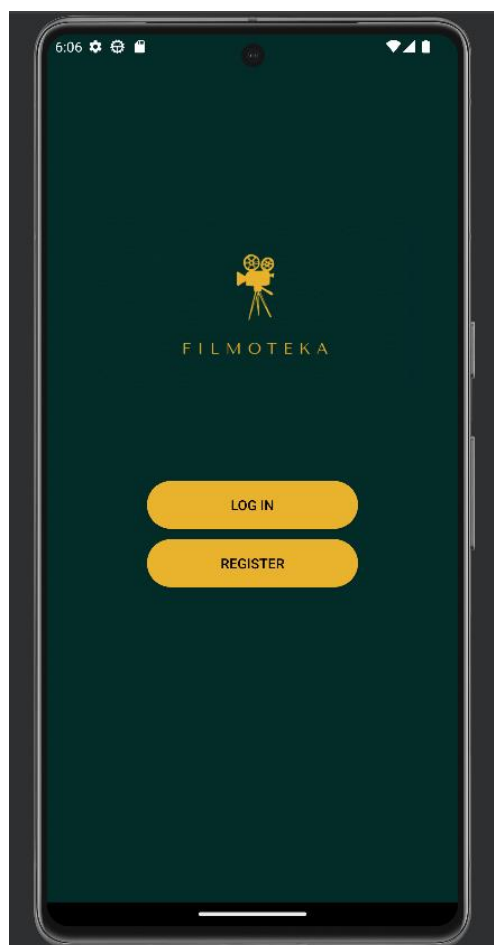


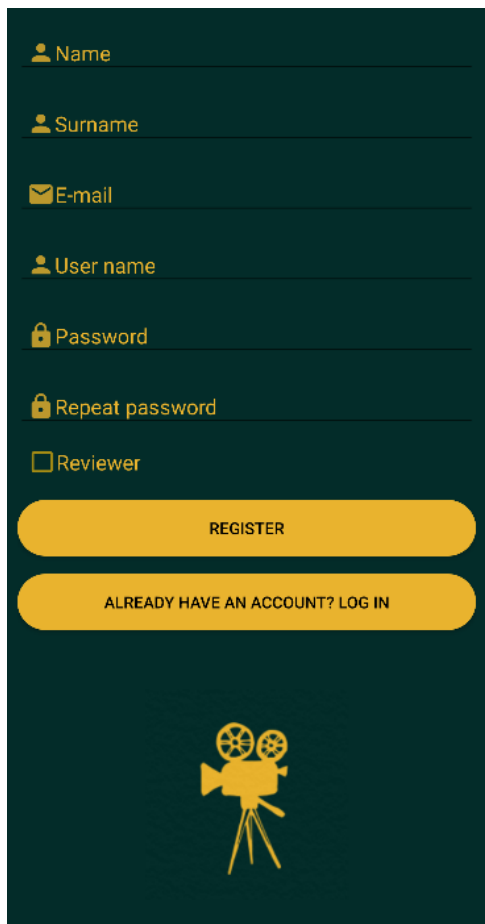
Wydział Informatyki Politechniki Białostockiej Przedmiot: Systemy Mobilne	Data oddania: 29.01.2024
Grupa PS2 Wykonawcy projektu: 1.Bartłomiej Wójcik 2.Michał Pragacz 3.Maja Rutkowska 4.Michalina Wasiluk	Prowadzący zajęcia: Dr. inż. Katarzyna Łukaszewicz

Filmoteka – aplikacja na system operacyjny ANDROID do zarządzania filmami. Zbiera w jednym miejscu funkcjonalności takie jak: przeglądanie informacji dot. filmów, oglądanie zwiastunów, tworzenie list filmowych, wyszukiwanie, przeglądanie ocen i opinii. Całość przedstawiona jest w formie przystępnej dla użytkownika, pozwala na wybór jak najbardziej interesujących nas filmów. Filmy pobierane są z API – TMDB i codziennie aktualizowane, co zapewnia najświeższe informacje. Aplikacja dostosowuje się do naszej lokalizacji, pobierając filmy wg. polskich lub amerykańskich list popularności. Aplikacja jest również umiędzynarodowiona oraz możliwe jest korzystanie z niej w trybie poziomym i pionowym telefonu. W celu zachęcenia użytkowników do częstszego korzystania z serwisu aplikacja wysyła codzienne powiadomienia.



Funkcjonalności aplikacji:

1. Rejestracja konta w lokalnej bazie danych, utworzenie konta z uprawnieniami użytkownika lub recenzenta.

A registration form with a dark teal background and light blue text. It contains input fields for Name, Surname, E-mail, User name, Password, and Repeat password. There is a checkbox for 'Reviewer' and two buttons: 'REGISTER' and 'ALREADY HAVE AN ACCOUNT? LOG IN'. A yellow film camera icon is at the bottom.

