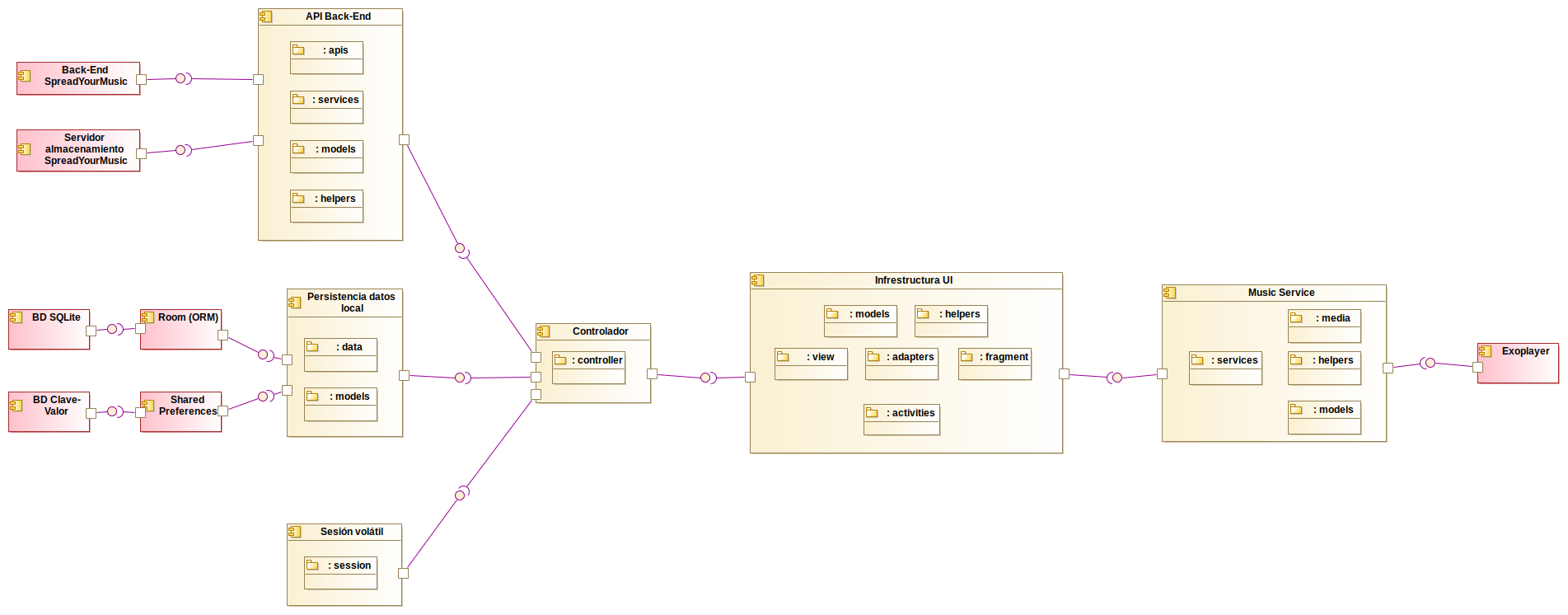
**Arquitectura Android**

La aplicación Android es un cliente ligero, ya que necesita conectividad plena para su correcto funcionamiento, aunque para evitar el consumo excesivo de datos se realiza un cacheo de forma automática de las imágenes que aparecen en ella, en esta línea se permite también al usuario descargar las canciones que desee.

**Diagrama estructural de la aplicación**



La estructura de la aplicación la podríamos ver separada en 4 capas, en las que se sigue un estilo arquitectural modelo-vista-controlador. La capa denominada Infraestructura UI, la cual representa la vista se relaciona con la capa Controlador, la cual representa el controlador y también con la capa Music Service, en esta comunicación el controlador se encuentra incrustado dentro de la capa Infraestructura UI. Por otro lado la capa controlador se relaciona con las capas que ofrecen datos que son API Back-End, Persistencia datos local y Sesión volátil.

