

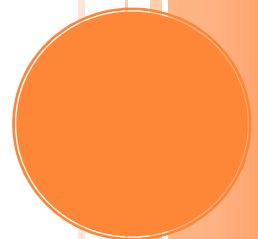
# MEDIATECA

## *Pattern Java Project*

Mediateca es una aplicación en Java que permite la gestión de préstamos de libros, comics, discos y películas a sus usuarios con una interfaz simple y amigable.

Adrián de Juan Lora

Ignacio Ribera Díaz



# Mediateca

*Pattern Java Project*

*Indices.*

## Contenido

Enunciado y requisitos .....	2
Instrucciones de ejecución y manual de usuario.....	3
Instrucciones de ejecución.....	3
Manual de usuario.....	4
Gestión usuarios .....	4
Creación de usuario.....	4
Borrar usuario .....	4
Hacer administrador.....	4
Gestión de artículos .....	5
Insertar articulo .....	5
Dar de baja artículo .....	5
Préstamo de artículo.....	6
Alquilar artículo .....	6
Devolver artículo .....	6
Diseño de la aplicación (uml) y patrones utilizados.....	7
Diseño de la aplicación (UML).....	7
Diagrama de clases .....	7
Diagrama de casos de uso .....	8
Patrones utilizados.....	10
Fachada .....	10
Estrategy .....	10
Singleton.....	11
Proxy.....	11
Iterator .....	12
Prototype .....	12
Observer .....	12
Factory method .....	12
Adapter.....	13

## ENUNCIADO Y REQUISITOS

Se desea crear un sistema gestor de una mediateca.

La mediateca contiene distintos artículos como libros, comics, cinematografía, y audio.

El sistema debe diferenciar los diferentes artículos según sus diferentes características.

Los usuarios deben de ser capaces de alquilar un artículo durante un tiempo determinado.

El administrador debe de ser capaz de gestionar tanto el alta y baja, hacer administrador a un usuario o darle de baja.

Los artículos y usuarios están registrados en una base de datos.

Cualquiera puede registrarse para utilizarla.

Un administrador puede hacer a otro usuario normal administrador.

Un administrador puede borrar un usuario no administrador.

El sistema debe permitirnos filtrar los artículos por tipo, duración, ilustrador o páginas (los tipos que lo soporten).

# INSTRUCCIONES DE EJECUCIÓN Y MANUAL DE USUARIO

## Instrucciones de ejecución.

Para la instalación del programa, descargar el instalador de NetBeans con todas las extensiones o la versión EE e instalarlo. También tiene la opción de instalar simplemente el servidor Derby en su distribución actual de NetBeans.

Una vez instalado NetBeans y el servidor, se deberá crear una base de datos de java y recrear las tablas con el script que proporcionamos en la práctica (que incluye las tablas y un usuario inicial admin con contraseña admin), la base de datos se debe llamar Mediateca.

Una vez hecho esto, importar el proyecto y cambiar la ruta de la base de datos (ir a la clase BDSingleton y cambiar la dirección en el String url)

```
public class BDSingleton {  
  
    private static BDSingleton instancia = new BDSingleton();  
    private static String url = "jdbc:derby://localhost:1527/Mediateca";  
    private static Connection conn;  
    private Statement stm;
```

Finalmente, ejecutamos el Home.java de la carpeta interfaz y podremos empezar a utilizar la aplicación.

# Manual de usuario

## Gestión usuarios

### Creación de usuario

Para crearse un usuario para acceder a la mediateca, simplemente pulsamos el botón “Crear usuario”, rellenamos los campos y le damos al botón “Registrarse”. En caso de no querer registrarse, pulsar “Cancelar”.



