



# MEMORIA - GESTIÓN DE TIENDA

- v.1.0

14 de diciembre de 2022  
Modelado de Software  
Grupo E – Ingeniería del Software



## INTEGRANTES DEL EQUIPO

Alberto Herrera García
Alfonso Burguete Gotor
Axel Kahou He Expósito
Beatriz Espinar Aragón
Daniel Barroso Casado
Jose María Gálvez Gómez
Miriam Choy Castillo
Nicolás Espinosa Mooser
Santiago Moral Santorio
Sara Sacó Baños
Steven Mallqui Aguilar
Unai Piris Ibáñez

# CONTENIDO

1. Introducción.....	4
2. Desarrollo .....	5
2.1. Resumen.....	5
2.2. Proceso .....	6
2.3. Factores de calificación .....	13
3. Producto.....	14
3.1. Casos de uso .....	14
3.2. Interfaces externas .....	15
4. Arquitectura.....	15
4.1. Patrones.....	19
5. Modelo .....	20
5.1. Diagramas de clase .....	21
5.2. Diagramas de secuencia .....	24
5.3. Diagrama de despliegue .....	32
6. Estructura del código.....	34
7. Pruebas.....	35
8. Recursos .....	37

## 1. Introducción

La finalidad de este documento es presentar un informe sobre la segunda parte del proyecto *Plateau* y cómo ha sido llevada a cabo. Esta segunda parte consiste en una aplicación para gestionar la tienda de un cine a nivel administrativo, tal y como se detalla en la Especificación de Requisitos Software (SRS).

El proyecto *Plateau* ha sido desarrollado durante los meses de noviembre y diciembre de 2022, y permite gestionar los productos que se venden en la tienda de un cine, con los proveedores que los suministran, las ventas que se realizan y sus correspondientes clientes. Además, permite gestionar los empleados que trabajan en la tienda y los departamentos a los que pertenecen.

Este documento queda dividido en secciones, cuyo contenido se explica brevemente a continuación:

- Desarrollo. En este apartado se describe el proceso de desarrollo del software, cómo se ha ido construyendo y cómo se ha realizado el reparto de tareas.
- Producto. En este apartado se muestra cómo ha quedado el producto final en comparación a lo establecido en la SRS, qué funcionalidades se han implementado y qué interfaces se han diseñado de cara al usuario.
- Arquitectura. En este apartado se especifica la arquitectura empleada, junto a los patrones que se han implementado para un correcto modelado del software.
- Modelo. En este apartado se presentan los distintos diagramas de clase, secuencia y despliegue que se han realizado antes de pasar al desarrollo del código, junto a algunos ejemplos visuales.
- Estructura del código. En este apartado se explica brevemente la estructura de carpetas seleccionada para mantener las clases Java ordenadas y dónde se encuentran otros archivos relevantes.
- Pruebas. En este apartado se describen las distintas pruebas que se han llevado a cabo para asegurar el correcto funcionamiento del software tanto a nivel de requisitos como a nivel puramente funcional.
- Recursos. En este apartado se comentan los principales recursos software empleados para poder desarrollar el proyecto.

Así, el lector podrá tener una visión global y, a su vez, detallada, de lo que ha supuesto el desarrollo de la segunda parte de *Plateau*, completando así el proyecto en su totalidad.

## 2. Desarrollo

### 2.1. Resumen

Esta segunda parte de *Plateau* ha dado comienzo tras la entrega y corrección de la primera parte del proyecto, correspondiente a la gestión de un cine. Esta corrección dio lugar a diversas modificaciones que se han tenido en cuenta para mejorar la calidad del software de la gestión de la tienda del cine. Además, antes de comenzar con el modelado de la aplicación, se revisaron tanto la SRS como el Modelo del Dominio, para comprobar que todo se seguía ajustando a los intereses iniciales.

Una vez se hubo dado el visto bueno de los casos de uso, y conociendo el procedimiento seguido en la parte DAO del proyecto, el equipo de *Plateau* pudo comenzar a elaborar el modelo del software. En esta ocasión, se decidió dividir el equipo en dos subgrupos, encargados de capas de la arquitectura distintas. El primer grupo ha sido encargado de elaborar, tanto en modelo como en código, la capa de presentación, que compartía ciertas similitudes con la primera parte del proyecto. Por otro lado, la segunda mitad del equipo se ha encargado de la capa de negocio (e integración), aplicando los conceptos de JPA, además de supervisar durante todo el desarrollo el trabajo del primer grupo. Dentro de cada subgrupo, el trabajo se dividía entre los 6 integrantes de acuerdo a los 6 módulos existentes en el dominio de la aplicación.

Así, con la experiencia adquirida tras el desarrollo de la primera parte de *Plateau*, en un par de semanas quedó finalizado todo el modelo, tanto los diagramas de clase como los de secuencia. Esto, sin embargo, no significa que el modelo no se fuera modificando a medida que avanzaba el proyecto, dado que los cambios son inherentes al proceso de desarrollo de software.

De este modo, el equipo pudo transformar el modelo en código y proceder a su elaboración. Antes de todos estos pasos, por supuesto, se instalaron los servicios necesarios para poder utilizar JPA y se reorganizaron las carpetas de trabajo, pues en la gestión del cine se produjeron ciertos problemas relativos a las ramas y otras carpetas. En el desarrollo del código se repartieron los módulos y las capas del mismo modo que para el modelo.

Por último, se realizaron todas las pruebas que se consideraron necesarias para poder aumentar todo lo posible el nivel de fiabilidad del software, y se comprobó con la SRS que todo actuaba de acuerdo con los requisitos especificados. Además, se hicieron diversos retoques para mejorar la estética de las interfaces de usuario y para mejorar la legibilidad del código, así como otras mejoras y correcciones.

Por supuesto, se ha mantenido el protocolo de reuniones que se llevó a cabo en la primera parte del equipo, para permitir al equipo estar al día del avance del proyecto, informar sobre posibles errores o contingencias que surgían, y planificar los próximos pasos. Esto se hacía aún más necesario al estar dividido el equipo por capas, pues estas deben conectarse de forma correcta y para ello se requiere de una coordinación entre miembros del equipo. De igual forma que las reuniones, se han utilizado otras plataformas como *Whastapp* o *Discord* para asegurar una comunicación activa, constante y eficaz, aumentando la visibilidad y mejorando el proceso de desarrollo, y fomentando continuamente el trabajo en equipo.

Además, en lo relativo a los cambios, es relevante mencionar la naturaleza ágil que ha mantenido el proceso de desarrollo. Esto es, se ha mantenido actualizado todo el material que se ha ido generando,

tanto el Modelo del Dominio, como la SRS y los casos de uso, como el modelo y como cualquier otro producto generado. Así, los cambios se han aceptado como parte natural e inherente al desarrollo del software, y el equipo ha sido capaz de adaptarse a todo tipo de contratiempos y modificaciones.

El proyecto concluye con la revisión y corrección por parte del profesor de la asignatura, que servirán como aprendizaje personal de cada integrante, y permitirán desarrollar futuros proyectos software con una calidad adecuada y cada vez más alta.

## 2.2. Proceso

En este apartado se incluye una tabla con todas las tareas que se han ido realizando durante el desarrollo de *Plateau*, con las fechas más relevantes y los encargados de dichas tareas. De este modo, el lector podrá visualizar con un alto nivel de detalle cómo ha evolucionado el proyecto a lo largo de estos meses y qué pasos se han ido dando hasta obtener el producto final que se presenta.

TAREA	AUTOR	FECHA
Corrección tras entrega DAO		17/11
Corrección SRS (parte JPA)	· Beatriz Espinar	-
Los casos de uso “mostrar” también muestran el atributo activo		-
Los casos de uso “listar” también muestran el ID y el atributo activo		-
Añadir atributo DNI a Empleado ( <i>primary key</i> ) y actualizar el modelo del dominio		-
En los casos de uso “modificar” no es necesario que se modifiquen los campos		-
Añadir a “Mostrar venta en detalle” la información de las <i>lineasVenta</i>		-
Añadir caso de uso “Listar ventas por empleado”		-
Atributo Dias (Empleado Parcial) → EurosPM		-
Arreglar repositorio y carpetas de trabajo	· Nicolás Espinosa · Santiago Moral	-
Modelo		11/11-24/11
Diagramas de clase negocio		-
Departamento	· Beatriz Espinar	-
Empleado	· Unai Piris	-
Venta	· Steven Mallqui	-
ClienteJPA	· Alberto Herrera	-
Producto	· Nicolás Espinosa	-
Proveedor	· Santiago Moral	-
Diagramas de clase presentación		11/11-19/11
Departamento	· Miriam Choy	-
Empleado	· Jose María Gálvez	-
Venta	· Alfonso Burguete	-

TAREA	AUTOR	FECHA
ClienteJPA	• Daniel Barroso	-
Proveedor	• Sara Sacó • Beatriz Espinar • Santiago Moral	-
Producto	• Axel Kahou	-
Diagramas secuencia negocio ClienteJPA	• Alberto Herrera	-
Alta		-
Baja		-
Modificar		-
Mostrar		-
Listar		-
Diagrama secuencia negocio Baja Departamento	• Beatriz Espinar	-
Diagramas secuencia negocio Producto		-
Alta	• Unai Piris	-
Baja	• Unai Piris	-
Modificar	• Steven Mallqui	-
Mostrar	• Steven Mallqui	-
Listar	• Unai Piris	-
Listar comida	• Unai Piris	-
Listar bebida	• Nicolás Espinosa	-
Listar por proveedor	• Nicolás Espinosa	-
Listar por venta	• Steven Mallqui	-
Añadir proveedor	• Nicolás Espinosa	-
Quitar proveedor	• Nicolás Espinosa	-
Diagramas secuencia negocio Venta		-
Cerrar	• Steven Mallqui	-
Devolver	• Beatriz Espinar	-
Mostrar	• Santiago Moral	-
Mostrar en detalle	• Beatriz Espinar	-
Listar	• Santiago Moral	-
Listar por producto	• Santiago Moral	-
Listar por empleado	• Beatriz Espinar	-

