PROYECTO VIDEOCLUB VIRTUAL

Programación II

Asier Marin

31/05/19

Índice

Índice de figuras

1. Contexto

El programa principalmente muestra un comportamiento de tratamiento de datos en una base de datos. Es posible introducir, obtener, analizar y eliminar datos de forma automatizada. Se trata de un videoclub virtual donde el usuario puede alquilar diferentes artículos en tiempos determinados o tener un suscripción mensual.

Los objetivos de la aplicación son los siguientes:

- Añadir datos de artículos diferentes en una base de datos.
- Proporcionar capacidad para crear usuarios diferentes e introducirlos en la base de datos.
- Cargar todos los datos, incluyendo usuarios y artículos así como sus alquileres en la base de datos.
- Obligar al usuario a validarse dentro de la aplicación.
- Métodos para eliminar alquileres o baja suscripción una vez se cargan los datos de forma automatizada.
- Método para validar que no existan usuarios con nombre de usuario y contraseña iguales.
- Métodos de ordenación de datos de artículos.
- Ect…

La aplicación debe ser capaz de comportarse como un videoclub y una plataforma de suscripción multimedia.

2. Estructura

La estructura del proyecto se compone a nivel de programación de 5 carpetas principales, siendo la lógica de presentación, la lógica de negocia, la lógica de datos, carpeta de excepciones y carpeta común.

Figura 2.1. Estructura principal

2.1. COMÚN

La carpeta común contiene las imágenes utilizadas en las interfaces gráficas, la clase clsConstantes (como su nombre indica con constantes públicas) y la interface itfProperty muy útil para obtener los datos que se guarda en memoria y llevarlo a la lógica de presentación.

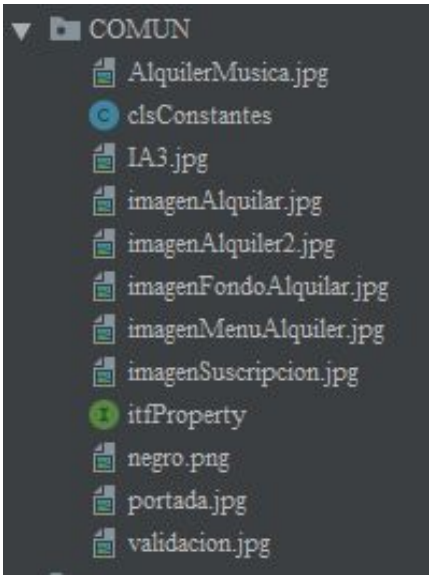

Figura 2.2. Carpeta común

2.2. LP

La lógica de presentación contiene todas aquellas clases que extienden de JFrame con constructores para crear las ventanas de la aplicación.

Figura 2.3. Carpeta lógica de presentación y clases

El punto de partida comience en el Main seguido por la clase clsMenuOpciones que comienza la secuencia o sucesión de ventanas.

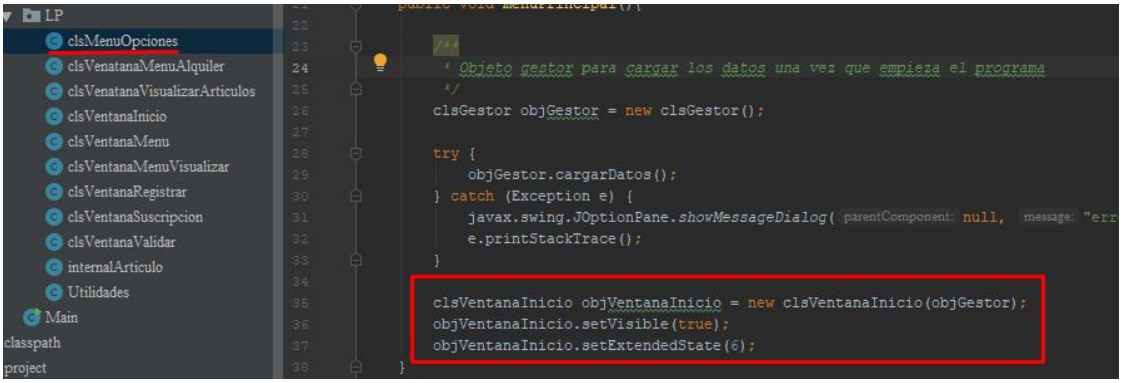

Figura 2.4. Comienzo de ventanas

2.3. LN

En la lógica de negocio ya se encuentran una profundidad de código mayor con más clases en su interior.

