



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

КУРСОВАЯ РАБОТА

по дисциплине «Архитектура операционных систем мобильных устройств»

Тема курсовой работы «»

Студент группы ИКБО-12-22

Солобай Александр Петрович

(подпись студента)



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Утверждаю

Заведующий кафедрой МОСИТ

_____ Головин С.А.

«___» _____ 2023 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	4
ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ	6
ПРОЕКТНАЯ ЧАСТЬ	13
ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ.....	16
ЗАКЛЮЧЕНИЕ	35
СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ	36

ВВЕДЕНИЕ

В современном информационном обществе, где цифровые технологии прочно вошли в повседневную жизнь, создание инновационных программных продуктов становится неотъемлемой частью развития. В контексте этой динамичной эпохи, домашняя медиатека, представляющая собой средство организации и управления мультимедийным контентом, приобретает особую актуальность.

Цель настоящей работы заключается в разработке и реализации программного комплекса, ориентированного на создание интуитивно понятного пользовательского интерфейса и обеспечение широкого спектра функциональных возможностей. Предлагаемый прототип мобильного приложения на платформе Android предполагает создание интерактивной системы, способной эффективно управлять медиаконтентом в домашних условиях.

Задачи работы включают в себя:

- Исследование предметной области разрабатываемой программной системы, анализ требований и формулирование технического задания в соответствии с ГОСТ 19.201-78.
- Проектирование и реализация ролевой модели безопасности для обеспечения конфиденциальности и целостности данных.
- Изучение жизненного цикла мобильных программ и компонентов, а также выбор и применение языков программирования высокого уровня, соответствующих целям и тематике курсовой работы.
- Разработка и внедрение визуальных элементов, сервисов и методов хранения данных в создаваемом программном комплексе.
- Тестирование и оптимизация разработанного приложения для обеспечения высокой производительности и удовлетворения потребностей пользователей.

Общая структура работы представляет собой последовательное выполнение данных задач, что позволит достичь поставленных целей и создать эффективный инструмент управления мультимедийным контентом в домашних условиях.

ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

Таблица 1 — Аналоги разрабатываемого приложения “Домашняя медиатека”

Приложение	ОС	Цена	Основной функционал	Плюсы	Минусы
Plex	Windows Mac Linux iOS Android	Бесплатно/ Платно	Организация и потоковая передача медиафайлов.	Поддерживает множество форматов медиафайлов. Возможность потоковой передачи на разные устройства Создание персонального сервера для доступа к медиа из любой точки мира.	Некоторые функции требуют подписки Plex Pass. Не всегда стабильная работа на некоторых устройствах. Для доступа к некоторым продвинутым функциям может потребоваться дополнительная настройка.
Kodi	Windows Mac Linux Android	Бесплатно	Мультимедийный центр с открытым исходным кодом.	Поддержка плагинов и различных тем оформления. Мощные настройки проигрывателя и библиотеки медиафайлов.	Не имеет встроенной функциональности для потоковой передачи медиафайлов. Некоторые пользователи могут считать интерфейс

				Гибкость настройки и расширяемос ть функционала.	сложным для начинающих. Требует наличия собственной библиотеки медиафайлов на устройстве, где установлен Kodi.
Emby	Windows Mac Linux iOS Android	Бесплатно/ Платно	Организация и потоковая передача медиафайлов.	Синхронизац ия между устройствами , медиа- сервер и клиентские приложения. Может использовать ся для доступа к медиа из дома или удаленно. Поддержка различных форматов медиафайлов.	Некоторые продвинутое функции доступны только в платной версии. Не всегда стабильная работа на некоторых устройствах требуется некоторое время на настройку и запуск медиа- сервера.
Jellyfin	Windows Mac Linux iOS Android	Бесплатно	Организация и потоковая передача медиафайлов.	Проект с открытым исходным кодом. Поддержка плагинов и	- Интерфейс и функционально сть могут показаться менее полными по сравнению с

				<p>широкий спектр функций. Возможность создания персонального медиа-сервера. Синхронизация и транскодирование медиафайлов.</p>	<p>коммерческими аналогами. Некоторые функции могут работать не столь стабильно или требуют дополнительных настройки. Поддержка может быть ограничена в сравнении с более популярными альтернативами.</p>
Infuse	iOS tvOS	Бесплатно/ Платно	Проигрыватель медиафайлов с поддержкой сетевых хранилищ.	<p>Интуитивный интерфейс и поддержка множества форматов медиафайлов. Поддержка сетевых хранилищ, включая облачные сервисы. Богатые функции, такие как автоматическ</p>	<p>Ограниченная поддержка платформ (только iOS и tvOS). Некоторые продвинутые функции доступны только в платной версии. Не подходит для пользователей, которые хотят</p>

				ая метаданные и трейлеры. Синхронизац ия просмотра через iCloud.	централизован ную библиотеку медиафайлов на сервере.
VLC	Windows Mac Linux iOS Android	Бесплатно	Медиаплеер с открытым исходным кодом.	Поддержка широкого спектра форматов медиафайлов и потоковое воспроизведе ние. Простой в использовани и не требует настройки сервера. Бесплатное и открытое ПО.	Интерфейс не всегда настолько интуитивен, как у других приложений. Ограниченные функции потоковой передачи и организации библиотеки медиафайлов по сравнению с специализиров анными медиа- серверами. Не поддерживает функции синхронизации между устройствами.
Serviio	Windows Mac Linux	Бесплатно/ Платно	Медиа-сервер для потоковой передачи медиафайлов.	Поддержка множества устройств и форматов медиафайлов.	Некоторые продвинутое функции доступны только в

				Транскодирование видео и аудио в реальном времени. Прост в настройке и поддерживает потоковую передачу видео, аудио и изображений на различные устройства в сети.	платной версии. Не так много расширений и дополнительных функций, как у некоторых других медиа-серверов. Интерфейс может показаться не таким интуитивным для новичков.
Universal Media Server	Windows Mac Linux	Бесплатно	Медиа-сервер с открытым исходным кодом.	Поддержка большого количества устройств и форматов медиафайлов. Прост в настройке и поддерживает потоковую передачу видео, аудио и изображений на различные устройства в сети.	Не так много продвинутых функций и настроек, как у некоторых других медиа-серверов. Интерфейс может показаться не таким интуитивным для новичков. Не всегда самое активное обновление и поддержка сообщества в

				Бесплатное и открытое ПО.	сравнении с другими проектами.
--	--	--	--	------------------------------	--------------------------------------

Аналоги Android Studio для разработки мобильных приложений:



1. IntelliJ IDEA: это интегрированная среда разработки (IDE), разработанная компанией JetBrains. Она предоставляет широкий спектр функций для разработки Android-приложений, включая умный редактор кода, поддержку языков программирования Kotlin и Java, а также инструменты для тестирования и отладки.



