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

Кафедра Прикладная математика и информатика
(наименование)

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

Разработка программного обеспечения
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка программного обеспечения информационной системы управления заказами ремонтной мастерской

Обучающийся А. Б. Петров
(Инициалы Фамилия) (личная подпись)

Руководитель д.т.н., доцент, С.В. Мкртычев
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Тема бакалаврской работы – «Разработка программного обеспечения информационной системы управления заказами ремонтной мастерской».

Структура дипломной работы представлена введением, тремя главами, заключением, списком использованной литературы и приложениями.

В введении представлены актуальность выбранной темы, объект и предмет исследования, основные цели и задачи исследования, используемые методики и практическая значимость полученных результатов.

В первой главе рассматриваются объект и предмет исследования, формулируется постановка задач на разработку программного обеспечения информационной системы.

Во второй главе рассматриваются методологии и технологии проектирования информационных систем и моделирование бизнес-процессов для предмета исследования.

В третьей главе описывается процесс прототипирования и отладки программного обеспечения информационной системы.

Заключение отражает основные проектные решения, разработанные методики и модели, общие выводы проведённых исследований.

Ключевые слова: ремонтная мастерская, программное обеспечение, прототип, информационная система.

Бакалаврская работа представлена на 49 страницах и включает в себя 18 рисунков, 4 таблицы и приложение.

Список используемой литературы и используемых источников содержит 25 источников.

Оглавление

Введение

В современном бизнесе эффективное управление заказами играет важную роль в обеспечении качественного обслуживания клиентов.

Программное обеспечение информационной системы управления заказами играет решающее значение в данном случае, так как оно позволит упростить процесс приёма и выполнения заказов, а также повысить скорость и качество обслуживания клиентов, вследствие чего улучшит работу ремонтной мастерской в целом.

Актуальность темы обусловлена необходимостью повышения эффективности работы ремонтной мастерской за счёт эффективного управления заказами. Разработка программного обеспечения повысит качество и прозрачность процесса обслуживания клиентов на всех этапах.

Объектом исследования бакалаврской работы является процесс управления заказами ремонтной мастерской.

Предметом исследования бакалаврской работы является программное обеспечение информационной системы управления заказами ремонтной мастерской.

Цель работы заключается в разработке программного обеспечения, которое позволит повысить эффективность и прозрачность процессов обработки заказов.

Работа основана на положениях теоретического и частного методах исследования, а именно анализа и практического моделирования.

Практическая значимость бакалаврской работы заключается в разработке программного обеспечения информационной системы управления заказами ремонтной мастерской, которое сможет показать важность наличия современных технологий в деятельности ремонтной мастерской.

В первой главе рассматриваются объект и предмет исследования, формулируется постановка задач на разработку программного обеспечения информационной системы управления заказами ремонтной мастерской.

Во второй главе рассматриваются методологии и технологии проектирования информационных систем и моделирование бизнес-процессов для предмета исследования. Описываются варианты использования системы и приводится их детальное описание. Выбор методологии проектирования программного обеспечения и логическое проектирование системы являются важными этапами, которые включают создание схемы базы данных и структуры приложения, что обеспечит правильное и эффективное функционирование всех модулей системы.

В третьей главе показывается процесс реализации и отладки программного обеспечения информационной системы. Данная глава начинается с выбора средств разработки, включая языки программирования, фреймворки и инструменты. Далее рассматривается разработка логической структуры базы данных и проектирование пользовательского интерфейса. Особое внимание уделяется тестированию. Отладка программного обеспечения проводится на разных уровнях тестирования, включая модульное, интеграционное и системное тестирование. На основе отчёта о тестировании проводится анализ результатов и внесение необходимых улучшений.

В заключении описываются результаты выполнения выпускной квалификационной работы.

Бакалаврская работа представлена на 49 страницах и включает в себя 18 рисунков, 4 таблицы и приложение.

Список используемой литературы и используемых источников содержит 25 источников [2].

Глава 1 Постановка задачи на разработку программного обеспечения информационной системы управления заказами ремонтной мастерской

1.1 Функциональные и архитектурные особенности информационных систем управления заказами предприятий сферы обслуживания

Программное обеспечение (ПО) информационной системы управления (ИСУ) заказами ремонтной мастерской должно быть основано на обеспечении прозрачности, оперативности и удобства управления заказами клиентов.

Функциональность ИСУ должна покрывать процесс управления заказом, т. е. система должна уметь создавать, отправлять и отслеживать заказ на ремонт от момента поступления заявки до завершения работы. ИСУ должна обеспечивать ввод, обработку и хранение данных по заказу, производить планирование и контроль, а также расчёт финансовых ресурсов компании. Система также должна иметь функциональность регистрации новых пользователей и авторизации в системе с разграничением прав. Работа в системе должна быть организована с различными правами доступа [12].

В сферах обслуживания типовая архитектура ИСУ заказами имеет пользовательский интерфейс, базу данных, функционал управления заказами, интеграцию с внешними системами (доставка, финансы), складской учёт, управление клиентскими данными, отчётность и аналитику. В целом ИСУ для ремонтных мастерских имеют такую же архитектуру, однако также необходимо добавить компоненты управления расписанием работ, учёта выполненных работ и материалов, функционал отслеживания статуса заказов и материалов в процессе выполнения ремонта.

В обоих случаях нужна база данных, которая является важным компонентом ИСУ.

Также важным аспектом является обеспечение безопасности данных и

управление доступом, чтобы защитить конфиденциальную информацию о клиентах и заказах.

Архитектура ИСУ направлена на оптимизацию процесса управления заказами, повышение эффективности и обеспечение точного учёта и отслеживания данных в соответствующих сферах деятельности.

1.2 Разработка требований к программному обеспечению информационной системы управления заказами

Для разработки требований необходимо общее представление об основных процессах, которые происходят при управлении заказа. Для классификации требований широко применяется технология FURPS. (Functionality Usability Reliability Performance Supportability). В настоящее время используется аббревиатура FURPS+. Термин FURPS+ происходит от усовершенствованной модели для классификации атрибутов качества программного обеспечения, включающих функциональные и нефункциональные требования. Эта технология широко используется в современной программной индустрии [7].

Формулировка требований к ПО ИСУ заказами ремонтной мастерской в соответствии с технологией FURPS+ представлена в таблице 1.

Таблица 1 – Требования к ПО ИСУ ремонтной мастерской

Требования	Описание
Функциональные Functionality	Управление заказами: система должна иметь возможность добавления номера, статуса и описания заказа, поиска заказов и изменения заказа. База данных для пользователей системы, клиентов и заказов. История взаимодействия с клиентом. Вход в систему.
Удобство Usability	Интерфейс системы должен быть интуитивно понятным для пользователей различного уровня. Должна быть обеспечена возможность быстрого доступа к основным функциям системы.

Продолжение таблицы 1

Требования	Описание
Надёжность Reliability	Система должна обеспечивать сохранность данных заказов и их целостность.
Производительность Performance	Система должна обеспечивать быстрый доступ к данным и оперативную обработку заказов. Время отклика системы должно быть минимальным.
Поддержка Supportability	Предусмотрены механизмы обновления системы и её компонентов. Обеспечена документация по использованию и сопровождению системы.
Дополнительные +	Проектирование: Хранение данных о заказе необходимо реализовать с помощью реляционной БД. Документация к коду должна быть написана в соответствии с соглашениями выбранного языка программирования. Документация сопровождения должна быть написана в MSWord. Разработка и кодирование: Логика программы должна быть написана на языке программирования JavaScript. В качестве СУБД должна быть использована SQLite.

Классификация требований важна в начале жизненного цикла разработки ПО ИСУ, поскольку она определяет основные цели и функциональные требования к системе. Благодаря систематическому подходу к сбору и анализу требований можно реализовать успешный проект и создать продукт, который будет соответствовать потребностям пользователей [8].

1.3 Обзор и анализ аналогов программного обеспечения информационной системы управления заказами

Обзор и анализ существующих аналогов ПО ИСУ позволит выявить их основные характеристики, функциональность, преимущества и недостатки.

Обзор также поможет составить таблицу сравнений на соответствие сформулированным требованиям, чтобы понять есть ли необходимость собственной разработки ПО ИСУ заказами ремонтной мастерской.

Система управления заказами и задачами Filta позволяет

автоматизировать бизнес-задачи, структурировать работу над заказами и получить контроль за работой организации. ИСУ реализует следующий функционал [16]:

– раздел заказов позволяет создавать задания на производство, добавлять позиции заказа, выбирать способ доставки и иную информацию по заказу;
– раздел задач позволяет ставить задачи с описанием и статусом между сотрудниками;
– дополнительные возможности позволяют делать экспорт заказа PDF и автоматически формировать задачи для сотрудников в зависимости от их роли.

Среди достоинств данной системы можно отметить структурирование работы над заказами, среди недостатков можно отметить нехватку функционала для процесса раздела заказов. Пользовательский интерфейс представлен на рисунке 1.

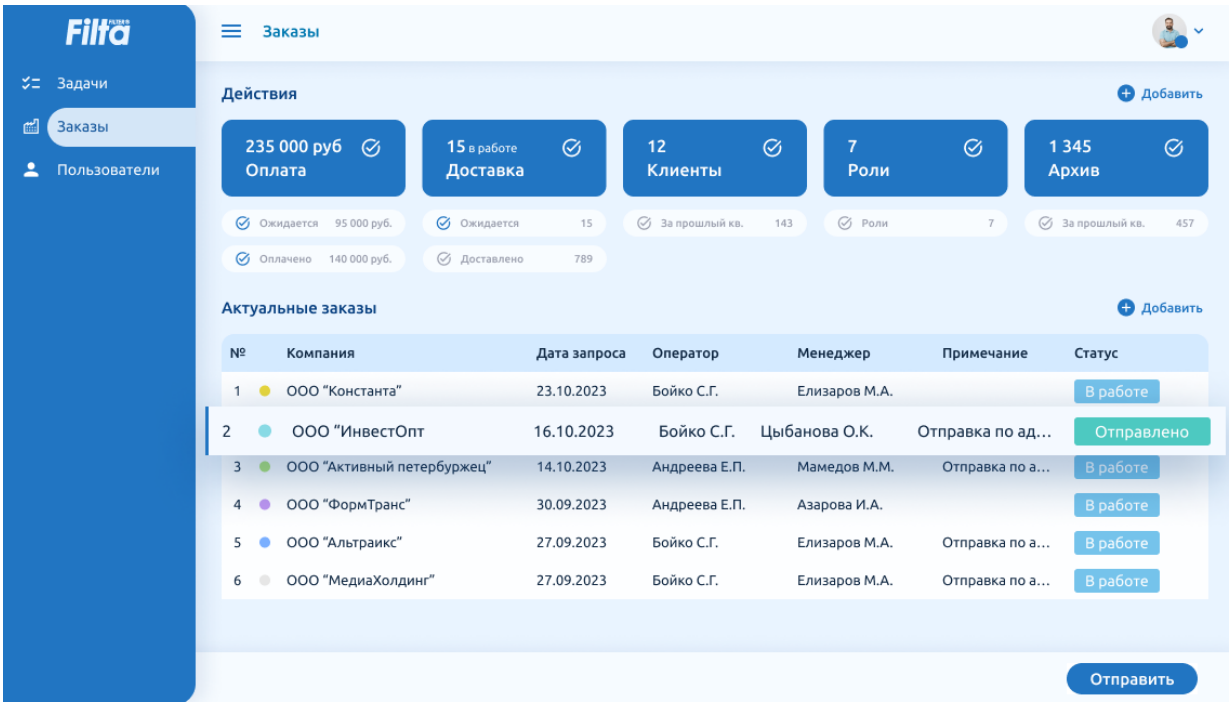

Рисунок 1 – Интерфейс раздела заказов

Generix OMS (система управления заказами) преобразует коммерческий заказ клиента в логистические заявки по видам потоков товаров и просчитывает перевозку по каждому отрезку цепи поставок с учётом нормативных сроков [15]. Основной функционал ИСУ:

- складской учёт визуализирует все доступные запасы и позволяет заключать сделки и управлять заявками;
- поддержка продаж автоматически формирует заказ с ближайшего к клиенту склада и создаёт заявки на сбор заказа с других складов;
- корректировка заказа предоставляет клиенту возможность вносить коррективы в заказ;
- контроль исполнения помогает отследить исполнение заказа на всех этапах цепи поставок.

