МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РЕСПУБЛИКИ КАЗАХСТАН

КАЗАХСТАНСКО-НЕМЕЦКИЙ УНИВЕРСИТЕТ

Факультет Инжиниринга и информационных технологий

Образовательная программа: 6B06106 – «Информационный инжиниринг в экономике»

ДИПЛОМНЫЙ ПРОЕКТ

на тему «Разработка веб приложения для инвестиционной строительной компании «МАСАТО»

Исполнитель: студент 4 курса

\_\_\_\_\_\_\_\_\_ Васильев М.А.

подпись

Научный руководитель: к.т.н.

\_\_\_\_\_\_\_\_\_ Дадаева И.Г.

подпись

Нормоконтролер:

\_\_\_\_\_\_\_\_\_ Жантлеуова А.К.

подпись

Работа допущена к защите перед Аттестационной комиссией решением факультета от «12» мая 2025 г., протокол №10.

Декан факультета к.т.н., профессор Кегенбеков Ж.К.

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

подпись

Алматы, 2025

**Подтверждение самостоятельности выполнения работы**

Я, Васильев Максим Александрович, подтверждаю, что данная работа выполнена мною самостоятельно. Все используемые цитаты обозначены по тексту работы и отражены в списке использованных источников согласно «Методическим рекомендациям по оформлению письменных работ (бакалавриат)».

Я согласен с тем, что моя работа может быть проверена на антиплагиат.

Алматы, 8 мая 2025 г.

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

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РЕСПУБЛИКИ КАЗАХСТАН

КАЗАХСТАНСКО-НЕМЕЦКИЙ УНИВЕРСИТЕТ

Факультет Инжиниринга и информационных технологий

Образовательная программа: 6B06106 – «Информационный инжиниринг в экономике»

Задание на выполнение дипломного проекта

Студент Васильев Максим Александрович

Тема дипломного проекта «Разработка веб приложения для инвестиционной строительной компании «МАСАТО»

утверждена приказом № 24/18-1 от «28» ноября 2024 г.

Срок сдачи дипломного проекта «10» мая 2025 г.

Исходные данные к проекту:

- задание, полученное при прохождении преддипломной практики;

- методические рекомендации и спецификации по разработке клиент – серверных информационных систем;

- в распоряжении имеется техническое задание, черновые таблицы нагрузки и примеры отчётных документов.

Перечень подлежащих разработке в дипломном проекте вопросов или краткое содержание дипломного проекта:

1. исследование и анализ проблем ручного учёта учебной нагрузки в вузах, выявление ограничений существующих решений;
2. сравнительный анализ современных технологий и фреймворков для разработки веб-приложений (NestJS, Laravel, Django и др.);
3. проектирование и реализация веб-приложения на основе стека React + NodeJS + MySQL;
4. проведение тестирования разработанного программного обеспечения, включая ручное функциональное тестирование, валидацию пользовательских сценариев и анализ устойчивости работы системы;
5. требуется централизованное веб-приложение с разграничением прав доступа (пользователь и администратор), с возможностью использования в локальной и серверной среде.

Рекомендуемая литература:

1. Буч, Гради. Язык UML. Руководство пользователя / Гради Буч, Джеймс Рамбо, Ивар Якобсон. – М.: ДМК Пресс, 2015. – 496 с.
2. NestJS Documentation. URL: <https://docs.nestjs.com/> (был доступен 17 февраля 2025).
3. Docker Inc. Docker Documentation. URL: <https://docs.docker.com/> (был доступен 17 февраля 2025).
4. React Documentation. URL: <https://react.dev/learn> (был доступен 17 февраля 2025).
5. ГОСТ 7.32-2001. Отчёт о научно-исследовательской работе. Структура и правила оформления.

Сроки представления разделов дипломного проекта

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование разделов | Сроки предоставления научному руководителю | Отметка о предоставлении |
| 1 | Исследование и анализ предметной области | 28.03.2025 | выполнено |
| 2 | Исследование современных методов, способов и средств реализации | 10.04.2025 | выполнено |
| 3 | Проектирование и разработка приложения | 20.04.2025 | выполнено |
| 4 | Тестирование и оформление пояснительной записки | 30.04.2025 | выполнено |

Дата выдачи задания «2» декабря 2024 г.

Декан факультета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кегенбеков Ж.К.

(подпись)

Научный руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дадаева И.Г.

(подпись)

Задание принял к исполнению

студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Васильев М.А.

(подпись)

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc199228468)

[1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 8](#_Toc199228469)

[1.1 Основные понятия бизнес-предметной области 8](#_Toc199228470)

[1.2 Основные принципы и методы исследования бизнес предметной области 9](#_Toc199228471)

[1.3 Принципы моделирования 10](#_Toc199228472)

[1.4 Использованные нотации и моделирование 11](#_Toc199228473)

[1.4.1 Диаграммы IDEF0 11](#_Toc199228474)

[1.4.2 Диаграммы DFD (Data Flow Diagram) 12](#_Toc199228475)

[1.4.3 Диаграммы UML 13](#_Toc199228476)

[1.5 Идентификация основных характеристик и специфика бизнес-предметной области 15](#_Toc199228477)

[1.6 Организационная структура бизнес-предметной области 16](#_Toc199228478)

[1.7 Проблема и задачи автоматизации бизнес-предметной области, сравнение с существующими веб-приложениями 17](#_Toc199228479)

[1.8 Логическое и технико-технологическое обоснование возможности и целесообразности автоматизации 18](#_Toc199228480)

[1.9 Итоги и постановка задачи 21](#_Toc199228481)

[2 ИССЛЕДОВАНИЕ СОВРЕМЕННЫХ МЕТОДОВ, СПОСОБОВ И СРЕДСТВ РЕШЕНИЯ ПРОБЛЕМЫ АВТОМАТИЗАЦИИ ИНФОРМАЦИОННЫХ БИЗНЕС-ПРОЦЕССОВ 22](#_Toc199228482)

[2.1 Основные подходы к автоматизации деятельности строительных компаний 22](#_Toc199228483)

[2.2 Обоснование выбора способа реализации 23](#_Toc199228484)

[2.3 Сравнительный анализ возможных решений 24](#_Toc199228485)

[2.4 Технико-экономическое обоснование проекта 25](#_Toc199228486)

[2.4.1 Клиентская часть: выбор React 25](#_Toc199228487)

[2.4.2 Оформление интерфейса 28](#_Toc199228488)

[2.4.3 Серверная часть: архитектура на Express.js 29](#_Toc199228489)

[2.4.4 База данных: выбор PostgreSQL 30](#_Toc199228490)

[2.5 Выбор технических средств для автоматизации 31](#_Toc199228491)

[2.6 Способы проектирования и моделирования 32](#_Toc199228492)

[2.7 Ресурсные затраты и потенциальные выгоды 33](#_Toc199228493)

[3 ПРОЕКТНАЯ ЧАСТЬ 34](#_Toc199228494)

[3.1 Общая характеристика проектируемой автоматизированной информационной системы 34](#_Toc199228495)

[3.2 Архитектура системы 35](#_Toc199228496)

[3.2.1 Аппаратная архитектура 35](#_Toc199228497)

[3.2.2 Телекоммуникационная архитектура 36](#_Toc199228498)

[3.2.3 Архитектура программного обеспечения 37](#_Toc199228499)

[3.3 Проектирование базы данных 38](#_Toc199228500)

[3.3.1 Функциональная декомпозиция (диаграмма IDEF0) 39](#_Toc199228501)

[3.4 Проектирование пользовательского интерфейса 41](#_Toc199228502)

[3.4.1 Интерфейс администратора 41](#_Toc199228503)

[3.4.2 Интерфейс пользователя 44](#_Toc199228504)

[3.5 Безопасность и разграничение прав доступа 49](#_Toc199228505)

[3.6 Моделирование пользовательских сценариев (UML) 50](#_Toc199228506)

[3.6.1 Диаграмма деятельности 50](#_Toc199228507)

[3.7 Выбор программных средств разработки 52](#_Toc199228508)

[4 ТЕСТИРОВАНИЕ И РАЗРАБОТКА 54](#_Toc199228509)

[4.1 Цели и задачи тестирования 54](#_Toc199228510)

[4.3 Результаты и выводы 55](#_Toc199228511)

[4.4 Методы и способы тестирования 55](#_Toc199228512)

[4.5 Результаты тестирования 56](#_Toc199228513)

[4.6 Практическая значимость и ожидаемый эффект 57](#_Toc199228514)

[ЗАКЛЮЧЕНИЕ 59](#_Toc199228515)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 60](#_Toc199228516)

[ПРИЛОЖЕНИЕ А 61](#_Toc199228517)

[ПРИЛОЖЕНИЕ Б 68](#_Toc199228518)

## ВВЕДЕНИЕ

Цифровизация бизнес-процессов становится неотъемлемой частью современного строительного и инвестиционного сектора. Особенно актуально это для компаний, работающих с большим количеством клиентов и арендных помещений. Своевременное управление арендой, прозрачность операций и снижение влияния человеческого фактора – ключевые аспекты, требующие автоматизации.

Дипломный проект направлен на разработку веб-приложения для автоматизации системы аренды помещений в инвестиционно-строительной компании «МАСАТО». Решение позволяет пользователям просматривать доступные объекты, подавать заявки на аренду, а сотрудникам компании – эффективно управлять заявками и помещениями через административную панель.

Актуальность проекта обусловлена необходимостью повышения эффективности и прозрачности арендных процессов, исключения дублирования информации, минимизации ошибок и обеспечения постоянного доступа к актуальным данным для обеих сторон – арендаторов и сотрудников компании.

Объект исследования: процессы управления арендой помещений в инвестиционно-строительной компании.

Предмет исследования: методы и средства автоматизации этих процессов на основе веб-технологий и реляционных баз данных.

Цель работы: разработать систему аренды помещений с разделением прав доступа и функциональностью для обработки заявок, просмотра информации об объектах, управлении клиентами и помещениями.

Для достижения поставленной цели решались следующие задачи:

- анализ предметной области и выявление текущих проблем;

- формализация требований к функциональности;

- выбор технологического стека и обоснование его применения;

- проектирование архитектуры веб-приложения, включая интерфейсы и модель базы данных;

- реализация клиентской и серверной части приложения с использованием React, Node.js, PostgreSQL и REST API;

- проведение тестирования системы и оценка соот тствия заявленным требованиям.

Научная новизна проекта заключается в разработке прикладной системы с учётом конкретных бизнес-потребностей локальной компании, что делает её адаптированной и потенциально масштабируемой для аналогичных организаций.

Практическая значимость работы заключается в возможности немедленного внедрения разработанного решения в реальную инфраструктуру компании «МАСАТО», что приведёт к снижению временных затрат, улучшению обслуживания клиентов и оптимизации работы сотрудников.

**1 АНАЛИТИЧЕСКАЯ ЧАСТЬ**

**1.1 Основные понятия бизнес-предметной области**

В бизнес-предметной области, связанной с деятельностью инвестиционно-строительной компании «МАСАТО», ключевую роль играет организация процесса аренды помещений. В условиях цифровизации и роста требований к качеству обслуживания клиентов всё большую значимость приобретает автоматизация взаимодействия между арендаторами и сотрудниками компании. Основное внимание в рамках рассматриваемой предметной области уделяется обеспечению доступности информации об аренде помещений, удобству подачи и обработке заявок, формированию договоров и ведению истории аренды.

В разработанной системе взаимодействие осуществляется между двумя основными категориями пользователей – администраторами и арендаторами. Арендатор – это пользователь системы, который осуществляет поиск помещений, фильтрацию по заданным параметрам (площадь, этаж, стоимость), подаёт заявку на аренду и отслеживает её статус. Все действия арендатора ограничены просмотром информации и её вводом в рамках предусмотренного функционала. После подтверждения заявки пользователь получает доступ к электронному договору аренды, который может быть скачан и использован в дальнейшем. Все данные, связанные с заявками и договорами, хранятся в централизованной базе данных и отображаются в личном кабинете арендатора.

Администратор представляет собой сотрудника компании, обладающего расширенными правами. Он работает через административную панель, управляет карточками помещений, обрабатывает входящие заявки арендаторов, принимает решение о подтверждении или отклонении заявок, формирует электронные договоры и управляет базой арендаторов. Кроме того, администратор может редактировать и обновлять характеристики помещений, включая их статус (свободно/занято), стоимость, площадь и другие параметры. Его задача – обеспечить актуальность и целостность информации, а также прозрачность всего процесса аренды.

Центральным элементом предметной области является заявка на аренду – цифровой запрос, формируемый арендатором на основе выбранного помещения. Она включает в себя контактные данные пользователя, срок аренды и комментарии. После отправки заявка сохраняется в базе данных и поступает на рассмотрение администратору, который меняет её статус в зависимости от результата проверки. В случае подтверждения заявки автоматически формируется договор аренды, содержащий все необходимые юридические сведения: сведения об участниках, срок действия, арендную плату и прочие параметры. Документ сохраняется в системе и предоставляется пользователю для просмотра и скачивания.

Отдельное место в предметной области занимает каталог помещений – структура, отображающая доступные объекты аренды. Каждое помещение представлено в виде карточки с указанием номера, площади, стоимости, типа, статуса и фотографии. Каталог позволяет пользователям фильтровать объекты по заданным критериям, упрощая процесс поиска подходящего варианта. Все данные о помещениях, пользователях, заявках и договорах хранятся в централизованной реляционной базе данных PostgreSQL. Кроме того, для хранения изображений и PDF-документов используется облачное хранилище (например, AWS S3), что повышает надёжность и масштабируемость системы.

Таким образом, бизнес-предметная область охватывает процессы поиска помещений, подачи и обработки заявок, формирования договоров аренды и управления данными о клиентах и объектах недвижимости. Разграничение прав доступа между арендатором и администратором позволяет минимизировать риски ошибок, упростить административную нагрузку и обеспечить прозрачность всех этапов аренды. Созданная система служит эффективным инструментом автоматизации бизнес-процессов в строительной компании и позволяет значительно повысить качество взаимодействия с клиентами.

# 1.2 Основные принципы и методы исследования бизнес предметной области

Изучение бизнес-предметной области, связанной с автоматизацией аренды помещений в инвестиционно-строительной компании «МАСАТО», проводилось с применением комплекса методов, направленных на детальное понимание текущих процессов и выявление их недостатков. Основной целью анализа стало определение потребностей как сотрудников компании, так и потенциальных арендаторов, а также выявление точек взаимодействия, подлежащих автоматизации.

Одним из ключевых методов стала серия наблюдений и бесед с представителями компании, выполняющими функции администрирования аренды. В ходе таких обсуждений выяснялось, какие задачи выполняются вручную, какие документы используются, как осуществляется обработка заявок и какие сложности возникают при взаимодействии с арендаторами. Особое внимание уделялось вопросам дублирования информации, сложности отслеживания статусов заявок и подготовки договоров. Такие беседы позволили сформировать представление о типичных действиях сотрудников и выявить элементы, требующие оптимизации.

Дополнительно проводился анализ электронных и бумажных документов, используемых в повседневной деятельности: заявок на аренду, договоров, таблиц с данными о помещениях. Эти документы рассматривались с точки зрения структуры, повторяемости полей, уровня стандартизации. Выяснилось, что отсутствие централизованной базы данных ведёт к расхождениям в информации и затрудняет поддержание актуальности сведений. Также были зафиксированы различные шаблоны договоров и заявок, применяемые в зависимости от ситуации, что усложняло процесс обработки.

Особое внимание в исследовании уделялось рассмотрению технической инфраструктуры, необходимой для реализации автоматизированной системы. Анализ проводился с учётом возможностей веб-технологий, включая архитектуру клиент-серверных приложений, использование микросервисов, REST API, а также облачных хранилищ (например, AWS S3). Изучались требования к обеспечению безопасности данных, защите пользовательской информации и разграничению доступа между арендаторами и администраторами [1].

Параллельно применялись методы визуального моделирования процессов – такие как построение DFD-диаграмм, UML-диаграмм, диаграмм классов, прецедентов и активностей. Эти модели позволили структурировать понимание логики системы, связей между сущностями (например, помещение, заявка, пользователь, договор) и жизненного цикла работы с данными – от подачи заявки до формирования договора и хранения истории аренды.

Таким образом, исследование бизнес-предметной области включало в себя беседы с сотрудниками компании, анализ используемых документов, изучение технических и организационных требований, а также моделирование всех ключевых процессов. Все полученные сведения были систематизированы и использованы для формализации требований к системе, выбора архитектурного подхода и последующего проектирования веб-приложения.

# 1.3 Принципы моделирования

При описании процессов, связанных с автоматизацией аренды помещений в компании «МАСАТО», используется комплексный подход к моделированию, охватывающий как структурные, так и поведенческие аспекты системы. Целью моделирования стало формирование целостного и визуально понятного представления о взаимодействии участников, потоке данных и логике работы всех ключевых компонентов веб-приложения. Выбранные методики позволяют точно отразить реальную деятельность компании в виде формализованных схем и диаграмм, что в дальнейшем облегчает проектирование и реализацию программного продукта.

Моделирование бизнес-процессов является ключевым этапом в формализации требований [1].

Одним из основных инструментов визуализации процессов является диаграмма потоков данных (DFD), которая применяется для отображения последовательности операций от подачи заявки арендатора до генерации договора. На таких диаграммах представлены источники данных (пользователь, администратор), процессы обработки (отображение помещений, подача и обработка заявок), а также соответствующие хранилища информации (база заявок, база договоров). Это позволяет увидеть полный путь информации в системе и понять, где происходят её трансформации.

Для детализации и структуризации логики предметной области применяются методы функциональной декомпозиции, в частности методология IDEF5. С помощью этой методики описываются объекты, их атрибуты и взаимосвязи, что особенно важно при проектировании базы данных и бизнес-логики приложения. Например, определяется, какие поля включаются в карточку помещения, как связаны заявки с пользователями и договорами, а также каким образом формируется история аренды. Это даёт возможность формализовать представление о сущностях системы и построить основу для хранения и обработки данных.

Для описания поведения участников системы и их взаимодействий используется нотация UML. Диаграммы прецедентов позволяют представить ключевые сценарии работы арендатора и администратора: регистрация, просмотр помещений, подача заявки, подтверждение, формирование договора и т. д. Также применяются диаграммы активностей и последовательностей, которые детализируют этапы выполнения каждой функции. Эти модели особенно полезны для последующего описания бизнес-логики в терминах API и пользовательских сценариев.

