

PROYECTO

Sistema de Venta para Fast Food Foodly

Curso: Desarrollo Web Integrado

Sección:

(100000I59N)

Docente: Jorge Chicana Aspajo

Integrantes:

- **Ariza Rosas Sergio Antonio**
- **Camiloaga Cuenca Johann Gudwig**
- **Gallo Tume Joel Jesús**
- **Flores Mamani Jorge**
- **Mosquera Perales Roberto Alonso**

Lima, Perú

2025

Tabla de Contenido

Introducción.....	4
Capítulo I.....	5
Aspectos generales.....	5
Nombre del Negocio.....	5
Antecedentes e historia.....	5
Rubro del Negocio.....	6
Misión y Visión:.....	6
Problemática:.....	7
Alternativas de Solución:.....	7
Solución y Justificación del Proyecto:.....	8
Solución elegida:.....	8
Justificación:.....	8
Capítulo 2.....	9
Requerimientos funcionales y no funcionales Dentro de las funcionalidades debe agregar:....	9
Modelamiento:.....	9
Requerimientos funcionales:.....	11
Requerimientos no funcionales:.....	11
Layout del sitio web.....	12
Sistemas de menú de navegación.....	15
Formulario de sugerencias (ver anexos).....	16
Preguntas frecuentes.....	17
Contáctenos / Ubícanos.....	19
Capítulo 3.....	30
Base de Datos.....	30
Modelo Físico:.....	30
Diagrama de modelo fisco.....	34
Conexión a la Base de Datos.....	35
Operaciones en Base de Datos.....	37
Controladores.....	37
Modelo.....	46
Vista.....	56
Repositorio.....	62
Funcionalidades.....	67

Servicios.....	67
Vistas funcionales.....	86
DTOs.....	90
Entity.....	90
Response.....	91
Capítulo 4.....	103
JwUtils.Java.....	108
SecurityConfig.Java.....	110
SERVICIO WEB DE PARA ESCRITURA.....	111
Glosario de términos.....	111
Bibliografía o Anexos.....	113
Conclusiones.....	114
Recomendaciones.....	114

Figuras

Figura 1	13
Figura 2	14
Figura 3	15
Figura 4	16
Figura 5	17
Figura 6	18
Figura 7	19
Figura 8	20
Figura 9	21
Figura 10	24
Admin	27
Arqueo	28
Cajero	29
CAJA	30
Orden del cliente	31
Conjunto de ordenes	31
Producto	33
Usuario	34
Ejecución del BackEnd	35

Tablas

Tabla 1

10

Introducción

El presente proyecto tiene como objetivo el desarrollo de un sistema de punto de venta (POS) orientado a un restaurante de comida rápida (fast food). Este sistema está diseñado para optimizar diversos procesos clave del negocio, como la atención al cliente, la gestión de empleados, el control de inventario y el manejo eficiente de pedidos en cocina.

Además, el sistema permitirá registrar ventas en tiempo real, emitir comprobantes, generar reportes detallados y administrar las órdenes desde su ingreso hasta su despacho. De esta manera, se busca ofrecer una experiencia más ágil y satisfactoria tanto para los clientes como para el personal del establecimiento, fortaleciendo así la eficiencia operativa y la calidad del servicio.

Capítulo I

Aspectos generales

Nombre del Negocio

- FOODLY

Antecedentes e historia

En los últimos años, los negocios de comida rápida han enfrentado una creciente demanda y una competencia cada vez más fuerte, lo que ha impulsado la necesidad de contar con herramientas tecnológicas que optimicen sus operaciones. Uno de los elementos clave en este proceso ha sido la incorporación de sistemas de gestión de ventas, también conocidos como sistemas POS (Point of Sale).

Estos sistemas permiten automatizar el proceso de venta, registrar las transacciones en tiempo real, llevar un control de inventarios, generar reportes financieros y mejorar la experiencia del cliente al reducir tiempos de espera y errores en los pedidos. Además, ofrecen funcionalidades adicionales como la gestión de promociones, seguimiento del comportamiento del consumidor y vinculación con plataformas de delivery.

Para negocios como Foodly, que manejan un volumen considerable de ventas diarias y una gran variedad de platillos, contar con un sistema de gestión de ventas representa una ventaja competitiva significativa, ya que facilita la toma de decisiones basada en datos reales y mejora la eficiencia operativa.

Según Sánchez y López (2020), "la implementación de sistemas de información en los negocios gastronómicos permite alinear los procesos administrativos y operativos, garantizando un mejor control del negocio y una experiencia más fluida para el cliente".

Foodly nació como una idea innovadora de un grupo de jóvenes emprendedores apasionados por la comida rápida, pero que deseaban ofrecer algo distinto al mercado. Inspirados en cadenas como Popeyes, decidieron crear un restaurante que no solo ofreciera pollo frito, sino una gran variedad de platillos con sabores únicos, inspirados en diferentes culturas. En sus inicios, Foodly fue un pequeño local en una zona urbana con mucho tránsito. Gracias a su propuesta variada y accesible, el restaurante creció rápidamente y ahora busca expandirse a través de estrategias digitales y sucursales físicas.

Rubro del Negocio

Foodly pertenece al rubro de restaurantes de comida rápida dentro del sector gastronómico y de servicios alimenticios. Su enfoque principal es ofrecer platillos variados, de preparación rápida, a precios competitivos, con una experiencia moderna, ágil y accesible para todo tipo de público.

Misión y Visión:

Misión:

Brindar una experiencia gastronómica rápida y diversa, combinando calidad, sabor y atención excepcional, con un menú amplio que satisfaga los gustos de todos nuestros clientes.

Visión:

Ser reconocidos a nivel nacional como una cadena de comida rápida innovadora, que destaca por su variedad de platillos, su compromiso con la calidad y su cercanía con el cliente.

Problemática:

A pesar del éxito inicial, Foodly enfrenta varios desafíos:

- Alta competencia en el mercado de comida rápida.
- Dificultad para posicionarse como una marca única.
- Falta de presencia digital sólida.
- Limitado alcance a nuevos públicos fuera de su zona actual.

Alternativas de Solución:

- Desarrollar una aplicación web para pedidos y promociones.
 - Ampliar el menú con opciones veganas y saludables.
 - Fortalecer el marketing digital (redes sociales, influencers, Google Ads).
 - Crear combos exclusivos por temporada o por región.
-

- Implementar un programa de fidelización con recompensas.

Solución y Justificación del Proyecto:

Solución elegida:

Desarrollar una estrategia integral de marketing digital y fidelización, acompañada de una aplicación web para mejorar la experiencia del cliente.

Justificación:

Esta solución permite a Foodly destacar frente a sus competidores al mejorar la conexión con sus clientes, aumentar su presencia en línea, facilitar los pedidos y ofrecer recompensas por lealtad. Con una aplicación web y una buena estrategia digital, se espera incrementar las ventas, atraer nuevos clientes y consolidar la marca.

Capítulo 2

Requerimientos funcionales y no funcionales Dentro de las funcionalidades debe agregar:

Modelamiento:

Tabla 1

Descripción de los requerimientos funcionales de registros de Ventas de Servicios

Control de Sistema para Fast Food	
Nro. Requerimiento	Descripción
Consultas/informes	
R1	
R2	Informe de los platos (Admin).
R3	Informe del precio de cada plato (Cajero y Admin).
Almacenamiento	
R4	Datos de plato (producto): id_plato, nombre_plato, precio_plato, categ_plato

R5	Datos de usuario (usuario): id_user, nombre_user, dni_user, tipo_user, pass_user, corre_user
R6	Detalle de orden: id_orden, nombreCliente_orden, platos_orden, cantidad_orden, precio_orden, hora_orden, forpag_pago (forma de pago), nota_orden
R7	Detalle de Producto stock: id_product_stock, ini_stock, current_stock, total_sold
Procesamiento	
R8	Los platos se van agregando al detalle de orden, al final se muestra el precio total de la orden con la opción de modificar la orden.
R9	El administrador o gerente, al hacer uso de su sección en la página, puede visualizar reportes de las ventas del día.
R10	Los platos se agregarán al listado de pedidos del cajero.

Nota. Esta tabla muestra la descripción de cada requerimiento dividido en consultas, almacenamiento y procesamiento. Adaptación propia.

Requerimientos funcionales:

- o El empleado en la caja podrá añadir algún plato, modificarlo o eliminarlo (tipo un CRUD) antes de confirmar el pedido.
- o El cliente va a tener la posibilidad de pagar con múltiples opciones o Los administradores podrán hacer consultas de informes (observación) con respecto a las ventas del día mediante otra pestaña habiendo iniciado sesión en la página.
- o El empleado tiene acceso a la caja, pedido.
- o El empleado en caja puede generar un informe para la gestión de ordenes realizados por los clientes el día en curso (Arqueo).

Requerimientos no funcionales:

- o Mantenibilidad: El sistema debe estar documentado para facilitar futuras modificaciones.
- o Escalabilidad.
- o Seguridad: Permisos de usuario.

Layout del sitio web

Figura 1

Vista de Login



Nota. Visaulizamos el inicio de la aplicación de la empresa de comida Foodly.

Figura 2

Registro de Arqueos

The screenshot shows a user interface for managing考古 (Arqueos). At the top left is a logo consisting of three colored squares (red, black, and red) with the letters 'UTP' in white. To its right, the text 'Universidad Tecnológica del Perú' is written. On the far right, the text 'Desarrollo Web Integrado' is displayed.

The main title 'Arqueo de Caja' is centered at the top in a large, orange font. Below it, there are two date input fields: 'Fecha Inicial' (Initial Date) and 'Fecha Final' (Final Date), each with a calendar icon and a clear button. A search bar labeled 'Filtrar' (Filter) with a magnifying glass icon is positioned between them. To the right of the search bar is a 'Limpiar' (Clear) button with a circular arrow icon.

A green navigation bar below the search area contains the text 'Listado de Arqueos' (List of Archaeological Excavations) next to a list icon.

At the bottom of the page, another navigation bar is visible, featuring a blue background and white text. It includes a circular icon with a dot and the text 'Consolidado de Datos' (Data Consolidation).

Nota. Vizualizamos la sección de Arqueos de la aplicación.

Figura 3

Registro para Pedidos

The screenshot displays a web-based application for managing orders. At the top, there is a header with a logo consisting of the letters 'UTP' in white on a red background, followed by the text 'Universidad Tecnológica del Perú'. To the right of the logo, the text 'Desarrollo Web Integrado' is displayed.

The main content area is titled 'Pedidos' (Orders) in large orange text. Below the title, there is a green navigation bar with several buttons: 'Finalizar' (Finish), 'Filtrado' (Filter), 'Fecha Inicial' (Initial Date), 'Fecha Final' (Final Date), 'Filtrar' (Filter), and 'Limpiar' (Clear). There is also a button labeled 'Listado de Pedidos' (List of Orders).

On the left side, there is a sidebar titled 'Categorías' (Categories). It contains a section for 'Categorías' with a dropdown menu. Below this, there is a table with columns: 'Producto' (Product), 'Cantidad' (Quantity), 'Precio Unitario' (Unit Price), 'Subtotal' (Subtotal), and 'Acciones' (Actions). The table shows the following data:

Producto	Cantidad	Precio Unitario	Subtotal	Acciones
			Subtotal: \$/0.00	
			Impuesto (10%) \$/0.00	
			Total \$/0.00	

At the bottom of the sidebar, there is a button labeled 'Frame:29' and a green button labeled 'Limpiar Carrito' (Clear Cart).

The main content area also features a table titled 'Listado de Pedidos' (List of Orders) showing the following data:

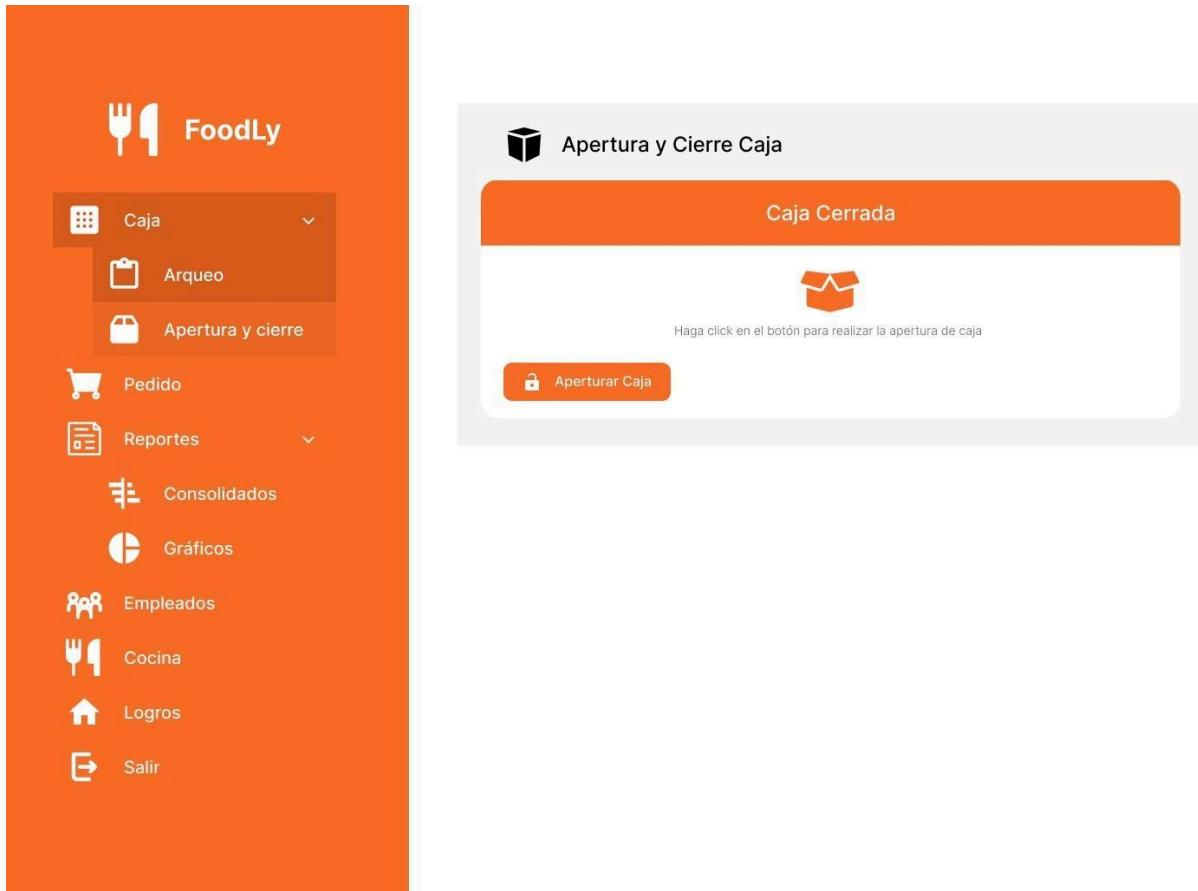
#	Cliente	Fecha	Total	Acción
1	Renzo	hoy 10:28 PM	\$/55.00	[button]
2	Jhon	hoy 10:34 PM	\$/63.00	[button]
3	Roxana	hoy 10:10 PM	\$/48.00	[button]
4	Pedro	hoy 10:50 PM	\$/55.00	[button]
5	carla	hoy 10:55 PM	\$/63.00	[button]
6	Roxana	hoy 10:58 PM	\$/48.00	[button]

Nota. El cliente puede realizar pedidos a través de la aplicación.

Sistemas de menú de navegación

Figura 4

Apertura y Cierre de cajas



Nota. Visualizamos las aperturas y cierres que el cliente a realizado.

Formulario de sugerencias (ver anexos)

Figura 5

Formularios y Sugerencias

Formulario de Sugerencias

Coloque sus sugerencias

Nombre

Email

Direccion 
Av.Perú N°2456

Telefono 
01 2554476
+51 955236587

Correo 
micorreo@gmail.com

> Enviar

Nota. En esta sección el cliente puede dejar las sugerencias sobre la empresa. .

Preguntas frecuentes

1. ¿Qué tipos de empleados están registrados o gestionados dentro del sistema?

El sistema gestiona tres tipos de empleados: Administrador, Cajero y Personal de Cocina.

Cada uno tiene acceso a secciones específicas según sus funciones, lo que garantiza una organización eficiente y un control adecuado dentro del sistema.

2. ¿Qué función cumple cada tipo de empleado en el sistema?

Administrador: Su principal función es gestionar el personal para cada área y asignar roles. Además, tiene la responsabilidad de reponer el stock de productos del sistema.

También puede visualizar todas las vistas y funcionalidades disponibles para los demás empleados.

Cajero: Es responsable de gestionar el punto de venta. Se encarga de registrar los pedidos realizados por los clientes, emitir comprobantes de pago y llevar a cabo los arqueos de caja al inicio y cierre de cada jornada.

Cocina: Tiene acceso al módulo de pedidos, donde puede visualizar las órdenes que han sido ingresadas por el cajero. Su función principal es preparar y despachar los productos solicitados, marcando los pedidos como listos una vez completados.

3. ¿Es posible modificar o eliminar un producto del menú?

Sí, el sistema permite modificar o eliminar productos del menú de forma sencilla. Los administradores pueden actualizar información como el nombre, precio, descripción o disponibilidad del producto, así como eliminar aquellos que ya no estén en oferta.

4. ¿El sistema permite llevar un control del stock de productos?

Sí, el sistema lleva un control individual del stock de cada producto. Se registra cuántas unidades han sido añadidas al inventario y cuántas han sido vendidas, lo que permite mantener un control preciso del nivel de existencias en todo momento.

5. ¿Cómo se registra una nueva venta en el sistema?

Cada vez que se ingresa una orden en el sistema como ocurre normalmente en establecimientos de comida rápida, se registra automáticamente como una venta. Esto incluye el detalle de los productos solicitados, la hora de la transacción y el responsable de la operación

6. ¿El sistema requiere conexión a internet para funcionar?

Sí, el sistema es una aplicación web, por lo que necesita conexión a internet. Al estar alojado en la nube, permite acceder desde cualquier dispositivo con navegador y mantener los datos actualizados en tiempo real

7. ¿Se puede generar un reporte de ventas?

Sí, el sistema permite generar reportes detallados de ventas. Por cada transacción registrada, se puede acceder a información como fecha, hora, productos vendidos, monto total y empleado responsable. Estos reportes pueden visualizarse en pantalla o exportarse en formatos como PDF para un mejor análisis contable y de gestión.

Contáctenos / Ubícanos

Figura 6

Contacto y Ubicación

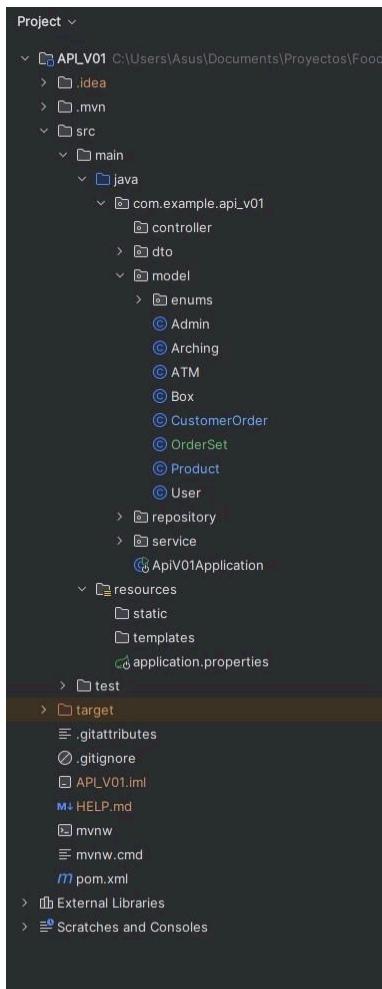


Nota. En esta figura mostramos un apartado del contacto y ubicación

Código

Figura 7

Estructura de paquetes

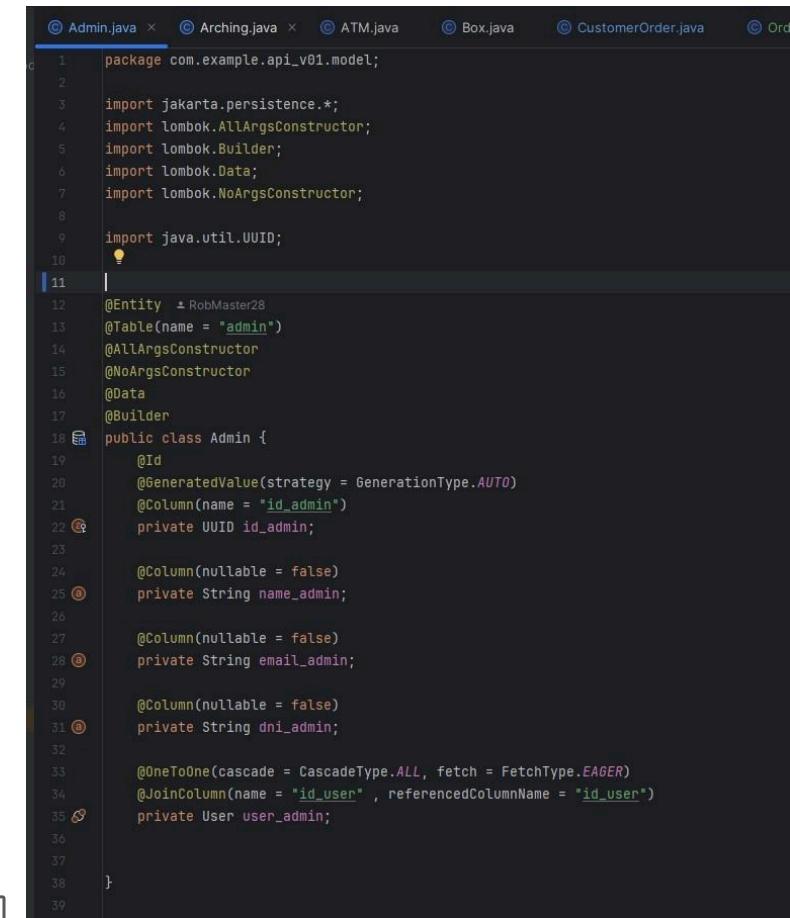


Nota. En esta figura visualizamos como esta estructurado los códigos,

Admin

Figura 8

Modelo de Administrador



```
 1 package com.example.api_v01.model;
 2
 3 import jakarta.persistence.*;
 4 import lombok.AllArgsConstructor;
 5 import lombok.Builder;
 6 import lombok.Data;
 7 import lombok.NoArgsConstructor;
 8
 9 import java.util.UUID;
10
11
12 @Entity
13 @Table(name = "admin")
14 @AllArgsConstructor
15 @NoArgsConstructor
16 @Data
17 @Builder
18 public class Admin {
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     @Column(name = "id_admin")
22     private UUID id_admin;
23
24     @Column(nullable = false)
25     private String name_admin;
26
27     @Column(nullable = false)
28     private String email_admin;
29
30     @Column(nullable = false)
31     private String dni_admin;
32
33     @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
34     @JoinColumn(name = "id_user", referencedColumnName = "id_user")
35     private User user_admin;
36
37
38 }
39
```

Nota. En esta figura mostramos un apartado de cada columna del administrador.

Arqueo

Figura 9

Visualización Arching Java

```

© Arching.java × © ATM.java × © Box.java      © CustomerOrder.java      © OrderSet.java      © Product.java      ©
1
2
3
4
5
6
7 import lombok.NoArgsConstructor;
8
9 import java.time.LocalDate;
10 import java.time.LocalDateTime;
11 import java.util.UUID;
12 @Entity
13 @Table(name = "arching")
14 @NoArgsConstructor
15 @AllArgsConstructor
16 @Data
17 @Builder
18 public class Arching {
19
20     @Id
21     @GeneratedValue(strategy = GenerationType.AUTO)
22     @Column(name = "id_arching")
23     private UUID id_arching;
24
25     @Column(nullable = false)
26     private LocalDate date;
27
28     @Column(nullable = false)
29     private LocalDateTime start_time;
30
31     @Column(nullable = false)
32     private LocalDateTime end_time;
33
34     @Column(nullable = false)
35     private Double init_amount;
36
37     @Column(nullable = false)
38     private Double final_amount;
39
40     @Column(nullable = false)
41     private Double total_amount;
42
43     @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
44     @JoinColumn(name = "id_box", referencedColumnName = "id_box")
45     private Box box;
46
47 }
48

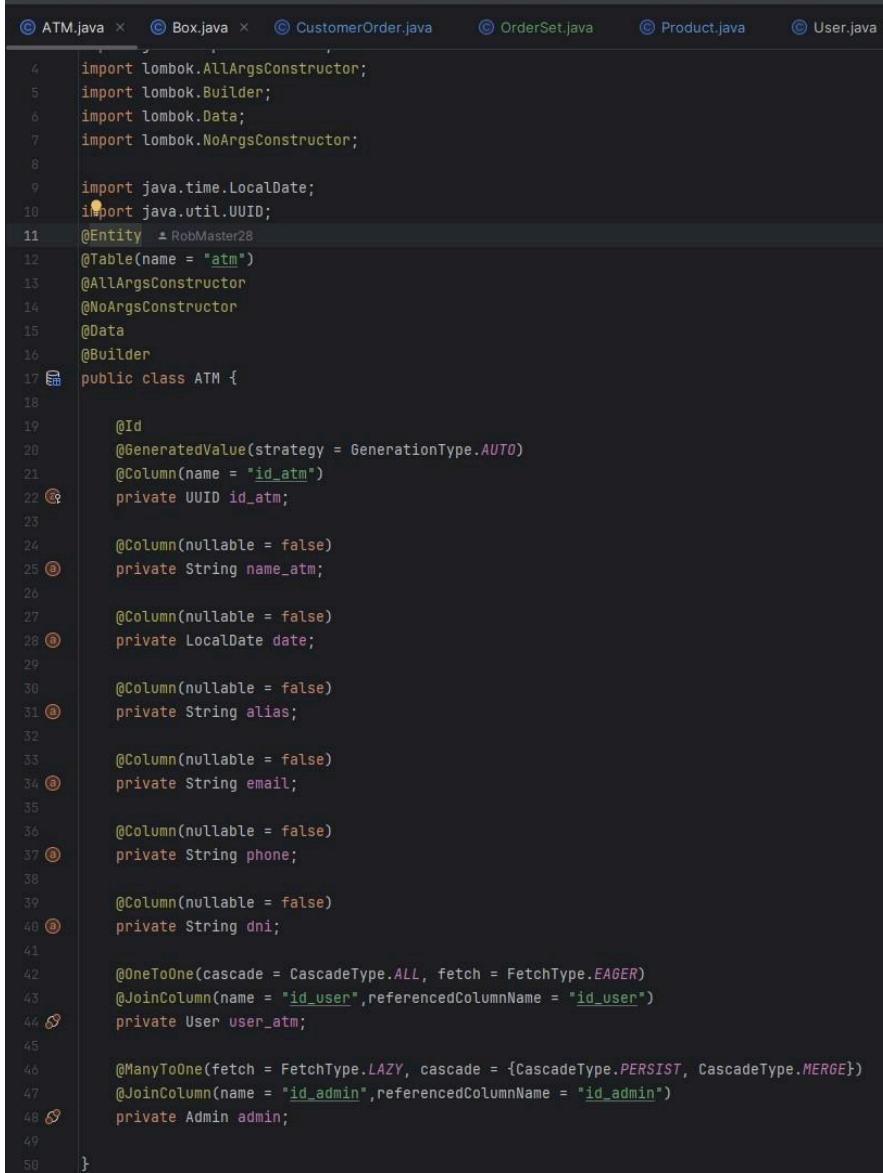
```

