МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2020 г.

Веб-приложение "СкладКонтроль"

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____И.Л. Надточий
«___»_____2020 г.

Автор работы,
студент группы КЭ-405
_____Д.В. Царенок
«___»_____2020 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2020 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Царенку Даниилу Викторовичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Веб-приложение "СкладКонтроль"» утверждена приказом по университету от 24.04.2020. № 627

2. **Срок сдачи студентом законченной работы:** 1 июня 2020 г.

3. **Исходные данные к работе:**

Задачей данной работы является разработка веб-приложения, с помощью, которого несколько пользователей могут, используя веб-браузер, вести складской учет, будучи не привязанными к ПК, на котором установлена база данных. Также задачей является переход на новое программное обеспечение, не зависящее от мощностей ПК.

4. **Перечень подлежащих разработке вопросов:**

1. Анализ предметной области и существующих решений.

2. Разработка технического задания для разработки веб-приложения для учета и контроля товара на складах с доступом через веб-браузер.

3. Выбор среды и средств реализации, разработка основных архитектурных решений.

4. Разработка дизайна веб-приложения для учета и контроля товара на складах.

5. Проектирование структуры базы данных.

6. Реализация пользовательской и серверной частей приложения.

7. Тестирование разработанного веб-приложения учета и контроля товара на складах.

5. **Дата выдачи задания:** 1 декабря 2019 г.

Руководитель работы _____/*И.Л. Надточий*/

Студент _____/*Д.В. Царенок* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2020	
Разработка модели, проектирование	01.04.2020	
Реализация системы	01.05.2020	
Тестирование, отладка, эксперименты	15.05.2020	
Компоновка текста работы и сдача на нормоконтроль	24.05.2020	
Подготовка презентации и доклада	30.05.2020	

Руководитель работы _____/*И.Л. Надточий*/

Студент _____/*Д.В. Царенок*/

Аннотация

Д.В. Царенок. Методические указания к выполнению выпускных квалификационных работ. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2020, 94 с., 17 ил., библиогр. список – 22 наим.

В данной выпускной квалификационной работе выполнена разработка веб-приложения для учета и контроля товара на складах. В ходе работы был произведен обзор существующих аналогов и выявлены их плюсы и минусы. Также было разработано техническое задание, на основе которого реализован основной функционал веб-приложения и произведено тестирование программы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Склад является неотъемлемой структурной составляющей любой производственной или торговой организации. Сложившаяся практика формирования логистики в торговых компаниях предполагает внедрение службы контроля за товарами в складских помещениях. Склад выполняет ряд логистических функций (складирование, грузопереработка, упаковка, снабжение, транспортировка, физическое распределение и т.д.) путем реализации соответствующих логистических операций.

Для удобной и быстрой работы с товаром на складе современные предприятия, вне зависимости от их размеров и оборотов капитала, используют специально разработанное ПО, позволяющее вести складской учет, при этом не требуя от оператора особых навыков, кроме базового умения работы с ПК.

В данных программах главным объектом является товар. Поэтому в модели выделим следующие основные бизнес-процессы, через которые проходит материальный поток на складе:

1) получение / прием товара - прием, проверка соответствия поставки сопроводительным документам (накладным) и целостности товара;

2) хранение товара - определение мест хранения поступающих на склад товаров, сортировка, построение оптимальных маршрутов, размещение грузов в зоне хранения;

3) комплектация товара - отбор товара из зоны хранения, комплектация и упаковка;

4) отгрузка товара - формирование партий отгрузки и отгрузки товара клиентам.

Данные процессы также сопровождаются документооборотом: Накладные на приход/отпуск.

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью данного проекта является разработка веб-приложения, с помощью, которого несколько пользователей могут, используя веб-браузер, вести складской учет, будучи не привязанными к ПК и его мощностям, на котором установлена база данных.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. Рассмотреть существующие на рынке проекты.
2. Провести детальный анализ найденных проектов с точки зрения их достоинств и недостатков, учесть полученные результаты при разработке.
3. Определить основной функционал.
4. Сформулировать основные сущности проекта.
5. Выбрать средства реализации проекта.
6. Разработать схему данных.
7. Разработать серверную часть приложения, опираясь на ранее созданную схему данных.
8. Разработать клиентскую часть приложения.
9. Протестировать итоговую версию продукта.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. ОБЗОР АНАЛОГОВ

Для создания информативного, удобного для пользователя и конкурентоспособного приложения, необходимо рассмотреть уже имеющиеся на рынке решения, проанализировать их достоинства и недостатки. Рассмотрим следующие проекты:

1. 1С Склад [6]

Данное приложение является частью большой ERP системы 1С. На данный момент является одной из самых популярных программ на территории РФ, поскольку обладает богатым инструментарием для управления логистикой предприятия. Однако то что это большая комплексная программа является её как достоинством, так и недостатком, поскольку она не подходит для малого бизнеса и самостоятельных индивидуальных предпринимателей из-за того что имеет большую цену за лицензию, и сложный интерфейс, из-за чего у программы высокий порог вхождения, и она требует много времени на обучение пользователя. На рынке труда крупные копании ищут специалистов, которые специализируются на работе в 1С. Не имеет веб версии и требует установки на ПК.

2. SAP [7]

Данное приложение является ERP системой, и помимо управления складом имеет большой функционал, обеспечивающий управление предприятием. Использует в основном дочерними предприятиями, с главными офисами за пределами РФ. Являет более современным решением по сравнению с 1С, однако имеет и общие недостатки: высокий порог вхождения и высокая цена. Многие пользователи жалуются на то, что

программа слишком сложна в освоении. Не имеет веб версии и требует установки на или сервер, что сопровождается дополнительными расходами, однако имеет собственное мобильное приложение.

3. СуперСклад [9]

Данная программа позиционируется как простая программ складского учета. Программа ориентирована исключительно на складской учет и не имеет никакой возможности кастомизации. Требует установки приложения на ПК и не сопровождается мобильным приложением. Является платным продуктом, не имеющей бесплатной версии.

4. Мой Склад [8]

Данная программа является веб-приложением, и не требует привязки к ПК. Имеет простой и понятный пользовательский интерфейс, и имеет низкий порог вхождения. На сегодняшний день является одним из лучших веб-приложений на российском рынке. Однако оно не имеет никакой возможности кастомизации, не ведет никакую статистику и не предлагает аналитических данных. И основным недостатком является цена, так как требует доплаты за весь предлагаемый функционал и количество пользователей.

1.2. АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

1.2.1 ВЫБОР ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

При выборе языка программирования рассмотрим языки наиболее популярных среди IT-проектов:

1) Python

Преимущества:

• довольно простой в изучении;

• более высокий уровень абстракции позволяет писать многие операции за меньшее число строк кода нежели языки среднего уровня;

• присутствует функциональное программирование и ООП;

• в стандартных библиотеках Python есть средства для работы с электронной почтой, FTP, HTTP, базами данных, и пр.;

• скрипты, написанные при помощи Python выполняются на большинстве современных ОС. Такая переносимость обеспечивает Python применение в самых различных областях;

• Python подходит для любых решений в области программирования, будь то офисные программы, веб-приложения, GUI-приложения и т. д.

Недостатки:

• динамическая типизация;

• значительно медленнее Си и ASM, как и большинство скриптовых языков.

2) Java

Преимущества:

• является эталонным ООП языком;

• строго типизированный язык программирования, что гарантирует стабильную работу в большом проекте;

• Java наследует подход языка Си, что позволяет человеку знающему последний быстро изучить Java;

• JVM. JVM определяет сразу несколько преимуществ: независимость от платформы, безопасность (так как программа исполняется не на прямую в среде операционной системы, а в виртуальной машине, соответственно все ошибки остается внутри её, а внешние инъекции кода просто невозможны);

• как компилируемый язык обладает сборщиком мусора;

• в стандартной библиотеке Java есть возможность работы с потоками;

• огромное сообщество, которое имеет большую базу знаний по Java и её фреймворкам.

Недостатки:

• низкая производительность. Но данный недостаток относится лишь к небольшим программам, запускаемым очень часто. Так как JVM – это большая абстрактная машина, то соответственно, чтобы ей подняться, требуется время. Но при разработке сервера, который планируется держать большое количество времени с возможностью масштабирования данный недостаток является пренебрежимо малым;

• многословный код.

3) JavaScript

Преимущества:

• JavaScript предоставляет большое количество возможностей для решения самых разнообразных задач. Гибкость языка позволяет использовать множество шаблонов программирования применительно к конкретным условиям;

• популярность JavaScript открывает перед программистом немалое количество готовых библиотек, которые позволяют значительно упростить написание кода и нивелировать несовершенства синтаксиса;

• применение во многих областях. Широкие возможности JavaScript дают программистам шанс попробовать себя в качестве разработчика самых разнообразных приложений.

Недостатки:

• необходимость обеспечивать кроссбраузерность. Код должен корректно выполняться во всех, или хотя бы самых популярных, браузерах;

• отсутствует стандартная библиотека. JavaScript не предоставляет никаких возможностей для работы с файлами, потоками ввода-вывода и прочими полезными вещами;

• синтаксис в целом затрудняет понимание. Красота и логичность кода – не преимущество JavaScript.

4) PHP

Преимущества:

• разработка с помощью PHP дает много возможностей. При должном уровне владения, с помощью шаблонизатора можно создавать не только сценарии для веб-приложений, но и полноценные программы;

• кроссплатформенность. PHP может быть запущен в любой операционной системе, включая юниксоиды;

• поддержка почти всех веб-серверов;

• невысокие расходы на разработку.

Недостатки:

• безопасность. У PHP есть средства безопасности уровня системы и уровня веб-приложения. Но, опять же, широкая используемость сыграла злую шутку: дыры в PHP находят быстрее, чем разработчики успевают их закрывать. В PHP 7 множество проблем решено, но злоумышленник всегда впереди. В силу того, что массы знают «препроцессор», трудно предугадать всё;

• противоречия в коде. Когда шаблонизатор был только создан, все программное обеспечение разрабатывались с помощью C. Потому в языке было применено множество синтаксиса из него. В то же время, современная аудитория больше сконцентрирована на Java. В итоге, код переполнен различными остатками из разных языков. И все они могут даже быть сконцентрированы в одном выражении кода.

