MEMORIA-TIENDA DE VIDEOJUEGOS WEILITTLE

MEMORIA - MS

ÁLVAREZ MEDINA, SAMUEL
BARBAS VÁZQUEZ, SERGIO
BONILLA TACO, ALEX GUILLERMO
CALVO FERNÁNDEZ, JORGE
MARTIN SOTO, AIRAM
MORENO GRACIA, ALBERTO
OCHOA DE ZABALEGUI VELASCO, SANTIAGO
RODRÍGUEZ MARGALLÓ, ADRIÁN
SÁNCHEZ CARRASCO, SERGIO
TARANCÓN MARTÍNEZ, SERGIO
VILLA ABAJO, MANUEL
ZHU, WEIHONG

20/04/2025

ING. SOFTWARE

ÍNDICE

1. Introducción

Tras el empleo del modelo de proceso unificado desarrollo de software que se inició en IS1, IS2, el desarrollo que se le ha dado al mismo en el proyecto de MS durante la primera y segunda entrega. Podemos decir que, se ha conseguido implementar los mecanismos y conocimientos adquiridos en la asignatura de Modelado de Software sobre el manejo de la transacción y concurrencia a través de JPA, creando una aplicación final estable, con una interfaz sencilla e intuitiva, que gestiona una tienda de videojuegos y una sala de arcade.
Se han seguido los pasos marcados por Antonio desde la especificación de la SRS, uso de la arquitectura multicapa, uso de los patrones obligatorios, gestión de la transaccionalidad pesimista para la parte DAO implementada para esta **Entrega Extraordinaria** y optimista para la parte JPA realizada en su día, en su correspondiente entrega y uso de GIT-HUB como Sistema de Gestión de Versiones.

2. Arquitectura

Usamos una arquitectura multicapa, la cual considera tres capas (integración, negocio y presentación) y que contendrán subsistemas de diseño, es decir, en cada capa se duplican los módulos.
Para la comunicación entre capas, hemos usado el patrón de transferencia, que se encargan de encapsular la lógica de acceso a datos e independizan el intercambio de datos entre ellas.
En la capa de presentación de JPA y DAO, hemos implementado la interfaz gráfica para el usuario a través de las vistas implementadas con Java Swing , que permiten al usuario moverse entre ellas para realizar ciertas acciones. Nuestro Controller será el encargado de recoger los contextos y solicitar a la factoría de comandos su correspondiente comando para luego ejecutarlo y redireccionar a vista el contexto de salida. Consiguiendo que nuestro proyecto tenga una alta cohesión y un bajo acoplamiento

<u>Gestión Tienda de Videojuegos(DAO):</u>
En integración, hemos implementado los Data Access Object (DAO) y Domain Store, que se encargará de interactuar con la capa de recursos, en nuestro caso una base de datos SQL, alojada en un servidor personal, desplegada en mySQL.
La capa de negocio la hemos usado como puente entre la capa de presentación y la base de datos. En ella implementamos el Servicio de Aplicación para coordinar y llevar a cabo las diversas operaciones de negocio necesarias para cumplir con las solicitudes de la capa de presentación.
<u>Gestión del Arcade(JPA):</u>
En integración, hemos implementado el Singleton EMFSingleton, que se encarga de crear la unidad de persistencia y desde la capa de negocio, el servicio de aplicación correspondiente se hace su correcto uso junto con los objetos de negocio(Entitys)

3. Patrones de diseño

- Patrón Service to worker: usado en nuestra arquitectura multicapa, que articula en él un Controller haciendo que sean independientes la interfaz del usuario y la lógica de nuestro modelo.
- Patrón Context: usado para encapsular las peticiones.
- Patrón Commad y FactoryCommand: usado para parcear las peticiones segun el comando y ejecutar su correspondiente servicio de aplicación.
- Patrón Transferencia: usados entre capas
- Patrón TOA: usado en venta para listar una venta completa
- Patrón Data Access Object (DAO): usados para la lectura y escritura de los datos con la capa de recursos (base de datos en MySQL) y son llamados por el servicio de aplicación correspondiente.
- Patrón Servicio de Aplicación: usados para encapsular la lógica del negocio de cada módulo creado y encapsula funcionalidades relacionadas.
- Patrones Auxiliares
    - Singleton: usadas en el controlador y las Factorías de Presentación, Negocio e Integración, garantizando que solo hay una instancia para cada clase.
    - Factoría abstracta: usadas para crear factorías del Servicio de Aplicación, Integración, Presentación, creando familias de objetos relacionados, independizando su creación y encapsulando sus implementaciones en interfaces.
    - Observador: no tenemos estrictamente uno de ellos puesto que nuestros objetos no se notifican y actualizan automáticamente pero nuestro Controller es una especie de ello que se encarga de mediar entre la vista y el modelo y actualizar una vista a través del método update().
- Patrón Business Object: usados en la capa de negocio que se encargan de la lectura y escritura de los datos con la capa de recursos.
- Domain Store: usado para gestionar concurrencia y transaccionalidad de forma pesimista en la parte DAO y de forma optimista en la parte JPA, implementado con JPA 2.5 a través de EclipseLink.

4. Diagramas Parte DAO

- Diagramas de clase:
    - Integración: todos
    - Negocio: todos
    - Presentación: todos
- Diagramas de secuencia:
    - Integración:
        - Plataforma: todos
        - Trabajador: vincular habilidad, desvincular habilidad, listar trabajadores por habilidad, listar trabajador por tipo
        - Venta: cerrar venta
        - Producto: alta producto
        - Transaccion: create, start, commit, rollback, getTransaction, transactionMySQL
    - Negocio:

        - Plataforma: todos
        - Trabajador: vincular habilidad, desvincular habilidad, leer trabajadores por habilidad, listar trabajador por tipo.
        - Venta: cerrar venta
    - Presentación:
        - Plataforma: todos
        - Trabajador: vincular habilidad, desvincular habilidad, listar trabajadores por habilidad, listar trabajador por tipo
        - Venta: cerrar venta
        - Command: commandAltaPlataforma
        - Controller
        - CommandFactory
- Diagrama de despliegue: en el paquete de despliegue
- Estructura de BBDD Videojuegos: en el paquete de despliegue
- Diagrama de componentes: en el paquete de despliegue

5. Diagramas Parte JPA

- Diagramas de clase:
    - Integración: todos
    - Negocio: todos
    - Presentación: todos
- Diagramas de secuencia:
    - Negocio:
        - ClienteJPA: alta, baja, listar todo, modificar.
        - MaquinaJPA: alta.
        - ModeloJPA: calcular coste seguro, vincular, desvincular
        - ProveedorJPA: modificar
        - AlquilerJPA: abrir, cerrar, buscar, cancelar
        - EmpleadoJPA: alta
    - Presentación:
        - ClienteJPA: alta, baja, listar todo, modificar.
        - ProveedorJPA: modificar
        - AlquilerJPA: abrir, cerrar, buscar, cancelar
        - CommandAlquilerJPA:abrir, cerrar

6. Repositorios

- Código

https://github.com/anavarro-fdi-ucm-es/ms2425videojuegoscod.git
Rama: ENTREGA-EXTRAORDINARIA-PARTE-DAO

- Modelo y Documentación:

https://github.com/anavarro-fdi-ucm-es/ms2425videojuegosmod.git
Rama: ENTREGA-EXTRAORDINARIA-PARTE-DAO
Tiene dos carpetas:

- Documentación: contiene la SRS, MEMORIA e INFORME DE TRABAJO
- TiendaVideojuegos: contiene el modelo