TAREA	AUTOR	FECHA
Listar por cliente	· Santiago Moral	-
Diagramas secuencia presentación Command Producto		20/11-22/11
Alta	· Miriam Choy	-
Baja	· Miriam Choy	-
Modificar	· Miriam Choy	-
Mostrar	· Jose María Gálvez	-
Listar	· Jose María Gálvez	-
Listar comida	· Alfonso Burguete	-
Listar bebida	· Alfonso Burguete	-
Listar por proveedor	· Alfonso Burguete	-
Listar por venta	· Daniel Barroso	-
Añadir proveedor	· Daniel Barroso	-
Quitar proveedor	· Daniel Barroso	-
Diagramas secuencia presentación Command Venta		20/11-22/11
Cerrar	· Jose María Gálvez	-
Devolver	· Jose María Gálvez	-
Mostrar	· Alfonso Burguete	-
Mostrar en detalle	· Alfonso Burguete	-
Listar	· Jose María Gálvez	-
Listar por producto	· Jose María Gálvez	-
Listar por cliente	· Jose María Gálvez	-
Listar por empleado	· Alfonso Burguete	-
Diagramas secuencia presentación Paneles Producto		20/11-22/11
Alta	· Sara Sacó	-
Baja	· Miriam Choy	-
Modificar	· Sara Sacó	-
Mostrar	· Jose María Gálvez	-
Listar	· Jose María Gálvez	-
Listar comidas	· Sara Sacó	-
Listar bebidas	· Sara Sacó	-
Listar por proveedor	· Sara Sacó	-
Listar por venta	· Steven Mallqui	-
Añadir proveedor	· Steven Mallqui	-
Quitar proveedor	· Steven Mallqui	-



TAREA	AUTOR	FECHA
Diagramas secuencia presentación Paneles Venta		20/11-22/11
Cerrar	· Steven Mallqui	-
Devolver	· Axel Kahou	-
Mostrar	· Axel Kahou	-
Mostrar en detalle	· Steven Mallqui	-
Listar	· Daniel Barroso	-
Listar por producto	· Daniel Barroso	-
Listar por cliente	· Daniel Barroso	-
Listar por empleado	· Daniel Barroso	-
Diagrama clase ErrorHandler	· Beatriz Espinar	-
Diagrama clase y secuencia EMFSingleton	· Beatriz Espinar	-
Corrección diagramas clase negocio	· Beatriz Espinar	26/11
Revisión y anotación de errores diagramas secuencia negocio	· Beatriz Espinar	26/11-27/11
Corrección diagramas secuencia negocio	· Alberto Herrera · Beatriz Espinar · Nicolás Espinosa · Santiago Moral · Steven Mallqui	26/11-27/11
Revisión y anotación de errores diagramas clase presentación	· Beatriz Espinar	27/11-29/11
Corrección diagramas clase presentación	· Beatriz Espinar · Nicolás Espinosa · Santiago Moral · Steven Mallqui	27/11-29/11
Corrección diagramas secuencia comandos	· Beatriz Espinar	04/12
Corrección diagramas secuencia paneles	· Steven Mallqui · Beatriz Espinar	03/12-04/12
Código Presentación		29/12-04/12
Paneles (sin ActionListener ni métodos, sólo vistas)		29/12-03/12
Departamento	· Miriam Choy	-
Empleado	· Jose María Gálvez	-

TAREA	AUTOR	FECHA
Venta	<ul style="list-style-type: none"> <li>• Alfonso Burguete</li> <li>• Daniel Barroso</li> </ul>	-
ClienteJPA	<ul style="list-style-type: none"> <li>• Daniel Barroso</li> <li>• Santiago Moral</li> </ul>	-
Producto	<ul style="list-style-type: none"> <li>• Axel Kahou</li> </ul>	-
Proveedor	<ul style="list-style-type: none"> <li>• Sara Sacó</li> </ul>	-
Comandos y ErrorHandler de cada entidad		04/12
Departamento	<ul style="list-style-type: none"> <li>• Beatriz Espinar</li> </ul>	-
Empleado	<ul style="list-style-type: none"> <li>• Alberto Herrera</li> </ul>	-
Venta	<ul style="list-style-type: none"> <li>• Steven Mallqui</li> </ul>	-
ClienteJPA	<ul style="list-style-type: none"> <li>• Santiago Moral</li> </ul>	-
Producto	<ul style="list-style-type: none"> <li>• Unai Piris</li> </ul>	-
Proveedor	<ul style="list-style-type: none"> <li>• Nicolás Espinosa</li> </ul>	-
ActionListener y métodos update(), hasError()... de los Paneles		04/12
Departamento	<ul style="list-style-type: none"> <li>• Beatriz Espinar</li> </ul>	-
Empleado	<ul style="list-style-type: none"> <li>• Alberto Herrera</li> </ul>	-
Venta	<ul style="list-style-type: none"> <li>• Steven Mallqui</li> </ul>	-
ClienteJPA	<ul style="list-style-type: none"> <li>• Santiago Moral</li> </ul>	-
Producto	<ul style="list-style-type: none"> <li>• Axel Kahou</li> </ul>	-
Proveedor	<ul style="list-style-type: none"> <li>• Nicolás Espinosa</li> </ul>	-
EventEnum, ContextEnum, Command, ApplicationController, ApplicationControllerImp, Dispatcher, DispatcherImp, EntityEnumJPA, ErrorHandlerManagerJPA, CommandFactory, CommandFactoryImp, ErrorHandler	<ul style="list-style-type: none"> <li>• Beatriz Espinar</li> <li>• Steven Mallqui</li> </ul>	04/12
Corrección paneles	<ul style="list-style-type: none"> <li>• Santiago Moral</li> <li>• Nicolás Espinosa</li> <li>• Alberto Herrera</li> <li>• Steven Mallqui</li> <li>• Beatriz Espinar</li> <li>• Unai Piris</li> </ul>	04/12
Código Negocio		29/12-08/12
AS, ASImp, Transfers, Entities		-

TAREA	AUTOR	FECHA
Departamento	• Beatriz Espinar • Steven Mallqui	-
Empleado	• Alberto Herrera	-
Venta	• Steven Mallqui • Beatriz Espinar	-
ClienteJPA	• Santiago Moral	-
Producto	• Axel Kahou	-
Proveedor	• Nicolás Espinosa	-
ASFactory y ASFactoryImp	• Santiago Moral	-
Revisión y anotación de errores negocio	• Beatriz Espinar	-
Corrección negocio	• Beatriz Espinar • Nicolás Espinosa • Alberto Herrera • Santiago Moral	-
Resto del código y testeo		-
EMFSingleton y EMFSingletonImp	• Santiago Moral • Beatriz Espinar	-
Archivo persistence.xml	• Miriam Choy • Axel Kahou	06/12
Tests JUnit Negocio		06/12-12/12
Departamento	• Miriam Choy • Santiago Moral • Nicolás Espinosa	-
Empleado	• Nicolás Espinosa • Santiago Moral	-
Venta	• Alfonso Burguete • Nicolás Espinosa	-
ClienteJPA	• Nicolás Espinosa • Daniel Barroso	-
Producto	• Nicolás Espinosa	-
Proveedor	• Santiago Moral • Sara Sacó	-
Depurar	• Santiago Moral • Beatriz Espinar	09/12-13/12
Pruebas de aceptación	• Beatriz Espinar	10/12-13/12
Corrección tras pruebas de aceptación	• Beatriz Espinar	10/12-13/12
Memoria y otros		17/11-14/12
Apuntar progreso y reparto de tareas	• Beatriz Espinar	-

TAREA	AUTOR	FECHA
Redacción apartados 1, 2, 3, 5, 6 y 7	· Beatriz Espinar	-
Redacción apartados 4 y 8	· Beatriz Espinar · Alberto Herrera	-
Capturas ejemplos	· Miriam Choy · Beatriz Espinar	-
Formulario	· Alberto Herrera · Unai Piris	13/12
Revisión modelo	· Beatriz Espinar	13/12-14/12
Diagrama despliegue	· Steven Mallqui	14/12
Modelo de datos	· Beatriz Espinar	14/12
Diagrama componentes	· Alberto Herrera	14/12

### 2.3. Factores de calificación

A continuación, se indica el factor de calificación que el equipo de *Plateau* ha decidido asignar a cada miembro del equipo. Se cuenta con 12 puntos, a repartir entre los 12 integrantes, en función de la cantidad de trabajo realizado. Dado que el equipo no lograba llegar a un acuerdo, se realizó una votación y se llegó a la siguiente conclusión:

Nombre y Apellidos	Factor de calificación
Alberto Herrera García	1.15
Alfonso Burguete Gotor	0.85
Axel Kahou He Expósito	1
Beatriz Espinar Aragón	1.15
Daniel Barroso Casado	0.85
Jose María Gálvez Gómez	0.85
Miriam Choy Castillo	0.85
Nicolás Espinosa Mooser	1.15
Santiago Moral Santorio	1.15
Sara Sacó Baños	0.85
Steven Mallqui Aguilar	1.15
Unai Piris Ibáñez	1

### 3. Producto

#### 3.1. Casos de uso

El equipo de *Plateau* ha logrado completar<sup>1</sup> el 100% de las funcionalidades que se pretendían implementar tras la SRS. Estas funcionalidades corresponden a los siguientes casos de uso<sup>2</sup>:

##### Módulo Departamento

- Alta departamento
- Baja departamento
- Modificar departamento
- Mostrar departamento
- Listar departamentos

##### Módulo Empleado

- Alta empleado
- Baja empleado
- Modificar empleado
- Mostrar empleado
- Listar empleados
- Listar empleados tiempo completo
- Listar empleados tiempo parcial
- Listar empleados por departamento

##### Módulo Proveedor

- Alta proveedor
- Baja proveedor
- Modificar proveedor
- Mostrar proveedor
- Listar proveedores
- Listar proveedores por producto

##### Módulo Producto

- Alta producto
- Baja producto
- Modificar producto
- Mostrar producto
- Listar productos
- Listar productos comida
- Listar productos bebida
- Listar productos por proveedor
- Listar productos por venta
- Añadir proveedor a producto
- Quitar proveedor a producto

##### Módulo Cliente

- Alta cliente
- Baja cliente
- Modificar cliente
- Mostrar cliente
- Listar clientes

##### Módulo Venta

- Abrir venta
- Añadir producto a venta
- Quitar producto de venta
- Cerrar venta
- Devolver venta

<sup>1</sup> Completar se define como tener elaborado el modelo, tener un código funcional que implemente la funcionalidad, y haber pasado todas las pruebas descritas en el apartado 7 de este documento.

