Содержание

[Введение 3](#_Toc193977606)

[1 Обзор аналогов и постановка задачи 4](#_Toc193977607)

[1.1 Постановка задачи 4](#_Toc193977608)

[1.2 Аналитический обзор аналогов 4](#_Toc193977609)

[1.2.1 Booking 5](#_Toc193977610)

[1.2.2 Cian 6](#_Toc193977611)

[1.2.3 Sutochno 7](#_Toc193977612)

[1.3 Выводы по разделу 8](#_Toc193977613)

[2 Проектирование web-приложения 10](#_Toc193977614)

[2.1 Разработка функциональных требований и вариантов использования 10](#_Toc193977615)

[2.2 Описание средств разработки 11](#_Toc193977616)

[2.2.1 Visual Studio Code 11](#_Toc193977617)

[2.2.2 Платформа Node.js 11](#_Toc193977618)

[2.2.3 Язык программирования TypeScript 11](#_Toc193977619)

[2.2.4 Библиотека React 12](#_Toc193977620)

[2.2.5 PostgreSQL 12](#_Toc193977621)

[2.3 Диаграммы UML, взаимосвязь всех компонентов. 13](#_Toc193977622)

[2.4 Выводы по разделу 16](#_Toc193977623)

# Введение

В современном мире, где повышенная мобильность и требования к качеству жизни становятся нормой, аренда жилья превращается в сложный процесс для обеих сторон — арендаторов и арендодателей. Постоянное увеличение количества информации о доступной недвижимости, условиях съёма и предпочтениях арендаторов подчёркивает необходимость эффективных инструментов для упрощения этого процесса. Веб-приложения, разработанные для таких целей, играют ключевую роль, предлагая удобные механизмы поиска, взаимодействия и подачи заявок.

В рамках данной дипломной работы разработано веб-приложение, ориентированное на аренду жилья. Приложение включает функционал поиска, фильтрации, визуализации объектов недвижимости и коммуникации между участниками сделки. Основная задача проекта — создание удобного и простого в использовании интерфейса, который поможет пользователям находить подходящие предложения, сравнивать их и легко взаимодействовать с арендодателями.

Для реализации приложения используются современные технологии: React для разработки пользовательского интерфейса, Node.js для серверной части и PostgreSQL для управления базой данных. Такой подход обеспечивает надёжность, высокую производительность и масштабируемость решения. Принципы модульности и асинхронности, лежащие в основе архитектуры, гарантируют гибкость системы, которая может адаптироваться к изменениям и запросам рынка.

Проект охватывает полный цикл разработки: от анализа требований и проектирования структуры данных до внедрения ключевых функций и тестирования. Итоговая система станет полезным инструментом как для арендаторов, ищущих жильё, соответствующее их запросам, так и для арендодателей, стремящихся найти надёжных жильцов.

Таким образом, данное приложение представляет собой комплексное решение, призванное облегчить процесс аренды жилья, минимизирую усилия пользователей и повышая их удовлетворённость результатом.

# 1 Обзор аналогов и постановка задачи

## 1.1 Постановка задачи

Важным этапом разработки любого веб-приложения является анализ существующих аналогов на рынке. Это позволяет выявить их сильные и слабые стороны, понять основные потребности пользователей и определить направления для внедрения уникальных решений.

Данное веб-приложение, ориентированное на аренду жилья, разработано с использованием современных технологий — React с TypeScript для фронтенда, Node.js для серверной части и PostgreSQL для работы с базой данных. Его цель — автоматизировать и упростить процессы поиска, бронирования и управления жильём. Приложение должно стать удобным инструментом для всех участников: администраторов, владельцев недвижимости и пользователей.

На рынке уже существует множество платформ для аренды жилья. Однако, многие из них имеют недостатки: ограниченная кастомизация поиска, сложность управления объявлениями для владельцев или недостаточное взаимодействие между арендаторами и арендодателями. Это открывает возможность для создания продукта, который предложит гибкий функционал, простой интерфейс и эффективное взаимодействие между пользователями.

Цель проекта — разработка современного веб-приложения, предоставляющего удобные инструменты для поиска, бронирования и управления недвижимостью.

Должны быть выполнены следующие требования:

* обеспечивать возможность регистрации и авторизации;
* поддерживать роли: администратор, владелец, пользователь;
* поиск, фильтрация и сортировку жилья;
* возможность управлять критериями и категориями жилья;
* возможность управлять пользователями, владельцами и их доступом;
* возможность управлять жильём в системе;
* возможность управлять статусами бронирования;
* поддержка чата между пользователем и владельцем;
* возможность просматривать информацию о жилье;
* возможность бронировать интересующее его жильё;
* возможность оставлять отзывы с оцениванием после сделки.

Проект включает несколько ключевых этапов: проектирование архитектуры приложения, разработка интерфейсов и серверной логики, создание базы данных, тестирование и развертывание готового продукта.

## 1.2 Аналитический обзор аналогов

Были проанализированы цели и задачи, поставленные в данном дипломном проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

### 1.2.1 Booking

**Booking** [1] — это одна из ведущих платформ для поиска и бронирования жилья, которая предоставляет пользователям возможность находить подходящие варианты аренды, используя удобные фильтры, просматривать подробные описания объектов и бронировать жильё в режиме онлайн. Пример интерфейса данного сервиса представлен на рисунке 1.1.

