Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

**РАЗРАБОТКА GAN нейросети**

Пояснительная записка к курсовому проекту

ПМ.01Разработка модулей программного обеспечения для компьютерных систем

МДК.02.01 Технология разработки программного обеспечения

НАТКиГ.202000.010.000ПЗ

Разработал:

студент группы ПР-21.106

Шестаков И.В.

2023

**Содержание**

[Введение 3](#_Toc117112970)

[1 Исследовательский раздел 3](#_Toc117112971)

[1.1 Описание предметной области 4](#_Toc117112972)

[1.2 Образ клиента 5](#_Toc117112973)

[1.3 Сценарии 5](#_Toc117112974)

[1.4 Сбор и анализ прототипов 5](#_Toc117112975)

[2 Проектирование приложения 11](#_Toc117112976)

[2.1 UI/UX дизайн проекта 11](#_Toc117112977)

[2.2 Выбор технологии, языка и среды программирования 16](#_Toc117112978)

[3 Разработка мобильного приложения 17](#_Toc117112979)

[3.1 Разработка базы данных 17](#_Toc117112980)

[3.2 Разработка мультимедийного контента 17](#_Toc117112981)

[3.3 Описание используемых плагинов 20](#_Toc117112982)

[3.4 Описание разработанных процедур и функций 23](#_Toc117112982)

[4 Тестирование 30](#_Toc117112983)

[4.1 Протокол тестирования дизайна приложения 30](#_Toc117112984)

[4.2 Протокол тестирования функционала приложения 34](#_Toc117112985)

[Заключение 39](#_Toc117112986)

[Библиография 40](#_Toc117112987)

[Приложение А 41](#_Toc117112988)

Введение

Нейронные сети вдохновлены биологической нейронной системой человека. Нейронные сети состоят из искусственных нейронов, которые объединяются в слои и обрабатывают данные, выполняя различные вычислительные задачи. Они используются для решения широкого спектра задач, включая распознавание образов, классификацию, регрессию, генерацию данных и многое другое.

Генеративно-состязательные сети в последние годы привлекают пристальное внимание исследователей в области машинного обучения и компьютерного зрения. Успехи в разработке GAN открывают путь к созданию более "интеллектуальных" систем, способных решать сложные прикладные задачи. Поэтому развитие GAN является важным направлением в развитии искусственного интеллекта.

Целью курсового проекта является создания генерирующей изображения нейросети.

Задачами курсового проекта в связи с указанной целью являются:

* изучение предметной области;
* рассмотрение проекта с точки зрения пользователя для выявления функций приложения;
* написание кода нейросети;
* тестирование полученного продукта.

Объект исследования – генерирующая нейросеть.

# Исследовательский раздел

## Описание предметной области

Генеративно-состязательные сети (GAN) — это класс нейросетевых архитектур, используемых для генерации синтетических изображений. В GAN состязаются друг с другом две нейронные сети: генератор и дискриминатор.

Генератор — это нейросеть, которая генерирует новые изображения на основе случайного входного вектора шума. Ее задача - создавать реалистичные изображения, которые нельзя отличить от настоящих.

Дискриминатор — это нейросеть, которая принимает на вход как настоящие, так и сгенерированные изображения. Ее задача - определить, является ли изображение настоящим или сгенерированным.

Пользователь, генерирующий изображение может:

1. Ввод данных:

* Подготовка входных данных (описание желаемого результата);
* Уточнение положительных и отрицательных запросов.

1. Настройка параметров генерации:

* Выбор размера генерируемого изображения;
* Задание количества генерируемых изображений;
* Настройка дополнительных параметров генератора и дискриминатора.

1. Запуск генерации:

* Отправка входных данных в модель GAN;
* Получение на выходе сгенерированных нейросетью изображений.

В процессе обучения GAN дискриминатор учится все лучше отличать настоящие изображения от сгенерированных, а генератор совершенствуется в генерации все более реалистичных изображений, чтобы одурачить дискриминатор.

Процесс обучения GAN, может быть сложным и требовать значительных вычислительных ресурсов. Также GAN могут страдать от проблем, как коллапс моды или нестабильность обучения. Поэтому при реализации GAN необходимо тщательно настраивать гиперпараметры и архитектуру сетей.

Процесс обучения нейросети, можно разделить на несколько ключевых этапов:

1. Подготовка данных:

* Собираются и подготавливаются данные, которые будут использованы для обучения нейросети.

1. Создание генератора и дискриминатора:

* Определяются архитектуры генератора и дискриминатора.

1. Определение функций потерь:

* Выбираются функции потерь для обеих нейросетей. Для GAN применяют бинарную кросс-энтропию.

1. Обучение:

* Генератор создает синтетические изображения на основе случайных входных данных;
* Дискриминатор оценивает синтетические и реальные изображения, предсказывая, является ли изображение реальным или сгенерированным;
* Генератор и дискриминатор обновляют свои веса на основе результатов.

1. Настройка параметров:

* Подбираются гиперпараметры, такие как скорость обучения, количество слоёв и нейронов в сетях.

Обучение GAN требует большого набора реальных изображений в качестве примеров. Чем больше и разнообразнее тренировочные данные, тем лучше GAN сможет научиться генерировать реалистичные изображения.

GAN могут использоваться для генерации фотографических изображений людей, животных, пейзажей и других объектов. Они позволяют создавать высококачественные изображения, неотличимые от реальных фотографий.

GAN открывают большие возможности для синтеза фотореалистичных изображений, которые могут использоваться в различных областях - от развлечений до науки.

В заключение, генеративно-состязательные сети (GAN) представляют собой захватывающую область искусственного интеллекта, которая вдохновлена биологической нейронной системой человека. Эти сети, состоящие из генератора и дискриминатора, открывают огромные возможности для создания синтетических изображений, неотличимых от настоящих фотографий.

Процесс обучения GAN может быть сложным и требовать внимательной настройки, но результаты могут быть впечатляющими. Они могут использоваться в различных областях, включая искусство, развлечения и научные исследования, что делает их важными для развития технологий и креативности. С развитием архитектур и методов обучения GAN можно ожидать, что эта область будет продолжать вдохновлять исследователей и инженеров, открывая новые горизонты в создании фотореалистичных изображений и других приложениях искусственного интеллекта.

## Образ клиента

Клиентами являются зарегистрированные пользователи видеохостинга. Приложение предназначено для людей различных возрастов, но основным контингентом будет молодёжь от 12 до 30 лет. Приложение удобно как для людей, которые часто делятся видеороликами с другими, так и для тех, кто желает удобно просматривать видеоролики.

## Сценарии

Молодой человек ведет социальную активность в интернете, и ему необходимо быстро загружать видеоролики, чтобы поделиться ими. До этого он загружал видеоролики в мессенджер и отправлял их по отдельности каждому своему знакомому. Но ему будет гораздо легче делиться видеороликами, если на его устройстве будет приложение видеохостинга, в которое можно быстро их загружать и делиться со всеми знакомыми одной ссылкой.

В другом случае, молодой человек, наоборот, любит посмотреть видеоролики, загруженные в сеть другими пользователями и сохранять их. Но для этого он пользуется несколькими сайтами, чтобы следить за всем, что интересует его. А для того, чтобы сохранять видео, приходится скачивать их на своё устройство. Но куда удобнее иметь мобильное приложение, которое позволит отсортировать видеоролики по интересующим темам, и сохранять их в разделе «Избранное».

Пожилой человек большой фанат любительского кинематографа, ведь раньше он и сам занимался этим. Он любит смотреть творчество других людей, однако постоянно приходится искать видеоролики, которые он не смотрел на разных сайтах. Но ему было бы куда удобнее, если бы было приложение, которое собирает видеоролики в одном месте.

## Сбор и анализ прототипов

В GooglePlay и AppStore существует два приложения, выполняющих роль видеохостинга. Первое – YouTube (Американское приложение), второе – RUTUBE (Русское приложение). Оба этих приложения имеют возможность просматривать видеоролики, сохранять понравившиеся и загружать собственные видеоролики с возможность поделиться ими. Рассмотрим эти два приложения подробнее и сравним их.

Оба этих приложения имеют главный экран, на котором собраны видеоролики от разных авторов (рисунок 1).

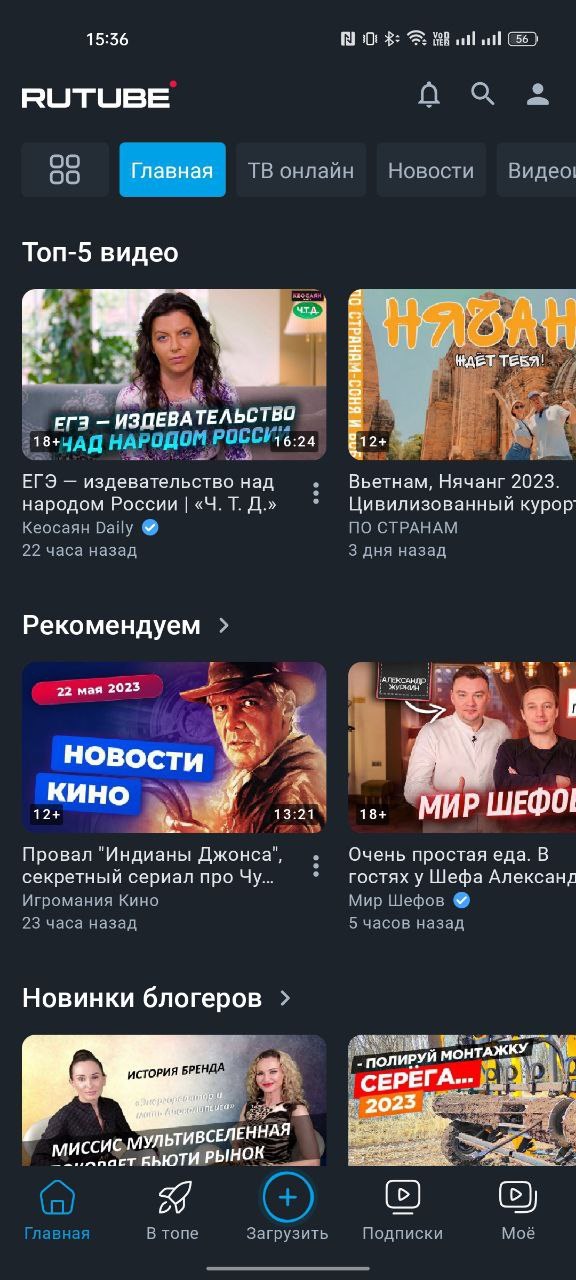
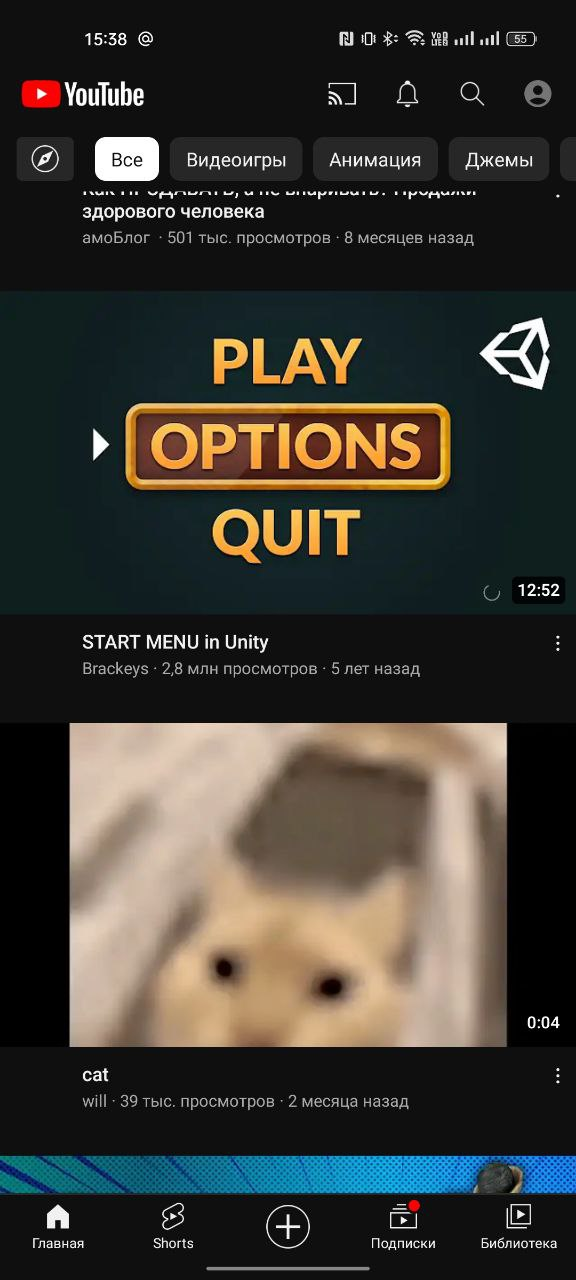
 

