DESARROLLO TALLER EDAKAFKA

Integrantes:

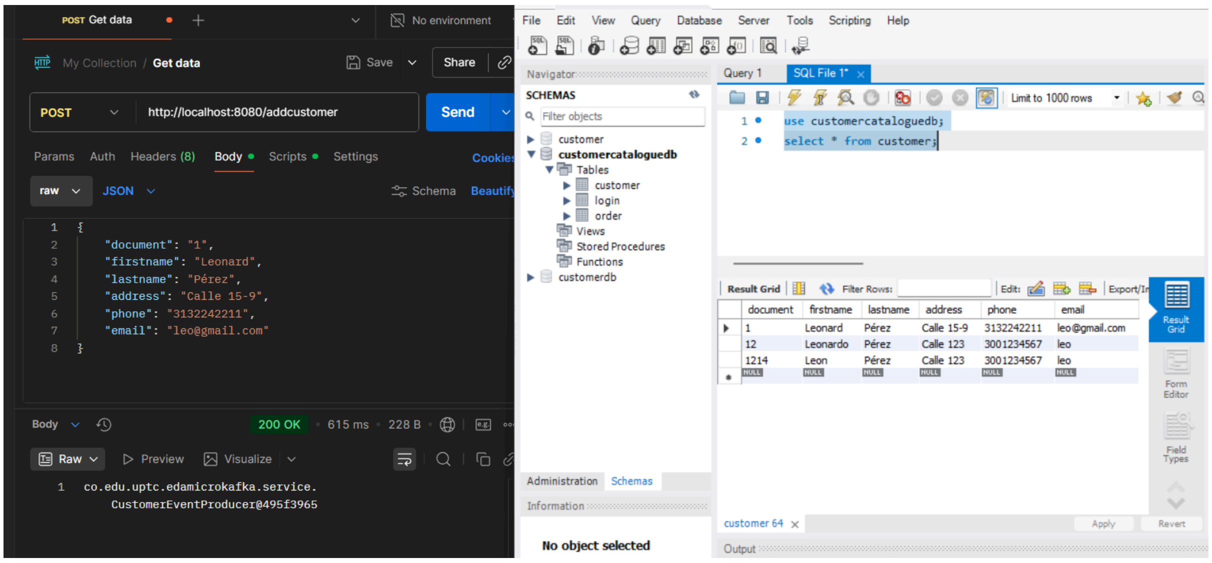
- Edison Ferney Gutierrez Buitrago
- Johan Sebastián Gil Salamanca
- Miguel Leonardo Avila Avila

Código fuente: https://github.com/mainAvilaMiguel/Taller-EDAKafka

Punto 1

Pruebe todos los microservicios de la entidad Customer, para asegurar que están respondiendo correctamente. Tome evidencia a través de imágenes de la correcta ejecución de la misma.

- **addcustomer**



POST http://localhost:8080/addcustomer

```
{
    "document": "1",
    "firstname": "Leonard",
    "lastname": "Pérez",
    "address": "Calle 15-9",
    "phone": "3132242211",
    "email": "leo@gmail.com"
}
```

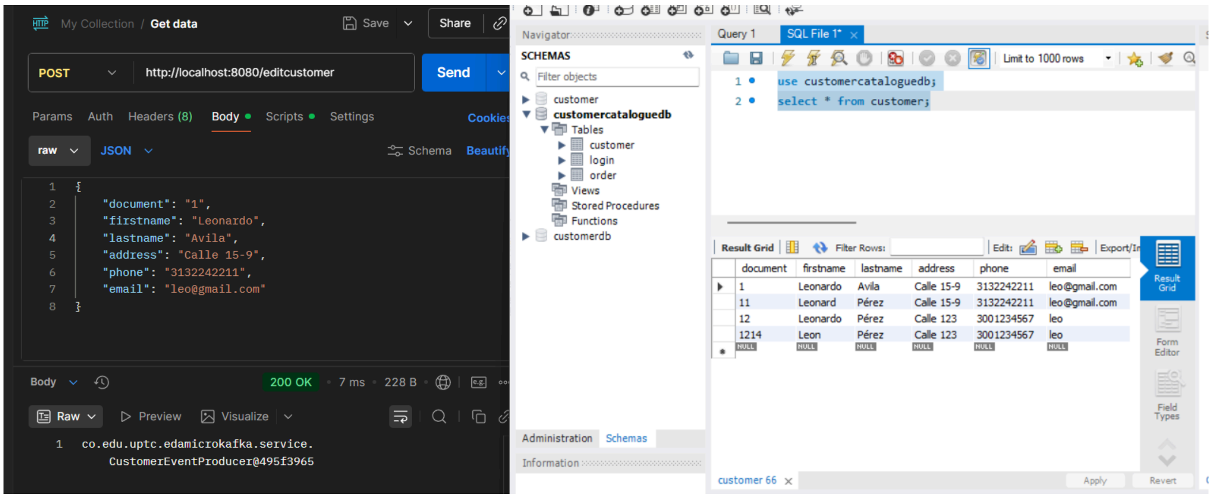
200 OK · 615 ms · 228 B

```
co.edu.uptc.edamicrokafka.service.
    CustomerEventProducer@495f3965
```

```
use customercataloguedb;
select * from customer;
```

document	firstname	lastname	address	phone	email
1	Leonard	Pérez	Calle 15-9	3132242211	leo@gmail.com
12	Leonardo	Pérez	Calle 123	3001234567	leo
1214	Leon	Pérez	Calle 123	3001234567	leo
NULL	NULL	NULL	NULL	NULL	NULL

- **editcustomer**



POST http://localhost:8080/editcustomer

```
{
    "document": "1",
    "firstname": "Leonardo",
    "lastname": "Avila",
    "address": "Calle 15-9",
    "phone": "3132242211",
    "email": "leo@gmail.com"
}
```

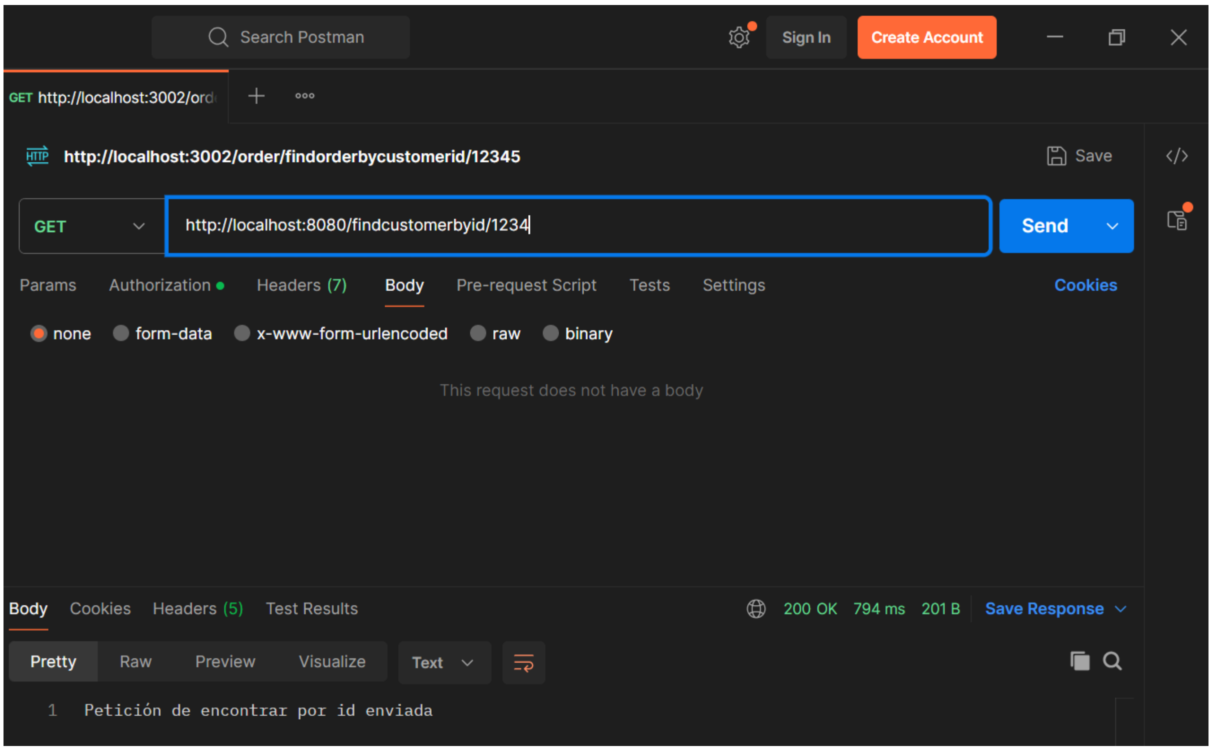
200 OK · 7 ms · 228 B

```
co.edu.uptc.edamicrokafka.service.
    CustomerEventProducer@495f3965
```

```
use customercataloguedb;
select * from customer;
```

document	firstname	lastname	address	phone	email
1	Leonardo	Avila	Calle 15-9	3132242211	leo@gmail.com
11	Leonard	Pérez	Calle 15-9	3132242211	leo@gmail.com
12	Leonardo	Pérez	Calle 123	3001234567	leo
1214	Leon	Pérez	Calle 123	3001234567	leo
NULL	NULL	NULL	NULL	NULL	NULL

- **findByID**



GET http://localhost:8080/findcustomerbyid/1234

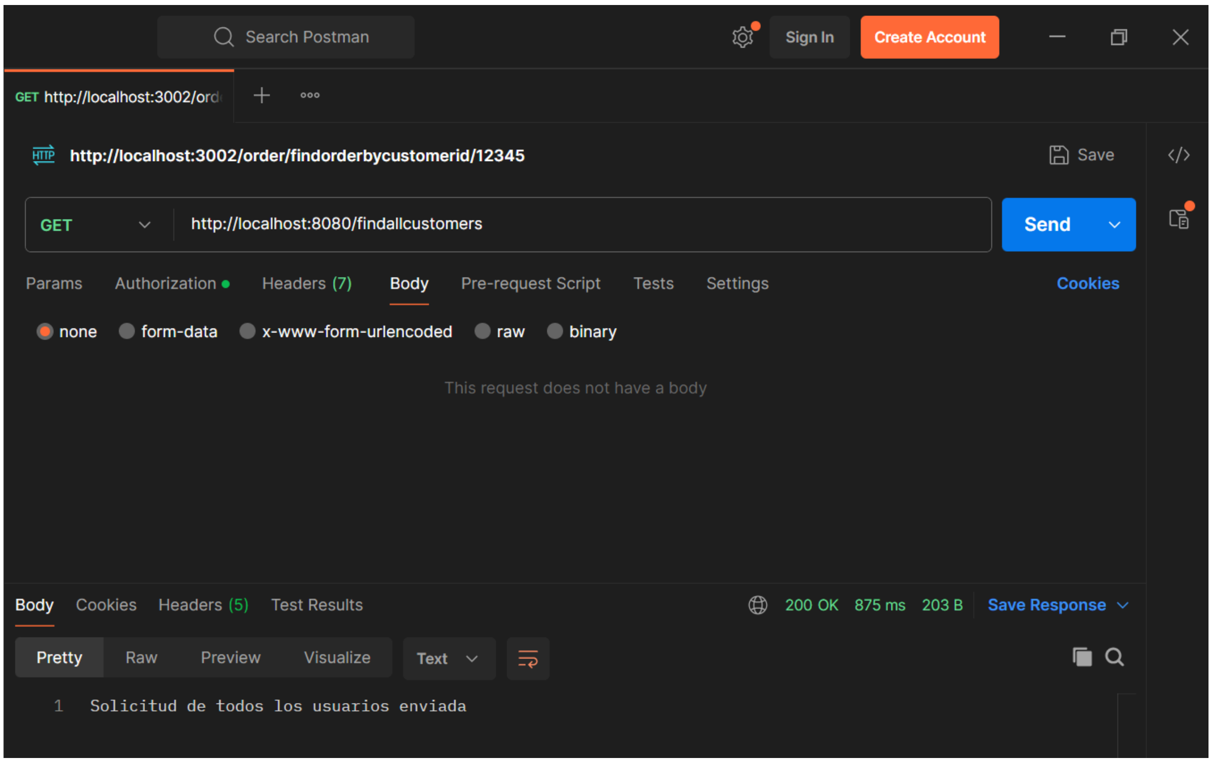
Body: none

This request does not have a body

200 OK 794 ms 201 B

```
Petición de encontrar por id enviada
```

- **findAllCustomers**



http://localhost:3002/order/findorderbycustomerid/12345

GET http://localhost:8080/findallcustomers

200 OK 875 ms 203 B

```
Solicitud de todos los usuarios enviada
```

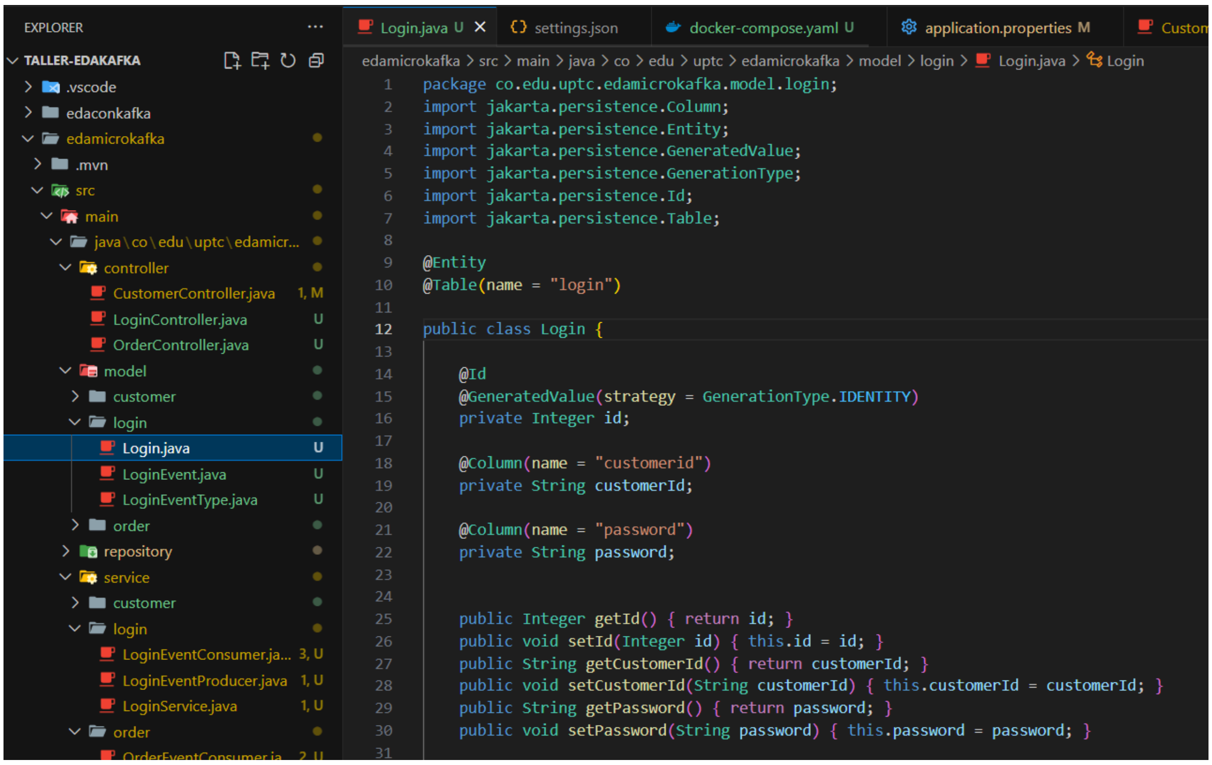
Punto 2

Basado en el presente taller, genere todas las clases para implementar el patrón EDA en las entidades Login y Order. Pruebe todos los microservicios de dichas entidades para asegurar que están respondiendo correctamente. Tome evidencia a través de imágenes de la correcta ejecución de las mismas.

Login

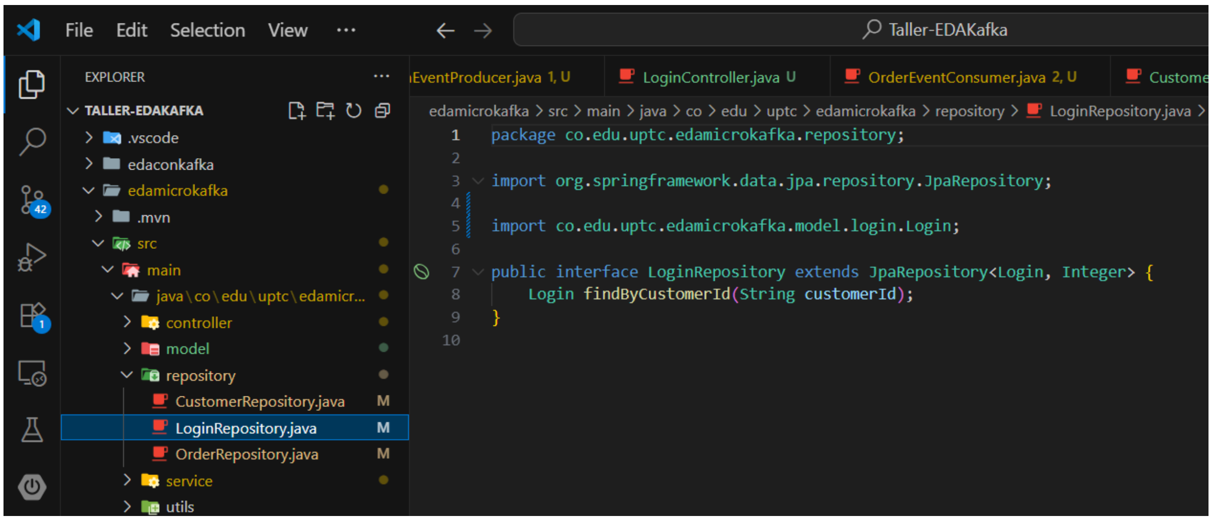
Model

Creación del model de login


```java
package co.edu.uptc.edamicrokafka.model.login;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "login")

public class Login {

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

    @Column(name = "customerid")
    private String customerId;

    @Column(name = "password")
    private String password;


    public Integer getId() { return id; }
    public void setId(Integer id) { this.id = id; }
    public String getCustomerId() { return customerId; }
    public void setCustomerId(String customerId) { this.customerId = customerId; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
```

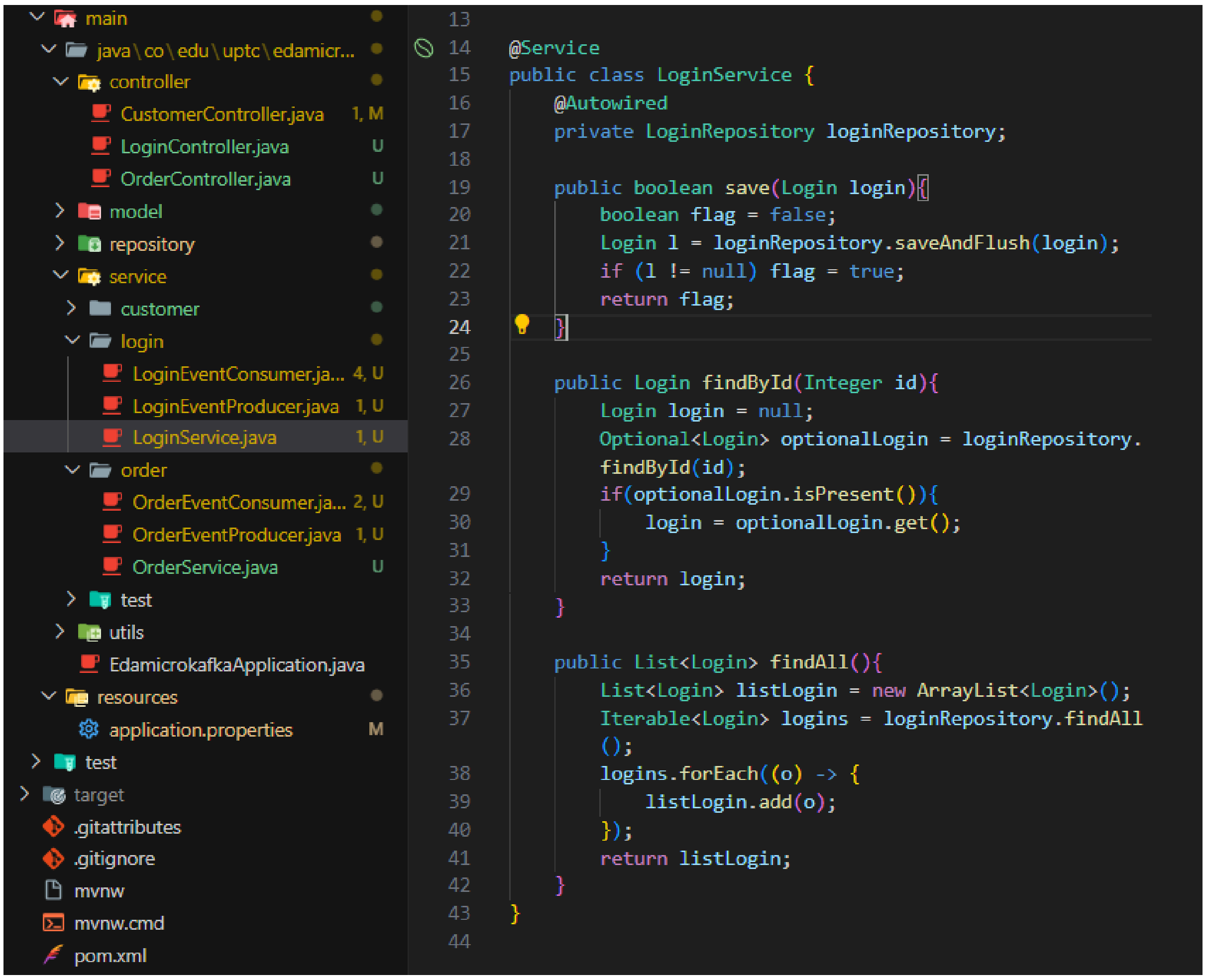
Repository

Creación del repository de login


```java
package co.edu.uptc.edamicrokafka.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import co.edu.uptc.edamicrokafka.model.login.Login;

public interface LoginRepository extends JpaRepository<Login, Integer> {
    Login findByCustomerId(String customerId);
}
```