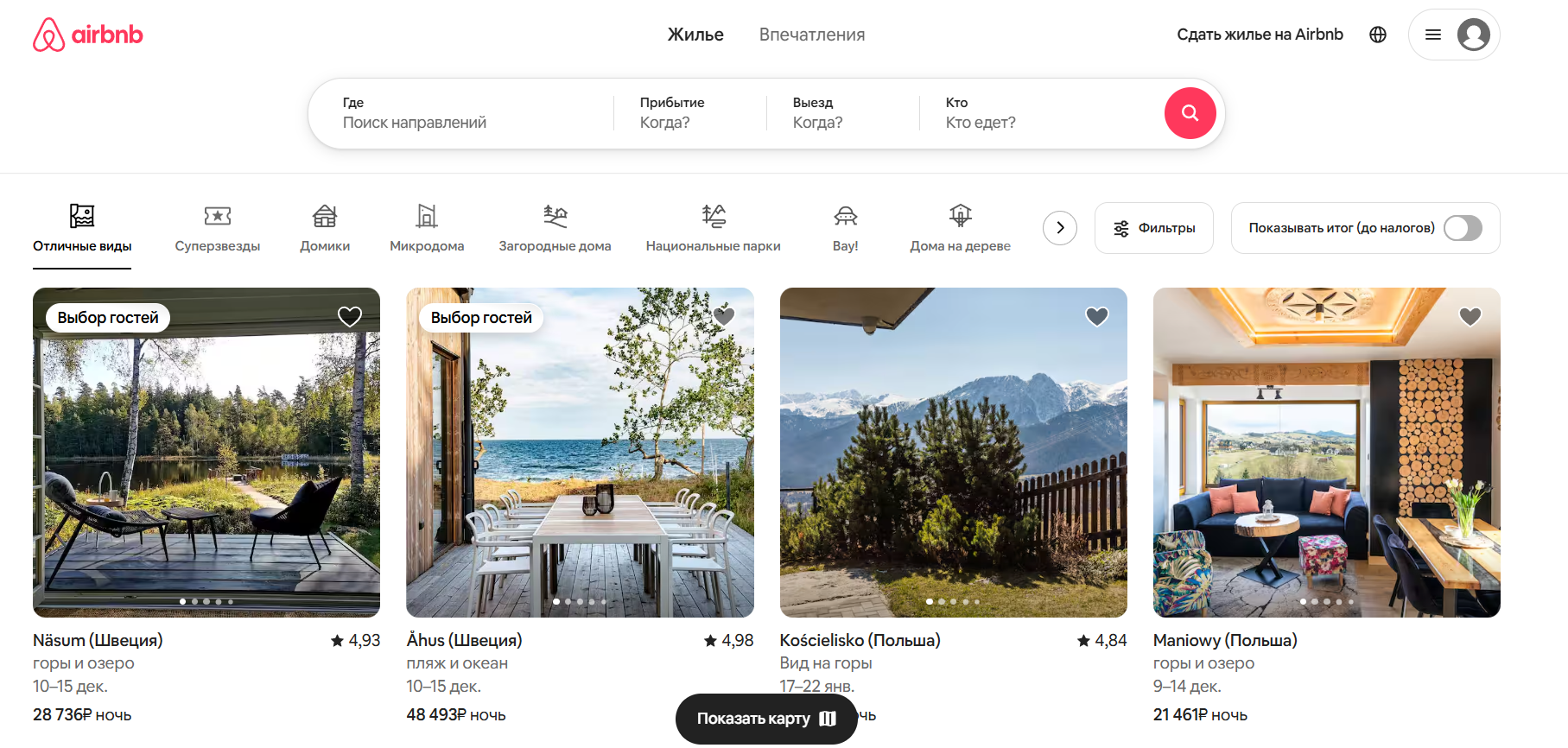


Рисунок 1.1 – Интернет-ресурс **Booking**

Одной из ключевых функций Booking является поиск жилья. Пользователи могут искать объекты недвижимости, применяя фильтры по местоположению, дате, цене, количеству гостей, рейтингу, удобствам и другим параметрам. Сервис также предоставляет персонализированные рекомендации на основе истории поиска и ранее забронированных объектов.

Для удобства пользователей Booking предлагает возможность создавать учётные записи, где можно управлять бронированиями, сохранять интересующие объекты в "Избранное" и оставлять отзывы после завершения проживания. Пользователи могут ознакомиться с отзывами других гостей, что помогает сделать более информированный выбор.

Для владельцев недвижимости Booking предоставляет платформу для регистрации и управления своими объектами. Они могут добавлять описания, фотографии, устанавливать цены, задавать правила отмены бронирования, а также управлять календарём доступности. Сервис предоставляет статистику по каждому объекту, включая количество просмотров, бронирований и средний рейтинг, что помогает владельцам анализировать спрос и оптимизировать свои предложения.

С технической стороны Booking использует сложные алгоритмы поиска и фильтрации, а также высокопроизводительные базы данных для обработки миллионов запросов пользователей ежедневно. Оптимизированная архитектура обеспечивает быструю загрузку страниц и стабильность работы системы даже при высоких нагрузках.

Эта платформа выделяется своей глобальной доступностью, поддержкой множества языков и валют, а также адаптивным интерфейсом, который обеспечивает удобство использования как на компьютерах, так и на мобильных устройствах. Благодаря широкому функционалу и пользовательскому комфорту, Booking остаётся одним из лидеров рынка онлайн-бронирования жилья.