**Resumen de funcionalidades de cada módulo:**

El módulo API Back-End se encarga tanto de la comunicación con los servidores de Back-End en los cuales se realizan peticiones mediante una REST-API, así como con los servidores de almacenamiento en los cuales se suben y eliminan tanto las canciones, sus letras así como las imágenes de perfil de usuario o caratulas.

El módulo persistencia datos local posee dos funciones, la primera es almacenar los datos se la sesión en un almacenamiento persistente, de esta forma tras cerrar la aplicación el usuario sigue manteniendo la sesión iniciada en el dispositivo. Para ello se utiliza un componente que proporciona Android denominado Shared Preferences el cual es una base de datos clave-valor. En esta base de datos se almacena tanto el usuario logeado como el token de sesión que permite identificarse frente al Back-End. Por otro lado, esta capa también se encarga de guardar las canciones descargadas en la memoria física del dispositivo. Para ello se utiliza una memoria no volátil asignada a la aplicación y protegida por el sistema frente a lecturas y escrituras de otras aplicaciones en la cual se guardan los archivos de sonido así como las caratulas de los álbumes a los que pertenecen las canciones. Para guardar la información de las canciones como el nombre, autor, el identificador y otros campos se utiliza un mapeador objeto-relacional llamado Room el cual está implementado sobre la base de datos SQLite que poseen las aplicaciones Android.

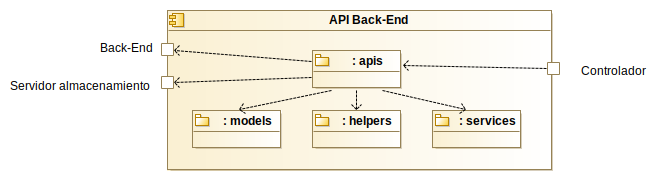
El módulo sesión volátil se encarga de guardar algunos de los datos del usuario para de esta forma evitar tener que hacer llamadas innecesarias al back-end.

El módulo controlador es el nexo entre las capas que ofrecen datos y la capa que controla la interfaz. Las capas que ofrecen datos, todas realizan operaciones muy costosas en tiempo como es comunicarse con otras máquinas o hacer accesos al almacenamiento persistente. Por ello esta capa realiza todas sus operaciones en un hilo de ejecución distinto al principal de forma asíncrona para evitar paralizar el hilo principal que es el que se encarga del control de la pantalla. Para llevar a cabo esto se utiliza el patrón observer, siendo el observador invocado cuando el controlador posee los datos.

El módulo Infraestructura UI es la que se encarga de controlar la vista de la aplicación y la interacción entre todas la pantallas.

Por último el módulo Music Service se encarga de la reproducción de la música así como de la reproducción de las letras de la canción. Esta capa se sustenta sobre la librería Exoplayer la cual ofrece funciones extendidas sobre la librería nativa de audio de Android.

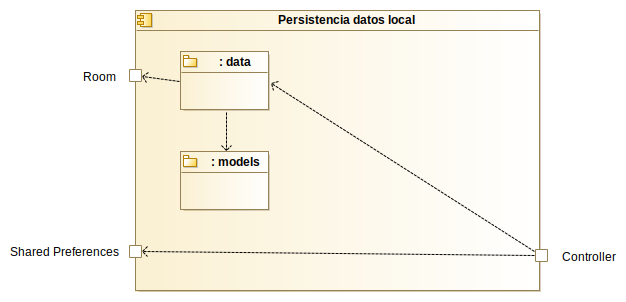
**Módulo: API Back-End:**



El módulo API Back-End recibe peticiones del módulo controlador. Estas peticiones pueden ser de dos tipos. La primera son las que requieren datos, las cuales se resuelven realizando llamadas a la REST-API del Back-End. Para realizar estas llamadas se hace uso del paquete helpers el cual posee unas cuantas utilidades para realizar estas llamadas así como para tratar los datos que estas devuelven. Los resultados de estas llamadas se devuelven como objetos del paquete models, el cual posee el modelo de datos interno de la aplicación.

