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 торгово-проектной организации

Обучающийся А.М. Пороженский
(Инициалы Фамилия) (личная подпись)

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

Тольятти 2025

Аннотация

Тема выпускной квалификационной работы: «Разработка программного обеспечения информационной системы управления заказами торгово-проектной организации».

Объектом работы является процесс управления заказами торгово-проектной организации.

Предметом работы является программное обеспечение информационной системы управления заказами торгово-проектной организации [28].

Цель работы – разработка программного обеспечения информационной системы управления заказами торгово-проектной организации.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы и используемых источников.

Во введении обоснована актуальность темы работы, представлены объект, предмет, цель и задачи работы.

В первой главе рассматривается объект и предмет работы, представлено обоснование разработки программного обеспечения, обзор аналогичных проектных решений, а также задачи, решаемые в процессе разработки.

Во второй главе представлено проектирование программного обеспечения.

В третьей главе представлена разработка программного обеспечения, выбор и описание программных средств, а также тестирование и отладка.

В заключении описываются результаты выполнения выпускной квалификационной работы [4].

Бакалаврская работа включает 63 страницы текста, 14 рисунков, 15 таблиц и 41 источник.

Abstract

The title of the graduation work is development of software for the order management information system of a trade and design organization.

The graduation work consists of an introduction, three parts, 14 figures, 15 tables, a conclusion, and a list of 41 references including foreign sources.

The aim of this graduation work is to development of software for the order management information system of a trade and design organization.

The object of the graduation work is the order management process of a trade and design organization.

The subject of the graduation work is the software for the order management information system of a trade and design organization.

The graduation work may be divided into several logically connected parts which are problem statement, review of analogues, formation of software requirements, design, development and testing of software.

The first part describes in details the object and subject of the research, the rationale for software development, an overview of similar design solutions, as well as the tasks being solved during the development process are presented.

The second part outlines the results of software design. A logical model was developed, as well as external and internal block diagrams of the software.

The third part consists of software development, selection and description of software tools, as well as testing and debugging

In conclusion we'd like to stress that the implementation of an order management information system for a trade and design organization has confirmed its importance as a tool for the digital transformation of business processes. The project has demonstrated not only its technical viability, but also its strategic value for improving the operational efficiency of the enterprise.

The work is of interest for wide circle of readers interested in automating the order management process in the field of trade.

Оглавление

Введение 5

Глава 1 Постановка задачи на разработку программного обеспечения информационной системы управления заказами 7

1.1 Разработка требований к программному обеспечению информационной системы управления заказами 7

1.2 Обзор и анализ аналогов программного обеспечения информационной системы управления заказами 9

Глава 2 Проектирование программного обеспечения информационной системы управления заказами 18

2.1 Логическое моделирование информационной системы управления заказами 18

2.2 Методологическое проектирование информационной системы управления заказами 23

2.3 Логическое проектирование информационной системы управления заказами 29

2.4 Логическое моделирование баз данных информационной системы управления заказами 31

Глава 3 Реализация и тестирование программного обеспечения информационной системы управления заказами 34

3.1 Выбор и описание программных средств разработки информационной системы управления заказами 34

3.2 Разработка физической модели базы данных информационной системы управления заказами 37

3.3 Моделирование интерфейса информационной системы управления заказами 42

3.4 Тестирование информационной системы управления заказами 46

Заключение 57

Список используемой литературы и используемых источников 58

Введение

В условиях современного бизнеса грамотное управление заказами является ключевым фактором для обеспечения высокого уровня обслуживания клиентов.

Информационная система для управления заказами играет решающую роль в этом процессе, так как способствует оптимизации приема и выполнения заказов, ускоряет обработку заявок и повышает качество сервиса. Это, в свою очередь, позитивно отражается на общей эффективности работы.

Актуальность данной темы связана с необходимостью повышения производительности и улучшения организации работы торгово-проектной организации за счет внедрения системы управления заказами. Разработка специализированного программного обеспечения позволит сделать процесс обслуживания клиентов более прозрачным, структурированным и качественным на каждом этапе взаимодействия.

Объектом работы является процесс управления заказами торгово-проектной организации.

Предметом работы является программное обеспечение информационной системы управления (далее - ПО ИСУ) заказами торгово-проектной организации [28].

Цель работы – разработка программного обеспечения информационной системы управления заказами торгово-проектной организации.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ имеющихся решений для постановки задачи на разработку;
- спроектировать программное обеспечение торгово-проектной организации;
- реализовать и выполнить тестирование программного обеспечения торгово-проектной организации.

Методы работы: методы и технологии проектирования программного обеспечения.

Практическая значимость работы заключается в том, что результаты работы могут быть использованы торгово-проектной организацией для внедрения программного обеспечения с целью повышения эффективности управления заказами.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы и используемых источников.

Во введении обоснована актуальность темы работы, представлены объект, предмет, цель и задачи работы.

В первой главе рассматривается объект и предмет работы, представлено обоснование разработки программного обеспечения, обзор аналогичных проектных решений, а также задачи, решаемые в процессе разработки.

Во второй главе представлено проектирование программного обеспечения. Были разработаны логическая модель, а также внешние и внутренние структурные схемы программного обеспечения.

В третьей главе представлена разработка программного обеспечения, выбор и описание программных средств, а также тестирование и отладка.

В заключении описываются результаты выполнения выпускной квалификационной работы [4].

Бакалаврская работа включает 63 страницы текста, 14 рисунков, 15 таблиц и 41 источник.

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

1.1 Разработка требований к программному обеспечению информационной системы управления заказами

Разработка программного обеспечения информационной системы управления заказами позволит организации решить несколько оптимизационных вопросов [10, 22]:

- повысить качество и скорость управления данными, информацией;
- автоматизировать процесс оформления заказов;
- снизить расходы на ручную обработку заказов;
- улучшить качество обслуживания клиентов и повысить их лояльность;
- повысить эффективность деятельности.

При разработке ПО ИСУ необходимо предусмотреть основные требования по системе FURPS+ в таблице 1 [15, 21].

Таблица 1 – Классификация требований к ПО ИСУ по системе FURPS+ [41]

Требование	Статус	Полезность	Риск
Функциональные требования			
Возможность добавления номера, статуса и описания заказа, поиска заказов и изменения заказа	Одобрено	Критическое	Средний
История взаимодействия с клиентом	Одобрено	Критическое	Средний
Разграничение прав доступа	Одобрено	Критическое	Низкий
Актуализация каталога в базе данных	Одобрено	Критическое	Средний
Экспорт данных в различные форматы	Одобрено	Критическое	Низкий
Службы отправки и получения сообщений	Одобрено	Критическое	Низкий
Средства для печати документов	Одобрено	Критическое	Низкий
Инструменты создания и получения отчетов	Одобрено	Критическое	Средний
Средства защиты доступа к определенным ресурсам информации	Одобрено	Критическое	Высокий

Продолжение таблицы 1

Требование	Статус	Полезность	Риск
Удобство использования			
Эстетика и логичность пользовательского интерфейса	Одобрено	Важное	Средний
Защита от человеческого фактора	Одобрено	Важное	Высокий
Эксплуатационная документация	Одобрено	Важное	Низкий
Справочная информация в системе	Одобрено	Важное	Низкий
Надежность			
«Не больше одного сбоя в месяц	Одобрено	Критическое	Высокий
Возможность восстановления системы после сбоев без потери данных	Одобрено	Критическое	Высокий
Система должна быть доступна 24 часа в сутки 7 дней в неделю	Одобрено	Критическое	Средний
Точность вычислений	Одобрено	Критическое	Средний
Производительность			
Время отклика на запрос пользователя не больше 20 секунд	Одобрено	Важное	Средний
Общее и допустимое количество одновременно работающих пользователей не более 50 человек	Одобрено	Важное	Средний
Скорость восстановления не более 30 секунд	Одобрено	Важное	Средний
Поддерживаемость			
Пригодность к проведению ремонтных работ	Одобрено	Важное	Средний
Работоспособность не должна зависеть от стороннего программного обеспечения и действий пользователей	Одобрено	Важное	Средний
Предусмотрен механизм обновления системы и её компонентов	Одобрено	Важное	Средний
Доступна документация по использованию и сопровождению системы	Одобрено	Важное	Средний
Ограничения			
Документация должна создаваться в MS Word	Одобрено	Важное	Средний
В качестве СУБД должна быть использована PostgreSQL	Одобрено	Важное	Средний» [10]

Требования, представленные в таблице 1, будут служить базой для проектирования.

1.2 Обзор и анализ аналогов программного обеспечения информационной системы управления заказами

«Современные системы управления заказами предлагают широкий спектр возможностей, включая встроенные функции для работы с биллингом, подписками и CRM» [7]. В этом детальном обзоре рассмотрим наиболее эффективные инструменты, которые помогают оптимизировать процесс обработки и отслеживания заказов.

Kibo OMS – это облачная платформа для управления заказами, предназначенная для автоматизации бизнес-процессов. Она упрощает контроль над запасами, управление заказами и их выполнение, обеспечивая удобные инструменты для оптимизации работы [29].

Одним из ключевых направлений развития Kibo OMS является совершенствование распределенного управления заказами (DOM), что позволяет ритейлерам настраивать сложные заказы и координировать работу поставщиков, снижая затраты и повышая скорость обработки.

Среди функциональных возможностей платформы – автоматические email-уведомления, печать этикеток, удобный drag-and-drop интерфейс, управление каталогом, анализ трендов и многое другое. Руководители могут настраивать параметры доставки для клиентов, а также использовать встроенный WYSIWYG-редактор для адаптации контента сайта и индивидуальных предпочтений пользователей. Дополнительно Kibo OMS поддерживает интеграцию с широким спектром сторонних сервисов, что делает ее универсальным инструментом для эффективного управления заказами. Пользовательский интерфейс представлен на рисунке 1.

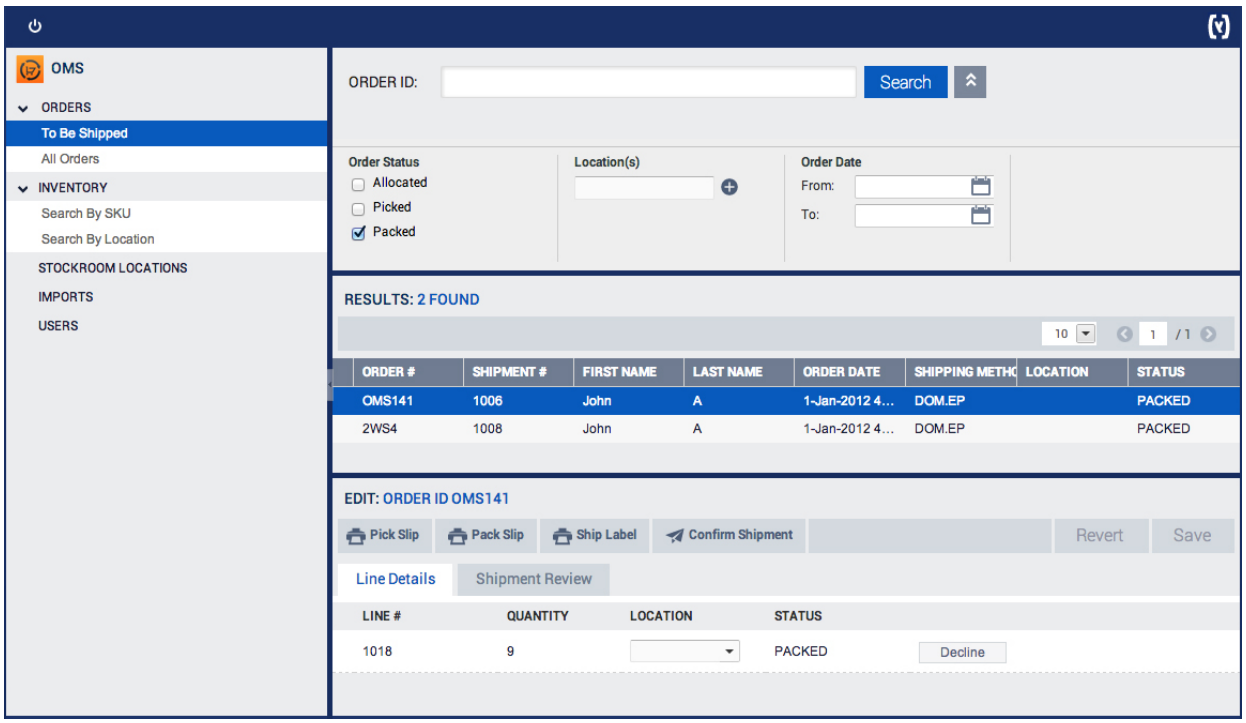

Рисунок 1 – Система управления заказами Kibo

Преимущества Kibo OMS:

- облачная архитектура – система работает в облаке, что обеспечивает доступность из любой точки и снижает затраты на локальную инфраструктуру;
- гибкость и масштабируемость – подходит для бизнеса разного уровня, легко адаптируется к меняющимся потребностям компаний;
- «распределенное управление заказами (DOM) – оптимизирует логистику, позволяя эффективно обрабатывать сложные заказы и управлять цепочками поставок;
- мощные инструменты аналитики – встроенные инструменты мониторинга и анализа трендов помогают принимать обоснованные решения;
- интеграция с CRM, ERP и сторонними приложениями – позволяет синхронизировать работу с различными бизнес-инструментами, упрощая управление процессами;

- продвинутые настройки доставки – пользователи могут выбирать различные способы доставки в зависимости от своих предпочтений» [29];
- интуитивный интерфейс – наличие drag-and-drop элементов и WYSIWYG-редактора упрощает работу с контентом и настройками системы.