2. Eclipse: это одна из самых популярных IDE для разработки программного обеспечения, включая приложения для Android. Eclipse предлагает широкий набор плагинов и инструментов для разработки, включая поддержку для языков Java и Kotlin, систему управления версиями, отладчик и многие другие.



3. Visual Studio Code (VS Code): хотя изначально разработан для веб-разработки, VS Code также имеет расширения для разработки приложений Android. Он обладает высокой настраиваемостью, легкостью использования и богатым функционалом, включая редактор кода с подсветкой синтаксиса, отладчик, систему управления версиями и многое другое.



4. NetBeans: еще одна мощная и гибкая IDE, которая поддерживает разработку Android-приложений с использованием языков Java и Kotlin. NetBeans предлагает широкий набор инструментов для разработки, включая поддержку множества плагинов и расширений.



5. Xcode: это интегрированная среда разработки (IDE), разработанная Apple для создания приложений для их операционных систем, таких как iOS, macOS, watchOS и tvOS. Хотя Xcode преимущественно ориентирован на разработку приложений для устройств Apple, таких как iPhone и iPad, его можно использовать и для создания приложений для Android через некоторые дополнительные инструменты и плагины. Xcode предоставляет широкий набор инструментов, включая редактор кода, интерфейс для проектирования пользовательского интерфейса, отладчик, профилировщик и многое другое.

ПРОЕКТНАЯ ЧАСТЬ

Для разработки мобильного приложения по теме курсовой работы была выбрана Android Studio.

Android Studio — это официальная интегрированная среда разработки (IDE) для создания приложений под операционную систему Android. Ее выбор для разработки мобильного приложения обосновывается несколькими преимуществами:

- **Официальная поддержка:** Android Studio разрабатывается и поддерживается напрямую командой Google, что обеспечивает соответствие последним требованиям и стандартам платформы Android.
- **Интеграция с Android SDK:** Android Studio включает в себя все необходимые инструменты для разработки под Android, включая компиляторы, отладчики, эмуляторы и т. д.
- **Удобство разработки интерфейса:** С помощью Android Studio удобно создавать пользовательские интерфейсы благодаря интуитивно понятному редактору макетов.
- **Мощный отладчик:** Инструменты отладки в Android Studio позволяют быстро находить и исправлять ошибки в приложении.
- **Поддержка Google Play Services:** Android Studio легко интегрируется с сервисами Google Play, такими как Google Maps, Firebase, аналитика и другие.
- **Активное сообщество и обширная документация:** Использование Android Studio поддерживается обширным сообществом разработчиков Android, что облегчает получение помощи и решение проблем.

