

Universidad Mariano Gálvez

Centro Universitario Boca del Monte

Facultad de Ingeniería en Sistemas de Información

Ingeniería en Sistemas de Información y Ciencias de la Computación

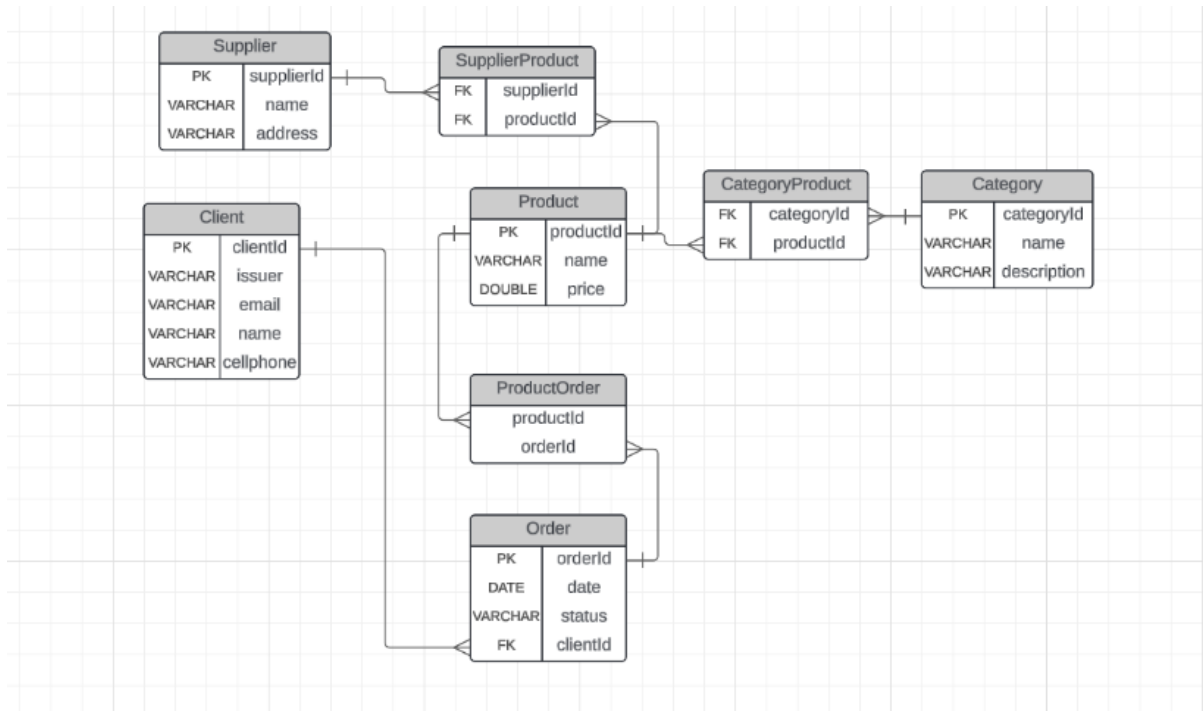


Christopher Herrera 7690-20-20773

Guatemala, 28 septiembre de 2024

El siguiente proyecto tiene como objetivo documentar las cosas utilizadas para el desarrollo.

El proyecto sigue el siguiente diagrama de base de datos.



Para su desarrollo se usó Spring Boot con Maven el cual es un framework de Java, se usaron las siguientes dependencias:

- JPA
- MYSQL-CONNECTOR-J
- LOMBOK

```

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.32</version>

```

El código está desarrollado bajo el estándar **clean code** para escribir código limpio y entendible y los principios **solid** implementando únicas responsabilidades, uso de interfaces etc, y patrones de diseño como el patrón **facade** para el manejo de inyección de dependencias y el patrón **prototype** para copiar objetos.

El proyecto en esta versión se maneja inserción de clientes en la base de datos a través de una API.

The screenshot displays a REST client interface with a POST request to `localhost:8080/client/`. The request body is a JSON object with the following fields: `issuer` (435656), `name` (Pepe Ochoa), `email` (pochoa@gmail.com), and `cellphone` (98565263). The response body is also a JSON object, including an additional `clientId` field (3) along with the same other fields. The interface includes tabs for Params, Authorization, Headers (8), Body, Scripts, and Settings. The Body tab is active, showing the raw JSON. Below the request, there are tabs for Body, Cookies, Headers (5), and Test Results. The Body tab is active, showing the response JSON in a Pretty format.

```
POST localhost:8080/client/

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSC

1 {
2   "issuer": "435656",
3   "name": "Pepe Ochoa",
4   "email": "pochoa@gmail.com",
5   "cellphone": "98565263"
6 }
```

```
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

1 {
2   "clientId": 3,
3   "issuer": "435656",
4   "name": "Pepe Ochoa",
5   "email": "pochoa@gmail.com",
6   "cellphone": "98565263"
7 }
```

