Министерство образования Новосибирской области

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

**РАЗРАБОТКА Мобильной версии видеохостинга**

Пояснительная записка к курсовому проекту

ПМ.01Разработка модулей программного обеспечения для компьютерных систем

МДК.01.03 Разработка мобильных приложений

НАТКиГ.202000.010.000ПЗ

Разработал:

студент группы ПР-21.106

Шестаков И.В.

2022

**Содержание**

[Введение 3](#_Toc117112970)

[1 Исследовательский раздел 4](#_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 6](#_Toc117112975)

[2 Проектирование приложения 11](#_Toc117112976)

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

[2.2 Выбор технологии, языка и среды программирования 18](#_Toc117112978)

[3 Разработка мобильного приложения 19](#_Toc117112979)

[3.1 Разработка базы данных 19](#_Toc117112980)

[3.2 Разработка используемых плагинов 20](#_Toc117112981)

[3.3 Описание разработанных процедур и функций 21](#_Toc117112982)

[4 Тестирование 26](#_Toc117112983)

[4.1 Протокол тестирования дизайна приложения 26](#_Toc117112984)

[4.2 Протокол тестирования функционала приложения 26](#_Toc117112985)

[Заключение 29](#_Toc117112986)

[Библиография 30](#_Toc117112987)

[Приложение А 31](#_Toc117112988)

Введение

Большую часть жизни современных людей занимает медиапространство, выступающих, как пространство для поиска медиафайлов. Мобильное приложение видеохостинга облегчает просмотр и загрузку видео для пользователя. В любой момент пользователь может загрузить видео и посмотреть его. Понравившиеся видеоролики пользователь может добавить в закладки, и в будущем, найти их в соответствующей вкладке.

Вопрос, касающийся нахождения видеохостинга, подходящего для потребностей пользователя, очень актуален. Ведь каждый видеохостинг предлагает свою реализацию задачи, и не может удовлетворить каждого пользователя.

Целью курсового проекта является создание мобильного приложения видеохостинга.

Задачами курсового проекта в связи с указанной целью являются:

* изучение предметной области мобильного приложения видеохостинга;
* рассмотрение приложения с точки зрения пользователя для выявления функций приложения;
* разработка и реализация дизайна приложения;
* написание кода приложения;
* тестирование полученного продукта.

Объект исследования – мобильное приложение видеохостинга.

Предмет исследования – изучение принципов функционирования и инструментов приложения.

# Исследовательский раздел

## Описание предметной области

Видеохостинг является важной и неотъемлемой частью сферы развлечений в интернете. Он предоставляет возможность пользователям хранить, обмениваться и просматривать видеоролики. Видеохостинги играют важную роль в удобном хранении больших объемов видеоконтента, которые пользователи не всегда могут сохранить на своих устройствах. Они также обеспечивают безопасность данных и предотвращают потерю видеороликов в случае поломки устройства.

Видеохостинги стали неотъемлемой частью интернета для большинства людей, ведь они являются удобным инструментом для хранения личных видеороликов и просмотра интересующего контента. К тому же, они позволяют сохранять просмотренные и понравившиеся пользователю видеоролики в отдельные списки.

Приложение видеохостинга подойдёт и для тех, кто любит смотреть видео – зрителям, так и тем, кто любит выкладывать их – блогерам.

Зрители являются активными пользователями видеохостинга, которые просматривают различные видеоролики. Они могут использовать поиск, фильтры и категории для нахождения интересующего контента. Так же они могут оценивать видео и делится ими с друзьями. Оцененные видео добавляются в отдельный плейлист, в котором их можно будет с лёгкостью найти позже.

Блогеры же, это пользователи видеохостинга, которые загружают собственные видеоролики на платформу. Они могут настраивать приватность своего канала или отдельного видеоролика. Они так же могут изменять добавленные видеоролики, или вовсе удалять их с канала.

При входе в приложение пользователь сначала должен авторизоваться или зарегистрироваться в случае, если у него нет аккаунта. После авторизации пользователь сразу может просматривать доступные видео, загруженные на сервер. Так же, если он перейдет на страницу добавления видеоролика, он сможет загрузить своё видео из галереи, или же снять новое на камеру телефона. Загрузив видео, он должен заполнить обязательное поле: Название видеоролика. После чего он сможет загрузить видео на сервер.

Загруженные видео сразу появляются на главной странице. Их может посмотреть каждый пользователь, если только автор не ограничил доступ. Оно никогда не исчезнет с главной страницы только в случае, если автор сам не удалит его.

Каждый пользователь может добавить видеоролик в понравившееся. После чего он отобразится на отдельной вкладке. Так же, все видеоролики, загруженные пользователем, хранятся на вкладке “Загруженное”. На этой вкладке он может изменить: название видеоролика, удалить его, изменить настройки приватности и т.д.

Так же на вкладке профиля, можно перейти к настройкам своего аккаунта. Там пользователь может поменять свою: почту, пароль, аватарку и ник.

Таким образом, можно с уверенностью сказать, что видеохостинги очень облегчают жизнь пользователей, давая неограниченное место под загрузку своих видеороликов, и позволяя хранить их, с наименьшим риском потери.

Приложение «VideoHost» – программный продукт, который всегда будет под рукой, так как телефон у современного человека всегда с собой. Приложение является библиотекой видеороликов пользователей с необходимостью авторизации.

