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 социальных и экономических информационных систем
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка веб-сервиса для подачи заявок на ремонт и обслуживание бытовой техники»

Обучающийся С.Г. Сотников
(Инициалы Фамилия) (личная подпись)

Руководитель к.э.н., доцент, Т.А. Раченко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент, А.В. Егорова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

В представленной выпускной квалификационной работе дано описание разработки веб-сервиса по подаче заявок на ремонт и обслуживание бытовой техники. Данная работа является весьма актуальной на сегодняшний день, поскольку рассматривает процесс разработки программного обеспечения. Данное направление развития обрело популярность среди компаний в последние годы.

Структурно выпускная квалификационная работа представляет собой: введение, 3 глав и заключение.

Актуальность поднимаемой проблемы, цели и задачи, предмет и объект исследования полностью описаны во введении.

В первой главе проведено функциональное моделирование предметной области. Во второй главе описано логическое проектирование веб-сервиса. Спроектирована архитектура, база данных и пользовательский интерфейс веб-сервиса. В третьей главе рассмотрена разработка функциональности веб-сервиса, приведен программный код и пользовательский интерфейс, рассмотрена оценка экономической эффективности.

Основные выводи и итоги, сделанные в ходе работы над представленным веб-сервисом приведены в заключении.

Представленная выпускная квалификационная работа состоит из пояснительной записки на 121 страниц печатного текста и включает в себя введение на 3 страницы, 59 рисунков и 7 таблиц, список использованной литературы и источников из 31 источников на русском языке или в переводе.

Итогом работы стала разработка веб-сервиса по подаче заявок на ремонт и обслуживание бытовой техники. Работа завершена и внедрению не подлежит. В результате все поставленные задачи были выполнены, цель – достигнута.

Abstract

The title of the graduation work is Development of a web service for submitting applications for repair and maintenance of household appliances.

The senior paper consists of an introduction, five parts, a conclusion, figures, tables, and a list of references including foreign sources.

The key issue of the thesis is the creation of a user-friendly and efficient web service that facilitates the interaction between end-users and service centers specializing in the maintenance of household appliances. We touch upon the problem of improving the process of submitting and managing repair and maintenance requests through an intuitive online platform.

The aim of the work is to develop a web service that simplifies the submission of applications for repair and maintenance of household appliances and their management.

The graduation work may be divided into several logically connected parts which are: functional modeling of the domain; logical design of the web service including architecture, database, and user interface; development of the web service functionality with program code and user interface; evaluation of the economic efficiency of the developed web service.

Finally, we present the outcome of the project, which is a fully developed web service for submitting repair and maintenance requests for household appliances. The project has been completed and does not require further implementation.

In conclusion, we'd like to stress that this work is highly relevant today as it addresses the process of software development, which has become increasingly popular among companies in recent years. The developed web service can significantly improve the interaction between users and service centers, ensuring efficient and timely maintenance of household appliances.

Оглавление

Введение

В современном мире бытовая техника стала неотъемлемой частью нашей повседневной жизни, обеспечивая удобство и комфорт в выполнении различных задач. Однако, как и любая другая техника, она время от времени нуждается в ремонте и обслуживании. Часто пользователи сталкиваются с проблемой поиска квалифицированных специалистов и организации процесса обслуживания.

Так объектом выпускной квалификационной работы становится – сервис по ремонту и обслуживанию бытовой техники.

Сегодня, когда технический прогресс приводит к повсеместному внедрению современных технологий в повседневную жизнь, неотделимой частью которой становятся бытовые устройства, обеспечивающие комфорт и эффективность, вопрос обслуживания этой техники становится ключевым аспектом в обеспечении долгосрочной и бесперебойной работы устройств.

Так предметом выпускной квалификационной работы становится – подача заявки на ремонт или обслуживание бытовой техники.

Актуальность разработки веб-сервиса для подачи заявок на ремонт и обслуживание бытовой техники обусловлена необходимостью создания удобного и эффективного инструмента для пользователей, который обеспечит своевременное и качественное выполнение заявок на ремонт. Такой веб-сервис предоставит возможность клиентам легко и быстро подавать заявки, отслеживать статус выполнения работ и получать оперативную обратную связь.

Целью настоящей выпускной квалификационной работы является разработка веб-сервиса по подаче заявок на ремонт и обслуживание бытовой техники. Веб-сервис улучшит и упростит процессы взаимодействия между конечными пользователями и сервисными центрами, специализирующимися на обслуживании бытовой техники. Веб-сервис также предоставит

возможность клиентам эффективно и удобно подавать заявки на ремонт, отслеживать статус выполнения работ.

Для достижения этой цели перед нами стоят следующие задачи:

– Изучить предметную область.

– Изучение существующих веб-сервисов и приложений для подачи заявок на ремонт бытовой техники с целью выявления основных проблем и недостатков.

– Определение функциональных и не функциональных требований к разрабатываемому веб-сервису.

– Проектирование архитектуры и интерфейса веб-сервиса с учетом современных тенденций пользовательского опыта и удобства использования.

– Реализация программного кода веб-сервиса с использованием современных технологий веб-разработки.

– Тестирование и отладка веб-сервиса для обеспечения стабильной и безопасной работы.

– Оценка эффективности разработанного веб-сервиса.

Для выполнения поставленных задач будет использоваться CASE-технологии структурного и объектно-ориентированного анализа и проектирования.

Для выполнения поставленных задач была сформирована структура работы:

– Функциональное моделирование бизнес-процесса.

На данном этапе будет проведено исследование предметной области, выявлен необходимый функционал системы, выявлены недостатки текущего состояния в предметной области.

– Логическое проектирование веб-сервиса.

На основе полученной информации, будут выбраны средства проектирования, проведено проектирование базы данных, интерфейса, концептуальное моделирование.

– Разработка функциональности веб-сервиса.

Будет выбрана архитектура веб-сервиса, выбраны средства разработки, разработана база данных согласно проектированию, разработан функционал веб-сервиса и интерфейс пользователя.

– Тестирование и отладка.

Проведено тестирование всех разработанных компонентов и функций на полное соответствие требованиям, при необходимости проводить отладку для приведения к необходимым требованиям.

– Оценка экономической эффективности.

Будет проведена экономическая эффективность разработанного веб-сервиса. Определены траты на разработку, экономические причины выбора веб-сервиса в сравнении с аналогами.

Будут рассмотрены основные этапы разработки веб-сервиса, его ключевые характеристики и преимущества, а также потенциал для дальнейшего совершенствования и расширения функционала в соответствии с требованиями рынка и потребностями пользователей.

Глава 1 Функциональное моделирование бизнес-процесса «подача заявок на ремонт и обслуживание бытовой техники»

1.1 Описание предметной области

Организация по ремонту и обслуживанию бытовой техники создана с целью предоставления высококачественных услуг по ремонту и техническому обслуживанию бытовых приборов. Основная миссия предприятия заключается в обеспечении надёжной и быстрой помощи владельцам бытовой техники, столкнувшимся с поломками и неисправностями. Для достижения этой цели предприятие ставит перед собой несколько задач: обеспечение высокого уровня сервиса, своевременное выполнение заявок, поддержка клиентов, развитие и обучение персонала, а также оптимизация процессов для повышения экономической эффективности.

Быт представляет собой повседневный уклад жизни человека, в котором удовлетворяются его основные физиологические потребности.

Бытовая техника – это электрические или механические устройства, предназначенные для использования в домашних условиях. В основном, эта техника предназначена для помощи в выполнении повседневных задач, таких как уборка и приготовление пищи, поскольку эти процессы обычно требуют значительных затрат времени.

Ремонт техники – это комплекс мероприятий, направленных на восстановление работоспособного или исправного состояния оборудования, а также на продление его срока службы. Заявка на ремонт оборудования – это официальная бумага, которая, во-первых, фактически подтверждает, что используемое предприятием устройство вышло из строя, а во-вторых, утверждает необходимость его починки.

Обеспечение высокого уровня сервиса предполагает предоставление клиентам услуг, соответствующих их ожиданиям и потребностям.

Своевременное выполнение заявок требует минимизации сроков ремонта и профилактического обслуживания, что важно для удовлетворения потребностей клиентов и поддержания их доверия. Поддержка клиентов осуществляется через консультации и рекомендации по эксплуатации и уходу за бытовой техникой. Постоянное развитие и обучение персонала направлено на повышение квалификации сотрудников, чтобы они могли справляться с новыми и сложными задачами. Оптимизация процессов способствует снижению затрат и повышению прибыльности компании.

Организационная структура предприятия по ремонту и обслуживанию бытовой техники включает несколько ключевых подразделений, каждое из которых играет важную роль в обеспечении эффективного функционирования компании. Важным элементом структуры является система обработки заявок клиентов и их выполнения.

Верховным руководителем предприятия является директор, который несёт ответственность за общее руководство и управление компанией, стратегическое планирование и принятие ключевых решений. Директор осуществляет руководство всеми подразделениями компании, утверждает бюджет и принимает решения по развитию бизнеса.

Административный отдел координирует административные функции и управляет офисными процессами. Руководитель административного отдела организует работу офиса и взаимодействует с другими отделами. Секретарь или офис-менеджер ведёт делопроизводство, принимает звонки, организует встречи и обрабатывает заявки от клиентов, передавая их в отдел сервиса и ремонта для дальнейшего выполнения.

Отдел сервиса и ремонта играет центральную роль в деятельности предприятия. Руководитель отдела сервиса координирует работу мастеров и контролирует качество выполнения ремонтных работ. После получения заявки от административного отдела, руководитель отдела сервиса распределяет задания между мастерами по ремонту. Мастера выполняют диагностику и ремонт бытовой техники, взаимодействуя с клиентами по

вопросам ремонта. Стажёры и помощники мастеров помогают в выполнении простых ремонтных задач под руководством опытных сотрудников.

Отдел закупок и снабжения обеспечивает предприятие необходимыми запчастями и материалами. Менеджер по закупкам ведёт переговоры с поставщиками, заказывает и учитывает запчасти, необходимые для ремонта техники. При необходимости мастера по ремонту подают запросы на необходимые детали в отдел закупок.

Отдел маркетинга и продаж занимается разработкой и реализацией маркетинговой стратегии, а также продвижением услуг компании. Руководитель отдела маркетинга управляет рекламными кампаниями и анализирует рынок и конкурентов. Менеджеры по продажам работают с клиентами, продают услуги компании и ведут клиентскую базу.

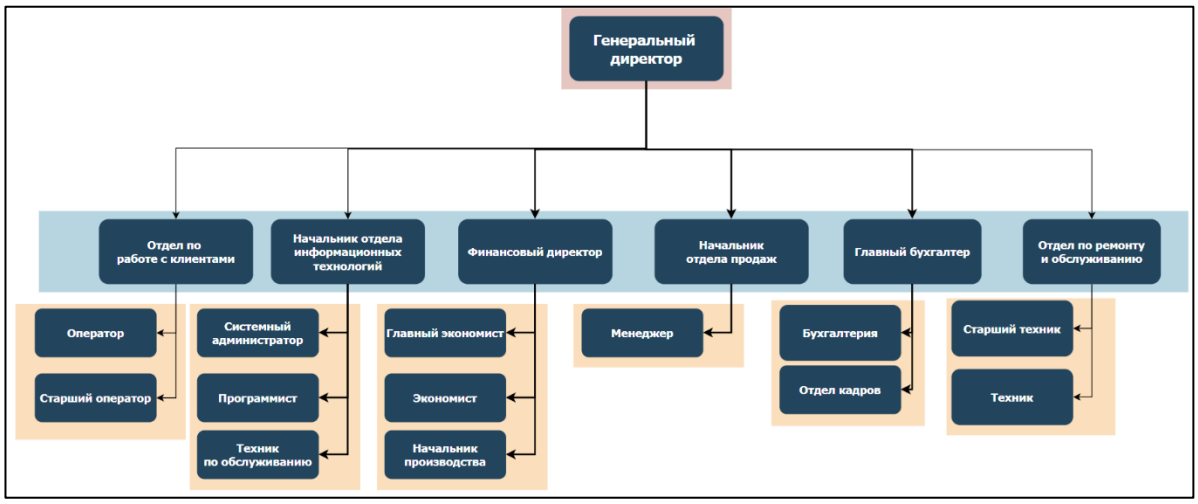

Рисунок 1 - Структурная схема предприятия

Взаимоотношения между работниками предприятия строятся на чётком распределении ответственности и полномочий, что позволяет эффективно организовать работу и обеспечивать высокий уровень сервиса. Директор координирует работу всех отделов, устанавливает стратегические цели и контролирует их выполнение. Руководители отделов несут ответственность за выполнение конкретных задач и координацию своих подразделений.

Процесс обработки заявок начинается с их приёма административным отделом, который затем передаёт заявки в отдел сервиса и ремонта. Руководитель отдела сервиса распределяет задания между мастерами, которые выполняют ремонт и обслуживание техники. Мастера могут взаимодействовать с отделом закупок для получения необходимых запчастей. Отдел маркетинга и продаж работает над привлечением новых клиентов и удержанием существующих, обеспечивая поток заявок.

Таким образом, структура и взаимоотношения в организации по ремонту и обслуживанию бытовой техники направлены на достижение основной цели – предоставление клиентам качественных и надёжных услуг по ремонту и обслуживанию бытовой техники. Это достигается через слаженную работу всех подразделений и чёткое распределение обязанностей между сотрудниками.

В следствие, целью предприятия является оказание услуг по ремонту и по обслуживанию. Результатом деятельности предприятия становится заполненная, оформленная, выполненная мастером и оплаченная клиентом заявка на услугу. Предприятие предоставляет услуги по ремонту и обслуживания бытовой техники. Ресурсами выступают необходимые компоненты техники, а также различные необходимые для ремонта материалы.

1.2 Концептуальное моделирование предметной области

Одним из важнейших этапов проектирования автоматизированной системы остается моделирование предметной области. Существует несколько нотаций концептуального моделирования. Чтобы совершить выбор технологии моделирования, проведем сравнительный анализ наиболее популярных средств создания моделей ИС: UML, IDEF0, ARIS, BPMN.

UML – это язык моделирования, который используется для спецификации, визуализации, разработки и документирования компонентов

программных систем. UML включает несколько типов диаграмм (например, диаграммы классов, последовательностей, состояний и действий), которые могут быть полезны для моделирования различных аспектов систем. Однако UML больше ориентирован на разработку программного обеспечения и может быть менее эффективным для чисто бизнес-процессов по сравнению с IDEF0.

IDEF0 (Integration Definition for Function Modeling) – технология моделирования предметной области, основанная на функциональном подходе. В нотации IDEF0 отражаются структура, функции системы, и связывающие эти функции потоки информации. Контекстная диаграмма в нотации IDEF0 представляется блоком в форме прямоугольника с указанием входов и выходов моделируемого бизнес-процесса, а также используемых ресурсов и регламентирующих документов

Данная нотация наиболее легка в изучении, также удобна в использовании и не ограничена в уровнях декомпозиции. Отличием от UML является разница в подходе к моделированию. ARIS (Architecture of Integrated Information Systems) – технология моделирования бизнес-процессов организаций, основанная на процессном подходе к моделированию. Модели в данной нотации хранятся в 12 объектной базе данных. Эта технология моделирования более сложна в изучении и использовании, чем нотации, описанные выше.

BPMN – это стандарт для моделирования бизнес-процессов, который фокусируется на визуализации бизнес-процессов в виде диаграмм. Он используется для описания процессов в виде последовательности действий и событий, что облегчает понимание и анализ. BPMN подходит для моделирования сложных процессов с множеством вариантов развития, однако его основной фокус на диаграммах событий и потоков может быть менее детализированным в описании функций и потоков данных по сравнению с IDEF0.

IDEF0 методология является оптимальным выбором для проектирования процессов подачи заявок на ремонт и обслуживание бытовой техники благодаря своей способности детализировать и структурировать функциональные аспекты системы. Этот подход позволяет чётко определить функции и процессы, что упрощает управление и оптимизацию деятельности. Модели IDEF0 строятся в виде иерархической структуры, где каждый уровень детализации раскрывает отдельные аспекты процесса, начиная с общего обзора и углубляясь в детали.

Кроме того, IDEF0 визуализирует взаимодействие между различными процессами и функциями, а также потоки данных и материалов. Это помогает лучше понять взаимосвязи и взаимодействия элементов системы, что критично для повышения эффективности работы. Методология также способствует эффективному распределению ресурсов и оптимизации потоков данных, чётко определяя входы, выходы, управление и механизмы, необходимые для выполнения каждой функции.

Использование IDEF0 позволяет выявить возможные проблемы и узкие места в процессах подачи заявок и выполнения ремонтных работ, что даёт возможность проактивно их устранять. Стандартизация бизнес-процессов с помощью IDEF0 обеспечивает единообразие в выполнении работ и обслуживании клиентов, повышая качество сервиса и удовлетворенность клиентов. Гибкость и адаптивность моделей IDEF0 позволяют легко вносить изменения в процессы и адаптировать их к новым условиям и требованиям рынка.

Функциональные модели IDEF0 предоставляют управленческому персоналу чёткое понимание текущего состояния процессов, что облегчает принятие обоснованных решений по их улучшению и развитию бизнеса. Также они служат хорошей документацией и могут использоваться для обучения новых сотрудников или повышения квалификации существующих.

Таким образом, выбор IDEF0 для концептуального проектирования процессов подачи заявок на ремонт и обслуживание бытовой техники

обоснован его способностью структурировано описывать и визуализировать процессы, эффективно управлять ресурсами и потоками данных, выявлять проблемы и стандартизировать деятельность, что в конечном итоге способствует повышению эффективности и качества обслуживания клиентов.

Процесс начинается с того, что клиент, имеющий неисправную бытовую технику, прибывает в офис предприятия. В офисе клиент обращается к сотрудникам, чтобы подать заявку на ремонт или обслуживание своей техники. Сотрудник офиса принимает технику и фиксирует все необходимые данные, включая тип техники, её модель, описание проблемы и контактную информацию клиента. На рисунке 2 представлена контекстная диаграмма, а рисунке 3 представлена диаграмма декомпозиции.

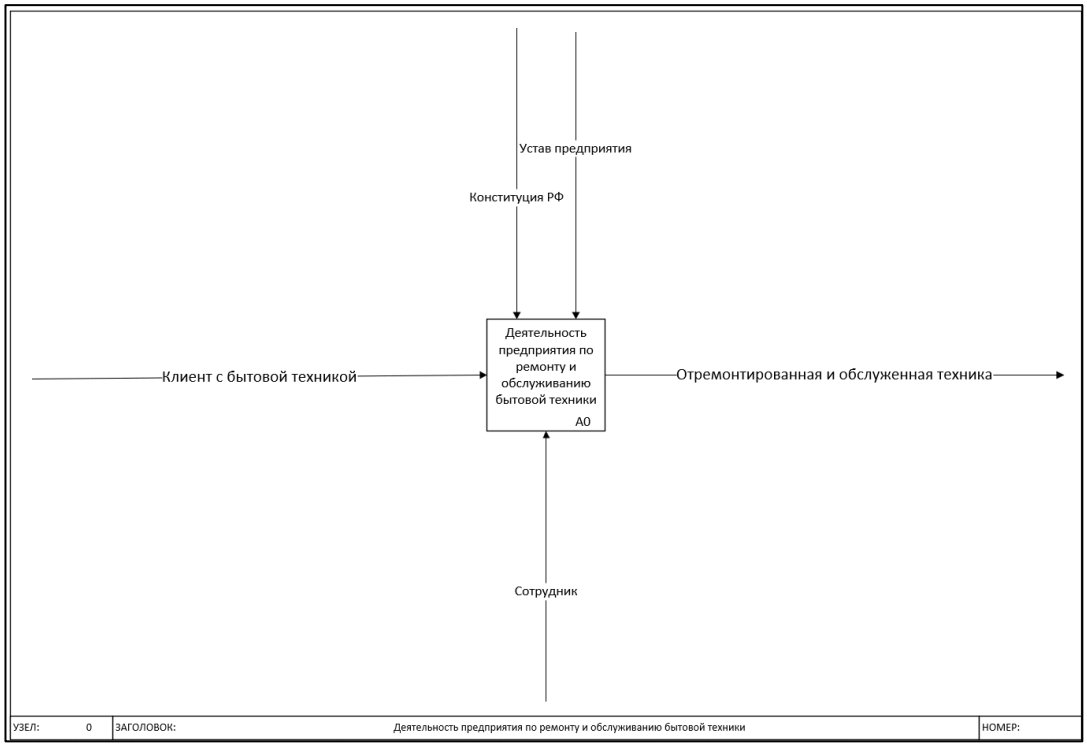

Рисунок 2 – Контекстная диаграмма

После этого заявка поступает в систему, где её регистрируют и присваивают уникальный номер для отслеживания. Далее информация передается в отдел диагностики, где техника проходит первичное обследование. Специалист диагностирует проблему, определяет причину неисправности и оценивает объем необходимых работ. По завершении диагностики специалист сообщает клиенту результаты, а также предоставляет оценку стоимости и сроков ремонта.

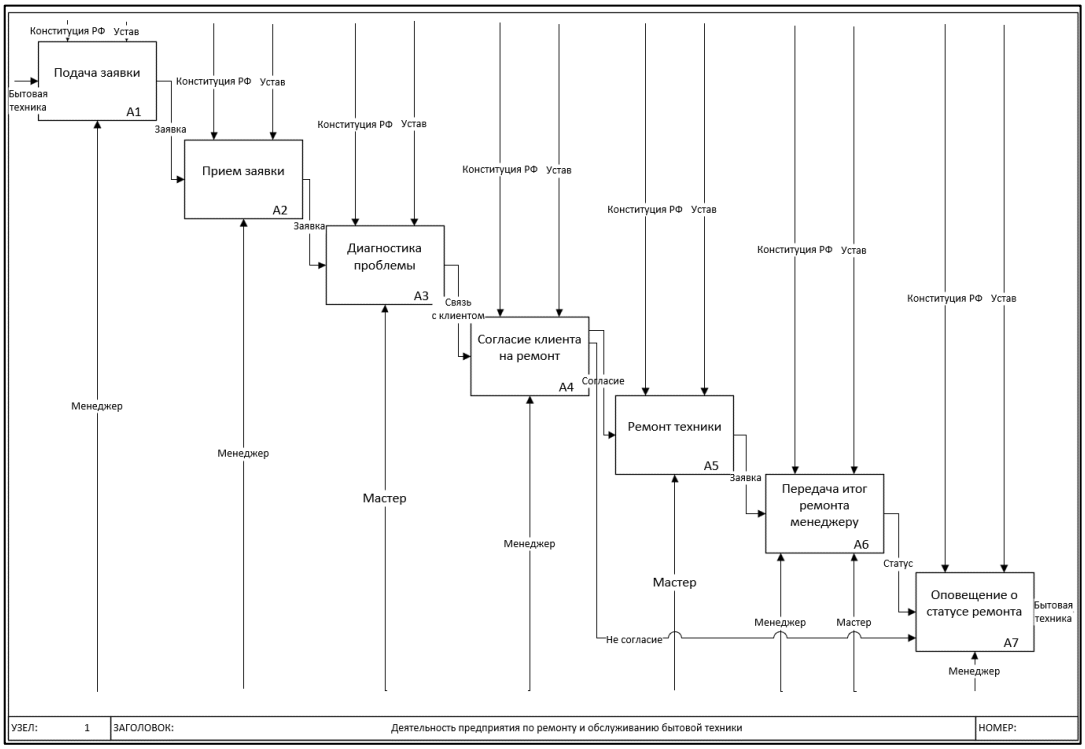

Рисунок 3 – Диаграмма декомпозиции

Если клиент соглашается с предложенными условиями, он подтверждает начало ремонта. В случае несогласия, клиент может обсудить детали или отказаться от услуг, что фиксируется в системе. После получения согласия клиента, техника передается в ремонтный отдел.

В ремонтном отделе техника проходит все необходимые процедуры по устранению неисправностей. Сотрудники используют требуемые запчасти и материалы, а также соблюдают все стандарты и нормы, указанные в

регламенте предприятия. По завершении ремонтных работ техника проходит контроль качества, чтобы убедиться в устранении всех неисправностей.

Когда ремонт завершен, клиенту направляется уведомление о готовности техники. Он может получить это уведомление по телефону, через электронную почту или СМС. После получения уведомления клиент приходит в офис, где ему передают отремонтированную технику. Сотрудник проверяет соответствие всех выполненных работ с заявкой и получает подтверждение клиента о выполнении ремонта.

Завершающим этапом является закрытие заявки. Сотрудник фиксирует в системе, что ремонт выполнен и техника возвращена клиенту. Все данные по заявке архивируются для дальнейшего анализа и отчетности. Клиент, в свою очередь, может оставить отзыв о качестве предоставленных услуг и уровне сервиса.

Таким образом, процесс подачи заявки на ремонт и обслуживание бытовой техники включает взаимодействие между клиентом и сотрудниками предприятия, начиная от подачи заявки до завершения ремонта и закрытия заявки, с обязательным соблюдением всех регламентов и стандартов.

В настоящее время процесс обработки заявок и ремонта техники осуществляется вручную. Клиенты имеют возможность обратиться в сервисный центр лично или подать заявки на бумаге. Однако это часто сопровождается задержками и ошибками. Это приводит к неэффективной работе, задержкам в обработке заявок и увеличению времени на выполнение ремонта. На рисунке 4 представлена контекстная диаграмма работы организации «как есть» в аннотации IDEF0.

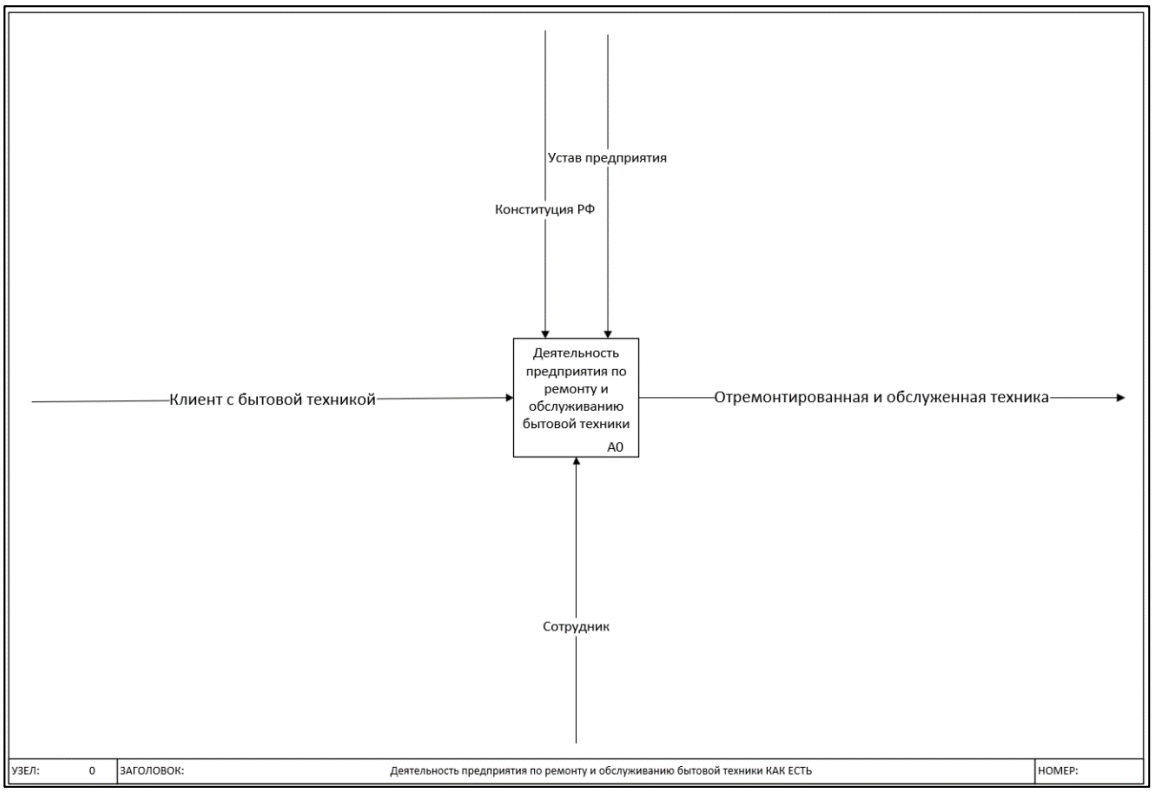

Рисунок 4 – Контекстная диаграмма «как есть»

В центре диаграммы находится блок, обозначенный как "Деятельность предприятия по ремонту и обслуживанию бытовой техники".

Слева к этому блоку поступает входной элемент, представляющий клиента с бытовой техникой, нуждающейся в ремонте или обслуживании. В результате обработки заявки клиент получает отремонтированную и обслуженную технику, что указано на выходе блока, направленном вправо.

Снизу к центральному блоку подключены ресурсы или механизмы, используемые для выполнения процесса, в данном случае это сотрудник, который выполняет ремонтные работы.

Сверху центрального блока находятся элементы управления, которые включают Конституцию РФ и Устав предприятия. Эти документы регулируют правовые и внутренние рамки деятельности предприятия.

Диаграмма представляет собой упрощённое описание процесса, в котором клиент подаёт заявку, сотрудник выполняет ремонт, а нормативные документы обеспечивают соблюдение правил и стандартов.

Основные причины необходимости изменений заключаются в том, что ручная обработка заявок требует дополнительного времени и может привести к потере информации или ошибкам. Это снижает производительность и эффективность работы предприятия. Отсутствие централизованной системы управления данными затрудняет доступ к информации о статусе ремонта, а ручной процесс увеличивает вероятность ошибок и недочетов в данных. На рисунке 5 представлена диаграмма декомпозиции.

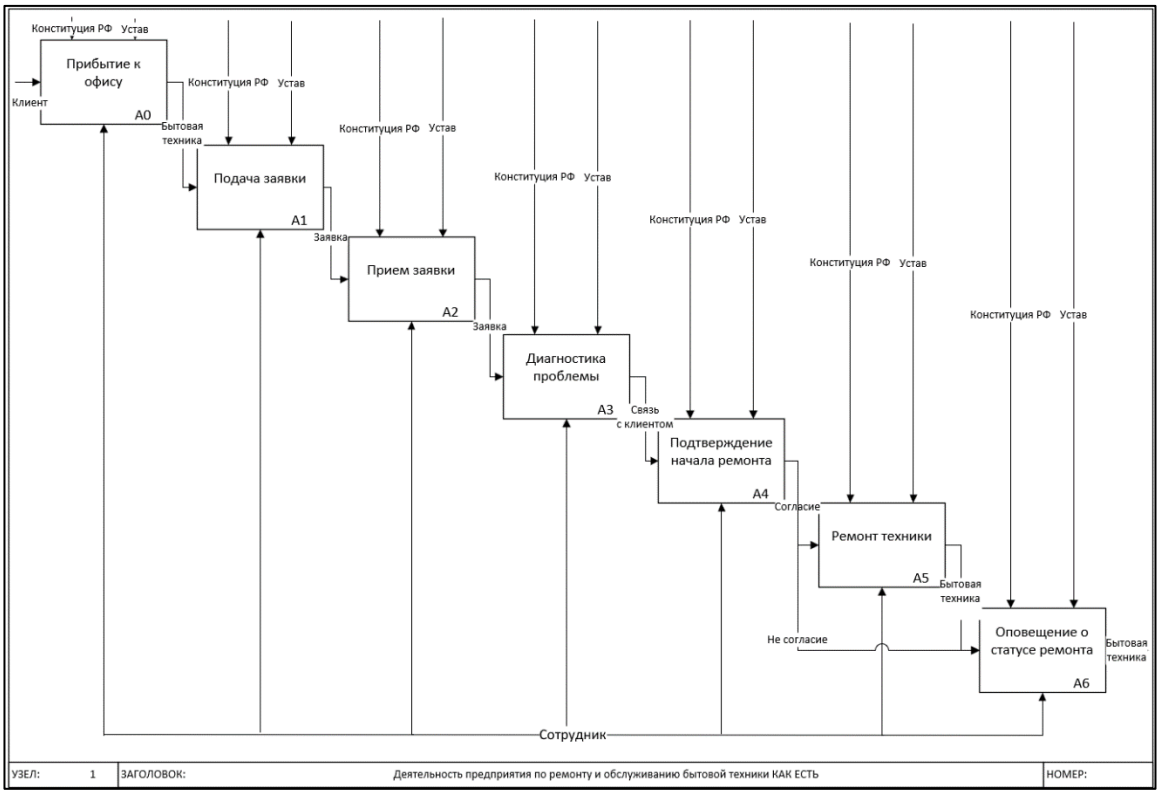

Рисунок 5 – Диаграмма декомпозиции «как есть»

На изображении представлена развернутая диаграмма IDEF0, подробно описывающая процесс деятельности предприятия по ремонту и обслуживанию бытовой техники. Диаграмма состоит из нескольких

взаимосвязанных блоков, отражающих этапы процесса от прибытия клиента до завершения ремонта и уведомления о его статусе.

