**Содержание**

[Введение 5](#_Toc167042592)

[1. Аналитический обзор литературы 6](#_Toc167042593)

[1.2 Аналог «Fandango» 6](#_Toc167042594)

[1.2 Аналог «AMC Theatres» 7](#_Toc167042595)

[1.3 Аналог «Cinemark» 8](#_Toc167042596)

[2. Проектирование приложения 9](#_Toc167042597)

[2.1 Проектирование базы данных 9](#_Toc167042598)

[2.2 Описание информационных объектов и ограничений целостности 9](#_Toc167042599)

[2.2.1 Коллекция Data 10](#_Toc167042600)

[2.2.2 Коллекция Film 10](#_Toc167042601)

[2.2.3 Коллекция Theatre 10](#_Toc167042602)

[2.2.4 Коллекция Time 10](#_Toc167042603)

[2.3 Проектирование мобильного приложения 10](#_Toc167042604)

[3. Реализация приложения 13](#_Toc167042605)

[3.1 Технические средства разработки 13](#_Toc167042606)

[3.2 Разработка мобильного приложения 14](#_Toc167042607)

[4. Тестирование приложения 17](#_Toc167042608)

[5. Руководство по использованию 19](#_Toc167042609)

[5.1 Руководство гостя 19](#_Toc167042610)

[5.2 Руководство администратора 20](#_Toc167042611)

[5.3 Руководство пользователя 21](#_Toc167042612)

[Заключение 26](#_Toc167042613)

[Список использованных источников 27](#_Toc167042614)

[ПРИЛОЖЕНИЕ А 28](#_Toc167042615)

# Введение

Неотъемлемой частью жизни большинства людей в современном информационном обществе являются разнообразные развлекательные приложения. Одним из видов таких приложений являются мобильные приложения для кинотеатров, позволяющие бронировать билеты, просматривать расписание сеансов и получать информацию о новинках кино. Такие приложения предоставляют возможность легко и быстро планировать походы в кинотеатр и оставаться в курсе последних кинопремьер. Стремительный рост числа пользователей интернета и мобильных устройств, а также развитие киноиндустрии сделали такие приложения весьма востребованными.

Мобильные приложения для кинотеатров ориентированы на широкую аудиторию. Они позволяют увлекательно провести досуг, предоставляя пользователям удобный доступ к информации о фильмах и возможности забронировать билеты в режиме онлайн.

В процессе работы над проектом были поставлены следующие задачи:

• Реализовать функции гостя, зарегистрированного пользователя и администратора.

• Обеспечить регистрацию и авторизацию пользователей.

• Реализовать возможность бронирования и покупки билетов.

• Обеспечить доступ к информации о фильмах, включая описание, трейлеры и отзывы.

• Реализовать возможность управления расписанием и контентом со стороны администратора.

Основная цель курсового проекта: создание мобильного приложения для кинотеатра, которое позволит пользователям удобно и быстро планировать походы в кино. Язык разработки проекта – Flutter. В ходе выполнения курсового проекта будет использована облачная база данных Firebase.

# Аналитический обзор литературы

В современном мире мобильные приложения для кинотеатров становятся все более популярными и востребованными. Их основная задача — это предоставление пользователям удобного способа планирования походов в кинотеатр, включая просмотр расписания, бронирование и покупку билетов. При разработке приложения для своего проекта я обратила внимание на самые успешные примеры популярных мобильных приложений для кинотеатров. Аналоги рассмотрены ниже.

## **Аналог «Fandango»**

Одним из наиболее известных аналогов является приложение «Fandango», представленное на рисунке 1.1. Это приложение предназначено для просмотра расписания киносеансов, бронирования и покупки билетов, а также получения информации о новинках кино. Пользователи могут читать отзывы, смотреть трейлеры и узнавать о предстоящих релизах.

.

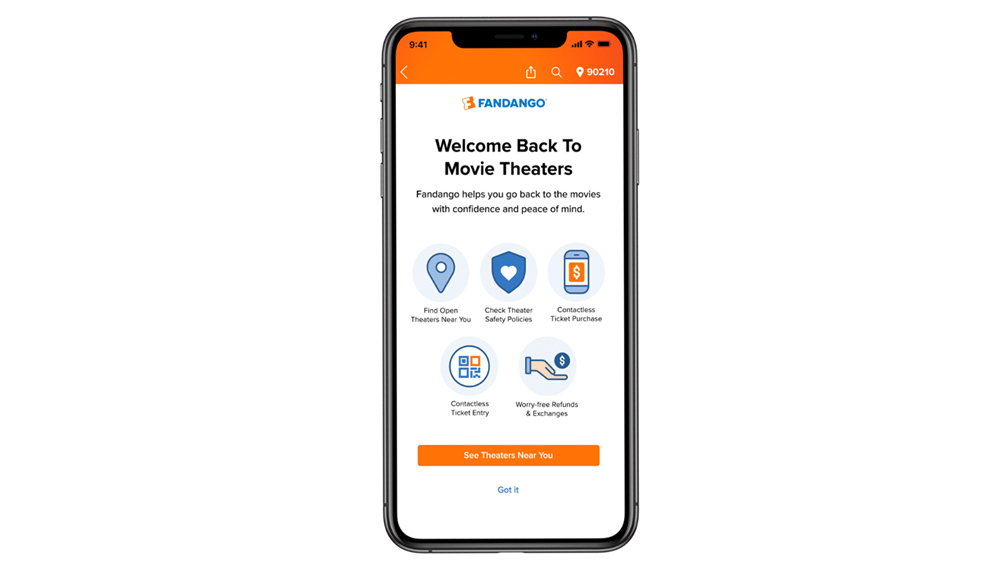


Рисунок 1.1 – Приложение «Fandango»

Из плюсов хотелось бы отметить интуитивно понятный интерфейс, возможность покупки билетов онлайн, обширная база данных фильмов и кинотеатров, функция чтения и написания отзывов.

