ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения заочная

«К ЗАЩИТЕ ДОПУЩЕН(А)»
Заведующий кафедрой
_____
(подпись)
Черняева С. Н.
(ФИО)
_____ 20___

Выпускная квалификационная работа

Обучающегося Корнетовой Екатерины Романовны
(фамилия, имя, отчество)
Вид работы выпускная квалификационная работа бакалавра
(выпускная квалификационная работа бакалавра, специалиста, магистра)

Пояснительная записка

Тема Разработка веб-приложения системы реализации товаров через электронный портал организации (на примере ООО «АГРОТОРГ»)

(полное название темы квалификационной работы, в соответствии с приказом об утверждении тематики ВКР)

Руководитель работы к.э.н., доцент Скрипников О.А. 21.06.2024
(должность, подпись, фамилия, инициалы, дата)
Консультант _____
(при наличии) (должность, подпись, фамилия, инициалы, дата)
Консультант _____
(должность, подпись, фамилия, инициалы, дата)
Обучающийся Корнетова Е.Р. 21.06.2024
(подпись, фамилия, инициалы, дата)

Воронеж
2024

ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения заочная

УТВЕРЖДАЮ
Заведующий кафедрой
_____
(подпись)
Черняева С. Н.
(ФИО)
_____ 2024

Задание
на выпускную квалификационную работу

Вид работы ВКР бакалавра
(ВКР бакалавра, ВКР специалиста, ВКР магистра)

Обучающемуся Корнетовой Екатерине Романовной
(фамилия, имя, отчество)

Тема Разработка веб-приложения системы реализации товаров через электронный портал организации (на примере ООО «АГРОТОРГ»)

Утверждена приказом ректора университета от _____ 20___, № _____

Срок сдачи законченной работы _____ 20___

Исходные данные (или цель ВКР):
Разработать веб-приложение системы реализации товаров через электронный портал организации ООО «АГРОТОРГ»

Перечень подлежащих исследованию, разработке, проектированию вопросов (краткое содержание ВКР):
(актуальность темы, цели и задачи ВКР; аналитический обзор литературных источников; постановка задачи исследования, разработки, проектирования; содержание процедуры исследования, разработки, проектирования; обсуждение результатов; дополнительные вопросы, подлежащие разработке; заключение – выводы по работе в целом, оценка степени решения поставленных задач, практические рекомендации; и др.)

– Введение. Актуальность выбранной темы, цель и задачи ВКР

(наименование вопроса, раздела и его краткое содержание)

– Исследовательский раздел. Профиль деятельности ООО «АГРОТОРГ». Постановка задачи. Анализ существующих разработок, выбор и обоснование стратегии автоматизации и способа приобретения информационной системы. Обоснование выбора технологии разработки. Определение обоснования разработанных стратегий.

(наименование вопроса, раздела и его краткое содержание)

– Проектный раздел. Информационное обеспечение задачи (комплекса задач, автоматизированное рабочее место). Программное обеспечение задачи (комплекса задач, автоматизированное рабочее место). Технологическое обеспечение задачи (комплекса задач, автоматизированное рабочее место). Руководство пользователя

(наименование вопроса, раздела и его краткое содержание)

– Заключение. Выводы по работе в целом. Оценка степени решения поставленных задач

(наименование вопроса, раздела и его краткое содержание)

Практические рекомендации

Перечень графического материала (или презентационного материала):

1. Титульный лист
2. Цель и задачи ВКР
3. Корпоративная ИТ-система
4. Системные требования
5. Необходимое программное обеспечение
6. Необходимые данные
7. Типы данных
8. Используемые средства
9. Технологические операции
10. Интерфейс пользователя
11. Результаты ВКР

Консультанты по разделам ВКР (при наличии):

1. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

2. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

3. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

Дата выдачи задания: _____ 20___

Задание согласовано и принято к исполнению: _____ 20___

Руководитель ВКР: к.э.н., доцент, Скрипников Олег Александрович _____

(должность, ученая степень, ученое звание, ФИО) (подпись)

Обучающийся: ИТ-4-2, Корнетова Екатерина Романовна _____

(учебная группа, ФИО) (подпись

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В эру цифровизации торговые процессы активно перемещаются в интернет, делая электронную коммерцию ключевым направлением развития рынка. Создание веб-приложений, фокусирующихся на электронной дистрибуции продукции чрезвычайно важно, поскольку оно направлено на оптимизацию торговых операций и повышение качества обслуживания, укрепляя при этом связь между компанией и потребителями.

Создание веб-приложений, направленных на сферу электронной торговли, предоставит компании ООО «АГРОТОРГ», владеющей сетью «Пятерочка», инструменты для формирования индивидуализированных интерфейсов, что существенно повысит удобство пользования сервисами и укрепит верность клиентов. Эти программные решения упрощают процесс поиска и приобретения товаров через Интернет. Они также открывают пути для реализации рекламных акций, сбора и анализа информации о транзакциях и предпочтениях потребителей, что является ключевым для совершенствования бизнес-модели и стратегического планирования. В итоге, внедрение таких разработок значительно повышает конкурентные преимущества предприятия и способствует его виртуальному расширению.

Целью дипломной работы является создание веб-приложения для ООО «АГРОТОРГ» (сеть магазинов «Пятерочка»), упрощающего процесс управления продажами товаров через веб-интерфейс, исключая необходимость установки дополнительных программ на компьютеры сотрудников. Это решение предоставляет оперативный доступ к перечню ассортимента, а также функционал для онлайн-управления заказами.

Задачи выпускной квалификационной работы:

– анализ и исследование механизма регистрации заказов товаров в компании ООО «АГРОТОРГ» (торговая сеть «Пятерочка»);

– изучение доступных программных инструментов для ведения учета и управления заказами;

– создание веб-приложения для продажи и инвентаризации продукции через веб-сайт компании;

– разработка основополагающей схемы веб-платформы для цифровой торговли продуктами в рамках электронной витрины компании ООО «АГРОТОРГ» (торговая сеть «Пятерочка»);

– инструкция по применению системы для пользователей.

Тема исследования – механизмы продажи продукции в предприятии ООО «АГРОТОРГ» (торговая сеть «Пятерочка»).

Исследование затрагивает программное и аппаратное обеспечение, используемое для создания и разработки автоматизированной информационной системы (АИС).

Методы исследования:

– объектно-ориентированное программирование (ООП);

– анализ структурно-функциональный;

– принципы и методики шифрования и категоризации;

– подходы оценки экономической результативности

– подход к разработке диаграммы сущность-связь.

В ходе анализа будут исследованы ключевые аспекты создания веб-приложений для системы электронной коммерции, включая техники взаимодействия с базами данных и платежными шлюзами, а также принципы разработки интерфейса, направленные на повышение удобства для конечных пользователей.

Этот проект позволит компании ООО «АГРОТОРГ», владеющей розничной сетью «Пятерочка», совершенствовать свои складские операции и логистику, внедрить автоматизированные системы для упрощения процедур заказа и оплаты товаров, а также улучшить качество обслуживания благодаря оптимизации взаимодействия с клиентами.

1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 Профиль деятельности ООО «АГРОТОРГ»

ООО «АГРОТОРГ» зарегистрировано в 1998 году. Входит в X5 Retail Group и является крупной федеральной торговой сетью «Пятерочка», одна из самых больших российских сетей продуктовых магазинов «у дома». X5 Retail Group была образована более 25 лет назад из предпринимательской инициативы создать в России компанию актуальной розничной торговли по ведущим мировыми практикам. Первый «Перекресток» был открыт в 1995 году, а уже в 1999-ом – первая «Пятерочка». X5 Retail Group продолжает сохранять предпринимательский настрой и нацелена на создание ценности для общества и акционеров за счет достижения нового уровня технологичности и конкурентоспособности, оставаясь игроком номер один на рынке продовольствия.

ООО «АГРОТОРГ» – юридическое лицо по законодательству Российской Федерации. Организация для достижения установленной уставной цели имеет право от совершать сделки от своего имени, приобретать имущественные и личные неимущественные права, быть истцом и ответчиком в суде, нести обязанности. Предприятие осуществляет пользование, владение и распоряжение своим имуществом в соответствии с целями своей деятельности. ООО «АГРОТОРГ» имеет свой баланс. Одним из основных видов деятельности предприятия ООО «АГРОТОРГ» является торговля в неспециализированных магазинах.

Целью компании ООО «АГРОТОРГ» является предоставление своим покупателям высококачественных продуктов питания, свежести которых гарантирована, по ценам, доступным для широкого круга потребителей.

Основание торговой марки «Пятерочка» датируется 1998 годом, при этом первый торговый объект под этим брендом открыл свои двери в 1999 году на территории Санкт-Петербурга. Ключевым этапом в развитии

компании стал 2016 год, когда был запущен процесс ребрендинга, зафиксированны изменения в стратегии позиционирования и реализована обширная программа модернизации сети. В настоящее время под эгидой корпорации функционирует более 11 тысяч магазинов на просторах России. «Пятерочка» выделяется на фоне конкурентов рядом ключевых достоинств:

Во всех торговых точках данной сети обеспечивается наличие только свежих продуктов благодаря нововведению – роли директора по качеству и свежести. Этот специалист бдительно следит за тем, чтобы на полках присутствовали исключительно товары, соответствующие требованиям по срокам годности. Такой подход гарантирует покупателям возможность приобретения продукции высокого качества, укрепляя их доверие к торговой марке.

Аффордабельные и предпочтительные стоимости. Лидирующие по популярности товары предлагаются по минимуму и справедливым ценникам, тогда как товары из категории социально необходимых реализуются без прибавочной стоимости. Это обеспечивает повышенную доступность продукции для широкого круга потребителей, несмотря на их финансовое положение:

– льготы для пожилых людей. Во всех торговых точках сети «Пятерочка» предусмотрены дисконтные предложения для пенсионеров до указанного часа и в заранее определенные дни;

– преференции для клиентов с несовершеннолетними. Во все среды, начиная с 9 утра и заканчивая 17 вечера, предусмотрены скидочные предложения на ассортимент товаров для посетителей, сопровождающихся детьми до 12 лет.

Сеть супермаркетов «Пятерочка» находится в шаговой доступности для многих и постоянно стремится к улучшению, интегрируя передовые технологические новинки для повышения уровня комфорта своих покупателей. В рамках этого процесса происходит развертывание цифровых

ценников, ввод в эксплуатацию пунктов самостоятельного расчета и обновление линейки торгового оборудования, включая холодильные и замораживающие устройства.

Торговая сеть «Пятерочка» охватывает всю территорию Российской Федерации, и в контексте выпускной квалификационной работы анализируется один из ее объектов, расположенный по адресу: г. Лебедянь, ул. Тургенева, 12, помещение 2. Размер данного торгового зала составляет 594 кв. м.

Экономические показатели выбранного магазина:

Ежедневный объем продаж торговой точки составляет 495 000 рублей.

Дневной оборот магазина составляет 489 193 руб.

Ежемесячный оборот товаров магазина составляет 14 850 000 рублей.

Ежемесячный оборот торговой точки составляет 14 675 790 рублей.

Ассортимент продуктов в «Пятерочке»:

1. Торговля продовольствием – ритейлер выставляет на витрины многообразие позиций, включая агропродукцию в виде овощей и фруктов, а также лактозные изделия, мясные и рыбные деликатесы, сладости и прочее.

2. Основные бытовые товары – в ассортименте присутствуют основные бытовые товары, включающие в себя бытовую химию, гигиенические продукты, чистящие агенты и прочее. В ассортименте присутствуют основные бытовые товары, включающие в себя бытовую химию, гигиенические продукты, чистящие агенты и прочее.

3. В магазине регулярно организуются промоакции, сезонные уценки и дисконты на ассортимент, что способствует экономии средств покупателей.

4. Магазин нацелен на максимизацию удовлетворенности клиентов, гарантируя простоту доступа к торговым точкам, разнообразие предлагаемых продуктов и высокое качество обслуживания.

5. В ассортименте определенных торговых точек Пятерочка представлена функция интернет-покупок, позволяющая клиентам

осуществлять заказ продуктов через сеть Интернет с последующей доставкой к вашей двери.

6. Экологическая ответственность – сеть «Пятерочка» активно занимается ее соблюдением.

На сегодняшний день, торговая сеть «Пятерочка» предоставляет услугу доставки товаров непосредственно из своих филиалов, используя для этого специализированное мобильное приложение. Электронное приложение «Пятерочка» обеспечивает простоту и доступность онлайн-заказов, позволяя клиентам выбирать необходимые продукты и получать их на домашний адрес с максимальным комфортом.

1.2 Постановка задачи

Задачей проекта является создание веб-приложения, способствующего эффективности процесса продажи продукции через электронную платформу компании. Это предполагает реализацию ряда задач:

1. Анализ методик учета ассортимента в секторе ретейла с акцентом на специфику ведения учета в ритейлере «Пятерочка» и нормативные требования к учетным процедурам товаров.

2. Исследование текущих методов записи товарных запасов и цифровых инструментов для ритейла, охватывающее интернет-программы.

3. Указать функциональные спецификации для интернет-приложения, охватывая реализацию механизмов регистрации пользователей, архивации данных, коммерциализации продуктов, проведения инвентаризации имущества и генерации докладных записок по приему товаров.

4. Разработка архитектурного плана веб-приложений, определение подходящих технологических решений и инструментария программирования.

5. Создание дизайн интерфейса приложения, нацеленного на легкость взаимодействия и производительность.

6. Разработка программного обеспечения, обеспечивающего учет продукции и интеграцию с системой базы данных.

7. Проверка выполненного программного продукта на соответствие установленным функциональным спецификациям и корректность функционирования.

8. Развертывание созданного программного обеспечения для оптимизации работы в торговом пространстве сети «Пятерочка» и тренинг сотрудников для эффективного использования новых функций.

9. Оценка производительности интегрированного приложения с использованием метрик точности.

Необходимость разработать интернет-приложение, оптимизирующее и акселерирующее процесс торговли в сети «Пятерочка», повышая продуктивность персонала и точность инвентаризации продукции.

Веб-аппликации функционируют на стороне сервера и осуществляют обмен данными с пользовательскими устройствами через гипертекстовый передаточный протокол Hyper Text Transfer Protocol (HTTP). Устройствами пользователя, взаимодействующими с таким приложением, могут выступать ПК, мобильные телефоны, планшеты или иные гаджеты, соединенные с Всемирной паутиной.

Ключевые элементы интернет-программы охватывают:

1. Фронтенд, или клиентская часть – это то, что пользователь видит и с чем взаимодействует в своем веб-браузере. Эта часть разрабатывается с использованием технологий веб-разработки, включая HyperText Markup Language (HTML) для структурирования контента, Cascading Style Sheets (CSS) для визуального оформления и JavaScript для добавления интерактивности. Фронтенд отвечает за презентацию информации и управление пользовательскими запросами.

2. Серверный компонент (бэкенд) представляет собой сегмент приложения, который функционирует на стороне сервера, выполняя критические задачи. Эта часть отвечает за обработку входящих запросов из

клиентской секции, управление доступом к базам данных, а также реализацию бизнес-логики программного продукта. Поддержка разнообразия программных языков, включая Python, Ruby, Java, и Hypertext Preprocessor (PHP), позволяет разработчикам выбирать наиболее подходящий инструмент для создания серверной логики.

