TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.02 Информационные системы и технологии
Отделение информационных технологий

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Разработка комплексной системы управления складом для интернет-магазинов

УДК 004.415.2:004.455.1:004.774

Студент

Группа	ФИО	Подпись	Дата
8ИМ91	Трофимова Анастасия Евгеньевна		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н.		

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН	Верховская Марина Витальевна	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ООД	Федоренко Ольга Юрьевна	д.м.н		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н.		

Томск – 2021 г.

ЗАПЛАНИРОВАННЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ПРОГРАММЕ: 09.04.02 «ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ» (СИСТЕМНАЯ ИНЖЕНЕРИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ)

Код результатов	**Результаты обучения (выпускник должен быть готов)**	**Требования ФГОС ВО (3++), СУОС, критерии АИОР, требования профессиональных стандартов (ПК-1, …, ПК-11)**
Р1	Воспринимать и самостоятельно приобретать, развивать и применять математические, естественно-научные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте.	Требования ФГОС ВО (3++) (ОПК-1,2; ПК-1; УК-1,4,6), критерий 5 АИОР (п. 1.1), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р2	Владеть и применять методы и средства получения, хранения, переработки и представления информации посредством современных компьютерных технологий, в том числе в глобальных компьютерных сетях.	Требования ФГОС ВО (3++) (ОПК-1,2,6,7; ПК-1,2,3,5,10; УК-1), критерий 5 АИОР (п. 1.1, 1.2), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р3	Демонстрировать способность выстраивать логику рассуждений и высказываний, основанных на интерпретации данных, интегрированных из разных областей науки и техники, выносить суждения на основании неполных данных, анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями.	Требования ФГОС ВО (3++) (ОПК-1,3,6; ПК-5,6; УК-1,6), критерий 5 АИОР (п. 1.2), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р4	Анализировать и оценивать уровни своих компетенций в сочетании со способностью и готовностью к саморегулированию дальнейшего образования и профессиональной мобильности. Демонстрировать способность к самостоятельному обучению новым методам исследования, способность самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения.	Требования ФГОС ВО (3++) (ОПК-1,4,6; УК-6), критерий 5 АИОР (п. 1.6, п. 2.2,2.6.), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.

Код результатов	Результаты обучения (выпускник должен быть готов)	Требования ФГОС ВО (3++), СУОС, критерии АИОР, требования профессиональных стандартов (ПК-1, …, ПК-11)
Р5	Владеть современными коммуникативными технологиями, в том числе на иностранном языке для академического и профессионального взаимодействия. Владеть, по крайней мере, одним из иностранных языков на уровне социального и профессионального общения, применять специальную лексику и профессиональную терминологию языка.	Требования ФГОС ВО (3++) (ОПК-1,3; УК-3,4,5; ПК-7,8,9). Запросы студентов, отечественных и зарубежных работодателей.
Р6	Использовать на практике умения и навыки в организации исследовательских, проектных работ и профессиональной эксплуатации современных программных и информационных систем, в управлении коллективом. Способность организовывать и эффектно руководить работой команды проекта при разработке программных и информационных систем.	Требования ФГОС ВО (3++) (УК-2,3,5; ПК-5,6,7,8,11; ОПК1,8), критерий 5 АИОР (п. 2.1, п. 2.3, п. 1.5), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р7	Разрабатывать стратегии проектирования, критерии эффективности и ограничения применимости новых методов и средств проектирования и разработки программных систем.	Требования ФГОС ВО (3++) (УК-1,3; ПК-1,3,10; ОПК2,4,6,7), критерий 5 АИОР (п. 2.2), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р8	Планировать и проводить теоретические и экспериментальные (численные) исследования в области создания программных систем. Оценивать и выбирать вариант архитектуры программной/информационной системы.	Требования ФГОС ВО (3++) (ОПК-1,4,6,7; ПК-1,3,10; УК1,3), критерий 5 АИОР (п. 1.4), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р9	Владеть методами и средствами инженерии требований к системам, управления качеством программного обеспечения и системной интеграции/модернизации программного обеспечения.	Требования ФГОС ВО (3++) (УК-1; ОПК-4,5,7; ПК-1,2,4,8,11). Запросы студентов, отечественных и зарубежных работодателей.

Код результатов	Результаты обучения (выпускник должен быть готов)	Требования ФГОС ВО (3++), СУОС, критерии АИОР, требования профессиональных стандартов (ПК-1, …, ПК-11)
Р10	Владеть современными инструментальными средствами программирования и технологиями управления данными. Использовать их при разработке требований, при проектировании и создании программного обеспечения, информационных систем/автоматизированных систем управления производством.	Требования ФГОС ВО (3++) (ПК-1,2,4,5,7,9,11; ОПК-2,5,7; УК-2). Запросы студентов, отечественных и зарубежных работодателей.
Р11	Осуществлять проектирование и разработку веб и мультимедийных приложений в среде корпоративных и глобальных информационно-телекоммуникационных систем.	Требования ФГОС ВО (3++) (ПК-1,2,3,5,6,9,11; ОПК2,4,5,7; УК-2,3,5). Запросы студентов, отечественных и зарубежных работодателей.
Р12	Осуществлять управление процессами внедрения/сопровождения (модернизации, интеграции) программных и информационных систем на основе принципов и методов системной инженерии.	Требования ФГОС ВО (3++) (ОПК-4,6,8; ПК-1,4,5,6,8,9,11; УК-2,3,4), критерий 5 АИОР (п. 2.6), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.02 Информационные системы и технологии
Отделение информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____ Савельев А.О.
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Магистерской диссертации

(бакалаврской работы, дипломного проекта/работы, магистерской диссертации)

Студенту:

Группа	ФИО
8ИМ91	Трофимова Анастасия Евгеньевна

Тема работы:

Разработка комплексной системы управления складом для интернет-магазинов	
Утверждена приказом директора (дата, номер)	№40-4/с от 09.02.2021

Срок сдачи студентом выполненной работы:	08.06.2021

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе *(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к особенностям функционирования (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.).*	Объектом исследования данной работы является создание комплексной системы управления складскими процессами. Областью применения объекта исследования является сфера малого и среднего бизнеса интернет-магазинов.

Перечень подлежащих исследованию, проектированию и разработке вопросов *(аналитический обзор по литературным источникам с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования, конструирования; содержание процедуры исследования, проектирования, конструирования; обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе).*	Проведение литературного анализа предметной области. Обзор рынка систем управления складом и анализ конкурентных решений. Формирование функциональных требований к разрабатываемой системе на основе сравнения конкурентных решений Обзор современного стека технологий MEAN для создания веб-приложений. Проектирование и разработка веб-приложения на основе стека MEAN с функционалом, позволяющим вести учет и визуализировать расположение поступающей на склад продукции, вести учет сформированных заказов и сопутствующих расходов интернет-магазина, отображать складскую статистику товарооборота компании, а также позволяющим интегрировать интернет-магазин в систему управления складом с помощью REST API.
Перечень графического материала *(с точным указанием обязательных чертежей)*	Схема взаимодействия бизнес-процессов интернет-магазина для работы с приложением Схема коллекций NoSQL базы данных MongoDB Дерево компонентов приложения

Консультанты по разделам выпускной квалификационной работы
(с указанием разделов)

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Верховская Марина Витальевна
Социальная ответственность	Федоренко Ольга Юрьевна
Раздел на иностранном языке	Сидоренко Татьяна Валерьевна

Названия разделов, которые должны быть написаны на русском и иностранном языках:

3.1 Выбор стека технологий для создания веб-приложения

3.2 Обзор работы REST API

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	01.03.2021

Задание выдал руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ91	Трофимова Анастасия Евгеньевна		

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.02 Информационные системы и технологии
Отделение информационных технологий
Период выполнения весенний семестр 2020 /2021 учебного года

Форма представления работы:

Магистерской диссертации

(бакалаврская работа, дипломный проект/работа, магистерская диссертация)

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	08.06.2021

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
15.03.2021	Раздел «Аналитический обзор предметной области»	15
31.03.2021	Раздел «Анализ конкурентных решений систем управления складом»	15
09.05.2021	Раздел «Проектирование и разработка комплексной системы управления складом»	45
16.05.2021	Раздел «Экономическая часть»	10
24.05.2021	Раздел «Социальная ответственность»	10
31.05.2021	Раздел на иностранном языке	5

СОСТАВИЛ:
Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н.		

СОГЛАСОВАНО:
Руководитель ООП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н.		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8ИМ91	Трофимова Анастасия Евгеньевна

Школа	Инженерная школа информационных технологий и робототехники	Отделение (НОЦ)	Отделение информационных технологий
Уровень образования	Магистратура	Направление/специальность	09.04.02 Информационные системы и технологии

Тема ВКР:

Разработка комплексной системы управления складом для интернет-магазинов	
Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих	Стоимость ресурсов научного исследования = 263085,58 руб.
2. Нормы и нормативы расходования ресурсов	-
3. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	– коэффициент отчислений во внебюджетные фонды – 30 %.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Оценка инновационного потенциала НТИ	– Анализ конкурентных технических решений с позиции ресурсоэффективности и ресурсосбережения
2. Планирование процесса управления НТИ	Построение плана-графика выполнения ВКР, составление соответствующей сметы затрат, расчет цены результата ВКР.
3. Определение ресурсной, финансовой, экономической эффективности	Оценка экономической эффективности использования результатов ВКР
Перечень графического материала *(с точным указанием обязательных чертежей):*	
1. Оценочная карта для сравнения конкурентных технических решений 2. Оценка конкурентоспособности технических решений 3. Матрица SWOT 4. График проведения и бюджет НТИ 5. Оценка ресурсной, финансовой и экономической эффективности НТИ	

Дата выдачи задания для раздела по линейному графику	01.03.2021

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН	Верховская Марина Витальевна	к.э.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ91	Трофимова Анастасия Евгеньевна		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8ИМ91	Трофимова Анастасия Евгеньевна

Школа	Инженерная школа информационных технологий и робототехники	Отделение (НОЦ)	Отделение информационных технологий
Уровень образования	Магистратура	Направление/специальность	09.04.02 Информационные системы и технологии

Тема ВКР:

Разработка комплексной системы управления складом для интернет-магазинов	
Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	Объект исследования: комплексная система, представляющая собой веб-приложение для управления складскими процессами и ресурсами компании. Рабочая зона: офис площадью 45 кв.м, отоплением мощностью 4,4 кВт, со световым потоком 51300 лм , с оборудованием: офисные стулья, столы, принтер, компьютеры, кондиционер, шкаф, светодиодные лампы. Область применения: сфера интернет-магазинов.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности: – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	Законодательные и нормативные документы: – ГОСТ 12.2.032-78 ССБТ – ГОСТ Р 50923-96 – ТК РФ от 30.12.2001 N 197-ФЗ (ред. от 27.12.2018) – ГОСТ 12.0.003-2015 ССБТ – СНиП 23-05-95 – СанПиН 2.2.4.548–96 – СН 2.2.4/ 2.1.8.562-96 – ГОСТ 12.1.038-82 ССБТ – ГОСТ 12.1.004-91 ССБТ – ГОСТ 12.1.006-84 ССБТ – ГОСТ 12.1.003-2014 ССБТ – ГОСТ 17.4.3.04-85 ССОП
2. Производственная безопасность: 2.1. Анализ выявленных вредных и опасных факторов 2.2. Обоснование мероприятий по снижению воздействия	Выявить вредные факторы: – Повышенная или пониженная температура воздуха рабочей зоны; – Недостаточная освещённость рабочей зоны; – Повышенный уровень шума; – Вибрации – Нарушение микроклимата

	– Нервно-психические перегрузки: монотонность труда Выявить опасные факторы: – Поражение электрическим током; – Короткое замыкание; – Статическое электричество.
3. Экологическая безопасность:	Рассмотреть утилизацию компонентов ПК, макулатуры, люминесцентных ламп, периферийных устройств (клавиатура, принтеры, МФУ, веб-камеры, наушники, колонки, телефоны).
4. Безопасность в чрезвычайных ситуациях:	Определить возможные ЧС: пожары в помещении, грозы, ураганы, оползни. Рассмотреть наиболее типичную ЧС: Пожары в помещении, сопровождающиеся эвакуацией рабочего персонала

Дата выдачи задания для раздела по линейному графику	01.03.2021

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ООД	Федоренко Ольга Юрьевна	д.м.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ИМ91	Трофимова Анастасия Евгеньевна		

РЕФЕРАТ

Выпускная квалификационная работа содержит 121 страницу, 32 рисунка, 25 таблиц, 47 источников, 6 приложений.

Ключевые слова: интернет-магазин, складские процессы, система управления складом, WMS, веб-приложение, стек MEAN, JavaScript, HTML, TypeScript, Angular, MongoDB, Express.js, Node.js, REST API, RFID.

Объектом исследования данной работы является создание комплексной системы управления складскими процессами.

Областью применения объекта исследования является сфера малого и среднего бизнеса интернет-магазинов.

Цель работы – проектирование и разработка веб-приложения с функционалом, позволяющим вести учет и визуализировать расположение поступающей на склад продукции с помощью технологии RFID или вручную, интегрировано вести учет сформированных заказов и сопутствующих расходов интернет-магазина, отображать складскую статистику бюджета и товарооборота компании.

В процессе исследования проводились литературный анализ предметной области и анализ существующих конкурентных систем управления складом. На основе сравнения конкурентных решений сформированы функциональные требования к разрабатываемой системе управления складом.

В результате исследования спроектирована и разработана система управления складом в виде веб-приложения с помощью современного стека технологий MEAN (MongoDB, Express.js, Angular, Node.js) с возможностью синхронизации с интернет-магазином через REST API.

Значимость работы заключается в том, что разработанная система позволяет повысить производительность, избежать множество ошибок при ведении учета продукции и снизить эксплуатационные расходы склада за счет возможности кастомизации под индивидуальные нужды пользователя, гибкости в настройке системы, 2D визуализации местоположения продукции на складе и возможности интеграции системы по REST API.

СПИСОК ТЕРМИНОВ, СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

PayPal — крупнейшая дебетовая электронная платёжная система. Позволяет клиентам оплачивать счета и покупки, отправлять и принимать денежные переводы.

Ритейлер — компания, занимающаяся продажей различных товаров конечному потребителю.

WMS (англ. Warehouse Management System, «система управления складом») — информационная система, обеспечивающая автоматизацию управления бизнес-процессами складской работы профильного предприятия.

RFID (англ. Radio Frequency IDentification, «радиочастотная идентификация») — способ автоматической идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или RFID-метках.

SKU (англ. Stock Keeping Unit, «складская учётная единица») — идентификатор товарной позиции (артикул), единица учёта запасов, складской номер, используемый в торговле для отслеживания статистики по реализованным товарам/услугам.

R-LRMS (англ. Radio Frequency Logistics Resource Management System, «Система управления логистическими ресурсами на основе радиочастотной идентификации») — система управления логистическими ресурсами при помощи автоматической радиочастотной идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или RFID-метках.

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

TS (TypeScript) — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

JSON (англ. JavaScript Object Notation)— текстовый формат обмена данными, основанный на JavaScript.

REST (англ. Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы.

API (англ. application programming interface «программный интерфейс приложения, интерфейс прикладного программирования») — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

Angular — открытая и свободная платформа для разработки веб-приложений, написанная на языке TypeScript, разрабатываемая командой из компании Google, а также сообществом разработчиков из различных компаний.

MongoDB — документоориентированная система управления базами данных, не требующая описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных.

Express.js — фреймворк web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API.

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API.

MEAN (аббревиатура от MongoDB, Express.js, Angular.js, Node.js) — набор («стек») серверного программного обеспечения, который, подобно LAMP, используется для веб-разработки, ориентированный на JavaScript: все компоненты стека поддерживают программирование на JavaScript, и серверная и

клиентская часть MEAN-приложений может быть написана на этом языке программирования. За слой хранения в стеке отвечает документоориентированная СУБД MongoDB; основная платформа исполнения — серверная JavaScript-платформа Node.js, серверная разработка реализуется посредством работающего поверх Node.js каркаса Express.js, а разработка интерфейсной, браузерной части веб-приложений — MVC-фреймворком Angular.js.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Количество пользователей Интернета во всем мире увеличивается каждый день. Около 60% населения мира сейчас пользуется Интернетом [1]. По этой причине предприятия начали увеличивать свое присутствие в Интернете. На актуальность Интернет-торговли повлияло увеличение использования клиентами различных интернет-сервисов. Онлайн-покупатели все больше приобретают уверенность в том, что их ждёт хороший опыт онлайн-покупок, они ищут в интернете товары более высокого качества по более низким ценам. Также потребители предпочитают интернет-магазины за широкий выбор товаров [1,2].

Согласно прогнозу Data Insight, к 2024 году при подобной тенденции объемы электронной торговли повысятся на 27% [2]. Следовательно, на данный момент происходит рост актуальности разработки и использования функционала интернет-магазинов. Возникает необходимость мелким и средним торговым предприятиям переходить в онлайн-формат. Им требуется представительство в интернете, склад и система управление складом для эффективного ведения дел.

Однако, согласно исследованиям эффективности внедрения WMS, проводимые Координационным советом по логистике в период с 2015 по 2018 г., большинству компаниям не удается интегрировать существующие WMS в свой рабочий цикл из-за низкой гибкости и возможности кастомизации представленных продуктов [3]. Внедрение большинства WMS позволяет ощутить пользу от внедрения системы через длительное время. В большинстве случаев данная система позволяет вести контроль и собирать статистику процессов, не ускоряя и не оптимизируя сам рабочий процесс. В особенности сложно оценить пользу от внедрения WMS в случае, если бизнес развивается из мелкого в средний. В этом случае ведение складского учета меняется и находится в переходном процессе, что дает низкую пользу от ее интеграции. Вдобавок польза от интеграции еще меньше в случае WMS, в которых отсутствует возможность интеграции системы с интернет-магазином. Функционал модуля 2D визуализации склада и возможность интеграции систем по REST API дает видимое преимущество сразу после момента введения в

эксплуатацию, т.к. позволяет избежать множество ошибок при ведении учета продукции, а также позволяет оптимизировать и ускорить складские процессы по поиску и размещению продукции на складе [2,3].

Немаловажными факторами при выборе систем для ведения бизнеса являются безопасность и надежность. Большинство WMS представленных на рынке предоставляются с закрытым для пользователей исходным кодом, что не дает возможность провести аудит и оценить безопасность и надежность данного ПО [4].

Исходя из представленных данных, а также отсутствия на рынке отечественных WMS с открытым исходным кодом, актуальным является создание отечественной системы WMS в виде веб-приложения с открытым исходным кодом, так как веб-приложения обладают такими преимуществами, как кроссплатформенность, доступность с различных устройств и отсутствие необходимости установки.

Целью работы является проектирование и разработка веб-приложения с функционалом, позволяющим вести учет и визуализировать расположение поступающей на склад продукции с помощью технологии RFID или вручную, интегрировано вести учет сформированных заказов и сопутствующих расходов интернет-магазина, отображать складскую статистику бюджета и товарооборота компании.

Задачами данной работы являются:

- Проведение литературного анализа предметной области.
- Проведение анализа существующих конкурентных систем управления складом.
- Формирование функциональных требований к разрабатываемой системе управления складом на основе сравнения конкурентных решений.
- Выбор стека технологий для создания веб-приложения.
- Проектирование и разработка комплексной интегрируемой системы управления складом.

1 АНАЛИТИЧЕСКИЙ ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Склад играет решающую роль в работе интегрированной цепочки поставок и вносит положительный вклад. Соответствующие операции на складе не только улучшают обслуживание клиентов, но и приносят прибыль. Чтобы обеспечить безупречную работу склада, необходимо продумать планировку и рабочий процесс. Отдел закупок также должен разработать деликатную программу оценки поставщиков и установить хорошие отношения со всеми поставщиками, чтобы обеспечить своевременное и обильное пополнение запасов [4].

Каждое предприятие является звеном в логистической цепочке, которая представляет собой физическую структуру, отражающую логистические услуги, предоставляемые поставщикам и получателям. Его роль заключается в осуществлении деятельности, связанной с логистикой, коммерческими организациями, которые сотрудничают в управлении физическим потоком продуктов и предметов [5].

Внедрение информационной системы управления складом (WMS) позволяет компании достичь эффективной производительности складских операций, необходимой на сегодняшнем рынке.

Системы управления складом (WMS) становятся все более популярной технологией, которую дистрибьюторы и производители полагаются на автоматизацию движения складов, чтобы лучше отслеживать запасы, включая усовершенствования в выполнении унифицированного и эффективного метода для правильного движения инвентаря.

1.1 Анализ алгоритма основных складских операций

Для грамотного проектирования системы управления складом нужно учесть весь цикл бизнес-потока, который протекает в складском хозяйстве. Складские операции играют ключевые функции в цепочке поставок, связывая движение материала между производителем и клиентом [5,6].

Складирование включает в себя ряд действий, связанных с временным получением, хранением, сбором, обслуживанием, мониторингом и отгрузкой товарно-материальных запасов [6].

Фундаментальный процесс складирования включает четыре основных этапа: прием, хранение, сбор и отгрузка хранимых запасов.

Алгоритм основных складских операций [6]:

1. Прием товаров на склад. Получение – это первый складской процесс, включающий планирование и выполнение действий по разгрузке товара, поступившего на склад. Для правильного выполнения этого процесса важно удостовериться в получении нужной продукции, в нужном количестве, в нужном состоянии и в нужное время. Неправильное выполнение этого бизнес-процесса может повлиять на все последующие операции. При правильном получении продукции от поставщиков отфильтровываются поврежденные товары.

2. Размещение и хранение товаров в определенное место в зоне хранения на складе. Размещение является вторым складским процессом и представляет собой перемещение материала из зоны разгрузки в назначенное оптимальное место складского хранения. Эта деятельность, включающая в себя погрузочно-разгрузочные работы, размещение продукта и подтверждающие местонахождение продукта. Неправильное размещение продукции снижает производительность складских операций.

При правильном размещении продукции на складе:

- Эффективное распределение продукции по складскому помещению.
- Максимальное использование складских площадей.
- Товар легче и быстрее находить, отслеживать и извлекать.

3. Комплектование заказов клиентов. Это складской процесс поиска, извлечения и подготовка заказанного товара потребителю. Считается основным и трудоемким видом деятельности складов. Оптимизация комплектования позволит значительно снизить затраты и повысить эффективность работы склада. Неправильное выполнение этой операции влияет на удовлетворенность

клиентов компании. В данном процессе также объединяются выбранные товары в заказ на продажу. Затем продукция подготавливается к отгрузке клиенту.

4. Отгрузка товаров. Отгрузка – это последний складской процесс, который включает планирование и назначение грузовиков для стыковки заказов, упаковку после комплектации и загрузку грузовиков.

1.2 Обзор размещения товаров в складском помещении

Функциональность комплектации в системах управления складом – это основная функциональность, предназначенная для переноса процесса сбора бумаги на беспроводное устройство. Склады бывают разных форм и размеров.

На этапе организации склада важным аспектом является правильное его проектирование, что включает размещение и хранение складских единиц в складском помещении организаций отрасли [6].

По нормам вся складская продукция должна быть промаркирована, по-другому, иметь идентификатор, и рационально размещена по всему помещению, под складские операции должна быть реализована и использована наибольшая часть площади и емкости складского помещения. Также для каждого складского помещения должна существовать схема размещения промаркированных стеллажей и схема размещения складских единиц [6,7].

Для обеспечения хорошего ведения складских операций очень важна планировка склада. Перегрузка склада может повлиять на всю складскую деятельность. Поэтому очень важно управлять эффективностью компоновки склада с точки зрения внутреннего пространства, чтобы потенциальные узкие места не повлияли на производительность склада.

Скорость производства и типы заказов также влияют на планировку склада и, следовательно, на стратегии комплектования. Компании, которые отгружают клиентам поддоны с товарами из одного артикула, будут вести складские операции, значительно отличные от тех, которые отгружают на трейлерах поддоны из разных артикулов (хороший пример - бакалея) [6,7].

Существует полочный способ хранения. Для него характерен сбор неупакованных товаров, которые помещаются в специальные ячейки. Такое хранение очень удобно, поскольку экспозиция размещается на поддонах, которые размещаются на полках, расположенных на любой доступной для механизмов высоте. Внизу можно развернуть товары, выбор которых осуществляется только вручную, а вверху - отправленные на поддоне [7].

На рисунке 1.1 представлена схема управления пространством склада для оптимизации расположения стеллажей, где хранится продукция.

