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

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 [1].

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 [1].

Задачи работы включают в себя:

- Исследование предметной области разрабатываемой программной системы, анализ требований и формулирование технического задания в соответствии с ГОСТ 19.201-78 [2].
- Проектирование и реализация ролевой модели безопасности для обеспечения конфиденциальности и целостности данных [2].
- Изучение жизненного цикла мобильных программ и компонентов, а также выбор и применение языков программирования высокого уровня, соответствующих целям и тематике курсовой работы [2].
- Разработка и внедрение визуальных элементов, сервисов и методов хранения данных в создаваемом программном комплексе [2].
- Тестирование и оптимизация разработанного приложения для обеспечения высокой производительности и удовлетворения потребностей пользователей [2].

Общая структура работы представляет собой последовательное выполнение данных задач, что позволит достичь поставленных целей и создать эффективный инструмент управления мультимедийным контентом в домашних условиях [2].

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 расширяемость 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.	Синхронизация между устройствами, медиа-сервер и клиентские приложения. Может использоваться для доступа к медиа из дома или удаленно. Поддержка различных форматов медиафайлов.	Некоторые продвинутые 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 функциональность могут показаться менее полными по сравнению с

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

				ая метаданные и трейлеры. Синхронизация просмотра через iCloud.	централизованную библиотеку медиафайлов на сервере.
VLC	Windows Mac Linux iOS Android	Бесплатно	Медиаплеер с открытым исходным кодом.	Поддержка широкого спектра форматов медиафайлов и потоковое воспроизведение. Простой в использовании 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 специализированными 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 продвинутые 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 [3].

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].

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 [3].

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 [3].

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 [3].

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 [4]:

- Официальная поддержка: Android Studio разрабатывается и поддерживается напрямую командой Google, что обеспечивает соответствие последним требованиям и стандартам платформы Android [4].
- Интеграция с Android SDK: Android Studio включает в себя все необходимые инструменты для разработки под Android, включая компиляторы, отладчики, эмуляторы и т. Д [4].
- Удобство разработки интерфейса: С помощью Android Studio удобно создавать пользовательские интерфейсы благодаря интуитивно понятному редактору макетов [4].
- Мощный отладчик: Инструменты отладки в Android Studio позволяют быстро находить и исправлять ошибки в приложении [4].
- Поддержка Google Play Services: Android Studio легко интегрируется с сервисами Google Play, такими как Google Maps, Firebase, аналитика и другие [4].
- Активное сообщество и обширная документация: Использование Android Studio поддерживается обширным сообществом разработчиков Android, что облегчает получение помощи и решение проблем [4].

