НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота № 7

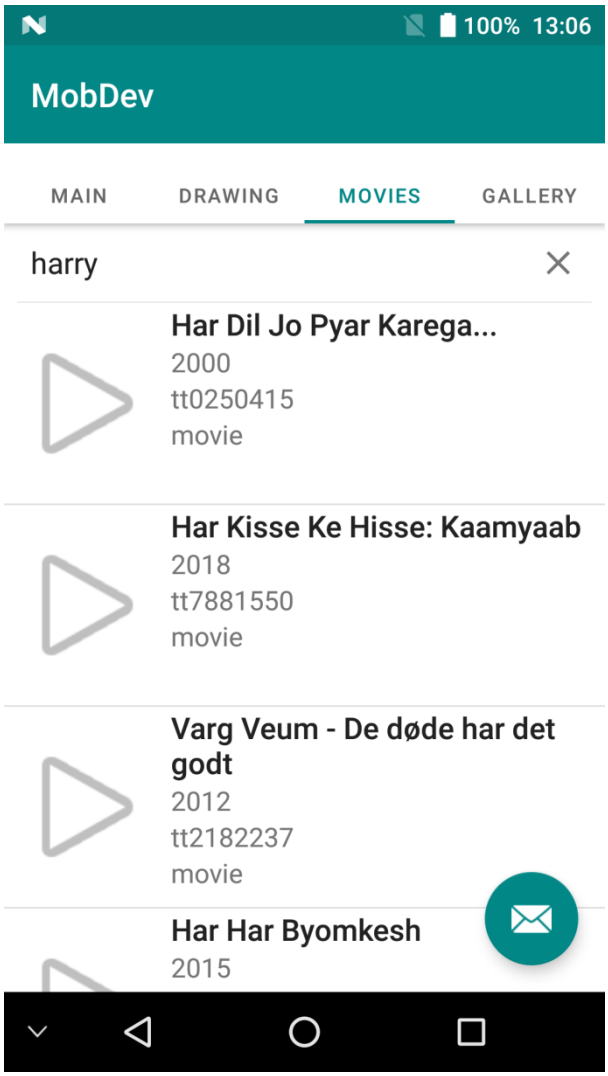
з дисципліни

“Програмування мобільних систем”

