МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет Компьютерных наук

Кафедра программирования и информационных технологий

Мобильное приложение “Домострой”: сервис для поиска и аренды оборудования и инструментов для ремонта и строительства на дому

Курсовая работа

Направление: 09.03.04. Программная инженерия

Зав. Кафедрой _____ д. ф.-м. н, доцент С.Д. Махортов

Руководитель _____ ст. преподаватель В.С. Тарасов

Руководитель практики _____ А.В. Москаленко

Обучающийся _____ М.Н. Андреева, 3 курс, д/о

Обучающийся _____ И.А. Караваева, 3 курс, д/о

Обучающийся _____ М.В. Мошкин, 3 курс, д/о

Обучающийся _____ И.В. Пустыльник, 3 курс, д/о

Обучающийся _____ А.А. Фетисова, 3 курс, д/о

Обучающийся _____ А.С. Шапор, 3 курс, д/о

Воронеж 2025

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Термины, используемые в данном документе описаны в таблице 1.

Таблица 1 - Термины, используемые в курсовом проекте

Термин	Значение
API	Интерфейс, предоставляемый программой для использования ее в другой программе.
Back-end	Часть программного обеспечения, отвечающая за обработку данных и представляющая собой серверное приложение.
Git	Распределенная система управления версиями, которая обеспечивает контроль изменений в коде, возможность ветвления и слияния кода.
GitHub	Платформа для хостинга проектов на базе Git, которая обеспечивает возможность хранения кода, управления задачами, рецензирования кода и совместной работы над проектами.
HTTP	Протокол передачи данных в сети Интернет, который используется для передачи информации между клиентом и сервером.
HTTPS	Защищенная версия протокола HTTP, использующая шифрование для безопасной передачи данных.
iOS	Мобильная операционная система для смартфонов, электронных планшетов, носимых проигрывателей, разрабатываемая и выпускаемая американской компанией Apple.

Продолжение таблицы 1

Термин	Значение
Java	Строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems.
LLM (Large Language Model)	Большая языковая модель, тип программы искусственного интеллекта, которая может распознавать и генерировать текст
PostgreSQL	Объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом.
REST API	Архитектурный стиль взаимодействия между клиентом и сервером через HTTP.
UIKit	Среда разработки приложений и набор инструментов для создания графического интерфейса пользователя от Apple Inc., используемый для создания приложений для операционных систем iOS, iPadOS и tvOS.
Арендатор	Физическое или юридическое лицо, берущее во временное владение и пользование (либо только пользование) имущество другого лица.
Арендодатель	Физическое или юридическое лицо, которое владеет имуществом или другими ресурсами и сдаёт их в аренду другим лицам или организациям на условиях, определенных в договоре аренды.
Аутентификация	Процесс проверки подлинности личности или учетных данных пользователя для подтверждения его идентичности.

Продолжение таблицы 1

Авторизованный пользователь	Пользователь, который прошел процедуру аутентификации для доступа к определенным ресурсам, функциям или услугам в рамках системы или приложения.
Неавторизованный пользователь	Пользователь, который не прошел процедуру аутентификации или идентификации при доступе к ресурсам, функциям или услугам в рамках системы или приложения.
Система управления базами данных (СУБД)	Совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.
Таск-менеджер	Специальное программное обеспечение или онлайн-сервис, предназначенный для управления задачами и проектами.
Токен аутентификации	Специальный текстовый код, используемый для подтверждения легитимности и подлинности пользователя при доступе к определённым ресурсам или сервисам.
Эндпоинт	Конечная точка в API, к которой можно обратиться для выполнения нужного действия или получения данных.

ВВЕДЕНИЕ

Приобретение необходимого строительного оборудования является дорогостоящим и не всегда целесообразным в связи с его разнообразием и применимостью для узкого списка задач. Решение данной проблемы заключается в поиске необходимого оборудования и аренды его на срок, необходимый для проведения работ. Внедрение цифровых технологий может служить для ускорения и облегчения данного процесса.

Традиционные способы аренды, такие как обращение в прокатные пункты или строительные магазины, нередко сопряжены с ограниченным выбором, недостатком информации об оборудовании, а также неудобным процессом бронирования. Кроме того, для большинства пользователей важно не только получить нужный инструмент, но и сделать это быстро, с возможностью выбора ближайшего пункта аренды и уточнения всех деталей онлайн.

Создание цифровой платформы, ориентированной на аренду строительного оборудования, позволит объединить арендодателей и потенциальных арендаторов, предложив им удобный, безопасный и прозрачный способ взаимодействия. Такой подход отвечает современным тенденциям цифровизации, устойчивого потребления и шеринговой экономики.

Данная работа направлена на разработку мобильного приложения и серверной инфраструктуры, основанных на принципах модульности, масштабируемости и удобства использования, с целью создания эффективного инструмента для аренды оборудования, способствующего снижению затрат и времени на проведение ремонтных и строительных работ в домашних условиях.

1 Постановка задачи

Цель курсовой работы – разработка мобильного приложения для поиска и аренды оборудования и инструментов для ремонта и строительства на дому. Мобильное приложение должно быть написано для операционной системы iOS версии 17 и выше. Серверная часть должна быть основана на микросервисной архитектуре.

1.1 Цели проекта

Целью курсового проекта является создание системы, которая должна обеспечивать выполнение следующих целей:

– Формирование релевантной и активной группы пользователей численностью не менее 30 человек, каждый из которых оставит отклик как минимум на 1 товар с использованием разрабатываемой системы;

– Создание пользовательского интерфейса для поиска и аренды оборудования и инструментов для ремонта и строительства с интуитивной навигацией, качество которого должно составлять не менее 7 по десятибалльной шкале по результатам опроса тестовой группы пользователей численностью не менее 30 человек. Опрос должен быть проведен командой исполнителей по завершении проекта.

1.2 Задачи приложения

Задачами мобильного приложение является предоставление пользователям следующих возможностей:

– Создание и публикация объявлений об аренде оборудования и инструментов для ремонта;

– Просмотр и поиск объявлений, опубликованных в системе;

– Создание откликов на объявления, созданные другими пользователями;

– Ответ на отклики, оставленные другими пользователями, на объявления, принадлежащие пользователю.

1.3 Функциональные требования

Пользователи разделены на следующие группы:

– Неавторизованный пользователь;

– Авторизованный пользователь;

– Администратор.

Каждой группе система должна позволять решить определенные задачи. Диаграмма представлена на рисунке ААА.

Список функций, которые система должна предоставлять неавторизованному пользователю:

– Регистрация;

– Авторизация;

– Просмотр объявлений;

– Просмотр профиля пользователей;

– Наложение фильтров при поиске объявлений;

– Применение сортировки при поиске объявлений;

– Просмотр доступных дат при просмотре объявления.

Авторизованному пользователю должны быть доступны все функции, доступные неавторизованному пользователю, за исключением авторизации и регистрации, а также следующие функции:

– Просмотр информации о своем профиле;

– Обновление информации о своем профиле;

– Просмотр объявлений;

– Просмотр профиля пользователей;

– Создание и публикация объявлений;
– Редактирование своих объявлений;
– Удаление своих объявлений;
– Просмотр списка своих объявлений;
– Отклик на объявления;
– Выбор дат при оформлении отклика;
– Получение уведомлений на электронную почту при откликах;
– Настройка уведомлений на почту;
– Редактирование списка избранных объявлений;
– Просмотр списка избранных объявлений;
– Обработка откликов.

Администратору должны быть доступны все функции, доступные авторизованному пользователю, а также следующие функции:

– Блокировка объявлений;
– Удаление объявлений;
– Блокировка пользователей;
– Удаление пользователей;
– Поиск пользователей с фильтрацией.

1.4 Задачи, решаемые в процессе разработки

Разработка системы включает следующие задачи:

– Разделение задач между участниками команды;
– Анализ предметной области;
– Исследование аналогов;
– Написание и согласование технического задания;
– Проектирование структуры системы;
– Создание UML-диаграмм и схемы базы данных;
– Проектирование дизайна мобильного приложение;

– Проектирование схемы API backend-приложения;
– Создание системы в соответствии с техническим заданием;
– Анализ перспектив развития системы.

2 Анализ существующих решений

В данной главе будут рассмотрены решения, которые позволяют решить все задачи или же часть задач, для выполнения которых создается описываемая в работе система.

Аналоги предлагается разделить на 2 категории:

– Сервисы, позволяющие брать и сдавать предметы в аренду, в том числе (или исключительно) инструменты и технику для ремонта;

– Сервисы, представляющие собой магазины аренды.

Стоит отдельно отметить, что в процессе разработки рассматривались также сервисы «Auto.ru» и «Циан». Они предоставляют возможность брать и сдавать в аренду товары иного рода, а потому аналогами или конкурентами считаться не могут и в данной работе описаны не будут.

2.1 Сервисы аренды

В таблице 2 представлено краткое сравнение сервисов аренды.

Таблица 2 - Сравнение сервисов аренды

Параметр сравнения	Авито	YouTool	Polka
Поиск и фильтрация при поиске	Есть	Есть	Есть
Профиль пользователя	Есть	Есть	Есть
Профиль арендодателя	Есть	Есть	Есть
Описание объявления	Есть	Есть	Есть
Фото в объявлении	Есть	Есть	Есть

Продолжение таблицы 2

Уведомления	Есть	Есть	Нет
Раздел «Избранное»	Есть	Есть	Нет
Отображение доступных дат аренды	Есть	Нет	Нет

2.1.1 Авито

Авито - российский интернет-сервис для размещения объявлений о товарах, недвижимости, вакансиях и резюме на рынке труда, а также услугах. Данный сервис широко известен на территории РФ и активно применяется жителями всех регионов.

На рисунке 1 представлена титульная страница мобильного приложения Авито.

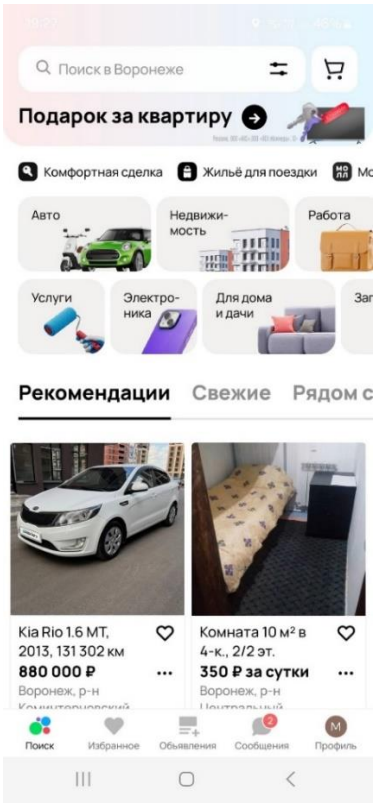

Рисунок 1 – Титульная страница мобильного приложения Авито

У данного решения можно выделить следующие преимущества:

– Наиболее широкий ассортимент товаров;
– Гибкие настройки объявлений;
– Ведение деятельности на всей территории РФ;
– Встроенная система чатов и звонков;
– Высокая устойчивость системы;
– Возможность доставки через сервис;
– Удобный интерфейс приложения.