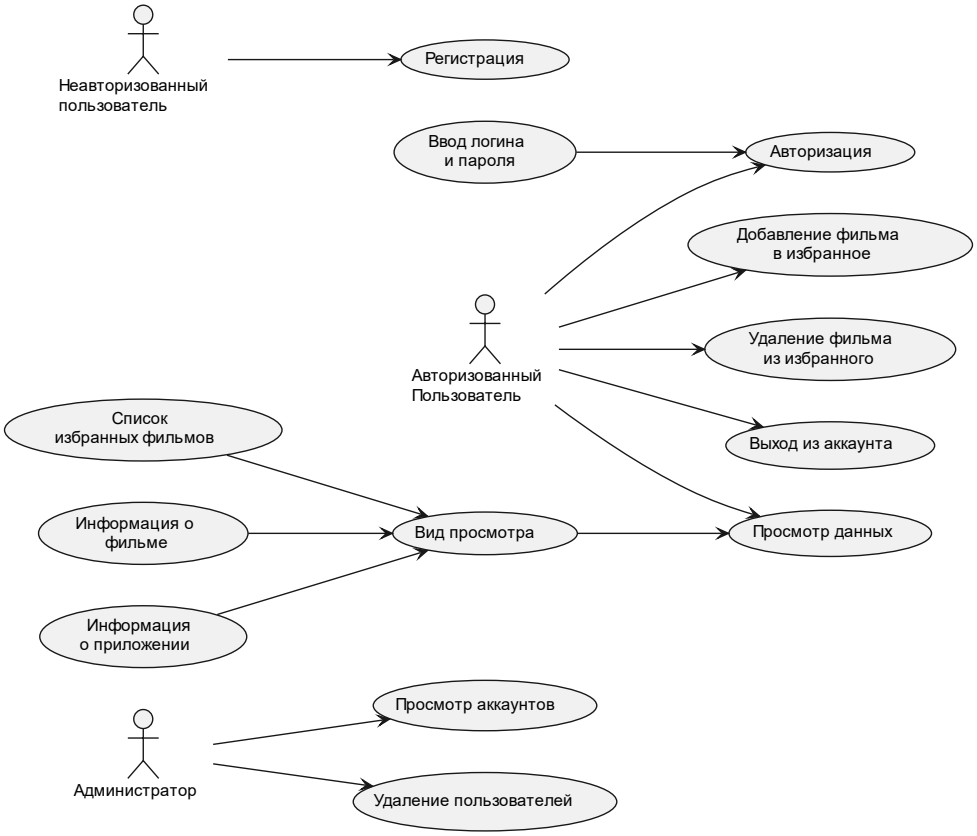

Рисунок 1 – диаграмма вариантов использования для разрабатываемого приложения

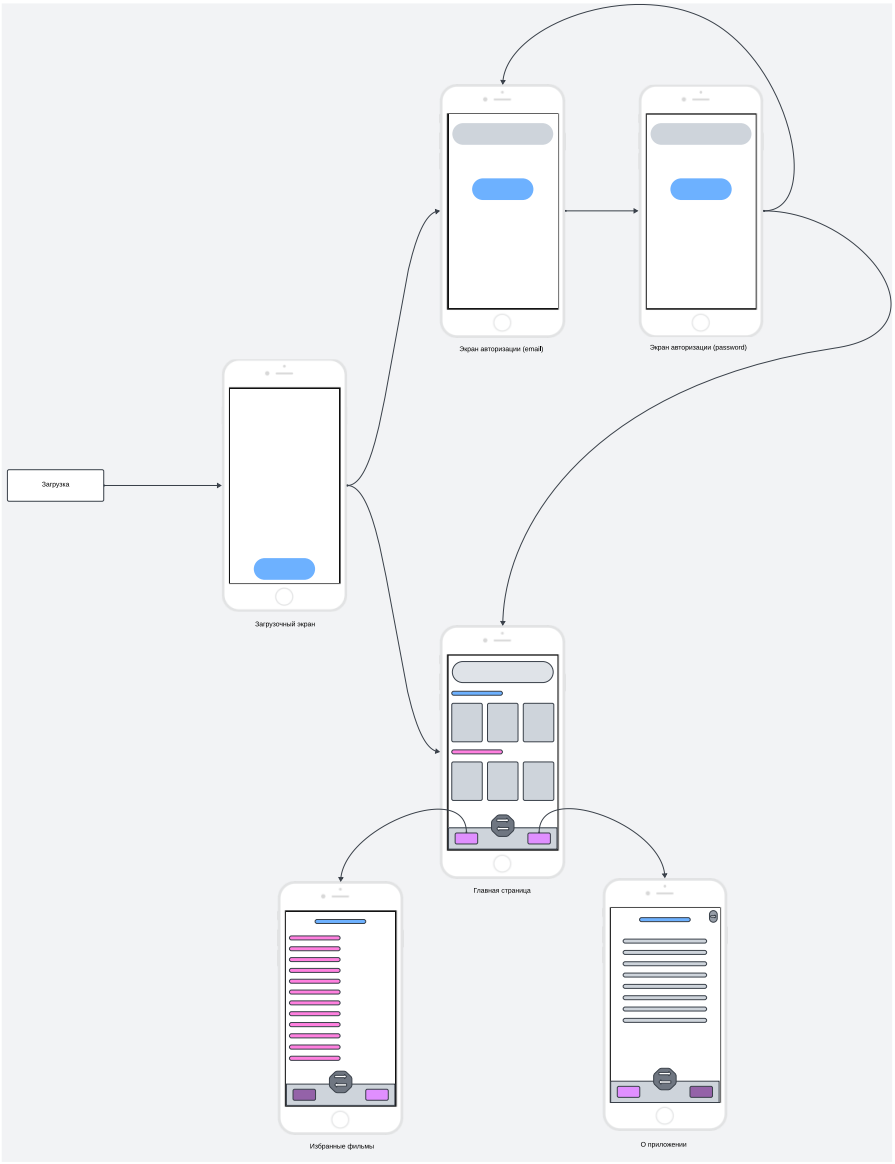

Рисунок 2 – проектирование пользовательского интерфейса

ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

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

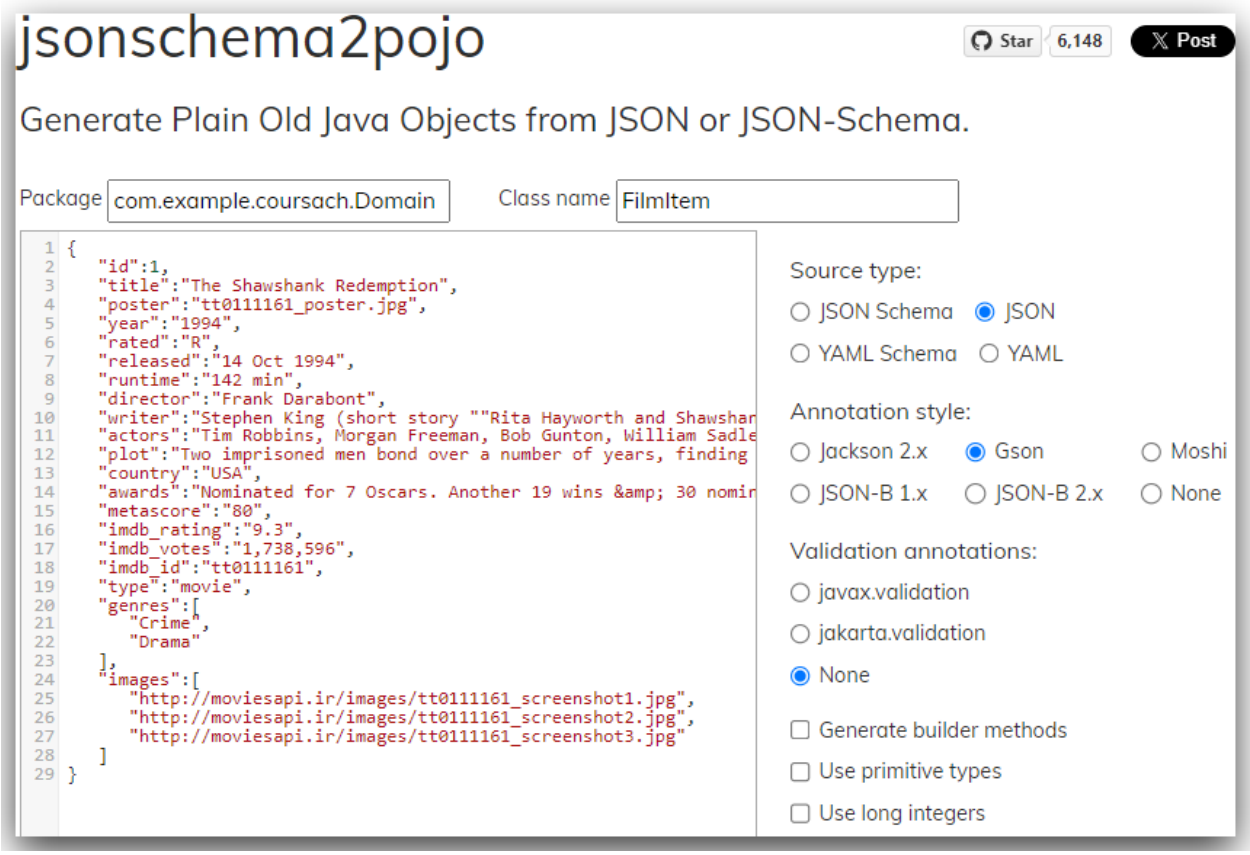

Рисунок 3 – JSON с конкретным фильмом

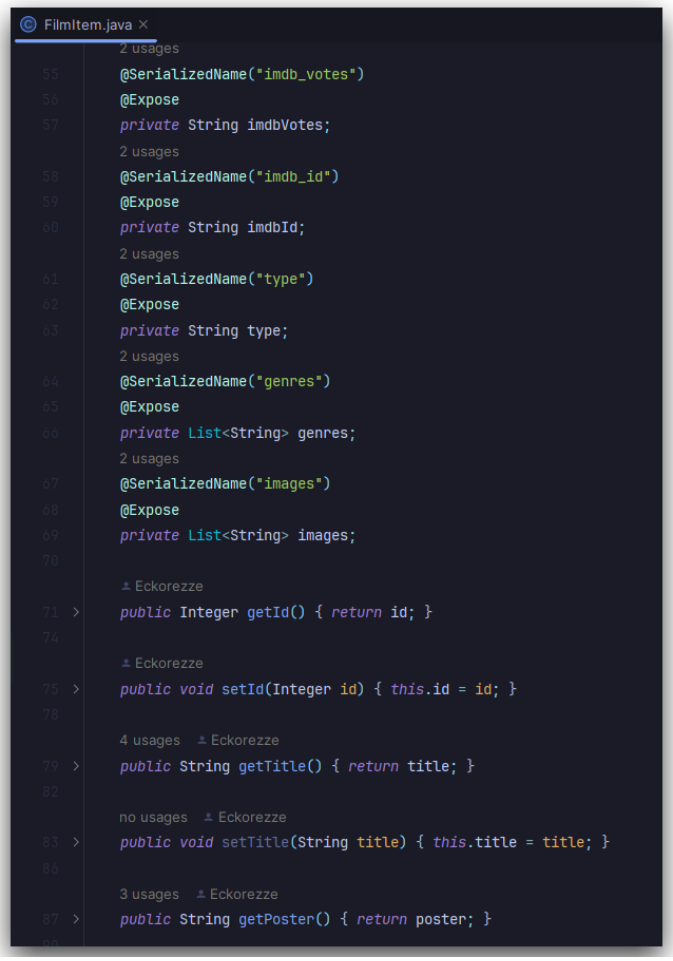

Рисунок 4 – полученный Java-объект

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 {

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

        LayoutInflater inflater =
LayoutInflater.from(requireContext());
        @SuppressLint("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_intr
oFragment_to_loginFragment);
            } else {
                showCustomSnackbar();
            }
        });
    }

    private boolean checkUserStatus() {
        FirebaseUser 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”.