Por otro lado están las peticiones que requieren subir o eliminar ficheros en los servidores de almacenamiento. Estos servidores utilizan la API de Amazon S3 como forma de comunicación. Para establecer conexiones con esa API, se utiliza una clase auxiliar que se encuentra en el módulo services que se encarga de manejar esta API, debido a que para utilizarla se requiere instanciar objetos que son costosos, encargándose esta clase de hacer esto sólo una vez así como de manejar las conexiones y controlar el estado de las peticiones que se realicen.

**Módulo: Persistencia datos local**

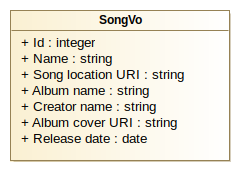
****

El módulo persistencia datos local posee dos funciones, la primera es almacenar los datos se la sesión en un almacenamiento persistente, de esta forma tras cerrar la aplicación el usuario sigue manteniendo la sesión iniciada en el dispositivo.

Para esta función aunque a alto nivel se realiza dentro del módulo de persistencia de datos, las clases que la implementan se encuentran dentro del paquete controlador. Estas clases controlan el componente denominado Shared Preferences de Android el cual es una base de datos clave-valor. En ella en el caso de que esté el usuario logueado se almacena tanto el nick de este y el token de sesión que le permite identificarse frente al Back-End.

Por otro lado, para almacenar la información de las canciones descargadas, las que también se gestionan desde este módulo, se utiliza un mapeador objeto-relacional sobre la base de datos SQLite interna de la aplicación llamado Room.

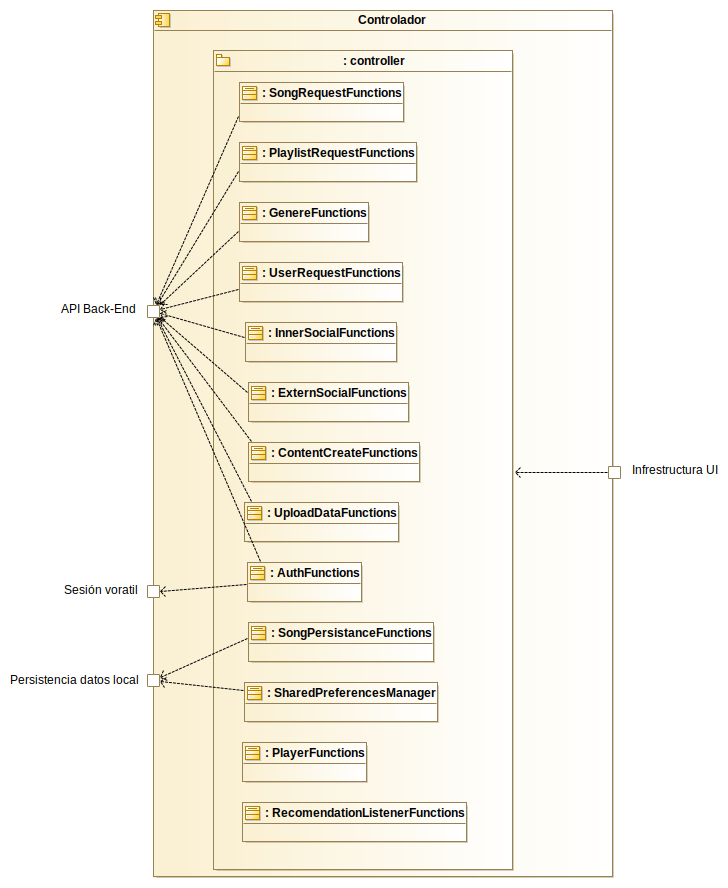
En él se creó una entidad en la cual se almacena toda la información relativa a las canciones y el objeto que la represa es el siguiente:



Además de estos campos de las canciones, se necesita almacenar los ficheros de audio, así como las caratulas de los álbumes a los que pertenecen.

