Kotlin 4.2 - API Calls

Aufgaben

Hinweis: Zu bearbeiten ist Aufgabe 1.

1. LACH DICH KAPUTT - API Call Strings

In dieser App holen wir mit einem API-Call eine Liste von Witzen. Diese werden wir in der App anzeigen lassen.

- Die Dokumentation der API, die wir in dieser App benutzen werden, findest Du hier: JokeAPI schau sie Dir einmal an.
- Öffne das Projekt “Lach Dich kaputt”
- Bevor es losgehen kann, müssen wir noch die richtigen Berechtigungen und Dependencies für den API-Call angeben:

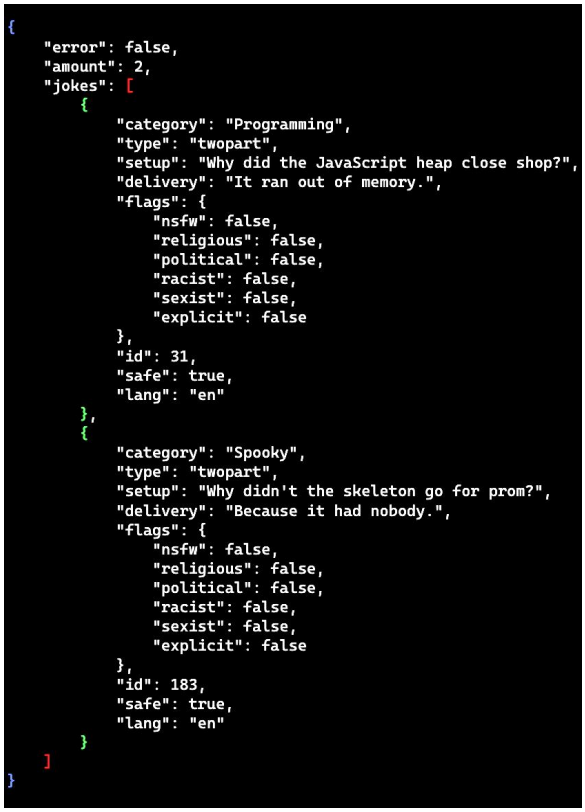
  In der `AndroidManifest.xml` Datei:

```
<uses-permission android:name="android.permission.INTERNET" />
```

In der `build.gradle(Module)` Datei unter `dependencies`:

```
//Retrofit
implementation "com.squareup.retrofit2:retrofit:2.9.0"
implementation "com.squareup.retrofit2:converter-moshi:2.9.0"
implementation "com.squareup.moshi:moshi-kotlin:1.13.0"
```

- Wir werden gleich den API-Call einbauen, vorher sehen wir uns noch einmal an, wie die Antwort des JokeAPI Servers aussieht:
    - Die gesamte Antwort ist in ein JSON Objekt verpackt {...}
    - Hinter der Variable "jokes" im JSON Objekt verbirgt sich wiederum ein Array [...] aus JSON Objekten
    - Jedes der JSON Objekte im Array steht für einen Witz mit mehreren Informationen, wie "setup" (Vorlage) und "delivery" (Antwort). Die JSON Witz-Objekte sind wieder durch {...} gekennzeichnet
    - Wenn der API Call in unserer APP verarbeitet wird, wird dieser in Kotlin Objekte übersetzt, daher brauchen wir für jedes JSON Objekt eine passende Klasse, mit der man die Objekte erzeugen kann:



```
{
    "error": false,
    "amount": 2,
    "jokes": [
        {
            "category": "Programming",
            "type": "twopart",
            "setup": "Why did the JavaScript heap close shop?",
            "delivery": "It ran out of memory.",
            "flags": {
                "nsfw": false,
                "religious": false,
                "political": false,
                "racist": false,
                "sexist": false,
                "explicit": false
            },
            "id": 31,
            "safe": true,
            "lang": "en"
        },
        {
            "category": "Spooky",
            "type": "twopart",
            "setup": "Why didn't the skeleton go for prom?",
            "delivery": "Because it had nobody.",
            "flags": {
                "nsfw": false,
                "religious": false,
                "political": false,
                "racist": false,
                "sexist": false,
                "explicit": false
            },
            "id": 183,
            "safe": true,
            "lang": "en"
        }
    ]
}
```

- Erstelle zuerst eine `data` Klasse `Joke` im package `datamodels`. Diese Klasse steht für ein einzelnes Witz-Objekt und enthält Variablen, die genauso heißen wie in der Serverantwort. Wir brauchen nicht alle Informationen und daher enthält die `data` Klasse `Joke` nur zwei Variablen: `setup` und `delivery` vom Typ `String`
- Erstelle nun eine `data` Klasse `JokeList` im package `datamodels`. Hier wird die Liste aus Witz Objekten gespeichert, die sich in der Antwort hinter "jokes" verbirgt. Daher braucht die Klasse für die Übersetzung genau eine Variable `jokes` vom Typ "Liste vom Typ `Joke` "
- Jetzt können wir den API-Service einbauen. Erstelle dazu ein **File** `JokeApiService` im package remote und programmiere sie folgendermaßen:

  Hinweis: ⇨ siehe Folien zu API Calls, Seite 13 & 14

    - Erstelle eine konstante Variable `BASE_URL`, in der die URL der API gespeichert ist:

      ```
      "https://v2.jokeapi.dev/joke/"
      ```

    - Erstelle eine private Variable `moshi`, in der ein Moshi mit einer KotlinJsonAdapterFactory gespeichert ist
    - Erstelle eine private Variable `retrofit`, in der ein Retrofit mit einer MoshiConverterFactory und der `BASE_URL` gespeichert ist
    - Erstelle ein Interface `JokeApiService`, in dem genau eine `suspend` Funktion zum Holen der Informationen namens `getJokes()` enthalten ist. Sie liefert den Typ `JokeList` zurück. In der `@GET` Annotation wird die URL folgendermaßen spezifiziert:

      ```
      "Any?lang=de&blacklistFlags=nsfw,religious,political,racist,sexist,
      explicit&type=twopart&amount=10"
      ```

      Hinweis: Füge die URL ohne Zeilenumbruch ein. Ändere `lang=de` zu `lang=en` für Witze auf Englisch (oder: `cs`, `es`, `fr`, `pt` für weitere Sprachen)
    - Erstelle schließlich noch ein `object` `JokeApi`, das eine Variable namens `retrofitService` enthält. Die Variable ist vom Typ `JokeApiService` von `lazy{retrofit.create(JokeApiService::class.java)}`
- Jetzt können wir den API-Service nutzen und die Informationen in der Klasse `AppRepository` abrufen, speichern und dem Rest der App zur Verfügung stellen. Gehe in die Klasse `AppRepository`
    - Füge dem Konstruktor der Klasse als Erstes eine private Variable `api` vom Typ `JokeApi` hinzu, da wir die API für den Abruf der Informationen hier brauchen
    - Erstelle eine LiveData Variable `jokes` als MutableLiveData vom Typ "Liste vom Typ `Joke`", in der alle Witze in einer Liste gespeichert werden sollen. Nutze dafür die bekannte Verschachtelung
    - Schreibe eine `suspend` Funktion `getJokes()`, die die Informationen aus der `JokeAPI` holt. Schreibe dafür einen `try catch` Block.
        - Im `try` Teil rufst du zuerst die Funktion `getJokes()` aus dem `retrofitService` der API auf und speicherst die Antwort in einer Variablen `jokeList`. Anschließend weist du der LiveData Variablen `jokes` die mit `shuffled` vermischte `jokes` Liste aus der Variable `jokeList` zu
        - Falls dabei ein Fehler auftritt, also im `catch` Block, soll mit `Log` die Fehlermeldung ausgegeben werden
- Jetzt sind die Informationen aus dem API-Call in der Klasse `AppRepository` zu finden. Wechsle in die Klasse `JokesViewModel`, hier können wir die Informationen jetzt verwerten
    - Erstelle ein `AppRepository` Objekt namens `repository`. Übergib hierbei als Parameter die `JokeApi`
    - Speichere die `jokes` aus dem Repository in einer eigenen Variable namens `jokes`

    - Schreibe eine Funktion `loadData()`, die dafür verantwortlich ist, den API Call über die Funktion `getJokes()` aus dem Repository durchzuführen. Dies tut sie in einer Coroutine, da der Abruf etwas dauert.
- Abschließend müssen wir nur noch die erhaltenen Informationen in der UI anzeigen. Wechsle in die Klasse `JokesFragment`
    - Hier wird die Funktion `loadData()` des ViewModels bereits zum Zeitpunkt des LayoutAufbaus, also in der Funktion `onCreateView` aufgerufen.
    - In der Funktion `onViewCreated` können wir jetzt die Liste in der Variable `jokes` aus dem ViewModel beobachten und bei jeder Änderung soll der RecyclerView `rvJokes` ein neuer `JokeAdapter` mit eben dieser Liste als Parameter zugewiesen werden
    - Damit dies funktioniert, musst du in der Klasse `JokeAdapter` noch ein paar kleine Änderungen vornehmen:
        - Ändere den Typ des `dataset` im Konstruktor von `Any` zu `Joke`
        - In der Funktion `onBindViewHolder`:
          Hole den aktuellen `Joke` aus der `position` im `dataset` und speichere ihn in einer Variablen. Weise den TextViews `tvSetup` und `tvDelivery` den Text aus der entsprechenden Variable des `Joke` zu
    - Zurück in der Klasse `JokesFragment` soll schließlich noch ein ClickListener auf den Button `btnRefresh` gesetzt werden, in dem die Funktion `loadData()` aus dem ViewModel aufgerufen wird. Dadurch wird jedes Mal ein neuer API-Call ausgeführt und die Liste wird aktualisiert.
- Die App mit API-Call sollte jetzt funktionieren! Führe sie aus und “lach dich kaputt”

Viel Erfolg! 😂