

## Tecnologías de Desarrollo de Software

(3º Curso del Grado en Informática)

Curso 2020/2021

22 de septiembre de 2020

---

### Caso Práctico

#### *AppMusic*

Existen varias aplicaciones web y de escritorio muy extendidas para la reproducción de música vía streaming, como son Spotify o StreamSquid. Esta especificación de requisitos está inspirada en este tipo de aplicaciones y considera una parte de su funcionalidad básica. En nuestro caso se tratará de una aplicación de escritorio. En este documento, primero se detallarán las funcionalidades que deberá ofrecer la aplicación. Luego se mostrarán y comentarán algunos ejemplos de ventanas que podrían componer la interfaz gráfica de usuario. Después se indicará cuál debe ser la arquitectura software de la aplicación. Finalmente se proporcionarán detalles sobre la gestión del proyecto, documentación a entregar y la evaluación.

#### Requisitos

---

##### *Login y Registro de usuarios*

Para utilizar los servicios del sistema, los usuarios deben estar registrados y realizar un *login* con su nombre y contraseña. Para registrarse un usuario debe indicar su nombre y apellidos, fecha de nacimiento, email, usuario y contraseña.

##### *Canciones*

Una **canción** es definida por su título, intérprete, estilo y la ruta de un fichero mp3. Los ficheros mp3 de las canciones estarán almacenados en disco y separados en carpetas cuyo nombre corresponderá a un estilo musical (Pop, Rock, Clásico, Flamenco, Jazz, etc.). Cada nombre de fichero de canción está formado por el nombre del intérprete y el título separados por un guion “-”, por ejemplo: “Nina Simone-Fly Me To The Moon.mp3”. Si es interpretada por varios artistas, sus nombres se separan con “&”, por ejemplo “Cameron & Tomatito & Paco De Lucia - Como El Agua.mp3”.

##### *Buscar y Reproducir Canciones*

Un usuario podrá buscar canciones utilizando como filtros de búsqueda el título, intérprete y estilo musical. Las búsquedas para intérprete y título se harán por subcadenas (textos de búsqueda contenidos en el título o el nombre del intérprete). Al finalizar una búsqueda, el **usuario** podrá seleccionar y reproducir cualquiera de la lista de canciones retornada. **Una canción puede ser pausada mientras se reproduce. También se podrá avanzar o retroceder en la lista para escuchar la siguiente o anterior canción,** respectivamente (la siguiente canción a la última será la primera, y la anterior a la primera la última canción).

##### *Listas de reproducción de canciones (playlists)*

Un usuario podrá crear listas de canciones (**playlist**) que tendrán un nombre. Las canciones que integran una *playlist* son elegidas realizando búsquedas en la colección de canciones existente. En cualquier momento se

pueden añadir y eliminar canciones a una playlist. El usuario podrá seleccionar una playlist para reproducir sus canciones.

### ***Canciones recientes***

Además de las listas de canciones, la aplicación mantendrá una **colección de las canciones más recientemente** reproducidas por el usuario. El tamaño de esta colección tendrá un valor por defecto de 10 que el usuario no puede modificar. El usuario podrá acceder a esta colección para reproducir sus canciones.

### ***Usuario “Premium”***

Una vez registrado un usuario podrá, en cualquier momento, activar su cuenta como “Premium” para tener beneficios como disponer de más canciones (no se considerará en nuestro caso), generar un archivo PDF con las playlists y reproducir las 10 canciones más escuchadas en toda la aplicación. Un usuario deberá pagar una cantidad de dinero para ser premium. La aplicación debe poder aplicar descuentos tales como “reducción de un 20% durante un período de tiempo” o “reducción del 50% para los mayores de 65 años”. No se aplicarán dos descuentos al mismo tiempo. El sistema incorporará al menos dos tipos de descuentos y debe permitir añadir nuevos descuentos en el futuro. La forma y realización de pago quedan fuera del ámbito de este proyecto.