Service

Service de login

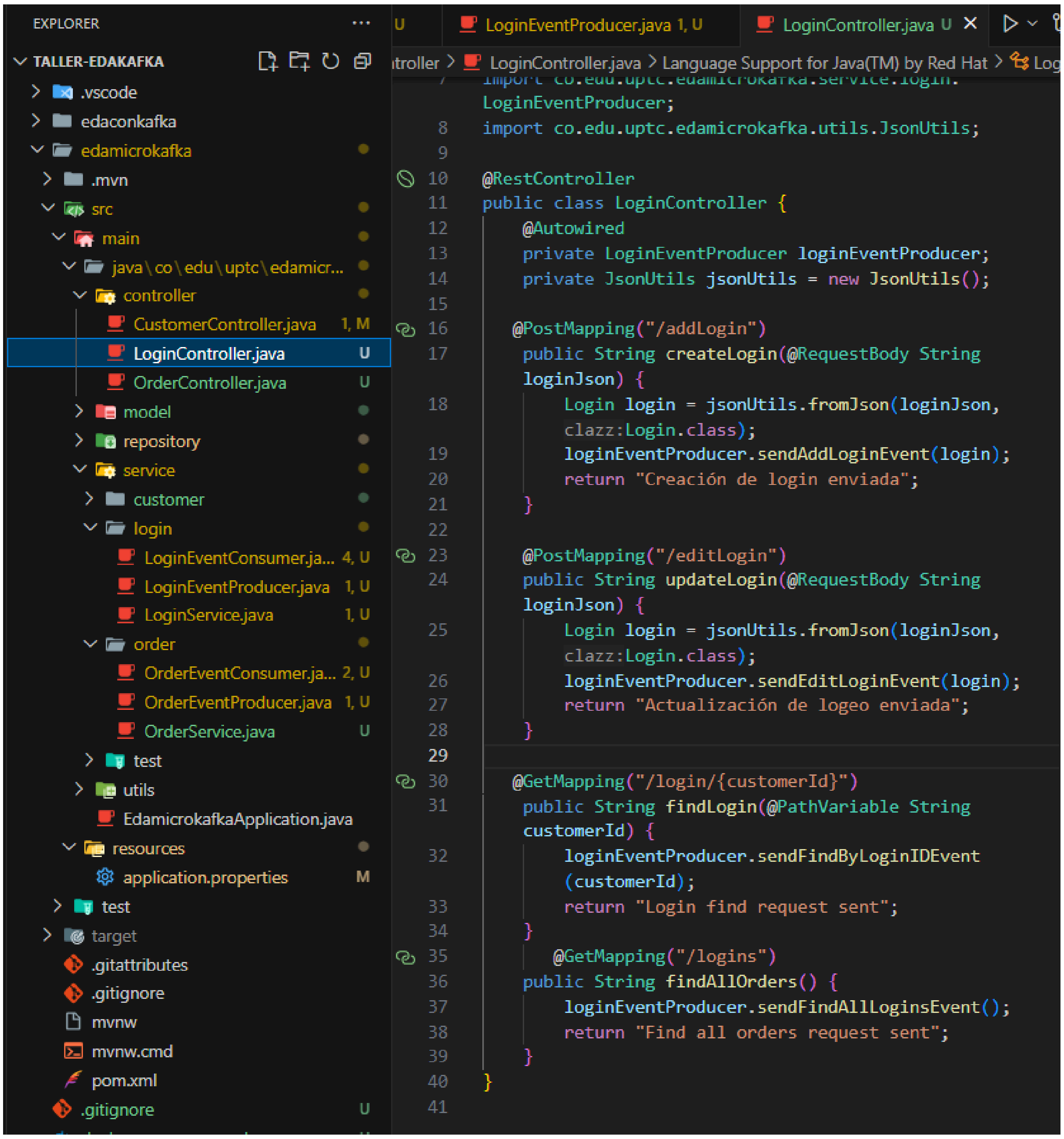


```
main
  java\co\edu\uptc\edamicr...
    controller
      CustomerController.java      1, M
      LoginController.java         U
      OrderController.java         U
    model
    repository
    service
      customer
      login
        LoginEventConsumer.ja...   4, U
        LoginEventProducer.java    1, U
        LoginService.java          1, U
      order
        OrderEventConsumer.ja...   2, U
        OrderEventProducer.java    1, U
        OrderService.java          U
      test
    utils
    EdamicrokafkaApplication.java
  resources
    application.properties         M
test
target
.gitattributes
.gitignore
mvnw
mvnw.cmd
pom.xml
```

```java

@Service
public class LoginService {
    @Autowired
    private LoginRepository loginRepository;

    public boolean save(Login login){
        boolean flag = false;
        Login l = loginRepository.saveAndFlush(login);
        if (l != null) flag = true;
        return flag;
    }

    public Login findById(Integer id){
        Login login = null;
        Optional<Login> optionalLogin = loginRepository.
        findById(id);
        if(optionalLogin.isPresent()){
            login = optionalLogin.get();
        }
        return login;
    }

    public List<Login> findAll(){
        List<Login> listLogin = new ArrayList<Login>();
        Iterable<Login> logins = loginRepository.findAll
        ();
        logins.forEach((o) -> {
            listLogin.add(o);
        });
        return listLogin;
    }
}

```

Controller

Controller de login



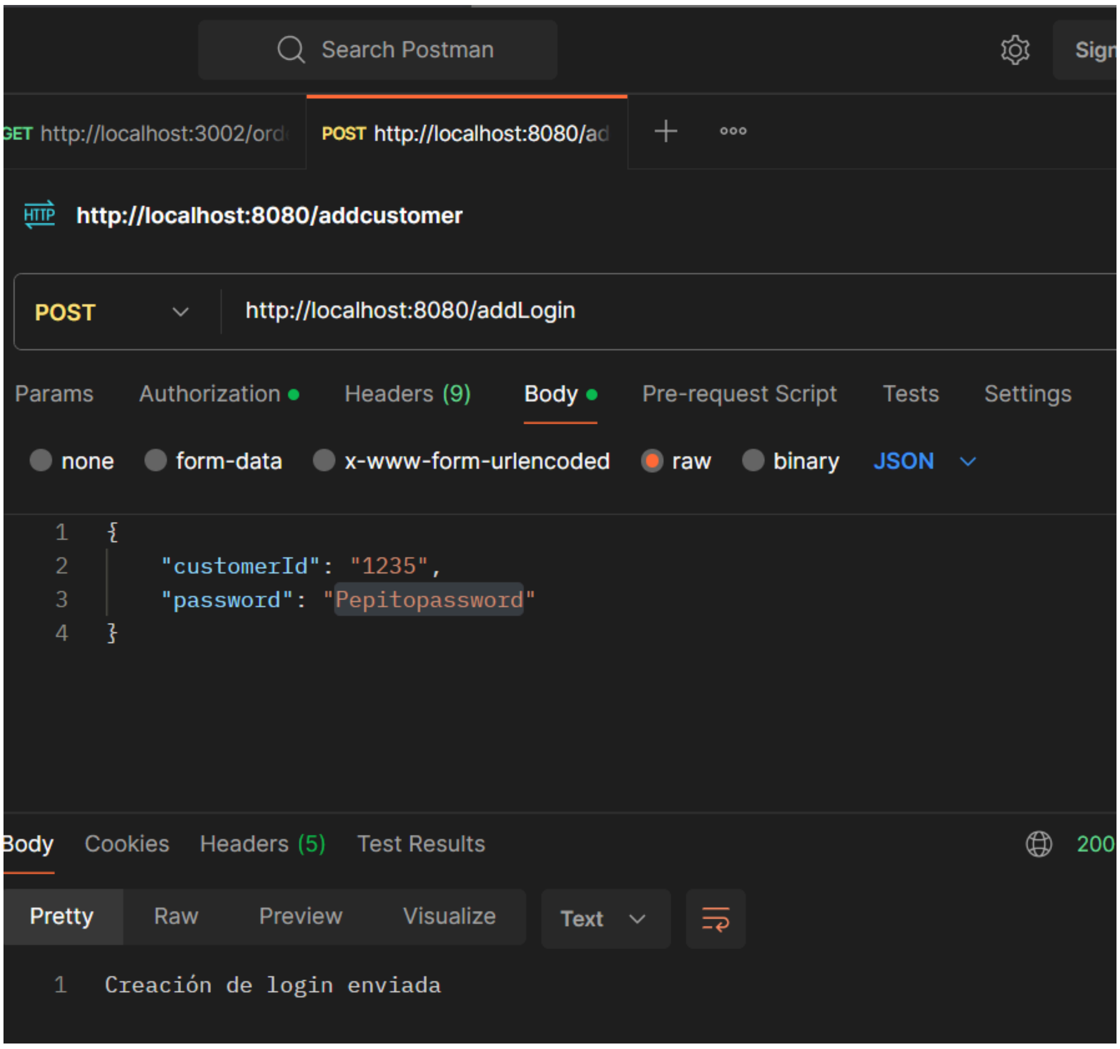
```
EXPLORER ···
TALLER-EDAKAFKA
  > .vscode
  > edaconkafka
  v edamicrokafka
    > .mvn
    v src
      v main
        v java\co\edu\uptc\edamicr...
          v controller
              CustomerController.java        1, M
              LoginController.java           U
              OrderController.java           U
          > model
          > repository
          v service
            > customer
            v login
                LoginEventConsumer.ja...     4, U
                LoginEventProducer.java      1, U
                LoginService.java            1, U
            v order
                OrderEventConsumer.ja...     2, U
                OrderEventProducer.java      1, U
                OrderService.java            U
            > test
          > utils
            EdamicrokafkaApplication.java
        v resources
            application.properties           M
      > test
    > target
      .gitattributes
      .gitignore
      mvnw
      mvnw.cmd
      pom.xml
    .gitignore                               U
```

```
U | LoginEventProducer.java 1, U | LoginController.java U ×
troller > LoginController.java > Language Support for Java(TM) by Red Hat > Log
```

```java
import co.edu.uptc.edamicrokafka.service.login.
LoginEventProducer;
import co.edu.uptc.edamicrokafka.utils.JsonUtils;

@RestController
public class LoginController {
    @Autowired
    private LoginEventProducer loginEventProducer;
    private JsonUtils jsonUtils = new JsonUtils();

   @PostMapping("/addLogin")
    public String createLogin(@RequestBody String
    loginJson) {
        Login login = jsonUtils.fromJson(loginJson,
        clazz:Login.class);
        loginEventProducer.sendAddLoginEvent(login);
        return "Creación de login enviada";
    }

    @PostMapping("/editLogin")
    public String updateLogin(@RequestBody String
    loginJson) {
        Login login = jsonUtils.fromJson(loginJson,
        clazz:Login.class);
        loginEventProducer.sendEditLoginEvent(login);
        return "Actualización de logeo enviada";
    }

   @GetMapping("/login/{customerId}")
    public String findLogin(@PathVariable String
    customerId) {
        loginEventProducer.sendFindByLoginIDEvent
        (customerId);
        return "Login find request sent";
    }
      @GetMapping("/logins")
    public String findAllOrders() {
        loginEventProducer.sendFindAllLoginsEvent();
        return "Find all orders request sent";
    }
}
```

Pruebas

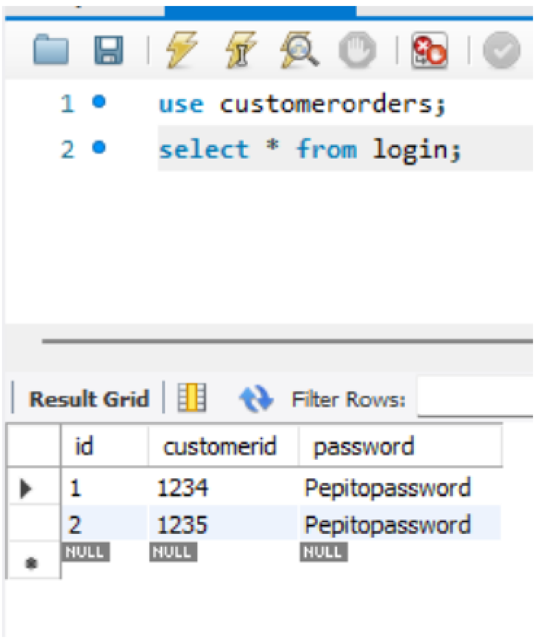
- **addlogin**



POST http://localhost:8080/addLogin

```
{
    "customerId": "1235",
    "password": "Pepitopassword"
}
```

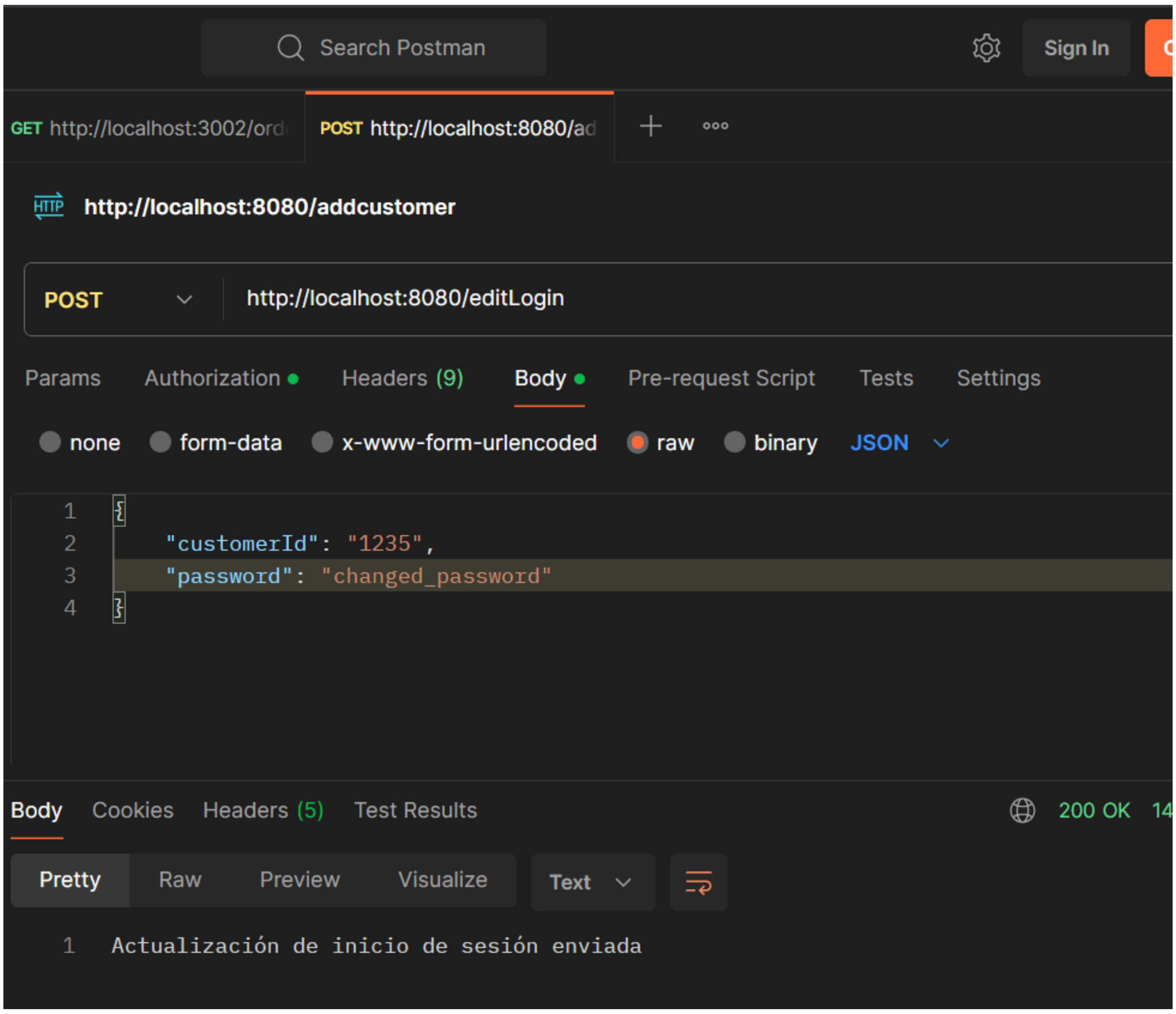
```
Creación de login enviada
```



```
use customerorders;
select * from login;
```

id	customerid	password
1	1234	Pepitopassword
2	1235	Pepitopassword
NULL	NULL	NULL

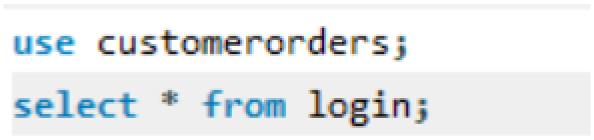
- editlogin



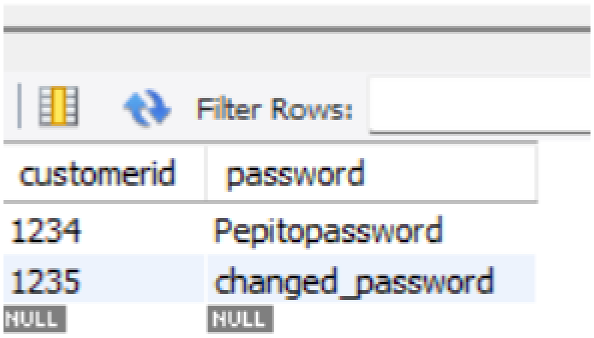
POST http://localhost:8080/editLogin

```
{
    "customerId": "1235",
    "password": "changed_password"
}
```

```
Actualización de inicio de sesión enviada
```

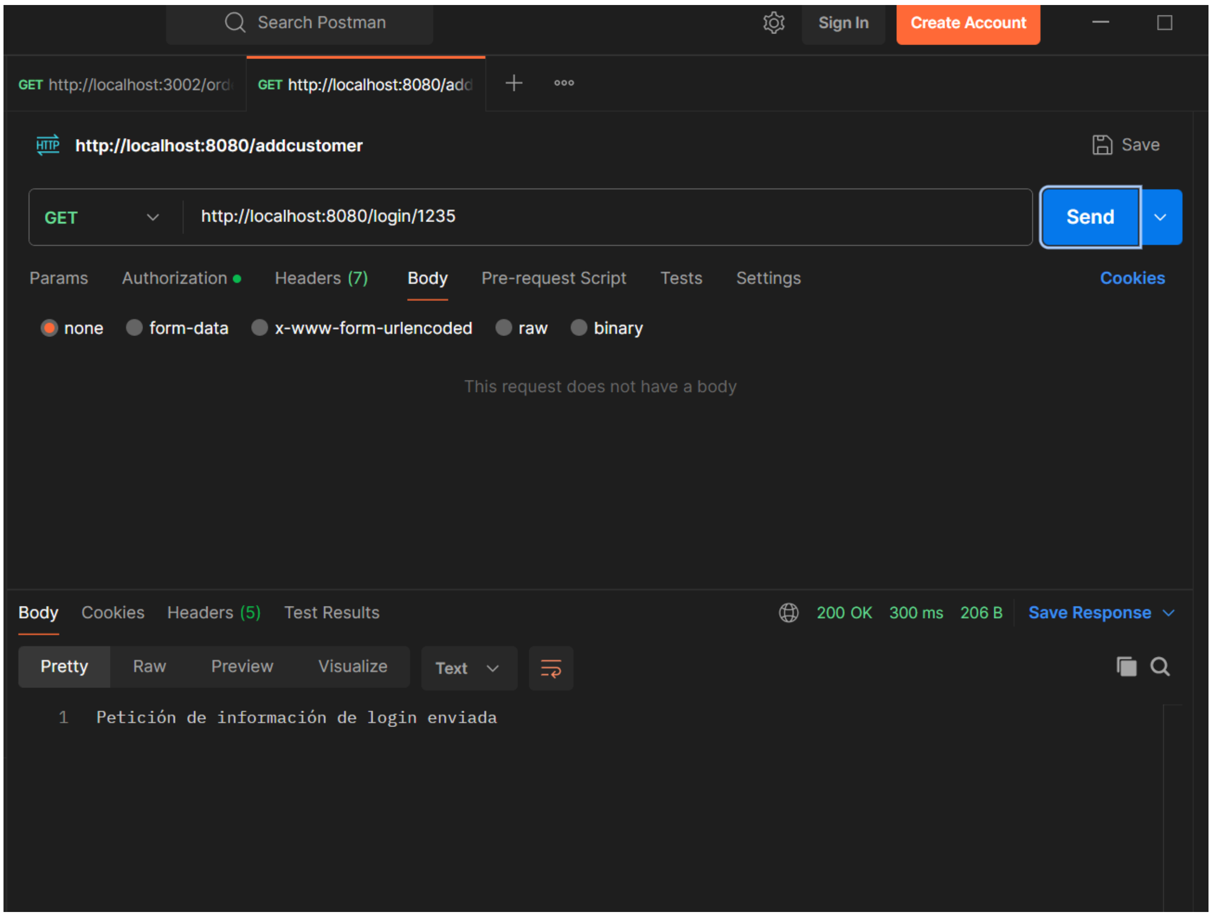


```
use customerorders;
select * from login;
```



customerid	password
1234	Pepitopassword
1235	changed_password
NULL	NULL

- Encontrar login por id



http://localhost:8080/addcustomer

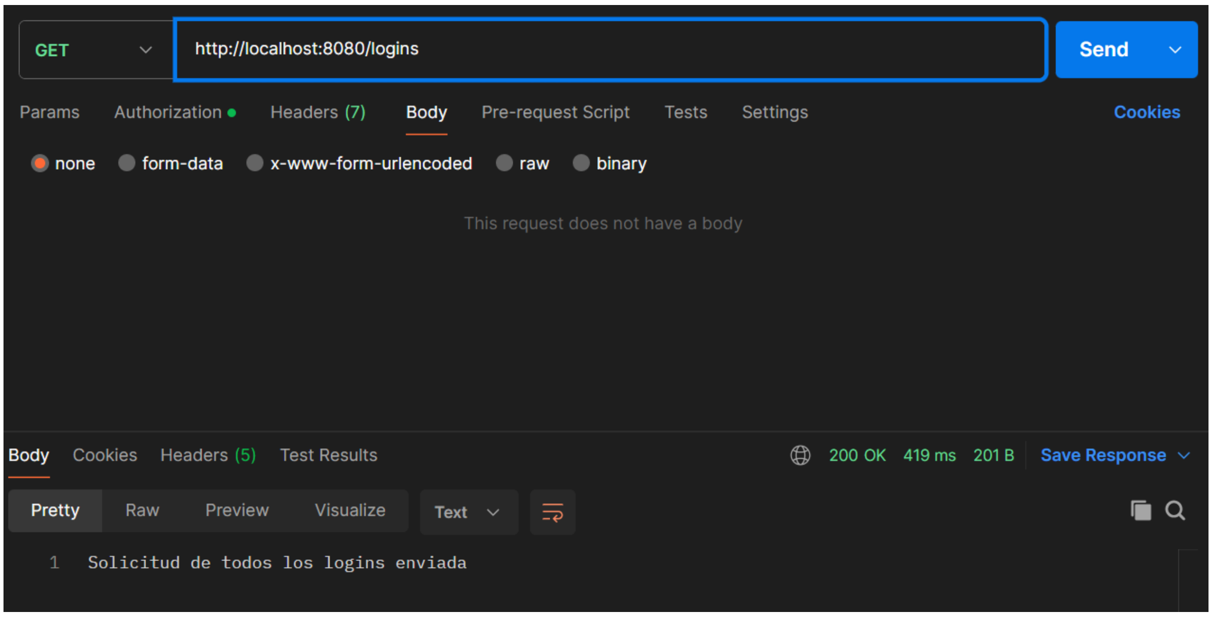
GET http://localhost:8080/login/1235

This request does not have a body

200 OK 300 ms 206 B

```
Petición de información de login enviada
```

- Obtener toda la información de login



GET http://localhost:8080/logins

This request does not have a body

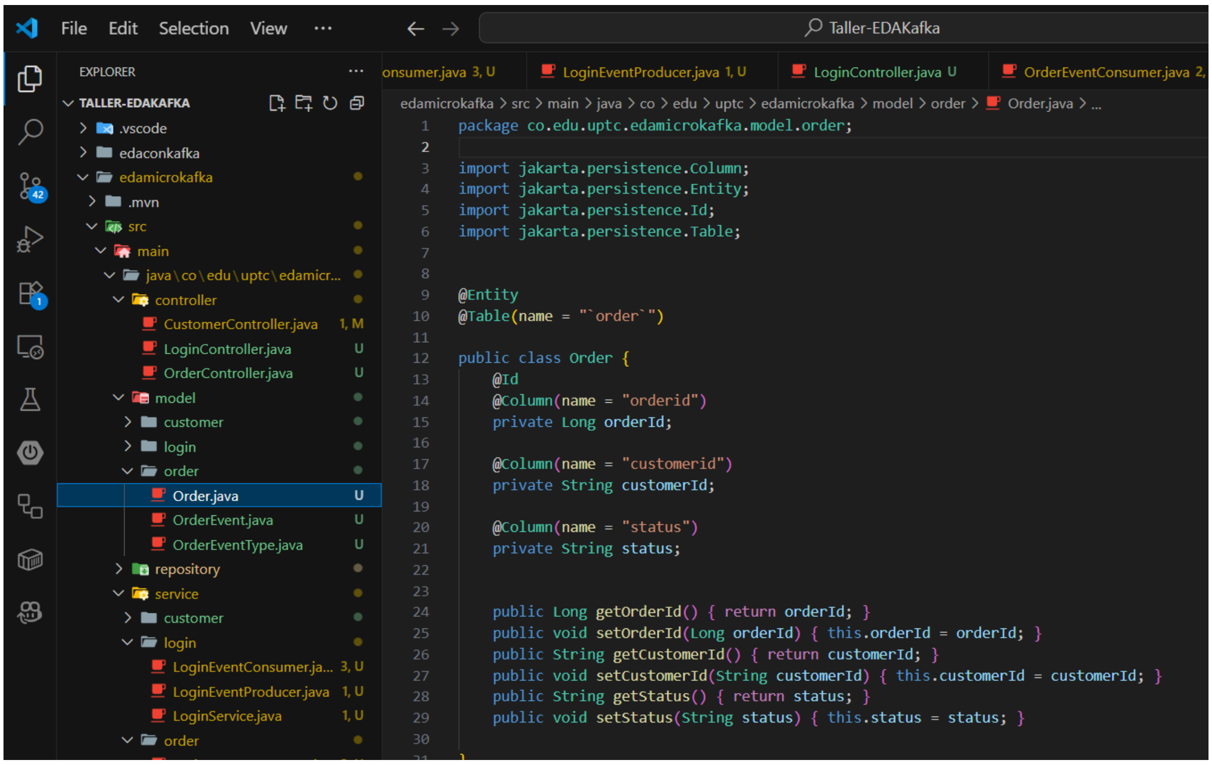
200 OK 419 ms 201 B

```
Solicitud de todos los logins enviada
```

