

Registry: Para registrar y descubrir los microservicios.

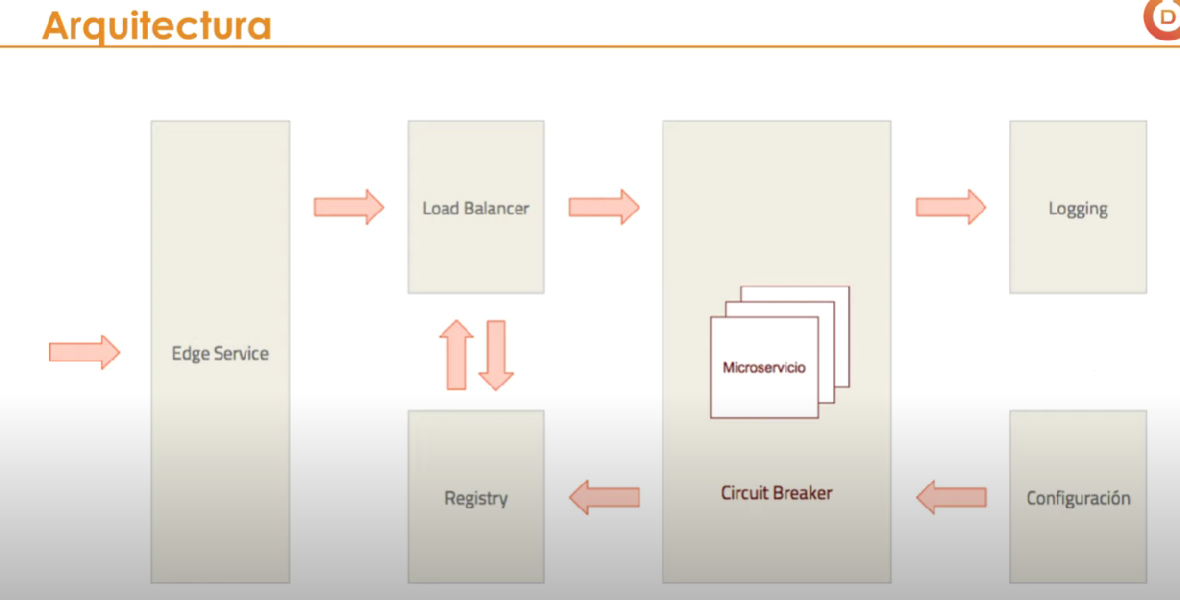
Config: Para tener una configuración centralizada de los MS.

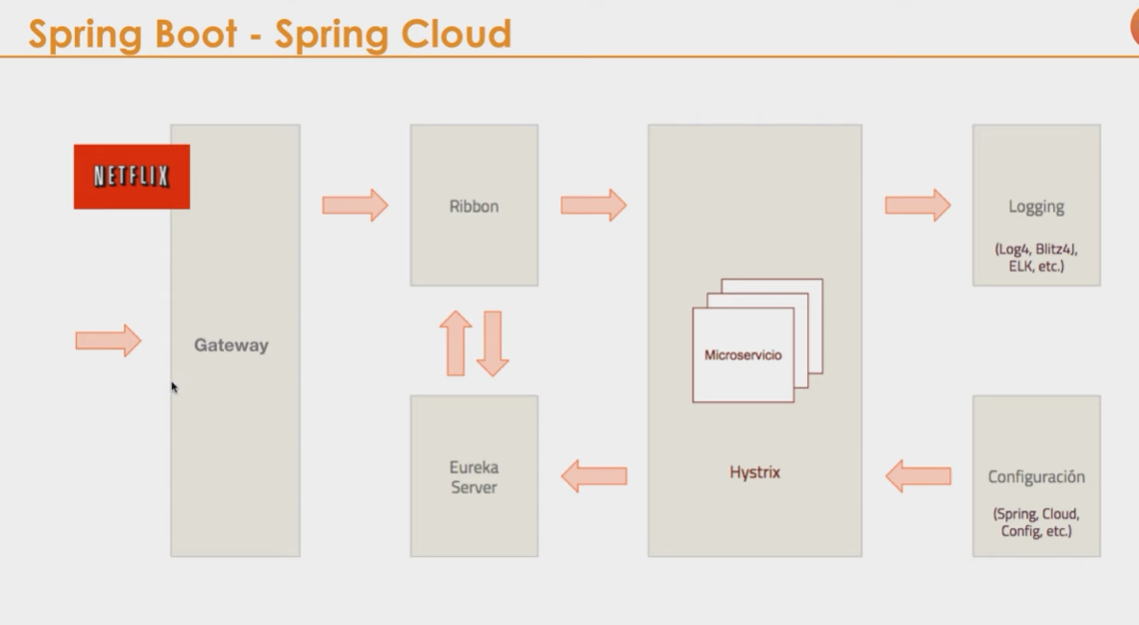
Load Balancer: Para hacer balanceo de carga.

Circuit Breaker: Para manejar la tolerancia a fallo de los MS.

Api Gateway: Para tener un único punto de acceso a los MS

Log Center: Para guaradar en un punto nuestros log’s.





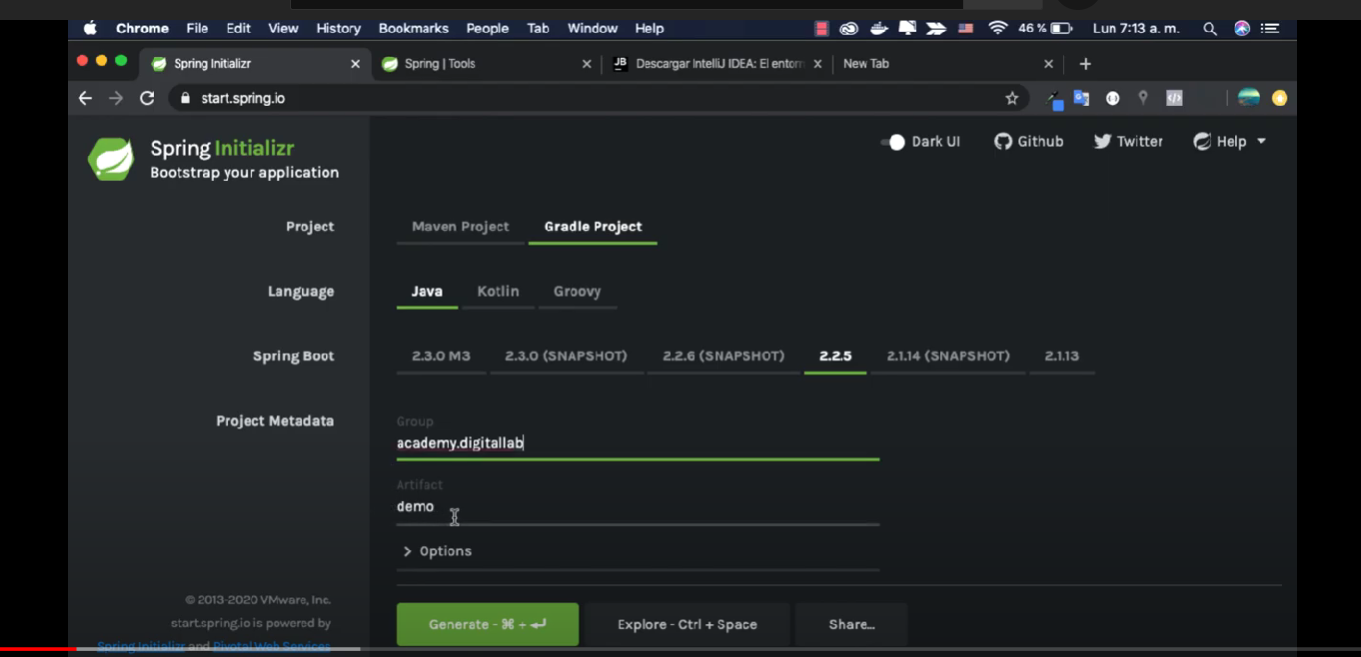
Gateway: único punto de entrada (Netflix)

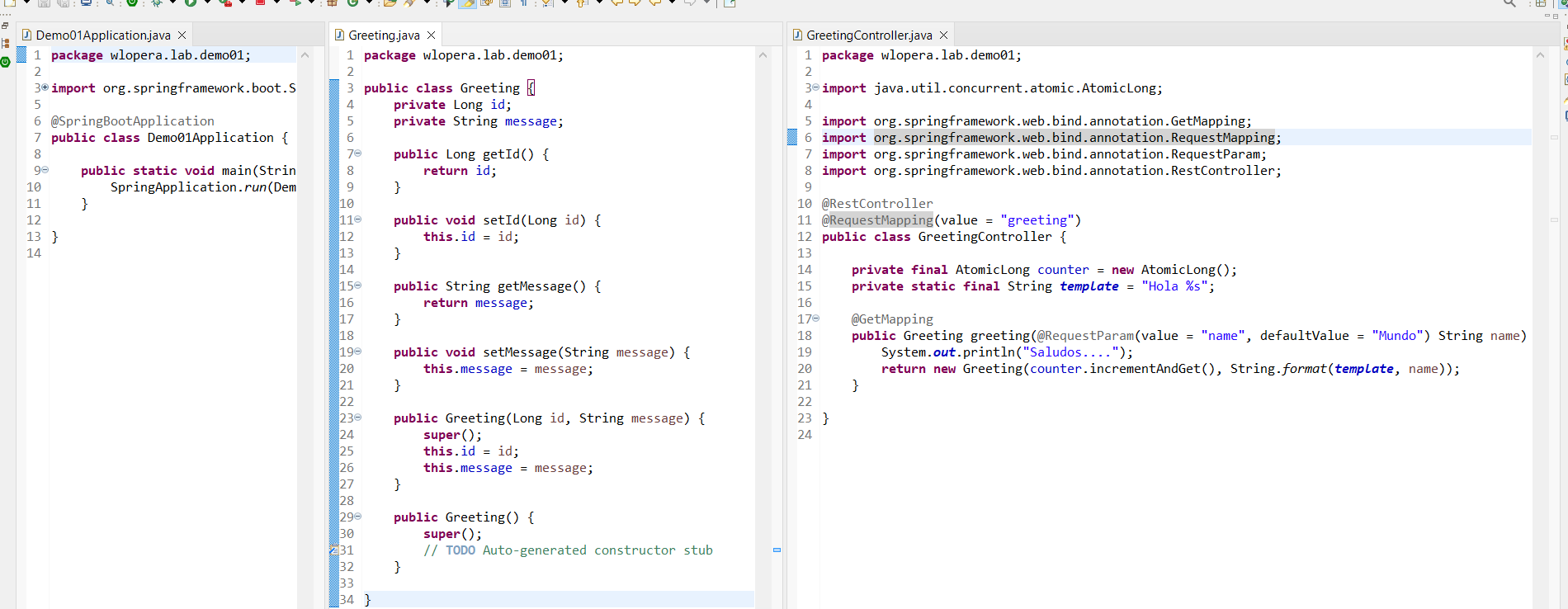
Ribbon: Balanceador de carga.

