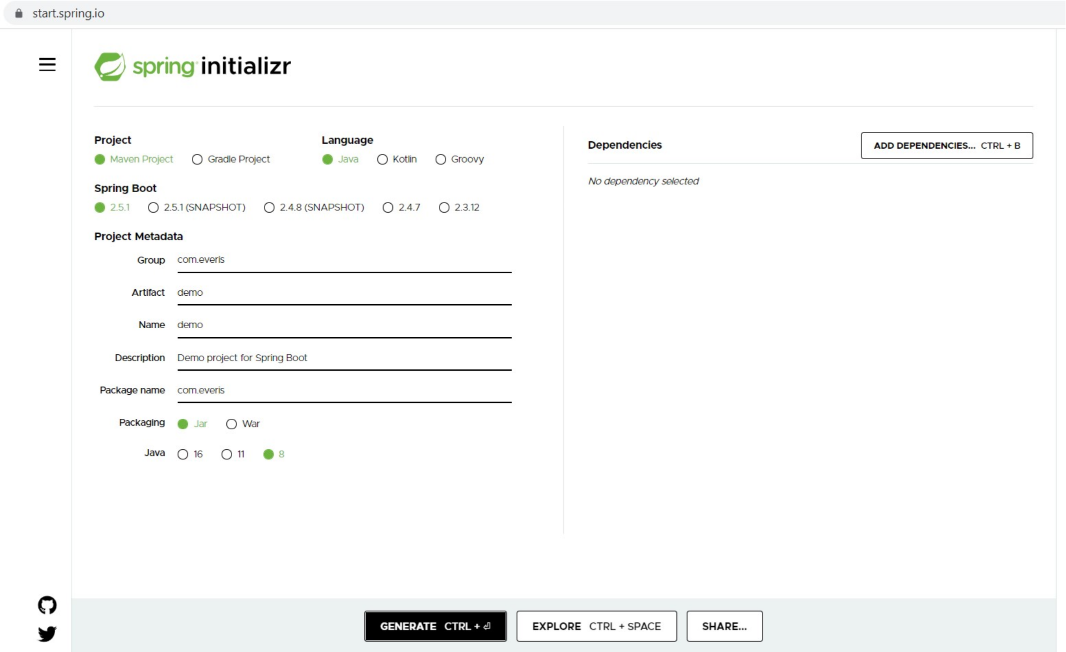
Creación de proyecto: se puede crear a partir del spring initializr en la siguente url start.spring.io


start.spring.io

spring initializr

Project
● Maven Project ○ Gradle Project

Language
● Java ○ Kotlin ○ Groovy

Spring Boot
● 2.5.1 ○ 2.5.1 (SNAPSHOT) ○ 2.4.8 (SNAPSHOT) ○ 2.4.7 ○ 2.3.12

Project Metadata

Group: com.everis

Artifact: demo

Name: demo

Description: Demo project for Spring Boot

Package name: com.everis

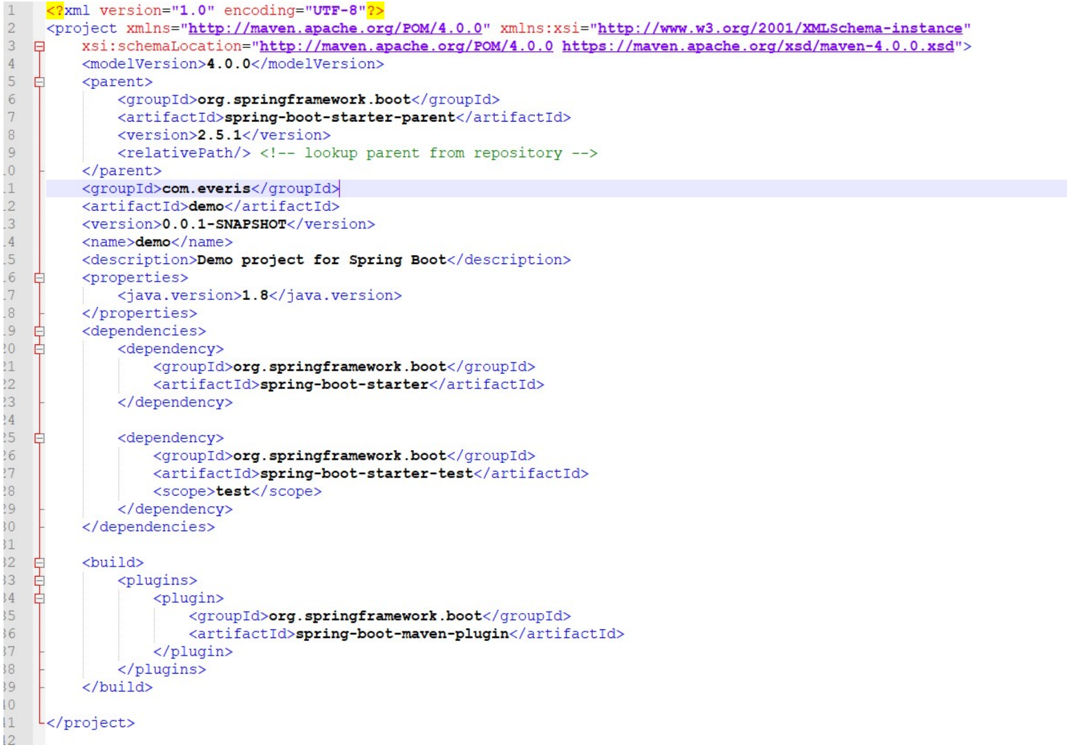
Packaging: ● Jar ○ War

Java: ○ 16 ○ 11 ● 8

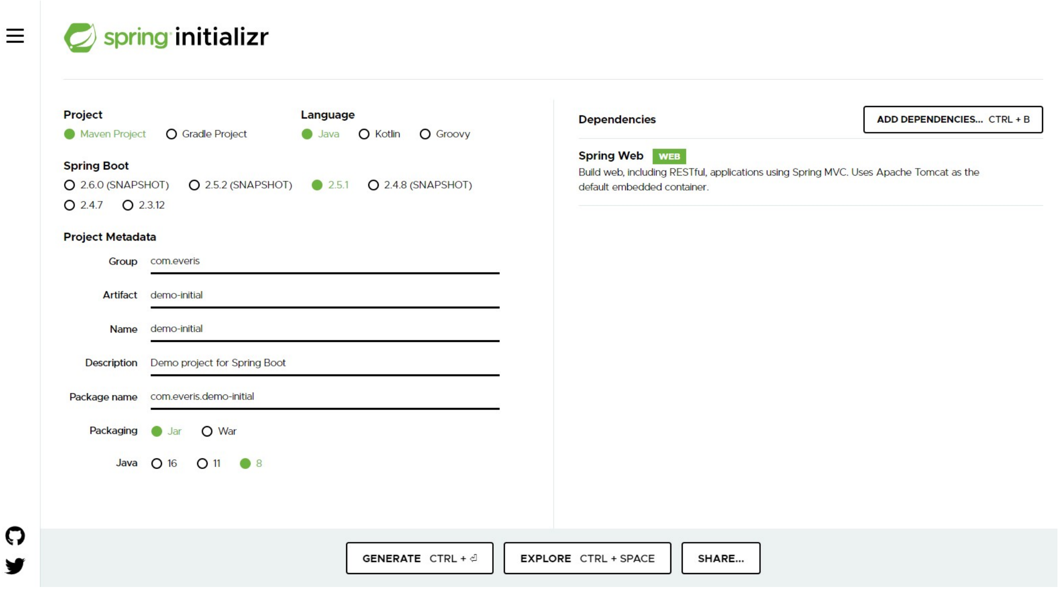
Dependencies

No dependency selected

Podemos notar las 2 dependecias base y el build de maven.


```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.everis</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Para desarrollo de aplicaciones rest vamos agregar la dependencia **Spring web**



Project
● Maven Project ○ Gradle Project

Language
● Java ○ Kotlin ○ Groovy

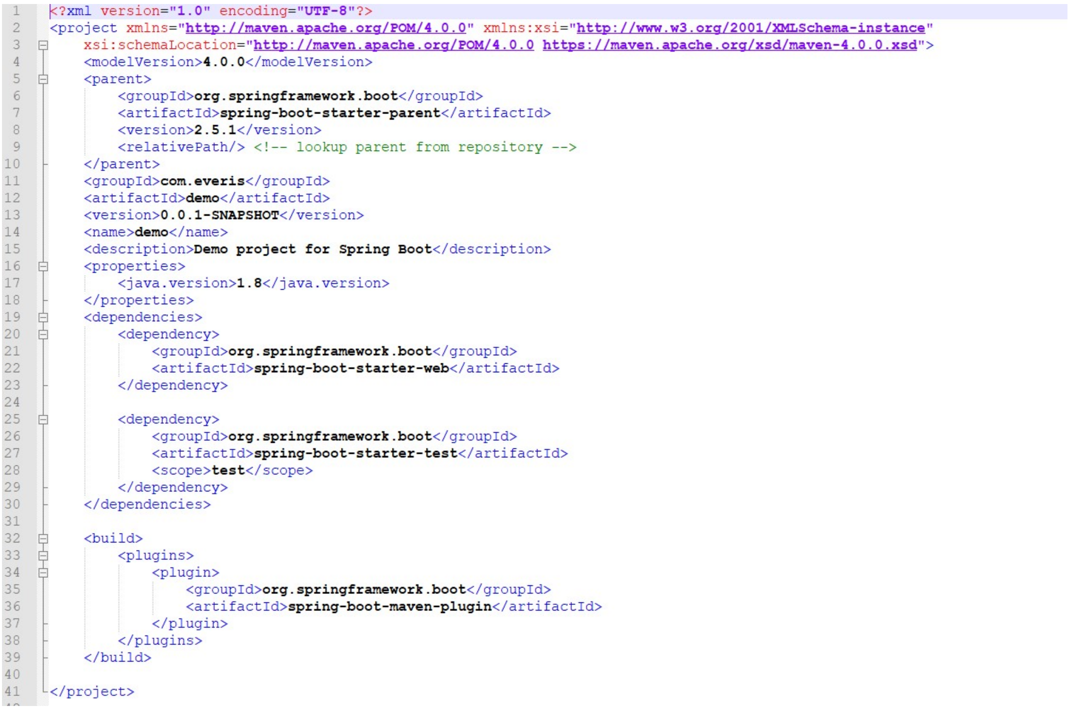
Spring Boot
○ 2.6.0 (SNAPSHOT) ○ 2.5.2 (SNAPSHOT) ● 2.5.1 ○ 2.4.8 (SNAPSHOT)
○ 2.4.7 ○ 2.3.12

Project Metadata

Group: com.everis
Artifact: demo-initial
Name: demo-initial
Description: Demo project for Spring Boot
Package name: com.everis.demo-initial
Packaging: ● Jar ○ War
Java: ○ 16 ○ 11 ● 8

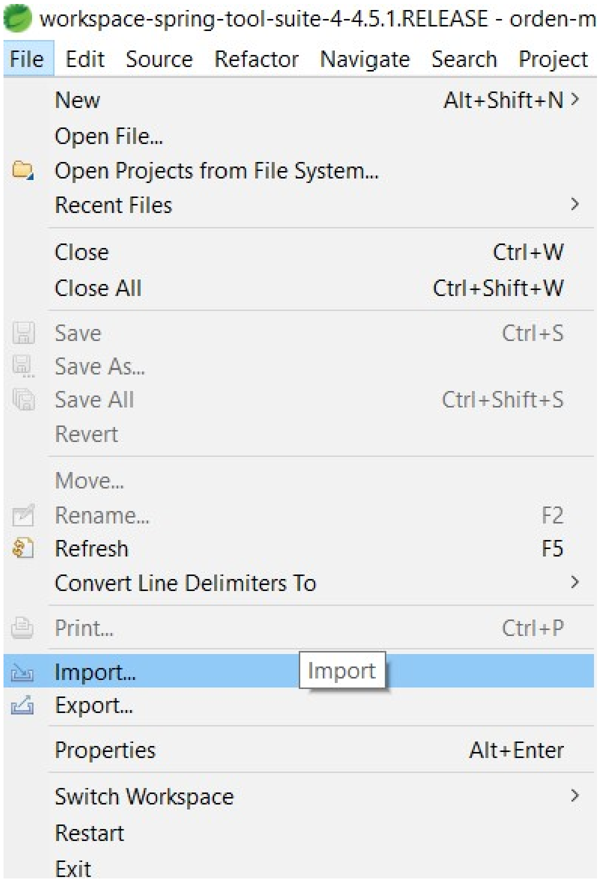
Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

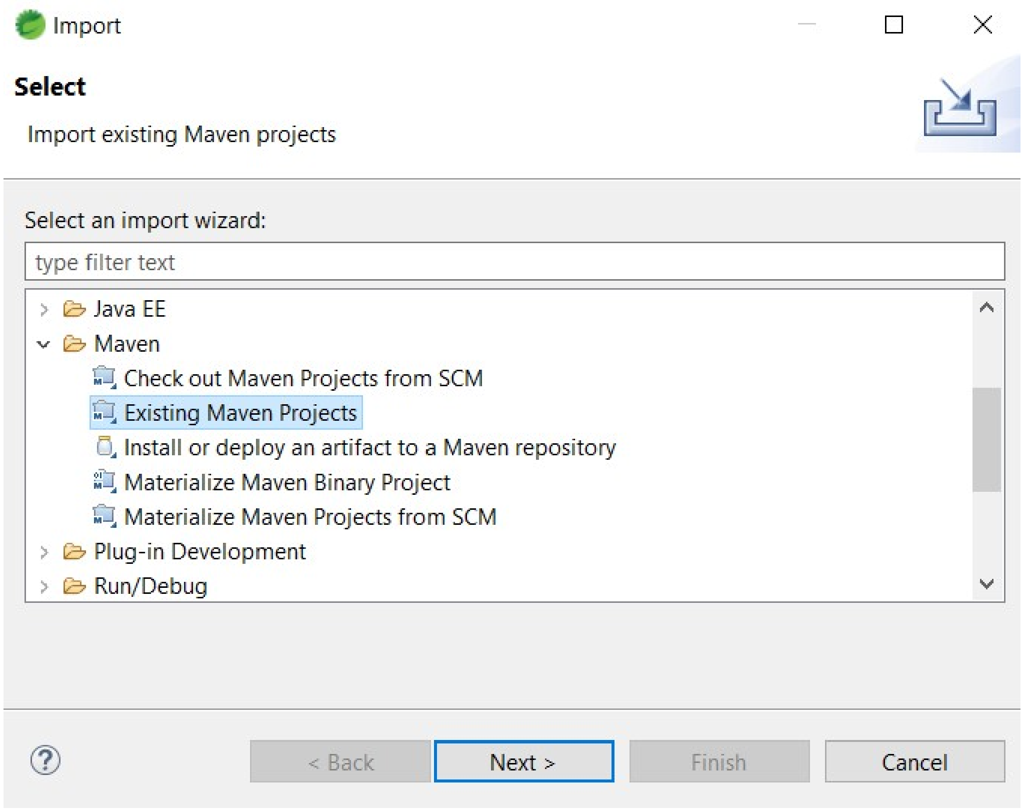


```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.everis</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

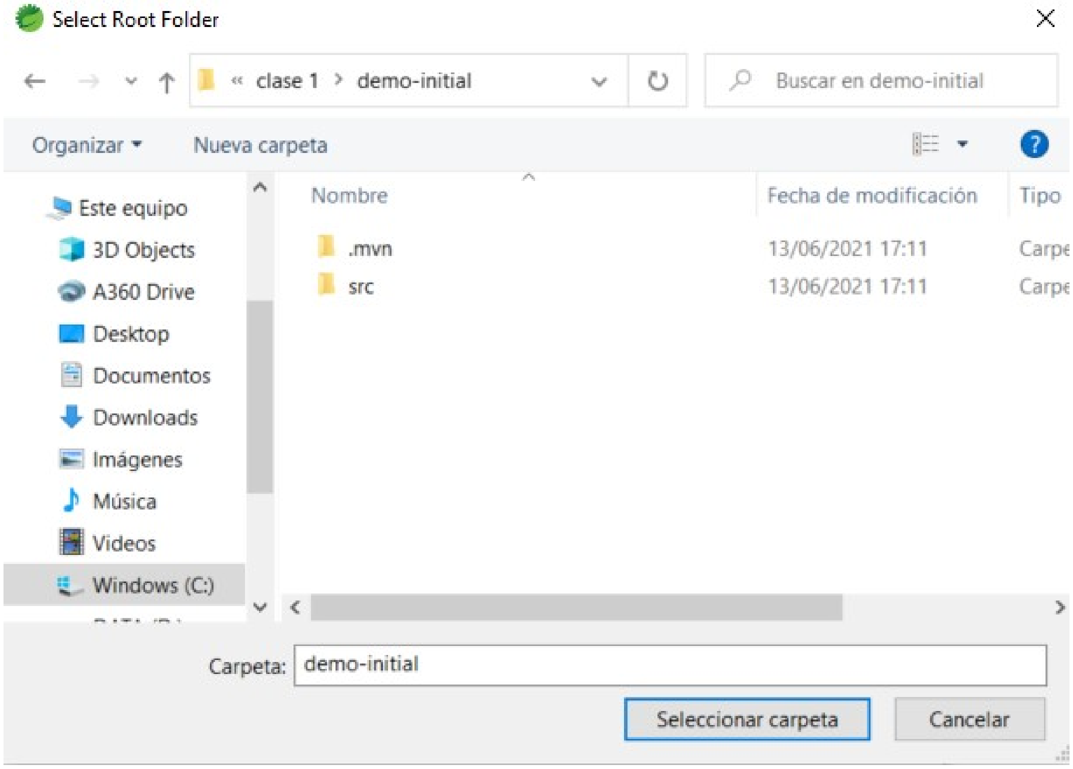
Importación del proyecto generado en el ide STS.



Seleccionar el proyecto maven existente.

