

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

КП.09.02.07-3.23.201.14. ПЗ

**ИНФОРМАЦИОННАЯ СИСТЕМА
«МАГАЗИН СЛАДОСТЕЙ»**

Председатель ВЦК:	_____	(А.С. Александрова)
	(подпись, дата)	
Руководитель:	_____	(А.С. Александрова)
	(подпись, дата)	
Студент:	_____	(В.О. Никифоров)
	(подпись, дата)	

Иркутск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Описание предметной области	4
2 Анализ инструментальных средств разработки, используемых при реализации информационной системы.....	6
3 Техническое задание	12
4 Проектирование информационной системы	13
4.1 Структурная схема информационной системы.....	13
4.2 Функциональная схема информационной системы	16
4.3 Проектирование базы данных.....	20
4.4 Проектирование пользовательского интерфейса.....	25
5 Разработка информационной системы.....	29
5.1 Разработка интерфейса информационной системы.....	29
5.2 Разработка базы данных информационной системы.....	30
5.3 Разработка информационной системы.....	33
5.4 Тестирование информационной системы.....	38
6 Документирование программного продукта	44
6.1 Руководство пользователя информационной системы	44
6.2 Руководство администратора информационной системы	46
6.3 Руководство пользователя для курьеров информационной системы.....	55
ЗАКЛЮЧЕНИЕ	60
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	61
Приложение А – Техническое задание	62
Приложение Б – Листинг.....	68

					КП.09.02.07-3.23.201.14. ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Никифоров В.О.			ИНФОРМАЦИОННАЯ СИСТЕМА «МАГАЗИН СЛАДОСТЕЙ» пояснительная записка	Лит.	Лист	Листов
Провер.		Александрова А.С.					2	68
						ГБПОУИО «ИАТ» ВЕБ-20-1		

ВВЕДЕНИЕ

В информационной системе магазина сладостей можно купить множество различных сладостей и вкусовностей.

Актуальность данного магазина заключается в том, что такая еда как сладости будут актуальны на протяжении многих десятков лет, так как люди очень любят иногда себя побаловать, данная информационная система должна обеспечить это и помочь проще 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 данного курсового проекта является разработка информационной системы 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 информационной системы;
- разработать интерфейс пользователя информационной системы в соответствии с техническим заданием;
- разработать информационную систему магазина сладостей и составить программную документацию в виде руководства пользователя.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

1 Описание предметной области

Информационная система магазина сладостей – это удобный способ покупки нужных товаров, в данном случае необходимые сладости, которые можно купить в интернете в любое удобное время, а так же 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. Информационная система позволит повысить производительность и качество обслуживания, и не заставит покупателя долго искать свои сладости, а также предусмотрит их продажу.

Для персонала будут свои модули информационной системы, позволяющие удобно управлять и пользоваться информационной системой.

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, номер телефона и адрес;

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						4
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 после подтверждения заказа администратором, заказ передаётся курьерам.

– курьер получает точный адрес на карте, данные заказа и покупателя, и после принятия заказа отвозит ему его сладости;

– пользователь оплачивает товар у курьера;

– производится регистрация оплаты;

– заказ вносится в отчёт.

После того как заказ будет оформлен, администратору необходимо будет подтвердить заказ, после чего пользователь уже не сможет его отменить, но если заказ был только создан и ещё не подтверждён администратором, то пользователь при необходимости всё же может сделать отмену заказа.

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

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

Но если пользователь не был авторизован, то он всё равно после заказа попадает на страницу истории заказов, но заказы будут храниться не в базе данных, а в памяти браузера. Потому после регистрации или авторизации, история заказа преимущественно будет загружаться из базы данных, из-за чего история заказов, сохраненная в памяти браузера, не будет отображаться, однако если выйти из аккаунта, заказы, хранившиеся в памяти браузера, будут снова отображаться.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

2 Анализ инструментальных средств разработки, используемых при реализации информационной системы

Инструменты разработки программного продукта сильно влияют на процесс создания информационной системы, они упрощают и ускоряют работу.

Чтобы было удобно проектировать структуру информационной системы можно использовать Draw.io, а его дизайн – через онлайн-сервис Figma. Информационная система 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. В данном проекте используется для создания прототипа страниц.

Figma – онлайн-сервис для дизайнеров, веб-разработчиков и маркетологов. Он предназначен для создания прототипов сайтов или приложений, иллюстраций и векторной графики. В редакторе можно настроить совместную работу, вносить и обсуждать правки, причем как в браузере, так и через приложение на компьютере. 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.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

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. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам – функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания – что придаёт языку дополнительную гибкость.

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

Информационная система 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. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

На сегодняшний день является самой популярной серверной базы данных (далее – БД), за счёт своей простоты, скорости работы и внушительного функционала.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

SQLite – компактная встраиваемая СУБД с исходным кодом. В 2005 году проект получил награду Google-O'Reilly Open Source Awards. SQLite поддерживает динамическое типизирование данных. Возможные типы значений: INTEGER, REAL, TEXT и BLOB. Также поддерживается специальное значение NULL.

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 отличие от MySQL не требует 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.

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

Таблица 1 – Сравнение средств реализации базы данных

Название БД	MySQL	SQLite	MongoDB
Большое кол-во типов данных	+	+	+
Популярность	+	+	+
Не требует удаленного сервера	-	+	-
Простота использования	+	+	+
Портативность	+	+	+

Таким образом, в качестве базы данных 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. Язык является полностью объектно-ориентированным – всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов.

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.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

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 лёгкая 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 как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

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

Несмотря на столь высокую популярность программы, ее функционал до сих пор остается не таким очевидным, из-за чего многие пользователи предпочитают продукт конкурента, нежели VS Code. Чтобы решить все проблемы, возникающие при работе с Visual Studio Code, рекомендую ознакомиться с подробным гайдом по настройке этого редактора кода. В нем я покажу, что VS Code – это мощный инструмент, которым может пользоваться каждый.

Sublime Text 3 – это текстовый редактор, разработанный для верстальщиков и программистов. Он позволяет работать с кодом разных языков программирования: от Erlang до C++. Свою популярность он получил благодаря кроссплатформенной поддержке и расширенным настройкам, которые позволяют пользователю легко «играть» с параметрами программы.

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 CSS/HTML/JS	+	+	+

Таким образом, после рассмотрения вариантов средств разработок, было принято решение использовать VS Code. Visual Studio и другие тоже оказались неплохими, но в них есть множество мелких преимуществ и удобств которых нет или сделаны хуже, чем в VS Code.

3 Техническое задание

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 информационной системы;
3. требования к информационной системе в целом;
 - 3.1. требования к структуре и функционированию информационной системы;
 - 3.2. требования к надёжности;
 - 3.3. требования к безопасности;
 - 3.4. требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов информационной системы;
4. требования к документированию;
5. состав и содержание работ по созданию информационной системы.

Техническое задание на разработку информационной системы представлено в приложении А.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

4 Проектирование информационной системы

4.1 Структурная схема информационной системы

Одним из важнейших этапов разработки информационных систем 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 информационной системы.

Диаграмма вариантов использования, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать информационную систему 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 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 1 изображена Use Case View, которая показывает структурную схему информационной системы «Сладости». Кроме этого, она отображает действия и возможности пользователей приложения. Актёрами являются: «Администратор», «Зарегистрированный пользователь», «Гость», «Курьер».