Рисунок 1 – Главный экран приложений

В приложениях на главном экране можно увидеть нижнюю и верхнюю панель, отвечающие за навигацию по приложению. На верхней панели в обоих приложениях почти одинаковые кнопки:

* уведомления;
* поиск;
* профиль.

Нижняя панель отличается только элементами, к которым она перенаправляет, но общими являются:

* главная;
* загрузить;
* подписки.

Вывод видеороликов отличается. В приложении «RUTUBE» - видео выводятся по группам, а в «YouTube» – как общий список рекомендованных видео. Логотип находится в одинаковом месте – в верхнем левом углу.

Так же в них реализована функция добавления видеороликов в «Понравившиеся» (рисунок 2):

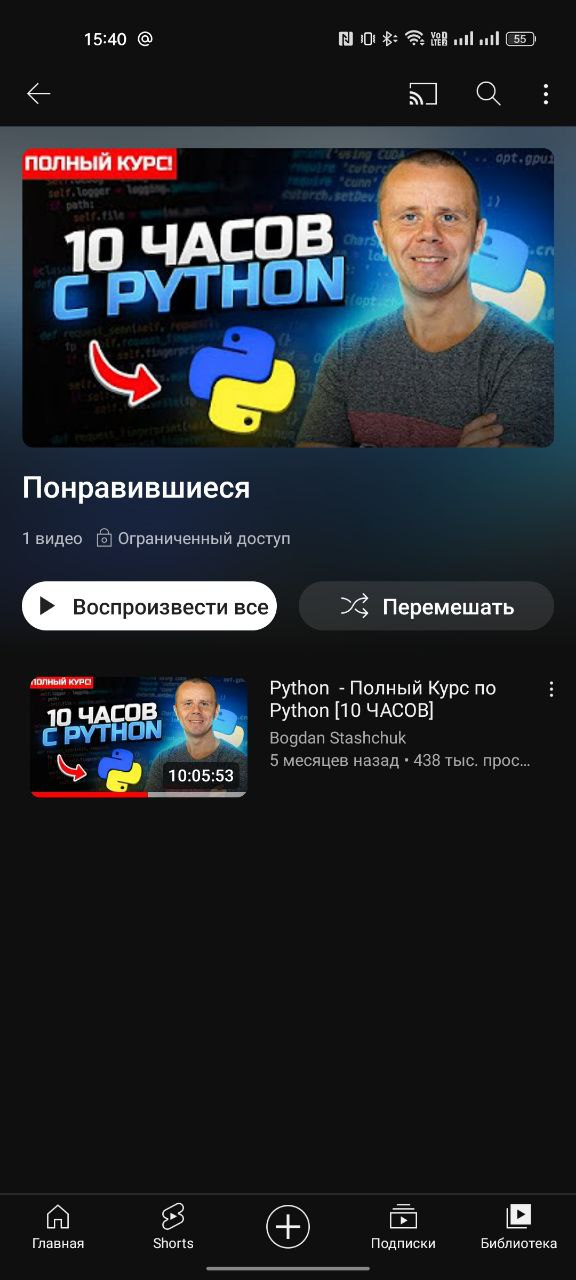
 

Рисунок 2 – Экран с понравившимися видеороликами

Все добавленные в «Понравившиеся» видеоролики, отображаются на отдельной вкладке. Их так же можно оттуда удалить или посмотреть повторно. Однако в приложении «RUTUBE» на этой вкладке отсутствует верхняя панель. Поэтому в нём нельзя найти нужный видеоролик с помощью поиска, что не очень удобно, если список понравившегося очень большой.

Окно профиля, с возможностью перехода к настройкам и т.п. в этих приложениях сильно отличаются своими функциями (Рисунок 3):

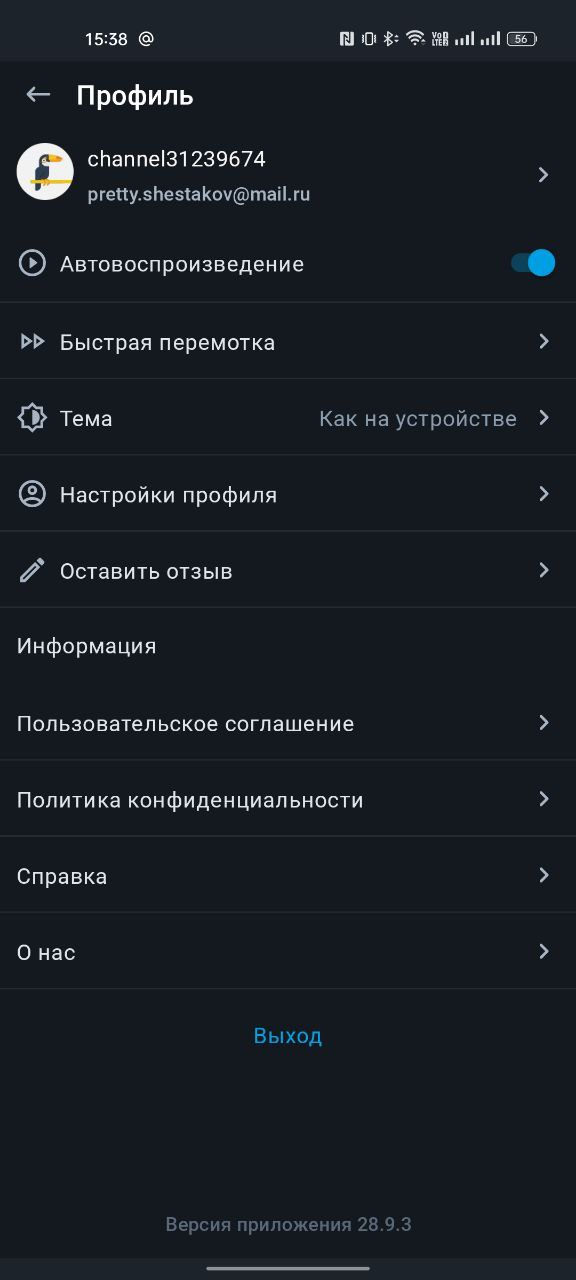
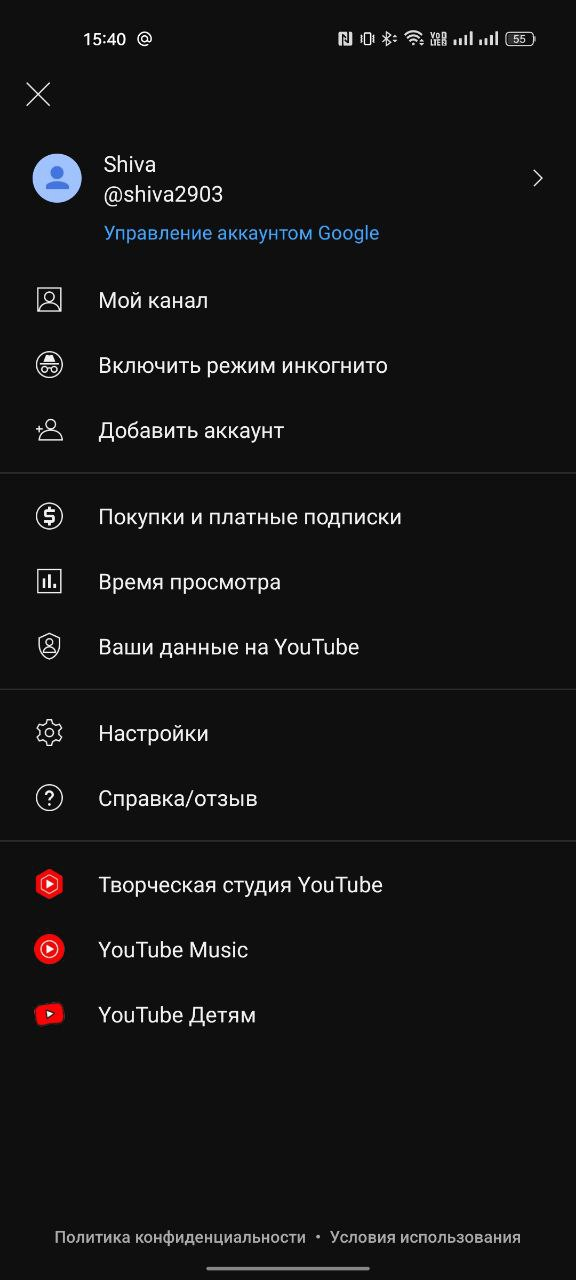
 

Рисунок 3 – Экран профиля пользователя

У обоих приложений свои страницы профиля, кардинально отличающиеся друг от друга. Из общего у них только два элемента:

* справка;
* переход на канал.

