

Tecnologías de Desarrollo de Software

(3º Curso del Grado en Informática)

Curso 2023/2024

22 de septiembre de 2023

Caso Práctico

AppMusic

Existen varias aplicaciones web y de escritorio muy extendidas para la reproducción de música vía streaming, como son Spotify o Amazon Music. 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 email, fecha de nacimiento, usuario y contraseña. La aplicación también permitirá acceder utilizando una cuenta de GitHub como se explica más adelante.

Canciones

Una canción es caracterizada por su título, intérprete y estilo, y es almacenada en formato mp3 en disco. Los archivos mp3 de las canciones estarán organizados en carpetas cuyos nombres serán los de un estilo musical, por ejemplo, Pop, Rock, Flamenco, o Jazz. Los nombres de los archivos mp3 de las canciones estarán formados por el nombre del intérprete, seguido de un guion “-” y del título. 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”. Los profesores proporcionarán una colección de archivos mp3 de canciones.

Buscar y Reproducir Canciones

Un usuario podrá buscar canciones utilizando los siguientes filtros de búsquedas: *título*, *interprete*, *estilo musical*, y *si es canción favorita* (ver más adelante). Las búsquedas para intérprete y título se harán por subcadenas (textos de búsqueda contenidos en el título o el nombre del intérprete). Las búsquedas pueden ser ***combinadas***, por ejemplo, canciones de “Nina Simone” cuyo título contiene la palabra “love”. Al finalizar una búsqueda, el usuario podrá seleccionar y reproducir cualquiera de las canciones en 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 canciones (playlists)

Un usuario podrá crear sus listas de canciones (*playlists*), indicando un nombre. Un usuario podrá añadir canciones a la *playlist* eligiéndolas de una lista de canciones obtenida mediante una búsqueda con algún filtro. En cualquier momento, un usuario podrá añadir o quitar canciones de una playlist. Un usuario podrá seleccionar cualquier canción de una playlist para su reproducción y también elegir el modo de reproducción aleatorio (opcional).

Canciones recientes

Además de las *playlists* del usuario, la aplicación mantendrá una lista de las canciones más recientemente reproducidas por el usuario. Esta lista tendrá un tamaño máximo *max*, y contendrá las últimas *max* canciones reproducidas por el usuario. 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 obtener dos funcionalidades extra de la aplicación: (1) generar un archivo PDF con la información de las *playlists* del usuario, y (2) reproducir las 10 canciones más escuchadas considerando las reproducciones de todos los usuarios. Un usuario debe pagar una cantidad de dinero para ser premium. La aplicación debe poder aplicar descuentos tales como, por ejemplo: “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 gestión del pago a una entidad bancaria queda fuera del ámbito de este proyecto. Pero sí se debe calcular la cantidad a pagar cuando un usuario activa la opción “Premium”.

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](#)) para generar un archivo (cuyo nombre sea ‘*nombre del usuario.pdf*’) que incluya las playlists del usuario: nombre de la *playlist* y listado de canciones (de cada canción se desea mostrar el *título*, *intérprete*, *estilo*). En el siguiente sitio se puede encontrar información sobre el uso de esta librería:

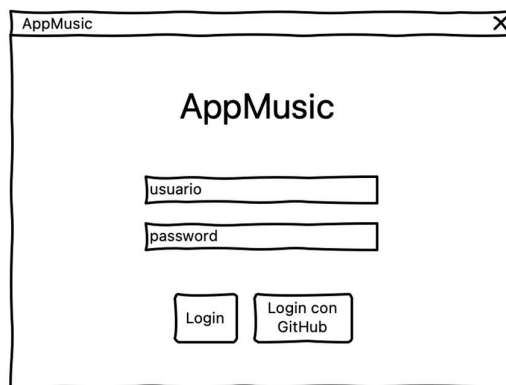
<http://soloinformaticayalgomas.blogspot.com.es/2010/12/generar-un-documento-pdf-desde-java.html>

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 describen las ventanas (en forma de *mockups*) que formarán parte de la aplicación con el objetivo de facilitar su desarrollo. No obstante, las imágenes mostradas son sólo orientativas y **los alumnos tienen libertad para optar por otros diseños, componentes, layouts, etc.**

Ventana Login: Se trata de la ventana de entrada a la aplicación. Si el usuario y clave introducidas son correctas aparecerá la ventana principal (que se comentará más adelante). En caso contrario debe aparecer un mensaje de error. La autenticación del login se puede hacer sobre la contraseña registrada durante el proceso de registro de usuario (ver ventana) o, de forma opcional, ofrecer un proceso de autenticación delegando en GitHub (ver ejemplo de anexo al final del enunciado).