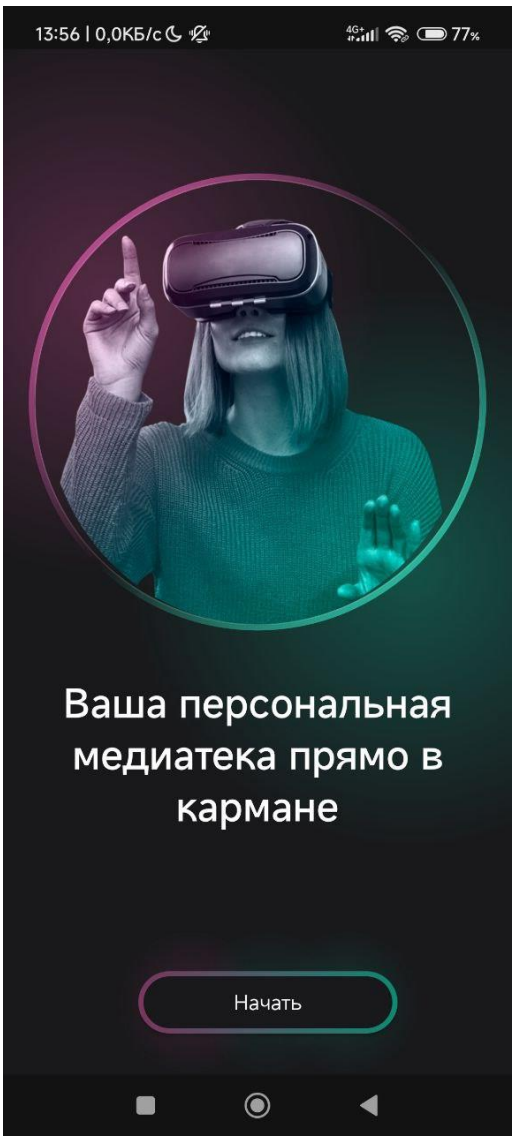

Рисунок 5 – стартовый экран

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 isEmailValid(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_logi
nFragment_to_mainFragment);
        } else {
            LoginFragmentDirections.ActionLoginFragmentToPassFragment
direction =

LoginFragmentDirections.actionLoginFragmentToPassFragment(email);

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

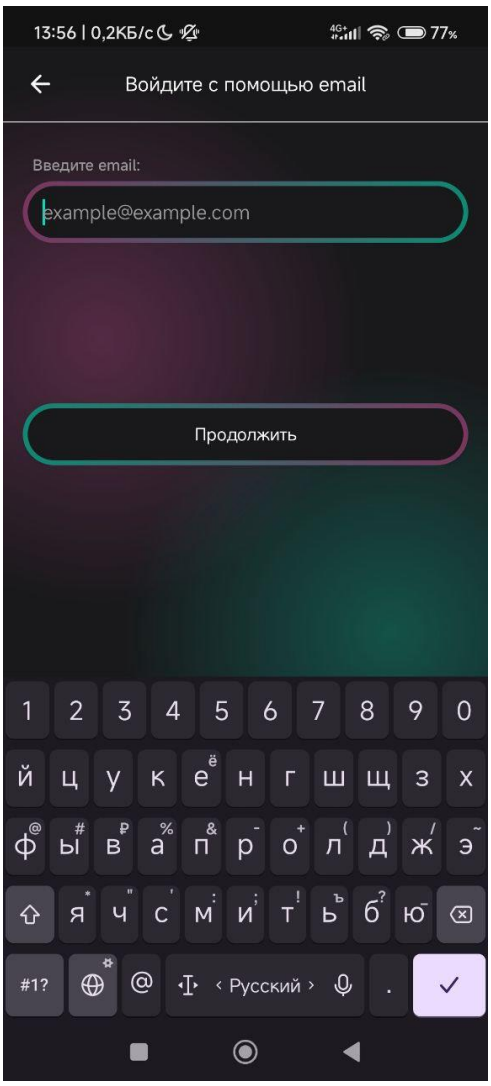

Рисунок 6 – экран ввода почты

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()) {
                    FirebaseUser newUser = mAuth.getCurrentUser();
                    if (newUser != null) {
                        saveUserAuthState(newUser.getUid());
                        sendEmailVerification();
                    }
                } else {
                    showCustomSnackbar(GetString(R.string.errreg));
                }
            }).addOnFailureListener(e ->
showCustomSnackbar(GetString(R.string.errregexist)));
        }
    });
}

private void sendEmailVerification() {
    FirebaseUser 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));
    }
}
```

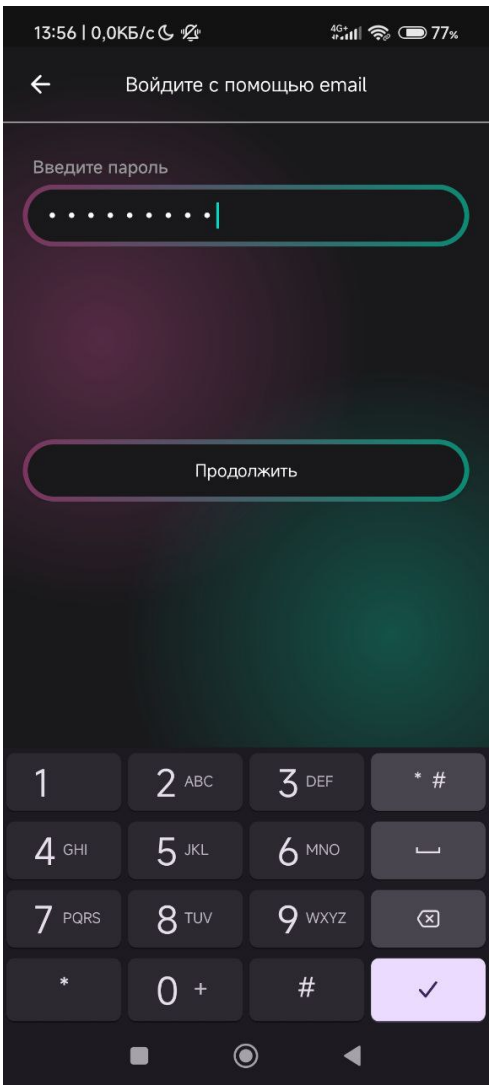

Рисунок 7 – экран ввода пароля

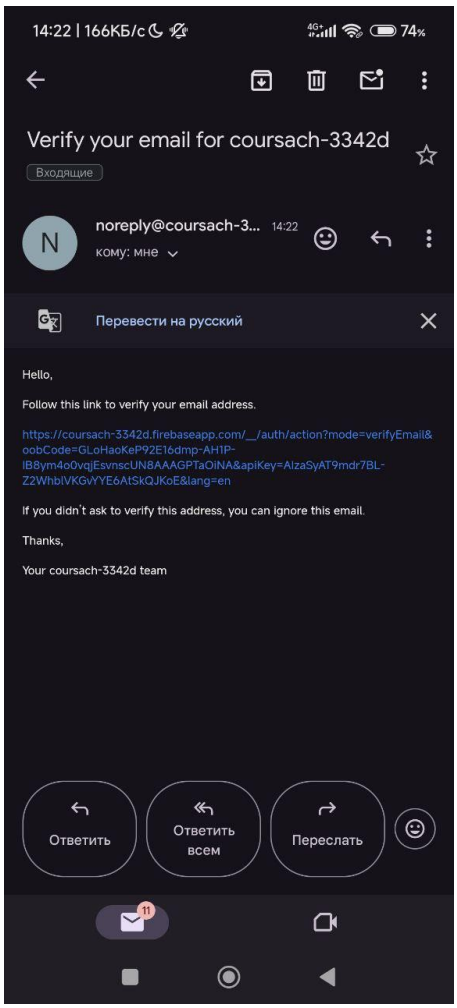

Рисунок 8 – письмо с подтверждением почты

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

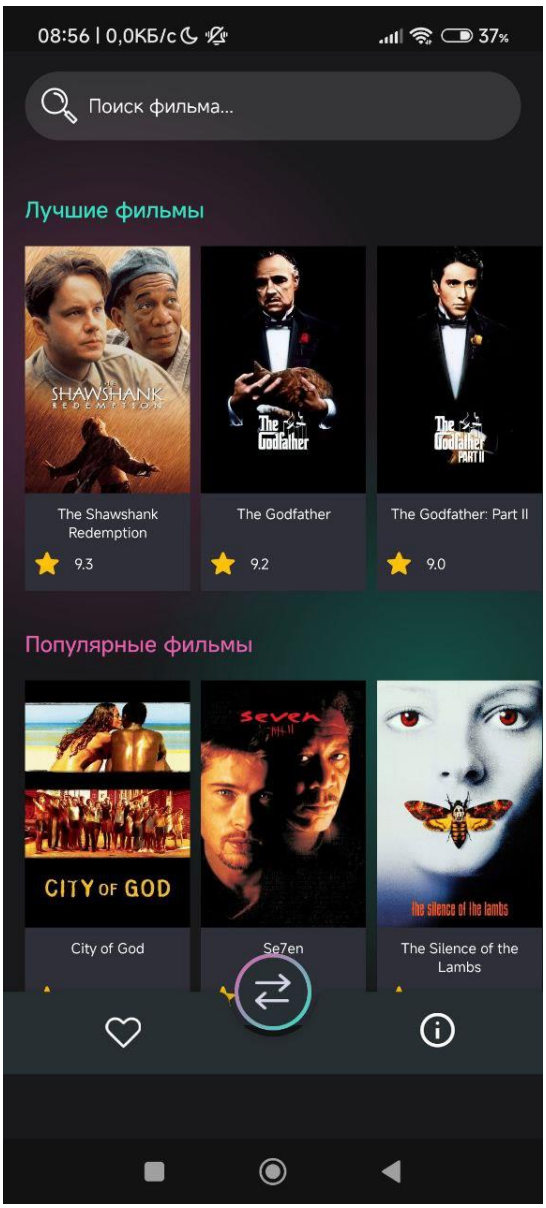

Рисунок 9 – главный экран

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);
}
```