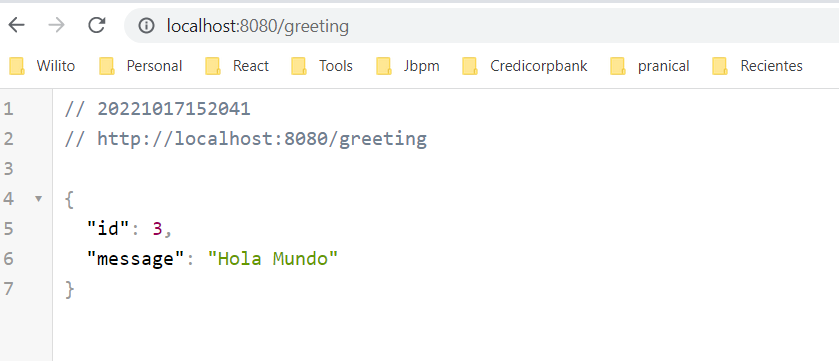
Eureka Server: Permite registrar servicios y realizar el descubrimiento o publicación de los MS.

Hystrix: Implemantar circuit breaker (tolerancia a fallos)

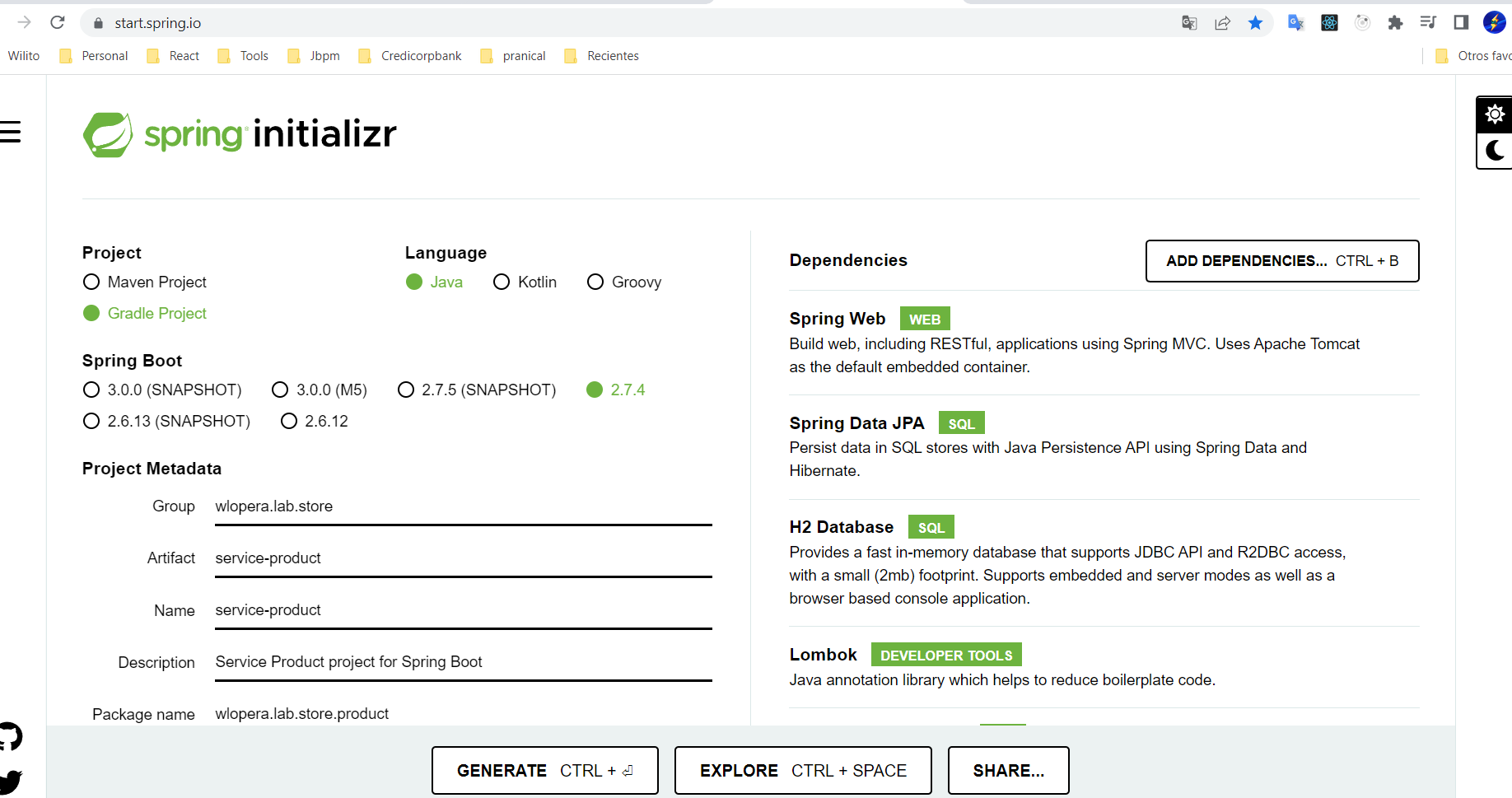
MS: Api Rest

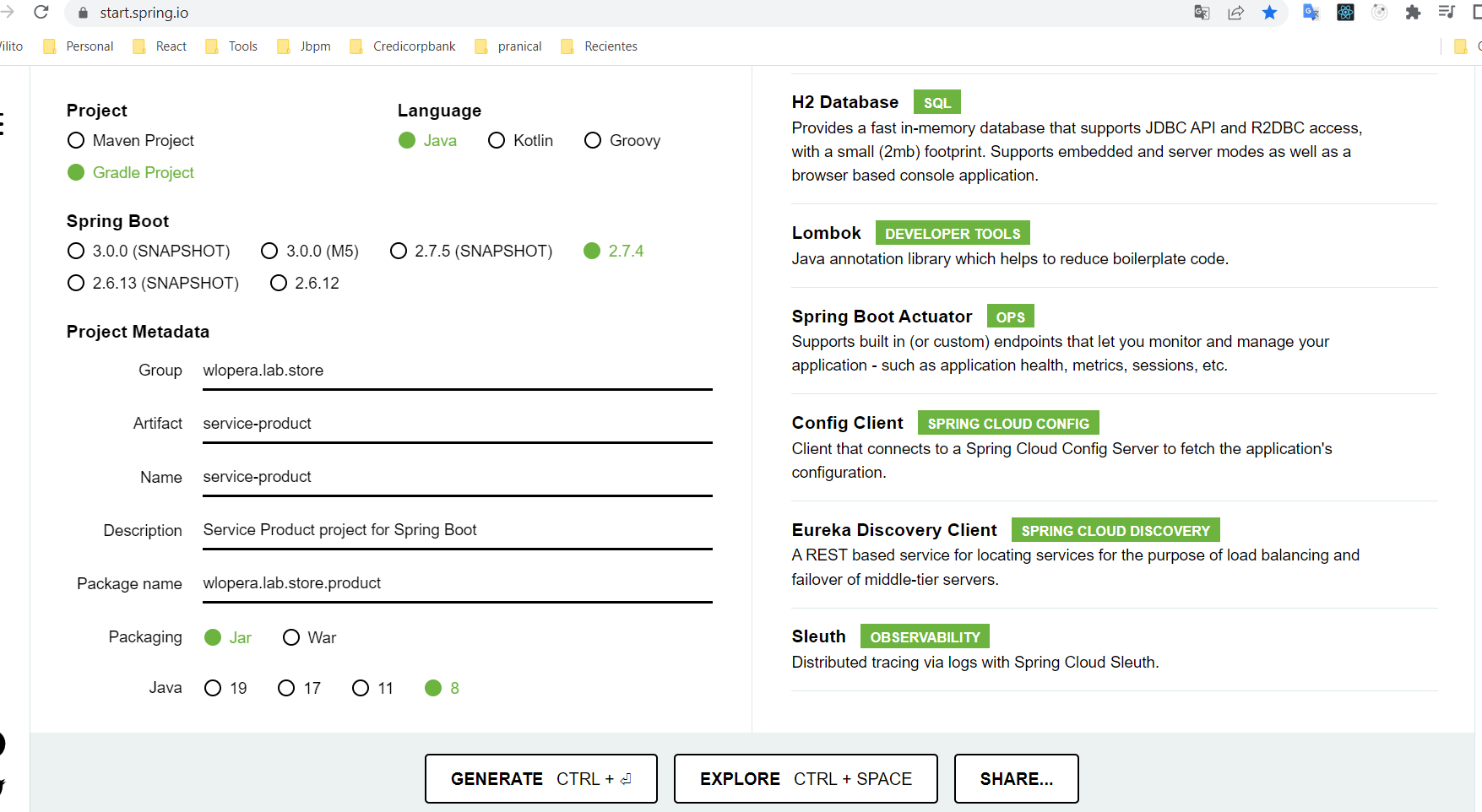


