

Microservicios

Nelson Jose Luque Mamani

Link: <https://github.com/ZethGecko/DAD---01>

Preparación de ambiente de desarrollo

1. **Instalación de Laragon**
2. **Instalación y configuración de Git**
3. **Instalación de Java 17 o 21**
4. **Instalación de IntelliJ idea**
5. **Instalación de HTTPie/Postman**

Creación de la base de datos:

Crear base de datos ...

Nombre:

Collation:

Predeterminado del servidor: utf8mb4_0900_ai_ci

Código CREATE:

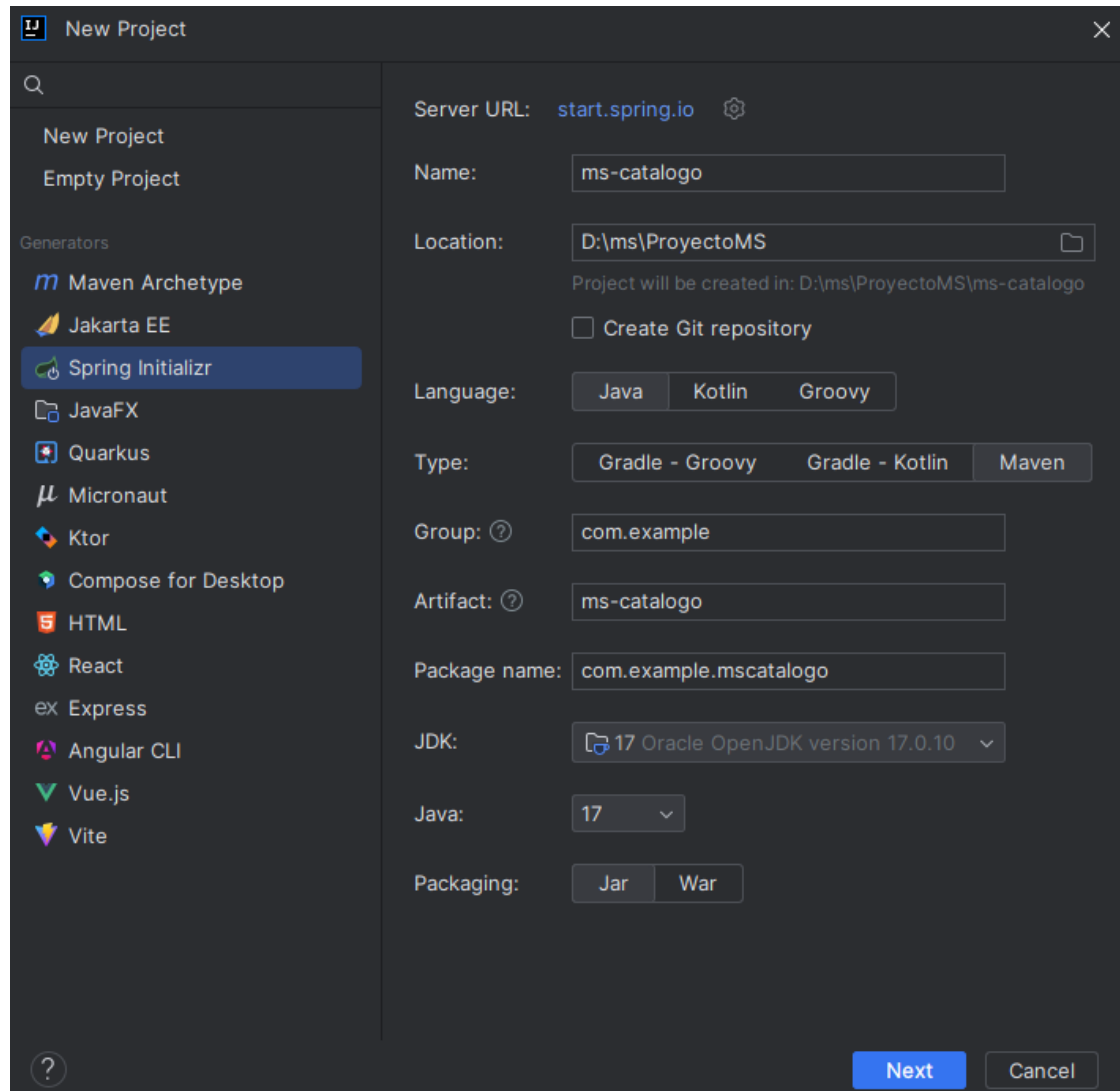
```
CREATE DATABASE `ms_catalogo` /*!40100 COLLAT
```

Creación de del proyecto: ms-catalogo

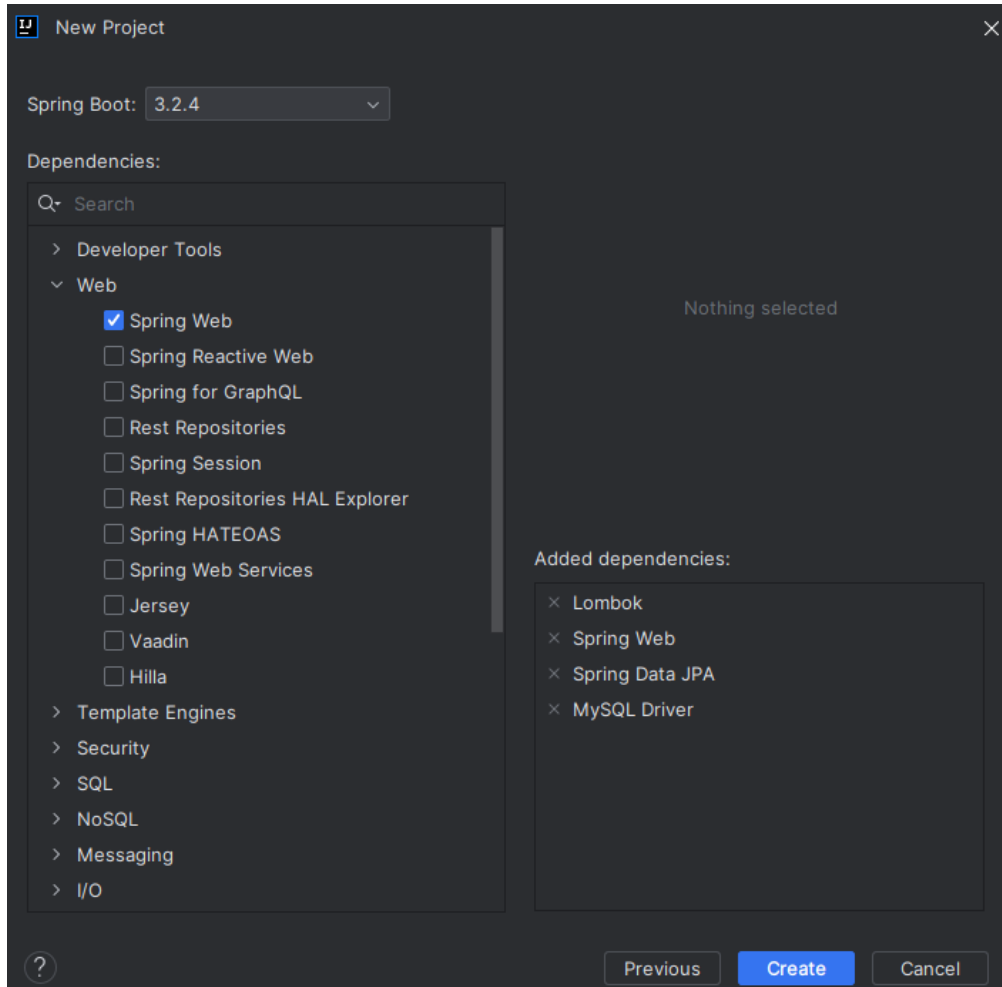
Nombre: ms-catalogo

Localización: D:\ms\ProyectoMS

Tipo: Maven








Seleccionar las librerías necesarias:

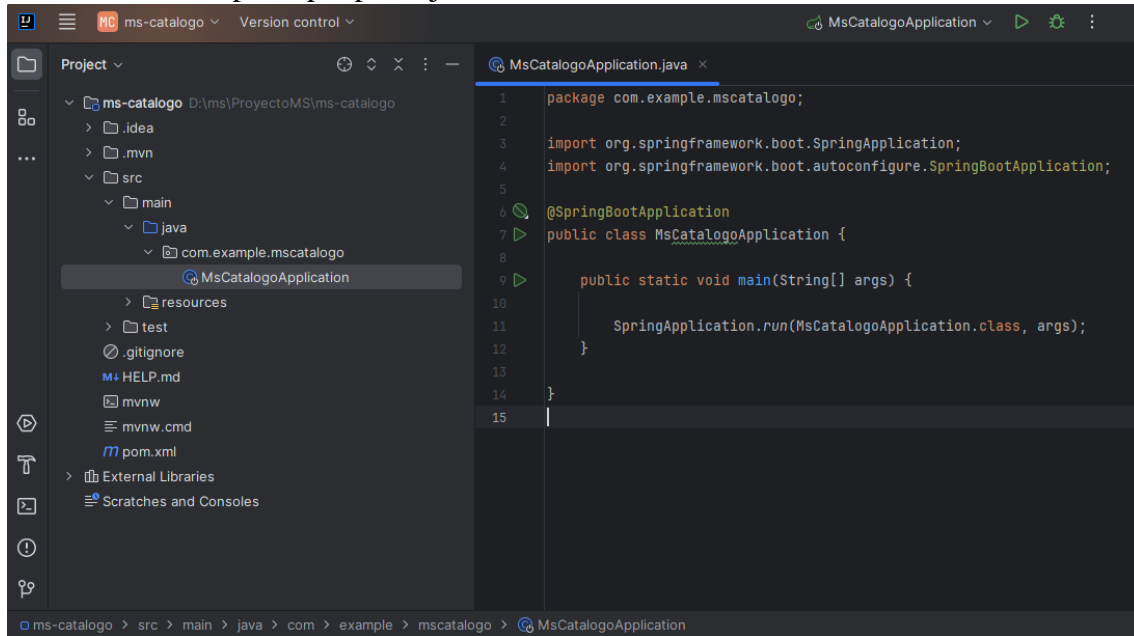


Comprobar las dependencias antes de iniciar en el POM.xml

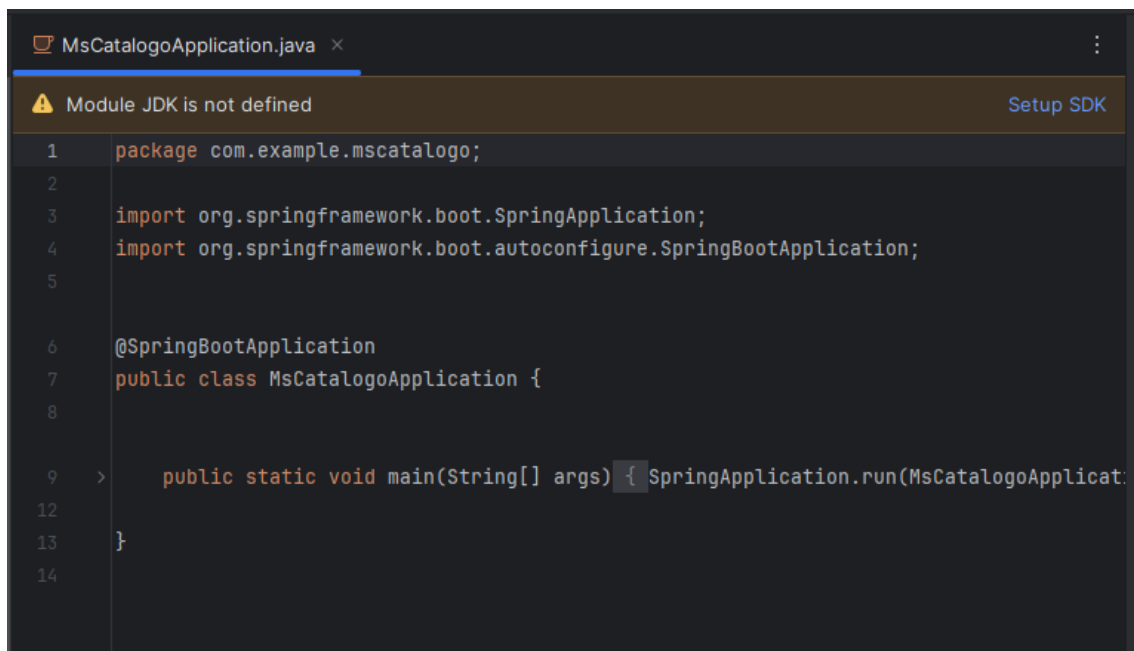
</> pom.xml x

```
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
```

Habilitar la clase principal para ejecutarlo:

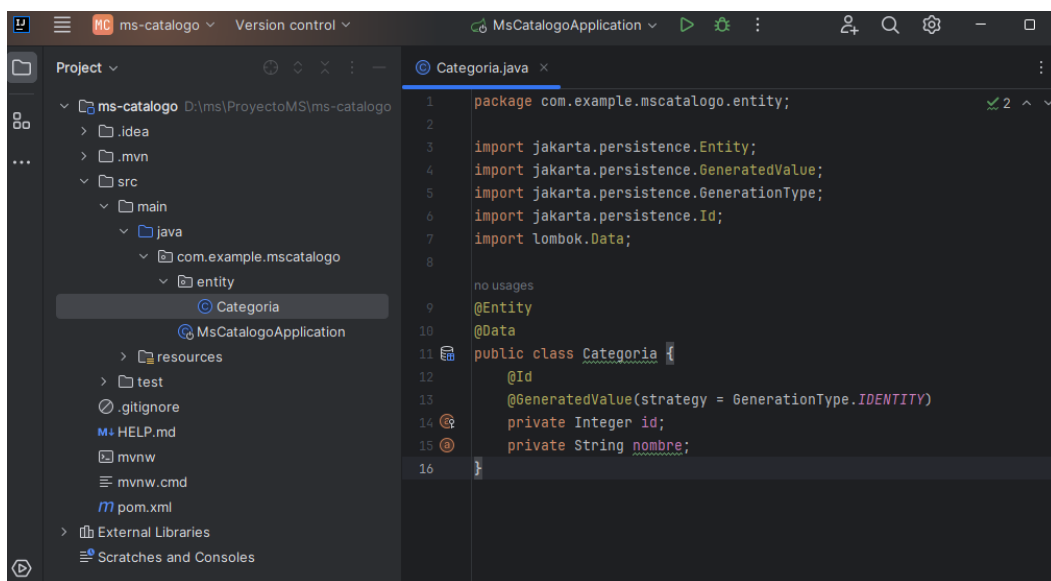
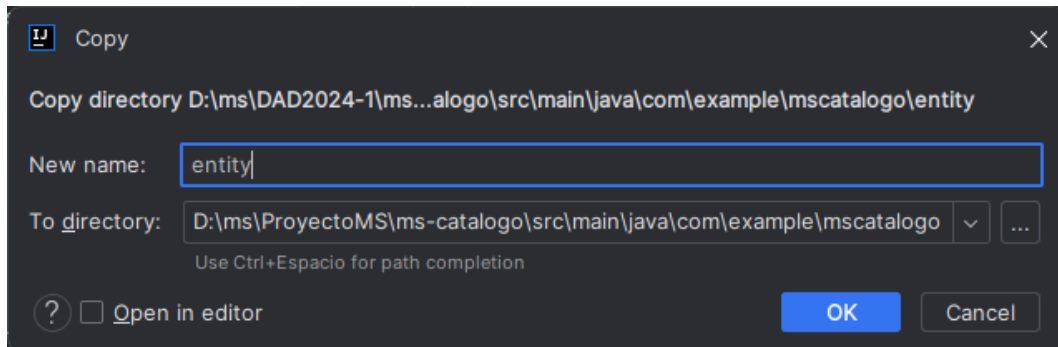