3. Веб-приложения могут интегрировать системы управления базами данных (СУБД) для организации, хранения и управления данными. Эти базы данных обеспечивают централизованный репозиторий для информации, включая детали пользователей, каталог продукции, записи заказов, и другие сущности, критические для функциональности приложения.

4. Application Programming Interface (API), или интерфейс прикладного программирования, устанавливает стандарты и протоколы, необходимые для взаимодействия различных элементов программного обеспечения. Он обеспечивает взаимосвязь между клиентской и серверной сторонами, позволяя обмениваться данными.

Создание веб-приложений охватывает программирование как фронтенд, так и бэкенд частей, конфигурирование баз данных и deployment приложений на веб-серверах.

Интеграция систем автоматизации для решения задач эффективно воздействует на улучшение механизмов сбыта и расширяет долю рыночных возможностей через активное внедрение и использование настольных персональных компьютеров. Это дает возможность автоматизировать широкий спектр бизнес-операций, охватывая аспекты от маркетинга и продаж до обслуживания клиентов и управления информационными потоками. Применение такой стратегии значительно увеличивает эффективность труда, минимизирует временные затраты и повышает стандарты обслуживания, что, в свою очередь, способствует росту объемов продаж и усилению конкурентных преимуществ на рынке.

Рекомендуется применить Hypertext Preprocessor (PHP), оптимизированный для создания гипертекстовых документов, в связке с

фреймворком Laravel для разрабатывания приложений, целью которых является автоматизация циклов торговли и расширение доли рынка. Hypertext Preprocessor (PHP) – это высокопроизводительный языковой стандарт для веб-разработки, тогда как Laravel предлагает обширный набор средств для эффективного построения веб-приложений, оснащенных продвинутыми возможностями управления данными, системами навигации, проверки подлинности пользователей и др. Применение данных технологий дает возможность разработать стабильное, расширяемое и защищенное решение, способное эффективно способствовать управлению торговыми процессами и обеспечивать эффективное взаимодействие с клиентурой.

Для анализа качества и верификации соответствия выпускаемой продукции первоначальным целям разработки критически важно составить техническое задание. В этом документе должны быть детализированы все аспекты и детали предполагаемого итога, а также спецификации и критерии, которым должен соответствовать конечный продукт.

Техническая спецификация для создания веб-приложения по продаже товаров через интернет-платформу компании:

1. Введение.

1.1 Проект направлен на создание интернет-платформы, предназначенной для коммерческой деятельности организации посредством онлайн-торговли продукцией.

1.2 Анализ – Веб-платформа должна предоставлять возможность администраторам осуществлять просмотр каталога товаров, создавать заказы и мониторить уровень запасов на складе.

2. Функциональные требования.

2.1 Создание учетной записи и верификация личности.

– Участники способны регистрировать профили и осуществлять доступ к платформе через e-mail либо через платформы социальных медиа.

2.2 Просмотр товаров.

– Пользователи имеют возможность просматривать ассортимент товаров, осуществлять их сортировку по различным категориям, производителям и другим характеристикам.

2.3 Формирование заказа.

– Клиенты имеют возможность вносить в корзину артикулы, убирать их и корректировать объем добавленных позиций.

2.4 Управление заказами.

– Администраторы обладают возможностью просмотра и управления заказами, регулируя их статусы.

2.5 Отслеживание заказа.

– Пользователи имеют возможность мониторинга текущего положения своих покупок и получают алерты при любых обновлениях статуса.

3. Требования к безопасности.

3.1 Защита данных.

– Пользовательская конфиденциальная информация должна обрабатываться и транслироваться с использованием шифрования.

3.2 Идентификация и разрешение доступа.

– Доступ к персональной информации и возможностям приложения должен предоставляться только после верификации личности пользователя.

4. Критерии производительности.

4.1 Масштабируемость.

– Программное обеспечение должно обладать возможностью параллельной обработки запросов множества пользователей.

4.2 Быстродействие.

– Открывание страниц и выполнение запросов требуют быстроты и безотлагательности.

5. Требования к технологиям.

5.1 Программирование на языке: Препроцессированный гипертекст Hypertext Preprocessor (PHP).

5.2 Фреймворк: Laravel.

5.3 База данных: MySQL.

5.4 Разработка пользовательского интерфейса: язык гипертекстовой разметки (HTML), каскадные таблицы стилей (CSS), язык программирования JavaScript.

6. Тестирование.

– Требуется выполнить анализ функциональности, защищенности и эффективности программного обеспечения.

7. Документация.

7.1 Требуется разработать технические спецификации, детализирующие архитектурное строение приложения, его возможности и рабочие механизмы.

8. Развертывание и поддержка.

8.1 Требуется выполнить установку приложения на сервер и обеспечить его непрерывную эксплуатацию.

9. Сроки реализации.

9.1 Установить временные рамки для каждого этапа разработки и крайний срок реализации проекта.

1.3 Анализ существующих разработок, выбор и обоснование стратегии автоматизации и способа приобретения информационной системы. Обоснование выбора технологии разработки

«Пятерочка» является ритейлером, фокусирующимся на реализации пищевых продуктов. Ассортимент магазина охватывает все необходимое для полноценного питания: от свежих плодов и овощей, разнообразия мясных и рыбных изделий, до молокосодержащих продуктов, изделий из теста, глубоко замороженных продуктов и предметов бытового назначения. Визуальная

схема, демонстрирующая структурирование процессов в магазине, представлена на рисунке под номером 1.

Рисунок 1 – Организационная схема структурных процессов

Ритейлер «Пятерочка» стратегически размещает свои торговые точки в локациях, максимально удобных для потребителя, обеспечивая тем самым легкий доступ к их услугам. Бренд отличается стремлением предложить товары по конкурентным ценам и проводит разнообразные промоакции, включая скидки на широкий ассортимент продукции. Дополнительно, «Пятерочка» интегрирует в свои магазины элементы современных технологий, предлагая покупателям возможности для самостоятельного выбора товаров и оперативного завершения покупки на кассах самообслуживания, что способствует улучшению их шопинг-опыта. Такая модель работы делает «Пятерочку» предпочтительным выбором для ежедневных покупок благодаря сочетанию доступности, удобства и эффективности. В документе, обозначенном как рисунок 2, представлена диаграмма, иллюстрирующая структуру корпоративной информационной системы сети.

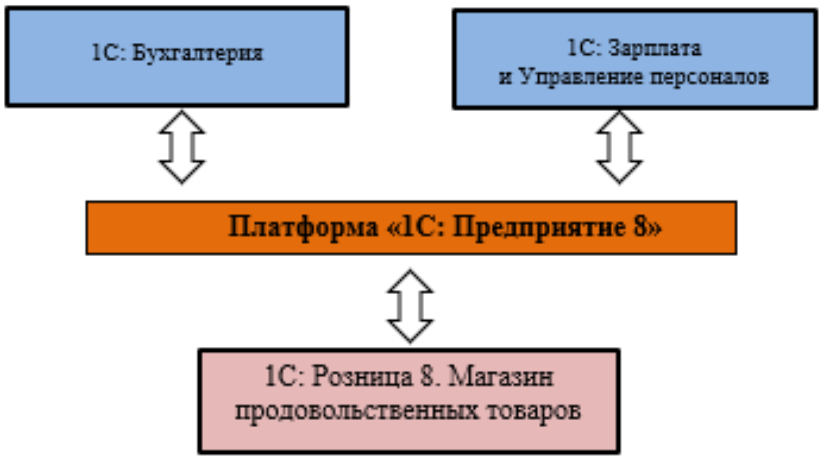

Рисунок 2 – представляет собой схему корпоративной информационной системы ритейлера.

Корпоративная ИТ-система реализована на базе платформы «1С: Предприятие 8» и включает в себя множество модулей.

1. Программный модуль «1С: Зарплата и управление персоналом» является частью информационной системы. Он обеспечивает выполнение функций по управлению кадрами в ритейле.

2. Элемент программного обеспечения (ПО) «1С: Бухгалтерия 8» предназначен для ведения бухгалтерского учета в компании.

3. Программный модуль «1С: Розница 8. Магазин продовольственных товаров» является частью программного обеспечения для автоматизации продаж. Эта система предназначена для эффективного управления розничными операциями в продовольственном магазине.

Для оптимизации процессов управления и повышения эффективности продаж в сети магазинов «Пятерочка», целесообразно разработать веб-приложение. Это позволит избежать необходимости загрузки и установки отдельных программных решений, таких как «1С: Предприятие 8». В рамках выполнения дипломной работы будет реализовано данное веб-приложение, которое будет хоститься на централизованном сервере и доступ к которому будет осуществляться через стандартные веб-браузеры.

1.4 Определение обоснования разработанных стратегий

Техническая поддержка объединяет в себе все оборудование и программное обеспечение, необходимые для работы информационных систем и IT-инфраструктуры организации.

Проектируемая информационно-вычислительная система предназначена для прямой эксплуатации на серверной машине, которая в своем арсенале должна обладать, по меньшей мере, процессором от Intel Pentium или от Advanced Micro Devices (AMD), памятью типа оперативного записывающего устройства (ОЗУ) размером минимум в 2024 мб и накопителем с доступным объемом не менее 500 мб. Критически важным требованием к архитектуре системы является географическое размещение ее центра обработки данных (ЦОД) в пределах той же страны, что и основная аудитория пользователей, чтобы обеспечить высокий уровень защиты данных и оптимизировать скорость обработки запросов [12].

Информационное программное обеспечение (ПО) представляет собой комплекс подходов, утилит и техник, применяемых для аккумуляции, обработки, сохранения, распространения и применения данных.

Информационная защита – это защищенность данных о продукции, объемах, разнообразии и наличии в информационной системе.

При выполнении задачи требуется создать набор документации, охватывающий:

- техническое задание;
- программу и методологию тестирования;
- описание применения;
- описание программы.

Предполагается, что проектируемое программное обеспечение будет оснащено функцией генерации оповещений об ошибке при некорректном вводе данных, а также обладать способностью вести диалог с пользователем в рамках предложенного функционала. Важно, чтобы программа надежно

реагировала на любые изменения или удаление информации, обеспечивая тем самым неразрывность и непротиворечивость данных. В ситуациях, когда действия пользователя не соответствуют установленным требованиям, система должна инициировать выдачу точных предупреждений. В случае необходимости изменений в коде, обратная связь с разработчиками становится обязательной, подразумевая ограниченный доступ конечных пользователей к программным файлам системы, предотвращая тем самым несанкционированное редактирование или просмотр исходного кода.

Разработка технических решений, согласованных с создаваемым программным обеспечением (ПО), подразумевает разработку алгоритмов и структур данных, которые персональный компьютер (ПК) способен обрабатывать для решения специфических задач. В состав программного обеспечения (ПО) входят операционные системы (ОС), приложения для работы и творчества (такие как текстовые процессоры, инструменты для создания графики, веб-обозреватели) и также служебное программное обеспечение (к примеру, интерфейсы для управления аппаратной частью, support libraries и др.).

MySQL является реляционной системой управления базами данных (РСУБД), широко применяемой в сфере веб-разработки для эффективного хранения и администрирования данных. Эта платформа обеспечивает разработчиков необходимым функционалом для проектирования, обслуживания и модификации баз данных, а также позволяет выполнять сложные запросы к сохраненной информации.

Ряд факторов объясняют, почему база данных MySQL обладает высокой популярностью в области управления данными:

1. Open Source – MySQL принадлежит к категории открытого программного продукта, допускающего свободное использование, модификацию и распространение без взимания платы.

2. Эффективность обработки данных – MySQL зарекомендовал себя как система управления базами данных с высокими показателями производительности, особенно при работе с объемными наборами данных.

3. MySQL, благодаря своей архитектуре, гарантирует безопасное сохранение информации и располагает функциями, направленными на поддержание целостности данных.

4. Масштабируемость в MySQL обеспечивает гибкость адаптации под данные разной величины, поддерживая работу как с компактными, так и с объемными базами данных, удовлетворяя потребности от малых проектов до разветвленных крупномасштабных систем.

5. Соответствие MySQL стандартам Structured Query Language (SQL) обеспечивает его взаимодействие с различными реляционными базами данных, благодаря единому использованию SQL-норм и правил.

MySQL активно применяется в сфере веб-разработки для управления данными, включая информацию о пользователях, содержание сайтов, финансовые операции и прочее. Эта СУБД находит применение во множестве веб-проектов, начиная от личных блогов и заканчивая крупными онлайн-платформами.

Выбранная система управления базами данных удовлетворяет всем необходимым критериям, что обусловило решение остановить свой выбор на данной системе программирования.

Эффективное управление веб-сервером Apache HTTP критично. Apache HTTP Server представляет собой ведущий веб-сервер, занимая значительные позиции среди аналогов. Его задачей является проведение кооперативного анализа для надлежащей обработки веб-страниц, файлов, а также другой контент, доступный через протокол Hyper Text Transfer Protocol (HTTP).

Некоторые преимущества Apache включают:

1. Бесплатное и открытое программное обеспечение (ПО): Apache – это свободно доступное ПО с открытым кодом, что подразумевает бесплатный доступ к исходному коду для его просмотра, модификации и дистрибуции.

2. Совместимость с многочисленными операционными системами (ОС): Apache обладает широким спектром поддержки операционных систем, включая Linux, Windows, macOS и прочие, предоставляя тем самым эластичные возможности для внедрения на разнообразных платформах.

3. Масштабируемость и адаптивность: Apache обеспечивает обширную гамму конфигураций и расширений, способствующих их тонкой регулировке под уникальные требования вашего пространства. Эта платформа также выдерживает интенсивную нагрузку, предоставляя значительные возможности для масштабирования.

4. Активное и многочисленное сообщество: Apache выделяется на фоне благодаря широкой базе пользователей и разработчиков, оказывающих поддержку через множество доступных ресурсов, подробную документацию и обширную поддержку.

5. Стабильность и защита: Apache зарекомендовал себя как стабильное и защищенное решение. В его арсенале присутствует обширный набор инструментов для предотвращения атак и гарантии безопасной работы веб-приложений.

Информационно-технологическое оснащение охватывает широкий спектр аппаратных и программных элементов, критически важных для функционирования информационных систем в пределах предприятия или конкретного проекта. В его состав входят вычислительные устройства, серверные мощности, элементы сетевой инфраструктуры, программные платформы, системы управления базами данных, операционные системы, различные прикладные решения и дополнительные технологические инструменты, задействованные для создания и поддержки информационных систем.

Технологическое программное обеспечение (ПО) охватывает широкий спектр аппаратуры и программ, критичных для специализированных операций, включая веб-сервера, базы данных, облачную инфраструктуру, защиту данных, аналитические инструменты, среди прочего.

Архитектура веб-сайта должна строиться вокруг модели MVC (Model-View-Controller).

Model-View-Controller (MVC) – архитектурный паттерн в программировании для веб-платформ, применимый в том числе к Hypertext Preprocessor (PHP). Этот подход декомпозирует приложение на три ключевых составляющих:

1. Модель: отвечает за взаимодействие с данными приложения, включая базы данных, файлы и различные источники информации. Она управляет бизнес-логикой, доступом к данным и их обновлением.

2. Вид: эта компонента отвечает за визуализацию данных для пользователя и реализует отображение информации. В его задачи не входит обработка логики; он лишь визуализирует данные, получаемые из модели.

3. Контроллер: анализирует запросы от пользователей, коммуницирует с моделью для извлечения информации и направляет данные во View для визуализации. Он также регулирует процесс работы приложения, определяя действия по полученным данным.

Применение архитектурного шаблона Model-View-Controller (MVC) предлагает ряд выгод:

1. Принцип разделения ответственностей: способствует четкому разграничению бизнес-логики, пользовательского интерфейса и управления данными. Такой подход улучшает читаемость, логичность и обслуживаемость кода.

2. Повышенная масштабируемость: декомпозиция системы упрощает интеграцию новых возможностей или модификацию имеющихся, минимизируя потребность в полной ревизии кода.

3. Переиспользование кода улучшает разработку: модули кода вновь используются в разных секторах приложения, способствуя оптимизации процесса разработки.

4. Модульное тестирование: позволяет изолированно проверять функциональность каждой части приложения (модели данных,

пользовательского интерфейса, бизнес-логики), облегчая и оптимизируя процедуру верификации программного обеспечения.

5. Усиленная защита обеспечивает поддержку разработчиков в осуществлении ключевых мер безопасности, включая защиту от поперечного сайтового запроса (CSRF) и защиту от межсайтового скриптинга (XSS), через строгое разграничение между логической частью и пользовательским интерфейсом.

Программное обеспечение необходимо создать, применяя инструментарий PHP как основной язык программирования и базы данных MySQL в качестве основы для хранения и управления данными.

Программное обеспечение должно функционировать на платформах, использующих Linux Kernel, включая все операционные системы (ОС) этого класса, построенные на одинаковой основе. Linux не представляет собой монолитную ОС подобно Windows или macOS. Разнообразие дистрибутивов Linux обусловлено специализацией каждого из них на выполнение уникальных функций.

Для целей моделирования операционных процедур предприятия выбрали веб-платформу DGRM. Указанная возможность выполнена через инструментарий Case.

DGRM – это редактор диаграмм, цель которого – превратить его в карту знаний. DGRM можно использовать для графического представления бизнес-процессов. Графическое представление схемы работы, обмена информацией, в документе показана бизнес-модель. Отличительные особенности: аскетичный, работает на телефонах (одно из немногих веб-решений), с открытым исходным кодом. Это также помогает четко документировать основные аспекты на каждом этапе бизнес-процессов, которые необходимо выполнить, как они контролируются и внедряются, необходимые ресурсы и, наконец, позволяет просматривать результаты этих действий.

2. ПРОЕКТНЫЙ РАЗДЕЛ

2.1 Информационное обеспечение задачи (комплекса задач, автоматизированное рабочее место)

Для успешного управления процессом распространения товаров посредством цифровой платформы компании требуется овладевать множественными данными. Вот перечень необходимых сведений:

1. Информация о товарах:

– наименование товара;

– описание товара;

– артикул или индивидуальный номер продукции;

– цена товара;

– количество товара на складе;

– товарная категория или группы категорий, к которым принадлежит продукт;

– изображения товара;

– спецификации продукта;

– срок использования (при наличии применимости);

– доступность для заказа.

2. Информация о поставщиках:

– название компании-поставщика;

– данные для связи с поставщиком (номер телефона, адрес электронной почты, местоположение);

– условия доставки (минимальный объем заказа, временные рамки доставки и прочее);

– соглашения о стоимости и предоставлении уценок.

3. Информация о заказах:

– номер заказа;

– дата оформления заказа;

– фаза исполнения заказа (обработка, завершен, доставлен и прочее);

– список товаров в заказе;

– сумма заказа;

– информация о транспортировке (местоположение, метод отправки).

4. Информация о пользователях:

– логин и пароль пользователя;

– функции участника в архитектуре программы (админ, руководитель проектов, заказчик и т.д.);

– личные сведения получателя (место для отправки, контактный номер, электронная почта);

– история заказов пользователя;

– пользовательские предпозиции (при соответствии).

5. Информация о платежах:

– способы оплаты;

– данные о выполненных транзакциях;

– статус оплаты заказа.

6. Информация о складе:

– инвентаризационная информация о запасах продукции в хранилище;

– данные о логистике товаров внутри складских помещений;

– ведение записей о приемке и отправке продукции.

Это комплексный перечень данных, необходимых для эффективного продвижения товаров через веб-платформу компании.

Использование баз данных необходимо для эффективного организации и систематизации данных. Облегчение процесса понимания ключевых аспектов, их взаимосвязей внутри информационного массива, становится первоочередной задачей систематизации данных. Это, в свою очередь, способствует более легкому запоминанию данных, формированию ассоциативных связей и эффективному использованию мнемонических

приемов. Для достижения поставленных целей в процессе структурирования информации применяются два основных методологических принципа.

Система управления базами данных (СУБД) представляет собой программное обеспечение, задачей которого является организация, сохранение и обработка данных. Она служит фундаментом для многообразия цифровых платформ, например, интернет-банкинга, социальных медиа, поисковиков и интернет-магазинов.

Процесс перевода входящих данных в формат, понятный компьютеру, может осуществляться различными методами, включая использование цифровых форм или сканирование штрих-кодов. Когда речь идет о продукции, маркированной штрих-кодами, оптимальным решением становится применение специализированных сканеров, которые можно настроить для генерации нужного формата данных. В случаях, когда продукция, такая как фрукты или овощи, не имеет штрих-кода, целесообразно использовать электронные формы ввода данных. Важно тщательно проанализировать каждый из этих методов для обеспечения эффективности обработки информации.

Форма служит ключевым механизмом интерактивности, позволяя приложению, работающему в веб-пространстве, передавать специфичные данные на сервер или другой endpoint для последующего анализа или хранения. Отправка осуществляется методом передачи упорядоченных данных, где каждый элемент представлен в формате ключ (имя переменной) и значение. Важно использовать английский алфавит для обозначения ключей, что обеспечивает их корректное восприятие и обработку. Полученные данные обрабатываются на стороне сервера как текстовые строки, даже если передаваемое значение является цифровым, требуя соответствующего преобразования для дальнейших операций.

Сканер штрих-кода представляет собой технологически продвинутое устройство, задачей которого является декодирование информации, заключенной в виде штрих-кода. Оно способно эффективно распознавать

символы, приложенные к различным материалам, включая бумагу, картон, пластик, и даже металл. В коммерческом и складском оборудовании сканеры штрих-кодов играют ключевую роль в ускорении процесса сбора и анализа данных, подавая себя в качестве более надежной альтернативы ручному вводу. Этот инструмент стал незаменимым атрибутом во многих отраслях, от розничной торговли до логистики, благодаря своей способности предоставлять точную информацию о товарах в реальном времени. Основная цель его использования – это автоматизация обработки данных продукции, оснащенной штриховой маркировкой, что находит широкое применение не только на продажных точках, но и в услугах, складах, а также во множестве других бизнес-операций.

После импорта данных через сканирование или форму ввода, эти данные доступны для последующей обработки или представления пользователям. Это может включать интеграцию списка продуктов в базу данных, где можно производить манипуляции, такие как корректировка запасов, обновление информации о товарах и др.

Для сбора исходных данных необходимо иметь каталоги продукции, информацию о складах и сведения о пользователях. Каталог продукции может быть сформирован через процесс сканирования ассортимента или его предварительного составления отделом закупок. В этих списках отражена информация о наименовании продукта, марке, категориях, идентификаторе продукта (артикуле), наличии на складе, закупочной и продажной ценах, а также цене со скидкой, если таковая предусмотрена. Инвентарь складов включает в себя наименование склада, его вместимость в единицах продукции и местоположение. Вся эта информация может быть предоставлена руководителем производственной практики от соответствующей организации.

Поскольку разрабатывается веб-приложение, все результаты его работы представляются в формате HyperText Markup Language (HTML), то есть в структурированном виде HyperText Markup Language. Это означает, что, к примеру, после внесения информации в базу данных, доступ к ним можно

получить через специальный интерфейс в веб-браузере. В случае запроса, например, списка продуктов, происходит обращение к базе данных, на основании которого формируется ответ. Этот ответ генерируется поэтапно, а не одним полным списком, чтобы избежать чрезмерной нагрузки на систему, которая может привести к сбою или замедлению ее работы. Для оптимизации вывода больших объемов данных используется метод пагинации. Пагинация представляет собой метод организации большого количества информации путем ее разбиения на множество страниц, что позволяет пользователю постепенно просматривать данные, переходя от страницы к странице.

На рисунке 3 представлена модель данных.

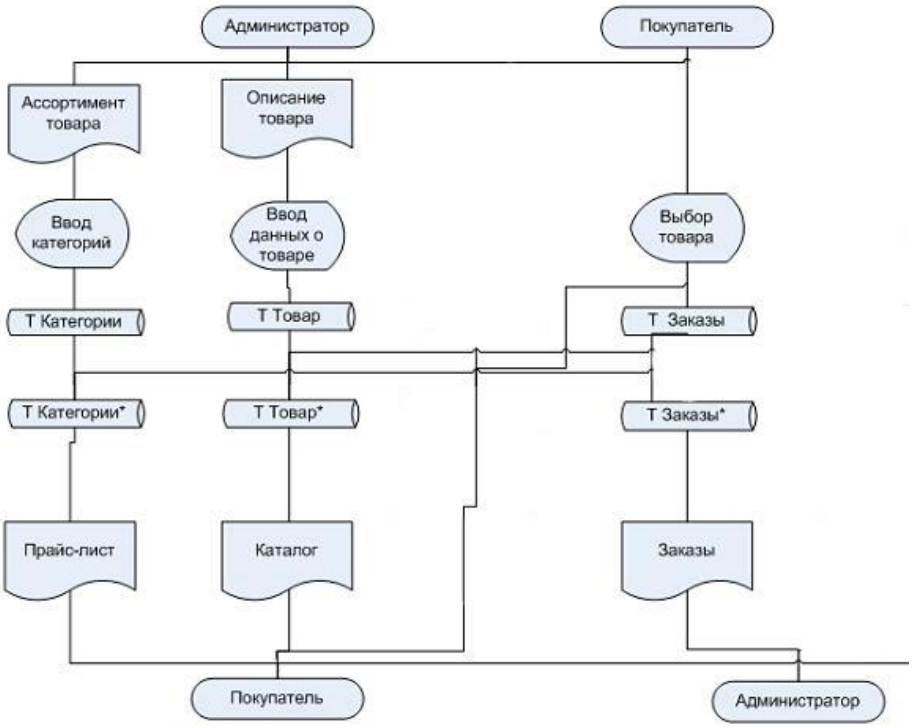

Рисунок 3 – Модель данных

На рисунке 4 представлена диаграмма потоков данных в соответствии с методологией Гейна/Сарсона, отражающая структурно-функциональную модель.

Рисунок 4 – Демонстрация структурно-функциональной схемы

В процессе получения и обработки данных критично предоставлять их в сжатом и доступном виде, ассоциировав определенные символьные идентификаторы с конкретными элементами, то есть осуществлять их кодирование. Под кодированием понимается назначение элементу уникального символьного представления.

Кодирование данных необходимо из-за:

– с высокой плотностью буквенных данных;

– большими объемами данных;

– с ростом объемов данных, требующих передачи через коммуникационные каналы;

– доминированием логических процедур в ходе информационной обработки.

Кодирование позволяет эффективнее идентифицировать и мониторить атрибуты объекта, облегчает и ускоряет внесение данных в любую систему хранения, обеспечивая обширные возможности для ссылок. Применение кодов значительно упрощает категоризацию данных.

Кодирование в экономической информатике обозначает набор принципов для преобразования экономических данных в стандартизированные коды. В этом процессе применяются различные методики, включая порядковое кодирование, последовательное, позиционное, циклическое и смешанное кодирование элементов информации.

В методике последовательного кодирования каждый элемент заданного множества получает уникальный порядковый номер без оставления пустых интервалов, позволяя эффективно использовать возможности кода и минимизировать его размер. Однако, это не предоставляет простор для внесения новых элементов в систему.

Система линейной классификации применяется для организации элементов, относящихся к неперекрывающимся категориям с однородной структурой, например, основываясь на двух критериях.

Битовое кодирование применяется для представления сложных структур данных. Каждый элемент внутри закодированной группы относится к категориям на основании уникальных атрибутов, а им назначается число бит, соответствующее объему элементов в этой категории.

Методика повторного использования служит для присваивания уникальных идентификаторов различным единицам. В контексте позиционного кодирования, применяются как числовые, так и алфавитно-цифровые обозначения, которые напрямую отражают сущность данного объекта.

Интегрированная система учета обеспечивает строгое разделение атрибутов товаров и услуг. Одновременно позволяет кодирование каждого атрибута по различным методикам: либо используя последовательное, либо позиционное кодирование. Ее гибкость делает интегрированную систему идеально подходящей для применения в сфере экономики, гарантируя автоматическую обработку и анализ данных согласно определенным параметрам товаров и услуг.

Начальные три знака штрих-кода идентифицируют страну-производитель изделия в соответствии с международной системой GS1. Система GS1 на данный момент объединяет 108 государств-участников. К примеру, продукция, произведенная в Германии, маркируется номерами от 460 до 469.

Четвертая по счету группа из четырех цифр в коде обозначает уникальный идентификатор производителя, который присваивается локальными отделениями международной сети GS1. В пределах России эту функцию выполняет организация под наименованием «Национальная организация GS1 Россия».

Эти пять знаков представляют собой уникальный идентификатор продукции, который назначается ее производителем. В процессе присвоения данного идентификатора конкретному товару нет унифицированного подхода; компания-производитель сама задает критерии уникальности кода для определенного изделия или серии продукции, часто опираясь на такие данные, как серийный номер, дата производства, номер партии и прочее.

Последняя цифра кода называется контрольным числом.

Например, для кода 4607051690950 можно дать следующую характеристику:

460 – код России;

7051 – код предприятия;

69095 – код товара;

0 – контрольное число.

Контрольное число определяется согласно специфическому алгоритму, представленному в таблице 1.

Таблица 1 – Методика шифрования

Код	Страна	Код	Страна
000 – 019	Иордания	621	Сирия
020 – 029	США (ограниченное применение, внутренняя нумерация)	300 – 379	Франция
030 – 039	США	624	США
040 – 049, 050 – 059	США (ограниченное применение, внутренняя нумерация)	625	Ливия
300 – 379	Египет	627	Кувейт
380	Болгария	628	Саудовская Аравия
383	Китай	629	Эмираты
385	Хорватия	640 – 649	Финляндия
387	Босния и Герцеговина	383	Словения
389	Норвегия	389	Кыргызстан
400 – 440	Германия	729	Россия
450 – 459 490 – 499	Мексика	460 – 469	Израиль
460 – 469	Швеция	740	Гватемала
470	Кыргызстан Черногория	741	Сальвадор
471	Тайвань	742	Гондурас
474	Панама	743	Никарагуа
746	Республика Доминикана	744	Коста-Рика
476	Азербайджан	474	Эстония
477	Литва	475	Латвия

Для расчета контрольной цифры штрих-кода EAN-13 применяется специфический алгоритм.

В заголовке таблицы (табл. 2) отмечают уникальные номера 13-ти разрядов бинарного кода, располагая их с правой стороны на левую. В

следующей строке демонстрируется заполнение таблицы конкретным примером кода, таким как 4018993404787, детализация которого представлена в таблице 2.