### 1.2.2 ****Cian****

**Cian** [2] — это одна из крупнейших платформ для поиска и аренды недвижимости, которая предоставляет пользователям возможность находить подходящие варианты жилья, используя гибкие фильтры, просматривать подробные описания объектов и напрямую связываться с арендодателями. Пример интерфейса данного сервиса представлен на рисунке 1.2.

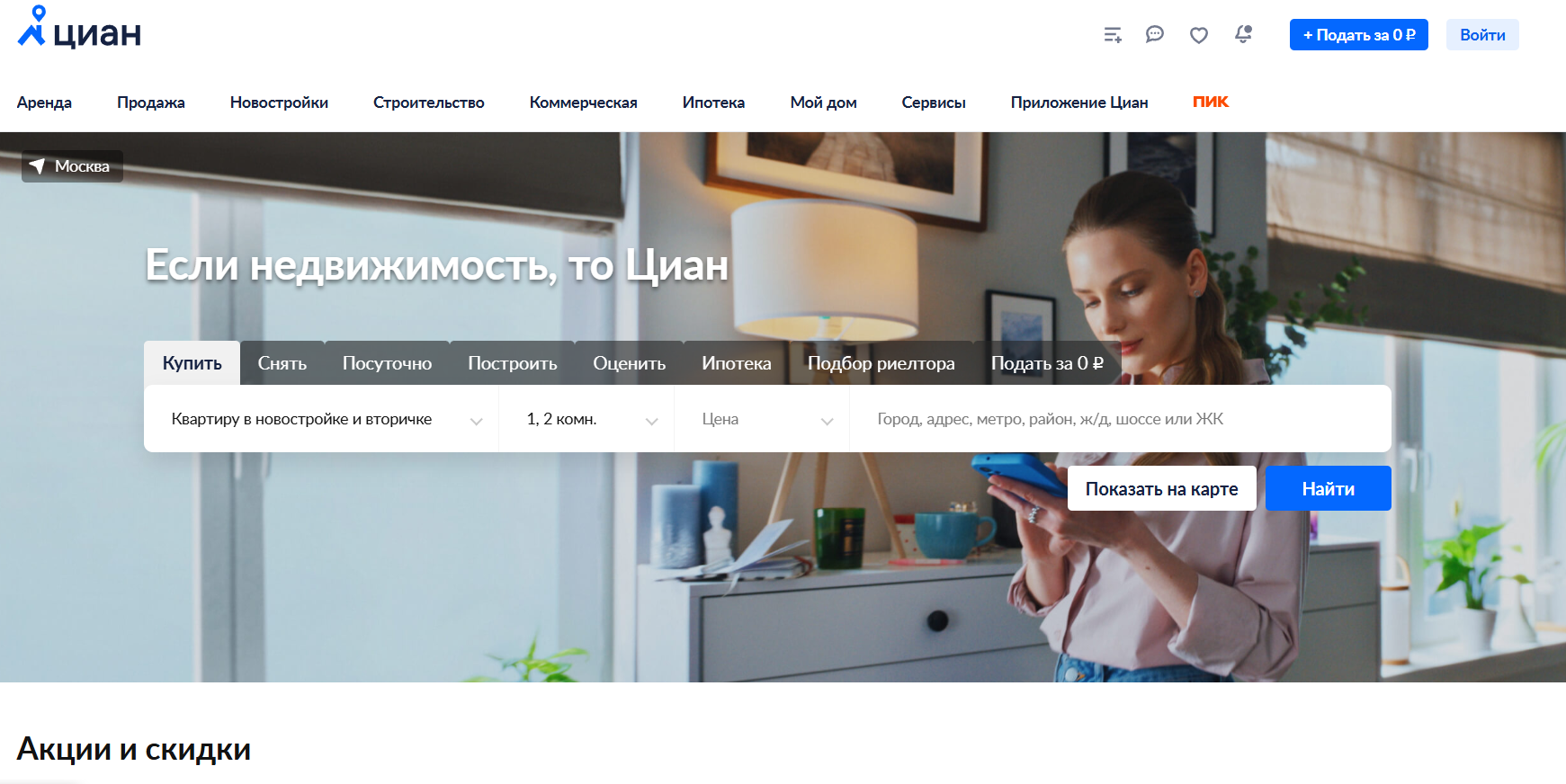


Рисунок 1.2 – Интернет-ресурс **Cian**

Ключевой функцией Cian является поиск недвижимости. Пользователи могут находить объекты, применяя разнообразные фильтры: по цене, району, типу жилья, площади, количеству комнат, сроку аренды и другим параметрам. Также платформа предоставляет удобную карту, где объекты отображаются с указанием точного местоположения, что упрощает выбор подходящего варианта.

Для удобства пользователей Cian предлагает возможность создавать личные кабинеты, где можно сохранять интересующие объявления в "Избранное", управлять своими объявлениями (для арендодателей), отслеживать статус откликов и получать уведомления о новых предложениях.

Для арендодателей и продавцов недвижимости Cian предоставляет функционал для размещения и управления объявлениями. Владельцы могут добавлять фотографии, описания, указывать стоимость и условия аренды, а также редактировать или удалять объекты. Платформа предоставляет статистику для каждого объявления, включая количество просмотров и откликов, что помогает анализировать спрос и корректировать предложения.