asegurarse de actualizar el IDE



Entity

Realizar la entidad clase java para representar la tabla en la base de datos, en este caso llamada catalogo:



```
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 {

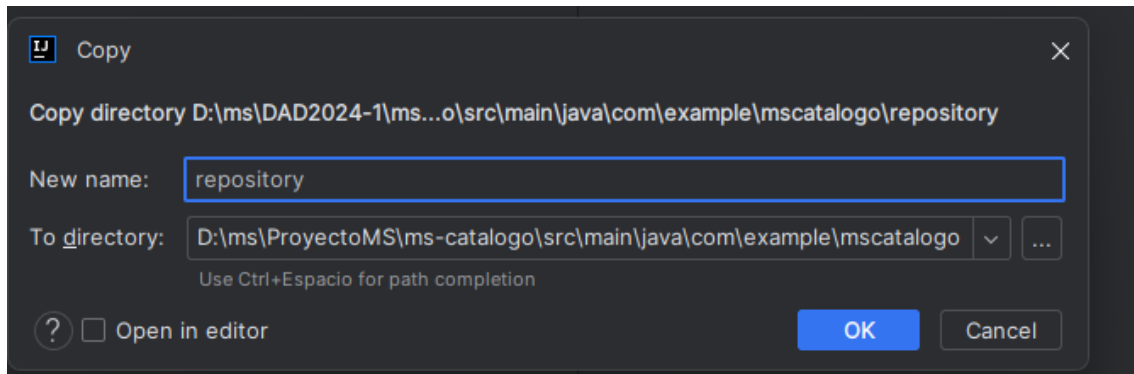
    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
```

```
private String nombre;  
}
```

Repositorio:

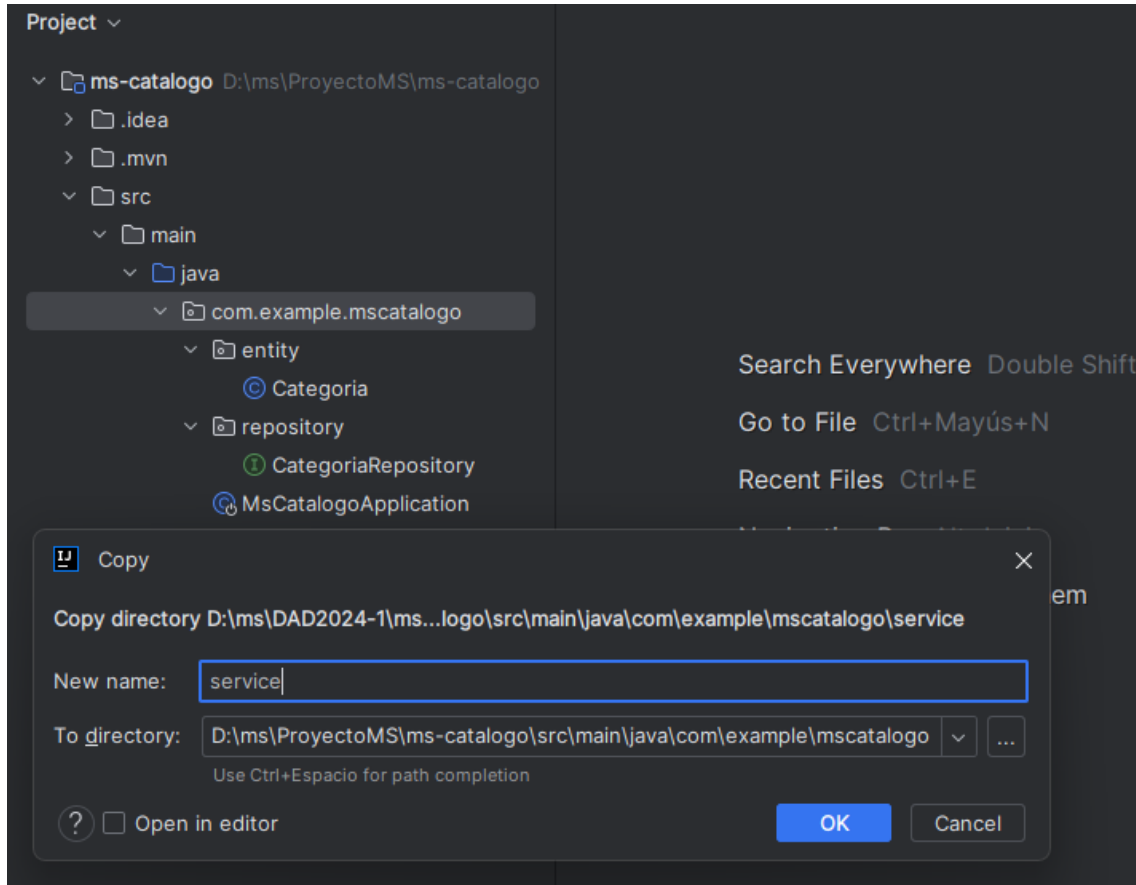
Una interfaz que funciona como una capa de abstracción sobre la capa de almacenamiento de datos de la aplicación. Su propósito es ofrecer métodos y acciones específicas para interactuar con la base de datos y ejecutar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en las entidades.



```
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);

}
```



```
Project
├── repository
│   └── CategoriaRepository
├── service
│   └── impl
│       ├── CategoriaServiceImpl
│       └── CategoriaService
├── util
│   ├── CategoriaSeeder
│   └── MsCatalogoApplication
├── resources
│   ├── static
│   └── templates
└── application.yml
```

```
1 package com.example.mscatalogo.service;
2
3 import com.example.mscatalogo.entity.Categoria;
4
5 import java.util.List;
6 import java.util.Optional;
7
8 public interface CategoriaService {
9     public List<Categoria> listar();
10    public Categoria guardar(Categoria categoria);
11    public Categoria actualizar(Categoria categoria);
12    public Optional<Categoria> listarPorId(Integer id);
13    public void eliminarPorId(Integer id);
14}
```

