МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Разработка программного обеспечения
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка веб-приложения для учета услуг и заказов в сервисной компании»

Обучающийся Д.В. Тарабордин
(Инициалы Фамилия) (личная подпись)

Руководитель Н.Н. Рогова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант канд. филол. наук, доцент М.В. Дайнеко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Выпускная квалификационная работа на тему «Разработка веб-приложения для учета услуг и заказов в сервисной компании» посвящена созданию программного обеспечения для автоматизации ключевых бизнес-процессов в сфере сервисного обслуживания.

Актуальность работы обусловлена необходимостью повышения эффективности, прозрачности и скорости обработки заказов. Объектом исследования является деятельность компании «ВорлдИнтерТех Рус» по работе с заказами и клиентами. Предмет – разработка информационной системы, автоматизирующей прием заказов, управление услугами, взаимодействие с пользователями и формирование отчетности.

В рамках работы выполнены: анализ архитектуры текущей информационной системы, формализация требований (FURPS+), проектирование серверной части, интерфейса и базы данных, реализация клиент-серверного веб-приложения на стеке MERN, проведение функционального и нагрузочного тестирования.

Структура выпускной квалификационной работы включает введение, три главы и заключение. Работа содержит 34 рисунка, 8 таблиц и приложения. По итогам разработки создано программное решение, рекомендованное к внедрению в рамках предприятия, с возможностью масштабирования и дальнейшего развития.

Abstract

The title of the graduation work is «Development of a Web Application for Service and Order Management in a Service Company».

The graduation work consists of an introduction, three chapters, 34 figures, 8 tables, a conclusion, a list of 20 references including foreign sources, and two appendices.

The aim of this graduation work is to develop a web application for managing services and orders in a service company.

The object of the research is the operational activities of the company related to service and order processing.

The subject of the research is the development of software for managing services and orders.

The key issue of the graduation work is the development of a functional and user-friendly application that automates service and order management.

The graduation work may be divided into several logically connected parts: formulation of the development task, software design, development, and testing.

The first part describes the importance of software for the enterprise information system, its main functions, an overview and analysis of existing solutions, and the requirements for functionality and user interface of the application.

The second part outlines the design methodology, structural modeling, selection of technologies and tools, the architecture and interaction of components, and user interface design.

The third part presents an implementation example of the web application, organization of the testing process, functional and load testing, and conclusions based on testing results.

In conclusion, it should be emphasized that the developed web application meets all established requirements and is ready for deployment.

Оглавление

Введение

В условиях активного развития цифровых технологий особую значимость приобретает внедрение программного обеспечения, направленного на повышение эффективности и прозрачности работы предприятий. Это особенно актуально для сервисных организаций, обрабатывающих значительные объемы услуг и заказов, где недостаточный уровень цифровизации ограничивает производительность, способствует росту ошибок и снижает качество обслуживания клиентов.

В компании «ВорлдИнтерТех Рус» рассматривается текущий процесс учета услуг и заказов. Анализ показывает, что отсутствует единая система приема заказов и взаимодействия с клиентами, что обусловливает необходимость разработки специализированного веб-приложения. Разрабатываемое программное обеспечение автоматизирует прием заказов, управление услугами и формирование отчетности, а также обеспечивает разграничение ролей пользователей и интеграцию с существующей ИТ-инфраструктурой.

Объект исследования – деятельность компании по обработке заказов и услуг.

Предмет исследования – разработка программного обеспечения для учета услуг и заказов.

Цель выпускной квалификационной работы – разработка веб-приложения для учета услуг и заказов в сервисной компании.

Для достижения цели необходимо решить следующие задачи:

− провести анализ архитектуры существующей информационной системы предприятия;

− сформулировать функциональные и нефункциональные требования к разрабатываемому программному обеспечению;

− спроектировать структуру серверной части, пользовательского интерфейса и модели базы данных;

− разработать веб-приложение с использованием стека MERN (MongoDB, Express.js, React, Node.js);

− провести функциональное и нагрузочное тестирование реализованной системы.

Работа включает три главы. В первой главе формулируется задача, проводится анализ предметной области, аналогов и требований. Во второй главе рассматривается проектирование архитектуры, интерфейса и логики приложения. В третьей главе описывается реализация и представлены результаты тестирования.

Итогом выполненной выпускной квалификационной работы становится веб-приложение, готовое к внедрению и направленное на повышение эффективности работы сервисной компании.

Глава 1 Постановка задачи на разработку программного обеспечения для информационной системы предприятия

1.1 Значение программного обеспечения для информационной системы предприятия

Программное обеспечение (ПО) представляет собой совокупность программ, алгоритмов и данных, предназначенных для выполнения вычислительных задач на цифровых устройствах. Являясь основой современных информационных систем, ПО обеспечивает автоматизацию процессов, повышение производительности и снижение влияния человеческого фактора.

В условиях цифровой трансформации экономики программное обеспечение приобретает особую значимость в таких сферах, как сервисное обслуживание, где оперативность и точность обработки данных напрямую влияют на качество взаимодействия с клиентами. Несмотря на это, в предметной области по-прежнему широко применяются ручные методы учета, что обуславливает ряд проблем, включая:

– высокую нагрузку на персонал;

– дублирование информации и рост вероятности ошибок;

– низкую прозрачность процессов и отсутствие своевременного контроля;

– сложности доступа к информации и формированию отчетности.

Сервисные компании ежедневно обрабатывают большое количество заказов и услуг, что при отсутствии автоматизации приводит к увеличению времени выполнения операций, ограниченной обратной связи с клиентами и затруднениям в контроле качества предоставляемых услуг. Кроме того, ручная обработка затрудняет аналитическую оценку показателей и принятие управленческих решений.

В компании «ВорлдИнтерТех Рус», являющейся объектом исследования, процессы учета услуг и заказов частично выполнялись вручную, без централизованной базы данных. Это негативно сказывалось на производительности, усложняло планирование и ограничивало доступ к актуальной информации о загрузке специалистов.

Разработка специализированного программного обеспечения необходима для устранения указанных ограничений, повышения эффективности работы и внедрения цифрового инструмента, обеспечивающего взаимодействие между клиентами, ремонтными мастерами и администрацией. Для систематизации факторов, влияющих на текущую проблему, построена диаграмма «Причина–следствие», изображенная на рисунке 1.

Рисунок 1 – Диаграмма «Причина-следствие»

Диаграмма отражает основные группы причин: человеческий фактор, методы работы, организацию среды, инфраструктуру и данные. Среди ключевых проблем – ручной ввод, перегрузка сотрудников, отсутствие стандартов и прозрачности, слабая коммуникация, низкое качество данных и

ошибки в отчетности. Эти взаимосвязанные недостатки обосновывают необходимость внедрения автоматизированной системы с централизованной базой данных, улучшенным контролем и стандартизированными процессами.

Одним из примеров успешного внедрения цифровых решений является «Сервисный центр». Программный продукт позволяет автоматизировать ключевые процессы: прием и обработку заказов, контроль статуса ремонта, взаимодействие с клиентами, формирование отчетности и управления складом.

Внедрение «Сервисного центра» помогает решать типичные проблемы, а именно:

– избыточная ручная обработка данных и высокая зависимость от человеческого фактора;

– отсутствие централизованной базы заказов и услуг, слабая прозрачность для клиента;

– перегрузка персонала при большом объеме заказов.

Автоматизация с помощью ПО решает эти проблемы за счет:

– единого интерфейса для регистрации и отслеживания заявок;

– уведомлений и напоминаний по заказам;

– возможности генерировать отчеты и вести статистику по загрузке и выполнению услуг.

На практике это приводит к следующим улучшениям:

– сокращение времени на прием и обработку заказов;

– рост клиентской удовлетворенности благодаря прозрачному и удобному интерфейсу;

– повышение производительности сотрудников, за счет снижения рутинной нагрузки.

На рисунке 2 представлены обобщенные показатели, демонстрирующие улучшения до и после внедрения системы.

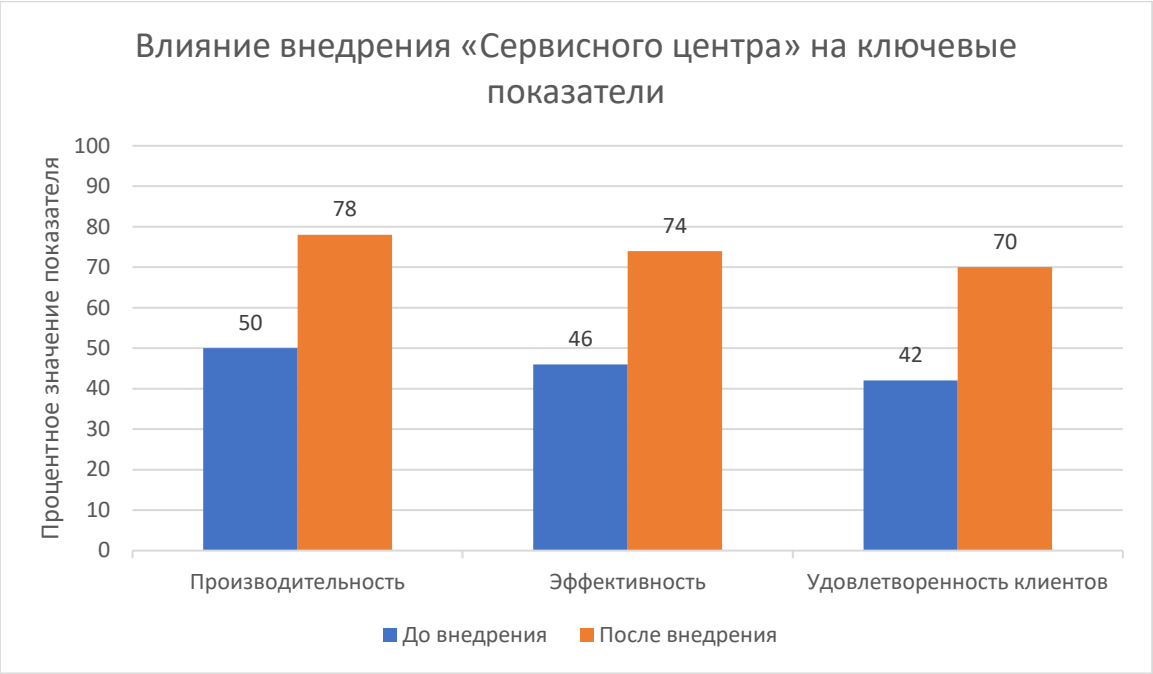


Рисунок 2 – Влияние внедрения «Сервисного центра» на производительность, эффективность и удовлетворенность клиентов

Программное обеспечение оказывает комплексное влияние на внутренние процессы и внешний клиентский опыт. Разработка и внедрение решений, аналогичных «Сервисному центру», рассматривается как стратегически важный шаг в повышении устойчивости и конкурентоспособности организаций в сфере сервисных услуг.

Рост объемов данных, развитие облачных технологий и широкое распространение мобильных устройств формируют устойчивый спрос на масштабируемые и безопасные цифровые решения. Предприятия, не адаптирующиеся к этим изменениям, теряют позиции на рынке.

Разработка индивидуального веб-приложения для «ВорлдИнтерТех Рус» представляет собой не только способ устранения текущих проблем, но и основу для будущих инноваций – аналитики, уведомлений, интеграции с внешними сервисами и интеллектуального анализа данных. Программное обеспечение выступает не только средством автоматизации, но и основой для долгосрочного развития и цифровой адаптивности организации.

Экономические преимущества внедрения цифровых решений включают:

– сокращение затрат за счет отказа от бумажного документооборота и ручной обработки;

– оптимизацию использования ресурсов и повышение производительности персонала;

– увеличение пропускной способности компании;

– рост клиентской лояльности за счет доступности и прозрачности сервиса.

На уровне общества цифровизация сервисных процессов упрощает доступ к услугам и повышает общий уровень удовлетворенности. Возможность самостоятельной онлайн-записи, получения уведомлений и контроля статуса заказов и услуг становится стандартом современного клиентского взаимодействия.

Таким образом, программное обеспечение выступает как ключевой элемент цифровой трансформации и стратегического развития, обеспечивая эффективность, надежность и гибкость бизнеса в условиях современной экономики.

1.2 Основные функции программного обеспечения для информационной системы предприятия

Перед разработкой веб-приложения по учету заказов и услуг необходимо провести анализ текущего состояния архитектуры информационных технологий предприятия. Это позволит определить реализованные компоненты информационной системы, на каких платформах построены, а также в какой степени удовлетворяют потребности бизнеса. Такой подход обеспечивает выбор оптимального пути интеграции нового модуля в существующую ИТ-инфраструктуру.

Внедряемый модуль разрабатывается с учетом действующей архитектуры компании, представленной на рисунке 3, где отображены ключевые слои и их роль в цифровой экосистеме предприятия.

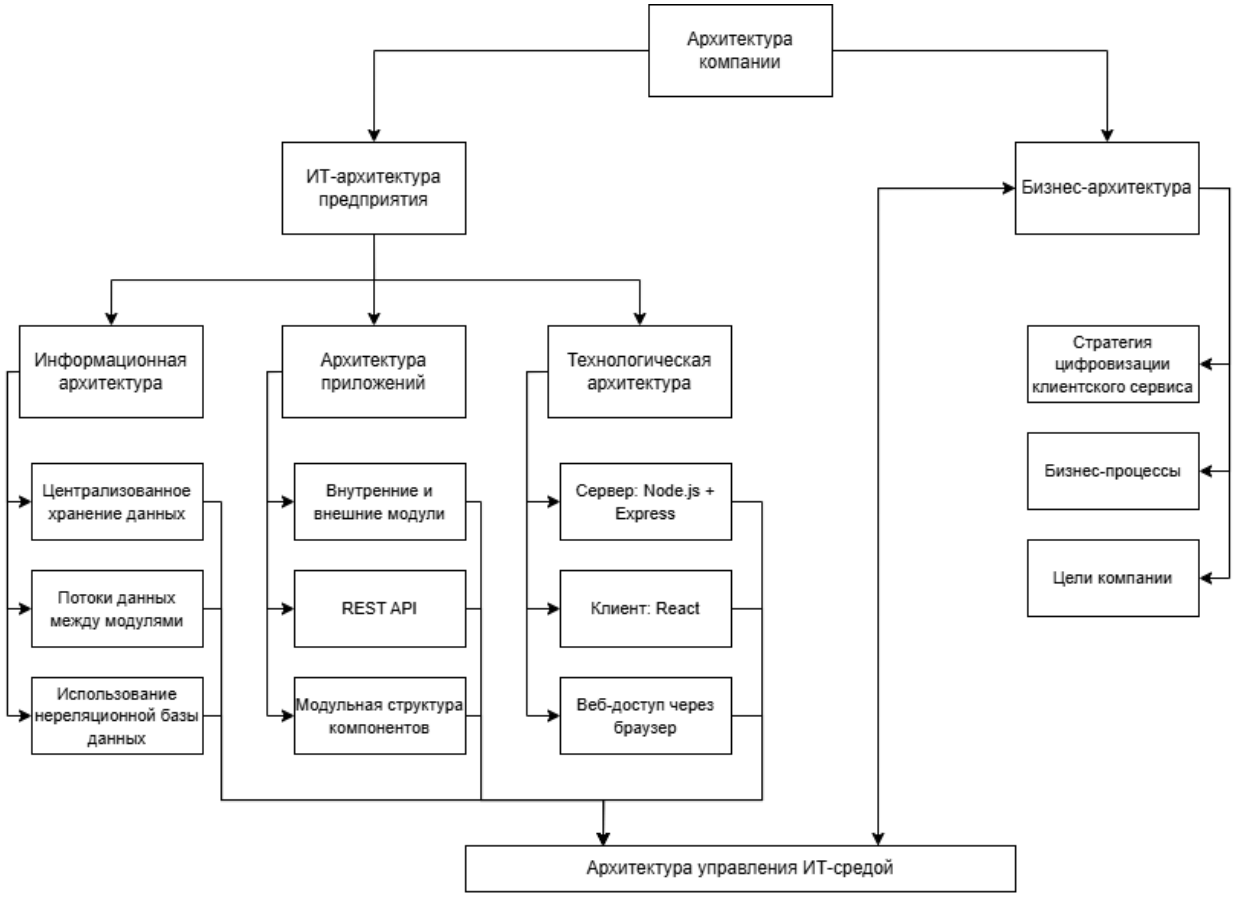


Рисунок 3 –Архитектурная модель организации

Инфраструктурный слой представлен аппаратными и сетевыми ресурсами, обеспечивающими работоспособность и надежность программных решений. Слой данных реализован на базе централизованного хранилища, использующего нереляционную базу данных, что позволяет эффективно управлять разнородными данными и оперативно обрабатывать большие объемы информации.

На прикладном уровне функционирует внутренняя информационная система, реализованная с использованием технологического стека MERN (MongoDB, Express.js, React, Node.js). ИС автоматизирует отдельные

внутренние процессы компании, такие как хранение информации о клиентах, формирование отчетности, мониторинг загрузки персонала. В текущий момент отсутствует модуль, обеспечивающий полноценное взаимодействие с клиентами, прием и обработку онлайн-заказов и выдачу отчетов по завершенным услугам.

В рамках действующей системы используются следующие программные средства:

– внутренняя панель на базе React, предназначенная для работы сотрудников, содержит функции просмотра базы клиентов, заказов и административных настроек. Система обладает удобным интерфейсом, однако не охватывает пользовательский опыт со стороны клиента и не обеспечивает возможности самостоятельного бронирования услуг.

– средства коммуникации – для связи с клиентами применяются мессенджеры, телефон и электронная почта. Данные инструменты не интегрированы в информационную систему, что делает невозможным автоматизированный учет и анализ обращений.

Проведенный анализ показал, что существующая информационная система ориентирована на внутренние потребности и не охватывает весь цикл обслуживания клиентов. Это снижает уровень прозрачности, замедляет выполнение операций и требует значительных трудозатрат со стороны персонала. Возникает необходимость разработки нового модуля, обеспечивающего цифровое взаимодействие с клиентами, автоматизированный прием заказов на услуги и их учет.

Разрабатываемое программное обеспечение будет представлено в виде веб-приложения, взаимодействующего с серверной частью через REST API. Выбор технологий основан на уже применяемом в компании MERN-стеке, что обеспечит совместимость и упростит процесс интеграции. Внедрение модуля позволит реализовать:

– веб-интерфейс для клиентов с функцией авторизации, выбора услуг и оформления заказов;

– автоматизированную обработку заказов с присвоением статусов;

– подключение к существующей базе данных и использование общей модели данных;

– формирование отчетности на стороне администратора;

– разграничение прав доступа между клиентами и сотрудниками через систему авторизации.

Таким образом, разрабатываемое программное обеспечение органично встраивается в прикладной уровень архитектуры предприятия, усиливая связь между бизнес-процессами и цифровыми сервисами. Это решение позволит устранить существующие ограничения и значительно повысить эффективность работы с клиентскими заказами.

Одной из важнейших функций программного обеспечения является обработка данных, которая включает сбор, хранение, анализ и извлечение информации. Эта функция критична для компаний, где данные о клиентах и их заказах используются для анализа и управления процессами.

Корректно организованная система обработки данных позволяет получать информацию в режиме реального времени и проводить аналитические оценки, выявлять тенденции и принимать обоснованные управленческие решения. В частности, в контексте сервисной компании, обработка данных обеспечивает:

– регистрацию информации о клиентах и заказах;

– формирование и хранение истории заказов;

– анализ текущей загруженности;

– генерацию отчетов по завершенным работам.

Для формализации и визуального описания бизнес-процессов, предшествующих автоматизации, применяется диаграмма потоков данных (DFD) – один из наиболее распространенных инструментов структурного анализа. Диаграмма позволяет наглядно представить, как информация перемещается внутри системы, через какие процессы проходит, какие

хранилища задействуются, а также с какими внешними сущностями осуществляется взаимодействие.

Диаграмма потоков данных, приведенная на рисунке 4, описывает текущий порядок обработки заказов от клиентов до внедрения автоматизированного программного обеспечения. Диаграмма демонстрирует последовательность действий между клиентом, менеджером, ремонтным мастером и хранилищами информации. Это позволяет выявить узкие места, ручные операции и потенциальные точки автоматизации.

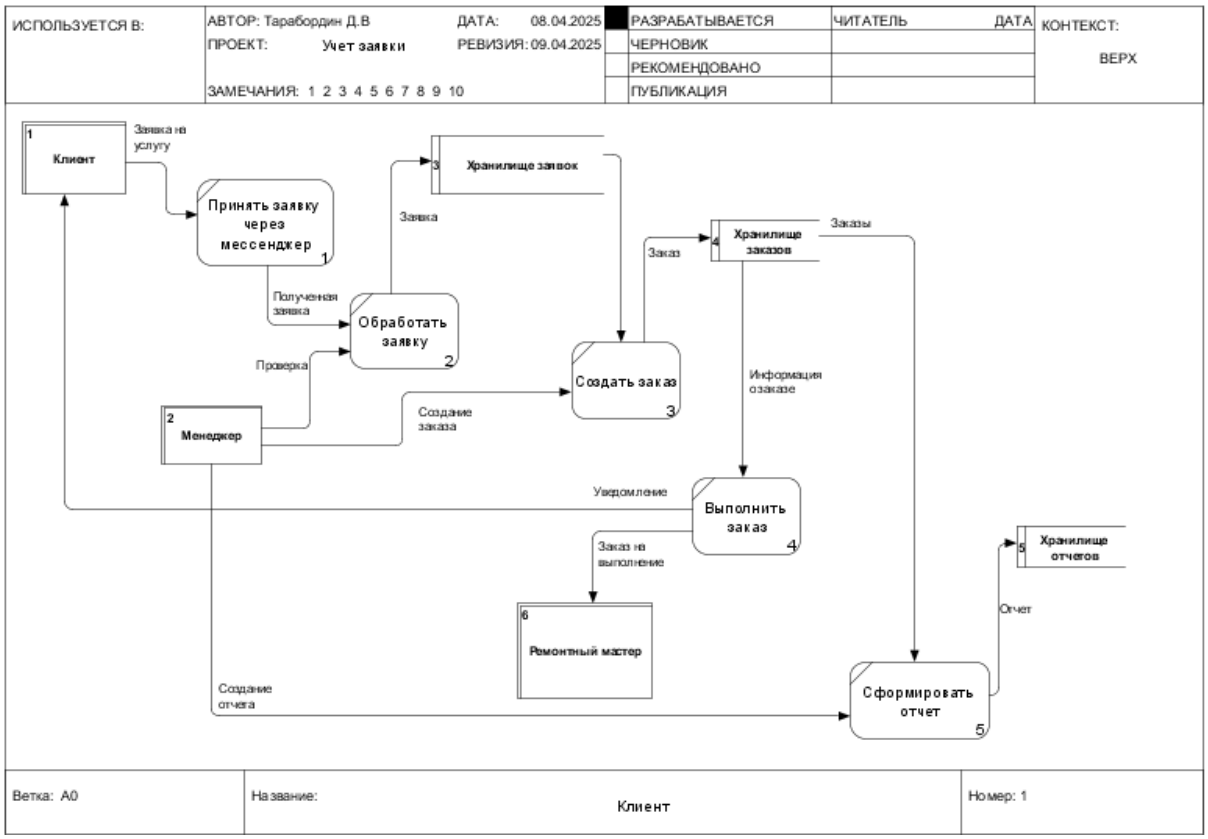


Рисунок 4 –Диаграмма потока данных

Процесс начинается с того, что клиент отправляет заявку на услугу через мессенджер. Заявка фиксируется в процессе «Принять заявку через мессенджер» и передается на обработку.

В процессе «Обработать заявку» менеджер вручную проверяет доступность услуги. При положительном решении данные заявки сохраняются в хранилище заявок и используются для формирования заказа.

На этапе «Создать заказ» менеджер оформляет заказ, который записывается в хранилище заказов. Далее через процесс «Выполнить заказ» информация передается ремонтному мастеру, а клиент получает уведомление.

После выполнения заказ попадает в процесс «Сформировать отчет», где менеджер вручную подготавливает отчет, используя данные из хранилища заказов. Итоговый документ сохраняется в хранилище отчетов.

Диаграмма иллюстрирует текущее состояние бизнес-процесса до автоматизации: большинство операций выполняется вручную, взаимодействие происходит через мессенджеры и почту, а данные хранятся локально. Это подчеркивает необходимость внедрения веб-приложения для повышения скорости, прозрачности и эффективности обслуживания клиентов.

Программное обеспечение также играет важную роль в управлении ресурсами предприятия. К таким ресурсам относятся:

– человеческие – сотрудники компании и их занятость;

– материальные – оказываемые услуги, оборудование, документы;

– временные – расписание выполнения работ и сроки заказов.

Разрабатываемое веб-приложение позволяет:

– распределять заказы и услуги между сотрудниками;

– отслеживать актуальные и завершенные заказы;

– оптимизировать нагрузку на персонал;

– формировать отчеты, что облегчает контроль и планирование.

Таким образом, программное обеспечение повышает прозрачность процессов и снижает нагрузку на сотрудников за счет централизованного учета и удобного интерфейса.

Одной из ключевых функций современного программного обеспечения является обеспечение удобного и интуитивно понятного взаимодействия с

пользователями. Качество интерфейса напрямую влияет на то, насколько эффективно пользователь может работать с системой.

Разрабатываемое веб-приложение предоставляет:

– чистый и оптимизированный интерфейс для ПК;

– структурированную навигацию, обеспечивающую быстрый доступ к услугам, заказам, уведомлениям и профилю;

– строку поиска и фильтры для быстрого нахождения нужных услуг;

– панель администратора для управления заказами, услугами, сотрудниками и отчетами.

Одним из ключевых преимуществ современного программного обеспечения является возможность автоматизации рутинных и повторяющихся операций. Это особенно актуально для сервисных компаний, где необходимо быстро обрабатывать множество однотипных задач, связанных с оформлением, выполнением и контролем заказов.

В рамках разрабатываемого веб-приложения автоматизированы следующие процессы:

– обновление доступного времени услуг после оформления заказа;

– присвоение и изменение статусов заказов и услуг;

– формирование отчетов на основе информации о заказах, услугах и мастерах;

– проверка уникальности данных при регистрации и создании заказов;

– сокращение числа ручных операций при работе с заказами.

Функции позволяют снизить нагрузку на сотрудников, ускорить обработку заказов, а также повысить точность выполнения операций.

Одной из главных задач любого информационного системы является обеспечение безопасности данных. В условиях растущих угроз кибербезопасности крайне важно гарантировать защиту информации на всех уровнях взаимодействия с системой.

Для защиты данных, а также предотвращения несанкционированного доступа, веб-приложение использует несколько уровней безопасности:

Аутентификация – процесс проверки подлинности пользователя при входе в систему. Используется сессионная аутентификация через логин и пароль с дополнительной защитой через хэширование паролей [16].

Авторизация – разграничение прав доступа. Клиенты имеют доступ только к просмотру и бронированию услуг и формированию заказов, в то время как администраторы могут редактировать данные и генерировать отчеты.

Шифрование данных – все персональные данные и данные заказов, передаваемые через сеть.

Каждый из этих уровней представляет собой важную защиту на своем этапе взаимодействия с пользователем и сервером, что минимизирует риски утечки данных и повышает уровень доверия со стороны клиента.

Разрабатываемое программное обеспечение выполняет критически важные функции, направленные на оптимизацию процессов и повышение общей эффективности информационной системы предприятия. Веб-приложение включает инструменты для управления данными, ресурсами, пользовательским взаимодействием, а также для автоматизации повторяющихся операций:

– обработка данных обеспечивает прозрачность и точность операций;

– управление ресурсами минимизирует затраты и повышает производительность;

– взаимодействие с пользователями улучшает клиентский опыт и ускоряет процессы;

– автоматизация процессов значительно сокращает временные затраты и минимизирует ошибки, повышая общую эффективность;

– безопасность данных обеспечивается многослойной защитой, включая аутентификацию, авторизацию и шифрование, что гарантирует сохранность информации.

Понимание этих функций помогает более эффективно использовать ПО для решения конкретных задач и достижения поставленных целей компании,

а также способствует улучшению качества обслуживания клиентов и поддержанию конкурентоспособности.

1.3 Обзор существующих решений и их анализ

Представлен обзор существующих программных решений для учета услуг и заказов в сервисных компаниях. Рассматриваются два основных продукта, их функциональные возможности, преимущества и недостатки, а также проводится сравнительный анализ для выбора наиболее подходящего решения в зависимости от потребностей.

Анализ программных решений проводится по следующим критериям:

– функциональность – степень соответствия системы поставленным задачам;

– стоимость – финансовые затраты на приобретение, внедрение и поддержку;

– удобство использования – оценка интерфейса и взаимодействия с пользователем;

– поддержка – доступность технической помощи и документации;

– безопасность – уровень защиты данных и предотвращение несанкционированного доступа.

а) Функциональность:

1) «Сервисный центр» предлагает более широкие функциональные возможности, включая интеграцию с другими системами, работу с несколькими филиалами и более детализированное управление заказами;
2) «AltSC» имеет более узкую направленность, предлагая простые функции для малого бизнеса, что делает его менее гибким для крупных компаний;

б) Стоимость:

1) «Сервисный центр» имеет более высокую стоимость, что может быть оправдано расширенной функциональностью и возможностями настройки;
2) «AltSC» значительно дешевле, что делает его более доступным для небольших сервисных центров с ограниченным бюджетом;

в) Удобство использования:

1) Оба решения предлагают интуитивно понятный интерфейс, но «AltSC» имеет более простую навигацию, ориентированную на малый бизнес. В то время как «Сервисный центр» требует более длительного времени на освоение из-за своей функциональности;

г) Поддержка:

1) «Сервисный центр» предоставляет отличную техническую поддержку, включая обучающие материалы и консультации по индивидуальным настройкам;
2) «AltSC» предлагает ограниченную поддержку, что может затруднить решение нестандартных вопросов;

д) Безопасность:

1) Обе системы используют стандарты безопасности, включая шифрование и аутентификацию, но «Сервисный центр» имеет более продвинутые функции защиты данных, такие как резервное копирование и более детализированные настройки безопасности.