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

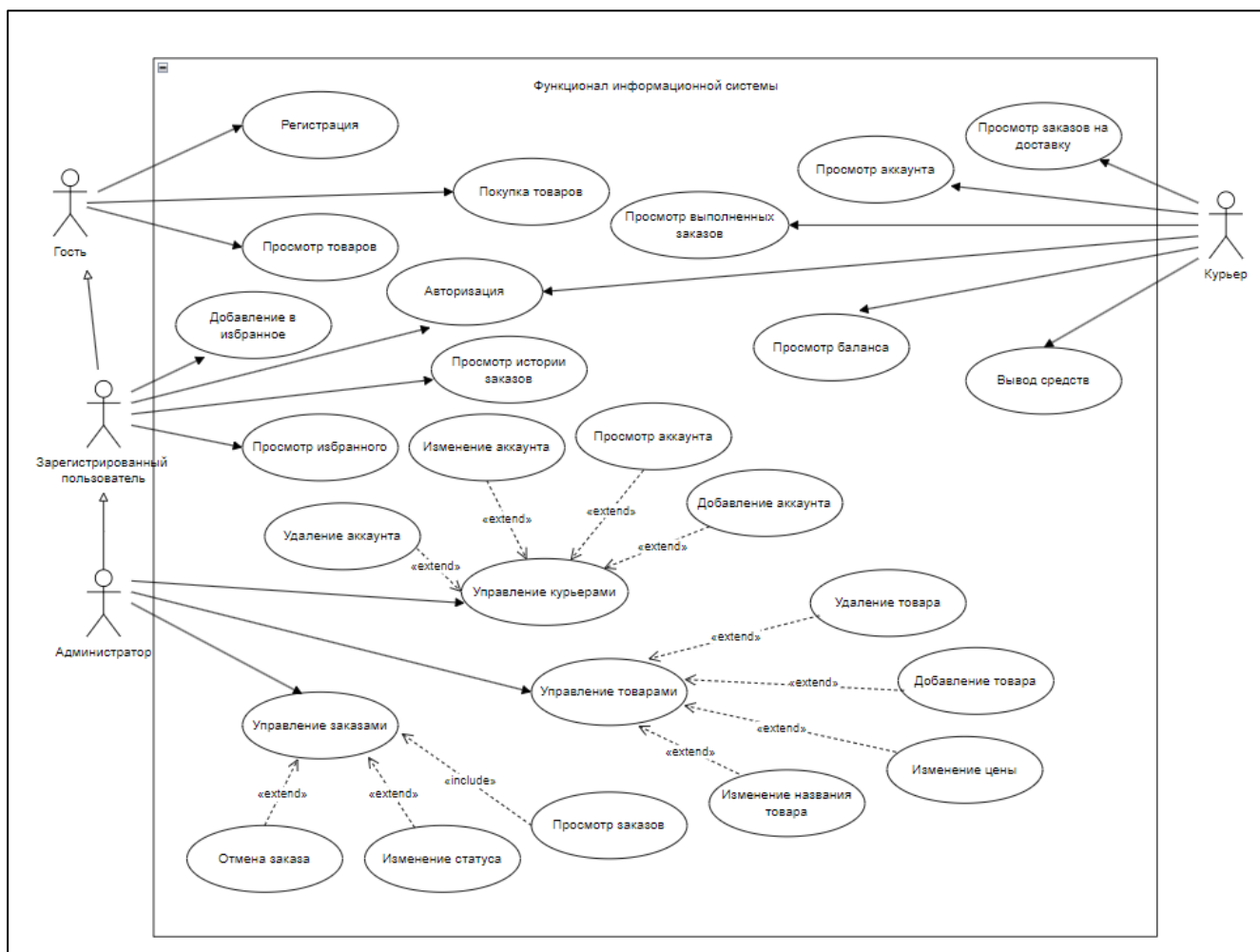


Рисунок 1 –Диаграмма прецедентов

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 2.

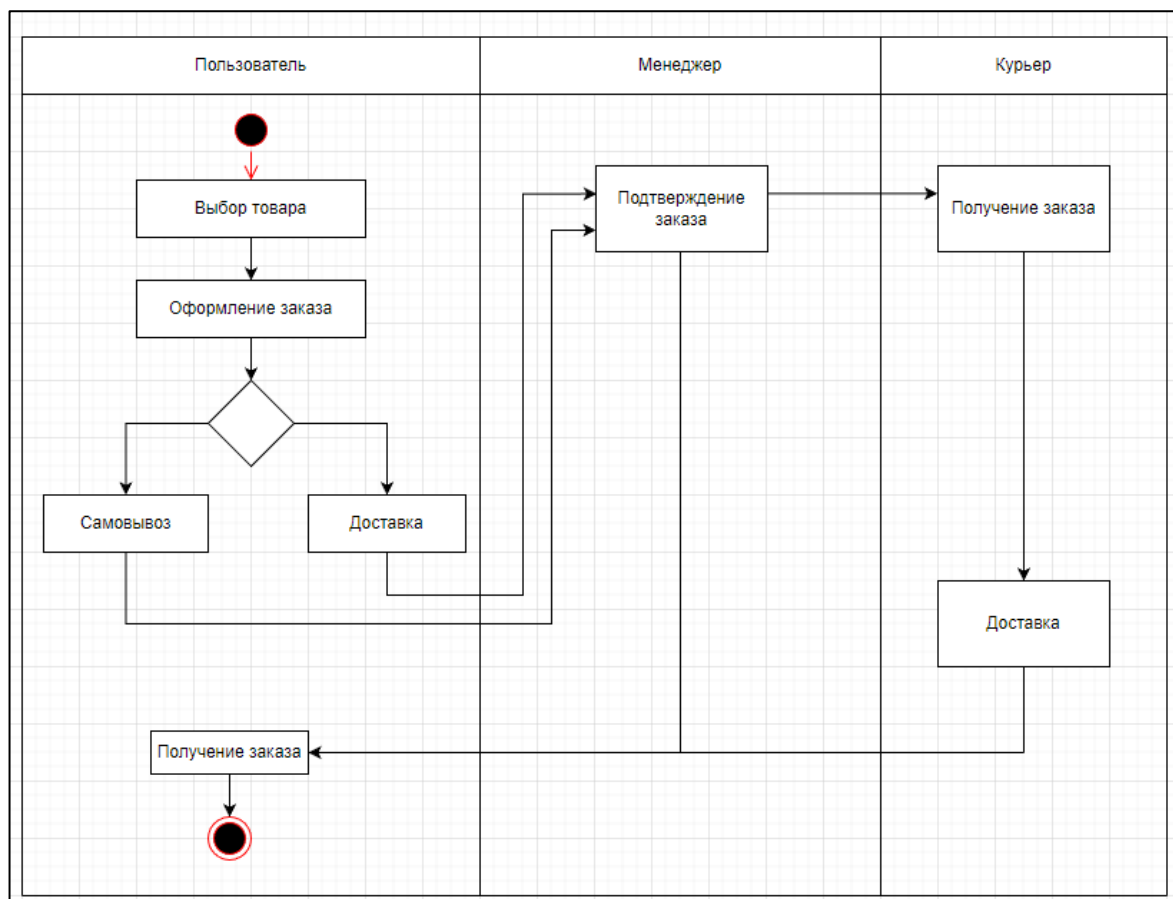


Рисунок 2 – Диаграмма деятельности

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

Диаграмма компонентов – это UML-диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и тому подобные. Диаграмма компонентов изображена на рисунке 3.

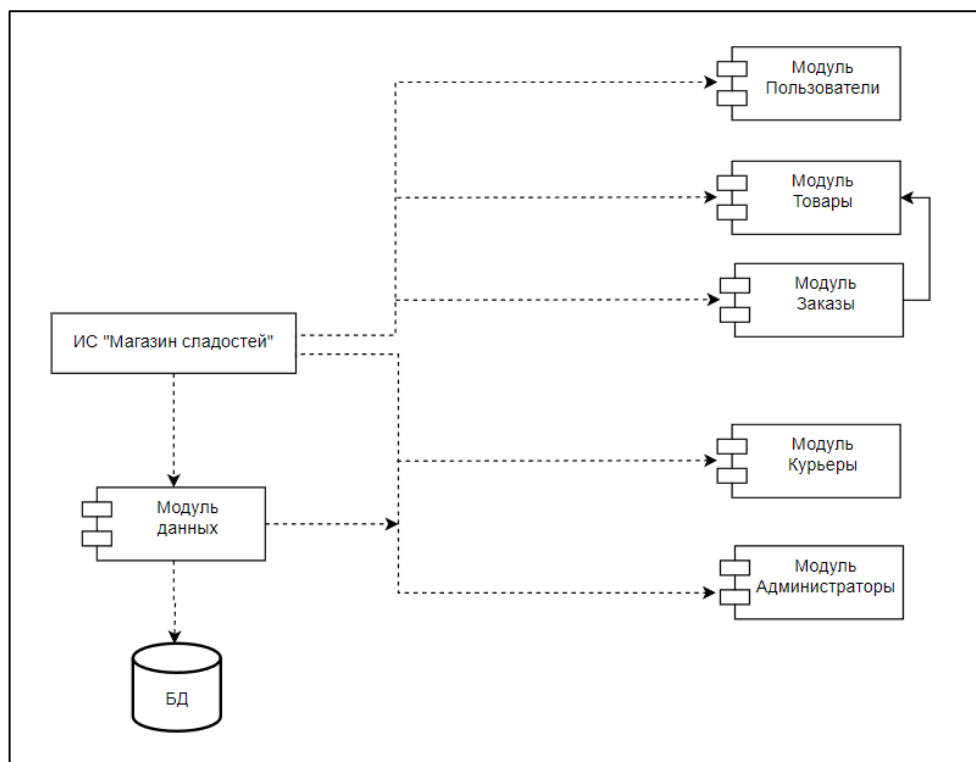


Рисунок 3 – Диаграмма компонентов

После проектирования диаграммы компонентов можно увидеть структурные компоненты информационной системы со связи между ними.