Nota. En esta figura mostramos un apartado de cada columna de Arching Java

Cajero

Figura 10

Visualización del Cajero



```

 4 import lombok.AllArgsConstructor;
 5 import lombok.Builder;
 6 import lombok.Data;
 7 import lombok.NoArgsConstructor;
 8
 9 import java.time.LocalDate;
10 import java.util.UUID;
11 @Entity
12 @Table(name = "atm")
13 @NoArgsConstructor
14 @AllArgsConstructor
15 @Data
16 @Builder
17 public class ATM {
18
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     @Column(name = "id_atm")
22     private UUID id_atm;
23
24     @Column(nullable = false)
25     private String name_atm;
26
27     @Column(nullable = false)
28     private LocalDate date;
29
30     @Column(nullable = false)
31     private String alias;
32
33     @Column(nullable = false)
34     private String email;
35
36     @Column(nullable = false)
37     private String phone;
38
39     @Column(nullable = false)
40     private String dni;
41
42     @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
43     @JoinColumn(name = "id_user", referencedColumnName = "id_user")
44     private User user_atm;
45
46     @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
47     @JoinColumn(name = "id_admin", referencedColumnName = "id_admin")
48     private Admin admin;
49 }
50

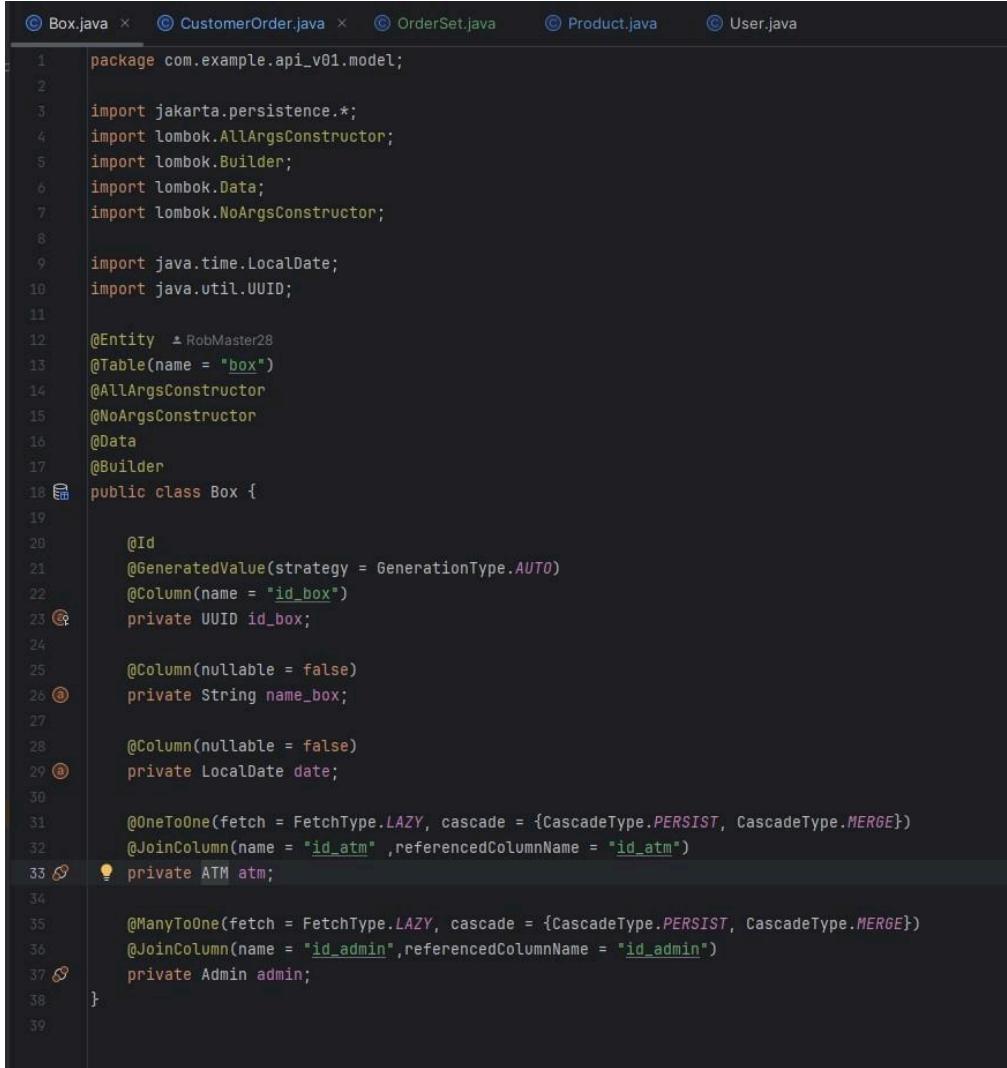
```

Nota. En esta figura mostramos un apartado de cada columna de Cajero

CAJA

Figura 11

Visualización de la Caja



```

 ① Box.java × ② CustomerOrder.java × ③ OrderSet.java      ④ Product.java    ⑤ User.java
 1 package com.example.api_v01.model;
 2
 3 import jakarta.persistence.*;
 4 import lombok.AllArgsConstructor;
 5 import lombok.Builder;
 6 import lombok.Data;
 7 import lombok.NoArgsConstructor;
 8
 9 import java.time.LocalDate;
10 import java.util.UUID;
11
12 @Entity  ✎ RobMaster28
13 @Table(name = "box")
14 @AllArgsConstructor
15 @NoArgsConstructor
16 @Data
17 @Builder
18 public class Box {
19
20     @Id
21     @GeneratedValue(strategy = GenerationType.AUTO)
22     @Column(name = "id_box")
23     private UUID id_box;
24
25     @Column(nullable = false)
26     private String name_box;
27
28     @Column(nullable = false)
29     private LocalDate date;
30
31     @OneToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
32     @JoinColumn(name = "id_atm" ,referencedColumnName = "id_atm")
33     private ATM atm;
34
35     @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
36     @JoinColumn(name = "id_admin" ,referencedColumnName = "id_admin")
37     private Admin admin;
38
39 }

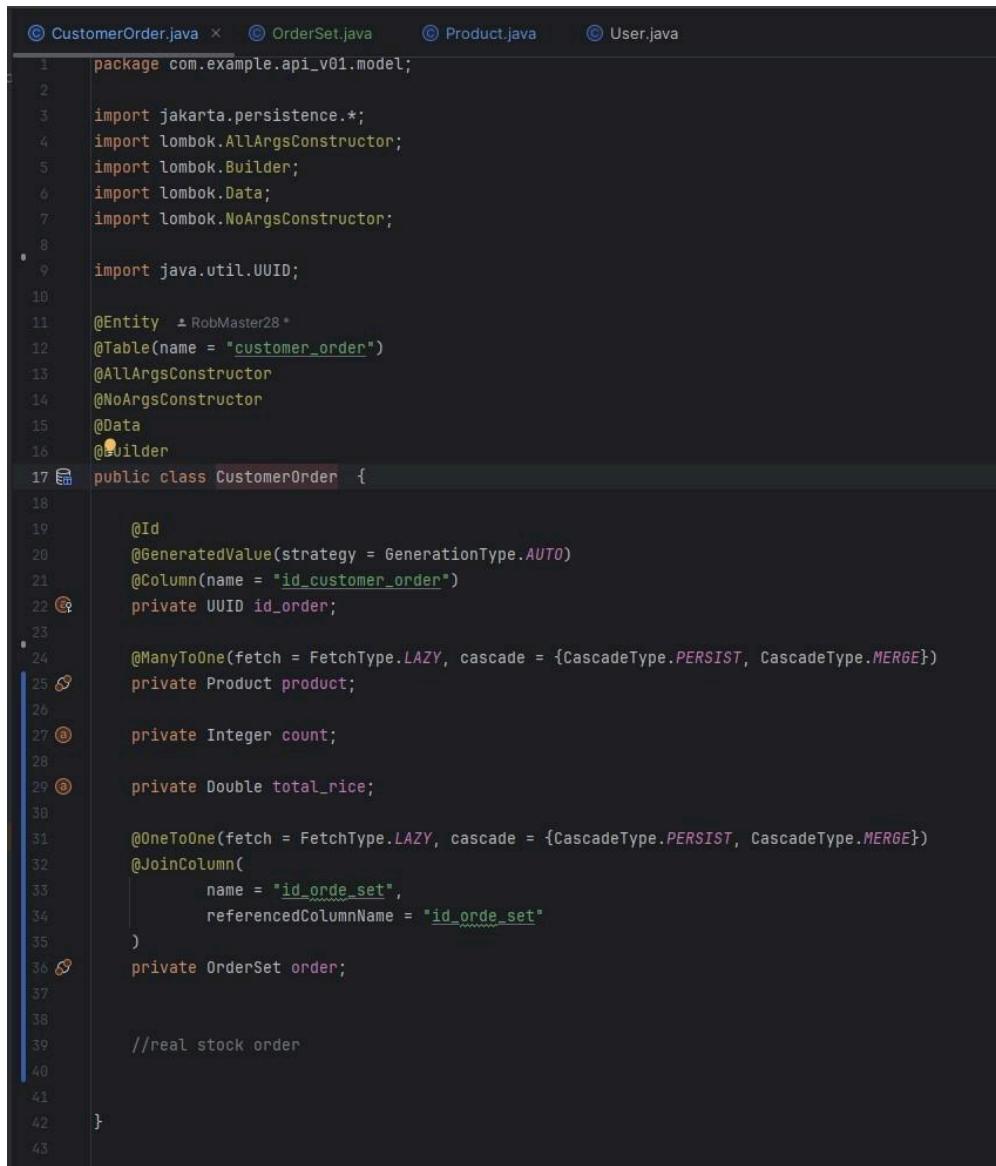
```

Nota. En esta figura mostramos un apartado de cada columna de la Caja

Orden del cliente

Figura 12

Visualización de la Orden del Cliente



```
CustomerOrder.java
package com.example.api_v01.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.UUID;

@Entity
@Table(name = "customer_order")
@AllArgsConstructor
@NoArgsConstructor
@Data
@Builder
public class CustomerOrder {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id_customer_order")
    private UUID id_order;

    @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
    private Product product;

    private Integer count;

    private Double total_rice;

    @OneToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
    @JoinColumn(
        name = "id_order_set",
        referencedColumnName = "id_order_set"
    )
    private OrderSet order;

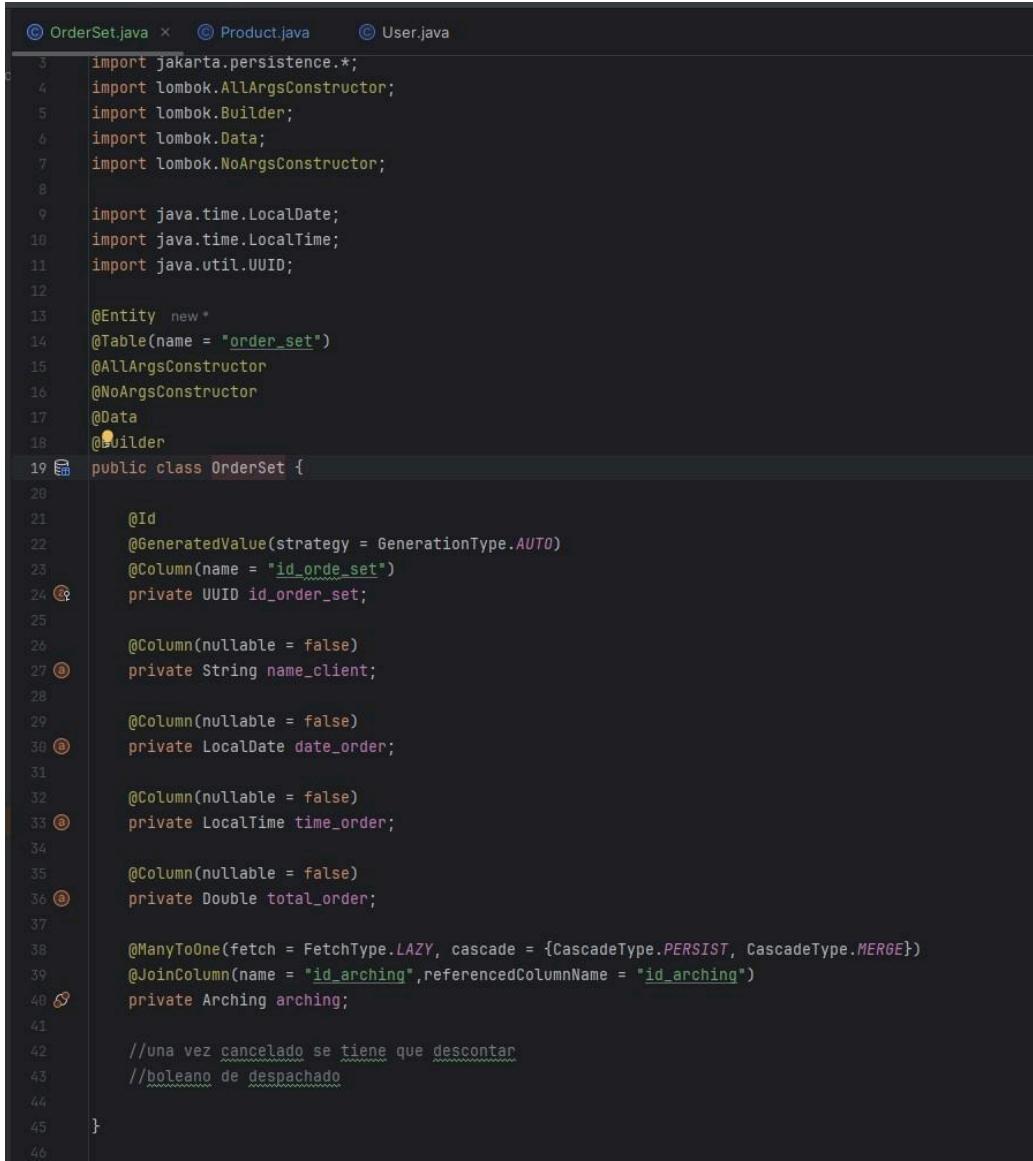
    //real stock order
}
```

Nota. En esta figura mostramos un apartado de la Orden del Cliente

Conjunto de ordenes

Figura 13

Visualización del Conjunto de Ordenes



```

OrderSet.java x Product.java User.java
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.UUID;

@Entity
@Table(name = "order_set")
@AllArgsConstructor
@NoArgsConstructor
@Data
@AllArgsConstructor
public class OrderSet {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id_order_set")
    private UUID id_order_set;

    @Column(nullable = false)
    private String name_client;

    @Column(nullable = false)
    private LocalDate date_order;

    @Column(nullable = false)
    private LocalTime time_order;

    @Column(nullable = false)
    private Double total_order;

    @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
    @JoinColumn(name = "id_arching", referencedColumnName = "id_arching")
    private Arching arching;

    //una vez cancelado se tiene que descontar
    //booleano de despachado
}

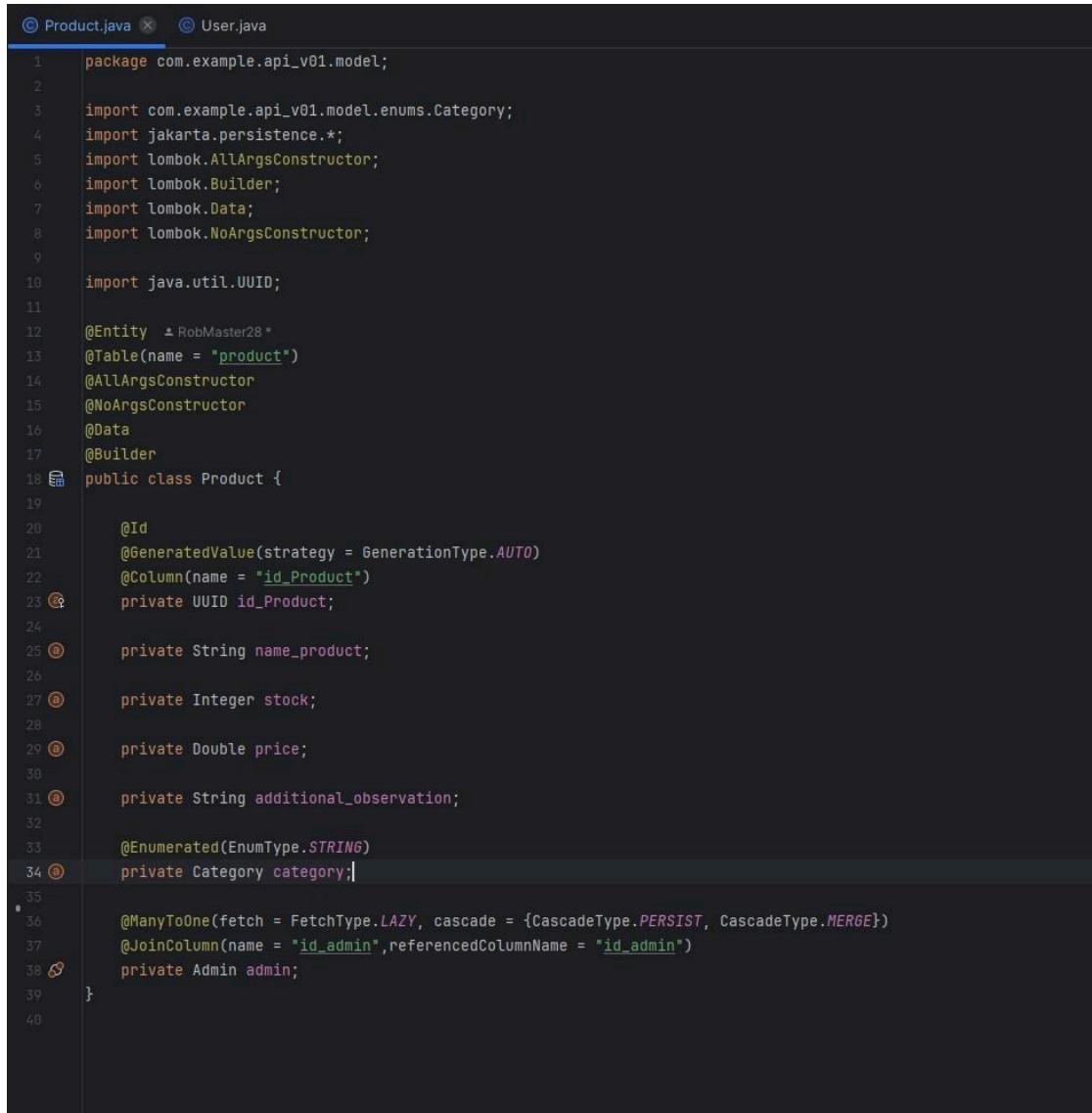
```

Nota. En esta figura mostramos un apartado de la Orden del Cliente

Producto

Figura 14

Visualización del Producto



The screenshot shows a Java code editor with two tabs: 'Product.java' and 'User.java'. The 'Product.java' tab is active, displaying the following code:

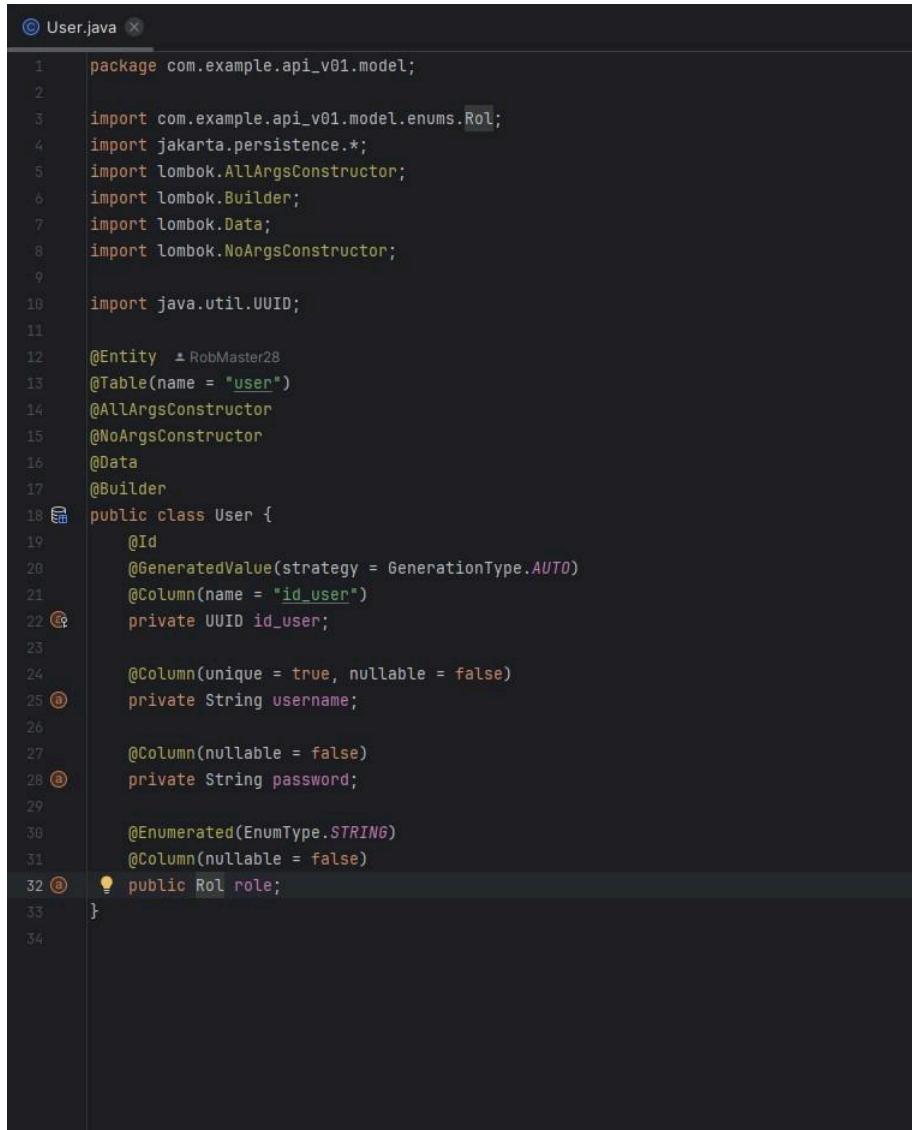
```
1 package com.example.api_v01.model;
2
3 import com.example.api_v01.model.enums.Category;
4 import jakarta.persistence.*;
5 import lombok.AllArgsConstructorConstructor;
6 import lombok.Builder;
7 import lombok.Data;
8 import lombok.NoArgsConstructorConstructor;
9
10 import java.util.UUID;
11
12 @Entity
13 @Table(name = "product")
14 @AllArgsConstructorConstructor
15 @NoArgsConstructorConstructor
16 @Data
17 @Builder
18 public class Product {
19
20     @Id
21     @GeneratedValue(strategy = GenerationType.AUTO)
22     @Column(name = "id_product")
23     private UUID id_product;
24
25     private String name_product;
26
27     private Integer stock;
28
29     private Double price;
30
31     private String additional_observation;
32
33     @Enumerated(EnumType.STRING)
34     private Category category;
35
36     @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
37     @JoinColumn(name = "id_admin", referencedColumnName = "id_admin")
38     private Admin admin;
39 }
40
```

Nota. En esta figura mostramos un apartado de la Orden del Cliente

Usuario

Figura 15

Visualización del Producto



The screenshot shows a code editor window with the file 'User.java' open. The code defines a class 'User' with annotations for Entity, Table, and various Lombok annotations. The class has fields for id_user, username, password, and role.

```
 1 package com.example.api_v01.model;
 2
 3 import com.example.api_v01.model.enums.Rol;
 4 import jakarta.persistence.*;
 5 import lombok.AllArgsConstructor;
 6 import lombok.Builder;
 7 import lombok.Data;
 8 import lombok.NoArgsConstructor;
 9
10 import java.util.UUID;
11
12 @Entity
13 @Table(name = "user")
14 @NoArgsConstructor
15 @AllArgsConstructor
16 @Data
17 @Builder
18 public class User {
19     @Id
20     @GeneratedValue(strategy = GenerationType.AUTO)
21     @Column(name = "id_user")
22     private UUID id_user;
23
24     @Column(unique = true, nullable = false)
25     private String username;
26
27     @Column(nullable = false)
28     private String password;
29
30     @Enumerated(EnumType.STRING)
31     @Column(nullable = false)
32     public Rol role;
33 }
34
```

Nota. En esta figura mostramos un apartado de la Orden del Cliente

Ejecución del BackEnd

Figura 16

Visualización del Producto

```

:: Spring Boot ::          (v3.4.4)

2025-04-20T20:18:55.545-05:00 INFO 4372 --- [API_V01] [ restartedMain] com.example.api_v01.ApiV01Application      : Starting ApiV01Application using Java 17.0.6 with PID 4372
2025-04-20T20:18:55.545-05:00 INFO 4372 --- [API_V01] [ restartedMain] com.example.api_v01.ApiV01Application      : No active profile set, falling back to 1 default profile: 
2025-04-20T20:18:55.734-05:00 INFO 4372 --- [API_V01] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode
2025-04-20T20:18:55.768-05:00 INFO 4372 --- [API_V01] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 32 ms. Found 0 repository interfaces
2025-04-20T20:18:55.859-05:00 INFO 4372 --- [API_V01] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer      : Tomcat initialized with port 8080 (http)
2025-04-20T20:18:55.860-05:00 INFO 4372 --- [API_V01] [ restartedMain] o.apache.catalina.core.StandardService       : Starting service [Tomcat]
2025-04-20T20:18:55.860-05:00 INFO 4372 --- [API_V01] [ restartedMain] o.apache.catalina.core.StandardEngine        : Starting Servlet engine: [Apache Tomcat/10.1.39]
2025-04-20T20:18:55.876-05:00 INFO 4372 --- [API_V01] [ restartedMain] w.s.c.ServletWebServerApplicationContext      : Initializing Spring embedded WebApplicationContext
2025-04-20T20:18:55.876-05:00 INFO 4372 --- [API_V01] [ restartedMain] o.s.w.c.RootWebApplicationContext           : Root WebApplicationContext: initialization completed in 3ms
2025-04-20T20:18:55.912-05:00 INFO 4372 --- [API_V01] [ restartedMain] o.hibernate.jpa.internal.util.LogHelper      : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-04-20T20:18:55.915-05:00 INFO 4372 --- [API_V01] [ restartedMain] o.h.c.internal.RegionFactoryInitiator        : HHH000026: Second-level cache disabled
2025-04-20T20:18:55.924-05:00 INFO 4372 --- [API_V01] [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo           : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-04-20T20:18:55.925-05:00 INFO 4372 --- [API_V01] [ restartedMain] com.zaxxer.hikari.HikariDataSource        : HikariPool-5 - Starting...
2025-04-20T20:18:55.946-05:00 INFO 4372 --- [API_V01] [ restartedMain] com.zaxxer.hikari.HikariPool           : HikariPool-5 - Added connection com.mysql.cj.jdbc.Connection@53a33f2
2025-04-20T20:18:55.946-05:00 INFO 4372 --- [API_V01] [ restartedMain] com.zaxxer.hikari.HikariDataSource        : HikariPool-5 - Start completed.
2025-04-20T20:18:55.947-05:00 INFO 4372 --- [API_V01] [ restartedMain] org.hibernate.orm.connections.Pooling      : HHH010001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-5)']
Database driver: undefined/unknown
Database version: 8.0.40
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-04-20T20:18:56.095-05:00 INFO 4372 --- [API_V01] [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to 'true' if you want to use JBoss Seam's JTA implementation)
2025-04-20T20:18:56.175-05:00 INFO 4372 --- [API_V01] [ restartedMain] j.LocalContainerEntityManagerFactoryBean      : Initialized JPA EntityManagerFactory for persistence unit
2025-04-20T20:18:56.196-05:00 WARN 4372 --- [API_V01] [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration      : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during a view's rendering. Explicitly configure spring.jpa.open-in-view=false if you don't want this.
2025-04-20T20:18:56.374-05:00 WARN 4372 --- [API_V01] [ restartedMain] .s.s.UserDetailsServiceAutoConfiguration      :
Using generated security password: 72bd04ff71-666e-40fa-a206-d4efe25a3970
This generated password is for development use only. Your security configuration must be updated before running your application in production.

2025-04-20T20:18:56.375-05:00 INFO 4372 --- [API_V01] [ restartedMain] n$InitializeUserDetailsServiceManagerConfigurer : Global AuthenticationManager configured with UserDetailsService