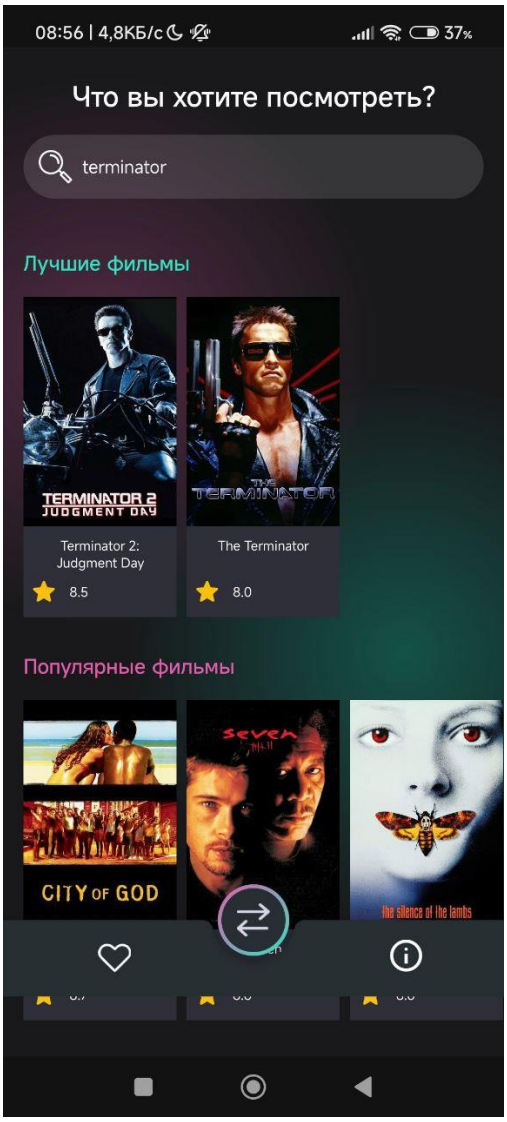

Рисунок 10 – поиск фильмов

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.

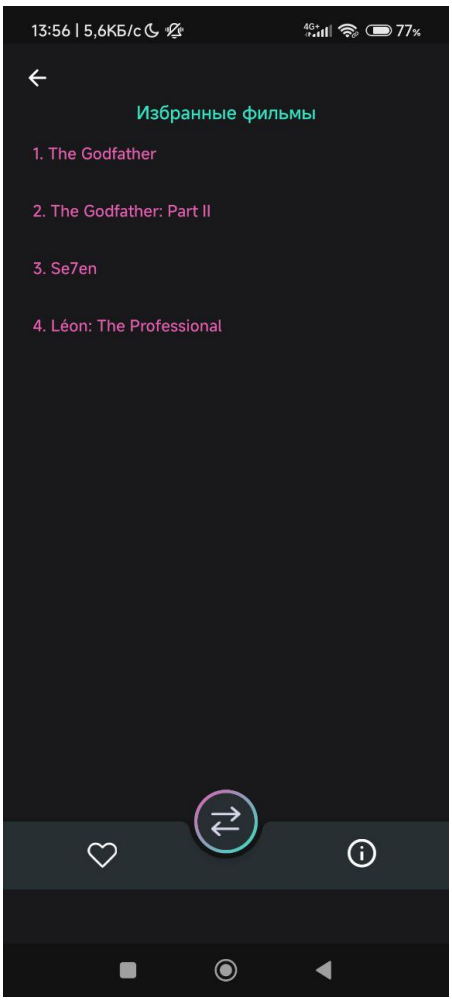

Рисунок 11 – избранные фильмы

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.getUid();
    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_cirlcle_dark);
    }
}