4.2 Функциональная схема информационной системы

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.

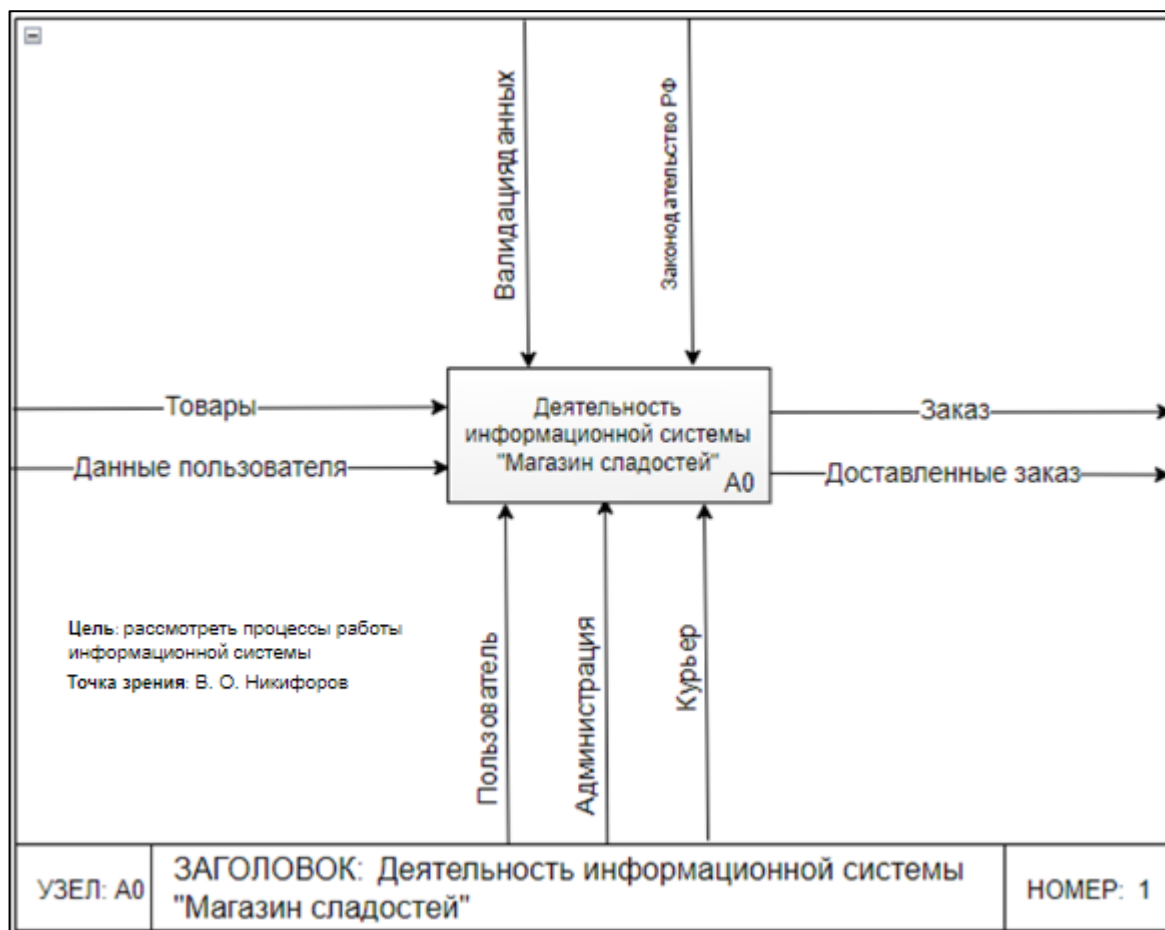


Рисунок 4 – Контекстная диаграмма

В результате проектирования контекстной диаграммы можно увидеть функциональную модель оформления заказа информационной системы.

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 узла первого уровня можно увидеть более подробную функциональную модель оформления заказа информационной системы.

Следующей диаграммой является диаграмма декомпозиции узла второго уровня, которую можно увидеть на рисунке



Рисунок 6 – Диаграмма декомпозиций узла A2

В результате проектирования диаграммы декомпозиции узла второго уровня можно рассмотреть подробно блоки, которые описывают исходный блок.

На рисунке 7 находится диаграмма классов, демонстрирующая классы системы, их атрибуты, операции (или методы) и взаимосвязи между объектами.

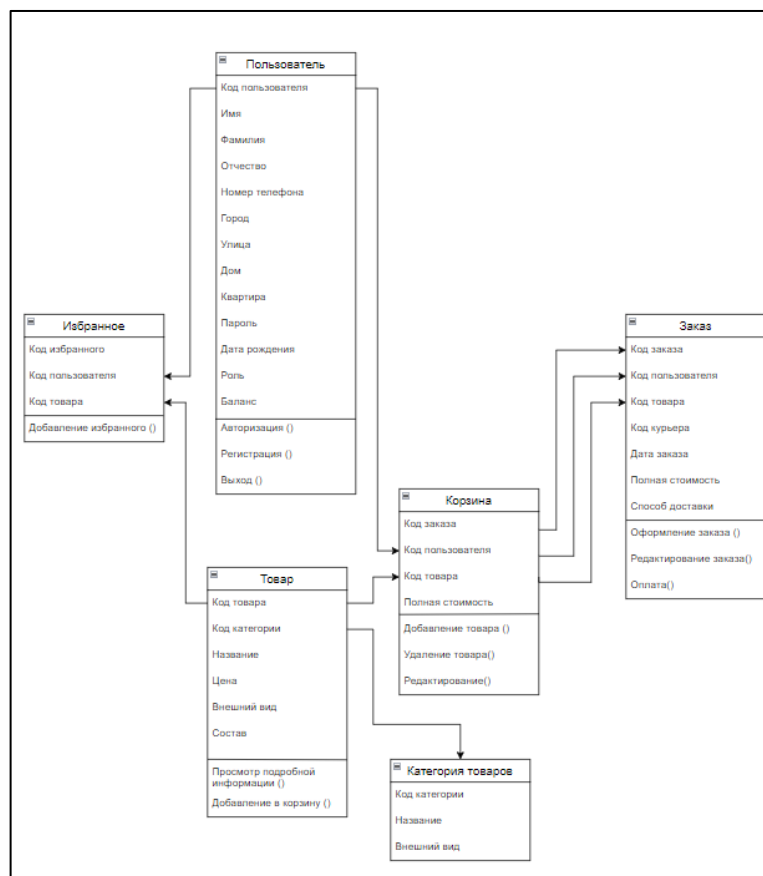


Рисунок 7 – Диаграмма классов

На диаграмме классов можно увидеть, что есть семь объектов, «Пользователь», «Заказ», «Товар», «Избранное», «Категория товаров», «Корзина», а связь между ними осуществляется по их идентификационным кодам.

DFD – это диаграмма потоков данных, которая изображена на рисунке 8. 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. С помощью этой

диаграммы проводится структурный анализ и проектируются информационные системы.