Para ello se hace uso de una memoria no volátil asignada a la aplicación y protegida por el sistema frente a lecturas y escrituras de otras aplicaciones que posee Android denominada Almacenamiento Interno. La diferencia entre guardar las canciones en este almacenamiento y en el almacenamiento externo es que aquí nos aseguramos que los datos de las canciones sólo pueden ser accedidos desde nuestra aplicación forzando al usuario a utilizarla para poder escuchar las canciones descargadas.

**Módulo: Controlador**



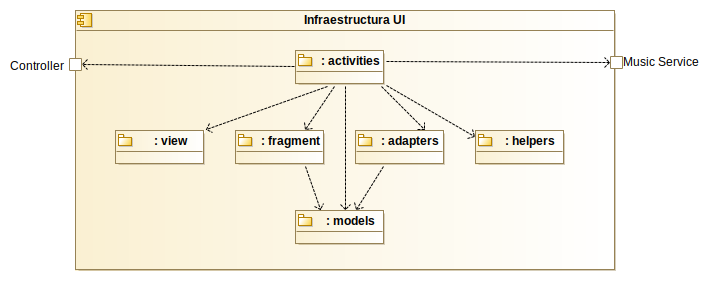
El módulo controlador es el nexo entre las capas que ofrecen datos y la capa que controla la interfaz.

Las capas que ofrecen datos realizan operaciones muy costosas en tiempo debido a que en unos casos realizan peticiones a otras máquinas a través de la red y en otros acceden al almacenamiento persistente del dispositivo. Para gestionar esta situación este módulo realiza todas sus operaciones en un hilo de ejecución distinto al principal de forma asíncrona para evitar ralentizar o incluso paralizar el hilo de ejecución principal que es el que se encarga del control y dibujado de las pantallas. La ralentización o paralización de este hilo produce una mala experiencia de usuario ya que este ve como las pantallas tardan mucho en aparecer y como el teléfono durante unos instantes no responde, además en Android si se mantiene este hilo ocupado durante unos cuantos segundos (actualmente 5), el sistema toma el control de la aplicación mostrando un dialogo en el cual avisa de que la aplicación no es capaz de responder.

Para llevar a cabo esto se utiliza el patrón observer, siendo el observador invocado cuando el controlador posee los datos.

Debido a que para programar la aplicación se ha utilizado el lenguaje Kotlin el cual posee características de programación funcional, el observador que se pasa es una función que será ejecutada cuando se posean los datos. Como esta función es pasada desde la Infraestructura de la UI, y normalmente con estos datos se modificará parte de la interfaz de usuario, aunque la obtención de datos se realiza en un hilo aparte, se fuerza a que esta función se ejecute siempre en el hilo principal de la aplicación que es desde el único que se puede modificar la interfaz de usuario.