5) C#

Преимущества:

• для небольших компаний и некоторых индивидуальных разработчиков бесплатными являются такие инструменты, Visual Studio, облако Azure, Windows Server, Parallels Desktop для Mac Pro и многие другие;

• большое количество синтаксической поддержки, представляющего собой специальные конструкции, разработанные для понимания и написания кода. Они не имеют значения при компиляции;

• порог вхождения у языка C# низкий. Его синтаксис имеет много схожего с другими языками программирования, благодаря чему облегчается переход для программистов. Язык C# считается наиболее понятным и подходящим для новичков;

• после покупки Xamarin на C# можно писать программы и приложения для таких операционных систем, как iOS, Android, MacOS и Linux.

Недостатки:

• приоритетная ориентированность на Windows платформу;

• язык бесплатен только для небольших фирм, индивидуальных программистов, стартапов и учащихся. Крупной компании покупка лицензионной версии этого языка стоит больших затрат;

• в языке осталась возможность использования оператора безусловного перехода.

C# не был выбран из-за ориентированности на Windows платформу, а платформа .NET ещё развивается в области Web.

JavaScript, PHP и Python не являются строго-типизированными языками. При разработки большого алгоритма данного проекта, возможны Runtime

Exception из-за приведения типов и тому подобное. Но все данные языки в купе с фреймворками дают относительно быстрый результат. Но незнание python потребует время на его изучение, так как он не Си подобный язык. Javascript часто используется в роли бекэнд языка, но в роли инструмента для описания сложной алгоритмической логики данный язык будет не лучшим вариантом, однако он подойдёт для разработки удобного клиентского приложения.

Java – строго-типизированный ООП язык. PHP с 5 версии также стал ООП направленным. При выборе между данными языками следует обратиться к критериям скорости разработки, возможность работы в веб, безопасность, масштабируемость. Скорость работы на PHP при работе с проектами простой логики, например интернет-магазин или новостной блог, естественно будет ощутимой по сравнению с Java. И PHP, и Java имеют инструменты для работы в вебе.

Немаловажным фактом будет являться то, что разработчик проекта является PHP-программистом с опытом работы.

Для работы были выбраны языки программирования PHP для серверной части и JavaScript для клиентской части приложения.

1.2.2 ВЫБОР ФРЕЙМВОРКОВ

Laravel:

Эта специфическая структура известна своим элегантным синтаксисом, который легко понять и с которым приятно работать. В Lavarel можно быстро приступить к работе над проектом. Имеется доступ к таким функциям, как аутентификация пользователей, управление сеансами и кэширование. В целом, Laravel обладает всеми функциональными возможностями, которые понадобятся для создания современного PHP-приложения. Ядро Laravel

является надёжным с точки зрения производительности, и можно расширить платформу, используя множество дополнений. Laravel также прекрасно интегрируется с другими сторонними библиотеками и платформами, что позволяет создавать высокомасштабируемые приложения. Для долгосрочных задач есть возможность поставить их в очередь для асинхронного выполнения в фоновом режиме, что ещё больше повышает производительность.

Именно поэтому для работы серверной часть будет использован Laravel.

Для отображения всех результатов работы будет использоваться веб-интерфейс. Поскольку JavaScript используется исключительно для клиентской части, не рассматриваются такие фреймворки как: Java Spring Framework, Java Spark, JavaEE. Сейчас на рынке фреймворков клиента существует множество продуктов для создания веб-приложений под разные задачи. Практически все они завязаны под язык JavaScript. Перечислим их и их особенности, и недостатки каждого в рамках данного проекта. [13]

1) Vue.js [22]

Vue.js (также называемый Vue) представляет собой среду JavaScript с открытым исходным кодом, предназначенную для упрощения и упорядочения разработки пользовательского интерфейса. Фреймворк, также называемый идеальной смесью Angular и React, зарекомендовал себя как идеальный выбор для разработки приложения с двухсторонней привязкой данных к структуре углов и серверному рендерингу React JS framework. Необходимо отметить, что Vue.js был номинирован как самый популярный JavaScript интерфейс на GitHub с 118 тысячами звёзд в прошлом году. Vue.js может показаться идеальной JavaScript структурой для разработки программного обеспечения (ПО), однако у нее также есть свои плюсы и минусы.

Преимущества

• быстрый набор популярности: всего за несколько лет с момента своего создания большое количество предприятий добавили Vue.js в свой технический стек;

• быстрая настройка: Vue имеет встроенную привязку данных и модель MVC (модель, вид, контроллер), что значительно упрощает настройку по сравнению с Angular.js и React.js;

• более простая интеграция: платформа поддерживает более легкую интеграцию с элементами HTML;

• маленькая кривая обучения: по сравнению с Angular JS Framework, Vue намного легче изучать, понимать и использовать.

Недостатки

• мало ресурсов: структура по-прежнему слишком молода для поиска полезных решений в Интернете и самообучения.

2) React

Поддерживаемый Facebook, Instagram и другими известными организациями, React является одним из лучших JavaScript фреймворков на протяжении последних 5 лет. Также называемая React.js или React JS, инфраструктуру frontend использует более чем 38% разработчиков по всему миру. Netflix, Flipboard, PayPal и BBC являются первыми организациями, которые использовали React.

Преимущества

• множество документации и онлайн-ресурсов. Благодаря поддержке Facebook существует множество возможностей использовать тонну документации и онлайн-ресурсов для обучения и использования фреймворка Javascript для React;

• быстрая, гибкая, эффективная и лёгкая технология: система JS широко рекомендуется благодаря своей эффективности, небольшому размеру блоков, гибкости и более быстрому подходу к работе благодаря простой компонентной модели и функциональности рендеринга на стороне сервера;

• переход между версиями. Миграция между версиями, как правило, очень проста, Facebook предоставляет “codemods” для автоматизации огромной части процессов;

• отлично себя чувствует при работе с ES6/7 ReactJS, может брать на себя любую нагрузку;

• структура имеет компонентную архитектуру, которая революционизировала веб-разработку приложений и повлияла на другие технологии;

• DOM (Document Object Model – «объектная модель документа») позволяет объединять HTML, XHTML или XML документы по определенным критериям, чаще всего в дерево, поэтому React отлично подходит для веб-браузеров при анализе разнообразных элементов веб-приложений.

Недостатки

• необходимы средства сборки: данная инфраструктура JavaScript может работать некорректно без адекватных инструментов сборки или может отображать несовместимость с другими библиотеками и кодами из-за высокой DOM;

• большая кривая обучения: в отличие от Vue, React требует больше време-ни для изучения концепций и реализации. React JS требует огромное количество знаний в том как интегрировать пользовательский интерфейс в структуру MVC;

• отсутствие упорядоченной документации - сверхбыстрый обмен решениями в ReactJS не оставляет места для упорядочения документации, документы размещены немного хаотично, поскольку многие разработчики индивидуально вносят их в базу данных без какого-либо систематического подхода.

3) Angular.js

Angular.js – это полнофункциональная интерфейсная JavaScript среда, поддерживаемая Google и другими популярными корпорациями. Эта структура известна своими потенциальными возможностями, например, быстрым созданием кода, двусторонней привязкой данных, тестированием частей приложения. Рассмотрим плюсы и минусы этой технологии:

Преимущества:

• двусторонняя привязка данных;

• мобильный подход к веб-разработке;

• поддержка PWA (Progressive Web App – «прогрессивное веб-приложение»);

• стабильная и долгосрочная поддержка Google;

• универсальный MVVM(Model-View-ViewModel) модуль (есть возможность использовать одни и те же данные в одной части приложения);

• взаимозависимость функций, в виду их связанности с компонентами и модулями;

• RXJS, молниеносная компиляция (менее 2,9 секунд), изменённый пуск HttpClient.

Недостатки:

• плохая оптимизация: приложения на основе Angular.js требуют большей оптимизации для решения проблем с низкой производительностью;

• большая кривая обучения: Angular показывает высокую шкалу обучения, вам нужно больше времени, чтобы освоить эту структуру;

• интеграционные ошибки, которые могут возникать при переходе от старой версии к новой;

• достаточно сложный язык программирования

Для реализации проекта достаточен фреймворк с достаточно низким порогом вхождения. Таковым является Vue.js, а также у данной технологии большая документация на русском языке.

1.2.3 ВЫБОР СУБД

Базы данных – это специально разработанное хранилище для различных типов данных. Каждая база данных имеет определённую модель (реляционная, сетевая, документно-ориентированная и др.), которая обеспечивает удобный доступ к данным. Системы управления базами данных (СУБД) – специальные приложения (или библиотеки) для управления базами данных различных размеров и форм.

Критериями для выбора СУБД были: производительность, стабильность, масштабируемость, возможность разработки веб-приложений, качество и полнота документации, поддержка архитектуры клиент-сервер, совместимость работы с .NET Core.

Нередко возникает ситуация, когда модель меняется. Но при этом уже имеется существующая база данных, в которой есть какие-то данные. Чтобы без потерь обновить базу данных, существует такой механизм как миграция. Функция миграции в Entity Framework Core позволяет последовательно

применять изменения схемы к базе данных, чтобы синхронизировать ее с моделью данных в приложении без потери существующих данных.

Миграции включают средства командной строки и API-интерфейсы, которые помогают в решении следующих задач:

1. Создание миграции.

2. Обновление базы данных.

3. Настройка кода миграции.

4. Удаление миграции.

5. Отмена миграции.

6. Создание скриптов SQL.

7. Применение миграции во время выполнения.

1) SQLite

Легко встраиваемая в приложения СУБД. Так как эта система базируется на файлах, она предоставляет довольно широкий набор инструментов для работы с ней по сравнению с сетевыми СУБД. При работе с этой СУБД обращения происходят напрямую к файлам (в этих файлах хранятся данные) вместо портов и сокетов в сетевых СУБД. Именно поэтому SQLite очень быстрая, а также мощная благодаря технологиям обслуживающих библиотек.

Преимущества SQLite:

1. Файловая структура - вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины.

2. Используемые стандарты – хотя может показаться, что эта СУБД примитивная, но она использует SQL. Некоторые особенности опущены (RIGHT OUTER JOIN или FOR EACH STATEMENT), но основные все-таки поддерживаются.

3. Отлично подходит при разработке и тестировании. SQLite предлагает всё, что необходимо для этих целей, так как состоит всего из одного файла и библиотеки, написанной на языке C.