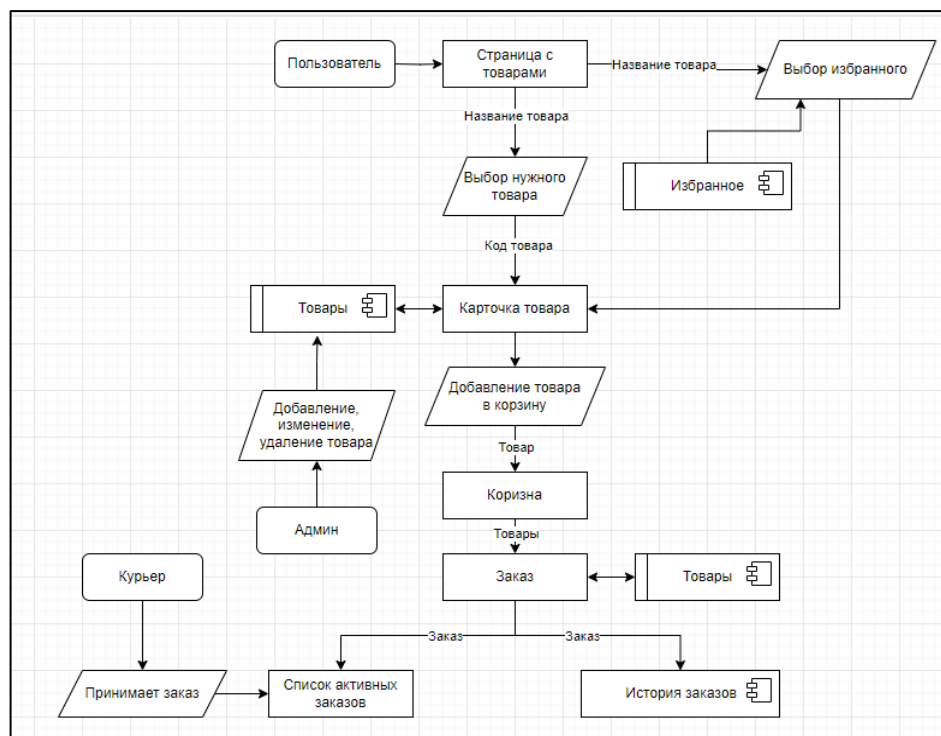


Рисунок 8 – Диаграмма потоков данных DFD

При проектировании диаграммы потоков данных DFD и классов стало наглядно видно, что будет представлять из себя информационная система, и как будет происходить обработка и передача данных.

4.3 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.

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных.

Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

Инфологическую модель базы данных можно увидеть на рисунке 9.

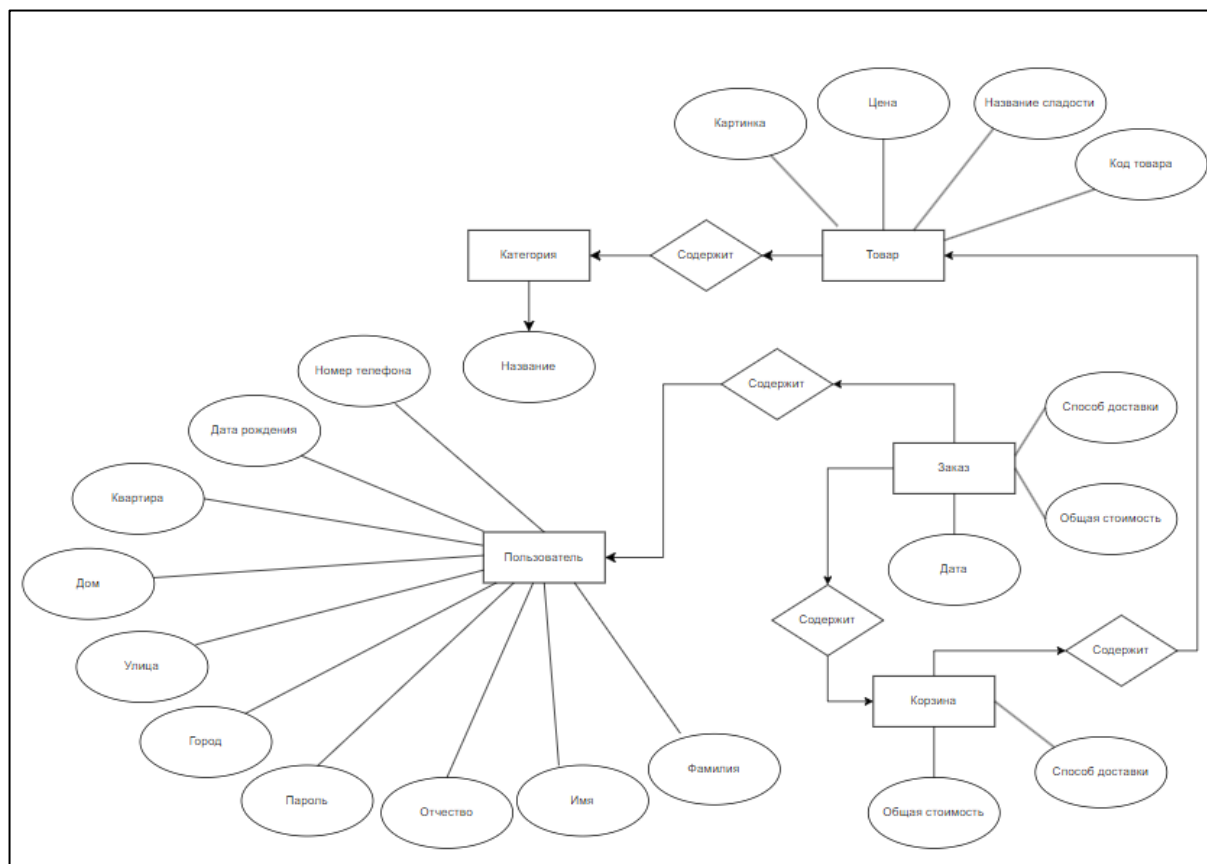


Рисунок 9 – Инфологическая модель базы данных

На модели видно три сущности, которые имеют свои уникальные атрибуты. Также сущности связаны между собой и в сумме получается структура под названием – инфологическая модель.

На рисунке 10 можно увидеть даталогическую модель, которая показывает, какие данные отправляются и хранятся в базе данных.

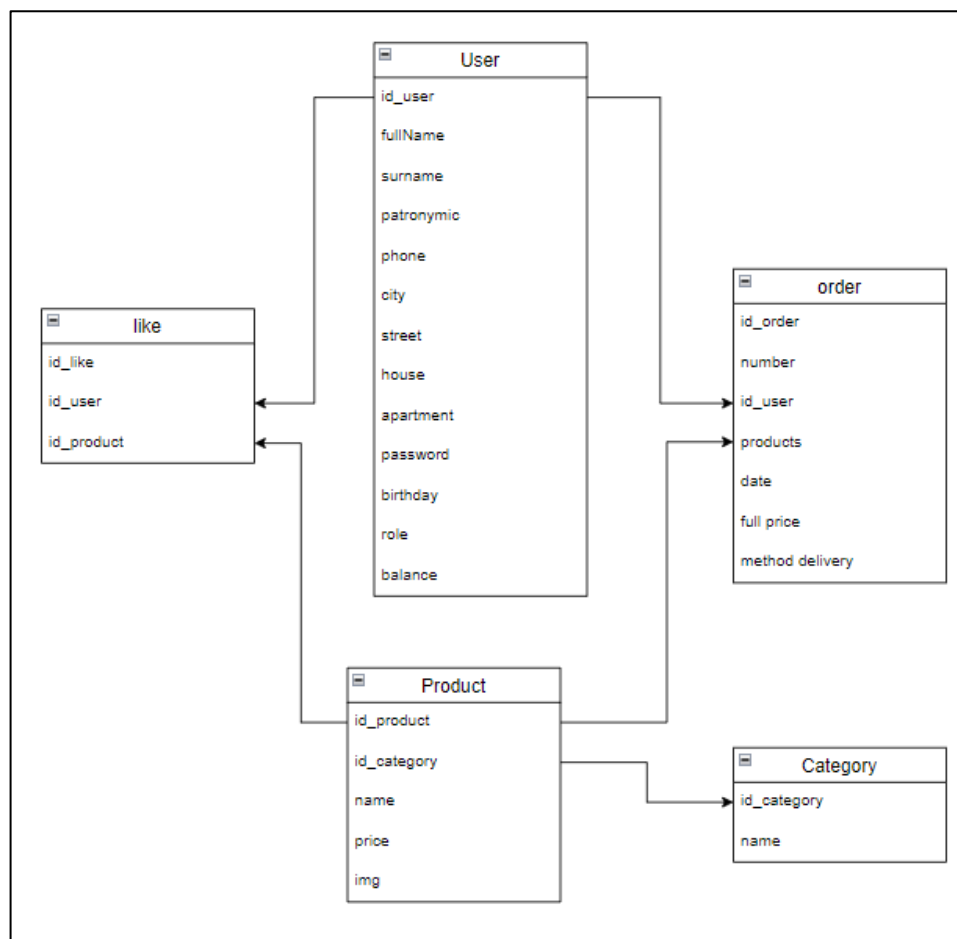


Рисунок 10 – Даталогическая модель

Проектирование базы данных не ограничивается только этим этапом разработки, а проходит на протяжении всей разработки, до того момента пока в системе не появятся данные, которые нельзя потерять. Результатом проектирования базы данных является ER-модель, которая изображена на рисунке 11.