Каждому из критериев присваивается оценка по шкале от 1 до 5, где 1 означает минимальное удовлетворение критерия, а 5 – максимальное.

Результаты сравнительного анализа представлены в таблице 1.

Таблица 1 – Сравнительный анализ программных решений

Критерии	Сервисный центр	AltSC
1	2	3
Функциональность	5	4
Стоимость	3	5
Удобство использования	4	4

Продолжение таблицы 1

Поддержка	5	3
Безопасность	4	4
Итого:	21	20

Проведенный анализ существующих решений в области автоматизации сервисных услуг показал, что ни одно из изученных программных продуктов не удовлетворяет в полной мере предъявляемым требованиям. Несмотря на наличие функционала, способного частично покрыть базовые задачи, каждое из решений имеет ограничения, которые снижают эффективность его применения в условиях реальной эксплуатации.

«Сервисный центр» обладает широкими возможностями, однако его высокая стоимость, избыточная для малого и среднего бизнеса функциональность, а также зависимость от внешнего разработчика затрудняют оперативную адаптацию под индивидуальные потребности. Сложность настройки и необходимость длительного обучения персонала также препятствуют быстрому внедрению.

«AltSC», в свою очередь, предлагает простое и доступное решение, но ориентировано преимущественно на предприятия с типовым набором задач. Недостаточная гибкость, ограниченная поддержка и отсутствие возможности расширения функциональности делают его непригодным для масштабируемых сценариев или интеграции с другими системами.

С учетом специфики бизнес-процессов организации, наличия уникальных требований к интерфейсу, архитектуре и логике обработки данных, наиболее рациональным и обоснованным решением является разработка собственного веб-приложения. Такой подход позволит:

– реализовать строго необходимый функционал без избыточных модулей;

– учесть внутренние процессы компании, не поддающиеся стандартной автоматизации;

– обеспечить гибкость в масштабировании и доработке;

– снизить зависимость от сторонних разработчиков и лицензионных ограничений;

– внедрить индивидуальные механизмы безопасности и отчетности;

– достичь оптимального соотношения затрат и эффективности.

Таким образом, разработка собственного программного обеспечения представляет собой стратегически обоснованное решение, направленное на повышение эффективности управления заявками, улучшение клиентского опыта и достижение максимальной адаптации информационной системы под цели предприятия.

1.4 Требования к функционалу и интерфейсу приложения

Рассмотрим модель классификации требований FURPS+, которая помогает систематизировать и структурировать их для более понятного описания и последующего анализа. FURPS была разработана компанией Hewlett-Packard (HP) в 1980-х годах. «Модель FURPS/FURPS+ получила свое название по первым буквам основных категорий показателей качества программного обеспечения:

– Functionality – функциональность, которая включает в себя в качестве дополнительных показателей особенности, возможности и безопасность используемого программного обеспечения в инновационных программных проектах;

– Usability – практичность, которая включает в себя в качестве дополнительных показателей возможность учета человеческого фактора, эргономичность и наличие полного комплекта пользовательской документации (инструкции, правила инсталляции и деинсталляции и др.);

– Reliability – надежность, которая включает в себя в качестве дополнительных показателей частоту отказов в работе программного обеспечения, наличие возможностей для восстановления информации, а также прогнозируемость действий пользователя в ближайшей перспективе;

– Performance – производительность, которая включает в себя в качестве дополнительных показателей время отклика, пропускную способность обработки определенных объемов информации, точность получаемых решений, простоту и доступность программного обеспечения для пользователей, использование всех возможностей информационных ресурсов обработки данных;

– Supportability – эксплуатационная пригодность, которая включает в себя в качестве дополнительных показателей возможности тестирования программного обеспечения и его отдельных компонентов (модулей), его способности к расширению до более совершенных версий, уровень адаптируемости используемого программного обеспечения, необходимость организации сопровождения в процессе эксплуатации, возможности совместимости используемого программного обеспечения с другими информационными программами и платформами, возможности изменения конфигурации, простоту обслуживания в процессе эксплуатации, стандартные требования к установке, а также возможности локального функционирования»[9], [19].

Указанные требования охватывают различные стороны процесса разработки программного обеспечения. Знак «+» добавлен для учета дополнительных факторов, например, безопасность или масштабируемость.

Функциональные требования:

а) управление пользователями

1) регистрация по ФИО, телефону, почте и паролю с проверкой уникальности и согласием на обработку персональных данных [13];

2) аутентификация/авторизация с выдачей JWT-токена. разграничение доступа по ролям: клиент, мастер или администратор;

3) администратор может добавлять, редактировать и удалять мастеров через интерфейс управления персоналом;

4) клиенты, мастера и администратор могут редактировать личные данные (ФИО, телефон и почту), а также выходит из аккаунта;

б) работа с услугами и заказами:

1) отображение списка услуг с возможностью поиска по ключевым словам и фильтрации по типу и стоимости;

2) просмотр описания, цены и доступных временных слотов;

3) администратор имеет доступ к функциям добавления, редактирования и удаления услуг (CRUD-операции);

в) оформление и управление заказами:

1) добавление одной или нескольких услуг в корзину с выбором доступных даты и времени (прошедшее время недоступно);
2) возможность удаления отдельных услуг из корзины, а также полной очистки содержимого;
3) при оформлении заказа указывается место оказания услуги (офис или выезд), способ оплаты (наличные или онлайн), комментарий и данные клиента;
4) после оформления заказ попадает в список заказов клиента, разделенный на «Актуальные» и «Завершенные»;
5) каждая услуга в заказе отображает индивидуальный статус: «В очереди», «В работе», «На проверке», «Готово», «Отменено»;
6) клиент может отменить услугу до момента выполнения;
7) мастер и администратор могут менять статус услуг, при этом действия подтверждаются через модальное окно;
8) назначение мастера может происходить вручную (админом) или через самоназначение (мастером);
9) канбан-доска отображает заказы по статусам, доступна мастерам и администраторам, статус заказа определяется по минимальному статусу всех входящих в него услуг;

10) реализован расширенный просмотр заказов с фильтрацией по клиенту, дате, статусу, типу и названию услуги;

г) отчетность:

1) отчеты формируются по заказам и услугам;

2) реализованы три типа отчетов:

– финансовый отчет с общей суммой, детализацией по заказам и экспортом в Excel;

– статистический отчет с графиками, количественной информацией по статусам, топ-услугам и средней загрузкой;

– отчет по исполнителям, содержащий услуги и итоговую сумму по каждому мастеру;

3) фильтрация по типу услуг, датам и мастерам, экспорт в Excel, визуализация графиков;

д) уведомления:

1) уведомления отображаются в виде пуш-сообщений и в отдельной вкладке «Уведомления»;

2) клиент получает уведомления об изменении статусов, отмене или завершении услуг, а мастер о назначении/самоназначении и отмене;

3) клиент может отметить все уведомления как прочитанные;

4) в базе данных хранятся до 20 уведомлений на пользователя, старые удаляются автоматически;

е) дополнительные возможности по ролям

1) неавторизованный пользователь:

– предоставляется доступ к поиску и фильтрации услуг, просмотру описания и доступных дат;

– при попытке добавить услугу в корзину отображается форма авторизации;

– доступны страницу «Главная» и «Контакты»;

2) для администратора:

– интерфейс администратора включает быстрый доступ к разделам: «Заказы», «Услуги», «Сотрудники» и «Отчеты»;

– администратор может управлять услугами и сотрудниками (CRUD-операции);

– доступен полный контроль над заказами через канбан-доску и расширенную таблицу заказов;

Нефункциональные требования:

а) производительность

1) время отклика не более 2 секунд;

2) поддержка до 100 одновременных пользователей;

б) безопасность

1) все персональные данные и данные заказов должны быть зашифрованы при передаче и хранении;

2) защита маршрутов и данных: реализовано разграничение доступа по ролям, валидация на клиенте и сервере;

3) реализовано резервное копирование;

в) удобство использования

1) интерфейс должен быть интуитивно понятным и не требовать технических знаний;

2) основные функции должны быть доступны с главной страницы через логичную навигацию;

3) формы содержат валидацию, всплывающие подсказки и предупреждение об ошибках;

4) реализована система уведомлений: всплывающие сообщения и отдельный центр уведомлений;

5) используется стандартные элементы управления (кнопки, выпадающие списки, поля ввода) с визуальной обратной связью;

г) сопровождаемость и расширяемость

1) приложение должно иметь модульную архитектуру;

2) «изменение одного фрагмента системы не должно влиять на ее другие фрагменты»[11].

д) ограничения

1) регистрация пользователя возможна только при согласии на обработку персональных данных;

Сформулированные в данной главе требования образуют методологическую основу для проектирования приложения с учетом логики бизнес-процессов и специфики пользовательского взаимодействия. Разграничение спецификаций на функциональные и нефункциональные аспекты позволило охватить как технические характеристики, так и эксплуатационные условия использования системы.

Систематизированные требования представлены в таблице 2.

Таблица 2 – Сформированные требования

Функциональные требования	Нефункциональные требования
Регистрация, аутентификация/авторизация, разграничение ролей	Интуитивно интерфейс, логичная навигация
Поиск, фильтрация и просмотр услуг	Высокая доступность (отклик не более 2 секунд), поддержка 100 пользователей
Оформление заказов с выбором параметр и отслеживанием статусов	Шифрование данных
Управление услугами, заказами и сотрудниками (CRUD)	Модульная архитектура, масштабируемость
Канбан-доска и расширенный просмотр заказов	Обновление без прерывания работы
Формирование отчетов (финансовый, статистический, по мастерам)	Резервное копирование
Система уведомлений	Валидация данных, подсказки, визуальная обратная связь
Личный кабинет клиента, мастера, администратора	Простота использования веб-приложения
Ограниченный доступ для неавторизованных пользователей	Регистрация только при согласии на обработку персональных данных

«Диаграмма вариантов использования описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Диаграмма вариантов использования

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

Разработка диаграммы вариантов использования преследует такие цели:

– определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;

– сформулировать общие требования к функциональному поведению проектируемой системы;

– разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей»[10].

На рисунке 6 представлена диаграмма вариантов использования.

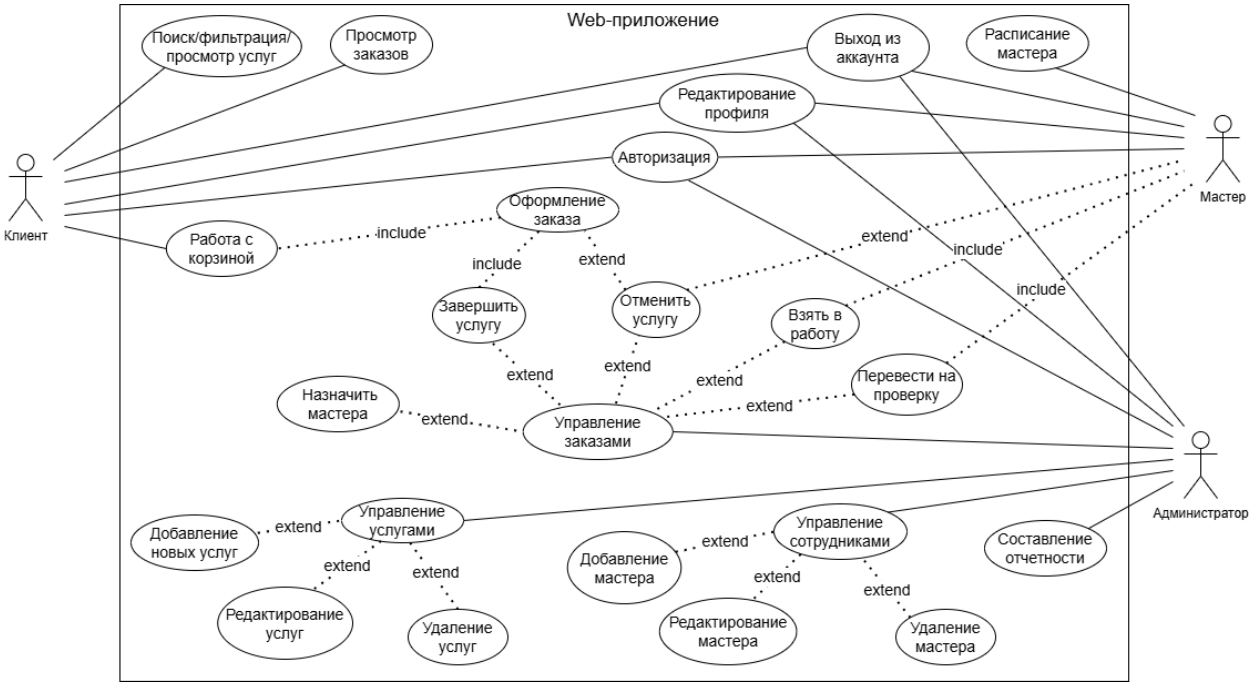


Рисунок 5 – Диаграмма вариантов использования

Далее в приложении А в таблицах А1-15 представлены спецификации диаграммы вариантов использования.

В результате анализа и систематизации требований определены основные функциональные и нефункциональные характеристики веб-приложения, направленные на автоматизацию процессов учета заказов и

услуг, а также взаимодействия с клиентами. Применение модели FURPS+ позволило структурировать требования и охватить все ключевые аспекты: от авторизации пользователей до обеспечения безопасности и масштабируемости.

Формализованные спецификации создают прочную основу для проектирования архитектуры, интерфейса и логики приложения. Визуализация в виде диаграмм и таблиц способствует лучшему пониманию сценариев работы системы, поддерживает взаимодействие между разработчиком и заказчиком, а также облегчает планирование этапов реализации и тестирования.

Представленные в главе требования являются фундаментом для перехода к следующему этапу – проектированию структуры информационной системы, описанию компонентов и взаимодействий между ними.

Выводы по первой главе

В результате проведенного анализа предметной области были выявлены ключевые проблемы, связанные с отсутствием автоматизации учета услуг и заказов, низкой прозрачностью процессов и перегрузкой персонала. Обоснована необходимость разработки собственного веб-приложения, позволяющего устранить данные ограничения. Принято решение о создании индивидуального программного продукта с учетом архитектуры предприятия. На основе модели FURPS+ сформулированы функциональные и нефункциональные требования к разрабатываемому программному обеспечению.

Глава 2 Проектирование программного обеспечения

2.1 Методология проектирования

Методология проектирования на основе UML (Unified Modeling Language), «который предназначен для описания, визуализации и документирования объектно-ориентированных систем и бизнес-процессов с ориентацией на их последующую реализацию в виде программного обеспечения»[10].

Методология обладает определенным рядом достоинств:

– стандартизированность – UML является признанным международным стандартом моделирования программных систем;

– гибкость – поддержка объектно-ориентированного и сервисно-ориентированного проектирования;

– наглядность – облегченный анализ, разработка и сопровождение программного продукта за счет визуального представления системы;

– универсальность – применение в различных областях разработки;

– документирование – диаграммы помогают формализовать требования и упростить понимание системы.

Для проектирования системы используются следующие UML-диаграммы:

– диаграмма вариантов использования – описывает основные сценарии взаимодействия пользователей с системой;

– диаграмма классов – определяет основные сущности системы и их атрибуты;

– диаграмма последовательности – моделирует взаимодействие между объектами во времени, демонстрируя порядок вызовов и сообщений, необходимых для выполнения определенного сценария;

– диаграмма состояния – отображает возможные состояния объекта в системе переходы между ними в зависимости от событий и условий;

– диаграмма компонентов – представляет архитектурные элементы системы и их связи;

– диаграмма развертывания – демонстрирует, как программные компоненты разворачиваются на физическом оборудовании.

2.2 Структурное моделирование

«Диаграмма классов описывает типы объектов системы и различного рода статические отношения, которые существуют между ними. На диаграммах классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами» [12]. На рисунке 6 представлена диаграмма классов.

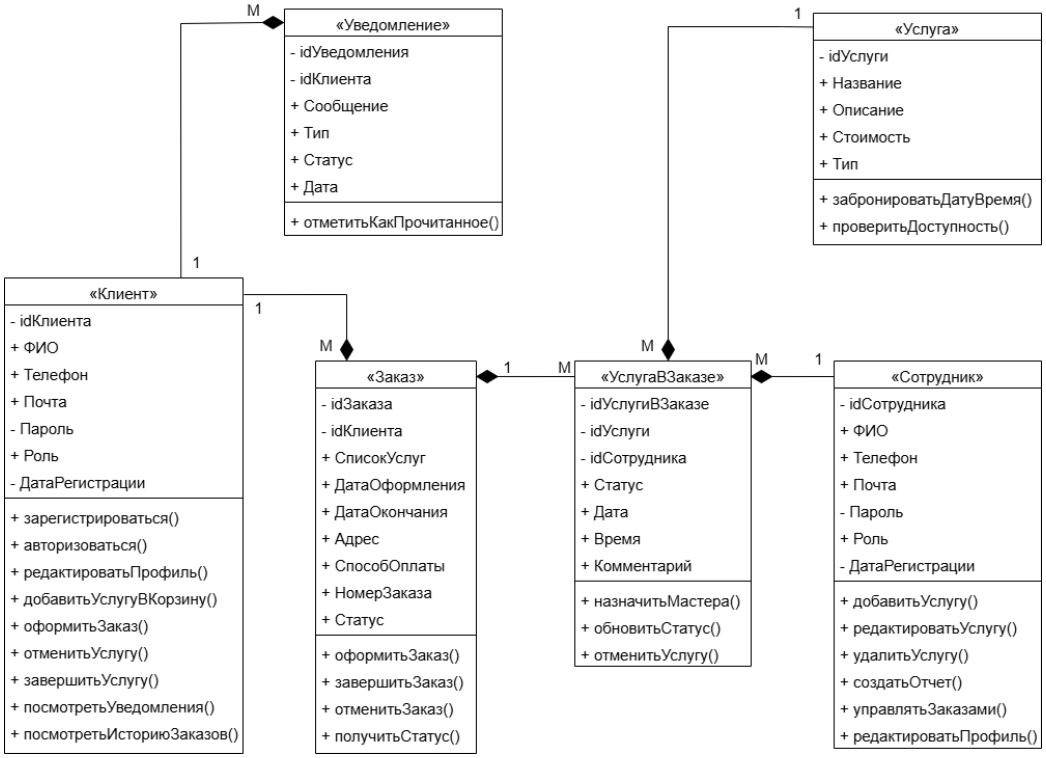


Рисунок 6 – Диаграмма классов

Описание классов веб-приложения:

– клиент – класс, представляющий пользователя системы. Поддерживает регистрацию, авторизацию, редактирование профиля, оформление и отмену услуг, а также взаимодействие с системой уведомлений и просмотр истории заказов;

– сотрудник – класс, описывающий пользователя с ролью администратора или исполнителя. Реализует функции управления услугами, работы с заказами и генерации отчетов. Связан с назначением и выполнением конкретных услуг;

– заказ – сущность, отражающая факт обращения клиента за выполнением одной или нескольких услуг. Содержит список вложенных записей, представляющих конкретные услуги, дату создания, параметры оплаты и адреса, а также текущий статус;

– услуга – класс, описывающий основные характеристики оказываемых сервисов: название, описание, стоимость и тип. Используется как справочная сущность и ссылка в рамках заказов;

– услуга в заказе – это вспомогательная сущность, представляющая конкретную услугу в рамках одного заказа с индивидуальными параметрами: датой и временем выполнения, статусом, исполнителем, комментариями и историей изменений. Размещение данных напрямую в сущности «Услуга» нарушает принцип единой ответственности, усложняет повторное использование и может привести к логическим ошибкам – например, при изменении статуса услуга изменится во всех заказах. Выделение отдельного класса позволяет точно управлять каждой услугой, поддерживать независимое выполнение, привязку уведомлений и обеспечивает корректную и масштабируемую архитектуру системы;

– уведомление – класс, обеспечивающий информирование пользователя о событиях в системе. Содержит текст уведомления, статус прочтения, дату и ссылку на объект, к которому оно относится.

Диаграмма классов предоставляет наглядное представление архитектуры системы, способствует формализации требований и упрощает

реализацию программного обеспечения за счет четкой структуры и ясных связей между сущностями. Снижает риски ошибок на этапе проектирования, повышает качество коммуникации между участниками разработки и служит основой для построения логики клиентской и серверной частей приложения.

Диаграмма «сущность-связь» представляет собой абстрактную модель, отражающую логическую структуру предметной области и взаимосвязи между ее ключевыми компонентами. Данный тип диаграммы применяется на этапе концептуального проектирования и служит основой для формирования логической и физической модели базы данных [7].

На рисунке 7 отображены основные сущности информационной системы: клиент, заказ, услуга, сотрудник, промежуточная сущность «услуга в заказе». Взаимодействие между ними реализовано посредством связей, описывающих основные процессы и правила обработки данных.

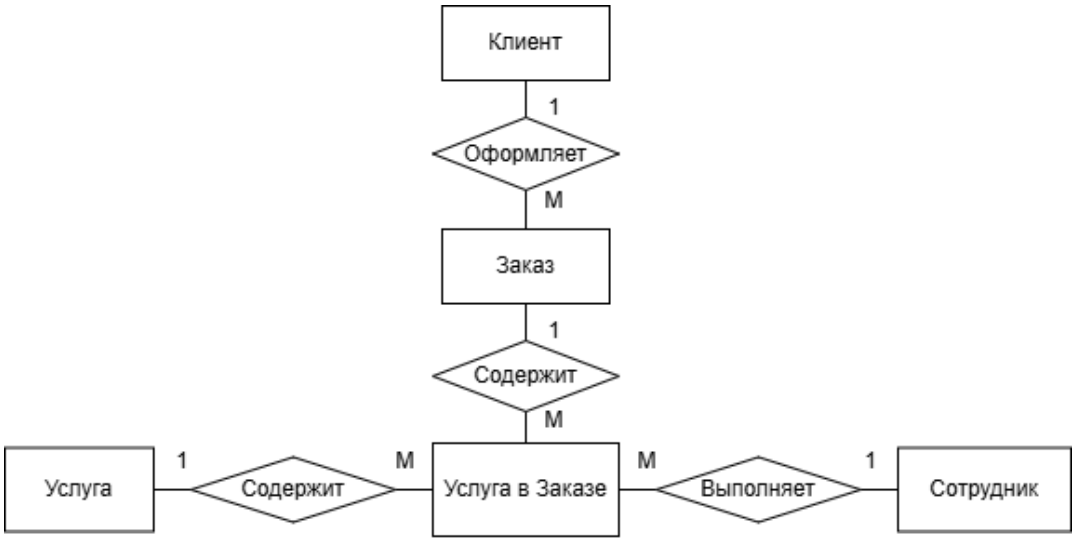


Рисунок 7 – Диаграмма «Сущность-связь»

Сущность «Клиент» связана с сущностью «Заказ» через связь «Оформляет», что отражает возможность создания одним клиентом нескольких заказов. Каждый заказ связан с множеством конкретных исполнений услуг – сущностью «Услуга в заказе», которая позволяет хранить

индивидуальные параметры исполнения: дату, время, назначенного сотрудника и статус.

Связь между «Услугой» и «Услугой в заказе» обеспечивает привязку справочной информации о конкретной услуге (название, описание, стоимость) к ее выполнению в рамках конкретного заказа. Связь «Выполняет» между «Сотрудником» и «Услугой в заказе» отражает участие сотрудников в реализации каждой услуги и допускает индивидуальное распределение заданий.

Представленная модель четко визуализирует связи «один ко многим» между сущностями и обеспечивает основу для построения масштабируемой и логически непротиворечивой архитектуры базы данных. Упрощает переход к следующему этапу проектирования, повышает структурированность данных и способствует формализации требований к информационной системе.

Логическая модель базы данных в контексте документоориентированной СУБД описывает структуру хранения информации на основе сущностей, их атрибутов и связей. В отличие от реляционных систем, поддерживает вложенные документы и более гибкую схему. Модель служит промежуточным этапом между концепцией и физическим уровнем хранения, упрощая проектирование и реализацию.

На рисунке 8 показана логическая структура информационной системы, отражающая основные сущности и их взаимодействие.

Краткое описание сущностей:

– клиент – пользователь системы, данные которого включают контакты, пароль и дату регистрации. Связан с заказами и уведомлениями.

– сотрудник – исполнитель или администратор. Назначается на выполнение услуг и управляет заказами.

– услуга – справочная сущность с названием, описанием, ценой и типом. Используется в заказах как шаблон услуги.

– заказ – фиксирует обращение клиента. Содержит параметры оформления, оплаты и список услуг в виде вложенного массива.

– услуга в заказе – вложенная структура в заказе, включающая дату, время, статус, исполнителя и комментарий. Позволяет индивидуально управлять каждой услугой.

– уведомление – отдельная коллекция сообщений пользователям. Содержит ссылку на заказ ии конкретную услугу, дату и статус прочтения.

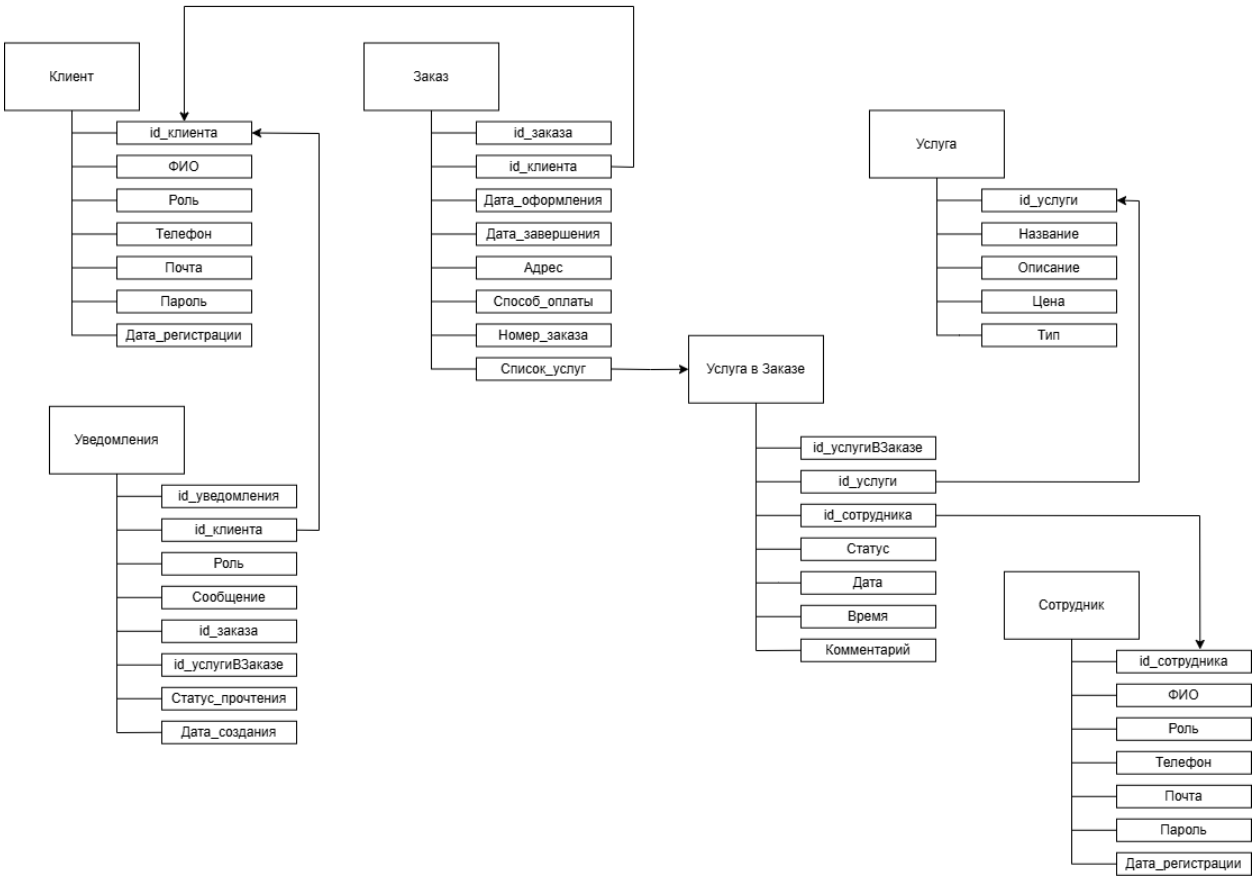


Рисунок 8 – Логическая модель базы данных

Модель отражает вложенность и связи между сущностями, обеспечивает масштабируемость и гибкость, что критически важно для документоориентированных СУБД.

Физическая модель базы данных – это конкретная реализация логической модели с указанием типов данных, структур и ограничений, необходимых для хранения и обработки информации. В отличие от логической, включает детали реализации: типы полей, индексы и схемы

связей. Такая модель повышает эффективность работы с данными, оптимизирует запросы и снижает вероятность ошибок при разработке[18].

В системе используется «MongoDB – это документо-ориентированная база данных, предназначенная для гибкой, масштабируемой и очень быстрой работы даже при больших объемах данных»[2]. Это упрощает адаптацию под изменяющиеся требования, ускоряет внедрение новых функций и позволяет работать с вложенными структурами. СУБД поддерживает горизонтальное масштабирование и хорошо интегрируется с Node.js, что делает ее особенно удобной для веб-приложений.

Сравнение MongoDB с альтернативными решениями представлено в таблице 3.

Таблица 3 – Сравнительный анализ баз данных

Критерий	MongoDB	PostgreSQL	MySQL
Тип БД	Документо-ориентированная	Реляционная	Реляционная
Гибкость структуры	Высокая	Низкая	Низкая
Масштабируемость	Отличная	Хорошая	Хорошая
Сложность запросов	Простые	Сложные, но мощные	Сложные, но популярные

Описание физической модели базы данных представлено на рисунке 9.

Физическая модель базы данных отражает реализацию логической структуры, включая конкретные типы данных и связи между сущностями в среде MongoDB. Каждая коллекция содержит набор атрибутов, определяющих правила хранения, обработки и доступа к данным.