Из минусов — некоторые функции доступны только для премиум-пользователей и ограниченная поддержка локальных кинотеатров за пределами США.

## **Аналог** «AMC Theatres»

Другим примером является приложение «AMC Theatres», показанное на рисунке 1.2. Это приложение разработано одной из крупнейших кинотеатральных сетей в США и предоставляет полный спектр услуг, включая покупку билетов, просмотр расписания и участие в программе лояльности.

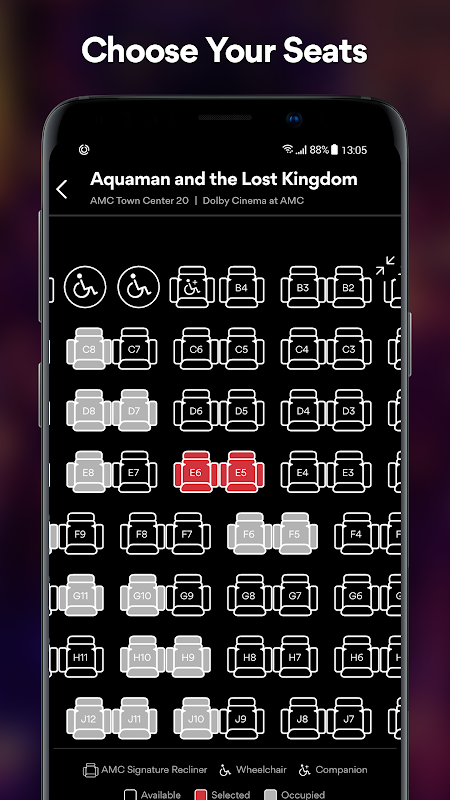


Рисунок 1.2 – Приложение «We Heart It»

Из плюсов — программа лояльности, возможность выбора места в зале, информация о специальных показах и событиях, удобное управление учетной записью.  
 Из минусов — некоторые пользователи отмечают проблемы с производительностью.

## **Аналог** «Cinemark»

Третьим примером является приложение «Cinemark», представленное на рисунке 1.3. Это приложение позволяет пользователям бронировать и покупать билеты, просматривать расписание сеансов, а также управлять своей программой лояльности Cinemark Movie Rewards.

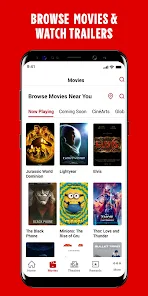


Рисунок 1.3 – Приложение «Cinemark»

Из плюсов — программа лояльности, возможность предзаказа еды и напитков, удобный и понятный интерфейс, поддержка множества кинотеатров по всему миру.

Из минусов — ограниченная поддержка некоторых регионов, случаются временные сбои в работе приложения.

Все рассмотренные приложения предлагают широкий спектр функционала для удобства пользователей. Они позволяют не только бронировать и покупать билеты, но и предоставляют обширную информацию о фильмах, включая отзывы и трейлеры. Однако каждое из приложений имеет свои особенности и ограничения, которые следует учитывать при разработке собственного мобильного приложения для кинотеатра.

# Проектирование приложения

Проектирование программного средства — это процесс создания детального плана или концепции программного продукта до его реализации. Оно включает в себя определение структуры и организации программы, выбор подходящих архитектурных решений, проектирование интерфейсов и определение функциональности системы.

## Проектирование базы данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь.

Для данного курсового проекта была спроектирована база данных «Nepflix». Логическая модель базы данных представлена на рисунке 2.1.