К достоинствам можно отнести складской учёт, но ограничение функционала процесса управления заказами является недостатком данной ИСУ. На рисунке 2 представлено окно для создания заказа.

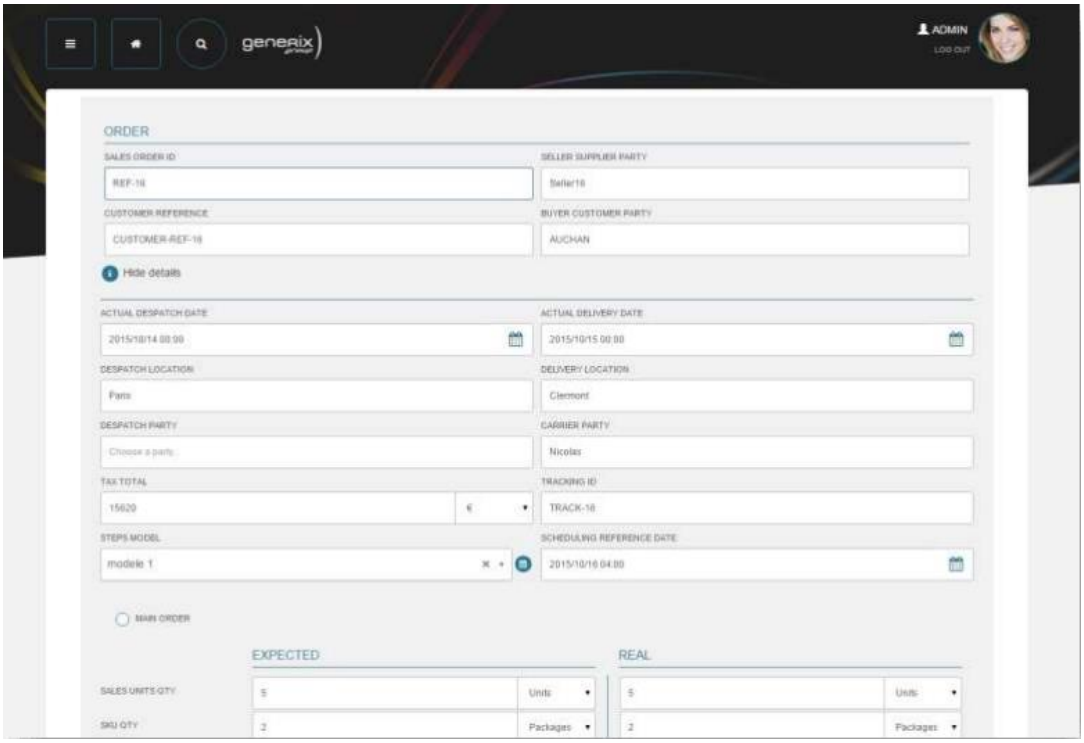

Рисунок 2 – Интерфейс окна создания заказов

Система управления заказами от Comindware – это гибкое решение для управления заказами, которое помогает автоматизировать процессы обработки заказов и сопутствующий документооборот, а также обеспечить контроль надлежащего выполнения заказов и повысить лояльность клиентов. Основные возможности данной ИСУ [17]:

– планирование и управление ресурсами в ходе исполнения заказов;
– редактирование, контроль статуса, отслеживание и отмена заказов;
– составление отчётности и аналитики;
– автоматическое формирование счетов, актов выполненных работ и других документов.

Среди достоинств можно отметить функционал для управления заказами. Недостатком данной ИСУ является стоимость лицензии. Окно с выборкой всех заказов представлено на рисунке 3.

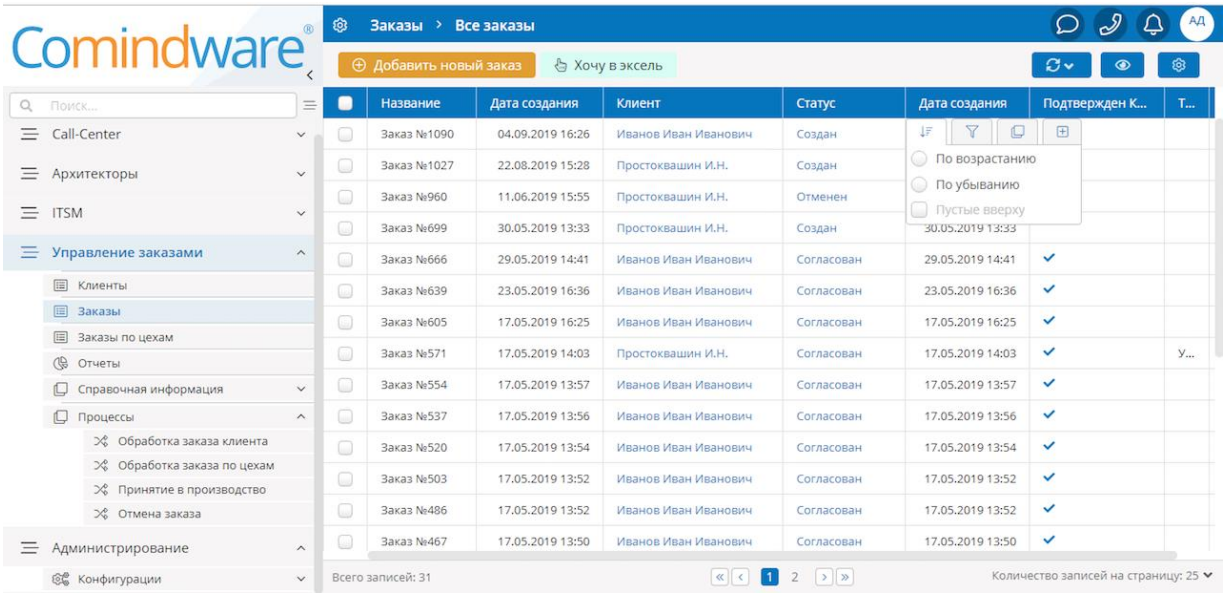

Рисунок 3 – Пользовательский интерфейс окна всех заказов

В таблице 2 показан сравнительный анализ рассмотренных аналогов ИСУ с оценкой их характеристик по шкале от 0 до 3, где 0 – не соответствует разработанным требованиям, 3 – полностью соответствует.

Таблица 2 – Сравнительный анализ аналогов

Наименование функционала	Filta OMS	Generix OMS	Comindware
Управление заказами	1	1	2
История взаимодействия с клиентом	0	1	1
Удобный интерфейс	2	0	1
Быстрый доступ к заказам	1	0	0
Документация сопровождения	0	0	0
Итого (макс. 15)	4	2	4

Проведённый анализ показал, что с одной стороны аналоги предлагают базовый набор функций, таких как управление заказами и клиентской информацией, генерация отчётов и выставление счётов.

Однако, некоторые из них могут иметь ограниченную масштабируемость или гибкость в настройке под конкретные потребности ремонтных мастерских. Интерфейс должен быть простым и легко понятным для пользователей разного уровня навыков.

С другой стороны, помимо базовых функций аналоги предлагают различные интеграции, отчётность, графики и аналитику, но их интерфейс остаётся сложным и запутанным, также этот функционал не всегда представляет прямой интерес в рамках ремонтной мастерской.

Из-за сложного и запутанного интерфейса навигация по системе не является логичной и интуитивной, а это в свою очередь затруднит работу будущих пользователей, а также потребует дополнительного времени на их обучение.

Таким образом, на основании проведённого анализа существующих аналогов программного обеспечения можно сделать вывод о необходимости разработки собственного решения, которое будет отвечать разработанным требованиям, сочетать в себе оптимальный набор функций, удобство использования и возможность гибкой настройки под специфические потребности ремонтных мастерских.

Выводы по главе 1

Для формирования требований на разработку ПО ИСУ заказами ремонтной мастерской были предприняты следующие шаги:

- учтена специфика работы ремонтной мастерской для определения назначения разрабатываемого ПО ИСУ;
- определены базовые функции и архитектура с учётом особенностей информационных систем управления заказами;
- разработаны требования по модели FURPS+ с учётом специфики обслуживания заказами ремонтной мастерской.

Проведён обзор и анализ существующих аналогов, который позволил выявить основные характеристики, преимущества и недостатки существующих решений, а также обосновать разработку собственного ПО ИС управления заказами ремонтной мастерской.

Глава 2 Проектирование программного обеспечения информационной системы управления заказами ремонтной мастерской

2.1 Разработка диаграммы вариантов использования программного обеспечения

Концепция – это определённый способ понимания и трактовки каких-либо явлений, основная точка зрения. На этапе определения концепции продукта проводится работа с его заказчиком(инвестором), целью которой является выработка единого видения будущего программного продукта.

Концепция программного обеспечения (ПО) информационной системы управления заказами ремонтной мастерской должна быть основана на обеспечении прозрачности, оперативности и удобства управления заказами клиентов, а также оптимизации рабочих процессов мастерской [4].

Для описания системы на концептуальном уровне воспользуемся диаграммой вариантов использования, которая входит в состав UML (Unified Modeling Language, унифицированный язык моделирования). Она позволит отразить отношения существующие между акторами и прецендентами [10].

Основная цель диаграммы заключается в том, чтобы служить универсальным инструментом, позволяющим заказчику, конечному пользователю и разработчику взаимодействовать и обсуждать функциональность и поведение системы совместно [11].

Диаграмма вариантов использования обработки заказа ремонтной мастерской показана на рисунке 4.

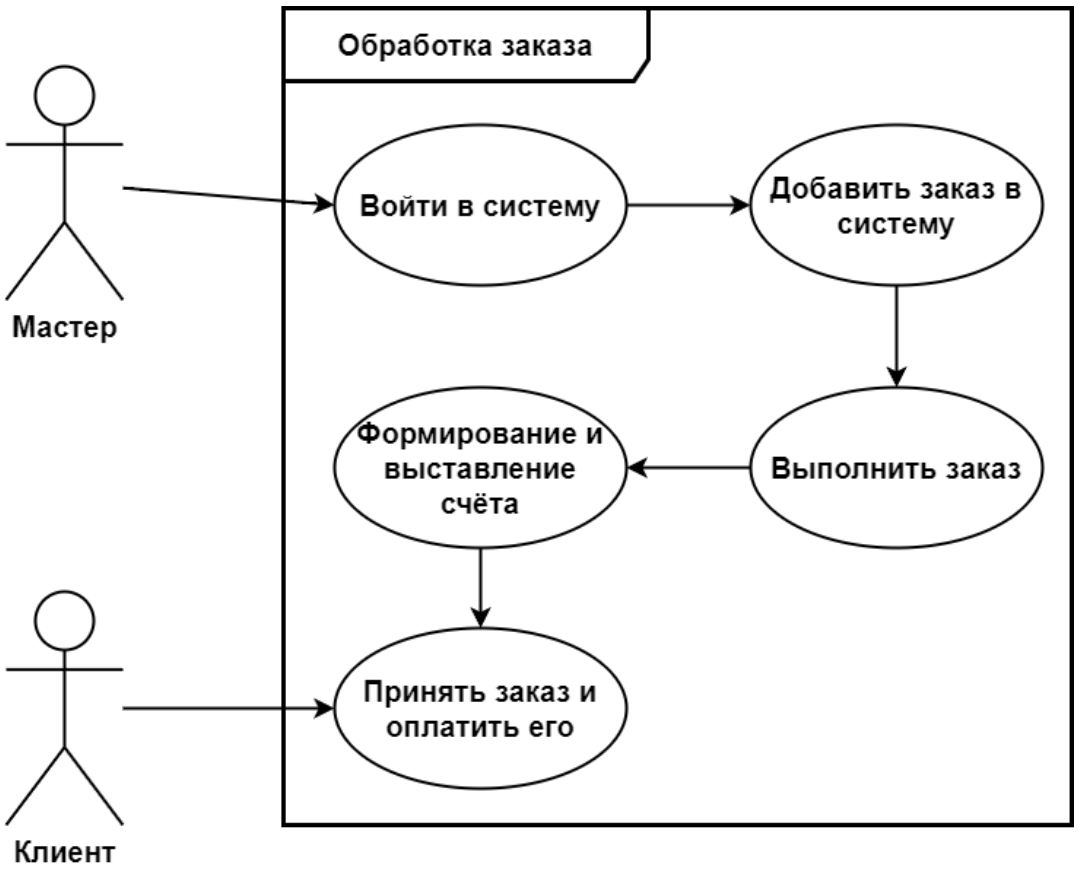

Рисунок 4 – Основные процессы при обработке заказа

Ранее сформированные функциональные требования должны полностью покрывать основные процессы при обработке заказа.