«Клиент» включает поля id_клиента (ObjectID), ФИО, Роль, Телефон, Почта, Пароль (все – string), а также Дата_регистрации (date). Эти данные обеспечивают аутентификацию и взаимодействие клиента с системой.

«Сотрудник» содержит аналогичный набор данных: id_сотрудника (ObjectID), личные и контактные данные, а также дату регистрации. Используется для назначения исполнителей на услуги и управления задачами.

«Заказ» фиксирует обращение клиента и содержит поля: id_заказа и id_клиента (ObjectID), Дата_оформления, Дата_завершения (date), Адрес, Способ_оплаты, Номер_заказа (string), Список_услуг – массив вложенных документов , где каждая запись представляет собой объект «Услуга в заказе».

«Услуга в заказе» – вложенная структура, хранящая информацию об отдельных услугах в заказе: id_услугиВзаказе, id_услуги, id_сотрудника (ObjectID), Статус, Дата, Время, Комментарий (string/date). Это позволяет вести независимый учет выполнения каждой услуги.

«Услуга» – справочная сущность, содержащая поля id_услуги (ObjectID), Название, Описание, Тип (string), Цена (number). Представляет каталог доступных сервисов.

«Уведомления» реализуют информирование пользователей о действиях в системе. Содержат id_уведомления (ObjectID), ссылки на клиента, заказ и услугу, текст сообщения, статус прочтения (boolean) и дату создания (date).

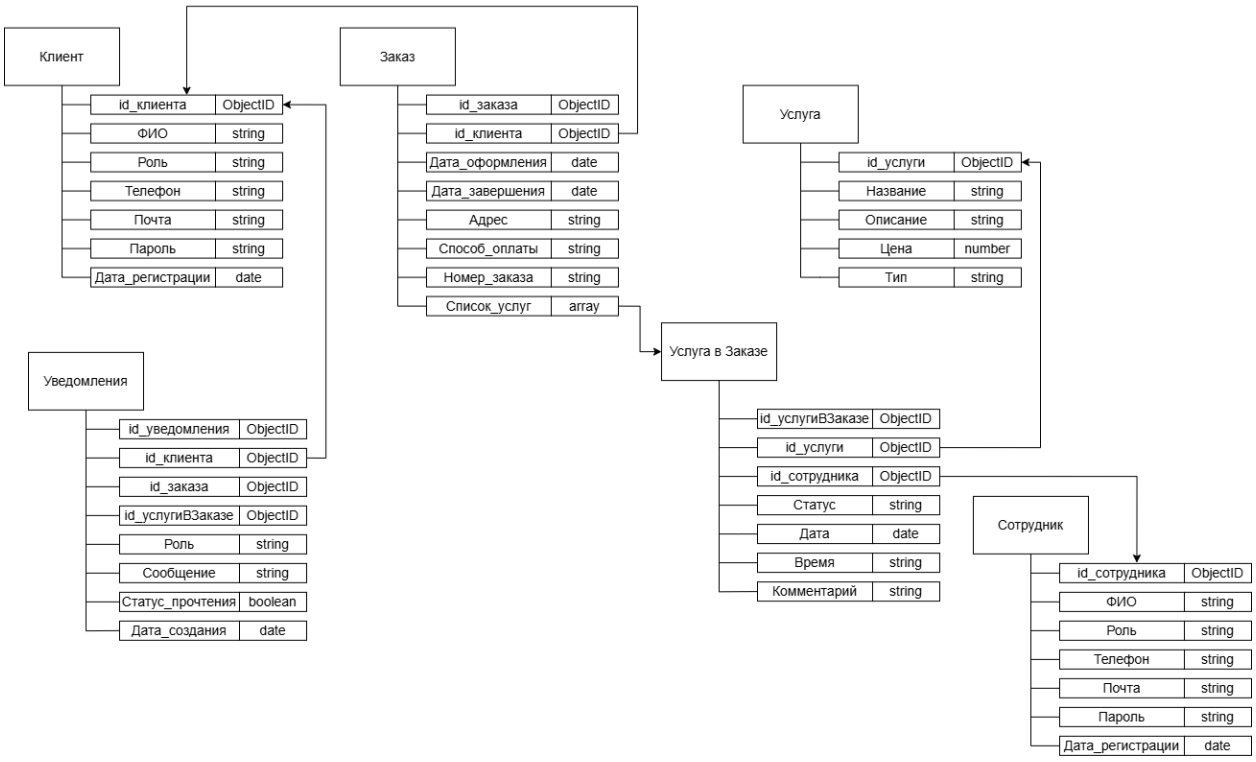


Рисунок 9 – Физическая модель базы данных

Такая структура модели обеспечивает надежность, масштабируемость и гибкость при работе с данными в СУБД, учитывая особенности документо-ориентированного подхода.

Диаграмма последовательности – это поведенческая диаграмма UML, предназначенная для отображения порядка сообщений между объектами при выполнении сценария. Помогает формализовать поведение системы, выявить синхронность операций и уточнить структуру взаимодействий.

Диаграмма на рисунке 10 отражает взаимодействие компонентов системы, улучшая точность проектирования и согласованность при реализации.

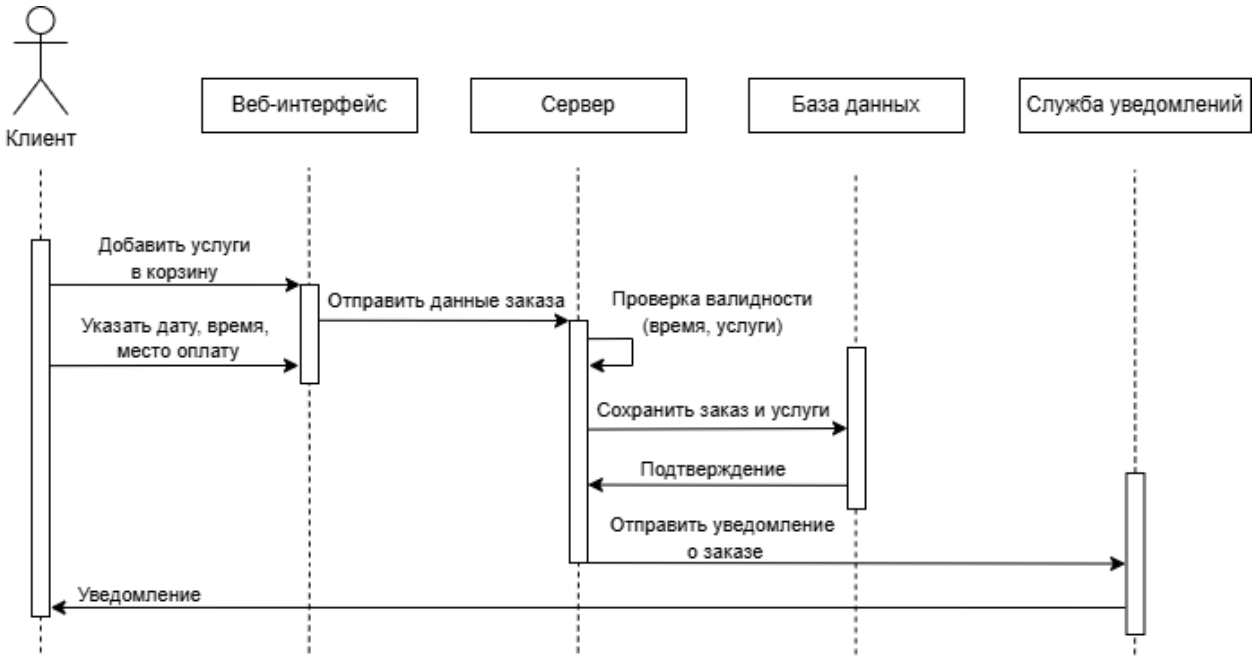


Рисунок 10 – Диаграмма последовательности оформления заказа на услугу

На диаграмме представлен сценарий оформления заказа на услугу через пользовательский интерфейс. В процессе участвуют пять ключевых объекта: «Клиент», «Веб-интерфейс», «Сервер», «База данных» и «Служба уведомлений».

– клиент выбирает услуги и добавляет их в корзину;

– через веб-интерфейс указывает параметры оформления (дата, место, способ оплаты);

– данные заказа передаются на сервер;

– сервер проверяет валидность услуг и даты;

– после успешной проверки заказ сохраняется в базу данных;

– база данных подтверждает запись;

– сервер отправляет информацию в службу уведомлений;

– уведомление доставляется клиенту;

– клиент получает подтверждение оформления.

Последовательность сообщений позволяет отследить логику прохождения данных, выявить все точки взаимодействия компонентов и установить порядок обработки операций, что критично при проектировании и тестировании системного поведения.

«Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий»[10]. Описывает, какие состояния может принимать объект (в данном случае – отдельная услуга в заказе), и какие события вызывают переходы между этими состояниями. Такая диаграмма позволяет формализовать бизнес-логику, повысить прозрачность процессов и снизить вероятность логических ошибок при реализации и сопровождении программного обеспечения.

Применение диаграммы состояний особенно актуально при наличии нескольких этапов выполнения задачи, четко определенных ролей (например, клиента и мастера) и условий, при которых допускается переход между состояниями.

На рисунке 11 представлена диаграмма состояния услуги в заказе, а именно поведение одной конкретной услуги, входящей в заказ клиента, с момента добавления в корзину и до завершения или отмены.

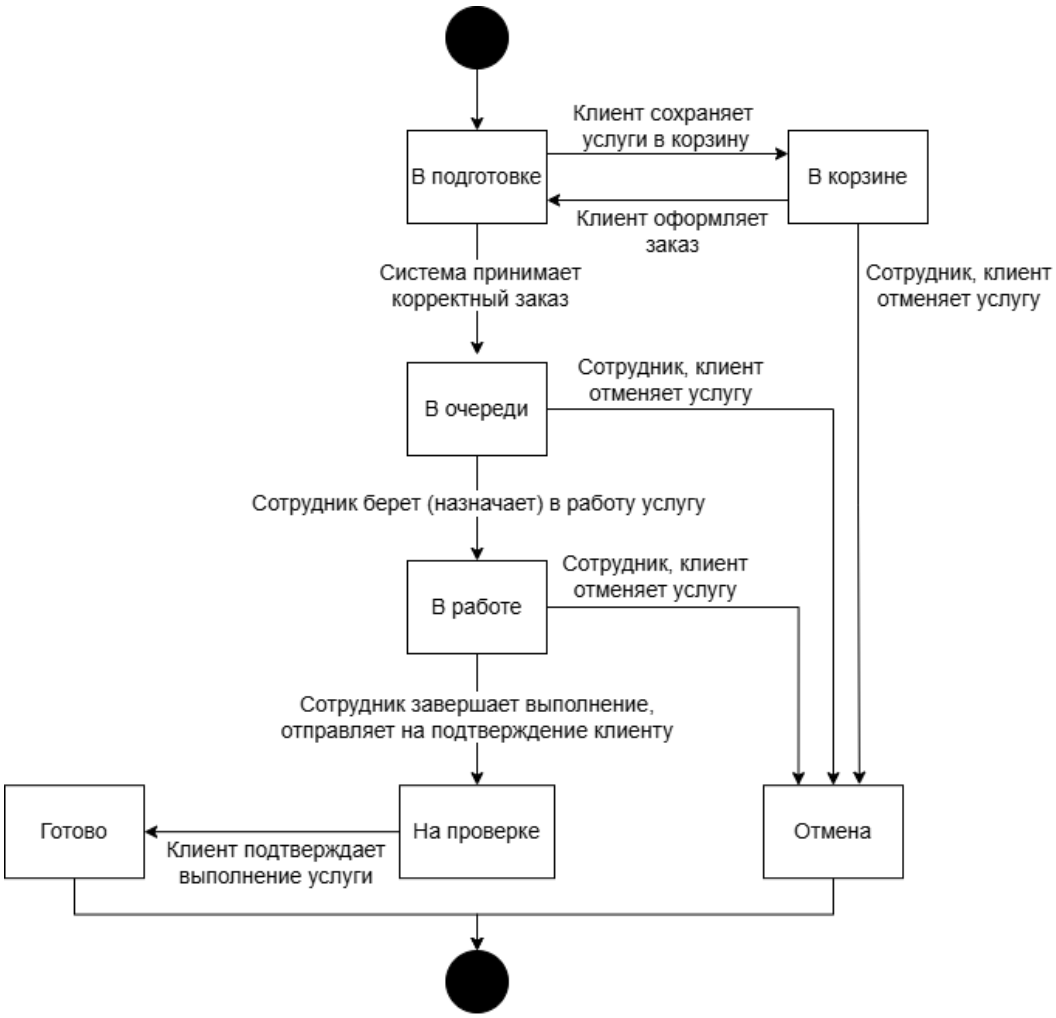


Рисунок 11 – Диаграмма состояния «Услуги в заказе»

Модель охватывает ключевые этапы, действия пользователя и сотрудников, а также отображает допустимые переходы между состояниями:

– «В корзине» – стартовое состояние, в котором услуга сохраняется клиентом для последующего оформления заказа. На этом этапе клиент может удалить ее из корзины или приступить к оформлению.

– «В подготовке» – временное состояние после выбора услуг, когда клиент начал оформление, но еще не отправил заказ в систему. Позволяет редактировать состав, адрес и параметры выполнения.

– «В очереди» – система принимает корректный заказ, и каждая услуга попадает в очередь на исполнение. Здесь ожидается, что сотрудник назначит себя или будет назначен администратором.

– «В работе» – услуга была назначена конкретному мастеру и находится в процессе выполнения. Назначение возможно как вручную (админом), так и автоматически (самоназначением сотрудника).

– «На проверке» – сотрудник завершает выполнение услуги и отправляет ее клиенту на подтверждение. Пока клиент не примет результат, услуга остается на этом этапе.

– «Готово» – финальное состояние, в которое услуга переходит после подтверждения выполнения клиентом. С этого момента она считается завершенной.

– «Отмена» – любая услуга может быть отменена на любом этапе, кроме завершенного, как со стороны клиента, так и со стороны администратора или сотрудника. Это также финальное состояние, означающее, что услуга не будет выполнена.

Такая структура диаграммы позволяет обеспечить отслеживание прогресса каждой услуги в заказе, разграничить действия по ролям (клиент, сотрудник, администратор), управлять отдельными услугами независимо в рамках одного заказа, сохранять историю переходов для логирования и отчетности и формализовать правила переходов и условий отмены.

Диаграмма состояния услуги в заказе обеспечивает предсказуемость работы системы, минимизирует ошибки в логике исполнения и облегчает тестирование реализованных процессов.

2.3 Выбор технологий и инструментов для разработки

В качестве основных технологий проекта, рассмотренных в сравнительной таблице 4, выбраны Node.js, Express.js, React и MongoDB. Node.js и Express.js применяются благодаря высокой производительности, достигаемой за счет асинхронной обработки запросов, что критически важно для масштабируемых веб-приложений. Использование единого языка – JavaScript – на клиенте и сервере упрощает разработку и снижает порог входа.

Express.js, как легкий и гибкий фреймворк, не навязывает жесткую архитектуру, в отличие от Django и Spring Boot, позволяя адаптировать серверную часть под специфику проекта. Богатая экосистема npm обеспечивает доступ к тысячам модулей, что существенно ускоряет процесс разработки [4], [5], [14], [17], [20].

Таблица 4 – Сравнительный анализ серверных технологий

Критерий	Node.js + Express.js	Django	Spring Boot
Язык программирования	JavaScript	Python	Java
Производительность	Высокая для асинхронных операций	Подходит для приложений средней сложности	Высокая, но требует больше ресурсов
Гибкость	Высокая, минималистичный фреймворк	Хорошая, но включает много встроенных решений	Высокая, но требует конфигурации
Экосистема	Богатая (npm)	Меньше, чем у Node.js	Обширная (Maven, Gradle)

Интерфейс пользователя реализуется с использованием React, что обосновано данными сравнительного анализа таблицы 5. Как и Angular или Vue.js, React поддерживает компонентный подход, обеспечивающий модульность, переиспользуемость и масштабируемость интерфейса. За счет Virtual DOM достигается высокая производительность при обновлении данных. В отличие от Angular, React имеет более простую архитектуру и невысокий порог входа, что ускоряет обучение и внедрение. Несмотря на легкость освоения, Vue.js уступает React по популярности и масштабности экосистемы, что может ограничить доступ к документации и сторонним библиотекам. React является одним из наиболее распространенных инструментов для фронтенд-разработки, что делает его выбор оправданным с точки зрения технологической зрелости, поддержки сообщества и наличия готовых решений [6].

Таблица 5 – Сравнительный анализ технологий пользовательских интерфейсов

Критерий	React	Angular	Vue.js
Компонентный подход	Да	Да	Да
Производительность	Высокая	Высокая, но тяжелее для начинаюших	Высокая
Кривая обучения	Средняя	Высокая	Низкая
Популярность	Очень высокая	Высокая	Средняя

В проекте реализован комплекс мер по обеспечению безопасности. Пароли пользователей хранятся в хэшированном виде с применением алгоритма bcrypt, что исключает возможность их восстановления даже при компрометации базы данных. Аутентификация осуществляется с помощью JSON Web Tokens (JWT), обеспечивающих надежную идентификацию и разграничение доступа. Кроме того, реализованы меры по резервному копированию.

2.4 Архитектура и взаимодействие компонентов

Система построена на клиент-серверной архитектуре, обеспечивающей логическое разделение обязанностей между компонентами. Сервер реализует бизнес-логику, обрабатывает HTTP-запросы и управляет данными, хранящимися в базе MongoDB. Для связи между объектами приложения и коллекциями используется библиотека Mongoose (ODM), упрощающая работу с документо-ориентированными структурами. Клиентская часть реализована как веб-приложение на React и взаимодействует с сервером через REST API, передавая и получая данные в формате JSON.

Данный подход обеспечивает масштабируемость, надежность и гибкость системы. Централизованное хранение данных снижает вероятность ошибок и обеспечивает актуальность информации. Обновление клиентского интерфейса или серверной логики может выполняться независимо, что

упрощает сопровождение. Архитектура легко адаптируется под рост нагрузки за счет масштабирования серверных ресурсов и распределения базы данных.

На рисунке 12 визуализированы основные модули программной системы, их структура и взаимосвязи. Диаграмма отражает взаимодействие между фронтендом, бэкендом и базой данных, позволяя выделить ключевые функциональные блоки и проследить потоки данных в системе.

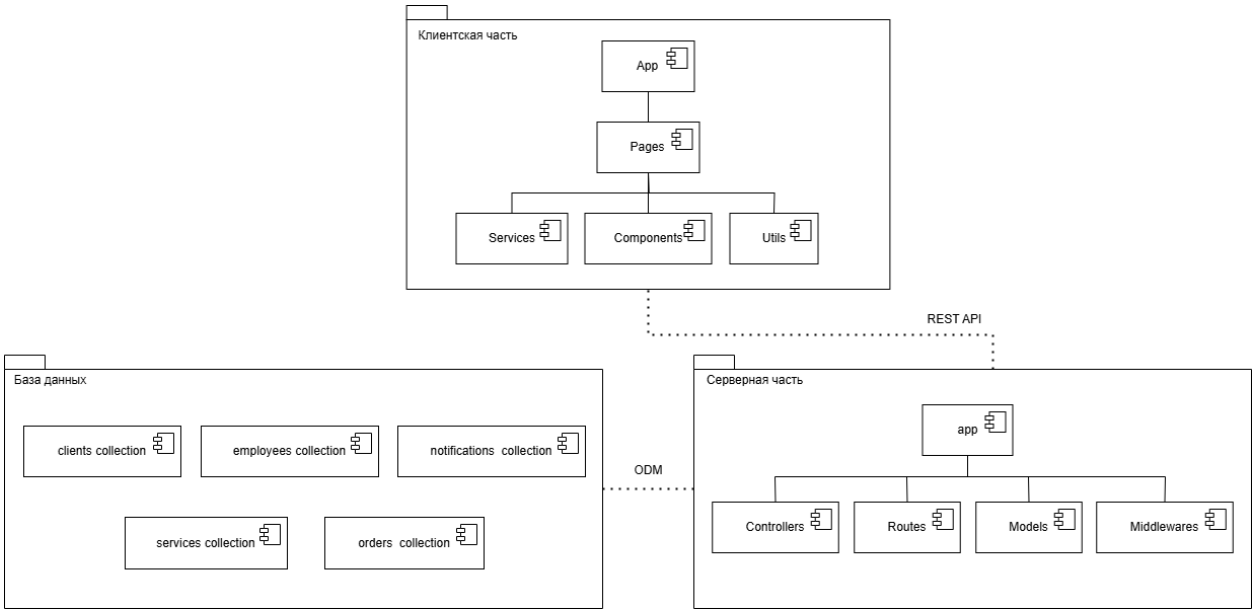


Рисунок 12 – Диаграмма компонентов

На представленной диаграмме отображена архитектура, разделенная на: Frontend (клиентская часть) и Backend (серверная часть), а также их связь с базой данных.

Клиентская часть реализована с использованием React и включает следующие основные модули:

а) Components – переиспользуемые элементы интерфейса:

1) «Navbar» – навигационная панель;

2) «KanbanBoard» –визуализация заказов по статусам;

3) «OrderForm», «OrdersTable», «ServiceCard», «Filters» – оформление заказов и фильтрация услуг;

4) «Notification» – отображение уведомлений;

5) «reports» – генерация отчетов (финансового, статистического, по услугам и сотрудникам).

б) «Pages» – страницы для различных ролей: клиента, сотрудника, администратора («ClientOrdersPage», «AdminProfilePage», «EmployeeApplicationsPage», «ReportPage», «RequestsPage», «AddServicePage», «NotificationsPage» и др);

в) «Services» – взаимодействие с API. Например, «authService.js» реализует авторизацию и регистрацию;

г) «utils» – вспомогательные утилиты, включая «localStorageUtils.js» и другие функции, обеспечивающие работу клиентского интерфейса.

Серверная часть реализована с использованием Node.js и Express.js и включает следующие основные модули:

а) «Controllers» – обработка входящих HTTP-запросов и реализацию бизнес-логики. Контроллеры «authController», «profileController» и «serviceController» отвечают за авторизацию, управление профилями пользователей и обработку инфоформации об услугах;

б) «Middlewares» – промежуточные обработчики, обеспечивающие безопасность системы. Включают «authMiddleware» (проверка JWT-токенов), «checkRole» и «checkAdmin» (контроль доступа по ролям);

в) «Models» – схемы данных для работы с MongoDB. Модели «Client», «Employee», «Service», «Order» и «Notification» описывают структуру и связи ключевых сущностей системы, обеспечивая целостность и корректность данных;

MongoDB служит централизованным хранилищем данных, где в виде документов сохраняется информация о пользователях, заказах, услугах и уведомлениях. Документо-ориентированная структура обеспечивает гибкость, масштабируемость и удобную работу с вложенными структурами, такими как массив «services» внутри заказов.

Взаимодействие компонентов:

а) клиентская часть отправляет запросы по защищенным маршрутам API;

б) сервер валидирует и авторизует запрос, обращается к соответствующей модели данных;

в) полученные результаты возвращаются на клиент в формате JSON;

г) при необходимости создаются уведомления и фиксируются изменения в баз.

Диаграмма компонентов демонстрирует четкое разграничение логики между слоями, что способствует надежности, масштабируемости и упрощает сопровождение и развитие системы.

Диаграмма развертывания отражает, как программные компоненты системы распределены по физическим и логическим узлам – от пользовательского браузера до серверной инфраструктуры. На рисунке 13 представлена структура развертывания веб-приложения, реализованного на стеке MERN.

Пользователь взаимодействует с системой через браузер, в котором выполняется клиентское React-приложение. Приложение отправляет HTTPS-запросы на веб-сервер, обеспечивая защищенную передачу данных.

На стороне сервера работает Node.js-приложение, в котором используется фреймворк Express.js для обработки маршрутов, логики запросов и взаимодействия с базой данных. Обмен между сервером приложений и базой данных осуществляется через библиотеку Mongoose – официальную ODM для MongoDB, позволяющую работать с коллекциями и документами на основе схем.

Сервер базы данных хранит структурированную информацию о пользователях, заказах, услугах и уведомлениях. Благодаря документо-ориентированной модели и поддержке вложенных структур, СУБД обеспечивает гибкость и масштабируемость.

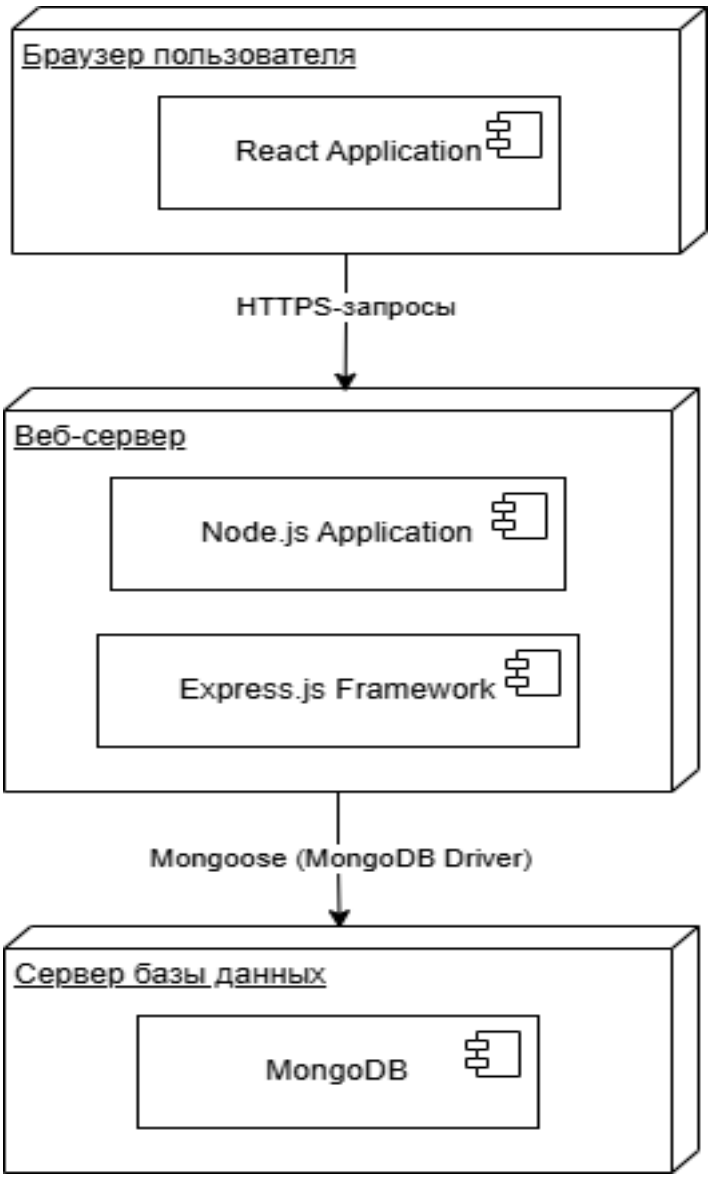


Рисунок 13 – Диаграмма развертывания

Диаграмма развертывания наглядно демонстрирует маршруты взаимодействия между слоями системы, распределение логики, используемые технологии и протоколы обмена. Это способствует анализу архитектуры с точки зрения масштабируемости, надежности и безопасности. В случае роста нагрузки система легко масштабируется как на уровне сервера приложений, так и на уровне хранения данных.

2.5 Интерфейс пользователя

Диаграмма на рисунке 14 отражает иерархию страниц веб-приложения, сгруппированных по ролям пользователей: неавторизованный пользователь, клиент, мастер и администратор. Такая структура позволяет наглядно представить состав интерфейса, логику доступа и навигацию, доступную каждой категории пользователей.

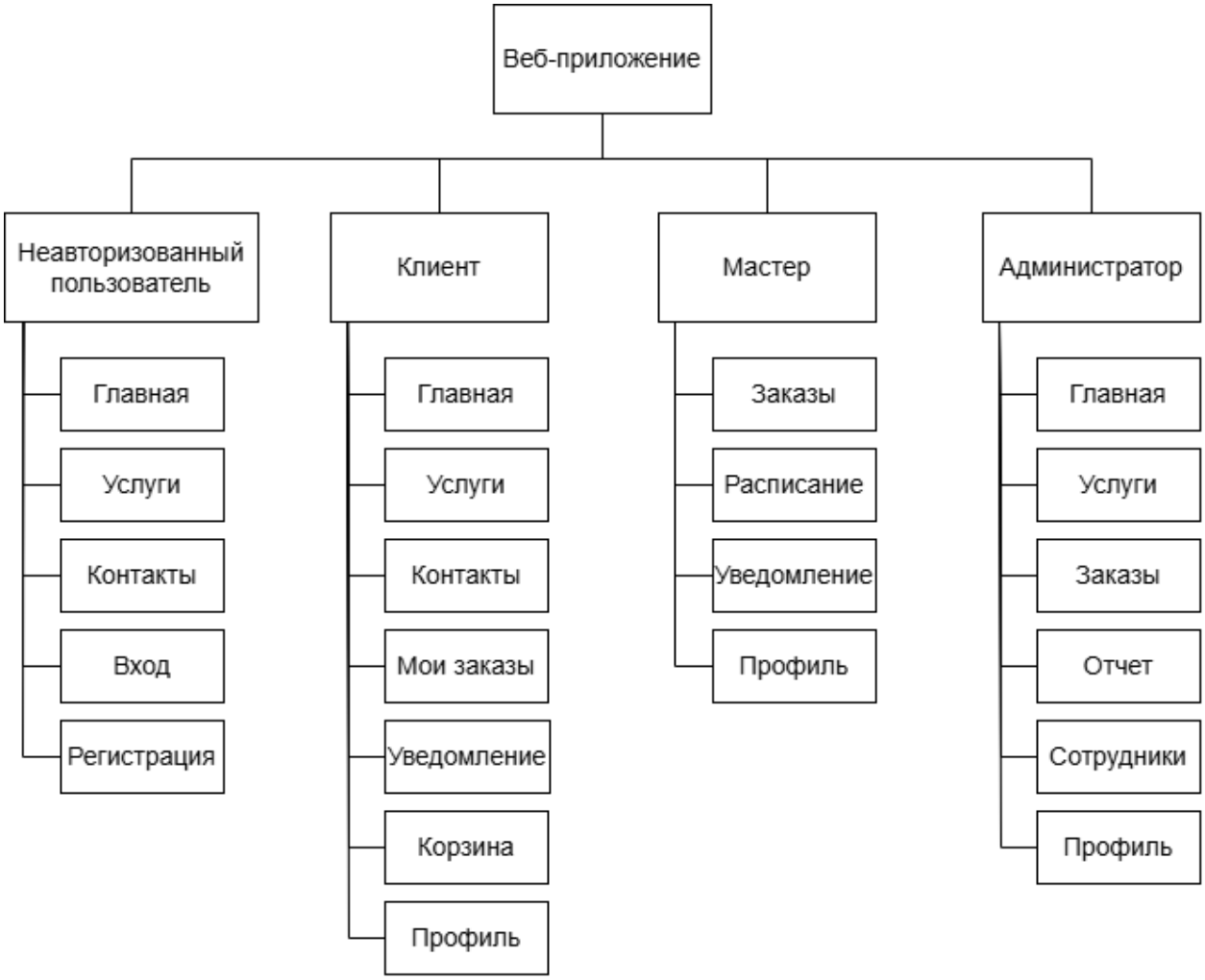


Рисунок 14 – Диаграмма структуры интерфейса

Для неавторизованного пользователя доступны базовые разделы: главная страница с поиском, услуги (с ограниченным доступом), контакты, а

также формы входа и регистрации. При попытке оформления заявки система перенаправляет пользователя на страницу авторизации.