### ***Generar archivo PDF con listas de canciones***

Para implementar esta funcionalidad, solo disponible para usuarios “premium”, se utilizará un API de creación de archivos PDF (por ejemplo, iText<sup>1</sup>) para generar un archivo que incluya las listas de canciones de un usuario: nombre de la lista y listado de canciones (título, intérprete, estilo). En el siguiente sitio se puede encontrar información sobre el uso de esta librería:

<http://soloinformaticayalgomas.blogspot.com.es/2010/12/generar-un-documento-pdf-desde-java.html>

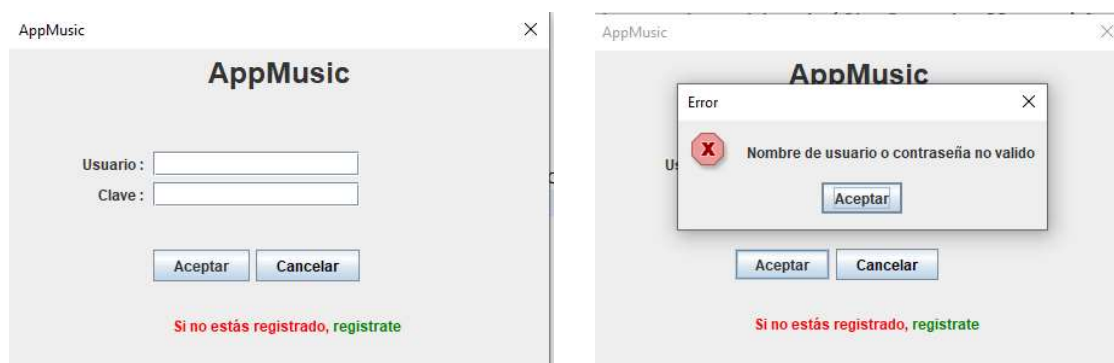
Cada grupo tiene libertad para considerar funcionalidad adicional o realizar cambios sobre la funcionalidad aquí explicada, **pero deberá consultarlo con el profesor de la asignatura.**

## **Interfaces de usuario**

---

A continuación, se detallan las ventanas que formarán parte de la aplicación con el objetivo de facilitar su desarrollo. No obstante, las capturas mostradas son sólo orientativas y **los alumnos tienen libertad para optar por otros diseños, componentes, layouts, etc.**

**Ventana Login:** La ventana de entrada a la aplicación aparece abajo a la izquierda. Si el usuario y clave introducidas son correctas aparecerá la ventana principal que se comenta más abajo. En caso contrario debe aparecer una ventana de error como la mostrada abajo a la derecha. La ventana “Login” también permite registrarse haciendo clic con el ratón sobre el rótulo mostrad0: “Si no estás registrado, regístrate”.

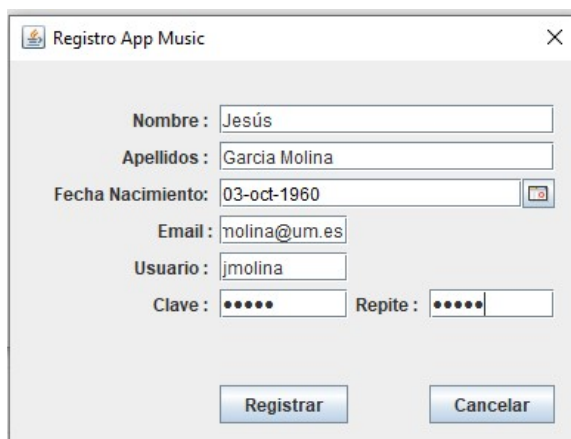


---

<sup>1</sup> <https://itextpdf.com/es>

Se entregará a los alumnos un proyecto que implementa login y registro en todas sus capas y que podrán adaptar a AppMusic.

**Ventana ‘Registro’.** Los usuarios deberán introducir su nombre, apellidos, fecha de nacimiento, email, usuario y contraseña para registrarse. La contraseña se repite dos veces y debe coincidir. Se comprueba que se hayan introducido valores en todos los campos de entrada, las dos claves coincidan y no haya registrado un usuario con el mismo nombre o email. Se deben mostrar mensajes de error como se observa en la ventana de abajo a la derecha. Para introducir la fecha se utilizará el componente JCalendar disponible en <https://toedter.com/jcalendar/>.