**Mediateca**

DNI:

Contraseña:



**Alta de usuario**

Nombre:

Apellido:

DNI:

Sexo:

Edad:

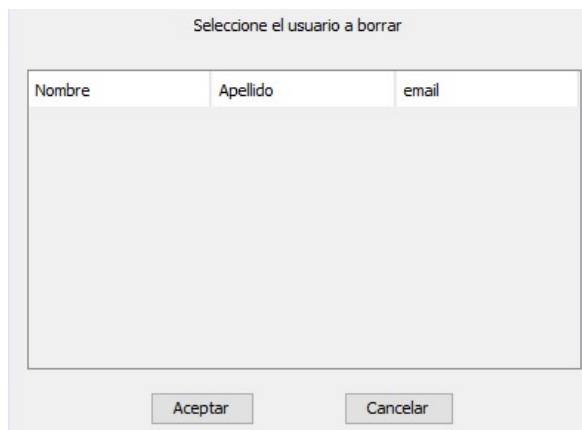
Email:

Contraseña:

Repetir contraseña:

### Borrar usuario

Para borrar un usuario del sistema, debemos ser administradores y entrar a la opción “Borrar usuario”. Una vez ahí, elegiremos el usuario que queremos hacer administrador y le damos a “Aceptar”.



Seleccione el usuario a borrar

Nombre	Apellido	email

### Hacer administrador

Para hacer a un usuario administrador del sistema, debemos ser administradores y entrar a la opción “Hacer admin”. Una vez ahí, elegiremos el usuario que queremos hacer administrador y le damos a “Hacer admin”.

**¿Qué usuario quieres hacer administrador?**

Nombre	Apellido	email

Hacer administrador Cancelar

## Gestión de artículos

Solo un usuario administrador puede gestionar los artículos de la mediateca.

### Insertar artículo

Para insertar un artículo a la mediateca, el administrador seleccionará la opción “Insertar artículo”, esto le llevará a una pestaña donde tendrá que elegir el tipo de artículo que quiere crear (libro, comic, película, disco). Cuando lo tenga seleccionado le dará a aceptar y le aparecerá la pantalla del artículo correspondiente. En esta pantalla introduciremos los datos del artículo y le daremos a “Introducir”. En el caso de no querer, darle a “Cancelar”.

**Seleccione el tipo de elemento a introducir**

Libro ▾

Introducir artículo Cancelar

**Rellene los campos del comic**

Título:

Autor:

Ilustrador:

Género:

Páginas:

Síntesis:

Introducir Cancelar

### Dar de baja artículo

Para dar de baja un artículo, el administrador debe seleccionar la opción “Dar de baja artículo”. Aquí seleccionará el artículo que desea dar de baja y le dará a “Aceptar”.

Título	Genero	Autor

Aceptar Cancelar

## Préstamo de artículo

Tanto un usuario administrador como uno normal, pueden tanto alquilar un artículo como devolverlo.

## Alquilar artículo

Para alquilar un artículo, el usuario debe darle a “Alquilar”. Le aparecerá una pantalla en la que puede buscar de diferentes maneras. Podemos seleccionar el tipo de artículo que queremos buscar (libro, comic, película, disco) y dependiendo del que seleccionemos nos deja filtrar por unos campos u otros: Duración (audio y película), páginas (libro y comic), ilustrador (comic), actores (película). Una vez seleccionado el objeto deseado, nos aparecerán sus atributos. Para alquilarlo le damos a “Aceptar”.

Filtros generales

Título:

Autor:

Orden:

Titulo

Genero:

Filtros específicos

Ninguno

Duración:

Paginas

Ilustrador:

Actores:

Comparar:

>

Buscar

Cancelar

Titulo	Autor	Genero
--------	-------	--------

## Devolver artículo

Para devolver el artículo que se tiene alquilado, le daremos a “Devolver” y nos aparece una pantalla mostrando el artículo que tiene el usuario logeado. Para confirmar la devolución le damos a “Devolver” y si queremos cancelarlo a “Cancelar”.

¿Desea devolver su artículo?

Artículo.