```

Capítulo 3

Base de Datos

Modelo Físico:

```
CREATE TABLE `atm` (
  `id_atm` binary(16) NOT NULL,
  `alias` varchar(255) NOT NULL,
  `date` date NOT NULL,
  `dni` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `name_atm` varchar(255) NOT NULL,
  `phone` varchar(255) NOT NULL,
  `id_admin` binary(16) DEFAULT NULL,
  `id_user` binary(16) DEFAULT NULL,
  PRIMARY KEY (`id_atm`),
  UNIQUE KEY `UKnuqum0ohke2948wx1qt535syk` (`id_user`),
  KEY `FKgad4t1td06cv38d0adcu8n1iv` (`id_admin`),
  CONSTRAINT `FKgad4t1td06cv38d0adcu8n1iv` FOREIGN KEY (`id_admin`) REFERENCES `admin` (`id_admin`),
  CONSTRAINT `FKn88etm9oaj5c01ad7mcbx2bt4` FOREIGN KEY (`id_user`) REFERENCES `user` (`id_user`)
)

CREATE TABLE `box` (
  `id_box` binary(16) NOT NULL,
  `date` date NOT NULL,
  `name_box` varchar(255) NOT NULL,
  `id_admin` binary(16) DEFAULT NULL,
  `id_atm` binary(16) DEFAULT NULL,
  `is_open` bit(1) DEFAULT NULL,
  PRIMARY KEY (`id_box`),
  UNIQUE KEY `UKsuplhgy19ebc4t8bko9skqyrl` (`id_atm`),
  KEY `FKjrbbfuwcxx5px6jbddqq818pvg` (`id_admin`),
  CONSTRAINT `FKigcq3oxl5at8e0hsde90w1s6` FOREIGN KEY (`id_atm`) REFERENCES `atm` (`id_atm`),
  CONSTRAINT `FKjrbbfuwcxx5px6jbddqq818pvg` FOREIGN KEY (`id_admin`) REFERENCES `admin` (`id_admin`)
)
```

```
CREATE TABLE `customer_order` (
  `id_customer_order` binary(16) NOT NULL,
  `count` int DEFAULT NULL,
  `total_rice` double DEFAULT NULL,
  `id_orde_set` binary(16) DEFAULT NULL,
  `product_id_product` binary(16) DEFAULT NULL,
  `id_product` binary(16) DEFAULT NULL,
  PRIMARY KEY (`id_customer_order`),
  KEY `FK3pe7mou9vbyrd43r6ewyd47tfp` (`id_orde_set`),
  KEY `FKrqcux88e70n3sv9jp3i5m1261` (`product_id_product`),
  KEY `FKsnkhbuafrwggqoopc2w1pwu` (`id_product`),
  CONSTRAINT `FK3pe7mou9vbyrd43r6ewyd47tfp` FOREIGN KEY (`id_orde_set`) REFERENCES `order_set` (`id_orde_set`),
  CONSTRAINT `FKrqcux88e70n3sv9jp3i5m1261` FOREIGN KEY (`product_id_product`) REFERENCES `product` (`id_product`),
  CONSTRAINT `FKsnkhbuafrwggqoopc2w1pwu` FOREIGN KEY (`id_product`) REFERENCES `product` (`id_product`)
)

CREATE TABLE `order_set` (
  `id_orde_set` binary(16) NOT NULL,
  `date_order` date NOT NULL,
  `dispatch` bit(1) DEFAULT NULL,
  `name_client` varchar(255) NOT NULL,
  `time_order` time(6) NOT NULL,
  `total_order` double DEFAULT NULL,
  `id_arching` binary(16) DEFAULT NULL,
  `active` bit(1) DEFAULT NULL,
  PRIMARY KEY (`id_orde_set`),
  KEY `FKnartswa41qmhkqkkw52kp6x2s` (`id_arching`),
  CONSTRAINT `FKnartswa41qmhkqkkw52kp6x2s` FOREIGN KEY (`id_arching`) REFERENCES `arching` (`id_arching`)
)
```

```
CREATE TABLE `product` (
  `id_product` binary(16) NOT NULL,
  `additional_observation` varchar(255) DEFAULT NULL,
  `category` enum('BEBIDAS','COMIDA_RAPIDA','CRIOLLOS','MARINO','PASTAS') DEFAULT NULL,
  `name_product` varchar(255) DEFAULT NULL,
  `price` double DEFAULT NULL,
  `id_admin` binary(16) DEFAULT NULL,
  `id_product_stock` binary(16) DEFAULT NULL,
  PRIMARY KEY (`id_product`),
  UNIQUE KEY `UKcwr1t8gurq0rsrm1210v5t9kq` (`id_product_stock`),
  KEY `FKBe6b0u54i6glqmapv3xc86rj8` (`id_admin`),
  CONSTRAINT `FKBe6b0u54i6glqmapv3xc86rj8` FOREIGN KEY (`id_admin`) REFERENCES `admin` (`id_admin`),
  CONSTRAINT `FKrggikpry7cylkr8vibqgayfq` FOREIGN KEY (`id_product_stock`) REFERENCES `product_stock` (`id_product_stock`)
)

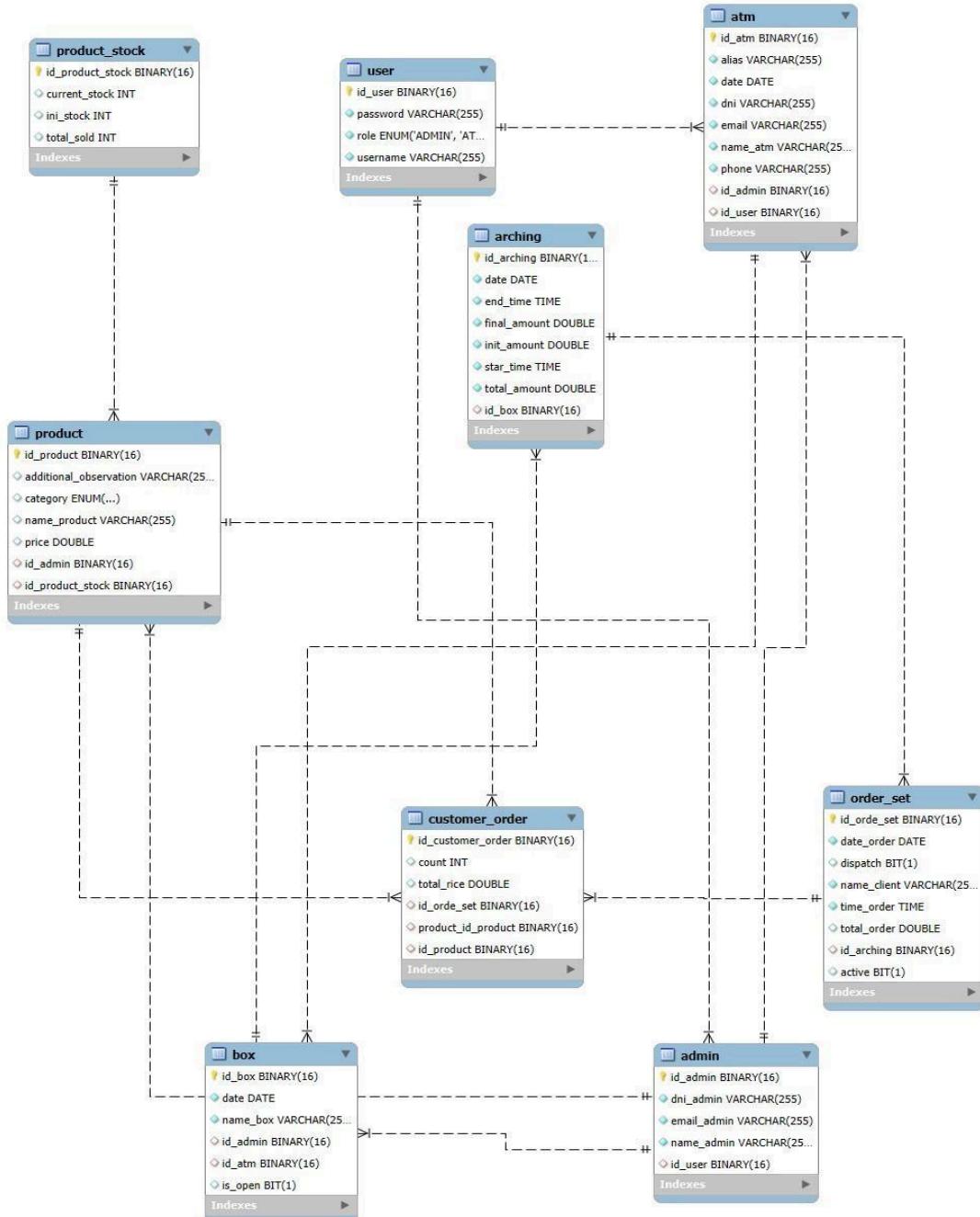
CREATE TABLE `product_stock` (
  `id_product_stock` binary(16) NOT NULL,
  `current_stock` int DEFAULT NULL,
  `ini_stock` int DEFAULT NULL,
  `total_sold` int DEFAULT NULL,
  PRIMARY KEY (`id_product_stock`)
)

CREATE TABLE `user` (
  `id_user` binary(16) NOT NULL,
  `password` varchar(255) NOT NULL,
  `role` enum('ADMIN','ATM') NOT NULL,
  `username` varchar(255) NOT NULL,
  PRIMARY KEY (`id_user`),
  UNIQUE KEY `UKsbBbbouerSwak8vyily4pf2bx` (`username`)
)
```

```
CREATE TABLE `admin` (
  `id_admin` binary(16) NOT NULL,
  `dni_admin` varchar(255) NOT NULL,
  `email_admin` varchar(255) NOT NULL,
  `name_admin` varchar(255) NOT NULL,
  `id_user` binary(16) DEFAULT NULL,
  PRIMARY KEY (`id_admin`),
  UNIQUE KEY `UK9vm4uu4nvcvilgmrqdo5vlrg` (`id_user`),
  CONSTRAINT `FK4ert0ej65rd0yse7b7ukmi82t` FOREIGN KEY (`id_user`) REFERENCES `user` (`id_user`)
)

CREATE TABLE `arching` (
  `id_arching` binary(16) NOT NULL,
  `date` date NOT NULL,
  `end_time` time(6) NOT NULL,
  `final_amount` double NOT NULL,
  `init_amount` double NOT NULL,
  `star_time` time(6) NOT NULL,
  `total_amount` double NOT NULL,
  `id_box` binary(16) DEFAULT NULL,
  PRIMARY KEY (`id_arching`),
  KEY `FKsrd11gwdpcbn0yir8kyxgsk8a` (`id_box`),
  CONSTRAINT `FKsrd11gwdpcbn0yir8kyxgsk8a` FOREIGN KEY (`id_box`) REFERENCES `box` (`id_box`)
)
```

Diagrama de modelo fisco

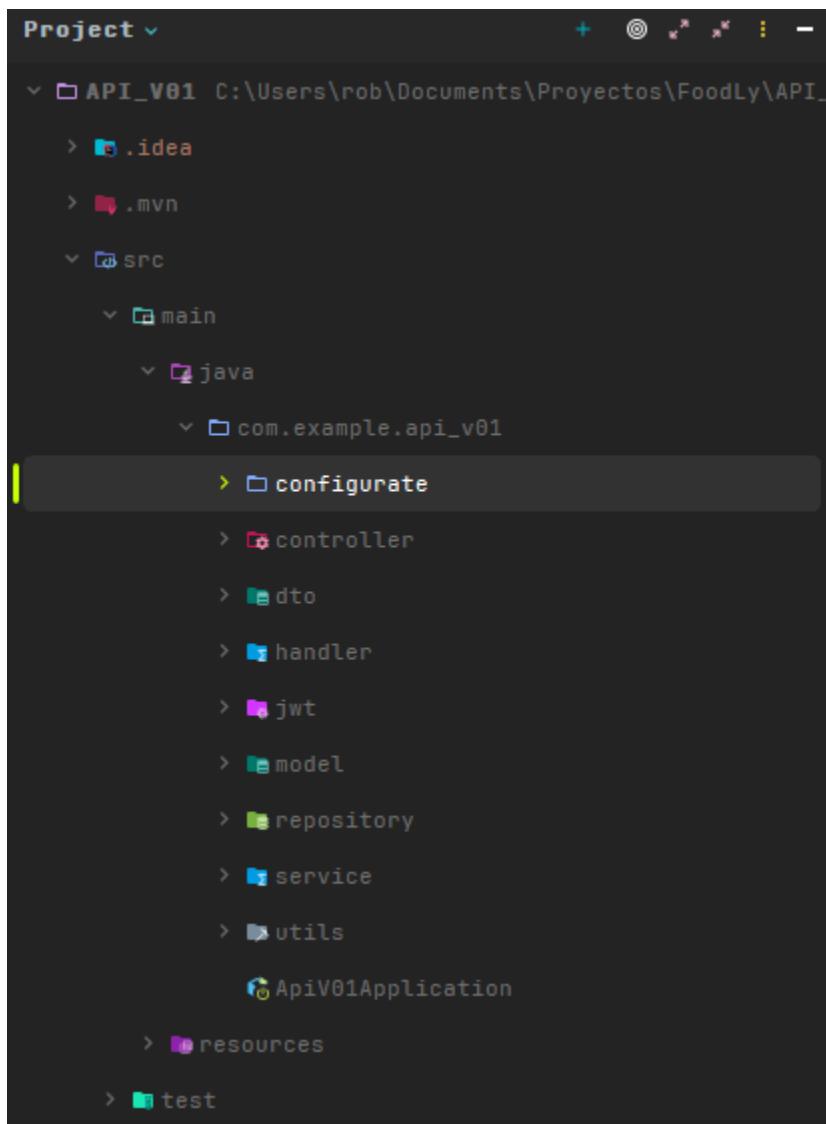


Conexión a la Base de Datos

application.properties

```
application.properties x
1  spring.application.name=API_V01
2  spring.datasource.username=root
3  spring.datasource.password=672004
4  spring.datasource.url=jdbc:mysql://localhost:3306/restaurant_system
5  server.servlet.context-path=/api/v1.5/
6  spring.jpa.generate-ddl=true
7  spring.jpa.show-sql=true
8  spring.jpa.hibernate.ddl-auto=update
9
10 #--- JWT ---
11 jwt-secret-word=53eddfc395bbaff035dbf636b8aa47b555eab31c42ee8e0fc16f7c1d825d6b23
12 jwt-seconds=600
13 |
```

Estructura del proyecto



Operaciones en Base de Datos

Controladores

AdminController	ProductController	BoxController
<pre>AdminController (AdminService) adminservice IminService getAdmin (UUID) Iity <?> getAdminList () Iity <?></pre>	<pre>ProductController (ProductService) productService ProductService getAllProductsByCategory (Category) Ity <?> getAllProducts () ResponseEntity <?> getProduct (UUID) ResponseEntity <?> updateProduct (UUID, ProductResponseDTO) > deleteProduct (UUID) ResponseEntity <?> getAllProductsByName (String) IseEntity <?> CreateProduct (UUID, ProductResponseDTO) ></pre>	<pre>BoxController (BoxService) boxservice BoxService getBoxes () ResponseEntity <?> getBoxesByAtm (UUID) IseEntity <?> getBoxInfo (UUID) responseEntity <?> OnBox (UUID, ArchingInitDTO) > <?> assignAtmToBox (UUID, UUID) Iy <?> OffBox (UUID, UUID) onseEntity <?> saveBox (UUID, BoxNameDTO) Ity <?></pre>
AuthController		
<pre>AuthController (AuthService) authService AuthService login (LoginDTO) Iity <?></pre>		
ATMController	ArchingController	StockController
<pre>ATMController (ATMService) atmservice ATMService getAllATMs () ResponseEntity <?> deleteATM (UUID) ResponseEntity <?> getAtmByName (String) ResponseEntity <?> updateATM (UUID, AtmResponseDTO) Iity <?> assignUserATM (UUID, RegisterAtmUserDTO) > saveATM (UUID, AtmResponseDTO) IseEntity <?> getAtmById (UUID) ResponseEntity <?></pre>	<pre>ArchingController (ArchingService) archingService ArchingService getArchingByATM (UUID) eEntity <?> getArchingByNameATM (String) <?> getAllArching () leponseEntity <?> getArchingById (UUID) seEntity <?> getArchingByNameBox (String) <?> getArchingByBox (UUID) eEntity <?></pre>	<pre>StockController (ProductStockService) productStockService ProductStockService discountStock (StockChangeRequestDTO) ?> getAllStocks () ResponseEntity <?> getStock (UUID) ResponseEntity <?> getAllStocksCategory (Category) Iity <?> updateProduct (ProductStockDTO) Iity <?> increaseStock (StockChangeRequestDTO) ?> cleanStock (UUID) ResponseEntity <?> getAllStocksNameProduct (String) Ity <?></pre>
OrderSetController		
<pre>OrderSetController (OrderSetOrchestratorService) orderSetOrchestratorService OrchestratorService saveOrderSet (UUID, SaveOrderSetListCustomerDTO) getOrderSet (UUID) ResponseEntity <?> getListOrderSetByCustomer (String) onseEntity <?> getListOrderSetByArching (UUID) esponseEntity <?></pre>		

AdminController

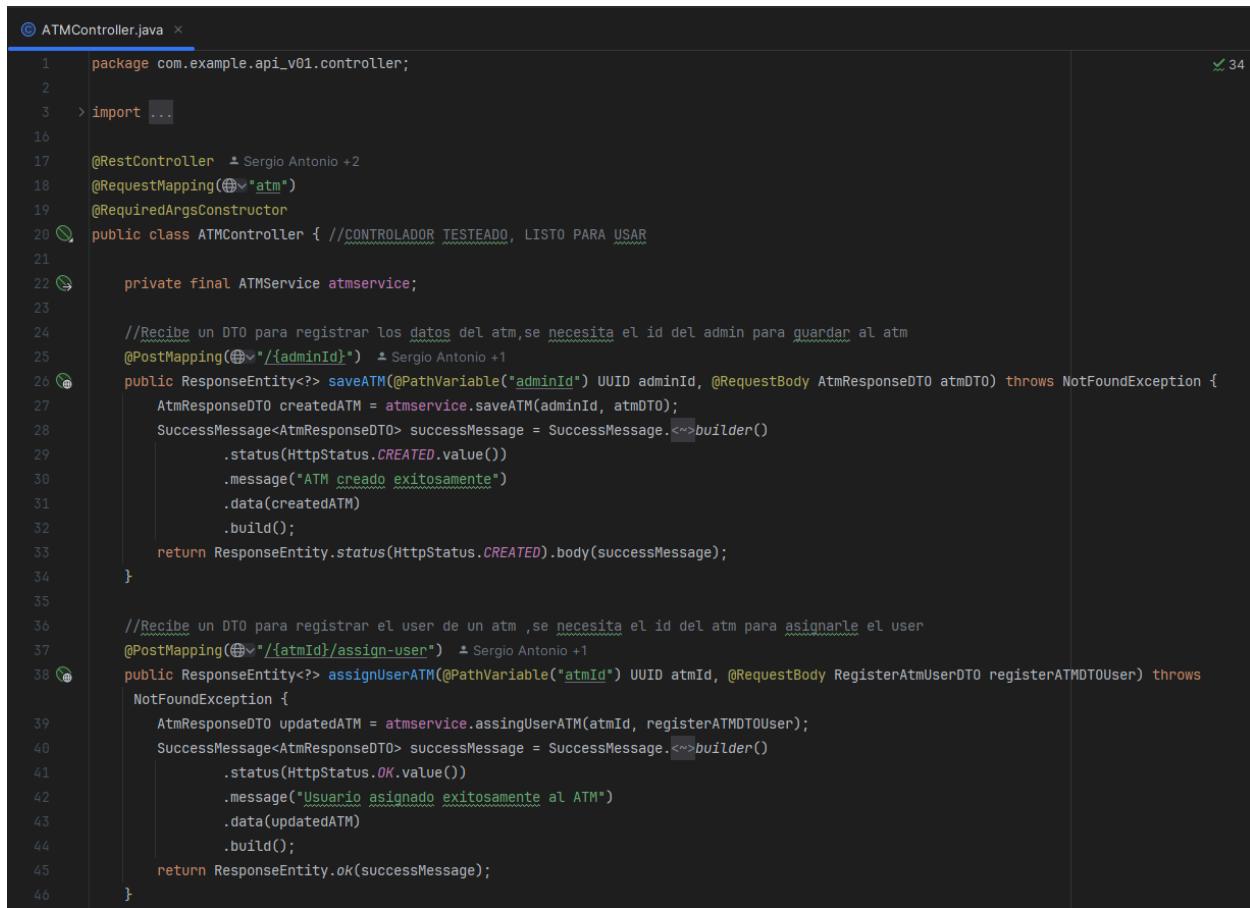
```
© AdminController.java ×

1 package com.example.api_v01.controller;
2
3
4 > import ...
17
18 @RestController ▲ RobMaster28
19 @RequestMapping(ℳℳ"admin")
20 @RequiredArgsConstructor
21 Q public class AdminController {
22
23     private final AdminService adminservice;
24
25     @GetMapping(ℳℳ"/{id_admin}") ▲ RobMaster28
26     public ResponseEntity<?> getAdmin(@PathVariable UUID id_admin) throws NotFoundException {
27         SuccessMessage<?> successMessage = SuccessMessage.builder()
28             .status(HttpStatus.OK.value())
29             .message("Se encontro el admin")
30             .data(adminservice.findById(id_admin))
31             .build();
32         return ResponseEntity.ok(successMessage);
33     }
34
35     @GetMapping(ℳℳ"/list") ▲ RobMaster28
36     public ResponseEntity<?> getAdminList() { return ResponseEntity.ok(adminservice.findAll()); }
37
38 }
39
40 }
41 |
```

ArchingController

```
② ArchingController.java ×
1 package com.example.api_v01.controller;
2
3 > import ...
4
5
6 @RestController ← Sergio Antonio +2
7 @RequestMapping(@PathVariable "arching")
8 @RequiredArgsConstructor
9 public class ArchingController { //CONTROLADOR TESTEADO, LISTO PARA USAR
10
11
12
13
14     private final ArchingService archingService;
15     // Retorna todos los Arching
16     @GetMapping(@PathVariable "list") ← Sergio Antonio +1
17     public ResponseEntity<?> getAllArching() {
18         List<ArchingDTO> response = archingService.getAllArching();
19         return ResponseEntity.ok(
20             new SuccessMessage<>(HttpStatus.OK.value(), message: "Listado de Arching obtenido correctamente", response)
21         );
22     }
23
24
25     // Traer Arching por su ID
26     @GetMapping(@PathVariable "id_arching") ← Sergio Antonio +1
27     public ResponseEntity<?> getArchingById(
28         @PathVariable UUID id_arching
29     ) throws NotFoundException {
30         ArchingDTO response = archingService.getArchingDTOById(id_arching);
31         return ResponseEntity.ok(
32             new SuccessMessage<>(HttpStatus.OK.value(), message: "Arching obtenido correctamente", response)
33         );
34     }
35
36 }
```

ATMController

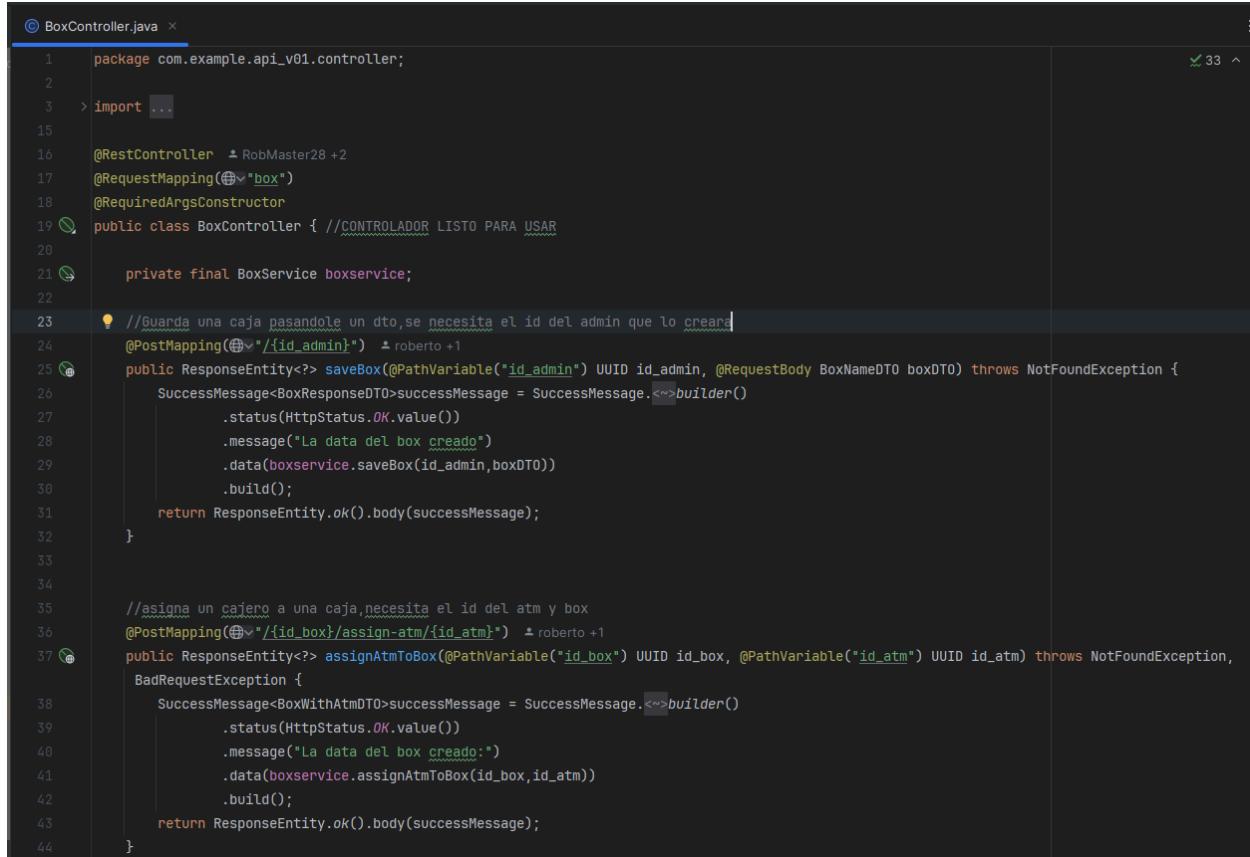


```
① package com.example.api_v01.controller;
②
③ > import ...
④
⑤ @RestController ▲ Sergio Antonio +2
⑥ @RequestMapping("atm")
⑦ @RequiredArgsConstructor
⑧ public class ATMController { //CONTROLADOR TESTEADO, LISTO PARA USAR
⑨
⑩     private final ATMService atmService;
⑪
⑫
⑬     //Recibe un DTO para registrar los datos del atm, se necesita el id del admin para guardar al atm
⑭     @PostMapping("/{adminId}") ▲ Sergio Antonio +1
⑮     public ResponseEntity<AtmResponseDTO> saveATM(@PathVariable("adminId") UUID adminId, @RequestBody AtmResponseDTO atmDTO) throws NotFoundException {
⑯         AtmResponseDTO createdATM = atmService.saveATM(adminId, atmDTO);
⑰         SuccessMessage<AtmResponseDTO> successMessage = SuccessMessage.<~>builder()
⑱             .status(HttpStatus.CREATED.value())
⑲             .message("ATM creado exitosamente")
⑳             .data(createdATM)
⑳             .build();
⑳
⑳         return ResponseEntity.status(HttpStatus.CREATED).body(successMessage);
⑳     }
⑳
⑳
⑳     //Recibe un DTO para registrar el user de un atm , se necesita el id del atm para asignarle el user
⑳     @PostMapping("/{atmId}/assign-user") ▲ Sergio Antonio +1
⑳     public ResponseEntity<AtmResponseDTO> assignUserATM(@PathVariable("atmId") UUID atmId, @RequestBody RegisterAtmUserDTO registerATMDTOUser) throws
⑳         NotFoundException {
⑳         AtmResponseDTO updatedATM = atmService.assingUserATM(atmId, registerATMDTOUser);
⑳         SuccessMessage<AtmResponseDTO> successMessage = SuccessMessage.<~>builder()
⑳             .status(HttpStatus.OK.value())
⑳             .message("Usuario asignado exitosamente al ATM")
⑳             .data(updatedATM)
⑳             .build();
⑳
⑳         return ResponseEntity.ok(successMessage);
⑳     }
⑳ }
```

