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

Лабораторна робота № 6

з дисципліни

“Програмування мобільних систем”

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

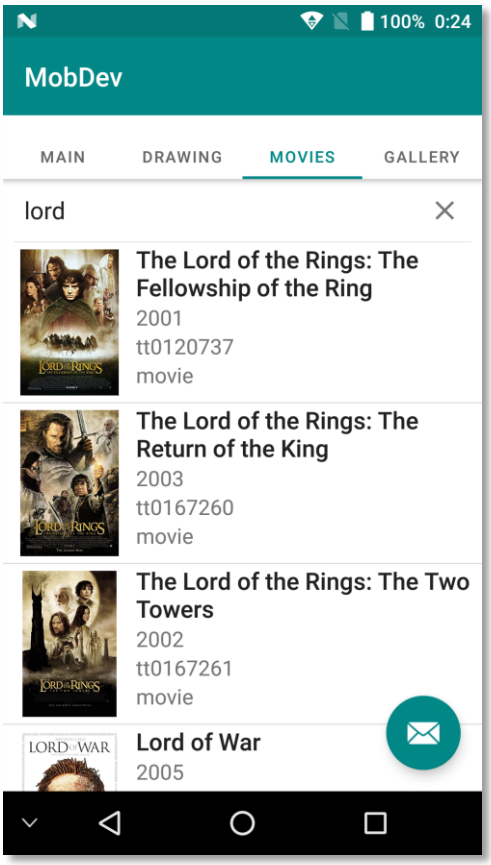
Київ 2021

Варіант 1
Предметна область – фільми URL-адреса для отримання даних – http://www.omdbapi.com/?apikey=**API_KEY**&s=**REQUEST**&page=1, де **API_KEY** – 7e9fe69e, **REQUEST** – запит із поля для пошуку. Зверніть увагу на те, що використання ключа **API_KEY** обмежене до 1000 запитів на день, якщо буде вичерпано кількість запитів на день для наведеного ключа, то можна отримати власний ключ, зареєструвавшись за посиланням.

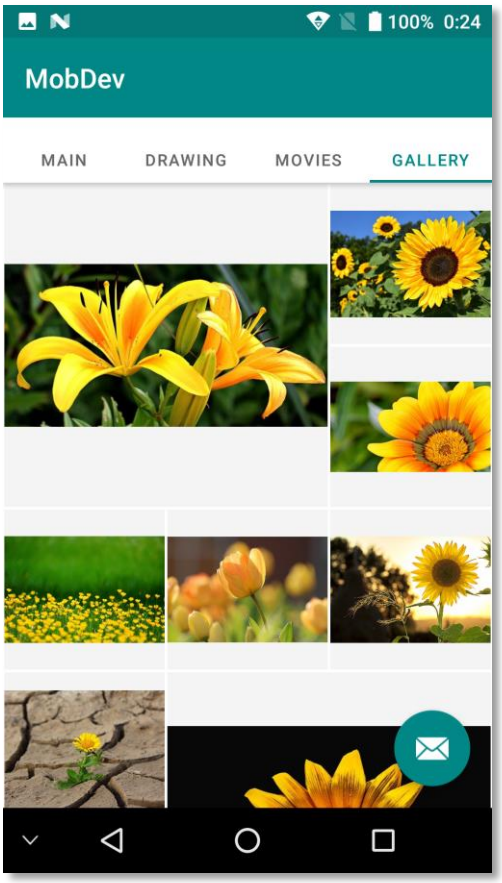
Варіант 1
Предметна область – фільми URL-адреса для отримання даних – http://www.omdbapi.com/?apikey=**API_KEY**&i=**IDENTIFIER**, де **API_KEY** – 7e9fe69e, **IDENTIFIER** – ідентифікатор відповідного фільму (поле **imdbID** із сутності фільму). Зверніть увагу на те, що використання ключа **API_KEY** обмежене до 1000 запитів на день, якщо буде вичерпано кількість запитів на день для наведеного ключа, то можна отримати власний ключ, зареєструвавшись за посиланням.