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.

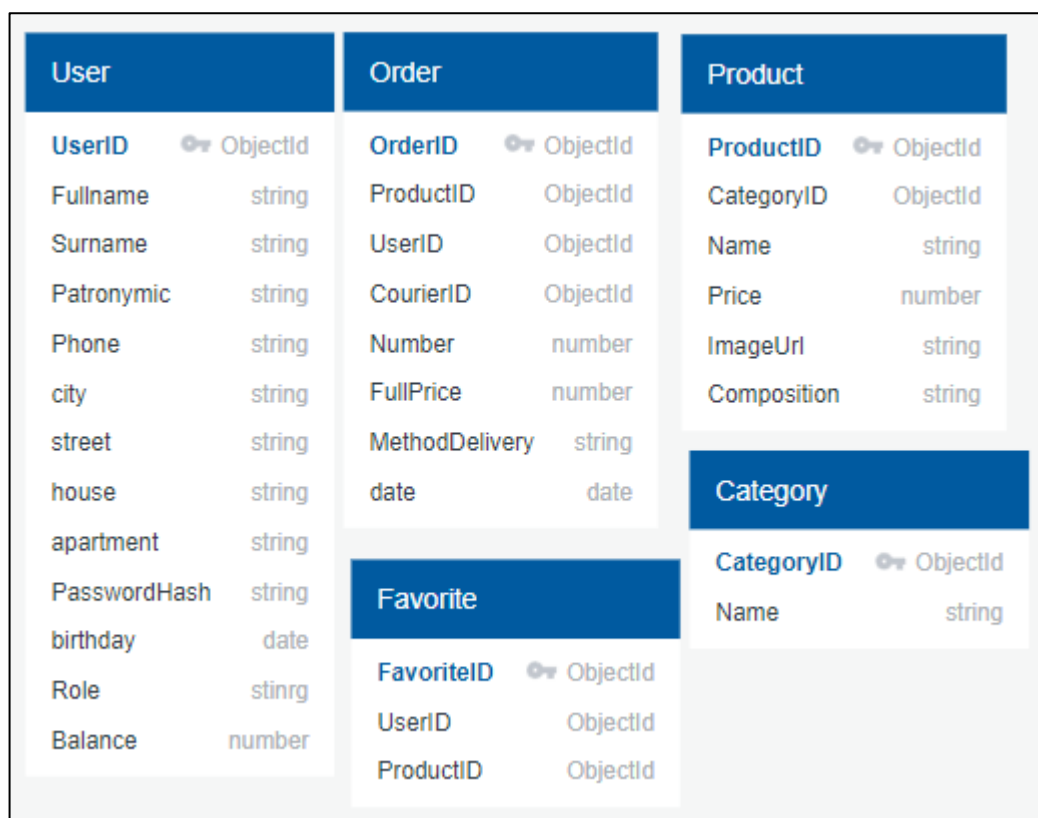


Рисунок 11 –ER – модель базы данных

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);
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);
 - ProductID (идентификатор товара);
 - UserID (идентификатор пользователя);
 - Number (номер заказа);
 - FullPrice (итоговая цена);
 - MethodDelivery (способ доставки);
 - Date (дата 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.

4.4 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, 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 12 отображена страница с каталогом, на которой можно выбрать, нужную для нас категорию сладостей.



Рисунок 12 – Каталог

При выборе категории «Торты», пользователь попадает на страницу на рисунке 13.

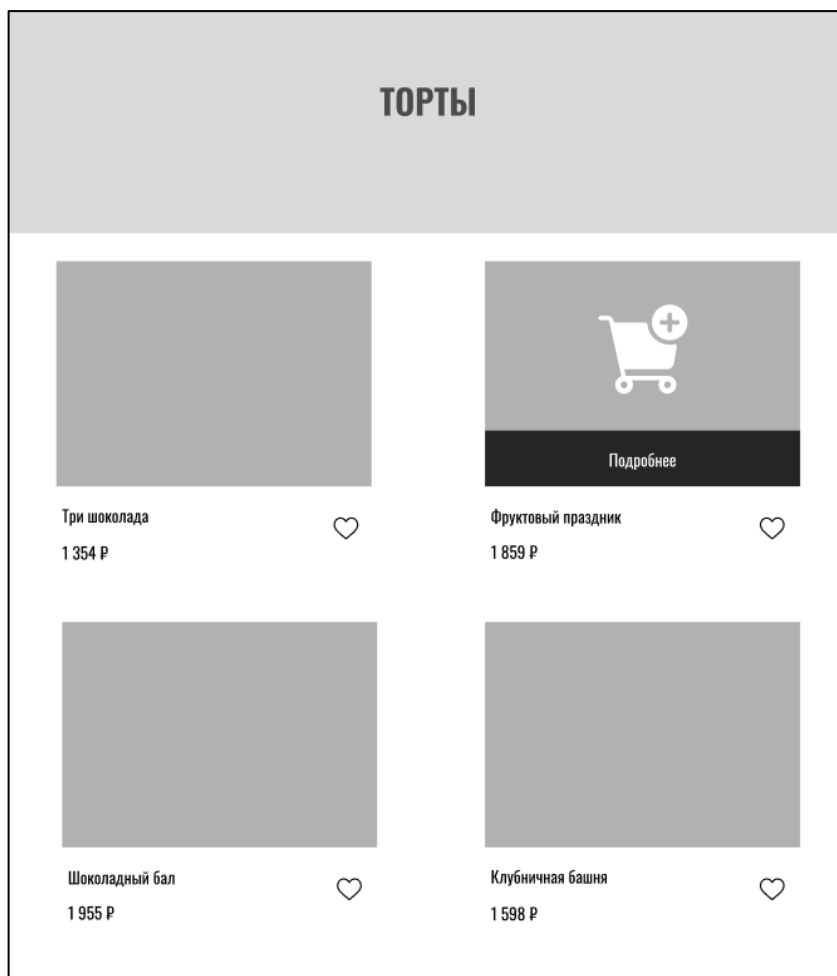


Рисунок 13 – Страница «Торты»

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 14.

На любой странице можно авторизоваться или выйти из аккаунта. Саму авторизации можно увидеть на рисунке 15.

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 14 – Страница корзины

×

Авторизация

Введите номер телефона

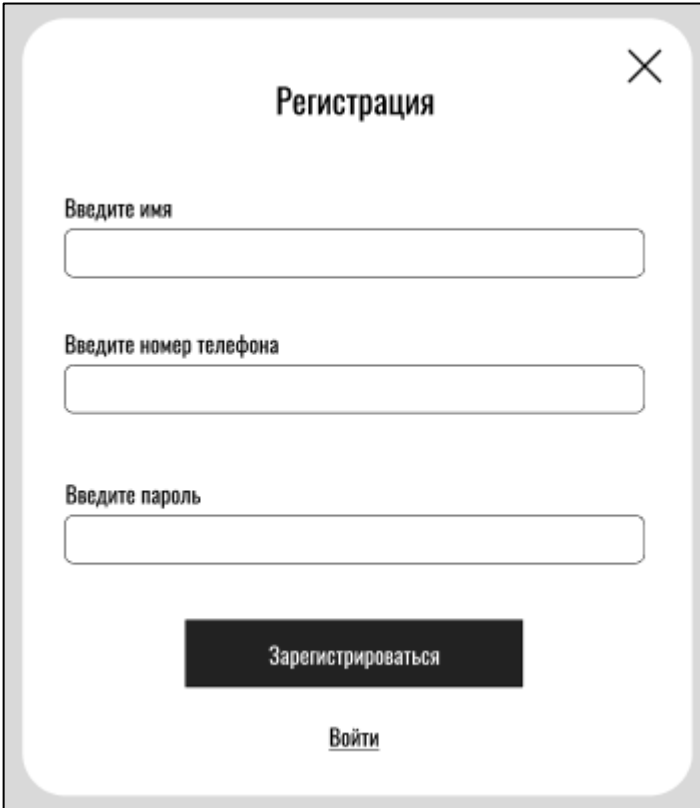
Введите пароль

Войти

Зарегистрироваться

Рисунок 15 – 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 16.