С технической стороны Cian использует современные технологии для работы с большими объёмами данных, что обеспечивает быструю загрузку страниц, точный поиск и стабильную работу сервиса. Платформа поддерживает интеграцию с внешними системами, такими как геолокационные сервисы и платёжные системы, что делает её удобной как для арендаторов, так и для арендодателей.

Удобный и интуитивно понятный интерфейс Cian адаптирован для работы как на компьютерах, так и на мобильных устройствах, благодаря чему пользователи могут быстро находить и просматривать предложения в любое время. Благодаря обширной базе данных объектов, гибкости поиска и дополнительным функциям, Cian остаётся одним из лидеров среди платформ для аренды и покупки недвижимости.

### 1.2.3 Sutochno

Sutochno [3] — это онлайн-платформа, специализирующаяся на краткосрочной аренде жилья. Она предоставляет пользователям удобные инструменты для поиска и бронирования квартир, домов, комнат и других объектов недвижимости по всей стране. Пример интерфейса данного сервиса представлен на рисунке 1.3.

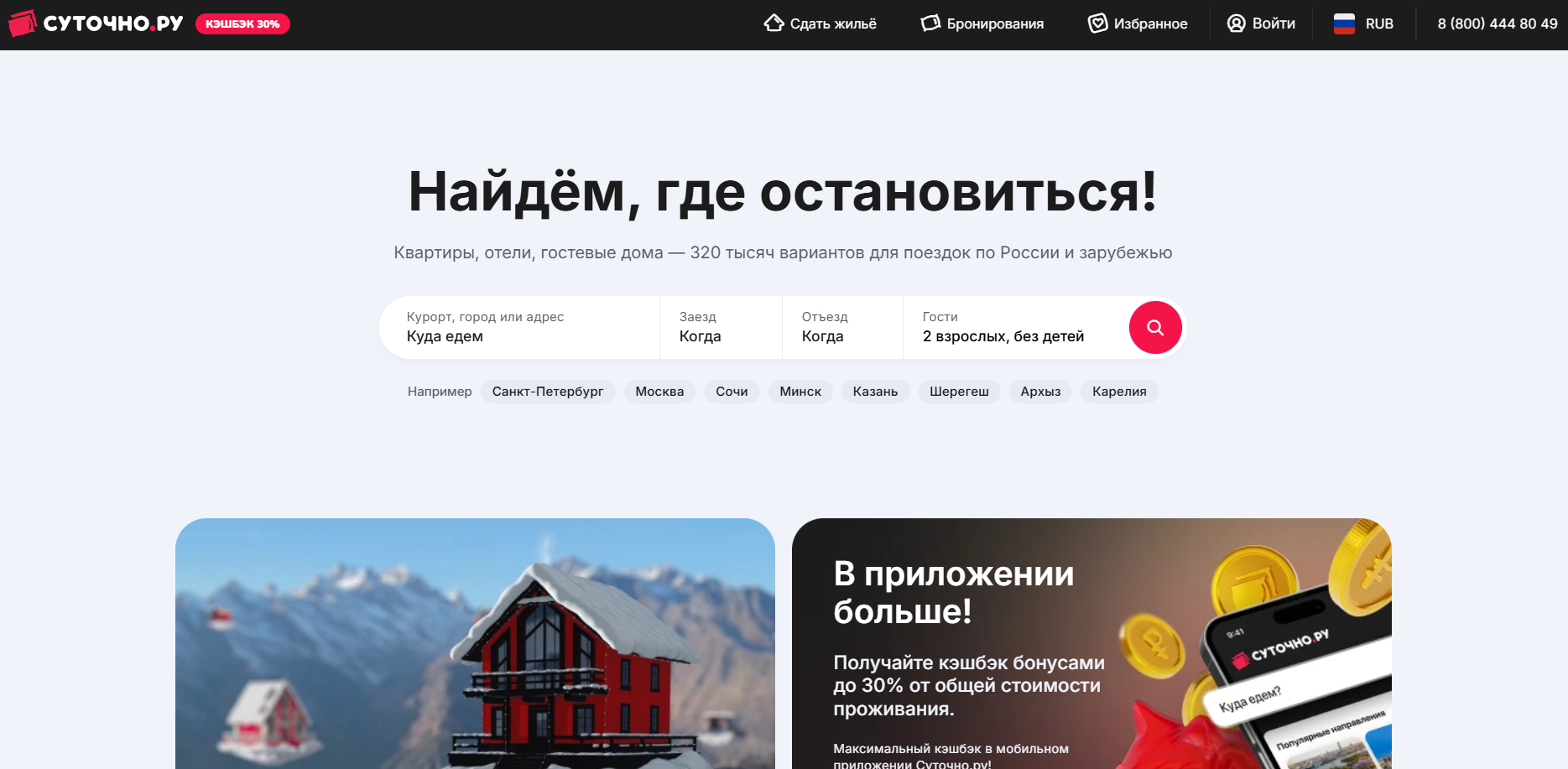


Рисунок 1.3 – Интернет-ресурс Sutochno

Основной функционал Sutochno сосредоточен на быстром поиске и аренде жилья на сутки или более короткие сроки. Пользователи могут искать объекты, применяя фильтры по цене, дате, количеству комнат, типу жилья, местоположению и другим параметрам. Система также предлагает карту для визуального поиска, на которой объекты отмечены с указанием стоимости аренды.

