

## App de Discos Favoritos

Elaborar una app para almacenar los datos de los discos favoritos del usuario. Leer bien todo el enunciado hasta el final antes de comenzar. Una vez hecho esto, partir del proyecto adjunto al enunciado que ya contiene una **activity** sencilla de Compose.

La aplicación contiene un *datasource* dentro del paquete **data** llamado *DiscosData.kt* el cual contiene una variable *startingDiscos* que es una lista de objetos *Disco*. Este modelo de datos viene ya definido en el archivo *Disco.kt* contenido en el mismo paquete. Este modelo representa además una **Entity** de **Room** para crear una base de datos que almacene los discos de la aplicación.

Cómo se puede observar en el archivo, de cada disco nos interesa almacenar un **id**, el **título** o nombre del disco, el **autor/a/es** del disco, el **número de canciones**, el año de **publicación** y una **valoración** del disco que estará entre 1 a 5 estrellas.

Es posible realizar toda la aplicación sin la parte de la persistencia de datos en Room pero esto tendrá una penalización en la nota final del examen ya que la persistencia de datos forma parte de la rúbrica de evaluación, de la cuál tenéis disponible una versión preliminar y sujeta a cambios al final de este enunciado.

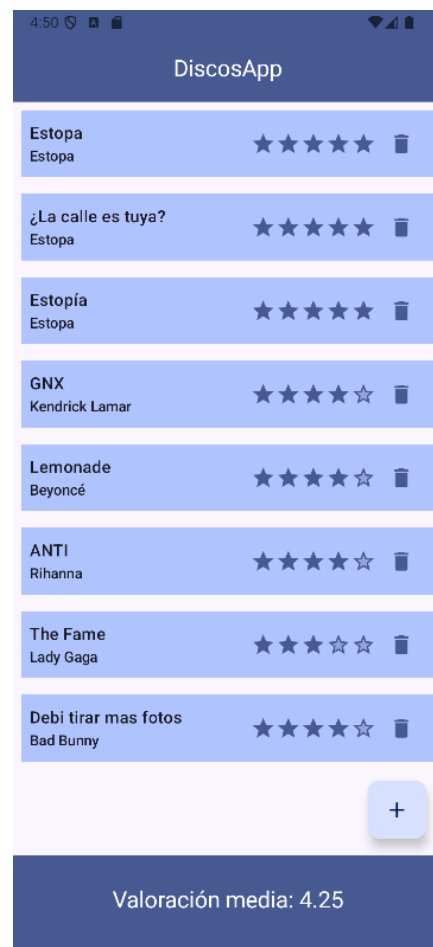
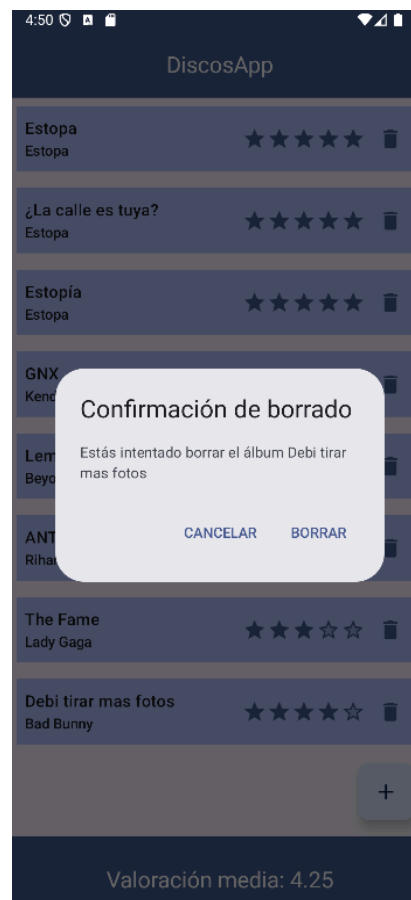


Imagen 1: Pantalla Home

La aplicación debe constar de 3 pantallas:

- Una pantalla de **Home**.
  - En esta pantalla **se mostrarán todos los discos añadidos** con parte de la información de los mismos, en concreto el título y el autor, junto con la valoración (*Imagen 1*).
  - Debe ser posible **al pulsar en cada ítem** navegar hacia la pantalla de detalle, dónde se mostrarán el resto de los datos del disco.
  - Debe existir también un botón de borrado, el cuál tras pulsarlo mostrará un **cuadro de diálogo** pidiendo confirmación de borrado (*Imagen 2*).