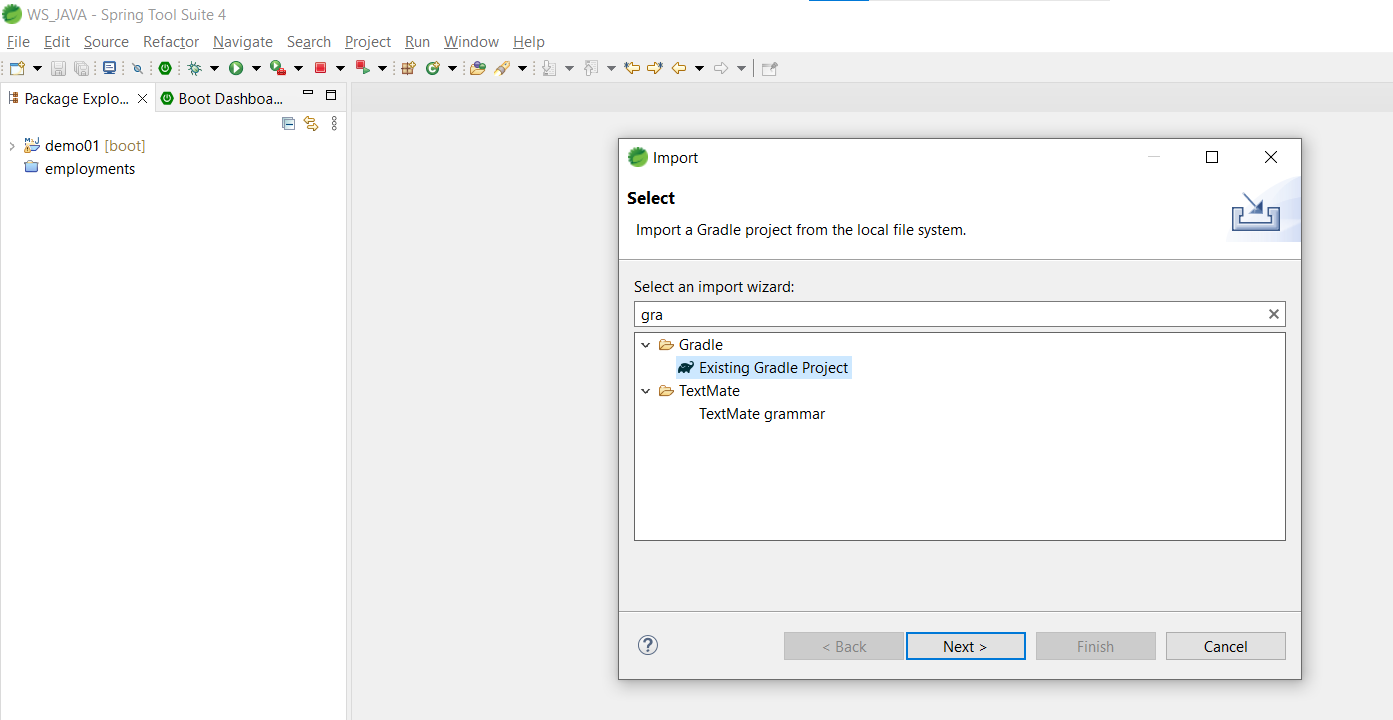


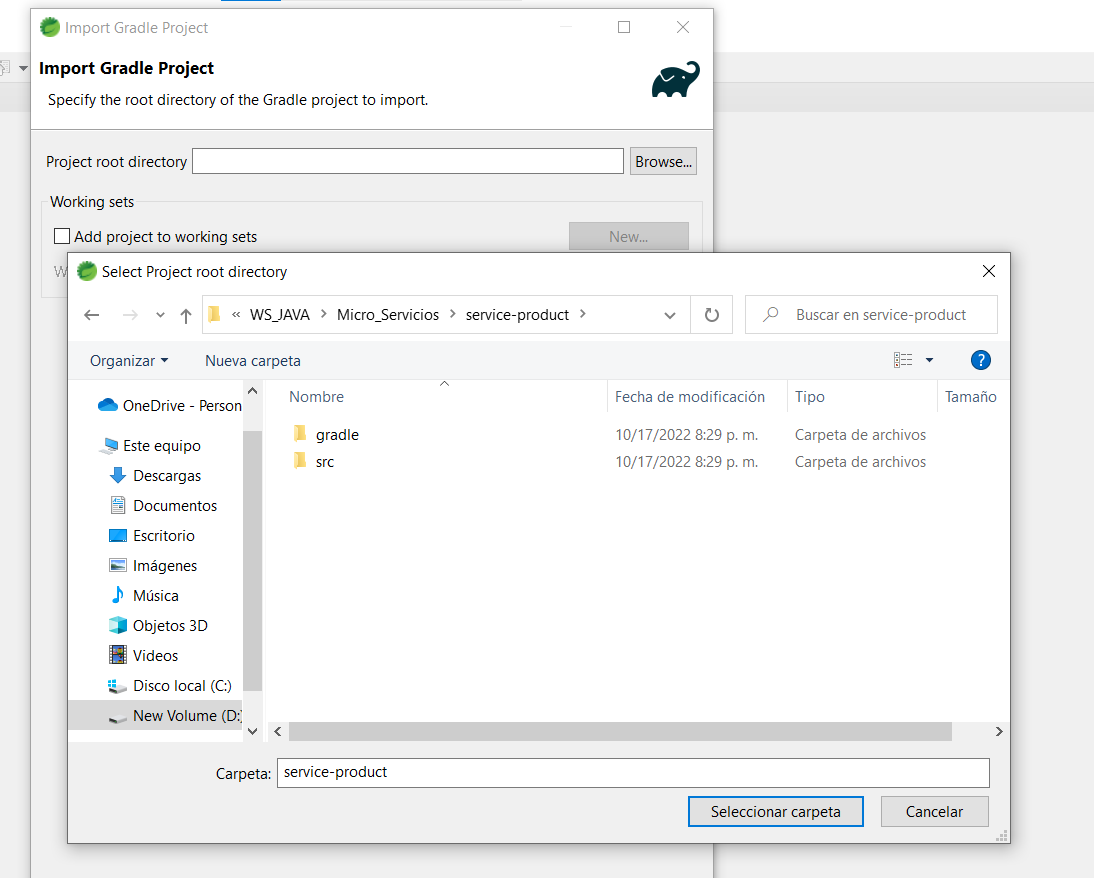


Nuevo Proyecto

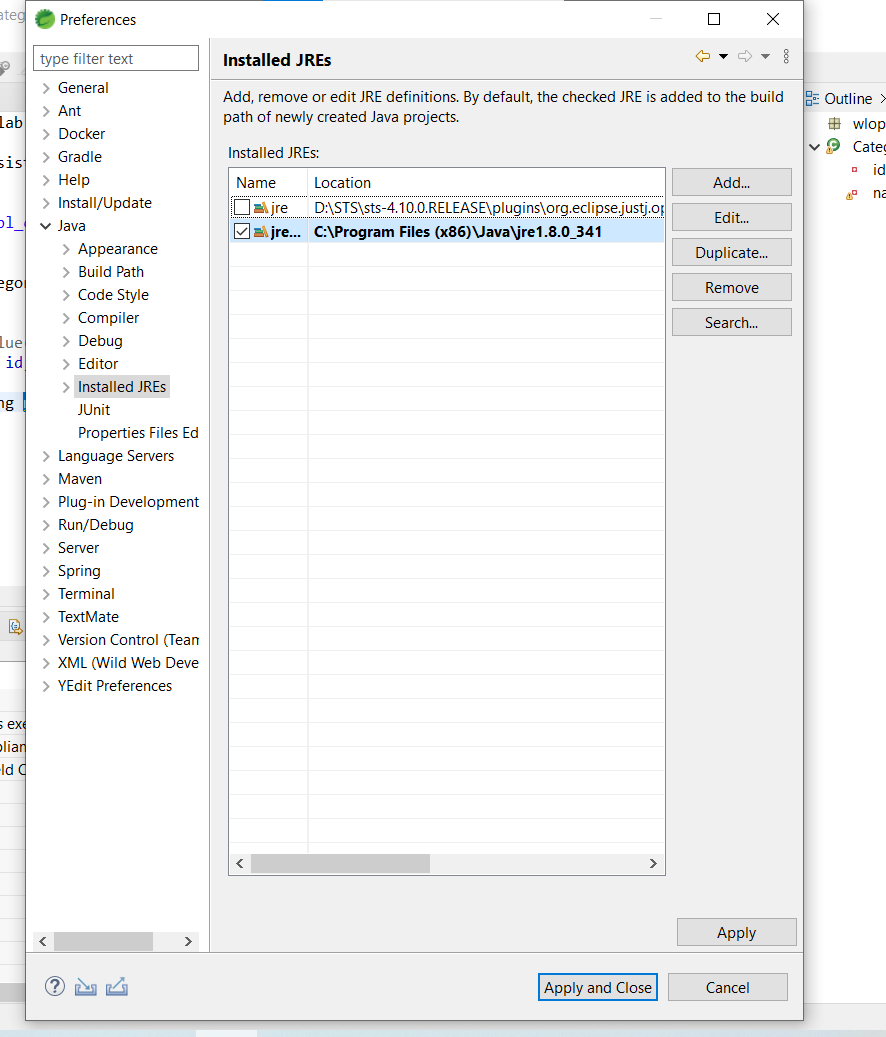


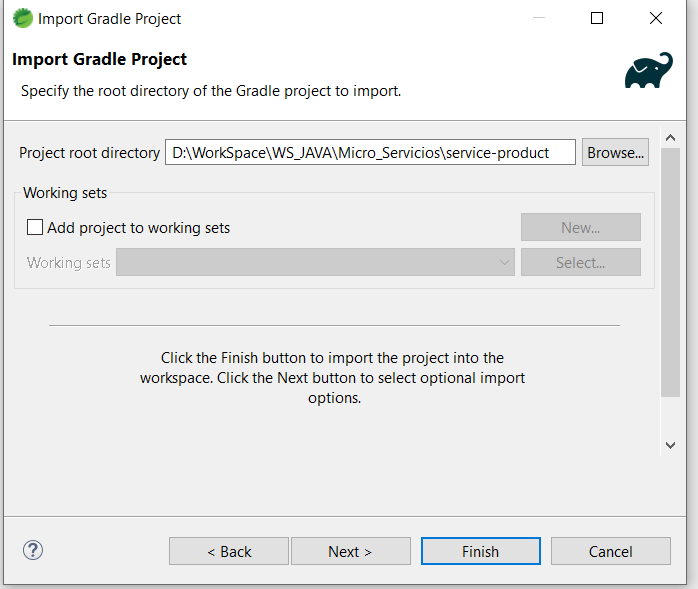


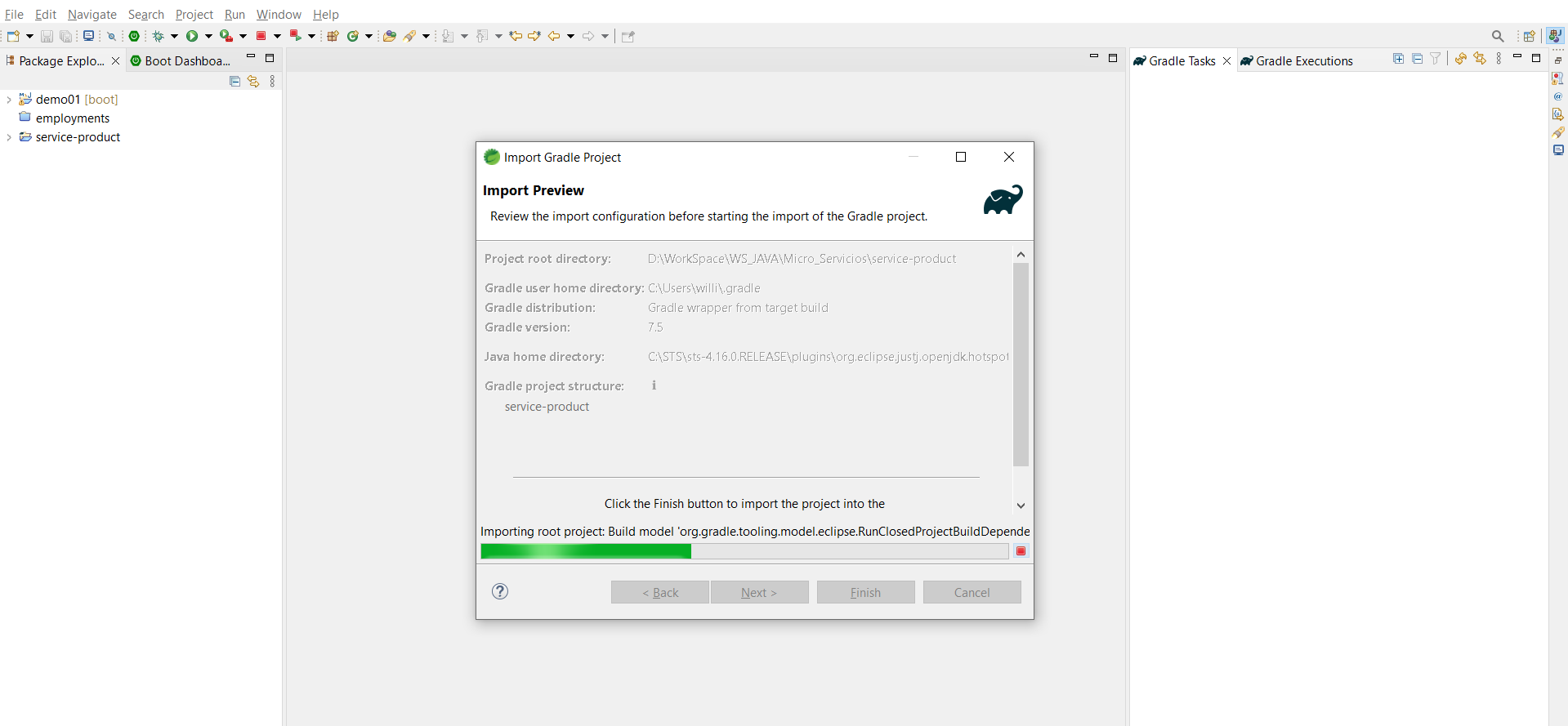


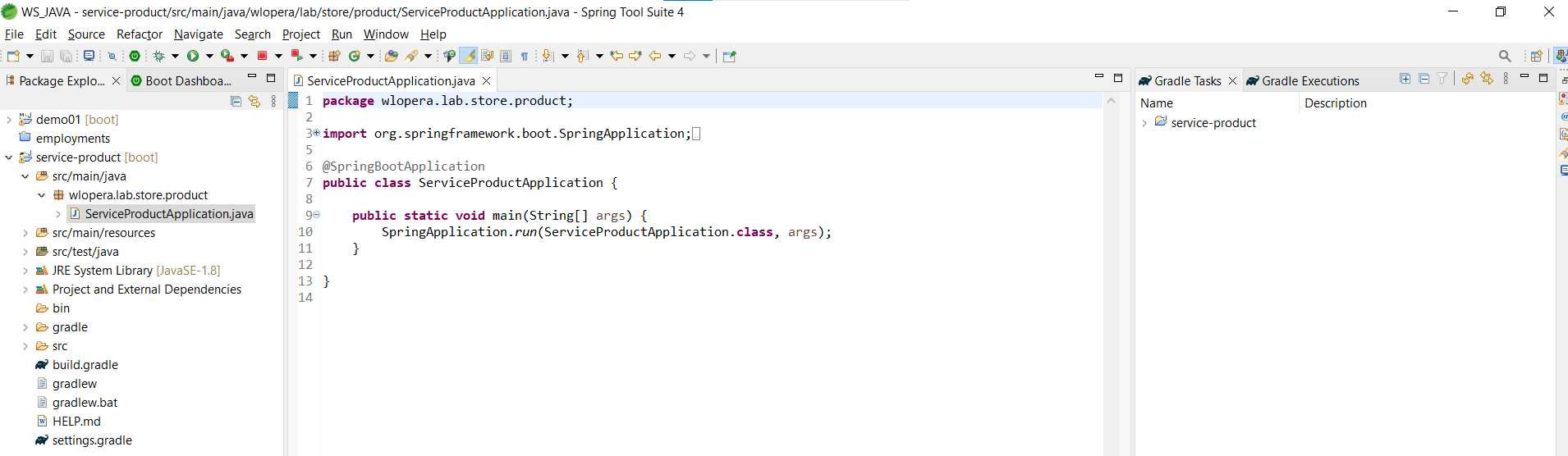