<sup>2</sup> El detalle de lo que hace cada caso se puede consultar en el documento de Especificación de Requisitos Software.

- Mostrar venta
- Mostrar venta en detalle
- Listar ventas
- Listar ventas por cliente
- Listar ventas por producto
- Listar ventas por empleado

### 3.2. Interfaces externas

A continuación, se adjuntan capturas del producto final para algunos de sus casos de uso, incluyendo la vista inicial al seleccionar el módulo Empleado (Ilustración 1), el caso de uso Modificar producto, en el caso particular de una bebida (Ilustración 2), el caso de uso Listar proveedores por producto (Ilustración 3) y el caso de uso Mostrar venta en detalle (Ilustración 4).

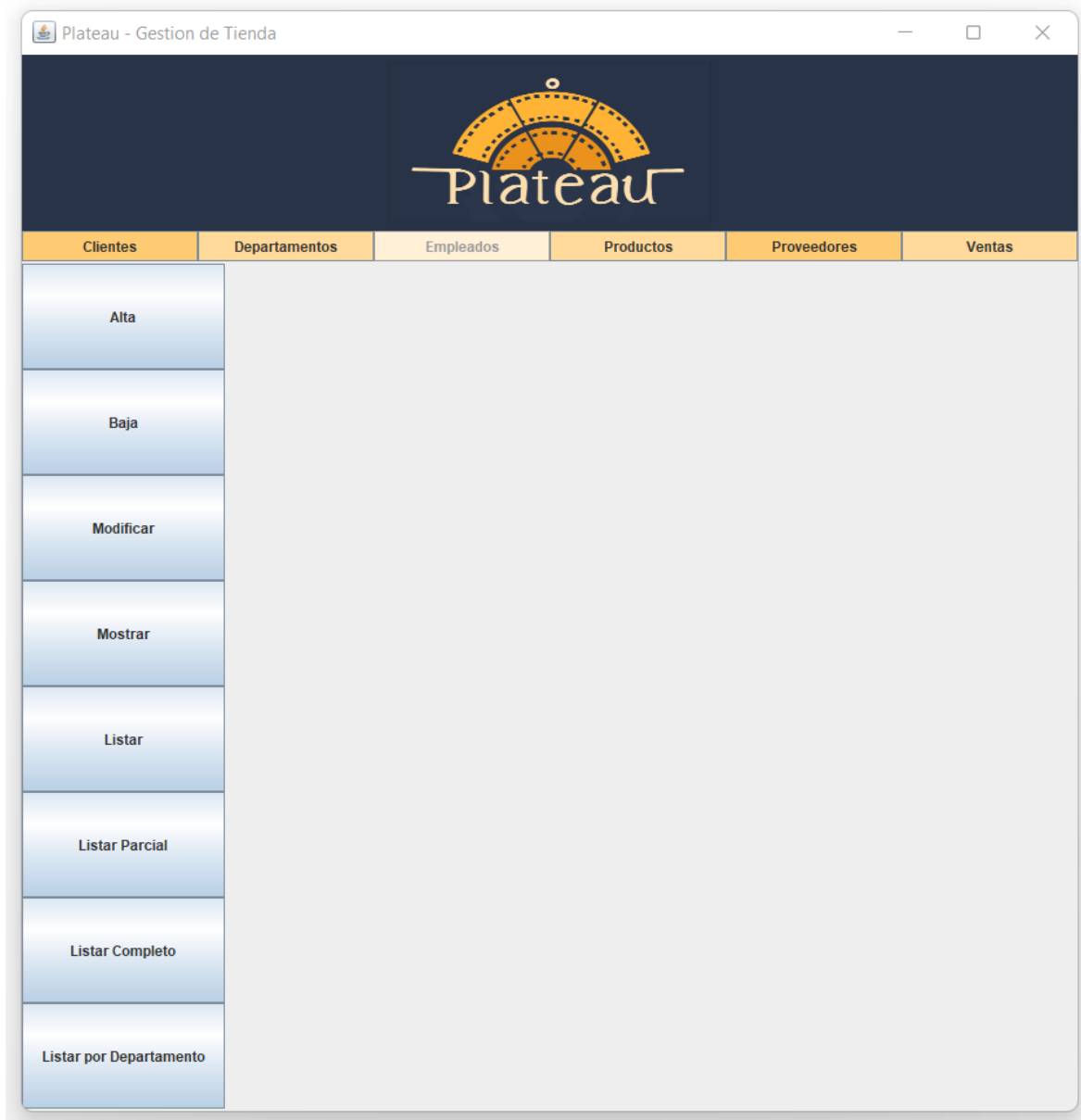



Ilustración 1. Interfaz principal Empleado

Plateau - Gestion de Tienda



Cientes

Departamentos

Empleados

Productos

Proveedores

Ventas

Alta

Baja

Modificar

Mostrar

Listar

Listar Comidas

Listar Bebidas

Listar por Proveedor

Listar por Venta

Añadir proveedor

Quitar proveedor

MODIFICAR PRODUCTO

ID: 1

Buscar

Nombre:

Tamano:

Precio:


Stock:

Modificar

Ilustración 2. Interfaz Modificar Producto



Plateau - Gestion de Tienda



Cientes

Departamentos

Empleados

Productos

Proveedores

Ventas

Alta

Baja

Mostrar

Modificar

Listar

Listar Por Producto

# LISTAR POR PRODUCTO

ID Producto:

ID	Telefono	Nombre	CIF
2	641018656	Gastromundo	20832017C
5	934970077	Euroestrellas	B65905721
1	910270632	Jalanda Stock S.L.	B87226205
3	972526061	Alregi SI	B17419201
4	972490428	Delicat Plural	B55235873

Ilustración 3. Interfaz Listar Proveedores por Producto

*Ilustración 4. Interfaz Mostrar Venta en detalle*

## 4. Arquitectura

La arquitectura empleada en el proyecto ha sido multicapa, que divide el software en 3 capas principales (aunque también podrían considerarse la de clientes y la de recursos):

1. Capa de presentación. Encapsula toda la lógica de presentación necesaria para dar servicio a los clientes que acceden al sistema. En esta capa se implementa la interfaz gráfica y los correspondientes *ActionListener* de los componentes que crean los *transfers* y los envían al controlador, también incluido en esta capa y que sirve como intermediario entre esta capa y la de negocio. Además, se incluyen en esta capa los comandos, encargados de invocar al servicio de aplicación, traduciendo el contexto que recibe del controlador, y el despachador que redirecciona a las vistas en función del contexto recibido.
2. Capa de negocio. Proporciona los servicios del sistema e implementa la lógica del negocio (procesamiento, *workflow*, reglas de negocio y datos). En esta capa se encuentran los *transfers* para comunicarse con presentación y las entidades JPA que se traducen en tablas de la BD, así como los servicios de aplicación, encargados de implementar dicha lógica y asegurar que los datos de entrada cumplen las reglas de negocio y los requisitos del sistema. Además, controla el inicio y fin de transacciones haciendo uso de los *EntityManager*, que gestionan las entidades y se comunicarán con el almacén persistente.
3. Capa de integración. Responsable de la comunicación con los recursos y sistemas externos. En el caso de *Plateau*, interactúa con la base de datos, por medio de los *EntityManager*, encargados de gestionar el contexto de persistencia, y generados por su correspondiente factoría.

Así, mediante esta arquitectura, se promueve el aislamiento entre las capas, cada una encargada de su ámbito, de forma que los cambios, como modificar una regla de negocio, no afectan a las demás. También favorece la encapsulación y la reutilización del código, y proporciona mayor seguridad y un sistema con mayor rendimiento y muy empaquetado.

### 4.1. Patrones

Seguidamente se describen brevemente los patrones software empleados en este proyecto, algunos de los cuales fueron ya mencionados en la Especificación de Requisitos Software, pues se trataban de requerimientos imprescindibles a nivel lógico del sistema.

#### ❖ Patrón *Transfer Object*

Es un conocido patrón j2ee que porta datos sobre alguna entidad del modelo entre capas, encapsulando sus atributos. Es utilizado (con operaciones mutadoras y accesoras) en todo nuestro proyecto, para guardar en objetos nuestras entidades del modelo.

#### ❖ Patrón *Transfer Object Assembler (TOA)*

Se trata de un patrón que compone objetos *Transfer* a partir de otras fuentes de datos (comúnmente, otros *transfers*), así, simplifica algunas peticiones costosas haciendo menos llamadas. En nuestro proyecto, el módulo Venta hace uso de este patrón para evitar concatenaciones de *Transfers*, como proponía Fowler.