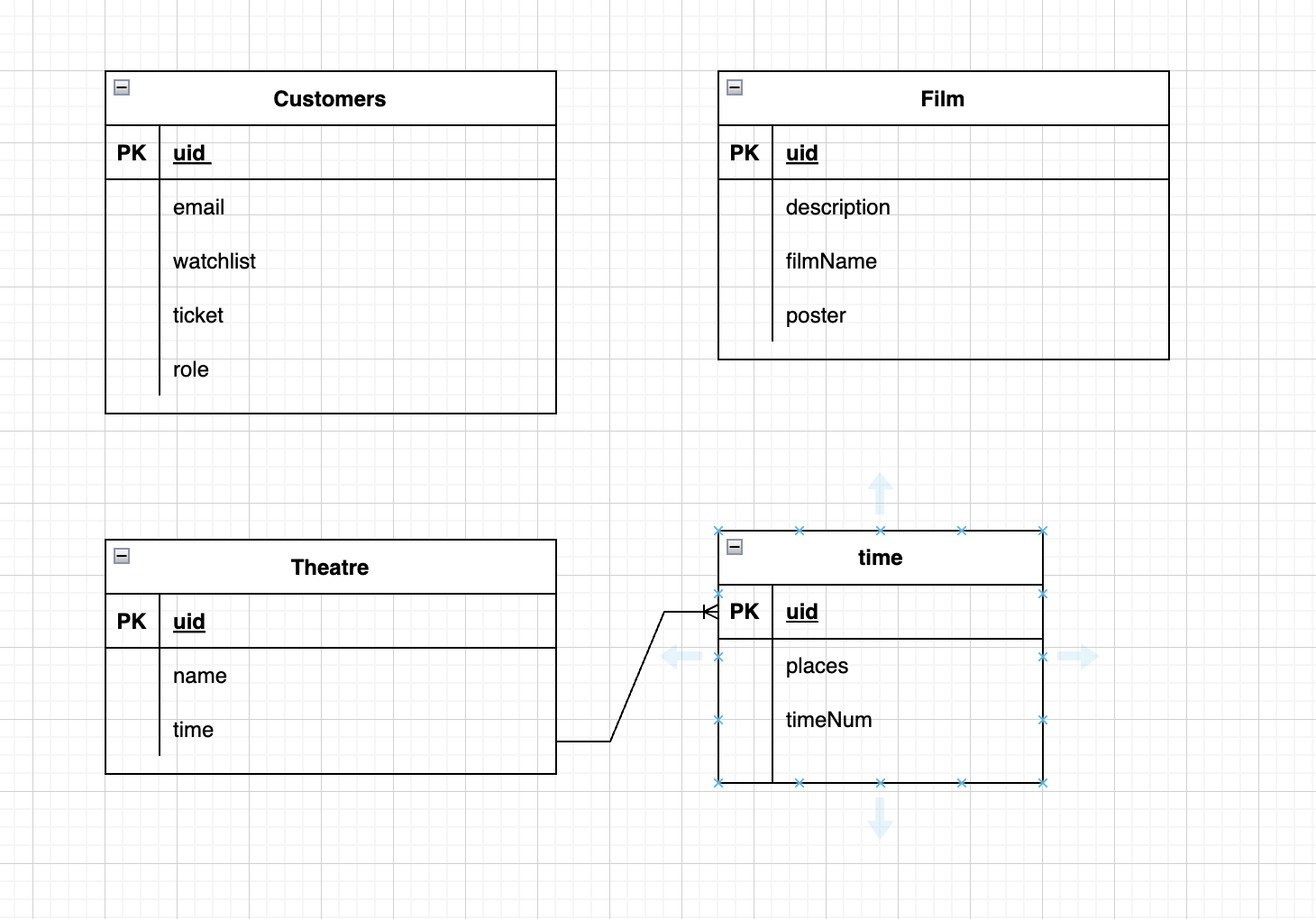


Рисунок 2.1 – Модель БД

Созданная база данных содержит в себе 4 коллекции: Users, Posts, Favorites и Comments.

* 1. Описание информационных объектов и ограничений целостности

Далее будет описана структура каждой коллекции. В коллекции Data хранятся данные для идентификации пользователей, зарегистрированных в приложении, Film – данные о загруженных фильмах, Theatre для хранения кинотеатров.

### **Коллекция Data**

Коллекция состоит из следующих полей:

* uid. Хранит уникальный идентификатор пользователя;
* email. Хранит email пользователя;
* watchlist. Хранит список сохранённых фильмов;
* ticket. Хранит информацию о приобретенных билетах;
* role. Хранит роль пользователя.

### **Коллекция Film**

Коллекция Film состоит из следующих полей:

* uid. Хранит уникальный идентификатор фильма;
* description. Описание фильма;
* filmName. Название фильма;
* poster. Хранит постер фильма.

### **Коллекция Theatre**

В состав коллекции Theatre входят следующие поля:

* uid. Хранит уникальный кинотеатр;
* name. Хранит название кинотеатра.

Коллекция Theatre который содержит ссылку на коллекцию Time

### **Коллекция Time**

В состав коллекции favorites входят следующие поля:

* uid. Хранит уникальный идентификатор времени;
* places. Хранит места, которые можно забронировать;
* timeNum. Хранит время.

## Проектирование мобильного приложения

Интерфейс пользователя в приложении служит точкой контакта между пользователем и программой, формируя первоначальное впечатление о продукте, которое может быть как положительным, так и отрицательным.

В рамках данного проекта ключевым является фокус на создании мобильного приложения, учитывающего особенности проекта, и здесь пользователи ожидают наиболее высокого качества работы. Оптимальным подходом к организации кода клиентской части является его разбиение на модули, позволяющие работать над каждой частью независимо друг от друга. С учетом всех требований к функционалу, структура клиентской части приложения может быть организована по двум основным направлениям: взаимодействие с пользовательским интерфейсом и внутренние процессы приложения.

Навигация в мобильных приложениях играет ключевую роль, обеспечивая логическую связь между различными экранами и страницами. При проектировании навигации важно учитывать, как пользователи будут перемещаться внутри приложения, чтобы сделать процесс максимально интуитивным и удобным.

При первом запуске приложения пользователь попадает на страницу входа. Оттуда он может попасть на страницу регистрации, либо продолжить без входа. После успешной авторизации/регистрации пользователь попадет на экран с тремя страницами: все фильмы, поиск, профиль.

На странице со всеми постами пользователь может их просмотреть. При клике на фильм можно перейти на детальную страницу фильма, где ко всему прочему можно еще и посмотреть трейлер, добавить в избранное.

На странице поиска можно найти интересующие фильмы.

В профиле пользователь может просмотреть свои персональные данные и выйти из аккаунта.