*Imagen 2: Diálogo de confirmación de borrado.*

- La pantalla debe contener una barra superior con el título de la aplicación, el cuál debéis cambiar a **“DiscosApp” + vuestro nombre y primer apellido** (*Imagen 1*).
- Debe contener también un ***FloatingActionButton*** que nos lleve a la pantalla de añadir un nuevo disco (*Imagen 1*).
- Debe contener también una **barra inferior** que nos muestre la media de las valoraciones de los discos añadidos (*Imagen 1*) o un **mensaje** de “todavía no hay discos añadidos” en caso de que la lista esté vacía (*Imagen 3*).
- Además, **si la lista está vacía**, la parte central de la aplicación debe mostrar un **mensaje** de que no hay discos añadidos todavía junto con un **icono** y un botón tal y como se muestra en la imagen (*Imagen 3*). Si se pulsa en ese

**botón** deben volver a insertarse los discos por defecto contenidos en el archivo **DiscosData.kt** (*Imagen 4*).

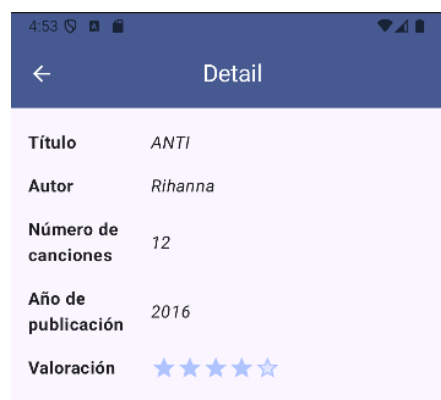


*Imagen 3: Pantalla de Home con la lista vacía.*



*Imagen 4: Pantalla de Home con datos por defecto*

- Una pantalla de **Detalle**.
  - En esta pantalla se deben **mostrar todos los datos del disco seleccionado** previamente en la pantalla de Home tal y cómo se muestran en la imagen (*Imagen 5*).
  - Además debe existir una **barra superior** como la de la imagen con la opción de **volver** a la pantalla anterior a través del **icono** de la parte izquierda de la barra (*Imagen 5*).



*Imagen 5: Pantalla de detalle*

- Una pantalla de **Añadir**.

- En esta pantalla debe existir una **barra superior** tal y como se muestra en la imagen, con el mismo título y el **icono** para **volver** a la pantalla anterior en la parte izquierda de la pantalla (*Imágenes 6 y 7*).
- Además, esta pantalla debe contener un **formulario** para obtener los datos de un nuevo disco y añadirlo a la lista (*Imágenes 6 y 7*). Sin embargo, el botón de **añadir** sólo debe **activarse** cuándo los datos sean correctos en base a las siguientes **condiciones**:
  - El **título** y el **autor** no pueden estar vacíos ni estar en blanco.
  - El **número de canciones** no puede ser menor a 1 ni superior a 99.
  - El **año de publicación** no puede ser anterior al año 1000 ni posterior al año 2030.
  - La **valoración** no puede ser menor a 1 ni superior a 5.
  - El número de canciones, el año de publicación y la valoración sólo pueden **contener dígitos**.



4:48

← Añadir disco

Título

Autor

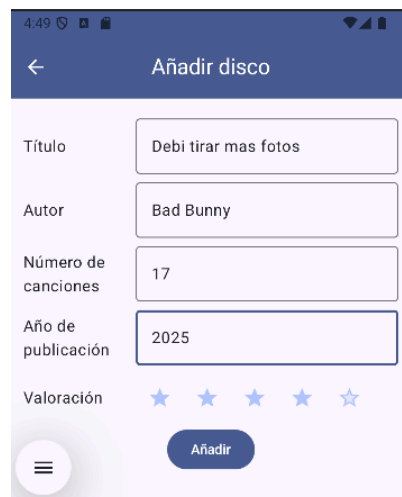
Número de canciones

Año de publicación

Valoración ★ ★ ★ ★ ★

Añadir

*Imagen 6:*  
*Pantalla de Añadir con botón desactivado*



4:49

← Añadir disco

Título

Autor

Número de canciones

Año de publicación

Valoración ★ ★ ★ ★ ★

☰ Añadir

*Imagen 7:*  
*Pantalla de añadir con botón activado*

## Formato de entrega

Debe entregarse un único fichero **zip** con el proyecto de Android Studio comprimido dentro. El nombre del archivo debe coincidir con la siguiente estructura:

**apellido\_nombre\_examen2eval.zip**

Dónde el apellido y el nombre corresponderá con el alumno/a que realiza la entrega.

## A tener en cuenta

- Se recomienda crear un repositorio local para ir añadiendo *commits* según se van completando partes del examen para así poder volver a un *commit* anterior si fuese necesario a la hora de hacer la entrega.
- La aplicación debería de reutilizar el mismo composible para la barra superior en todas las pantallas, pasándole el título cuándo este cambia y un booleano para saber si tiene que mostrar el icono de navegar hacia la pantalla anterior. Este composible debería estar colocado a nivel de arquitectura en el lugar correcto.
- En caso de usar navegación, que es la opción más recomendable para esta aplicación, la aplicación debe gestionar la navegación enviando lambdas desde un componente **NavHost** y no propagando el **NavController** por los diferentes composibles. Además, para navegar a la pantalla de **detalle** es necesario utilizar navegación con parámetros.
- La aplicación debe realizarse utilizando persistencia de datos a través de **Room**, el **proyecto base ya contiene todas las dependencias necesarias** para trabajar con Room integradas, por lo que no es necesario tocar la configuración de ningún archivo Gradle.
- El estado de todas las pantallas debe controlarse de forma aislada, es decir, deberíamos gestionar todos los eventos desde uno o varios **ViewModel** y definir en cada caso el modelo de la **UI**. Es recomendable utilizar un **ViewModelProvider** en el caso de integrar la persistencia de datos para poder realizar el ejercicio siguiendo una arquitectura correcta.
- Todas las String fijas del proyecto deberían almacenarse en los recursos de la aplicación dentro del paquete *res* y, en concreto, dentro del archivo *Strings.xml*.
- Para conseguir un resultado óptimo tratando de imitar el diseño visual de la app se recomienda el uso de los colores y tamaños de letra proporcionados por la clase **MaterialTheme**. También se recomienda el uso de la *extension function* **weight** de la clase *Modifier* para conseguir las proporciones adecuadas en el formulario de añadir y en la pantalla de detalle, así como de las correctas opciones en los *Arrangement* y los *Align* en los lugares correspondientes.

