Grability Android developer challenge.

Capas de la aplicación

Pattern: MVVM

View:

Responsable de definir la estructura, el diseño y la apariencia de lo que el usuario ve en la pantalla en Android se consigue mediante un conjunto layouts en xml normalmente. Es importante mencionar que una vista puede tener su propio ViewModel.

- ../ui/base
  - BaseActivity.java
    - Clase padre para el manejo de funciones comunes de la aplicación.
- ../ui/content
  - MoviesContentActivity.java
    - Clase encargada de mostrar el contenido/listado de las películas según su categoría/filtro.
- ../ui/details
  - MovieDetailsActivity.java
    - Clase encargada de mostrar los detalles del ítem(pelicula) seleccionada.

Model:

Es el encargado de encapsular y mantener la lógica de negocio y validación de nuestra aplicación.

- ../data/model/details
  - Genre.java
  - MovieDetails.java
  - ProductionCompany.java
  - ProductionCountry.java
  - SpokenLanguage.java
- ../data/model/movies
  - MovieObject.java
  - Result.java

ViewModel:

Es el encargado de actuar como un enlace entre la vista y el modelo pero también se encarga de mantener la lógica de presentación (Tomará las decisiones de cómo se tienen que pintar nuestros datos en nuestra UI) y el estado de la vista para ser capaz de notificar y estar en contacto con todos los elementos o propiedades de nuestras vistas que van a ser reactivas.

- ../ui/content
    - MoviesContentViewModel.java
        - Clase encargada de mantener la lógica de presentación, realiza el llamado y consulta a la base de datos (repositorio) para obtener el listado de peliculas
- ../ui/details
    - MovieDetailsViewModel.java
        - Clase encargada de mantener la lógica de presentación, realiza el llamado y consulta a la base de datos (repositorio) para obtener el detalles de la película seleccionada

Pattern: Repository Pattern

- ../data/model/repository
    - MoviesRepository.java
        - Encargado de la persistencia de los datos (peliculas), si los datos requeridos no se encuentran en la base de datos se hace el llamado al servicio WEB y se persisten una vez el response sea OK.
    - MoviesDetailsRepository.java
        - Encargado de la persistencia de los datos (peliculas), si los datos requeridos no se encuentran en la base de datos se hace el llamado al servicio WEB y se persisten una vez el response sea OK

Packages:

- **Remote**
    - *ApiConstants.java*
        - *Constantes de la aplicación.*
    - *DataManager.java*
        - *Clase encargada de la comunicación con la clase MoviesApiService.java, su función es retornar un Observable<Object> con los datos requridos según la solicitud.*
    - *MoviesApiService.java*

        - *Interface necesaria para la interacción de comunicación con los servicios WEB usando Retrofit.*
    - **Database**
        - *AppDatabase.java*
            - Clase abstracta encargada de la comunicación SQLite-ROOM con las clases DAO(data Access object).
        - *DataBaseCreator.java*
            - *C*lase encargada de crear la instancia de la base de datos.
        - *DataBaseQueries.java*
            - *Queries de comunicación con la base de datos.*
        - *DetailsDao.java*
            - Interface encargada de la comunicación directa con SQLite – ROOM para la persistencia de los detalles de las películas
        - MoviesDao.java
            - Interface encargada de la comunicación directa con SQLite – ROOM para la persistencia del listado de las películas
    - **Factory**
        - *Factory.java*
            - Clase encargada de crear la instancia de comunicación con el servicio WEB usando Retrofit + GSON.
- **Injection**
    - **Component**
        - *ActivityComponent.java*
            - Clase para el manejo de inyección de dependencias – a nivel Activities haciendo referencia anotación @component y sus respectivos módulos.
        - ApplicationComponent.java
            - Clase para el manejo de inyección de dependencias – a nivel Application haciendo referencia anotación @component y sus respectivos módulos.
    - **Module**
        - *ActivityModule.java*
            - Clase tipo módulo a nivel Activity, encargada de retornar el activityContext y proveer las instancias de los adapters usandos dentro de las actividades.
        - *ApplicationModule.java*
            - Clase tipo módulo a nivel Application, encargada de retornar el applicationContext y proveer las instancias de los adapters usandos dentro de las actividades
        - *NetworkModule.java*

        - *Clase tipo módulo encargada de la inyección de dependencias usadas para la comunicación con los servicios WEB.*
    - **Qualifiers**
        - *ActivityContext.java*
            - *Clase tipo @Qualifier a nivel Activity, encargada de evitar la implementación de la misma clase para no generar un exception en el arranque.*
        - *ApplicationContext.java*
            - *Clase tipo @Qualifier a nivel Application, encargada de evitar la implementación de la misma clase para no generar un exception en el arranque.*
            -
    - **Scope**
        - *PerActivity.java*
            - Controla las instancias de la clase inyectada.
- **Application**
    - *ApplicationController.java*
        - *Clase a nivel Application encargada de setear la fuente usada en la aplicación.*
        - *Crear las instancias de los módulos usados a nivel application.*