Недостатки SQLite:

1. Отсутствие системы пользователей – более крупные СУБД имеют системы управления правами доступа пользователей. Обычно применение этой функции не критично, так как эта СУБД используется в небольших приложениях.

2. Отсутствие возможности увеличения производительности – опять, исходя из проектирования, довольно сложно выжать что-то более производительное из этой СУБД.

Таким образом, SQLite стоит использовать, когда важна возможность легкого переноса приложения и не важна масштабируемость, когда необходимо напрямую обращаться к диску, а также есть необходимость использования дополнительных процессов при тестировании. Но если речь идет о многопользовательском приложении, в котором доступ к данным осуществляется несколькими пользователями одновременно, да к тому же присутствует разграничение прав пользователей либо идет работа с большим объемом данных, то данная СУБД будет неприемлема для работы.

2) MySQL

Это одна из самых распространенных полноценных серверных СУБД. MySQL очень функциональная, свободно распространяемая СУБД, которая успешно работает с различными сайтами и веб-приложениями. Обучиться использованию этой СУБД довольно просто, так как существует большое количество документации. Несмотря на то, что в ней не реализован весь SQL функционал, MySQL предлагает довольно много инструментов для разработки

приложений. Так как это серверная СУБД, приложения для доступа к данным, в отличии от SQLite работают со службами MySQL.

Преимущества MySQL:

1. Простота в работе – установить MySQL довольно просто. Дополнительные приложения, например GUI, позволяет довольно легко работать с БД.

2. Богатый функционал – MySQL поддерживает большинство функционала SQL.

3. Безопасность – большое количество функций, обеспечивающих безопасность, которые поддерживаются по умолчанию.

4. Масштабируемость – MySQL легко работает с большими объемами данных и легко масштабируется.

5. Скорость – упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность.

Недостатки MySQL:

1. Известные ограничения – по задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях.

2. Проблемы с надежностью – из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по надежности.

Таким образом, MySQL можно применять, когда важными критериями являются безопасность, работа с распределенными операциями, работа с веб-приложениями, а также масштабируемость.

3) PostgreSQL

PostgreSQL является самой профессиональной из выше рассмотренных СУБД. Она свободно распространяемая и максимально соответствует стандартам SQL. В PostgreSQL стараются полностью применять ANSI/ISO SQL стандарты своевременно с выходом новых версий.

От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, то есть. атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability (ACID).) Благодаря мощным технологиям PostgreSQL имеет высокий уровень производительности. Параллельность достигнута не за счет блокировки операций чтения, а благодаря реализации управления многовариантным параллелизмом (MVCC), что также обеспечивает соответствие ACID. PostgreSQL очень легко расширять своими процедурами, которые называются хранимые процедуры. Эти функции упрощают использование постоянно повторяемых операций.

Достоинства PostgreSQL:

1. Открытое программное обеспечение, соответствующее стандарту SQL. PostgreSQL – бесплатное программное обеспечение (ПО) с открытым исходным кодом. Эта СУБД является очень мощной системой.

2. Большое сообщество – существует довольно большое сообщество, в котором можно найдёте ответы на интересующие вопросы.

3. Большое количество дополнений – несмотря на огромное количество встроенных функций, существует очень много дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими.

4. Расширения – существует возможность расширения функционала за счет сохранения своих процедур.

5. Объектность – PostrgreSQL это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования и много другого.

Недостатки PostgreSQL:

1. Производительность – при простых операциях чтения PostgreSQL может значительно замедлить сервер и быть медленнее своих конкурентов.

2. Популярность – популярностью эта СУБД похвастаться не может, хотя и присутствует довольно большое сообщество.

3. Хостинг – в силу вышеперечисленных факторов иногда довольно сложно найти хостинг с поддержкой этой СУБД.

Таким образом, если основными критериями при выборе СУБД будут целостность данных, использование сложных пользовательских процедур и сложных структур данных, то можно выбрать PostgreSQL. Но если скорость, простая настройка и репликации являются так же важными критериями, то лучше выбрать другую СУБД.

Таким образом, Microsoft SQL Server и MySQL удовлетворяют всем критериям выбора СУБД для разработки веб – приложения. Поскольку у разработчика имеется опыт работы с MySQL будет выбрана именно она.

1.3. ВЫВОД

Языки программирования:

- Backend: PHP 7.2 - это распространенный язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML;

- Frontend: JavaScript - (кратко: "JS") - это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах.

Фреймворки:

- PHP фреймворк Laravel v.5.4 - бесплатный фреймворк для удобной разработки веб-приложений на языке PHP, использующий архитектурную модель MVC (Model View Controller - модель-представление-контроллер);

- JS фреймворк VueJS - фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек.

СУБД:

- MySQL версии 8.0.17, которая предустановлена на большинстве хостингов и стабильно работает на них.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

ПОЛЬЗОВАТЕЛЬСКИЕ РОЛИ

Каждый пользователь должен иметь одну из следующих ролей, которая будет определять доступный функционал приложения:

1. Неавторизованный пользователь. (Guest)

Получает доступ к сайту в режиме онлайн. Может просматривать главную страницу с основной информацией о сервисе.

Для того чтобы продолжить использование веб-приложения необходимо пройти процедуру авторизации. Вход осуществляется посредством ввода логина и пароля. Если пользователь не имеет аккаунт, то он имеет возможность пройти процедуру регистрации. При регистрации ему необходимо указать следующую информацию: логин – имя пользователя в системе, почта, название организации, пароль.

В случае если пользователь не может получить доступ к аккаунту возможно запросить восстановление пароля.

2. Зарегистрированный пользователь. (User)

Зарегистрированный пользователь является основной ролью для получения доступа к функционалу приложения. Пользователь имеет возможность управлять своим аккаунтом и своими персональными данными: он может их добавлять, редактировать и удалять. При необходимости предусмотрена возможность изменить пароль.

Также зарегистрированный пользователь получает доступ к основному функционалу приложения: управление складами и товарами, получение аналитики и статистики, создание документов.

3. Администратор (Admin)

Администратор — это специальная роль для управления пользователями и модулями приложения. В случае необходимости администратор может временно отключить модуль приложения, до устранения возникших ошибок. Также администратор может добавить и удалить пользователя. Для возможности восстановления аккаунта удаление проходит в режиме «soft delete» то есть аккаунт не удаляется из базы данных, однако доступ к нему из веб-интерфейса получить невозможно.

СТРУКТУРА СИСТЕМЫ

Необходимо обеспечить следующие характеристика работы склада:

1. Основными объектами в программе является товар и склад.
2. Товар хранится на складе, причем складов может быть несколько.
3. Одинаковый товар может храниться на нескольких складах.
4. Товар не может храниться без склада.
5. Склад поделен на зоны. Зоны можно добавлять, удалять и перераспределять.
6. Возможность оформить возврат уже отправленного товара на склад с указанием причины возврата.
7. Возможность оформить возврат товара со склада поставщику.

Согласно техническому заданию, программа должна иметь следующий функционал:

1) Работа со складами:
   - создание нового склада;
   - просмотр и редактирование существующего склада;

- удаление склада;
- резервное копирование и восстановление;
- сохранение последних изменений (в течении недели) с возможностью отката.

2) Работа с товарами:
   - просмотр имеющихся товаров;
   - добавление товара;
   - редактирование информации о товаре;
   - удаление товара;
   - перемещение товара между складами;
   - оформление возврата товара на склад;
   - оформление возврата товара поставщику.
3) Информация о товаре:
   - артикул;
   - группа;
   - название;
   - количество;
   - цена оптом;
   - цена розница;
   - расположение на складе;
   - дата поставки;
   - поставщик;
   - срок годности (при наличии);
   - не для продажи (в случае просрочки, брака, возврата и т.п.);
   - дополнительная информация.
4) Создание документов:

- накладная о прибытии товара;
- накладная об отбытии товара;
- документ об имеющихся товарах и их артикулах;
- документ о наличии товара по выбранным критериям (группы, наименования, цены и т.д.);
- документ о возврате товара на склад;
- документ о возврате товара поставщику;
- документ об отсутствующем товаре;
- документ о товарах с истекшими сроками годности;
- документ о товарах со сроками годности, с выделением товаров с истекающими сроками годности.

5) Аналитика и статистика:

- данные о популярности товара (количества и частоты отгрузки);
- финансовые данные о прибыли;
- данные о нереализованных товарах (с истекшими сроками годности, браком, возвратом и т.п.);
- данные о надёжности поставщика (наличие брака, недостачи).

2.2 ТРЕБОВАНИЯ К НАДЕЖНОСТИ

Для предотвращения потери информации необходимо реализовать автоматическое создание резервных копий по определенному графику, установленному администратором системы, и по окончании сессии пользователя. Резервные копии должны быть на сервере.

Для обеспечения надежности и правильной работы системы необходима валидация всех приходящих на сервер данных.

2.3. ТРЕБОВАНИЯ К БЕЗОПАСНОСТИ И ЗАЩИТЕ ИНФОРМАЦИИ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

Должно быть четкое разграничение доступа к инструментам пользователей. Простой пользователь не должен иметь возможность пользоваться системой администратора. Доступ к данным должен осуществляться в соответствии с правилами категорий пользователей.

Все пароли пользователей должны быть зашифрованы любым алгоритмом.

Учетные записи пользователей должны блокироваться после 5 неудачных попыток захода. Исключение делается только для главной учетной записи администратора.

Администратор должен иметь возможность запретить пользователям вход в систему.

Все пароли должны содержать буквы и цифры, минимальная длина пароля должна быть не менее восьми символов.

В случае если пользователь сообщает о том, что кто-то посторонний имеет доступ к его аккаунту необходима возможность временной блокировки аккаунта.

2.4. ТРЕБОВАНИЯ К ТЕХНИЧЕСКОЙ ЭСТЕТИКЕ

Дизайн веб-приложения должен удовлетворять следующим требованиям по эргономике и технической эстетике:

1. Иметь интуитивно понятный интерфейс.

2. Иметь адаптивный дизайн, чтобы обеспечить корректное отображение при всех возможных разрешениях монитора.

3. Сохранять идентичность отображения на большинстве современных ОС и браузерах.

4. Обладать системой подсказок в местах, где у пользователя потенциально могут возникнуть затруднения.