2.2 Описание вариантов использования

Чтобы более подробно понять, как ПО ИСУ будет использоваться конечными пользователями, необходимо описать варианты использования.

Описание поможет определить, зачем продукт нужен, какие цели и задачи он должен решать. Каждое описание включает в себя:

- описание, представляющее собой сжатый обзор сценария использования;
- пред- и постусловия для сценариев;
- основной поток событий, который описывает взаимодействие между

системой и участниками, приводящее к достижению цели основного участника;

- альтернативные поток событий, описывающий обработку ошибок и нештатных ситуаций;
- подчинённые потоки, которые облегчают описание основного и альтернативных потоков (таблица 3) [6].

Таблица 3 – Вариант использования процесса управления заказом

Краткое описание	Этот сценарий использования описывает процесс обработки заказов
Предусловия	Включить ЭВМ. Клиент с деталями заказа.
Основные потоки сценария входа в систему	Данные потоки событий начинают своё выполнение с момента запуска системы: Пользователь запускает систему. Система запрашивает логин и пароль. Пользователь вводит логин и пароль. Система проверяет корректность введённого логина и пароля и отображает основное меню, предоставляя доступ к функциям системы. Пользователь обрабатывает заказ.
Основные потоки сценария регистрации клиента	Если клиент не зарегистрирован в системе, то пользователь добавляет данные клиента (ФИО, номер телефона), чтобы создать заказ.
Основные потоки сценария поиска клиента	Если клиент зарегистрирован в системе, пользователь ищет клиента по номеру телефона, чтобы создать заказ.
Основные потоки сценария создания заказа	Пользователь добавляет данные заказа клиента в систему (статус заказа, статус оплаты, описание, комментарий). Система автоматически добавляет id клиента, номер заказа и дату создания заказа.
Основные потоки сценария работы с заказом	Пользователь находит нужный заказ для просмотра или изменения статуса заказа, статуса оплаты, даты, описания и комментария После выполнения заказа пользователь изменяет статус заказа, например «выполнен», и статус оплаты, например «не оплачен». Пользователь формирует счёт на оплату и после оплаты изменяет статус заказа на «выполнен».

Продолжение таблицы 3

Краткое описание	Этот сценарий использования описывает процесс обработки заказов
Альтернативный поток. Некорректный логин или пароль	Система выявляет, что введённая комбинация логина и пароля неверна, Система уведомляет пользователя об ошибки и в зависимости от некорректных данных предлагает заново ввести логин и/или пароль. Пользователь либо заново вводит логин и/или пароль, либо отказывается от входа в систему. Система, в зависимости от выбора пользователя, либо возвращается к основному потоку, либо сценарий завершается.
Постусловия	При корректно введённом логине и пароле пользователю предоставляется доступ к основному функционалу системы, который зависит от его роли. Если введённые данные некорректны, то будет гарантировано, что доступ к основному функционалу предоставлен не будет.

Для варианта использования построим UML диаграмму деятельности или диаграмму активностей (activity diagram) (рисунок 5), которая позволит проиллюстрировать поток процессов проходящих через систему, т. е. наглядно показать как поток управления переходит от одной деятельности к другой.

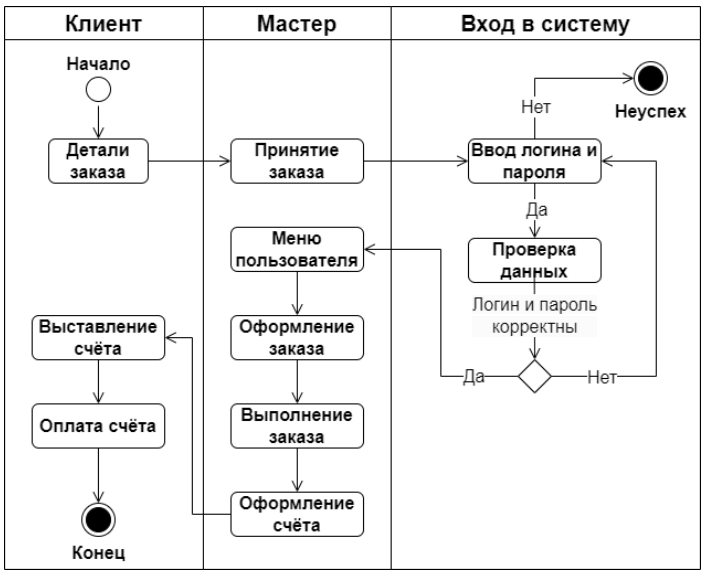

Рисунок 5 – Диаграмма деятельности для процесса обработки заказа

Если заказ клиента выполнен, то мастер входит в систему посредством ввода логина и пароля, чтобы начать процесс обработки заказа.

В случае неправильного ввода логина и пароля система предложит ввести их заново. Пользователь может отказаться от доступа и система завершит свою работу.

После построения диаграммы вариантов использования, её описания и построения диаграммы деятельности можно перейти к выбору методологии разработки.

2.3 Выбор методологии проектирования программного обеспечения

Термины «методология проектирования» и «методология разработки» используются как синонимы и обозначают набор принципов и подходов, используемых при разработке программного обеспечения. Методология задаёт структуру для работы и определяет этапы создания программы и роли в команде. От правильного выбора методологии зависит успех реализации программного обеспечения, эффективность процесса разработки, конечное качество продукта, а также способность адаптироваться к изменяющимся условиям и потребностям рынка [9].

К основным методологиям разработки ПО относятся Waterfall, RAD (Rapid Application Development), Agile, Lean, Scrum, XP (Extreme Programming), Prototype model, FDD (Feature Driven Development) [3].

Рассмотрим первые шесть методологий и их применимость к разработке информационной системы управления заказами ремонтной мастерской. Для того, чтобы обосновать выбор методологии проведём обзор особенностей каждой из методологий и составим таблицу их сравнения, в которой укажем их преимущества и недостатки.

Каскадная или водопадная модель (Waterfall, водопад) – линейный подход к разработке, который происходит последовательно, шаг за шагом, от

определения требований до развёртывания и поддержки. Имеет строгий и структурированный процесс, где каждая фаза проекта должна быть завершена перед началом следующей. Требования чётко сформулированный и строго зафиксированы. Всё это обеспечивает прозрачность процесса, выстраивает чёткие рамки, облегчает распределение времени и управление ресурсами, но делает процесс разработки менее гибким при изменениях требований.

Быстрая разработка приложений RAD (Rapid Application Development) – это итеративная методология, которая стремится к созданию приложений за счёт быстрого цикла разработки и применения прототипирования. Метод основан на активном вовлечения пользователей и их обратной связи, а также на совместной работе всех членов команды и сокращения формальной документации, что позволяет ускорить выполнение проекта и избежать проблем после запуска. Этот методологический подход уделяет особое внимание быстрому созданию итеративных прототипов, которые затем модифицируются и дорабатываются на основе обратной связи от заказчика.

Методология Agile – итеративный подход к разработке, в котором основное внимание уделяется гибкости, т. е. готовности к изменениям вместо следования первоначальному плану, работающему продукту вместо исчерпывающей документации, сотрудничеству с заказчиком, постоянному улучшению и оптимизации процессов для реализации качественного проекта.

Lean Development бережливая разработка – методология разработки основанная на Agile. Основной принцип направлен на исключении лишних задач из процесса для повышения эффективности разработки. Разработчики избегают ненужную работу и делают акцент только на необходимых и важных задачах, тем самым повышая качество продукта.

Scrum является системой управления проектами, основанной на принципах Agile. Проект разбивается на короткие временные интервалы, называемые спринтами, и на каждом спринте команда сосредотачивается на выполнении определённого набора задач. Данная методика помогает командам адаптироваться к меняющимся условиям рынка и потребностям

пользователей, структурировать работу и управлять ею на основе набора ценностей, принципов и практик.

XP (Extreme Programming) методика акцентирует внимание на качестве кода и реакции на изменения в требованиях заказчика. Она придерживается принципов, таких как парное программирование, тестирование на каждом этапе разработки, постоянная обратная связь со стороны заказчика и короткие итерации разработки. XP также поддерживает коллективное владение кодом и интеграцию изменений в код как можно чаще.

Для наглядности достоинств и недостатков построим таблицу сравнения (таблица 4).

Таблица 4 – Сравнение методологий

Методология	Размер проекта	Достоинства	Недостатки
Waterfall	Любой	Точная структура и документация проекта	Низкая вовлечённость заказчика, дорогой и сложный процесс устранения ошибок
RAD	Маленький, средний	Быстрая разработка, активное взаимодействие с заказчиком	Не подходит для больших и сложных проектов, команда должна состоять из опытных разработчиков
Agile	Любой	Гибкость, активное взаимодействие с заказчиком	Команда должна состоять из опытных разработчиков, минимум документации
Lean	Маленький, средний	Высокая эффективность проекта, устранение лишних процессов	Не подходит для больших проектов, минимум документации
Scrum	Маленький, средний	Прозрачность процессов проекта, постоянный анализ проделанной работы	Активное взаимодействие между членами команды требует наличия опытных разработчиков
XP	Маленький, средний	Быстрая разработка, высокое качество продута, постоянное улучшение	Большой объём тестирования и проверки кода, быстрый темп работы

Бережливая разработка Lean в полной мере будет соответствовать

разработке ПО ИС ремонтной мастерской, т. к. она нацелена на исключение лишних деталей и оптимизацию процессов. Такой подход поможет повысить эффективность проекта и качество будущего продукта, уменьшить использование ресурсов и в целом обеспечить соответствие результатов работы сформулированным требованиям [25].

2.4 Логическое проектирование программного обеспечения

Архитектура программного обеспечения – это структурный дизайн программной системы, описывающий её структурные элементы, механизмы взаимодействия, принципы и паттерны проектирования. Она определяет общую организацию системы и обеспечивает основу для её разработки, развёртывания, поддержки и сопровождения.

Структурные элементы – это компоненты системы, такие как модули и классы, а также их свойства и отношения друг с другом.

Механизмы взаимодействия – это способы, с помощью которых компоненты системы обмениваются информацией и координируют свою работу. Это может быть API, протоколы и различные события.

Для удовлетворения проектируемой системы атрибутам качества применяются архитектурные шаблоны (паттерны). Существует множество паттернов. Каждый из них имеет свои преимущества и недостатки [18], [23].

