Министерство образования Новосибирской области

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

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПЛАНИРОВКИ ИНТЕРЬЕРА КВАРТИРЫ

Пояснительная записка к курсовому проекту

ПМ.01 Разработка модулей программного обеспечения  
для компьютерных систем

МДК.01.03 Разработка мобильных приложений

НАТКиГ.200300.010.000ПЗ

Разработал:

студент группы ПР-22.106

Бочаров Д.В.

2024

**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 10](#_Toc117112976)

[2.1 UI/UX дизайн проекта 10](#_Toc117112977)

[2.2 Выбор технологии, языка и среды программирования 12](#_Toc117112978)

[3 Разработка мобильного приложения 13](#_Toc117112979)

[3.1 Разработка базы данных 13](#_Toc117112980)

[3.2 Разработка мультимедийного контента 14](#_Toc117112981)

[3.3 Описание используемых плагинов 15](#_Toc117112982)

[3.4 Описание разработанных процедур и функций 18](#_Toc117112982)

[4 Тестирование 23](#_Toc117112983)

[4.1 Протокол тестирования дизайна приложения 23](#_Toc117112984)

[4.2 Протокол тестирования функционала приложения 24](#_Toc117112985)

[Заключение 26](#_Toc117112986)

[Библиография 27](#_Toc117112987)

[Приложение А Техническое задание 28](#_Toc117112988)

Приложение Б 35

**Введение**

В современном мире мобильные приложения стали неотъемлемой частью нашей повседневной жизни. Одной из самых актуальных областей в разработке приложений является область интерьерного дизайна. В этом контексте разработка мобильного приложения для планировки интерьера квартиры представляет собой важную задачу, которая объединяет в себе инновационные технологии, функциональность и удобство использования. В данном курсовом проекте рассматривается процесс разработки такого приложения с использованием современных методов и технологий, а также оценивается потенциал данной темы для рынка мобильных приложений и потребностей пользователей.

Целью курсового проекта является создание мобильного приложения для планировки интерьера квартиры.

Задачами курсового проекта в связи с указанной целью являются:

– изучение предметной области;

– рассмотрение приложения с точки зрения пользователя для выявления функций приложения;

– написание кода приложения;

– тестирование полученного продукта.

Объект исследования – мобильное приложение для планировки интерьера квартиры.

Предмет исследования – изучение принципов функционирования и инструментов приложения.

**Исследовательский раздел**

**1.1 Описание предметной области**

Все хотят создать уютное и стильное пространство в своей квартире, но не всегда у них достаточно опыта или времени для планировки интерьера. Правильно подобранное мобильное приложение для планировки интерьера квартиры может стать отличным помощником в этом деле. Если разработать грамотное приложение с удобным интерфейсом и широким выбором функций, планирование интерьера станет легким и увлекательным процессом.

Планировка интерьера - это искусство создания гармоничного и функционального пространства, которое отражает индивидуальность его владельца. Мобильное приложение для планировки интерьера позволяет визуализировать идеи, подобрать цветовые решения, мебель и декор, а также распределить пространство оптимальным образом.

Планирование интерьера квартиры может вестись по-разному. Иногда для этого требуется доступ к коллекциям мебели и декора, чтобы увидеть, как они будут смотреться в конкретном интерьере. Мобильные приложения для планировки интерьера обычно предлагают широкий выбор объектов интерьера, которые можно добавлять и редактировать по своему усмотрению. При этом, важно выбрать приложение, которое соответствует вашим потребностям и ожиданиям, чтобы сделать процесс планировки интерьера максимально удобным и эффективным.

**1.2 Образ клиента**

Клиентами являются владельцы квартир, а также люди, интересующиеся планировкой и улучшением своего жилища. Приложение рассчитано на широкий возрастной диапазон, но основной аудиторией будут взрослые, стремящиеся к уютному оформлению квартиры. Приложение будет удобным как для самостоятельного проектирования и планирования, так и для тех, кто ищет вдохновение и новые идеи для улучшения своего жилища.

**1.3 Сценарии**

Молодая пара планирует ремонт в новой квартире и хочет создать уютное пространство, отражающее их стиль. Они могли бы использовать мобильное приложение, которое позволяет загружать фотографии комнат и экспериментировать с различными стилями декора, мебели и цветовых схем, чтобы поделиться своими идеями друг с другом и получить обратную связь в реальном времени.