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

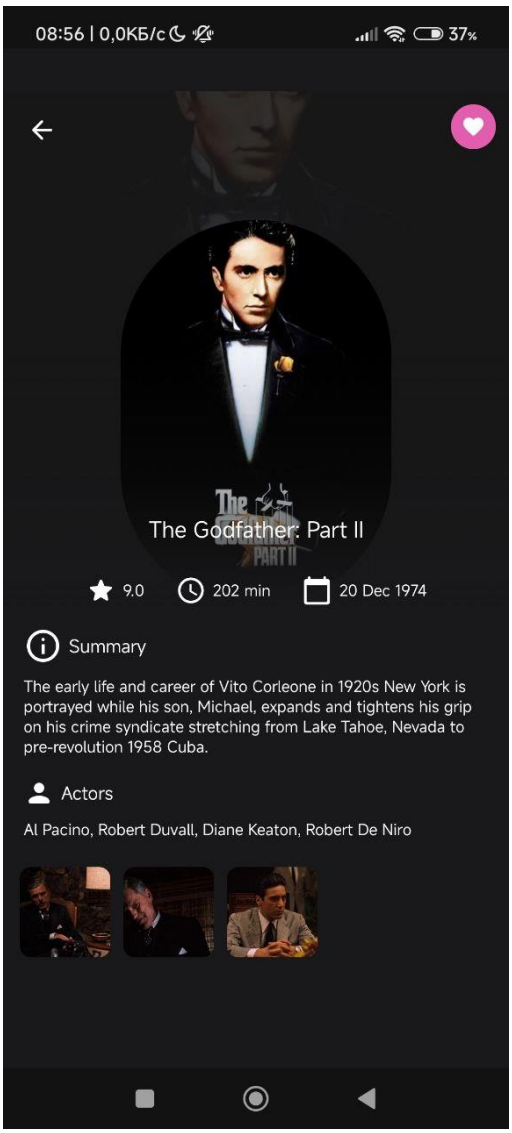

Рисунок 12 – детальная информация о фильме

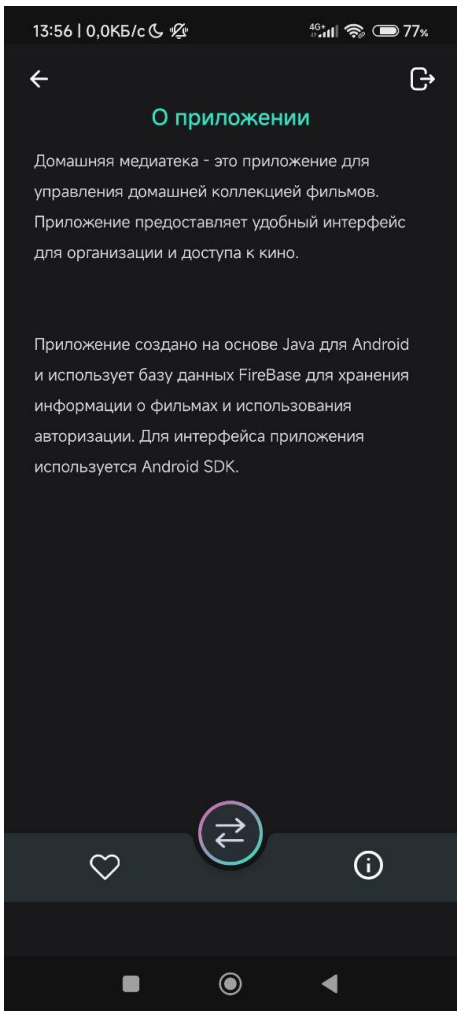

Рисунок 13 – раздел с информацией о приложении и кнопкой выхода из аккаунта

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 [5].

Преимущества использования Fragments над Activity:

- Модульность. Фрагменты могут быть переиспользованы в различных активити, что облегчает повторное использование кода [5].

- Адаптивный интерфейс. С помощью фрагментов можно легко создавать адаптивные макеты, которые работают как на телефонах, так и на планшетах [5].
- Более гранулированное управление жизненным циклом. Фрагменты обладают собственным жизненным циклом, который тесно связан с жизненным циклом их хост-активити, позволяя более тонко управлять ресурсами [5].
- Лучшая поддержка многопанельных интерфейсов. Используя фрагменты, можно легко реализовать сложные пользовательские интерфейсы с несколькими панелями, как, например, в мастер-детальных интерфейсах [5].