**Módulo: Infraestructura UI**

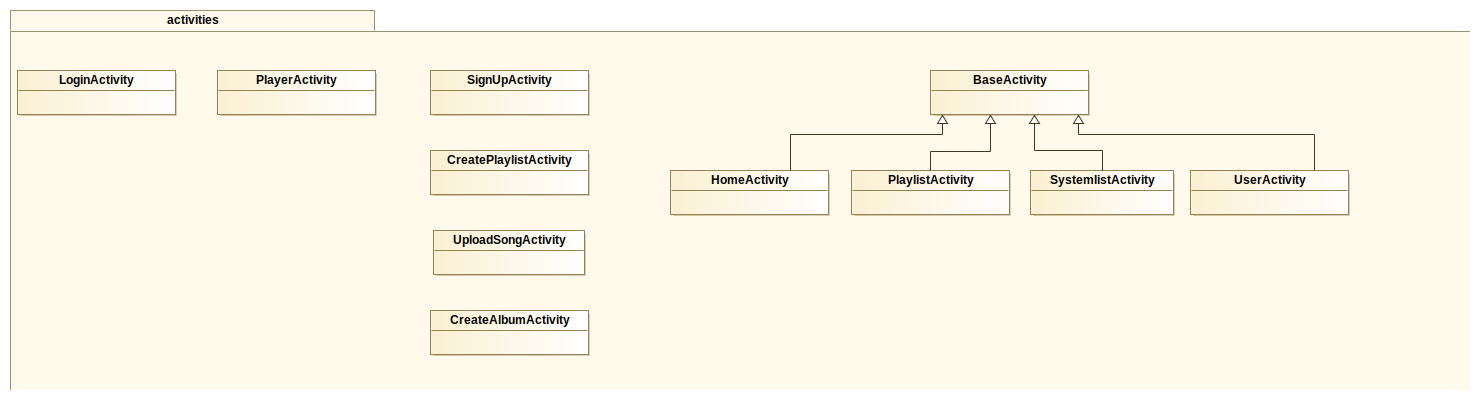


El módulo infraestructura UI es el encargado de controlar la interfaz de usuario y el dibujado de las pantallas. Para ello se utilizan varios paquetes, siendo el principal el paquete activities, en el que cada clase representa al menos una pantalla de Android.

El paquete fragments contiene elementos que permiten dibujar trozos de pantallas de Android. Aquí se han incluido elementos recurrentes en las pantallas.

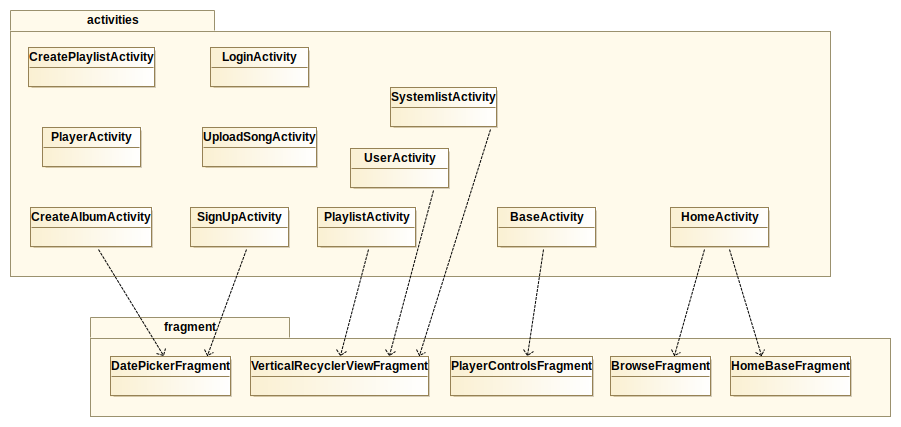
El paquete adapters contiene elementos que permiten dibujar las listas.

Por último el paquete view contiene los elementos que permiten dibujar la onda de sonido circular y la barra de búsqueda circular.

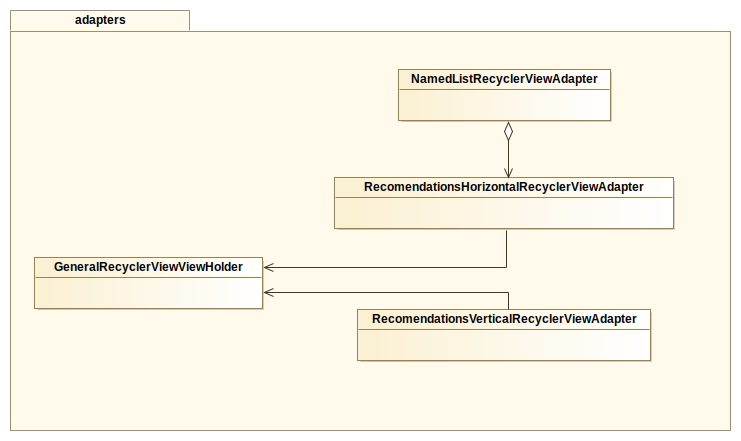
Las activities las podríamos diferenciar en dos clases, desde las que se puede reproducir música y desde las que no. Desde las que se puede reproducir música todas extienden de BaseActivity salvo PlayerActivity, que es la encargada de controlar el reproductor en pantalla completa.