Исходя из базовой функциональности можно понять, что ПО будет иметь интерфейс, базу данных и логику, которая будет управлять заказами. Все модули программы могут находиться вместе и быть тесно связаны друг с другом, т. е. иметь монолитную архитектуру.

Монолитная архитектура представляет собой иерархическую структуру, где каждый слой приложения отвечает за определённую функциональность, такую как взаимодействие с базой данных (БД), ведение журнала действий и пользовательский интерфейс. Это способствует простоте проведения тестирования от начала до конца (e2e) и упрощает процесс развёртывания

системы.

У монолитной архитектуры есть несколько видов шаблонов проектирования:

– MVC (Model View Controller, Модель Представление Контроллер);
– MVP (Model View Presenter, Модель Представление Представитель);
– MVVM (Model View ViewModel).

Шаблон MVC (рисунок 6) обычно используется для разработки пользовательских интерфейсов и разделяет соответствующую программную логику на три взаимосвязанных элемента. Этими элементами являются внутренние представления информации (модель), интерфейс (представление), который представляет информацию пользователю и принимает её от него, и контроллер, связывающий эти два элемента [1].

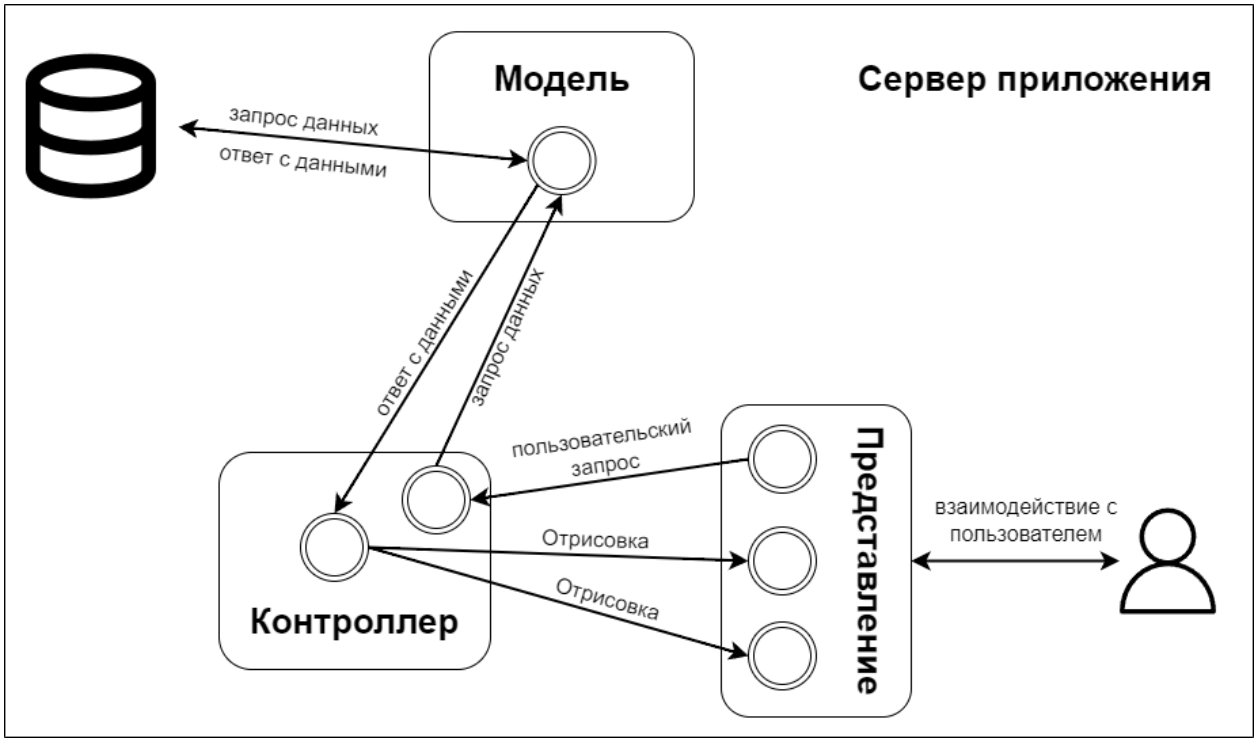

Рисунок 6 – Шаблон MVC относительно пользователя

MVP является производной от MVC и используется в основном для создания пользовательских интерфейсов [20].

Шаблон MVVM происходит от MVC и применяется в основном для разработки пользовательского интерфейса [14].

MVC способствует эффективному разделению логики приложения. Модульность системы позволит разделить функциональность на отдельные компоненты, обеспечивая лёгкость поддержки и возможного масштабирования.

Для логического проектирования архитектуры ПО ИСУ необходимо определить основные модули, их взаимосвязь и взаимодействие. Затем для каждого модуля определяются данные и функции. Далее можно построить диаграмму пакетов или классов по выбранному шаблону проектирования. Выше было принято решение проектировать систему по модели MVC (рисунок 7).

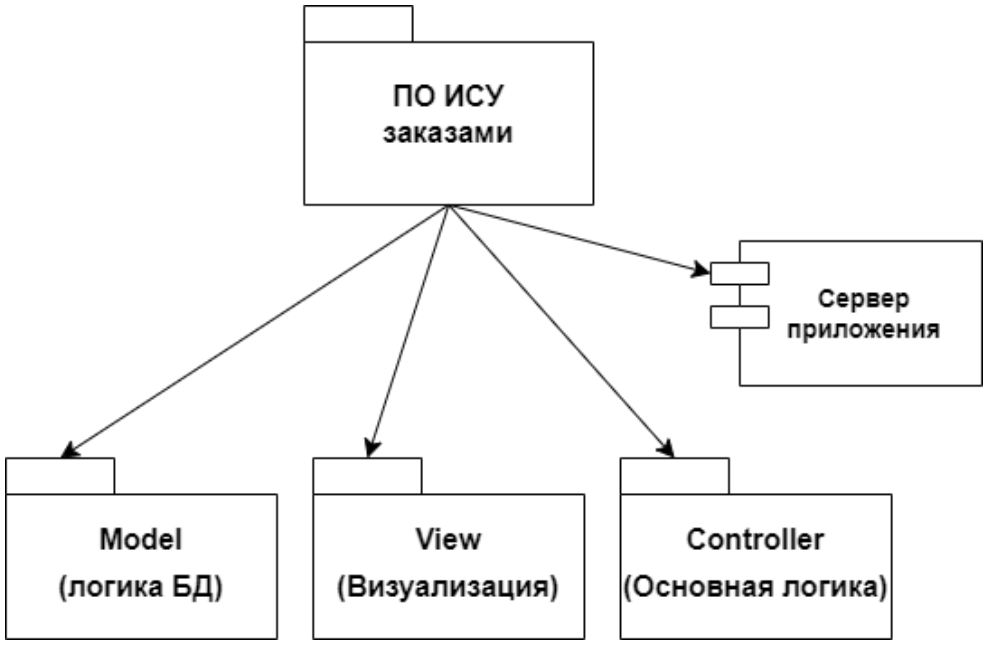

Рисунок 7 – Диаграмма пакетов по шаблону MVC

На рисунке 8 показано, что входит в каждый из пакетов.

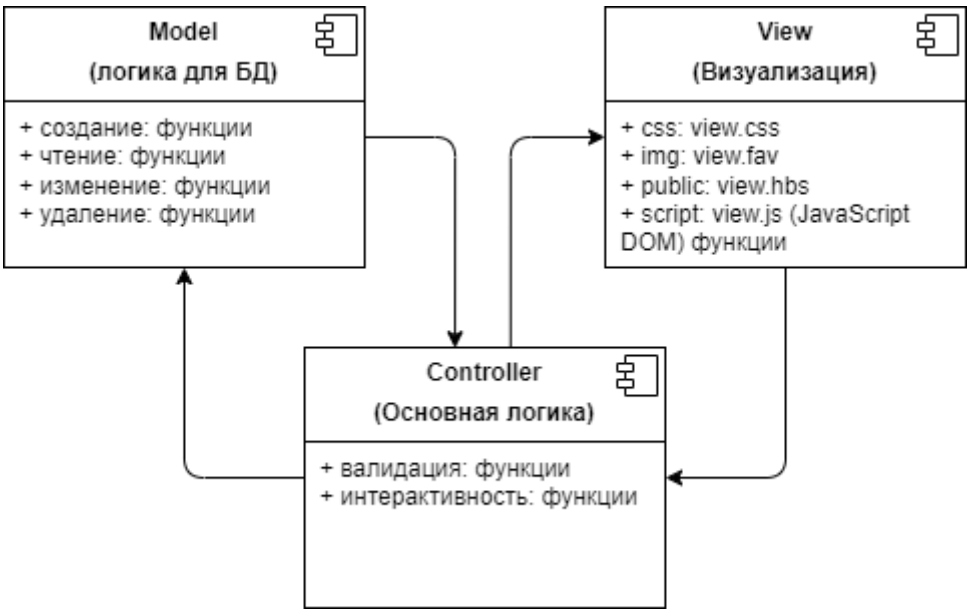

Рисунок 8 – Диаграмма компонентов по шаблону MVC

Функции, отвечающие за логику программы на различных уровнях, не заключены в классы, поэтому диаграмма классов отсутствует.

Архитектура ПО является основой для успешного проекта, поскольку она определяет общую структуру и организацию системы, влияя на её гибкость, расширяемость и эффективность.

2.5 Проектирование модели данных

Чтобы обеспечить надёжное хранение и быстрый доступ к информации необходимо использовать базы данных (БД).

Базы данных (БД) предоставляют эффективный механизм для хранения, организации и управления данными. В программном обеспечении ИСУ БД нужна, чтобы обеспечить хранение информации о пользователях системы и клиентах ремонтной мастерской, а также обеспечивать целостность этих данных. Рассмотрим два основных вида БД, реляционные и нереляционные [19].

В реляционных БД имеется несколько таблиц между которыми

проставлены отношения [13].

В каждой таблице имеются строки, которые называются записями и столбцы. Записи содержат информацию о сущностях, например пользователях ИС, клиентах мастерской, информации о заказах и не только.

Столбцы в отличие от строк содержат информацию одного типа, которая присутствует в каждой строке, например описание заказов для всех клиентов.

В нереляционных БД имеется одна таблица со всеми данными и не используется табличная схема строк и столбцов в отличие от большинства традиционных систем баз данных. В таких БД применяется модель хранения, оптимизированная под конкретные требования типа хранимых данных.

Эффективное управление данными является краеугольным камнем для разработки функционально полноценных ИСУ заказами. Логическая структура БД определяет способ организации и хранения информации, обеспечивая эффективный доступ к данным, их целостность и безопасность.

Для построения логической структуры базы данных необходимо учесть вид БД и данные, которые будут использованы.

ИСУ будут пользоваться мастера, которые будут управлять заказами ремонтной мастерской с помощью таких операций как создание заказов и редактирование заказов клиентов, выборка всех заказов и т. п., т. е. должна присутствовать информация о ФИО и номере телефона клиента, статусе, дате и описании заказа.

Из вышесказанного можно сделать вывод, что будет удобно использовать реляционную БД с тремя таблицами:

- таблица «Сотрудники» будет содержать уникальный идентификатор, логин, почту и пароль;
- таблица «Клиенты» будет содержать уникальный идентификатор, ФИО и телефон;
- таблица «Заказы» будет содержать уникальный идентификатор, идентификатор клиента и сотрудника, статус заказа, статус оплаты, дату и описание.

Представим логическую модель данных с помощью UML диаграммы (рисунок 9).

Рисунок 9 – Спроектированная логическая модель данных

Каждый сотрудник может создать много заказов, т. к. у каждого клиента может быть много заказов. Отсюда и получаются связи один ко многим.