AuthController

```
① AuthController.java ×
1 package com.example.api_v01.controller;
2
3 > import ...
11
12 @RequiredArgsConstructor ✎ RobMaster28
13 @RestController
14 @RequestMapping(BootApplication)
15 🐛 public class AuthController {
16
17     🐛     public final AuthService authService;
18
19     //Me va pedir el user y password de un usuario para identificarlo
20     @PostMapping("/Login") ✎ RobMaster28
21     🐛     public ResponseEntity<?> login(@RequestBody LoginDTO loginDTO) throws NotFoundException {
22         AuthResponse authResponse = authService.authenticate(loginDTO.getUsername(), loginDTO.getPassword());
23         return ResponseEntity.ok(authResponse);
24     }
25
26 }
```

BoxController



```
BoxController.java
1 package com.example.api_v01.controller;
2
3 > import ...
4
5 @RestController & RobMaster28 +2
6 @RequestMapping(@<>"box")
7 @RequiredArgsConstructor
8 public class BoxController { //CONTROLADOR LISTO PARA USAR
9
10    private final BoxService boxservice;
11
12    //Guarda una caja pasandole un dto, se necesita el id del admin que lo crea
13    @PostMapping(@<>"/id_admin") & roberto +1
14    public ResponseEntity<> saveBox(@PathVariable("id_admin") UUID id_admin, @RequestBody BoxNameDTO boxDTO) throws NotFoundException {
15        SuccessMessage<BoxResponseDTO>successMessage = SuccessMessage.<>builder()
16            .status(HttpStatus.OK.value())
17            .message("La data del box creado")
18            .data(boxservice.saveBox(id_admin,boxDTO))
19            .build();
20        return ResponseEntity.ok().body(successMessage);
21    }
22
23    //asigna un cajero a una caja, necesita el id del atm y box
24    @PostMapping(@<>"/{id_box}/assign-atm/{id_atm}") & roberto +1
25    public ResponseEntity<> assignAtmToBox(@PathVariable("id_box") UUID id_box, @PathVariable("id_atm") UUID id_atm) throws NotFoundException,
26    BadRequestException {
27        SuccessMessage<BoxWithAtmDTO>successMessage = SuccessMessage.<>builder()
28            .status(HttpStatus.OK.value())
29            .message("La data del box creado:")
30            .data(boxservice.assignAtmToBox(id_box,id_atm))
31            .build();
32        return ResponseEntity.ok().body(successMessage);
33    }
34
35    //asigna un cajero a una caja, necesita el id del atm y box
36    @PostMapping(@<>"/{id_box}/assign-atm/{id_atm}") & roberto +1
37    public ResponseEntity<> assignAtmToBox(@PathVariable("id_box") UUID id_box, @PathVariable("id_atm") UUID id_atm) throws NotFoundException,
38    BadRequestException {
39        SuccessMessage<BoxWithAtmDTO>successMessage = SuccessMessage.<>builder()
40            .status(HttpStatus.OK.value())
41            .message("La data del box creado:")
42            .data(boxservice.assignAtmToBox(id_box,id_atm))
43            .build();
44        return ResponseEntity.ok().body(successMessage);
45    }
46}
```

OrderSetController

```
OrderSetController.java × 1 2 3 > import ... 4 5 6 @RestController ▲ RobMaster28 +1 7 @RequestMapping(PathVariable("orderSet")) 8 @RequiredArgsConstructor 9 10 public class OrderSetController { //CONTROLADOR TESTEADO, LISTO PARA USAR 11 12 private final OrderSetOrchestratorService orderSetOrchestratorService; 13 14 15 //Guardar el nombre del cliente y la lista de pedidos de este en un orderSet pasandole el id del arching y el DTO SaveOrderSetListCustomerDTO 16 @PostMapping("/{id_arching}") ▲ RobMaster28 +1 17 public ResponseEntity<?> saveOrderSet(@PathVariable UUID id_arching,@RequestBody SaveOrderSetListCustomerDTO NameClientWithListOrdes) throws 18 NotFoundException, BadRequestException { 19     String nameClient = NameClientWithListOrdes.getName_cliente(); 20     List<CustomerOrderDTO> orders = NameClientWithListOrdes.getOrders(); 21     Tuple<OrderSetResponseDTO, UUID> tuple = orderSetOrchestratorService.saveCompleteOrderSet(id_arching,nameClient,orders); 22     URI location = UriGeneric.GenereURI(Ruta_GetByld_Entity: "/orderSet/{id_orderSet}",tuple.getSecond()); 23     SuccessMessage<?>.successMessage=SuccessMessage.builder() 24         .status(HttpStatus.OK.value()) 25         .message("Se guardo la lista de ordenes correctamente") 26         .data(tuple.getFirst()) 27         .build(); 28     return ResponseEntity.created(location).body(successMessage); 29 } 30 31 //Me devuelve los pedidos mediante el id del OrderSet 32 @GetMapping("/{id_orderSet}") ▲ RobMaster28 33 public ResponseEntity<?> getOrderSet(@PathVariable UUID id_orderSet) throws NotFoundException { 34     SuccessMessage<?>.successMessage=SuccessMessage.builder() 35         .status(HttpStatus.OK.value()) 36         .message("Lista de cliente encontrada") 37         .data(orderSetOrchestratorService.getOrderSetDTO(id_orderSet)) 38         .build(); 39 }
```

ProductController

```
① ProductController.java ×
1 package com.example.api_v01.controller;
2
3 > import ...
19
20 @RestController ▲ roberto +2
21 @RequestMapping("product")
22 @RequiredArgsConstructor
23 public class ProductController {    //CONTROLADOR TESTEADO, LISTO PARA USAR
24
25     private final ProductService productService;
26
27     //Busca todos los productos por categoría
28     @GetMapping("/list/category") ▲ roberto +1
29     public ResponseEntity<?> getAllProductsByCategory(@RequestParam Category category) throws NotFoundException{
30         SuccessMessage <List<ProductDTO>> successMessage = SuccessMessage.<~>builder()
31             .status(HttpStatus.OK.value())
32             .message("Lista de productos por categoría")
33             .data(productService.getProductByCategory(category))
34             .build();
35         return ResponseEntity.status(HttpStatus.OK).body(successMessage);
36     }
37     //Busca todos los productos que tengan el mismo nombre
38     @GetMapping("/list/name") ▲ roberto +1
39     public ResponseEntity<?> getAllProductsByName(@RequestParam String name) throws NotFoundException {
40         SuccessMessage <List<ProductDTO>> successMessage = SuccessMessage.<~>builder()
41             .status(HttpStatus.OK.value())
42             .message("Lista de productos por nombre")
43             .data(productService.getProductByName(name))
44             .build();
45         return ResponseEntity.status(HttpStatus.OK).body(successMessage);
46     }
}
```

StockController

```
① StockController.java ×
1 package com.example.api_v01.controller;
2
3 > import ...
19
20 @RestController  ↳ RobMaster28 +1
21 @RequestMapping("stock")
22 @RequiredArgsConstructor
23 ⚡ public class StockController { //CONTROLADOR TESTEADO, LISTO PARA USAR
24
25     private final ProductStockService productStockService;
26
27     //Me devuelve la lista de stock con los nombre del producto al que pertenecen
28     @GetMapping("/list")  ↳ RobMaster28 +1
29     public ResponseEntity<?> getAllStocks() {
30         SuccessMessage<List<ProductWithStockDTO>> successMessage = SuccessMessage.<~>builder()
31             .status(HttpStatus.OK.value())
32             .message("La lista de stock (!puede que la lista este vacia)")
33             .data(productStockService.getAllProductStock())
34             .build();
35         return ResponseEntity.ok(successMessage);
36     }
37
38     //Me devuelve la lista de stock con los nombre del producto al que pertenecen por la categoria del producto
39     @GetMapping("/list/category")  ↳ roberto +1
40     public ResponseEntity<?> getAllStocksCategory(@RequestParam Category category) throws NotFoundException {
41         SuccessMessage<List<ProductWithStockDTO>> successMessage = SuccessMessage.<~>builder()
42             .status(HttpStatus.OK.value())
43             .message("La lista de stock (!puede que la lista este vacia)")
44             .data(productStockService.getAllProductStockByCategory(category))
45             .build();
46         return ResponseEntity.ok(successMessage);
47     }
}
```

Modelo

User

```
public class User {  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "id_user")  
    private UUID id_user;  
  
    @Column(unique = true, nullable = false)  
    private String username;  
  
    @Column(nullable = false)  
    private String password;  
  
    @Enumerated(EnumType.STRING)  
    @Column(nullable = false)  
    public Rol role;  
}
```

Admin

```
public class Admin {  
|  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "id_admin")  
    private UUID id_admin;  
  
    @Column(nullable = false)  
    private String name_admin;  
  
    @Column(nullable = false)  
    private String email_admin;  
  
    @Column(nullable = false)  
    private String dni_admin;  
  
    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)  
    @JoinColumn(name = "id_user" , referencedColumnName = "id_user")  
    private User user_admin;  
}
```

Arqueo

```
public class Arching {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "id_arching")  
    private UUID id_arching;  
  
    @JsonFormat(pattern = "yyyy-MM-dd")  
    private LocalDate date;  
  
    @JsonFormat(pattern = "HH:mm:ss")  
    private LocalTime star_time;  
  
    @JsonFormat(pattern = "HH:mm:ss")  
    private LocalTime end_time;  
  
    private Double init_amount;  
  
    private Double final_amount;  
  
    private Double total_amount;  
  
    @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})  
    @JoinColumn(name = "id_box", referencedColumnName = "id_box")  
    private Box box;  
}
```

Activar
Ve a Conf

Cajero

```
public class ATM {  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "id_atm")  
    private UUID id_atm;  
  
    @Column(nullable = false)  
    private String name_atm;  
  
    @Column(nullable = false)  
    @JsonFormat(pattern = "yyyy-MM-dd")  
    private LocalDate date;  
  
    @Column(nullable = false)  
    private String alias;  
  
    @Column(nullable = false)  
    private String email;  
  
    @Column(nullable = false)  
    private String phone;  
  
    @Column(nullable = false)  
    private String dni;  
  
    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)  
    @JoinColumn(name = "id_user", referencedColumnName = "id_user")  
    private User user_atm;  
  
    @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})  
    @JoinColumn(name = "id_admin", referencedColumnName = "id_admin")  
}
```

Caja

```
public class Box {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id_box")
    private UUID id_box;

    @Column(nullable = false)
    private String name_box;

    @Column(nullable = false)
    @JsonFormat(pattern = "yyyy-MM-dd")
    private LocalDate date;

    @Column
    @Builder.Default
    private Boolean is_open = false;

    @OneToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
    @JoinColumn(name = "id_atm" ,referencedColumnName = "id_atm")
    private ATM atm;

    @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.PERSIST, CascadeType.MERGE})
    @JoinColumn(name = "id_admin" ,referencedColumnName = "id_admin")
    private Admin admin;
}
```

Orden de cliente

```
public class CustomerOrder {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "id_customer_order")  
    private UUID id_order;  
  
    @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.MERGE})  
    @JoinColumn(  
        name = "id_product",  
        referencedColumnName = "id_product"  
    )  
    private Product product;  
  
    private Integer count;  
  
    private Double total_rice;  
  
    @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.MERGE})  
    @JoinColumn(  
        name = "id_order_set",  
        referencedColumnName = "id_order_set"  
    )  
    private OrderSet order;  
}
```

Conjunto de Orden

```
public class OrderSet {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "id_order_set")  
    private UUID id_order_set;  
  
    @Column(nullable = false)  
    private String name_client;  
  
    @Column(nullable = false)  
    @JsonFormat(pattern = "yyyy-MM-dd")  
    private LocalDate date_order;  
  
    @Column(nullable = false)  
    @JsonFormat(pattern = "HH:mm:ss")  
    private LocalTime time_order;  
  
    @Column(nullable = true)  
    private Double total_order;  
  
    @ManyToOne(fetch = FetchType.LAZY, cascade = {CascadeType.MERGE , CascadeType.PERSIST, CascadeType.REFRESH})  
    @JoinColumn(name = "id_arching", referencedColumnName = "id_arching")  
    private Arching arching;  
  
    @Builder.Default  
    private Boolean active = false;  
  
    @Builder.Default  
    private Boolean dispatch = false;  
}
```

Activar W
Ve a Configu

Empleado

```
public class Employee { no usages ↗ RobMaster28

    private UUID id_atm; no usages

    private String name; no usages

    private LocalDate date; no usages

    private String alias; no usages

    private String email; no usages

    private String phone; no usages

    private String dni; no usages

    private Admin admin; no usages

}
```

Producto

```
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id_product")
    private UUID id_product;

    private String name_product;

    private Double price;

    private String additional_observation;

    @Enumerated(EnumType.STRING)
    private Category category;

    @OneToOne(
        cascade = {CascadeType.ALL},
        fetch = FetchType.EAGER
    )
    @JoinColumn(
        name = "id_product_stock",
        referencedColumnName = "id_product"
    )
    private ProductStock stock;

    @ManyToOne(
        cascade = {CascadeType.PERSIST, CascadeType.MERGE},
        fetch = FetchType.LAZY
    )
}
```

Activar
Ve a Con

Stock

```
public class ProductStock {  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "id_product_stock")  
    private UUID id_product_stock;  
  
    private Integer ini_stock;  
  
    private Integer current_stock;  
  
    private Integer total_sold;  
}
```

Vista

Estructura

```
✓ SISTEMA-RESTAURANTE
  > .next
  > app
  > components
  > context
  > hooks
  > node_modules
  > public
  > services
  > styles
  > types
  > utils
  ♦ .gitignore
  JS eslint.config.mjs
  TS middleware.ts
  TS next-env.d.ts
  TS next.config.ts
  {} package-lock.json
  {} package.json
  ⓘ README.md
  TS tsconfig.json
```

Login form

The screenshot shows a code editor interface with a dark theme. On the left is a tree view of the project structure:

- SISTEMA-RESTAURANTE
 - app
 - graficos
 - inventario
 - logout
 - layout.tsx
 - page.tsx
 - logros
 - pedido
 - productos
 - # globals.css
 - layout.tsx
 - page.tsx
 - components
 - apertura-cierre
 - arqueo
 - charts
 - cocina
 - common
 - empleados
 - layout
 - pedido
 - providers
 - login-form.tsx
 - context
 - hooks
 - node_modules
 - public
 - services
 - styles
 - types
 - utils
 - .gitignore
 - eslint.config.mjs
 - middleware.ts
 - next-env.d.ts
 - next.config.ts
 - package-lock.json
 - package.json
 - README.md

The right pane displays the content of `login-form.tsx`:

```
1  "use client";
2
3  import { useState, type FormEvent, type ChangeEvent, useEffect } from "react";
4  import { useAuth } from "@/hooks/useAuth";
5  // Asegurarnos de importar los estilos correctos
6  import "@/styles/login.css";
7
8  interface FormData {
9    username: string;
10   password: string;
11 }
12
13 Tabnine | Edit | Test | Explain | Document
14 export default function LoginForm() {
15   const { login, isLoading, error } = useAuth();
16   const [formData, setFormData] = useState<FormData>({
17     username: "",
18     password: ""
19   });
20
21   useEffect(() => {
22     // Log para depuración
23     console.log("Estado de isLoading:", isLoading);
24   }, [isLoading]);
25
26   const handleChange = (
27     e: ChangeEvent<HTMLInputElement | HTMLSelectElement>
28   ) => {
29     const { name, value } = e.target;
30     setFormData((prev) => ({
31       ...prev,
32       [name]: value,
33     }));
34   };
35
36   const handleSubmit = async (e: FormEvent) => {
37     e.preventDefault();
38     console.log("Intentando iniciar sesión con:", formData);
39     try {
40       await login(formData.username, formData.password);
41     } catch (err) {
42       console.error("Error de inicio de sesión:", err);
43     }
44   };
}
```

Product mtble

SISTEMA-RESTAURANTE

```

app > productos > components > ProduktoTable.tsx > ProduktoTable > produkto.map() callback
  1 "use client"
  2
  3 import { useEffect, useState } from "react"
  4 import Cookies from "js-cookie"
  5 import useProductos from "../hooks/useProductos"
  6 import { Producto } from "../types/producto"
  7 import { createProducto, deleteProducto, updateProducto } from "../services/productoService"
  8 import DataTable from "@/components/common/DataTable"
  9 import ModalPortal from "@/components/common/ModalPortal"
10 import { toast, ToastContainer } from "react-toastify"
11 import Swal from "sweetalert2"
12 import withReactContent from "sweetalert2-react-content"
13 import "react-toastify/dist/ReactToastify.css"
14
15 // Inicializa SweetAlert con React
16 const MySwal = withReactContent(Swal)
17
18 export default function ProduktoTable() {
19     // Token para autenticación
20     const [token, setToken] = useState<string | null>(null)
21
22     // Estado del formulario de producto (crear o editar)
23     const [formData, setFormData] = useState<Partial<Producto>>({})
24
25     // Hook para obtener productos y estado de carga
26     const { productos, loading, setProductos } = useProductos(token)
27
28     // Obtener token del localStorage o cookie al montar
29     useEffect(() => {
30         const localToken = typeof window !== "undefined" ? localStorage.getItem("authToken") : null
31         const cookieToken = Cookies.get("token")
32         setToken(localToken || cookieToken || null)
33     }, [])
34
35     // Abrir modal y llenar datos si es edición
36     const openModal = (producto: Producto | null = null) => {
37         setFormData(producto || {})
38         const modalEl = document.getElementById("productoModal")
39         modalEl && new (window as any).bootstrap.Modal(modalEl).show()
40     }
41
42     // Manejar cambios en inputs del formulario
43     const handleChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement>) => {
44         setFormData({ ...formData, [e.target.name]: e.target.value })
45     }

```

Tabla inventario

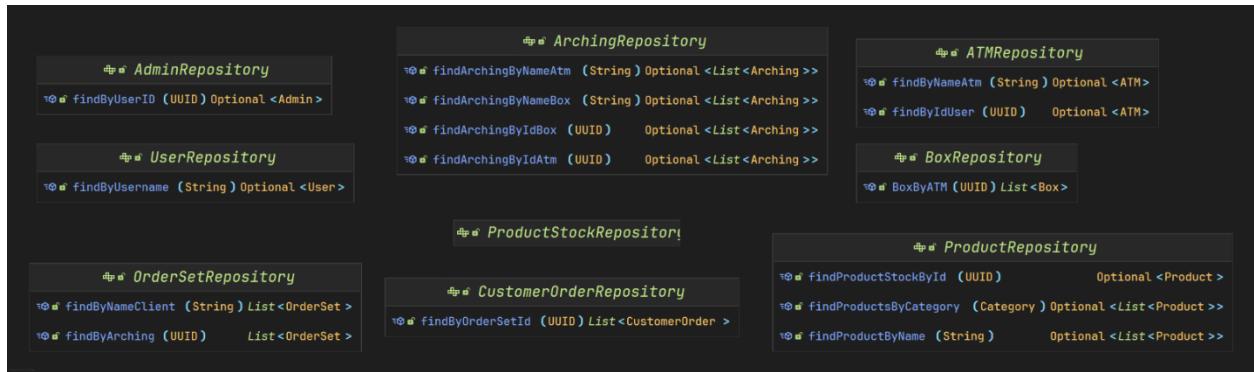
The screenshot shows a code editor with two panes. The left pane displays a file tree for a project named 'SISTEMA-RESTAURANTE'. The right pane shows the source code for a file named 'InventarioTable.tsx'.

File Tree (Left):

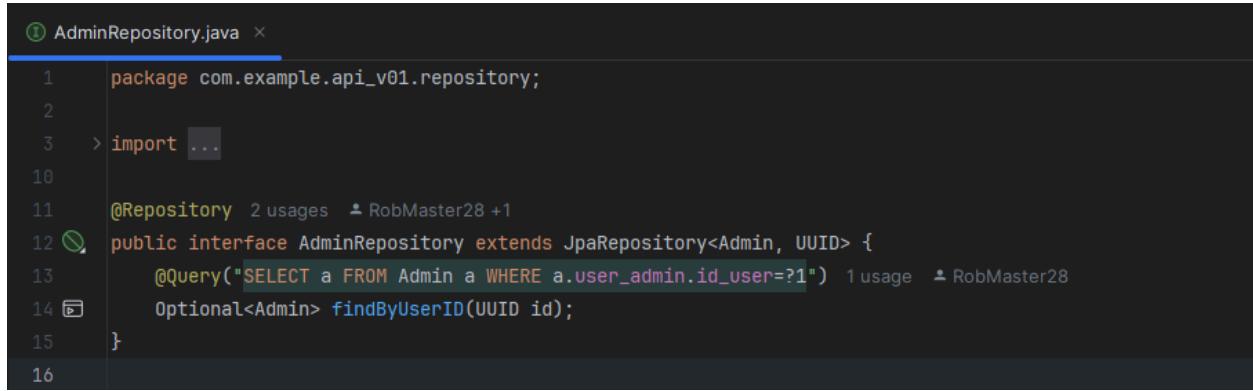
- > .next
- < app
 - < auth
 - layout.tsx
 - > apertura-cierre
 - > arqueo
 - > cocina
 - > consolidados
 - > empleados
 - > graficos
- < inventario
 - < components
 - StockModal.tsx
 - StockModalContent.tsx
 - InventarioTable.tsx
 - < common
 - < modals
- < hooks
 - useProductoStock.ts
- < services
 - productStockService.ts
- < types
 - productoStok.ts
 - layout.tsx
 - page.tsx
 - > logout
 - > logros
 - > pedido
 - > productos
 - # globals.css
 - layout.tsx
 - page.tsx
 - > components
 - > context
 - > hooks
 - > node_modules
 - > public
 - > services
 - > styles
 - > types

```
app > inventario > components > InventarioTable.tsx > ...
1  "use client"
2
3  import { useContext, useState } from "react"
4  import useProductoStock from "../hooks/useProductoStock"
5  import { ProductStock } from "../types/productoStok"
6  import {
7    updateIncrementStockProduct,
8    updateDecrementStockProduct,
9    resetStockProduct
10   } from "../services/productStockService"
11  import DataTable from "@/components/common/DataTable"
12  import { toast, ToastContainer } from "react-toastify"
13  import Swal from "sweetalert2"
14  import withReactContent from "sweetalert2-react-content"
15  import "react-toastify/dist/ReactToastify.css"
16  import StockModalContent from "./modals/StockModalContent"
17  import { AuthContext } from "@/context/AuthContext"
18
19 // Inicialización de SweetAlert con React
20 const MySwal = withReactContent(Swal)
21
22 Tabnine | Edit | Test | Fix | Explain | Document
23 export default function InventarioTable() {
24   // Obtenemos el token del usuario desde el contexto de autenticación
25   const { token } = useContext(AuthContext)
26
27   // Estados locales
28   const [formData, setFormData] = useState<Partial<ProductStock>>({})
29   const [cantidad, setCantidad] = useState<number>(0)
30   const [modalType, setModalType] = useState<"increment" | "decrement">("increment")
31   const [isModalOpen, setIsModalOpen] = useState(false)
32
33   // Hook personalizado para obtener productos y gestionar estado de carga
34   const { productos, loading, setProductos } = useProductoStock(token)
35
36   /**
37    * Abre el modal con los datos del producto y tipo de operación (entrada o salida)
38    * @param producto - Objeto del producto seleccionado
39    * @param type - Tipo de operación: "increment" o "decrement"
40   */
41   const openModal = (producto: ProductStock, type: "increment" | "decrement") => {
42     setFormData(producto)
43     setModalType(type)
44     setCantidad(0)
45     setIsModalOpen(true)
46 }
```

Repositorio



AdminRepository



```

① AdminRepository.java ×
1 package com.example.api_v01.repository;
2
3 > import ...
10
11 @Repository 2 usages ▲ RobMaster28 +1
12 public interface AdminRepository extends JpaRepository<Admin, UUID> {
13     @Query("SELECT a FROM Admin a WHERE a.user_admin.id_user=?1") 1 usage ▲ RobMaster28
14     Optional<Admin> findByUserID(UUID id);
15 }
16

```

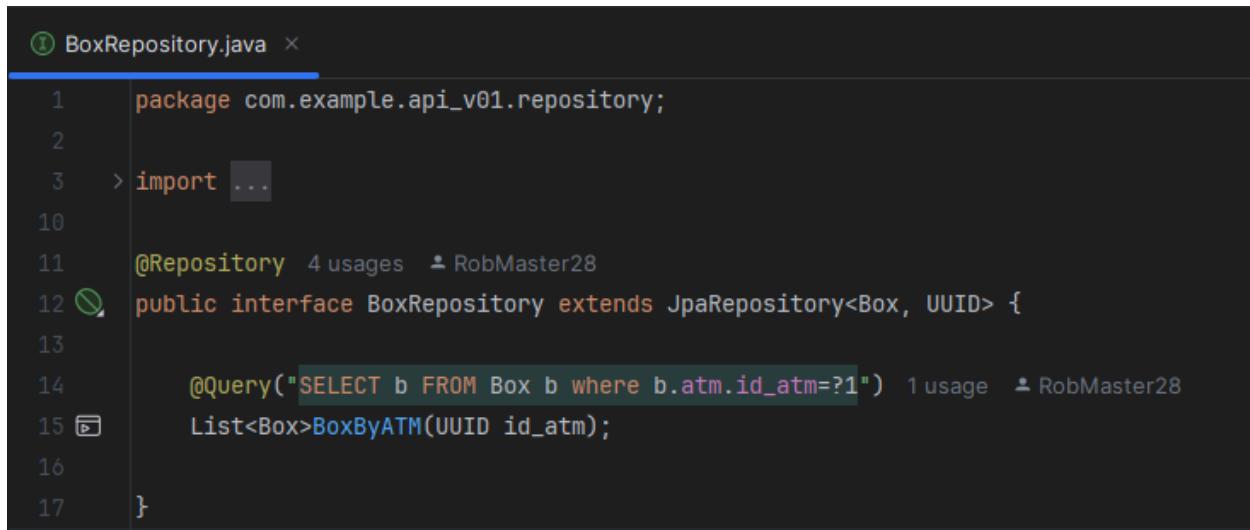
ArchingRepository

```
① ArchingRepository.java ×
1 package com.example.api_v01.repository;
2
3 > import ...
11
12 @Repository 2 usages  ↗ roberto +1
13 ⚡ public interface ArchingRepository extends JpaRepository<Arching, UUID> {
14
15     @Query("SELECT a FROM Arching a WHERE a.box.id_box=?1") 1 usage  ↗ roberto
16 ⚡         Optional<List<Arching>> findArchingByIdBox(UUID id_box);
17
18     @Query("SELECT a FROM Arching a WHERE a.box.name_box=?1") 1 usage  ↗ roberto
19 ⚡         Optional<List<Arching>> findArchingByNameBox(String name_box);
20
21     @Query("SELECT a FROM Arching a WHERE a.box.atm.id_atm=?1") 1 usage  ↗ roberto
22 ⚡         Optional<List<Arching>> findArchingByIdAtm(UUID id_atm);
23
24     @Query("SELECT a FROM Arching a WHERE a.box.atm.name_atm=?1") 1 usage  ↗ roberto
25 ⚡         Optional<List<Arching>> findArchingByNameAtm(String name_atm);
26 }
```