2.5. ТРЕБОВАНИЯ К ПАТЕНТНОЙ ЧИСТОТЕ

Должна быть обеспечена патентная чистота разрабатываемой информационной системы и ее частей согласно действующему законодательству Российской Федерации.

3. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

3.1. АРХИТЕТУРА ПРОГРАММЫ

В основе работы приложения лежит модель взаимодействия клиент-сервер, которая позволяет разделять функционал и вычислительную нагрузку между клиентскими приложениями (заказчиками услуг) и серверными приложениями (поставщиками услуг). Клиент и сервер какого-либо ресурса могут находиться как в рамках одной вычислительной системы, так и на различных компьютерах, связанных сетью.

Клиент и сервер взаимодействую друг с другом в сети Интернет или в любой другой компьютерной сети при помощи различных сетевых протоколов, например, IP протокол, HTTP протокол, FTP и другие. Например, при помощи HTTP протокола браузер отправляет специальное HTTP сообщение, в котором указано какую информацию и в каком виде он хочет получить от сервера, сервер, получив такое сообщение, отсылает браузеру в ответ похожее по структуре сообщение (или несколько сообщений), в котором содержится нужная информация, обычно это HTML документ.

Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты, потребители этих функций.

Существует два вида архитектуры взаимодействия клиент-сервер: первый получил название двухзвенная архитектура клиент-серверного взаимодействия, второй – многоуровневая архитектура клиент-сервер (иногда его называют трехуровневая архитектура или трехзвенная архитектура, но это частный случай).

Принцип работы двухуровневой архитектуры (Рисунок 3.1) взаимодействия клиент-сервер заключается в том, что обработка запроса происходит на одной машине без использования сторонних ресурсов. Двухзвенная архитектура предъявляет жесткие требования к производительности сервера, но в тоже время является очень надежной.

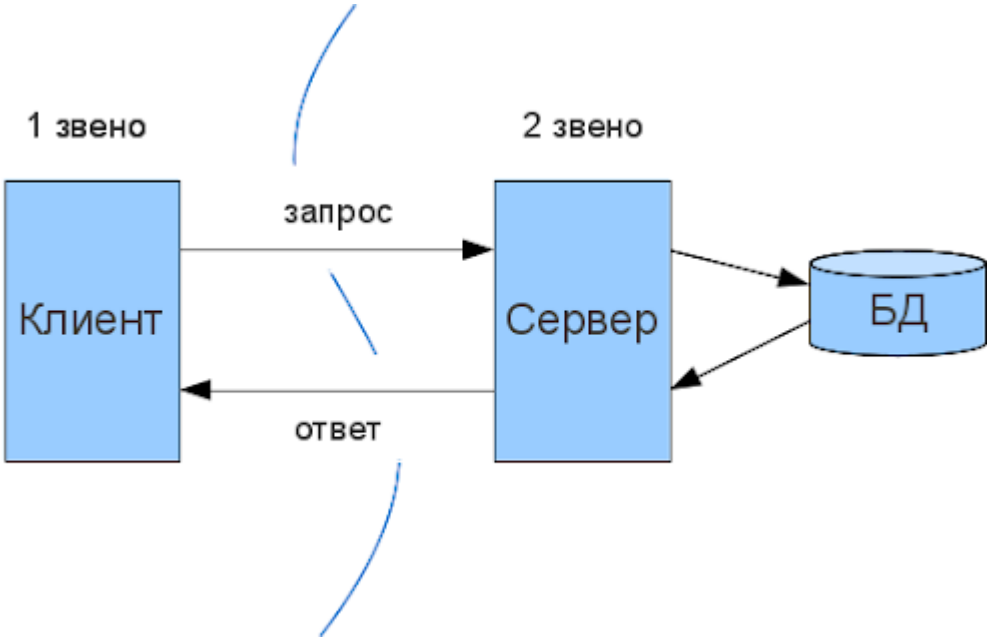

Рисунок 3.1 – Двухуровневая архитектура клиент-сервер

На рисунке 3.1 видно, что есть клиент (1-ый уровень), который позволяет человеку сделать запрос, и есть сервер (2-ой уровень), который обрабатывает запрос клиента.

Многоуровневую архитектура взаимодействия клиент-сервер, имеет, например отдельный сервер СУБД (за исключением, наверное, библиотеки SQLite, которая в принципе не использует концепцию клиент-сервер). Суть многоуровневой архитектуры заключается в том, что запрос клиента обрабатывается сразу несколькими серверами. Такой подход позволяет значительно снизить нагрузку на сервер из-за того, что происходит

распределение операций, но в то же самое время данный подход не такой надежный, как двухзвенная архитектура. На рисунке 3.2 пример многоуровневой архитектуры клиент-сервер.

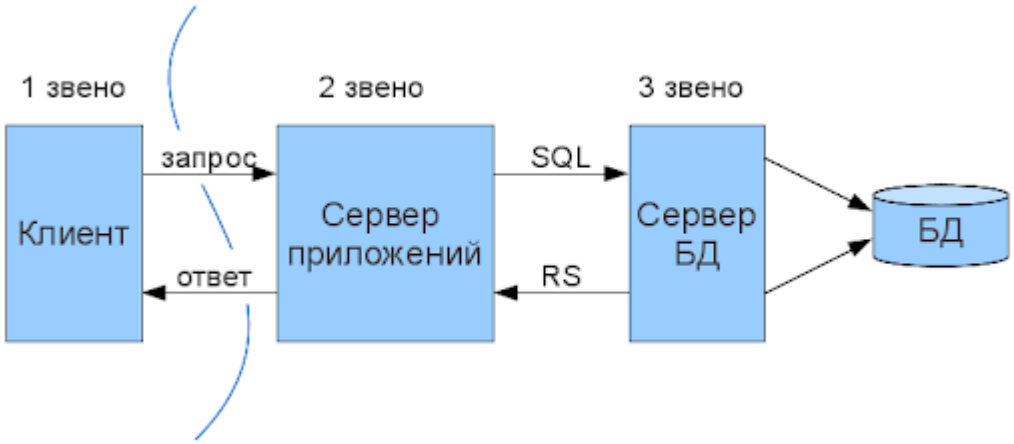

Рисунок 3.2 - Многоуровневая архитектура клиент-сервер

На рисунке 3.2 пример трехуровневой модели клиент-сервер. Если говорить в контексте систем управления базами данных, то первый уровень – это клиент, который позволяет нам писать различные SQL запросы к базе данных. Второй уровень – это движок СУБД, который интерпретирует запросы и реализует взаимодействие между клиентом и файловой системой, а третий уровень – это хранилище данных.

Если посмотреть на данную архитектуру с позиции сайта, то первый уровень можно считать браузером, с помощью которого посетитель заходит на сайт, второй уровень – это связка Apache + PHP, а третий уровень – это база данных.

Вывод:

Для реализации данного веб-приложения будет удобно использовать трехуровневую архитектуру клиент-сервер, так как в проектируемой системе присутствует сервер БД.

3.2 ВЫБОР АРХИТЕКТУРНОГО ШАБЛОНА ДЛЯ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ

Шаблон проектирования – архитектурная конструкция, предназначенная для проектирования некоторых часто возникающих контекстов. Шаблоны являются наиболее похожими на готовые библиотеки.

Подходящими шаблонами для веб-приложений являются MVC. В контексте разработки веб-приложения возможно использовать классификацию шаблонов проектирования Мартина Фаулера [16]. Согласно ей шаблоны можно разделить по классам:

1. Базовые шаблоны.
2. Шаблоны веб-представления.
3. Шаблоны архитектурных источников данных.
4. Шаблон объектно-реляционной логики.
5. Шаблоны объектно-реляционного структурирования.
6. Шаблоны логики сущности.
7. Шаблоны распределения данных.
8. Шаблоны локальной конкуренции.

Любой из этих классов включает в себя некоторый набор шаблонов, одним из которых и является MVC (Model – View – Control), а также производные:

1. MVP (Model – View – Presenter).
2. MVVM (Model –View – View – Model).
3. HMVC (Hierarchical MVC).
4. PAC (Presentation – Abstraction – Control).

MVC [17] (см. рисунок 3.3) позволяет реализовать бизнес-логику приложения без необходимости затрачивать значительные усилия на программирование. Он разделяет работу веб-приложения на три отдельные функциональные роли: модель данных (model), пользовательский интерфейс (view) и управляющую логику (controller). Таким образом, изменения, вносимые в один из компонентов, оказывают минимально возможное воздействие на другие компоненты. В данном паттерне модель не зависит от представления или управляющей логики, что делает возможным проектирование модели как независимого компонента и, например, создавать несколько представлений для одной модели.

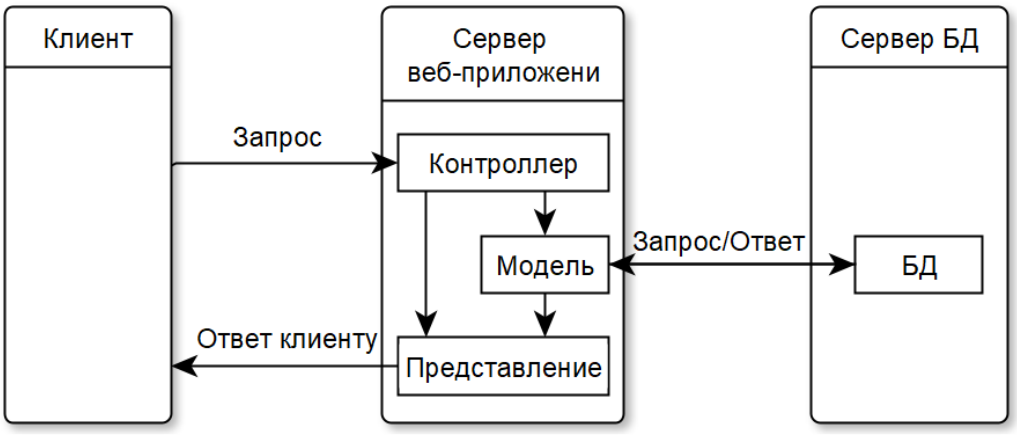

Рисунок 3.3 – Модель MVC

Паттерн PAC [18] (Presentation–Abstraction–Control) – программный архитектурный паттерн. Это ориентированная на взаимодействие программная архитектура, которая разделяет интерактивную систему на три типа компонентов, отвечающих за конкретные аспекты функциональности приложения. Компонент абстракции извлекает и обрабатывает данные, компонент представления форматирует визуальное и звуковое представление данных, а компонент управления обрабатывает такие вещи, как поток управления и связь между двумя другими компонентами. В отличие от MVC, PAC используется в качестве иерархической структуры агентов, каждый из которых состоит из триады частей представления, абстракции и контроля. Агенты (или триады) связываются друг с другом только через контрольную часть каждой триады. Он также отличается от MVC тем, что в каждой триаде он полностью изолирует представление (представление в MVC) и абстракцию (модель в MVC).