В противовес можно выделить недостатки:

– Отсутствие специализации на аренде строительных инструментов;
– Отсутствие гарантий со стороны сервиса;
– Отсутствие консультации в сервисе.

2.1.2 YouTool

YouTool — это мобильный агрегатор для аренды строительных инструментов и спецтехники.

На рисунке 2 представлена титульная страница мобильного приложения YouTool.

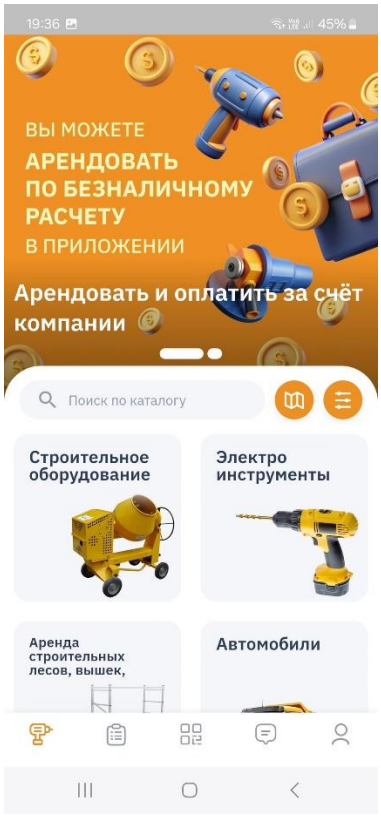

Рисунок 2 – Титульная страница мобильного приложения YouTool

У данного решения можно выделить следующие преимущества:

– Специализация на аренде строительного оборудования;

– Возможность выбора сроков аренды и расчёта стоимости в самом приложении;

– Юридическое оформление условий аренды.

В противовес можно выделить недостатки:

– Сравнительно небольшая база пользователей в сравнении с Авито;

– Необходимость документального подтверждения личности для работы с платформой;

– Низкая надежность сервиса, выражающаяся в долгих загрузках.

2.1.3 Polka

Polka — это российский онлайн-сервис аренды вещей, функционирующий как маркетплейс, где пользователи могут сдавать и брать

в аренду различные предметы, включая строительные инструменты, технику, спортинвентарь и многое другое.

На рисунке 3 представлена титульная страница сайта Polka.

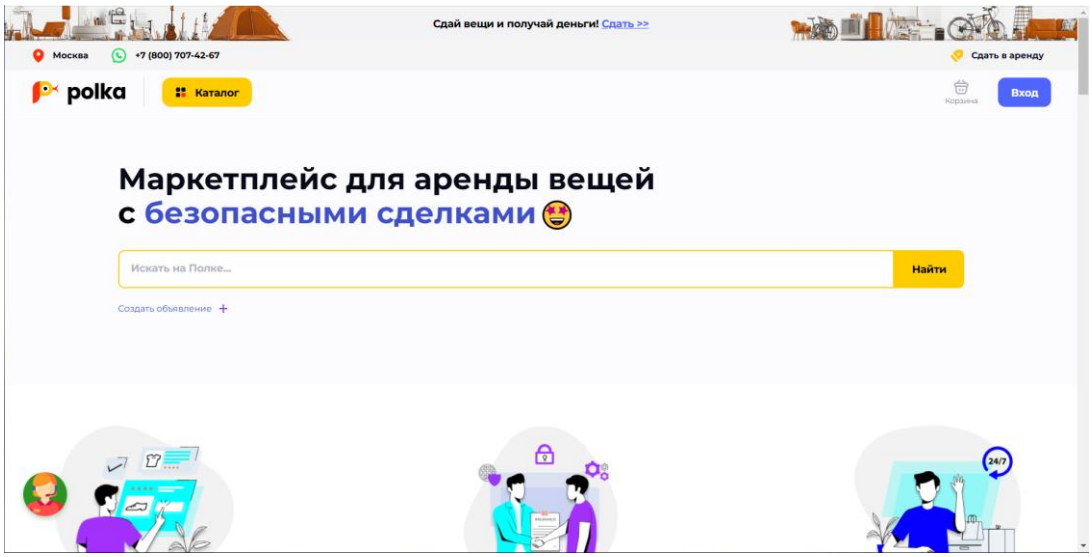

Рисунок 3 – Титульная страница сайта Polka

У данного решения можно выделить следующие преимущества:

– Специализация на аренде строительного оборудования;

– Возможность полного цикла оформления аренды через сайт;

– Возможность доставки.

В противовес можно выделить недостатки:

– Сравнительно небольшая база пользователей в сравнении с Авито;

– Необходимость документального подтверждения личности для работы с платформой;

– Ограниченное географическое покрытие: сервис может быть доступен не во всех регионах РФ.

2.2 Магазины аренды

В таблице 3 представлено краткое сравнение магазинов аренды.

Таблица 3 - Сравнение магазинов аренды

Параметр сравнения	Прокат36	ВиРент
Поиск и фильтрация при поиске	Есть	Есть
Профиль пользователя	Есть	Есть
Профиль арендодателя	Нет (сам магазин)	Нет (сам магазин)
Описание объявления	Есть	Есть
Фото в объявлении	Есть	Есть
Уведомления	Есть	Есть
Раздел «Избранное»	Нет	Есть
Отображение доступных дат аренды	Нет	Нет

2.2.1 Прокат36

Прокат36 представляет собой компанию по аренде инструментов, работающий на территории города Воронеж. Помимо деятельности, связанной с арендой оборудования, предоставляет услуги по ремонту и обслуживанию строительных инструментов.

На рисунке 4 представлена титульная страница сайта Прокат36.

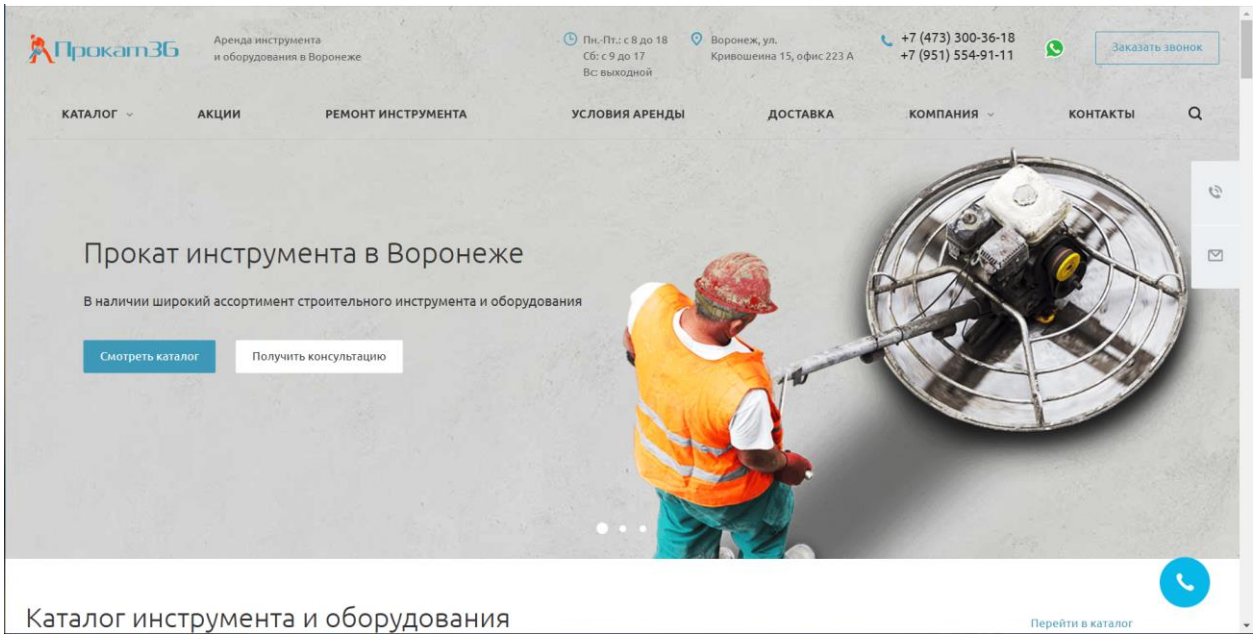

Рисунок 4 – Титульная страница сайта Прокат36

У данного решения можно выделить следующие преимущества:

– Сравнительно низкие цены аренды;

– Возможность сопутствующего ремонта оборудования;

– Предоставляемая консультация;

– Скидочные программы.

В противовес можно выделить недостатки:

– Относительно небольшой ассортимент оборудования;

– Оформление аренды требует обсуждения с сотрудниками компании с этапа выбора дат.

2.2.2 ВиРент

Компания ВиРент.ру входит в группу компаний ВсеИнструменты.ру. Сервис, предоставляемый данной компанией, служит для поиска и аренды строительных инструментов. Благодаря своему ведущему положению на рынке, фирма обеспечивает широкий ассортимент товаров и гарантирует его

качество. Сложности доставки и ремонта оборудования также должны решаться компанией.

На рисунке 5 представлена титульная страница сайта ВиРент.

