

Министерство образования Иркутской области
Государственное бюджетное профессиональное
образовательное учреждение Иркутской области
«Иркутский авиационный техникум»
(ГБПОУИО «ИАТ»)

ДП.09.02.07-3.24.201.15.ПЗ

УТВЕРЖДАЮ

Зам. директора по УР, к.т.н.

_____ Е.А. Коробкова

**ИНТЕРНЕТ-МАГАЗИН
«СЛАДОСТИ»**

Нормоконтролер:

(подпись, дата)

(С.Е. Кудрявцева)

Руководитель:

(подпись, дата)

(Ю.Ю. Хромовских)

Студент:

(подпись, дата)

(В.О. Никифоров)

Иркутск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Предпроектное исследование.....	5
1.1 Описание предметной области.....	5
1.2 Анализ инструментальных средств реализации	7
2 Техническое задание на разработку программного продукта.....	12
3 Проектирование	13
3.1 Архитектура программного продукта	13
3.2 Функциональное проектирование.....	14
3.3 Проектирование базы данных	19
3.4 Проектирование пользовательского интерфейса	22
4 Реализация программного продукта	27
5 Отладка и тестирование программного продукта.....	35
6 Руководство пользователя программного продукта	40
6.1 Руководство администратора интернет-магазина.....	42
ЗАКЛЮЧЕНИЕ	51
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	52
Приложение А – Листинг кода сервера	54
Приложение Б – Работа с файлами	56
Приложение В – Создание заказа.....	58
Приложение Г – Тесты функций авторизации	60

					ДП.09.02.07-3.24.201.15.ПЗ							
Изм.	Лист	№ докум.	Подпись	Дата								
Разраб.		Никифоров В.О.			ИНТЕРНЕТ-МАГАЗИН «СЛАДОСТИ» пояснительная записка			Лит.	Лист	Листов		
Провер.		Хромовских Ю.Ю.								2	62	
Реценз.								ГБПОУИО «ИАТ» ВЕБ-20-1				
Н. Контр.		Кудрявцева Е.С.										
Утверд.		Коробкова Е.А.										

ВВЕДЕНИЕ

В интернет-магазине «Сладости» можно купить множество различных сладостей и вкусностей, а также использовать другой функционал магазина.

Актуальность данного интернет-магазина заключается в том, что такая еда как сладости будут актуальны на протяжении многих десятков лет, так как люди очень любят иногда себя побаловать, данный интернет-магазин должен обеспечить это и помочь удобнее выбрать и найти свои любимые сладости.

В данный момент существует множество похожих магазинов и нужно взять самое лучшее и объединить всё в одном месте. Чтобы он был наиболее удобен в использовании для пользователя нужно соблюдать следующие требования:

- краткая и понятная информация;
- внешний вид должен выглядеть хорошо и современно;
- навигация по сайту должна быть интуитивно понятна.

При соблюдении данных требований, магазин преобразится в лучшую сторону и станет приятным для посещения пользователю.

Целью дипломного проекта является разработка интернет-магазина сладостей, который позволит более удобно покупать сладости в интернете для пользователей, а также удобное и понятное пользование для всего персонала.

Основными задачами данного проекта являются:

- произвести исследование деятельности магазина сладостей;
- создать структурную и функциональную схемы;
- спроектировать базу данных интернет-магазина;
- разработать интерфейс пользователя в соответствии с техническим заданием;
- разработать интернет-магазин сладостей и составить программную документацию в виде руководства пользователя.

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

Интернет-магазин «Сладости» будет предоставлять не только возможность покупки товаров, но и другой функционал для пользователя, такой как:

- добавление товара в избранное;
- просмотр избранных товаров;
- установка своей даты рождения, для дальнейшего получения акций;
- просмотр, редактирование и удаление своего аккаунта;
- просмотр и повтор выполненных заказов;
- просмотр статуса заказов.

Также магазин будет предоставлять функционал не только пользователям, но и персоналу, для это будут созданы панели управления с нужным функционалом для каждой роли, которая может и должна контактировать с интернет-магазинами.

В этих панелях управления у персонала будет функционал:

- управление товарами;
- управление аккаунтами;
- управление заказами;
- управление рецептами;
- просмотр статистики заказов;
- возможность формирования таблицы аккаунтов;
- наличие обратной связи.

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

1 Предпроектное исследование

1.1 Описание предметной области

Интернет-магазин сладостей – это удобный способ покупки нужных товаров, в данном случае необходимые сладости, которые можно купить в интернете в любое удобное время, а также это удобное место для управления и пользования функционалом для персонала.

Основным видом деятельности является продажа сладостей для пользователей и предоставление удобного пользования функционалом для персонала. Пользователь может выбрать тот продукт, который хочет купить, может выбрать доставку, добавить товар в корзину, заполнить свой профиль для ускорения и упрощения повторных заказов, добавить товар в избранное. Это позволит увеличить скорость обслуживания покупателей, повысит надёжность работы с информацией. Интернет-магазин «Сладости» позволит повысить производительность и качество обслуживания, и не заставит покупателя долго искать свои сладости.

В данном магазине будет множество разновидностей сладостей, таких как торты, пирожные, мороженное и конфеты.

Процесс покупки сладостей в магазине можно представить так:

- изначально пользователь попадает на сайт и у него есть возможность авторизоваться или зарегистрироваться на сайте для дальнейшего упрощения повторных заказов на сайте, однако это не обязательная процедура, чтобы не отпугивать покупателей лишней, по их мнению, регистрацией;
- затем пользователь идёт в каталог и ищет нужные ему сладости;
- после нахождения нужных ему сладостей пользователь отправляет товар в корзину для дальнейшей оплаты;
- далее пользователь должен пройти в корзину, чтобы указать имя и номер телефона;

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

– после ввода информации, нужно выбрать способ доставки, если был выбран «самовывоз», то покупатель едет в магазин, чтобы самому забрать заказ. Если же способ доставки был «доставка на дом», то просто ожидать доставки курьером.

У заказа будет несколько статусов, которые отображают на каком этапе он сейчас находится. У персонала, который контактирует с заказом, будет возможность менять статус заказа на следующий этап. Таким образом всего будет 7 статусов.

Сперва после оформления заказ будет иметь статус «Новый». Далее модератор должен проверить заказ и если он соответствует требованиям, то подтвердить заказ, тем самым установить статус «Подтвержден». Затем заказ переходит к кондитерам, которые могут принять его, и он получит статус «Готовится», а после подтвердить завершение готовки и установить статус «Готов».

Если был выбран способ доставки самовывоз, то статус «Готов» говорит о том, что заказ готов и ожидает покупателя. Если же способ доставки был «Доставка на дом», то заказ переходит к курьерам, где после принятия заказ получает статус «Доставляется», а после вручения «Завершен», так же заказ получает этот статус при получении заказа покупателем при самовывозе.

После того как заказ будет подтвержден модератором, пользователь уже не сможет отменить заказ вручную, но если заказ был только создан и ещё не подтверждён, то пользователь при необходимости всё же может сам сделать отмену заказа.

Если пользователь был авторизован, то после оформления заказа, он попадает в историю заказов, где может посмотреть текущий заказ и его статус, а также повторить предыдущие заказы.

При повторении предыдущего заказа, товары сами добавляются в корзину, и пользователь отправляется на оформление заказа.

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1.2 Анализ инструментальных средств реализации

Инструменты разработки программного продукта сильно влияют на процесс создания интернет-магазина, они упрощают и ускоряют работу.

Чтобы было удобно проектировать структуру интернет-магазина можно использовать Draw.io, а его дизайн – через онлайн-сервис Figma.

Интернет-магазин будет состоять из двух частей – клиентская и серверная.

Для реализации клиентской части отлично подойдут следующие инструменты: HTML5, CSS3 и JS, включая библиотеку React. Серверная часть будет действовать на Node JS, с использованием базы данных MongoDB.

MongoDB Compass – это графический интерфейс для взаимодействия с системой управления, а также интегрирования, проектирования, моделирования, создания и эксплуатации данных из базы данных MongoDB.

Draw.io – это удобное бесплатное онлайн-приложение для создания диаграмм для рабочих процессов, организационных, сетевых диаграмм, блок-схем, UML и принципиальных электросхем.

Figma – онлайн-сервис для дизайнеров, веб-разработчиков и маркетологов. Он предназначен для создания прототипов сайтов или приложений, иллюстраций и векторной графики.

HTML – язык разметки гипертекста. Язык разметки дает браузеру необходимые инструкции о том, как отображать тексты и другие элементы страницы на мониторе. Язык HTML интерпретируется браузерами и отображается в виде документа, в удобной для человека форме.

CSS – каскадные таблицы стилей, которые используются для определения стилей (правил) оформления документов – включая дизайн, вёрстку и вариации макета для различных устройств и размеров экрана.

JavaScript – это полноценный динамический язык программирования, который применяется к HTML-документу и может обеспечить динамическую интерактивность. JavaScript является объектно-ориентированным языком, но

используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками.

JSX – это расширение синтаксиса JavaScript. Обычно оно используется с React для описания элементов пользовательского интерфейса.

Интернет-магазин будет содержать в себе информацию – её необходимо хранить, изменять, структурировать и использовать. Это реализуется благодаря базе данных. Были рассмотрены следующие варианты реализации СУБД:

- MySQL.
- SQLite.
- MongoDB.

MySQL – свободная реляционная система хранения и управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. На сегодняшний день является самой популярной серверной базы данных (далее – БД), за счёт своей простоты, скорости работы и внушительного функционала.

SQLite – компактная встраиваемая СУБД с исходным кодом. В 2005 году проект получил награду Google-O'Reilly Open Source Awards. SQLite поддерживает динамическое типизирование данных. SQLite напрямую хранит информацию в одном файле, что облегчает его копирование. Большая популярность в мобильной разработке и небольших автономных приложениях, поскольку она занимает меньше места на дисковом пространстве, имеет высокую скорость работы и не требует в отличии от MySQL наличие сервера для запуска. Минусы: ограничения на запись, всего 5 типов данных, отсутствие встроенного механизма аутентификации.

MongoDB – это ориентированная на документы база данных NoSQL с открытым исходным кодом, которая использует для хранения структуру JSON.

Модель данных MongoDB позволяет представлять иерархические отношения, проще хранить массивы и другие более сложные структуры.

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

Для наглядности сравнения вариантов реализации базы данных была составлена таблица 1.

Таблица 1 – Сравнение средств реализации базы данных

Название СУБД	MySQL	SQLite	MongoDB
Большое кол-во типов данных	+	+	+
Масштабируемость	+	-	+
Не требует удаленного сервера	-	+	-
Простота использования	+	+	+
Сложность реализации JSON-структур	-	+	+

Таким образом, в качестве СУБД для будущего продукта была выбрана MongoDB, так как она предоставляет весь необходимый функционал для разработки продукта.

Для взаимосвязи баз данных и северной части продукта необходимо использовать серверный язык. Для реализации этого были рассмотрены языки программирования – Python, Php, JS, а точнее его библиотеки – Express, Mongoose.

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

PHP – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. Действует, как и самостоятельно, так и с фреймворками.

Язык автоматически поддерживает HTTP Cookies в соответствии со стандартами Netscape. Это позволяет проводить установку и чтение небольших

сегментов данных на стороне клиента. Работа с Cookies организована посредством сеансов (сессий). У сессий есть срок действия (после его истечения данные удаляются), в сессиях можно хранить и редактировать разные типы данных, в том числе сериализованные PHP-объекты, пропущенные через serialize (процесс происходит автоматически).

Для наглядности сравнения языков программирования была составлена таблица 2.

Таблица 2 – Сравнение языков программирования для разработки программного продукта

Название языка программирования	Php	Python	Node JS
Наличие библиотек	+	+	+
Инструменты для работы с БД	+	+	+
Объектно-ориентированные возможности	+	+	+
Лёгкий понятный синтаксис	-	+	+
Более активное сообщество	+	+	+
Более простая модульность	-	+	+

Таким образом, NodeJS будет более лучшим вариантом, так как он имеет большое количество библиотек и имеет более активное сообщество.

Для разработки программного продукта рассмотрены следующие инструментальные средства разработки программных продуктов:

- Visual Studio.
- Visual Studio Code.
- Sublime Text 3.

Visual Studio – Линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других

инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и веб-сайты, веб-приложения, веб-службы.