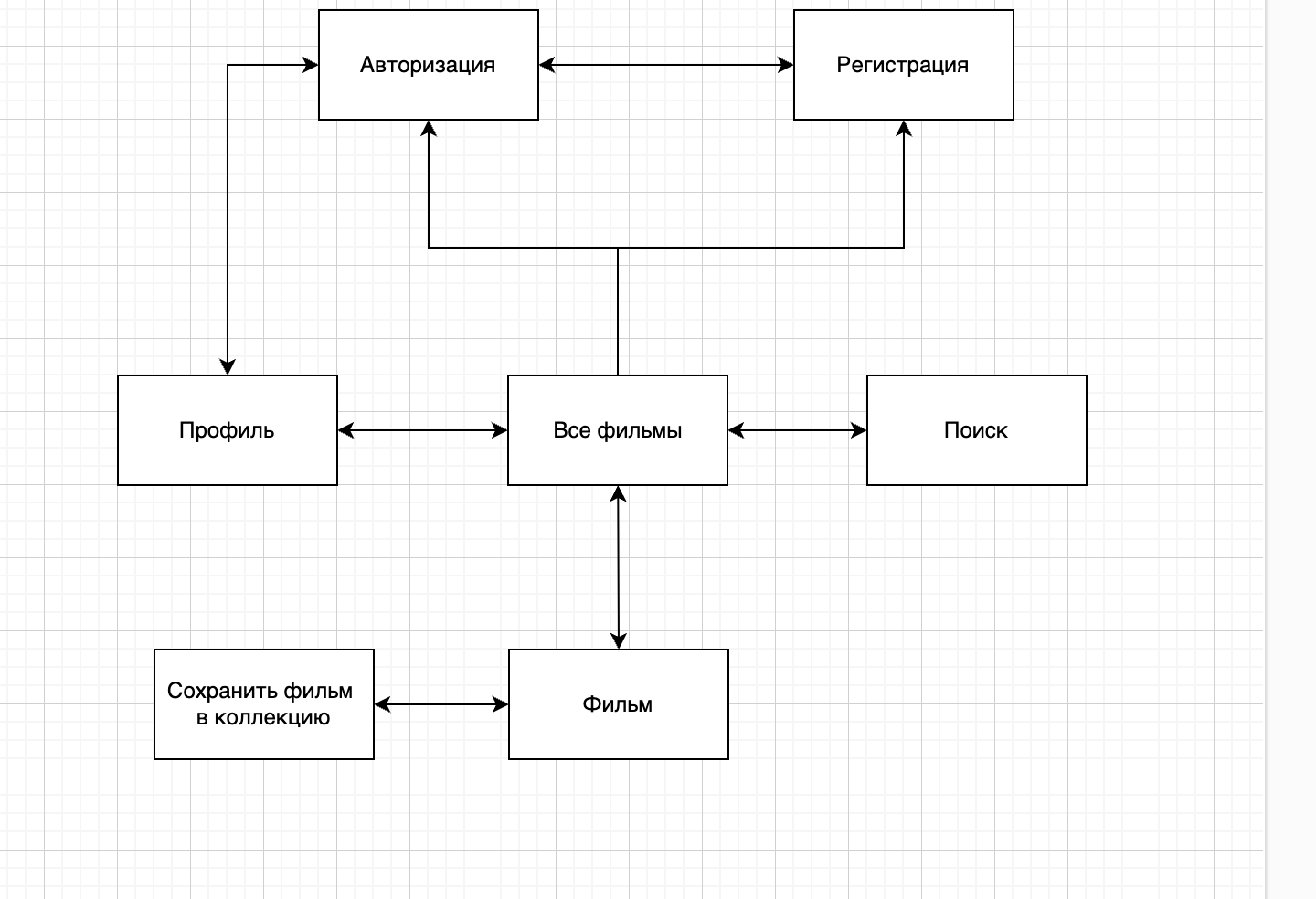
****

Рисунок 2.2 – Схема навигации для пользователя

Если пользователь решил продолжить без входа, то у него доступны три страницы: все фильмы, поиск и можно опять перейти на страницу входа.

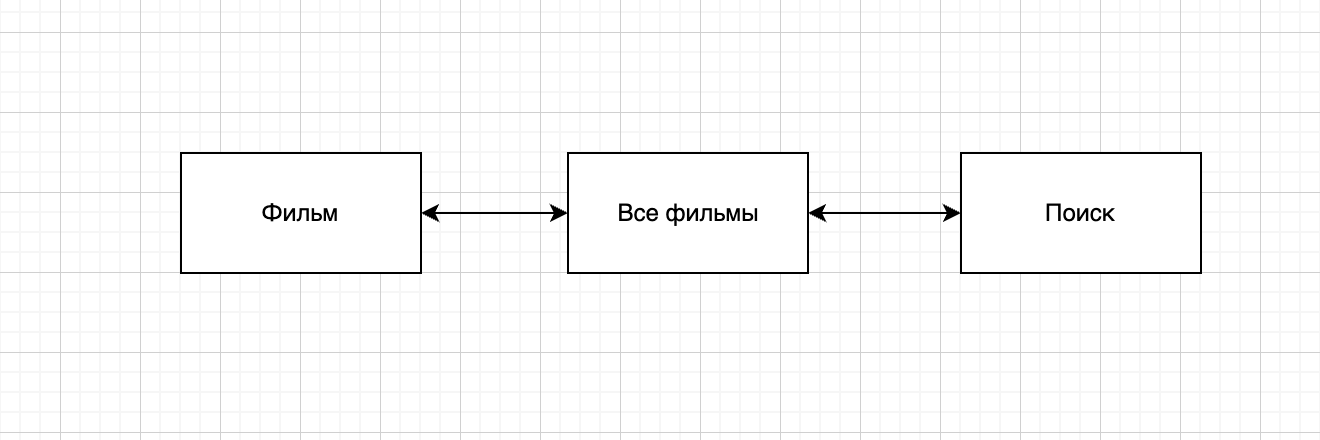


Рисунок 2.3 – Схема навигации для гостя.

# Реализация приложения

Реализация программного средства — это процесс создания и внедрения программного обеспечения, включающий в себя разработку, тестирование, отладку и выпуск продукта для конечных пользователей. Этот процесс состоит из нескольких ключевых этапов, которые обеспечивают функциональность, надежность и удовлетворение требований пользователей.

## Технические средства разработки

Для реализации курсового проекта был выбран язык программирования Flutter. Flutter — это открытая платформа разработки от компании Google, которая позволяет создавать высокопроизводительные мобильные приложения для различных платформ, таких как Android и iOS, а также веб-приложения и настольные приложения из одной кодовой базы. Одним из ключевых преимуществ Flutter является возможность использования одного и того же кода для разных платформ. Разработчики пишут приложение на языке Dart, который является основным языком для Flutter, и затем компилируют его в нативный код для каждой платформы. Это обеспечивает высокую производительность и единообразный пользовательский интерфейс на разных устройствах.

База данных для проекта была разработана с помощью Firebase. Firebase — это облачная платформа разработки приложений, предоставляемая компанией Google. Она предлагает набор инструментов и сервисов, которые упрощают создание и управление мобильными и веб-приложениями, избавляя разработчиков от забот о сложностях инфраструктуры.

Firebase Authentication — это сервис аутентификации, обеспечивающий безопасную регистрацию и вход пользователей в приложение. Он поддерживает различные методы аутентификации, включая электронную почту и социальные сети.