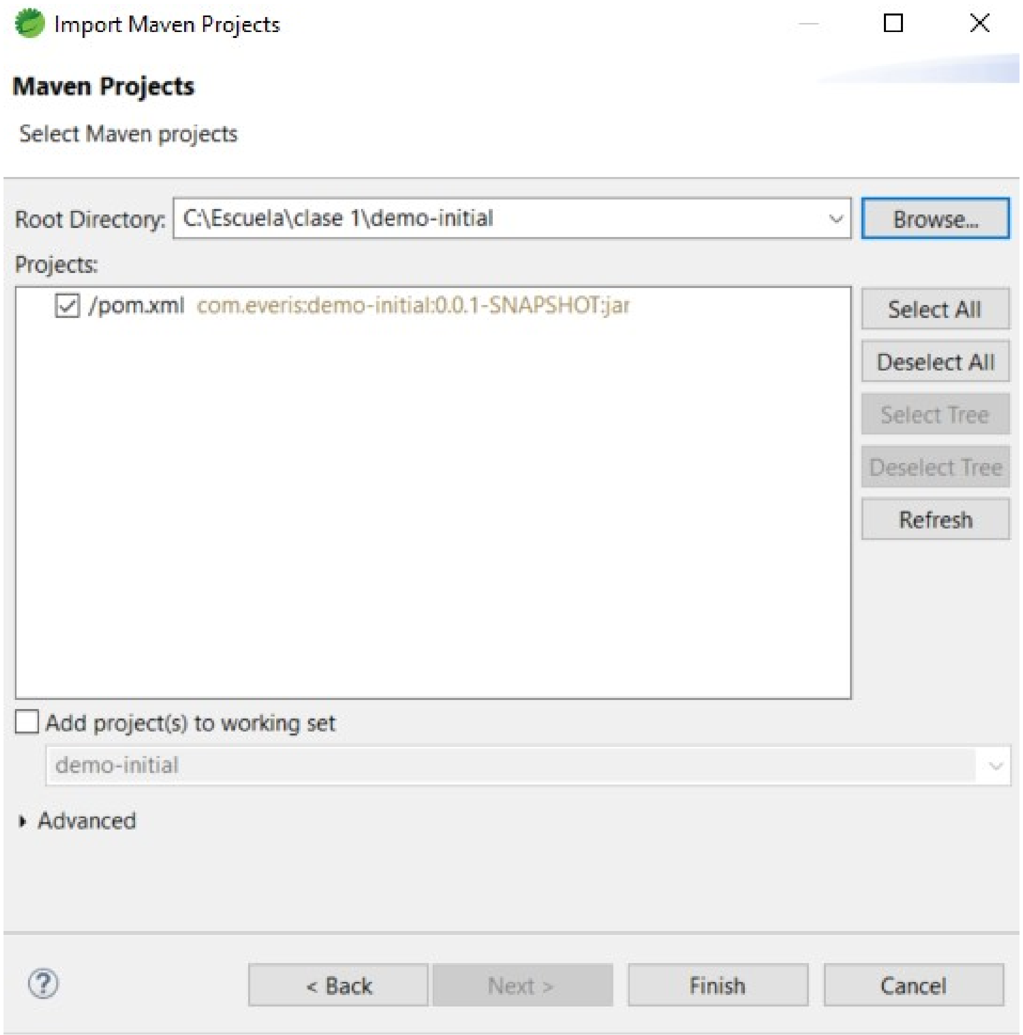

Import
Select
Import existing Maven projects
Select an import wizard:
type filter text
Java EE
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Binary Project
Materialize Maven Projects from SCM
Plug-in Development
Run/Debug
?
< Back
Next >
Finish
Cancel

Seleccionar la carpeta donde se encuentra el proyecto (archivo pom.xml)



Nombre	Fecha de modificación	Tipo
.mvn	13/06/2021 17:11	Carpe
src	13/06/2021 17:11	Carpe

Carpeta: demo-initial



Maven Projects

Select Maven projects

Root Directory: C:\Escuela\clase 1\demo-initial

Projects:

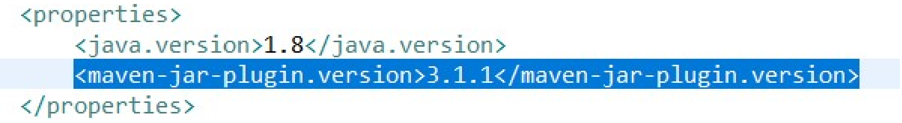
☑ /pom.xml com.everis:demo-initial:0.0.1-SNAPSHOT:jar

□ Add project(s) to working set

demo-initial

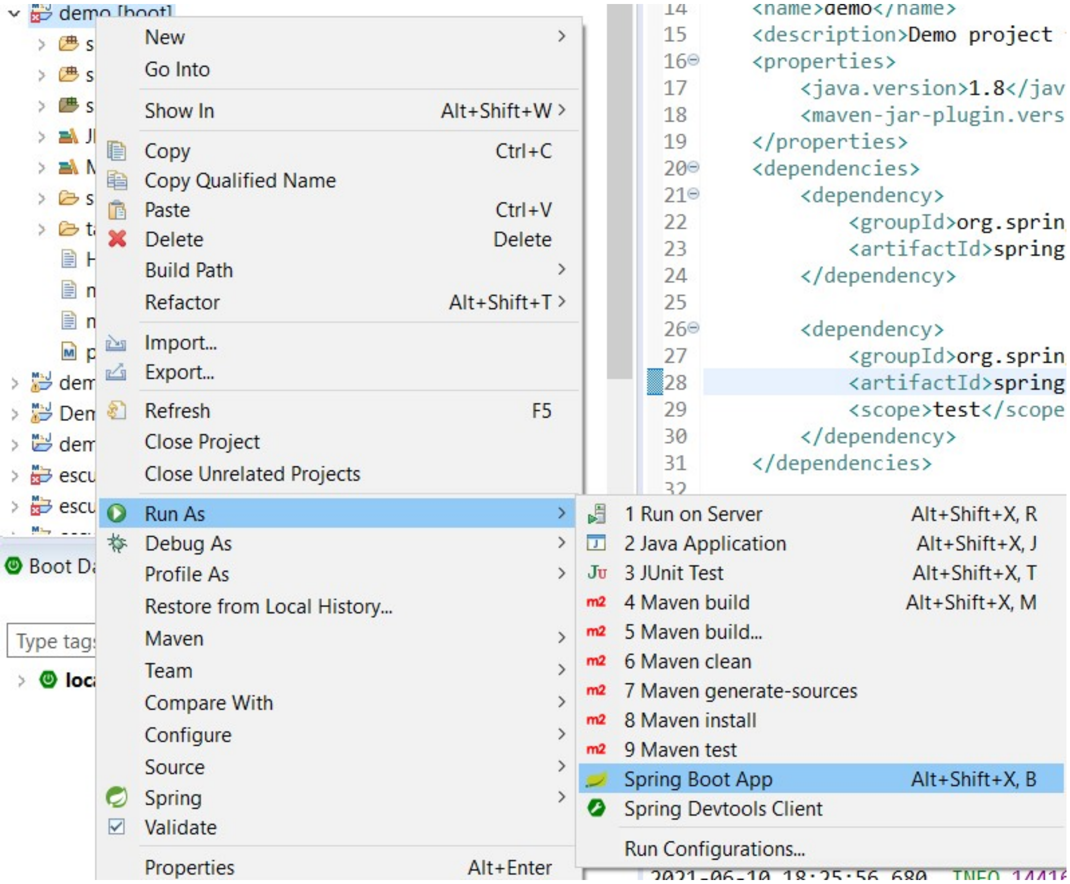
Para versiones a partir de la versión 2.1.5, el pom te marca errores, para ello se agrega la siguiente línea de la versión de maven en el pom.xml

<maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>



```
<properties>
    <java.version>1.8</java.version>
    <maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>
</properties>
```

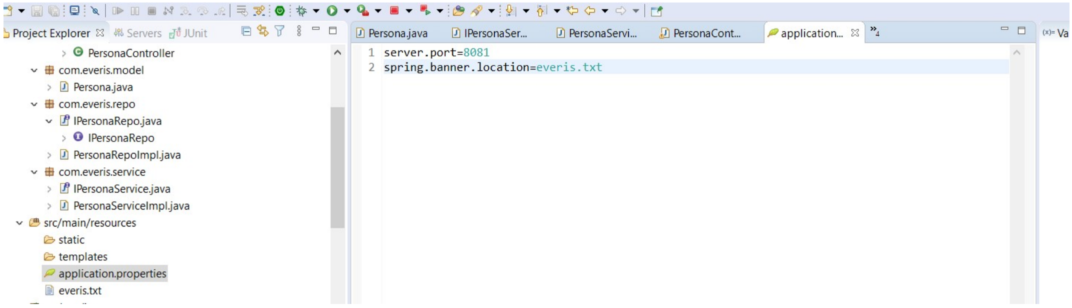
Ejecución del proyecto:



```
14      <name>demo</name>
15      <description>Demo project
16      <properties>
17          <java.version>1.8</jav
18          <maven-jar-plugin.vers
19      </properties>
20      <dependencies>
21          <dependency>
22              <groupId>org.sprin
23              <artifactId>spring
24          </dependency>
25
26          <dependency>
27              <groupId>org.sprin
28              <artifactId>spring
29              <scope>test</scope
30          </dependency>
31      </dependencies>
32
```

```
2021-06-10 18:25:56.680  INFO 14416
```

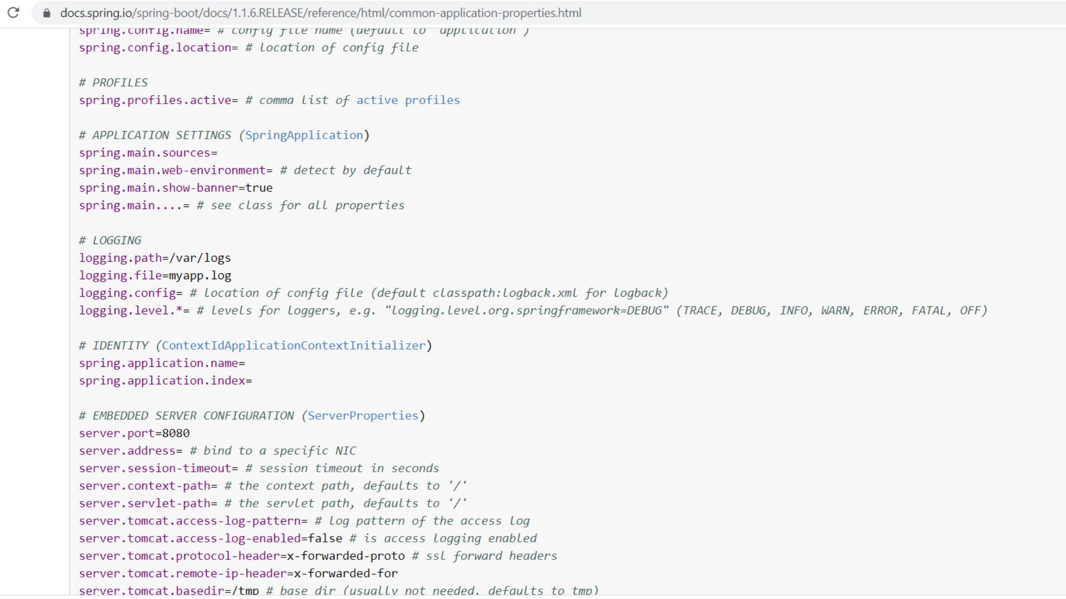
La configuración se realiza en el archivo **Application.properties**



```
server.port=8081
spring.banner.location=everis.txt
```

Documentación de la configuración de properties :

https://docs.spring.io/spring-boot/docs/1.1.6.RELEASE/reference/html/common-application-properties.html



docs.spring.io/spring-boot/docs/1.1.6.RELEASE/reference/html/common-application-properties.html

```
spring.config.name= # config file name (default to 'application')
spring.config.location= # location of config file

# PROFILES
spring.profiles.active= # comma list of active profiles

# APPLICATION SETTINGS (SpringApplication)
spring.main.sources=
spring.main.web-environment= # detect by default
spring.main.show-banner=true
spring.main....= # see class for all properties

# LOGGING
logging.path=/var/logs
logging.file=myapp.log
logging.config= # location of config file (default classpath:logback.xml for logback)
logging.level.*= # levels for loggers, e.g. "logging.level.org.springframework=DEBUG" (TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF)

# IDENTITY (ContextIdApplicationContextInitializer)
spring.application.name=
spring.application.index=

# EMBEDDED SERVER CONFIGURATION (ServerProperties)
server.port=8080
server.address= # bind to a specific NIC
server.session-timeout= # session timeout in seconds
server.context-path= # the context path, defaults to '/'
server.servlet-path= # the servlet path, defaults to '/'
server.tomcat.access-log-pattern= # log pattern of the access log
server.tomcat.access-log-enabled=false # is access logging enabled
server.tomcat.protocol-header=x-forwarded-proto # ssl forward headers
server.tomcat.remote-ip-header=x-forwarded-for
server.tomcat.basedir=/tmp # base dir (usually not needed, defaults to tmp)
```

Para cambiar el banner en la consola, la siguiente url te genera el texto para agregar en un archivo.

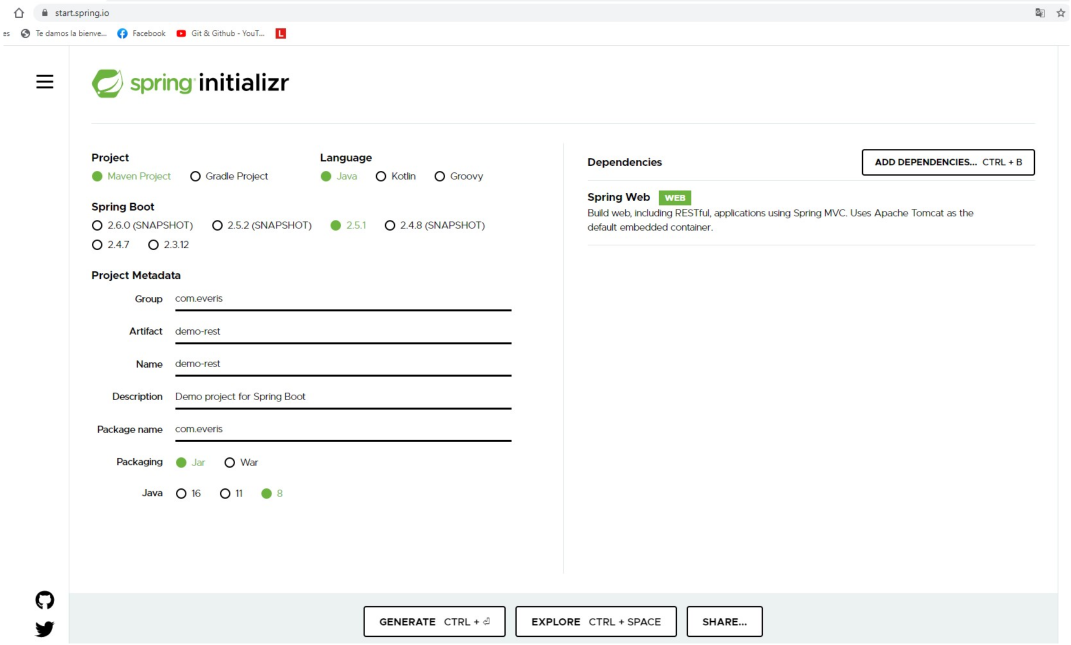
https://devops.datenkollektiv.de/banner.txt/index.html

Agregar el pom, si que se compile **automáticamente** ante cualquier cambio, se debe utilizar en desarrollo y que tenga una buena máquina. En desarrollo es interesante usarlo.

```
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
</dependency>
```

Proyecto Demo 1:

1.- Creación de proyecto


start.spring.io
Te damos la bienve... Facebook Git & Github - YouT...
spring initializr
ADD DEPENDENCIES... CTRL + B
GENERATE CTRL + ⏎
EXPLORE CTRL + SPACE
SHARE...

Project ● Maven Project ○ Gradle Project

Language ● Java ○ Kotlin ○ Groovy

Spring Boot ○ 2.6.0 (SNAPSHOT) ○ 2.5.2 (SNAPSHOT) ● 2.5.1 ○ 2.4.8 (SNAPSHOT) ○ 2.4.7 ○ 2.3.12

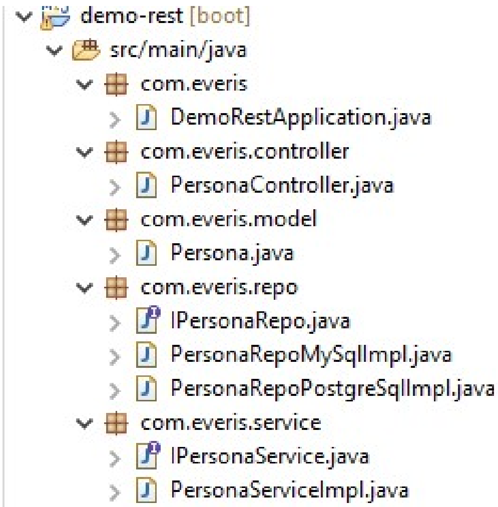
Project Metadata

Group	com.everis
Artifact	demo-rest
Name	demo-rest
Description	Demo project for Spring Boot
Package name	com.everis
Packaging	● Jar ○ War
Java	○ 16 ○ 11 ● 8

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

2.- Creación de paquetes



```
demo-rest [boot]
  src/main/java
    com.everis
      DemoRestApplication.java
    com.everis.controller
      PersonaController.java
    com.everis.model
      Persona.java
    com.everis.repo
      IPersonaRepo.java
      PersonaRepoMySqlImpl.java
      PersonaRepoPostgreSqlImpl.java
    com.everis.service
      IPersonaService.java
      PersonaServiceImpl.java
```

3. implementar clase modelo : `la clase persona`

```
package com.everis.model;

publicclass Persona {
	private Integer idPersona;
	private String nombres;

	public Integer getIdPersona() {
		returnidPersona;
	}

	public String getNombres() {
		returnnombres;
	}

	publicvoid setIdPersona(Integer idPersona) {
		this.idPersona = idPersona;
```

```
        }

        publicvoid setNombres(String nombres) {
                this.nombres = nombres;
        }
}
```

4 creación de Interfaz repositorio

```
package com.everis.repo;

public interface IPersonaRepo {

  void saludar();

  String despedir();

  List<Persona> listarPersonas();
}
```

5 Implementación de repositorios:

```
@Repository
@Qualifier("personaMySql")
public class PersonaRepoMySqlImpl implements IPersonaRepo {

        @Override
        public void saludar() {
                System.out.print("Hola Nttd");

        }

        @Override
        public String despedir() {
                return "Bye Nttd";
        }

        @Override
        public List<Persona> listarPersonas() {
                List<Persona> personas = new ArrayList<Persona>();
                Persona persona = new Persona();
                persona.setIdPersona(2);
                persona.setNombres("Nttd");
                personas.add(persona);
                return personas;
        }
}


@Repository
@Qualifier("personaPostgreSql")
public class PersonaRepoPostgreSqlImpl implements IPersonaRepo {

        @Override
        public void saludar() {
                System.out.print("Hola Everis");

        }

        @Override
        public String despedir() {
```

```
		return "Bye Everis";
	}

	@Override
	public List<Persona> listarPersonas() {
		List<Persona> personas = new ArrayList<Persona>();
		Persona persona = new Persona();
		persona.setIdPersona(1);
		persona.setNombres("Everis");
		personas.add(persona);
		return personas;
	}
}
```

6 creación de Interfaz service :

```
package com.everis.service;

public interface IPersonaService {
 public void saludar();
 public String despedir();
 public List<Persona> listarPersonas();
}
```

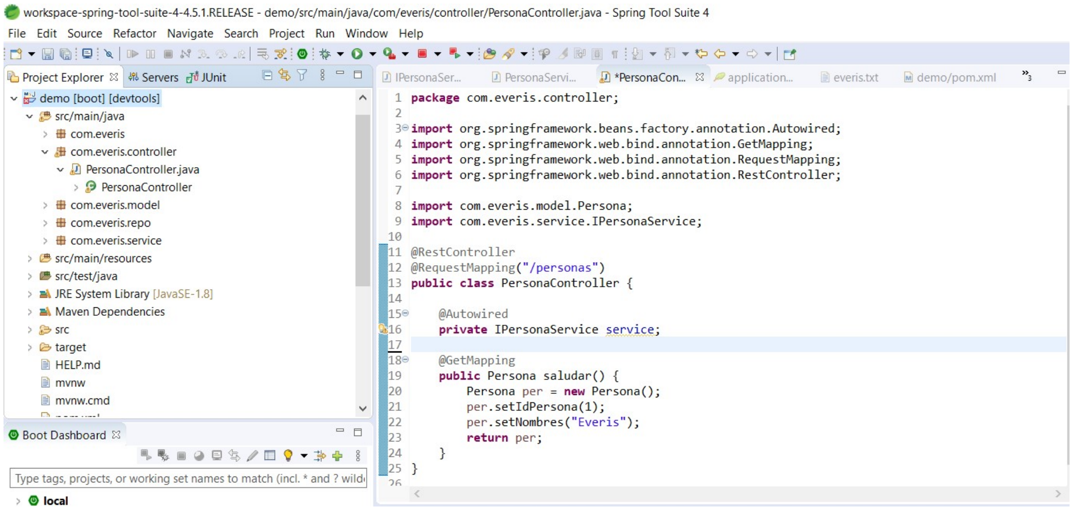
7. Implementación del service:

```
package com.everis.service;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;
import com.everis.model.Persona;
import com.everis.repo.IPersonaRepo;

@Service
public class PersonaServiceImpl implements IPersonaService {

	@Autowired
	@Qualifier("personaMySql")
	private IPersonaRepo repo;

	@Override
	public void saludar() {
		this.repo.saludar();
	}

	@Override
	public String despedir() {
		return this.repo.despedir();
	}

	@Override
	public List<Persona> listarPersonas() {
		return this.repo.listarPersonas();
	}

}
```

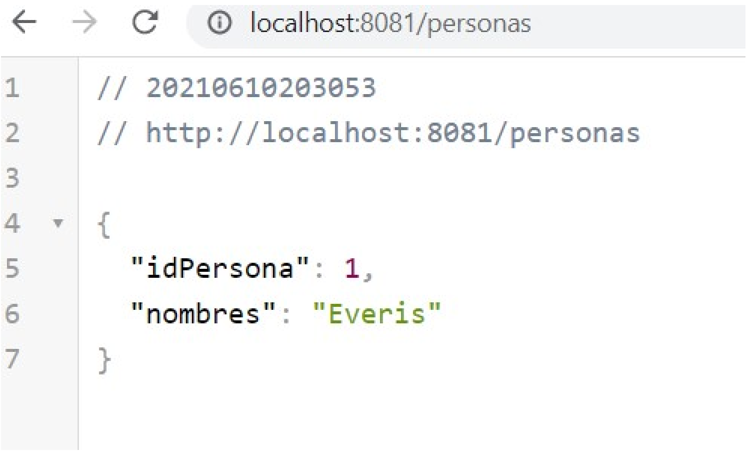
7 implementar la clase controller:

```
import org.springframework.web.bind.annotation.RestController;

import com.everis.model.Persona;
import com.everis.service.IPersonaService;

@RestController
@RequestMapping("/personas")
publicclass PersonaController {

	@Autowired
	private IPersonaService service;

@GetMapping
publicvoid saludar() {
	this.service.saludar();
    }

}
```

Retornando un json :



```
package com.everis.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.everis.model.Persona;
import com.everis.service.IPersonaService;

@RestController
@RequestMapping("/personas")
public class PersonaController {

    @Autowired
    private IPersonaService service;

    @GetMapping
    public Persona saludar() {
        Persona per = new Persona();
        per.setIdPersona(1);
        per.setNombres("Everis");
        return per;
    }
}
```



```
// 20210610203053
// http://localhost:8081/personas

{
  "idPersona": 1,
  "nombres": "Everis"
}
```