Configurar el jKE



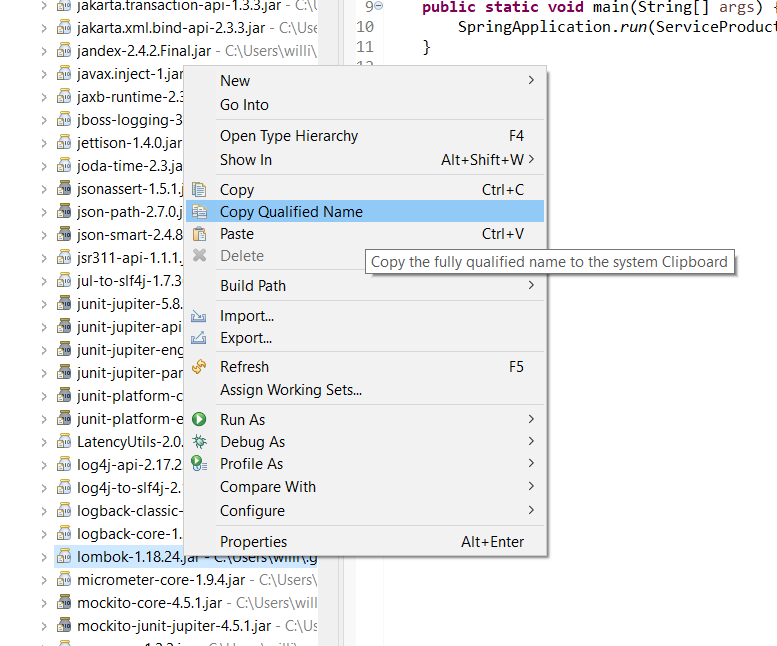




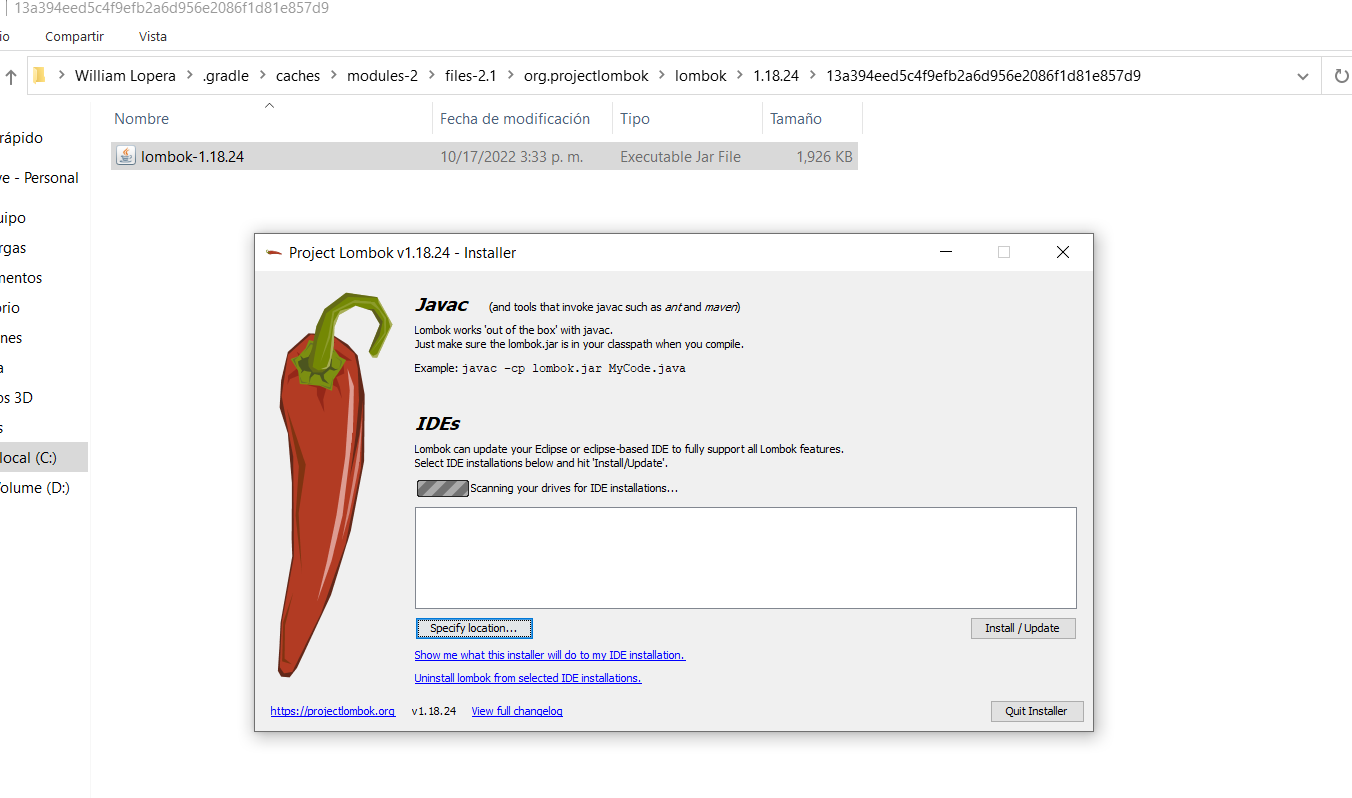


**Configurar Lombok**

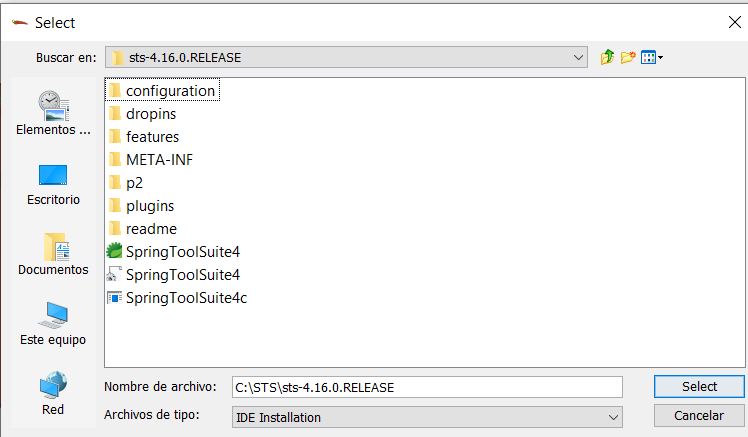
Copiar dirección de Lombok

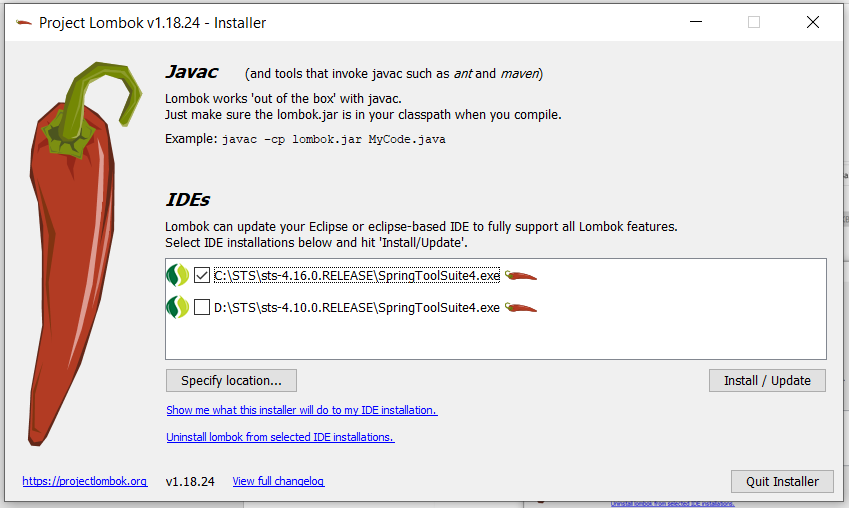


Abrir Lombok (.jar)