Cloud Firestore — это гибкая и масштабируемая база данных в реальном времени, которая позволяет хранить и синхронизировать данные между клиентскими устройствами и сервером. Firestore предлагает мощные функции запросов, мгновенные обновления данных и автоматическую синхронизацию.

Firebase Storage — это сервис для хранения файлов, позволяющий загружать и скачивать файлы, такие как изображения, видео и документы, в облачное хранилище. Он обеспечивает простой доступ к файлам и интеграцию с другими сервисами Firebase.

Firebase Cloud Messaging — это сервис для отправки push-уведомлений и сообщений на мобильные устройства пользователей. Он обеспечивает надежную доставку уведомлений и поддерживает настройку аудитории.

Firebase Hosting — это сервис для размещения веб-сайтов и статических веб-приложений, который позволяет разработчикам развертывать и обслуживать свои приложения из облака Google. Он обеспечивает быструю и надежную доставку контента по всему миру.

Firebase Analytics — это инструмент аналитики, предоставляющий подробную информацию о поведении пользователей, использовании приложений и эффективности маркетинговых кампаний. Он помогает разработчикам принимать обоснованные решения, оптимизировать свои приложения и улучшать пользовательский опыт.

Firebase также предлагает множество других сервисов, таких как машинное обучение, тестирование, анализ ошибок и аутентификация через социальные сети. Он интегрируется с различными платформами разработки, включая Android, iOS, веб и серверы, что делает его мощным инструментом для создания современных и масштабируемых приложений.

Для данного проекта используются Firebase Authentication, Firebase Cloud Firestore и Firebase Storage.

## Разработка мобильного приложения

Мобильное приложение содержит в себе несколько папок. Каждая из перечисленных папок в мобильном приложении выполняет определенную функцию:

* Models содержит классы со свойствами которые описывают сущности используемые в мобильном приложении.
* Screens содержит основные экраны приложения которые видит пользователь.
* Services содержит классы для работы с Firebase, различные константы, сервис аутентификации пользователя и другие сервисы.
* Widgets содержит виджеты из которых состоят основные экраны приложения.

Также в проекте есть файл routes.dart, который обеспечивает роутинг в проекте. В данном файле прописаны пути, по переходу на которые будет представлен нужный нам экран. Пути, описанные в проекте представлены в листинге 3.1

GoRouter router = GoRouter(initialLocation: '/', routes: [

GoRoute(

path: '/',

builder: (context, state) => const NavScreen(),

),

GoRoute(

path: '/login',

builder: (context, state) => const LoginScreen(),

pageBuilder:defaultPageBuilder<LoginScreen>(const LoginScreen()),

),

GoRoute(

path: '/main',

builder: (context, state) => const MainScreen(),

pageBuilder: defaultPageBuilder<MainScreen>(const MainScreen()),

),

GoRoute(

path: '/search',

builder: (context, state) => const SearchScreen(),

pageBuilder: defaultPageBuilder<SearchScreen>(const SearchScreen()),

),

GoRoute(

path: '/profile',

builder: (context, state) => const ProfileScreen(),

pageBuilder: defaultPageBuilder<ProfileScreen>(const ProfileScreen()),

),

GoRoute(

path: '/create-film',

builder: (context, state) => const CreateFilmScreen(),

pageBuilder: defaultPageBuilder<CreateFilmScreen>(const CreateFilmScreen()),

),

GoRoute(

path: '/movie/:id',

builder:(context,state)=> MovieScreen(state.pathParameters['id']!),

),

GoRoute(

path: '/movieAdmin/:id',

builder:(context,state)=> MovieScreen(state.pathParameters['id']!),

),

GoRoute(

path: '/tv/:id',

builder:(context,state)=> TVShowScreen(state.pathParameters['id']!),

)

]);

Листинг 3.1 – Пути доступные в проекте

Реализация асихронной функции, которая выполняет запрос к Firebase для получения информации о фильме, представлена в листинге 3.2. Она принимает идентификатор фильма в качестве параметра. Здесь используются функции для работы с Firebase, что сильно упрощает получение объектов из базы данных

Future getOneFilm(id) async {

var documentSnapshot = await \_firestore

.collection('Film').doc(id).get().then((doc) => doc.data());

return documentSnapshot;

}

Листинг 3.2 – Получение информации о фильме

Также одним из основных функционалом является добавление фильма к личному списку просмотренных. Реализацию данной функции можно увидеть на листинге 3.3

Future<void> addWatching(String id, String status, String mediaType) async {

List watchlist = await \_firestore

.collection("Data")

.doc(\_auth.currentUser!.uid)

.get()

.then((value) => value.data()!["WatchList"]);

for (var i = 0; i < watchlist.length; i++) {

if (watchlist[i]["Id"] == id) {

await \_firestore

.collection("Data")

.doc(\_auth.currentUser!.uid)

.update({

"WatchList": FieldValue.arrayRemove([

{

"Id": id,

"status": watchlist[i]["status"],

"mediaType": watchlist[i]["mediaType"],

}

])

})

.then((value) {})

.catchError((error) {});

}

}

await \_firestore

.collection("Data")

.doc(\_auth.currentUser!.uid)

.update({

"WatchList": FieldValue.arrayUnion([

{

"Id": id,

"status": status,

"mediaType": mediaType,

}

])

})

.then((value) => {})

.catchError((error) {});

}

Листинг 3.3 – Функция добавления к списку просмотренных

Пример реализации авторизации можно найти в приложении Б.

# Тестирование приложения

Тестирование мобильных приложений важно и необходимо для обеспечения их качества и производительности. Оно помогает уменьшить риск сбоев приложения, гарантирует отсутствие ошибок и улучшает восприятие пользователей, что в свою очередь способствует увеличению количества загрузок и успешному запуску приложения на рынке.