В приложении «YouTube» элементы профиля более сгруппированы. В приложении же «RUTUBE» нет разделения на группы, из-за чего ориентироваться труднее. Настройки, которые на «YouTube» вынесены в отдельную вкладку, в «RUTUBE» находятся прямо в самом окне профиля. Именно поэтому, экран профиля «YouTube», является более удобным для понимания и использования пользователем, чем «RUTUBE».

Сравнение по основным критериям данных приложений представлено в таблице 1.

Таблица 1 – Сравнение приложений

|  |  |  |
| --- | --- | --- |
| Параметры | YouTube | RUTUBE |
| Стоимость | Бесплатно | Бесплатно |
| Основной язык | Английский | Русский |
| Просмотр видеороликов | Есть | Есть |
| Возможность сохранять видеоролики в «Избранное» | Есть | Есть |
| Подробное описание к контенту | Есть | Есть |
| Наличие поиска | Есть | Есть |
| Выбор темы интерфейса | Есть | Есть |
| Список просмотренного | Есть | Есть |
| Оценки видеороликов | Есть | Есть |
| Возможность загружать свои видеоролики | Есть | Есть |
| Возможность продавать свой контент | Есть | Отсутствует |

Рассмотрев пару приложений, выполняющих похожие задачи, было решено написать приложение, которое имело бы простой интерфейс без лишних функций и с акцентом внимания на важных для пользователя элементах, в котором можно смотреть видеоролики и добавлять их в «Избранное». За основу будут взяты разные элементы двух приложений, так как каждое из них имеет как свои плюсы, так и минусы.

# Проектирование приложения

* 1. **UI/UX дизайн приложения**

Дизайн проекта разработан в программе Figma.

Для проекта были определены основные экраны:

* регистрация;
* авторизация;
* главный экран с видеороликами;
* личный кабинет;
* понравившиеся видеоролики;
* загрузка нового видеоролика;
* загруженные видеоролики;
* просмотр видеоролика.

Для тем приложения определены следующие две цветовые схемы. Первая тема реализуется в тёмных тонах, поэтому в ней основными цветами являются: серый, черный, белый (Рисунок 4).

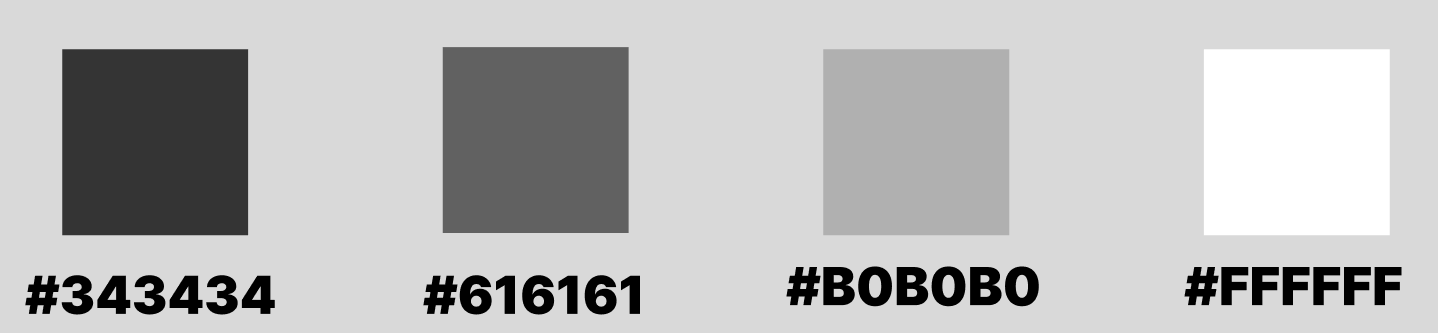


Рисунок 4 – Тёмная цветовая схема

Вторая тема будет светлого цвета: белый и светло-серый (Рисунок 5).



Рисунок 5 – Светлая цветовая схема

Но стоит заметить, что данные цвета, разбавленные изображениями, не выглядят скучно, а весьма лаконично. Необходимость разделения цветовых схем на тёмную и светлую, заключается в том, что пользователю может быть удобнее использовать какую-либо из них в разное время суток.

Логотип приложения общий как для тёмной темы, так и для светлой. Он делится на две половины разных цветов, на которых написано название приложения «VideoHost».

Ниже на рисунке 7 представлен логотип видеохостинга.



Рисунок 6 – Логотип видеохостинга

Цветовая схема логотипа состоит из: красного и белого (Рисунок 7). Это броские цвета, которые могут привлечь внимание пользователя к приложению.

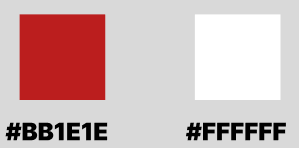


Рисунок 7 – Цветовая схема логотипа

Определившись с цветовой схемой приложения и создав его логотип, был разработан дизайн следующих экранов:

* авторизация (Login);
* регистрация (Registration)
* главный экран (Main)
* профиль (Profile)
* понравившиеся видеоролики (Like)
* загрузка видеороликов (UploadVideo)
* загруженные видеоролики (DoneUpload)
* просмотра видеороликов (Play)

Ниже на рисунке 8 представлен дизайн приложения с тёмной цветовой схемой.





Рисунок 8 – Дизайн приложения с тёмной темой

Ниже на рисунке 9 представлен дизайн приложения со светлой цветовой схемой.





Рисунок 9 – Дизайн приложения со светлой темой

На рисунке 10 показано перемещение пользователя в приложении.

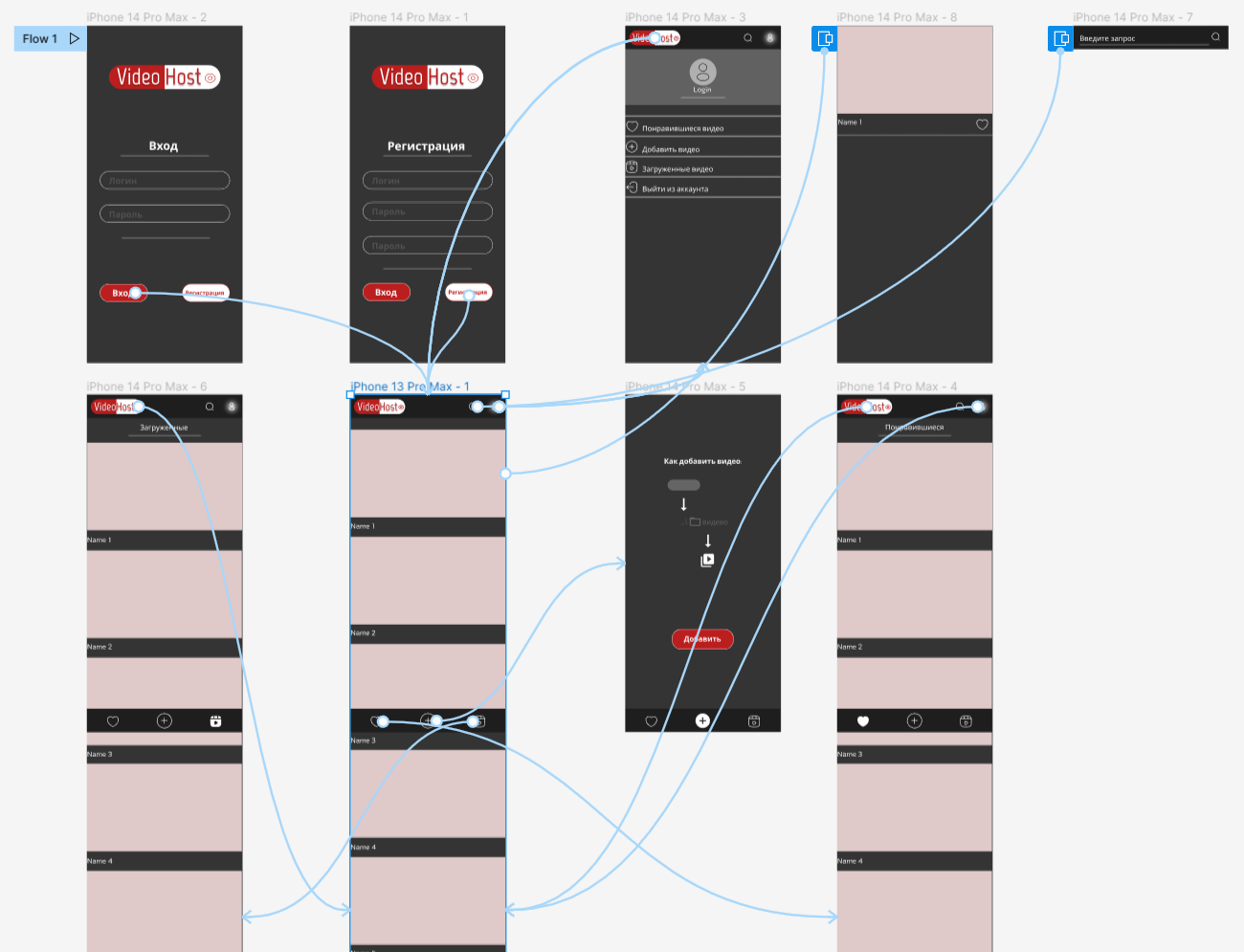


Рисунок 10 – Перемещение пользователя в приложении

При первом запуске приложения пользователь попадает на экран авторизации. Если у него уже есть аккаунт, то он может ввести свои данные и нажать на кнопку «Вход», после чего он попадёт на главный экран. В случае, если у него нет аккаунта, он может нажать на кнопку «Регистрация», которая откроет экран регистрации, где он сможет создать новый аккаунт. С каждого экрана можно перейти в любую часть приложения, однако на рисунке 7 обозначены только действия при нажатии кнопок, это сделано для того, чтобы не нагружать рисунок одинаковыми действиями. При нажатии на кнопку «Понравившееся» пользователь перейдёт на экран со списком видеороликов, которые ему понравились. Нажимая на кнопку «Профиль» откроется экран профиля, на котором пользователь может выбрать другие окна приложения. При нажатии на область превью видеоролика, пользователь начнет его просмотр. Если пользователь нажмёт на кнопку «Загрузить видеоролик», то он перейдёт на окно с его загрузкой. А при нажатии на кнопку «Загруженные видео», откроется список видео, которые загрузил сам пользователь.