## 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 приходится сидеть на нескольких сайтах, чтобы следить за всем, что интересует его. А для того, чтобы сохранять их, приходится скачивать их на своё устройство. Но куда удобнее иметь мобильное приложение, которое позволит собрать 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, которые

## 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):

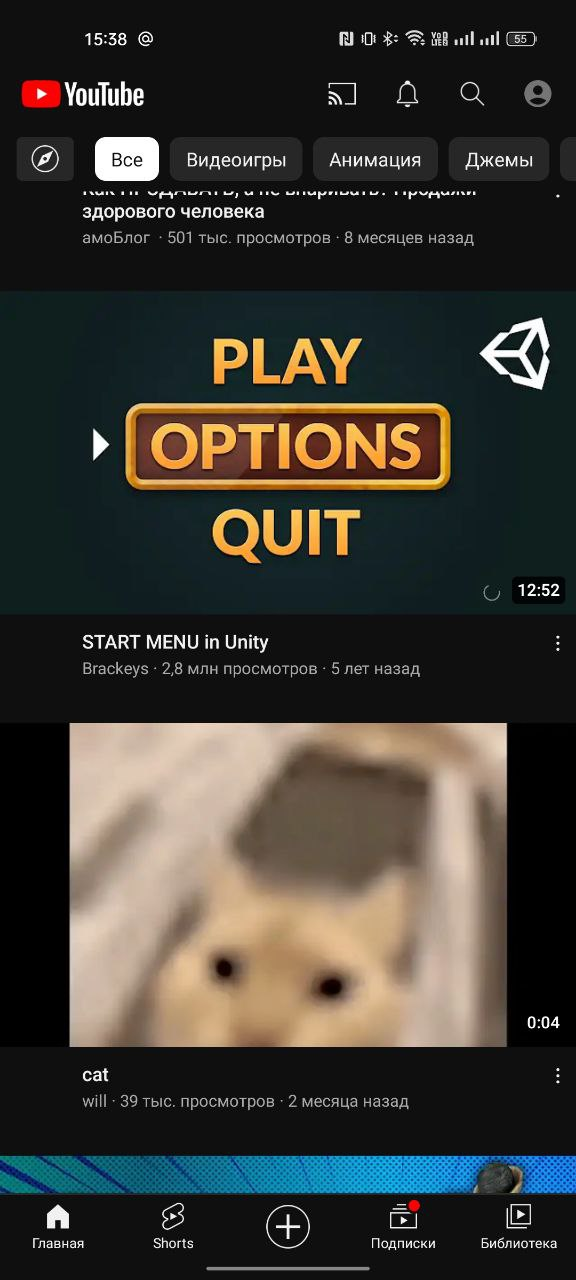
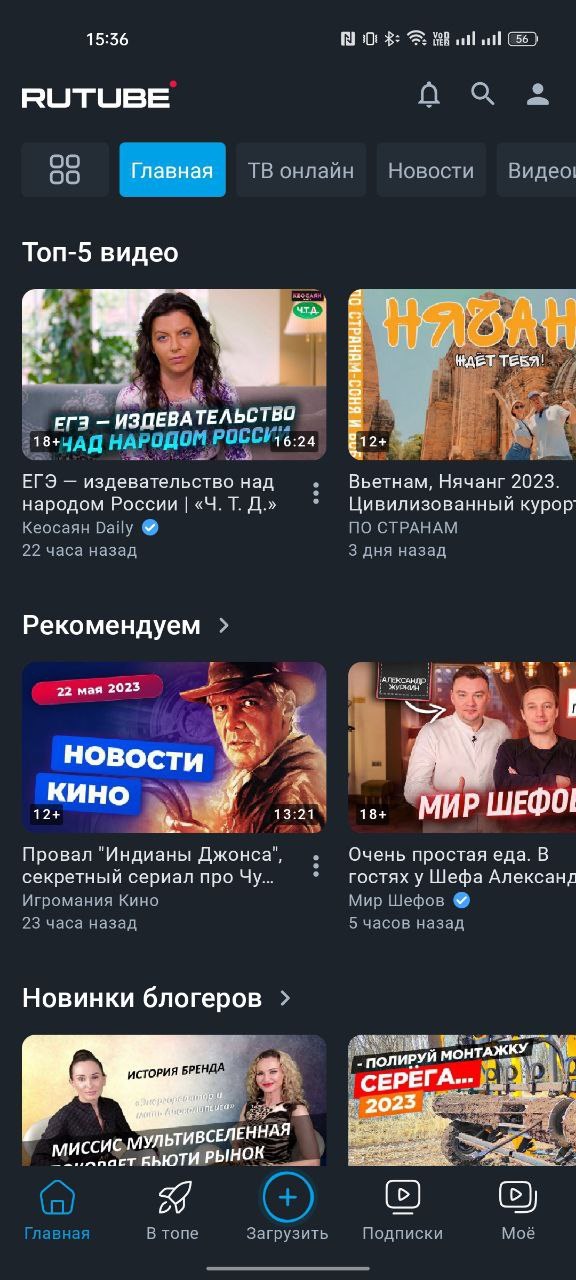


Рисунок 1 – Главный экран приложений

Так же в них реализована функция добавления видеороликов в понравившееся. Они выводятся на отдельном экране (Рисунок 2):