«К недостаткам Kibo OMS можно отнести:

- стоимость – цена может быть высокой для малого бизнеса, особенно с учетом затрат на интеграцию и обслуживание;
- сложность настройки – для полной адаптации системы под бизнес-процессы может потребоваться техническая поддержка или участие специалистов» [29];
- зависимость от интернета – как и любая облачная система, Kibo OMS требует стабильного интернет-соединения, что может стать проблемой в регионах с нестабильной сетью;
- «ограниченная кастомизация без дополнительной разработки – в некоторых случаях возможности настройки могут быть недостаточными без доработок со стороны разработчиков;
- кривая обучения – хотя система удобна, новые пользователи могут столкнуться с необходимостью обучения для эффективного использования всех функций» [29].

Aptos OMS выделяется своей способностью интегрировать коммерческие процессы, охватывая весь цикл управления заказами – от их оформления до выполнения. Особенно полезна эта система для розничных компаний, поскольку она способствует увеличению количества обработанных заказов за счет расширенной их видимости, эффективной маршрутизации и организации выполнения в магазинах, что также улучшает качество обслуживания клиентов.

Aptos предоставляет облачную платформу [30], которая позволяет централизованно управлять складскими запасами, помогая бизнесу принимать

стратегически важные решения с учетом потребностей клиентов. Это способствует росту рентабельности и оптимизации логистики. Среди ключевых возможностей системы – мобильное приложение для управления заказами, которое помогает синхронизировать спрос и доступные товары, а также обрабатывать онлайн-покупки, заказы между магазинами и доставку товаров в торговые точки. Пользовательский интерфейс представлен на рисунке 2.

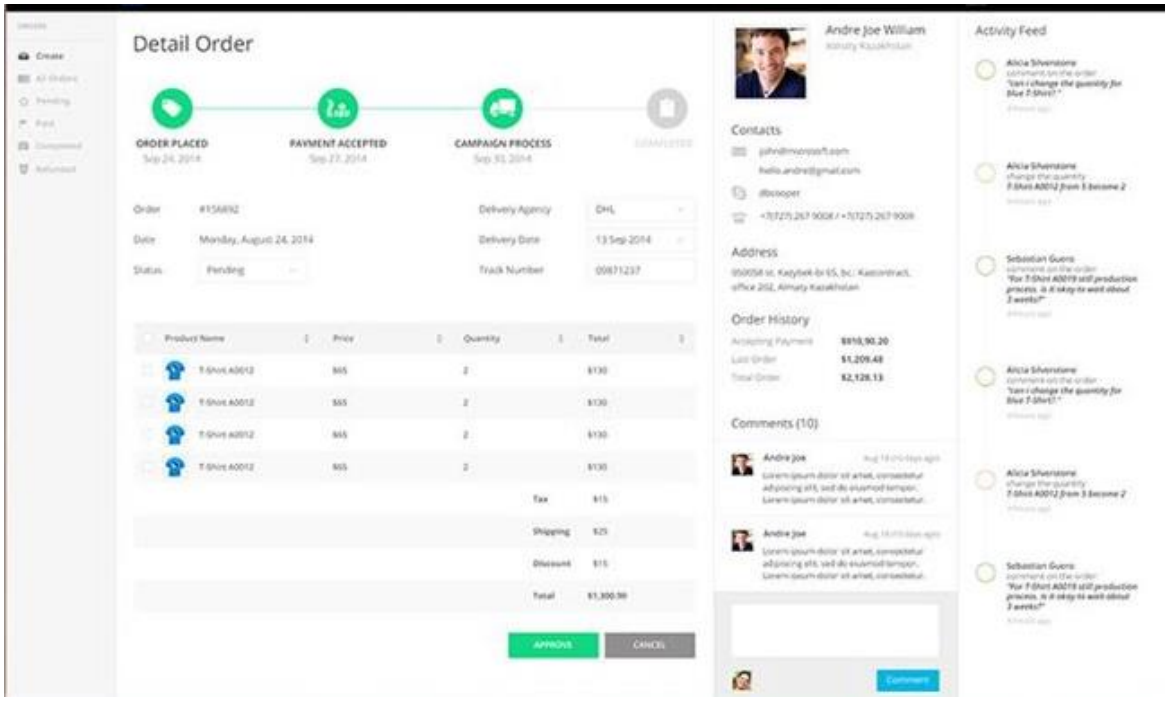

Рисунок 2 – Система управления заказами Aptos

Преимущества Aptos OMS:

- гибкость в управлении заказами – система объединяет онлайн- и офлайн-каналы, позволяя ритейлерам эффективно управлять заказами из различных источников;
- облачная инфраструктура – обеспечивает централизованный доступ к данным, снижает затраты на локальное оборудование и упрощает масштабирование;
- продвинутая маршрутизация заказов – оптимизирует логистику,

помогая быстро направлять заказы в нужные точки выполнения (магазины, склады и т. д.);

- интеграция с магазинами – мобильное приложение Aptos позволяет сотрудникам обрабатывать онлайн-заказы, оформлять покупки в магазине и управлять межмагазинными заказами;
- ориентация на клиента – улучшает взаимодействие с покупателями за счет высокой прозрачности заказов, оптимального распределения запасов и удобных вариантов доставки;
- рост рентабельности – использование складских запасов в масштабах всей компании помогает минимизировать издержки и повышать доходность бизнеса.

«К недостаткам Aptos OMS можно отнести:

- высокая стоимость – покупка и внедрение системы могут оказаться затратными, особенно для малого и среднего бизнеса;
- сложность настройки – несмотря на удобный интерфейс, полная адаптация системы под бизнес-процессы требует времени и технической экспертизы;
- зависимость от стабильного интернет-соединения – облачная архитектура требует постоянного доступа к сети, что может стать проблемой в некоторых регионах;
- ограниченная кастомизация без дополнительных интеграций – для реализации специфических функций может потребоваться доработка с привлечением разработчиков;
- кривая обучения для новых пользователей – освоение всех возможностей системы может занять время, особенно для сотрудников» [30], не имеющих опыта работы с подобными решениями.

Brightpearl, розничная операционная система, создана специально для розничной и оптовой торговли и предлагает специализированные функции, которые адаптируются к требованиям бизнеса в сфере электронной

коммерции. В отличие от традиционных ERP (планирование ресурсов предприятия), автономных OMS, WMS (система управления складом) или систем управления запасами, Brightpearl использует уникальный подход к расширению возможностей бизнеса [31].

Превосходная автоматизация, подробная отчетность и расширенная аналитика Brightpearl обеспечивают эффективную обработку заказов, управление запасами и повышение эффективности деятельности, сокращая при этом человеческие ошибки и затраты на рабочую силу. Пользовательский интерфейс представлен на рисунке 3.

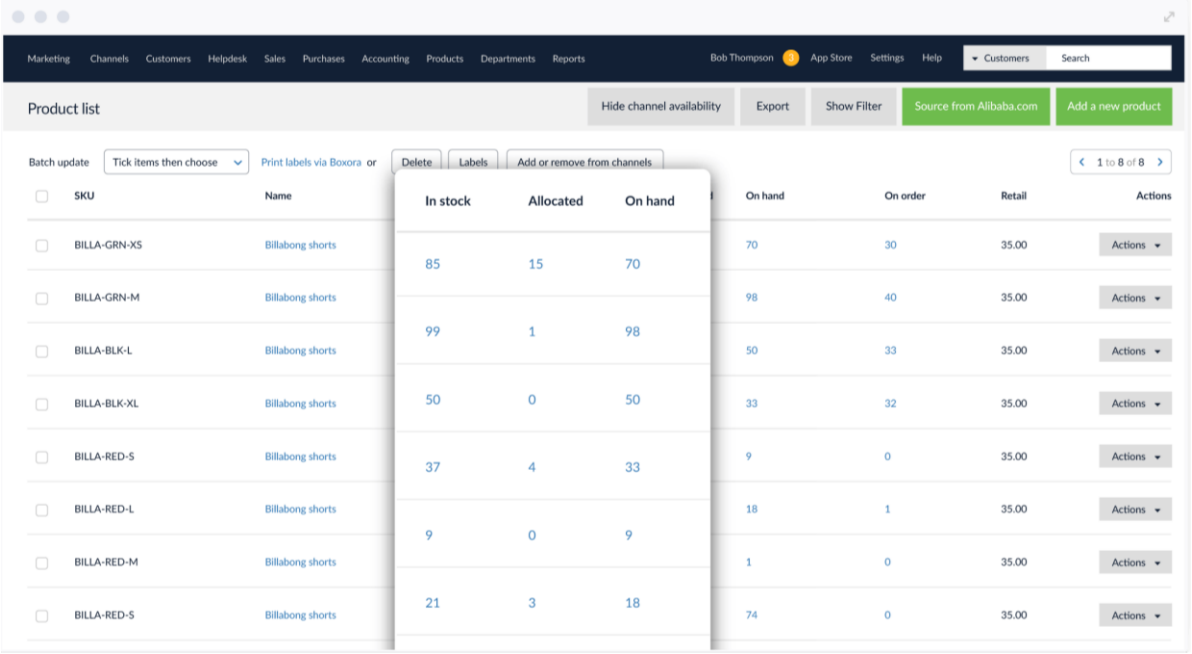

Рисунок 3 – Система управления заказами Brightpearl

Преимущества Brightpearl:

- быстрое внедрение – Brightpearl обеспечивает высокую скорость развертывания с показателем успешности 97%, благодаря продуманному процессу внедрения под руководством опытных специалистов;
- масштабируемость без ограничений – платформа легко адаптируется к росту бизнеса, предлагая расширенные интеграции и развитую

партнерскую экосистему, что позволяет беспрепятственно расширять возможности компании;

- автоматизация для эффективного роста – использование Brightpearl помогает снизить затраты на персонал до 50%, сократить количество ошибок на 65% и экономить в среднем два месяца рабочего времени в год;
- круглосуточная поддержка – клиенты получают оперативную помощь 24/7, а также доступ к бизнес-консультированию без дополнительных затрат.

К недостаткам Brightpearl можно отнести:

- сложность настройки – система требует тщательной конфигурации, что может потребовать привлечения специалистов в области ИТ;
- ограниченные возможности интеграции – по количеству готовых интеграций Brightpearl уступает некоторым конкурентам;
- неинтуитивный интерфейс – новые пользователи могут столкнуться с трудностями при навигации из-за сложной структуры интерфейса;
- проблемы с отчетностью – система может выдавать некорректные данные о доходах при продаже товаров в наборах.

«Соотнесем выявленные выше особенности по имеющимся аналогам проектных решений для сравнения и получения выводов в таблице 2, используя следующие критерии оценивания:

– несоответствие требованиям – 0;
– значительное несоответствие требованиям – 1;
– незначительное несоответствие требованиям – 2;
– соответствие требованиям – 3» [26].

Учитывая максимум 69 баллов по сравнительным характеристикам, можно сделать вывод, что представленные проектные решения не удовлетворяют запрос торгово-проектной организации, так как имеют низкую оценку легкости использования, скорости внедрения, параметров отчетности и аналитики, а также масштабности параметров настройки.

Таблица 2 – Сравнительный анализ аналогов проектных решений ПО ИСУ заказами

Факторы	Kibo	Aptos	Brightpearl
Бесплатная пробная версия/версия	0	0	0
Легко использовать	3	3	0
Отличная производительность	3	1	3
Обработка заказов	3	3	3
Система управления запасами	3	3	3
Многоканальная поддержка заказов	3	3	3
Возможности интеграции	3	1	1
Возможности многоканального взаимодействия	3	2	2
Управление возвратами и обменами	3	3	3
Функции управления складом	3	3	3
Управление транспортом	2	2	2
Управление цепочками поставок	3	2	2
Решение для колл-центра	2	3	2
Управление поставками	3	3	3
Параметры настройки	3	1	3
Предотвращение мошенничества	3	3	3
Управление взаимоотношениями с клиентами (CRM)	3	3	3
Отчетность и аналитика	1	3	1
Возможности настройки	3	3	2
Безопасность и конфиденциальность данных	3	3	3
Масштабируемость	2	3	2
Быстрое внедрение	3	1	0
Отличная поддержка клиентов	3	3	3
Итого	61	55	50

Следовательно, существует необходимость в разработке программного обеспечения информационной системы управления заказами со следующими параметрами:

- «интуитивная простота интерфейса и логики;
- отсутствие ограничений по наполнению базы;
- отсутствие ограничений масштабируемости параметров настройки под конкретные потребности;
- быстрота внедрения;
- легкоуправляемая и доступная отчетность и аналитические данные» [17].

Выводы по главе 1

«Для формирования требований на разработку ПО ИСУ заказами торгово-проектной организации была проведена следующая работа:

- проведена функциональная характеристика специфики работы торгово-проектной организации для определения особенностей разрабатываемого ПО ИСУ;
- определены базовые функции с учётом особенностей информационных систем управления заказами;
- разработаны требования по системе FURPS+ с учётом специфики работы торгово-проектной организации.

Проведен обзор и анализ существующих аналогов, который позволил выявить основные характеристики, преимущества и недостатки существующих решений, а также обосновать разработку собственного ПО ИСУ заказами торгово-проектной организации» [17].

Глава 2 Проектирование программного обеспечения информационной системы управления заказами

2.1 Логическое моделирование информационной системы управления заказами

Исходя из требований к ПО ИСУ заказами, сформированных в первой главе работы, можно определить следующий функционал:

Аутентификация:

- вход по логину и паролю;
- ролевая модель: пользователь (работа с заказами).

Работа с клиентами:

- регистрация нового клиента (ФИО, телефон);
- поиск клиента по номеру телефона.