Таблица 2 – Информация для расчета проверочного числа шифра

Позиция цифры в коде (идет справа налево)	13	12	11	10	9	8	7	6	5	4	3	2	1
Значение кода	**4**	**0**	**1**	**8**	**9**	**9**	**3**	**4**	**0**	**4**	**7**	**8**	7
Четные значения кода, их сумма		0	+	8	+	9	+	4	+	4	+	8	=33
Необходимо объяснить, о чем идет речь в запросе пользователя?	4	+	1	+	9	+	3	+	0	+	7		=24

В третьей рядке таблицы фиксируются исходные коды, идущие далее в операцию сложения, итоговый результат которой утраивается, к примеру:

(0+8+9+4+4+8) 3=99.

В четвертом ряду таблицы записывают и складывают значения кода с нечетными индексами, не учитывая ведущий бит, предназначенный для расчета, как показано в примере.

4+1+9+3+0+7=24.

Общая сумма, полученная утроением суммы четных позиций и их последующим сложением с суммой значений нечетных позиций в коде, например:

99+24=123.

Выполненное округление числа происходит с прибавлением до ближайшего значения, делящегося на 10 без остатка, к примеру.

123 Округляется до 130.

Из приближенного числа вычитают исходное, неокругленное значение, и полученный результат служит проверочным числом.

130-123=7.

Документация вводных данных и нормативно-справочная информация организуются в список товаров, перечень поставщиков и каталог складских учреждений. Важно провести тщательный анализ каждого индивидуального документа, где каждый из них представляет собой Excel файл, включающий в себя табличные данные.

Товары – документ включает информацию о разнообразии предлагаемых товаров, охватывает данные:

1. Наименование. Установленное наименование продукции, включая его особенности.
2. Артикул.
3. Вес. Вес товара в граммах.
4. В наличии кол-во штук.
5. Информативный обзор. Комплексная характеристика продукта при его доступности.
6. Склад. Название склада.
7. Производители, наименование для заключения договоров.

На рисунке 5 представлена таблица под наименованием «Товары».

Артикул	Вес	В наличии, шт	Название	Информация	Склад	Поставщик
1060509	170 г	300	Сметана Пестравка		Склад РЦ ЗТЛ	ОАО «Маслозавод Пестравский»
1080666	900 г	500	Торт Шоколадный Mirel		Склад Воронеж-Рамонь	ООО «Хлебпром»
1520565	5 л	100	Вода минеральная Заповедный источник негазированная		Склад РЦ Казань	Вода красноглинская, «Лагуна»
2560589	200 г	150	Колбаса Миланская Дубки		Склад Воронеж-Рамонь	«Дубки»
1090333	300 г	230	Хлеб Новокуйбышевскхлеб Оригинальный ржано-пшеничный нарезка		Склад РЦ ЗТЛ	Новокуйбышевский хлеб
1020444	200 г	350	Белуга холодного копчения ломтики		Склад Воронеж-Рамонь	ООО «Рыбная база»
1070525	900 мл	78	МОЛОКО ПИТЬЕВОЕ ПАСТЕРИЗОВАННОЕ МАССОВАЯ ДОЛЯ ЖИРА 2,5%		Склад РЦ Новая Рига	ООО "ЛебедяньМолоко"
1450365	90 г	250	Брикет ГОСТ малый пломбир ваниль		Склад Воронеж-Рамонь	ООО «Северная земля»
3650041	300 г	410	ФРИКАДЕЛЬКИ ИЗ ГОВЯЖЬЕГО ФАРША 7,5%		Склад РЦ Вятка	«Самсон»
1240358		500	Фрикадельки из говяжьего фарша 7,5%, 300 г		Склад Воронеж-Рамонь	ИП «Игнатьев»

Рисунки 5 – Таблица «Товары»

Далее представлена информация о складских помещениях, включая ограниченный объем сведений, такие как наименование и местоположение склада, а также его вместимость в единицах хранения (рис. 6).

Склад	Адрес	Кол-во штук на складе
Склад РЦ ЗТЛ	г.Санкт-Петербург, ул. Седова, д.9/4	2500
Склад Воронеж-Рамонь	Воронежская область, село Айдарово, ул. Промышленная, д.1	25682
Склад РЦ Казань	Казань, ул. Владимира Кулагина, 23	125765
Склад РЦ Вятка	г. Киров, ул. Опарина, 5А, помещ. 1008	86538
Склад РЦ Новая Рига	Московская область, Новорижское шоссе, с1	86538

Рисунок 6 – Таблица «Склад»

И завершающим документом из перечня справочной документации становятся сведения о поставщиках. В данном документе указаны юридическое наименование и адрес регистрации компании или индивидуального предпринимателя (рис. 7).

Юридическое имя поставщика	Юридический адрес
ОАО «Маслозавод Пестравский»	446160, Самарская обл., с. Пестравка, ул. Степная, д. 7
ООО «Хлебпром»	54038, Челябинская обл., Челябинский г.о., Металлургический вн.р-н, г. Челябинск, ул. Хлебозаводская, Д. 20
Вода красноглинская, «Лагуна»	443112, Самарская обл., г. Самара, ул.Ветвистая, 20
«Дубки»	410530, Саратовская Область, г. Саратов, п. Дубки
Новокуйбышевский хлеб	446205, Самарская Область, г.о. Новокуйбышевск, г Новокуйбышевск, ул Суворова, д. 4
ООО «Рыбная база»	443022, Самарская Область, г. Самара, ш. Заводское, б/н, литера л
ООО "ЛебедяьМолоко"	399610, Липецкая Область, м.р-н Лебедянский, г.п. Город Лебедянь, г Лебедянь, ул Южная, д. 6
ООО «Северная земля»	443022, Самарская Область, г.о. Самара, вн.р-н Советский, г Самара, проезд Мальцева, д. 9
«Самсон»	115470, Москва, Кленовый бульвар, д.7 корп.2
ИП «Игнатьев»	398059, г. Липецк, ул. Неделина, д. 4, корп А,

Рисунок 7 – Перечень, содержащий информацию об поставках

Дизайн изделия включает множество элементов:

Публикация. Арифметическое пространство и приглашение к действию в строке для данных.

Масса. Числовой параметр, отражающий массу одного экземпляра товара, включающий цифровое значение.

Заголовок. Размер поля для ввода наименования ограничен 255 символами, ввиду того, что тип данных varchar допускает хранение не более 255 символов.

Данные. Текстовая область предназначенная для размещения основных данных о товаре может оказаться не заполненной, так как не для всех изделий существует текстовое описание.

Складское хозяйство. Опцион выбора, где представлен перечень доступных запасов, позволяющий определить единственный склад из предложенного списка, откуда происходит поступление товара.

Поставщики. Это поле предлагает список зарегистрированных поставщиков товаров, позволяя пользователю выбрать из него единственного поставщика, от которого произведен данный продукт.

На рисунке 8 представлена форма товаров.

Рисунок 8 – Форма для товаров

Документ поставщика включает несколько разделов:

1. Официальное наименование. Поле ввода для указания официального наименования с контекстной подсказкой внутри поля ввода.

2. Юридический адрес: поле для заполнения юридического адреса организации, включая примечания, если таковые имеются.

На рисунке 9 описана формализация для поставщиков.

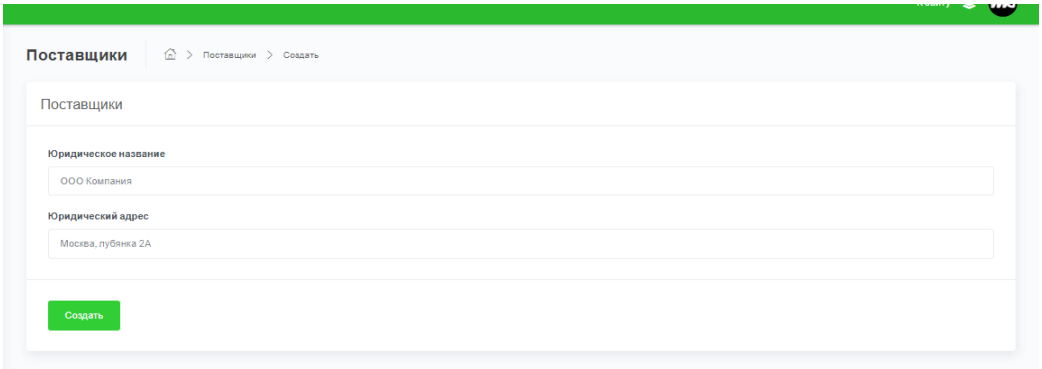

Рисунок 9 – Анкета для поставщиков товаров

Для инвентаризации складских помещений используется документ, включающий несколько разделов:

1. Наименование. Поле ввода для указания имени хранилища, без вспомогательных указаний.

2. Локация склада. Поле ввода для указания местоположения склада без вспомогательных указаний.

Демонстрация в интерфейсе находится на рисунке 10.

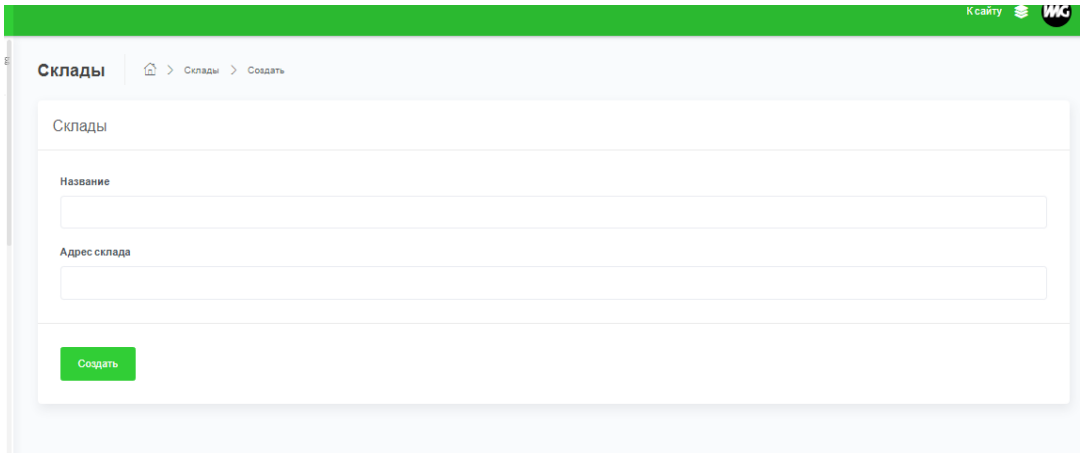

Рисунок 10 – Форма для складов

Концептуальная модель данных предназначена для отображения комплексной структуры данных, ключевой для оптимизации процессов ведения товарооборота, администрирования заказов, учета поставщиков, клиентов и взаимодействия с другими бизнес-процессами. Описание особенностей концептуальной модели.

Сущности:

1. Товары:

– наименование;

– категория;

– описание;

– идентификатор товара;

– цена;

– артикул.

2. Поставщики:

– идентификатор поставщика;
– юридический адрес;
– название компании.

3. Заказы:

– идентификатор заказа;
– статус (обрабатывается, завершен и т.п.);
– сумма;
– дата оформления;
– данные о доставке.

4. Пользователи:

– идентификатор пользователя;
– роль в системе;
– контактные данные;
– логин и пароль.

5. Платежи:

– данные об осуществленных транзакциях;
– способы оплаты;
– данные о выполненных транзакциях;
– статус оплаты заказа.

6. Склад:

– адрес склада;
– название склада.

Связи между сущностями:

1. Транзакции ассоциируются с продуктами и клиентами.
2. Продукция ассоциируется с поставщиками и хранилищами.
3. Транзакции ассоциируются с заказами и клиентами.

Описание поля таблицы products с идентификатором id: применяется тип данных integer с максимальной длиной значений в 11 символов и особенностью автоматического увеличения значения.

Поле с именем «name» является строкового типа varchar, с ограничением длины в 255 символов.

weight – Целочисленный тип данных, максимальная длина: 11 символов.

Информация – Это текстовое поле, в котором не установлено ограничение по количеству символов. Дата создания – Поле типа Timestamp, используемое для хранения временных отметок.

sklad_id – Целочисленный тип поля (integer) с максимальной длиной значения в 11 символов, служит первичным ключом в таблице providers.

updated_at – время последнего обновления, формат – Timestamp.

Таблица products представлена на рисунке 11.

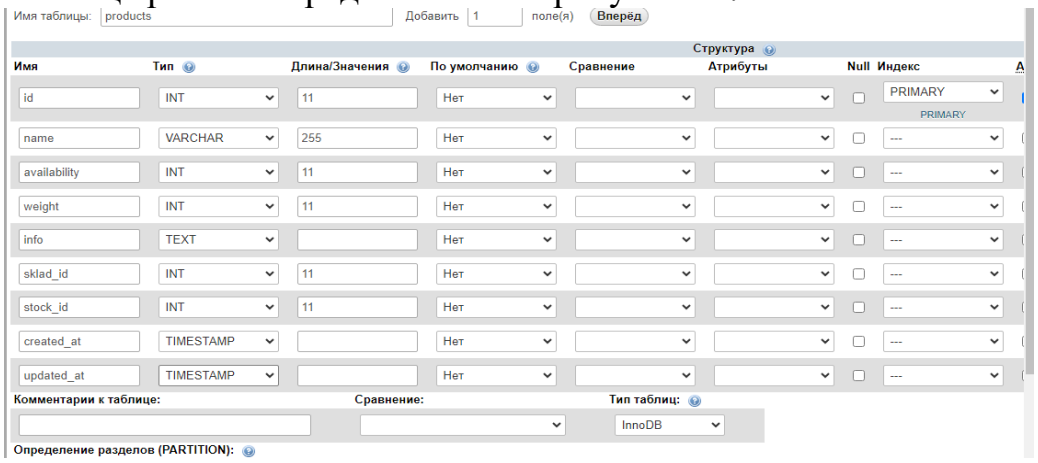

Рисунок 11 – Таблица products

Таблица акций «stocks» имеет следующую структуру: идентификатор «id» – это поле целочисленного типа (integer) с максимальной длиной значения в 11 символов, оснащено функцией автоматического увеличения.

адрес – тип колонки varchar, верхний предел 255 символов.

Поле name является строковым типом данных (varchar) с максимальной длиной в 255 символов.

created_at – Момент создания, поле с типом данных Timestamp.

updated_at – метка времени, поле типа Timestamp.

На рисунке 11 демонстрируется таблица с акциями.