Order

Model

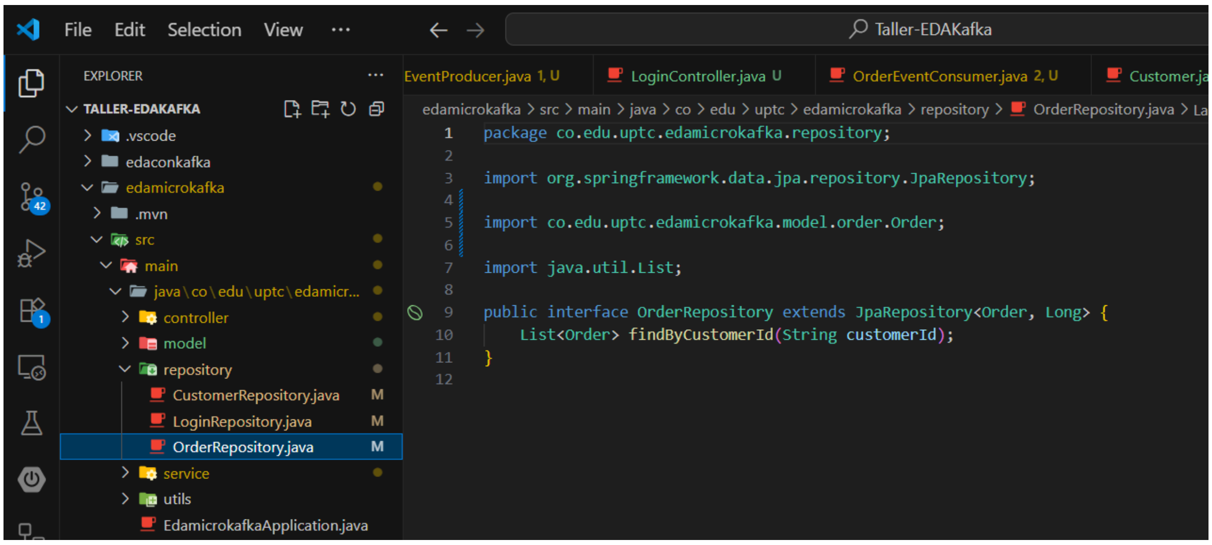
Creación del model de order



```java
package co.edu.uptc.edamicrokafka.model.order;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;


@Entity
@Table(name = "`order`")

public class Order {
    @Id
    @Column(name = "orderid")
    private Long orderId;

    @Column(name = "customerid")
    private String customerId;

    @Column(name = "status")
    private String status;


    public Long getOrderId() { return orderId; }
    public void setOrderId(Long orderId) { this.orderId = orderId; }
    public String getCustomerId() { return customerId; }
    public void setCustomerId(String customerId) { this.customerId = customerId; }
    public String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }

}
```

Repository

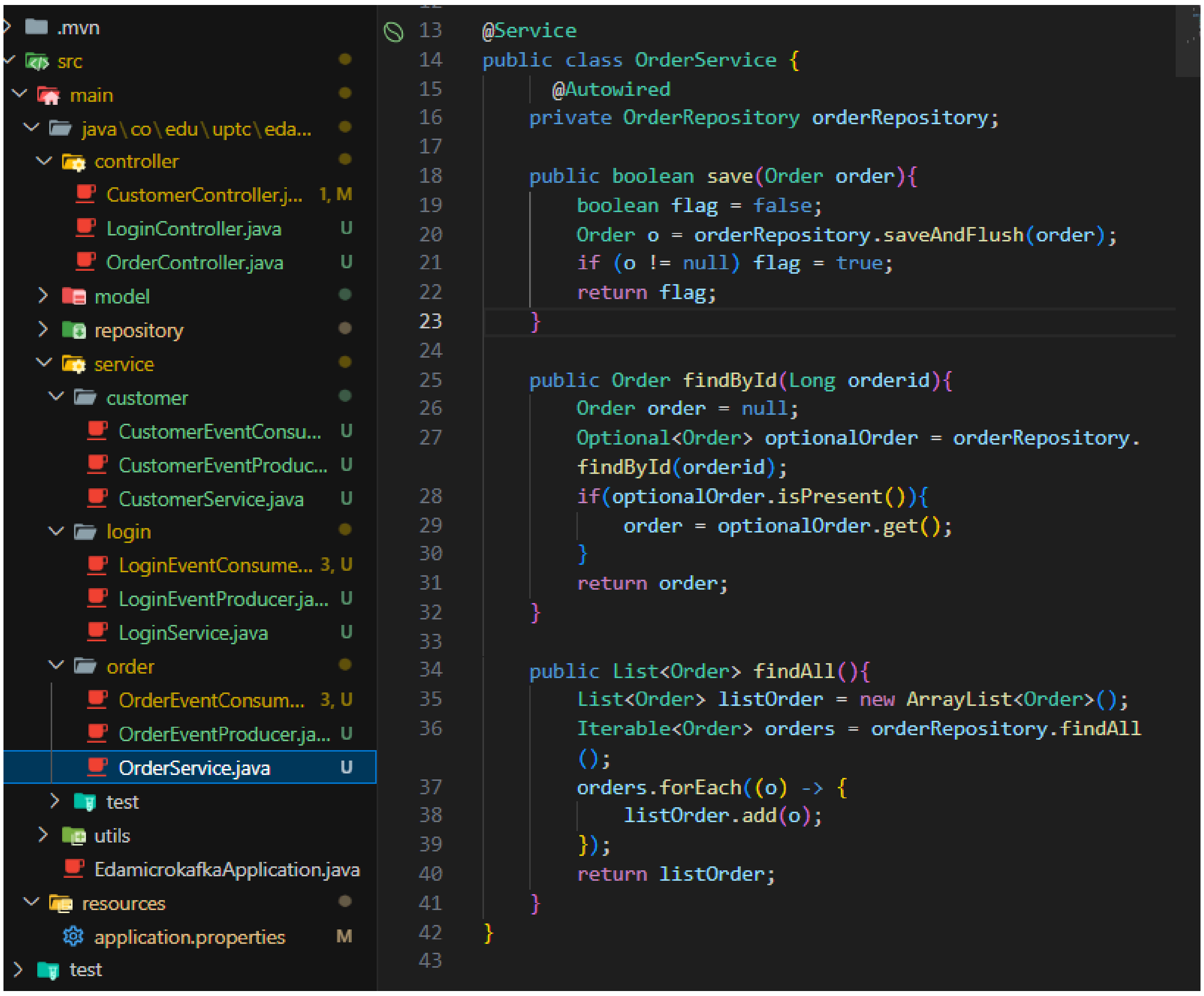
Creación del repository de order



```java
package co.edu.uptc.edamicrokafka.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import co.edu.uptc.edamicrokafka.model.order.Order;

import java.util.List;

public interface OrderRepository extends JpaRepository<Order, Long> {
    List<Order> findByCustomerId(String customerId);
}
```

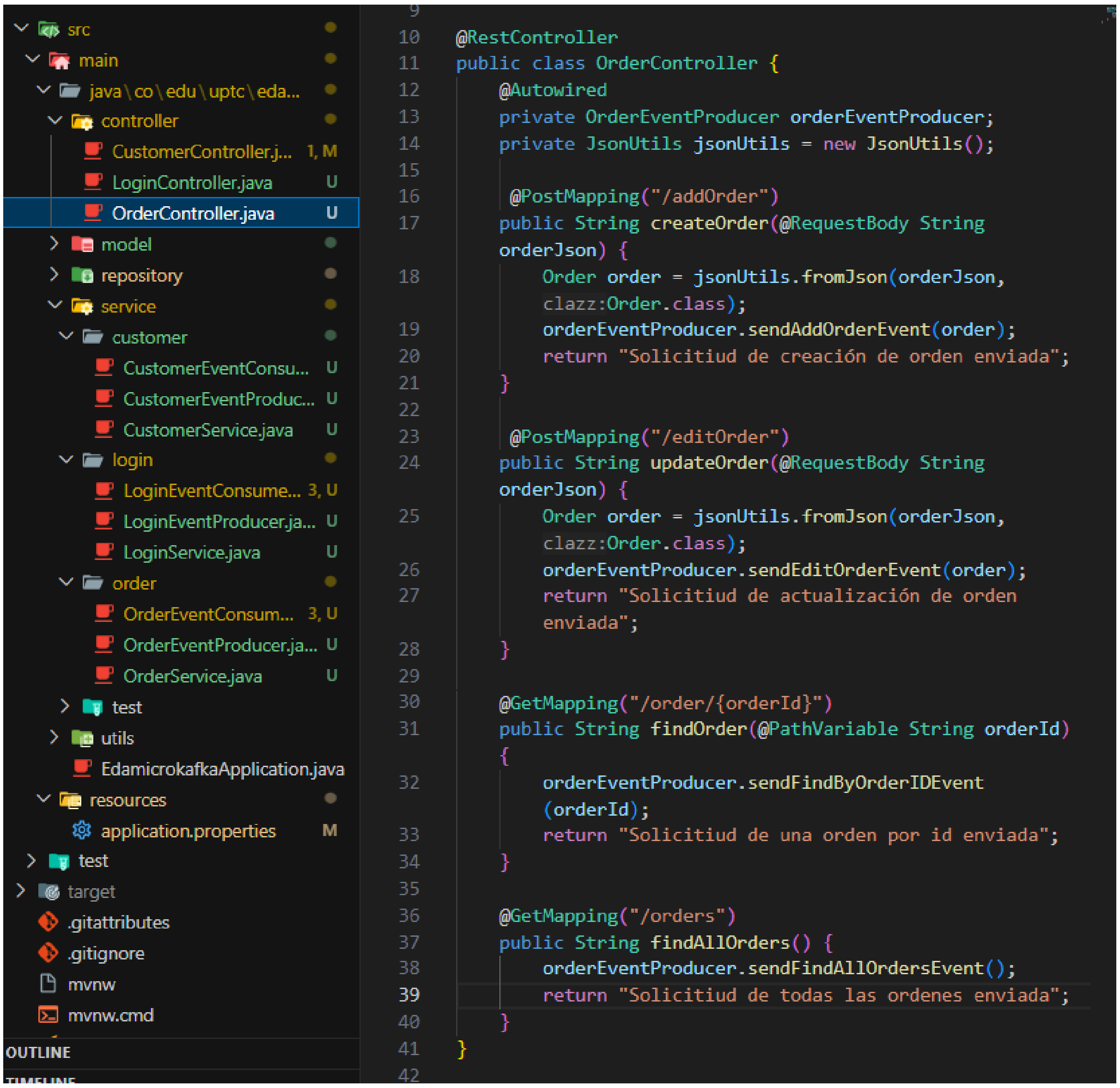
Service

Service de order



```java
@Service
public class OrderService {
    @Autowired
    private OrderRepository orderRepository;

    public boolean save(Order order){
        boolean flag = false;
        Order o = orderRepository.saveAndFlush(order);
        if (o != null) flag = true;
        return flag;
    }

    public Order findById(Long orderid){
        Order order = null;
        Optional<Order> optionalOrder = orderRepository.
        findById(orderid);
        if(optionalOrder.isPresent()){
            order = optionalOrder.get();
        }
        return order;
    }

    public List<Order> findAll(){
        List<Order> listOrder = new ArrayList<Order>();
        Iterable<Order> orders = orderRepository.findAll
        ();
        orders.forEach((o) -> {
            listOrder.add(o);
        });
        return listOrder;
    }
}
```

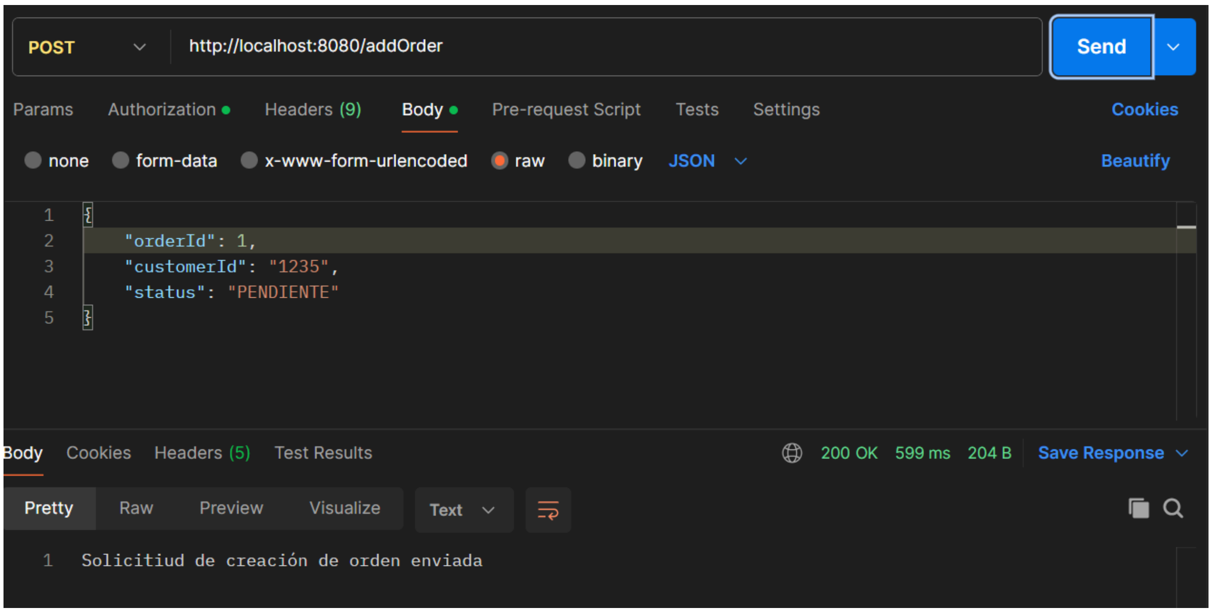
Controller



```java
@RestController
public class OrderController {
    @Autowired
    private OrderEventProducer orderEventProducer;
    private JsonUtils jsonUtils = new JsonUtils();

     @PostMapping("/addOrder")
    public String createOrder(@RequestBody String orderJson) {
        Order order = jsonUtils.fromJson(orderJson, clazz:Order.class);
        orderEventProducer.sendAddOrderEvent(order);
        return "Solicitiud de creación de orden enviada";
    }

     @PostMapping("/editOrder")
    public String updateOrder(@RequestBody String orderJson) {
        Order order = jsonUtils.fromJson(orderJson, clazz:Order.class);
        orderEventProducer.sendEditOrderEvent(order);
        return "Solicitiud de actualización de orden enviada";
    }

    @GetMapping("/order/{orderId}")
    public String findOrder(@PathVariable String orderId) {
        orderEventProducer.sendFindByOrderIDEvent(orderId);
        return "Solicitiud de una orden por id enviada";
    }

    @GetMapping("/orders")
    public String findAllOrders() {
        orderEventProducer.sendFindAllOrdersEvent();
        return "Solicitiud de todas las ordenes enviada";
    }
}
```

Pruebas

- addorder

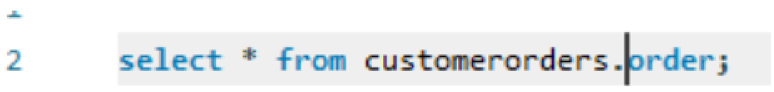


http://localhost:8080/addOrder

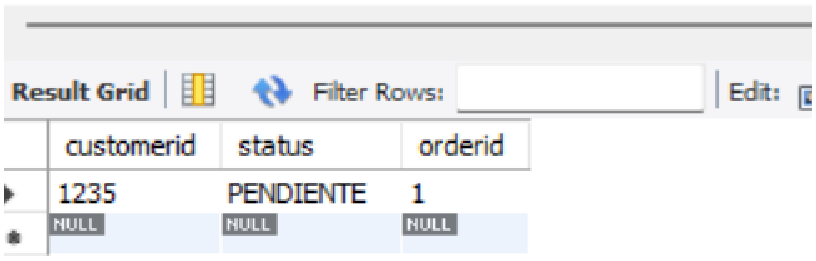
```
{
    "orderId": 1,
    "customerId": "1235",
    "status": "PENDIENTE"
}
```

200 OK 599 ms 204 B

```
Solicitiud de creación de orden enviada
```

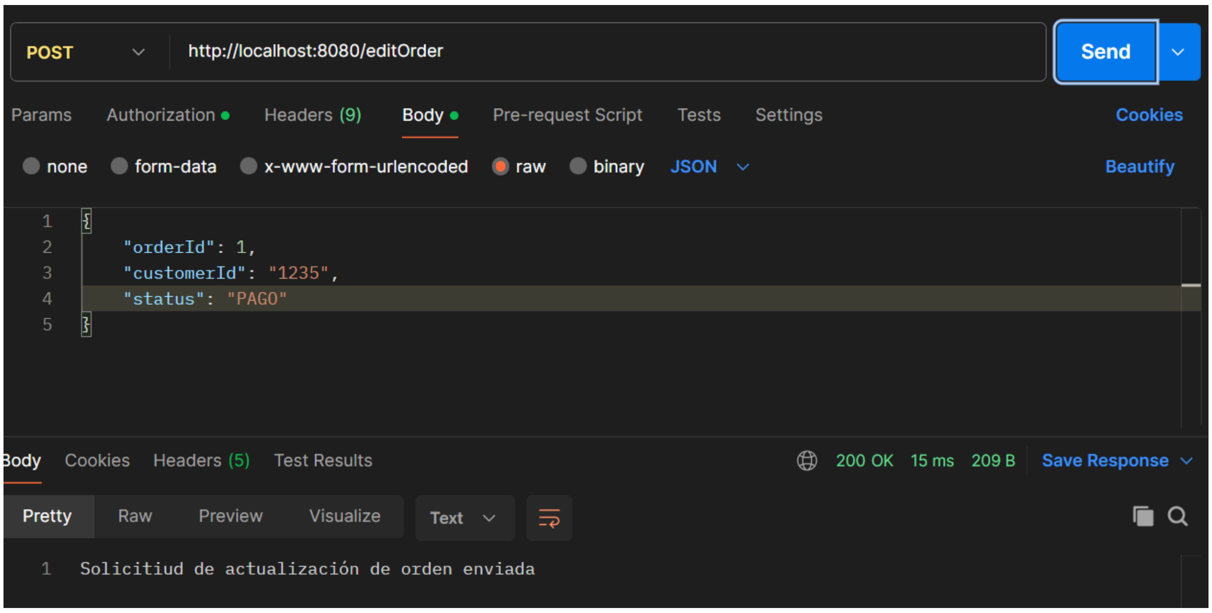


```
select * from customerorders.order;
```



customerid	status	orderid
1235	PENDIENTE	1
NULL	NULL	NULL

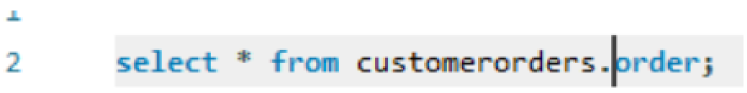
- editOrder



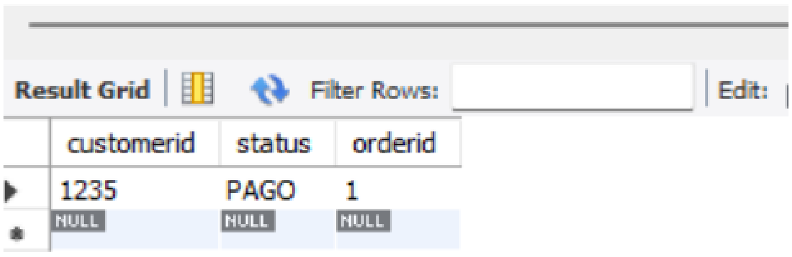
```
http://localhost:8080/editOrder
```

```
{
    "orderId": 1,
    "customerId": "1235",
    "status": "PAGO"
}
```

```
Solicitiud de actualización de orden enviada
```

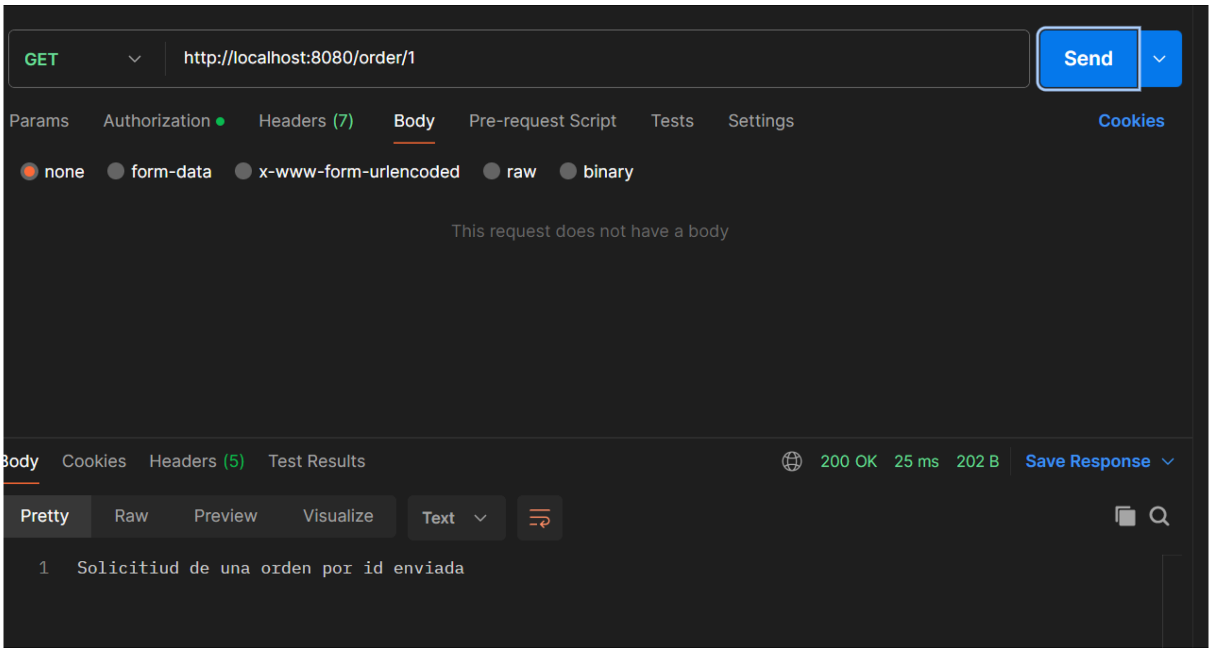


```
select * from customerorders.order;
```



customerid	status	orderid
1235	PAGO	1
NULL	NULL	NULL

- Buscar por id

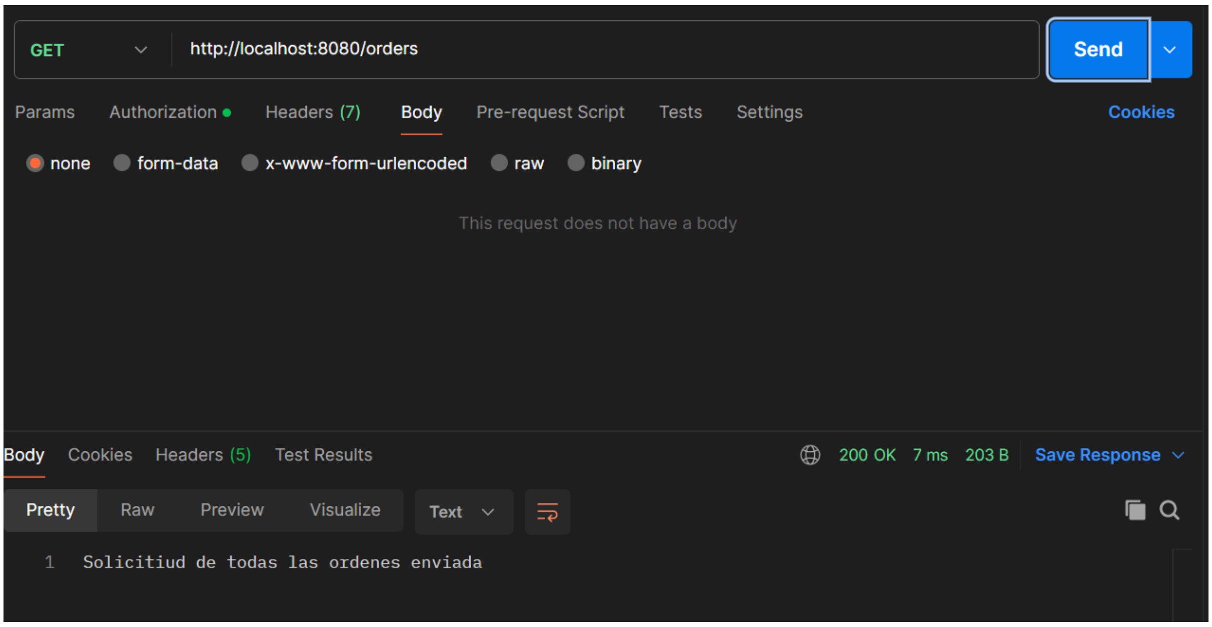


http://localhost:8080/order/1

200 OK 25 ms 202 B

```
Solicitiud de una orden por id enviada
```

- Buscar todas las ordenes



http://localhost:8080/orders

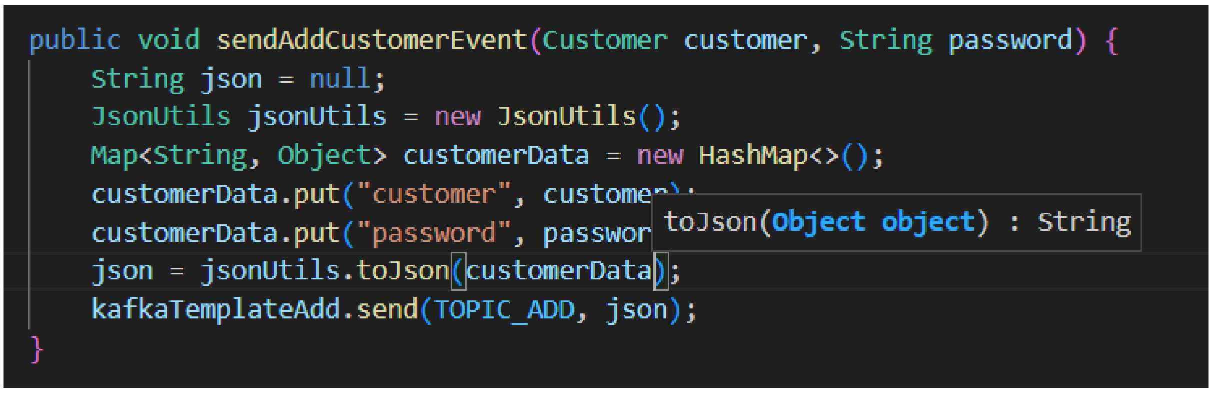
200 OK 7 ms 203 B

```
Solicitiud de todas las ordenes enviada
```

Punto 3

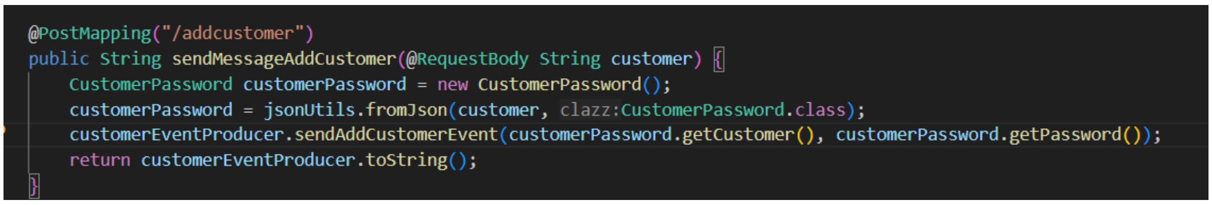
Modifique las clases CustomerEventProducer y CustomerEventConsumer para asegurarse que cuando se cree el cliente se cree el registro de la entidad Login que permita guardar la contraseña.

Primero se editó la clase CustomerEventProducer en el método sendAddCustomerEvent para que reciba la contraseña y así enviar un hashmap el cual contenga el usuario y contraseña



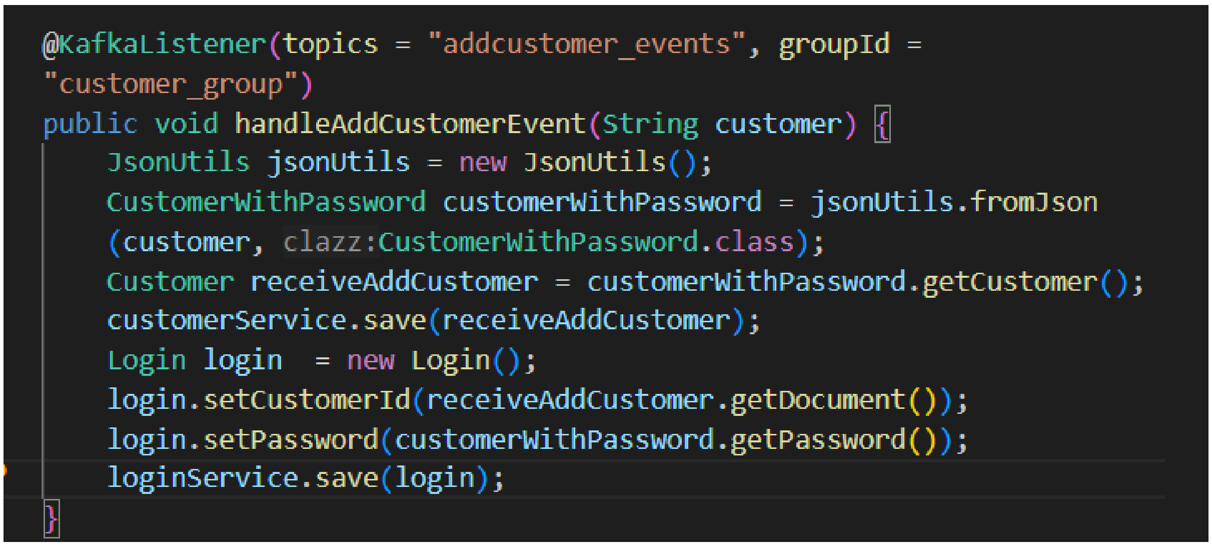
```java
public void sendAddCustomerEvent(Customer customer, String password) {
    String json = null;
    JsonUtils jsonUtils = new JsonUtils();
    Map<String, Object> customerData = new HashMap<>();
    customerData.put("customer", custome
    customerData.put("password", passwor  toJson(Object object) : String
    json = jsonUtils.toJson(customerData);
    kafkaTemplateAdd.send(TOPIC_ADD, json);
}
```

Lo cual generó el hecho de modificar el controller como se evidencia en la siguiente imagen.



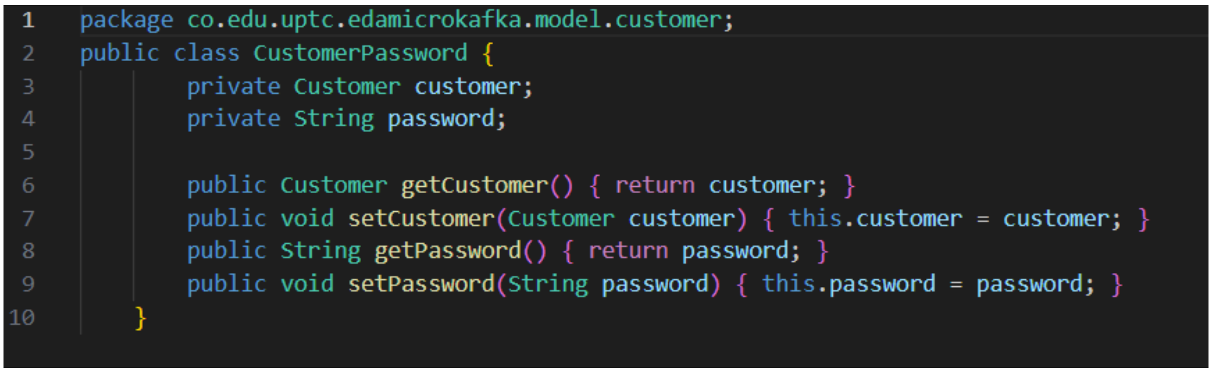
```java
@PostMapping("/addcustomer")
public String sendMessageAddCustomer(@RequestBody String customer) {
    CustomerPassword customerPassword = new CustomerPassword();
    customerPassword = jsonUtils.fromJson(customer, clazz:CustomerPassword.class);
    customerEventProducer.sendAddCustomerEvent(customerPassword.getCustomer(), customerPassword.getPassword());
    return customerEventProducer.toString();
}
```

En cuanto al consumer se modificó para que pueda obtener el customer y la contraseña por medio de un objeto de la clase CustomerPassword pueda obtener.
Luego crear el customer y crear el login con su respectiva contraseña.



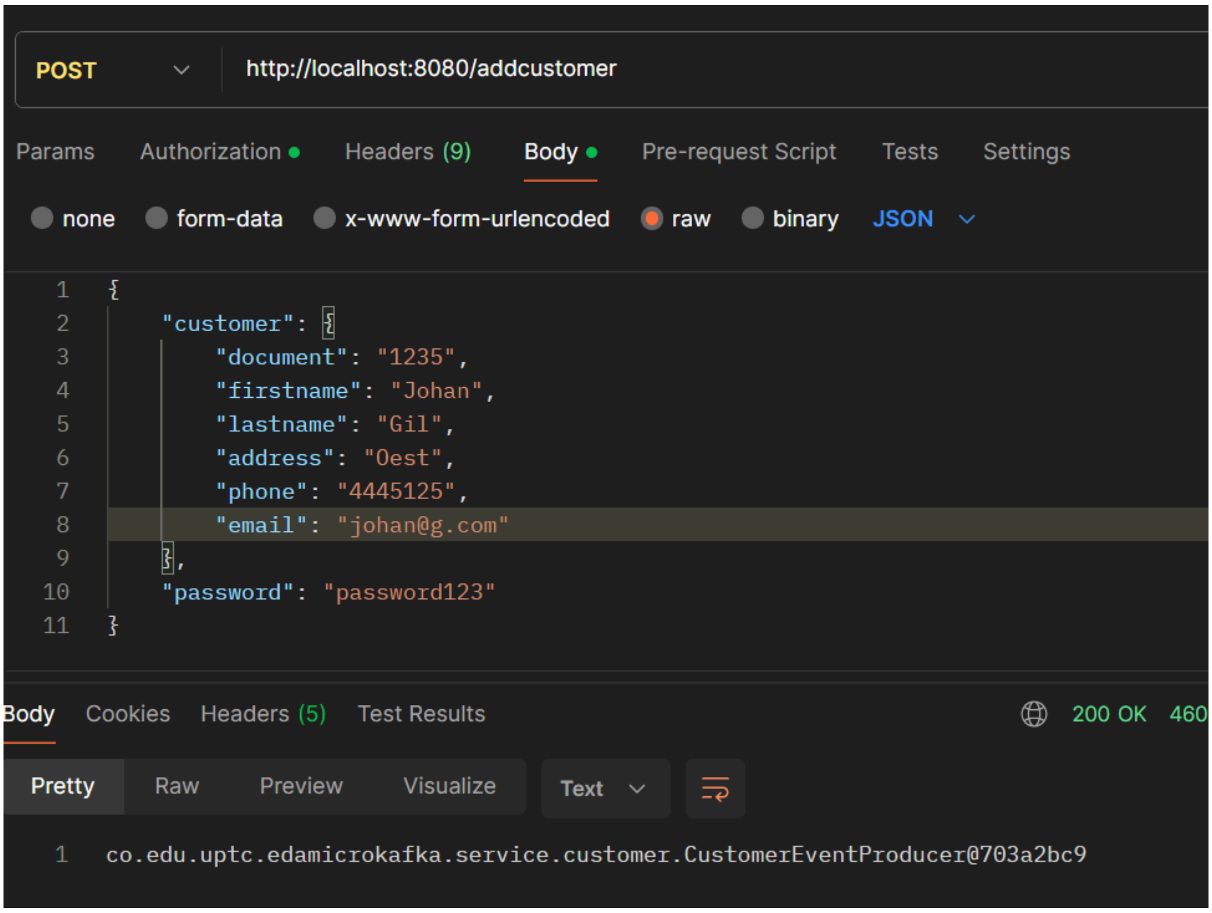
```java
@KafkaListener(topics = "addcustomer_events", groupId =
"customer_group")
public void handleAddCustomerEvent(String customer) {
    JsonUtils jsonUtils = new JsonUtils();
    CustomerWithPassword customerWithPassword = jsonUtils.fromJson
    (customer, clazz:CustomerWithPassword.class);
    Customer receiveAddCustomer = customerWithPassword.getCustomer();
    customerService.save(receiveAddCustomer);
    Login login  = new Login();
    login.setCustomerId(receiveAddCustomer.getDocument());
    login.setPassword(customerWithPassword.getPassword());
    loginService.save(login);
}
```

Donde la clase CustomerPassword es



```java
package co.edu.uptc.edamicrokafka.model.customer;
public class CustomerPassword {
        private Customer customer;
        private String password;

        public Customer getCustomer() { return customer; }
        public void setCustomer(Customer customer) { this.customer = customer; }
        public String getPassword() { return password; }
        public void setPassword(String password) { this.password = password; }
    }
```

Prueba

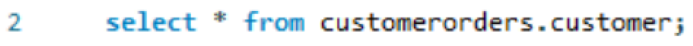


POST http://localhost:8080/addcustomer

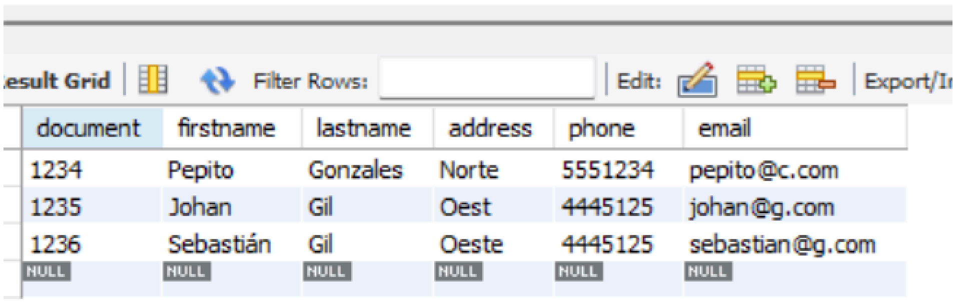
```
{
    "customer": {
        "document": "1235",
        "firstname": "Johan",
        "lastname": "Gil",
        "address": "Oest",
        "phone": "4445125",
        "email": "johan@g.com"
    },
    "password": "password123"
}
```

```
co.edu.uptc.edamicrokafka.service.customer.CustomerEventProducer@703a2bc9
```

Tabla de customers

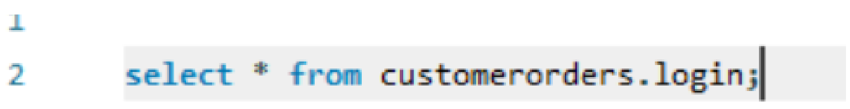


```
select * from customerorders.customer;
```

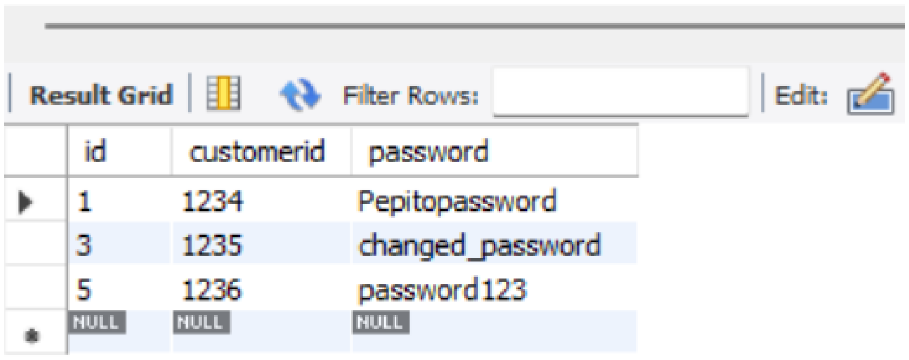


document	firstname	lastname	address	phone	email
1234	Pepito	Gonzales	Norte	5551234	pepito@c.com
1235	Johan	Gil	Oest	4445125	johan@g.com
1236	Sebastián	Gil	Oeste	4445125	sebastian@g.com
NULL	NULL	NULL	NULL	NULL	NULL

Tabla de login



```
select * from customerorders.login;
```



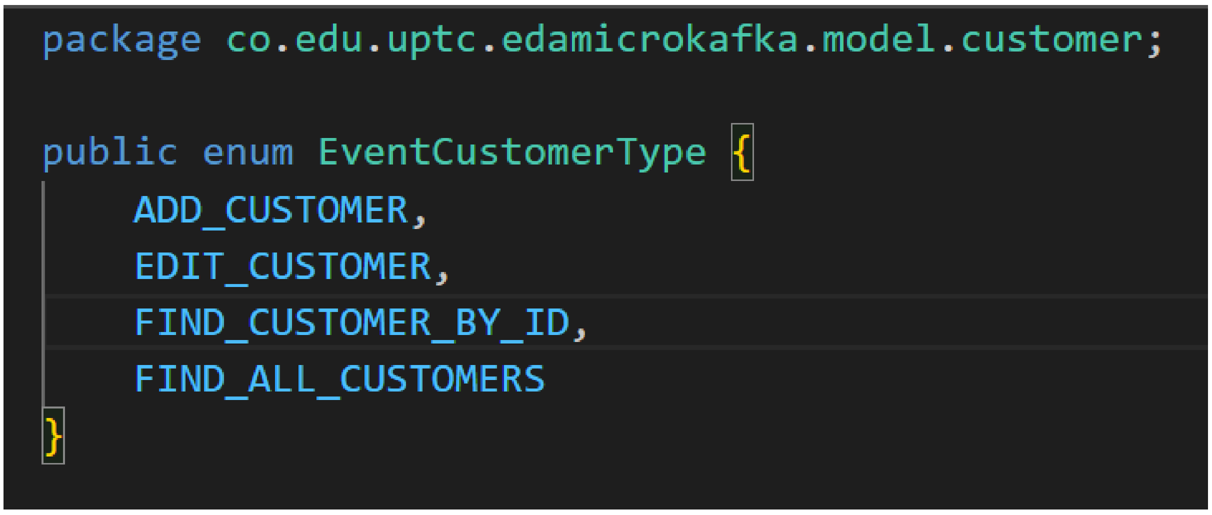
id	customerid	password
1	1234	Pepitopassword
3	1235	changed_password
5	1236	password123
NULL	NULL	NULL

Punto 4

Investigue cómo modificar el código de la clase para que a través de un único tópico por cada entidad de negocio (Customer, Order, Login) pueda manejar todos los eventos de los mismos (Por ejemplo en Customer en un solo tópico manejar addCustomer, editCustomer, findByCustomerID, findAllCustomers).

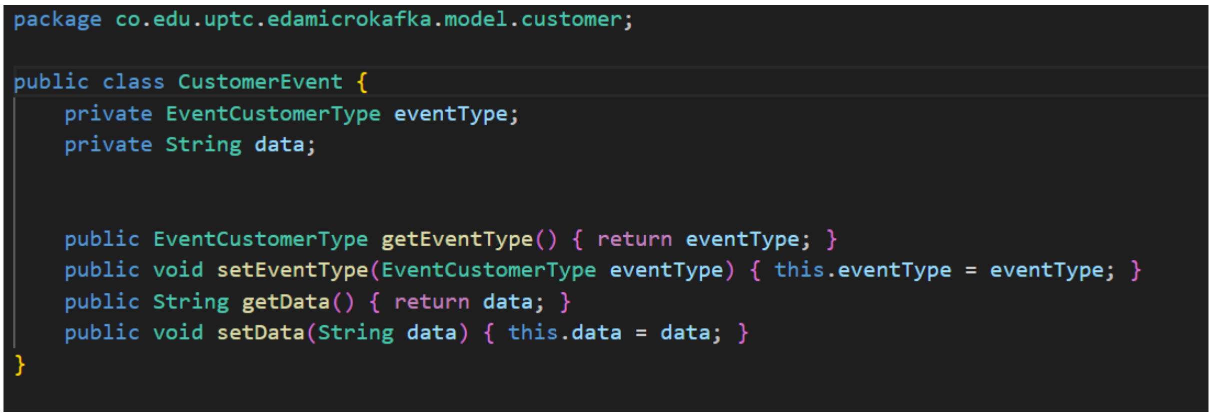
Customer

Primero se creó un conjunto de enumerados para poder manejar los distintos tipos del topic que inicialmente se manejaban como topics.



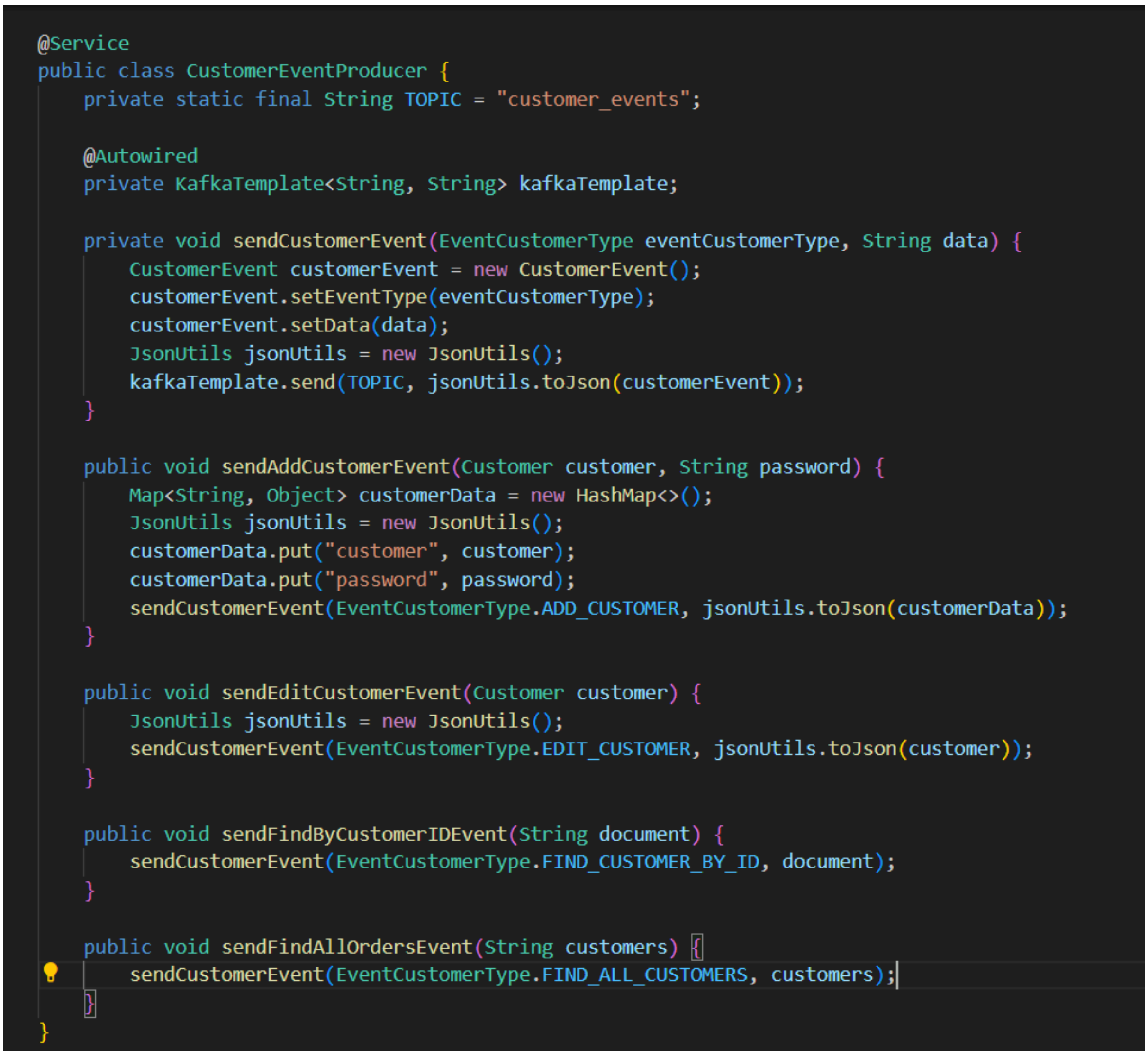
```java
package co.edu.uptc.edamicrokafka.model.customer;

public enum EventCustomerType {
    ADD_CUSTOMER,
    EDIT_CUSTOMER,
    FIND_CUSTOMER_BY_ID,
    FIND_ALL_CUSTOMERS
}
```

Posteriormente se creó la clase CustomerEvent en la cual se une la información a enviar y el tipo de evento que se producirá.