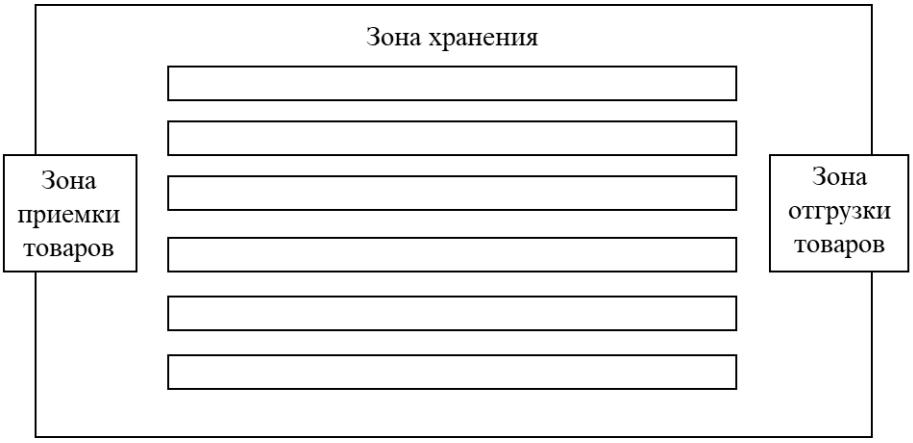

Рисунок 1.1 – Примерная схема склада

Складское помещение в большинстве случаев разделяют на области приема продукции на склад, хранения продукции на складе и отправления продукции со склада [7]. Таким образом, схему складского помещения можно представить в прямоугольной области.

Такая система позволяет экономно использовать помещение, облегчает разгрузку, погрузку и нахождение нужной складской единицы. Необходимо расставить товары на складе таким образом, чтобы получить максимальную вместимость склада, без потери эффективности складских процессов, учитывая параметры хранимого груза.

1.3 Анализ применения технологии RFID для управления складскими операциями

1.3.1 Актуальность применения технологии RFID в складской среде

Вследствие глобального расширения сетей цепочки поставок, которые сами по себе являются ценными каналами обмена информацией, предприятиям необходимо сотрудничать с поставщиками, клиентами или даже конкурентами в разных часовых поясах, и через многочисленные организационные границы для облегчения основных деловых операций, связанных с продажей, производством и доставкой определенного продукта. В этих обстоятельствах задача распределения производственных, транспортных и товарно-материальных ресурсов для удовлетворения спроса становится важной.

В результате высококонкурентной рыночной среды компании постоянно вынуждены расширять свои складские операции в системных приложениях. Необходимо рационально и эффективно распределять складские ресурсы, чтобы повысить производительность и снизить эксплуатационные расходы склада. Следовательно, является важным использовать методы управления данными в реальном времени для точного определения местоположения ресурсов для эффективной поддержки складских операций [8].

В настоящее время сложно обновлять ежедневные операции, касающиеся уровня запасов, местоположения складских единиц в режиме реального времени, используя системы управления складом на основе штрих-кода или вручную. Поэтому актуально использование технологии радиочастотной идентификации RFID для оптимизации получения и обновления данных на складе. [8].

RFID технология широко применяется в различных средах, таких как производство, складирование, розничная торговля, в других различных бизнес-операциях для идентификации, обнаружения и отслеживания любых физических объектов [10]. Хотя это намного дороже, чем технология штрихкодирования, предприятия готовы применять такие методы, чтобы повысить точность сбора данных. Благодаря использованию технологии RFID обеспечивается

возможность автоматического сбора данных и исключается влияние человеческого фактора. На рисунке 1.2, показано, что на основе ввода неверной информации из-за человеческой ошибки о складских единицах и месте их хранения из системы WMS создаются неточные отчеты для сотрудников склада, вследствие чего возникают ненадежные решения для погрузочно-разгрузочных работ [9].

Рисунок 1.2 – Типичные проблемы, возникающие в складских операциях

Таблица 1.1 показывает, что RFID имеет значительные преимущества по сравнению с традиционными системами штрихкодирования. Как видно, у характеристик RFID-оборудования по множествам пунктам превосходит достоинства использования других идентификаторов, а также является более удобной и долговечной, поэтому применение данной технологии с самого начала ведения бизнеса на сегодняшний день является актуальным [9,10].

Таблица 1.1 – Сравнение RFID-технологии со штрихкодированием

RFID	**Штрих-код**
Возможность считывания метки без прямой видимости	Требуется прямая видимость для считывания метки
Одновременное считывание нескольких меток	Невозможность одновременного считывания нескольких меток
Возможность считывания метки в агрессивных средах	Невозможность считывания метки в агрессивных средах, при повреждении метки
Возможность идентифицировать конкретный предмет	Возможность определить только тип предмета
Новая информация может быть перезаписана	Информацию невозможно перезаписать
Возможность автоматического отслеживания, исключая человеческую ошибку	Требуется ручное отслеживание, возможна человеческая ошибка

Как видно, у характеристик RFID-оборудования по множествам пунктам превосходит достоинства использования других идентификаторов, а также является более удобной и долговечной, поэтому применение данной технологии

с самого начала ведения бизнеса на сегодняшний день является актуальным [9,10].

1.3.2 Принцип действия RFID-оборудования

Система RFID, что поясняет рисунок 1.3, обычно включает приемопередатчик, связанную с ним антенну и транспондеры (теги), несущие данные. С пассивными метками считыватель передает маломощный радиосигнал через антенну, который метка принимает через свою собственную антенну для питания, встроенного чип. Используя энергию, которую он получает от сигнала, метка кратко беседует со считывателем для проверки и обмена данными. Как только считыватель получит данные, они могут быть отправлены в контролирующий компьютер и хранится в базе данных для дальнейшей обработки и анализа. Активные теги работают в так же, как пассивный; разница в том, что активные метки передают сигнал через свою антенну, постоянно используя питание от внутренней батареи [9].

Рисунок 1.3 – Внедрение технологии RFID в складской среде

Считыватели RFID собирают два типа данных, а именно статические и динамические данные о ресурсах склада, чтобы визуализировать фактический статус складских операций. Статические данные о ресурсах склада включают в себя местонахождение и количество хранимых артикулов, типы артикулов, доступное пространство для поступающих продуктов. Динамические данные о ресурсах склада включают в себя местонахождение и уровни запасов на каждой

стойке, статус операций по подбору заказов и т. д. С помощью беспроводной сети, то есть сети WIFI, данные о ресурсах склада, которые были собраны передаются и сохраняются в базах данных [9,10].

1.3.3 Классификация и характеристики RFID-оборудования

На существующем рынке доступны два распространенных типа RFID-оборудования, а именно активная RFID-технология и пассивная RFID-технология. Элементы оборудования этих технологий различаются по размеру, стоимости, производительности чтения и областям применения. Наиболее часто используемым RFID-оборудованием на складах является активные (серия Alien2850MHz) и пассивные (серия Alien 9800) RFID-устройства. Использование технологии RFID улучшает видимость складских операций и повышает производительность склада в реальной рабочей практике [9,10].

Таблица 1.2 – Характеристики RFID-считывателей

Спецификация считывателя	Активный считыватель 	Пассивный считыватель 
Частота работы считывателя	2410 MHz – 2471.64 MHz	902.75 MHz – 927.25 MHz
Поляризация антенны	Круговая	Линейная

Все RFID-метки устойчивы к агрессивным средам. Это преимущество позволяет применять технологию RFID, где угодно. Кроме того, все метки имеют лучшую производительность, когда они размещены на том же уровне, что и антенны. На основе метода прецедентного обоснования происходит подбор подходящего оборудования RFID для конкретной складской среды, а также мест, подходящих для установки устройств RFID на складе. После эффективной установки оборудования RFID точность полученной складской информации значительно повышается [9,10].

Таблица 1.3 – Характеристики RFID-меток

RFID-Метки	Активная 	Пассивная 
Источник питания	Работает от внутренней батареи с конечным сроком службы	Не требуют источника питания для работы
Диапазон приема сигнала	На длинные расстояния 0-10 м	На короткие расстояния 0-2 м
Чувствительность к помехам	Менее чувствителен к помехам	Чувствителен к помехам
Скорость передачи данных	Высокая скорость передачи данных	Низкая скорость передачи данных
Количество считываемых меток	Может читать сразу несколько меток	Может читать сразу несколько меток
Местоположение считывателя	Метки можно прочитать без точного прицеливания	Считыватель должен быть нацелен на метку

Согласно характеристикам, показанным в таблице 2, расстояние, на котором активная метка может принимать сигнал, составляет около 10 м, но пассивная метка не может принимать сигнал за пределами расстояния примерно 2 м. Скорость чтения активного устройства RFID лучше, чем у пассивного устройства RFID [10].

Активные RFID-метки имеют отдельный и независимый источник питания. Благодаря этому они показывают лучшие результаты на дальнем расстоянии, могут быть дополнены дополнительными функциями, но при этом имеют больший размер в сравнении с пассивными радиочастотными метками. Активные метки могут передавать сигнал на сотни метров, что значительно расширяет сферу их применения. Обладают собственной памятью и могут хранить значительно больше информации в сравнении с пассивными [10].

Пассивные RFID-метки не требуют источника питания для работы, потому что они работают за счет энергии магнитного поля, которое формирует считыватель. Это значительно упрощает использование данного вида меток за счет их маленького размера и многообразия видов исполнения. Но пассивные RFID-метки имеют ряд недостатков перед активными, такие как низкая скорость передачи данных, считыватель должен быть нацелен на метку, метки чувствительны к помехам и т.д. [10].

2 АНАЛИЗ КОНКУРЕНТНЫХ РЕШЕНИЙ СИСТЕМ УПРАВЛЕНИЯ СКЛАДОМ

2.1 Обзор рынка систем управления складом

Объем мирового рынка систем управления складами будет расти со среднегодовым темпом роста 15,3% с 2021 по 2028 год [3].Растущие прикладные секторы, такие как здравоохранение, производство и розничная торговля, прогнозируется, что во всем мире будет стимулироваться спрос на системы управления складом (WMS) для эффективных операций с целью увеличения объемов производства и удовлетворения растущего потребительского спроса. Пытаясь удовлетворить растущий спрос, логистические компании постоянно работают над преодолением проблем, возникающих из-за колебаний товарных рынков и графиков отгрузки. WMS помогает сократить время выполнения заказа, увеличить скорость доставки продукта и минимизировать затраты на распространение. Программное обеспечение предназначено для выполнения сложных складских операций, а также для решения менее сложных операций с ограниченными ресурсами [11].

Согласно исследованиям по эффективности внедрения систем управления складом, проводимые Координационным советом по логистике с 2015 по 2018 г. показывают, что более 80% складов используют функциональные возможности систем в пределах 15% его функциональности [11,12].

В связи с технологическими и рыночными тенденциями, такими как рост оцифровки, рост автоматизации или использование методов искусственного интеллекта, провайдеры WMS также сталкиваются с проблемой разработки новых подходов и адаптации к рынку. В результате устанавливаются новые функциональные и технологические приоритеты развития.

Большинство пользователей WMS не придерживаются корректной последовательности и критериях выбора программного обеспечения (рисунок 2.1). Согласно исследованиям [12]., потребители WMS приобретают системы, которые

- используются их конкурентами;
- разработаны и представлены наиболее известными и крупными поставщиками таких систем;
- имеют минимальную стоимость приобретения и внедрения.

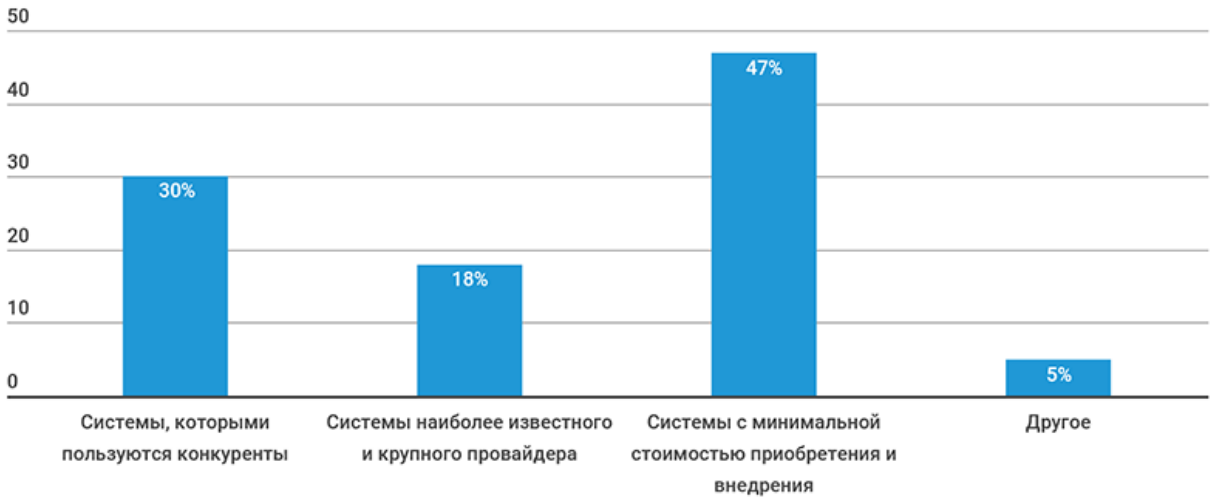

Рисунок 2.1 – Критерии выбора пользователями систем WMS

Как видно на практике, после приобретения и внедрения системы WMS, поспешное принятие решения о выборе системы может повлиять на деятельность компании. Потребители WMS обычно не получают систему для решения собственных задач [12].

2.2 Анализ конкурентных решений

Существующие информационные системы, такие как системы управления складом (WMS), рассчитаны на управление складскими процессами в крупных компаниях, где ведение складского учета товарооборота происходит иным образом нежели в мелком бизнесе, а также такие системы не распространяются в свободном бесплатном доступе [12].

В настоящее время в российском сегменте не существует бесплатных и с открытым исходным кодом систем WMS, то для анализа конкурентоспособности разрабатываемой системы управления складом были рассмотрены функциональные возможности самого известного российского аналога данных систем «РосБизнесСофт WMS». Также для рассмотрены два бесплатных

зарубежных аналога с открытым исходным кодом «OpenBoxes» и «myWMS». Все анализируемые продуты являются веб-приложениями и применимы для малого бизнеса [13].

Веб-приложение «РосБизнесСофт WMS» позволяет вести учет продукции, информировать об остатках, вести контроль запасов. Однако технология RFID в данном ПО не предусмотрена, учет товаров происходит с помощью штрихкодирования. Также недостатком данного продукта является отсутствие визуализации и предложение вариантов расположения товаров на складе. Продукт не распространяется бесплатно, можно приобрести единоразово либо оформить подписку на месяц. Код данного продукта не открыт для доступа [13].

Веб-приложение «OpenBoxes» – это система управления складом с открытым исходным кодом. Эта система управления цепочкой поставок позволяет просматривать историю запасов для любого продукта. Пользователи могут анализировать легкодоступные данные о спросе. Используя стандарты данных, это веб-приложение помогает улучшить отслеживаемость. Он может эффективно обрабатывать исключительные события жизненного цикла продукта при отзыве продукта [13].

Веб-приложение «myWMS» – это программное обеспечение для управления складом с открытым исходным кодом для складов, управляемых вручную. Имеется легко настроить этот бесплатный инструмент для управления складом. Исходный код этого приложения можно использовать и улучшать. Он имеет модульную структуру для WMS. Этот инструмент WMS имеет простой пользовательский интерфейс и поддерживает все необходимые процессы, а также мобильные устройства и сканеры штрих-кода. Он предоставляет модель данных и необходимые услуги для управления запасами, которые помогают в поддержке разработки системы управления складом. Программное обеспечение может предложить индивидуальный и эффективный процесс управления запасами с помощью различных способов интеграции [13].

Таблица 2.1 – Сравнение функциональных возможностей конкурентных WMS с разрабатываемой системой

Критерии сравнения	**Система управления складом**			
	Разрабатываемая система	РосБизнесСофт WMS	OpenBoxes	myWMS
Веб-приложение	Да	Да	Да	Да
Сфера малого бизнеса	Да	Да	Да	Да
Бесплатно	Да	Нет	Да	Да
С открытым исходным кодом	Да	Нет	Да	Да
API для интеграции	Да	Нет	Да	Нет
Экспорт/импорт данных	Да	Только экспорт	Только экспорт	Нет
Статистика склада	Да	Да	Да	Да
2D Визуализация местоположения товаров	Да	Нет	Нет	Нет
RFID система	Да	Да	Да	Нет
Управление заказами	Да	Да	Да	Да
Управление расходами	Да	Да	Да	Да
Управление запасами	Да	Да	Да	Да
Управление расходами	Да	Да	Да	Да
Размещение на складе	Да	Да	Да	Да
Кастомизация страниц и сущностей приложения	Да	Нет	Нет	Нет

Делая выводы по критериям выбора WMS из таблицы 2.1, преимуществами разрабатываемого в исследуемой работе веб-приложения в сравнении с конкурентами являются бесплатное распространение в России с открытым исходным кодом, что позволит начинающим предпринимателям легко и быстро начать работать с системой. Также у конкурентных систем отсутствует возможность импорта накладных продукции, приходящие от поставщиков и

экспорта информации о расходах компании, кастомизация сущностей и страниц приложения, 2D визуализация вариантов расположения товаров на складе, применение современной и популярной RFID системы регистрирования товаров. Также функционал разрабатываемого приложения позволяет интегрировать интернет-магазин в систему управления складом с помощью REST API для автоматического учета в системе управления складом покупок и заказов товаров, произведенными покупателями в интернет-магазине, что является более быстрым и менее затратным способом.

2.3 Формирование функциональных требований к системе в результате анализа конкурентных решений

Проанализировав предметную область и функциональные возможности конкурентов WMS и выявив их недостатки, сформированы функциональные требования, представленные в таблице 2.2, которым должна соответствовать разрабатываемая в данной работе система управления складом.

Таблица 2.2 – Функциональные требования к системе управления складом

Код требования	Требование
F1	Система должна предоставлять возможность управлять складскими заказами
F2	Система должна предоставлять возможность управлять складскими расходами
F3	Система должна предоставлять возможность управлять складскими запасами склада
F4	Система должна предоставлять возможность импорта накладных продукции, приходящие от поставщиков
F5	Система должна предоставлять возможность экспорта информации о расходах компании
F6	Система должна предоставлять добавления и изменения нужных пользователю сущностей
F7	Система должна предоставлять возможность добавления неограниченного количества параметров для сущности
F8	Система должна предоставлять возможность добавления и изменения нужных пользователю страниц приложения
F9	Система должна предоставлять возможность размещения продукции на карте склада вручную и (или) с помощью RFID технологии

Продолжение таблицы 2.2

F10	Система должна предоставлять возможность добавления и изменения карты 2D визуализации складов, пропорциональных параметрам реальных складских помещений, а также их параметров
F11	Система должна предоставлять возможность добавления, изменения и размещения стеллажей по карте склада, а также их параметров
F12	Система должна предоставлять возможность использования технологии RFID для местонахождения продукции на карте склада
F13	Система должна предоставлять возможность просмотра статистики выручки, прибыли, расходов компании
F14	Система должна предоставлять возможность просмотра статистики товарооборота компании
F15	Система должна предоставлять возможность просмотра статистики количества продукции на складе
F16	Система должна предоставлять возможность просмотра статистики даты истечения сроков годности имеющихся продуктов на складе

Опираясь на функциональные требования к разрабатываемой в данной работе системе, было спроектировано и разработано веб-приложение для управления складскими процессами компании.

3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА КОМПЛЕКСНОЙ СИСТЕМЫ УПРАВЛЕНИЯ СКЛАДОМ

3.1 Выбор стека технологий для создания веб-приложения

На сегодняшний день компаниям необходимо усиление своего присутствия в Интернете. Веб-приложение необходимо предприятиям из-за его многофункциональности и преимуществ, связанных с ориентацией на пользователя. Веб-приложение представляет собой сочетание двух скриптов, а именно клиентского и серверного. Служба разработки веб-приложений в первую очередь включает анализ проекта, планирование, разработку.

Веб-приложения предоставляют предприятиям множество преимуществ. Одно из основных преимуществ - сокращение бизнес-затрат. Кроме того, наиболее важным фактором в веб-приложениях является взаимодействие с пользователем. с помощью адаптивного веб-приложения компании могут взаимодействовать со своими клиентами, что увеличивает шансы на конверсию бизнеса. Как правило, предприятия всегда ищут решения для более эффективного управления различными видами деятельности и процессами. В такой ситуации веб-приложение - лучший способ упростить ведение всей бизнес-деятельности [14].

Преимущества веб-приложений:

1. Разработка веб-приложений не зависит от операционной системы пользователей.
2. Доступность на разных устройствах для пользователей.
3. Веб-приложения не требуют установки на устройстве и не занимают много места, в отличие от мобильных.
4. Веб-приложения дают гораздо больше возможностей для интеграции с другими системами, чем настольные приложения.

5. Для функционирования веб-приложений необходимы только браузер и интернет, других специальных программных обеспечений не требуется.

6. При внесении обновлений в веб-приложение они осуществляются на сайте автоматически.

7. Низкие затраты на разработку веб-приложений, по сравнению с настольными или мобильными приложениями.

8. Разработка клиент-серверных приложений обычно обходится дешевле, чем мобильных.

Недостатки веб-приложений:

1. Ограниченные возможности при разработке клиент-серверных приложений, по сравнению с настольными или мобильными.

2. Вероятность возможности вирусным заражением или попытки неправомерного использования личных данных выше, чем у настольных приложений.

3. Отсутствует возможность работать офлайн.

Веб-приложения могут предложить бизнесу ряд преимуществ. Легкость начального развития и роста делает их полезными в управлении, а также обеспечивает гибкость работы для сотрудников [14,15].

Новым и одним из самых перспективным стеком технологий для создания веб-приложений является стек MEAN/ Термин «стек MEAN» относится к набору технологий на основе JavaScript, используемых для разработки веб-приложений. MEAN – это аббревиатура от MongoDB, ExpressJS, Angular и Node.js. Эта группа технологий, которые объединяют технологию стека MEAN в мобильных и веб-приложениях. В последнее время некоторые сложные веб-сайты и веб-приложения (адаптивные) работают на базе стека MEAN. Стек технологий MEAN - одна из самых быстрорастущих платформ разработки стека с открытым исходным кодом, которая помогает разработчикам или командам с помощью популярных инструментов или плагинов сократить время на системное администрирование, а также позволяет быстрее развертывать веб-приложения,

веб-сайты и API-интерфейсы, чтобы сосредоточиться на сложном процессе разработки проекта [15].

Для взаимодействия пользователя с набором данных без ущерба базе данных требуется пользовательский интерфейс. В стеке MEAN в роли такого интерфейса между пользовательским интерфейсом пользователя и данными в базе данных выступает API (интерфейс прикладного программирования). Для создания пользовательского интерфейса в данном стеке используется фреймворк Angular. Angular отображает данные для пользователя в организованном порядке, и пользователи могут взаимодействовать с данными, нажимая кнопки, вводя поля и нажимая на элементы. API создается с использованием NodeJS (Express framework). API получает запросы через конечные точки, такие как / users / getusers или / profile / update, и запускает пользовательские сценарии для получения данных, обновления данных или удаления данных из базы данных. API – это то, что также называется сервером, поскольку его роль заключается в том, чтобы отвечать на запросы и в основном предоставлять данные клиентам. Сервер – это и оборудование, и программное обеспечение, работающие на главном компьютере. Наконец, в роли базы данных применяется СУБД MongoDB. Это база данных без схемы, которая набирает большую популярность. Данная технология также обеспечивает большую организованность разработки веб-приложений. Слияние четырех технологий стека предоставляет доступ к обширному набору инструментов. Открытый исходный код и большое сообщество поддержки делает данный стек привлекательным для разработчика [15].

В Приложении А проведен обзор стека MEAN, приведены преимущества и недостатки каждого из его инструмента.

3.1.1 Обзор базы данных MongoDB

MongoDB – это база данных NoSQL без схемы, которая работает с документами и коллекциями. В отличие от баз данных SQL, MongoDB может хранить документы в формате JSON, и его структура может варьироваться, что подчеркивает высокую масштабируемость системы и снижение сложности развертывания [16].

В реляционных базах данных отношения обеспечивают целостность данных. Но в MongoDB и других базах данных NoSQL нет отношений между документами. Следовательно, документы независимы. Однако есть несколько подходов к моделированию этих отношений между документами.

Возможные подходы к моделированию отношений в MongoDB [16].:

1. Использование ссылок. Этот метод называется нормализацией. Этот подход аналогичен подходу к реляционным базам данных. Манипуляции с данными сосредоточены в одной точке, что способствует согласованности данных. Помимо свойств объекта сущности базы данных, идентификатор объекта сущности передается как ссылка на объект.
2. Использование встроенных документов. Этот метод называется денормализацией. Возможность встроить объект или массив объектов сущности базы данных в объект другой сущности и его соответствующие свойства. Встраивая один документ в другой, мы можем улучшить производительность запросов системы, тем самым сэкономив время и повысив скорость.

Нельзя использовать оба подхода одновременно. У обоих подходов есть свои плюсы и минусы. Следовательно, необходимо найти компромисс между согласованностью и производительностью запросов.

При использовании ссылок документы выглядят аккуратными и менее загруженными по сравнению со встроенными документами. В ситуации, когда необходимо произвести изменения, они делаются в одном объекте. Следовательно, такой подход является последовательным. Этот подход аналогичен подходу к реляционным базам данных.

Однако для загрузки данных может потребоваться выполнить дополнительные запросы, что снизит их производительность. Иногда бывают сценарии, в которых необходимо быстро выполнить запрос. В таких ситуациях использование ссылок не рекомендуется.