Figura 2.5. Clases de la lógica de presentación

La clases que alberga la lógica de presentación se podrían dividir en 4 grupos, siendo la clase más relevante de la gestión entre la lógica de presentación y la lógica de datos en clsGestor. También se encuentran las clases de los propios artículos y las clases de alquiler. Por último destacar la clase clsUsuario que también contiene los atributos propios de usuarios y sus métodos.

Clases para usuario	clsUsuario			
Clases para articulos	clsArticulo	clsPeliculas	clsVideojuegos	clsMusica_CD
Clase de gestión	clsGestor			
Clases para alquiler	clsAlquiler	clsAlquilerPeliculas	clsAlquilerVideojuegos	clsAlquilerMusica

Tabla 2.1.

2.4. LD

La lógica de datos contiene las clases que interactúa con la base de datos.

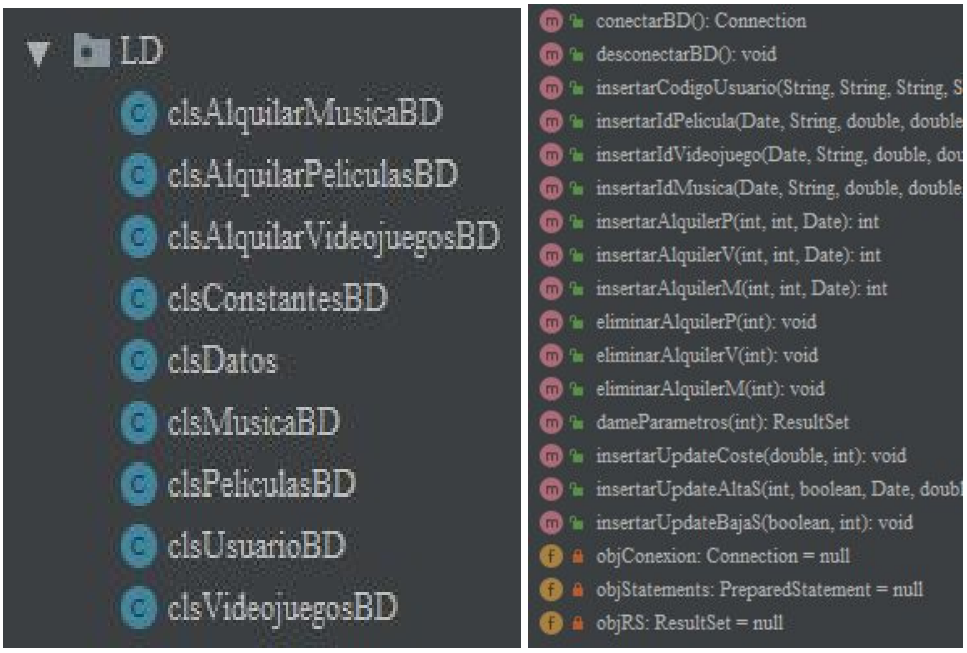

Figura 2.6. Clases lógica de datos y métodos clsDatos

Para la interconexión de métodos se encuentra la clase clsDatos. Contiene los métodos que transportan los métodos y los parámetros a la las clases propias las consultas, insert, update y delete con la base de datos.

3. Detalles técnicos

Existen dos tipos de herencia en la aplicación, siendo herencia de artículos con películas, música y videojuegos. La segunda herencia se encuentra con alquiler respecto al alquiler de los diferentes artículos.

