I/O RECORDS

MEMORIA

David Arroyo Segovia

Ignacio Cepeda Bajo

Ángel Cruz Alonso

Hao Hao He

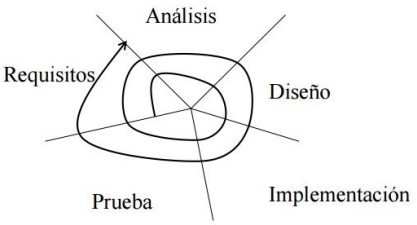
Carla Paola Peñarrieta Uribe

ÍNDICE

I/O RECORDS

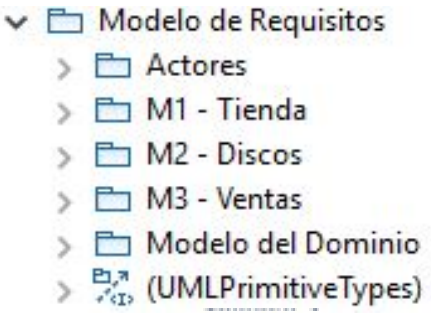
1. Estructura del proyecto

El proceso que se sigue en el proyecto es el proceso unificado de desarrollo, en el cual hemos seguido una estructura que está formada por cinco flujos de trabajo que se iteran: requisitos, análisis, diseño, implementación, pruebas.



De esta manera, el proyecto está organizado en tres paquetes:

En primer lugar, el *modelo de requisitos*, el cual consta de los siguientes pasos (no hace falta que se realicen por separado): enumerar los requisitos candidatos, comprender el contexto del sistema, capturar los requisitos funcionales, capturar los requisitos no funcionales. Se divide en tres paquetes.



-Modelo del dominio: el objetivo de este modelo es ayudar a comprender y describir las clases más importantes dentro del contexto del sistema que se trata, por ello se proporciona un vocabulario común a clientes y desarrolladores facilitando la comprensión del conjunto.

-Un paquete en el que aparecen los actores de todos los casos de uso, en nuestro caso, los actores son empleado, cliente y usuario.

-Un paquete por cada módulo:

- **M1-Tienda**
- **M2-Discos**
- **M3-Ventas**

Cada uno de estos paquetes está dividido en requisitos funcionales que contienen los casos de uso, los cuales, a su vez, incluyen los diagramas de actividad. Tanto los diagramas de casos de uso (ayudan a capturar los requisitos funcionales y hacer hincapié en cada usuario individual) y de actividad (describen el caso de uso correspondiente) hacen de guía para el desarrollo del proyecto. Los requisitos funcionales (según el módulo) presentes en nuestro proyecto son:

➔ **M1**

- **Añadir producto**
- **Darse de alta**
- **Darse de baja**
- **Devolver producto**
- **Modificar producto**
- **Quitar producto**
- **Adquirir producto**
- **Ver catálogo**
- **Ver datos personales**
- **Ver estadísticas.**

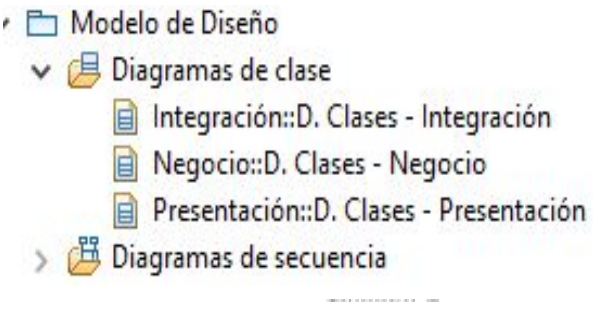
➔ **M2:**

- **Descatalogar discos**
- **Establecer ofertas**
- **Modificar discos**
- **Ver catálogo**
- **Ver disco.**

➔ **M3:**

- **Añadir pedido**
- **Eliminar pedido**
- **Generar factura**
- **Modificar pedido**
- **Ver beneficio.**

En segundo lugar, hemos realizado el modelo de diseño, el cual describe la realización física de los casos de uso centrándose en los requisitos del sistema obtenidos anteriormente.

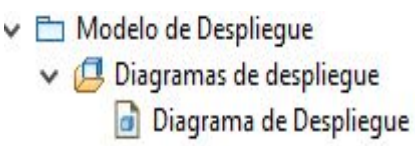


Lo hemos estructurado en dos paquetes, diagramas de clases y de secuencia.