## Выбор технологии, языка и среды программирования

Средой программирования выбрана программа Android Studio, так как она довольно удобна для создания приложения и является довольно популярной, поэтому в случае возникновения ошибки, легче найти способ её решения. Языком программирования является Kotlin, так как это новый и развивающийся язык программирования, к которому очень часто добавляются новые удобные функции, которые пригодятся во время разработки.

Используемый сервис для базы данных и аутентификации – Firebase. Это бесплатная база данных, которая подходит под поставленную задачу. Она довольно быстрая и легка для понимания. Так же в ней представлены все необходимые функции: Authentication (для регистрации и авторизации пользователей), Realtime Database (для синхронизации данных), Storage (для хранения файлов в системе).

# Разработка мобильного приложения

## Разработка базы данных

В качестве разрабатываемой базы данных выбрана облачная база данных Firebase, интегрируемая в Android Studio. В ней данные хранятся в формате JSON, то есть, она является NoSQL – базой данных. Пример хранения данных в базе представлен на рисунке 11.



Рисунок 11 – Страница Firebase с информацией о видео

База данных организована в виде древовидной структуры, где каждое видео представлено уникальным идентификатором (ID), который служит вершиной дерева. Каждому ID соответствует набор информации о конкретном видеоролике.

В базе данных хранится вся информация о видеоролике: название, время загрузки и ссылка на него.

## Разработка мультимедийного контента

Весь мультимедийный контент разрабатывался с помощью языка разметки XML. Вёрстка выполнялась по дизайну, разработанному ранее в приложении Figma. Однако, по мере разработки, в дизайн были внесены изменения.

Иконки, кнопки и другие элементы приложения, были импортированы в проект в виде XML-файлов (Рисунок 12). Такой способ хранения уменьшает вес приложения, а также, избавляет от проблем с потерей качества мультимедийного контента. Все ресурсы хранятся в папке «drawable»

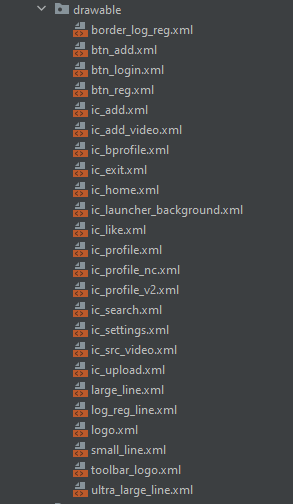


Рисунок 12 – Мультимедийный контент

Так как в приложении используется две цветовые схемы, то необходимо хранить обе палитры в файле colors.xml (Листинг 1)

Листинг 1 –Палитра цветов

<colorname="purple\_200">#FFBB86FC</color>

<colorname="purple\_500">#FF6200EE</color>

<colorname="purple\_700">#FF3700B3</color>

<colorname="teal\_200">#FF03DAC5</color>

<colorname="teal\_700">#FF018786</color>

<colorname="black">#FF000000</color>

<colorname="white">#FFFFFFFF</color>

<colorname="BgNtheme">#383838</color>

<colorname="BgNthemeTop">#1e1e1e</color>

<colorname="sel\_tabbar">#FFFFFF</color>

<colorname="unsel\_tabbar">#707070</color>

<colorname="text\_color">#DCDCDC</color>

<colorname="BackColorProfile">#616161</color>

<colorname="BackColorProfileImage">#B0B0B0</color>

<colorname="Red">#BB1E1E</color>

<colorname="Grey">#DDDDDD</color>

Эти цвета используются для описания цветов темы приложения. Темы приложений описываются в двух файлах: themes.xml–для светлой темы, а themes.xml (night) – для тёмной темы. На листинге 2 представлены тёмная и светлая тема.

Листинг 2 – Светлая и тёмная тема приложения

<style name="Theme.VideoHost" parent="Theme.MaterialComponents.DayNight.NoActionBar">  
 <!-- Primary brand color. -->  
 <item name="colorPrimary">@color/sel\_tabbar</item>  
 <item name="colorPrimaryDark">@color/BgNthemeTop</item>  
 <item name="android:textColorPrimary">@color/white</item>  
 <item name="android:textColor">@color/text\_color</item>  
 <item name="colorOnSurface">@color/unsel\_tabbar</item>  
 <item name="colorAccent">@color/unsel\_tabbar</item>  
 <!-- Secondary brand color. -->  
 <item name="colorSecondary">@color/Red</item>  
 <item name="android:windowBackground">@color/BgNtheme</item>  
 <item name="windowActionBar">false</item>  
 <item name="windowNoTitle">true</item>  
  
 <!-- Customize your theme here. -->  
</style>  
<style name="txt\_header">  
 <item name="fontFamily">@font/montserrat\_bold</item>  
 <item name="android:textSize">16sp</item>  
</style>  
  
<style name="txt\_log\_reg">  
 <item name="fontFamily">@font/montserrat\_bold</item>  
 <item name="android:textSize">36sp</item>  
</style>  
  
<style name="txt\_profile">  
 <item name="fontFamily">@font/montserrat\_medium</item>  
 <item name="android:textSize">18sp</item>  
 <item name="textAllCaps">false</item>  
</style>  
  
<style name="txt\_mail\_pass">  
 <item name="fontFamily">@font/montserrat\_medium</item>  
 <item name="android:textSize">16sp</item>  
 <item name="textAllCaps">false</item>  
</style>  
  
<style name="Profile\_Theme">  
 <!-- Primary brand color. -->  
 <item name="colorPrimary">@color/sel\_tabbar</item>  
 <item name="colorPrimaryDark">@color/BgNthemeTop</item>  
 <item name="android:textColorPrimary">@color/white</item>  
 <item name="android:textColor">@color/text\_color</item>  
 <item name="colorOnSurface">@color/BackColorProfileImage</item>  
 <item name="colorAccent">@color/BackColorProfile</item>  
 <!-- Secondary brand color. -->  
 <item name="colorSecondary">@color/Red</item>  
 <item name="android:windowBackground">@color/BgNtheme</item>  
 <item name="windowActionBar">false</item>  
 <item name="windowNoTitle">true</item>  
</style>  
  
<style name="AddVideo\_Theme">  
 <!-- Primary brand color. -->  
 <item name="colorPrimary">@color/sel\_tabbar</item>  
 <item name="colorPrimaryDark">@color/sel\_tabbar</item>  
 <item name="android:textColorPrimary">@color/white</item>  
 <item name="android:textColor">@color/text\_color</item>  
 <item name="colorOnSurface">@color/BackColorProfileImage</item>  
 <item name="colorAccent">@color/BackColorProfile</item>  
 <!-- Secondary brand color. -->  
 <item name="colorSecondary">@color/Red</item>  
 <item name="android:windowBackground">@color/BgNtheme</item>  
 <item name="windowActionBar">false</item>  
 <item name="windowNoTitle">true</item>  
</style>

То, какая тема будет использоваться, пользователь решает сам. Кому-то может быть неприятно использовать какую-то определенную тему, поэтому выбор между тёмной и светлой темой – достойная черта приложения. Именно поэтому, в приложение есть выбор, какую тему использовать.

## Описание используемых плагинов

В проекте используются библиотеки с различными компонентами и функциями. Список всех библиотек, а также их описание представлен в таблице 2.

Таблица 2 – Библиотеки и их описание

|  |  |
| --- | --- |
| Библиотека | Описание |
| com.google.firebase:firebase-bom:31.3.0 | Обеспечивает согласованную версию всех Firebase-библиотек |

Продолжение таблицы 2

|  |  |
| --- | --- |
| com.google.firebase:firebase-auth-ktx | Библиотека для аутентификации пользователей через различные методы (электронная почта, социальные сети и др.). |
| com.google.firebase:firebase-analytics-ktx | Библиотека для сбора и анализа аналитических данных о приложении. |
| androidx.core:core-splashscreen:1.0.1 | Библиотека для создания и управления экраном загрузки (splashscreen) в приложении. |
| androidx.appcompat:appcompat:1.6.1 | Библиотека, обеспечивающая обратную совместимость с новыми функциями и возможностями Android для более старых версий платформы. |
| com.google.android.material:material:1.9.0 | Библиотека, содержащая MaterialDesign компоненты пользовательского интерфейса. |
| androidx.legacy:legacy-support-v4:1.0.0 | Библиотека, обеспечивающая обратную совместимость с более старыми версиями Android и предоставляющая поддержку различных функций и компонентов, которые были введены в более новых версиях платформы. |
| androidx.recyclerview:recyclerview:1.3.0 | Библиотека, предоставляющая RecyclerView - мощный компонент пользовательского интерфейса, который позволяет эффективно отображать списки данных, обеспечивая повторное использование представлений и поддержку различных макетов. |
| androidx.annotation:annotation:1.6.0 | Библиотека, содержащая аннотации, используемые во время компиляции для предоставления информации о коде, например, для обозначения намерений, проверок на статический анализ или документации. |

Окончание таблицы 2

|  |  |
| --- | --- |
| com.google.firebase:firebase-database-ktx:20.2.1 | Библиотека для работы с FirebaseRealtimeDatabase, которая предоставляет возможность хранения и синхронизации данных в реальном времени на облачной платформе Firebase. |
| com.google.firebase:firebase-storage-ktx:20.2.0 | Библиотека для работы с FirebaseStorage, которая предоставляет возможность хранения и управления медиафайлами (например, изображениями и видео) на облачной платформе Firebase. |
| junit:junit:4.13.2 | Библиотека для написания и запуска модульных тестов в Java. |
| androidx.test.ext:junit:1.1.5 | Библиотека, расширяющая возможности JUnit для тестирования Android-приложений. |
| androidx.test.espresso:espresso-core:3.5.1 | Библиотека для автоматизированного UI-тестирования Android-приложений с использованием Espresso фреймворка. |
| androidx.navigation:navigation-fragment-ktx:2.5.3, androidx.navigation:navigation-ui-ktx:2.5.3 | Библиотеки, предоставляющие навигационные компоненты для управления переходами и навигацией между фрагментами и активностями в приложении. |
| androidx.lifecycle:lifecycle-livedata-ktx:2.6.1, androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.1 | Библиотеки, обеспечивающие жизненный цикл компонентов Android (таких как активности и фрагменты) и предоставляющие LiveData и ViewModel для удобной работы с данными в приложении. |

Данные библиотеки являются неотъемлемой частью приложения, обеспечивая его правильное функционирование и реализацию всех задуманных возможностей. Без этих библиотек приложение не сможет работать в полной мере, так как они предоставляют необходимые инструменты и функции, которые необходимы для его работы.

