**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 маг азина по продаже кормов** |
|  | **для живтоных** |

|  |
| --- |
| **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: 22 с., 20 рис., 6 источников, 2 прил., 2 табл.

UNITY, UNITY3D, ГЕОМЕТРИЧЕСКИЕ ПРИМИТИВЫ, РАЗРАБОТКА ПРИЛОЖЕНИЯ, С#, КЛАСС, WINDOWS.

Объект исследования — среда разработки React, NodeJS.

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

Цель проекта — разаработка 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 5](#_Toc166661496)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 7](#_Toc166661497)

[2.1 Построение и описание моделей базы данных 7](#_Toc166661498)

[2.2 Построение UML-диаграмм проекта 8](#_Toc166661499)

[2.3 Описание разработанных триггеров и хранимых процедур 13](#_Toc166661500)

[2.4 Описание методов защиты информации в приложении 14](#_Toc166661501)

[2.4 Руководство пользователя программным продуктом 16](#_Toc166661502)

[ЗАКЛЮЧЕНИЕ 21](#_Toc166661503)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 22](#_Toc166661504)

[ПРИЛОЖЕНИЯ 23](#_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. 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 адмнистраторскую панель, в которй должен быть реализован следующий функционал: добавлние новых товаров, редатикрование товраов, 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, получить список товаров для конкретного заказа.

## 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, а также аксессуары и средства по уходу за животными. 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.

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.
* 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.

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.
* 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.

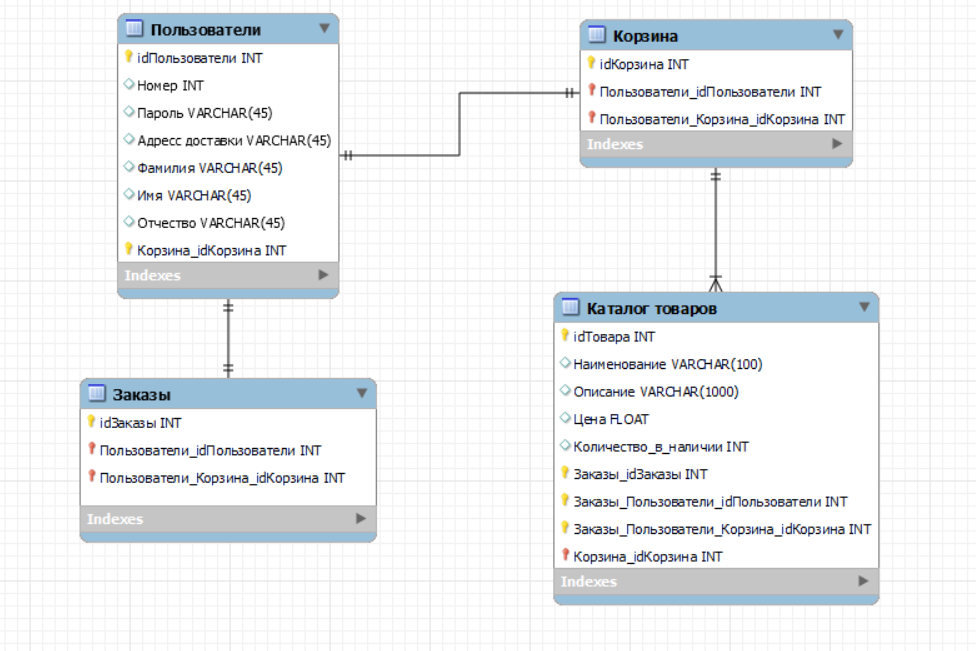
# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 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.

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 2.1.



**Рисунок 2.1 — Физическая модель база 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

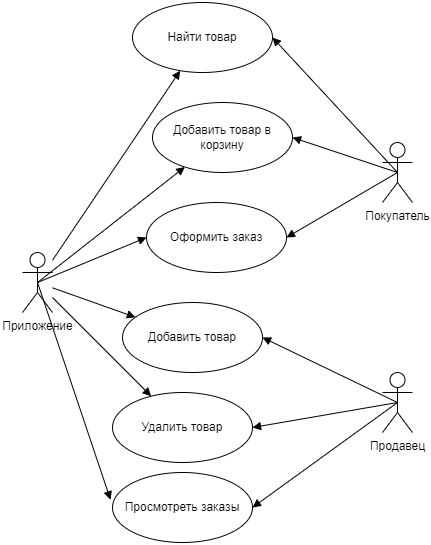
Диаграмма вариантов использования (use case diagram) представляет собой визуальное описание взаимодействий между пользователями (акторами) и системой для выполнения определённых функций (вариантов использования). Основные элементы и их функции включают.

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

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

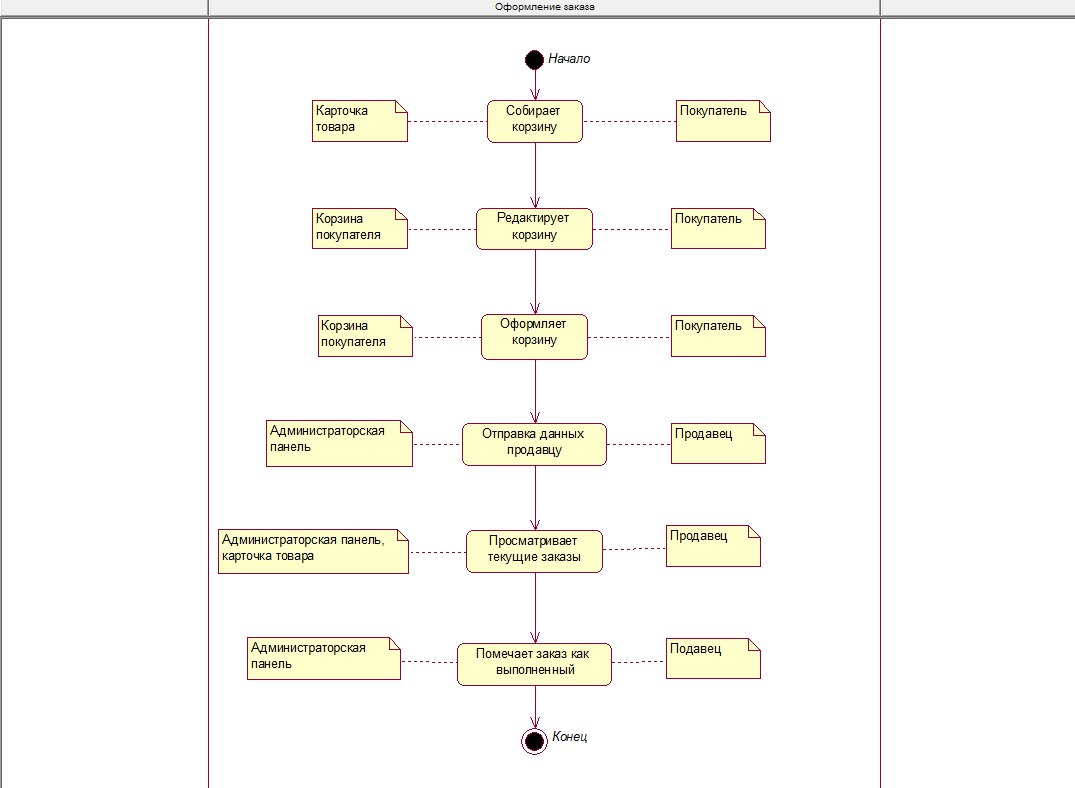
Приложение: Обеспечивает функциональность для нахождения, принятия, добавления и удаления товаров, а также для добавления товаров в корзину, оформления заказов и просмотра заказов.

На рисунке 2.2 представлиена use case диаграмма разрабатываемой информационной системы.



**Рисунок 2.2 — Диаграмма вариантов использования**

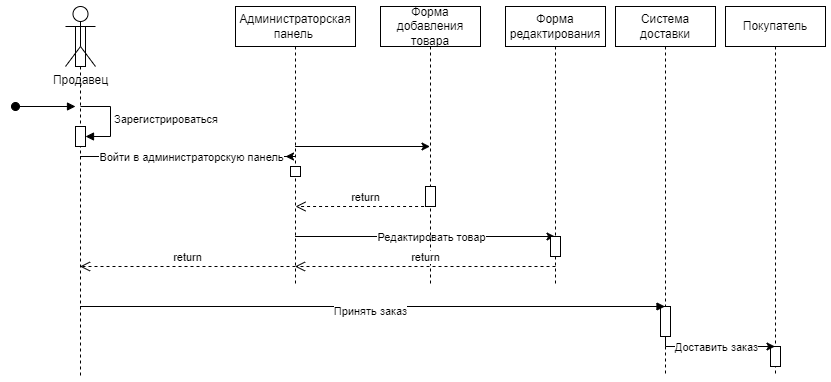
Диаграмма деятельности — это один из типов диаграмм UML, используемый для моделирования динамического поведения системы. Она помогает визуализировать последовательность действий и их потоки управления в процессе выполнения определенного процесса или функции. На рисунке 2.3 представлена диаграмма деятельности разрабатываемой инфомрационной системы.



**Рисунок 2.2 — Диаграмма деятельности**

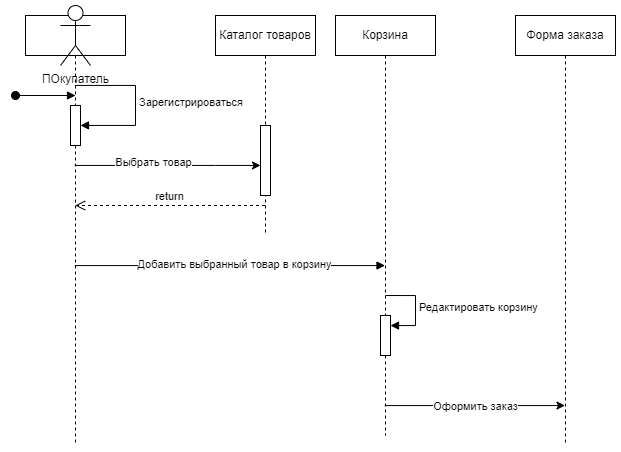
Эта диаграмма демонстрирует весь процесс оформления заказа от начальной стадии добавления товаров в корзину покупателем до завершения заказа продавцом. Она отражает взаимодействие между покупателем и продавцом, а также использование различных компонентов системы, таких как карточка товара, корзина покупателя 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 2.3 представлена диаграмма последовательности для продавца. На рисунке 2.4 представлена диагрмма последовательности для попкупателя



**Рисунок 2.3 — Диаграмма последоавтельности для продавца**

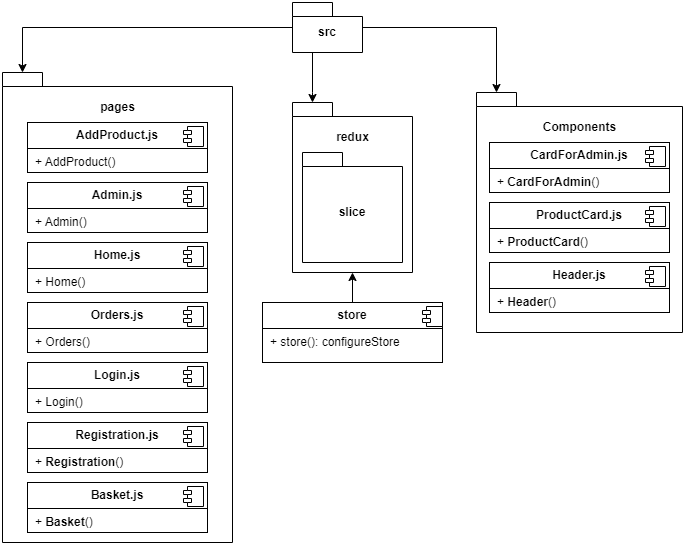
Описание взаимодействий: регистрация продавца, вход в администраторскую панель, продавец открывает форму добавления товара через администраторскую панель, продавец редактирует товар через форму редактирования. продавец принимает заказ через администраторскую панель, администраторская панель отправляет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



**Рисунок 2.4 — Диаграмма последоавтельности для покупателя**

Описание взаимодействий: регистрация покупателя, покупаетль 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 2.5. 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

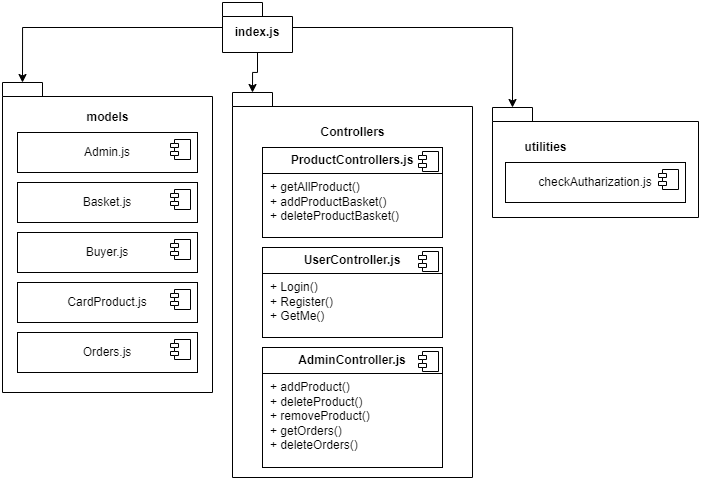


**Рисунок 2.5 — Диаграмма компонетов**

Таблица 2.1 – Описание диаграммы компонентов

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

Диаграмма компонентов представленая на риснуке 2.6, 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 реалищуют обработку запросов на сервер. Пакет *utilities* хранит в себе функцию для проверки авторизации пользователя



**Рисунок 2.*6* — Диаграмма *компонетов***

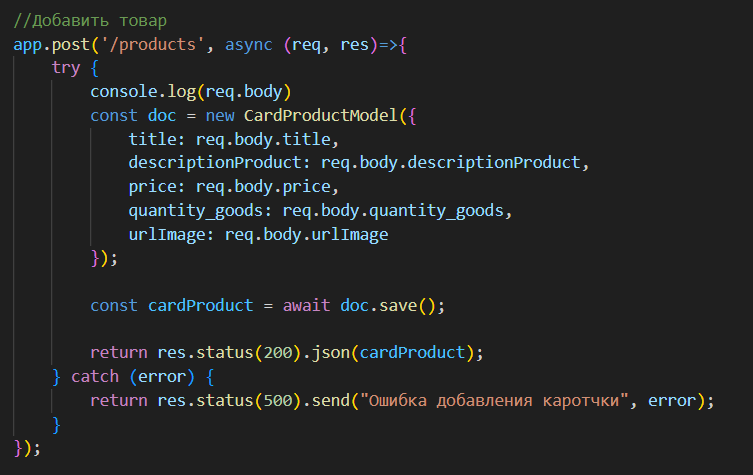
Таблица 2.2 – Описание функционла на стороне сервера

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| getAllProduct() | Возвращает все товары из базы данных |
| addProductBasket() | Реализует добавление товаров в корзину и в базу данных |
| deleteProductBasket() | Удалает 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 выполненый заказ |

## 2.3 Описание разработанных триггеров и хранимых процедур

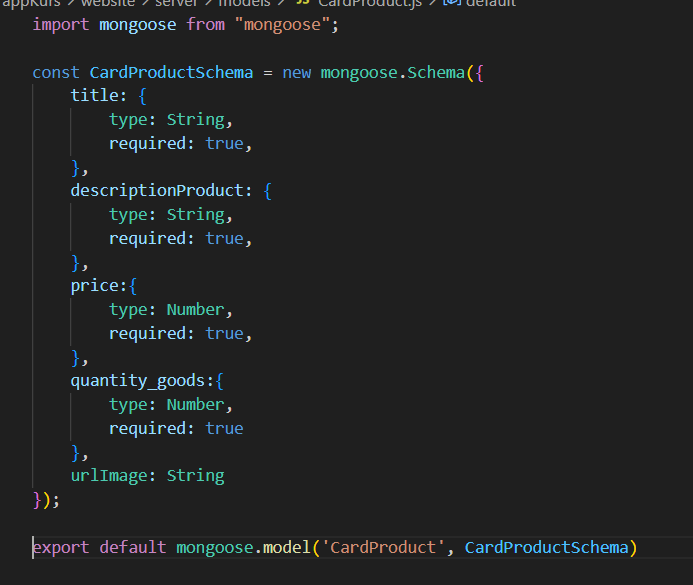
Работа с базами данных на сайте реализовано через запросы на сервер, который обрабатывает данные и заносит их в базу данных.

Пример добавления товара в каталог товаров. Предствален на рисунке 2.7.



**Рисунок 2.*7* — Запрос выполяющий добавление товара**

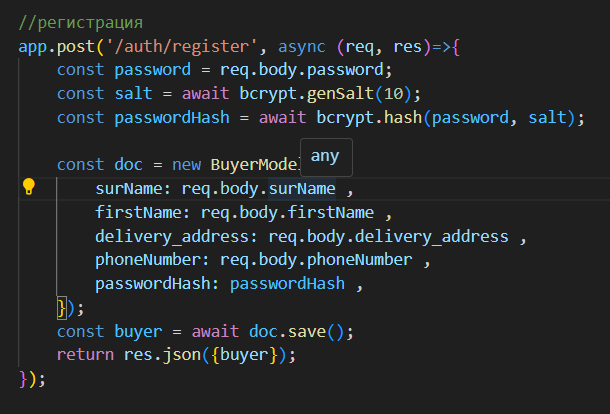
Пример описания таблицы базы данных представлен на риснуке 2.8.



**Рисунок 2.*8* — Описание карточки товара**

## 2.4 Описание методов защиты информации в приложении

Реализовано несколько методов защиты данных. При регистрации пользователей ввденный пароль шифруется. Код реализущий шифрование показан на рисунке 2.9. Шифрование реализуется с помощью библиотеки bcrypt.

****

**Рисунок 2.*9* — Описание карточки товара**

При каждом запросе, где необходимо получить данные о конкретном пользователе, выполняется метод *checkAutharization.* Который проверяет пользователя с поомщью токена авторизации. Код реализующий метод *checkAutharization*  представлен на рисунке 2.10.



**Рисунок 2.*10* — Проверка токена авторизации**

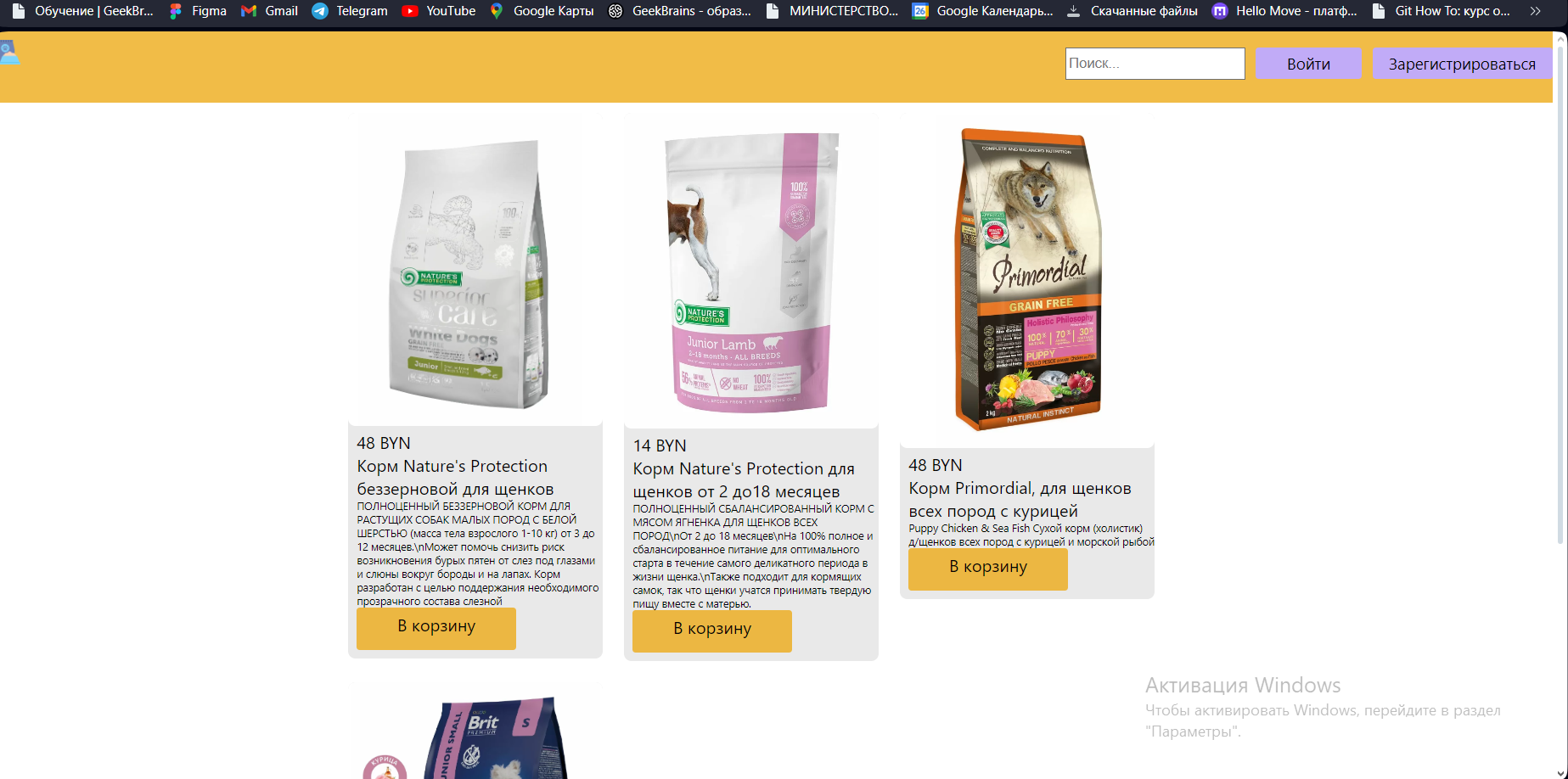
Сам же токен генерируется в момент авторизации пользователя. Код реализующий генерация токена показан на рисунке 2.11.



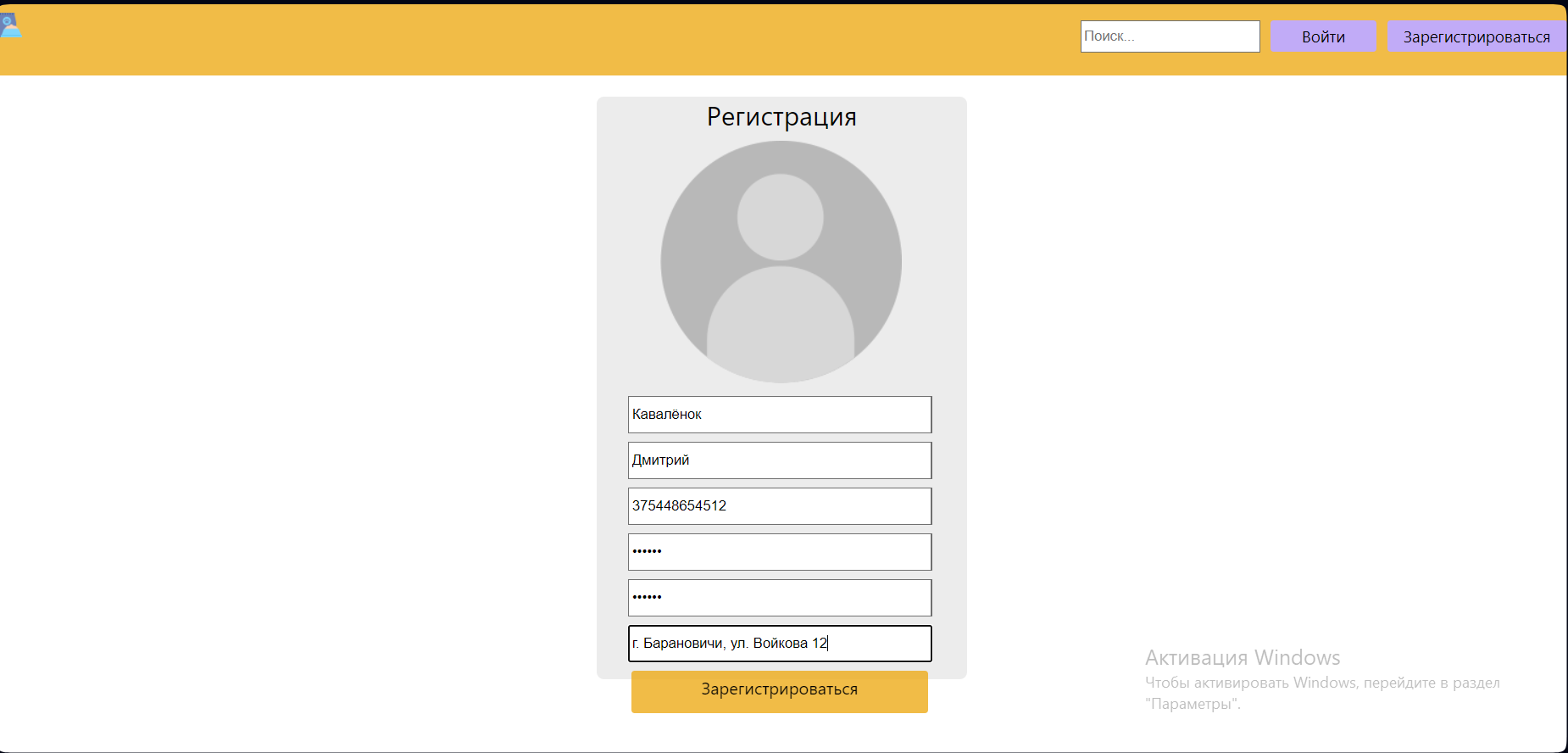
**Рисунок 2.*11* — Генерация токена**

## 2.4 Руководство пользователя программным продуктом

При первом запуске на встречает главаня страница, на которой находится все товары. Пример показан на рисунке 2.12

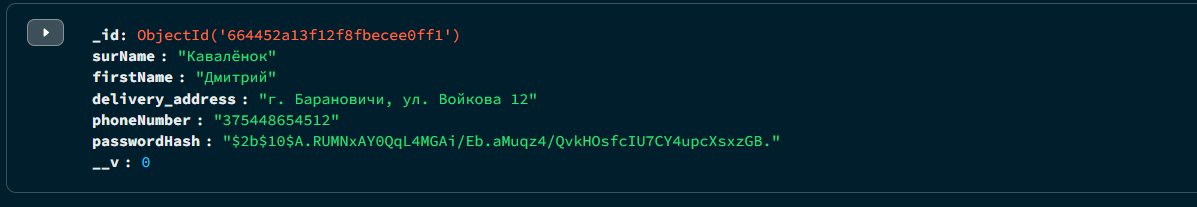


**Рисунок 2.*12* — Главня страница**

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 заругистрироваться. Нужно надать на кнопку регистрация. После нажатия пользователь попдает на форму регистрации. Заполняем все поля и нажимаем на кнопку регитсрация. Пример показн на риунке 2.13  


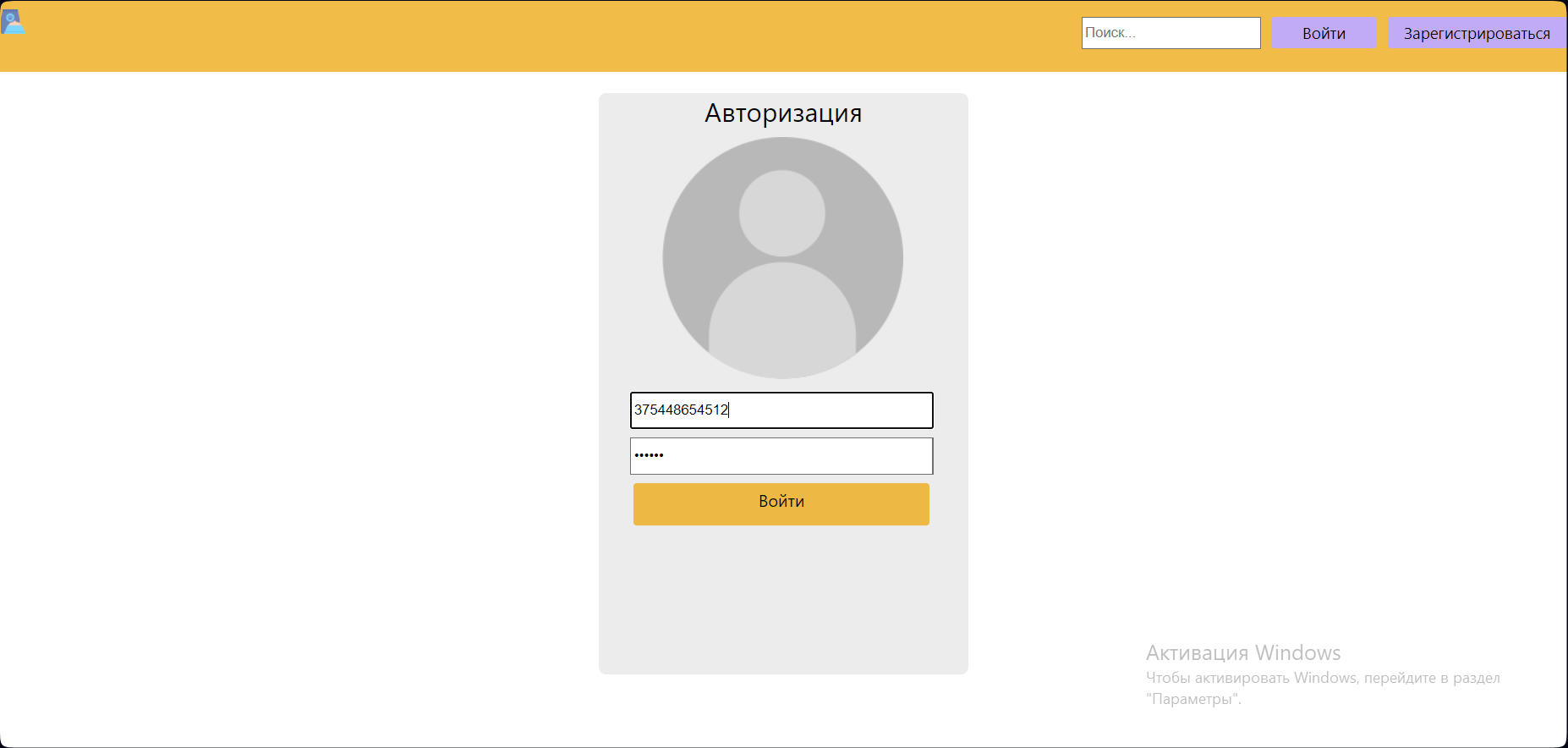
**Рисунок 2.*13* — Регистрация**

После регистрации новый пользователь появится в базе данных. Рисунок 2.14



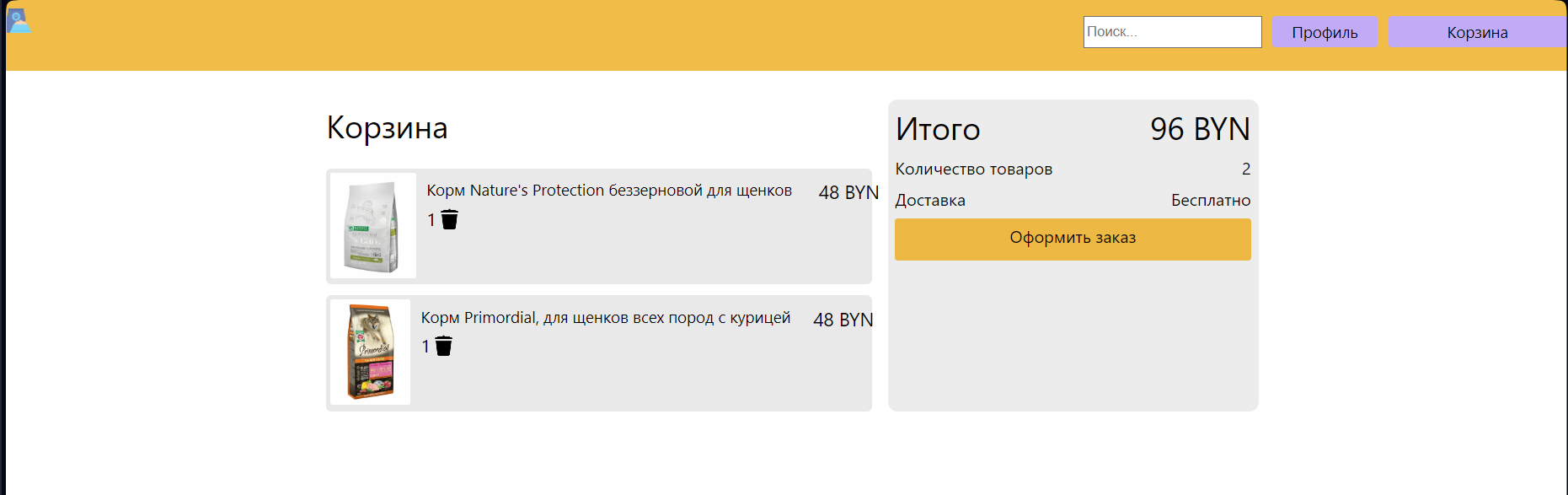
**Рисунок 2.*13* — Пользователь в базе данных**

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 2.14.



**Рисунок 2.*14* — Форма авторизации**

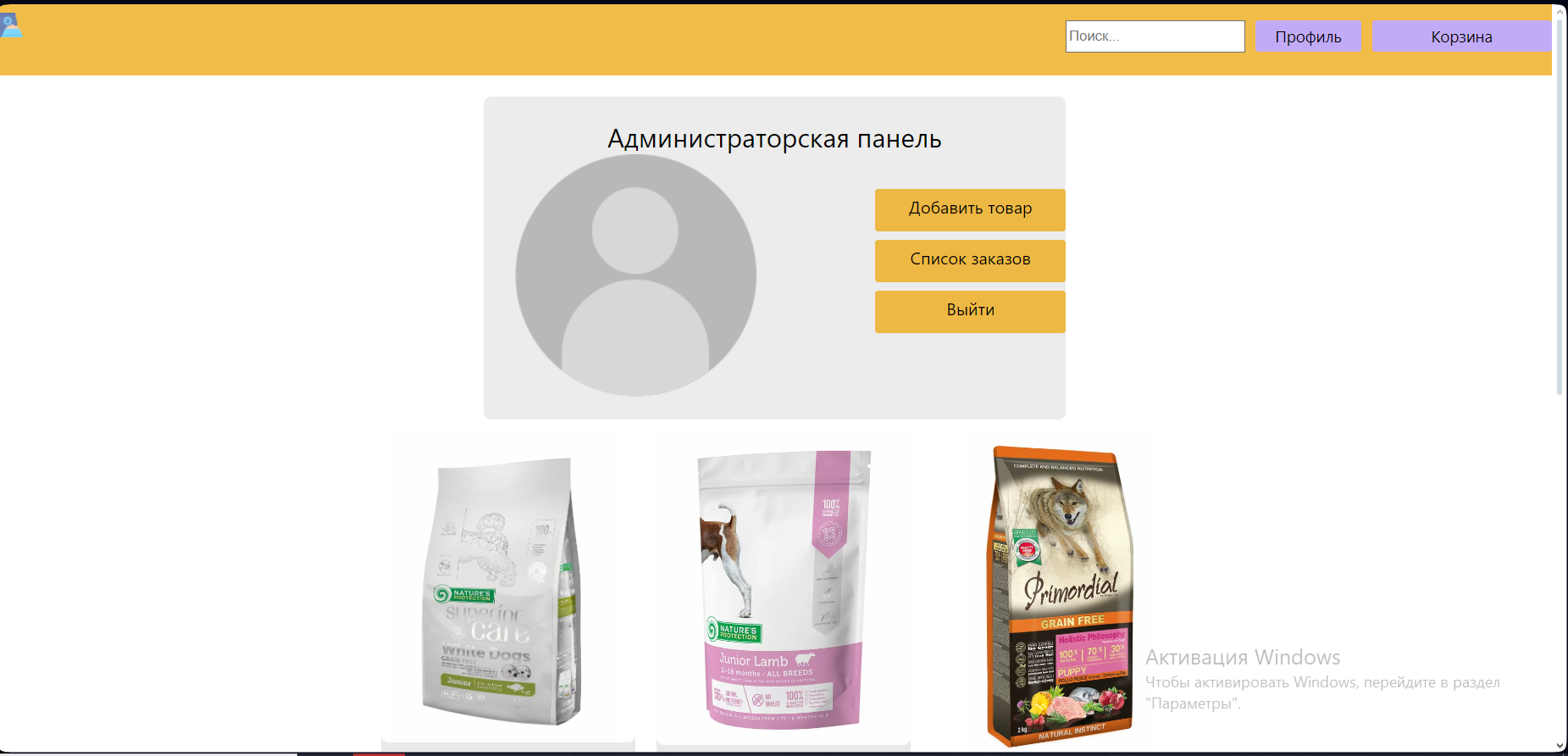
Теперь пользователь может добавлять товар в корзину. Чтобы добавить товар в корзину нужно нажать на кнопку «В корзину» на соответствующем товаре. После нажимаем кнопку *Корзина,* котором находятся все товары, которые мы добавили. Пример показан на рисунке 2.15.



**Рисунок 2.*15* — Корзина**

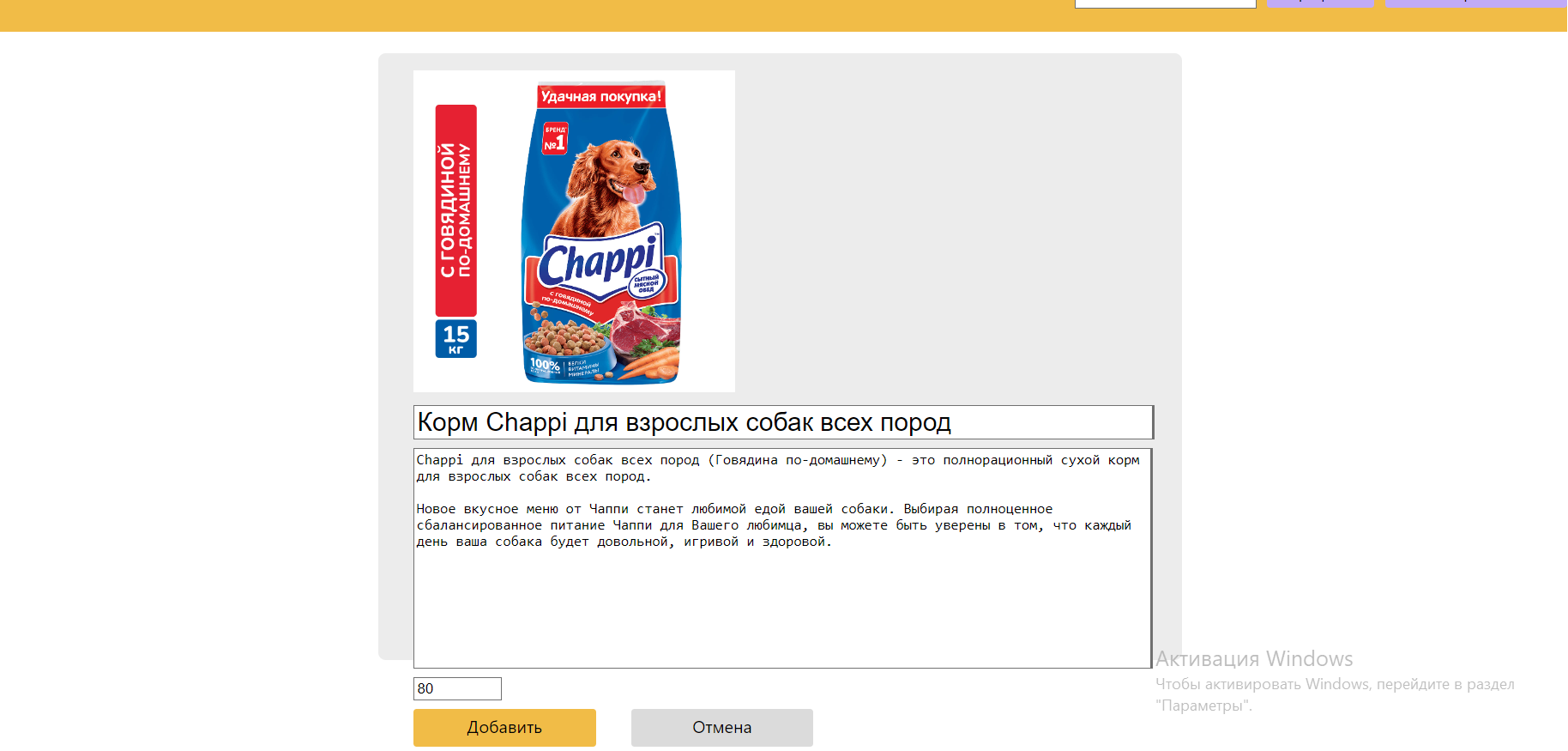
Если нам какой-то товра не нужен, нажимаем на кнопку в виде мусорного ведра, тем самым удалим товар из корзины. После того как корзина удовлетворяет потребности. Нажимаем на кнопку «Оформить закза». После этого товары из корзины пропадут и отправться продавцу.

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 2.16.



**Рисунок 2.*16* — 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 2.17.



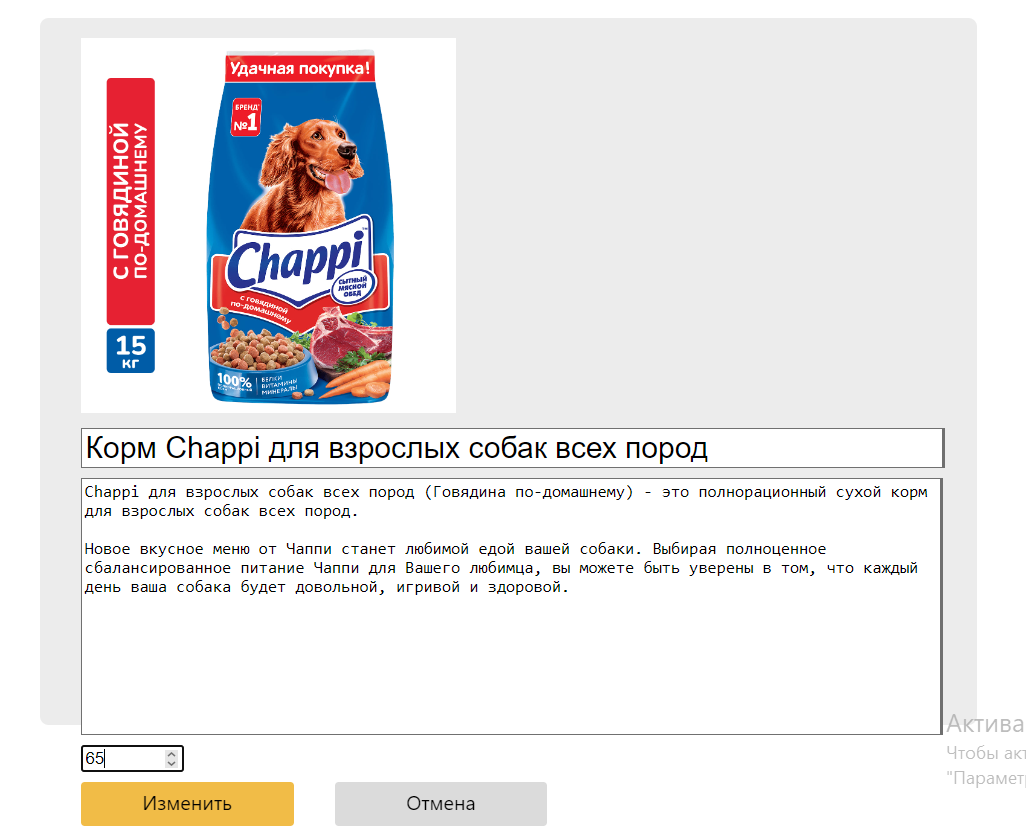
**Рисунок 2.*17* — Форма добавления товара**

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 риснуке 2.18.