## Rúbrica orientativa de calificación - no definitiva, podría sufrir cambios

Total de puntos: 100

Criterio	Puntos	Descripción del nivel de cumplimiento
<b>Funcionamiento general de la app (40%)</b>		
Funcionamiento principal de la aplicación	15	<b>Alta (10):</b> Todas las funcionalidades descritas en el enunciado funcionan correctamente. <b>Media (5-9):</b> Algunas funcionalidades presentan errores menores. <b>Baja (0-4):</b> Múltiples errores graves.
Gestión del estado	10	<b>Alta (10):</b> El estado está bien gestionado siguiendo la arquitectura MVVM. <b>Media (5-9):</b> Algunos problemas con la gestión del estado. <b>Baja (0-4):</b> Arquitectura no implementada correctamente.
Recomposición eficiente	10	<b>Alta (10):</b> Recomposición controlada y eficiente en todo momento. <b>Media (5-9):</b> Recomposición innecesaria o poco optimizada. <b>Baja (0-4):</b> Problemas graves de recomposición o bloqueos frecuentes.
Inserción de elementos iniciales	5	<b>Alta (10):</b> Los elementos iniciales se agregan correctamente y pueden insertarse tras el vaciado de la lista. <b>Media (5-9):</b> Los elementos iniciales se agregan o se pueden insertar tras el vaciado de la lista pero no ambas. <b>Baja (0-4):</b> Los elementos iniciales no se agregan ni se pueden insertar tras el vaciado.
<b>UI y UX (15%)</b>		
Diseño visual y usabilidad	10	<b>Alta (10):</b> El diseño es idéntico a las capturas y la usabilidad está bien conseguida. <b>Media (5-9):</b> El diseño presenta detalles diferentes o fallos en la usabilidad. <b>Baja (0-4):</b> El diseño no es como el requerido y hay fallos de usabilidad.
Uso correcto de los Composables proporcionados por la API o de los propios.	5	<b>Alta (10):</b> Se hace un uso correcto de los composables de la API y de los propios. <b>Media (5-9):</b> El uso de los composables de la API o de los propios no aprovecha bien la jerarquía. <b>Baja (0-4):</b> No se hace un uso correcto de los composables de la API ni de los propios.

---

**Navegación entre pantallas (15%)**

---

Definición de rutas mediante los métodos correctos	5	<b>Alta (10):</b> La definición de las rutas y el grafo de navegación es correcta y sigue las pautas recomendadas. <b>Media (5-9):</b> La definición de las rutas y el grafo de navegación presenta deficiencias. <b>Baja (0-4):</b> La definición de las rutas y el grafo de navegación presenta fallos o no funciona.
Paso de parámetros	10	<b>Alta (10):</b> El paso de parámetros en la navegación funciona correctamente y se hace de la manera adecuada. <b>Media (5-9):</b> El paso de parámetros en la navegación funciona correctamente pero no se hace de la forma adecuada. <b>Baja (0-4):</b> El paso de parámetros en la navegación no funciona correctamente o presenta fallos graves.

---

**Persistencia de datos (30%)**

---

Creación y consulta de la base de datos e inyección de dependencias de la misma	15	<b>Alta (10):</b> La base de datos se crea y se consulta correctamente y se inyectan las dependencias de forma segura. <b>Media (5-9):</b> La base de datos se crea y se consulta correctamente pero hay errores en la inyección de dependencias. <b>Baja (0-4):</b> Hay errores en la creación y/o consulta de la base de datos y/o en la inyección de dependencias.
Inserción correcta en la base de datos	10	<b>Alta (10):</b> Se insertan los datos de forma correcta en la base de datos. <b>Media (5-9):</b> Existen errores en la inserción de los datos. <b>Baja (0-4):</b> No es posible insertar datos en la base de datos.
Eliminación de datos exitosa	5	<b>Alta (10):</b> La eliminación de tuplas se lleva a cabo con normalidad. <b>Media (5-9):</b> Existen errores en la eliminación de tuplas. <b>Baja (0-4):</b> No es posible eliminar tuplas de la base de datos.

---