Visual Studio Code – это один из наиболее популярных редакторов кода, разработанный корпорацией Microsoft. Он распространяется в бесплатном доступе и поддерживается всеми актуальными операционными системами.

Sublime Text 3 – это текстовый редактор, разработанный для верстальщиков и программистов.

Сравнение IDE для разработки программного продукта наглядно представлено в таблице 3.

Таблица 3 – Сравнение IDE для разработки программного продукта

Название IDE	VS Code	Visual Studio	Sublime
Общедоступное	+	+	-
Автоматическое сохранение	+	+	+
Подсказки по коду	+	+	+
Интеграция с (GIT)	+	+	-
Множество библиотек	+	+	+
Поддержка расширений	+	-	-

Таким образом, после рассмотрения вариантов средств разработок, было принято решение использовать VS Code. Другие среды разработки тоже оказались неплохими, но VS Code выглядит более подходящим для выполнения всех необходимых для проекта задач.

2 Техническое задание на разработку программного продукта

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602-2020.

Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

1. общие сведения;
2. назначение и цели создания;
3. требования в целом;
 - 3.1. требования к структуре и функционированию;
 - 3.2. требования к надёжности;
 - 3.3. требования к безопасности;
 - 3.4. технические требования;
4. требования к документированию;
5. состав и содержание работ по созданию интернет-магазина.

Техническое задание на разработку интернет-магазина представлено отдельным документом.

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

3 Проектирование

3.1 Архитектура программного продукта

Архитектура программного продукта определяет основную структуру системы, включая компоненты, их взаимосвязи и взаимодействие с внешней средой, а также принципы, определяющие проектирование и развитие системы.

Для данного приложения было выбрано использование клиент-серверной архитектуры (рисунок 1), где основное взаимодействие происходит между клиентом и сервером. Клиент отправляет запросы на получение данных серверу, который осуществляет проверку доступа к информации, обращается к базе данных для получения и обработки нужной информации, а затем передает ее обратно клиенту.

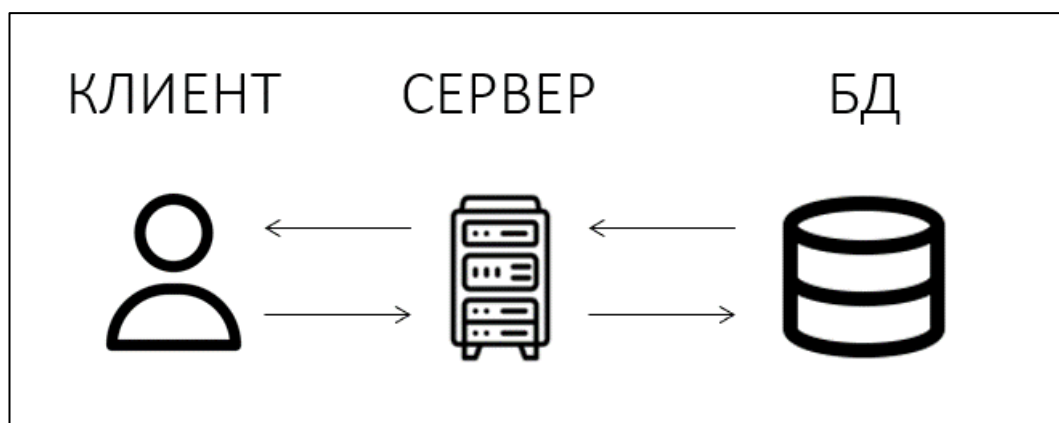


Рисунок 1 – Клиент-серверная архитектура

Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него.

Сервер – специальное системное оборудование, которое предназначается для разрешения определенного круга задач по процессу выполнения программных кодов. Он выполняет работы сервисного обслуживания по клиентским запросам, предоставляет пользователям доступ к определенным системным ресурсам, сохраняет данные или БД.

Параметры, которые могут реализоваться на стороне сервера:

- хранение и доступ к данным;
- работа с поступающими клиентскими запросами;
- процесс отправки ответа клиенту.

Клиент – локальный компьютер на стороне виртуального пользователя, который выполняет отправку запроса к серверу для возможности предоставления данных или выполнения определенной группы системных действий.

Параметры, которые могут реализоваться на стороне клиента:

- формулировка запроса к серверу и его последующая отправка;
- получение итогов запроса и отправка дополнительной группы команд (запросы на добавление, обновление информации, удаление группы данных).

Таким образом, в архитектуре «клиент-сервер» клиент посылает запрос на предоставление данных и получает только те данные, которые действительно были затребованы.

3.2 Функциональное проектирование

Одним из важнейших этапов разработки интернет-магазина является проектирование диаграмм, которые играют ключевую роль в понимании и анализе структуры и работоспособности системы.

Диаграммы служат визуальным инструментом для ясного и точного отображения структуры и функционала интернет-магазина.

Диаграмма вариантов использования, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать интернет-магазин на концептуальном уровне.

Прецедент – возможность моделируемой системы, благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.

На диаграммах UML для связывания элементов используются различные соединительные линии, которые называются отношениями. Каждое такое

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

отношение имеет собственное название и используется для достижения определённой цели.

На диаграмме прецедентов варианты использования представляют собой эллипсы, которые отображают функциональные возможности системы или ее части. Они описывают конкретные задачи, которые система должна выполнять для достижения цели, и представляют собой сценарии использования системы с точки зрения ее пользователей.

На рисунке 2 изображена Use Case View, которая показывает структурную схему интернет-магазина. Кроме этого, она отображает действия и возможности пользователей приложения. Актёрами являются: «Администратор», «Зарегистрированный пользователь», «Гость», «Курьер».

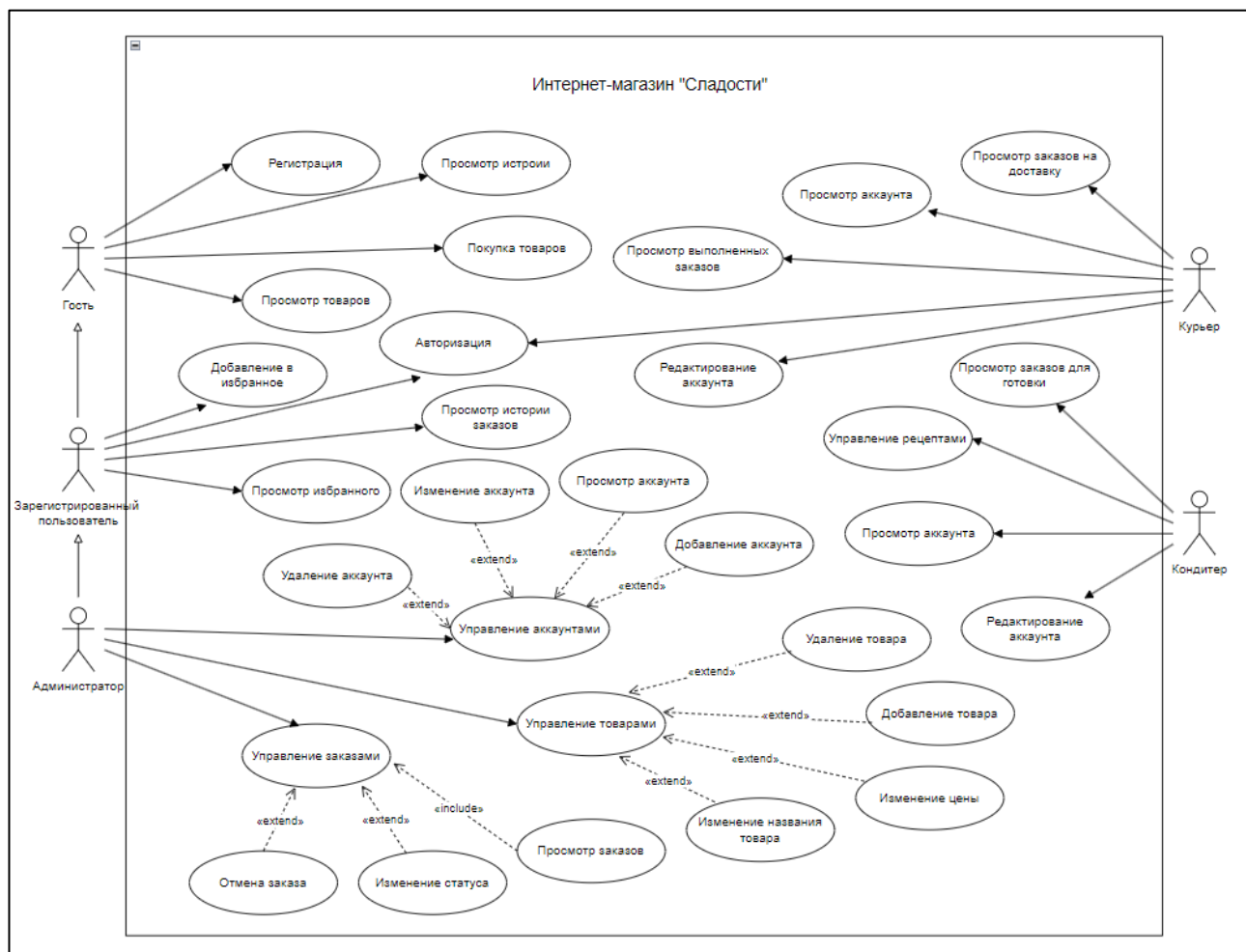


Рисунок 2 – Диаграмма прецедентов

Диаграмма деятельности – это UML-диаграмма.

Она показывает действия, состояния которых описаны на диаграмме состояний.

Деятельность означает спецификацию поведения в виде упорядоченной последовательности в выполнении действий, которые бывают последовательные и параллельные.

Действия выполняются элементами. Элементы – это определённые процессы, связываются между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграмма деятельности используется при моделировании технологических и бизнес-процессов, разных вычислений, её можно увидеть на рисунке 3.

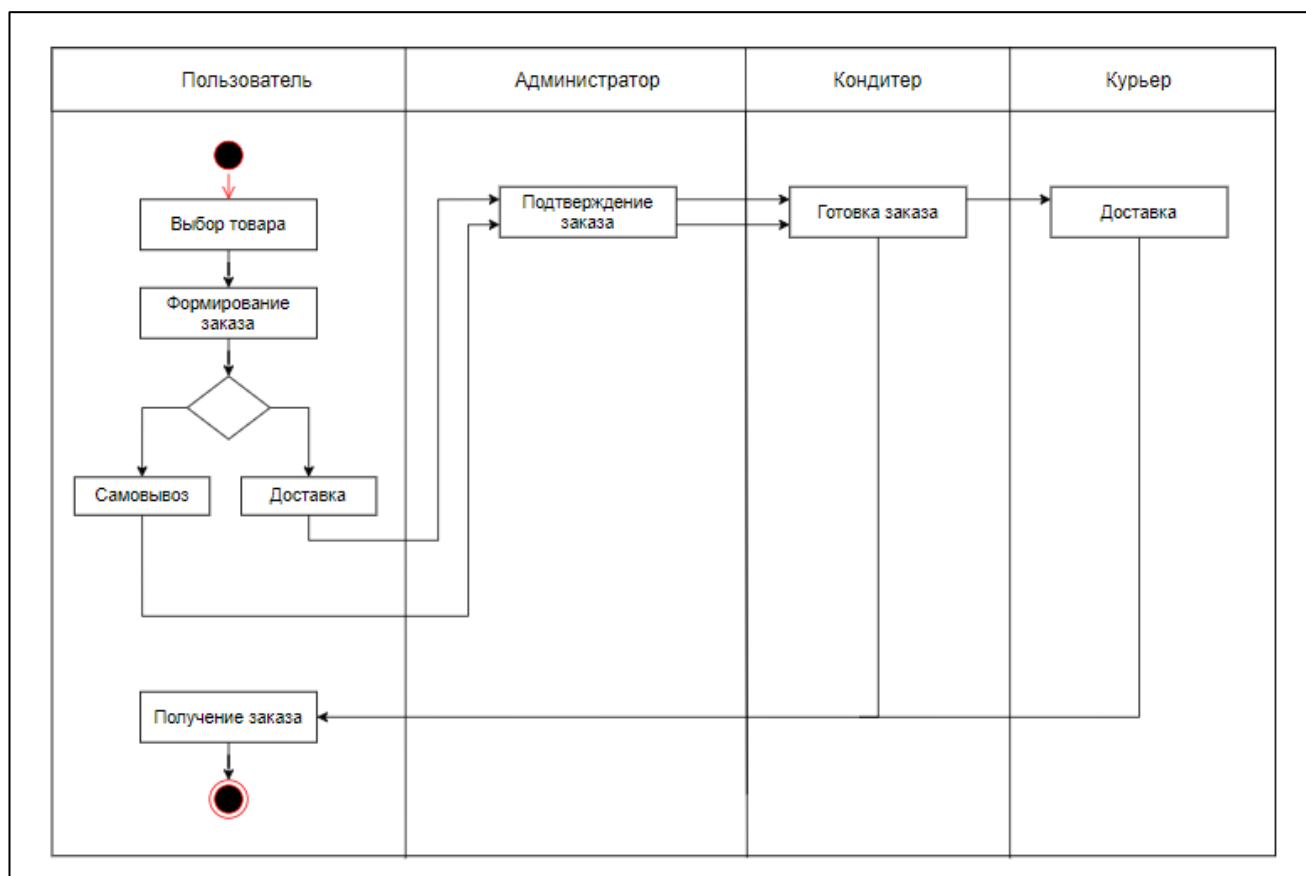


Рисунок 3 – Диаграмма деятельности

В итоге проектирования диаграммы деятельности были выделены основные возможные действия пользователя с программным продуктом.