Name

Surname

E-mail

User name


Password

Repeat password

☐ Reviewer

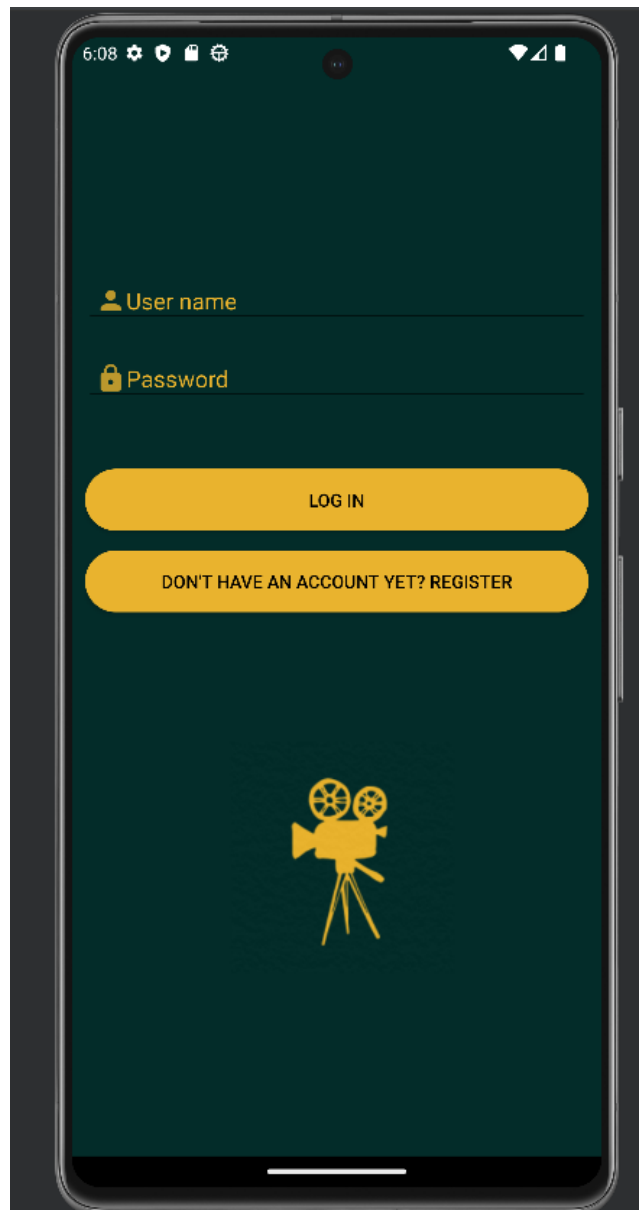
REGISTER

ALREADY HAVE AN ACCOUNT? LOG IN

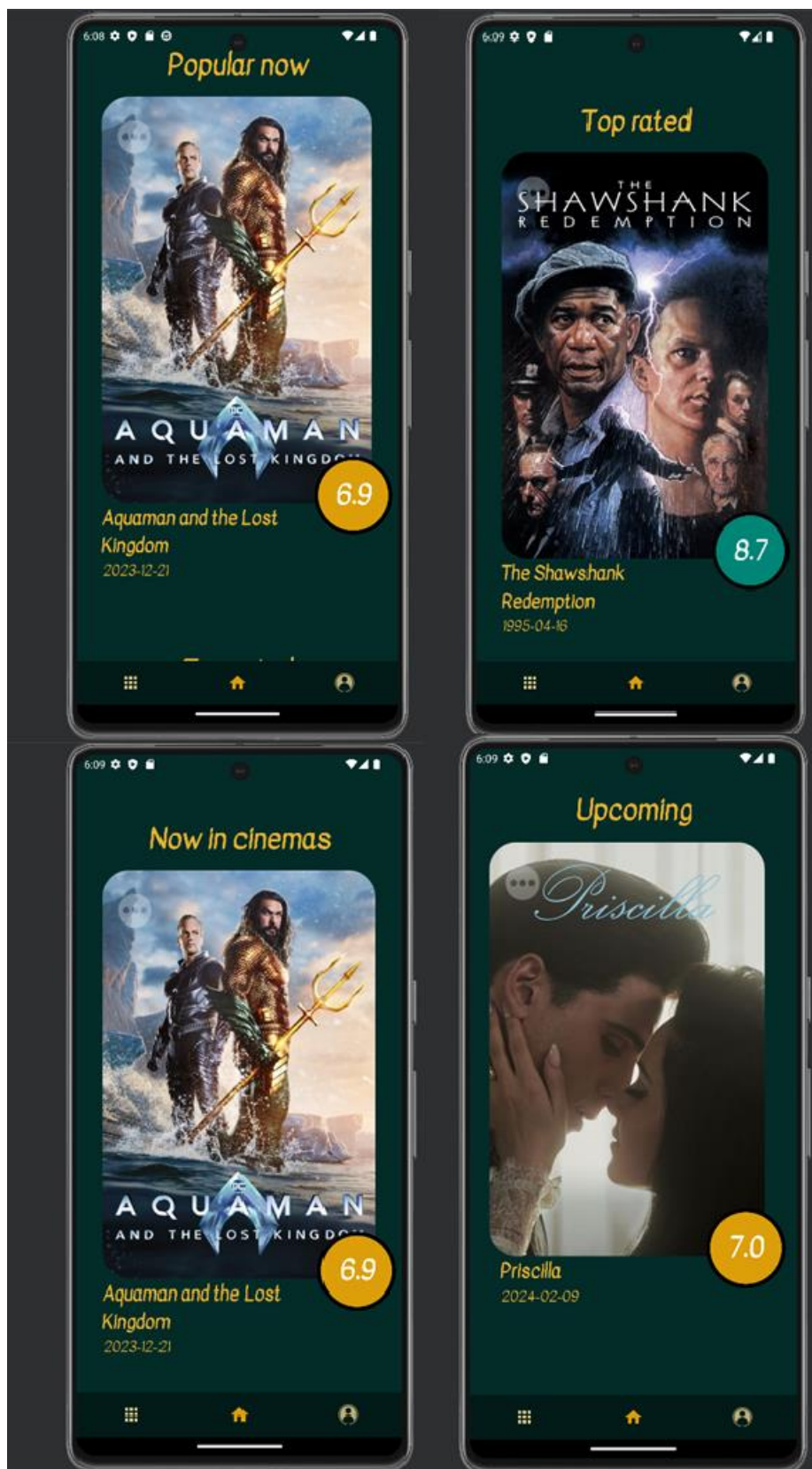


W celu założenia konta pobieramy dane takie jak: imię, nazwisko, email, nazwę użytkownika, hasło oraz informację, o roli, którą użytkownik chce sobie przypisać. Niektóre tych informacji potem będą mogły ulec edycji.

2. Logowanie do konta w celu skorzystania z niego.

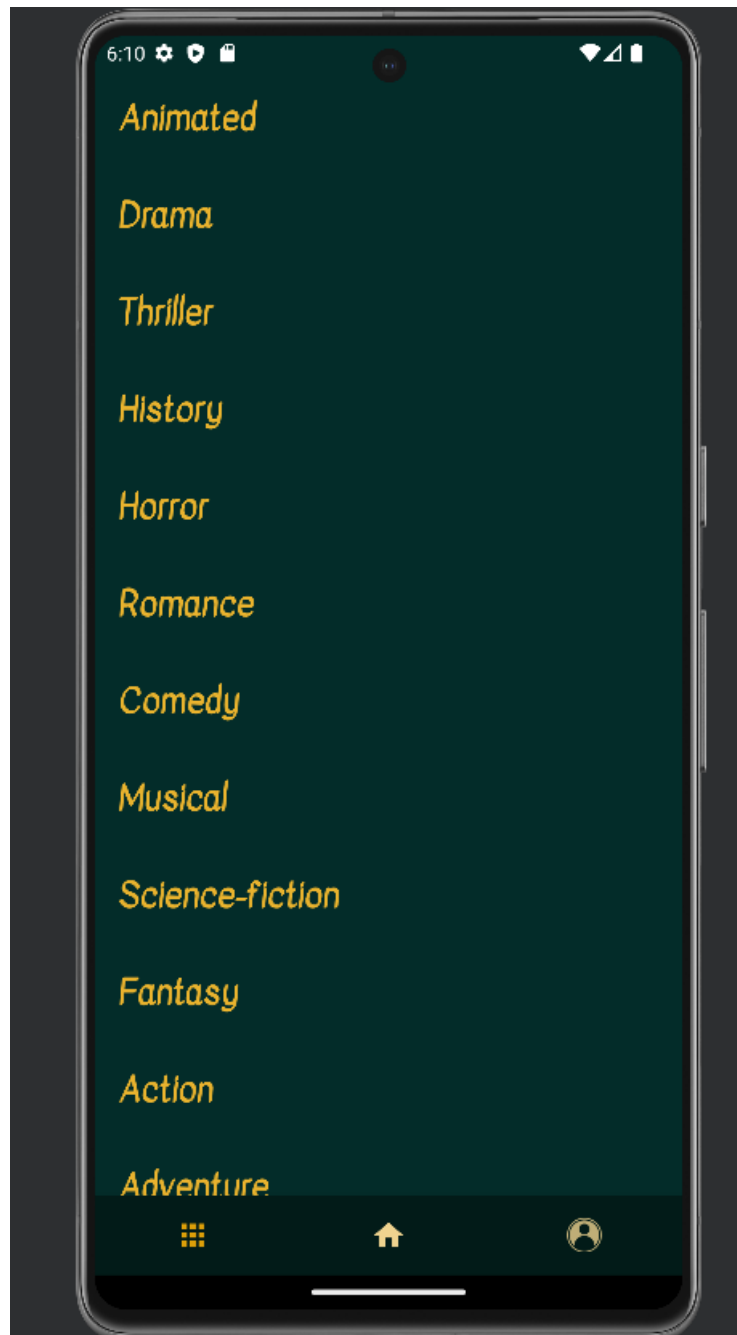


3. Przeglądanie filmów na stronie głównej aplikacji.

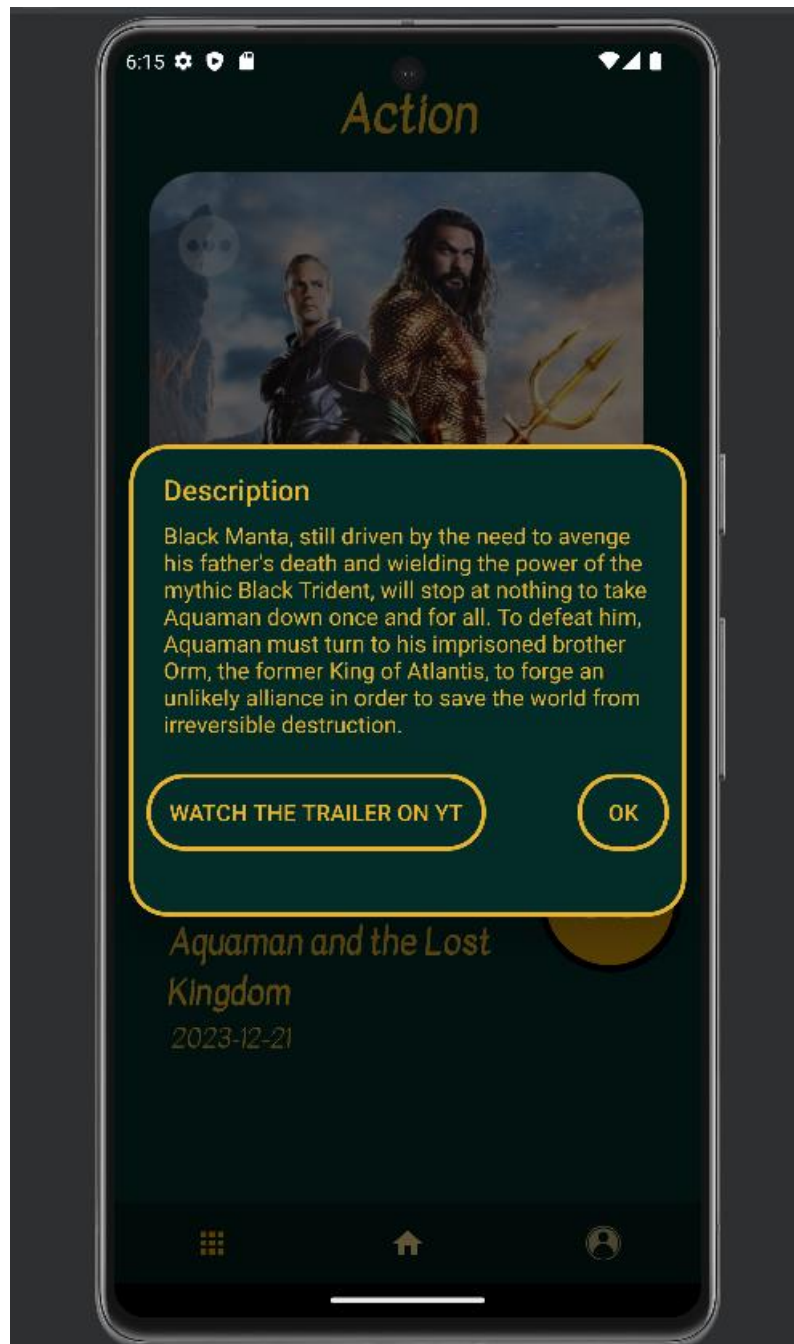


Filmy przegląda się za pomocą akcji przesuwania w dół, przejścia po podanych wyżej kategoriach oraz akcją przesuwania w bok, w celu poruszania się w aktualnie wyświetlającej się kategorii.