## Описание разработанных процедур и функций

В приложении реализованы следующие методы (таблица 3):

Таблица 3 – Методы приложения

|  |  |
| --- | --- |
| Метод | Описание |
| uploadVideoFirebase() | Загрузка видеоролика на сервер по средствам запросов |
| videoPickDialog() | Выбор метода загрузки приложения |
| requestCameraPermission() | Запрос разрешения на использование камеры |
| checkCameraPermissions() | Проверка разрешения на использование камеры |
| videoPickGallery() | Выбор видео из галереи |
| videoPickCamera() | Запись видео с помощью камеры |
| replaceFragment() | Размещение определенного фрагмента на главном экране |
| onBindViewHolder() | Получение информации о видео для его вывода на экране |
| setVideoUrl() | Получение ссылки на видео |
| toggleFavorite() | Добавление видео в избранное |

Так же в приложении есть классы, которые не имеют методов (таблица 4):

Таблица 4 – Классы приложения

|  |  |
| --- | --- |
| Класс | Описание |
| RegisterActivity | Регистрация пользователя в системе |
| ProfileActivity | Окно профиля с выводом токена пользователя |
| LoginActivity | Авторизация в приложение |

Авторизация в приложении реализована по средству запросов в базу данных. Ниже представлен код авторизации (листинг 3).

Листинг 3 – Авторизация в приложении

// Привязываем действие к кнопке входа (btnLog)  
btnLog.setOnClickListener **{** when{ // Проверяем, пустое ли поле электронной почты (txtMail)  
 TextUtils.isEmpty(txtMail.*text*.toString().*trim* **{ it** <= ' ' **}**) -> {  
 Toast.makeText( // Если поле пустое, выводим сообщение с просьбой заполнить поле электронной почты  
 this@LoginActivity,  
 "Пожалуйcта, заполните поле - e-mail.",  
 Toast.*LENGTH\_SHORT* ).show()  
 } // Проверяем, пустое ли поле пароля (txtPass)  
 TextUtils.isEmpty(txtPass.*text*.toString().*trim* **{ it** <= ' ' **}**) -> {  
 Toast.makeText( // Если поле пустое, выводим сообщение с просьбой заполнить поле пароля  
 this@LoginActivity,  
 "Пожалуйcта, заполните поле - Пароль.",  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 else -> { // Если поля электронной почты и пароля заполнены, выполняем вход пользователя  
 val email: String = txtMail.*text*.toString().*trim* **{ it** <= ' ' **}** val password: String = txtPass.*text*.toString().*trim* **{ it** <= ' ' **}** // Входим как юзер.  
 FirebaseAuth.getInstance().signInWithEmailAndPassword(email, password)  
 .addOnCompleteListener **{** task **->** //Если вход прошёл успешно  
 if (task.*isSuccessful*) {  
  
 Toast.makeText(  
 this@LoginActivity,  
 "Вы успешно авторизировались.",  
 Toast.*LENGTH\_SHORT* ).show()  
 // Создаём интент для перехода на главный экран (MainActivity)  
 val intent = Intent (this@LoginActivity, MainActivity::class.*java*)  
 intent.*flags* = Intent.*FLAG\_ACTIVITY\_NEW\_TASK* or Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK* intent.putExtra("user\_id", FirebaseAuth.getInstance().*currentUser*!!.*uid*)  
 intent.putExtra("email\_id" , email)  
 startActivity(intent)  
 finish()  
 } else {  
 // Если что-то пошло не так и вылезла ошибка  
 Toast.makeText(  
 this@LoginActivity,  
 task.*exception*!!.message.*toString*(),  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 **}** }  
 }  
**}**

В случае, если пользователю нужно зарегистрироваться, сперва нужно перейти на экран регистрации (листинг 4).

Листинг 4 – Переход к другому экрану

fun GoReg(view: View) { // Переход к экрану регистрации  
 val regIntent = Intent(this, RegisterActivity::class.*java*)  
 startActivity(regIntent)  
 finish()  
}

При регистрации, в базу данных запросами заносятся новые данные пользователя (листинг 5).

Листинг 5 – Регистрация пользователя

// Привязываем действие к кнопке регистрации (btnReg)  
btnReg.setOnClickListener **{**

when{ // Проверяем, пустое ли поле электронной почты (txtMail)  
 TextUtils.isEmpty(txtMail.*text*.toString().*trim* **{ it** <= ' ' **}**) -> {  
 Toast.makeText( // Если поле пустое, выводим сообщение с просьбой заполнить поле электронной почты  
 this@RegisterActivity,  
 "Пожалуйcта, заполните поле - e-mail.",  
 Toast.*LENGTH\_SHORT* ).show()  
 } // Проверяем, пустое ли поле пароля (txtPass)  
 TextUtils.isEmpty(txtPass.*text*.toString().*trim* **{ it** <= ' ' **}**) -> {  
 Toast.makeText( // Если поле пустое, выводим сообщение с просьбой заполнить поле пароля  
 this@RegisterActivity,  
 "Пожалуйcта, заполните поле - Пароль.",  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 else -> {  
 val email: String = txtMail.*text*.toString().*trim* **{ it** <= ' ' **}** val password: String = txtPass.*text*.toString().*trim* **{ it** <= ' ' **}** // Создаём Юзера по введённым данным.  
 FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password)  
 .addOnCompleteListener(

*OnCompleteListener* **{** task **->** //Если регистрация прошла успешно  
 if (task.*isSuccessful*) {  
 // Firebase регистрирует юзера  
 val firebaseUser: FirebaseUser = task.*result*!!.*user*!!  
  
 Toast.makeText( // Выводим сообщение об успешной регистрации  
 this@RegisterActivity,  
 "Регистрация прошла успешно.",  
 Toast.*LENGTH\_SHORT* ).show()  
 // Создаём интент для перехода на главный экран (MainActivity)  
 val intent = Intent (this@RegisterActivity, MainActivity::class.*java*)  
 intent.*flags* = Intent.*FLAG\_ACTIVITY\_NEW\_TASK* or Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK* intent.putExtra("user\_id" , firebaseUser.*uid*)  
 intent.putExtra("email\_id" , email)  
 // Запускаем главный экран и закрываем текущий экран  
 startActivity(intent)  
 finish()  
 } else {  
 // Если что-то пошло не так и вылезла ошибка  
 Toast.makeText(  
 this@RegisterActivity,  
 task.*exception*!!.message.*toString*(),  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 **}** )  
 }  
 }  
**}**

При переходе на главный экран, пользователю выводятся все видеоролики, загруженные в базу данных (листинг 6). Они выводятся с помощью компонента для отображения элементов списка RecyclerView.

Листинг 6 – Вывод видеороликов

private fun loadVideosFromFirebase() {  
 videoArrayList = ArrayList()  
 // Получаем ссылку на базу данных Firebase  
 val ref = FirebaseDatabase.getInstance("https://videohost-project-default-rtdb.europe-west1.firebasedatabase.app").getReference("Videos")  
 ref.addValueEventListener(object : ValueEventListener{  
  
 override fun onDataChange(snapshot: DataSnapshot) {  
 // Очищаем список видео перед загрузкой новых данных  
 videoArrayList.clear()  
 // Проходим по всем дочерним элементам снимка данных  
 for (ds in snapshot.*children*){  
 // Получаем объект видео из снимка данных  
 val modelVideo = ds.getValue(ModelVideo::class.*java*)  
 // Добавляем видео в список  
 videoArrayList.add(modelVideo!!)  
 }  
 // Сортируем список видео в порядке убывания по временной метке  
 videoArrayList.*sortByDescending* **{ it**.timestamp **}** // Создаём адаптер для отображения видео  
 adapterVideo = AdapterVideo(requireContext(), videoArrayList, firebaseAuth)  
 val videosRv: RecyclerView = *view*!!.findViewById(R.id.*videosRv*)  
 videosRv.*adapter* = adapterVideo  
 }  
 // В случае ошибки при получении данных из базы данных Firebase, выводим сообщение об ошибке  
 override fun onCancelled(error: DatabaseError) {  
 Toast.makeText(*context*, "Database error: " + error.*message*, Toast.*LENGTH\_SHORT*).show()  
 }  
 })  
}

В случае, если пользователь решит добавить видео в избранное, то с помощью запроса в базе данных создастся новая запись (листинг 7).

Листинг 7 – Добавление видео в избранное

private fun toggleFavorite(userId: String, videoId: String?, isFavorite: Boolean) {  
 // Получаем ссылку на базу данных Firebase  
 val databaseRef =  
 FirebaseDatabase.getInstance("https://videohost-project-default-rtdb.europe-west1.firebasedatabase.app").getReference("Users/$userId/Favorites/$videoId")  
 // Устанавливаем значение isFavorite в базе данных  
 databaseRef.child("isFavorite").setValue(isFavorite)  
 .addOnCompleteListener **{** task **->** if (task.*isSuccessful*) {  
 // Если успешно обновили избранное состояние, вызываем слушатель для обработки события переключения избранного  
 val listener = context as? FavoriteToggleListener  
 listener?.onFavoriteToggle(videoId!!, isFavorite)  
 } else {  
 // Обработка ошибки  
 Toast.makeText(context, "Failed to update favorite status", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**}

Добавленные в избранное видеоролики, выводятся на другом экране, по средствам двух запросов в базу данных (листинг 8).

Листинг 8 – Вывод избранных видеороликов

private fun loadFavoriteVideosFromFirebase() {  
 val favoriteVideoIds = ArrayList<String>()  
 videoArrayList = ArrayList()  
 // Получение идентификатора текущего пользователя  
 val userId = firebaseAuth.*currentUser*?.*uid* userId?.*let* **{** uid **->** // Ссылка на коллекцию избранных видео для текущего пользователя  
 val favoriteRef =  
 FirebaseDatabase.getInstance("https://videohost-project-default-rtdb.europe-west1.firebasedatabase.app")  
 .getReference("Users/$uid/Favorites")  
 favoriteRef.addValueEventListener(object : ValueEventListener {  
 override fun onDataChange(snapshot: DataSnapshot) {  
 favoriteVideoIds.clear()  
 // Перебор дочерних элементов снимка данных, чтобы получить идентификаторы избранных видео  
 for (ds in snapshot.*children*) {  
 val videoId = ds.*key*.*toString*() // Получаем идентификаторы видео  
 favoriteVideoIds.add(videoId)  
 }  
  
 // Запрос для получения списка видео из основной коллекции и их фильтрация по идентификаторам  
 val videosRef =  
 FirebaseDatabase.getInstance("https://videohost-project-default-rtdb.europe-west1.firebasedatabase.app")  
 .getReference("Videos")  
 videosRef.addValueEventListener(object : ValueEventListener {  
 override fun onDataChange(snapshot: DataSnapshot) {  
 videoArrayList.clear()  
 // Перебор дочерних элементов снимка данных основной коллекции видео  
 for (ds in snapshot.*children*) {  
 val modelVideo = ds.getValue(ModelVideo::class.*java*)  
 modelVideo?.*let* **{** video **->** // Проверка, является ли видео избранным (по идентификатору)  
 if (favoriteVideoIds.contains(ds.*key*.*toString*())) {  
 videoArrayList.add(video)  
 }  
 **}** }  
 videoArrayList.*sortByDescending* **{ it**.timestamp **}** adapterVideo.videoArrayList = videoArrayList  
 adapterVideo.notifyDataSetChanged()  
 }  
  
 override fun onCancelled(error: DatabaseError) {  
 Toast.makeText(  
 *context*,  
 "Database error: " + error.*message*,  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 })  
 }  
  
 override fun onCancelled(error: DatabaseError) {  
 Toast.makeText(*context*, "Database error: " + error.*message*, Toast.*LENGTH\_SHORT*)  
 .show()  
 }  
 })  
 **}**}

При добавлении видеоролика пользователь должен дать разрешение на использование камеры или галереи (листинг 9).