RE ORDER BY

client_id	issuer	email	name	cellphone
1	106748793	chris.syst3@gmail.com	Christopher Herrera	45899016
3	435656	pochoa@gmail.com	Pepe Ochoa	98565263

Modificación del cliente:

PUT

localhost:8080/client/

Params

Authorization

Headers (8)

Body ●

Scripts

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

1

{

2

"clientId": 3,

3

"issuer": "3554545",

4

"name": "Pepe Doe",

5

"email": "pdoe@gmail.com",

6

"cellphone": "55556666"

7

}

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

↺

1

{

2

"clientId": 3,

3

"issuer": "3554545",

4

"name": "Pepe Doe",

5

"email": "pdoe@gmail.com",

6

"cellphone": "55556666"

7

}

client_id	issuer	email	name	cellphone
1	106748793	chris.syst3@gmail.com	Christopher Herrera	45899016
3	3554545	pdoe@gmail.com	Pepe Doe	55556666

Listar clientes:

GET

localhost:8080/client/

Params

Authorization

Headers (6)

Body

Scripts

Settings

Query Params

	Key
	Key

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1  [  
2    {  
3      "clientId": 1,  
4      "issuer": "106748793",  
5      "name": "Christopher Herrera",  
6      "email": "chris.syst3@gmail.com",  
7      "cellphone": "45899016"  
8    },  
9    {  
10     "clientId": 3,  
11     "issuer": "3554545",  
12     "name": "Pepe Doe",  
13     "email": "pdoe@gmail.com",  
14     "cellphone": "55556666"  
15   }  
16 ]
```

Obtener:

localhost:8080/client/3

GET localhost:8080/client/3

Params Authorization Headers (6) Body Scripts Settings

Query Params

	Key
	Key

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "clientId": 3,
3   "issuer": "3554545",
4   "name": "Pepe Doe",
5   "email": "pdoe@gmail.com",
6   "cellphone": "55556666"
7 }
```



Eliminar

DELETE

▼

localhost:8080/client/3

Params

Authorization

Headers (6)

Body

Scripts

Settings

Query Params

	Key	Value
	Key	Value

Body

Cookies

Headers (3)

Test Results

Pretty

Raw

Preview

Visualize

Text

▼

≡

1

	client_id	issuer	email	name	cellphone
1	1	106748793	chris.syst3@gmail.com	Christopher Herrera	45899016

# Inserción de categorías de productos

POSTlocalhost:8080/category/

ParamsAuthorizationHeaders (8)BodyScriptsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1{

2  "name": "Electrodomesticos",

3  "description": "Cosas electronicas"

4}

BodyCookiesHeaders (4)Test Results

200 OK15

PrettyRawPreviewVisualizeText

1

	category_id	name	description
1	1	Para el hogar	Articulos para el hogar
2	3	Para el Colegio	Cosas de alta calidad
3	5	Electrodomesticos	Cosas electronicas

## Actualización de categorías de productos

The screenshot shows a REST client interface with a PUT request to `localhost:8080/category/`. The 'Body' tab is selected, showing a JSON payload. Below the body, there are tabs for 'Body', 'Cookies', 'Headers (4)', and 'Test Results'. The 'Body' tab is further divided into 'Pretty', 'Raw', 'Preview', and 'Visualize', with 'Pretty' selected. A line number '1' is visible at the bottom left of the body section.

```
1 {
2   "categoryId": 5,
3   "name": "Electrodomesticos",
4   "description": "Cosas electronicas ---"
5 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize Text

1

WHERE		ORDER BY		
	category_id	name	description	
1	1	Para el hogar	Articulos para el hogar	
2	3	Para el Colegio	Cosas de alta calidad	
3	5	Electrodomesticos	Cosas electronicas ---	

Selección categorías

GET

▼

localhost:8080/category/5

ParamsAuthorizationHeaders (8)Body ●ScriptsSettings

Query Params

	Key	Value
	Key	Value

BodyCookiesHeaders (5)Test Results

PrettyRawPreviewVisualizeJSON ▼

1

{

2

"categoryId": 5,

3

"name": "Electrodomesticos",

4

"description": "Cosas electronicas ---"

5

}

Listar categorías

GETlocalhost:8080/category/

Params

Authorization

Headers (8)

Body

Scripts

Settings

Query Params

	Key	Value
	Key	Value

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

[

2

{

3

"categoryId": 1,

4

"name": "Para el hogar",

5

"description": "Articulos para el hogar"

6

},

7

{

8

"categoryId": 3,

9

"name": "Para el Colegio",

10

"description": "Cosas de alta calidad"

11

},

12

{

13

"categoryId": 5,

14

"name": "Electrodomesticos",

15

"description": "Cosas electronicas ---"

16

}

17

]

WHERE		ORDER BY	
	category_id	name	description
1	1	Para el hogar	Articulos para el hogar
2	3	Para el Colegio	Cosas de alta calidad
3	5	Electrodomesticos	Cosas electronicas ---

# Eliminar categoría

localhost:8080/category/5

DELETE

localhost:8080/category/5

Params

Authorization

Headers (8)

Body

Scripts

Settings

Query Params

Key	Value	Description
Key	Value	Description

200 OK20 ms123 B

PrettyRawPreviewVisualizeText

1

	category_id	name	description
1	1	Para el hogar	Articulos para el hogar
2	3	Para el Colegio	Cosas de alta calidad
3	5	Electrodomesticos	Cosas electronicas ---

## Crear productos

POST

localhost:8080/products/

Params

Authorization

Headers (8)

Body ●

Scripts

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