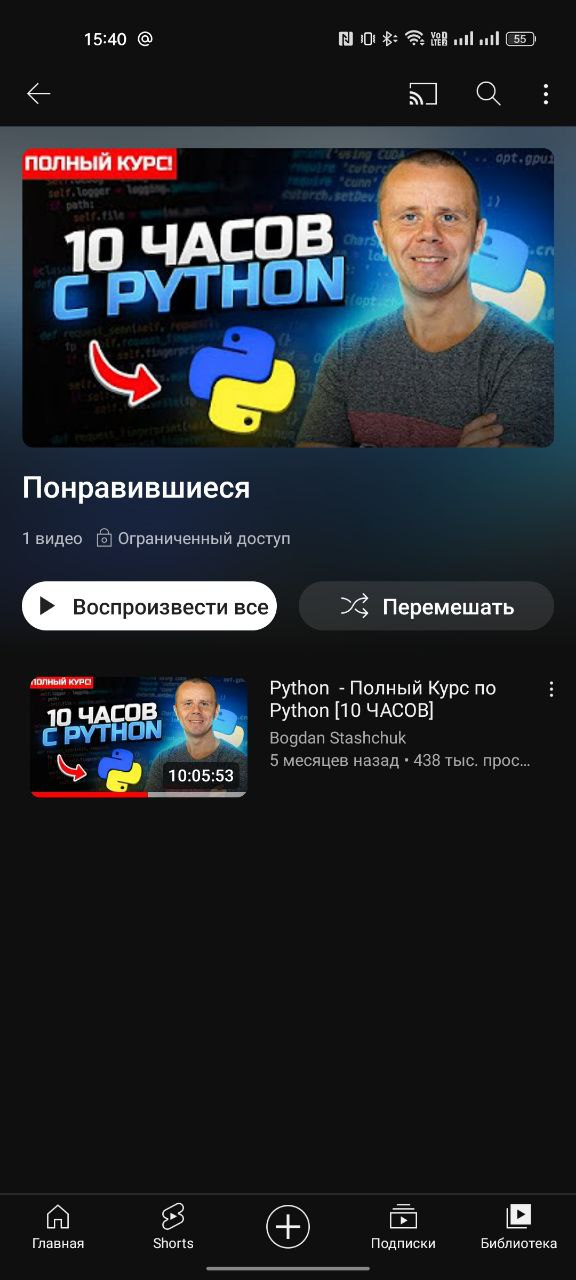


Рисунок 2 – Экран с понравившимися видеороликами

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

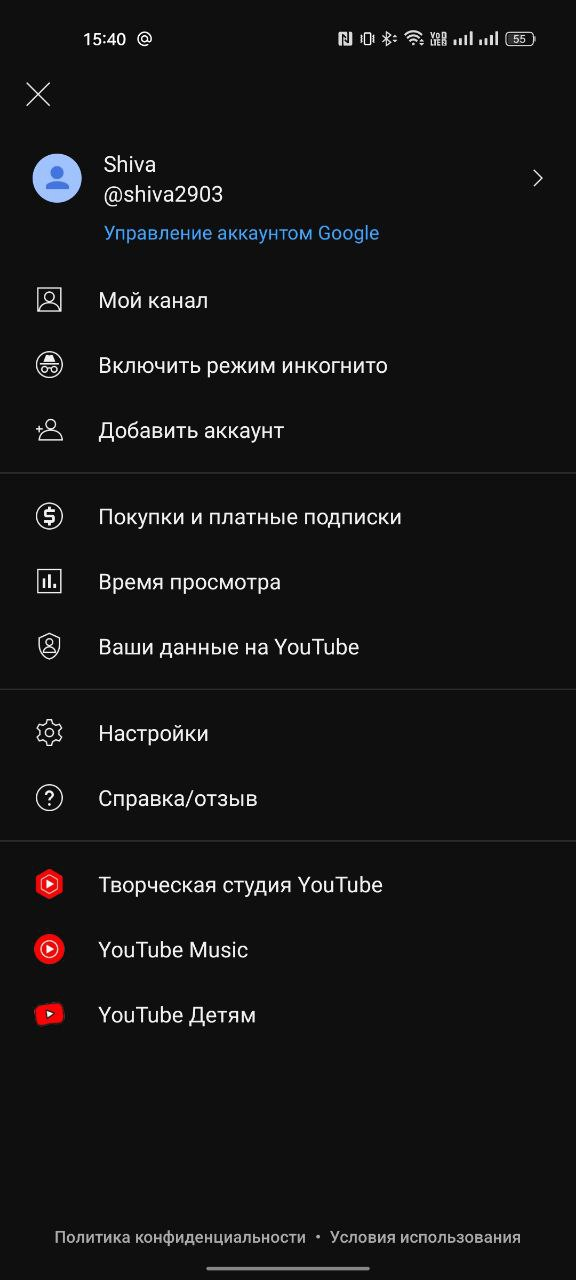
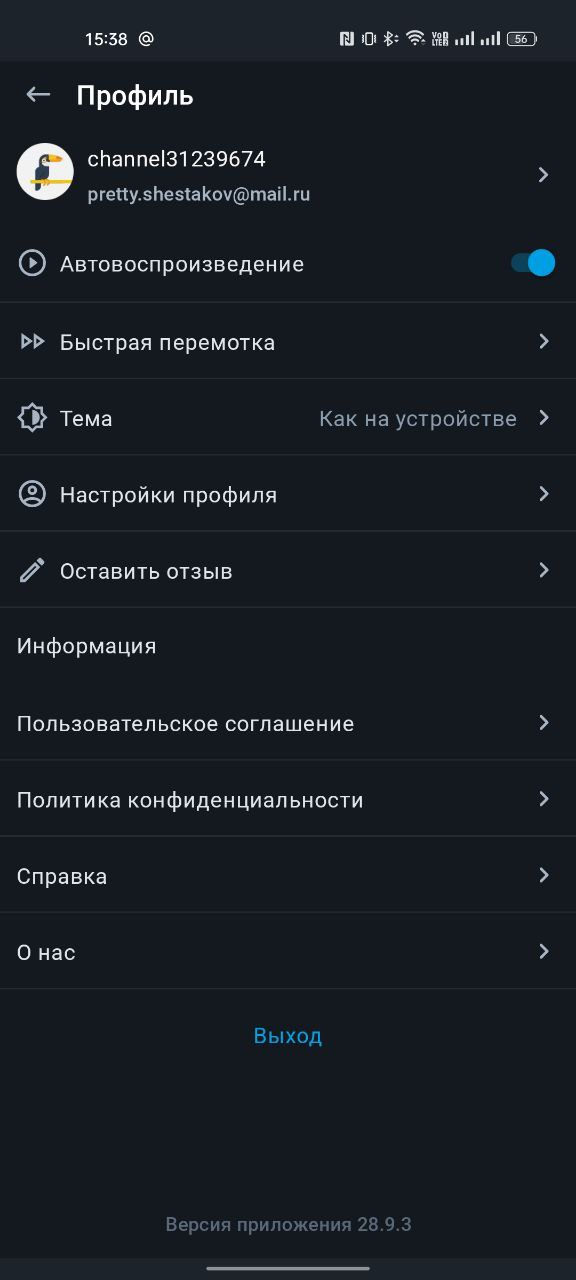


Рисунок 3 – Экран профиля пользователя

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».

# Проектирование приложения

* 1. **UI/UX дизайн приложения**

Дизайн проекта разработан в программе Figma.

Для проекта были определены основные экраны:

* экран регистрации;
* экран авторизации;
* главный экран с видеороликами;
* экран личного кабинета;
* экран с понравившимися видеороликами;
* экран загрузки нового видеоролика;
* экран загруженных видеороликов;
* экран просмотра видеоролика.

Основными темами приложения определены две цветовые схемы. Первая тема будет реализована в тёмных тонах, поэтому в ней основными цветами будут: серый, черный, белый (Рисунок 4).

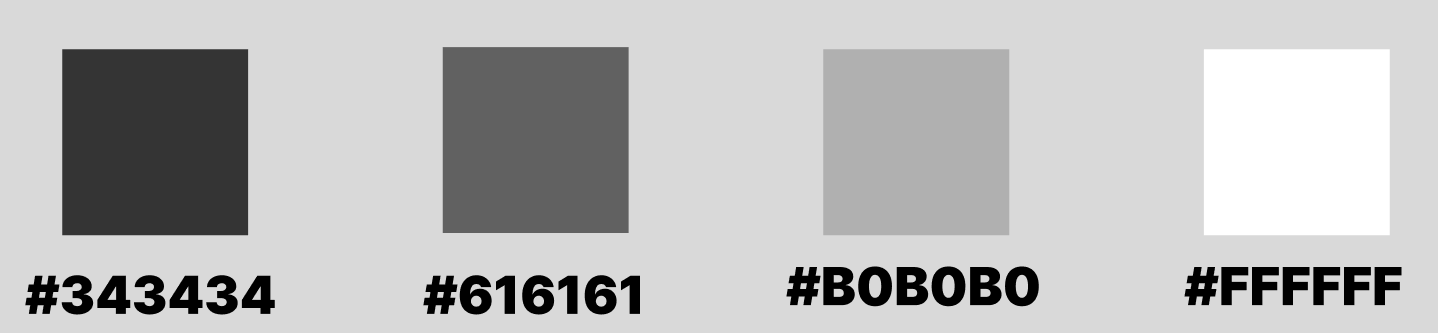


Рисунок 4 – Тёмная цветовая схема

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 (Рисунок 6) 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.

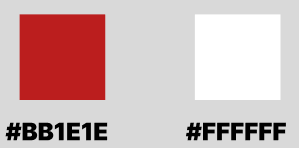


Рисунок 7 – Цветовая схема логотипа

Определившись с цветовой схемой приложения и создав его логотип, был разработан полный дизайн, состоящий из 8 экранов:

* Экран авторизации (Login);
* Экран регистрации (Registration)
* Главный экран (Main)
* Профиль (Profile)
* Экран понравившихся видеороликов (Like)
* Экран загрузки видеороликов (UploadVideo)
* Экран загруженных видеороликов (DoneUpload)
* Экран проигрывания видеоролика (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.

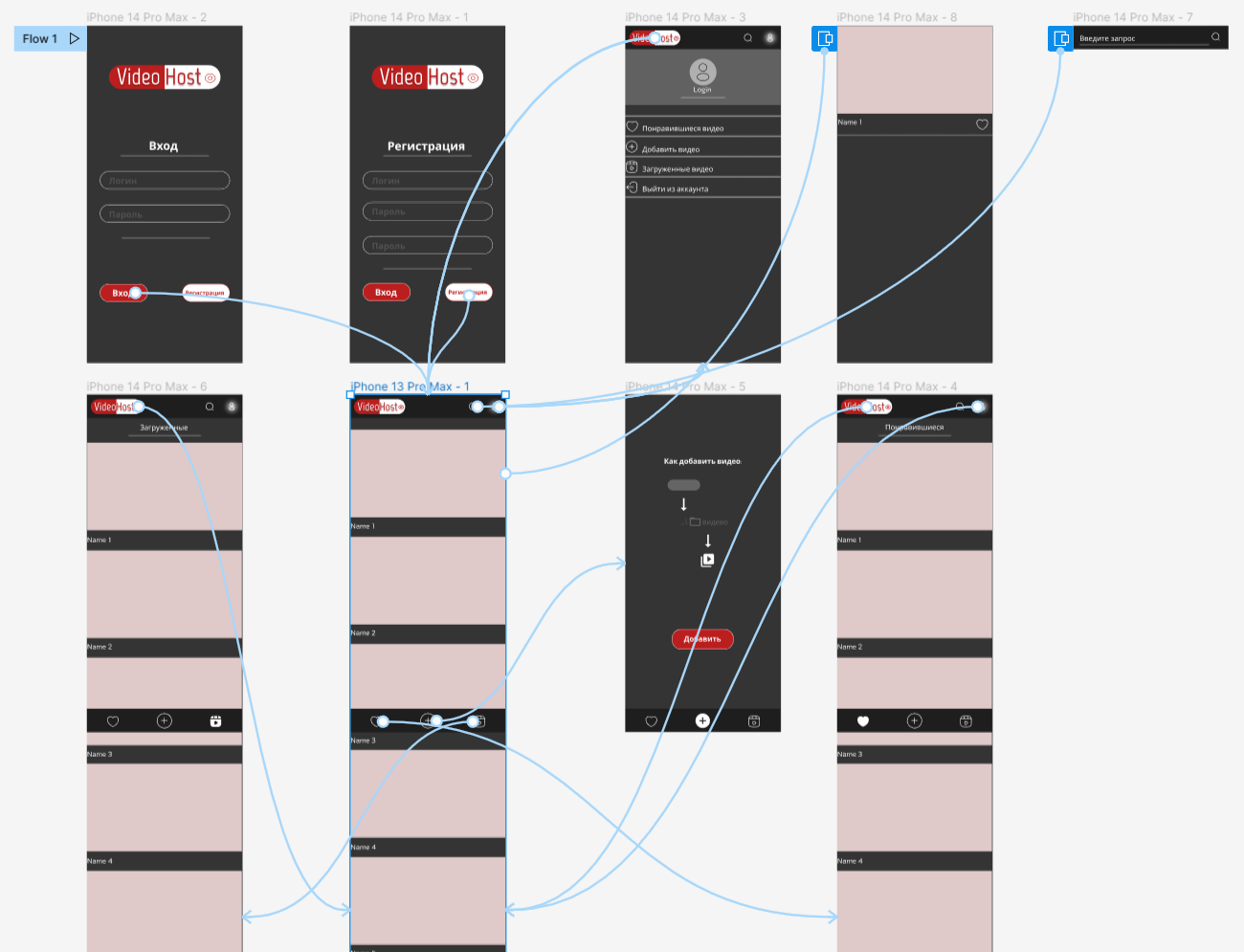


Рисунок 10 – Перемещение пользователя в приложении

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.

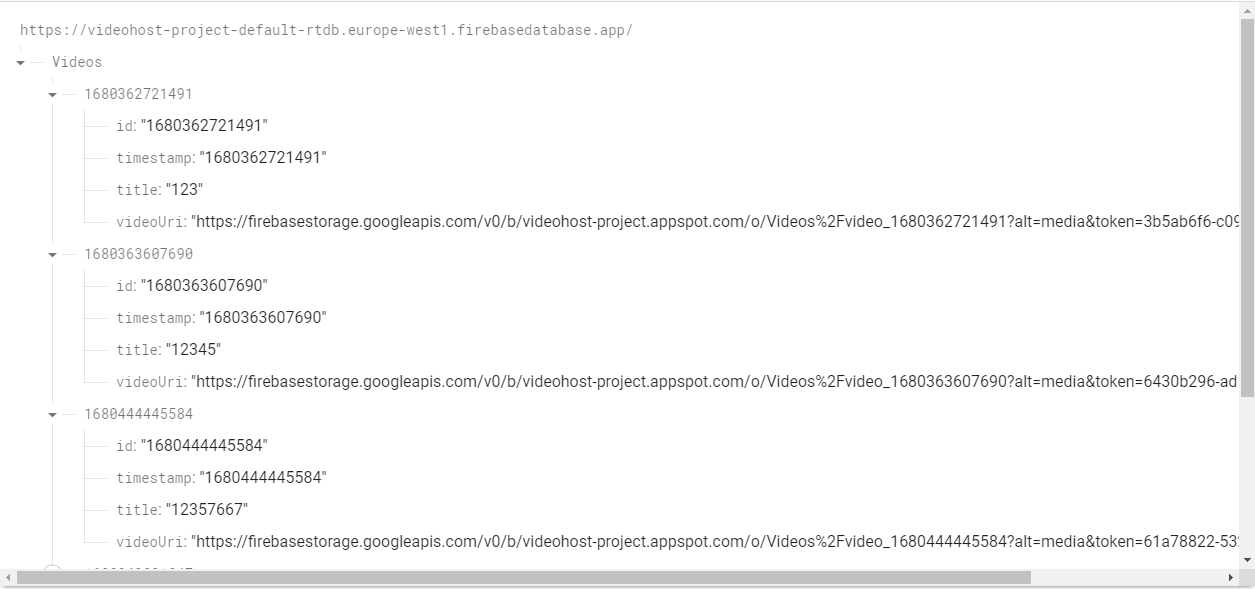


Рисунок 11 – Страница Firebase с информацией о видео

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»

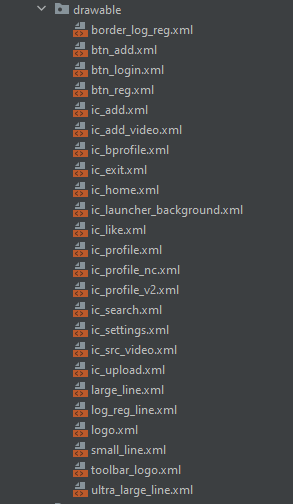


Рисунок 12 – Мультимедийный контент

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

Листинг 1 – Палитра цветов

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

<color name="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 |
| 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 (splash screen) в приложении. |
| androidx.constraintlayout:constraintlayout:2.1.4 | Библиотека для работы с ConstraintLayout, который предоставляет гибкие возможности по размещению и организации элементов пользовательского интерфейса. |
| 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 Material Design 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. |
| com.google.firebase:firebase-database-ktx:20.2.1 | Библиотека для работы с Firebase Realtime Database, 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 Firebase Storage, 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. |

## Описание разработанных процедур и функций

В приложении разработаны следующие функции:

* регистрация и авторизация;
* переход между страницами с помощью навигационного меню;
* добавление видеоролика на канал;
* добавление видеоролика в избранное;
* просмотр видеороликов;

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

Листинг 3 – Авторизация в приложение

btnLog.setOnClickListener **{** when{  
 TextUtils.isEmpty(txtMail.*text*.toString().*trim* **{ it** <= ' ' **}**) -> {  
 Toast.makeText(  
 this@LoginActivity,  
 "Пожалуйcта, заполните поле - e-mail.",  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 TextUtils.isEmpty(txtPass.*text*.toString().*trim* **{ it** <= ' ' **}**) -> {  
 Toast.makeText(  
 this@LoginActivity,  
 "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** <= ' ' **}** // 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()  
  
 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) {  
 val regIntent = Intent(this, RegisterActivity::class.*java*)  
 startActivity(regIntent)  
 finish()  
}

При регистрации, в базу данных запросами заносятся новые данные пользователя (Листинг 4).

Листинг 4 – Регистрация пользователя

btnReg.setOnClickListener **{** when{  
 TextUtils.isEmpty(txtMail.*text*.toString().*trim* **{ it** <= ' ' **}**) -> {  
 Toast.makeText(  
 this@RegisterActivity,  
 "Пожалуйcта, заполните поле - e-mail.",  
 Toast.*LENGTH\_SHORT* ).show()  
 }  
 TextUtils.isEmpty(txtPass.*text*.toString().*trim* **{ it** <= ' ' **}**) -> {  
 Toast.makeText(  
 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(  
 this@RegisterActivity,  
 "Регистрация прошла успешно.",  
 Toast.*LENGTH\_SHORT* ).show()  
  
 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)  
 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 5). Они выводятся с помощью компонента для отображения элементов списка RecyclerView.

Листинг 5 – Вывод видеороликов

private fun loadVideosFromFirebase() {  
 videoArrayList = ArrayList()  
  
 val ref = FirebaseDatabase.getInstance("https://videohost-project-default-rtdb.europe-west1.firebasedatabase.app").getReference("Videos")  
 ref.addValueEventListener(object : ValueEventListener{  
  
 override fun onDataChange(snapshot: DataSnapshot) {  
 videoArrayList.clear()  
 for (ds in snapshot.*children*){  
  
 val modelVideo = ds.getValue(ModelVideo::class.*java*)  
  
 videoArrayList.add(modelVideo!!)  
 }  
 videoArrayList.*sortByDescending* **{ it**.timestamp **}** adapterVideo = AdapterVideo(requireContext(), videoArrayList, firebaseAuth)  
 val videosRv: RecyclerView = *view*!!.findViewById(R.id.*videosRv*)  
 videosRv.*adapter* = adapterVideo  
 }  
  
 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 6).

Листинг 6 – Добавление видео в избранное

private fun toggleFavorite(userId: String, videoId: String?, isFavorite: Boolean) {  
 val databaseRef =  
 FirebaseDatabase.getInstance("https://videohost-project-default-rtdb.europe-west1.firebasedatabase.app").getReference("Users/$userId/Favorites/$videoId")  
 databaseRef.child("isFavorite").setValue(isFavorite)  
 .addOnCompleteListener **{** task **->** if (task.*isSuccessful*) {  
 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 7).

Листинг 7 – Вывод избранных видеороликов

private fun loadFavoriteVideosFromFirebase() {  
 val favoriteVideoIds = ArrayList<String>()  
 videoArrayList = ArrayList()  
  
 val userId = firebaseAuth.*currentUser*?.*uid* userId?.*let* **{** uid **->** 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()  
 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()  
 for (ds in snapshot.*children*) {  
 val modelVideo = ds.getValue(ModelVideo::class.*java*)  
 modelVideo?.*let* **{** video **->** if (favoriteVideoIds.contains(ds.*key*.*toString*())) {  
 videoArrayList.add(video)  
 }  
 **}** }  
 videoArrayList.*sortByDescending* **{ it**.timestamp **}** adapterVideo.videoArrayList = videoArrayList  
 adapterVideo.notifyDataSetChanged()  
 }

При добавлении видеоролика пользователь должен дать разрешение на использование камеры или галереи (Листинг 8).

Листинг 8 – Запрос прав

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  
}

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) {  
 //Приняли  
 videoPickCamera()  
 } else{  
 Toast.makeText(this, "Права подтверждены", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
 }  
  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults)  
}  
  
private val resultLauncher = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) **{** result **->** if (result.*resultCode* == Activity.*RESULT\_OK*) {  
 val data: Intent? = result.*data* videoUri = data!!.*data* }  
**}**

После получения прав доступа пользователь может загрузить видеоролик, заполнив обязательные поля (Листинг 9).

Листинг 9 – Загрузка видеоролика

private fun uploadVideoFirebase() {  
 progressDialog.show()  
  
 val timestamp = ""+System.currentTimeMillis()  
 //Добавим время файлу  
 val filePathAndName = "Videos/video\_$timestamp"  
  
 val storageReference = FirebaseStorage.getInstance().getReference(filePathAndName)  
  
 storageReference.putFile(videoUri!!)  
 .addOnSuccessListener **{** taskSnapshot **->** val uriTask = taskSnapshot.*storage*.*downloadUrl* while (!uriTask.*isSuccessful*);  
 val downloadUri = uriTask.*result* if (uriTask.*isSuccessful*){  
 //Детали для Firebase  
 val hashMap = HashMap<String, Any>()  
 hashMap["id"] = "$timestamp"  
 hashMap["title"] = "$title"  
 hashMap["timestamp"] = "$timestamp"  
 hashMap["videoUri"] = "$downloadUri"  
  
 val dbReference = FirebaseDatabase.getInstance("https://videohost-project-default-rtdb.europe-west1.firebasedatabase.app").getReference("Videos")  
 dbReference.child(timestamp)  
 .setValue(hashMap)  
 .addOnSuccessListener **{** taskSnapshot **->** progressDialog.dismiss()  
 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

Для проведения тестирования дизайна приложение было выбрано устройство Honor 10ic разрешение экрана 2340×1080.

Проверка была проведена на:

* оптимальный размер кнопок;
* читабельный размер шрифта;
* приятное глазу цветовое сочетание;
* корректное расположение элементов;

## Протокол тестирования функционала приложения

Для проверки правильности ввода пароля при регистрации были проведены UnitTests, которые изображены ниже на листинге (Листинг 3):

public class PasswordUtils {  
  
 private static final String *VALID\_PASSWORD\_REGEX* = "((?=.\*\\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[@#$%=:\\?]).{8,12})";  
  
 @Test  
 public void validatePassword\_EmptyString() {  
*// setup*String password = "";  
  
*// execute*boolean actual = PasswordUtils.*validatePassword*(password);  
  
*// assert  
assertFalse*(actual);  
 }  
  
 @Test  
 public void validatePassword\_Null() {  
*// setup*String password = null;  
  
*// execute*boolean actual = PasswordUtils.*validatePassword*(password);  
  
*// assert  
assertFalse*(actual);  
 }  
  
 @Test  
 public void validatePassword\_Missing\_OneNumber() {  
*// setup*String password = "Abcdefg#";  
  
*// execute*boolean actual = PasswordUtils.*validatePassword*(password);  
  
*// assert  
assertFalse*(actual);  
 }  
  
 @Test  
 public void validatePassword\_LengthTooLong() {  
*// setup*String password = "Abcdefg5#abcdefgabcd";  
  
*// execute*boolean actual = PasswordUtils.*validatePassword*(password);  
  
*// assert  
assertFalse*(actual);  
 }  
  
 @Test  
 public void validatePassword\_Missing\_OneUpperCaseLetter() {  
*// setup*String password = "abcdefg5#";  
  
*// execute*boolean actual = PasswordUtils.*validatePassword*(password);  
  
*// assert  
assertFalse*(actual);  
 }  
  
  
  
 public static booleanvalidatePassword(String password) {  
 return (password != null &&password.matches(*VALID\_PASSWORD\_REGEX*));  
 }  
}

Листинг 3–UnitTests

Так же для тестирования функционала были разработаны TestCase.

Test-Case №1

Название тест кейса: Регистрация

Предусловия тест кейса: приложение открыто02

Шаги тест кейса:

* 1. шаг тест кейс №1:открытьстраницу регистрации;
  2. шаг тест кейса №2: заполнить все поля;
  3. шаг тест кейса №3: нажать на кнопку «регистрация».

Ожидаемый результат тест кейса:

Пользователь зарегистрирован в системе.

Статус кейса: выполнено.

Test-Case №2

Название тест кейса: Авторизация

Предусловия тест кейса: пользователь зарегистрирован

Шаги тест кейса:

* 1. шаг тест кейс №1: открыть страницу авторизации;
  2. шаг тест кейса №2: заполнить все поля;
  3. шаг тест кейса №3: нажать на кнопку «Вход».

Ожидаемый результат тест кейса:

Пользователь войдёт в систему.

Статус кейса: выполнено.

Test-Case №3

Название тест кейса: Переход на другую активность

Предусловия тест кейса: пользователь вошёл в систему

Шаги тест кейса:

1. шаг тест кейс №1: открыть страницу с навигацией;
2. шаг тест кейса №2: нажать на желаемую кнопку.

Ожидаемый результат тест кейса:

Пользователь перешёл на необходимую ему страницу.

Статус кейса: выполнено.

Заключение

В результатеработыбыло разработано клиентское мобильное приложение для автосервиса.

Во время разработки возникли трудности с подключением сервиса Firebase к проекту, проблема была решена изменением Android на AndroidX. AndroidX– это переработанная библиотека, чтобы сделать имена пакетов более понятными.

В дальнейшем планируется совершенствовать приложение и добавить уже к имеющемуся функционал.