Клиент, после входа в систему, получает расширенный функционал: оформление и просмотр заказов, управление профилем, уведомления, доступ к корзине и полноценная работа с услугами через фильтры и формы записи.

Мастер работает с заказами, имеет доступ к календарю расписания, уведомлениям и может редактировать личный профиль. Интерфейс упрощен и адаптирован под задачи исполнителя.

Администратор управляет всеми заказами через канбан-доску и таблицу, редактирует услуги, добавляет сотрудников, формирует отчеты и администрирует данные пользователей через панель управления.

Диаграмма обеспечивает визуальное понимание логики интерфейса и разграничения доступа, что упрощает проектирование, разработку и тестирование.

Для оценки визуального оформления была подготовлена демонстрационная версия главной страницы интерфейса. Макет, представленный на рисунке 15, иллюстрирует ключевые элементы взаимодействия и особенности визуального восприятия.

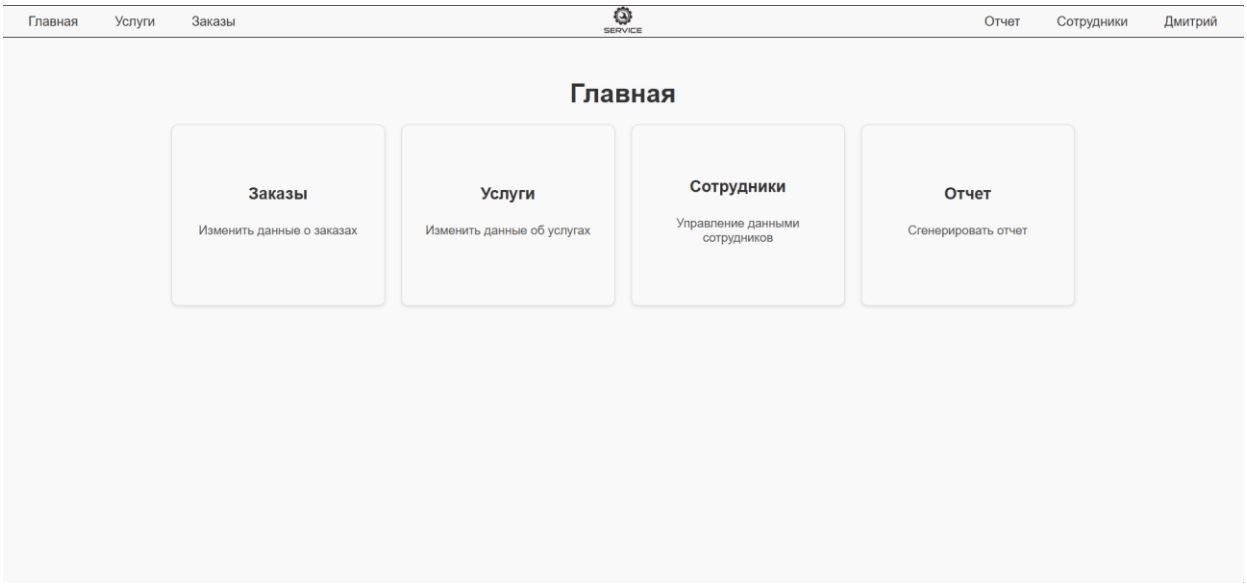


Рисунок 15 – Главная страница администратора

На рисунке показан макет главной страницы администратора с карточками для перехода к ключевым разделам: «Заказы», «Услуги», «Сотрудники» и «Отчет». Интерфейс выполнен лаконично, элементы размещены интуитивно, что упрощает навигацию и ускоряет работу.

Верхнее меню обеспечивает быстрый доступ ко всем функциям – от управления данными до перехода в личный профиль. Оформление поддерживает единый стиль всего приложения и ориентировано на удобство и функциональность.

Выводы по второй главе

В ходе проектирования программного обеспечения была разработана архитектура системы на основе клиент-серверной модели с использованием стека MERN. Сформированы диаграммы классов, компонентов, состояний, последовательностей и развертывания, что обеспечило формализацию логики и структуры приложения. Произведено моделирование базы данных с учетом документо-ориентированной структуры, обеспечивающей гибкость и масштабируемость хранения. Обоснован выбор технологий и реализованы меры безопасности, соответствующие современным требованиям. Определена структура пользовательского интерфейса с учетом ролей и функциональных особенностей системы.

Глава 3 Разработка и тестирование программного обеспечения

3.1 Пример реализации веб-приложения учета услуг и заказов в сервисной компании

Приложение реализует полный цикл взаимодействия с заказами и услугами: от выбора до отчетности.

Основные функции включают:

а) каталог услуг с фильтрацией и поиском:

Страница отображает список доступных услуг. Реализована фильтрация по категории и цене, а также поиск по ключевым словам на главной странице. При выборе конкретной услуги клиент может ознакомиться с ее описанием. На рисунке 16 представлено визуальное отображение интерфейса каталога.

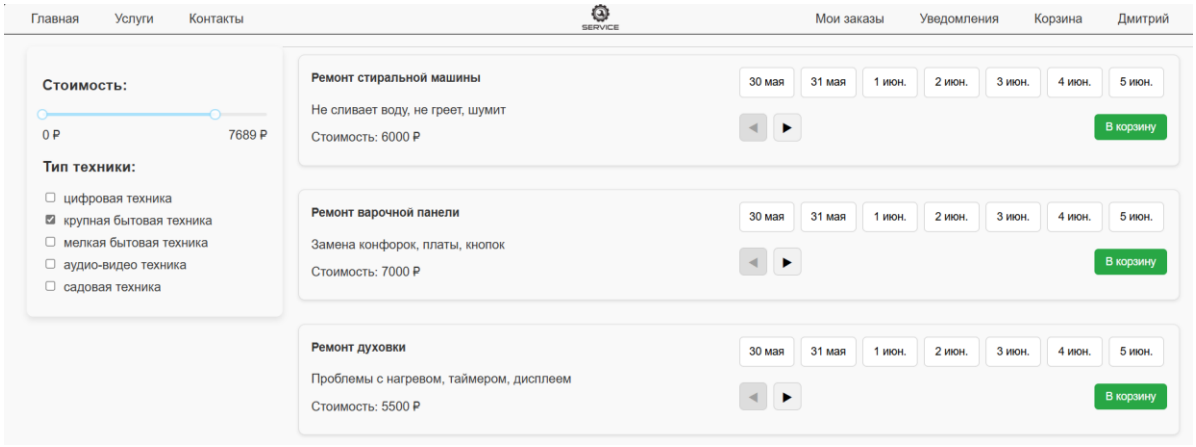


Рисунок 16 – Интерфейс каталога услуг с фильтрацией

б) оформление заказа с выбором даты и времени:

Для оформления заказа клиент выбирает услугу, указывает дату и время выполнения, после чего добавляет ее в корзину. На странице корзины дополнительно указывается место оказания услуги: в офисе или на выезде. Также предусмотрен выбор способа оплаты. На текущем этапе доступен только вариант наличного расчета, поскольку основной акцент веб-

приложения сделан на учете услуг и заказов, а не на реализации платежных механизмов. На рисунках 17 и 18 представлены интерфейс выбора даты, времени и оформления заказа.

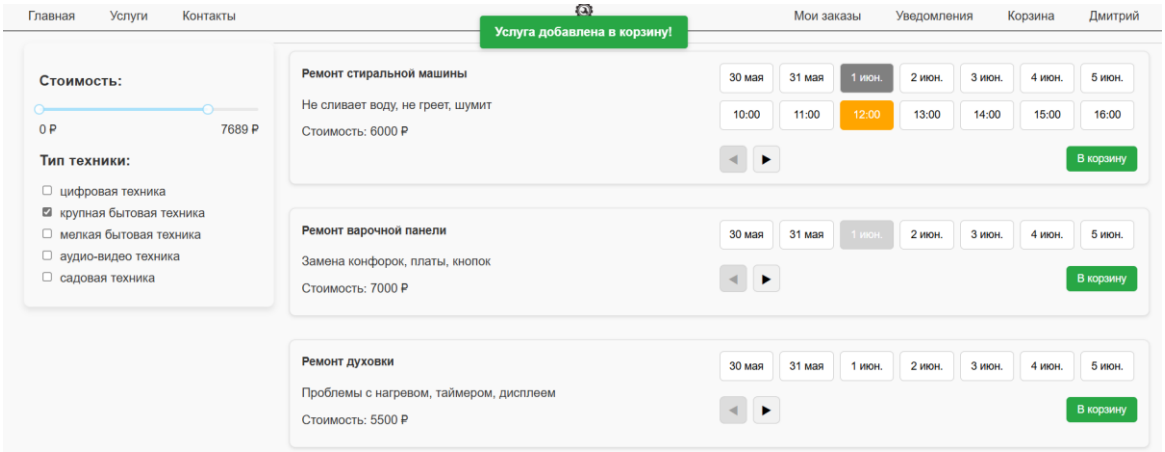


Рисунок 17 – Выбор времени услуги

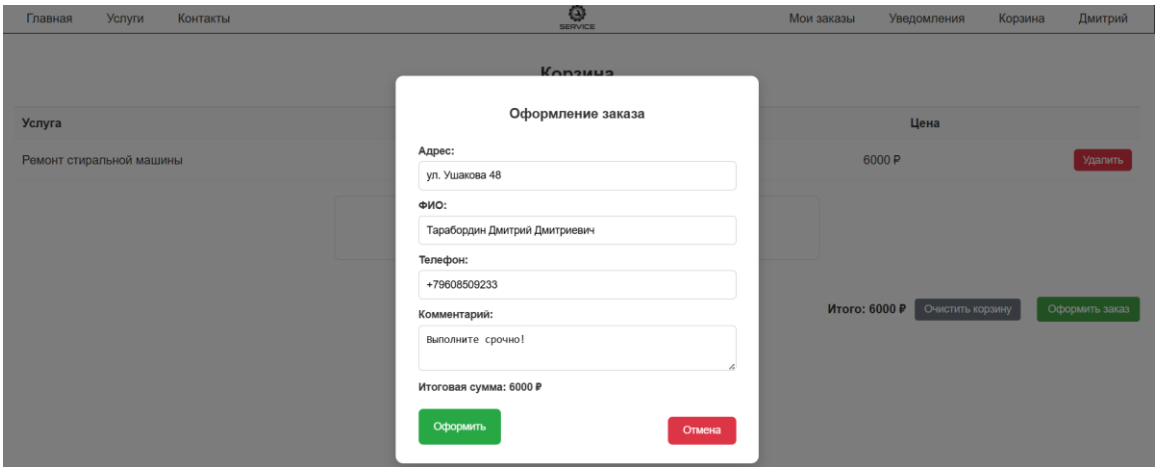


Рисунок 18 – Оформление заказа с указанием места и способа оплаты

в) управление заказами:

Актуальные и завершенные заказы, изображенные на рисунке 19, клиент может просматривать в разделе «Мои заказы». Отображается информация о дате, месте оказания услуги, комментариях и статусах. Клиент при

необходимости может отменить заказ или подтвердить его выполнение нажатием на кнопку «Завершить».

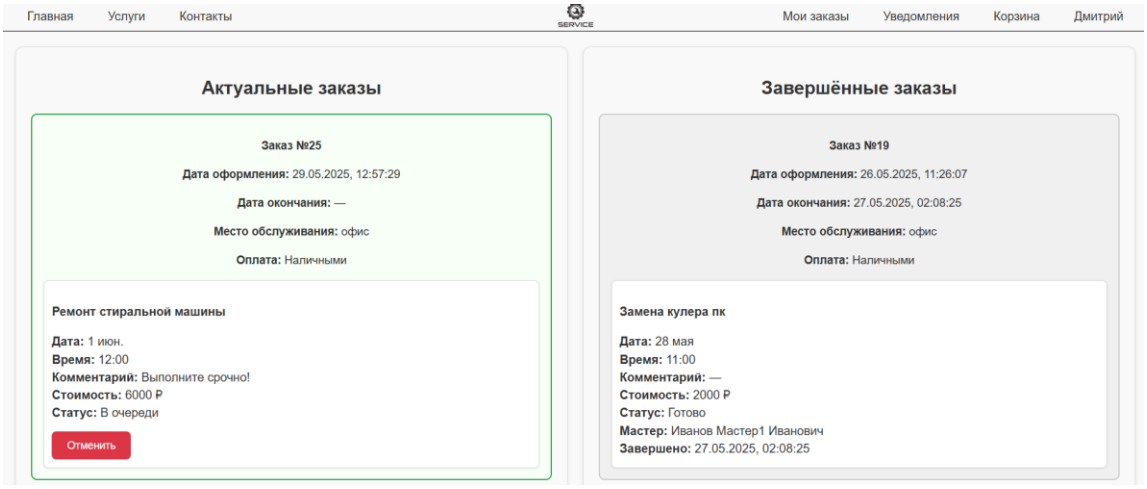


Рисунок 19 – Раздел «Мои заявки» у клиента

Мастер управляет заказами через два интерфейса, продемонстрированные на рисунках 20 и 21: канбан-доску и расширенный просмотр заказов. Канбан-доска визуализирует текущие заказы по статусам: «В очереди», «В работе», «На проверке» и «Готово». Мастер может взять заказ в работу, отправить на проверку или отменить выполнение.

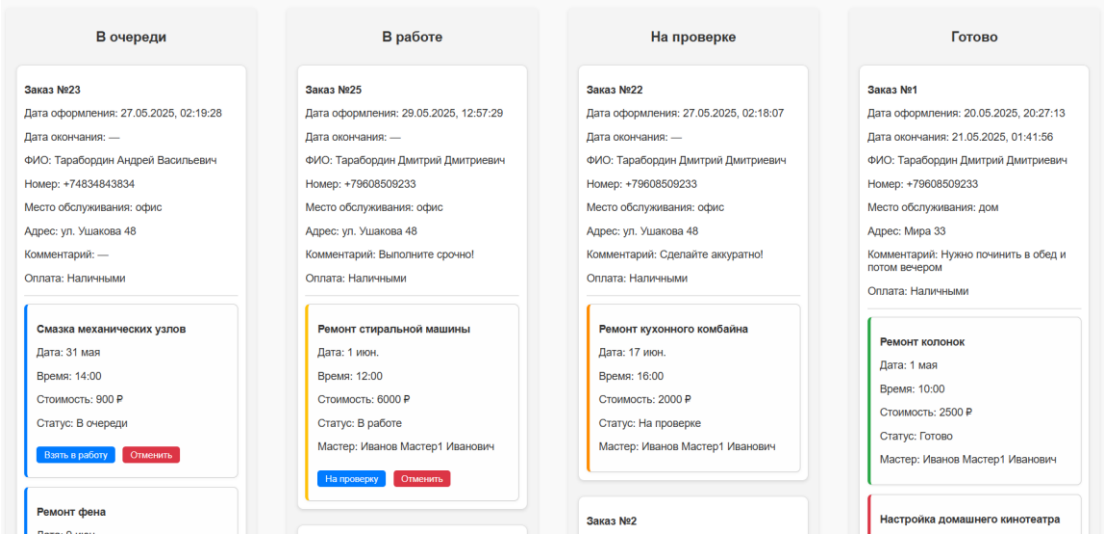


Рисунок 20 – Канбан-доска у мастера

При увеличении количества заказов мастер может использовать таблицу расширенного просмотра. Таблиц позволяет искать и фильтровать заказы по клиенту, услуге, дате, статусу и типу. Также доступны групповые действия с несколькими услугами.

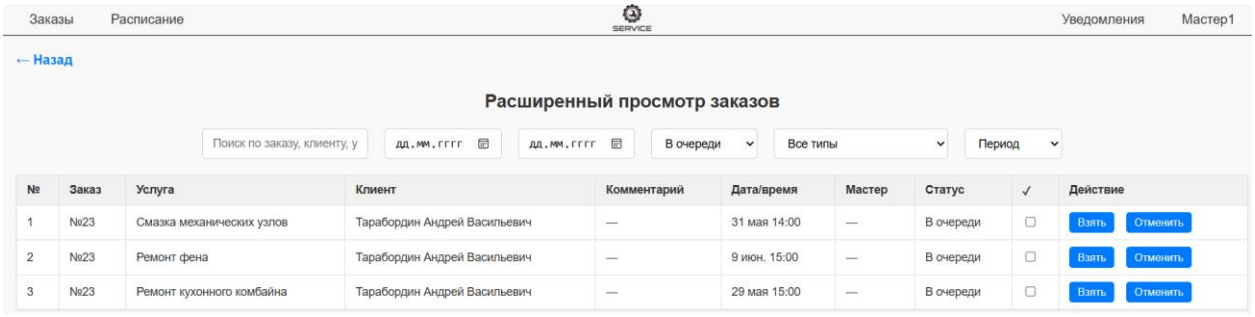


Рисунок 21 –Расширенный просмотр заказов у мастера

Администратор имеет доступ ко всем заказам и использует те же интерфейсы: канбан-доску и расширенную таблицу. Администратор может назначать мастеров на услуги, просматривать полную информацию по заказам и управлять всем процессом обслуживания.

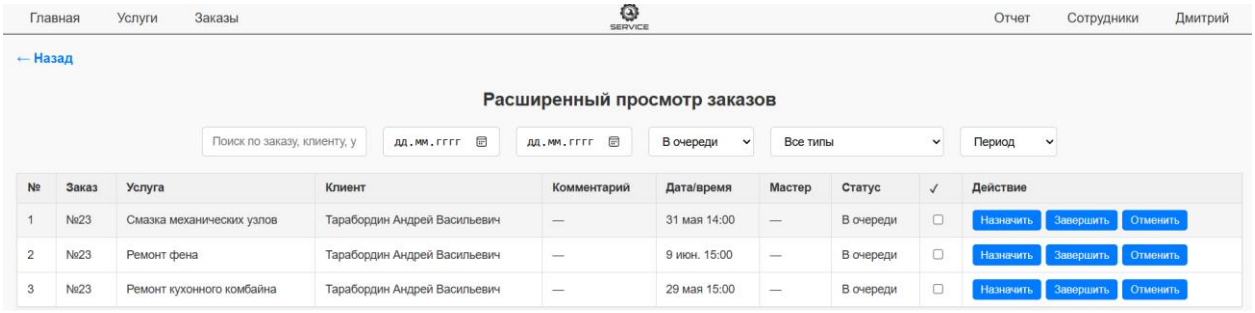


Рисунок 22 –Расширенный просмотр заказов у администратора

г) генерацию отчетов по заказам, услугам и по исполнителям:

Раздел «Отчет» предоставляет администратору доступ к четырем видам отчетов: финансовому, статическому, по услугам и исполнителям.

Каждый из отчетов имеет свою направленность, например, финансовый отчет, изображенный на рисунке 23, отображает сведения по каждому заказу: номер, ФИО клиента, дату создания и завершения заказа, количество услуг, общую сумму и текущий статус. Также автоматически рассчитывается итоговая сумма всех заказов.

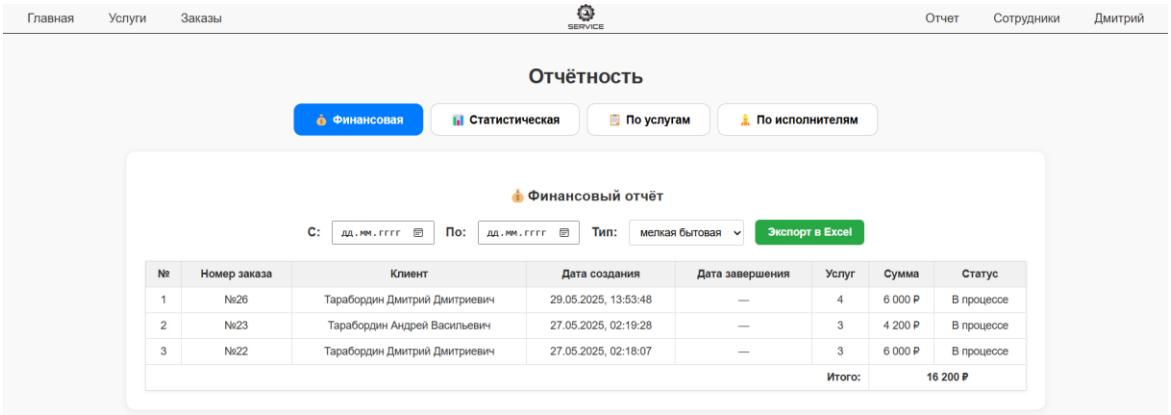


Рисунок 23 – Финансовый отчет

Статистический отчет формирует сводку по заказам:

– всего заказов;

– количество завершенных;

– в работе;

– на проверке;

– отмененных;

– всего услуг;

– среднее количество услуг на заказ.

Также отображается топ-5 популярных услуг и два визуальных отображения данных: круговая диаграмма по статусам и гистограмма по количеству заказов на каждую услугу.

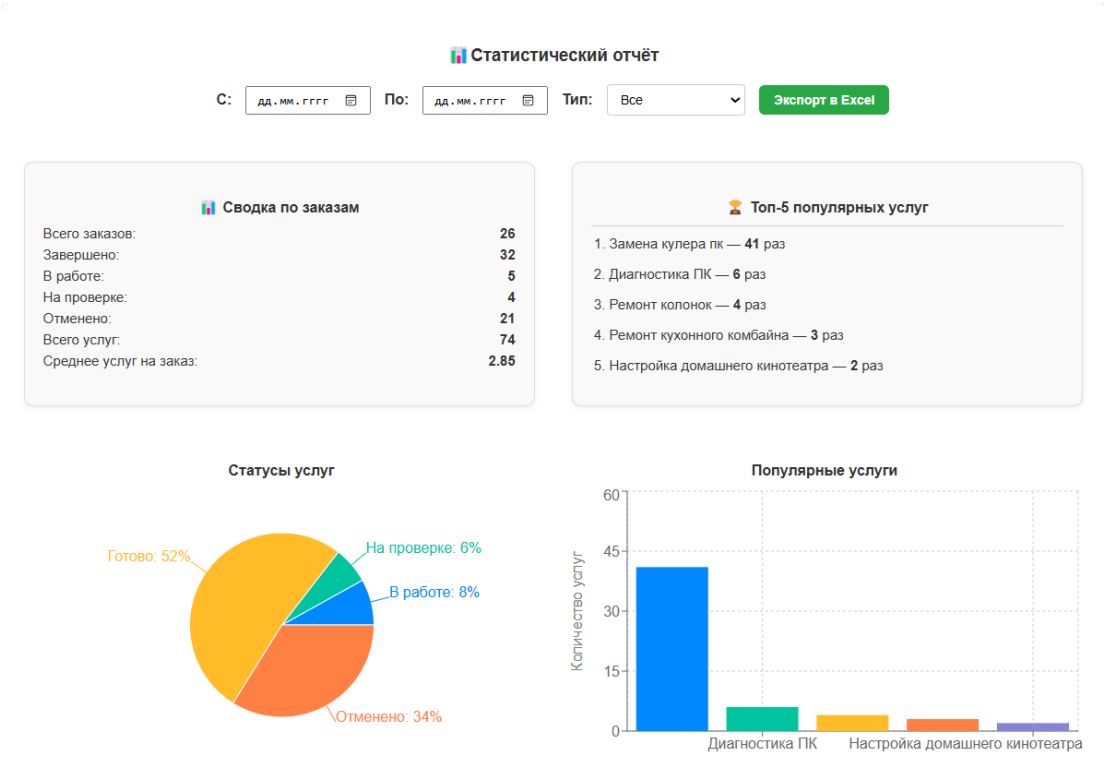
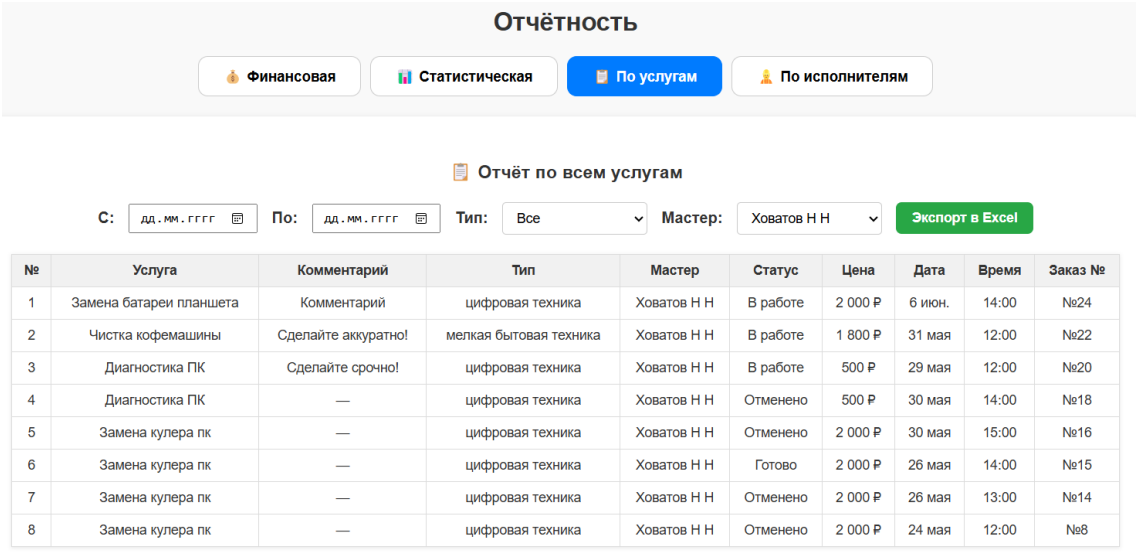


Рисунок 24 –Статистический отчет

Отчет по услугам выводит список всех оказанных услуг, с указанием их типа, комментариев, статуса, стоимости, времени оказания и исполнителя. Предусмотрена фильтрация по типу, дате и мастеру.



№	Услуга	Комментарий	Тип	Мастер	Статус	Цена	Дата	Время	Заказ №
1	Замена батареи планшета	Комментарий	цифровая техника	Ховатов Н Н	В работе	2 000 ₽	6 июн.	14:00	№24
2	Чистка кофемашины	Сделайте аккуратно!	мелкая бытовая техника	Ховатов Н Н	В работе	1 800 ₽	31 мая	12:00	№22
3	Диагностика ПК	Сделайте срочно!	цифровая техника	Ховатов Н Н	В работе	500 ₽	29 мая	12:00	№20
4	Диагностика ПК	—	цифровая техника	Ховатов Н Н	Отменено	500 ₽	30 мая	14:00	№18
5	Замена кулера пк	—	цифровая техника	Ховатов Н Н	Отменено	2 000 ₽	30 мая	15:00	№16
6	Замена кулера пк	—	цифровая техника	Ховатов Н Н	Готово	2 000 ₽	26 мая	14:00	№15
7	Замена кулера пк	—	цифровая техника	Ховатов Н Н	Отменено	2 000 ₽	26 мая	13:00	№14
8	Замена кулера пк	—	цифровая техника	Ховатов Н Н	Отменено	2 000 ₽	24 мая	12:00	№8

Рисунок 25 – Отчет по услугам

Отчет по исполнителям позволяет просмотреть, какие конкретно услуги были выполнены каждым мастером, а также получить сводку: общее количество оказанных услуг и суммарную стоимость по каждому сотруднику.

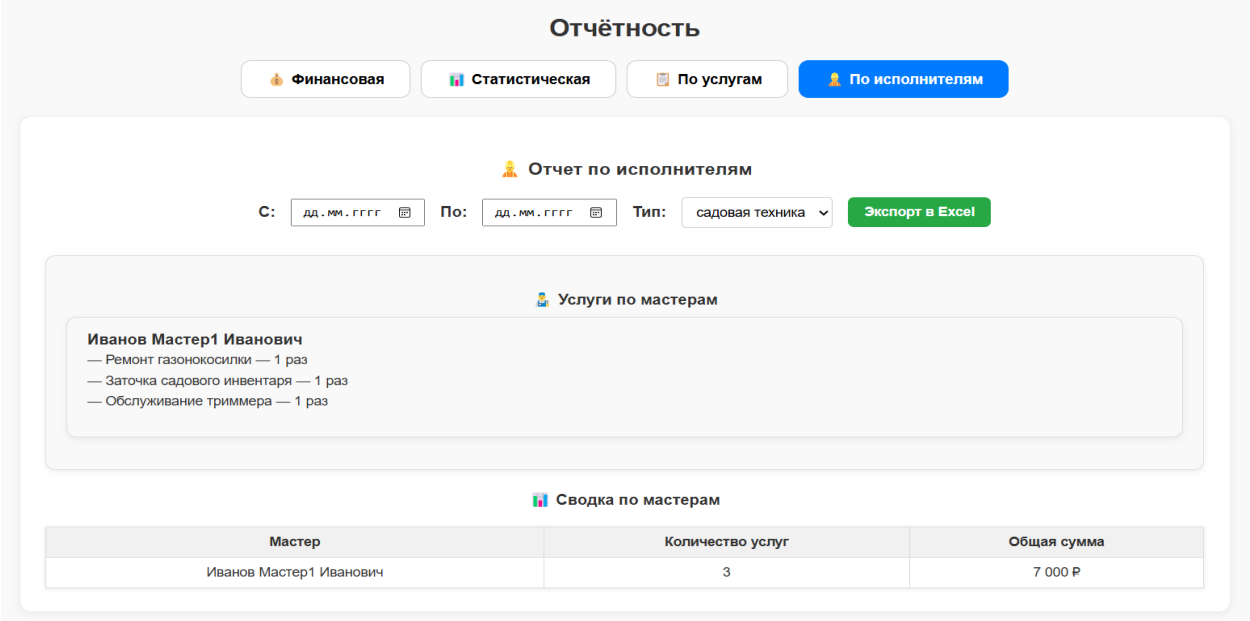


Рисунок 26 – Отчет по исполнителям

Вся отчетность может быть экспортирована в формате Excel. Это обеспечивает удобство анализа и формирование документов для последующей обработки.