Esto es debido a que en BaseActivity se encuentra incrustado el controlador que permite realizar las comunicaciones con Music Service, de esta forma si se desea reproducir música en una nueva pantalla, solamente ha de extender de BaseActivity y añadir unas pequeñas modificaciones al definir el xml desde el que se dibuja su pantalla.

En el caso de PlayerActivity, el controlador se incrusta dentro de la propia actividad debido a que las comunicaciones que realiza con Music Service son más avanzadas que el resto de pantallas.

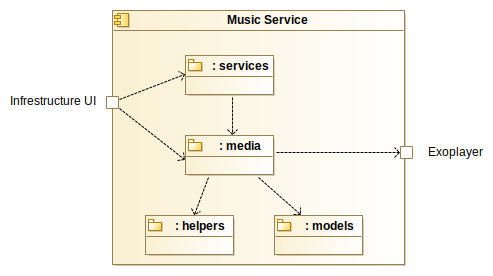


Para simplificar el desarrollo de la interfaz se ideo una estructura de fragments, para construir cada activity. Un fragment permite dibujar un trozo de una pantalla, lo que posibilita en elementos recurrentes en distintas pantallas crear un objeto que pueda ser reusable. En esta arquitectura el caso que mejor ejemplifica esto es el de HomeBaseFragment. Con ese mismo fragment se consigue crear la pantalla principal, la pantalla de novedades, la pantalla de tendencias y la pantalla de canciones más populares de cada género, siendo creada cada pantalla a partir de la instanciación del objeto y una llamada al módulo controller.