Выводы по главе 2

Благодаря логическому проектированию программного обеспечения можно создать качественный и надёжный в использовании продукт, который соответствует требованиям пользователей и бизнеса, легко расширяется, обновляется и поддерживается, а также обеспечивает удобный и интуитивно понятный интерфейс.

Глава 3 Реализация и тестирование программного обеспечения ИС

3.1 Выбор средств разработки ПО

Этап выбора среды разработки и технологического стека оказывает принципиальное влияние на успешное воплощение проекта, поскольку он формирует основу инструментов, языков программирования (ЯП), фреймворков и прочих технологий, которые будут задействованы в процессе разработки программного продукта, а также способствует оптимальным условиям разработки ПО, определяя эффективность и качество конечного результата.

От правильного подхода к определению среды разработки зависит не только удобство работы разработчиков, но и способность программного продукта адаптироваться к изменяющимся требованиям рынка и пользователям, т. к. выбор конкретных инструментов и технологий имеет долгосрочные последствия для проекта.

Из вышесказанного следует, что для реализации проекта требуется выбрать:

- язык программирования;
- интегрированную среду разработки поддерживающую выбранный ЯП;
- модули и библиотеки для улучшения и ускорения процесса разработки.

Из составленных требований следует, что при реализации ПО ИСУ необходимо использовать ЯП JavaScript и СУБД SQLite. Соответственно ПО ИСУ будет разрабатываться посредством веб-технологий. Для браузерной версии можно использовать простой, доступный и понятный технологический стек HTML, CSS и JS:

- HTML (Hyper-Text Markup Language) язык гипертекстовый разметки документов. С помощью него выстраивают скелет будущих веб-

страниц;

- CSS (Cascading Style Sheets) каскадная таблица стилей. Представляет собой формальный язык описания внешнего вида HTML-документа;
- JS (JavaScript) это мультипарадигменный ЯП, который используется как для визуальной разработки так и для логической или серверной части веб-приложения.

Для ЯП JS воспользуемся следующими инструментами разработки:

- Node.js – это среда выполнения JavaScript кода, построенная на движке V8 Chrome, которая превращает узкоспециализированный ЯП JavaScript в ЯП общего назначения и обеспечивает возможность запускать JS-код на сервере. Данная среда поддерживает асинхронное программирование, событийно-ориентированную архитектуру и модульность. Благодаря использованию движка V8 и асинхронной модели выполнения, Node.js обеспечивает высокую производительность при обработке запросов;
- Express.js – это минималистичный и гибкий веб-фреймворк для Node.js, который облегчает создание веб-приложений. Express строится поверх Node.js и предоставляет инструменты для обработки HTTP-методов, маршрутизации, работу с шаблонами и многое другое;
- SQLite – это компактная встраиваемая СУБД, которая обеспечивает локальное хранение данных без необходимости отдельного сервера. Она предлагает множество функций, типичных для реляционных баз данных, таких как поддержка SQL запросов, транзакции, ограничения целостности данных и многое другое.

Для реализации основной логики приложения требуется создать несколько страниц, которые позволят визуализировать и протестировать функционал, а также выявить потенциальные улучшения и корректировки.

Реализация ПО ИСУ по методологии Lean и модели MVC обеспечит

прозрачность процесса разработки, улучшение коммуникации между командой разработчиков и заказчиком, а также минимизирует риски, связанные с непониманием требований и ожиданиями заказчика. Благодаря бережливой разработке можно оперативно и эффективно адаптировать программное обеспечение к потребностям пользователей и сделать процесс разработки более гибким и результативным.

Таким образом, грамотный выбор среды разработки и технологического стека является неотъемлемой частью стратегического планирования и успешной реализации проекта по разработке программного обеспечения.

3.2 Разработка логической структуры базы данных

Преимущество SQLite заключается в её простоте использования и установке. Поскольку SQLite является встраиваемой, она не требует отдельного сервера, и СУБД обычно представляет собой один файл, что облегчает её интеграцию в приложения. Однако во встраиваемом характере есть и недостаток, это ограничение в масштабируемости и производительности при работе с большими объёмами данных или высокими нагрузками.

При создании логической структуры БД важно учесть особенности бизнес-процессов компании, возможные изменения в будущем и кроме того, необходимо придерживаться принципов нормализации данных, чтобы избежать избыточности и несогласованности в дальнейшем.

Структура БД ПО ИСУ заказами будет состоять из трёх таблиц со следующими названиями и столбцами:

- таблица сотрудников системы (Staff) – уникальный идентификатор (УИД) (первичный ключ, целое число, не пустое значение, авто увеличение), логин (текст), почта (текст) и пароль (текст);
- таблица клиентов системы (Clients) – УИД (первичный ключ, целое число, не пустое значение, авто увеличение), ФИО (текст) и номер

телефона (текст);

– таблица заказов клиентов (Orders) – УИД (первичный ключ, целое число, не пустое значение, авто увеличение), УИД клиента (вторичный ключ таблицы «Clients» столбика УИД, целое число, не пустое значение), УИД сотрудника (вторичный ключ таблицы «Staff» столбика УИД, целое число, не пустое значение), статус заказа (текст), статус оплаты (текст), дата (текст), описание (текст).

Рассмотрим процесс проектирования и разработки интерфейса ПО ИСУ.

3.3 Проектирование и разработка пользовательского интерфейса

Интерфейс приложения позволяет пользователям взаимодействовать с продуктом, и его качество может определить, будет ли пользователь продолжать использовать программное обеспечение, или откажется от него в пользу конкурентов.

Проектирование пользовательского интерфейса – это не только процесс создания красивого дизайна, но и понимание потребностей и ожиданий пользователей, создание простой и интуитивно понятной структуры, а также обеспечение доступности для всех пользователей, вне зависимости от их способностей и ограничений.

Для создания дизайна интерфейса с использованием графических элементов, цветовой схемы и макетов воспользуемся онлайн-графическим редактором Figma [21].

При реализации интерфейса особое внимание будет уделено интуитивной понятности, доступности для пользователей различного уровня опыта, а также соответствию требованиям и ожиданиям потенциальных пользователей. Также будет заложена адаптивность для одинакового отображения интерфейса в разных, привлекательный и ненавязчивый дизайн, управление с клавиатуры (табуляция) и т. п.

На рисунке 10 представлен макет страницы всех заказов для выбранного

клиента:

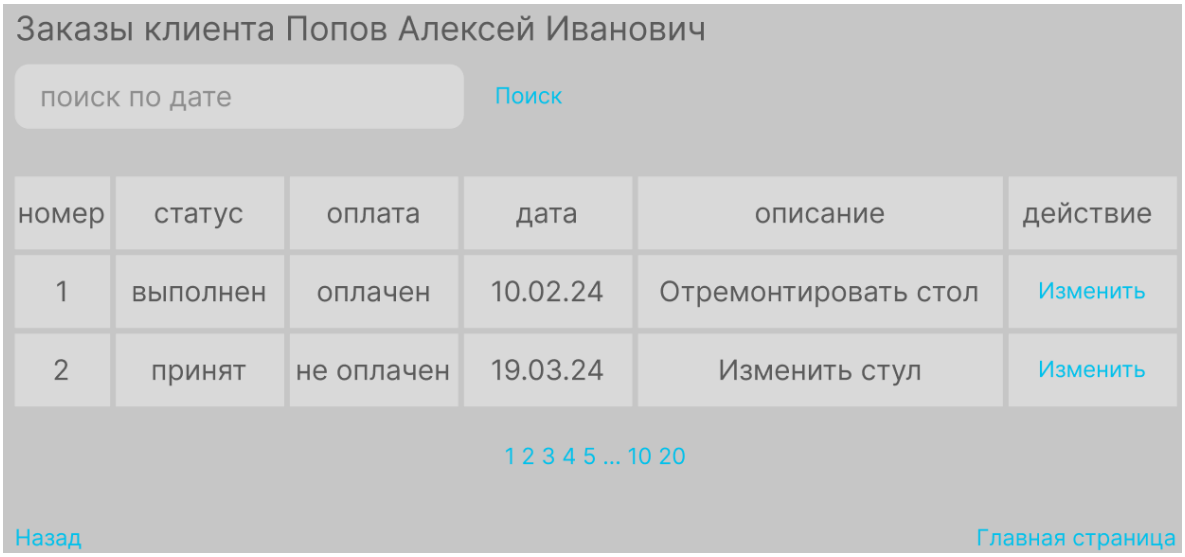

Рисунок 10 – Макет формы списка заказов клиента

Пользовательский интерфейс является мостом между пользователями и функционалом системы, определяя удобство и эффективность взаимодействия с ПО ИСУ.

3.4 Маршрутизация веб-страниц приложения

Правильная маршрутизация путей на сайте зависит от структуры веб-приложения и того, как разработчик хочет организовать доступ к различным страницам. В целом, есть два подхода:

- абсолютный путь, который подразумевает, что путь к определённой странице будет включать в себя все предыдущие сегменты. Это удобно, когда структура URL отражает иерархию навигации на сайте;
- относительный путь может быть предпочтительнее, если нужно предоставить прямой доступ к определённой странице, не зависимо от того, где находится пользователь на сайте.

Если важна чёткая иерархия и понимание пути пользователя по сайту, лучше использовать абсолютный путь. Если же приложение имеет множество

различных страниц и нужно упростить URL и сделать доступ к странице более независимым от контекста, можно использовать относительный путь.

Важно также учитывать SEO и удобство пользователей при выборе структуры URL. Длинные URL могут быть менее удобны для восприятия, но они могут лучше отражать структуру сайта, что полезно для поисковых систем. Короткие URL проще запомнить и вводить, но они могут быть менее информативными. Выбор должен соответствовать общей архитектуре сайта и потребностям пользователей.

ПО ИСУ заказами будет иметь следующую структуру веб-страниц:

- вход в систему – поля ввода (логин и пароль), кнопки (запомнить пользователя, восстановления и входа);
- восстановления пароля – поле ввода почты и кнопки возвращения назад и восстановить;
- установка нового пароля – поля ввода нового пароля и повторения нового пароля и кнопки отмена и восстановить;
- главное меню – кнопки личного кабинета, добавления клиента и списка всех клиентов;
- личный кабинет – кнопки изменения пароля, почты, возвращения назад и выхода из системы;
- изменения пароля – поля почты, старого пароля и нового пароля, кнопки изменение, возвращение назад и главного меню;
- изменения почты – поля старой почты и новой почты, кнопки изменение, возвращение назад и главного меню;
- добавление клиента – поля ФИО и номера телефона, кнопки добавить и возвращения назад;
- список клиентов – таблица с данными клиентов, поле для поиска по критерию, кнопки возвращения назад, поиска, сброса поиска, нового заказа и всех заказов;
- новый заказ – выпадающие меню статуса заказа и оплаты, ФИО

клиента, описания, кнопки создать, возвращения назад и главной страницы;

– все заказы – таблица заказов, поле для поиска по дате, кнопки возвращения назад, главной страницы, поиска, сброса поиска, изменения заказа;
– изменить заказ – выпадающие меню статуса заказа и оплаты, описания, кнопки изменить, возвращения назад и главной страницы.

Готовый интерфейс страницы всех заказов для выбранного клиента представлен на рисунке 11.