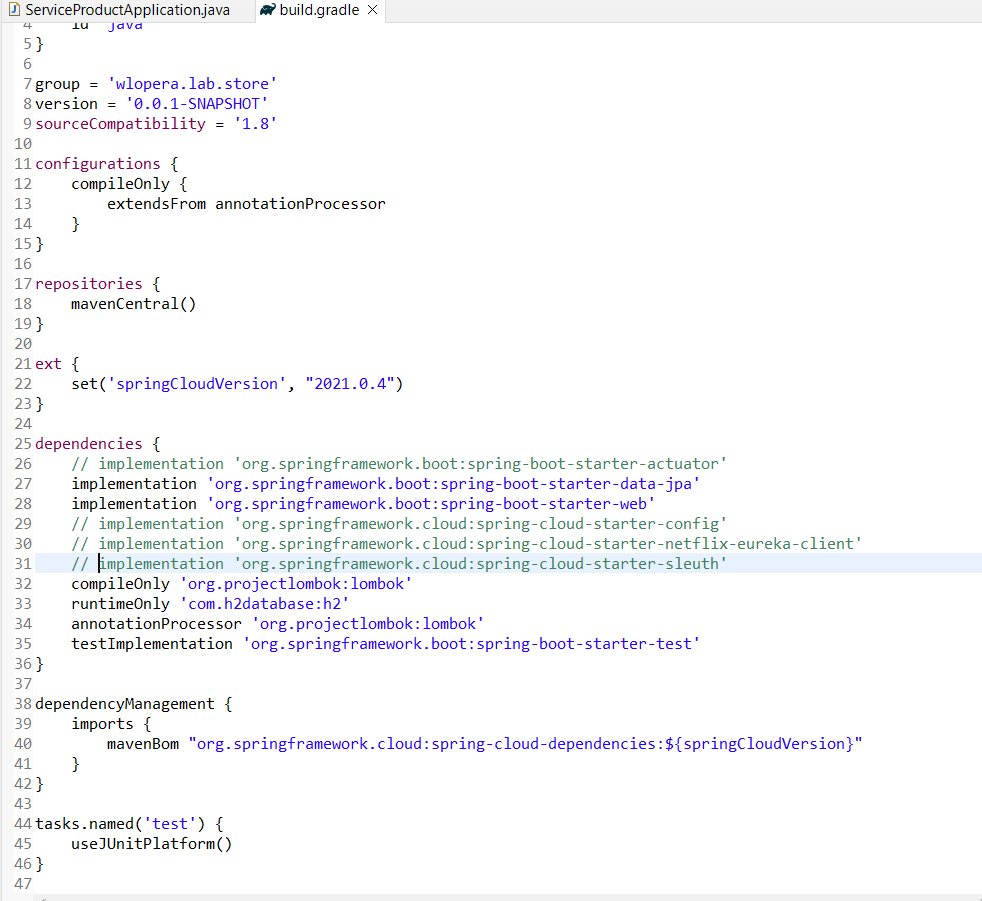
Especificar la ubicación de mi ID (en este caso STS)

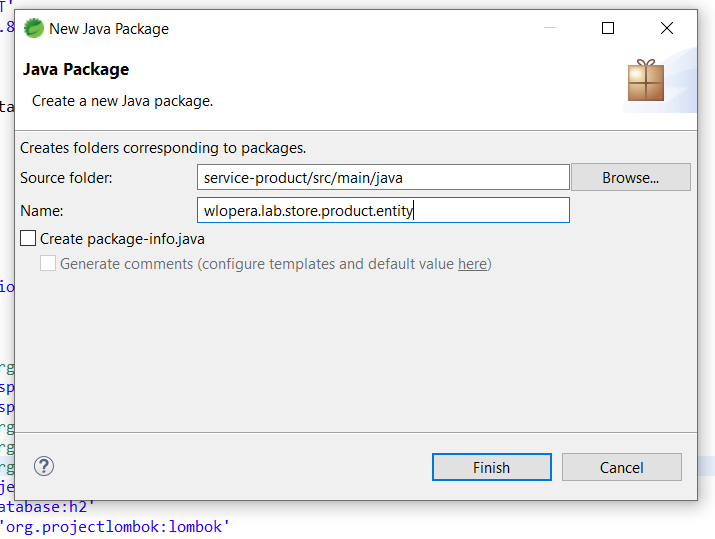


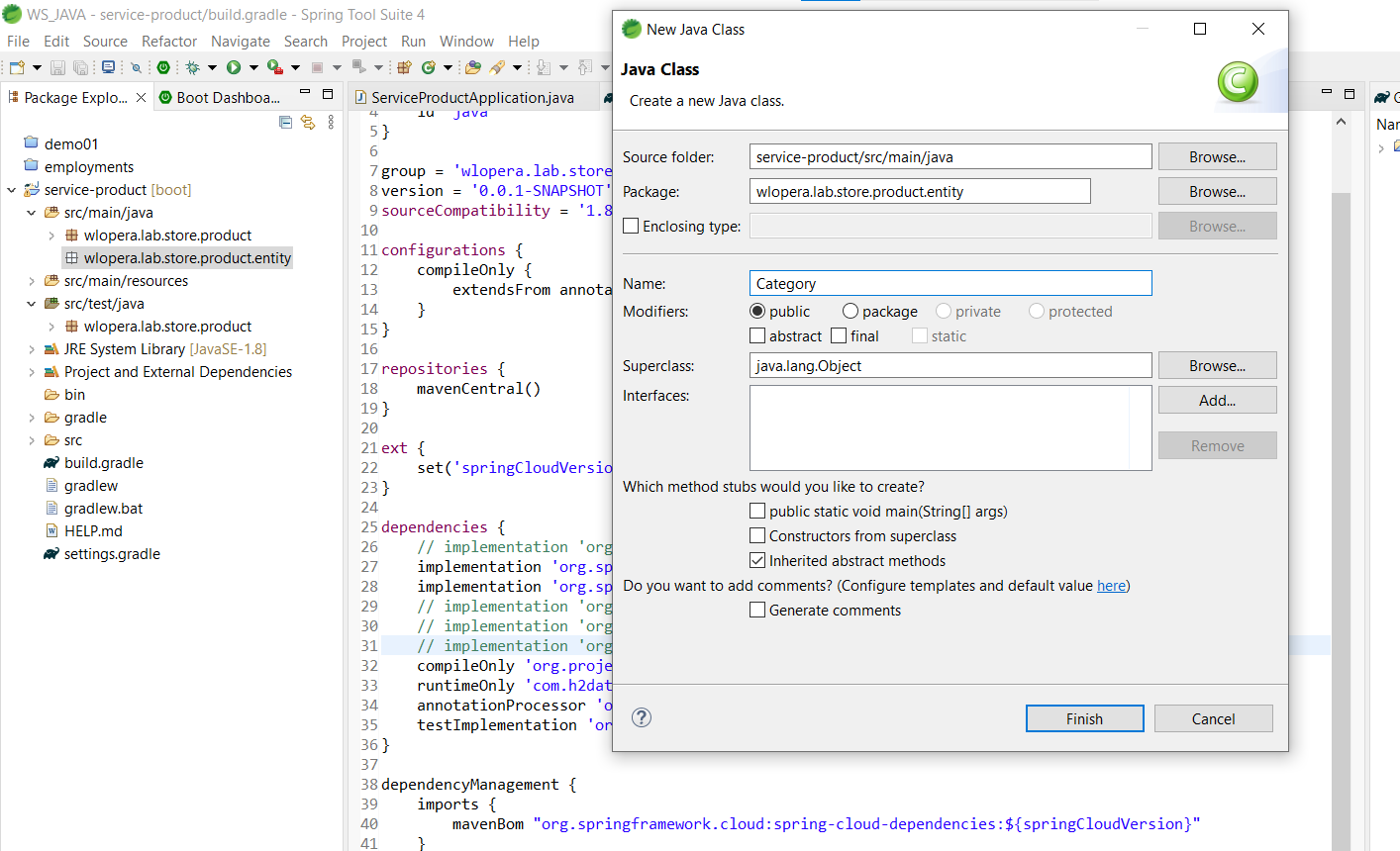


Cerrar y abrir STS

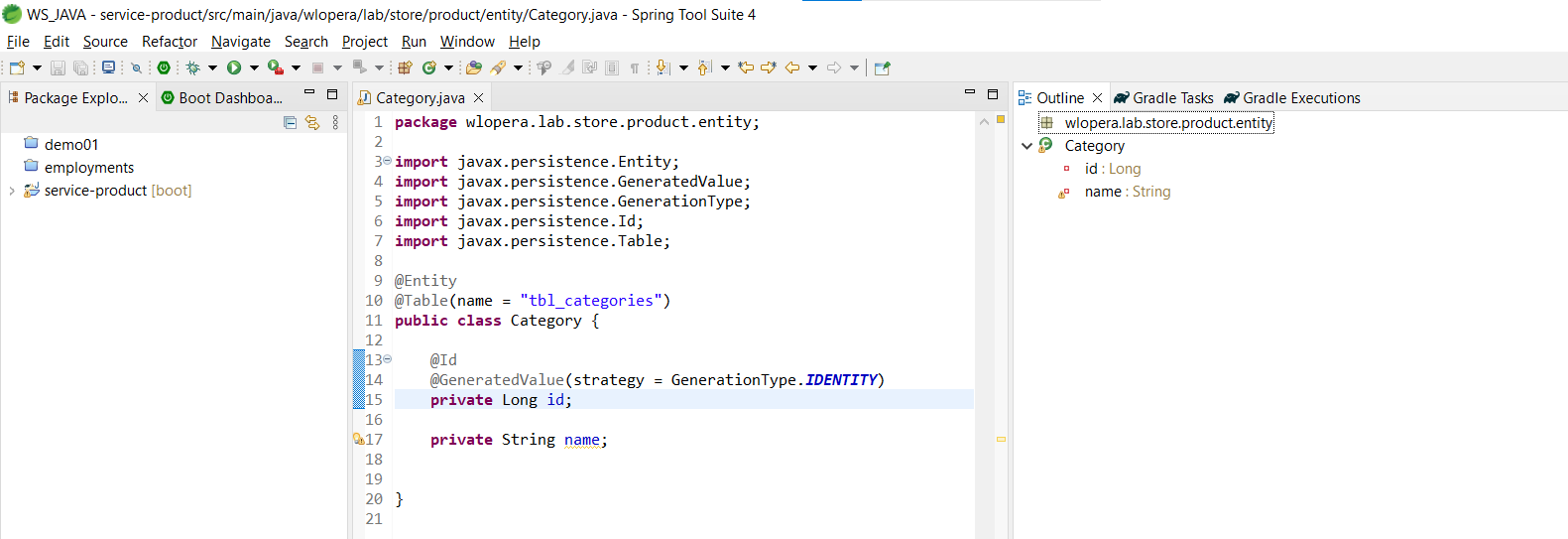
Luego deshabilitar .jar que no se utilizaran por ahora