### ❖ Patrón *Application Service (AS)*

Se trata de un patrón que hemos aplicado en la capa de negocio, para centralizar la lógica de nuestro programa. Evalúa las condiciones estipuladas en nuestros casos de uso, a partir de un transfer generado en los *ActionListener* de cada panel. Los casos de uso tienen asociada una operación de negocio en cada Servicio de aplicación particular (particular a su módulo). Además, añade una capa más a negocio y permite un código reutilizable y no duplicado.

### ❖ Patrón *Service to Worker*

La finalidad de este patrón es llevar a cabo el manejo de peticiones y la invocación de lógica de negocio antes de pasar el control a la vista. *Service to Worker* articula diversos patrones de presentación (controlador frontal, controlador de aplicación, ayudante de vista). Sin embargo, en esta parte del proyecto no se ha implementado la versión original del patrón, sino una adaptada. Por ejemplo, no se utiliza el patrón *ViewHelper* ni el *FrontController*.

### ❖ Patrón *Context*

La finalidad de este patrón es evitar utilizar información del sistema específica del protocolo fuera de su contexto relevante, se encarga de encapsular el estado de una forma independiente del protocolo para ser compartida por toda la aplicación

### ❖ Patrón *Command*

La finalidad de este patrón es encapsular una petición en un objeto, permitiendo así parametrizar a los clientes con diferentes peticiones, hacer cola o llevar un registro de las peticiones, y poder deshacer las operaciones.

### ❖ Patrón *Singleton*

La finalidad de este patrón es garantizar que una clase solo tenga una instancia (o un número controlado de instancias) proporcionando un punto de acceso global a ella.

### ❖ Patrón *Abstract Factory*

La finalidad de este patrón es proporcionar una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas.

### ❖ Patrón *Business Object*

La finalidad de este patrón es la de separar los datos de negocio de la lógica de negocio. Para ello, se construye un modelo del dominio compuesto por varios objetos de negocio (*Business Objects*) que permiten reflejar el dominio de la aplicación.

### ❖ Patrón *Domain Store*

La finalidad de este patrón es la de separar la persistencia del modelo de objetos. Se emplea para persistir de manera transparente un modelo de objetos, y resuelve los problemas de Concurrencia, Carga dinámica, Transaccionalidad y Persistencia.

## 5. Modelo

En esta sección se explica el modelo realizado, con un listado de los diagramas que se han elaborado en el proyecto y algún ejemplo visual.

### 5.1. Diagramas de clase

Por un lado, se han realizado diagramas de clase. Estos se han dividido en 3 carpetas, correspondientes a las 3 capas de la arquitectura multicapa: Integración, Negocio y Presentación.

#### ➤ Integración

En Integración sólo se incluye el diagrama correspondiente al EntityManagerFactorySingleton, dado que en esta parte del proyecto no se hace uso de los DAOs. Se incluye la ruta al diagrama a continuación, y se adjunta posteriormente una captura del diagrama en cuestión (Ilustración 5).

- *Modelos/Modelo de Diseño/Integración/EMFSingleton/Diagrama de Integración – EMFSingleton*

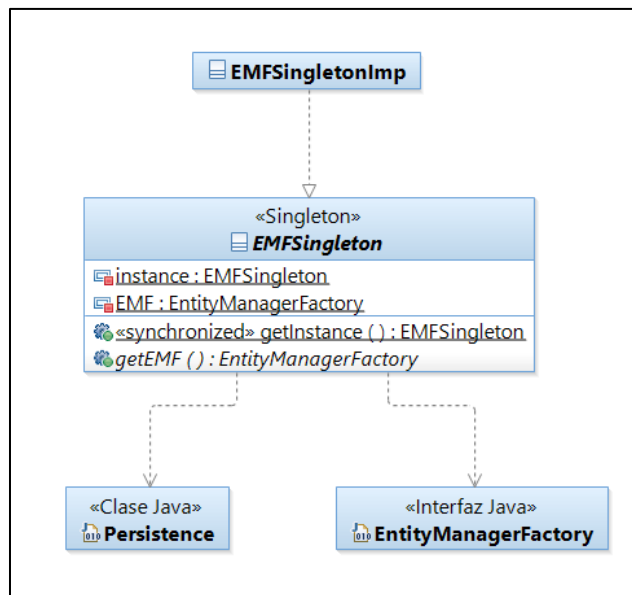


Ilustración 5. Diagrama clase EMFSingleton

#### ➤ Negocio

Al igual que en la primera parte del proyecto, se ha creado una carpeta para cada módulo, para tener organizados los diagramas. Aparte de los 6 módulos (ClienteJPA, Departamento, Empleado, Producto, Proveedor y Venta) se ha mantenido la carpeta para el ASFactory, al que se han añadido los AS correspondientes.

Así, se ha elaborado un diagrama de negocio para cada módulo, quedando los diagramas que se listan a continuación. Se adjunta posteriormente una captura del diagrama de Empleado (Ilustración 6), a modo de ejemplo.

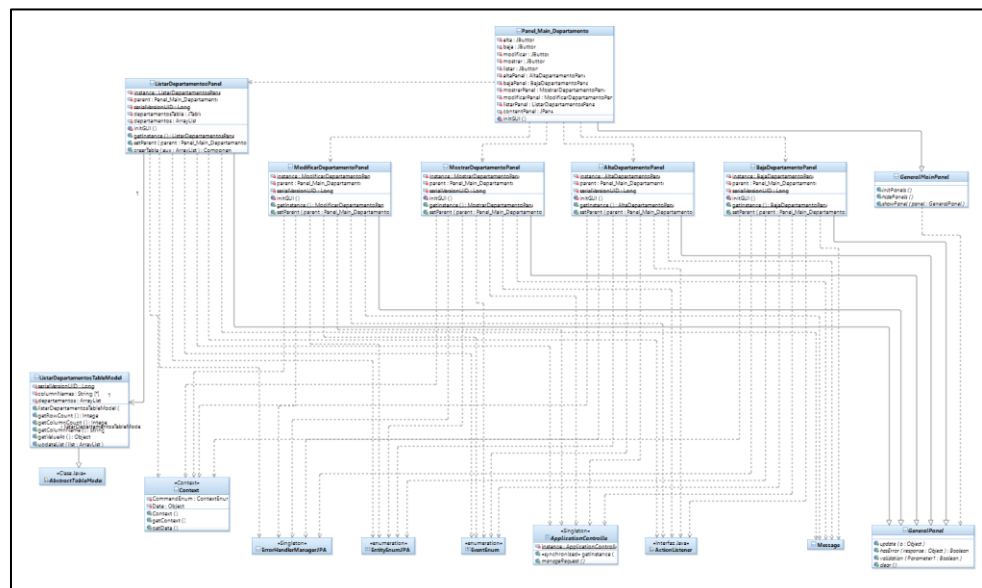
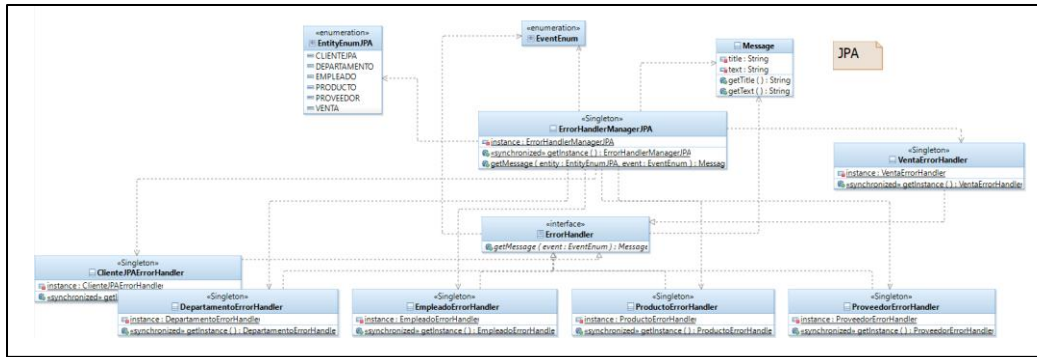
- [illegible]

➤ **Presentación**

Se han realizado, por tanto, los diagramas que se listan a continuación, y se adjuntan posteriormente una captura del diagrama de CommandVenta (Ilustración 7), el diagrama del ErrorHandler (Ilustración 8) y el diagrama de PanelDepartamento (Ilustración 9), a modo de ejemplo.

- [illegible]