Registro App Music

Nombre : Jesús

Apellidos : Garcia Molina

Fecha Nacimiento: 03-oct-1960

Email : molina@um.es

Usuario : jmolina

Clave : \*\*\*\*\* Repite : \*\*\*\*\*

Registrar Cancelar



AppMusic

Registro App Music

Nombre : Jesús

Apellidos : Garcia Molina

Fecha Nacimiento: 03-oct-1960

Email : jmolina@um.es

Usuario : jesus

Clave : \*\*\*\*\* Repite : \*\*\* \*

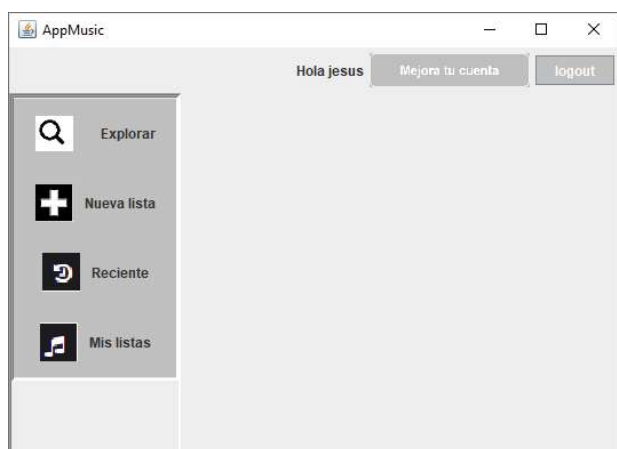
Registrar Cancelar

\* Las dos claves deben coincidir

\* El usuario ya existe

**Ventana Principal.** Como se puede ver abajo, la ventana incluye 3 componentes en la zona norte y alineados a la derecha: Un saludo al usuario en la forma “Hola <nombre de usuario>”, un botón para convertir la cuenta en premium, y un botón para salir de la aplicación. Por otro lado, en la zona oeste de la ventana aparece una columna con una lista de botones que permiten acceder a la funcionalidad ofrecida por AppMusic: *buscar canciones*, *crear una nueva lista*, *mostrar lista de canciones recientemente escuchadas* y *mostrar playlists del usuario*. Cada botón muestra un icono y un rótulo, y al clicar sobre cada uno de ellos se mostrará un panel diferente siempre en la zona centro de la ventana principal como se explica abajo.

Al abrir esta ventana, las nuevas canciones en disco serán registradas en la base de datos. Este registro consistirá en grabar los objetos que representan las canciones.