Para retornar en formato xml, se cambia los siguiente :

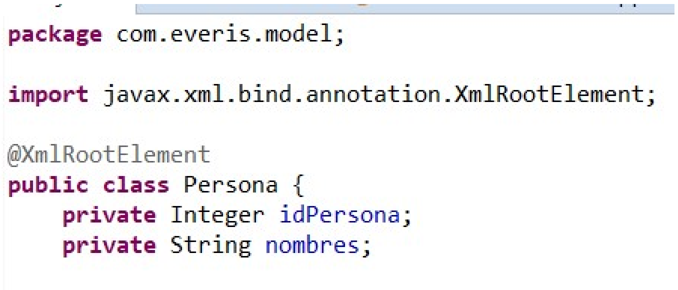
1.- En el controler :

```
@GetMapping(produces = "application/xml")
public Persona saludar() {
        Persona per = new Persona();
        per.setIdPersona(1);
        per.setNombres("Everis");
        returnper;
    }
```

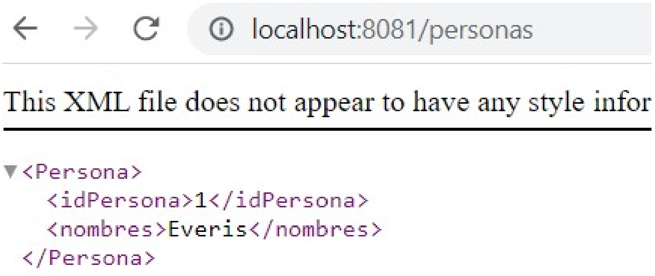
2.- En el pom.xml

```
        <dependency>
              <groupId>com.fasterxml.jackson.dataformat</groupId>
               <artifactId>jackson-dataformat-xml</artifactId>
        </dependency>
```

3.- Al objeto a devolver colocar el @XmlRootElement



```
package com.everis.model;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Persona {
    private Integer idPersona;
    private String nombres;
```



localhost:8081/personas

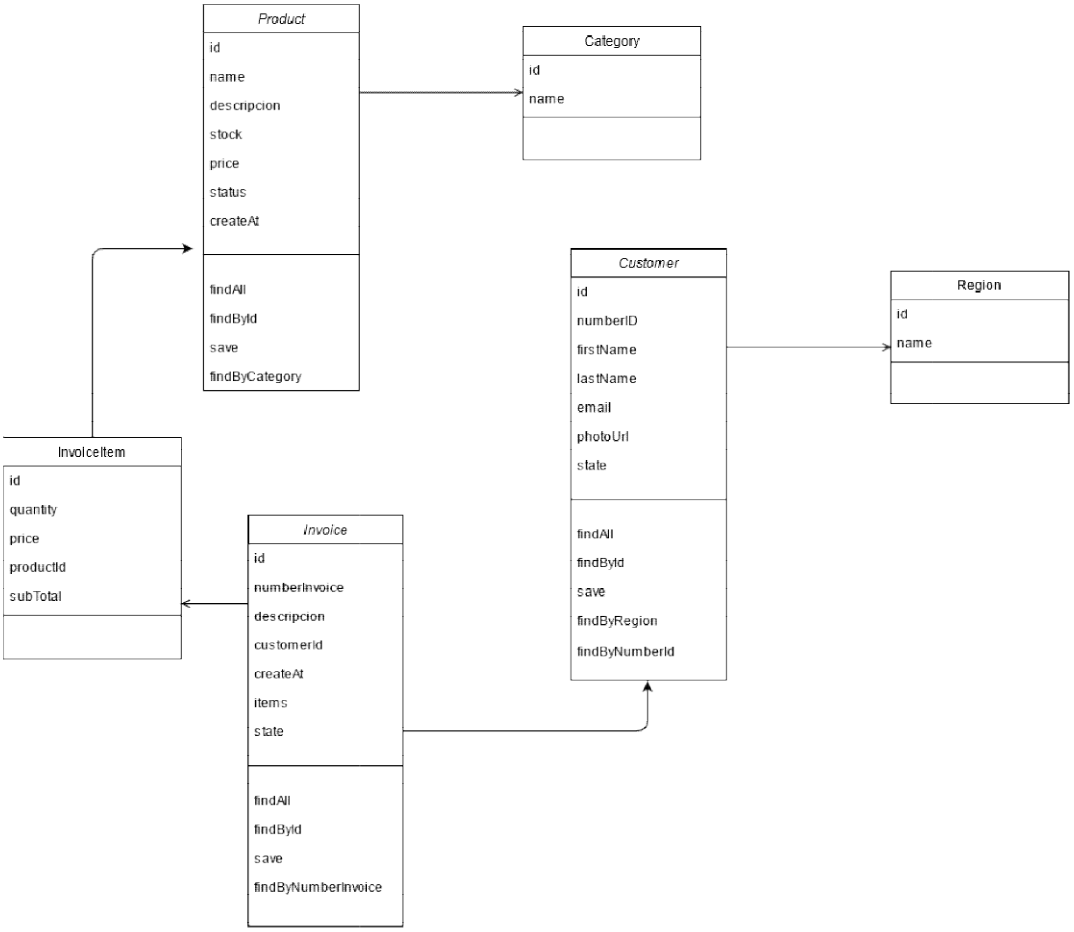
This XML file does not appear to have any style infor

```
▼<Persona>
   <idPersona>1</idPersona>
   <nombres>Everis</nombres>
 </Persona>
```

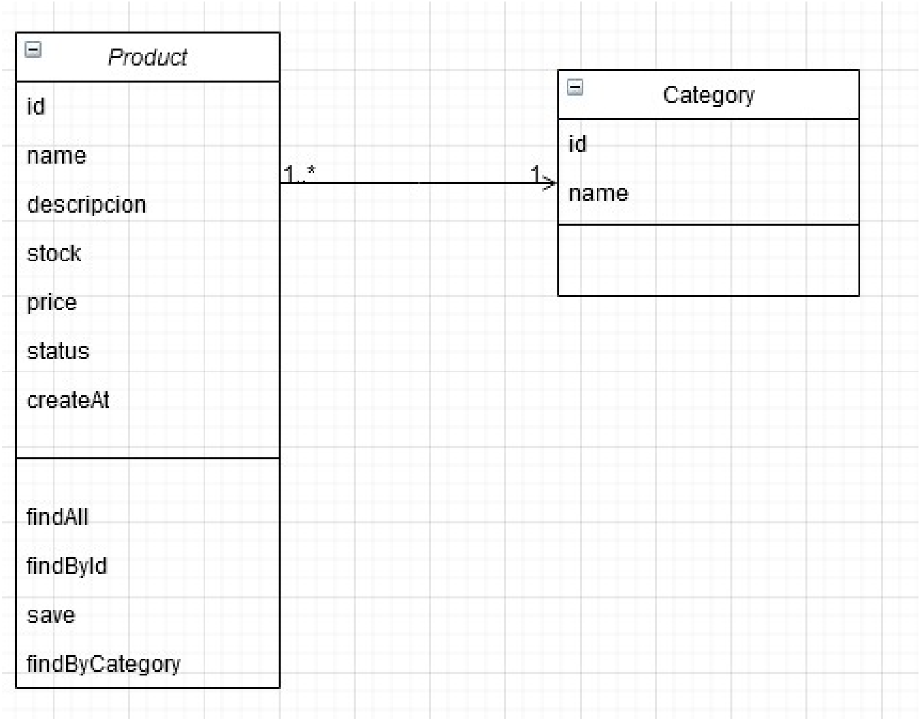
Nota : tener encuenta la siguiente dependencia, que es una librería en java para escribir menos código : constructores, get, set, data.

```
              <dependency>
                     <groupId>org.projectlombok</groupId>
                     <artifactId>lombok</artifactId>
                     <optional>true</optional>
              </dependency>
```

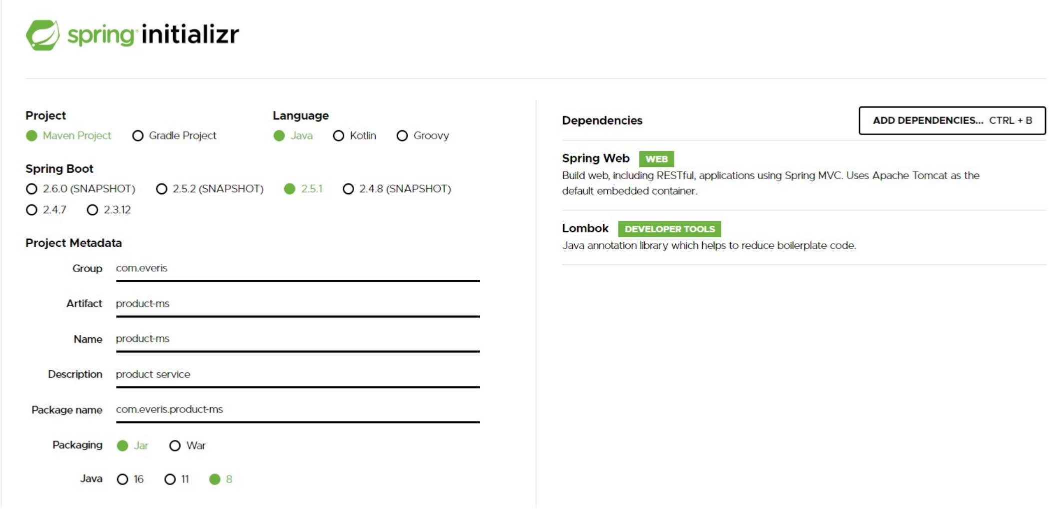
Diagrama del proyecto a desarrollar



Servicios rest : Product-ms



1. Creación de proyecto del microservicio



spring initializr

Project
● Maven Project ○ Gradle Project

Language
● Java ○ Kotlin ○ Groovy

Spring Boot
○ 2.6.0 (SNAPSHOT) ○ 2.5.2 (SNAPSHOT) ● 2.5.1 ○ 2.4.8 (SNAPSHOT)
○ 2.4.7 ○ 2.3.12

Project Metadata

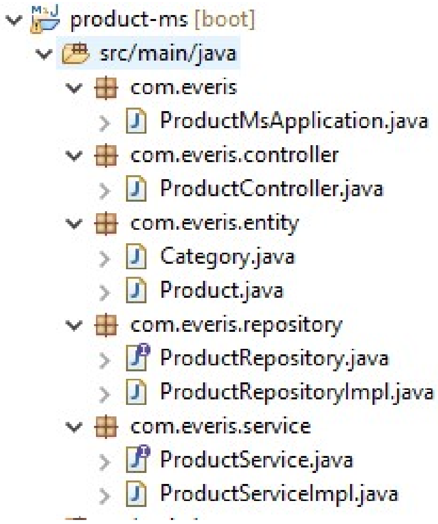
Group	com.everis
Artifact	product-ms
Name	product-ms
Description	product service
Package name	com.everis.product-ms
Packaging	● Jar ○ War
Java	○ 16 ○ 11 ● 8

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

2.- Creación de paquetes: controller, entity, repository y service



```
product-ms [boot]
  src/main/java
    com.everis
      ProductMsApplication.java
    com.everis.controller
      ProductController.java
    com.everis.entity
      Category.java
      Product.java
    com.everis.repository
      ProductRepository.java
      ProductRepositoryImpl.java
    com.everis.service
      ProductService.java
      ProductServiceImpl.java
```

3.- implementación de clase entity: product y Category

```
package com.everis.entity;

import java.util.Date;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor @NoArgsConstructor @Builder
public class Product {

	private Long id;
	private String name;
	private String description;
	private Double stock;
	private Double price;
	private String status;

	private Date createAt;
	private Category category;

}
```

```
package com.everis.entity;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor @NoArgsConstructor @Builder
public class Category {

	private Long id;

	private String name;

}
```

4.- creación de repositorio:

```
package com.everis.repository;

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

import com.everis.entity.Category;
import com.everis.entity.Product;

public interface ProductRepository {
	public List<Product> findAll();
	public Optional<Product> findById(Long id) ;
	public <S extends Product> S save(S entity);
    public List<Product> findByCategory(Category category);
}
```

5.- implementación de repositorio , se tendrá en cuenta una carga de datos estáticos.

```
package com.everis.repository;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

import org.springframework.stereotype.Repository;

import com.everis.entity.Category;
import com.everis.entity.Product;

@Repository
public class ProductRepositoryImpl implements ProductRepository{

	private List<Category> categories = new ArrayList<Category>();
```

```
        private List<Product> products = new ArrayList<Product>();

        private void init() {
                Category c1 = new Category(1L, "shoes");
                Category c2 = new Category(2L, "books");
                Category c3 = new Category(3L, "electronics");

                categories.add(c1);
                categories.add(c2);
                categories.add(c3);

                Product p1 = new Product(1L, "adidas Cloudfoam Ultimate","Walk
in the air in the back", 2.0, 20.0, "CREATED", new Date(),c1);
                Product p2 = new Product(2L,"under armour men","Micro G assert",
4.0, 40.0, "CREATED", new Date(),c1);
                Product p3 = new Product(3L, "Spring Boot in Action","Caig walls
is a software developer", 6.0, 60.0, "CREATED",  new Date(), c2);

                products.add(p1);
                products.add(p2);
                products.add(p3);
        }

        public ProductRepositoryImpl( ) {
                init();
        }

        @Override
        public List<Product> findAll() {
                return products;
        }

        @Override
        public Optional<Product> findById(Long id) {
                return products.stream().filter(p ->   p.getId() ==
id).findAny();
        }

        @Override
        public <S extends Product> S save(S entity) {
                products.add(entity);
                return entity;
        }

        @Override
        public List<Product> findByCategory(Category category) {
                return products.stream().filter(p ->   p.getCategory().getId()
== category.getId()).collect(Collectors.toList());
        }

}
```

6.- creación de la interfaz del servicio:

```
package com.everis.service;

import java.util.List;

import com.everis.entity.Category;
import com.everis.entity.Product;

public interface ProductService {
  public List<Product> listAllProduct();
  public Product getProduct(Long id);
  public Product createProduct(Product product);
  public Product updateProduct(Product product);
  public Product deleteProduct(Long id);
  public List<Product> findByCategory(Category category);
  public Product updateStock(Long id, Double quantity);
}
```

7.- implementación del servicio:

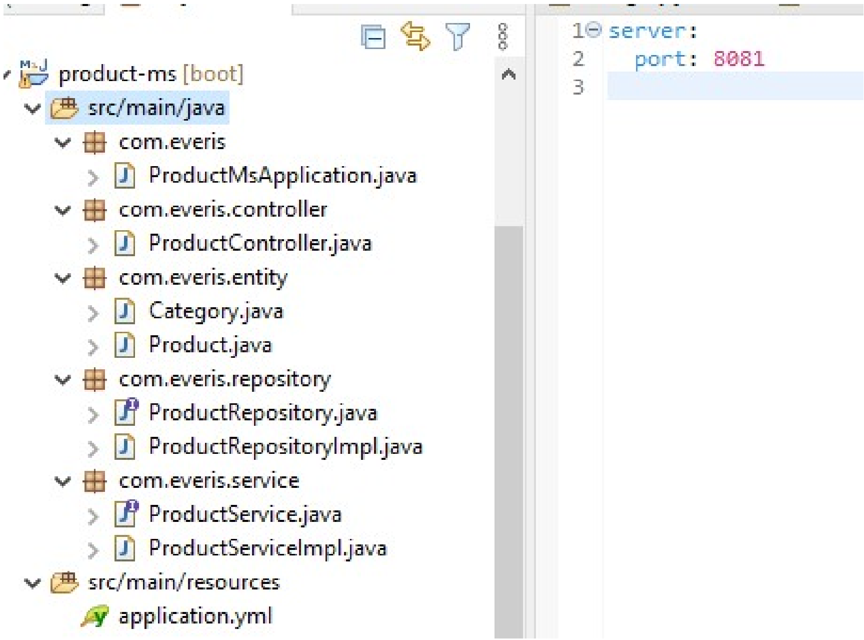
```
package com.everis.service;

import java.util.Date;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.everis.entity.Category;
import com.everis.entity.Product;
import com.everis.repository.ProductRepository;

@Service
public class ProductServiceImpl implements ProductService {

    @Autowired
      private ProductRepository productRepository;
      @Override
      public List<Product> listAllProduct() {
            return productRepository.findAll();
      }

      @Override
      public Product getProduct(Long id) {
            return productRepository.findById(id).orElse(null);
      }

      @Override
      public Product createProduct(Product product) {
            product.setStatus("CREATED");
            product.setCreateAt(new Date());
            return productRepository.save(product);
      }

      @Override
      public Product updateProduct(Product product) {
            Product productDB = getProduct(product.getId());
            if(productDB == null) {
                  return null;
            }
            productDB.setName(product.getName());
            productDB.setDescription(product.getDescription());
            productDB.setCategory(product.getCategory());
            productDB.setPrice(product.getPrice());

            return productRepository.save(productDB);
      }

      @Override
      public Product deleteProduct(Long id) {
            Product productDB = getProduct(id);
            if(productDB == null) {
                  return null;
            }
            productDB.setStatus("DELETED");
            return productRepository.save(productDB);
      }

      @Override
      public List<Product> findByCategory(Category category) {
            return productRepository.findByCategory(category);
      }

      @Override
      public Product updateStock(Long id, Double quantity) {
            Product productDB = getProduct(id);
            if(productDB == null) {
                  return null;
            }
            Double stock = productDB.getStock() + quantity;
            productDB.setStock(stock);
            return productRepository.save(productDB);
      }

}
```

7.- implementación del controlador:

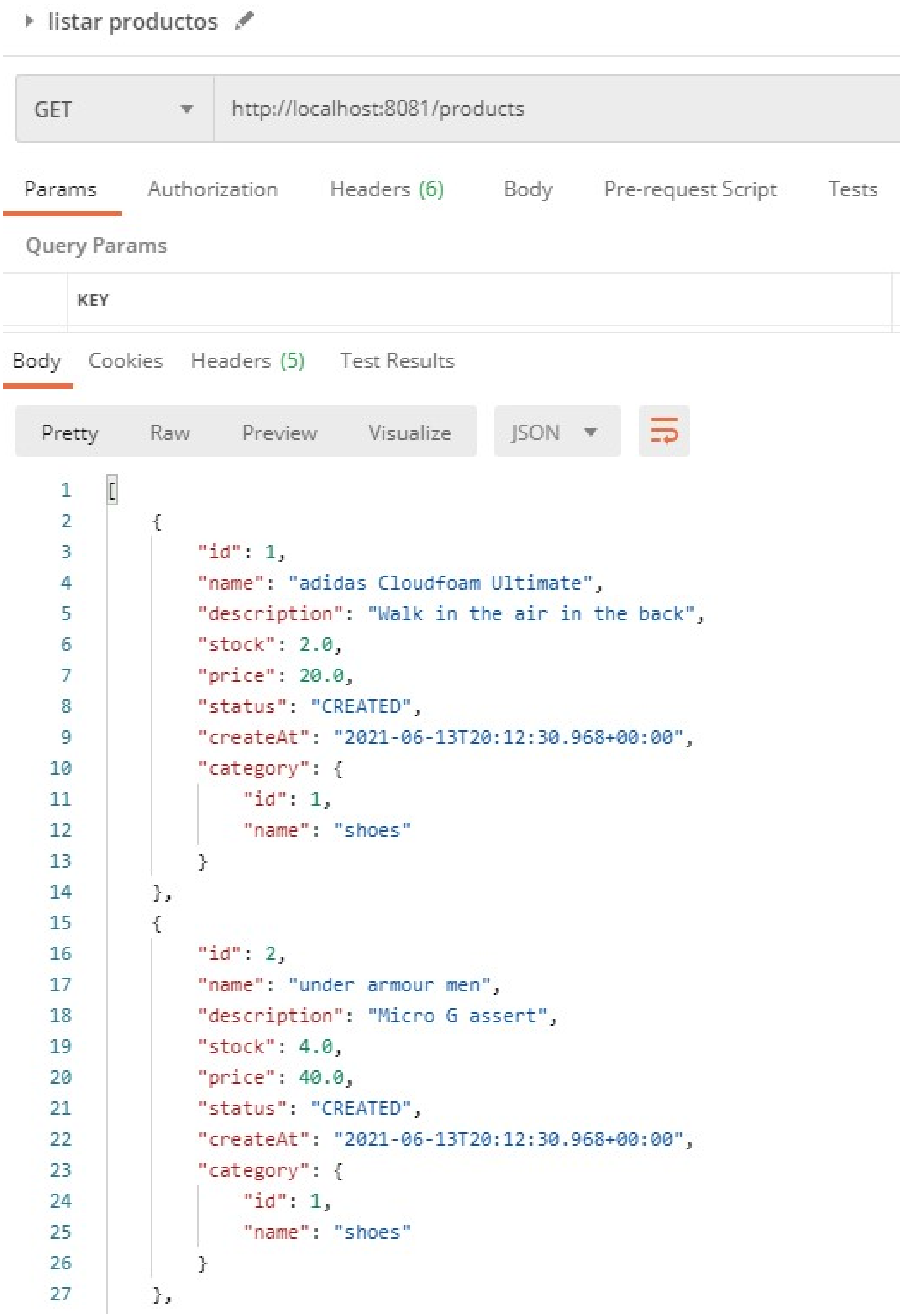
```
package com.everis.controller;


import java.util.List;


import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;
import com.everis.entity.Category;

import com.everis.entity.Product;

import com.everis.service.ProductService;


@RestController

@RequestMapping(value="/products")

public class ProductController {


    @Autowired

    private ProductService productService;


    @GetMapping

    public ResponseEntity<List<Product>> listProduct(@RequestParam(name="categoryId", required = false) Long categoryId) {

        List<Product> products = null;

        if(categoryId == null) {

            products = productService.listAllProduct();

            if(products.isEmpty()) {

                    return ResponseEntity.notFound().build();

            }

        }else {

            products = productService.findByCategory(Category.builder().id(categoryId).build());

            if(products.isEmpty()) {

                    return ResponseEntity.notFound().build();

            }

        }


        return ResponseEntity.ok(products);

    }


    @GetMapping(value = "/{id}")

    public ResponseEntity<Product> getProduct(@PathVariable("id") Long id) {
        Product product = productService.getProduct(id);
        if(product == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(product);
    }

    @PostMapping
    public ResponseEntity<Product> createProduct(@RequestBody Product product) {
        Product productCreate = productService.createProduct(product);
        return ResponseEntity.status(HttpStatus.CREATED).body(productCreate);

    }

    @PutMapping(value = "/{id}")
    public ResponseEntity<Product> updateProduct(@PathVariable("id")Long id, @RequestBody Product product) {
        product.setId(id);
        Product productDB = productService.updateProduct(product);
        if(productDB == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(productDB);
    }

    @DeleteMapping(value = "/{id}")
    public ResponseEntity<Product> deleteProduct(@PathVariable ("id") Long id) {
        Product productDelete = productService.deleteProduct(id);
        if(productDelete == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(productDelete);
    }
```

```
	@GetMapping(value = "/{id}/stock")
	public ResponseEntity<Product> updateStockProduct(@PathVariable("id") Long id, @RequestParam(name="quantity", required = true) Double quantity) {
		Product product = productService.updateStock(id, quantity);
		if(product == null) {
			return ResponseEntity.notFound().build();
		}
		return ResponseEntity.ok(product);
	}


}
```