23





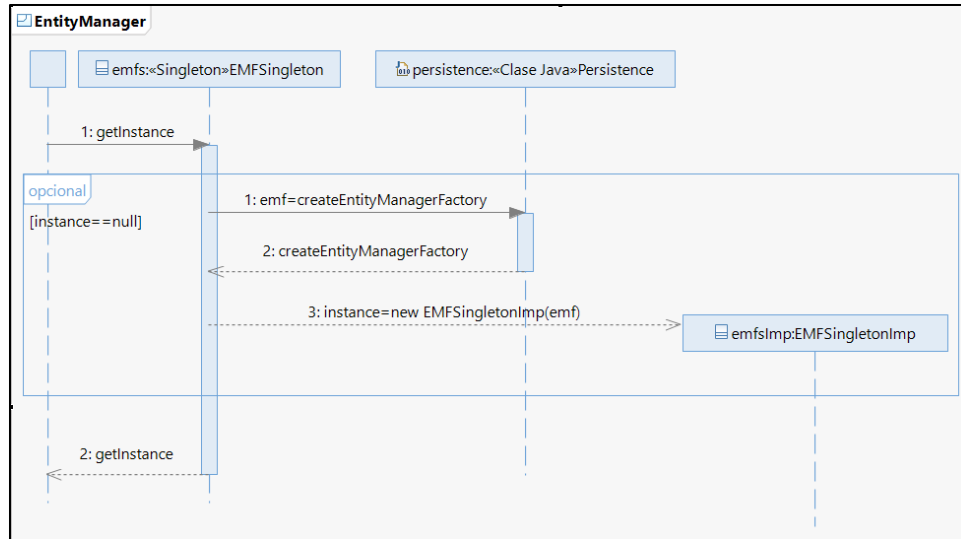


Ilustración 10. Diagrama secuencia EntityManager

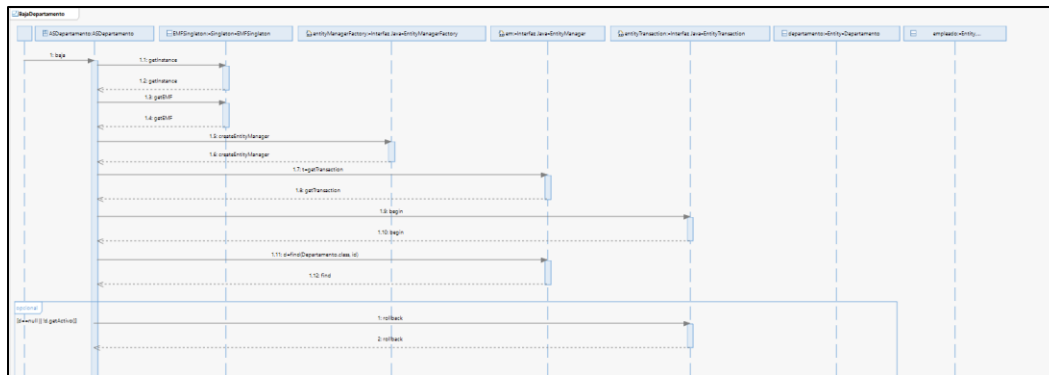
## ➤ Negocio

En este caso, no se han realizado los diagramas de secuencia de negocio de todos los módulos, puesto que varios resultaban innecesarios al tener excesivas similitudes con los demás. Por ello, se han seleccionado 3 módulos que destacaban por los siguientes motivos: en primer lugar, se han realizado diagramas de ClienteJPA, pues esta es una entidad básica y sirve como modelo para las demás. Por otro lado, se han realizado los diagramas de Producto, como ejemplo de entidad especializada (y por tener además los casos particulares de asociar entidades de una relación M a N), sirviendo como modelo para Empleado, que también es especializada. Por último, se han realizado los de Venta, por ser una entidad claramente distinguible de las demás, que opera de otro modo. Aparte de estos 3 módulos, se ha realizado el diagrama de “BajaDepartamento”, por su dependencia con la entidad Empleado al formar parte de una relación 1 a N.

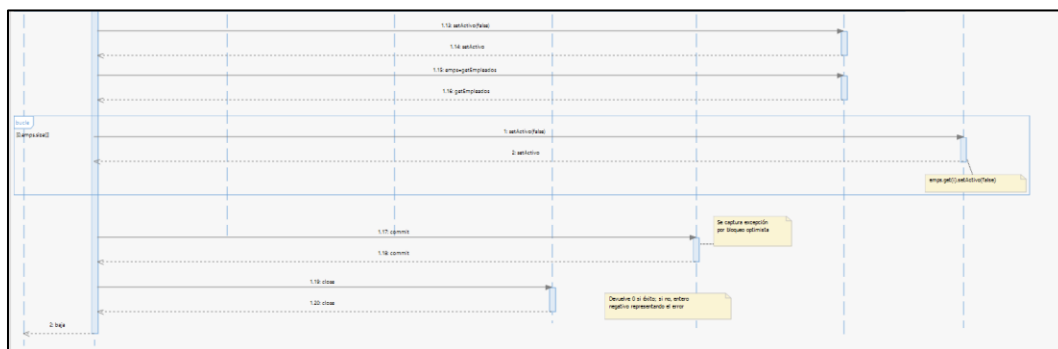
Queda entonces el listado de diagramas que sigue, y se adjuntan posteriormente una captura del diagrama de BajaDepartamento (Ilustraciones 11 y 12), el diagrama de AnadirProveedor (Ilustraciones 13, 14 y 15) y el diagrama de CerrarVenta (Ilustración 16, 17, 18 y 19), a modo de ejemplo.

- Modelos/Modelo de Diseño/Negocio/ClienteJPA/DiagramasDeSecuencia/AltaClienteJPA
- Modelos/Modelo de Diseño/Negocio/ClienteJPA/DiagramasDeSecuencia/BajaClienteJPA
- Modelos/Modelo de Diseño/Negocio/ClienteJPA/DiagramasDeSecuencia/ListarClientesJPA
- Modelos/Modelo de Diseño/Negocio/ClienteJPA/DiagramasDeSecuencia/ModificarClienteJPA
- Modelos/Modelo de Diseño/Negocio/ClienteJPA/DiagramasDeSecuencia/MostrarClienteJPA
- Modelos/Modelo de Diseño/Negocio/Producto/DiagramasDeSecuencia/AltaProducto
- Modelos/Modelo de Diseño/Negocio/Producto/DiagramasDeSecuencia/ AnadirProveedor
- Modelos/Modelo de Diseño/Negocio/Producto/DiagramasDeSecuencia/BajaProducto
- Modelos/Modelo de Diseño/Negocio/Producto/DiagramasDeSecuencia/ListarBebidas
- Modelos/Modelo de Diseño/Negocio/Producto/DiagramasDeSecuencia/ListarComidas
- Modelos/Modelo de Diseño/Negocio/Producto/DiagramasDeSecuencia/ListarProductos

- *Modelos/Modelo de Diseño/Negocio /Producto/DiagramasDeSecuencia/ListarProductosPorProveedor*
- *Modelos/Modelo de Diseño/Negocio /Producto/DiagramasDeSecuencia/ListarProductosPorVenta*
- *Modelos/Modelo de Diseño/Negocio/Producto/DiagramasDeSecuencia/ModificarProducto*
- *Modelos/Modelo de Diseño/Negocio/Producto/DiagramasDeSecuencia/MostrarProducto*
- *Modelos/Modelo de Diseño/Negocio/Producto/DiagramasDeSecuencia/ QuitarProveedor*
- *Modelos/Modelo de Diseño/Negocio/Venta/DiagramasDeSecuencia/CerrarVenta*
- *Modelos/Modelo de Diseño/Negocio/Venta/DiagramasDeSecuencia/DevolverVenta*
- *Modelos/Modelo de Diseño/Negocio/Venta/DiagramasDeSecuencia/ListarVentas*
- *Modelos/Modelo de Diseño/Negocio/Venta/DiagramasDeSecuencia /ListarVentasPorCliente*
- *Modelos/Modelo de Diseño/Negocio/Venta/DiagramasDeSecuencia/ ListarVentasPorEmpleado*
- *Modelos/Modelo de Diseño/Negocio/Venta/DiagramasDeSecuencia/ ListarVentasPorProducto*
- *Modelos/Modelo de Diseño/Negocio/Venta/DiagramasDeSecuencia/MostrarVenta*
- *Modelos/Modelo de Diseño/Negocio/Venta/DiagramasDeSecuencia/MostrarVentaEnDetalle*



*Ilustración 11. Diagrama secuencia BajaDepartamento (1)*



*Ilustración 12. Diagrama secuencia BajaDepartamento (2)*

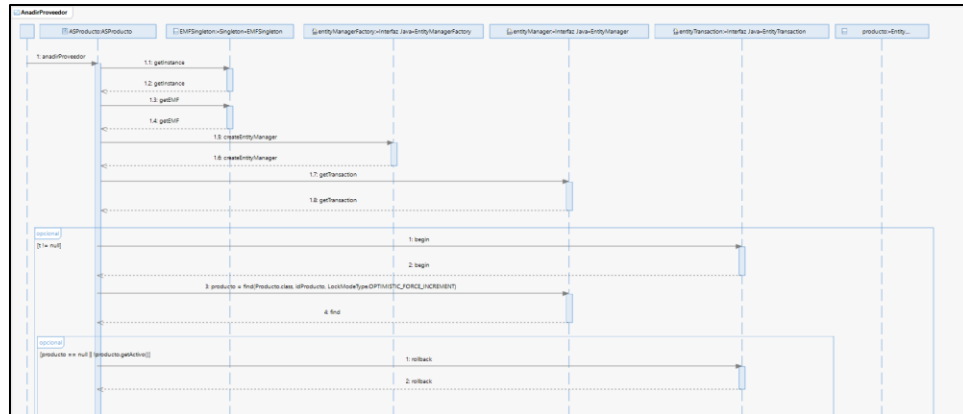


Ilustración 13. Diagrama secuencia AñadirProveedor (1)



Ilustración 14. Diagrama secuencia AñadirProveedor (2)

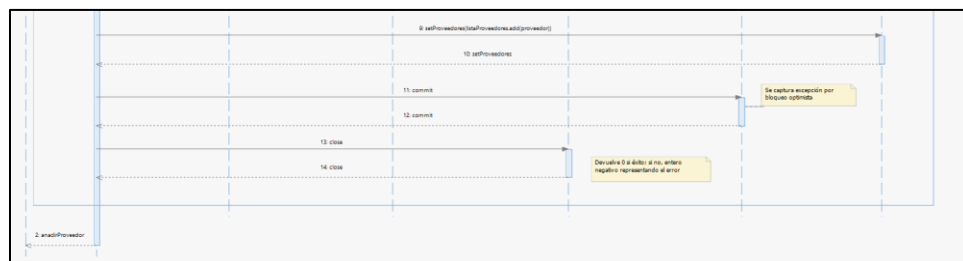


Ilustración 15. Diagrama secuencia AñadirProveedor (3)