Для реализации приложения был выбран шаблон MVC, поскольку он обеспечивает простую реализацию бизнес-логики и является типичным решением при разработке веб-приложений. Поэтому рассмотрим подробно компоненты MVC:

1. **Модель** [21]

Она содержит в себе всю логику приложения, хранит и обрабатывает данные, при этом не взаимодействуя с пользователем напрямую. Например, сохранение информации в БД, проверка правильности введенных в форму данных – это задача Модели, получение этих данных от пользователя или вывод информации на экран или обработка нажатия на кнопку - нет.

Модель не должна никак зависеть и не должна ничего знать о Контроллерах и Видах. Модель – это не один класс или набор однотипных классов. Это основная часть приложения, которая может содержать много разных классов: сервисы, классы для взаимодействия с БД, сущности, валидаторы.

2. **Представление** [21]

Преставление – это код, который отвечает за отображение информации на экране, отрисовку кнопок и других элементов интерфейса. В веб-приложении оно обычно состоит из HTML-шаблонов страниц.

3. **Контроллер** [21]

Контроллер отвечает за выполнение запросов, пришедших от пользователя. В веб-приложении обычно контроллер обращается к модели, чтобы получить или изменить какие-то данные, и в конце вызывает Представление, чтобы отобразить результат выполнения запроса. Один Контроллер может работать с несколькими Моделями, и наоборот, одна Модель может использоваться в нескольких Контроллерах.

В веб-приложении Контроллеры – это набор однотипных классов, каждому разделу на сайте соответствует свой класс, и в нем делаются методы (их называют "действия", "action").

3.3 МОДЕЛЬ MVC В ВЕБ-ПРИЛОЖЕНИИ

В таблице 3. 1 представлены все контроллеры веб-приложения, а также их назначение и связь с представлениями и моделями.

Таблица 3.1 – Контроллеры веб-приложения

Наименование	**Назначение**	**Представления**	**Модели**
HomeController	Домашняя страница	_conctruct; Index.	
LoginController	Проверка автоирзации	_conctruct; Index.	
RegisterController	Работа с регистрацией	_construct; validator; create.	UserModel
CompanyController	Работа с данными о компании	Index; Create; Store; Show; Update; Destroy;	CompanyModel
DashboardController	Работа с главной страницей	Index; infoBox; create; store; show; update; destroy;	ProductModel; StockModel; AnalyticModel;
MenuController	Работа с меню навигации	Index; create; store; show; update; destroy;	MenuModel.
PremissionController	Настройка доступа	Index; store; show; edit; update; destroy;	PermissionModel .

Наименование	Назначение	Представления	Модели
ProductController	Работа с товарами	Index; Create; productList; productByCategory; store; show; edit; update; destroy;	ProductModel; StockModel; CategoryModel.
ReportingController	Проверка данных и поиск ошибок	Index; Store; Print;	CategoryModel; StockModel; UserModel; VendorModel; CompanyModel;
RoleController	Работа с пользовательским и ролями	Index; RoleList; Create; Store; Show; Permission; Edit; Destroy;	RoleModel; MenuModel; PermissionModel;
SettingController	Работа с настройками для админа	Index; Create; Store; Update;	UserModel.
StockController	Работа со складами	Index; stockList; ChalanList; Create;	StockModel.

Наименование	**Назначение**	**Представления**	**Модели**
		Store; Show; Edit; StockUpdate; Destroy;	
UserController	Работа с пользователями	Index; Logout.	UserModel
UserManageController	Работа с пользователями для админа	Index; userList; create; store; show; edit; update; destroy;	UserModel
VendorConroller	Работа с поставщиками	Index; Vendor; Store; Edit; Update; Destroy;	VendorModel

Таблица 3.2 – Модели веб-приложения

Наименование	**Назначение**
CategoryModel	Сортировка товара
CompanyModel	Компании
MenuModel	Меню
AnalyticModel	Аналитика и статистика

PermissionModel	Доступ
ProductModel	Товар
RoleModel	Пользовательские роли
StockModel	Склад
UserModel	Пользователи
VendorModel	Поставщики

В таблице 3.3 содержатся представления веб-приложеня и их назначение

Таблица 3.3 – представления веб-приложения

Представление	**Назначение**
_conctruct	Конструктор
index	Список
create	Создание
store	Запись в бд
edit	Редактирование
update	Обновление
destroy	Удаление
infoBox	Вывести статистику
productList	Список товаров
productByCategory	Отсортированый список

	товаров
RoleList	Список ролей
Permission	Доступ
stockList	Склады
ChalanList	Склады с товарами
StockUpdate	Обновление склада
Logout	Завершение сессии
userList	Список пользователей
Vendor	Поставщики

3.4 ОПИСАНИЕ ДАННЫХ

Схема базы данных приведена на рисунке 3.4.

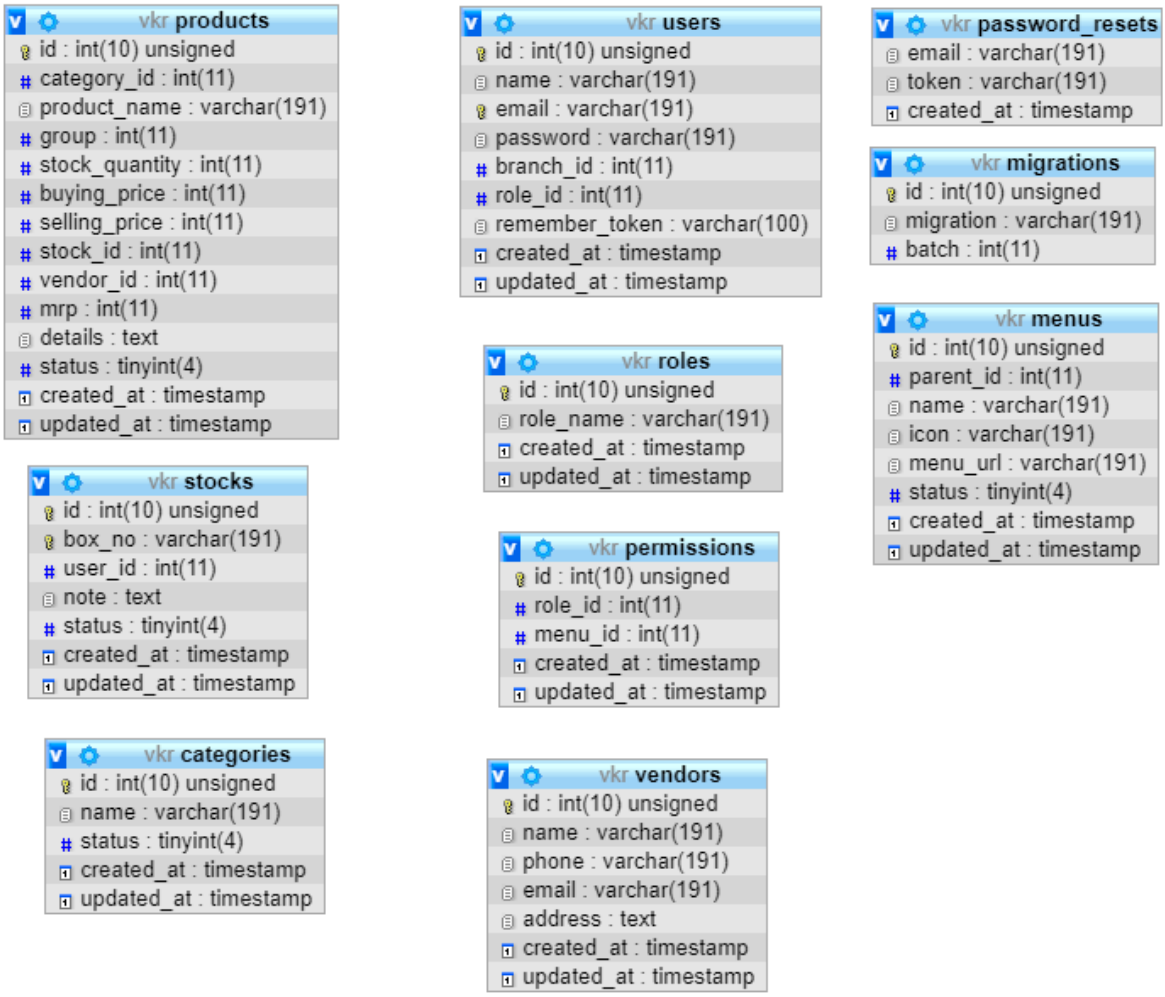

Рисунок 3.4 – Схема базы данных

4. РЕАЛИЗАЦИЯ

4.1. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСОВ

1. Форма авторизации пользователя

На рисунке 4.1 изображена форма входа для доступа к веб-приложению для зарегистрированных пользователей. В форме отслеживается необходимость заполнения полей, проверяется корректность ввода данных. Имеется возможность восстановить пароль.