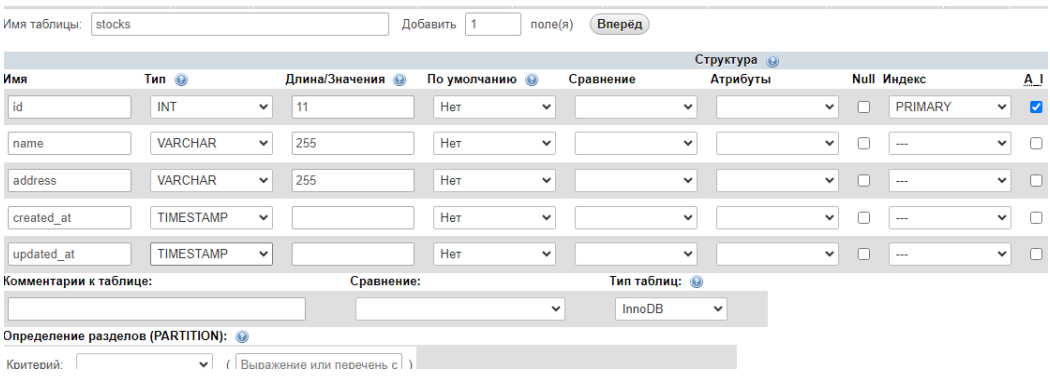

Рисунок 11 – Таблица stocks

В таблице «providers», поле «id» характеризует собой целочисленный тип данных (integer) с максимальным количеством цифр равным 11, и функцией автоматического увеличения значения.

name – это поле с типом данных varchar, допускающее максимум 255 символов.

поле с названием address представлено в виде типа varchar, допуская максимальное количество символов до 255.

created_at – это временной отпечаток, представлен в формате Timestamp.

updated_at – Timestamp-поле, представляющее временную метку.

На рисунке 12 представлена таблица providers.

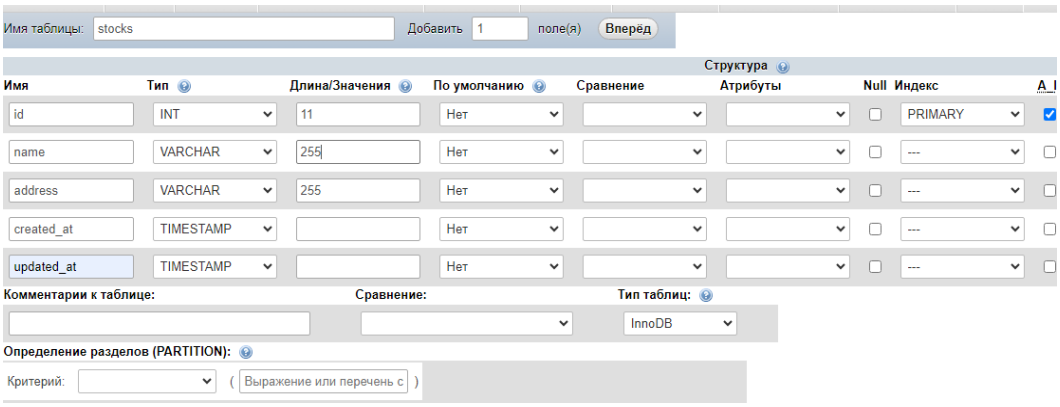

Рисунок 12 – Таблица providers

Таблица «orders» содержит столбец «id»: тип данных – целочисленный (integer), максимальная длина значения – 11 символов, с автоматическим приростом.

Status – это строковый тип данных, допускающий максимум 255 символов.

Поле Sum – это десятичный тип, с максимумом восемь цифр, имеющее ограничение в два знака после десятичной точки.

Delivery – это тип данных поля varchar с максимальной длиной 255 символов.

created_at – временной отметкой, тип данных Timestamp.

updated_at – Timestamp тип данного, который отображает временную отметку.

Иллюстрация таблицы orders представлена на рисунке 13.

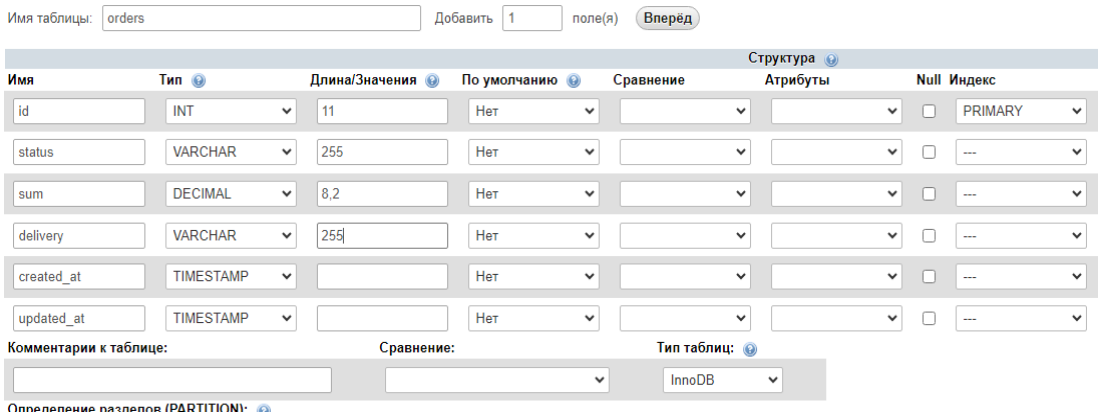

Рисунок 13 – Таблицы orders

В таблице payments поле id определяется как целочисленное (integer) с максимальной длиной записи 11 символов и автоматическим приращением (auto_increment).

Метод – это столбец типа varchar; предельная длина – 255 символов.

Поле с названием info – это строковый тип данных (varchar) с максимальной длиной в 255 символов.

status – тип поля Boolean.

created_at – временной индекс, поле типа Timestamp.

updated_at – временная отметка, принадлежащая к типу данных Timestamp.

На рисунке 14 представлена таблица payments.

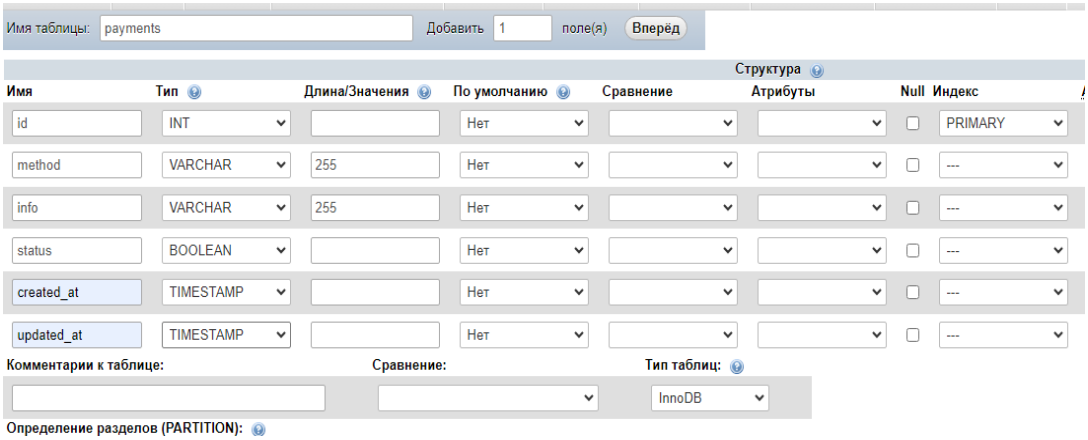

Рисунок 14 – Таблица payments

Взаимосвязи таблицы products демонстрируются на рисунке 15.

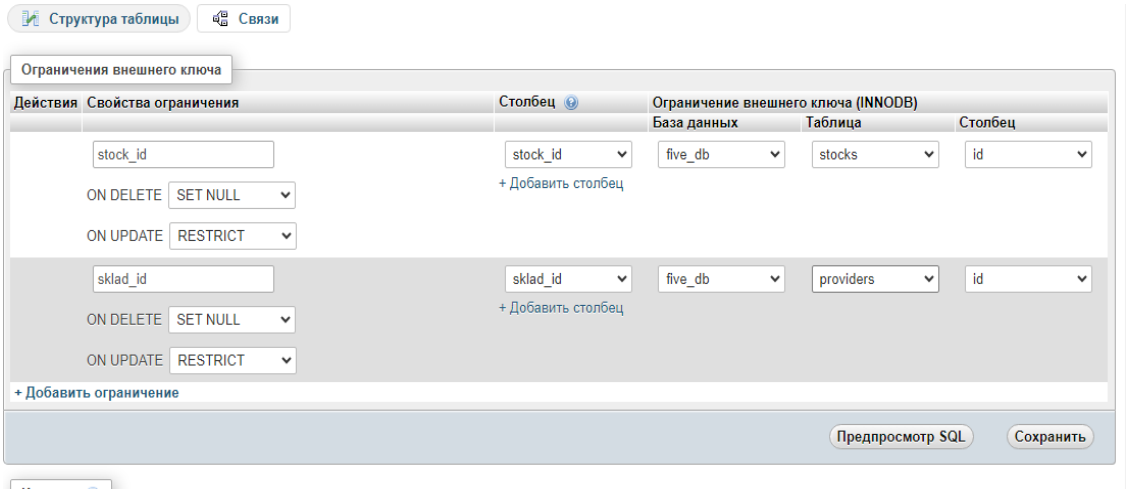

Рисунок 15 – Взаимосвязи таблицы товаров

Форма представляет собой элемент интерфейса, применяемый главным образом для введения информации, ее визуализации или контроля функционирования программы. Эти элементы могут служить для графического демонстрирования информации в таблицах или коллекциях данных, полученных в результате выполнения запросов.

Использование форм значительно облегчает процесс редактирования, ввода и удаления данных. Она автоматизирует выборку данных из связанных источников, выполнение вычислений для выводимой информации, а также скрытие или показ определенных элементов в зависимости от значений других полей в данных или настройках, заданных пользователем.

Формы служат инструментом для занесения информационных данных в базы данных. Эти формы обеспечивают простоту и эффективность ввода и модификации данных. Скрипты, выполняющие обработку данных форм, находятся в приложениях А, Б, В, Г, Д, Е, Ж.

В комплексе разработаны четыре интерфейсные структуры для заполнения данных, иллюстрированные на изображениях 16, 17, 18, 19.

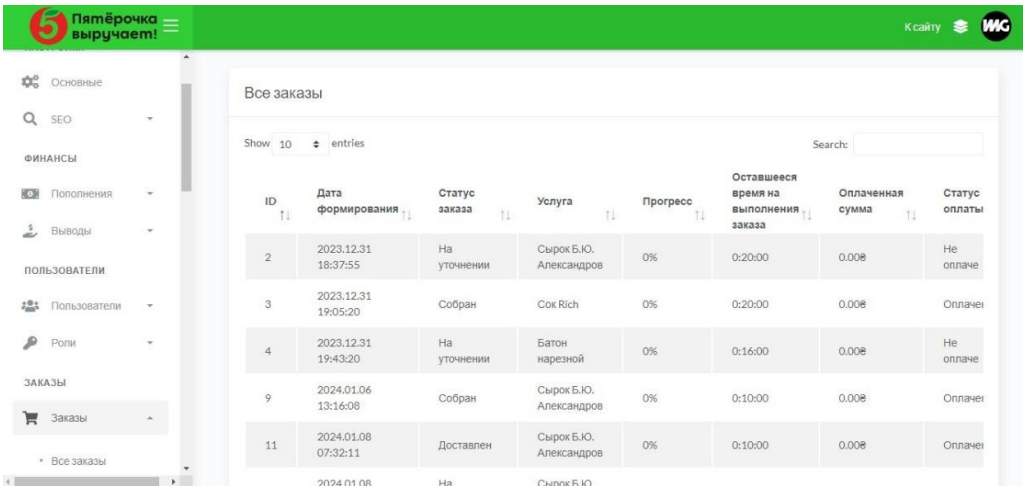

Рисунок 16 – Работа с заказами

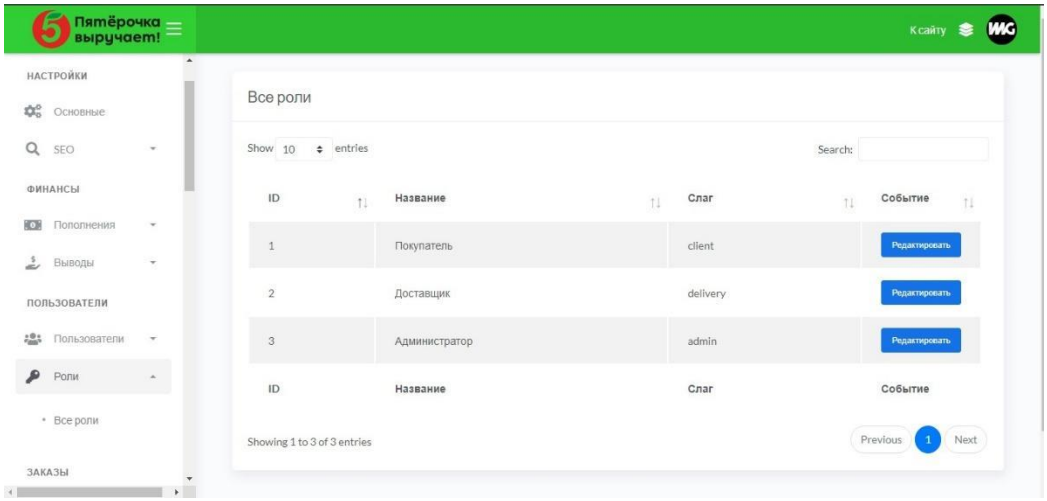

Рисунок 17 – Работа с ролями

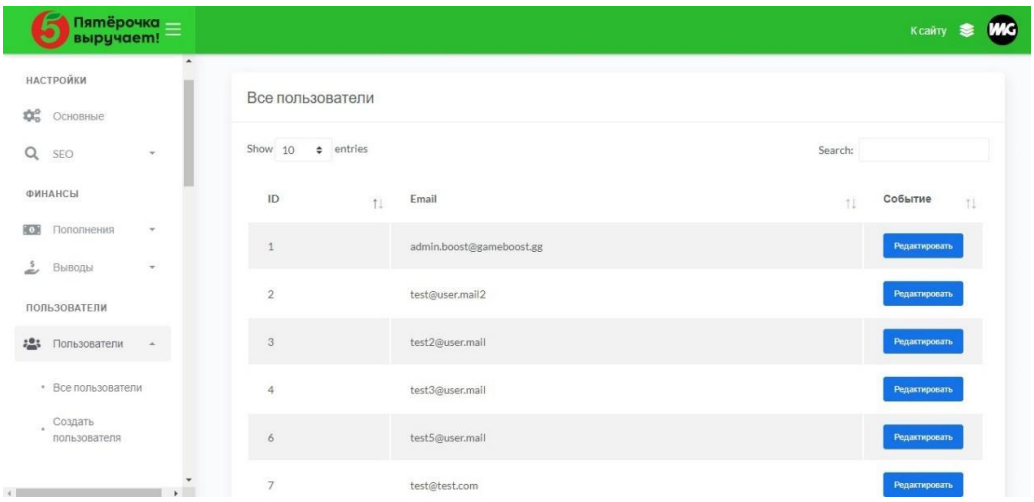

Рисунок 18 – Управление пользователями

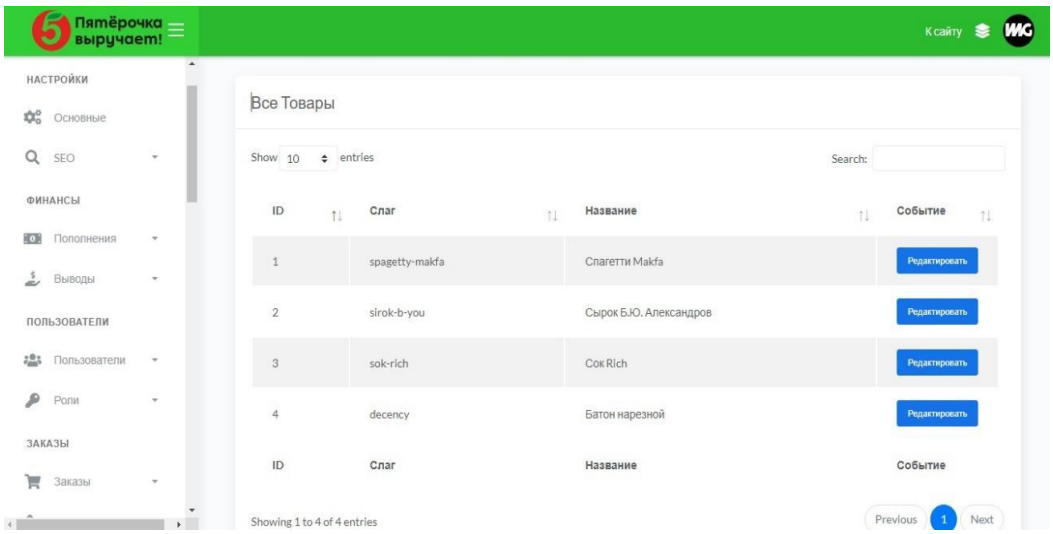

Рисунок 19 – Работа с товарами

Результативные данные представлены через запросы и аналитические отчеты.

