TheMovieDb App

SOFTWARE DESIGN DOCUMENT

Nilesh Deokar

Technologies Used :

- Native android development using Kotlin / JAVA

Design Pattern :

- Android MVC

Libraries Used :

- Mosby (MVP library) :
  To keep the per MVP modul short and to eliminate the extra boilerplate code required to write onConfiguration change of Activities.

- Dagger 2.0
  To support the Dependency Injection. Which eliminates the need of creating same objects again and again.

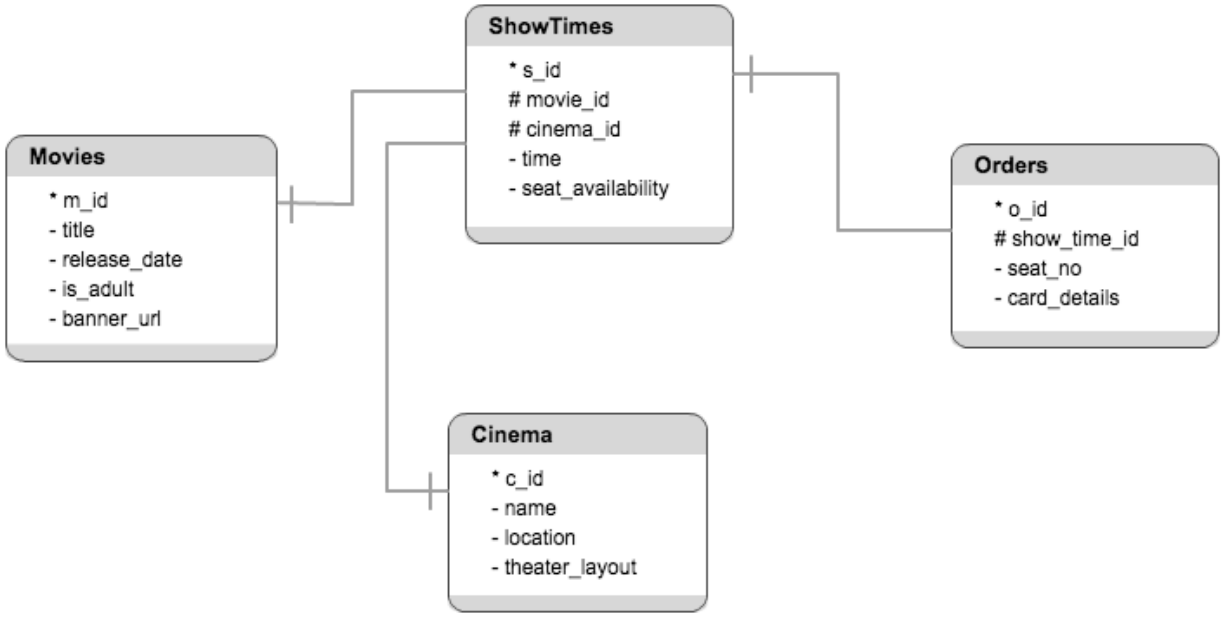
- Retrofit
  For handling the network requests like API calls.

- Room database
  To store the data persistently.

- RxAndroid
  To simply the asynchronous access pattern of the Room db. We have to separate Disk and Network request from the UI. which in term improves the app performance.

- Picasso
  An Image Loading library used for loading images from the network. Also improves the app memory management and speed by separating the bitmaps access inside app.

Database Design :



onCreate of Database :

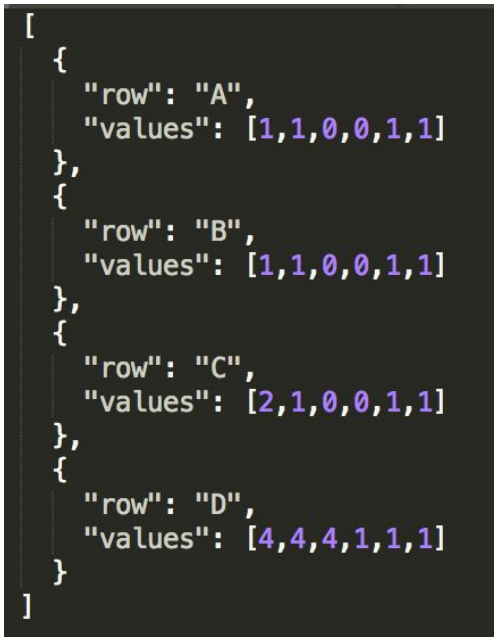
Since we are provided with the only data which belongs to the **Movie**. We need to add some data to database before starting use of it. As mentioned in above diagram.
As we need to show the **cinema, location** to the user and then he will select the seats based on his convenient show time, we have to put some data regarding the current show times.