ATMRepository

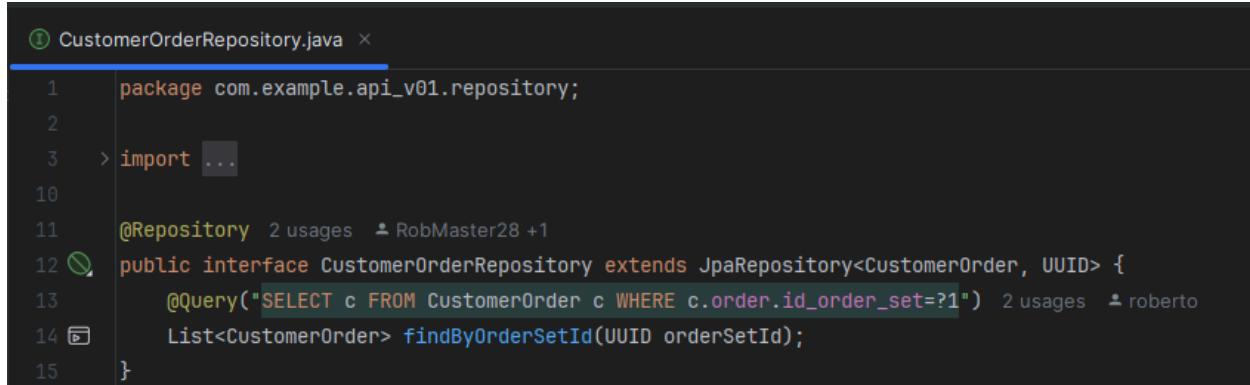
```
① ATMRepository.java ×
1 package com.example.api_v01.repository;
2
3 > import ...
10
11 @Repository 2 usages  ↗ RobMaster28 +1
12 ⚡ public interface ATMRepository extends JpaRepository<ATM, UUID> {
13
14     // Metodo para buscar el ATM por nombre, alias o dni
15
16     @Query("SELECT a FROM ATM a WHERE a.name_atm=?1") 1 usage  ↗ Sergio Antonio
17 ⚡         Optional<ATM> findByNameAtm(String nameAtm);
18
19
20     @Query("SELECT a FROM ATM a WHERE a.user_atm.id_user=?1") 1 usage  ↗ RobMaster28
21 ⚡         Optional<ATM> findByIdUser(UUID id_user);
22 }
```

BoxRepository



```
BoxRepository.java
1 package com.example.api_v01.repository;
2
3 > import ...
10
11 @Repository 4 usages  ± RobMaster28
12 🐛 public interface BoxRepository extends JpaRepository<Box, UUID> {
13
14     @Query("SELECT b FROM Box b where b.atm.id_atm=?1") 1 usage  ± RobMaster28
15     List<Box> BoxByATM(UUID id_atm);
16
17 }
```

CostumerOrderRepository



```
CustomerOrderRepository.java
1 package com.example.api_v01.repository;
2
3 > import ...
10
11 @Repository 2 usages  ± RobMaster28 +1
12 🐛 public interface CustomerOrderRepository extends JpaRepository<CustomerOrder, UUID> {
13     @Query("SELECT c FROM CustomerOrder c WHERE c.order.id_order_set=?1") 2 usages  ± roberto
14     List<CustomerOrder> findByOrderSetId(UUID orderSetId);
15 }
```

OrderSetRepository

```
① OrderSetRepository.java ×  
1 package com.example.api_v01.repository;  
2  
3 > import ...  
10  
11 @Repository 2 usages  ↳ RobMaster28 +1  
12 🐛 public interface OrderSetRepository extends JpaRepository<OrderSet, UUID> {  
13  
14     @Query("SELECT o FROM OrderSet o WHERE o.name_client=?1") 1 usage  ↳ RobMaster28  
15     List<OrderSet> findByNameClient(String name_client);  
16  
17     @Query("SELECT o FROM OrderSet o WHERE o.arching.id_arching=?1 ") 2 usages  ↳ roberto  
18     List<OrderSet> findByArching(UUID id_arching);  
19  
20 }
```

ProductRepository

```
① ProductRepository.java ×  
1 package com.example.api_v01.repository;  
2  
3 > import ...  
13  
14 @Repository 5 usages  ↳ roberto +1  
15 🐛 public interface ProductRepository extends JpaRepository<Product, UUID> {  
16  
17     @Query("SELECT p FROM Product p WHERE p.category=?1") 2 usages  ↳ roberto  
18     Optional<List<Product>> findProductsByCategory(Category category);  
19  
20     @Query("SELECT p FROM Product p WHERE p.name_product=?1") 2 usages  ↳ roberto  
21     Optional<List<Product>> findProductByName(String name);  
22  
23     @Query("SELECT p FROM Product p WHERE p.stock.id_product_stock=?1") 6 usages  ↳ roberto  
24     Optional<Product> findProductStockById(UUID id_stock);  
25  
26 }
```

ProductStockRepository

```
① ProductStockRepository.java ×  
1 package com.example.api_v01.repository;  
2  
3 > import ...  
4  
10 @Repository 2 usages  ↳ RobMaster28  
11 ② public interface ProductStockRepository extends JpaRepository<ProductStock, UUID> {  
12  
13 }
```

UserRepository

```
① UserRepository.java ×  
1 package com.example.api_v01.repository;  
2  
3 > import ...  
4  
11 @Repository 4 usages  ↳ RobMaster28 +1  
12 ② public interface UserRepository extends JpaRepository<User, UUID> {  
13     @Query("SELECT u FROM User u WHERE u.username=?1") 2 usages  ↳ Sergio Antonio  
14     Optional<User> findByUsername(String username);  
15  
16 }
```

Funcionalidades

Servicios



ATMService	
<code>getAllATMs ()</code>	<code>List<AtmDTO></code>
<code>saveATM (UUID , AtmResponseDTO)</code>	<code>AtmResponseDTO</code>
<code>getAtmByName (String)</code>	<code>AtmDTO</code>
<code>updateATM (UUID , AtmResponseDTO)</code>	<code>AtmResponseDTO</code>
<code>deleteATM (UUID)</code>	<code>void</code>
<code>getAtmByUser (UUID)</code>	<code>AtmDTO</code>
<code>getAtmById (UUID)</code>	<code>AtmDTO</code>
<code>getAtmEntityById (UUID)</code>	<code>ATM</code>
<code>assingUserATM (UUID , RegisterAtmUserDTO)</code>	<code>AtmResponseDTO</code>

⊕ ArchingService

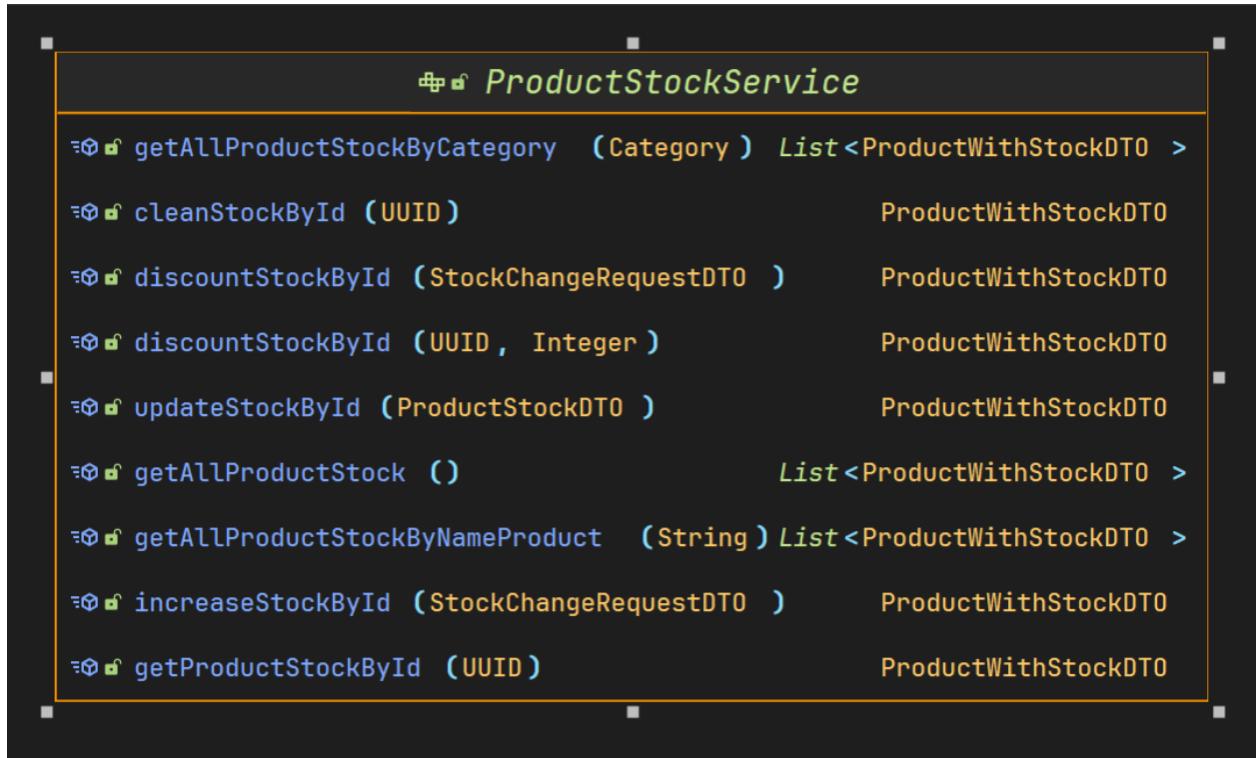
```
=@ getArchingById (UUID) Arching
=@ getArchingByNameATM (String) List<ArchingWithAtmDTO >
=@ saveArchingResponseDTO (UUID, ArchingInitDTO ) Tuple<ArchingResponseDTO , UUID>
=@ getArchingByATM (UUID) List<ArchingWithAtmDTO >
=@ getAllArching () List<ArchingDTO >
=@ getArchingByBox (UUID) List<ArchingWithBoxDTO >
=@ saveArching (Arching) Arching
=@ getArchingDTOById (UUID) ArchingDTO
=@ getArchingByNameBox (String) List<ArchingWithBoxDTO >
```

⊕ CustomerOrderService

```
=@ saveCustomerOrder (UUID , UUID , Integer ) CustomerOrder
=@ listCustomerOrdersByOrderSet (UUID) List<CustomerOrder >
=@ TotalAmountOrderSet (UUID) Double
```

ProductService	
<code>=@ getProductName (String)</code>	<code>List<ProductDTO></code>
<code>=@ updateProduct (UUID, ProductResponseDTO)</code>	<code>ProductResponseDTO</code>
<code>=@ getProducts ()</code>	<code>List<ProductDTO></code>
<code>=@ getProductByCategory (Category)</code>	<code>List<ProductDTO></code>
<code>=@ getProductDTO (UUID)</code>	<code>ProductDTO</code>
<code>=@ deleteProduct (UUID)</code>	<code>void</code>
<code>=@ getProduct (UUID)</code>	<code>Product</code>
<code>=@ saveProduct (UUID, ProductResponseDTO)</code>	<code>Tuple</code>

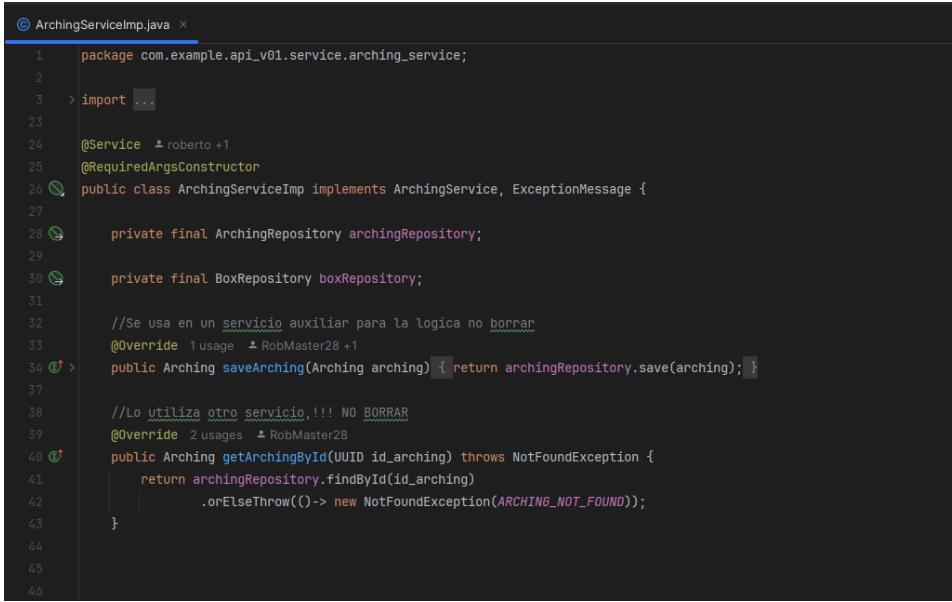
BoxService	
=@ [] toggleBoxDeactivationStatus (UUID , UUID)	BoxWithArchingDTO
=@ [] getBoxesByAtm (UUID)	List < BoxDTO >
=@ [] getBoxes ()	List < BoxDTO >
=@ [] saveBox (UUID , BoxNameDTO)	BoxResponseDTO
=@ [] toggleBoxActiveStatus (UUID , ArchingInitDTO)	BoxResponseWithArchingDTO
=@ [] getBoxInfo (UUID)	BoxDTO
=@ [] assignAtmToBox (UUID , UUID)	BoxWithAtmDTO
=@ [] getBox (UUID)	Box



UserService`loadUserByUsername (String) UserDetails``save (User) void`**AuthService**`authenticate (String , String) AuthResponse <?>`**AdminServiceImp**

```
② AdminServiceImp.java ×
1 package com.example.api_v01.service.admin_service;
2
3 > import ...
4
5 < @Service ▲ RobMaster28 +1
6 @RequiredArgsConstructor
7 public class AdminServiceImp implements AdminService , ExceptionMessage {
8
9     private final AdminRepository adminRepository;
10    private final PasswordEncoder passwordEncoder;
11
12    //Solo se usa una vez para crear al admin
13    @Override 1 usage ▲ RobMaster28
14    public Admin saveAdmin(RegisterAdmin registerAdmin) {
15
16        User user = User.builder()
17            .role(Rol.ADMIN)
18            .username(registerAdmin.getUsername())
19            .password(passwordEncoder.encode(registerAdmin.getPassword()))
20            .build();
21
22        Admin admin = Admin.builder()
23            .name_admin(registerAdmin.getName())
24            .email_admin(registerAdmin.getEmail())
25            .dni_admin(registerAdmin.getDni())
26            .user_admin(user)
27            .build();
28
29        return adminRepository.save(admin);
30
31    }
32
33 }
```

ArchingService



```
① ArchingServiceImp.java ×
1 package com.example.api_v01.service.arching_service;
2
3 > import ...
4
5 @Service ▲ roberto +1
6 @RequiredArgsConstructor
7 public class ArchingServiceImp implements ArchingService, ExceptionMessage {
8
9     private final ArchingRepository archingRepository;
10
11     private final BoxRepository boxRepository;
12
13     //Se usa en un servicio auxiliar para la logica no borrar
14     @Override 1 usage ▲ RobMaster28 +1
15     public Arching saveArching(Arching arching) { return archingRepository.save(arching); }
16
17     //Lo utiliza otro servicio, !!! NO BORRAR
18     @Override 2 usages ▲ RobMaster28
19     public Arching getArchingById(UUID id_arching) throws NotFoundException {
20         return archingRepository.findById(id_arching)
21             .orElseThrow(() -> new NotFoundException(ARCHING_NOT_FOUND));
22     }
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
```

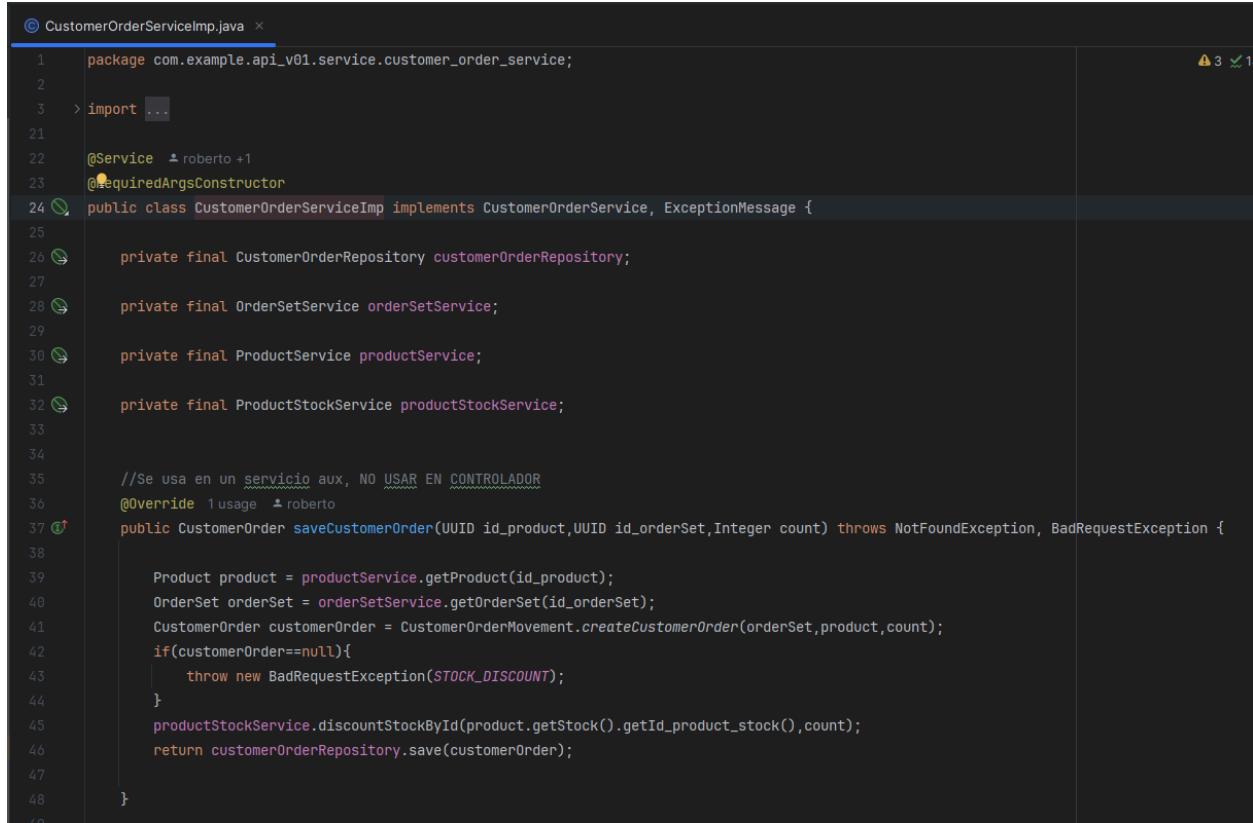
ATMServiceImp

```
① ATMServiceImp.java ×
1 package com.example.api_v01.service.atm_service;
2
3 > import ...
4
5 @Service ▲ RobMaster28 +1
6 @RequiredArgsConstructor
7 public class ATMServiceImp implements ATMService, ExceptionMessage {
8
9     private final ATMRepository atmRepository;
10    private final AdminService adminService;
11    private final PasswordEncoder passwordEncoder;
12
13    @Override 1 usage ▲ RobMaster28 +1
14    public AtmResponseDTO saveATM(UUID id_admin, AtmResponseDTO atm) throws NotFoundException { //Funciona Bien
15        Admin admin = adminService.findById(id_admin);
16        ATM savedATM = atmRepository.save(ATMMovement.saveATM(atm,admin));
17        return ATMMovement.convertToResponseDTO(savedATM);
18    }
19
20    @Override 1 usage ▲ RobMaster28 +1
21    public AtmResponseDTO assingUserATM(UUID id_atm, RegisterAtmUserDTO atm) throws NotFoundException { //Funciona bien
22
23        ATM atmOptional = atmRepository.findById(id_atm)
24            .orElseThrow(() -> new NotFoundException(ExceptionMessage.ATM_NOT_FOUND));
25
26        User user = User.builder()
27            .role(Rol.ATM)
28            .username(atm.getUsername())
29            .password(passwordEncoder.encode(atm.getPassword()))
30            .build();
31
32        atmOptional.setUser_atm(user);
33
34        ATM savedATM = atmRepository.save(atmOptional);
```

BoxServiceImp

```
① BoxServiceImp.java ×
1 package com.example.api_v01.service.box_service;
2
3 > import ...
4
5 @Service + RobMaster28+2
6 @RequiredArgsConstructor
7 public class BoxServiceImp implements BoxService, ExceptionMessage {
8
9     private final BoxRepository boxRepository;
10    private final ATMService atmService;
11    private final AdminService adminService;
12
13    private final ArchingProcessOrderSet archingProcessOrderSet;
14
15    @Override 1 usage + roberto +1
16    public BoxResponseDTO saveBox(UUID id_admin, BoxNameDTO box) throws NotFoundException {
17        Admin admin = adminService.findById(id_admin);
18        Box newBox = boxRepository.save(BoxMovement.CreateBox(admin, box));
19        return BoxMovement.CreateBoxResponseDTO(newBox);
20    }
21
22
23    @Override 1 usage + roberto +1
24    public BoxResponseWithArchingDTO toggleBoxActiveStatus(UUID id_box, ArchingInitDTO archingInitDTO) throws NotFoundException,
25        BadRequestException {
26
27        Box box = boxRepository.findById(id_box)
28            .orElseThrow( () -> new NotFoundException(BOX_NOT_FOUND));
29
30        if(box.getIs_open() == true) {
31            throw new BadRequestException(BOX_OPEN);
32        }
33
34        box.setIs_open(true);
35
36        boxRepository.save(box);
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
```

CustomerOrderServiceImp



```
CustomerOrderServiceImp.java
1 package com.example.api_v01.service.customer_order_service;
2
3 > import ...
4
5 @Service + roberto +1
6 @RequiredArgsConstructor
7 public class CustomerOrderServiceImp implements CustomerOrderService, ExceptionMessage {
8
9     private final CustomerOrderRepository customerOrderRepository;
10
11     private final OrderSetService orderSetService;
12
13     private final ProductService productService;
14
15     private final ProductStockService productStockService;
16
17
18     //Se usa en un servicio aux, NO USAR EN CONTROLADOR
19     @Override 1 usage + roberto
20     public CustomerOrder saveCustomerOrder(UUID id_product,UUID id_orderSet,Integer count) throws NotFoundException, BadRequestException {
21
22         Product product = productService.getProduct(id_product);
23         OrderSet orderSet = orderSetService.getOrderSet(id_orderSet);
24         CustomerOrder customerOrder = CustomerOrderMovement.createCustomerOrder(orderSet,product,count);
25         if(customerOrder==null){
26             throw new BadRequestException(STOCK_DISCOUNT);
27         }
28         productStockService.discountStockById(product.getStock().getId_product_stock(),count);
29         return customerOrderRepository.save(customerOrder);
30     }
31
32 }
```

OrderSetServiceImp

```
OrderSetServiceImp.java ×

19
20 @Service ▲ RobMaster28 +1
21 @RequiredArgsConstructor
22 public class OrderSetServiceImp implements OrderSetService, ExceptionMessage {
23
24     private final OrderSetRepository orderSetRepository;
25     private final ArchingService archingService;
26
27     //Lo utiliza para guardar el OrderSet junto con sus ordenes, es usado en un servicio aux
28     //Se utiliza para un servicio no para controlador NO USAR EN CONTROLADOR
29
30
31     @Override ▲ RobMaster28
32     public OrderSet save(OrderSet orderSet) { return orderSetRepository.save(orderSet); }
33
34
35     @Override 1 usage ▲ RobMaster28 +1
36     public OrderSet saveBaseOrderSet(UUID id_arching, String name_client) throws NotFoundException, BadRequestException {
37         Arching arching = archingService.getArchingById(id_arching);
38         if(arching.getEnd_time() != null){
39             throw new BadRequestException(ARCHING_FINISHED);
40         }
41         return orderSetRepository.save(OrderSetMovement.CreateOrderSet(arching, name_client));
42     }
43
44     //Se utiliza para un servicio no para un controlador NO USAR EN CONTROLADOR
45     //Poner una advertencia en el controlador encaso de que la devolucion sea 0.0 (La lista es vacia)
46     @Override 1 usage ▲ roberto
47     public Double totalAmountArching(UUID id_arching) {
48         return orderSetRepository.findByArching(id_arching).List<OrderSet>
49             .stream() Stream<OrderSet>
50             .map( OrderSet orderSet -> orderSet.getTotal_order()) Stream<Double>
51             .reduce( identity: 0.0, Double::sum);
52     }
53 }
54
```