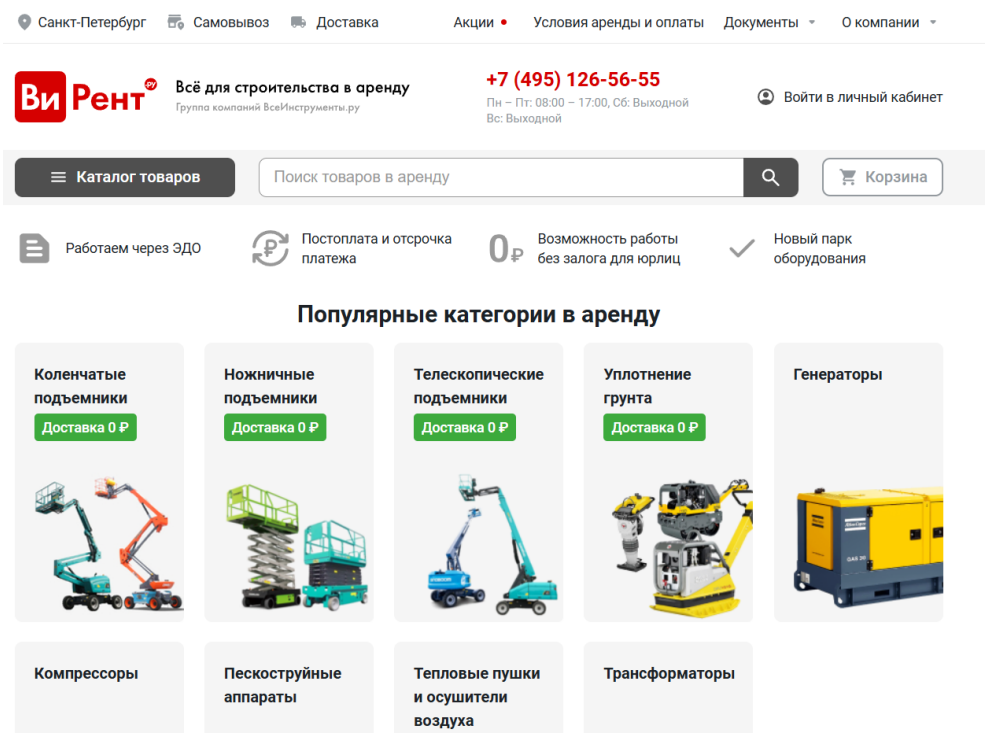

Рисунок 5 – Титульная страница сайта ВиРент

У данного решения можно выделить следующие преимущества:

– Широкий ассортимент товаров;

– Быстрая доставка;

– Возможность консультации;

– Обслуживание оборудование на месте;

– Ширина распространения (работа на территории всей РФ);

– Гибкие сроки аренды.

В противовес можно выделить недостатки:

– Более высокая стоимость оборудования;

– Штрафные санкции за просрочку использования и оплаты;

– Необходимость внесения залоги при оформлении некоторых видов товаров.

2.3 Вывод по анализу существующих решений

В процессе анализа было выявлено следующее: каждая платформа предоставляет либо широкий выбор за счет участия пользователей приложения в формировании ассортимента, либо гарантирует качество и безопасность аренды. Совмещение двух названных подходов может позволить системе конкурировать с имеющимися на рынке игроками. Стоит отметить, что реализуемая в ходе проекта система представляет собой MVP продукта, а потому не ставит перед собой цели реализовать данных подход. Тем не менее, выдвинутая гипотеза может стать фактором успеха при развитии системы.

Более практическим выводом является список элементов и функций, которые присутствуют во всех или большинстве проанализированных приложений, среди которых находятся:

- Поиск и фильтрация при поиске;
- Профиль пользователя;
- Профиль арендодателя;
- Описание объявления;
- Фото в объявлении;
- Уведомления;
- Раздел «избранное»;
- Отображение доступных дат аренды.

При этом предлагается избегать следующих ошибок:

- Запрос слишком большого числа документов на раннем этапе;
- Отсутствие уведомлений;
- Отсутствие наглядного представления доступных для аренды дат.

3 Анализ целевой аудитории

Целевой аудиторией являются лица, которые осуществляют самостоятельный ремонт или строительство дома или граждане, оказывающие услуги по ремонту и строительству.

Группа лиц, осуществляющих ремонт своего жилья, более многочисленна, но при этом активность ее участников значительно ниже, так как связано с эпизодической деятельностью, а не ведущей. В свою очередь, группа граждан, оказывающих услуги по строительству и ремонту сравнительно малочисленна, но более активна в связи с тем, что ее деятельность постоянна. Стоит отдельно отметить, что представителями данной группы будут более молодые представители профессии, так как со временем в их собственности появляются необходимые для работы инструменты.

4 Реализация

4.1 Способ ведения проекта

Разработка системы основывалась на использовании гибкой методологии Kanban. Данная методология подходит для команд, участники которых обладают узкоспециализированными навыками и выполняют задачи в пределах своей компетенции. Процесс разработки был поделен на равные отрезки времени – спринты, в течение которых перед каждым участником команды стояли заранее обговоренные и выставленные по приоритету задачи. Реализации проекта была организована следующим образом:

– Распределение ролей внутри команды – каждому из участников команды была отведена роль в соответствие с имеющимися у него навыками;

– Обсуждение и формирование требований к системе – был обговорен и сформирован список функциональных и нефункциональных требований к системе. Все требования были разделены на 4 категории в соответствии с методом приоритизации MoSCoW;

– Создание и оформление командного таск-трекера – на платформе «Yougile» создан проект «Домострой», на досках которого размещаются карточки с задачами всех участников команды. В соответствие с выбранной методологией ведения проекта основная доска разделена на колонки, отражающие статус задач и этап, на котором они находятся в данный момент в текущий момент времени;

– Формирование потока задач – на регулярных встречах команды перед началом каждого спринта определяются задачи для каждого из участников команды, которые он должен будет выполнять в рамках отведенного времени. Все задачи размещаются в общем таск-трекере команды. Каждый участник выбирает задачи, соответствующие его

специализации и зоне ответственности и выполняет их в соответствие с указанной приоритетностью;

– Гибкое управление приоритетами – в случае изменения требований заказчика, проводится встреча внутри команды, на которой происходит переопределение приоритетов задач в соответствие с новыми требованиями. Встречи такого типа проводятся оперативно без ожидания завершения текущего спринта;

– Регулярные обзоры прогресса – каждую неделю команда проводит оперативные встречи внутри команды. Таким образом, в течение каждого спринта проводятся 2 встречи. Первая – в начале спринта – для обсуждения проблем и итогов предыдущего цикла, а также определения задач и их приоритетности на текущий отрезок времени. Вторая встреча происходит в середине спринта, когда требуется синхронизировать текущий прогресс по задачам и выявить барьеры на пути к их успешной реализации в рамках текущего этапа.

– Регулярная отчетность – для обеспечения прозрачности процесса разработки и информирования заказчика о ходе работы команда регулярно предоставляет отчеты по итогам каждого спринта.

Процесс разработки был поделен на две крупные стадии. В рамках первой было сделано следующее:

– Проведен анализ предметной области и собрана необходимая информация для предпроектного исследования;

– Проведен анализ рынка и выявлена целевая аудитория разрабатываемой системы

– Составлена необходимая документация и спроектированы основные диаграммы системы;

– Созданы макеты для основных экранов приложения и оформлена единая дизайн-система для обеспечения визуальной целостности проекта;

– Разработана основная часть сервиса “Core”;

– Разработан и протестирован сервис “Authorization”;

– Разработан и протестирован сервис “Gateway”;

– Разработана основная часть мобильного приложения;

– Проведено первичное тестирование приложения и обнаружены баги, требующие доработки;

– Разработана и протестирована База данных.

– В рамках второй стадии было сделано следующее:

– Скорректирован способ решения задач: возросла частота командных встреч, а также разработчики начали проводить отдельные технические обсуждения;

– Созданы макеты для всех экранов приложения;

– Завершена разработка сервисов, созданных во период первой стадии;

– Разработан сервис “Notification”;

– Интегрирована нейронная сеть

– Завершена разработка мобильного приложения;

– Проведено тестирование приложения;

– Составлена ПМИ.

4.2 Средства реализации

4.2.1 Серверная часть

Для реализации серверной части системы были выбраны следующие технологии:

– Язык программирования Java версии 21;

– Фреймворк Spring Boot версии 3.4.4, с модулями Spring Security, Spring Gateway, Spring Mail;

– Система управления базами данных PostgreSQL;

– Объектное хранилище Yandex Object Storage;

– Платформа контейнеризации Docker;
– Система управления миграциями базы данных Liquibase;
– Брокер сообщений RabbitMQ;
– Хранилище Docker образов Yandex Cloud Registry;
– Система автоматизации задач GitHub actions на платформе GitHub;
– Платформа Yandex Cloud Compute.

Преимуществами языка программирования Java являются его платформенная независимость, стабильность, а также развитая экосистема библиотек и инструментов. Версия Java 21 предоставляет улучшения производительности, расширенные возможности работы с шаблонами, современный синтаксис и поддержку виртуальных потоков, что позволяет создавать более эффективные и масштабируемые серверные приложения.

Spring Boot значительно упрощает создание полнофункциональных приложений на базе Spring Framework, предоставляя разработчику готовую инфраструктуру и автоматическую конфигурацию. Использование модулей Spring Security, Spring Gateway и Spring Mail позволяет реализовать централизованную безопасность, маршрутизацию и обработку email-уведомлений.

Преимуществами PostgreSQL являются его высокая производительность, надежность и поддержка сложных запросов. Благодаря широкому набору встроенных функций, поддержке расширяемости и соответствию стандартам SQL, PostgreSQL эффективно справляется с хранением и обработкой больших объемов данных, обеспечивая стабильную работу приложения в условиях высокой нагрузки.

Yandex Object Storage предоставляет безопасное, масштабируемое и отказоустойчивое решение для хранения файлов. Его совместимость с S3-протоколом позволяет легко интегрироваться с существующими инструментами. Преимуществом является возможность хранения и быстрой доставки большого количества медиафайлов, таких как изображения и документы, загружаемые пользователями.

Docker предоставляет возможность упаковывать приложения и их зависимости в изолированные контейнеры. Это обеспечивает высокую переносимость, быструю настройку среды и упрощает процесс развертывания.

Liquibase обеспечивает безопасное и контролируемое внесение изменений в структуру базы данных. Преимуществами являются автоматизация процесса миграции, отслеживание версий схемы и минимизация человеческого фактора.

RabbitMQ позволяет реализовать надёжную и масштабируемую асинхронную коммуникацию между микросервисами.

Yandex Cloud Registry предоставляет удобный способ облачного хранения созданных образов для Docker.

GitHub actions - сервис для автоматизации процессов CI/CD, встроенный в платформу GitHub. Он позволяет настраивать и запускать рабочие процессы для сборки, тестирования и деплоймента приложений при изменениях в репозитории.

Yandex Cloud Compute предоставляет возможность создавать виртуальные машины и закреплять за ними публичный IP-адрес, что позволяет маршрутизировать запросы к ней, а следовательно подходит для развертывания серверной части.

4.2.2 Клиентская часть

Для реализации мобильного приложения были выбраны следующие технологии:

– Язык программирования Swift версии 5.10;

– Фреймворк для создания пользовательского интерфейса UIKit;

– Библиотека для верстки SnapKit версии 8.7.1;

– Библиотека для работы с табличными и коллекционными представлениями ReactiveDataDisplayManager версии 7.4.0;

– Библиотека для работы с сетью NodeKit версии 5.0.2;

– Библиотека для асинхронной загрузки изображений Kingfisher версии 8.3.1;

– Библиотека для работы с календарем HorizonCalendar;

– Библиотека для отображения временных уведомлений Drops версии 1.7.0;

– Утилита для кодогенерации Generamba;

– Утилита для кодогенерации SwiftGen версии 6.6.3.