ServiceImpl

```
Project
├── repository
│   └── CategoriaRepository
├── service
│   └── impl
│       ├── CategoriaServiceImpl
│       └── CategoriaService
├── util
│   ├── CategoriaSeeder
│   └── MsCatalogoApplication
├── resources
│   ├── static
│   └── templates
└── application.yml
```

```
10 import java.util.Optional;
11
12 @Service
13 public class CategoriaServiceImpl implements CategoriaService {
14     @Autowired
15     private CategoriaRepository categoriaRepository;
16
17     @Override
18     public List<Categoria> listar() {
19
20         return categoriaRepository.findAll();
21     }
22
23     @Override
24     public Categoria guardar(Categoria categoria) {
25
26         return categoriaRepository.save(categoria);
27     }
28}
```

```
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);
    }

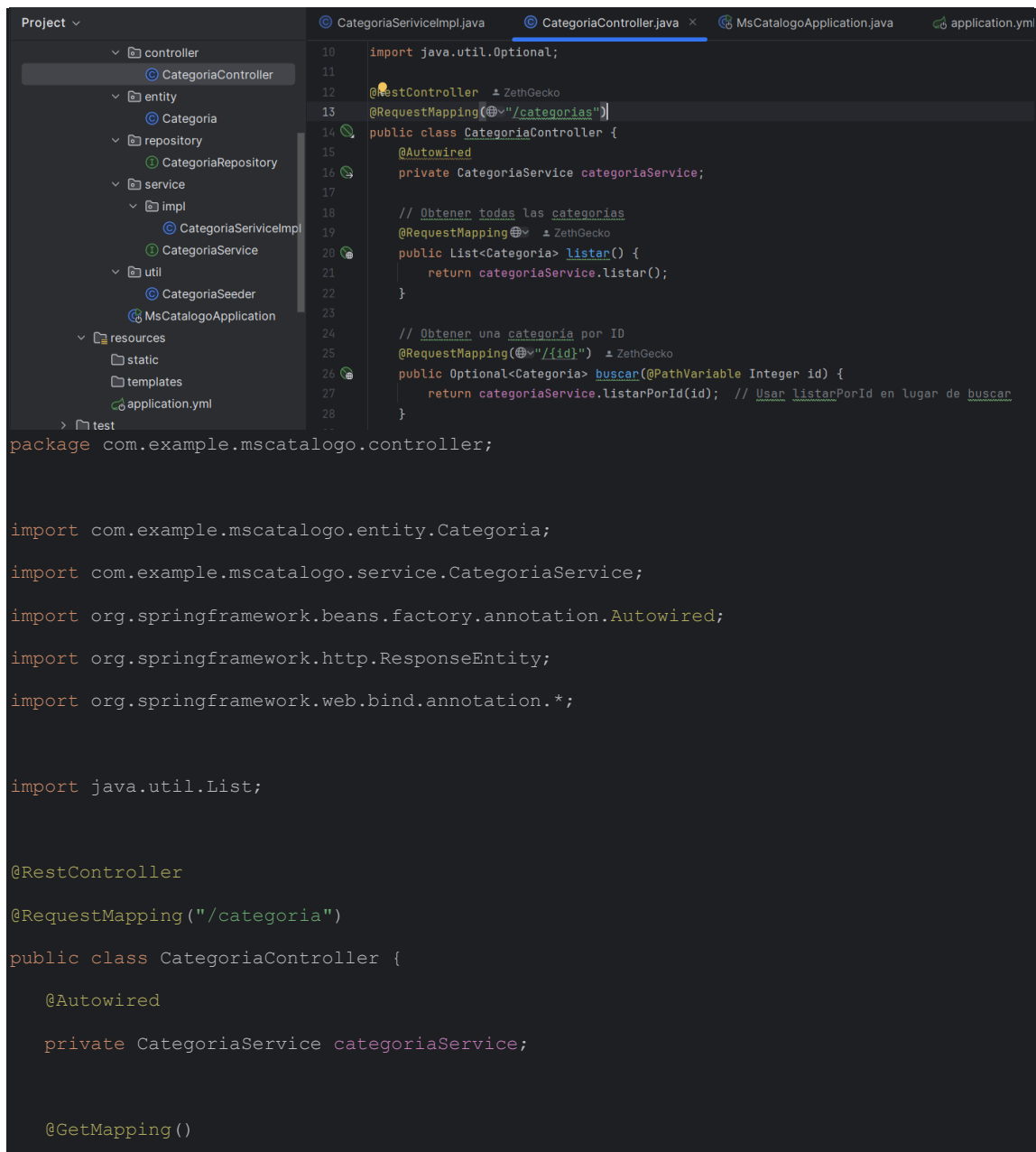
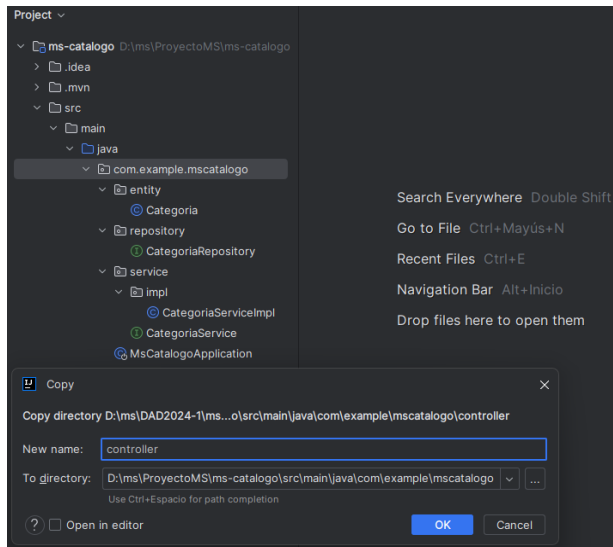
    @Override
    public Optional<Categoria> listarPorId(Integer id) {
        return categoriaRepository.findById(id);
    }

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

```

Controller:

Es una clase que funciona como punto de entrada para las solicitudes HTTP provenientes del cliente (como un navegador web, una aplicación móvil, etc.). Su objetivo es gestionar las peticiones del cliente y devolver la respuesta adecuada, que puede ser en formatos como HTML, JSON, XML, entre otros.



```

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));
}

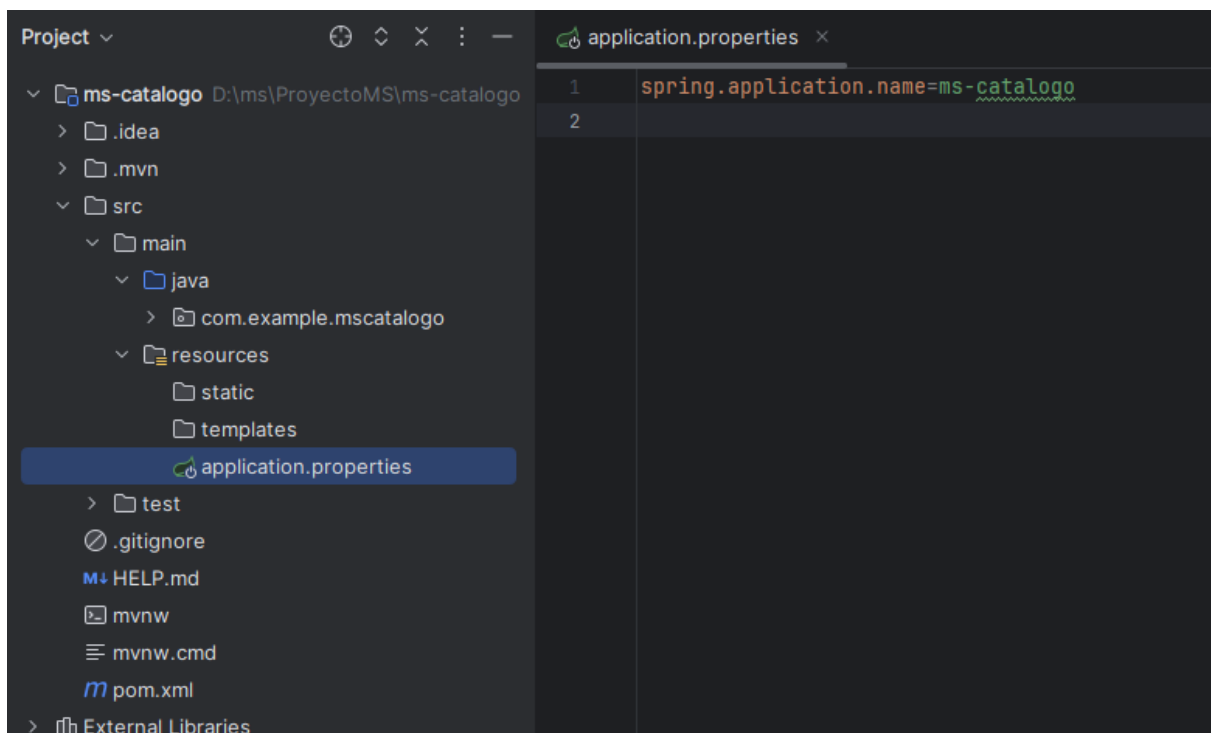
@GetMapping("/{id}")
public ResponseEntity<Categoria> listById(@PathVariable(required = true) Integer id){
    return ResponseEntity.ok().body(categoriaService.listarPorId(id).get());
}

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