ProductServiceImp

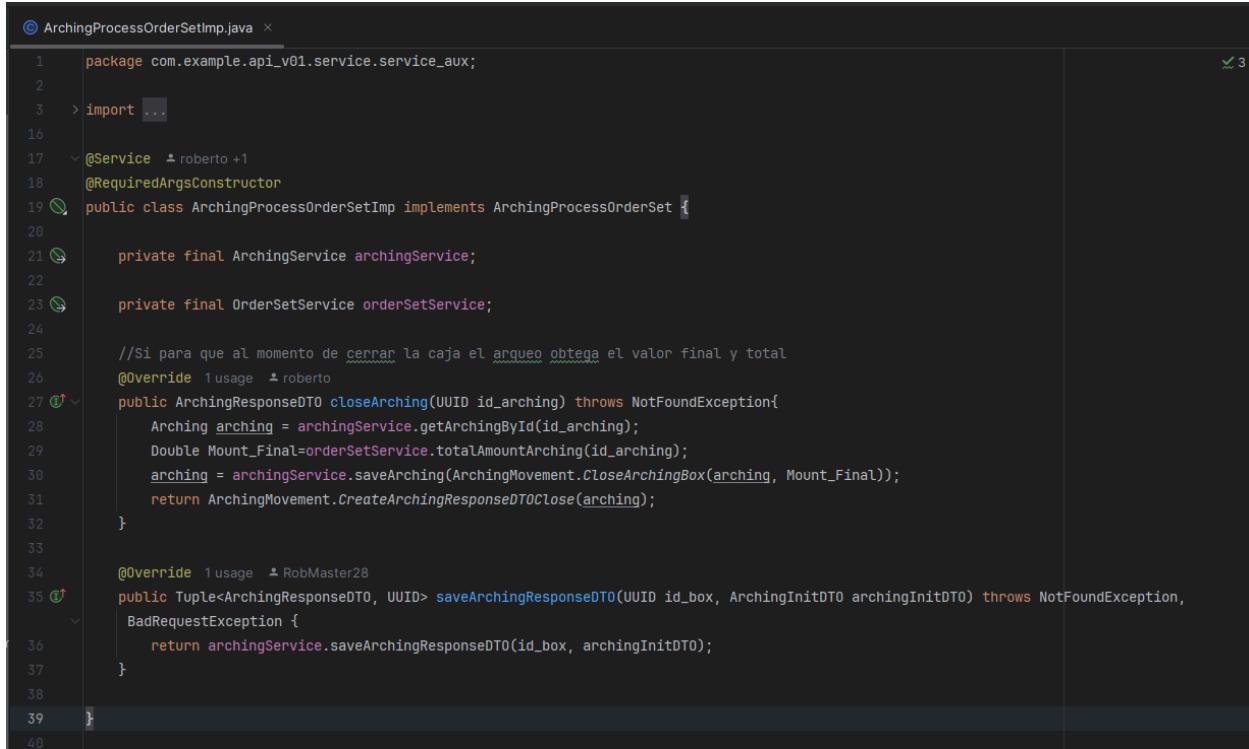
```
① ProductServiceImp.java ×
1 package com.example.api_v01.service.product_service;
2
3 > import ...
4
5 < @Service ▲ RobMaster28 +2
6 @RequiredArgsConstructor
7 public class ProductServiceImp implements ProductService , ExceptionMessage {
8
9     private final ProductRepository productRepository;
10    private final AdminService adminService;
11
12    @Override 1 usage ▲ roberto +1
13    public Tuple<ProductResponseDTO,UUID> saveProduct(
14        UUID id_admin,
15        ProductResponseDTO productDTO
16    ) throws NotFoundException {
17        Admin admin = adminService.findById(id_admin);
18        Product productAdd = productRepository.save(ProductMovement.createProductAndStock(admin,productDTO));
19        return new Tuple<>(ProductMovement.moveProductResponseDTO(productAdd),productAdd.getId_Product());
20    }
21
22    @Override 1 usage ▲ RobMaster28
23    public void deleteProduct(UUID id) throws NotFoundException {
24        if(!productRepository.existsById(id)){
25            throw new NotFoundException(PRODUCT_NOT_FOUND);
26        }
27        productRepository.deleteById(id);
28    }
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
```

ProductStockServiceImp

```
© ProductStockServiceImp.java ×

1 package com.example.api_v01.service.product_stock_service;
2
3 > import ...
21
22
23 @Service ▲ roberto +1
24 @RequiredArgsConstructor
25 Q public class ProductStockServiceImp implements ProductStockService, ExceptionMessage {
26
27     private final ProductStockRepository productStockRepository;
28     private final ProductRepository productRepository;
29
30     @Override 1 usage ▲ roberto
31     public ProductWithStockDTO getProductStockById(UUID id_productStock) throws NotFoundException {
32         Product product = productRepository.findProductStockById(id_productStock)
33             .orElseThrow( () -> new NotFoundException(STOCK_NOT_FOUND));
34         return InventoryMovement.getProductWithStockDTO(product);
35     }
36
37     @Override 1 usage ▲ roberto
38     public List<ProductWithStockDTO> getAllProductStock() {
39         return InventoryMovement.getListProductWithStock(productRepository.findAll());
40     }
41
42     @Override 1 usage ▲ roberto
43     public List<ProductWithStockDTO> getAllProductStockByCategory(Category category) throws NotFoundException {
44         List<Product>ListProductByCategory=productRepository.findProductsByCategory(category)
45             .orElseThrow( () -> new NotFoundException(CATEGORY_NOT_FOUND) );
46         return InventoryMovement.getListProductWithStock(ListProductByCategory);
47     }
48 }
```

ArchingProcessOrderSetImp

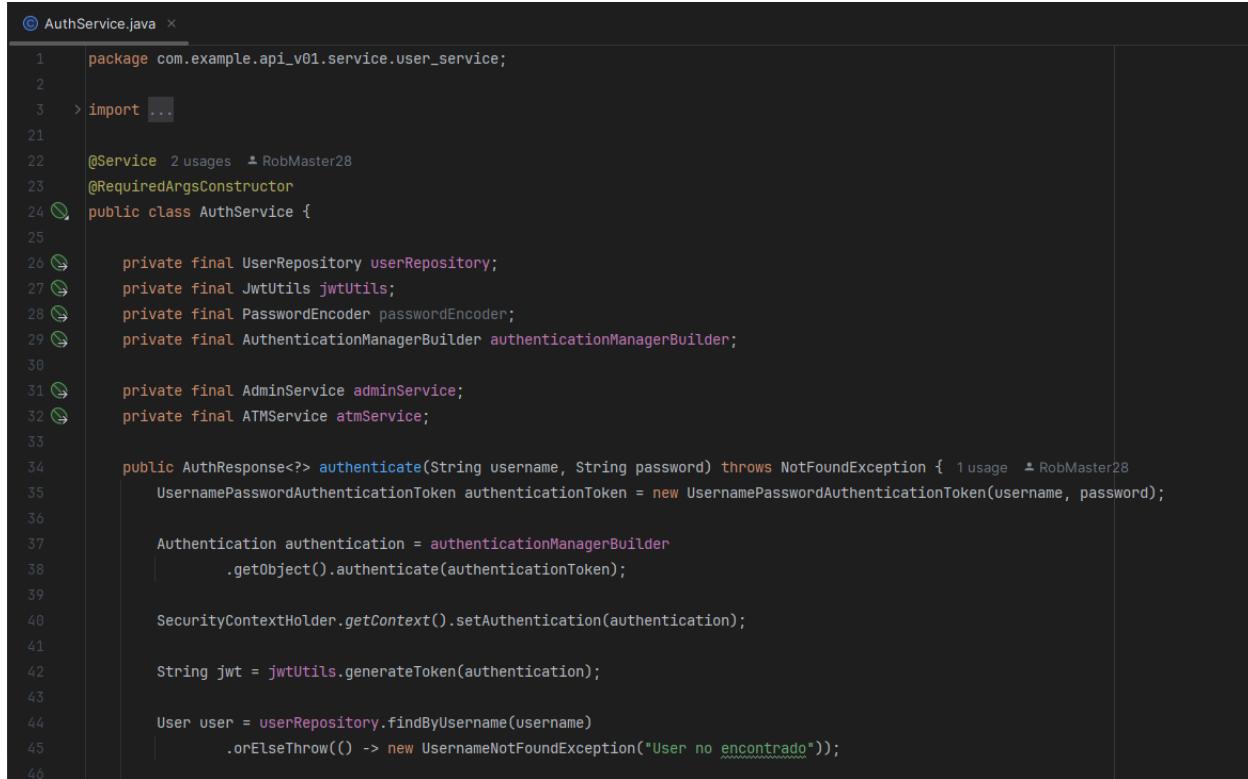


```
① ArchingProcessOrderSetImp.java x
1 package com.example.api_v01.service.service_aux;
2
3 > import ...
4
5 @Service ▲ roberto +1
6 @RequiredArgsConstructor
7 public class ArchingProcessOrderSetImp implements ArchingProcessOrderSet {
8
9     private final ArchingService archingService;
10
11     private final OrderSetService orderSetService;
12
13     //Si para que al momento de cerrar la caja el arqueo obtenga el valor final y total
14     @Override 1 usage ▲ roberto
15     public ArchingResponseDTO closeArching(UUID id_arching) throws NotFoundException{
16         Arching arching = archingService.getArchingById(id_arching);
17         Double Mount_Final=orderSetService.totalAmountArching(id_arching);
18         arching = archingService.saveArching(ArchingMovement.CloseArchingBox(arching, Mount_Final));
19         return ArchingMovement.CreateArchingResponseDTOclose(arching);
20     }
21
22     @Override 1 usage ▲ RobMaster28
23     public Tuple<ArchingResponseDTO, UUID> saveArchingResponseDTO(UUID id_box, ArchingInitDTO archingInitDTO) throws NotFoundException,
24     BadRequestException {
25         return archingService.saveArchingResponseDTO(id_box, archingInitDTO);
26     }
27
28 }
29
30 }
```

OrderSetOrchestratorServiceImp

```
① OrderSetOrchestratorServiceImp.java ×
1 package com.example.api_v01.service.service_aux;
2
3 > import ...
4
5 @Service ▲ RobMaster28 +1
6 @RequiredArgsConstructor
7 public class OrderSetOrchestratorServiceImp implements OrderSetOrchestratorService, ExceptionMessage {
8
9     private final OrderSetService orderSetService;
10    private final CustomerOrderService customerOrderService;
11
12    //Sirve para guardar la lista junto con sus órdenes
13    @Override 1 usage ▲ RobMaster28 +1
14    public Tuple<OrderSetResponseDTO, UUID> saveCompleteOrderSet(
15        UUID id_arching,
16        String name_client,
17        List<CustomerOrderDTO> listCustomer
18    ) throws NotFoundException, BadRequestException {
19
20        if(listCustomer.isEmpty()){
21            throw new BadRequestException(IS_EMPTY_LIST_ORDER_SET);
22        }
23
24        OrderSet orderSet = orderSetService.saveBaseOrderSet(id_arching, name_client);
25
26        for (CustomerOrderDTO customerOrderDTO : listCustomer) {
27            customerOrderService.saveCustomerOrder(
28                customerOrderDTO.getId_product(),
29                orderSet.getId_order_set(),
30                customerOrderDTO.getCount()
31            );
32        }
33    }
34
35 }
```

AuthService

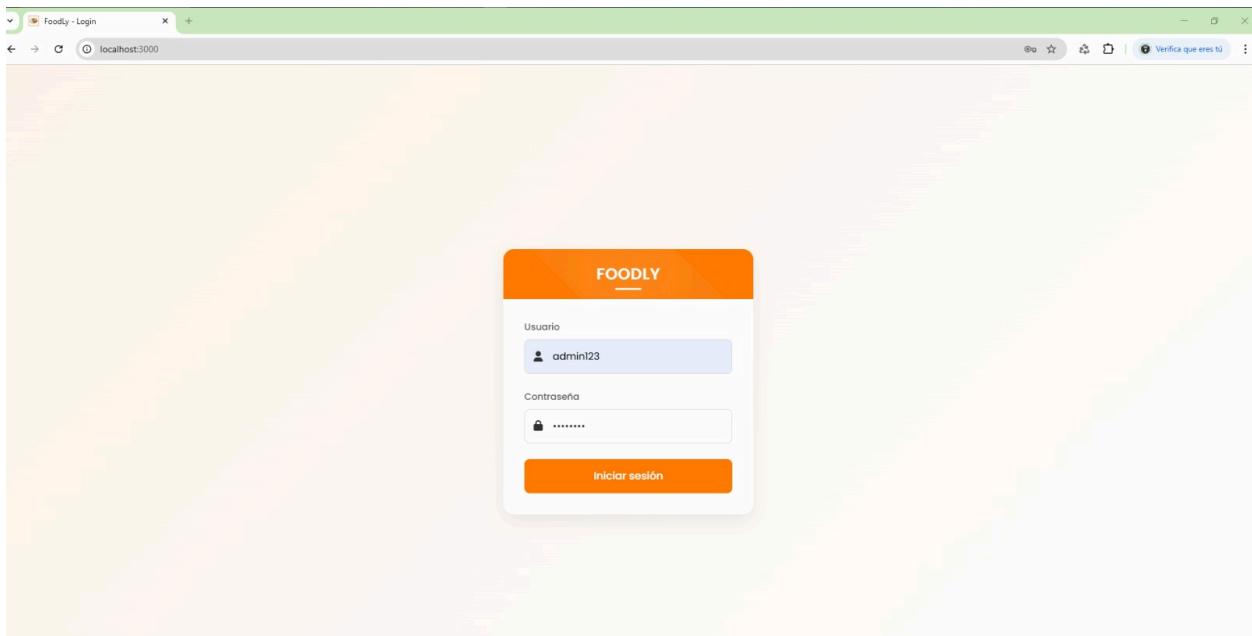


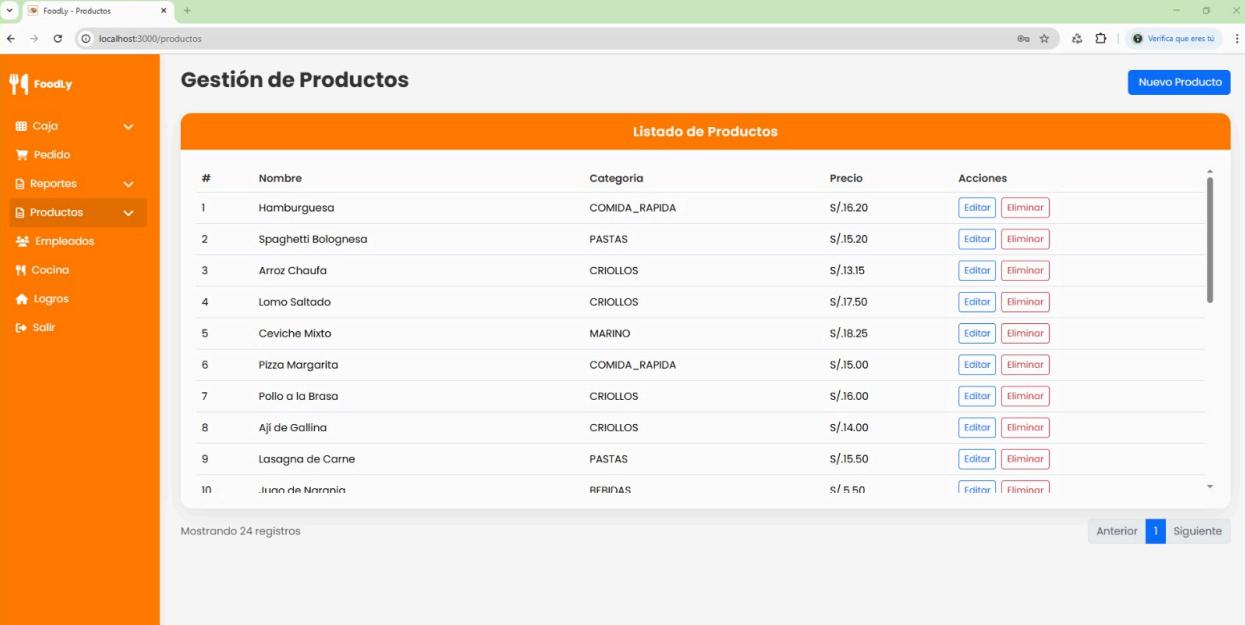
```
AuthService.java
1 package com.example.api_v01.service.user_service;
2
3 > import ...
4
5 @Service 2 usages ▲ RobMaster28
6 @RequiredArgsConstructor
7 public class AuthService {
8
9     private final UserRepository userRepository;
10    private final JwtUtils jwtUtils;
11    private final PasswordEncoder passwordEncoder;
12    private final AuthenticationManagerBuilder authenticationManagerBuilder;
13
14    private final AdminService adminService;
15    private final ATMService atmService;
16
17    public AuthResponse<?> authenticate(String username, String password) throws NotFoundException { 1 usage ▲ RobMaster28
18        UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(username, password);
19
20        Authentication authentication = authenticationManagerBuilder
21            .getObjects().authenticate(authenticationToken);
22
23        SecurityContextHolder.getContext().setAuthentication(authentication);
24
25        String jwt = jwtUtils.generateToken(authentication);
26
27        User user = userRepository.findByUsername(username)
28            .orElseThrow(() -> new UsernameNotFoundException("User no encontrado"));
29
30    }
31
32 }
```

User Service

```
UserService.java ×
1 package com.example.api_v01.service.user_service;
2
3 > import ...
4
5 @Service ▲ RobMaster28
6 @RequiredArgsConstructor
7 public class UserService implements ExceptionMessage, UserDetailsService {
8
9     private final UserRepository userRepository;
10
11     @Override 1 usage ▲ RobMaster28
12     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
13         User user = userRepository.findByUsername(username)
14             .orElseThrow(() -> new UsernameNotFoundException(USER_NOT_FOUND));
15
16         SimpleGrantedAuthority authority = new SimpleGrantedAuthority(role: "ROLE_" + user.getRole().name());
17
18         return new org.springframework.security.core.userdetails.User(
19             user.getUsername(),
20             user.getPassword(),
21             List.of(authority)
22         );
23     }
24
25     public void save(User user) { userRepository.save(user); }
26
27 }
```

Vistas funcionales





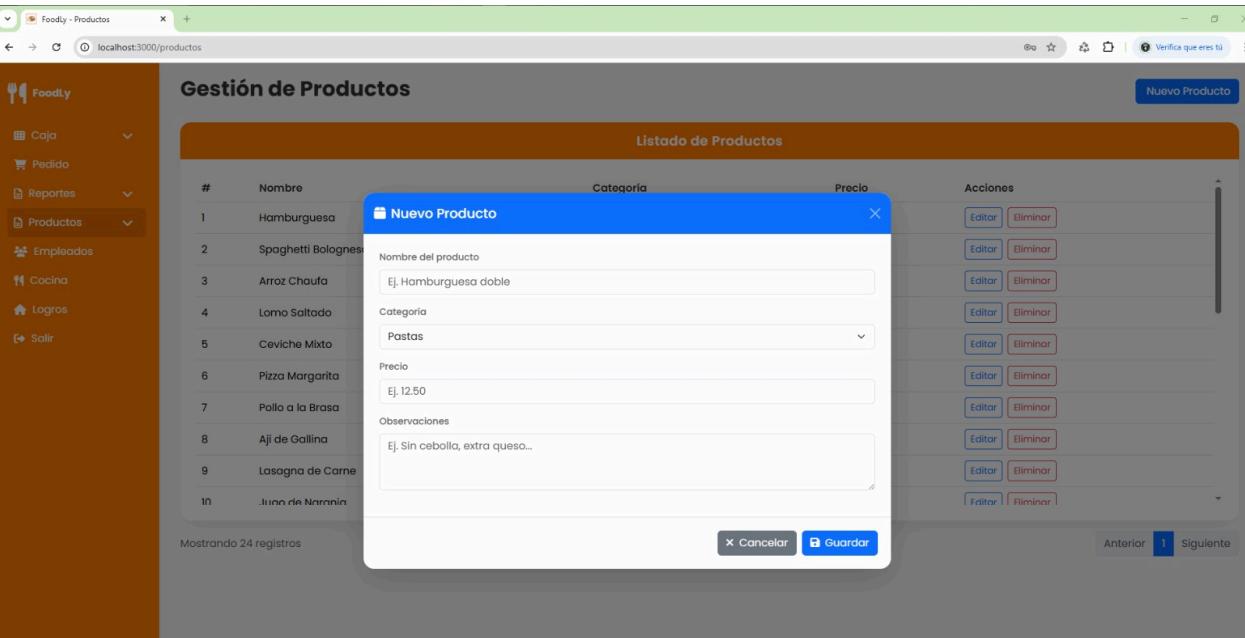
Gestión de Productos

Listado de Productos

#	Nombre	Categoría	Precio	Acciones
1	Hamburguesa	COMIDA_RAPIDA	S/.16.20	[Editor] [Eliminar]
2	Spaghetti Bolognesa	PASTAS	S/.15.20	[Editor] [Eliminar]
3	Arroz Chaufa	CRIOLOS	S/.13.15	[Editor] [Eliminar]
4	Lomo Saltado	CRIOLOS	S/.17.50	[Editor] [Eliminar]
5	Ceviche Mixto	MARINO	S/.18.25	[Editor] [Eliminar]
6	Pizza Margarita	COMIDA_RAPIDA	S/.15.00	[Editor] [Eliminar]
7	Pollo a la Brasa	CRIOLOS	S/.16.00	[Editor] [Eliminar]
8	Aji de Gallina	CRIOLOS	S/.14.00	[Editor] [Eliminar]
9	Lasagna de Carne	PASTAS	S/.15.50	[Editor] [Eliminar]
10	Jugo de Naranja	REFRÍDENS	S/.5.50	[Filtrar] [Eliminar]

Mostrando 24 registros

Anterior | Siguiente



Gestión de Productos

Listado de Productos

Nuevo Producto

Nombre del producto:
Ej: Hamburguesa doble

Categoría:
Pastas

Precio:
Ej: 12.50

Observaciones:
Ej: Sin cebolla, extra queso...

Cancelar Guardar

Gestión de Productos

Listado de Productos

#	Nombre	Categoría	Precio	Acciones
1	Hamburguesa			Editar Eliminar
2	Spaghetti Bolognesa			Editar Eliminar
3	Arroz Chaufa			Editar Eliminar
4	Lomo Saltado			Editar Eliminar
5	Ceviche Mixto	Comida Rápida		Editar Eliminar
6	Pizza Margarita			Editar Eliminar
7	Pollo a la Brasa			Editar Eliminar
8	Aji de Gallina			Editar Eliminar
9	Lasagna de Carne			Editar Eliminar
10	Jugo de Naranja			Editar Eliminar

Mostrando 24 registros

[Cancelar](#) [Guardar](#)

[Nuevo Producto](#)

Gestión de Productos

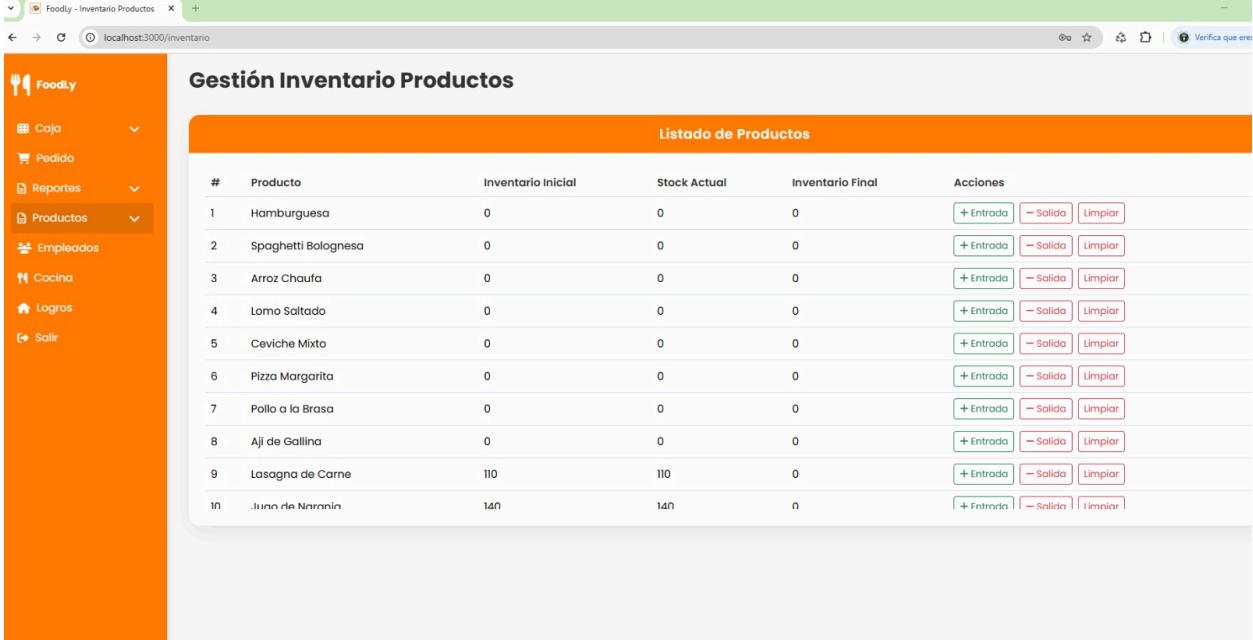
Listado de Productos

#	Nombre	Categoría	Precio	Acciones
1	Hamburguesa	COMIDA_RAPIDA	S/.16.20	Editar Eliminar
2	Spaghetti Bolognesa	PASTAS	S/.15.20	Editar Eliminar
3	Arroz Chaufa		S/.13.15	Editar Eliminar
4	Lomo Saltado		S/.17.50	Editar Eliminar
5	Ceviche Mixto		S/.18.25	Editar Eliminar
6	Pizza Margarita		S/.15.00	Editar Eliminar
7	Pollo a la Brasa		S/.16.00	Editar Eliminar
8	Aji de Gallina		S/.14.00	Editar Eliminar
9	Lasagna de Carne		S/.15.50	Editar Eliminar
10	Jugo de Naranja	REFRÍDENS	S/. 5.50	Editar Eliminar

Mostrando 24 registros

[Cancelar](#) [Si, eliminar](#)

[Nuevo Producto](#)

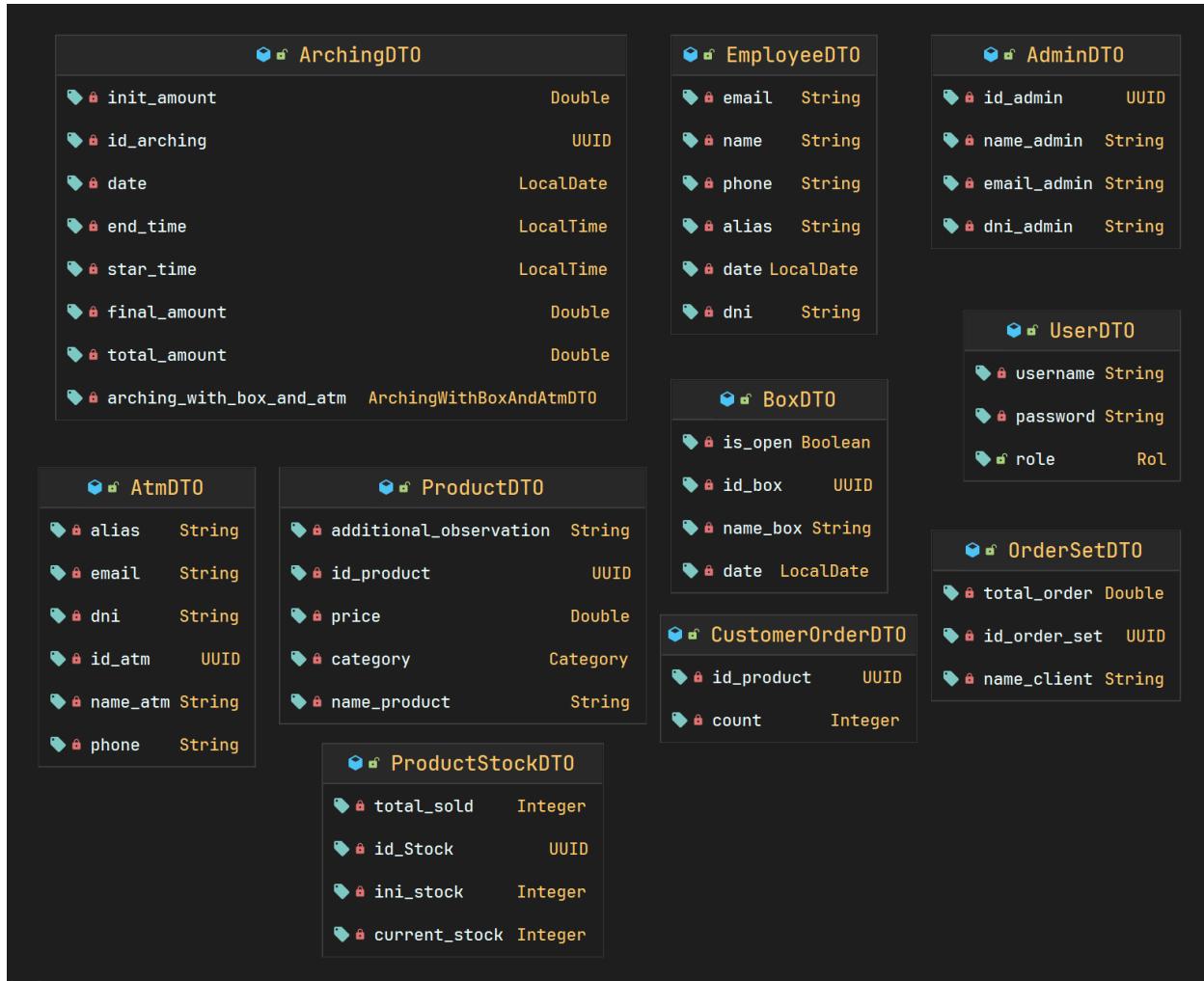


The screenshot shows a web-based inventory management system for a restaurant called "Foodly". The interface includes a sidebar with navigation links like Caja, Pedido, Reportes, Productos (selected), Empleados, Cocina, Logros, and Salir. The main content area is titled "Gestión Inventario Productos" and displays a table titled "Listado de Productos". The table lists 10 items with columns for #, Producto, Inventario Inicial, Stock Actual, Inventario Final, and Acciones. Each row has buttons for '+ Entrada' (green), '- Salida' (red), and 'Limpiar' (grey). The final column shows the net change in stock for each item.