Листинг 9 – Запрос прав

//Запрос прав на камеру  
private fun requestCameraPermission(){  
 ActivityCompat.requestPermissions(  
 this,  
 cameraPermission,  
 CAMERA\_REQUEST\_CODE  
 )  
}  
  
private fun checkCameraPermissions():Boolean{  
 //Проверяет наличие прав  
 val result1 = ContextCompat.checkSelfPermission(  
 this,  
 android.Manifest.permission.*CAMERA* ) == PackageManager.*PERMISSION\_GRANTED* return result1  
}

// Переопределение метода для обработки результатов запроса разрешений  
override fun onRequestPermissionsResult(  
 requestCode: Int,  
 permissions: Array<out String>,  
 grantResults: IntArray  
) {  
 when(requestCode){  
 CAMERA\_REQUEST\_CODE ->  
 if (grantResults.*isNotEmpty*()) {  
 val cameraAccepted = grantResults[0] == PackageManager.*PERMISSION\_GRANTED* if (cameraAccepted) { // Разрешение на использование камеры получено  
 //Приняли  
 videoPickCamera()  
 } else{  
 Toast.makeText(this, "Права подтверждены", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults)  
}

После получения прав доступа пользователь может загрузить видеоролик. На листинге представлен код загрузки видеоролика в базу данных.

Листинг 10 – Загрузка видеоролика

private fun uploadVideoFirebase() {  
 progressDialog.show()  
 // Генерация временной метки для имени файла  
 val timestamp = ""+System.currentTimeMillis()  
 //Добавим время файлу  
 val filePathAndName = "Videos/video\_$timestamp"  
  
 val storageReference = FirebaseStorage.getInstance().getReference(filePathAndName)  
 // Загрузка выбранного видео в хранилище Firebase  
 storageReference.putFile(videoUri!!)  
 .addOnSuccessListener **{** taskSnapshot **->** val uriTask = taskSnapshot.*storage*.*downloadUrl* while (!uriTask.*isSuccessful*);  
 val downloadUri = uriTask.*result* if (uriTask.*isSuccessful*){  
 // Создаем HashMap с данными для Firebase  
 val hashMap = HashMap<String, Any>()  
 hashMap["id"] = "$timestamp"  
 hashMap["title"] = "$title"  
 hashMap["timestamp"] = "$timestamp"  
 hashMap["videoUri"] = "$downloadUri"  
 // Запись данных о видео в базу данных Firebase  
 val dbReference = FirebaseDatabase.getInstance("https://videohost-project-default-rtdb.europe-west1.firebasedatabase.app").getReference("Videos")  
 dbReference.child(timestamp)  
 .setValue(hashMap)  
 .addOnSuccessListener **{** taskSnapshot **->** progressDialog.dismiss() // Закрытие диалогового окна прогресса  
 Toast.makeText(this, "Видео загружено", Toast.*LENGTH\_SHORT*).show()  
 **}** .addOnFailureListener **{** e **->** progressDialog.dismiss()  
 Toast.makeText(this, "${e.message}", Toast.*LENGTH\_SHORT*).show()  
 **}** }  
 **}** .addOnFailureListener**{**e**->** progressDialog.dismiss()  
 Toast.makeText(this, "${e.message}", Toast.*LENGTH\_SHORT*).show()  
 **}**

# Тестирование

## Протокол тестирования дизайна приложения

Тестирование дизайна приложения проводится на самом минимальном (Android SDK 24) и на более позднем (Android SDK 33) с различной диагональю экранов для проверки разметки страниц и вёрстки приложения.

Примеры проверок отображения элементов на экране представлены на рисунках 13–17.

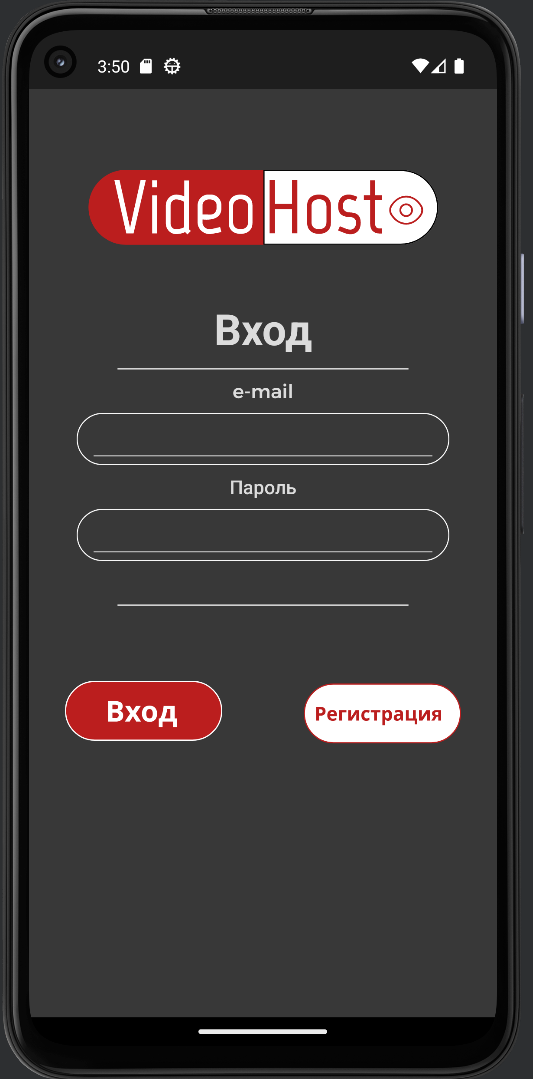
 

Рисунок 13 – Экраны авторизации

Элементы интерфейса в обоих случаях отображаются корректно.

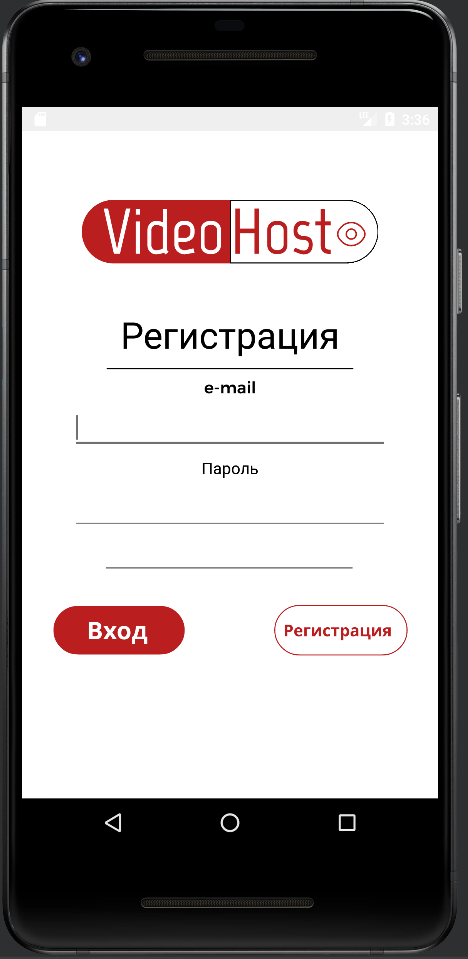
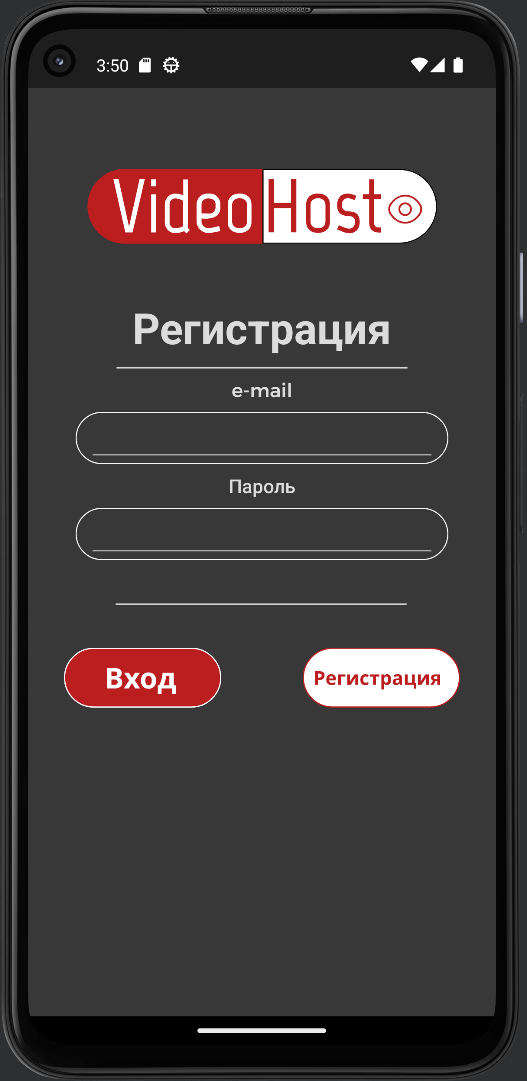
 

Рисунок 14 – Экраны авторизации

Экран авторизации так же отображается корректно на обоих устройствах, все элементы интерфейса расположены на своих местах.