Запрос – это структурированный объект, обеспечивающий доступ и модификацию данных через выборку из одной или более таблиц. Он поддерживает генерацию новых данных в таблице на основе информации, полученной из одной или нескольких изначально имеющихся таблиц.

Отчет представляет собой инструмент, предназначенный для генерации, анализа и вывода на печать специфического набора данных. Перед тем как направить документ на печать, его можно просмотреть на дисплее.

Запросы к базе данных обеспечивают извлечение информации, суммирование данных, их сортировку и соединение таблиц для получения

нужных выводов. Также они применяются для внесения, модификации и устранения записей в таблицах баз данных.

Отчеты служат инструментом для эффективной организации и визуализации информации, облегчая понимание и анализ представленных данных благодаря включению элементов, таких как графики, диаграммы и сводные таблицы, которые демонстрируют ключевые показатели в доступном виде. Функция предварительного просмотра отчетов на экране предоставляет возможность верификации актуальности и точности данных до момента их печати или применения в аналитических целях.

2.2 Программное обеспечение задачи (комплекса задач, автоматизированное рабочее место)

Функциональное дерево представляет собой иерархию функций, упорядоченных по уровням. В его структуре выделяются ключевые типы функций: основные, обеспечивающие выполнение главных задач системы, и вспомогательные, поддерживающие их исполнение. Визуальное представление автоматизированных функций, доступных пользователю, демонстрируется на рисунке 20 как иерархическое дерево функций.

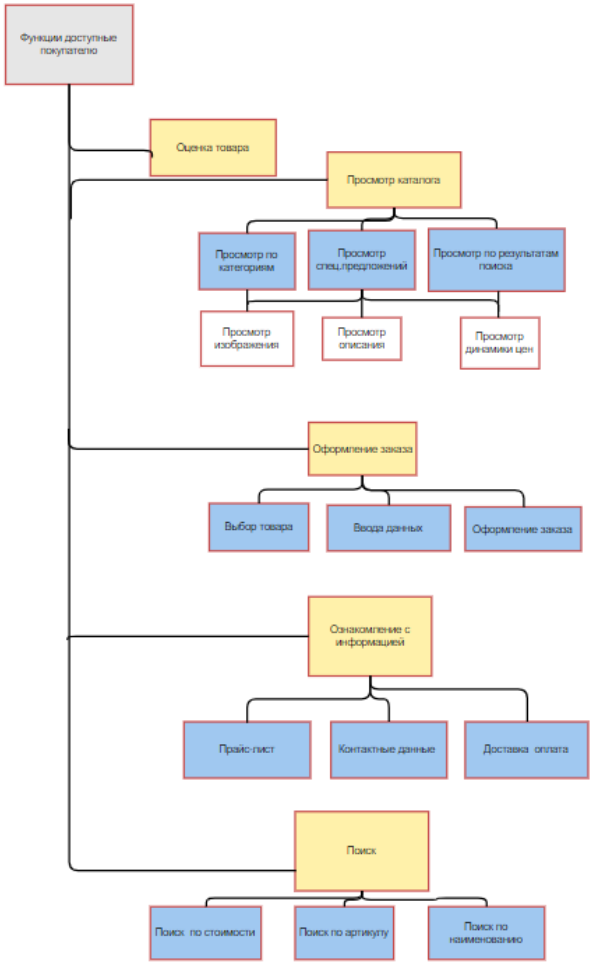

Рисунок 20 – Структурная схема функциональных возможностей системы.

Следовательно, у администратора есть доступ к следующим ключевым возможностям:

– перечень ассортимента продукции;

– поиск;

– изучение предоставленных данных;

– Модификация конкретного товарного наименования.

Для оптимального функционирования системы требуется сервер. В первую очередь, при подборе хостинга необходимо учитывать следующие аспекты:

– организация тех. поддержки;

– набор программного обеспечения и инструментов для администрирования веб-сайта;

– емкость дисковой памяти, выделяемая под хостинг сайта;

– ценовая политика и качество хостинг-сервисов.

Виды дисковых устройств хранения данных – различают три основных типа: механический жесткий диск (HDD), твердотельный накопитель (SSD) и твердотельный накопитель с неизменяемой памятью и интерфейсом NVMe (NVMe SSD).

Для серверов предпочтительным выбором является Ubuntu, версия операционной системы GNU/Linux, построенная на базе Debian GNU/Linux. Ее разработкой и финансовой поддержкой занимается компания Canonical.

Требуется инсталлировать на сервере комплекс ПО LAMP, состоящий из Linux, Apache, MySQL/MariaDB и PHP.

Linux Apache MySQL/MariaDB PHP (LAMP) представляет из себя комплект программного обеспечения на основе открытого кода, агрегирующий в себе четыре ключевых компонента, часто упоминаемых в качестве «слоев».

Linux служит надежной серверной платформой, обеспечивая выполнение критически важных задач. Эта операционная система, зарекомендовавшая себя мировым стандартом, находит применение в различных секторах за счет своей адаптивности и широких настраиваемых функций, превосходя в этом аспекте своих конкурентов.

Apache – это веб-сервер, оснащенный дополнительными модулями, которые обеспечивают совместимость веб-сервера со сценариями, написанными на разных языках программирования.

MySQL – это система управления базами данных, функционирующая на платформах Unix и Windows, характеризующаяся удобством в эксплуатации и применимая как для маломасштабных задач, так и для крупных, многофункциональных интернет-порталов.

Hypertext Preprocessor (PHP) – это серверный скриптовый язык программирования для создания динамического контента, аналогично его

применению возможно в Perl или Python. В сочетании с Apache, PHP способен генерировать динамические веб-страницы. Необходима поддержка PHP версии 7.4 и выше.

Laravel представляет собой бесплатный Hypertext Preprocessor (PHP) - фреймворк с открытым кодом, оптимизированный для разработки как отдельных, так и комплексных веб-сайтов. Он облегчает процессы аутентификации пользователей, маршрутизации запросов, управления сессиями и кэширования данных, а также упрощает структурирование приложения и взаимодействие с базами данных.

Программный компонент представляет собой отдельно функционирующий элемент софта, задача которого – осуществление конкретных операций или их комбинации. Он служит основной единицей в архитектуре ПО, предназначенной для достижения специфических результатов. Такие компоненты разрабатываются для выполнения уникальных задач и могут быть реинтегрированы или переадаптированы для использования в различных приложениях. Каждый компонент владеет уникальным интерфейсом, обеспечивающим связь между ним и другими частями системы, что способствует гибкости и масштабируемости программного кода.

Система идентификации и разрешений доступа:

– гарантирует надежную аутентификацию пользователей;

– администрирует привилегии пользователей для использования функций приложения;

– осуществляет процедуры проверки подлинности пользователя, предоставляя возможность авторизации через имя пользователя и пароль, а также поддерживает аутентификацию с использованием внешних сервисов.

1. Модуль управления товарами:

– позволяет администраторам добавлять, редактировать и удалять товары;

– обеспечивает управление атрибутами продукции, включая наименование, подробное описание, стоимость, визуальные материалы и другие параметры;

– позволяет классифицировать продукцию для облегчения поиска и управления.

2. Модуль управления складами:

– управление складами.

– обеспечивает инструменты для контроля параметров складирования.

– позволяет классифицировать склады для облегчения их поиска и навигации.

3. Система взаимодействия с поставщиками:

– обеспечивает пользовательский интерфейс для ввода информации о поставщиках;

– осуществляет верификацию корректности и целостности данных;

– обеспечивает направление поставщика в структуру товарных взаимоотношений для дальнейшего выполнения обязательств.

4. Управление пользователями включает в себя следующие аспекты: создание, редактирование, удаление и управление правами доступа пользователей через специализированные интерфейсы. В этом контексте, модуль управления пользователями представляет собой часть программного обеспечения, которая обеспечивает администраторам инструменты для эффективного управления учетными записями пользователей внутри системы, включая возможность настройки и контроля за уровнями доступа к различным ресурсам. Также этот модуль предусматривает механизмы для аутентификации и авторизации пользователей, обеспечивая тем самым безопасность системы:

– дает возможность администраторам упорядочивать контроль доступа пользователей, в том числе через блокирование и удаление аккаунтов;

– реализует возможности для анализа активности пользователей и контроля над их уровнями доступа;

– предоставляет инструменты для восстановления доступа через пароль и изменения персональных данных пользователей.

5. Модуль аналитики и отчетности:

– собирает информацию об активности пользователей, транзакциях и объемах продаж;

– внедряет методики обработки информации чтобы идентифицировать тенденции и модели;

– дает администраторам инструменты для создания отчетов и панелей мониторинга, что позволяет отслеживать работоспособность приложений и производительность бизнес-операций.

6. Интерфейс связи с внешними платформами:

– фасилитирует коммуникацию между платежными инфраструктурами для проведения транзакций;

– связывается с платформами доставки для упрощения и автоматизации распределения товаров;

– обеспечивает Application Programming Interface (API) для взаимодействия с другими информационными системами предприятия.

Эти программные компоненты расширяют возможности интернет-платформы компании, предназначенной для онлайн-торговли, упрощая клиентам процесс приобретения товаров и предоставляя администраторам инструменты для оптимизации и контроля за коммерческими операциями.

2.3 Технологическое обеспечение задачи (комплекса задач, автоматизированное рабочее место)

Автоматизированный технологический процесс обработки информации охватывает ряд последовательно выполняемых операций, начиная с сбора первичных данных и завершая выдачей конечных результатов.

Процедура – это комплекс процессов, осуществляемых над данными в рамках рабочей области.

Прогресс в технологических методах должен гарантировать полную автоматизированность в обработке данных с помощью разнообразных технических устройств, а также обеспечивать высокое качество получаемых результатов при минимализации времени и ресурсов, затрачиваемых на процесс.

Набор операций и последовательность их реализации определяются спецификой поставленных задач и конкретной технологической инфраструктурой на различных стадиях процесса обработки данных.

Основные факторы, влияющие на характер задач, включают объем обрабатываемых данных, регулярность их решения, а также уровень сложности применяемых алгоритмов. На каждом этапе решения задачи могут применяться разнообразные подходы и методики вычислений.

В процессе определения технологических процедур открывается потенциал для выбора оптимального метода обработки данных.

Конструирование технологического процесса для автоматизации обработки информации предусматривает детальную оценку бизнес-операций и критериев, предъявляемых компанией. Процесс влечет за собой идентификацию фаз обработки информационных потоков, отбор соответствующих программных и аппаратных решений для воплощения, а также формирование систем обеспечения качества и защиты данных.

При проектировании технологических процессов критические факторы включают улучшение эффективности работы систем, гарантирование надежности хранения данных, реализацию мероприятий по кибербезопасности и соответствие нормативно-правовым актам в сфере защиты информации. Эссенциально также предусматривать способность системы к расширению с целью обеспечения обработки увеличивающегося объема данных и ее адаптивность к эволюции бизнес-требований.

При поступлении товара в компанию процесс учета запускается сканированием штрих-кода через специализированное оборудование или ручным вводом данных, после чего эта информация транслируется в интерфейс веб-приложения для сохранения. Дальнейшая обработка собранных данных адаптируется под задачи и требования конечных пользователей.

Технологический процесс инициируется этапом аккумуляции данных. Этот этап критичен для производителей и специалистов, работающих в Excel.

Информация подается в систему вручную для ее каталогизации, в то время как ценники обновляются автоматически. Далее задачи обработки данных берет на себя компьютер. Проверка введенной информации осуществляется пользователями вручную. Параллельно, автоматизированный мониторинг информационных систем обеспечивает их надежность и безопасность.

Оператор отвечает за сортировку и актуализацию данных при получении новой информации. Обмен данными происходит через электронные письма. Все данные систематизированы и хранятся в архиве на бумажных носителях.

Для автоматизации и улучшения процесса, оптимисты высоко ценят автоагрегацию информации, используя, к примеру, специализированное ПО для экстракции данных через Neteller или функционал экспорта из Excel. Критически важным шагом является интеграция механизмов верификации правильности и полноты собранной информации перед ее загрузкой в цифровую обрабатывающую систему (компьютер), что значительно уменьшает риск внесения некорректных данных.

Программы для автоматизированной обработки данных способствуют оптимизации ранжирования и модификации информации, обеспечивая тем самым повышение производительности операторов при взаимодействии с входящими данными. Эффективно также сохранять информацию в цифровом формате, используя облачные хранилища или системы управления базами данных, для улучшения доступа к данным и их защиты.

Реализация электронной системы для обеспечения безопасного обмена информацией через защищенные каналы связи способствует быстрому обмену данными и уменьшению риска их утраты. В дополнение, важно разработать стратегию плавного перехода с бумажных архивов на электронные базы данных, что повысит эффективность обработки и удобство доступа к информации.

На рисунке 12 изображена процедура технологических операций по сбору, обработке и распределению информации.

Рисунок 12 – Процесс сборки, анализа и распространения данных.

Процесс автоматизированной обработки экономической информации охватывает ряд действий, которые проводятся последовательно от момента их начала до момента достижения запланированных итогов. Этот процесс организован по четырем основным фазам: сбору исходных данных, предварительной обработке, первичной обработке и завершающей фазе. В фазе сбора данных собранная информация регистрируется и загружается в систему обработки данных для дальнейшей работы. Вторая фаза, предварительная обработка, включает в себя прием, проверку и подготовку информации к основному этапу обработки. На этапе первичной обработки происходит непосредственное преобразование данных в предусмотренном порядке. Заключительная фаза включает в себя проверку полученных

результатов, их подготовку к использованию и распространению среди потребителей.

В диалоговой обработке связи между технологиями, используемыми за пределами и внутри компьютерных систем, настолько переплетены, что границы между ними стираются, аналогично отсутствию явного разграничения между начальным, основным и конечным этапами технологических операций. Это обусловлено непоследовательным характером работы в диалоговом модусе, предполагающим адаптивный подход без строго определенной последовательности действий. В результате, технологическая схема в диалоговом режиме принимает форму комплекса операций, которые укладываются в диалоговую структуру решаемой задачи и представляются через системное моделирование.