Кроме того, используются диаграммы классов и пакетов, которые описывают структуру программы на уровне сущностей, их полей и методов. Это позволяет отразить архитектурные решения, выбранные при реализации системы, и продемонстрировать связь между модулями: пользовательский интерфейс, бизнес-логика и база данных. Компонентное моделирование дополняет общее представление о физической архитектуре системы, включая распределение нагрузки, хранение файлов и взаимодействие между микросервисами.

Таким образом, принципы моделирования в рамках проекта автоматизации аренды помещений базируются на использовании взаимосвязанных диаграмм DFD, IDEF5 и UML. Они позволяют перейти от изучения бизнес-процессов к точному и структурированному описанию системы, формализовать все этапы работы и подготовить основу для технической реализации. Такое многослойное представление предметной области обеспечивает надёжность проектирования, сокращает вероятность ошибок и делает систему понятной для всех участников разработки.

# 1.4 Использованные нотации и моделирование

В рамках проекта по разработке системы автоматизации аренды помещений для инвестиционно-строительной компании «МАСАТО» были использованы три ключевые нотации: IDEF0, DFD и UML. Каждая из них позволила по-разному отразить архитектуру, логику и взаимодействие компонентов системы, а также участников процесса аренды. Их совместное применение обеспечило полное понимание предметной области, переход от бизнес-анализа к проектированию, и задало основу для разработки веб-приложения.

# 1.4.1 Диаграммы IDEF0

Методология IDEF0, представленная на рисунке 1, применялась для описания функциональной декомпозиции системы, с фокусом на определение входов, выходов, управляющих воздействий и механизмов. На первом уровне была построена контекстная диаграмма A0, отражающая, каким образом веб-приложение обрабатывает входные данные от арендаторов и администратора, и какие результаты выдаёт на выходе.

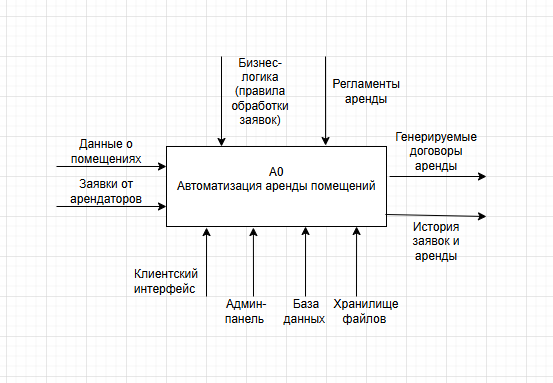


Рисунок 1 – Диаграмма IDEF0 (составлено автором)

В центре диаграммы расположен блок «Автоматизация аренды помещений», описывающий главный процесс системы. Слева указаны входные данные: «Заявки от арендаторов» и «Данные о помещениях», поступающие через пользовательский интерфейс. Сверху отображены управляющие воздействия: «Регламенты аренды» и «Бизнес-логика обработки», формирующие нормативные рамки и правила обработки данных. В нижней части находятся механизмы – технические компоненты системы: React-клиент, административная панель на Node.js, централизованная база данных PostgreSQL и облачное хранилище AWS S3, задействованные в реализации процессов. Справа расположены выходные результаты: «Генерируемые договоры аренды» и «История аренды», отражающие продуктивный итог обработки данных. Такая структура наглядно показывает связи между участниками, информационными потоками и техническими средствами, формируя основу для детальной декомпозиции и проектирования подсистем.

# 1.4.2 Диаграммы DFD (Data Flow Diagram)

Диаграммы потоков данных использовались для описания логики перемещения информации внутри системы. DFD-диаграмма отразила, как данные от арендаторов и администраторов проходят через модули, сохраняются в базе данных, обрабатываются и возвращаются пользователю.

На рисунке 2 выделены четыре ключевых процесса: «Отображение доступных помещений», «Подача заявки», «Обработка заявок» и «Формирование договора». Пользователь (арендатор) выступает инициатором первого процесса – он направляет запрос на просмотр помещений, после чего информация извлекается из хранилища и отображается в виде карточек. Далее пользователь подаёт заявку на аренду, которая сохраняется в базе и переходит в обработку. Администратор, как отдельный субъект системы, проверяет заявку и присваивает ей статус. Подтверждённые заявки поступают в процесс формирования договора, где автоматически создаётся документ и сохраняется в базе договоров. В результате пользователь получает договор, доступный для скачивания в личном кабинете. Все процессы связаны с централизованной базой данных, где хранятся сведения о пользователях, помещениях, заявках и истории аренды. Диаграмма демонстрирует не только движение данных, но и взаимодействие пользователей с системой, подчёркивая важность централизации информации и разделения ролей.

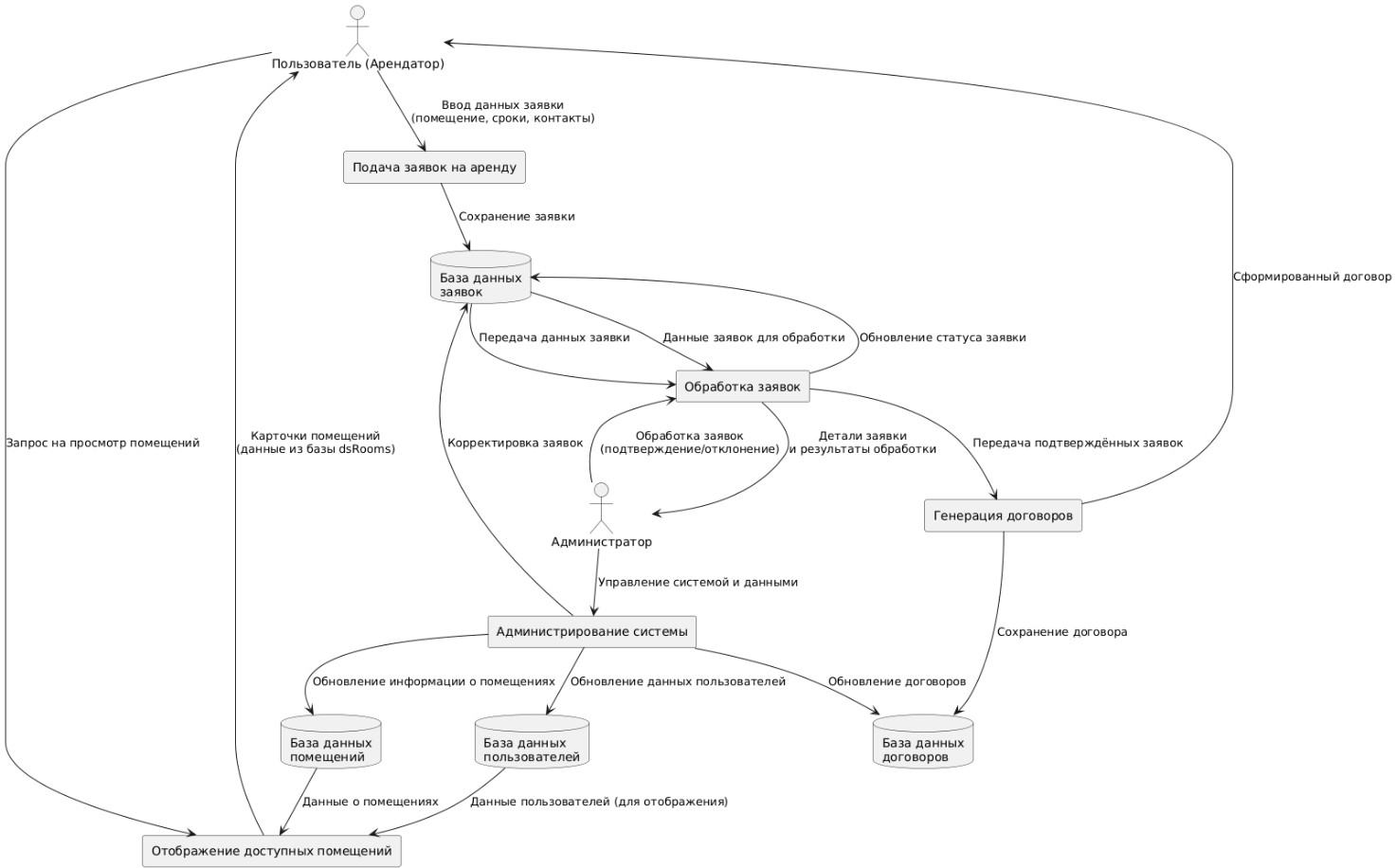


Рисунок 2 – Диаграмма потоков данных (DFD) (составлено автором)

# 1.4.3 Диаграммы UML

Для описания взаимодействий между пользователями системы и её функциональностью применялась нотация UML, в частности – диаграмма прецедентов (Use Case). Эта модель позволила показать, какие действия доступны каждому из типов пользователей и как они связаны с функциональными модулями приложения.

На рисунке 3 представлены два основных актёра: арендатор и администратор. Арендатор взаимодействует с такими прецедентами, как «Регистрация и авторизация», «Просмотр каталога помещений», «Подача заявки на аренду», «Отслеживание статуса» и «Получение договора». Он работает только с уже утверждённой информацией и инициирует процессы, не влияя напрямую на изменения данных. Администратор, в свою очередь, имеет доступ к функциям «Обработка заявок», «Формирование договоров», «Управление помещениями», а также «Редактирование профилей пользователей». Оба типа пользователей объединяются сценарием авторизации. Такой подход позволяет обеспечить надёжное разграничение прав, высокую безопасность и удобство использования системы. Диаграмма также показывает, что арендатор участвует только в подаче и получении информации, тогда как администратор контролирует и наполняет систему данными.

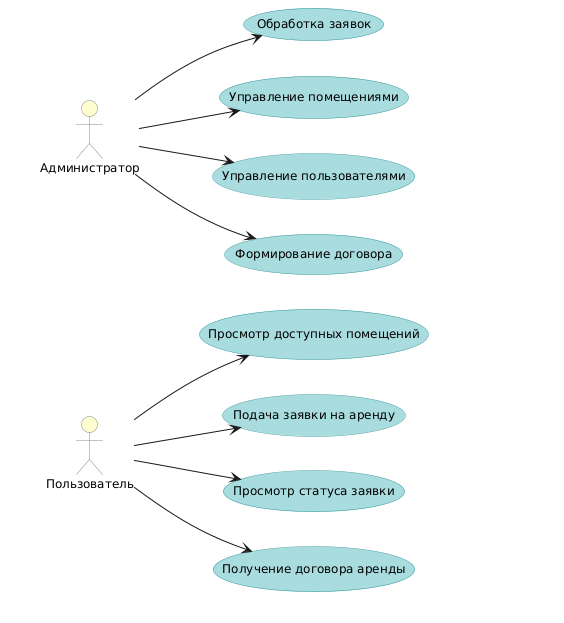


Рисунок 3 – Диаграмма Use Case (составлено автором)

# 1.5 Идентификация основных характеристик и специфика бизнес-предметной области

В бизнес-предметной области, связанной с автоматизацией аренды помещений, центральную роль играет администратор, который управляет данными о помещениях, обрабатывает заявки арендаторов и формирует договоры. Система, построенная для компании «МАСАТО», отражает не только техническую сторону автоматизации, но и организационно-экономические особенности сферы, где каждое помещение рассматривается как ограниченный ресурс, а процесс аренды требует координации между клиентом и представителем компании.

Администратор системы осуществляет весь цикл управления: от добавления карточек помещений до подтверждения заявок и формирования договоров. На основе поступающих от пользователей заявок администратор определяет, какие помещения заняты, какие доступны и какие документы необходимо сформировать. Весь процесс происходит в пределах цифровой среды, где данные сохраняются в централизованной базе, а взаимодействие пользователей с системой строго регламентировано. Пользователь (арендатор), в свою очередь, играет роль потребителя услуги – он подаёт заявки, отслеживает их статус и получает итоговые документы, не вмешиваясь в логику управления недвижимостью. Такое разграничение полномочий упрощает использование системы и снижает вероятность ошибок.

Особенностью данной предметной области является то, что система аренды помещений тесно связана с распределением ограниченных ресурсов – помещений, времени и персонала. Каждый объект недвижимости имеет фиксированные характеристики: площадь, этаж, стоимость, тип и статус (свободен или занят). Это делает систему аналогичной складской или ресурсной модели учёта, где необходимо вести строгую фиксацию доступных и занятых объектов. Администратор, как управляющее звено, фактически выполняет функции логиста и экономиста, обеспечивая равномерную и своевременную загрузку помещений в соответствии с запросами клиентов.

Экономический аспект проявляется в необходимости эффективного распределения арендуемых объектов, оптимизации заполнения площадей и минимизации времени простоя. Все заявки фиксируются и отслеживаются, что обеспечивает прозрачность процессов, а формирование договоров и хранение истории аренды создаёт юридическую обоснованность каждой сделки. Благодаря цифровизации, бумажная нагрузка сотрудников сводится к минимуму, а пользователь получает быстрый и надёжный доступ к услугам компании.

Техническая архитектура системы также отражает специфику бизнес-процессов: веб-приложение состоит из клиентской части (для арендаторов) и административной панели (для сотрудников компании), объединённых через REST API. Хранение и обработка данных обеспечивается с помощью PostgreSQL, а медиафайлы и договоры – через облачные хранилища (например, AWS S3). Это создаёт единую информационную среду, в которой вся деятельность упорядочена и документирована.

Таким образом, специфика предметной области заключается в сочетании коммерческой и административной логики. Система ориентирована на оптимальное использование ресурсов компании, снижение издержек, упрощение коммуникации с клиентом и надёжную фиксацию всех этапов взаимодействия. В совокупности это позволяет рассматривать систему не только как программный продукт, но и как инструмент повышения эффективности бизнеса в сфере недвижимости и аренды.

# 1.6 Организационная структура бизнес-предметной области

В рамках бизнес-предметной области, связанной с автоматизацией аренды помещений в компании «МАСАТО», ключевую роль играют две основные категории пользователей – администраторы и арендаторы, каждая из которых обладает строго определёнными функциями и уровнем доступа. Такая структура построена на принципах разделения ответственности и контроля за ресурсами, где каждый участник выполняет свою часть задач, не выходя за рамки полномочий, что обеспечивает устойчивость системы и исключает конфликт интересов.

Администратор – это сотрудник компании, на которого возложены задачи по обработке всей информации, связанной с арендой. Он занимается добавлением и редактированием данных о помещениях, формирует карточки с указанием параметров (номер, этаж, площадь, стоимость, статус), управляет заявками от пользователей и составляет договоры. Все действия администратора влияют на основное хранилище данных и отражаются на интерфейсе, доступном арендаторам. При этом система предусматривает строгую логику: каждое помещение должно соответствовать определённым параметрам, а каждая заявка – пройти проверку и подтверждение. Если возникают конфликты в заявках, ошибки в заполнении или неточное сопоставление по статусу помещения, администратор выполняет корректировку или связывается с пользователем для уточнения.

Арендаторы – это зарегистрированные пользователи, которые взаимодействуют с системой через публичный интерфейс. Они могут просматривать доступные помещения, фильтровать их по параметрам и подавать заявки на аренду. После отправки заявки они отслеживают её статус, получают доступ к сгенерированному договору, и при необходимости загружают его. При этом арендатор не может изменять информацию о помещениях или заявках – он лишь получает финальные данные, сформированные администратором. Такое разграничение ролей позволяет упростить взаимодействие и исключить вмешательство пользователей в процессы управления недвижимостью.

Особенностью предметной области является иерархия и распределение функциональности в зависимости от роли: администратор выполняет функции координатора, логиста и оператора данных, тогда как арендатор является потребителем, выполняющим базовые действия по выбору помещения и получению договора. В дальнейшем, по мере развития проекта, возможно расширение и детализация ролей, включая, например, юристов (для согласования договоров) или технических специалистов (для обслуживания цифровой инфраструктуры).

# 1.7 Проблема и задачи автоматизации бизнес-предметной области, сравнение с существующими веб-приложениями

В рамках текущей деятельности инвестиционно-строительной компании «МАСАТО» значительная часть операций, связанных с арендой помещений, осуществляется вручную либо с использованием несвязанных между собой файлов и инструментов. Это приводит к ряду организационных и технических проблем. Администраторы компании вынуждены повторно вводить одни и те же данные при изменении статуса помещений, обработке заявок или формировании договоров. Нередко возникают ситуации, когда в разных таблицах указана противоречивая информация, а арендаторы получают устаревшие сведения о доступности объектов или сроках аренды.

Отсутствие единой централизованной базы данных усложняет управление: при обновлении информации необходимо вносить правки сразу в несколько документов, что требует дополнительного времени и приводит к риску ошибок. Пользователь, подавший заявку, не имеет возможности отследить её статус в реальном времени и часто вынужден обращаться в офис или ожидать ответ по электронной почте. Договоры аренды формируются вручную, что усложняет процесс, особенно в условиях высокой нагрузки или при работе с большим количеством арендаторов. Хранение данных в виде локальных файлов затрудняет аналитику, историю аренды и восстановление информации при возникновении спорных ситуаций.

Введение системы автоматизации аренды помещений направлено на устранение указанных проблем за счёт создания единого цифрового пространства. Все данные – о помещениях, арендаторах, заявках и договорах – хранятся централизованно в базе PostgreSQL, что позволяет избежать дублирования информации и повысить её достоверность. Администратор вносит сведения о помещениях и обрабатывает заявки в административной панели, после чего вся информация автоматически обновляется в пользовательском интерфейсе. Арендатор в свою очередь получает прозрачный доступ к актуальным данным: может отследить статус заявки, скачать договор и просмотреть историю аренды.

Система обеспечивает автоматическую генерацию PDF-документов – договоров и сопроводительных форм – которые могут быть скачаны или отправлены по электронной почте. Это значительно снижает нагрузку на сотрудников и упрощает документооборот. Все изменения фиксируются, что исключает несанкционированную правку и повышает юридическую надёжность процессов. Кроме того, административный интерфейс предусматривает контроль за статусами помещений, нормативами по загрузке, а также возможностью фильтрации объектов по параметрам, что делает работу более управляемой и аналитически обоснованной.

Сравнение с существующими решениями, такими как CRM-системы общего назначения или стандартные формы учёта в Excel, показывает, что они не учитывают специфики арендного процесса и не обеспечивают нужной гибкости. Большинство таких систем ориентированы на обработку сделок или клиентов, но не предполагают управления физическими объектами с их индивидуальными характеристиками (площадь, этаж, статус, фото). Кроме того, они не включают готовые модули для юридической генерации договоров аренды или истории аренды в структурированном виде.

