**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**ЛАБОРАТОРНАЯ РАБОТА**

Дисциплина: Программная инженерия

Студент: Шатохина В.С.

Группа: НФИбд-02-21

Студенческий билет № 1032217046

Москва

2023

Сервис бронирования жилья AirBnB

* **Airbnb** — это сервис, где путешественники арендуют места для проживания в 191 стране мира.

Пользователи могут арендовать или сдавать в аренду жилые помещения.

**Техническое задание**

1. Введение

Мобильное приложение AirBnB разрабатывается с целью предоставить пользователям удобный доступ к информации о недвижимости, а также возможность проведения поиска и оценки объектов. Основная цель приложения - обеспечить надежный и удобный сервис для пользователей, интересующихся арендой недвижимости.

2. Функциональные требования

2.1 Поиск недвижимости

* Пользователь должен иметь возможность осуществлять поиск недвижимости по различным критериям, таким как местоположение, тип объекта, количество комнат, площадь и цена.
* Поиск должен быть осуществлен с использованием фильтров, чтобы пользователь мог уточнить свои предпочтения.
* Результаты поиска должны быть представлены в удобной форме, с возможностью сортировки по релевантности, цене или другим параметрам.
* Каждый объект должен иметь подробную информацию, включая фотографии, описание, характеристики и контактные данные владельца недвижимости.
* Арендодатель должен иметь возможность публиковать и редактировать объявления о недвижимости со всеми критериями.

2.2 Оценка объекта

* Пользователь должен иметь возможность просмотреть оценку объекта, основанную на различных параметрах, таких как расстояние до магазинов, общественного транспорта, оценка безопасности района и т.д.
* Оценка должна быть представлена в виде рейтинга или числовой оценки, а также сопровождаться комментариями или отзывами.

2.3 Избранное

* Пользователь должен иметь возможность добавлять интересующие его объекты в избранное, чтобы легко находить их в будущем.
* Избранные объекты должны быть доступны пользователю даже после выхода из приложения или перезагрузки устройства.

2.4 Регистрация и профиль

* Пользователь должен иметь возможность зарегистрироваться в приложении, используя свою учетную запись или электронную почту.
* После регистрации пользователь должен иметь возможность создать профиль, указав свои предпочтения по недвижимости, а также настройки уведомлений.
* Пользователь должен иметь возможность войти в приложение, используя свои учетные данные.
* Арендодатель должен иметь возможность зарегистрироваться в приложении, используя свою учетную запись или электронную почту.
* Пользователь должен иметь возможность войти в приложение, используя свои учетные данные.

3. Архитектура и технологии

3.1 Платформа

* Мобильное приложение Циан будет разработано для операционной системы Android и iOS.

3.2 Язык программирования

* Для разработки мобильного приложения будет использоваться язык программирования Kotlin для Android и Swift для iOS.

3.3 Серверная инфраструктура

* Для обработки запросов пользователей и хранения данных приложения будет разработана и использована собственная серверная инфраструктура.
* Для хранения данных объектов недвижимости будет использоваться реляционная база данных.

3.4 Технические требования:

* Загрузка данных о недвижимости через API Циан.
* Использование базы данных для хранения информации о зарегистрированных пользователях и избранных объектах недвижимости.
* Внедрение уведомлений для информирования пользователей о новых объектах недвижимости и ответах от агентов.

4. Требования к дизайну и пользовательскому интерфейсу:

* Минималистичный и интуитивно понятный пользовательский интерфейс.
* Современный дизайн со средним количеством анимации и эффектов.
* Соответствие корпоративному стилю с использованием логотипа, цветовой гаммы и шрифтов.
* Разработка адаптивного дизайна для поддержки различных разрешений экранов мобильных устройств.

5. Тестирование

* Приложение должно проходить серию тестов на различные случаи использования, чтобы убедиться в его правильной работе и отсутствии ошибок.
* Тестирование должно включать как функциональное, так и нагрузочное тестирование.

6. Требования к безопасности:

* Защита пользовательских данных с использованием шифрования и безопасных протоколов.
* Защита от несанкционированного доступа к приложению и его функциональности.
* Защита от ошибок и исключений, включая обработку ошибок при соединении с сервером Циан.