4. Przeglądanie filmów podzielonych na kategorie w celu łatwiejszego znalezienia interesujących treści.

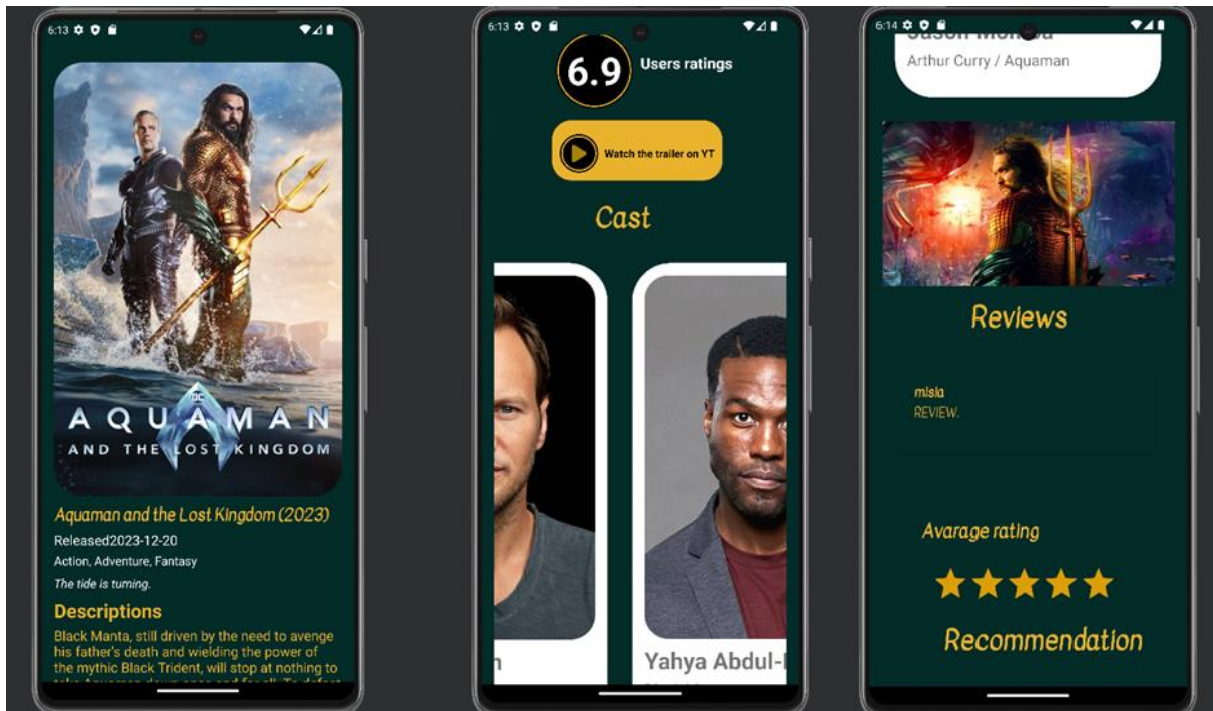


5. Wyświetlanie szczegółów dotyczących filmów.
 - a. Przy pomocy przytrzymania plakatu.



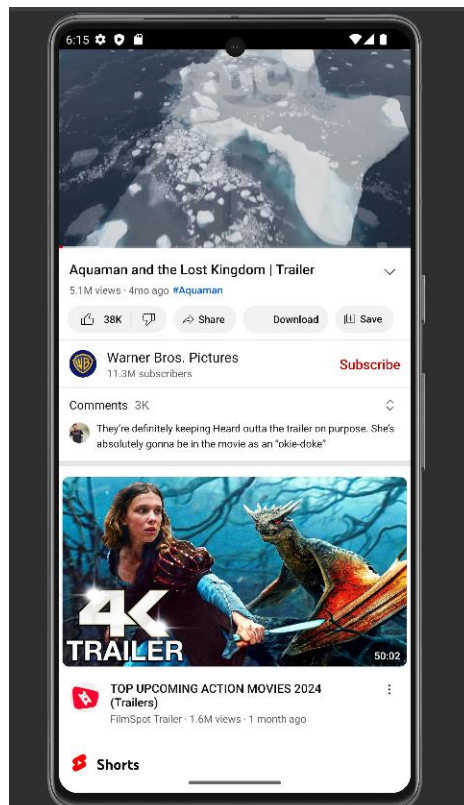
Po przytrzymaniu plakatu wyświetla nam się krótki opis filmu, a także odnośnik do strony youtube, gdzie można obejrzeć zwiastun podanego filmu.

b. Przy pomocy pojedynczego kliknięcia w plakat.

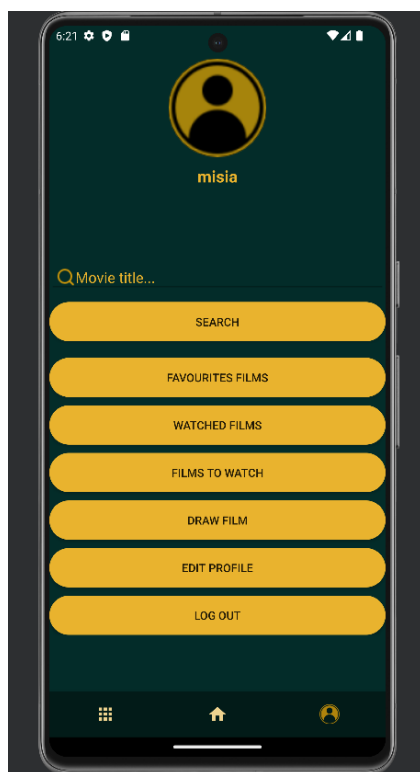


Po pojedynczym przyciśnięciu plakatu wyświetla nam się dokładniejszy opis filmu, jego kategorię, ocenę nadaną przez użytkowników, odnośnik do zwiastunu, członkowie obsady, a także opinie i średnia ocena wystawione przez recenzentów serwisu.