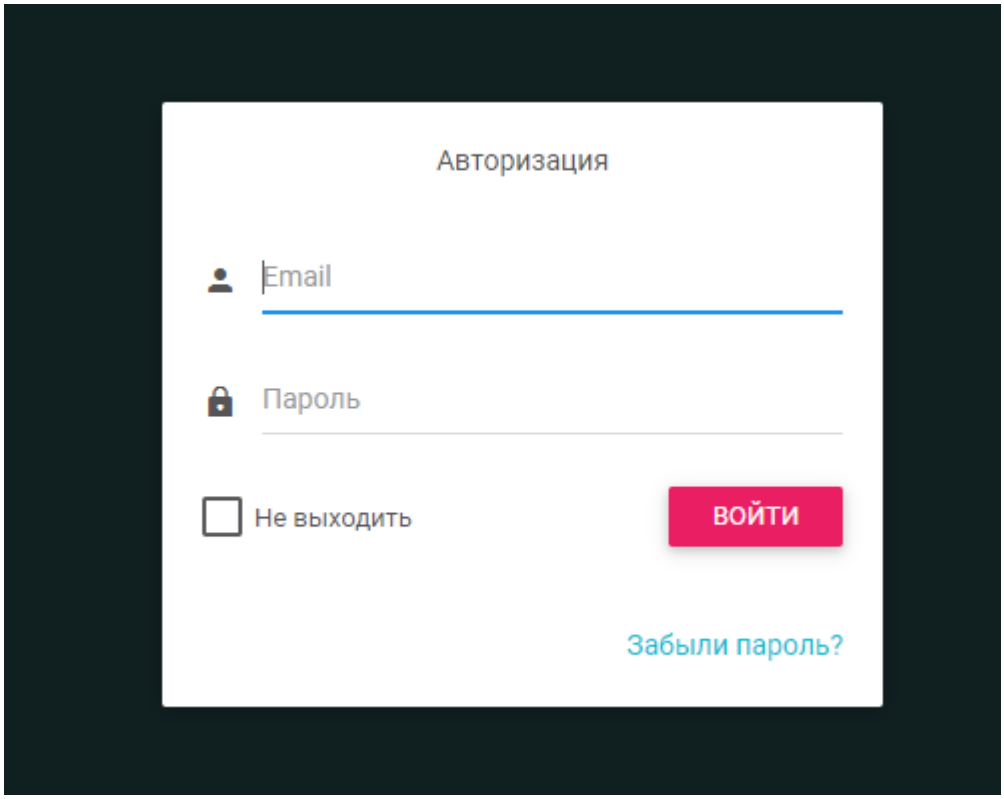

Рисунок 4.1 – Экранная форма авторизации

2. Домашняя страница

На рисунке 4.2 представлена домашняя страница и основной интерфейс приложения. Она состоит из трёх элементов: навигационной панели, верхней панели и рабочего блока. Первые два элемента присутствуют всегда. Верхняя

панель содержит логотип, и выпадающий список системных уведомлений, например об обновлении системы. Рабочий блок содержит информацию и инструменты в зависимости от выбранного пункта меню. На главной странице располагается статистическая информация.

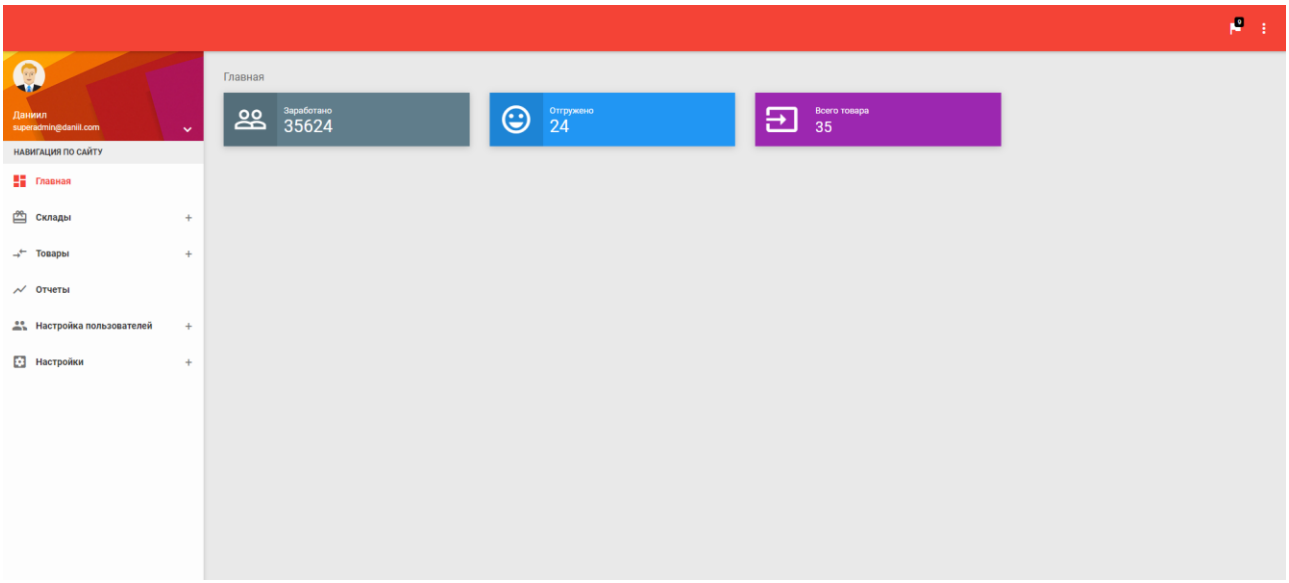

Рисунок 4.2 – Экранная форма домашней страницы приложения

4.3 Настройка ролей

На рисунке 4.3 представлено меню настроек ролей доступа. Данное меню доступно только администраторам. Здесь можно создать роль (рисунок 4.4), изменить уже доступы для уже имеющихся ролей и удалить роль. В настройках доступа также реализована система доступа модулей приложения. То есть, если необходимо отключить или подключить модуль достаточно изменить соответствующий пункт в настройках роли пользователя.

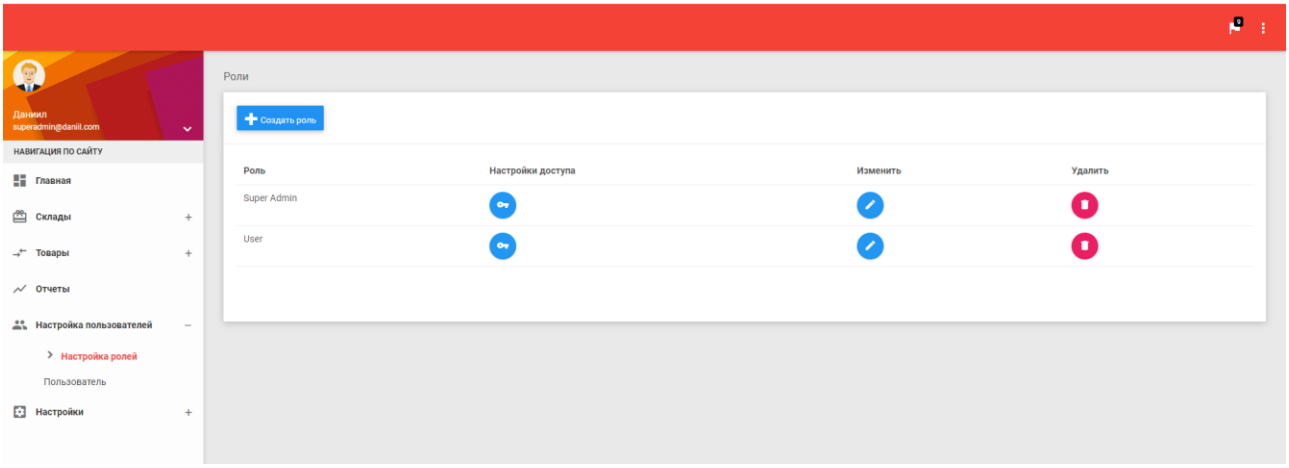

Рисунок 4.3 – Экранная форма настройки пользовательских ролей

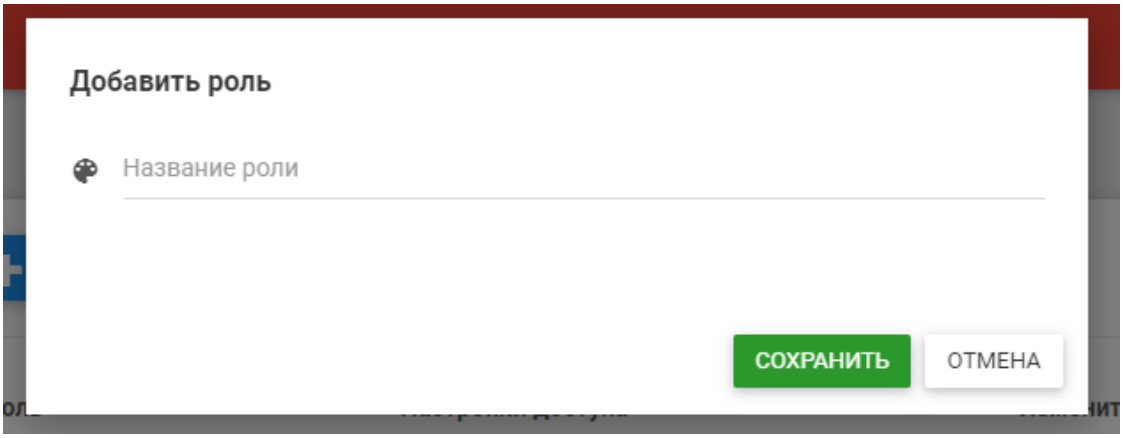

Рисунок 4.4 – Экранная форма добавления роли

На рисунке 4.5 представлена форма изменения доступа к модулям для обычных пользователей. Как видно из рисунка пользователь имеет доступ к работе со складом, товаром, документами, настройками аккаунта и не может получить доступ для настроек других пользователей, так как это администраторские инструменты. Пользователь также не имеет доступ к разделу аналитики, поскольку он был отключен администратором. На рисунке 4.6 представлен результат изменения доступа для пользователя «Сергей».