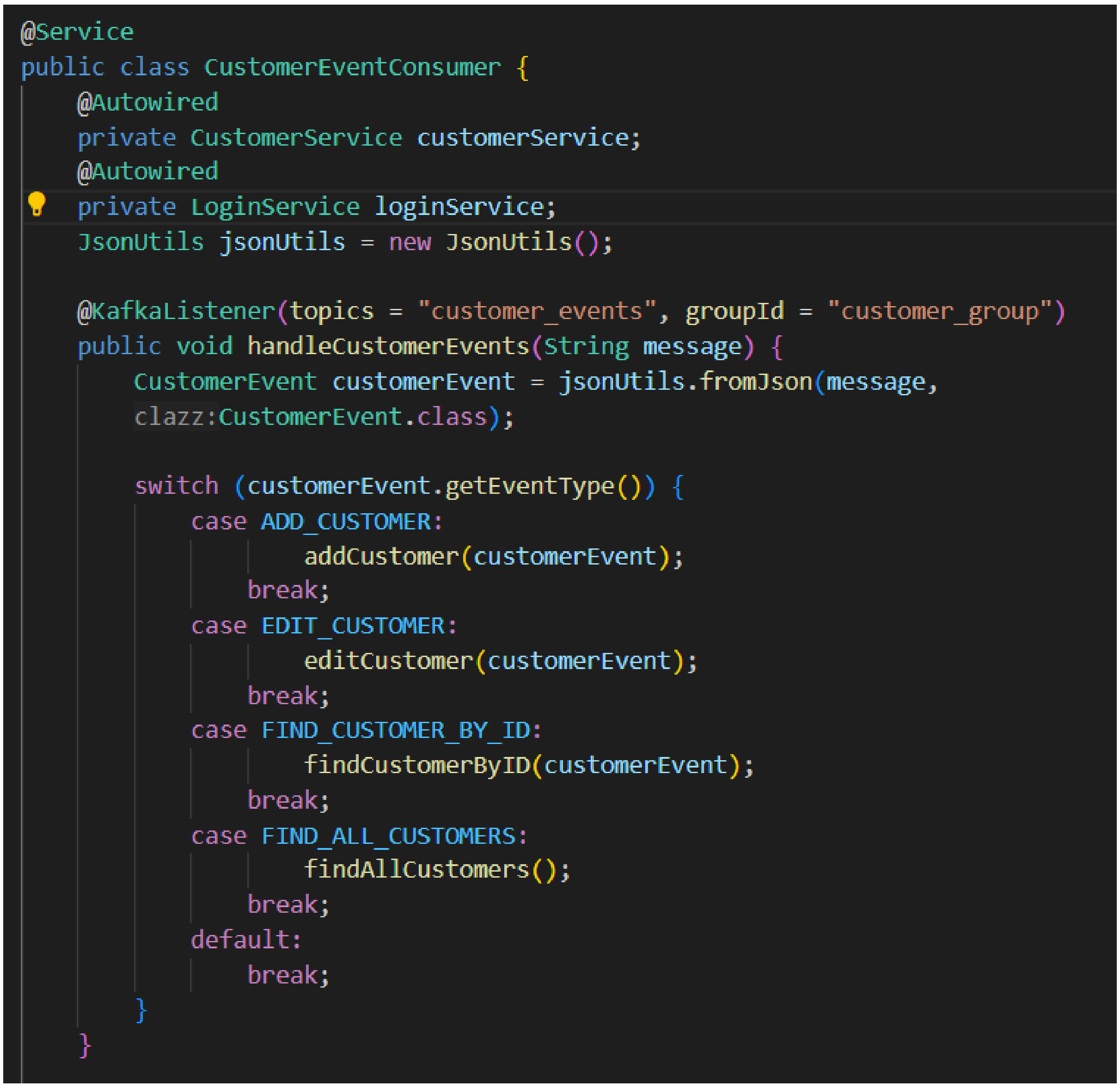
```java
package co.edu.uptc.edamicrokafka.model.customer;

public class CustomerEvent {
    private EventCustomerType eventType;
    private String data;


    public EventCustomerType getEventType() { return eventType; }
    public void setEventType(EventCustomerType eventType) { this.eventType = eventType; }
    public String getData() { return data; }
    public void setData(String data) { this.data = data; }
}
```

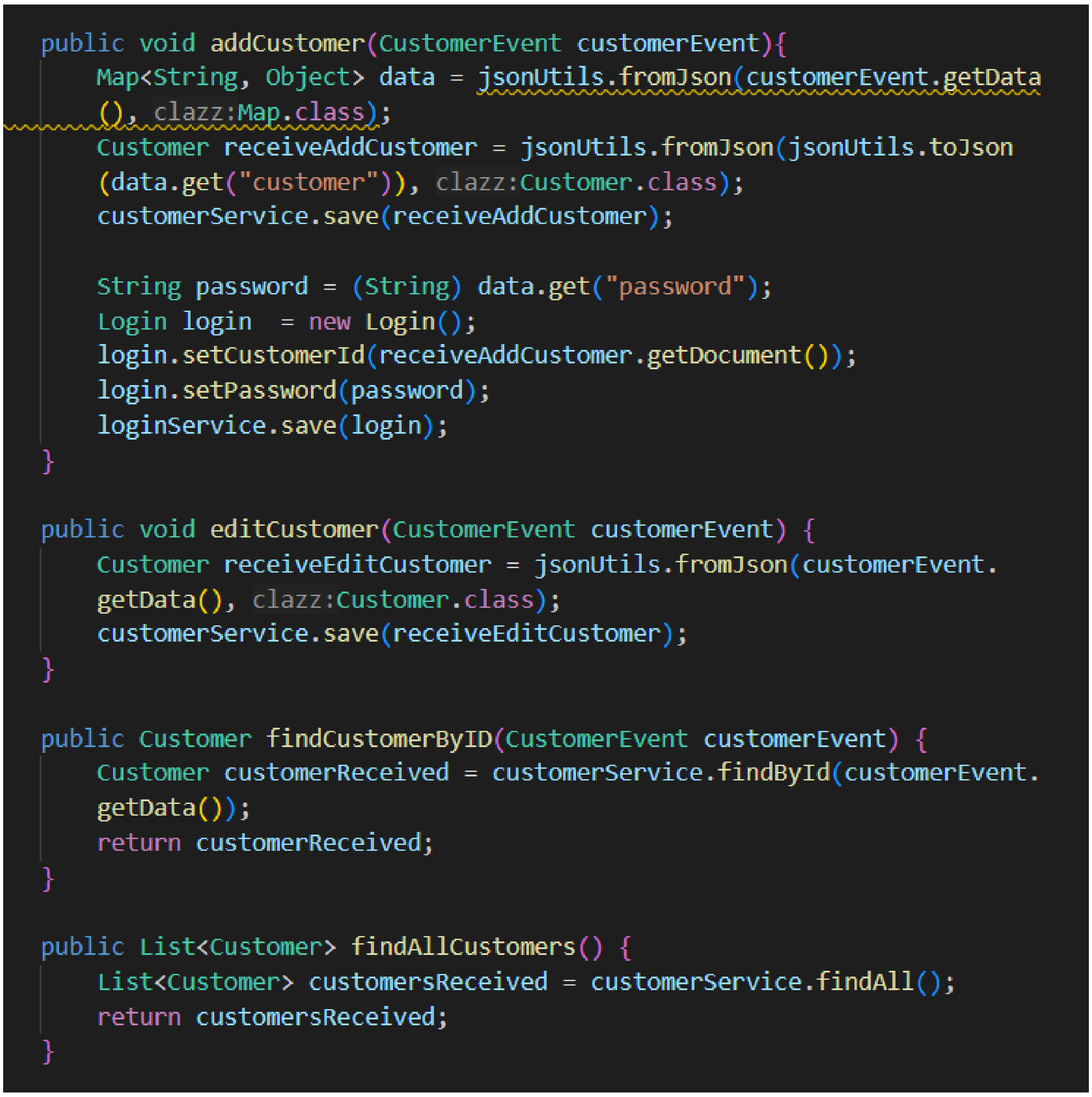
Luego se modificó el producer para manejar solo un topic y que lo enviado fuera un json del objeto CustomerEvent para definir en este el tipo de evento.



```java
@Service
public class CustomerEventProducer {
    private static final String TOPIC = "customer_events";

    @Autowired
    private KafkaTemplate<String, String> kafkaTemplate;

    private void sendCustomerEvent(EventCustomerType eventCustomerType, String data) {
        CustomerEvent customerEvent = new CustomerEvent();
        customerEvent.setEventType(eventCustomerType);
        customerEvent.setData(data);
        JsonUtils jsonUtils = new JsonUtils();
        kafkaTemplate.send(TOPIC, jsonUtils.toJson(customerEvent));
    }

    public void sendAddCustomerEvent(Customer customer, String password) {
        Map<String, Object> customerData = new HashMap<>();
        JsonUtils jsonUtils = new JsonUtils();
        customerData.put("customer", customer);
        customerData.put("password", password);
        sendCustomerEvent(EventCustomerType.ADD_CUSTOMER, jsonUtils.toJson(customerData));
    }

    public void sendEditCustomerEvent(Customer customer) {
        JsonUtils jsonUtils = new JsonUtils();
        sendCustomerEvent(EventCustomerType.EDIT_CUSTOMER, jsonUtils.toJson(customer));
    }

    public void sendFindByCustomerIDEvent(String document) {
        sendCustomerEvent(EventCustomerType.FIND_CUSTOMER_BY_ID, document);
    }

    public void sendFindAllOrdersEvent(String customers) {
        sendCustomerEvent(EventCustomerType.FIND_ALL_CUSTOMERS, customers);
    }
}
```

Posteriormente se modificó el consumer para que por medio de un switch-case pueda definir que hacer de acuerdo al tipo de evento enviado

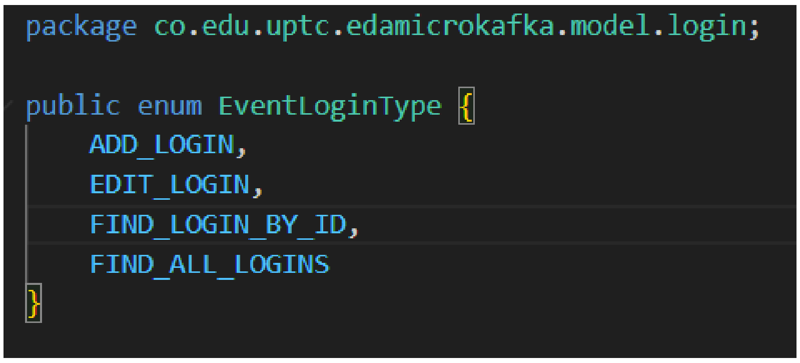

```java
@Service
public class CustomerEventConsumer {
    @Autowired
    private CustomerService customerService;
    @Autowired
    private LoginService loginService;
    JsonUtils jsonUtils = new JsonUtils();

    @KafkaListener(topics = "customer_events", groupId = "customer_group")
    public void handleCustomerEvents(String message) {
        CustomerEvent customerEvent = jsonUtils.fromJson(message,
        clazz:CustomerEvent.class);

        switch (customerEvent.getEventType()) {
            case ADD_CUSTOMER:
                addCustomer(customerEvent);
                break;
            case EDIT_CUSTOMER:
                editCustomer(customerEvent);
                break;
            case FIND_CUSTOMER_BY_ID:
                findCustomerByID(customerEvent);
                break;
            case FIND_ALL_CUSTOMERS:
                findAllCustomers();
                break;
            default:
                break;
        }
    }
```



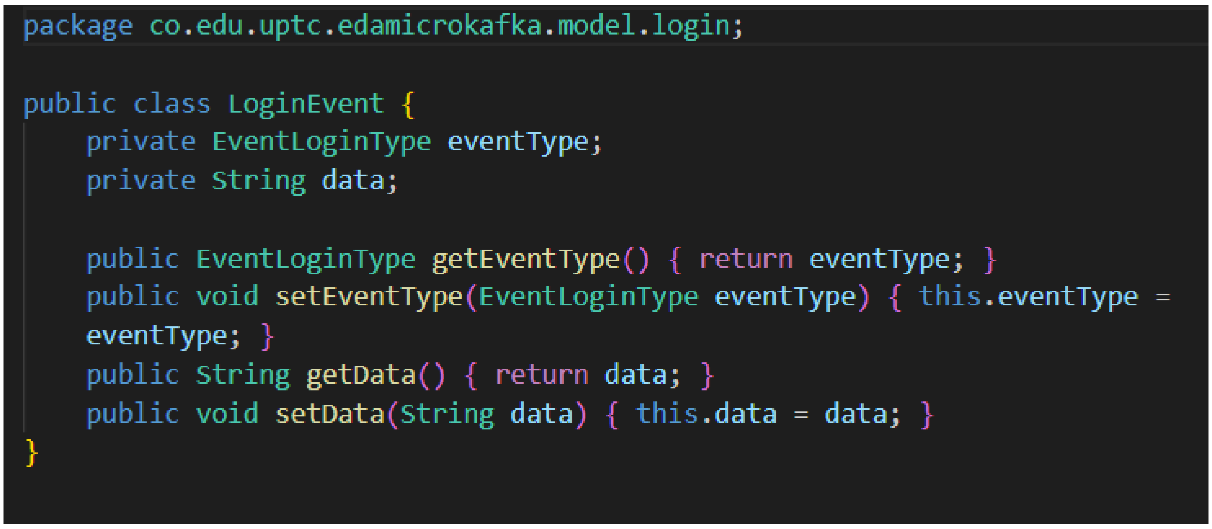
```java
public void addCustomer(CustomerEvent customerEvent){
    Map<String, Object> data = jsonUtils.fromJson(customerEvent.getData
    (), clazz:Map.class);
    Customer receiveAddCustomer = jsonUtils.fromJson(jsonUtils.toJson
    (data.get("customer")), clazz:Customer.class);
    customerService.save(receiveAddCustomer);

    String password = (String) data.get("password");
    Login login  = new Login();
    login.setCustomerId(receiveAddCustomer.getDocument());
    login.setPassword(password);
    loginService.save(login);
}

public void editCustomer(CustomerEvent customerEvent) {
    Customer receiveEditCustomer = jsonUtils.fromJson(customerEvent.
    getData(), clazz:Customer.class);
    customerService.save(receiveEditCustomer);
}

public Customer findCustomerByID(CustomerEvent customerEvent) {
    Customer customerReceived = customerService.findById(customerEvent.
    getData());
    return customerReceived;
}

public List<Customer> findAllCustomers() {
    List<Customer> customersReceived = customerService.findAll();
    return customersReceived;
}
```

De la misma manera se manejó para el login y orders

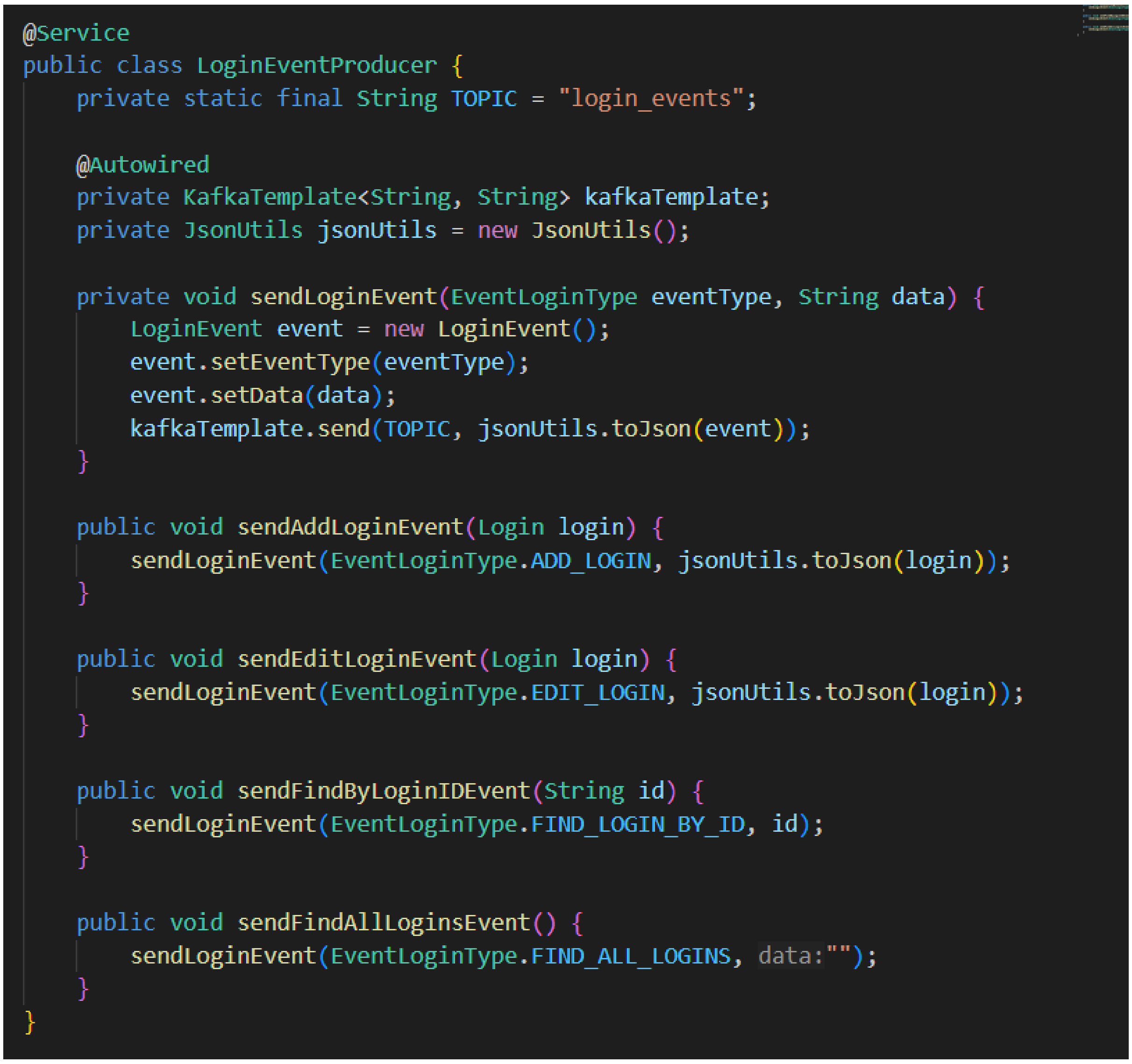
Login



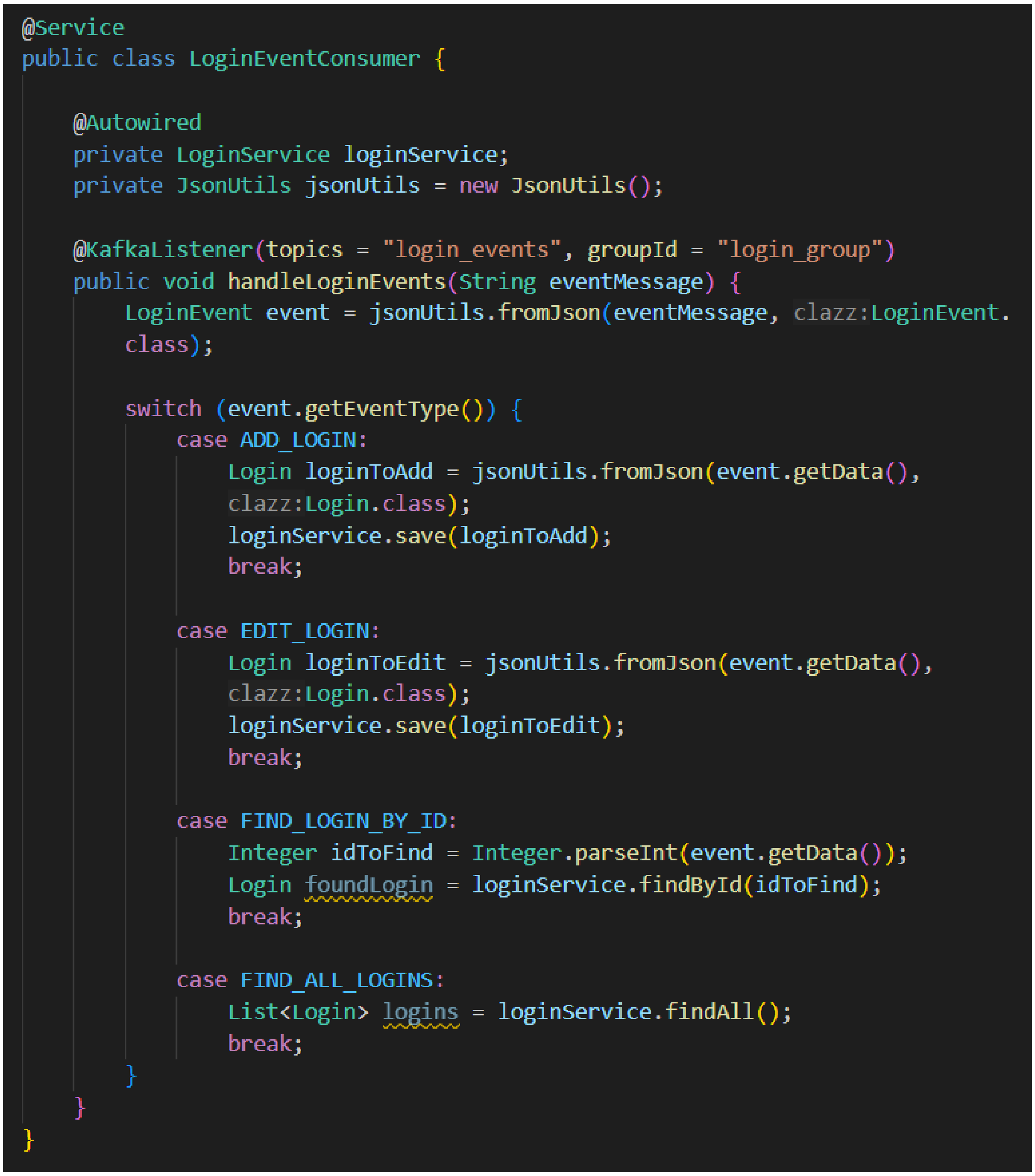
```java
package co.edu.uptc.edamicrokafka.model.login;

public enum EventLoginType {
    ADD_LOGIN,
    EDIT_LOGIN,
    FIND_LOGIN_BY_ID,
    FIND_ALL_LOGINS
}
```



```java
package co.edu.uptc.edamicrokafka.model.login;

public class LoginEvent {
    private EventLoginType eventType;
    private String data;

    public EventLoginType getEventType() { return eventType; }
    public void setEventType(EventLoginType eventType) { this.eventType =
    eventType; }
    public String getData() { return data; }
    public void setData(String data) { this.data = data; }
}
```