Configuración de properties :



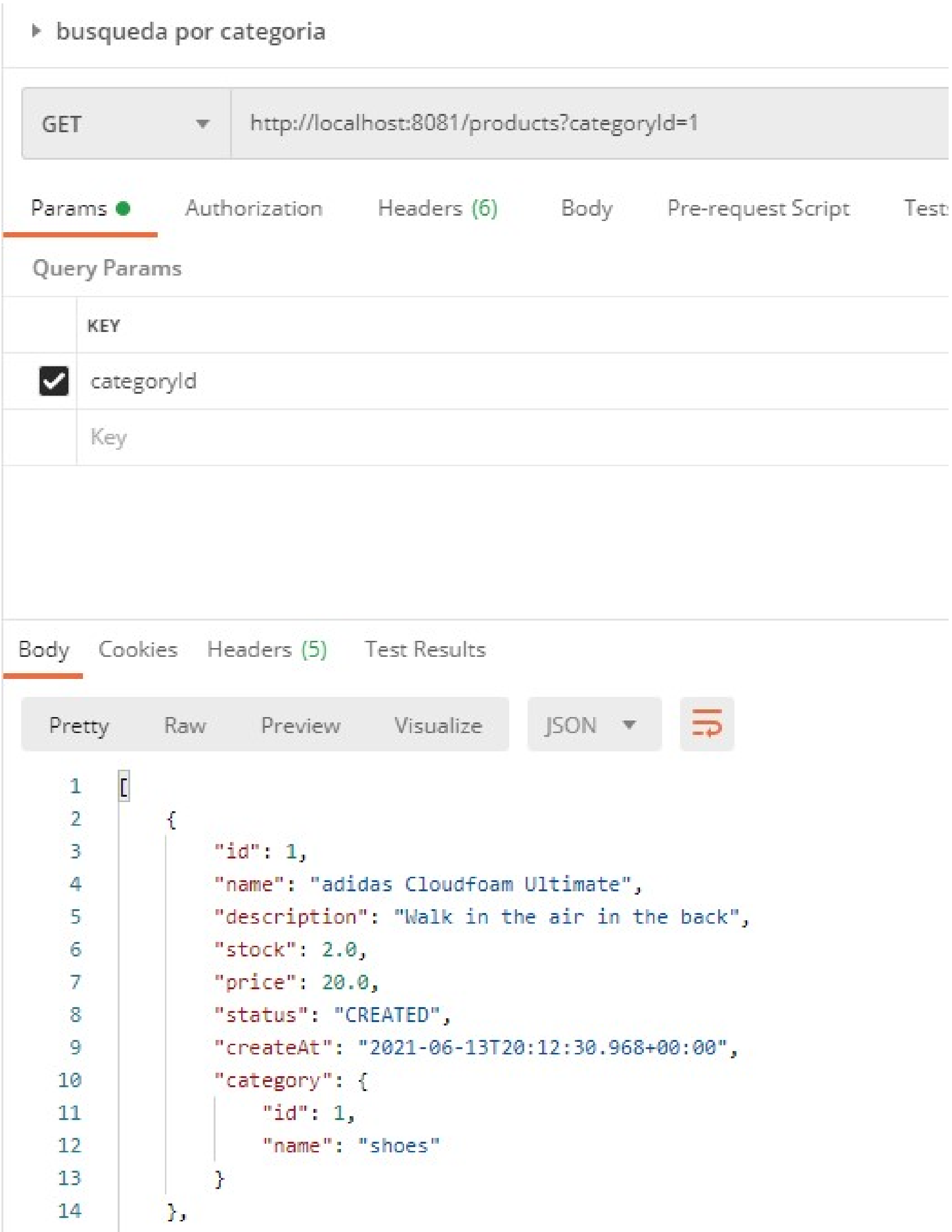
```yaml
server:
  port: 8081
```



listar productos

GET http://localhost:8081/products

```
[
    {
        "id": 1,
        "name": "adidas Cloudfoam Ultimate",
        "description": "Walk in the air in the back",
        "stock": 2.0,
        "price": 20.0,
        "status": "CREATED",
        "createAt": "2021-06-13T20:12:30.968+00:00",
        "category": {
            "id": 1,
            "name": "shoes"
        }
    },
    {
        "id": 2,
        "name": "under armour men",
        "description": "Micro G assert",
        "stock": 4.0,
        "price": 40.0,
        "status": "CREATED",
        "createAt": "2021-06-13T20:12:30.968+00:00",
        "category": {
            "id": 1,
            "name": "shoes"
        }
    },
```

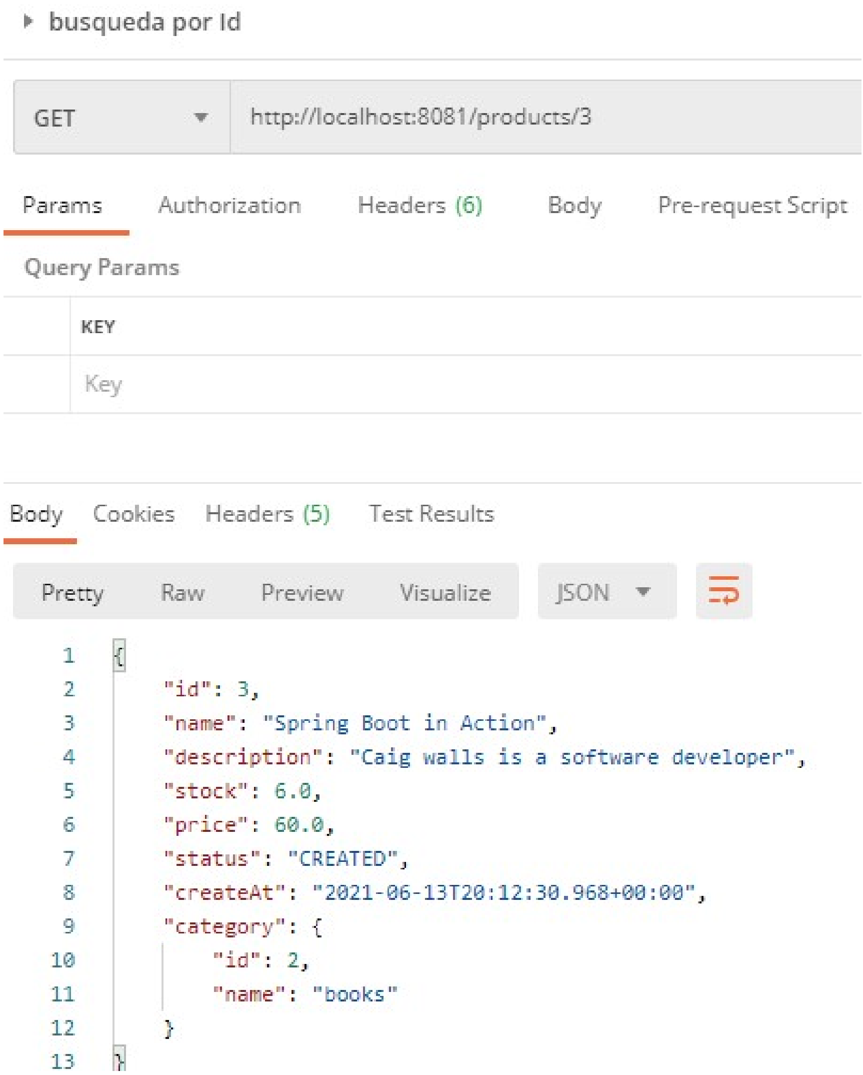


GET http://localhost:8081/products?categoryId=1

Query Params

	KEY
☑	categoryId
	Key

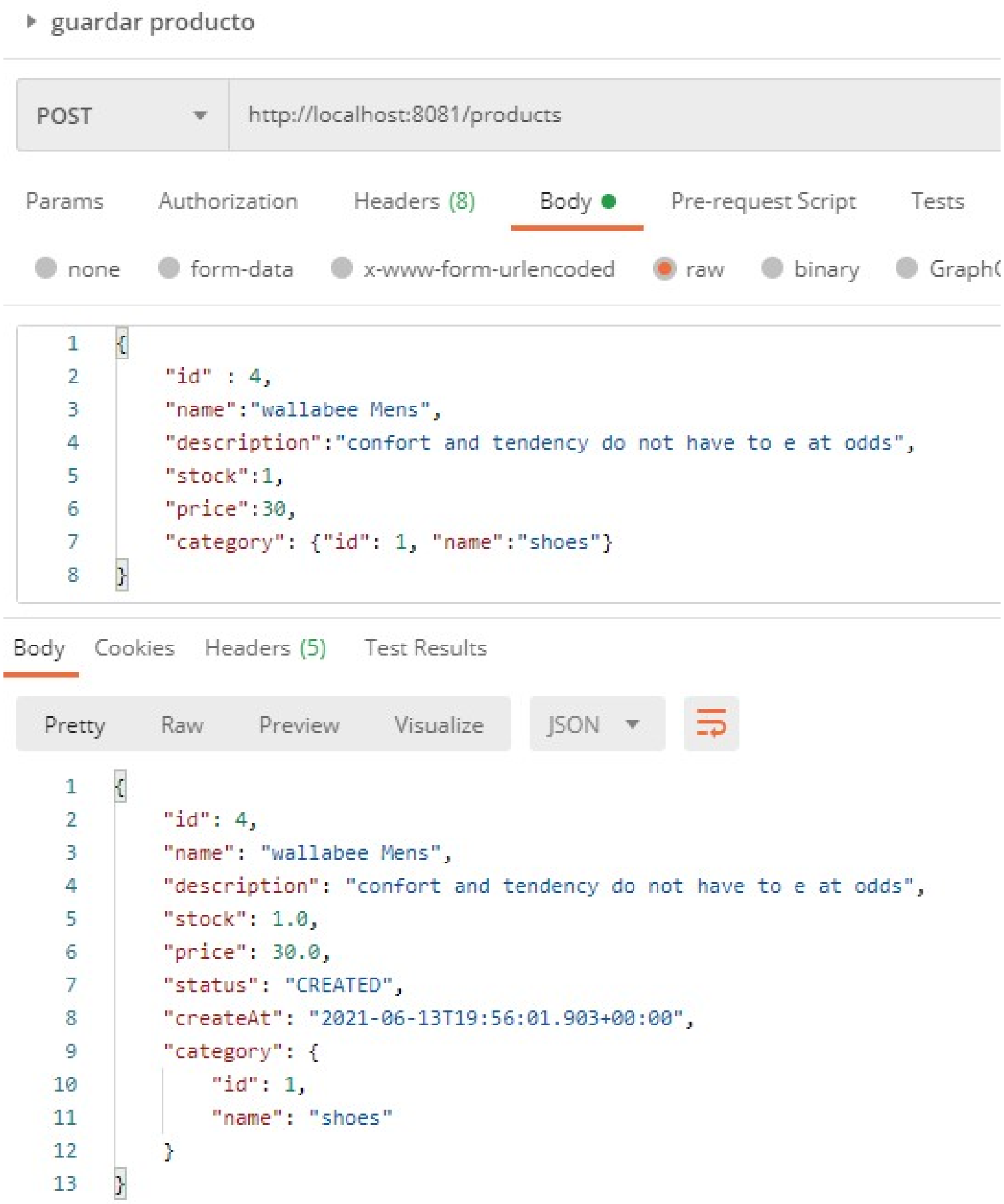
```
[
    {
        "id": 1,
        "name": "adidas Cloudfoam Ultimate",
        "description": "Walk in the air in the back",
        "stock": 2.0,
        "price": 20.0,
        "status": "CREATED",
        "createAt": "2021-06-13T20:12:30.968+00:00",
        "category": {
            "id": 1,
            "name": "shoes"
        }
    },
```



busqueda por Id

GET http://localhost:8081/products/3

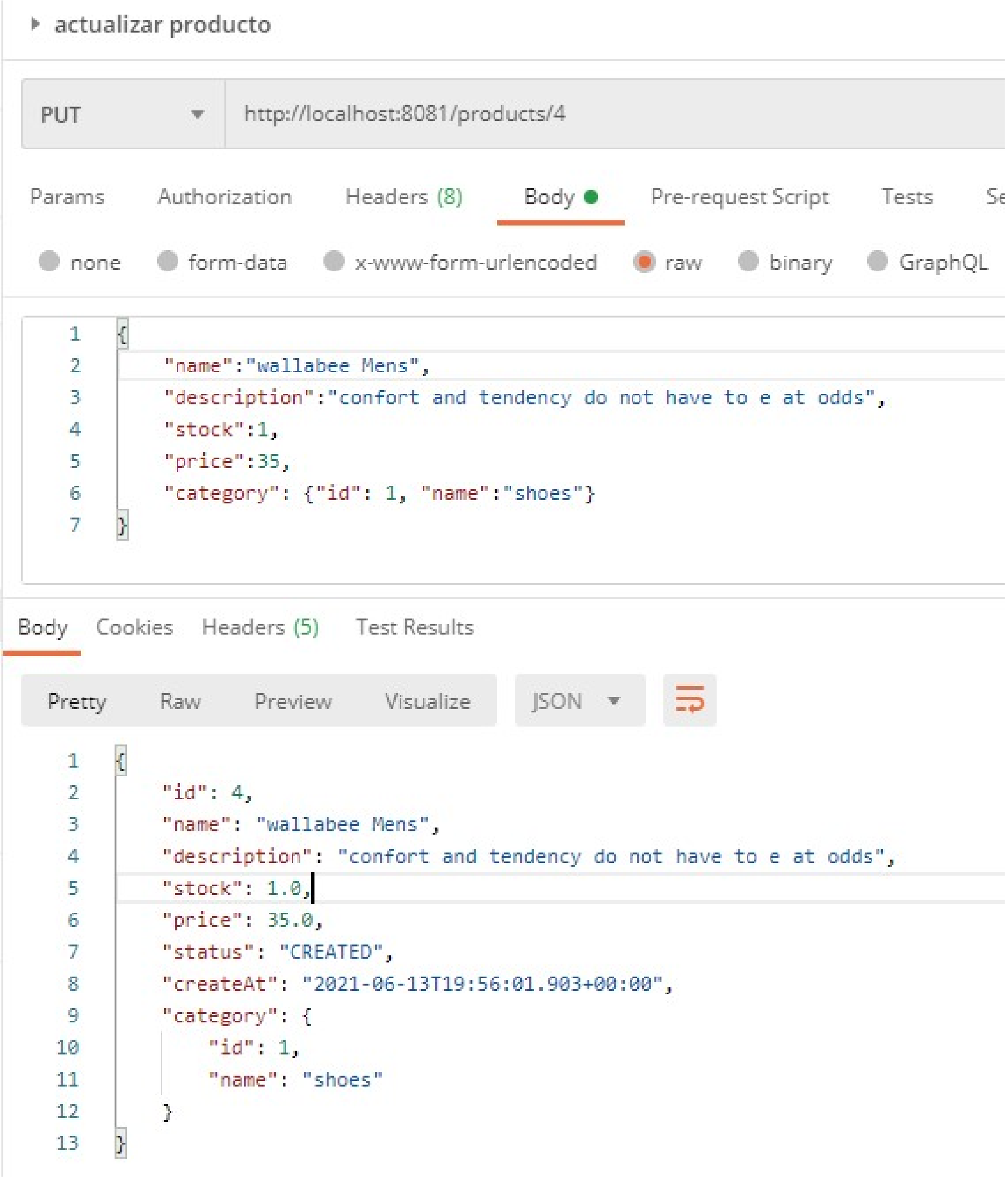
```
{
    "id": 3,
    "name": "Spring Boot in Action",
    "description": "Caig walls is a software developer",
    "stock": 6.0,
    "price": 60.0,
    "status": "CREATED",
    "createAt": "2021-06-13T20:12:30.968+00:00",
    "category": {
        "id": 2,
        "name": "books"
    }
}
```



POST http://localhost:8081/products

```
{
    "id" : 4,
    "name":"wallabee Mens",
    "description":"confort and tendency do not have to e at odds",
    "stock":1,
    "price":30,
    "category": {"id": 1, "name":"shoes"}
}
```

```
{
    "id": 4,
    "name": "wallabee Mens",
    "description": "confort and tendency do not have to e at odds",
    "stock": 1.0,
    "price": 30.0,
    "status": "CREATED",
    "createAt": "2021-06-13T19:56:01.903+00:00",
    "category": {
        "id": 1,
        "name": "shoes"
    }
}
```

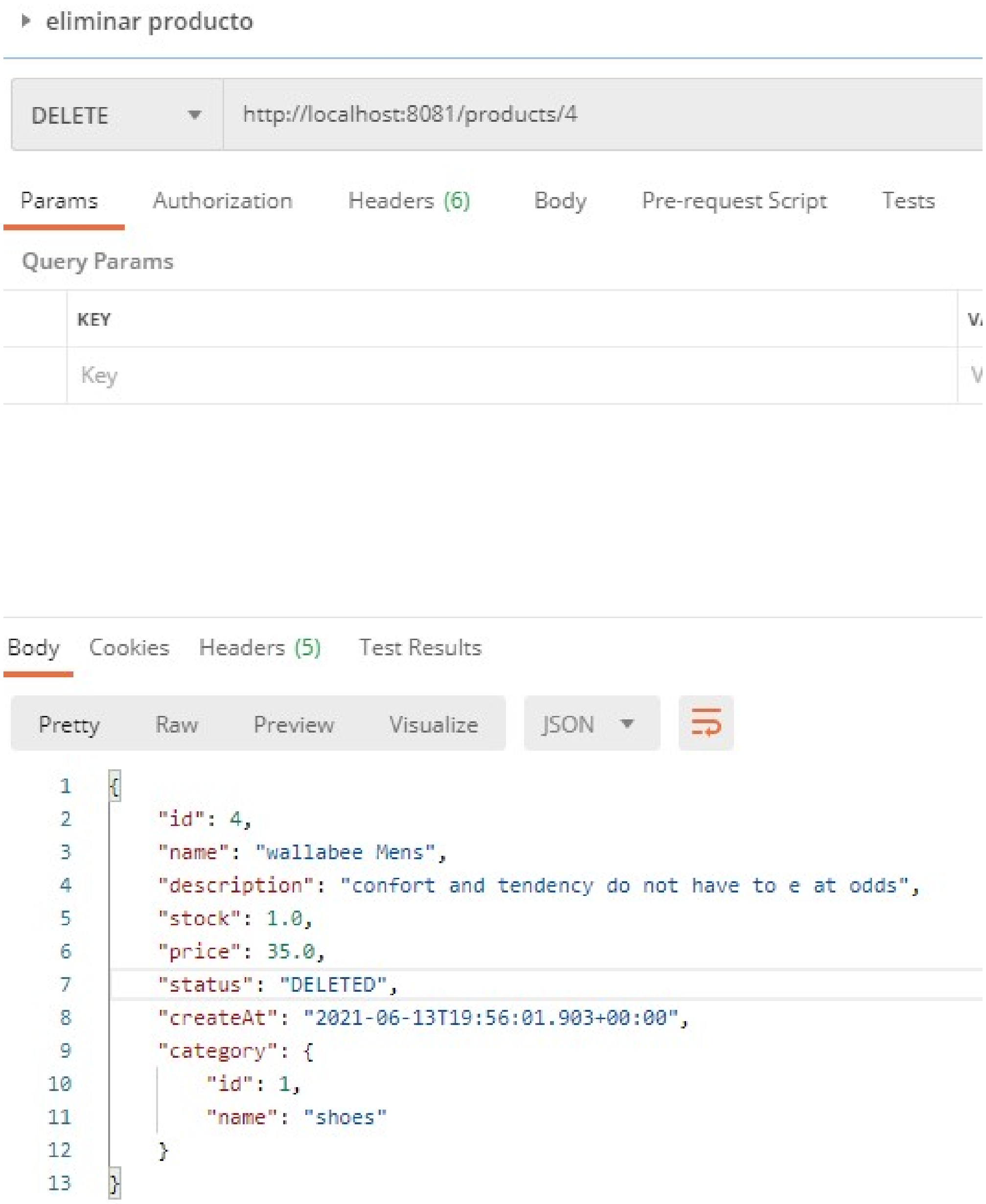


actualizar producto

PUT http://localhost:8081/products/4

```
{
    "name":"wallabee Mens",
    "description":"confort and tendency do not have to e at odds",
    "stock":1,
    "price":35,
    "category": {"id": 1, "name":"shoes"}
}
```