Devolver

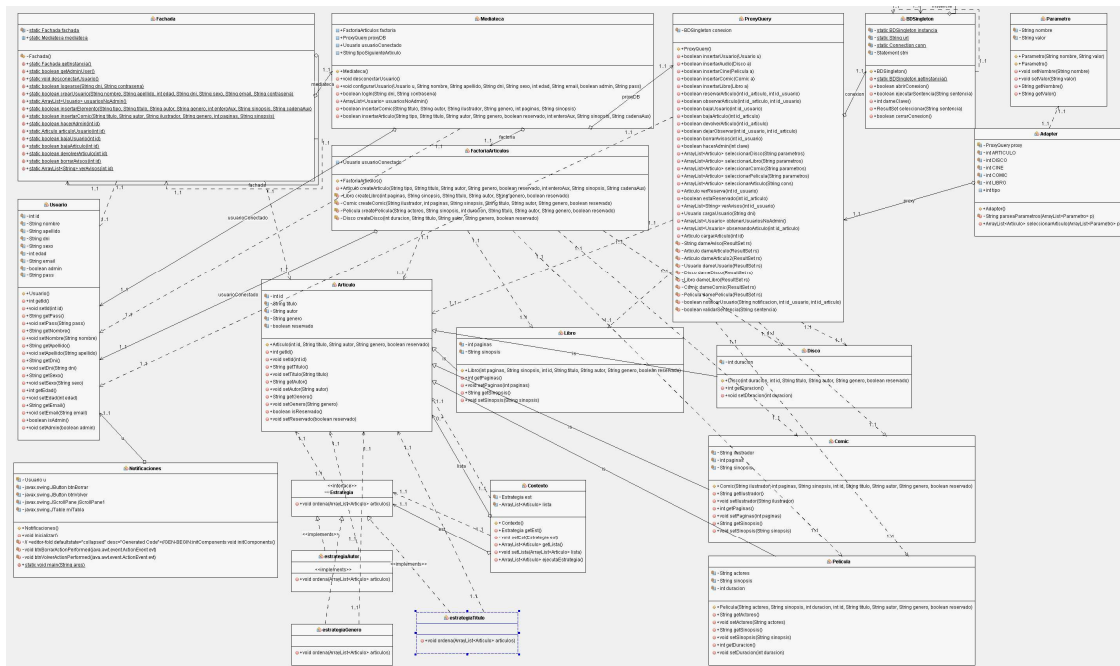
Cancelar

# DISEÑO DE LA APLICACIÓN (UML) Y PATRONES UTILIZADOS

## Diseño de la aplicación (UML)

### Diagrama de clases

A continuación, aparece el diagrama de clases del programa.



El programa está estructurado de la siguiente manera:

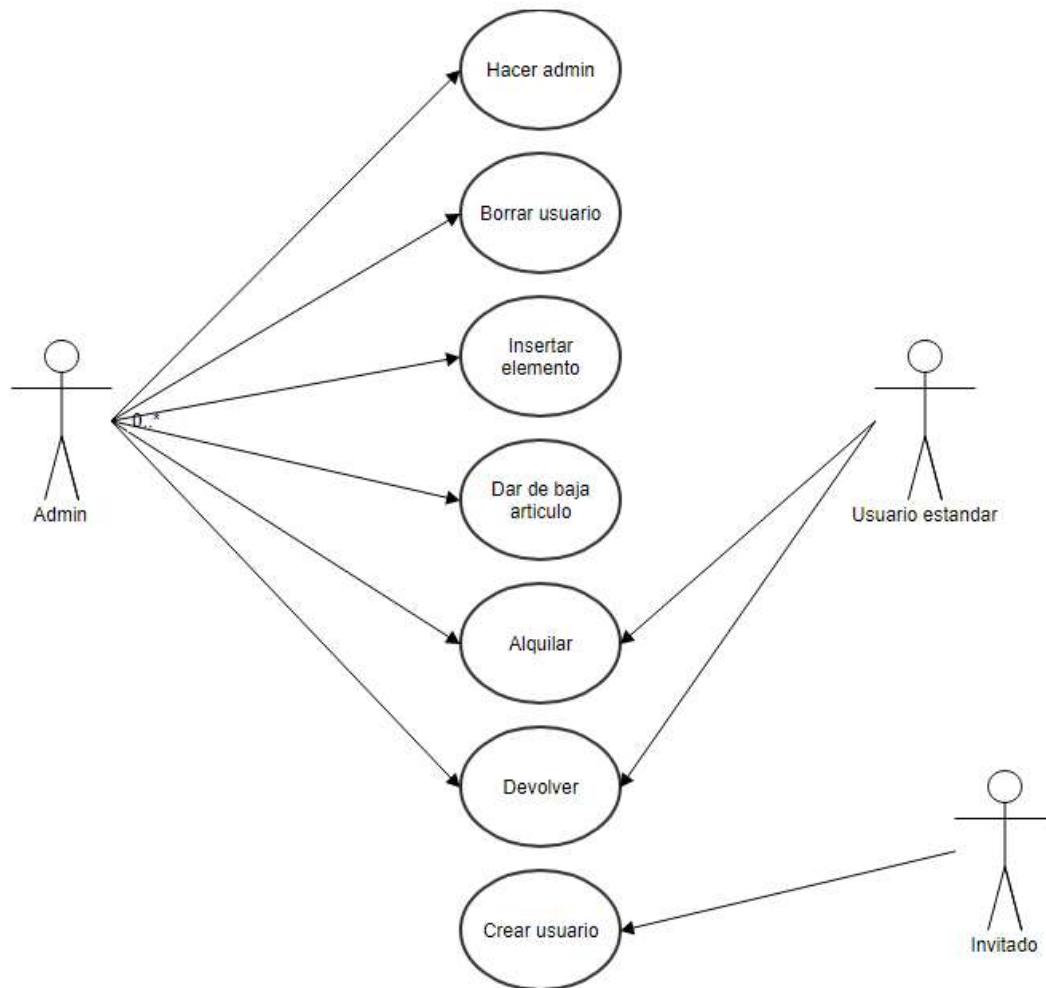
- Hay una clase **Articulo** no abstracta (utilizamos instancias para la base de datos) de la que heredan los 4 tipos de artículos: libros, comics, discos y películas. De esta manera, los cuatro compartirán los atributos `id`, `titulo`, `autor` y `género`.
- Estas clases serán instanciadas por un método en la clase fachada, el cual devuelve un objeto de tipo **Articulo**.
- La **Fachada** es una clase estática en la que está almacenada la mediateca, y es la clase que da interfaz unificada del programa.
- La mediateca contiene el objeto **proxy**, el cual nos comunicará con el objeto **DBSingleton** que accede a la base de datos. También tiene un `al` usuario logueado en ese momento y un objeto **factoría** para construir los artículos.
- La clase **usuario** se inicializa con unos valores predeterminados, que luego se configuran para construir cada usuario concreto y cumplir el patrón **prototype**.
- La clase **ProxyQueryDB** es la clase que tiene acceso a la clase estática **DBSingleton**, para unificar el acceso a la base de datos y solo instanciar una conexión a la base de datos (**DBSingleton**):



- La clase Parametro se usa para las búsquedas en base de datos, para meter condiciones de búsqueda. Tiene nombre y valor, por ejemplo: Nombre:titulo Valor:DragonBall.
- La clase Adapter se preocupa de saber en qué tabla hay que buscar y cuál es la sentencia que hay que enviar para la búsqueda.
- La clase Estrategia implementa 3 estrategias diferentes para ordenar las búsquedas (estrategiaGenero, estrategiaAutor, estrategiaTitulo).

## Diagrama de casos de uso

Ahora presentaremos los casos de uso planteado en nuestro programa, aquí tenéis un diagrama explicativo.



Crear usuario: un usuario no registrado, podrá registrarse en la mediateca y obtendrá una cuenta de nivel no administrador.