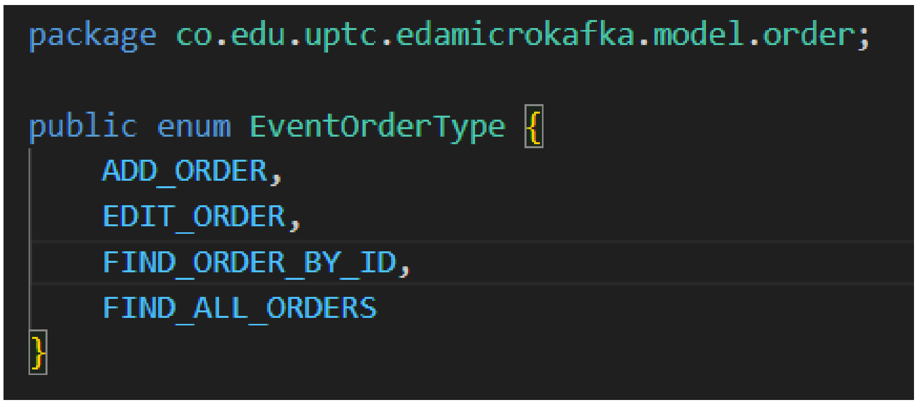


```java
@Service
public class LoginEventProducer {
    private static final String TOPIC = "login_events";

    @Autowired
    private KafkaTemplate<String, String> kafkaTemplate;
    private JsonUtils jsonUtils = new JsonUtils();

    private void sendLoginEvent(EventLoginType eventType, String data) {
        LoginEvent event = new LoginEvent();
        event.setEventType(eventType);
        event.setData(data);
        kafkaTemplate.send(TOPIC, jsonUtils.toJson(event));
    }

    public void sendAddLoginEvent(Login login) {
        sendLoginEvent(EventLoginType.ADD_LOGIN, jsonUtils.toJson(login));
    }

    public void sendEditLoginEvent(Login login) {
        sendLoginEvent(EventLoginType.EDIT_LOGIN, jsonUtils.toJson(login));
    }

    public void sendFindByLoginIDEvent(String id) {
        sendLoginEvent(EventLoginType.FIND_LOGIN_BY_ID, id);
    }

    public void sendFindAllLoginsEvent() {
        sendLoginEvent(EventLoginType.FIND_ALL_LOGINS, data:"");
    }
}
```

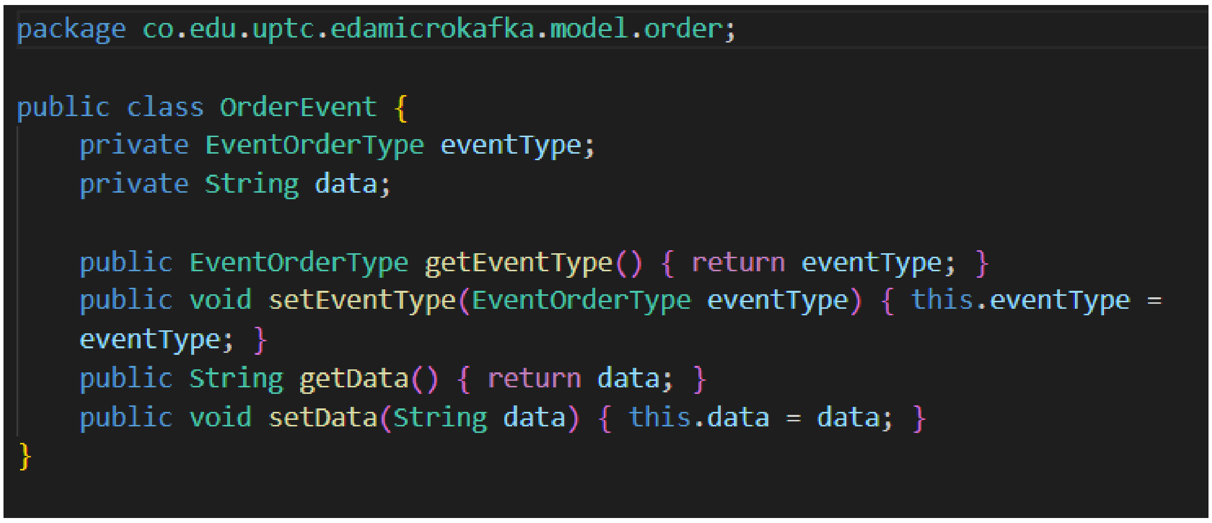


```java
@Service
public class LoginEventConsumer {

    @Autowired
    private LoginService loginService;
    private JsonUtils jsonUtils = new JsonUtils();

    @KafkaListener(topics = "login_events", groupId = "login_group")
    public void handleLoginEvents(String eventMessage) {
        LoginEvent event = jsonUtils.fromJson(eventMessage, clazz:LoginEvent.
        class);

        switch (event.getEventType()) {
            case ADD_LOGIN:
                Login loginToAdd = jsonUtils.fromJson(event.getData(),
                clazz:Login.class);
                loginService.save(loginToAdd);
                break;

            case EDIT_LOGIN:
                Login loginToEdit = jsonUtils.fromJson(event.getData(),
                clazz:Login.class);
                loginService.save(loginToEdit);
                break;

            case FIND_LOGIN_BY_ID:
                Integer idToFind = Integer.parseInt(event.getData());
                Login foundLogin = loginService.findById(idToFind);
                break;

            case FIND_ALL_LOGINS:
                List<Login> logins = loginService.findAll();
                break;
        }
    }
}
```

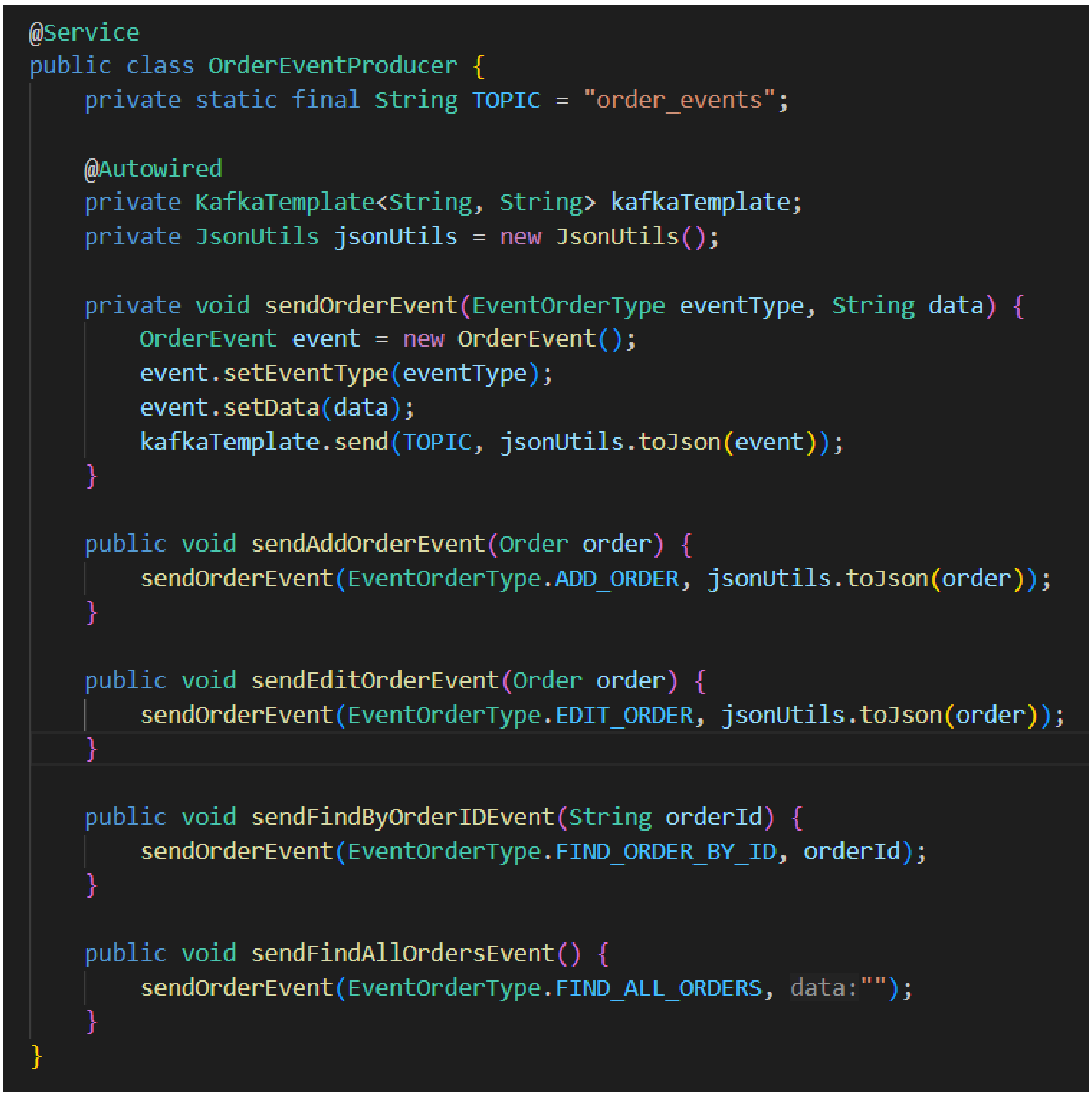
Order



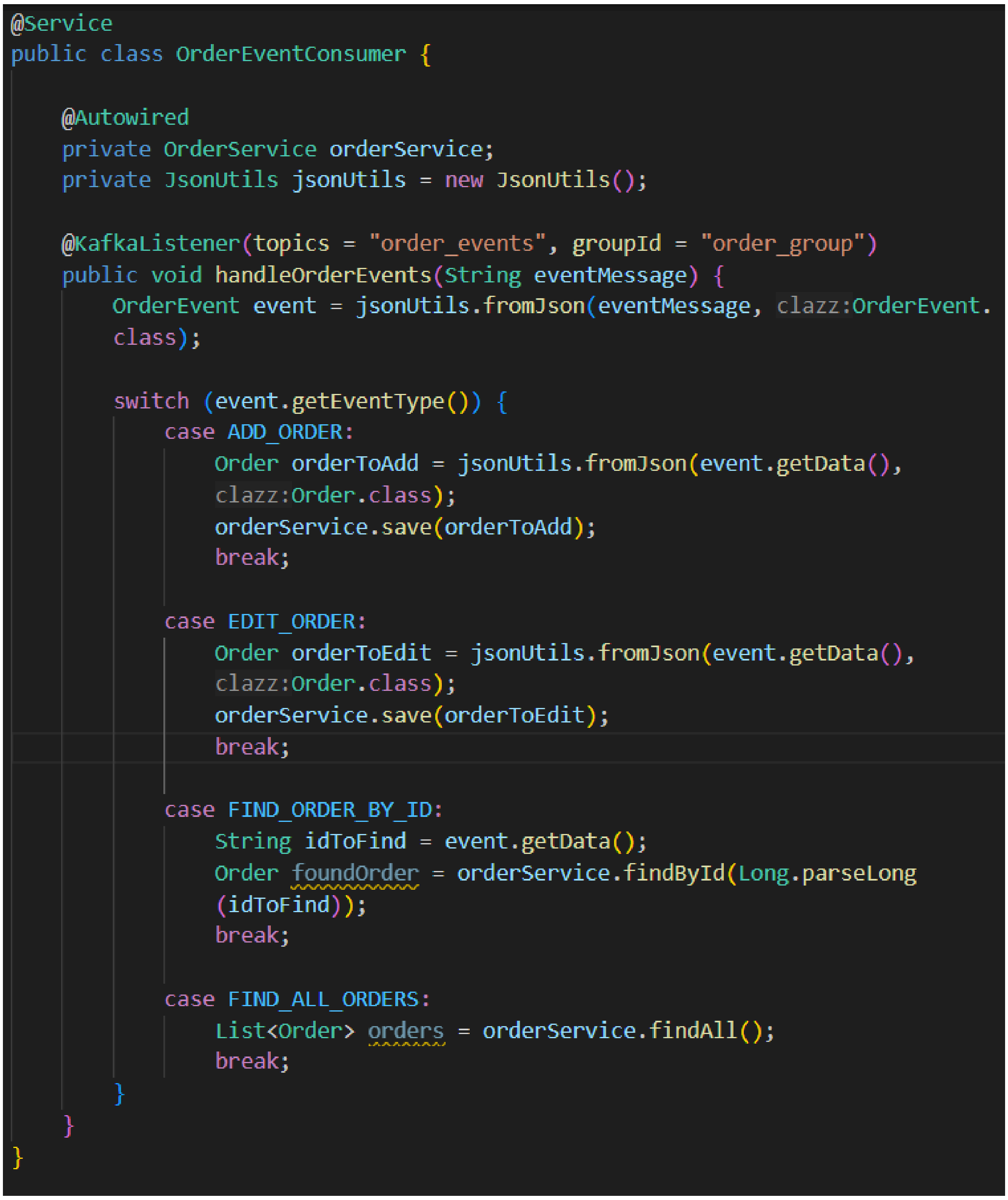
```
package co.edu.uptc.edamicrokafka.model.order;

public enum EventOrderType {
    ADD_ORDER,
    EDIT_ORDER,
    FIND_ORDER_BY_ID,
    FIND_ALL_ORDERS
}
```



```java
package co.edu.uptc.edamicrokafka.model.order;

public class OrderEvent {
    private EventOrderType eventType;
    private String data;

    public EventOrderType getEventType() { return eventType; }
    public void setEventType(EventOrderType eventType) { this.eventType =
    eventType; }
    public String getData() { return data; }
    public void setData(String data) { this.data = data; }
}
```



```java
@Service
public class OrderEventProducer {
    private static final String TOPIC = "order_events";

    @Autowired
    private KafkaTemplate<String, String> kafkaTemplate;
    private JsonUtils jsonUtils = new JsonUtils();

    private void sendOrderEvent(EventOrderType eventType, String data) {
        OrderEvent event = new OrderEvent();
        event.setEventType(eventType);
        event.setData(data);
        kafkaTemplate.send(TOPIC, jsonUtils.toJson(event));
    }

    public void sendAddOrderEvent(Order order) {
        sendOrderEvent(EventOrderType.ADD_ORDER, jsonUtils.toJson(order));
    }

    public void sendEditOrderEvent(Order order) {
        sendOrderEvent(EventOrderType.EDIT_ORDER, jsonUtils.toJson(order));
    }

    public void sendFindByOrderIDEvent(String orderId) {
        sendOrderEvent(EventOrderType.FIND_ORDER_BY_ID, orderId);
    }

    public void sendFindAllOrdersEvent() {
        sendOrderEvent(EventOrderType.FIND_ALL_ORDERS, data:"");
    }
}
```

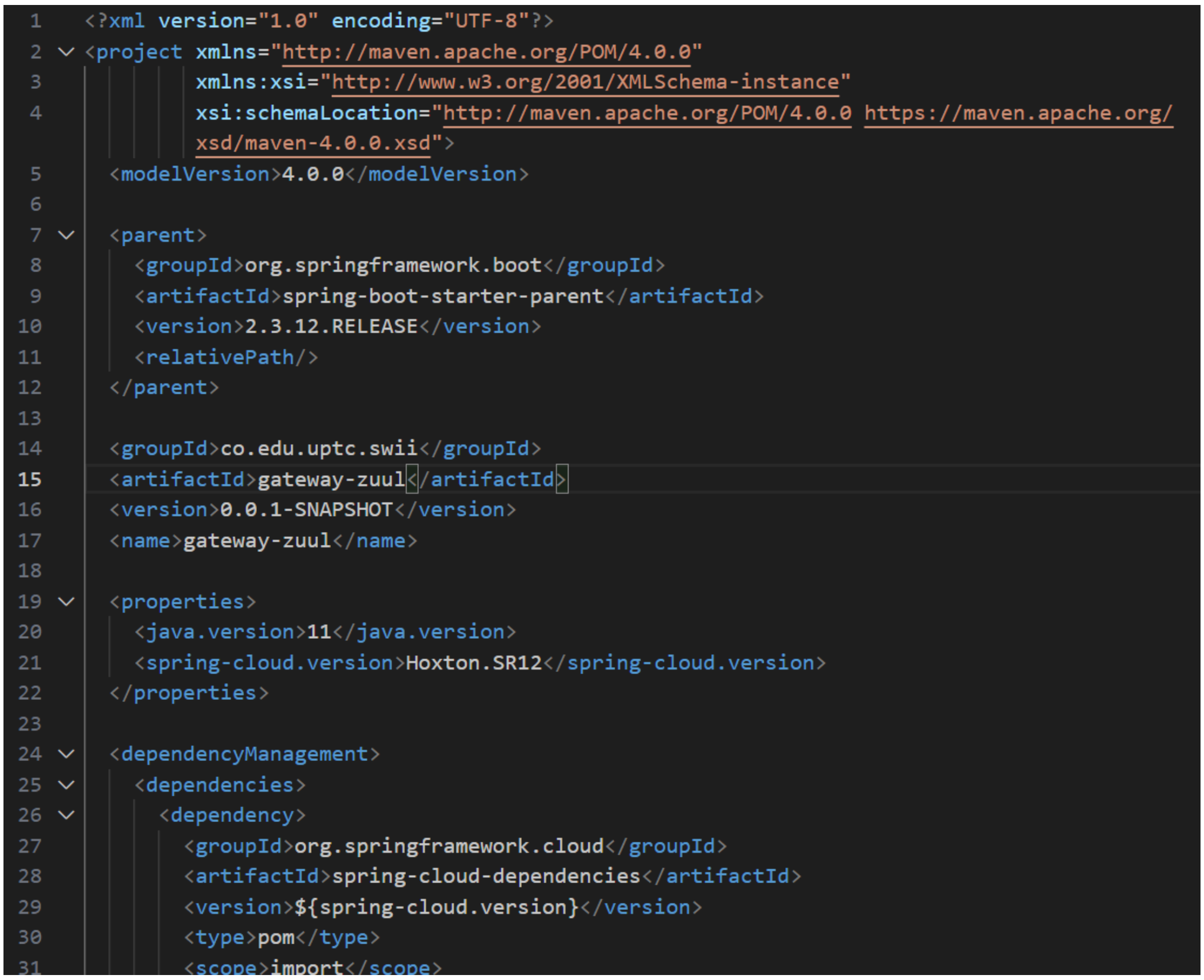


```java
@Service
public class OrderEventConsumer {

    @Autowired
    private OrderService orderService;
    private JsonUtils jsonUtils = new JsonUtils();

    @KafkaListener(topics = "order_events", groupId = "order_group")
    public void handleOrderEvents(String eventMessage) {
        OrderEvent event = jsonUtils.fromJson(eventMessage, clazz:OrderEvent.
        class);

        switch (event.getEventType()) {
            case ADD_ORDER:
                Order orderToAdd = jsonUtils.fromJson(event.getData(),
                clazz:Order.class);
                orderService.save(orderToAdd);
                break;

            case EDIT_ORDER:
                Order orderToEdit = jsonUtils.fromJson(event.getData(),
                clazz:Order.class);
                orderService.save(orderToEdit);
                break;

            case FIND_ORDER_BY_ID:
                String idToFind = event.getData();
                Order foundOrder = orderService.findById(Long.parseLong
                (idToFind));
                break;

            case FIND_ALL_ORDERS:
                List<Order> orders = orderService.findAll();
                break;
        }
    }
}
```

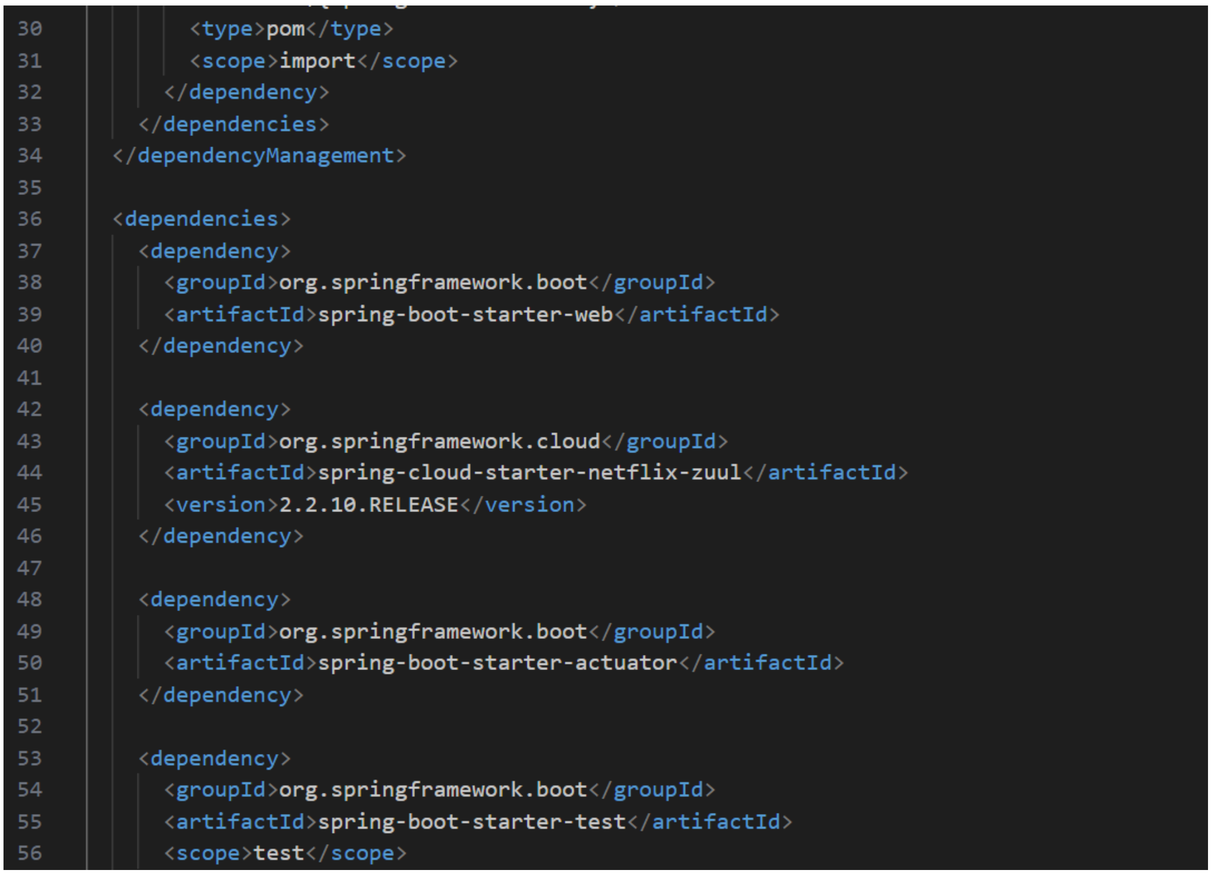
Punto 5

Adicione al proyecto el patrón arquitectónico API Gateway a través de una solución que está en SpringBoot: Netflix Zuul. Instale las dependencias para el proyecto y configúrelo para que sea la puerta única de acceso a sus microservicios.