Платформа позволяет пользователям регистрировать личные кабинеты, где можно управлять бронированиями, сохранять интересующие объекты в "Избранное" и оставлять отзывы после завершения проживания. Пользователи могут общаться с арендодателями через встроенный чат, уточняя детали проживания.

Для арендодателей Sutochno предоставляет возможность размещать и редактировать объявления. Владельцы могут добавлять описания, фотографии, указывать правила проживания, условия отмены бронирования и доступные даты. Платформа поддерживает аналитические инструменты, позволяющие владельцам отслеживать популярность их объектов: количество просмотров, бронирований и отзывы.

С технической стороны Sutochno использует современные инструменты для обеспечения быстродействия и стабильности работы платформы. Высокопроизводительные базы данных и алгоритмы фильтрации позволяют обрабатывать большие объёмы информации, обеспечивая пользователям точные результаты поиска.

Sutochno выделяется своей ориентацией на рынок, предлагая широкий выбор объектов в крупных городах и популярных туристических направлениях. Благодаря функционалу, направленному на удобство и прозрачность для всех участников, платформа остаётся востребованным инструментом как для арендаторов, так и для арендодателей, обеспечивая комфортное взаимодействие и безопасные сделки.

## 1.3 Выводы по разделу

В данном разделе был проведён обзор аналогичных веб-приложений, таких как Booking, Cian и Sutochno, что позволило выявить ключевые функции, которые должны быть реализованы в нашем приложении. Эти платформы предоставляют пользователям широкий спектр возможностей: от регистрации и авторизации до поиска и бронирования жилья с использованием удобных фильтров, а также управления объектами недвижимости для владельцев.

Особое внимание уделено функционалу для различных ролей пользователей. Для владельцев объектов аренды важными элементами являются инструменты для добавления, редактирования и удаления объектов, управления бронированиями, а также ведения переписки с арендаторами. Пользователям необходимы функции просмотра и поиска жилья, бронирования, отслеживания статуса бронирований, а также возможности оставлять отзывы.

Также обсуждены требования к удобному интерфейсу, который должен быть интуитивно понятным и обеспечивать эффективное взаимодействие между владельцами недвижимости, арендаторами и администраторами системы. Для визуализации основных процессов взаимодействия и проектирования системы была разработана UML-диаграмма, отражающая ключевые сущности и связи между ними.

Результаты обзора аналогичных приложений позволили определить направления для улучшения и выделить уникальные особенности, которые будут учтены при разработке функционала нашего веб-приложения.

# 2 Проектирование web-приложения

## 2.1 Разработка функциональных требований и вариантов использования

Приложение должно предоставлять функционал для регистрации и авторизации, позволяя пользователям идентифицировать себя и получать доступ к возможностям в зависимости от их роли: администратор, владелец недвижимости или пользователь.

Администратор получит возможность управлять системой, включая редактирование категорий и критериев для поиска жилья, управление пользователями и их доступом, а именно блокировка пользователей, верификация пользователья, чтобы он мог становиться владельцем, одобрять или отклонять создание жилья, а также просмотр и анализ статистики по всем объектам и бронированиям. Это позволит обеспечить порядок и прозрачность работы приложения.

Владелец недвижимости сможет добавлять, редактировать и удалять свои объекты аренды, управлять статусами бронирований, а также просматривать подробную статистику по своим объектам, включая количество просмотров, бронирований и отзывы, также он сам может создать бронирование на какие-то даты. Для удобства взаимодействия с пользователями владельцу будет доступен встроенный чат, через который можно уточнять детали бронирования и отвечать на вопросы.

Пользователь сможет искать жильё, используя удобные фильтры и сортировку, бронировать интересующие варианты, просматривать статус своих бронирований и историю операций. Для улучшения качества сервиса пользователь сможет оставлять отзывы и оценки после завершения аренды, а также управлять своими личными данными через настройки профиля.

На рисунке 2.1 представлена диаграмма вариантов использования.

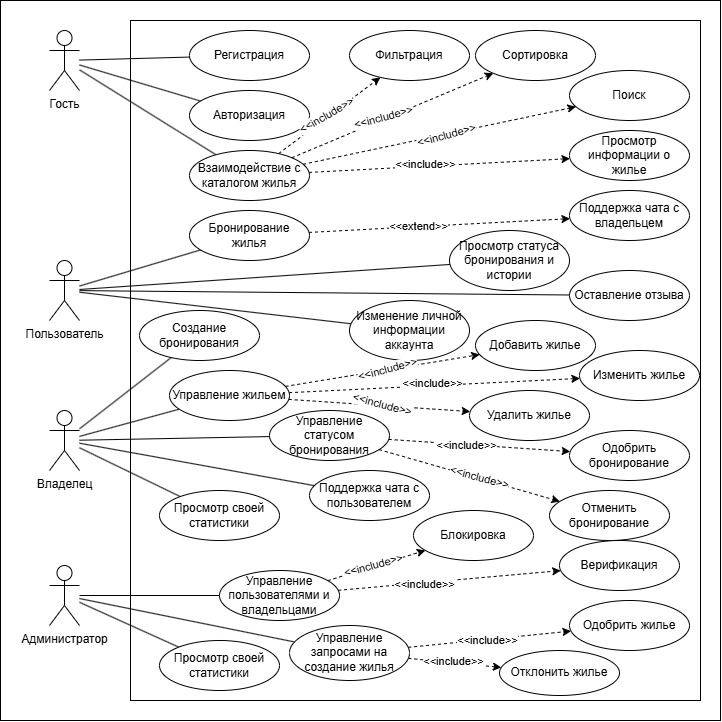


Рисунок 2.1 – Диаграмма вариантов использования

Диаграмма UML вариантов использования позволяет понять, что доступно каждой роли, доступной в данном веб-приложении.