Таким образом, предлагаемая система автоматизации является необходимым шагом для повышения эффективности внутреннего документооборота, качества обслуживания клиентов и снижения трудозатрат на рутинные процессы. Её внедрение позволяет не только ускорить обработку заявок, но и обеспечить прозрачность, единообразие данных и высокую скорость формирования всех необходимых документов как для арендаторов, так и для администрации.

# 1.8 Логическое и технико-технологическое обоснование возможности и целесообразности автоматизации

Автоматизация сегмента, связанного с управлением арендой помещений в инвестиционно-строительной компании «МАСАТО», представляется логически обоснованной и технически реализуемой задачей. В существующей модели взаимодействия значительная часть операций выполняется вручную, через неструктурированные документы, что порождает дублирование информации, несогласованность между отделами и избыточные трудозатраты на рутинные действия. Внедрение автоматизированной системы позволяет устранить эти недостатки за счёт использования единого хранилища данных и цифровой обработки заявок, договоров и историй аренды. С позиции управления процессом аренды, компания сталкивается с постоянной необходимостью в актуализации данных о помещениях, отслеживании заявок, контроле их обработки и формировании юридически значимых документов. Без централизованной платформы каждое изменение требует ручного редактирования и уведомления клиентов, что создаёт риски задержек и ошибок. Автоматизация этих процедур с использованием веб-приложения, построенного на современных и доступных технологиях (React, Node.js, PostgreSQL), устраняет проблему многократного ввода данных и обеспечивает синхронность между пользовательским интерфейсом и административной панелью. Технические предпосылки для реализации проекта уже имеются: базовая серверная инфраструктура, наличие доступа к интернету, использование открытых программных решений, не требующих лицензирования.

Реализация веб-интерфейса позволяет арендаторам регистрироваться, подавать заявки, отслеживать их статус и скачивать договоры, в то время как сотрудники компании через админ-панель управляют данными, подтверждают заявки и редактируют информацию. Простота интерфейса и логика взаимодействия делают систему доступной даже для пользователей без специальной технической подготовки. Таким образом, автоматизация этого сегмента полностью оправдана: она способствует повышению прозрачности процессов, уменьшает количество ошибок и ускоряет документооборот. Система предоставляет участникам актуальные сведения в режиме реального времени, снижает административную нагрузку и повышает лояльность клиентов за счёт улучшенного сервиса.

Сегмент бизнес-предметной области, подлежащий автоматизации, логично разделяется на четыре функциональных компонента, которые обеспечивают полный цикл управления арендой помещений. Первый – модуль учёта помещений – отвечает за создание и редактирование карточек помещений с указанием их характеристик, таких как номер, этаж, площадь, стоимость, статус и тип. Администратор добавляет новые объекты, обновляет информацию и изменяет статусы, отражающие текущую доступность. Второй – модуль обработки заявок – реализует логику подачи, хранения и обработки заявок от арендаторов. Пользователь подаёт заявку через интерфейс, после чего она поступает в систему и обрабатывается администратором, который меняет статус и уведомляет клиента. Третий – модуль генерации договоров – формирует PDF-документы на основании подтверждённой заявки; договор сохраняется в облачное хранилище (например, AWS S3) и становится доступен пользователю в личном кабинете, также возможна отправка документа по электронной почте. Четвёртый – модуль управления доступом – обеспечивает разграничение прав между администраторами и арендаторами. Пользователи проходят регистрацию и авторизацию, после чего получают доступ только к релевантным функциям: арендаторы – к просмотру и подаче заявок, администраторы – к управлению данными и формированию документов. Все модули взаимодействуют через единую базу данных PostgreSQL и REST API, что обеспечивает согласованность данных и расширяемость системы. Такая структура позволяет легко масштабировать решение, подключать дополнительные функции (например, аналитику, интеграцию с электронной подписью) и гарантирует надёжную работу при росте числа пользователей и объёма данных. Автоматизация сегмента, связанного с управлением арендой помещений в инвестиционно-строительной компании «МАСАТО», представляется логически обоснованной и технически реализуемой задачей.

В существующей модели взаимодействия значительная часть операций выполняется вручную, через неструктурированные документы, что порождает дублирование информации, несогласованность между отделами и избыточные трудозатраты на рутинные действия. Внедрение автоматизированной системы позволяет устранить эти недостатки за счёт использования единого хранилища данных и цифровой обработки заявок, договоров и историй аренды.

С позиции управления процессом аренды, компания сталкивается с постоянной необходимостью в актуализации данных о помещениях, отслеживании заявок, контроле их обработки и формировании юридически значимых документов. Без централизованной платформы каждое изменение требует ручного редактирования и уведомления клиентов, что создаёт риски задержек и ошибок. Автоматизация этих процедур с использованием веб-приложения, построенного на современных и доступных технологиях (React, Node.js, PostgreSQL), устраняет проблему многократного ввода данных и обеспечивает синхронность между пользовательским интерфейсом и административной панелью. Технические предпосылки для реализации проекта уже имеются: базовая серверная инфраструктура, наличие доступа к интернету, использование открытых программных решений, не требующих лицензирования. Реализация веб-интерфейса позволяет арендаторам регистрироваться, подавать заявки, отслеживать их статус и скачивать договоры, в то время как сотрудники компании через админ-панель управляют данными, подтверждают заявки и редактируют информацию. Простота интерфейса и логика взаимодействия делают систему доступной даже для пользователей без специальной технической подготовки. Таким образом, автоматизация этого сегмента полностью оправдана: она способствует повышению прозрачности процессов, уменьшает количество ошибок и ускоряет документооборот.

Система предоставляет участникам актуальные сведения в режиме реального времени, снижает административную нагрузку и повышает лояльность клиентов за счёт улучшенного сервиса. Сегмент бизнес-предметной области, подлежащий автоматизации, логично разделяется на четыре функциональных компонента, которые обеспечивают полный цикл управления арендой помещений. Первый – модуль учёта помещений – отвечает за создание и редактирование карточек помещений с указанием их характеристик, таких как номер, этаж, площадь, стоимость, статус и тип. Администратор добавляет новые объекты, обновляет информацию и изменяет статусы, отражающие текущую доступность. Второй – модуль обработки заявок – реализует логику подачи, хранения и обработки заявок от арендаторов. Пользователь подаёт заявку через интерфейс, после чего она поступает в систему и обрабатывается администратором, который меняет статус и уведомляет клиента. Третий – модуль генерации договоров – формирует PDF-документы на основании подтверждённой заявки; договор сохраняется в облачное хранилище (например, AWS S3) и становится доступен пользователю в личном кабинете, также возможна отправка документа по электронной почте. Четвёртый – модуль управления доступом – обеспечивает разграничение прав между администраторами и арендаторами. Пользователи проходят регистрацию и авторизацию, после чего получают доступ только к релевантным функциям: арендаторы – к просмотру и подаче заявок, администраторы – к управлению данными и формированию документов.

Все модули взаимодействуют через единую базу данных PostgreSQL и REST API, что обеспечивает согласованность данных и расширяемость системы. Такая структура позволяет легко масштабировать решение, подключать дополнительные функции (например, аналитику, интеграцию с электронной подписью) и гарантирует надёжную работу при росте числа пользователей и объёма данных.

# 1.9 Итоги и постановка задачи

В завершение рассмотрения бизнес-предметной области, выявленных проблем и текущих процессов становится очевидным, что без внедрения централизованной автоматизированной системы компания «МАСАТО» сталкивается с рядом существенных ограничений. Ручное ведение заявок, редактирование информации о помещениях в разрозненных источниках, а также необходимость вручную формировать договоры аренды создают дополнительную нагрузку на сотрудников, увеличивают риск ошибок и снижают скорость обслуживания клиентов. Арендаторы, в свою очередь, лишены оперативного доступа к информации об актуальных объектах и не могут в реальном времени отслеживать статус своих заявок или быстро получить необходимые документы. Всё это указывает на необходимость создания интегрированного цифрового решения.

Технические предпосылки для реализации проекта уже имеются: серверная инфраструктура, доступ к сети интернет и возможность использовать открытые технологии без значительных финансовых вложений. Это позволяет сформулировать задачу автоматизации как создание веб-приложения, предназначенного для централизованного управления арендой. Такое приложение должно позволять администраторам добавлять и редактировать карточки помещений, обрабатывать заявки арендаторов, формировать договоры аренды и управлять пользовательскими учётными записями. В то же время арендаторы должны получать возможность регистрироваться, просматривать доступные объекты, подавать заявки, отслеживать их статус, загружать готовые договоры и, при необходимости, пересылать документы по электронной почте.

Цель автоматизации заключается в том, чтобы заменить устаревшие ручные процедуры на единый цифровой процесс, обеспечить прозрачность взаимодействия между участниками, повысить точность и снизить нагрузку на персонал компании. В результате внедрения автоматизированной информационной системы значительно снижается вероятность дублирования данных, ошибок ввода и несогласованности между различными источниками информации. Благодаря централизованному хранению данных и синхронизации между модулями системы, информация обновляется в режиме реального времени и всегда остаётся актуальной как для арендаторов, так и для администраторов.

Это приводит к существенному ускорению документооборота – заявки обрабатываются быстрее, договоры формируются автоматически, а доступ к документам и информации осуществляется без задержек. Сотрудники компании получают удобный инструмент.

**2 ИССЛЕДОВАНИЕ СОВРЕМЕННЫХ МЕТОДОВ, СПОСОБОВ И СРЕДСТВ РЕШЕНИЯ ПРОБЛЕМЫ АВТОМАТИЗАЦИИ ИНФОРМАЦИОННЫХ БИЗНЕС-ПРОЦЕССОВ**

**2.1 Основные подходы к автоматизации деятельности строительных компаний**

Автоматизация бизнес-процессов в строительных компаниях играет важную роль в повышении эффективности работы, улучшении взаимодействия с клиентами и снижении операционных затрат. Современные подходы к автоматизации включают внедрение специализированных информационных систем, которые решают задачи управления данными о строительных объектах, заявках и договорах аренды. Как отмечает А. И. Борисов, автоматизация бизнес-процессов ведёт к снижению транзакционных издержек и росту прозрачности [3].

Одним из ключевых направлений автоматизации является централизация данных, позволяющая использовать единую базу данных для хранения информации о помещениях, заявках и договорах. Это исключает дублирование данных и обеспечивает их актуальность. Также важным направлением является упрощение взаимодействия через внедрение веб-приложений с удобным пользовательским интерфейсом, которые предоставляют клиентам и сотрудникам компании возможность работать с данными в режиме реального времени. Кроме того, интеграция бизнес-процессов, таких как регистрация заявок, обработка данных и генерация договоров, в единую систему способствует повышению прозрачности и скорости выполнения задач [4].

На текущий момент в компании «МАСАТО» полностью отсутствует автоматизация процессов, связанных с управлением арендой. Все операции выполняются вручную, что приводит к ряду сложностей. Среди наиболее существенных проблем можно выделить следующие: высокая вероятность ошибок при работе с данными, трудности с оперативным обновлением информации о статусе помещений, отсутствие прозрачности процессов для клиентов и сотрудников. Эти факторы создают дополнительные временные затраты и снижают общую эффективность деятельности компании.

Отсутствие автоматизации также ограничивает возможности компании в части расширения клиентской базы и оптимизации внутренних процессов. Например, сотрудники вынуждены тратить значительное время на выполнение рутинных задач, таких как ведение записей и обработка заявок, вместо того чтобы сосредоточиться на стратегически важных направлениях. Кроме того, клиенты сталкиваются с неудобствами, связанными с получением актуальной информации о доступных помещениях или статусе заявок.

С точки зрения технико-технологических аспектов автоматизации, одним из приоритетных шагов является внедрение веб-приложения, которое будет включать в себя публичный сайт для пользователей-арендаторов и административную панель для сотрудников компании. Такое решение позволит централизовать управление данными, обеспечить удобный доступ к информации для всех участников процесса и значительно снизить риски, связанные с человеческим фактором.

Для реализации подобной системы необходимо использовать современные технологии, такие как React для разработки клиентской части, Node.js с Express.js для серверной логики и PostgreSQL для хранения данных. Эти инструменты обеспечивают гибкость, масштабируемость и надежность системы, а также позволяют интегрировать её с другими сервисами, такими как электронная подпись или облачные хранилища.

**2.2 Обоснование выбора способа реализации**

В процессе выбора способа реализации системы автоматизации управления арендой для компании «МАСАТО» были рассмотрены три основных подхода: приобретение готового программного обеспечения, кастомизация существующих решений и разработка системы с нуля. Анализ каждого из этих вариантов выявил их преимущества и недостатки, что позволило обоснованно выбрать оптимальный путь для компании.​

Приобретение готового программного обеспечения предполагает использование уже разработанных решений, таких как «1С: Аренда и управление недвижимостью», «УСУ» или «БИТ.Строительство/Аренда». Эти системы предлагают базовый функционал для учета аренды и управления недвижимостью. Однако они часто ориентированы на широкий круг пользователей и могут не учитывать специфические потребности компании «МАСАТО». Кроме того, внедрение таких решений может потребовать значительных затрат на лицензии и обучение персонала.​

Кастомизация существующих решений предусматривает адаптацию готового программного обеспечения под нужды компании. Этот подход может быть эффективным, если базовая система близка к требованиям бизнеса. Однако в случае компании «МАСАТО» отсутствуют решения, которые бы полностью соответствовали её специфике. Кастомизация может потребовать значительных ресурсов и времени, а также зависеть от возможностей и поддержки разработчика исходной системы.​

Разработка системы с нуля предоставляет возможность создать решение, полностью соответствующее требованиям и бизнес-процессам компании. Это позволяет учесть все особенности работы «МАСАТО», обеспечить гибкость и масштабируемость системы. Несмотря на более высокие первоначальные затраты и длительный срок разработки, этот подход обеспечивает наилучшее соответствие системы потребностям компании и снижает зависимость от сторонних разработчиков.​

Учитывая отсутствие готовых решений, полностью удовлетворяющих требованиям компании «МАСАТО», а также необходимость учета специфики её бизнес-процессов, было принято решение о разработке собственной системы автоматизации управления арендой. Это позволит создать эффективный инструмент, способствующий повышению производительности и конкурентоспособности компании.

**2.3 Сравнительный анализ возможных решений**

В процессе выбора оптимального технического решения для автоматизации управления арендой в компании «МАСАТО» был проведён анализ современных технологий, учитывая специфику деятельности компании и её потребности.​

Для разработки пользовательского интерфейса были рассмотрены фреймворки React.js и Vue.js. React.js предоставляет высокую производительность и широкие возможности для создания интерактивных интерфейсов, что делает его популярным выбором для крупных веб-приложений. Vue.js, в свою очередь, отличается простотой освоения и лёгкостью интеграции, что может быть преимуществом при ограниченных ресурсах. Оба фреймворка активно используются в разработке приложений для аренды недвижимости, обеспечивая адаптивность и удобство использования.​

Среди серверных технологий были рассмотрены Node.js, Django и Laravel.

Node.js обеспечивает высокую производительность и масштабируемость, особенно при работе с большим количеством одновременных запросов, что актуально для систем управления арендой.

Django, основанный на языке Python, предлагает быстрый старт разработки и встроенные инструменты для администрирования, что может ускорить процесс создания приложения.

Laravel, фреймворк на PHP, предоставляет элегантный синтаксис и широкие возможности для построения веб-приложений, включая встроенные механизмы аутентификации и маршрутизации.​

Для хранения данных были рассмотрены PostgreSQL и MySQL. PostgreSQL отличается высокой надёжностью, поддержкой сложных запросов и расширяемостью, что делает его подходящим выбором для корпоративных систем. MySQL является популярной СУБД с широким сообществом и обширной документацией, обеспечивая стабильную работу и простоту использования.​

Для размещения приложения и обеспечения его доступности были рассмотрены облачные сервисы AWS, Google Cloud и Azure. AWS предлагает широкий спектр услуг и высокую надёжность, что делает его популярным выбором для масштабируемых приложений. Google Cloud предоставляет мощные аналитические инструменты и интеграцию с другими сервисами Google, что может быть полезно для анализа данных аренды. Azure обеспечивает тесную интеграцию с продуктами Microsoft, что может быть преимуществом при использовании других решений этой компании.​

Учитывая специфику компании «МАСАТО», отсутствие готовых решений, полностью удовлетворяющих её требованиям, а также необходимость учёта особенностей бизнес-процессов, было принято решение о разработке собственной системы автоматизации управления арендой. Это позволит создать эффективный инструмент, способствующий повышению производительности и конкурентоспособности компании.

**2.4 Технико-экономическое обоснование проекта**

Технико-экономическое обоснование проекта разработки веб-приложения для компании «МАСАТО» направлено на оценку целесообразности инвестиций в создание собственной системы автоматизации управления арендой. В условиях отсутствия готовых решений, полностью удовлетворяющих специфике деятельности компании, разработка индивидуального программного продукта представляется наиболее рациональным вариантом.​

Расчёт затрат на разработку веб-приложения включает следующие основные статьи расходов: заработная плата исполнителей, включая оплату труда программистов, дизайнеров, тестировщиков и других специалистов, участвующих в проекте; накладные расходы, такие как аренда помещений, коммунальные услуги, амортизация оборудования и другие сопутствующие затраты; прочие расходы, включая приобретение необходимых программных лицензий, обучение персонала, маркетинговые мероприятия и т.д.​

Ожидаемые преимущества от реализации проекта включают повышение эффективности бизнес-процессов за счёт автоматизации управления арендой, что позволит сократить время на обработку заявок, снизить количество ошибок и повысить прозрачность операций; улучшение клиентского сервиса посредством внедрения веб-приложения с удобным интерфейсом для клиентов, возможностью онлайн-бронирования и отслеживания статуса заявок; маркетинговые выгоды, связанные с наличием современной системы управления арендой, что повысит имидж компании, привлечёт новых клиентов и укрепит доверие существующих; экономию ресурсов благодаря снижению операционных затрат и более рациональному использованию ресурсов компании.

**2.4.1 Клиентская часть: выбор React**

Клиентская часть информационной системы «МАСАТО» реализована с использованием библиотеки React – одного из наиболее популярных инструментов JavaScript для построения интерфейсов современных одностраничных приложений (SPA). Выбор React обусловлен его высокой производительностью, модульной архитектурой, поддержкой сообщества и отличной интеграцией с другими технологиями, входящими в стек разработки системы (Node.js, REST API, PostgreSQL и др.).

Архитектура SPA особенно актуальна в контексте корпоративного портала и административной панели компании, поскольку позволяет осуществлять навигацию между страницами, управление заявками, работу с формами и договорами без перезагрузки страниц, что обеспечивает высокую отзывчивость и улучшает пользовательский опыт как арендаторов, так и администраторов [5].