Регистрация

Введите имя

Введите номер телефона

Введите пароль

Зарегистрироваться

[Войти](#)

Рисунок 16 – 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.

5 Разработка информационной системы

5.1 Разработка интерфейса информационной системы

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 17.

```
<!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 17 – Корневая страница

Точкой входа (рисунок 18) является index.js, который получает корневой элемент из корневой страницы и записывает в него контент, полученный из React-компонентов, определенных в компоненте маршрутов (рисунок 19), который описывает, какие компоненты должны быть загружены для каждого пути URL.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						29
Изм.	Лист	№ докум.	Подпись	Дата		

```
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 18 – 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 19 – Компонент маршрутов

5.2 Разработка базы данных информационной системы

Разработка базы данных информационной системы реализовывалась в СУБД MongoDB. База данных состоит из 5 коллекций.

Структуры таблиц соответствуют схеме базы данных из пункта 4.3.

На рисунке 20 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 20 – Документ коллекции «Пользователи»

На рисунке 21 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 21 – 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 так же 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 22.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						31
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 22 – Документ коллекции «Заказы»

В информационной системе есть функция добавления товара в избранное, для того чтобы это сделать создается документ коллекции «Избранное» с идентификатором пользователя и понравившегося товара, как на рисунке 23.

```

_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 23 – 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 24.

```
_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 24 – Объект таблицы «Категории»

5.3 Разработка информационной системы

В соответствии с заданием, в данной информационной системе выполнены все необходимые функции.

Разработка функционала информационной системы была реализована на Node.js

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

Первым делом подключаем базу данных через mongoose, а её адрес для подключения, мы достаем для безопасности из локального окружения, это можно увидеть на рисунке 25.

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 25 – Подключение базы данных

В этот же файл подключаются маршруты из специального файла с маршрутами, их пример можно увидеть на рисунке 26, а подключение можно увидеть на рисунке 27.

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

Рисунок 26 – Маршруты

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

Рисунок 27 – Подключение маршрутов

Так же подключатся файлы для работы с файлами и сокетом. Работу с файлами можно увидеть на рисунке 28, а одно из событий сокета на рисунке 29.

```
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)
  },
});

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}`,
  });
});
```

Рисунок 28 – Работа с файлами

```

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 29 – 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 30, а после результат с координатами и остальной информацией о заказе отправляются на сервер, где этот запрос будет ожидать маршрут создания заказа (рисунок 31).

```

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 30 – Запрос на геокодирование

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						35
Изм.	Лист	№ докум.	Подпись	Дата		

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

Рисунок 31 – Маршрут создания заказа

После того как маршрут получит запрос он передает его на проверку валидации (рисунок 32), а после контроллеру для создания заказа (рисунок 33).

```
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 31 – Валидация запроса на создание нового 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 currentTime = new Date().toISOString();

    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,
      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');

    // res.json(order)
  } catch (err) {
    console.log(err)
    res.status(500).json({
      message: "Не удалось создать заказ"
    })
  }
}
```

Рисунок 32 – Функция создания 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 33.

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

Рисунок 33 – Отправка события от клиента

После того как сокет получил событие о создании заказа, он отправляет другое событие (рисунок 34), 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 34 – 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.

5.4 Тестирование информационной системы

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 информационной системы был разработан сценарий тестирования для роли пользователя предметной области.

В таблице 4 представлен сценарий для роли курьер.

Далее рассмотрим сами процессы и действия при тестировании информационной системы.

Сперва перед самым написанием тестов нужно было подготовить среду, для этого необходимо было создать дополнительную базу данных, чтобы при тестировании ни как 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

Поле	Описание
Дата теста	15.10.2023
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 35).

```

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 35 – Очистка коллекции

Далее рассмотрим файл с тестами функций авторизации (рисунок 36).

```

const expect = chai.expect;
chai.use(chaiHttp);

describe('Авторизация', () => {
  let user = {
    fullName: 'Name',
    phone: '88005553535',
    password: '12345'
  };
  let userRegistered = false;
  let token;

  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);

          // Авторизация
          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;
              token = res.body.token;
            });
        });
    }
    if((token && userRegistered){
      // Авторизация
      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();
  });

  // Проверка регистрации (пропала была в before())
  it('Регистрация существующего аккаунта', (done) => {
    chai.request(server)
      .post('/auth/register')
      .send(user)
      .end((err, res) => {
        expect(res.status).to.be.equal(409);
        expect(res.body).to.be.a('object');
        expect(res.body.msg).to.be.equal('Аккаунт с таким номером телефона уже существует');
        // 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);
        // let token = res.body.token;
        done();
      });
  });

  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 36 – Тесты функций авторизации

Первым делом мы создаем переменные для хранения токена и шаблона пользователя.

Затем создаем хук который будет постоянно срабатывать перед каждым тестом и проверять был ли проведен тест регистрации пользователя или нет (рисунок 37), 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 37 – Хук регистрации и авторизации

Далее мы тестируем, что регистрация уже была выполнена и при повторной регистрации ожидаем ошибку, это можно увидеть на рисунке 38.

```

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 38 – Тест повторной регистрации

Потом идут ещё два теста, один на авторизацию, а второй тестирует, что запросы для авторизованных пользователей будут проходить успешно (рисунок 39).

```

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 39 – Тест защищенного запроса

Кроме того, в рамках тестирования был создан чек-лист таблица 5.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

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 Документирование программного продукта

6.1 Руководство пользователя информационной системы

Информационная система «Магазин сладостей» имеет простой и интуитивно понятный интерфейс, что позволяет легко понять обычному пользователю, как с ней работать.

При вводе URL адреса в поисковую строку браузера перед пользователем открывается главная страница, которой является каталог продуктов (рисунок 12).

Далее пользователь выбирает нужную ему категорию и отправляется на страницу с товарами (рисунок 13). На этой странице пользователь может добавить товар в корзину, добавить в избранное, а если хочет узнать о нем подробную информацию, то при наведении появится кнопка подробнее, где при нажатие появится модальное окно (рисунок 32), в котором можно узнать подробную информацию, а также выбрать нужное количество товара.

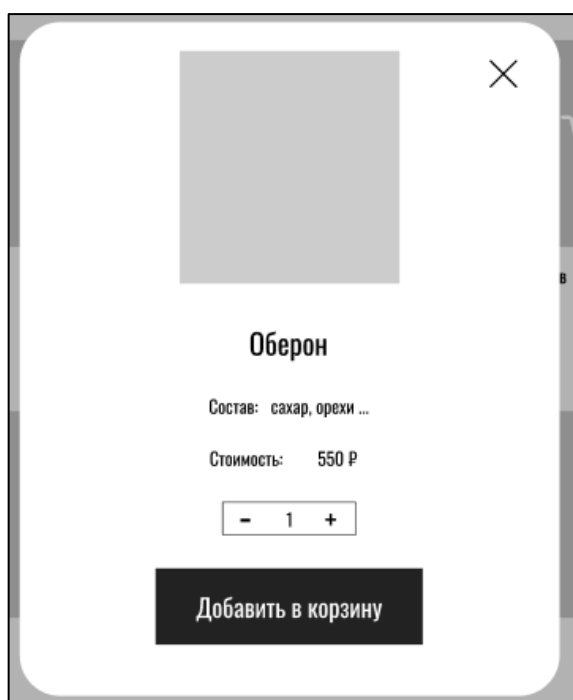


Рисунок 40 – Подробная информация о товаре

После того как все нужные товары были добавлены в корзину, пользователь может нажать на корзину в правом верхнем углу страницы (рисунок 41).



Рисунок 41 – Шапка сайта

Далее пользователь попадает в корзину (рисунок 14) и может в ней, при необходимости, удалить товар или изменить его количество.

Когда пользователь определился с заказом, пользователь приступает к оформлению заказа (рисунок 42), 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 42 – 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.

6.2 Руководство администратора информационной системы

Сначала нужно попасть на панель администратора, для этого нужно зайти или по URL адресу 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 43).