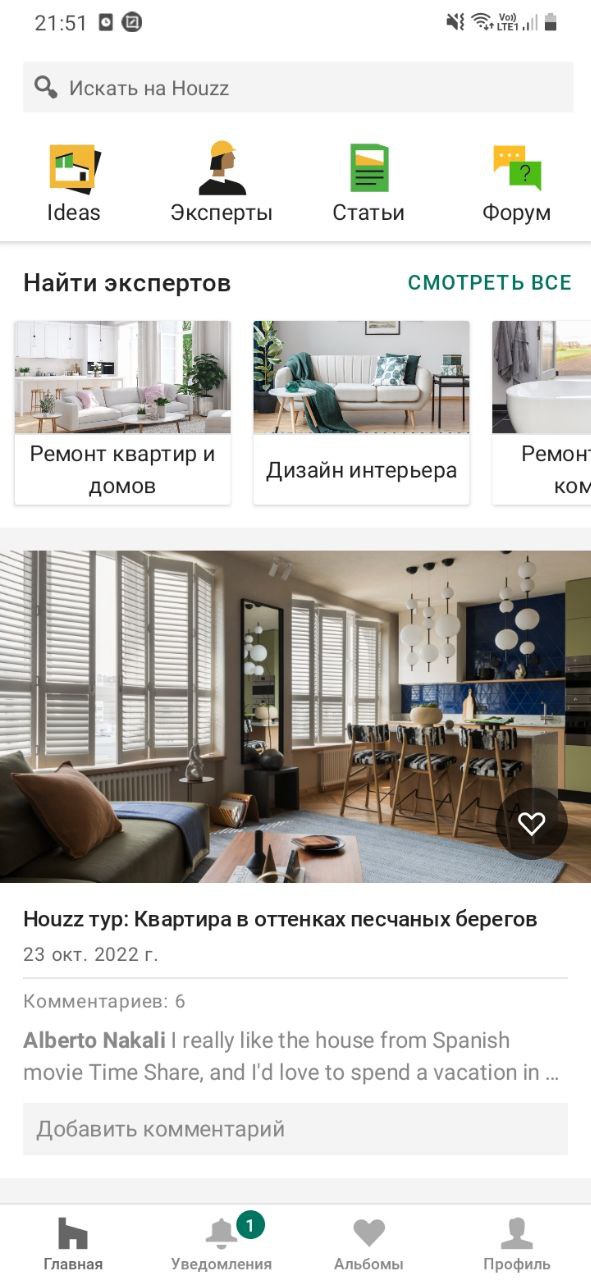
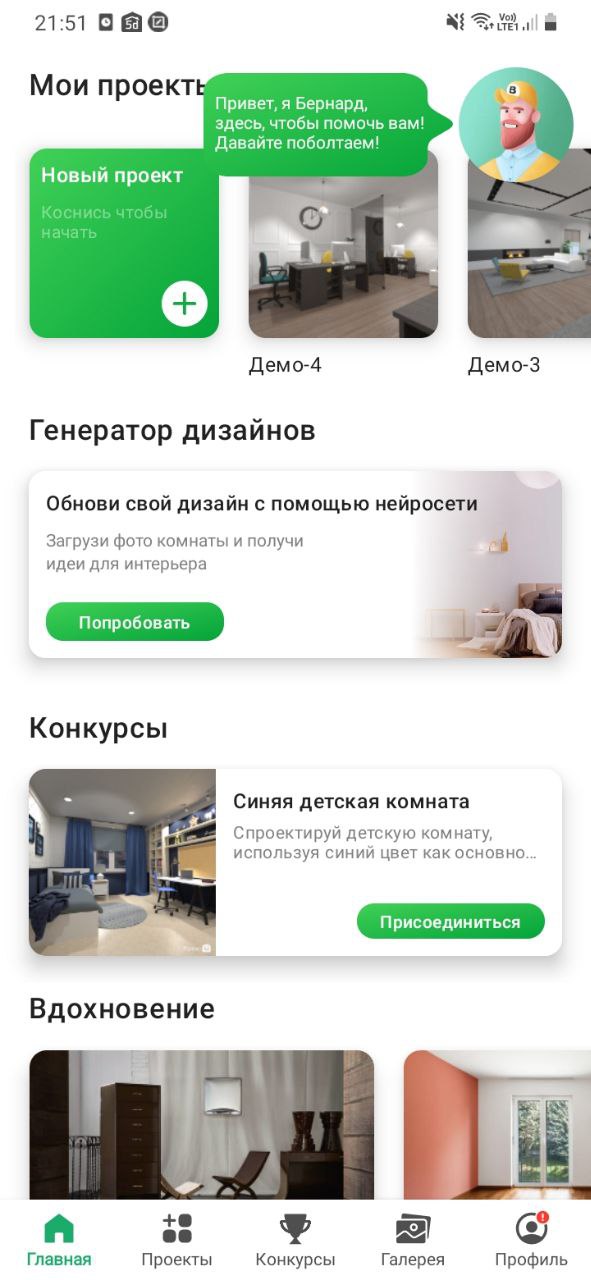
Студент архитектор работает над проектированием квартиры и стремится найти инновационные решения для оптимального использования пространства. Мобильное приложение, предлагающее расширенные возможности планирования и визуализации, поможет ему представить идеи и лучше понять, как различные элементы дизайна будут сочетаться и функционировать в реальной квартире.

Пожилой человек, стремящийся освежить интерьер с минимальными затратами и усилиями, может воспользоваться приложением для интерьера, которое позволяет просматривать различные варианты оформления, автоматически подходящие к размерам его комнат. Это дает возможность удобно выбирать идеальные стили и предметы декора, не покидая своей квартиры.

**1.4 Сбор и анализ прототипов**

В GooglePlay и AppStore существует приложения для планировки интерьера квартиры . Первое – Houzz (Американское приложение), второе – Planner5D (Русское приложение). Оба этих приложения предоставляют возможности для создания и редактирования дизайна квартиры, а также для просмотра готовых проектов и сохранения понравившихся идей. Давайте рассмотрим эти два приложения подробнее и сравним их.

Оба этих приложения имеют главный экран, на котором собраны рекомендации и советы 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:

* поиск;
* проекты.

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

* главная;
* профиль;
* галерея.

Вывод фотографий отличается. В приложении «Houzz» - фото выводятся по группам, а в «Planner5D» – как общий список рекомендованных фото (рисунок 2).

Так же в них реализована функция добавления фотографий в «Понравившиеся» .