При встраивании документов имеется единый документ со всеми деталями обоих документов. Одним из самых сильных преимуществ встраивания документов является то, что можно загружать данные разных сущностей с помощью одного запроса, что повышает их скорость и производительность. Однако документы могут выглядеть перегруженными, особенно если они сами по себе обладают множеством свойств [17].

Учитывая проблемы, рассмотренные выше, при разработке сложных приложений велика вероятность того, что у документов будет ряд свойств (более 50 у каждого). В таких ситуациях ссылок или встраивания будет недостаточно, и это напрямую повлияет на производительность системы.

При гибридном подходе создаются два документа, каждый из которых имеет свой список свойств. Затем необходимые свойства поддокумента извлекаются и встраиваются в пользовательский документ [17].

Таким образом, это избавляет от необходимости хранить все свойства поддокумента во вложенном документе в пользовательском документе. Поскольку вложенный документ имеет исключительно необходимые свойства, этот подход влияет на оптимизированную производительность запросов и согласованность во всей системе [18].

3.1.2 Обзор платформы NodeJS

Node.js – это кроссплатформенная среда выполнения JavaScript с открытым исходным кодом и библиотека для запуска веб-приложений вне браузера клиента. Райан Даль разработал его в 2009 году, а его последняя итерация, версия 15.14, была выпущена в апреле 2021 года. Разработчики используют Node.js для создания серверных веб-приложений, и он идеально подходит для приложений с интенсивным использованием данных, поскольку он использует асинхронное событие -приводная модель [19].

Node.js – это чрезвычайно мощная платформа на основе JavaScript, используемая для разработки приложений онлайн-чата, сайтов потокового видео, одностраничных приложений и многих других веб-приложений, и веб-приложений с интенсивным вводом-выводом. Построенный на движке JavaScript V8 в Google Chrome, он используется как крупными, устоявшимися компаниями, так и недавно созданными стартапами (Netflix, Paypal, NASA и Walmart, и это лишь некоторые из них) [20].

Node.js имеет открытый исходный код и полностью бесплатен, им пользуются тысячи разработчиков по всему миру. Он имеет множество преимуществ, что делает его лучшим выбором по сравнению с другими серверными платформами, такими как Java или PHP.

Express.js – это минималистичный и расширяемый веб-фреймворк, созданный для экосистемы Node.js. Это позволяет вам создать веб-сервер, который будет более читаемым, гибким и удобным в обслуживании, чем вы могли бы создать, используя только библиотеку Node HTTP, которая может быть многословной и сложной даже для самых простых веб-серверов. Express значительно упростит создание веб-сервера! На самом деле трудно даже найти примеры реальных веб-приложений, которые используют только библиотеку Node HTTP, потому что для этого нужно быть садистом [20].

Node.js стал инструментом для разработки серверных и сетевых приложений по нескольким причинам [21].:

1. Node.js действительно быстр: его библиотека, созданная на движке Google Chrome V8 JavaScript, чрезвычайно быстра для выполнения кода.
2. Диспетчер пакетов узлов (NPM): диспетчер пакетов узлов имеет более 50 000 пакетов, поэтому любые функции, необходимые для приложения, можно легко импортировать из NPM.
3. Node.js использует асинхронное программирование: все API библиотеки Node.js являются асинхронными (т.е. неблокирующими), поэтому сервер на основе Node.js не ждет, пока API вернет данные. Сервер вызывает API, и в случае, если данные не возвращаются, сервер переходит к следующему API. Модуль событий Node.js помогает серверу получить ответ от предыдущего вызова API. Это также помогает повысить скорость Node.js.
4. Без буферизации: Node.js значительно сокращает время обработки при загрузке аудио- и видеофайлов. Приложения Node.js никогда не буферизуют данные, а просто выводят данные фрагментами.
5. Однопоточный: Node.js использует однопоточную модель с циклом событий. В результате он может обслуживать гораздо большее количество запросов, чем традиционные серверы, такие как HTTP-сервер Apache.
6. Высокая масштабируемость: сервер Node.js отвечает неблокирующим образом, что делает его хорошо масштабируемым по сравнению с традиционными серверами, которые создают ограниченные потоки для обработки запросов.

Эти причины более чем оправдывают популярность платформы Node.js и то, почему она принимается большим количеством организаций и предприятий. Теперь давайте познакомимся с различными частями Node.js [22].

3.1.3 Обзор фреймворка Angular

Angular - одна из самых популярных платформ JavaScript с открытым исходным кодом. Angular, поддерживаемый Google, используется для создания веб-приложений с богатым набором функций, он делает код JavaScript намного проще и хорошо структурированным [22].

Также он используется для разработки мобильных и настольных приложений. Например, Ionic, мощная платформа для разработки кроссплатформенных мобильных приложений, построена на Angular. Обычно термин «Angular» применяется к Angular 2 и всем последующим версиям [23].

Многие известные веб-приложения, в том числе Eat24, Radio.com, Udacity, Freelancer, Crunchbase, NBA.com, Google Express и другие, созданы на основе Angular. С таким количеством продуктов, созданных при поддержке Angular и сильной поддержкой Google, очевидно, что популярность фреймворка не уменьшится [23].

Преимущества Angular Framework [24].:

- Использование TypeScript. Хотя Angular считается фреймворком JavaScript, он основан на TypeScript. Код, написанный на TypeScript, легко компилируется в JavaScript. По сравнению с JavaScript, сильным преимуществом TypeScript является идентификация ошибок при вводе кода, что позволяет разработчикам исправлять ошибки сразу [24].
- Стабильность. Все версии фреймворка, начиная с Angular 2, совместимы. Это означает, что существует возможность обновить проект до новой версии Angular без изменения кодовой базы. Кроме того, Angular работает на базе Google, что еще больше повышает стабильность фреймворка [24].
- Модульность. Приложение Angular организовано в виде модулей. Это означает, что код состоит из отдельных модулей, отвечающих за различные функции. Существует возможность использования модулей из стандартного пакета Angular, написать свой собственный или интегрировать готовые компоненты, разработанные сообществом Angular. Кроме того, эта структура

допускает «отложенную загрузку», которая загружает только требуемые функции и оптимизирует приложение.

- Согласованность кода. Компонентная природа Angular-приложений обеспечивает упорядоченность базы кода и ее простоту в обслуживании. Компоненты многократно используются и гораздо более удобочитаемы для инженеров, которые не знакомы с кодом.
- Простое тестирование. Модульная система также упрощает тестирование, поскольку отдельные компоненты легче тестировать. Кроме того, существуют разные технологии тестирования Angular. Например, Protractor эффективен для сквозного тестирования, а Karma - для модульного тестирования.
- Кроссплатформенность. Имеется возможность использования Angular для разработки различных приложений, таких как веб, нативные и мобильные приложения. Кроме того, Angular позволяет создавать прогрессивные веб-приложения - приложения, которые загружаются как веб-страницы, но предоставляют дополнительные функциональные возможности, которые обычно предлагаются только нативными мобильными приложениями [24].

Сообщество. Angular чрезвычайно популярен среди разработчиков и имеет большое сообщество. В Интернете можно найти множество сообществ Angular (Stack Overflow, группы Gitter, LinkedIn, каналы Slack и т.д.), а также конференции Angular по всему миру. Это облегчает обмен знаниями [25].

3.2 Обзор работы REST API

Особенностью приложения является его API, позволяющий создать набор ссылок, на которые будет отправляться информация от интернет-магазина во время покупки товара, тем самым предоставляя возможность интегрирования интернет-магазин в систему управления складом с помощью REST API. За счет использования принципов REST API приложение не завит от технологии, которая используется на стороне интернет-магазина, а также особо удобна при смене технологий интернет-магазин, пользователь в этом случае не испытывает дополнительных проблем в настройке новой системы управления складом [26].

Архитектурный стиль REST стал преобладающим выбором для распределенных ресурсов, таких как северный API Software-Defined Networking (SDN). Поскольку службы часто подвергаются частым изменениям и обновлениям, соответствующие API-интерфейсы REST необходимо соответственно изменять и обновлять. Чтобы позволить REST API изменяться и развиваться, не нарушая работы своих клиентов, REST API может быть разработан для облегчения навигации на основе гипертекста и связанных с ней механизмов для работы с изменениями структуры в API [26].

Сеть – это сложная распределенная система, структура, функции, ресурсы и поведение которой могут динамически изменяться. RESTful стал преобладающим выбором для северного интерфейса в SDN, где хорошо спроектированный REST API может быть легко расширяемым и поддерживаемым для управления распределенными ресурсами, такими как сети данных. Расширяемость REST означает, что REST API может одновременно предоставлять различные функции, а также может со временем вносить определенные изменения в эти функции, не нарушая работу своих клиентов [27].

REST API состоит из подключенных ресурсов REST, которые предоставляют различные услуги через единые интерфейсы. В случае SDN он включает сервисы как из плоскости данных, так и из плоскости управления, такие как коммутаторы, маршрутизаторы, подсети, сети, устройства NAT и контроллеры.

Существует множество случаев, когда API-интерфейсы REST должны претерпевать быстрые изменения и обновления, о чем свидетельствуют многие проекты разработки с открытым исходным кодом.

Стандартные методы REST (методы HTTP) [27]:

- GET – получение записей без их изменения.
- POST – создание нового объекта.
- PUT – изменение существующих записей.
- PATCH – изменение идентификатора существующих записей.
- DELETE – удаление записей.

REST API часто использует текстовый формат для представления состояния ресурса в виде набора значимых полей. Сегодня наиболее часто используется сочетание двух технологий, а именно стиля архитектуры REST и схемы JSON. В JSON фигурные скобки используются для иерархической структуры информации в документе [27].

3.3 Реализация технологии RFID в системе управления складом

Данные с RFID-оборудования в систему управления складом приходит в формате JSON посредством HTTP-запросов: get, post, delete, patch согласно архитектурному стилю RESTful API [28] (таблица 3.1).

Таблица 3.1 – HTTP-запросы для получения информации со считывателей

Запрос	**GET**	**POST**	**PUT**	**DELETE**
/	Страница с описанием API	—	—	—
/readers/	Возвращает список настроек для считывателей	Добавляет настройки для считывателей	Заменяет все настройки считывателей переданными	Удаляет все настройки считывателей
/readers/<reader_id>/	Возвращает настройки считывателей	—	Обновляет настройки считывателей	Удаляет считыватели
/readers/<reader_id>/tags/inventory/	Возвращает идентификаторы меток	—	—	—
/readers/<reader_id>/tags/	Возвращает информацию с меток	—	Записывает информацию в метки	Очищает информацию с меток

В качестве ответа на запросы выступают данные в формате JSON.

Общий формат ответа:

```
{
   "response": "какие-то данные"
}
```

Перед началом работы со считывателями нужно получить данные о уже существующих считывателях: так как настройки сохраняются после каждой

манипуляции с ними, то может оказаться так, что RFID-модуль будет инициализироваться с полученными ранее данными [28].

GET /readers/ — получение данных о считывателях.

Пример запроса:

GET /readers/ HTTP/1.1

Примеры ответа:

HTTP/1.1 200 OK

Content-Type: application/json


```
{
   "response": {
      "2": {
         "bus_addr": 1,
         "port_number": 1,
         "state": false
      },
      "3": {
         "bus_addr": 50,
         "port_number": 1,
         "state": true
      }
   }
}
```


Если нет ни одного считывателя:

HTTP/1.1 200 OK

Content-Type: application/json


```
{
   "response": {}
}
```


Подготовить считыватели к работе можно с помощью следующих действий

- Добавить новый считыватель: *POST /readers/*

- Обновить настройки считывателя: *PUT /readers/<reader_id>/*
- Если необходимо заменить все настройками считывателей своими: *PUT /readers/*

Изменить состояния считывателя: подключение/отключение считывателя может производиться с помощью методов *POST /readers/, PUT /readers/<reader_id>/, PUT /readers/* при обязательном наличии ключа "state": true в теле запроса

Работа с метками осуществляется с помощью методов:

- Возвращает идентификаторы меток: *GET /readers/<reader_id>/tags/inventory/*
- Возвращает информацию с меток: *GET /readers/<reader_id>/tags/*
- Записывает информацию в метки: *PUT /readers/<reader_id>/tags/*
- Очищает информацию с меток: *DELETE /readers/<reader_id>/tags/*

Пример запроса:

GET /readers/1/tags/inventory/ HTTP/1.1

Пример ответа:

HTTP/1.1 200 OK

Content-Type: application/json

{

"response": ["meow", "woof"]

}

Пример информации, получаемой со считывателя: его номер, IP и координаты местоположения [28] (рисунок 3.1).

Reader	IP	Coordinates	
		x	y
0001	192.168.100.1	0	0
0002	192.168.100.2	0	10
0003	192.168.100.3	0	20
0004	192.168.100.4	0	30
0005	192.168.100.5	0	40

Рисунок 3.1 – Таблица информации о RFID–считывателях

Пример информации, получаемой с RFID–меток включает в себя идентификатор и количество SKU и каким считывателем было обнаружено [28] (рисунок 3.2).

Tag	Carton	Detected By
urn:epc:id:sgtin:45671234.00001.1000001002	2	0003
urn:epc:id:sgtin:45671234.00001.1000001003	3	0003
urn:epc:id:sgtin:45671234.00001.1000001006	1	0004
urn:epc:id:sgtin:45671234.00001.1000001012	4	0005

Рисунок 3.2 – Таблица информации о RFID-метках

Внешняя система склада состоит из RFID-меток, RFID-считывателей, фиксированной антенны и программного обеспечения, которое поставляется вместе с приобретением RFID-оборудования [28].

3.4 Проектирование взаимодействия бизнес-процессов интернет-магазина для работы с приложением

Особенностью приложения является его API, который позволяет создать набор ссылок, на которые будет отправляться информация от интернет-магазина во время заказа товара, что позволит интегрировать интернет-магазин в систему управления складом. За счет использования принципов REST API приложение не завит от технологии, которая используется на стороне интернет-магазина, а также особо удобна при смене платформы интернет-магазина. Пользователь системы управления складом в этом случае не испытывает дополнительных проблем в ее использовании [26].

В Приложении Б показана схема взаимодействия бизнес-процессов интернет-магазина для работы с приложением. На ней показаны внешние субъекты, с которыми взаимодействует интернет-магазин, в виде желтого блока. Это Поставщики и Клиенты-покупатели товаров. В зеленых блоках приведены бизнес-процессы, а стрелки с подписями показывают поток этих процессов.

На самой толстой выделенной линии внизу схемы показаны основные складские стадии. Другие процессы показывают поток данных, которые происходят внутри основных стадий.

3.5 Архитектура серверной части веб-приложения

Архитектура серверной части веб-приложения реализована на паттерне Model-View-Controller (MVC). MVC или «Модель-Представление-Контроллер» — это архитектурный паттерн проектирования, который включает разделение логики приложения на три взаимосвязанных элемента: Модель, Представление и Контроллер. Каждый из этих компонентов создан для обработки определенных аспектов разработки приложения [29].

Паттерн MVC является одним из распространенных паттернов, применяемых в веб-приложениях и в приложениях на Node.js. MVC ориентирована на надежность и делает процесс разработки более простым и легким, поскольку она состоит из трех отдельных частей.

Паттерн MVC включает ряд компонентов [29]:

- Модели. Модель - это интерфейс базы данных, который позволяет вам взаимодействовать с API базы данных и создавать различные схемы сущностей вашего приложения в базе данных (MySQL, MongoDB), она вызывается из контроллера в зависимости от запроса клиента, если ему нужны сохраненные данные, тогда контроллер будет попросите интерфейс модели предоставить ей необходимые данные.

- Представления. определяет визуальную часть, как данные будут отображаться. Задачу представлений в приложении реализует клиентская часть. Представление – это то, что компилируется и визуализируется в простой HTML, и то, что клиент в большинстве случаев собирается получить в ответ на то, что он запросил (например, для просмотра деталей своего профиля), представление должно использовать механизм шаблонов для выполнения визуализации. Процесс, в котором контроллер снабжает его необходимыми данными (данными из базы данных и клиента), а представление отображает и преобразует все в простой HTML, который браузер может понять и отобразить.

- Контроллеры. Контроллер – это часть, которая занимается обработкой клиентского запроса, обрабатывает HTTP-запрос и возвращает

ответ, ответ может быть либо JSON, если вы вызываете конечную точку API, либо обычную веб-страницу HTML. Контроллеры обычно представляют собой функции обратного вызова, которые соответствуют маршрутизаторам для обработки запросов. Сохранение краткости и удобочитаемости кода - хороший принцип разработки.

Система маршрутизации как дополнительный уровень абстракции, сопоставляющий запросы с маршрутами и осуществляющий соответствующий контроллер.

Таким образом, три разных компонента прекрасно взаимодействуют друг с другом, чтобы обеспечить надежную и гибкую обработку клиентских запросов, будь то для простых задач, таких как доступ к домашней странице или обновление данных его профиля, поэтому создание вашего веб-сервера и архитектуры MVC действительно круто и позволит с множеством вариантов и гибкими мыслями.

Как показано на рисунке 3.3 в общем случае, когда к серверной части приложения приходит запрос, система маршрутизации выбирает нужный контроллер для обработки запроса. Контроллер обрабатывает запрос. В процессе обработки он может обращаться к данным через модели и формировать ответ. Результат обработки контроллера отправляется в клиентскую часть приложения. Ответ представляет собой JSON-файл, используемый для отображения данных на клиентской части.

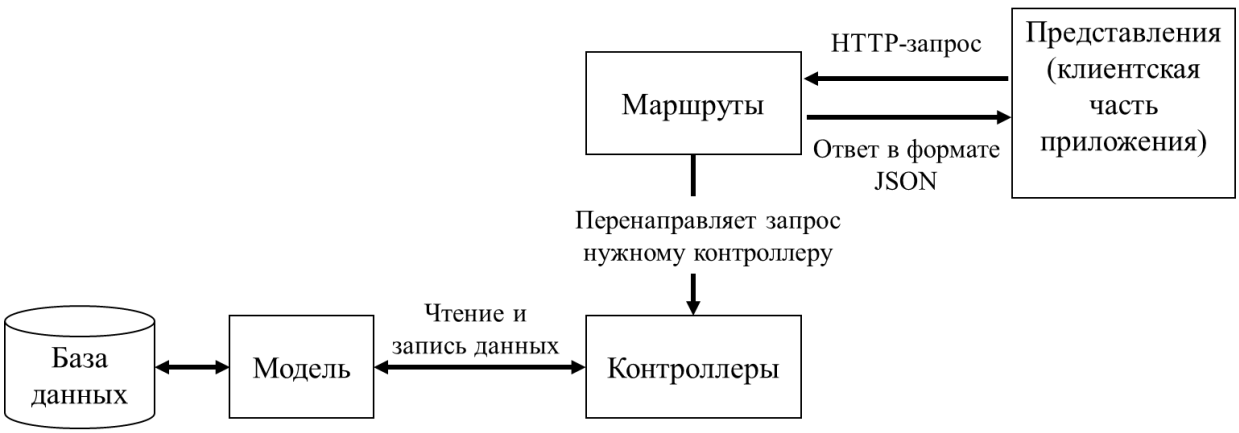

Рисунок 3.3– Схема реализации архитектуры MVC

Проектирование модели представляет собой объявление сущности и создание ее основных полей. Создание сущности происходит благодаря библиотеки Mongoose, которая необходима для взаимодействия с MongoDB.

Контроллеры в паттерне MVC позволяют связать модели и выполняют некоторую логику по обработке запроса. Контроллеры обрабатывают четыре типа http-запросов: get, post, delete, patch.

В запросе post вызывается метод save(), который сохраняет объект в базу данных. В запросе get вызывается метод find(), который получает данные из базы данных и передает их в представление. В запросе delete вызывается метод remove(), который удаляет модель из базы данных по ее id . В запросе patch вызывается метод findOneAndUpdate (), который получает модель по id из базы данных и обновляет ее поля.

3.6 Схема базы данных

MongoDB – это документально-ориентированная база данных с открытым исходным кодом, разработанная для исключительно высокой производительности и разработанная на C ++. Данные хранятся и запрашиваются в BSON в MongoDB, что похоже на JSON. Он имеет динамические схемы и более гибкий, что упрощает и ускоряет интеграцию данных по сравнению с традиционным производством баз данных [20].

База данных приложения на рисунке 3.4 включает в себя такие коллекции (сущности) как Пользователи (Users), Шаблоны (Templates), Страницы (Pages), Продукты (Products), Поставщики (Vendors), Заказы (Orders), Расходы (Expenditures), Склады (Warehouses) и Чеки (Checks).

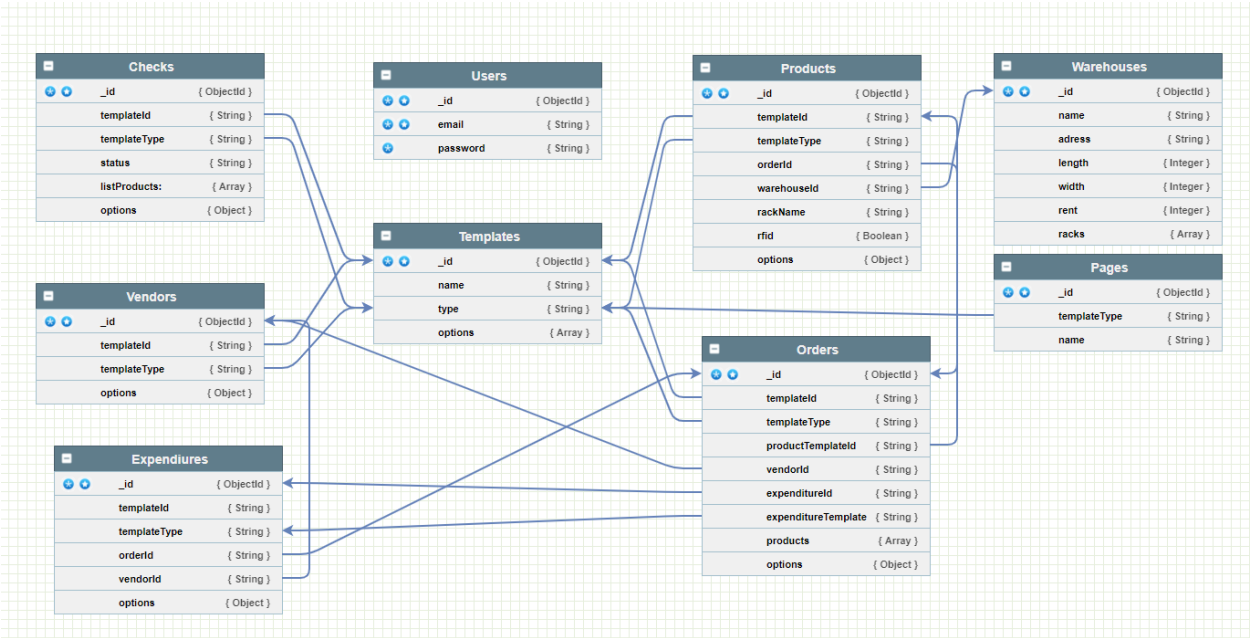

Рисунок 3.4 – Схема коллекций NoSQL базы данных MongoDB

Представленным сущностям базы данных соответствуют интерфейсы на клиентской части веб приложения.

Интерфейс определяет свойства и методы, которые объект должен реализовать. Другими словами, интерфейс – это определение кастомного типа данных, но без реализации, структурный контракт, который определяет, какие свойства объекта должны иметь как имя и как тип [30].

Интерфейсы – это мощные структуры, которые облегчают не только объектно-ориентированное программирование, но и проверку типов в TypeScript. Интерфейс – это группа связанных свойств и методов, которые описывают объект, но не обеспечивают их реализацию или инициализацию. Интерфейс – это виртуальная структура, существующая только в контексте TypeScript. Компилятор TypeScript использует интерфейсы исключительно для проверки типов. После того, как код переведен на целевой язык, он будет удален из интерфейсов [30].

3.7. Структура клиентской части веб-приложения

В клиентской части приложении вся визуальная часть реализуется с помощью компонентов. Компоненты в Angular – это часть интерфейса приложения с собственной логикой и являются основными строительными блоками. Они могут отображать информацию, отображать шаблоны и выполнять действия с данными. Лучшая практика предполагает, что компоненты состоят из трех отдельных файлов: HTML-файл для шаблона, CSS-файл для стилизации и TS-файл для контроля. Следуя этому подходу, код приложения становится более организованным и структурированным. Компоненты делят код на более мелкие, многократно используемые части [25].

За создание компонента отвечает декоратор *@Component()*. Основные свойства объекта, который принимает декоратор [25]:

- *selector* - название компонента;
- *template* (или templateUrl) - HTML-разметка в виде строки (или путь к HTML-файлу);
- *providers* - список сервисов, поставляющих данные для компонента;

styles - массив путей к CSS-файлам, содержащим стили для создаваемого компонента.