Для начала протестируем страницу авторизации. Она показана на рисунке 4.1. Мы видим две кнопки, которые позволяют пользователю зайти с разными условиями, как гость или войти с помощью Google.

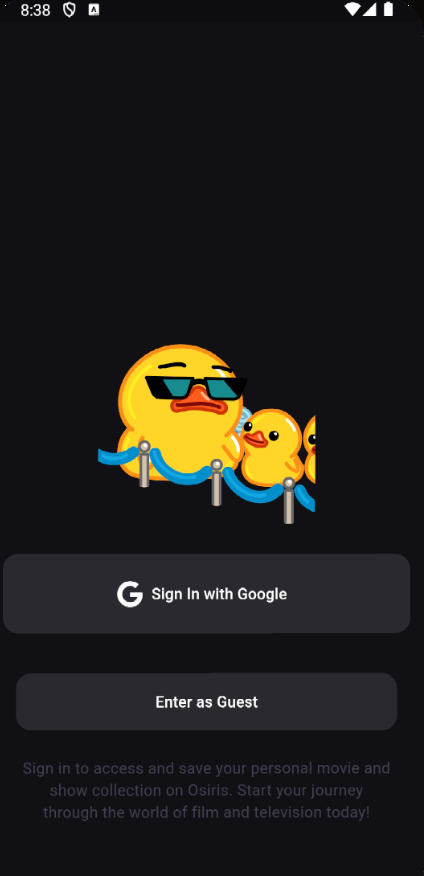


Рисунок 4.1 – Тестирование авторизации

Далее после авторизации мы должны попасть на главный экран приложения. Результат тестирования авторизации можно увидеть на рисунке 4.2.

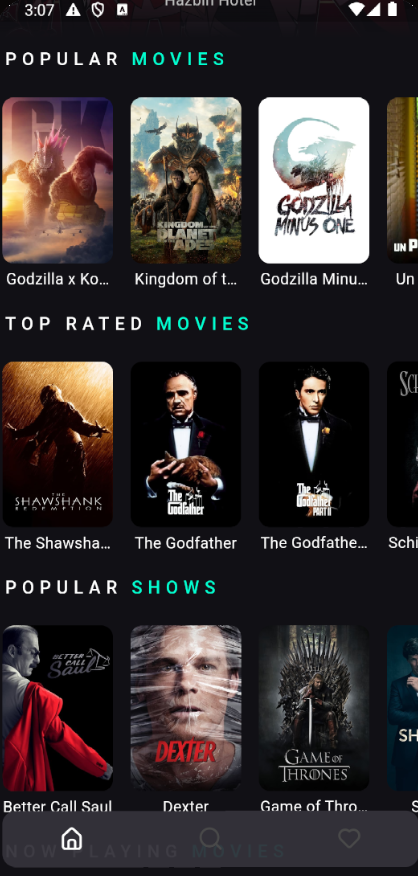


Рисунок 4.2 – Результат тестирования авторизации

После авторизации или регистрации пользователь попадает на страницу со всеми фильма. Протестируем добавление нового фильма. Тестирование продемонстрировано на рисунке 4.3.

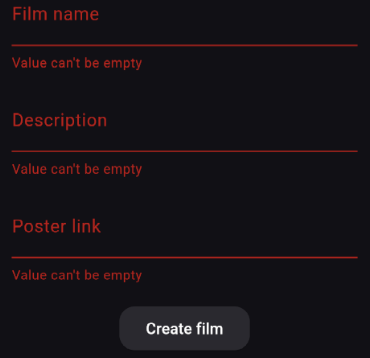


Рисунок 4.3 – Тестирование добавления фильма

Таким образом, в разработанном мобильном приложении корректно обработаны возможные ошибки при использовании функций.

# Руководство по использованию

## Руководство гостя

Для того чтобы пользователь зашел как гость, при авторизации или регистрации нужно нажать на кнопку «Enter as Guest». После этого гость попадает на страницу, показанную на рисунке 5.1.

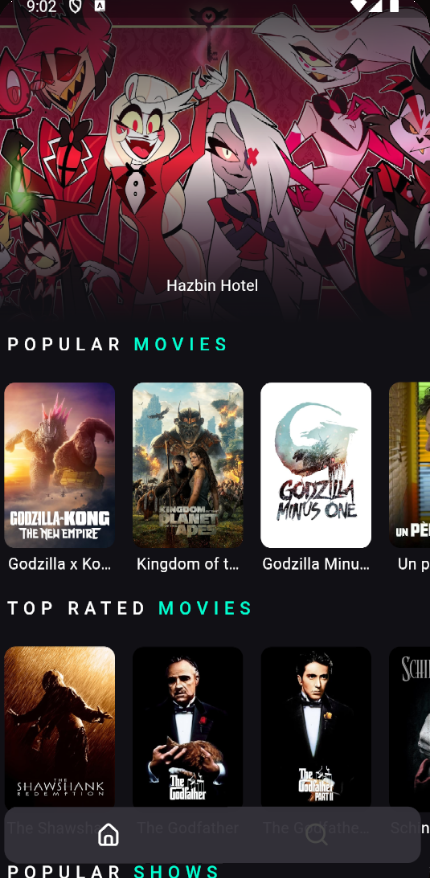


Рисунок 5.1 – Страница для гостя

При клике на любой фильм, откроется детальная страница фильм, где гость может увидеть постер, жанр фильма, дату релиза, описание фильма и рекомендованные фильмы. При желании есть кнопка, открывающая трейлер фильма. Детальная страница поста показана на рисунке 5.2.



Рисунок 5.2 – Детальная страница фильма для гостя

Купить билет и добавить в свой список фильмов гость не может.

Кроме этого, также предусмотрена возможность воспользоваться поиском и найти нужные фильмы. Или гость может перейти обратно на страницу авторизации/регистрации.