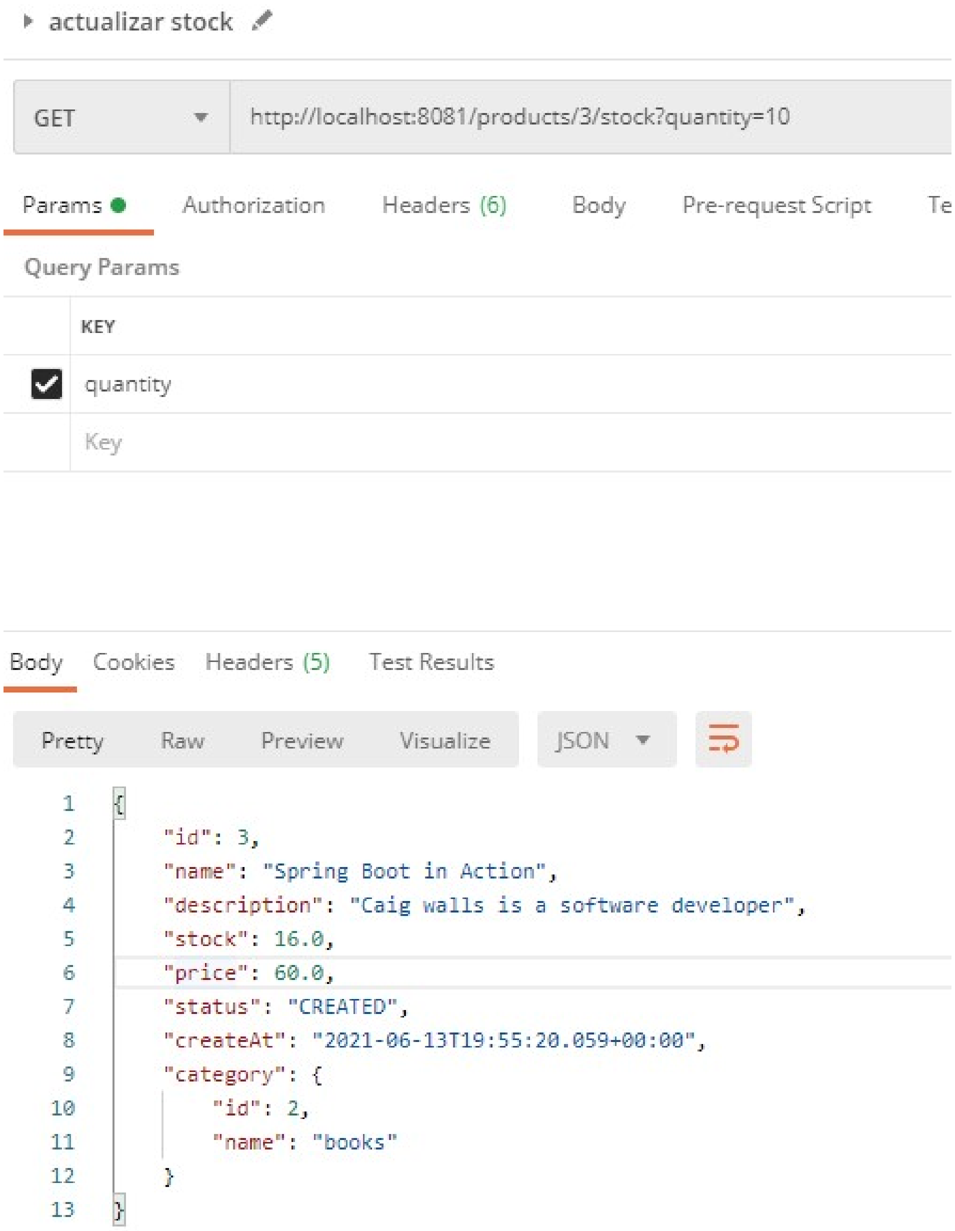
```
{
    "id": 4,
    "name": "wallabee Mens",
    "description": "confort and tendency do not have to e at odds",
    "stock": 1.0,
    "price": 35.0,
    "status": "CREATED",
    "createAt": "2021-06-13T19:56:01.903+00:00",
    "category": {
        "id": 1,
        "name": "shoes"
    }
}
```



eliminar producto

DELETE http://localhost:8081/products/4

```
{
    "id": 4,
    "name": "wallabee Mens",
    "description": "confort and tendency do not have to e at odds",
    "stock": 1.0,
    "price": 35.0,
    "status": "DELETED",
    "createAt": "2021-06-13T19:56:01.903+00:00",
    "category": {
        "id": 1,
        "name": "shoes"
    }
}
```



actualizar stock

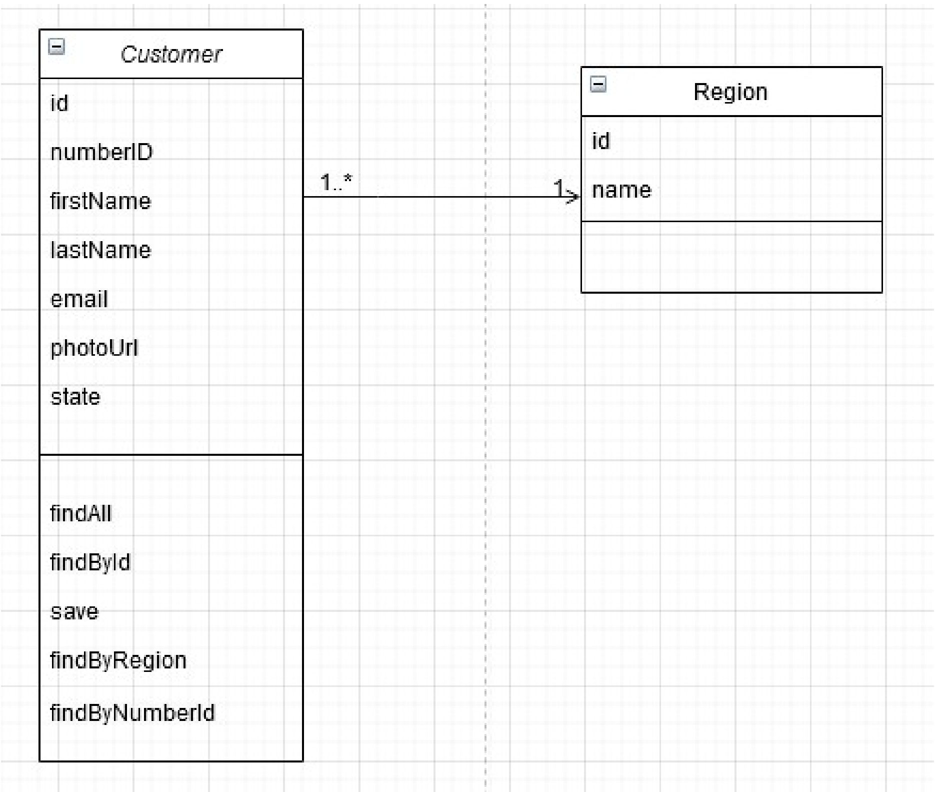
GET http://localhost:8081/products/3/stock?quantity=10

Query Params

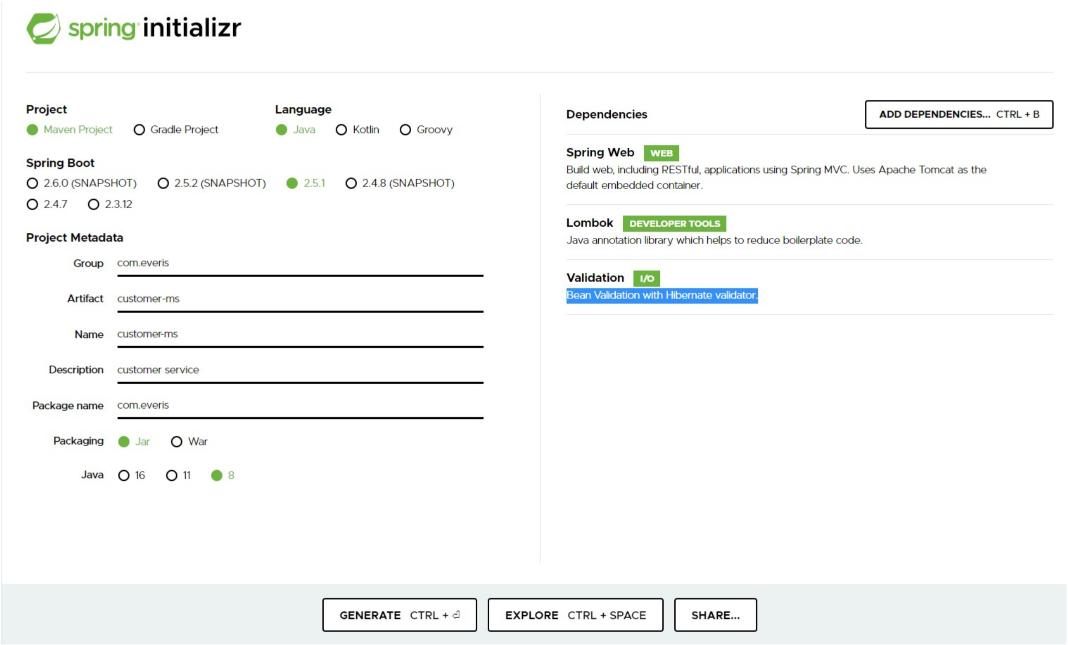
	KEY
☑	quantity
	Key

```
{
    "id": 3,
    "name": "Spring Boot in Action",
    "description": "Caig walls is a software developer",
    "stock": 16.0,
    "price": 60.0,
    "status": "CREATED",
    "createAt": "2021-06-13T19:55:20.059+00:00",
    "category": {
        "id": 2,
        "name": "books"
    }
}
```

Servicios rest : customer-ms



1.- Creación de proyecto del microservicio



Project
● Maven Project ○ Gradle Project

Language
● Java ○ Kotlin ○ Groovy

Spring Boot
○ 2.6.0 (SNAPSHOT) ○ 2.5.2 (SNAPSHOT) ● 2.5.1 ○ 2.4.8 (SNAPSHOT)
○ 2.4.7 ○ 2.3.12

Project Metadata

Group com.everis

Artifact customer-ms

Name customer-ms

Description customer service

Package name com.everis

Packaging ● Jar ○ War

Java ○ 16 ○ 11 ● 8

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

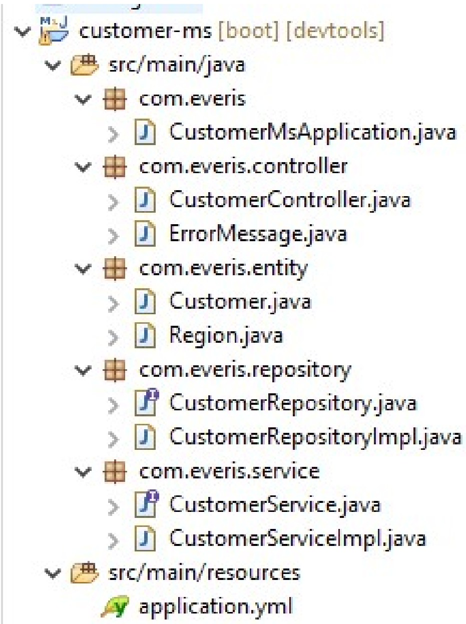
Validation I/O
Bean Validation with Hibernate validator.

Agregándose al pom las siguientes dependencias:

```
<dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
</dependency>

<dependency>
          <groupId>javax.validation</groupId>
```

```
			<artifactId>validation-api</artifactId>
		</dependency>
```

2.- Creación de paquetes: controller, entity, repository y service



```
customer-ms [boot] [devtools]
  src/main/java
    com.everis
      CustomerMsApplication.java
    com.everis.controller
      CustomerController.java
      ErrorMessage.java
    com.everis.entity
      Customer.java
      Region.java
    com.everis.repository
      CustomerRepository.java
      CustomerRepositoryImpl.java
    com.everis.service
      CustomerService.java
      CustomerServiceImpl.java
  src/main/resources
    application.yml
```

3.- implementación de clase entity: customer y región

```
package com.everis.entity;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import java.io.Serializable;

@Data
@AllArgsConstructor @NoArgsConstructor @Builder
public class Customer implements Serializable {


	private static final long serialVersionUID = 1L;

	private Long id;

    @NotEmpty(message = "El número de documento no puede ser vacío")
    @Size( min = 8 , max = 8, message = "El tamaño del número de documento es 
8")
    private String numberID;

    @NotEmpty(message = "El nombre no puede ser vacío")
    private String firstName;
    @NotEmpty(message = "El apellido no puede ser vacío")
    private String lastName;

    @NotEmpty(message = "el correo no puede estar vacío")
    @Email(message = "no es un dirección de correo bien formada")
    private String email;

    private String photoUrl;

    @NotNull(message = "la región no puede ser vacia")
    private Region region;

    private String state;
}
```

```
package com.everis.entity;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.io.Serializable;

@Data
@AllArgsConstructor @NoArgsConstructor @Builder
public class Region implements Serializable {

        private static final long serialVersionUID = 1L;

        private Long id;

        private String name;

}
```

4.- creación de repositorio:

```
package com.everis.repository;

import com.everis.entity.Customer;
import com.everis.entity.Region;

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

public interface CustomerRepository  {
        public Customer findByNumberID(String numberID);
        public List<Customer> findByRegion(Region region);

        public List<Customer> findAll();
        public <S extends Customer> S save(S entity);
        public Optional<Customer> findById(Long id);
}
```

5.- implementación de repositorio , se tendrá en cuenta una carga de datos estáticos.

```
package com.everis.repository;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

import org.springframework.stereotype.Repository;

import com.everis.entity.Customer;
import com.everis.entity.Region;

@Repository
public class CustomerRepositoryImpl implements CustomerRepository {

        private List<Region> regiones = new ArrayList<Region>();
        private List<Customer> customers = new ArrayList<Customer>();

        private void init() {
                Region r1 = new Region(1L, "Sudamérica");
                Region r2 = new Region(2L, "Centroamérica");
                Region r3 = new Region(3L, "Norteamérica");
                Region r4 = new Region(4L, "Europa");
                Region r5 = new Region(5L, "Asia");
                Region r6 = new Region(6L, "Africa");
                Region r7 = new Region(7L, "Oceanía");
                Region r8 = new Region(8L, "Antártida");


                regiones.add(r1);
                regiones.add(r2);
                regiones.add(r3);
                regiones.add(r4);
                regiones.add(r5);
                regiones.add(r6);
                regiones.add(r7);
                regiones.add(r8);

                Customer c1 = new Customer(1L, "32404580","Andrés",
"Guzmán","profesor@bolsadeideas.com", "",r1, "CREATED");

                customers.add(c1);
        }

        public CustomerRepositoryImpl( ) {
                init();
        }

        @Override
        public List<Customer> findAll() {
                return customers;
        }

        @Override
        public Optional<Customer> findById(Long id) {
                return customers.stream().filter(p ->   p.getId() ==
id).findFirst();
        }

        @Override
        public <S extends Customer> S save(S entity) {
                customers.add(entity);
                return entity;
        }

        @Override
        public List<Customer> findByRegion(Region region) {
```

```
		return customers.stream().filter(p ->   p.getRegion().getId() == 
region.getId()).collect(Collectors.toList());
	}

	@Override
	public Customer findByNumberID(String numberID) {
		return customers.stream().filter(p -> 
p.getNumberID().equals(numberID)).findAny().orElse(null);
	}

}
```

6.- creación de la interfaz del servicio:

```
package com.everis.service;

import java.util.List;

import com.everis.entity.Customer;
import com.everis.entity.Region;

public interface CustomerService {

    public List<Customer> findCustomerAll();
    public List<Customer> findCustomersByRegion(Region region);

    public Customer createCustomer(Customer customer);
    public Customer updateCustomer(Customer customer);
    public Customer deleteCustomer(Customer customer);
    public Customer getCustomer(Long id);

}
```

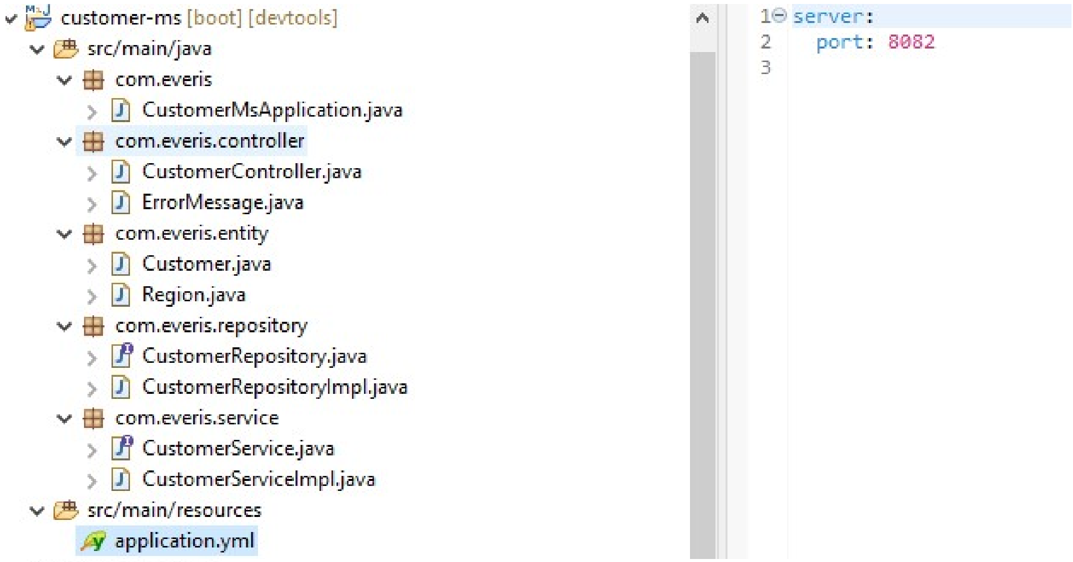
7.- implementación del servicio:

```
package com.everis.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.everis.entity.Customer;
import com.everis.entity.Region;
import com.everis.repository.CustomerRepository;

import java.util.List;
@Service
public class CustomerServiceImpl  implements CustomerService {

    @Autowired
    CustomerRepository customerRepository;

    @Override
    public List<Customer> findCustomerAll() {
        return customerRepository.findAll();
    }

    @Override
    public List<Customer> findCustomersByRegion(Region region) {
        return customerRepository.findByRegion(region);
    }

    @Override
    public Customer createCustomer(Customer customer) {
        Customer customerDB = customerRepository.findByNumberID (
customer.getNumberID () );
        if (customerDB != null){
            return  customerDB;
        }

        customer.setState("CREATED");
        customerDB = customerRepository.save ( customer );
        return customerDB;
    }

    @Override
    public Customer updateCustomer(Customer customer) {
        Customer customerDB = getCustomer(customer.getId());
        if (customerDB == null){
            return  null;
        }
        customerDB.setFirstName(customer.getFirstName());
        customerDB.setLastName(customer.getLastName());
        customerDB.setEmail(customer.getEmail());
        customerDB.setPhotoUrl(customer.getPhotoUrl());

        return  customerRepository.save(customerDB);
    }

    @Override
    public Customer deleteCustomer(Customer customer) {
        Customer customerDB = getCustomer(customer.getId());
        if (customerDB ==null){
            return  null;
        }
        customer.setState("DELETED");
        return customerRepository.save(customer);
    }

    @Override
    public Customer getCustomer(Long id) {
        return  customerRepository.findById(id).orElse(null);
    }
}
```