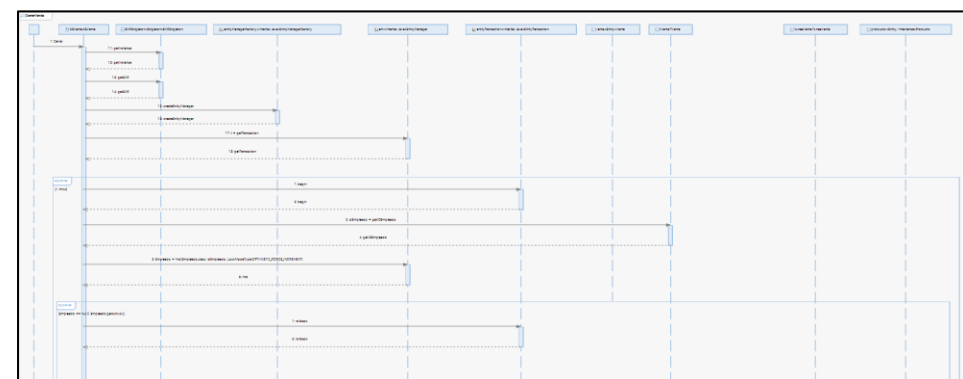


Ilustración 16. Diagrama secuencia CerrarVenta (1)

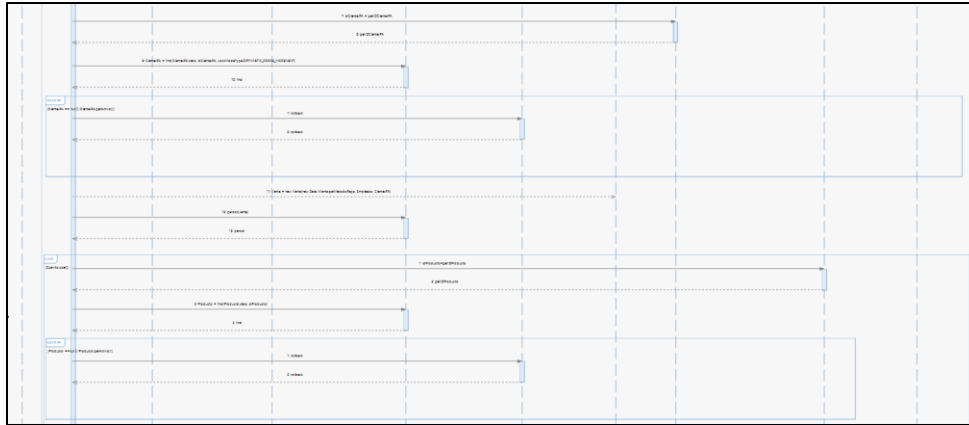


Ilustración 17. Diagrama secuencia CerrarVenta (2)

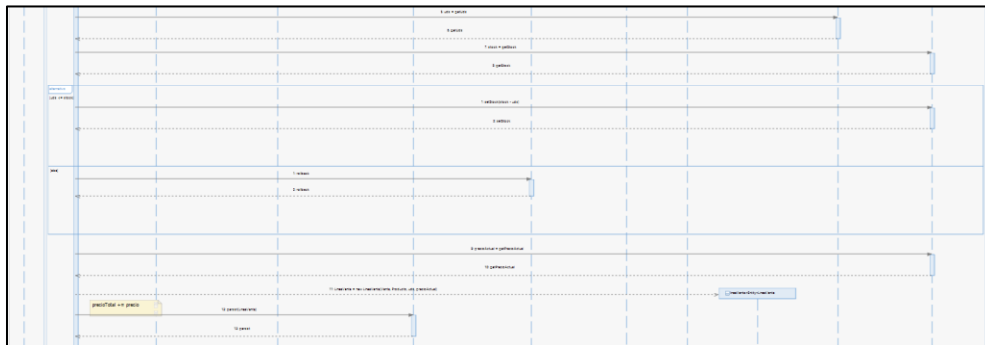


Ilustración 18. Diagrama secuencia CerrarVenta (3)

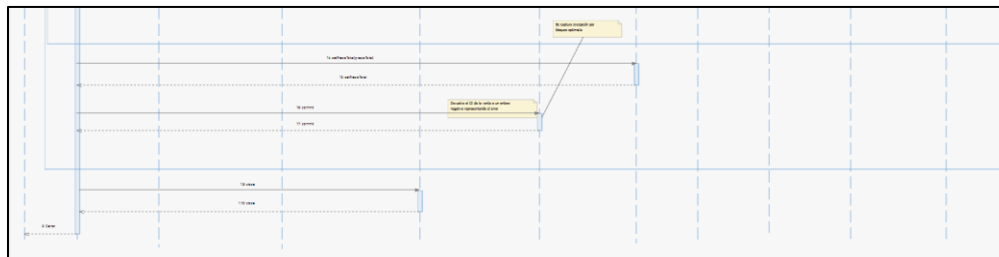


Ilustración 19. Diagrama secuencia CerrarVenta (4)

## ➤ Presentación

Por un lado, en la carpeta Command, se han elaborado los diagramas de los módulos Producto y Venta, por ser las entidades más características, así como los de los paneles en la carpeta Panels. Además, se ha reutilizado el diagrama de secuencia del ApplicationController, por ser igual al del proyecto anterior.

Quedan por tanto los diagramas que se listan a continuación, y se adjuntan posteriormente una captura del diagrama de ListarProductosPorProveedorCommand (Ilustración 20) y el diagrama de DevolverVentaPanel (Ilustraciones 21 y 22), a modo de ejemplo.

- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/AltaProductoCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/AnadirProveedorCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/BajaProductoCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/ListarBebidasCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/ListarComidasCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/ListarProductosCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/ListarProductoPorProveedorCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/ListarProductoPorVentaCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/ModificarProductoCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/MostrarProductoCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Producto/DiagramasDeSecuencia/QuitarProveedorCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Venta/DiagramasDeSecuencia/CerrarVentaCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Venta/DiagramasDeSecuencia/DevolverVentaCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Venta/DiagramasDeSecuencia/ListarVentasCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Venta/DiagramasDeSecuencia/ListarVentasPorClienteCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Venta/DiagramasDeSecuencia/ListarVentasPorEmpleadoCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Venta/DiagramasDeSecuencia/ListarVentasPorProductoCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Venta/DiagramasDeSecuencia/MostrarVentaCommand*
- *Modelos/Modelo de Diseño/Presentación/Command/Venta/DiagramasDeSecuencia/MostrarVentaEnDetalleCommand*
- *Modelos/Modelo de Diseño/Presentación/Controller/DiagramasDeSecuencia/ApplicationController*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/AltaProductoPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/AnadirProveedorPanel*

- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/BajaProductoPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/ListarBebidasPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/ListarComidasPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/ListarProductosPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/ListarProductosPorProveedorPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/ListarProductosPorVentaPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/ModificarProductoPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/MostrarProductoPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Producto/DiagramasDeSecuencia/QuitarProveedorPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Venta/DiagramasDeSecuencia/CrearVentaPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Venta/DiagramasDeSecuencia/DevolverVentaPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Venta/DiagramasDeSecuencia/ListarVentasPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Venta/DiagramasDeSecuencia/ListarVentasPorClientePanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Venta/DiagramasDeSecuencia/ListarVentasPorEmpleadoPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Venta/DiagramasDeSecuencia/ListarVentasPorProductoPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Venta/DiagramasDeSecuencia/MostrarVentaPanel*
- *Modelos/Modelo de Diseño/Presentación/Gui/Panels/Venta/DiagramasDeSecuencia/MostrarVentaEnDetallePanel*

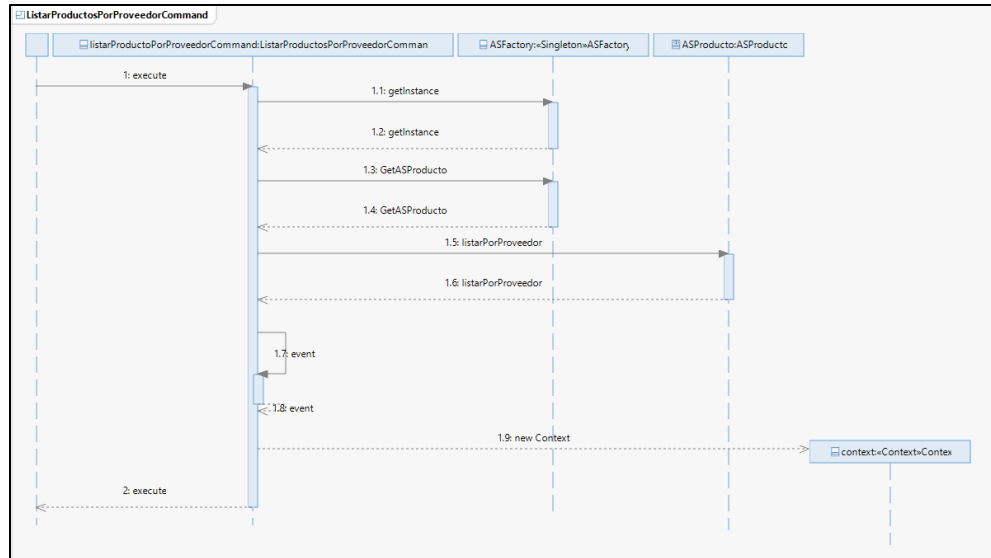


Ilustración 20. Diagrama secuencia ListarProductosPorProveedorCommand

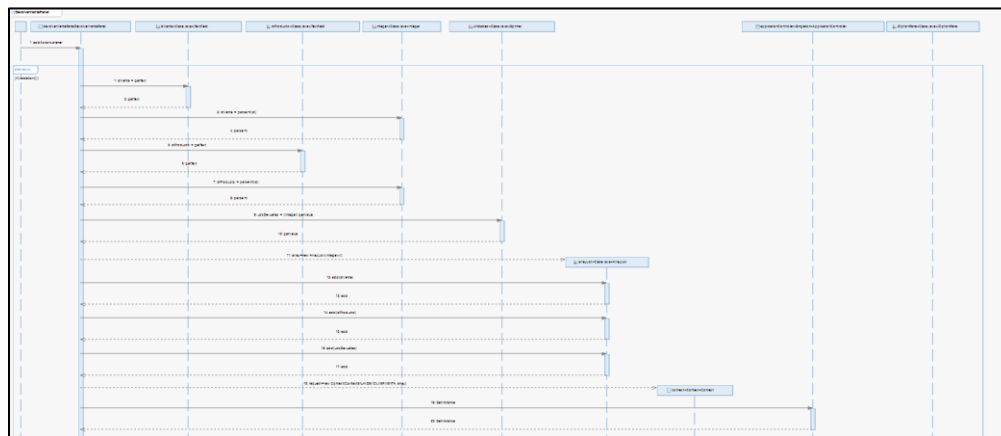


Ilustración 21. Diagrama secuencia DevolverVentaPanel (1)



Ilustración 22. Diagrama secuencia DevolverVentaPanel (2)

### 5.3. Diagrama de despliegue

#### DIAGRAMA DE DESPLIEGUE

El diagrama de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Se adjunta a continuación (Ilustración 23) el diagrama de despliegue correspondiente a *Plateau*, mostrando la arquitectura de ejecución de este proyecto.