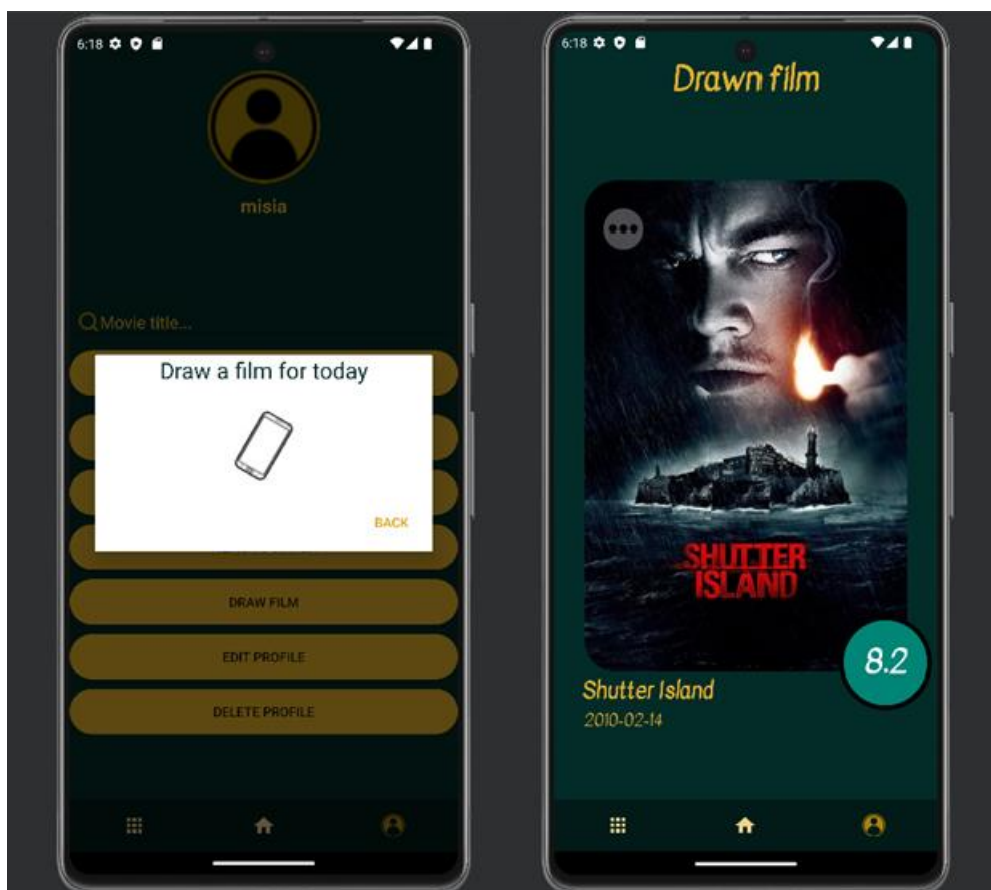
6. Wyświetlanie zwiastunów filmowych w serwisie youtube.



7. Przeglądanie swojego profilu.

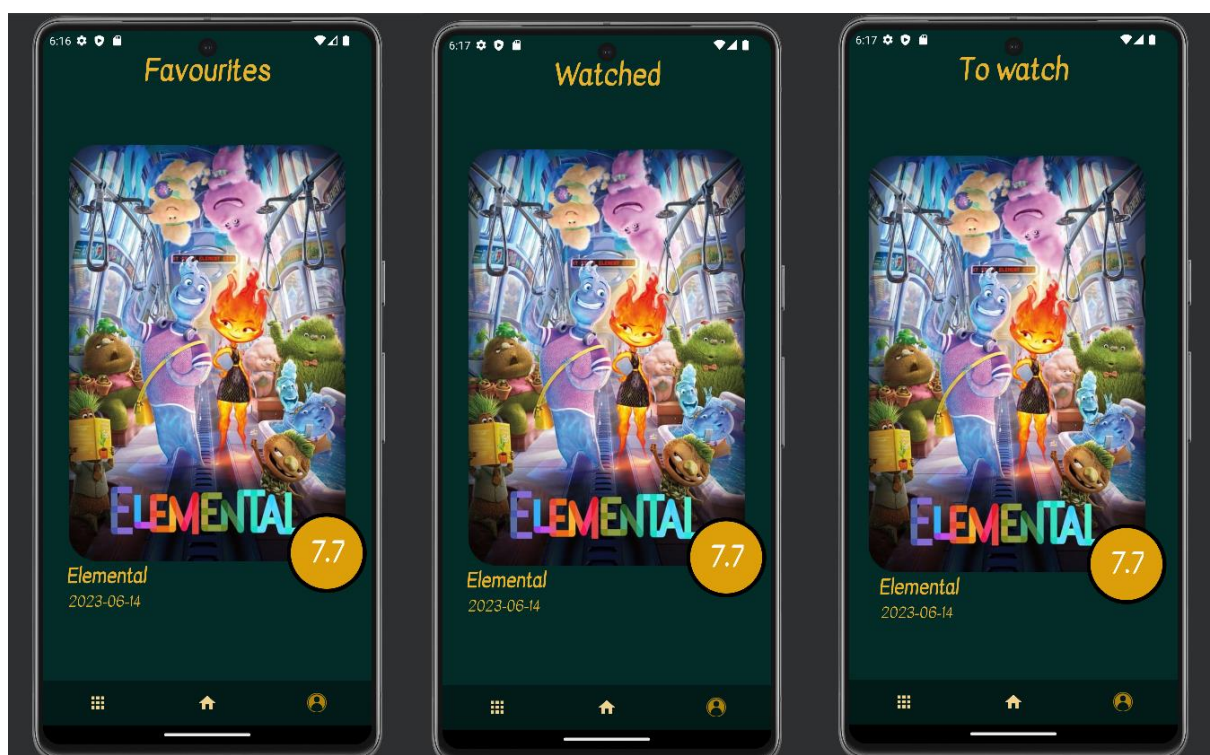
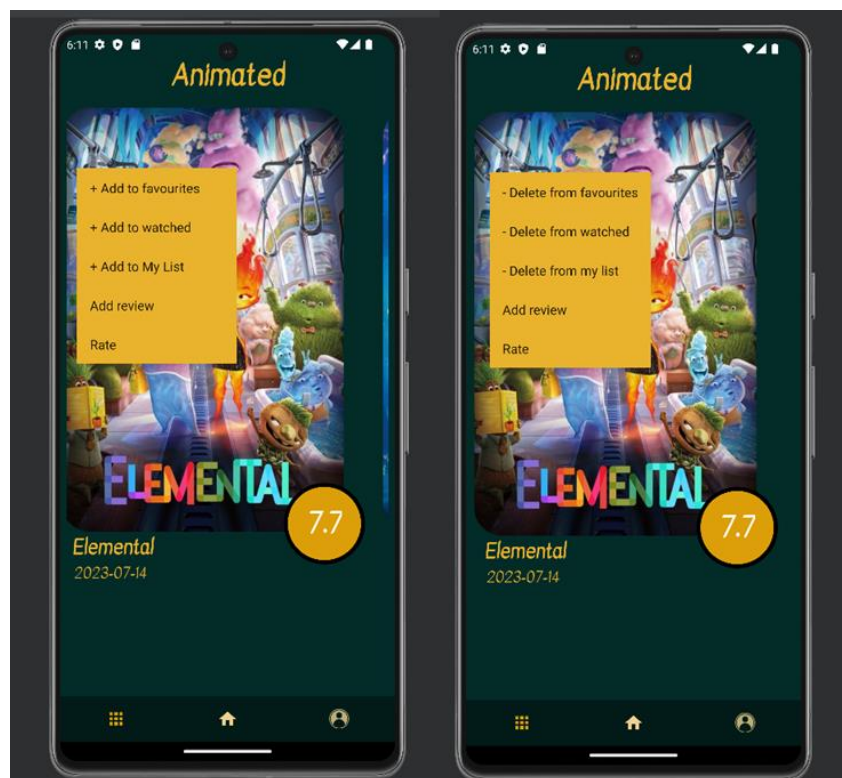