Swift представляет собой открытый мультипарадигмальный компилируемый язык программирования общего назначения, разработанный и поддерживаемый компанией Apple.

UIKit представляет собой среду разработки приложений и набор инструментов для создания графического интерфейса пользователя от Apple Inc., используемых для создания приложений для операционных систем iOS, iPadOS и tvOS.

SnapKit — это библиотека для упрощённого создания и управления автолэйаутами с помощью декларативного кода на Swift. Она позволяет создавать адаптивные и гибкие интерфейсы с меньшим количеством кода по сравнению с использованием стандартных NSLayoutConstraint.

ReactiveDataDisplayManager — библиотека, обеспечивающая удобное и эффективное управление таблицами и коллекциями с использованием реактивного подхода, что улучшает читаемость и масштабируемость кода при работе с данными и их отображением.

NodeKit — это сетевой слой, предоставляющий простые и расширяемые инструменты для организации сетевых запросов, обработки ответов и управления асинхронными операциями в приложении.

Kingfisher — библиотека для асинхронной загрузки, кэширования и отображения изображений, обеспечивающая высокую производительность и минимальные задержки при работе с графическим контентом.

HorizonCalendar — мощная и гибкая библиотека для работы с календарём, позволяющая легко создавать и настраивать календарные компоненты с поддержкой различных режимов отображения.

Drops — библиотека для отображения временных, неинвазивных уведомлений и подсказок, которые не прерывают взаимодействие пользователя с приложением.

Generamba — утилита для автоматической генерации кода по шаблонам, которая помогает стандартизировать структуру проекта и ускорить разработку.

SwiftGen — инструмент для автоматической генерации безопасных и удобных в использовании констант и ресурсов, таких как строки, цвета, изображения, что уменьшает вероятность ошибок при работе с ресурсами проекта.

4.3 Архитектура системы

Архитектура системы представлена на рисунке 6.

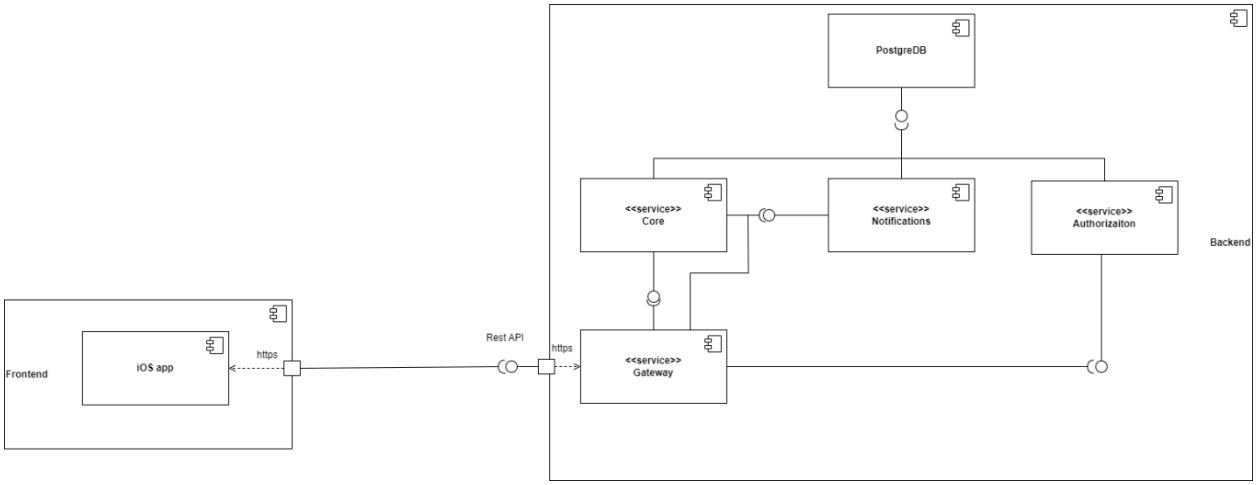

Рисунок 6 – Архитектура системы

Система реализует классическую трехзвенную архитектуру, то есть состоит из следующих частей:

– Клиентское приложение;
– Серверное приложение;
– База данных.

4.4 Серверная часть

Сервер предоставляет REST API для использование клиентской стороной.

4.4.1 Общая структура

Серверная часть состоит из базы данных и 4 микросервисов:
– “Gateway” для управления запросами;
– “Authentication” для авторизации, регистрации и проверки токена аутентификации;
– “Core” как сервис, реализующий основной функционал приложения;
– “Notifications” для отправки уведомлений.
Схема взаимодействия микросервисов представлена на рисунке 7.

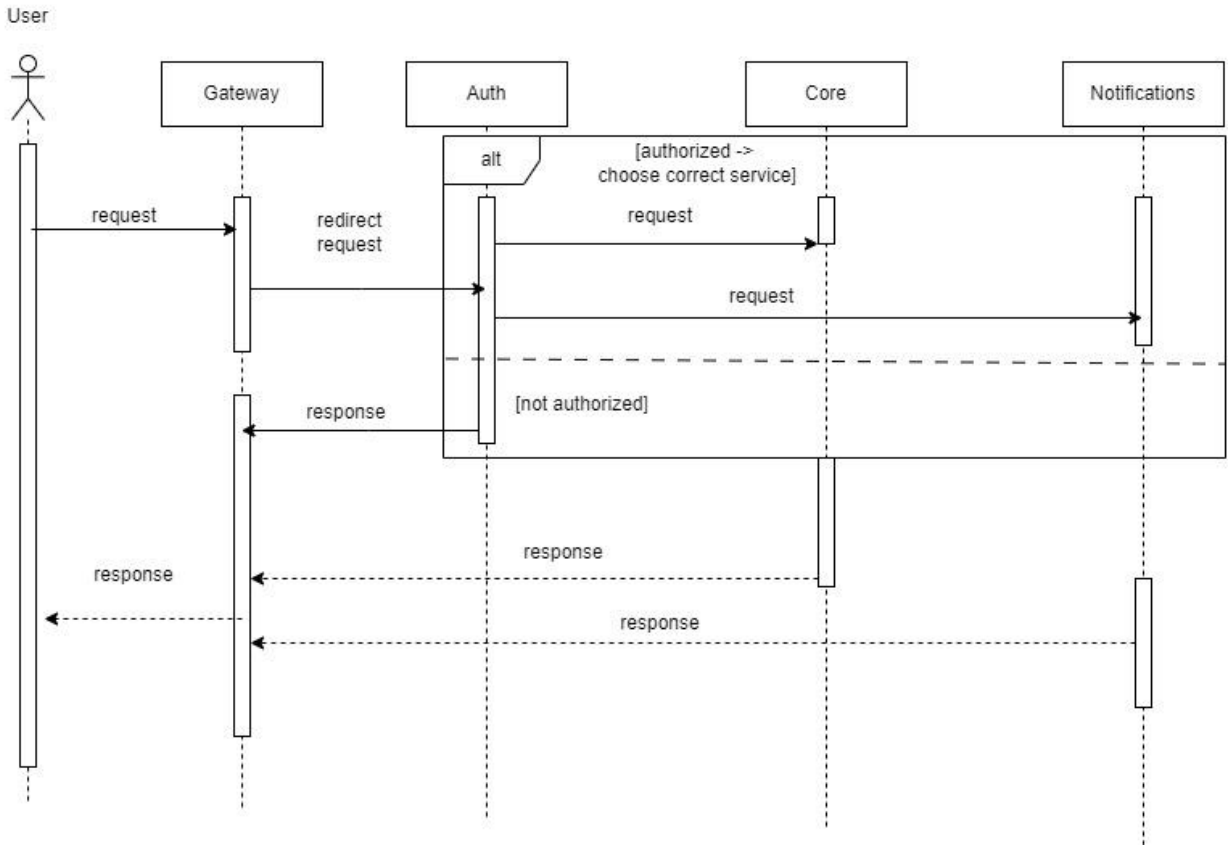

Рисунок 7 – Схема взаимодействия микросервисов

Далее будут описаны функции, выполняемые каждым микросервисом.

4.4.2 База данных

База данных используется для хранения информации отдельно от бизнес-логики, предоставления общего доступа разным элементам и ее безопасного сохранения данных.

Схема базы данных представлена на рисунке 8.

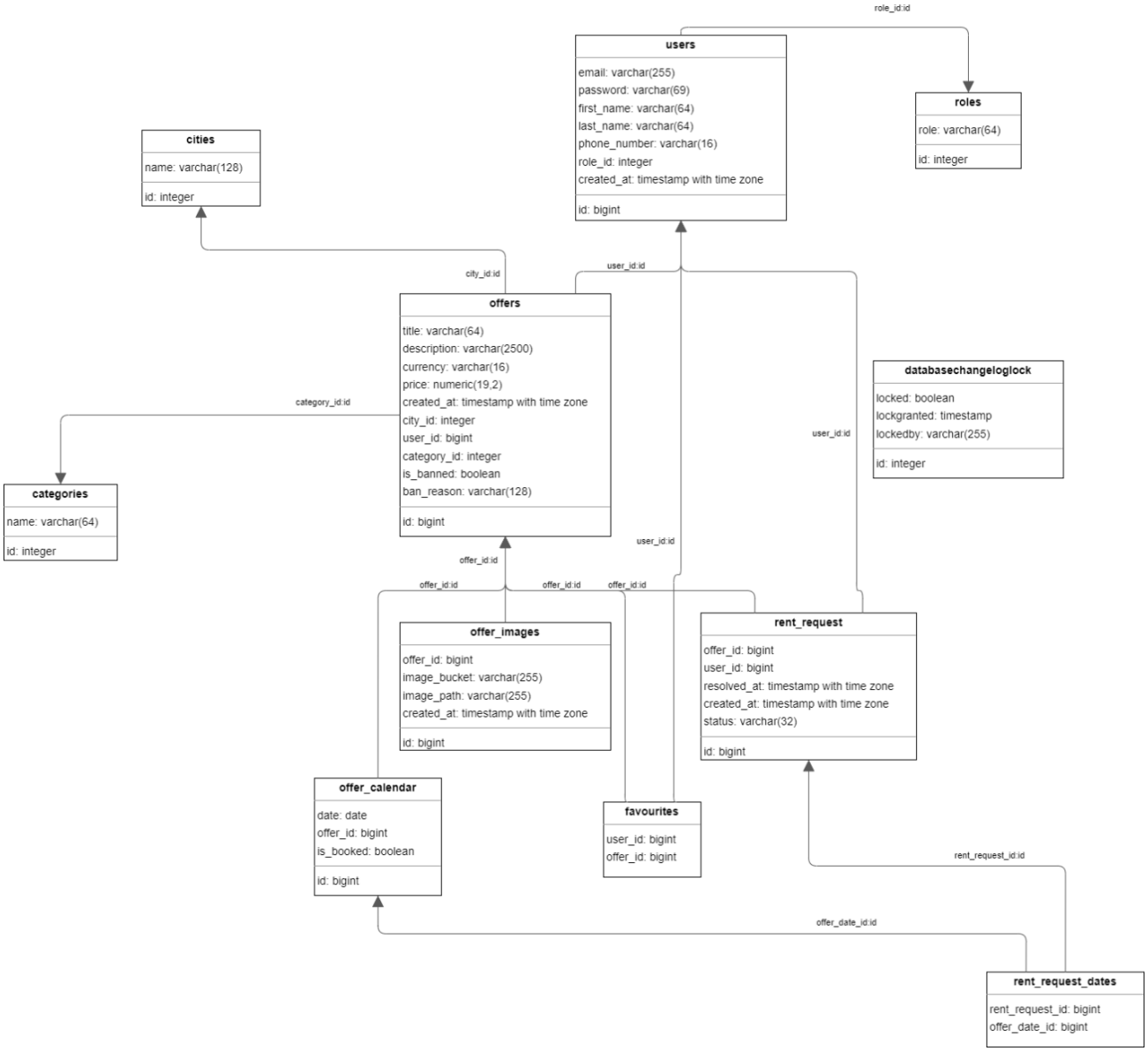

Рисунок 8 – Схема базы данных

Приведенная база данных находится в 3 нормальной форме, что позволяет избежать ошибок при вставке, удалении или обновлении данных.