AppMusic

Hola jesus Mejora tu cuenta logout

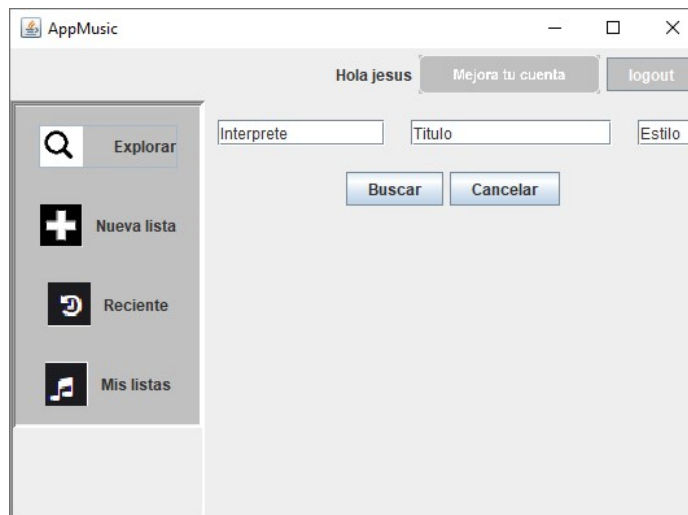
Explorar

Nueva lista

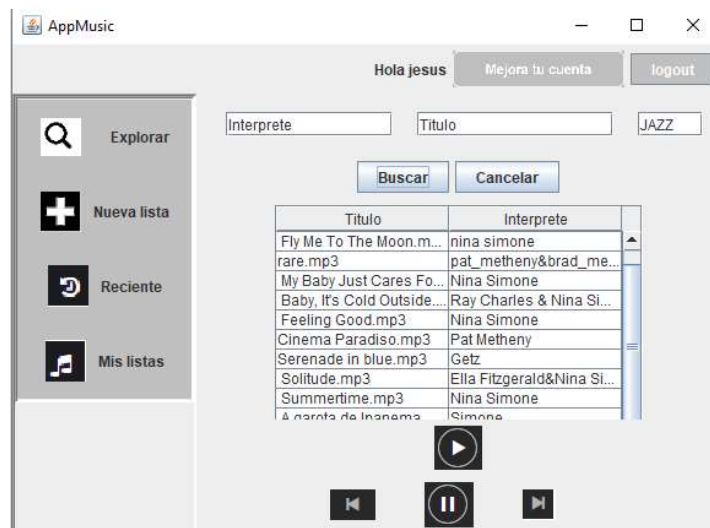
Reciente

Mis listas

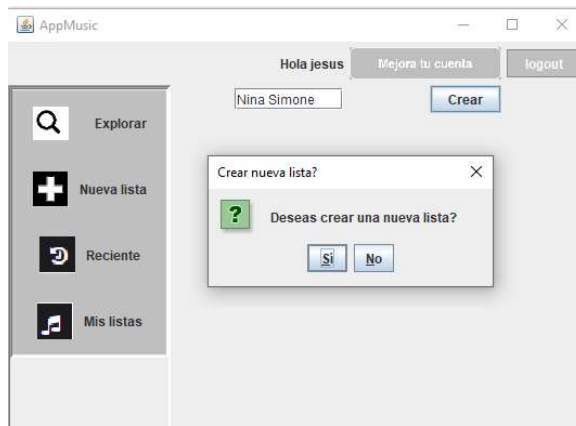
Ventana “Explorar canciones”. Primero aparece un panel con tres campos de entrada para los filtros: Intérprete, título y estilo, y debajo de ellos los botones para arrancar la búsqueda y cancelar. El *estilo* musical debe ser introducido a través de una lista desplegable (*combobox*).



Cuando el usuario introduzca los valores de los filtros y haga clic en el botón “Buscar” aparecerá una tabla con las canciones que satisfacen los filtros. La ventana de abajo muestra una tabla para canciones de estilo “POP”. Debajo de la tabla de canciones aparecen los botones para reproducir, detener reproducción, reproducir siguiente y anterior canción en la lista. Se podrían combinar filtros, por ejemplo, canciones de jazz de una intérprete cuyo nombre contenga “Nina” o canciones de estilo pop cuyo título incluya “Tears in Heaven”. Una canción también se puede reproducir con un doble-clic sobre una canción seleccionada.



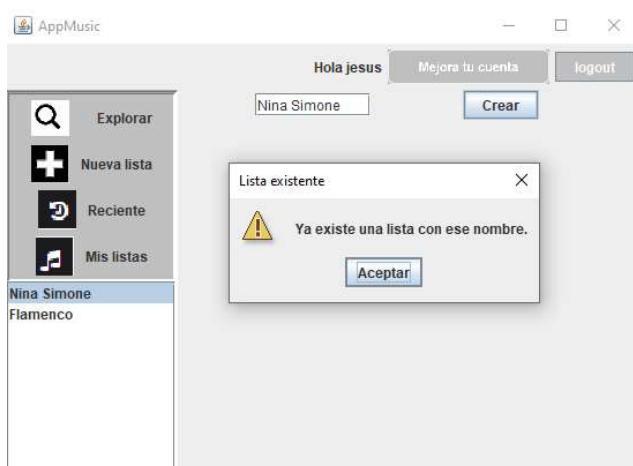
Ventana ‘Nueva Lista’: Permite crear una *playlist*. En la parte superior aparece un campo para introducir el nombre de la playlist y junto a él un botón para crearla. Al hacer clic sobre este botón se abre una ventana de dialogo que pide confirmación sobre si se quiere crear o no, como se observa en la ventana de la siguiente página que crea una playlist llamada “Nina Simone”.