7. Управление проектом

* Разработка приложения будет осуществляться в соответствии с гибкой методологией разработки (например, Scrum).
* Весь процесс разработки приложения должен быть задокументирован и подлежать обзору, чтобы обеспечить прозрачность и отслеживаемость разработки.

8. Сроки

* Анализ требований и составление спецификации - 2 недели.
* Разработка пользовательского интерфейса и дизайна - 3 недели.
* Разработка бэкэнда и интеграция с API AirBnB - 4 недели.
* Тестирование, отладка и оптимизация - 2 недели.
* Подготовка к запуску и выпуск в App Store и Google Play - 1 неделя.

Данное техническое задание описывает функциональные требования и основные аспекты разработки мобильного приложения AirBnB. В случае необходимости требования могут быть дополнены или изменены в процессе разработки.

**Архитектура**

Модель-представление-контроллер (Model-View-Controller, MVC):

* Модель (Model): Отвечает за бизнес-логику и обработку данных. Включает классы и структуры для управления данными, как например, объекты для представления недвижимости, фильтры, запросы на сервер и т.д.
* Представление (View): Отвечает за отображение данных и пользовательский интерфейс. Включает экраны, элементы управления, макеты, стили и т.д.
* Контроллер (Controller): Служит связующим звеном между моделью и представлением. Обрабатывает пользовательский ввод, обновляет модель и обновляет представление с учетом изменений в модели.

Разделение ответственности:

* Условия использования (Use Cases): Отвечают за выполнение конкретных действий, связанных с приложением. Например, поиск недвижимости, фильтрация результатов, просмотр деталей объекта и т.д.
* Сетевой слой: Обрабатывает взаимодействие с сервером AirBnB. Включает классы для запросов, обработки ответов, проверки соединения и управления кэшированием данных.

Архитектурные компоненты:

* База данных: Хранит локальные данные и кэшированную информацию для быстрого доступа. Может быть SQLite или Realm.
* API-клиент: Отвечает за взаимодействие с сервером посредством API. Включает методы для отправки запросов, обработки ответов и управления аутентификацией.
* Навигация: Управляет переходами и навигацией между экранами, включая навигационную панель, табы, модальные окна и др.

Использование паттернов проектирования:

* Одиночка (Singleton): Для доступа к общим ресурсам, таким как база данных и API-клиент.
* Фабрика (Factory): Для создания экземпляров классов, таких как объекты недвижимости и фильтры.
* Наблюдатель (Observer): Для обновления представления при изменении данных или состояния.

Сложность и масштабируемость:

* Разбиение на модули: Приложение может быть разделено на модули, связанные с различными функциональными областями (например, поиск, карта, профиль пользователя и т.д.), чтобы облегчить разработку и поддержку.
* Использование внешних библиотек: Для ускорения разработки и повышения надежности могут использоваться проверенные и стабильные сторонние библиотеки.

Тестирование:

* Unit-тестирование: Проверка отдельных компонентов приложения, таких как модели и классы API-клиента, с использованием фреймворков для тестирования, например, XCTest и JUnit.
* Интеграционное тестирование: Тестирование взаимодействия между компонентами приложения, включая проверку работы сетевого слоя и корректной обработки данных.

Это общая архитектура для мобильного приложения AirBnB. Конкретные детали архитектуры и выбор используемых технологий могут зависеть от требований проекта и предпочтений разработчиков.

**Диаграмма Use Case**

Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Изображение выглядит как текст, диаграмма, круг, снимок экрана