– Прибытие к офису (A0).

Процесс начинается с прибытия клиента в офис предприятия с бытовой техникой, нуждающейся в ремонте. Этот этап обозначен блоком "Прибытие к офису" и регламентируется Конституцией РФ и Уставом предприятия.

– Подача заявки (A1).

Клиент подает заявку на ремонт, предоставляя информацию о неисправной технике. Этот процесс фиксируется в блоке "Подача заявки", куда поступает бытовая техника. Подача заявки также регулируется Конституцией РФ и Уставом.

– Прием заявки (A2).

После подачи заявки информация поступает в блок "Прием заявки", где заявка регистрируется в системе. Здесь фиксируются все данные, необходимые для дальнейшего процесса ремонта. Этот этап также регулируется Конституцией РФ и Уставом предприятия.

– Диагностика проблемы (A3).

На этом этапе заявка поступает в блок "Диагностика проблемы", где сотрудник предприятия проводит диагностику неисправной техники. Результаты диагностики передаются клиенту для согласования дальнейших действий. Связь с клиентом осуществляется для уточнения деталей и получения согласия на ремонт. Этот процесс регулируется Конституцией РФ и Уставом.

– Подтверждение начала ремонта (A4).

После диагностики клиент подтверждает начало ремонта, что фиксируется в блоке "Подтверждение начала ремонта". Если клиент соглашается с условиями ремонта, процесс продолжается. В случае несогласия процесс возвращается к обсуждению с клиентом. Этот этап также регулируется Конституцией РФ и Уставом.

– Ремонт техники (A5).

Здесь происходит непосредственно ремонт бытовой техники, который выполняется сотрудниками предприятия. В блоке "Ремонт техники" выполняются все необходимые работы по устранению неисправностей. По завершении ремонта техника готова к возврату клиенту. Ремонт техники регулируется Конституцией РФ и Уставом.

– Оповещение о статусе ремонта (A6).

После завершения ремонта клиент уведомляется о статусе ремонта через блок "Оповещение о статусе ремонта". Клиент получает информацию о том, что его техника отремонтирована и готова к выдаче. Этот процесс также регулируется Конституцией РФ и Уставом.

Каждый блок в диаграмме сопровождается входами, выходами, механизмами и управлениями. Входы включают заявки клиентов и данные о технике, выходы – отремонтированная техника и уведомления для клиентов. Механизмы представлены сотрудниками, выполняющими различные операции, а управление осуществляется на основе Конституции РФ и Устава предприятия.

Диаграмма демонстрирует последовательность действий, необходимых для обеспечения качественного ремонта и обслуживания бытовой техники, а также взаимодействие между клиентами, сотрудниками и нормативными документами, регулирующими процесс.

Из этого следует, что недостатками текущего положения является отсутствие истории заявок, отсутствие у клиента возможности узнать прогресс заявки, усложненный процесс подачи заявок, ручное заполнение шаблона заявки.

1.3 Анализ существующих разработок на предмет соответствия сформулированным требованиям

Для оценки рациональности разработки нового программного продукта является обязательным осуществить анализ уже имеющихся разработок в этой сфере. Самым оптимальным решением будет сделать выбор, основанный на соответствии системы предъявляемым требованиям:

- стоимость;
- удобный интерфейс;
- доступность, возможность зайти с любого устройства;
- отсутствие ограничения по числу вносимых данных;
- уведомление клиентов и мастеров;
- ограничение прав доступа;
- поиск и фильтрация данных;
- история заявок;
- печать шаблона заявок.

Перед разработкой веб-сервиса необходимо изучить рынок уже имеющихся решений обозначенной задачи.

Для анализа будут выбраны наиболее популярные и используемые на рынке решения, которые зарекомендовали себя среди пользователей и имеют положительные отзывы. Рассматриваемые системы будут оцениваться по выявленным ранее требованиям. Такой подход позволит объективно оценить существующие разработки и определить области для улучшения при разработке собственного веб-сервиса.

Для решения задачи было выявлено несколько средств:

- LiveSklad - это CRM-система, разработанная для автоматизации бизнес-процессов. Она ускоряет оформление заказов, предотвращает их потерю и контролирует сроки исполнения. Система позволяет гибко взаимодействовать с клиентами и сохранять историю работы с заказами.

LiveSklad позиционируется как облачная система для малого бизнеса, обеспечивающая хранение данных клиентов и организаций на сервере, что позволяет получить доступ к информации из любого места. Внешний вид продукта представлен на рисунке 6.

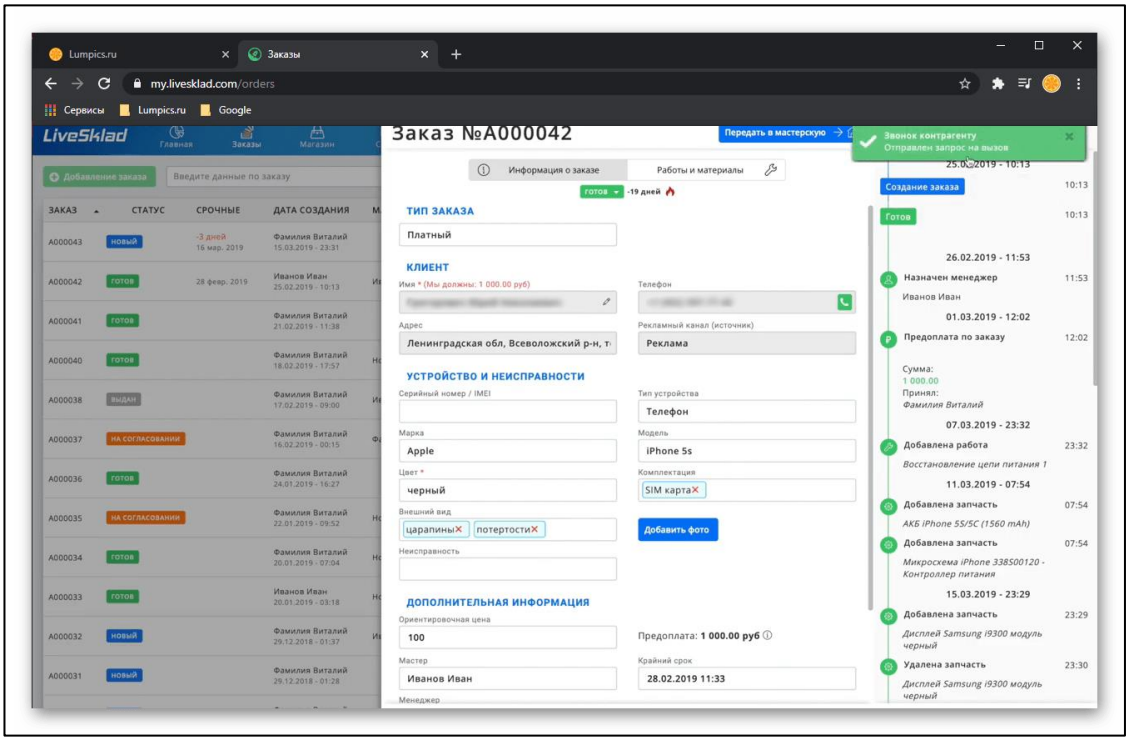

Рисунок 6 - Внешний вид LiveSklad

– GinCore - это программа для комплексного учета в магазине, сервисном центре или на складе, которая контролирует выполнение задач сотрудниками и взаиморасчеты с контрагентами.

GinCore работает в браузере и не требует установки на рабочий компьютер, что позволяет управлять деятельностью предприятия из любого места с помощью ноутбука или планшета. Система предоставляет возможность удаленно настраивать права доступа для сотрудников и отслеживать ключевые бизнес-показатели компании из любой точки мира. Внешний вид интерфейса показан на рисунке 7.

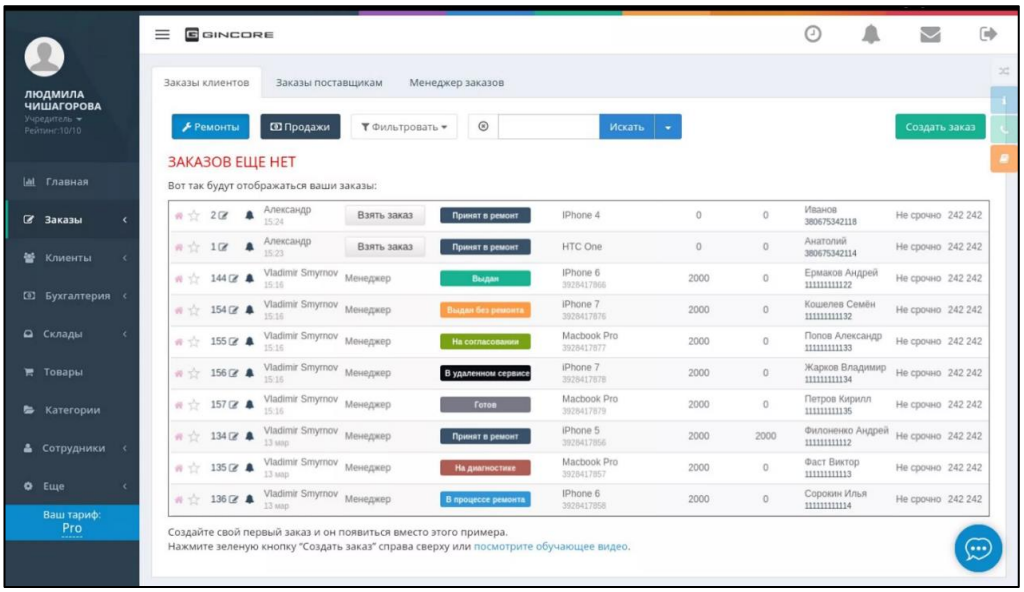

Рисунок 7 - Внешний вид GinCore

– HelloClient - онлайн-сервис для управления заказами, складом, расчёта зарплаты, финансами. Программа для автоматизации и продвижения ремонтных сервисных центров и бизнесов услуг. Автоматизирует учёт заказов, склад, подсчёт заплаты, розничные продажи.

HelloClient - это облачный сервис для автоматизации и продвижения бизнесов услуг. В программе вы сможете автоматизировать основные бизнес-процессы:

– учёт заказов;
– складское хранение;
– розничные продажи;
– заработная плата;
– финансовый учёт;
– оповещение клиентов на всех этапах;
– продвижение компании на картах Яндекс, 2ГИС, Google.

HelloClient подходит для управления одним сервисом, или сетью. Так как является облачным сервисом, то поддерживается на любой операционной системе, позволяющей использовать браузер, а значит HelloClient можно

использовать как на MacOS, Windows, Linux, так и на IOS, Android. Внешний вид представлена на рисунке 8.

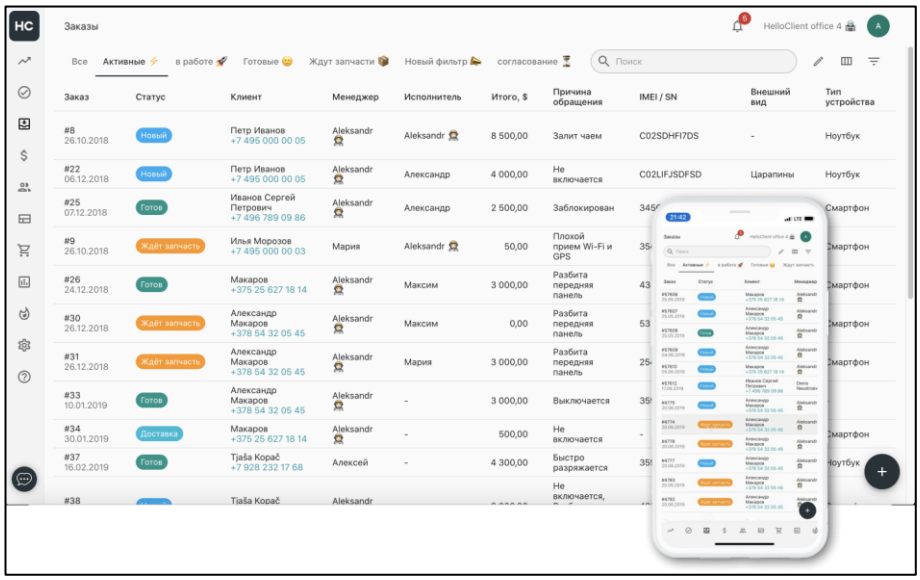

Рисунок 8 - Внешний вид HelloClient

Кому подходит:

– сервисные центры по ремонту цифровой техники;
– сервисные центры по ремонту бытовой техники и оборудования;
– ателье по ремонту и пошиву одежды;
– мастерские по ремонту и химчистке обуви;
– мастерские по ремонту велосипедов и спортивного инвентаря;
– другие бизнесы сферы услуг.

Для определения лучших практик конкурентных продуктов была составлена сравнительная таблица реализованных функций. Она представлена в таблице 1. Эта таблица позволяет сравнить три программных продукта - LiveSklad, GinCore и HelloClient.

Это поможет определить характеристики о выбранных решений для автоматизации процесса подачи заявок на ремонт и обслуживание бытовой техники.

Таблица 1 – Сравнительная таблица характеристик продуктов

Характеристика / Функциональность	LiveSklad	GinCore	HelloClient
Стоимость	Платная	Бесплатная / Платная	Платная
Интеграция с онлайн-заявками	Да	Да	Да
Управление клиентской базой	Да	Да	Да
Отслеживание заявок и обращений	Да	Да	Да
Управление статусом ремонта	Да	Да	Да
Управление запчастями и складом	Да	Да	Нет
Генерация отчетов	Да	Да	Да
Мобильное приложение	Да	Да	Да
Поддержка русского языка	Да	Да	Да
Кастомизация и настройка	Ограниченная	Высокая	Ограниченная
Облачное хранилище	Да	Да	Да
Техническая поддержка	Есть	Есть	Есть

Каждое из рассмотренных приложений имеет свои плюсы и минусы. Однако ни одна из имеющихся разработок не подходит под все выдвинутые критерии. Таким образом, было принято решение о проектировании информационной системы под нужды компании.

1.4 Определение и обоснование требований к новому программному продукту

Необходимость автоматизации процессов подачи заявок на ремонт и обслуживание бытовой техники обоснована несколькими ключевыми факторами, направленными на повышение эффективности, улучшение качества обслуживания клиентов и оптимизацию работы предприятия.

Основные обоснования для автоматизации:

– Повышение оперативности и точности обработки заявок.

Автоматизация позволит существенно ускорить процесс подачи и регистрации заявок. Вместо ручного заполнения шаблонов заявки информация будет автоматически фиксироваться в системе, что снизит риск

ошибок и неточностей. Быстрая обработка заявок сократит время ожидания для клиентов и ускорит начало ремонтных работ.

– Улучшение клиентского опыта.

С внедрением автоматизированной системы клиенты смогут подавать заявки онлайн через веб-сайт или мобильное приложение, что значительно упростит и ускорит этот процесс. Автоматизированная система будет предоставлять клиентам возможность отслеживать статус своих заявок в режиме реального времени, что повысит прозрачность и удовлетворенность клиентов.

– Систематизация и хранение данных.

Автоматизация позволит создать централизованную базу данных, где будет храниться история всех заявок. Это облегчит доступ к информации для анализа и отчетности, а также позволит отслеживать повторные обращения клиентов и выявлять типичные проблемы с техникой. Систематизация данных поможет принимать более обоснованные управленческие решения.

– Оптимизация управления ресурсами.

Автоматизированная система позволит более эффективно распределять задачи между сотрудниками, отслеживать загрузку мастерских и контролировать запасы запчастей. Это приведет к более рациональному использованию ресурсов предприятия, снижению затрат и повышению производительности.

– Повышение качества обслуживания.

С автоматизацией процессов станет возможным более тщательно контролировать качество выполняемых ремонтных работ, фиксировать все этапы ремонта и получать обратную связь от клиентов. Это позволит улучшить стандарты обслуживания и быстрее реагировать на жалобы и предложения клиентов.

– Снижение административной нагрузки.

Автоматизация снимет часть рутинных административных задач с сотрудников, позволив им сосредоточиться на более важных и творческих

задачах. Это повысит мотивацию и удовлетворенность работой персонала, а также снизит риск ошибок, связанных с человеческим фактором.

– Поддержка принятия решений.

Автоматизированная система будет предоставлять менеджменту актуальную и полную информацию о текущем состоянии процессов, что позволит более оперативно и эффективно принимать управленческие решения. Данные о заявках, статусе ремонта, загрузке сотрудников и запасах запчастей будут доступны в режиме реального времени.

Таким образом, автоматизация процессов подачи заявок на ремонт и обслуживание бытовой техники обеспечит значительные улучшения в оперативности, качестве и эффективности работы предприятия. Она позволит создать прозрачную и удобную систему для клиентов, оптимизировать внутренние процессы и повысить конкурентоспособность предприятия на рынке.

Прежде чем формировать требования к новой технологии необходимо рассмотреть классификацию требований к ИС FURPS+. На рисунке 9 схематично представлены данная классификация требований.

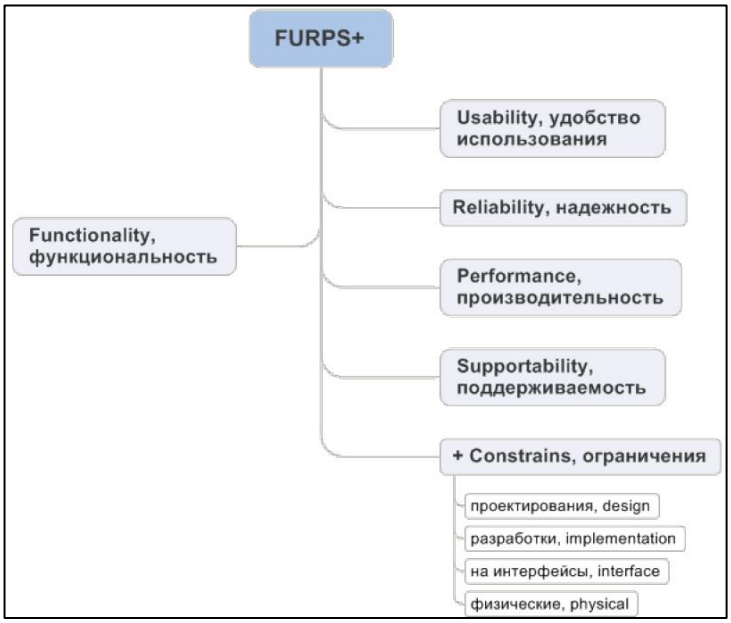

Рисунок 9 – Методология FURPS+

Как видно из рисунка 9, данная классификация состоит из следующих элементов:

– удобство. (данный критерий подразумевает простоту и читаемость интерфейса, наличие инструкций по использованию веб-сервиса для работников, недопустимость некорректных действий пользователей, стандартизацию оформления);

– функциональность (данный критерий зависит в первую очередь от предметной области, в которой разрабатывается система, разделяют множество возможных вариантов функциональности системы).

Например: журналирование (внесения в журнал и отслеживания необходимых событий); лицензирование (отслеживание необходимых лицензий); локализация (нацеленность на работу с языками); почта (функционал отправки, получения сообщений); техническая поддержка (помощь пользователям с системой).

– надежность (это минимизация ошибок и сбоев системы, восстановляемость данных в системе, стабильность и корректность функционирования);

– производительность (под производительностью системы подразумевается время отклика, эффективность системы (в том числе и экономическая), максимальный объем обрабатываемых данных и количества авторизированных пользователей);

– поддерживаемость (например, ее сочетаемость с различными операционными системами, приспособляемость, возможность изменения и так далее).

Классификация FURPS+ предполагает так же установку ограничений для системы. Выделяют такие основные виды ограничений как, ограничения проектирования (например, необходимость использования конкретных средств разработки), ограничения интерфейса (пример: ограничение формата данных), физические ограничения технических средств. Так же разрабатываемая система должна соответствовать законодательству.

Функциональные требования к веб-сервису выявлены следующие:

– наличие единой системы оборудований и их компонентов;

– автоматизация подачи заявки;

– уведомление клиента о смене статуса его заявки;

– уведомление мастера о назначении ему новой заявки;

– облегчение работы с заявками для сотрудников;

– история заявок;

– печать шаблона заявки на случай необходимости бумажного хранения;

– ограничение прав доступа в системе на различные действия;

– минимальные затраты системных ресурсов для работы системы;

– клиент должен иметь возможность отслеживать статус своей заявки в режиме реального времени, включая этапы диагностики, ремонта и готовности к выдаче;

– единая централизованная база данных.

Для пользовательского интерфейса формируются следующие требования:

– Интерфейс должен позволять клиенту легко и быстро заполнить и отправить заявку на ремонт бытовой техники. Включает обязательные поля для контактной информации, описания проблемы.

– Интерфейс должен предоставлять доступ к истории всех поданных заявок, включая детали выполненных ремонтов и обслуживаний.

– Интерфейс должен быть интуитивно понятным, с логичной навигацией и доступом ко всем основным функциям. Использование понятных иконок и описательных меток для облегчения понимания.

– Процесс подачи заявки должен быть максимально упрощен и не требовать от клиента более пяти шагов. Валидация данных должна

происходить в реальном времени для предотвращения ошибок при заполнении формы.

Требования к надежности:

– Интерфейс должен обеспечивать безотказную работу и корректное выполнение всех функций, включая обработку и сохранение данных.

– Система должна обеспечивать сохранность всех данных.

Требования к производительности:

– Интерфейс должен загружаться быстро и быть отзывчивым, обеспечивая моментальный отклик на действия пользователя.

– Отправленные данные должны обрабатываться не дольше 5 секунд;

Требования к поддерживаемости:

– Наличие руководства пользователя;

– Система должна иметь возможность расширяться и масштабироваться на добавление нового функционала;

В таблице 2 представлены требования к системе в удобном для понимания виде.

Таблица 2 – Требования к системе по методологии FURPS+

Группа требований по классификации FURPS+	Требования
Функциональность	наличие единой базы оборудований, автоматизация подачи заявки, уведомления клиента и мастера, история заявок, печать шаблона заявки, ограничение прав, единая база данных
Удобство использования	единообразие интерфейса: быстрое и удобное заполнение форм, поиск и фильтрация данных, интуитивно понятный интерфейс
Надежность	безотказная работа, сохранность всех данных;

Производительность	быстрая работа интерфейса, отзывчивость интерфейса, обработка данных не дольше 5 секунд
Поддерживаемость	наличие руководства пользователя, расширяемость

Как видно из таблицы 2, система должна полностью соответствовать требованиям методологии FURPS+.

1.5 Постановка задачи на разработку веб-сервиса

По сравнительной таблице было определено, что в будущем веб-сервисе необходимо реализовать следующие функции:

- подача заявок на ремонт;
- управление статусом ремонта;
- управление пользователями;
- история заявок;
- управление правами доступа;
- хранение данных в единой базе данных.

Для реализации веб-сервиса по оптимизации процессов в организации по ремонту бытовой техники были сформулированы следующие бизнес-цели и требования.

Бизнес-цели включают в себя улучшение клиентского сервиса, повышение эффективности работы предприятия, управление пользователями и доступом, а также обеспечение анализа и отчетности для принятия управленческих решений.

В рамках проекта требуется разработать и внедрить веб-приложение или мобильное приложение для подачи заявок на ремонт и отслеживания статуса ремонта. Также необходимо создать систему управления заявками и ремонтами, модуль управления пользователями и доступом, базу данных для хранения информации и механизмы анализа и отчетности. Интеграция с

другими системами, такими как системы управления складом, также требуется для полной автоматизации процессов.

Календарный план предусматривает проведение анализа требований, разработку дизайна и интерфейса, разработку приложений, создание системы управления заявками и ремонтами, модуля управления пользователями, интеграцию с другими системами, разработку механизмов анализа и отчетности, а также тестирование, обучение персонала и запуск системы в эксплуатацию.

Из анализа предметной области «Ремонт и обслуживание бытовой техники» можно сделать следующие выводы:

– Цифровизация процессов: Ручная обработка заявок и организация ремонтных процессов на бумаге приводят к задержкам, ошибкам и неэффективности. Внедрение цифровой системы управления заявками и ремонтом техники является необходимым для оптимизации процессов и повышения эффективности работы сервисного центра.

– Удовлетворение потребностей клиентов: Предоставление возможности онлайн-подачи заявок на ремонт и отслеживания статуса ремонта улучшит удобство обслуживания для клиентов и повысит их удовлетворенность.

– Применение лучших практик: Изучение и применение лучших практик в области ремонта и обслуживания, а также использование современных разработок и технологий, помогут организации стать более эффективной и конкурентоспособной.

– Ориентация на бизнес-цели: Разработка и внедрение ИТ-проекта должны быть ориентированы на достижение бизнес-целей организации, таких как улучшение клиентского сервиса, повышение эффективности работы и управление рисками.

Для улучшения ситуации необходимо внедрение цифровой системы управления заявками и ремонтом техники. Это позволит оптимизировать процессы и повысить эффективность работы организации. Клиенты смогут

подавать заявки онлайн, что упростит регистрацию и сократит время ожидания. В результате организация станет более конкурентоспособной и клиент-ориентированной. На рисунке 10 представлена контекстная диаграмма работы организации «как должно быть».

Диаграмма представляет деятельность предприятия по ремонту и обслуживанию бытовой техники. В центре диаграммы находится блок "Деятельность предприятия по ремонту и обслуживанию бытовой техники (A0)". Слева к этому блоку поступает входной элемент - "Клиент с бытовой техникой". Справа выходит выходной элемент - "Отремонтированная и обслуженная техника".

Сверху на блок влияют несколько управляющих элементов: Конституция РФ, Устав предприятия и Требования ИС (информационной системы). Эти элементы обеспечивают правовую и регламентную основу для деятельности предприятия.

Снизу к блоку подключены ресурсы - "Мастер" и "Администратор". Эти сотрудники выполняют основные работы по приему, диагностике, ремонту и выдаче бытовой техники.

Таким образом, диаграмма показывает процесс, где клиент приносит бытовую технику, которая проходит через этапы ремонта и обслуживания, и в итоге клиент получает отремонтированную технику. Весь процесс регулируется законодательными и внутренними нормативными документами и выполняется с участием соответствующих сотрудников предприятия.

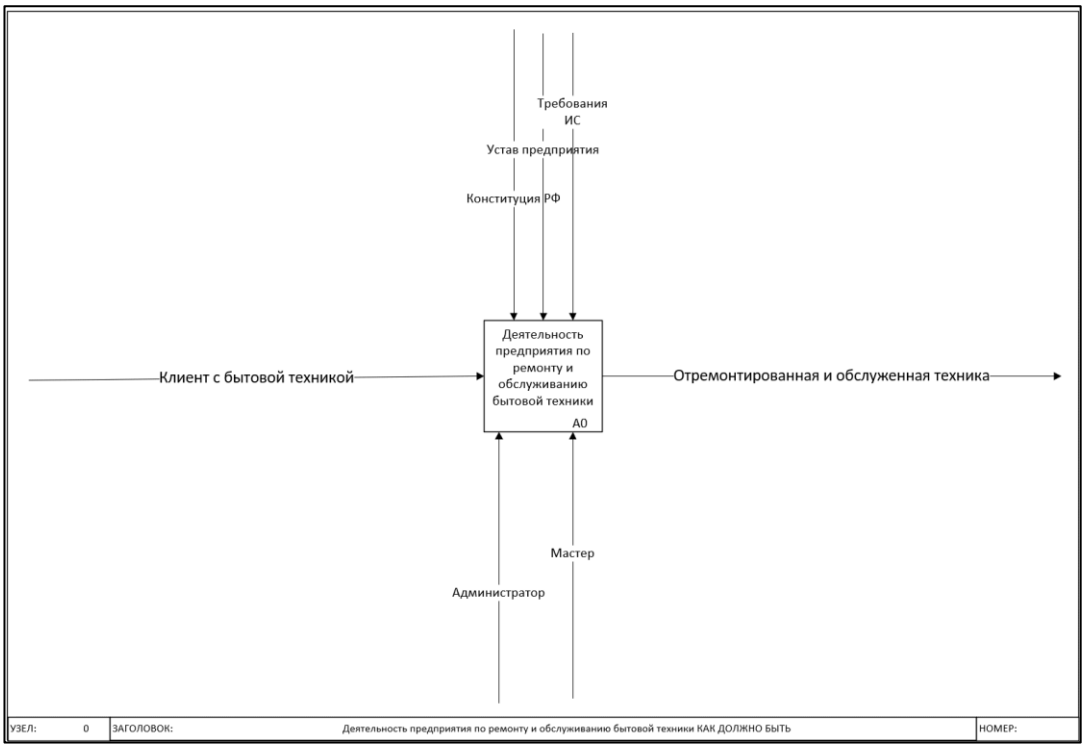

Рисунок 10 – Контекстная диаграмма «как должно быть»

Соблюдение лучших практик, таких как хранение данных в базе и предоставление возможности подачи онлайн-заявок, является ключевым для оптимизации процессов в организации по ремонту и обслуживанию бытовой техники.

На рисунке 11 представлена диаграмма декомпозиции работы организации «как должно быть».

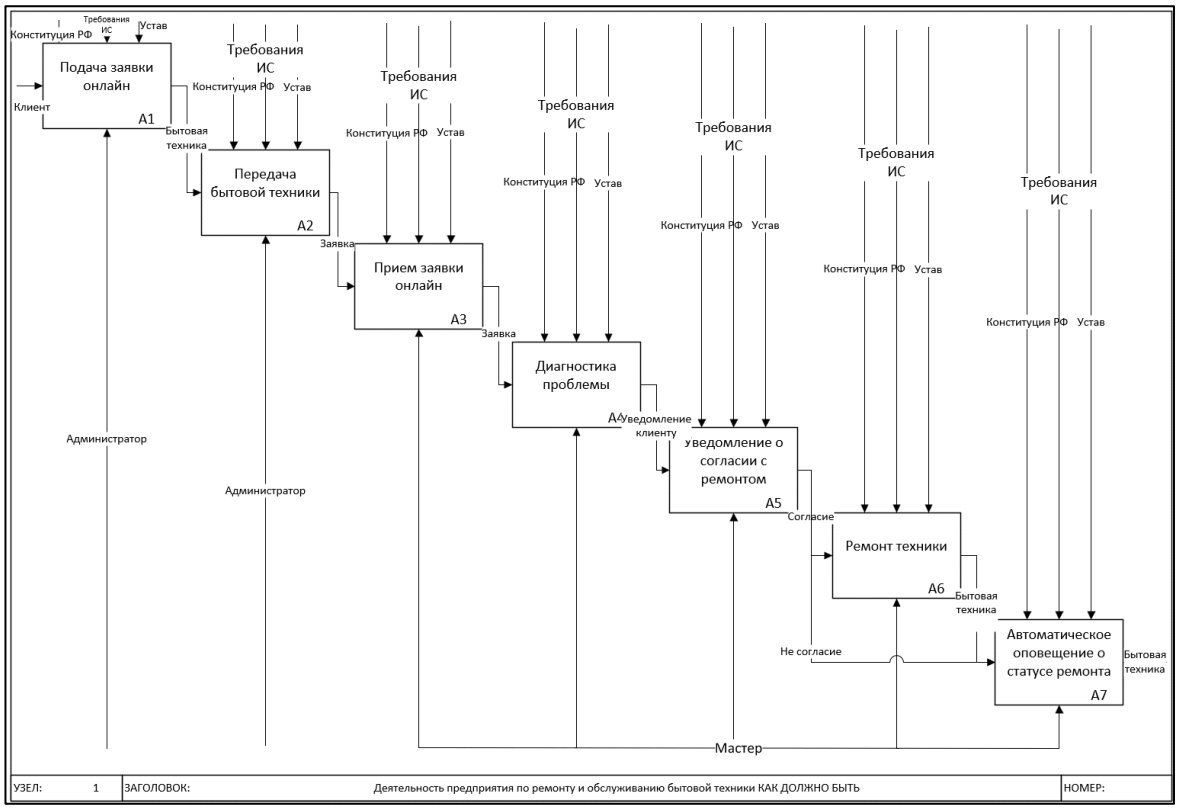

Рисунок 11 - Диаграмма декомпозиции «как должно быть»

Диаграмма представляет собой детализированное описание процесса подачи заявок на ремонт и обслуживание бытовой техники, начиная с подачи заявки клиентом и заканчивая автоматическим оповещением о статусе ремонта.