Управление заказами:

- создание заказа (автоматическое присвоение ID и даты);
- изменение статуса заказа («выполнен, не оплачен», «оплачен», «отгружен»);
- отметка об оплате;
- отметка об отгрузке;
- генерация платежных документов в PDF;
- генерация отгрузочных документов в PDF.

Отчетность:

- фильтрация заказов по статусу и дате;
- экспорт данных в Excel.

Логическое моделирование информационной системы управления заказами позволяет не только структурировать данные и процессы, но и выявить возможные узкие места ещё на этапе проектирования [16].

Данный подход основан на представлении информации в виде взаимосвязанных сущностей, правил и процессов, что делает систему

прозрачной, предсказуемой и легко масштабируемой. С помощью логической модели можно чётко определить, как заказы поступают, обрабатываются и передаются на выполнение, обеспечивая минимизацию рисков и оптимизацию ресурсов [9].

Рассмотрим ключевые принципы логического моделирования, его роль в разработке систем управления заказами и способы повышения эффективности бизнес-процессов с помощью грамотно спроектированной информационной структуры [5].

С помощью языка объектного моделирования UML построена концептуальная модель в форме диаграммы вариантов использования, представлена на рисунке 4 [20, 34].

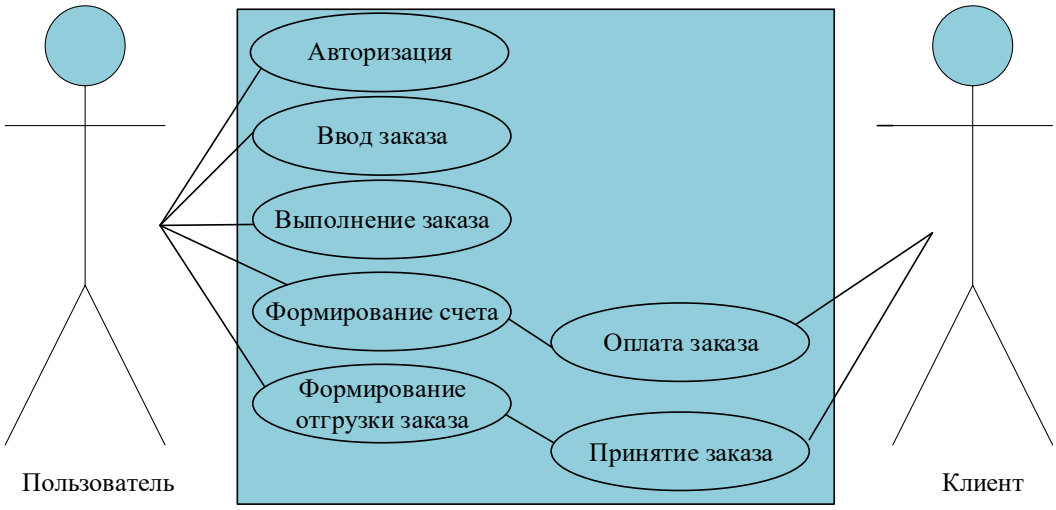

Рисунок 4 – Диаграмма вариантов использования ИСУ

Описание вариантов использования ИСУ в последовательности от исходных к итоговым:

– авторизация пользователя в ИСУ;
– формирование заказа пользователем ИСУ;
– выполнение задания пользователем в виде проставления отметок о выполнении (статус заказа «Выполнен», «Не оплачен»);

– формирование итоговых платежных документов для выполненного заказа;
– формирование отгрузочных документов для выполненного заказа;
– после оплаты заказа клиентом проставление пользователем отметки в ИСУ статуса заказа «Оплачен»;
– после принятия заказа клиентом проставление пользователем отметки в ИСУ статуса заказа «Отгружен».

На рисунке 5 представлена диаграмма деятельности для демонстрации процесса управления заказами [23, 24]. Процесс управления заказами включает следующий сценарий:

- пользователь запускает ИСУ и авторизуется в ней;
- при неуспешной авторизации ИСУ запускает повторную авторизацию;
- при успешной авторизации пользователь осуществляет поиск клиента по номеру мобильного телефона;
- при отсутствии клиента в системе пользователь добавляет клиента в ИСУ;
- «пользователь вводит данные заказа, осуществляет поиск заказа для смены его статусов и просмотра;
- после выполнения заказа пользователь присваивает заказу статус Выполнен, не оплачен, формирует платежные документы и после оплаты заказа клиентом меняет статус заказа на Выполнен, оплачен;
- пользователь формирует отгрузочные документы для выполненного и оплаченного заказа, после принятия заказа клиентом пользователь меняет статус заказа на Отгружен» [12].

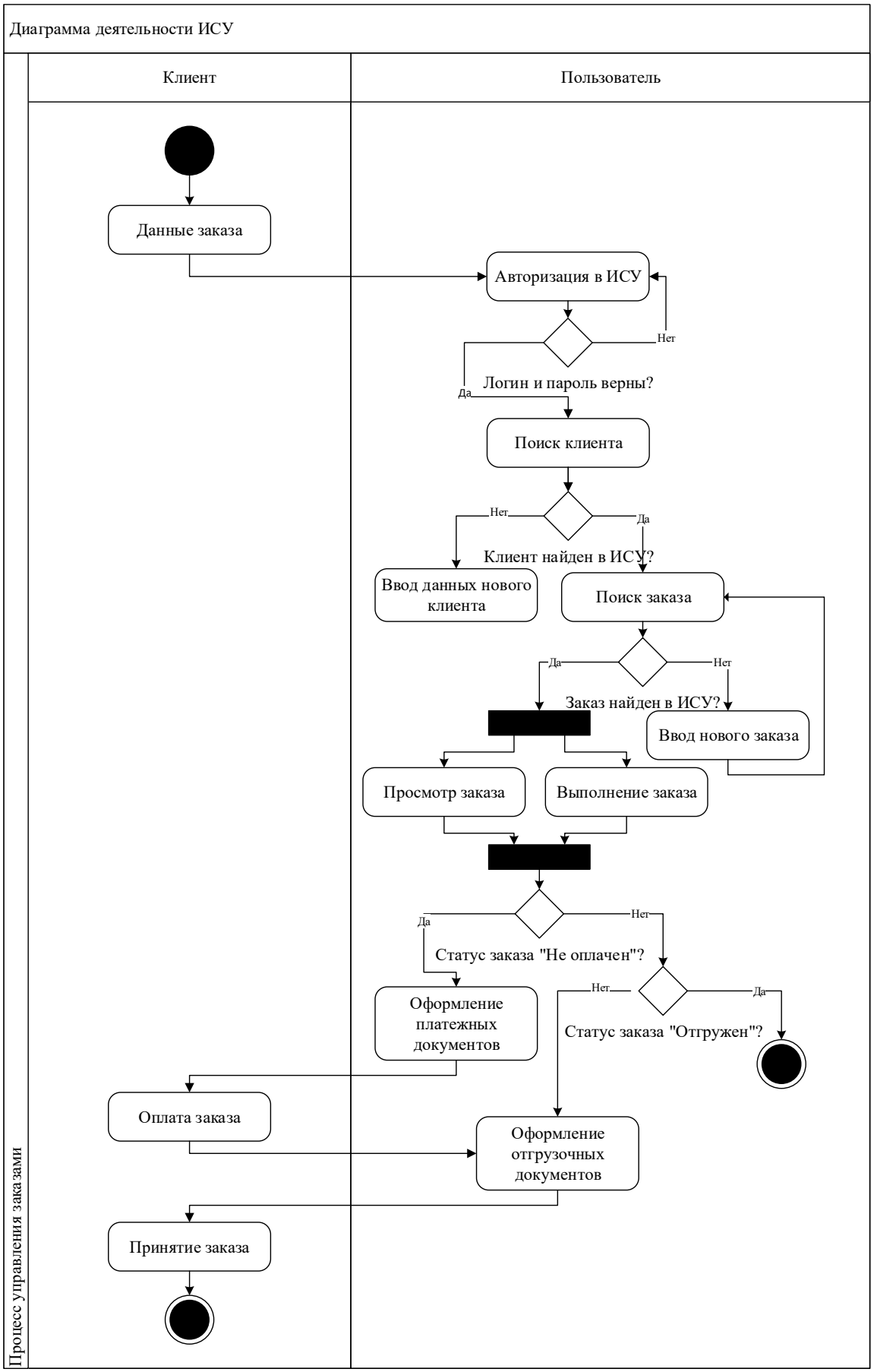

Рисунок 5 – Диаграмма деятельности в процессе управления заказами

Взаимосвязи между компонентами системы отражены в диаграмме классов на рисунке 6.

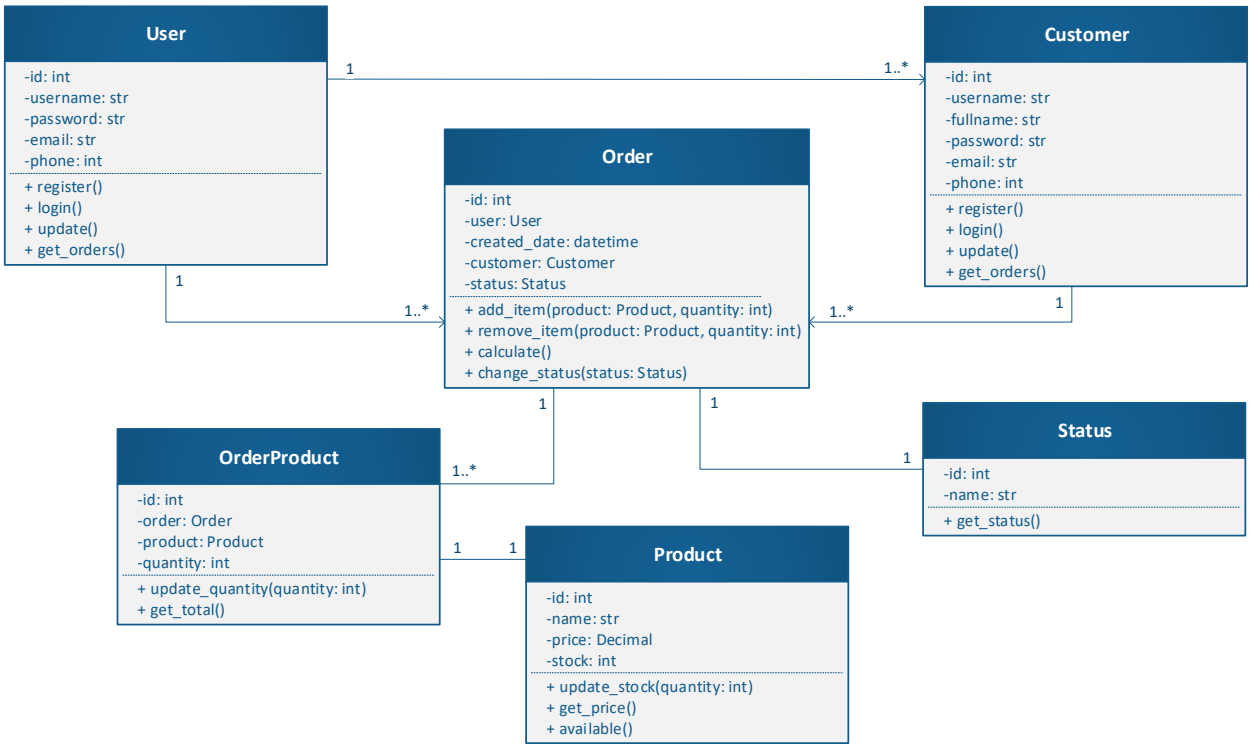

Рисунок 6 – Диаграмма классов ИСУ

В данном методе проектирования каждый класс изображается в форме прямоугольника, содержащего три внутренних сегмента: имя класса, атрибуты и операции. Связь между классами устанавливается стрелками, указывающими на направление передачи информации.

На диаграмме описаны следующие классы:

- «User – содержит идентификационные данные пользователя (уникальный идентификатор пользователя, логин, почту и пароль);
- Order – содержит идентификационные данные заказа (уникальный идентификатор заказа, уникальный идентификатор клиента, статус заказа, дату и содержание);

- OrderProduct – содержит идентификационные данные продуктов в заказе (уникальный идентификатор, уникальный идентификатор заказа, продукт и его количество);
- Product – содержит идентификационные данные продукта (уникальный идентификатор продукта, наименование, стоимость, количество)» [19];
- Status – содержит идентификационные данные статуса (уникальный идентификатор статуса и его наименование);
- Customer – содержит идентификационные данные клиента (уникальный идентификатор клиента, ФИО, логин, пароль, почту и телефон).

Каждый из уникальных пользователей может создать множество заказов, также, как и у каждого уникального клиента может быть множество заказов. Также в заказе может быть несколько позиций продукта. Следовательно классы User – Order, User – Customer, Customer – Order, OrderProduct – Product связаны N-арной ассоциацией (один-ко-многим) [19]. Классы Order – Status и Order – OrderProduct имеют связь один к одному.

2.2 Методологическое проектирование информационной системы управления заказами

Для эффективного создания программного обеспечения необходимо выбрать методологию разработки, набор принципов, процессов и практик. Основные методологии можно разделить на две крупные группы: гибкие (Agile) и традиционные (Waterfall и другие каскадные модели) [1].

Традиционные методологии характеризуются строгой последовательностью этапов и формальным управлением проектом. К основным традиционным методологиям относятся следующие.

«Каскадная модель (Waterfall) строится на принципах:

- разработка идет по четким этапам, и переход на следующий возможен только после завершения предыдущего;

- разделение на этапы: анализ требований, проектирование, реализация, тестирование, внедрение, сопровождение;
- подходит для проектов с четко определенными требованиями, но сложна при внесении изменений.