Подробное тестирование изложено в следующем разделе.

3.2 Организация процесса тестирования

«Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта»[8], [3].

Цели тестирования:

– проверка соответствия функциональным и нефункциональным требованиям;

– оценка надежности и производительности системы;

– выявление и устранение ошибок до внедрения;

– проверка устойчивости при высокой нагрузке.

Для систематизации подхода к контролю качества был разработан тест-план, включающий описание функциональных и нефункциональных требований, целей тестирования и методов. В таблице 6 представлен обобщенный план проведения тестов:

Таблица 6 – Тест-план

Функциональность	Цель тестирования	Метод тестирования
Регистрация	Создание учетной записи с уникальными данными и согласием	Функциональное тестирование
Аутентификация	Проверка входа по логину и паролю	Функциональное тестирование
Авторизация	Проверка JWT и ролевого доступа	Функциональное тестирование
Редактирование профиля	Изменение данных и выход из аккаунта	Функциональное тестирование
Поиск и фильтрация услуг	Поиск и фильтрация по ключевым словам, по категории и цене	Функциональное тестирование
Просмотр услуги	Просмотр описания, цены и свободных слотов	Функциональное тестирование
Управление услугами	Добавление, редактирование и удаление услуг администратором	Функциональное тестирование
Управление сотрудниками	Добавление, редактирование и удаление сотрудников администратором	Функциональное тестирование
Корзина	Добавление и удаление услуг, выбор даты и времени	Функциональное тестирование
Оформление заказа	Указание места, оплаты, комментария и подтверждение	Функциональное тестирование
Список заказов	Разделение заказов у клиента на актуальные и завершенные	Функциональное тестирование
Статусы услуг	Отображение и изменение статуса услуг	Функциональное тестирование
Отмена услуг	Клиент или сотрудник могут отменить до выполнения	Функциональное тестирование
Назначение мастера	Ручное или самоназначение	Функциональное тестирование

Продолжение таблицы 6

Функциональность	Цель тестирования	Метод тестирования
Канбан-доска	Визуализация заказов по статусам	Функциональное тестирование
Расширенный просмотр	Фильтрация, поиск и групповые действия	Функциональное тестирование
Финансовый отчет	Сумма по заказам, детализация, экспорт	Функциональное тестирование
Статистический отчет	Диаграммы, топ-услуги, экспорт	Функциональное тестирование
Отчет по услугам	Информация по каждой услуге, фильтрация, экспорт	Функциональное тестирование
Отчет по исполнителям	Услуги и суммы по мастерам, экспорт	Функциональное тестирование
Уведомления	Пуш-уведомления, в разделе «Уведомления» отмечание как прочитанные, автоудаление	Функциональное тестирование
Неавторизированный пользователь	Поиск и фильтрация, редирект при заказе	Функциональное тестирование
Производительность	Время отклика до 2 секунд, работа при 100 одновременных пользователях	Нагрузочное тестирование
Безопасность	Резервное копирование	Функциональное тестирование
Шифрование данных	Хэширование паролей	Функциональное тестирование
Удобство	Валидация форм, логичная навигация	Функциональное тестирование
Расширяемость	Модульность и обновление без остановки	Функциональное тестирование

Объекты тестирования:

– регистрация, авторизация и разграничение доступа по ролям;

– работа с корзиной и оформление заказов;

– фильтрация и просмотр услуг;

– управление услугами и сотрудниками (администратор);

– визуализация и контроль заказов через канбан-доску и расширенный просмотр;

– генерация отчетов (финансовый, статистический, по услугам и исполнителям);

– система уведомлений;

– защита данных и безопасность;

– производительность и масштабируемость.

Для проведения тестирования используются следующие инструменты:

– ручное тестирование интерфейса – для проверки пользовательских сценариев, отображения компонентов и реакции на действия;

– Apache Benchmark (ab) – точечная оценка отклика при множестве запросов [15];

– K6 – проведение нагрузочного тестирования с возможностью имитации длительной, стрессовой и нарастающей нагрузки.

Процесс тестирования реализуется в два этапа. Сначала проводится функциональное тестирование для проверки соответствия всех реализованных функций установленным требованиям. Далее выполняется нагрузочное тестирование, направленное на оценку производительности системы при одновременном обращении большого числа пользователей.

Критериями успешного тестирования являются:

– корректная работа всех функций;

– соответствие показателей производительности заданным ограничениям;

– устойчивость системы при высокой нагрузке.

Таким образом, процесс тестирования в рамках проекта реализуется поэтапно и охватывает ключевые аспекты качества программного обеспечения. Комплексный подход позволяет обеспечить стабильную и надежную работу веб-приложения, повысить устойчивость в условиях реальной эксплуатации и заложить основу для масштабируемости и дальнейшего сопровождения системы.

3.3 Функциональное тестирование

«Функциональное тестирование – вид тестирования, направленный на проверку корректности работы функциональности приложения»[8]. Цель – убедиться, что система корректно выполняет функции при различных

сценариях использования и выявить возможные отклонения до ввода в эксплуатацию.

В рамках веб-приложения для учета услуг и заказов тестирование охватывало все основные пользовательские сценарии: регистрацию и авторизацию, редактирование профиля, оформление и управление заказами, фильтрацию и просмотр услуг, генерацию отчетов и разграничение доступа по ролям.

Тестирование проводилось вручную через пользовательский интерфейс. Проверялись как стандартные действия, так и пограничные случаи: ввод некорректных данных, отсутствие обязательных полей.

Для систематизации применялись «тест-кейс – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства поведения программного средства»[8]. Такой подход обеспечивает прозрачность, воспроизводимость и точную фиксацию ошибок.

В таблице 7 представлены разработанные тест-кейсы с указанием всех необходимых параметров и результатов их выполнения.

Таблица 7 – Тест-кейсы функционального тестирования

Название	Предусловие	Шаги	Ожидаемый результат	Фактический результат	Статус
Регистрация клиента	Форма регистрации открыта	1. Ввести в форме «Регистрация» ФИО, телефон, почту, пароль 2. Поставить галочку согласия обработки персональных данных 3. Нажать «Отправить»	Появляется сообщение «Регистрация прошла успешно», клиент переходит в личный кабинет на главную страницу	Переход в личный кабинет на главную страницу (Б1)	Успех

Продолжение таблицы 7

Регистрация с повторной почтой	Клиент с такой почтой уже зарегистрирован	1. Повторить регистрацию с той же почтой	Ошибка: «Клиент с таким email уже существует»	Ошибка отображается корректно (Б2)	Успех
Регистрация с неполными данными	Клиент заполнил все поля некорректно	1. Ввести в форме «Регистрация» некорректные ФИО, телефон, почту, пароль 2. Поставить галочку согласия обработки персональных данных 3. Нажать «Отправить»	Появляются ошибки для каждого неправильного поля ввода	Ошибки отображаются корректно (Б3)	Успех
Аутентификация с верными данными	Клиент зарегистрирован	1. Ввести почту и пароль 2. Нажать «Отправить»	Сообщение «Вход выполнен успешно!», вход в систему, переход на главную страницу	Профиль загружается (Б4)	Успех
Аутентификация с ошибкой пароля	Ввод некорректного пароля	1. Ввести почту и неверный пароль 2. Нажать «Отправить»	Появляется ошибка «Неверный пароль»	Ошибка отобразилась (Б5)	Успех
Аутентификация с ошибкой почты	Ввод некорректной почты	1. Ввести неверную почту и пароль 2. Нажать «Отправить»	Появляется ошибка «Пользователь не найден»	Ошибка отобразилась (Б6)	Успех
Доступ администратора к панели	Войти под админом	1. Перейти на «/admin/table» 2. Открыть раздел «Отчет»	Страница открывается, отчетность доступна	Доступ получен (Б7)	Успех
Редактирование профиля	Клиент или сотрудник авторизованы	1. Перейти в профиль 2. Нажать «Редактировать» 3. Изменить данные 4. Нажать «Сохранить» 5. При необходимости нажать кнопку «Выход»	Получаем сообщение «Данные успешно обновлены!», отображаются новые значения	Информация обновлена (Б8)	Успех

Продолжение таблицы 7

Поиск услуги	Главная страница загружена	1. Ввести ключевое слово услуги в строку поиска 2. Нажать «Поиск услуги»	При введении слова показываются совпадающие подсказки услуг, после нажатия кнопки отображаются релевантные услуги	Услуги найдены (Б9-10)	Успех
Фильтрация и просмотр услуг по категории	Услуги загружены	1. Выбрать категорию «Цифровая техника» 2. Просмотреть информацию	Отображаются услуги из выбранной категории, есть возможность просмотреть описание, цену и свободные слоты	Фильтрация сработала (Б11)	Успех
Корзина	Услуги загружены	1. Выбрать дату и время конкретной услуги 2. Нажать «В корзину»	Выбранная услуга перешла в раздел «Корзина»	Услуга появилась в разделе «Корзина» (Б12-13)	Успех
Оформление заказа	Услуги добавлены в корзину	1. Выбрать место оказания услуги (на выезде или офис) 2. Выбрать способ оплаты 3. Нажать «Оформить заказ» 4. При необходимости заполнить адрес и комментарий 5. Нажать «Оформить»	Появляется уведомление об успешном оформлении, появляется пустая корзина	Появилось уведомление и пустая корзина (Б14-15)	Успех
Отмена услуги (клиент)	У клиента есть в заказе одна или несколько услуг	1. Перейти на страницу «Мои заказы» 2. Нажать «Отменить»	Услуга принимает статус «Отменено», появляется пуш-уведомление об отмене услуги	Услуга приняла статус «Отменено», появилось уведомление (Б16)	Успех
Отмена услуги (сотрудник)	У клиента есть в заказе услуги	1. Перейти на страницу «Заказы» 2. Нажать «Отменить»	Услуга принимает статус «Отменено»	Услуга приняла статус «Отменено» (Б17)	Успех

Продолжение таблицы 7

Управление услугами (администратор)	Доступ к разделу «Услуги» администратора	1. Перейти в раздел «Услуги» 2. При необходимости использовать фильтры по цене и типу услуг 3. Выбрать необходимое действие (редактировать, удалить) 4. При добавлении услуги прописать: название, описание, цену и тип	Выбранные действия выполняются корректно	Выбранные действия выполняются корректно (Б18-19)	Успех
Управление сотрудниками (администратор)	Доступ к разделу «Сотрудники» администратора	1. Перейти в раздел «Сотрудники» 2. Выбрать необходимое действие (редактировать, удалить) 4. При добавлении сотрудника прописать: фио, почту, телефон и пароль	Выбранные действия выполняются корректно	Выбранные действия выполняются корректно (Б20-21)	Успех
Список заказов	У клиента есть хотя бы 1 заказ	1. Перейти к разделу «Мои заказы» 2. Просмотреть актуальные и завершенные заказы	Клиент может просматривать информацию о заказах, либо взаимодействовать с ними (завершить или отменить)	Заявленные функции доступны (Б22)	Успех
Назначение мастера	Сотрудники имеют доступ к аккаунтам	1. Мастер в «Заказы» может самостоятельно «Взять в работу» услугу 2. Администратор в разделе «Заказы» может «Назначить» мастера	Ручное или самоназначение мастера	Мастер назначается (Б23-24)	Успех

Продолжение таблицы 7

Канбан-доска	Сотрудники имеют доступ к аккаунтам	1. Сотрудники могут наблюдать в разделе «Заказы» канбан-доску, в которой отображаются статусы заказов и услуг, а также кнопки для взаимодействия	Канбан-доска открывается, кнопки работают	Канбан-доска открывается, кнопки работают (Б25)	Успех
Расширенный просмотр	Сотрудники имеют доступ к аккаунтам	1. Сотрудники могут наблюдать в разделе «Заказы» кнопку «Расширенный просмотр», при открытии которой отображаются в виде таблицы данные: номер заказа, услуга, клиент, комментарий, дата/время, мастер, статус, чекбокс для массовых действий.	Расширенный просмотр заказов открывается, фильтрация и кнопки работают	Расширенный просмотр заказов открывается, фильтрация и кнопки работают (Б26)	Успех
Финансовый отчет	Клиенты оформили хотя бы один заказ	1. Перейти в раздел «Отчет» 2. Выбрать «Финансовая» отчетность 3. Выбрать необходимые параметры	Финансовый отчет по заказам отображается в таблице	Таблица отчета создана (Б27)	Успех
Статистический отчет	Есть завершенные или отмененные или в работе заказы	1. Перейти в раздел «Отчет» 2. Выбрать «Статистический» 3. Выбрать параметры	Отчет формируется, отображаются графики	Данные отображены (Б28)	Успех
Отчет по услугам	Есть оказанные услуги	1. Перейти в раздел «Отчет» 2. Выбрать «По услугам» 3. Задать параметры	Таблица формируется с фильтрацией по типу, дате и мастеру	Отчет сформирован (Б29)	Успех

Продолжение таблицы 7

Отчет по исполнителям	Мастера выполнили хотя бы 1 услугу	1. Перейти в раздел «Отчет» 2. Выбрать «По исполнителям» 3. Выбрать тип услуги	Отчет отображает список услуг по мастерам и сумму	Данные отображены по мастерам (Б30)	Успех
Экспорт отчета в Excel	Отчет сгенерирован	1. Нажать «Экспорт»	Файл Excel скачивается	Файл скачан (Б31)	Успех
Уведомления	Сформированы уведомления	1. Перейти во вкладку «Уведомления» 2. Прочитать список 3. Нажать «Отметить как прочитанные»	Уведомления отображаются, отмечаются как прочитанные	Уведомления отобразились и обновились (Б32)	Успех
Неавторизованный пользователь	Пользователь не авторизован	1. Перейти на страницу услуг 2. Нажать «В корзину»	Переадресация на форму авторизации	Открылась форма авторизации (Б33)	Успех
Шифрование данных	Регистрация или вход доступен	1. Зарегистрировать пользователя 2. Убедиться, что пароль не передается открыто	Пароль шифруется, данные защищены	Пароль хэшируется, токен выдается (Б34)	Успех
Резервное копирование	В бд есть хотя бы один заказ	1. Настроить планировщик заданий 2. Дождаться 14:00 3. Удостовериться, что резервное копирование выполнено	Открывается строка PowerShell, добавляется backup базы данных	Строка открылась, backup выполнен (Б35-37)	Успех

В рамках функционального тестирования были проверены все основные модули веб-приложения для учета услуг и заказов. Проверке подверглись пользовательские сценарии для всех ролей: клиента, администратора и ремонтного мастера. Это позволило убедиться в корректной реализации разграничения прав доступа, устойчивой работе интерфейса и точной обработке данных.

Всего было успешно выполнено 34 тест-кейса, охватывающих полный цикл использования системы: от регистрации и входа до управления заказами, отчетностью и уведомлениями. Каждый кейс включал предусловие, пошаговые действия, ожидаемый и фактический результат, а также статус выполнения. Тестирование проводилось вручную через пользовательский интерфейс и включало как типовые, так и пограничные случаи.

Проверке подвергались:

– регистрация и вход, включая обработку ошибок и валидацию данных;

– поиск, фильтрация и выбор услуг;

– оформление заказов, управление корзиной, выбор места и способа оплаты;

– отмена услуг со стороны клиента и сотрудников;

– административные действия: добавление, редактирование и удаление услуг и сотрудников;

– отображение заказов и расширенный просмотр;

– использование канбан-доски для управления услугами;

– генерация и экспорт различных видов отчетов;

– система уведомлений и ее взаимодействие с пользователем;

– поведение системы при неавторизованных действиях;

– безопасность хранения и передачи пользовательских данных.

Результаты подтвердили полное соответствие реализованного функционала заявленным требованиям.

3.4 Нагрузочное тестирование

«Нагрузочное тестирование – исследование способности приложения сохранять заданные показатели качества при нагрузке в допустимых пределах и некотором превышении этих пределов»[8], основная цель которого – определить, насколько стабильно работает система при увеличении количества одновременных пользователей, объема запросов и интенсивности

операций за определенный период времени. Это позволяет выявить пределы допустимой нагрузки, определить узкие места и подтвердить готовность системы к реальным условиям эксплуатации.

В ходе разработки веб-приложения для учета услуг и заказов нагрузочное тестирование было сфокусировано на проверке следующих нефункциональных требований:

– время отклика интерфейса при стандартной нагрузке – не более 2 секунд;

– корректная обработка до 100 одновременных пользователей;

– отсутствие сбоев при массовой отправке заказов, авторизации и формировании отчетов.

Для всесторонней оценки применялись четыре типа нагрузочного тестирования с использованием специализированных инструментов, обеспечивающих моделирование различных сценариев эксплуатации.

1) Пиковое тестирование (Spike Testing)

Цель: анализ реакции системы на резкие кратковременные всплески трафика, чтобы проверить устойчивость приложения при внезапной нагрузке.

Реализация: с помощью утилиты Apache Benchmark выполнена команда: «ab -n 100 -c 100 http://localhost:5000/api/services», где:

– n 100 – общее количество запросов;

– c 100 – все 100 запросов отправляются одновременно (максимальный пик нагрузки);

– http://localhost:5000/api/services – целевой API -эндпоинт для получения списка услуг.

Результаты:

– Requests per second: 77.11 – сервер обрабатывал в среднем 77 запросов в секунду;

– Time per request: 1296.895 мс – среднее время отклика на один запрос;

– Failed requests: 0 – все запросы успешно обработаны;

– 90% запросов завершились за примерно 1.2 секунды, 99% – за 1.25 с;

– Transfer rate: 231.17 Kbytes/sec – хороший объем данных при передаче даже при высокой нагрузке.

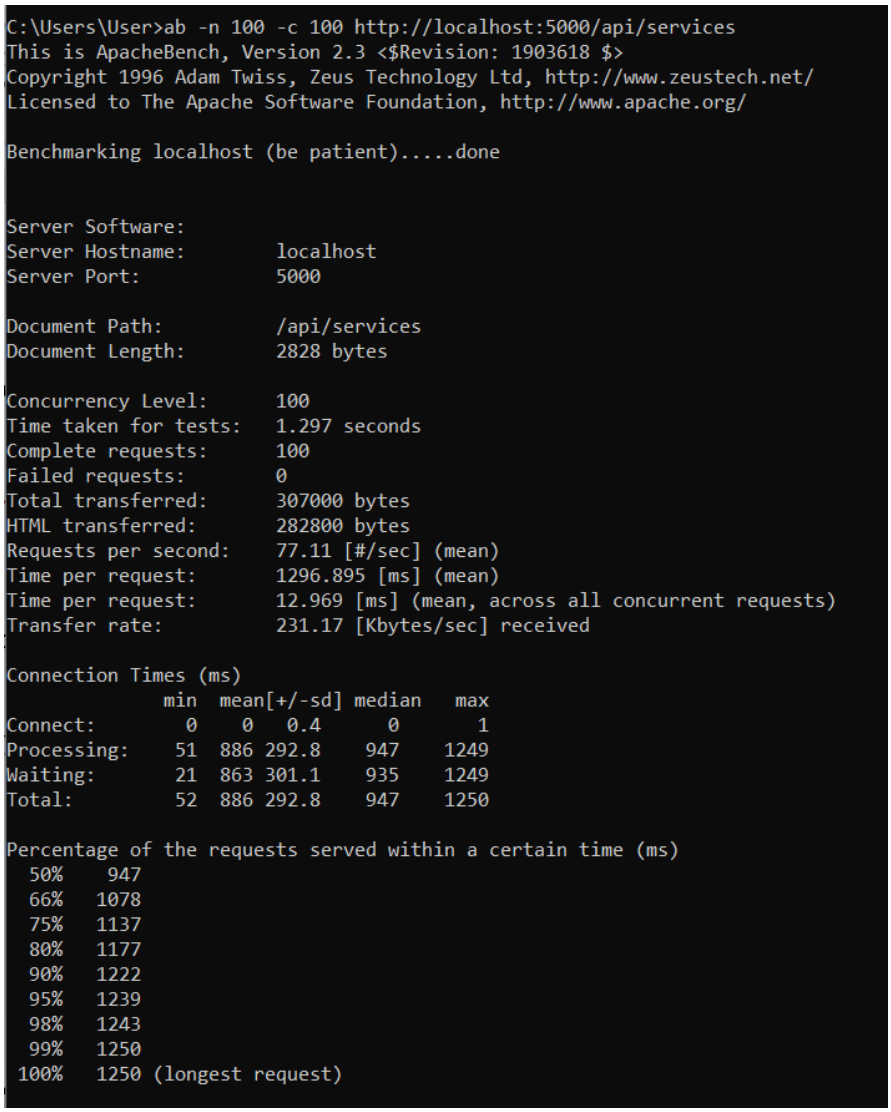


Рисунок 27 – Результат пикового тестирования

Система продемонстрировала устойчивость при резком всплеске запросов. Все обращения были успешно обработаны, с приемлемым средним временем отклика. Небольшой рост задержки до 1.3 секунд указывает на потенциальные точки оптимизации при экстремальных нагрузках.

2) Продолжительное тестирование (Soak Testing)

«Продолжительное тестирование заключается в попытке имитации классической нагрузки в течение заданного времени, для проверки того, что уровень использования ресурсов остается неизменным»[1]. Такой подход позволяет выявить скрытые проблемы, возникающие не сразу: утечки памяти, перегрев, сбои в подключении к базе данных, медленное нарастание задержек.

Реализация: при помощи инструмента K6 создаем специальный скрипт на JavaScript, представленный на рисунке 28.

```
import http from 'k6/http';
import { sleep } from 'k6';

export let options = {
  vus: 100,          // 100 пользователей одновременно
  duration: '15m'   // длительность теста
};

export default function () {
  http.get('http://localhost:5000/api/services');
  sleep(1);
}
```

Рисунок 28 – Скрипт soak_test.js

Параметры сценария:

– vus: 100 – симулируется 100 одновременных пользователей;

– duration: '15m' – нагрузка поддерживается в течение 15 минут;

– sleep(1) – пауза между действиями пользователей;

– http.get(...) – каждый пользователь делает запрос к сервису /api/services.

Результаты тестирования:

– Среднее время отклика (http_req_duration avg): 24.32ms – высокая производительность при стабильной нагрузке;

– Процент ошибок (http_req_failed): 0.00% – все 87 835 запросов обработаны без сбоев;

– Максимальное время отклика (max http_req_duration): 1.64s – даже в пике задержка не превысила допустимые пределы;

– p(90)=23.84ms, p(95)=57.07ms – 95% запросов завершались менее чем за 60 мс;

– vus_max: 100 – система стабильно выдерживала нагрузку от 100 виртуальных пользователей;

– Средняя продолжительность итерации (iteration_duration avg): 1.02s – подтверждает отсутствие деградации со временем;

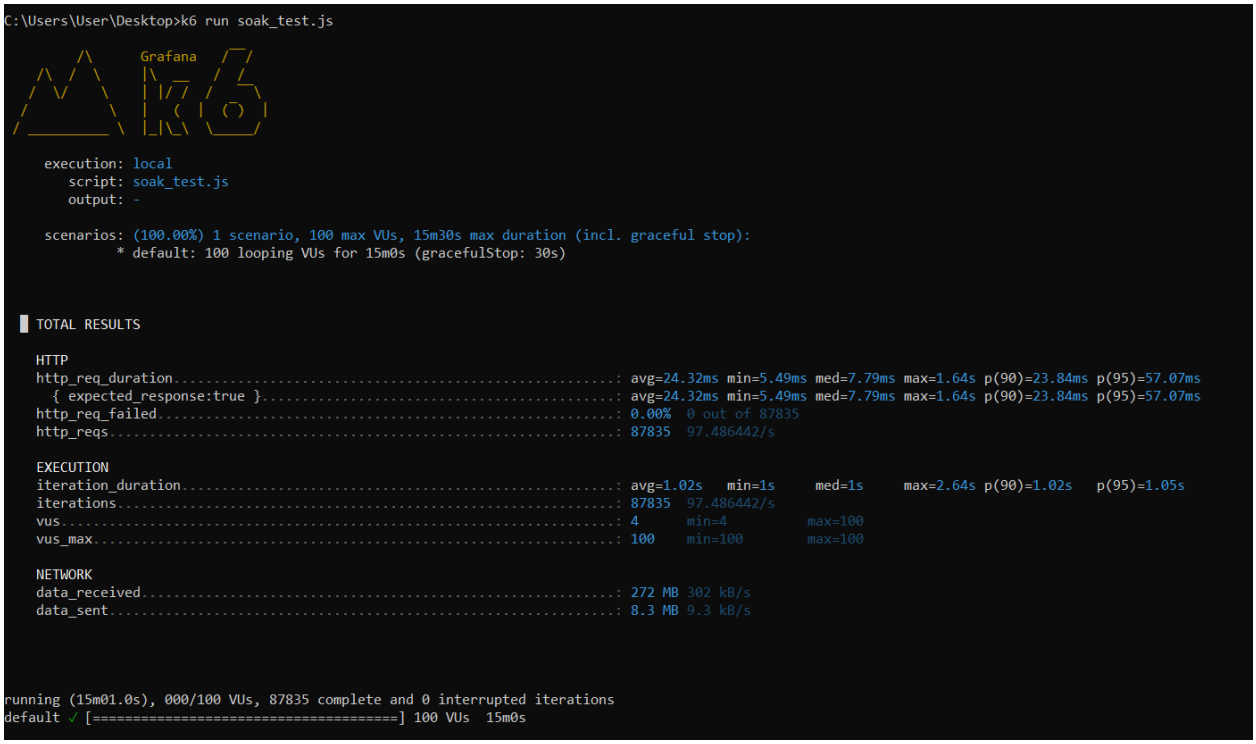


Рисунок 29 – Результат продолжительного тестирования

Веб-приложение продемонстрировало высокую устойчивость и эффективность при длительной эксплуатации. Все запросы обрабатывались корректно, без ошибок. Среднее время отклика было значительно ниже предельного значения (2 секунды), утечек памяти или признаков деградации

производительности не зафиксировано. Система готова к работе в условиях реальной умеренной постоянной нагрузки.

3) «Стрессовое тестирование (stress testing) – исследование поведения приложения при нештатных изменениях нагрузки, значительно превышающих расчётный уровень, или в ситуациях недоступности значительной части необходимых приложению ресурсов»[8].

Цель: определить пределы устойчивости системы при нагрузке выше нормы, выявить момент деградации производительности и точку отказа.

Реализация: используется инструмент K6 с режимом постепенного увеличения нагрузки выше нормального уровня. Для этого создаем скрипт stress_test.js представленный на рисунке 30.

```
import http from 'k6/http';
import { sleep } from 'k6';

export let options = {
  stages: [
    { duration: '1m', target: 100 }, // Рамп-ап до 100 пользователей за 1 минуту
    { duration: '2m', target: 200 }, // Увеличиваем до 200
    { duration: '2m', target: 300 }, // До 300
    { duration: '2m', target: 400 }, // До 400
    { duration: '2m', target: 500 }, // До 500
    { duration: '2m', target: 600 }, // Пик – 600 пользователей
    { duration: '1m', target: 0 },   // Спад
  ],
};

export default function () {
  http.get('http://localhost:5000/api/services');
  sleep(1);
}
```

Рисунок 30 – Конфигурация для стресс-теста

Пояснение параметров:

– stages: конфигурация поэтапного увеличения количества пользователей;

– target: количество виртуальных пользователей (VU), создающих нагрузку;

– duration: длительность каждого этапа;

– sleep(1): симуляция ожидания между запросами;

– http.get(...): тестируемый API-эндпоинт.