```

Conexión a la base de datos:

Para conectar y realizar la operación CRUD con MySQL DB con la aplicación Spring Boot, se debe configurar dentro del archivo [application.properties](#) de la siguiente manera:



```

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

```

application.properties x

```

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

```

Ejecutar:

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project structure with folders for controller, entity, repository, service, impl, util, resources, static, templates, and application.yml.
- Editor:** Displays the `MsCatalogoApplication.java` file. The code is as follows:


```

package com.example.msCatalogo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MsCatalogoApplication {

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

}

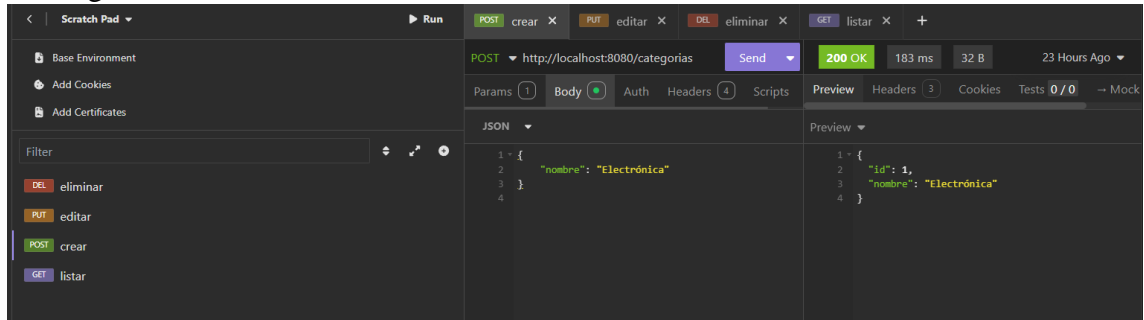
```
- Run Console:** Shows the output of the application. The logs indicate that the application started successfully on port 8081. The logs are as follows:


```

2025-03-27T10:11:25.883-05:00 INFO 15264 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-03-27T10:11:25.962-05:00 WARN 15264 --- [main] org.hibernate.orm.deprecation : HHH90000026: MySQL8Dialect has been deprecated
2025-03-27T10:11:26.219-05:00 INFO 15264 --- [main] o.h.b.i.BytecodeProviderInitiator : HHH000021: Bytecode provider name : bytebuddy
2025-03-27T10:11:26.978-05:00 INFO 15264 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation:
2025-03-27T10:11:27.086-05:00 INFO 15264 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for pers:
2025-03-27T10:11:27.473-05:00 WARN 15264 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default
2025-03-27T10:11:28.552-05:00 INFO 15264 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8081 (http) with g
2025-03-27T10:11:28.654-05:00 INFO 15264 --- [main] c.e.msCatalogo.MsCatalogoApplication : Started MsCatalogoApplication in 13.009 seconds

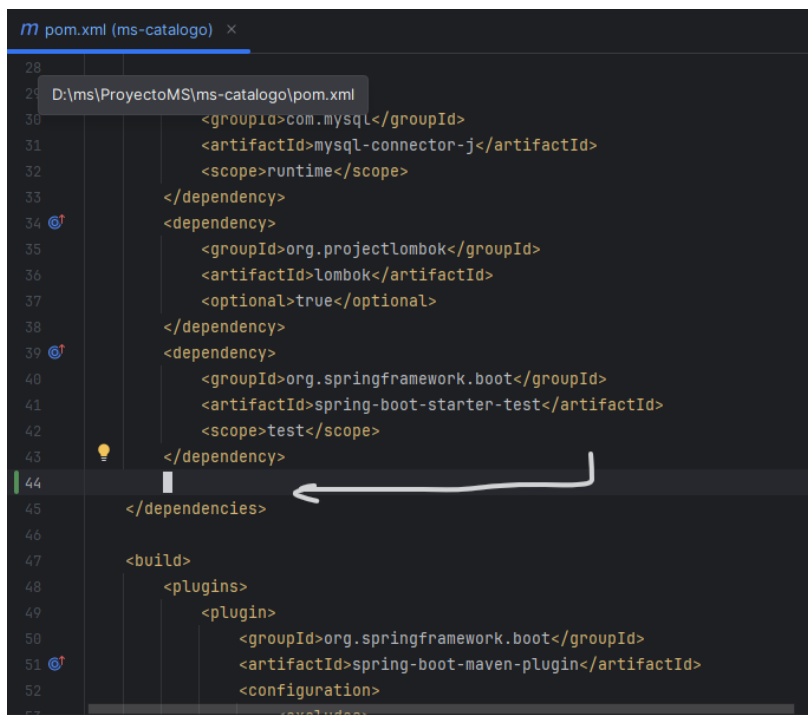
```

Testing:



OpenAPI Specification

Agregar dependencias en maven pom.xml (si usas maven)



```
<dependency>

    <groupId>org.springdoc</groupId>

    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

    <version>2.0.2</version>

</dependency>

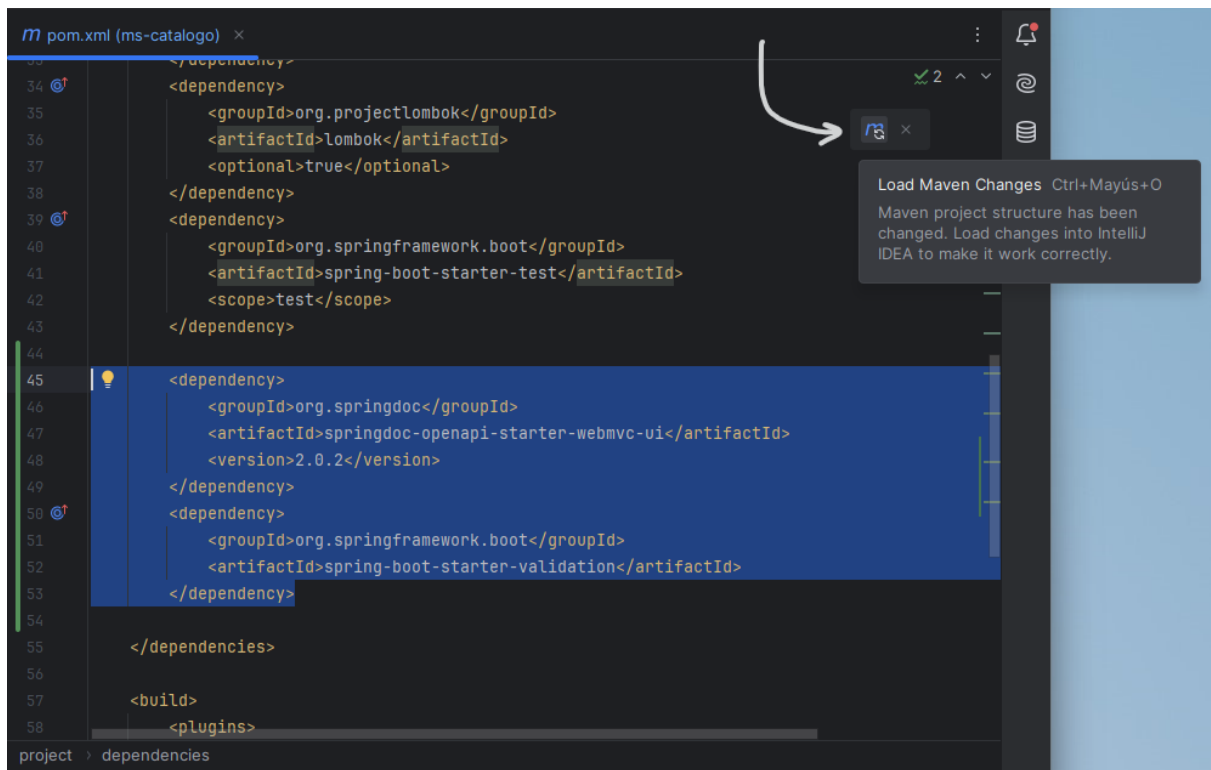
<dependency>

    <groupId>org.springframework.boot</groupId>

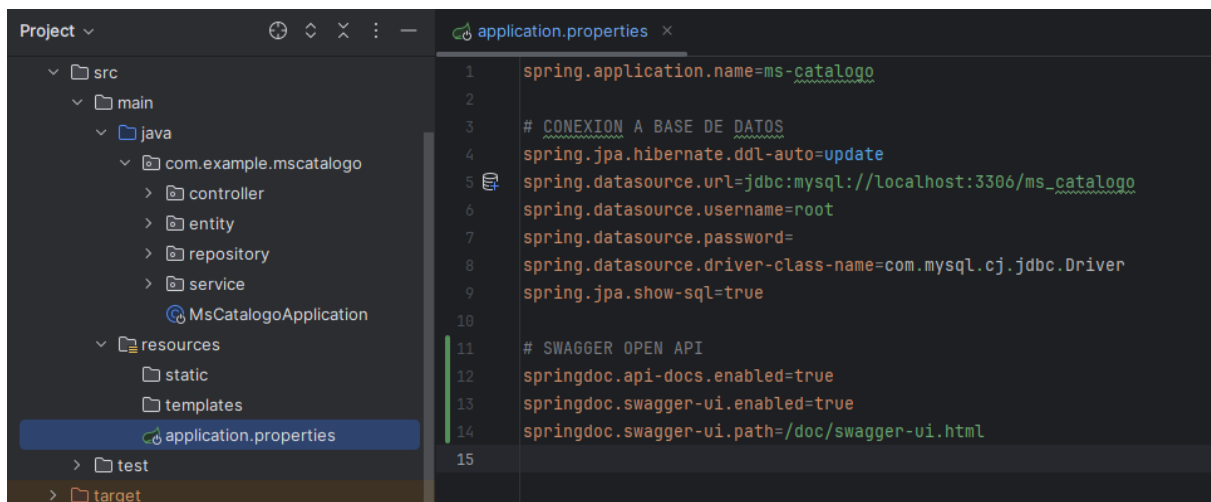
    <artifactId>spring-boot-starter-validation</artifactId>

</dependency>
```

Ahora, debes actualizar la descarga de las librerías



Configurar en: application.properties



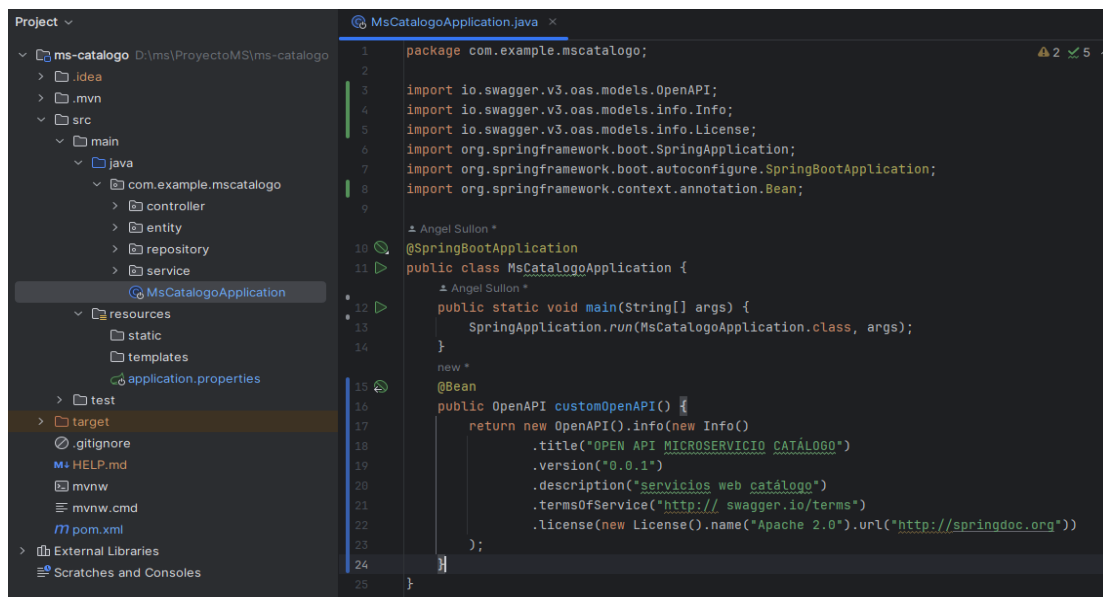
```
# SWAGGER OPEN API

springdoc.api-docs.enabled=true

springdoc.swagger-ui.enabled=true

springdoc.swagger-ui.path=/doc/swagger-ui.html
```

Configurar Bean: MsCatalogoApplication



```
package com.example.mscatalogo;

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")))

        );

    }

}
```


OPEN API MICROSERVICIO CATÁLOGO 0.0.1 OAS3

/v3/api-docs

servicios web catálogo

[Terms of service](#)

[Apache 2.0](#)

Servers

[http://localhost:8080](#) - Generated server url

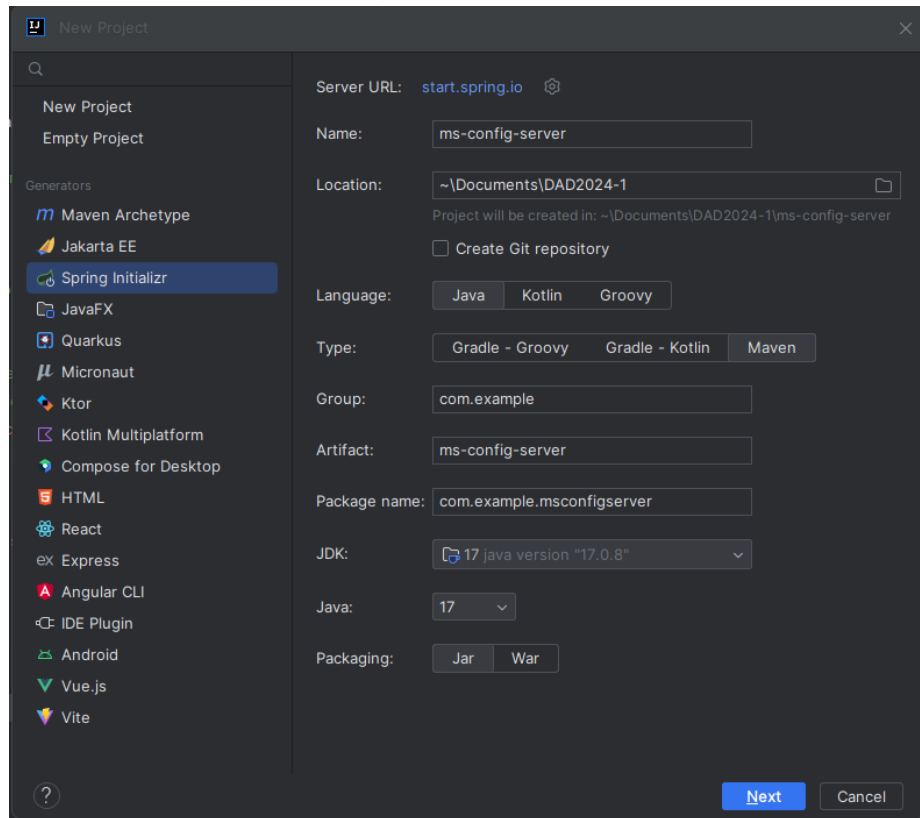
categoria-controller		^
GET	/categoria	✓
PUT	/categoria	✓
POST	/categoria	✓
GET	/categoria/{id}	✓
DELETE	/categoria/{id}	✓
Schemas		^