Дерево компонентов показано на рисунке 3.6. Все компоненты приложения задекларированы в корневом модуле app.module (Root Module). Корневой компонент AppComponent является точкой входа приложения,

принимает в себя настройки цветовой темы, содержит в себе реализацию компонентов. и импортирует модуль app-routing.module, который предназначен для маршрутизации. По соглашению класс модуля app-routing.module называется AppRoutingModule. В массиве Routes модуля Angular обрабатывает логику того, какой компонент должен отображаться в зависимости от текущего активного URL-пути, который создает маршрутизатор при навигации по этому маршруту, при нажатии на ссылку или копировании URL в адресную строку браузера [25,30].

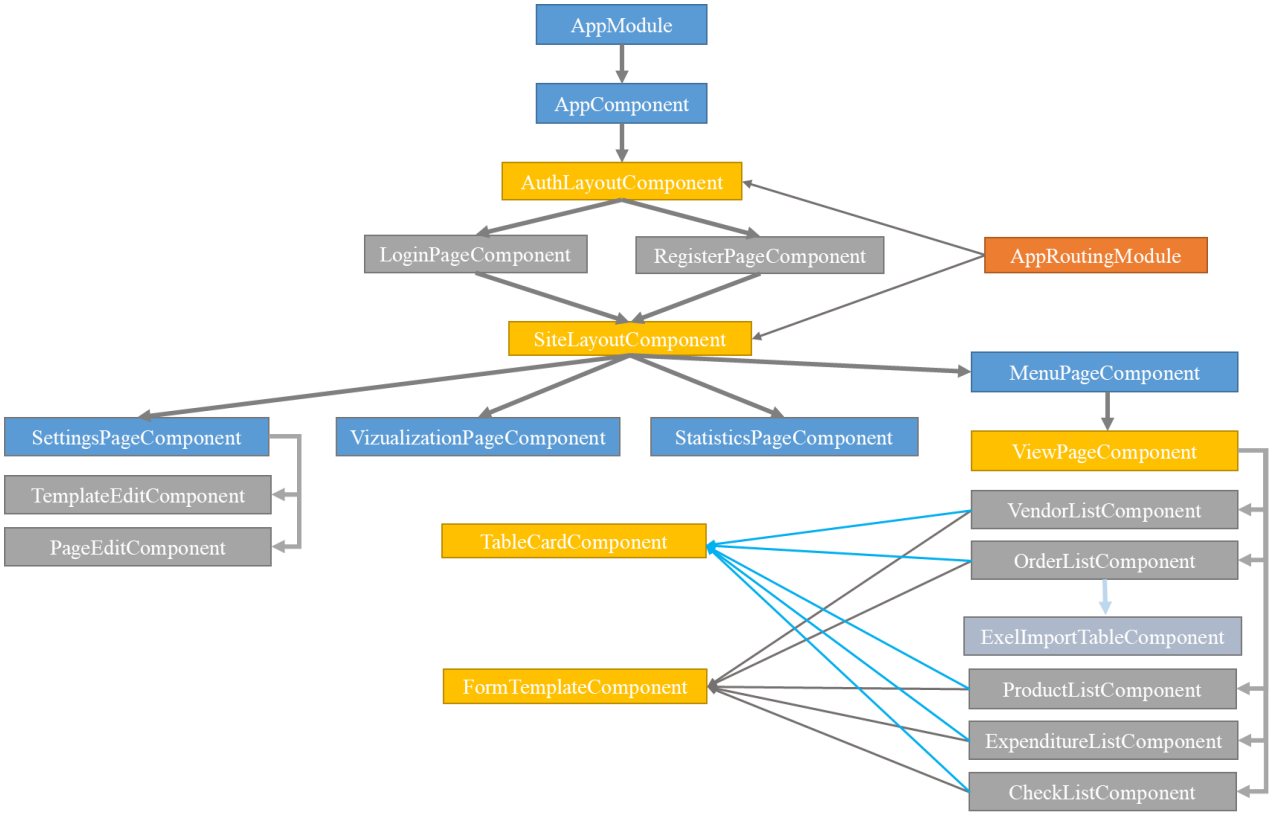

Рисунок 3.6 – Дерево компонентов приложения

Все компоненты реализуют интерфейс метода ngOnInit(), что позволяет управлять процессами при инициализации компонента. Компоненты организованы иерархически это значит, что информация может передаваться между родительскими и дочерними компонентами, между двумя или более дочерними компонентами, а также между двумя или более полностью несвязанными компонентами. Одним из специальных компонентов является корневой каталог приложения AppComponent. Он представляет собой узел верхнего уровня дерева компонентов, и это точка входа, в которой фреймворк

инициализирует приложение. Через него используются остальные компоненты [30].

При входе в приложение отображается пользовательский интерфейс авторизации и регистрации пользователей с помощью компонентов LoginPageComponent и RegisterPageComponent соответственно (рисунки 3.7 и 3.8).

Рисунок 3.7 – Пользовательский интерфейс регистрации пользователей

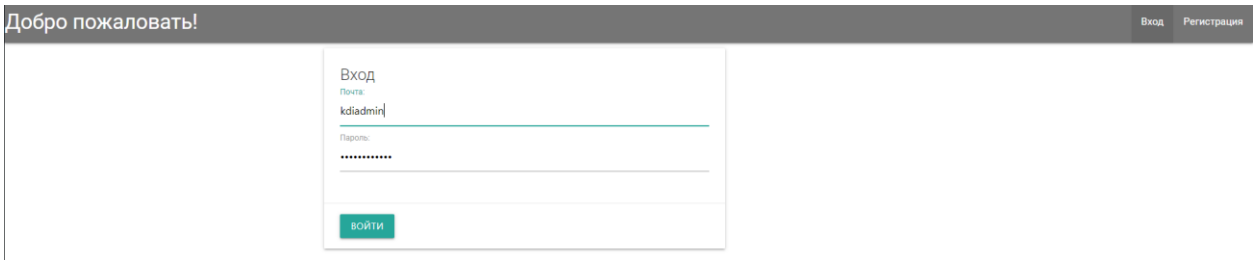

Рисунок 3.8 – Пользовательский интерфейс авторизации пользователей

После аутентификации пользователя приложение запускает компонент SiteLayoutComponent, который включает в себя ссылки на меню, визуализацию, статистику и настройки приложения (рисунок 3.9). Эти ссылки объявлены в модуле AppRoutingModule.

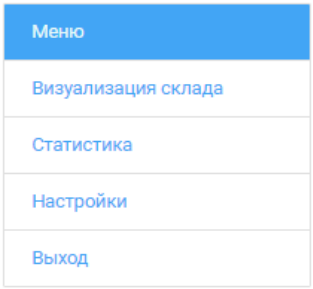

Рисунок 3.9 – Пользовательский интерфейс отображения меню приложения

В меню приложения на рисунке 3.10 отображаются сущности, созданные пользователем путем добавления в настройках приложения шаблонов этих сущностей и добавления их страниц в меню.

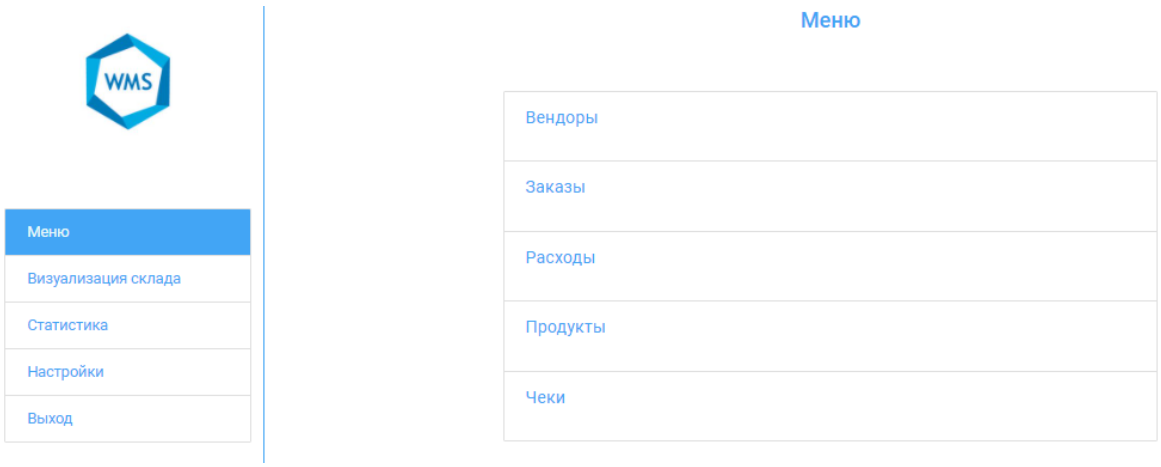

Рисунок 3.10 – Пользовательский интерфейс отображения сущностей приложения

После перехода по ссылкам меню, открывается пользовательский интерфейс отображения данной сущности. Этот фрагмент отображения включает в себя такие компоненты, как TableCardComponent и FormTemplateComponent.

Компонент TableCardComponent включен в компоненты ListComponent и отображает список всех сущностей продукта, вендора и т.д. в виде таблицы-карточки с применением соответствующего шаблона.

Компонент FormTemplateComponent включен в компоненты PageComponent и отображает формы для создания и редактирования сущностей с дополнительными параметрами, к примеру, валидация, тип поля и тд.

В приложении В представлен TypeScript код компонента VendorListComponent, отображающего список всех экземпляров сущности «Поставщик».

При загрузке страницы сущности в методе ngOnInit() флаг loading переходит в состояние true, тем самым показывая пользователю экран загрузки во время получения данных, подготовки и отображения страницы. Затем происходит вызов сервиса и подписка на обрабатывание GET-запроса функции

fetch() для того чтобы загрузить всех вендоров по шаблону типа 'vendor' с данными из базы данных.

3.7.1 Реализация шаблонов для создания сущностей приложения

Важной функциональной особенностью приложения и основным элементом кастомизации является использование шаблонов для отображения, создания и редактирования полей сущностей базы данных. Шаблоны помогают расширять сущности под нужды, пользователя. Например, пользователь приложения может добавить необходимые поля продукту, в соответствии накладной, поступившей от поставщиков.

Интерфейс шаблона Template на рисунке 3.11 содержит две группы свойств:

- общие свойства для всех сущностей такие как id, тип type, который определяет к какой сущности относится шаблон, поле название шаблона name,
- различное свойство для каждой сущности в виде массива объектов options: Array, который содержит в себе различные для каждого шаблона поля.

У каждого объекта массива options присутствуют свои параметры, которые описаны в интерфейсе TemplateItem: название поля name, например, «email», параметр обязательного заполнения поля reqiered, параметр типа заполняемого поля для проверки валидации validators, и параметр show, который указывает будет ли отображаться поле на странице сущности.

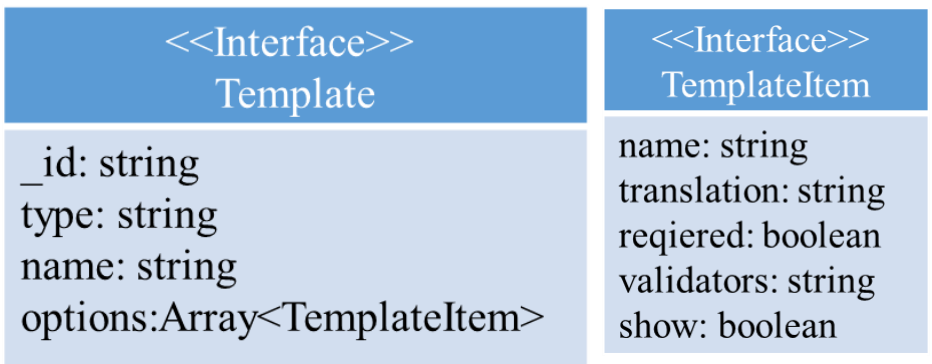

Рисунок 3.11 – Интерфейсы Template и TemplateItem

Для сущности можно создать пользовательский интерфейс (отдельную страницу), которая будет отображаться в меню приложения. Например, пользователь может создать шаблон «Водители» и назначить ему страницу. Таким образом, у пользователя появляется возможность вести учет водителей на отдельной странице. Пользовательский интерфейс отображения сущности отображает информацию с помощью компонентов TableCardComponent и FormTemplateComponent, в которых заключена логика применения шаблонов. Таким образом, выполняется методология повторного использования кода (Code Reuse) и его универсальности.

Компонент TableCardComponent используется в компонентах ListComponent с помощью тега <app-table-card>. Данный компонент отображает сущность «Продукт», «Заказ» и т.д. в виде таблицы-карточки с применением соответствующего шаблона.

В приложении Г представлен HTML код шаблона App-table-card, который отображает в пользовательском интерфейсе приложения таблицу-карточку сущности. В первой ячейке данных таблицы находятся элементы массива showNames, которые отображают названия всех полей объектов options шаблона, во второй ячейке – элементы массива showData, который хранит значения полей, которые присвоены этим объектам и хранятся в базе данных.

В методе ngOnInit() компонента шаблона App-table-card реализован цикл записи названия полей объектов options шаблона и их значений в массивы showNames и showData соответственно, как представлено в приложении Д TypeScript кода шаблона.

Пример применения шаблона App-table-card показан на рисунке 3.15 для поставщика с полями Name, Phone Number, Adress и Email объектов options и их значениями напротив соответственно.

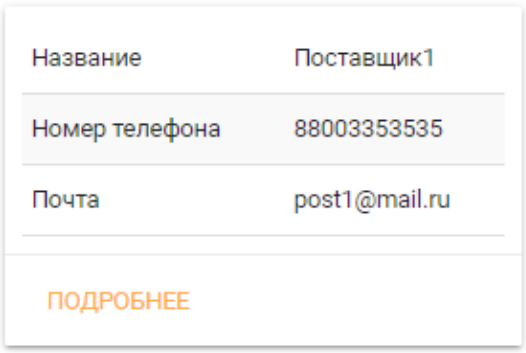

Рисунок 3.12 – Таблица-карточка поставщика

Компонент FormTemplateComponent виспользуется в компонентах PageComponent с помощью тега <app-form-template>. Этот компонент отображает форму для создания и редактирования сущностей с дополнительными параметрами, к примеру, валидация, тип поля и тд.

В случае, если поле сущности имеет дополнительный параметр reqiered: false, т.е. оно не обязательное для заполнения, то на странице не будет никаких уведомлений и значение поля может отображаться пустым при сохранении данных. Если поле сущности имеет дополнительный параметр reqiered: true, т.е. оно обязательное для заполнения, то на странице будет отображаться текст-уведомление «should not be empty» и символ * рядом с названием поля, также пока значение этого поля не будет заполнено, не произойдет сохранение данных.

Названия полей хранятся в поле options.name шаблона, с помощью них можно обратиться к соответствующему значению отображаемой сущности следующим образом: data.options[item.name], которые проходят валидацию в методе ngOnInit() компонента шаблона App-form-template, что представлено в приложении E TypeScript кода шаблона.

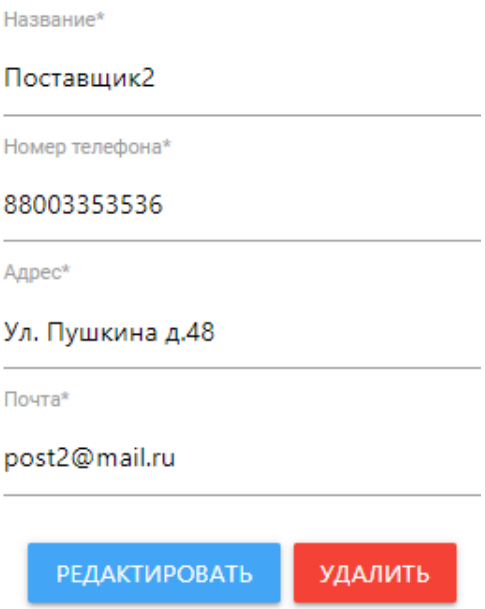

Рисунок 3.13 – Пользовательский интерфейс с формой редактирования сущности «Поставщик»

Например, при редактировании сущности «Поставщик» на рисунке 3.18 отображается форма с соответствующими полями выбранного шаблона сущности для заполнения данных этих полей сущности «Поставщик» с применением валидации заполнения данных, а также проверки на обязательность заполнения поля (параметр reqiered).

3.7.2 Пользовательский интерфейс настроек сущностей

В пользовательском интерфейсе настроек сущностей приложения реализован функционал создания и редактирования шаблонов сущностей и страниц для них (рисунок 3.14).

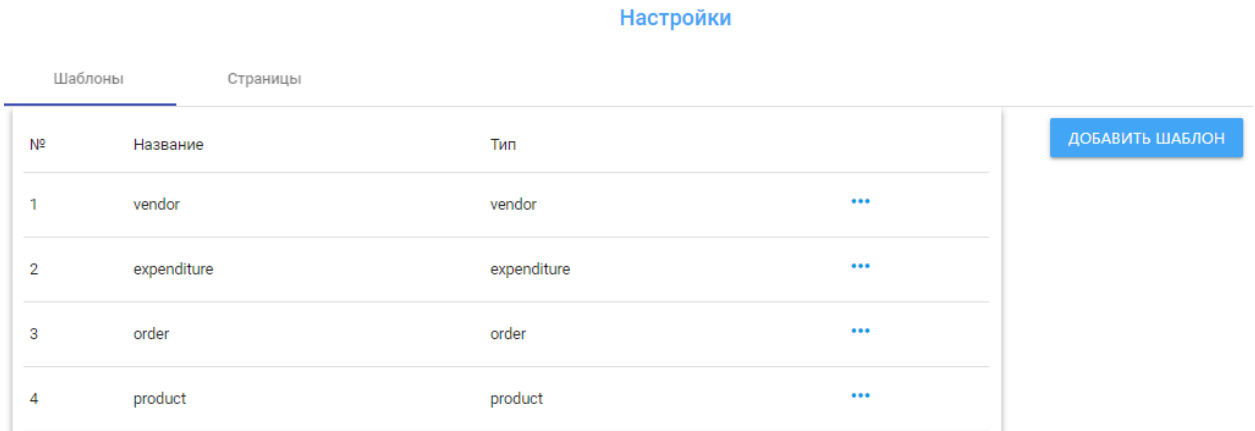

Рисунок 3.14 – Пользовательский интерфейс настроек шаблоноы

При создании шаблона сущности указываются обязательные поля Тип и название шаблона, а также имеется возможность добавления неограниченного количества параметров для данной сущности (рисунок 3.15).

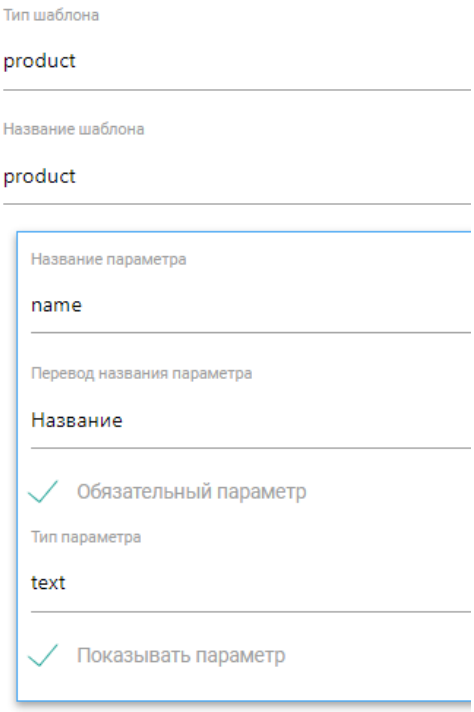

Рисунок 3.15 – Создание шаблона сущности

В параметре сущности требуется указать название этого параметра на английском языке для возможности его использования в TS-коде, его перевод на русском языке, отметить в checkbox будет ли параметр обязательный для заполнения и отображения в пользовательском интерфейсе карточки сущности, указать тип заполняемого поля для проверки валидации.

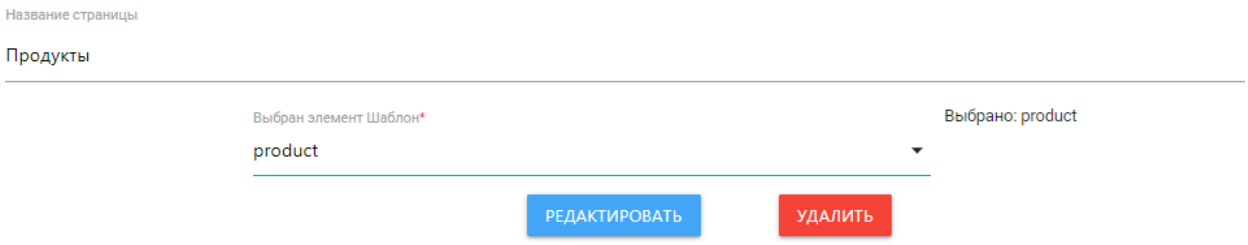

Рисунок 3.16 – Создание страницы для сущности

При создании новой страницы сущности требуется указать название страницы, которая отобразится на странице Меню, и выбрать уже созданный

шаблон, по которому будут отображать экземпляры сущности на данной странице (рисунок 3.16).

3.7.2 Пользовательский интерфейс отображения сущностей

В пользовательском интерфейсе «Меню» выводятся ссылки на страницы сущностей, созданные пользователем в пользовательском интерфейсе «Настройки» (рисунок 3.17).

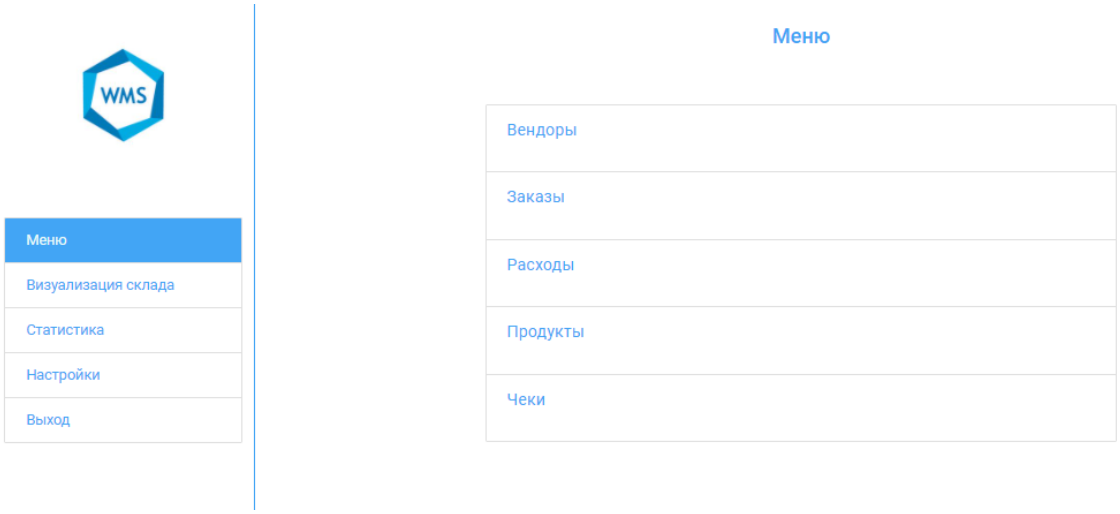

Рисунок 3.17 – Пользовательский интерфейс отображения сущностей

При переходе на страницу сущности (рисунок 3.18) отображаются экземпляры данной сущности в виде таблиц-карточек, которые отображаются при помощи компонента TableCardComponent.

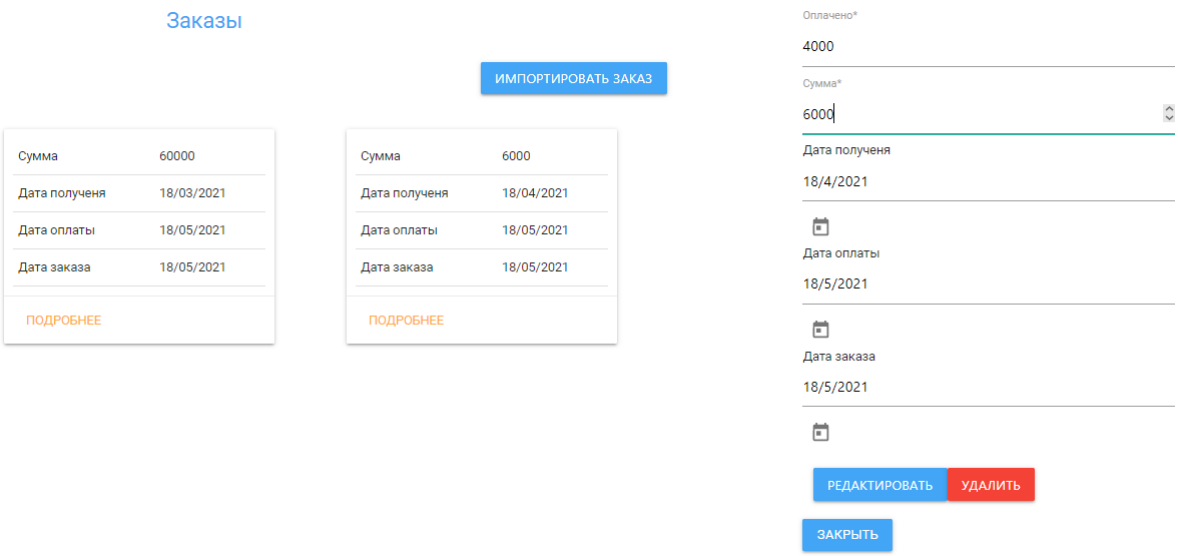

Рисунок 3.18 – Пользовательский интерфейс сущности Заказы

При желании увидеть все параметры экземпляра сущности, открывается форма редактирования/создания сущности, которая отображается при помощи компонента FormTemplateComponent.

