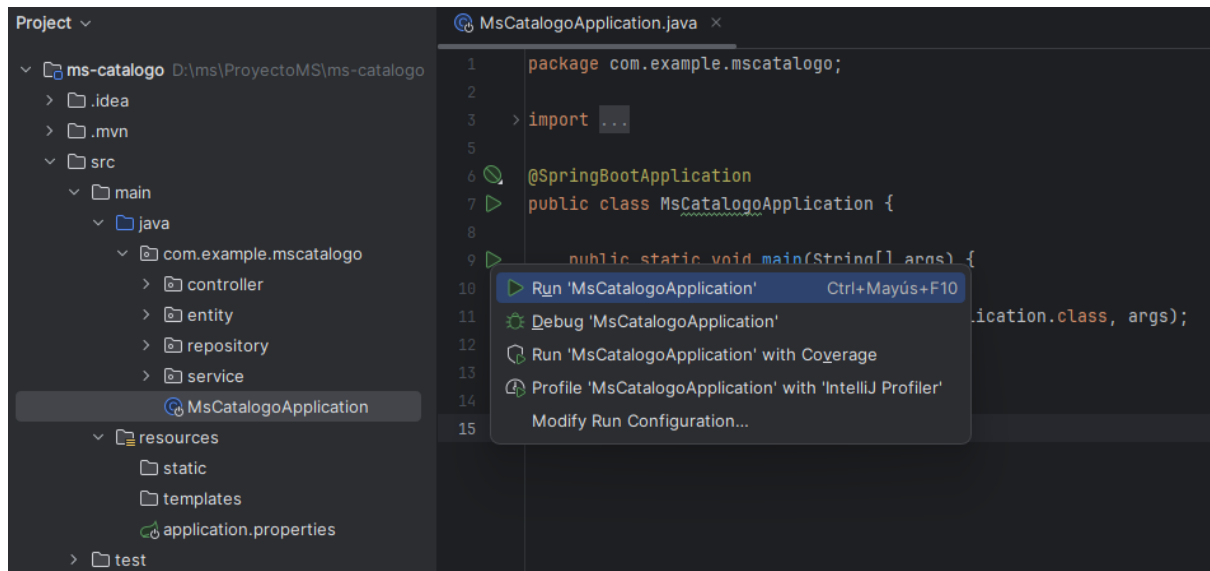


```
spring.application.name=ms-catalogo

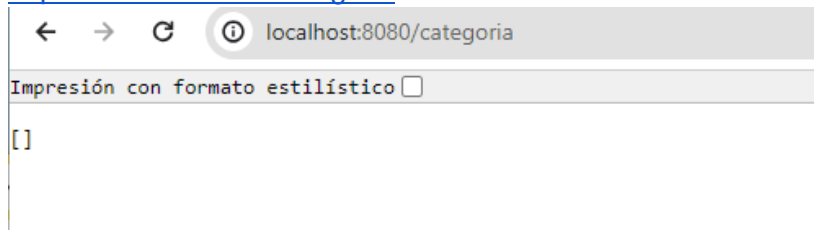
# CONEXION A BASE DE DATOS
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/ms_catalogo
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql=true
```