Библиография

1. AndroidDevelopers [электронный ресурс]: ДокументацияAndroidStusio – https://developer.android.com/docs/
2. HelpCenter [электронныйресурс]: ДокументацияJava: https://docs.oracle.com/en/java/
3. Автостат[электронный ресурс]:Структура российских автовладельцев по возрасту и маркамhttps://www.autostat.ru/infographics/40710/
4. Firebase [электронныйресурс]: Документация Firebase - https://firebase.google.com/docs?hl=uk
5. Material Design [электронныйресурс]: BottomNavigation: https://material.io/components/bottom-navigation/android
6. Git[электронныйресурс]: ДокументацияGit: https://git-scm.com/
7. Figma[электронныйресурс]: Прототипирование проектовhttps://www.figma.com/

Приложение А

Техническое задание

Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж

имени Б.С. Галущака»

**разработка мобильного приложения для автосервиса**

НАТКиГ.202100.010.000ПЗ

Выполнил:

студент группы

ПР-20.106

Петрищева А.В.

2022

**Содержание**

Введение 34

1Назначения разработки 35

2Требованияк мобильному приложению 36

2.1 Требования к функциональным характеристикам 36

2.2 Требования к надёжности 36

2.3 Условия эксплуатации 37

2.4 Требования к составу и параметрам технических средств 2337

2.5 Требования к информационной и программной совместимости 2337

2.6 Требования к защите информации 38

2.7 Требования к маркировке и упаковке 2338

3Требования к программной документации 2338

4Технико-экономические показатели 2338

5Стадии и этапы разработки 23638

6Порядок контроля и приёмки 2339

**Введение**

Настоящее техническое задание распространяется на разработку информационной системы «Разработкамобильногоприложениядляавтосервиса», используемого для записи на приём, просмотра услуг, чата с оператором.

Наименование приложения: «Garage54».

Краткая характеристика области применения: приложение для автосервиса – полезное оснащение телефона автовладельца. При необходимости пользователь может просмотреть имеющиеся в автосервисе услуги, записаться на приём и обратиться за помощью к оператору.

Условные обозначения и сокращения:

БД – База данных;

ИС – Информационная система;

ОС – Операционная система;

СТО – Станция технического обслуживания.

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

Наименование темы разработки – «Разработка мобильного приложения для автосервиса».

Условное обозначение темы разработки – «Автосервис – Garage54».

**1Назначение разработки**

Основное назначение мобильного приложения заключается в:

* осведомлении клиентов об услугах;
* возможности клиента записи на приём.

Лица, которые могут работать с данной Системой:

пользователь–просматривает услуги, регистрируется, авторизуется, оформляет запись;

администратор–добавляет и редактирует услуги, просматривает записи.

**2Требования к мобильному приложению**

2.1 Требования к функциональным характеристикам

Требования к составу выполняемых функций:

* регистрация и авторизация пользователей;
* просмотр имеющихся услуг;
* запись на приём;
* чат с оператором.

Входные данные должны быть организованы в виде вводимого, в специальную форму, текста, соответствующего определённому шаблону. Данные, вводимые вручную, проверяются на корректность.

**2.2 Требования к надёжности**

Обеспечение устойчивого функционирования должно выполняться несколькими действиями:

* организация бесперебойного питания оборудования пользователя;
* использование лицензионного программного обеспечения;
* организация стабильного интернет-соединения.

Приложение должно контролировать входную информацию:

* соблюдение типов данных при заполнении полей;
* операции изменения, удаления и сохранения.

Время восстановления после отказа, вызванного сбоем электропитания технических средств, не фатальным сбоем ОС или файловой системы, не должно превышать 15 минут при соблюдении условий эксплуатации технических и программных средств и правильной настройки операционной системы.

**2.3 Условия эксплуатации**

Обслуживание ИС включает в себя:

1. информационное обслуживание – ввод и редактирование информации БД;
2. системное администрирование БД ИС.

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

Для работы приложения на устройстве требуется: объем свободной памяти не менее 3 ГБ, версия устройства Android 10 и выше.

**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.

**3Требования к программной документации**

Состав программной документации должен включать в себя:

* техническое задание;
* проектную документацию.

**4Технико-экономические показатели**

Экономические преимущества разработки и ориентировочная экономическая эффективность не рассчитывается.

**5Стадии и этапы разработки**

Таблица 1 – Стадии разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название этапа | Срок,даты | Отчётность |
| 1 | Исследование предметной области | 25.03.2022-28.03.2022 | Пояснительная записка |
| 2 | Выбор моделей, описывающих предметную область | 01.04.2022-03.04.2022 | Пояснительная записка |
| 3 | Разработка технического задания | 03.04.2022-05.04.2022 | Техническое задание |
| 4 | Изучение Java и Firebase | 01.04.2022-  18.04.2022 | Пояснительная записка |
| 5 | Анализ требований и уточнение спецификаций | 14.04.2022 | Спецификации программного обеспечения |
| 6 | Проектирование структуры мобильного приложения, проектирование компонентов (технический проект) | 15.04.2022-29.04.2022 | Схема структурная системы и спецификации компонентов |
| 8 | Кодирование клиентской части | 01.05.2022-20.06.2022 | Программный  Продукт |
| 9 | Тестирование компонентов.  Сборка и комплексное тестирование | 20.06.2022 | Тексты программных компонентов |
| 10 | Разработка программной  документации | 24.06.2022 – 26.06.2022 | Программная  документация |
| 12 | Защита | 27.06.2022 |  |

**6Порядок контроля и приёмки**

Виды испытаний – защита курсового проекта.

Общее требования к приёмке:

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