8. Losowanie propozycji filmowej.

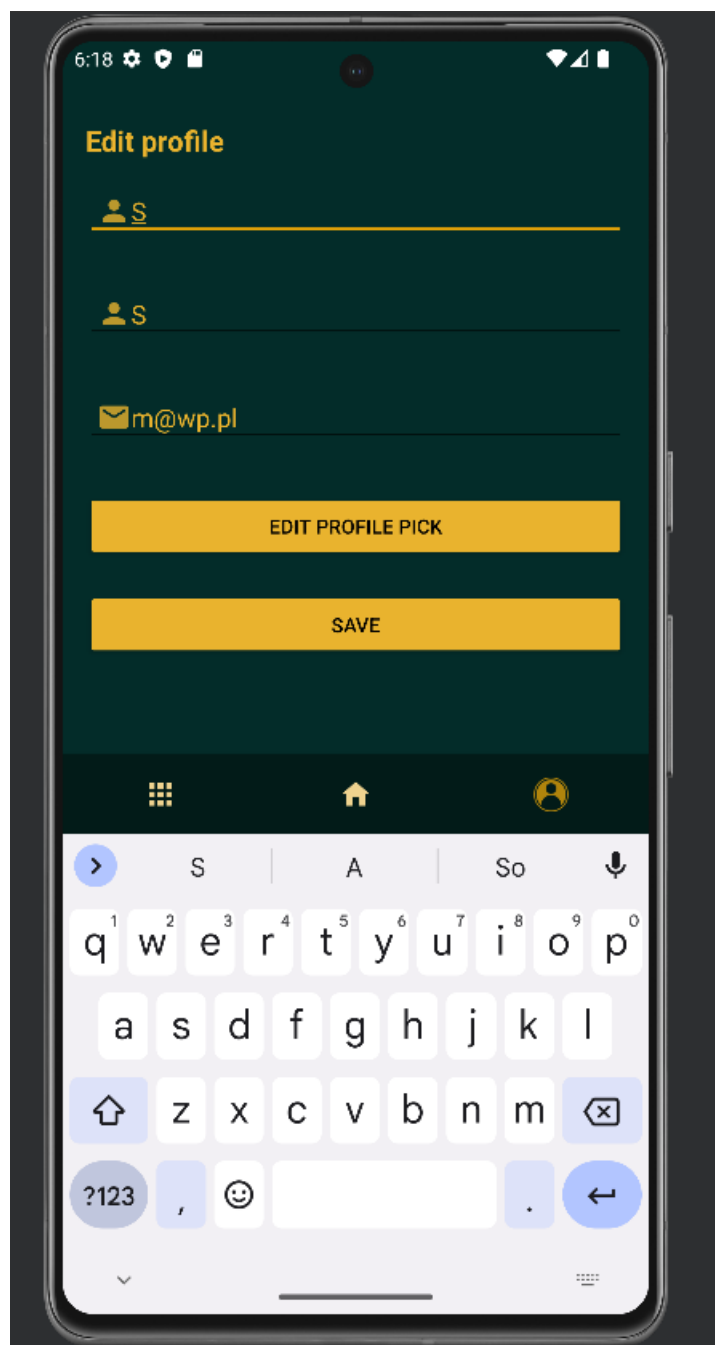


Po wciśnięciu przycisku do losowania filmu pojawia się animacja informująca o tym, żeby rozpocząć odpowiedni ruch telefonem, a po wyczuciu odpowiedniego zachowania wyświetlany jest film, który wylosował się spośród listy 200 najlepiej ocenianych.

9. Tworzenie list filmowych do zarządzania: ulubionymi, obejrzanymi i chcianymi obejrzeć filmami.



10. Edycję profilu.

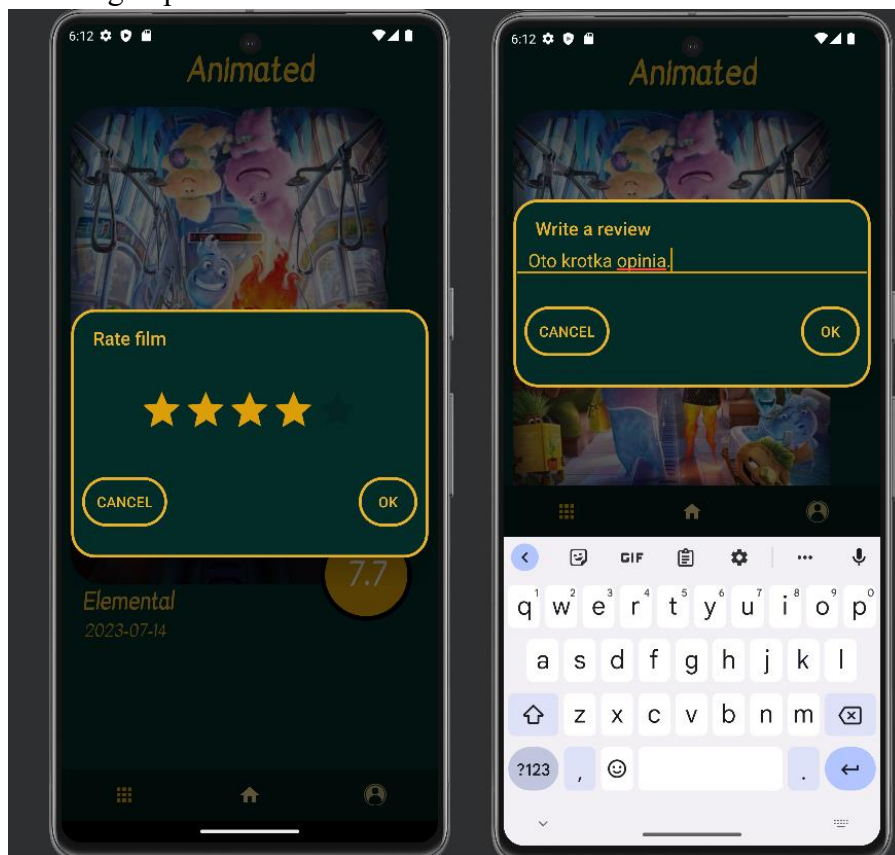


Możliwymi do edycji danymi są imię, nazwisko oraz adres mailowy użytkownika. Dodatkowo jest opcja edycji swojego zdjęcia profilowego, które wybiera się z galerii telefonu.

