МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА:**

По дисциплине «Программирование Мобильных Систем»

Тема **«**Мобильное приложение агентства по аренде недвижимости»

**Исполнитель**

студент 3 курса группы 8 Иванова С.И.

подпись, дата

**Руководитель**

ассистент Уласевич Н.И.

должность, ученая степень, ученое звание подпись, дата

Допущен(а) к защите \_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Уласевич Н.И.

подпись дата инициалы и фамилия

**Содержание**

[Введение 5](#_Toc166927268)

[1 Постановка задачи 6](#_Toc166927269)

[2 Обзор аналогичных решений 7](#_Toc166927270)

[2.1 Мобильное приложение Realt.by 7](#_Toc166927271)

[2.1 Программное средство «Flatbook» 8](#_Toc166927272)

[2.3 Мобильное приложение «Domovita» 9](#_Toc166927273)

[2.4 Вывод по разделу 10](#_Toc166927274)

[3 Проектирования приложения 11](#_Toc166927275)

[3.1 Проектирование базы данных 11](#_Toc166927276)

[3.2 Проектирование ролей 11](#_Toc166927277)

[3.3 Проектирование навигации 12](#_Toc166927278)

[3.4 Описание структуры проекта 12](#_Toc166927279)

[3.5 Вывод по разделу 14](#_Toc166927280)

[4 Реализация приложения 15](#_Toc166927281)

[4.1 База данных и взаимодействие с ней 15](#_Toc166927282)

[4.2 Авторизация, регистрация 16](#_Toc166927283)

[4.3 Вывод по разделу 17](#_Toc166927284)

[5 Тестирование приложения 18](#_Toc166927285)

[5.1 Вывод по разделу 20](#_Toc166927286)

[6 Руководство пользователя 21](#_Toc166927287)

[6.1 Вывод по разделу 26](#_Toc166927288)

[Заключение 27](#_Toc166927289)

[Список используемых источников 28](#_Toc166927290)

[Приложение А. Логическая диаграмма базы данных 29](#_Toc166927291)

[Приложение Б. Диаграмма вариантов использования 30](#_Toc166927292)

[Приложение В. Классы для работы с базой данных 31](#_Toc166927293)

# Введение

В современном мире, где удобство, эффективность и надежность играют ключевую роль в повседневной жизни, сфера аренды недвижимости становится особенно важной. Мобильные приложения, предоставляющие услуги по аренде недвижимости, становятся неотъемлемой частью этой динамично развивающейся индустрии.

Для успешного функционирования такого приложения неотъемлемым компонентом является качественное программное обеспечение, спроектированное с учетом потребностей как собственников недвижимости, так и арендаторов. Технология Flutter, с использованием языка программирования Dart, предоставляет непревзойденные возможности для создания современных, интуитивно понятных и визуально привлекательных мобильных приложений.

Основной целью в рамках данного проекта является разработка мобильного решения, обеспечивающего эффективное управление арендой недвижимости с использованием технологии Flutter. Основные аспекты проекта включают в себя создание интуитивного пользовательского интерфейса, разработку функционала для учета объектов недвижимости, обеспечение механизмов бронирования и аналитики.

Цель проекта – предоставить надежное и удобное мобильное приложение, полностью соответствующее требованиям современного рынка аренды недвижимости. Результатом работы будет готовое приложение, адаптированное под современные технологические тренды и специфику бизнеса в области аренды недвижимости. Это позволит приобрести ценный опыт в разработке высокофункциональных программных решений на базе Flutter и Dart.

Разработанное приложение будет представлять собой универсальный инструмент для собственников недвижимости, агентств по аренде и самих арендаторов города Минск. Оно позволит упростить процессы поиска, бронирования и управления объектами недвижимости, обеспечивая максимальную прозрачность и комфорт для всех сторон взаимодействия.

1. **Постановка задачи**

Задачей курсового проекта является разработка мобильного приложения на платформе Flutter, которое содержит четыре роли: гость, пользователь-арендодатель, пользователь-арендатор.

Функции администратора:

* Авторизация. Вход в систему с использованием уникальных учетных данных;
* добавление объявлений. Администратор имеет право добавлять новые объявления о недвижимости в базу данных. Это включает в себя заполнение всех необходимых данных, таких как описание, характеристики, фотографии и т.д.;
* удаление объявлений. Администратор может удалить определенное объявление из базы данных, если оно не соответствует требованиям или нарушает правила платформы;
* Изменение объявления. Администратор имеет возможность редактировать существующие объявления, позволяя вносить изменения в описание, характеристики и другие детали.

Функции пользователя-арендатора:

* Авторизация/регистрация. Пользователь может зарегистрироваться в системе, предоставив уникальные учетные данные, либо авторизоваться, если у него уже есть аккаунт;
* просмотр объявлений. Пользователь имеет доступ к списку объявлений о недвижимости с возможностью просмотра всех деталей, включая описание, характеристики и фотографии;
* добавление объявления в избранное;
* написание отзыва. Пользователь может оставить свой отзыв о недвижимости, которую он арендовал.

Функции пользователя (арендодателя):

* Предложение объявления. Собственник недвижимости может добавить свою недвижимость в систему. После этого администратор может проверить и опубликовать объявление или отклонить его;
* просмотр отзывов.

Дополнительные возможности:

* Сортировка. Все пользователи могут сортировать объявления по различным параметрам, таким как цена, площадь, количество комнат и т.д.;
* фильтрация. Пользователи могут применять фильтры для поиска недвижимости по определенным критериям, например, по типу объекта, району, наличию мебели и другим параметрам.

# Обзор аналогичных решений

В настоящее время существует множество программных решений, ориентированных на сферу управления арендой недвижимости, которые представлены на различных специализированных платформах. Перед началом разработки необходимо провести анализ и изучение аналогичных приложений, а также ознакомиться с соответствующей литературой и статьями, посвященными данной тематике. Этот этап позволяет выявить прототипы компаний, работающих в сфере аренды недвижимости, и использовать полученные знания для разработки высококачественного и конкурентоспособного продукта.

# 2.1 Мобильное приложение Realt.by

Приложение realt.by, представленное на рисунке 2.1 представляет собой мобильное приложение, предназначенную для поиска и аренды недвижимости в Республике Беларусь. В рамках данного веб-ресурса пользователи имеют возможность найти подходящее жилье, ознакомиться с его описанием и характеристиками, а также связаться с владельцами недвижимости.

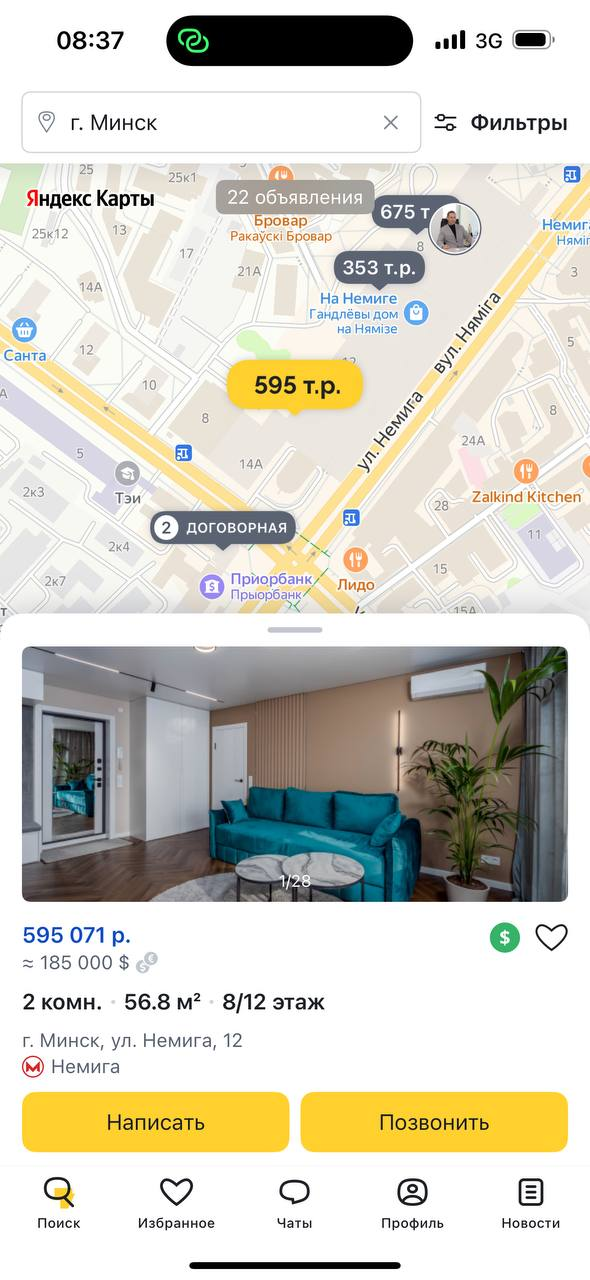


Рисунок 2.1 – Программное средство «Realt.by»

Анализировав мобильное приложение, можно выделить ряд его преимуществ и недостатков:

Основные плюсы:

* Удобный поиск недвижимости. Приложение предлагает интуитивно понятный интерфейс для поиска объектов аренды с различными фильтрами и сортировкой по различным параметрам;
* подробная информация о недвижимости. Пользователи имеют доступ к детальному описанию объектов, включая характеристики, фотографии и контактные данные владельцев;
* возможность связи с владельцами. Приложение предоставляет удобные средства обратной связи, что позволяет пользователям задавать вопросы или договариваться о деталях аренды.

Основные минусы:

* Отсутствие персонализации тарифов. Нет возможности настроить предпочитаемые параметры и фильтры для поиска недвижимости в соответствии с предпочтениями пользователя;
* дефицит подробной истории аренды. Не предоставляется информации о предыдущих арендаторах или о рейтинге объекта.

В целом, приложение realt.by представляет собой полезный и удобный ресурс для поиска недвижимости в Республике Беларусь. Однако, возможности по персонализации поиска и интеграции карт могли бы сделать использование платформы еще более эффективным и удобным для пользователей.

* 1. **Программное средство «Flatbook»**

Сайт https://flatbook.by, представленный на рисунке 1.2, – это интернет-платформа, предназначенная для поиска и аренды жилья в Беларуси. Пользователям предоставляется возможность ознакомиться с различными вариантами недвижимости, описанием и условиями аренды.

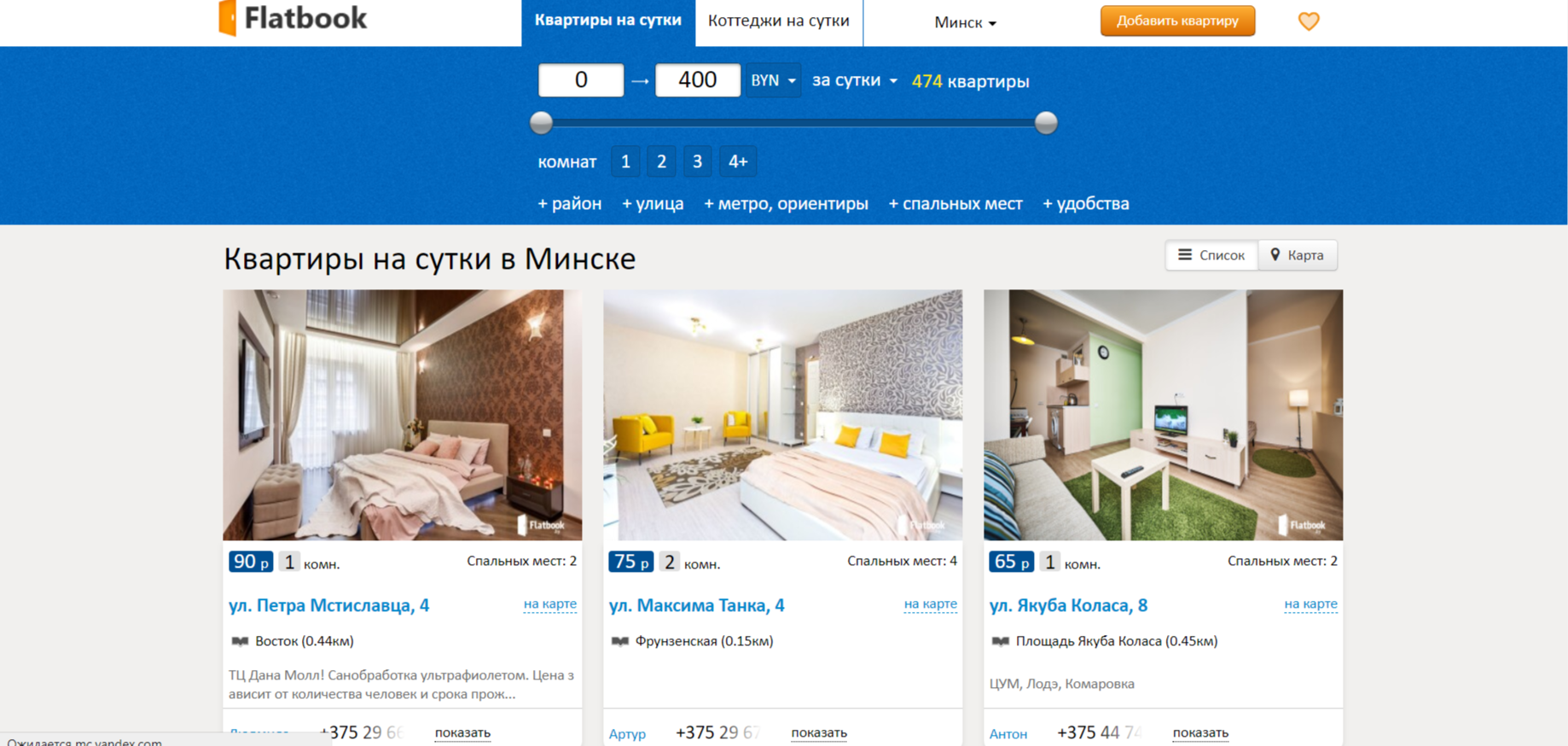


Рисунок 1.2 – Программное средство «Flatbook»

Проанализировав ресурс https://flatbook.by, можно выделить следующие характеристики:

Основные плюсы:

* Детальная информация о недвижимости. Сайт предоставляет подробные описания объектов с фотографиями, планами и характеристиками помещений, что помогает пользователям сориентироваться и выбрать подходящий вариант;
* фильтрация и поиск. Платформа предоставляет возможность настроить фильтры для поиска недвижимости по различным параметрам (цена, площадь, район и др.), что упрощает процесс выбора;
* удобство контакта с арендодателем. Пользователи могут легко связаться с владельцами недвижимости через встроенные средства обратной связи.

Основные минусы:

* Не все объекты имеют отзывы. Некоторые объекты могут быть недостаточно оценены и описаны, что может вызвать дополнительные вопросы у потенциальных арендаторов;
* ограниченный выбор в некоторых категориях. В некоторых районах или с определенными параметрами поиска может быть ограниченный выбор объектов.

В целом, сайт https://flatbook.by представляет собой полезный ресурс для поиска недвижимости в Беларуси. Улучшение навигации и интеграция карты могли бы сделать платформу еще более привлекательной и удобной для пользователей.

## **2.3 Мобильное приложение «Domovita»**

Приложение https://domovita.by, представленное на рисунке 2.3, – это платформа, предназначенная для поиска и аренды недвижимости в Беларуси.

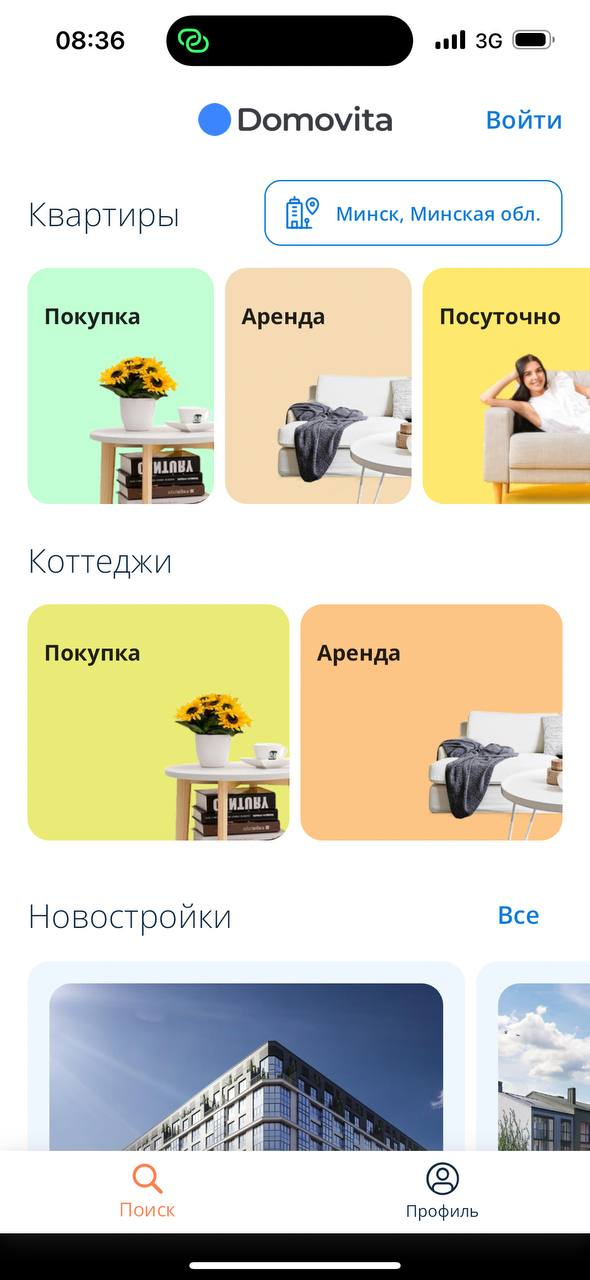


Рисунок 1.3 – Программное средство «Domovita»

Проанализировав приложение, можно выделить следующие характеристики:

Основные плюсы:

* Детальное описание объектов недвижимости. Приложение предоставляет подробные описания с фотографиями, планами помещений и характеристиками, что обеспечивает пользователям полную информацию для принятия решения;
* функционал поиска и фильтрации. Пользователи могут настраивать фильтры для поиска недвижимости по различным параметрам (цена, площадь, район и др.), что значительно упрощает процесс подбора;
* простота связи с арендодателями. Средства обратной связи позволяют пользователям легко связаться с владельцами недвижимости для обсуждения деталей.

Основные минусы:

* ограниченный выбор в некоторых категориях. В определенных районах или с определенными параметрами поиска может быть ограниченный выбор недвижимости;
* наличие неактуальных объявлений. Некоторые объявления могут быть устаревшими или неактуальными.
  1. **Вывод по разделу**

После рассмотрения аналогов можно сделать вывод, что примеры реализуют все функции, которые необходимо разработать в конечном приложении. В примерах используются элементы навигации, механизмы авторизации и регистрации. В обоих приложениях присутствует функция поиска рейсов с заданными условиями.

В разрабатываемом приложении должны быть реализованы элементы навигации, а также разработаны все необходимые формы: регистрации, авторизации, поиска рейса, бронирования билетов.

В целом, проведение анализа и изучения аналогичных приложений, литературы и прототипов компаний является неотъемлемой частью процесса разработки программного решения для управления арендой недвижимости. Этот этап позволяет получить ценные знания, которые помогут создать конкурентоспособный и высококачественный продукт, отвечающий потребностям пользователей в этой сфере.

1. **Проектирования приложения**

В этом разделе рассматриваются ключевые аспекты проектирования разрабатываемого приложения. Будет рассмотрена структура приложения, выбор технологий, а также схема навигации в приложении. Этот раздел поможет лучше понять, как разрабатываемое приложение будет работать и какие задачи должны быть выполнены для успешной реализации.

* 1. **Проектирование базы данных**

Первым этапом курсового проекта является разработка структуры базы данных, включающей логически взаимосвязанные таблицы. Для этого необходимо предварительно определить, какая информация будет храниться в каждой из этих таблиц, а затем спроектировать связи с использованием первичных и внешних ключей. После завершения проектирования структуры базы данных была создана диаграмма, которая наглядно отображает связи между таблицами и их атрибуты. Данная диаграмма, представленная в приложении А.

База данных, основанная на реляционной модели данных. В ее структуру входят следующие таблицы: USERS, Property, Photos, Favorites, Reviews.

Таблица USERS содержит информацию о зарегистрированных пользователях системы. Каждый пользователь имеет уникальный идентификатор UserID, а также данные о его имени, фамилии, типе пользователя (например, арендодатель или арендатор), номере телефона, электронной почте и пароле для входа в систему.

Таблица Property хранит данные о недвижимости, предложенной пользователями. Каждая запись содержит уникальный идентификатор (Id) и информацию о недвижимости.

Таблица Photos содержит изображения недвижимости, представленные в виде последовательности байт. Каждая фотография имеет свой уникальный идентификатор Id, а также связана с конкретной недвижимостью через поле PropertyId.

Таблица Favorites сохраняет связи между пользователями и недвижимостью, которая им нравится. Записи включают уникальный идентификатор Id, UserId, PropertyId и дату добавления в избранное AddingDate.

Таблица Reviews хранит отзывы пользователей о недвижимости. Каждый отзыв имеет уникальный идентификатор ReviewID и включает в себя информацию о недвижимости PropertyID, авторе отзыва ReviewerID, оценке Rating и комментарии Comment.

* 1. **Проектирование ролей**

Возможности каждой роли представлены на диаграмме использования в приложении Б. Диаграмма прецедентов – это графическое представление функциональности системы и взаимодействия между системой и ее актерами. Она помогает идентифицировать основные функциональные требования и сценарии использования системы.

Диаграмма прецедентов используется для следующих целей:

– определение актеров, которые взаимодействуют с системой;

– идентификация основных функций и возможностей системы;

– понимание сценариев использования системы;

– определение границ системы.

Диаграмма прецедентов является наглядным инструментом, который помогает объяснить функциональность системы и ее взаимодействие с актерами.

На основе разработанной логической схемы БД, описанного основного функционала программы и описания будущих окон приложения, были определены основные возможности пользователей приложения. Для наглядного отображения возможностей пользователя была создана User Case диаграмма, представленная на рисунке 2.2.

В процессе построения диаграммы использования в системе было выделено 4 актера: администратор, пользователь-арендатор, пользователь-арендодатель, гость. Среди выделенных актеров образуется следующая иерархия: администратору и всем пользователям доступны все функции гостя.

* 1. **Проектирование навигации**

Приложение будет содержать некоторое количество страниц, которые должны быть как-то связаны между собой. Схема навигации выглядит так: на главной странице у нас будет элемент с навигационным меню – BottomNavigationBar, каждая его вкладка представляет собой отдельную страницу. С главной страницы мы можем получить доступ к четырем основным страницам приложения. Страница «Профиль» представляет собой страницу, на которой отображается личная информация пользователя. Страница «Главная» содержит список объявлений. Страница «Мои объявления» также имеет условный рендер: если пользователь является администратором, то отображаются предложенные объявления, если нет – страница с опубликованными объявлениями. Страница «Избранное» содержит список объявлений, которое сохранил пользователь текущей сессии.

* 1. **Описание структуры проекта**

Pubspec.yaml – это конфигурационный файл, используемый в проектах Flutter и Dart. Этот файл содержит метаданные о проекте, его зависимости, а также другую важную информацию, необходимую для управления и сборки проекта. Содержимое данного файла представлено на листинге 3.1.

Пакеты, используемые в проекте, обеспечивают разнообразную функциональность и значительно улучшают пользовательский опыт. Пакет «font\_awesome\_flutter» предоставляет доступ к набору иконок «Font Awesome», добавляя стильные иконки для улучшения визуального оформления. Пакет «google\_nav\_bar» позволяет создавать современные и настраиваемые нижние панели навигации в стиле Google.

Пакет «sqflite» используется для локального хранения данных, предоставляя возможности управления данными оффлайн с помощью SQLite, а «path\_provider» помогает получить доступ к стандартным каталогам файловой системы, таким как временные файлы и документы. Пакет «shared\_preferences» используется для сохранения простых данных в виде пар ключ-значение, что полезно для хранения пользовательских настроек. Пакет «url\_launcher» позволяет приложению открывать внешние ссылки и запускать другие приложения, такие как браузеры. Также в проект добавлен «carousel\_slider», который добавляет возможность создания каруселей изображений или других элементов, улучшая визуализацию контента. Пакет «image\_picker» предоставляет функциональность для выбора изображений из галереи устройства или камеры, что позволяет пользователям загружать и редактировать фотографии.

Наконец, «photo\_view» обеспечивает удобный просмотр и масштабирование изображений, что делает взаимодействие с фотографиями более приятным.

|  |
| --- |
| dependencies:  flutter:  sdk: flutter  font\_awesome\_flutter: ^10.7.0  google\_nav\_bar: ^5.0.0  sqflite: ^2.0.0  path: ^1.8.0  path\_provider: ^2.0.2  firebase\_core: ^2.30.1  firebase\_auth: ^4.19.4  shared\_preferences: ^2.0.8  url\_launcher: ^6.0.9  carousel\_slider: ^4.2.1  image\_picker: ^1.1.1  photo\_view: ^0.15.0 |

Листинг 3.1 – Содержимое файла pubspec.yaml

Организация папки lib проекта представлена на рисунке 3.2. Она разделена на разделы: data, models, pages и utils. В директории models хранятся классы, представляющие объекты данных, такие как favorite, photo, property, review и user. В директории data находится класс DBHelper, обеспечивающий взаимодействие с базой данных.

В директории pages расположены классы, отвечающие за макеты страниц приложения, такие как account, advDialog, advEdit, bottomnavi, createAcc, filter, home, myproperty, otp, places\_explain, purpose, ReviewDialog, saved, signIn и startedscreen.

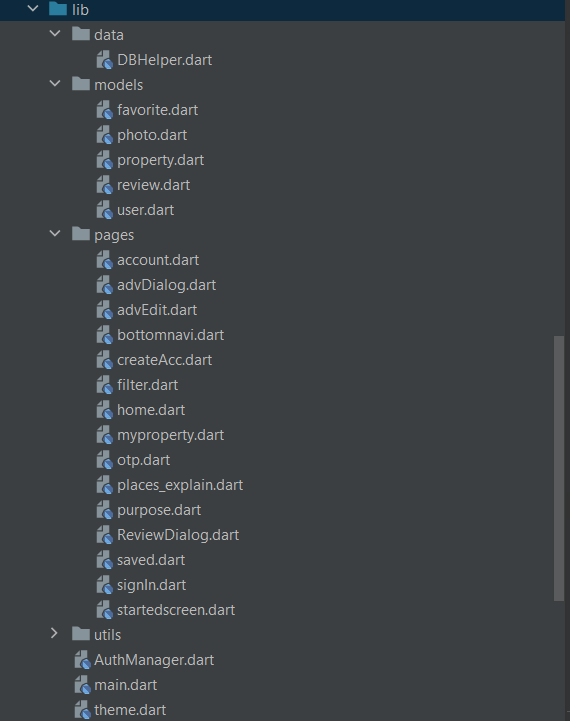


Рисунок 3.2 – Структура папки lib проекта

Файл main.dart определяет процесс инициализации приложения, а DatabaseNotifier организует взаимодействие с базой данных, создавая экземпляр DatabaseHelper в конструкторе, который в свою очередь в методе init вызывает остальные конструкторы классов для работы с базой данных.

* 1. **Вывод по разделу**

В данном разделе были рассмотрены ключевые элементы проекта. В частности, было выполнено проектирование базы данных приложения, включая его физическую и логическую структуру, которая будет поддерживать все функции приложения.

Также произведено проектирование ролей в контексте приложения, определяя различные уровни доступа и функциональность для каждой роли, была разработана диаграмма использования.

Была спроектирована модель навигации приложения с использованием схемы, обеспечивающей интуитивно понятный и эффективный пользовательский интерфейс.

Была описана структура проекта, включая все необходимые зависимости проекта с обоснованием необходимости их использования, а также файловую структуру проекта, обеспечивающую организованность и эффективность кода.

В целом, этот раздел предоставляет основу для дальнейшего развития и реализации проекта.

1. **Реализация приложения**

Данный раздел содержит в себе описание шагов разработки приложения и его элементов. Разработка приложения будет выполнена в IDE Android Studio, версия Flutter SDK – 3.19.5. Первоначальная реализация будет протестирована на Android-устройстве Google Pixel 7.

* 1. **База данных и взаимодействие с ней**

Для работы с базой данных разработан класс Databasehelper. Рассмотрим содержимое Databasehelper – Листинг 4.1. Полный скрипт работы с базой данных представлен в приложении В.

|  |
| --- |
| class DatabaseHelper {  static final DatabaseHelper \_instance = DatabaseHelper.\_internal();  factory DatabaseHelper() => \_instance;  static Database? \_database;  DatabaseHelper.\_internal();  Future<Database> get database async {  if (\_database != null) {  return \_database!;  }  \_database = await initDatabase();  return \_database!;  }  Future<Database> initDatabase() async {  Directory documentsDirectory = await getApplicationDocumentsDirectory();  String path = join(documentsDirectory.path, 'realetate2.db');  bool exists = await databaseExists(path);  if (!exists) {  // Копирование файла базы данных из assets  ByteData data = await rootBundle.load('assets/realetate2.db');  List<int> bytes = data.buffer.asUint8List();  await File(path).writeAsBytes(bytes);  }  return await openDatabase(  path,  version: 1,  );} |

Листинг 4.1 – Содержимое класса DatabaseHelper

В данном классе мы определяем экземпляр DatabaseHelper, который будет единственным в приложении благодаря реализации паттерна синглтон. Это достигается с помощью статического поля \_instance, которое инициализируется через внутренний конструктор DatabaseHelper.\_internal(). Конструктор класса возвращает этот единственный экземпляр, что обеспечивает централизованное управление базой данных в приложении.

Класс DatabaseHelper хранит поле \_database, представляющее экземпляр базы данных, и метод initDatabase, который отвечает за инициализацию этого поля. В методе database проверяется, был ли уже инициализирован экземпляр базы данных. Если \_database не равен null, возвращается существующий экземпляр, иначе вызывается метод initDatabase для его инициализации.

В методе initDatabase происходит определение пути к файлу базы данных в директории документов приложения. Если база данных не существует по указанному пути, она копируется из ресурсов приложения (assets). Для этого загружается файл базы данных из assets, преобразуется в список байтов и записывается в указанный путь. После этого база данных открывается с помощью метода openDatabase, и возвращается ее экземпляр.

Этот класс DatabaseHelper используется как глобальное состояние, обеспечивая централизованный доступ к базе данных для различных частей приложения. Это упрощает работу с базой данных, обеспечивая единый интерфейс для доступа и инициализации, а также гарантирует, что только один экземпляр базы данных используется в приложении.

Также созданы классы, которые будут представлять модель нашей базы данных. Их названия и наличие можно увидеть на рисунке 3.2 в директории Database. Более полное содержимое классов можно найти в приложении Б.

* 1. **Авторизация, регистрация**

В данной секции описывается механизм авторизации, регистрации и управления личным кабинетом пользователя в приложении. В основе этого функционала лежит класс AuthManager, который управляет состоянием аутентификации пользователя.

Для проверки состояния входа пользователя используется метод isLoggedIn, который обращается к хранилищу SharedPreferences и проверяет наличие ключа \_loggedInKey. Если значение этого ключа установлено в true, это означает, что пользователь уже авторизован. Процесс входа в систему реализован в методе login. При вызове этого метода, сначала проверяются учетные данные пользователя (email и пароль) с помощью класса DatabaseHelper, реализация авторизации представлена на листинге 4.2.

Метод checkCredentials предназначен для проверки учетных данных пользователя (email и пароль) в базе данных. Этот метод асинхронный и возвращает значение типа Future<bool>, указывающее, успешна ли проверка. Сначала метод ожидает инициализации и получения экземпляра базы данных, вызывая геттер database класса DatabaseHelper. Это обеспечивает доступ к базе данных, с которой будет выполнен запрос.

|  |
| --- |
| Future<bool> checkCredentials(String email, String password) async {  Database db = await database;  List<Map<String, dynamic>> result = await db.query(  'users',  where: 'Email = ? AND Password = ?',  whereArgs: [email, password],  );  return result.isNotEmpty;  } |

Листинг 4.2 – Реализация авторизации

Метод выполняет запрос к таблице users в базе данных. Запрос проверяет наличие записи, где столбцы Email и Password совпадают с переданными значениями аргументов email и password. Для защиты от SQL-инъекций используются параметры запроса whereArgs.

После выполнения запроса метод проверяет, не пуст ли список результатов. Если список не пустой, это означает, что учетные данные верны, и в базе данных найдена соответствующая запись. Метод возвращает true. В противном случае, если список пуст, метод возвращает false, указывая на неверные учетные данные.

Класс AuthManager обеспечивает централизованное управление процессом аутентификации в приложении. Он взаимодействует с локальной базой данных через DatabaseHelper для проверки учетных данных и хранения состояния аутентификации в SharedPreferences, что обеспечивает удобный и безопасный процесс авторизации и выхода из системы.

* 1. **Вывод по разделу**

В данном разделе была рассмотрена реализация главных функций приложения – регистрация, авторизация, просмотр объявлений, управление объявлениями, панель администратора, личный кабинет пользователя, а также технологии и подходы, которые обеспечили их реализацию.

1. **Тестирование приложения**

Тестирование работоспособности является заключительным этапом проектировании приложения. Разработанная в рамках курсового проекта платформа поддерживает три вида ролей у пользователя: администратор, пользователь-арендатор, пользователь-арендодатель. Рассмотрим возможности каждого из них.

При некорректном логине или пароле пользователю выводится оповещение, представленное на рисунке 5.1, уведомляющее о том, что были введены некорректные данные.

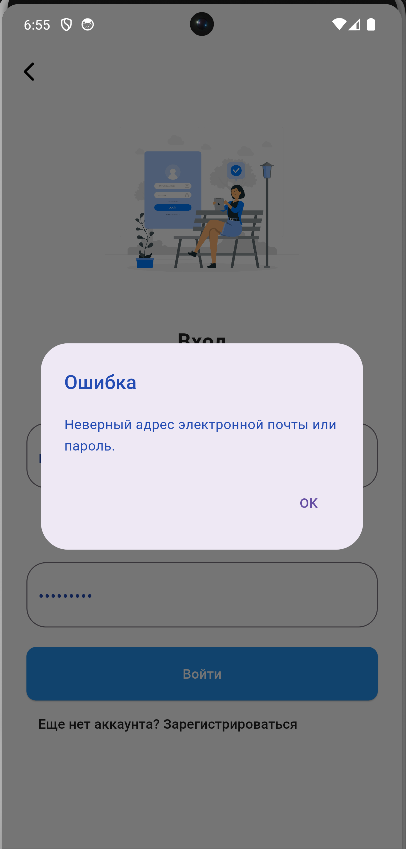


Рисунок 5.1 – Некорректные данные для входа

После неудачного входа пользователь может зарегистрироваться, на рисунке 5.2 представлена проверка полей при регистрации.

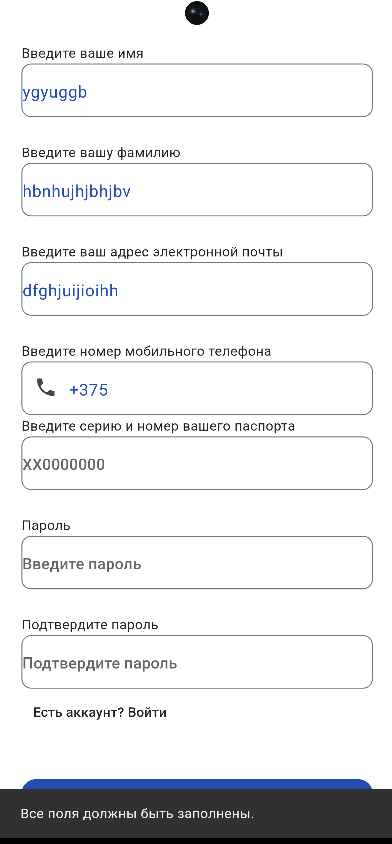


Рисунок 5.2 – Валидация в окне Регистрации

Валидацию полей можно рассмотреть более подробно. Поле электронной почты проверяется регулярным выражением на формат Email,проверка представлена на рисунке 5.3.

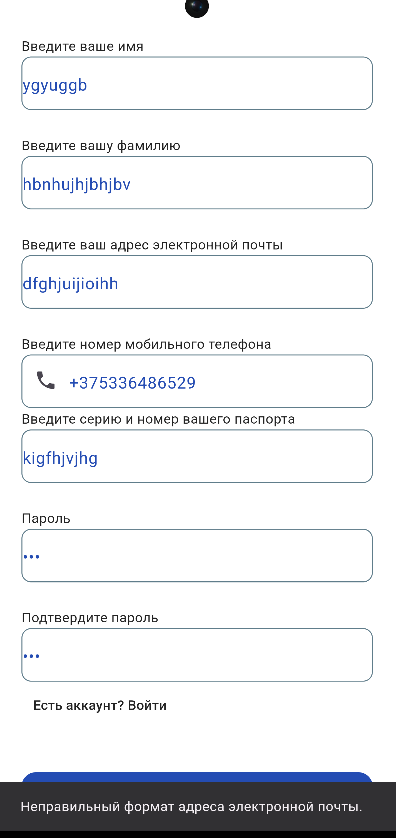


Рисунок 5.3 – Проверка электронной почты

А также значение этого поля сравнивается с другими email базы данных, во избежание одинаковых логинов, сообщение об ошибке представлено на рисунке 5.4.

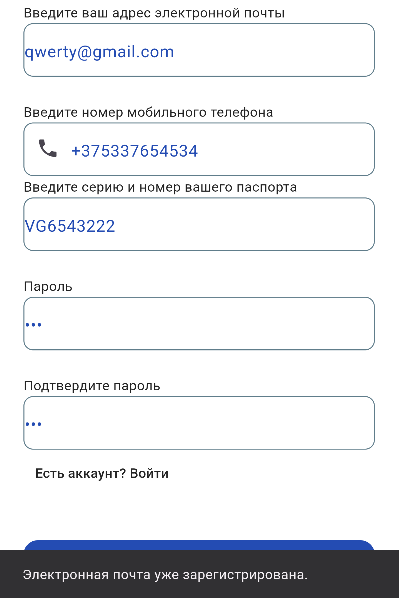


Рисунок 5.4 – Проверка электронной почты на уникальность

На рисунке 5.5 представлена валидация полей, при добавлении нового объявления.

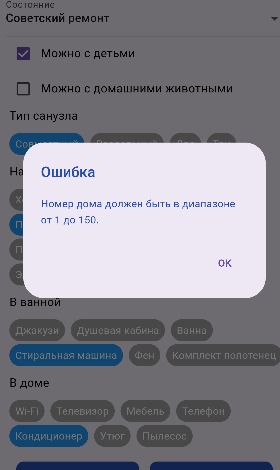


Рисунок 5.5 –Валидация при добавлении объявления

Кроме валидации также есть проверка на пустые значения, представленная на рисунке 5.6.

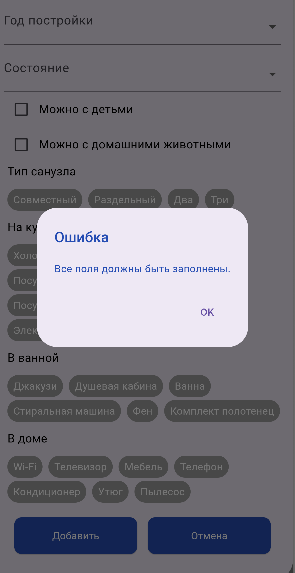


Рисунок 5.6 – Проверка на пустые значения при добавлении объявления

* 1. **Вывод по разделу**

В ходе тестирования была проверена обработка исключений и было подтверждено, что пользователь не сможет случайно выбить не обрабатывающееся исключение, что приведет к сбою выполнения программы, проверена валидация полей. Подтвердилось, что история созданий содержит все последние созданные элементы, которые можно сохранить и/или удалить с помощью кнопок, которые располагаются рядом с каждым элементом. Проведенный анализ получаемых данных подтвердил работоспособность программного средства.

1. **Руководство пользователя**

При запуске приложения открывается окно входа, которое предоставляет авторизованным пользователям доступ к функциям и ресурсам платформы.

Это окно, как правило, представлено формой ввода логина (адреса электронной почты) и пароля. Аутентификация происходит путем сопоставления введенных пользователем учетных данных с данными, хранящимися в базе данных.

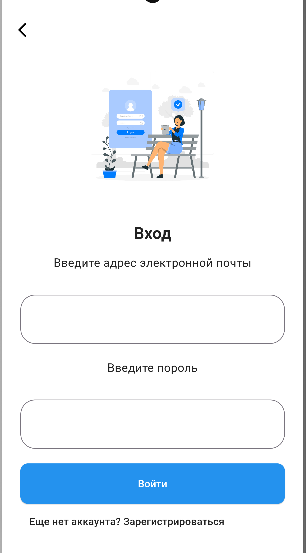


Рисунок 5.1 – Окно Входа

В случае, если у пользователя нет аккаунта, то он может перейти в окно регистрации, отображенного на рисунке 6.2. Для регистрации пользователь должен ввести корректно свои данные. А именно имя, фамилию, пароль, адрес электронной почты, серия и номер паспорта.

Для вводимых полей предусмотрена валидация. Пользователь не сможет зарегистрироваться, если хотя бы одно из обязательных полей пустое. Также у пользователя не будет возможности зарегистрироваться с адресом электронной почты, номер телефона или серией и номером паспорта, которые являются неуникальными.

При регистрации происходит проверка на уникальность этих данных. При процессе регистрации также обеспечивается проверка на совпадение пароля и поля подтверждения пароля, гарантируя, что пользователь вводит один и тот же пароль дважды для подтверждения его правильности. Это помогает предотвратить случайные ошибки при вводе пароля. В случае успешного прохождения всех проверок и валидации, пользователь получает уведомление о успешной регистрации и перенаправляется на страницу входа в систему. В случае возникновения ошибок, пользователю отображаются соответствующие сообщения об ошибках, чтобы он мог исправить введенные данные.

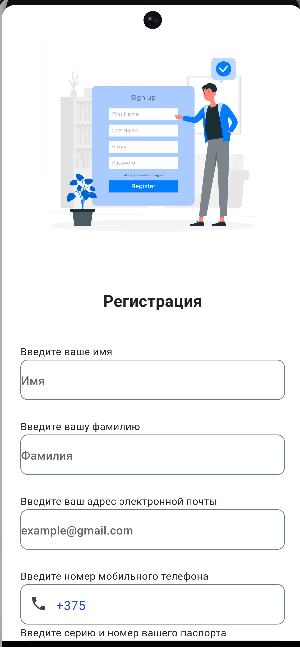


Рисунок 6.2 – Окно Регистрации

После успешной регистрации пользователь снова возвращается в окно входа и попробует заново авторизироваться. Важно отметить, что при повторной попытке авторизации пользователь должен ввести свои учетные данные, такие как адрес электронной почты и пароль, которые были указаны при регистрации. При этом система выполняет проверку введенных данных на корректность и соответствие существующим учетным записям. Если данные введены правильно и соответствуют тем, что были сохранены в базе данных во время регистрации, пользователь успешно авторизуется и получает доступ к системе.

После успешной авторизации пользователю отображается окно Каталог, представленное на рисунке 6.3, функционал которого настроен под тип пользователя. В зависимости от роли пользователя (например, арендатор, арендодатель или администратор), интерфейс и доступные функции могут различаться. Для арендаторов Каталог может включать в себя список доступных для аренды объектов недвижимости с возможностью фильтрации по различным параметрам, таким как стоимость аренды, размер площади, количество комнат и другие характеристики. Пользователь может просматривать детальную информацию о каждом объекте, включая фотографии, описание, условия аренды и контактные данные арендодателя.

Для арендодателей Каталог может предоставлять функционал для управления своими объектами недвижимости. Они могут добавлять новые объекты, редактировать информацию о существующих, обновлять статус (например, доступен для аренды или занят), а также просматривать заявки от потенциальных арендаторов.

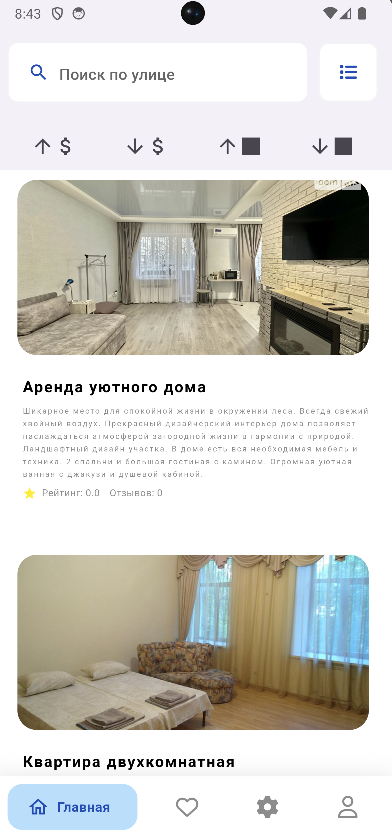


Рисунок 6.3 – Окно Каталог для пользователя

На рисунке 6.4 отображена панель для фильтрации данных. Она является доступной для всех типов пользователей.

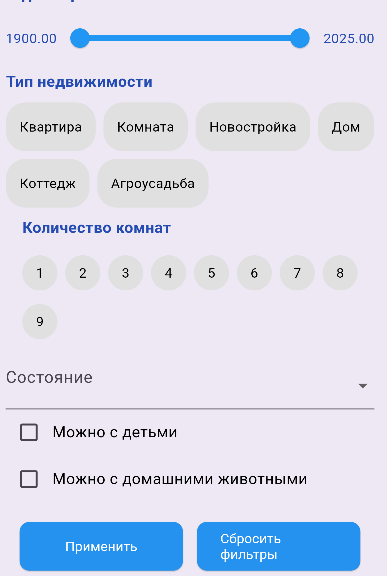


Рисунок 6.4 – Панель поиска и фильтрации

В данном окне расположено меню, с которым может взаимодействовать пользователь. Ему доступен такой функционал как, доступ к избранному, сортировка, обновление данных и выход. Объекты пользователь может добавить в избранное или посмотреть их подробную информацию, которая также доступна для всех пользователей, представлена на рисунке 5.5, о нем.

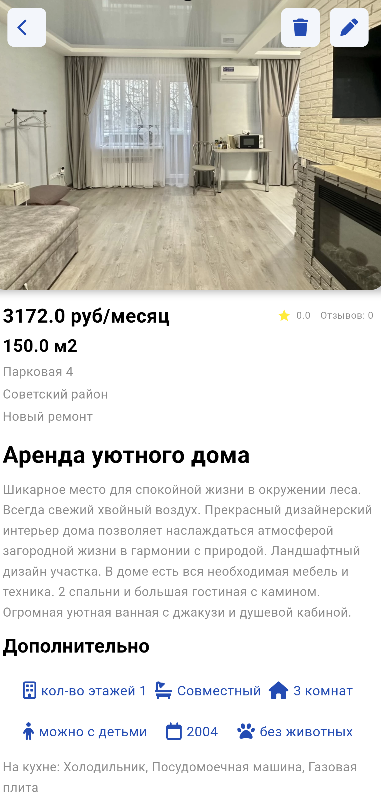


Рисунок 6.5 – Подробная информация об объекте

В данном окне в правом верхнем углу отображаются кнопки для взаимодействия с объявлением. Кнопки отображаются в зависимости от пользователя и привязки объявления к определенному пользователю.

В данном окне также находятся отзывы к объявлению. Если пользователь является арендатором, у него имеется возможность добавить отзыв. Добавление отзыва представлено на рисунке 6.6.

Отзывы отображаются в виде карусели, где каждый отзыв содержит информацию о пользователе, оценке и тексте отзыва. Отзывы помогают другим пользователям получить представление о качестве и состоянии объекта недвижимости, а также о взаимодействии с арендодателем. Если пользователь является арендатором, у него имеется возможность добавить отзыв. Добавление отзыва представлено на рисунке 6.6, где пользователь может ввести текст отзыва и выбрать оценку, например, в виде звезд или баллов.

Процесс добавления отзыва прост и интуитивно понятен. Пользователь нажимает на кнопку "Добавить отзыв", после чего открывается форма для ввода текста отзыва и выбора оценки.

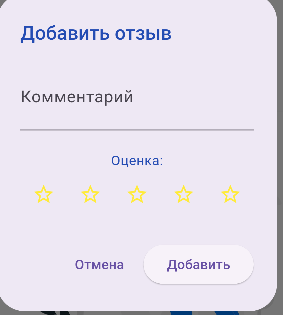


Рисунок 6.6 – Добавление отзыва

На рисунке 5.7 представлено окно избранного для пользователя текущей сессии.



Рисунок 6.7 – Избранное пользователя

Для арендодателя и администратора предусмотрена возможность добавления объявления (рисунок 6.8) с одним отличием: объявления добавленные пользователем не публикуются сразу, а помещаются в предложенное администратора, а объявления добавленные администратором публикуются сразу.

У пользователя есть возможность загрузить от 0 до 10 различных изображений. При нажатии на иконку добавления изображения открывается галерея, откуда пользователь сможет загрузить изображение к своему объявлению.



Рисунок 6.8 – Добавления объявления

При возникновении ситуации, когда пользователь случайно добавил ненужное изображение, при нажатии на него будет отображено диалоговое окно, с помощью которого пользователь сможет удалить или загрузить новое изображение.

* 1. **Вывод по разделу**

В данном разделе было представлено подробное руководство пользователя для каждой роли приложения с описание доступного функционала. Мы начали с процесса регистрации и авторизации, отметив важность корректного ввода учетных данных для обеспечения безопасного доступа к системе. Затем мы рассмотрели окно Каталога, которое является центральным элементом системы, предоставляя пользователям инструменты для поиска, фильтрации и управления объектами недвижимости. В этом окне пользователи могут находить подходящие варианты жилья, просматривать детальную информацию об объектах, добавлять объявления в избранное и связываться с арендодателями.

# Заключение

В рамках курсового проекта было разработано приложение, в котором был реализован такой функционал как:

* Вход и регистрация новых пользователей;
* управление объявлениями;
* удаление объявления;
* изменение объявления;
* добавление нового объявления;
* просмотр детальной информации об объявлении;
* подбор объявления по определенным фильтрам;
* поиск объявления по улице;
* сортировка доступных объявлений;
* добавление отзывов к объявлению.

Основной целью разработки данной системы было создание удобного и эффективного инструмента для поиска, аренды и управления объектами недвижимости. Система предоставляет пользователям удобный интерфейс для регистрации и авторизации, обеспечивает легкий доступ к каталогу недвижимости, а также включает функционал для размещения объявлений, управления ими и взаимодействия между арендаторами и арендодателями.

Особое внимание было уделено функциональности отзывов, которая позволяет арендатором оставлять обратную связь о качестве объектов и взаимодействии с арендодателями. Эта функция способствует созданию прозрачной и надежной среды для всех пользователей платформы.

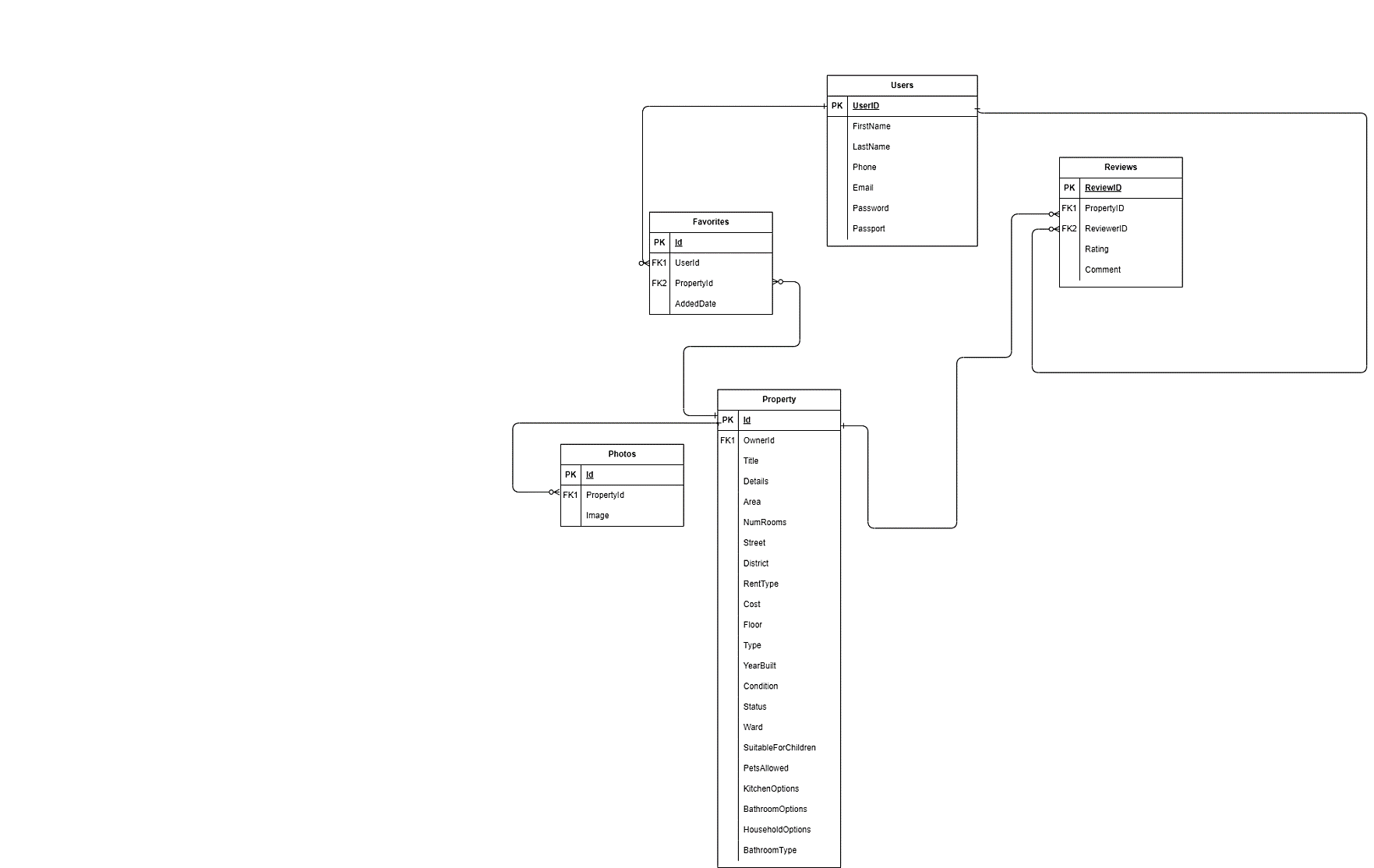
В процессе разработки системы были соблюдены современные стандарты программирования и дизайна, что обеспечило высокую производительность, безопасность и удобство использования. Была реализована адаптивная верстка, позволяющая комфортно работать с системой на различных устройствах, включая компьютеры, планшеты и смартфоны.

По завершению разработки проекта можно сказать, что поставленные цели были достигнуты.

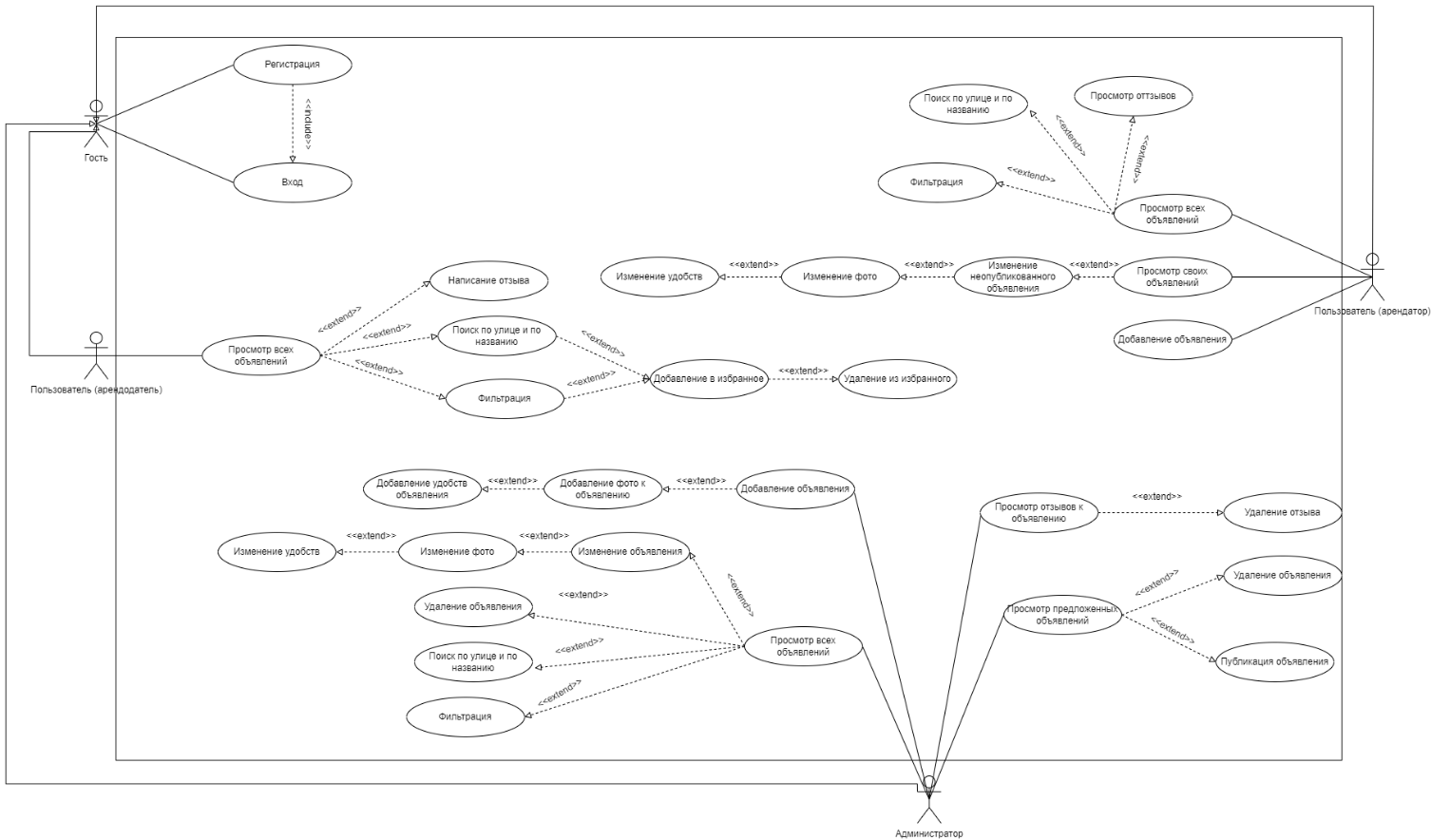
# Список используемых источников

1. Андрей Алеев, Быстрый старт Flutter-разработчика [Книга]/ А.В. Алеев. – Москва: Ridero, 2020. – 72 с. – Дата доступа: 29.02.2024
2. Alessandro Biessek. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2 [Книга]/ A. Biessek., 2019. – 450 с. Дата доступа – 05.03.2024.
3. Документация пакета sqflite [Электронный ресурс]. – Режим доступа: https://pub.dev/documentation/sqflite/latest/ – Дата доступа: 10.04.2024.
4. . Flutter API reference documentation [Электронный ресурс]. – Режим доступа: https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html – Дата доступа: 14.04.2024
5. Flutter & Dart from Stratch [Электронный ресурс]. –Pluralsight, 2022. – Режим доступа: https://www.pluralsight.com/courses/flutter-dart-getting-started – Дата доступа: 22.04.2024.

**Приложение А. Логическая диаграмма базы данных**



**Приложение Б. Диаграмма вариантов использования**



# Приложение В. Классы для работы с базой данных

|  |
| --- |
| class DatabaseHelper {  static final DatabaseHelper \_instance = DatabaseHelper.\_internal();  factory DatabaseHelper() => \_instance;  static Database? \_database;  DatabaseHelper.\_internal();  Future<Database> get database async {  if (\_database != null) {  return \_database!;  }  \_database = await initDatabase();  return \_database!;  }  Future<Database> initDatabase() async {  Directory documentsDirectory = await getApplicationDocumentsDirectory();  String path = join(documentsDirectory.path, 'realetate2.db');  bool exists = await databaseExists(path);  if (!exists) {  // Копирование файла базы данных из assets  ByteData data = await rootBundle.load('assets/realetate2.db');  List<int> bytes = data.buffer.asUint8List();  await File(path).writeAsBytes(bytes);  }  return await openDatabase(  path,  version: 1,  );  }  Future<int> insert(String table, Map<String, dynamic> data) async {  Database db = await database;  return await db.insert(table, data);  }  Future<int> insertUser(User user) async {  Database db = await database;  return await db.insert('users', user.toMap());  }  Future<List<Map<String, dynamic>>> query(String table) async {  Database db = await database;  return await db.query(table);  }  Future<bool> isEmailExists(String email) async {  Database db = await database;  List<Map<String, dynamic>> result = await db.query(  'Users',  where: 'Email = ?',  whereArgs: [email],  );  return result.isNotEmpty;  }  Future<bool> isPhoneExists(String phone) async {  Database db = await database;  List<Map<String, dynamic>> result = await db.query(  'Users',  where: 'Phone = ?',  whereArgs: [phone],  );  return result.isNotEmpty;  }  Future<bool> isPassportExists(String passport) async {  Database db = await database;  List<Map<String, dynamic>> result = await db.query(  'Users',  where: 'Passport = ?',  whereArgs: [passport],  );  return result.isNotEmpty;  }  Future<int> update(String table, Map<String, dynamic> data, String where,  List<dynamic> whereArgs) async {  Database db = await database;  return await db.update(table, data, where: where, whereArgs: whereArgs);  }  Future<int> delete(String table, String where,  List<dynamic> whereArgs) async {  Database db = await database;  return await db.delete(table, where: where, whereArgs: whereArgs);  }  Future<int?> getUserIdByEmail(String email) async {  Database db = await database;  List<Map<String, dynamic>> result = await db.query(  'Users',  columns: ['UserID'],  where: 'Email = ?',  whereArgs: [email],  );  if (result.isNotEmpty) {  return result.first['UserID'] as int;  } else {  return null;  }  }  Future<bool> isPropertyInFavorites(int userId, int propertyId) async {  Database db = await database;  List<Map<String, dynamic>> result = await db.query(  'Favorites',  where: 'UserId = ? AND PropertyId = ?',  whereArgs: [userId, propertyId],  );  return result.isNotEmpty;  }  Future<void> deleteReview(int reviewId) async {  try {  // Получаем экземпляр базы данных  Database db = await database;  // Выполняем запрос на удаление комментария  await db.delete(  'Reviews',  where: 'ReviewID = ?',  whereArgs: [reviewId],  );  } catch (e) {  // Обработка ошибок, если необходимо  print('Ошибка при удалении комментария: $e');  }  }  Future<bool> checkCredentials(String email, String password) async {  Database db = await database;  List<Map<String, dynamic>> result = await db.query(  'users',  where: 'Email = ? AND Password = ?',  whereArgs: [email, password],  );  return result.isNotEmpty;  } |