Основным преимуществом React является его компонентный подход – интерфейс разбивается на независимые, переиспользуемые блоки: формы подачи заявок, карточки помещений, защищённые маршруты, интерфейсы личного кабинета, модальные окна и т.д. Это упрощает разработку, сопровождение и масштабирование интерфейса, а также позволяет обеспечивать единый стиль и поведение компонентов на всём сайте.

В рамках реализации проекта использовались и дополнительные библиотеки из экосистемы React:

- React Router (react-router-dom) – для организации маршрутизации и управления доступом к различным страницам в зависимости от роли пользователя;

- Axios – для обмена данными с backend-частью системы через REST API;

- React Hook Form – для управления состоянием форм и валидации пользовательского ввода;

- Redux – для централизованного управления состоянием приложения.

Одним из важнейших аспектов клиентской части является реализация механизма аутентификации и управления доступом. Для этих целей в системе используется контекст авторизации, реализованный с помощью API React Context. Он обеспечивает централизованное хранение и распространение информации об авторизованном пользователе по всему приложению.

На рисунке 4 представлен основной код контекста. В данном фрагменте создаётся контекст AuthContext, который используется для предоставления состояния авторизации (токен, роль пользователя и его идентификатор) всем компонентам React-приложения, независимо от их вложенности. Контекст оборачивает всё приложение внутри компонента AuthProvider, предоставляя глобальный доступ к информации о текущем пользователе.

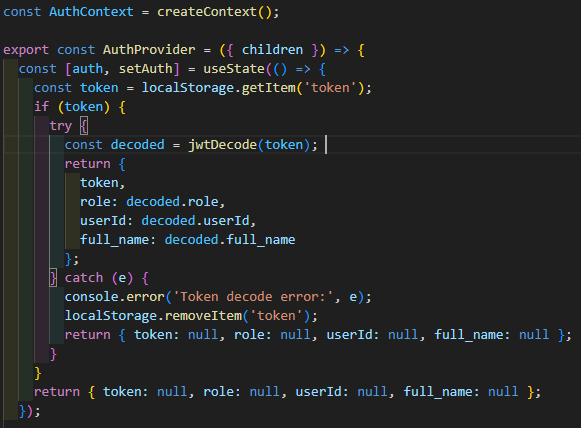


Рисунок 4 – Компонент AuthContext (составлено автором)

Инициализация состояния auth происходит с помощью useState и функции инициализации, которая позволяет прочитать токен из локального хранилища браузера (localStorage). Если токен найден, он декодируется с помощью функции jwtDecode(token).

Функция jwtDecode(token) используется для декодирования JSON Web Token (JWT) – стандартизированного формата передачи информации между сторонами в зашифрованном виде. Внутри JWT хранится полезная нагрузка (payload), содержащая, в частности, сведения о роли пользователя и его идентификаторе (рисунок 5). Функция jwtDecode, предоставляемая библиотекой jwt-decode, безопасно извлекает эти данные, не обращаясь к серверу. В результате, из токена извлекаются значения role и userId, которые далее используются в логике отображения интерфейса и разграничения доступа (например, для отображения административных разделов только для пользователей с ролью администратора).

Если токен отсутствует или недействителен, состояние auth инициализируется значениями null, что соответствует неавторизованному пользователю.

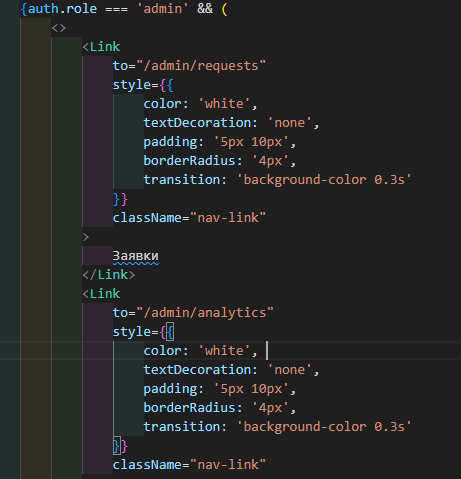


Рисунок 5 – Компонент навигации в Navbar.js (составлено автором)

Таким образом, пользователь (арендатор) видит только клиентские страницы, а администратор – страницы с расширенным функционалом.

Также в проекте реализованы компоненты форм – RequestFormPage.js, AddRoomForm.js, которые обеспечивают удобный ввод данных, проверку корректности и мгновенную обратную связь с пользователем. Все формы взаимодействуют с backend-сервером через REST API и отображают уведомления о статусе отправки данных. В приложении Б продемонстрирован полный код RequestFormPage.js.

Благодаря использованию React, клиентская часть веб-приложения МАСАТО получилась модульной, адаптивной, удобной и безопасной для всех категорий пользователей. Поддержка современного JavaScript, гибкая маршрутизация, наличие защищённых маршрутов, удобные формы и масштабируемость делают React оптимальным выбором для корпоративных информационных систем такого типа.

### 2.4.2 Оформление интерфейса

При разработке клиентской части веб-приложения МАСАТО особое внимание уделялось не только функциональности, но и визуальной составляющей интерфейса. Для стилизации компонентов был выбран гибридный подход, основанный на использовании встроенных стилей на JavaScript (JSS) и токенов централизованного проектирования. Такой подход позволил нам обеспечить баланс между скоростью разработки, гибкостью управления стилями и удобством сопровождения проекта [6].

Онлайн-стилизация предполагает определение CSS-свойств непосредственно в JSX-коде компонентов React. Это решение обеспечивает локальную область видимости стилей, исключает конфликты имён классов и позволяет динамически изменять внешний вид элементов в зависимости от состояния компонента или переданных параметров (рисунок 6).

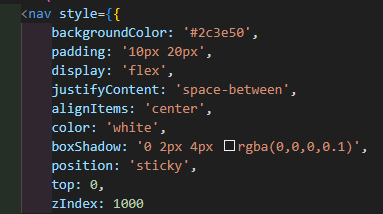
****

Рисунок 6 – Пример использования в React-компоненте (составлено автором)

Такой способ стилизации позволяет разработчику сосредоточиться на логике компонента без необходимости переключения между JSX и отдельными CSS-файлами, что особенно эффективно в условиях индивидуальной или малокомандной разработки.

Цветовая палитра, параметры отступов и другие свойства описываются через JavaScript-константы (рисунок 7).

****

Рисунок 7 – Пример использования в React-компоненте (составлено автором)

### 2.4.3 Серверная часть: архитектура на Express.js

Серверная часть информационной системы автоматизации аренды помещений компании «МАСАТО» реализована с использованием фреймворка Express.js, входящего в экосистему платформы Node.js. Выбор данного инструмента обусловлен его высокой производительностью, гибкостью настройки и широкими возможностями по организации REST-ориентированных веб-сервисов. Express.js представляет собой минималистичное и расширяемое решение, которое позволяет эффективно выстраивать серверную логику как для малых, так и для масштабируемых корпоративных приложений. Node.js и Express позволяют реализовать RESTful API с высокой пропускной способностью [7].

В контексте проектирования серверной архитектуры особое внимание уделялось модульности и логической декомпозиции. Это позволяет обеспечить надёжность, повторное использование кода и удобство сопровождения проекта на всех этапах жизненного цикла системы. Благодаря широкому сообществу и богатой экосистеме промежуточных компонентов (middleware), Express.js предоставляет все необходимые инструменты для быстрой реализации обработки HTTP-запросов, интеграции с базой данных PostgreSQL, а также реализации аутентификации и авторизации пользователей.

Архитектура backend-части системы построена по принципу разделения ответственности, с выделением следующих логических слоёв:

- маршруты (routes) – отвечают за регистрацию REST-эндпоинтов и перенаправление запросов к соответствующим контроллерам. Для каждого ключевого функционального блока определены отдельные модули маршрутов, такие как authRoutes.js, roomRoutes.js, adminRoutes.js;

- контроллеры (controllers) – реализуют бизнес-логику обработки входящих запросов, включая проверку прав доступа, работу с базой данных и формирование ответов клиенту. Примеры таких модулей – authController.js, adminController.js;

- сервисы (services) – взаимодействуют с базой данных PostgreSQL через пул соединений, выполняя SQL-запросы, агрегацию данных и трансформацию полученных результатов;

- модели (models) – представляют структуры данных, обмениваемых между слоями, включая описания сущностей системы: пользователей, помещений, заявок, договоров аренды.

Данная архитектура соответствует принципам «чистой архитектуры» и обеспечивает изоляцию логики приложения от конкретных технологий хранения и передачи данных.

Приведён метода контроллера getAllRequests, реализующего логику получения всех заявок на аренду на рисунке 8. Данный эндпоинт доступен только пользователям с ролью администратора и является частью модуля adminController.js.

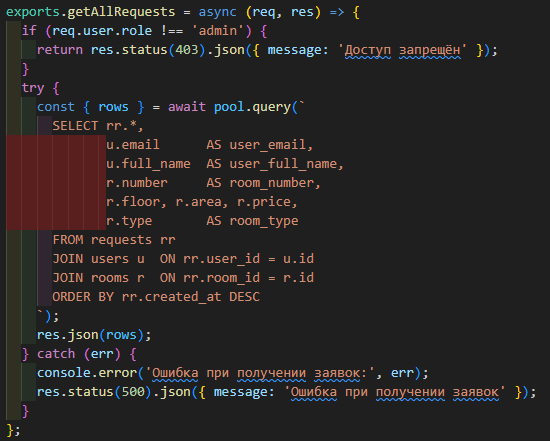


Рисунок 8 – Пример метода контроллера getAllRequests (составлено автором)

# 2.4.4 База данных: выбор PostgreSQL

Для хранения и управления данными в информационной системе «МАСАТО» была выбрана система управления базами данных PostgreSQL – современная объектно-реляционная СУБД с открытым исходным кодом. PostgreSQL обладает высокой стабильностью, поддержкой транзакций, расширяемостью и полной реализацией стандартов SQL, что делает её одним из наиболее надёжных, стабильных решений для корпоративных информационных систем [7].

Выбор PostgreSQL был обусловлен рядом факторов:

- надёжность и стабильность при эксплуатации в production-среде;

- расширяемость – возможность добавления собственных функций, типов данных и индексов;

- совместимость с Node.js/Express через pg-драйвер или ORM).

На рисунке 9 представлена архитектура базы данных была спроектирована на основе ключевых сущностей предметной области: пользователи, помещения, заявки, договоры аренды, уведомления и платежи. Каждая сущность представлена отдельной таблицей с чётко определёнными полями, ограничениями целостности и связями.

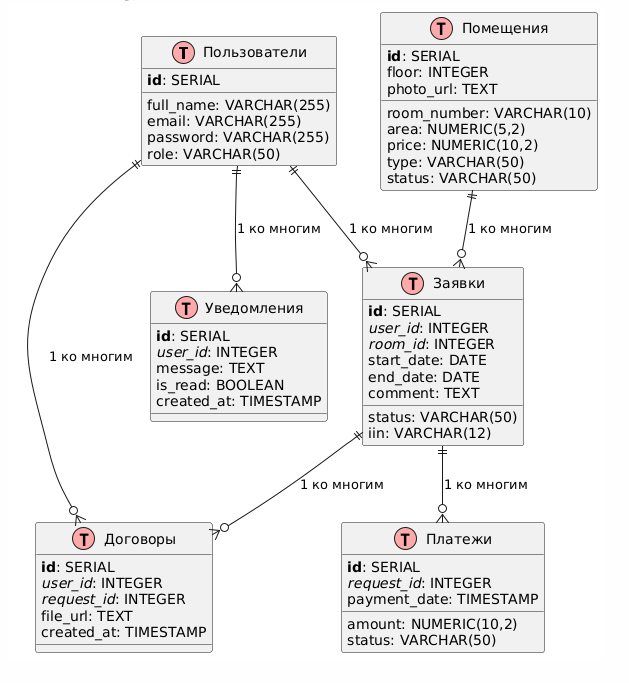


Рисунок 9 – ER-диаграмма (составлено автором)

# 2.5 Выбор технических средств для автоматизации

Фронтенд: Для разработки пользовательского интерфейса было принято решение использовать фреймворк React.js. React обеспечивает высокую производительность и позволяет создавать динамичные одностраничные приложения с отзывчивым дизайном, что особенно важно для обеспечения удобства пользователей при взаимодействии с системой. Благодаря широкому сообществу и активной поддержке, React является надёжным выбором для долгосрочных проектов.​

В качестве серверной платформы выбран Node.js в сочетании с Express.js. Node.js позволяет обрабатывать большое количество одновременных запросов, обеспечивая высокую производительность и масштабируемость системы. Express.js, как лёгкий и гибкий фреймворк, упрощает разработку RESTful API, необходимых для взаимодействия между клиентской и серверной частями приложения.​

Для хранения данных используется PostgreSQL – мощная объектно-реляционная система управления базами данных. PostgreSQL обеспечивает надёжность, поддерживает сложные запросы и транзакции, что критически важно для управления информацией о клиентах, объектах недвижимости и договорах аренды.​

Для хранения данных и обеспечения доступности приложения рассматриваются облачные платформы, такие как Google Cloud Platform (GCP) и Amazon Web Services (AWS). Обе платформы предлагают надёжные решения для хранения и обработки данных, обеспечивая масштабируемость, безопасность и высокую доступность. Выбор между GCP и AWS будет зависеть от конкретных требований проекта, бюджета и предпочтений команды разработки.

# 2.6 Способы проектирования и моделирования

Способы проектирования и моделирования для автоматизации управления арендой в компании «МАСАТО» основаны на применении микросервисной архитектуры и использовании UML-диаграмм для описания системы. Такой подход позволяет обеспечить гибкость, масштабируемость и удобство работы с системой как для пользователей, так и для администраторов.

Применение микросервисной архитектуры обусловлено необходимостью разделения функциональности системы на независимые компоненты. Каждый микросервис отвечает за выполнение одной из ключевых функций, таких как управление пользователями, обработка заявок, управление данными о помещениях и генерация договоров. Благодаря этому подходу можно легко масштабировать отдельные компоненты, добавлять новые функции и упрощать процесс тестирования. Например, сервис обработки заявок может быть обновлён или переработан без влияния на другие части системы [8].

Каждый микросервис взаимодействует с другими через API, обеспечивая независимость и изоляцию логики. Такой подход снижает риски, связанные с ошибками в коде, и упрощает поддержку системы. Кроме того, использование микросервисов позволяет адаптировать систему к различным нагрузкам, что важно для эффективного управления процессами аренды.

Использование UML-диаграмм помогает формализовать и визуализировать структуру системы, её компоненты и взаимодействие между ними. Для описания архитектуры системы и её процессов были созданы диаграммы, такие как диаграммы компонентов, диаграммы классов и диаграммы последовательностей. Они позволяют подробно рассмотреть взаимодействие между микросервисами, описать логику обработки данных и понять, как реализуются ключевые функции [9].

Диаграмма UML, отражающая взаимодействие между пользователем (арендатором) и системой, представляет основные этапы: регистрацию пользователя, просмотр доступных помещений, подачу заявки, обработку её администратором и генерацию договора. Визуализация процессов помогает команде разработчиков и других участников проекта согласовать требования и структуру системы.

Применение микросервисной архитектуры и использование UML-диаграмм обеспечивает упрощение управления проектом, повышение прозрачности процессов разработки и создания системы, а также делает её более адаптивной и устойчивой к изменениям. Это важно для компании «МАСАТО», стремящейся оптимизировать свои бизнес-процессы и улучшить клиентский сервис.

# 2.7 Ресурсные затраты и потенциальные выгоды

Ресурсные затраты и потенциальные выгоды от внедрения системы автоматизации управления арендой в компании «МАСАТО» требуют тщательной оценки, учитывающей как временные и материальные ресурсы, необходимые для разработки и поддержки системы, так и ожидаемые экономические и операционные преимущества.​

Оценка времени и ресурсов для создания и поддержки системы включает в себя несколько ключевых аспектов. На этапе разработки потребуется сформировать команду специалистов, включая аналитиков, разработчиков, дизайнеров и тестировщиков. Время, затраченное на разработку, будет зависеть от сложности функционала и архитектуры системы. После внедрения системы необходимо обеспечить её поддержку и обновление, что также потребует выделения ресурсов.​

Описание экономических и операционных выгод от внедрения системы автоматизации управления арендой в компании «МАСАТО» включает следующие аспекты: снижение операционных затрат за счёт уменьшения количества ручных операций, что приведёт к сокращению затрат на персонал и уменьшению вероятности ошибок; повышение эффективности благодаря более быстрому и точному выполнению операций, связанных с управлением арендой; улучшение качества обслуживания клиентов за счёт автоматизации процессов, что повысит их удовлетворённость и лояльность; конкурентные преимущества, связанные с внедрением современной системы управления арендой, что позволит компании выделиться на рынке, привлекая новых клиентов и укрепляя позиции среди конкурентов.​

Таким образом, несмотря на первоначальные инвестиции в разработку и внедрение системы автоматизации управления арендой, компания «МАСАТО» может ожидать значительные экономические и операционные выгоды, способствующие её устойчивому развитию и росту конкурентоспособности.​

**3 ПРОЕКТНАЯ ЧАСТЬ**

# 3.1 Общая характеристика проектируемой автоматизированной информационной системы

Проектируемая автоматизированная информационная система (АИС) «МАСАТО» предназначена для цифровизации и автоматизации бизнес-процессов, связанных с арендой коммерческой недвижимости в рамках деятельности инвестиционно-строительной компании. Система ориентирована на обеспечение эффективного взаимодействия между клиентами (арендаторами) и сотрудниками компании, предоставляя им удобный доступ к информации о помещениях, возможностям подачи заявок, отслеживанию статусов и заключению договоров аренды в электронном виде.

Необходимость разработки АИС обусловлена рядом проблем, выявленных в процессе анализа текущей практики аренды помещений: отсутствие централизованного хранения информации, высокая зависимость от бумажного документооборота, затруднённый контроль над обработкой заявок, а также недостаточная прозрачность и оперативность в работе с клиентами. В условиях цифровизации строительной отрасли и растущих требований к качеству клиентского сервиса, создание единой цифровой платформы становится стратегически важной задачей [10].

Разрабатываемая система предназначена для двух категорий пользователей: арендаторов (внешние пользователи) и администраторов (сотрудники компании). Для арендаторов реализован функционал регистрации, просмотра доступных помещений, подачи заявки на аренду, отслеживания её статуса и загрузки договора. Администраторы, в свою очередь, имеют доступ к административной панели, через которую могут управлять каталогом помещений, обрабатывать поступившие заявки, формировать и прикреплять договоры, а также вести базу клиентов. Таким образом, система охватывает полный цикл взаимодействия – от выбора помещения до юридического оформления аренды.