El primer método para destacar es el uso de un hashCode() y equals para verificar dos atributos siendo identificador y contraseña para verificar la validación de usuario.

```java
@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((identificador == null) ? 0 : identificador.hashCode());
    result = prime * result
            + ((contrasena == null) ? 0 : contrasena.hashCode());
    return result;
}

@Override
public boolean equals(Object obj)    {
    if (this == obj) return true;

    if (obj == null) return false;

    if (getClass() != obj.getClass())
        return false;

    clsUsuario other = (clsUsuario) obj;
    if (identificador == null) {
        if (other.identificador != null)
            return false;
    } else if (!identificador.equals(other.identificador))
        return false;

    if (contrasena == null) {
        if (other.contrasena != null)
            return false;
    } else if (!contrasena.equals(other.contrasena))
        return false;
```

Figura 2.7. Método hasCode y equals

Se destaca el uso de dos métodos de ordenación Comparator para ordenar los objetos películas y videojuegos dependiendo del atributo “puntos” de forma descendente. Tambíen con el atributo “año” para los objetos de tipo musica_CD.

```java
/**
 * Clase que implementa Comparator para ordenar los datos de pelicula
 * descendente
 */
public class clsPeliculasPtos implements Comparator<clsPeliculas> {
    @Override
    public int compare(clsPeliculas peli_1, clsPeliculas peli_2) {
        return peli_2.getPuntuacion() - peli_1.getPuntuacion();
    }

}
```

Figura 2.8.Ordenación Comparator para “puntos” de peliculas

Destacar también el comparador compareTo para comprar fechas actuales con fechas pasadas para eliminar aquellos alquileres que a los que su fecha ha expirado. También se ha utilizado para comprobar la fecha de suscripción, una vez que se inicia la aplicación se compara la fecha para que dar de baja de suscripción sí ha expirado.

```java
public void comprobarSuscripcion() {

    try {
        objDatos.conectarBD();
        Date fechaHoy = new Date();
        for (clsUsuario usuario : listaUsuarios) {
            if (usuario.getFechaSuscripcion().compareTo(fechaHoy) < 0) {
                usuario.setSuscripcion(false);
                objDatos.insertarUpdateBajaS(usuario.isSuscripcion(), usuario.getCodigoAleatoria());

            }
        }
        objDatos.desconectarBD();
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    }
}
```

Figura 2.9. Método compareTo

La capa de presentación utiliza ventanas JFrame para visualizar los artículos. Para mostrar alquileres, por tema de tiempo también se decidió utilizar la misma técnica.

Figura 2.10. Mostrar en pantalla artículos disponibles

Para el diseño de la base de datos se optó por el uso de tres tablas para artículos y otras tres tablas para alquileres. También como no, la tabla usuarios que está conectada con todas.