4.4.3 Микросервис “Gateway”

Данный микросервис служит для управления запросами. Он представляет собой единую точку доступа, основной функцией которой является маршрутизация, то есть перенаправление входящих HTTP-запросов к соответствующим микросервисам.

4.4.4 Микросервис “Authorization”

Данный микросервис отвечает за управление процессами аутентификации и авторизации пользователей и выполняет следующие функции:

– Проверка учетных данных пользователя (адрес электронной почты и пароль) при входе в систему. Для успешной авторизации адрес электронной почты и пароль, указанные пользователем, должны совпадать с соответствующими на сервере;

– Генерирование и предоставление JWT-токена. Токен должен быть предоставлен пользователю после успешной авторизации;

– Регистрация пользователей. Для успешной регистрации адрес электронной почты, указанный пользователем, не должен быть закреплен за другим пользователем.

4.4.5 Микросервис “Core”

Данный микросервис отвечает за предоставление основной функциональности системы.

Для неавторизованного пользователя:

– Получение информации о пользователе. Должны быть доступны имя пользователя, контактная информация и список открытых объявлений данного пользователя. В случае, если пользователь заблокирован, должна отображаться только информация о том, что пользователь заблокирован;

– Получение информации об объявлении. Информация должна предоставляться только для опубликованных объявлений;

– Получение информации о списке объявлений с параметрами фильтрации и сортировки;

Для авторизованного пользователя:

– Обновление информации о пользователе. Обновление электронной почты должно быть невозможно. Для обновления пароля должен быть корректно введен текущий пароль;

– Обновление информации об объявлении. Название объявления не должно быть пустым;

– Создание объявлений. Название объявления не должно быть пустым. При создании объявления должна проводиться автоматическая модерация. Объявления, название или описание которых содержат нецензурные выражения, должны быть автоматически заблокированы. Если название или описание объявления содержат ссылки, то эти ссылки должны быть удалены;

– Создание откликов;

– Получение откликов;

Для администратора:

– Блокировка объявления;

– Разблокировка объявлений;

– Удаление объявления;

– Блокировка пользователей;

– Разблокировка пользователей;

– Удаление пользователей;

– Получение списка пользователей с фильтрацией по совпадению текста. Под совпадением понимается вхождение введенного текста как подстроки в имя пользователя или его электронную почту.

4.4.6 Микросервис “Notifications”

Данная подсистема отвечает за управление уведомлениями и реализует следующие функции:

– Настройка уведомлений на почту;
– Отправление уведомлений на почту.

4.4.7 Использование нейронной сети

В проекте используется нейронная сеть типа LLM для автоматической модерации объявлений. Обращение к ней происходит из микросервиса “Core”. Схема взаимодействия представлена на рисунке 9.

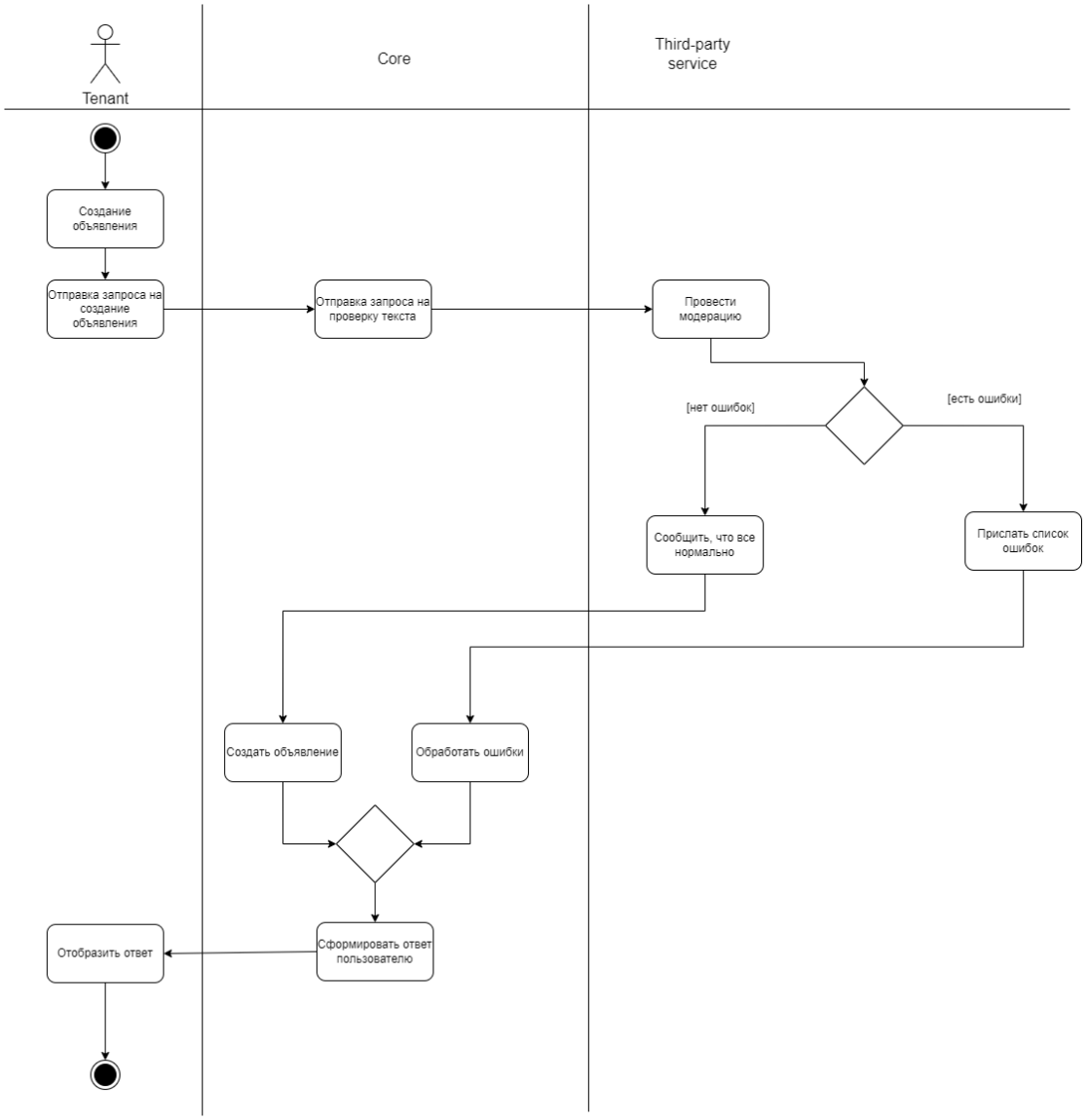

Рисунок 9 – Схема использования LLM

В данном проекте используется сторонняя нейронная сеть и обращение к ней происходит посредством API, предоставляемого openrouter.ai.

4.4.8 Развертывание

Для реализации развертывания серверной части приложения были выполнены следующие шаги:

– На платформе Yandex Cloud Compute была создана виртуальная машина со статическим IP-адресом, на которой запускаются все микросервисы;

– Написаны Docker файлы для создания образов микросервисов;

– Написан Docker-compose файл для автоматического управления всеми элементами системы;

– Написан скрипт на GithubActions для автоматической отправки изменений на виртуальную машину при изменении в github репозитории.

Диаграмма развертывания для разрабатываемой системы представлена на рисунке 10.

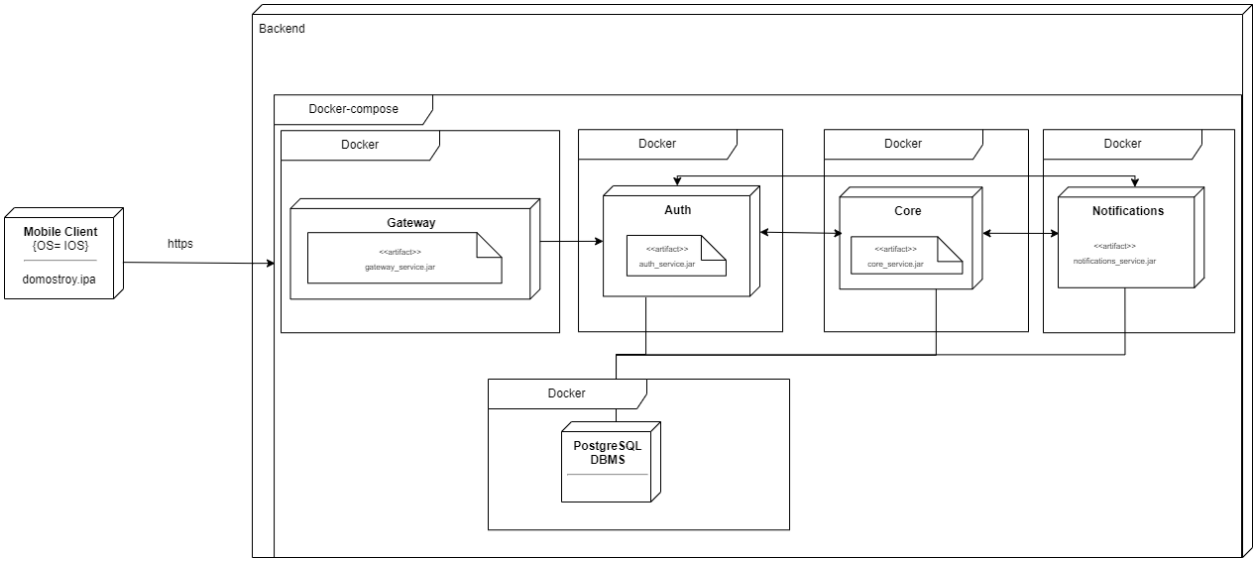

Рисунок 10 – Диаграмма развертывания

На диаграмме представлено, каким образом элементы системы разворачиваются в Docker контейнерах.