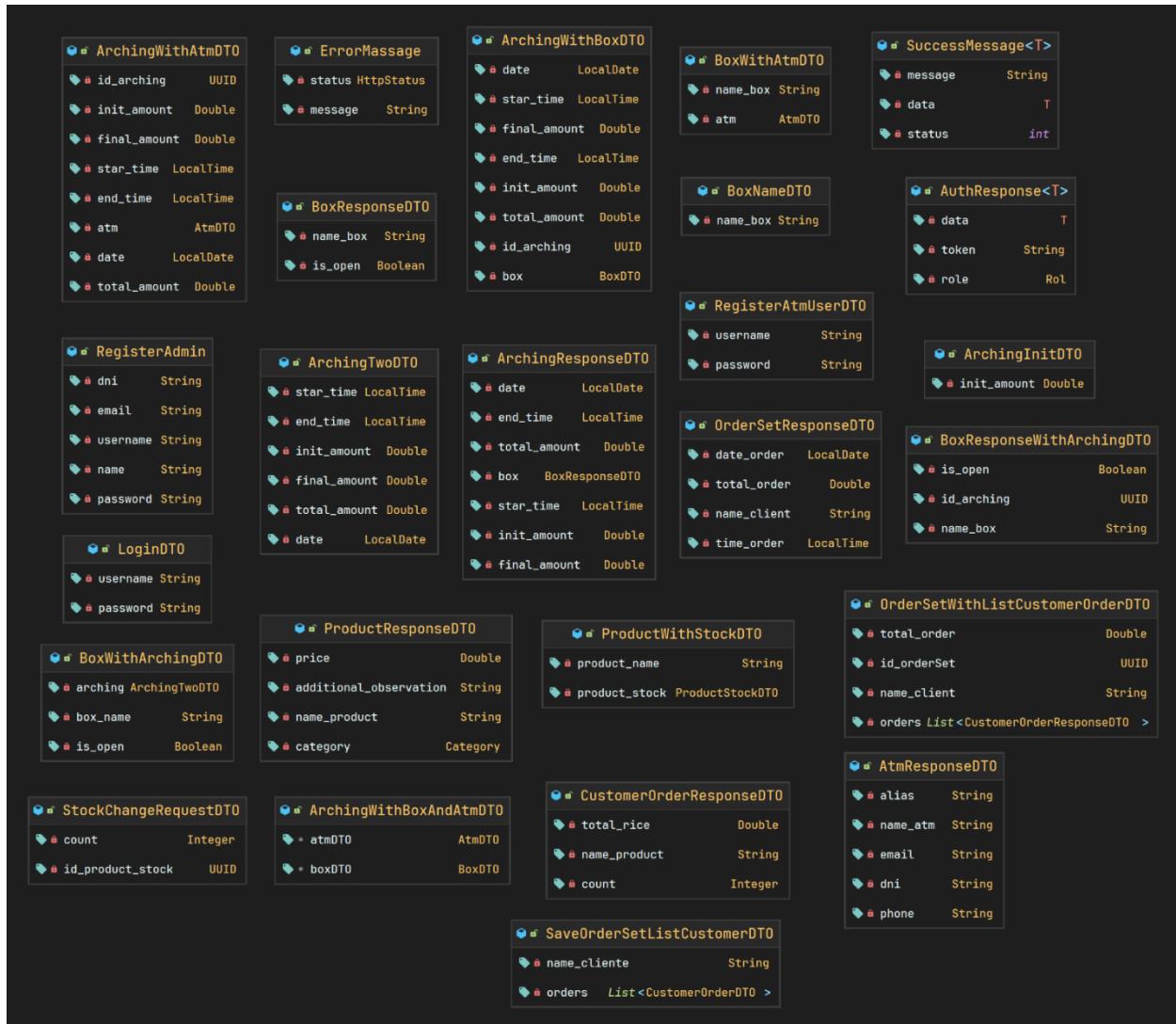
#	Producto	Inventario Inicial	Stock Actual	Inventario Final	Acciones
1	Hamburguesa	0	0	0	+ Entrada - Salida Limpiar
2	Spaghetti Bolognesa	0	0	0	+ Entrada - Salida Limpiar
3	Arroz Chafua	0	0	0	+ Entrada - Salida Limpiar
4	Lomo Saltado	0	0	0	+ Entrada - Salida Limpiar
5	Ceviche Mixto	0	0	0	+ Entrada - Salida Limpiar
6	Pizza Margarita	0	0	0	+ Entrada - Salida Limpiar
7	Pollo a la Brasa	0	0	0	+ Entrada - Salida Limpiar
8	Aji de Gallina	0	0	0	+ Entrada - Salida Limpiar
9	Lasagna de Carne	110	110	0	+ Entrada - Salida Limpiar
10	Jugo de Naranja	140	140	0	+ Entrada - Salida Limpiar

DTOs

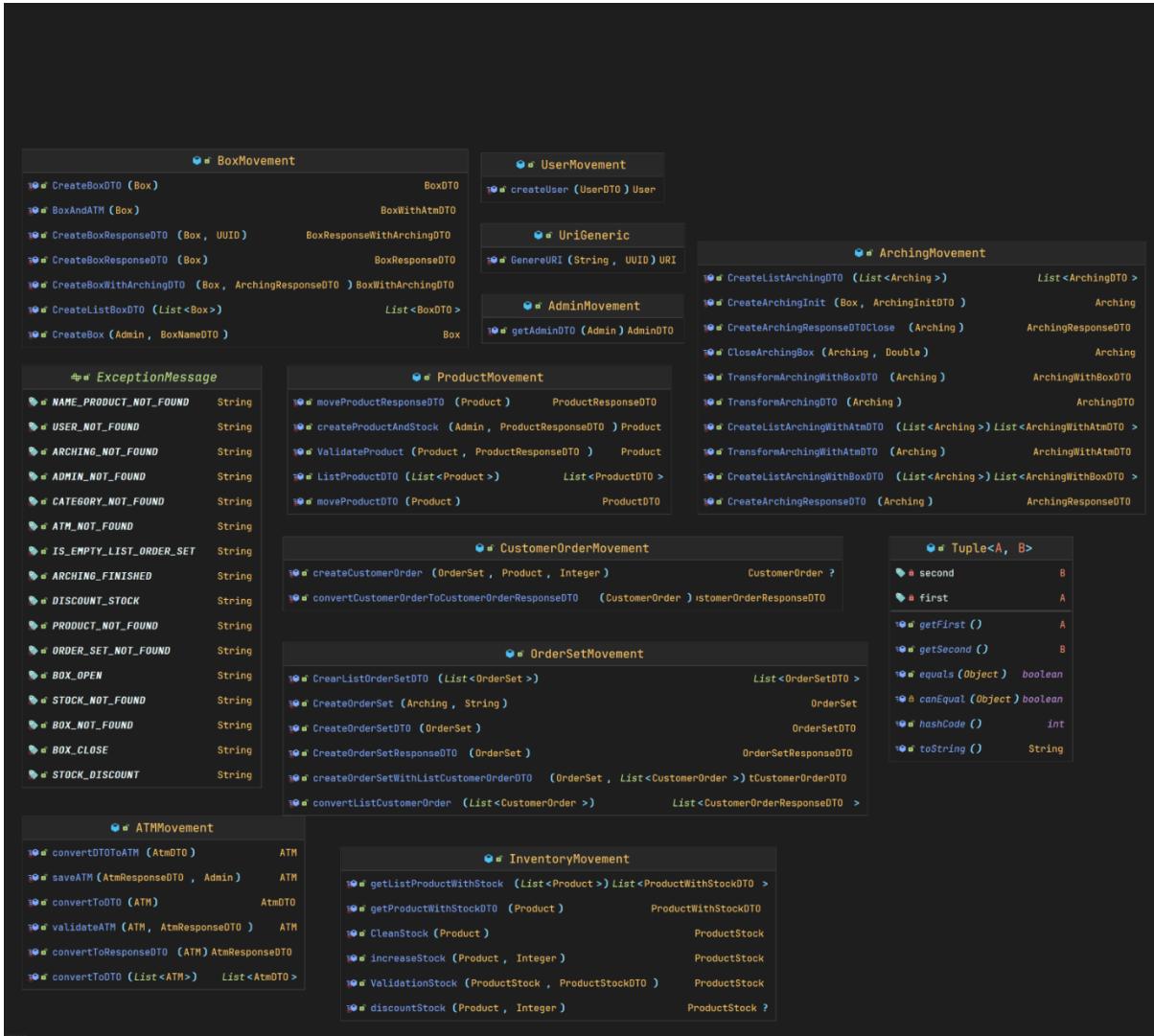
Entity



Response



Utils



AdminMovement

```
© AdminMovement.java ×  
1 package com.example.api_v01.utils;  
2  
3 > import ...  
5  
6 public class AdminMovement { 2 usages • RobMaster28  
7 @ ...     public static AdminDTO getAdminDTO(Admin admin) { 1 usage • RobMaster28  
8         return AdminDTO.builder()  
9             .id_admin(admin.getId_admin())  
10            .email_admin(admin.getEmail_admin())  
11            .name_admin(admin.getName_admin())  
12            .dni_admin(admin.getDni_admin())  
13            .build();  
14     }  
15 }
```

ArchingMovement

```
① ArchingMovement.java ×
1 package com.example.api_v01.utils;
2
3 > import ...
13
14 public class ArchingMovement { 15 usages  ± roberto +1
15 @   public static Arching CreateArchingInit(Box box, ArchingInitDTO archingInitDTO) { 1 usage  ± RobMaster28 +1
16     return Arching.builder()
17         .date(LocalDate.now())
18         .star_time(LocalTime.now())
19         .init_amount(archingInitDTO.getInit_amount())
20         .box(box)
21         .build();
22     }
23
24 @   public static ArchingResponseDTO CreateArchingResponseDTOClose(Arching arching) { 1 usage  ± roberto
25     return ArchingResponseDTO.builder()
26         .date(arching.getDate())
27         .star_time(arching.getStar_time())
28         .end_time(arching.getEnd_time())
29         .init_amount(arching.getInit_amount())
30         .final_amount(arching.getFinal_amount())
31         .total_amount(arching.getTotal_amount())
32         .box(
33             BoxResponseDTO.builder()
34                 .name_box(arching.getBox().getName_box())
35                 .is_open(arching.getBox().getIs_open())
36                 .build()
37             )
38         .build();
39     }
```

ATMMovement

```
© ATMMovement.java ×

1 package com.example.api_v01.utils;
2
3 > import ...
4
5
6
7 public class ATMMovement { 12 usages ± Sergio Antonio +2
8
9
10 @    public static ATM saveATM(AtmResponseDTO atm, Admin admin) { 1 usage ± RobMaster28 +1
11         return ATM.builder()
12             .name_atm(atm.getName_atm())
13             .date(LocalDate.now())
14             .alias(atm.getAlias())
15             .email(atm.getEmail())
16             .phone(atm.getPhone())
17             .dni(atm.getDni())
18             .admin(admin)
19             .build();
20     }
21
22
23 @    public static ATM validateATM(ATM atm, AtmResponseDTO atmDTO) { 1 usage ± roberto +1
24         if(atmDTO.getName_atm() != null){
25             atm.setName_atm(atmDTO.getName_atm());
26         }
27         if(atmDTO.getAlias() != null){
28             atm.setAlias(atmDTO.getAlias());
29         }
30         if(atmDTO.getEmail() != null){
31             atm.setEmail(atmDTO.getEmail());
32         }
33         if(atmDTO.getPhone() != null){
34             atm.setPhone(atmDTO.getPhone());
35         }
36         if(atmDTO.getDni() != null){
37             atm.setDni(atmDTO.getDni());
38         }
39     }
40
41
42     if(atmDTO.getName_atm() != null){
43         atm.setName_atm(atmDTO.getName_atm());
44     }
45 }
```

BoxMovement

```
② BoxMovement.java ×

1 package com.example.api_v01.utils;
2
3 > import ...
4
5
6 public class BoxMovement { 10 usages + roberto +1
7     @
8         public static List<BoxDTO> CreateListBoxDTO(List<Box> boxes){ 2 usages + roberto
9             return boxes.stream() Stream<Box>
10                .map(BoxMovement::CreateBoxDTO) Stream<BoxDTO>
11                .toList();
12            }
13
14     @
15         public static BoxDTO CreateBoxDTO(Box box) { 2 usages + roberto
16             return BoxDTO.builder()
17                 .id_box(box.getId_box())
18                 .name_box(box.getName_box())
19                 .date(box.getDate())
20                 .is_open(box.getIs_open())
21                 .build();
22             }
23
24     @
25         public static Box CreateBox(Admin admin, BoxNameDTO boxDTO) { 1 usage + RobMaster28 +1
26             return Box.builder()
27                 .name_box(boxDTO.getName_box())
28                 .date(LocalDate.now())
29                 .admin(admin)
30                 .build();
31             }
32
33     @
34         public static BoxResponseDTO CreateBoxResponseDTO(Box box) { 1 usage + roberto
35             return BoxResponseDTO.builder()
36                 .name_box(box.getName_box())
37                 .is_open(box.getIs_open())
38                 .build();
39             }
```

CustomerOrderMovement

```
⑥ CustomerOrderMovement.java ×
1 package com.example.api_v01.utils;
2
3 > import ...
11
12 public class CustomerOrderMovement { 3 usages ▲ RobMaster28
13 @  public static CustomerOrder createCustomerOrder(OrderSet orderSet,Product product, Integer count) { 1 usage ▲ RobMaster28
14     if(product.getStock().getCurrent_stock() >= count){
15         return CustomerOrder.builder()
16             .product(product)
17             .order(orderSet)
18             .count(count)
19             .total_rice( product.getPrice() * count )
20             .build();
21     }
22     return null;
23 }
24 @  public static CustomerOrderResponseDTO convertCustomerOrderToCustomerOrderResponseDTO(CustomerOrder customerOrder){ 1 usage
25     return CustomerOrderResponseDTO.builder()
26         .name_product(customerOrder.getProduct().getName_product())
27         .count(customerOrder.getCount())
28         .total_rice(customerOrder.getTotal_rice())
29         .build();
30     }
31 }
32 }
```

ExceptionMessage

```
① ExceptionMessage.java ×
1 package com.example.api_v01.utils;
2
3
4 ④ public interface ExceptionMessage { 10 implementations  ▲ RobMaster28 +1
5     final String ATM_NOT_FOUND = "ATM no encontrado"; 9 usages
6     final String STOCK_NOT_FOUND = "Stock no encontrado"; 6 usages
7     final String DISCOUNT_STOCK = "La cantidad ingresada a descontar supera la actual del stock"; 2 usages
8     final String USER_NOT_FOUND = "Usuario no encontrado"; 1 usage
9     final String PRODUCT_NOT_FOUND = "El producto no encontrado"; 5 usages
10    final String BOX_NOT_FOUND = "El box no encontrado"; 8 usages
11    final String ARCHING_NOT_FOUND = "El arqueo no encontrado"; 2 usages
12    final String ADMIN_NOT_FOUND = "El administrador no encontrado"; 2 usages
13
14    final String BOX_OPEN="La caja ya esta abierta"; 1 usage
15    final String BOX_CLOSE="La caja ya esta cerrada"; 1 usage
16    final String STOCK_DISCOUNT="No se puede quitar mas del stock actual del producto"; 1 usage
17    final String ORDER_SET_NOT_FOUND = "La lista de ordenes del cliente no fue encontrada"; 1 usage
18    final String CATEGORY_NOT_FOUND = "El categoria no encontrado"; 2 usages
19    final String NAME_PRODUCT_NOT_FOUND = "Ninguno producto coincide con este nombre"; 2 usages
20    final String IS_EMPTY_LIST_ORDER_SET = "No se puede guardar el orderSet si su lista de productos a ordenar esta vacia"; 1 usage
21    final String ARCHING_FINISHED = "El arqueo que quiere asignar dejó de estar activo, Ya no puede ser usado"; 1 usage
22 }
```

InventoryMovement

```
① InventoryMovement.java ×
1 package com.example.api_v01.utils;
2
3 > import ...
4
5
6 public class InventoryMovement { 16 usages  ▲ roberto +1
7
8
9
10 @    public static List<ProductWithStockDTO> getListProductWithStock(List<Product> products) { 3 usages  ▲ roberto
11         return products.stream() Stream<Product>
12             .map(InventoryMovement::getProductWithStockDTO) Stream<ProductWithStockDTO>
13             .toList();
14     }
15
16 @    public static ProductWithStockDTO getProductWithStockDTO(Product product) { 7 usages  ▲ roberto
17         return ProductWithStockDTO.builder()
18             .product_name(product.getName_product())
19             .product_stock(
20                 ProductStockDTO.builder()
21                     .id_Stock(product.getStock().getId_product_stock())
22                     .ini_stock(product.getStock().getIni_stock())
23                     .current_stock(product.getStock().getCurrent_stock())
24                     .total_sold(product.getStock().getTotal_sold())
25                     .build()
26             )
27             .build();
28     }
29
30 }
31 }
```

OrderSetMovement

```
⑤ OrderSetMovement.java x
1 package com.example.api_v01.utils;
2
3 > import ...
4
5
6 public class OrderSetMovement { 8 usages  ± RobMaster28 +1
7     //Se utiliza para crear el orderset
8     public static OrderSet CreateOrderSet(Arching arching, String name_client){ 1 usage  ± RobMaster28 +1
9         return OrderSet.builder()
10             .name_client(name_client)
11             .date_order(LocalDate.now())
12             .time_order(LocalTime.now())
13             .arching(arching)
14             .build();
15     }
16
17     @
18     public static OrderSetResponseDTO CreateOrderSetResponseDTO(OrderSet orderSet){ 1 usage  ± RobMaster28
19         return OrderSetResponseDTO.builder()
20             .name_client(orderSet.getName_client())
21             .time_order(orderSet.getTime_order())
22             .date_order(orderSet.getDate_order())
23             .total_order(orderSet.getTotal_order())
24             .build();
25     }
26
27     @
28     public static OrderSetDTO CreateOrderSetDTO(OrderSet orderSetDTO){ 1 usage  ± RobMaster28
29         return OrderSetDTO.builder()
30             .id_order_set(orderSetDTO.getId_order_set())
31             .name_client(orderSetDTO.getName_client())
32             .total_order(orderSetDTO.getTotal_order())
33             .build();
34     }
35
36 }
```

ProductMovement

```
① ProductMovement.java ×
1 package com.example.api_v01.utils;
2
3 > import ...
10
11 public class ProductMovement { 10 usages  ▲ roberto +1
12
13 @  public static List<ProductDTO> ListProductDTO(List<Product> products) { 3 usages  ▲ roberto
14     return products.stream() Stream<Product>
15         .map(ProductMovement::moveProductDTO) Stream<ProductDTO>
16         .toList();
17 }
18
19 @  public static ProductDTO moveProductDTO(Product product) { 2 usages  ▲ roberto
20     return ProductDTO.builder()
21         .id_product(product.getId_Product())
22         .name_product(product.getName_product())
23         .price(product.getPrice())
24         .category(product.getCategory())
25         .additional_observation(product.getAdditional_observation())
26         .build();
27 }
28
29 @  public static ProductResponseDTO moveProductResponseDTO(Product product) { 2 usages  ▲ roberto
30     return ProductResponseDTO.builder()
31         .name_product(product.getName_product())
32         .price(product.getPrice())
33         .category(product.getCategory())
34         .additional_observation(product.getAdditional_observation())
35         .build();
36 }
```

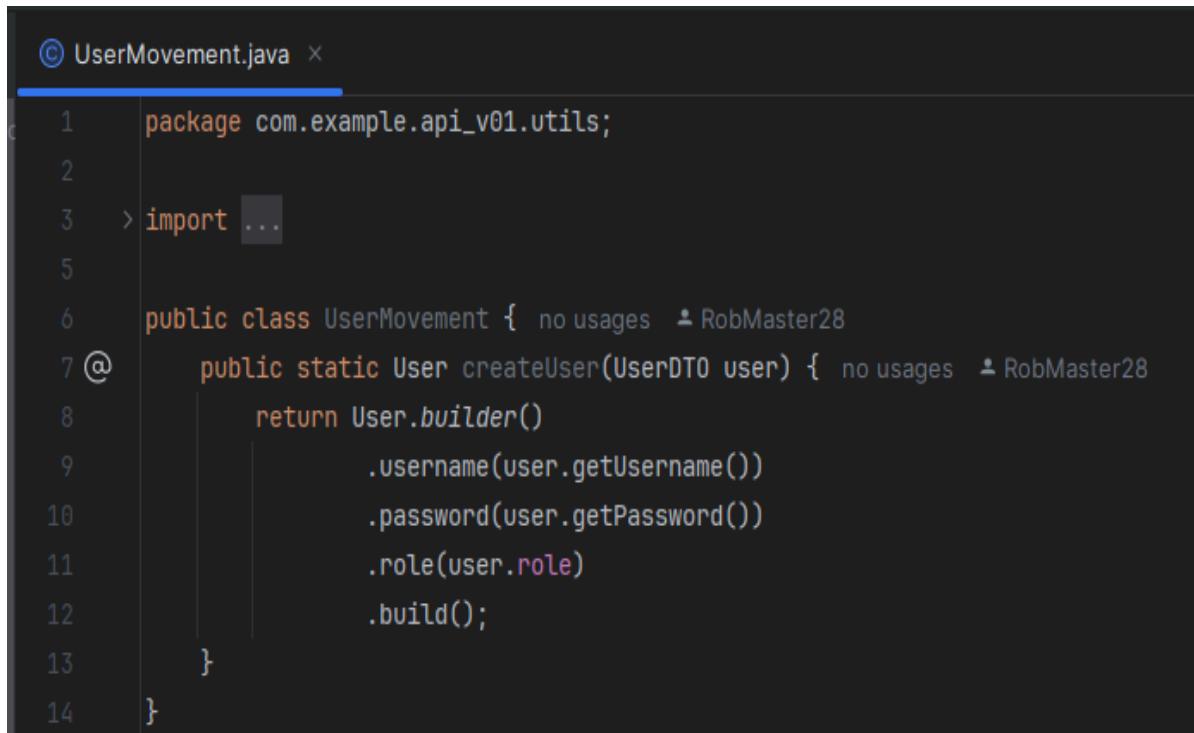
Tuple

```
© Tuple.java ×  
1 package com.example.api_v01.utils;  
2  
3 > import ...  
7  
8  
9 @Data  ↳ RobMaster28  
10 @RequiredArgsConstructor  
11 public class Tuple<A,B>{  
12     private final A first;  
13     private final B second;  
14 }
```

UriGeneric

```
© UriGeneric.java ×  
1 package com.example.api_v01.utils;  
2  
3 > import ...  
7  
8 public class UriGeneric { 4 usages ↳ roberto  
9 @    public static URI GenereURI(String Ruta_GetById_Entity,UUID IdEntity) { 2 usages ↳ roberto  
10        return ServletUriComponentsBuilder  
11            .fromCurrentContextPath()  
12            .path(Ruta_GetById_Entity)  
13            .buildAndExpand(IdEntity)  
14            .toUri();  
15    }  
16 }
```

UserMovement



The screenshot shows a code editor window with a dark theme. The file is named "UserMovement.java". The code defines a static method "createUser" that takes a "UserDTO" object and returns a "User" object created using a builder pattern.

```
© UserMovement.java ×

1 package com.example.api_v01.utils;
2
3 > import ...
5
6 public class UserMovement { no usages  ↗ RobMaster28
7 @     public static User createUser(UserDTO user) { no usages  ↗ RobMaster28
8         return User.builder()
9             .username(user.getUsername())
10            .password(user.getPassword())
11            .role(user.role)
12            .build();
13     }
14 }
```

Capítulo 4

DATATABLE

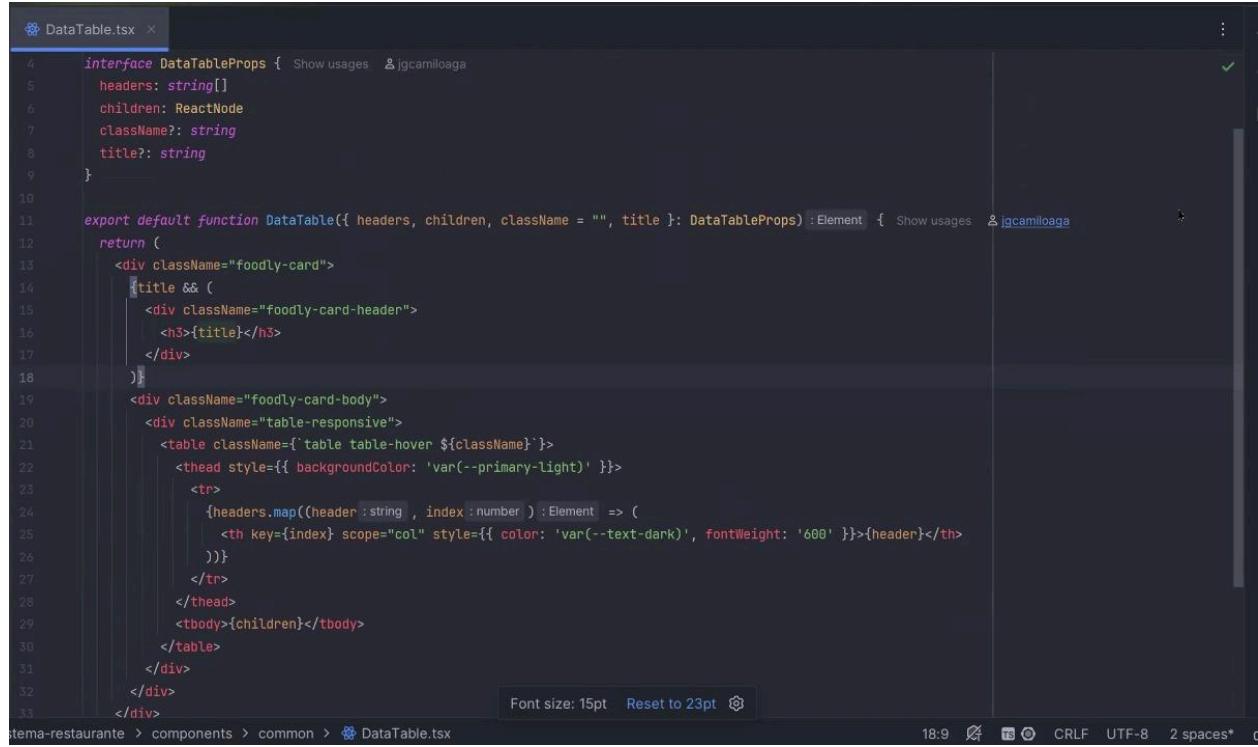
(Actualizar)

```
/**  
 * Actualiza un producto existente parcialmente.  
 *  
 * @param id - ID del producto que se desea actualizar  
 * @param producto - Objeto con los campos a actualizar  
 * @param token - Token de autenticación JWT  
 * @returns Promesa con la respuesta del backend  
 */  
export const updateProducto = ( Show usages & Jota  
  id: number,  
  producto: Partial<Producto>, // Se permite actualizar solo algunos campos  
  token: string  
) =>  
  axios.patch(`${PRODUCT_API_URL}/${id}`, producto, {  
    headers: {  
      Authorization: `Bearer ${token}`,  
      "Content-Type": "application/json",  
    },  
  })
```