Además, seguimos una arquitectura multicapa, las cuales son:

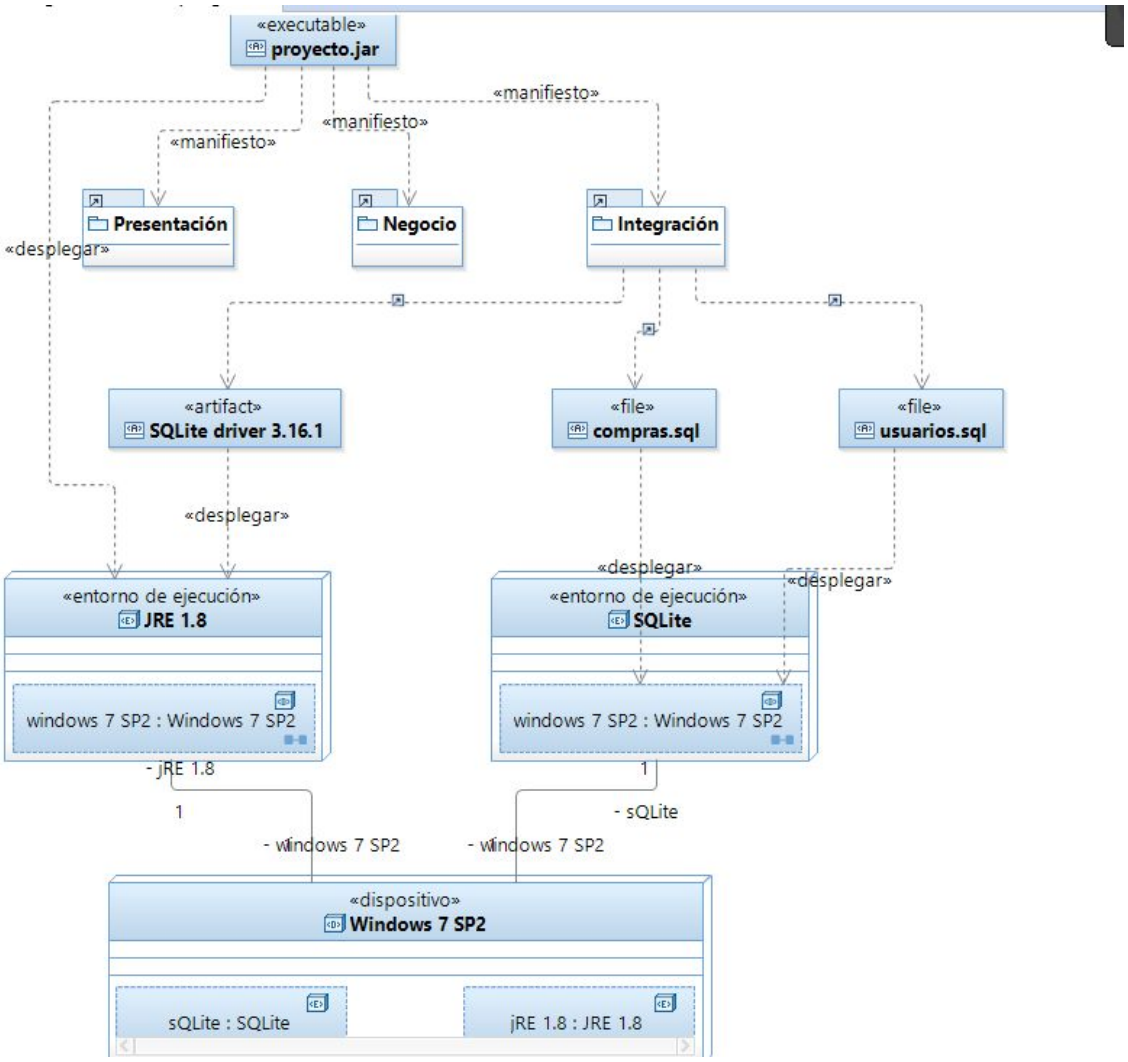
- **Presentación: hace referencia a la interfaz gráfica o textual. Se le atribuye la lógica para que los clientes puedan acceder al sistema.**
- **Negocio: Realización de acciones a base de la información recopilada desde la capa de presentación y desde la capa de integración. Esta capa también proporciona los servicios del sistema.**
- **Integración: hace referencia a los recursos y sistemas externos que guarda el programa.**

Por último, el modelo de despliegue que es un



modelo de objetos que sigue una distribución física de cómo funciona la conexión entre los nodos. Cada nodo representa un recurso del sistema que normalmente es un procesador o dispositivo hardware.

Dicho modelo está formado por un *diagrama de despliegue*



así como los entornos de ejecución (hardware utilizado) y artefactos. Estos, representan elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. Ejemplos de artefactos son archivos ejecutables, bibliotecas, archivos, esquemas de bases de datos, archivos de configuración, etc.