На рисунке 4 находится контекстная диаграмма созданной в нотации IDEF0, она используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции.

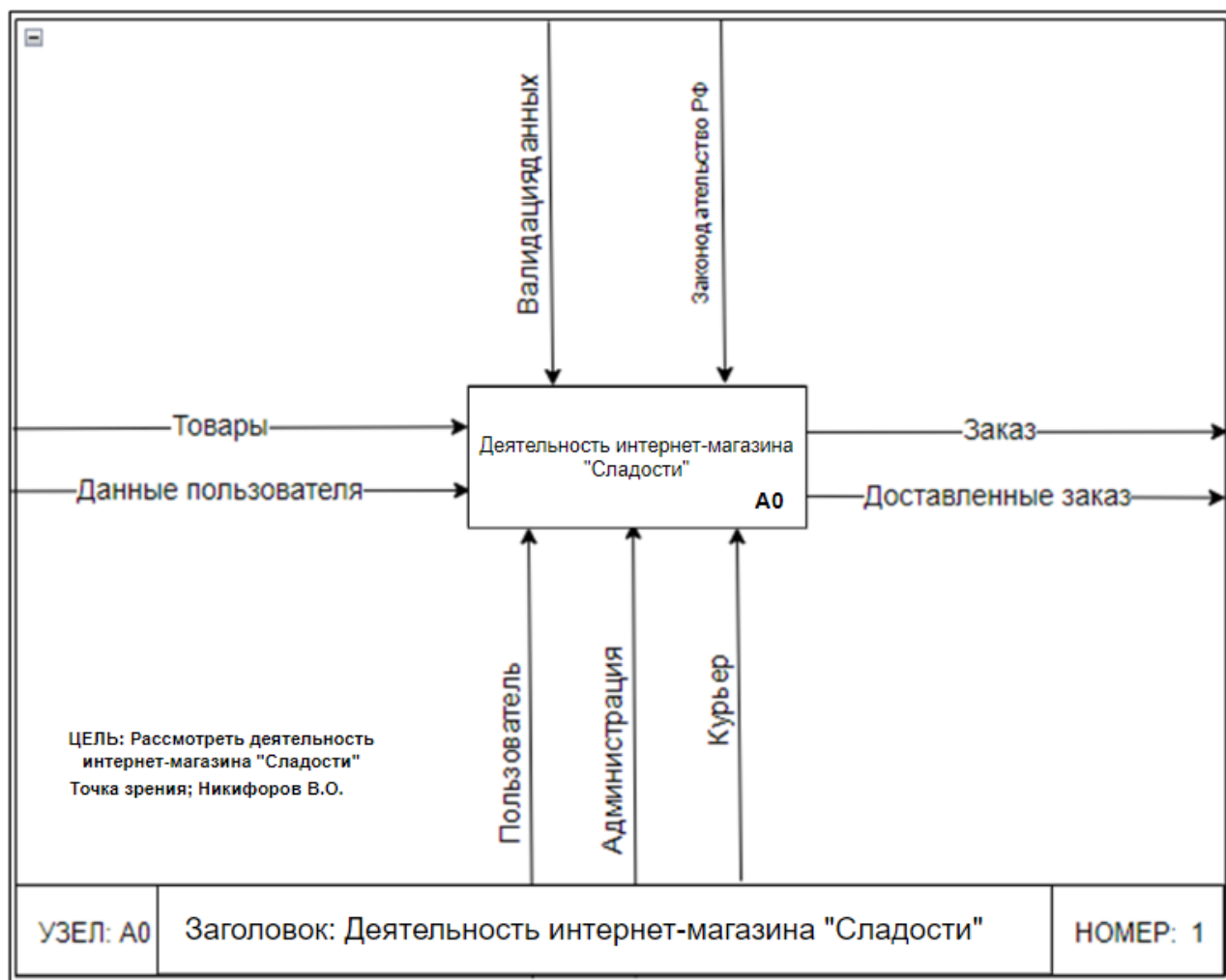


Рисунок 4 – Контекстная диаграмма

В результате проектирования контекстной диаграммы можно увидеть функциональную модель оформления заказа интернет-магазина.

Далее на рисунке 5 можно увидеть диаграмму декомпозиции узла A1, на которой можно увидеть разделенные бизнес-процессы на более мелкие составляющие.

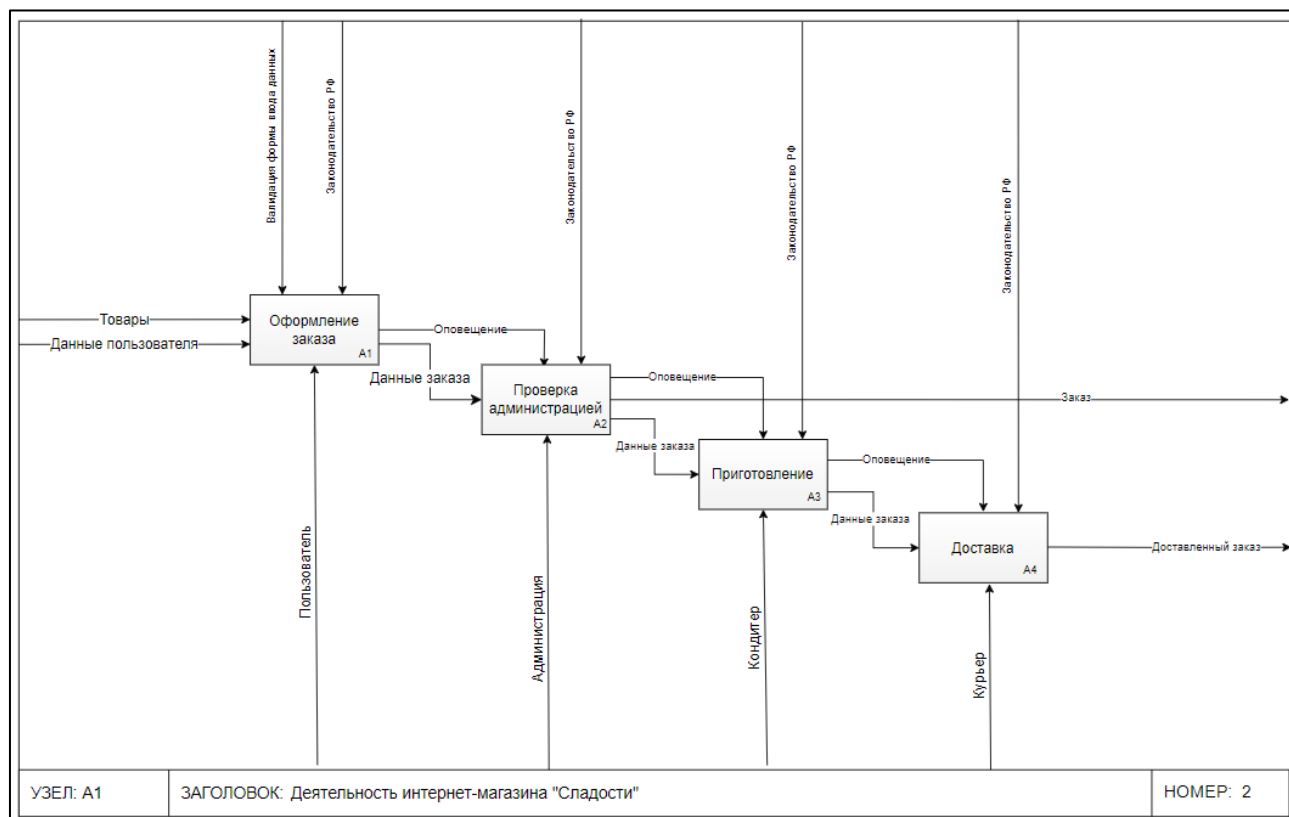


Рисунок 5 – Диаграмма декомпозиций узла A1

В итоге проектирования диаграммы декомпозиции можно увидеть более подробную функциональную модель оформления заказа интернет-магазина «Сладости».

DFD – это диаграмма потоков данных, которая изображена на рисунке 6.

Так же это метод, с помощью которого проводится графический структурный анализ, в котором описаны внешние для системы источники данных, функции, потоки и хранилища данных, к которым имеется доступ.

При проектировании диаграммы потоков данных DFD стало наглядно видно, как будет происходить обработка и передача данных.

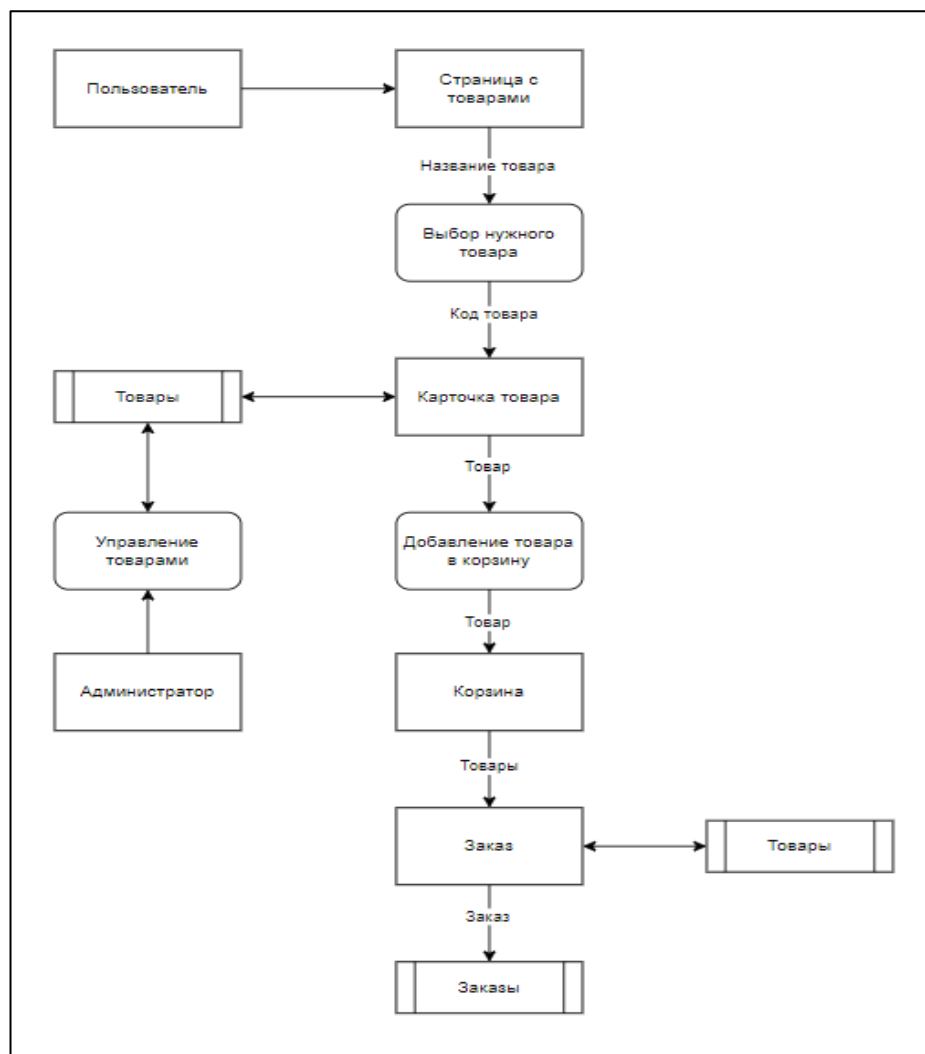


Рисунок 6 – Диаграмма потоков данных DFD

3.3 Проектирование базы данных

Прежде чем приступить разработке программного обеспечения необходимо спроектировать базу данных, а именно, определить с какими данными будут работать участники системы, и чем данные связаны между собой. В этом заключается процесс проектирования.