В проектировании АИС использован комбинированный методологический подход, сочетающий объектно-ориентированный и функциональный подходы. Объектно-ориентированная модель системы описывает основные сущности предметной области – помещения, заявки, пользователи, договоры, уведомления и платежи – как взаимосвязанные объекты с определёнными атрибутами и поведением. Функциональный подход позволяет формализовать основные бизнес-процессы, такие как подача заявки, обработка аренды, генерация договора и отправка уведомлений, в виде последовательности действий над данными.

Для моделирования логики системы использованы языки UML и IDEF0. В рамках UML построены диаграммы вариантов использования, диаграммы классов, диаграммы последовательностей и активностей, отражающие структуру и поведение системы с точки зрения пользователей и администраторов. Методология IDEF0 позволила определить ключевые функции системы и связи между ними, что стало основой для построения архитектуры приложения.

Таким образом, проектируемая система «МАСАТО» представляет собой многофункциональное, адаптивное веб-приложение, обеспечивающее автоматизацию ключевых бизнес-процессов инвестиционно-строительной компании. Система позволяет повысить прозрачность, надёжность и оперативность взаимодействия с клиентами, снизить долю ручного труда, минимизировать ошибки и обеспечить единый цифровой интерфейс для всех участников процесса аренды.

# 3.2 Архитектура системы

Одним из важнейших этапов проектирования автоматизированной информационной системы является формирование её архитектурной модели, определяющей принципы организации программных, аппаратных и сетевых компонентов. Архитектура оказывает прямое влияние на масштабируемость, отказоустойчивость, безопасность, удобство эксплуатации и сопровождения разрабатываемой системы.

Информационная система «МАСАТО» реализуется как централизованное веб-приложение, предоставляющее пользователям доступ к функционалу через браузер без необходимости установки дополнительного программного обеспечения. Такой подход обеспечивает кроссплатформенность, гибкость и снижение затрат на развертывание и обслуживание. Архитектура системы охватывает три основных уровня: аппаратный, телекоммуникационный и программный, каждый из которых играет важную роль в обеспечении целостности и надёжности функционирования АИС. Мартин Фаулер в своей работе подчёркивает необходимость разделения логики представления, бизнес-логики и доступа к данным [11].

# 3.2.1 Аппаратная архитектура

Надёжность и стабильность функционирования автоматизированной информационной системы «МАСАТО» напрямую зависят от корректного выбора аппаратной архитектуры. Система реализована в виде централизованного веб-приложения, доступ к которому осуществляется посредством браузера. Основная нагрузка ложится на серверную часть, осуществляющую обработку пользовательских запросов, выполнение бизнес-логики, работу с базой данных и генерацию договорных документов.

Серверная конфигурация.

Сервер должен обеспечивать стабильную работу backend-приложения, реализованного с использованием Express.js, базы данных PostgreSQL и клиентской части на React. Все компоненты развёртываются в локальной сети, что не требует дополнительной гибкости в управлении ресурсами [12].

Рекомендуемая конфигурация серверного оборудования:

- центральный процессор – не менее 4 ядер с тактовой частотой от 2.4 ГГц (например, Intel Xeon или AMD Ryzen 5);

- оперативная память – от 8 ГБ (оптимально – 16 ГБ при предполагаемом увеличении нагрузки);

- дисковая подсистема – SSD-накопитель объёмом не менее 100 ГБ;

- сетевой адаптер – гигабитный Ethernet-интерфейс;

- операционная система – Ubuntu Server 22.04 LTS или другой стабильный дистрибутив семейства Linux.

Возможны два сценария размещения серверной инфраструктуры: на физическом сервере в локальной сети организации либо в облачной среде (например, VPS или IaaS-платформах).

Рабочие станции пользователей.

Для конечных пользователей системы – арендаторов и администраторов – не предъявляются повышенные требования к техническим характеристикам. Единственным необходимым условием является наличие актуального веб-браузера и стабильного интернет-соединения. Возможны следующие конфигурации рабочих мест:

- настольные персональные компьютеры под управлением Windows 10/11;

- ноутбуки с macOS, Linux или ChromeOS;

- мобильные устройства с адаптивной версией интерфейса;

- минимальные требования обусловлены тем, что вся вычислительная логика реализована на стороне сервера, а клиентская часть представляет собой лёгкое SPA-приложение на React.

Масштабируемость и модернизация.

Архитектура предусматривает гибкие механизмы масштабирования:

- горизонтальное масштабирование – развёртывание дополнительных экземпляров сервисов и их распределение по узлам;

- вертикальное масштабирование – увеличение вычислительных ресурсов серверного оборудования.

# 3.2.2 Телекоммуникационная архитектура

В информационной системе «MACATO» реализована телекоммуникационная архитектура с внешним сетевым размещением, ориентированная на круглосуточную работу пользователей через интернет, независимо от их физического местоположения. Это особенно важно с учётом гибкого графика арендаторов и необходимости удалённого управления со стороны администраторов.

Система размещается на внешнем сервере с постоянным публичным IP-адресом и зарегистрированным доменным именем (например, <https://masato.kz>). Взаимодействие с системой осуществляется по защищённому протоколу HTTPS, обеспечивающему шифрование передаваемых данных и защиту от перехвата.

Пользовательская активность начинается с открытия веб-приложения в браузере. Клиентская часть, реализованная на React, инициирует HTTP-запросы к серверу через REST API. Backend-приложение, построенное на Express.js (Node.js), обрабатывает запросы, взаимодействует с базой данных PostgreSQL и возвращает ответы в формате JSON. В случае необходимости пользователи могут загружать и скачивать файлы (например, PDF-договоры) через защищённые маршруты.

Для обеспечения безопасности применяются следующие технологии и меры:

- SSL-сертификация, обеспечивающая шифрование всего трафика;

- JWT-токены (JSON Web Token) для аутентификации и разграничения прав доступа пользователей;

- CORS-политика, ограничивающая допустимые источники запросов;

- фильтрация IP-адресов на уровне административных маршрутов.

В целом, выбранная архитектура обеспечивает защищённое, гибкое и стабильное подключение пользователей к функционалу системы «МАСАТО» вне зависимости от их устройств и местонахождения.

# 3.2.3 Архитектура программного обеспечения

Программная архитектура информационной системы «МАСАТО» реализована на основе трёхуровневой клиент–серверной модели, обеспечивающей логическое разделение между представлением, бизнес-логикой и уровнем хранения данных. Такая архитектура обеспечивает масштабируемость, модульность, безопасность и удобство сопровождения. Структура системы включает в себя следующие ключевые компоненты.

Клиентская часть (Frontend) разработана с использованием библиотеки React и реализует одностраничное веб-приложение (SPA). Интерфейс включает в себя страницы авторизации, подачи заявок, личный кабинет арендатора и административную панель. Все компоненты взаимодействуют с сервером посредством HTTP-запросов к REST API, обмениваясь данными в формате JSON. Реализация на базе React позволяет обеспечить интерактивность интерфейса и быстрое реагирование на действия пользователя [13].

Серверная часть (Backend) построена на платформе Node.js с использованием Express.js. Основные функции backend-части включают реализацию бизнес-логики, маршрутизацию запросов, аутентификацию и авторизацию пользователей на основе JWT, а также обработку заявок, управление договорами и взаимодействие с базой данных. Серверное приложение организовано по модульной структуре (routes, controllers, services), что упрощает поддержку и развитие проекта.

Система управления базами данных (PostgreSQL) в качестве СУБД используется реляционная система PostgreSQL, где хранятся структурированные данные о пользователях, помещениях, заявках, договорах, уведомлениях и платежах. База данных спроектирована на основе ER-диаграммы и использует внешние ключи, каскадное удаление, ограничения целостности и строгую типизацию данных.

Файловая система используется для хранения PDF-договоров аренды и изображений помещений. Пути к файлам сохраняются в базе данных, а доступ к ним осуществляется через защищённые маршруты которые находятся в backend-приложения.

Программная архитектура системы «МАСАТО» отвечает современным требованиям к корпоративным веб-приложениям – она масштабируема, безопасна, надёжна и легко расширяема. Использование открытых технологий и модульный подход позволяют быстро вносить изменения, адаптировать систему под новые задачи и обеспечивать её стабильную работу при росте нагрузки.

# 3.3 Проектирование базы данных

Эффективное проектирование базы данных является критически важным этапом разработки информационной системы, поскольку от логики хранения и структуры данных напрямую зависит надёжность, согласованность и масштабируемость всего проекта. В рамках реализации автоматизированной системы «МАСАТО» база данных выполняет роль центрального элемента, обеспечивающего хранение, обработку и взаимосвязь ключевых сущностей, связанных с арендой помещений.

Разрабатываемая система предназначена для автоматизации бизнес-процессов инвестиционно-строительной компании, занимающейся предоставлением коммерческой недвижимости в аренду. Это определило состав предметной области, включающей такие основные сущности, как: пользователь, помещение, заявка на аренду, договор, платёж, а также уведомление. Каждая из этих сущностей получила отражение в виде отдельной таблицы в реляционной структуре, с чётко определёнными атрибутами и связями между ними [14].

Проектирование базы данных осуществлялось на основе анализа требований, приведённых в аналитической части, и логики функционирования системы. Все таблицы разработаны в соответствии с принципами реляционной модели, что позволяет обеспечить логическую целостность, транзакционную устойчивость и масштабируемость при многопользовательском доступе. В процессе проектирования применялись методы нормализации, направленные на устранение избыточности данных и минимизацию возможных аномалий при обновлении, вставке и удалении информации.

Связи между таблицами реализованы с использованием внешних ключей с каскадным удалением (ON DELETE CASCADE), что обеспечивает автоматическое поддержание согласованности данных. Например, при удалении пользователя автоматически удаляются все связанные с ним заявки, уведомления, договоры и платежи. Некоторые связи, такие как между заявками и пользователями или заявками и помещениями, представляют собой типичные отношения «многие к одному».

Конструкция базы данных также предусматривает возможность расширения: при необходимости могут быть добавлены дополнительные таблицы, например, для ведения истории аренды, логирования действий пользователей, интеграции с платёжными шлюзами или подключения аналитических модулей.

Таким образом, разработанная структура базы данных представляет собой прочную и масштабируемую основу, полностью соответствующую логике предметной области системы «МАСАТО». Она обеспечивает централизованное управление данными, упрощает выполнение операций по заявкам и договорам, поддерживает корректность всех бизнес-процессов и предоставляет широкие возможности для дальнейшего развития системы.

# 3.3.1 Функциональная декомпозиция (диаграмма IDEF0)

Для формализации функциональной архитектуры информационной системы «МАСАТО» использовалась методология IDEF0, которая широко применяется в области системного анализа и проектирования. Данный подход позволяет представить структуру процессов в виде иерархически организованных функций, описывающих, что именно делает система, с какими данными она работает, какие ресурсы использует и какие результаты выдаёт на выходе.

Применение IDEF0 способствует наглядному и логически последовательному описанию процессов, что особенно важно при разработке сложных информационных систем с множеством взаимосвязанных компонентов. Методология помогает структурировать бизнес-логику, выделить основные функции системы, определить связи между ними, а также указать, какие внешние воздействия, входные данные, механизмы и регламенты участвуют в каждом этапе обработки информации [15].

В аналитической части дипломной работы (см. раздел 1.4.1, рисунок 1) была представлена контекстная диаграмма уровня A0, которая отображает обобщённое представление всей системы как одного большого функционального блока. На этой диаграмме определяются главные входы (заявки, данные о помещениях), управляющие воздействия (правила обработки, бизнес-логика), механизмы (техническая реализация: база данных, интерфейсы, API) и выходы (сформированные договоры, история аренды). Это позволяет взглянуть на систему с высшего уровня абстракции.

В данном разделе представлена диаграмма следующего уровня декомпозиции – A1, которая раскрывает внутреннюю структуру одного из важнейших компонентов системы – подсистемы обработки заявок на аренду помещений. Эта часть системы является ключевой, так как именно через неё проходит основное взаимодействие арендатора с платформой и осуществляется контроль со стороны администратора.

На рисунке 10 представлена диаграмма IDEF0 уровня A1, демонстрирующая детализированный процесс обработки заявки на аренду помещения. Диаграмма включает основные этапы: получение заявки от арендатора через веб-интерфейс, её автоматическую регистрацию в системе, последующую проверку и анализ администратором, а также принятие решения о подтверждении или отклонении заявки. Каждый блок диаграммы сопровождается входными, выходными, управляющими и механическими связями, что обеспечивает наглядное представление логики бизнес-процесса и взаимодействия участников.



Рисунок 10 – Диаграмма IDEF0 уровня A1: обработка заявки

(составлено автором)

Входные данные (Input):

- данные заявки – информация, поступающая от арендатора через форму на клиентской части (React);

- контроллер request – API-маршрут, обрабатывающий полученные данные на стороне сервера (Express.js);

- JWT-токен – маркер авторизации, обеспечивающий верификацию роли пользователя и доступ к защищённым маршрутам.

Управляющие воздействия (Control):

- Node.js-сервер – реализует бизнес-логику и маршруты обработки заявок;

- PDF-генератор – вспомогательный модуль, формирующий договор аренды в формате PDF.

Выходные данные (Output):

- обновлённый статус заявки – «одобрено» или «отклонено», сохраняется в базе данных;

- уведомление – сообщение, отправляемое пользователю о результате рассмотрения заявки;

- PDF-файл – сгенерированный договор аренды, сохраняемый в файловой системе и отображаемый в личном кабинете.

Применение IDEF0 позволило формализовать взаимодействие между модулями backend- и frontend-части, выделить основные точки интеграции и упростить логику реализации при программировании. Такая модель играет важную роль на этапе реализации, тестирования и сопровождения системы, обеспечивая предсказуемость бизнес-процессов и удобство в построении пользовательских сценариев.

# 3.4 Проектирование пользовательского интерфейса

Проектирование пользовательского интерфейса является одним из ключевых этапов разработки информационной системы, поскольку именно через интерфейс осуществляется непосредственное взаимодействие пользователя с функциональностью программного продукта. Грамотно спроектированный интерфейс повышает удобство использования, снижает вероятность ошибок и способствует более быстрому освоению системы [16].

В рамках проекта «МАСАТО» пользовательский интерфейс разрабатывается с учётом двух основных ролей: арендатора (внешнего пользователя) и администратора (сотрудника компании). Разделение интерфейса по ролям позволяет адаптировать доступный функционал, навигационные элементы и отображаемую информацию в соответствии с задачами каждого типа пользователя.

# 3.4.1 Интерфейс администратора

Интерфейс администратора в системе «МАСАТО» был спроектирован с учётом требований эргономики, простоты восприятия и эффективности повседневной работы сотрудников компании. Основной акцент в дизайне сделан на минимализм, структурированность и оперативный доступ ко всем ключевым разделам системы. Это позволяет администраторам сосредоточиться на выполнении своих задач без отвлечения на второстепенные элементы и лишние переходы между экранами.

Администратор работает преимущественно в режиме просмотра, контроля и редактирования данных. После авторизации в системе он сразу попадает на главную страницу административной панели, где отображаются основные вкладки для управления системой: «Заявки», «Аналитика», «Каталог». Такое расположение элементов навигации делает интерфейс интуитивно понятным и позволяет быстро перейти к нужному разделу.

В разделе «Каталог» администратор видит полный список помещений с указанием основных характеристик: номер, этаж, площадь, стоимость, статус и тип. Здесь доступны функции добавления новых помещений, редактирования существующих карточек и удаления объектов. Отображено на рисунке 11.

Добавление и редактирование помещений происходит через простые формы с полями для ввода параметров и загрузки изображения. Все изменения моментально сохраняются в базе данных и отображаются в пользовательском интерфейсе. Представлено на рисунке 12.

Каталог является важным инструментом, позволяющим поддерживать актуальность информации о недвижимости и эффективно управлять арендным фондом. Он обеспечивает структурированный доступ к информации, упрощает поиск помещений для арендаторов.

## 

Рисунок 11 – Страница каталог (составлено автором)

## 

Рисунок 12 – Каталог помещений (составлено автором)

Такое отображение позволяет администратору оперативно контролировать помещения, а так же редактировать в случае необходимости. Раздел выполнен в виде простой таблицы без лишних элементов, что повышает читаемость и снижает когнитивную нагрузку при работе с интерфейсом.

Перейдя в раздел заявки, администратор видит список всех заявок. Имеет возможность одобрить заявку, отклонить заявку. В случае одобрения заявки сформировать договор аренды помещения. Отображено рисунке 13.

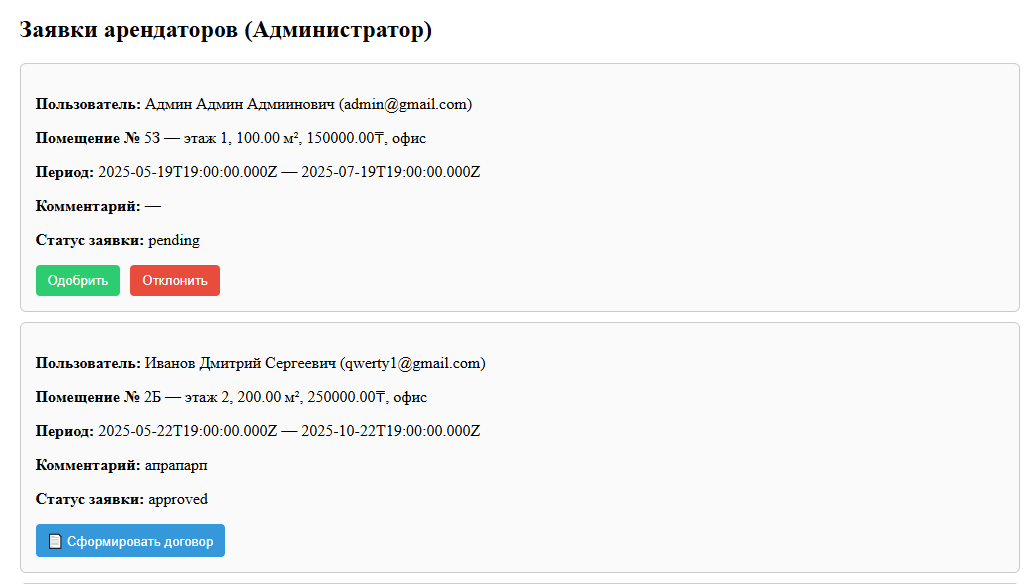


Рисунок 13 – Заявки арендаторов (составлено автором)

Такое отображение позволяет администратору своевременно рассматривать заявки арендаторов с последующей генерацией договора аренды.

