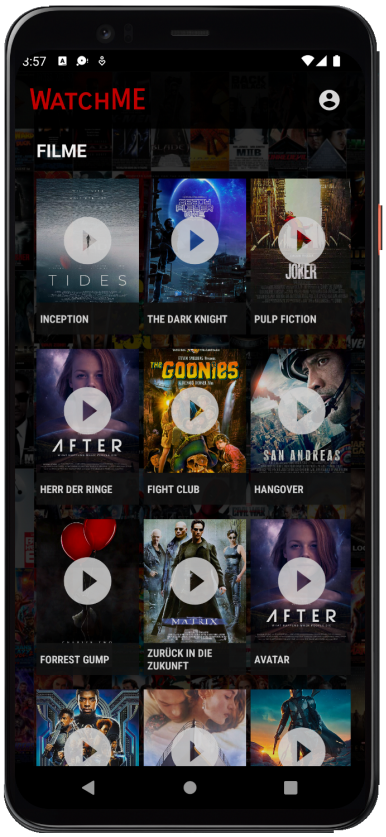
Kotlin 2.7 - RecyclerView II

Aufgaben

Hinweis: Zu bearbeiten ist Aufgabe 1. Aufgabe 2 ist eine Bonusaufgabe

1. Filme Streaming Service - RecyclerView

In dieser Aufgabe richten wir für die uns bekannte App “WatchMe” eine weitere effiziente `RecyclerView` mit einem GridLayout ein. Du kannst dich bei der Bearbeitung der Aufgabe an den Aufgaben aus 2.6 orientieren.



- Öffne das Projekt “Filme Streaming Service”, die `RecyclerView` befindet sich bereits im Layout `activity_main.xml`
- Zuerst wollen wir das Layout stylen, das dann letztendlich einen Listeneintrag darstellt
    - Erstelle eine neue “Layout Resource File” im “layout” Ordner und nenne sie `list_item.xml`, verwende anstatt des Constraint Layouts eine `MaterialCardView` als Root Element

    - Passe die folgenden Attribute (Eigenschaften) der `MaterialCardView` an, bis der Listeneintrag deinen Vorstellungen entspricht
        - `layout_width`: Breite der View
        - `layout_height`: Höhe der View
        - `cardCornerRadius`: Abrundung der Ecken
        - `cardElevation`: Erhebung der View
        - `layout_margin`: Abstand zu den Nachbarelementen
        - `cardBackgroundColor`: die Transparenz und die Hintergrundfarbe
    - Füge innerhalb der `MaterialCardView` ein `ConstraintLayout` ein, welches die gleiche Größe wie die `MaterialCardView` besitzt
    - Platziere innerhalb des Constraint Layouts drei Elemente und passe ihre Attribute nach Belieben an:
        - Eine ImageView, diese enthält später das FilmPoster Bild
        - Eine TextView unter der ImageView, diese enthält später den Filmtitel
        - Einen ImageButton, der über der ImageView liegt und das Icon `ic_play` aus dem Resource Manager anzeigt

      Hinweis: Du kannst dir dein `list_item` bereits in der `RecyclerView` anzeigen lassen, indem du in der `activity_main.xml` der `RecyclerView` das Attribut `tools:listitem="@layout/list_item"` gibst
- Die Klasse `Movie` enthält alle Informationen, die ein Listeneintrag für den Film braucht.
    - Öffne die Datei `data/model/Movie` und ändere den Typ der Klasse von `class` zu `data class`
    - Füge der Data Klasse einen Konstruktor hinzu, der zwei Parameter vom Typ `Int` für die String- und Bild-Ressource enthält.
- Die Klasse `Datasource` holt alle Ressourcen aus der Quelle und bereitet sie für die weitere Verwendung vor
    - Programmiere die Klasse `loadMovies`
    - Für jeden Titel in der Ressource soll ein `Movie` Objekt in einer Liste zurückgeliefert werden

      Hinweis: nutze die Methode `getTitle` und `getImage`
- Die Klasse `ItemAdapter` ist dafür verantwortlich, die Listeneinträge zu erstellen und ihnen die richtigen Informationen zuzuweisen.
  Die innere Klasse `ItemViewHolder` erstellt ein `ViewHolder` Objekt, welches die `ItemView` umschließt und einen Listeneintrag darstellt.

    - erstelle in der Klasse `ItemViewHolder` zwei Variablen für die TextView und ImageView und weise die Views aus dem Laoyut `list_item` zu

        - nutze dafür die übergebene `itemView`
            - über View Elemente ist es möglich die Funktion findViewById (**id**) aufzurufen
            - setze die **id** des TextView und ImageView aus dem `list_item` Layout

        Hinweis: Gib der TextView und ImageView in deinem `list_item` Layout jeweils eine ID, mit der du die Views hier finden kannst

        Hinweis: id kann man über die **R** Klasse ansprechen

- Die Funktion `onCreateViewHolder` erstellt neue `ViewHolder` Objekte, welche die aus dem list_item Layout erstellten `ItemViews` enthält
    - Nutze den `LayoutInflater`, um das `list_item` Layout zu einer `ItemView` “aufzublasen” (siehe Folie 13)
    - Gebe ein neues `ItemViewHolder` Objekt zurück (`return`), welches die eben erstellte `ItemView` als Parameter erhält
- Die Funktion `onBindViewHolder` gibt der `ItemView` aus dem `ViewHolder` den richtigen Inhalt
    - Die Klasse `ItemAdapter` bekommt im Konstruktor die in `Datasource` erstellte Liste mit `Movies` übergeben. Nutze die als Parameter übergebene `position`, um das entsprechende `Movie` Objekt aus der Liste zu holen: `dataset[position]`
    - Hole die TextView aus dem `ItemViewHolder` und weise ihr den Text zu, der als String Ressource im `Movie` Objekt gespeichert ist

        Hinweis: Um in dieser Klasse auf die Ressourcen zugreifen zu können, musst du auf die `resources` des übergebenen `context` verweisen
- Die Funktion `getItemCount` liefert lediglich die Größe der `dataset` Liste zurück, programiere diese Funktionalität
- Abschließend müssen wir die erstellten Klassen nur noch in der `MainActivity` Klasse zusammenbringen
    - Hole die Liste aus Filmtiteln, indem du die Funktion `loadMovies` aus der Klasse `Datasource` aufrufst
    - Erstelle ein neues `ItemAdapter` Objekt und weise es dem `adapter` Attribut der `RecyclerView` zu
    - Weise dem Attribut `layoutManager` einen GridLayoutManager zu

        - dem Konstruktor des GridLayoutManager müssen folgende Parameter mitgegeben werden
            - contex: Contex → Kontext der Activity
            - spanCount: Int → Anzahl der Spalten, die das Grid Layout erstellen soll

    Hinweis: übergebe hier als Kontext immer `this`

- Führe die App aus, wenn alles richtig implementiert ist, solltest du nun die Filme in der `RecyclerView` sehen!

Viel Erfolg! 📜

2. FINDE DEN FEHLER - Fehlersuche

Bei der Programmierung dieser APP ist etwas schiefgelaufen, begib dich auf die Fehlersuche und versuche das Problem zu lösen

- Öffne das Projekt “Finde den Fehler”
- Versuche die APP auszuführen und auf dem Emulator zu starten
- Versuche den Fehler zu finden
- Wenn der Fehler behoben ist, führe die APP erneut auf dem Emulator aus und überprüfe, ob die APP funktioniert

Viel Erfolg!