7.- implementación del controlador:

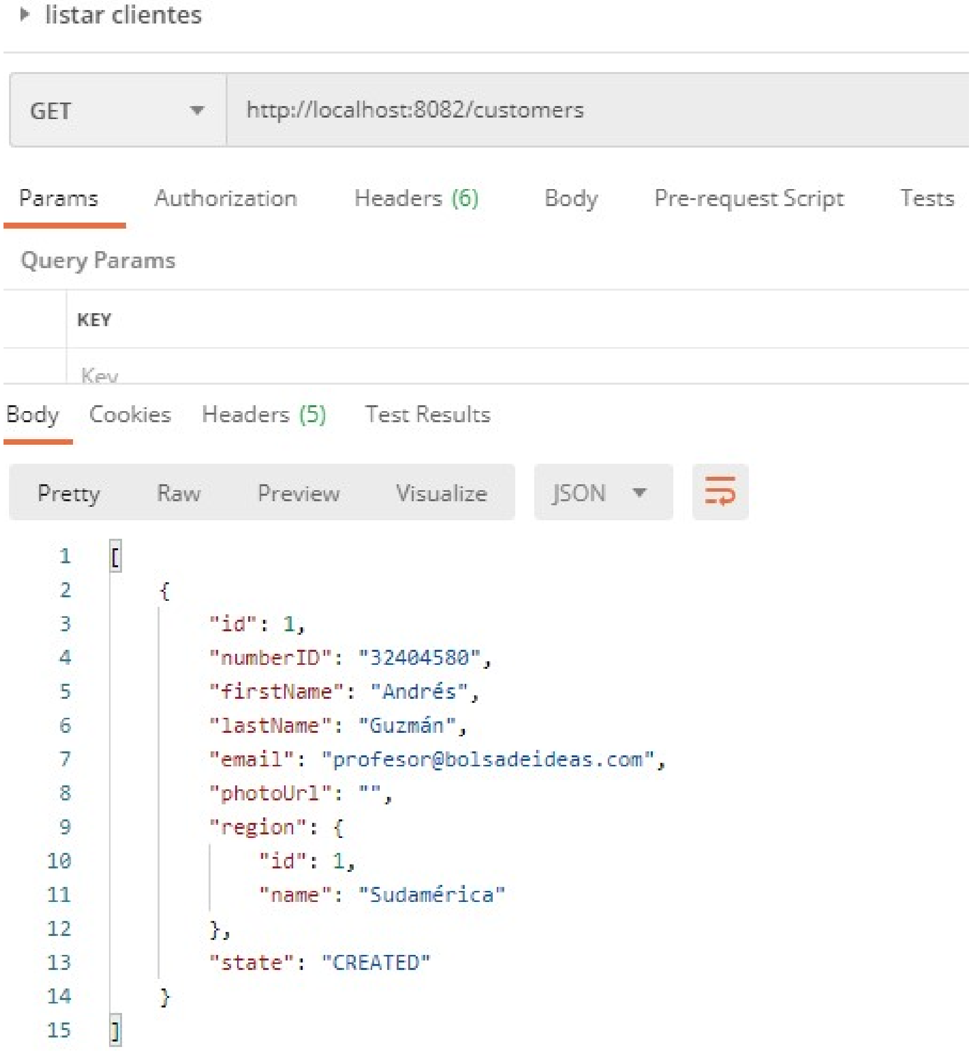
```
package com.everis.controller;

import com.everis.entity.Customer;
import com.everis.entity.Region;
import com.everis.service.CustomerService;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.server.ResponseStatusException;

import javax.validation.Valid;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

@Slf4j
@RestController
@RequestMapping("/customers")
public class CustomerController {

    @Autowired
    CustomerService customerService;

    @GetMapping
    public ResponseEntity<List<Customer>> listAllCustomers(@RequestParam(name
= "regionId" , required = false) Long regionId ) {
        List<Customer> customers =  new ArrayList<>();
        if (null ==  regionId) {
            customers = customerService.findCustomerAll();
            if (customers.isEmpty()) {
                return ResponseEntity.noContent().build();
            }
        }else{
            Region Region= new Region();
            Region.setId(regionId);
            customers = customerService.findCustomersByRegion(Region);
            if ( null == customers ) {
                log.error("Customers with Region id {} not found.",
regionId);
                return  ResponseEntity.notFound().build();
            }
        }

        return  ResponseEntity.ok(customers);
    }

    @GetMapping(value = "/{id}")
    public ResponseEntity<Customer> getCustomer(@PathVariable("id") long id)
{
        log.info("Fetching Customer with id {}", id);
        Customer customer = customerService.getCustomer(id);
        if (  null == customer) {
            log.error("Customer with id {} not found.", id);
            return  ResponseEntity.notFound().build();
        }
        return  ResponseEntity.ok(customer);
    }

    @PostMapping
    public ResponseEntity<Customer> createCustomer(@Valid @RequestBody
Customer customer, BindingResult result) {
        log.info("Creating Customer : {}", customer);
        if (result.hasErrors()){
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST,
this.formatMessage(result));
        }

       Customer customerDB = customerService.createCustomer (customer);

        return  ResponseEntity.status( HttpStatus.CREATED).body(customerDB);
    }
    @PutMapping(value = "/{id}")
    public ResponseEntity<?> updateCustomer(@PathVariable("id") long id,
@RequestBody Customer customer) {
        Log.info("Updating Customer with id {}", id);

        Customer currentCustomer = customerService.getCustomer(id);

        if ( null == currentCustomer ) {
            Log.error("Unable to update. Customer with id {} not found.",
id);
            return  ResponseEntity.notFound().build();
        }
        customer.setId(id);
        currentCustomer=customerService.updateCustomer(customer);
        return  ResponseEntity.ok(currentCustomer);
    }

    @DeleteMapping(value = "/{id}")
    public ResponseEntity<Customer> deleteCustomer(@PathVariable("id") long
id) {
        Log.info("Fetching & Deleting Customer with id {}", id);

        Customer customer = customerService.getCustomer(id);
        if ( null == customer ) {
            Log.error("Unable to delete. Customer with id {} not found.",
id);
            return  ResponseEntity.notFound().build();
        }
        customer = customerService.deleteCustomer(customer);
        return  ResponseEntity.ok(customer);
    }

    private String formatMessage( BindingResult result){
        List<Map<String,String>> errors = result.getFieldErrors().stream()
                .map(err ->{
                    Map<String,String>  error =  new HashMap<>();
                    error.put(err.getField(), err.getDefaultMessage());
                    return error;

                }).collect(Collectors.toList());
        ErrorMessage errorMessage = ErrorMessage.builder()
                .code("01")
                .messages(errors).build();
        ObjectMapper mapper = new ObjectMapper();
        String jsonString="";
        try {
            jsonString = mapper.writeValueAsString(errorMessage);
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
        return jsonString;
    }

}
```

Configuración de properties :



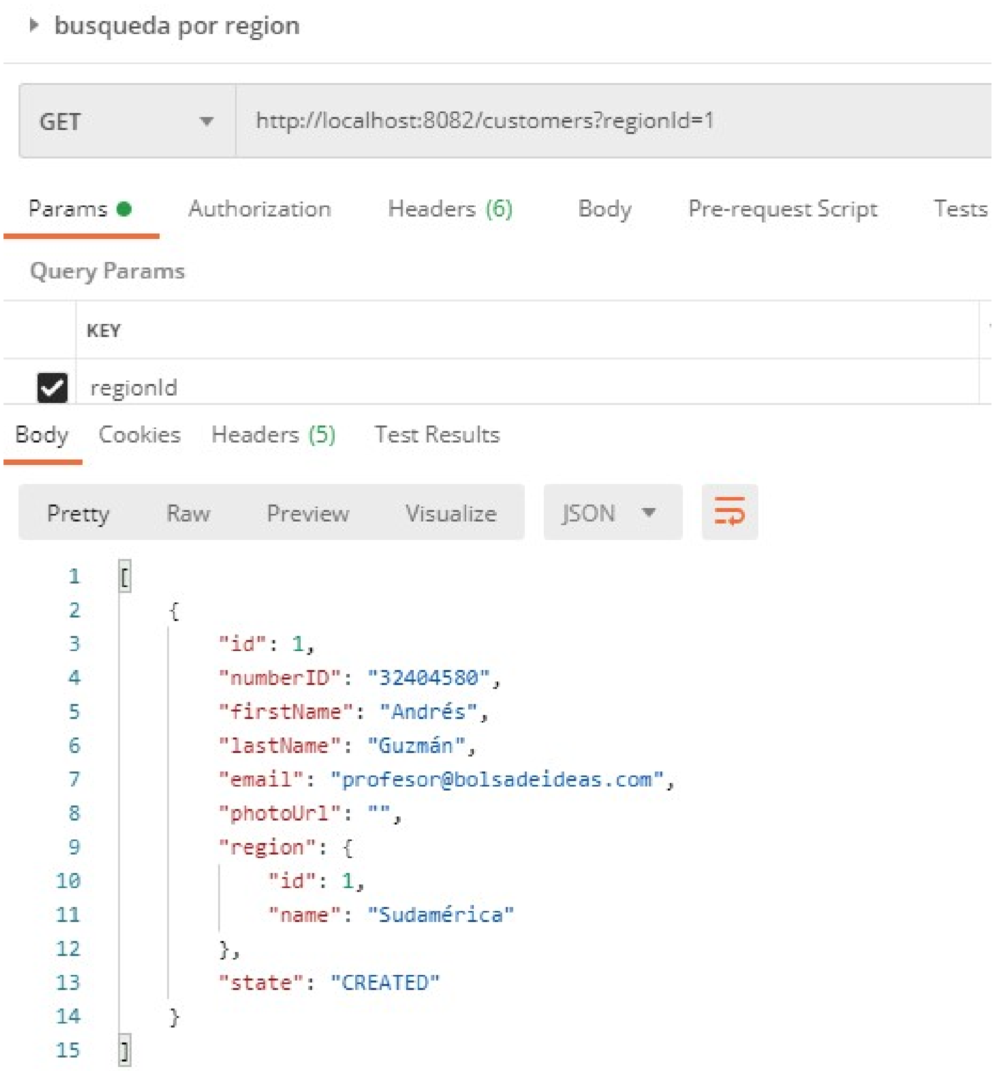
```
customer-ms [boot] [devtools]
  src/main/java
    com.everis
      CustomerMsApplication.java
    com.everis.controller
      CustomerController.java
      ErrorMessage.java
    com.everis.entity
      Customer.java
      Region.java
    com.everis.repository
      CustomerRepository.java
      CustomerRepositoryImpl.java
    com.everis.service
      CustomerService.java
      CustomerServiceImpl.java
  src/main/resources
    application.yml
```

```yaml
server:
  port: 8082
```



GET http://localhost:8082/customers

```json
[
    {
        "id": 1,
        "numberID": "32404580",
        "firstName": "Andrés",
        "lastName": "Guzmán",
        "email": "profesor@bolsadeideas.com",
        "photoUrl": "",
        "region": {
            "id": 1,
            "name": "Sudamérica"
        },
        "state": "CREATED"
    }
]
```



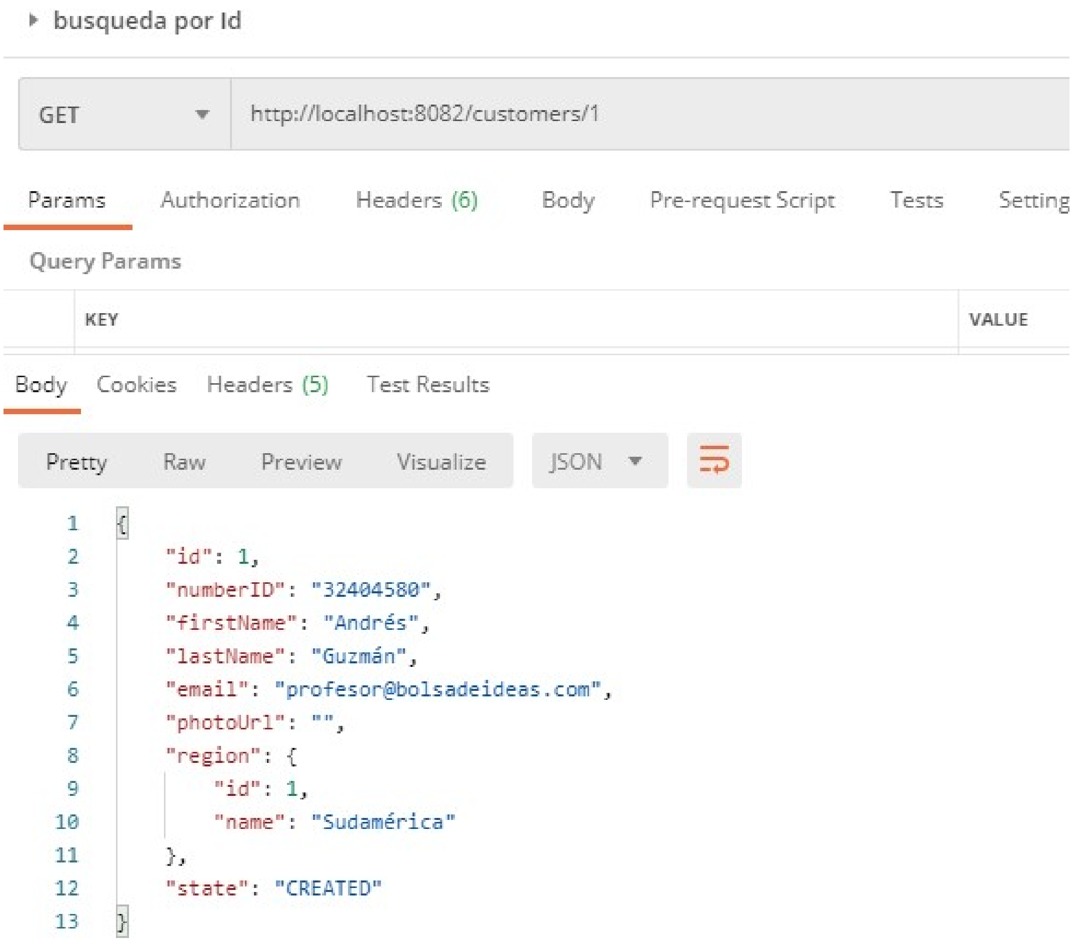
busqueda por region

GET http://localhost:8082/customers?regionId=1

	KEY
☑	regionId

JSON

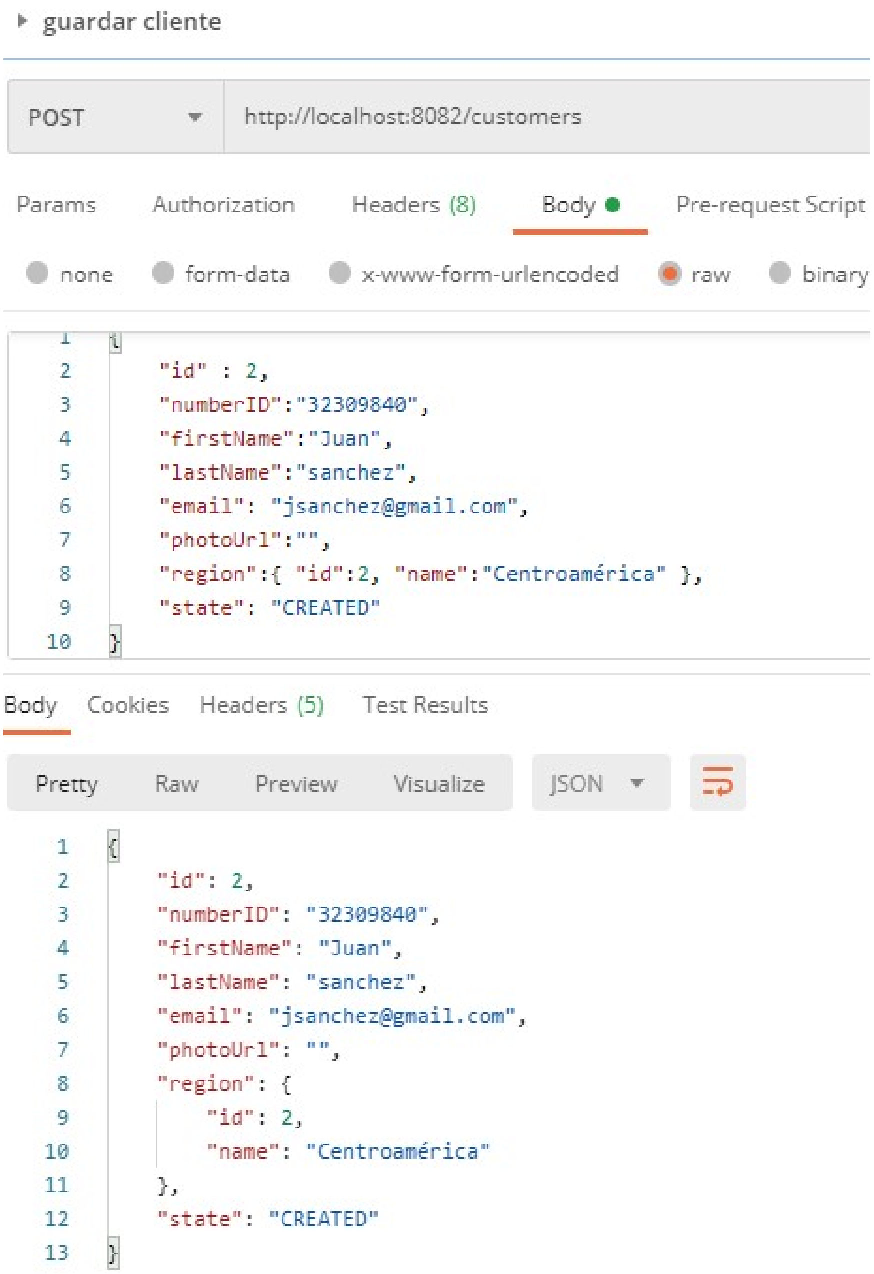
```
[
    {
        "id": 1,
        "numberID": "32404580",
        "firstName": "Andrés",
        "lastName": "Guzmán",
        "email": "profesor@bolsadeideas.com",
        "photoUrl": "",
        "region": {
            "id": 1,
            "name": "Sudamérica"
        },
        "state": "CREATED"
    }
]
```



busqueda por Id

GET http://localhost:8082/customers/1

```
{
    "id": 1,
    "numberID": "32404580",
    "firstName": "Andrés",
    "lastName": "Guzmán",
    "email": "profesor@bolsadeideas.com",
    "photoUrl": "",
    "region": {
        "id": 1,
        "name": "Sudamérica"
    },
    "state": "CREATED"
}
```

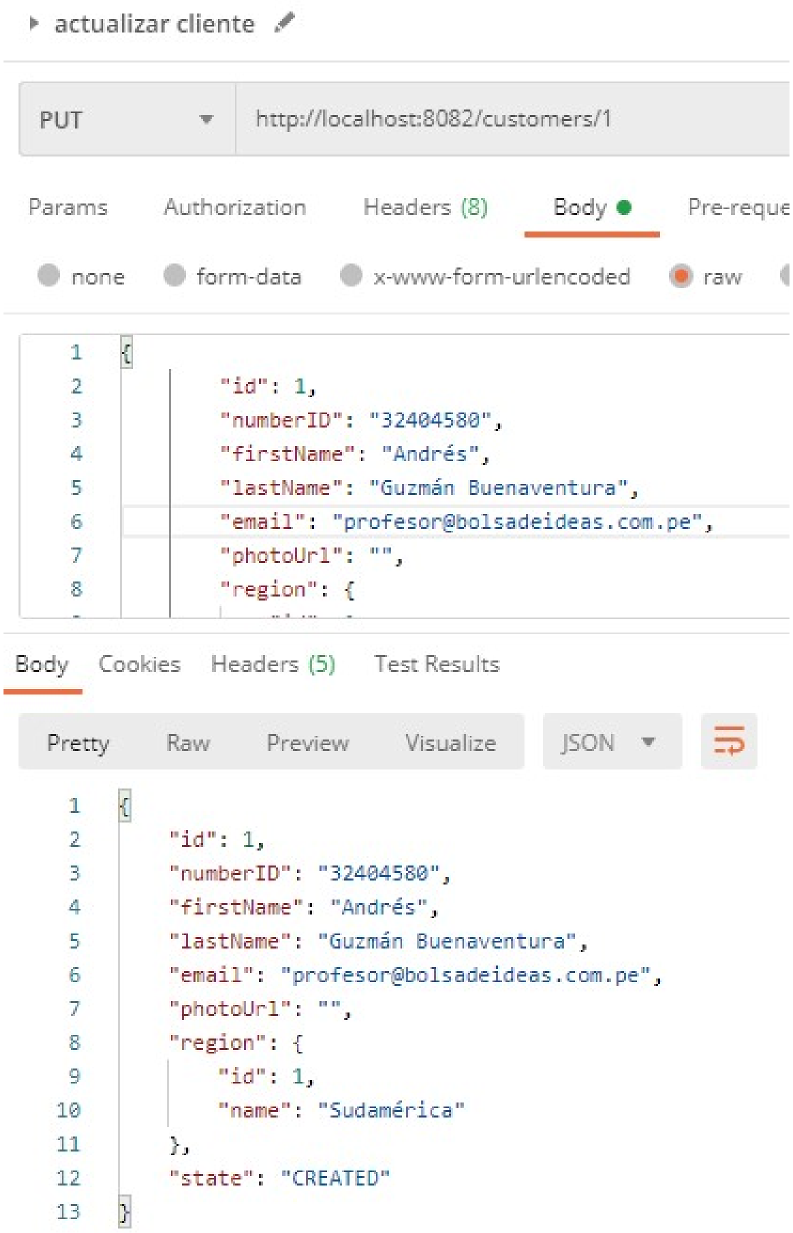


POST http://localhost:8082/customers

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary

```
{
    "id" : 2,
    "numberID":"32309840",
    "firstName":"Juan",
    "lastName":"sanchez",
    "email": "jsanchez@gmail.com",
    "photoUrl":"",
    "region":{ "id":2, "name":"Centroamérica" },
    "state": "CREATED"
}
```

```
{
    "id": 2,
    "numberID": "32309840",
    "firstName": "Juan",
    "lastName": "sanchez",
    "email": "jsanchez@gmail.com",
    "photoUrl": "",
    "region": {
        "id": 2,
        "name": "Centroamérica"
    },
    "state": "CREATED"
}
```