V-образная модель (V-Model) это улучшенная версия Waterfall, где каждому этапу разработки соответствует этап тестирования, что позволяет раньше выявлять ошибки, но сохраняет низкую гибкость» [17].

Спиральная модель (Spiral Model) сочетает каскадный подход с итерациями и анализом рисков. Включает принцип прохождения проектом через последовательные циклы: планирование, анализ рисков, проектирование, реализация, тестирование. Подходит для сложных проектов с высокими рисками, но требует больших затрат.

«Гибкие методологии (Agile) ориентированы на адаптивное управление, взаимодействие с заказчиком и быстрое реагирование на изменения» [39].

Общий подход Agile основан на манифесте, который включает принципы гибкости, взаимодействия и адаптивного планирования [27]. ПО разрабатывается небольшими итерациями (спринтами). Акцент ставится на командную работу, обратную связь и регулярные улучшения.

К основным гибким методологиям можно отнести следующие [14].

Методология Scrum как одна из самых популярных Agile-методологий [6]. Основные ее принципы:

- работа делится на спринты (обычно 1–4 недели);
- включает ключевые роли: Scrum Master, Product Owner, команда разработки;
- используются артефакты: бэклог продукта, бэклог спринта, инкремент.

Следующая гибкая методология Kanban, основанная на принципе ограничения количества одновременно выполняемых задач (WIP – Work In Progress) [8]. Использует визуальный метод управления задачами с

применением доски (например, Trello, Jira) и подходит для поддержки и непрерывного улучшения процессов.

Гибкая методология Lean (Бережливая разработка) основана на принципах минимизации потерь, ускорения поставки ценности пользователям [40]. Ее важными элементами являются минимально жизнеспособный продукт (MVP) и непрерывное улучшение.

Методология Extreme Programming (XP) подходит для команд с частыми изменениями требований и включает практики парного программирования, непрерывной интеграции, автоматизированного тестирования, рефакторинга.

Также можно выделить гибридные методологии, которые совмещают элементы традиционных и гибких подходов:

- RUP (Rational Unified Process) – итерационный процесс, сочетающий строгие фазы (инициация, разработка, конструирование, эксплуатация) с гибкостью в их исполнении;
- Scrumban – гибрид Scrum и Kanban, который использует спринты из Scrum, но задачи управляются с помощью Kanban-доски;
- DevOps – культурная и технологическая методология, объединяющая разработку и эксплуатацию ПО, включающая автоматизацию, CI/CD (непрерывную интеграцию и развертывание), мониторинг.

Сравнение описанных выше методологий приведено в таблице 3.

Таблица 3 – Сравнение методологий разработки ПО

Методология	Тип	Гибкость	Скорость разработки	Наиболее подходящие проекты	Основные особенности	Примеры проектов
Waterfall (Каскадная модель)	Традиционная	Низкая	Медленная	Проекты с четкими требованиями	Последовательные этапы, сложно вносить изменения	Банковские системы, госуслуги, системы учета

Продолжение таблицы 3

Методология	Тип	Гибкость	Скорость разработки	Наиболее подходящие проекты	Основные особенности	Примеры проектов
V-Model	Традиционная	Низкая	Медленная	Критически важные проекты	Тестирование на каждом этапе, высокая надежность	Медицинское ПО, авиационные системы
Spiral Model	Гибридная	Средняя	Медленная	Сложные проекты с высокими рисками	Итеративный процесс с анализом рисков	ПО для крупных предприятий, разработка ERP-систем
Agile	Гибкая	Высокая	Быстрая	Динамичные проекты, стартапы	Инкрементальная разработка, взаимодействие с заказчиком	Мобильные приложения, SaaS-сервисы, веб-платформы
Scrum	Agile	Высокая	Быстрая	Командные проекты с изменяющимися требованиями	Специфические роли, короткие спринты	Разработка CRM-систем, e-commerce платформ
Kanban	Agile	Высокая	Быстрая	Постоянные задачи, поддержка, DevOps	Управление задачами через визуальную доску	Поддержка веб-приложений, служба поддержки, DevOps
XP (Extreme Programming)	Agile	Высокая	Очень быстрая	Разработка с быстрым изменением требований	Парное программирование, тестирование, рефакторинг	Стартапы, быстрые MVP, fintech-проекты

Продолжение таблицы 3

Методология	Тип	Гибкость	Скорость разработки	Наиболее подходящие проекты	Основные особенности	Примеры проектов
Lean	Agile	Высокая	Быстрая	Оптимизация процессов, стартапы	Минимизация потерь, быстрые поставки	Оптимизация бизнес-процессов, производство, e-commerce
RUP (Rational Unified Process)	Гибридная	Средняя	Средняя	Средние и крупные проекты	Комбинация строгих этапов и гибкости	Корпоративные системы, финансовое ПО
Scrumban	Гибридная	Высокая	Быстрая	Гибкие проекты, улучшение процессов	Совмещает Scrum и Kanban	Разработка сложных продуктов с постепенными изменениями
DevOps	Гибридная	Высокая	Быстрая	Непрерывная поставка и поддержка ПО	Автоматизация, CI/CD, совместная работа Dev и Ops	Облачные сервисы, веб-платформы, CI/CD-пайплайны

Исходя из основных классификационных требований, в частности для оптимизации бизнес-процессов, наиболее подходящей методологией для цели данной работы будет гибкая методология Lean [18].

Таким образом разработка ПО ИСУ будет основываться на принципах методологии Lean:

- исключение потерь (Eliminate Waste) – все, что не приносит ценности конечному пользователю, считается потерями и должно быть устранено (избыточная документация, долгое ожидание обратной

связи, многократное исправление кода из-за слабого тестирования, разработка ненужных функций);

- ускорение поставки (Build Fast & Learn) – разработка ведется небольшими итерациями, что позволяет быстрее получать обратную связь и вносить улучшения;
- «усиление обучения (Amplify Learning) – в процессе работы команда постоянно анализирует результаты и улучшает процесс путем использования различных инструментов (A/B тестирование, анализ метрик и пользовательских отзывов, непрерывное тестирование и автоматизация)» [18];
- откладывание решений (Defer Commitment) – важные технические решения принимаются как можно позже, чтобы учитывать актуальные данные и избегать ненужных затрат;
- быстрая доставка (Deliver as Fast as Possible) – чем быстрее продукт попадает к пользователю, тем быстрее он приносит ценность;
- «вовлечение команды (Empower the Team) – Lean поощряет самостоятельные и ответственные команды, минимизируя бюрократию;
- встроенное качество (Build Integrity In) – качество продукта закладывается на всех этапах разработки с использованием различных практик (автоматизированное тестирование, Code Review, CI/CD непрерывная интеграция и развертывание), а не проверяется только перед релизом» [18].

Бережливая разработка Lean соответствовует требованиям к разработке ПО ИСУ заказами, т. к. она исключает лишние детали и оптимизирует процессы.

2.3 Логическое проектирование информационной системы управления заказами

«Согласно функциональным требованиям разрабатываемое ПО ИСУ будет включать пользовательский интерфейс, базу данных и бизнес-логику, управляющую процессами обработки заказов. Все компоненты системы могут быть объединены в единую структуру, где элементы тесно взаимодействуют между собой, что характерно для монолитной архитектуры» [33].

«Монолитная архитектура подразумевает упорядоченную многослойную организацию, где каждый слой выполняет свою задачу. Например, один отвечает за взаимодействие с базой данных, другой – за обработку логики и ведение журнала событий, а третий – за отображение информации пользователю. Такая структура облегчает тестирование всей системы (end-to-end) и делает процесс развертывания более удобным» [13].

Существует несколько популярных паттернов проектирования, применяемых в монолитной архитектуре:

- «MVC (Model-View-Controller) – разделяет логику приложения на три компонента: модель (обработка данных), представление (графический интерфейс) и контроллер (связующее звено между ними)» [2];
- MVP (Model-View-Presenter) – улучшает раздельность слоев, передавая управление пользовательским интерфейсом через презентер;
- MVVM (Model-View-ViewModel) – делает представление независимым от логики за счет использования промежуточного слоя (ViewModel).

Паттерн MVC широко применяется в разработке интерфейсов, поскольку позволяет эффективно разделять ответственность между обработкой данных, отображением информации и управлением взаимодействием пользователя с системой. Использование архитектурного паттерна MVC помогает структурировать ПО, разделяя его логику на

отдельные компоненты. Такой подход делает систему более гибкой, облегчая поддержку, модернизацию и возможное расширение функционала.

При проектировании архитектуры информационной системы важно определить ключевые модули, их связи и способы взаимодействия [25]. После этого для каждого модуля следует установить набор данных и функций, которые он будет обрабатывать. На следующем этапе разрабатывается диаграмма компонентов в соответствии с выбранным архитектурным шаблоном MVC, которая представлена на рисунке 7.

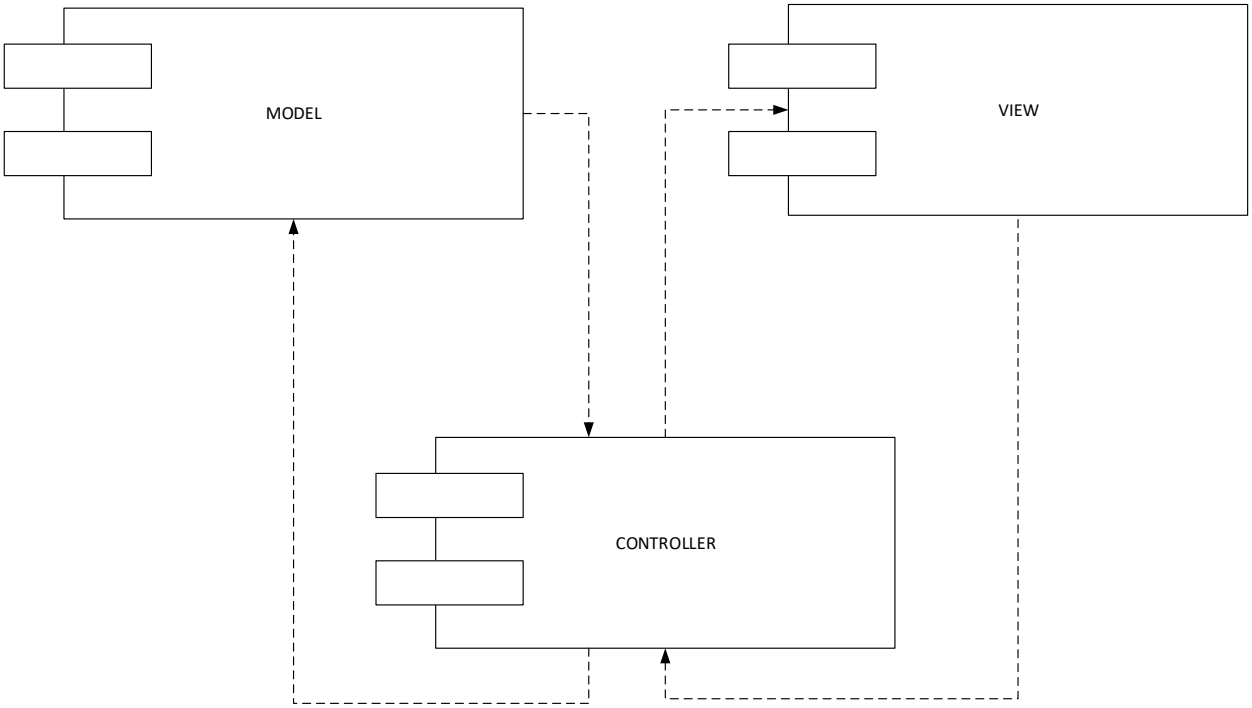

Рисунок 7 – Диаграмма компонентов

В диаграмме компонентов отображается разделение системы на логические модули и их взаимосвязь [37].

Основные модули:

- View (Представление) – отвечает за пользовательский интерфейс, содержит стили (CSS: order_styles.css: стили для таблицы заказов), шаблоны (HBS/HTML: order_list.hbs: страница со списком заказов,

create_order.hbs: форма создания заказа), JavaScript (DOM: order_script.js: динамическое обновление статуса заказа;

- «Controller (Контроллер) – обрабатывает запросы и управляет логикой, содержит функции – валидация (validation.js, проверка корректности данных) и интерактивность (обработка событий);
- Model (Модель) – работает с данными и бизнес-логикой, содержит следующие функции: Create (создание данных), Read (чтение данных)» [15], Update (редактирование данных), Delete (удаление данных), Search (поиск данных).

«Взаимодействие между компонентами:

- Пользователь взаимодействует с View (например, заполняет форму заказа);
- View передает данные в Controller;
- Controller обращается к Model, чтобы сохранить данные в БД и получить данные для отображения;
- Model возвращает результат (например, список заказов)» [15];
- Controller передает данные в View;
- View обновляет интерфейс (например, показывает созданный заказ).

«Архитектура ПО является основой для успешного проекта, поскольку определяет общую структуру и организацию системы, влияя на её гибкость, расширяемость и эффективность» [15].

2.4 Логическое моделирование баз данных информационной системы управления заказами

В информационной системе управления (ИСУ) основными пользователями будут менеджеры, которые займутся обработкой заказов в организации. Им потребуется функционал для создания новых заказов, внесения изменений в уже оформленные заявки, а также просмотра списка всех заказов. В каждой записи должна храниться ключевая информация,

включая ФИО клиента, его контактный номер, статус заказа, дату оформления и подробное описание.

«Исходя из этого, оптимальным решением станет использование реляционной базы данных, в которой данные будут структурированы в шести таблицах, обеспечивая удобное хранение и управление заказами» [32].

Логическая модель базы данных представлена на рисунке 8.