4.5 Клиентская часть

4.5.1 Структура мобильного приложения

Основные компоненты проекта по директориям:

– App: инициализация приложения, корневой координатор;

– Flows: пользовательские сценарии;

– Services: бизнес-логика (сетевые запросы);

– Models: модели данных, используемые для парсинга данных с сервера;

– Library: протоколы (разновидность интерфейсов в Swift), базовые классы, расширения стандартных классов, переиспользуемые UI-компоненты и утилиты;

– Resources: изображения и цвета, доступные в любом месте в коде, локализация.

4.5.2 Архитектура мобильного приложения

Для проектирования архитектуры мобильного приложения под операционную систему iOS была выбрана архитектура MVP с использованием координаторов для навигации, известная под названием Coordinated SurfMVP (Surf — студия разработки, создавшая данную архитектуру). Данный подход сочетает принципы классического MVP (Model-View-Presenter) с паттерном координаторов, что позволяет эффективно управлять сложными пользовательскими сценариями. На рисунке 11 изображена схема модуля Coordinated SurfMVP.

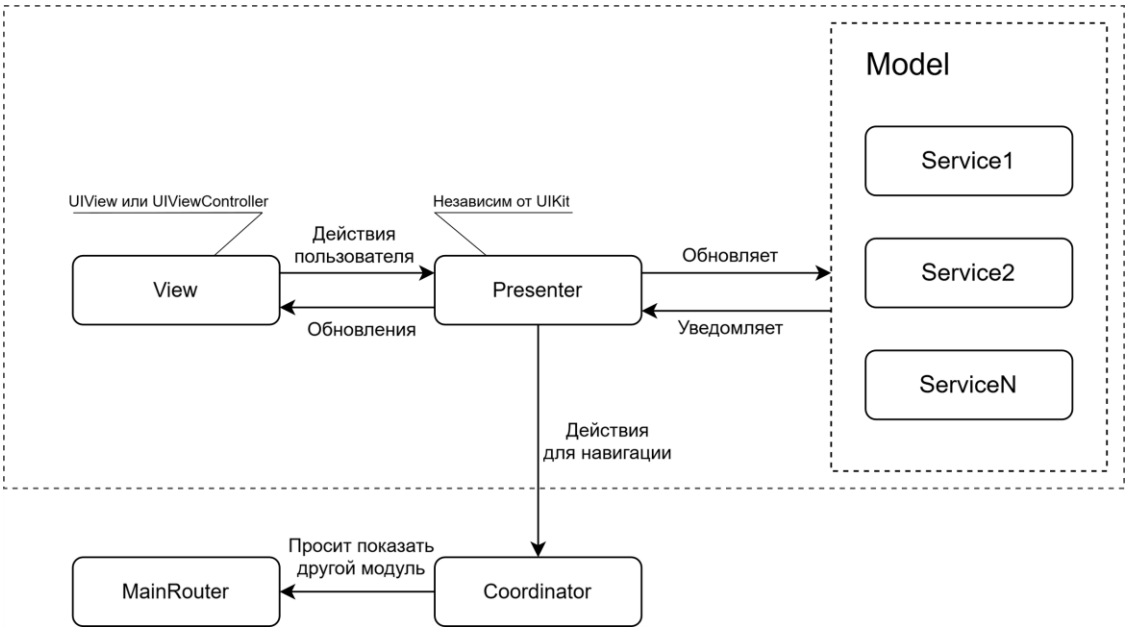

Рисунок 11 – Схема модуля Coordinated SurfMVP

View – отображает данные на экране и оповещает Presenter о действиях пользователя. View никогда не запрашивает данные, только получает их от Presenter.

Presenter – получает от View информацию о действиях пользователя и реагирует на них. Передает события в Model для обновления или обработки внутри себя.

Model – заключает в себе всю бизнес-логику, необходимую для работы модуля.

Уточним, что в рамках данной архитектуры наследники класса UIViewController также считаются View. Это связано с тем, что в UIKit Apple подразумевала использование архитектуры MVC, в которую входит понятие Controller. В MVP же задачи контроллера частично были переданы в Presenter, но названия сущностей, придуманные Apple, остались.

Независимые модули MVP объединяются в пользовательские сценарии (Flows), предназначенные для выполнения какого-то общего действия, приводящего пользователя к желаемому результату. Например, набор экранов авторизации может является примером Flow.

Configurator отвечает за сборку отдельного модуля. Он инициализирует все необходимые компоненты и устанавливает зависимости между ними.

Router отвечает за конфигурацию и отображение других модулей. Под другими модулями не обязательно подразумевается UIViewController. Это может быть дочерний UIView, какое-либо всплывающее сообщение об ошибке.

Координатор (Coordinator) — отвечает за работу навигации не одного отдельного модуля, а набора модулей, которые связаны друг с другом логически. Это упрощает навигацию и работу с приложением. На рисунке 12 изображен пример схемы приложения с использованием Coordinated SurfMVP.

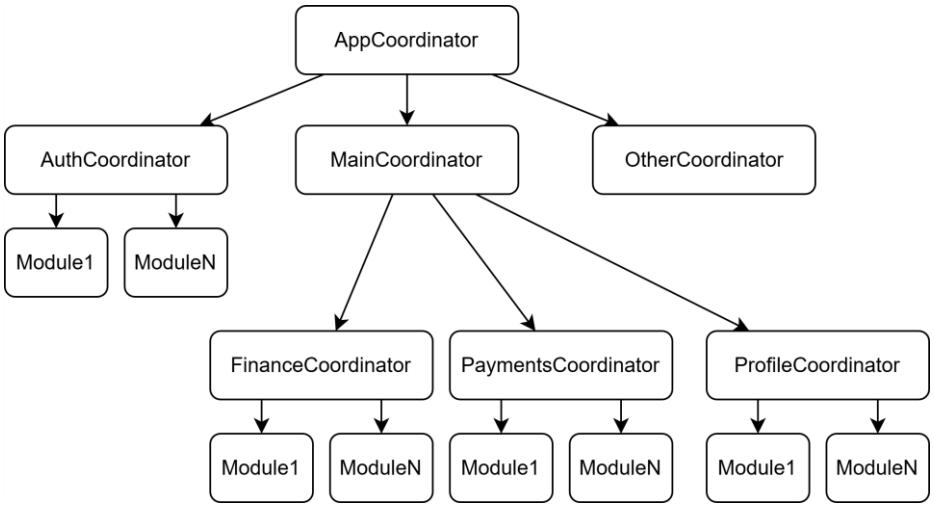

Рисунок 12 – Схема приложения с использованием Coordinated SurfMVP

4.5.3 Графический интерфейс

Графический интерфейс приложения построен на логике пользовательских сценариев и back-end части.

При запуске пользователь увидит приветственные экраны (пример такого – на рисунке 13, которые вкратце ознакомят со спецификой приложения.

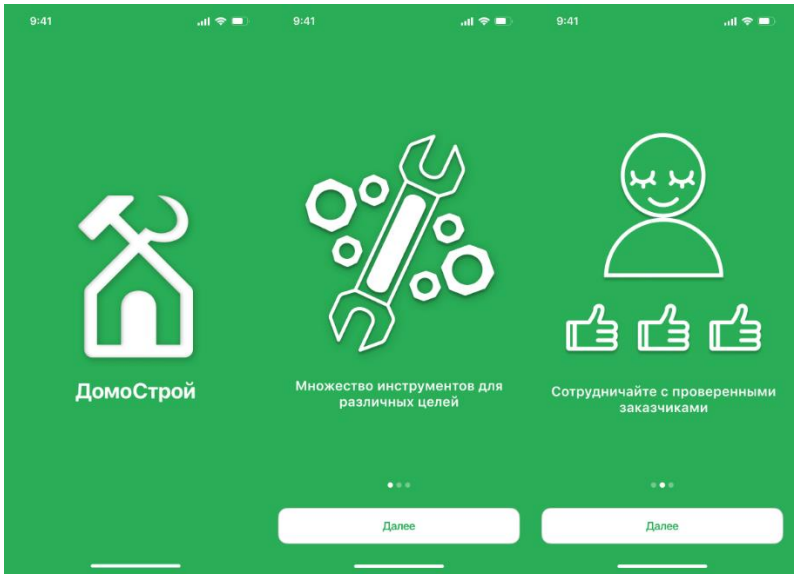

Рисунок 13 – Примеры начальных экранов в приложении

У неавторизованного пользователя есть возможность просматривать объявления (в т.ч. сортировать их, а также фильтровать по таким параметрам, как город или цена), но нет возможности публиковать собственные. Это отражено на рисунке 14.

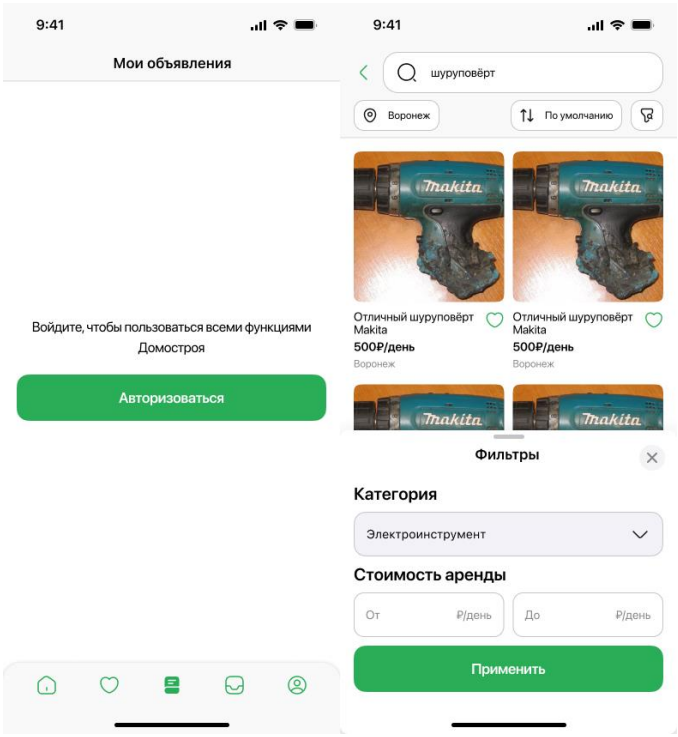

Рисунок 14 – Примеры сценария для неавторизованного пользователя

Авторизация пользователя может произойти только после регистрации, экраны авторизации показаны на рисунке 15. После этого пользователь имеет собственный профиль, возможность создавать объявления и страницу, где можно отслеживать их статус, а также избранные объявления.