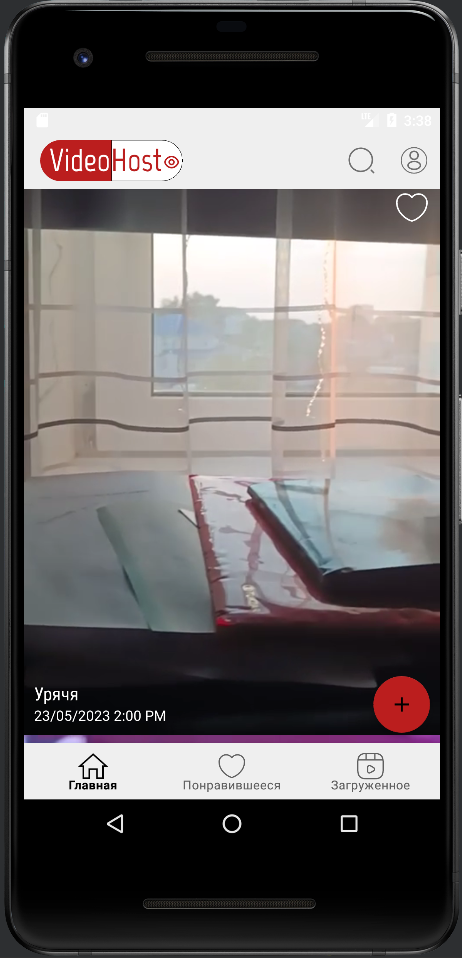
 

Рисунок 15 – Главные экраны

Главный экран тоже отображается корректно. Верхняя и нижняя панель нормальных размеров, элементы на них не сдвинуты. Логотип так же находится на своём месте. Кнопки избранного привязаны к видеопроигрывателю.

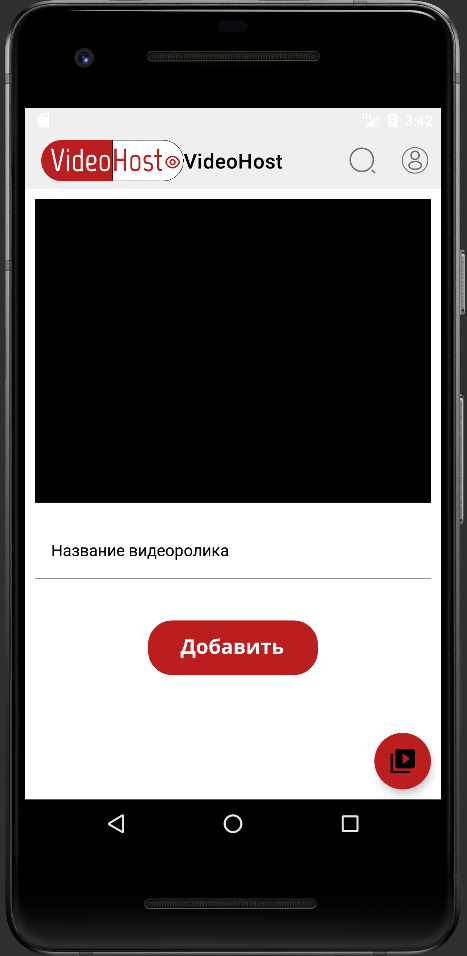
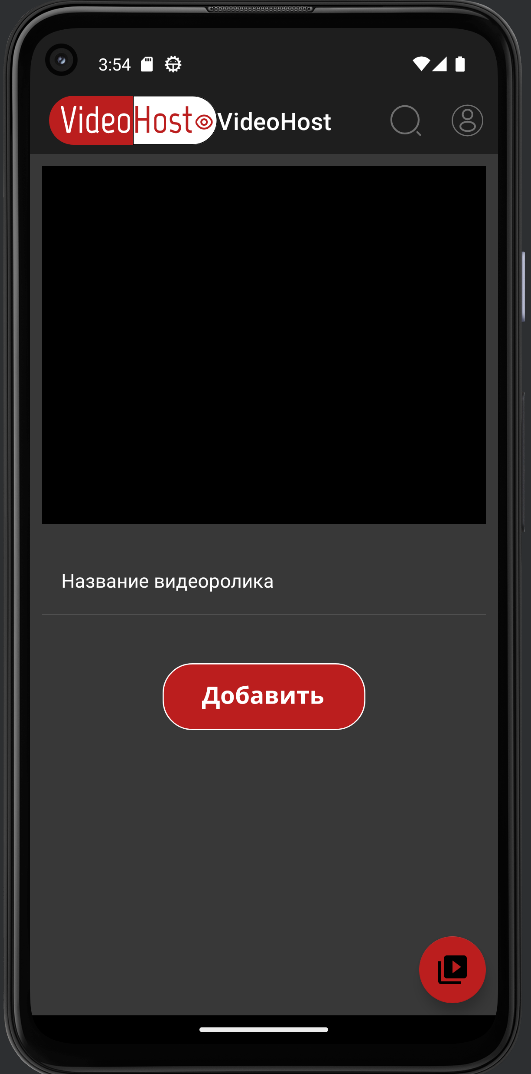
 

Рисунок 16 – Экран добавления видеоролика

На экране добавления видеоролика так же не обнаружено ошибок. Масштаб элементов сохранен. Верхняя панель находится на своём месте и имеет нормальные размеры.

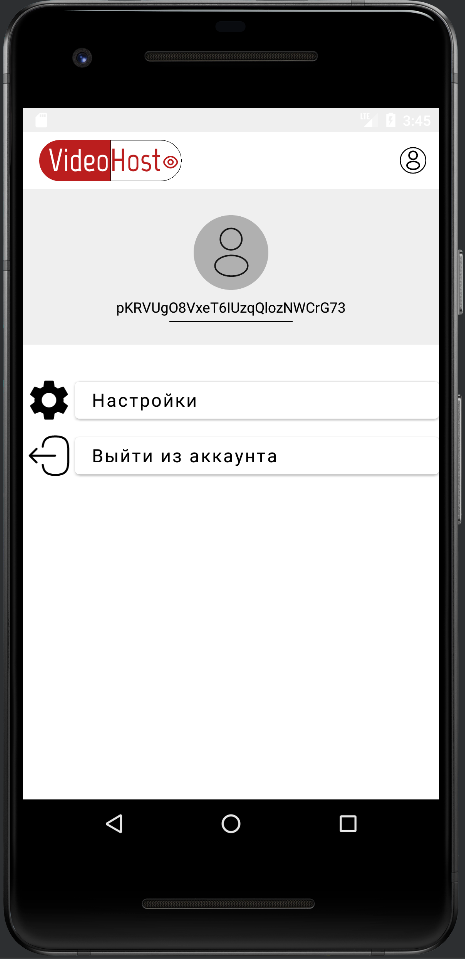
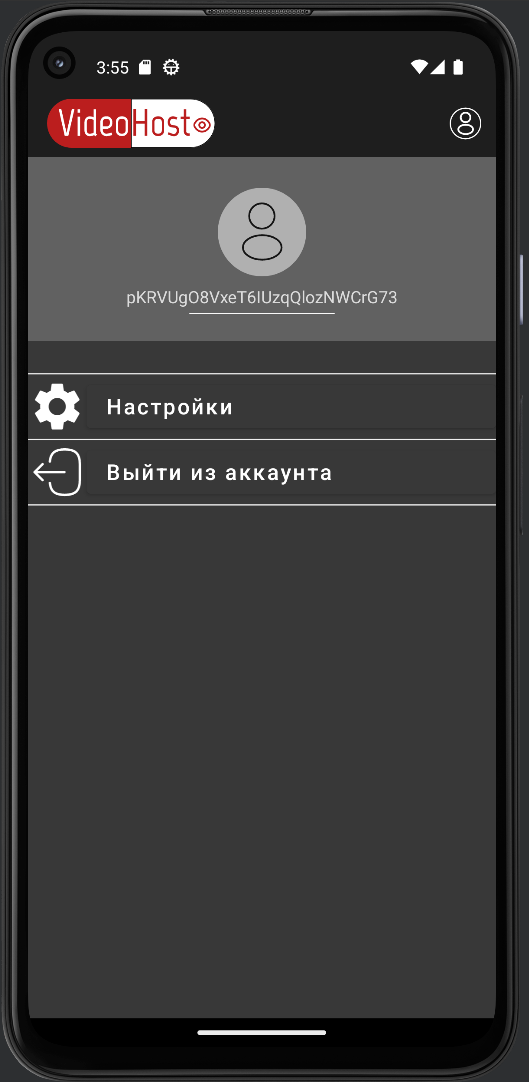
 

Рисунок 17 – Экраны профиля

Элементы экрана профиля отображаются корректно. Фотография пользователя не поменяла свои масштабы. Элементы меню профиля находятся на своих местах. Текст не растянут и не выходит за пределы экрана.

## Протокол тестирования функционала приложения

Так же необходимо проверить функционал приложения. Для этого, для каждой функции были разработаны TestCase.

В таблице 5 представлено тестирование функции вывода видеоролика на экран.

Таблица 5 – Тестирование функции вывода видео на экран

|  |  |  |
| --- | --- | --- |
| **Название**: | VideoHost | |
| **Функция**: | Вывод видео на экран | |
| **Действие** | **Ожидаемый** **результат** | **Результат** **теста**:  пройден  провал  заблокирован |
| **Предусловие**: |  | |
| Запустить приложение «VideoHost» | Приложение открылось корректно |  |
|  |  |  |
| **Шаги** **теста (positive)**: |  | |
| Дождаться загрузки главного экрана | Окно для вывода видеороликов открылось | пройден |

В таблице 6 представлено тестирование функции добавления видео в «Понравившиеся».

Таблица 6 – Тестирование функции добавление видео в «Понравившиеся».

|  |  |  |
| --- | --- | --- |
| **Название**: | VideoHost | |
| **Функция**: | Добавление видео в «Понравившиеся» | |
| **Действие** | **Ожидаемый** **результат** | **Результат** **теста**:  пройден  провал  заблокирован |
| **Предусловие**: |  | |
| Запустить приложение «VideoHost» | Приложение открылось корректно |  |
|  |  |  |
| **Шаги** **теста (positive)**: |  | |
| Дождаться загрузки главного экрана | Окно для вывода видеороликов открылось | пройден |
| Нажать на кнопку добавления видео в избранное | В базу данных отправлен запрос на добавление видео в избранное | пройден |
| Перейти на вкладку «Понравившиеся» | Добавленный ролик отобразится в списке | пройден |

В таблице 7 представлено тестирование функции загрузки видео в базу данных.