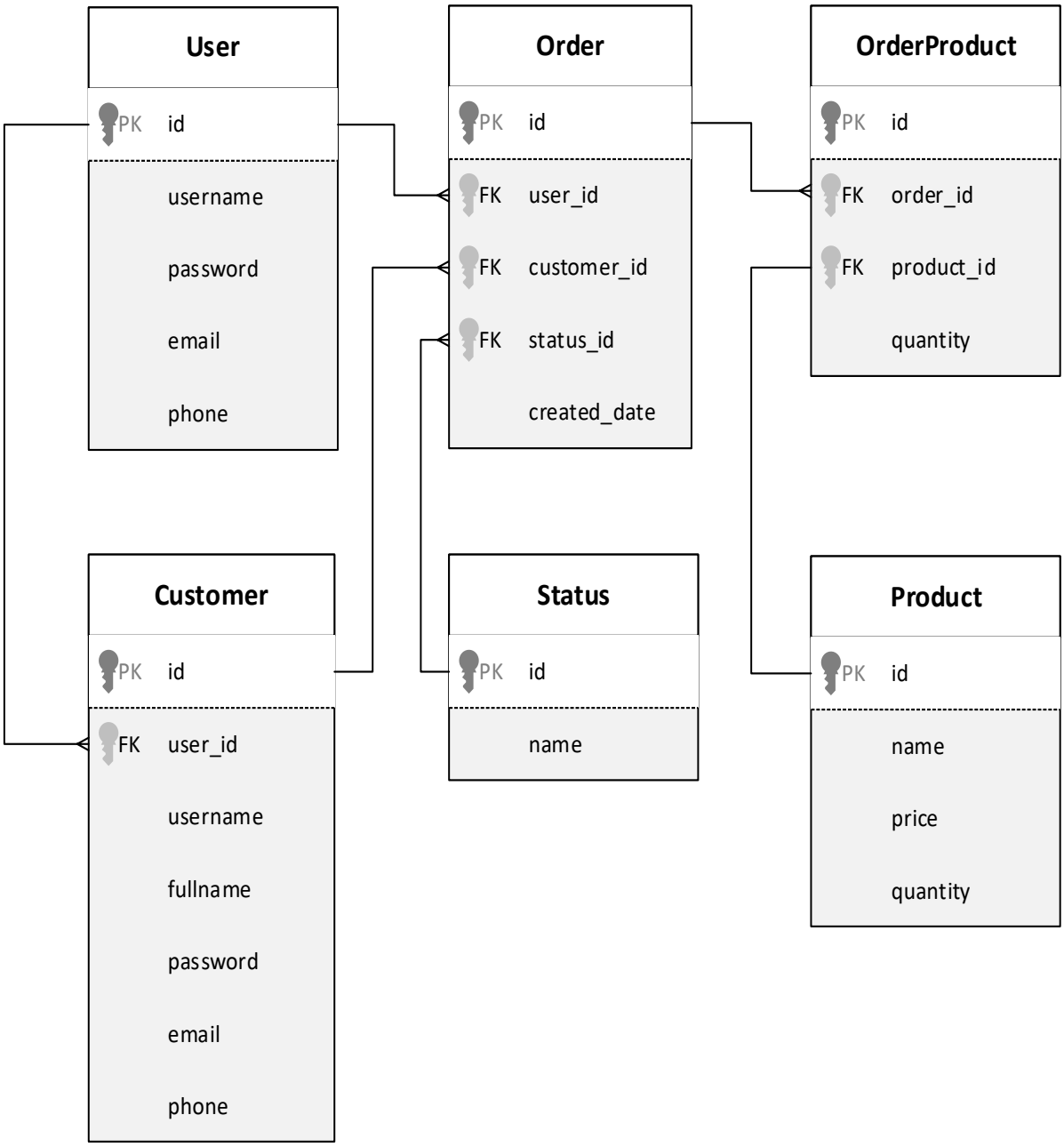

Рисунок 8 – Логическая модель базы данных

Каждый из уникальных пользователей может создать множество заказов, также, как и у каждого уникального клиента может быть множество заказов.

Также в заказе может быть несколько позиций продукта.

Выводы по главе 2

Использование методологии проектирования способствует упрощению процесса разработки программного продукта за счет четкого и тщательного описания этого процесса и применения наиболее современных методов и технологий.

Моделирование на языке UML позволяет рассмотреть систему со всех сторон и учесть все требования и параметры в разработке и последующем ее эксплуатировании [34].

Исходя из этого, оптимальным решением станет использование реляционной базы данных, в которой данные будут структурированы в шести таблицах, обеспечивая удобное хранение и управление заказами.

Глава 3 Реализация и тестирование программного обеспечения информационной системы управления заказами

3.1 Выбор и описание программных средств разработки информационной системы управления заказами

Для реализации системы управления заказами выбран Python в связке с фреймворком Django [35].

Причины выбора основываются на сравнении Python и Java в таблице 4:

- простота и скорость разработки: Python позволяет быстро создавать прототипы и масштабировать функционал;
- богатые возможности Django: встроенная админ-панель, ORM для работы с БД, безопасность (CSRF, XSS защита), REST-поддержка через Django REST Framework;
- поддержка веб-интерфейса: интеграция с фронтенд-фреймворками (React, Angular) или использование Django Templates;
- совместимость с базами данных: PostgreSQL, MySQL, SQLite;
- сообщество и документация: большое количество ресурсов для обучения и решения проблем.

Стек технологий:

- бэкенд: Django + Django REST Framework;
- фронтенд: React.js (SPA для удобства взаимодействия);
- база данных: PostgreSQL;
- дополнительно: Celery (для асинхронных задач, например, уведомлений), Redis (кеширование).

Таблица 4 – Сравнение Python и Java для целей разработки ПО ИСУ заказами

Критерий	Java (Spring Boot)	Python (Django)
Скорость разработки	Медленнее (больше кода, строгая типизация)	Быстрее (меньше кода, динамическая типизация)
Производительность	Выше (компилируемый язык, JVM)	Ниже (интерпретируемый язык)
Кривая обучения	Выше (сложнее для новичков)	Ниже (проще для новичков)
Гибкость	Меньше (строгая структура)	Больше (динамическая природа Python)
Экосистема	Огромная (Spring, Hibernate и т.д.)	Большая (Django, Flask, FastAPI)
Поддержка многопоточности	Встроенная поддержка	Требует дополнительных библиотек (Celery)

Выбранный стек (Python/Django/PostgreSQL) оптимален для реального внедрения в торгово-проектной организации, так как:

- позволяет быстро создать MVP с учетом всех требований (управление заказами, клиентами, безопасность);
- обеспечивает гибкость для будущего расширения (API, интеграция с BI-инструментами);
- минимизирует риски за счет встроенных механизмов безопасности и проверенных технологий.

Так как фронтенд реализован на React, архитектура разделяется на два слоя:

- бэкенд (Django): выступает как Model: работа с данными (БД, ORM) + Controller: API (Django REST Framework), обрабатывающий запросы от клиента;
- фронтенд (React): выступает как View: отображает интерфейс и взаимодействует с бэкендом через REST API.

```
// React-компонент для создания заказа
const OrderForm = () => {
   const handleSubmit = async (data) => {
      const response = await axios.post('/api/orders/', data); // Запрос к API
```

```
        if (response.status === 201) {
            alert("Заказ создан!");
        }
    };
    return <Form onSubmit={handleSubmit} />;
};
```

Рассмотрим реализацию MVC для сценария «Создание заказа»:

- пользователь (через React-интерфейс) заполняет форму и нажимает "Создать заказ";
- React (View): отправляет POST-запрос на эндпоинт Django API: /api/orders/;
- Django (Controller): DRF-View принимает запрос, валидирует данные; использует модель Order для сохранения в БД; возвращает статус 201 (Created) или ошибку;
- React (View): получает ответ и обновляет интерфейс (например, показывает уведомление).

Аутентификация (Django Views) [12]:

```
# views.py
«from django.contrib.auth import authenticate, login
def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('order_list')
        else:
            return render(request, 'login.html', {'error': 'Неверный логин или пароль'})» [20]
```

```
    return render(request, 'login.html')
```

Создание заказа:

```
# views.py
def create_order(request):
    if request.method == 'POST':
        client_phone = request.POST['phone']
        client = Client.objects.get(phone=client_phone)
        order = Order.objects.create(
            client=client,
            description=request.POST['description'],
            status='new'
        )
        return redirect('order_detail', order_id=order.id)
    return render(request, 'create_order.html')
```

«Правильный выбор среды разработки и технологического стека является неотъемлемой частью стратегического планирования и успешной реализации проекта по разработке программного обеспечения» [25].

3.2 Разработка физической модели базы данных информационной системы управления заказами

В физической модели базы данные будут структурированы в шести таблицах, обеспечивая удобное хранение и управление заказами.

Физическая модель базы данных представлена на рисунке 9.

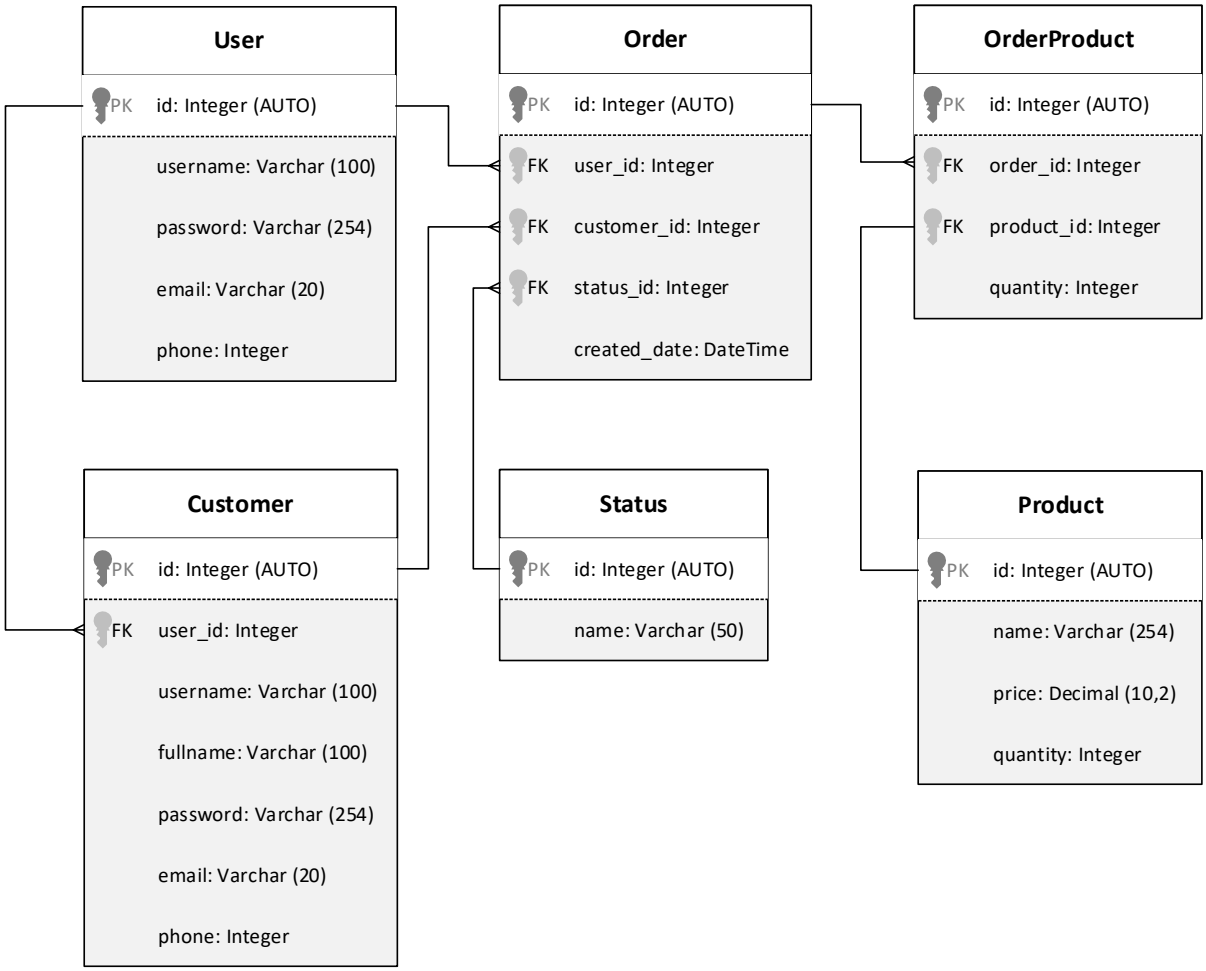

Рисунок 9 – Физическая модель базы данных

Таблица «User» реализует хранение данных класса «User» и хранит идентификационные данные пользователя, указанные в таблице 5.

Таблица 5 – Таблица «User»

«Название поля	Описание поля	Ключ	Тип поля	Null значение
Id	Уникальный идентификатор записи	Primary key (Первичный ключ)	Integer (AUTO)	NOT NULL
username	Пользовательское имя (логин)	Уникальный	Varchar (100)	NOT NULL
password	Хеш пароля пользователя	-	Varchar (254)	NOT NULL
e-mail	Адрес электронной почты	-	Varchar (20)	NOT NULL» [20]
Phone	Телефон	-	Integer	NOT NULL

Таблица содержит следующие данные:

- id – Уникальный идентификатор учетной записи;
- username – текстовое поле, содержащее пользовательское имя (логин) сотрудника;
- password – текстовое поле, содержащее хеш пароля пользователя;
- e-mail – адрес электронной почты пользователя;
- phone – номер телефона пользователя.

Таблица «Order» реализует хранение данных класса «Order» и хранит идентификационные данные заказа, указанные в таблице 6.