De esta manera, existe una correspondencia entre la arquitectura software y la arquitectura del sistema (el hardware).

2. Análisis, diseño e implementación:

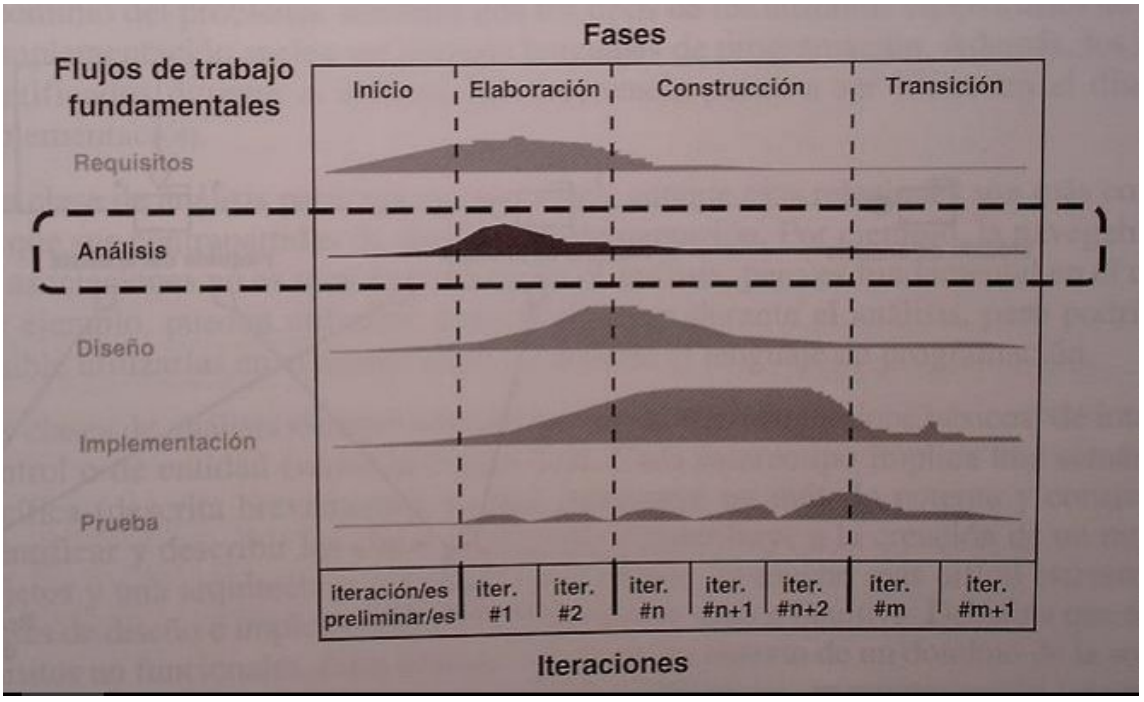
2.1 Análisis

En la etapa de análisis nos hemos encargado de capturar los requisitos y el dominio del

problema, centrándonos en lo que hay que hacer, es decir, detallando los requisitos solicitados. Para ello, descompusimos las tareas en la realización de los diagramas de casos de uso, de actividades y los diagramas de secuencia para cada módulo.

En primer lugar, implementamos todos los casos de uso de cada módulo a partir de los requisitos sacados de la SRS. Estos describen de forma reactiva el el comportamiento del sistema, definiendo los límites del mismo y sus relaciones con el entorno.

Ciñéndonos a la teoría aprendida, tal y como se puede observar en la imagen siguiente, en nuestro proyecto el análisis ha cobrado mayor importancia en la fase de elaboración y principios de la construcción, la cual da pie a las etapas de diseño e implementación, que hemos realizado de forma posterior.



A continuación, se nombrará brevemente las finalidades de nuestra aplicación:

1. **Facilitar el trabajo, tanto al empleado como al cliente.**
2. **Mejorar la interacción, consiguiendo que disminuya la tasa de errores que pueda cometer una persona, poniendo en función una aplicación.**
3. **Realizar multitareas para mejorar el rendimiento de la tienda.**
4. **Una apariencia moderna y elegante.**