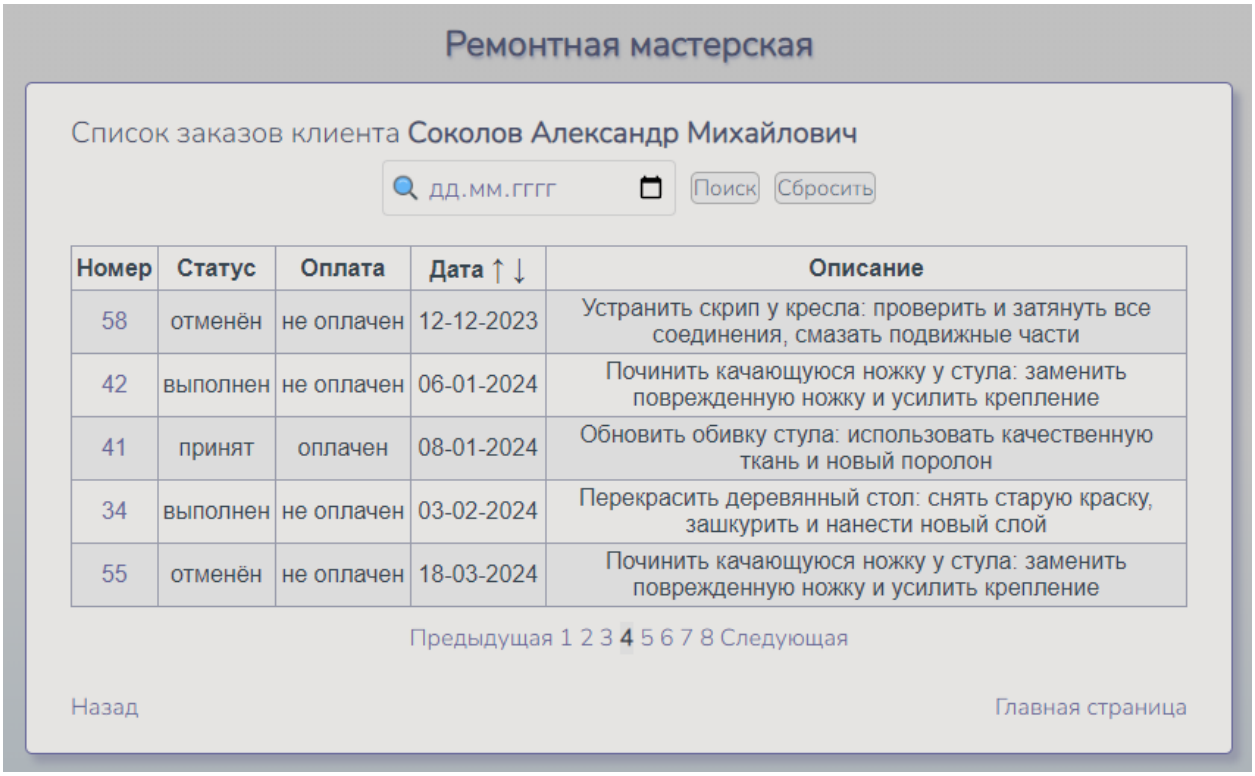

Рисунок 11 – Интерфейс формы списка заказов клиента

При нажатии на номер заказа открывается форма изменения выбранного заказа. Заказы можно сортировать по дате в порядке убывания или возрастания. Поиск заказа также осуществляется по дате.

В приложении А можно ознакомиться с другими основными формами

интерфейса ПО ИСУ заказами ремонтной мастерской.

3.5 Тестирование программного обеспечения информационной системы управления заказами

Тестирование – это проверка программы на соответствие требований. Во время процесса тестирования идёт поиск разницы между ожидаемым и фактическим поведением продукта.

Целью тестирования является предоставление актуальной информации о продукте на данный момент и поиск дефектов на ранних этапах разработки. Роль тестировщика – сделать продукт качественным.

Существует много видов и типов тестирования веб-приложений, но в основном в веб-разработке используются следующие виды (рисунок 12) [5].

Рисунок 12 – Пирамида тестирования

Модульные тесты – это автоматизированные тесты, которые проверяют

отдельные модули или компоненты программы на правильность их работы [24].

Также в модульных тестах можно реализовать мокирование базы данных (рисунок 13), которое будет включать создание поддельных версий функций или методов, которые взаимодействуют с БД, чтобы тесты не зависели от реальной БД. Это полезно для изоляции тестируемого кода и повышения стабильности и скорости тестов.

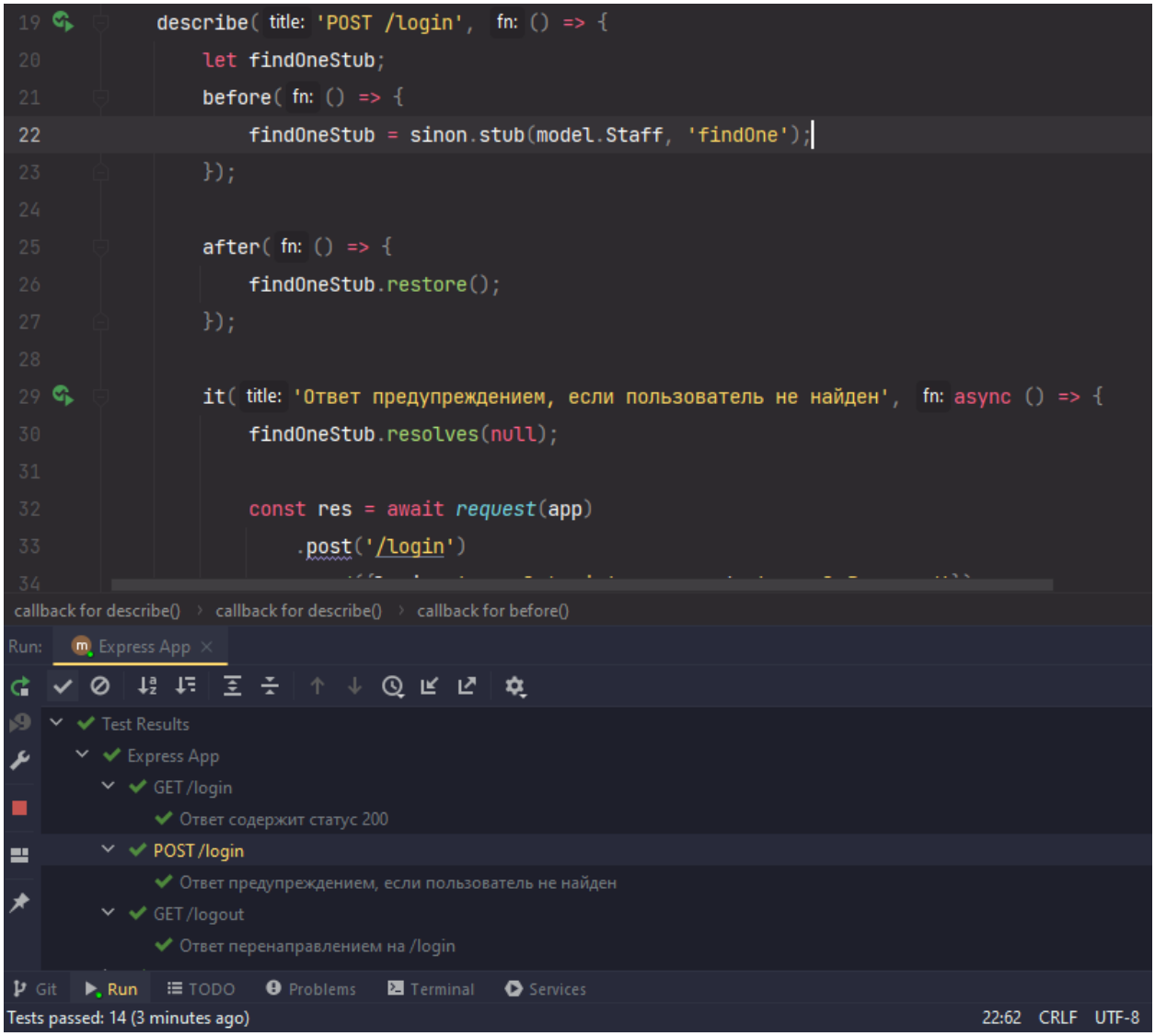

Рисунок 13 – Модульный тест с мокированием базы данных с помощью дополнительной библиотеки sinon

Для мокирования используется библиотека для JavaScript «sinon»,

предназначенная для создания тестовых двойников (test doubles) таких, как моки (mocks), стабы (stubs), шпионы (spies) и фейки (fakes).

Она широко используется в модульном тестировании для замены реальных объектов или функций на их тестовые аналоги с целью изоляции тестируемого кода и точного контроля за его поведением.

В целом модульные тесты имеют следующие преимущества:

- обнаруживают ошибки и дефекты в коде на ранних стадиях разработки;
- грамотно составленные тесты служат в качестве дополнительной документации;
- регулярное выполнение тестов поможет поддерживать высокое качество кода;
- когда код покрыт модульными тестами, разработчики могут безопасно вносить изменения и рефакторить код;
- добавление модульных тестов в процессы непрерывной интеграции (CI, Continuous Integration) позволит быстро обнаруживать и исправлять ошибки благодаря автоматическому выполнению тестов при каждом изменении кода;
- протестированный код обычно легче поддерживать.

Интеграция с другими системами отсутствует, поэтому следует воспользоваться ручным тестированием, которое будет заключаться в проверке графической составляющей, т. е. удобства использования, штатного поведения, т. е. положительных сценариев, и не штатного поведения, т. е. не ожидаемых программой входных данных.

Ручное тестирование делится на функциональное (проверка ключевой логики программы) и нефункциональное (дизайн, удобство использования, локализация и т. д.). Для ручного тестирования составляют чек-лист, тест-план тест-кейсы.

Также можно воспользоваться встроенной в браузер панелью

инструментов разработчика или devtools, открывающейся по нажатию кнопки на клавиатуре F12, которая имеет множество различных вкладок, названия которых варьируются в зависимости от выбранного браузера.

Рассмотрим панель инструментов разработчика в рамках браузера Google Chrome.

Основные вкладки devtools, которые помогут проверить работоспособность приложения:

- toggle device toolbar (панель инструментов устройства) – позволяет посмотреть, как выглядит страница на каком-либо устройстве, также можно вручную задать формат экрана;
- elements (элементы веб-страницы) – просмотр гипертекстовой разметки документов (HTML, структура веб-сайта) и применённую к этой разметке каскадную таблицу стилей (CSS);
- console (консоль) – при наличии ошибок в JS-коде будет выводить их с указанием файла и строкой кода. Также позволяет написать и исполнить код на ЯП JavaScript;
- network (Сеть) – выводит список истории запросов. Здесь можно очистить историю запросов, установить фильтр и тип для них, посмотреть адрес, статус кода и тело запросов и ответов, заголовки. Также можно скачать запросы (export HAR) и открыть их в программе анализатора трафика, установить скорость интернета (throttling, fast 3G, Slow 3G и offline);
- application (хранилище браузера) – это маленькая БД браузера содержит данные локальных и сессионных хранилищ, куки, кэш приложения, шрифтов и изображений, а также таблиц стилей.

Для тестирования API БД следует воспользоваться программой Postman. Данное ПО используется для разработки и тестирования API (Application Programming Interface). Она предоставляет удобный интерфейс для отправки HTTP-запросов к API и получения ответов [22].

На рисунке 14 представлен пример запроса к API с автотестами.