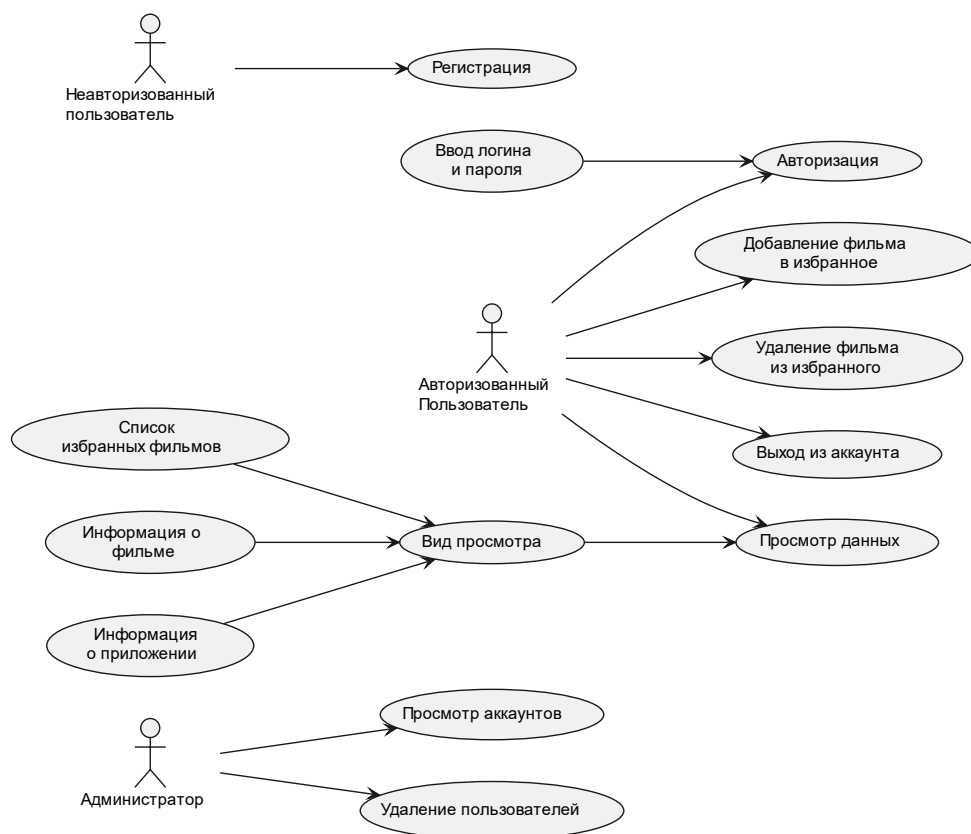


Рисунок 1 – диаграмма вариантов использования для разрабатываемого приложения

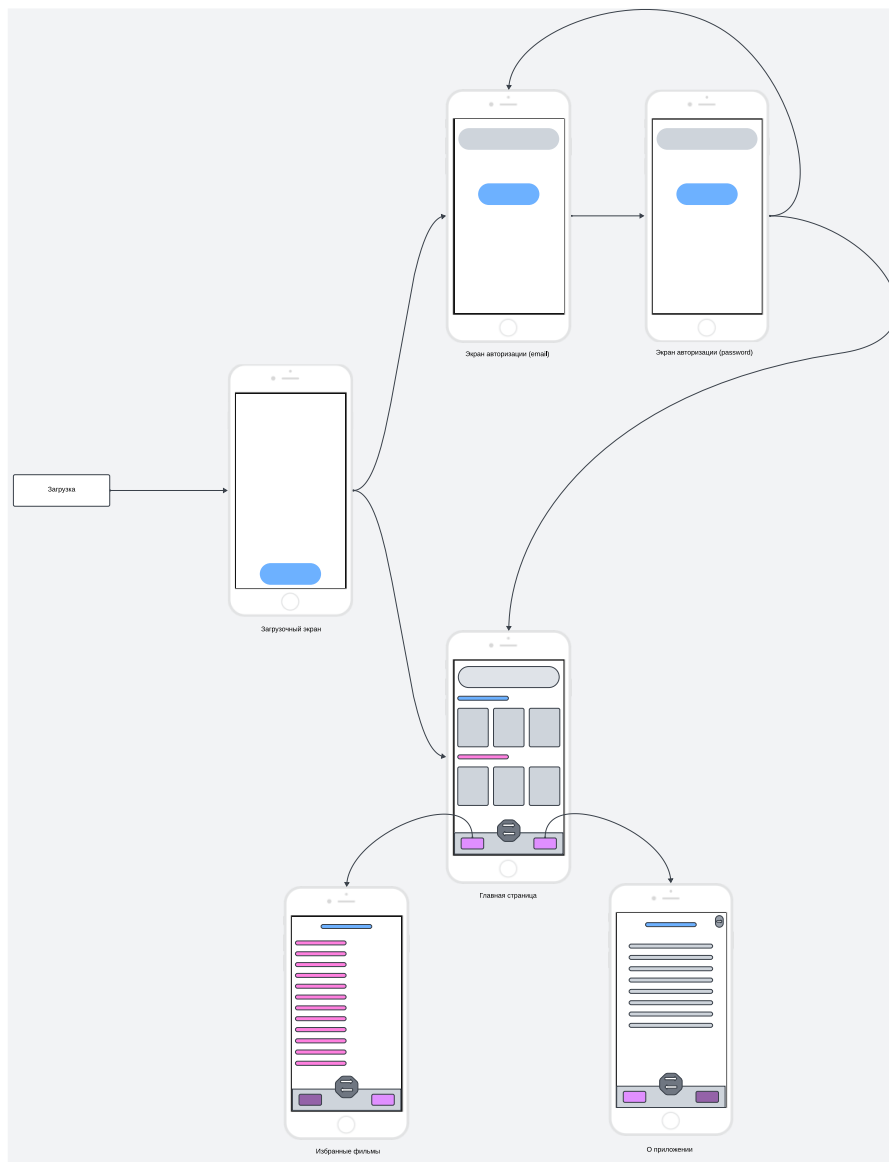


Рисунок 2 – проектирование пользовательского интерфейса

ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

Для работы с главным экраном нужно сгенерировать Java-объекты из JSON.[7]

The screenshot shows the 'jsonschema2pojo' web application. At the top, it says 'Generate Plain Old Java Objects from JSON or JSON-Schema.' Below this, there are input fields for 'Package' (com.example.coursach.Domain) and 'Class name' (FilmItem). The main area displays a JSON schema on the left and configuration options on the right. The JSON schema is for a movie, with fields like id, title, poster, year, rated, released, runtime, director, writer, actors, plot, country, awards, metascore, imdb_rating, imdb_votes, imdb_id, type, genres, and images. The configuration options on the right include 'Source type' (JSON selected), 'Annotation style' (Gson selected), 'Validation annotations' (None selected), and checkboxes for 'Generate builder methods', 'Use primitive types', and 'Use long integers'.

```
1 {
2   "id":1,
3   "title":"The Shawshank Redemption",
4   "poster":"tt0111161_poster.jpg",
5   "year":"1994",
6   "rated":"R",
7   "released":"14 Oct 1994",
8   "runtime":"142 min",
9   "director":"Frank Darabont",
10  "writer":"Stephen King (short story) \"Rita Hayworth and Shawshank\"",
11  "actors":"Tim Robbins, Morgan Freeman, Bob Gunton, William Sadle",
12  "plot":"Two imprisoned men bond over a number of years, finding",
13  "country":"USA",
14  "awards":"Nominated for 7 Oscars. Another 19 wins & 30 nomin",
15  "metascore":"80",
16  "imdb_rating":"9.3",
17  "imdb_votes":"1,738,596",
18  "imdb_id":"tt0111161",
19  "type":"movie",
20  "genres":[
21    "Crime",
22    "Drama"
23  ],
24  "images":[
25    "http://moviesapi.ir/images/tt0111161_screenshot1.jpg",
26    "http://moviesapi.ir/images/tt0111161_screenshot2.jpg",
27    "http://moviesapi.ir/images/tt0111161_screenshot3.jpg"
28  ]
29 }
```

Source type:

☐ JSON Schema ☒ JSON

☐ YAML Schema ☐ YAML

Annotation style:

☐ Jackson 2.x ☒ Gson ☐ Moshi

☐ JSON-B 1.x ☐ JSON-B 2.x ☐ None

Validation annotations:

☐ javax.validation

☐ jakarta.validation

☒ None

☐ Generate builder methods

☐ Use primitive types

☐ Use long integers

Рисунок 3 – JSON с конкретным фильмом



Рисунок 4 – полученный Java-объект

Те же действия нужно проделать со списком фильмов.

Напишем IntroFragment, который будем показываться неавторизованным пользователям при первом запуске нашего приложения.

Листинг 1 – IntroFragment.java

```

package com.example.coursach.Fragments;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.Network;
import android.net.NetworkCapabilities;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

```

```

import androidx.fragment.app.Fragment;
import androidx.navigation.Navigation;

import com.example.coursach.R;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class IntroFragment extends Fragment {

    @SuppressWarnings("SetTextI18n")
    private void showCustomSnackbar() {
        Snackbar snackbar = Snackbar.make(requireView(), "",
Snackbar.LENGTH_LONG);
        @SuppressWarnings("RestrictedApi") Snackbar.SnackbarLayout layout
= (Snackbar.SnackbarLayout) snackbar.getView();

        LayoutInflater inflater =
LayoutInflater.from(requireContext());
        @SuppressWarnings("InflateParams") View customView =
inflater.inflate(R.layout.custom_snackbar, null);

        TextView textView =
customView.findViewById(R.id.snackbar_text);
        textView.setText("Internet connection is required");

        layout.setPadding(0, 0, 0, 0);
        layout.addView(customView, 0);

        snackbar.show();
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_intro, container,
false);
    }
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        if (checkUserStatus()) {
Navigation.findNavController(view).navigate(R.id.action_introFragment
_to_mainFragment);
        } else {
            setupStartButton(view);
        }
    }

    private void setupStartButton(View view) {
        Button getStartBtn = view.findViewById(R.id.getStart);

        getStartBtn.setOnClickListener(v -> {