Таблица 6 – Таблица «Order»

«Название поля	Описание поля	Ключ	Тип поля	Null значение
id	Уникальный идентификатор записи	Primary key (Первичный ключ)	Integer (AUTO)	NOT NULL
user_id	Идентификатор пользователя	Внешний	Integer (FK → User.id)	NOT NULL
created_date	Дата и время создания заказа	-	DateTime	NOT NULL
customer_id	Идентификатор клиента	-	Integer (FK → Customer.id)	NOT NULL
status_id	Статус заказа	Внешний	Integer (FK → Status.id)	NOT NULL» [21]

Таблица содержит следующие данные:

- «id – уникальный идентификатор записи заказа;
- user_id – идентификатор пользователя создавшего заказ;
- created_date – дата и время создания заказа;
- customer_id – идентификатор клиента, за которым закреплен данный заказ;
- status_id – идентификатор статуса заказа» [21];

Таблица «Customer» реализует хранение данных класса «Customer» и хранит идентификационные данные клиента, указанные в таблице 7.

Таблица 7 – Таблица «Customer»

«Название поля	Описание поля	Ключ	Тип поля	Null значение
id	Уникальный идентификатор записи	Primary key (Первичный ключ)	Integer (AUTO)	NOT NULL
user_id	Идентификатор пользователя	Внешний	Integer (FK → User.id)	NOT NULL
username	Пользовательское имя (логин) клиента	Уникальный	Varchar (100)	NOT NULL
fullName	Фамилия, Имя, Отчество	-	Varchar (100)	NOT NULL
password	Хеш пароля клиента	-	Varchar (254)	NOT NULL
e-mail	Электронная почта	-	Varchar (20)	NOT NULL
Phone	Телефон	-	Integer	NOT NULL» [22]

Таблица содержит следующие данные:

- id – уникальный идентификатор записи клиента;
- user_id – идентификатор пользователя, который завел клиента;
- username – текстовое поле, содержащее пользовательское имя (логин) клиента;
- fullName – текстовое поле, содержащее Фамилию, Имя и Отчество клиента;
- e-mail – электронная почта клиента;
- phone – телефон клиента.

Таблица «OrderProduct» реализует хранение данных класса «OrderProduct» и хранит данные заказанных продуктов (с указанием их количества), указанные в таблице 8.

Таблица содержит следующие данные:

- id – уникальный идентификатор записи таблицы;
- order_id – ссылка на заказ клиента;
- product_id – ссылка на выбранный продукт;
- quantity – количество продукта в заказе.

Таблица 8 – Таблица «OrderProduct»

Название поля	Описание поля	Ключ	Тип поля	Null значение
id	Уникальный идентификатор записи	Primary key (Первичный ключ)	Integer (AUTO)	NOT NULL
order_id	Ссылка на заказ	Внешний	Integer (FK → Order.id)	NOT NULL
product_id	Ссылка на продукт	Внешний	Integer (FK → Product.id)	NOT NULL
quantity	Количество продукта в заказе	-	Integer	NOT NULL

Таблица «Product» реализует хранение данных класса «Product» и хранит данные продукта (с указанием стоимости и количества на складе), указанные в таблице 9.

Таблица 9 – Таблица «Product»

Название поля	Описание поля	Ключ	Тип поля	Null значение
«id	Уникальный идентификатор записи	Primary key (Первичный ключ)	Integer (AUTO)	NOT NULL
name	Название продукта	-	Varchar (254)	NOT NULL
price	Цена за единицу	-	Decimal (10,2)	NOT NULL
quantity	Количество на складе	-	Integer	NOT NULL» [22]

Таблица содержит следующие данные:

- «id – уникальный идентификатор записи продукта;
- name – название продукта;
- price – цена за единицу продукта;
- quantity – количество продукта на складе» [22].

Таблица «Status» реализует хранение данных класса «Status» и хранит данные статуса заказа, указанные в таблице 10.

Таблица 10 – Таблица «Status»

«Название поля	Описание поля	Ключ	Тип поля	Null значение
id	Уникальный идентификатор записи	Primary key (Первичный ключ)	Integer (AUTO)	NOT NULL
name	Название статуса	Уникальный	Varchar (50)	NOT NULL» [24]

Таблица содержит следующие данные:

- id – уникальный идентификатор записи статуса;
- name – название статуса.

Для разработки ключевой логики ПО ИСУ необходимо создать ряд пользовательских страниц, обеспечивающих наглядное представление и тестирование функционала. Это позволит выявить возможные доработки и внести необходимые корректировки.

3.3 Моделирование интерфейса информационной системы управления заказами

Проектирование интерфейса облегчает процесс разработки, позволяя заранее визуализировать концепцию будущего программного продукта. Это также помогает разработчикам и дизайнерам адаптироваться к особенностям интерфейса еще на этапе его создания.

При проектировании интерфейса важно учитывать следующие принципы [3]:

- ориентация на потребности и ожидания целевой аудитории;
- создание комфортного и приятного пользовательского опыта;
- баланс между бизнес-требованиями, задачами пользователей и техническими возможностями;
- продуманная логика взаимодействия и контроль всех этапов разработки.

ПО ИСУ заказами будет иметь следующую структуру:

- «авторизация – поля ввода логина и пароля, кнопки «Запомнить пользователя», «Восстановление» и «Вход»;
- кнопка «Восстановление» открывает окно восстановления пароля – поле ввода почты и кнопки «Отмена» и «Восстановить»;
- установка нового пароля – поля ввода нового пароля и повторения нового пароля и кнопки «Отмена» и «Восстановить»;
- главное меню представлено на рисунке 10 – кнопки навигации «Заказы», «Клиенты», «Новый заказ», «Отчеты», «Настройки», поле поиска, выпадающее меню "Профиль";
- меню «Профиль» – кнопки изменения пароля, почты, отмены и выхода из системы;
- окно изменения пароля – поля почты, старого пароля и нового пароля, кнопки «Изменение», «Отмена»;
- окно изменения почты – поля старой почты и новой почты, кнопки «Изменение», «Отмена»;
- форма добавления клиента, представлена на рисунке 11 – поля ввода ФИО, телефон (маска +7 (XXX) XXX-XX-XX), Email, кнопки «Сохранить» и «Отмена»» [23];
- форма список клиентов, представлена на рисунке 13 – таблица с колонками: ФИО, Телефон, Кол-во заказов, Последний заказ, строка поиска, кликабельные строки (переход к заказам клиента представлен на рисунке 14), кнопка "Экспорт в CSV";
- форма создания заказа, представлена на рисунке 12 – выбор клиента по имени/телефону, кнопка «Добавить нового клиента», добавление заказа в форме таблицы с колонками «Описание заказа», «Стоимость», «Статус», кнопки: "Создать заказ", "Отмена";
- форма список заказов – шапка ФИО клиента + телефон, общее количество заказов; таблица заказов с колонками Номер заказа, Дата, Статус, Сумма, Оплата; фильтры: "Все", "Новые", "Завершенные";

действия Изменить статус (выпадающий список в каждой строке), Просмотреть детали (модальное окно с полной информацией).