Варіант 1
REQUEST – “yellow+flowers” **COUNT** – 27

Скріншоти роботи додатку


100% 0:24
MobDev
MAIN
DRAWING
MOVIES
GALLERY
lord
The Lord of the Rings: The Fellowship of the Ring
2001
tt0120737
movie
The Lord of the Rings: The Return of the King
2003
tt0167260
movie
The Lord of the Rings: The Two Towers
2002
tt0167261
movie
Lord of War
2005


100% 0:24
MobDev
Title: The Lord of the Rings: The Fellowship of the Ring
Year: 2001
Genre: Action, Adventure, Drama, Fantasy
Director: Peter Jackson
Writer: J.R.R. Tolkien (novel), Fran Walsh (screenplay), Philippa Boyens (screenplay), Peter Jackson (screenplay)
Actors: Alan Howard, Noel Appleby, Sean Astin, Sala Baker
Country: New Zealand, USA
Language: English, Sindarin
Production: New Line Cinema, WingNut Films, Saul Zaentz Company
Released: 19 Dec 2001
Runtime: 178 min


100% 0:24
MobDev
MAIN
DRAWING
MOVIES
GALLERY

Лістинг коду

FragmentMoviesList.java

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

import android.annotation.SuppressLint;
import android.content.Intent;
import android.net.ParseException;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;

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

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 com.google.gson.Gson;

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

import ua.kpi.compsys.io8226.R;
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 {

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

    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) {

        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;
            }
        });
    }


    @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);
        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 {
            return parseMovies(sendRequest(url));

        } 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) {
        StringBuilder result = new StringBuilder();

        try {
            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();

            } catch (MalformedURLException e) {
                System.err.println(String.format("Incorrect URL <%s>!", url));
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }

            return result.toString();
        }
    }
}
```

FragmentGallery.java

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

import android.annotation.SuppressLint;
import android.net.ParseException;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ProgressBar;

import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.constraintlayout.widget.ConstraintSet;
import androidx.constraintlayout.widget.Guideline;
import androidx.fragment.app.Fragment;

import com.google.gson.Gson;
import com.squareup.picasso.Picasso;

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

import ua.kpi.compsys.io8226.R;


public class FragmentGallery extends Fragment {

    private static final String API_KEY = "19193969-87191e5db266905fe8936d565";
```

```
    private static final String REQUEST = "yellow+flowers";
    private static final int COUNT = 27;

    private InternetImagesList internetImagesList;
    private ArrayList<ImageView> imageViews;
    private ArrayList<ArrayList<Object>> placeholders;
    private LinearLayout linearLayout;
    private ProgressBar progressBar;
    private View view;


    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {

        view = inflater.inflate(R.layout.fragment_gallery, container, false);
        setRetainInstance(true);

        linearLayout = view.findViewById(R.id.linear_main);

        internetImagesList = new InternetImagesList();
        imageViews = new ArrayList<>();
        placeholders = new ArrayList<>();

        progressBar = (ProgressBar) view.findViewById(R.id.galleryProgressBar);

        AsyncLoadImagesInfo asyncLoadImagesInfo = new AsyncLoadImagesInfo();
        asyncLoadImagesInfo.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);

        for (ImageView image : imageViews) {
            image.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
        }

        return view;
    }

    private Guideline createGuideline(int orientation, float percent){
        Guideline guideline = new Guideline(view.getContext());
        guideline.setId(guideline.hashCode());

        ConstraintLayout.LayoutParams guideline_Params =
                new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.WRAP_CONTENT,
                        ConstraintLayout.LayoutParams.WRAP_CONTENT);
        guideline_Params.orientation = orientation;

        guideline.setLayoutParams(guideline_Params);

        guideline.setGuidelinePercent(percent);

        return guideline;
    }

    private void putImage(LinearLayout scrollMain, ArrayList<ImageView> allImages,
                          ArrayList<ArrayList<Object>> placeholderList, String
imageUrl) {

        ImageView newImage = new ImageView(view.getContext());

        progressBar.setVisibility(View.VISIBLE);
```

```
        Picasso.get().load(imageUrl).into(newImage, new
com.squareup.picasso.Callback() {

            @Override
            public void onSuccess() {
                progressBar.setVisibility(View.GONE);
            }

            @Override
            public void onError(Exception e) {
                e.printStackTrace();
            }
        });

        newImage.setBackgroundResource(R.color.image_background);

        ConstraintLayout.LayoutParams imageParams =
                new
ConstraintLayout.LayoutParams(ConstraintLayout.LayoutParams.MATCH_CONSTRAINT,
                        ConstraintLayout.LayoutParams.MATCH_CONSTRAINT);
        imageParams.setMargins(3, 3, 3, 3);
        imageParams.dimensionRatio = "1";
        newImage.setLayoutParams(imageParams);
        newImage.setId(newImage.hashCode());

        imageParams.setMargins(3, 3, 3, 3);
        imageParams.dimensionRatio = "1";
        progressBar.setLayoutParams(imageParams);
        progressBar.setId(newImage.hashCode());

        ConstraintLayout tmpLayout = null;
        ConstraintSet tmpSet = null;
        if (allImages.size() > 0) {
            tmpLayout = (ConstraintLayout) getConstraint(0, placeholderList);
            tmpSet = (ConstraintSet) getConstraint(1, placeholderList);

            tmpSet.clone(tmpLayout);

            tmpSet.setMargin(newImage.getId(), ConstraintSet.START, 3);
            tmpSet.setMargin(newImage.getId(), ConstraintSet.TOP, 3);
            tmpSet.setMargin(newImage.getId(), ConstraintSet.END, 3);
            tmpSet.setMargin(newImage.getId(), ConstraintSet.BOTTOM, 3);
        }

        if (allImages.size() % 9 != 0)
            tmpLayout.addView(newImage);

        switch (allImages.size() % 9){
            case 0:{
                placeholderList.add(new ArrayList<>());

                ConstraintLayout newConstraint = new
ConstraintLayout(view.getContext());
                placeholderList.get(placeholderList.size()-1).add(newConstraint);
                newConstraint.setLayoutParams(
                        new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
                                ViewGroup.LayoutParams.WRAP_CONTENT));
                scrollMain.addView(newConstraint);
```

```
                Guideline vertical_33 =
createGuideline(ConstraintLayout.LayoutParams.VERTICAL,
                        0.333333f);
                Guideline vertical_66 =
createGuideline(ConstraintLayout.LayoutParams.VERTICAL,
                        0.666666f);

                Guideline horizontal_20 =
createGuideline(ConstraintLayout.LayoutParams.HORIZONTAL,
                        0.2f);
                Guideline horizontal_40 =
createGuideline(ConstraintLayout.LayoutParams.HORIZONTAL,
                        0.4f);
                Guideline horizontal_60 =
createGuideline(ConstraintLayout.LayoutParams.HORIZONTAL,
                        0.6f);
                Guideline horizontal_80 =
createGuideline(ConstraintLayout.LayoutParams.HORIZONTAL,
                        0.8f);

                newConstraint.addView(vertical_33, 0);
                newConstraint.addView(vertical_66, 1);
                newConstraint.addView(horizontal_20, 2);
                newConstraint.addView(horizontal_40, 3);
                newConstraint.addView(horizontal_60, 4);
                newConstraint.addView(horizontal_80, 5);

                newConstraint.addView(newImage);