Hacer administrador: un usuario administrador podrá acceder a la lista de usuarios no administrador y darle estos derechos.

Borrar usuario: un usuario administrador podrá borrar un usuario (no administrador) de la aplicación.

Insertar elemento: un usuario administrador podrá insertar nuevos artículos a la mediateca (de los 4 tipos).

Dar de baja artículo: un usuario administrador podrá quitar un artículo del sistema.

Alquilar: un usuario (tanto administrador como normal) podrá alquilar un artículo a través de la aplicación.

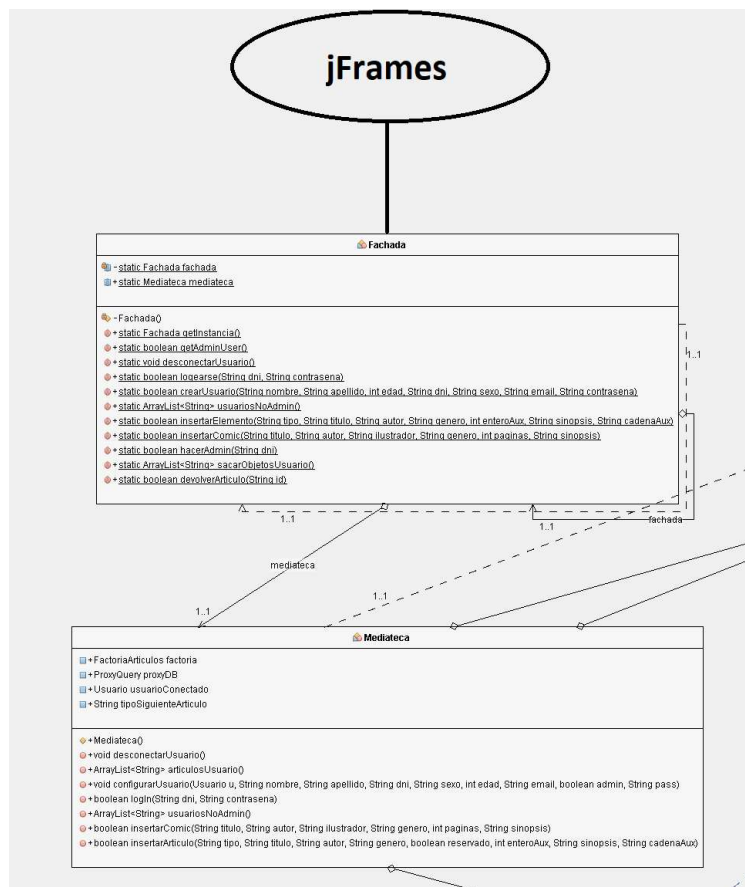
Devolver: un usuario (tanto administrador como normal) podrá devolver el artículo que tiene alquilado.

## Patrones utilizados

En esta parte explicaremos todos los patrones utilizados en la práctica.

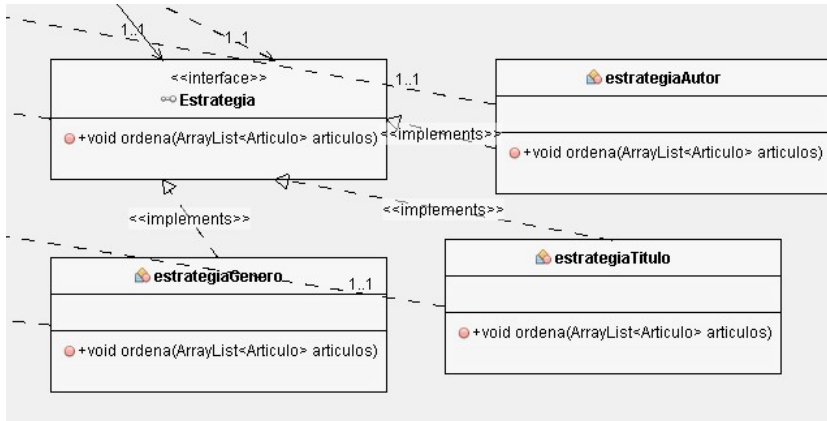
### Fachada

Para la implementación de este patrón hemos definido una clase estática fachada, en la que se encontraba instanciado un objeto mediateca. De esta manera, conectamos la mediateca y las ventanas de la interfaz gráfica unificando la interfaz en un solo objeto, llamando a funciones de la fachada exclusivamente. Esto ha sido combinado con el patrón singleton, ya que solo se puede instanciar un objeto de tipo Fachada (clase de tipo static).