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 [6].

Преимущества использования Navigation:

- Упрощение управления стеком фрагментов. Navigation автоматически обрабатывает сложности управления стеком фрагментов, что упрощает переходы между экранами [6].
- Визуальное редактирование. Используя Navigation Editor в Android Studio, разработчик может визуально конструировать навигационную структуру приложения, что делает процесс более интуитивно понятным [6].
- Улучшенное взаимодействие с архитектурными компонентами. Navigation работает отлично с другими компонентами Jetpack, такими как ViewModel и LiveData, обеспечивая лучшую интеграцию и производительность [6].

- Обработка глубоких ссылок. Navigation упрощает реализацию глубоких ссылок, которые направляют пользователя на конкретный экран в приложении, даже если приложение не запущено [6].

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>
```

Листинг 7 – .github/workflows/android.yml

```
name: Android CI

on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]

jobs:
  build:

    runs-on: ubuntu-latest
    needs: [tests]
    steps:
    - uses: actions/checkout@v3
    - name: set up JDK 17
      uses: actions/setup-java@v3
      with:
        java-version: '17'
        distribution: 'temurin'
        cache: gradle

    - name: Grant execute permission for gradlew
      run: chmod +x gradlew
    - name: Build with Gradle
      run: ./gradlew build

  lint:
    name: perform lint check
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'
          cache: gradle
      - name: Cache Gradle
        uses: actions/cache@v2
        with:
          path: ~/.gradle/caches
          key: ${{ runner.os }}-gradle-${{ hashFiles('**/*.gradle')
}}
          restore-keys: ${{ runner.os }}-gradle-
      - name: Grant execute permission for gradlew
        run: chmod +x gradlew
      - name: Lint
        run: ./gradlew lint
      - name: upload html lint report
        uses: actions/upload-artifact@v4.3.1
        with:
            name: lint.html
            path: app/build/reports/lint-results-debug.html
```

```
  tests:
    name: tests
    runs-on: ubuntu-latest
    needs: [lint]
    steps:
      - uses: actions/checkout@v3
      - name: set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'
          cache: gradle
      - name: Grant execute permission for gradlew
        run: chmod +x gradlew
      - name: Run tests
        run: ./gradlew test
      - name: upload html test report
        uses: actions/upload-artifact@v4.3.1
        with:
            name: test
            path: app/build/reports/tests/testDebugUnitTest/
```

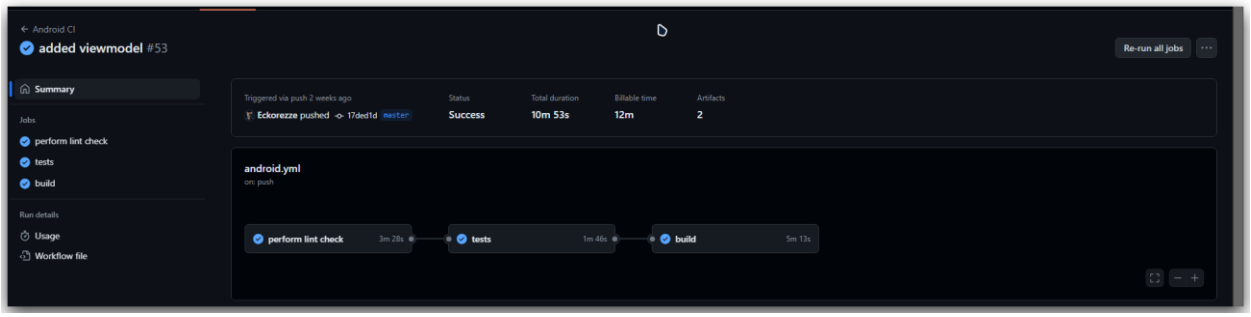

Рисунок 14 – успешное CI

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. Аналоги Android Studio. — Текст : электронный // RuProgi : [сайт]. — URL: https://ruprogi.ru/software/android-studio (дата обращения: 30.05.2024).
4. Meet Android Studio. — Текст : электронный // Android Developers : [сайт]. — URL: https://developer.android.com/studio/intro (дата обращения: 30.05.2024).
5. Fragments. — Текст: электронный // Developers: [сайт]. — URL: https://developer.android.com/guide/fragments (дата обращения: 30.04.2024).
6. Navigation. — Текст: электронный // Developers: [сайт]. — URL: https://developer.android.com/guide/navigation (дата обращения: 30.04.2024).
7. 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).
8. 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).

9. Generate Plain Old Java Objects from JSON or JSON-Schema.. — Текст : электронный // jsonschema2pojo : [сайт]. — URL: https://www.jsonschema2pojo.org/ (дата обращения: 06.05.2024).