```

```

        if (isNetworkAvailable()) {
Navigation.findNavController(requireView()).navigate(R.id.action_introFragment_to_loginFragment);
        } else {
            showCustomSnackbar();
        }
    });
}

private boolean checkUserStatus() {
    FirebaseAuth currentUser =
FirebaseAuth.getInstance().getCurrentUser();
    return currentUser != null && currentUser.isEmailVerified()
&& isUserLoggedIn();
}

private boolean isUserLoggedIn() {
    SharedPreferences sharedPreferences =
requireContext().getSharedPreferences("user_auth",
Context.MODE_PRIVATE);
    return sharedPreferences.contains("userId");
}

private boolean isNetworkAvailable() {
    ConnectivityManager connectivityManager =
(ConnectivityManager)
requireActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
    if (connectivityManager != null) {
        Network network = connectivityManager.getActiveNetwork();
        if (network != null) {
            NetworkCapabilities networkCapabilities =
connectivityManager.getNetworkCapabilities(network);
            return networkCapabilities != null &&
(networkCapabilities.hasTransport(NetworkCapabilities.TRANSPORT_WIFI)
||
networkCapabilities.hasTransport(NetworkCapabilities.TRANSPORT_CELLUL
AR) ||
networkCapabilities.hasTransport(NetworkCapabilities.TRANSPORT_ETHERN
ET));
        }
    }
    return false;
}
}

```

Теперь пользователей будет встречать стартовый экран с кнопкой “Начать”.

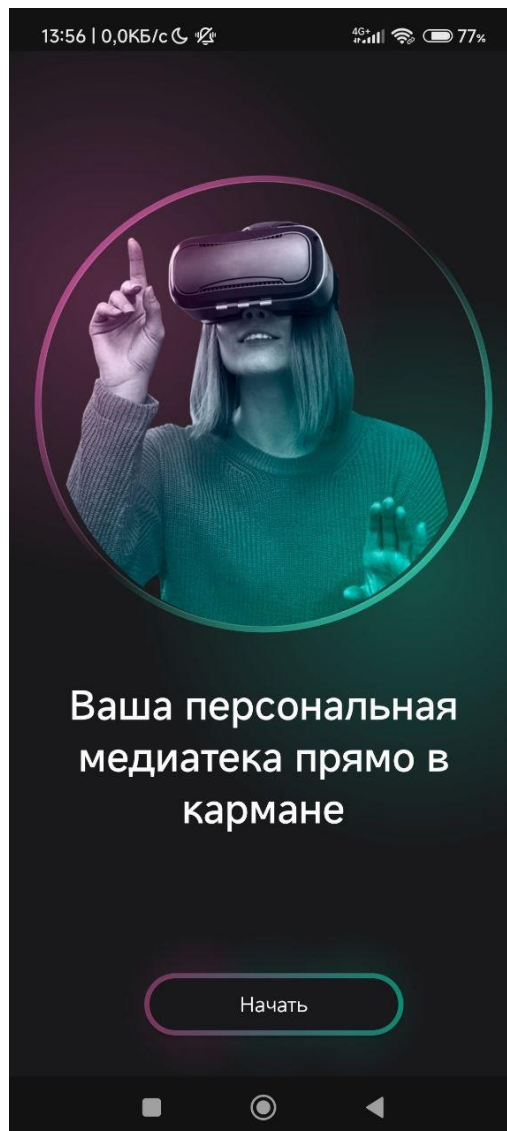


Рисунок 5 – стартовый экран

Теперь нужно написать авторизацию пользователей в приложении с помощью ссылки с подтверждением, которая будет отправляться на введенную почту.

В курсовой работе будет использоваться NoSQL база данных Firebase, которая предоставляет возможность авторизации через почту.

Листинг 2 – фрагмент кода файла LoginFragment.java с проверкой на корректность почты и переходом к экрану ввода пароля

```
private boolean isValidEmail(String email) {  
    String emailPattern = getString(R.string.pattern);  
    return email.matches(emailPattern);  
}  
  
private void attemptLogin() {  
    String email = editTextEmail.getText().toString().trim();
```

```

        if (email.isEmpty()) {
            showCustomSnackbar(getString(R.string.emailinp));
            return;
        }

        if (!isEmailValid(email)) {
            showCustomSnackbar(getString(R.string.erremail));
            return;
        }

        mAuth.fetchSignInMethodsForEmail(email).addOnCompleteListener(task ->
        {
            boolean isNewUser =
            Objects.requireNonNull(task.getResult().getSignInMethods()).isEmpty()
            ;
            if (!isNewUser) {

                Navigation.findNavController(requireView()).navigate(R.id.action_loginFragment_to_mainFragment);
            } else {
                LoginFragmentDirections.ActionLoginFragmentToPassFragment
                direction =
                LoginFragmentDirections.actionLoginFragmentToPassFragment(email);

                Navigation.findNavController(requireView()).navigate(direction);
            }
        }).addOnFailureListener(e ->
        showCustomSnackbar(getString(R.string.errenter)));
    }