Раздел «Аналитика» предоставляет администратору доступ к ключевым показателям арендной деятельности компании. После перехода в этот раздел отображаются сводные данные о выручке, количестве заявок, активных договорах и финансовой динамике за выбранный период. Продемонстрировано на рисунке 14.

Графики и диаграммы позволяют наглядно оценить рост или снижение доходов компании, отслеживать изменения в динамике аренды, а также сравнивать ключевые показатели по дням, неделям или месяцам. Это даёт возможность выявлять тренды, сезонные колебания и прогнозировать будущие финансовые показатели. Отдельный блок аналитики отображает статистику по наиболее популярным помещениям – тем объектам, на которые подаётся наибольшее количество заявок со стороны арендаторов. Данные формируются автоматически на основе активности пользователей, что позволяет оценить спрос и вовремя скорректировать ценовую политику.

Также в разделе предусмотрена удобная генерации отчётов, необходимых как для внутреннего контроля, так и для передачи информации в налоговые или контролирующие органы. Отчёты формируются автоматически по заданным параметрам. Представлено на рисунке 15.

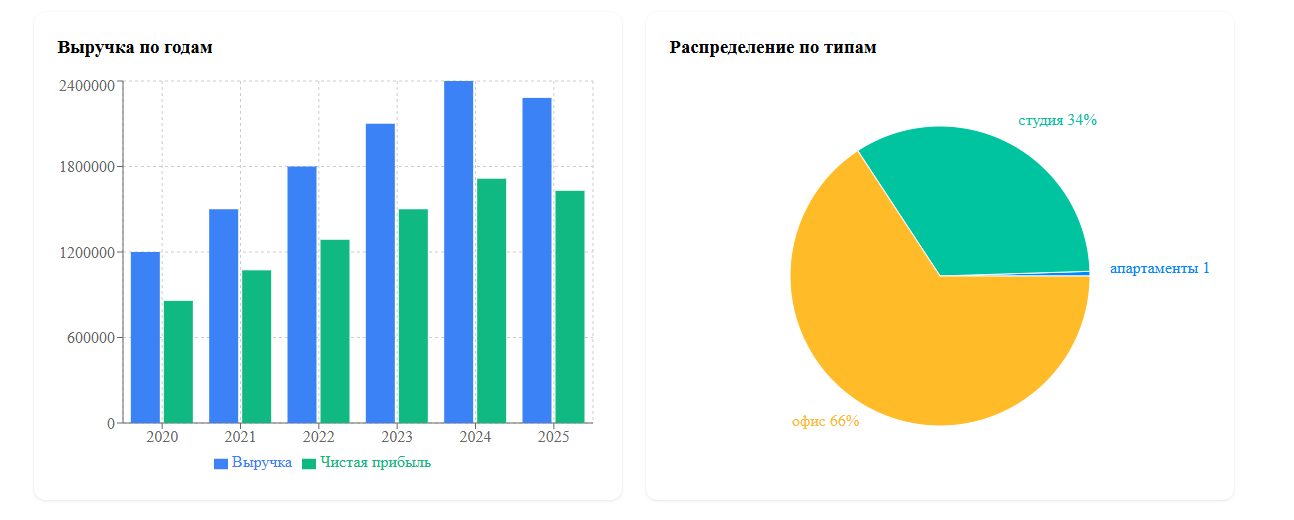


Рисунок 14 – Финансовая аналитика (составлено автором)

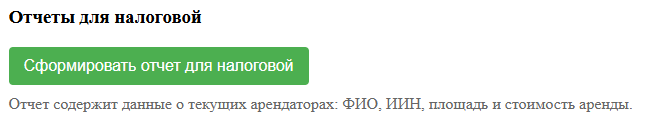


Рисунок 15 – Генерация отчета (составлено автором)

Таким образом администратор получает актуальную статистику в реальном времени о выручки и доходах. Может своевременно генерировать отчеты для налоговой службы. Отчет включает в себя личные заказчика, площадь арендуемого помещения заказчиком, а также стоимость аренды выбранного помещения.

# 3.4.2 Интерфейс пользователя

Интерфейс пользователя в системе «МАСАТО» спроектирован с акцентом на удобство, минимализм и быстрый доступ к информации. После входа в систему администратор попадает на главную страниц. Где видит удобные вкладки: «Каталог», «Заявки», а так же в случае авторизации видит «Личный кабинет».

При входе на сайт пользователь сразу попадает на страницу с каталогом помещений, доступных для аренды. В каталоге отображается список всех объектов с указанием основных характеристик: номер, этаж, площадь, стоимость аренды и текущий статус (свободно/занято).

Каждое помещение представлено в виде отдельной карточки, что делает просмотр информации удобным и визуально наглядным. Пользователь может быстро оценить параметры и выбрать подходящий вариант. Предусмотрена система фильтров по этажам, площади, цена и статусе (рисунок 16).

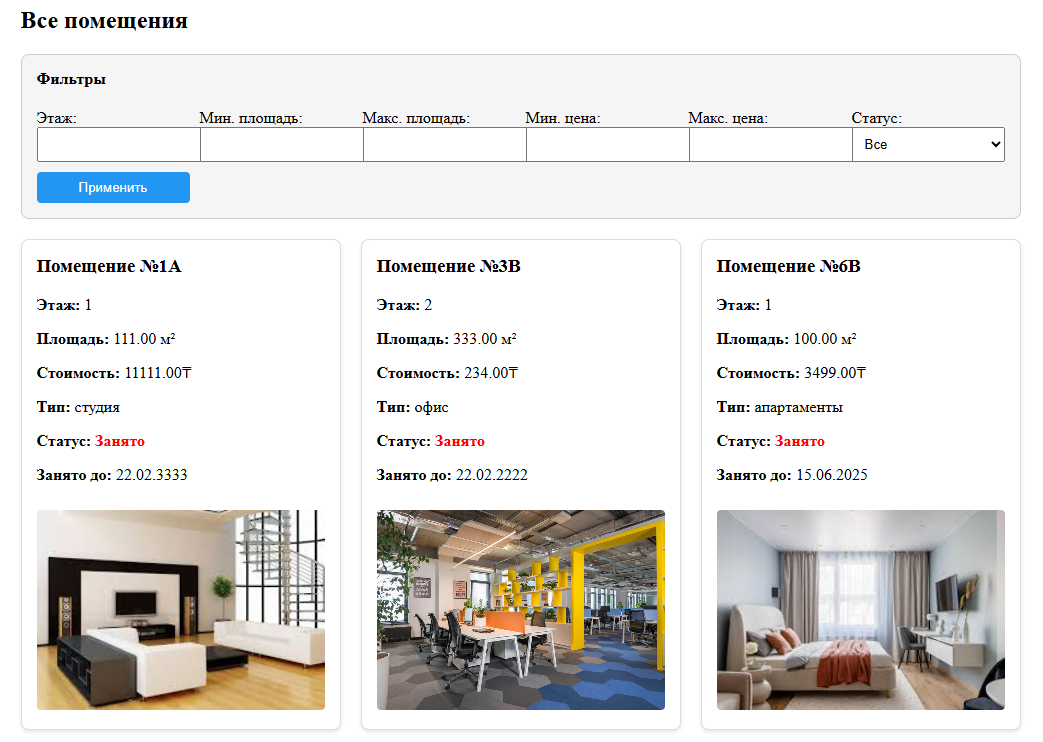


Рисунок 16 – Каталог помещений (составлено автором)

Не авторизованный пользователь решает подать заявку на аренду помещения. Рисунок 17 демонстрирует подачу заявки.

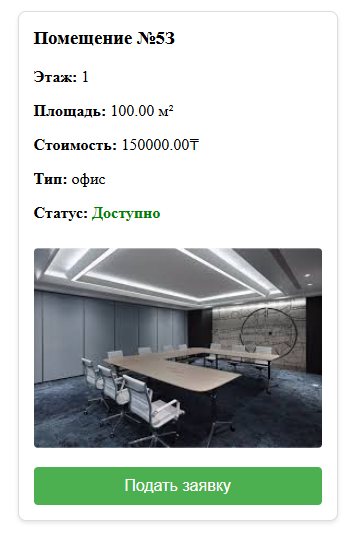


Рисунок 17 – Подача заявки (составлено автором)

После нажатия на кнопку не авторизованный пользователей будет перенаправлен в раздел регистрации, авторизации. Представлено на рисунках 18 и 19.

## 

Рисунок 18 – Авторизация пользователя (составлено автором)

## 

Рисунок 19 – Регистрация пользователя (составлено автором)

Такое отображение позволяет пользователю легко ориентироваться в системе. С легкостью проходить процесс регистрации в системе и без труда оформлять заявку на аренду помещения.

После прохождения авторизации пользователь получает доступ в форме оформления заявки на аренду помещения. Отображает рисунок 20.

## 

Рисунок 20 – Оформление заявки (составлено автором)

После заполнения всех данных и окончательного оформления заявки пользователю нажимает на кнопку «Отправить заявку». Его заявка появляется на странице «мои заявки». Представлено на рисунке 21.

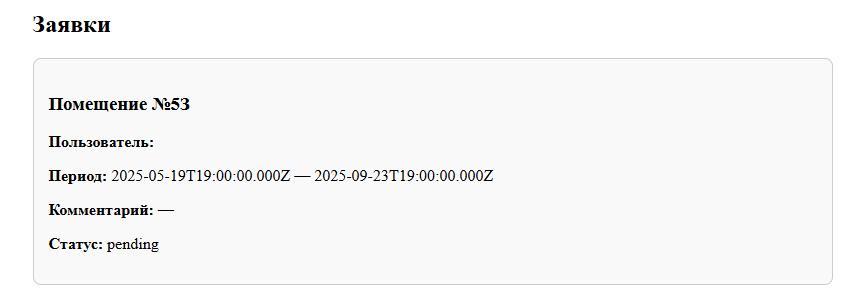


Рисунок 21 – Заявка пользователя (составлено автором)

Такой интерфейс помогает пользователю интуитивно формировать заявку на аренду помещения.

После авторизации пользователю становится доступен его личный кабинет где отражены его личные данные. Так же пользователь может контролировать статус своей заявки путем просмотра уведомлений системы. Отображено на рисунке 22.

## 

Рисунок 22 – Личный кабинет пользователя (составлено автором)

При рассмотрении и одобрении заявки администратором пользователь получает возможность оплатить выбранное помещение, а так же скачать договор аренды. Представлено рисунке 22.

## 

Рисунок 23 – Оплата, скачивание договора аренды (составлено автором)

Для удобства оплаты долгосрочных аренд пользователь имеет доступ к своей истории платежей, где отображаются все совершённые транзакции. Это позволяет отслеживать даты и суммы поступлений, а также понимать, когда проводилась последняя оплата, тем самым упрощая финансовое планирование и контроль. Представлено на рисунке 24.

## 

Рисунок 24 – История платежей пользователя (составлено автором)

Таким образом пользователь имеет удобный интерфейс для отслеживания статуса заявок. Имеет возможность к скачиванию договора аренды, а также инструменты для оплаты помещения.

# 3.5 Безопасность и разграничение прав доступа

В рамках разработки системы управления арендой помещений МАСАТО особое внимание было уделено вопросам информационной безопасности и разграничению прав доступа. Поскольку система оперирует конфиденциальными данными, включая персональную информацию пользователей, финансовые детали аренды и юридические документы, архитектура безопасности была спроектирована с учетом современных требований к защите информации.

Основой системы авторизации стала токен-ориентированная модель на базе JSON Web Tokens (JWT), реализованная через механизм аутентификации в Express.js. После успешной процедуры входа пользователь получает цифровой токен, содержащий зашифрованные данные о его идентификаторе и ролевой принадлежности. Данный токен сохраняется на клиентской стороне и автоматически включается в заголовки всех последующих запросов к серверному API. На уровне серверной обработки запросов реализована многоуровневая система проверок, включающая middleware для верификации подлинности токена и контроля доступа к защищенным маршрутам [17].

Архитектура системы предусматривает четкое разделение пользователей по ролевому принципу, где выделены две основные категории: стандартные пользователи (арендаторы) и администраторы. Первые обладают ограниченным набором прав, позволяющим лишь просматривать информацию о доступных помещениях, подавать заявки на аренду и отслеживать статус своих запросов. Администраторский интерфейс, в свою очередь, предоставляет расширенные возможности по управлению всем жизненным циклом арендных отношений - от модерации входящих заявок до генерации юридически значимых документов и аналитики финансовых показателей.

Техническая реализация защиты данных включает несколько взаимодополняющих механизмов. Все пароли пользователей хранятся в базе данных исключительно в хешированном виде с применением алгоритма bcrypt, что исключает возможность их компрометации даже в случае несанкционированного доступа к серверу. Для передачи конфиденциальной информации между клиентом и сервером используется исключительно защищенное HTTPS-соединение. На уровне API реализована строгая валидация всех входящих параметров запросов, предотвращающая как случайные ошибки ввода, так и потенциальные атаки инъекционного типа.

Особое внимание при проектировании системы было уделено защите от межсайтовой подделки запросов (CSRF) и несанкционированного межсайтового взаимодействия (CORS). Серверная часть настроена на прием запросов только с доверенных доменов, а для всех административных операций требуется обязательное наличие валидного JWT-токена с соответствующей ролевой меткой. Дополнительный уровень безопасности обеспечивается за счет реализации механизма автоматического аннулирования токенов при попытках доступа к запрещенным ресурсам.

С точки зрения защиты персональных данных, система соответствует основным требованиям современных стандартов информационной безопасности. Все договоры аренды и финансовые документы хранятся в зашифрованном виде, а доступ к ним осуществляется через строгую систему проверки полномочий. Архитектура безопасности системы предусматривает возможность дальнейшего расширения функционала защиты, включая потенциальное внедрение двухфакторной аутентификации или интеграцию с корпоративными системами управления идентификацией.

# 3.6 Моделирование пользовательских сценариев (UML)

При проектировании системы применялись UML-диаграммы (Unified Modeling Language) для визуализации пользовательских сценариев и бизнес-процессов, что позволило детально проанализировать последовательность действий пользователей, ключевые точки принятия решений и возможные варианты развития событий при взаимодействии с системой. Данный подход обеспечил четкое понимание логики работы функциональных модулей, выявление потенциально проблемных мест в процессах обработки данных, а также создал основу для последующей разработки тестовых сценариев, охватывающих как стандартные, так и исключительные ситуации. Визуальное моделирование послужило эффективным инструментом согласования требований между разработчиками и заказчиками, способствуя созданию единой концептуальной модели системы на ранних этапах проектирования.

# 3.6.1 Диаграмма деятельности

Диаграмма деятельности отображает последовательность действий арендатора при взаимодействии с системой. Процесс начинается с открытия веб-сайта, где пользователь должен пройти регистрацию или авторизацию. Если вход выполнен успешно, пользователь получает доступ к каталогу помещений, где он может применить фильтры по этажу, стоимости и площади для удобного поиска подходящего варианта (рисунок 25) [18].

## 

Рисунок 25 – Диаграмма деятельности (составлено автором)

 После выбора помещения пользователь переходит к этапу подачи заявки. Он вводит контактные данные, указывает желаемый срок аренды и оставляет комментарии. Система сохраняет заявку и переводит её в статус «на рассмотрении». На этом этапе пользователь ожидает решения от администратора.

 Если заявка подтверждается администратором, система переходит к следующему действию – формированию договора аренды. Сформированный договор отображается в личном кабинете арендатора, откуда его можно скачать для дальнейшего использования. В случае отклонения заявки система уведомляет пользователя об отказе.

 Если вход в систему не был успешным, пользователю отображается сообщение об ошибке, и дальнейшие действия невозможны до прохождения авторизации. После завершения всех возможных сценариев взаимодействие пользователя с системой заканчивается.

# 3.7 Выбор программных средств разработки

При разработке автоматизированной информационной системы «MAСATO» был применён современный технологический стек, соответствующий требованиям надёжности, гибкости, масштабируемости и безопасности. Каждый компонент программной архитектуры был выбран на основе анализа функциональных и нефункциональных требований, выявленных на этапе проектирования, а также с учётом сопоставления с альтернативными решениями.

Выбор программных средств обусловлен необходимостью обеспечения надёжной работы системы в условиях реального использования – в сфере аренды коммерческой недвижимости с участием большого количества пользователей, а также учётом индивидуальной разработки, минимизирующей затраты на сопровождение и развёртывание [19].

Серверная часть Express.js.

В качестве фреймворка для разработки серверной части системы был выбран Express.js – минималистичное, производительное и гибкое решение, широко используемое в экосистеме Node.js. Он предоставляет базовую инфраструктуру для построения REST API, позволяет легко интегрироваться с базой данных PostgreSQL, а также масштабировать бизнес-логику за счёт модульной структуры (routes, controllers, services).

В отличие от более тяжёлых решений, таких как NestJS, Express.js требует меньше конфигурации и предоставляет разработчику полный контроль над архитектурой приложения. Это делает его особенно удобным в рамках индивидуальных проектов, где важны скорость реализации и гибкость архитектурных решений.

Клиентская часть React.

Пользовательский интерфейс системы реализован с использованием библиотеки React, которая позволяет разрабатывать динамичные и отзывчивые одностраничные приложения (SPA). React обладает высокой степенью модульности, что облегчает разработку повторно используемых компонентов (формы, таблицы, уведомления и т.д.), а также хорошо масштабируется для будущего расширения.

React был выбран как наиболее популярная и зрелая библиотека с большим сообществом, обширной документацией и широким набором дополнительных инструментов (например, React Router, React Hook Form). Он превосходит альтернативы (например, Vue.js) по количеству готовых решений и уровню промышленного применения.

Система управления базами данных PostgreSQL.

В качестве СУБД в проекте используется PostgreSQL – объектно-реляционная СУБД с открытым исходным кодом. Она обеспечивает поддержку транзакций, строгую типизацию, каскадное удаление, внешние ключи и высокую надёжность при работе с большим объёмом табличных данных.

В отличие от MySQL, PostgreSQL предоставляет более развитые возможности для реализации сложных связей, поддерживает работу с JSON, оконные функции, вложенные транзакции и более гибкую схему нормализации данных. Это делает её оптимальным решением для хранения информации о пользователях, помещениях, заявках, уведомлениях.

Стилизация Inline CSS + JSS.