## 2.2 Описание средств разработки

При разработке приложения были использованы:

* интегрированная среда разработки Visual Studio Code;
* платформа Node.js;
* язык разработки проекта TypeScript;
* библиотека React;
* база данных PostgreSQL.

### 2.2.1 Visual Studio Code

Visual Studio Code (VS Code) — это популярная и мощная интегрированная среда разработки, которая поддерживает широкий спектр языков программирования и технологий, включая JavaScript, TypeScript, Python, C++, а также фреймворки и библиотеки, такие как React и Node.js. Одной из главных особенностей VS Code является его легковесность и гибкость, что делает его идеальным инструментом для разработчиков, которые ценят скорость работы и настройку под свои потребности. VS Code предоставляет множество встроенных инструментов для написания, отладки и тестирования кода, а также поддерживает расширения, которые помогают работать с различными базами данных, фреймворками и инструментами. Это позволяет разработчикам эффективно работать на всех этапах разработки, начиная от написания кода и заканчивая его тестированием и деплоем.

### 2.2.2 Платформа Node.js

**Node.js** — это серверная среда выполнения для JavaScript, основанная на движке V8 от Google Chrome. Она позволяет запускать JavaScript-код на сервере, а не только в браузере, что делает возможным создание полноценного серверного приложения. Node.js особенно популярен для создания асинхронных и масштабируемых веб-приложений благодаря своей неблокирующей, ориентированной на события модели ввода-вывода. Она идеально подходит для приложений, которым требуется высокая производительность, таких как чат-системы, онлайн-игры, системы обмена сообщениями, а также для приложений, обрабатывающих большое количество одновременных запросов. Благодаря большому количеству доступных библиотек в **npm** (Node Package Manager), Node.js предоставляет гибкие возможности для разработки приложений различной сложности.

### 2.2.3 Язык программирования TypeScript

**TypeScript** — это язык программирования, являющийся надстройкой над JavaScript, который добавляет статическую типизацию. TypeScript был разработан для того, чтобы сделать процесс разработки на JavaScript более предсказуемым и безопасным, обеспечивая строгую проверку типов на этапе компиляции. Это позволяет легче обнаруживать ошибки и предотвращать их еще до того, как код будет запущен. TypeScript также поддерживает более удобное масштабирование приложений, улучшает читаемость кода и способствует более качественной поддержке крупных проектов. В TypeScript можно использовать все стандартные возможности JavaScript, при этом добавляются дополнительные функции, такие как интерфейсы, перечисления и типы данных, что помогает создавать более структурированные и поддерживаемые приложения.

### 2.2.4 Библиотека React

**React** — это популярная JavaScript-библиотека для создания пользовательских интерфейсов, разработанная компанией Facebook. React позволяет строить веб-приложения на основе компонентного подхода, где каждый компонент представляет собой независимую часть интерфейса с собственным состоянием и поведением. Это упрощает разработку и повторное использование компонентов, что особенно важно при создании крупных и сложных интерфейсов. Одной из ключевых особенностей React является использование виртуального DOM, что помогает повысить производительность приложения. Виртуальный DOM позволяет React минимизировать количество операций с реальным DOM, обновляя только те части интерфейса, которые изменились. Это приводит к ускорению рендеринга и улучшению общего пользовательского опыта. React активно используется в веб-разработке для создания динамичных и отзывчивых интерфейсов, особенно для одностраничных приложений (SPA).

### 2.2.5 PostgreSQL

**PostgreSQL** — это объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом, которая поддерживает сложные запросы, транзакции, различные типы данных и расширяемость. PostgreSQL является одним из самых популярных решений для хранения данных в крупных и высоконагруженных веб-приложениях. Благодаря поддержке ACID-транзакций, PostgreSQL обеспечивает надежность и консистентность данных. Он также поддерживает работу с географическими данными, а также предоставляет возможность использования хранимых процедур и триггеров, что делает его идеальным выбором для сложных приложений. PostgreSQL также известен своей масштабируемостью и высокими показателями производительности, особенно при обработке больших объемов данных. Его гибкость и расширяемость позволяют интегрировать его в широкий спектр приложений, от малых стартапов до крупных корпоративных решений.

## 2.3 Диаграммы UML, взаимосвязь всех компонентов.

Диаграмма базы данных таблиц (Database Table Diagram) – это визуальное представление структуры базы данных и отношений между таблицами, которые хранятся в этой базе данных. Диаграмма представлена на рисунке 2.2.