```

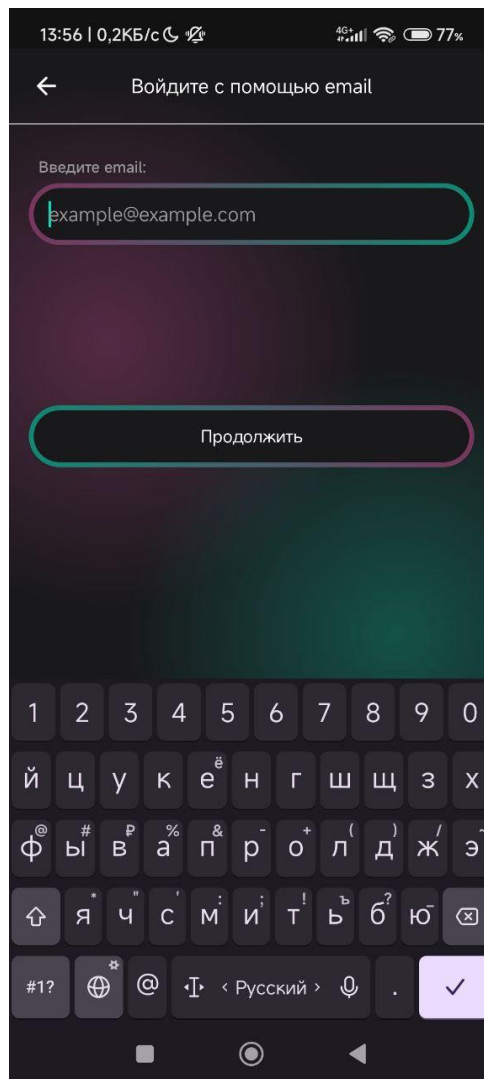


Рисунок 6 – экран ввода почты

Листинг 3 – фрагмент кода файла PassFragment.java с отправкой письма для подтверждения почты и регистрацией пользователя

```
private void registerUser() {
    String password = passEdt.getText().toString().trim();

    if (password.isEmpty()) {
        showCustomSnackbar("Пожалуйста, введите пароль.");
        return;
    }

    mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            FirebaseUser user = mAuth.getCurrentUser();
            if (user != null) {
                saveUserAuthState(user.getUid());
            }
        }
    });

    Navigation.findNavController(requireView()).navigate(R.id.action_pass
Fragment_to_mainFragment);
}
```

```

        } else {
            mAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(registerTask -> {
                if (registerTask.isSuccessful()) {
                    FirebaseAuth newUser = mAuth.getCurrentUser();
                    if (newUser != null) {
                        saveUserAuthState(newUser.getId());
                        sendEmailVerification();
                    }
                } else {
                    showCustomSnackbar(GetString(R.string.errreg));
                }
            }).addOnFailureListener(e ->
showCustomSnackbar(GetString(R.string.errregexist)));
        });
    }

private void sendEmailVerification() {
    FirebaseAuth user = mAuth.getCurrentUser();
    if (user != null) {
        user.sendEmailVerification().addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                showCustomSnackbar(GetString(R.string.emailversend));
            }
        });
        Navigation.findNavController(requireView()).navigate(R.id.action_pass
Fragment_to_mainFragment);
    } else {
        showCustomSnackbar(GetString(R.string.erremailversend));
    }
}

    } else {
        showCustomSnackbar(GetString(R.string.errnotexist));
    }
}
}

```

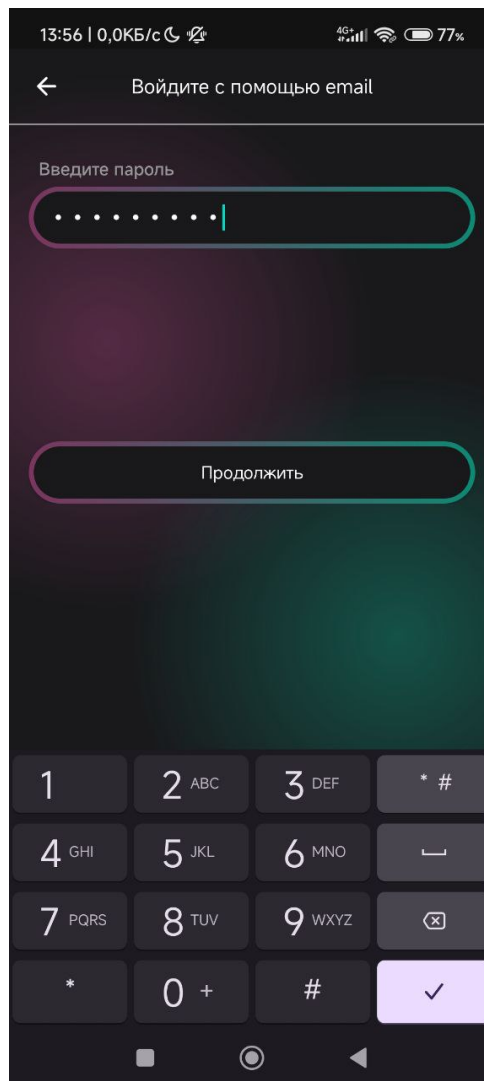


Рисунок 7 – экран ввода пароля

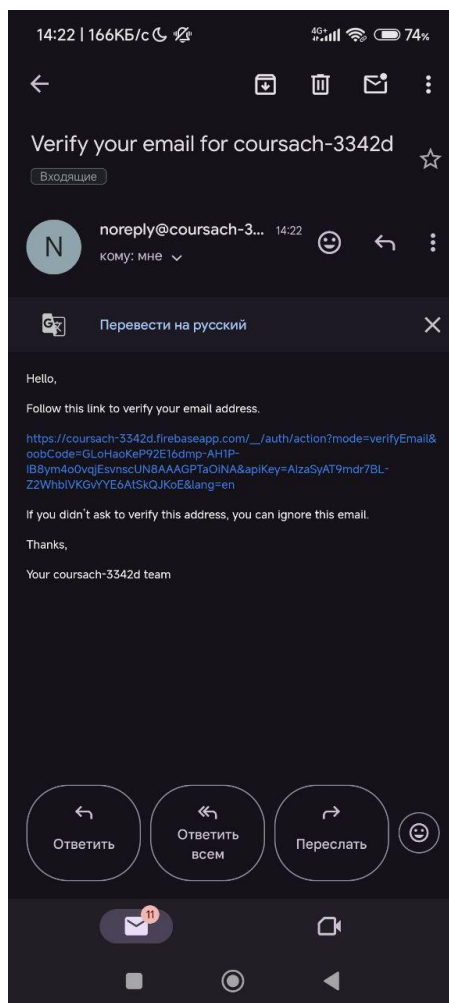


Рисунок 8 – письмо с подтверждением почты

После успешной авторизации данные пользователя сохранятся и при последующих открытиях приложения он сразу будет попадать на главный экран с фильмами.