MongoDB - это документоориентированная NoSQL база данных, которая отличается от классических реляционных баз данных, таких как SQL. Вместо использования связей и таблиц, MongoDB хранит данные в гибких документах формата JSON, что позволяет гибко изменять структуру документов

без необходимости миграции схемы, а следовательно, в MongoDB отсутствует жесткое определение схемы данных. На рисунке 7 представлена ER-модель.

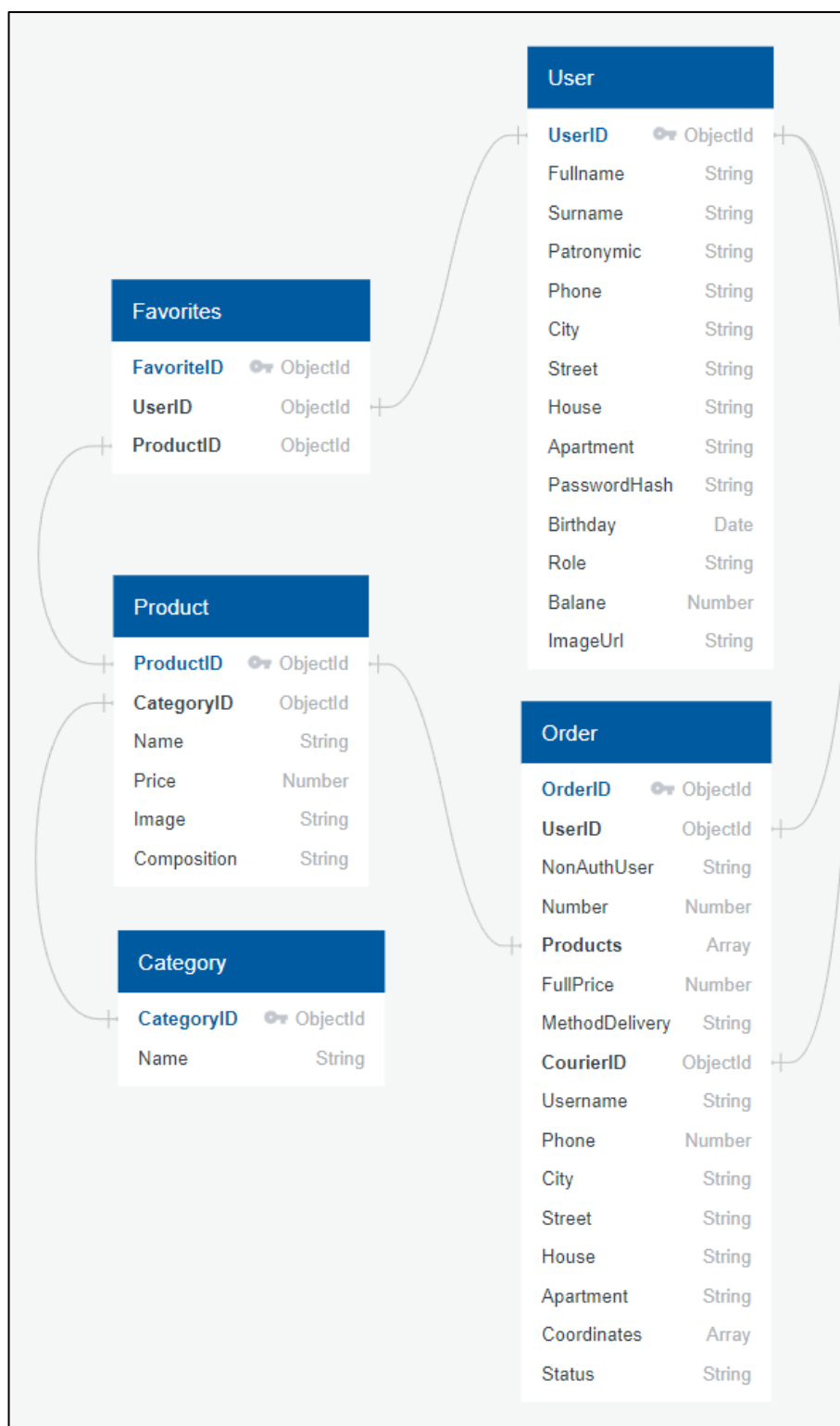


Рисунок 7 – ER – модель базы данных

Всего она содержит 5 таблицы:

- 1) Таблица «User», которая хранит данные о пользователе, такие как:
 - UserID (идентификатор).
 - Fullname (имя пользователя).
 - Surname (фамилия пользователя).
 - Patronymic (отчество пользователя).
 - Phone (номер телефона пользователя).
 - City (город проживания пользователя).
 - Street (улица проживания пользователя).
 - House (дом проживания пользователя).
 - Apartment (квартира проживания пользователя).
 - PasswordHash (зашифрованный пароль пользователя).
 - Birthday (дата рождения пользователя).
 - Role (роль пользователя).
 - Balance (баланс пользователя).
 - ImageUrl (адрес фотографии пользователя).
- 2) Таблица «Product», которая хранит данных о товаре:
 - ProductID (идентификатор).
 - CategoryID (идентификатор категории).
 - Name (название товара).
 - Price (цена).
 - Image (внешний вид товара).
 - Composition (Состав).
- 3) Таблица «Order», которая хранит информацию о заказе, такую как:
 - OrderID (идентификатор).
 - UserID (идентификатор пользователя).
 - NonAuthUser (идентификатор неавторизованного пользователя).
 - Number (номер заказа).

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

- Products (список товаров).
 - FullPrice (итоговая цена).
 - MethodDelivery (способ доставки).
 - CourierID (идентификатор курьера).
 - Username (имя заказчика).
 - Phone (номер телефона заказчика).
 - City (город для доставки).
 - Street (улица для доставки).
 - House (дом для доставки).
 - Apartment (квартира для доставки).
 - Coordinates (координаты для доставки).
 - Status (статус заказа).
- 4) Таблица «Favorites», которая хранит информацию о заказе, такую как:
- FavoriteID (идентификатор категории).
 - UserID (идентификатор пользователя).
 - ProductID (идентификатор товара).
- 5) Таблица «Category», которая хранит информацию о заказе, такую как:
- CategoryID (идентификатор категории).
 - Name (название категории).

Таким образом, можно увидеть всю необходимую информацию для понимания системы хранения данных.

3.4 Проектирование пользовательского интерфейса

Пользовательский интерфейс – это средства взаимодействия между человеком и компьютером. Говоря простыми словами, интерфейс – внешняя часть программы или устройства, с которыми работает пользователь.

Интерфейсы являются основой взаимодействия всех современных веб-приложений. Если интерфейс какого-либо объекта не изменяется (стабилен,

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

стандартизирован), это даёт возможность модифицировать сам объект, не перестраивая принципы его взаимодействия с другими объектами.

Данный прототип интерфейса был построен в онлайн конструкторе figma.com.

На рисунке 8 отображена страница с каталогом, на которой можно выбрать, нужную для нас категорию сладостей.



Рисунок 8 – Каталог

При выборе категории «Торты», пользователь попадает на страницу на рисунке 9.

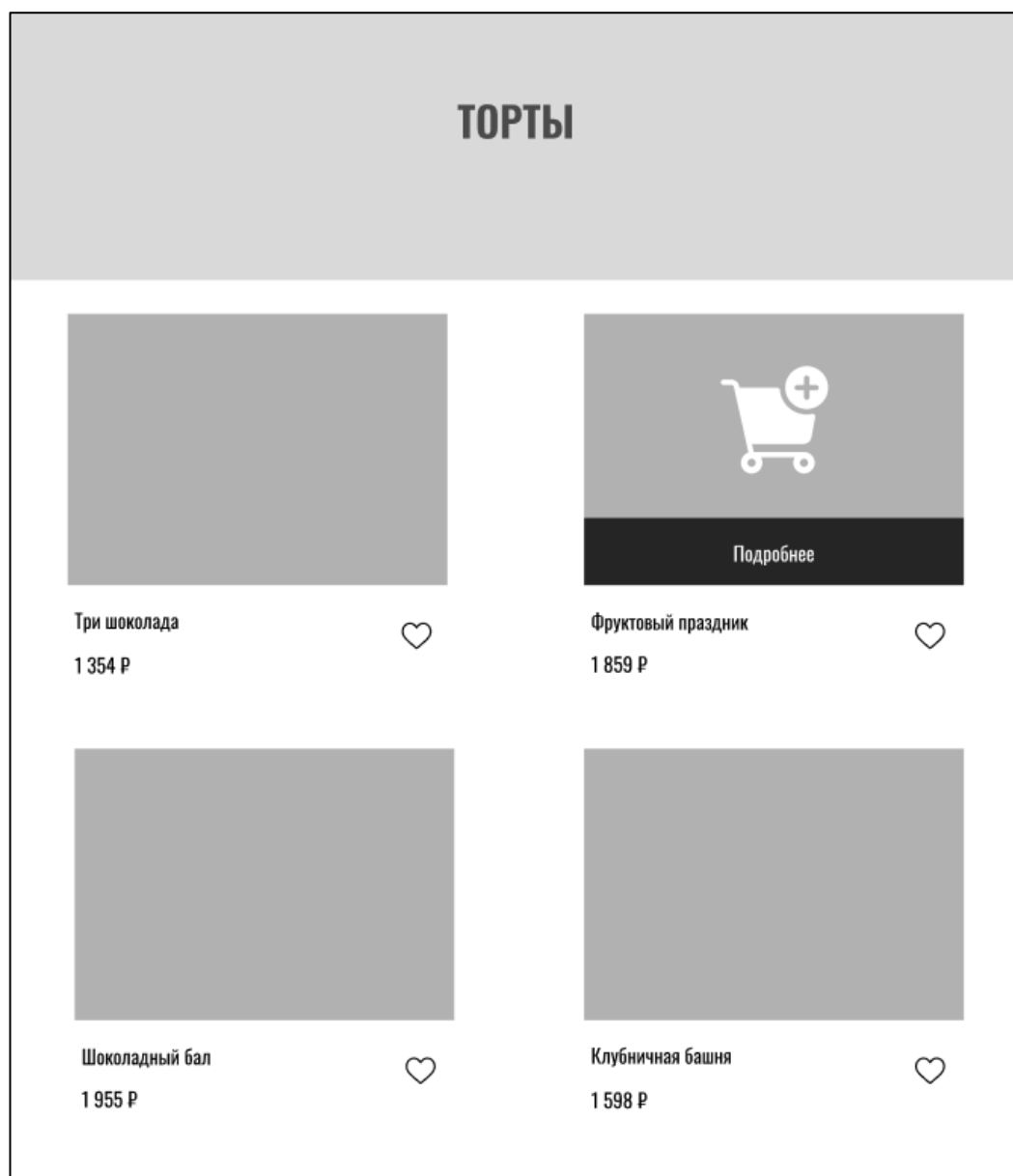


Рисунок 9 – Страница «Торты»

При выборе другой категории, пользователь попадает на похожую страницу только с другими товарами и шапкой сайта.

При наведении на товар в любой категории, на товаре будет появляться кнопка добавления товара в корзину, а после можно нажать на кнопку корзины, и перейти для оплаты на страницу корзины, она отображена на рисунке 10.

На любой странице можно авторизоваться или выйти из аккаунта. Саму авторизации можно увидеть на рисунке 11.

Ваш заказ

Обереон

Состав: сахар, орехи

-

1

+

550 Р

×

Три шоколада

Состав: сахар, клубника, тесто

-

1

+

1 354 Р

×

Шоколадный кекс

Состав: сахар, лист, шоколад

-

1

+

1 859 Р

×

Итого 3 763

Оформление заказа

Способ доставки

Доставка

Самовывоз

Рисунок 10 – Страница корзины

×

Авторизация

Введите номер телефона

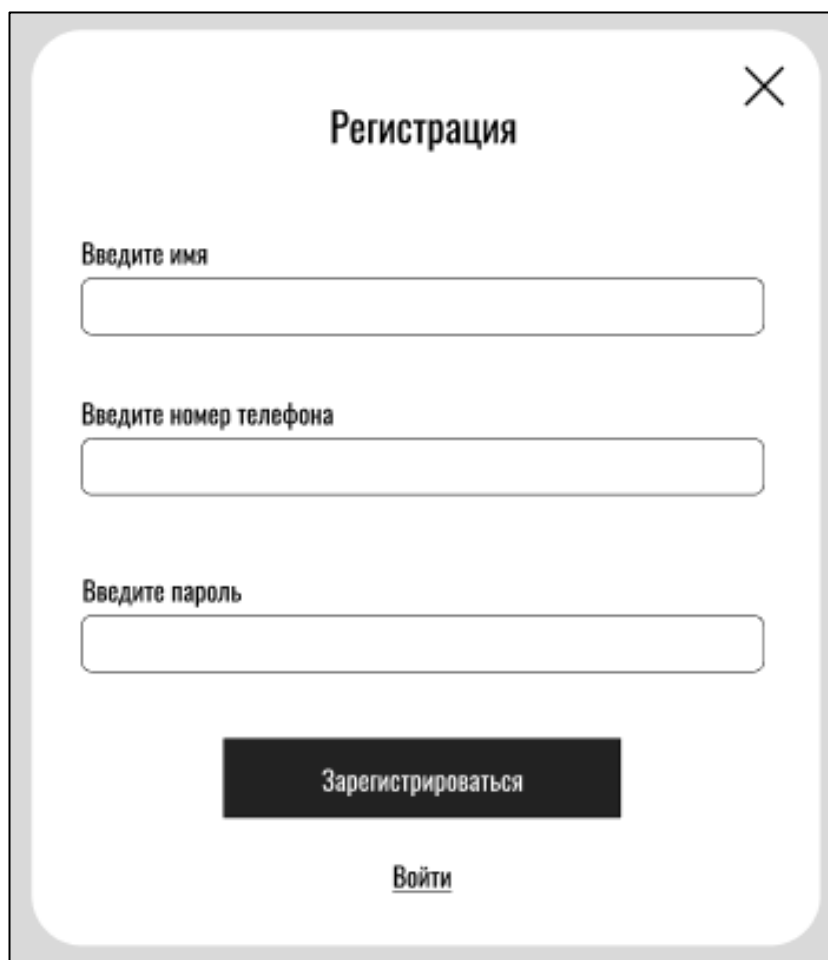
Введите пароль

Войти

Зарегистрироваться

Рисунок 11 – Модальное окно авторизации

Но перед тем, как авторизоваться, нужно сначала зарегистрироваться, а для этого нужно через окно авторизации, открыть регистрацию, её можно увидеть на рисунке 12.



Регистрация

Введите имя

Введите номер телефона

Введите пароль

Зарегистрироваться

[Войти](#)

Рисунок 12 – Модальное окно регистрации

Авторизация даёт возможность получить токен авторизации и полезную информацию о пользователе, которая будет использоваться для отображения на странице и удобного оформления заказа.

4 Реализация программного продукта

В разработке клиентской части приложения будет использоваться JavaScript-библиотека React. На данный момент он является одним из лидирующих библиотек для разработки клиентских веб-приложение.

С помощью React можно использовать различные страницы как компоненты, чтобы при необходимости, не изменяя основную структуру страницы изменять содержимое страницы.

При входе на страницу пользователь запускает основной скрипт React библиотеки, который, через систему маршрутизации, изменяет содержимое корневой страницы на определенный контент.

Корневая страница представляет собой обычный html-документ с корневым элементом, который обычно задаётся через идентификатор root, пример корневой страницы можно увидеть на рисунке 13.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="https://cdn-icons-png.flaticon.com/512/2682/2682446.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Candy Store</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Рисунок 13 – Корневая страница

Точкой входа (рисунок 14) является index.js, который получает корневой элемент из корневой страницы и записывает в него контент, полученный из React-компонентов, определенных в компоненте маршрутов (рисунок 15), который описывает, какие компоненты должны быть загружены для каждого пути URL.

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import {BrowserRouter} from 'react-router-dom';
import {ContextProvider} from './context.js';
import App from './app.js';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <ContextProvider>
      <App/>
    </ContextProvider>
  </BrowserRouter>
);

```

Рисунок 14 – Точка входа

```

<Routes>
  <Route path="/" element = {<Catalog/>}/>
  <Route path="/cakes" element = {<Cakes/>}/>
  <Route path="/candies" element = {<Candies/>}/>
  <Route path="/cart" element = {<Cart/>}/>
  <Route path="/history/" element = {<History/>}/>
  <Route path="/ice-cream" element = {<Ice_cream/>}/>
  <Route path="/desserts" element = {<Desserts/>}/>
  <Route path="/favorites" element = {<Favorites/>}/>
  <Route path="*" element = {<ER404/>}/>
  <Route path="/admin" element = {<Admin/>}/>
  <Route path="/courier" element = {<Courier/>}/>
</Routes>

```

Рисунок 15 – Компонент маршрутов

Разработка базы данных интернет-магазина реализовывалась в СУБД MongoDB. База данных состоит из 5 коллекций.

Структуры коллекций соответствуют схеме базы данных из пункта 4.3.

На рисунке 16 представлен один из документов коллекции «Пользователь», в нем хранится личная информация, роль и дата создания пользователя.

```

_id: ObjectId('65541e72ba7bf9a7e76b3ec5')
fullName: "Дмитрий"
surname: "Лосев"
patronymic: "Адреевич"
phone: "88005553533"
city: "Иркутск"
street: "Ленина"
house: "78"
apartment: "202"
passwordHash: "$2b$10$PmrR2mexXcoPIx0NAs1sFOU7ji0qd8JBoaGdmlHwFm4Gx18VFGJp0"
birthday: 1981-09-05T00:00:00.000+00:00
role: "courier"
balance: 0
imageUrl: "/uploads/portrait-of-smiley-businessman.jpg"
createdAt: 2023-11-15T01:27:14.340+00:00
updatedAt: 2023-11-17T02:15:12.718+00:00
__v: 0

```

Рисунок 16 – Документ коллекции «Пользователи»

На рисунке 17 представлен один из документов коллекции «Товары», в котором используется идентификатор категории товара и его описание.

```

_id: ObjectId('6377987a6359d93c2c7e31ca')
name: "Клубничная башня"
price: "1598"
category: ObjectId('637871432dc9c0dfd59e467d')
imageUrl: "/uploads/town.png"
composition: "Пшеничная мука, яйца, сахар, сливочное масло, молоко, соль"
createdAt: 2022-11-18T14:36:42.642+00:00
updatedAt: 2023-05-18T11:06:47.795+00:00
__v: 0

```

Рисунок 17 – Документ коллекции «Товары»

При оформлении заказа он сохраняется в коллекции «Заказы», и содержит в себе список товаров, пользователя, способ доставки, статус заказа, по которому можно понять на каком этапе находится заказ, а также координаты доставки, чтобы показать точное место доставки для курьера, после принятия заказа курьером, в документ так же сохраняется идентификатор курьера. Пример документа коллекции «Заказы» можно увидеть на рисунке 18.

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						29
Изм.	Лист	№ докум.	Подпись	Дата		

```

_id: ObjectId('65561db5c293a4f36719ed7f')
number: 17
user: ObjectId('6388b33923a5034399706a77')
▼ products: Array (2)
  ▼ 0: Object
    value: 1
    product: ObjectId('6377987a6359d93c2c7e31ca')
  ▼ 1: Object
    value: 1
    product: ObjectId('637871922dc9c0dfd59e468a')
fullPrice: 3548
methodDelivery: "delivery"
username: "Василий"
phone: 88005553535
city: "Иркутск"
street: "улица Терешковой"
house: "40"
apartment: "13"
▼ coordinates: Array (2)
  0: 52.276338
  1: 104.245141
status: "ended"
createdAt: 2023-11-16T13:48:37.730+00:00
updatedAt: 2023-11-16T14:09:29.144+00:00
__v: 0
courier: ObjectId('65541e72ba7bf9a7e76b3ec5')

```

Рисунок 18 – Документ коллекции «Заказы»

В интернет-магазине есть функция добавления товара в избранное, для того чтобы это сделать создается документ коллекции «Избранное» с идентификатором пользователя и понравившегося товара, как на рисунке 19.

```

_id: ObjectId('644f9fd9c07965da071ef41b')
user: ObjectId('6381c588ca472d5aa815fb7f')
product: ObjectId('6377987a6359d93c2c7e31ca')
createdAt: 2023-05-01T11:17:45.393+00:00
updatedAt: 2023-05-01T11:17:45.393+00:00
__v: 0

```

Рисунок 19 – Документ коллекции «Избранное»

Для адаптивности к изменениям названия категории товаров была создана коллекция «Категории», которая содержит название и сам идентификатор, по которому будут обращаться документы коллекции товаров. Пример документа коллекции «Категории» изображен на рисунке 20.

```
_id: ObjectId('637871432dc9c0dfd59e467d')
name: "cakes"
createdAt: 2022-11-19T06:01:39.632+00:00
updatedAt: 2022-11-19T06:01:39.632+00:00
__v: 0
```

Рисунок 20 – Объект таблицы «Категории»

В соответствии с заданием, в данном интернет-магазине выполнены все необходимые функции.

Разработка функционала интернет-магазина была реализована на Node.js

Сначала запускается файл index.js, в котором находится основная логика и подключения сервера, это можно подробно увидеть в приложение А.

Первым делом подключаем базу данных через mongoose, а её адрес для подключения, мы достаем для безопасности из локального окружения, это можно увидеть на рисунке 21.

В дальнейшем мы будем тестировать наш сервер, и чтобы это было безопасно для нашей базы данных, нам придется создать другую пустую базу данных, и при тестировании использовать чистую базу данных.

```
mongoose
.connect(process.env.DB)
.then(() => console.log('DB OK'))
.catch((err) => console.log('DB ERROR', err));
```

Рисунок 21 – Подключение базы данных

В этот же файл подключаются маршруты из специального файла с маршрутами, их пример можно увидеть на рисунке 22, а подключение можно увидеть на рисунке 23.

```
app.post('/auth/login', loginValidation, handleValidationErrors, UserController.login);
app.post('/auth/register', registerValidation, handleValidationErrors, UserController.register);
app.get('/auth/me', checkAuth, UserController.getMe);
```

Рисунок 22 – Маршруты

```
import routes from './routes.js';
app.use('/', routes);
```

Рисунок 23 – Подключение маршрутов

Так же подключатся файлы для работы с файлами и сокетом. Работу с файлами можно увидеть в приложение Б, а одно из событий сокета на рисунке 24

```
io.on('connection', (socket)=>{
  socket.on('order', (action, obj) =>{
    socket.broadcast.emit('update-orders-list', action, obj)
    socket.disconnect();
  })
  socket.on('start-change-status', (id, prevStatus, status, room) =>{
    socket.to(room).emit('change-status', id, status)
    socket.to('admin').emit('change-status', prevStatus, status)
    socket.to('courier').emit('change-status', prevStatus, status)
  })
})
```

Рисунок 24 – Событие в сокете

Для понимания принципа работы сервера интернет-магазина, разберем любое действие пользователя, например создание заказа.

Когда пользователь собрал все нужные ему товары в корзину, он приступает к оформлению заказа, сперва он вводит личную информацию, такую как имя, телефон, адрес, способ доставки. Если способом доставки был выбран самовывоз,

то заказ просто отправляется администратору на проверку и после подтверждения заказ будет ожидать пользователя в кондитерской.

Но если пользователь выбрал способом доставки «доставка на дом», то он должен будет ввести ещё адрес доставки и после того как пользователь нажмет «заказать» сперва произойдёт отправка адреса доставки на API карт для геокодирования, это можно увидеть на рисунке 25, а после результат с координатами и остальной информацией о заказе отправляются на сервер, где этот запрос будет ожидать маршрут создания заказа (рисунок 26).

```
const address = `${city}, ${street}, ${house}`
await fetch(`https://geocode-maps.yandex.ru/1.x/?apikey=${process.env.REACT_APP_MAP_API}&format=json&geocode=${address}`)
.then(response => response.json())
.then(data => {
  // Получение координат из ответа
  const coordinatesData = data.response.GeoObjectCollection.featureMember[0].GeoObject.Point.pos.split(' ');
  const addressData = data.response.GeoObjectCollection.featureMember[0].GeoObject.metaDataProperty.GeocoderMetaData.Address.Components

  // Заполнение адреса, найденными данными адреса
  addressData.map((obj) =>{
    Address[obj.kind] = obj.name
  })
  coordinates = [Number(coordinatesData[1]), Number(coordinatesData[0])]
})
.catch(error => {
  console.error('Ошибка при геокодировании:', error);
});
```

Рисунок 25 – Запрос на геокодирование

```
app.post('/orders', orderCreateValidation, handleValidationErrors, OrderController.create);
```

Рисунок 26 – Маршрут создания заказа

После того как маршрут получит запрос он передает его на проверку валидации (рисунок 27), а после контроллеру для создания заказа (приложение В).

```
export const orderCreateValidation = [
  body('products', 'Выберите товары'),
  body('methodDelivery', 'Выберите способ доставки').isLength({min: 2}),
  body('username', 'Укажите имя заказчика').isLength({min: 1}),
  body('phone', 'Укажите номер, не менее 11 символов').isLength({min: 11, max: 11}),
];
```

Рисунок 27 – Валидация запроса на создание нового заказа

В начале функции определится номер нового заказа на основе номеров заказов. Далее создается новая модель заказа, в которой будет храниться полученная информация из запроса и всё готово – заказ создан.

Однако в момент отправки запроса сервер, клиент так же отправил событие для сокета, которое можно увидеть на рисунке 28.

```
const socket = io(process.env.REACT_APP_API_HOST)
socket.emit('order', 'add', res.data)
```

Рисунок 28 – Отправка события от клиента

После того как сокет получил событие о создании заказа, он отправляет другое событие (рисунок 29), но уже для администраторов, для отображения нового заказе без перезагрузки страницы.

```
socket.on('order', (action, obj) =>{
    socket.broadcast.emit('update-orders-list', action, obj)
    socket.disconnect();
})
```

Рисунок 29 – Отправка события от сокета

Далее администратор проверяет заказ и подтверждает его. После того как заказ был подтвержден, через сокет курьерам приходит новый заказ, и они могут принять его для доставки. Курьер доставляет заказ, подтверждает это и заказ готов.

5 Отладка и тестирование программного продукта.

Одним из этапов разработки любого программного продукта является тестирование и отладка.

Тестирование – это процесс проверки функциональности и корректности программного продукта путем выполнения тестовых сценариев.

Главная цель тестирования программного продукта – обнаружить ошибки и проверить его соответствие требованиям.

Данный программный продукт тестировался методом модульного тестирования с использованием Mocha и Chai.

Модульное тестирование предполагает тестирование отдельных модулей программы, в данном случае, функций и компонентов пользователя.

Отладка – это процесс идентификации и исправления ошибок в программном коде.

Отладка проводилась в процессе написания программного кода, путем поиска и устранения ошибок.

Для тестирования интернет-магазина был разработан сценарий тестирования для роли пользователя предметной области.

В таблице 4 представлен сценарий тестирования для роли курьер.

Далее рассмотрим сами процессы и действия при тестировании интернет-магазина.

Сперва перед самым написанием тестов нужно было подготовить среду, для этого необходимо было создать дополнительную базу данных, чтобы при тестировании никак не повлиять на основную и запускать сервер на тестовой базе данных во время проведения тестирования.

Далее была создана специальная папка для хранения в ней тестов, а после и сами тесты.

Таблица 4 – Сценарий тестирования для роли курьер

Поле	Описание
Дата теста	03.05.2024
Приоритет тестирования	Высокий
Название теста	Создание курьера
Этапы теста	1. Вход в систему 2. Ввод данных курьера 3. Сохранение нового курьера
Тестовые данные	Введенные данные нового курьера и токен администратора
Ожидаемый результат	Система создает нового курьера. Если введенные данные не соответствуют требованиям или токен администратора не прошел аутентификацию, то курьер не будет создан и вернется ошибка.
Фактический результат	Система создала нового курьера.

Первый файл был создан для того, чтобы перед и после каждого блока тестов очищать коллекции и их индексы в базе данных, это нужно для того, чтобы каждый тест был проведен в конкретных условиях (рисунок 30).

```
process.env.NODE_ENV = 'test';

import User from '../models/user.js';

// Очистить бд перед и после теста
before((done) => {
  User.deleteMany({}, function(err) {
    // Очистить индексы после удаления документов
    User.collection.dropIndexes(function(err, result) {});
  });
  done();
});

after((done) => {
  User.deleteMany({}, function(err) {
    User.collection.dropIndexes(function(err, result) {});
  });
  done();
});
```

Рисунок 30 – Очистка коллекции

Далее рассмотрим файл с тестами функций авторизации (приложение Г).

В начале создаются переменные для хранения токена и шаблона пользователя. Затем создается хук который будет постоянно срабатывать перед каждым тестом и проверять был ли проведен тест регистрации пользователя или нет (рисунок 31), если теста не было, тогда сделать тест регистрации, а если тест уже был произведен, тогда будет проверен токен, и если его нету, то авторизоваться. Это нужно для того, чтобы каждый следующий тест был от авторизованного пользователя.

```
beforeEach((done) => {
  if(!userRegistered){
    chai.request(server)
      .post('/auth/register')
      .send(user)
      .end((err, res) => {
        expect(res.status).to.be.equal(200);
        expect(res.body).to.be.a('object');
        expect(res.body.error).to.be.equal(null || undefined);
        chai.request(server)
          .post('/auth/login')
          .send({
            "phone": user.phone,
            "password": user.password
          })
          .end((err, res) => {
            expect(res.status).to.be.equal(200);
            expect(res.body).to.be.a('object');
            expect(res.body.error).to.be.equal(null || undefined);
            userRegistered = true;
          });
      });
  }
  if(!token && userRegistered){
    chai.request(server)
      .post('/auth/login')
      .send({
        "phone": user.phone,
        "password": user.password`
      })
      .end((err, res) => {
        expect(res.status).to.be.equal(200);
        expect(res.body).to.be.a('object');
        expect(res.body.error).to.be.equal(null || undefined);
        token = res.body.token;
      });
  }
  done();
});
```

Рисунок 31 – Хук регистрации и авторизации

Далее тестируется, что регистрация уже была выполнена и при повторной регистрации ожидается ошибка, как показано на рисунке 32.

```
it('Ошибка при регистрация существующего аккаунта', (done) => {
  chai.request(server)
    .post('/auth/register')
    .send(user)
    .end((err, res) => {
      expect(res.status).to.be.equal(403);
      expect(res.body).to.be.a('object');
      expect(res.body.msg).to.be.equal('Аккаунт с таким номером телефона уже существует');
      done();
    });
});
```

Рисунок 32 – Тест повторной регистрации

Потом идут ещё два теста, один на авторизацию, а второй тестирует, что запросы для авторизованных пользователей будут проходить успешно (рисунок 33).

```
it('Запрос для авторизованных пользователей', (done) => {
  chai.request(server)
    .get('/auth/me')
    .set('Authorization', `Bearer ${token}`)
    .end((err, res) => {
      expect(res.status).to.be.equal(200);
      expect(res.body).to.be.a('object');
      expect(res.body.error).to.be.equal(null || undefined);
      done();
    });
});
```

Рисунок 33 – Тест защищенного запроса

Кроме того, в рамках тестирования был создан чек-лист для роли пользователя (таблица 5).

Таблица 5 – Чек-лист для роли пользователя

Тест	Входные данные	Ожидаемый результат	Фактический результат	Результат тестирования	Комментарий
Регистрация	Имя, номер телефона, пароль	Пользователь создан	Пользователь создан	Успешно	-
Авторизация	Номер телефона, пароль	Пользователь прошел аутентификацию и был возвращен токен авторизации	Пользователь прошел аутентификацию и был возвращен токен авторизации	Успешно	-
Получение истории заказов	Токен авторизации	Пользователь получает список заказов	Пользователь получает список заказов	Успешно	-

По итогу тестирования можно сказать, что тестирование функциональности в данном программном продукте выполнено успешно согласно разработанному сценарию. Выявленных ошибок нет, функции работают корректно.

6 Руководство пользователя программного продукта

Интернет-магазин «Сладости» имеет простой и интуитивно понятный интерфейс, что позволяет легко понять обычному пользователю, как с ней работать.

При вводе URL адреса интернет-магазина в поисковую строку браузера перед пользователем открывается главная страница, которой является каталог продуктов (рисунок 8).

Далее пользователь выбирает нужную ему категорию и отправляется на страницу с товарами (рисунок 9). На этой странице пользователь может добавить товар в корзину, добавить в избранное, а если хочет узнать о нем подробную информацию, то при наведение появится кнопка подробнее, где при нажатие появится модальное окно (рисунок 34), в котором можно узнать подробную информацию, а также выбрать нужное количество товара.

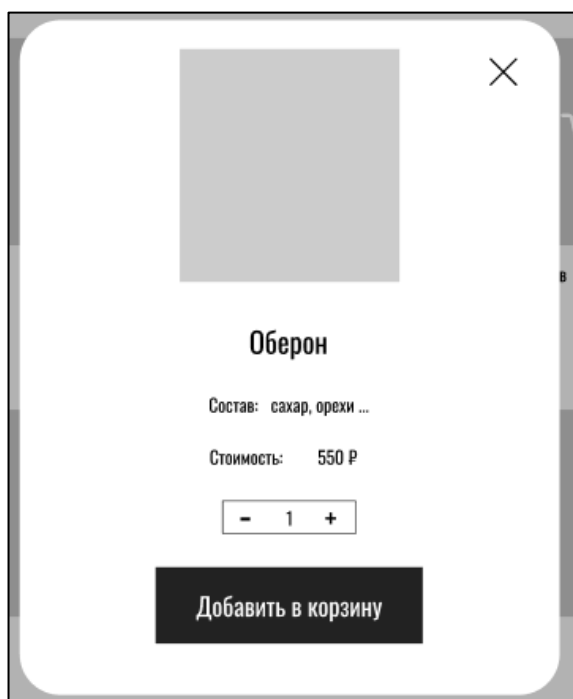


Рисунок 34 – Подробная информация о товаре

После того как все нужные товары были добавлены в корзину, пользователь может нажать на корзину в правом верхнем углу страницы (рисунок 35).



Рисунок 35 – Шапка интернет-магазина

Далее пользователь попадает в корзину (рисунок 10) и может в ней, при необходимости, удалить товар или изменить его количество.

Когда пользователь определился с заказом, пользователь приступает к оформлению заказа (рисунок 36), для этого нужно выбрать способ доставки и заполнить личную информацию, после заказ отправляется администратору.

Способ доставки

Доставка

Самовывоз

Личные данные

Использовать сохраненные данные

Введите имя *

Номер телефона *

Улица *

Дом *

Квартира *

Заказать

Рисунок 36 – Оформление заказа

После подтверждения заказа администратором его можно считать оформленным и остается только приехать забрать его, если способ доставки был самовывоз, или ожидать доставки.

Так же сразу после создания заказа, он будет отображаться на странице истории заказов, где можно будет отслеживать статус заказа, отменить его, но только если заказ ещё не был подтвержден, а также по завершению заказа его можно будет повторить.

6.1 Руководство администратора интернет-магазина

Сначала нужно попасть на панель администратора, для этого нужно зайти или по адресу «/admin» или нажать на кнопку «шестеренку» в шапке страницы.

После попадания на панель администратора у администратора появляется основная навигационная панель, которая располагается слева и при необходимости её можно свернуть (рисунок 37).



Рисунок 37 – Навигационная панель

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

Всего будет 5 разделов, о которых подробно написано снизу.

Первый раздел «Главная», на котором располагается статистика интернет-магазина имеет два блока «Заказы» и «Курьеры». На блоке с заказами отображается вся статистика о заказах, их общее количество и количество по статусам заказов. На втором блоке «Курьеры» отображается статистика о курьерах, общее количество курьеров, курьеры онлайн и сколько курьером сейчас на заказах. Вся статистика обновляется в реальном времени, но, если что-то пойдет не так, всегда можно вручную обновить статистику нажав на кнопку справа сверху любого блока. Раздел «Главная» можно увидеть на рисунке 38.

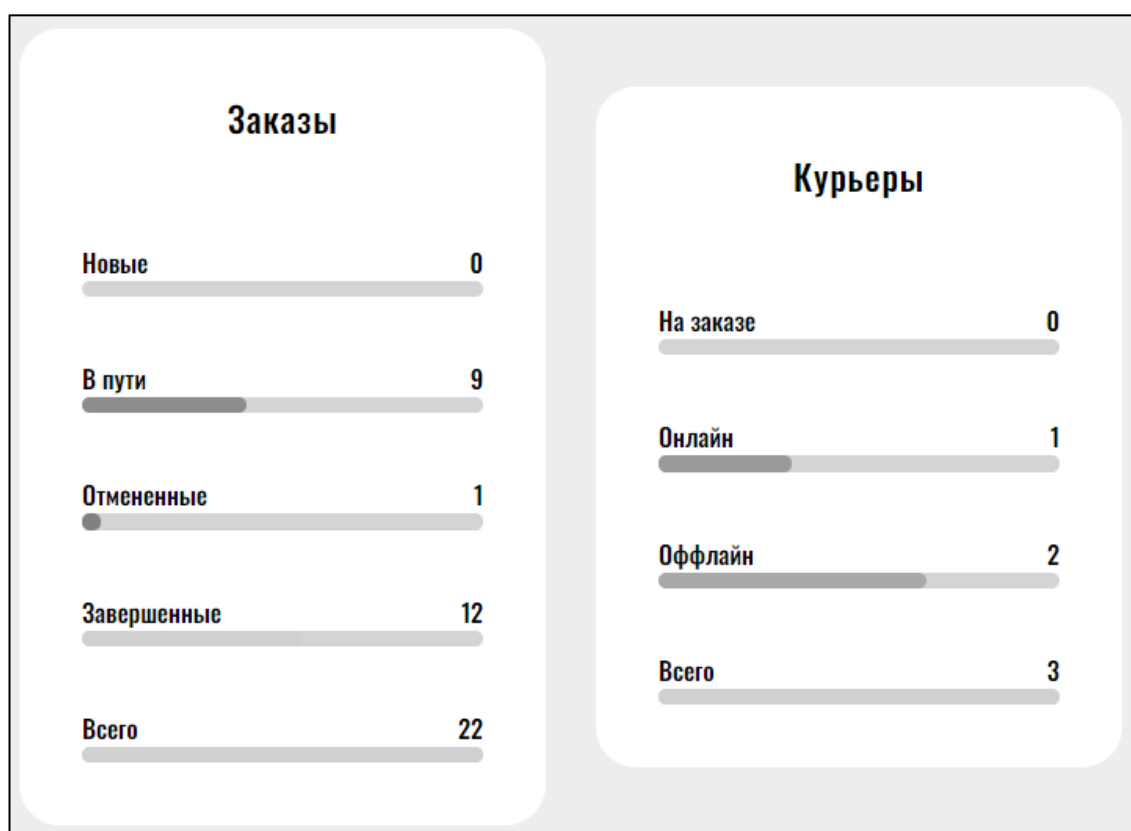


Рисунок 38 – Раздел «Главная»

Второй раздел «Заказ» дает возможность управления заказами (рисунок 39). В верхней части раздела можно увидеть фильтр статуса заказов.

При выборе отображения заказов нужного статуса на страницу выведутся все заказы выбранного статуса и появится три функции:

- Просмотреть подробную информацию о заказе, для этого нужно нажать на заказ.
- Подтвердить заказ, после чего статус заказа переходит на следующий этап, то есть если у заказа был статус «новый», то он получит статус «активный».
- Отменить заказ.

Заказы						
Новые		Активные		Завершенные		
	№ 21	Самовывоз	19:42	<u>1 товар</u>	Итого: 1 598 Р	Подтвердить
						Отменить
	№ 19	Доставка	11:33	<u>3 товара</u>	Итого: 5 498 Р	Подтвердить
						Отменить
	№ 18	Доставка	11:32	<u>4 товара</u>	Итого: 7 096 Р	Подтвердить
						Отменить
	№ 14	Доставка	17:31	<u>2 товара</u>	Итого: 163 348 Р	Подтвердить
						Отменить

Рисунок 39 – Раздел «Заказы»

Третий раздел «Товары» позволяет создавать, редактировать и удалять товары. Сперва будет предложен выбор создать или редактировать (удалить) товар (рисунок 40). При выборе создания товара отобразится форма создания заказа и предварительная карточка товара (рисунок 41), а при выборе редактирования или удаления товаров отобразится поиск товаров и список всех товаров, соответствующих поиску, которые после можно редактировать или удалить, это можно увидеть на рисунке 42, поиск можно производить как по названию товара, так и по составу.

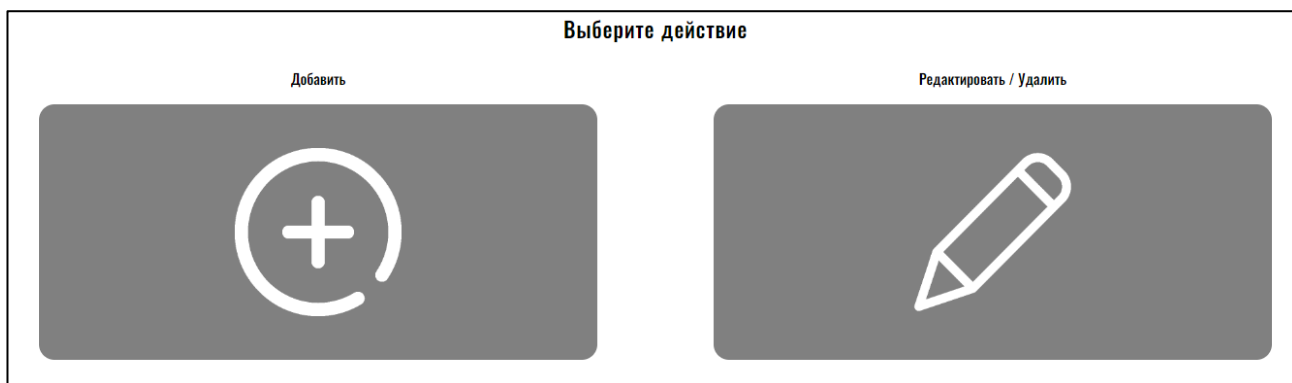


Рисунок 40 – Раздел «Товары»

Добавить товар

Торты

Конфеты

Мороженое

Десерты

Название:

Введите название

Цена:

Введите цену

руб.

Введите состав

Название

0 Р

Сохранить

Рисунок 41 – Создания товара

Редактирование

клубни

Найти

Клубничная башня

Состав: Пшеничная мука, яйца, сахар, сливочное масло, молоко, соль

✎ ✕

1 598 Р

Лето

Состав: Крем, рожок, клубника

✎ ✕

110 Р

Рисунок 42 – Редактирование товара

Четвертый раздел «Аккаунты» содержит в себе таблицу с аккаунтами, которые можно редактировать или удалить, фильтр по ролям и поиск по имени или номеру телефона (рисунок 43).

Чтобы редактировать или удалить аккаунт нужно нажать на кнопки справа в строке с нужным аккаунтом.

Так же здесь можно создать новый аккаунт, для этого нужно нажать на кнопку «Создать» и откроется форма создания аккаунта (рисунок 44), где нужно не забыть выбрать роль и записать данные аккаунта, а снизу будет отображаться предварительная карточка аккаунта.

Так же при выборе аккаунтов с ролью «курьер» таблица немного поменяется, и будет отображать личную информацию о курьере и находится ли он сейчас в сети или нет (рисунок 45).

Аккаунты

Создать (+)

Все
Пользователи
Курьеры

Найти

Имя	Телефон	кол-во заказов	роль	присоединился	РЕДАКТИРОВАТЬ
Имя	88005553536	1	moderator	26.11.2022	
Василий	88005553535	19	Администратор	01.12.2022	
Антон	88005553519	0	Пользователь	08.05.2023	
Gena	88005553532	0	Пользователь	19.05.2023	
test1	88005553538	0	Пользователь	03.07.2023	
...

Найдено записей: 10

Рисунок 43 – Раздел «Аккаунты»

Создать аккаунт

Пользователь
Курьер

Введите имя

Введите телефон

Введите пароль

Пользователь

ФИО:

Телефон:

Сохранить

Рисунок 44 – Создание аккаунта



ИМЯ	ТЕЛЕФОН	КОЛ-ВО ЗАКАЗОВ	ДАТА РОЖДЕНИЯ	АДРЕС ПРОЖИВАНИЯ	ПРИСОЕДИНИЛСЯ	ОНЛАЙН	БАЛАНС	РЕДАКТИРОВАТЬ
Фамилия И. О.	89642775410	1	07.06.2004	г. Иркутск ул. Байкальская д. 100 кв. 1	29.10.2023	●	0	 

Рисунок 45 – Аккаунты курьеров

В пятый раздел «Поддержка» будут приходить обращения от курьеров и другого персонала. При отсутствии обращений раздел будет выглядеть как на рисунке 46.

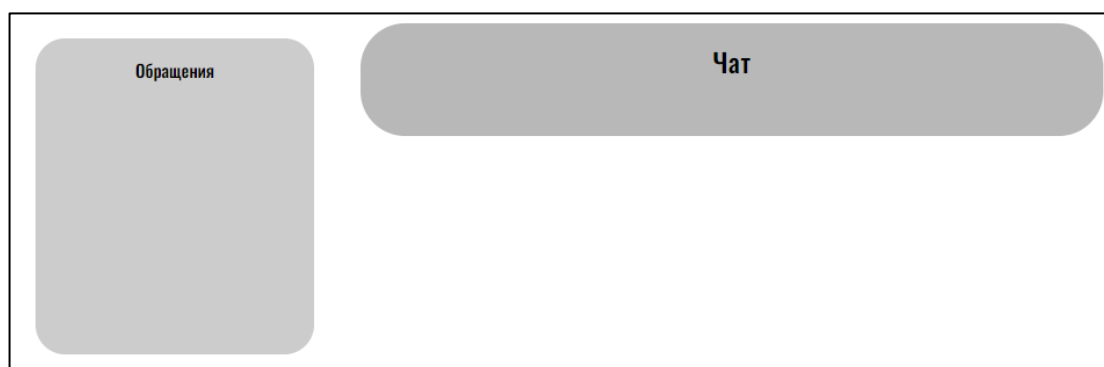


Рисунок 46 – Раздел «Поддержка»

Слева находится блок с обращениями, а справа при выборе обращения будет открываться чат.

При активных обращениях на боковой панели появится пометка, как на рисунке 47.

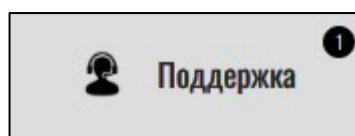


Рисунок 47 – Уведомление о количестве обращений

Если зайти в раздел «Поддержка», когда есть активные обращения и выбрать одно из них, то это будет выглядеть как на рисунке 48.



Рисунок 48 – Открытое обращение

Как видно на рисунке 48 при открытом обращении можно что-то написать или отменить обращение, ничего страшного если чат пустой, это значит, что пользователь ещё не написал текст обращения.

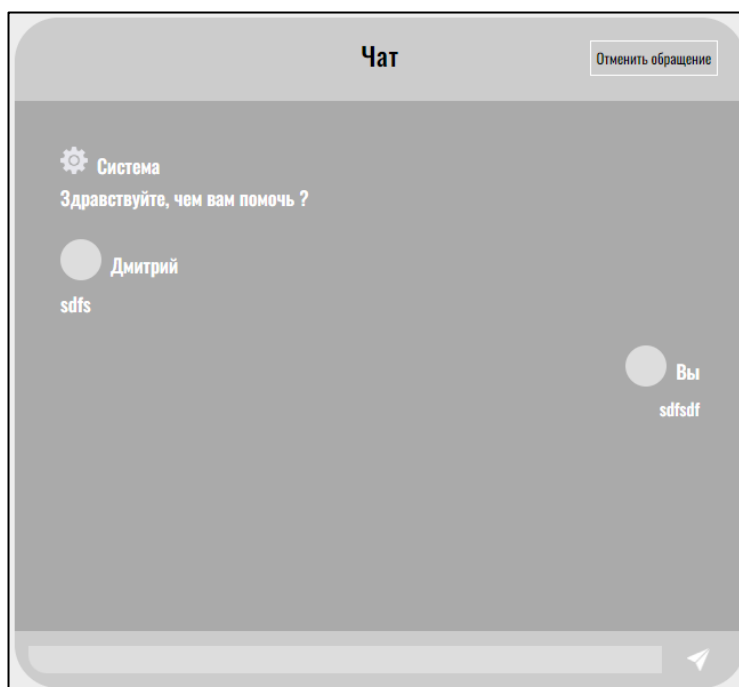


Рисунок 49 – Диалог с пользователем

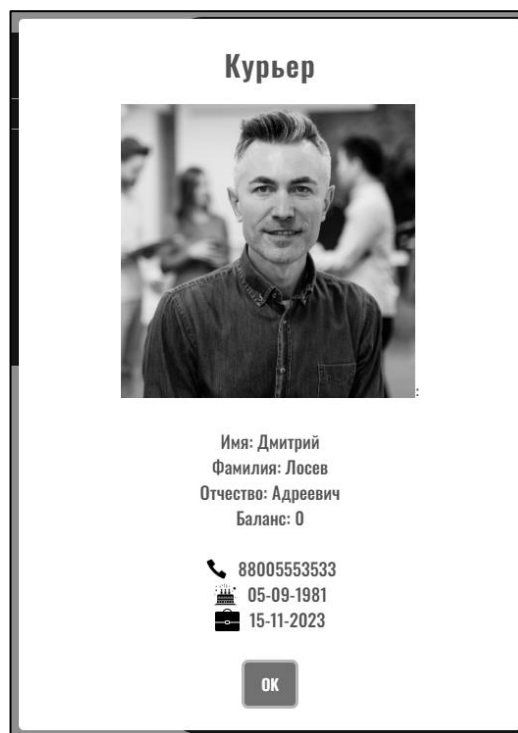


Рисунок 50 – Подробная информация о пользователе

После того как пользователь напишет текст обращения, оно в реальном времени отобразится в окне чата (рисунок 49) и можно будет нажать на имя пользователя чтобы увидеть подробную информацию о пользователе (рисунок 50).

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

ЗАКЛЮЧЕНИЕ

В ходе дипломного проекта была поставлена задача разработать интернет-магазин «Сладости». Были определены требования для продукта. Исходя из требований продукта были рассмотрены возможные способы реализации интернет-магазина при использовании различных языков программирования, типов баз данных и рабочих сред. Выбором для серверной части стал Node.js, так как подходил под все требования и был удобен для разработки интернет-магазина.

Для клиентской части был выбран React - библиотека JavaScript, ведь он является одним из лидирующих библиотек для разработки клиентской части интернет-магазина. Для создания базы данных был выбран MongoDB, так как его использование систематизирует создание и управление коллекциями базы данных, а также позволяет удобно хранить множество типов данных в одном документе.

Рабочей средой был выбран VSCode, потому что он имеет современный вид и поддерживает множество удобных плагинов для разработки. Выбор был обоснован и основывался на полезных функциях и удобстве данных инструментов.

Были разработаны диаграммы и схемы, позволяющие проще и удобнее понять принцип действия функций и их структуры. Заранее до разработки были продуманы макеты страниц и структура баз данных. Однако в самом процессе разработки появились новые идеи и лучшие решения, в связи чего приходилось несколько раз менять структуру, схемы, диаграммы баз данных и улучшать макеты страниц.

По итогу в дипломном проекте были выполнены все главные цели, разработан удобный и понятный для пользователей интернет-магазин, который подходит под все критерии технического задания и соответствует задумке самого дипломного проекта.

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Devhints.io – Справочник React – URL: <https://devhints.io/react> (дата обращения: 20.03.2024). – Текст: электронный.

2 Express-rate-limit – Официальная документация Express-rate-limit – URL: <https://github.com/express-rate-limit/express-rate-limit> (дата обращения: 30.03.2024). – Текст: электронный.

3 Expressjs.com – Официальная документация Express.js – URL: <https://expressjs.com/ru/guide/routing.html> (дата обращения: 20.03.2024). – Текст: электронный.

4 Expressjs/multer – Официальная документация multer – URL: <https://github.com/expressjs/multer> (дата обращения: 24.03.2024). – Текст: электронный.

5 GitHub.com – Официальная документация MERN – URL: <https://github.com/vasansr/pro-mern-stack> (дата обращения: 20.03.2024). – Текст: электронный.

6 HTML5CSS.ru – JavaScript учебник – URL: <https://html5css.ru/js/default.php> (дата обращения: 22.03.2024). – Текст: электронный.

7 JavaScript.ru – JavaScript учебник – URL: <https://learn.javascript.ru> (дата обращения: 22.03.2024). – Текст: электронный.

8 Mongodb.com – Официальная документация MongoDB – URL: <https://www.mongodb.com/docs/> (дата обращения: 23.03.2024). – Текст: электронный.

9 Mongoosejs.com – Официальная документация Mongoose – URL: <https://mongoosejs.com/docs/> (дата обращения: 23.03.2024). – Текст: электронный.

10 Nodejs.org – Официальная документация Node.js – URL: <https://nodejs.org/ru/docs> (дата обращения: 23.03.2024). – Текст: электронный.

11 React.dev – Официальная документация React – URL: react.dev/learn (дата обращения: 25.03.2024). – Текст: электронный.

12 Reactrouter.com – Официальная документация React-router – URL: <https://reactrouter.com/en/main/start/tutorial> (дата обращения: 26.03.2024). – Текст: электронный.

13 Socket.io – Официальная документация Socket.io – URL: <https://socket.io/docs/v4/> (дата обращения: 27.03.2024). – Текст: электронный.

14 Sweetalert2.github.io/ – Официальная документация Sweetalert2 – URL: <https://sweetalert2.github.io/> (дата обращения: 29.03.2024). – Текст: электронный.

15 Stack Overflow – Решение ошибок – URL: <https://stackoverflow.com> (дата обращения: 24.03.2024). – Текст: электронный.

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А – Листинг кода сервера

