# Movie database: Android Kotlin app

The app has been built using the Kotlin language in Android Studio. The app is made up by two different activities, where the first activity is the MainActivity and the other is the MovieDetails activity.

The MainActivity is responsible for starting up the app and populate the database if the database is empty. The screen of the MainActivity is made up by a search bar and button, and a list of the movies in the database as seen in Figure 1, which has been implemented using a RecycleView.

Figure 1, the layout of the mainactivity

The RecycleView is a way of creating dynamic lists in Kotlin, where the element in the list is reusing the same definition of an element (Layout), and only creates the elements in the list, when they are needed. If an element is in the bottom of the list, it will only be created when it is about to show on the screen, which greatly increases the performance of the app. In the app, the elements have different TextViews, and an Image. The TextViews holds information about the movie such as the ID of the movie in the database, but these information lays behind the picture. This makes the elements in the list only seem to be a picture, as in Figure 1, but it has the information needed to load the moviedetails, when a user clicks on an image.

When the user clicks on a movie in the list, or when the user searches for a movie id, the MovieDetails activity is getting activated. This activity is responsible for showing all the information of the given movie, such as the title the director etc. This activity can be seen in Figure 2. The MovieDetails could have been implemented as a fragment instead of an activity, but I felt that the experience felt better when it was a separate activity.

The database in the app is a Room database. Room is a local database in the app, which uses SQLite. The Room database is made up from 3 different Kotlin classes. In the app I have implemented these three classes in the package called “persistence”. Room needs a DAO class, which handles the CRUD operations of an entity, which in the app is the MovieDAO class. The entity is being described in a data class called “Movie”, and the whole database creation happens in the MoviePersistence class. In the app only 1 entity is needed, which is the movie which is going to be stored in the database. The movie entity has the following fields: id, title, release\_year, director, cast and image\_reference. The cast field in the entity is being saved as an ArrayList with strings. Room does not know how to convert an ArrayList to an JSON object, so another class called CastTypeConverter has been implemented. This class handles the conversion from ArrayList to JSON, and the JSON to ArrayList. To make Room use this Typeconverter I needed to add it in the MoviePersistence class with the annotation @TypeConverters(CastTypeConverter∷class).

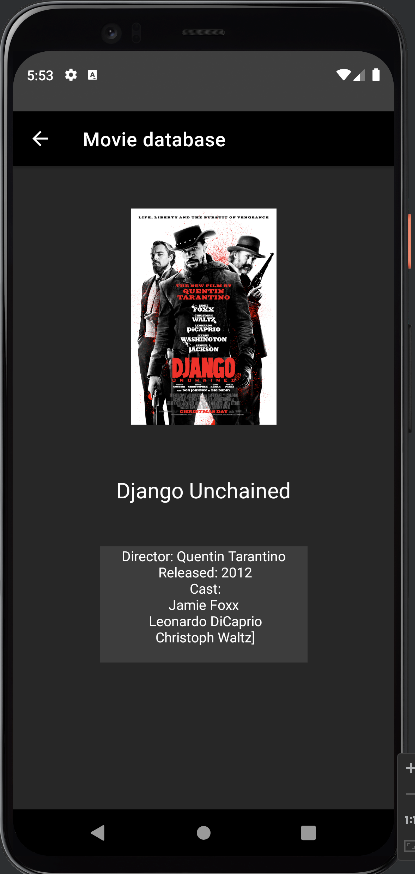


Figure moviedetails activity