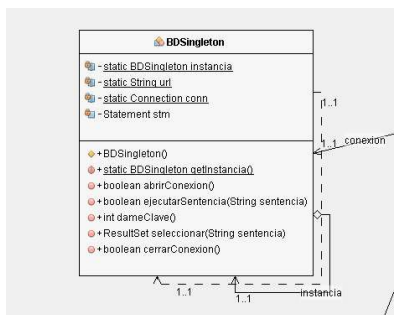
### Estrategy

La clase Estrategia tiene un método ordenar y puede implementar 3 estrategias distintas: `estrategiaGenero`, `estrategiaTitulo` y `estrategiaAutor`.



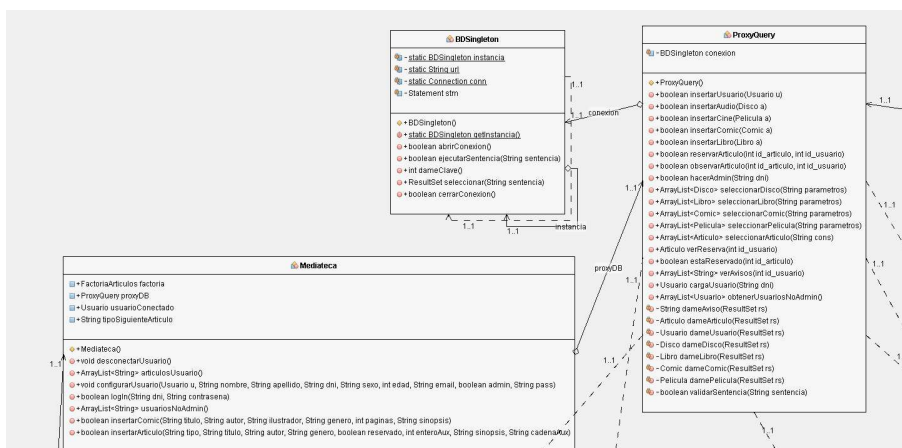
## Singleton

Hemos utilizado este patrón en dos ocasiones. La primera que pensamos fue para el acceso a la base de datos, para limitarlo a un único objeto (DBSingleton de tipo static). La segunda fue para comunicar el programa con la interfaz, implementamos el singleton a la fachada, para que todos los JFrame tuvieran acceso a una única (y la misma) mediateca (instanciada dentro de la fachada).



## Proxy

Para limitar el acceso a base de datos, se ha creado una clase ProxyDB, que es la única clase con acceso directo a la clase DBSingleton, que es la que se comunica con la base de datos. Este patrón ha sido combinado con el Singleton, de tal manera, que DBSingleton solo pueda instanciarse una vez, y así tener un único acceso a la base de datos.