```
1 {
2   "name": "Pizzas",
3   "price": 120,
4   "categories": [3, 1]
5 }
```

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "productId": 5,
3   "name": "Pizzas",
4   "price": 120.0
5 }
```

	product_id	name	price
1	2	Patito	4
2	4	Bombones	120
3	5	Pizzas	120

## Listar los productos registrados

The screenshot displays a REST client interface with a GET request to `localhost:8080/products/`. The response body is shown in JSON format, containing an array of three product objects. The interface includes tabs for Params, Authorization, Headers (6), Body, Scripts, and Settings. The Body tab is active, showing the raw response. Below the request, there are tabs for Body, Cookies, Headers (5), and Test Results. The Body tab is active, showing the response in Pretty, Raw, Preview, and Visualize formats. The Pretty format is selected, displaying the JSON response with line numbers 1 through 17.

```
1  [
2    {
3      "productId": 2,
4      "name": "Patito",
5      "price": 4.0
6    },
7    {
8      "productId": 4,
9      "name": "Bombones",
10     "price": 120.0
11   },
12   {
13     "productId": 5,
14     "name": "Pizzas",
15     "price": 120.0
16   }
17 ]
```



# Eliminar productos

DELETE

localhost:8080/products/5

Params

Authorization

Headers (8)

Body

Scripts

Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body

Cookies

Headers (4)

Test Results

200 OK

29

Pretty

Raw

Preview

Visualize

Text

1

	WHERE		ORDER BY
	product_id	name	price
1	2	Patito	4
2	4	Bombones	120

## Crear ordenes

The screenshot displays a REST client interface for a POST request to `localhost:8080/order/`. The request body is a JSON object with `clientId` and `products` fields. The response body is a JSON object with `orderId`, `date`, `status`, `clientId`, and `products` fields.

**Request:**

```
POST localhost:8080/order/

{
  "clientId": 1,
  "products": [2, 4]
}
```

**Response:**

```
{
  "orderId": null,
  "date": "2024-09-29T03:58:31.696+00:00",
  "status": "Pendiente",
  "clientId": 1,
  "products": [
    2,
    4
  ]
}
```

## Actualizar el estado de la orden

The screenshot shows a REST client interface with the following components:

- Header:** HTTP icon, PROJECTO 2 PROGRA / Order - status
- Request Method and URL:** PUT localhost:8080/order/12
- Request Body Type:** none, form-data, x-www-form-urlencoded, **raw** (selected), binary, GraphQL
- Request Body:** 1
- Response Body:** Pretty, Raw, Preview, Visualize, JSON (selected), and a menu icon.
- Response Body Content:**

```
1 {
2   "orderId": 12,
3   "date": "2024-09-29T00:57:44.000+00:00",
4   "status": "DELIVERED",
5   "clientId": 1
6 }
```

Buscar orden

GET

localhost:8080/order/12

Params

Authorization

Headers (8)

Body

Scripts

Settings

Query Params

	Key	Value
	Key	Value

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    "orderId": 12,
3    "date": "2024-09-29T00:57:44.000+00:00",
4    "status": "DELIVERED",
5    "clientId": 1,
6    "products": [
7      {
8        "productId": 2,
9        "name": "Patito",
10       "price": 4.0
11      },
12      {
13        "productId": 4,
14        "name": "Bombones",
15        "price": 120.0
16      }
17    ]
18  }
```