В случае создания нового экземпляра сущности на его странице, сначала появляется выбор созданного шаблона для страницы этой сущности, а затем появляется форма заполнения с параметрами, соответствующими выбранному шаблону.

3.7.3 Пользовательский интерфейс визуализации склада

В пользовательском интерфейсе «Визуализация» склада реализован функционал отображения визуальной схемы склада, пропорциональной параметрам реального складского помещения с помощью элемента HTML5 canvas, предназначенного для создания растрового двухмерного изображения при помощи скриптов на языке JavaScript.

Функционал приложения позволяет создать неограниченное количество сущностей индивидуальных складов и индивидуальное размещение на них складских стеллажов. Интерфейсы склада и стеллажа показаны на рисунке 3.19.

<<Interface>> Warehouse	<<Interface>> Rack
_id: string name: string adress: string length: number width: number rent: number racks: Array<Rack>	name: string length: number width: number setY: number setX: number readers: Array<any> products: Array<Product>

Рисунок 3.19 – Интерфейсы склада Warehouse и стеллажа Rack

При создании и редактировании склада, как показано на рисунке 3.20, присутствует возможность установления соответствующих ему параметров

ширины и длины складского помещения. Периметр схемы склада отображается пунктирной линией.

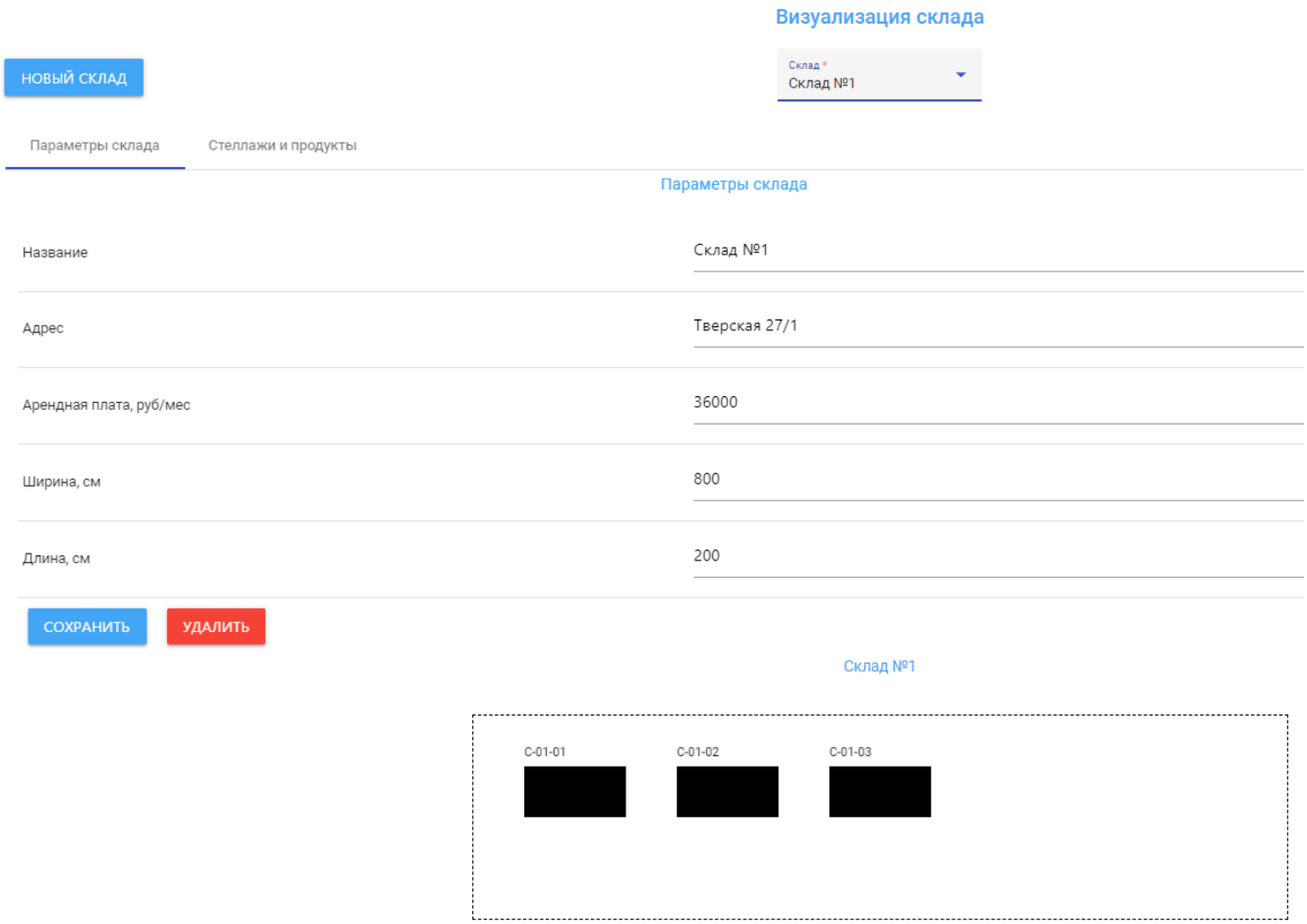

Рисунок 3.20 – Пользовательский интерфейс визуализации интерфейс создания, редактирования и визуализации склада и его параметров

При создании и редактировании конкретного стеллажа склада, как показано на рисунке 3.21, присутствует возможность установления соответствующих ему параметров и идентификаторов RFID считывателей стеллажам.

При импорте заказа можно для местоположения каждого конкретного продукта имеется возможность выбрать склад и стеллаж вручную либо можно выбрать технологию RFID, при которой на сервер приложения с сервера ПО для регистрирования меток RFID будет приходить запрос получения местоположения продуктов путем сравнения Barcoda продукта с меткой и

получение идентификатора считывателя, который зарегистрировал данную метку.

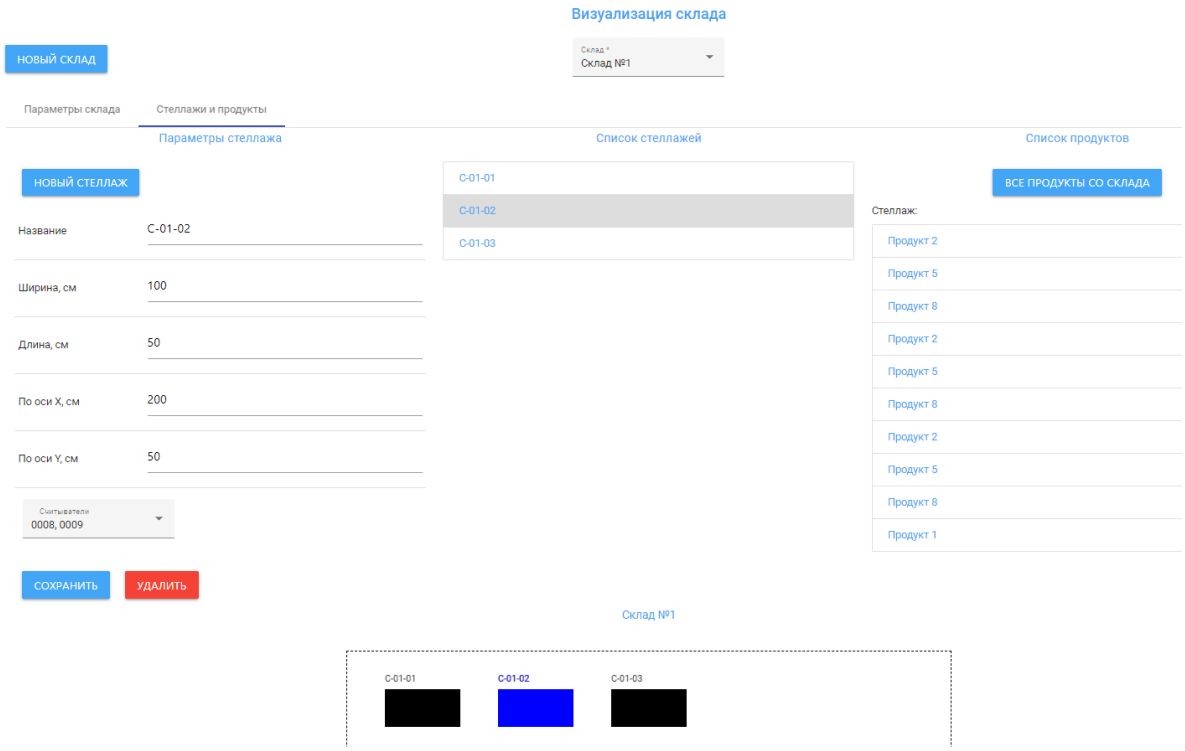

Рисунок 3.21 – Пользовательский интерфейс создания, редактирования и визуализации стеллажа склада и его параметров

Таким образом, при выборе стеллажа показывается его местоположение на складе, а также список продуктов, которые зарегистрировали считыватели этого стеллажа. И наоборот, можно получить список всех имеющихся продуктов на складе, и выбрав любой, отображается в каком стеллаже на складе оно присутствует.

3.7.4 Пользовательский интерфейс статистики

В пользовательском интерфейсе «Статистика» реализован функционал отображения статистических графиков с помощью элемента HTML5 canvas и библиотеки для визуализации данных Chart.js.

Как показывает рисунок 3.22, для реализации графика расходов компании ведется подсчет общей суммы заказов у поставщиков за конкретный период времени. Для реализации графика выручки компании ведется подсчет общей суммы чеков покупателей за конкретный период времени. Для реализации

графика прибыли компании ведется подсчет разницы общей суммы заказов у поставщиков и общей суммы чеков покупателей за конкретный период времени.

Рисунок 3.22 – График расходов, выручки и прибыли компании

Как показывает рисунок 3.23, для реализации графика учета имеющихся продуктов на сладе компании ведется подсчет общей суммы продуктов, числящаяся на складе в конкретный период времени. Для реализации графика учета дата выхода срока годности продуктов на сладе ведется подсчет общей суммы продуктов, у которых выходит срок годности в конкретный период времени.

Рисунок 3.23 – График учета имеющихся продуктов на сладе и дата выхода срока годности продуктов

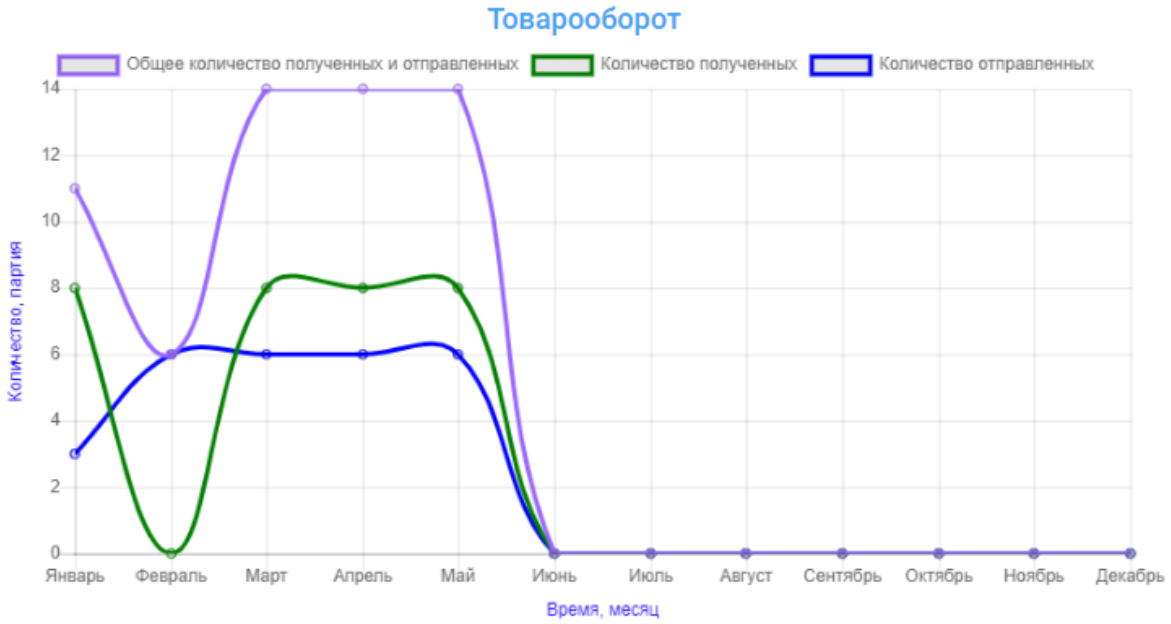

Рисунок 3.24 – График товарооборота компании

Как показывает рисунок 3.24, для реализации графика учета товарооборота компании ведется подсчет общей количества полученных от поставщиков и отправленных покупателям продуктов в конкретный период времени.

3.7.5 Пользовательский интерфейс импорта заказов

В пользовательском интерфейсе заказов реализован функционал импорта накладной заказа от поставщика товаров из excel-файла. Excel-накладная должна включать в себя две страницы: первая с параметрами заказа, вторая с параметрами продуктов. Правило заполнения файла: в первой строке должны быть перечислены параметры, в следующих строках идут значения параметров. Для продуктов количество строк, начиная второй совпадает с количеством продуктов.

Перед импортом, как показано на рисунке 3.25, пользователь должен выбрать шаблоны для поставщика, заказа, расхода и продуктов, которые он создал ранее в Настройках.

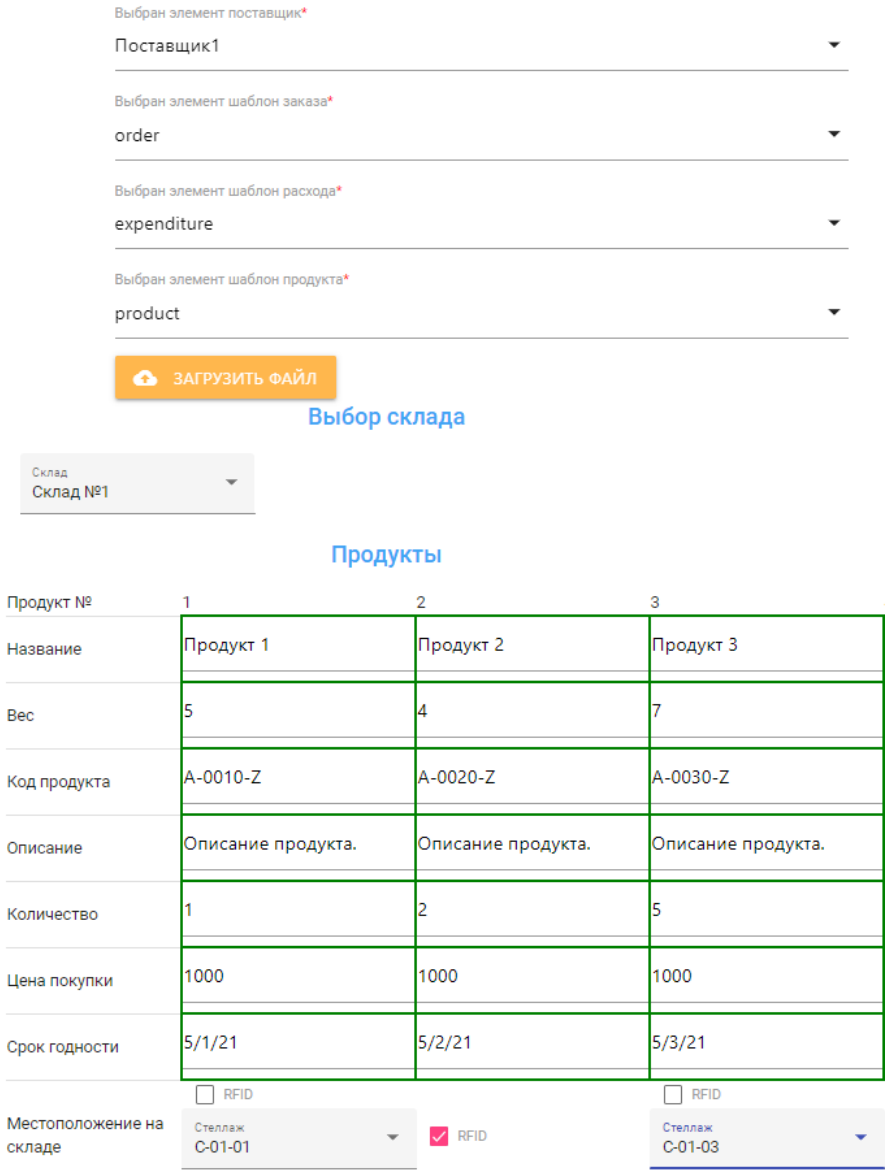

Рисунок 3.25 – Создание импорта заказа из файла excel-формата

При импорте заказа имеется возможность выбрать склад и стеллаж вручную либо можно выбрать технологию RFID для местоположения каждого конкретного продукта.

Алгоритм импорта представлен на рисунке 3.26.

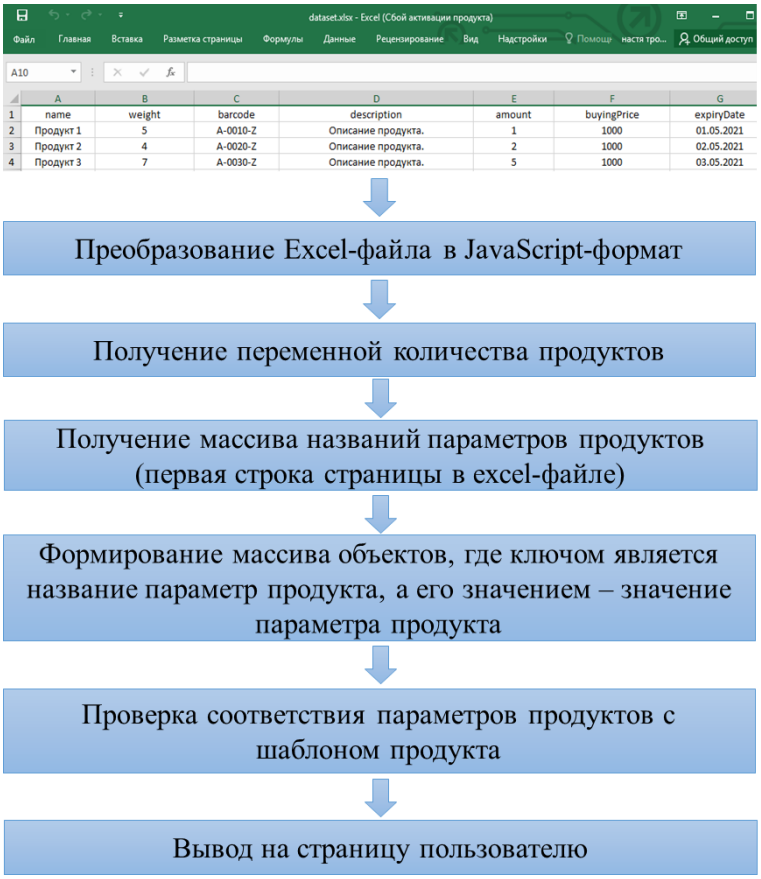

Рисунок 3.26 – Алгоритм импорта

После проверки накладной на соответствие правилам заполнения excel-файла, TS-функция в приложении преобразует его в JavaScript-формат. Затем происходит получение переменной количества продуктов. После этого функция импорта реализует получение массива названий параметров продуктов, которые записаны в первой строке страницы excel-файла. Затем происходит формирование массива объектов, где ключом является название параметр продукта, а его значением – значение параметра продукта. Далее идет проверка соответствия параметров продуктов с шаблоном продукта. Последним действием функция импорта выводит все параметры продуктов и заказа в пользовательский интерфейс пользователю, где он может перед сохранением проверить и редактировать параметры. После сохранения импорта приложение создает экземпляры сущностей расхода, заказа и продуктов.

3.7.6 Пользовательский интерфейс экспорт расходов

В пользовательском интерфейсе расходов реализован функционал экспорта всех существующих расходов в excel-файл. Алгоритм экспорта представлен на рисунке 3.27.

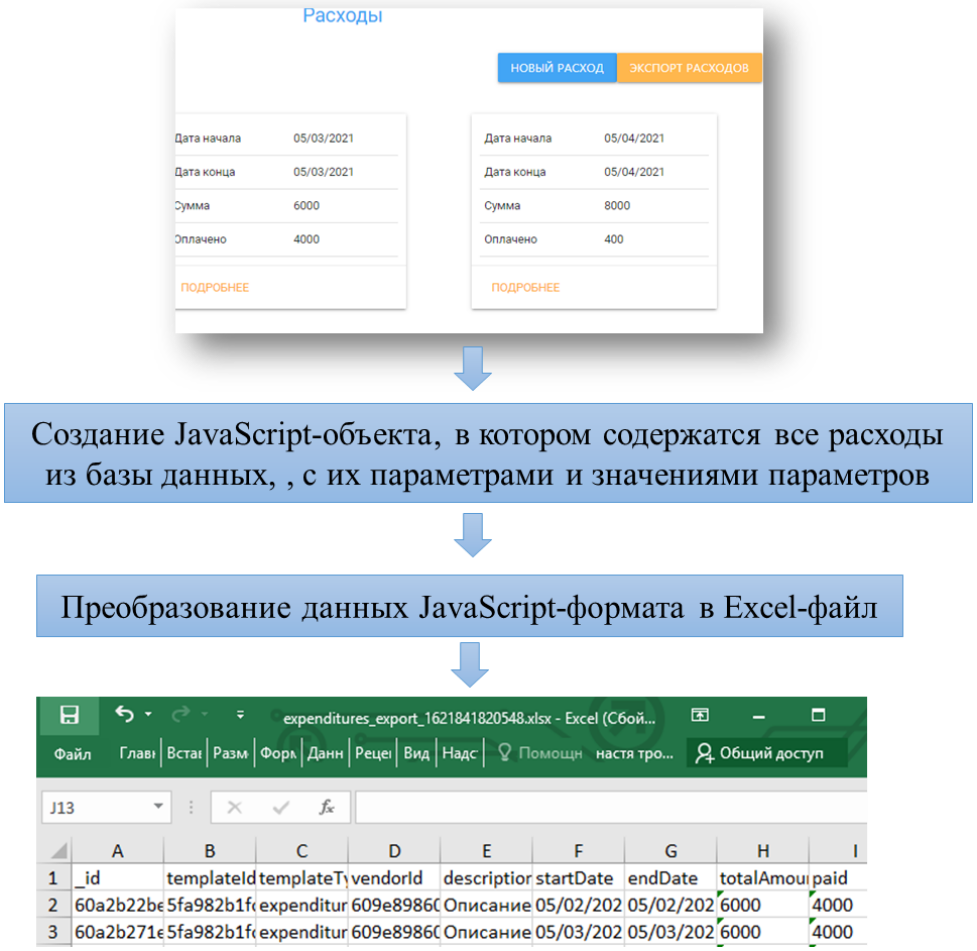

Рисунок 3.27 – Алгоритм экспорта расходов в файл excel-формата

В функции экспорта реализовано создание JavaScript-объекта, в котором содержатся все расходы из базы данных, с их параметрами и значениями параметров. Затем из JavaScript-объекта формируется excel-файл. Также в функции присутствуют присвоение названия excel-файлу, назначение ему расширения .xlsx, присвоение названия странице файла. С помощью FileSaver происходит сохраниние excel-файла на компьютер пользователя.

4 ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ

В настоящее время перспективность научного исследования определяется не столько масштабом открытия, оценить которое на первых этапах жизненного цикла высокотехнологического и ресурсоэффективного продукта бывает достаточно трудно, сколько коммерческой ценностью разработки. Оценка коммерческой ценности разработки является необходимым условием при поиске источников финансирования для проведения научного исследования и коммерциализации его результатов.

Целью раздела является определение перспективности и успешности научно-исследовательского проекта, разработка механизма управления и сопровождения конкретных проектных решений на этапе реализации.

4.1 Предпроектный анализ

4.1.1 Анализ конкурентных технических решений

Проектом данной научно-исследовательской работы является комплексная система, представляющая собой веб-приложение для управления складскими операциями и ресурсами интернет-магазинов. Веб-приложение позволяет вести учет и визуализировать графически расположение поступающей на склад продукции и имеющихся складских единиц с помощью технологии RFID, вести учет сформированных заказов и сопутствующих расходов интернет-магазина, отображать финансовую статистику бюджета компании, а также позволяющим интегрировать интернет-магазин в систему управления складом с помощью REST API.

В настоящее время в российском сегменте не существует бесплатных и с открытым исходным кодом систем WMS, поэтому для анализа конкурентоспособности разрабатываемой системы управления складом были рассмотрены функциональные возможности самого известного российского аналога данных систем «РосБизнесСофт WMS» (К1). Также рассмотрен

бесплатный зарубежный аналог с открытым исходным кодом «myWMS» (К2). Все анализируемые продуты являются веб-приложениями и применимы для малого бизнеса.