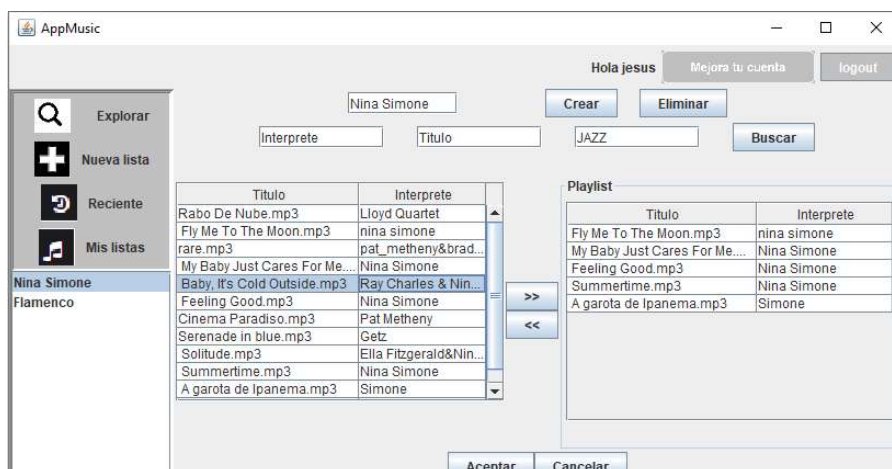
Si se acepta crear la playlist aparecen en una misma fila los campos para introducir los tres filtros y el botón para buscar canciones, al igual que en la ventana “Explorar canciones”, pero ahora cuando se realiza una búsqueda junto a la tabla con las canciones que satisfacen los filtros se abre otra que mostrará las canciones añadidas a la playlist en cada instante. Los botones “>>” y “<<” situados entre ambas tablas permiten ir añadiendo canciones seleccionadas en la tabla de la izquierda o eliminando canciones añadidas y seleccionadas en la playlist, respectivamente. El botón “aceptar” permite finalizar la creación de la playlist.



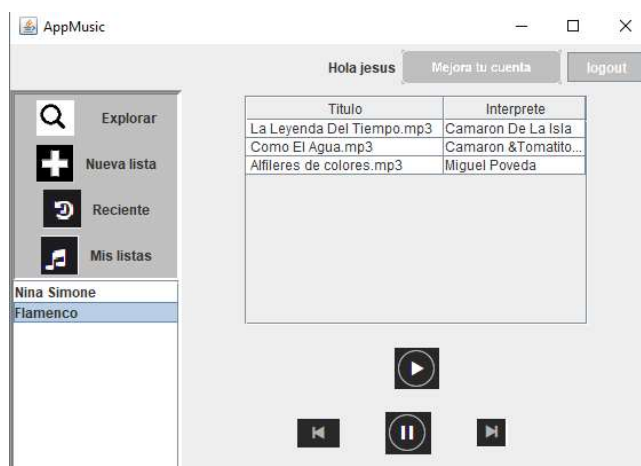
Esta ventana también permite actualizar una playlist. Cuando el nombre dado coincide con una playlist ya creada se avisa al usuario.



Y cuando el usuario acepta, aparecerán los campos para el filtro, el botón “Buscar” y un botón “Eliminar” que permitirá borrar la lista. Al realizar una búsqueda, la tabla “Playlist” mostrará las canciones que tenga en ese momento, como sucede abajo para la playlist “Nina Simone”.



**Ventana ‘Mis listas’:** Al hacer clic sobre este botón se abre debajo una lista con las playlists creadas. Por defecto aparece seleccionada la primera. Al seleccionar una se muestra una tabla con las canciones y debajo los botones de reproducción. En la ventana de abajo aparecen dos listas: “Nina Simone” y “Flamenco”, y se ha seleccionado la segunda.