```
import express from 'express';
import mongoose from 'mongoose';
import cors from 'cors';
import * as dotenv from 'dotenv-flow';
import routes from './routes.js';
import Socket from './utils/socket.js';
import rateLimit from 'express-rate-limit'

dotenv.config()

// Подключение базы данных
mongoose
.set('strictQuery', true)
.connect(process.env.DB)
.then(() => console.log('DB OK'))
.catch((err) => console.log('DB ERROR', err));

// Получение значений из локального окружения
const PORT = process.env.PORT
const HOST = process.env.HOST
const ENV = process.env.NAME
const CLIENT = process.env.CLIENT
```

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

```

// Настройка сервера

const app = express();

app.use(express.json());

app.use(cors({ origin: CLIENT }));


// Ограничение запросов

const limiter = rateLimit({

windowMs: 1000, // 1 сек

max: 20, // Запросов в {windowMs} сек

message: 'Too many requests from this IP, please try again later.',

});

app.use(limiter);


// Подключение маршрутов

app.use('/', routes);


const server = app.listen(PORT, HOST, (err) =>{

if (err){ return console.log(err) }

if (ENV){ console.log(ENV) }

console.log(`Server running at http://${HOST}:${PORT}/`);

});


// Использование сокета

Socket(server);

export default server

```

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение Б – Работа с файлами

```
import checkAuth from './checkAuth.js';

import multer from "multer"

import express from 'express';

import path from 'path';

import crypto from 'crypto';


const app = express();


// Функция для фильтрации файлов по типу

const fileFilter = (req, file, cb) => {

  if (file.mimetype === 'image/jpeg' || file.mimetype === 'image/jpg' || file.mimetype
  === 'image/png' || file.mimetype === 'image/gif') {

    cb(null, true);

  } else {

    cb(new Error('Допустимы только файлы типов JPEG, JPG, PNG, GIF'), false);

  }

};


const storage = multer.diskStorage({

  destination: (req, res, cb) =>{

    cb(null, 'uploads')

  },

  filename: (_, file, cb) =>{ cb(null, file.originalname) },

});
```

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		


```
const upload = multer({ storage, fileFilter});
```

```
app.use('/uploads', express.static('uploads'));
```

```
app.post('/uploads', checkAuth, upload.single('image'), (req,res) => {  
  res.json({  
    url : `/uploads/${req.file.originalname}`,  
  }));
```

```
export default app
```

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В – Создание заказа

```
export const create = async (req,res) =>{  
  try {  
    const count = async () => {  
      return new Promise(async(res, rej) => {  
        const orders = await OrderModel.find()  
        if(orders.length >= 1){  
          OrderModel.findOne().sort({ number: -1 }).exec(function(err, doc) {  
            if (err) { res(rej)}  
            else { res(doc.number + 1) })))  
          else{ res(1) })})  
        }  
        const quantity = await count();  
        const doc = new OrderModel({  
          number: quantity,  
          user: req.body.user,  
          products: req.body.products,  
          fullPrice: req.body.fullPrice,  
          username: req.body.username,  
          phone: req.body.phone,  
          city: req.body.city,  
          street: req.body.street,  
          house: req.body.house,  
          apartment: req.body.apartment,
```

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

```

coordinates: req.body.coordinates,
methodDelivery: req.body.methodDelivery,
status: 'new',
nonAuthUser: req.body.nonAuthUser
}))

const order = await doc.save()

OrderModel.findOne({ _id: doc._id },
(err,doc) =>{
if (err){
console.log(err)
return res.status(500).json({
message:"Не удалось вернуть заказ"
});
}
if (!doc){
return res.status(404).json({
message:"Заказ не найден"
});
}
res.json(doc)
}).populate('products.product').populate('user');
} catch (err) { console.log(err); res.status(500).json({ message:"Не удалось создать заказ" })}
}

```

Приложение Г – Тесты функций авторизации

```
import mongoose from "mongoose";

import product from '../models/product.js'

import chai from 'chai'

import chaiHttp from 'chai-http'

import server from '../index.js'


const expect = chai.expect;

chai.use(chaiHttp);


let token;

const user = {
  fullName: "Name",
  phone: "88005553535",
  password: "12345"
};

describe('Авторизация', () => {
  it('Регистрация аккаунта', (done) => {
    chai.request(server)
      .post('/auth/register')
      .send(user)
      .end((err, res) => {
        expect(res.status).to.be.equal(200);
        expect(res.body).to.be.a('object');
```

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

```

expect(res.body.error).to.be.equal(null || undefined);

done()});

});

it('Авторизация', (done) => {

chai.request(server)

.post('/auth/login')

.send({

"phone": user.phone,

"password": user.password

})

.end((err, res) => {

expect(res.status).to.be.equal(200);

expect(res.body).to.be.a('object');

expect(res.body.error).to.be.equal(null || undefined);

token = res.body.token;

done()});

});

it('Получение пользователя', (done) => {

chai.request(server)

.get('/auth/me')

.set('Authorization', `Bearer ${token}`)

.end((err, res) => {

expect(res.status).to.be.equal(200);

expect(res.body).to.be.a('object');

```

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

```

expect(res.body.error).to.be.equal(null || undefined);

done()}}));

});

describe('Middleware', () => {

it('Запрос для авторизованных пользователей', (done) => {

chai.request(server)

.get('/auth/me')

.set('Authorization', `Bearer ${token}`)

.end((err, res) => {

expect(res.status).to.be.equal(200);

expect(res.body).to.be.a('object');

expect(res.body.error).to.be.equal(null || undefined);

done() });

});

it('Ошибка при запросе на защищенный ресурс', (done) => {

chai.request(server)

.get('/orders/active')

.set('Authorization', `Bearer ${token}`)

.end((err, res) => {

expect(res.status).to.be.equal(403);

expect(res.body.message).to.be.equal('Нет доступа');

done() }}});

});

```

					ДП.09.02.07-3.24.201.15.ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		