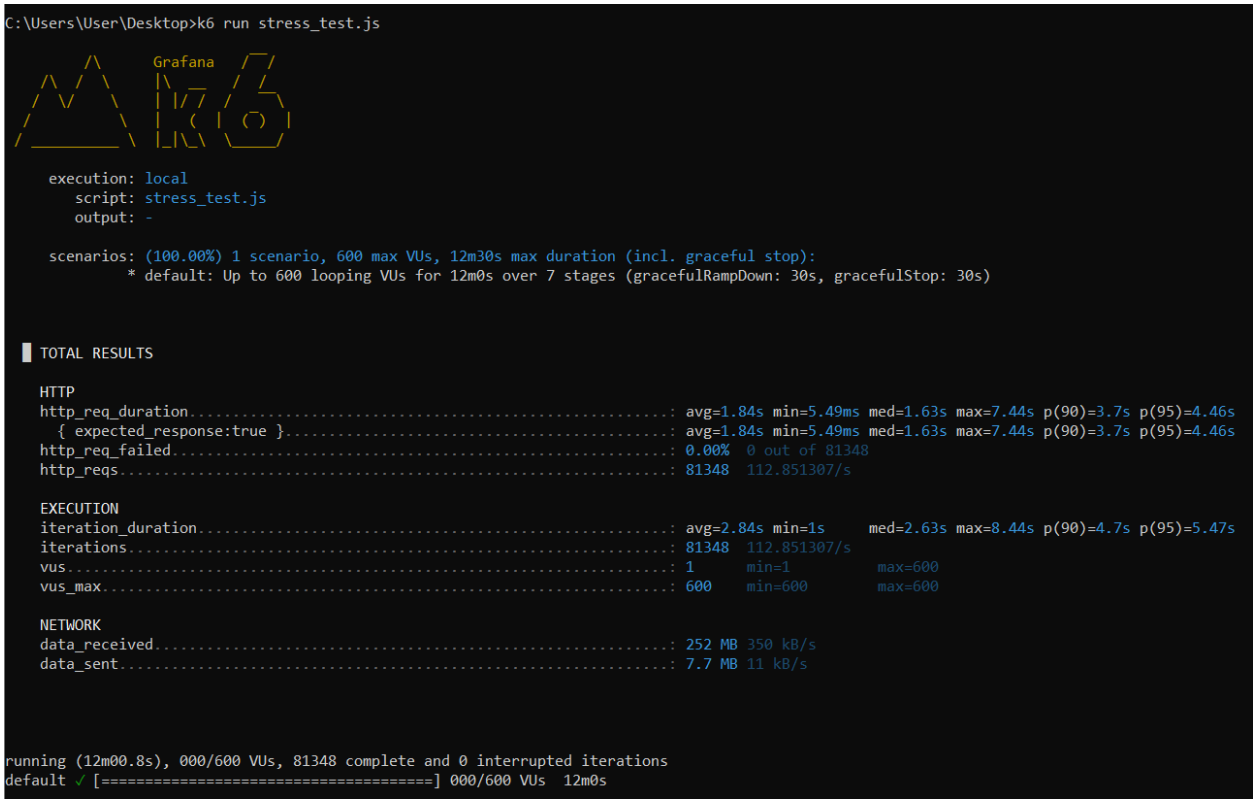


Рисунок 31 – Результат стресс-теста

Результаты теста:

– общее количество запросов: 81 348;

– ошибки: 0.00% – все запросы завершены успешно;

– среднее время отклика:1.84 сек;

– 90% запросов укладывались в 3.7 сек;

– 95% запросов в 4.46 секунды;

– максимальное время отклика: 7.44 сек;

– максимальная нагрузка: 600 одновременных пользователей.

Система успешно выдержала резкое увеличение нагрузки до 600 одновременных пользователей, не допустив ни одной ошибки. Однако

заметно увеличение времени отклика – в среднем до 1.84 секунды, а в отдельных случаях свыше 4 секунд. Это указывает на точку начала деградации производительности, которую следует учитывать при масштабировании. Приложение остается функциональным, но при длительной нагрузке такой интенсивности может потребоваться оптимизация базы данных или распределение нагрузки через балансировщик.

4) Постепенное наращивание нагрузки (Ramp-Up Testing)

Цель: проверить, при каком уровне нагрузки система начинает замедляться или давать сбои. Это позволяет определить порог производительности и спрогнозировать масштабируемость.

Реализация: для проведения Ramp-Up теста использовался инструмент K6. В скрипте ramp_up_test.js задается последовательное увеличение количества виртуальных пользователей в несколько этапов, с сохранением времени ожидания между запросами. Структура скрипта показана на рисунке 32.

```
import http from 'k6/http';
import { sleep } from 'k6';

export let options = {
  stages: [
    { duration: '2m', target: 50 },   // медленный старт
    { duration: '2m', target: 100 },  // плавное увеличение
    { duration: '2m', target: 200 },  // далее больше
    { duration: '2m', target: 400 },  // выше нормы
    { duration: '2m', target: 600 },  // близко к пределу
    { duration: '2m', target: 0 },    // спад нагрузки
  ],
};

export default function () {
  http.get('http://localhost:5000/api/services');
  sleep(1);
}
```

Рисунок 32 – Конфигурация для постепенного наращивания нагрузки

Пояснение параметров:

– stages – последовательные этапы увеличения нагрузки. Каждый этап задает продолжительность (duration) и целевое число виртуальных пользователей (target), например:

– 2 мин – 50 VU;

– 2 мин – 100 VU;

– 2 мин – 200 VU;

– 2 мин – 400 VU;

– 2 мин – 600 VU;

– 2 мин – 0 VU (снижение до нуля, завершение теста).

– vus – виртуальные пользователи, одновременно выполняющие запросы;

– sleep(1) – пауза между действиями, имитирующая реальное поведение пользователя;

– http.get(...) – вызов целевого эндпоинта API /api/services.

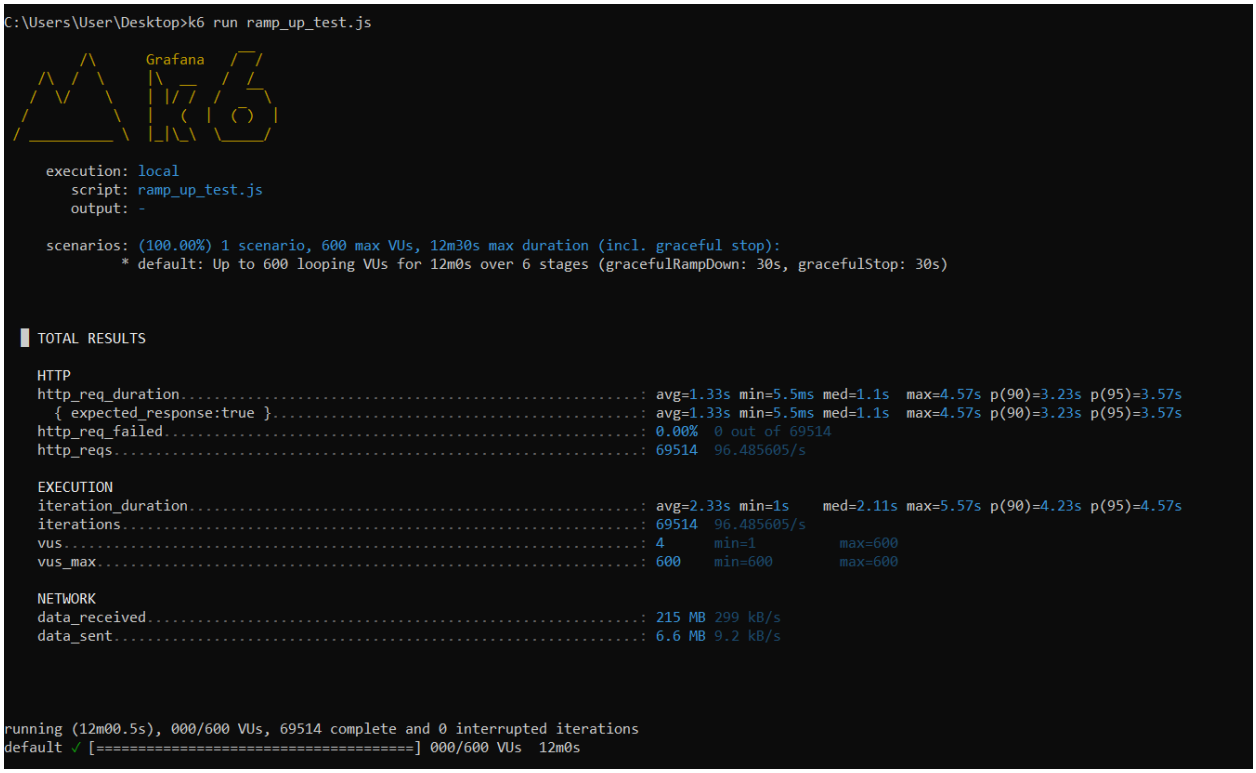


Рисунок 33 – Результат нагрузочного тестирования

Результаты тестирования:

– количество запросов: 69 514;

– ошибки: 0%;

– среднее время отклика (avg): 1.33 сек;

– 90% запросов укладывались в 3.23 сек;

– 95% запросов в 3.57 сек;

– максимальный отклик: 4.57 сек.

Система адекватно реагировала на постепенное увеличение количества пользователей, не допуская сбоев или потерь запросов. Однако начиная с 400-600 виртуальных пользователей время отклика заметно увеличилось (до 3-4.5 секунд), что указывает на начало деградации производительности.

Сравнительный анализ результатов нагрузочного тестирования:

Для наглядного сопоставления производительности веб-приложения при разных сценариях нагрузок была составлена итоговая таблица 8, включающая ключевые метрики: количество пользователей, число запросов, ошибки, среднее и максимальное время отклика, а также значения p90 и p95.

Таблица 8 – Сравнение метрик при различных типах нагрузочного тестирования

Методика	Кол-во VU	Кол-во запросов	Ошибки (%)	Среднее время отклика (s)	Макс. Время отклика (s)	90% запросов меньше или равно (с)	P95 (s)
Пиковая нагрузка	100	100	0	1.3	1.25	1.2	1.25
Длительная нагрузка	100	87835	0	0.024	1.64	0.02384	0.05707
Стресс-тестирование	600	81348	0	1.84	7.44	3.7	4.46
Постепенное наращивание	600	69514	0	1.33	4.57	3.23	3.57

Также была построена диаграмма, представленная на рисунке 34, для визуального сопоставления времени отклика по трем основным метрикам (среднее, 90% запросов, максимум) при каждом типе тестирования.

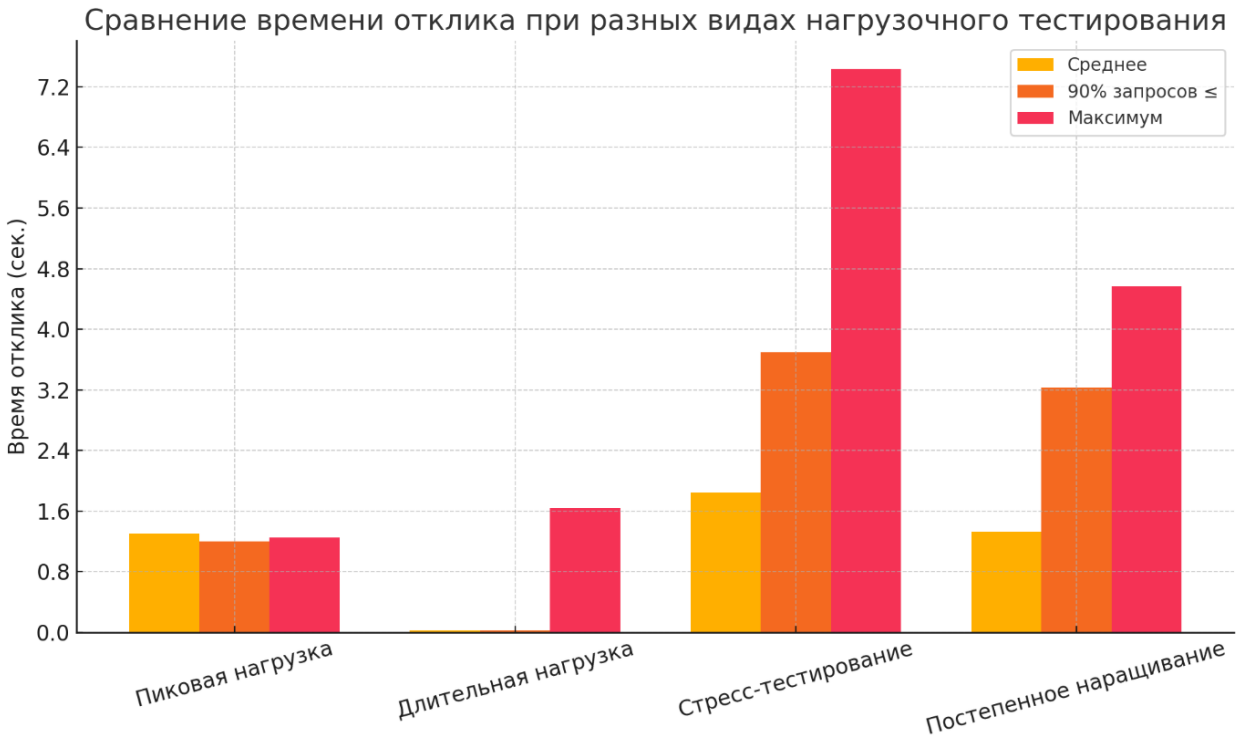


Рисунок 34 – Сравнение времени отклика при различных нагрузках

Полученные результаты помогают определить границу стабильной работы системы – примерно до 300-400 одновременных пользователей. Это ценный ориентир при планировании будущих масштабов использования и необходимости внедрения балансировщика нагрузки или оптимизации кода и запросов к БД.

Для проведения нагрузочного тестирования использовались два инструмента:

Apache Benchmark (ab) – для проверки устойчивости к пиковым нагрузкам [15];

K6 – для моделирования реалистичных сценариев: длительная нагрузка (Soak), стресс (Stress) и постепенное наращивание нагрузки (Ramp-Up).

Проведенное нагрузочное тестирование подтвердило стабильную и предсказуемую работу веб-приложения при различных типах нагрузки. Система успешно выдержала как кратковременные пики, так и длительную работу под высокой нагрузкой, без ошибок или сбоев. Однако при приближении к 600 одновременным пользователям фиксируется рост времени отклика, что обозначает порог производительности. Это позволяет сделать вывод о надежности реализованного решения в реальных условиях и сформулировать рекомендации для последующего масштабирования и оптимизации.

3.5 Выводы по тестированию

В ходе тестирования веб-приложения для учета услуг и заказов была проведена проверка функциональности, производительности и устойчивости к нагрузке. Результаты подтвердили соответствие системы установленным требованиям и ее готовность к реальной эксплуатации.

Функциональное тестирование:

Проверены все основные пользовательские сценарии для клиента, администратора и ремонтного мастера. Тестированию подверглись модули регистрации, входа, редактирования профиля, фильтрации и выбора услуг, корзины, оформления и управления заказами, генерации отчетов и система уведомлений.

Из 34 разработанных тест-кейсов успешно выполнены все. Зафиксированы корректная обработка ошибок, надежное разграничение прав доступа и стабильная реакция интерфейса на типовые и граничные действия.

Нагрузочное тестирование:

Проверка охватила четыре сценария: пиковую нагрузку (100 параллельных запросов), длительное воздействие (100 пользователей в течение 15 минут), стресс (до 600 пользователей) и плавное наращивание нагрузки.

Среднее время отклика при стандартной и умеренной нагрузке не превышало 2 секунд. При 400-600 одновременных пользователях отмечен рост задержек, что указывает на начало деградации производительности. Порог стабильной работы – до 300-400 пользователей.

Инструменты:

Использовались Apache Benchmark (для оценки реакции на пиковые запросы) и K6 (для длительных и прогрессивных сценариев нагрузки). Проверка охватила интерфейс, API и серверную часть приложения.

Приложение реализует заявленный функционал, устойчиво работает под нагрузкой, не содержит критичных сбоев и готово к внедрению. Результаты тестирования подтверждают надежность системы и дают основу для ее масштабирования при росте пользовательской базы.

Выводы по третьей главе

В третьей главе рассмотрены процессы разработки и всестороннего тестирования веб-приложения для учета услуг и заказов. Проведенное функциональное тестирование охватило все основные пользовательские сценарии и подтвердило корректную реализацию заявленного функционала. Нагрузочное тестирование показало стабильную работу системы при различных типах нагрузки, включая пиковую, длительную и стрессовую, без потери запросов и с допустимым временем отклика. Результаты тестирования подтвердили надежность, масштабируемость и готовность приложения к реальному использованию в сервисной компании.

Заключение

В ходе выпускной квалификационной работы проведен анализ текущей информационной системы компании «ВорлдИнтерТех Рус» и определены ее ограничения. Сформулированы функциональные и нефункциональные требования к будущему решению, разработана архитектура информационной системы.

Разработанное веб-приложение реализует регистрацию и авторизацию пользователей с разграничением ролей, оформление и отслеживание заказов, работу с услугами, генерацию отчетов и систему уведомлений. Были внедрены механизмы безопасности, включая JWT-аутентификацию, защиту маршрутов и шифрование данных.

Для подтверждения корректности работы программного обеспечения проведено комплексное тестирование. Система продемонстрировала стабильную работу до 300-400 одновременных пользователей, сохраняя приемлемое время отклика и отказоустойчивость.

Практическая ценность работы заключается в готовом решении, адаптированном под нужды конкретной компании и пригодном для дальнейшего внедрения. Разработанное ПО может служить основой для масштабируемых систем учета, аналитики и взаимодействия с клиентами.

В дальнейшем возможно расширение функциональности системы: интеграция с платежными сервисами, реализация мобильной версии, подключение аналитических модулей и внедрение средств искусственного интеллекта для предиктивной оценки нагрузки. Это позволит повысить гибкость и конкурентоспособность информационной системы в условиях роста требований и пользовательской базы.

Таким образом, цели выпускной квалификационной работы достигнуты, задачи успешно решены, а созданный программный продукт полностью соответствует предъявленным требованиям и готов к практическому применению.

Список используемой литературы и используемых источников

1. Бевзенко С. А. Основные виды классификации нагрузочного тестирования // Вестник науки. 2024. №1 (70). URL: https://cyberleninka.ru/article/n/osnovnye-vidy-klassifikatsii-nagruzochnogo-testirovaniya (дата обращения: 14.03.2025).

2. Бэнкер К. MongoDB в действии / Пер. с англ. Слинкина А. А. – М. : ДМК Пресс, 2012. – 394 с.

3. ГОСТ Р ИСО/МЭК 12119-2000 – Информационная технология. Пакеты программ. Требования к качеству и тестирование.

4. Документация Express.js [Электронный ресурс] – URL: https://expressjs.com/ (дата обращения: 31.11.2024).

5. Документация Node.js [Электронный ресурс] – URL: https://nodejs.org/docs/latest/api/ (дата обращения: 31.11.2024).

6. Документация React [Электронный ресурс] – URL: https://ru.legacy.reactjs.org/docs/getting-started.html (дата обращения: 31.11.2024).

7. Дейт, К. Дж. Основы систем баз данных / К. Дж. Дейт. – М.: Вильямс, 2019. – 1040 с..

8. Куликов С.С. Тестирование программного обеспечения. Базовый курс. – 3-е изд. – Минск: Четыре четверти, 2020. – 312 с.

9. Ларин С.Н., Лазарева Л.С., Ларина Т.С. Модели, методы, показатели, характеристики и метрики, применяемые в экспертных системах оценки качества разработки и создания инновационных программных проектов // Региональная экономика: теория и практика. – 2017. – Т. 15, № 6. – С. 1187–1198. – DOI: 10.24891/re.15.6.1187.

10. Леоненков А.В. Самоучитель UML. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2004. – 432 с.

11. Макконнелл С. Совершенный код. Мастер-класс / Пер. с англ. – М. : Издательство «Русская редакция», 2010. – 896 с.

12. Фаулер М. UML. Основы. 3-е изд. / пер. с англ. А. Петухова. – СПб. : Символ-Плюс, 2004. – 192 с.

13. Федеральный закон РФ от 27.07.2006 № 152-ФЗ "О персональных данных" (с изм. от 29.12.2024).

14. Фриман, Э. Изучаем JavaScript: Полное руководство / Э. Фриман. – М.: Вильямс, 2020. – 624 с.

15. Apache HTTP Server Project. ab – Apache HTTP server benchmarking tool [Электронный ресурс]. – Режим доступа: https://httpd.apache.org/docs/2.4/programs/ab.html (дата обращения: 15.03.2025).

16. Beyer B., Lewandowski P., Oprea A., Blankinship P., Adkins H., Stubblefield A. Building Secure and Reliable Systems: O'Reilly Media, 2020.

17. Goodwin, T. Web Application Architecture: Principles and Practices / T. Goodwin. – London: Wiley, 2020. – 450 p.

18. Hunter II, T. Distributed Systems with Node.js. – O'Reilly Media, Inc., 2021. – 377 p.

19. ISO/IEC 9126-1:2001. Software engineering – Product quality – Part 1: Quality model [Электронный ресурс]. – Geneva: International Organization for Standardization, 2001. – Режим доступа: https://standards.iteh.ai/catalog/standards/sist/d4ab62c2-7a1f-4586-b33d-25bcf8cf19c1/iso-iec-9126-1-2001 (дата обращения: 10.01.2025).

20. Simpson, K. You Don't Know JS Yet (book series). – 2nd ed. – GetiPub & Leanpub, 2020. – 145 p.

Приложение А

Спецификации диаграммы вариантов использования

Таблица А1 – Описательная спецификация прецедента «Поиск, фильтрация и просмотр услуг»

Прецедент	Поиск, фильтрация и просмотр услуг
Краткое описание	Клиент может получать информацию об услугах с возможностью фильтрации
Главные актеры	Клиент
Второстепенные актеры	Авторизованный и неавторизованный клиенты
Предусловия	Доступ к приложению
Основной поток	Выбор фильтров, поиск по ключевым словам, просмотр описания
Постусловия	1. Клиент открывает страницу услуг 2. Применяет фильтры и/или выполнен поиск 3. Результаты отображаются на экране

Таблица А2 – Описательная спецификация прецедента «Просмотр заказов»

Прецедент	Просмотр заказов
Краткое описание	Клиент просматривает список оформленных заказов
Главные актеры	Клиент
Второстепенные актеры	Авторизованный клиент
Предусловия	Клиент авторизовался
Основной поток	Отображение заказов, разделение на актуальные и завершенные
Постусловия	1. Клиент открывает раздел «Заказы» 2. Просматривает актуальные и завершенные заказы

Таблица А3 – Описательная спецификация прецедента «Авторизация»

Прецедент	Авторизация
Краткое описание	Вход в систему с логином и паролем
Главные актеры	Клиент, мастер, администратор
Второстепенные актеры	Неавторизованный клиент
Предусловия	Наличие учетной записи
Основной поток	Ввод логина (почты) и пароля
Постусловия	1. Открыта форма входа 2. Пользователь вводи необходимые данные 3. После успешной авторизации получен доступ к функционалу согласно своей роли

Таблица А4 – Описательная спецификация прецедента «Выход из аккаунта»

Прецедент	Выход из аккаунта
Краткое описание	Завершение пользовательской сессии
Главные актеры	Клиент, мастер, администратор
Второстепенные актеры	Авторизованный клиент
Предусловия	Успешная авторизация
Основной поток	Нажатие на кнопку выхода
Постусловия	1. Пользователь инициирует завершение сессии 2. Выполнен выход из системы 3. Пользователь перенаправлен на главную страницу

Таблица А5 – Описательная спецификация прецедента «Редактирование профиля»

Прецедент	Редактирование профиля
Краткое описание	Изменение ФИО, телефона и почты пользователя
Главные актеры	Клиент, мастер, администратор
Второстепенные актеры	Авторизованный клиент
Предусловия	Пользователь авторизован
Основной поток	Ввод новых данных и их сохранение
Постусловия	4 Клиент открывает форму редактирования в разделе профиля 5 Заполняет обновленные данные 6 Сохраняет изменения

Таблица А6 – Описательная спецификация прецедента «Расписание мастера»

Прецедент	Расписание мастера
Краткое описание	Просмотр календаря услуг со статусом «В работе»
Главные актеры	Мастер
Второстепенные актеры	Нет
Предусловия	Авторизация мастера и услуги со статусом «В работе»
Основной поток	Открытие календаря, отображение занятых слотов
Постусловия	1. Мастер авторизуется под своим логином 2. Открывает раздел календаря 3. Просматривает услуги, взятые в работу

Таблица А7 – Описательная спецификация прецедента «Работа с корзиной и оформление заказа»

Прецедент	Работа с корзиной / Оформление заказа
Краткое описание	Клиент добавляет услуги в корзину, оформляет заказ
Главные актеры	Клиент
Второстепенные актеры	Авторизованный клиент
Предусловия	Выбор услуги, свободные дату и время, места оказания услуги, способ оплаты и комментарий
Основной поток	Заказ оформлен и передан в систему
Постусловия	1. Клиент выбирает услуги, указывает дату и время, добавляет в корзину 2. Указывает адрес и способ оплаты 3. Оформляет заказ, который сохраняется в системе

Таблица А8 – Описательная спецификация прецедента «Завершить услугу»

Прецедент	Завершить услугу
Краткое описание	Изменение статуса услуги на «Готово»
Главные актеры	Клиент
Второстепенные актеры	Авторизованный клиент, администратор
Предусловия	Статус услуги «На проверке»
Основной поток	Подтверждение завершения услуги
Постусловия	1. Клиент открывает раздел «Заказы» 2. Нажимает «Завершить», подтверждая выполнение 3. Статус изменен на «Готово» 4. Подтверждает действие

Таблица А9 – Описательная спецификация прецедента «Отменить услугу»

Прецедент	Отменить услугу
Краткое описание	Отмена услуги из заказа
Главные актеры	Клиент
Второстепенные актеры	Авторизованный клиент, мастер, администратор
Предусловия	Услуга еще не выполнена
Основной поток	Нажатие кнопки отмены
Постусловия	1. Необходимо найти интересующую услугу 2. Инициировать действие отмены 3. Статус услуги обновлен на «Отменено»

Таблица А10 – Описательная спецификация прецедента «Взять в работу / Перевести на проверку»

Прецедент	Взять в работу / Перевести на проверку
Краткое описание	Изменение статуса услуги мастером
Главные актеры	Мастер
Второстепенные актеры	Администратор
Предусловия	Назначение исполнителя, доступ к заказу
Основной поток	Нажатие по кнопке смены статуса
Постусловия	1. Открыты канбан-доска или расширенный просмотр заказов 2. В нужной колонке выбрать услугу 3. Перевести статус «В работе» или «На проверке»

Таблица А11 – Описательная спецификация прецедента «Назначить мастера»

Прецедент	Назначить мастера
Краткое описание	Присвоение исполнителя к услуге
Главные актеры	Администратор
Второстепенные актеры	Нет
Предусловия	Заказ без исполнителя
Основной поток	Выбор мастера из списка
Постусловия	1. Открыт интерфейс управления заказами 2. Администратор выбирает нужную услугу 3. Назначает конкретного мастера

Таблица А12 – Описательная спецификация прецедента «Управление заказами»

Прецедент	Управление заказами
Краткое описание	Изменение статуса, фильтрация, просмотр, взаимодействие с заказами
Главные актеры	Администратор, мастер
Второстепенные актеры	Нет
Предусловия	Авторизация под нужной ролью
Основной поток	Изменение статуса, фильтрация, работа с канбан-доской
Постусловия	1. Выполнена фильтрация по параметрам 2. Изменены статусы соответствующих услуг 3. Интерфейс заказов обновлен с учетом новых данных

Таблица А13 – Описательная спецификация прецедента «Управление сотрудниками»

Прецедент	Управление сотрудниками
Краткое описание	Добавление, редактирование, удаление мастеров
Главные актеры	Администратор
Второстепенные актеры	Мастер
Предусловия	Авторизация администратора
Основной поток	CRUD-операции по сотрудникам
Постусловия	1. Добавлен, изменен или удален профиль мастера 2. Изменения сохранены 3. Список сотрудников актуализирован

Таблица А14 – Описательная спецификация прецедента «Управление услугами»

Прецедент	Управление услугами
Краткое описание	Добавление, редактирование, удаление услуг
Главные актеры	Администратор
Второстепенные актеры	Нет
Предусловия	Авторизация администратора
Основной поток	CRUD-операции по услугам
Постусловия	1. Выполнены действия добавления, редактирования или удаления услуги 2. Обновлены параметры услуги 3. Информация отражена в пользовательском интерфейса

Таблица А15 – Описательная спецификация прецедента «Составление отчетности»

Прецедент	Составление отчетности
Краткое описание	Формирование финансовых и статистических отчетов, а также по услугам и мастерам
Главные актеры	Администратор
Второстепенные актеры	Нет
Предусловия	Авторизация администратора
Основной поток	Выбор необходимых параметров, фильтрация, экспорт
Постусловия	1. Выбраны параметры отчета 2. Сформирован отчет по выбранным критериям 3. Данные экспортированы в нужном формате