## Руководство администратора

Чтобы продолжить работу в приложении как администратор, пользователь должен иметь специальную роль и войти в приложение под соответствующей учетной записью. При входе в приложение как администратора будет открыта страница, показанная на рисунке 5.3.

В отличие от гостя, у администратора появляется добавлять, удалять и изменять фильмы.

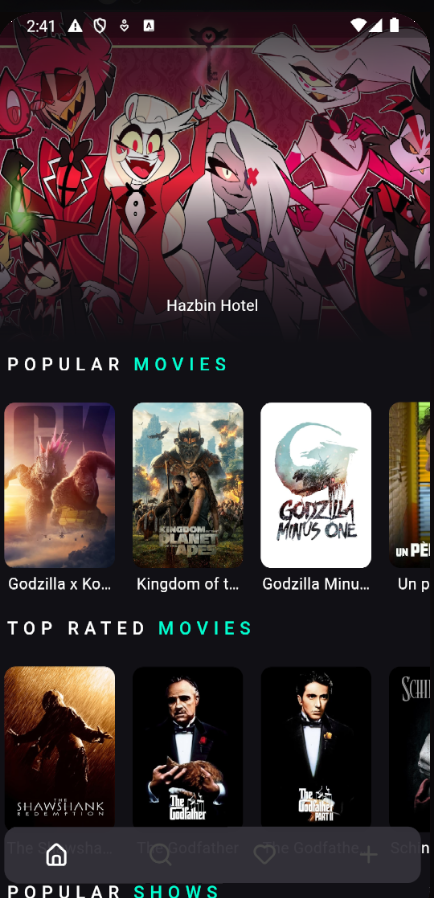


Рисунок 5.3 – Страница для админа

При переходе на детальную страницу поста, админ имеет дополнительные кнопки, которые отвечают за редактирование и удаление. Показано на рисунке 5.4.

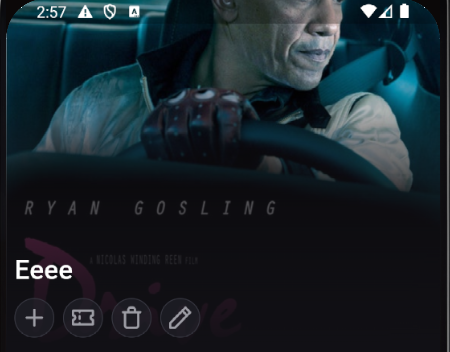


Рисунок 5.4 – Дополнительные кнопки админа

При нажатии на кнопку с мусоркой фильм будет удален, и администратор будет перенаправлен на главную страницу приложения. Как и у гостя, администратор может воспользоваться поиском фильмов, просмотреть трейлер.

## Руководство пользователя

У пользователей ранее не зарегистрированных есть возможность пройти регистрацию.

Регистрация осуществляется с помощью Google. Страница регистрации показана на рисунке 5.5.

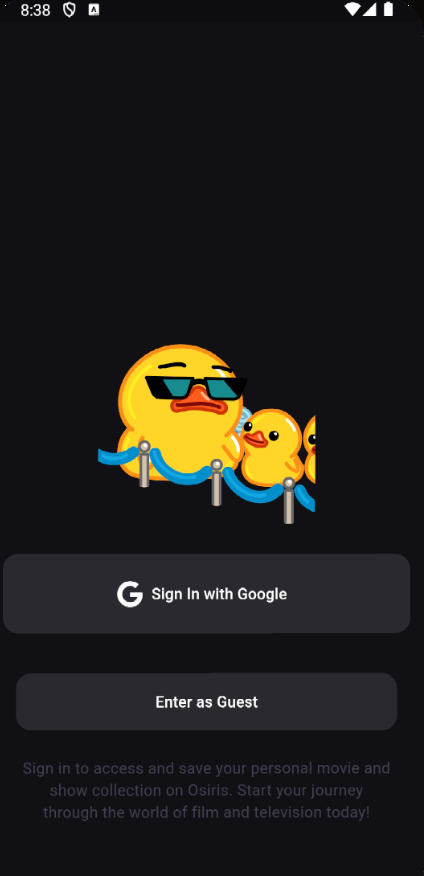


Рисунок 5.5 – Регистрация пользователя

После регистрации и авторизации пользователь переходит в основное окно приложения. Там он может просмотреть все фильмы, перейти к поиску или зайти в личный профиль. Показано на рисунке 5.6.

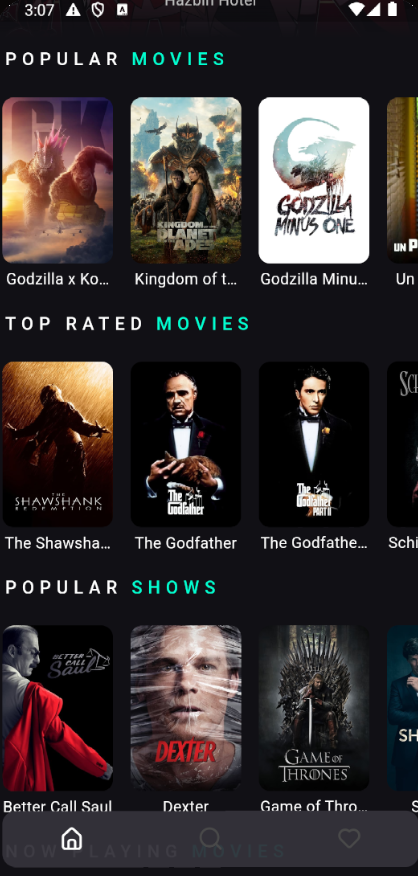


Рисунок 5.6 – Главная страница пользователя

На детальной странице фильма пользователь может посмотреть трейлер, добавить фильм в просмотренные, купить билет, перейти по рекомендованным фильмам. Можно увидеть на рисунке 5.7.