HECHO EN AXIOS

(CREAR)

```
/**  
 * Crea un nuevo producto en el sistema, asociado a un administrador.  
 *  
 * @param producto - Objeto que contiene los datos del nuevo producto (sin ID)  
 * @param idAdmin - ID del administrador que está creando el producto  
 * @param token - Token de autenticación JWT  
 * @returns Promesa con la respuesta del backend  
 */  
export const createProducto :(producto: Omit<Producto, "id_product">, ... ) = ( Show usages & Jota  
    producto: Omit<Producto, "id_product">, // Se omite el ID porque lo genera el backend  
    idAdmin: number,  
    token: string  
) :Promise<AxiosResponse<any, any>> =>  
    axios.post( url: `${PRODUCT_API_URL}/${idAdmin}`, producto, {  
        headers: {  
            Authorization: `Bearer ${token}`, // Token JWT para autenticación  
            "Content-Type": "application/json", // Se especifica que se envía JSON  
        },  
    })
```



```
4  interface DataTableProps { Show usages & jgcamiloga
5    headers: string[]
6    children: ReactNode
7    className?: string
8    title?: string
9  }
10
11 export default function DataTable({ headers, children, className = "", title }: DataTableProps): Element { Show usages & jgcamiloga
12   return (
13     <div className="foodly-card">
14       {title && (
15         <div className="foodly-card-header">
16           <h3>{title}</h3>
17         </div>
18       )}
19       <div className="foodly-card-body">
20         <div className="table-responsive">
21           <table className={`table table-hover ${className}`}>
22             <thead style={{ backgroundColor: 'var(--primary-light)' }}>
23               <tr>
24                 {headers.map((header: string, index: number): Element => (
25                   <th key={index} scope="col" style={{ color: 'var(--text-dark)', fontWeight: '600' }}>{header}</th>
26                 ))}
27               </tr>
28             </thead>
29             <tbody>{children}</tbody>
30           </table>
31         </div>
32       </div>
33     </div>
34   )
35 }
```

(ELIMINAR)

```
/***
 * Elimina un producto por su ID.
 *
 * @param id - ID del producto a eliminar
 * @param token - Token de autenticación JWT
 * @returns Promesa con la respuesta del backend
 */
export const deleteProducto = (id: number, token: string) => Show usages & Jota
  axios.delete(`${PRODUCT_API_URL}/${id}`, {
    headers: { Authorization: `Bearer ${token}` },
  })

```

(CONSULTAS)

```
// Importamos axios para realizar solicitudes HTTP al backend
import axios from "axios"

// Importamos el tipo Producto para definir correctamente las estructuras de datos esperadas
import { Producto } from "../types/producto"

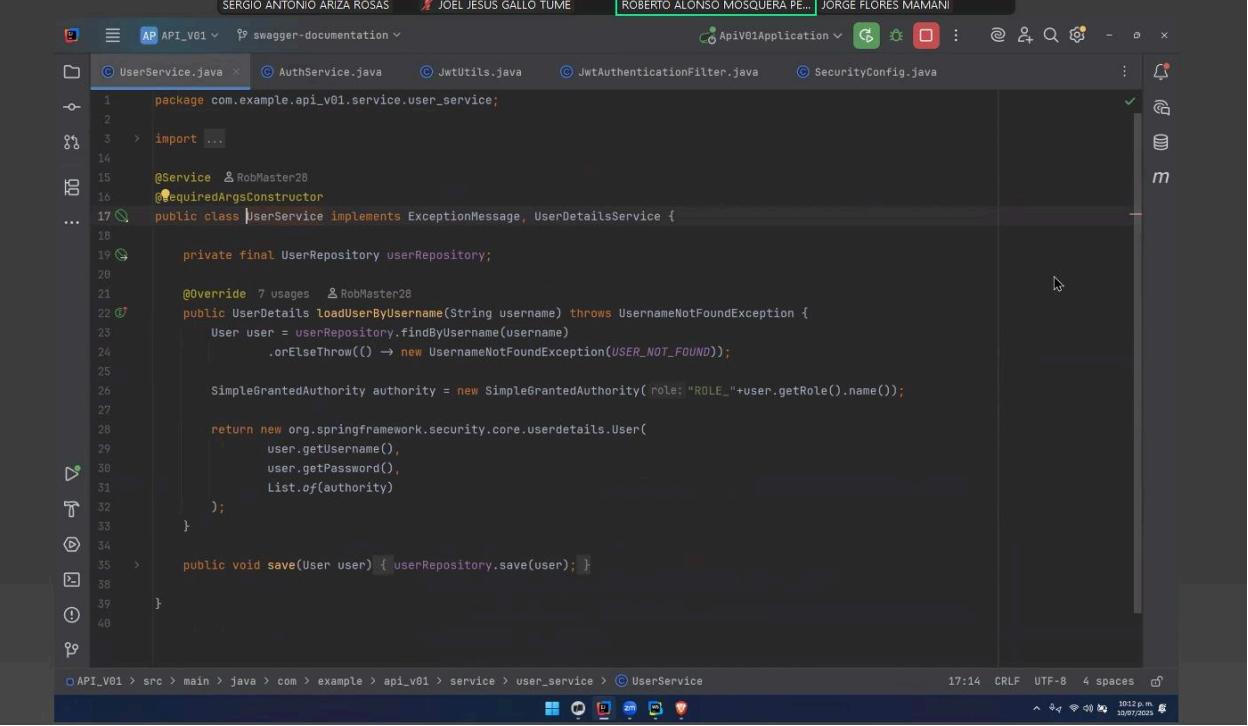
// Importamos URL Base de la API
import { API_BASE_URL } from "@services/api";

// URL base de la API para productos del backend en Spring Boot
const PRODUCT_API_URL = `${API_BASE_URL}/product`;

/*
 * Obtiene todos los productos desde el backend.
 * Se espera un token JWT para la autorización.
 *
 * @param token - Token de autenticación JWT
 * @returns Promise con un array de productos y el estado de la respuesta
 */
export const getAllProducts : (token: string, page?: number) => Promise<{ status: number, data: Producto[] }> = (token: string, page: number = 0) => axios.get(`${PRODUCT_API_URL}/list?page=${page}`, {
    headers: { Authorization: `Bearer ${token}` }, // Se pasa el token en el encabezado
})
```

MANEJO DE SESIONES

UserService.java



The screenshot shows a Java IDE interface with several tabs at the top: SERGIO ANTONIO ARIZA ROSAS, JOEL JESUS GALLO TUME, ROBERTO ALONSO MOSQUERA PE..., and JORGE FLORES MAMANI. Below the tabs, there is a file tree and a code editor. The code editor displays the UserService.java file:

```
package com.example.api_v01.service.user_service;

import ...;

@Service & RobMaster28
@RequiredArgsConstructor
public class UserService implements ExceptionMessage, UserDetailsService {

    private final UserRepository userRepository;

    @Override 7 usages & RobMaster28
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepository.findByUsername(username)
            .orElseThrow(() -> new UsernameNotFoundException(USER_NOT_FOUND));

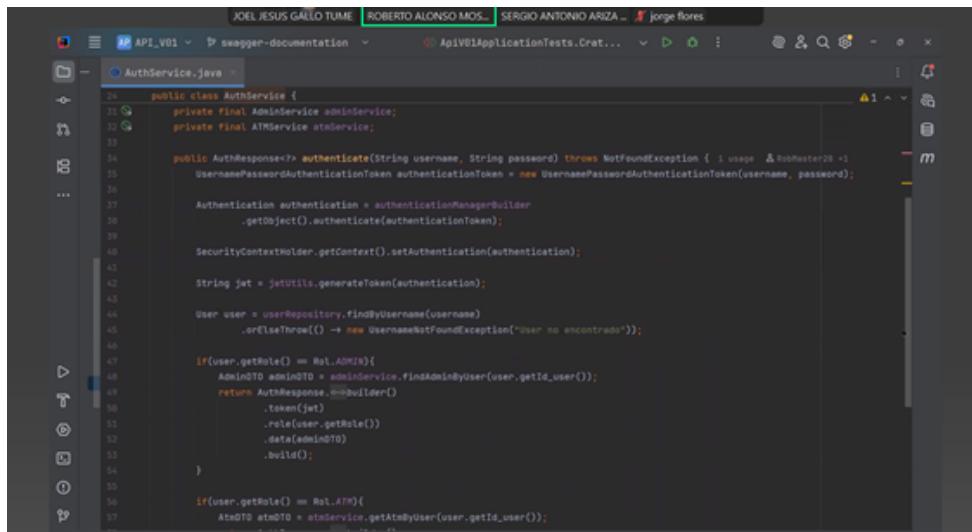
        SimpleGrantedAuthority authority = new SimpleGrantedAuthority(role: "ROLE_" + user.getRole().name());

        return new org.springframework.security.core.userdetails.User(
            user.getUsername(),
            user.getPassword(),
            List.of(authority)
        );
    }

    public void save(User user) { userRepository.save(user); }
}
```

The code uses annotations like @Service, @RequiredArgsConstructor, and @Override. It interacts with a UserRepository to load user details by username and save new users. The code is part of a Spring Security context, as indicated by the UserDetailsService interface and the use of SimpleGrantedAuthority.

AuthService.java



```

JOEL JESÚS GÁLLO TUME ROBERTO ALONSO MOS... SERGIO ANTONIO ARIZA ... jorge flores
API_V01 swagger-documentation ApiV01ApplicationTests.Crat...
AuthService.java

public class AuthService {
    private final AdminService adminService;
    private final ATMService atmService;

    public AuthResponse<?> authenticate(String username, String password) throws NotFoundException {
        UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(username, password);

        Authentication authentication = authenticationManagerBuilder
            .getBuilder()
            .authenticate(authenticationToken);

        SecurityContextHolder.getContext().setAuthentication(authentication);

        String jwt = jwtUtils.generateToken(authentication);

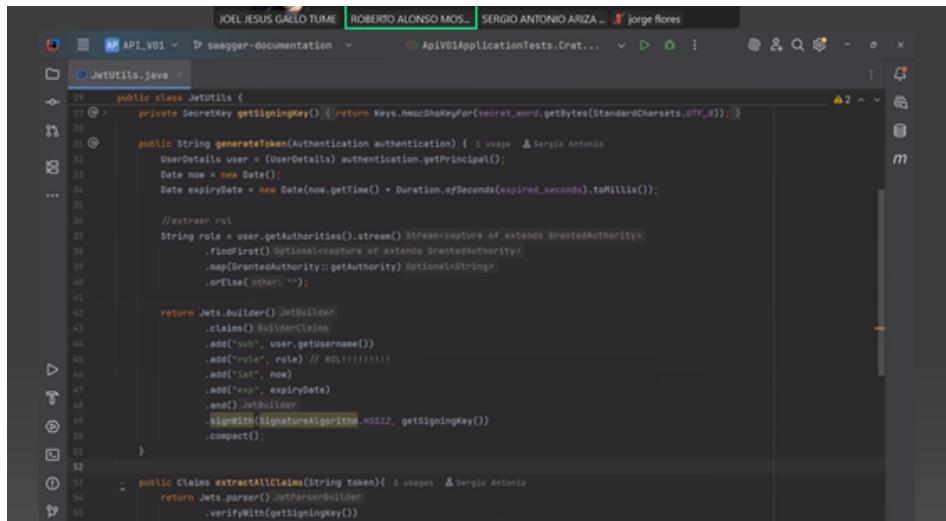
        User user = userRepository.findByUsername(username)
            .orElseThrow(() -> new UsernameNotFoundException("User no encontrado"));

        if(user.getRole() == Rol.ADMIN){
            AdminDTO adminDTO = adminService.findAdminByUser(user.getId_user());
            return AuthResponse.<?>.builder()
                .token(jwt)
                .role(user.getRole())
                .data(adminDTO)
                .build();
        }

        if(user.getRole() == Rol.ATM){
            AtmDTO atmDTO = atmService.getAtmByUser(user.getId_user());
        }
    }
}

```

JwUtils.java



```

JOEL JESÚS GÁLLO TUME ROBERTO ALONSO MOS... SERGIO ANTONIO ARIZA ... jorge flores
API_V01 swagger-documentation ApiV01ApplicationTests.Crat...
JwUtils.java

public class JwUtils {
    private SecretKey getSigningKey() { return Keys.hmacShaKeyFor(secret_word.getBytes(StandardCharsets.UTF_8)); }

    public String generateToken(Authentication authentication) { : usage & Sergio Antonio
        UserDetails user = (UserDetails) authentication.getPrincipal();
        Date now = new Date();
        Date expiryDate = new Date(now.getTime() + Duration.ofSeconds(expired_seconds).toMillis());

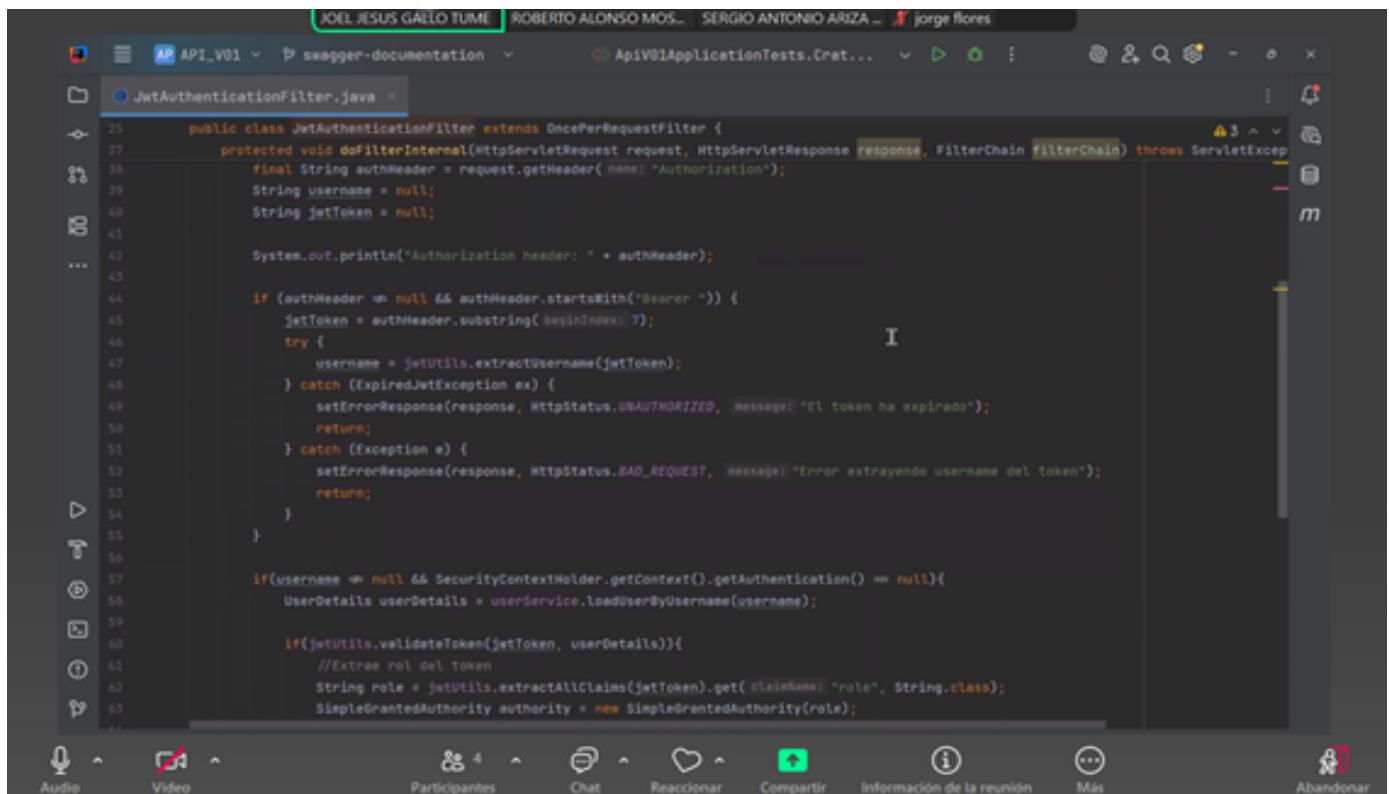
        //extraer rol
        String role = user.getAuthorities().stream()
            .findFirst()
            .mapGrantedAuthority()
            .getAuthority();

        return Jwts.builder()
            .claims()
            .add("sub", user.getUsername())
            .add("role", role) // ROL!!!!!!!
            .add("iat", now)
            .add("exp", expiryDate)
            .and()
            .signWith(SignatureAlgorithm.HS512, getSigningKey())
            .compact();
    }

    public Claims extractAllClaims(String token){ : usages & Sergio Antonio
        return Jwts.parser()
            .setSigningKey(getSigningKey())
    }
}

```

JwAuthenticationFilter.java



The screenshot shows a video conference interface with a code editor window overlaid. The code editor displays Java code for a JwAuthenticationFilter class. The code handles the 'Authorization' header from a request, extracts the username and token, and then checks if the user exists in the database and if the token is valid. It also sets up a simpleGrantedAuthority based on the role claim from the token.

```
JOEL JESUS GALLO TUME ROBERTO ALONSO MOS... SERGIO ANTONIO ARIZA ... jorge flores
JwAuthenticationFilter.java
public class JwAuthenticationFilter extends OncePerRequestFilter {
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {
        final String authHeader = request.getHeader("Authorization");
        String username = null;
        String jwtToken = null;

        System.out.println("Authorization header: " + authHeader);

        if (authHeader == null || !authHeader.startsWith("Bearer ")) {
            jwtToken = authHeader.substring(7);
            try {
                username = jwtUtils.extractUsername(jwtToken);
            } catch (ExpiredJwtException ex) {
                setErrorResponse(response, HttpStatus.UNAUTHORIZED, message: "El token ha expirado");
                return;
            } catch (Exception e) {
                setErrorResponse(response, HttpStatus.BAD_REQUEST, message: "Error extrayendo username del token");
                return;
            }
        }

        if(username == null && SecurityContextHolder.getContext().getAuthentication() == null){
            UserDetails userDetails = userService.loadUserByUsername(username);

            if(jwtUtils.validateToken(jwtToken, userDetails)){
                //Extrae rol del token
                String role = jwtUtils.extractAllClaims(jwtToken).get("role", String.class);
                SimpleGrantedAuthority authority = new SimpleGrantedAuthority(role);
            }
        }
    }
}
```

SecurityConfig.java

```

public class SecurityConfig {
    private final String[] EndpointsFree = { "/usage", "/auth/login", "/swagger-ui/**", "/v3/api-docs/**", "/swagger-ui.html" };
    private final String[] EndpointsCompartidos = { "/archiving/**", "/box/**", "/orderSet/**", "/product/**", "/stock/**" };
    private final String[] EndpointsSoloAdmin = { "/admin/**", "/ata/**" };

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
                .csrf(AbstractHttpConfigurer::disable)
                .cors(corsConfiguration -> corsConfiguration.configurationSource(corsConfigurationSource()))
                .authorizeHttpRequests(auth -> auth
                        .requestMatchers(EndpointsFree).permitAll()
                        .requestMatchers(EndpointsCompartidos).hasAnyRole("ADMIN", "ATH")
                        .requestMatchers(EndpointsSoloAdmin).hasRole("ADMIN")
                        .anyRequest().authenticated())
                .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);
        return http.build();
    }
}

```

SERVICIO WEB PARA LECTURA

```

// Importamos axios para realizar solicitudes HTTP al backend
import axios from "axios"

// Importamos el tipo Producto para definir correctamente las estructuras de datos esperadas
import { Producto } from "../types/producto"

// Importamos URL Base de la API
import { API_BASE_URL } from "@services/api";

// URL base de la API para productos del backend en Spring Boot
const PRODUCT_API_URL = `${API_BASE_URL}/product`;

/**
 * Obtiene todos los productos desde el backend.
 * Se espera un token JWT para la autorización.
 *
 * @param token - Token de autenticación JWT
 * @returns Promesa con un array de productos y el estado de la respuesta
 */
export const getAllProducts : (token: string, page?: number) => Promise<{ status: number; data: Producto[] }> = (token: string, page: number) => axios.get<{ status: number; data: Producto[] }>(`${PRODUCT_API_URL}/list?page=${page}`, {
    headers: { Authorization: `Bearer ${token}` }, // Se pasa el token en el encabezado
})

```

SERVICIO WEB DE PARA ESCRITURA

```
/**
 * Crea un nuevo producto en el sistema, asociado a un administrador.
 *
 * @param producto - Objeto que contiene los datos del nuevo producto (sin ID)
 * @param idAdmin - ID del administrador que está creando el producto
 * @param token - Token de autenticación JWT
 * @returns Promesa con la respuesta del backend
 */
export const createProducto :(producto: Omit<Producto, "id_product">, ... = ( Show usages & Jota
    producto: Omit<Producto, "id_product">, // Se omite el ID porque lo genera el backend
    idAdmin: number,
    token: string
) :Promise<AxiosResponse<any, any>> =>
  axios.post( url: `${PRODUCT_API_URL }/${idAdmin}`, producto, {
    headers: {
      Authorization: `Bearer ${token}`, // Token JWT para autenticación
      "Content-Type": "application/json", // Se especifica que se envía JSON
    },
  })
}
```

Glosario de términos

- **Spring Boot:** Herramienta que simplifica la creación de aplicaciones Spring mediante configuración automática y un modelo de proyecto "opinionated".
- **Spring Core:** Módulo central de Spring que proporciona inyección de dependencias y contenedores de beans.
- **Spring Data JPA:** Abstracción sobre JPA para simplificar el acceso a bases de datos relacionales mediante interfaces y anotaciones.

- **Spring Security:** Framework para manejar la seguridad de aplicaciones, incluyendo autenticación, autorización, protección CSRF, etc.
- **Spring Web / Spring MVC:** Utilizado para estructurar la aplicación siguiendo el patrón Modelo-Controlador, donde la vista es sustituida por respuestas JSON propias de una API REST.
- **Inyección de Dependencias (DI):** Patrón que permite a los objetos recibir sus dependencias de fuentes externas.
- **Controlador (Controller):** Clase que maneja solicitudes HTTP y coordina la lógica entre el modelo y la vista.
- **Servicio (Service):** Capa intermedia que contiene la lógica de negocio de la aplicación.
- **Repositorio (Repository):** Capa de acceso a datos que comunica la aplicación con la base de datos.
- **Entidad (Entity):** Clase que representa una tabla en la base de datos, usando anotaciones como @Entity.
- **Autenticación:** Proceso de verificar la identidad del usuario (por ejemplo, mediante usuario y contraseña).
- **Autorización:** Proceso de determinar qué recursos puede acceder un usuario autenticado.
- **JWT (JSON Web Token):** Token seguro usado para transmitir información entre partes como método de autenticación.
- **UserDetailsService:** Interfaz usada por Spring Security para cargar datos específicos del usuario.

- **Security Filter Chain:** Cadena de filtros que intercepta peticiones para aplicar reglas de seguridad.
- **BCrypt:** Algoritmo de hashing seguro usado para encriptar contraseñas.
- **CORS (Cross-Origin Resource Sharing):** Mecanismo que controla el acceso a recursos desde otros dominios.
- **REST API:** Estilo de arquitectura que usa HTTP para exponer recursos como servicios web.
- **DTO (Data Transfer Object):** Objeto que se usa para transferir datos entre capas, sin exponer directamente entidades.
- **JPA (Java Persistence API):** Estándar para el mapeo objeto-relacional (ORM).
- **Hibernate:** Implementación popular de JPA usada en Spring.
- **application.properties / application.yml:** Archivos de configuración de la aplicación.
- **Maven:** Herramientas para la construcción de proyectos y manejo de dependencias.
- **pom.xml:** Archivos donde se definen las dependencias y configuraciones del proyecto.

Bibliografía o Anexos

- Sánchez, D., & López, A. (2020). *Sistemas de información aplicados a la gestión de ventas en negocios gastronómicos*. Editorial Académica Española.
- Popeyes Louisiana Kitchen. (2024). *About us*. <https://www.popeyes.com/about-us>

Conclusiones

- Automatización exitosa de procesos clave: El desarrollo del sistema de punto de venta (POS) permitió automatizar la gestión de pedidos, el control de inventario, la atención al cliente y la emisión de reportes, lo cual optimiza significativamente la operación diaria del restaurante Foodly.
- Mejora en la experiencia del usuario: La aplicación web ofrece una experiencia más ágil y moderna tanto para los clientes como para los empleados. La interfaz intuitiva y las funcionalidades bien distribuidas facilitan el uso del sistema.
- Control y gestión centralizada: Gracias a la arquitectura implementada (con roles diferenciados para administrador, cajero y cocina), se logró establecer un flujo de trabajo ordenado y seguro, permitiendo un mejor control del stock, pedidos y ventas.
- Escalabilidad y seguridad aseguradas: El uso de herramientas como Spring Boot, Spring Security y JWT refuerza la seguridad del sistema y permite una base sólida para futuras expansiones o integraciones, asegurando la escalabilidad del proyecto.
- Base para una estrategia digital: El sistema no solo cumple una función operativa, sino que también actúa como punto de partida para consolidar la presencia digital de Foodly, elemento esencial frente a la fuerte competencia del sector.

Recomendaciones

- Implementar una aplicación móvil: Para alcanzar una mayor cobertura de mercado, se recomienda desarrollar una app móvil con funcionalidades similares o complementarias a la versión web.
- Integrar métodos de pago en línea: Incorporar pasarelas de pago (como Yape, Plin o tarjetas) facilitaría el proceso de compra y atraería a más clientes acostumbrados a pagos digitales.
- Monitoreo continuo y mejoras: Se debe realizar un monitoreo constante del sistema para identificar errores, necesidades de mejora o nuevas funcionalidades que se ajusten a los cambios del negocio.

- Capacitación constante al personal: Capacitar a los empleados sobre el uso adecuado del sistema garantizará un mejor aprovechamiento de sus herramientas y reducirá los errores operativos.
- Ampliar funcionalidades de fidelización: Incluir funcionalidades como historial de pedidos, puntos acumulables, promociones personalizadas o encuestas de satisfacción puede aumentar la lealtad del cliente.
- Refuerzo en marketing digital: Acompañar este sistema con una estrategia activa de marketing digital (redes sociales, campañas segmentadas) permitirá posicionar mejor la marca y atraer más clientes.