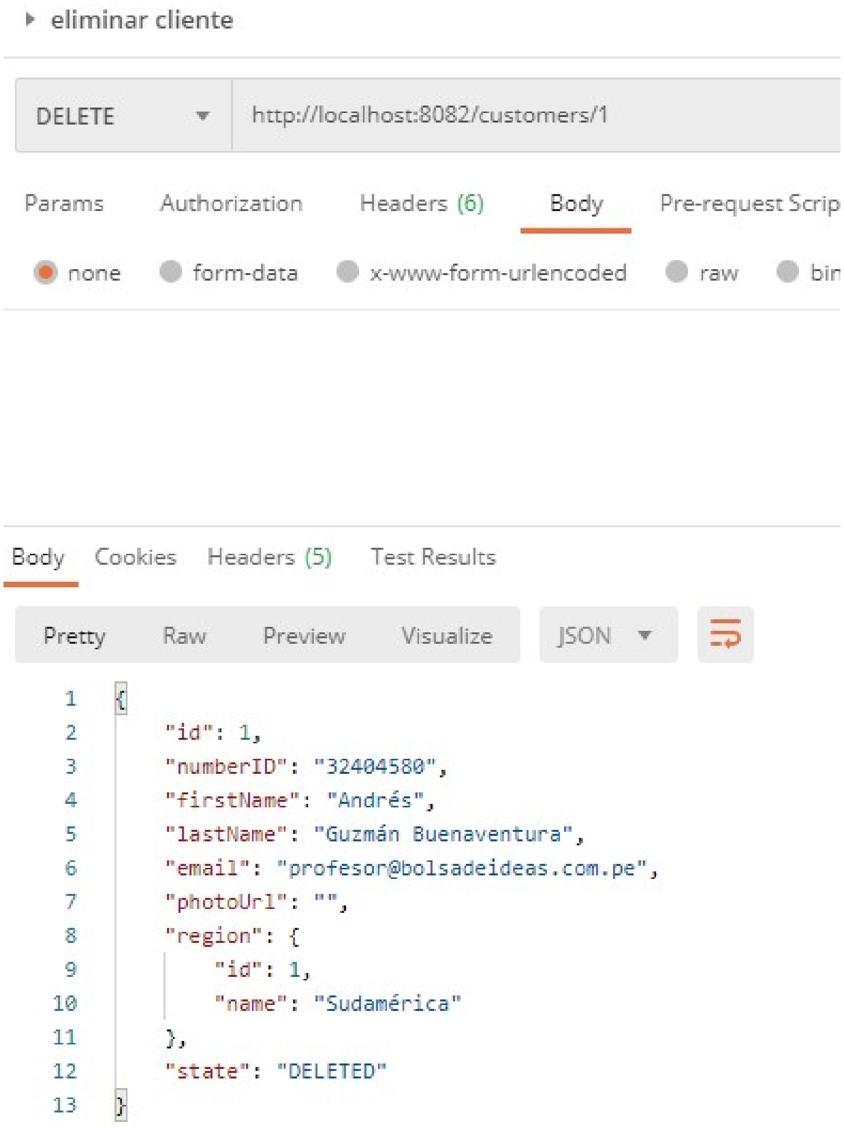
PUT http://localhost:8082/customers/1

Body: ● raw

```
{
        "id": 1,
        "numberID": "32404580",
        "firstName": "Andrés",
        "lastName": "Guzmán Buenaventura",
        "email": "profesor@bolsadeideas.com.pe",
        "photoUrl": "",
        "region": {
```

Pretty JSON

```
{
    "id": 1,
    "numberID": "32404580",
    "firstName": "Andrés",
    "lastName": "Guzmán Buenaventura",
    "email": "profesor@bolsadeideas.com.pe",
    "photoUrl": "",
    "region": {
        "id": 1,
        "name": "Sudamérica"
    },
    "state": "CREATED"
}
```



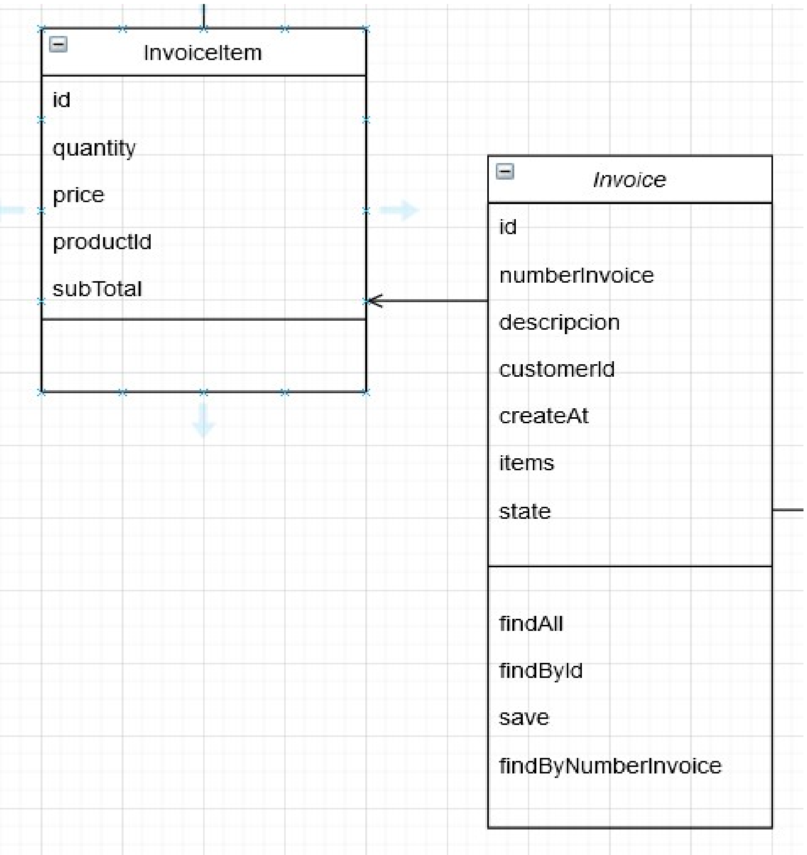
▸ eliminar cliente

DELETE http://localhost:8082/customers/1

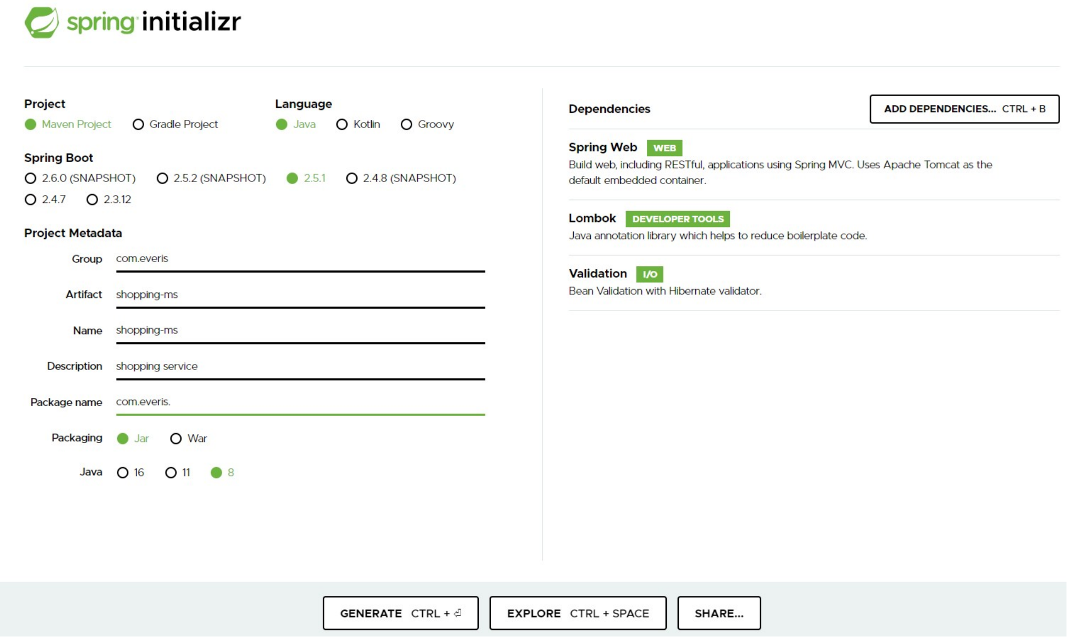
● none ○ form-data ○ x-www-form-urlencoded ○ raw ○ bir

```
{
    "id": 1,
    "numberID": "32404580",
    "firstName": "Andrés",
    "lastName": "Guzmán Buenaventura",
    "email": "profesor@bolsadeideas.com.pe",
    "photoUrl": "",
    "region": {
        "id": 1,
        "name": "Sudamérica"
    },
    "state": "DELETED"
}
```

Servicios rest : shopping-ms



1.- Creación de proyecto del microservicio



Project
● Maven Project ○ Gradle Project

Language
● Java ○ Kotlin ○ Groovy

Spring Boot
○ 2.6.0 (SNAPSHOT) ○ 2.5.2 (SNAPSHOT) ● 2.5.1 ○ 2.4.8 (SNAPSHOT)
○ 2.4.7 ○ 2.3.12

Project Metadata

Group	com.everis
Artifact	shopping-ms
Name	shopping-ms
Description	shopping service
Package name	com.everis.
Packaging	● Jar ○ War
Java	○ 16 ○ 11 ● 8

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

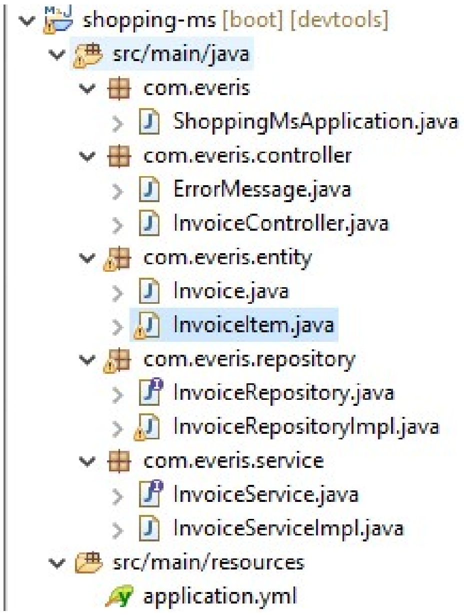
Validation I/O
Bean Validation with Hibernate validator.

Agregándose al pom las siguientes dependencias:

```
<dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
</dependency>
```

```
<dependency>
        <groupId>javax.validation</groupId>
       <artifactId>validation-api</artifactId>
</dependency>
```

2.- Creación de paquetes: controller, entity, repository y service



```
shopping-ms [boot] [devtools]
  src/main/java
    com.everis
      ShoppingMsApplication.java
    com.everis.controller
      ErrorMessage.java
      InvoiceController.java
    com.everis.entity
      Invoice.java
      InvoiceItem.java
    com.everis.repository
      InvoiceRepository.java
      InvoiceRepositoryImpl.java
    com.everis.service
      InvoiceService.java
      InvoiceServiceImpl.java
  src/main/resources
    application.yml
```

3.- implementación de clase entity: Invoice e InvoiceItem

```
package com.everis.entity;


import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

@Data
@AllArgsConstructor @Builder
public class Invoice {


    private Long id;

    private String numberInvoice;

    private String description;

    private Long customerId;

    private Date createAt;

    private List<InvoiceItem> items;
    private String state;

    public Invoice(){
        items = new ArrayList<>();
    }

}
package com.everis.entity;


import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;

@Data
@AllArgsConstructor @Builder
public class InvoiceItem  {

    private Long id;
    private Double quantity;
    private Double  price;
    private Long productId;
    private Double subTotal;

    public InvoiceItem(Long id, Double quantity, Double  price,Long 
productId){
        this.id = id;
        this.quantity = quantity;
        this.price = price;
        this.productId = productId;
    }
    public Double getSubTotal(){
        if (this.price >0  && this.quantity >0 ){
            return this.quantity * this.price;
        }else {
            return (double) 0;
        }
    }
    public InvoiceItem(){
        this.quantity=(double) 0;
        this.price=(double) 0;
    }
}
```

4.- creación de repositorio:

```
package com.everis.repository;

import com.everis.entity.Invoice;

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

public interface InvoiceRepository  {
    public Invoice findByNumberInvoice(String numberInvoice);

    public List<Invoice> findAll();
    public <S extends Invoice> S save(S entity);
    public Optional<Invoice> findById(Long id) ;
}
```

5.- implementación de repositorio , se tendrá en cuenta una carga de datos estáticos.

```
package com.everis.repository;

import com.everis.entity.Invoice;
import com.everis.entity.InvoiceItem;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Optional;

import org.springframework.stereotype.Repository;

@Repository
public class InvoiceRepositoryImpl implements InvoiceRepository {

      private List<InvoiceItem> items = new ArrayList<InvoiceItem>();
      private List<Invoice> invoices = new ArrayList<Invoice>();

      private void init() {
            InvoiceItem c1 = new InvoiceItem(1L, 1.0, 178.89,1L);
            InvoiceItem c2 = new InvoiceItem(2L, 2.0, 12.5,  2L);
            InvoiceItem c3 = new InvoiceItem(3L, 1.0, 40.06, 3L);

            items.add(c1);
            items.add(c2);
            items.add(c3);

            Invoice p1 = new Invoice(1L, "001","invoice office items" , 1L,
new Date(), items, "CREATED");

            invoices.add(p1);

      }

      public InvoiceRepositoryImpl( ) {
            init();
      }

      @Override
      public List<Invoice> findAll() {
            return invoices;
      }

      @Override
      public Optional<Invoice> findById(Long id) {
            return invoices.stream().filter(p ->   p.getId() ==
id).findFirst();
      }
      @Override
      public <S extends Invoice> S save(S entity) {
            invoices.add(entity);
            return entity;
      }

      @Override
      public Invoice findByNumberInvoice(String numberInvoice) {
            return invoices.stream().filter(p ->
p.getNumberInvoice().equals(numberInvoice)).findAny().orElse(null);
```

```
      }

}
```

6.- creación de la interfaz del servicio:

```
package com.everis.service;


import java.util.List;

import com.everis.entity.Invoice;

public interface InvoiceService {
    public List<Invoice> findInvoiceAll();

    public Invoice createInvoice(Invoice invoice);
    public Invoice updateInvoice(Invoice invoice);
    public Invoice deleteInvoice(Invoice invoice);

    public Invoice getInvoice(Long id);
}
```

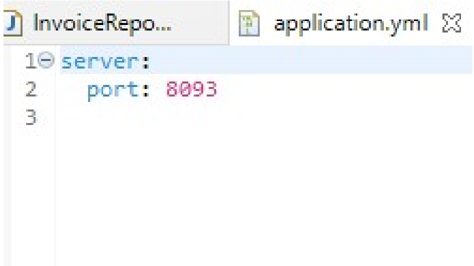
7.- implementación del servicio:

```
package com.everis.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.everis.entity.Invoice;
import com.everis.repository.InvoiceRepository;

import java.util.List;


@Service
public class InvoiceServiceImpl implements InvoiceService {

    @Autowired
    InvoiceRepository invoiceRepository;

    @Override
    public List<Invoice> findInvoiceAll() {
        return  invoiceRepository.findAll();
    }


    @Override
    public Invoice createInvoice(Invoice invoice) {
        Invoice invoiceDB = invoiceRepository.findByNumberInvoice (
invoice.getNumberInvoice () );
        if (invoiceDB !=null){
            return  invoiceDB;
        }
        invoice.setState("CREATED");
        return invoiceRepository.save(invoice);
    }


    @Override
    public Invoice updateInvoice(Invoice invoice) {
        Invoice invoiceDB = getInvoice(invoice.getId());
        if (invoiceDB == null){
            return  null;
        }
        invoiceDB.setCustomerId(invoice.getCustomerId());
        invoiceDB.setDescription(invoice.getDescription());
        invoiceDB.setNumberInvoice(invoice.getNumberInvoice());
        invoiceDB.getItems().clear();
        invoiceDB.setItems(invoice.getItems());
        return invoiceRepository.save(invoiceDB);
    }


    @Override
    public Invoice deleteInvoice(Invoice invoice) {
        Invoice invoiceDB = getInvoice(invoice.getId());
        if (invoiceDB == null){
            return  null;
        }
        invoiceDB.setState("DELETED");
        return invoiceRepository.save(invoiceDB);
    }

    @Override
    public Invoice getInvoice(Long id) {
        return invoiceRepository.findById(id).orElse(null);
    }
}
```