Рисунок 43 – Навигационная панель

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

Всего будет 5 разделов, о которых подробно написано снизу.

Первый раздел «Главная» на котором располагается статистика информационной системы 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 44.

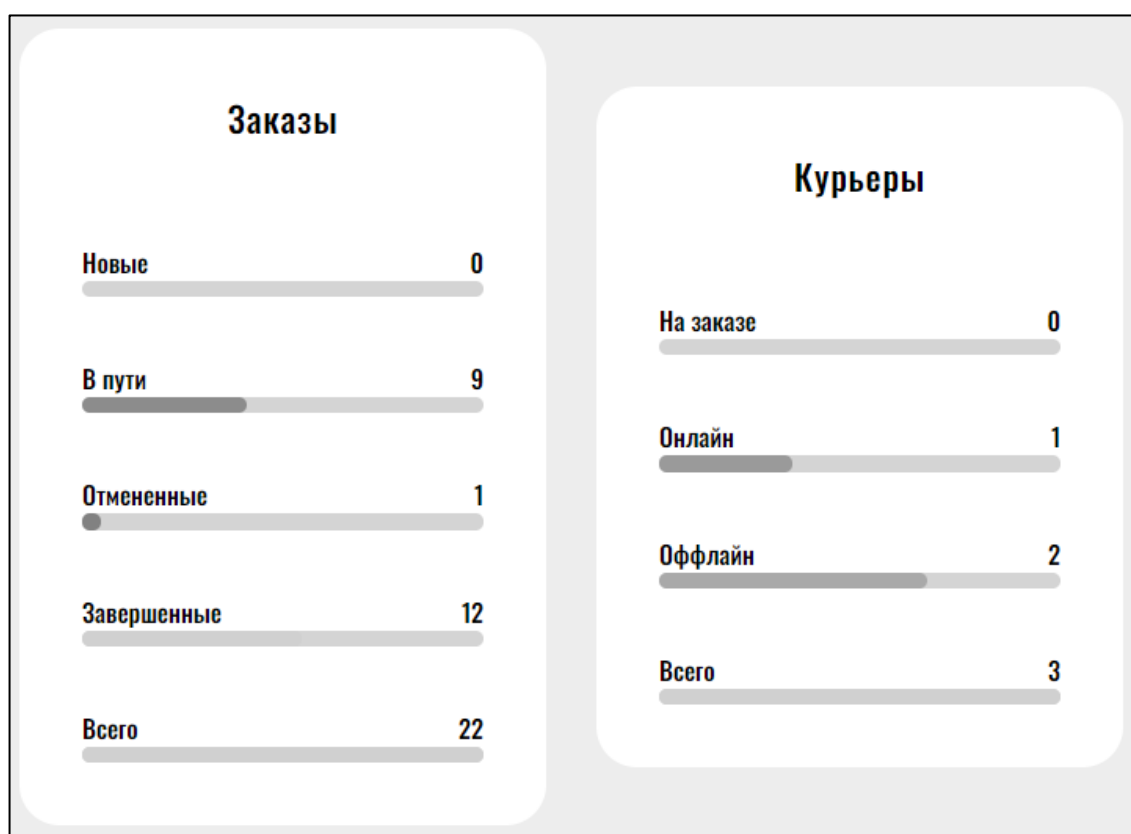


Рисунок 44 – Раздел «Главная»

Второй раздел «Заказ» дает возможность управления заказами (рисунок 45). 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 45 – Раздел «Заказы»

Третий раздел «Товары» позволяет создавать, редактировать и удалять товары. Сперва будет предложен выбор создать или редактировать (удалить) товар (рисунок 46). При выборе создания товара отобразится форма создания заказа и предварительная карточка товара (рисунок 47), 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 48, поиск можно производить как по названию товара, так и по составу.

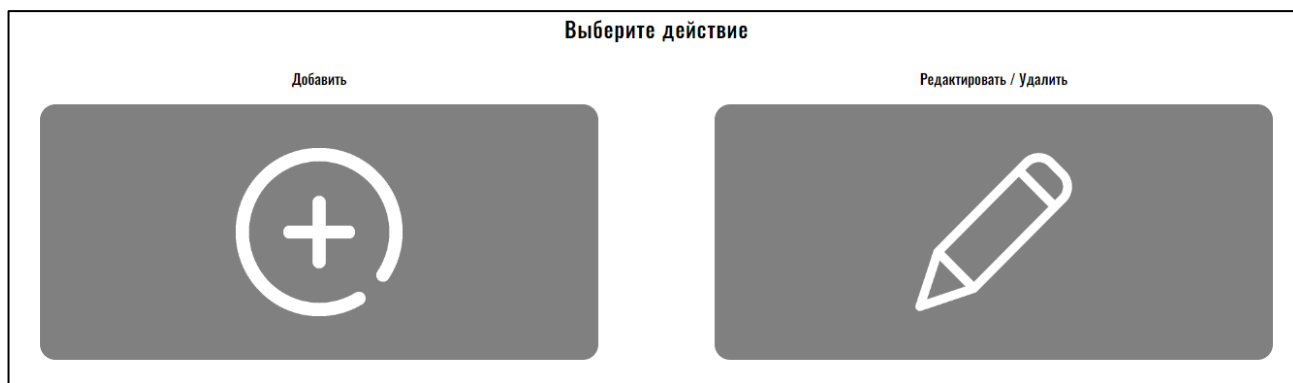


Рисунок 46 – Раздел «Товары»

Добавить товар

Торты

Конфеты

Мороженое

Десерты

Название:

Введите название

Цена:

Введите цену

руб.

Введите состав

Название

0 Р

Сохранить

Рисунок 47 – 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 48 – 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 49).

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 50), где нужно не забыть выбрать роль и записать данные аккаунта, а снизу будет отображаться предварительная карточка аккаунта.

Аккаунты

Создать

Все

Пользователи

Курьеры

Введите фιο или телефон

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 49 – Раздел «Аккаунты»

Создать аккаунт

Пользователь

Курьер

Введите имя

Введите телефон

Введите пароль

Пользователь

ФИО:

Телефон:

Сохранить

Рисунок 50 – Создание 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 51).

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 51 – Аккаунты курьеров

В пятый раздел «Поддержка» будут приходить обращения от курьеров и другого персонала. При отсутствии обращений раздел будет выглядеть как на рисунке 52.

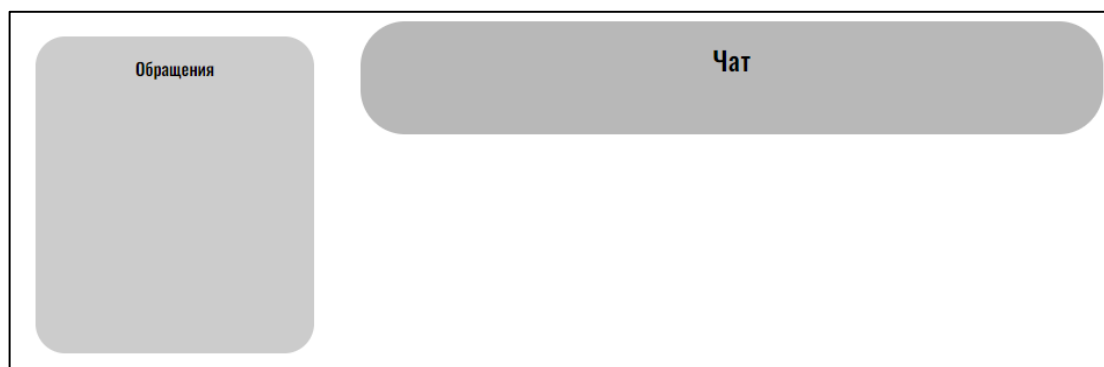


Рисунок 52 – Раздел «Поддержка»

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

При активных обращениях на боковой панели появится пометка, как на рисунке 53.

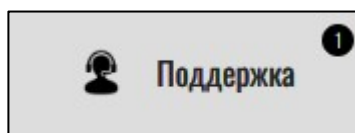


Рисунок 53 – Уведомление о количестве обращений

Если зайти в раздел «Поддержка» когда есть активные обращения и выбрать одно из них, то это будет выглядеть как на рисунке 54.