Также, пользователь может видеть заявки в разных вариациях, исходящие и входящие, с разным статусами – «принято», «отклонено», «ожидает ответа». Визуальная часть заявок во всех вариациях почти одинаковая, за исключением пометок о статусе и раздела (исходящие или входящие).

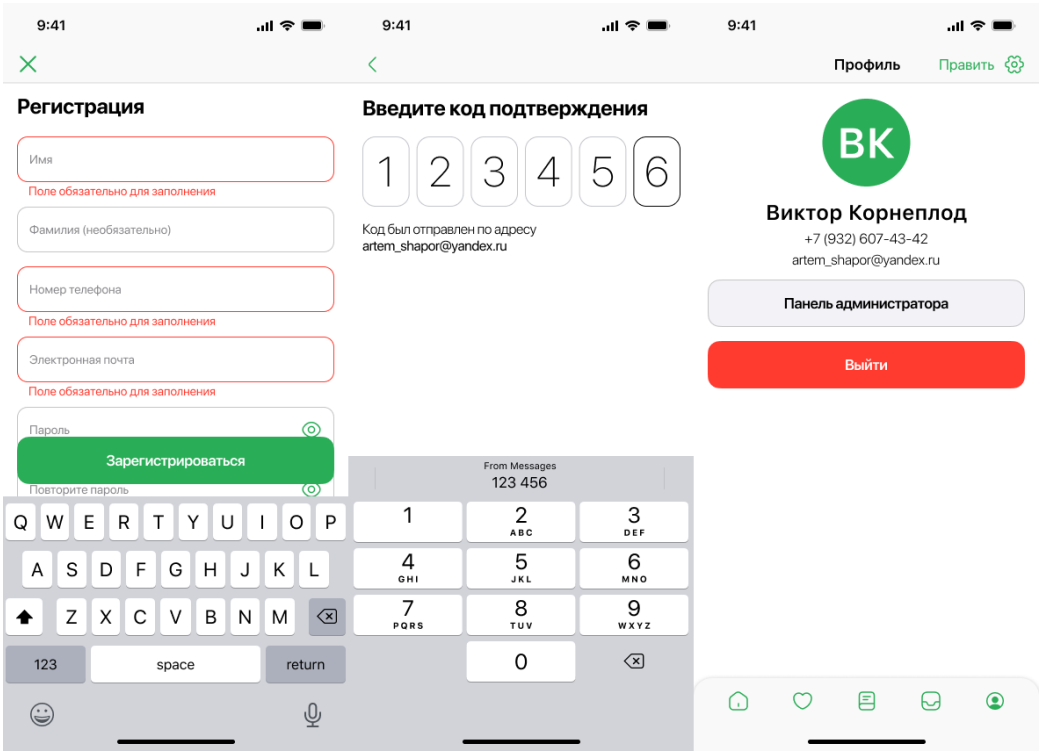

Рисунок 15 – Регистрация пользователя и его профиль

В приложении существует также и роль администратора. Его возможности шире, чем у пользователя – например, он может заблокировать профиль или объявление. Это отражается на UI приложения, что можно увидеть на рисунке 16.

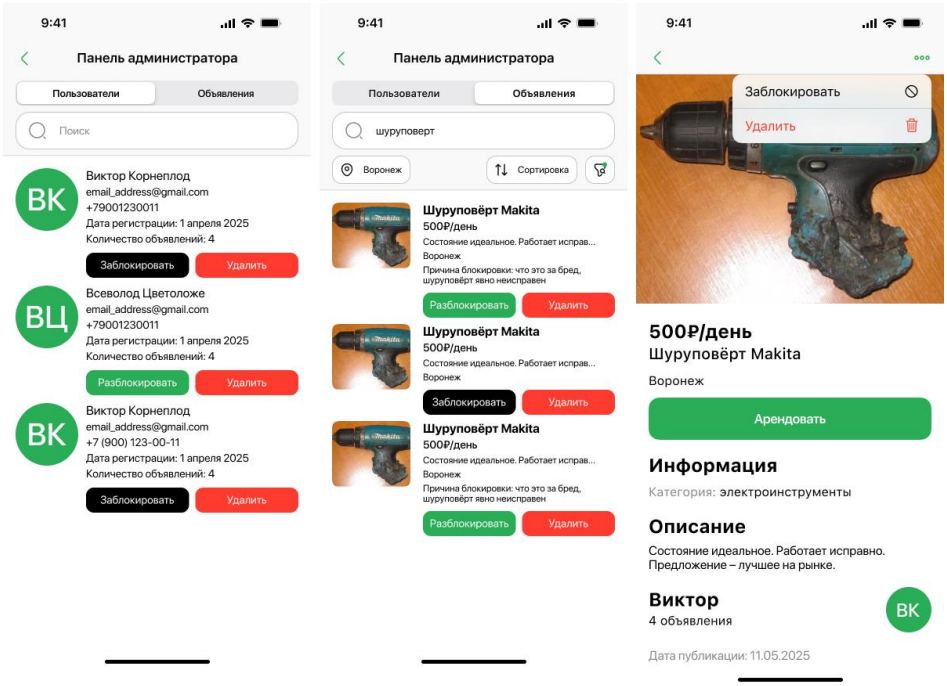

Рисунок 16 – Функционал приложения, доступный администратору

4.6 Взаимодействие компонентов при использовании

В данном параграфе будет описан основной сценарий аренды в приложении. Он будет описывать взаимодействие компонентов системы от входа пользователя в приложение до оформления отклика на объявление. Для наглядной демонстрации сценарий разбит на 2 сценария:

– Поиск нужного объявления;

– Оформление отклика.

Диаграмма взаимодействия компонентов при поиске нужного объявления представлена на рисунке 17.

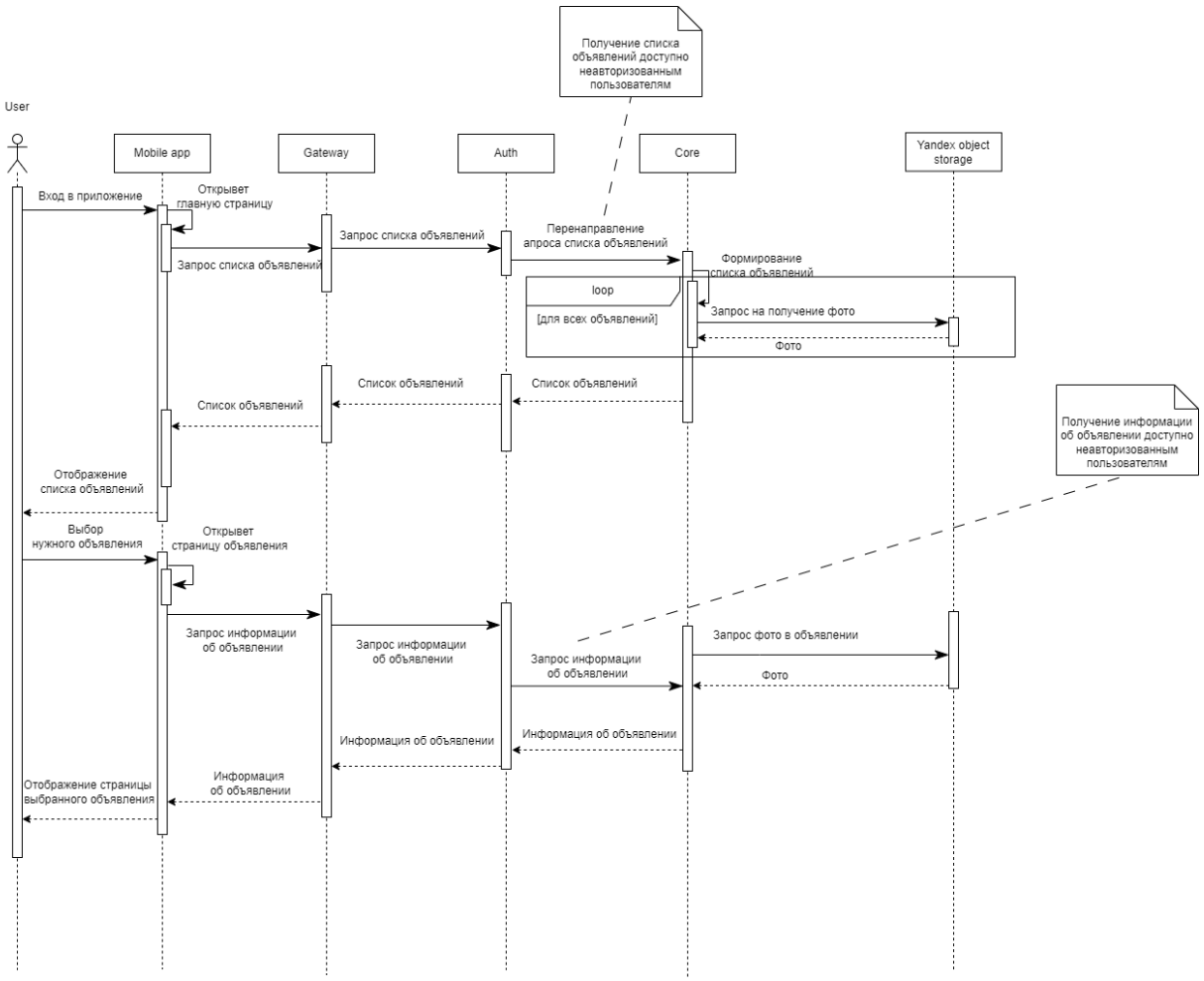

Рисунок 17 – Sequence диаграмма для поиска объявления

Диаграмма взаимодействия компонентов при оформлении отклика на объявление представлена на рисунке 18.