Автоматически созданное описание

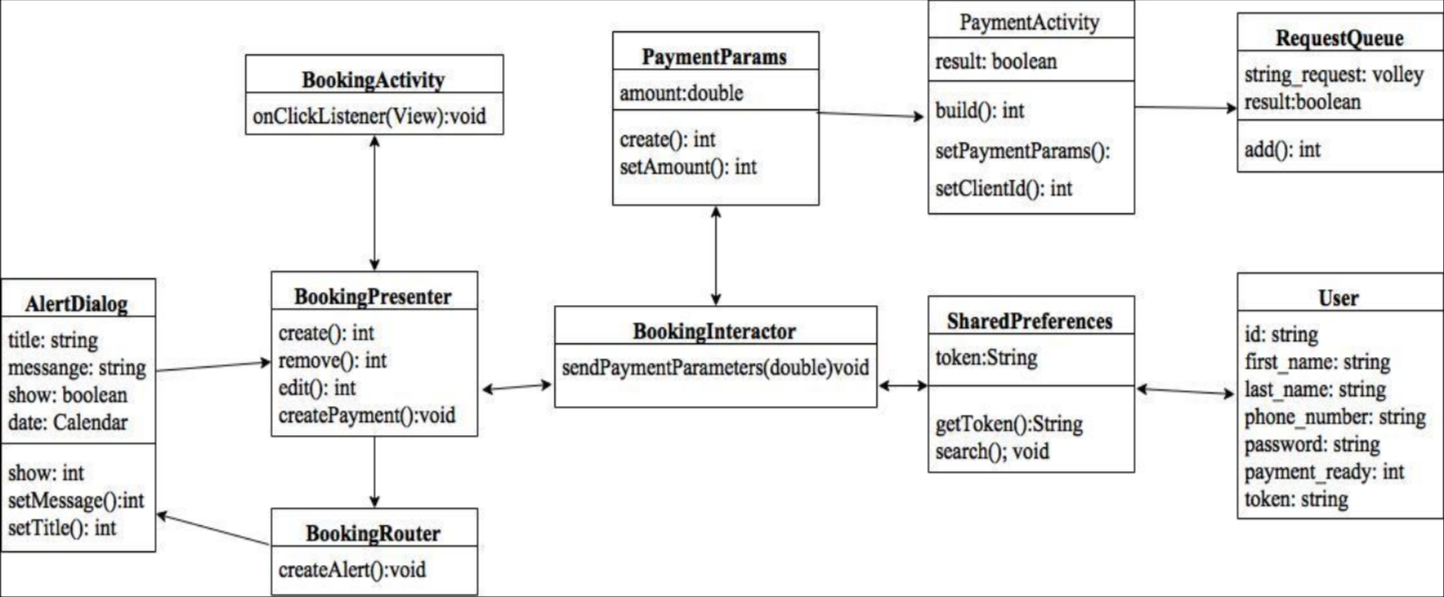
* Пользователь
* Агент недвижимости
* Администратор

Uses:

1. Поиск жилья: Пользователь выбирает критерии поиска (тип жилья, цена, кол-во комнат, город и т.д.). Система отображает список доступных вариантов жилья, соответствующих критериям. Пользователь выбирает желаемый вариант жилья. Система отображает подробную информацию о выбранном варианте жилья.
2. Бронирование жилья: Пользователь выбирает вариант жилья для бронирования. Пользователь вводит детали бронирования (даты, количество гостей и т.д.). Система проверяет доступность жилья в указанные даты. Если жилье доступно, система запрашивает подтверждение бронирования у пользователя. Пользователь подтверждает бронирование. Система создает бронь для выбранного жилья.
3. Управление объявлениями: Агент недвижимости создает новое объявление о жилье. Агент недвижимости вводит информацию о жилье (описание, фотографии, цена и т.д.). Система проверяет введенные данные на корректность. Если данные корректны, система сохраняет объявление и делает его доступным для поиска.
4. Управление агентами: Администратор создает нового агента недвижимости. Администратор вводит информацию о новом агенте (имя, контактная информация и т.д.). Система проверяет введенные данные на корректность. Если данные корректны, система сохраняет информацию агента.
5. Управление бронями: Агент недвижимости просматривает список активных броней для его объявлений. Агент недвижимости проверяет детали бронирования и связывается с клиентом для подтверждения бронирования. Агент недвижимости обновляет статус брони (подтверждена, отклонена и т.д.). Система обновляет статус брони и уведомляет пользователя о результате.

Диаграмма Классов

Определение программистом собственных классов объектов для конкретного набора задач должно позволить описывать отдельные задачи в терминах самого класса задач. При разработке программы должны быть определены классы используемых в программе объектов и построены их описания, затем созданы экземпляры этих объектов и определено взаимодействие между ними.