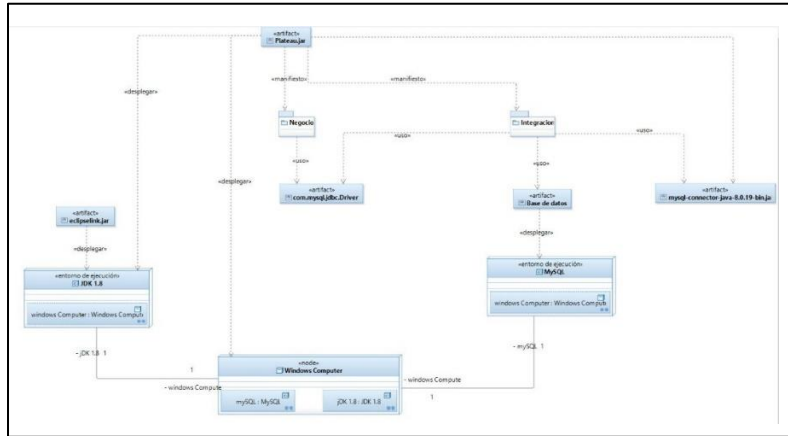


Ilustración 23. Diagrama de despliegue

#### DIAGRAMA DE COMPONENTES

El diagrama de componentes representa la división y organización del sistema en componentes software, sus interfaces y las dependencias entre ellos. Se adjunta a continuación (Ilustración 24) el diagrama de componentes correspondiente a *Plateau*.



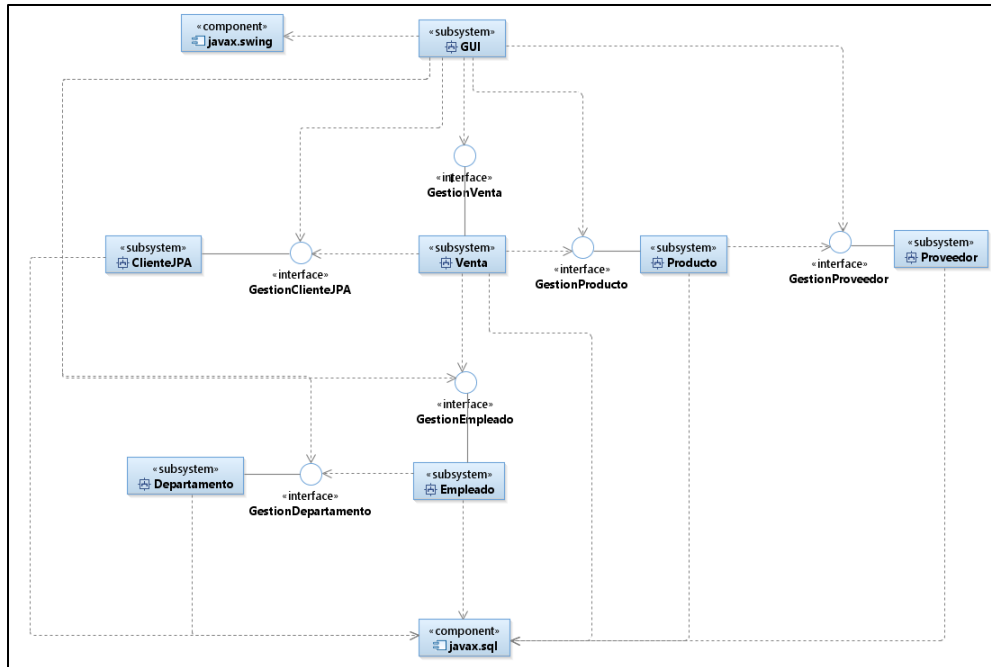


Ilustración 24. Diagrama de componentes

## MODELO DE DATOS

El modelo de datos representa la estructura de los datos (tipos) y la forma en la que se relacionan entre sí. Se adjunta a continuación (Ilustración 25) el modelo de datos correspondiente a *Plateau*.

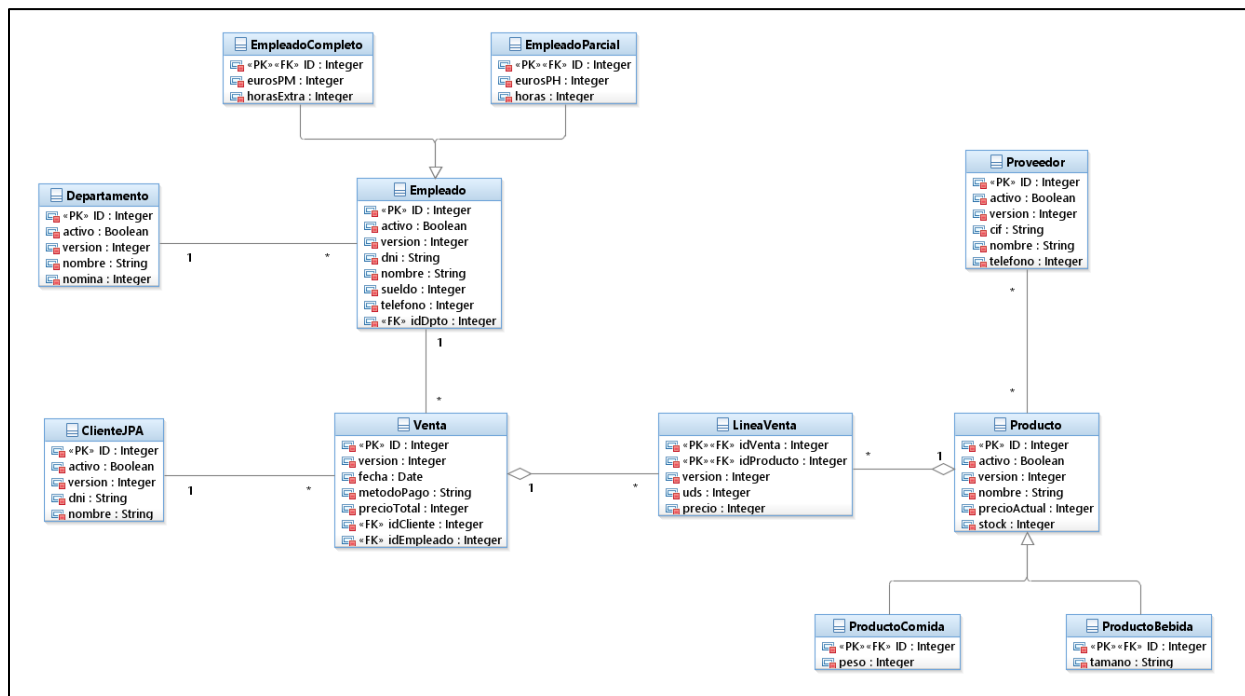


Ilustración 25. Modelo de datos

## 6. Estructura del código

En primer lugar, es relevante aclarar que el código fue generado a partir del modelo una vez terminado este, y gracias a la herramienta de transformación que proporciona el entorno de desarrollo.

Todo el código de *Plateau* ha sido desarrollado en el repositorio de «cod/trunk», utilizando las carpetas «src» y «test». Antes de explicar su contenido, existe también dentro de este repositorio una carpeta «images», donde se almacena el logo de la aplicación, y «lib», donde se encuentran los .jar necesarios para importar JPA y EclipseLink.

Dentro de la carpeta «src», existe una subcarpeta por capa (Integración, Negocio y Presentación) la carpeta «utilities» con una clase auxiliar necesitada en varios módulos, y la carpeta «META-INF», con la configuración de la BD y el archivo “persistence.xml” para la unidad de persistencia. Dentro de las subcarpetas de las capas existen subcarpetas iguales a las explicadas en el apartado del [Modelo](#), con las clases e interfaces necesarias. Por otro lado, se encuentra la carpeta «test» hermana de «src», donde se incluyen los tests unitarios implementados con JUnit.

Se adjunta a continuación una captura (Ilustración 26) de los distintos paquetes que existen para facilitar al lector la visualización de esta estructura.