**Opcional:** El alumno puede proporcionar la funcionalidad de reproducir todas las canciones de la lista por defecto o activar un modo aleatorio, además de la selección personal. En clase práctica se proporcionarán detalles necesarios.

**Ventana ‘Reciente’:** Muestra una tabla con una lista de las últimas canciones reproducidas (por defecto se mostrarán las 10 últimas canciones reproducidas por el usuario) y los botones para su reproducción. Será la ventana que se mostrará por defecto al realizar el login.



Ventana “Canciones más reproducidas por usuario”: Mostrará la lista de canciones más escuchadas por un usuario (canción, interprete y número de veces reproducida).

## Reproducción de música

Se reproducirán canciones en formato mp3. Para ello, el alumno podrá utilizar la clase `MediaPlayer` de la librería JavaFX de Java (parte del JDK y JRE desde Java 7) que proporciona los métodos `play()`, `stop()` and `pause()` para reproducir, detener y pausar una canción, respectivamente. El siguiente código permite reproducir una canción supuesto que un objeto que representa una canción tiene el método `getRutaFichero()` que retorna la ruta del fichero mp3.

```
import java.io.File;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;

// activar reproductor
try {
    com.sun.javafx.application.PlatformImpl.startup(()->{});
} catch (Exception ex) {
    ex.printStackTrace();
    System.out.println("Exception: " + ex.getMessage());
}

// reproducir una canción

String fileName = cancion.getRutaFichero();
File f = new File("C:\\tds\\canciones\\"+fileName);
Media hit = new Media(f.toURI().toString());
MediaPlayer = new MediaPlayer(hit);
mediaPlayer.play()
```

## Arquitectura de la aplicación

Se organizará la aplicación de acuerdo a un modelo de tres capas (Presentación, Lógica de Negocio y Almacenamiento) que será descrito en un seminario de prácticas (también en el primer tema de teoría se discutirán los aspectos básicos). *Se entregará un proyecto de ejemplo “Tienda” para ilustrar cómo aplicar esta arquitectura.*

Se utilizará la tecnología **Java Swing** para la implementación de la interfaz de usuario y un **servicio de persistencia** desarrollado en la asignatura para el almacenamiento de los datos de la aplicación. Se entregará documentación sobre el uso de estas tecnologías y se explicarán además en seminarios de prácticas.



Se utilizará el **patrón DAO** para desacoplar la capa de almacenamiento del resto de la aplicación, de acuerdo a las explicaciones proporcionadas en las clases de teoría y prácticas. Cuando se entregue a los alumnos el diagrama de clases de la aplicación **se indicará cuáles son las clases persistentes**.

Todas las dependencias de librerías externas deben manejarse con **Maven** a excepción de la librería del servicio de persistencia, que debe ser importada en la carpeta *lib* del proyecto y añadida al *build path*

## Componentes

---

Los alumnos deberán crear un componente propio y usar componentes de terceras partes.

- Para introducir la fecha de nacimiento en el registro se utilizará alguno de los componentes Java beans disponibles por la red que ofrecen esta funcionalidad, se sugiere usar el componente JCalendar (<http://toedter.com/jcalendar/>) cuyo aspecto se muestra abajo.



- Más adelante se entregará a los alumnos la especificación de un sencillo componente Java Bean que deberán implementar.
- El componente *Luz*, que se manejará en las clases prácticas, será utilizado para disparar al anterior componente.

## Grupos de prácticas

---

Los alumnos deberán formar **grupos de dos** que deberán ser del mismo grupo de teoría salvo casos excepcionales. Se recomienda organizar el trabajo de forma que los dos miembros de un grupo puedan trabajar en paralelo. Por ejemplo, un alumno puede implementar las clases de la interfaz de usuario mientras otro se encarga de las clases relativas a la persistencia de los datos. Los dos alumnos deben tener un conocimiento de todos los aspectos de la práctica y deberían haber programado en cada capa de la arquitectura.