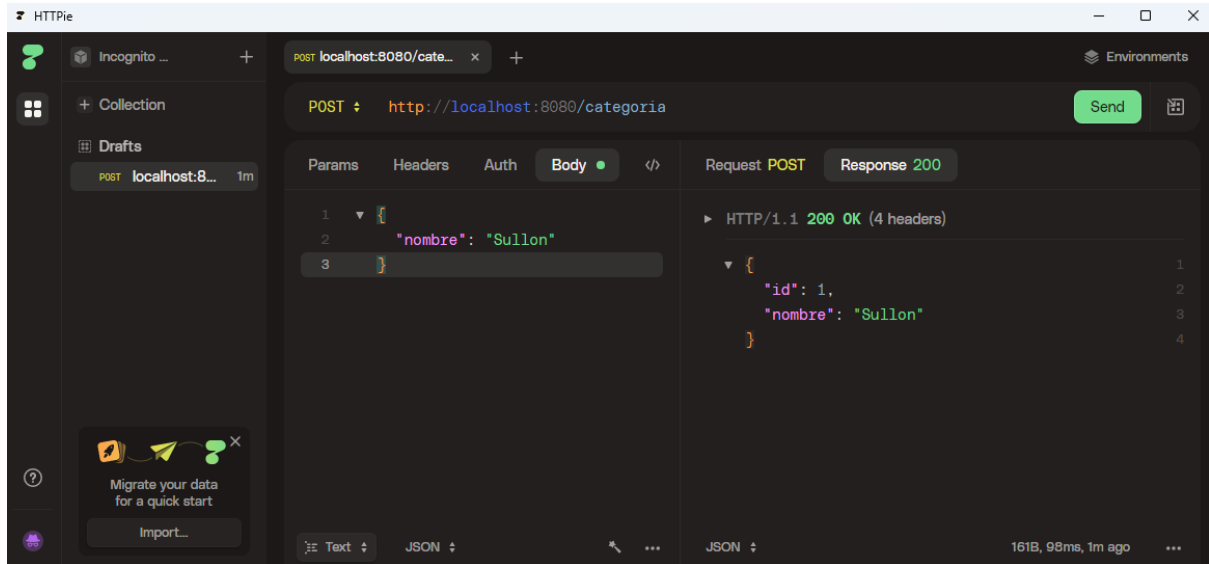
Ejecutamos



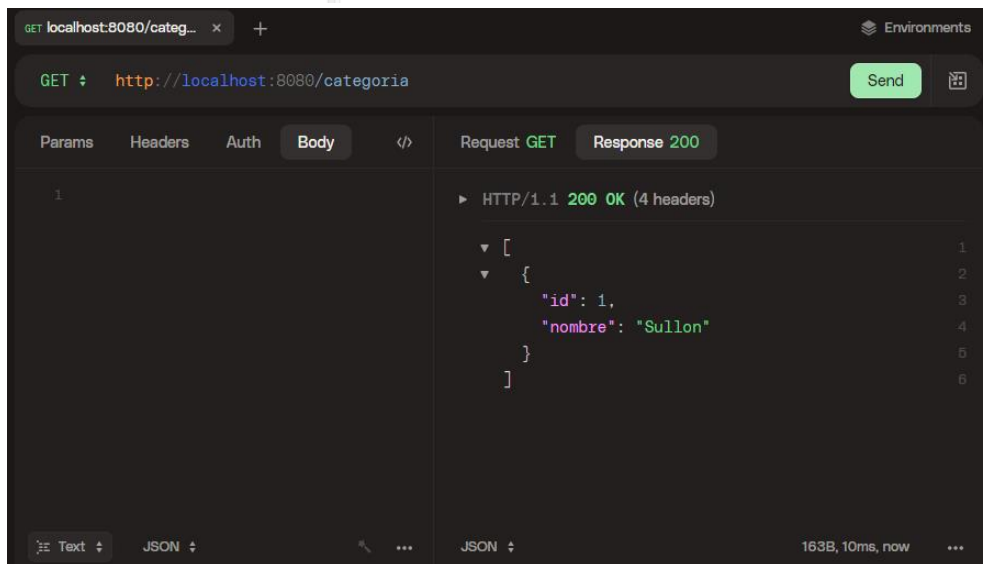
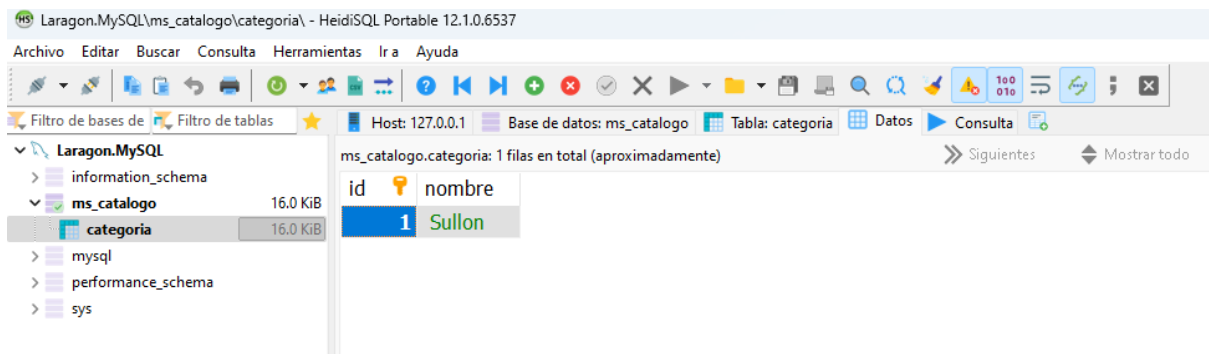
<http://localhost:8080/categoria>