11. Wyszukiwanie interesującego nas filmu.



12. Będąc recenzentem dodatkowo posiadamy możliwość ocenienia filmu, a także napisania do niego opinii.



13. Wylogowywanie z konta.

Aplikacja korzysta z różnorodnych zapytań do bazy danych oraz pobierania danych z API.

Przykładowe zapytania do bazy danych:

- SELECT

```
public interface ReviewDao {  
    1 usage 1 implementation 👤 Michał  
    @Insert(onConflict = OnConflictStrategy.IGNORE)  
    long insertReview(Review review);  
  
    1 usage 1 implementation 👤 Michał  
    @Query("SELECT COUNT(*) FROM reviews WHERE userId = :userId AND movieId = :movieDbId")  
    int checkIfReviewExists(long userId, int movieDbId);  
  
    1 usage 1 implementation 👤 Michał  
    @Query("SELECT reviews.*, users.userName AS authorUserName " +  
            "FROM reviews " +  
            "INNER JOIN users ON reviews.userId = users.userId " +  
            "WHERE reviews.movieId = :movieId")  
    List<ReviewWithAuthor> getReviewsWithAuthorByMovieId(int movieId);  
}
```

- INSERT

```
@Insert(onConflict = OnConflictStrategy.IGNORE)  
long insertMyListMovie(MyListMovies myListMovies);
```

- DELETE

```
1 usage 1 implementation 👤 Bartłomiej Wojcik  
@Query("DELETE FROM favourite_movies WHERE userId = :userId AND movieDbId = :movieDbId")  
void deleteFavouriteMovie(long userId, int movieDbId);
```

- UPDATE

```
1 usage 1 implementation 👤 Michał  
@Query("UPDATE ratings SET rating = :newRating WHERE userId = :userId AND movieId = :movieDbId")  
void updateRatings(long userId, int movieDbId, float newRating);
```

Przykładowe tworzone zapytania do API:

```
Michał
MovieList.getTopRatedMovies( accessToken: "eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI2NmI1OTA2OTU4ZDY0YjRmOWM1MjMzMzY0YiIsInN1YiI6IjY1O...
11 usages  Michał
@Override
public void onMoviesFetched(List<Movie> movies) {
    topRatedMovieList = movies;
    topRatedMovieAdapter.setMovies(topRatedMovieList);
    topRatedMovieAdapter.notifyDataSetChanged();
}
});
recyclerNowPlaying = view.findViewById(R.id.recyclerNowPlaying);
recyclerNowPlaying.setLayoutManager(new LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, reverseLayout: false));
nowPlayingMovieAdapter = new MovieAdapter(userId,appDatabase,userRoleId);
recyclerNowPlaying.setAdapter(nowPlayingMovieAdapter);
Michał
MovieList.getReleasedMovies( accessToken: "eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI2NmI1OTA2OTU4ZDY0YjRmOWM1MjMzMzY0YiIsInN1YiI6IjY1O...
11 usages  Michał
@Override
public void onMoviesFetched(List<Movie> movies) {
    nowPlayingMovieList = movies;
    nowPlayingMovieAdapter.setMovies(nowPlayingMovieList);
    nowPlayingMovieAdapter.notifyDataSetChanged();
}
});
recyclerViewUpcoming = view.findViewById(R.id.recyclerViewUpcoming);
recyclerViewUpcoming.setLayoutManager(new LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, reverseLayout: false));
upcomingMovieAdapter = new MovieAdapter(userId,appDatabase,userRoleId);
recyclerViewUpcoming.setAdapter(upcomingMovieAdapter);
```

Rezultatem tego zapytania do API jest strona główna, na której wyświetlają się najlepiej oceniane filmy.

Definicja struktury bazy danych:

```
@Database(entities = {User.class, UserRole.class, Role.class, Movie.class, FavouriteMovies.class, WatchedMovies.class, MyListMovies.class})
public abstract class AppDatabase extends RoomDatabase {
6 usages  1 implementation  Michał Wojcik
    public abstract UserDao userDao();

    no usages  1 implementation  Michał Wojcik
    public abstract UserRoleDao userRoleDao();

    3 usages  1 implementation  Michał Wojcik
    public abstract RoleDao roleDao();

    no usages  1 implementation  Michał Wojcik
    public abstract MovieDao movieDao();

    5 usages  1 implementation  Michał Wojcik
    public abstract FavouriteMoviesDao favouriteMoviesDao();
5 usages  1 implementation  Michał
    public abstract WatchedMoviesDao watchedMoviesDao();
5 usages  1 implementation  Michał
    public abstract MyListMoviesDao myListMoviesDao();
3 usages  1 implementation  Michał
    public abstract ReviewDao reviewDao();
5 usages  1 implementation  Michał
    public abstract RatingDao ratingDao();
3 usages
    private static AppDatabase instance;
```

Michał

```
public static synchronized AppDatabase getInstance(Context context) {  
    if (instance == null) {  
        instance = Room.databaseBuilder(context.getApplicationContext(),  
            AppDatabase.class, name: "my-database")  
                .build();  
    }  
    return instance;  
}
```

Przykładowa funkcja wykorzystująca instancję bazy danych:

1 usage Michał Wojcik +1

```
private class InsertUserAsyncTask extends AsyncTask<User, Void, Void> {  
    Michał Wojcik  
    @Override  
    protected Void doInBackground(User... users) {  
        Log.d(tag: "InsertUserAsyncTask", msg: "doInBackground started");  
        try {  
            UserDao userDao = appDatabase.userDao();  
            userDao.insert(users[0]); // Insert the user into the database  
        } catch (Exception e) {  
            Log.e(tag: "InsertUserAsyncTask", msg: "Error inserting user", e);  
        }  
        Log.d(tag: "InsertUserAsyncTask", msg: "doInBackground finished");  
        return null;  
    }  
}
```