Процесс начинается с подачи заявки онлайн клиентом (A1). В этом этапе фиксируются все необходимые данные о бытовой технике и проблеме. Затем происходит передача бытовой техники в сервисный центр (A2), где администратор принимает технику и передает заявку на следующий этап.

На этапе приема заявки онлайн (A3) администратор регистрирует заявку в системе. Далее следует диагностика проблемы (A4), где мастер проводит первичную оценку состояния техники и определяет необходимый объем работ. Результаты диагностики передаются клиенту, и происходит уведомление о согласии с ремонтом (A5). Если клиент соглашается с условиями ремонта, мастер приступает к ремонту техники (A6).

После завершения ремонта происходит автоматическое оповещение о статусе ремонта (А7), информирующее клиента о готовности техники к выдаче. Весь процесс регулируется Конституцией РФ и Уставом предприятия, а также внутренними требованиями веб-сервиса. Администраторы и мастера выполняют ключевые роли в каждом из этапов, обеспечивая выполнение всех необходимых процедур.

Использование базы данных для хранения информации о клиентах, статусе ремонта и доступных запчастях позволяет упростить доступ к этой информации и улучшить её целостность. Это помогает предотвратить потерю данных, снижает вероятность ошибок и обеспечивает более эффективное управление информацией.

Предоставление возможности клиентам подавать заявки на ремонт онлайн сокращает время на обработку заявок и устраняет необходимость постоянного оповещения клиентов об статусе ремонта. Это улучшает удобство обслуживания для клиентов и сокращает время ожидания, что повышает удовлетворенность клиентов и способствует увеличению объема заявок.

Таким образом, использование базы данных для хранения информации и внедрение онлайн-заявок позволяют оптимизировать процессы в организации по ремонту и обслуживанию бытовой техники, повышая эффективность работы и обеспечивая более высокий уровень обслуживания для клиентов. Разрабатываемый веб-сервис должен соответствовать всем описанным требования по методологии FURPS+. Современное управление предприятием по ремонту и обслуживанию бытовой техники требует цифровизации процессов, применения лучших практик и ориентации на достижение бизнес-целей, что позволит организации эффективно функционировать и успешно конкурировать на рынке.

Выводы по главе 1

Первая глава данной работы включает анализ предметной области подача заявок на ремонт и обслуживание бытовой техники, была смоделирована работа бизнес-процесса на предприятии, разработана модель «как-есть».

Также в этом разделе были определены необходимые требования для системы и рассмотрены популярные готовые программные обеспечения: LiveSklad, GinCore, HelloClient. Сравнение показало превосходство GinCore в большинстве параметров, однако для выполнения всех требования программное обеспечение не подходит.

Исходя их чего были сформированы требования для разрабатываемого веб-сервиса. Требования были сформированы согласно методологии FURPS+. На основании сформированных требований, а также рассмотренных популярных программных обеспечений, была спроектирована модель «как должно быть». Модель соответствует всем определенным требованиям, что позволит спроектировать веб-сервис как необходимо.

Глава 2 Логическое проектирование веб-сервиса

2.1 Проектирование архитектуры веб-сервиса

При создании современных веб-сервисов немаловажным фактором является выбор способов и инструментов логического моделирования. Систематизировать и автоматизировать процесс разработки веб-сервиса возможно благодаря актуальным CASE-средствам. CASE-средства используются для преобразования общей, сформулированной без точности информации о требованиях к веб-сервису в точные определения.

Существуют следующие группы средств, используемых при логическом моделировании: отражающие нужную функциональность системы, иллюстрирующие отношения между данными, указывающие независящее от времени поведение системы.

Логическое моделирование в данной работе создано с помощью ER, UML диаграмм, а также диаграмм IDEF0. Такой выбор сделан из-за простоты понимая языка обозначения.

IDEF0 - это методология моделирования, предназначенная для создания функциональных моделей, которые описывают систему или процесс с точки зрения его функций и потоков данных. IDEF0 диаграммы используют блоки и стрелки для отображения функций (действий, процессов) и их взаимодействий через входы, выходы, управляющие элементы и механизмы. Этот метод широко используется для анализа, разработки и документирования сложных систем и процессов.

UML (Unified Modeling Language) - это универсальный язык моделирования, используемый для спецификации, визуализации, разработки и документирования программных систем. UML включает несколько типов диаграмм, таких как диаграммы классов, последовательностей, состояний и действий, которые помогают моделировать различные аспекты системы. UML применяется в основном в области разработки программного

обеспечения, позволяя создавать структурированные и понятные модели сложных программных систем.

ER-диаграмма - это метод моделирования данных, который используется для представления сущностей системы и их взаимосвязей. ER-диаграммы включают три основных компонента: сущности (объекты или понятия, представляющие данные), атрибуты (характеристики сущностей) и связи (отношения между сущностями). Этот метод широко применяется для проектирования и разработки баз данных, помогая визуализировать и структурировать данные, а также их взаимосвязи.

Следует отметить, что для разработки веб-сервиса далеко не всегда нужно выстраивать совершенно каждую из существующего списка диаграмм. Разработчик самостоятельно выбирает необходимый уровень конкретизации. Инструменты UML, с помощью которых создаются модели уже на этапе проектирования показывают корректность принятых архитектурных решений, целостность модели. Это полезно с целью снижения вероятности неудачи проекта.

Для проектирования диаграмм были использованы веб-приложения draw.io, dbdiagram.io.

Draw.io – это удобное бесплатное онлайн-приложение для создания диаграмм для рабочих процессов, BPM, организационных, сетевых диаграмм, блок-схемм (флоучарты), UML и принципиальных электросхем.

Dbdiagram.io - это простой инструмент проектирования баз данных, позволяющий рисовать диаграммы ER (Entity Relationship) простым написанием кода. Это один из бесплатных инструментов erd, предназначенный для разработчиков и аналитиков данных.

Выбранные средства способствуют ускорению разработки и росту продуктивности, минимизируют ручной труд, повышают ориентацию программы на потребителя, дают возможность управлять большими проектами или несколькими взаимосвязанными проектами, учитывает возможность общения между другими разработчиками.

Разработанные модели и диаграммы способствуют более конкретному пониманию функций, которые должны быть включены в функционал будущего веб-сервиса.

Логическая модель строится для создания визуализированного изображения изучаемой области в форме логической структуры. Логическая модель изображает сущности и их взаимоотношения между собой. Связи иллюстрируют отношения сущностей. Связи имеют свои свойства, которые регламентируют эти связи. Наиболее часто на связи влияет сила взаимосвязей между сущностями, то есть от качества взаимовлияния сущностей.

Первым делом была спроектирована диаграммы по методологии IDEF0, а именно контекстная диаграмма и диаграмма декомпозиции.

Контекстная диаграмма демонстрирует деятельность веб-сервиса в целом. Создается на одних из первых этапов проектирования и позволяет выявить основные функции будущего программного продукта.

На рисунке 12 изображена контекстная диаграмма веб-сервиса подачи заявок по ремонту и обслуживанию бытовой техники.

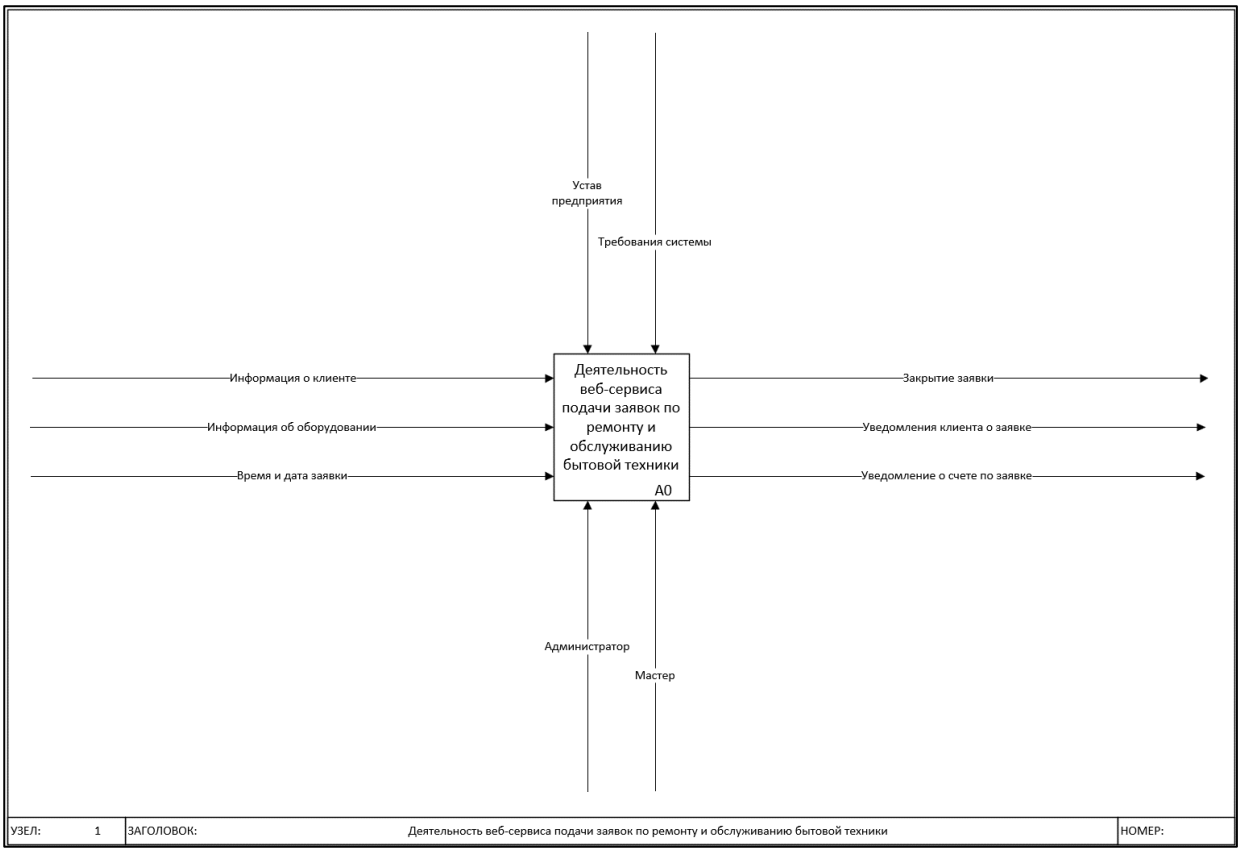

Рисунок 12 - Контекстная диаграмма

На контекстной диаграмме отображаются следующие сущности: Клиент, Сотрудник, Требования ИС, Устав предприятия и Заявка.

Клиент подает заявки на ремонт и обслуживание техники, и в результате обработки заявок получает информацию о статусе своей заявки.

Сотрудники вводят данные о выполнении работ по заявкам и контролируют процесс выполнения заявок, а также получают информацию о новых заявках, которые необходимо обработать.

Веб-сервис функционирует в соответствии с определенными требованиями, которые включают правила обработки данных, безопасность, надежность и другие аспекты.

Деятельность веб-сервиса регулируется уставом предприятия, который задает общие бизнес-правила и регламенты работы.

Обработанные заявки передаются на выход, чтобы их можно было отслеживать и управлять их выполнением.

Контекстная диаграмма используется для представления системы на высоком уровне и определения основных взаимодействий между системой и внешними сущностями. Она помогает понять, какие данные входят в систему, какие данные выходят из системы и какие внешние факторы влияют на систему.

В данном контексте контекстная диаграмма иллюстрирует, как веб-сервис по ремонту и обслуживанию бытовой техники взаимодействует с клиентами, сотрудниками и другими важными элементами, такими как требования ИС и устав предприятия.

Для детализации функций контекстную диаграмму разбивают на несколько частей, получается диаграмма декомпозиций. На рисунке 13 представлена диаграмма декомпозиций (A0) отображающая деятельность веб-сервиса более подробно чем предыдущая диаграмма.

Декомпозиция показывает, как высокоуровневый процесс "Деятельность веб-сервиса подачи заявок по ремонту и обслуживанию бытовой техники" разделяется на более мелкие подпроцессы, каждый из которых выполняет определенную функцию.

Декомпозиция контекстной диаграммы позволяет более детально рассмотреть внутренние процессы веб-сервиса, показать основные этапы работы с заявками и оборудованием, а также определить ключевые точки взаимодействия с пользователями и внешними системами. Это способствует лучшему пониманию структуры системы и помогает в дальнейшем проектировании и разработке отдельных компонентов.

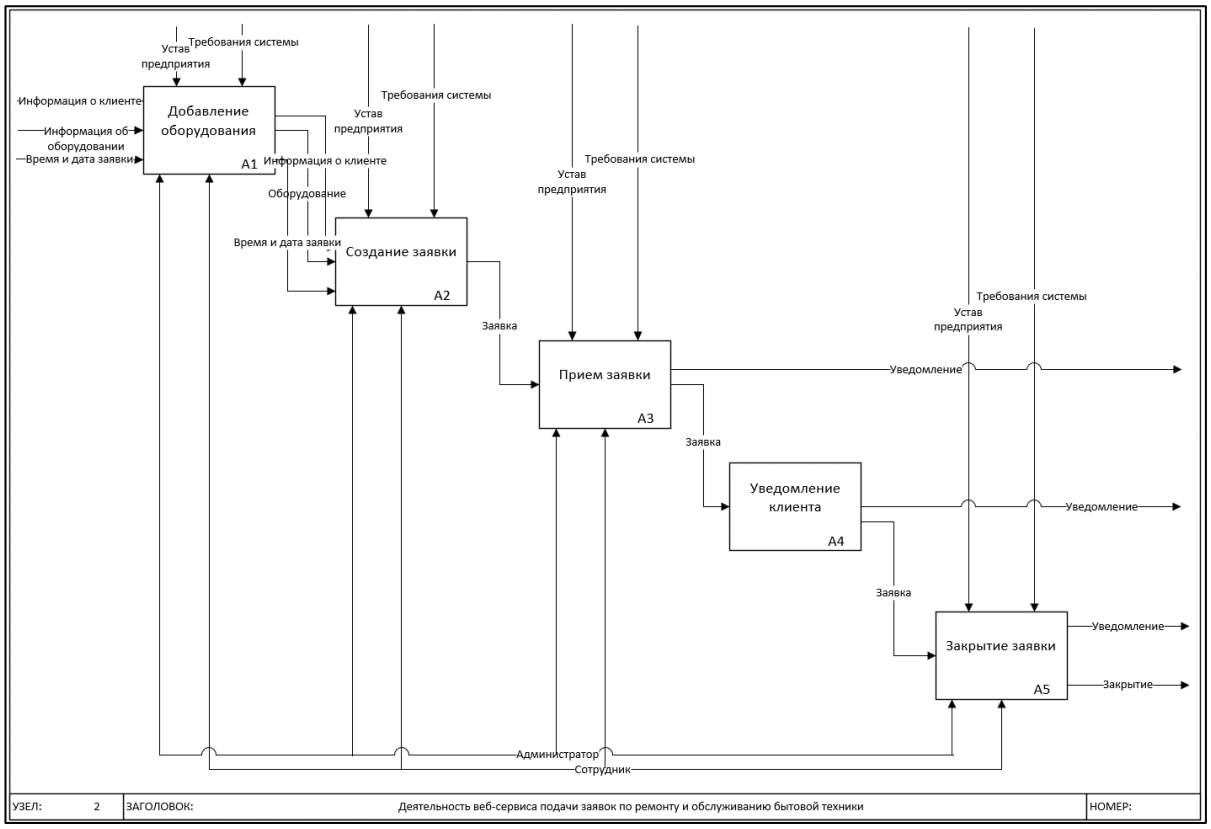

Рисунок 13 – Диаграмма декомпозиции

Диаграмма декомпозиции описывает основные процессы веб-сервиса по подаче заявок на ремонт и обслуживание бытовой техники, детализируя их входящие и исходящие потоки данных.

Первый процесс, "Добавление оборудования" (A1), включает ввод информации об оборудовании клиентом, а также учет бизнес-правил, регулируемых уставом предприятия, и технических требований системы. На выходе из этого процесса получаются данные об оборудовании, добавленном в систему.

Второй процесс, "Создание заявки" (A2), использует данные об оборудовании, добавленном клиентом, и учитывает бизнес-правила и технические требования, чтобы создать заявку на ремонт или обслуживание оборудования. Результатом этого процесса является сформированная заявка.

Третий процесс, "Прием заявки" (A3), начинается с входящей созданной заявки и также учитывает бизнес-правила и технические

требования системы. На выходе этого процесса заявка принимается для выполнения, и информация о новой заявке передается сотруднику, ответственному за её обработку.

Четвертый процесс, "Закрытие заявки" (A4), обрабатывает заявку, готовую к закрытию, следуя бизнес-правилам и техническим требованиям. На выходе получается закрытая заявка, завершенная после выполнения всех необходимых действий.

Эти процессы обеспечивают структурированный и последовательный подход к обработке заявок на ремонт и обслуживание, учитывая все необходимые бизнес-правила и технические требования для обеспечения надежной работы веб-сервиса.

Также было определено какие пользователи будут использовать веб-сервис. Для этого была спроектирована диаграмма вариантов использования.

Диаграмма вариантов использования является одним из основных инструментов в разработке программного обеспечения, используемым для моделирования функциональных требований системы. Она необходима для визуализации взаимодействий между пользователями (актёрами) и системой, а также для определения различных сценариев использования, которые система должна поддерживать. На рисунке 14 представлена диаграмма вариантов использования.

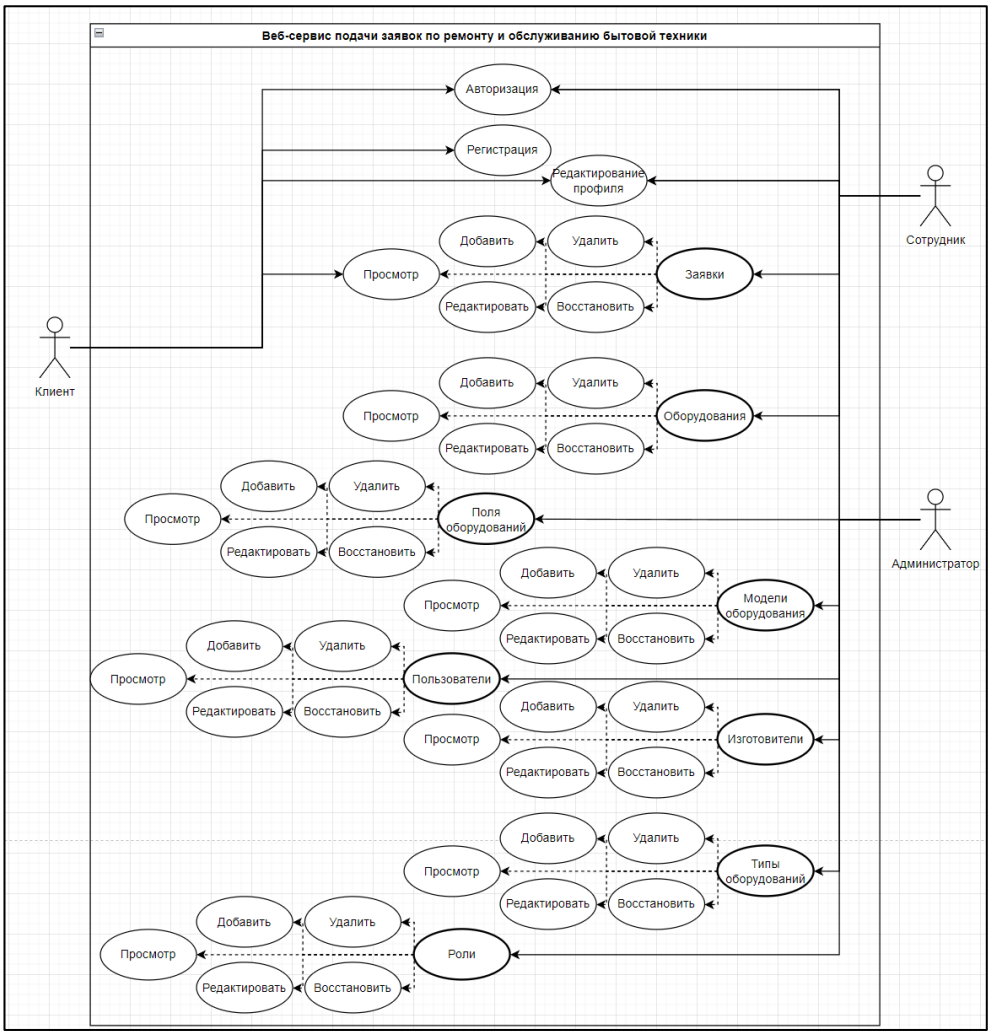

Рисунок 14 – Диаграмма вариантов использования

Диаграмма вариантов использования для веб-сервиса подачи заявок на ремонт и обслуживание бытовой техники отображает взаимодействие трех типов пользователей: Клиента, Сотрудника и Администратора, с системой. Она демонстрирует различные сценарии использования, в которых пользователи могут участвовать, а также показывает, какие действия они могут выполнять в системе.

Клиент имеет доступ к функциям авторизации, регистрации и редактирования профиля. Он может просматривать свои заявки и взаимодействовать с ними, например, добавлять, удалять, редактировать и восстанавливать заявки.

Сотрудник имеет доступ ко всем действиям, доступным клиенту, но также может управлять заявками, включая их просмотр, добавление, удаление, редактирование и восстановление. Он также может просматривать, добавлять, удалять, редактировать и восстанавливать информацию об оборудовании и полях оборудования.

Администратор обладает самыми широкими возможностями в системе. Помимо функций, доступных сотрудникам и клиентам, администратор может управлять пользователями, моделями оборудования, изготовителями и типами оборудования. Это включает в себя просмотр, добавление, удаление, редактирование и восстановление данных для каждой из этих категорий. Администратор также может управлять ролями пользователей, включая добавление, удаление, редактирование и восстановление ролей.

Диаграмма вариантов использования помогает визуализировать различные уровни доступа и функциональность, доступные различным типам пользователей системы. Она ясно показывает, какие действия могут выполняться каждым пользователем и как эти действия связаны с основными функциями системы, обеспечивая единое понимание функциональных возможностей и взаимодействий внутри веб-сервиса.

Диаграмма деятельности показывает поток выполнения или бизнес-процессы и операции, которые система или пользователи выполняют для достижения определенной цели. На основе диаграммы вариантов использования, спроектирована диаграмма деятельности для одного из ключевых процессов, например, процесса управления заявками. На рисунке 15 представлена диаграмма деятельности.

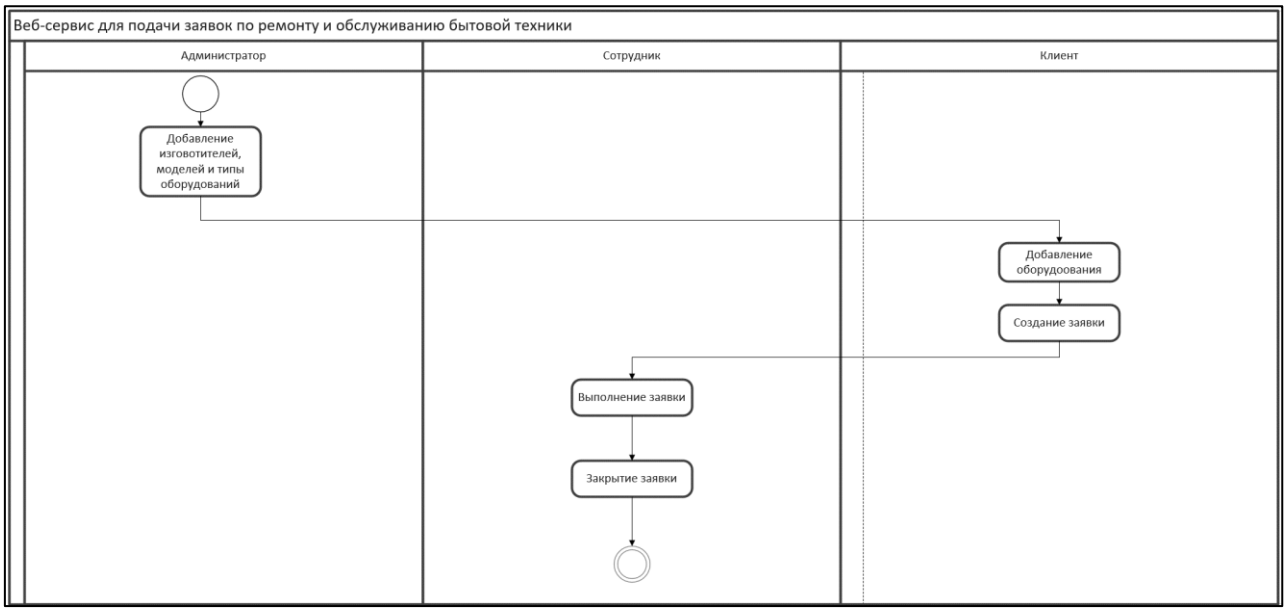

Рисунок 15 – Диаграмма деятельности

Диаграмма деятельности показывает процесс подачи заявок на ремонт и обслуживание бытовой техники в веб-сервисе, охватывая роли Администратора, Сотрудника и Клиента. Администратор начинает с добавления изготовителей, моделей и типов оборудования, что обеспечивает наличие всех необходимых данных о типах оборудования и моделях для создания заявок и управления оборудованием. Клиент добавляет оборудование в систему, а затем создает заявку на ремонт или обслуживание, вводя информацию о проблеме и выбирая необходимую услугу. После создания заявки, сотрудник берет на себя выполнение заявки, выполняя все необходимые шаги для ремонта или обслуживания оборудования. По завершении работы сотрудник закрывает заявку, завершая процесс.

Таким образом, диаграмма деятельности визуализирует весь процесс от начала до конца, начиная с настройки системы администратором, через взаимодействие клиента с системой для создания заявок, и заканчивая выполнением и закрытием заявок сотрудниками. Это помогает четко понять последовательность действий и взаимодействие между различными ролями в

системе, обеспечивая эффективную координацию и управление процессом подачи заявок на ремонт и обслуживание бытовой техники.

Диаграмма потоков данных - это графический инструмент, используемый для моделирования движения данных в информационной системе. Она показывает, как данные перемещаются между различными процессами, хранилищами данных и внешними сущностями (например, пользователями или другими системами). Основной целью диаграммы потоков данных является предоставление ясного и наглядного представления того, как информация обрабатывается и передается внутри системы. На рисунке 16 представлена диаграмма потоков данных.

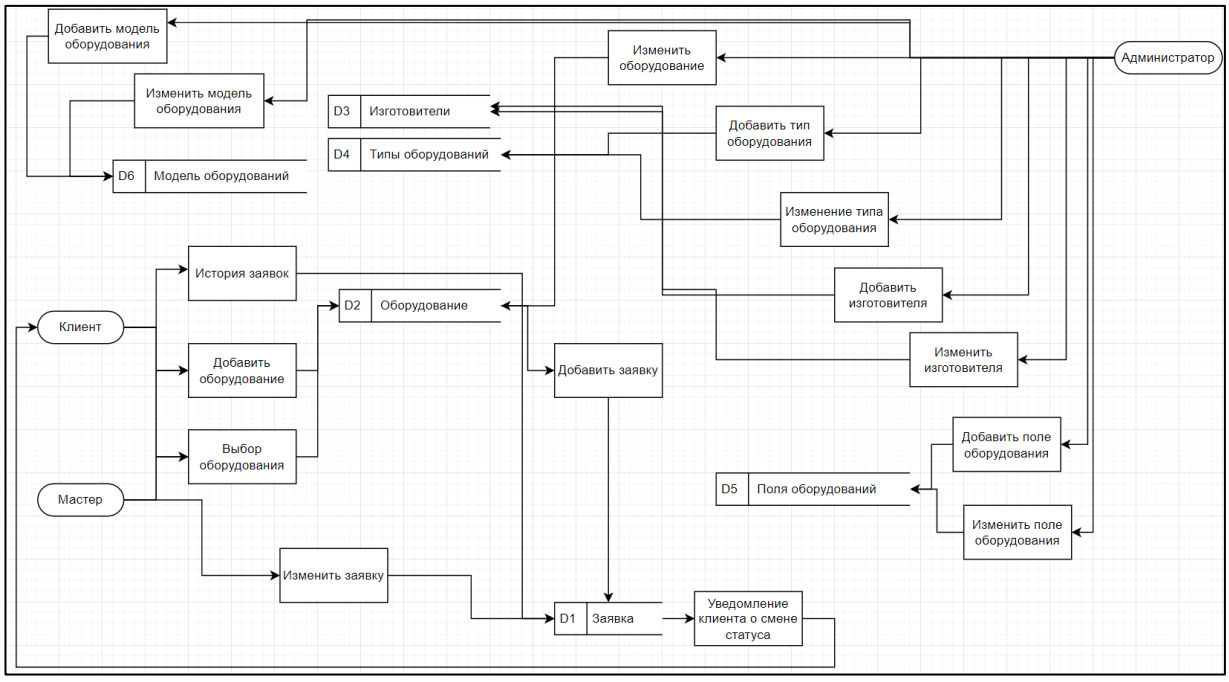

Рисунок 16 – Диаграмма потоков данных

Диаграмма потоков данных иллюстрирует процессы взаимодействия между клиентом, сотрудником и системой для управления заявками на ремонт и обслуживание бытовой техники. Клиент может инициировать несколько действий: просмотреть историю заявок, добавить оборудование и выбрать оборудование. Эти действия приводят к взаимодействию с

хранилищем данных "Оборудование" (обозначено как D2), откуда система получает или записывает информацию.

Когда клиент добавляет оборудование, данные записываются в хранилище "Оборудование". Затем клиент может создать заявку, выбрав оборудование из списка, что приводит к добавлению заявки в хранилище данных "Заявка" (обозначено как D1). После создания заявки система отправляет уведомление клиенту о смене статуса заявки.

Сотрудник имеет доступ к изменению заявки, что также взаимодействует с хранилищем данных "Заявка". Внесенные изменения в заявку могут привести к обновлению его статуса, после чего клиент получает уведомление о смене статуса заявки.

Таким образом, диаграмма показывает поток данных между клиентом, сотрудником и различными хранилищами данных, демонстрируя процессы добавления и выбора оборудования, создания и изменения заявок, а также отправки уведомлений клиенту. Эта модель помогает визуализировать, как информация перемещается и обрабатывается в системе, обеспечивая ясное понимание рабочих процессов и взаимодействий.

Диаграмма развертывания - это тип диаграммы UML, который показывает физическое расположение программных компонентов и их взаимосвязи на аппаратных платформах. Она используется для моделирования архитектуры системы, отображая, как программные модули и компоненты развертываются на физических устройствах, таких как серверы, клиентские компьютеры и другие устройства. Представлена на рисунке 17.

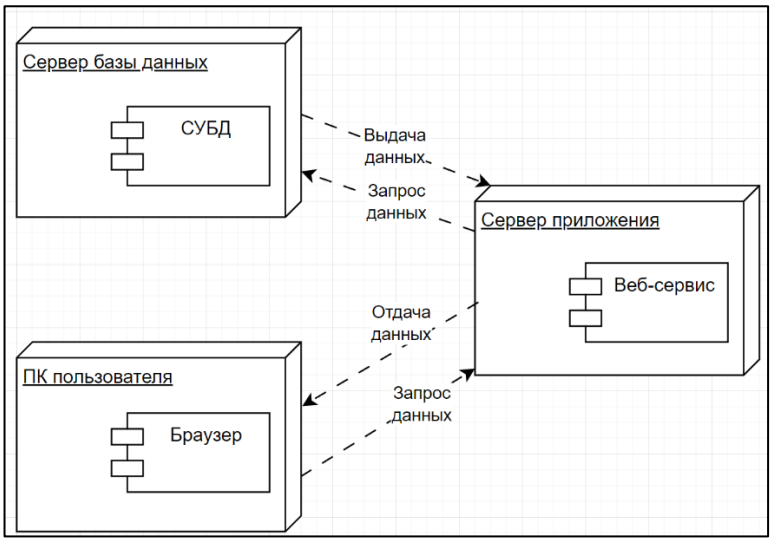

Рисунок 17 – Диаграмма развертывания

Диаграмма развертывания показывает взаимодействие между различными компонентами системы в физическом окружении. На диаграмме представлены три основных узла: сервер базы данных, сервер приложения и ПК пользователя.