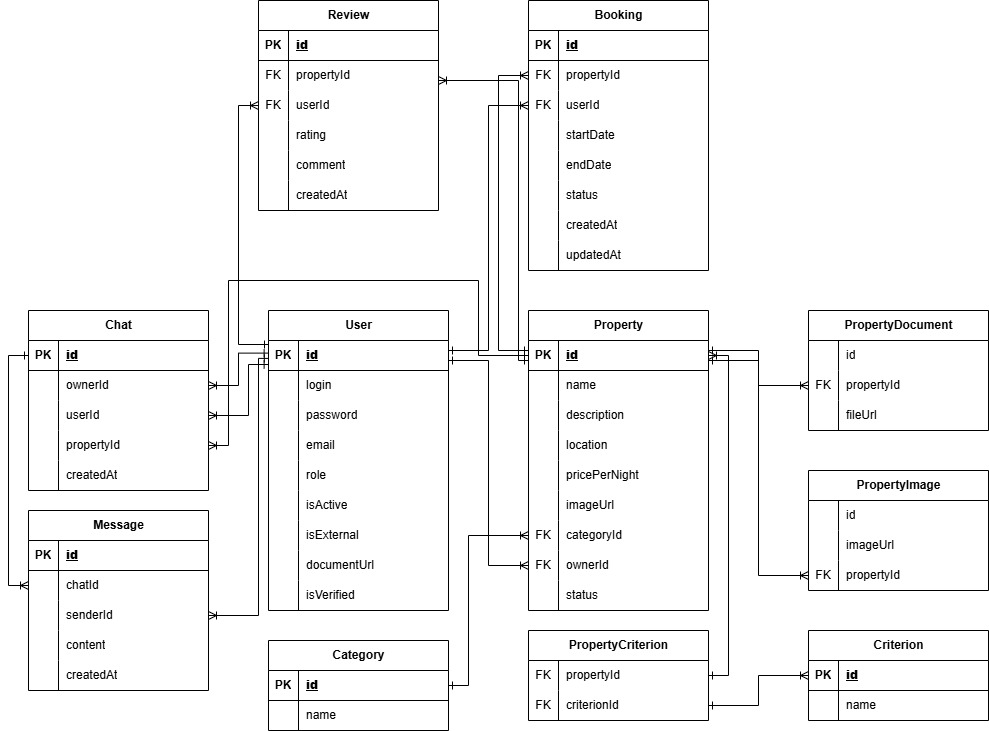


Рисунок 2.2 – Логическая схема базы данных

В этой базе данных используется сложная структура для управления пользователями, недвижимостью, бронированиями и отзывами. Модели и таблицы в базе данных связаны между собой с помощью различных связей и перечислений, что позволяет эффективно хранить и извлекать данные. Давайте подробно разберем каждую из моделей.

Таблица User представляет собой пользователей системы. В этой таблице хранятся такие данные, как уникальный идентификатор пользователя (поля id), его логин (login), зашифрованный пароль (password), а также уникальный адрес электронной почты (email). Это основные поля, необходимые для аутентификации пользователей в системе. Поле role указывает на роль пользователя в системе, которая может быть одной из трех: пользователь (USER), владелец жилья (OWNER), или администратор (ADMIN). Каждому пользователю присваивается одна из этих ролей, которая определяет его доступ к различным функциям приложения. Роль по умолчанию для пользователя — это (USER). Каждый пользователь может иметь несколько связанных объектов недвижимости, если он является владельцем жилья, и это отражается через связь с таблицей Property. Кроме того, пользователи могут оставлять отзывы о жилье и бронировать жилье, что также фиксируется в базе данных в таблицах Review и Booking. Также есть поля isActive (отвечает за блокировку пользователя), isExternal (означает что пользователь создан владельцем), isVerified (отвечает за возможность пользователя стать владельцем), documentUrl (поле хранит документ пользователя, который должен помочь администратору принят решение на верификацию).

Таблица Property описывает объекты недвижимости, которые могут быть забронированы пользователями. В этой таблице хранится информация о каждом жилье, включая его уникальный идентификатор (id), название (name), описание (description), местоположение (location) и цену за ночь (pricePerNight). Также указывается ownerId, который связывает данное жилье с конкретным пользователем, являющимся его владельцем. Связь с владельцем осуществляется через поле owner, которое ссылается на модель User. Важно, что одно жилье может быть связано только с одним владельцем, и это позволяет организовать структуру, где каждый объект недвижимости принадлежит конкретному пользователю. Кроме того, жилье может быть связано с категориями через поле categoryId, которое указывает на таблицу Category. Категории позволяют классифицировать недвижимость, например, на дома, апартаменты, квартиры и так далее. Связь с категорией реализована через поле category, которое ссылается на модель Category. Для каждого объекта недвижимости можно указать несколько критериев (например, наличие Wi-Fi, бассейна и прочее), которые описаны в таблице Criterion. Эта связь реализована через промежуточную таблицу PropertyCriterion, где каждый объект недвижимости может быть связан с несколькими критериями, что предоставляет гибкость в поиске и фильтрации объектов. Также данная таблица связана с таблицей PropertyImage связью один ко многим, а также с таблицей PropertyDocument. Помимо этого, есть поле status, котоырое может принимать значение PENDING, APPROVED или REJECTED.

Таблица PropertyDocument хранит документы, прикреплённые к конкретному объекту недвижимости. id — уникальный идентификатор, propertyId — внешний ключ к объекту недвижимости и fileUrl — ссылка на загруженный документ. Каждый документ принадлежит конкретному объекту, и его цель — предоставить подтверждающие документы владельца.