Приложение Б

Результаты функционального тестирования

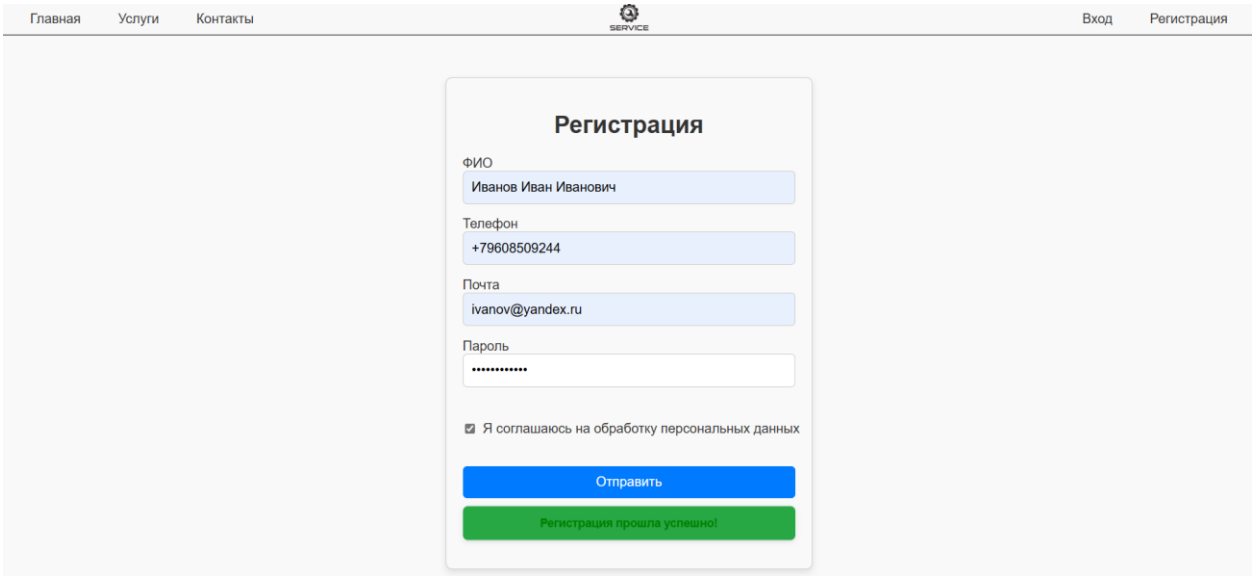


Рисунок Б1 – Успешная регистрация клиента

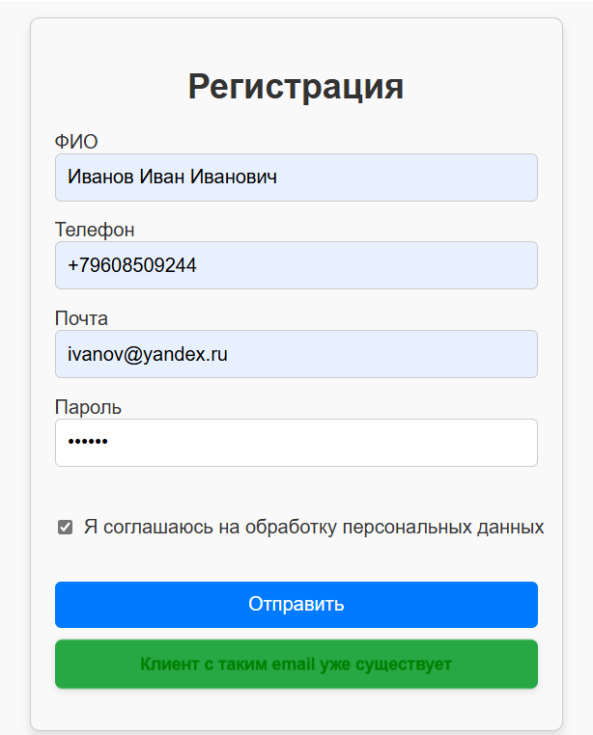


Рисунок Б2 – Регистрация с повторной почтой

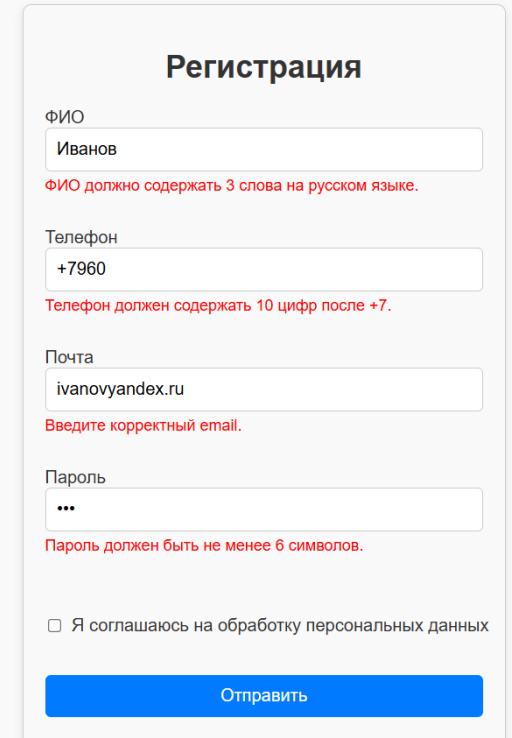


Рисунок Б3 – Неправильное заполнение формы регистрации

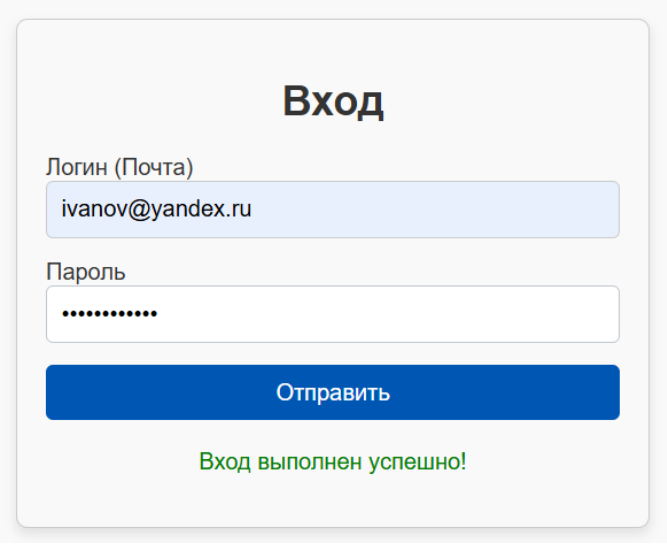


Рисунок Б4 – Успешная аутентификация

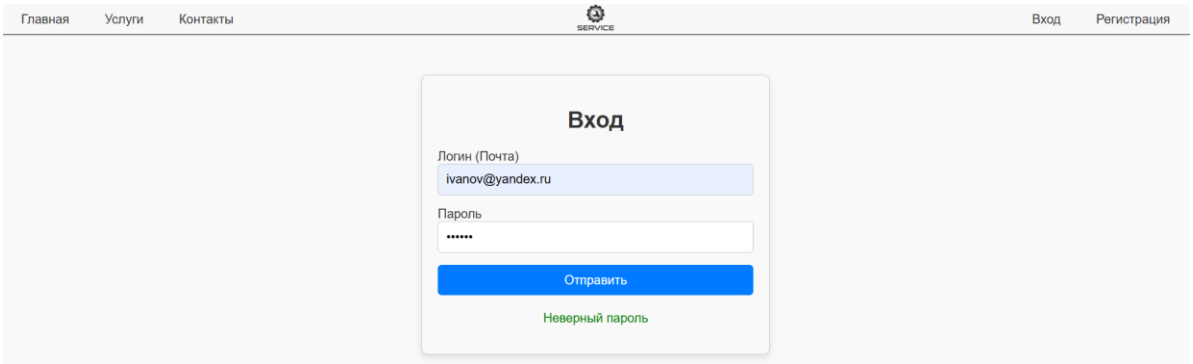


Рисунок Б5 – Аутентификация с неверными данными

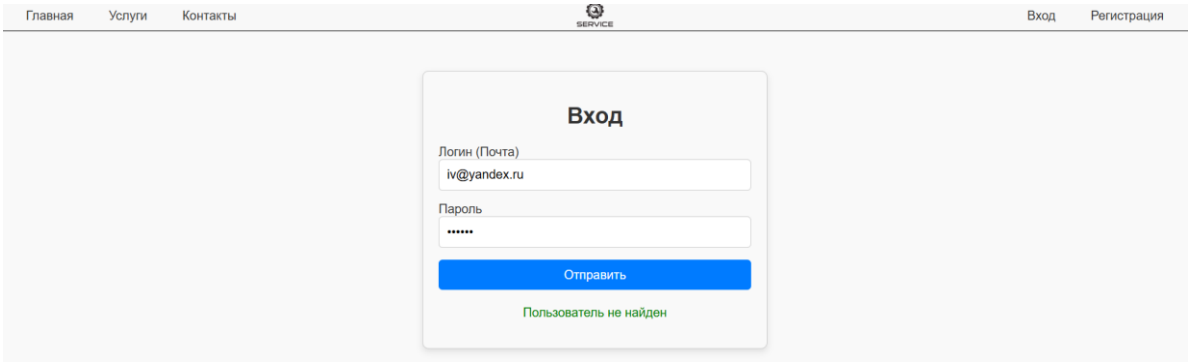


Рисунок Б6 – Аутентификация с неверными данными

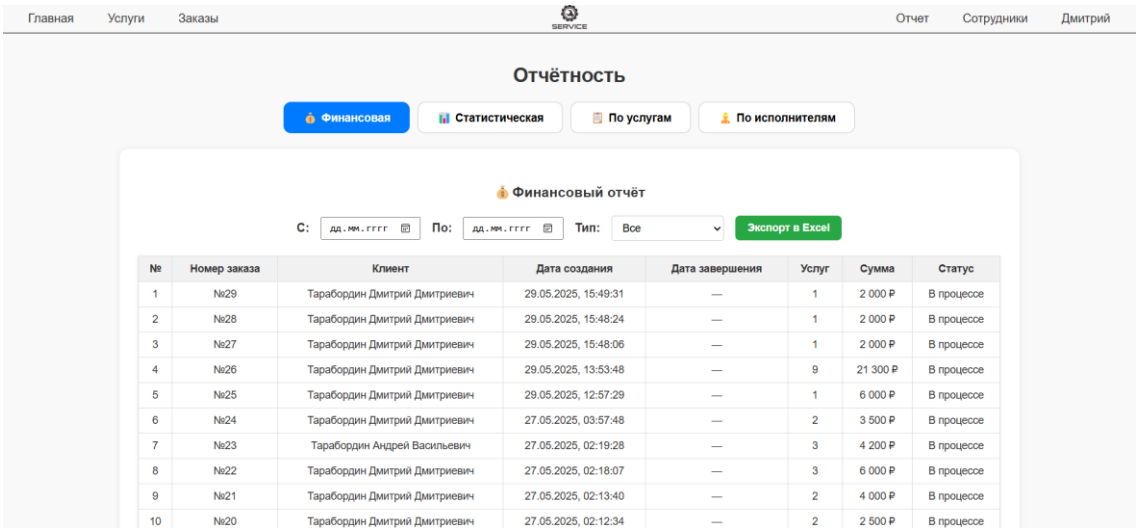


Рисунок Б7 – Проверка доступности по ролям

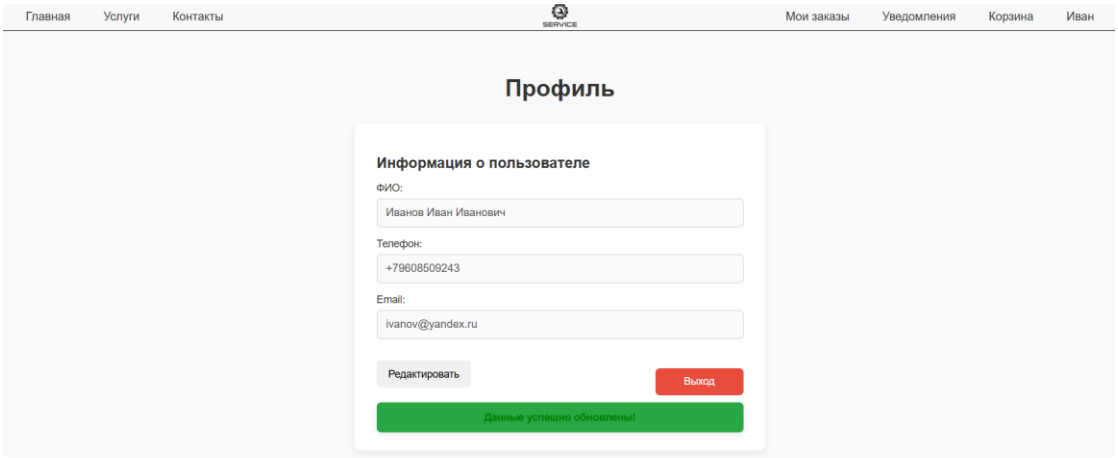


Рисунок Б8 – Успешное редактирование данных в профиле

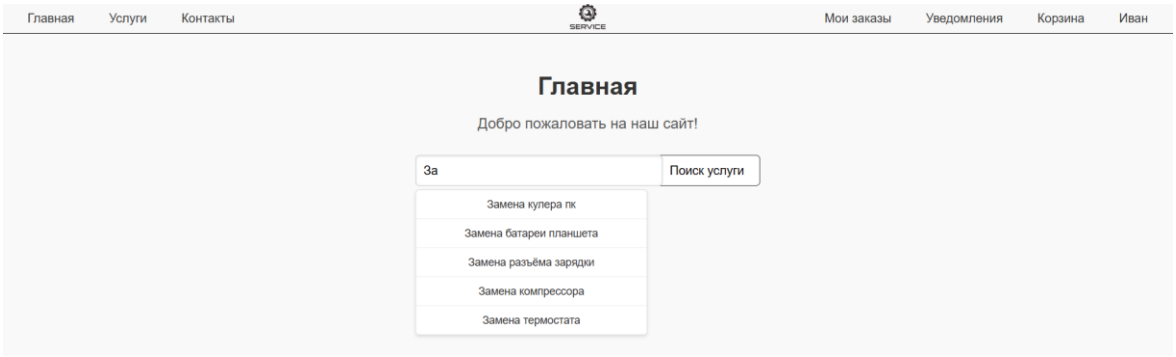


Рисунок Б9 – Поиск в строке услуг

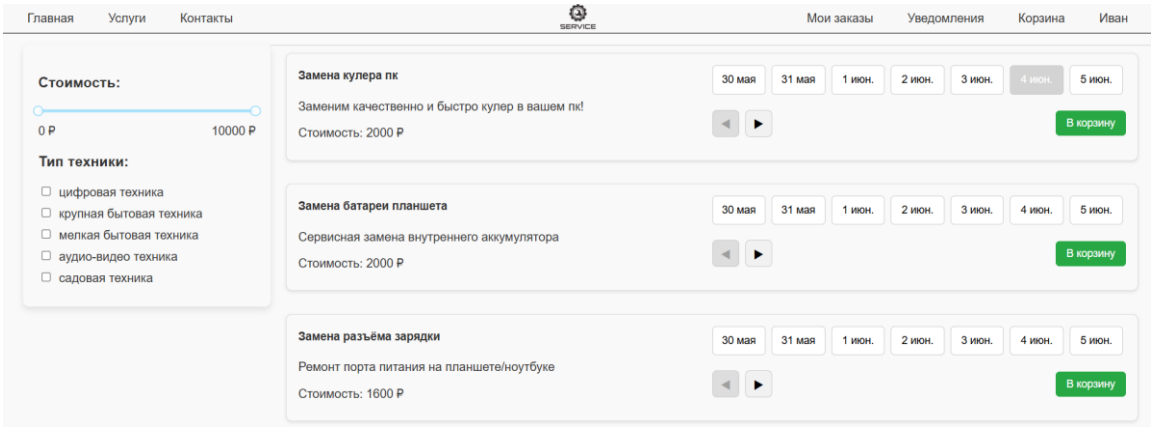


Рисунок Б10 – Результат поиска услуг

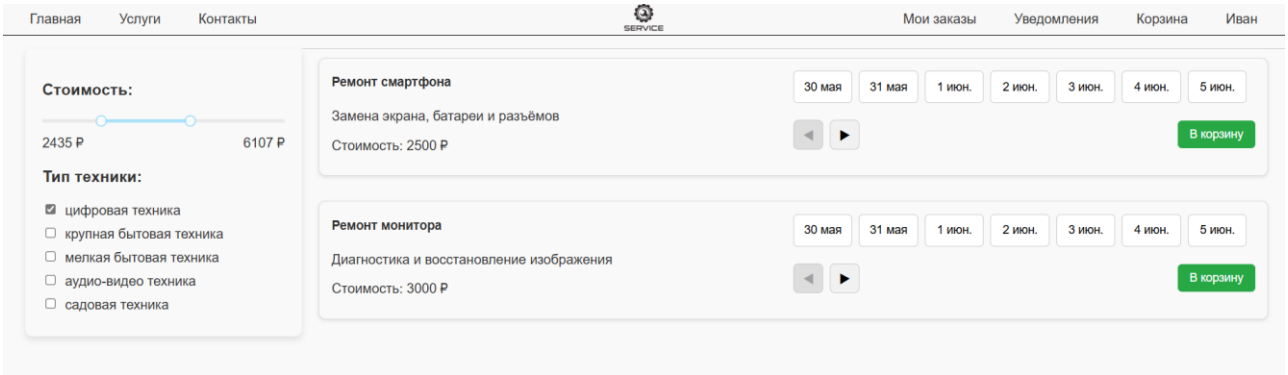


Рисунок Б11 – Результат фильтрации услуг

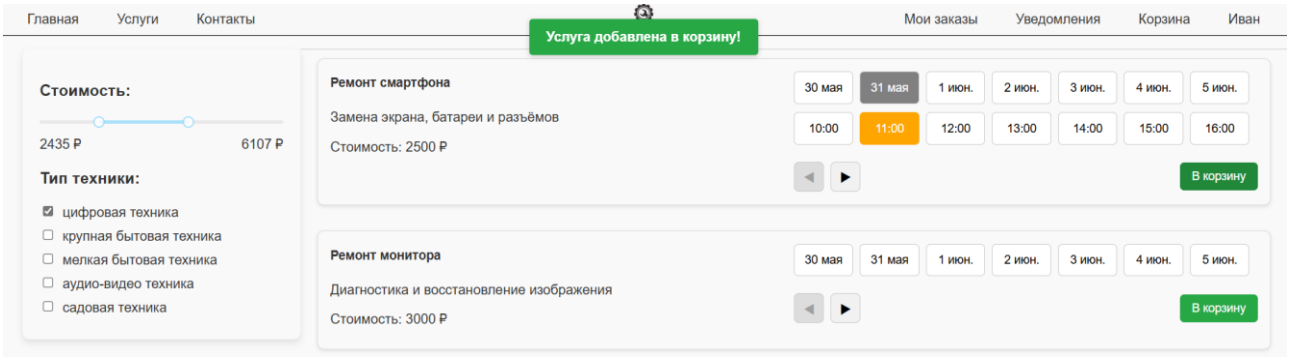


Рисунок Б12 – Результат выбора времени и даты услуги, нажатия «В корзину»

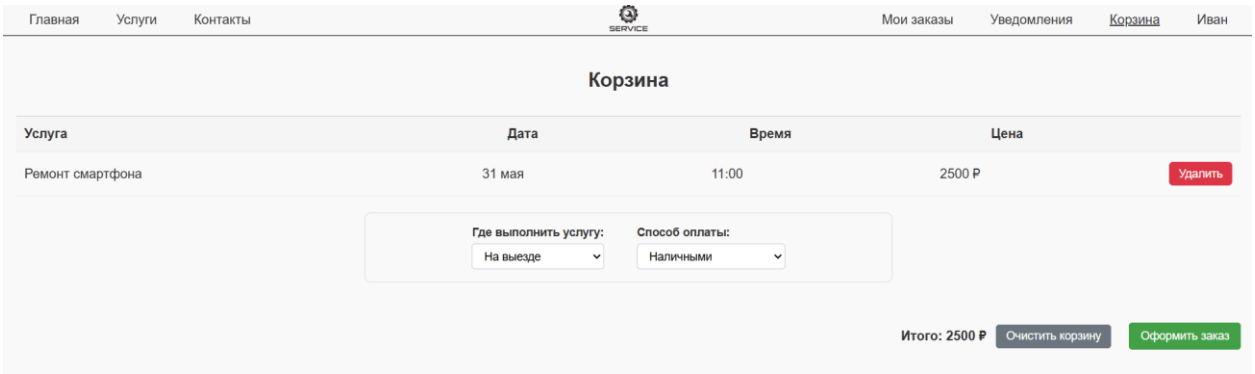


Рисунок Б13 – Результат выбора услуги в «Корзине»