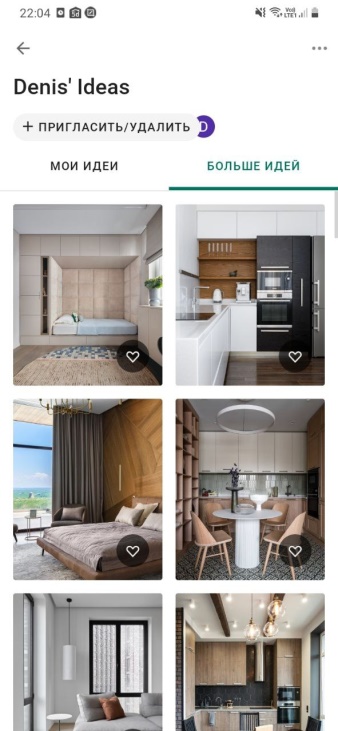
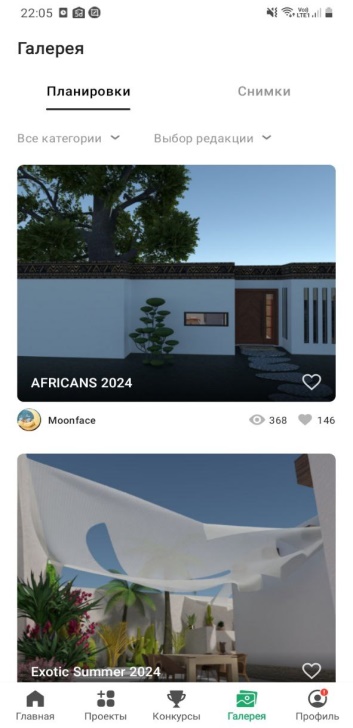
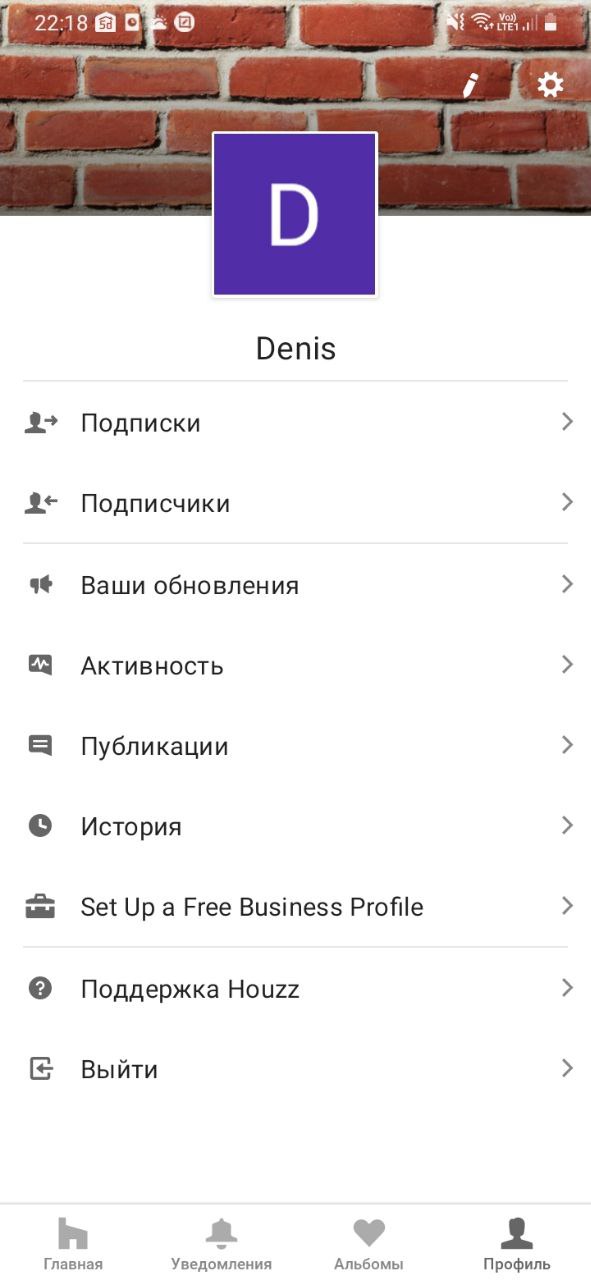
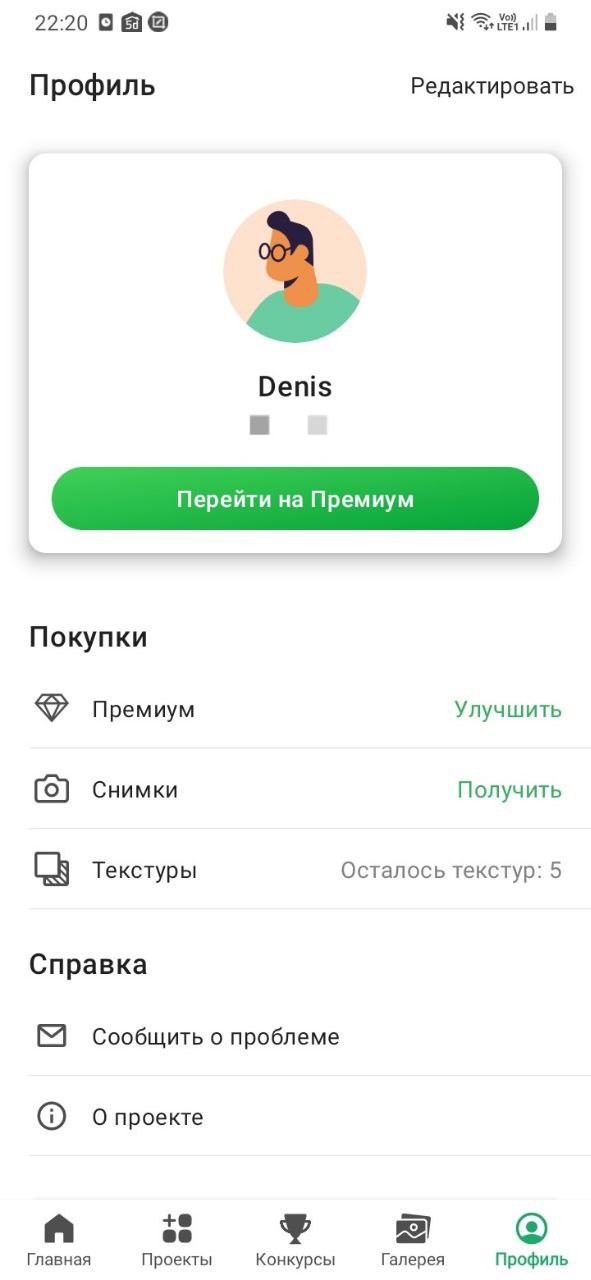


Рисунок 2 – Экран с фотографиями

Все добавленные в «Понравившиеся» фотографии, отображаются на отдельной вкладке. Их так же можно оттуда удалить или посмотреть повторно.

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

  
Рисунок 3 – Экран профиля пользователя

У обоих приложений схожие страницы профиля.

В обоих приложениях элементы профиля сгруппированы, что позволяет легче ориентироваться. Настройки, которые в «Houzz» вынесены в отдельную кнопку, в «Planner5D» находятся прямо в самом внизу профиля. Именно поэтому, экран профиля «Houzz», является более удобным для понимания и использования пользователем, чем «Planner5D».

Сравнение по основным критериям данных приложений представлено в таблице 1.

Таблица 1 – Сравнение приложений

|  |  |  |
| --- | --- | --- |
| Параметры | Houzz | Planner5D |
| Стоимость | Бесплатно | Бесплатно |
| Основной язык | Английский | Русский |
| Просмотр фотографий | Есть | Есть |
| Возможность сохранять фотографии в «Избранное» | Есть | Есть |
| Подробное описание к контенту | Есть | Есть |
| Наличие поиска | Есть | Есть |
| Выбор темы интерфейса | Есть | Есть |
| Список просмотренного | Есть | Есть |
| Оценки фотографий | Есть | Есть |
| Возможность загружать свои фотографии | Есть | Есть |
| Возможность продавать свой контент | Отсутствует | 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 просматривать фотографии 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

**2.1 UI/UX Дизайн приложения**

Для проекта был определен основной экран для публикации фотографий.

Для темы приложения определены следующие цветовые схемы.

Тема реализуется в тёмном и белом фоне (в зависимости от времени суток), а кнопки выполнены в ярком цвете (Рисунок 4).

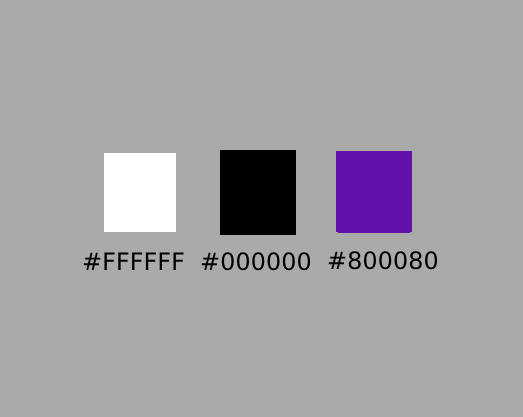


Рисунок 4 – Цветовая схема приложения

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.

Определившись с цветовой схемой приложения, был разработан дизайн основного экрана для публикации фотографий.

Ниже на рисунке 5 представлен дизайн приложения со светлой цветовой схемой.

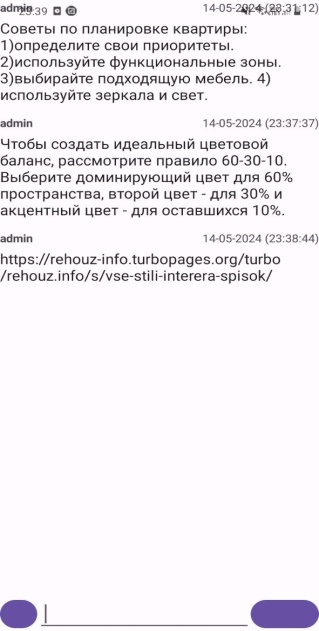


Рисунок 5 – Дизайн приложения со светлой темой

Ниже на рисунке 6 представлен дизайн приложения со светлой цветовой схемой.

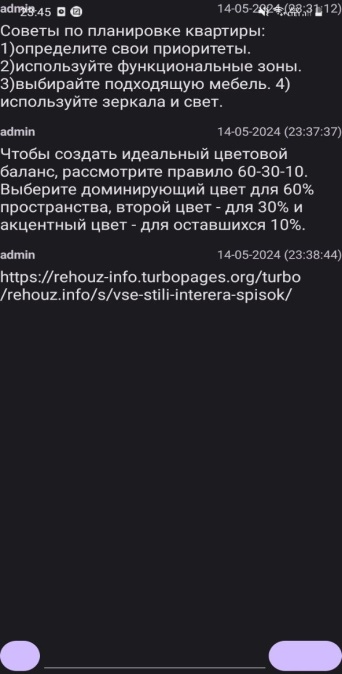


Рисунок 6 – Дизайн приложения с тёмной темой

При первом запуске приложения происходит авторизация через Google аккаунт. Это упрощает процесс авторизации и позволяет мгновенно переходить к основным функциям. После входа, пользователь напрямую попадает в ленту публикаций, где он может увидеть последние обновления и контент от других пользователей. В ленте доступны следующие действия: просмотр публикаций и возможность их собственной загрузки. Все эти функции организованы таким образом, чтобы обеспечить удобный и интуитивно понятный пользовательский интерфейс.

**2.2 Выбор технологии, языка и среды программирования**

Для разработки приложения была выбрана среда программирования Android Studio, которая пользуется заслуженной популярностью среди разработчиков. Её удобство и широкое распространение облегчают поиск решений при возникновении технических проблем. В качестве языка программирования используется Java — проверенный временем язык, который отличается стабильностью и обширной поддержкой. Java постоянно обновляется и включает в себя множество библиотек и инструментов, что делает его идеальным выбором для создания надёжных и эффективных приложений.

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

**3.1 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 7.

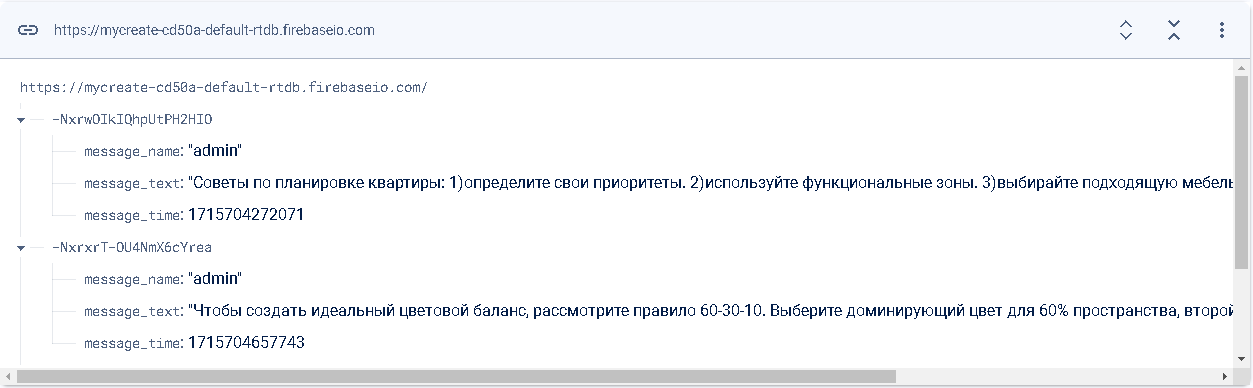


Рисунок 7 – Страница Firebase с информацией о публикациях

База данных структурирована таким образом, что каждое сообщение идентифицируется уникальным ключом, который представляет собой вершину в иерархической структуре. Для каждого ключа сообщения хранится соответствующий набор данных, включая имя отправителя, текст сообщения и время его отправки.

В этой структуре message\_id служит уникальным идентификатором для каждого сообщения, а message\_details содержит всю информацию, связанную с сообщением, включая имя отправителя (name), текст сообщения (text) и временную метку (timestamp), указывающую на время отправки сообщения.

**3.2 Разработка мультимедийного контента**

Весь мультимедийный контент разрабатывался с помощью языка разметки XML.

Экраны приложения и их интерфейсные компоненты были разработаны и интегрированы в проект с использованием файлов разметки XML, которые хранятся в папке «layout» (Рисунок 8). Этот метод организации позволяет не только упростить процесс разработки интерфейса, но и обеспечивает лёгкость внесения изменений и поддержки. XML-файлы обеспечивают чёткое разделение структуры интерфейса от бизнес-логики приложения, что способствует лучшей масштабируемости и поддержке кода. Кроме того, использование XML способствует оптимизации производительности приложения, так как XML-файлы компилируются в бинарный формат, что уменьшает время загрузки интерфейса при запуске приложения.

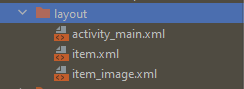


Рисунок 8 – Мультимедийный контент

В приложении предусмотрена автоматическая адаптация интерфейса к текущему времени суток, благодаря чему пользователю не требуется самостоятельно выбирать тему. Файл colors.xml содержит две цветовые палитры, которые обеспечивают гармоничное сочетание с естественным освещением от утреннего света до ночной темноты. Это позволяет приложению оставаться комфортным для глаз пользователя в любое время, без необходимости переключения между светлой и тёмной темами вручную.

**3.3 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.firebase.ui.database.FirebaseListAdapter | Упрощает отображение данных из базы данных Firebase Realtime Database в ListView на Android. |
| com.firebase.ui.database.FirebaseListOptions | Позволяет разработчикам определить, какие данные из Firebase Realtime Database будут отображаться в ListView или RecyclerView, включая параметры запроса, размер пула и поведение адаптера при изменении данных. |
| com.google.android.gms.auth.api.signin.GoogleSignIn | Предоставляет статические методы для аутентификации, позволяя пользователям входить с их аккаунтами Google. |
| com.google.android.gms.auth.api.signin.GoogleSignInAccount; | Содержит информацию о вошедшем пользователе, такую как имя, адрес электронной почты и токен аутентификации. |
| com.google.android.gms.auth.api.signin.GoogleSignInClient | Предоставляет API для управления входом в Google и выходом из аккаунта в приложении. |

Продолжение таблицы 2

|  |  |
| --- | --- |
| com.google.android.gms.auth.api.signin.GoogleSignInOptions | Определяет параметры запроса для входа в Google. |
| com.google.android.gms.tasks.OnCompleteListener | Используется для получения уведомлений о завершении задачи. |
| com.google.android.gms.tasks.Task | Предоставляет методы для проверки статуса задачи, добавления слушателей для её завершения и получения результата выполнения. |
| com.google.firebase.auth.AuthCredential | Представляет учетные данные аутентификации, используемые для аутентификации пользователя в Firebase. |
| com.google.android.gms.common.api.ApiException | Исключение, которое выбрасывается, когда вызов к Google API завершается неудачей. |
| com.google.firebase.auth.AuthResult | Представляет результат операции аутентификации |

Окончание таблицы 2

|  |  |
| --- | --- |
| com.google.firebase.auth.FirebaseAuth | Управляет всеми аспектами аутентификации пользователя. |
| com.google.firebase.auth.GoogleAuthProvider | Позволяет аутентифицировать пользователей через Google. |
| com.google.firebase.database.DataSnapshot | Cодержит данные из Firebase Realtime Database. |
| com.google.firebase.database.DatabaseError | Представляет ошибку, которая может возникнуть во время операций с базой данных. |
| com.google.firebase.database.DatabaseReference | Ссылка на конкретное место в Firebase Realtime Database. |
| com.google.firebase.database.FirebaseDatabase | Предоставляет доступ к корню Firebase Realtime Database и позволяет управлять данными. |
| com.google.firebase.database.Query | Позволяет сортировать и фильтровать данные. |
| com.google.firebase.database.ValueEventListener | Реагирует на изменения данных в месте, на которое указывает DatabaseReference. |
| java.net.HttpCookie | Может быть использован для управления информацией о сессии между клиентом и сервером. |
| java.util.ArrayList | Представляет массив, размер которого может динамически увеличиваться. |
| java.util.List | Представляет упорядоченную коллекцию. |

Эти библиотеки критически важны для приложения, так как они являются основой для его эффективной работы и воплощения заложенных функций. Отсутствие этих компонентов сделает невозможным полноценное функционирование приложения, поскольку они обеспечивают ключевые инструменты и возможности, требуемые для его операций.

**3.4 Описание разработанных процедур и функций**

В приложении реализованы следующие методы (таблица 3):

Таблица 3 – Методы приложения

|  |  |
| --- | --- |
| Метод | Описание |
| onActivityResult | Обрабатывает результаты запуска активности, в данном случае входа пользователя через Google. |
| onCreate | Инициализирует аутентификацию Firebase, настройки входа Google и интерфейс пользователя. |
| onClick | Реализует функционал отправки пользовательских сообщений в базу данных. |
| populateView | Отвечает за отображение текста, имени пользователя и времени сообщения в интерфейсе пользователя. |
| firebaseAuthWithGoogle | Метод, который использует токен ID для аутентификации пользователя через Google в Firebase. |
| getImage | Метод для создания интента, который позволяет пользователю выбрать изображение из галереи устройства. |
| onComplete | Является обработчиком завершения задачи аутентификации. |
| Message(String N, String T) | Устанавливает текущее время в message\_time. |
| getMessage\_text() | Возвращает текст сообщения. |
| getMessage\_name() | Возвращает имя отправителя сообщения. |
| setMessage\_text(String message\_text) | Устанавливает текст сообщения. |
| getMessage\_time() | Возвращает время отправки сообщения. |
| setMessage\_time(long message\_time) | Устанавливает время отправки сообщения. |
| setMessage\_name(String message\_name) | Устанавливает имя отправителя сообщения. |

В приложении есть классы (таблица 4):

Таблица 4 – Классы приложения

|  |  |
| --- | --- |
| Класс | Описание |
| **AppCompatActivity** | Обеспечивает совместимость с Android-интерфейсом. |
| **GoogleSignInClient** | Клиент для управления входом в Google и получения данных учетной записи пользователя. |
| **Button** | Виджет пользовательского интерфейса, который представляет собой кнопку на экране. |
| **FirebaseAuth** | Класс для управления аутентификацией Firebase, включая вход и регистрацию пользователей. |
| **ActivityResultLauncher** | Класс для запуска активности с ожиданием результата и последующей обработкой этого результата. |
| **Intent** | Класс для выполнения различных действий, таких как запуск активности или службы. |
| **Task** | Класс, представляющий асинхронную операцию и ее результат. |
| **GoogleSignInAccount** | Класс, содержащий информацию об учетной записи пользователя Google. |
| **ApiException** | Исключение, которое может быть выброшено при попытке выполнить задачу, связанную с API Google. |
| **GoogleSignInOptions** | Класс для настройки параметров входа в Google. |
| **FirebaseDatabase** | Класс для доступа к базе данных Firebase и выполнения операций с данными. |
| **Query** | Класс для создания запросов к базе данных Firebase. |
| **FirebaseListOptions** | Класс для настройки параметров списка Firebase. |
| **ListView** | Виджет пользовательского интерфейса для отображения списка элементов. |
| **FirebaseListAdapter** | Адаптер для связывания данных из Firebase с ListView. |
| **View** | Базовый класс для виджетов пользовательского интерфейса. |
| **TextView** | Виджет для отображения текста. |
| **DateFormat** | Класс для форматирования и анализа дат в удобочитаемый формат. |

Окончание таблицы 4

|  |  |
| --- | --- |
| **AuthCredential** | Класс для хранения учетных данных, используемых для аутентификации. |
| **GoogleAuthProvider** | Поставщик учетных данных для входа через Google. |
| **OnCompleteListener** | Интерфейс слушателя, который вызывается при завершении Task. |
| **AuthResult** | Класс, представляющий результат операции аутентификации. |
| **Toast** | Класс для создания всплывающих уведомлений. |

Авторизация в приложении осуществляется через Google-аккаунт пользователя. При запуске приложения, пользователь сразу перенаправляется на экран входа в систему Google. После успешного входа в Google-аккаунт, приложение использует полученный токен для аутентификации пользователя в Firebase. В случае успешной аутентификации, пользователь переходит на главный экран приложения, где может отправлять сообщения, которые сохраняются в базе данных Firebase.

На главном экране также отображается список всех сообщений, загруженных в базу данных. Список сообщений реализован с помощью компонента ListView, который использует адаптер FirebaseListAdapter для отображения данных. Каждое сообщение в списке содержит текст сообщения, имя пользователя и временную метку, когда сообщение было отправлено. Форматирование временной метки выполняется с помощью класса DateFormat.

Листинг 1 – Авторизация и вывод сообщений

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// Включение полноэкранного режима без учета системных окон

EdgeToEdge.enable(this);

// Установка пользовательского интерфейса из XML файла

setContentView(R.layout.activity\_main);

// Получение экземпляра FirebaseAuth

auth = FirebaseAuth.getInstance();

// Настройка параметров входа в Google

GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT\_SIGN\_IN)

.requestIdToken(String.valueOf(com.firebase.ui.auth.R.string.default\_web\_client\_id))

.requestEmail()

.build();

// Создание клиента GoogleSignIn

mGoogleSignInClient = GoogleSignIn.getClient(this, gso);

// Запуск интента входа в Google

resultLauncher.launch(new Intent(mGoogleSignInClient.getSignInIntent()));

// Настройка кнопки отправки сообщений

btnCl = findViewById(R.id.button\_set);

btnCl.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// Получение текстового поля для ввода сообщений

EditText textField = findViewById(R.id.messageField);

String sTextField = textField.getText().toString();

// Проверка на пустое сообщение и отправка в Firebase

if (!sTextField.trim().isEmpty()) {

FirebaseDatabase.getInstance()

.getReference()

.push()

.setValue(new Message("admin", sTextField));

}

// Очистка поля ввода после отправки

textField.setText("");

}

});

// Настройка Firebase Database для получения сообщений

FirebaseDatabase database = FirebaseDatabase.getInstance();

Query messages = database.getReference();

// Настройка адаптера для отображения сообщений

FirebaseListOptions<Message> options = new FirebaseListOptions.Builder<Message>()

.setQuery(messages, Message.class)

.setLayout(R.layout.item)

.build();

// Настройка списка сообщений и адаптера

ListView messageList = findViewById(R.id.list\_view);

FirebaseListAdapter<Message> adapter = new FirebaseListAdapter<Message>(options) {

@Override

protected void populateView(View v, Message model, int position) {

// Получение и установка текста, пользователя и времени сообщения

TextView messageText = v.findViewById(R.id.message\_Text);

TextView messageUser = v.findViewById(R.id.message\_User);

TextView messageTime = v.findViewById(R.id.message\_Time);

messageText.setText(model.getMessage\_text());

messageUser.setText(model.getMessage\_name());

messageTime.setText(DateFormat.format("dd-MM-yyyy (HH:mm:ss)",

model.getMessage\_time()));

}

};

// Установка адаптера и начало прослушивания базы данных

messageList.setAdapter(adapter);

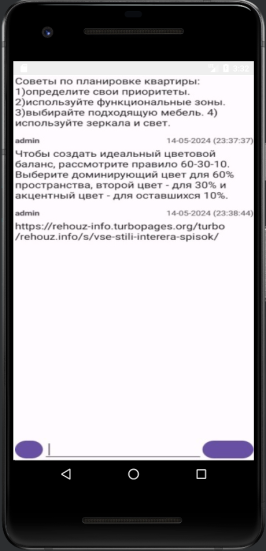
adapter.startListening();

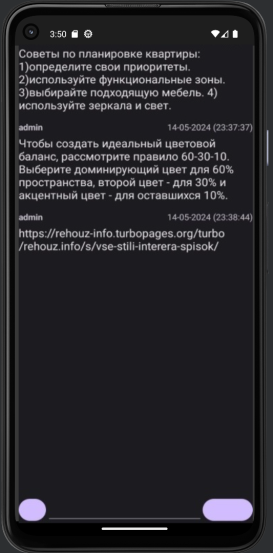
}

**Тестирование**

**4.1 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 9–10.

  
Рисунок 9 – Экраны публикаций

Элементы интерфейса в обоих случаях отображаются корректно.

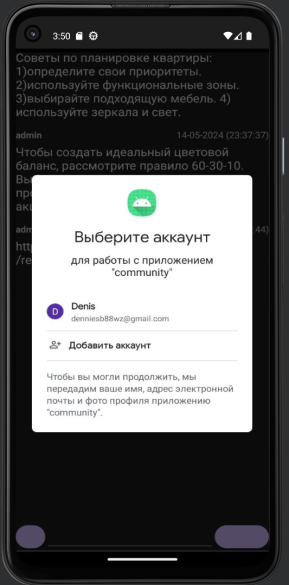
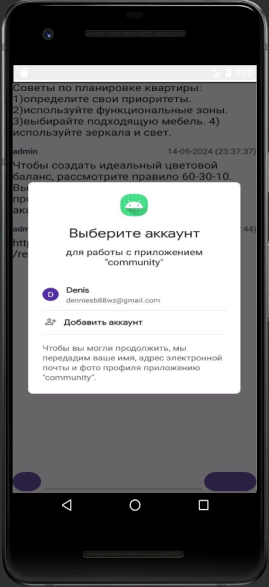


Рисунок 10 – Экраны авторизации

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

**4.2 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 публикаций на экран.

Таблица 5 – Тестирование функции вывода публикаций на экран

|  |  |  |
| --- | --- | --- |
| Название: | Мобильное приложение для планировки интерьера квартиры | |
| Функция: | Вывод публикаций на экран | |
| Действие | Ожидаемый результат  1.Список публикаций отображается без задержек и ошибок  2.Изображения и тексты четкие и корректно отформатированы  3.Пользователи могут оставлять публикации | Результат теста:  Совпадает с ожидаемым |
| Предусловие: | 1.У Пользователя установлено приложение на телефоне.  2.Пользователь авторизован.  3.У Пользователя имеется доступ к интернету. | |
| Запустить приложение | Приложение открылось корректно |  |
|  |  |  |
| Шаги теста (positive): |  | |
| 1.Запустить приложение на телефоне | Приложение запустилось | пройден |
| 2.Авторизоваться в приложении | Авторизация прошла успешно | пройден |
| 3.Оставить публикацию | Публикация загружена и отображается в ленте | пройден |

В таблице 6 представлено тестирование функции авторизации.

|  |  |  |
| --- | --- | --- |
| Название: | Мобильное приложение для планировки интерьера квартиры | |
| Функция: | Авторизация пользователей. | |
| Действие | Ожидаемый результат  1.Данные аккаунта введены верно, пользователь вошел в систему  2.Пользователю отображается главный экран приложения | Результат теста:  Совпадает с ожидаемым |
| Предусловие: | 1.У Пользователя установлено приложение на телефоне.  2.Наличие аккаунта у пользователя.  3.У Пользователя имеется доступ к интернету. | |
| Запустить приложение | Приложение открылось корректно |  |
|  |  |  |
| Шаги теста (positive): |  | |
| 1.Запустить приложение на телефоне | Приложение запустилось | пройден |
| 2.В всплывающем окне ввести корректные учетные данные | Данные введены верно | пройден |
| 3.Нажать на кнопку авторизации | Пользователь вошел в систему | пройден |

Разработанные TestCase подтверждают правильность функционирования приложения. В ходе тестирования не обнаружено недочётов в оформлении или логической структуре программы. Проверка на разнообразных API показала, что все элементы интерфейса отображаются корректно на соответствующих экранах.

**Заключение**

В ходе разработки мобильного приложения для планировки интерьера квартиры были учтены основные потребности пользователей, желающих упростить процесс организации пространства своего жилища. Приложение предоставляет ограниченныйнабор функций, что позволяет пользователям эффективно управлять своими публикациями и делиться ими с сообществом.

**Авторизация** в приложении позволяет пользователям создавать личный аккаунт, что обеспечивает безопасный доступ к их личным данным и публикациям.

Пользователи могут выкладывать фотографии и описания своих проектов, облегчая процесс обмена идеями и получения обратной связи. Это способствует созданию сообщества, где каждый может найти вдохновение и поддержку.

Интерфейс приложения разработан таким образом, чтобы обеспечить легкость в использовании и минимизировать количество шагов, необходимых для достижения желаемого результата. Это позволяет пользователям сосредоточиться на планировке, а не на навигации по приложению.

Тестирование подтвердило, что приложение удобно в использовании и отвечает основным требованиям пользователей, что является важным шагом в достижении целей проекта.

Таким образом, разработка приложения для планировки интерьера квартиры успешно завершена. Работа над проектом заложила основу для будущего развития и добавления новых функций, что сделает приложение еще более адаптированным к потребностям пользователей.

Библиография

Нормативно-правовые акты:

1 ГОСТ Р 2.105-2019. ЕСКД. Общие требования к текстовым документам. – Москва: Стандартинформ, 2019. – 36 с

Электронные ресурсы:

1 Android Developers [Электронный ресурс]. – Firebase

– URL: <https://developer.android.com/studio/write/firebase>

2 Firebase Documents [Электронный ресурс]. – Документация Firebase – URL: <https://firebase.google.com/docs?hl=en>

3 Google Identity [Электронный ресурс]. – Руководство по интеграции авторизации Google в мобильные приложения. – URL: <https://developers.google.com/identity?hl=ru>

4 Mobile UI/UX Design Android Developers [Электронный ресурс]. – Основной макет – URL: <https://developer.android.com/design/ui/mobile/guides/layout-and-content/layout-basics>

5 Firebase Realtime [Электронный ресурс]. – Firebase в реальном времени – URL: <https://firebase.google.com/docs/database>

6 Android Architecture [Электронный ресурс]. – Рекомендации по архитектуре Android-приложений. – URL: <https://developer.android.com/topic/architecture>

**Приложение А**

Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж

имени Б.С. Галущака»

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПЛАНИРОВКИ ИНТЕРЬЕРА КВАРТИРЫ

Техническое задание

НАТКиГ.200300.010.000ПЗ

Выполнил:

Студент группы ПР-22.106

Бочаров Д.В.

2024

**Содержание**

Введение 30

1 Назначения разработки 31

2 Требования к мобильному приложению 31

2.1 Требования к функциональным характеристикам 31

2.2 Требования к надёжности 32

2.3 Условия эксплуатации 32

2.4 Требования к составу и параметрам технических средств 32

2.5 Требования к информационной и программной совместимости 32

2.6 Требования к защите информации 32

2.7 Требования к маркировке и упаковке 32

3 Требования к программной документации 33

4 Технико-экономические показатели 33

5 Стадии и этапы разработки 33

6 Порядок контроля и приёмки 34

**Введение**

Настоящее техническое задание распространяется на разработку мобильного приложения «Разработка мобильного приложения для планировки интерьера квартиры», используемого для просмотра и загрузки фотографий и других публикаций.

Наименование приложения: «InterioPlan».

Краткая характеристика области применения: мобильное приложение для планировки интерьера квартиры предоставляет пользователям функциональные возможности для создания и публикации фотографий планировочных решений, а также просмотра и обсуждения различных планировочных проектов других пользователей. Возможность авторизации через Google гарантирует удобный и безопасный доступ к личному аккаунту и сохранность данных. Приложение служит эффективным инструментом для воплощения планировочных идей, позволяя обмениваться опытом и находить новые идеи для оптимизации жилого пространства.

Основанием для проведения разработки является Протокол №Уч-29/4 от «22» февраля 2024г.