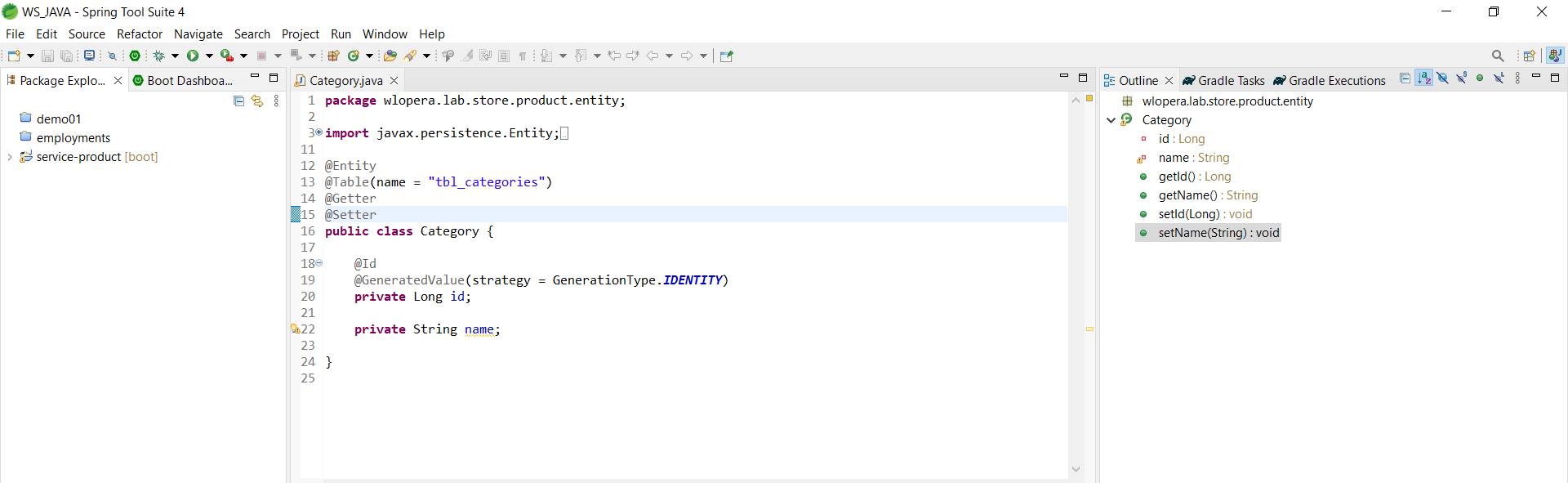




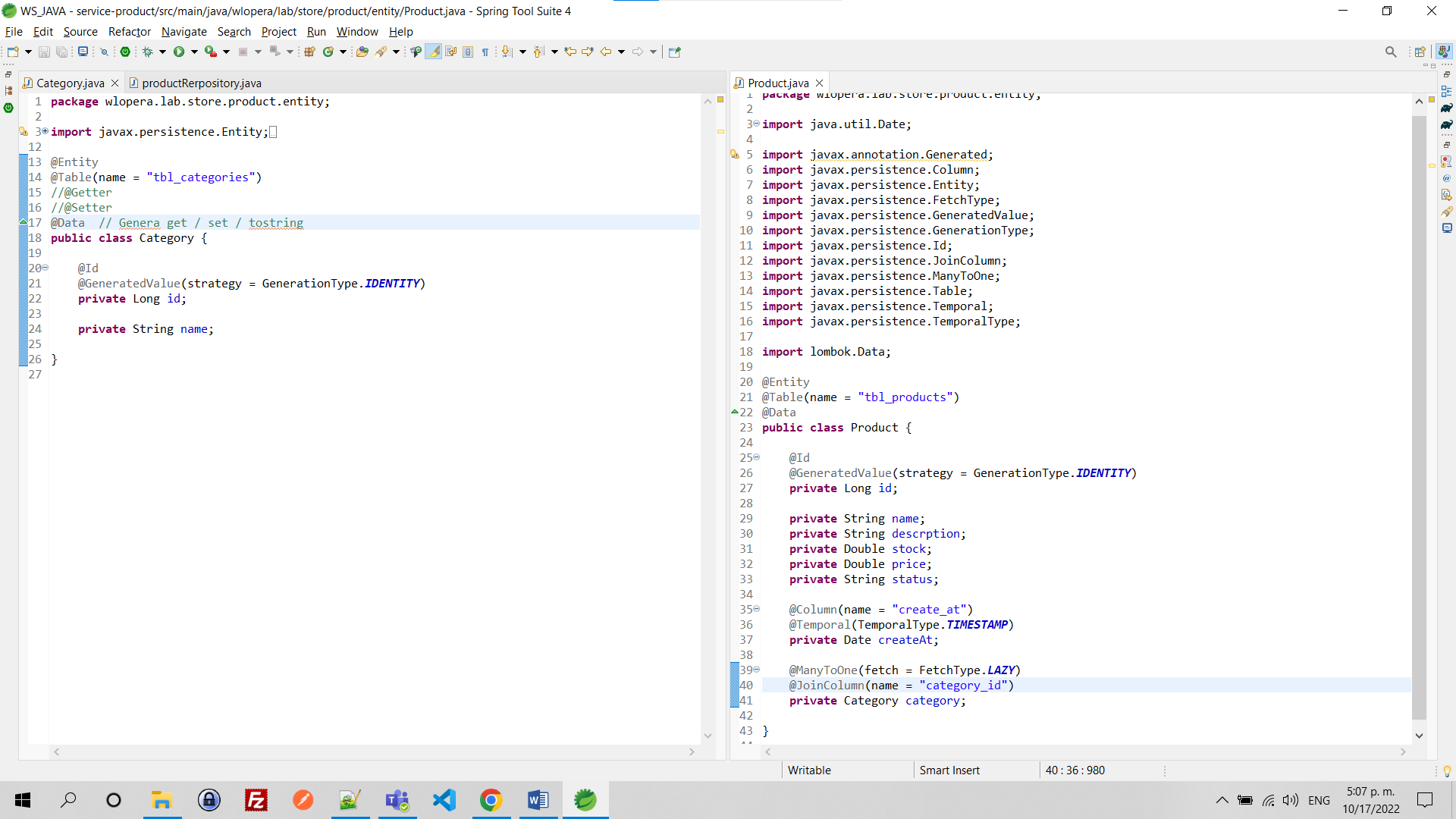


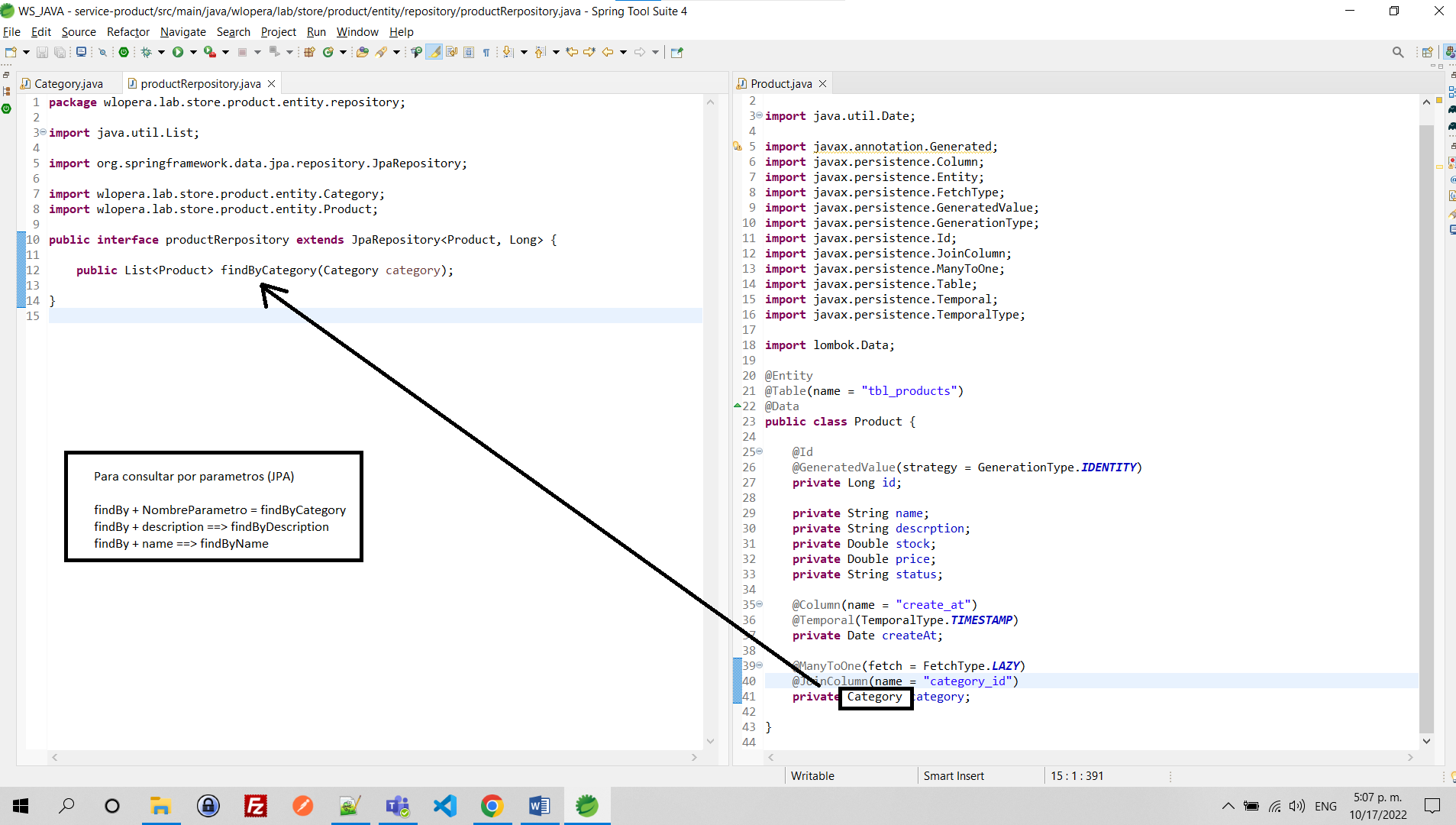
Uso de lombok





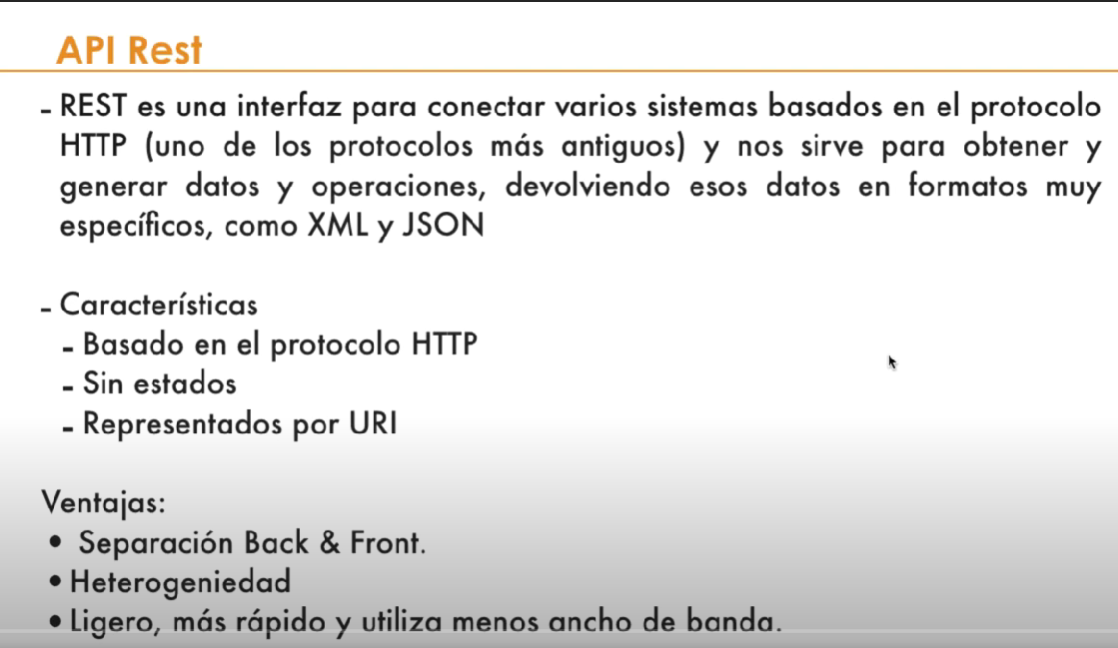
Crear las entidades



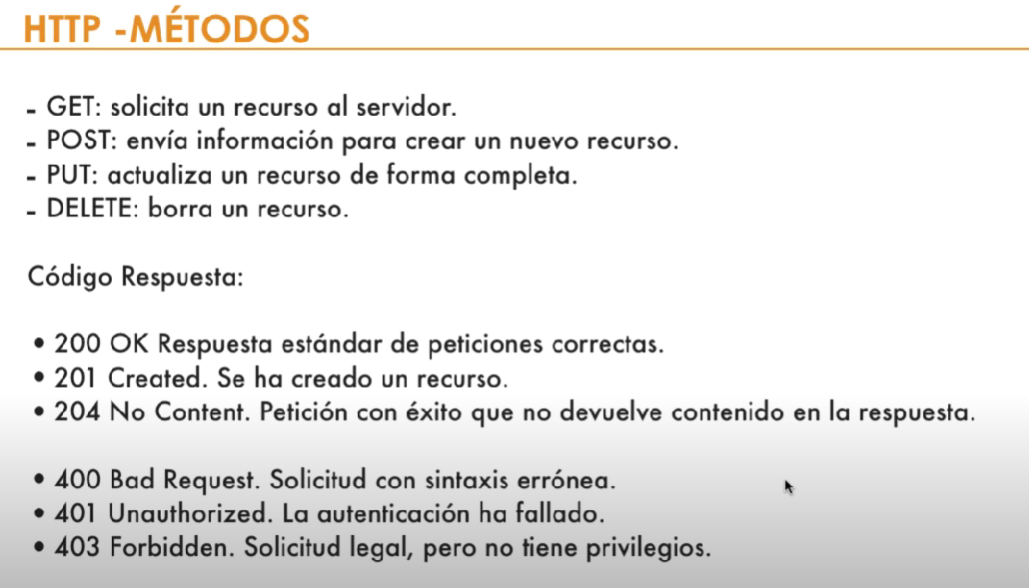


Crear Servicios (Repositorios, servicios y api Rest (controladores)).

**Api Rest**

****

****

****

#Application Configuration

server:

port: 8091

# To See H2 Console in Browser:

# http://localhost:8091/h2-console

# Enabling H2 Console

spring:

h2:

console:

enabled: true

# ===============================

# DB

# ===============================

datasource:

url: jdbc:h2:mem:productDB;DB\_CLOSE\_DELAY=-1;DB\_CLOSE\_ON\_EXIT=FALSE

driverClassName: org.h2.Driver

username: sa

password:

# ===============================

# JPA / HIBERNATE

# ===============================

jpa:

show-sql: true

defer-datasource-initialization: true

hibernate:

ddl-auto: update

database-platform: org.hibernate.dialect.H2Dialect

properties:

hibernate:

show\_sql: true

use\_sql\_comments: false

format\_sql: false

sql:

init:

platform: h2

logging:

pattern:

console: "%d{yyyy-MM-dd HH:mm:ss} ${LOG\_LEVEL\_PATTERN:-%5p} %m%n"

**Hibernate y *data.sql***

De forma predeterminada, **el script *data.sql se* ejecuta antes de la inicialización de Hibernate** . Esto alinea la inicialización basada en secuencias de comandos con otras herramientas de migración de bases de