Сервер базы данных содержит систему управления базами данных (СУБД), которая отвечает за хранение и выдачу данных. Сервер приложения включает веб-сервис, который обрабатывает запросы от клиентов и взаимодействует с СУБД для получения и обновления данных. ПК пользователя содержит браузер, который используется для отправки запросов к серверу приложения и получения данных от него.

Процесс начинается с того, что пользователь через браузер на ПК пользователя отправляет запрос данных на сервер приложения. Сервер приложения принимает запрос, обрабатывает его и при необходимости отправляет запрос к серверу базы данных для получения необходимых данных. Сервер базы данных обрабатывает запрос и возвращает данные на сервер приложения. Затем сервер приложения отправляет полученные данные обратно на ПК пользователя через браузер.

Диаграмма также показывает направление потоков данных между компонентами: запрос данных от браузера к серверу приложения, запрос данных от сервера приложения к серверу базы данных, выдача данных от сервера базы данных к серверу приложения и отдача данных от сервера приложения к браузеру пользователя. Эти потоки данных обеспечивают взаимодействие между различными частями системы, позволяя пользователю получать необходимые данные через веб-сервис.

2.2 Проектирование базы данных веб-сервиса

Для проектирования баз данных были выбраны case-средства draw.io и dbdiagram.io.

Draw.io предлагает интуитивно понятный интерфейс и богатый набор инструментов для создания различных типов диаграмм, включая ER-диаграммы, которые используются для проектирования структур баз данных.

Draw.io доступен бесплатно, что делает его удобным для всех пользователей, включая студентов и профессионалов. Инструмент поддерживает интеграцию с различными облачными сервисами, такими как Google Drive, OneDrive, Dropbox, что позволяет сохранять и делиться диаграммами легко и безопасно. В Draw.io можно работать с различными шаблонами и формами, которые упрощают создание сложных ER-диаграмм, UML-диаграмм и других типов визуальных представлений. Поддержка совместной работы в реальном времени позволяет нескольким пользователям одновременно редактировать одну и ту же диаграмму, что особенно полезно для командных проектов.

Draw.io является мощным и гибким инструментом для проектирования баз данных, который сочетает в себе богатый функционал и простоту использования. С его помощью можно создать профессиональные ER-диаграммы, которые помогут в проектировании и документировании структур баз данных.

Dbdiagram.io позволяет легко проектировать структуры баз данных с использованием простого и интуитивно понятного синтаксиса. Этот инструмент подходит как для начинающих, так и для опытных разработчиков.

Dbdiagram.io позволяет пользователям описывать структуру базы данных с помощью текстового языка, что упрощает процесс создания диаграмм. Он позволяет просто написать определения таблиц и их отношений, и инструмент автоматически сгенерирует соответствующую диаграмму. Это делает процесс проектирования базы данных быстрым и удобным.

Инструмент поддерживает интеграцию с различными системами управления базами данных (СУБД), что позволяет экспортировать схемы в SQL-код для популярных СУБД, таких как MySQL, PostgreSQL и SQLite. Это упрощает переход от проектирования к реализации базы данных.

Концептуальная схема - это высокоуровневое описание структуры данных, которое включает в себя основные компоненты системы и их взаимоотношения. Она не привязана к конкретной реализации или технологии и служит мостом между бизнес-требованиями и физической реализацией базы данных. Концептуальная схема обеспечивает ясное понимание того, как данные организованы и как они будут использоваться в различных частях системы.

Концептуальная схема является фундаментом для дальнейшего развития системы и выполнения её функциональных требований. Она служит абстрактным представлением данных, определяя основные сущности и их взаимосвязи, что в свою очередь позволяет создавать структурированные и эффективные модели данных. На рисунке 18 представлена концептуальная схема.

Концептуальная схема обеспечивает логическую основу для дальнейшего создания детализированных схем данных, проектирования базы

данных и разработки приложений, тем самым способствуя успешной реализации веб-сервиса по ремонту и обслуживанию бытовой техники.

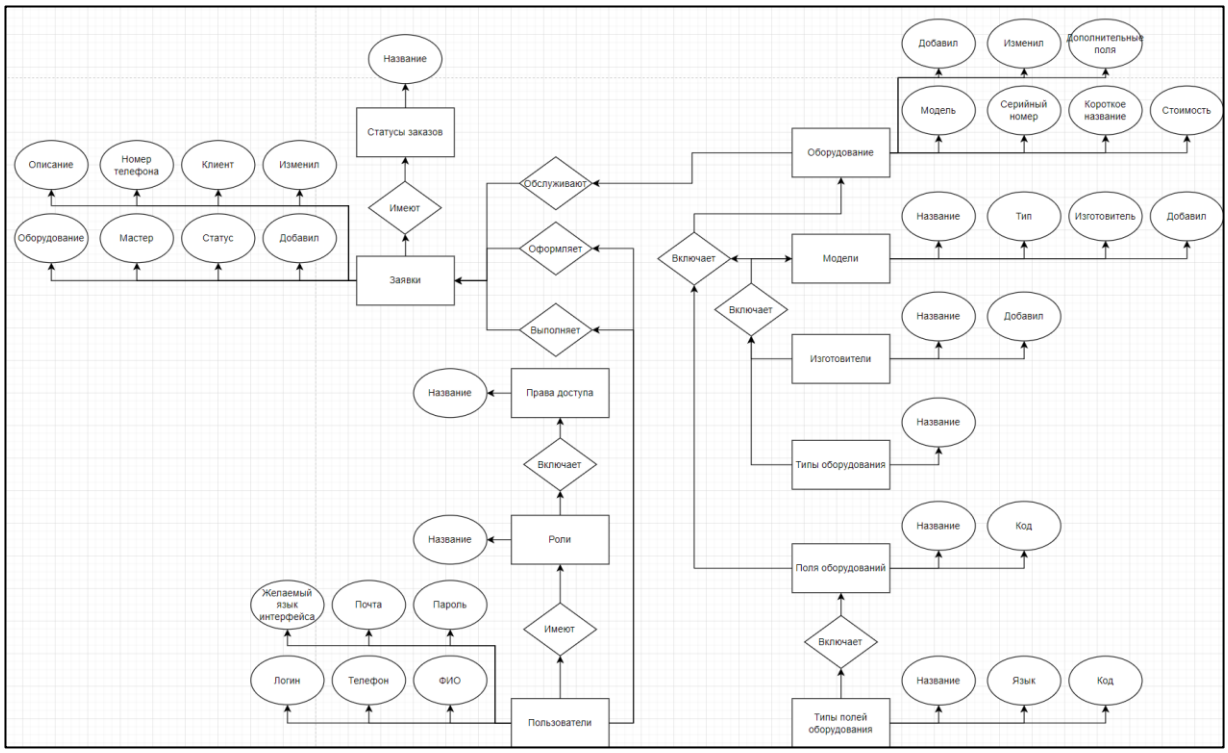

Рисунок 18 – Концептуальная схема

Концептуальная схема, представленная на рисунке 18, описывает структуру данных веб-сервиса по ремонту и обслуживанию бытовой техники. Схема включает основные сущности и их взаимосвязи, необходимые для эффективного функционирования системы.

Сущность "Оборудование" включает такие атрибуты, как описание, номер телефона, клиент, мастер, статус, изменил и добавил. Эта информация позволяет отслеживать данные о бытовой технике, которую клиенты приносят на ремонт или обслуживание.

Сущность "Заявки" включает атрибуты, которые позволяют идентифицировать и управлять заявками, поступающими в систему. Заявки связаны с различными статусами, такими как обслуживают или оформляют.

Это позволяет системе отслеживать состояние каждой заявки на разных этапах его обработки.

Статусы заявок описываются сущностью "Статусы заявок", которая содержит атрибуты, определяющие текущее состояние каждой заявки. Это помогает контролировать прогресс выполнения заявок и управлять ими.

Сущность "Модели" описывает данные о различных моделях оборудования, которые могут включать название, серийный номер, тип, изготовителя, а также информацию о том, кто добавил или изменил модель. Дополнительные поля для моделей могут включать такие характеристики, как короткое название, стоимость и другие важные параметры.

Сущность "Изготовители" включает информацию о производителях оборудования, таких как название и данные о том, кто добавил или изменил информацию об изготовителе. Это позволяет системе поддерживать актуальные данные о компаниях, производящих бытовую технику.

Сущность "Пользователи" включает атрибуты, такие как логин, почта, телефон, пароль, ФИО и желаемый язык интерфейса. Это позволяет системе управлять учетными записями пользователей, обеспечивая им доступ к функционалу веб-сервиса в соответствии с их ролями и правами доступа.

Сущность "Роли" и "Права доступа" описывают различные уровни доступа и привилегии, предоставляемые пользователям системы в зависимости от их ролей. Это включает атрибуты, определяющие, какие действия могут выполнять пользователи в зависимости от их роли в системе.

Типы оборудования и поля оборудования включают информацию о различных категориях бытовой техники и их специфических характеристиках. Это помогает системе эффективно классифицировать и управлять данными о различном оборудовании, поступающем на обслуживание.

В целом, концептуальная схема предоставляет детализированное описание структуры данных веб-сервиса, обеспечивая эффективное

управление заявками, оборудованием, пользователями и их взаимодействиями.

При разработке рассматриваемого веб-сервиса был выбран вариант реляционной модели базы данных.

Реляционная база данных является оптимальным выбором для хранения и управления данными в проекте по следующим причинам:

– позволяют хранить данные в структурированной форме с использованием таблиц, что обеспечивает четкую организацию информации. Таблицы, представленные в реляционной модели данных, соответствуют сущностям, что упрощает понимание и управление данными.

– обеспечивают целостность данных с помощью ограничений, таких как первичные и внешние ключи. Это гарантирует, что данные во всех связанных таблицах остаются согласованными, предотвращая ошибки и нарушения целостности.

– поддерживают процесс нормализации, который помогает минимизировать избыточность данных и улучшить их целостность. Нормализация структурирует данные таким образом, чтобы минимизировать дублирование информации и улучшить логическую организацию базы данных.

– поддерживают мощный язык запросов SQL, который позволяет выполнять сложные запросы для извлечения, обновления и анализа данных. Это особенно полезно для выполнения аналитических отчетов и получения сложных выборок данных, что необходимо для управления заявками, пользователями и оборудованием.

– современные реляционные системы управления базами данных обладают высокими возможностями масштабирования и производительности. Они могут эффективно обрабатывать большие объемы данных и обеспечивать высокую скорость выполнения

транзакций и запросов, что важно для систем с большим количеством пользователей и заявок.

Логическая модель данных организована таким образом, чтобы обеспечить комплексное управление данными о пользователях, оборудовании, заявках и системных операциях.

Таким образом, выбор реляционной базы данных для данного проекта обоснован необходимостью структурированного хранения данных, обеспечения их целостности, поддержки сложных запросов, безопасности, масштабируемости и надежности. Модель предоставляют все необходимые инструменты для эффективного управления данными и обеспечения высокого уровня обслуживания клиентов.

После создания концептуальной схемы данных, следующим этапом проектирования является разработка логической модели. Концептуальная схема, будучи абстрактным и высокоуровневым представлением данных и их взаимосвязей, предоставляет основу для дальнейшей детализации и уточнения данных. Логическая модель уточняет концептуальную схему, добавляя конкретные атрибуты сущностей, их типы данных и ключевые ограничения, такие как первичные и внешние ключи.

Этот этап важен для того, чтобы подготовить данные к реализации в конкретной СУБД. Логическая модель служит промежуточным шагом между абстрактной концепцией данных и их физической реализацией, обеспечивая, что все бизнес-правила и требования системы будут учтены при создании базы данных. В процессе разработки логической модели может потребоваться дополнительно проанализировать взаимосвязи данных, их нормализацию и оптимизацию для повышения производительности и надежности системы.

На рисунке 19 представлена логическая модель данных веб-сервиса.

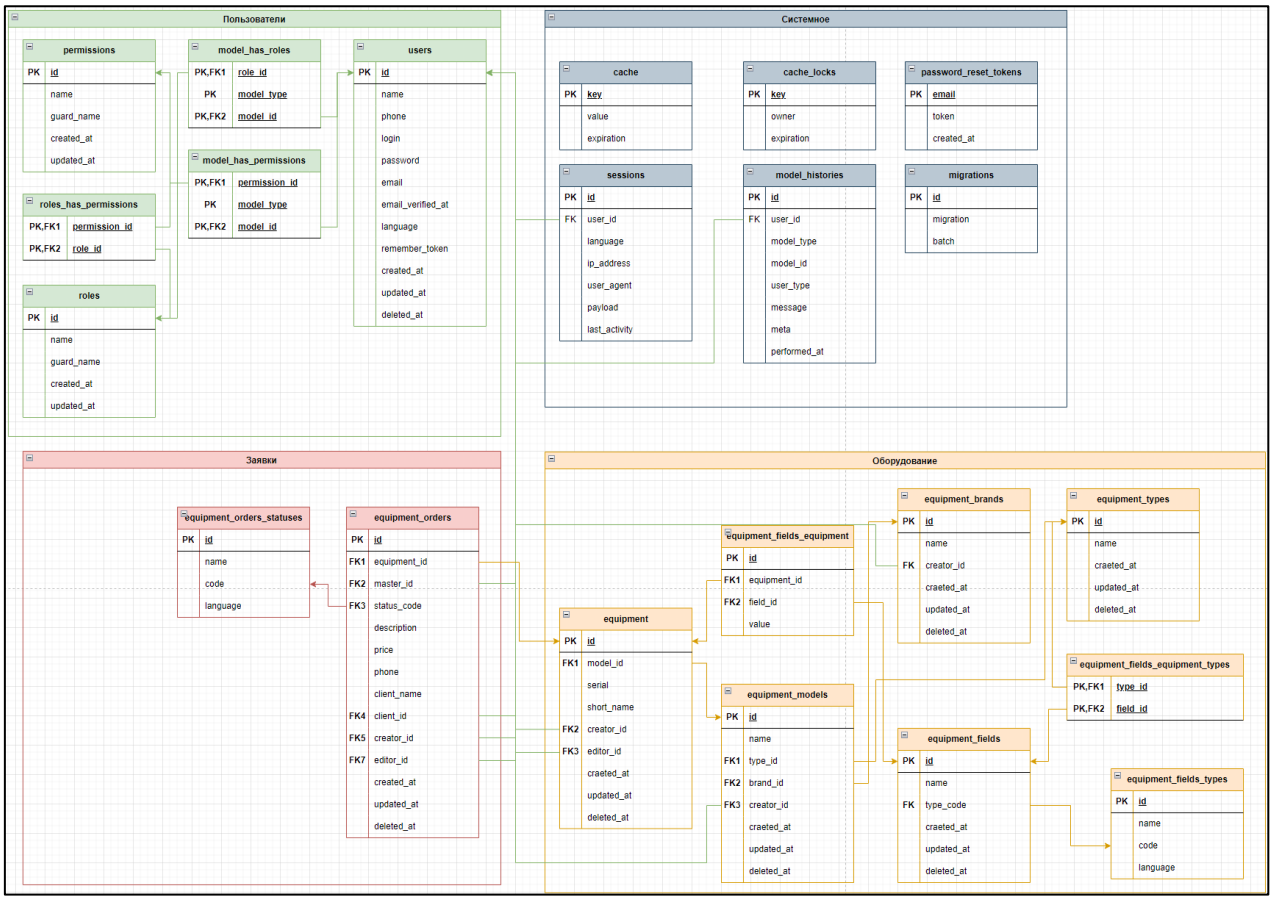

Рисунок 19 – Логическая модель данных

Логическая модель данных, представленная на рисунке 19, подробно описывает структуру базы данных для веб-сервиса по ремонту и обслуживанию бытовой техники. Она включает несколько ключевых областей: пользователи, заявки, оборудование и системные таблицы.

В области пользователей находятся таблицы, которые управляют информацией о пользователях и их ролями. Таблица "users" содержит атрибуты, такие как имя, телефон, логин, пароль, email и язык. Также здесь есть таблицы для управления ролями и правами доступа, включая "roles", "permissions" и связующие таблицы "model_has_roles" и "model_has_permissions", которые обеспечивают гибкую настройку прав доступа и ролей пользователей в системе.

Область заявок включает таблицы для управления заявками на ремонт и обслуживание. Таблица "equipment_orders" содержит данные о заявках,

такие как описание, цена, телефон клиента, имя клиента, идентификаторы оборудования, мастера, создателя и редактора заявки. Таблица "equipment_orders_statuses" управляет статусами заявок, включая такие атрибуты, как название и код статуса.

В области оборудования представлены таблицы для управления информацией об оборудовании и его характеристиках. Таблица "equipment" содержит атрибуты оборудования, включая серийный номер, короткое название, идентификаторы модели, создателя и редактора. Связанные таблицы, такие как "equipment_models", "equipment_brands" и "equipment_types", управляют данными о моделях, производителях и типах оборудования. Дополнительные характеристики оборудования описаны в таблицах "equipment_fields" и "equipment_fields_equipment", которые связывают конкретное оборудование с его дополнительными полями и их значениями.

Системные таблицы включают различные технические данные, необходимые для функционирования системы. Таблицы "cache" и "cache_locks" управляют кэшем, "sessions" - пользовательскими сессиями, "password_reset_tokens" - токенами для сброса паролей, а "migrations" - миграциями базы данных. Таблица "model_histories" отслеживает изменения в моделях, фиксируя такие атрибуты, как идентификатор пользователя, тип модели, сообщение и метаданные изменений.

Физическая модель данных - это детализированное описание структуры базы данных, разработанное с учетом конкретной системы управления базами данных (СУБД). Она включает все технические аспекты, необходимые для реализации базы данных на физическом уровне, и обеспечивает точное представление данных, оптимизацию производительности и управление доступом к данным.

Физическая модель отображает физическую реализацию логической модели данных для веб-сервиса по ремонту и обслуживанию бытовой техники. Она включает таблицы, их атрибуты и связи между ними,

необходимые для функционирования системы. На рисунке 20 представлена физическая модель данных веб-сервиса.

Центральное место в модели занимают таблицы "users", "equipment_orders" и "equipment". Таблица "users" содержит информацию о пользователях, включая такие атрибуты, как имя, телефон, логин, пароль и язык интерфейса. Таблица "equipment_orders" управляет заказами и содержит поля для описания заказа, идентификаторы мастера, клиента, статуса и других связанных сущностей. Таблица "equipment" хранит данные об оборудовании, такие как серийный номер, краткое название, идентификаторы модели и создателя.

Другие важные таблицы включают "equipment_models", "equipment_brands" и "equipment_types", которые управляют данными о моделях, производителях и типах оборудования соответственно. Таблица "equipment_orders_statuses" хранит статусы заказов, а "equipment_fields" и связанные с ней таблицы управляют дополнительными характеристиками оборудования.

Связанные таблицы, такие как "roles", "permissions" и "model_has_roles", обеспечивают управление ролями и правами доступа пользователей. Таблицы "sessions", "cache", "migrations" и "password_reset_tokens" поддерживают системные функции, такие как управление сессиями, кэширование и сброс паролей.

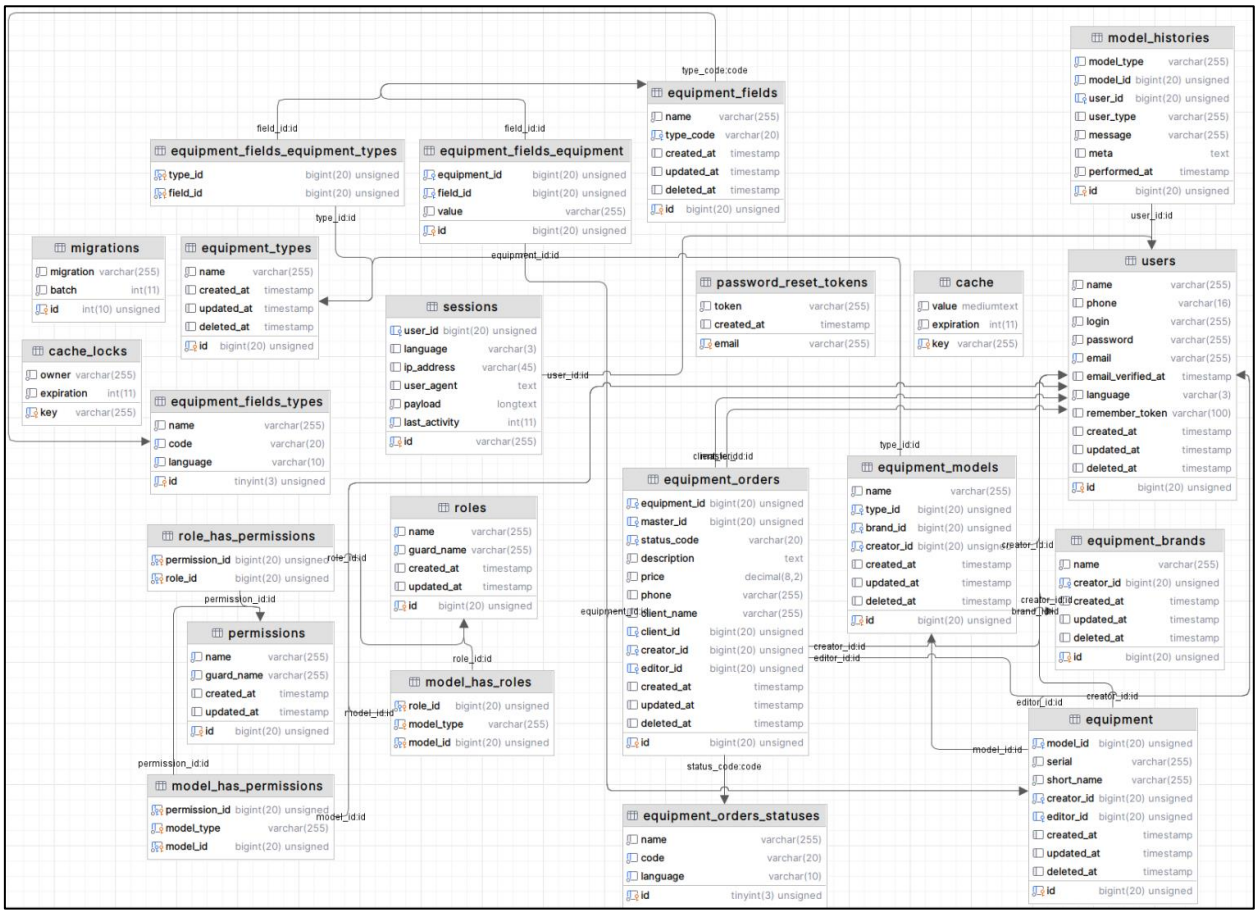

Рисунок 20 – Физическая модель данных

Переход от логической модели к физической модели включает преобразование абстрактных сущностей и атрибутов в конкретные таблицы и столбцы базы данных. Этот процесс также включает определение типов данных для каждого столбца, создание ограничений целостности, таких как первичные и внешние ключи, и оптимизацию структуры данных для повышения производительности. Физическая модель реализуется в конкретной системе управления базами данных (СУБД), что позволяет эффективно хранить, извлекать и управлять данными в рамках веб-сервиса.

2.3 Проектирование пользовательского интерфейса

Проектирование интерфейса включает в себя создание макетов экранов, определение навигации и пользовательских потоков, а также обеспечение взаимодействия пользователя с системой, чтобы выполнить задачи, выявленные на этапе анализа требований и развертывания.

Первым делом проектировались страницы авторизации пользователя. Для это спроектирована страница входа и регистрации. На рисунке 21 представлена прототип страница авторизации.

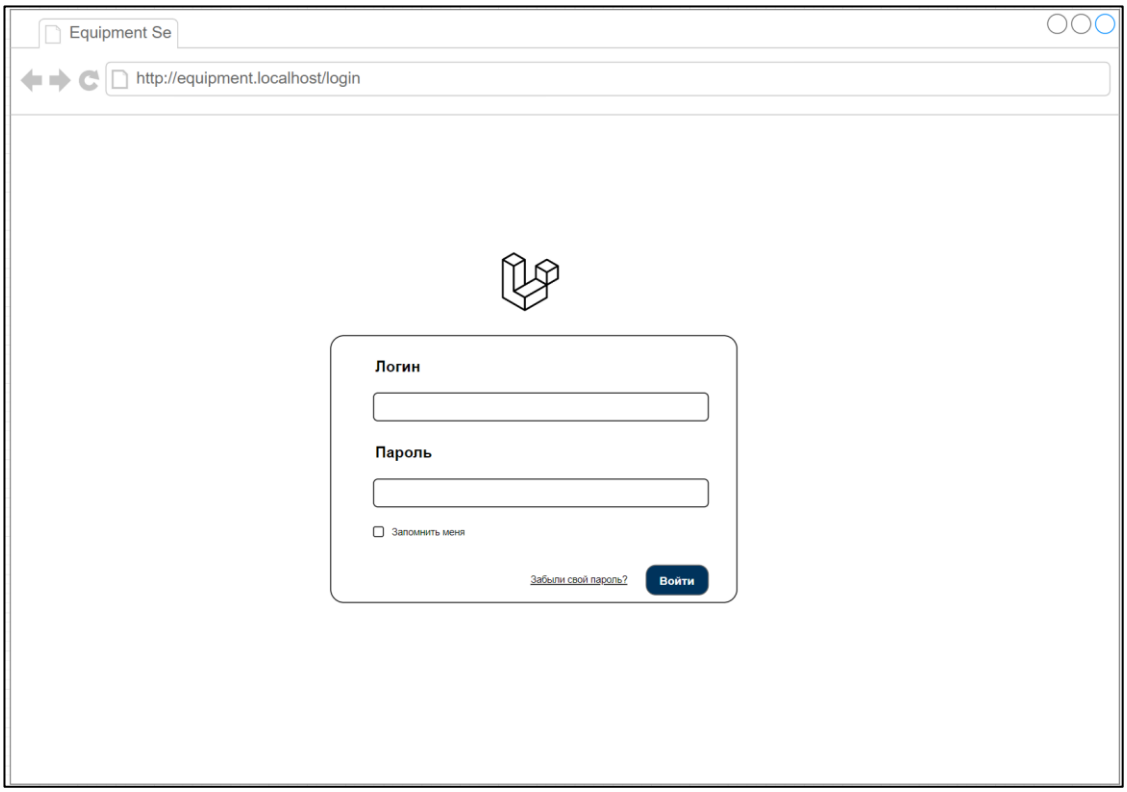

Рисунок 21 – Прототип страницы авторизации

В верхней части формы отображается логотип приложения или используемого фреймворка. Под логотипом расположены два текстовых поля для ввода логина и пароля. Поле логина предназначено для ввода имени пользователя или идентификатора, а поле пароля скрывает вводимые

символы для обеспечения безопасности. Под полями для ввода находится чекбокс "Запомнить меня", позволяющий браузеру запомнить логин и пароль пользователя для автоматической аутентификации при последующих входах.

Ниже чекбокса расположена ссылка "Забыли свой пароль?", которая позволяет пользователю перейти на страницу восстановления пароля в случае утери учетных данных. Завершает форму кнопка "Войти", которая отправляет введенные данные для аутентификации пользователя. При нажатии на эту кнопку происходит проверка введенных данных, и, если они верны, пользователь получает доступ к системе.

Форма входа размещена в центре страницы, что делает её легко доступной и заметной для пользователей. Оформление страницы простое и интуитивно понятное, что способствует удобству использования.

На рисунке 22 представлен прототип страницы регистрации.

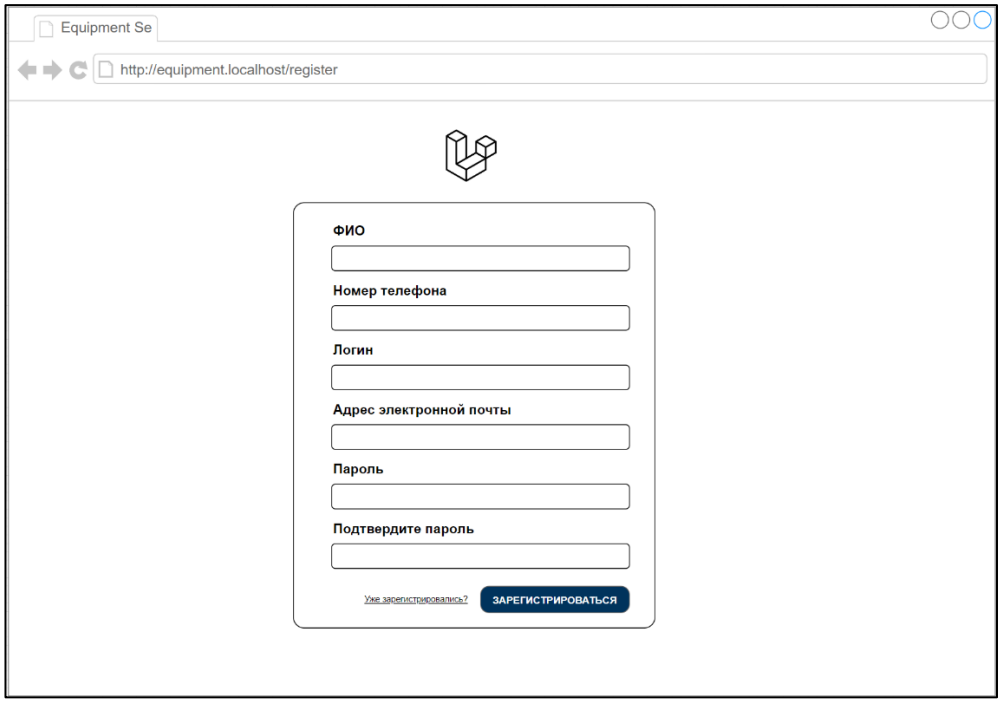

Рисунок 22 – Прототип страницы регистрации

Форма регистрации начинается с отображения логотипа приложения или фреймворка в верхней части. Под логотипом располагается несколько

текстовых полей, предназначенных для ввода информации пользователем. Поля включают ввод ФИО, номера телефона, логина, адреса электронной почты, пароля и подтверждения пароля. Эти поля позволяют пользователю предоставить необходимую информацию для создания учетной записи.

В нижней части формы размещена кнопка "ЗАРЕГИСТРИРОВАТЬСЯ", которая отправляет введенные данные для создания новой учетной записи в системе. Под кнопкой также расположена ссылка "Уже зарегистрированы?", которая позволяет пользователю перейти на страницу входа, если у него уже есть учетная запись.

Форма регистрации расположена в центре страницы, обеспечивая её видимость и удобный доступ для пользователя. Дизайн формы прост и интуитивно понятен, что способствует легкому заполнению и созданию учетной записи.

После авторизации пользователь попадает на рабочую страницу системы. На рисунке 23 представлена рабочая страница веб-сервиса.

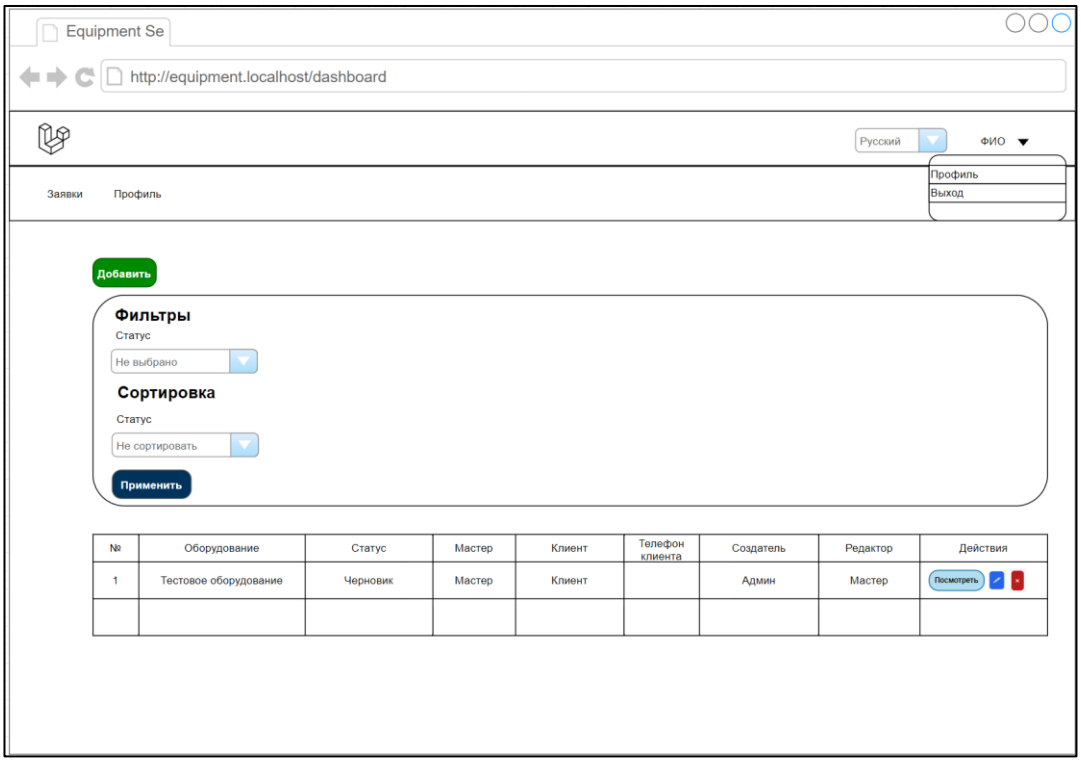

Рисунок 23 – Прототип рабочей страницы

Интерфейс включает логотип приложения в верхнем левом углу. Справа от логотипа расположены элементы управления языком и профилем пользователя. Выбран язык "Русский", а рядом с ФИО пользователя есть выпадающий список с опциями "Профиль" и "Выход", позволяющими пользователю просмотреть и изменить свои данные или выйти из системы.