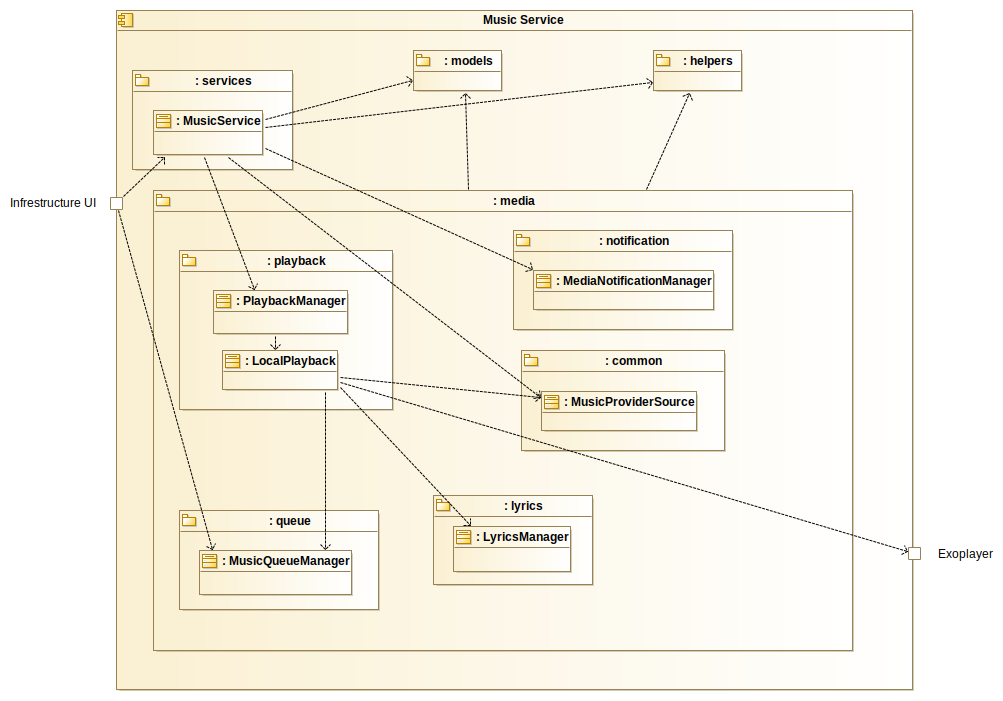


Otro elemento que también permite ser reusados con los adaptadores para listas, debido a que todas las listas se muestran con el mismo estilo, solamente varía en el caso de que sea lista horizontal o lista verical.

**Modulo: Music Service**



El módulo Music Service se encarga de la reproducción de música en el dispositivo, para ello hace uso de la biblioteca Exoplayer en su versión 2.7, además también aporta el mecanismo que permite mostrar las letras de forma dinámica.



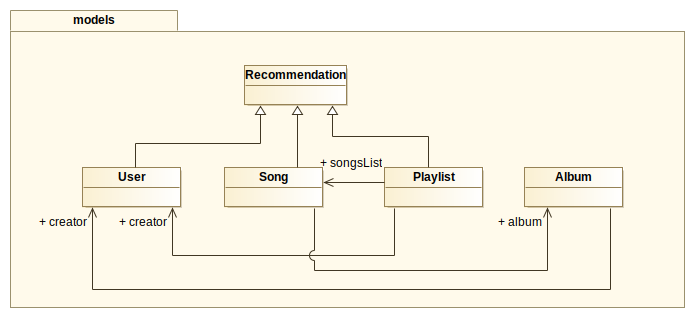
Este módulo posee dos interfaces, la primera es la que se accede mediante el paquete queue. Esta se utilizará para añadir o eliminar canciones en la lista de reproducción en curso y para activar o desactivar la reproducción aleatoria, con esta interfaz se interactúa de forma síncrona.

La otra interfaz se encuentra en el paquete services con la cual se interactúa de forma asíncrona mediante un modelo cliente-servidor. Esta otra interfaz es la que permite obtener las letras de forma dinámica y usar los controles de manejo sobre las canciones como son reanudar canción , parar canción, siguiente canción, canción anterior y avanzar en una canción hasta un momento exacto.

Esta interfaz internamente se comunica con el paquete playback que es el que sirve de interfaz para realizar el manejo del sonido y las letras, siendo esta la que se comunica con Exoplayer.

Por otro lado, desde este mismo módulo también se controla la creación y gestión de la notificación que aparece cuando se está reproduciendo música con la aplicación en segundo plano mediante el paquete notification.

**Modelo de datos aplicación:**



El modelo de datos interno de la aplicación está compuesto por 4 clases que representan cada uno de los elementos básicos sobre los que está asentada la aplicación que son Usuarios, Canciones, Playlist y Albums. Los 3 primeros por cuestiones de implementación extienden de la clase Recommendation, esto permite que muchos elementos como son los adaptadores de las listas se puedan crear de forma genérica y no haga falta crear un adaptador para cada tipo de dato, solamente dentro del adaptador en este caso se comprobaría a que tipo de dato hay que adaptar la lista, esto complica un poco la lógica de estos elementos pero reduce drásticamente el número clases, así mejorando el mantenimiento del código.

**Diagrama paquetes aplicación:**

****

**Estructura vista aplicación:**

Además del módulo Infraestructura UI, la vista de la aplicación se genera a partir de unos ficheros xml donde se definen los elementos que aparecen en cada pantalla. Estos ficheros han sido desarrollados siguiendo las guías de diseño establecidas por material design, además de haber sido creados de forma responsive para que puedan adaptarse a cualquier tamaño de pantalla. En ella las imágenes e iconos utilizados han sido los proporcionados por Google, los cuales cumplen este estilo de diseño.