**Capas de la Aplicación LastFm**



**Persistencia - Modelo**: Encargada de almacenar la información obtenida del servicio consumido, o las preferencias del usuario.

* **com.nitram940.valid.martinprueba.model.database.ArtistsDatabase**
* Implementa una base de Datos en SQLITE que guardara la información consumida desde el webservice y la mantendrá actualizada cada vez que detecte que la información consumida es diferente a la almacenada.
* Sirve de persistencia en el caso que la aplicación este en modo offline.
* Apoyada por:
* **com.nitram940.valid.martinprueba.model.helper.Constants**
* Tiene todas las constantes usadas en la aplicación, desde la creación de las tablas, las consuntas a usar, la actualización de la bd.
* **com.nitram940.valid.martinprueba.model.model\_json.Artist**
* Tiene el modelo de datos de la Tabla Artist – “Los Artistas ”
* **com.nitram940.valid.martinprueba.model.model\_json.ArtistResponse**
* Tiene la respuesta del Json top Artist que contiene el listado e los artistas.
* **com.nitram940.valid.martinprueba.model.model\_json.AttributosModel**
* Contiene informacion relevante a la pagina actual, la cantidad de resultados, la cantidad de Paginas.

**Ui - Vista**: Encargada de generar las vistas y la interacción con el usuario, mediante los Xml Layouts los cuales son los recursos que van a ser usados por la Ui.

* **com.nitram940.valid.martinprueba.ui.FragmentBaseLastFm**
* Fragment Base el cual tiene la interaccion con la capa de Persistencia, Presentador
* **com.nitram940.valid.martinprueba.ui.FragmentLastFm**
* Fragment instanciado
* **com.nitram940.valid.martinprueba.ui.MainActivity**
* Actividad principal, encargada de inicializar las vistas e invocar los fragments.

**Presentador**: Clases encargadas de llenar con datos las vistas generadas por la Ui y tener la lógica para comunicar Las vistas con el Modelo.

* **com.nitram940.valid.martinprueba.model.callback.LastFmApis**
* Interface Contiene la referencia a los servicios que se van a consumir.
* **com.nitram940.valid.martinprueba.model.callback.LastFmApisImpl**
* Clase Ejecuta el proceso de consumir los servicios referenciados en **LastFmApis,** con la ayuda de **ApiRetrofit.**
* **com.nitram940.valid.martinprueba.model.callback.LastFmFetchListener**
* Interface Encargado de interactuar con las vistas para comunicarles que los datos han sido procesados para Inflarlos posteriormente en los RecyclerView.
* **com.nitram940.valid.martinprueba.model.callback.LastFmService**
* Interface intermediario entre **LastFmApisImpl** y **LastFmApis.**
* **com.nitram940.valid.martinprueba.controller.adapter.ArtistsAdapter**
* Clase que media como adaptador para para inflar los Reciclerview que mostraran las películas.
* **com.nitram940.valid.martinprueba.controller.adapter.inflate\_items\_Artist**
* Clase Encargada de inflar con Datos de los Artistas las vistas con la información proporcionada por la base de datos o la misma respuesta del servicio consumido**.**
* **com.nitram940.valid.martinprueba.controller.adapter.animViews**
* Clase encargada de animar la aparición de cada ítem dentro de un ReciclerView
* **com.nitram940.valid.martinprueba.controller.ApiRetrofit**
* Case Base usada como intermediara entre la acción de consumir un WebService y procesar la respuesta del mismo, esto mediante la ayuda de la librería Gson la cual adaptara la respuesta de Json a los modelo de datos. **Cast, CreditsResponse, Genre, GenreResponse, Artist, ArtistsResponse.**
* **com.nitram940.valid.martinprueba.model.helper.NetworkCheckValidation**
* Interface escargada de avisar a la actividad que la implemente que si hay o no conexión y navegación a internet para realizar algún proceso que lo requiera como consumir el webservice de las películas o en el caso que no halla conexión a internet solicitar la información desde la base de datos SQLITE.
* **com.nitram940.valid.martinprueba.model.helper.Utils**
* Contiene las funciones de apoyo a la aplicación.
* La lógica para validar si se tiene conexión a internet y navegación.
* Preferencia del usuario en el caso que aunque se tenga conexión a internet prefiera ver la información que tiene actualmente almacenada en el dispositivo, aplica para los datos de persistencia de sqlite y las imágenes de cache de Picasso.
* **com.nitram940.valid.martinprueba.model.helper.picture**
* Clase encargada de manejar las imágenes mediante la librería Picasso.
* Se encarga de :
* Cache, Descarga, almacenamiento, visualización y transformación.
* Según la información ingresada cambiara el tamaño de la imagen y:
* Intentara cargarla primero desde cache.
* Luego intentara Cargarla desde el Dispositivo
* Si no se obtiene la imagen la intentara descargar desde internet.
* Si no se puede descargar mostrara una imagen de error.
* **com.nitram940.valid.martinprueba.model.helper.BusProvider**
* Clase basada en eventos que sirve como base para informar a las actividades o los fragmentos algún evento publicado en el Bus, actualmente se usa para **searchSubscribed** Entre la actividad principal y el fragmento de búsqueda de la película, envía el texto ingresado en la barra de búsqueda como un evento que recibe el fragmento para luego invocar el proceso de búsqueda sea desde el webservice o la base de datos.
* **com.nitram940.valid.martinprueba.model.callback.searchSubscribed**
* Clase que sirve como evento para el **BusProvider**.
* **com.nitram940.valid.martinprueba.model.helper.LocaleChangedReceiver**
* Encargada de Detectar el Cambio de Idioma en el dispositivo y teniendo en cuenta esto cambiara las preferencias de idioma en el momento de consumir el servicio web.

**Inyeccion de Depencencias.**

Permite mejorar la arquitectura de la aplicación al intentar separar cada una de las capas y tenerlas independientes en tresi.

* **com.nitram940.valid.martinprueba.di.AppComponent**
* Interface encargada de conocer los @Modulos que se instanciaran para luego ser inyectados en la Clases o Actividades que los invoquen @Inyect.
* Sirve como referencia para que el compilador de **dagger** genere el código necesario para realizar la inyeccion
* **com.nitram940.valid.martinprueba.di.AppModule**
* Instancia el contexto de la aplicación el cual será inyectado en las Clases @Modulo que lo invoquen @Inyect
* **com.nitram940.valid.martinprueba.di.DaggerApplication**
* Usada desde el momento que inicia la aplicación para controlar los @Modulos que serán Inyectados a las Clases que lo soliciten.
* **com.nitram940.valid.martinprueba.di.ArtistModule**
* sirve como Proveedor de la interface usada en el proceso de Consumir los servicos de la clase ArtistApisImpl