Laboratorijska vježba 10

LiveData

Zadatak 1

Omogućiti da se omiljeni filmovi iz baze učitavaju korištenjem `LiveData`, odnosno da se mijenjaju automatski s promjenom podataka (Observer pattern).

U prvom koraku dodat ćemo odgovarajući dependency u gradle.

```
implementation("androidx.lifecycle:lifecycle-livedata-ktx:2.3.1")
```

Obzirom da ćemo koristiti `LiveData`, izbacit ćemo `suspend` i `coroutine` stavke.

`MovieDao` metodu za dobavljanje svih filmova ćemo srediti na sljedeći način:

```
@Query("SELECT * FROM movie WHERE favourite=1")
    fun getAll(): LiveData<List<Movie>>
```

Vidimo da metoda više nije `suspend` i da vraća `LiveData<List<Movie>>`.

Izvršit ćemo odgovarajuće izmjene u metodi `MovieRepository` klase:

```
fun getFavorites(context: Context) : LiveData<List<Movie>> {
            var db = AppDatabase.getInstance(context)
            var movies = db!!.movieDao().getAll()
            return movies
    }
```

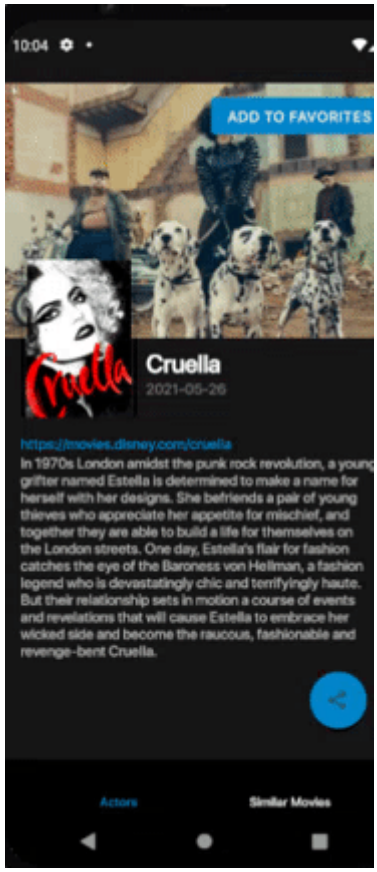
Konstruktor klase `MovieListViewModel` ćemo proširiti s `Context` atributom. Dodat ćemo privatni atribut `MutableLiveData<List<Movie>>` u klasu koji ćemo odmah inicijalizirati.

```
class MovieListViewModel(private val context: Context) {
    val favoriteMovies = MovieRepository.getFavorites(context)
```

U sklopu `OnCreate` metode fragmenta omiljenih filmova ćemo instancirati `MovieListViewModel` te registrovati observer nad atributom iste klase, koji će u slučaju promjene podataka ažurirati listu filmova.

```
context?.let { movieListViewModel= MovieListViewModel(it) }
val moviesObserver = Observer<List<Movie>> { movies ->
        favoriteMoviesAdapter.updateMovies(movies)
}
movieListViewModel.favoriteMovies.observe(this, moviesObserver)
```

Na ovaj način smo omogućili korištenje `LiveData` za dobavljanje najnovijih podataka, bez korištenja posebnih metoda i `coroutine`-a.



Rezultantna aplikacija je data na sljedećoj slici: