**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«БАРАНОВИЧСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

|  |  |
| --- | --- |
| **Факультет** | **Инженерный** |
| **Кафедра** | **Информационных технологий и физико-математических** |
|  | **дисциплин** |
| **Секция** | **Информационных систем и технологий** |

|  |  |
| --- | --- |
| **Дата регистрации работы на кафедре** |  |
| **Отметка о допуске к защите** |  |
| **Оценка за защиту** |  |

**КУРСОВОЙ ПРОЕКТ**

|  |  |
| --- | --- |
| **По дисциплине** | **Визуальные средства разработки программных** |
|  | **приложений** |
| **Тема:** | **Разработка интернет-магазина по продаже кормов** |
|  | **для животных** |

|  |
| --- |
| **Исполнитель:** |
| **Студент 3 курса, группа ИСТ-31** |
| **студент (факультет, курс, группа)** |
| **Коваль А.Н.** |
| **фамилия, имя, отчество** |
|  |
|  |
| **Руководитель:** |
| **ст. преподаватель** |
| **ученое звание, ученая степень, должность** |
| **Кравчук О. Д.** |
| **фамилия, имя, отчество** |

**Барановичи 2024 год**

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**«БАРАНОВИЧСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

РЕЦЕНЗИЯ

на курсовой проект

(регистрационный №\_\_\_\_\_)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студента | | Коваль Александр Николаевич | | | |
|  | | (фамилия, имя, отчество) | | | |
| Факультет | | Инженерный | | | |
| Курс | | 3 | | | |
| Дисциплина | | Визуальные средства разработки программных приложений | | | |
| Рецензент | | Кравчук Ольга Дмитриевна | | | |
|  | | (фамилия, имя, отчество) | | | |
| Дата получения к/п для рецензирования | | | |  | |
| Дата возвращения к/п после рецензирования | | | |  | |
| Оценка |  | | Подпись преподавателя-рецензента | |  |
| Текст рецензии: | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |
|  | | | | | |

**РЕФЕРАТ**

Курсовой проект: 33 с., 37 рис., 6 источников, 12 прил., 2 табл.

REACT, NODEJS, NPM, MONGODB, ИНТЕРНЕТ-МАГАЗИН, JAVASCRIPT.

Объект исследования — среда разработки React, NodeJS.

Предмет исследования — применение технологии React для разработки сайтов под Windows.

Цель проекта — разработка интернет-магазина с администраторской панелью и хранением данных в базе данных.

Основой для выполнения проекта стала справочная, научная и учебная литература.

При выполнении проекта использовался метод: анализ научно-методической литературы, проектов.

Областью возможного практического применения данной работы является разработка интернет-магазинов с помощью фреймворков React, NodeJS.

Автор подтверждает, что приведенный в работе расчетно-аналитический материал правильно и объективно отражает состояние исследуемого процесса, а все заимствованные из литературных и других источников теоретические, методологические и методические положения и концепции сопровождаются ссылками на их авторов.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc166661492)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc166661493)

[1.1 Постановка задачи 5](#_Toc166661494)

[1.2 Описание предметной области 5](#_Toc166661495)

[1.3 Обоснование выбора программных средств разработки программного приложения 6](#_Toc166661496)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 8](#_Toc166661497)

[2.1 Построение и описание моделей базы данных 8](#_Toc166661498)

[2.2 Построение UML-диаграмм проекта 11](#_Toc166661499)

[2.3 Описание разработанных триггеров и хранимых процедур 15](#_Toc166661500)

[2.4 Описание методов защиты информации в приложении 23](#_Toc166661501)

[2.4 Руководство пользователя программным продуктом 25](#_Toc166661502)

[ЗАКЛЮЧЕНИЕ 31](#_Toc166661503)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 32](#_Toc166661504)

[ПРИЛОЖЕНИЯ 33](#_Toc166661505)

# ВВЕДЕНИЕ

В современном цифровом мире интернет-магазины стали неотъемлемой частью глобальной экономики. Рост популярности электронной коммерции изменил привычки потребителей, предоставив им удобство и доступ к широкому ассортименту товаров и услуг, не выходя из дома. Это преобразование привело к значительному увеличению спроса на высококачественные и функциональные интернет-магазины, которые способны удовлетворить потребности как пользователей, так и продавцов.

Интернет-магазины предлагают многочисленные преимущества как для бизнеса, так и для потребителей. Для бизнеса это возможность расширить охват рынка, сократить издержки на аренду и обслуживание физических магазинов, а также получить подробную аналитику поведения покупателей для более эффективного маркетинга [1]. Для потребителей интернет-магазины предоставляют удобство круглосуточного доступа, широкий выбор продуктов, возможность сравнения цен и чтения отзывов, что делает процесс покупки более информированным и комфортным.

Разработка интернет-магазина требует использования различных технологий и инструментов. Среди них можно выделить системы управления контентом (например, Magento, Shopify, WooCommerce), фреймворки для создания пользовательского интерфейса (React, Angular, Vue.js), серверные технологии (Node.js, Django, Ruby on Rails) и базы данных (MySQL, PostgreSQL, MongoDB).

Основные задачи проекта включают разработку функционального и красивого интерфейса, интеграцию надежной системы управления инвентарем и логистикой, а также внедрение механизмов персонализации предложений и рекомендаций для клиентов. Кроме того, особое внимание будет уделено вопросам безопасности и конфиденциальности пользовательских данных, а также оптимизации работы сайта для поисковых систем.

Процесс разработки интернет-магазина включает несколько ключевых этапов, каждый из которых играет важную роль в создании успешного проекта:

* Планирование и анализ.
* Проектирование и дизайн.
* Разработка и тестирование.

Одной из ключевых целей разработки интернет-магазина является создание удобной и эффективной платформы для продажи кормов для животных. С ростом числа домашних питомцев и увеличением осведомленности владельцев о важности качественного питания, спрос на специализированные корма и аксессуары для животных значительно возрос. Интернет-магазин, ориентированный на продажу кормов для животных, предоставляет возможность покупателям легко найти и приобрести нужные продукты для своих питомцев, не выходя из дома. Это особенно важно для занятых владельцев домашних животных, которые ценят свое время и удобство онлайн-покупок.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## Постановка задачи

Разработка сайта, предоставляющий возможность заказа товаров, который должен иметь следующий функционал: просмотр товаров, добавление товаров в корзину, редактирование товаров, оформление корзины.

Сайт должен иметь администраторскую панель, в которой должен быть реализован следующий функционал: добавление новых товаров, редактирование товаров, удаление товаров, просмотр активных заказов.

Также необходимо, чтобы на сайте была возможность проходить регистрацию и авторизацию, с использование хеширования пароля.

Нужно реализовать следующие запросы: вывести список всех товаров, вывести данные для одного товара, получить данные об пользователе, получить список товаров из корзины, добавление нового пользователя, добавить новый товар, получить список заказов, вывести заказ с максимальным чеком.

Реализовать следующие триггеры:

* Отслеживать количество товаров у продавца, при добавлении их в корзину.
* При работе с персональными данными, добавление и редактировании товаров, оформлении заказа, проверять авторизацию пользователя.
* Реализовать триггер, который при выполнении заказа, создаст отчёт в виде документа

## Описание предметной области

Предметная область, в рамках которой разрабатывается информационная система, представляет собой рынок зоотоваров, с особым акцентом на продажу кормов для животных. Этот рынок включает в себя разнообразные категории товаров, такие как корма для собак, кошек, птиц, рыб и других домашних питомцев. Основными участниками рынка являются производители кормов, дистрибьюторы, розничные продавцы и конечные потребители — владельцы домашних животных.

Для добавления новых товаров необходимы следующие данные: название товара, описание товара, цена товара, фото. Для добавления пользователей необходимы полное имя, номер телефона, адрес доставки.

## 1.3 Обоснование выбора программных средств разработки программного приложения

При разработке интернет-магазина для продажи кормов для животных был сделан выбор в пользу использования следующих программных средств: React для фронтенда, Node.js для бэкенда и MongoDB для базы данных. Этот выбор обусловлен рядом факторов, которые обеспечивают высокую производительность, масштабируемость, удобство разработки и соответствие современным стандартам веб-разработки [2].

React был выбран в качестве фронтенд-библиотеки по следующим причинам:

* Компонентный подход: React позволяет разрабатывать интерфейс с помощью многократно используемых компонентов, что способствует модульности и повторному использованию кода [3].
* Одностороннее связывание данных: Этот подход упрощает отладку и прогнозирование поведения приложения, так как данные передаются в одном направлении.
* Сообщество и экосистема: React имеет большое сообщество разработчиков, множество готовых библиотек и инструментов, которые облегчают и ускоряют процесс разработки [4].

Node.js был выбран в качестве серверной платформы по следующим причинам:

* Асинхронная и событийная модель: Node.js позволяет обрабатывать большое количество запросов одновременно без блокировки выполнения, что особенно важно для высоконагруженных приложений.
* Единый язык разработки: Использование JavaScript как на фронтенде, так и на бэкенде упрощает процесс разработки и обучения, а также позволяет повторно использовать код и библиотеки.
* Быстрота и масштабируемость: Node.js построен на движке V8 от Google, что обеспечивает высокую производительность и быстродействие [5].
* Богатая экосистема: npm, менеджер пакетов для Node.js, предоставляет доступ к огромному количеству библиотек и модулей, которые могут быть использованы для решения различных задач.

MongoDB был выбран в качестве базы данных по следующим причинам:

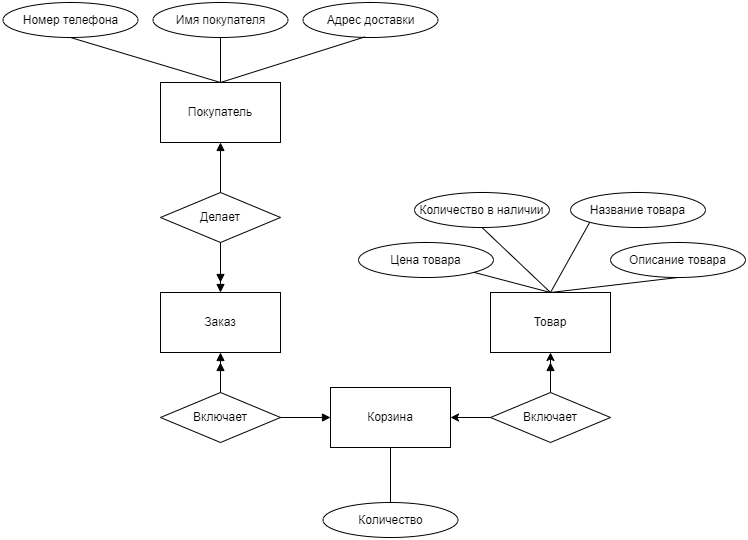
* Гибкость в работе с данными: MongoDB является документно-ориентированной базой данных, что позволяет хранить данные в формате JSON-подобных документов. Это облегчает работу с данными, особенно для приложений, требующих гибкой схемы.
* Масштабируемость: MongoDB легко масштабируется как вертикально, так и горизонтально, что позволяет справляться с увеличением нагрузки и объема данных.
* Простота интеграции с Node.js: MongoDB легко интегрируется с Node.js через библиотеку Mongoose или нативный драйвер, что упрощает работу с базой данных и ускоряет разработку [6].

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Построение и описание моделей базы данных

Концептуальное моделирование — это процесс создания абстрактного представления структуры данных для информационной системы. Оно позволяет понять и формализовать требования к данным, определяя основные сущности, их свойства и отношения между ними. Основная цель концептуального моделирования — разработка модели, которая описывает, что будет представлено в базе данных, без учета физических аспектов её реализации.

Целью концептуального моделирования является создание четкого и понятного представления о данных, их связях и правилах, которые будут использоваться для управления данными. Это помогает разработчикам, аналитикам и заинтересованным сторонам согласовать требования и ожидания относительно структуры данных и обеспечить правильность их интерпретации. Концептуальная модель базы данных представлена на рисунке 2.1.



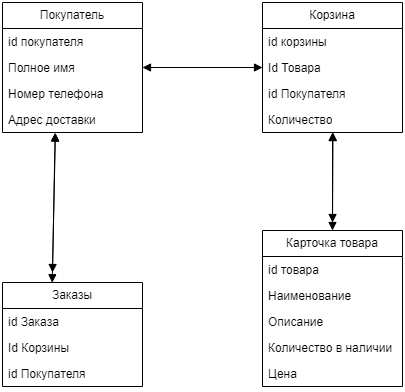
**Рисунок 2.1 — Концептуальная модель базы данных**

Модель включает следующие основные сущности: покупатель, заказ, корзина, товар.

Связи:

* Покупатель имеет связь с Заказом: один покупатель может сделать несколько заказов.
* Заказ связан с Корзиной: каждый заказ включает одну корзину.
* Корзина имеет связь с Товаром: корзина может содержать несколько товаров, и каждый товар в корзине имеет указание количества.

Логическая модель базы данных (БД) является важным шагом в процессе проектирования базы данных. Она служит мостом между концептуальной моделью, которая определяет, что должно храниться в базе данных с точки зрения бизнеса, и физической моделью, которая описывает, как данные будут храниться на конкретной системе управления базами данных (СУБД). Логическая модель базы данных представлена на рисунке 2.2.

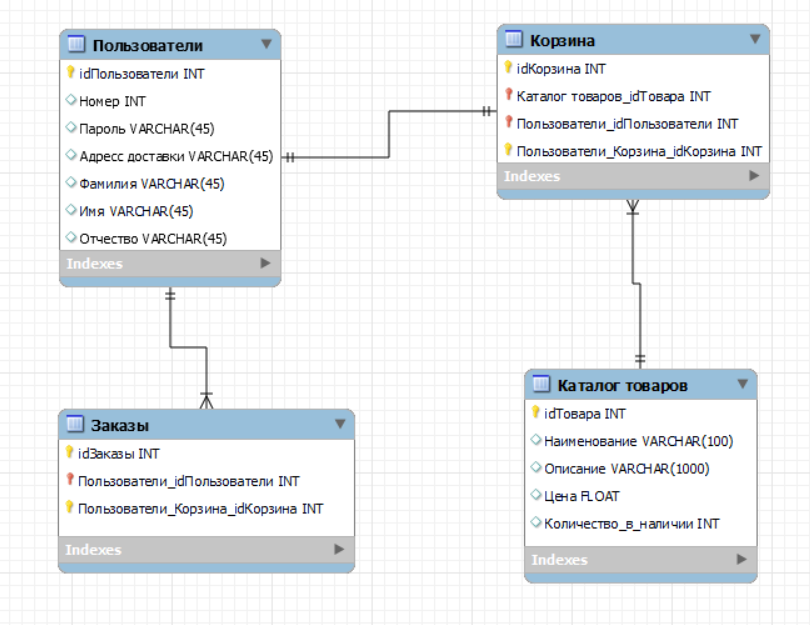


**Рисунок 2.2 — Логическая модель базы данных**

На представленной диаграмме изображена логическая модель базы данных, включающая пять основных таблиц: "Покупатель", "Заказы", "Корзина", "Карточка товара" и связи между ними. Эта модель предназначена для управления заказами в интернет-магазине.

* Покупатель делает один или несколько Заказов.
* Каждый Заказ включает одну Корзину.
* Корзина содержит один или несколько товаров, представленных в таблице Карточка товара.

Физическая модель базы данных описывает, как данные будут реально храниться, организованы и управляться в системе управления базами данных (СУБД). Она включает в себя детализированное представление структуры базы данных, определяющее физические реализации таблиц, индексов, ограничений, и других объектов базы данных. Физическая модель помогает проектировщикам и администраторам баз данных понимать, как эффективно хранить и получать доступ к данным с учетом производительности, надежности и масштабируемости. Физическая модель базы данных представлена на рисунке 2.3.



**Рисунок 2.3 — Физическая модель базы данных**

На представленной диаграмме сущность-связь (ERD) для интернет-магазина по продаже кормов для животных показаны основные сущности: Пользователи, Корзина, Каталог товаров и Заказы. В таблице пользователи хранятся данные о пользователях. В таблице корзина хранится id пользователя и id товара, который он добавил. В таблице заказы хранится id пользователя, которому принадлежит заказ, и id корзина, для получения списка товаров.

Ниже приводится описание связей и кардинальности между этими сущностями.

Связь между сущностями:

Пользователи – Корзина. Связь один к одному (1:1). У каждого пользователя может быть только одна активная корзина. Каждая корзина принадлежит только одному пользователю.

Корзина - Каталог товаров. Связь многие ко многим (1:N). Одна корзина может содержать много товаров.

Пользователи – Заказы. Связь один ко многим (1:M). Один пользователь может сделать много заказов, но каждый заказ связан только с одним пользователем.

## 2.2 Построение UML-диаграмм проекта

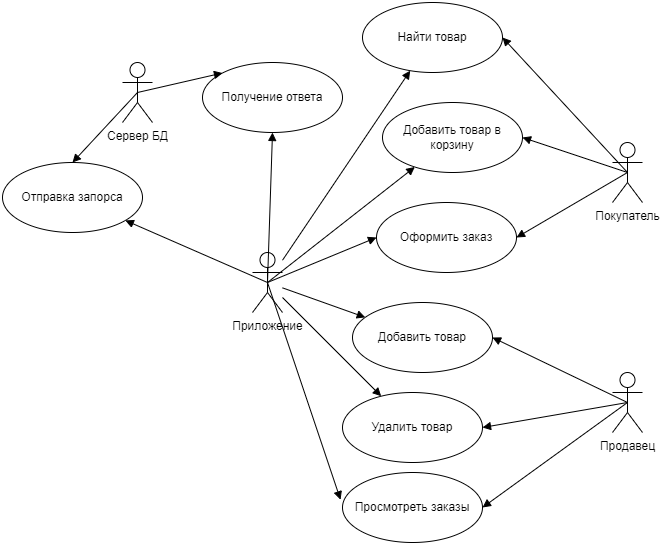
Диаграмма вариантов использования (use case diagram) представляет собой визуальное описание взаимодействий между пользователями (актерами) и системой для выполнения определённых функций (вариантов использования). Основные элементы и их функции включают.

Возможности продавца: принять товар, добавить товар в систему, удалить товар, просмотреть заказы.

Возможности покупателя: найти товар, добавить товар в корзину, оформить заказ, просмотреть заказы.

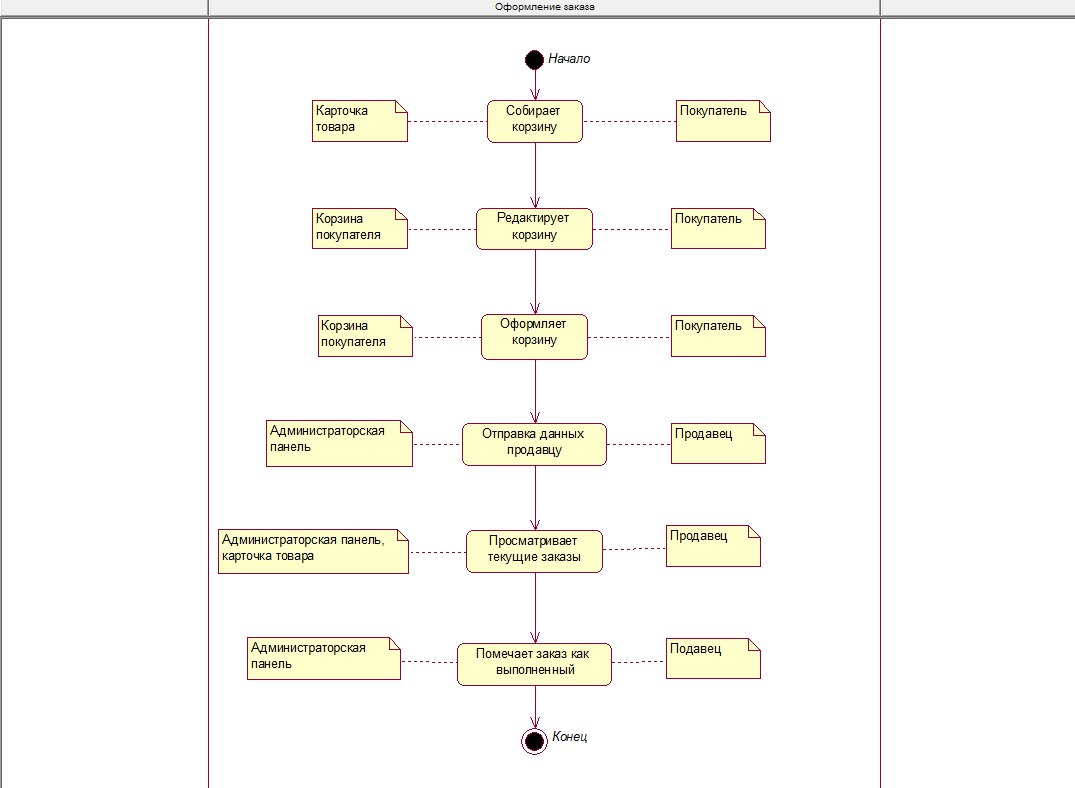
Приложение: обеспечивает функциональность для нахождения, принятия, добавления и удаления товаров, а также для добавления товаров в корзину, оформления заказов и просмотра заказов.

На рисунке 2.4 представлена use case диаграмма разрабатываемой информационной системы.



**Рисунок 2.4 — Диаграмма вариантов использования**

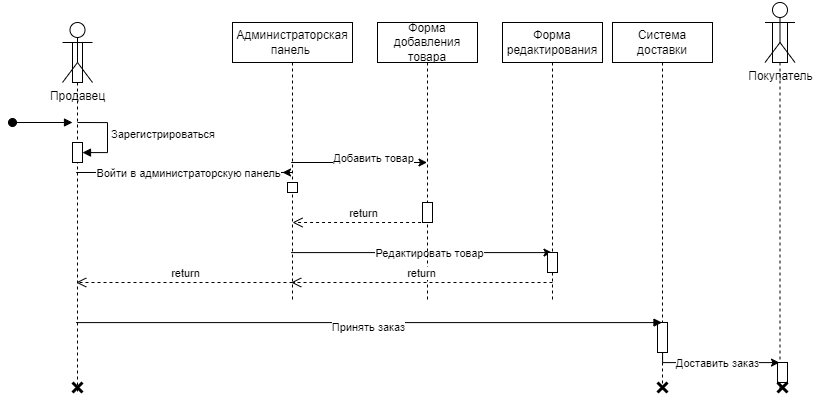
Диаграмма деятельности — это один из типов диаграмм UML, используемый для моделирования динамического поведения системы. Она помогает визуализировать последовательность действий и их потоки управления в процессе выполнения определенного процесса или функции. На рисунке 2.5 представлена диаграмма деятельности разрабатываемой информационной системы.



**Рисунок 2.5 — Диаграмма деятельности**

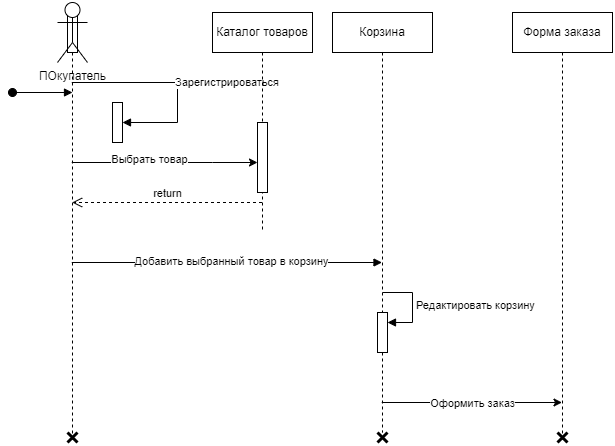
Эта диаграмма демонстрирует весь процесс оформления заказа от начальной стадии добавления товаров в корзину покупателем до завершения заказа продавцом. Она отражает взаимодействие между покупателем и продавцом, а также использование различных компонентов системы, таких как карточка товара, корзина покупателя и администраторская панель.

Диаграмма последовательности — это один из типов диаграмм UML (Unified Modeling Language), используемый для моделирования взаимодействий между объектами в определенной последовательности времени. На рисунке 2.6 представлена диаграмма последовательности для продавца. На рисунке 2.7 представлена диагрмма последовательности для попкупателя.



**Рисунок 2.6 — Диаграмма последоавтельности для продавца**

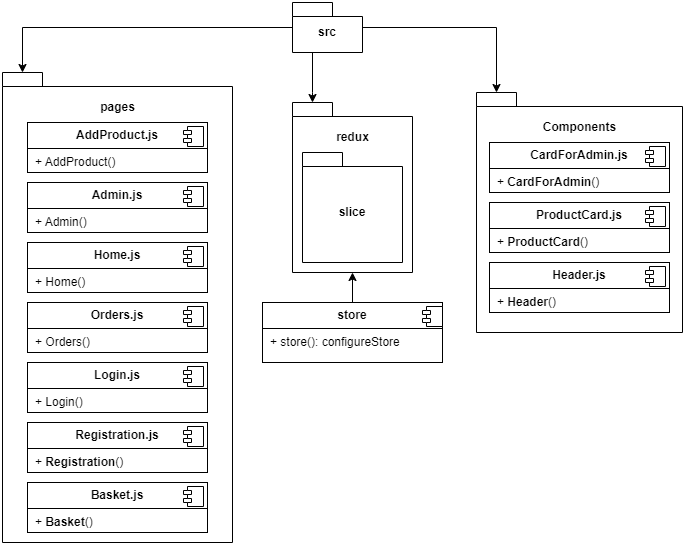
Описание взаимодействий: регистрация продавца, вход в администраторскую панель, продавец открывает форму добавления товара через администраторскую панель, продавец редактирует товар через форму редактирования. продавец принимает заказ через администраторскую панель, администраторская панель отправляет данные в систему доставки.



**Рисунок 2.7 — Диаграмма последоавтельности для покупателя**

Описание взаимодействий: регистрация покупателя, покупатель выбирает товар в каталоге товаров, после добавляет выбранный товар в корзину, по необходимости редактирует корзину, оформляет заказ.

Диграмма компонентов описывает общую структуру проекта. Пример показан на рисунке 2.8. Пакет pages хранит функции для отображения всех страниц сайта, пакет redux хранит в себе метод store который хранит в себе все состояния сайта, пакет components хранит в себе функции компонентов, которые часто используются для отображения информации

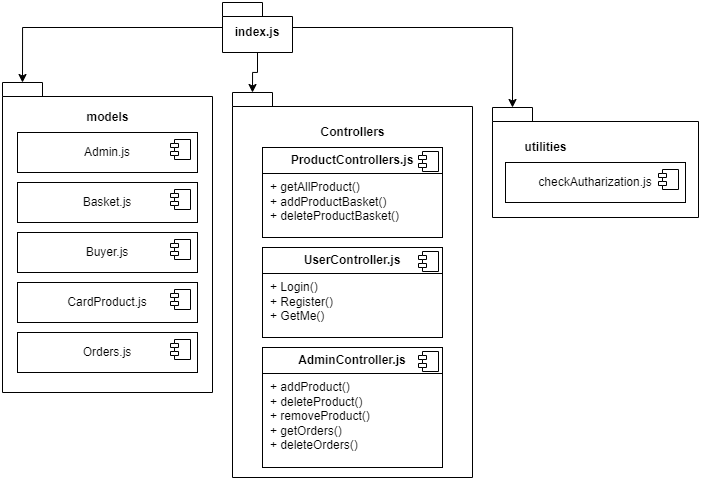


**Рисунок 2.8 — Диаграмма компонетов**

Таблица 2.1 – Описание диаграммы компонентов

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| AddProduct() | Реализует добавление товаров на стороне администратора |
| Admin() | Реализует функционал для администратора |
| Home() | Главная страница, на которой отображен каталог товаров |
| Orders() | Показывает список активных заказов продавцу |
| Login() | Реализует авторизацию пользователей |
| Registrtion() | Реализует регистрацию пользователей |
| Basket() | Функционал корзины товара |
| CardForAdmin() | Компонент, обрисовывающий карточку товара для администратора (включает функционал удаление и редактирования) |
| ProductCard() | Компонент, обрисовывающий карточку товара для покупателя (функционал: добавление товара в корзину) |
| Header() | Компонент, обрисовывающий шапку сайта |

Диаграмма компонентов представленая на риснуке 2.9, описывает общую структуру функционала, реализованого на стороне сервера. Сервер работает с файла index.js. Пакет models реализует создание таблиц для базы данных NoSQL. Пакет Controllers хранит модули, которые реализуют обработку запросов на сервер. Пакет utilities хранит в себе функцию для проверки авторизации пользователя.



**Рисунок 2.9 — Диаграмма компонетов**

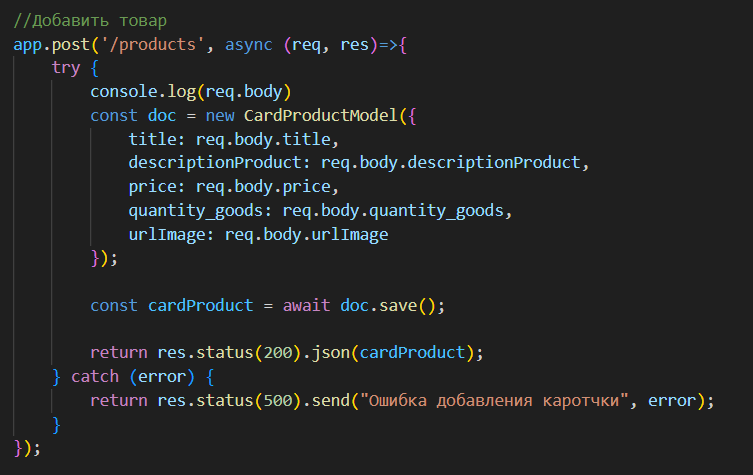
Таблица 2.2 – Описание функционла на стороне сервера

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| getAllProduct() | Возвращает все товары из базы данных |
| addProductBasket() | Реализует добавление товаров в корзину и в базу данных |
| deleteProductBasket() | Удаляет товар из корзины и базы данных |
| Login() | Реализует авторизацию пользователей |
| Registration() | Реализует регистрацию пользователей |
| GetMe() | Возвращает токен авторизации |
| addProduct() | Реализует добавление новых товаров в каталог |
| deleteProduct() | Реализует удаление товаров из каталога |
| removeProduct() | Реализует изменение товаров |
| getOrders() | Возвращает все активные заказы |
| deleteOrders() | Удаляет выполненный заказ |

## 2.3 Описание разработанных триггеров и хранимых процедур

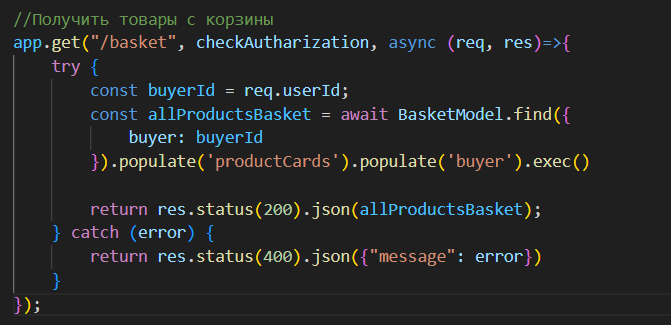
Работа с базами данных на сайте реализовано через запросы на сервер, который обрабатывает данные и заносит их в базу данных.

Пример добавления товара в каталог товаров. Представлен на рисунке 2.10.



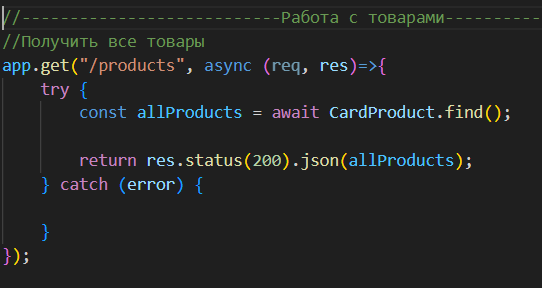
**Рисунок 2.10 — Запрос, выполняющий добавление товара**

Запрос, который возвращает список товаров, которые находятся в корзине пользователя представлен на рисунке 2.11. Запрос осуществляется с помощью метода find, который принимает id покупателя и выводит советующие товары.



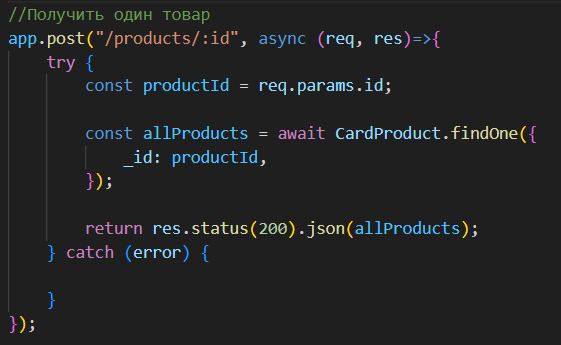
**Рисунок 2.11 — Запрос, возвращающий список товаров из корзины**

Запрос, реализующие вывод всех товаров из базы банных представлен на рисунке 2.12



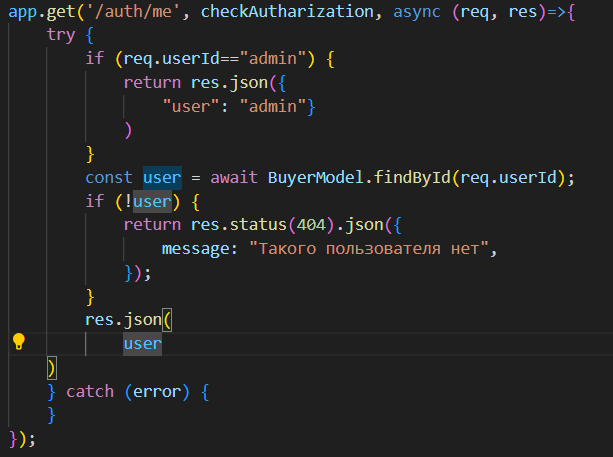
**Рисунок 2.12 — Запрос, возвращающий все товары из базы данных**

Запрос, реализующий вывод данных для одного товара по id представлен на рисунке 2.13.



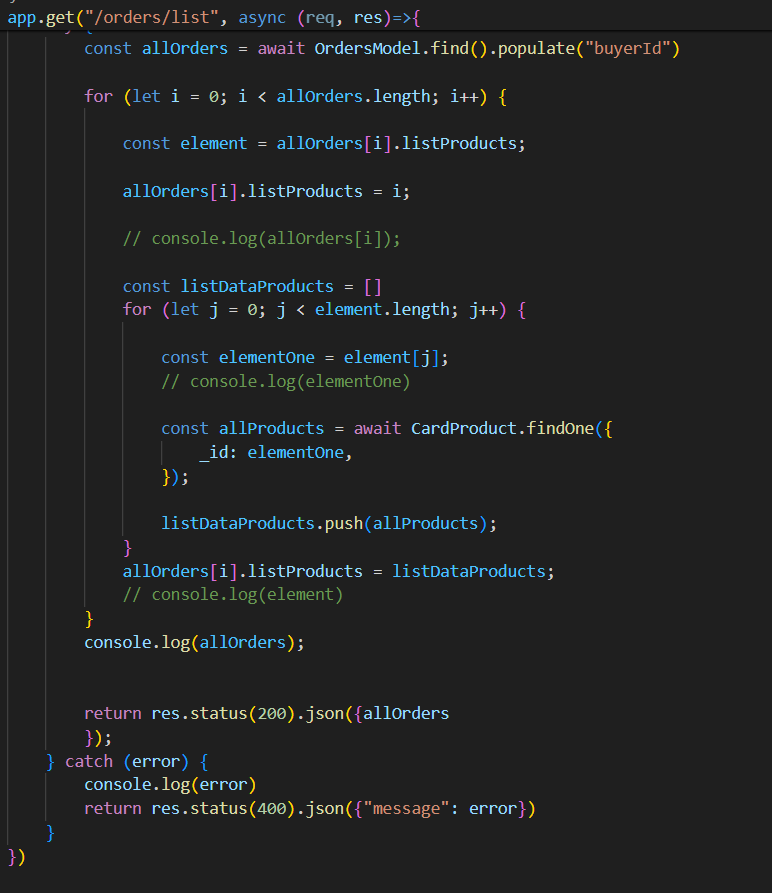
**Рисунок 2.13 — Запрос, реализующий вывод одного товара**

Запрос реализующий вывод информации об покупателе представлен на рисунке 2.14.



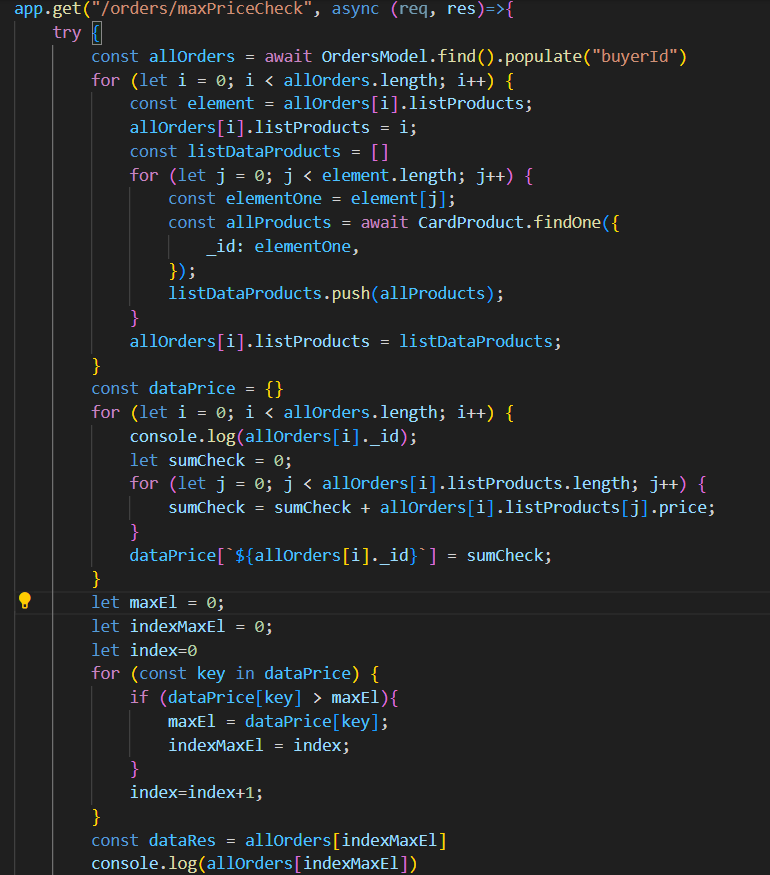
**Рисунок 2.14 — Получение информации об пользователе**

Запрос, реализующий вывод списка заказов представлен на рисунке 2.15



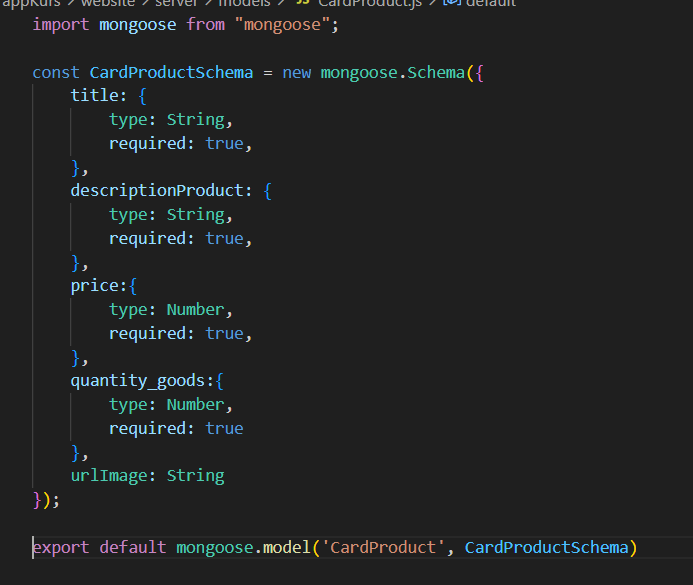
**Рисунок 2.15 — Вывод списка заказов**

Запрос, который выводит заказ с максимальным чеком представлен на рисунке 2.16.



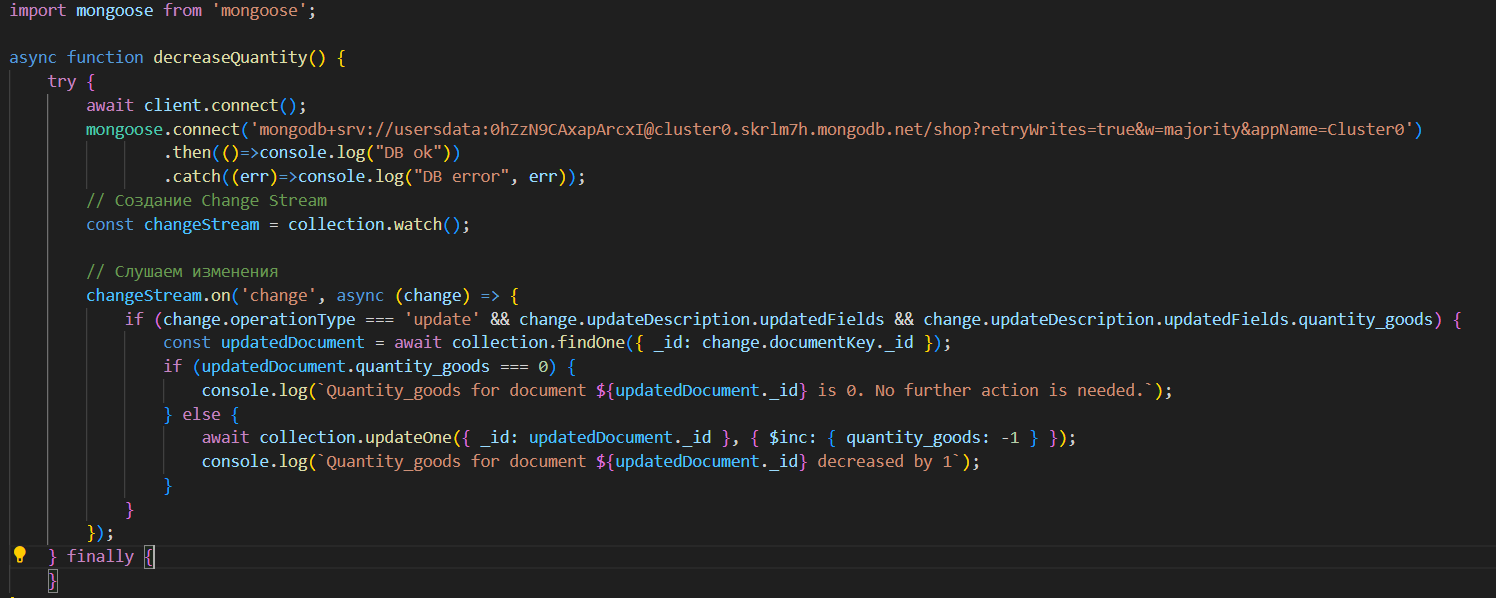
**Рисунок 2.16 — Вывод заказа с максимальным чеком**

Пример описания таблицы базы данных представлен на рисунке 2.17.



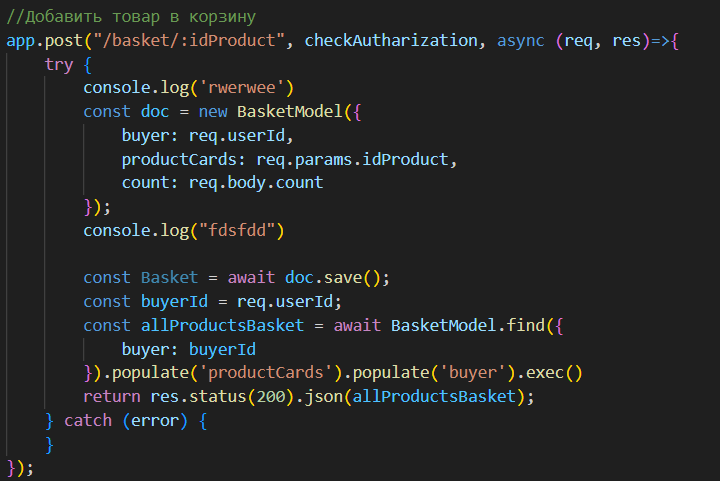
**Рисунок 2.17 — Описание таблицы товара**

В MongoDB триггеры реализуются с помощью механизма Change Streams. Change Streams позволяют вам слушать изменения в коллекциях и базах данных в реальном времени и выполнять определенные действия при наступлении этих изменений. Простой триггер можно реализовать с использованием Change Streams в MongoDB. Пример триггера, который запрещает добавление товара в корзину, если товаров больше нет у продавца, показан на рисунке 2.18.



**Рисунок 2.18 — Триггер, отслеживающий количество товаров у продавца**

Пример проверки пользователя, авторизован ли он, показан на примере добавлении товара в корзину (Рисунок 2.19).



**Рисунок 2.19 — Триггер, проверяющий авторизацию пользователя**

При попытке добавить товар в корзину, выполняется функция checkAutharization, которая проверяет токен авторизации пользователя. Реализация функции показана на рисунке 2.20.



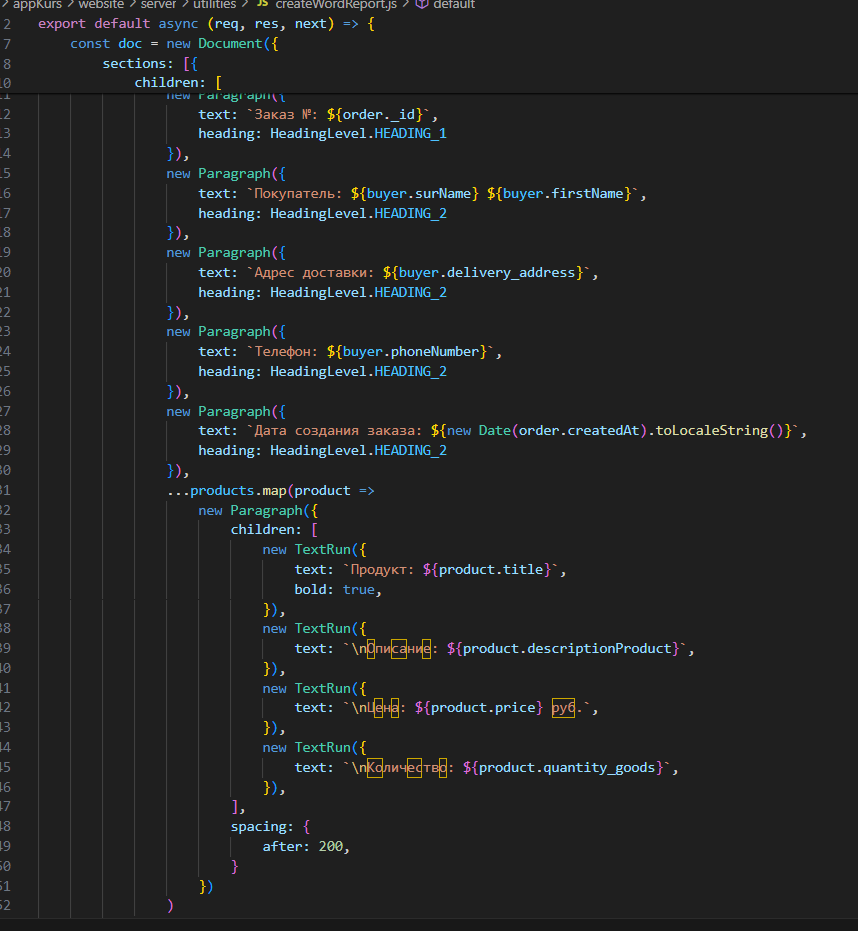
**Рисунок 2.20 — Функция проверяющая авторотационный токен**

При выполнении заказа, происходит скачивание отчёта. Пример показан на рисунке 2.21.



**Рисунок 2.21 — Код, реализующий скачивание отчёта**

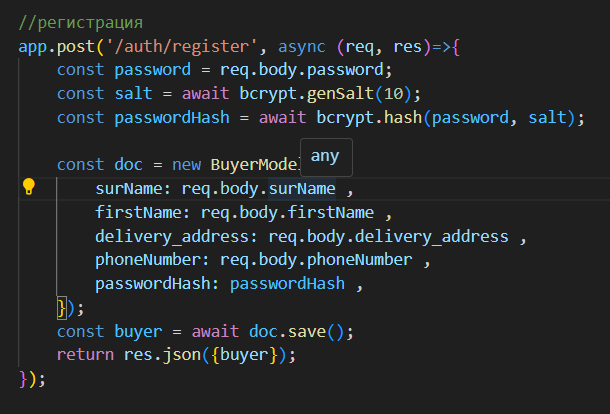
Формирование отчёта реализуется с помощью функции createWordReport. Код показан на рисунке 2.22.



**Рисунок 2.22 — Реализация создания отчёта**

## 2.4 Описание методов защиты информации в приложении

Реализовано несколько методов защиты данных. При регистрации пользователей введённый пароль шифруется. Код, реализующий шифрование показан на рисунке 2.23. Шифрование реализуется с помощью библиотеки bcrypt.

****

**Рисунок 2.23 — Пример регистрации покупателя**

При каждом запросе, где необходимо получить данные о конкретном пользователе, выполняется метод checkAutharization*.* Который проверяет пользователя с помощью токена авторизации. Код реализующий метод checkAutharizationпредставлен на рисунке 2.24.



**Рисунок 2.24 — Проверка токена авторизации**

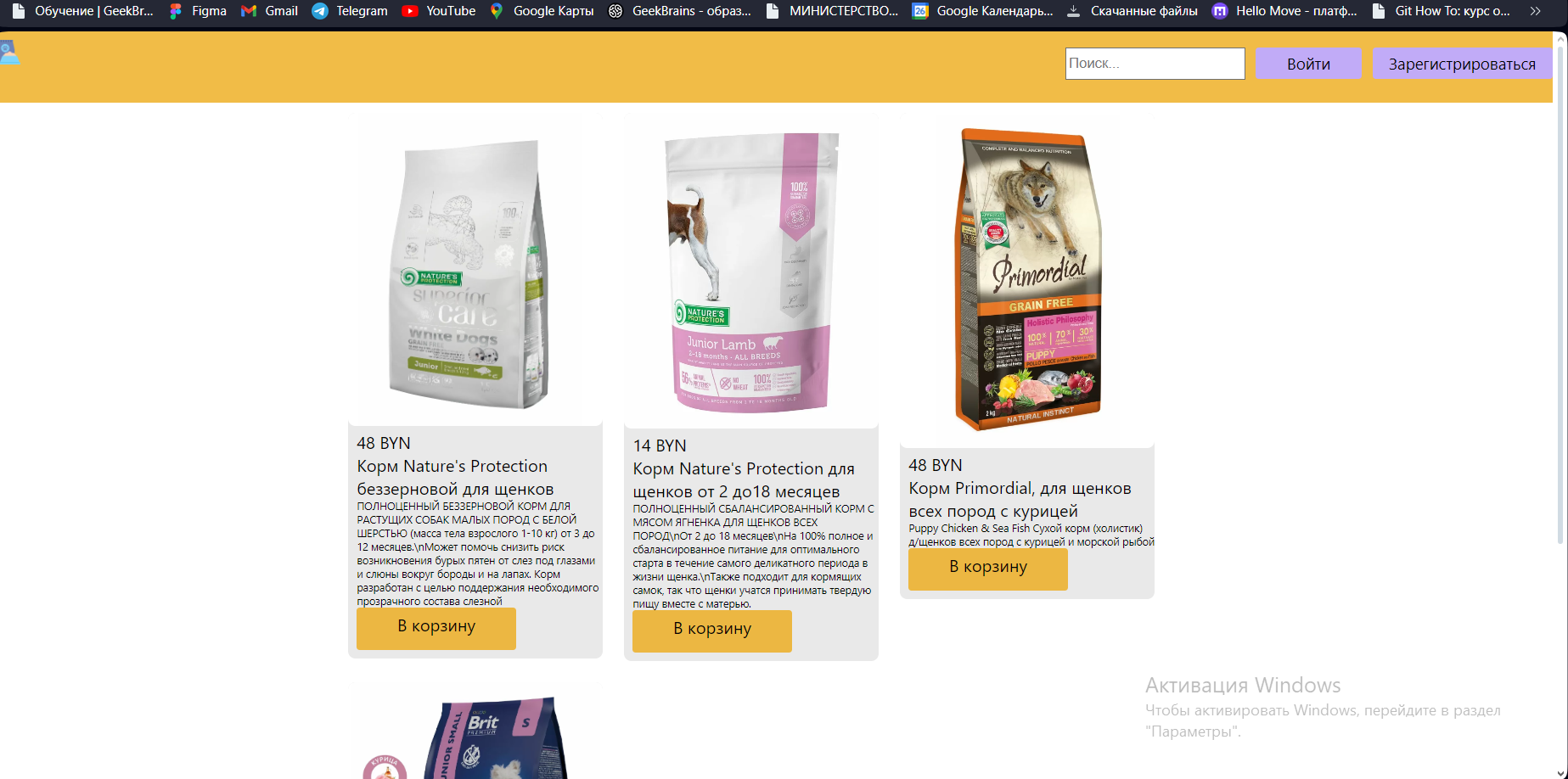
Сам же токен генерируется в момент авторизации пользователя. Код, реализующий генерацию токена показан на рисунке 2.25.



**Рисунок 2.25 — Генерация токена**

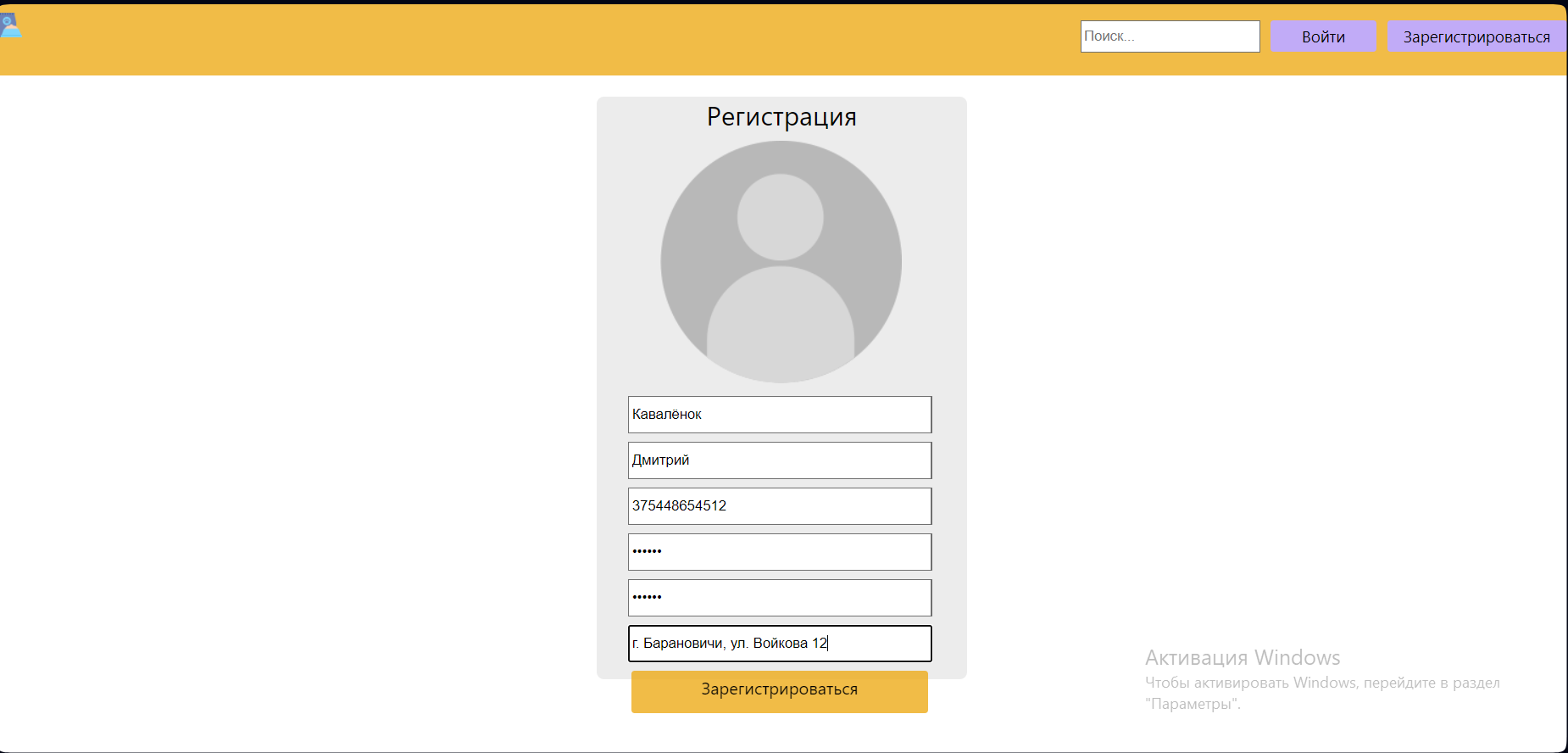
## 2.4 Руководство пользователя программным продуктом

При первом запуске на встречает главная страница, на которой находится все товары. Пример показан на рисунке 2.26.



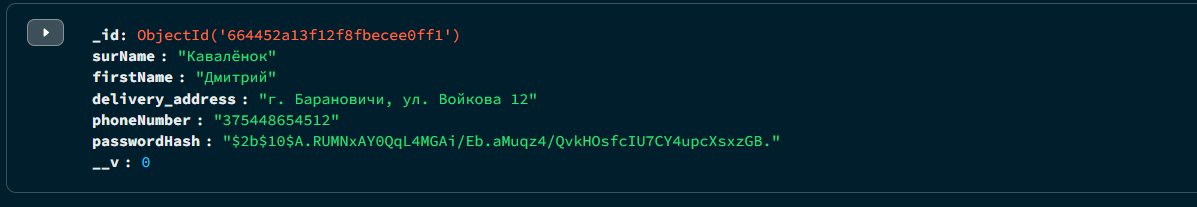
**Рисунок 2.26 — Главная страница**

Пока что есть возможность только просматривать товары, чтобы можно было их добавить в корзину и потом оформить нужно авторизоваться, если покупатель на сайте находится в первые, ему нужно зарегистрироваться. Нужно нажать на кнопку регистрация. После нажатия пользователь попадёт на форму регистрации. Заполняем все поля и нажимаем на кнопку регистрация. Пример показан на рисунке 2.27.



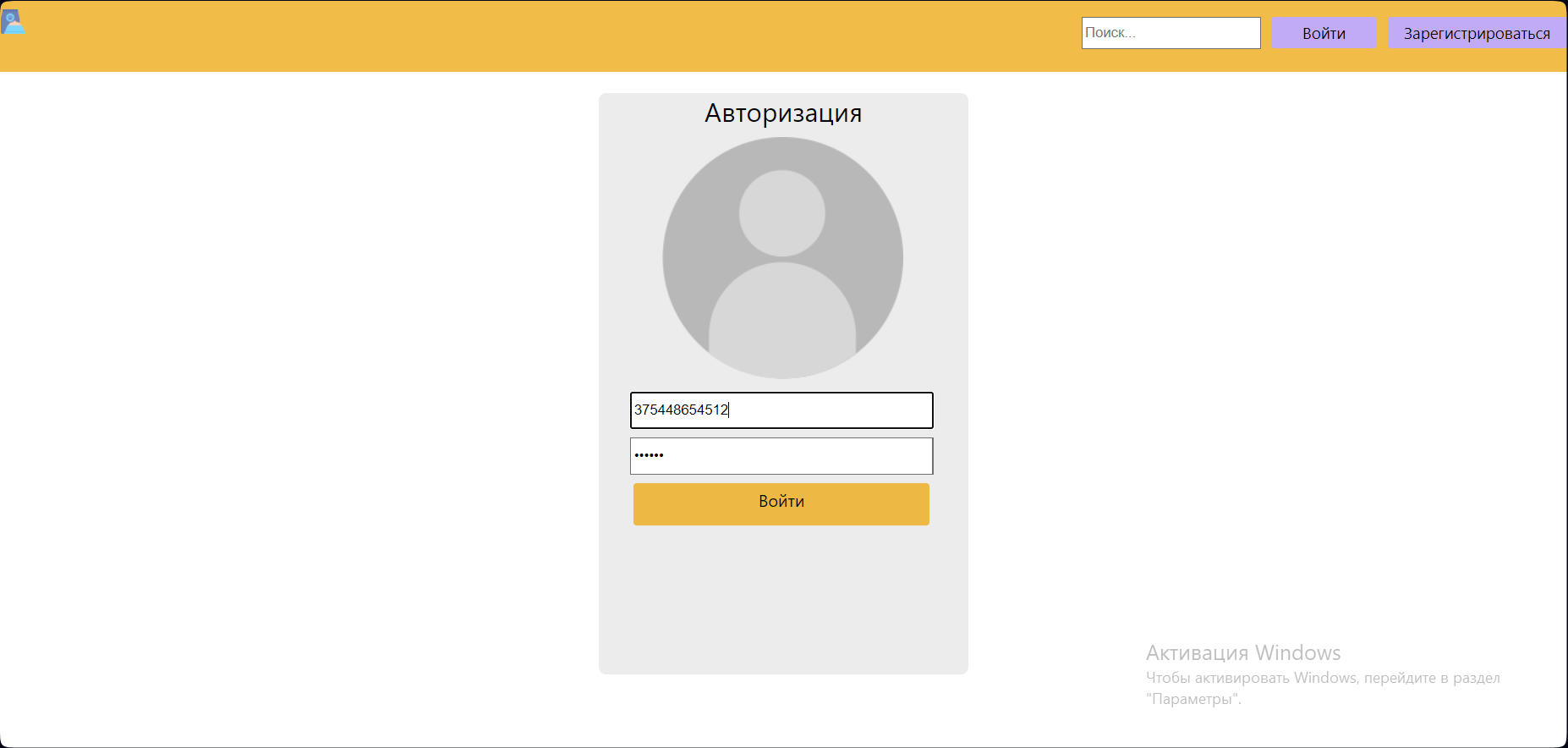
**Рисунок 2.27 — Регистрация**

После регистрации новый пользователь появится в базе данных (Рисунок 2.28).



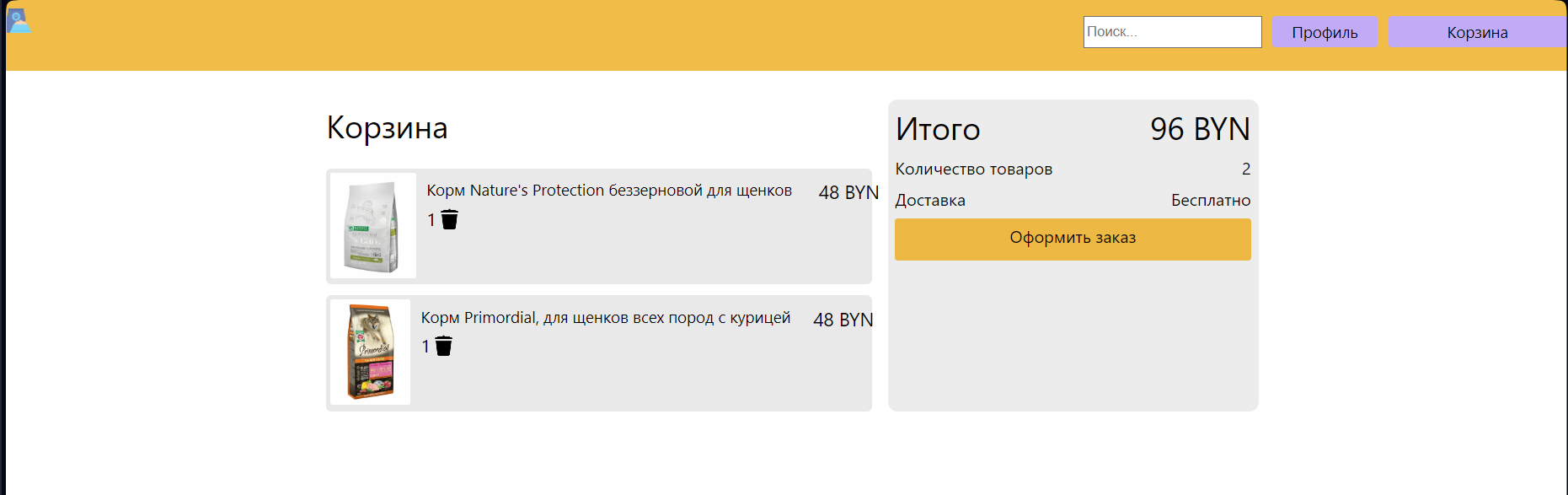
**Рисунок 2.28 — Пользователь в базе данных**

После регистрации новому пользователю нужно авторизоваться. Нажимаем на кнопку войти на главной странице. На форме авторизации вводим данные и нажимаем кнопку войти. Пример показан на рисунке 2.29.



**Рисунок 2.29 — Форма авторизации**

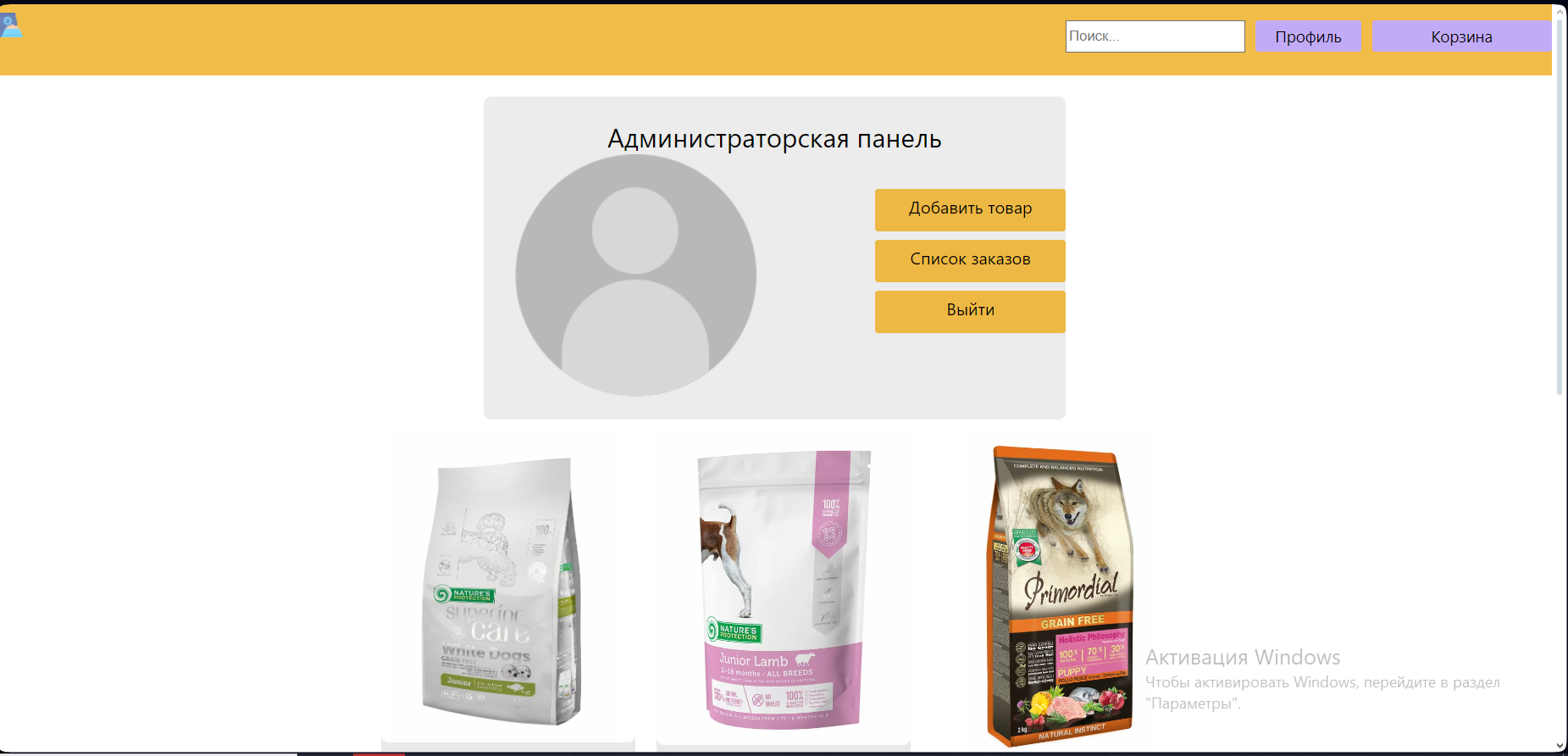
Теперь пользователь может добавлять товар в корзину. Чтобы добавить товар в корзину нужно нажать на кнопку «В корзину» на соответствующем товаре. После нажимаем кнопку Корзина, котором находятся все товары, которые мы добавили. Пример показан на рисунке 2.30.



**Рисунок 2.30 — Корзина**

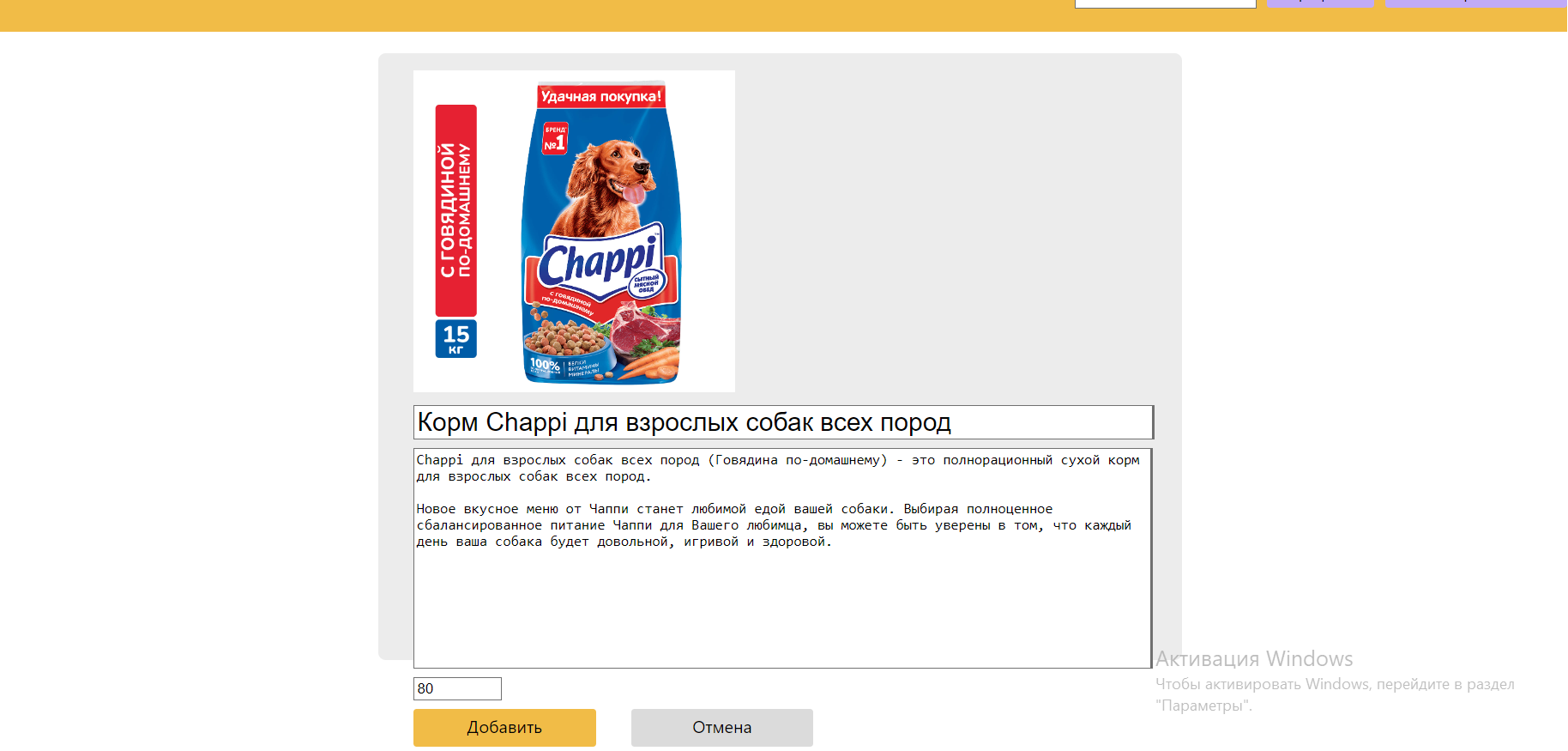
Если нам какой-то товара не нужен, нажимаем на кнопку в виде мусорного ведра, тем самым удалим товар из корзины. После того как корзина удовлетворяет потребности, нажимаем на кнопку «Оформить заказ». После этого товары из корзины пропадут и список товаров отправятся продавцу.

Чтобы войти в администраторскую панель, в форме авторизации в поля номер телефона и пароль вводи admin. После мы попадаем в администраторскую панель. Пример показан на рисунке 2.31.



**Рисунок 2.31 — Администраторская панель**

Администратор может добавить новый товар нажав на кнопку «Добавить товар». После нажатия открывается форма добавления товара. Заполняем все поля и нажимаем на кнопку «Добавить». Пример показан на рисунке 2.32.



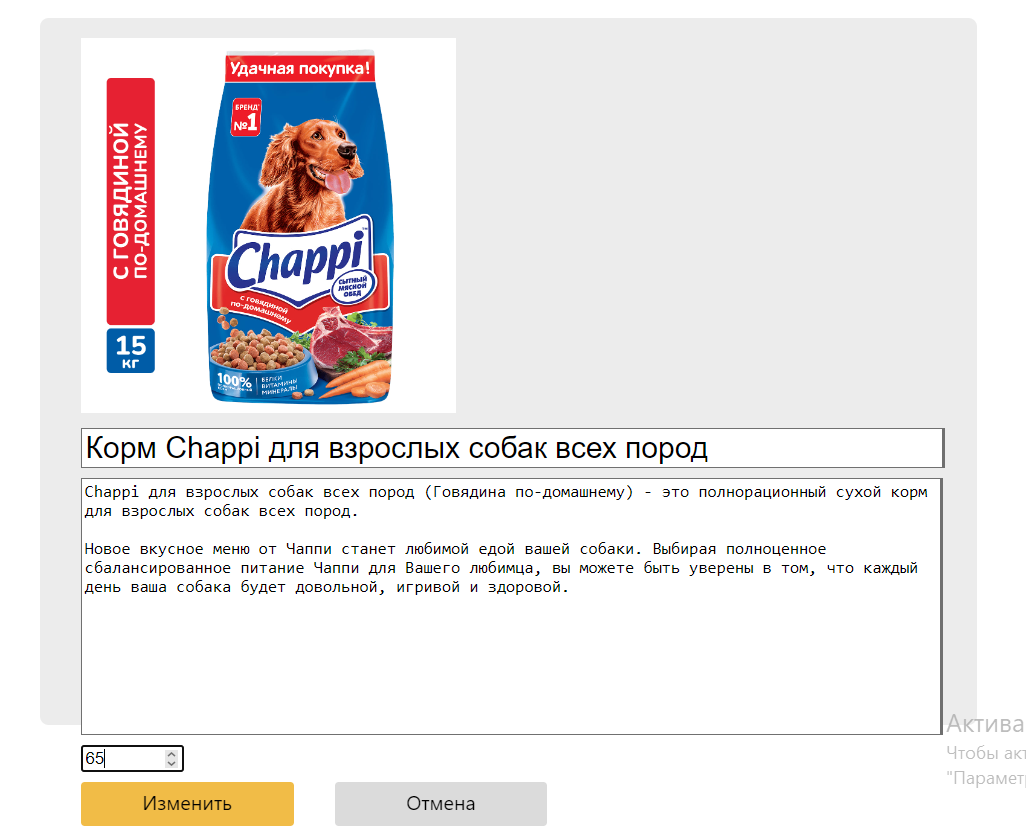
**Рисунок 2.32 — Форма добавления товара**

Если администратор неправильно ввёл какие-либо данные или они изменились, есть возможность изменить товар. Для этого нужно напротив товара нажать кнопку «Изменить». Пример показан на рисунке 2.33.



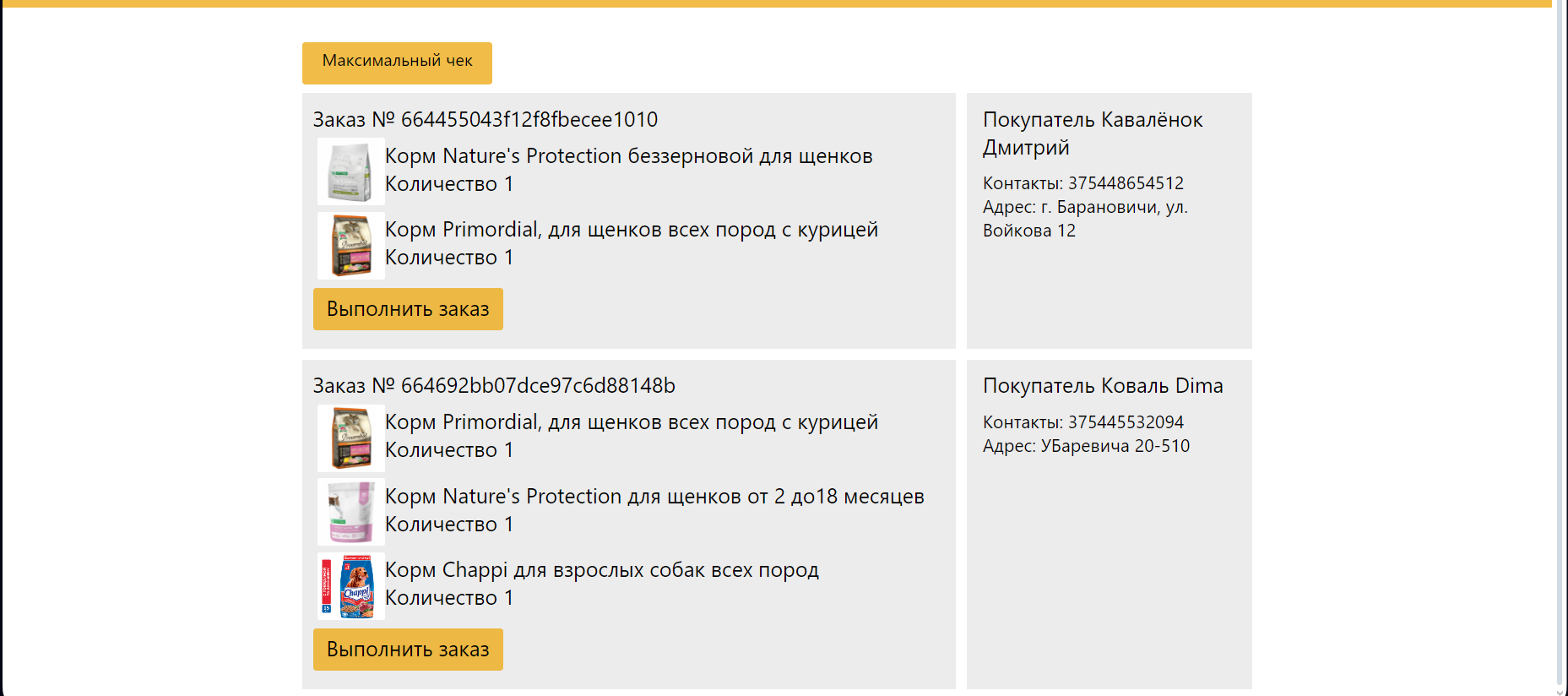
**Рисунок 2.33 — Пример изменения**

После нажатия открывается форма изменения. Изменяем необходимые значения и нажимаем кнопку «Изменить». Пример показан на рисунке 2.34.



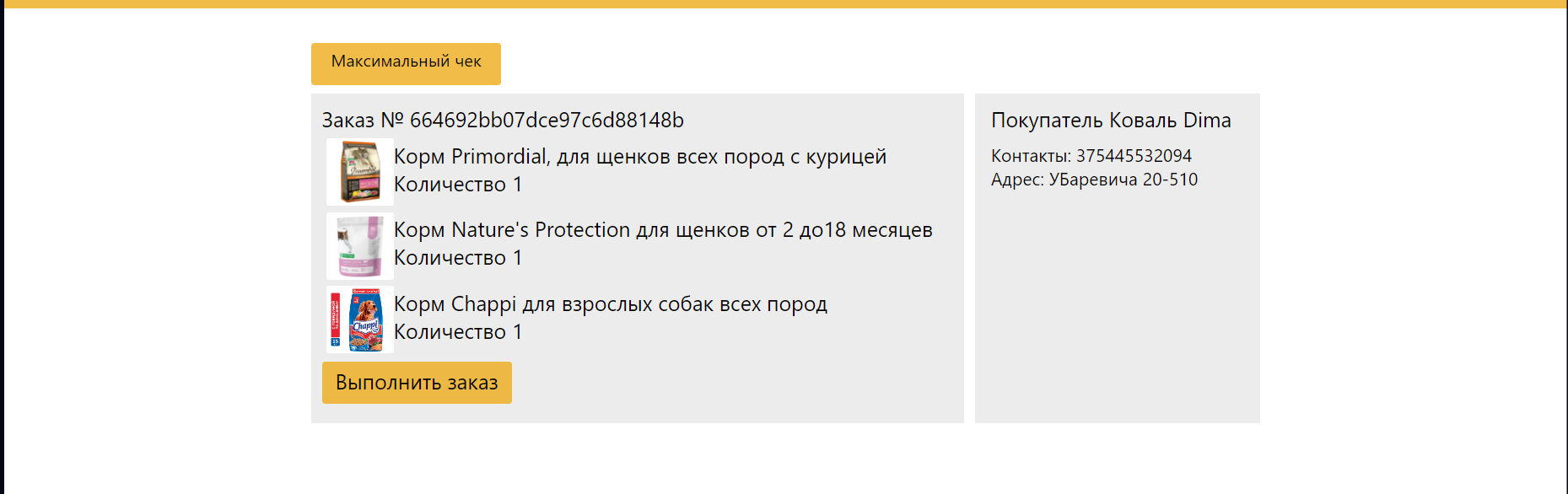
**Рисунок 2.34 — Форма изменения**

Чтобы просмотреть активные заказы, нужно нажать на кнопку «Список заказов». Пример отображения активных заказов представлен на рисунке 2.35.



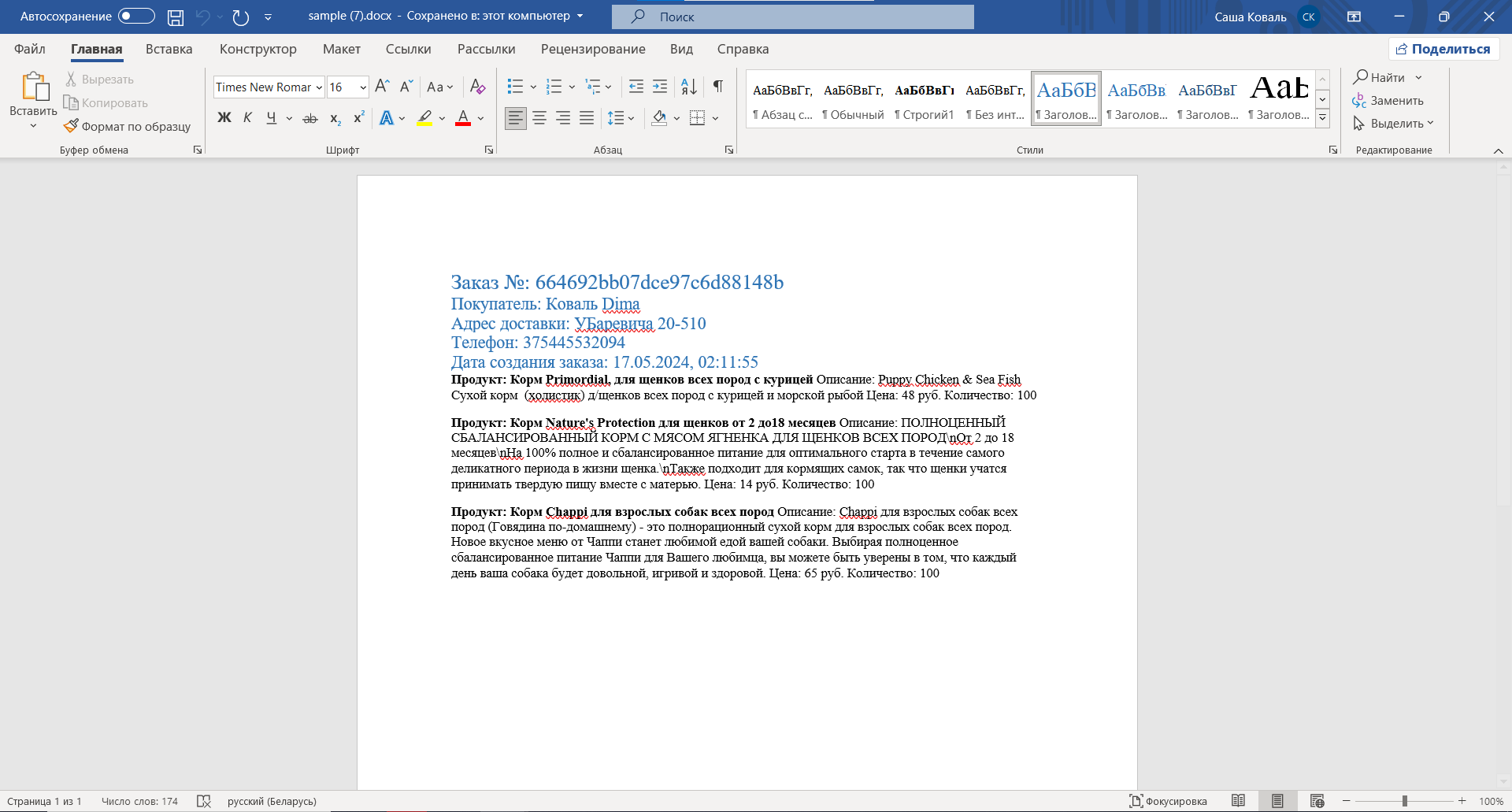
**Рисунок 2.35 — Список заказов**

Чтобы получить заказ с максимальным чеком нажимаем на кнопку «Максимальный чек». Результат показан на рисунке 2.36.



**Рисунок 2.36 — Максимальный чек**

Чтобы выполнить заказ, нужно нажать на кнопку выполнить заказ. После чего он исчезнет из списка и начнётся скачивание отчёта заказа. Пример отчёта показан на рисунке 2.37.



**Рисунок 2.37 — Отчёт заказа**

# ЗАКЛЮЧЕНИЕ

В заключение, создание сайта для заказа товаров с функциональностью просмотра, редактирования, оформления заказов и административной панелью представляет собой важный и комплексный проект для любого бизнеса. Внедрение системы регистрации и авторизации с использованием хеширования паролей обеспечивает высокий уровень безопасности пользовательских данных, защищая их от несанкционированного доступа и злоумышленных атак.

Такой сайт не только улучшает пользовательский опыт, обеспечивая удобство и доступность для заказа товаров, но и оптимизирует бизнес-процессы, автоматизируя управление заказами и запасами товаров. Административная панель позволяет эффективно управлять ассортиментом товаров, их ценами, а также отслеживать и обрабатывать заказы клиентов.

Безопасность данных играет ключевую роль в успехе онлайн-бизнеса. Защищенная система регистрации и авторизации, а также хранение данных в зашифрованном виде помогают предотвратить утечки и несанкционированный доступ к конфиденциальной информации клиентов.

Использование современных технологий и лучших практик веб-разработки не только увеличивает удовлетворенность клиентов, обеспечивая им безопасность и удобство использования сайта, но также повышает эффективность работы администраторов и управленцев, способствуя росту и успеху бизнеса в целом.

Создание сайта для заказа товаров с такими обширными функциональными возможностями представляет собой комплексную задачу, требующую продуманного подхода к разработке и безопасности. Включение функций просмотра и редактирования товаров, оформления заказов, а также наличие административной панели для управления товарами и заказами обеспечат удобство и эффективность как для пользователей, так и для администраторов сайта.

Внедрение системы регистрации и авторизации с хешированием паролей повысит безопасность пользовательских данных, защищая их от несанкционированного доступа. Реализация запросов для работы с товарами, пользователями и заказами обеспечит гибкость и функциональность системы. Использование триггеров для автоматического управления запасами товаров и проверки авторизации пользователей при критических операциях позволит поддерживать целостность данных и безопасность системы.

# СПИСОК **ИСПОЛЬЗУЕМЫХ** ИСТОЧНИКОВ

1. Галимов, Р. Интернет-магазин. Плюсы и минусы интернет-магазинов / Р. Галимов, Э. С. Саитова // Студент и аграрная наука : Материалы IV Всероссийской студенческой конференции, Уфа, 31 марта – 01 2010 года. – Уфа: Башкирский государственный аграрный университет, 2010. — С. 293-294.
2. Красавин, А. С. Разработка сайта с использованием фреймворка React / А. С. Красавин, О. И. Белозеров // Научно-техническое и экономическое сотрудничество стран АТР в XXI веке. – 2023. – Т. 2. — С. 401-404.
3. Клочков, Д. В. Общие компоненты при кроссплатформенной разработке для web-и мобильных приложений с использованием react и react-native / Д. В. Клочков // Молодой ученый. – 2018. – № 36(222). — С. 1-5.
4. Бронская, В. В. Применение JAVASCRIPT-библиотеки REACT для создания приложения интернет-магазина одежды / В. В. Бронская, А. Р. Ибатуллина // Известия высших учебных заведений. Технология текстильной промышленности. – 2023. – № 4(406). — С. 169-175.
5. Косинов, Г. П. Разработка веб-приложения с использованием nodejs и Angular5, разработка сервиса рекомендаций / Г. П. Косинов, П. Ю. Дроздов, А. А. Корчмин // Информационно-коммуникационные технологии: достижения, проблемы, инновации (ИКТ-2018) : Электронный сборник статей I международной научно-практической конференции, посвященной 50-летию Полоцкого государственного университета, Новополоцк, 14–15 июня 2018 года. – Новополоцк: Учреждение образования «Полоцкий государственный университет»=Установа адукацыі "Полацкі дзяржаўны універсітэт", 2018. — С. 33-36.
6. Степовик, А. Н. Практическое применение MongoDB / А. Н. Степовик, Д. А. Замотайлова // Информационное общество: современное состояние и перспективы развития : Сборник материалов XI международного студенческого форума, Краснодар, 23–27 июля 2018 года. – Краснодар: Кубанский государственный аграрный университет имени И.Т. Трубилина, 2018. — С. 305-306.

# ПРИЛОЖЕНИЯ

**ПРИЛОЖЕНИЕ А**

**Главная страница**

import logo from '../logo.svg';

import './Home.css';

import { ProductCard } from '../components/ProductCard';

import { Header } from '../components/Header';

import { Link } from 'react-router-dom';

import { useEffect, useState } from 'react';

import axios from "axios";

import { useSelector } from 'react-redux';

import { selectAdmin } from '../redux/slices/auth';

import { Navigate } from 'react-router-dom';

export function Home({user}) {

const [items, setData] = useState([]);

const auth = useSelector(selectAdmin);

useEffect(()=>{

const data = axios.get('http://localhost:4444/products')

.then(({data})=>{

setData(data);

});

},[])

try {

console.log(auth.user)

if (auth.user == "admin") {

return <Navigate to={"/admin"}></Navigate>

}

} catch (error) {

}

return (

<div className='App'>

<div>{user}</div>

<div className="ListProducts">

{items.map((obj)=>

<ProductCard obj={obj}/>)}

</div>

</div>

);

}

**ПРИЛОЖЕНИЕ Б**

**Регистрация пользователей**

 //регистрация

app.post('/auth/register', async (req, res)=>{

const password = req.body.password;

const salt = await bcrypt.genSalt(10);

const passwordHash = await bcrypt.hash(password, salt);

const doc = new BuyerModel({

surName: req.body.surName ,

firstName: req.body.firstName ,

delivery\_address: req.body.delivery\_address ,

phoneNumber: req.body.phoneNumber ,

passwordHash: passwordHash ,

});

const buyer = await doc.save();

return res.json({buyer});

});

**ПРИЛОЖЕНИЕ В**

**Авторизация пользователей**

//авторизация

app.post("/auth/login", async (req, res)=>{

try {

if (req.body.phoneNumber == "admin" && req.body.password == "admin") {

const token = jwt.sign({

\_id: "admin",

}, 'secret123', {expiresIn: '30d'});

return res.json({

"user": "admin",

"token": token

});

}

const buyer = await BuyerModel.findOne({phoneNumber: req.body.phoneNumber});

if (!buyer){

return res.status(404).json({

message: "Пользователь не найден",

});

}

const isValuePass = await bcrypt.compare(req.body.password, buyer.\_doc.passwordHash); //возвращает true or false

if (!isValuePass) {

return res.status(404).json({

message: "Неверный логин или пароль",

});

}

const token = jwt.sign({

\_id: buyer.\_id,

}, 'secret123', {expiresIn: '30d'});

res.json({

...buyer.\_doc,

token

});

} catch (error) {

return res.json({"ошибка": error});

}

});

app.get('/auth/me', checkAutharization, async (req, res)=>{

try {

if (req.userId=="admin") {

return res.json({

"user": "admin"}

)

}

const user = await BuyerModel.findById(req.userId);

if (!user) {

return res.status(404).json({

message: "Такого пользователя нет",

});

}

res.json(

user

)

} catch (error) {

}

**ПРИЛОЖЕНИЕ Г**

**Проверка токена авторизации**

export default (req, res, next)=> {

const token = (req.headers.authorization || '').replace(/Bearer\s?/, '');

if (token) {

try {

const decoded = jwt.verify(token, 'secret123');

if (decoded.\_id == "admin") {

req.userId == "admin"

next()

}

req.userId = decoded.\_id

next();

} catch (error) {

return res.status(403).json({

message: 'Нет доступа',

})

}

} else{

return res.status(403).json({

message: 'Нет доступа',

})

}

};

**ПРИЛОЖЕНИЕ Д**

**Получить один товар**

//Получить один товар

app.post("/products/:id", async (req, res)=>{

try {

const productId = req.params.id;

const allProducts = await CardProduct.findOne({

\_id: productId,

});

return res.status(200).json(allProducts);

} catch (error) {

}

});

**ПРИЛОЖЕНИЕ Е**

**Добавление товара в корзину**

//Добавить товар в корзину

app.post("/basket/:idProduct", checkAutharization, async (req, res)=>{

try {

console.log('rwerwee')

const doc = new BasketModel({

buyer: req.userId,

productCards: req.params.idProduct,

count: req.body.count

});

console.log("fdsfdd")

const Basket = await doc.save();

const buyerId = req.userId;

const allProductsBasket = await BasketModel.find({

buyer: buyerId

}).populate('productCards').populate('buyer').exec()

return res.status(200).json(allProductsBasket);

} catch (error) {

}

});

**ПРИЛОЖЕНИЕ Е**

**Получить все товары с корзины**

//Получить товары с корзины

app.get("/basket", checkAutharization, async (req, res)=>{

    try {

        const buyerId = req.userId;

        const allProductsBasket = await BasketModel.find({

            buyer: buyerId

        }).populate('productCards').populate('buyer').exec()

        return res.status(200).json(allProductsBasket);

    } catch (error) {

        return res.status(400).json({"message": error})

    }

});

**ПРИЛОЖЕНИЕ Ж**

**Получить список заказов**

//Получить список заказов

app.get("/orders/list", async (req, res)=>{

    try {

        const allOrders = await OrdersModel.find().populate("buyerId")

        for (let i = 0; i < allOrders.length; i++) {

            const element = allOrders[i].listProducts;

            allOrders[i].listProducts = i;

            // console.log(allOrders[i]);

            const listDataProducts = []

            for (let j = 0; j < element.length; j++) {

                const elementOne = element[j];

                // console.log(elementOne)

                const allProducts = await CardProduct.findOne({

                    \_id: elementOne,

                });

                listDataProducts.push(allProducts);

            }

            allOrders[i].listProducts = listDataProducts;

            // console.log(element)

        }

        console.log(allOrders);

        return res.status(200).json({allOrders

        });

    } catch (error) {

        console.log(error)

        return res.status(400).json({"message": error})

    }

})

**ПРИЛОЖЕНИЕ З**

**Модель, описывающая таблицу корзины**

import mongoose from "mongoose";

const BasketSchema = new mongoose.Schema({

    buyer: {

        type: mongoose.Schema.Types.ObjectId,

        ref: "Buyer",

        required: true

    },

    productCards: {

        type: mongoose.Schema.Types.ObjectId,

        ref: "CardProduct",

        required: true

    },

    count: {

        type: Number

    }

});

export default mongoose.model('Basket', BasketSchema)

**ПРИЛОЖЕНИЕ И**

**Модель, описывающая таблицу покупателя**

import mongoose from "mongoose";

const BuyerSchema = new mongoose.Schema({

    surName: {

        type: String,

        required:true,

    },

    firstName: {

        type: String,

        required: true,

    },

    delivery\_address: {

        type: String,

        required: true,

    },

    phoneNumber: {

        type: String,

        required: true,

        unique: true,

    },

    passwordHash:{

        type: String,

        required: true,

    }

});

export default mongoose.model('Buyer', BuyerSchema)

**ПРИЛОЖЕНИЕ К**

**Модель, описывающая таблицу карточки товара**

import mongoose from "mongoose";

const CardProductSchema = new mongoose.Schema({

    title: {

        type: String,

        required: true,

    },

    descriptionProduct: {

        type: String,

        required: true,

    },

    price:{

        type: Number,

        required: true,

    },

    quantity\_goods:{

        type: Number,

        required: true

    },

    urlImage: String

});

export default mongoose.model('CardProduct', CardProductSchema)

**ПРИЛОЖЕНИЕ Л**

**Модель, описывающая таблицу список заказов**

import mongoose from "mongoose";

const Orders = new mongoose.Schema({

    buyerId: {

        type: mongoose.Schema.Types.ObjectId,

        ref: "Buyer",

        required: true

    },

    listProducts: {

        type: Array,

        default: []

    }

},

{

    timestamps: true

});

export default mongoose.model('Orders', Orders)