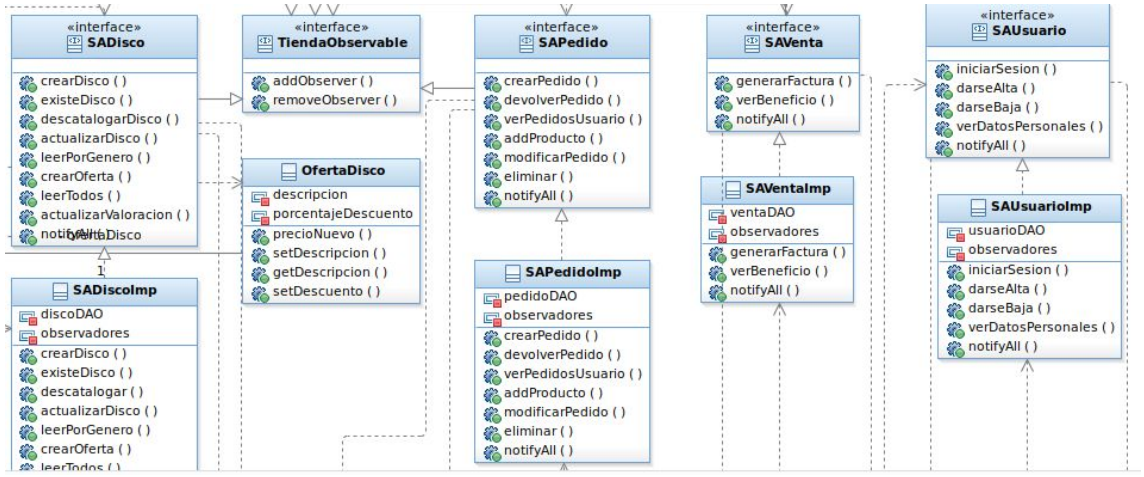
Hemos realizado por tanto un trabajo de análisis buscando la adecuación a los contenidos y métodos exigidos en la asignatura, tratando en todo momento de conseguir a la vez que nuestra aplicación tenga las características anteriormente enumeradas.

2.2 Diseño

Tal y como hemos mencionado en la estructura del proyecto, hemos realizado un sistema de arquitectura multicapa, compuesto por:

- **Negocio**
- **Presentación**
- **Integración**

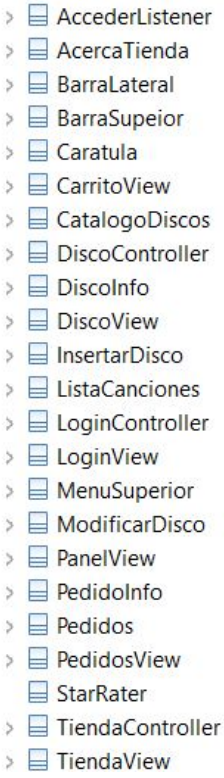
Capa de negocio: En ella hemos implementado los servicios de aplicación, encargados de encapsular la lógica de negocio de cada módulo, los cuales tienen asignados servicios de aplicación de forma individual .



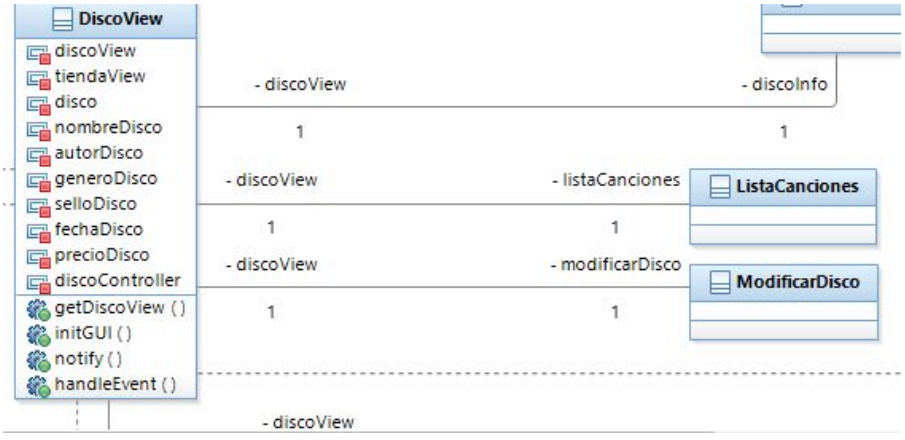
Capa de integración: En ella se encuentran los DAOs, los cuales se conectan con el sistema de persistencia, que en nuestro caso se trata de una base de datos SQLite. Al igual que con los servicios de aplicación en la capa de negocio, existe una DAO para cada módulo.

Capa de presentación: Encargada de los componentes relacionados con la vista gráfica

de la aplicación, la hemos realizado de la forma más modular posible, utilizando clases separadas para cada componente.