The mockup shows a window titled "AppMusic" with a standard macOS-style title bar (including a close button). The window content is centered and contains the text "AppMusic" in a large, bold font. Below this, there are two text input fields, the first labeled "usuario" and the second labeled "password". At the bottom of the window, there are two buttons: one labeled "Login" and another labeled "Login con GitHub".

La *ventana de Registro* podría tener el siguiente el siguiente aspecto.

AppMusic

usuario password

nombre completo

fecha nacimiento 12/12/2023

Registro

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 “Bienvenido <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 un panel con una lista de botones que permiten acceder a la funcionalidad ofrecida por AppMusic: *buscar canciones*, *gestión de playlist*, *mostrar lista de canciones recientemente escuchadas* y *mostrar mis 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.

AppMusic

Buscar

Gestión Playlists

Recientes

Mis Playlists

Bienvenido, usuario Premium logout

Ventana “Buscar canciones”. Primero aparece un panel con cuatro campos de entrada para los filtros: intérprete, título, estilo y favoritas. Debajo de ellos el botón para realizar la búsqueda. El *estilo* musical debe ser introducido a través de una lista desplegable (*combobox*) y *favoritas* mediante una casilla de marcado (*checkbox*).

AppMusic

Buscar

Gestión Playlists

Recientes

Mis Playlists

Bienvenido, usuario Premium logout

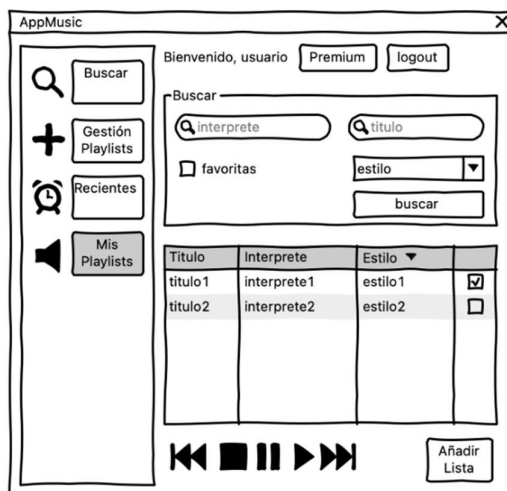
Buscar

interprete titulo

favoritas estilo

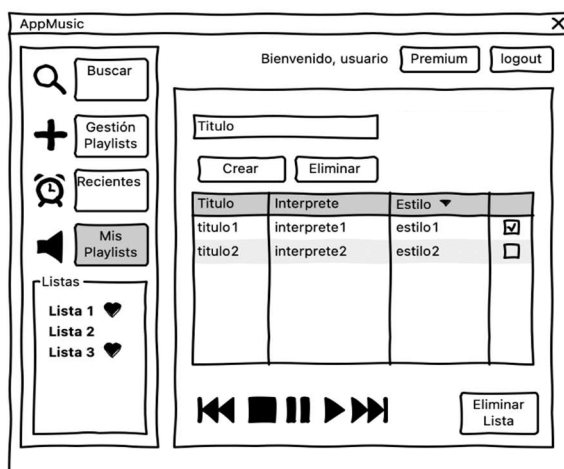
buscar

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. Debajo de la tabla de canciones aparecen los botones para reproducir, detener reproducción, pausar, reproducir siguiente y anterior canción en la lista de resultados. 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”. La casilla de marcado ‘Favoritas’ indica que las canciones de la búsqueda deben estar incluidas en una *playlist*. Una canción también se puede reproducir con un doble-clic sobre la canción seleccionada. La tabla muestra cuatro columnas: título de la canción, nombre del intérprete, estilo, y una casilla de marcado que se usa como se explica abajo.