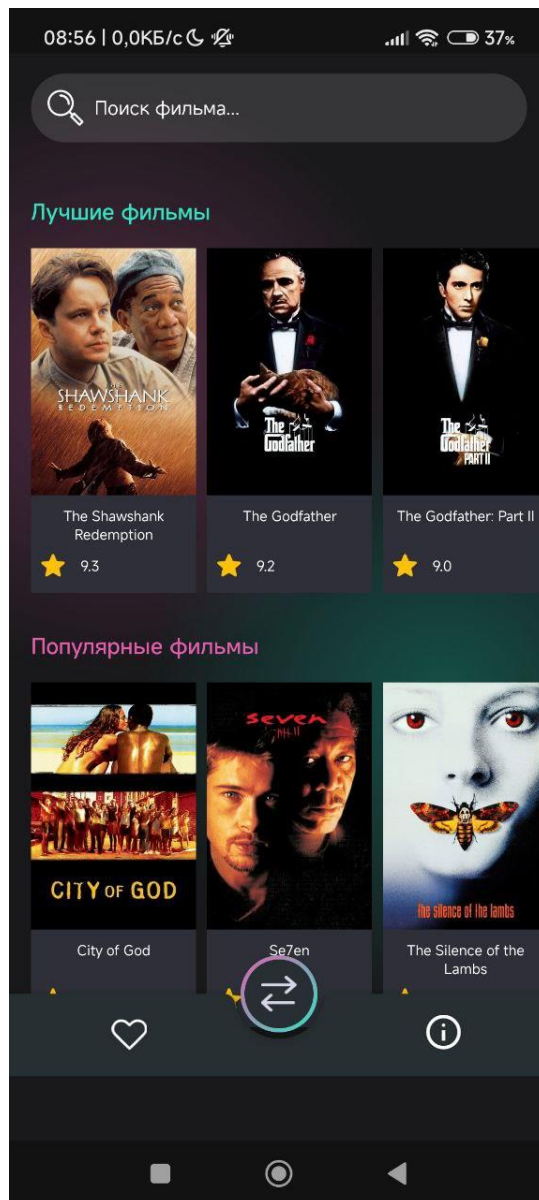


Рисунок 9 – главный экран

Листинг 4 – фрагмент кода файла *MainFragment.java* с функцией поиска фильмов

```
private void searchMovies(String query) {
    String searchUrl = "https://moviesapi.ir/api/v1/movies?q=" +
    query;
    loading1.setVisibility(View.VISIBLE);
    StringRequest mStringRequest = new
    StringRequest(Request.Method.GET, searchUrl, response -> {
        Gson gson = new Gson();
        loading1.setVisibility(View.GONE);

        ListFilm items = gson.fromJson(response, ListFilm.class);
        NavController navController =
        Navigation.findNavController(requireActivity(),
        R.id.nav_host_fragment);
        FilmListAdapter adapter = new FilmListAdapter(items,
        navController);
        recyclerViewNewMovies.setAdapter(adapter);
    });
}
```

```
}, error -> loading1.setVisibility(View.GONE));  
mRequestQueue.add(mStringRequest);  
}
```

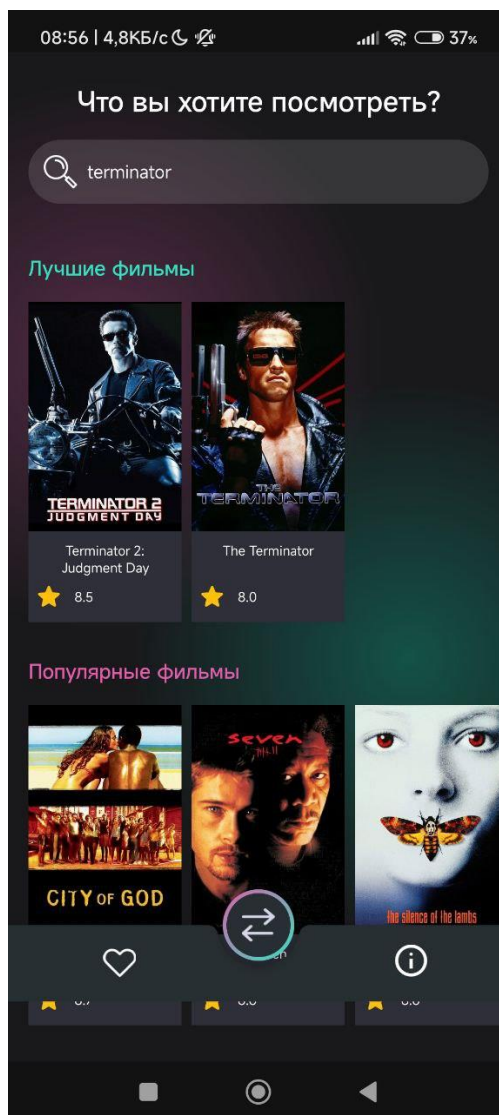


Рисунок 10 – поиск фильмов

С главного экрана пользователь может перейти к списку избранных фильмов или посмотреть информацию о приложении, а также выполнить выход из аккаунта. При нажатии на постер с фильмом пользователь может посмотреть детальную информацию о нем.

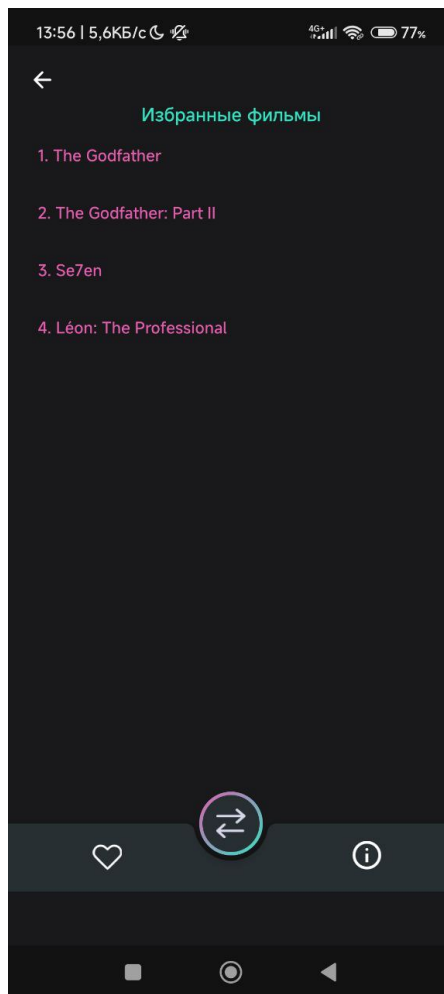


Рисунок 11 – избранные фильмы

При нажатии на элемент из списка пользователя будет перекидывать на экран с детальной информацией об избранном фильме.

