Lando

Programación II

31/05/2019

Aritz Garitano

Rubén Domínguez

# Índice

Contenido

[Índice 1](#_Toc10153737)

[Contexto y motivación 2](#_Toc10153738)

[Estructura de aplicación desarrollada 2](#_Toc10153739)

[Paquetes 2](#_Toc10153740)

[Común 2](#_Toc10153741)

[Excepciones 2](#_Toc10153742)

[LD 2](#_Toc10153743)

[LN 3](#_Toc10153744)

[LP 3](#_Toc10153745)

[Desarrollos de terceros 3](#_Toc10153746)

[Detalles Técnicos 4](#_Toc10153747)

[Planificación 5](#_Toc10153748)

[Desvíos 5](#_Toc10153749)

[Objetivos fijados 5](#_Toc10153750)

[Desarrollos futuros y opciones de ampliación 5](#_Toc10153751)

[Bibliografía 6](#_Toc10153752)

# Contexto y motivación

La idea de nuestra aplicación se nos ocurrió debido a que los días anteriores al comienzo del proyecto de estuvimos conversando acerca de comics, CDs, DVDs, libros, etc que teníamos por casa. En particular de los problemas de uno de los integrantes de tener en varios sitios diferentes objetos y luego echarlos en falta.

Ya había aplicaciones para organizar el almacenaje de libros y de películas, pero por separado. Entonces, nosotros pensamos en una que aplicación que integrara en una misma plataforma la gestión de los diferentes artículos. Por ello originalmente nuestra base de datos estaba formada por libros, canciones, películas, videojuegos y objetos de colección. Pero al crearla nos dimos cuenta de que quizá estábamos abarcando demasiado, por lo que la recortamos a libros, canciones y películas.

Nuestros principal requisito para el proyecto era que se pudieran agrupar canciones, películas y libros en “Librerías Multimedia”. Estos grupos podrían representar, por ejemplo, una colección en particular, ej: Marvel (Nombre), colección de comics y películas de Marvel que está en casa de los abuelos (Descripción). De esta manera el usuario identifica el grupo de artículos con un nombre a su elección, y tiene la opción de añadirle una descripción para posibles aclaraciones.

# Estructura de aplicación desarrollada

## Paquetes

### Común

Consta de una clase formada por las constantes de inserción y consulta, y otra clase que contiene la interfaz itfpropertyV2 (Mejora de la interfaz itfProperty presente en otro paquete) usada para mantener la independencia entre los paquetes LN y LP, y que devuelve la propiedad solicitada mediante un String, como un objeto.

### Excepciones

Alberga las excepciones de la aplicación, en este caso 4. Una para los campos con contenidos vacíos, otra para cuando programamos mal la petición de un case, una tercera para las repeticiones de títulos y una ultima para en caso de pedir una propiedad que no existe.

### LD

LD contiene todas las clases que tengan como propósito interactuar con la base de datos; estos son:

1.-El principal, clsConexiónBD, es la que establece conexión con la base de datos, se le dan los parámetros de conexión y con ellos forma una expresión de conexión que luego es leída por el driver de MySQL. Además es la clase padre de todas las clases que hacen, de forma directa inserciones o lecturas sobre la base de datos. Implementa los cuatro métodos principales (Insert, update, delete y select), que luego serán sobrescritos en función de las necesidades en cada una de las clases.

2.-Clases antiguas de inserción y consulta, que al cambiar el método de inserción y lectura, teniendo ahora una clase con el sufijo *BD* para cada tabla que queremos insertar, han quedado en desuso, pero las utilizamos de referencia para generar los nuevos métodos.

### LN

LN contiene todas las clases que conforman la parte "funcional" de la aplicación, estas son, las que gestionan la creación de objetos y listas de estos en la RAM, y las que realizan algún tipo de operación sobre estos, o entre las clases. Se diferencian las siguientes:

La interfaz itfproperty (sustituida por una versión superior) que devolvía en diferentes tipos de datos en lugar de un objeto único como la versión nueva.

El gestor clsGestorLN, que es el que controla el flujo del programa y contiene los métodos que permiten el traspaso de información entre LN y LP, así como los métodos que llaman a la parte de LD para las interacciones con la base de datos.

Una clase de comparación con Comparator.

Y todas las clases de tratamiento de datos, que serán necesarias para albergar toda la información de la base de datos en memoria antes de visualizarla.

### LP

Este paquete contiene todas las clases dedicadas a la interacción con el usuario; ya sean tablas para visualizar contenidos, formularios para insertar información, o la propia pantalla principal que debe tener toda aplicación.

## Desarrollos de terceros

La clase de UtilidadesLP por cortesía de Javier Cerro Fernandez.

Conector de MySQL de ALUD.