Таблица 7 – Тестирование функции загрузки видео

|  |  |  |
| --- | --- | --- |
| **Название**: | VideoHost | |
| **Функция**: | Загрузка видео | |
| **Действие** | **Ожидаемый** **результат** | **Результат** **теста**:  пройден  провал  заблокирован |
| **Предусловие**: |  | |
| Запустить приложение «VideoHost» | Приложение открылось корректно |  |
|  |  |  |
| **Шаги** **теста (positive)**: |  | |
| Перейти на вкладку добавления видеоролика | Окно корректно загрузилось | пройден |
| Выбрать видеоролик из галереи или снять новый на камеру | Приложение получило файл видеоролика и информацию о нём | пройден |
| Ввести название видеоролика | Приложение получило название видеоролика | пройден |
| Нажать на кнопку «Загрузить» | Файл видеоролика отправлен в базу данных | пройден |
|  |  |  |
|  |  |  |
| **Шаги** **теста (negative)**: |  |  |
| Перейти на вкладку добавления видеоролика | Окно корректно загрузилось | пройден |
| Не выбирать видеоролик | Приложение получит код ошибки | пройден |
| Не вводить название видео | Приложение получит код ошибки | пройден |
| Нажать на кнопку «Загрузить» | Появится всплывающее сообщение с определенной ошибкой | пройден |

В таблице 8 представлено тестирование функции регистрации нового пользователя.

Таблица 8 ­– Тестирование функции регистрации пользователя

|  |  |  |
| --- | --- | --- |
| **Название**: | VideoHost | |
| **Функция**: | Регистрация нового пользователя | |
| **Действие** | **Ожидаемый** **результат** | **Результат** **теста**:  пройден  провал  заблокирован |
| **Предусловие**: |  | |
| Запустить приложение «VideoHost» | Приложение открылось корректно |  |
|  |  |  |
| **Шаги** **теста (positive)**: |  | |
| Перейти на вкладку добавления регистрации | Окно корректно загрузилось | пройден |
| Заполнить обязательные поля корректными данными | Успешное заполнение обязательных полей | пройден |
| Нажать на кнопку «Регистрация» | Приложение проверило введенные данные и зарегистрировало пользователя | пройден |
|  |  |  |
|  |  |  |
| **Шаги** **теста (negative)**: |  |  |
| Перейти на вкладку добавления регистрации | Окно корректно загрузилось | пройден |
| Не заполнять обязательные поля или ввести некорректные данные | Успешное заполнение обязательных полей | пройден |
| Нажать на кнопку «Регистрация» | Появится всплывающее сообщение с определенной ошибкой | пройден |

В таблице 9 представлено тестирование функции авторизации пользователя.

Таблица 9 ­– Тестирование функции авторизации

|  |  |
| --- | --- |
| **Название**: | VideoHost |
| **Функция**: | Авторизация |

Продолжение таблицы 9

|  |  |  |  |
| --- | --- | --- | --- |
| **Действие** | **Ожидаемый** **результат** | | **Результат** **теста**:  пройден  провал  заблокирован |
| **Предусловие**: |  | | |
| Запустить приложение «VideoHost» | Приложение открылось корректно |  | |
|  |  |  | |
| **Шаги** **теста (positive)**: |  | | |
| Заполнить обязательные поля корректными данными | Успешное заполнение обязательных полей | пройден | |
| Нажать на кнопку «Войти» | Приложение проверило введенные данные и авторизировало пользователя | пройден | |
|  |  |  | |
|  |  |  | |
| **Шаги** **теста (negative)**: |  |  | |
| Не заполнять обязательные поля или ввести некорректные данные | Успешное заполнение обязательных полей | пройден | |
| Нажать на кнопку «Войти» | Появится всплывающее сообщение с определенной ошибкой | пройден | |

Разработанные TestCase демонстрируют корректную работу функций приложений. Так же в процессе тестирования не выявлено ошибок в дизайне приложения и его логике. Каждый экран прошёл проверку на разных API на корректное отображение элементов соответствующих экранов.

Заключение

В ходе разработки мобильного приложения для видеохостинга учтены предпочтения и потребности пользователей, в формате коротких видеороликов. Функция ведения списка просмотренного контента успешно реализована, позволяя пользователям отслеживать свои предпочтения и легко вернуться к уже просмотренному контенту.

Интерфейс приложения разработан с учетом удобства использования и минимального количества действий для достижения пользовательской цели. Основные экраны спроектированы таким образом, чтобы не перегружать пользователей лишней информацией, и добавлены полезные функции, такие как просмотр трейлеров и возможность создания собственных списков просмотренного и избранного контента.

Важным элементом концепции приложения является возможность просмотра коротких видеороликов, которая позволяет пользователям сократить время в дороге на работу, учебу или в перерывах между делами. Это удобный способ получать информацию на ходу.

Поддержка списка избранного контента также оказалась значимой для пользователей, предоставляя удобный способ отслеживания и доступа к контенту, который им особенно нравится.

Важным этапом в разработке приложения было тестирование дизайна, которое подтвердило его корректность и успешное функционирование. Это гарантирует, что разработанное приложение соответствует поставленным целям и требованиям.

В целом разработка мобильной версии видеохостинга позволила создать функциональное и удобное приложение, учитывающее потребности пользователей. Проделанная работа успешно достигла поставленной цели курсового проекта и является основой для дальнейшего развития и улучшения приложения.

Библиография

1 Yasno [Электронный ресурс]: Мобильный интерфейс: разработка дизайна приложений и проектирование макетов на примерах – Режим доступа к руководству: <https://yasno.mobi/blog/mobilnyy-interfeys-razrabotka-dizayna-prilozheniy-i-proektirovanie-maketov-na-primerakh/>

2 DevelopersAndroid [Электронный ресурс]: Полная документация по Android–Режим доступа к руководству: <https://developer.android.com/>

3 METANIT [Электронный ресурс]: Документация по языку программирования Kotlin – Режим доступа к руководству: <https://metanit.com/kotlin/>

4 Webkyrs [Электронный ресурс]: Создание базы данных Firebase – Режим доступа к руководству: <https://webkyrs.info/page/kak-sozdat-bazu-dannykh-firebase-realtime-database>

5 Epochta [Электронный ресурс]: как составить портрет клиента – Режим доступа: <https://www.epochta.ru/blog/portret-klientov>

**Приложение А**

Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж

имени Б.С. Галущака»

**РАЗРАБОТКА Мобильной версии видеохостинга**

Техническое задание

НАТКиГ.202000.010.000ПЗ

Выполнил:

Студент группы ПР-21.106

Шестаков И.В.

2023

**Содержание**

Введение 43

1 Назначения разработки 44

2 Требования к мобильному приложению 45

2.1 Требования к функциональным характеристикам 45

2.2 Требования к надёжности 45

2.3 Условия эксплуатации 45

2.4 Требования к составу и параметрам технических средств 45

2.5 Требования к информационной и программной совместимости 46

2.6 Требования к защите информации 46

2.7 Требования к маркировке и упаковке 46

3 Требования к программной документации 47

4 Технико-экономические показатели 48

5 Стадии и этапы разработки 49

6 Порядок контроля и приёмки 50

**Введение**

Настоящее техническое задание распространяется на разработку мобильного приложения «Разработка мобильной версии видеохостинга», используемого для просмотра видеороликов, ведения списков загруженных видеороликов и понравившегося.

Наименование приложения: «VideoHost».

Краткая характеристика области применения: мобильное приложение предоставляет возможность просмотра видеороликов, загруженных на видеохостинг, а также ведения списка загруженных видеороликов самого пользователя и наиболее понравившегося ему контента.

Основанием для проведения разработки является Протокол №6 от 21 февраля 2022 года.

Наименование темы разработки − «Разработка мобильной версии видеохостинга».

Условное обозначение темы разработки – «VideoHost».

**1 Назначение разработки**

Основное назначение приложения заключается в:

* обеспечении удобного и понятного интерфейса приложения для пользователя;
* предоставление большой библиотеки видеороликов.

Лица, которые могут работать с данной системой:

администратор – управляет полностью всем приложением, следит за его работоспособностью, обновляет информацию в базе данных приложения;

пользователь приложения – может просматривать видеоролики загруженные на видеохостинг, а также вести списки понравившихся и загруженных им видеороликов.

**2 Требования к мобильному приложению**

**2.1 Требования к функциональным характеристикам**

Требования к составу выполняемых функций:

* просмотр видеороликов;
* просмотр списка загруженных пользователем видеороликов;
* просмотр списка понравившихся видеороликов;

**2.2 Требования к надёжности**

Обеспечение устойчивого функционирования должно выполняться несколькими действиями:

* организация стабильного интернет-соединения.

Приложение должно контролировать входную информацию:

* соблюдение типов данных при заполнении полей.

**2.3 Условия эксплуатации**

Пользователь должен иметь практические навыки использования мобильного устройства под управлением операционной системы Android.

**2.4 Требования к составу и параметрам технических средств**

Для работы приложения необходимо мобильное устройство с установленной операционной системой Android не ниже версии 7.0.

**2.5 Требования к информационной и программной совместимости**

Проектирование взаимодействия с файловой системой должно быть выполнено в рамках разработки курсового проекта. При разработке взаимодействия с файловой системой должен быть использован язык программирования Java.

**2.6 Требования к защите информации**

Доступ к информации БД предоставляется только администратору базы данных.

**2.7** **Требования к маркировке и упаковке**

Требования к маркировке и упаковке не предъявляются.

**3 Требования к программной документации**

Состав программной документации должен включать в себя:

* техническое задание;
* пояснительная записка.

**4 Технико-экономические показатели**

Экономические преимущества разработки и ориентировочная экономическая эффективность не рассчитывается.

**5 Стадии и этапы разработки**

Таблица 1 – Стадии разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | **Этапы разработки КП** | **Сроки выполнения** | **Отчётность** |
| 1 | Определение цели и задач, объекта и предмета исследования | 13.03.2023 | Пояснительная записка |
| 2 | Описание предметной области | 08.04.2023 | Пояснительная записка |
| 3 | Выбор технологии, языка и среды программирования | 10.04.2023 | Пояснительная записка |
| 4 | Оформление технического задания | 15.04.2023 | Техническое задание |
| 5 | Проектирование UI/UX дизайна | 17.04.2023 | Спецификации программного обеспечения |
| 6 | Разработка мобильного приложения | 22.04.2023 | Схема структурная системы и спецификации компонентов |
| 7 | Разработка базы данных | 02.05.2023 | Программный  продукт |
| 8 | Отладка и тестирование приложения | 13.05.2023 | Тексты программных компонентов |
| 9 | Оформление документации | 22.05.2023 | Программная  документация |
| 10 | Защита | 27.05.2023 |  |

**6 Порядок контроля и приёмки**

Виды испытаний – защита курсового проекта.

Общее требования к приёмке:

* техническое задание;
* пояснительная записка;
* программный продукт;