Ниже логотипа находится горизонтальное меню навигации с вкладками "Заявки" и "Профиль". Текущая вкладка - "Заявки". Под меню навигации располагается основное содержимое панели управления.

Основное содержимое включает кнопку "Добавить" для создания новой заявки, а также блок фильтров и сортировки. Фильтры позволяют выбирать статус заявки, а сортировка - сортировать заявки по статусу. Кнопка "Применить" позволяет применить выбранные фильтры и сортировку.

Ниже блока фильтров и сортировки находится таблица с данными о заявках. Таблица включает следующие столбцы: номер, оборудование, статус, мастер, клиент, телефон клиента, создатель, редактор и действия. В таблице представлена одна запись с тестовым оборудованием, статусом "Черновик", указанием мастера, клиента, телефона клиента, создателя и редактора. В столбце "Действия" находятся кнопки для просмотра, редактирования и удаления заявки.

Интерфейс панели управления обеспечивает удобный доступ к основным функциям системы, включая создание и управление заявками, а также фильтрацию и сортировку данных для облегчения поиска и организации информации.

На этой страницы пользователю позволяется создать заявку. На рисунке 24 представлена форма создания заявки на рабочей странице.

Форма содержит оборудование, которое указывают в заявку, а также описание проблемы, которую необходимо решить.

Форма спроектирована как модальное окно. Оборудование пользователь выбирает из имеющихся в базе данных или добавляет своё собственное.

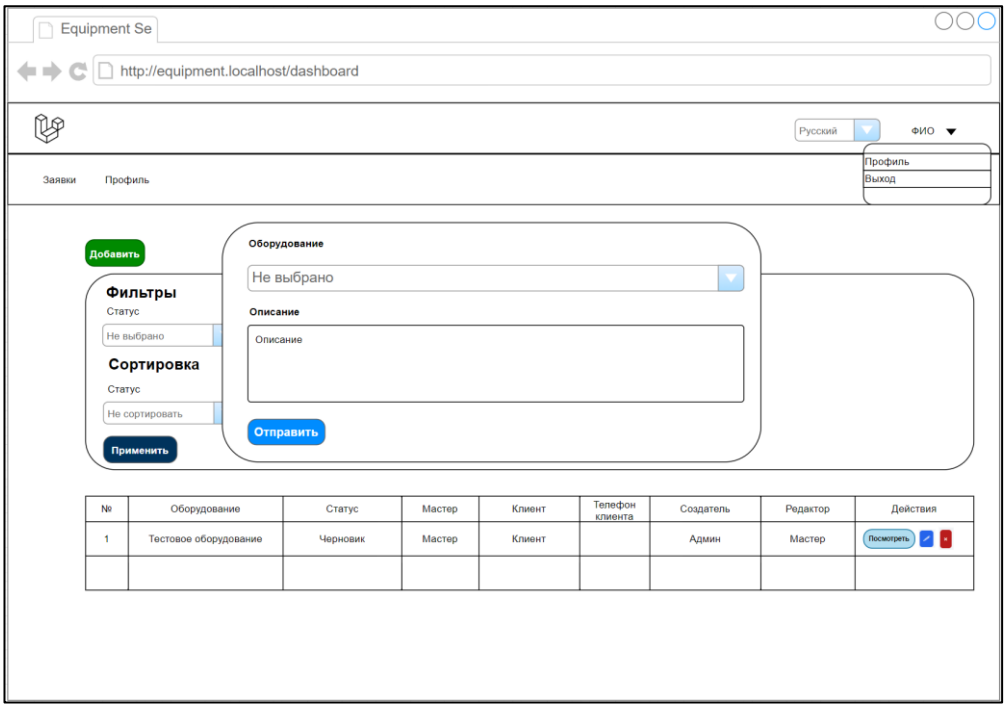

Рисунок 24 – Прототип формы создания заявки на рабочей страницы

Также для администратора и сотрудника спроектирован раздел «Заявки», в котором пользователь сможет манипулировать данными раздела. Так на рисунке 25 представлен прототип формы создания заявки.

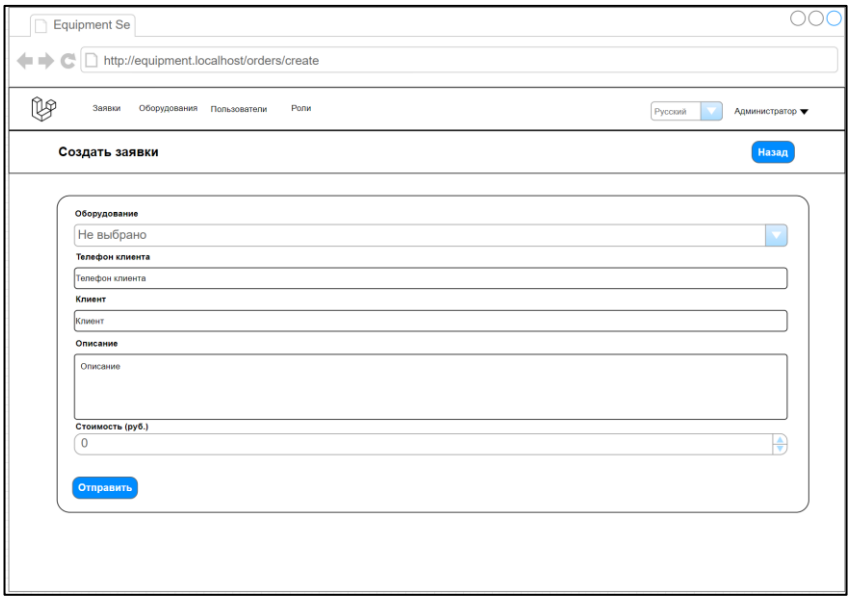

Рисунок 25 – Прототип формы создания заявки

Основное содержимое страницы посвящено форме для создания новой заявки. В верхней части формы находится заголовок "Создать заявку". Под заголовком расположены несколько полей ввода, предназначенных для заполнения информации о заявке.

Первое поле предназначено для выбора оборудования из выпадающего списка, который по умолчанию установлен на "Не выбрано". Ниже находится поле для ввода телефона клиента, затем поле для ввода имени клиента. Далее идет большое текстовое поле для ввода описания заявки, в котором пользователь может подробно описать проблему или требуемую услугу. Последнее поле позволяет указать стоимость услуги в рублях.

Под полями ввода располагаются две кнопки: кнопка "Назад", расположенная справа и позволяющая вернуться на предыдущую страницу, и кнопка "Отправить" внизу формы, которая отправляет введенные данные для создания новой заявки в системе.

На рисунке 26 представлена страница редактирования заявки в соответствующем разделе.

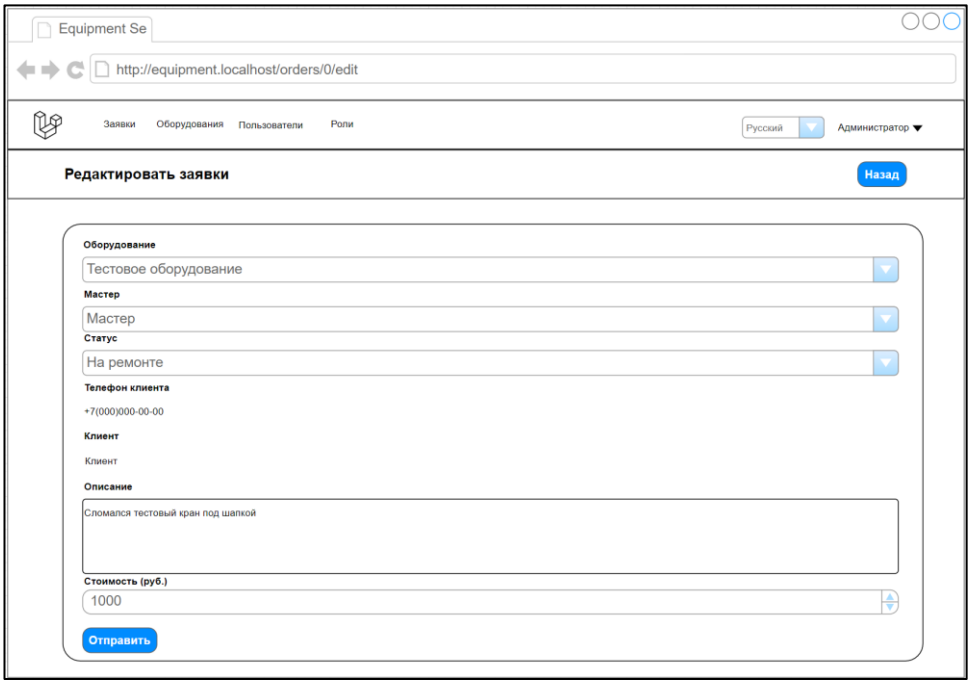

Рисунок 26 – Прототип формы редактирования заявки

Форма реализована аналогично форме создания заявки.

Администратору и Сотруднику также необходимы страницы для управления оборудованием. Так на рисунке 27 представлен прототип формы добавления оборудования.

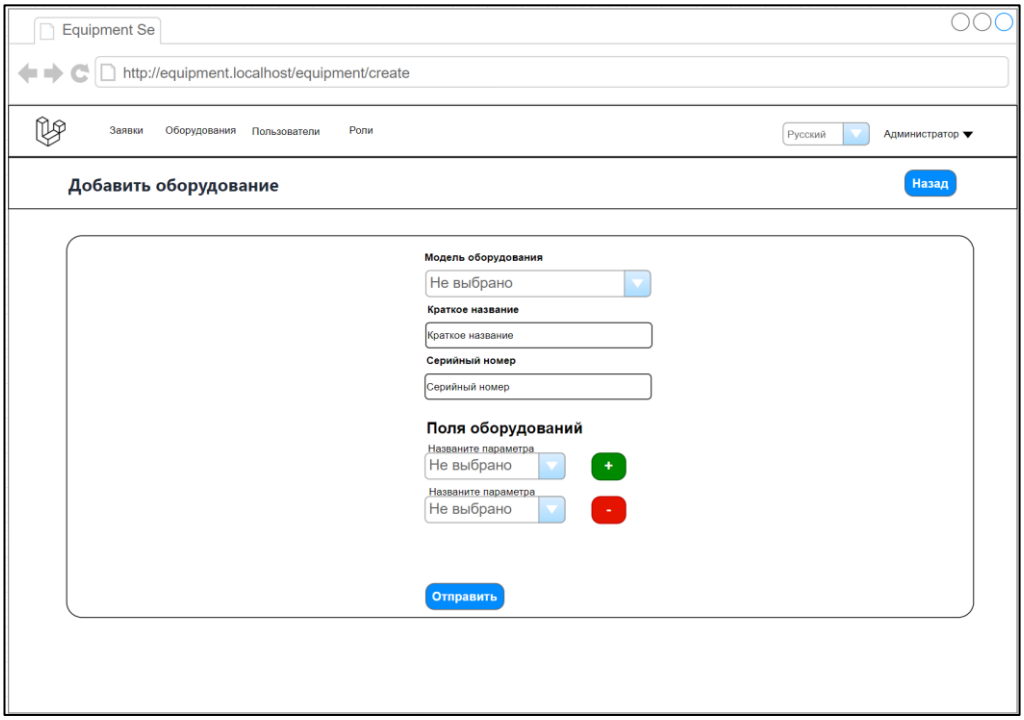

Рисунок 27 – Прототип формы добавления оборудования

Основное содержимое страницы начинается с заголовка "Добавить оборудование". Под заголовком располагается форма для ввода информации о новом оборудовании.

Первое поле предназначено для выбора модели оборудования из выпадающего списка, который по умолчанию установлен на "Не выбрано". Далее следуют текстовые поля для ввода краткого названия и серийного номера оборудования.

Под этими полями расположен раздел для добавления дополнительных полей оборудования. В этом разделе пользователь может выбрать название параметра из выпадающего списка и добавить его, нажав на зеленую кнопку

с плюсом. Если параметр нужно удалить, используется красная кнопка с минусом.

В правом верхнем углу формы находится кнопка "Назад", позволяющая вернуться на предыдущую страницу. В нижней части формы расположена кнопка "Отправить", которая отправляет введенные данные для сохранения нового оборудования в системе.

На рисунке 28 представлен прототип формы редактирования оборудования.

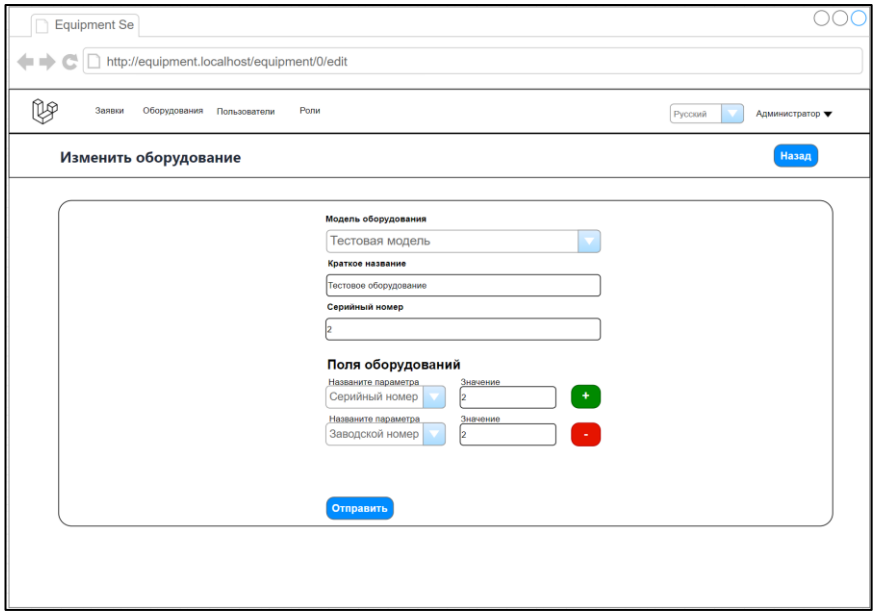

Рисунок 28 – Прототип формы редактирования оборудования

Форма реализована аналогично форме редактирования оборудования.

Выводы по главе 2

Во второй главе были выбраны технологии для логического моделирования веб-сервиса, такие как draw.io и dbdiagram.io.

Были спроектированы: контекстная диаграмма, диаграмма декомпозиции, диаграмма вариантов использования, диаграмма деятельности, диаграмма развертывания.

Были определены нормативно-справочные данные, необходимые входные сущности и атрибуты, а также выходная информация. Для хранения данных была выбрана реляционная модель базы данных, которая позволит хранить данные в виде таблиц, что упростит её понимание. По предметной области, по всем определенным требованиям была спроектирована концептуальная, логическая, физическая модели данных. Физическая модель спроектирована так, что соответствует третьей нормальной форме.

Также были описаны спроектированные пользовательские интерфейсы, которые позволят пользователю удобно и успешно взаимодействовать с разрабатываемым веб-сервисом.

Глава 3 Разработка функциональности веб-сервиса

3.1 Реализация функциональности веб-сервиса

Архитектура программного обеспечения (ПО) – это высокоуровневое описание компонентов и связей между ними, которое определяет общую структуру, принципы и правила организации ПО. Цель архитектуры ПО – обеспечить эффективность, гибкость, масштабируемость и надежность ПО.

Для реализации веб-сервиса была выбрана клиент-серверная архитектура программного продукта.

«Клиент – сервер» – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

Клиенты и серверы обмениваются сообщениями в шаблоне запрос-ответ. Клиент отправляет запрос, а сервер возвращает ответ. Этот обмен сообщениями является примером межпроцессного взаимодействия. Для взаимодействия компьютеры должны иметь общий язык, и они должны следовать правилам, чтобы и клиент, и сервер знали, чего ожидать. Язык и правила общения определены в протоколе связи. Все протоколы клиент-серверной модели работают на уровне приложений. Протокол прикладного уровня определяет основные шаблоны диалога.

Сервер может получать запросы от множества различных клиентов за короткий период времени. Компьютер может выполнять только ограниченное количество задач в любой момент и полагается на систему планирования для определения приоритетов входящих запросов от клиентов для их удовлетворения. Чтобы предотвратить злоупотребления и максимизировать доступность серверное программное обеспечение может ограничивать доступность для клиентов. Атаки типа «отказ в обслуживании» используют обязанности сервера обрабатывать запросы, такие атаки

действуют путем перегрузки сервера чрезмерной частотой запросов. Шифрование следует применять, если между клиентом и сервером должна передаваться конфиденциальная информация.

На рисунке 29 представлена клиент-серверная архитектура.

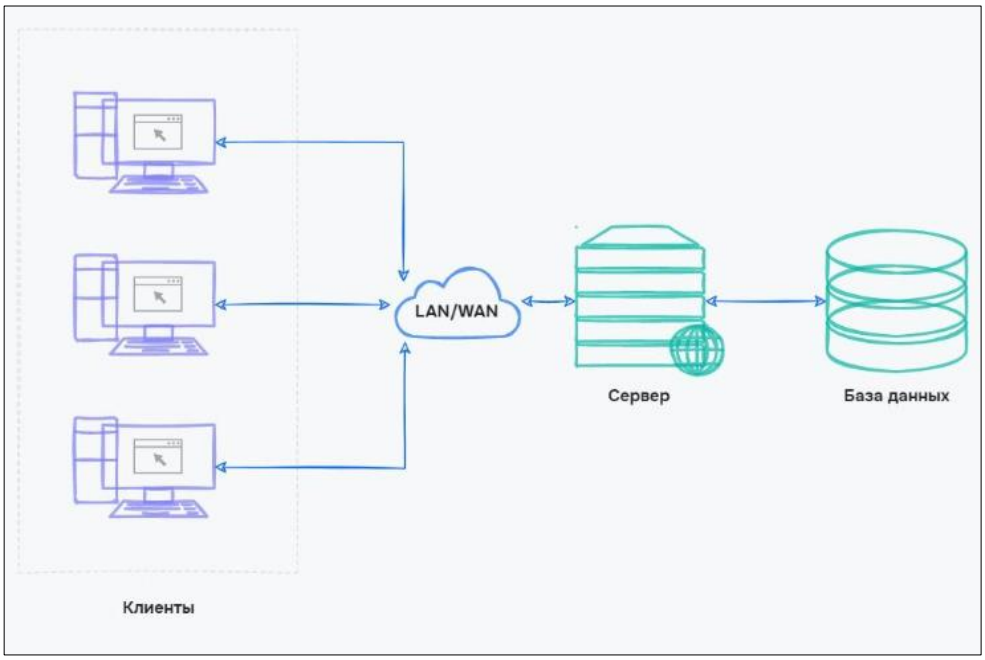

Рисунок 29 – Модель клиент-серверная архитектура

Плюсы клиент-серверной архитектуры:

– Отсутствие дублирования кода программы-сервера программами-клиентами.

– Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.

– Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще организовать контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа.

Минусы клиент-серверной архитектуры:

− Неработоспособность сервера может сделать неработоспособной всю вычислительную сеть.

Неработоспособным сервером следует считать сервер, производительности которого не хватает на обслуживание всех клиентов, а также сервер, находящийся на ремонте, профилактике и т. п.

− Поддержка работы данной системы требует отдельного специалиста – системного администратора.

− Высокая стоимость оборудования.

Для реализации веб-сервиса необходимо придерживаться браузерной разработки, связи с чем были выявлены преимущества данной технологии:

− Доступность и кроссплатформенность.

Браузерная разработка обеспечивает доступ к информационной системе с любого устройства, имеющего веб-браузер, включая компьютеры, планшеты и смартфоны. Это значительно расширяет круг потенциальных пользователей, позволяя клиентам и сотрудникам сервиса работать с системой без необходимости установки специализированного программного обеспечения. Кроссплатформенность браузерных приложений упрощает управление доступом и использование системы в различных операционных системах, будь то Windows, macOS, Linux, iOS или Android.

− Удобство обновлений и поддержки.

Обновления и поддержка браузерных приложений значительно упрощаются по сравнению с настольными или мобильными приложениями. Поскольку все данные и логика находятся на сервере, любые изменения или улучшения могут быть мгновенно доступны всем пользователям без необходимости обновления клиентской части. Это снижает затраты на поддержку и уменьшает вероятность того, что пользователи будут работать с устаревшей версией системы.

− Централизованное управление данными.

Браузерные приложения обычно работают по модели клиент-сервер, где серверная часть обрабатывает все данные и выполняет бизнес-логику, а

клиентская часть отвечает за представление данных и взаимодействие с пользователем. Это позволяет централизованно управлять данными, обеспечивая их целостность и безопасность.

− Высокая масштабируемость.

Браузерные приложения легко масштабируются. Серверная архитектура позволяет добавлять новые серверы для обработки возрастающего количества запросов без необходимости значительных изменений в клиентской части.

− Интеграция с другими системами.

Браузерные приложения легко интегрируются с другими веб-сервисами и API. Это позволяет расширить функциональность системы, добавляя новые возможности без значительных усилий. Например, можно интегрировать систему с платежными сервисами, службами уведомлений или сторонними CRM-системами для улучшения качества обслуживания клиентов и автоматизации бизнес-процессов.

− Улучшенный пользовательский опыт.

Современные браузерные технологии, такие как HTML5, CSS3 и JavaScript, позволяют создавать богатые и интерактивные пользовательские интерфейсы. Использование фреймворков и библиотек, таких как Laravel с Blade для серверной части и Tailwind CSS для стилизации, обеспечивает создание интуитивно понятных и отзывчивых интерфейсов. Это улучшает пользовательский опыт и повышает удовлетворенность клиентов, что особенно важно для информационной системы, связанной с ремонтом и обслуживанием бытовой техники.

Выбор браузерной разработки обоснован её доступностью, кроссплатформенностью, удобством обновлений и поддержки, централизованным управлением данными, высокой масштабируемостью, возможностью интеграции с другими системами и улучшенным пользовательским опытом. Эти преимущества делают браузерное приложение наиболее подходящим решением для создания эффективной и

удобной информационной системы, отвечающей потребностям как клиентов, так и сотрудников сервиса.

Выбор серверного языка программирования и фреймворка имеет значительное влияние на успешность разработки и эксплуатации информационной системы. В случае разрабатываемого веб-сервиса выбор PHP в сочетании с фреймворком Laravel предоставляет множество преимуществ, которые делают эту комбинацию идеальным выбором для реализации проекта.

PHP является одним из самых популярных серверных языков программирования в мире. Его популярность обусловлена рядом факторов, таких как простота в изучении и использовании, широкая поддержка со стороны хостинг-провайдеров и обширное сообщество разработчиков. Большое количество ресурсов, обучающих материалов и готовых решений на PHP значительно упрощают процесс разработки и сокращают время на обучение новых членов команды. Это особенно важно для проектов с ограниченными ресурсами и сроками.

Laravel, как один из самых популярных PHP-фреймворков, предоставляет разработчикам мощные инструменты и средства для создания веб-приложений. Laravel упрощает многие рутинные задачи, такие как маршрутизация, аутентификация, управление сессиями, кеширование и обработка запросов. Это позволяет сосредоточиться на бизнес-логике приложения, а не на решении низкоуровневых технических задач. Кроме того, Laravel следует принципам чистого кода и предоставляет удобную архитектуру, что облегчает поддержку и расширение системы.

Одним из главных преимуществ Laravel является его способность ускорять процесс разработки. Используя такие инструменты, как Laravel Artisan для автоматизации рутинных задач, Eloquent ORM для работы с базой данных и Blade для создания шаблонов, разработчики могут быстро создавать прототипы и реализовывать функциональность.

Laravel обеспечивает высокий уровень безопасности, предоставляя встроенные механизмы защиты от распространенных угроз, таких как SQL-инъекции, межсайтовый скриптинг (XSS) и подделка межсайтовых запросов (CSRF). Встроенные средства для управления аутентификацией и авторизацией пользователей помогают защитить конфиденциальные данные и обеспечить безопасность системы.

Laravel предлагает удобные инструменты для работы с базой данных, включая миграции, сидеры и Eloquent ORM. Это упрощает управление схемой базы данных и обеспечивает удобство взаимодействия с данными. Использование Eloquent ORM позволяет писать лаконичный и читаемый код для выполнения операций с базой данных, что ускоряет процесс разработки и уменьшает количество ошибок.

Для разработки пользовательского интерфейса использовались браузерные языки.

HTML – язык разметки гипертекста. Язык разметки дает браузеру необходимые инструкции о том, как отображать тексты и другие элементы страницы на мониторе. Язык HTML интерпретируется браузерами и отображается в виде документа, в удобной для человека форме.

CSS – каскадные таблицы стилей, которые используются для определения стилей (правил) оформления документов – включая дизайн, вёрстку и вариации макета для различных устройств и размеров экрана.

JavaScript – это полноценный динамический язык программирования, который применяется к HTML-документу и может обеспечить динамическую интерактивность. JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками.

Использование различных библиотек и фреймворков ускорит разработку. Так для CSS можно использовать утилитарный CSS-фреймворк,

который позволяет разработчикам быстро создавать адаптивные и стилизованные пользовательские интерфейсы - Tailwind CSS.

В отличие от традиционных CSS-фреймворков, таких как Bootstrap или Foundation, Tailwind CSS предоставляет набор низкоуровневых утилитарных классов, которые можно комбинировать для построения любых пользовательских интерфейсов без необходимости написания пользовательского CSS.

Tailwind CSS легко настраивается с помощью конфигурационного файла, где можно определить собственные цвета, размеры, шрифты и другие параметры. Эта гибкость позволяет адаптировать фреймворк под специфические требования проекта.

Выбранные технологии позволяют реализовывать красивые и качественный программные продукты.

Выбор системы управления базами данных (СУБД) является ключевым аспектом при разработке информационной системы, так как она отвечает за хранение, управление и доступ к данным.

Были рассмотрены следующие варианты реализации СУБД:

– MySQL.
– SQLite.
– PostgreSQL.

MySQL – свободная реляционная система хранения и управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

На сегодняшний день является самой популярной серверной базы данных (далее – БД), за счёт своей простоты, скорости работы и внушительного функционала. Поддерживаются такие основные движки MyISAM, InnoDB, MEMORY, Berkeley DB. Реализация всех новых возможностей стандарта SQL отсутствует в пользу простоты использования.

PostgreSQL – свободная объектно-реляционная система хранения и управления базами данных. Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

Работает только на одном движке – Storage Engine. Все таблицы представлены в виде объектов, они могут наследоваться, а все действия с таблицами выполняются с помощью объектно-ориентированных функций. Обладает открытым исходным кодом, разрабатывается командой энтузиастов, при этом старается максимально соответствовать стандарту SQL. Реализует все самые новые стандарты, что приводит к ущербу простоты, из-за чего PostgreSQL очень сложный и уступает в популярности MySQL.

SQLite – компактная встраиваемая СУБД с исходным кодом. Возможные типы значений: INTEGER, REAL, TEXT и BLOB. Также поддерживается специальное значение NULL. Размеры значений типа TEXT и BLOB не ограничены ничем, кроме константы SQLITE_MAX_LENGTH в исходниках sqlite, равной миллиарду.

SQLite напрямую хранит информацию в одном файле, что облегчает его копирование. Большая популярность в мобильной разработке и небольших автономных приложениях, поскольку она занимает меньше места на дисковом пространстве, имеет высокую скорость работы и не требует в отличии от MySQL не требует наличие сервера для запуска. Минусы: ограничения на запись, всего 5 типов данных, отсутствие встроенного механизма аутентификации.

MySQL зарекомендовала себя как высокопроизводительная СУБД, которая способна эффективно обрабатывать большие объемы данных и поддерживать высокие нагрузки.

MySQL является одной из самых популярных СУБД в мире, что обеспечивает её широкую поддержку и обширное сообщество пользователей и разработчиков. Это означает, что для MySQL доступно множество ресурсов, таких как документация, обучающие материалы, форумы и готовые решения, что значительно упрощает разработку, настройку и эксплуатацию системы. Более того, её популярность среди разработчиков гарантирует наличие специалистов, знакомых с MySQL, что облегчает процесс найма и обучения персонала.

Для наглядности сравнения вариантов реализации базы данных была составлена таблица 3.

Таблица 3 – Сравнение средств реализации базы данных

Параметр	MySQL	SQLite	PostgreSQL
Тип базы данных	Серверная	Встраиваемая	Серверная
Лицензия	GPL (с открытым исходным кодом)	В публичном достоянии	PostgreSQL License (с открытым исходным кодом)
Типы данных	Основные типы данных: INT, VARCHAR, TEXT, BLOB, DATE, DATETIME и т.д.	Основные типы данных: INTEGER, TEXT, BLOB, REAL и т.д.	Богатый набор типов данных, включая JSON, ARRAY, HSTORE
Поддержка JSON	Ограниченная	Ограниченная	Полная поддержка
Масштабируемость	Высокая	Низкая	Очень высокая
Сообщество и поддержка	Большое сообщество, коммерческая поддержка (Oracle)	Активное сообщество, обширная документация	Большое сообщество, коммерческая поддержка
Производительность	Высокая для чтения и записи	Высокая для небольших и встраиваемых приложений	Высокая, особенно для сложных запросов и больших данных

MySQL хорошо интегрируется с PHP и фреймворком Laravel, что упрощает разработку серверной части информационной системы. Laravel предоставляет удобные инструменты для работы с MySQL, такие как Eloquent ORM и миграции, что позволяет разработчикам легко создавать, модифицировать и управлять схемой базы данных. Эта интеграция снижает сложность разработки и ускоряет процесс создания функциональных модулей системы.

MySQL предоставляет множество встроенных механизмов безопасности, включая управление пользователями и правами доступа, шифрование данных и протоколов связи. Эти функции помогают защитить данные от несанкционированного доступа и атак, обеспечивая безопасность информации о клиентах и процессах обслуживания. Поддержка ролей и привилегий позволяет точно настраивать доступ пользователей к различным частям базы данных, что способствует защите конфиденциальной информации.

Благодаря использованию фреймворка Laravel в реализации веб-сервиса, то разрабатывать базу данных можно посредством языка программирования PHP и так называемых миграций.

Миграции в Laravel - это способ управления изменениями в структуре базы данных проекта с помощью кода. Они позволяют создавать и изменять таблицы, индексы, ограничения и другие элементы базы данных без использования SQL-запросов. Миграции облегчают развёртывание и поддержку структуры базы данных, а также синхронизацию изменений между участниками команды.

Ниже будут представлены код миграций основных таблиц базы данных веб-сервиса.

Код миграции создания таблицы «users» и сопутствующие ей системные таблицы «password_reset_tokens» и «sessions».

```php
<?php
// Подключение необходимых классов для реализации миграции
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
// Безименной класс, наследующий необходимую логику от класса Migration
return new class extends Migration
{
    /**
     * Run the migrations.
     * Метод up, который запускает создание таблиц и необходимых действий с
     * таблицами
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('phone', 16)->nullable();
            $table->string('login');
            $table->string('password');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('language', 3)->default('ru');
            $table->rememberToken();
            $table->timestamps();
            $table->softDeletes();
        });

        Schema::create('password_reset_tokens', function (Blueprint $table) {
            $table->string('email')->primary();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });

        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('language', 3)->nullable();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();

            $table->foreign('user_id')->references('id')->on('users');
        });
    }
    /**
     * Reverse the migrations.
     * Метод down, который выполняется для отката миграции
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
        Schema::dropIfExists('password_reset_tokens');
        Schema::dropIfExists('sessions');
    }
};
```

В представленном коде демонстрируется создание таблицы пользователей с необходимыми атрибутами, такими как логин, пароль, фио, почта.