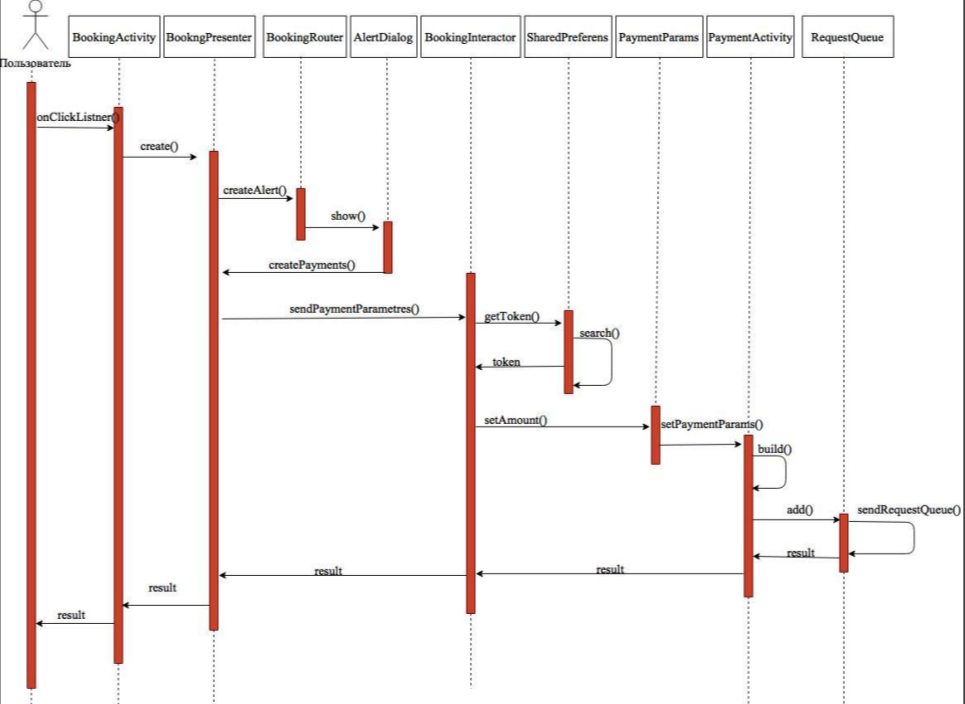
Классы:

* Класс User: представляет пользователя AirBnB, содержит информацию о имени, электронной почте, пароле, идентификаторе пользователя и другой персональной информации.
* Класс AlertDialog: формируется во время процесса выбора жилья для подтверждение бронирования.
* Класс PaymentParams: формируется при подтверждении, желания пользователя оплатить заказ выводит стоимость заказа для оплаты.
* Класс PaymentActivity: формируется для оплаты.
* Класс RequestQueue: создаётся для отправки информации о заказе на сервер.
* Класс SharedPreference: репозиторий приложения.
* Класс BookingInteractor: содержит в себе бизнес-логику процесса бронирования
* Класс BookingPresenter: определяет куда отдать управление.
* Класс BookingRouter: отвечает за переключение между различными экранами.
* Класс BookingActivity: содержит в себе окно выбора приложения.

**Диаграмма последовательности действий**

Диаграмма последовательности предназначена для представления динамики поведения объектов, отображая передачу сообщений между соответствующими классами.

Прецедент: Бронирование и оплата



Краткое описание:

Действующие лицо этого прецедента пользователь.

Основываясь на диаграмме классов, выберем объекты, задействованные в данной диаграмме:

* User.
* AlertDialog.
* PaymentParams.
* PaymentActivity.
* RequestQueue.
* SharedPreference.
* BookingInteractor.
* BookingPresenter
* BookingRouter.
* BookingActivity.

Прецедент начинается, когда пользователь хочет забронировать жильё:

* Объект «Пользователь», осуществляет регистрацию в приложение.
* «Пользователь» посылает объекту «User» сообщение «create», которое инициирует создание учетной записи пользователя.
* Объект «User» посылает «setMessage» и «setTitle» объекту «AlertDialog», что означает выбор пользователя для бронирования жилья.
* Объект «AlertDialog» посылает «setAmount» объекту «PaymentParams», содержащий в себе стоимость заказа.
* Объект «PaymentParams» посылает «setPaymentParams» и «setClientid» объекту «PaymentActivity» для произведения оплаты.
* Объект «PaymentActivity» объекту «RequestQueue» переменную «result» сообщающую о исполнение оплаты заказа.