                ConstraintSet newConstraintSet = new ConstraintSet();
                placeholderList.get(placeholderList.size()-
1).add(newConstraintSet);
                newConstraintSet.clone(newConstraint);

                newConstraintSet.connect(newImage.getId(), ConstraintSet.START,
                        ConstraintSet.PARENT_ID, ConstraintSet.START);
                newConstraintSet.connect(newImage.getId(), ConstraintSet.TOP,
                        ConstraintSet.PARENT_ID, ConstraintSet.TOP);
                newConstraintSet.connect(newImage.getId(), ConstraintSet.END,
                        vertical_66.getId(), ConstraintSet.START);
                newConstraintSet.connect(newImage.getId(), ConstraintSet.BOTTOM,
                        horizontal_40.getId(), ConstraintSet.TOP);

                newConstraintSet.applyTo(newConstraint);
                break;
            }

            case 1: {
                tmpSet.connect(newImage.getId(), ConstraintSet.START,
                        tmpLayout.getChildAt(1).getId(), ConstraintSet.START);
                tmpSet.connect(newImage.getId(), ConstraintSet.TOP,
                        ConstraintSet.PARENT_ID, ConstraintSet.TOP);
                tmpSet.connect(newImage.getId(), ConstraintSet.END,
                        ConstraintSet.PARENT_ID, ConstraintSet.END);
                tmpSet.connect(newImage.getId(), ConstraintSet.BOTTOM,
                        tmpLayout.getChildAt(2).getId(), ConstraintSet.TOP);

                tmpSet.applyTo(tmpLayout);
```

```
    break;
}

case 2: {
    tmpSet.connect(newImage.getId(), ConstraintSet.START,
            tmpLayout.getChildAt(1).getId(), ConstraintSet.START);
    tmpSet.connect(newImage.getId(), ConstraintSet.TOP,
            tmpLayout.getChildAt(2).getId(), ConstraintSet.TOP);
    tmpSet.connect(newImage.getId(), ConstraintSet.END,
            ConstraintSet.PARENT_ID, ConstraintSet.END);
    tmpSet.connect(newImage.getId(), ConstraintSet.BOTTOM,
            tmpLayout.getChildAt(3).getId(), ConstraintSet.TOP);

    tmpSet.applyTo(tmpLayout);
    break;
}

case 3: {
    tmpSet.connect(newImage.getId(), ConstraintSet.START,
            ConstraintSet.PARENT_ID, ConstraintSet.START);
    tmpSet.connect(newImage.getId(), ConstraintSet.TOP,
            tmpLayout.getChildAt(3).getId(), ConstraintSet.BOTTOM);
    tmpSet.connect(newImage.getId(), ConstraintSet.END,
            tmpLayout.getChildAt(0).getId(), ConstraintSet.START);
    tmpSet.connect(newImage.getId(), ConstraintSet.BOTTOM,
            tmpLayout.getChildAt(4).getId(), ConstraintSet.TOP);

    tmpSet.applyTo(tmpLayout);
    break;
}

case 4: {
    tmpSet.connect(newImage.getId(), ConstraintSet.START,
            tmpLayout.getChildAt(0).getId(), ConstraintSet.START);
    tmpSet.connect(newImage.getId(), ConstraintSet.TOP,
            tmpLayout.getChildAt(3).getId(), ConstraintSet.BOTTOM);
    tmpSet.connect(newImage.getId(), ConstraintSet.END,
            tmpLayout.getChildAt(1).getId(), ConstraintSet.START);
    tmpSet.connect(newImage.getId(), ConstraintSet.BOTTOM,
            tmpLayout.getChildAt(4).getId(), ConstraintSet.TOP);

    tmpSet.applyTo(tmpLayout);
    break;
}

case 5: {
    tmpSet.connect(newImage.getId(), ConstraintSet.START,
            tmpLayout.getChildAt(1).getId(), ConstraintSet.END);
    tmpSet.connect(newImage.getId(), ConstraintSet.TOP,
            tmpLayout.getChildAt(3).getId(), ConstraintSet.BOTTOM);
    tmpSet.connect(newImage.getId(), ConstraintSet.END,
            ConstraintSet.PARENT_ID, ConstraintSet.END);
    tmpSet.connect(newImage.getId(), ConstraintSet.BOTTOM,
            tmpLayout.getChildAt(4).getId(), ConstraintSet.TOP);

    tmpSet.applyTo(tmpLayout);
    break;
}
```

```
            case 6: {
                tmpSet.connect(newImage.getId(), ConstraintSet.START,
                        ConstraintSet.PARENT_ID, ConstraintSet.START);
                tmpSet.connect(newImage.getId(), ConstraintSet.TOP,
                        tmpLayout.getChildAt(4).getId(), ConstraintSet.BOTTOM);
                tmpSet.connect(newImage.getId(), ConstraintSet.END,
                        tmpLayout.getChildAt(0).getId(), ConstraintSet.START);
                tmpSet.connect(newImage.getId(), ConstraintSet.BOTTOM,
                        tmpLayout.getChildAt(5).getId(), ConstraintSet.TOP);

                tmpSet.applyTo(tmpLayout);
                break;
            }

            case 7: {
                tmpSet.connect(newImage.getId(), ConstraintSet.START,
                        tmpLayout.getChildAt(0).getId(), ConstraintSet.END);
                tmpSet.connect(newImage.getId(), ConstraintSet.TOP,
                        tmpLayout.getChildAt(4).getId(), ConstraintSet.BOTTOM);
                tmpSet.connect(newImage.getId(), ConstraintSet.END,
                        ConstraintSet.PARENT_ID, ConstraintSet.END);
                tmpSet.connect(newImage.getId(), ConstraintSet.BOTTOM,
                        ConstraintSet.PARENT_ID, ConstraintSet.BOTTOM);

                tmpSet.applyTo(tmpLayout);
                break;
            }

            case 8: {
                tmpSet.connect(newImage.getId(), ConstraintSet.START,
                        ConstraintSet.PARENT_ID, ConstraintSet.START);
                tmpSet.connect(newImage.getId(), ConstraintSet.TOP,
                        tmpLayout.getChildAt(5).getId(), ConstraintSet.BOTTOM);
                tmpSet.connect(newImage.getId(), ConstraintSet.END,
                        tmpLayout.getChildAt(0).getId(), ConstraintSet.START);
                tmpSet.connect(newImage.getId(), ConstraintSet.BOTTOM,
                        ConstraintSet.PARENT_ID, ConstraintSet.BOTTOM);

                tmpSet.applyTo(tmpLayout);
                break;
            }
        }

        allImages.add(newImage);
    }


    private Object getConstraint(int index, ArrayList<ArrayList<Object>> list){
        return list.get(list.size()-1).get(index);
    }


    @SuppressLint("StaticFieldLeak")
    public class AsyncLoadImagesInfo extends AsyncTask<Void, Void,
ArrayList<InternetImage>> {

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