В качестве основного метода стилизации компонентов использован гибридный подход – инлайн-стилизация (JSS) в сочетании с JavaScript-константами (дизайн-токены). Такой подход позволяет создавать самодостаточные компоненты, избегать конфликтов имён классов и легко масштабировать интерфейс. В таблице 1 представлена сравнительная информация выбранных программных средств с их альтернативами.

Таблица 1 – Сравнение выбранных программных средств с альтернативами

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Компонент | |  | | --- | |  |   Выбранное средство   |  | | --- | |  | | Альтернатива | Преимущества выбранного варианта |
| Backend-фреймворк | Express.js | NestJS | |  | | --- | |  |   Простота, гибкость, минимальная конфигурация   |  | | --- | |  | |
| Frontend-библиотека | React | |  | | --- | |  | | Vue.js | | | Компонентность, популярность, широкое сообщество |
| СУБД | |  | | --- | |  | | PostgreSQL | | | MySQL | Расширенные возможности, поддержка каскадов и транзакций |
| Стилизация | |  | | --- | |  |   Inline CSS + JSS   |  | | --- | |  | | |  | | --- | |  | | TailwindCSS | | | Полный контроль, отсутствие CSS-файлов, динамические стили |
| Сборка фронтенда | Vite | Webpack | Высокая скорость, HMR |
| Примечание – составлено автором | | | |

# 

# 4 ТЕСТИРОВАНИЕ И РАЗРАБОТКА

# 4.1 Цели и задачи тестирования

Тестирование системы управления арендой помещений МАСАТО являлось важнейшим этапом разработки, направленным на подтверждение соответствия реализованного функционала исходным требованиям и выявление возможных отклонений в работе системы. Основной акцент был сделан на проверку корректности выполнения ключевых бизнес-процессов, включая подачу заявок на аренду, их модерацию администратором, генерацию договоров, обработку платежей и систему уведомлений. Особое внимание уделялось устойчивости системы при работе с некорректными данными, таким как неверные форматы дат, отсутствующие обязательные поля или попытки несанкционированного доступа.

В основе методологии тестирования лежал принцип комплексной оценки качества, охватывающий как функциональные, так и нефункциональные характеристики системы. С точки зрения функциональности проверялась способность системы корректно обрабатывать все заявленные сценарии работы, включая создание заявок, их одобрение или отклонение, формирование юридически значимых документов в формате PDF, а также корректное обновление статусов помещений. Надежность системы оценивалась через призму устойчивости к ошибкам ввода и способности сохранять работоспособность при возникновении исключительных ситуаций [20].

Удобство использования проверялось через анализ логики пользовательского интерфейса, доступности функций и понятности системы обратной связи при совершении действий. Эффективность работы системы оценивалась по времени отклика на типовые операции, таким как загрузка списка помещений или генерация договора. Отдельное внимание было уделено сопровождаемости кода – возможности относительно простого внедрения изменений и расширения функциональности в будущем.

**4.2 Организация процесса тестирования**

Тестирование системы МАСАТО проводилось в рамках ограниченных ресурсов, что определило выбор методов и глубину проверки. Основной акцент был сделан на ручное тестирование ключевых сценариев работы, что обусловлено масштабом проекта и составом команды, где разработчик одновременно выступал в роли тестировщика.

Процесс тестирования был интегрирован в жизненный цикл разработки и выполнялся после реализации основных функциональных модулей. Проверка начиналась с unit-тестирования отдельных компонентов, таких как валидация входящих данных в формах или корректность работы middleware для проверки JWT-токенов. Затем проводилось интеграционное тестирование, направленное на проверку взаимодействия между клиентской и серверной частями, включая обработку заявок и формирование договоров.

Основные ограничения процесса тестирования касались невозможности проведения нагрузочного тестирования и проверки системы в условиях высокой одновременной нагрузки. Также в рамках проекта не проводилось кросс-браузерное тестирование и оценка производительности на мобильных устройствах. Однако, несмотря на эти ограничения, удалось достичь достаточного уровня покрытия основных пользовательских сценариев и подтвердить стабильность работы системы в предусмотренных условиях эксплуатации.

# 4.3 Результаты и выводы

Проведенное тестирование позволило верифицировать корректность работы всех ключевых модулей системы. Были успешно проверены сценарии подачи заявок различными пользователями, их обработка администратором, включая одобрение и отклонение, а также последующее формирование договоров в формате PDF. Особое внимание уделялось проверке граничных условий, таких как попытки бронирования уже занятых помещений, ввод некорректных периодов аренды или обработка заявок с просроченными токенами авторизации.

Система продемонстрировала устойчивость к основным видам некорректного ввода, предоставляя понятные сообщения об ошибках при нарушении бизнес-правил. Проверка безопасности подтвердила эффективность реализации ролевой модели и механизма JWT-аутентификации. Все выявленные в процессе тестирования проблемы, в основном связанные с обработкой крайних случаев и валидацией данных, были своевременно устранены до момента сдачи проекта.

В перспективе рекомендуется расширить тестовое покрытие за счет внедрения автоматизированного тестирования критически важных сценариев с использованием таких инструментов как Jest для unit-тестов и Cypress для end-to-end тестирования. Также целесообразно провести нагрузочное тестирование для оценки производительности системы в условиях одновременной работы множества пользователей. Тем не менее, проведенных тестовых мероприятий оказалось достаточно для подтверждения готовности системы к эксплуатации в рамках заявленных требований. Тестирование позволяет обнаружить до 70% критических ошибок до запуска продукта .

# 4.4 Методы и способы тестирования

Тестирование информационной системы управления арендой помещений МАСАТО осуществлялось преимущественно вручную и охватывало все основные пользовательские сценарии, отражённые в техническом задании. Методология тестирования основывалась на применении позитивных и негативных функциональных проверок, направленных на выявление ошибок в логике работы системы, тестирование прав доступа и устойчивости при различных входных данных. Выбор такого подхода был обусловлен ограниченностью ресурсов, отсутствием отдельной команды тестирования и экспериментальным характером проекта в рамках индивидуальной разработки.

Проверка проводилась в среде локального развёртывания, что обеспечивало полную изоляцию и воспроизводимость среды. Компоненты системы – клиентское приложение на React, серверное API на Express.js и база данных PostgreSQL – были развернуты в отдельных контейнерах. Это позволило эмулировать реальную работу системы в условиях, приближенных к эксплуатации.

Тестирование включало следующие ключевые направления.

Функциональное тестирование – проверка корректности реализации заявленного функционала, включая регистрацию пользователей, подачу заявок, их подтверждение или отклонение администратором, генерацию договоров в формате PDF, отображение уведомлений и истории заявок.

Интерфейсное тестирование – проверка визуальной части: работа кнопок, отображение таблиц, загрузка изображений, реакция на пользовательские действия.

Тестирование авторизации и разграничения прав – проверка доступа к административным маршрутам и интерфейсам только при наличии действующего JWT-токена с ролью администратора.

Негативное тестирование – имитация ошибок ввода: отсутствие обязательных полей, ввод некорректных дат, попытки доступа к защищённым разделам без авторизации.

Тестирование выполнялось вручную, в пошаговом режиме, соответствующем логике предполагаемого поведения конечного пользователя. В процессе проверки ошибки и некорректное поведение интерфейса фиксировались в виде замечаний, после чего соответствующий участок кода дорабатывался. Далее следовала повторная проверка, что позволяло обеспечить непрерывный цикл устранения дефектов.

Хотя в проекте не использовались инструменты автоматизированного тестирования, ручной подход позволил достичь высокого уровня покрытия пользовательских сценариев, что подтверждает его практическую эффективность в условиях ограниченного масштаба проекта.

# 4.5 Результаты тестирования

Финальное ручное тестирование информационной системы МАСАТО проводилось после реализации всех основных модулей. Его целью было подтверждение корректности выполнения ключевых сценариев, а также проверка устойчивости и надёжности системы. Особое внимание уделялось проверке взаимодействия между клиентом и сервером, обработке данных, соблюдению бизнес-логики, корректному отображению статусов заявок и генерации документов.

Все проверки выполнялись в реальном браузере (Google Chrome) на основе типовых пользовательских действий. Были смоделированы сценарии работы с системой от лица как арендатора, так и администратора. Для каждого из них был составлен список ожидаемых результатов, с которыми сверялись фактические данные. В ходе тестирования не было выявлено критических ошибок, а единичные недочёты – в частности, отсутствие валидации при подаче заявки с пустым полем или неправильной датой – были оперативно устранены.

Среди успешно пройденных проверок можно выделить:

- авторизацию и доступ к функциональности по ролям;

- подачу заявок на аренду;

- проверку данных и их валидацию;

- обработку заявок администратором (подтверждение/отклонение);

- генерацию договора аренды в PDF-формате;

- отображение уведомлений и статусов в личном кабинете пользователя;

- блокировку доступа к административным маршрутам при отсутствии прав.

Таким образом, проведённое тестирование подтвердило корректность работы всех основных компонентов системы и готовность проекта к дальнейшему развертыванию в продуктивной среде.

# 4.6 Практическая значимость и ожидаемый эффект

Разработка и внедрение информационной системы МАСАТО имеет выраженную практическую направленность и может быть эффективно применена в деятельности инвестиционно-строительных компаний, занимающихся арендой коммерческой недвижимости. Система автоматизирует ключевые процессы взаимодействия между арендаторами и администрацией: от подачи заявки до заключения договора, тем самым снижая количество ручных операций и повышая прозрачность документооборота.

Практическая значимость заключается в устранении проблем, характерных для ручной или фрагментированной работы с заявками и помещениями: дублирования данных, ошибок при вводе, отсутствия централизованной истории и прослеживаемости действий. Внедрение МАСАТО позволяет централизовать управление помещениями, автоматизировать типовые операции и сократить временные издержки на обработку запросов.

Для пользователей система предоставляет удобный, интуитивно понятный интерфейс, доступный через браузер с любого устройства. Возможность отслеживания статуса заявки, получения договора в личном кабинете и оперативного получения уведомлений делает взаимодействие с арендодателем более прозрачным и предсказуемым.

Для администрации система предоставляет инструменты быстрой модерации заявок, управления помещениями, генерации договоров, аналитики и контроля процессов аренды. Разделение доступа по ролям, защита маршрутов и авторизация через JWT обеспечивают безопасность работы и предотвращают несанкционированные действия.

Использование современных технологий (React, Express.js, PostgreSQL) делает систему легко масштабируемой и адаптируемой под новые требования. Возможность быстрого развёртывания в изолированной среде, а также отсутствие привязки к лицензированному ПО повышают экономическую эффективность решения.

Таким образом, МАСАТО не только реализует учебные цели дипломного проекта, но и демонстрирует потенциал к практическому применению в сфере управления арендуемой недвижимостью, а также служит примером эффективного использования информационных технологий в бизнес-среде.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была достигнута основная цель – разработана и протестирована информационная система аренды помещений для компании «МАСАТО», реализованная в виде веб-приложения с публичной и административной частью.

В первой части работы проведён анализ предметной области, выявлены основные проблемы в текущем подходе к аренде помещений, определены ключевые бизнес-процессы и обоснована необходимость их автоматизации.

Во второй части проекта рассмотрены и сопоставлены современные инструменты и технологии. Выбран стек разработки, включающий React для фронтенда, Node.js с Express для серверной части и PostgreSQL для хранения данных, что обеспечивает надёжность, масштабируемость и высокую скорость работы приложения.

Третья часть посвящена проектированию архитектуры, интерфейсов и базы данных, с учётом логики ролей «администратор» и «арендатор». Предусмотрена регистрация, авторизация, создание и модерация заявок, просмотр информации об объектах, управление пользователями и помещениями.

На этапе реализации была создана полноценная система с REST API, адаптивным интерфейсом и возможностью масштабирования. Проведено тестирование, подтвердившее корректность функционирования основных модулей и соответствие заявленным требованиям.

К достоинствам разработанного решения можно отнести:

- модульную архитектуру и чистую структуру кода;

- удобный интерфейс, адаптированный под разные устройства;

- разделение прав доступа и учёт бизнес-логики;

- возможность масштабирования и расширения функциональности.

Практическая значимость системы подтверждается её потенциальной интеграцией в бизнес-процессы компании «МАСАТО», где она способна существенно ускорить обработку заявок, повысить прозрачность и контроль над арендными операциями.

Направления дальнейшего развития включают:

- добавление функции онлайн-оплаты аренды;

- внедрение уведомлений и напоминаний;

- интеграцию с бухгалтерскими системами;

- реализацию мобильной версии приложения;

- расширение ролевой модели и отчётности.

Таким образом, разработанная система представляет собой эффективный инструмент для автоматизации арендных процессов.

# 

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сэммервиль И. Инженерия программного обеспечения. – 10-е изд. – М.: Вильямс, 2020. – 832 с.

2. Куликов С. В. Проектирование информационных систем. – М.: Форум, 2022. – 288 с.

3. Фаулер М. Архитектура корпоративных приложений. – СПб.: Символ-Плюс, 2020. – 512 с.

4. Борисов А. И. Бизнес-процессы и информационные технологии. – М.: Финансы и статистика, 2021. – 384 с.

5. Фриман А. React и Redux: разработка и архитектура SPA-приложений. – СПб.: Питер, 2023. – 672 с.

6. Холмс Т. Full-Stack Web Development with Node.js and Express. – Packt Publishing, 2022. – 452 p.

7. Хопкроф А., Мотвани Р., Ульман Д. Введение в теорию алгоритмов. – М.: Вильямс, 2019. – 832 с.

8. Фаулер М. Архитектура корпоративных программных приложений. – СПб.: Питер, 2020. – 576 с.

9. Макконнелл С. Совершенный код. Мастер-класс. – М.: «Русская редакция», 2019. – 896 с.

10. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models // <https://www.iso.org/standard/35733.html>.

11. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. – Addison-Wesley, 2003.

12. Sommerville I. Software Engineering. – 10th ed. – Pearson, 2015.

13. Pressman R.S., Maxim B.R. Software Engineering: A Practitioner’s Approach. – McGraw-Hill Education, 2019.

14. Refactoring: Improving the Design of Existing Code (2nd Edition), 2018. – 448 с.

15. Карлссон А. Реактивное программирование с React. – СПб.: Питер, 2022. – 288 с.

16. Holovaty A., Kaplan-Moss J. The Definitive Guide to Django: Web Development Done Right. – Apress, 2009.

17. Postgresql Global Development Group. PostgreSQL Documentation // https://www.postgresql.org/docs/.

18. JWT.IO – JSON Web Tokens Introduction // https://jwt.io/introduction.

19. Тарасов В. Бизнес-процессы: инструменты улучшения. – М.: Инфра-М, 2021. – 312 с.

20. Хопкроф А., Мотвани Р., Ульман Д. Введение в теорию алгоритмов. – М.: Вильямс, 2019. – 832 с.

20. Овчинников А. М. Тестирование программного обеспечения. – М.: Солон-Пресс, 2021. – 256 с.

# 

# ПРИЛОЖЕНИЕ А

**Техническое задание**

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

КАЗАХСТАНСКО–НЕМЕЦКИЙ УНИВЕРСИТЕТ

Факультет Инжиниринга и информационных технологий

Образовательная программа: 6В06106 - «Информационный инжиниринг в экономике»

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на разработка веб приложения для инвестиционной строительной компании «МАСАТО»

**1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

**1.1 Функциональные требования системы**

Целью данной разработки является создание веб-приложения для автоматизации ключевых процессов инвестиционно-строительной компании «МАСАТО». Приложение призвано обеспечить эффективное управление арендой помещений, повысить прозрачность внутренних операций, упростить взаимодействие с клиентами и централизовать хранение всех данных, связанных с арендой – от заявок до договоров.

В разработку включены два функциональных модуля: публичная часть, предназначенная для арендаторов, и административная панель для сотрудников компании. Публичная часть позволяет пользователям регистрироваться, просматривать каталог помещений, фильтровать их по параметрам, подавать заявки и отслеживать статус аренды. Административный модуль обеспечивает обработку заявок, управление помещениями, формирование договоров и администрирование пользовательских данных.

Назначение системы – заменить устаревшие или частично цифровые процессы современным, централизованным и безопасным веб-решением. Это должно повысить оперативность обработки данных, минимизировать ручной труд, улучшить клиентский сервис и обеспечить масштабируемость бизнес-процессов.

**1.2 Область применения**

Областью применения веб-приложения является деятельность компании «МАСАТО», связанная с арендой недвижимости. Система охватывает как внешнее взаимодействие с арендаторами (информирование, заявочная система, подписание договоров), так и внутреннюю организацию работы сотрудников (учёт помещений, статусная обработка заявок, хранение PDF договоров, хранение PDF отчетов).

Платформа используется как арендаторами – через публичный веб-интерфейс, так и администраторами – через защищённую административную панель. Приложение работает на клиентских устройствах через веб-браузеры и не требует установки. В качестве серверной инфраструктуры применяются облачные технологии и микросервисная архитектура, обеспечивающие надёжную и масштабируемую работу системы в различных эксплуатационных условиях. Такой подход позволяет гибко распределять нагрузку между сервисами.

**2 ОБЩЕЕ ОПИСАНИЕ**

**2.1 Пользовательский интерфейс**

Система предоставляет два основных пользовательских интерфейса: клиентский (публичный) и административный. Клиентский интерфейс разработан для арендаторов, которым предоставляется доступ к функциям регистрации, авторизации, просмотра и фильтрации доступных помещений, подачи заявок и отслеживания их статуса. Интерфейс реализован с использованием библиотеки React.js и адаптирован для различных устройств, включая мобильные.

Административная панель предназначена для сотрудников компании и предоставляет расширенный набор функций: управление карточками помещений, обработка заявок, формирование и загрузка договоров аренды, а также управление данными пользователей. Панель разработана с учётом безопасности, включая разграничение прав доступа и защиту административных маршрутов.

**2.2 Аппаратные интерфейсы**

Для функционирования веб-приложения не требуется специфическое оборудование. Пользовательская часть системы работает на любых клиентских устройствах с доступом в интернет и современным веб-браузером. Серверная часть может быть развернута на облачной инфраструктуре или физическом сервере, поддерживающем Docker-контейнеризацию. Минимальные требования к серверу включают процессор с несколькими ядрами, не менее 8 ГБ оперативной памяти и SSD-диск для быстрого доступа к данным.

**2.3 Программные интерфейсы**

Взаимодействие между клиентской и серверной частями системы осуществляется через REST API, реализованный на базе Express.js. API обеспечивает доступ к функциональности управления пользователями, помещениями, заявками и договорами. Кроме того, интерфейс позволяет интеграцию с внешними сервисами, такими как электронная подпись или система уведомлений.