Код миграции таблицы «equipment_orders», в которой будут хранится все необходимые данные о заявках на бытовое оборудование.

```
<?php
// Подключение необходимых классов для реализации миграции
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
// Безименной класс, наследующий необходимую логику от класса Migration
return new class extends Migration
{
    /**
     * Run the migrations.
     * Метод up, который запускает создание таблиц и необходимых действий с
     * таблицами
     */
    public function up(): void
    {
        Schema::create('equipment_orders', function (Blueprint $table) {
            $table->id();
            $table->foreignId('equipment_id');
            $table->foreignId('master_id')->nullable();
            $table->string('status_code', 20);
            $table->text('description');
            $table->decimal('price');
            $table->string('phone')->nullable();
            $table->string('client_name')->nullable();
            $table->foreignId('client_id')->nullable();
            $table->foreignId('creator_id');
            $table->foreignId('editor_id')->nullable();
            $table->timestamps();
            $table->softDeletes();

            $table->foreign('equipment_id')->on('equipment')-
>references('id');
            $table->foreign('creator_id')->on('users')->references('id');
            $table->foreign('editor_id')->on('users')->references('id');
            $table->foreign('master_id')->on('users')->references('id');
            $table->foreign('client_id')->on('users')->references('id');
            $table->foreign('status_code')->on('equipment_orders_statuses')-
>references('code');
        });
    }

    /**
     * Reverse the migrations.
     * Метод down, который выполняется для отката миграции
     */
    public function down(): void
    {
        Schema::dropIfExists('equipment_orders');
    }
};
```

В представленном коде демонстрируется создание таблицы заявок со всеми необходимыми атрибутами. Видно, что таблица имеет связи с другими таблицами: с таблицей оборудований, пользователей и статусов заявок.

По итогу выполнения всех разработанных миграций будет создана база данных согласно физической модели данных

Модель демонстрирует 22 таблицы различного назначения. Для хранения данных о заявках, оборудованиях, пользователях, их ролях и правах доступа. Также имеется таблица «migrations», в которой и хранятся все выполненные миграции, чтобы фреймворк знал, какие миграции уже выполнялись, а какие ещё нет.

Веб-сервис реализовался на фреймворке Laravel из-за чего реализует архитектурный шаблон MVC.

Модель MVC (Model-View-Controller) - это архитектурный шаблон, который разделяет приложение на три основные части: Модель (Model), Представление (View) и Контроллер (Controller). Этот шаблон используется для разделения бизнес-логики, пользовательского интерфейса и ввода управления, чтобы обеспечить четкую структуру и улучшить поддержку и расширяемость кода.

Функцию подачи заявки на ремонт и обслуживание бытовой техники содержит контроллер «OrdersController», а конкретно метод «store».

```
/**
Метод store – обрабатывает сохранение данных с формы создания заявки
$request – полученные данные от пользователя
возвращает переадресацию на другой маршрут
**/

public function store(Request $request): RedirectResponse
{
// Валидация полученных данных на корректность
    $request->validate([
        'description' => ['required','string'],
        'price' => ['numeric'],
        'equipment_id' => ['required', 'integer',
Rule::exists(Equipment::class, 'id')],
    ]);
    $fields = [
        'equipment_id' => $request->post('equipment_id'),
        'description' => $request->post('description'),
        'price' => $request->post('price', 0),
        'creator_id' => Auth::id(),
    ];
    $route = 'orders.index';
// Определение логики сохранения данных
    if ($request->has('client_name')) {
        $request->validate([
            'client_name' => ['required','string'],
            'phone' => ['required','string'],
```

```
        ]);
        $fields['client_name'] = $request->post('client_name');
        $fields['phone'] = $request->post('phone');
        $fields['master_id'] = Auth::id();
        $fields['status_code'] = 'diagnostic';
    } else {
        $fields['client_id'] = Auth::id();
        $fields['status_code'] = 'draft';
        $route = 'dashboard';
    }
// Сохранение данных в базу
    EquipmentOrder::create($fields);
// Переадресация пользователя на необходимую страницу с информационным
сообщением
    return redirect()->route($route)
        ->with('success', __('orders.messages.store'));
}
```

Метод сохраняет полученные с формы данные в базу данных и возвращает пользователю информационное сообщение.

За работу с заявками в базе данных отвечает модель EquipmentOrder.

Модели в Laravel - это объекты таблиц базы данных. Они позволяют взаимодействовать с таблицами базы данных, как будто они являются объектами или классами PHP.

```
<?php
// Пространство имен класса, чтобы php понимал, к какому именно классу
// обращается разработчик
namespace App\Models\Orders;
// Подключение необходимых классов
use App\Models\Equipment\Equipment;
use App\Models\User;
use App\Traits\HistoryModelTrait;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\SoftDeletes;
use Illuminate\Support\Carbon;
use Panoscape\History\HasHistories;
// Объявление модели
class EquipmentOrder extends Model
{
// Подключение класса для «мягкого удаления» записей в базе данных
    use SoftDeletes;
// Класс для записи истории изменения модели
    use HasHistories;
// Класс для обработки истории, для преобразования в читаемый для человека
вид
    use HistoryModelTrait;
// Класс для создания «завода» тестовых данных
    use HasFactory;
// Указание названия таблицы, с которой работает модель
    protected $table = 'equipment_orders';
// Перечисление атрибутов, которые не нужно возвращать с таблицы
    protected $guarded = [];
// Подтверждение, что модели используются атрибуты даты создания, даты
изменения
```

```
// записи
    public $timestamps = true;
// Перечисление методов реализующие связь таблиц, для удобного получения
данных
    public function equipment(): BelongsTo
    {
        return $this->belongsTo(Equipment::class, 'equipment_id', 'id')-
>withTrashed();
    }

    public function status(): BelongsTo
    {
        return $this->belongsTo(OrdersStatus::class, 'status_code', 'code')
            ->where('language', '=', app()->getLocale());
    }

    public function client(): BelongsTo
    {
        return $this->belongsTo(User::class, 'client_id')->withTrashed();
    }

    public function creator(): BelongsTo
    {
        return $this->belongsTo(User::class, 'creator_id')->withTrashed();
    }

    public function editor(): BelongsTo
    {
        return $this->belongsTo(User::class, 'editor_id')->withTrashed();
    }

    public function master(): BelongsTo
    {
        return $this->belongsTo(User::class, 'master_id')->withTrashed();
    }
// Указание под каким названием записывать изменения в истории у модели
    public function getModelLabel(): string
    {
        return '"' . $this->equipment->short_name . ' ' . $this->equipment-
>serial . ' ' . Carbon::parse($this->created_at)->format('d.m.Y H:i') . '"';
    }
// Преобразование атрибутов таблицы из истории, в читаемый для человека вид
// с учетом возможной локализации
    public function historyMetaFields(): array
    {
        return [
            'equipment_id' => [
                'name' => __('equipment.headers.main.single'),
                'table' => [
                    'name' => 'equipment',
                    'value' => 'id',
                    'label' => 'short_name'
                ],
            ],
            'master_id' => [
                'name' =>   ('orders.fields.master'),
                'table' => [
                    'name' => 'users',
                    'value' => 'id',
                    'label' => 'name',
                ]
            ],
            'client_id' => [
```

```
                'name' => __('orders.fields.client'),
                'table' => [
                    'name' => 'users',
                    'value' => 'id',
                    'label' => 'name',
                ]
            ],
            'client_name' => [
                'name' => __('orders.fields.client'),
            ],
            'phone' => [
                'name' => __('orders.fields.phone'),
            ],
            'status_code' => [
                'name' => __('orders.fields.status'),
                'table' => [
                    'name' => 'equipment_orders_statuses',
                    'value' => 'code',
                    'label' => 'name',
                    'language' => true,
                ]
            ],
            'description' => [
                'name' => __('orders.fields.description'),
            ],
        ];
    }
}
```

Так в модели указаны методы для связи с другими таблицами, а из-за наследования от родительского класса Model других методов, модель может работать с указанной в нем таблице.

Для реализации функций создания и редактирования заявок был разработан контроллер «OrdersController», модель «EquipmentOrder», представления и необходимые маршруты.

```
<?php
/** Подключение используемых классов
OrdersController – класс контроллер, в котором определена логика работы с
данными о заявках
Route – класс фреймворка, отвечающий за маршрутизацию

**/
use App\Http\Controllers\OrdersController;
use Illuminate\Support\Facades\Route;
/**
Объявление основных маршрутов для вызова методов по заявкам
**/
Route::middleware('auth')->group(function () {
    Route::resource('orders', OrdersController::class)->names('orders');
    Route::patch('orders/{id}/recovery', [OrdersController::class,
'recovery'])->whereNumber('id')->name('orders.recovery');
});
Route::get('/orders/{id}/status/{status_code}', [OrdersController::class,
'changeStatus'])->whereNumber('id')->name('orders.status');
```

В маршрутах указаны все необходимы пути, которые может использовать пользователь веб-сервиса для работы с заявками.

Далее представлен фрагмент кода контроллера «OrdersController», который демонстрирует логику работы функций создания заявки в веб-сервисе.

```
/**
Метод create – возвращает пользователю представление с формой создания заявки
Возвращает представление
**/
public function create(): Response
{
    return response()->view('orders.create', [
        'order_statuses_select' => OrdersStatus::autocomplete(),
        'users_select' => User::autocomplete(),
        'equipment_select' => Equipment::autocomplete(),
    ]);
}

/**
Метод store – обрабатывает сохранение данных с формы создания заявки
$request – полученные данные от пользователя
возвращает переадресацию на другой маршрут
**/

public function store(Request $request): RedirectResponse
{
// Валидация полученных данных на корректность
    $request->validate([
        'description' => ['required','string'],
        'price' => ['numeric'],
        'equipment_id' => ['required', 'integer',
Rule::exists(Equipment::class, 'id')],
    ]);
    $fields = [
        'equipment_id' => $request->post('equipment_id'),
        'description' => $request->post('description'),
        'price' => $request->post('price', 0),
        'creator_id' => Auth::id(),
    ];
    $route = 'orders.index';
// Определение логики сохранения данных
if ($request->has('client_name')) {
        $request->validate([
            'client_name' => ['required','string'],
            'phone' => ['required','string'],
        ]);
        $fields['client_name'] = $request->post('client_name');
        $fields['phone'] = $request->post('phone');
        $fields['master_id'] = Auth::id();
        $fields['status_code'] = 'diagnostic';
    } else {
        $fields['client_id'] = Auth::id();
        $fields['status_code'] = 'draft';
        $route = 'dashboard';
    }
// Сохранение данных в базу
    EquipmentOrder::create($fields);
// Переадресация пользователя на необходимую страницу с информационным
сообщением
    return redirect()->route($route)
        ->with('success', __('orders.messages.store'));
}
```

Также основной из функций веб-сервиса является редактирование заявки. Код данных методов:

```
/**
Метод edit – метод, отвечающий за открытие формы редактирования заявки
Ожидает числовой параметр $id – идентификатор заявки
Возвращает переадресацию, если у пользователя недостаточно прав, или
представление, в которой отображается форма редактирования заявки
**/
public function edit(int $id): RedirectResponse|Response
{
// Получение данных заявки
    $order = EquipmentOrder::query()->with([
        'equipment.model.type',
        'equipment.model.brand',
        'client',
        'creator',
        'editor',
        'master',
        'status'
    ])->withTrashed()->find($id);
// Проверка наличия прав на редактирования
    if (
        Auth::user()->can('equipment_orders_my_edit')
        && !Auth::user()->can('equipment_orders_edit')
        && $order->creator_id !== Auth::id()
        && $order->master_id !== Auth::id()
    ) {
// Переадресация пользователя
        return redirect()->route('dashboard');
    }
// Представление
    return response()->view('orders.edit', [
        'order' => $order,
        'order_statuses_select' => OrdersStatus::autocomplete(),
        'users_select' => User::autocomplete(),
        'equipment_select' => Equipment::autocomplete(),
    ]);
}

/**
Метод update – метод, отвечающий за сохранение данных при изменении заявки
Ожидает числовой параметр $id – идентификатор заявки
$request - полученные данные от пользователя
возвращает переадресацию на другой маршрут
**/

public function update(Request $request, int $id): RedirectResponse
{
// Валидация полученных данных на корректность
    $request->validate([
        'description' => ['required','string'],
        'price' => ['required', 'numeric'],
        'equipment_id' => ['required', 'integer',
Rule::exists(Equipment::class, 'id')],
        'status_code' => ['required', 'string',
Rule::exists(OrdersStatus::class, 'code')],
    ]);
// Формирование сохраняемых данных
    $fields = [
        'equipment_id' => $request->post('equipment_id'),
        'description' => $request->post('description'),
```

```
            'price' => $request->post('price'),
            'editor_id' => Auth::id(),
            'status_code' => $request->post('status_code'),
        ];
// Определение логики сохранения данных
        if ($request->has('client_name')) {
            $request->validate([
                'client_name' => ['required','string'],
                'phone' => ['required','string'],
            ]);
            $fields['client_name'] = $request->post('client_name');
            $fields['phone'] = $request->post('phone');
            $fields['master_id'] = Auth::id();
        } elseif ($request->has('client_id')) {
            $fields['client_id'] = $request->post('client_id');
        }
        $oldOrder = EquipmentOrder::query()->with(['client'])->find($id);
        EquipmentOrder::updateOrCreate(['id' => $id], $fields);
// Определение необходимости отправки уведомления клиенту
        $order = EquipmentOrder::query()->with(['client', 'equipment.model.type',
'equipment.model.brand',])->find($id);
        if ($oldOrder->status_code !== $order->status_code && !empty($order-
>client)) {
// Определение какое именно необходимо отправить уведомление
            $mailTemplate = match ($order->status_code) {
                'approval' => new ApprovalRepair($order),
                'canceled' => new CanceledRepair($order),
                'closed' => new ClosedRepair($order),
                'completed' => new CompletedRepair($order),
                default => null,
            };
            if (!empty($mailTemplate)) {
// Отправка уведомления на почту клиента
                Mail::to($order->client->email)->send($mailTemplate);
            }
        }
// Переадресация пользователя на необходимую страницу с информационным
сообщением
        return redirect()->route('orders.index')
            ->with('success', __('orders.messages.update'));
}
```

Представленный фрагмент открывает форму создания заявки пользователю, а также обрабатывает отправленные данные с этой формы, после чего заявка сохраняется, а при случае некорректных данных возвращается информационное сообщение пользователю.

Также в контроллере имеются методы по удалению, восстановлению и отправке клиенту уведомления, что статус заявки изменен.

```
// Метод, который вызывает мягкое удаление заявки
// $id - идентификатор заявки
public function destroy(int $id): RedirectResponse
{
// Поиск заявки по переданном id
    $order = EquipmentOrder::withTrashed()->find($id);
// Изменяется статус заявки на «Удалено»
    $order->update([
```

```
            'status_code' => 'deleted',
        ]);
    // Вызов мягкого удаления
        $order->delete();
    // Возращение пользователю информационного сообщения
        return redirect()->route('orders.index')
            ->with('success', __('orders.messages.delete'));
    }
    // Метод, восстановления заявки
    // $id - идентификатор заявки

    public function recovery(int $id): RedirectResponse
    {
    // Поиск заявки по переданном id
        $order = EquipmentOrder::withTrashed()->find($id);
    // Восстановление заявки
        $order->restore();
    // Смена статуса заявки на «Черновик»
        $order->update([
            'status_code' => 'draft',
        ]);
    // Возращение пользователю информационного сообщения
        return redirect()->route('orders.show', $id)
            ->with('success', __('orders.messages.recovery'));
    }
    // Метод, отправки клиенту на почту сообщения о смене статуса
    // $orderID - идентификатор заявки
    // $statusCode - код нового статуса заявки
    public function changeStatus(
        int $orderID,
        string $statusCode
    ): RedirectResponse {
    // Поиск заявки по переданном orderID
        $order = EquipmentOrder::query()->find($orderID);
    // Проверка, что такая заявка существует и она не удалена
        if (!empty($order)) {
    // Проверка, что старый статус заявки соответствует, тому который должен был
    // быть
            $checkOldStatus = match ($statusCode) {
                'signed', 'canceled' => 'approval',
            };
            if ($checkOldStatus === $order->status_code) {
                $order->update(['status_code' => $statusCode]);
            }
    // Определение какой именно шаблон нужно отправить клиент
            $mailTemplate = match ($statusCode) {
                'approval' => new ApprovalRepair($order),
                'canceled' => new CanceledRepair($order),
                'closed' => new ClosedRepair($order),
                'completed' => new CompletedRepair($order),
                default => null,
            };
    // Отправка сообщения
            if (!empty($mailTemplate) && !empty($order->client)) {
                Mail::to($order->client->email)->send($mailTemplate);
            }
        }

        return redirect()->route('welcome');
    }
```

Так метод «destroy» «мягко» удаляет заявку из базы данных, что позволяет восстановить заявку при необходимости.

Мягкое удаление - это режим, который позволяет пометить данные как удалённые без фактического удаления их из базы данных. Это снижает риск потери данных и позволяет быстро сделать некоторые записи недоступными для пользователей.

В представлениях реализован спроектированный пользовательский интерфейс, с которым пользователь может взаимодействовать для выполнения необходимых функций.

Код представления по созданию заявки:

```
{{--
Подключение основного шаблона интерфейса, с меню навигации
--}}
<x-app-layout>
{{--
Указание заголовка страницы
--}}

    <x-slot name="header">
        <div class="mb-5">
            <div class="float-left">
{{--
Указание мультиязычного название раздела Создать заявку
--}}
                <h2 class="font-semibold text-xl text-gray-800 leading-
tight">{{ __('orders.headers.create') }}</h2>
            </div>
            <div class="float-right">
                <x-a href="{{ route('orders.index') }}">{{ __('actions.back')
}}</x-a>
            </div>
        </div>
    </x-slot>
{{--
Отображение информационного сообщения об ошибке
--}}
    @if (count($errors) > 0)
        <div class="w-full px-10 py-5 bg-red-700">
            <strong>{{ __('validation.whoops') }}</strong>
            <ul>
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
    @endif
    <div class="container mx-auto my-5 bg-gray-50 rounded">
        <div class="py-5 mx-5">
{{--
Форма для создания заявки со всеми необходимыми полями
--}}
            <form action="{{route('orders.store')}}" method="POST">
                @csrf
                <div class="my-5 mx-5">
                    <div class="flex flex-col">
                        <x-input-label
```

```
                            for="equipment_id"
                            :value="__('equipment.headers.main.single')"
                        />
                        <x-select
                            id="equipment_id"
                            name="equipment_id"
                            class="mt-1"
                            :data="$equipment_select"

:placeholder="__('equipment.headers.main.single')"
                        />
                    </div>
                    <div class="flex flex-col">
                        <x-input-label
                            for="phone"
                            :value="__('orders.fields.phone')"
                        />
                        <x-text-input type="text" id="phone" name="phone"

:placeholder="__('orders.fields.phone')"></x-text-input>
                    </div>
                    <div class="flex flex-col">
                        <x-input-label
                            for="client_name"
                            :value="__('orders.fields.client')"
                        />
                        <x-text-input type="text" name="client_name"

:placeholder="__('orders.fields.client')"></x-text-input>
                    </div>
                    <div class="flex flex-col">
                        <x-input-label
                            for="description"
                            :value="__('orders.fields.description')"
                        />
                        <x-textarea
                            id="description"
                            name="description"
                            class="mt-1"
                            :placeholder="__('orders.fields.description')"
                        ></x-textarea>
                    </div>
                    <div class="flex flex-col">
                        <x-input-label
                            for="price"
                            :value="__('orders.fields.price')"
                        />
                        <x-text-input
                            id="price"
                            name="price"
                            type="number"
                            class="mt-1 block w-full"
                            min="0.00"
                            step="0.01"
                            :value="0.00"
                            :placeholder="  ('orders.fields.price')"
                            required
                        />
                    </div>
                    <div class="mt-2">
                        <x-btn type="submit">{{ __('actions.submit') }}</x-
btn>
                    </div>
```

```
                </div>
            </form>
        </div>
    </div>
</x-app-layout>
{{--
Подключение библиотеки для валидации номера телефона
--}}
<script src="{{asset('js/imask.js')}}"></script>
<script>
    IMask(
        document.getElementById('phone'),
        {
            mask: '+{7}(000)000-00-00'
        }
    )
</script>
```

На рисунке 30 представлен результат выполнения представленного кода. Пользователю открывается форма создания заявки.

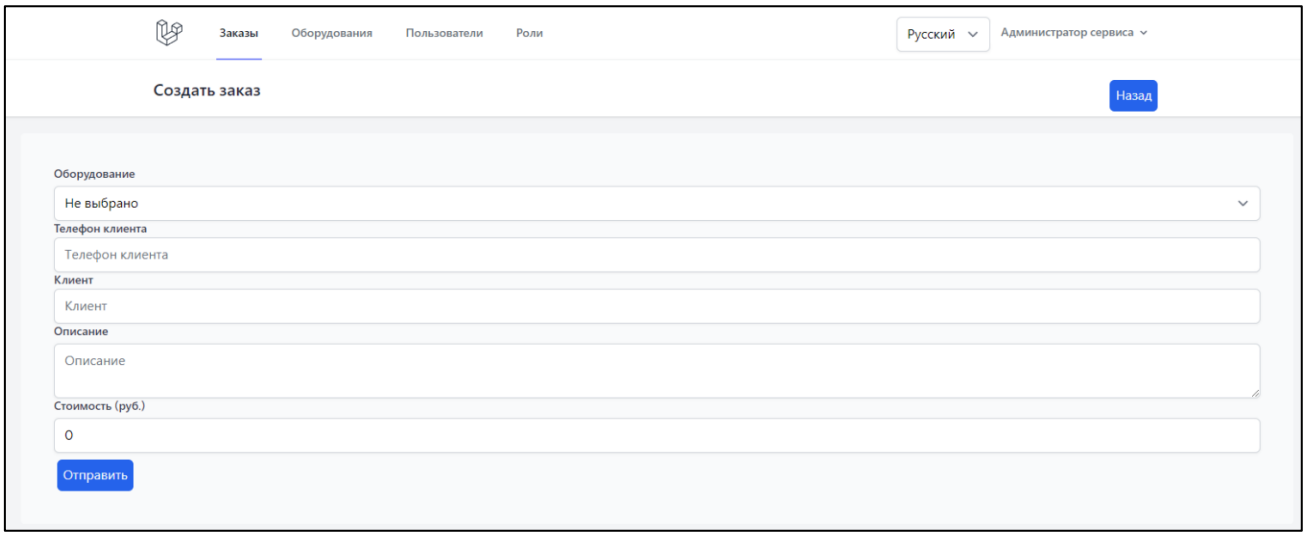

Рисунок 30 – Форма создания заявки

Также на этой форме выводятся сообщения, если по результату отправки заполненной формы данные оказываются не корректными. На рисунке 31 представлен пример информационного сообщения на форме создания заявки.

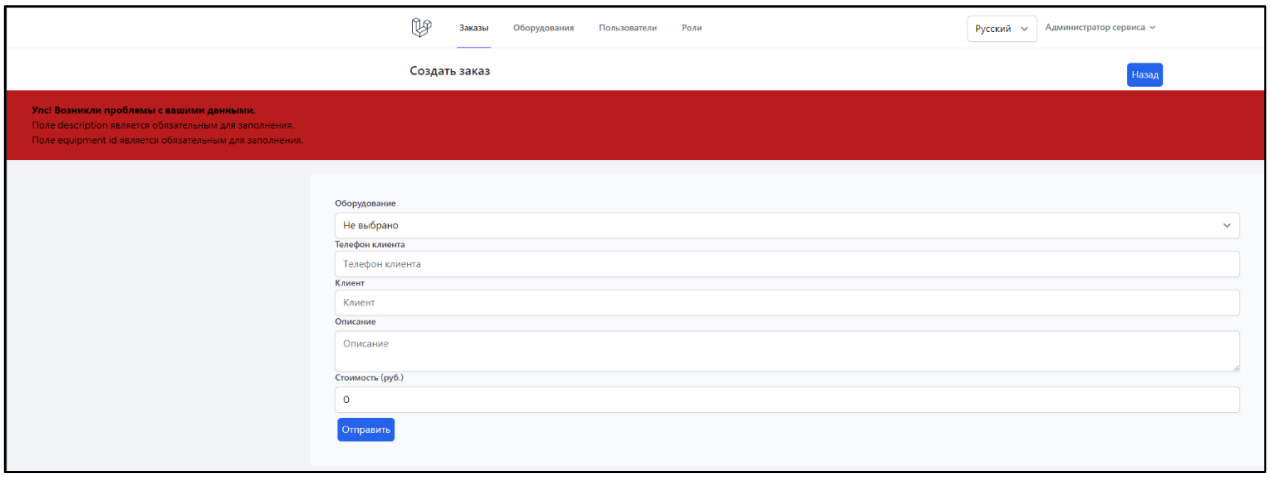

Рисунок 31 – Информационное сообщение на форме создания заявки

Далее представлен фрагмент кода формы редактирования заявки. Код демонстрирует отрисовку формы, её заполнение данными из базы, а также реализацию многоязычности веб-сервиса.

```
<div class="container mx-auto my-5 bg-gray-50 rounded">
    <div class="py-5 mx-5">
{{-- Форма для редактирования заявки--}}
        <form action="{{route('orders.update', $order->id)}}" method="POST">
            @csrf
            @method('PATCH')
            <div class="my-5 mx-5">
{{-- Поле для указания оборудования--}}
                <div class="flex flex-col">
{{-- Мультиязычное название поля/ей--}}
                    <x-input-label
                        for="equipment_id"
                        :value="__('equipment.headers.main.single')"
                    />
                    <x-select
                        id="equipment_id"
                        name="equipment_id"
                        class="mt-1"
                        :data="$equipment_select"
                        :selected="$order->equipment_id"
                        :placeholder="__('equipment.headers.main.single')"
                    />
                </div>
{{-- Поле для указания мастера выполняющего заявку--}}
                <div class="flex flex-col">
                    <x-input-label
                        for="master_id"
                        :value="__('orders.fields.master')"
                    />
                    <x-select
                        id="master_id"
                        name="master_id"
                        class="mt-1"
                        :data="$users_select"
                        :selected="$order->master_id"
                        :placeholder="__('orders.fields.master')"
```

```
                    />
                </div>
{{-- Поле для указания статуса заявки--}}
                <div class="flex flex-col">
                    <x-input-label
                        for="status_code"
                        :value="__('orders.fields.status')"
                    />
                    <x-select
                        id="status_code"
                        name="status_code"
                        class="mt-1"
                        :data="$order_statuses_select"
                        :selected="$order->status_code"
                        :placeholder="__('orders.fields.status')"
                    />
                </div>
{{-- Поле для указания телефона клиента--}}
                <div class="flex flex-col">
                    <x-input-label
                        for="phone"
                        :value="__('orders.fields.phone')"
                    />
                    @if(empty($order->phone))
                        {{$order->client->phone}}
                    @else
                        <x-text-input type="text" id="phone" name="phone"
value="{{$order->phone}}" :placeholder="__('orders.fields.phone')"></x-text-
input>
                    @endif
                </div>
{{-- Поле для указания ФИО клиента--}}
                <div class="flex flex-col">
                    <x-input-label
                        for="client_name"
                        :value="__('orders.fields.client')"
                    />
                    @if(empty($order->client_name))
                        {{$order->client->name}}
                    @else
                        <x-text-input type="text" name="client_name"
value="{{$order->client_name}}"
:placeholder="__('orders.fields.client')"></x-text-input>
                    @endif
                </div>
                @if(!empty($order->client))
                    <input type="hidden" name="client_id" value="{{$order-
>client_id}}">
                @endif
{{-- Поле для указания описания проблемы в оборудовании--}}
                <div class="flex flex-col">
                    <x-input-label
                        for="description"
                        :value="__('orders.fields.description')"
                    />
                    <x-textarea
                        id="description"
                        name="description"
                        class="mt-1"
                        :placeholder="__('orders.fields.description')"
                    >{{$order->description}}</x-textarea>
                </div>
```

```
{{-- Поле для указания стоимости к оплате--}}
                <div class="flex flex-col">
                    <x-input-label
                        for="price"
                        :value="__('orders.fields.price')"
                    />
                    <x-text-input
                        id="price"
                        name="price"
                        type="number"
                        class="mt-1 block w-full"
                        min="0.00"
                        step="0.01"
                        :value="$order->price"
                        :placeholder="__('orders.fields.price')"
                        required
                    />
                </div>
                <div class="mt-2">
                    <x-btn type="submit">{{ __('actions.submit') }}</x-btn>
                </div>
            </div>
        </form>
    </div>
</div>
```

На рисунке 32 и 33 демонстрируется форма редактирования заявки при разных выбранных языках. Так языковая часть интерфейса изменяется в зависимости от выбранного языка пользователем.

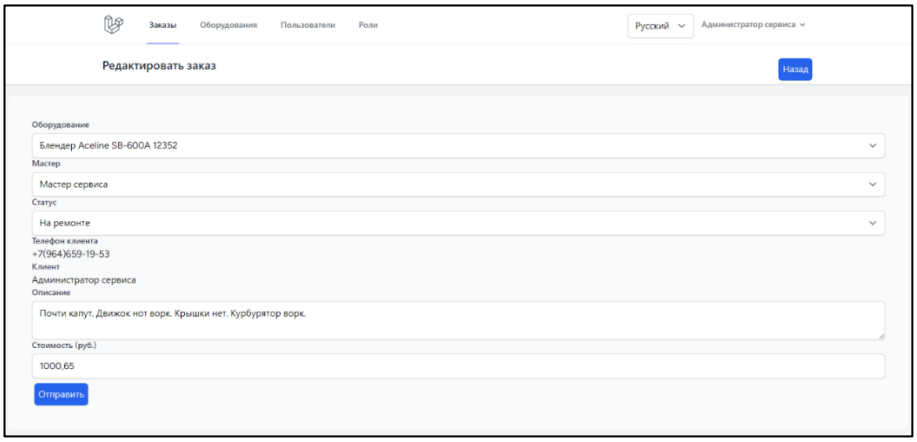

Рисунок 32 – Форма редактирования заявки на русском

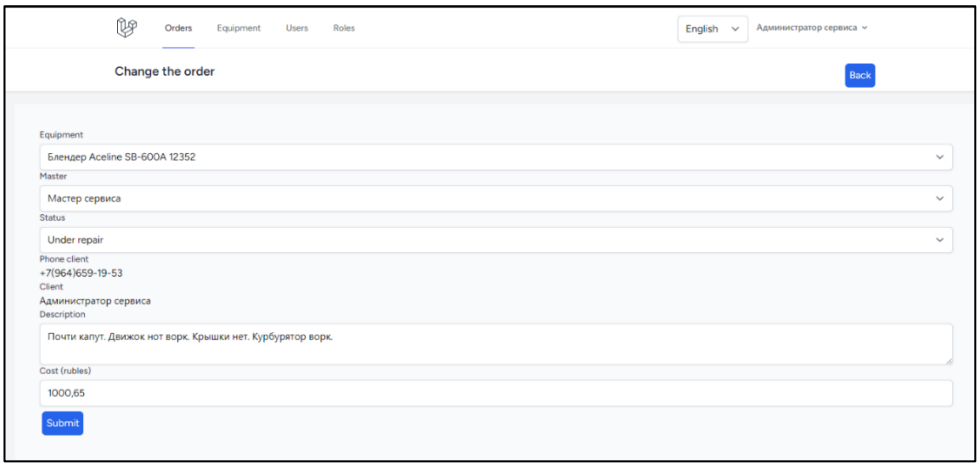

Рисунок 33 – Форма редактирования заявки на английском

Данная функциональность разработана для компаний и пользователей, которые ориентированы на другой язык.

При положительном ответе на редактирование или добавление данных пользователю выводится соответствующее сообщение, которое представлено на рисунке 34.

Рисунок 34 – Пример сообщения о положительном результате

Все разработанные разделы имеют схожую функциональность и внешний вид, что позволяет пользователю интуитивно понимать ожидаемый результат, работу и ответ.

3.2 Демонстрация пользовательского интерфейса веб-сервиса

При посещении веб-сервиса пользователь увидит информационную страницу предприятия, с которой пользователь может пройти на

авторизацию или регистрацию. На рисунке 35 представлен внешний вид информационной страницы.