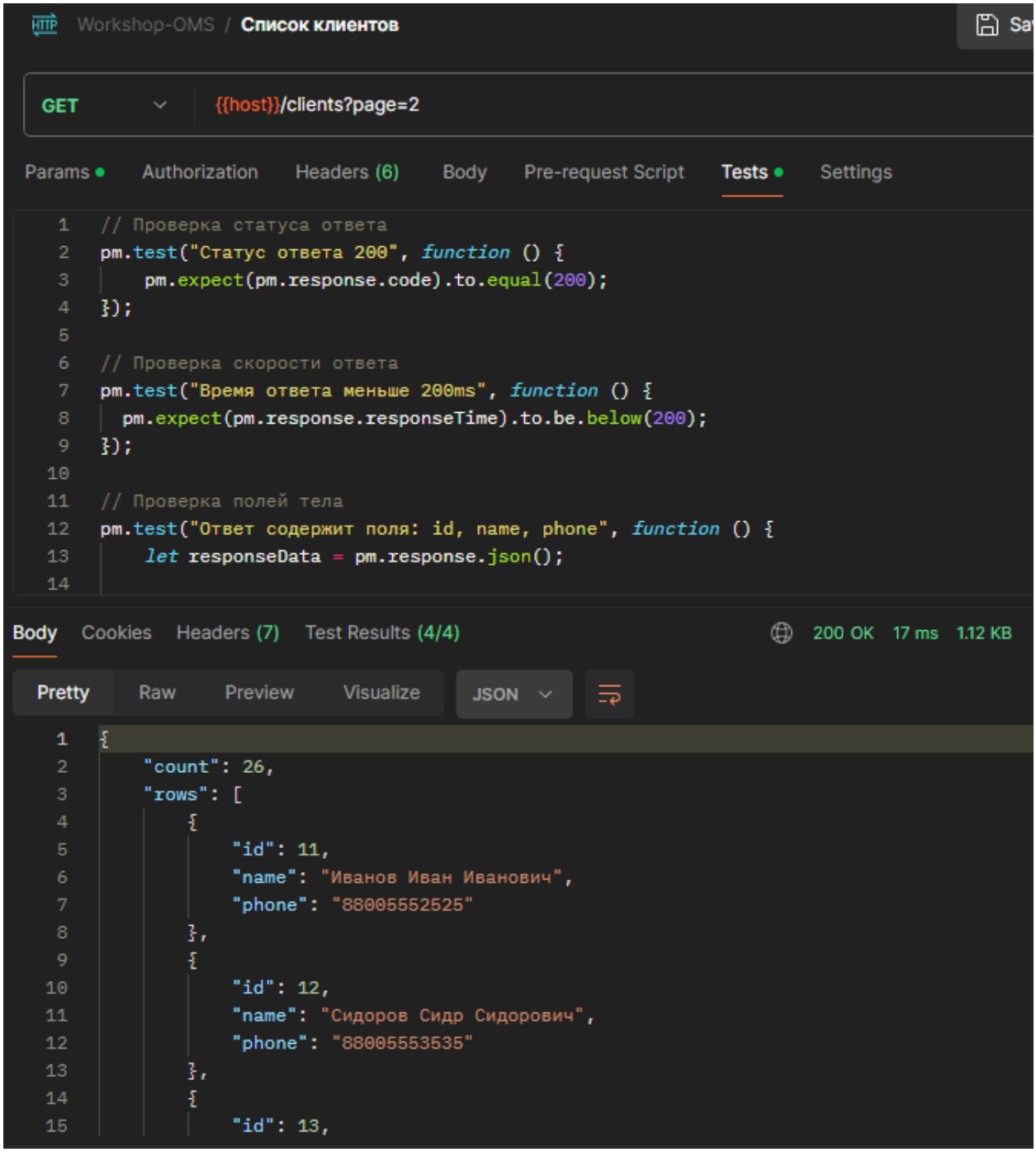

Рисунок 14 – Запрос с автотестами к API второй страницы списка клиентов

Основные функции Postman:

– отправка запросов различных типов запросов (GET, POST, PUT, DELETE и другие);

– отправка предварительных запросов для получения данных, которые можно сохранить в переменной коллекции и использовать для основного запроса;
– возможность писать автотесты на JavaScript для проверки ответов API;
– инструменты для мониторинга API;
– создание и поддержка документации для API;
– организация запросов в коллекции для структурированного подхода к тестированию.

Рассмотрим процесс доработки ПО.

3.6 Доработка программного обеспечения

В ходе тестирования веб-приложения были проведены различные виды тестов, включая модульное тестирование, ручное тестирование, и тестирование API с использованием Postman.

На основании полученных результатов тестирования были внесены следующие улучшения:

– был убран повторяющийся код (DRY, Don't Repeat Yourself). В процессе модульного тестирования были выявлены участки кода, где логика дублировалась. Код был рефакторирован, и повторяющиеся участки были заменены общими функциями и методами. Это позволило значительно улучшить читаемость кода и облегчить его поддержку;
– были обнаружены неэффективные функции и запросы к БД, которые приводили к увеличенному времени отклика системы. Поэтому была проведена оптимизация алгоритмов и реорганизация запросов, что позволило снизить нагрузку на сервер и ускорить выполнение операций;

В процессе ручного тестирования были выявлены проблемы, связанные с навигацией и организацией файлов проекта. Структура проекта была переработана, что улучшило логику размещения файлов и папок.

В результате тестирования пользовательских сценариев было решено внедрить динамические страницы, что позволило улучшить пользовательский опыт. Это изменение дало возможность пользователям взаимодействовать с приложением более гибко и получать актуальную информацию без необходимости полной перезагрузки страницы.

В ходе тестирования с помощью Postman были выявлены недостатки в обработке ошибок, которые могли привести к некорректному поведению приложения. Была разработана и внедрена более надёжная система обработки ошибок, что позволило улучшить устойчивость и надёжность системы.

Проведённое тестирование и последующее внедрение улучшений позволили значительно повысить качество веб-приложения. Убран дублирующийся код, оптимизированы функции и запросы, улучшена структура проекта и созданы динамические страницы. В результате тестирования API удалось оптимизировать работу с запросами и улучшить обработку ошибок. Эти меры обеспечили более стабильную и эффективную работу приложения, что положительно сказалось на общем пользовательском опыте.

Выводы по главе 3

В ходе реализации и тестирования ПО ИСУ были выполнены следующие шаги:

- выбраны средства разработки;
- разработана логическая структура базы данных;
- отлажено ПО ИСУ на разных уровнях тестирования с дальнейшим внесением улучшений на основе результатов тестирования.

Выбор средств разработки программного обеспечения был основан на требованиях проекта и предпочтениях разработчика, что позволило

эффективно приступить к работе над ПО ИСУ.

Разработка логической структуры БД была проведена с учётом требований к хранению и обработке данных в ИСУ, что обеспечило эффективное функционирование системы.

Проектирование и разработка пользовательского интерфейса включали в себя создание удобного и интуитивно понятного интерфейса для конечных пользователей, что повысило удовлетворённость пользователей и улучшило их работу с системой.

Маршрутизация веб-страниц приложения была организована таким образом, чтобы обеспечить удобное перемещение пользователей по функциональным частям приложения, что способствовало повышению его пользовательской доступности.

Отладка программного обеспечения на разных уровнях тестирования была проведена для выявления и исправления ошибок и недочётов в работе системы, что обеспечило её стабильную и безопасную работу.

Наконец, улучшения на основе результатов о тестировании позволили оптимизировать процессы разработки и повысить качество программного обеспечения, что является хорошим критерием успешного завершения проекта.

Заключение

В ходе выполнения данного исследования были рассмотрены главные аспекты разработки программного обеспечения информационной системы управления заказами ремонтной мастерской.

Главы работы, охватывающие постановку задачи на разработку ПО ИСУ, проектирование программного обеспечения, а также реализацию и отладку, представляют собой важные этапы в создании надёжного и эффективного программного обеспечения информационной системы управления заказами.

Практическая значимость выполненной работы заключается в улучшении бизнес-процессов ремонтной мастерской благодаря оптимизации процесса управления заказами.

В целом, выполнение поставленных задач на разработку программного обеспечения информационной системы для ремонтной мастерской позволило не только углубить знания в области разработки ПО, но и создать функциональную и надёжную систему, соответствующую основным целям и задачам, которые необходимы для эффективного управления заказами ремонтной мастерской.

Отладка программного обеспечения на разных уровнях тестирования была проведена для выявления и исправления ошибок и недочётов в работе системы, что обеспечило её стабильную и безопасную работу.

Наконец, улучшения на основе результатов о тестировании позволили оптимизировать процессы разработки и повысить качество программного обеспечения, что является хорошим критерием успешного завершения проекта.

Дальнейшее развитие данной системы следует продолжить на основании отзывов пользователей системы. Это позволит внедрить функционал, который будет действительно важен для ремонтных мастерских в целом.

Список используемой литературы и используемых источников

1. Архитектурные решения информационных систем : учебник для вузов / А. И. Водяхо, Л. С. Выговский, В. А. Дубенецкий, В. В. Цехановский. — 3-е изд., стер. — Санкт-Петербург : Лань, 2022. — 356 с. — ISBN 978-5-507-44710-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/254624 (дата обращения: 20.04.2024). — Режим доступа: для авториз. пользователей.

2. Богомолов, А. В. Методические рекомендации по подготовке, выполнению и оформлению курсовых работ по дисциплине «Проектирование информационных систем» для бакалавров, обучающихся по направлению подготовки 09.03.03 «Прикладная информатика» : методические рекомендации / А. В. Богомолов, Э. А. Игнатьева, К. Н. Фадеева. — Чебоксары : ЧГПУ им. И. Я. Яковлева, 2022. — 48 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/354065 (дата обращения: 13.04.2024). — Режим доступа: для авториз. пользователей.

3. Восемь лучших методологий разработки ПО в 2024 году. URL: https://www.purrweb.com/ru/blog/metodologii-dlya-razrabotki-po/ (дата обращения: 14.04.2024).

4. Зубкова, Т. М. Технология разработки программного обеспечения : учебное пособие / Т. М. Зубкова. — Оренбург : ОГУ, 2017. — ISBN 978-5-7410-1785-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/110632 (дата обращения: 13.04.2024). — Режим доступа: для авториз. пользователей.

5. Игнатьев, А. В. Тестирование программного обеспечения / А. В. Игнатьев. — 3-е изд., стер. — Санкт-Петербург : Лань, 2023. — 56 с. — ISBN 978-5-507-45425-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/269873 (дата обращения: 27.04.2024). — Режим доступа: для авториз. пользователей.

6. Кафедра системного программирования ВМК МГУ. Моделирование системы обработки заказов. URL: http://sp.cs.msu.ru/ooap/exerb2014.html (дата обращения: 14.04.2024).

7. Классификация требований к программным системам. URL: https://dit.isuct.ru/Publish_RUP/core.base_rup/guidances/concepts/requirements_62E28784.html (дата обращения 13.04.2024).

8. Машкин, А. В. Технология разработки программного обеспечения : учебное пособие / А. В. Машкин. — Вологда : ВоГУ, 2014. — 75 с. — ISBN 978-5-87851-526-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/93087 (дата обращения: 13.04.2024). — Режим доступа: для авториз. пользователей.

9. Методологии управления проектами: 12 популярных подходов. URL: https://asana.com/ru/resources/project-management-methodologies (дата обращения: 14.04.2024).

10. Нафикова, А. Р. Объектно-ориентированный анализ и проектирование программного обеспечения на языке UML : учебное пособие / А. Р. Нафикова. — Уфа : БГПУ имени М. Акмуллы, 2022. — 118 с. — ISBN 978-5-907475-48-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/219221 (дата обращения: 14.04.2024). — Режим доступа: для авториз. пользователей.

11. Петрова, О. Б. Разработка и анализ требований проектирования программного обеспечения: практикум : учебное пособие / О. Б. Петрова. — Санкт-Петербург : СПбГУТ им. М.А. Бонч-Бруевича, 2022. — 37 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/279218 (дата обращения: 14.04.2024). — Режим доступа: для авториз. пользователей.

12. Разработка автоматизированной системы учета заказов ООО "ИнформТехСервис". URL: https://core.ac.uk/download/pdf/322817192.pdf (дата обращения 13.04.2024).

13. Семенова, И. И. SQL стандарт в современных СУБД: манипулирование данными : учебное пособие / И. И. Семенова, Е. О. Шершнева. — 2-е изд., дериватив., испр. и доп. — Омск : СибАДИ, 2023. — 54 с. — ISBN 978-5-00113-242-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/407393 (дата обращения: 21.04.2024). — Режим доступа: для авториз. пользователей.

