Retrofit

Zadatak 1

Kreirati poziv pomoću Retrofit klijenta koji dohvata najnovije filmove. Prikazati filmove u sklopu RecentFragment-a.

U prvom koraku ćemo dodati dependency u gradle file.

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

Klasu Movie ćemo prilagoditi da bude serializible i da je konverter može pretvoriti u JSON i obratno:

```
data class Movie (
    @SerializedName("id") var id: Long,
    @SerializedName("title") var title: String,
    @SerializedName("overview") var overview: String,
    @SerializedName("release_date") var releaseDate: String,
    @SerializedName("homepage") var homepage: String?,
    @SerializedName("poster_path") var posterPath: String?,
    @SerializedName("backdrop_path") var backdropPath: String?
)
```

Napomena: Obzirom da se žanr različito vraća u ovisnosti od poziva, izbacit ćemo isti, radi lakšeg rada.

Dodat ćemo još jednu klasu GetMoviesResponse koja predstavlja rezultate pretrage:

```
data class GetMoviesResponse(
    @SerializedName("page") val page: Int,
    @SerializedName("results") val movies: List<Movie>,
    @SerializedName("total_pages") val pages: Int
)
```

Kreirat ćemo interfejs Api u sklopu kojeg ćemo čuvati odgovarajuće pozive korištenjem Retrofit klijenta:

```
interface Api {
    @GET("movie/upcoming")
    suspend fun getUpcomingMovies(
        @Query("api_key") apiKey: String = BuildConfig.TMDB_API_KEY
    ): Response<GetMoviesResponse>
}
```

Trenutno se u interfejsu nalazi samo poziv za dobavljanje najnovijih filmova. Dokumentacija je data na sljedećem <u>linku</u>

Kreirat ćemo singleton objekat u kojem će se nalaziti retrofit instanca. Definisan je bazni URL web servisa, konverter i interfejs koji će klijent implementirati.

```
object ApiAdapter {
   val retrofit : Api = Retrofit.Builder()
        .baseUrl("https://api.themoviedb.org/3/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()
        .create(Api::class.java)
}
```

U sklopu MovieRepository klase je kreirana metoda koja će dohvatiti podatke unutar pozadinske coroutine. Korištenjem Retrofit-a se ne moraju koristiti coroutine, ali je dobra praksa sada koristiti iste.

```
suspend fun getUpcomingMovies(
) : GetMoviesResponse?{
   return withContext(Dispatchers.IO) {
      var response = ApiAdapter.retrofit.getUpcomingMovies()
      val responseBody = response.body()
      return @ withContext responseBody
   }
}
```

Unutar MovieListViewModel -a kreirana je metoda u sklopu koje se pokreće data coroutine-a i koja će s High-Order funkcijama vratiti listu filmova RecentMoviesFragment -u.

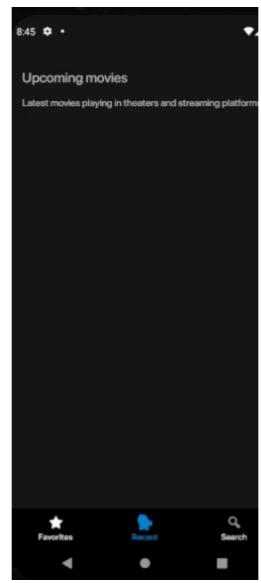
U fragmentu ćemo u onCreate izvršiti poziv ove metode, i implementirati ćemo funkcije onScucess i onError.

```
movieListViewModel.getUpcoming(
    onSuccess = ::onSuccess,
    onError = ::onError
)
```

Novokreirane metode:

```
fun onSuccess(movies:List<Movie>){
  val toast = Toast.makeText(context, "Upcoming movies found", Toast.LENGTH_SHORT)
  toast.show()
```

```
recentMoviesAdapter.updateMovies(movies)
}
fun onError() {
  val toast = Toast.makeText(context, "Search error", Toast.LENGTH_SHORT)
  toast.show()
}
```



Izgled aplikacije na kraju zadatka:

Zadatak za samostalni rad

1. Preuredite sve pozive ka web servisu da dohvataju podatke korištenjem Retrofita.