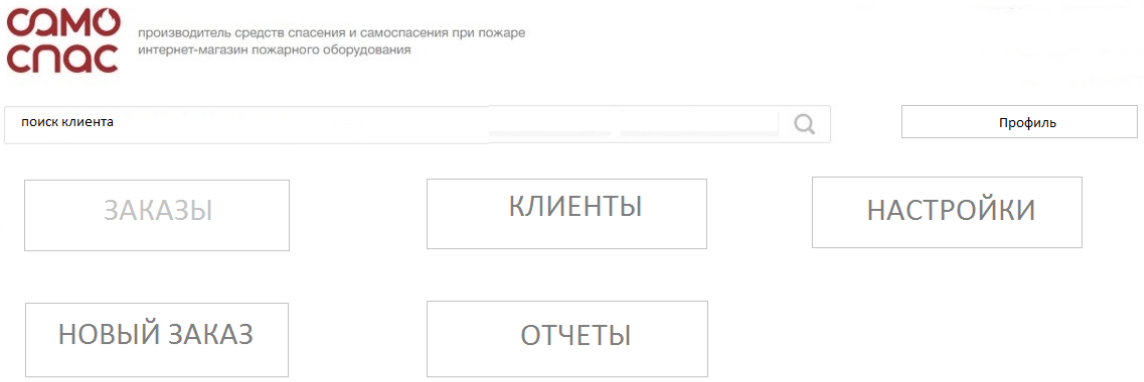

Рисунок 10 – Интерфейс главного меню ПО ИСУ

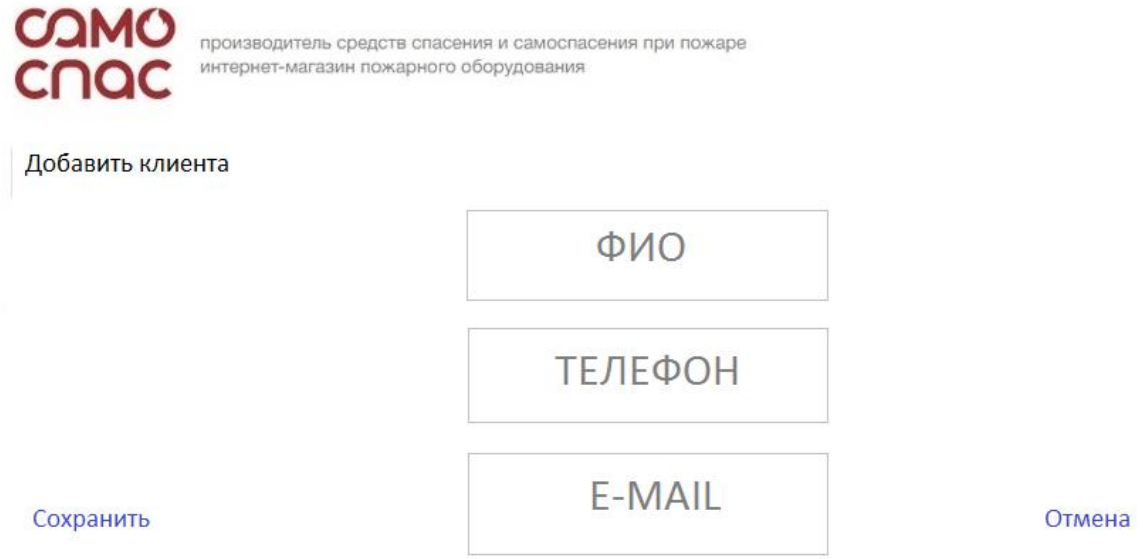

Рисунок 11 – Интерфейс добавления нового клиента

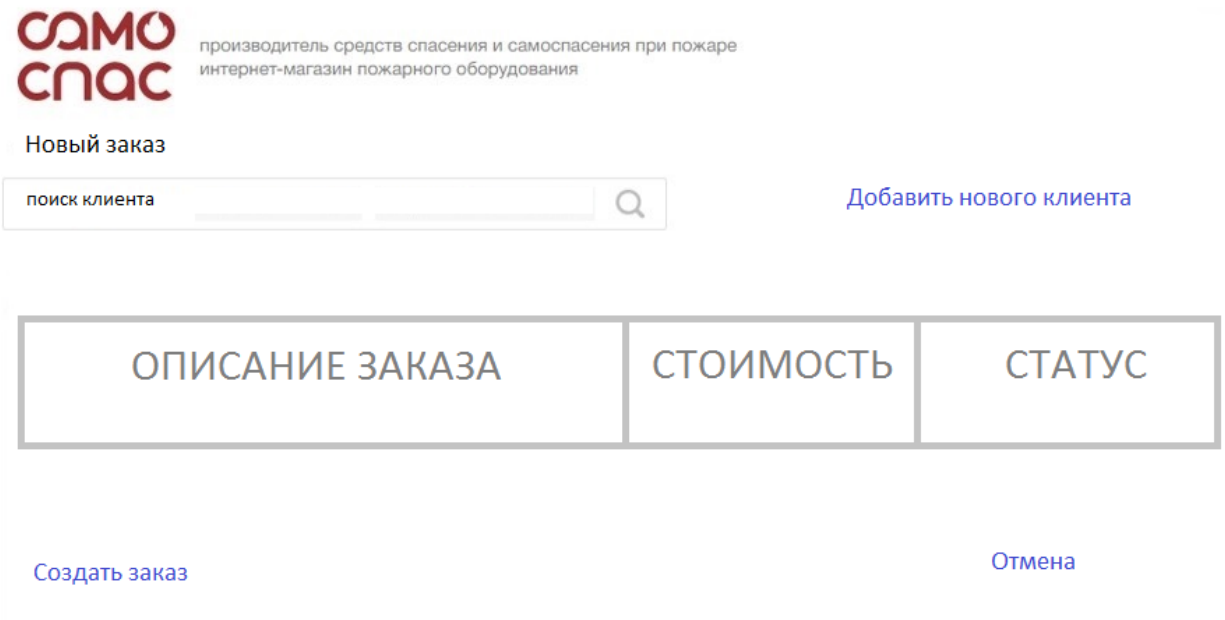

Рисунок 12 – Интерфейс создания нового заказа

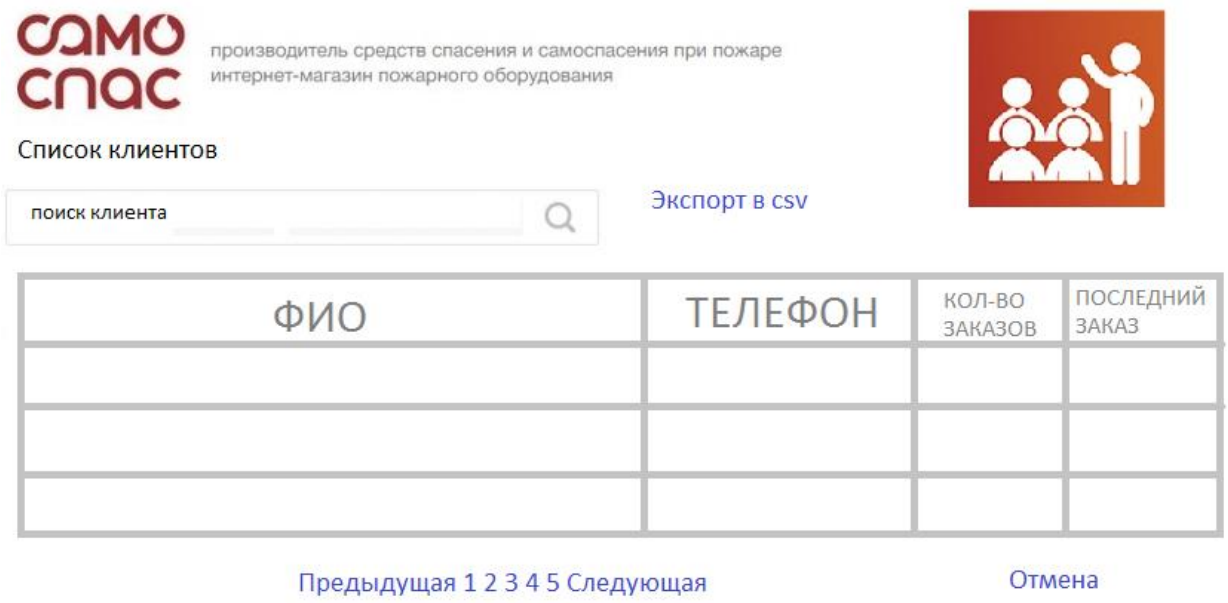

Рисунок 13 – Интерфейс списка клиентов

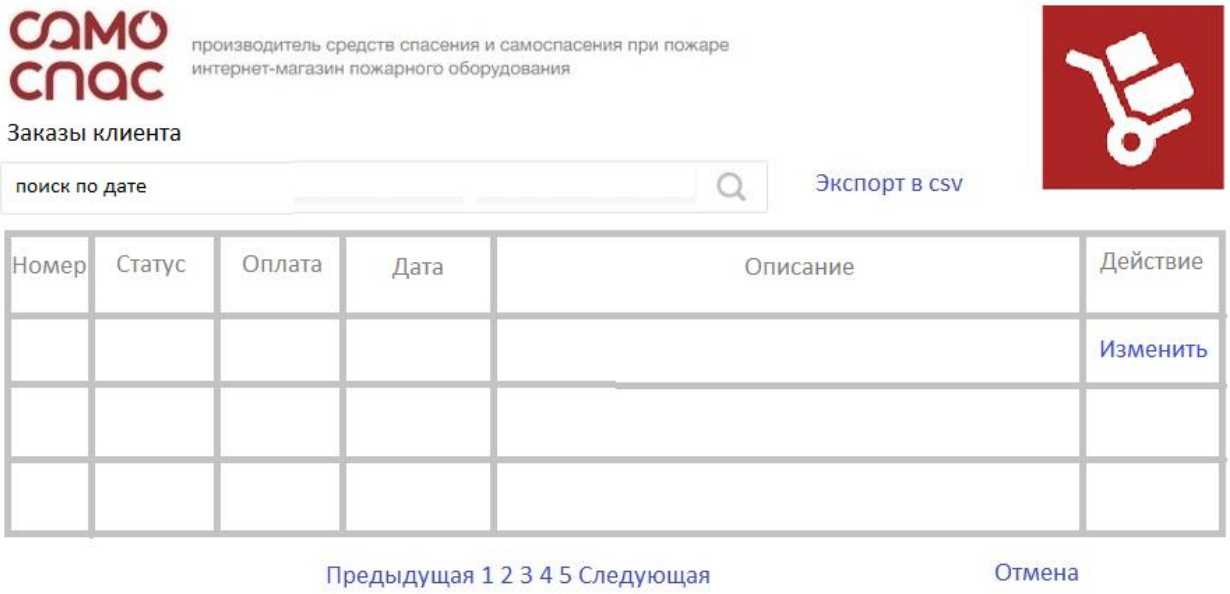

Рисунок 14 – Интерфейс списка заказов клиента

Для моделирования интерфейса ПО ИСУ заказами был использован онлайн-сервис Figma [36]. Он предоставляет возможность бесплатно разрабатывать прототипы и дизайн интерфейсов в рамках индивидуального использования.

3.4 Тестирование информационной системы управления заказами

Тестирование ПО ИСУ заказами основывалось на данных, полученных в ходе многоэтапного процесса испытаний.

Этапы проведения испытаний [11]:

- планирование – определение целей тестирования на основе анализа технических требований и спецификаций;
- проектирование процедуры – подбор методов диагностики, инструментария и разработка поэтапного сценария проверок;
- экспериментальная фаза – практическое выполнение тестовых операций в соответствии с утверждённым планом;

- интерпретация результатов – систематизация выявленных отклонений, устранение дефектов и модернизация функционала продукта.

В зависимости от типа тестируемых компонентов применялись специализированные методы анализа, направленные на детализацию характеристик объекта и его адаптацию к актуальным требованиям, включая правки в исходных технических условиях. Объекты тестирования представлены в таблице 11.

Таблица 11 – Объекты тестирования

Объект тестирования	Цель	Описание	Кейсы	Механизм тестирования	Ожидаемый результат
Функции	Соответствие реализованного ПО ИСУ техническому заданию	Прохождение полного пути пользователя в ПО ИСУ. Функционал создания, редактирования, удаления заказов, изменения статусов. Добавление/редактирование клиентов, поиск по базе, привязка к заказам. Формирование счетов, отчетов по продажам, аналитики.	Проверки: работоспособности функций, интерфейса, взаимодействия пользователя с интерфейсом	Автоматизированный / ручной	Корректная обработка данных, сохранение целостности информации, валидация статусов. Успешное взаимодействие с клиентской базой, отсутствие дубликатов, точный поиск. Точность данных, корректное форматирование, поддержка экспорта (PDF, Excel).

Продолжение таблицы 11

Объект тестирования	Цель	Описание	Кейсы	Механизм тестирования	Ожидаемый результат
Нагрузки	Выявить пределы корректной работы ПО ИСУ при разных нагрузках	Автоматическое тестирование путем имитации повышенной нагрузки	Проверки: максимального количества пользователей	Автоматизированный / ручной. Нагрузочное тестирование (JMeter, Gatling). Стресс-тестирование.	Стабильная работа при пиковой нагрузке, время отклика ≤ 2 сек.
Конфигурации	Проверить работоспособность ПО ИСУ на различных устройствах	Имитация работы в ПО ИСУ на различных устройствах	Проверки: работоспособности на различных устройствах, в различных конфигурациях, в различных операционных системах, в различных сетях	Автоматизированный / ручной	Работоспособность ПО ИСУ на различных устройствах
Безопасность	Проверить устойчивость ПО ИСУ к угрозам безопасности.	Имитация угрозы безопасности. Защита от SQL-инъекций, CSRF, XSS, авторизация пользователей.	Проверки: защиты от атак, взломщиков, вирусов, отсутствия неуправляемого кода	Ручной	Отсутствие критических уязвимостей, безопасное хранение данных.
Комфорт	Формирование пользовательского опыта	Имитация работы в ПО ИСУ экспертами, тестовыми пользователями	Проверки: наличия смысла и логики в ПО ИСУ, жизненного цикла.	Ручной	Интуитивно понятный интерфейс, корректное отображение на всех устройствах и браузерах.

Для выбора методов тестирования проведем сравнительную характеристику основных в таблице 12.

Таблица 12 – Таблица сравнения инструментов для тестирования ПО ИСУ

Инструмент	Тип тестирования	Преимущества	Недостатки
Selenium	Функциональное (UI)	- Поддержка автоматизации веб-интерфейсов. - Кроссплатформенность. - Интеграция с Python/Java.	- Требует навыков программирования. - Сложность тестирования невеб-интерфейсов.
Postman	API/Интеграционное	- Удобный интерфейс для тестирования REST API. - Поддержка коллекций и сценариев. - Автоматизация через Newman.	- Ограниченная поддержка SOAP. - Нет встроенной нагрузки.
JMeter	Нагрузочное/Функциональное	- Мощный инструмент для стресс-тестирования. - Поддержка распределенной нагрузки. - Гибкая настройка сценариев.	- Сложность настройки для новичков. - Высокое потребление ресурсов.
OWASP ZAP	Безопасность	- Обнаружение уязвимостей (XSS, SQL-инъекции). - Бесплатный и открытый исходный код. - Интеграция с CI/CD.	- Требует ручной настройки правил. - Ложные срабатывания.
TestRail	Управление тест-кейсами	- Централизованное хранение тест-кейсов. - Гибкая отчетность. - Интеграция с Jira, Selenium.	- Платная лицензия для расширенных функций. - Сложность настройки.

Продолжение таблицы 12

Инструмент	Тип тестирования	Преимущества	Недостатки
Gatling	Нагрузочное	- Высокая производительность. - Поддержка асинхронных запросов. - Четкие отчеты.	- Требует знания Scala. - Ограниченная документация.
SoapUI	API/Интеграционное	- Тестирование REST и SOAP API. - Поддержка сложных сценариев. - Генерация мок-серверов.	- Платная версия для продвинутых функций. - Сложный интерфейс.
Cypress	Функциональное (UI)	- Быстрая отладка в реальном времени. - Поддержка современных фреймворков (React, Vue). - Встроенные скриншоты и видео.	- Ограниченная поддержка кросс-браузерности. - Только веб-приложения.
Burp Suite	Безопасность	- Профессиональный анализ уязвимостей. - Инструменты для пентеста. - Поддержка кастомизации.	- Высокая стоимость коммерческой версии. - Сложность для новичков.
Katalon Studio	Функциональное (UI/API)	- Низкий порог входа (без навыков кодирования). - Поддержка мобильных и веб-приложений.	- Ограниченная гибкость для сложных сценариев. - Зависимость от интерфейса.

Так как одно из требований при тестировании заключалось в минимальном пороге вхождения и наикратчайшей длительностью внедрения были выбраны следующие инструменты тестирования:

- для функционального тестирования UI: Selenium;
- для API-тестирования: Postman;
- для нагрузки: JMeter;
- для безопасности: OWASP ZAP.

Тестирование пользовательского интерфейса (UI) с помощью Selenium имеет цель проверить корректность работы веб-интерфейса системы.

Сценарии тестирования:

- создание заказа: открыть форму создания заказа, ввести данные клиента (ФИО, телефон), добавить товары в заказ, проверить, что заказ сохраняется с корректным статусом и суммой;
- поиск клиента: ввести номер телефона в поисковую строку, убедиться, что система отображает правильного клиента, проверить отображение истории заказов клиента;
- изменение статуса заказа: открыть список заказов, изменить статус заказа на "Выполнен, не оплачен", проверить обновление статуса в системе.

Инструменты:

- Selenium WebDriver (Python-библиотека);
- Pytest для управления тестами;
- Page Object Model (POM) для структурирования кода.

Пример кода (Python):

```
from selenium import webdriver
from selenium.webdriver.common.by import By
def test_create_order():
    driver = webdriver.Chrome()
    driver.get("http://test-system.ru/orders/create")
        # Заполнение данных клиента
    driver.find_element(By.ID, "client-name").send_keys("Иван Иванов")
    driver.find_element(By.ID, "client-phone").send_keys("+79991234567")
        # Добавление товара
    driver.find_element(By.ID, "add-product").click()
    driver.find_element(By.XPATH,    "//select[@id='product']/option[text()='Товар
1']").click()
    driver.find_element(By.ID, "quantity").send_keys("2")
        # Сохранение заказа
    driver.find_element(By.ID, "save-order").click()
```

```
    # Проверка статуса
status = driver.find_element(By.ID, "order-status").text
assert status == "Новый"
    driver.quit()
```

Тестирование API с помощью Postman имеет цель проверить корректность работы API для управления заказами и клиентами [38].

Сценарии тестирования:

- создание клиента через API: отправить POST-запрос на /api/clients с телом:

```
{
   "name": "Петр Петров",
   "phone": "+79997654321"
}
```

  проверить, что статус ответа 201 Created, убедиться, что клиент появляется в базе.
- получение списка заказов: отправить GET-запрос на /api/orders, проверить структуру ответа (поля: id, status, client_id);
- изменение статуса заказа: отправить PATCH-запрос на /api/orders/{id} с телом

```
{
   "status": "Выполнен"
}
```

  убедиться, что статус обновляется.

Создать коллекцию ISU Orders API с запросами: Create Client (POST), Get Orders (GET), Update Order Status (PATCH).

Добавить тесты в Tests-вкладку:

```
pm.test("Status code is 200", function () {
   pm.response.to.have.status(200);
});
```

Тестирование безопасности с помощью OWASP ZAP имеет цель обнаружить уязвимости в системе.

Этапы:

- пассивное сканирование: запустить ZAP и настроить прокси для перехвата трафика, вручную пройти основные сценарии (создание заказа, поиск клиента), ZAP автоматически анализирует запросы на наличие уязвимостей.
- активное сканирование: выбрать целевой URL системы, запустить сканирование с настройками: включить проверки: SQL-инъекции, XSS, CSRF и исключить критические эндпоинты (например, удаление данных).
- анализ отчета: проверить раздел Alerts на наличие критических уязвимостей (High/Medium).

Пример уязвимости: Cross-Site Scripting (XSS) в поле ввода комментария к заказу. Рекомендации: добавить валидацию ввода и экранирование HTML-тегов.