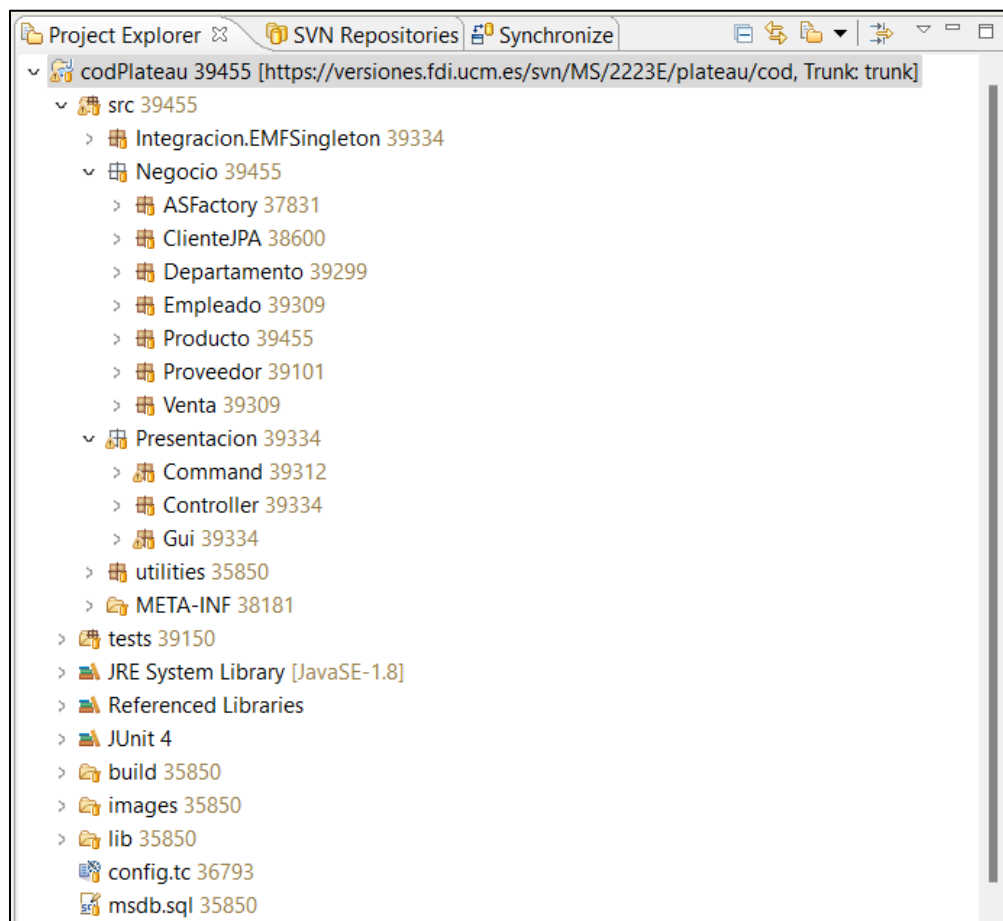


Ilustración 26. Estructura código

## 7. Pruebas

Para garantizar el funcionamiento e implementación de las reglas de negocio de manera correcta, se hicieron 2 tipos de pruebas: pruebas unitarias de los AS y comprobaciones manuales.

Las pruebas unitarias de los AS se enfocan en el funcionamiento correcto de la capa de negocio y la implementación de las reglas lógicas. Por cada módulo se creó un conjunto de pruebas, cada prueba enfocada en un método distinto del SA. Las pruebas unitarias SA aseguran que no se rompen las reglas de negocio, probando todos los casos posibles.

Además, desde que el código empezó a ser funcional, se han realizado continuas comprobaciones manuales desde la interfaz de usuario para testear los módulos en su totalidad. Estas pruebas han dado lugar a depuraciones exhaustivas pasando por todos métodos de todas las capas hasta localizar los errores que se producían.

Por último, una vez realizadas todas las pruebas mencionadas, se han realizado lo que se ha denominado pruebas de aceptación, en las cuales un integrante del equipo, actuando como cliente, ha testado la aplicación probando todos los posibles casos que se pueden dar para cada funcionalidad. Así, se ha ido comprobando cada caso de uso comparando con los requisitos establecidos en la SRS, apuntando qué casos fallan para arreglarlos posteriormente, y qué detalles de usabilidad podrían mejorarse. Toda esta información, junto a otras observaciones relevantes, se incluye a continuación:

- ✓ Mensaje incorrecto al dar de baja con éxito un cliente
- ✓ La tabla LineaVenta en la BD tiene repetidos los ID de la Venta y Producto
- ✓ La redirección de vistas no es correcta en los casos de mostrar y mostrar para modificar
- ✓ Los ArrayList de entidades para la relación Producto-Proveedor son muy ineficientes (se realizan búsquedas constantemente)
- ✓ En AltaProducto, cuando no seleccionas el tipo salta una excepción que no debería (se muestra un mensaje incorrecto)
- ✓ El mensaje de error sintáctico en AltaProducto está mal escrito
- ✓ Si intentas dar de alta un producto con el mismo nombre que uno activo no te salta error (aunque no lo inserta, es por los códigos de error)
- ✓ En el panel de ModificarProducto los campos correspondientes a atributos especializados están mal ubicados
- ✓ Al dar de baja un departamento debería reiniciarse la nómina a 0 (si no, cuando se reactive su nómina acumulará sueldos de empleados inactivos)
- ✓ En el panel de ModificarDepartamento el botón para modificar está mal ubicado
- ✓ En AltaEmpleado, el spinner del campo euros/mes no tiene valores adecuados
- ✓ En AltaEmpleado, en la reactivación la nómina de los departamentos no se actualiza correctamente
- ✓ En AltaEmpleado, si reactivas un empleado cambiándole el departamento al que pertenece no se actualiza en la BD (aunque sí la nómina del nuevo departamento)
- ✓ En el panel de MostrarEmpleado el campo euros/mes para los empleados completos está mal ubicado
- ✓ En ModificarEmpleado, el DNI no se actualiza en la BD
- ✓ En ModificarEmpleado no salta error si se introduce un DNI existente

- ✓ En ModificarEmpleado, si cambias el departamento al que pertenece no se actualiza en la BD (aunque sí las nóminas de los departamentos antiguo y nuevo)
- ✓ En CrearVenta, no se hace correctamente el clear()
- ✓ Se puede realizar una venta vacía
- ✓ El precio total al realizar una venta no se calcula bien (no se tienen en cuenta las unidades)
- ✓ En MostrarEnDetalle, falla si se introduce un ID de una venta que no existe
- ✓ En MostrarEnDetalle no se rellena el campo método pago
- ✓ En MostrarEnDetalle la fecha no se escribe con el formato correcto

Al ser un desarrollo continuo, las pruebas unitarias y manuales permitieron la comprobación del seguido cumplimiento de los requisitos funcionales tras hacer cualquier modificación y parcheo del código. Tras haber corregido los fallos anteriores, quedan las siguientes observaciones acerca del producto que resulta relevante conocer:

- ✓ En los JSpinners no se controlan valores que pueda meter el usuario manualmente y no estén entre los valores definidos para el spinner (por ejemplo, texto en lugar de números).
- ✓ En ListarPorProveedor, los productos no salen ordenados por ID.
- ✓ En Proveedor, no se controla el formato del CIF.
- ✓ Si se modifica el stock de un producto, este no puede estar incluido en ninguna venta.
- ✓ El sueldo de los empleados se calcula con la siguiente regla:
  - Si es a tiempo completo  $\rightarrow \text{Sueldo} = \text{euros/mes} + \text{horas extra} * 5 \text{ euros/h}$
  - Si es a tiempo parcial  $\rightarrow \text{Sueldo} = \text{horas/día} * \text{euros/hora} * 20 \text{ días/mes}$
- ✓ Se controla el formato del DNI, tanto para Empleado como para Cliente.
- ✓ Si se da de baja un cliente, no se actualiza el stock de los productos que se le hubieran podido vender. Se considera que sus compras no se devuelven.
- ✓ Al realizar una venta, no se considera error que se intenten quitar del carrito más unidades de un producto de las que se habían añadido. Se dejan a 0 y se elimina el producto del carrito.
- ✓ No se considera error que se intenten devolver más unidades de las que se habían vendido, se dejan a 0 (también en el caso particular de que ya se hubieran devuelto previamente todas las entradas).
- ✓ Al reactivar un departamento no se reactivan sus empleados (y, por tanto, su nómina queda a 0).
- ✓ No se controla que los DNIs de los clientes no coincidan con ningún DNI de los empleados

## 8. Recursos

El principal recurso utilizado para llevar a cabo el proyecto ha sido el entorno de desarrollo *IBM Rational Software Architect Designer*. Esta herramienta nos ha permitido tanto generar y almacenar código fuente, como elaborar los diagramas de clase, secuencia y despliegue. Se trata de un entorno de modelado y desarrollo que utiliza el lenguaje de modelado unificado (UML) para diseñar arquitectura para aplicaciones y servicios web C++ y Java EE (JEE). *Rational Software Architect* se basa en el marco de software de código abierto Eclipse e incluye capacidades centradas en el análisis de código arquitectónico, C++ y el desarrollo basado en modelos (MDD) con el UML para crear aplicaciones y servicios web.

Por otro lado, para la gestión de la base de datos se ha utilizado MySQL, y como entorno de trabajo el MySQL Workbench, uno de sus primeros productos. Este software incluye entre sus funcionalidades el modelado de datos, el desarrollo de SQL y otras herramientas de administración. Esto permite la depuración del código y la realización de todo tipo de pruebas software.

La aplicación ha sido desarrollada en Java, y el equipo de *Plateau* ha hecho uso del sistema de control de versiones de la Facultad de Informática (Universidad Complutense de Madrid) para gestionar tanto la documentación como el propio código, utilizando SVN para la administración del repositorio.

En materia de persistencia, el equipo de *Plateau* ha empleado Java Persistence API, más conocido como JPA, en su versión 2.2.3. JPA es un estándar que ofrece Java para implementar lo que comúnmente se conoce como *Object Relational Mapping (ORM)*, es decir, el mapeo de objetos Java a una base de datos relacional. El framework empleado para realizar la implementación de JPA ha sido EclipseLink en su versión 2.7.8.

Aparte de estas herramientas, se han utilizado plataformas como *Whatsapp* y *Discord* para la comunicación del equipo, *Microsoft Word Office 365* para la elaboración de la documentación, y la plataforma *One Drive* para almacenar las distintas versiones del Modelo del Dominio y de la SRS, y otros documentos borradores para facilitar el reparto de tareas y la organización del equipo.