14. Синицын, И. В. Разработка мобильных приложений : учебное пособие / И. В. Синицын, Е. А. Чернов, Ю. А. Воронцов. — Москва : РТУ МИРЭА, 2023 — Часть 1 — 2023. — 162 с. — ISBN 978-5-7339-1799-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/368735 (дата обращения: 20.04.2024). — Режим доступа: для авториз. пользователей.

15. Система управления заказами Generix OMS. URL: https://www.generixgroup.com/ru/решения/generix-oms-sistema-upravleniya-zakazami (дата обращения 13.04.2024).

16. Система управления заказами и задачами Фильта. URL: https://indins.ru/portfolio/filta (дата обращения 13.04.2024).

17. Система управления заказами на базе Comindware Business Application Platform. URL: https://www.comindware.ru/order-management-software/ (дата обращения 13.04.2024).

18. Соснин, П. И. Архитектурное моделирование автоматизированных систем : учебник для вузов / П. И. Соснин. — 2-е изд., стер. — Санкт-Петербург : Лань, 2024. — 180 с. — ISBN 978-5-507-49488-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/393065 (дата обращения: 20.04.2024). — Режим доступа: для авториз. пользователей.

19. Токмаков, Г. П. Базы данных: Модели и структуры данных, язык SQL, программирование баз данных : учебное пособие / Г. П. Токмаков. — Ульяновск : УлГТУ, 2021. — 362 с. — ISBN 978-5-9795-2184-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL:

https://e.lanbook.com/book/259706 (дата обращения: 21.04.2024). — Режим доступа: для авториз. пользователей.

20. Туманова, М. Б. Проектирование программных систем : учебное пособие / М. Б. Туманова, Е. К. Михайлова, Е. А. Муравьева. — Москва : РТУ МИРЭА, 2023. — 138 с. — ISBN 978-5-7339-2050-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/398273 (дата обращения: 20.04.2024). — Режим доступа: для авториз. пользователей.

21. Figma: The Collaborative Interface Design Tool. URL: https://www.figma.com/ (дата обращения 21.04.2024).

22. Postman API Network. URL: https://www.postman.com/api-evangelist/workspace/wikipedia/documentation/35240-c039dcb8-c468-46a5-bef2-944b0de198e7 (дата обращения 27.04.2024).

23. Software architecture design patterns to know. URL: https://www.redhat.com/architect/14-software-architecture-patterns (дата обращения: 14.04.2024).

24. The Practical Test Pyramid. URL: https://martinfowler.com/articles/practical-test-pyramid.html (дата обращения: 14.04.2024).

25. What is Lean methodology? URL: https://www.atlassian.com/agile/project-management/lean-methodology (дата обращения: 14.04.2024).

Приложение А

Основные формы интерфейса программного обеспечения

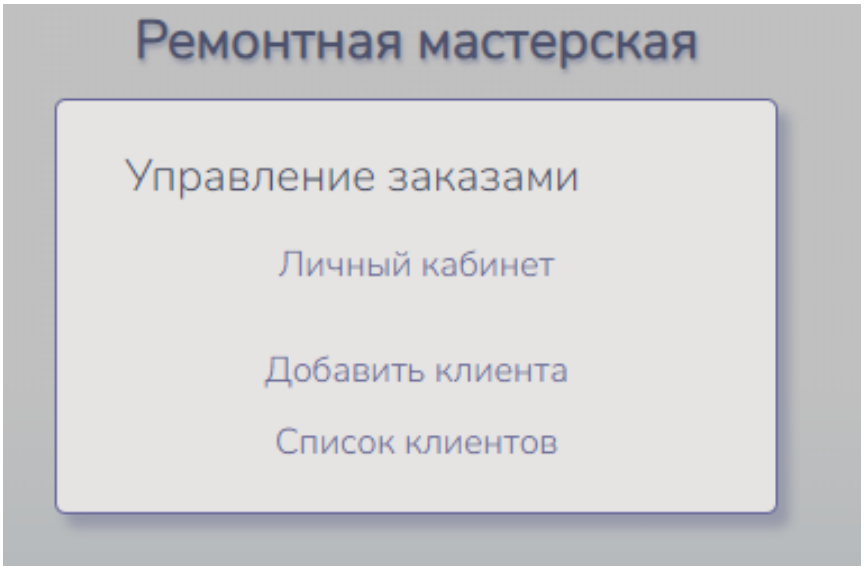

Рисунок А.1 – Интерфейс формы главного меню ИСУ

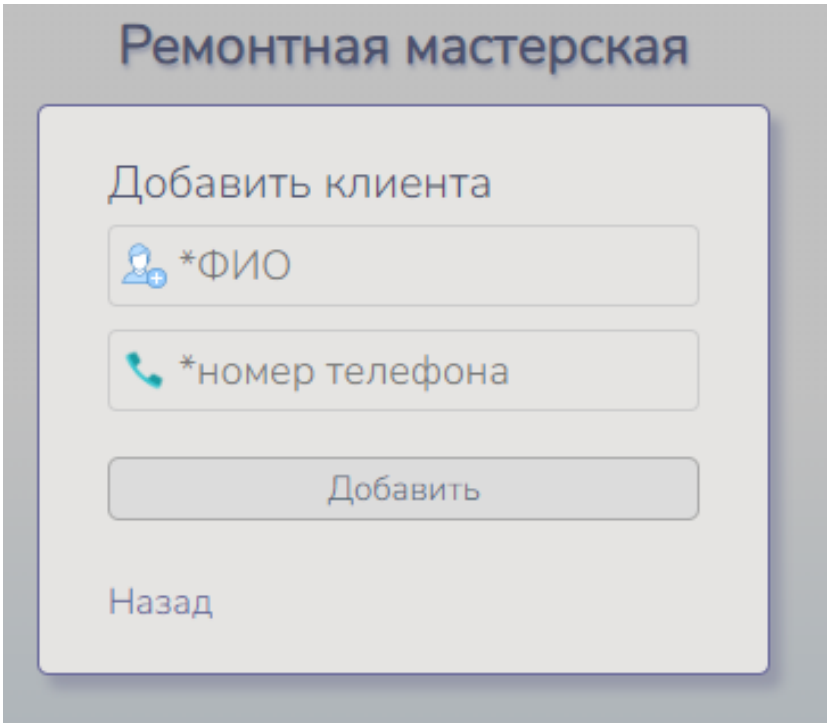

Рисунок А.2 – Интерфейс формы добавления нового клиента

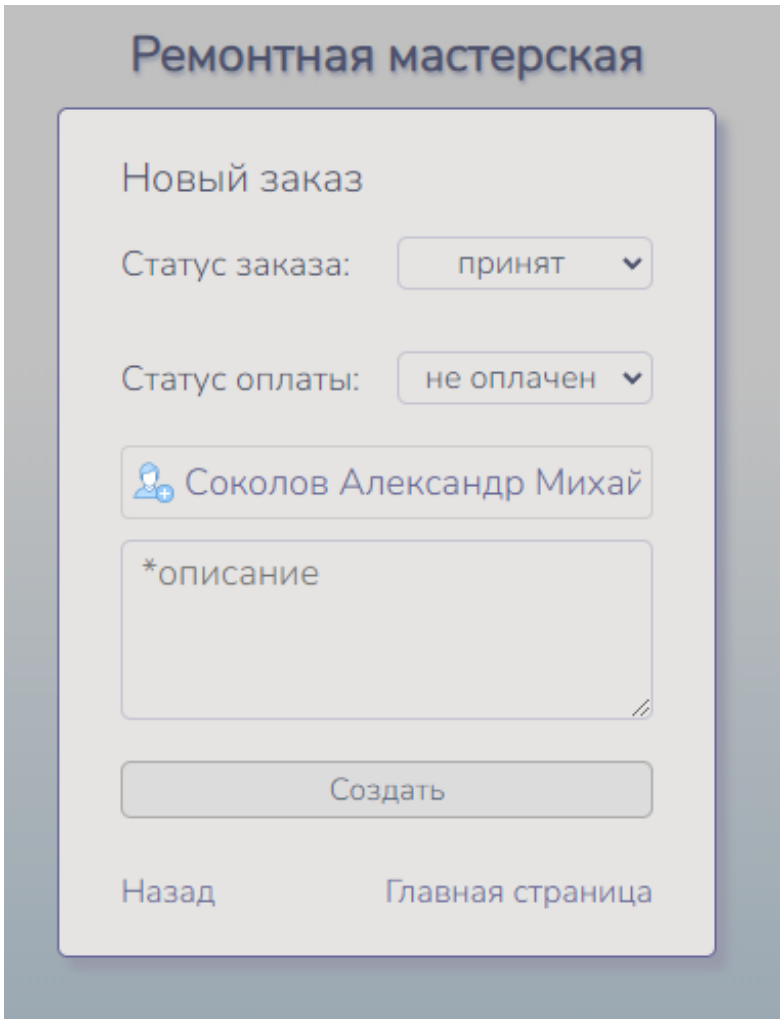

Рисунок А.3 – Интерфейс страницы создания нового заказа

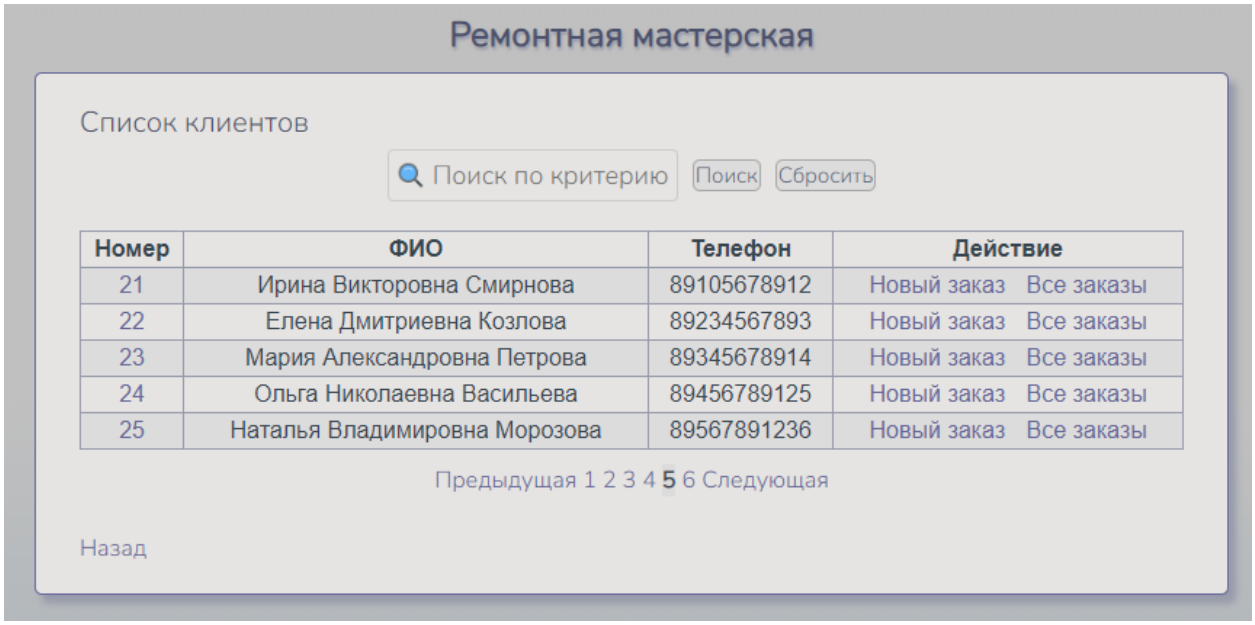

Рисунок А.4 – Интерфейс страницы списка всех клиентов