Таблица PropertyImage используется для хранения изображений, связанных с объектами недвижимости. id — уникальный идентификатор изображения, imageUrl — ссылка на изображение, propertyId — связь с объектом недвижимости (с каскадным удалением). Позволяет отображать галерею объектов недвижимости.

Таблица Review хранит отзывы пользователей о различных объектах недвижимости. Каждый отзыв включает в себя уникальный идентификатор (id), оценку (rating) и комментарий (comment). Эти данные помогают пользователям оценить качество жилья, с которым они сталкивались. Каждый отзыв связан с конкретным жильем через поле propertyId, которое ссылается на модель Property, и с пользователем, который оставил отзыв, через поле userId, которое ссылается на модель User. Кроме того, с помощью уникального ограничения на сочетание propertyId и userId обеспечивается, что каждый пользователь может оставить только один отзыв для конкретного объекта недвижимости. Это важно, чтобы избежать дублирования информации.

Таблица Booking фиксирует информацию о бронированиях, сделанных пользователями. Каждый объект бронирования связан с конкретным жильем через поле propertyId и с пользователем, который сделал бронирование, через поле userId. Бронирование содержит информацию о периоде (даты начала и окончания), а также о статусе бронирования, который может быть в одном из трех состояний: ожидает подтверждения (PENDING), подтверждено (CONFIRMED), или отменено (CANCELLED). Кроме того, в таблице сохраняются временные метки создания (createdAt) и последнего обновления (updatedAt), что позволяет отслеживать изменения в статусе бронирования. Каждое бронирование связано с конкретным жильем, которое было забронировано пользователем, что позволяет легко отслеживать все бронирования для конкретных объектов недвижимости.

Таблица Chat хранит информацию о чатах, используемых для общения между владельцем объекта недвижимости и пользователем, заинтересованным в этом объекте. Каждый чат имеет уникальный идентификатор (id) и связан с объектом недвижимости через поле propertyId, которое ссылается на таблицу Property. Поле ownerId указывает на владельца объекта недвижимости (пользователя из таблицы User), а поле userId — на пользователя, который инициировал общение или заинтересован в объекте. Также хранится временная метка (createdAt), фиксирующая время создания чата. Эта структура позволяет легко отслеживать чаты, группировать их по объектам недвижимости и участникам общения, что делает систему коммуникации удобной и прозрачной.

Таблица Message предназначена для хранения данных о сообщениях, отправленных в рамках чатов. Каждое сообщение имеет уникальный идентификатор (id) и связано с определенным чатом через поле chatId, которое ссылается на таблицу Chat. Поле senderId указывает на отправителя сообщения (пользователя из таблицы User), а в поле content хранится текст сообщения. Дополнительно сохраняется временная метка (createdAt), фиксирующая время отправки сообщения. Эта таблица обеспечивает структурированное хранение всех сообщений, упрощая управление историей общения, идентификацию участников и анализ переписки.

Таблица Category используется для классификации объектов недвижимости. В таблице Category хранится уникальное название каждой категории, например, "Дом", "Апартаменты", "Коттедж". Эта классификация позволяет пользователям удобно искать и фильтровать жилье по типам. Каждая категория может быть связана с несколькими объектами недвижимости, что отображается в поле properties, ссылающемся на таблицу Property.

Таблица Criterion хранит критерии, которые могут быть применены к объектам недвижимости. Примеры таких критериев включают наличие Wi-Fi, бассейна, парковки и так далее. Эти критерии помогают пользователям фильтровать объекты недвижимости по нужным им характеристикам. В таблице Criterion хранится уникальное название критерия, а через промежуточную таблицу PropertyCriterion осуществляется связь между объектами недвижимости и критериями.

Таблица PropertyCriterion является промежуточной таблицей, которая реализует связь "многие ко многим" между объектами недвижимости и критериями. Каждый объект недвижимости может быть связан с несколькими критериями, и каждый критерий может быть связан с несколькими объектами. Это позволяет гибко управлять фильтрацией объектов по множеству параметров. Составной первичный ключ таблицы, состоящий из полей propertyId и criterionId, гарантирует уникальность каждой связи.

Перечисление Role определяет возможные роли пользователей в системе. В системе три роли: USER (пользователь), OWNER (владелец жилья) и ADMIN (администратор). Роль пользователя влияет на его доступ к функционалу приложения. Например, только владельцы жилья могут создавать объекты недвижимости, а администраторы могут управлять всеми пользователями и объектами.

Перечисление BookingStatus определяет возможные статусы бронирования. Это позволяет отслеживать, на каком этапе находится каждое бронирование. Статусы включают: PENDING — бронирование ожидает подтверждения, CONFIRMED — бронирование подтверждено, CANCELLED — бронирование отменено. Это перечисление помогает системе отслеживать процесс бронирования и контролировать состояние каждой заявки.

## 2.4 Выводы по разделу

Разработка архитектуры проекта является важнейшим этапом, поскольку она служит основой для создания структурированной и функционально полноценной системы. В процессе разработки архитектуры важно не только определить основные компоненты приложения, но и четко представить, как эти компоненты будут взаимодействовать друг с другом для достижения цели. В данном разделе мы подробно рассматривали структуру базы данных для системы управления недвижимостью. На основе предложенной архитектуры можно сделать несколько важных выводов.