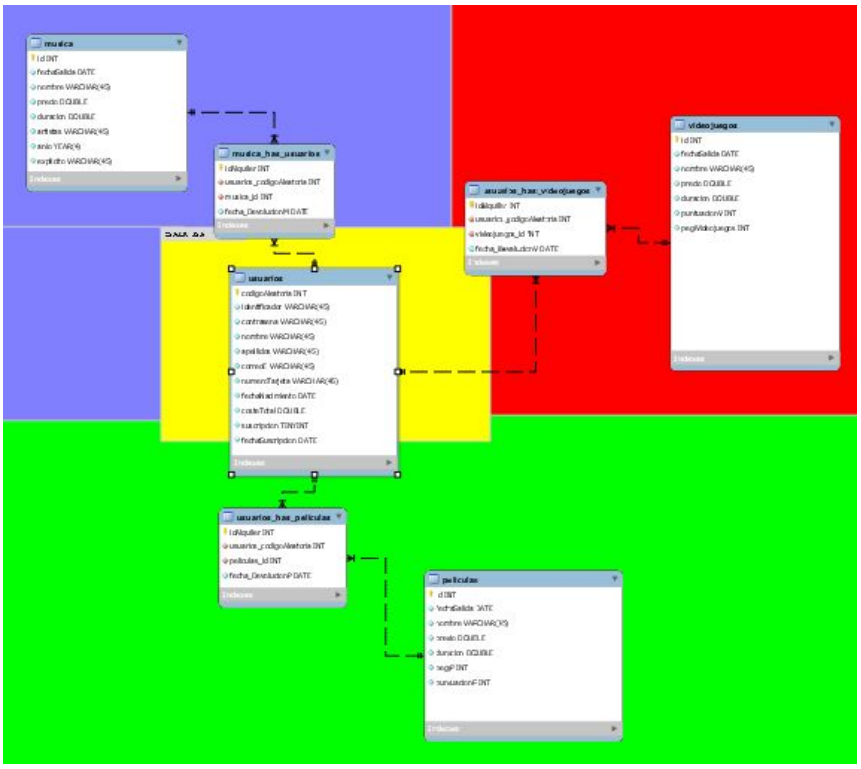

Figura 2.11. Esquema base de datos

Las tablas intermedias contiene los atributos “primary key” relevantes de las diferentes tablas entre artículos y usuario, muy importante en este caso para alquilar.

Para que el programa tenga un sentido viable tiene que tener la capacidad de tener bidireccionalidad de datos. Para ello se ha utilizado la interface itfProperty, pasando los datos a la lógica de presentación.

Destacar el uso de constantes para no repetir de forma necesaria texto o no tener en las clases de la lógica de negocio y la lógica de datos.

4. Planificación

La planificación del proyecto ha dejado mucho que desear. Los atrasos por el desarrollo de funcionalidades han sido constantes.

Se puede decir que la capa de presentación sí que ha tenido un ritmo acelerado respecto al desarrollo de funcionalidad. Existen objetivos que no se han podido cumplir por la inexistencia de tiempo necesario para ello.

		Descripción del proyecto a realizar y estudio y definición de las funciones que va a llevar a cabo la aplicación. Se debe contemplar la información que va a ser guardada en la aplicación a nivel de entidad, así como especificar el flujo de ejecución de la aplicación en caso de ser necesario alguno específico.	4	5
Diseño	T01	**Diseño y desarrollo de clases del proyecto (LN + LP)**		
		Diseño de las clases fundamentales para los datos del proyecto, su jerarquía de herencia y los atributos y métodos fundamentales. La clases deberán de ser programadas al menos con sus atributos básicos y con todas las descripciones javadoc pertinentes, estando distribuidas de forma correcta en paquetes adecuados.	5	7
Desarrollo	T02	**Diseño y Desarrollo de interfaces en las clases básicas**		
		Implementación de las interfaces necesarias para el intercambio entre LP-LN (itfProperty) y para el intercambio entre LN-LD (itfDatos), además de la clase clsDatos para el acceso a ficheros binarios. Entrega mínima de un "camino de funcionalidad" .	5	12
Diseño	T03	**Diseño de excepciones y gestión de errores en la aplicación y ordenación**		
		Diseño de las excepciones de tipo Exception que va a lanzar la aplicación, así como las de tipo runtimeException asociadas a itfProperty. Además, creación de las interfaces y clases necesarias para ordenación.		
Desarrollo	T04	**Desarrollo de excepciones y gestión de errores y ordenación**	15	24
		Identificación de las situaciones de error, codificación y atrapado de excepciones y comprobación de robustez de las clases de datos y codificación de funcionalidades de ordenación.		
Diseño	T05	**Diseño de ventanas e interfaz de usuario**	25	40
		Diseño del interfaz gráfico del proyecto y descripción de la funcionalidad para el usuario.		

Figura 2.12.Horas previstas e invertidas finalmente

El proyecto ha exigido una inversión de tiempo cada vez mayor. Se puede decir que se ha hecho un sacrificio de funcionalidad para poder tener una capa de presentación completa y trabajada. Aquí se destacan algunos puntos que con más tiempo se hubieran desarrollado:

- Introducir música de fondo desde el comienzo de la aplicación.
- Visualizar películas de manera real una vez se selecciona con el uso de una aplicación externa denominada Ace Player utilizando direcciones torrent.
- Mandar un email de validación de alta usuario una vez se ha hecho un registro.
- Capacidad para abarcar “series” en el videoclub.

5. Javadoc

La documentación javadoc está anexionada a este documento y se encuentra en una carpeta.