Сбор и документирование информации представляет собой преобразование информационных потоков в формат входных данных, кодируемых бинарными последовательностями. Хотя механизмы и методологии могут изменяться в зависимости от системы, существуют общие черты, присущие всем процессам сбора и регистрации данных.

Сбор информации представляет собой процесс переработки информационного потока, исходящего от управляемого объекта, через этапы восприятия и трансформации в документированную форму.

Подготовка информации представляет собой процедуру выборки данных, так как структура конкретной предметной области задает рамки для содержания и формы вносимой информации.

Мониторинг – это процесс, целью которого является предупреждение, выявление и корректировка недочетов в данных.

Обмен информацией является передачей данных между процессами в сфере информационных технологий (ИТ).

Передача может осуществляться:

– с помощью сетевых процедур;

– по каналам связи.

Процессы в области сетевых технологий включают:

– управление и направление трафика сети;

– коммутация;

– трансмиссия информации через коммуникационные каналы.

Передача данных по каналам связи включает:

– модуляцию – демодуляцию;

– кодирование – декодирование;

– координация и амплификация сигналов.

Технология связи строится на двух основных элементах:

1. Физический: управление водой / контактами, модемы, приборы.

2. Программное обеспечение: системное программирование устройств, включая алгоритмы кодирования и декодирования данных.

Информационная обработка – это трансформация данных. Архитектура обработки информации охватывает:

Структура управления компьютерными процессами обусловлена системным программированием, задачей которого является распределение и оптимизация вычислительных ресурсов. Эту функцию выполняют специализированные программные средства, направленные на взаимодействие с аппаратной частью компьютера. Совокупность этих программ, реализующих базовые алгоритмы работы с техническими и программными ресурсами, именуется операционной системой (ОС), представляющей собой мост между пользователями и компьютером.

Система автоматической обработки данных представляет собой комплекс программного обеспечения, задачей которого является выполнение структурированного преобразования входных данных в конечный продукт.

Программная модель обработки информации – это компьютерное приложение, предназначенное для трансформации машиночитаемых данных

в интерпретируемые человеком форматы, включающие текст, изображения, аудио и видео, обогащенные значимым контентом.

Управление информационными ресурсами включает формирование, сохранение и обновление базы данных, критически важной для выполнения оперативных задач управленческой системы. Процесс поиска информации тесно взаимосвязан с ее сохранением и представляет собой извлечение релевантных данных. Процедуры хранения охватывают ряд ключевых этапов:

1. Память включает формирование и сохранение структуры информации внутри системы памяти.
2. Выбор хранения
3. Модернизация (обеспечение актуальности хранимых данных, адекватно отражающих информационные запросы выполняемых операций в рамках системы)
4. Изъятие информации из базы данных в рамках операции селекции.

В процессе создания физической структуры базы данных используют трехуровневую схему моделирования:

1. Концептуальная схема информационной модели детально отражает суть предметной области, указывая на специфические данные и объем информации, требуемый для их хранения.
2. Схема логической структуры базы данных (определяет упорядоченный набор информационных элементов).
3. Физическое моделирование данных включает в себя определение способов их организации и доступа к устройствам хранения.
4. Прием – получение данных.
5. Обработка данных включает изучение информации вручную либо с использованием специализированного программного обеспечения (ПО).

2.4 Руководство пользователя

Руководство пользователя.

Инструкция предоставит знания по основополагающим возможностям системы для качественного контроля над продукцией и различными сторонами предпринимательства.

1. Вход в систему:

– Для входа в систему требуется аутентификация посредством логина и пароля, предоставленного администратором.

2. Управление товарами:

– Регистрация продукции: Кликните по ключу «Добавить товар», после чего введите всю важную информацию о продукте, включая его название, детальное описание, массу, код продукта и прочее.

– Модификация данных о продукции: доступна опция корректировки сведений по имеющимся товарам через активацию кнопки исправления.

– Ликвидация ассортимента: при возникновении потребности в исключении определенного наименования продукции, определитесь с нужным объектом и активируйте процесс, воспользовавшись функцией «Удалить».

3. Управление заказами:

– Просмотр заказов позволяет отслеживать все существующие заказы, ознакамливаясь с их текущим статусом исполнения и деталями заказа.

– Управление процессом выполнения заказов: актуализируйте статусы в соответствии с их текущей стадией (в работе, завершен и другие).

4. Управление поставщиками:

– Добавление поставщиков включает внесение данных о новом поставщике, например, наименование компании и другую информацию.

– Управление базой данных поставщиков: актуализируйте данные о поставщиках при возникновении такой необходимости.

5. Управление пользователями:

– Управление учетными записями: обладая административными привилегиями, вы в состоянии регистрировать новых участников и определять их функциональные обязанности.

– Управление доступом: модифицируйте уровни доступа участников в программе для точного сопоставления с их задачами.

Данное руководство предназначено для освоения системы управления товарами через веб-приложение, что станет ценным инструментом для развития вашего бизнеса. Инструментарий веб-приложения для управления товарами позволит вам не только аккуратно контролировать перечень товаров, их характеристики, стоимость и наличие на складе, но и предоставит функционал для мониторинга запасов, организации промо-акций и скидок, а также анализа данных о продажах. Внедрение данной системы способствует оптимизации процессов управления товарами, увеличению производительности труда и повышению качества обслуживания ваших клиентов.

Ввод в эксплуатацию инвентаризационной системы на базе веб-приложения начинается с разработки базы данных товаров, включающей информацию о наименованиях, описаниях, стоимости, наличии на складе и других параметрах. Далее осуществляется контроль за инвентарем, включая мониторинг прихода и расхода товаров.

Затем требуется сконфигурировать предложения и снижения цен, чтобы стимулировать интерес потенциальных покупателей и усилить приверженность нынешних клиентов. Критически важно реализовать систему мониторинга сбыта, обеспечивающую возможность оценивать востребованность продукции и результативность рекламных акций.

ЗАКЛЮЧЕНИЕ

В процессе реализации проекта была создана цифровая платформа для компании ООО «АГРОТОРГ» (торговая сеть «Пятерочка»), направленная на автоматизацию торговых операций через интернет. Эта платформа является современным digital-решением, направленным на оптимизацию процесса продаж, и предлагает потребителям индивидуальный подход и комфортное взаимодействие с маркой.

В ходе разработки применены инновационные технологические решения, способствующие созданию стабильной и легко масштабируемой платформы. Веб-приложение располагает обширными функциями, в том числе удобным механизмом поиска и сортировки товаров, интуитивным интерфейсом управления заказами, а также инструментами для организации рекламных кампаний и выполнения аналитики продаж.

В процессе реализации дипломной работы для достижения поставленной цели были выполнены следующие задачи:

– проведено исследование системы учета заказов на продукцию в предприятии ООО «АГРОТОРГ» (торговая сеть «Пятерочка»);

– исследованы программные решения для выполнения заказов;

– создано веб-приложение для реализации продукции через сайт компании;

– разработано веб-приложение для управления продажами продукции посредством цифровой платформы компании ООО «АГРОТОРГ» – ритейлер «Пятерочка»;

– разработана инструкция по применению системы для пользователей.

Выполненный анализ показывает, что созданное веб-приложение влияет на рост продаж, улучшает взаимодействие с покупателями, оптимизирует операционные процессы компании. В общей сложности, это веб-приложение

раскрывает новые возможности для усиления электронной торговли в контексте нашей компании.

В ходе реализации проекта участники приобрели значимые навыки в создании интернет-приложений и электронной торговли. Они также изучили предпочтения потребителей и научились учиться особенности общения с клиентами в цифровом пространстве.

В финале, создание интернет-приложения для электронной коммерческой платформы продажи продуктов компанией ООО «АГРОТОРГ» (торговая сеть «Пятерочка») играет ключевую роль в прогрессе данной фирмы, обеспечивая укрепление ее положения на рынке цифровой торговли.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Литература

1. Айзенменгер, Р. HTML 3.2/4.0. Справочник / Р. Айзенменгер. – М.: СИНТЕГ, 2022. – 368 с.

2. Воройский, Ф.С. Информатика. Новый систематизированный толковый словарь / Ф.С. Воройский. – Москва: Машиностроение, 2023. – 155 с..

3. Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. – М.: СПб. [и др.] : Питер, 2023. – 464 с.

4. Скляренко В.К. Экономика предприятия. М., 2022. – 32 с.

5. Управленческий учет: Учебное пособие / Под ред. А.Д. Шеремета. – 2-е изд., испр. – М.: ИДФБК-ПРЕСС, 2023. – 512с.

6. Гарнаев, Андрей WEB-программирование на Java и JavaScript / Андрей Гарнаев , Сергей Гарнаев. – М.: БХВ-Петербург, 2021. – 822 с.

7. Дебольт HTML и CSS. Совместное использование / Дебольт, Вирджиния. – М.: НТ Пресс, 2020. – 512 с.

8. Дронов, В. JavaScript в Web-дизайне / В. Дронов. – М.: СПб: БХВ, 2020. – 880 с.

9. Дронов, В. PHP, MySQL и Dreamweaver MX 2004 / В. Дронов. – М.: Книга по Требованию, 2021. – 441 с.

10. Макфарланд, Дэвид JavaScript и jQuery. Исчерпывающее руководство (+ DVD-ROM) / Дэвид Макфарланд. – М.: Эксмо, 2023. – 688 с.

11. Шапошников, И. Web-сайт своими руками / И. Шапошников. – М.: БХВ-Петербург, 2021. – 224 с.

12. Якобсон, Й. Концепция разработки Web-сайтов. Как успешно разработать Web-сайт с применением мультимедиа-технологий / Й. Якобсон. – М.: НТ Пресс, 2023. – 496 с.

13. Мэтью, Дэвид HTML5. Разработка веб-приложений / Дэвид Мэтью. – М.: Рид Групп, 2020. – 320 с.

14. Магда, Ю. С. Raspberry Pi. Руководство по настройке и применению / Ю.С. Магда. – М.: ДМК Пресс, 2020. – 188 с.

15. Колисниченко, Д.Н. PHP 5/6 и MySQL 6. Разработка Web-приложений (+ CD-ROM) / Д.Н. Колисниченко. – М.: БХВ-Петербург, 2020. – 735 с.

16. Дакетт, Джон Основы веб-программирования с использованием HTML, XHTML и CSS / Джон Дакетт. – М.: Эксмо, 2020. – 768 с.

Электронные ресурсы

17. https://dostavka.5ka.ru/

18. https://laravel.com/

19. https://www.php.net/manual/ru/intro-whatis.php

20. http://www.php.su/php/?p

ПРИЛОЖЕНИЕ А

Таблица PaymentController

```
<?php
namespace App\Http\Controllers\Admin; use App\Models\Payment;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller; use DataTables;
class PaymentController extends Controller
{
/**
*       Display a listing of the resource.
*
*       @return \Illuminate\Http\Response
*/
public function index()
{
if (request()->ajax()) {
return DataTables::eloquent(Payment::query())
->addColumn('action', function($payment) {
return sprintf('<a href="%s" class="btn btn-sm btn-primary">%s</a>',
route('admin.payments.edit', ['payment' => $payment->id]), ('Редактировать'));
})
->addColumn('method', function($payment) {
return config('app.methods')[$payment->method];
})
->rawColumns(['action', 'method'])
->make(true);
}
return view('admin.payments.list');
}
/**
*       Show the form for creating a new resource.
*
*       @return \Illuminate\Http\Response
*/
public function create()
{
return view('admin.payments.create');
}
/**
*       Store a newly created resource in storage.
*
*       @param \Illuminate\Http\Request $request
*       @return \Illuminate\Http\Response
*/
public function store(Request $request)
{
$request->validate([
'sum' => ['required', 'numeric'],
'method' => ['required', 'numeric'],
'info' => ['required', 'string'],
]);
```

```
        $payment = Payment::create($request->all());

        return response()->json(['type' => 'success', 'title' => ('Успешно'),
        'msg' =>
          ('Сохранено'), 'url' => route('admin.payments.edit', ['payment' => $payment->id ])]);
        }
        /**
        *       Show the form for editing the specified resource.
        *
        *       @param \App\Models\Payment $payment
        *       @return \Illuminate\Http\Response
        */
        public function edit(Payment $payment)
        {
        return view('admin.payments.edit', compact('payment'));
        }
        /**
        *       Update the specified resource in storage.
        *
        *       @param \Illuminate\Http\Request $request
        *       @param \App\Models\Payment $payment
        *       @return \Illuminate\Http\Response
        */
        public function update(Request $request, Payment $payment)
        {
        $request->validate([
        'sum' => ['required', 'numeric'],
        'method' => ['required', 'numeric'],
        'info' => ['required', 'string'],
        ]);

        $payment->update($request->all());

        return response()->json(['type' => 'success', 'title' => ('Успешно'),
        'msg' =>
          ('Сохранено'), 'url' => route('admin.payments.edit', ['payment' => $payment->id])]);
        }

        /**
        *       Remove the specified resource from storage.
        *
        *       @param \App\Models\Payment $payment
        *       @return \Illuminate\Http\Response
        */
        public function destroy(Payment $payment)
        {
        $payment->delete();

        return response()->json(['type' => 'success', 'title' =>  ('Успешно'), 'msg' =>  ('Удален'),
'url' => route('admin.payments.index')]);
        }
```

ПРИЛОЖЕНИЕ Б

Таблица ProductController

```
<?php
namespace App\Http\Controllers\Admin; use App\Models\Product;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller; use DataTables;
class ProductController extends Controller
{
/**
*       Display a listing of the resource.
*
*       @return \Illuminate\Http\Response
*/
public function index()
{
if (request()->ajax()) {
return DataTables::eloquent(Product::query())
->addColumn('action', function($product) {
return sprintf('<a href="%s" class="btn btn-sm btn-primary">%s</a>',
route('admin.products.edit', ['product' => $product->id]), ('Редактировать'));
})
->addColumn('provider_id', function($product) { return $product->provider->name;
})
->addColumn('stock_id', function($product) { return $product->stock->name;
})
->addColumn('availability', function($product) { return $product->availability ? 'Есть' :
'Нет';
})
->rawColumns(['action', 'provider_id', 'stock_id'])
->make(true);
}
return view('admin.products.list');
}
/**
*       Show the form for creating a new resource.
*
*       @return \Illuminate\Http\Response
*/
public function create()
{
return view('admin.products.create');
}
/**
*       Store a newly created resource in storage.
*
*       @param \Illuminate\Http\Request $request
*       @return \Illuminate\Http\Response
*/
public function store(Request $request)

{
$request->validate([
```

```
        'name' => ['required', 'string', 'max:255'], 'availability' => ['required', 'boolean'], 'weight'
=> ['required', 'numeric'],
        'info' => ['nullable', 'string'],
        'provider_id' => ['required', 'numeric', 'exists:providers,id'], 'stock_id' => ['required',
'numeric', 'exists:stocks,id'],
        ]);

        $product = Product::create($request->all());

        return response()->json(['type' => 'success', 'title' => ('Успешно'),
        'msg' =>
          ('Сохранено'), 'url' => route('admin.products.edit', ['product' => $product->id ])]);
        }

        /**
        *       Show the form for editing the specified resource.
        *
        *       @param \App\Models\Product $product
        *       @return \Illuminate\Http\Response
        */
        public function edit(Product $product)
        {
        return view('admin.providers.edit', compact('provider'));
        }

        /**
        *       Update the specified resource in storage.
        *
        *       @param \Illuminate\Http\Request $request
        *       @param \App\Models\Product $product
        *       @return \Illuminate\Http\Response
        */
        public function update(Request $request, Product $product)
        {
        $request->validate([
        'name' => ['required', 'string', 'max:255'], 'availability' => ['required', 'boolean'], 'weight'
=> ['required', 'numeric'],
        'info' => ['nullable', 'string'],
        'provider_id' => ['required', 'numeric', 'exists:providers,id'], 'stock_id' => ['required',
'numeric', 'exists:stocks,id'],
        ]);

        $product->update($request->all());

        return response()->json(['type' => 'success', 'title' => ('Успешно'),
        'msg' =>
          ('Сохранено'), 'url' => route('admin.products.edit', ['product' => $product->id ])]);
        }

        /**

        *       Remove the specified resource from storage.
```

```
        *
        *       @param \App\Models\Product $product
        *       @return \Illuminate\Http\Response
        */
        public function destroy(Product $product)
        {
        $product->delete();

        return response()->json(['type' => 'success', 'title' =>  ('Успешно'), 'msg' =>  ('Удален'),
'url' => route('admin.products.index')]);
        }
        }
```