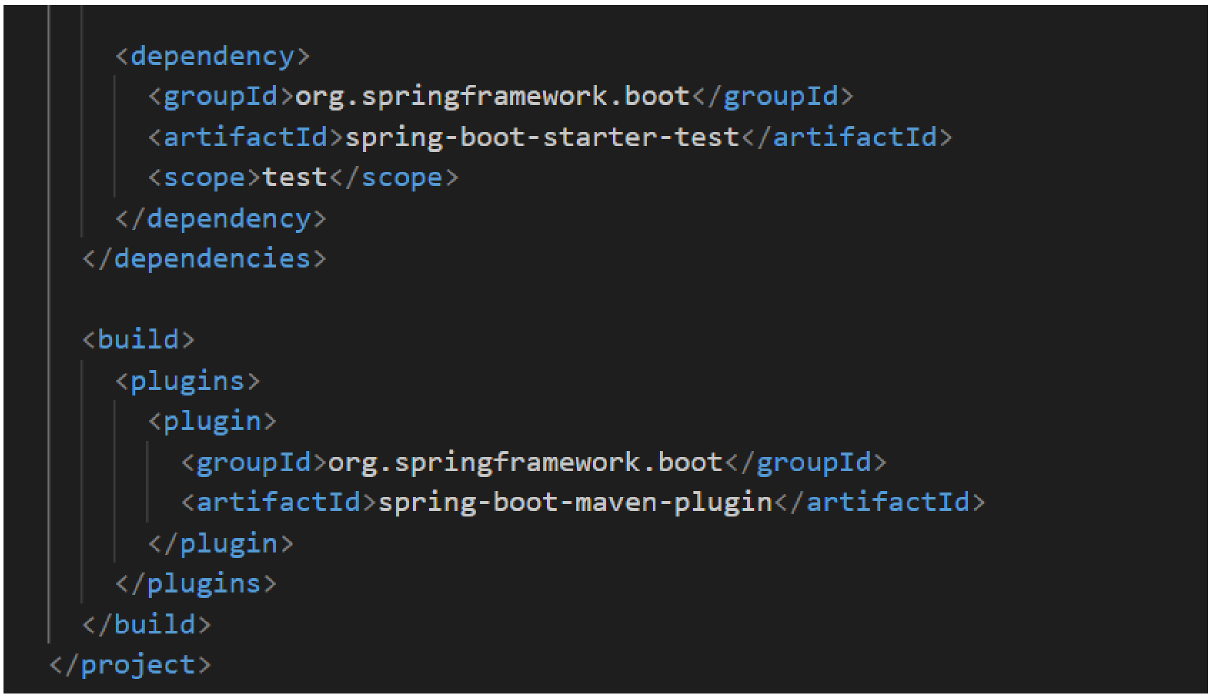
- Primero se debe crear un nuevo proyecto de spring-boot sin ninguna dependencia en sí.
- Se debe configurar el archivo pom de la siguiente forma:



```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>2.3.12.RELEASE</version>
    <relativePath/>
  </parent>

  <groupId>co.edu.uptc.swii</groupId>
  <artifactId>gateway-zuul</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>gateway-zuul</name>

  <properties>
    <java.version>11</java.version>
    <spring-cloud.version>Hoxton.SR12</spring-cloud.version>
  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies</artifactId>
        <version>${spring-cloud.version}</version>
        <type>pom</type>
        <scope>import</scope>
```

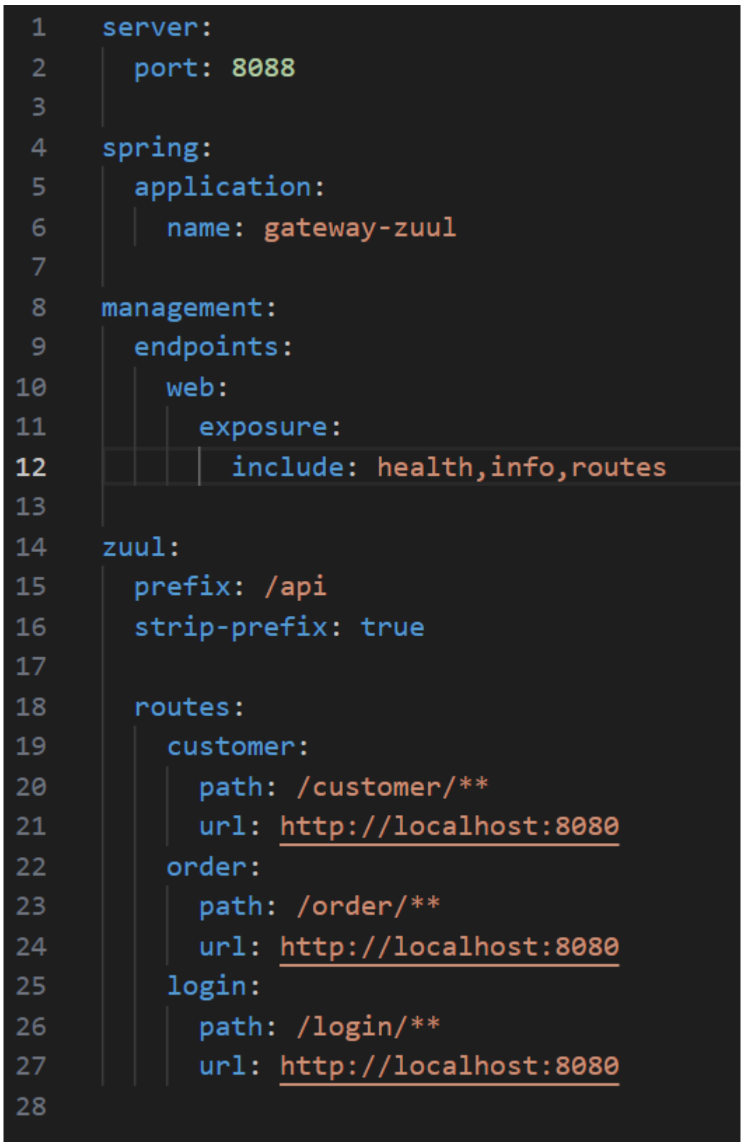


```
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

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

    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
      <version>2.2.10.RELEASE</version>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>

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



```
    <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>
```

- Luego se debe crear en la carpeta resources un archivo llamado application.yml y eliminar el archivo existente llamado application.properties
- Este archivo application.yml debe estar configurado de la siguiente manera:

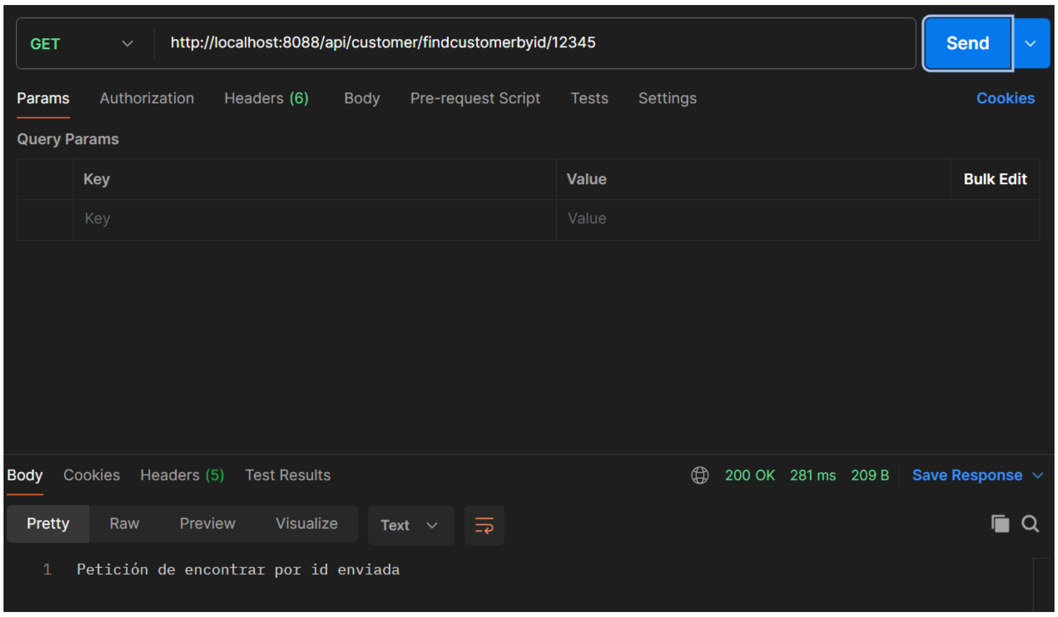


```yaml
server:
  port: 8088

spring:
  application:
    name: gateway-zuul

management:
  endpoints:
    web:
      exposure:
        include: health,info,routes

zuul:
  prefix: /api
  strip-prefix: true

  routes:
    customer:
      path: /customer/**
      url: http://localhost:8080
    order:
      path: /order/**
      url: http://localhost:8080
    login:
      path: /login/**
      url: http://localhost:8080

```

- Ahora ya se puede ejecutar el programa sin ningún problema.

CUSTOMER

- findById


GET http://localhost:8088/api/customer/findcustomerbyid/12345

Params | Authorization | Headers (6) | Body | Pre-request Script | Tests | Settings | Cookies

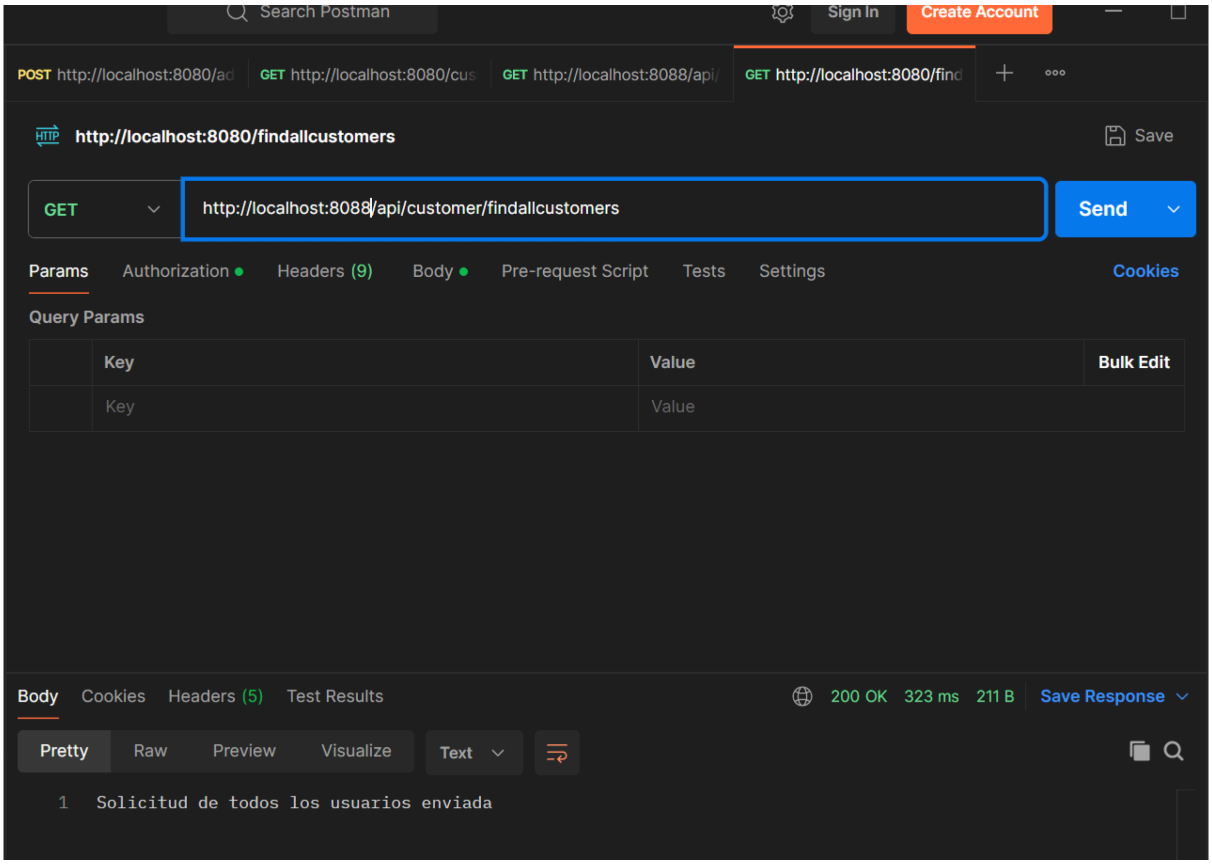
Query Params

Key	Value
Key	Value

Body | Cookies | Headers (5) | Test Results — 200 OK 281 ms 209 B

```
Petición de encontrar por id enviada
```

- findAllCustomers


http://localhost:8080/findallcustomers

GET http://localhost:8088/api/customer/findallcustomers

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings | Cookies

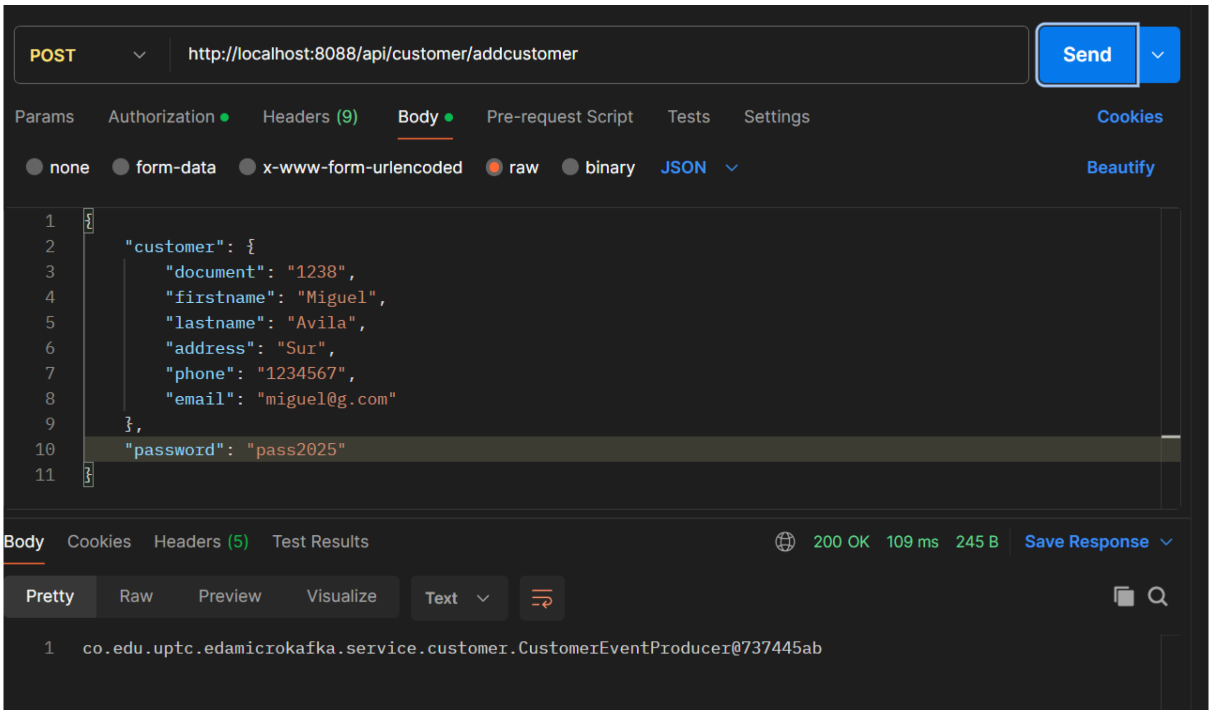
Query Params

Key	Value
Key	Value

Body | Cookies | Headers (5) | Test Results — 200 OK 323 ms 211 B

```
Solicitud de todos los usuarios enviada
```

- addCustomer

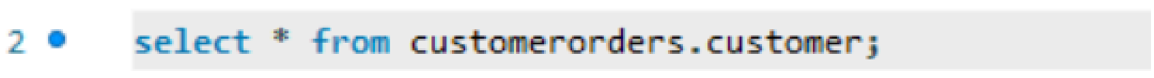


http://localhost:8088/api/customer/addcustomer

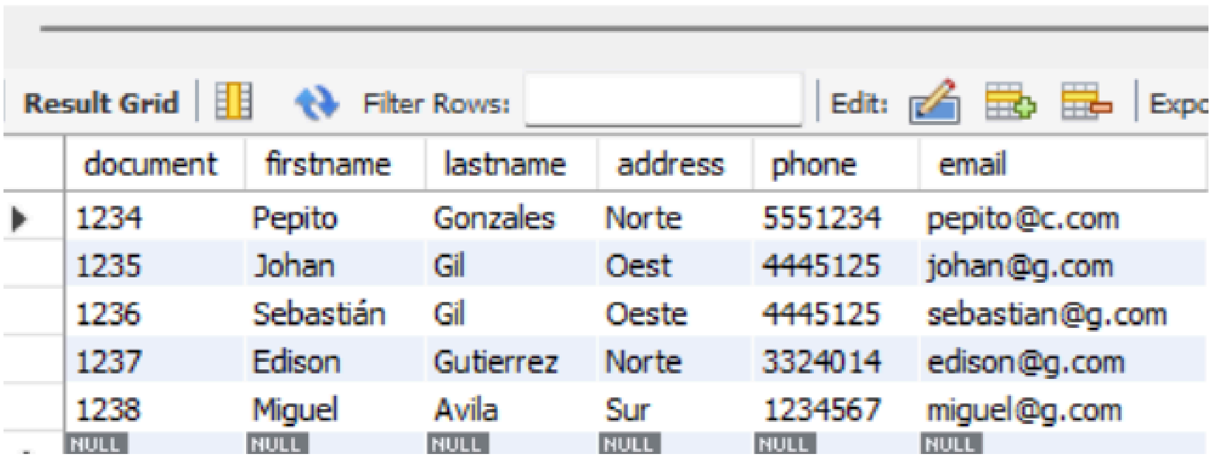
```
{
    "customer": {
        "document": "1238",
        "firstname": "Miguel",
        "lastname": "Avila",
        "address": "Sur",
        "phone": "1234567",
        "email": "miguel@g.com"
    },
    "password": "pass2025"
}
```

200 OK 109 ms 245 B

```
co.edu.uptc.edamicrokafka.service.customer.CustomerEventProducer@737445ab
```



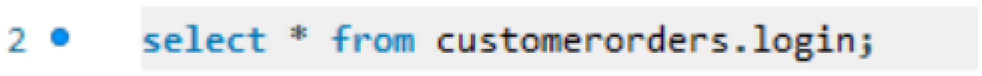
```
2  select * from customerorders.customer;
```



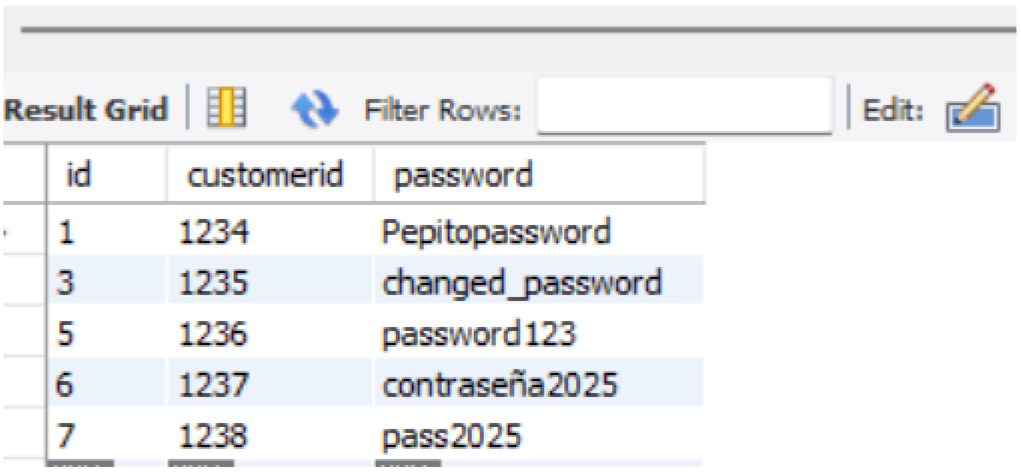
document	firstname	lastname	address	phone	email
1234	Pepito	Gonzales	Norte	5551234	pepito@c.com
1235	Johan	Gil	Oest	4445125	johan@g.com
1236	Sebastián	Gil	Oeste	4445125	sebastian@g.com
1237	Edison	Gutierrez	Norte	3324014	edison@g.com
1238	Miguel	Avila	Sur	1234567	miguel@g.com
NULL	NULL	NULL	NULL	NULL	NULL

- addLogin en el microservicio Login

SE LOGRA VER QUE SE CREA EL LOGIN AL MOMENTO DE CREAR EL CUSTOMER.



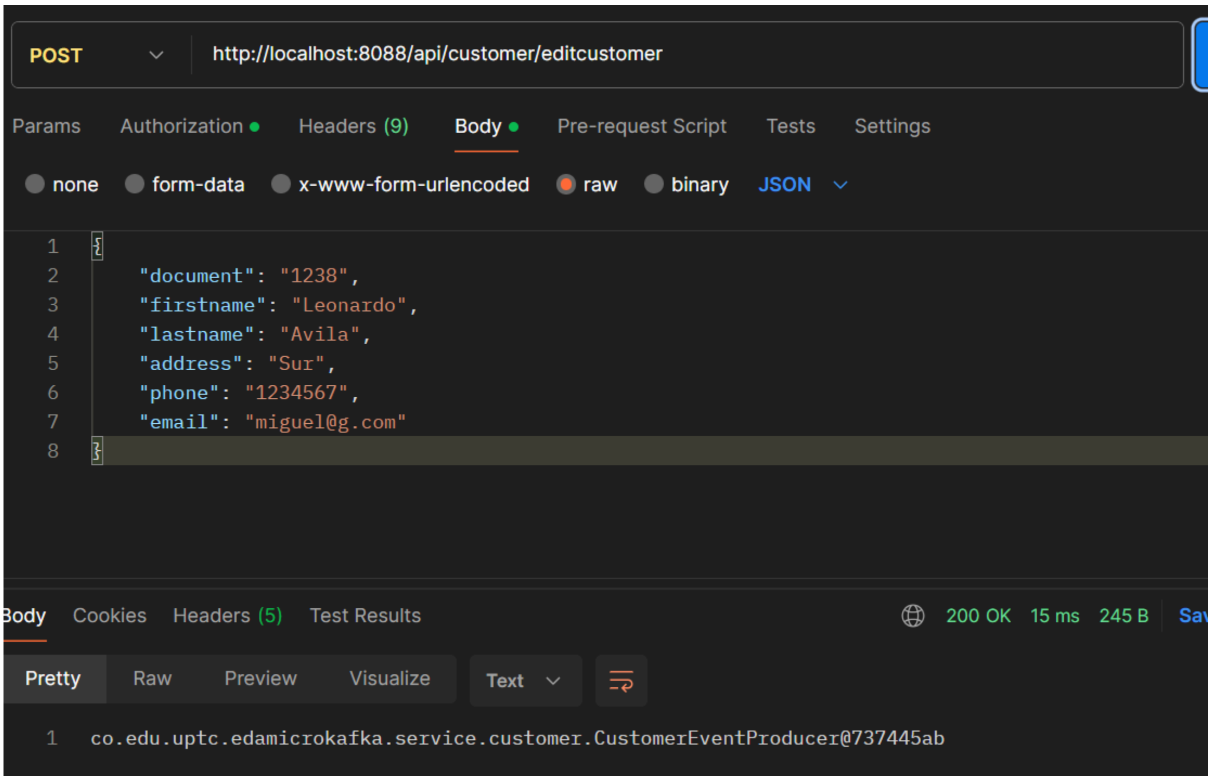
```
2 ●  select * from customerorders.login;
```



Result Grid | Filter Rows: | Edit:

id	customerid	password
1	1234	Pepitopassword
3	1235	changed_password
5	1236	password123
6	1237	contraseña2025
7	1238	pass2025

- editCustomer



POST http://localhost:8088/api/customer/editcustomer

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings

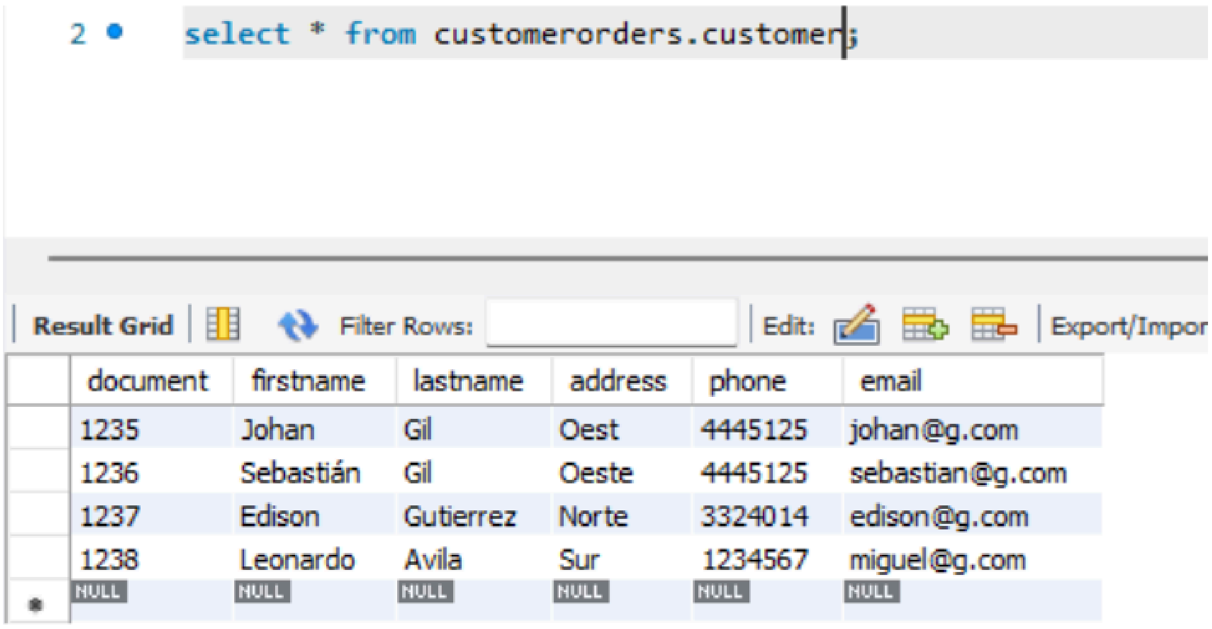
none | form-data | x-www-form-urlencoded | raw | binary | JSON