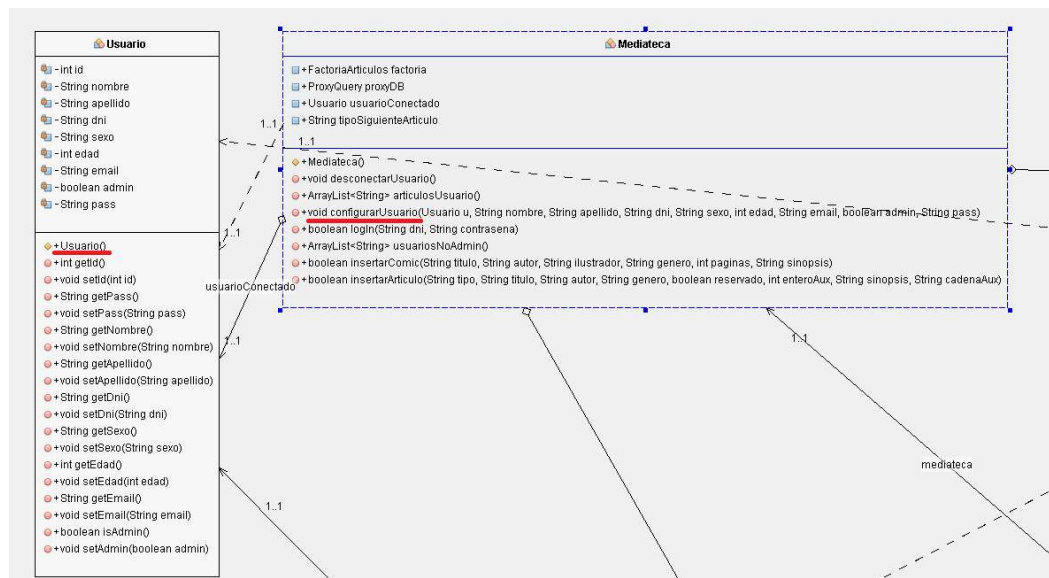


## Iterator

Este patrón es el más sencillo de implementar, simplemente, cuando tenemos que recorrer una colección (en este caso hemos usado ArrayList), declaramos un iterador (`Iterator iterador = arrayUsuarios.iterator();`) y tenemos acceso de manera cómoda a los elementos del ArrayList.

## Prototype

El patrón prototype ha sido implementado para la creación de usuarios, la clase usuario tiene un constructor sin parámetros que instancia siempre un usuario modelo. También dispone de un método `configurarUsuario`, al que se le pasan los parámetros del nuevo usuario.

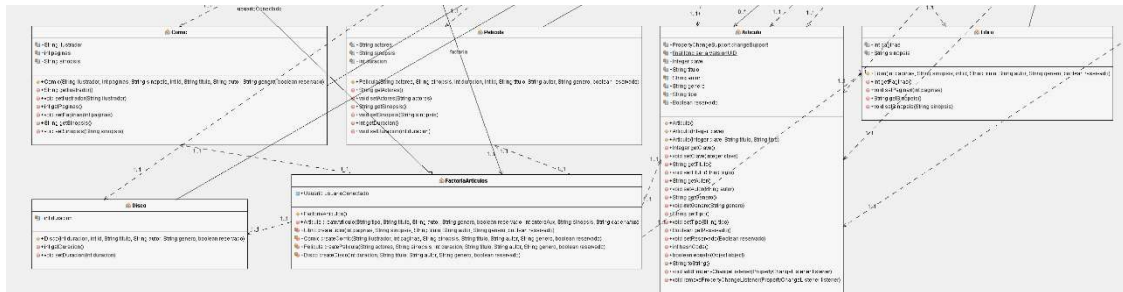


## Observer

Implementamos el patrón observer a nivel de base de datos, con el podremos seguir un artículo para cuando se devuelva le notifique al usuario que deseaba alquilarlo. Tenemos una tabla observaciones que tiene los observadores (que son los usuarios y los objetos que observa cada uno).

## Factory method

Tenemos 4 artículos diferentes, pero todos heredan de Artículo. Por esto, hemos creado una clase Factoría, que tiene 1 función para instanciar cada uno de los elementos, y luego una función genérica que combina las 4. De esta manera, con una sola función, instanciamos artículos de los 4 tipos, sin tener que usar 4 funciones diferentes.



## Adapter

El patrón adapter se implementa en la parte de base de datos con la clase Adapter. Lo utilizamos para saber en qué tabla hay que buscar y cuál es la sentencia que hay que enviar para la búsqueda