Листинг 5 – фрагмент кода файла DetailFragment.java с методами добавления/удаления фильма из избранного

```
private void toggleFavorite(FilmItem film) {
    if (film == null) {
        showCustomSnackbar(getString(R.string.infounav));
        return;
    }

    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    if (user == null) {
        Log.d("Authentication", "Пользователь не аутентифицирован.");
        return;
    }

    String userId = user.getId();
    DatabaseReference mDatabase =
        FirebaseDatabase.getInstance(getResources().getString(R.string.url)).
        getReference().child("users").child(userId).child("favorites").child(
            getString(film.getId()));
```

```

        SharedPreferences prefs =
requireActivity().getSharedPreferences("Favorites",
Context.MODE_PRIVATE);
        boolean isCurrentlyFavorite = prefs.getBoolean("Film_" +
film.getId(), false);

        if (isCurrentlyFavorite) {
            mDatabase.removeValue().addOnSuccessListener(aVoid -> {
                showCustomSnackbar(GetString(R.string.delfav));
                saveFavoriteStatus(film.getId(), false);
                updateFavoriteButtonBackground(false);
            }).addOnFailureListener(e ->
showCustomSnackbar(GetString(R.string.errdelfav)));
        } else {
            mDatabase.setValue(film).addOnSuccessListener(aVoid -> {
                showCustomSnackbar(GetString(R.string.addfav));
                saveFavoriteStatus(film.getId(), true);
                updateFavoriteButtonBackground(true);
            }).addOnFailureListener(e ->
showCustomSnackbar(GetString(R.string.erraddfav)));
        }
    }

    private void saveFavoriteStatus(int filmId, boolean isFavorite) {
        SharedPreferences prefs =
requireActivity().getSharedPreferences("Favorites",
Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
        editor.putBoolean("Film_" + filmId, isFavorite);
        editor.apply();
    }

    private void updateFavoriteButtonBackground(boolean isFavorite) {
        if (isFavorite) {
            favoriteImg.setBackgroundResource(R.drawable.bg_circle_pink);
        } else {
            favoriteImg.setBackgroundResource(R.drawable.bg_circle_dark);
        }
    }

    private void checkFavoriteStatus() {
        SharedPreferences prefs =
requireActivity().getSharedPreferences("Favorites",
Context.MODE_PRIVATE);
        boolean isFavorite = prefs.getBoolean("Film_" + idFilm, false);
        updateFavoriteButtonBackground(isFavorite);
    }

```

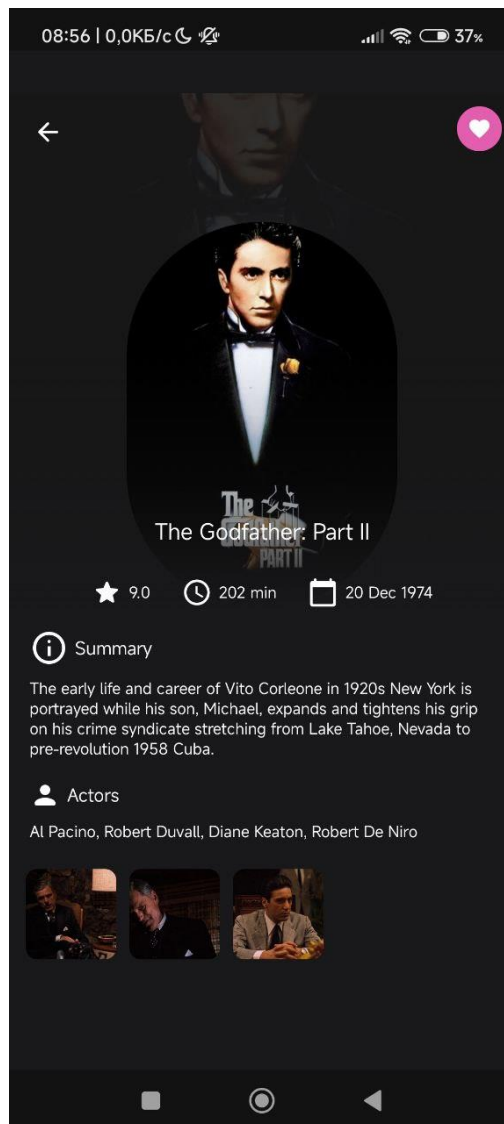


Рисунок 12 – детальная информация о фильме

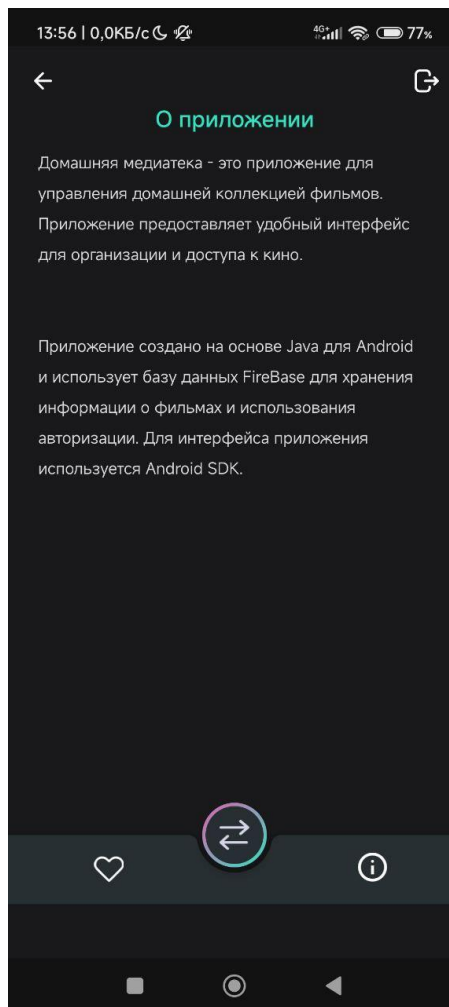


Рисунок 13 – раздел с информацией о приложении и кнопкой выхода из аккаунта

Переход между экранами осуществлен с помощью Navigation, сами экраны реализованы с помощью Fragments вместо Activities.

Fragments — это модульные компоненты в Android, которые представляют часть пользовательского интерфейса в Activity. Они предоставляют большую гибкость при организации и повторном использовании пользовательских интерфейсов в различных частях приложения.

Преимущества использования Fragments над Activity:

- Модульность. Фрагменты могут быть переиспользованы в различных активити, что облегчает повторное использование кода.

- Адаптивный интерфейс. С помощью фрагментов можно легко создавать адаптивные макеты, которые работают как на телефонах, так и на планшетах.
- Более гранулированное управление жизненным циклом. Фрагменты обладают собственным жизненным циклом, который тесно связан с жизненным циклом их хост-активности, позволяя более тонко управлять ресурсами.
- Лучшая поддержка многопанельных интерфейсов. Используя фрагменты, можно легко реализовать сложные пользовательские интерфейсы с несколькими панелями, как, например, в мастер-детальных интерфейсах.