## Gestión del proyecto

---

Cada grupo deberá crear un proyecto **Maven** que debe ser almacenado en un repositorio privado **Git** (es posible elegir entre *github* y *gitlab*). Las dependencias a las librerías externas usadas se manejarán con Maven. Los alumnos deben invitar a su profesor de prácticas al repositorio creado para que pueda acceder a su código en cualquier momento. Se dedicará un seminario de prácticas a explicar el uso de Maven y otro el de Git.

## Código

---

El código debería ser escrito utilizando **expresiones lambda** y **streams** en aquellos puntos en los que se considere apropiado. El alumno no debe cometer errores relacionados con los patrones GRASP y principio separación modelo-vista. Los conocimientos sobre patrones de diseño se valoran principalmente en el examen de teoría de la asignatura. No obstante, la realización de esta práctica conlleva el uso de algunos patrones aplicados directamente por el alumno o indirectamente al utilizar alguna construcción o funcionalidad de Java o



de los servicios proporcionados por los profesores. El alumno deberá indicar en la documentación cuáles son estos patrones y comentar brevemente su uso.

## Hitos

---

La práctica se organizará en las siguientes fases:

1. Realización del **diagrama de clases inicial**. La fecha tope de entrega será el **18 de octubre**. Se creará una tarea en el Aula Virtual para que cada grupo pueda enviar su diagrama de clases junto a una explicación de los detalles que considere oportuno (**un único envío por grupo**). El archivo PDF entregado debe incluir los nombres de los alumnos.
2. En los seminarios de prácticas de la semana del 19 de octubre se discutirá el diagrama de clases entregado por los profesores y que los alumnos deberían utilizar en la implementación del caso práctico.
3. Cada grupo debería aplicar una estrategia de desarrollo iterativo completando de forma incremental la funcionalidad y escribiendo el código necesario para cada capa: a) diseño e implementación de la interfaz de usuario a partir de los conocimientos adquiridos en los seminarios de Swing (el diseño de ventanas expuesto arriba pretende facilitar el trabajo de los alumnos); b) diseño e implementación del controlador; c) implementación de la lógica del dominio; d) implementación de la persistencia.
4. Entrevista con el profesor para seguimiento del trabajo práctico (**última semana de noviembre y primera de diciembre**). Para realizar esta entrevista el grupo debería haber abordado ya el desarrollo de cada capa.
5. Uso y desarrollo de componentes según se explicó arriba.
6. Se implementarán test unitarios para, al menos, una clase del modelo.

**Fecha de entrega: 10 de enero**

Se creará una **tarea** en el Aula Virtual para la entrega de la documentación comentada abajo (archivo pdf) y el proyecto Eclipse desarrollado exportado a un fichero zip.

## Documentación

---

Constará de las siguientes partes:

1. Diagrama de clases del dominio.
2. Un diagrama de colaboración o secuencia para la operación de añadir playlist (se ejecuta al hacer clic sobre el botón “*Aceptar*” del panel “*Nueva Lista*”) o equivalente.
3. Una breve explicación de la arquitectura de la aplicación y decisiones de diseño que se consideren de interés para la comprensión del trabajo.
4. Breve explicación de cada uno de los patrones de diseño utilizados (también aquellos indirectamente aplicados al ser parte de Swing).
5. Breve explicación sobre los componentes utilizados.
6. Tests unitarios implementados
7. Un pequeño manual de usuario que explique cómo usar la aplicación.
8. Observaciones finales que el alumno desee comentar (deben incluir una estimación del tiempo dedicado)

## Evaluación

---

Cada parte se valorará con el siguiente porcentaje:

- Modelo del dominio de clases (10%)
- Diseño de la interfaz (20%)
- Uso de componentes (15%),
- Aplicación de patrones (5%),
- Código (separación en tres capas, legibilidad, uso de principios básicos de programación OO, implementación de patrones de diseño, uso de Swing, comentarios) (35%),
- Uso de JUnit (5%),
- Documentación entregada (10%).
- Es obligatorio el uso de Git y Maven.