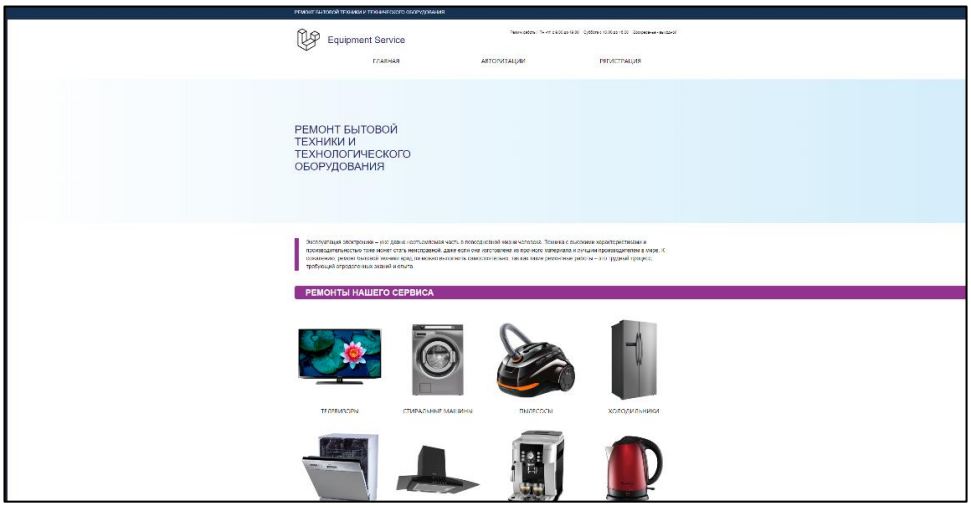

Рисунок 35 – Информационная страница

Нажав кнопку «Авторизация», пользователь перейдет на страницу формы авторизации (рисунок 36), в которой необходимо ввести авторизационные данные.

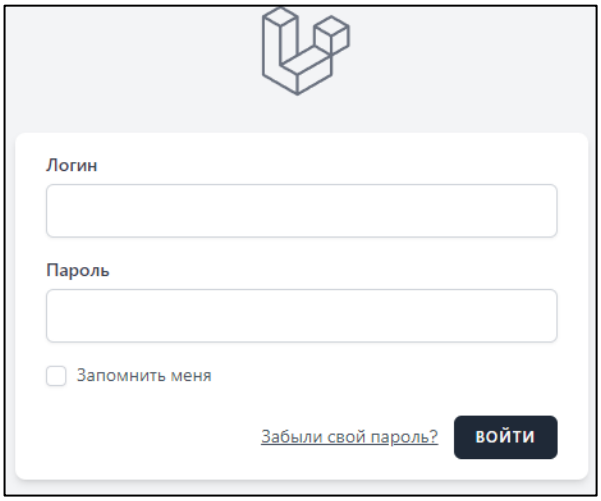

Рисунок 36 – Страница авторизации

Нажав кнопку «Регистрация», пользователь перейдет на страницу формы регистрации (рисунок 37), в которой необходимо указать ФИО, номер телефона, почту для уведомлений , а также пароль.

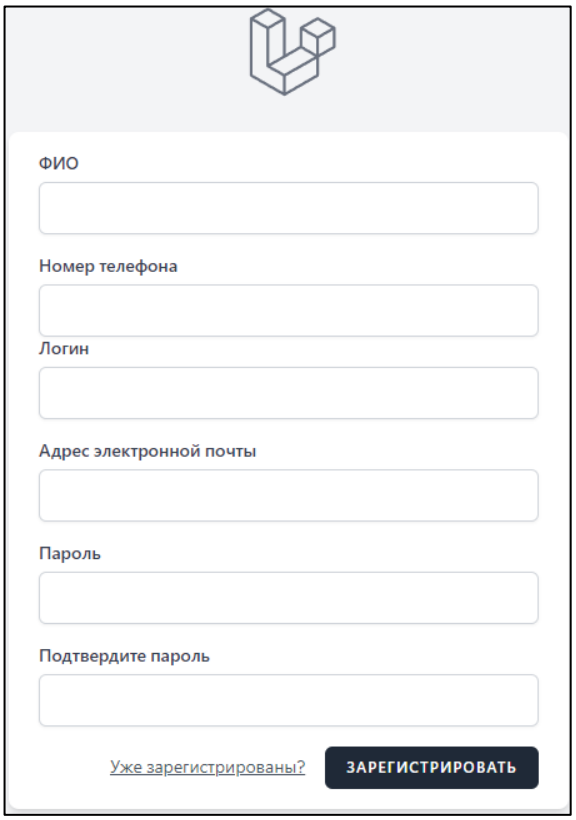

Рисунок 37 – Страница регистрации

После регистрации пользователю необходимо подтвердить указанную электронную почту. На неё будет отправлено сообщение, пример которого представлен на рисунке 38. Нажав на кнопку, пользователь подтвердит почту.

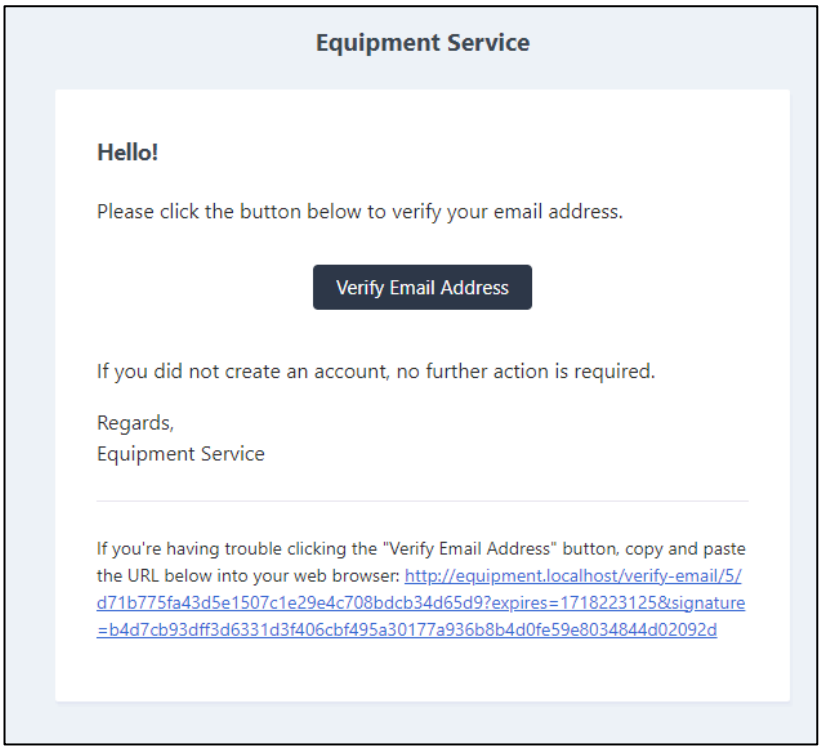

Рисунок 38 – Сообщение о подтверждении почты

После авторизации или подтверждения почты, пользователь окажется на рабочей странице (рисунок 39), на которой клиент сможет оформить заказ. А сотрудник сможет работать с заказами (рисунок 40).

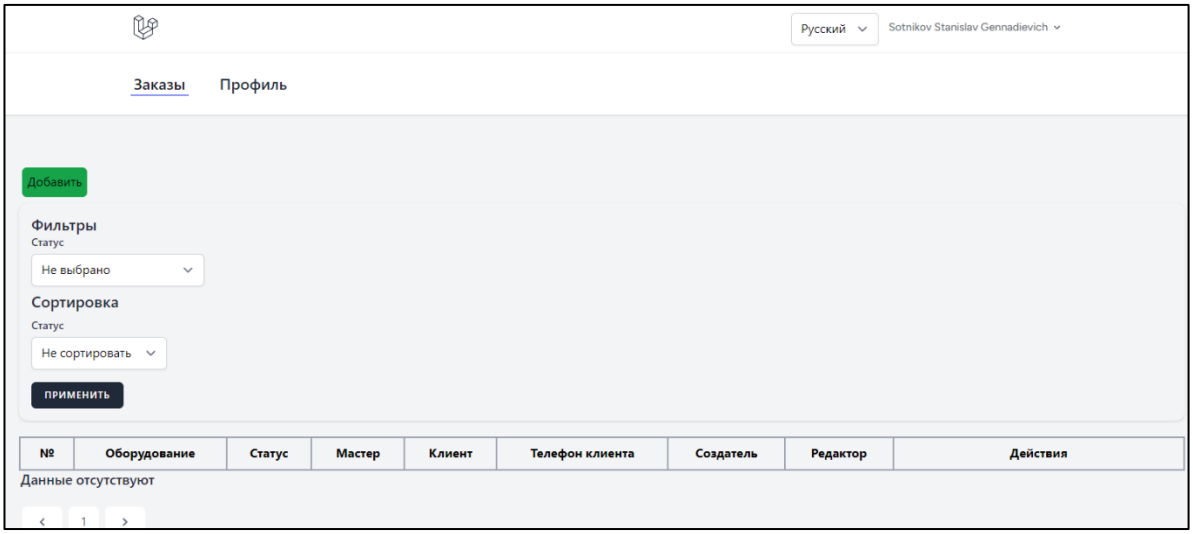

Рисунок 39 – Рабочая страница клиента

Если пользователь ранее оставлял заказы, то он сможет узнать статус заказов.

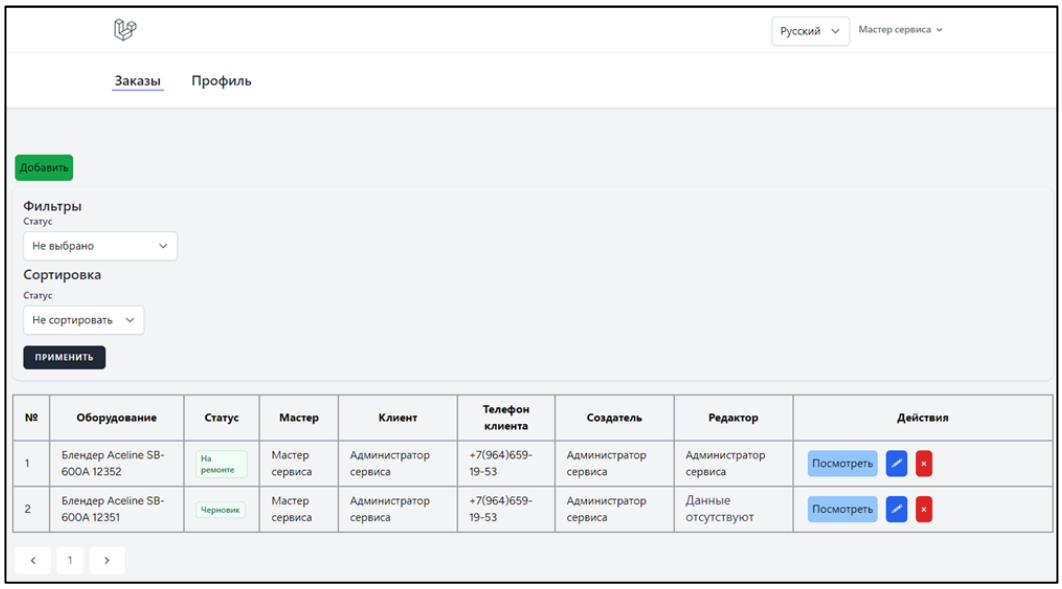

Рисунок 40 – Рабочая страница сотрудника

На рабочей страниц имеется кнопка добавить, которая позволяет добавить заказ (рисунок 41).

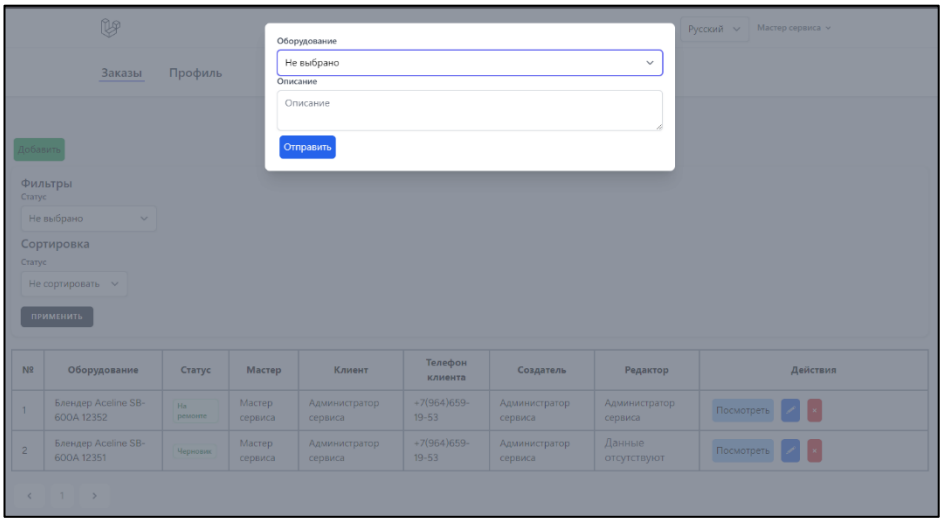

Рисунок 41 – Добавление заказа на рабочей странице

При нажатии кнопки, открывается модальное окно с формой, в которой необходимо выбрать оборудование и указать описание проблемы.

Также на рабочей странице пользователь может перейти на свой профиль и отредактировать информацию об аккаунте (рисунок 42).

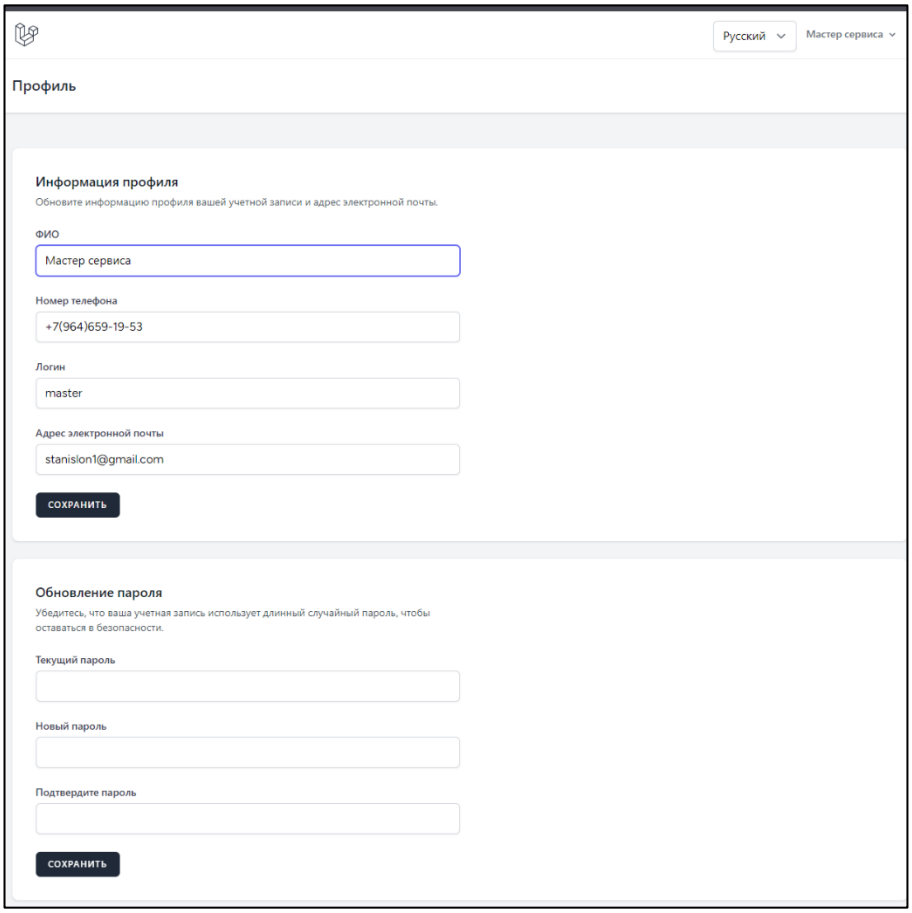

Рисунок 42 – Профиль пользователя

Администратору, или сотруднику с необходимыми правами, на рабочей странице открывается больше разделов. Также Администратор видит удаленные заказы и может их восстановить (рисунок 43).

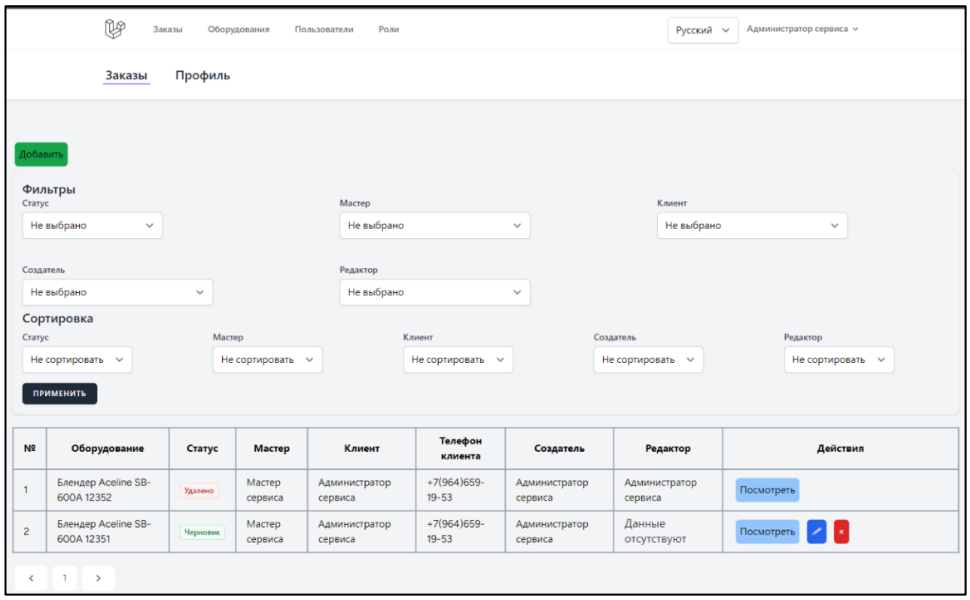

Рисунок 43 – Рабочая страница администратора

Раздел заказы позволяет манипулировать заказами в системе, то есть в этом разделе пользователь может добавить, отредактировать, сменить статус, удалить или восстановить заказы.

На рисунке 44 представлена главная страница раздела заказы.

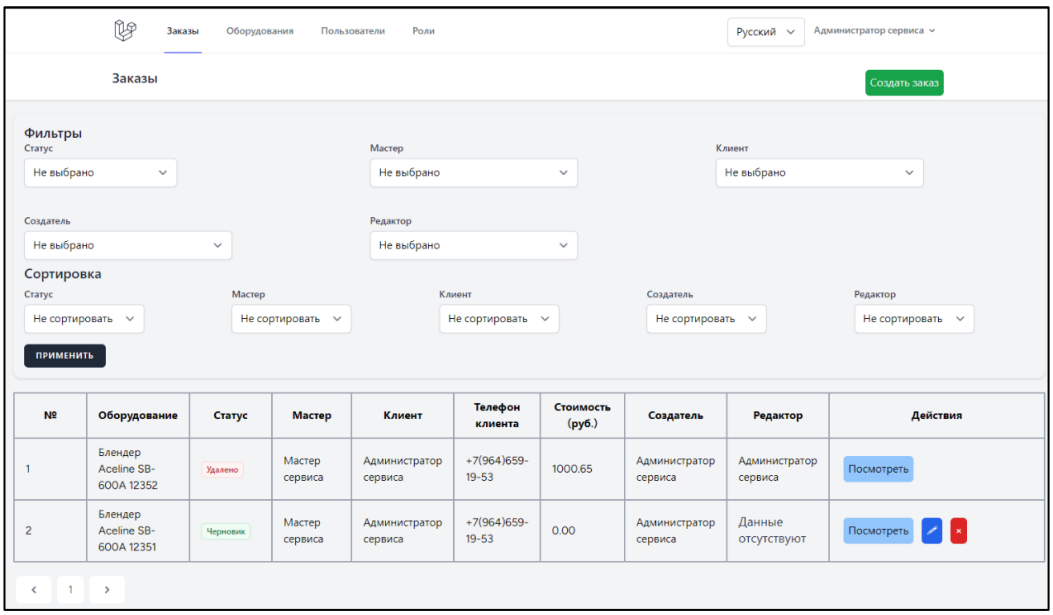

Рисунок 44 – Главная страница раздела заказы

Кнопка создать заказ открывает соответствующую форму, представленную на рисунке 45.

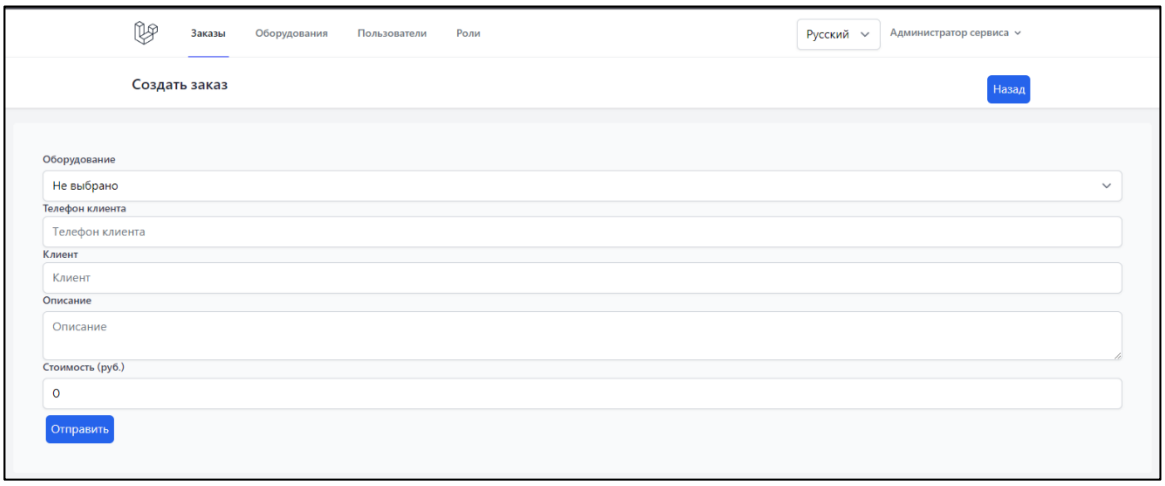

Рисунок 45 – Форма создания заказа

На этой странице администратор или сотрудник могут создать новый заказ, когда клиент прибыл в офис или позвонил.

Также сотруднику и администратору доступно редактирование заказа (рисунок 46).

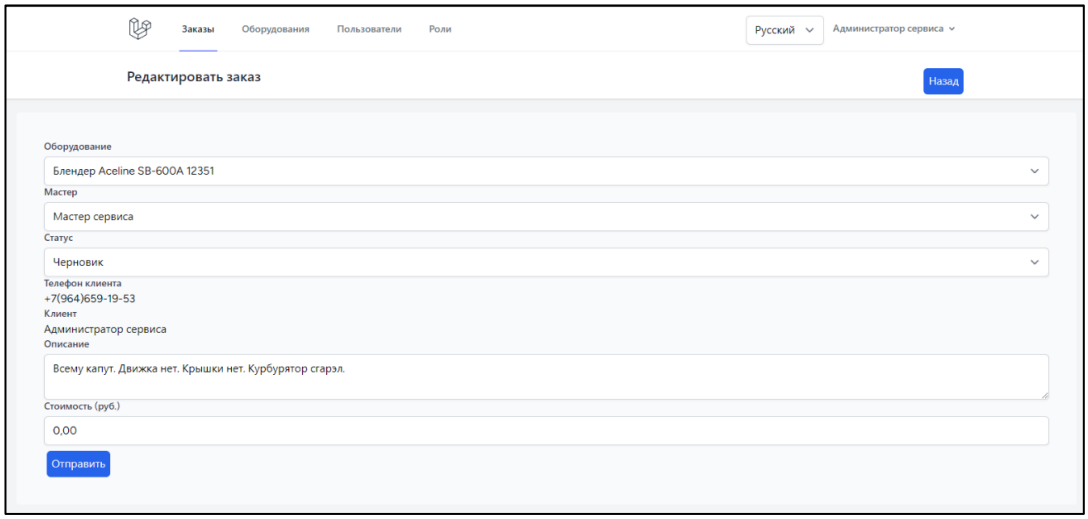

Рисунок 46 – Форма редактирования заказа

На этой форме пользователь может изменить заказ, дополнить данными, сменить статус. При достижении определенных статусов заказа клиенту будет отправлено уведомительное письмо.

При удалении заказа, статус заказа меняется на соответствующий. Восстановить заказ можно только при просмотре заказа на соответствующей странице (рисунок 47).

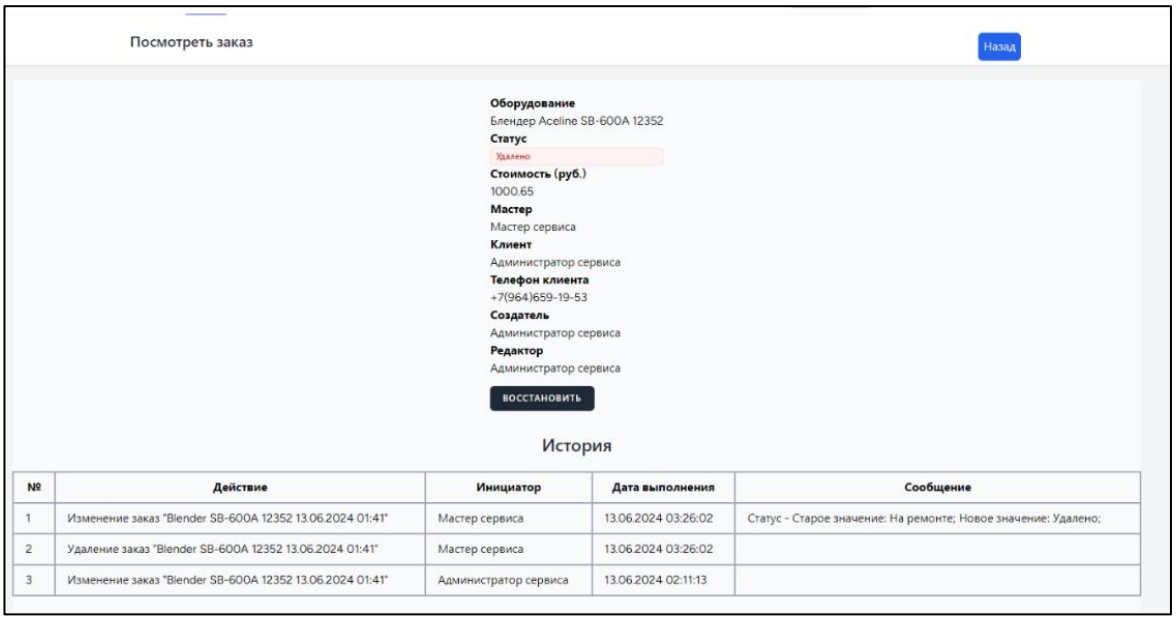

Рисунок 47 – Страница заказа

Кнопка восстановить позволяет изменить статус заказа из удалено на черновик. Также на этой странице имеется история заказа, как изменялись данные, и кто и когда их изменял.

Раздел оборудования позволяет манипулировать данными оборудований и сопутствующими данными. То есть в нем можно добавить оборудование, модель, тип оборудования, изменить необходимые поля типа оборудования, а также все эти данные редактировать, удалять или восстанавливать.

Так на рисунке 48 представлена главная страница раздела оборудования.

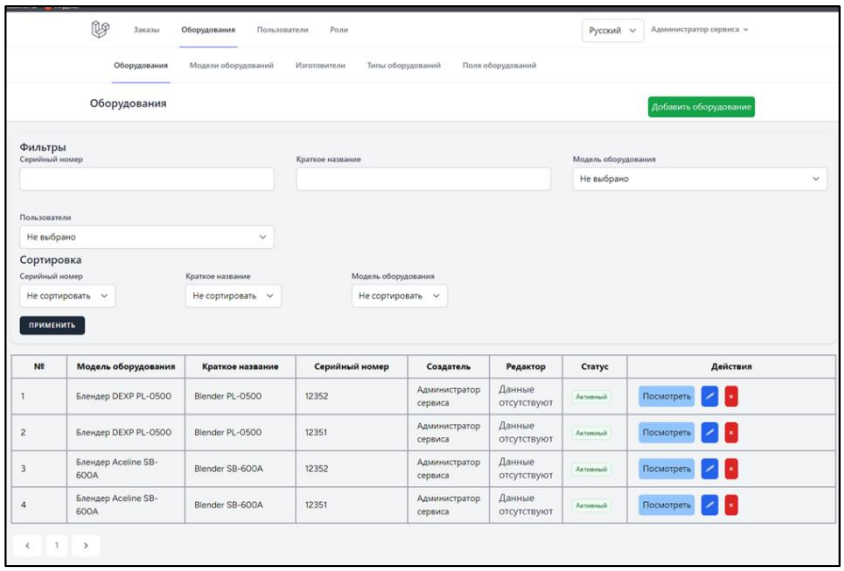

Рисунок 48 – Главная страница раздела оборудования

Этот раздел имеет подразделы, которые имеют аналогичный внешний вид и более простые формы добавления, редактирования, но необходимы для успешного создания оборудования.

В этом разделе также имеет кнопка добавить, редактировать и удалить. Так как всё аналогично предыдущему разделу, то на рисунке 49 представлена форма добавления оборудования.

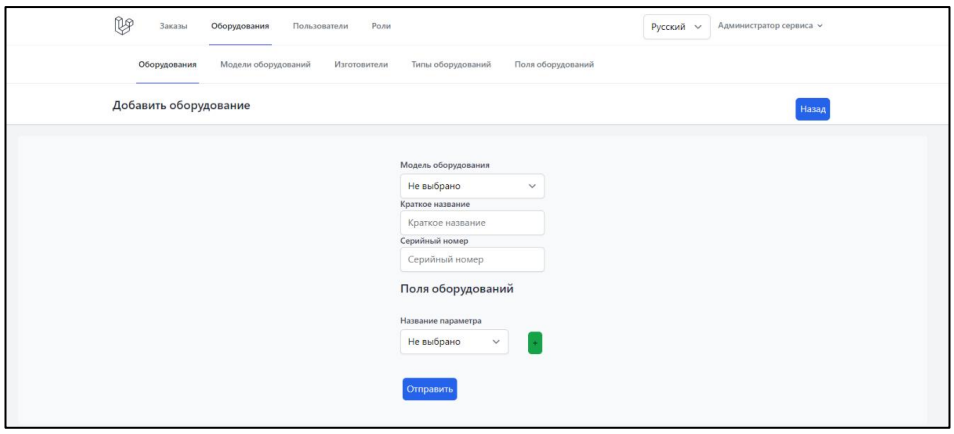

Рисунок 49 – Форма добавления оборудования

Особенностью данной формы является указание дополнительных полей и их заполнение.

На рисунке 50 представлена форма редактирования оборудования.

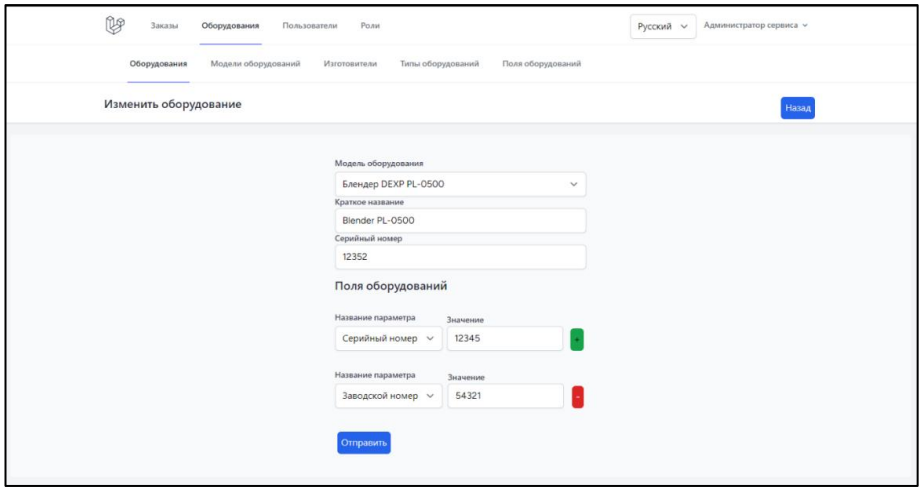

Рисунок 50 – Форма редактирования оборудования

В формах имеется выпадающий список под названием модель оборудования. Для моделей оборудования имеется отдельный подраздел, главная страница которого представлена на рисунке 51.