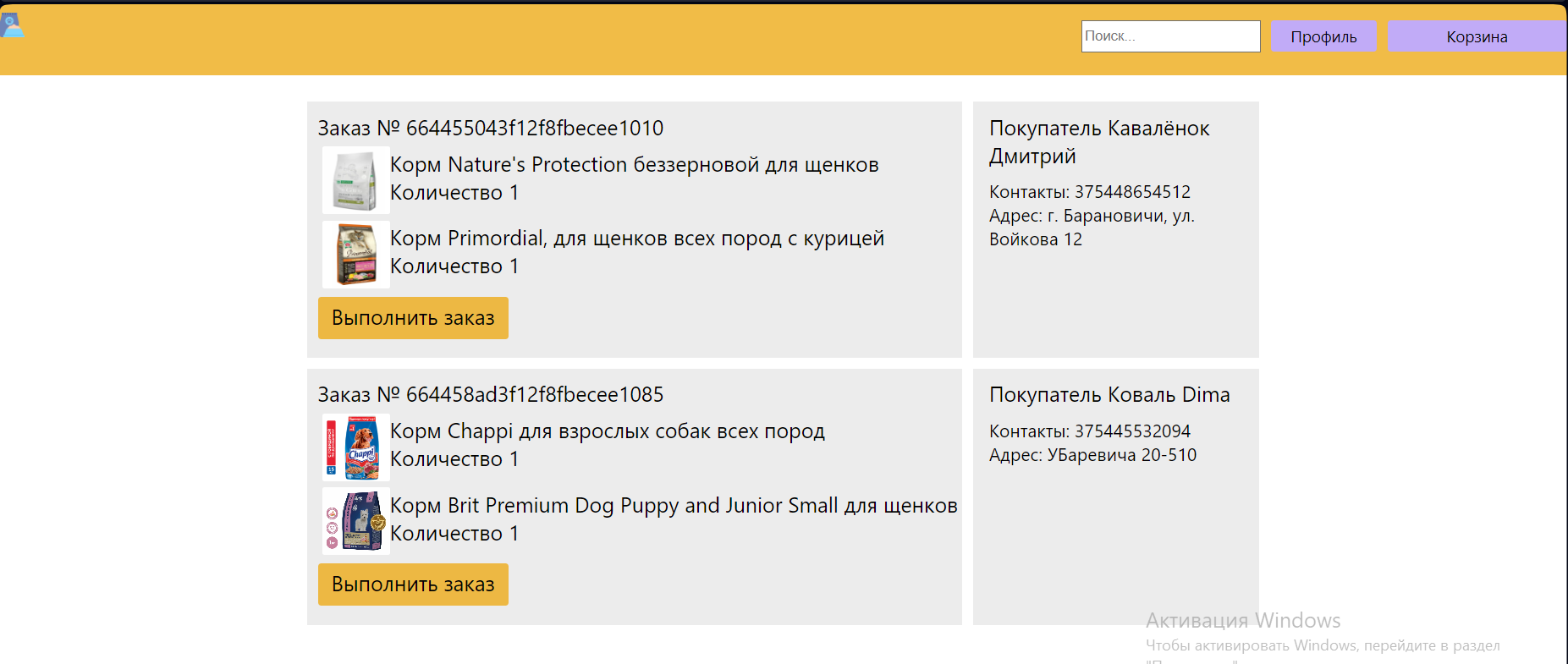
**Рисунок 2.*18* — Пример изменения**

После нажатия открывается форма изменения. Изменяем необоходимые значения и нажимаем кнопку «Изменить». Пример показан на риснуке 2.19.



**Рисунок 2.*19* — Форма изменения**

Чтобы просмотреть активные заказы, нужно нажать на кнопку «Список заказов». Пример отображения активных заказов представлен на рисунке 2.20



**Рисунок 2.*20* — Список заказов**

# ЗАКЛЮЧЕНИЕ

В заключение, создание сайта для заказа товаров, включающего функции просмотра и редактирования товаров, оформления заказов, а также административную панель для управления товарами и заказами, является комплексным проектом. Реализация системы регистрации и авторизации с хешированием паролей обеспечит безопасность пользовательских данных. Такой сайт не только улучшит пользовательский опыт и оптимизирует бизнес-процессы, но и повысит безопасность данных, что приведет к большей удовлетворенности клиентов и повышению эффективности работы администраторов.

# СПИСОК **ИСПОЛЬЗУЕМЫХ** ИСТОЧНИКОВ

1. [СРАВНИТЕЛЬНЫЙ АНАЛИЗ ТЕХНОЛОГИЙ УПРАВЛЕНИЯ СОСТОЯНИЕМ В REACT ПРИЛОЖЕНИЯХ: REDUX, MOBX И REACT CONTEXT (REACT HOOKS)](https://www.elibrary.ru/item.asp?id=60031245), *Мелихова К.А., Томилин Е.С., Кропотов А.С.*— c. 395-402.
2. [ОБЩИЕ КОМПОНЕНТЫ ПРИ КРОССПЛАТФОРМЕННОЙ РАЗРАБОТКЕ ДЛЯ WEB-И МОБИЛЬНЫХ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ REACT И REACT-NATIVE](https://www.elibrary.ru/item.asp?id=35588624), *Клочков Д.В* — С. 1-5.
3. [РАЗРАБОТКА АВТОНОМНОГО УСТРОЙСТВА МОНИТОРИНГА ОКРУЖАЮЩЕЙ СРЕДЫ НА ПЛАТФОРМЕ RASPBERRYPI И ФРЕЙМВОРКЕ NODEJS](https://www.elibrary.ru/item.asp?id=32755247), *Гусаров*— c. 118-122.
4. [РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ NODEJS И ANGULAR5, РАЗРАБОТКА СЕРВИСА РЕКОМЕНДАЦИЙ](https://www.elibrary.ru/item.asp?id=35564528)  
   *Косинов Г.П., Дроздов П.Ю., Корчмин А.А.*— С. 33-36.
5. [РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ С ПОМОЩЬЮ NODEJS](https://www.elibrary.ru/item.asp?id=41360604)  
   *Цыплаков В.С.* — С. 29-30.
6. [ОЦЕНКА ИСПОЛЬЗОВАНИЯ PROMETHEUS И GRAFANA ДЛЯ МОНИТОРИНГА БАЗ ДАННЫХ MONGODB](https://www.elibrary.ru/item.asp?id=49461999)  
   *Косенков В.В., Богданов А.В.* — С. 154-164.

# ПРИЛОЖЕНИЯ

**Приложение А**

**Главня 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',

})

}

};