Критерий оценки в таблице 2 Простота использования заключается в степени сложности и понятии использования приложения пользователями. Интеграция предполагает наличие функционала интегрирования интернет-магазина в систему управления складом с помощью REST API для автоматического учета в системе управления складом покупок и заказов товаров, произведенными покупателями в интернет-магазине, что является более быстрым и менее затратным способом. Доступность подразумевает свободный, бесплатный и открытый доступ продукта в сети Интернет. Критерий функциональность заключается в присутствии необходимого функционала приложения для начинающего малого бизнеса, и отсутствии лишних модулей для данной сферы, которые нагружают и усложняют использование системы. Внедрение подразумевает возможность развертывания системы на сервере, облаке или ПК. Производительность и скорость работы зависят от размера приложения. Гибкость настройки приложения подразумевает возможность кастомизации функционала под конкретного пользователя приложения.

Таблица 4.2 – Оценочная карта для сравнения конкурентных технических решений (разработок)

Критерии оценки	**Вес критерия**	**Баллы**			**Конкуренто-способность**		
		$Б_ф$	$Б_{к1}$	$Б_{к2}$	$К_ф$	$К_{к1}$	$К_{к2}$
1	2	3	4	5	6	7	8
Технические критерии оценки ресурсоэффективности							
1. Простота использования	0,2	5	3	4	1	0,60	0,80
2. Интеграция	0,1	4	4	1	0,40	0,40	0,10
3. Доступность	0,2	5	1	4	1	0,20	0,80
4. Функциональность	0,1	4	5	3	0,40	0,50	0,30
5. Простота внедрения	0,1	5	2	3	0,50	0,20	0,30

Продолжение таблицы 4.2

6. Производительность и скорость работы	0,2	5	5	3	1	1	0,60
7. Гибкость настройки приложения	0,1	5	2	4	0,50	0,20	0,40
Итого	**1**				**4,80**	**3,10**	**3,30**

Анализ конкурентных технических решений определяется по формуле:

$$К = \sum В_i * Б_i \quad (1)$$

где К – конкурентоспособность научной разработки или конкурента;

$В_i$ – вес показателя (в долях единицы);

$Б_i$ – балл i-го показателя в соответствии 5-бальной шкалой.

Согласно расчету, конкурентоспособность исследуемого в данной работе материала выше, чем у продуктов «РосБизнесСофт WMS» и «myWMS». Это связано с тем что преимуществами разрабатываемого в исследуемой работе веб-приложения являются бесплатное распространение в интернете с открытым исходным кодом, что позволит начинающим предпринимателям легко и быстро начать работать с системой, импорт накладных, приходящие от поставщиков и экспорт информации о расходах компании, предложение и 2D визуализация вариантов расположения товаров на складе, применение современной и популярной RFID системы регистрирования товаров, кастомизация полей и страниц приложения, модульность системы и прогнозирование спроса по статистике эффективности компании по месяцам.

4.2 Планирование управления научно-техническим проектом

4.2.1 План проекта

В данном разделе необходимо составить перечень этапов и работ в рамках проведения научного исследования, провести распределение исполнителей по видам работ. Примерный порядок составления этапов и работ, их длительность, распределение исполнителей по данным видам работ приведен в таблице 3. Распределение исполнителей: И – инженер (дипломник), Р – научный руководитель работы.

Таблица 4.3 – Календарный план проекта

Код работы	Название	Длительность, календарные дни	Дата начала работ	Дата окончания работ	Состав участников
1	Составление технического задания работы	4	08.02.2021	11.02.2021	Р,И
2	Анализ предметной области	14	12.02.2021	25.02.2021	И
3	Исследование существующих решений	6	26.02.2021	03.03.2021	И
4	Обзор стека технологий для создания веб-приложения	12	04.03.2021	15.03.2021	И
5	Проектирование архитектуры веб-приложения	21	16.03.2021	05.04.2021	И
6	Разработка веб-приложения	36	06.04.2021	11.05.2021	И
7	Анализ результатов работы	3	12.05.2021	14.05.2021	Р, И
8	Подготовка к защите диплома	15	15.05.2021	02.06.2021	И

Диаграмма Ганта – это тип столбчатых диаграмм (гистограмм), который используется для иллюстрации календарного плана проекта, на котором работы по теме представляются протяженными во времени отрезками, характеризующимися датами начала и окончания выполнения данных работ. Для удобства построения графика, длительность каждого из этапов работ приводится в календарных днях.

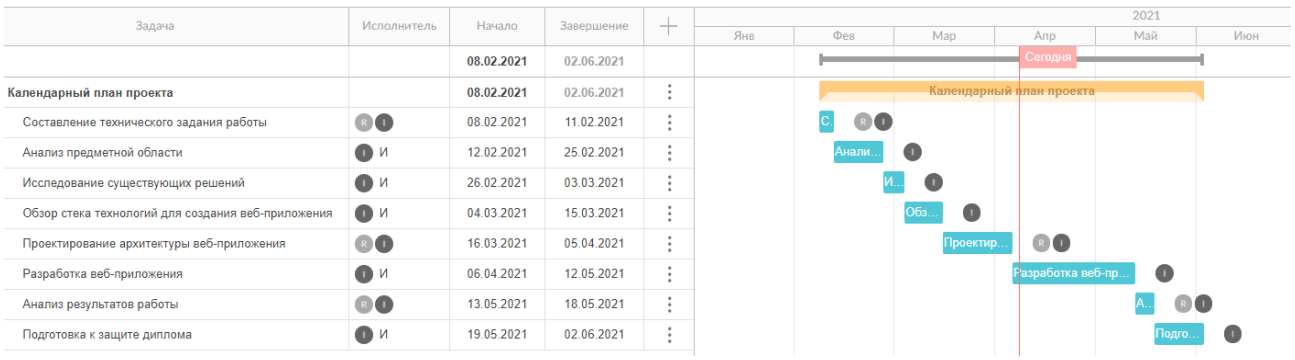

Рисунок 4.1 – Календарный план-график проведения НИОКР по теме

В течении всей научно-исследовательской работы научный руководитель занят 7 рабочих дней, инженер – 78 рабочих дней.

4.3 Бюджет научного исследования

4.3.1 Материальные затраты

Таблица 4.4 – Материальные затраты

Наименование	Затраты, руб.
Персональный компьютер Acer Veriton X2665G	30 000
Расходные материалы (макулатура, периферийные устройства)	2 000
Итого	32 000

4.3.2 Основная заработная плата

Инженер работает по 5-дневной рабочей неделе, а руководитель по 6-дневной.

Таблица 4.5 – Баланс рабочего времени

Показатели рабочего времени	Руководитель	Инженер
Календарное число дней	365	365
Количество нерабочих дней - выходные дни - праздничные дни	 52 14	 98 20
Потери рабочего времени - отпуск - невыходы по болезни	 56 0	 24 0
Действительный годовой фонд рабочего времени	243	223

Месячный должностной оклад взят из документа «Оклады в ТПУ». Руководитель Савельев А. Е. – доцент ОИТ, ктн. $З_{ок}$ = 35111,55 руб. Инженер $З_{ок}$ = 27430,9 руб.

Расчёт основной заработной платы приведен в таблице 6.

Месячный должностной оклад определяется по формуле:

$$З_м = З_{ок} * К_р \quad (2)$$

где $З_{ок}$, руб. – должностной оклад

$k_р$ – районный коэффициент, равный 1,3 (для Томска).

$З_{дн}$ – среднедневная заработная плата работника, руб.

Среднедневная заработная плата рассчитывается по формуле:

$$З_{дн} = \frac{З_{м} \cdot М}{F_{д}} \quad (3)$$

где $З_{м}$ – месячный должностной оклад работника, руб.;

М – количество месяцев работы без отпуска в течение года:

при отпуске в 24 раб. дня М =11,2 месяца, 5-дневная неделя;

при отпуске в 56 раб. дней М=10,4 месяца, 6-дневная неделя;

Fд – действительный годовой фонд рабочего времени научно-технического персонала, раб. дн.

Основная заработная плата ($З_{осн}$) рассчитывается по следующей формуле:

$$З_{осн} = З_{дн} \cdot T_{р} \quad (4)$$

где $З_{осн}$ – основная заработная плата одного работника;

$Т_{р}$ – продолжительность работ, выполняемых научно-техническим работником, раб. дн.

Таблица 4.6 – Расчёт основной заработной платы

Исполнители	**$З_{ок}$, руб.**	**$k_{р}$**	**$З_{м}$, руб**	**$З_{дн}$, руб.**	**$Т_{р}$, раб. дн.**	**$З_{осн}$, руб.**
Руководитель	35111,55	1,3	45645,02	1953,53	7	13674,71
Инженер	27430,9	1,3	35660,17	1791,0	78	139698
Итого						153372,71

4.3.3 Дополнительная заработная плата исполнителей темы

Дополнительная заработная плата рассчитывается исходя из 15% от основной заработной платы, работников, непосредственно участвующих в выполнение темы. Расчет дополнительной заработной платы ведется по следующей формуле:

$$З_{доп} = k_{доп} \cdot З_{осн} \quad (5)$$

где $k_{доп}$ – коэффициент дополнительной заработной платы равный 0,15.

Таблица 4.7 – дополнительная заработная плата

	Руководитель	**Инженер**
Основная заработная плата	13674,71	139698
Дополнительная з/п	2051,21	20954,7
Итого доп. з/п	23005,91	

4.3.4 Отчисления на социальные нужды

Величина отчислений во внебюджетные фонды определяется исходя из следующей формулы:

$$З_{внеб} = k_{внеб} \cdot (З_{осн} + З_{доп}) \quad (6)$$

где $k_{внеб}$ – коэффициент отчислений на уплату во внебюджетные фонды, равный 30% (налоговая нагрузка) по тарифу на 2021 год (статья 425 НК РФ).

Таблица 4.8 – Отчисления во внебюджетные фонды

Исполнитель	**Основная заработная плата, руб.**	**Дополнительная заработная плата, руб.**
Руководитель	13674,71	2051,21
Инженер	139698	20954,7
Коэффициент отчислений во внебюджетные фонды	0,3	
Итого $З_{внеб}$ = 0,3*(13674,71+2051,21)+0,3*(139698+20954,7)= 52913,59 руб.		

4.3.5 Накладные расходы

Расчет затрат на электроэнергию, потраченную в ходе выполнения проекта на работу используемого оборудования, рассчитываемые по формуле:

$$С_{эл.об.} = Р_{об} \cdot t_{об} \cdot Ц_Э \quad (7)$$

где $Р_{об}$ – мощность, потребляемая оборудованием, кВт;

$t_{об}$ – время работы оборудования, час;

$Ц_Э$ – тариф на 1кВт·час. Тариф на промышленную электроэнергию равен 5,748 руб. за 1 кВт·ч.

При этом время работы оборудования рассчитывается на основе данных таблицы 2 ($T_{рд}$ = 78 дней), продолжительность рабочего дня равна 8 часов ($T_{рд}$ = 78 *8 = 624 ч):

$$t_{об} = T_{рд} \cdot K_t \quad (8)$$

где $K_t \leq 1$ – коэффициент использования оборудования по времени, равный отношению времени его работы в процессе выполнения проекта к ТРД. Примем значение $K_t = 1$.

В свою очередь мощность, потребляемая оборудованием, определяется по формуле:

$$P_{ОБ} = P_{ном.} \cdot K_C \quad (9)$$

где $P_{ном.}$ – номинальная мощность оборудования, кВт (показатели для ПК – 0,5 кВт.

$K_C \leq 1$ – коэффициент загрузки, зависящий от средней степени использования номинальной мощности. Примем значение $K_C = 1$.

Затраты на электроэнергию получились:

$$C_{эл.об.} = (0,5*1)*(624*1)* 5,748=1793,376 \text{ руб.} \quad (10)$$

4.4 Формирование бюджета затрат научно-исследовательского проекта

На основании полученных данных по отдельным статьям затрат составляется калькуляция плановой себестоимости научно-исследовательской работы.

Таблица 4.9 – Расчет бюджета затрат НТИ

Наименование статьи	Сумма, руб.	Примечание
1. Материальные затраты	32000	Пункт 5.3.1
2. Основная заработная плата	153372,71	Пункт 5.3.2
3. Дополнительная заработная плата	23005,91	Пункт 5.3.3
4. Отчисления на социальные нужды	52913,59	Пункт 5.3.4
5. Накладные расходы	1793,38	Пункт 5.3.5
Бюджет затрат НТИ	263085,58	Сумма ст. 1- 5

Рассчитанная величина затрат научно-исследовательской работы = 279724,29 руб.

4.5 Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования

4.5.1 Оценка сравнительной эффективности исследования

Эффективность научного ресурсосберегающего проекта включает в себя социальную эффективность, экономическую и бюджетную эффективность. Показатели общественной эффективности учитывают социально-экономические последствия осуществления инвестиционного проекта как для общества в целом, в том числе непосредственные результаты и затраты проекта, так и затраты, и результаты в смежных секторах экономики, экологические, социальные и иные внеэкономические эффекты.

Определение эффективности происходит на основе расчета интегрального показателя эффективности научного исследования. Его нахождение связано с определением двух средневзвешенных величин: финансовой эффективности и ресурсоэффективности.

Интегральный финансовый показатель разработки определяется как:

$$I_{\text{финр}}^{\text{исп.i}} = \frac{\Phi_{\text{pi}}}{\Phi_{\max}} \qquad (11)$$

где $I_{\text{финр}}^{\text{исп.i}}$ – интегральный финансовый показатель разработки;

Φ_{pi} – стоимость i-го варианта исполнения;

$\Phi_{\max}$ – максимальная стоимость исполнения научно-исследовательского проекта (в т.ч. аналоги).

Интегральный показатель ресурсоэффективности вариантов исполнения объекта исследования можно определить следующим образом:

$$\text{I}_{\text{pi}} = \sum a_i \cdot b_i \qquad (12)$$

где I_{pi} – интегральный показатель ресурсоэффективности для i-го варианта исполнения разработки;

a_i – весовой коэффициент i-го варианта исполнения разработки;

b_i^a , b_i^p – балльная оценка i-го варианта исполнения разработки, устанавливается экспертным путем по выбранной шкале оценивания;

n – число параметров сравнения.

Таблица 4.10 – Сравнительная оценка характеристик вариантов исполнения проекта

ПО / Критерии	Весовой коэффициент параметра	Текущий проект	Аналог 1 «РосБизнесСофт WMS»	Аналог 2 «myWMS»
1. Простота использования	0,2	5	3	4
2. Интеграция	0,1	4	4	1
3. Доступность	0,2	5	1	4
4. Функциональность	0,1	4	5	3
5. Простота внедрения	0,1	5	2	3
6. Производительность и скорость работы	0,2	5	5	3
7. Гибкость настройки приложения	0,1	5	2	4
ИТОГО	1			

Интегральный показатель ресурсоэффективности по формуле (12):

$I_{р\ тек\ пр} = 5*0,2+4*0,1+5*0,2+4*0,1+5*0,1+5*0,2+5*0,1=4,80$

$I_{р\ аналог\ 1} = 3*0,2+4*0,1+1*0,2+5*0,1+2*0,1+5*0,2+2*0,1=3,10$

$I_{р\ аналог\ 2} = 3*0,2+4*0,1+1*0,2+5*0,1+2*0,1+5*0,2+2*0,1=3,30$

Интегральный финансовый показатель разработки по формуле (11):

$I_{финр\ тек\ пр} = 60000/230000 =0,26$

$I_{финр\ аналог\ 1} = 100000/230000 =0,43$

$I_{финр\ аналог\ 2} = 70000/230000 =0,30$

Таблица 4.11 – Сравнительная эффективность разработки

№ п/п	Показатели	Аналог 1	Аналог 2	Текущий проект
1	Интегральный финансовый показатель разработки	0,43	0,30	0,26
2	Интегральный показатель ресурсоэффективности разработки	3,10	3,30	4,80
3	Интегральный показатель эффективности	7,13	10,84	18,40
4	Сравнительная эффективность вариантов исполнения	0,39	0,59	1,00

Выводы: интегральные показатели ресурсоэффективности и эффективности разрабатываемого проекта выше, чем у аналогов. Сравнив эти значения, можно сделать вывод, что реализация технологии в разрабатываемом проекте является более эффективным вариантом решения задачи, поставленной в данной работе с позиции финансовой и ресурсной эффективности.

5 СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ

Существующие информационные системы, такие как системы управления складом (WMS), рассчитаны на управление складскими процессами в крупных компаниях, где ведение складского учета товарооборота происходит иным образом нежели в мелком бизнесе, а также такие системы не распространяются в свободном бесплатном доступе [1]. Поэтому очень важно предложить технологию RFID с функциями автоматического извлечения данных в реальном времени для решения проблем, связанных с транспортировкой складских ресурсов.

Проектом данной научно-исследовательской работы является комплексная система, представляющая собой веб-приложение для управления складскими операциями и ресурсами интернет-магазинов, которые, в свою очередь, являются потенциальными потребителями разрабатываемой системы. Веб-приложение позволяет вести учет и визуализировать графически расположение поступающей на склад продукции и имеющихся складских единиц с помощью технологии RFID, вести учет сформированных заказов и сопутствующих расходов интернет-магазина, отображать финансовую статистику бюджета компании, а также позволяющим интегрировать интернет-магазин в систему управления складом с помощью REST API.

Научно-исследовательская работа над проектом проводилась в офисном помещении IT-компании ООО «Ай Ти Приоритет.

Целью раздела является рассмотрение правовых и организационных вопросов обеспечения безопасности, определение производственной, экологической безопасности и рассмотрение безопасности в чрезвычайных ситуациях.

5.1 Правовые и организационные вопросы обеспечения безопасности

5.1.1 Правовые нормы трудового законодательства

Согласно ТК РФ от 30.12.2001 N 197-ФЗ (ред. от 09.03.2021) [31] работник имеет право на:

- рабочее место, соответствующее требованиям охраны труда;
- обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний в соответствии с федеральным законом;
- отказ от выполнения работ в случае возникновения опасности для его жизни и здоровья вследствие нарушения требований охраны труда, за исключением случаев, предусмотренных федеральными законами, до устранения такой опасности;
- обеспечение средствами индивидуальной и коллективной защиты в соответствии с требованиями охраны труда за счет средств работодателя;
- внеочередной медицинский осмотр в соответствии с медицинскими рекомендациями с сохранением за ним места работы (должности) и среднего заработка во время прохождения указанного медицинского осмотра.
- предоставление личных данных работодателю только после согласия самого работника.
- перерыв на обед или отдых в течение рабочего дня, согласно правилам внутреннего распорядка организации;
- продолжительность рабочего времени не может превышать 40 часов в неделю.

Все права работника при выполнении научно-исследовательской работы были соблюдены в соответствии ТК РФ от 30.12.2001 N 197-ФЗ (ред. от 09.03.2021).

5.1.2 Эргономические требования к правильному расположению и компоновке рабочей зоны

Выполнение требований к рабочему месту в офисном помещении ООО «Ай Ти Приоритет» отражено в таблице 1, согласно требованиям, ГОСТ 12.2.032-78 [32]. Конструкция рабочего места и взаимное расположение всех его элементов (сиденье, органы управления, средства отображения информации и т.д.) должны соответствовать антропометрическим, физиологическим и психологическим требованиям, а также характеру работы. на данном рабочем месте.

Таблица 5.1 – Требования к организации рабочего места

Требование	**Требуемое значение**	**Значение параметров в помещении**
Помещение	Площадь не менее 6 м², высота помещения должна быть не менее 4 м, а объем - не менее 20 м³ на одного человека	Соответствует
Рабочий стул	Подъемно-поворотный, регулируемый по и высоте и углу наклона спинки. Должен иметь дизайн, исключающий онемение тела из-за нарушения кровообращения при продолжительной работе на рабочем месте	Соответствует
Рабочий стол	Высота 680-800мм. Оптимальные размеры поверхности стола 1600 х 1000 мм². Должен быть устойчивым, иметь однотонное неметаллическое покрытие, не обладающее способностью накапливать статическое электричество	Соответствует
Расположение монитора от глаз пользователя	400-800 мм	Соответствует

Рабочее место в офисном помещении ООО «Ай Ти Приоритет, где выполнялась научно-исследовательская работа соответствует всем требованиям ГОСТ 12.2.032-78.

5.2. Производственная безопасность

5.2.1 Анализ вредных и опасных производственных факторов

Для выбора факторов использовался ГОСТ 12.0.003-2015 Опасные и вредные производственные факторы. Классификация. Перечень опасных и

вредных факторов [33], характерных для проектируемой производственной среды. Опасные и вредные факторы представлены в виде таблицы 2. Процесс разработки веб-приложения разделяется на проектирование и разработку веб-приложения.

В случае, где объектом рассмотрения является рабочее место, включая персональный компьютер и помещение, среди вредных и опасных факторов можно указать: микроклимат помещения, неправильное освещение, шум, опасность поражения электрическим током и нервно-психические перегрузки, такие как, умственное перенапряжение, перенапряжение анализаторов, монотонность труда, эмоциональные перегрузки.

Таблица 5.2 - Возможные опасные и вредные факторы

Факторы (ГОСТ 12.0.003-2015)	**Этапы работ**		**Нормативные документы**
	Проектирование	Разработка	
1. Отклонение показателей микроклимата	+	+	1. СанПиН 2.2.4.548-96. Гигиенические требования к микроклимату производственных помещений [36]. 2. СанПиН 2.1.6.1032-01. Гигиенические требования к качеству атмосферного воздуха [37]. 3. СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95 [38]. 4. СН 2.2.4/2.1.8.562-96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки [39]. 5. ГОСТ 12.0.003-2015 Опасные и вредные производственные факторы. Классификация. Перечень опасных и вредных факторов [40]. 6. ТОИ Р-45-084-01 Типовая инструкция по охране труда при работе на персональном компьютере [41]. 7. ГОСТ Р 12.1.019-2009. Электробезопасность. Общие требования и номенклатура видов защиты [42].
2. Недостаточная освещенность рабочей зоны	+	+	
3. Превышение уровня шума	+	+	
4. Нервно-психические перегрузки (умственное и эмоциональное перенапряжение, перенапряжение анализаторов, монотонность).	+	+	
5. Повышенное значение напряжение в электрической цепи, замыкание которой может пройти через тело человека	+	+	

1. Отклонение показателей микроклимата.

Микроклимат определяется действующими на организм человека показателями температуры, влажности и скорости движения воздуха. Длительное воздействие на человека неблагоприятных показателей

микроклимата ухудшает его самочувствие, снижает производительность труда и приводит к заболеваниям. Согласно СанПиН 2.2.4.548-96 [34] работа, производимая на рабочем месте в офисном помещении ООО «Ай Ти Приоритет» относится к работам категории Iб: относятся работы с интенсивностью энергозатрат 121-150 ккал/ч (140-174 Вт), производимые сидя, стоя или связанные с ходьбой и сопровождающиеся некоторым физическим напряжением (ряд профессий на предприятиях связи, контролеры, мастера в различных видах производства и т. п.). Также температура в помещении не должна выходить за пределы 21-24°.

Таблица 5.3 - Оптимальные величины показателей микроклимата на рабочих местах производственных помещений

Период года	**Катег. работ по уровню энер-гозатрат**	**Температура воздуха, °С**	**Температура поверхностей, °С**	**Относ. влажность воздуха, %**	**Скорость движения воздуха, м/с**
Холодный	Iб	21-23	20-24	40-60	0,1
Теплый	Iб	22-24	21-25	40-60	0,1

Микроклимат на рабочем месте в офисном помещении ООО «Ай Ти Приоритет» соответствует допустимым нормам СанПиН 2.1.6.1032-01 [35].

2 Недостаточная освещенность рабочей зоны

Согласно СП 52.13330.2016 Естественное и искусственное освещение [36], уровень общей освещённости на одно рабочее место в помещении должно составлять не менее 400 лк. На рабочем месте в офисном помещении ООО «Ай Ти Приоритет» имеется естественное и искусственное освещение. В качестве источников света применяются люминесцентные лампы. При равномерном размещении люминесцентных светильников последние располагаются обычно рядами – параллельно рядам оборудования. При высоких уровнях нормированной освещённости люминесцентные светильники обычно располагаются непрерывными рядами, для чего светильники сочленяются друг с другом торцами. Согласно СП 52.13330.2016 на рабочем месте в офисном помещении ООО «Ай Ти Приоритет» III разряд зрительной работы.

Таблица 5.4 – Нормативные значения освещенности

Характеристика зрительной работы	Наименьший или эквивалентный размер объекта различения, мм	Разряд зрительной работы	Подразряд зрительной работы	Контраст объекта с фоном	Характеристика фона	Естественное освещение		
						Освещенность, лк		
						при системе комбинированного освещения		при системе общего освещения
						всего	в том числе от общего	
Высокой точности	От 0,30 до 0,50	III	г	Средний	Светлый	400	200	200

Размеры офисного помещения ООО «Ай Ти Приоритет»: длина А = 7 м, ширина В = 6 м, высота Н = 4 м. Высота рабочей поверхности $h_{рп}$ = 0,8 м. Требуется создать освещенность Е = 400 лк. Площадь помещения S = 42 м^2. Коэффициент отражения стен (оклеенные светлыми обоями) $R_с$= 30 %, потолка (чистый бетонный) $R_п$=50%. Коэффициент запаса k=1,5, коэффициент неравномерности Z= 1,1.