ПРИЛОЖЕНИЕ В

Таблица ProviderController

```
<?php
namespace App\Http\Controllers\Admin; use App\Models\Provider;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller; use DataTables;
class ProviderController extends Controller
{
/**
*       Display a listing of the resource.
*
*       @return \Illuminate\Http\Response
*/
public function index()
{
if (request()->ajax()) {
return DataTables::eloquent(Provider::query())
->addColumn('action', function($provider) {
return sprintf('<a href="%s" class="btn btn-sm btn-primary">%s</a>',
route('admin.providers.edit', ['provider' => $provider->id]), ('Редактировать'));
})
->rawColumns(['action'])
->make(true);
}
return view('admin.providers.list');
}
/**
*       Show the form for creating a new resource.
*
*       @return \Illuminate\Http\Response
*/
public function create()
{
return view('admin.providers.create');
}
/**
*       Store a newly created resource in storage.
*
*       @param \Illuminate\Http\Request $request
*       @return \Illuminate\Http\Response
*/
public function store(Request $request)
{
$request->validate([
'name' => ['required', 'string', 'max:255'],
'address' => ['required', 'string', 'max:255'],
]);

$provider = Provider::create($request->all());
```

```
return response()->json(['type' => 'success', 'title' => ('Успешно'), 'msg' =>
  ('Сохранено'), 'url' => route('admin.providers.edit', ['provider' => $provider->id ])]);
}
/**
*  Show the form for editing the specified resource.
*
*  @param \App\Models\Provider $provider
*  @return \Illuminate\Http\Response
*/
public function edit(Provider $provider)
{
return view('admin.providers.edit', compact('provider'));
}
/**
*  Update the specified resource in storage.
*
*  @param \Illuminate\Http\Request $request
*  @param \App\Models\Provider $provider
*  @return \Illuminate\Http\Response
*/
public function update(Request $request, Provider $provider)
{
$request->validate([
'name' => ['required', 'string', 'max:255'],
'address' => ['required', 'string', 'max:255'],
]);
$provider->update($request->all());
return response()->json(['type' => 'success', 'title' => ('Успешно'), 'msg' =>
  ('Сохранено'), 'url' => route('admin.providers.edit', ['provider' => $provider->id])]);
}
/**
*  Remove the specified resource from storage.
*
*  @param \App\Models\Provider $provider
*  @return \Illuminate\Http\Response
*/
public function destroy(Provider $provider)
{
$provider->delete();
return response()->json(['type' => 'success', 'title' =>  ('Успешно'), 'msg' =>  ('Удален'),
'url' => route('admin.providers.index')]);
}
}
```

ПРИЛОЖЕНИЕ Г

Таблица RoleController

```
<?php
namespace App\Http\Controllers\Admin; use App\Models\Role;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller; use DataTables;
class RoleController extends Controller
{
/**
*       Display a listing of the resource.
*
*       @return \Illuminate\Http\Response
*/
public function index()
{
if (request()->ajax()) {
return DataTables::eloquent(Role::query())
->addColumn('action', function($role) {
return sprintf('<a href="%s" class="btn btn-sm btn-primary">%s</a>',
route('admin.roles.edit', ['role' => $role->id]), ('Редактировать'));
})
->rawColumns(['action'])
->make(true);
}
return view('admin.roles.list');
}
/**
*       Show the form for creating a new resource.
*
*       @return \Illuminate\Http\Response
*/
public function create()
{
return view('admin.roles.create');
}
/**
*       Store a newly created resource in storage.
*
*       @param \Illuminate\Http\Request $request
*       @return \Illuminate\Http\Response
*/
public function store(Request $request)
{
$request->validate([
'name' => ['required', 'string', 'max:255'],
'slug' => ['required', 'string', 'max:255', 'unique:roles,slug'],
]);
$role = Role::create($request->all());

return response()->json(['type' => 'success', 'title' => ('Успешно'),
'msg' =>
 ('Сохранено'), 'url' => route('admin.roles.edit', ['role' => $role->id ])]);
```

```
        }
        /**
        *       Show the form for editing the specified resource.
        *
        *       @param \App\Models\Role $role
        *       @return \Illuminate\Http\Response
        */
        public function edit(Role $role)
        {
        return view('admin.roles.edit', compact('role'));
        }
        /**
        *       Update the specified resource in storage.
        *
        *       @param \Illuminate\Http\Request $request
        *       @param \App\Models\Role $role
        *       @return \Illuminate\Http\Response
        */
        public function update(Request $request, Role $role)
        {
        $request->validate([
        'name' => ['required', 'string', 'max:255'],
        'slug' => ['required', 'string', 'max:255', Rule::unique('roles')->ignore($role->id),],
        ]);

        $role->update($request->except('permissions'));
        return response()->json(['type' => 'success', 'title' => ('Успешно'),
        'msg' =>
          ('Сохранено'), 'url' => route('admin.roles.edit', ['role' => $role->id])]);
        }

        /**
        *       Remove the specified resource from storage.
        *
        *       @param \App\Models\Role $role
        *       @return \Illuminate\Http\Response
        */
        public function destroy(Role $role)
        {
        $role->delete();
        return response()->json(['type' => 'success', 'title' =>  ('Успешно'), 'msg' =>  ('Удален'),
'url' => route('admin.roles.index')]);
        }
        }
```

ПРИЛОЖЕНИЕ Д

Таблица StockController

```
<?php
namespace App\Http\Controllers\Admin; use App\Models\Stock;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller; use DataTables;

class StockController extends Controller
{
/**
*       Display a listing of the resource.
*
*       @return \Illuminate\Http\Response
*/
public function index()
{
if (request()->ajax()) {
return DataTables::eloquent(stock::query())
->addColumn('action', function($stock) {
return sprintf('<a href="%s" class="btn btn-sm btn-primary">%s</a>',
route('admin.stocks.edit', ['stock' => $stock->id]), ('Редактировать'));
})
->rawColumns(['action'])
->make(true);
}
return view('admin.stocks.list');
}
/**
*       Show the form for creating a new resource.
*
*       @return \Illuminate\Http\Response
*/
public function create()
{
return view('admin.stocks.create');
}
/**
*       Store a newly created resource in storage.
*
*       @param \Illuminate\Http\Request $request
*       @return \Illuminate\Http\Response
*/
public function store(Request $request)
{
$request->validate([
'name' => ['required', 'string', 'max:255'],
'address' => ['required', 'string', 'max:255'],
]);

$stock = Stock::create($request->all());
```

```
        return response()->json(['type' => 'success', 'title' => ('Успешно'),
        'msg' =>
          ('Сохранено'), 'url' => route('admin.stocks.edit', ['stock' => $stock->id ])]);
        }
        /**
        *       Show the form for editing the specified resource.
        *
        *       @param \App\Models\Stock $stock
        *       @return \Illuminate\Http\Response
        */
        public function edit(Stock $stock)
        {
        return view('admin.stocks.edit', compact('stock'));
        }
        /**
        *       Update the specified resource in storage.
        *
        *       @param \Illuminate\Http\Request $request
        *       @param \App\Models\Stock $stock
        *       @return \Illuminate\Http\Response
        */
        public function update(Request $request, Stock $stock)
        {
        $request->validate([
        'name' => ['required', 'string', 'max:255'],
        'address' => ['required', 'string', 'max:255'],
        ]);
        $stock->update($request->all());

        return response()->json(['type' => 'success', 'title' => ('Успешно'),
        'msg' =>
          ('Сохранено'), 'url' => route('admin.stocks.edit', ['stock' => $stock->id])]);
        }
        /**
        *       Remove the specified resource from storage.
        *
        *       @param \App\Models\Stock $stock
        *       @return \Illuminate\Http\Response
        */
        public function destroy(Stock $stock)
        {
        $stock->delete();
        return response()->json(['type' => 'success', 'title' =>  ('Успешно'), 'msg' =>  ('Удален'),
'url' => route('admin.stocks.index')]);
        }
        }
```

ПРИЛОЖЕНИЕ Е

Таблица ProviderController

```
<?php
namespace App\Http\Controllers\Admin; use Illuminate\Http\Request;
use App\Http\Controllers\Controller; use Illuminate\Validation\Rule;
use App\Models\User; use DataTables;
class UserController extends Controller
{
/**
*	Display a listing of the resource.
*
*	@return \Illuminate\Http\Response
*/
public function index()
{
if (request()->ajax()) {
$builder = User::query();

if(!is_null(request()->get('role')))
$builder->where('role_id', request()->get('role'));

return DataTables::eloquent($builder)
->addColumn('action', function($user) {
return sprintf('<a href="%s" class="btn btn-sm btn-primary">%s</a>',
route('admin.users.edit', ['user' => $user->id]), ('Редактировать'));
})
->addColumn('role', function($user) { //Доделать роль return $user->role->name;
})
->addColumn('created_at', function($user) { //Доделать роль return $user->created_at-
>format('Y.m.d H:i');
})
->rawColumns(['action', 'role', 'created_at'])
->make(true);
}
return view('admin.users.list');
}
/**
*	Show the form for creating a new resource.
*
*	@return \Illuminate\Http\Response
*/
public function create()
{
return view('admin.users.create');
}
/**
*	Store a newly created resource in storage.
*
*	@param \Illuminate\Http\Request $request

*	@return \Illuminate\Http\Response
*/
```

```
public function store(Request $request)
{
$request->validate([
'email' => ['required', 'string', 'max:255', 'email', 'unique:users'], 'name' => ['required',
'string', 'max:255'],
'role_id' => ['required', 'numeric', 'exists:roles,id'],
]);
$user = User::create($request->all());

return response()->json(['type' => 'success', 'title' => ('Успешно'),
'msg' =>
 ('Сохранено'), 'url' => route('admin.users.edit', ['user' => $user->id])]);
}
/**
*  Show the form for editing the specified resource.
*
*  @param int $id
*  @return \Illuminate\Http\Response
*/
public function edit(User $user)
{
return view('admin.users.edit', compact('user'));
}
/**
*  Update the specified resource in storage.
*
*  @param \Illuminate\Http\Request $request
*  @param int $id
*  @return \Illuminate\Http\Response
*/
public function update(Request $request, User $user)
{
$request->validate([
'email' => ['required', 'string', 'max:255', 'email', Rule::unique('users')->ignore($user-
>id)], 'name' => ['required', 'string', 'max:255'],
'role_id' => ['required', 'numeric', 'exists:roles,id'],
]);
$user->update($request->all());
return response()->json(['type' => 'success', 'title' => ('Успешно'),
'msg' =>
 ('Сохранено'), 'url' => route('admin.users.edit', ['user' => $user->id])]);
}
/**
*  Remove the specified resource from storage.
*
*  @param int $id
*  @return \Illuminate\Http\Response
*/
public function destroy(User $user)
{
$user->delete();
```

```
        return response()->json(['type' => 'success', 'title' =>  ('Успешно'), 'msg' =>  ('Удален'),
'url' => route('admin.users.index')]);
        }
        }
```

ПРИЛОЖЕНИЕ Ж

Таблица OrderController

```
<?php
namespace App\Http\Controllers\Admin;

use App\Models\Order;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller; use DataTables;
class OrderController extends Controller
{
/**
*       Display a listing of the resource.
*
*       @return \Illuminate\Http\Response
*/
public function index()
{
if (request()->ajax()) {
return DataTables::eloquent(Order::query())
->addColumn('action', function($order) {
return sprintf('<a href="%s" class="btn btn-sm btn-primary">%s</a>',
route('admin.orders.edit', ['order' => $order->id]),  ('Р РµРґР°РєС‚РёСЂРѕРІР°С‚СЊ'));
})
->addColumn('status', function($order) { return config('app.statuses')[$order->status];
})
->addColumn('delivery', function($order) { return config('app.delivery')[$order-
>delivery];
})
->rawColumns(['action', 'status', 'delivery'])
->make(true);
}
return view('admin.orders.list');
}
/**
*       Show the form for creating a new resource.
*
*       @return \Illuminate\Http\Response
*/
public function create()
{
return view('admin.orders.create');
}
/**
*       Store a newly created resource in storage.
*
*       @param \Illuminate\Http\Request $request
*       @return \Illuminate\Http\Response
*/
public function store(Request $request)
{
$request->validate([
```

```
        'sum' => ['required', 'numeric'],
        'status' => ['required', 'numeric'], 'delivery' => ['required', 'numeric'],
        ]);

        $order = Order::create($request->all());
        return response()->json(['type' => 'success', 'title' =>  ('PJCЃPїPµC€PSPs'), 'msg' =>
          ('PЎPsC…CЂP°PSPµPSPs'), 'url' => route('admin.orders.edit', ['order' => $order->id
])]);
        }
        /**
        *       Show the form for editing the specified resource.
        *
        *       @param \App\Models\Order $order
        *       @return \Illuminate\Http\Response
        */
        public function edit(Order $order)
        {
        return view('admin.orders.edit', compact('order'));
        }
        /**
        *       Update the specified resource in storage.
        *
        *       @param \Illuminate\Http\Request $request
        *       @param \App\Models\Order $order
        *       @return \Illuminate\Http\Response
        */
        public function update(Request $request, Order $order)
        {
        $request->validate([
        'sum' => ['required', 'numeric'],
        'status' => ['required', 'numeric'], 'delivery' => ['required', 'numeric'],
        ]);
        $order->update($request->all());
        return response()->json(['type' => 'success', 'title' =>  ('PJCЃPїPµC€PSPs'), 'msg' =>
          ('PЎPsC…CЂP°PSPµPSPs'), 'url' => route('admin.orders.edit', ['order' => $order-
>id])]);
        }
        /**
        *       Remove the specified resource from storage.
        *
        *       @param \App\Models\Order $order
        *       @return \Illuminate\Http\Response
        */
        public function destroy(Order $order)
        {
        $order->delete();
        return response()->json(['type' => 'success', 'title' =>  ('PJCЃPїPµC€PSPs'), 'msg' =>
          ('PJPrP°P»PµPS'), 'url' => route('admin.orders.index')]);
```