- **Every Cinema hall has :**
  - Name
  - Location
  - Layout of a theater (seat arrangement)

  Using this all properties we have added **dummy cinema halls** into db at the time of app initialisation.

- **Layout of a theater : (Seat arrangement)**

Let’s consider the Seat layout as 2D array. Each row has some seats and some vacant places. No consider following JSONArray :



```
[
  {
    "row": "A",
    "values": [1,1,0,0,1,1]
  },
  {
    "row": "B",
    "values": [1,1,0,0,1,1]
  },
  {
    "row": "C",
    "values": [2,1,0,0,1,1]
  },
  {
    "row": "D",
    "values": [4,4,4,1,1,1]
  }
]
```

Here each **object** represents one **row** of seats in theater.

0 = Vacant place 1 = Seat available

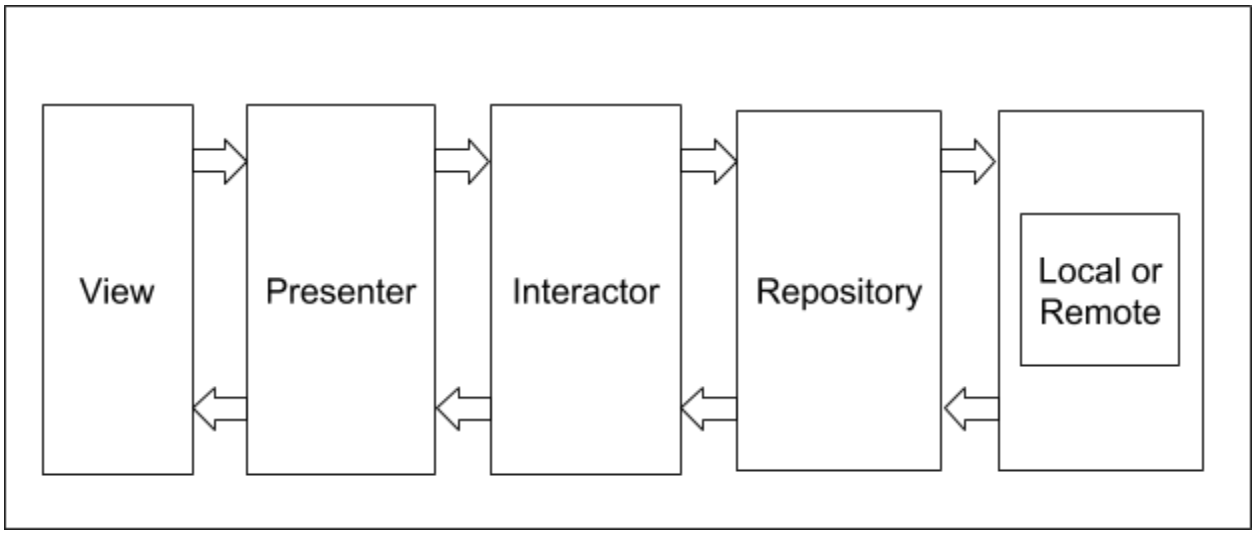
2 = Seat unavailable 4 = Seat booked

Using above data into tableLayout we render the seats.

- **Every Movie show has :**
  - A movie
  - A cinema hall
  - Time of movie play
  - Seats Availability

With above mentioned properties we have created **ShowTimes** table and added dummy data into this table after receiving the **moviesList** from TheMovie’sDb API and the cinemas hall data which we have inserted at the start of an app.

- **Components Interaction :**


View
Presenter
Interactor
Repository
Local or
Remote