datos, como [Flyway](https://www.baeldung.com/database-migrations-with-flyway) y [Liquibase](https://www.baeldung.com/liquibase-refactor-schema-of-java-app) . Como estamos recreando el esquema generado por Hibernate cada vez, necesitamos establecer una propiedad adicional:

spring.jpa.defer-datasource-initialization=true

**data.sql**

INSERT INTO tbl\_categories (id, name) VALUES (1, 'shoes');

INSERT INTO tbl\_categories (id, name) VALUES (2, 'books');

INSERT INTO tbl\_categories (id, name) VALUES (3, 'electronics');

INSERT INTO tbl\_products (id, name, description, stock,price,status, create\_at,category\_id)

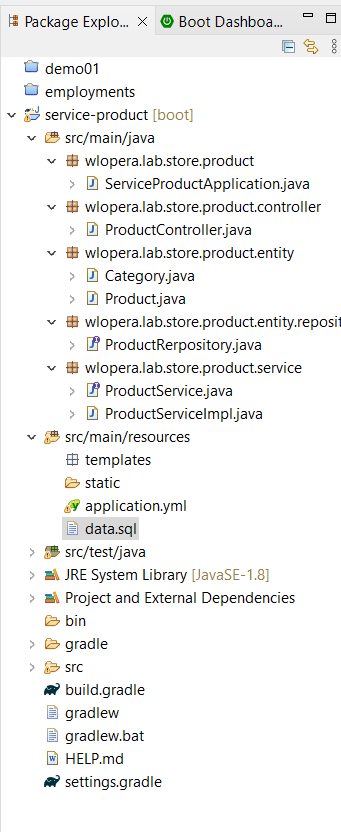
VALUES (1, 'adidas Cloudfoam Ultimate','Walk in the air in the black / black CLOUDFOAM ULTIMATE running shoe from ADIDAS',5,178.89,'CREATED','2018-09-05',1);

INSERT INTO tbl\_products (id, name, description, stock,price,status, create\_at,category\_id)

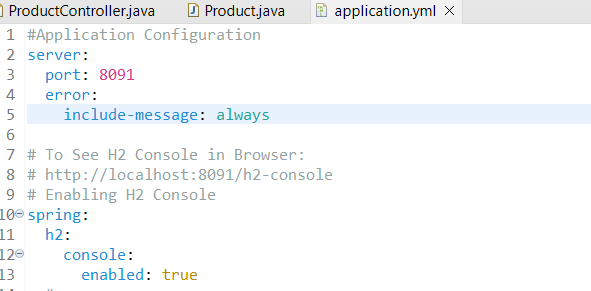
VALUES (2, 'under armour Men ''s Micro G Assert – 7','under armour Men ''Lightweight mesh upper delivers complete breathability . Durable leather overlays for stability ',4,12.5,'CREATED','2018-09-05',1);

INSERT INTO tbl\_products (id, name, description, stock,price,status, create\_at,category\_id)

VALUES (3, 'Spring Boot in Action','under armour Men '' Craig Walls is a software developer at Pivotal and is the author of Spring in Action',12,40.06,'CREATED','2018-09-05',2);