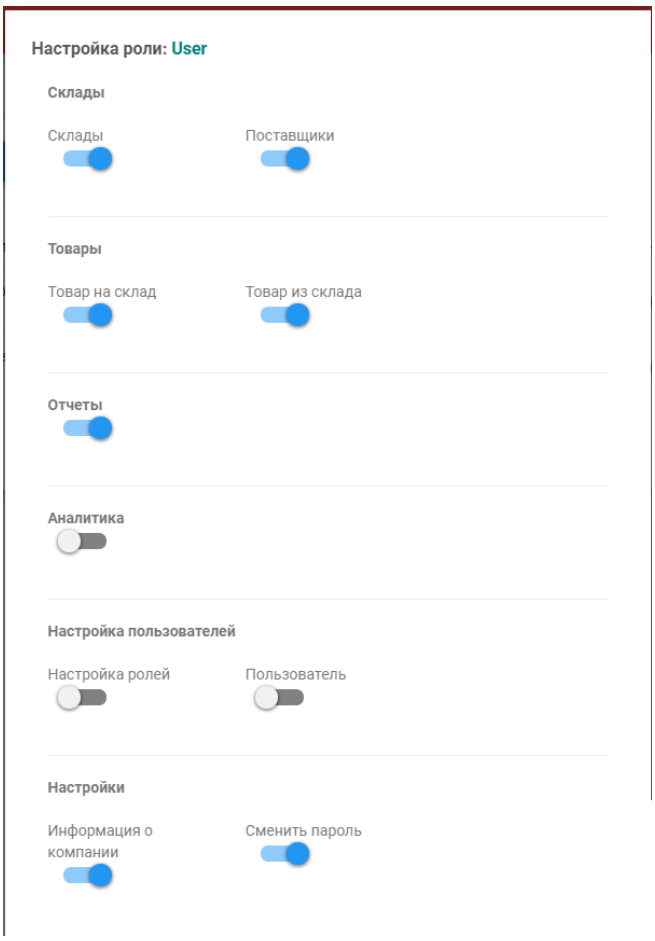

Рисунок 4.5 – Экранная форма пользовательской роли User

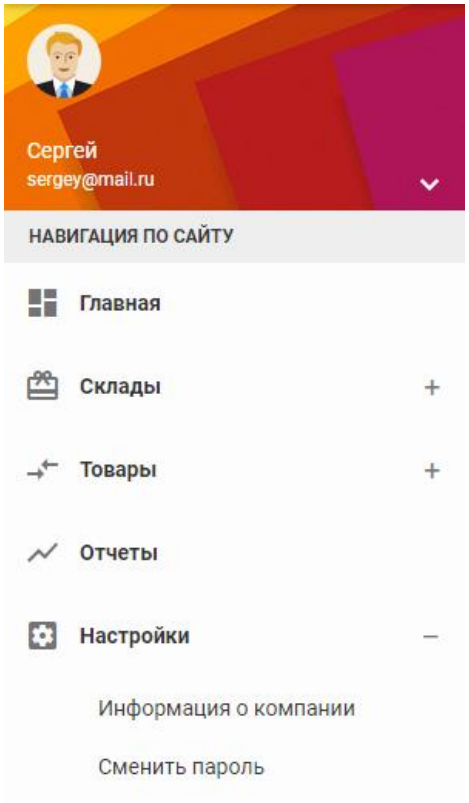

Рисунок 4.6 – Экранная форма навигационного меню пользователя после изменения настроек доступа

4.4 Смена пароля

На рисунке 4.7 представлена форма для смены пароля. Пароль пользователь может сменить в любой момент.

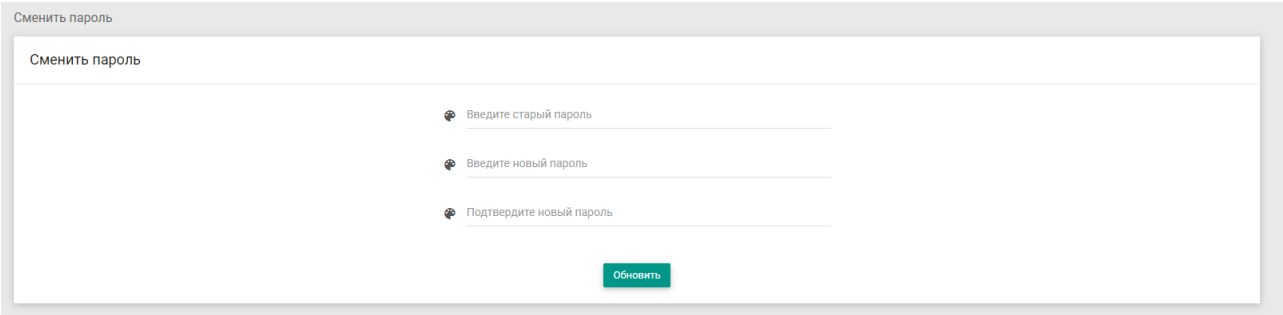

Рисунок 4.7 – Экранная форма смены пароля

4.5 Настройка пользовательских аккаунтов

Администратор имеет возможность просматривать имеющиеся пользовательские аккаунты, создавать новые и удалять имеющиеся.

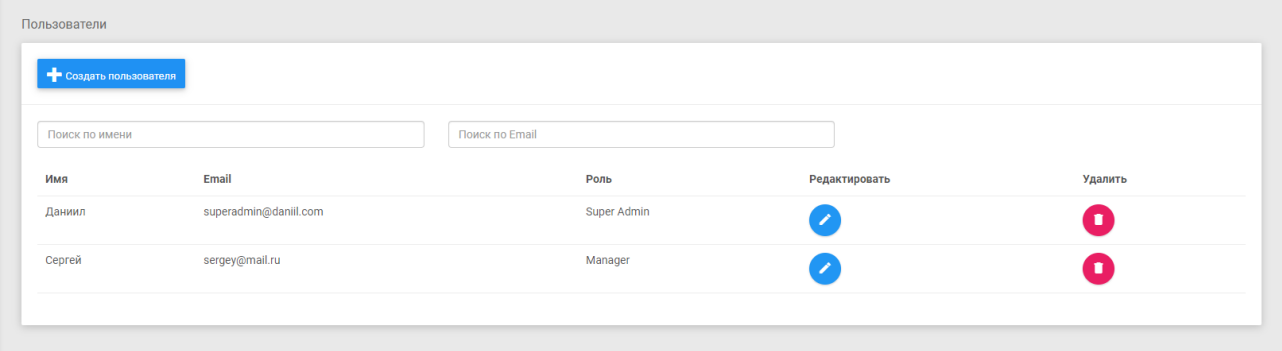

Рисунок 4.8 – Экранная форма настройки пользовательских аккаунтов

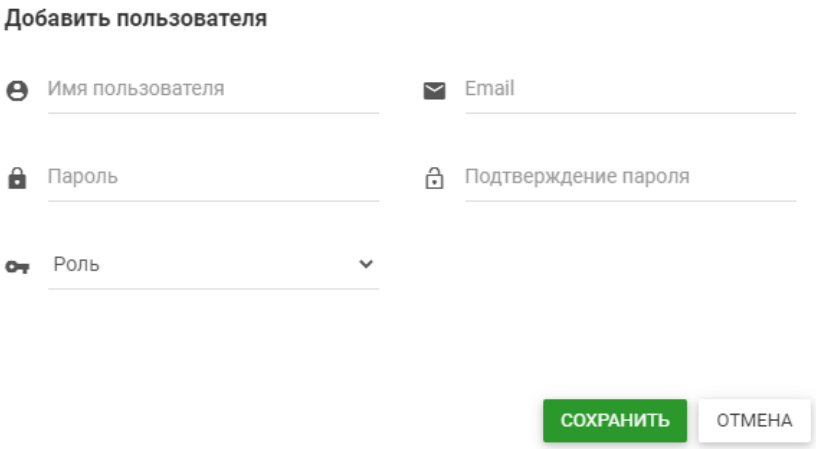

Рисунок 4.9 – Экранная форма для создания нового пользователя

4.6 Информация о компании

На рисунке 4.10 представлена форма для изменения юридической информации о компании. Это необходимо для автоматического заполнения в отчетах и документах.

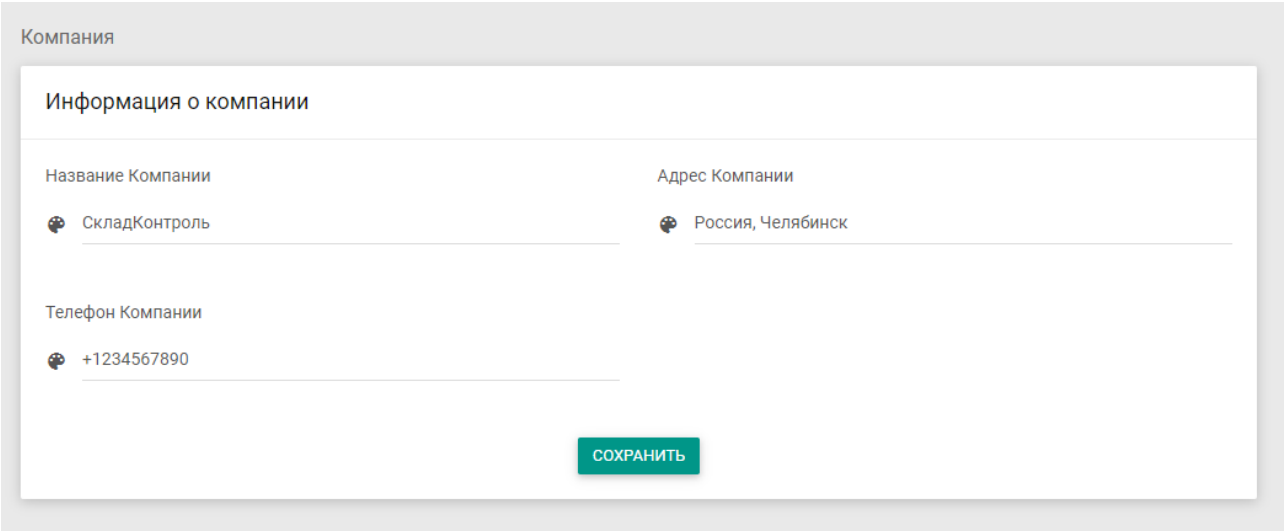

Рисунок 4.10 – Экранная форма информации о компании

4.7 Склады

На рисунке 4.11 представлен интерфейс меню настройки складов. Можно создать новый склад, изменить информацию о нём, и удалить ненужные склады. При удалении нужно учитывать что все товары привязанные к удаляемому складу также удалятся каскадно, о чем уведомляет всплывающее уведомление на рисунке 4.12