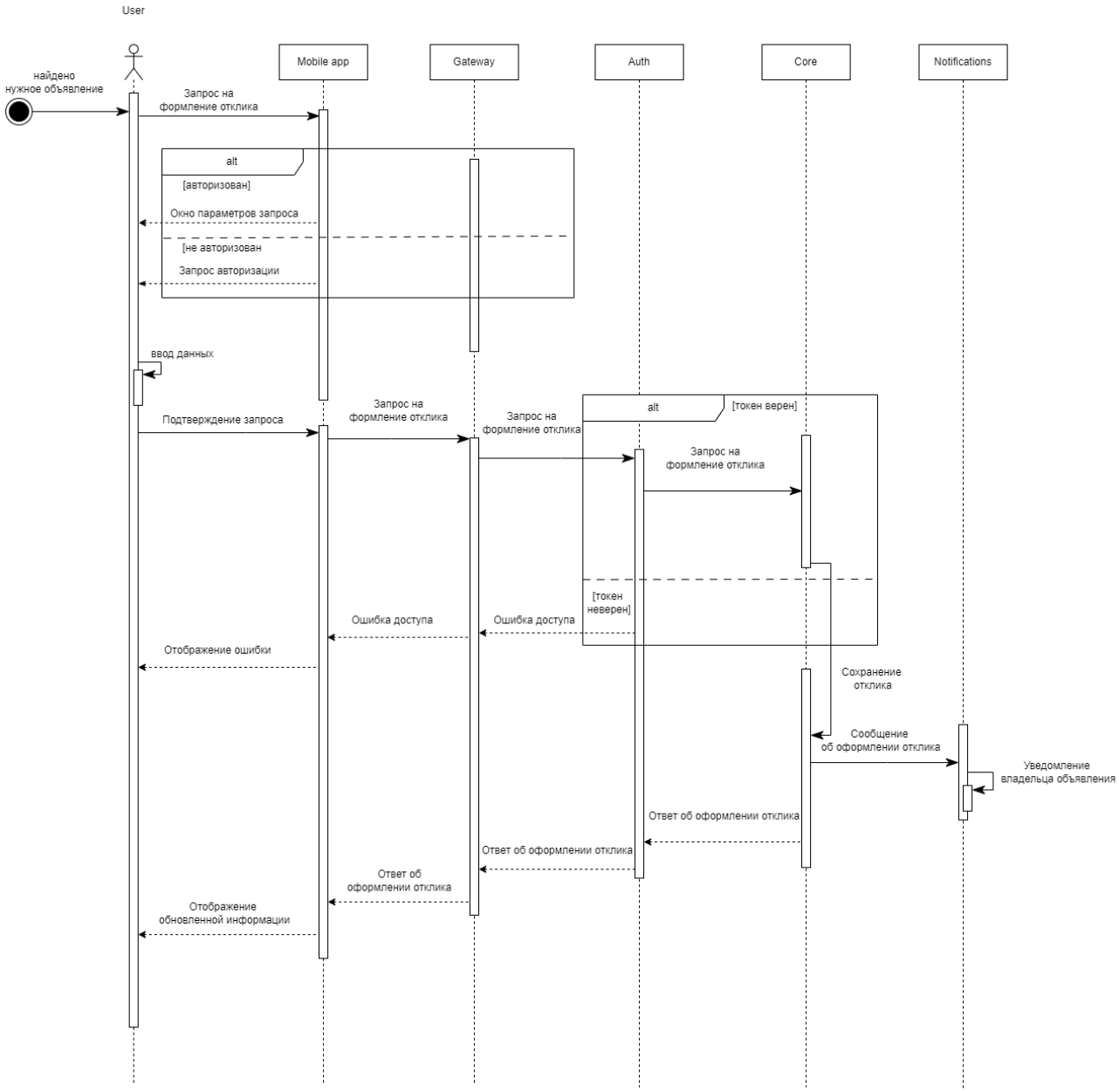

Рисунок 18 – Sequence диаграмма для оформления отклика

Стоит отметить, что подразумевается, что сценарий оформления отклика будет исполнен после сценария поиска объявления.

4.7 Тестирование

В ходе тестирования системы были проведены различные виды проверок:

– Функциональное тестирование;

– Тестирование удобства использования (UX/UI);

– API сервера.

Функциональное тестирование проводилось на устройстве iPhone 12 с установленной операционной системой iOS 17.6.1 и осуществлялось без доступа к исходному коду, что соответствует подходу «черного ящика», при котором тестировщик взаимодействует с мобильным приложением как обычный пользователь. В рамках проверки были протестированы ключевые функции, включая авторизацию, создание и сортировку объявлений.

UX/UI тестирование является важным этапом проверка качества интерфейса и общего пользовательского взаимодействия с мобильным приложением. UI тестирование было направлено на проверку корректности отображения визуальным элементов — таких как кнопки, поля ввода, шрифты. Все основные экраны отобразились корректно и элементы интерфейса функционировали согласно требованиям. UX тестирование — было сосредоточено на оценке удобства использования и логики взаимодействия с приложением.

Тестирование API проводилось с использованием ручного тестирования через Postman. Проверены ключевые эндпоинты: аутентификация, регистрация, поиск объявлений и получение категорий. Все запросы с валидными данными возвращали корректные ответы.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы был разработан сервис для поиска и аренды оборудования и инструментов для ремонта и строительства на дому «Домострой» и достигнуты поставленные цели и задачи.

Созданная система состоит из мобильного приложения и серверной части, реализованной на основе микросервисной архитектуры, и позволяет создавать объявления об аренде инструментов и оборудования для строительства, просматривать необходимую информацию и работать с откликами на объявления.

Для успешной командной деятельности ответственность была разделена между участниками проекта, а также согласованы правила ведения работы: оформлено ведение задач в YouGile и созданы правила коммитов в git.

В процессе работы был проведен анализ целевой аудитории и конкурентов, составлена необходимая документация и диаграммы.

На основе полученных данных были спроектированы интерфейс мобильного приложения, структура базы данных и архитектура взаимодействия компонентов системы. Реализация функциональности охватывает весь жизненный цикл взаимодействия пользователя с платформой: от регистрации и создания объявлений до получения откликов и настройки уведомлений.

Результаты тестирования системы на целевой аудитории показали заявленную степень удовлетворённости пользовательским интерфейсом и удобством использования. Использование микросервисной архитектуры обеспечило модульность, гибкость и масштабируемость решения, что позволит более эффективно вносить изменения в проект в процессе его развития.

Таким образом, разработанная система «Домострой» представляет собой MVP системы для аренды строительного оборудования, которая

обладает потенциалом для дальнейшего развития и коммерческого использования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация SpringBoot [Электронный ресурс]. – Режим доступа: URL:https://spring.io/projects/spring-boot – Заглавление с экрана. – (Дата обращения 17.04.2025).

2. Документация YandexCloud [Электронный ресурс]. – Режим доступа: URL:https://yandex.cloud/ru/docs/tutorials - Заглавление с экрана. – (Дата обращения 10.05.2025).

3. Документация Swift [Электронный ресурс]. – Режим доступа: URL:https://developer.apple.com/documentation/swift - Заглавление с экрана. – (Дата обращения 1.04.2025).

4. Документация UIKit [Электронный ресурс]. – Режим доступа: URL: https://developer.apple.com/documentation/uikit - Заглавление с экрана. – (Дата обращения 3.04.2025).

ПРИЛОЖЕНИЕ А

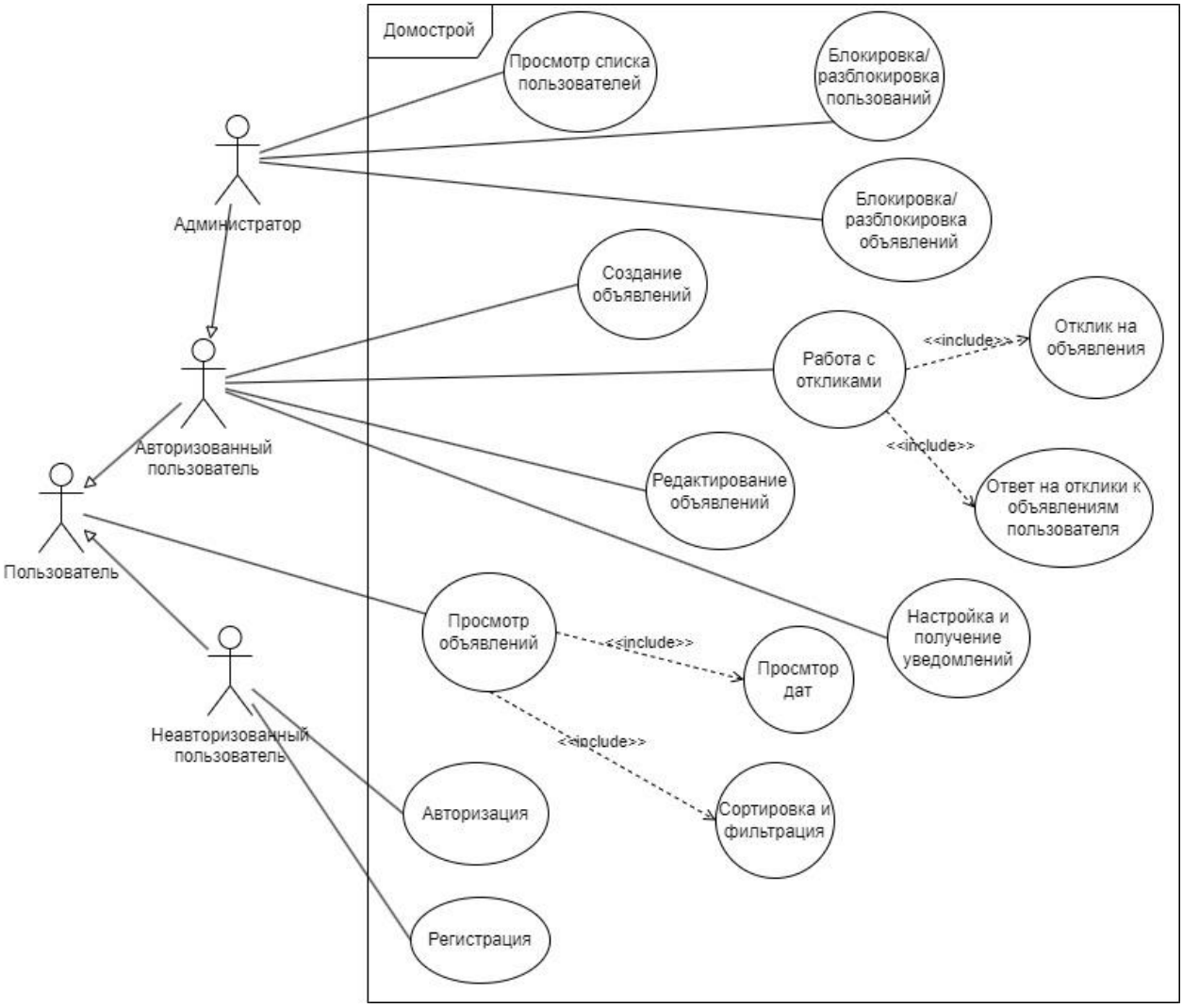

Рисунок 19 – Диаграмма использования приложения