Рассчитываем систему общего люминесцентного освещения. Выбираем светильники типа ОДР, λ= 1,3. Приняв $h_с$= 0,5 м, определяем расчетную высоту: $h=H-h_с-h_{рп}=4-0,5-0,8=2,7$ м

Расстояние между светильниками: $L=\lambda*h=1,2*2,7=3,24$ м

Расстояние от крайнего ряда светильников до стены: $L/3=1,08$ м

Количество рядов светильников с люминесцентными лампами определяется по формуле:

$$n_{\text{ряд}} = \frac{(B-\frac{2}{3}*L)}{L} + 1 = \frac{(6-\frac{2}{3}*3,24\ )}{3,24} + 1 = 2,19 \approx 2,$$

где $n_{ряд}$ – количество рядов; В – ширина помещения, м; L – расстояние между рядами светильников, м.

Количество светильников с люминесцентными лампами определяется по формуле:

$$n_{\text{СВ}} = \frac{(A-\frac{2}{3}*L)}{l_{\text{СВ}}+0,5} = \frac{(7-\frac{2}{3}*3,24\ )}{1,2+0,5} = 2,84 \approx 3,$$

где $n_{св}$ – количество светильников в ряду; А – длина помещения, м; $l_{св}$ – длина светильника, м.

Общее количество светильников с люминесцентными лампами в помещении определяется по формуле:

$$N = n_{ряд} \cdot n_{св} = 6$$

где N – общее количество светильников; $n_{ряд}$ – количество рядов; $n_{св}$ – количество светильников в ряду

Размещаем светильники в два ряда. В каждом ряду можно установить 12 светильников типа ОД – 2-40 мощностью 40 Вт (с длиной 1,2 м), при этом разрывы между светильниками в ряду составят 60 см. Изображаем в масштабе план помещения и размещения на нем светильников (рисунок 1). Учитывая, что в каждом светильнике установлено две лампы, общее число ламп в помещении 12.

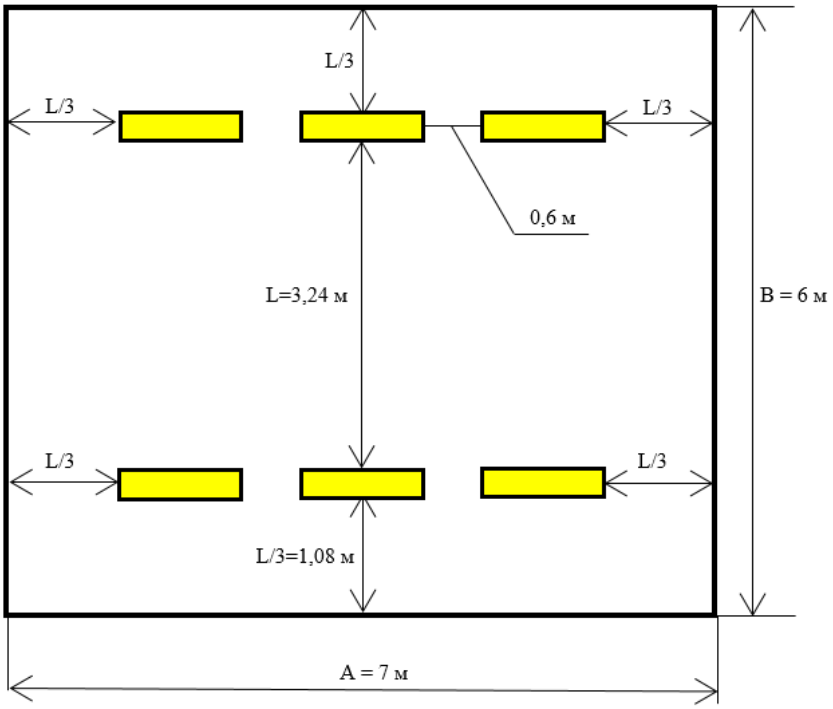

Рисунок 5.1 – План помещения и размещения светильников с люминесцентными лампами

Находим индекс помещения: $i = \frac{S}{h*(A+B)} = \frac{42}{2,7*(7+6)} = 1,196 \approx 1,2$,

Из справочника «Коэффициенты использования светового потока светильников с люминесцентными лампами» определяем коэффициент использования светового потока при R_c= 30 %, R_n=50%, $i = 1,25$: η= 0,50.

Определяем потребный световой поток ламп в каждом из рядов

$$\Phi = \frac{E_H * S * K_з * Z}{N_л * \eta} = \frac{400 * 42 * 1{,}5 * 1{,}1}{12 * 0{,}5} = 4620,$$

где $E_н$ – нормативная освещённость, лк; S – площадь освещаемого помещения, $м^2$; $K_з$ – коэффициент запаса, учитывающий загрязнение светильника, наличие в атмосфере цеха дыма, пыли; Z – коэффициент неравномерности освещения для люминесцентных ламп при расчётах берётся равным 1,1; $N_л$ – число ламп в помещении (необходимо учесть число ламп в светильнике); η – коэффициент использования светового потока.

Из справочника «Основные характеристики люминесцентных ламп» выбираем ближайшую стандартную лампу ЛТБ 5200 лм с мощностью 80 Вт. Делаем проверку выполнения условия:

$$-10\% \leq \frac{\Phi_{л.станд} - \Phi_{л.расч}}{\Phi_{л.станд}} * 100\% \leq +20\%,$$

$$-10\% \leq 11{,}1\% \leq +20\%,$$

Определяем электрическую мощность осветительной установки P= 12*80 = 960 Вт.

Согласно СП 52.13330.2016 условия освещения на рабочем месте в офисном помещении ООО «Ай Ти Приоритет» соответствуют нормам.

3. Превышение уровня шума

В соответствии с СН 2.2.4/2.1.8.562-96 [37] при ежедневной (кроме выходных дней) работе, но не более 40 часов в неделю в течение всего рабочего стажа, уровень шума не должен вызывать заболеваний или отклонений в состоянии здоровья, обнаруживаемых современными методами исследований в процессе работы или в отдаленные сроки жизни настоящего и последующих поколений.

Источниками шума на рабочем месте в офисном помещении ООО «Ай Ти Приоритет» являются: персональный компьютер. Уровень шума в помещениях не должен превышать 80 дБ согласно СН 2.2.4/2.1.8.562-96.

Таблица 5.5 – Допустимые значения уровней звукового давления в октавных полосах частот и уровня звука, создаваемого персональным компьютером

Уровни звукового давления в октавных полосах со среднегеометрическими частотами									**Уровни звука в дБА**
31,5 Гц	63 Гц	125 Гц	250 Гц	500 Гц	1000 Гц	2000 Гц	4000 Гц	8000 Гц	
86 дБ	71 дБ	61 дБ	54 дБ	49 дБ	45 дБ	42 дБ	40 дБ	38 дБ	50

Измерение уровня звука и уровней звукового давления проводится на расстоянии 50 см от поверхности оборудования и на высоте расположения источника(ков) звука.

Уровень шума в офисном помещении ООО «Ай Ти Приоритет» соответствует нормам, является допустимым, так как не вызывает значительного беспокойства, и не влияет на психологическое состояние.

4. Нервно-психические перегрузки

Нервно-психические перегрузки подразделяют на: умственное перенапряжение, эмоциональные перегрузки, перенапряжение анализаторов, монотонность туда.

При работе с большими объёмами данных существует опасность умственного перенапряжения и в следствие, невозможность дальнейшей продуктивной работы. При умственной работе потребность мозга в кислороде увеличивается в 15-20 раз и возрастает опасность повышенного артериального давления. Постоянное пребывание в подобных условиях может привести к сердечно-сосудистым заболеваниям. Согласно ТОИ Р-45-084-01. «Типовая инструкция по охране труда при работе на персональном компьютере» [38] для сотрудника в ООО «Ай Ти Приоритет» устанавливается категория работы III для группы В - уровень нагрузки по суммарному времени непосредственной работы с компьютером за рабочую смену (не более 6 часов за смену).

Таблица 5.6 – Уровень нагрузки за рабочую смену при видах работ с компьютером

Категория работ	**Уровень нагрузки за рабочую смену при видах работ с компьютером**		
	группа А, количество знаков	группа Б, количество знаков	группа В, час
III	До 60000	До 40000	До 6,0

Из методов для предупреждения развития нервного и психического напряжения при работе за ПК можно выделить: наличие частых перерывов, с возможностью небольших физических нагрузок; соответствие рабочего места нормативам, калорийные перекусы, для восстановления энергии. Каждый сотрудник в ООО «Ай Ти Приоритет» не подвержен нервно-психическим перегрузкам, помещение имеет зону отдыха и принятия пищи.

5. *Повышенное значение напряжение в электрической цепи, замыкание которой может пройти через тело человека*

Согласно ГОСТ Р 12.1.019-2009 [39] по опасности поражения электрическим током рабочее место в офисном помещении ООО «Ай Ти Приоритет» относится к первому классу – помещения без повышенной опасности: сухое, хорошо отапливаемое, помещение с токонепроводящими полами, с температурой 18-20°, с влажностью 40-50%).

Для предупреждения электротравматизма необходимо проводить соответствующие организационные и технические мероприятия: 1) оформление работы нарядом или устным распоряжением; 2) проведение инструктажей и допуск к работе; 3) надзор во время работы. Уровень напряжения для питания в офисе ООО «Ай Ти Приоритет» - 220 В.

На рабочем месте в офисном помещении ООО «Ай Ти Приоритет», согласно ГОСТ 12.1.030-81 [40] основную защиту обеспечивают посредством основной изоляции между опасными частями, находящимися под напряжением, и открытыми проводящими частями, а также посредством автоматического отключения питания, защитное отключение; защитное разделение сетей, предохранительные устройства.

На рабочем месте в офисе размещены дисплей, клавиатура и системный блок. При включении дисплея на электронно-лучевой трубке создается высокое напряжение в несколько киловольт. Поэтому запрещается прикасаться к тыльной стороне дисплея, вытирать пыль с компьютера при его включенном состоянии, работать на компьютере во влажной одежде и влажными руками.

Перед началом работы следует убедиться в отсутствии свешивающихся со стола или висящих под столом проводов электропитания, в целостности вилки и провода электропитания, в отсутствии видимых повреждений аппаратуры и рабочей мебели, в отсутствии повреждений и наличии заземления приэкранного фильтра.

Токи статического электричества, наведенные в процессе работы компьютера на корпусах монитора, системного блока и клавиатуры, могут приводить к разрядам при прикосновении к этим элементам. Такие разряды опасности для человека не представляют, но могут привести к выходу из строя компьютера. Для снижения величин токов статического электричества используются нейтрализаторы, местное и общее увлажнение воздуха, использование покрытия полов с антистатической пропиткой.

Электробезопасность в офисном помещении ООО «Ай Ти Приоритет», соответствует нормам.

5.2.2 Обоснование мероприятий по снижению уровней воздействия опасных и вредных факторов на исследователя

На основании рассмотренных ранее подразделов выделим те профилактические мероприятия, которые необходимы для снижения опасного и вредного воздействия во время работы с ПК на рабочем месте ООО «Ай Ти Приоритет»:

Мероприятия для поддержания комфортного микроклимата: оснащенность офисных помещений системами регулировки температуры воздуха; наличие центрального отопления в зимний период; правильно выстроенная система вентиляции и проветривания.

Мероприятия для достижения необходимого уровня освещенности: правильно подобранный тип освещения в зависимости от времени года и суток; достаточное количество и возможность регуляции источников искусственного освещения; наличие штор типа «жалюзи», для эффективного рассеивания естественного света.

Мероприятия по минимизации шумовых воздействий: наличие систем шумоизоляции и шумоподавления; грамотное расположение рабочих зон и источников шума относительно друг друга.

Мероприятия по профилактике нервно-психического напряжения: проведение частых перерывов, с возможностью физических нагрузок; наличие зон отдыха и принятия пищи в организации.

Мероприятия по защите от удара электрическим током: соблюдение техники безопасности при работе с электрическими приборами; подключение и регулярные проверки оборудования, в соответствии со всеми техническими требованиями.

5.3. Экологическая безопасность

Объект магистерской работы является нематериальным, и, следовательно, может рассматриваться в связке только с ПК. Производство ПК включает в себя токсичное сырье, которое подлежит специальной утилизации и переработке – без них материалы способны постепенно разрушаясь наносить непоправимый вред экологии и здоровью человека. Многие предметы офисной техники, после завершения срока своей эксплуатации, становятся опасными отходами, которые могут оказать вред атмосфере, гидросфере и литосфере. Например, ЖК-экраны являются большим источником парниковых газов, а люминесцентные лампы содержат в себе от 10 до 70 мг ртути. Согласно Кодексу Российской Федерации, об административных правонарушениях [41], отработанную технику (в том числе ПК) запрещается выбрасывать наряду с обыкновенным мусором, а необходимо обратиться в специальные службы для ее утилизации или переработки. Нормы ГОСТ 12.3.031-83 «Работы со ртутью.

Требования безопасности» [42] требуют, чтобы все отходы и приборы, содержащие ртуть, подлежали сбору и возврату только сертифицированным лицом (электромонтером). К отходам, производимым в помещении, также можно отнести макулатуру, пластиковые отходы, люминесцентные лампы. Бумажные отходы рекомендуется накапливать и передавать их в пункты приема макулатуры для дальнейшей переработки. Пластиковые бутылки складывать в специально предназначенные контейнеры. Люминесцентные лампы возможно сдавать организациям, имеющим специальную лицензию, которые занимаются переработкой таких отходов.

В качестве профилактики и предотвращения опасного воздействия электрической техники на окружающую среду, также нужно обращать внимание на соответствие используемых материалов в ПК нормам и стандартам экологической безопасности.

5.4. Безопасность в чрезвычайных ситуациях

5.4.1 Анализ возможных ЧС

Согласно ГОСТ Р 22.0.02-2016 «Безопасность в чрезвычайных ситуациях. Термины и определения» [43] Чрезвычайная ситуация (ЧС) - это обстановка на определенной территории, сложившаяся в результате аварии, опасного природного явления, катастрофы, стихийного или иного бедствия, которые могут повлечь или повлекли за собой человеческие жертвы, ущерб здоровью людей или окружающей среде, значительные материальные потери и нарушение условий жизнедеятельности людей.

С точки зрения выполнения проекта характерны следующие виды ЧС: пожары, внезапное обрушение зданий, сооружений, геофизические опасные явления (грозы, оползни).

Наиболее вероятной ЧС, которая может возникнуть при производстве объекта на рабочем месте в офисе ООО «Ай Ти Приоритет», является пожар. Пожар может возникнуть при коротком замыкании, искрении, статическом электричестве, неосторожном обращение с огнем, курении, оставлении без

присмотра нагревательных приборов, неработоспособное электрооборудование, неисправности в проводке, розетках и выключателях, перегрузка в электроэнергетической системе.

5.4.2 Обоснование мероприятий по предотвращению ЧС

Пожарная безопасность на рабочем месте в офисе ООО «Ай Ти Приоритет» должна обеспечиваться системами предотвращения пожара и противопожарной защиты, в том числе организационно-техническими мероприятиями. Под пожарной профилактикой понимается обучение пожарной технике безопасности и комплекс мероприятий, направленных на предупреждение пожаров.

Согласно НПБ 104-03 «Проектирование систем оповещения людей о пожаре в зданиях и сооружениях» [44] для оповещения о возникновении пожара на рабочем месте в офисе ООО «Ай Ти Приоритет» установлены дымовые оптико-электронные автономные пожарные извещатели 3 шт, а оповещение о пожаре осуществляется подачей звуковой и световой сигнализации.

Рабочее место в офисе ООО «Ай Ти Приоритет» оснащено такими средствами пожаротушения: огнетушителями типа ОУ-3 1 шт. 5 литров и ОП-3 1 шт. 3 литра (предназначены для тушения любых материалов, предметов и веществ, применяется для тушения ПК и оргтехники).

Таблица 5.7 – Типы используемых огнетушителей при пожаре в электроустановках

Напряжение, кВ	Тип огнетушителя (марка)
До 1,0	порошковый (серии ОП)
До 10,0	углекислотный (серии ОУ)

Согласно СП 12.13130.2009 [16] помещение в офисе в офисе ООО «Ай Ти Приоритет» относится к типу Д – пониженной пожароопасности.

Таблица 5.8 - Категории помещений по взрывопожарной и пожарной опасности

Категория помещения	Характеристика веществ и материалов, находящихся (обращающихся) в помещении
Д пониженная пожароопасность	Негорючие вещества и материалы в холодном состоянии

Мероприятия по повышению пожарной безопасности:

1. Технические: соблюдение противопожарных норм при выборе и монтаже оборудования; использование первичных средств пожаротушения;

2.Эксплуатационные: правильная эксплуатация оборудования; правильное содержание помещения;

3.Режимные: проведение профилактических осмотров; наблюдение за наличием и исправностью противопожарного оборудования. Следует выполнять требования правил пожарной безопасности.

В случае возникновения загорания необходимо обесточить электрооборудование, отключить систему вентиляции, принять меры тушения (на начальной стадии) и обеспечить срочную эвакуацию работников и сотрудников в соответствие с планом эвакуации помещения.

При возникновении пожара необходимо сообщить об этом в городскую пожарную охрану по телефону 01 (при этом необходимо сообщить точный адрес здания, место возникновения пожара или обнаружения признаков пожара, вероятную возможность угрозы людям, а также другие сведения, необходимые диспетчеру пожарной охраны). Кроме того, следует назвать себя и номер телефона, с которого делается сообщение о пожаре.

5.5 Выводы по разделу

Каждый работник должен проводить профессиональную деятельность с учетом социальных, правовых, экологических и культурных аспектов, вопросов здоровья и безопасности, нести социальную ответственность за принимаемые решения, осознавать необходимость устойчивого развития.

В данном разделе были рассмотрены основные вопросы соблюдения прав работника на труд, выполнения правил к безопасности труда, промышленной безопасности, экологии и ресурсосбережения.

Установлено, что рабочее место в офисном помещении ООО «Ай Ти Приоритет» удовлетворяет требованиям безопасности и гигиены труда во время реализации проекта, а вредное воздействие объекта исследования на окружающую среду не превышает норм.

ЗАКЛЮЧЕНИЕ

В процессе исследовательской работы были проведен литературный анализ предметной области, а также проведен анализ существующих конкурентных систем управления складом. На основе сравнения конкурентных решений были сформированы функциональные требования к разрабатываемой системе. После формирования требований к системе был проведен выбор стека технологий для создания веб-приложения.

В результате исследовательской работы спроектирована и разработана система управления складом в виде веб-приложения с помощью современного стека технологий MEAN (MongoDB, Express.js, Angular, Node.js) с возможностью синхронизации с интернет-магазином за счет REST API.

Были реализованы такие функциональные характеристики системы как:

- Управление заказами и расходами.
- Управление запасами.
- Импорт накладных, приходящие от поставщиков.
- Экспорт информации о расходах компании. Функционал добавления нужных пользователю сущностей.
- Настройка шаблонов сущностей для добавления неограниченного количества параметров.
- Создание неограниченного количества сущностей индивидуальных складов, пропорциональных параметрам реальных складских помещений и индивидуальное размещение складских стеллажов.
- Возможность вручную расставлять продукцию на карте склада
- Возможность использования технологии RFID для местонахождения продукции на карте склада.
- Просмотр статистики товарооборота, выручки, прибыли, расходов компании, наглядный просмотр даты истечения сроков годности имеющихся продуктов на складе.

Преимущества разработанной системы заключаются в том, что что разработанная система позволяет повысить производительность, избежать множество ошибок при ведении учета продукции и снизить эксплуатационные расходы склада за счет возможности кастомизации под индивидуальные нужды пользователя, гибкости в настройке системы, 2D визуализации местоположения продукции на складе и возможности интеграции системы по REST API. Также одним из преимуществ является исполнение системы в виде веб-приложения с открытым исходным кодом, что обеспечивает кроссплатформенность, доступность с различных устройств и отсутствие необходимости установки.

СПИСОК ПУБЛИКАЦИЙ СТУДЕНТА

1. Трофимова А.Е., Ткачев М., Ротарь В.Г. РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ СКЛАДСКИМИ ПРОЦЕССАМИ ИНТЕРНЕТ-МАГАЗИНОВ // Молодежь и современные информационные технологии: сборник трудов XVIII Международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 22–26 марта 2021 г. - Томск: ТПУ, 2021 - С. 458.
2. Ткачев М., Трофимова А.Е., Ротарь В.Г. ОБЗОР АКТУАЛЬНЫХ ФРЕЙМВОРКОВ ДЛЯ РАЗРАБОТКИ КЛИЕНТСКОЙ ЧАСТИ СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ // Молодежь и современные информационные технологии: сборник трудов XVIII Международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 22–26 марта 2021 г. - Томск: ТПУ, 2021 - С. 458.

СПИСОК ЛИТЕРАТУРЫ

3. Пьюривал С. Основы разработки веб-приложений //СПб.: Питер. – 2015. – 272 с.
4. Бекирова Э. А., Халилова З. Э. Основные этапы создания web-приложений //Информационно-компьютерные технологии в экономике, образовании и социальной сфере. – 2019. – №. 1. – С. 84-91.
5. Демин В., Яшина Ю. Оценка российского рынка систем класса WMS //Логистика. – 2015. – №. 11. – С. 16-21.
6. Эспозито Д. Разработка современных веб-приложений: анализ предметных областей и технологий //М.: Вильямс. – 2017. – 464 с.
7. Круглова Н. Ю. Основы бизнеса. – М.: Изд-во РДЛ, 2005. . – 560 с.
8. Савин В.И. Организация складской деятельности. М.: Дело и сервис, 2016. 544 с.
9. Иващенко Т. И. Организация эффективной работы складского хозяйства //Ученые заметки ТОГУ. – 2015. – Т. 6. – №. 1. – С. 229-232.
10. Poon T. C. et al. A RFID case-based logistics resource management system for managing order-picking operations in warehouses // Expert Systems with Applications. – 2009. – Т. 36. – №. 4. – С. 8277-8301.
11. Li M. et al. A RFID-based intelligent warehouse management system design and implementation //2011 IEEE 8th International Conference on e-Business Engineering. – IEEE, 2011. – С. 178-184.
12. Эспозито Д. Разработка современных веб-приложений: анализ предметных областей и технологий //М.: Вильямс. – 2017. – 464 с.
13. Костышева Я. В. Анализ российского рынка автоматизированных систем управления складом //Вестник молодых ученых Самарского государственного экономического университета. – 2014. – №. 1. – С. 80-86.
14. Терентьева В. И., Комкова А. В. АВТОМАТИЗАЦИЯ УПРАВЛЕНИЯ СКЛАДОМ: WMS РЕШЕНИЯ //Международный студенческий научный вестник. – 2015. – №. 5-5. – С. 648-648.

15. Стоякина В. Ю., Бехбудова С. УПРАВЛЕНИЕ СКЛАДОМ НА ОСНОВЕ WMS //МЕЖДИСЦИПЛИНАРНОСТЬ НАУКИ КАК ФАКТОР ИННОВАЦИОННОГО РАЗВИТИЯ. – 2017. – С. 250-253.
16. Гордеев С. Ю., Тимофеев А. В., Козлов В. В. Разработка веб-приложений с использованием angular. Js, node. Js, mongodb на примере системы психологической поддержки студентов-участников программы" полет" //Наука и образование сегодня. – 2017. – №. 2 (13).
17. Poulter A. J., Johnston S. J., Cox S. J. Using the MEAN stack to implement a RESTful service for an Internet of Things application //2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). – IEEE, 2015. – С. 280-285.
18. Бэнкер К. Mongodb в действии. – Litres, 2017.
19. Филиппов А. Н., Поволоцкий Я. А. Применение нереляционной субд mongodb в сапр тп. – 2018.
20. Лисин С. И. Обзор и сравнительный анализ систем управления нереляционными базами данных //Молодежный научно-технический вестник. – 2013. – №. 5. – С. 26-26.
21. Хэррон Д. Node. Js Разработка серверных веб-приложений на javascript. – Litres, 2017.
22. Satheesh M., d'mello B. J., Krol J. Web development with mongodb and nodejs. – Packt Publishing Ltd, 2015.
23. Liang L. Et al. Express supervision system based on nodejs and mongodb //2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS). – IEEE, 2017. – С. 607-612.
24. Sterling A. Nodejs and Angular Tools for JSON-LD //2019 IEEE 13th International Conference on Semantic Computing (ICSC). – IEEE, 2019. – С. 392-395.
25. Козловский П., Дарвин П. Б. Разработка веб-приложений с использованием angularjs. – Litres, 2017.
26. Чернова М. В. И др. Фреймворк javascript angular js //фундаментальные основы инновационного развития науки и образования. – 2019. – С. 75-77.