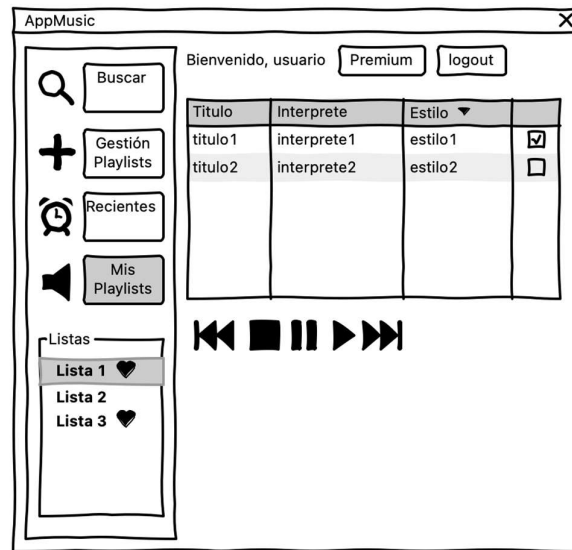


Se pueden añadir canciones a una playlist ya existente marcando en la tabla las canciones a añadir (casilla de marcado de la fila correspondiente) y pulsando el botón ‘Añadir a Lista’ que mostrará una ventana de diálogo para seleccionar la playlist en una lista de playlists (se puede usar un combobox). También se puede introducir el nombre de una lista no existente que se creará y a la que se añadirán las canciones.

Ventana ‘Gestión PlayList’: Permite crear y editar una *playlist*. En la parte superior aparece un campo para introducir el título de la *playlist*. Debajo hay un botón para crearla y otro para eliminarla. La tabla muestra la lista de canciones seleccionadas en la tabla de resultados de la búsqueda (ventana “Buscar Canciones”). Si no se seleccionaran canciones, la tabla mostrará una lista vacía (y la *playlist* se creará entonces vacía). Al hacer clic sobre el botón de “Crear” se abre un dialogo que pide confirmación sobre si se quiere crear o no la *playlist*. Si desde esta ventana se introduce un título de *playlist* ya existente en el sistema, entonces se cargan todos los datos de la *playlist* existente para poder editarla. Desde el botón “Eliminar de la Lista” se pueden eliminar las canciones seleccionadas en la tabla. El botón “Eliminar” borrará la lista cargada del sistema.



Ventana ‘Mis listas’: Al hacer clic sobre este botón se muestra una lista con las *playlists* creadas. Al seleccionar una, se muestra una tabla con las canciones y debajo los botones de reproducción.



Ventana ‘Reciente’: Al pulsar el botón “Reciente” se muestra la tabla con una lista de las últimas canciones reproducidas por el usuario y los botones para su reproducción (misma ventana que “Mis Listas”).

Ventana “Canciones más reproducidas”: Cuando el usuario es ‘Premium’ debajo de “Recientes” aparecerá un nuevo botón “Más Reproducidas” y otro “Generar PDF”. La primera mostrará la lista de canciones más reproducidas en el sistema. La ventana es similar a “Mis Listas” aunque la tabla mostrará ahora: título de canción, interprete, estilo y número de veces reproducida). El segundo botón mostrará un diálogo solicitando la ubicación del archivo PDF a generar en el sistema de archivos.

Opcional: El alumno puede implementar dos funcionalidades adicionales: (i) reproducir todas las canciones de una playlist en orden secuencial o bien activar un modo aleatorio y/o (ii) mostrar un componente *JSlider* que visualice el progreso de la reproducción de una canción. Para añadir estas funcionalidades, se proporcionará información en las FAQs de la asignatura y se puede consultar con el profesor tutor del grupo.