Виконав:
студент групи ІО-82
ЗК ІО-8226
Шевчук Олександр

Київ 2021

Скріншот додатку



Лістинг коду

FragmentMoviesList.java

```
package ua.kpi.compsys.io8226.tabs.tab_movies;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.ParseException;
import android.os.AsyncTask;
import android.os.Build;
```

```java
import android.os.Bundle;

import androidx.annotation.Nullable;
import androidx.annotation.RequiresApi;
import androidx.fragment.app.Fragment;
import androidx.appcompat.widget.SearchView;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.google.gson.Gson;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.net.UnknownHostException;
import java.util.ArrayList;

import ua.kpi.compsys.io8226.R;
import ua.kpi.compsys.io8226.repository.App;
import ua.kpi.compsys.io8226.repository.AppDatabase;
import ua.kpi.compsys.io8226.repository.SearchTable;
import ua.kpi.compsys.io8226.repository.TableMovies;
import ua.kpi.compsys.io8226.tabs.tab_movies.model.Movie;
import ua.kpi.compsys.io8226.tabs.tab_movies.model.MoviesList;


public class FragmentMoviesList extends Fragment {

    private static AppDatabase appDatabase;

    View view;
    ListView moviesListView;
    AdapterMoviesList adapterMoviesList;
    SearchView searchView;
    ProgressBar progressBar;
    ArrayList<Movie> movies = new ArrayList<>();
    TextView noResults;
    ImageButton addMovieButton;

   // change if expires
    String key = "7e9fe69e";


    @SuppressLint("UseCompatLoadingForDrawables")
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
```

```
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_movies_list, container, false);
    }

    @SuppressLint("UseCompatLoadingForDrawables")
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {

        this.view = view;
        noResults = view.findViewById(R.id.textView_noResults);
        searchView = view.findViewById(R.id.search_view);
        moviesListView = (ListView) view.findViewById(R.id.moviesListView);
        progressBar = view.findViewById(R.id.progressBar);
        adapterMoviesList = new AdapterMoviesList(this.getContext(), movies);
        addMovieButton = (ImageButton)
view.findViewById(R.id.button_addMovieButton);
        moviesListView.setAdapter(adapterMoviesList);


        //
        // searchView.setImeOptions(searchView.getImeOptions() |
EditorInfo.IME_FLAG_NO_EXTRACT_UI);


        moviesListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
              @Override
              public void onItemClick(AdapterView<?> adapterView, View view, int
position, long l) {

                  Movie item = adapterMoviesList.movies.get(position);

                  Intent intent = new Intent(view.getContext(),
MovieDetailsActivity.class);
                  intent.putExtra("imdbId", item.getImdbID());
                  intent.putExtra("poster", item.getPoster());

                  startActivity(intent);
              }
          }
        );

        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String s) {
                AsyncLoadMovies asyncLoadMovies = new AsyncLoadMovies();

                if (s.length() >= 3) {
                    //
                    // change key if invalid
                    //

asyncLoadMovies.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, s);

                    if (adapterMoviesList.getCount() == 0) {
                        moviesListView.setVisibility(View.GONE);
```

```
                    noResults.setVisibility(View.VISIBLE);
                } else {
                    noResults.setVisibility(View.GONE);
                    moviesListView.setVisibility(View.VISIBLE);
                }

            } else {
                adapterMoviesList.clear();
            }

            return true;
        }

        @Override
        public boolean onQueryTextChange(String s) {

            AsyncLoadMovies asyncLoadMovies = new AsyncLoadMovies();

            if (s.length() >= 3) {
                //
                // change key if invalid
                //

asyncLoadMovies.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, s);

                if (adapterMoviesList.getCount() == 0) {
                    moviesListView.setVisibility(View.GONE);
                    noResults.setVisibility(View.VISIBLE);
                } else {
                    noResults.setVisibility(View.GONE);
                    moviesListView.setVisibility(View.VISIBLE);
                }

            } else {
                adapterMoviesList.clear();
            }

            return true;
        }
    });

    appDatabase = App.getInstance().getDatabase();
}



private static class AsyncLoadMovieToDB extends AsyncTask<Movie, Void, Void> {
    @Override
    protected Void doInBackground(Movie... movies) {
        TableMovies tableMovies = new TableMovies(movies[0].getImdbID(),
                movies[0].getTitle(),
                movies[0].getYear(),
                movies[0].getType());
        String urlDisplay = movies[0].getPoster();
        Bitmap mIcon11 = null;
        try {
            InputStream in = new java.net.URL(urlDisplay).openStream();
            mIcon11 = BitmapFactory.decodeStream(in);
        } catch (Exception e) {
```

```
                Log.e("Error", e.getMessage());
                e.printStackTrace();
            }
            if (mIcon11 != null) {
                tableMovies.setPosterBitmap(mIcon11);

                appDatabase.movieDao().insert(tableMovies);
                SearchTable search = appDatabase.searchTableDao().getLastSearch();
                ArrayList<String> imdbIds = new ArrayList<>(search.foundMovies);
                imdbIds.add(tableMovies.getImdbID());
                search.foundMovies = imdbIds;

                appDatabase.searchTableDao().update(search);
            }
            return null;
        }
    }

    @SuppressLint("StaticFieldLeak")
    public class AsyncLoadMovies extends AsyncTask<String, Void, ArrayList<Movie>>
{

        @Override
        protected void onPreExecute() {
            super.onPreExecute();

            moviesListView.setVisibility(View.GONE);
            progressBar.setVisibility(View.VISIBLE);
        }

        @RequiresApi(api = Build.VERSION_CODES.M)
        @Override
        protected ArrayList<Movie> doInBackground(String... strings) {
            return search(strings[0]);
        }

        @RequiresApi(api = Build.VERSION_CODES.M)
        @Override
        protected void onPostExecute(ArrayList<Movie> movies) {
            super.onPostExecute(movies);
            if (movies == null) {
                Toast.makeText(view.getContext(), "Cannot load data!",
Toast.LENGTH_SHORT).show();
            } else {
                adapterMoviesList.update(movies);
            }

            progressBar.setVisibility(View.GONE);
            moviesListView.setVisibility(View.VISIBLE);
        }

        @RequiresApi(api = Build.VERSION_CODES.M)
        private ArrayList<Movie> search(String prompt) {
            String url = String.format(
                    "http://www.omdbapi.com/?apikey=%s&s=\"%s\"&page=1",
                    key,
                    prompt.trim().toLowerCase().replace("\\s+", "+"));

            try {
```

```
            ArrayList<Movie> movies = parseMovies(sendRequest(url));

            SearchTable newSearch = new SearchTable();
            newSearch.searchQueue = prompt;
            newSearch.foundMovies = new ArrayList<>();
            appDatabase.searchTableDao().insert(newSearch);

            for (Movie movie : movies) {
                new AsyncLoadMovieToDB().execute(movie);
            }

            return movies;

        } catch (UnknownHostException e) {
            System.err.println("Request timeout!");
            if (appDatabase.searchTableDao().getLastByQuery(prompt) != null) {
                return offlineLoad(prompt);
            }

        } catch (MalformedURLException e) {
            System.err.println(String.format("Incorrect URL <%s>!", url));
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ParseException e) {
            System.err.println("Incorrect content of JSON file!");
            e.printStackTrace();
        }
        return null;
    }

    private ArrayList<Movie> parseMovies(String jsonText)
            throws ParseException {

        ArrayList<Movie> results = new ArrayList<>();
        Gson gson = new Gson();

        try {
            MoviesList moviesList = gson.fromJson(jsonText, MoviesList.class);
            results.addAll(moviesList.getMovies());

        } catch (Exception e) {
            e.printStackTrace();
        }

        return results;
    }

    private String sendRequest(String url) throws IOException {
        StringBuilder result = new StringBuilder();

        URL getReq = new URL(url);
        URLConnection movieConnection = getReq.openConnection();
        BufferedReader in = new BufferedReader(new InputStreamReader(
                movieConnection.getInputStream()));
        String inputLine;

        while ((inputLine = in.readLine()) != null)
            result.append(inputLine).append("\n");
```

```
            in.close();

            return result.toString();
        }

        private ArrayList<Movie> offlineLoad(String query){
            ArrayList<Movie> newMovies = new ArrayList<>();
            ArrayList<String> imdbs =
appDatabase.searchTableDao().getLastByQuery(query).foundMovies;
            for (String imdb : imdbs) {

newMovies.add(appDatabase.movieDao().getMovieByImdbId(imdb).createMovieInfo());
            }
            return newMovies;
        }
    }
}
```

Висновок

В результаті виконання лабораторної я під’єднав базу даних SQLite до застосунку, щоб переглядати попередньо завантажені дані при відсутності під’єднання до мережі Інтернет.