27. Фримен А. Angular для профессионалов. – " Издательский дом"" Питер""", 2017.
28. Masse M. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. – " O'Reilly Media, Inc.", 2011.- С. 93.
29. Ebert J. SOA with REST: principles, patterns & constraints for building enterprise solutions with REST by Thomas Erl, Benjamin Carlyle, Cesare Pautasso, Raj Balasubramanian //ACM SIGSOFT Software Engineering Notes. – 2013. – Т. 38. – №. 3. – С. 32-33.
30. Chae S., Yoshida T. Application of RFID technology to prevention of collision accident with heavy equipment //Automation in construction. – 2010. – Т. 19. – №. 3. – С. 368-374.
31. Сохина А. А., Готская И. Б. Обзор популярных систем сборки для фронтенд-проектов //Альманах научных работ молодых ученых Университета ИТМО. – 2017. – С. 219-222.
32. Жуков С. В. Обзор и сравнение популярных фреймворков для frontend-разработки //Передовые инновационные разработки. Перспективы и опыт использования, проблемы внедрения в производство. – 2019. – С. 148-150.
33. Трудовой кодекс Российской Федерации" от 30.12.2001 N 197-ФЗ (ред. от 09.03.2021)
34. ГОСТ 12.2.032-78 Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования.
35. ГОСТ 12.0.003-2015 Опасные и вредные производственные факторы. Классификация. Перечень опасных и вредных факторов.
36. СанПиН 2.2.4.548-96. Гигиенические требования к микроклимату производственных помещений.
37. СанПиН 2.1.6.1032-01. Гигиенические требования к качеству атмосферного воздуха.
38. СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-0595.

39. СН 2.2.4/2.1.8.562-96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки.
40. ТОИ Р-45-084-01 Типовая инструкция по охране труда при работе на персональном компьютере (утв. Приказом Минсвязи РФ от 02.07.2001 N 162)
41. ГОСТ Р 12.1.019-2009. Электробезопасность. Общие требования и номенклатура видов защиты.
42. ГОСТ 12.1.030-81 Электробезопасность. Защитное заземление. Зануление
43. Кодекс Российской Федерации об административных правонарушениях от 30.12.2001 N 195-ФЗ (ред. от 24.04.2020) (с изм. и доп., вступ. в силу с 01.06.2020).
44. ГОСТ 12.3.031-83 Система стандартов безопасности труда. Работы со ртутью. Требования безопасности.
45. ГОСТ Р 22.0.02-2016 Безопасность в чрезвычайных ситуациях. Термины и определения.
46. Нормы пожарной безопасности 104-03. Системы оповещения и управления эвакуацией людей при пожарах в зданиях и сооружениях.
47. Свод правил 12.13130.2009. Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности.

Приложение А. Раздел на иностранном языке

(справочное)

Part 2 Object and methods of research
Part 3.1 Choosing a technology stack to build a web application

Студент

Группа	ФИО	Подпись	Дата
8ИМ91	Трофимова Анастасия Евгеньевна		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Савельев Алексей Олегович	к.т.н.		

Консультант-лингвист отделения иностранных языков ШБИП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИЯ	Сидоренко Татьяна Валерьевна	к.п.н.		

Abbreviations

JS (JavaScript) — is a programming language for programmatically accessing application objects.

TS (TypeScript) — is a programming language by Microsoft for developing web applications. It expands the possibilities JavaScript.

MEAN (MongoDB, Express.js, Angular and Node.js) — is a stack of server software for developing websites and web applications in JavaScript.

DB (Database) — it is an ordered collection of structured information or data. This data is stored electronically in a computer system.

NoSQL — it is a database that does not require a uniquely defined data storage structure.

DBMS (Database Management System) — it is a set of language and software tools. It is designed for the creation, maintenance and sharing of a database by many users.

JSON (JavaScript Object Notation) — it is a text file format for storing and exchanging information. The file contains only text and uses the extension. json.

MVC (Model-View-Controller) — it is a way of organizing code and separating application data into three separate components, such as model, view, and controller, to perform different tasks.

Part 3.1 Technology stack for web application development

Web development is a rapidly growing in the areas of commercial services, education, communications, household and healthcare. There are various technologies for developing web applications. They are constantly updated and improved to achieve functionality and efficiency.

The MEAN is an acronym for three JavaScript frameworks and a NoSQL document database. The combination of MongoDB, Express.js, Angular, Node.js tools allows web developers to reduce the time spent on system administration and deployment of web applications, websites and APIs.

The MEAN stack combines several free JavaScript frameworks. They can be used to create a single page application. JavaScript technologies are used not only on the frontend, but also on the backend.

JavaScript is one of the most in-demand programming languages according to TIOBE Index Programming Language Rankings statistics for February 2021. The specialists of the largest platform for hosting and joint development of information projects GitHub ranked JavaScript as the first in the ranking of programming languages in 2020 according to the results of the Octoverse. The figure A.1 shows the results of research.

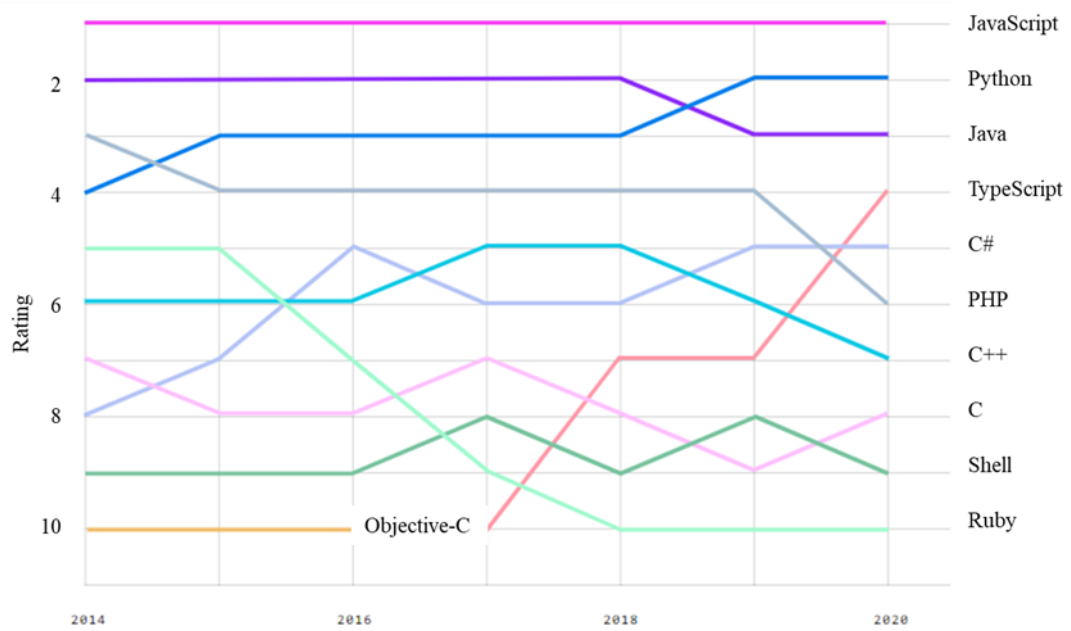

Figure A.1 – Most Popular programming languages according to Octoverse research for 2020

Document-oriented MongoDB is used as a database management system (DBMS) in the MEAN stack. The client side of web applications is developed using the Angular framework. JavaScript-platform Node.js is a server in the MEAN stack. The web application framework Express.js is used to run the backend of MEAN applications in addition to Node.js.

Part 3.1.1 Overview of MongoDB database

MongoDB is a free, document-oriented database. It provides scalability and flexibility for application development. Data is not written in tables or columns.

Instead, JSON-like documents with dynamic schemas are stored in MongoDB. MongoDB is faster, has better scalability, and is easier to use.

JavaScript and JSON structures are used as the query language in MongoDB. A document-oriented DBMS uses JSON format to represent documents and display results. JSON structures are stored in binary JSON format. This flexibility makes MongoDB well suited for rapid application development while dealing with rapidly changing requirements.

MongoDB is written in C ++, so it's easy to port it to a wide variety of platforms. MongoDB can be deployed on Windows, Linux, MacOS, Solaris platforms. Using the library is recommended from the official site.

The primary key is found in relational database management systems. This is the unique identifier _id. It is generated automatically for every document in MongoDB.

Tables are found in the traditional SQL world. Collections are present in the MongoDB world. A wide variety of objects can be contained in a collection. They can have a different structure and a different set of properties.

MongoDB solves several problems. It increases the speed, flexibility and safety of operations. Among the new tools are the following:

- Change streams. This tool automatically captures and streams changes from the DBMS logs. APIs have been developed. Previously, developers had to write separate code for this.
- Retryable writes. It automatically repeats the update operation of the database if it is interrupted for some reason. This will save developers time and reduce the number of failover scenarios by one. Retryable Writes will provide near-constant support for write operations.
- The related function causal consistency allows users to "read" their entries.
- Compass. The tool will allow you to interact with MongoDB through a visual interface instead of the command line. This feature will come from the cloud version of the database - Atlas.

Part 3.1.2 Overview of the NodeJS platform

Applications can run in the Node.js runtime on Microsoft Windows, OS, IBM AIX, Linux, FreeBSD, NonStop, IBM System, and IBM.

Unlike classic JavaScript, executable code in Node.js is executed on the server side (backend), not on the browser side. Node.js uses the V8 engine to interpret the code V8 engine is currently used in Google Chrome.

All mechanisms for processing requests and other input / output (I / O) operations are built on events. There is no way to block the currently running thread in Node.js. Each operation is performed asynchronously in Node.js. This is a huge advantage when code needs to be built around I / O: reading disks, connecting to a database, web services, etc.

Express.js is a framework for a Node.js application server. It allows to create single page, multi-page and hybrid web applications. Express.js is the standard server-side framework for Node.js.

The figure A.2 shows Node.js and Express.js principle of operation inside MEAN stack.

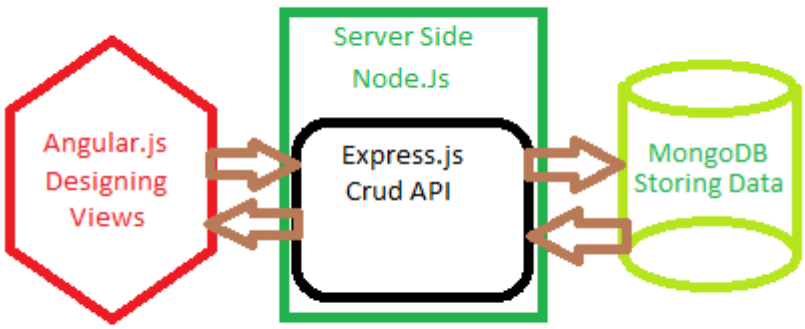

Figure A.2 –Node.js and Express.js principle of operation inside MEAN stack

Millions of front-end developers can write server-side and client-side code in the same programming language. At the same time, there is no need to learn a completely new tool for the transition to server-side development.

The browser and the server use the same language concepts. In addition, developers can quickly migrate to the new ECMAScript standards as they are

implemented on the platform. Server platform that is completely in control of the developer. As a result, new language features become available when a supporting version of Node.js is installed.

Google's open source V8 JavaScript engine is at the heart of Node.js. It is used in the Google Chrome browser and in other browsers. This means that Node.js draws on the work of thousands of engineers. They made the JavaScript Chrome runtime incredibly fast and continue to work towards improving V8.

All instructions are blocking when the code is executed asynchronously in traditional programming languages (C, Java, Python, PHP). The request thread will be suspended until the response is received and processed in that environment.

JavaScript makes it much easier to write asynchronous and non-blocking code using a single thread, callback functions, and an everyday approach to development. For each heavy operation, the callback is passed to the appropriate mechanism and is called immediately after the completion of this operation. There is no need to wait for the results of such operations while the program continues to run. A similar mechanism originated in browsers. Downloading data from the network, processing button clicks, must occur simultaneously.

Asynchronous mechanisms allow a single Node.js server to handle thousands of connections simultaneously. At the same time, without burdening the programmer with tasks for managing flows and organizing parallel code execution. Things like this are often sources of error.

Node.js provides the developer with non-blocking basic I / O mechanisms. Libraries in the Node.js environment are written using non-blocking paradigms. This makes blocking behavior the exception rather than the norm.

Node.js does not block waiting for the results of operations on the main thread. Node.js initiates an I / O operation and continues to do other things until the results of this operation are received.

There are now over half a million open source packages in the npm registry. Any Node.js developer can use them freely.

Part 3.1.3 An overview of Angular framework

Angular is a structural framework for dynamic web applications. This framework uses HTML as its templating language. It extends the HTML syntax to describe application components clearly and concisely. Angular Data Binding and Dependency Injection eliminates the need for a lot of coding.

Angular has adopted the core principles behind the original MVC architecture. This has transformed them into their own way of building client-side web applications. MVC schema separates application logic, view and data in Angular

The figure A.3 shows processing of a request from the client side to the application server occurs when routing through a specific controller. The controller sends a request to the database through the models during processing. The controller generates a response when it received information. The response is sent to the front-end, browser-based part of the application from the controller. The response is a JSON file used to display data on the client side of the application.

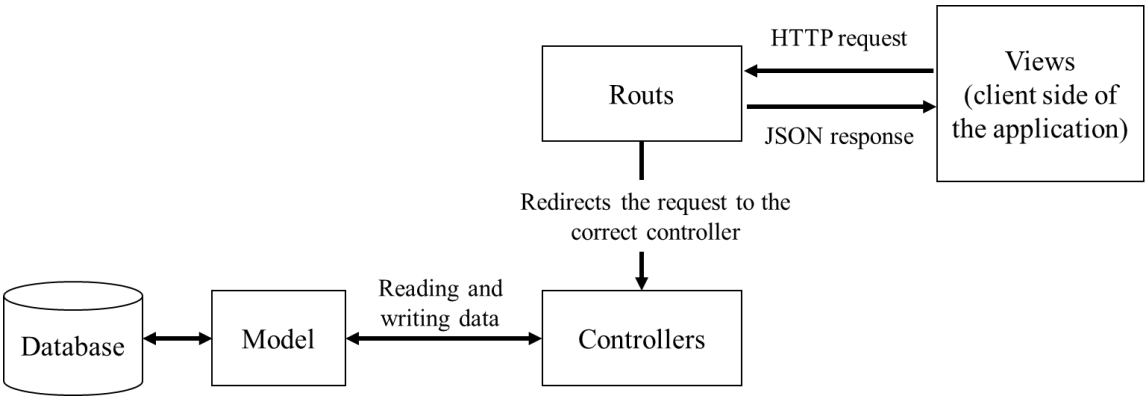

Figure A.3 – MVC architecture implementation diagram

MVC makes Angular applications more structured, easier to maintain, and better automated testability using services and the dependency injection pattern.

Services perform the same function as services in applications on the server in Angular. Services operate as permanent, easily replaceable blocks, objects, or components. Choosing the right framework for JavaScript environment, application, or simple website is a top priority for any business.

Angular allows to create applications that are large and complex in terms of business logic. Angular has been rewritten from AngularJS. The framework has become cleaner and more flexible, more enterprise-like and highly scalable.

The advantages of the framework:

- Support from Google, Microsoft;
- Developer Tools (CLI);
- Unified project structure;
- TypeScript out of the box;
- Reactive programming with RxJS;
- The only Dependency Injection framework out of the box;
- HTML extension based templates;
- Cross-browser Shadow DOM out of the box (or emulating it);
- Cross-browser support for HTTP, WebSockets, Service Workers;
- There is no need to configure anything further. No more wrappers;
- More modern framework than AngularJS (at the React, Vue level);
- Large community.

A drawbacks of the framework:

- Higher entry threshold due to Observable (RxJS) and Dependency Injeciton;
- A lot of time is spent on additional optimizations for the speed and quality of the application;
- There is no architecture for state management out of the box for enterprise applications;
- Angular-Univesal has many pitfalls;
- Dynamic component creation turns out to be a non-trivial task. They are linked together through the dependency injection pattern.

Part 3.2 Overview of REST API operation

A feature of the application is its API. It allows to create a set of links to which information from the online store will be sent during the purchase of a product. Thus, it is possible to integrate the online store into the warehouse management system using the REST API.

REST API is convenient when changing the technology of an online store. The user does not experience additional problems in setting up a new warehouse management system in this case.

The features of the REST architectural style:

• Each entity must have a unique identifier - URI.

• Entities must be related to each other.

Standard methods should be used to read and modify data.

• There must be support for several types of resources.

• Interaction must be stateless.

Standard REST Methods:

• GET - receiving data without changing it.

• POST - create a new object.

• PUT - change existing records.

• PATCH - change the identifier of existing records.

• DELETE - delete records.

There are five mandatory constraints for building distributed REST applications according to Fielding.

These restrictive requirements are mandatory for REST systems. The constraints imposed govern how the server works in how it can process and respond to client requests. Acting within these constraints, the system acquires desirable properties/

If a service application violates any of these constraints, the system cannot be considered a REST system.

The figure A.4 shows REST system scheme.

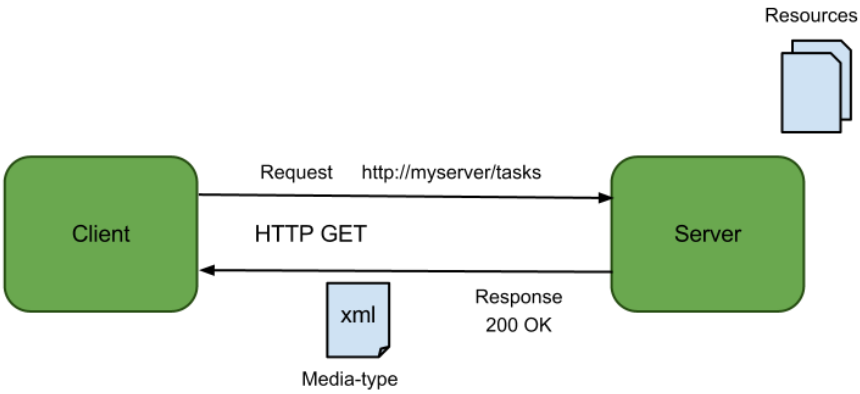

Figure A.4 – REST system scheme

The prerequisites-constraints REST:

1. The differentiation of needs is at the heart of this imposed constraint. Distinguishing the client interface from the server's needs increases the portability of the client interface code to other platforms. Simplifying the server side improves scalability. The greatest impact on the world wide web is the delineation between the individual parts. It helps these parts to develop independently of each other and to support the development needs of the Internet from various organizations.

2. A caching. Clients and intermediate hosts can cache server responses. Proper use of caching can completely or partially eliminate some client-server interactions. At the same time, the performance and expandability of the system is increased.

3.An unified interface. Having a consistent interface is a fundamental requirement for REST services. Unified REST interfaces allow each service to evolve independently.

There are four constraints on unified interfaces:

• Resource identification

• Manipulation of resources through representation

• "Self-describing" messages

• Hypermedia as a means of changing the state of the application

4. A layers. It is impossible to determine whether the client is communicating directly with the server or with an intermediate node due to the hierarchical structure of the networks. It is understood that such a structure forms layers. The use of intermediate servers can increase scalability through load balancing and distributed

caching. Intermediate sites can also be subject to security policies to maintain confidentiality of information.

5. An optional restriction. REST can extend the functionality of the client. An additional constraint allows the architecture to be designed and the desired functionality to be maintained in general, except in some contexts.

REST APIs often use text format to represent the state of a resource as a set of meaningful fields. The most commonly used combination of two technologies today. It is the REST architecture style and the JSON schema. Curly braces are used to hierarchically structure information in a JSON-document.

Приложение Б. Схема взаимодействия бизнес-процессов интернет-магазина для работы с приложением

Приложение В. Листинг TypeScript кода компонента VendorListComponent сущности «Поставщик»

```
import { Component, EventEmitter, Input, OnInit, Output } from "@angular/core"
import { VendorsService } from "../shared/services/vendors.service"
import { Vendor } from "../shared/interfaces"
import { TemplateService } from "../shared/services/template.service"
import { Router } from "@angular/router"

@Component({
  selector: "app-vendor-list",
  templateUrl: "./vendor-list.component.html",
  styleUrls: ["./vendor-list.component.scss"],
})
export class VendorListComponent implements OnInit {
  loading = false
  vendors: Array<Vendor> = []
  templates: Object = {}

  @Output("onChangeEvent") onChange = new EventEmitter<Object>()
  constructor(
    private vendorService: VendorsService,
    private templateService: TemplateService,
    private router: Router
  ) {}

  ngOnInit() {
    this.loading = true
    this.vendorService.fetch().subscribe((vendors) => {
      this.vendors = vendors
      this.templateService.getByType("vendor").subscribe((templates) => {
        this.templates = templates
        this.loading = false
      })
    })
  }
  createVendor() {
    this.onChange.emit("none")
  }
  updateSelected(id) {
    this.onChange.emit(id)
  }
}
```

Приложение Г. Листинг HTML кода карточки отображения сущности по заданному шаблону

```
<div class="card">
  <div class="card-content">
    <table class="highlight">
      <tbody>
        <tr *ngFor="let field of showData; let i = index">
          <td>{{ showNames[i] }}</td>
          <td>{{ showData[i] }}</td>
        </tr>
      </tbody>
    </table>
  </div>

  <div class="card-action">
    <a (click)="onAction()">Подробнее</a>
  </div>
</div>
```

Приложение Д. Листинг TypeScript кода карточки отображения сущности по заданному шаблону

```
import { Component, EventEmitter, Input, OnInit, Output } from "@angular/core"
import { Router } from "@angular/router"
import { Template } from "../../interfaces"

@Component({
  selector: "app-table-card",
  templateUrl: "./table-card.component.html",
  styleUrls: ["./table-card.component.scss"],
})
export class TableCardComponent implements OnInit {
  @Input() data: any
  @Input() template: Template

  @Output("onChangeEvent") onChange = new EventEmitter<Object>()

  showData: Array<string> = []
  showNames: Array<string> = []

  constructor(private router: Router) {}

  ngOnInit() {
    for (let item of this.template.options) {
      if (item.show) {
        this.showData.push(this.data.options[item.name])
        this.showNames.push(item.translation)
      }
    }
  }

  camelToText(text: string) {
    return text.replace(/([A-Z])/g, " $1").replace(/^./, function (str) {
      return str.toUpperCase()
    })
  }
  textToCamel(text: string) {}
  onAction() {
    if (this.template.type == "check") {
      this.router.navigate([`menu/check/${this.data._id}`])
    } else this.onChange.emit(this.data._id)
  }
}
```

Приложение Е. Листинг TypeScript кода формы для создания и редактирования сущности

```
import { Component, EventEmitter, Input, OnInit, Output } from "@angular/core"
import { FormArray, FormControl, FormGroup, Validators } from "@angular/forms"
import { Template } from "../../interfaces"
import { MaterialService } from "../../services/Material.service"
import { DateAdapter } from "@angular/material/core"

@Component({
  selector: "app-form-template",
  templateUrl: "./form-template.component.html",
  styleUrls: ["./form-template.component.scss"],
})
export class FormTemplateComponent implements OnInit {
  @Input() template: Template
  @Input() data: any
 @Output("onChangeEvent") onChange = new EventEmitter<Object>()
  loading = false
  form: FormGroup

  constructor(private dateAdapter: DateAdapter<Date>) {
    this.dateAdapter.setLocale("en-GB") //dd/MM/yyyy
  }
  ngOnInit() {
    this.loading = true
    this.form = new FormGroup({
      fields: new FormArray([]),
    })
    for (let item of this.template.options) {
      let initValue: any = ""
      if (this.data && this.data.options && this.data.options[item.name]) {
        if (
          String(this.data.options[item.name]).replace(/[0-9]/g, "") == "//"
        ) {
          let date = this.data.options[item.name].split("/")
          initValue = new Date(date[2], date[1] - 1, date[0])
        } else if (
          this.data.options[item.name] ||
          this.data.options[item.name] === 0
        ) {
          initValue = this.data.options[item.name]
        }
      }
```

```
      const optionsItem = new FormGroup({
        [item.name]: new FormControl(initValue, Validators.required),
      })
      ;(this.form.get("fields") as FormArray).push(optionsItem)
    }
    MaterialService.updateTextInputs()
    this.loading = false
  }
  oNchange() {
    this.onChange.emit(this.dataForSent(this.form.value))
  }
  dataForSent(formdata) {
    //перевод времени
    let data: any = {
      valid: this.form.valid,
      options: {},
    }
    for (let i = 0; i < this.template.options.length; i++) {
      if (this.template.options[i].validators == "date") {
        let date: any = new Date(
          formdata.fields[i][this.template.options[i].name]
        )
        let dd = String(date.getDate()).padStart(2, "0")
        let mm = String(date.getMonth() + 1).padStart(2, "0")
        let yyyy = date.getFullYear()
        date = dd + "/" + mm + "/" + yyyy
        data.options[this.template.options[i].name] = date
      } else {
        data.options[this.template.options[i].name] =
          formdata.fields[i][this.template.options[i].name]
      }
    }
    return data
  }
}
```