Наименование темы разработки − «Разработка мобильного приложения для планировки интерьера квартиры».

Условное обозначение темы разработки – «InterioPlan».

**1 Назначение разработки**

Основное назначение мобильного приложения для планировки интерьера квартиры заключается в следующем:

* Обеспечение удобного и интуитивно понятного интерфейса для пользователя, позволяющего легко ориентироваться в функциях приложения.
* Предоставление обширной библиотеки фотографий планировочных решений для вдохновения и обмена идеями.

Лица, которые могут работать с данной системой:

* Администратор – обладает полным доступом ко всем функциям приложения, отвечает за его стабильную работу, обновляет и поддерживает актуальность базы данных.
* Пользователь приложения – имеет возможность публиковать собственные фотографии планировок, просматривать публикации других пользователей.

**2 Требования к мобильному приложению**

**2.1 Требования к функциональным характеристикам**

Требования к составу выполняемых функций:

* Авторизация пользователей;
* Просмотр и возможность публикации фотографий, постов;

**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 Требования к 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 |  | Пояснительная записка |
| 2 | Описание предметной области |  | Пояснительная записка |
| 3 | Выбор технологии, языка и среды программирования |  | Пояснительная записка |
| 4 | Оформление технического задания |  | Техническое задание |
| 5 | Проектирование UI/UX дизайна |  | Спецификации программного обеспечения |
| 6 | Разработка мобильного приложения |  | Схема структурная системы и спецификации компонентов |
| 7 | Разработка базы данных |  | Программный  продукт |
| 8 | Отладка и тестирование приложения |  | Тексты программных компонентов |

Окончание таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | Оформление документации |  | Программная  документация |
| 10 | Защита |  |  |

**6 Порядок контроля и приёмки**

Виды испытаний – защита курсового проекта.

Общее требования к приёмке:

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

# Приложение Б

# Авторизация и вывод сообщений

public class MainActivity extends AppCompatActivity {  
  
 private GoogleSignInClient mGoogleSignInClient;  
 private Button btnCl;  
 private FirebaseAuth auth;  
 private ActivityResultLauncher<Intent> resultLauncher = registerForActivityResult(new ActivityResultContracts.StartActivityForResult(), new ActivityResultCallback<ActivityResult>() {  
 @Override  
 public void onActivityResult(ActivityResult o) {  
 if(o.getResultCode() == Activity.RESULT\_OK)  
 {  
 Intent intent = o.getData();  
 Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(intent);  
 try {  
 GoogleSignInAccount account = task.getResult(ApiException.class);  
  
 assert account != null;  
 firebaseAuthWithGoogle(account.getIdToken());  
  
 } catch (ApiException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 }  
 });  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.enable(this);  
 setContentView(R.layout.activity\_main);  
  
 auth = FirebaseAuth.getInstance();  
 GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT\_SIGN\_IN)  
 .requestIdToken(String.valueOf(com.firebase.ui.auth.R.string.default\_web\_client\_id)).requestEmail().build();  
  
 mGoogleSignInClient = GoogleSignIn.getClient(this, gso);  
  
 resultLauncher.launch(new Intent(mGoogleSignInClient.getSignInIntent()));  
  
  
 btnCl = findViewById(R.id.button\_set);  
 btnCl.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 EditText textField = findViewById(R.id.messageField);  
 String sTextField = textField.getText().toString();  
  
 if (textField.getText().toString().trim().length() != 0) {  
 FirebaseDatabase.getInstance()  
 .getReference()  
 .push()  
 .setValue(new Message(  
 "admin",  
 textField.getText().toString())  
  
 );  
 }  
  
 textField.setText("");  
 }  
 });  
  
  
 FirebaseDatabase database = FirebaseDatabase.getInstance();  
 Query messages = database.getReference();  
  
 FirebaseListOptions<Message> options = new FirebaseListOptions.Builder<Message>().setQuery(messages, Message.class).setLayout(R.layout.item).build();  
  
 ListView messageList = findViewById(R.id.list\_view);  
 FirebaseListAdapter<Message> adapter = new FirebaseListAdapter<Message>(options){  
 @Override  
 protected void populateView(View v, Message model, int position) {  
 TextView messageText = v.findViewById(R.id.message\_Text);  
 TextView messageUser = v.findViewById(R.id.message\_User);  
 TextView messageTime = v.findViewById(R.id.message\_Time);  
  
 messageText.setText(model.getMessage\_text());  
 messageUser.setText(model.getMessage\_name());  
 messageTime.setText(DateFormat.format("dd-MM-yyyy (HH:mm:ss)",  
 model.getMessage\_time()));  
 }  
 };  
  
 messageList.setAdapter(adapter);  
 adapter.startListening();  
  
 }  
  
 private void firebaseAuthWithGoogle(String idToken) {  
 AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);  
 auth.signInWithCredential(credential).addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 if(task.isSuccessful())  
 {  
 startActivity(new Intent(MainActivity.this, MainActivity.class));  
 finish();  
 Toast.makeText(MainActivity.this, "Successful login",Toast.LENGTH\_LONG).show();  
 }  
 else {  
 Toast.makeText(MainActivity.this, "Filed login",Toast.LENGTH\_LONG).show();  
 }  
 }  
 });  
 }  
  
 private void getImage()  
 {  
 Intent intentCh = new Intent();  
 intentCh.setType("image/\*");  
 intentCh.setAction(Intent.ACTION\_GET\_CONTENT);  
 }  
  
  
}

public class Message {  
  
 public String message\_name;  
 public String message\_text;  
 public long message\_time;  
 public ImageView image\_view;  
  
 Message() {}  
 Message(String N,String T) {  
 this.message\_name = N;  
 this.message\_text = T;  
  
 this.message\_time = new Date().getTime();  
 this.image\_view = null;  
 }  
  
 Message(String N,String T, ImageView I) {  
 this.message\_name = N;  
 this.message\_text = T;  
  
 this.message\_time = new Date().getTime();  
 this.image\_view = I;  
 }  
  
 public String getMessage\_name() {  
 return message\_name;  
 }  
  
 public void setMessage\_name(String message\_name) {  
 this.message\_name = message\_name;  
 }  
  
 public String getMessage\_text() {  
 return message\_text;  
 }  
  
 public void setMessage\_text(String message\_text) {  
 this.message\_text = message\_text;  
 }  
  
 public long getMessage\_time() {  
 return message\_time;  
 }  
  
 public void setMessage\_time(long message\_time) {  
 this.message\_time = message\_time;  
 }  
}