****

**Lanzar mensajes de error (Postman console):**

****

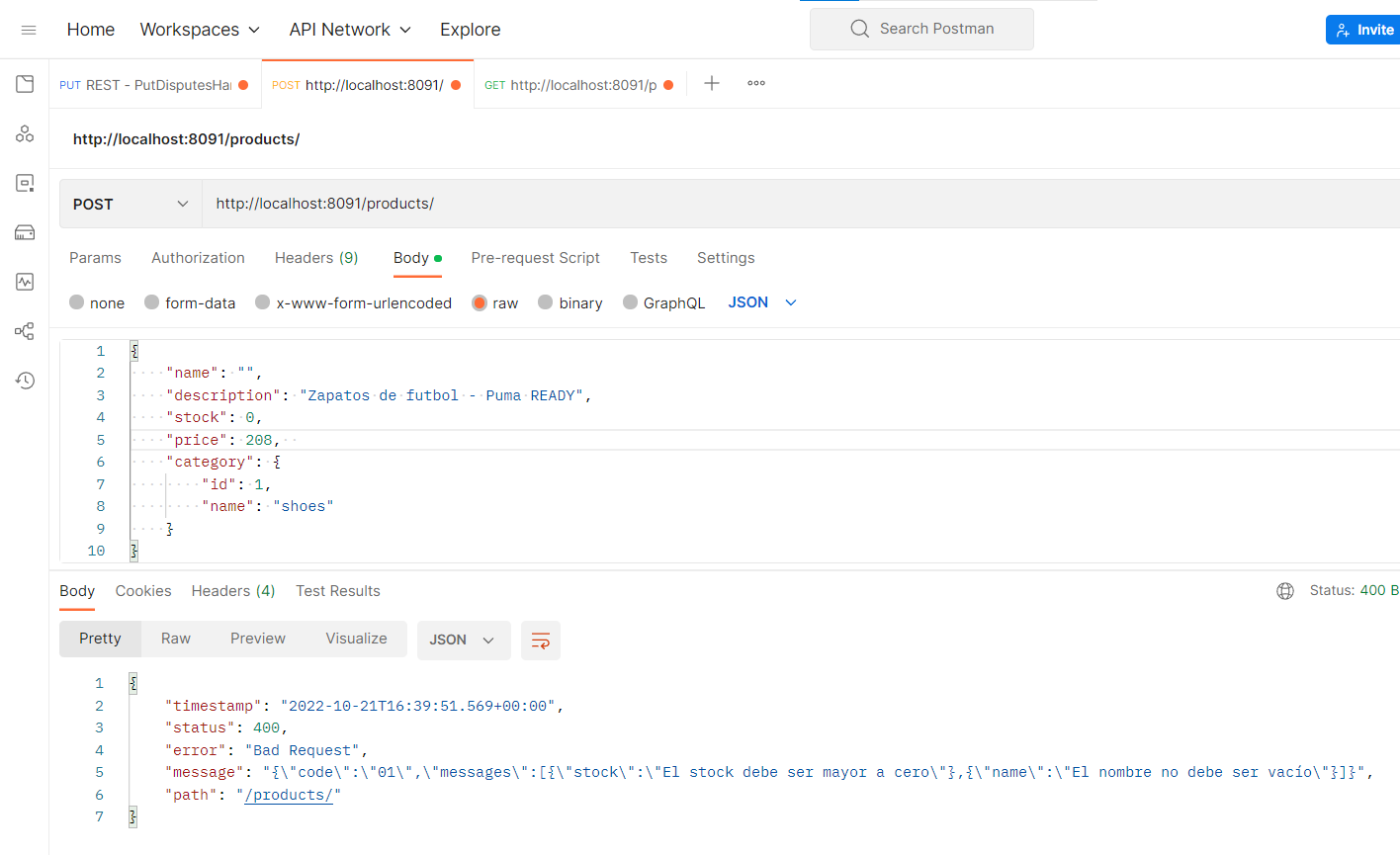
#Application Configuration

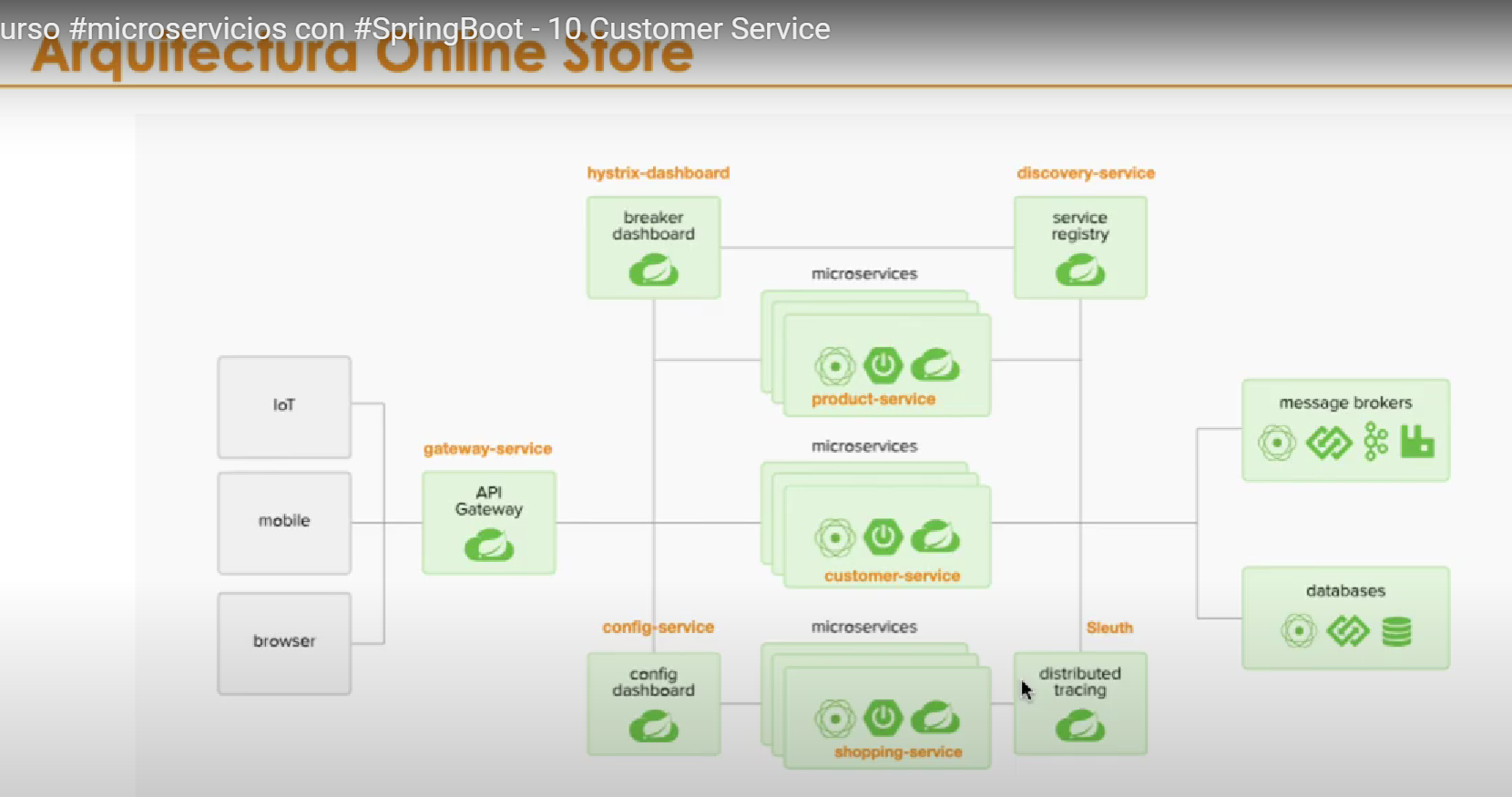
server:

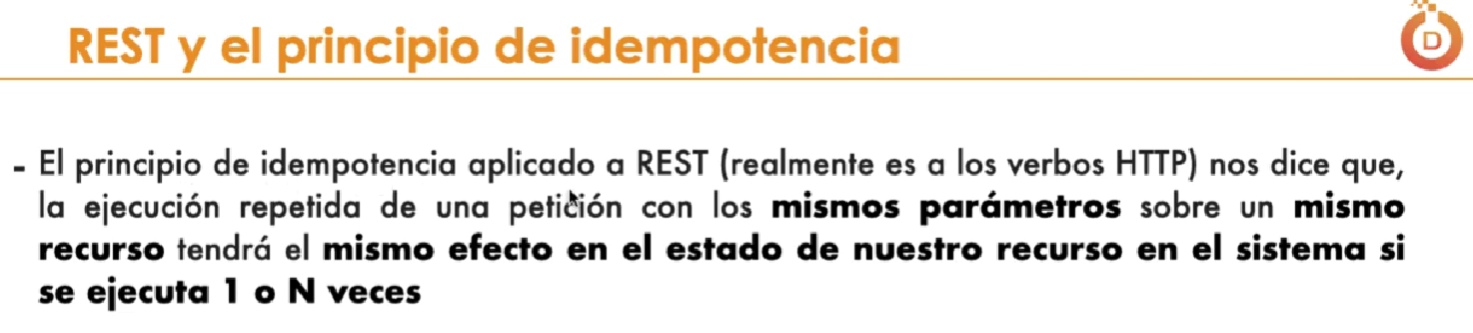
port: 8091

error:

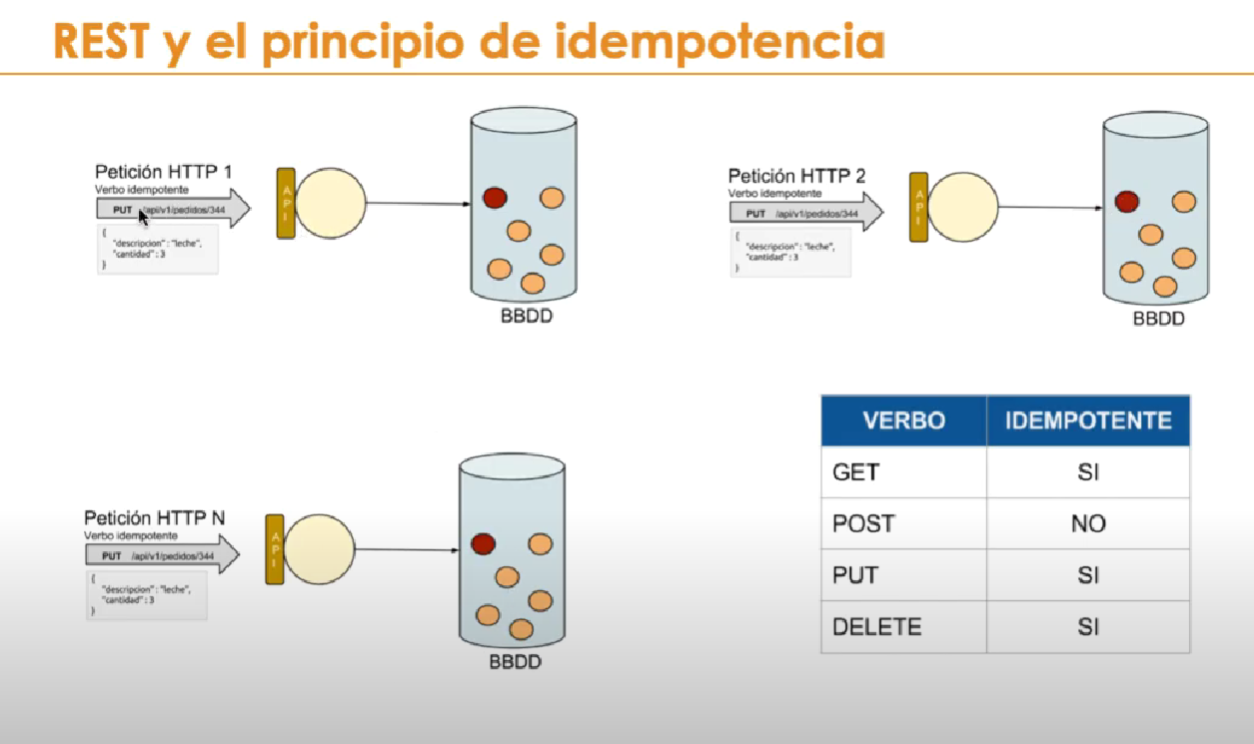
include-message: always

****

****

****

****

****

**Request Invoice:**

**POST:** http://localhost:8093/invoices

{

    "numberInvoice": "002",

    "description": "invoice store",

    "customerId": 2,

    "items": [

        {

            "quantity": 1,

            "price": 178.89

        },

        {

            "quantity": 2,

            "price": 40.06

        }

    ]

}

**Response:**

{

    "timestamp": "2022-10-25T16:04:55.975+00:00",

    "status": 400,

    "error": "Bad Request",

    "message": "{\"code\":\"01\",\"messages\":[{\"items[0].quantity\":\"El stock debe ser mayor que cero\"}]}",

    "path": "/invoices"

}

**Request:**

{

    "numberInvoice": "002",

    "description": "invoice store",

    "customerId": 2,

    "items": [

        {

            "quantity": 1,

            "price": 178.89

        },

        {

            "quantity": 2,

            "price": 40.06

        }

    ]

}

**Response:**

{

    "id": 2,

    "numberInvoice": "002",

    "description": "invoice store",

    "customerId": 2,

    "createAt": "2022-10-25",

    "items": [

        {

            "id": 4,

            "quantity": 1.0,

            "price": 178.89,

            "productId": **null**,

            "subTotal": 178.89

        },

        {

            "id": 5,

            "quantity": 2.0,

            "price": 40.06,

            "productId": **null**,

            "subTotal": 80.12

        }

    ],

    "state": "CREATED"

}

**Request Customer:**

POST: <http://localhost:8092/customers>

{

    "firstName":"william",

    "lastName":"Lopera",

    "numberID": "34254678",

    "email":"wlopera@test.com",

    "photoUrl":"",

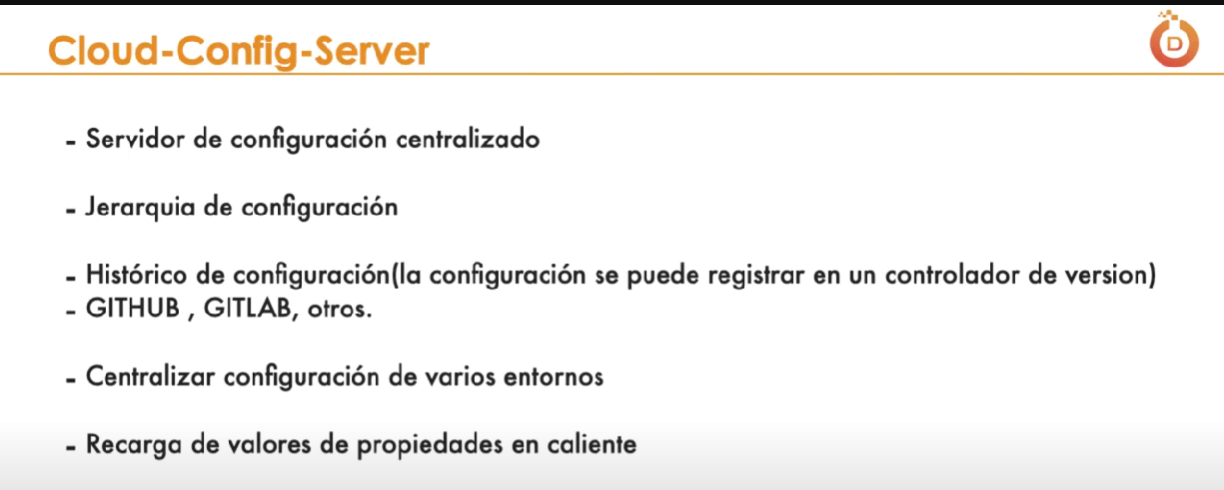
    "region":{

        "id":2,

        "name":"CentroAmérica"

    }

}

****