Рисунок 5.7 – Детальная страница фильма для пользователя

От гостя пользователя отличает возможность добавить фильм в свой личный список. Чтобы это сделать надо нажать на кнопку с плюсом, после чего появиться диалоговое окно, где пользователь должен выбрать куда именно сохранить фильм. Диалоговое окно представлено на рисунке 5.8.



Рисунок 5.8 – Диалоговое окно

После выбора куда сохранить фильм пользователь сможет найти его в своем личном кабинете.

Личный кабинет на рисунке 5.9.

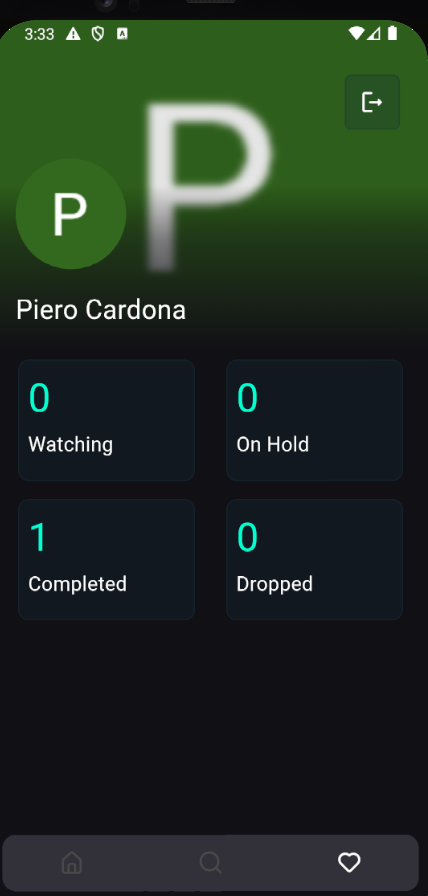


Рисунок 5.9 – Личный кабинет пользователя

Здесь пользователь выбирает нужный ему список и переходит на страницу с списком сохранённых фильмов. Cтраница представлена на рисунке 5.10.

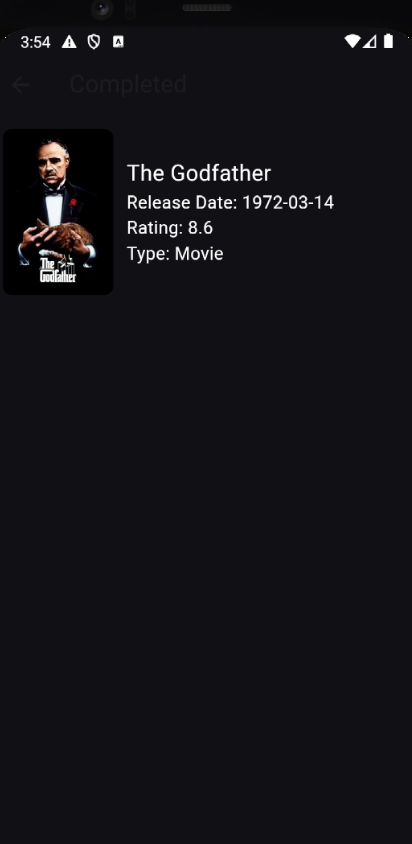


Рисунок 5.10 – Страница сохраненных фильмов

Также пользователь может просмотреть трейлер фильма, для этого необходимо нажать на кнопку с треугольником после чего пользователь попадет на www.youtube.com с нужным трейлером.

Результат можно увидеть на рисунке 5.11.

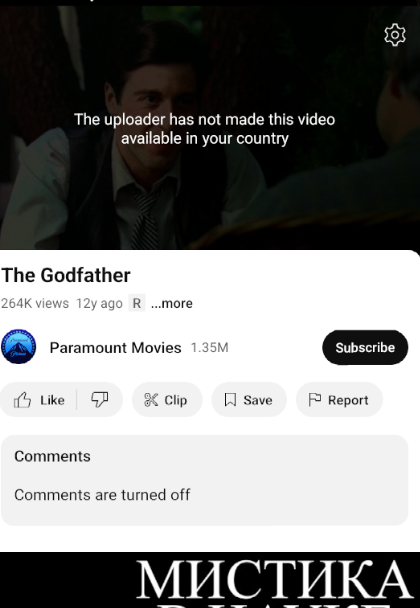


Рисунок 5.11 – Страница трейлера

Также в личном кабинете при желании пользователь может вернуться на страницу авторизации.

# Заключение

В результате выполнения данной курсовой работы было разработано мобильное приложение «Nepflix», которое предоставляет собой приложения для фильмов.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* регистрация;
* авторизация;
* просмотр информации о фильмах;
* просмотр трейлеров;
* добавление фильмов в списки просмотренных;
* поиск фильмов;
* бронирование билета;

Администратору предоставляет следующие функциональные возможности:

* авторизация;
* создание, удаление, редактирование фильмов;
* просмотр информации о фильмах;
* просмотр трейлеров;
* поиск фильмов;
* бронирование билета;
* добавление фильмов в списки просмотренных;

Гость имеет возможности:

* просмотр информации о фильмах;
* поиск фильмов;
* просмотр трейлеров;

Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список использованных источников

1. Flutter [Электронный ресурс]. – Режим доступа: <https://docs.flutter.dev/tools/devtools/overview> – Дата доступа: 16.04.2024
2. Solutions on Stackoverflow [Электронный ресурс]. – Электронные данные. – Режим доступа: [https://stackoverflow.com.](https://stackoverflow.com./) – Дата доступа: 20.04.2024;
3. Чернышев, С.А. Основы Dart. 2-е издание / С.А. Чернышев. – Москва: Издательство "Наука", 2023. – 350 с.
4. Flutterfor [Электронный ресурс]. – Режим доступа: https://flutterfor.dev/ – Дата доступа: 20.03.2024
5. Dartflutter [Электронный ресурс]. – Режим доступа: https://dartflutter.ru/: 25.03.2024

# ПРИЛОЖЕНИЕ А

UML диаграмма использований