# Detalles Técnicos

# Planificación

## Desvíos

Nada más comenzar con el proyecto ya tuvimos problemas para coincidir y perdíamos mucho tiempo en desplazamientos, esto se solucionó cuando comenzamos a usar git y discord, que nos permitieron tener más flexibilidad.

Uno de los primeros problemas de programación que nos encontramos fue el de conexión entre aplicación y la BD, fue por inexperiencia principalmente y se solucionó rápidamente. Aunque luego nos volvimos a encontrar problemas similares que solucionamos dedicándole tiempo.

Luego tuvimos un problema con los autoincrementales, que no incrementaban su valor a pesar de estar definidos en la base de datos como AI. Después de probar todas las posibilidades y de emplearle cantidad de horas, se nos sugirió rehacer la base de datos, cosa que hicimos, parcialmente, y al volver a generar la BD desde el modelo, el error desapareció.

Luego tuvimos que reestructurar todo LD y la BD para adecuarla a los requisitos que se iban añadiendo al proyecto a medida que avanzábamos en clase. Esto hizo que cosas que creíamos saber hacer tuviéramos que rehacerlas de nuevas maneras, teniendo que aprender otra vez como si empezáramos desde cero, con el tiempo que eso conllevó.

Para introducir la interfaz gráfica hubo que desechar toda la clase que hasta entonces era el menú y sustituirla por pantallas, cosa que, aunque en clase habíamos dado algunos componentes de los que forman una pantalla, se nos complicó muchísimo por la cantidad de cosas que teníamos que utilizar a la vez cuando apenas comprendíamos como generar pantallas simples. Por no hablar de cómo comunicar unas con otras, que al tener tantas restricciones de paquetes y de objetos se nos hizo muy cuesta arriba.

Y una vez hubimos adecuado el programa al funcionamiento con pantallas sueltas, tuvimos que cambiarlas por JInternalFrames, lo que supuso otro cambio, más código desechado, y más horas investigando como adecuar el código, que llegados a este punto habíamos estado más tiempo reformateándolo que aumentándolo.

Por último un problema que se ido repitiendo bastante es, debido a que usamos intellij teníamos problemas para usar el Windows Builder (realmente, su equivalente) por lo que al final tuvimos que hacer las ventanas añadiendo y ubicando los componentes por código. Por lo que perdimos mucho tiempo, para la siguiente usaremos eclipse que debe de ser más fácil.

## Objetivos fijados

Inicialmente los objetivos estaban un poco diluidos, teníamos una idea aproximada pero no sabíamos cómo íbamos a ir avanzando, queríamos tener unas librerías con la información de películas, libros, canciones, videojuegos… pero no sabíamos cómo quedaría. Por lo general nuestros objetivos han sido los de hacer lo que era necesario en el proyecto, intentando llegar a cada entrega con lo que ésta requería, olvidándonos un poco de lo que era la aplicación que teníamos en mente.

Como no habíamos hecho nunca nada del estilo fue muy difícil dimensionar el proyecto. Y visto lo visto, hemos utilizado tantos recursos en albergar una cantidad amplia de artículos en nuestra BD, cosa que nos parecía sencilla cuando no sabíamos lo que implicaba la programación, que no hemos podido realizar muchas de las funcionalidades que nos habría gustado.

## Desarrollos futuros y opciones de ampliación

El proyecto no está al 100% de su potencial, se podrían añadir más tablas en la BD, con sus respectivos inserts, consultas, deletes… también se podrían añadir diferentes tipos de ordenación a la hora de visualizar los datos de pantalla. Otra cosa que podría sumar seria la de hacer infalibles las inserts haciendo que no se pudieran repetir los datos.

También hay muchos detalles mejorables, relaciones que no nos ha dado tiempo a completar, funciones que planteamos al principio que no nos ha dado tiempo ni a plantearnos como programarlas, etc.

Aparte de eso tampoco estaría de mas generar mas javadocs, limpiar el código y hacer la interfaz de usuario mas decorada y con más imágenes. Siendo esto último muy difícil en el entorno de desarrollo que elegimos.

# Bibliografía

StackoverfLow:

<https://stackoverflow.com/questions/1990817/how-to-make-a-jtable-non-editable>

Otras páginas:

<https://www.codejava.net/java-se/swing/editable-jtable-example>

Youtube:

Canales:

<https://www.youtube.com/channel/UCdulIs-x_xrRd1ezwJZR9ww>

<https://www.youtube.com/channel/UCS3W5vFugqi6QcsoAIHcMpw>

Videos sueltos:

<https://www.youtube.com/watch?v=oBQjLgiBruM&t=8s>

<https://www.youtube.com/watch?v=yH_g6QGFqes&t=83s>

<https://www.youtube.com/watch?v=GAl1FSKvoFY>

<https://www.youtube.com/watch?v=nJBNMN4Dwss&frags=wn&ab_channel=pildorasinformaticas>

<https://www.youtube.com/watch?v=ZlyTn8PZ3Fc>