Llevar de monolito a microservicios:

Crear el ms-config-serve:

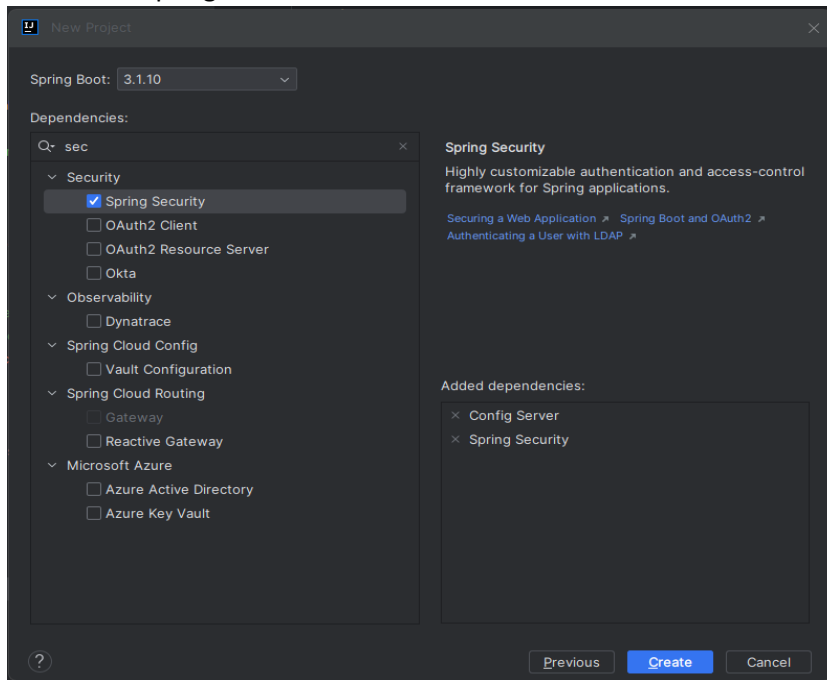
Nombre: ms-config-server

Tipo: Maven



Seleccionar las librerías:

Versión de Spring Boot: 3.1.10



Cambiar la extecion de application.properties en el proyecto config-server por application.yml

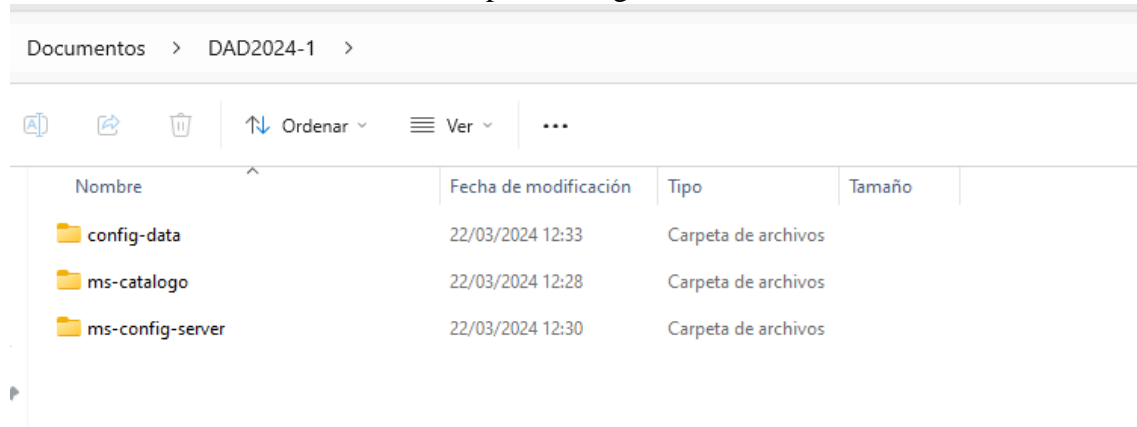
```
server:
  port: 7070

spring:
  application:
    name: config-server
  cloud:
    config:
      server:
        git:
          uri: https://github.com/ZethGecko/DAD---01.git # ruta del
repositorio git
          searchPaths: config-data
          default-label: main

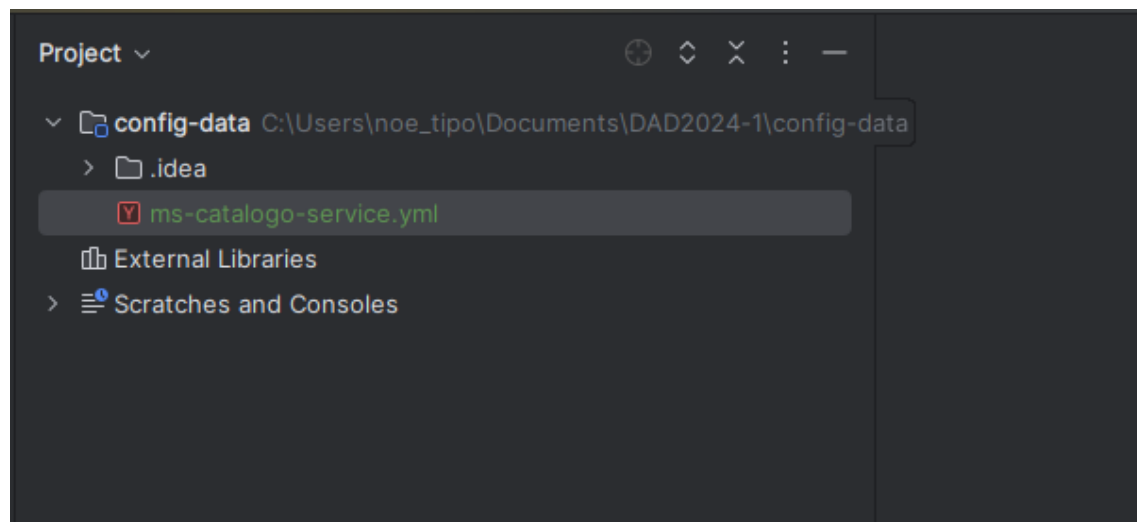
  security:
    user:
      name: root
      password: password
```

Agregar la anotación @EnableConfigServe en MsConfig Server Application

En la misma ruta crear una nueva carpeta config-data:



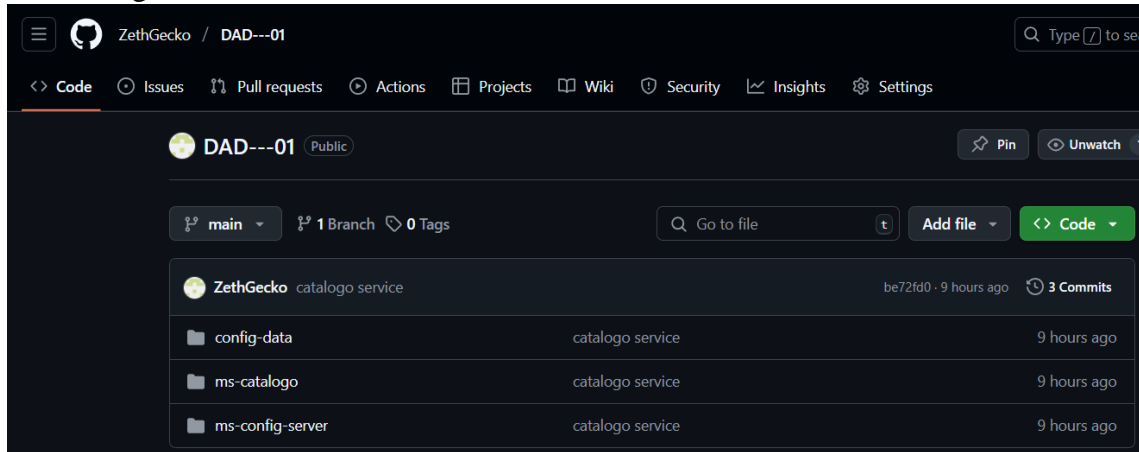
Editar el archivo ms-catalogo-service.yml