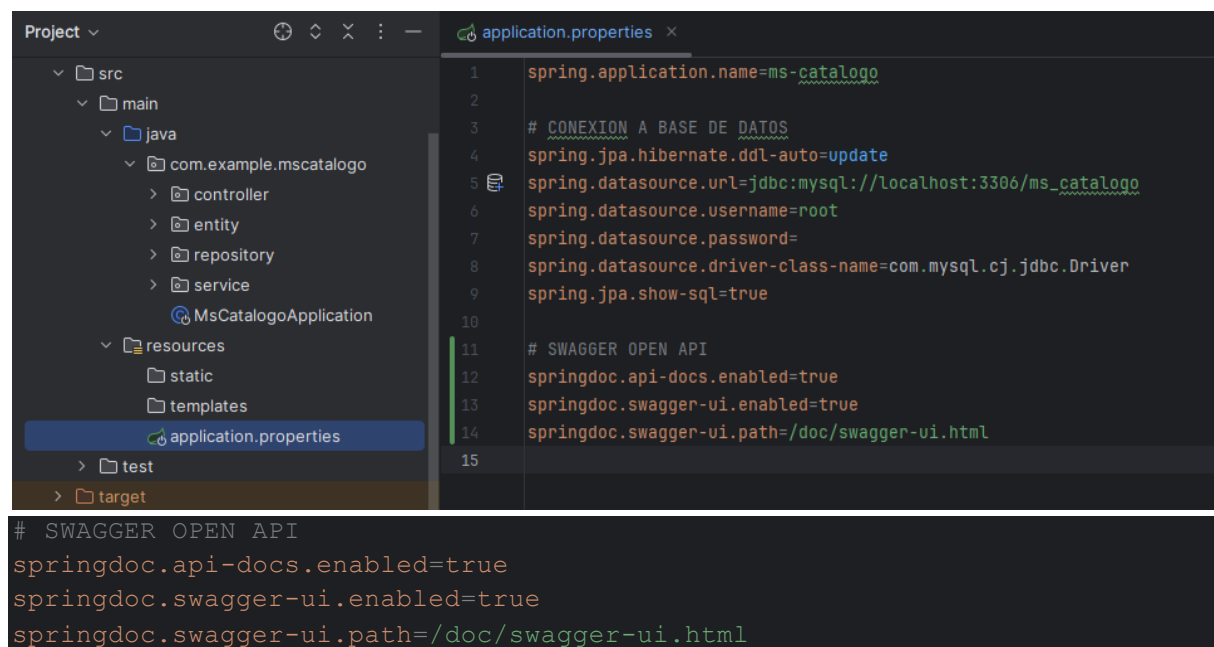
Testing



```
{
  "nombre": "Sullon"
}
```



Luego:



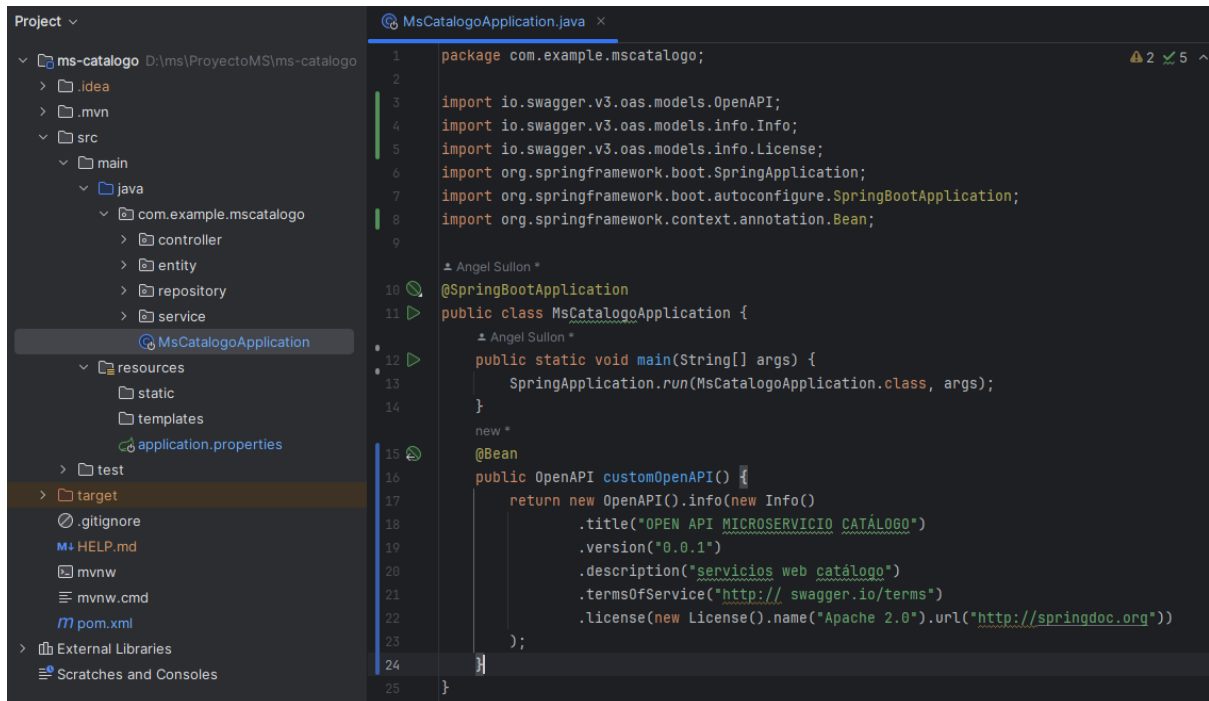
The screenshot shows an IDE with a project named 'ms-catalogo'. The left sidebar displays the project structure, including 'src/main/java/com.example.ms-catalogo' with sub-packages 'controller', 'entity', 'repository', and 'service', and a file 'MsCatalogoApplication'. The 'resources' folder contains 'static', 'templates', and 'application.properties'. The 'application.properties' file is open in the editor, showing the following configuration:

```
1 spring.application.name=ms-catalogo
2
3 # CONEXION A BASE DE DATOS
4 spring.jpa.hibernate.ddl-auto=update
5 spring.datasource.url=jdbc:mysql://localhost:3306/ms-catalogo
6 spring.datasource.username=root
7 spring.datasource.password=
8 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
9 spring.jpa.show-sql=true
10
11 # SWAGGER OPEN API
12 springdoc.api-docs.enabled=true
13 springdoc.swagger-ui.enabled=true
14 springdoc.swagger-ui.path=/doc/swagger-ui.html
15
```

Below the editor, the Swagger Open API configuration is repeated:

```
# SWAGGER OPEN API
springdoc.api-docs.enabled=true
springdoc.swagger-ui.enabled=true
springdoc.swagger-ui.path=/doc/swagger-ui.html
```


Configurar Bean: MsCatalogoApplication



```
1 package com.example.ms-catalogo;
2
3 import io.swagger.v3.oas.models.OpenAPI;
4 import io.swagger.v3.oas.models.info.Info;
5 import io.swagger.v3.oas.models.info.License;
6 import org.springframework.boot.SpringApplication;
7 import org.springframework.boot.autoconfigure.SpringBootApplication;
8 import org.springframework.context.annotation.Bean;
9
10 @SpringBootApplication
11 public class MsCatalogoApplication {
12     public static void main(String[] args) {
13         SpringApplication.run(MsCatalogoApplication.class, args);
14     }
15
16     @Bean
17     public OpenAPI customOpenAPI() {
18         return new OpenAPI().info(new Info()
19             .title("OPEN API MICROSERVICIO CATÁLOGO")
20             .version("0.0.1")
21             .description("servicios web catálogo")
22             .termsOfService("http:// swagger.io/terms")
23             .license(new License().name("Apache 2.0").url("http://springdoc.org"))
24         );
25     }
26 }
```

```
package com.example.ms-catalogo;

import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.info.License;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class MsCatalogoApplication {
    public static void main(String[] args) {
        SpringApplication.run(MsCatalogoApplication.class, args);
    }

    @Bean
    public OpenAPI customOpenAPI() {
        return new OpenAPI().info(new Info()
            .title("OPEN API MICROSERVICIO CATÁLOGO")
            .version("0.0.1")
            .description("servicios web catálogo")
            .termsOfService("http:// swagger.io/terms")
            .license(new License().name("Apache 2.0").url("http://springdoc.org"))
        );
    }
}
```