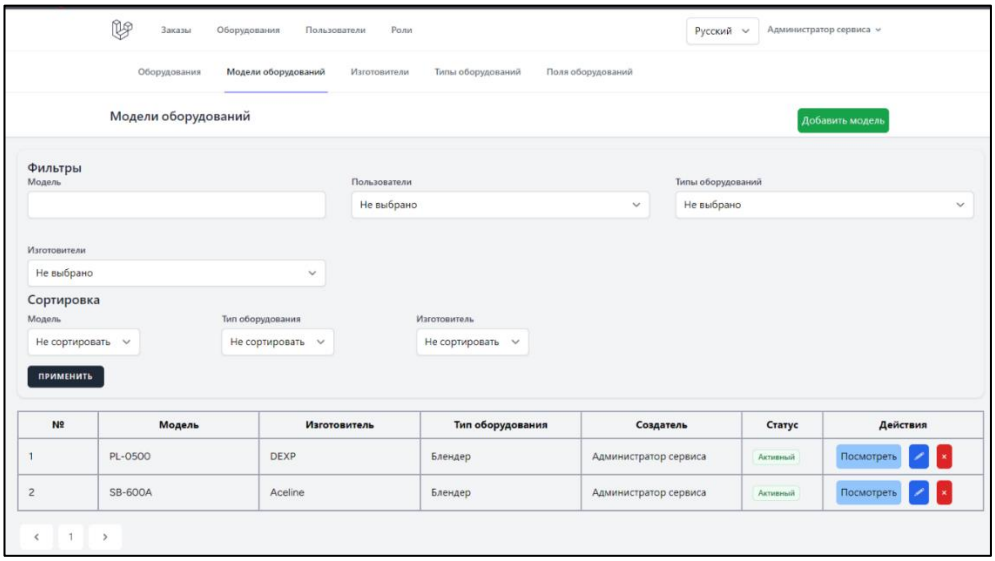

Рисунок 51 – Главная страница раздела модели оборудования

На рисунке 52 представлена форма добавления модели, в которой необходимо указать название модели, выбрать изготовителя и тип оборудования, к которому оно принадлежит.

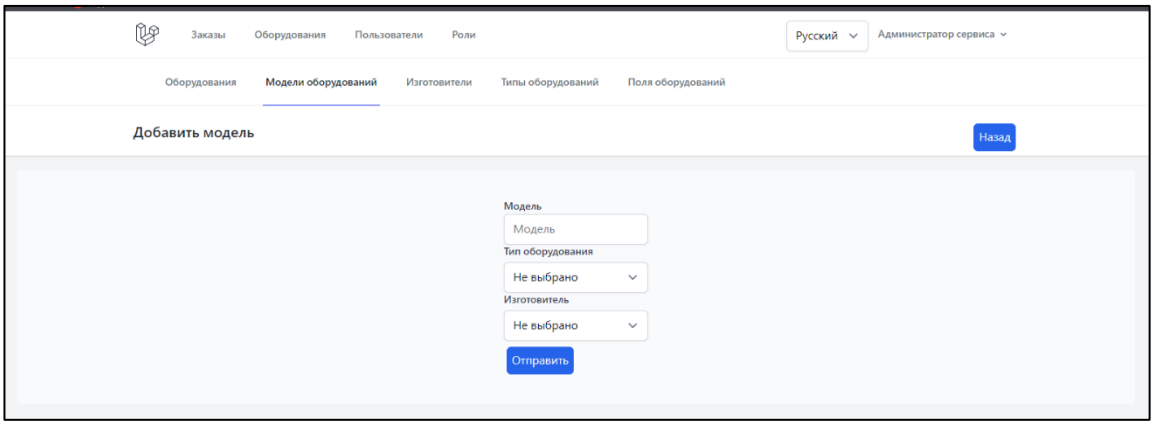

Рисунок 52 – Форма добавления модели

Форма редактирования аналогична форме добавления.

Главная страница раздела изготовители представлена на рисунке 53.

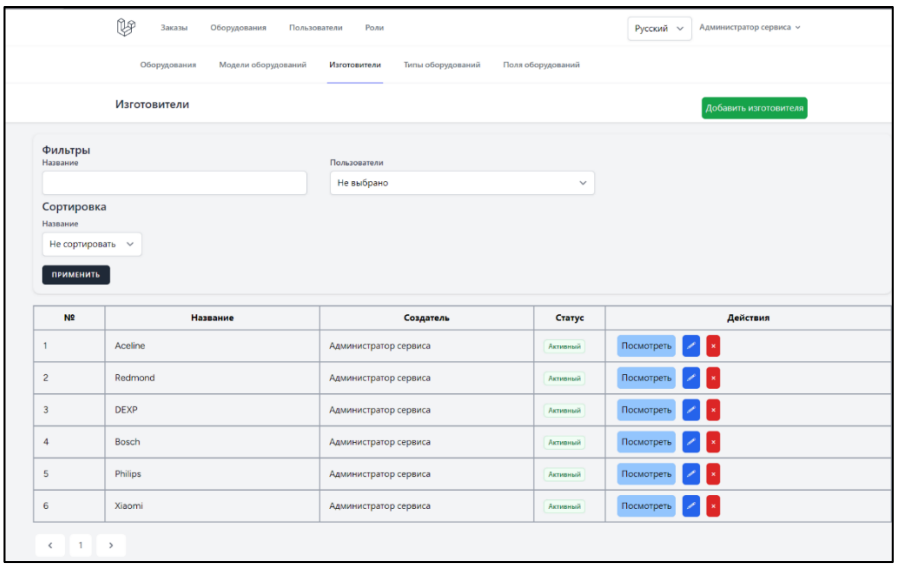

Рисунок 53 – Главная страница раздела изготовители

В форме добавления изготовителя необходимо указать название фирмы изготовителя.

Раздел типов оборудований абсолютно идентичен разделу изготовители, различны только в том, что указывается название типа оборудования, а не изготовителя оборудования.

Раздел поля оборудований необходим, чтобы настроить необходимые поля для оборудований, которые могут быть уникальны. Так на рисунке 54 представлена главная страница.

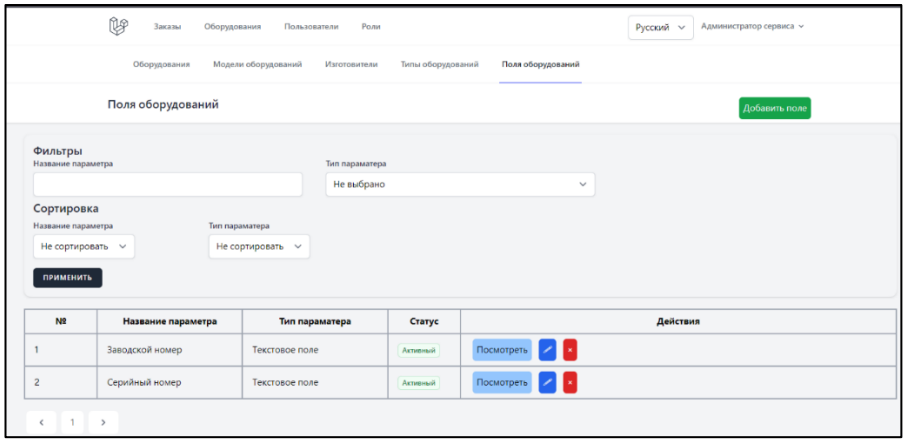

Рисунок 54 – Главная страница раздела поля оборудований

На рисунке 55 представлена форма добавления поля оборудования, в ней необходимо указать названия поле, а также какую информацию она должна содержать.

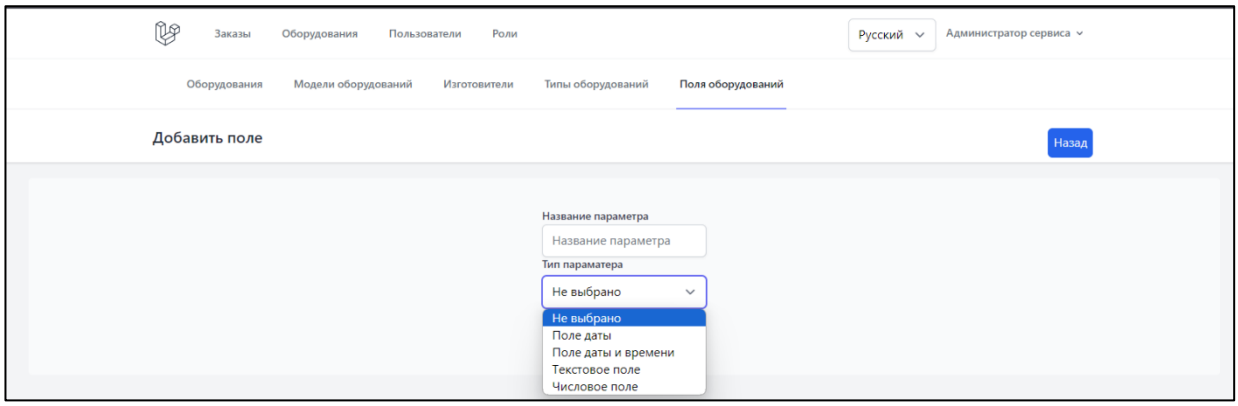

Рисунок 55 – Форма добавления поля оборудования

Раздел пользователи позволяет администратору регистрировать новых сотрудников, изменять им роли, а также личные данные. Так на рисунке 56 представлена главная страница раздела.

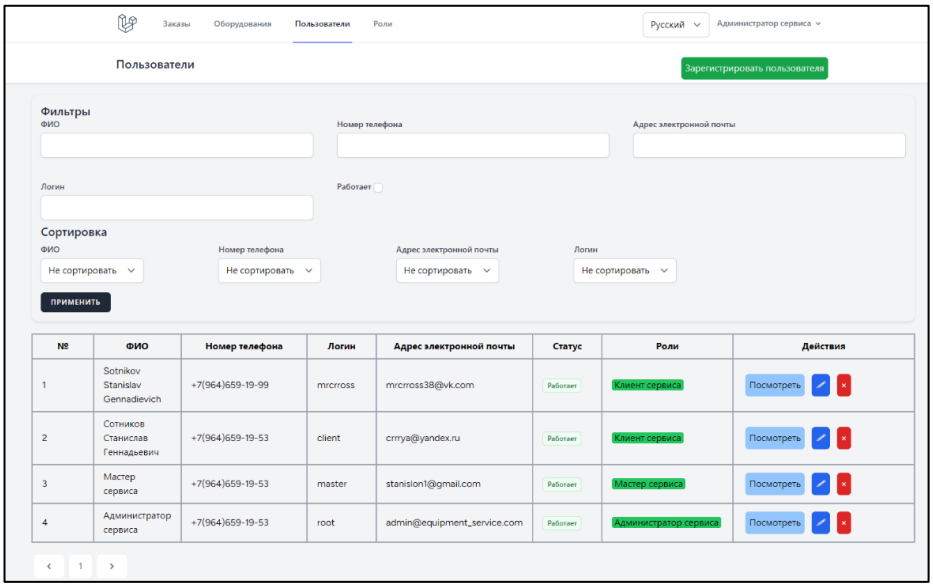

Рисунок 56 – Главная страница раздела пользователи

Форма регистрации пользователя представлена на рисунке 57, в ней администратору необходимо указать фио пользователя, номер телефона, почту, пароль, а также необходимые ему роли.

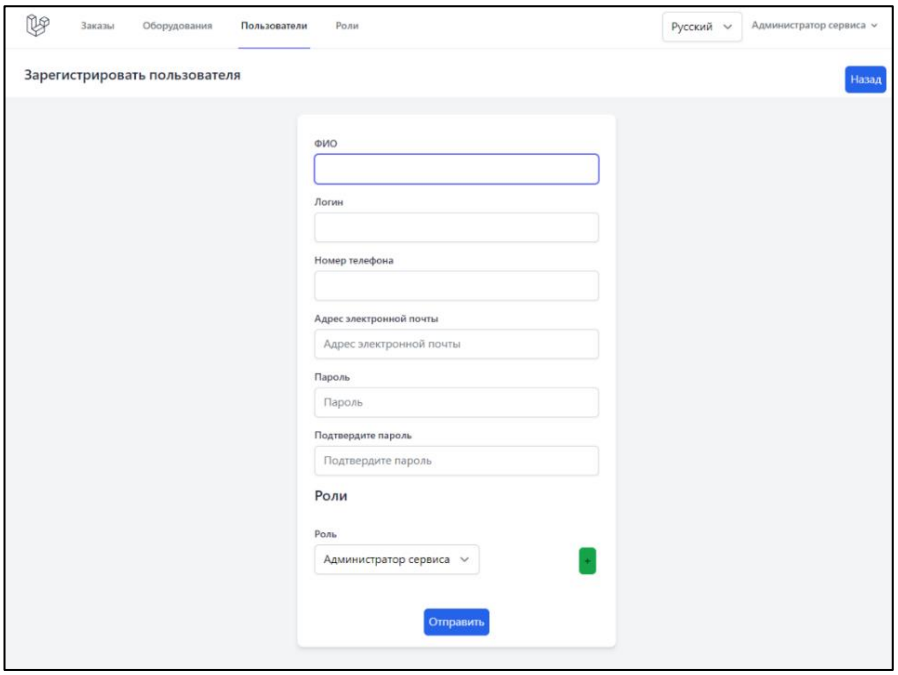

Рисунок 57 – Форма регистрации пользователя

Форма редактирования позволяет изменить указанные ранее данные.

Раздел роли позволяет добавить новые роли или изменить имеющиеся. Это позволяет точечно или массово изменять доступ пользователей к различным функциям веб-сервиса. Главная страница представлена на рисунке 58.

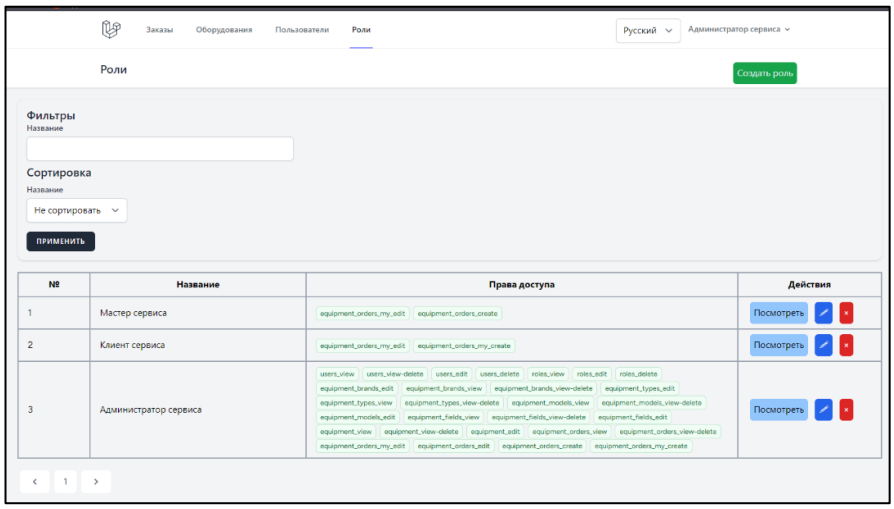

Рисунок 58 – Главная страница раздела роли

На рисунке 59 представлена форма создания роли, в ней администратору необходимо указать название роли, а также указать какие именно права доступа будут прикреплены к роли.

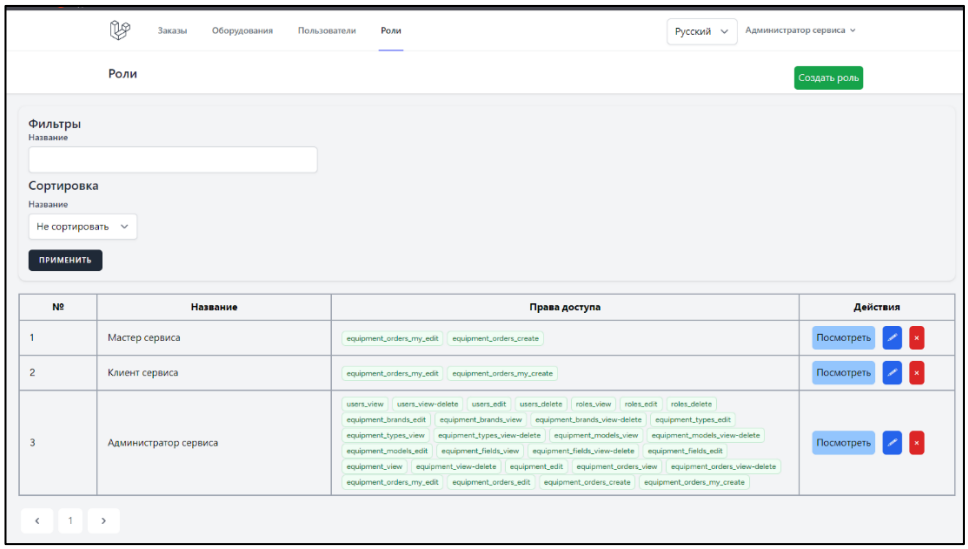

Рисунок 59 – Форма создания роли

В выпускной квалификационной работе реализованы все необходимые функции согласно требованиям выставленных для веб-сервиса по подаче заявок на ремонт и обслуживание бытовой техники.

3.3 Оценка экономической эффективности

Разрабатываемый веб-сервис будет использоваться для подачи заявок на ремонт и обслуживание бытовой техники.

Разработанный программный продукт может быть распространен в небольших сервисах. Стоимость разработки веб-сервиса по подаче заявок на ремонт и обслуживание бытовой техники составляет от 100 000 рублей, в зависимости от функционала.

Расчет полных затрат на разработку проектного решения ($К_{РПР}$) осуществляется по формуле 1:

$$К_{РПР}=З_{ОТР}+З_{ЭВМ}+З_{СПП}+З_{ХОН}+Е+А, \quad (1)$$

где $З_{ОТР}$ – сумма оплаты труда разработчика/разработчиков ПП;

$З_{ЭВМ}$ – затраты, связанные с эксплуатацией техники;

$З_{СПП}$ – затраты на специальные программные продукты, необходимые для разработки ПП;

$З_{ХОН}$ – затраты на хозяйственно-операционные нужды (бумага, литература, носители информации и т.п.);

Е – затраты на электроэнергию, руб.;

А – амортизация ПК, руб.

$$К_{РПР} = 56160 + 3556{,}8 + 0 + 405 + 723{,}84 + 1370 = 62215{,}64$$

Для подсчета фонда оплаты труда разработчика необходимо определить общее время разработки (таблица 4). Время, затрачиваемое на разработку проектного решения j-м разработчиком, определяется методом экспертных оценок или хронометража. Итоговое значение рассчитывается на основании приведенных исходных данных по формуле 2:

$$T_{РПРj} = \sum_{\beta=1}^{n} t_{\beta}, \quad (2)$$

где tβ– время β-го этапа разработки проектного решения, дн.

$$Т_{РПРj} = 3 + 3 + 2 + 2 + 58 + 3 + 5 + 2 = 78$$

Таблица 4 - Затраты времени на создание программного продукта

Этап создания		Затраты времени (в днях)	Затраты времени (в часах)	Машинное время работы над ПП (в часах)
Разработка	Исследование объекта автоматизации	3	24	24
	Анализ и уточнение требований	3	24	24
	Разработка технического задания	2	16	16
	Проектирование структуры	2	16	16

Продолжение таблицы 4 - Затраты времени на создание программного продукта

Этап создания		Затраты времени (в днях)	Затраты времени (в часах)	Машинное время работы над ПП (в часах)
Разработка	Программная реализация	58	464	464
	Тестирование программного продукта	3	24	24
	Отладка программного продукта	5	40	40
	Разработка описания	2	16	16
	ИТОГО	78	624	624

Рабочий день принимается равным 8 часам.

Сумму оплаты труда разработчика за время работы над программным продуктом рассчитаем исходя из часовой тарифной ставки и фонда фактического времени, затраченного на разработку программного продукта (по формуле 3):

$$З_{отр}=С_{т1}*Ф_{вр}, \quad (3)$$

где $С_{т1}$ - часовой тарифной ставки (принимаем равной 90 руб./час);

$Ф_{вр}$ – фонд фактического времени, затраченного на разработку программного продукта, час.

$$З_{отр} = 90 * 624 = 56160$$

Начисления на заработную плату рассчитываются в таблице 5.

Таблица 5 – Начисления на заработную плату

Начисления на заработную плату	Процент, %	Сумма, руб.
Пенсионный фонд (ПФ): – страховая часть – накопительная часть	16 6	8985,6 3369,6

Продолжение таблицы 5 – Начисления на заработную плату

Фонд социального страхования (ФСС)	2,9	1628,6
Федеральный фонд обязательного медицинского страхования (ФФОМС)	5,1	2864,2
ИТОГО	30	16848

Затраты, связанные с использованием вычислительной и оргтехники, рассчитываются по формуле 4:

$$З_{ЭВМ} = T_{МРПР} \cdot k_Г \cdot n \cdot C_{М-Ч}, \quad (4)$$

где $T_{МРПР}$ – машинное время работы над программным продуктом, час;

$k_Г$ – коэффициент готовности ЭВМ, $k_Г = 0{,}95$;

n – количество единиц техники, равно 1;

$C_{М-Ч}$ – себестоимость машино-часа, $C_{М-Ч}$ =6руб.

$$З_{ЭВМ} = 624 * 0{,}95 * 6 = 3556{,}8$$

Затраты на электроэнергию рассчитываются по следующей формуле 5:

$$E = W \times t \times T, \quad (5)$$

где W – мощность, потребляемая ПК, кВт/час;

t – время работы ПК, час;

T – тариф электроэнергии, руб.

Тариф 1,45 рубля за киловатт.

$$E = 0{,}8 * 624 * 1{,}45 = 723{,}84$$

Амортизация ПК рассчитывается по следующей формуле 6:

$$A = \frac{S \times q_{am}}{12}, \quad (6)$$

где S – первоначальная стоимость ПК, руб;

q_{am} – процент амортизации в год.

$$A = \frac{55000 * 0{,}3}{12} = 1370$$

При разработке программного продукта использовались следующие свободно распространяемые специальные продукты PhpStorm, Draw.IO. Отсюда следует, что $З_{СПП} = 0$.

Затраты на хозяйственно-организационные нужды приводятся в таблице 6 и вычисляются по формуле 7:

$$З_{ХОН} = \sum_{\tau=1}^{n} Ц_{\tau} \cdot К_{\tau} , \quad (7)$$

где $Ц_{\tau}$ – цена τ-го товара, руб.;

$К_{\tau}$ – количество τ-го товара.

$$З_{ХОН} = 50 + 255 + 100 = 405$$

Таблица 6 - Затраты на хозяйственно-организационные нужды

Наименование	Цена за единицу (руб.)	Кол-во (шт.)	Всего (руб.)
Диск CD-RW Digitex	50	1	50
Бумага	5	51	255
Папка скоросшиватель	50	2	100
Итого			405

Результаты выполненных расчетов сводятся в общей таблице (таблица 7).

Таблица 7 – Затраты на разработку

Наименование затрат	Условное обозначение	Значение
Оплата труда разработчика программного продукта	$З_{ОТР}$	56160
Затраты, связанные с эксплуатацией техники	$З_{ЭВМ}$	3556,8
Затраты на электроэнергию	Е	723,84
Затраты на амортизацию ПК	А	1370
Затраты на хозяйственно-операционные нужды	$З_{ХОН}$	405
Затраты на программное обеспечение	$З_{СПП}$	0
Итого затрат на разработку	$К_{РПР}$	62215,64

Веб-сервис не внедряется в предприятие, поэтому затрат на внедрение нет. Также все используемые программные средства бесплатны и свободно распространяются.

Исходя из расчетов стоимости разработки, можно прийти к выводу, что стоимость разработанного программного продукта будет составлять 5980,76

рублей (без учета оплаты труда). Данный программный продукт будет использоваться для подачи заявок на ремонт и обслуживание бытовой техники.

Из проведенных расчетов можно выявить, что итоговая стоимость затрат на разработку программного продукта, в которую включается оплата труда разработчика, затраты на эксплуатацию техники, в том числе и амортизация ПК, так же затраты на электроэнергию и затраты на хозяйственно-операционные нужды, составила 62215,64 рублей.

Из проведенных расчетов можно сделать вывод, что разработка веб-сервиса является оптимальным и недорогостоящим процессом.

Выводы по главе 3

В третьей главе была выбрана архитектура разрабатываемого веб-сервиса, выбран технологии при помощи, которых разрабатывался веб-сервис. Также был продемонстрирован код миграций для создания базы данных, код функции подачи заявок на ремонт и обслуживание бытовой техники, код управления заказами в контроллере «OrdersController» и маршруты этого контроллера, а также код и вид разработанного пользовательского интерфейса.

Были определены все необходимые формулы для подсчета эффективности. Были проведены расчеты и выявлено что разработка данного веб-сервиса обошлось дешевле, чем приобретение готовых решений.

Заключение

В ходе выпускной квалификационной работе был разработан веб-сервис по подаче заявок на ремонт и обслуживание бытовой техники, который содержал в себе API для соответствующих действий. Было реализовано добавление, обновление и удаление оборудования, типов оборудования, изготовителей и заявок. Также был разработан механизм подачи заявок и уведомлений клиентов о заявках.

Были приобретены практические и профессиональные навыки в области программирования, а именно: изучены новые информационные технологии, новые методы проектирования бизнес-процессов, изучен шаблон проектирования MVC. Приобретены навыки анализирования бизнес-процессов и существующих разработок.

Были закреплены навыки в концептуального моделирования предметной области при построении различных диаграмм.

Все поставленные задачи были выполнены, а именно:

– Изучить предметную область.

Была проанализирована предметная область подача заявок на ремонт и обслуживание бытовой техники. Было проведено концептуальное моделирование, спроектирована модель «как-есть», а также построена организационная структура.

– Изучение существующих веб-сервисов и приложений для подачи заявок на ремонт бытовой техники с целью выявления основных проблем и недостатков.

Были рассмотрены и проанализированы такие программные решения, как LiveSklad, GinCore, HelloClient. Была построена сравнительная таблица и выявлена необходимость разработки веб-сервиса.

– Определение функциональных и не функциональных требований к разрабатываемому веб-сервису.

Для разрабатываемого веб-сервиса были определены необходимые требования согласно методологии FURPS+. Были составлены аппаратно-программные требования, благодаря которым веб-сервис будет отвечать всем необходимым стандартам безопасности, надежности и удобства.

– Проектирование архитектуры и интерфейса веб-сервиса с учетом современных тенденций пользовательского опыта и удобства использования.

Были спроектированы контекстная диаграмма, диаграмма декомпозиции, диаграмма вариантов использования, диаграмма деятельности, диаграмма развертывания, концептуальная модель данных, логическая модель данных, физическая модель данных. Спроектирован пользовательский интерфейс и продемонстрирован проект рабочей страницы, страницы создания, редактирования заявок и оборудования.

– Реализация программного кода веб-сервиса с использованием современных технологий веб-разработки.

Для разработки были выбраны php, Laravel, mysql, javascript, tailwindCSS, которые позволили реализовать веб-сервис. Для разработки использовалась модель MVC.

– Тестирование и отладка веб-сервиса для обеспечения стабильной и безопасной работы.

Благодаря модульному тестирования было проведено полное тестирование функций веб-сервиса. Из-за чего были выявлены все ошибки и отлажены.

– Оценка эффективности разработанного веб-сервиса.

Была проведена оценка стоимости разработки, благодаря чему было выявлена экономическая эффективность.

Из-за того, что все задачи выполнены, можно считать, что цель: разработать веб-сервис по подачи заявок на ремонт и обслуживание бытовой техники - была достигнута.

Актуальность данной темы была доказана и подтверждена.

Список используемой литературы и используемых источников

1. Арсеньев Ю.Н., Шелобаев С.И., Давыдова Т.Ю. Информационные системы и технологии. Экономика. Управление. Бизнес: учеб. пособие для студентов вузов, обучающихся по направлениям 080500 «Менеджмент» и 080100 «Экономика». – Москва: ЮНИТИ-ДАНА, 2017. – 447 с.
2. Блинов А.О., Реинжиниринг бизнес-процессов: учеб. пособие для студентов вузов, обучающихся по специальностям экономики и управления – М.: ЮНИТИ-ДАНА, 2017 – 344 с.
3. Введение в СУБД MySQL: Лекция 1: Введение в MySQL [Электронный ресурс]: ИНТУИТ Национальный открытый университет. URL: https://intuit.ru/studies/courses/111/111/lecture/28508 (дата обращения: 23.04.2024).
4. Вдовенко Л.А. Информационная система предприятия: Учебное пособие/Вдовенко Л.А. - 2 изд., перераб. и доп. - М.: Вузовский учебник, НИЦ ИНФРА-М, 2015. – 304 с.
5. Виноградова Е., Шориков А. Разработка информационной системы комплексного управления предприятием. – 2022.
6. Гританс Я.М. Организационное проектирование и реструктуризация (реинжиниринг) предприятий и холдингов: экономические, управленческие и правовые аспекты: практ. пособие по управлен. и финанс. Консультированию – 2к изд., доп. – Москва, 2008. – 224 с.
7. Костенко Д.В., Харьков В.П. Исследование и разработка информационных систем предприятий с процессным типом управления – НИБ, 166 с.
8. Лешек А. Мацяшек. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML. - Москва-Санкт-Петербург-Киев: Вильямс, 2002.

9. Лисяк В.В. Разработка информационных систем – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2019. – 96 с.

10. Маклаков С.В. CASE-средства разработки информационных систем. – М.: ДИАЛОГ МИФИ, 2001. – 304 с.

11. Мартин Фаулер и Кендалл Скотт, UML. Основы. - СПб: Символ-Плюс, 2002.

12. Программа для автоматизации бизнеса услуг и продаж [Электронный ресурс]: HelloClient. URL: https://helloclient.ru/ (дата обращения: 25.04.2024).

13. Производственная практика. Электронное учебное-методическое пособие. Казаченок Н.Н., Михеева О.П. 2018 – 50 с.

14. Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов. - Москва: ДМК-издательство, 2002.

15. Умная автоматизация сервисных центров и мастерских [Электронный ресурс]: LiveSklad. URL: https://livesklad.com/ (дата обращения: 20.04.2024).

16. Хаммер М., Чампли Д. Реинжиниринг корпорации манифест революции в бизнесе, 1995, USA – 303 с.

17. Что такое API и как он работает [Электронный ресурс]: Skillbox Media. URL: https://skillbox.ru/media/code/chto_takoe_api (дата обращения: 11.04.2024).

18. Что такое MVC: базовые концепции и пример приложения [Электронный ресурс]: Skillbox Media. URL: https://skillbox.ru/media/code/chto_takoe_mvc_bazovye_kontseptsii_i_primer_prilozheniya/ (дата обращения: 01.05.2024).

19. Что такое PHP? [Электронный ресурс]: Руководство по PHP. URL: https://www.php.net/manual/ru/intro-whatis.php (дата обращения: 05.04.2024).

20. Controllers [Электронный ресурс]: Laravel. URL: https://laravel.com/docs/11.x/controllers (дата обращения: 02.05.2024).

21. David Flanagan: JavaScript: The Definitive Guide / David Flanagan – 6th ed. – Sebastopol: O`Reilly Media, 2011 – 1098 p.

22. Felipe Gutierrez: Pro Spring Boot 2: An Authoritative Guide to Building Microservices, Web and Enterprise Applications / Felipe Gutierrez – 1th ed. – Apress, 2019 – 520 p.

23. Fowler M., Scott K. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – Boston: Addison-Wesley, 2000 – 114 p.

24. Get started with Tailwind CSS [Электронный ресурс]: Tailwindcss. URL: https://tailwindcss.com/docs/installation (дата обращения: 04.05.2024).

25. Gincore с первого дня поможет вам [Электронный ресурс]: Gincore. URL: https://gincore.net/ (дата обращения: 09.05.2024).

26. GIO - инструмент для оптимизации процессов бизнеса [Электронный ресурс]: GIO. URL: https://gio.app/ (дата обращения: 21.04.2024).

27. Marcin Grzejszczak: Mockito Cookbook / Marcin Grzejszczak – 1th ed. – Birmingham: Packt Publishing Ltd, 2014 – 284 p.

28. MySQL Installer Guide [Электронный ресурс]: MySQL. URL: https://dev.mysql.com/doc/mysql-installer/en (дата обращения: 12.04.2024).

29. Security-first diagramming for teams [Электронный ресурс]: Draw.io. URL: https://app.diagrams.net/ (дата обращения: 15.05.2024).

30. Seyed M.M. Tahaghoghi, Hugh E. Williams: Learning MySQL / Seyed M.M. Tahaghoghi, Hugh E. Williams – 1th ed. – Sebastopol: O`Reilly Media, 2007 – 622 p.

31. UML для начинающих: Аве Кодер [Электронный ресурс]: Youtube. URL: https://www.youtube.com/playlist?list=PLPPIc-4tm3YTw3FUu75jsW4QgrXopfXhX (дата обращения: 15.04.2024).