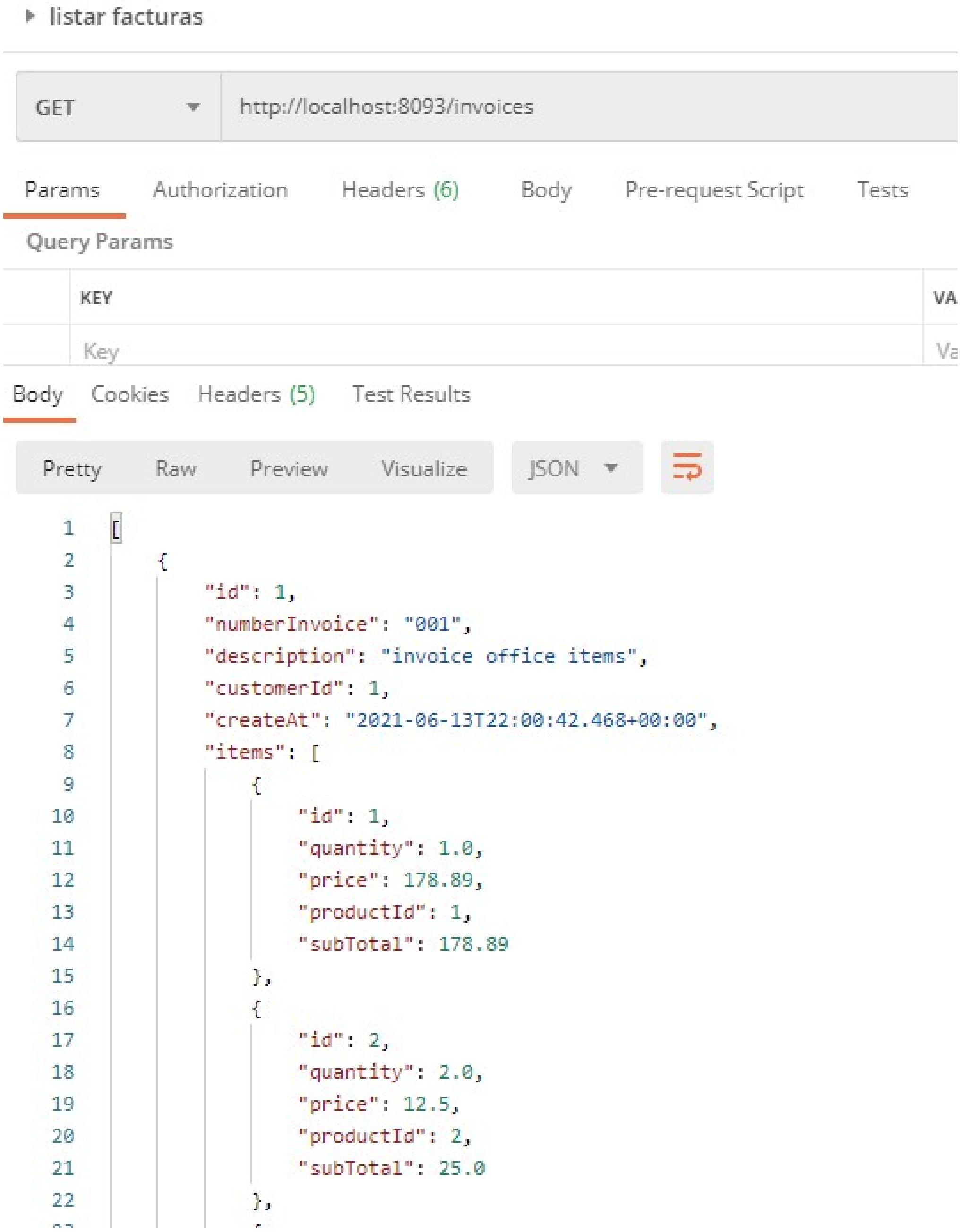
8.- implementación del controlador:

```
package com.everis.controller;

import com.everis.entity.Invoice;
import com.everis.service.InvoiceService;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.server.ResponseStatusException;

import javax.validation.Valid;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

@Slf4j
@RestController
@RequestMapping("/invoices")
public class InvoiceController {

    @Autowired
    InvoiceService invoiceService;

    @GetMapping
    public ResponseEntity<List<Invoice>> listAllInvoices() {
        List<Invoice> invoices = invoiceService.findInvoiceAll();
        if (invoices.isEmpty()) {
            return  ResponseEntity.noContent().build();
        }
        return  ResponseEntity.ok(invoices);
    }

    @GetMapping(value = "/{id}")
    public ResponseEntity<Invoice> getInvoice(@PathVariable("id") long id) {
        log.info("Fetching Invoice with id {}", id);
        Invoice invoice  = invoiceService.getInvoice(id);
        if (null == invoice) {
            log.error("Invoice with id {} not found.", id);
            return  ResponseEntity.notFound().build();
        }
        return  ResponseEntity.ok(invoice);
    }

    @PostMapping
    public ResponseEntity<Invoice> createInvoice(@Valid @RequestBody Invoice
invoice, BindingResult result) {
        log.info("Creating Invoice : {}", invoice);
        if (result.hasErrors()){
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST,
this.formatMessage(result));
        }
        Invoice invoiceDB = invoiceService.createInvoice (invoice);

        return  ResponseEntity.status( HttpStatus.CREATED).body(invoiceDB);
    }

    @PutMapping(value = "/{id}")
    public ResponseEntity<?> updateInvoice(@PathVariable("id") long id,
@RequestBody Invoice invoice) {
        log.info("Updating Invoice with id {}", id);

        invoice.setId(id);
        Invoice currentInvoice=invoiceService.updateInvoice(invoice);

        if (currentInvoice == null) {
            log.error("Unable to update. Invoice with id {} not found.", id);
            return  ResponseEntity.notFound().build();
        }
        return  ResponseEntity.ok(currentInvoice);
    }

    @DeleteMapping(value = "/{id}")
    public ResponseEntity<Invoice> deleteInvoice(@PathVariable("id") long id)
{
        log.info("Fetching & Deleting Invoice with id {}", id);

        Invoice invoice = invoiceService.getInvoice(id);
        if (invoice == null) {
            log.error("Unable to delete. Invoice with id {} not found.", id);
            return  ResponseEntity.notFound().build();
        }
        invoice = invoiceService.deleteInvoice(invoice);
        return ResponseEntity.ok(invoice);
    }

    private String formatMessage( BindingResult result){
        List<Map<String,String>> errors = result.getFieldErrors().stream()
                .map(err ->{
                    Map<String,String> error =  new HashMap<>();
                    error.put(err.getField(), err.getDefaultMessage());
                    return error;
                }).collect(Collectors.toList());
        ErrorMessage errorMessage = ErrorMessage.builder()
                .code("01")
                .messages(errors).build();
        ObjectMapper mapper = new ObjectMapper();
        String jsonString="";
        try {
            jsonString = mapper.writeValueAsString(errorMessage);
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
        return jsonString;
    }
}
```

Configuración de properties :



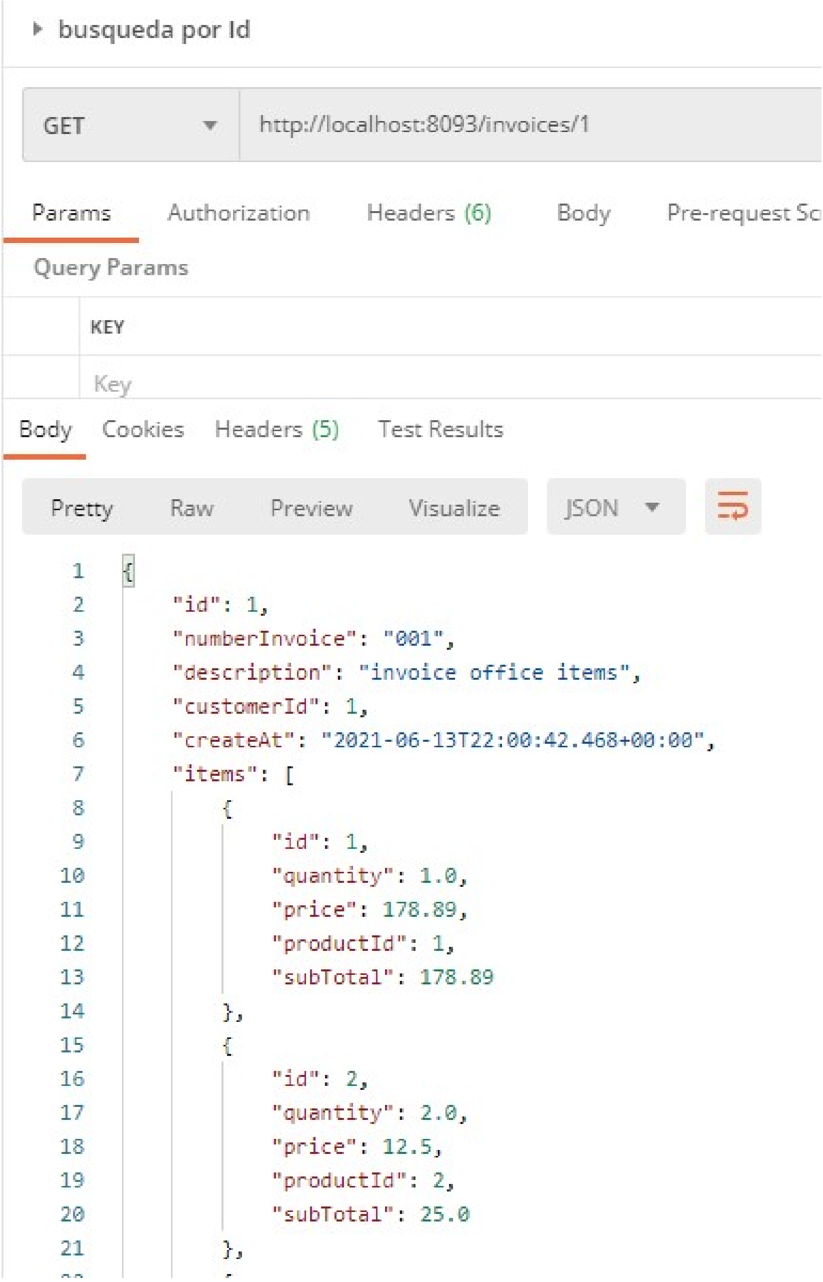
```yaml
server:
  port: 8093
```



listar facturas

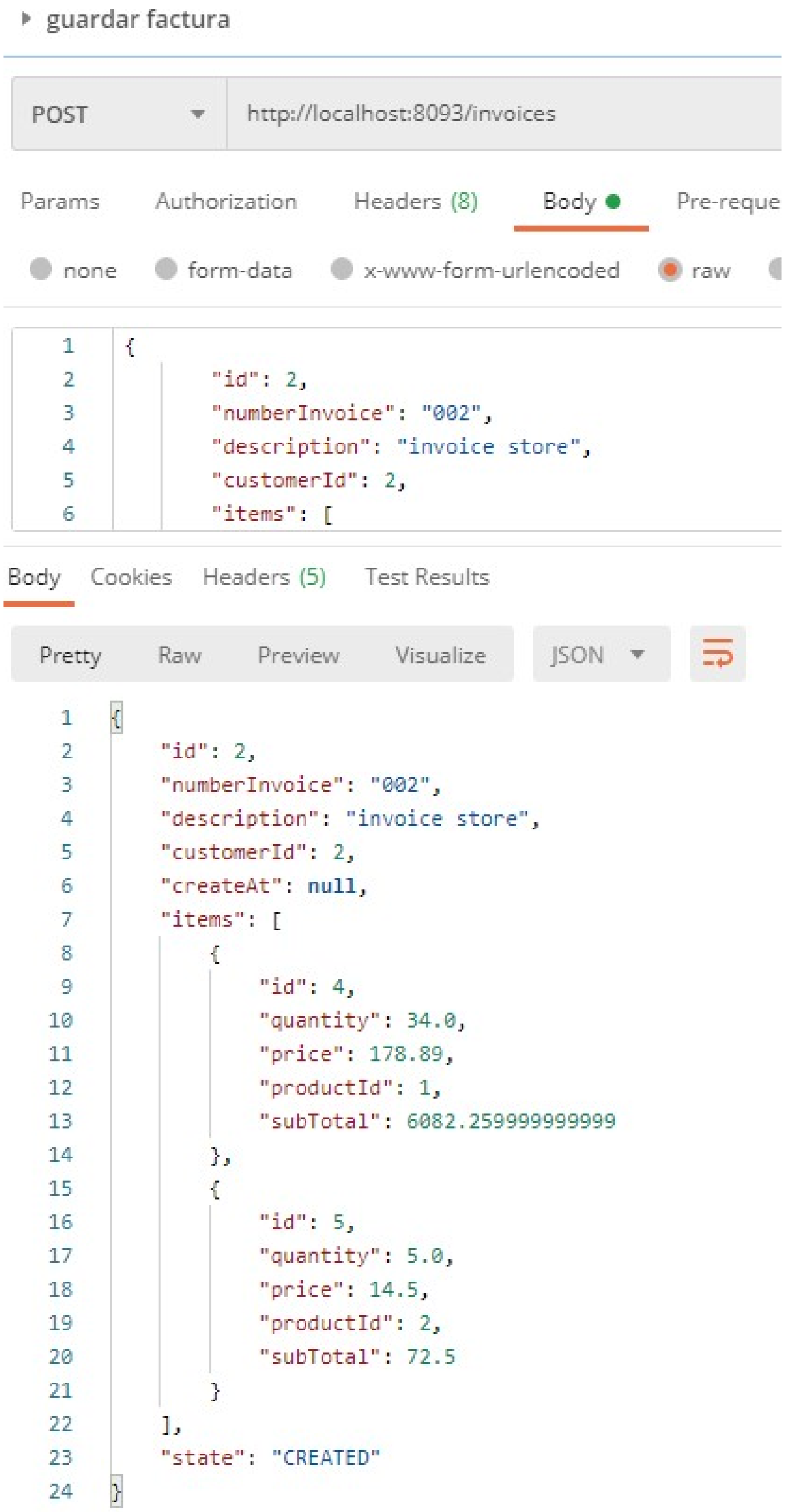
GET http://localhost:8093/invoices

```json
[
    {
        "id": 1,
        "numberInvoice": "001",
        "description": "invoice office items",
        "customerId": 1,
        "createAt": "2021-06-13T22:00:42.468+00:00",
        "items": [
            {
                "id": 1,
                "quantity": 1.0,
                "price": 178.89,
                "productId": 1,
                "subTotal": 178.89
            },
            {
                "id": 2,
                "quantity": 2.0,
                "price": 12.5,
                "productId": 2,
                "subTotal": 25.0
            },
```



GET http://localhost:8093/invoices/1

```
{
    "id": 1,
    "numberInvoice": "001",
    "description": "invoice office items",
    "customerId": 1,
    "createAt": "2021-06-13T22:00:42.468+00:00",
    "items": [
        {
            "id": 1,
            "quantity": 1.0,
            "price": 178.89,
            "productId": 1,
            "subTotal": 178.89
        },
        {
            "id": 2,
            "quantity": 2.0,
            "price": 12.5,
            "productId": 2,
            "subTotal": 25.0
        },
        {
```

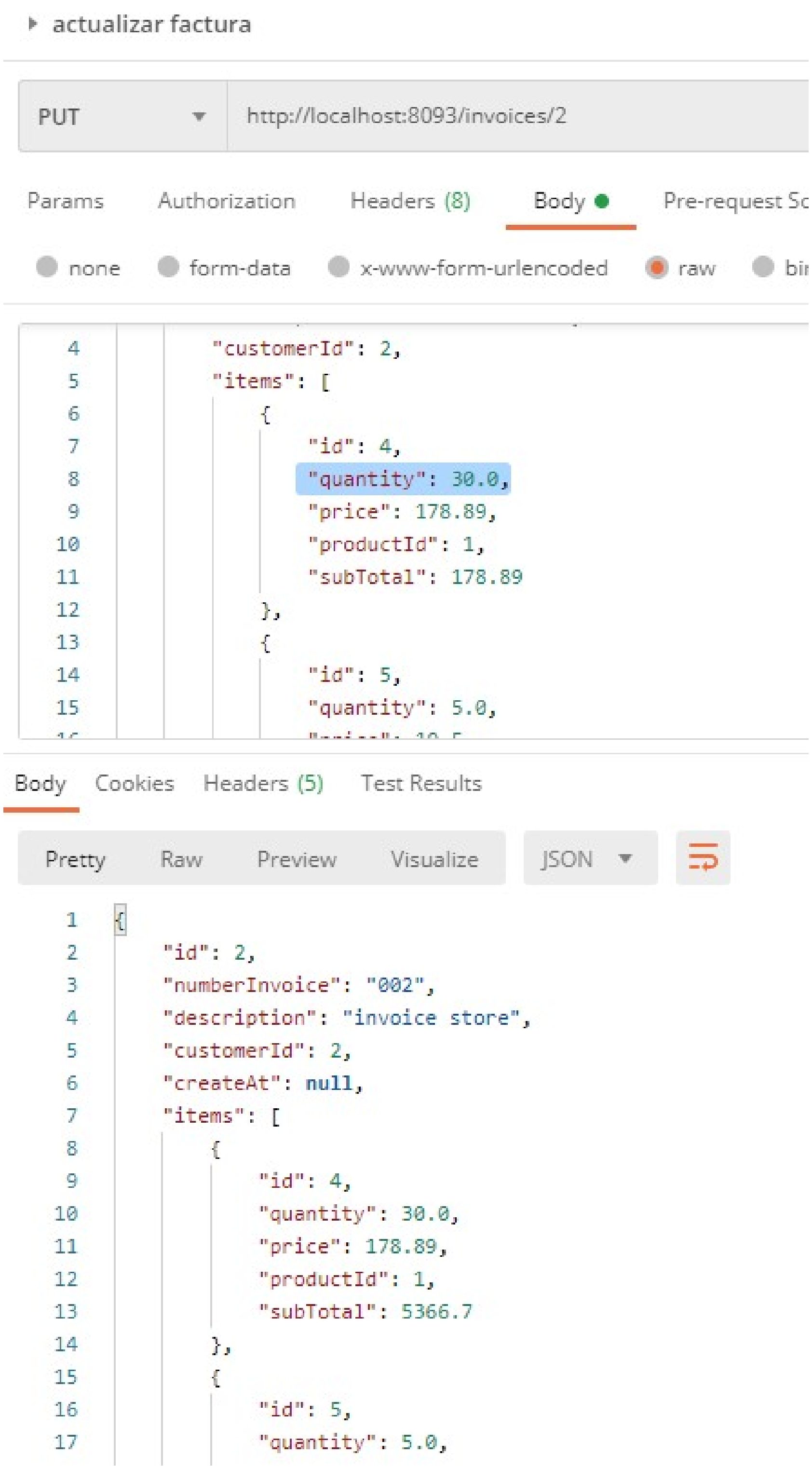


guardar factura

POST http://localhost:8093/invoices

```
{
        "id": 2,
        "numberInvoice": "002",
        "description": "invoice store",
        "customerId": 2,
        "items": [
```

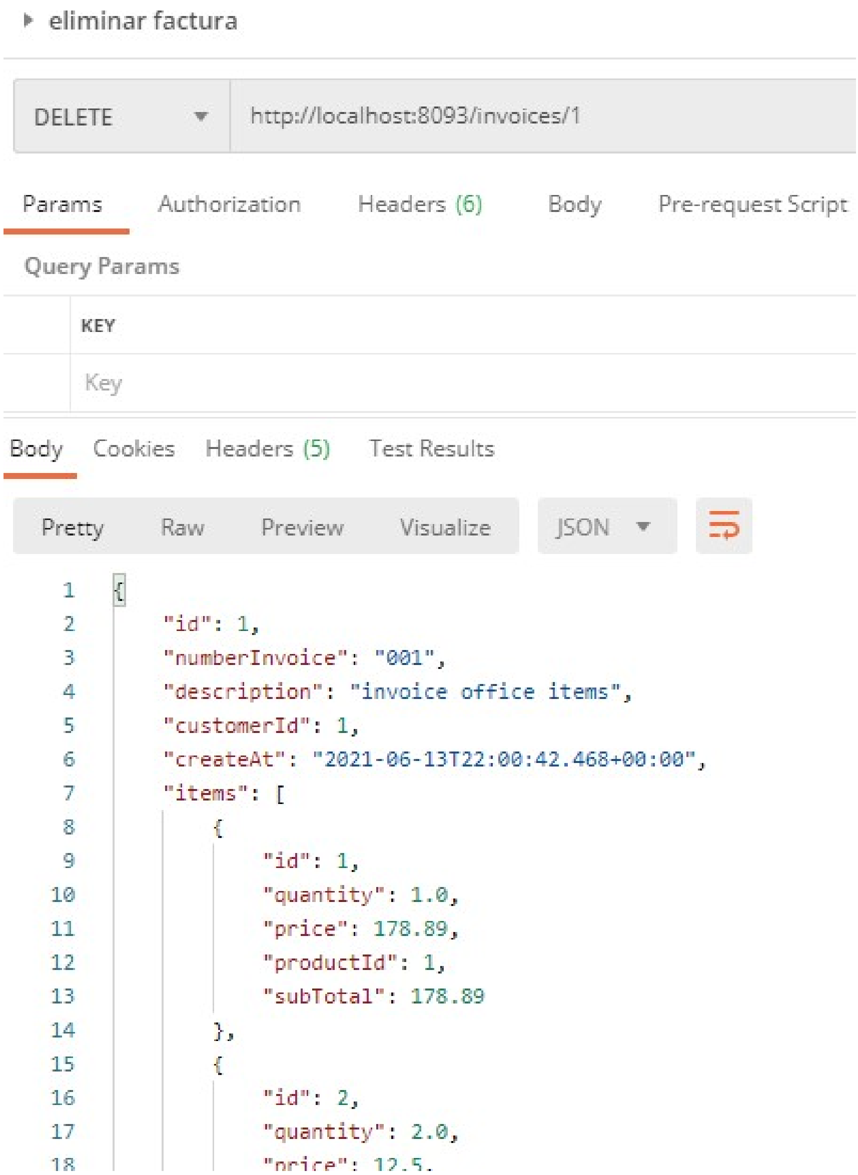
```
{
    "id": 2,
    "numberInvoice": "002",
    "description": "invoice store",
    "customerId": 2,
    "createAt": null,
    "items": [
        {
            "id": 4,
            "quantity": 34.0,
            "price": 178.89,
            "productId": 1,
            "subTotal": 6082.259999999999
        },
        {
            "id": 5,
            "quantity": 5.0,
            "price": 14.5,
            "productId": 2,
            "subTotal": 72.5
        }
    ],
    "state": "CREATED"
}
```



http://localhost:8093/invoices/2

```
        "customerId": 2,
        "items": [
            {
                "id": 4,
                "quantity": 30.0,
                "price": 178.89,
                "productId": 1,
                "subTotal": 178.89
            },
            {
                "id": 5,
                "quantity": 5.0,
```

```
{
    "id": 2,
    "numberInvoice": "002",
    "description": "invoice store",
    "customerId": 2,
    "createAt": null,
    "items": [
        {
            "id": 4,
            "quantity": 30.0,
            "price": 178.89,
            "productId": 1,
            "subTotal": 5366.7
        },
        {
            "id": 5,
            "quantity": 5.0,
```



http://localhost:8093/invoices/1

```
{
    "id": 1,
    "numberInvoice": "001",
    "description": "invoice office items",
    "customerId": 1,
    "createAt": "2021-06-13T22:00:42.468+00:00",
    "items": [
        {
            "id": 1,
            "quantity": 1.0,
            "price": 178.89,
            "productId": 1,
            "subTotal": 178.89
        },
        {
            "id": 2,
            "quantity": 2.0,
            "price": 12.5,
```