Przykładowy adapter (CategoryAdapter):

4 usages Michał Wojcik +1

```
public class CategoryAdapter extends RecyclerView.Adapter<CategoryAdapter.ViewHolder> {  
    3 usages  
    private List<Category> categoryList;  
    3 usages  
    private CategoryFragment.OnCategoryClickListener categoryClickListener;  
    1 usage   Michał +1  
    public CategoryAdapter(List<Category> categoryList, CategoryFragment.OnCategoryClickListener categoryClickListener) {  
        this.categoryList = categoryList;  
        this.categoryClickListener = categoryClickListener;  
    }  
  
    Michał Wojcik  
    @NonNull  
    @Override  
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_category, parent, attachToRoot: false);  
        return new ViewHolder(view);  
    }  
}
```

```

@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    Category category = categoryList.get(position);
    holder.categoryTextView.setText(category.getName());
    Michal
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        Michal
        @Override
        public void onClick(View v) {
            if (categoryClickListener != null) {
                categoryClickListener.onCategoryClick(category);
            }
        }
    });
}

```

Michal Wojcik

```

@Override
public int getItemCount() { return categoryList.size(); }

```

4 usages Michal Wojcik

```

public static class ViewHolder extends RecyclerView.ViewHolder {
    2 usages
    public TextView categoryTextView;

    1 usage Michal Wojcik
    public ViewHolder(@NonNull View itemView) {
        super(itemView);
        categoryTextView = itemView.findViewById(R.id.categoryTextView);
    }
}

```


Przykładowe activity (HomeActivity):

```
public class HomeActivity extends AppCompatActivity implements CategoryFragment.OnCategoryClickListener {  
    2 usages  
    private static final String KEY_CURRENT_INDEX = "currentIndex";  
  
    1 usage  
    private AppDatabase appDatabase;  
    10 usages  
    private SharedViewModel sharedViewModel;  
    2 usages  
    private String userName;  
    5 usages  
    private long userId;  
    3 usages  
    private long userRoleId;  
  
    👤 Bartłomiej Wojcik +2  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_home);  
        appDatabase = Room.databaseBuilder(getApplicationContext(), AppDatabase.class, "my-database")  
            .build();  
        Intent intent = getIntent();  
        if(intent != null){  
            userId = intent.getLongExtra("userId", defaultValue: 1);  
            userRoleId = intent.getLongExtra("userRoleId", defaultValue: 1);  
            Log.d("Logowanie", "Takie jest w Home Activity:"+userId);  
            userName = intent.getStringExtra("userName");  
        }  
    }  
}
```

```
sharedViewModel = new ViewModelProvider( owner: this).get(SharedViewModel.class);  
  
if (savedInstanceState == null) {  
    addMovieListFragment();  
} else {  
    int currentIndex = savedInstanceState.getInt(KEY_CURRENT_INDEX);  
    restoreFragment(currentIndex);  
}  
  
BottomNavigationView bottomNavigationView = findViewById(R.id.navigation_bar_item_icon_view);  
bottomNavigationView.setSelectedItemId(R.id.menu_home);  
  
👤 Bartłomiej Wojcik +1  
bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {  
    1 usage 👤 Bartłomiej Wojcik +1  
    @Override  
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
        int itemId = item.getItemId();  
        if (itemId == R.id.menu_category) {  
            addCategoryFragment();  
        } else if (itemId == R.id.menu_home) {  
            addMovieListFragment();  
        } else if (itemId == R.id.menu_profile) {  
            addProfileFragment();  
        }  
        return true;  
    }  
});
```

```

@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);

    outState.putInt(KEY_CURRENT_INDEX, sharedViewModel.getCurrentIndex());
}
2 usages  ⤴ Michal
@Override
public void onCategoryClick(Category category) {
    addCategoryMovies(category.getId(), category.getName());
}
3 usages  ⤴ Michal +1
private void addMovieListFragment() {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    MovieListFragment movieListFragment = new MovieListFragment();
    Bundle bundle = new Bundle();
    bundle.putLong("userId", userId);
    bundle.putLong("userRoleId", userRoleId);
    movieListFragment.setArguments(bundle);
    fragmentTransaction.replace(R.id.fragment_container, movieListFragment);
    fragmentTransaction.commit();
    sharedViewModel.setCurrentIndex(0);
}

```

```

private void addCategoryFragment() {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    CategoryFragment categoryFragment = new CategoryFragment();
    fragmentTransaction.replace(R.id.fragment_container, categoryFragment);
    fragmentTransaction.commit();
    sharedViewModel.setCurrentIndex(1);
}
2 usages  ⤴ Michal
private void addCategoryMovies(int categoryId, String name) {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    MovieListCategoryFragment movieListCategoryFragment = new MovieListCategoryFragment();
    Bundle bundle = new Bundle();
    bundle.putInt("categoryId", categoryId);
    bundle.putString("categoryName", name);
    bundle.putLong("userRoleId", userRoleId);
    bundle.putLong("userId", userId);
    movieListCategoryFragment.setArguments(bundle);
    fragmentTransaction.replace(R.id.fragment_container, movieListCategoryFragment);
    fragmentTransaction.commit();
    sharedViewModel.setCurrentIndex(2);
    sharedViewModel.setCurrentCategoryId(categoryId);
    sharedViewModel.setCurrentCategoryName(name);
}

```

```

private void addProfileFragment() {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    ProfileFragment profileFragment = new ProfileFragment();
    Bundle bundle = new Bundle();
    bundle.putString("userName", userName);
    bundle.putLong("userId", userId);
    profileFragment.setArguments(bundle);
    fragmentTransaction.replace(R.id.fragment_container, profileFragment);
    fragmentTransaction.commit();
    sharedViewModel.setCurrentIndex(3);
}
}
1 usage  ➤ Michal +1
private void restoreFragment(int index) {
    switch (index) {
        case 0:
            addMovieListFragment();
            break;
        case 1:
            addCategoryFragment();
            break;
        case 2:
            int categoryId = sharedViewModel.getCurrentCategoryId();
            String categoryName = sharedViewModel.getCurrentCategoryName();
            addCategoryMovies(categoryId, categoryName);
            break;
        case 3:
            addProfileFragment();
            break;
    }
}

```

Napotkane problemy:

1. Często psujący się emulator.
2. Dobór dependencies.
3. Zmiana layoutu z pionowego na poziomy (rozwiązanie z zastosowaniem funkcji `onConfigurationChanged()`).