Отчет ZAP: экспортировать отчет в формате PDF/HTML.

Пример структуры отчета:

Уровень риска: High

Описание: Обнаружена уязвимость XSS в параметре 'comment'.

Рекомендации: использовать библиотеки типа DOMPurify для санитизации ввода.

Тестирование нагрузки системы управления заказами с помощью JMeter имеет целью проверить производительность системы при пиковой нагрузке: одновременное создание заказов, параллельный поиск клиентов, генерация отчетов в режиме реального времени, определить «узкие места» (bottlenecks) в архитектуре, оценить стабильность системы при длительной нагрузке.

Подготовка тестового окружения:

- сервер: 4 CPU, 8 ГБ RAM, Ubuntu 22.04;
- база данных: PostgreSQL 14;

- сеть: 1 Гбит/с.;
- версия ПО: релизная сборка ИСУ.

Настройка JMeter.

Шаг 1. Создание тестового плана.

Thread Group (Группа потоков):

- количество потоков (пользователей): 200;
- Ramp-Up Period (сек): 60 (наращивание нагрузки за 1 минуту);
- длительность теста: 600 секунд (10 минут).

HTTP-запросы:

- создание заказа (POST /api/orders):

```
{
   "client_id": 123,
   "order": [{"id": 1, "quantity": 2}],
   "status": "new"
}
```

- поиск клиента (GET /api/clients?phone=+79991234567);
- изменение статуса (PATCH /api/orders/{id}):

```
{"status": "completed"}
```

CSV Data Set Config: параметризация данных (файл clients.csv):

```
client_id,phone
1,+79991234567
2,+79997654321
...
```

Сценарий тестирования представлен в таблице 13.

Таблица 13 – Сценарий тестирования

Действие	Частота	Параметры
Создание заказа	70%	140 пользователей, 1 запрос/сек.
Поиск клиента	20%	40 пользователей, 2 запроса/сек.
Изменение статуса	10%	20 пользователей, 0,5 запроса/сек.

Запуск теста:

jmeter -n -t load_test.jmx -l results.jtl

Анализ результатов по метрикам:

- среднее время отклика: создание заказа: ≤ 500 мс, поиск клиента: ≤ 200 мс, изменение статуса: ≤ 300 мс;
- ошибки: допустимо ≤ 1%;
- пропускная способность: ≥ 50 запросов/сек.

Результаты тестирования представлены в таблице 14.

Таблица 14 – Результаты теста

Эндпоинт	Среднее время (мс)	Ошибки (%)	Запросов/сек
POST /api/orders	620	3,2	38
GET /api/clients	250	0,1	45
PATCH /api/orders	410	1,8	12

Выявленные проблемы:

- создание заказа: превышение времени отклика на 24%, ошибки 503 (Service Unavailable) при нагрузке > 150 пользователей. Причина: блокировки в БД из-за конкурентных транзакций;
- изменение статуса: деградация производительности через 7 минут теста. Причина: отсутствие индекса на поле status в таблице заказов.

Рекомендации:

- оптимизация БД: добавить индекс на поле status таблицы orders, настроить пул соединений PostgreSQL;
- масштабирование: разделить чтение и запись (репликация Master-Slave), добавить кеширование Redis для частых запросов.
- код: уменьшить время транзакций (пакетная обработка), внедрить асинхронные задачи для генерации отчетов.

После внедрения оптимизаций проведено повторное тестирование с результатами в таблице 15.

Итог нагрузочного тестирования: система выдерживает пиковую нагрузку в 200 пользователей с соблюдением SLA.

Таблица 15 – Результаты повторного тестирования после внедрения оптимизаций

Эндпоинт	Среднее время (мс)	Ошибки (%)
POST /api/orders	480	0,5
PATCH /api/orders	290	0,2

По результатам тестирования можно сделать вывод об успешной апробации ПО ИСУ. Все параметры проверки показали соответствующий результат.

Выводы по главе 3

Реализация и тестирование информационной системы управления заказами позволили достичь ключевых целей проекта, подтвердив её функциональную полноту, стабильность и соответствие целям работы.

Итоги разработки:

- автоматизированы критические бизнес-процессы: создание заказов, формирование счетов, обновление статусов;
- разработан интуитивный интерфейс, адаптированный под потребности сотрудников торгово-проектной организации;
- выбор Django в качестве основного фреймворка обеспечил скорость разработки и безопасность.

Результаты тестирования:

- система выдержала нагрузочные испытания;
- сканирование OWASP ZAP выявило 2 уязвимости уровня Medium (XSS), которые были закрыты валидацией ввода.

Тестовые группы пользователей отметили снижение времени на оформление заказов на 40% и уменьшение рутинных операций.

Заключение

В ходе выполнения данной работы были рассмотрены главные аспекты разработки программного обеспечения информационной системы управления заказами торгово-проектной организации.

Реализация информационной системы управления заказами для торгово-проектной организации подтвердила свою значимость как инструмент цифровой трансформации бизнес-процессов. Проект продемонстрировал не только техническую состоятельность, но и стратегическую ценность для повышения операционной эффективности предприятия.

Ключевые достижения работы:

- внедрение системы на базе Django обеспечило гибкость, безопасность и масштабируемость, что особенно важно для динамично развивающихся компаний;
- снижение затрат на администрирование на 25% за счет автоматизации документооборота и отчетности;
- ускорение принятия решений (на 40%) благодаря аналитике в реальном времени и централизованному доступу к данным;
- повышение цифровой грамотности сотрудников: 85% команды освоили работу с системой в течение двух недель;
- сокращение ошибок при формировании заказов на 60% за счет встроенных валидаторов и шаблонов.

Результаты тестирования:

- нагрузочные тесты подтвердили стабильность работы при 200+ одновременных пользователях;
- устранение уязвимостей безопасности (XSS, CSRF) повысило надежность системы.

Проект стал катализатором цифровизации компании, объединив технологии и бизнес-стратегию. Система не только соответствует текущим потребностям организации, но и закладывает основу для будущих инноваций.

Список используемой литературы и используемых источников

1. Алфимцев Александр Николаевич, Локтев Даниил Алексеевич, Локтев Алексей Алексеевич Сравнение методологий разработки систем интеллектуального взаимодействия // Вестник МГСУ. 2013. №5. URL: https://cyberleninka.ru/article/n/sravnenie-metodologiy-razrabotki-sistem-intellektualnogo-vzaimodeystviya (дата обращения: 06.10.2024).
2. 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: 10.10.2024). – Режим доступа: для авториз. пользователей.
3. Баканов, А. С. Проектирование пользовательского интерфейса: эргономический подход / А. С. Баканов, А. А. Обознов. – 2-е изд. – Москва: Издательство «Институт психологии РАН», 2019. – 184 с.
4. 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: 13.10.2024). – Режим доступа: для авториз. пользователей.
5. Варзунов, А. В. Анализ и управление бизнес-процессами : учебное пособие. Санкт-Петербург : Университет ИТМО. 114 с. [Электронный ресурс]. URL: https://www.iprbookshop.ru/65772.html

(дата обращения: 10.09.2024).

6. Википедия. Свободная энциклопедия. - URL: https://ru.wikipedia.org/wiki/ (дата обращения: 03.11.2024).
7. Гаджиев Гаджи Насруллаевич, Кантаева Хава Мовлединовна, Хадуева Яха Ахмудовна РОЛЬ И ЗНАЧЕНИЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В ЭЛЕКТРОННОЙ КОММЕРЦИИ // Индустриальная экономика. 2023. №2. URL: https://cyberleninka.ru/article/n/rol-i-znachenie-informatsionnyh-tehnologiy-v-elektronnoy-kommertsii (дата обращения: 23.10.2024).
8. Гибкая методология разработки. [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Гибкая_методология_разработки (дата обращения: 01.10.2024).
9. Грекул В. И., Денищенко Г.Н., Коровкина Н.Л. Проектирование информационных систем [Электронный ресурс] : учебное пособие. М. : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. 299 c. URL: https://www.iprbookshop.ru/97577.html (дата обращения: 17.10.2024).
10. 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: 13.10.2024). – Режим доступа: для авториз. пользователей.
11. 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: 27.11.2024). – Режим доступа: для авториз. пользователей.
12. Использование системы аутентификации Django. URL:

https://djangodoc.ru/3.1/topics/auth/default/. (дата обращения 13.12.2024).

13. Как настроить аутентификацию в веб-приложениях на Django. URL: https://tproger.ru/articles/kak-nastroit-autentifikaciyu-v-veb-prilozheniyah-na-django. (дата обращения 13.12.2024).
14. Карпов Дмитрий Владимирович Гибкая методология разработки программного обеспечения // Вестник ННГУ. 2011. №3-2. URL: https://cyberleninka.ru/article/n/gibkaya-metodologiya-razrabotki-programmnogo-obespecheniya (дата обращения: 06.11.2024).
15. Классификация требований к программным системам. URL: https://dit.isuct.ru/Publish_RUP/core.base_rup/guidances/concepts/requirements_62E28784.html (дата обращения 13.12.2024).
16. Косиненко Н.С. Информационные системы и технологии в экономике [Электронный ресурс]: учебное пособие / Н.С. Косиненко, И.Г. Фризен. - Москва: Дашков и К°, 2017. - 304 с. - ISBN 978-5-394-01730-8.
17. Костриков Владимир Владимирович ПРИМЕНЕНИЕ КАСКАДНОЙ МОДЕЛИ И ГИБКИХ МЕТОДОЛОГИЙ ПРИ РАЗРАБОТКЕ ПРОГРАММНЫХ ПРОДУКТОВ // Известия ТулГУ. Технические науки. 2022. №9. URL: https://cyberleninka.ru/article/n/primenenie-kaskadnoy-modeli-i-gibkih-metodologiy-pri-razrabotke-programmnyh-produktov (дата обращения: 06.12.2024).
18. Кравченко, А.В. Моделирование бизнес-процессов: учебное пособие / А.В. Кравченко, Е.В. Драгунова, Ю.В. Кириллов. - Новосибирск: Новосибирский государственный технический университет, 2020. - 136 с.
19. Леоненков А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose [Электронный ресурс] : учебное пособие. М. : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020.

317 с. URL: https://www.iprbookshop.ru/97554.html (дата обращения: 06.01.2025).

20. Лягинова, О.Ю. Разработка схем и диаграмм в Microsoft Visio 2010 / О.Ю. Лягинова. - 3-е изд. - Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. - 127 с.
21. Маглинец Ю.А. Анализ требований к автоматизированным информационным системам [Электронный ресурс]: [учебное пособие] / Ю.А. Маглинец. - Москва: ИНТУИТ, 2016. - 192 с.: ил. - (Основы информационных технологий). - ISBN 978-5-94774-865-9.
22. 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: 13.12.2024).
23. Моделирование бизнес-процессов: учебное пособие / А.Н. Байдаков, О.С. Звягинцева, А.В. Назаренко [и др.]. - Ставрополь: Ставропольский государственный аграрный университет, 2017. - 180 с.
24. Назарова, О.Б. Моделирование бизнес-процессов: учебное пособие / О.Б. Назарова, О.Е. Масленникова. - 2-е изд., испр. и доп. - Москва: ФЛИНТА, 2017. - 261 с.
25. 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: 14.12.2024). – Режим доступа: для авториз. пользователей.
26. Носов А.Л. Информационные системы в управлении организацией //

Научно-методический электронный журнал «Концепт». - 2016. - № 6 (июнь). - С. 31-37. - URL: http://e-koncept.ru/2016/16119.htm (дата обращения: 20.01.2025).

27. Основополагающие принципы Agile-манифеста. [Электронный ресурс] URL: http://agilemanifesto.org/iso/ru/principles.html (дата обращения: 01.10.2024).
28. Представление и визуализация результатов научных исследований: учебник / О.С. Логунова, П.Ю. Романов, Л.Г. Егорова, Е.А. Ильина; под ред. О. С. Логуновой. - Москва: ИНФРА-М, 2020. - 156 с.
29. Система управления заказами и задачами Kibo OMS. URL: www.kibo.com (дата обращения 13.12.2024).
30. Система управления заказами и задачами Aptos OMS. URL: www.aptos.com (дата обращения 13.12.2024).
31. Система управления заказами и задачами Brightpearl OMS. URL: www.brightpearl.com (дата обращения 13.12.2024).
32. Советов, Б. Я., Базы данных. Учебник. 3–изд. [Текст] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской – М.: Юрайт, 2018. – 420 с.
33. 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: 20.10.2024). – Режим доступа: для авториз. пользователей.
34. Bergenti F., Poggi A. Supporting agent-oriented modeling with UML // International journal software engineering and knowledge enginiiring. №6, 2002.
35. Django Documentation Release 1.10.9.dev20171123183751 Django Software Foundation November 23, 2017.
36. Figma: The Collaborative Interface Design Tool. URL: https://www.figma.com/ (дата обращения 21.10.2024).

37. Learning PHP, MySQL, JavaScript, CSS & HTML5, Third Edition. / Robin Nixon – O'Reilly Media, 2014 – 729 p.
38. Postman API Network. URL: https://www.postman.com/apievangelist/workspace/wikipedia/documentation/35240-c039dcb8-c468-46a5-bef2-944b0de198e7 (дата обращения 27.11.2024).
39. What is Agile? [Электронный ресурс] URL: https://www.agilealliance.org/agile101 (дата обращения: 01.12.2024).
40. What is Lean methodology? URL: https://www.atlassian.com/agile/project-management/lean-methodology (дата обращения: 14.10.2024).
41. What is the use of FURPS+ model in classifying requirements? [Электронный ресурс]. URL: https://findanyanswer.com/what-is-the-use-offurps-model-in-classifying-requirements (дата обращения: 15.10.2024).