```
{
    "document": "1238",
    "firstname": "Leonardo",
    "lastname": "Avila",
    "address": "Sur",
    "phone": "1234567",
    "email": "miguel@g.com"
}
```

Body | Cookies | Headers (5) | Test Results — 200 OK 15 ms 245 B

Pretty | Raw | Preview | Visualize | Text

```
co.edu.uptc.edamicrokafka.service.customer.CustomerEventProducer@737445ab
```

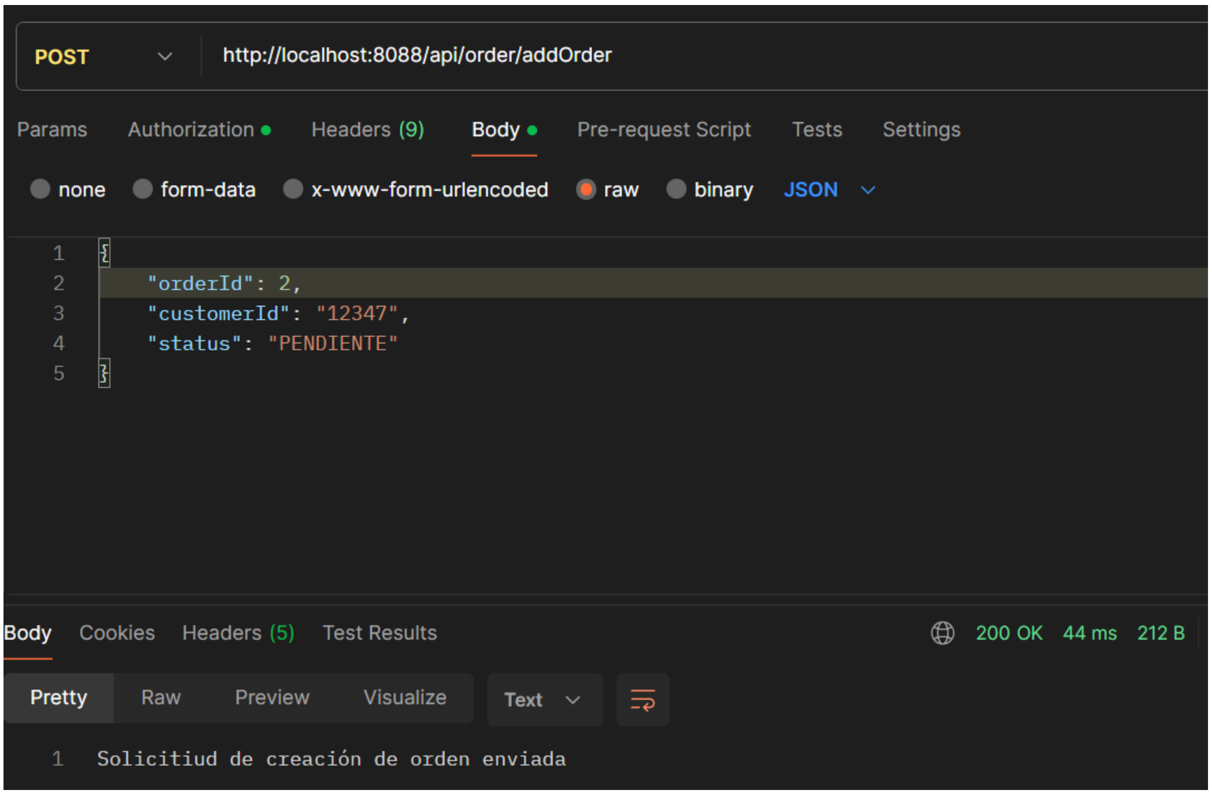


```
select * from customerorders.customer;
```

document	firstname	lastname	address	phone	email
1235	Johan	Gil	Oest	4445125	johan@g.com
1236	Sebastián	Gil	Oeste	4445125	sebastian@g.com
1237	Edison	Gutierrez	Norte	3324014	edison@g.com
1238	Leonardo	Avila	Sur	1234567	miguel@g.com
NULL	NULL	NULL	NULL	NULL	NULL

ORDER:

- **addOrder**

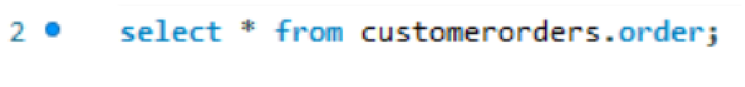


POST http://localhost:8088/api/order/addOrder

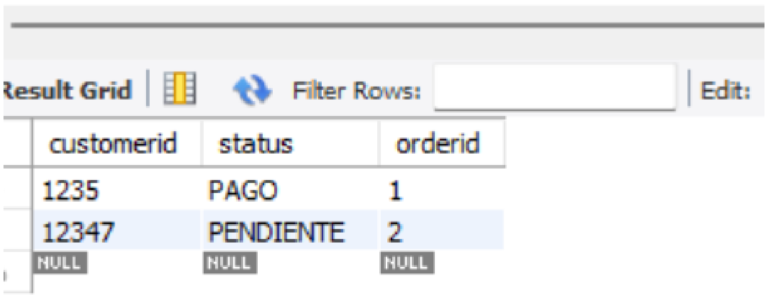
```
{
    "orderId": 2,
    "customerId": "12347",
    "status": "PENDIENTE"
}
```

200 OK 44 ms 212 B

```
Solicitiud de creación de orden enviada
```

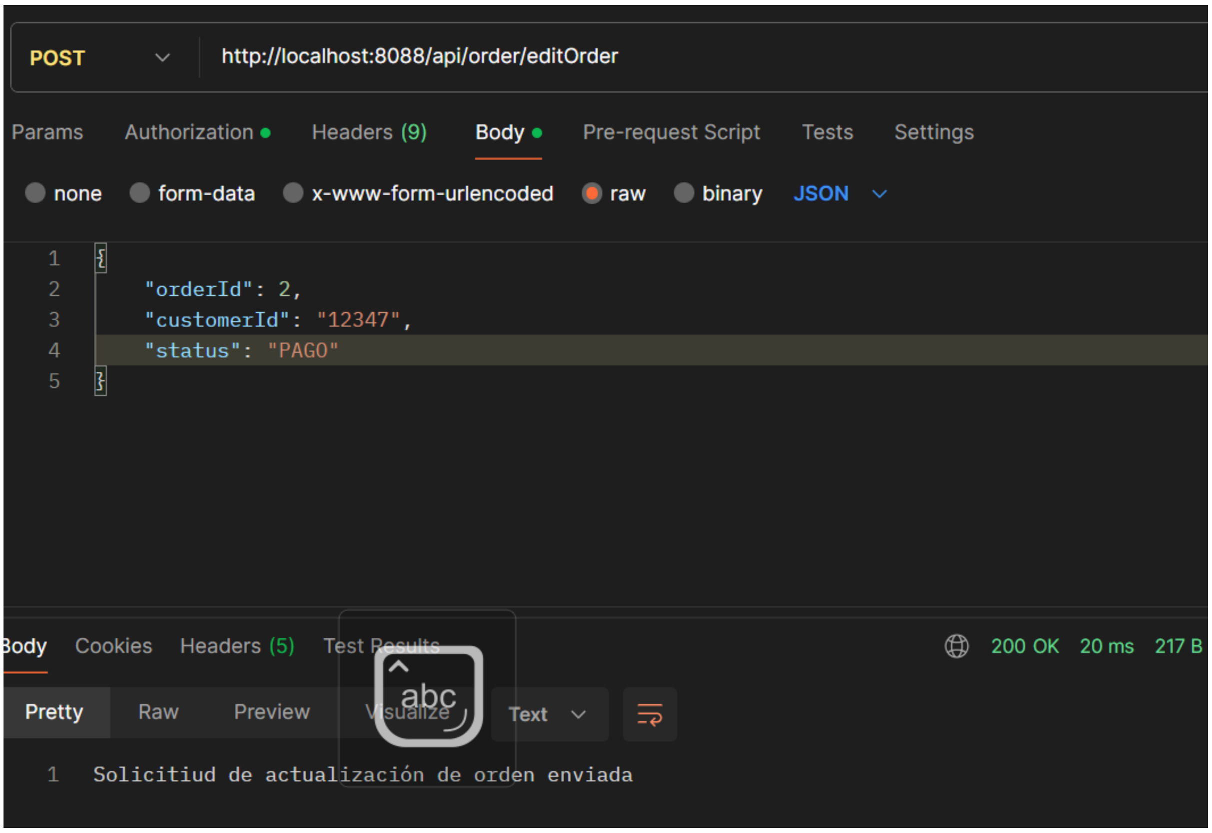


```sql
select * from customerorders.order;
```



customerid	status	orderid
1235	PAGO	1
12347	PENDIENTE	2
NULL	NULL	NULL

- editOrder

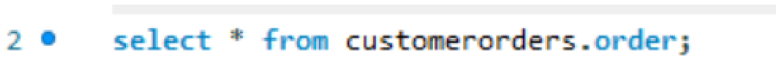


POST http://localhost:8088/api/order/editOrder

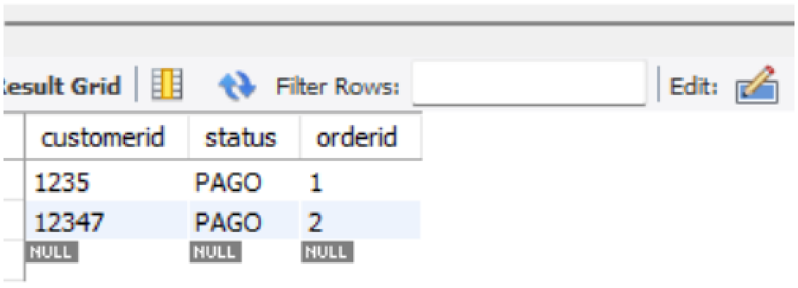
```json
{
    "orderId": 2,
    "customerId": "12347",
    "status": "PAGO"
}
```

200 OK 20 ms 217 B

```
Solicitiud de actualización de orden enviada
```

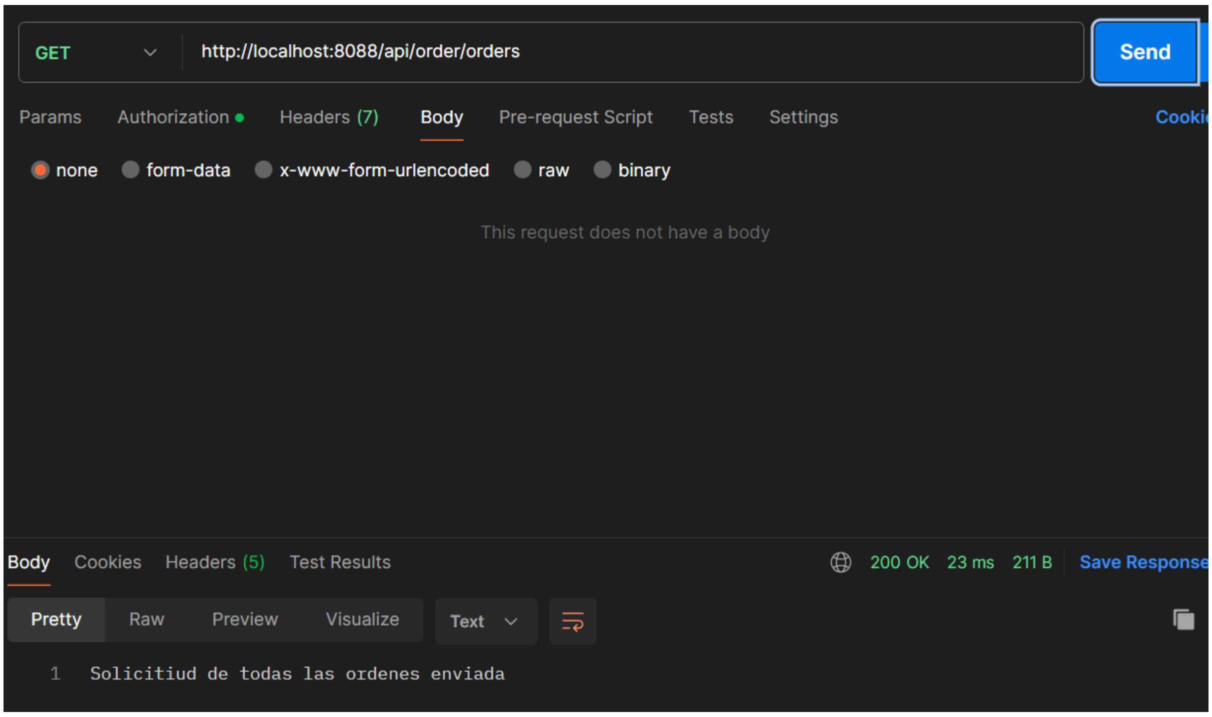


```
select * from customerorders.order;
```



customerid	status	orderid
1235	PAGO	1
12347	PAGO	2
NULL	NULL	NULL

- Obtener todas las ordenes



GET http://localhost:8088/api/order/orders

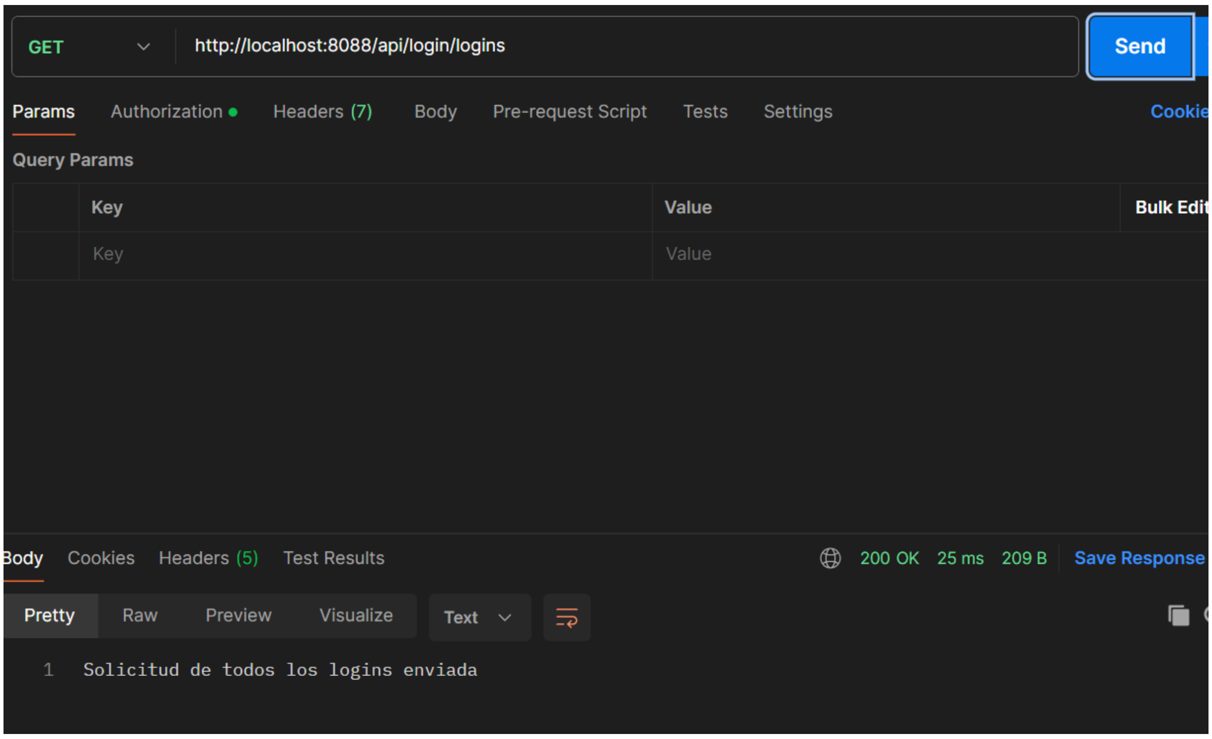
This request does not have a body

200 OK 23 ms 211 B

```
Solicitiud de todas las ordenes enviada
```

LOGIN

- Obtener todos los login:

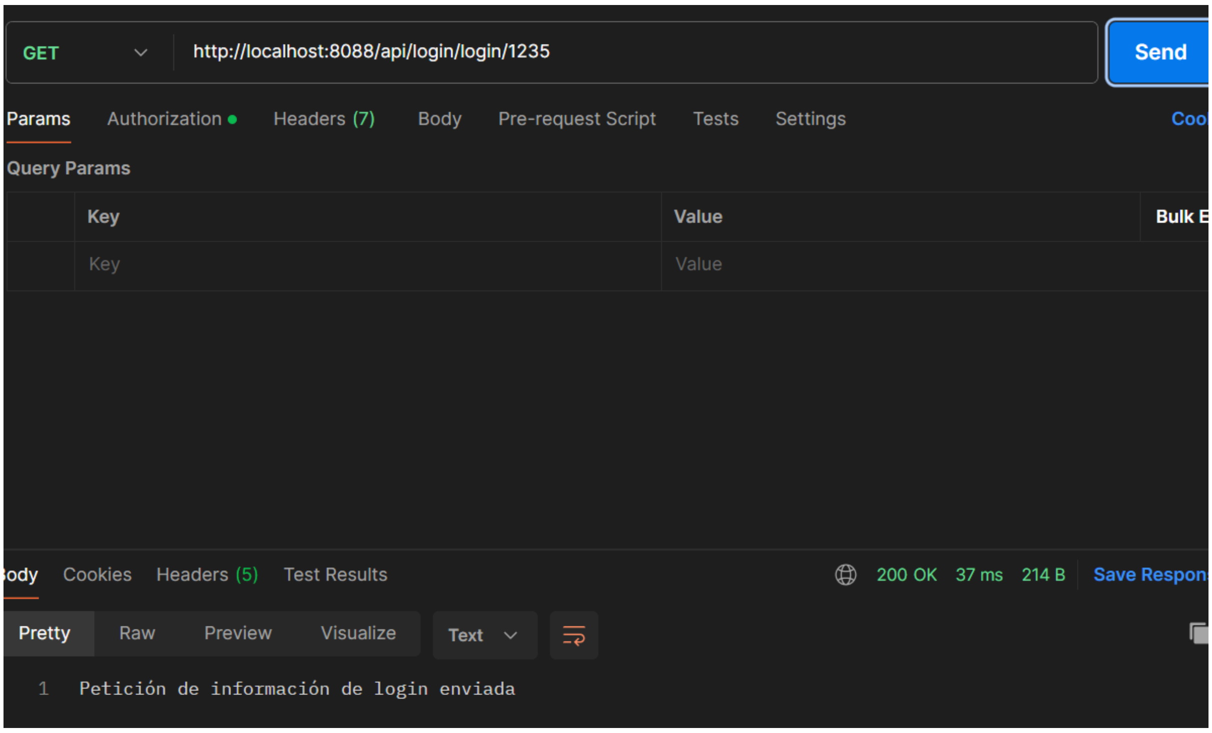


http://localhost:8088/api/login/logins

200 OK 25 ms 209 B

```
Solicitud de todos los logins enviada
```

- Obtener un login específico:

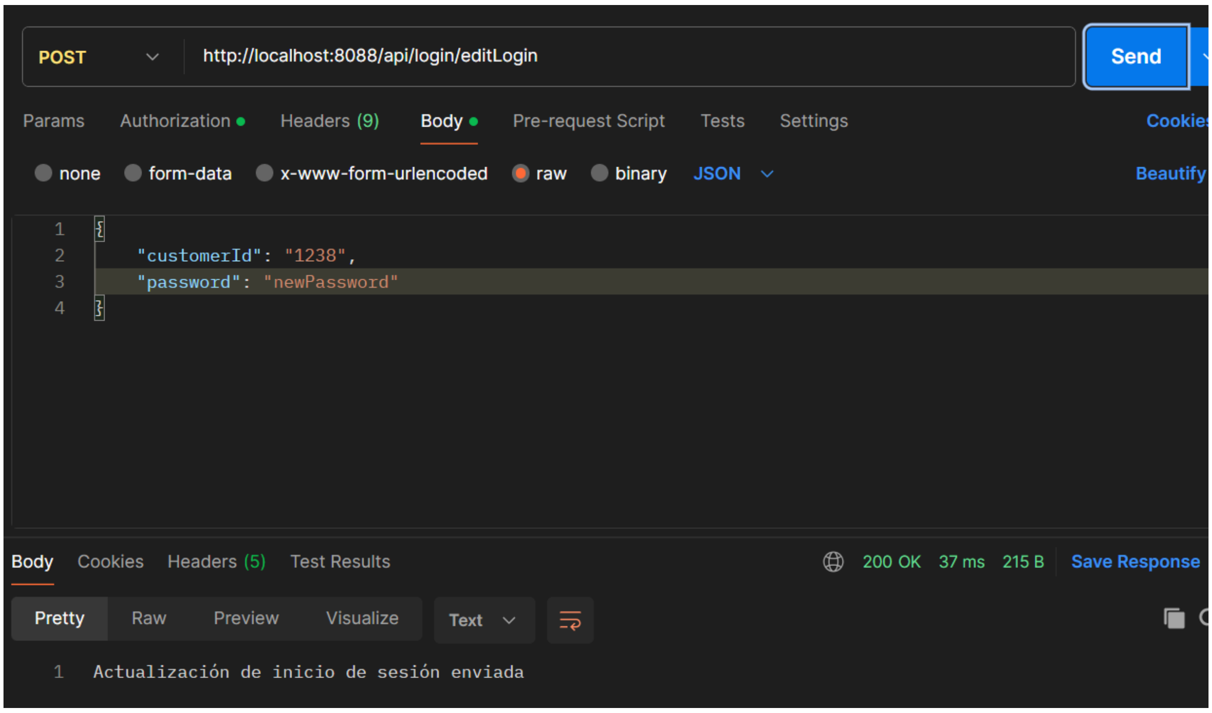


http://localhost:8088/api/login/login/1235

200 OK 37 ms 214 B

```
Petición de información de login enviada
```

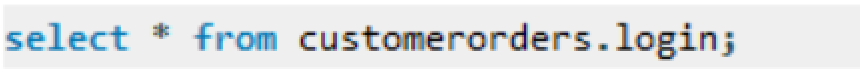
- Editar la información de un login:



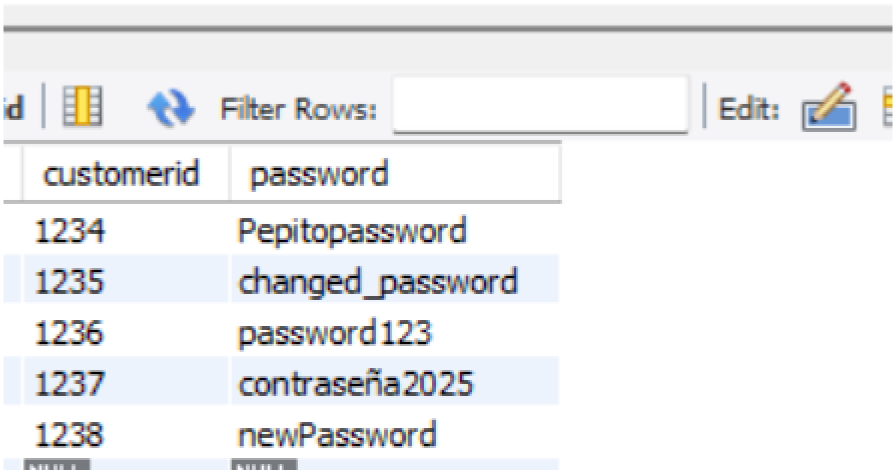
http://localhost:8088/api/login/editLogin

```
{
    "customerId": "1238",
    "password": "newPassword"
}
```

```
Actualización de inicio de sesión enviada
```



```
select * from customerorders.login;
```



customerid	password
1234	Pepitopassword
1235	changed_password
1236	password123
1237	contraseña2025
1238	newPassword