```
            progressBar.setVisibility(View.VISIBLE);
        }

        @RequiresApi(api = Build.VERSION_CODES.M)
        @Override
        protected ArrayList<InternetImage> doInBackground(Void... voids) {
            return search();
        }

        @RequiresApi(api = Build.VERSION_CODES.M)
        @Override
        protected void onPostExecute(ArrayList<InternetImage> images) {
            super.onPostExecute(images);
            internetImagesList.getInternetImages().addAll(images);

            for (InternetImage image : internetImagesList.getInternetImages()) {
                String imageUrl = image.getWebFormatUrl();
                putImage(linearLayout, imageViews, placeholders, imageUrl);
            }

            progressBar.setVisibility(View.GONE);
        }


        @RequiresApi(api = Build.VERSION_CODES.M)
        private ArrayList<InternetImage> search() {
            String url = String.format(

"https://pixabay.com/api/?key=%s&q=\"%s\"&image_type=photo&per_page=%d",
                    API_KEY,
                    REQUEST,
                    COUNT);

            try {
                return parseImagesInfo(sendRequest(url));

            } catch (ParseException e) {
                System.err.println("Incorrect content of JSON file!");
                e.printStackTrace();
            }
            return null;
        }

        private ArrayList<InternetImage> parseImagesInfo(String jsonText)
                throws ParseException {

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

            try {
                InternetImagesList imagesList = gson.fromJson(jsonText,
InternetImagesList.class);
                results.addAll(imagesList.getInternetImages());

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

```
            return results;
        }

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

            try {
                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();

            } catch (MalformedURLException e) {
                System.err.println(String.format("Incorrect URL <%s>!", url));
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }

            return result.toString();
        }
    }
}
```

InternetImage.java

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

import com.google.gson.annotations.SerializedName;

public class InternetImage {

    @SerializedName("webformatURL")
    String webFormatUrl;


    public InternetImage(String webFormatUrl) {
        this.webFormatUrl = webFormatUrl;
    }


    public String getWebFormatUrl() {
        return webFormatUrl;
    }

    public void setWebFormatUrl(String webFormatUrl) {
        this.webFormatUrl = webFormatUrl;
    }
}
```

InternetImagesList.java

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

import com.google.gson.annotations.SerializedName;

import java.util.ArrayList;
import java.util.List;

public class InternetImagesList {

    @SerializedName("hits")
    private List<InternetImage> internetImages = new ArrayList<>();


    public InternetImagesList(List<InternetImage> internetImages) {
        this.internetImages = internetImages;
    }

    public InternetImagesList() {
    }

    public List<InternetImage> getInternetImages() {
        return internetImages;
    }

    public void setInternetImages(List<InternetImage> internetImages) {
        this.internetImages = internetImages;
    }

    public InternetImage getInternetImage(int index) {
        return internetImages.get(index);
    }

    public void setInternetImage(int index, InternetImage internetImage) {
        internetImages.set(index, internetImage);
    }

    public void addInternetImage(InternetImage internetImage) {
        internetImages.add(internetImage);
    }
}
```

Висновок

В результаті виконання лабораторної я змінив логіку роботи вкладок зі списком фільмів та галереєю. Вкладки тепер підвантажують інформацію про сутності з

мережі Інтернет. Також реалізована анімація індикатора активності при завантаженні даних з мережі.