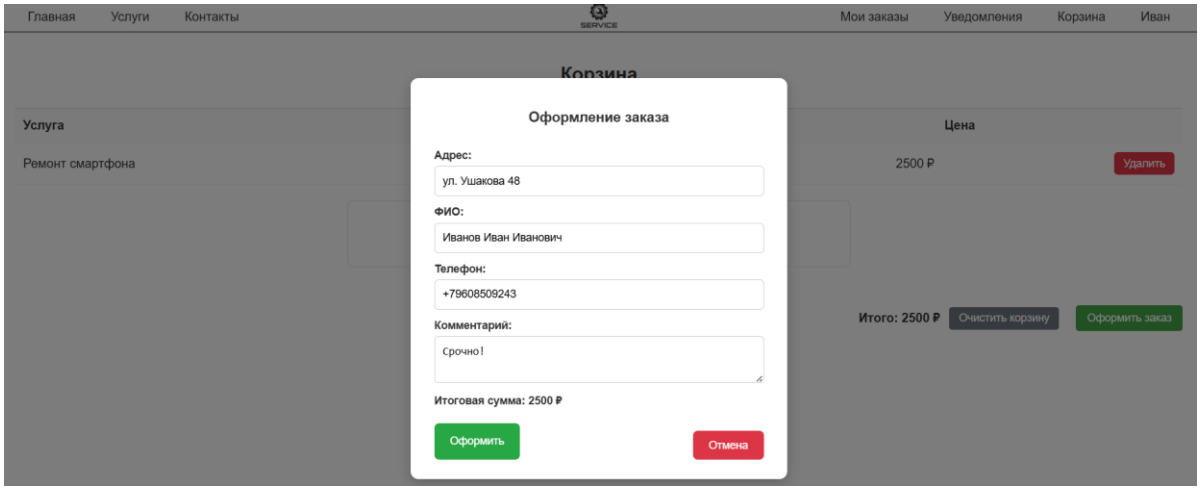


Рисунок Б14 – Оформление заказа

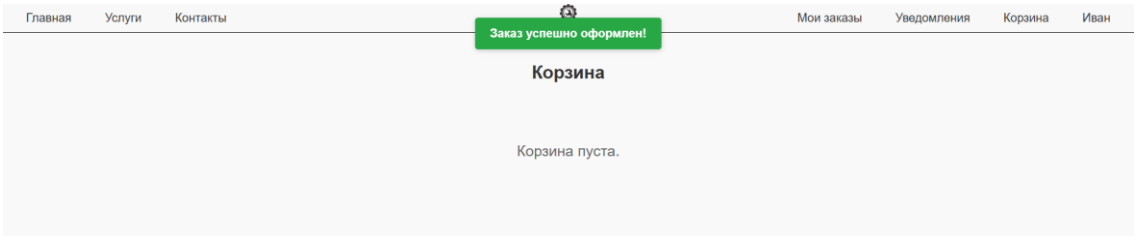


Рисунок Б15 – Результат оформления заказа

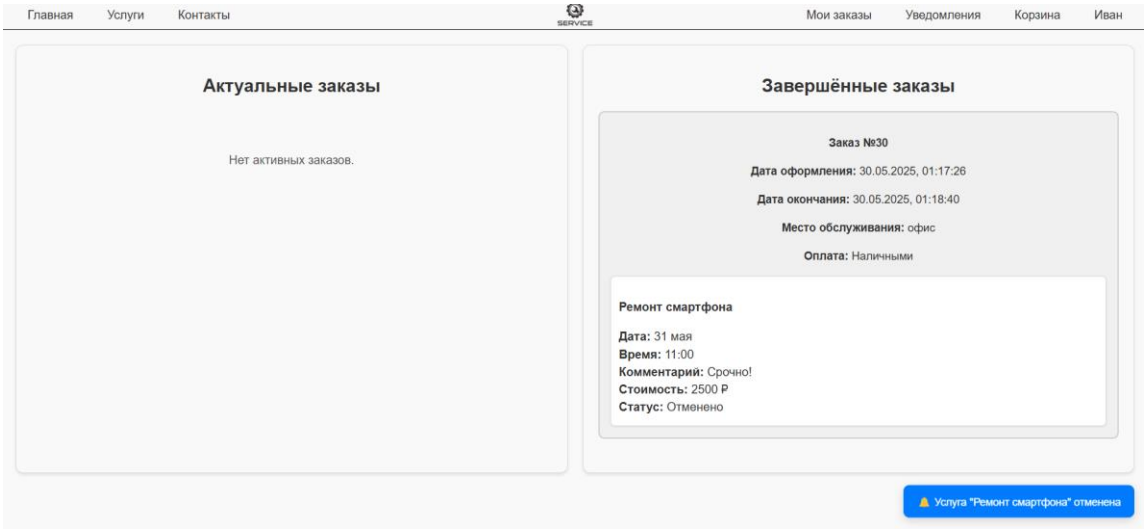


Рисунок Б16 – Отмена услуги клиентом

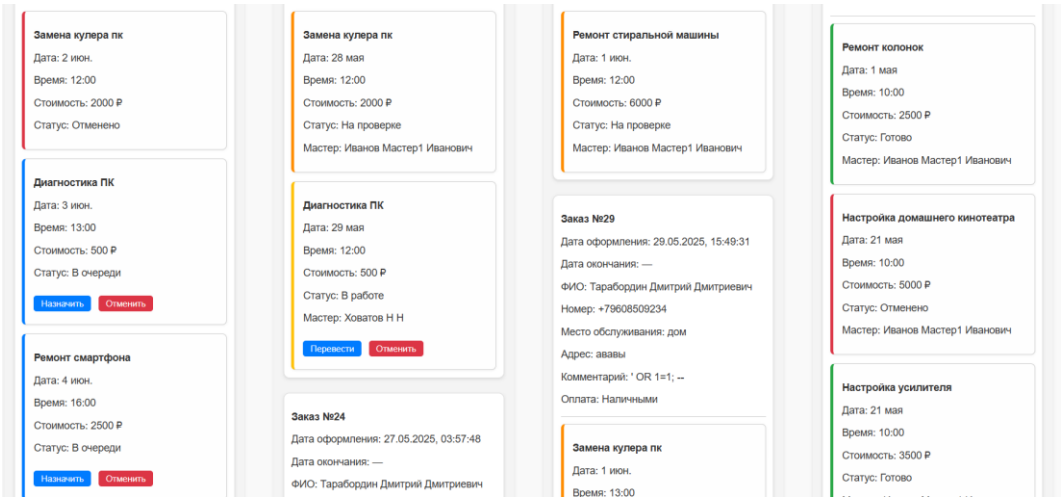


Рисунок Б17 – Отмена услуги администратором

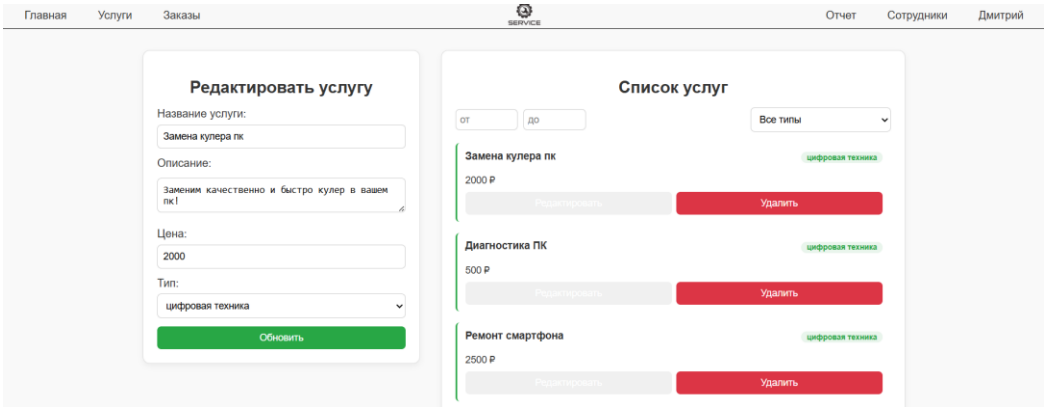


Рисунок Б18 – Редактирование цены услуги

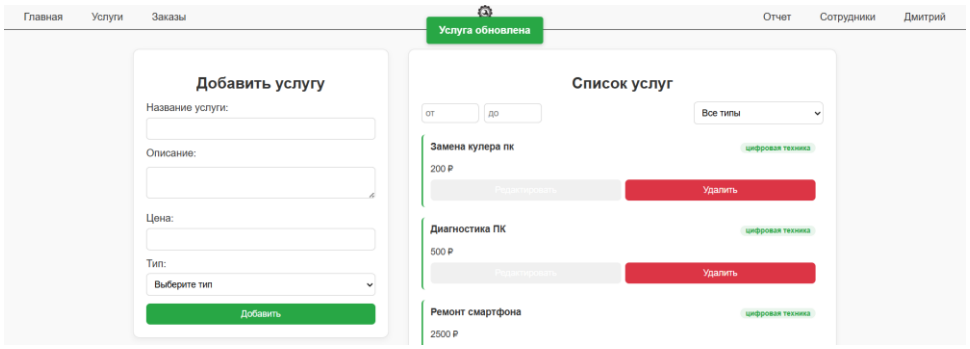


Рисунок Б19 – Результат редактирования цены услуги

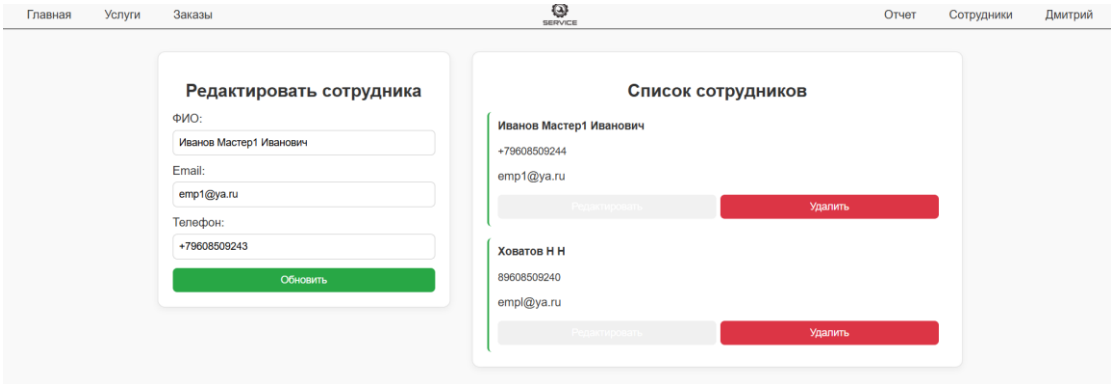


Рисунок Б20 – Редактирование номера сотрудника

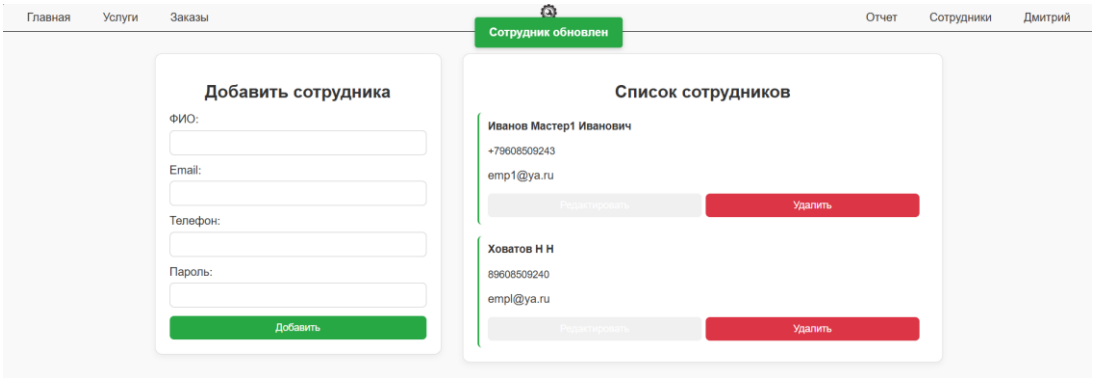


Рисунок Б21 – Результат редактирования номера сотрудника

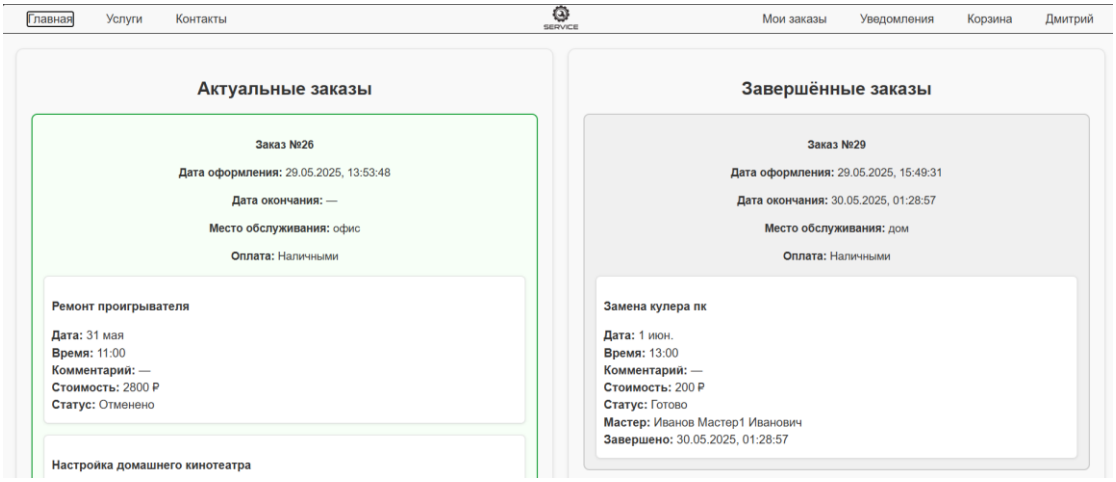


Рисунок Б22 – Список заказов у клиента

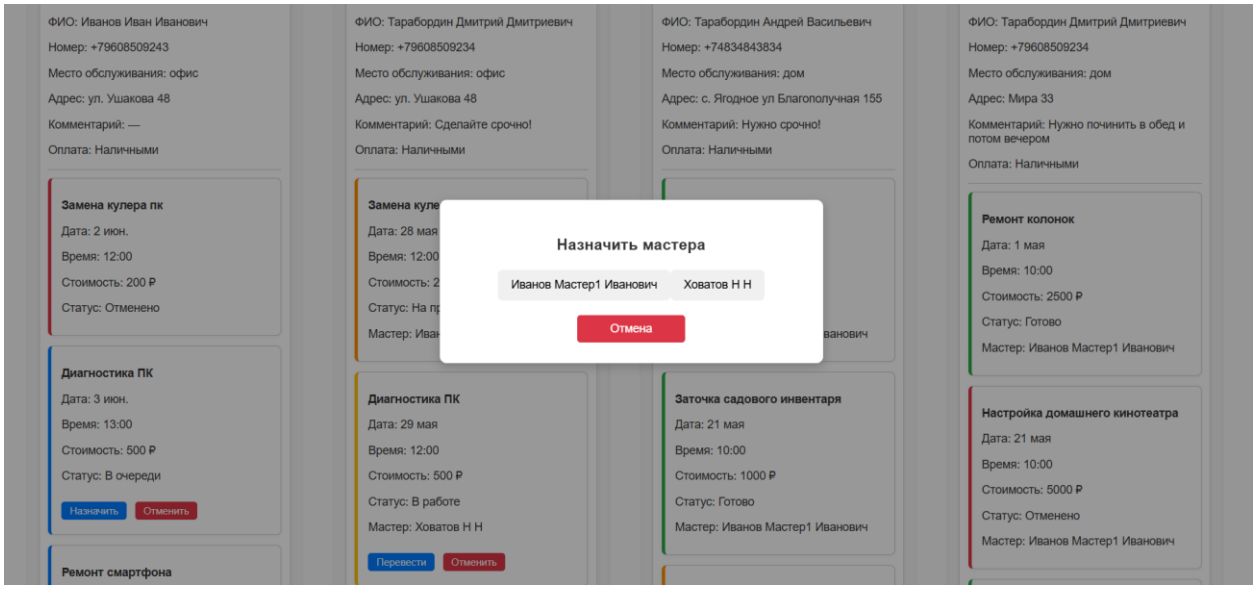


Рисунок Б23 – Назначение мастера администратором

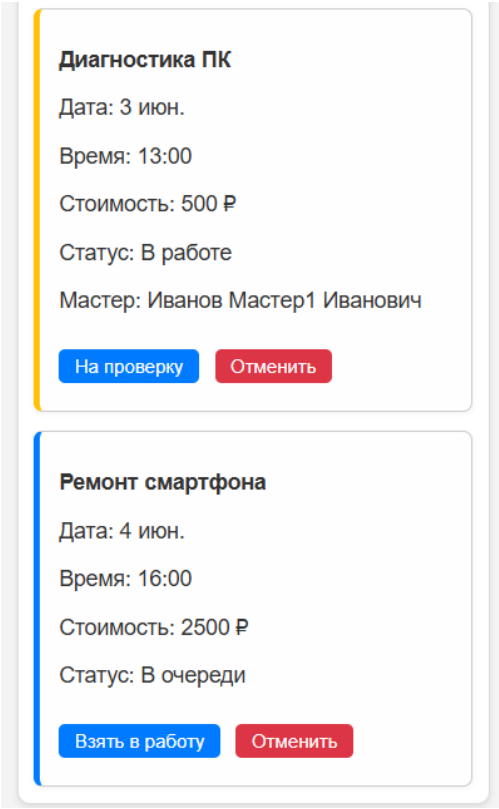


Рисунок Б24 – Самоназначение мастера

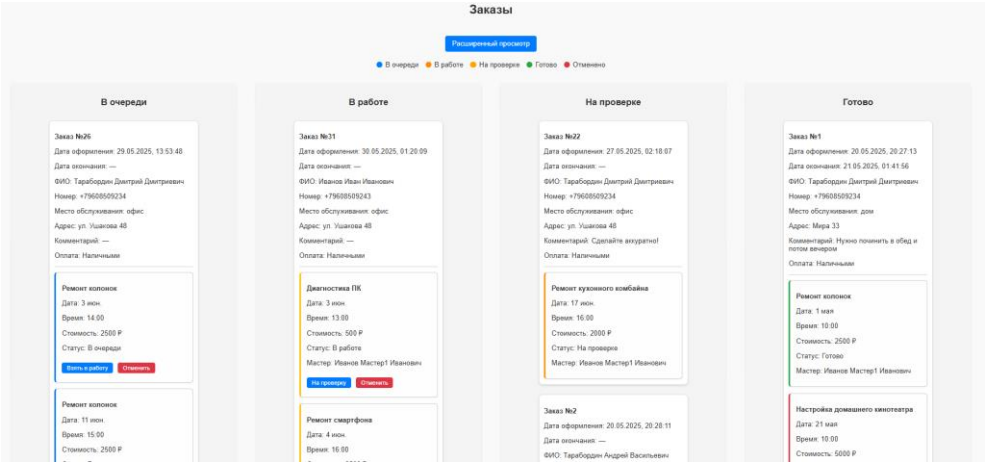


Рисунок Б25 – Канбан-доска мастера

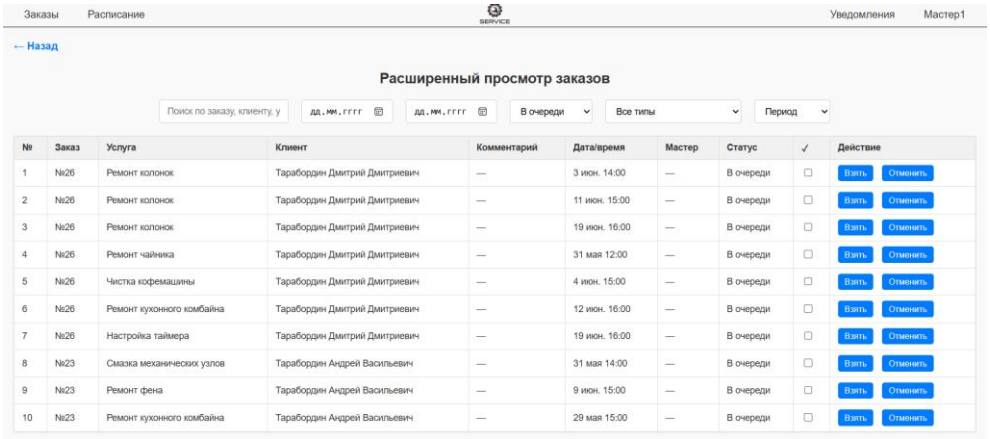


Рисунок Б26 – Расширенный просмотр заказов мастера

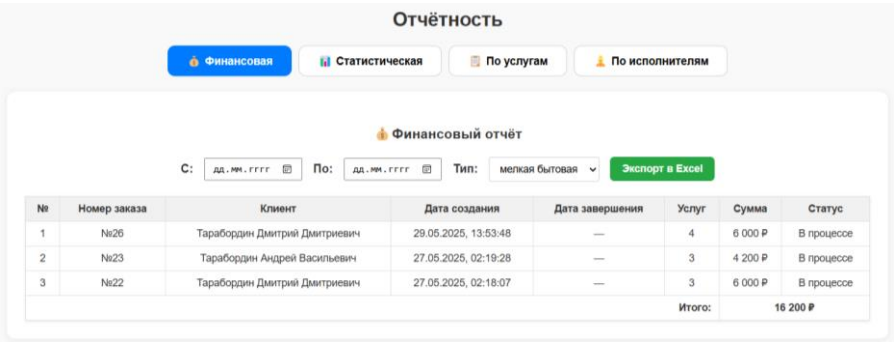


Рисунок Б27 – Финансовый отчет

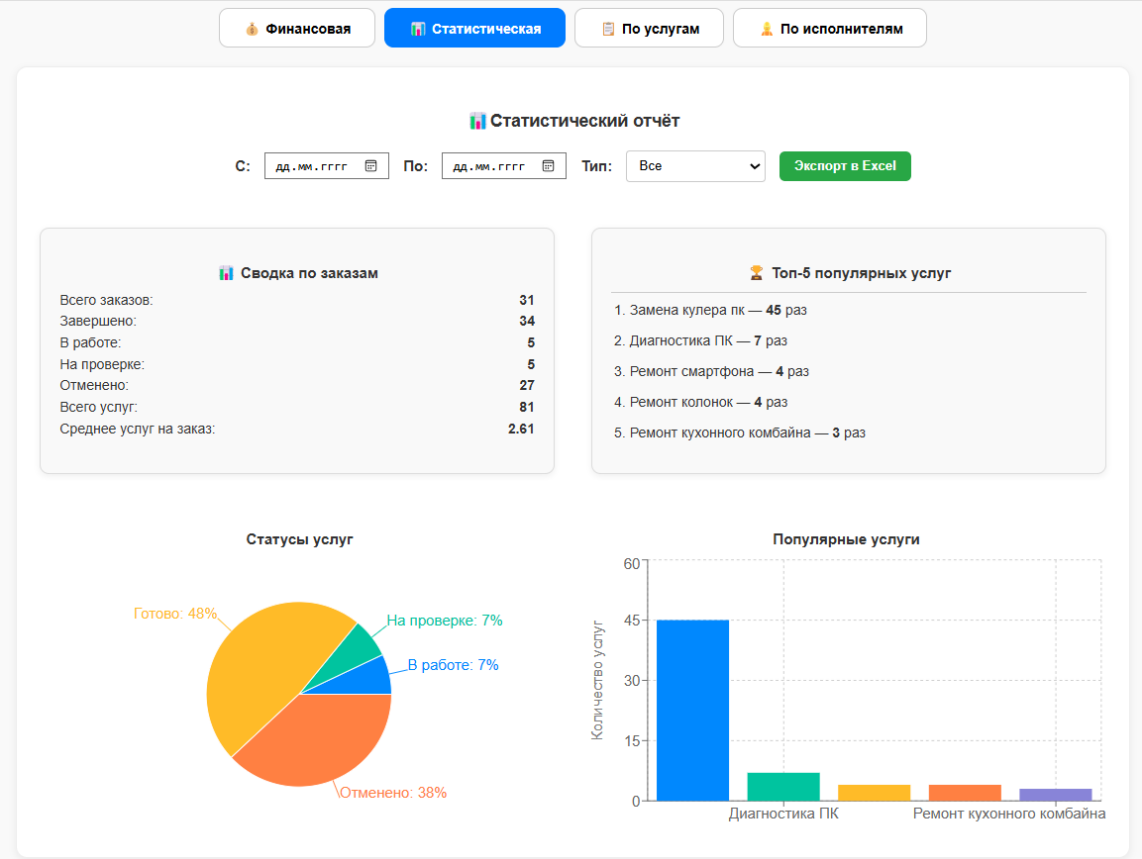


Рисунок Б28 – Статистический отчет

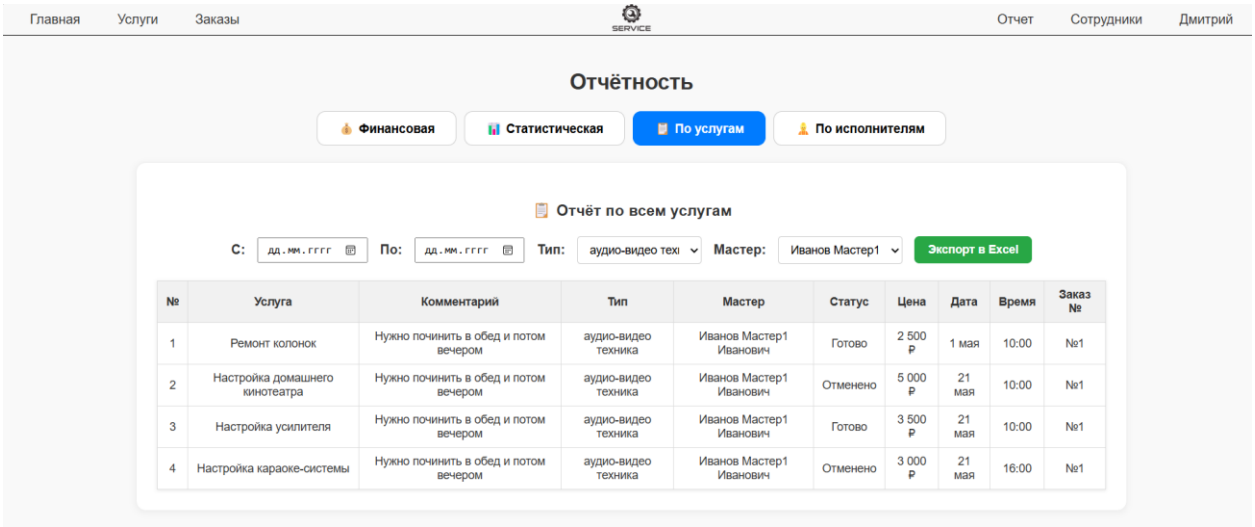


Рисунок Б29 – Отчет по услугам

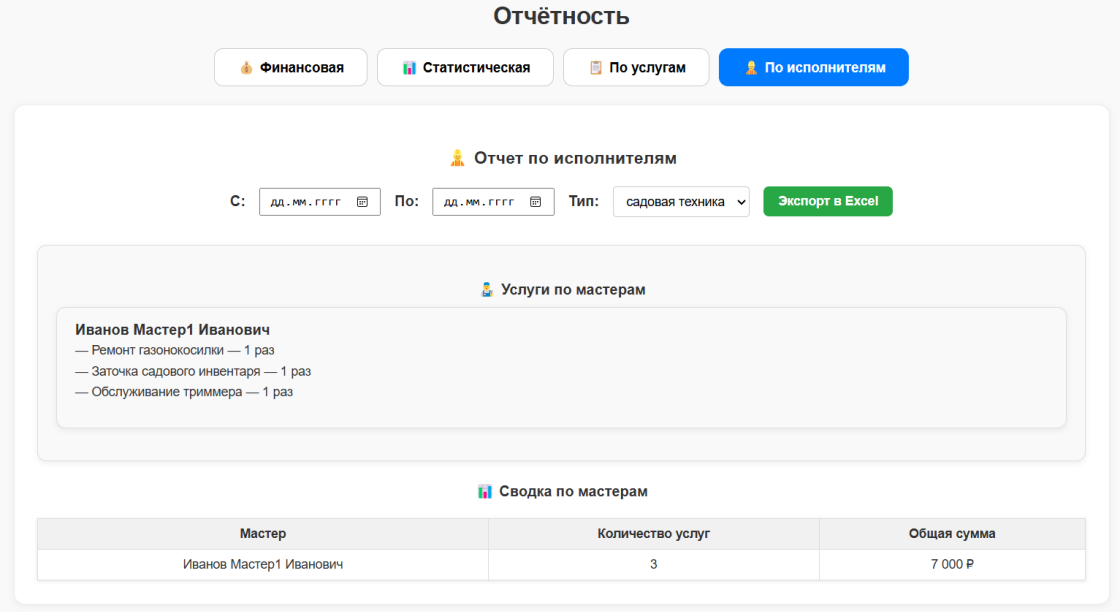


Рисунок Б30 – Отчет по исполнителям

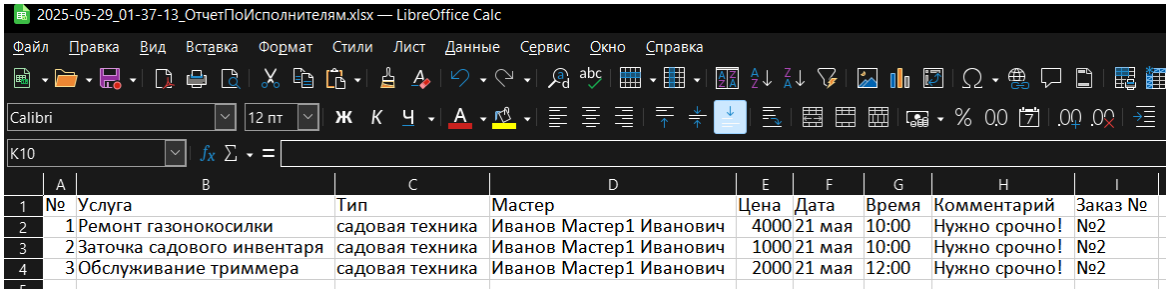


Рисунок Б31 – Экспорт отчета в Excel

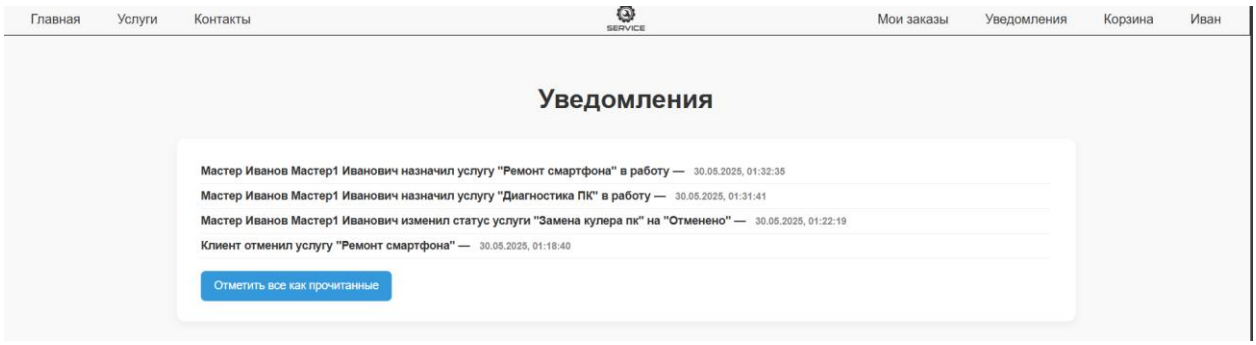


Рисунок Б32 – Раздел «Уведомления»

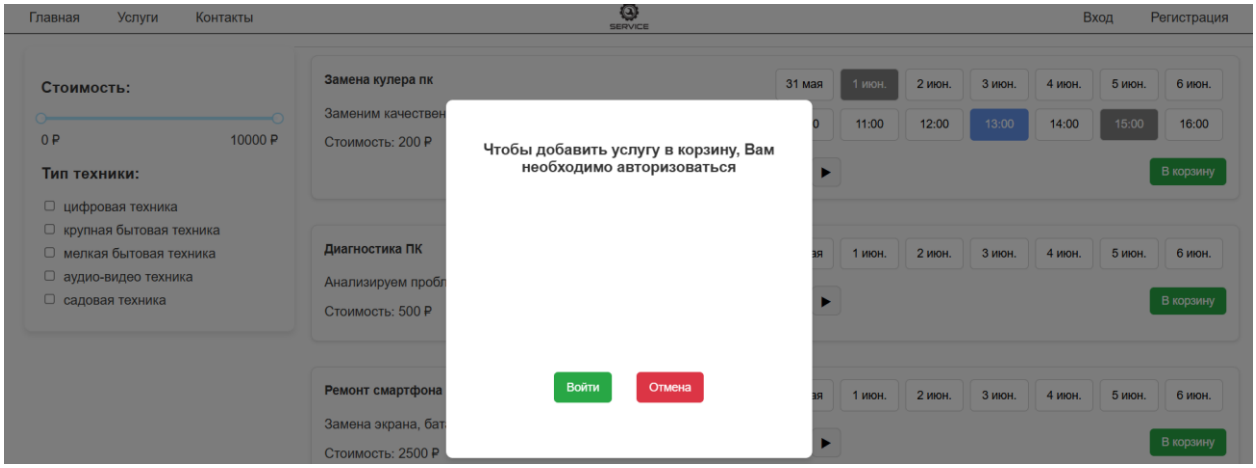


Рисунок Б33 – Ограничение неавторизованного пользователя

```
_id: ObjectId('68385de82047f9fe8f8e2a8f')
fullName : "Иванов Иван Иванович"
role : "client"
phoneNumber : "+79608509243"
email : "ivanov@yandex.ru"
password : "$2b$10$zAmNZLqYeWKTkZs5XlIdxOR5rHmWslys2QPF5bilNfQuVJHcWsIp6"
registrationDate : 2025-05-29T13:15:20.587+00:00
```

Рисунок Б34 – Хэширование пароля

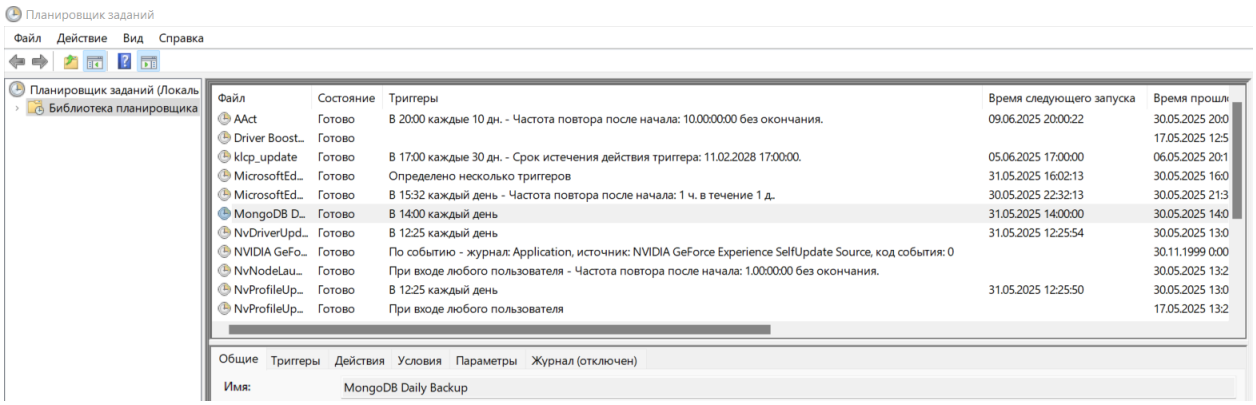


Рисунок Б35 – Резервное копирование (Планировщик заданий)

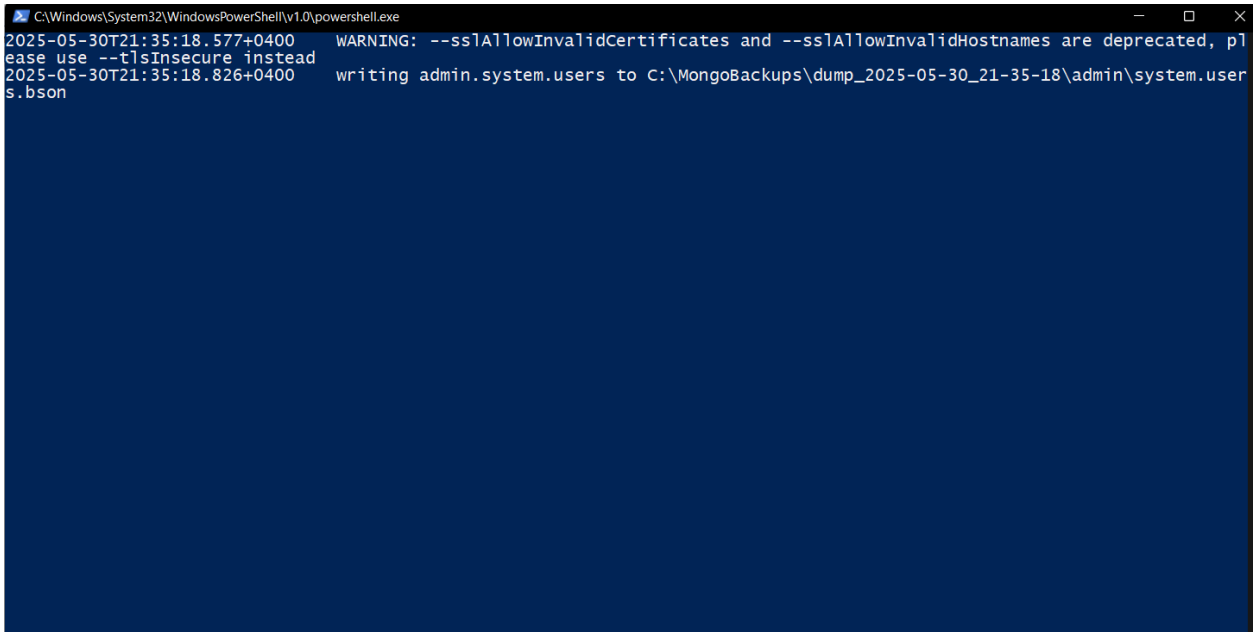


Рисунок Б36 – Резервное копирование (строка PowerShell)

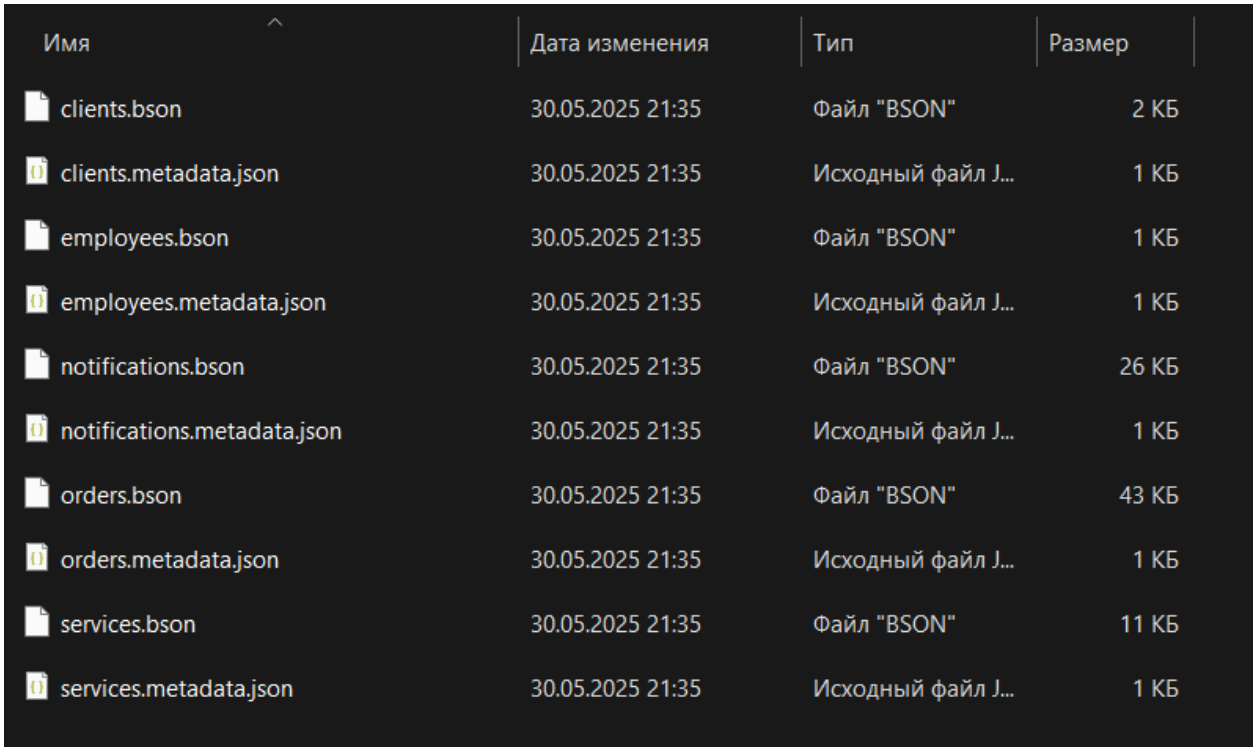


Рисунок Б37 – Резервное копирование (принудительно созданный backup)