Navigation в Android представляет собой компонент, который помогает организовать переходы между различными частями приложения (как правило, между различными фрагментами). Это достигается с помощью Navigation Graph, который является XML-файлом, описывающим все возможные пути навигации в приложении.

Преимущества использования Navigation:

- Упрощение управления стеком фрагментов. Navigation автоматически обрабатывает сложности управления стеком фрагментов, что упрощает переходы между экранами.
- Визуальное редактирование. Используя Navigation Editor в Android Studio, разработчик может визуальным образом конструировать навигационную структуру приложения, что делает процесс более интуитивно понятным.
- Улучшенное взаимодействие с архитектурными компонентами. Navigation работает отлично с другими компонентами Jetpack, такими как ViewModel и LiveData, обеспечивая лучшую интеграцию и производительность.

- Обработка глубоких ссылок. Navigation упрощает реализацию глубоких ссылок, которые направляют пользователя на конкретный экран в приложении, даже если приложение не запущено.

Листинг 6 – nav_graph.xml

```
<?xml version="1.0" encoding="utf-8"?>
<navigation
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/nav_graph"
app:startDestination="@id/introFragment">

    <fragment
        android:id="@+id/introFragment"
        android:name="com.example.coursach.Fragments.IntroFragment"
        android:label="fragment_intro"
        tools:layout="@layout/fragment_intro" >
        <action
            android:id="@+id/action_introFragment_to_loginFragment"
            app:destination="@id/mainFragment" />
        <action
            android:id="@+id/action_introFragment_to_mainFragment"
            app:destination="@id/loginFragment" />
    </fragment>

    <fragment
        android:id="@+id/loginFragment"
        android:name="com.example.coursach.Fragments.LoginFragment"
        android:label="fragment_login"
        tools:layout="@layout/fragment_login">
        <action
            android:id="@+id/action_loginFragment_to_mainFragment"
            app:destination="@id/mainFragment" />
        <action
            android:id="@+id/action_loginFragment_to_passFragment"
            app:destination="@id/passFragment" />
        <action
            android:id="@+id/action_loginFragment_to_introFragment"
            app:destination="@id/introFragment" />
    </fragment>

    <fragment
        android:id="@+id/likedFragment"
        android:name="com.example.coursach.Fragments.LikedFragment"
        android:label="fragment_liked"
        tools:layout="@layout/fragment_liked">
        <action
            android:id="@+id/action_likedFragment_to_detailFragment"
            app:destination="@id/detailFragment"
            app:enterAnim="@anim/nav_default_enter_anim"
            app:exitAnim="@anim/nav_default_exit_anim"
            app:popEnterAnim="@anim/nav_default_pop_enter_anim"
```

```

        app:popExitAnim="@anim/nav_default_pop_exit_anim">
        <argument
            android:name="id"
            app:argType="integer" />
    </action>
</fragment>
<fragment
    android:id="@+id/descriptionFragment"
    android:name="com.example.coursach.Fragments.DescriptionFragment"
    android:label="fragment_seen"
    tools:layout="@layout/fragment_description" />

<fragment
    android:id="@+id/passFragment"
    android:name="com.example.coursach.Fragments.PassFragment"
    android:label="fragment_register"
    tools:layout="@layout/fragment_pass">
    <argument
        android:name="email"
        app:argType="string"/>
    <action
        android:id="@+id/action_passFragment_to_mainFragment"
        app:destination="@id/loginFragment" />
</fragment>

<fragment
    android:id="@+id/mainFragment"
    android:name="com.example.coursach.Fragments.MainFragment"
    android:label="Main Fragment"
    tools:layout="@layout/fragment_main">
    <action
        android:id="@+id/action_mainFragment_to_detailFragment"
        app:destination="@id/detailFragment" />
    <action
        android:id="@+id/action_mainFragment_to_loginFragment"
        app:destination="@id/loginFragment" />
</fragment>

<fragment
    android:id="@+id/detailFragment"
    android:name="com.example.coursach.Fragments.DetailFragment"
    android:label="fragment_detail"
    tools:layout="@layout/fragment_detail">
    <argument
        android:name="id"
        android:defaultValue="0"
        app:argType="integer" />
</fragment>
</navigation>

```

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы был проведен полный цикл разработки программного комплекса – домашней медиатеки для мобильной платформы Android. Актуальность данного проекта подтверждается постоянным ростом интереса к мультимедийному контенту и необходимостью их организации и управления в современных домашних условиях.

В рамках работы были успешно реализованы все поставленные цели и задачи, включая анализ предметной области, проектирование безопасной ролевой модели, выбор и применение соответствующих языков программирования, а также разработку и оптимизацию пользовательского интерфейса и функциональности приложения.

Важным результатом является создание интуитивно понятного и функционального программного комплекса, способного удовлетворить потребности пользователей в организации и управлении мультимедийным контентом в домашних условиях.

Таким образом, разработанный программный комплекс представляет собой важный шаг в области создания инновационных решений для организации и управления мультимедийным контентом в домашних условиях, и может быть успешно применен в практических задачах повседневной жизни.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Разработка мобильных приложений (КР/КП). [Электронный ресурс]. URL: <https://online-edu.mirea.ru/course/view.php?id=7400> (дата обращения 02.03.2024).
2. Разработка мобильных приложений (часть 1/1). [Электронный ресурс]. URL: <https://online-edu.mirea.ru/course/view.php?id=6592> (дата обращения 05.03.2024).
3. Navigation. — Текст: электронный // Developers: [сайт]. — URL: <https://developer.android.com/guide/navigation> (дата обращения: 30.04.2024).
4. Fragments. — Текст: электронный // Developers: [сайт]. — URL: <https://developer.android.com/guide/fragments> (дата обращения: 30.04.2024).
5. How to make gradient background in android. — Текст: электронный // stackoverflow: [сайт]. — URL: <https://stackoverflow.com/questions/13929877/how-to-make-gradient-background-in-android> (дата обращения: 30.04.2024).
6. Add a floating action button. — Текст: электронный // Developers: [сайт]. — URL: <https://developer.android.com/develop/ui/views/components/floating-action-button> (дата обращения: 30.04.2024).
7. Generate Plain Old Java Objects from JSON or JSON-Schema.. — Текст : электронный // jsonschema2pojo : [сайт]. — URL: <https://www.jsonschema2pojo.org/> (дата обращения: 06.05.2024).