Todas estas características se pueden observar en el modelo de diseño. Como ejemplo concreto en nuestro proyecto:



Se trata de una captura de una parte de la capa de presentación incluida en el paquete de diagrama de clases. Se puede observar que existe una asociación bidireccional entre las clases DiscoView con *ListaCanciones* y *ModificarDisco*. Además, se identifican claramente los atributos y métodos de las clases representativas.

Gracias al uso de interfaces podemos hacer separaciones que permitan clarificar la interpretación gráfica de los diagramas descriptivos del proyecto.

Dentro del diseño también tuvimos que realizar el modelo de despliegue, consistente en una relación de objetos que describe la distribución física del sistema en términos de las interrelaciones de funcionalidad entre los nodos de cómputo que lo componen.

2.3 Implementación

Comenzamos esta fase una vez damos por concluida la etapa de diseño y hemos implementado el sistema en componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

Nuestros propósitos básicos en esta fase fueron:

- Definir la organización del código.

- Implementar clases y objetos en forma de componentes (fuente, ejecutables, etc.) encontrados durante la etapa de diseño.

- Probar las componentes desarrolladas individualmente .

- Integrar las componentes en un sistema ejecutable.

- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue.

El diagrama de despliegue de nuestro proyecto, en el que se representa gráficamente dicha implementación, se encuentra en la introducción de la memoria.

Las interfaces del modelo de implementación se corresponden con las del modelo de diseño.

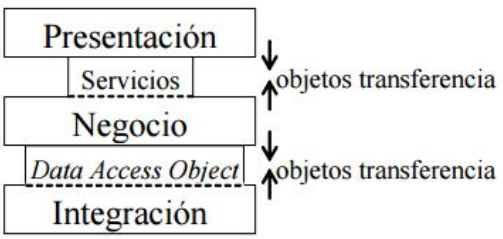
La última etapa del flujo que hemos realizado es la prueba, donde se verifica el resultado de la implementación probando cada construcción, así como las versiones finales del sistema.

3. Arquitectura

Este proyecto está basado en una arquitectura multicapa, es decir, un conjunto ordenado de subsistemas, cada uno de los cuales están constituidos en términos de los que tiene por debajo y proporciona la base de la implementación de aquellos que están por encima de él.

La arquitectura multicapa dispone de tres tipos de nodos:

- **Clientes que interactúan con los usuarios finales.**
- **Servidores de aplicación que procesan los datos para los clientes.**
- **Servidores de la base de datos que**



Arquitectura multicapa

almacenan la información para los servidores de aplicación.

Patrones utilizados

- **Singleton: este patrón es utilizado en la creación de las factorías abstracta, controlador y en distintas clases de la capa de presentación.**

- **Transferencia: este patrón lo utilizamos en las clases básicas que forman la aplicación, como pueden ser Usuario y Disco, quedando reflejado en los diagramas observando si el nombre de la clase es la palabra simple, sin prefijos ni sufijos añadidos.**

- **Data Access Object (DAO): este patrón es utilizado para las clase principales para generar una conexión con la DB (utilización en la capa de integración), además en estas se implementarán patrones como singleton y factoría abstracta.**

- **Servicio de aplicación (SA): este patrón, al igual que el anterior, se usa en las clases principales, y está en la capa de negocio, donde genera una encapsulación, además en estas se implementarán patrones como singleton y factoría abstracta.**

- **Modelo Vista-Controlador (MVC) y Observador:**
    - **Los componentes que forman la vista son, por ejemplo, PanelView (es un panel en el cual solo el empleado del negocio puede acceder a este),** *BarraLateral* **(es un panel donde viene la información del disco, y donde se puede generar un filtro de discos),** *CarritoView* **(es un panel donde se muestra los discos que se han añadido a este, para ser procesados en un nuevo pedido),** *CatalogoDiscos* **(es un panel donde se muestran los distintos discos que se encuentran en la DB), DiscoView (es un panel donde se mostrará la información del disco), LoginView (es un panel donde se iniciará sesión con una cuenta que este establecida en la DB)...**

    - **El patrón se ha implementado utilizando el patrón observador-observable para comunicar el modelo con la vista. El modelo manda un objeto Notificación a la vistas que encapsula toda la información que necesita representar.**

    - **Quienes desempeñan el papel de modelo son** *DAODisco*, *SADisco*,

DAOUsuario, *SAUsuario*... **estos son implementados a través de los patrones DAO y SA, que a su vez usa los patrones de factoría abstracta y singleton. Se encargan de encapsular y sacar la información necesaria de la DB.**

- **Factoría abstracta: este patrón es utilizado en las DAO y SA. Genera una interfaz de familia de objetos.**