Reproducción de música

JavaFX es una librería de de Java para crear GUIs. Aunque JavaFX puede utilizarse para construir aplicaciones completas, también es posible integrar JavaFX en aplicaciones Swing existentes. Una de las características útiles de JavaFX es su API de medios, que puede ser especialmente interesante para reproducir canciones. En nuestro caso, se reproducirán canciones en formato mp3. Para ello, el alumno podrá utilizar la clase `MediaPlayer` de la librería *JavaFX* de Java 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, en nuestro caso podría tener la forma `<nombre-estilo>/<nombre-canción>` por ejemplo “JAZZ/Nina Simone-Fly Me To The Moon.mp3”. Supondremos que las canciones están una carpeta “canciones” en la carpeta “src/main/resources” creada cuando se usa Maven:

```
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 rutaCancion = cancion.getRutaFichero();
URL resourceURL = getClass().getResource("/canciones" + "/" + rutaCancion);
if (resourceURL == null)
    throw new FileNotFoundException("Fichero canción no encontrado: " + rutaCancion);
Media hit = new Media(resourceURL.toExternalForm());
mediaPlayer = new MediaPlayer(hit);
```

Login de usuarios en GitHub

Se podrá realizar un *login* con una cuenta de Github (nombre y contraseña). A continuación, se ilustra cómo se puede llevar a cabo por medio de un API Java de autenticación que permite comprobar si las credenciales de un usuario de GitHub son válidas. Los alumnos tienen libertad para usar otras APIs.

//dependencia que se debe añadir al POM de Maven

```
<dependency>
    <groupId>org.kohsuke</groupId>
    <artifactId>github-api</artifactId>
    <version>1.315</version>
</dependency>
```

//Código que realiza el login

```
import org.kohsuke.github.GitHub;
import org.kohsuke.github.GitHubBuilder;

GitHub github = new GitHubBuilder().withPassword("mi_usuario", "mi_password").build();
System.out.println("¿Login válido?:" + github.isCredentialValid());
```

Arquitectura de la aplicación

Se organizará la aplicación en una arquitectura de 3 capas (Presentación, Lógica de Negocio y Almacenamiento) que será descrita en un seminario de prácticas (también se discutirán los aspectos básicos en el primer tema de teoría). *Se entregará el proyecto 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**.

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 y que se encargará de cargar lotes de canciones en el sistema. El componente Luz que se verá en las sesiones de laboratorio, 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** en Github. 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 debe 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 **8 de octubre**. Se creará una tarea en el Aula Virtual para que cada grupo pueda subir su diagrama de clases junto a una explicación de los detalles que considere oportuno (**un único envío por grupo**). Los alumnos pueden explorar el uso de ChatGPT como apoyo a la creación del diagrama de clases. En ese caso, deben presentar el diagrama de

clases creado por ellos y el diagrama creado con la ayuda de ChatGPT. El archivo PDF entregado debe incluir los nombres de los alumnos.

2. En los seminarios de prácticas de la semana del 9 de octubre se discutirá el **diagrama de clases proporcionado 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 tutor para seguimiento del trabajo práctico (**última semana de noviembre y primera de diciembre**). Recomendada para aquellos grupos que presentarán el proyecto en la convocatoria de enero.
5. Creación de un componente para cargar canciones a partir de un archivo XML.
6. Se implementarán test unitarios para, al menos, una clase del modelo.

Fecha de entrega: 12 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 como un archivo zip.

Documentación

Constará de las siguientes partes:

1. *Diagrama de clases* del dominio.
2. *Diagrama de colaboración o secuencia* para la operación de añadir playlist (se ejecuta al hacer clic sobre el botón “Aceptar” del panel “Gestión PlayLists”).
3. Una breve explicación de la arquitectura de la aplicación y las 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 (los implementados y también aquellos indirectamente aplicados al ser parte de Swing).
5. Breve explicación sobre los componentes utilizados + componente implementado.
6. Tests unitarios implementados para la ejecución de pruebas.
7. Breve *manual de usuario* que explique cómo usar la aplicación (captura de pantallas y texto informativo).
8. Observaciones finales que los alumnos deseen comentar (deben incluir una estimación del tiempo dedicado al proyecto).

Evaluación

Cada parte se valorará con el siguiente porcentaje:

- Diseño de la interfaz (20%)
- Uso de componentes (10%),
- Aplicación de patrones (10%),
- Código (separación en tres capas, legibilidad, uso de principios básicos de programación OO, implementación de patrones de diseño, uso de *Swing*, comentarios) (40%),
- Uso de *JUnit* (5%),
- Documentación entregada (15%).
- Es obligatorio el uso de *Git* y *Maven*.