Все внешние и внутренние вызовы API защищены через механизмы авторизации (OAuth 2.0) и реализуют контроль доступа на уровне маршрутов. Структура API проектируется с прицелом на модульность, повторное использование и простоту масштабирования.

**2.4 Коммуникационные интерфейсы**

Система использует стандартные интернет-протоколы для обмена данными. Основной канал передачи данных – защищённый протокол HTTPS с применением SSL/TLS-шифрования. Это обеспечивает безопасность передачи чувствительной информации между клиентом и сервером.

Также предусмотрено использование электронной почты для отправки уведомлений о смене статуса заявки и предоставлении доступа к договорам. Внутреннее взаимодействие между микросервисами осуществляется через REST-запросы или внутренние очереди событий, обеспечивая асинхронность обработки.

**2.5 Ограничение по памяти**

Система должна обеспечивать надёжное хранение больших объёмов структурированных и неструктурированных данных. В качестве хранилища используется реляционная СУБД PostgreSQL для работы с заявками, пользователями, помещениями и договорами, а для хранения медиафайлов и PDF-документов аренды – облачное решение, такое как Amazon S3.

Ограничения по памяти определяются типом размещения системы. В случае облачного хостинга применяется масштабируемая архитектура, позволяющая адаптироваться под растущие объёмы данных. На серверной стороне используется кэширование для ускорения работы с часто запрашиваемыми данными, а также реализованы механизмы архивирования и очистки устаревшей информации.

**2.6 Архитектурный фреймворк системы**

Таблица А.1 – Схема Захмана

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Перспектива | Что? (Данные) | Как? (Функции) | Где? (Сеть) | Кто? (Люди) | Когда? (Время) | Почему? (Мотивация) |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Контекст (Сфера бизнеса) | Помещения, заявки, пользователи, договоры, история аренды | Подача заявки, просмотр помещений, регистрация, генерация договора | Веб-приложение: клиентская часть (React), административная панель | Арендаторы, администраторы | В реальном времени (во время взаимодействия) | Оптимизация аренды и взаимодействия с клиентами |

Продолжение таблицы А1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Перспектива | Что?(Данные) | Как? (Функция) | Где? (Сеть) | Кто? (Люди) | Когда? (Время) | Почему? (Мотивация) |
| Контекст (Сфера бизнеса) | Помещения, заявки, пользователи, договоры, история аренды | Подача заявки, просмотр помещений, регистрация, генерация договора | Веб-приложение: клиентская часть (React), административная панель |  |  |  |
| Концепция (Модель бизнеса) | Сущности: Room, User, RentalRequest, Contract, History | Каталог помещений, фильтрация, заявки, статус, договор | Frontend Backend через REST API | Клиенты и сотрудники компании | С момента подачи заявки до подписания договора | Цифровизация и прозрачность процессов |
| Физика (Технологическая модель) | PostgreSQL, AWS S3, ORM (Sequelize/Prisma) | Модули на Node.js: заявки, помещения, пользователи, договоры | Docker, React, Nginx, Express.js | Разработчики, DevOps | Cron-задачи, API-запросы, обновления в БД | Надёжность, масштабируемость, безопасность |
| Компоненты (Конкретная модель) | Таблицы: users, rooms, rental\_requests, contracts, history | API: POST заявки, PATCH статуса, PDF договор | SPA админка, REST API, PostgreSQL, AWS | Арендатор, админ | Подача заявки, подтверждение, генерация договора | Автоматизация ручных процессов |
| Примечание – составлено автором | | | | | | |

**3 ТРЕБОВАНИЯ К РАЗРАБОТКЕ**

**3.1 Функциональные требования**

**3.1.1 Требования к функциональным характеристикам**

Разрабатываемая система предназначена для автоматизации процесса аренды помещений в компании «МАСАТО». Основной функционал ориентирован на два уровня пользователей: арендаторов и администраторов. Для арендаторов предусмотрена возможность регистрации, авторизации, просмотра помещений, фильтрации по параметрам (этаж, площадь, стоимость), подачи заявок, отслеживания их статуса и доступа к договорам аренды. Администраторы, в свою очередь, могут управлять карточками помещений, обрабатывать входящие заявки, формировать договоры, изменять их статус, а также редактировать данные пользователей.

Все действия пользователей фиксируются и сохраняются в централизованной базе данных. Система должна обеспечивать поддержку таких процессов, как создание и изменение заявок, подтверждение или отклонение администратором, автоматическое формирование договора аренды и загрузка его пользователем. Отдельно должна реализовываться логика смены статуса заявок с автоматическим уведомлением пользователя.

**3.1.2 Исходные данные**

В качестве исходных данных в систему загружается информация о помещениях, включая номер, этаж, площадь, стоимость, статус (свободно/занято), тип помещения и фотографии. Кроме того, создаются учётные записи арендаторов и администраторов. При подаче заявки пользователь вводит свои контактные данные, срок аренды и комментарий. Все данные сохраняются в базе и используются при формировании договоров.

**3.1.3 Результаты**

Результатом использования системы являются зарегистрированные заявки с текущим статусом, подтверждённые договоры аренды в формате PDF, история аренды пользователей, а также структурированная информация в базе данных, доступная для анализа и отчётности. Пользователь получает доступ к этим результатам в личном кабинете, а администратор – через административную панель.

**3.1.4 Организация функциональных требований**

**3.1.4.1 По диаграммам последовательности**

Диаграммы последовательности отражают шаги пользователя на протяжении всего процесса взаимодействия с системой: от регистрации и входа в учётную запись до выбора помещения, подачи заявки на аренду и последующей загрузки сгенерированного договора. Каждое пользовательское действие инициирует цепочку вызовов на стороне сервера, включая отправку запроса к API, обработку и валидацию данных, сохранение информации в базе данных и возврат соответствующего ответа клиентскому интерфейсу.

Для администратора диаграммы демонстрируют последовательность операций, связанных с модерацией поступивших заявок: просмотр деталей, подтверждение или отклонение аренды, формирование договора в формате PDF и прикрепление его в личный кабинет пользователя. Это позволяет наглядно представить внутреннюю логику работы приложения. Представлено на рисунке А.1.

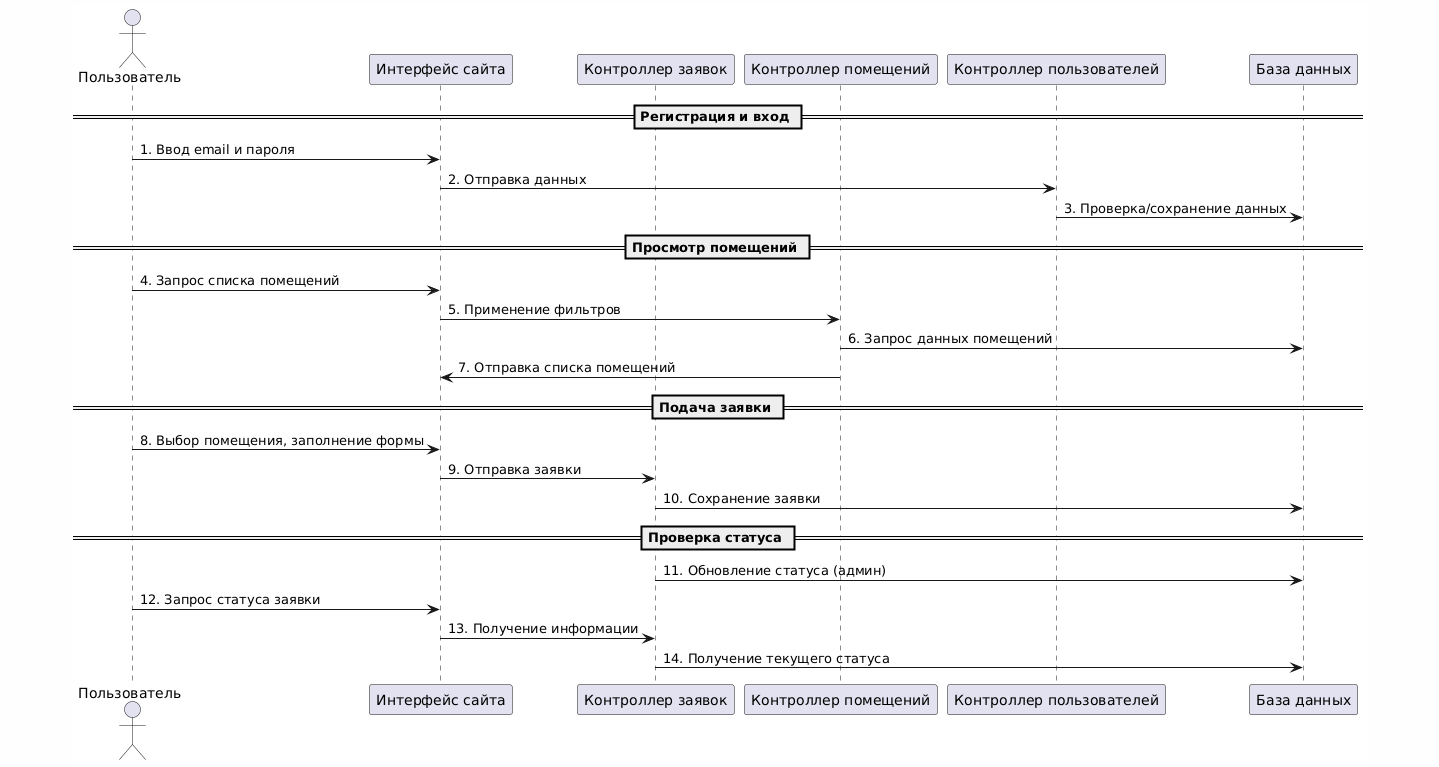


Рисунок А.1 – Диаграмма последовательности (составлено автором)

**3.2 Нефункциональные требования**

**3.2.1 Производительность**

Система должна обеспечивать стабильную и быструю работу при одновременном обслуживании большого количества пользователей. Время отклика пользовательского интерфейса не должно превышать 2 секунд при стандартных операциях (подача заявки, просмотр помещений). Для обеспечения производительности используется асинхронная обработка запросов на серверной стороне и кэширование часто используемых данных. Серверная архитектура предусматривает возможность масштабирования с помощью контейнеризации и балансировки нагрузки, что позволяет обеспечить бесперебойную работу даже при пиковых нагрузках.

**3.2.2 Надежность и доступность**

Веб-приложение должно быть доступно круглосуточно, с гарантированной доступностью не менее 99.5% в месяц. Использование отказоустойчивой архитектуры (кластеризация, резервное копирование, избыточные серверы) позволяет избежать сбоев и потери данных. Все изменения в системе логируются. Для критически важной информации, такой как заявки и договоры, реализуется автоматическое резервное копирование с возможностью восстановления.

**3.2.3 Интерфейсные требования**

Пользовательский интерфейс должен быть реализован в соответствии с принципами адаптивного дизайна и корректно отображаться на различных устройствах (мобильные, планшеты, десктопы). Интерфейс обязан быть интуитивно понятным и соответствовать требованиям доступности: понятная навигация, визуальное выделение статусов, стандартные элементы управления. Серверный интерфейс (REST API) должен иметь чётко определённые маршруты, логическую структуру и возможность подключения внешних систем (например, электронная подпись, системы отчётности).

**3.2.4 Ограничения**

В целях безопасности все действия, связанные с доступом к данным, должны проходить проверку подлинности и авторизации. Доступ к административной панели разрешён только пользователям с соответствующей ролью. Все пароли хранятся в зашифрованном виде с использованием хэширования (например, bcrypt). Передача данных между клиентом и сервером осуществляется по защищённому протоколу HTTPS. Ограничения также касаются загрузки файлов: допустимы только форматы JPEG, PNG (для фото) и PDF (для договоров), с ограничением размера файла не более 10 МБ.

**3.3 Обратные требования**

Разрабатываемая система автоматизации аренды помещений ограничивается задачами, связанными исключительно с арендной деятельностью и не предусматривает выполнение функций, выходящих за рамки этой области. В частности, система не предназначена для управления строительными проектами и не охватывает процессы планирования, проектирования и ведения строительных работ. Все действия связаны исключительно с операциями по аренде уже введённых в эксплуатацию объектов.

Также система не предполагает полноценной интеграции с внешними бухгалтерскими или ERP-системами компании. Несмотря на то, что в функциональность включена возможность онлайн-оплаты аренды, финансовая отчётность, формирование балансов, начисление налогов и контроль бухгалтерских операций остаются вне рамок приложения и реализуются с использованием других инструментов.

Веб-приложение ориентировано исключительно на одного заказчика – компанию «МАСАТО» – и не предназначено для работы в многоклиентском режиме. Архитектура системы не предусматривает разделения данных между несколькими независимыми компаниями или брендами.

Кроме того, пользователи не могут самостоятельно изменять шаблоны юридических документов через интерфейс. Все договоры формируются автоматически на основе заранее утверждённого шаблона. Внесение изменений в структуру договоров осуществляется исключительно на стороне разработчиков или администраторов системы.

Наконец, интерфейс арендатора не включает функции модерации или редактирования контента, связанного с помещениями, заявками других пользователей или документацией. Все административные действия доступны только сотрудникам компании через закрытую административную панель, что обеспечивает централизованное управление и безопасность данных

# ПРИЛОЖЕНИЕ Б

**RequestFormPage.js**

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { useNavigate, useParams } from 'react-router-dom';

function RequestFormPage() {

  const { roomId } = useParams();

  const [formData, setFormData] = useState({

    start\_date: '',

    end\_date: '',

    comment: '',

    iin: ''

  });

  const [error, setError] = useState('');

  const [totalCost, setTotalCost] = useState(0);

  const [roomInfo, setRoomInfo] = useState(null);

  const navigate = useNavigate();

  useEffect(() => {

    const fetchRoomInfo = async () => {

      try {

        const res = await axios.get(`http://localhost:5000/api/rooms/${roomId}`);

        setRoomInfo(res.data);

      } catch (err) {

        console.error('Ошибка загрузки информации о помещении:', err);

      }

    };

    fetchRoomInfo();

  }, [roomId]);

  useEffect(() => {

    if (formData.start\_date && formData.end\_date && roomInfo?.price) {

      const start = new Date(formData.start\_date);

      const end = new Date(formData.end\_date);

      const today = new Date();

      today.setHours(0, 0, 0, 0);

      if (start < today) {

        setError('Дата начала аренды не может быть в прошлом');

        return;

      }

      if (end <= start) {

        setError('Дата окончания должна быть позже даты начала');

        return;

      }

      // Вычисляем разницу в месяцах

      const months = (end.getFullYear() - start.getFullYear()) \* 12 +

                     (end.getMonth() - start.getMonth()) +

                     (end.getDate() >= start.getDate() ? 0 : -1);

      // Минимально 1 месяц

      const totalMonths = Math.max(1, months);

      setTotalCost(totalMonths \* roomInfo.price);

    }

  }, [formData.start\_date, formData.end\_date, roomInfo]);

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData(prev => ({

      ...prev,

      [name]: value

    }));

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    setError('');

    // Валидация ИИН (12 цифр)

    if (!/^\d{12}$/.test(formData.iin)) {

      setError('ИИН должен состоять из 12 цифр');

      return;

    }

    const token = localStorage.getItem('token');

    try {

      await axios.post(

        'http://localhost:5000/api/requests',

        {

          room\_id: roomId,

          start\_date: formData.start\_date,

          end\_date: formData.end\_date,

          comment: formData.comment,

          iin: formData.iin

        },

        { headers: { Authorization: `Bearer ${token}` } }

      );

      alert('Заявка успешно подана!');

      navigate('/requests');

    } catch (error) {

      console.error(error);

      setError(error.response?.data?.message || 'Ошибка при подаче заявки!');

    }

  };

  return (

    <div style={{

      maxWidth: '500px',

      margin: 'auto',

      padding: '20px',

      backgroundColor: '#fff',

      borderRadius: '8px',

      boxShadow: '0 0 10px rgba(0,0,0,0.1)',

      marginTop: '50px'

    }}>

      <h2 style={{ textAlign: 'center', marginBottom: '20px' }}>Подача заявки на аренду</h2>

      {error && (

        <div style={{

          color: 'red',

          marginBottom: '15px',

          padding: '10px',

          backgroundColor: '#ffeeee',

          borderRadius: '4px'

        }}>

          {error}

        </div>

      )}

      <form onSubmit={handleSubmit} style={{ display: 'flex', flexDirection: 'column', gap: '15px' }}>

        <div>

          <label>Дата начала аренды:</label>

          <input

            type="date"

            name="start\_date"

            value={formData.start\_date}

            onChange={handleChange}

            required

            min={new Date().toISOString().split('T')[0]}

            style={{ width: '100%', padding: '8px', borderRadius: '4px', border: '1px solid #ddd' }}

          />

        </div>

        <div>

          <label>Дата окончания аренды:</label>

          <input

            type="date"

            name="end\_date"

            value={formData.end\_date}

            onChange={handleChange}

            required

            min={formData.start\_date || new Date().toISOString().split('T')[0]}

            style={{ width: '100%', padding: '8px', borderRadius: '4px', border: '1px solid #ddd' }}

          />

        </div>

        {roomInfo && totalCost > 0 && (

          <div style={{

            padding: '10px',

            backgroundColor: '#f8f9fa',

            borderRadius: '4px',

            border: '1px solid #dee2e6'

          }}>

            <p><strong>Стоимость аренды:</strong> {totalCost}₸ ({roomInfo.price}₸ × {Math.round(totalCost / roomInfo.price)} мес.)</p>

          </div>

        )}

        <div>

          <label>ИИН (12 цифр):</label>

          <input

            type="text"

            name="iin"

            value={formData.iin}

            onChange={handleChange}

            required

            pattern="\d{12}"

            maxLength="12"

            style={{ width: '100%', padding: '8px', borderRadius: '4px', border: '1px solid #ddd' }}

          />

        </div>

        <div>

          <label>Комментарий:</label>

          <textarea

            name="comment"

            value={formData.comment}

            onChange={handleChange}

            style={{

              width: '100%',

              padding: '8px',

              borderRadius: '4px',

              border: '1px solid #ddd',

              minHeight: '100px'

            }}

          />

        </div>

        <button

          type="submit"

          style={{

            padding: '10px 15px',

            backgroundColor: '#4CAF50',

            color: 'white',

            border: 'none',

            borderRadius: '4px',

            cursor: 'pointer',

            fontSize: '16px',

            marginTop: '10px'

          }}

        >

          Отправить заявку

        </button>

      </form>

    </div>

  );

}

export default RequestFormPage;