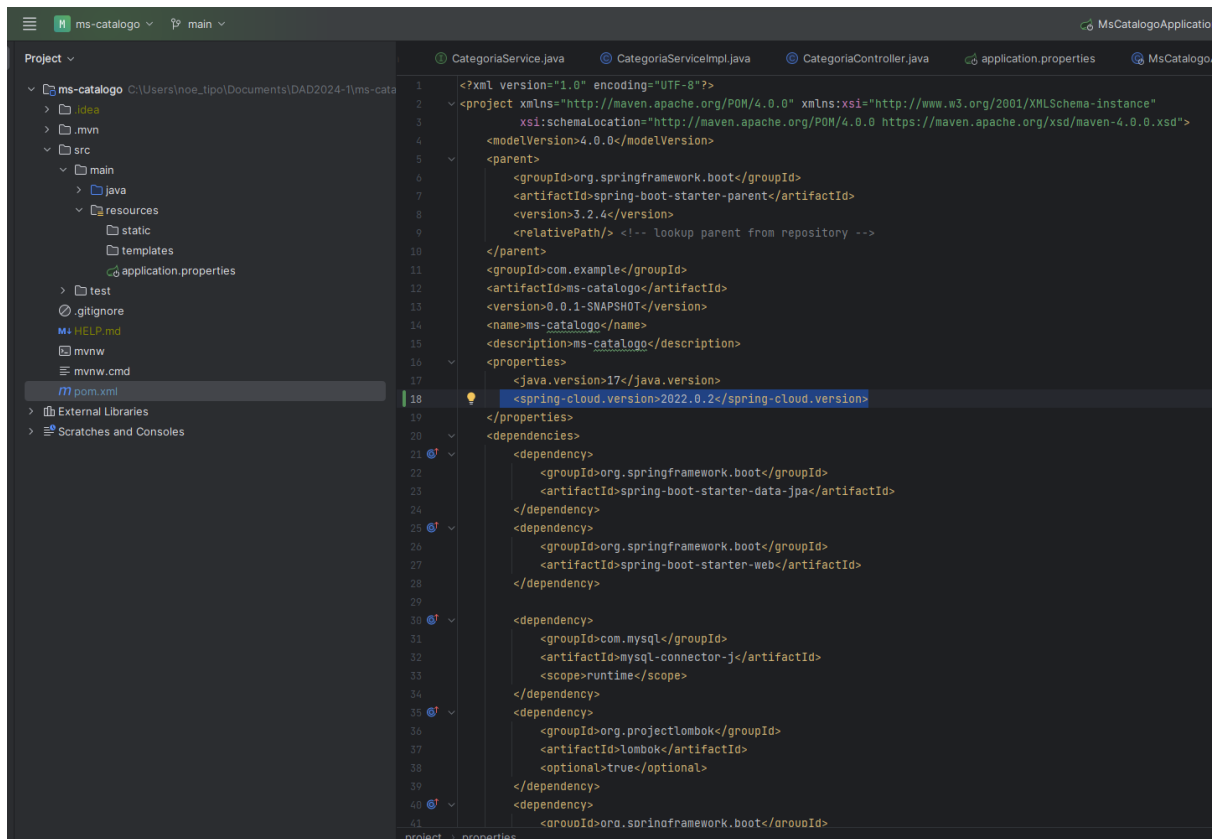
código para copiar y pegar

```
server:  
  port: 8081  
springdoc:  
  api-docs:  
    enabled: true  
  swagger-ui:  
    enabled: true  
    path: /doc/swagger-ui.html
```

Subirlo a github todo los archivos:



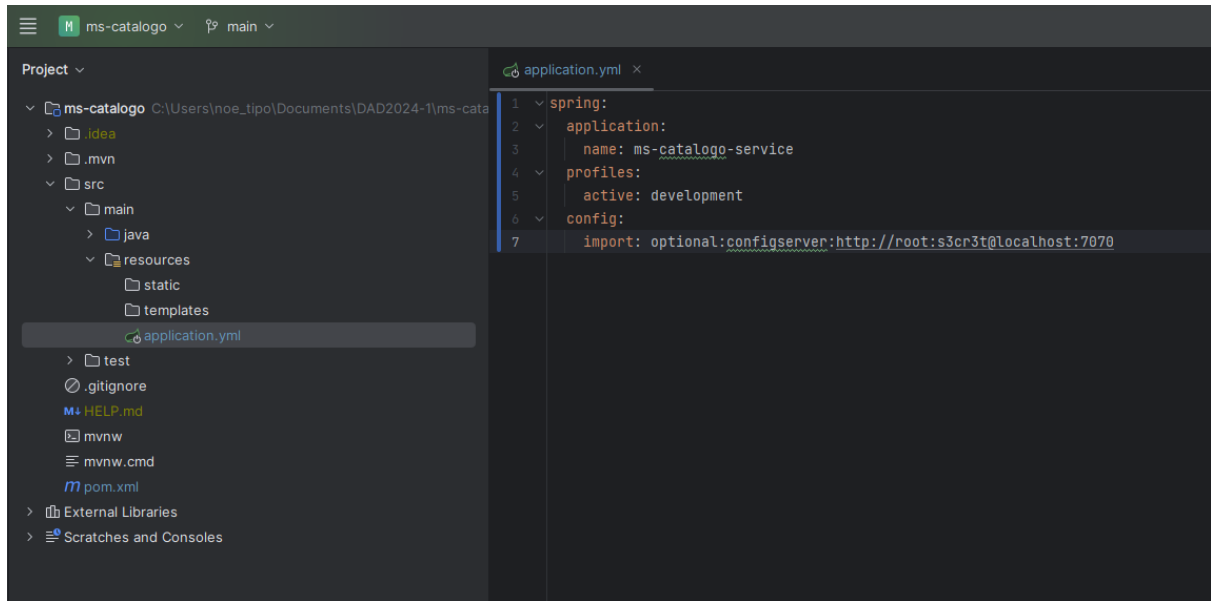
Luego agregar la versión de spring cloud en el proyecto ms-catálogo



código para copiar y pegar

```
<spring-cloud.version>2022.0.2</spring-cloud.version>
```

editar el archivo `application.yml` del proyecto `application.yml`



código para copiar y pegar

```
spring:
  application:
    name: ms-catálogo-service
  profiles:
    active: development
  config:
```

resultados finales:

Imperius_GG Stream - Watch Live

Swagger UI

Please sign in

Interfaz de persistencia CRUD

localhost8081/doc/swagger-ui/index.html#/categoria-controller/listar

ExecuteClear

Responses

Curl

curl -X 'GET' \n'http://localhost:8081/categorias' \n-H 'accept: */*'

Request URL

http://localhost:8081/categorias

Server response

CodeDetails

200

Response body

```
{\n  "id": 1,\n  "nombre": "Electrónica"\n},\n{\n  "id": 2,\n  "nombre": "Ropa"\n},\n{\n  "id": 3,\n  "nombre": "Hogar"\n},\n{\n  "id": 4,\n  "nombre": "Juguetes"\n},\n{\n  "id": 5,\n  "nombre": "Libros"\n}\n}
```

Response headers