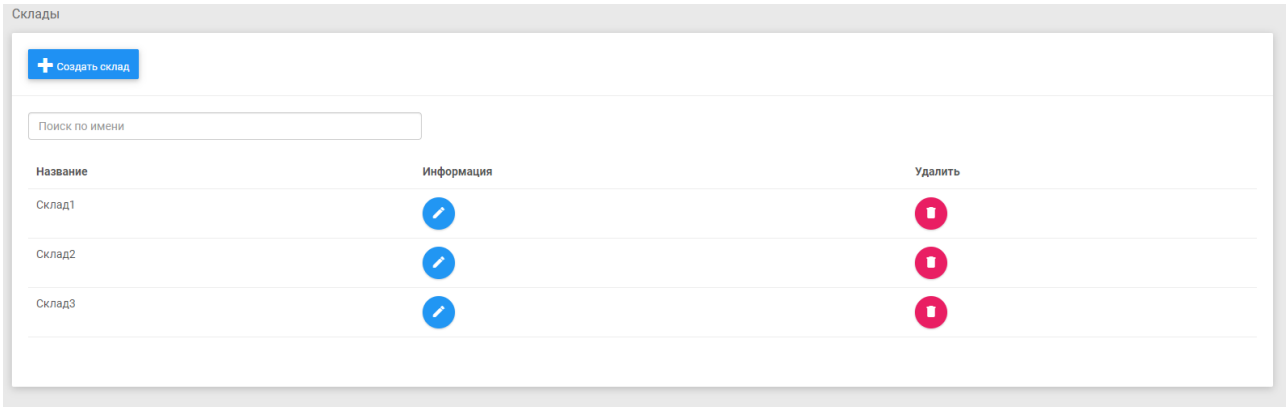

Рисунок 4.11 – Экранная форма управления складами

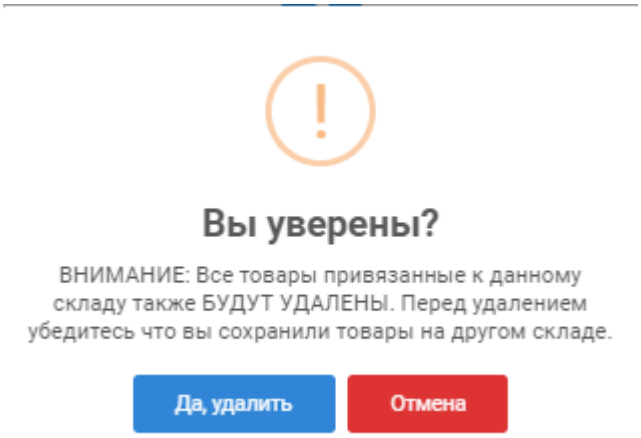

Рисунок 4.12 – Экранная форма подтверждения удаления склада

4.8 Товары

На рисунке 4.13 представлена таблица для работы с товарами. Товары можно добавлять, причем как по одному, так и по несколько сразу. Также можно изменить информацию о товаре, либо только количество.

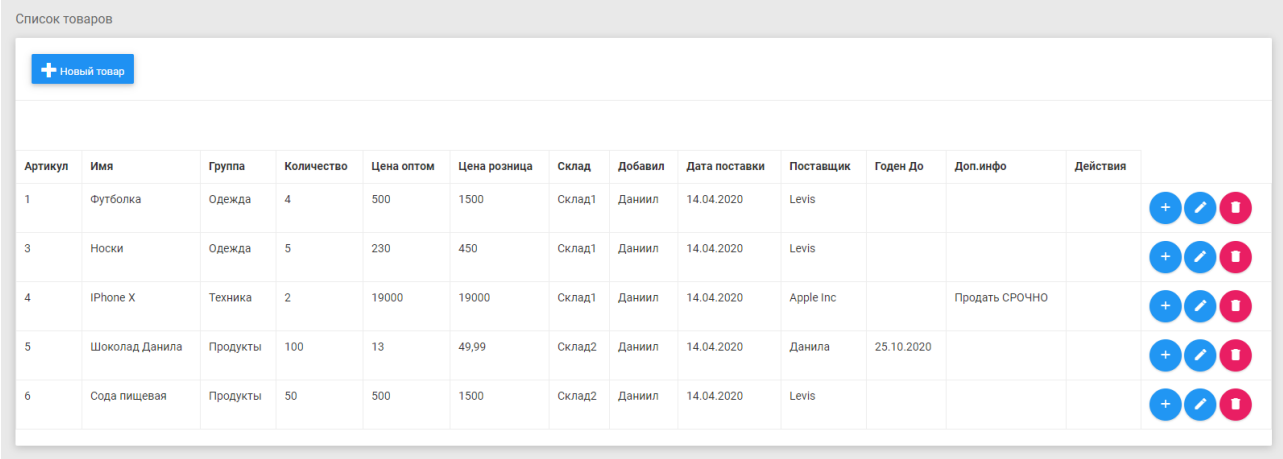

Рисунок 4.13 – Экранная форма списка товаров

4.9 Отчеты

На рисунке 4.14 представлена форма для получения отчета. Необходимо указать какой нужно получить отчет, указать дату или промежуток. Также можно получить отчет по конкретному товару, складу или поставщику.

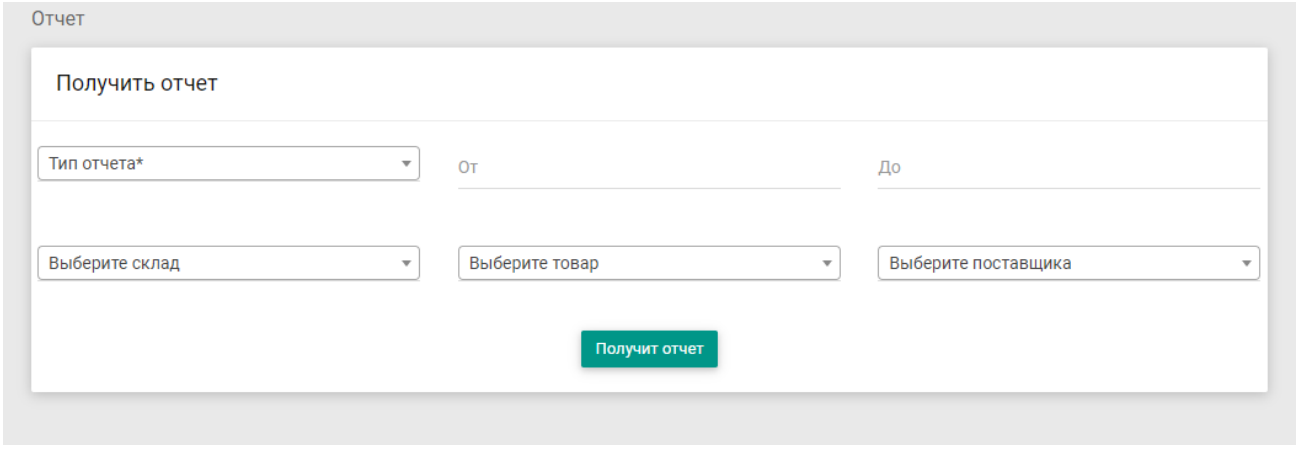

Рисунок 4.14 – Экранная форма получения отчетов

5. ЗАКЛЮЧЕНИЕ

В ходе дипломного проектирования выполнено следующее:

1. Проведен анализ предметной области и существующих решений;
2. Разработано техническое задание;
3. Выбрана среда и средства реализации, разработаны основные архитектурные решения;
5. Разработана база данных;
6. Реализована пользовательская и серверная части приложения;
7. Протестировано разработанное программное обеспечение.

В веб-приложении были реализованы следующие функции:

1. Регистрация и авторизация пользователей;
2. Создание складов;
3. Работа с товарами;
4. Работа с поставщиками;
5. Работа с отчетами и документами;
6. Управление модулями приложения и пользовательскими аккаунтами;
7. Аналитика.

В настоящее время веб-приложения ещё не готово к эксплуатации и находится в разработке, его необходимо доработать, в частности: доработать аналитику и статистику, добавить дополнительные отчеты.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 MS SQL Server. – https://navicongroup.ru/platforms/4025/. Дата обращения: 03.04.2020.

2 Критерии для выбора СУБД. – http://citforum.ru/database/articles/criteria/. Дата обращения: 03.04.2020.

3 Сравнение СУБД – http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/. Дата обращения: 03.04.2020.

4 Маркин Е.И. Разработка web-приложения с использованием архитектуры «клиент–сервер» / Е.И. Маркин, К.М. Рябов., Е.А. Артюшина // Международный студенческий научный вестник. – 2016. – № 3-1. – http://www.eduherald.ru/ru/article/view?id=14732. Дата обращения: 13.04.2020.

5 Описание фреймворков для веб-разработки – https://itproger.com/news/178. Дата обращения: 17.04.2020.

6 1С Склад – https://solutions.1c.ru/. Дата обращения: 10.03.2020.

7 Sap – https://www.sap.com/. Дата обращения: 10.03.2020.

8 Мой склад – https://www.moysklad.ru/. Дата обращения: 10.03.2020.

9 Лицензии MS SQL Server – https://www.microsoft.com/ru-ru/sql-server/sql-server-2017-pricing. Дата обращения: 15.04.2020.

10 .NET Core – http://tqm.com.ua/likbez/article/pochemu-net-ru. Дата обращения: 10.03.2020.

11 Razor Pages – https://docs.microsoft.com/ru-ru/aspnet/core/razor-pages/?view=aspnetcore-2.2&tabs=visual-studio. Дата обращения: 1.04.2020.

12 EF core – https://docs.microsoft.com/ru-ru/ef/core/managing-schemas/migrations/. Дата обращения: 31.03.2020.

13 Laravel Framework – https://laravel.com/. Дата обращения: 16.03.2020.

14 Архитектура клиент-сервер. – https://redcomrade.ru/materinskie-platy/izuchenie-setevogo-vzaimodeistviya-v-arhitekture-klient-server/. Дата обращения: 1.05.2020.

15 Классификация Мартина Фаулера – http://design-pattern.ru/patterns/. Дата обращения: 19.05.2020.

16 Патерн проектирования MVC – http://design-pattern.ru/patterns/mvc.html. Дата обращения: 19.05.2020.

17 Патерн проектирования PAC – http://ru.knowledgr.com. Дата обращения: 19.05.2020.

18 Патерн проектирования HMVC– https://ru.wikipedia.org/wiki/HMVC. Дата обращения: 19.05.2020.

19 Патерн проектирования MVP – https://ru.wikipedia.org/wiki/Model-View-Presenter. Дата обращения: 19.05.2020.

20 Патерн проектирования MVVM https://metanit.com/sharp/wpf/22.1.php. Дата обращения: 19.05.2020.

21 Взаимодействие компонентов MVC - https://github.com/codedokode/pasta/blob/master/arch/mvc.md. Дата обращения: 19.05.2020

22 VueJs Framework. https://ru.vuejs.org/index.html. Дата обращения: 22.03.2020.