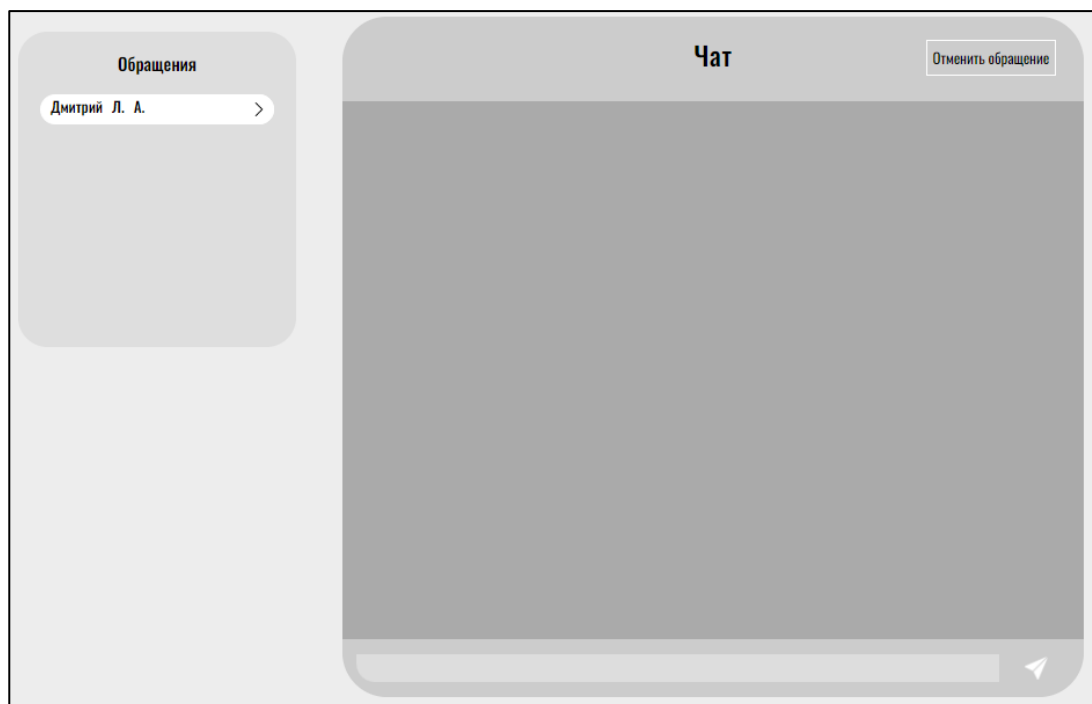


Рисунок 54 – Открытое обращение

Как видно на рисунке 54 при открытом обращении можно что-то написать или отменить обращение, ничего страшного если чат пустой, это значит, что пользователь ещё не написал текст обращения.

После того как пользователь напишет текст обращения, оно в реальном времени отобразится в окне чата (рисунок 55) и можно будет нажать на имя пользователя чтобы увидеть подробную информацию о пользователе (рисунок 56).

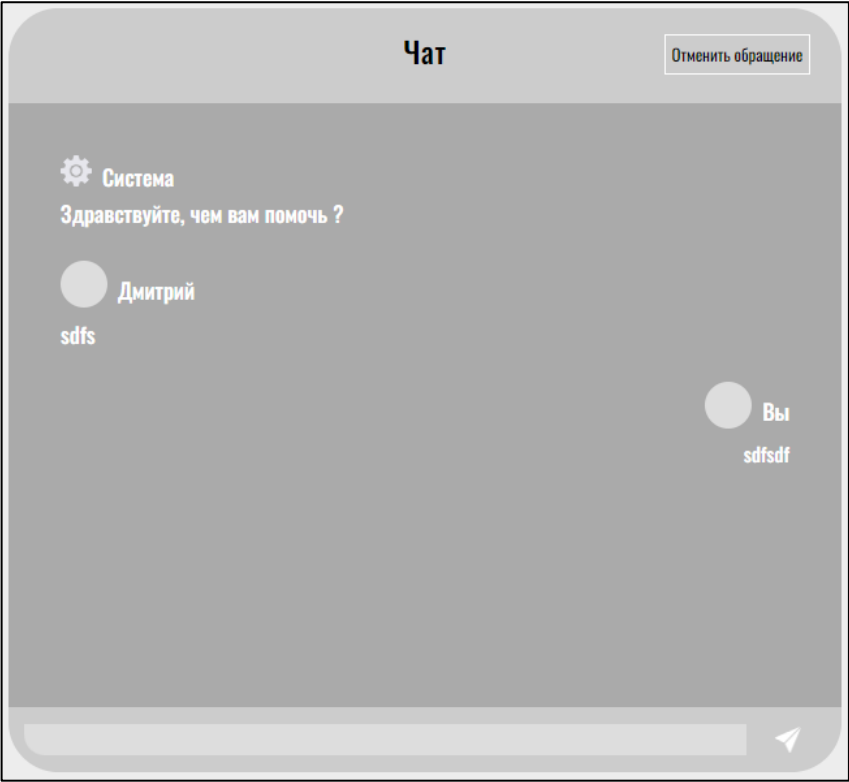


Рисунок 55 – Диалог с пользователем

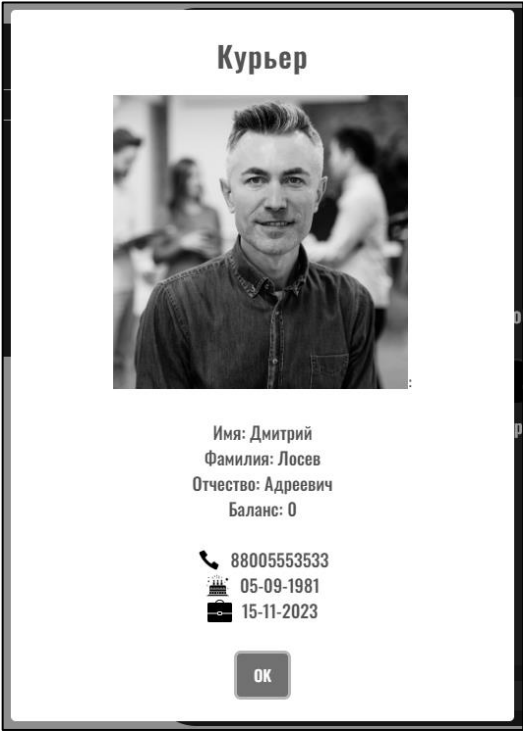


Рисунок 56 – Подробная информация о пользователе

6.3 Руководство пользователя для курьеров информационной системы

Сначала нужно перейти в личный кабинет, для этого нужно зайти или по URL адресу или нажать на кнопку «шестеренку» в шапке страницы.

Откроется окно авторизации, это можно увидеть на рисунке 57, и после успешной авторизации курьер попадет в личный кабинет.

Здесь для перемещения между разделами используется навигационная панель, которая располагается слева страницы (рисунок 58).



Авторизация

Введите номер телефона

Введите пароль

Войти

Рисунок 57 – Авторизация

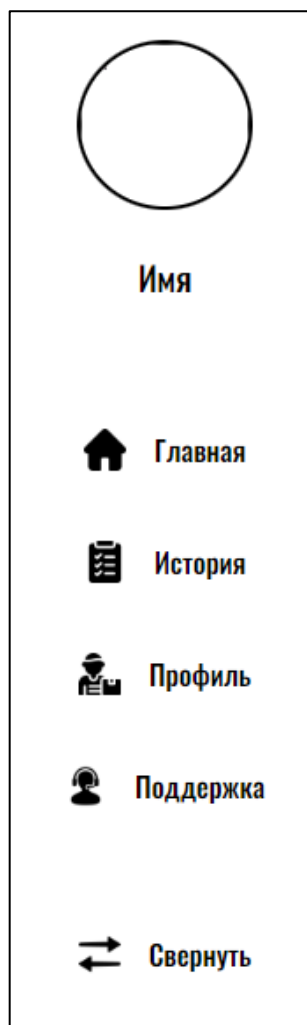


Рисунок 58 – Навигационная панель

Первый раздел «Главная» содержит в себе карту и список заказов для доставки (рисунок 59). На карте автоматически в реально времени ставятся точки для доставки и если на них нажать, то откроется окошко, в котором будет вся нужная информация о заказе (рисунок 60). Карту можно открыть в полноэкранный режим, а так же она может показать ваше текущее местоположение.

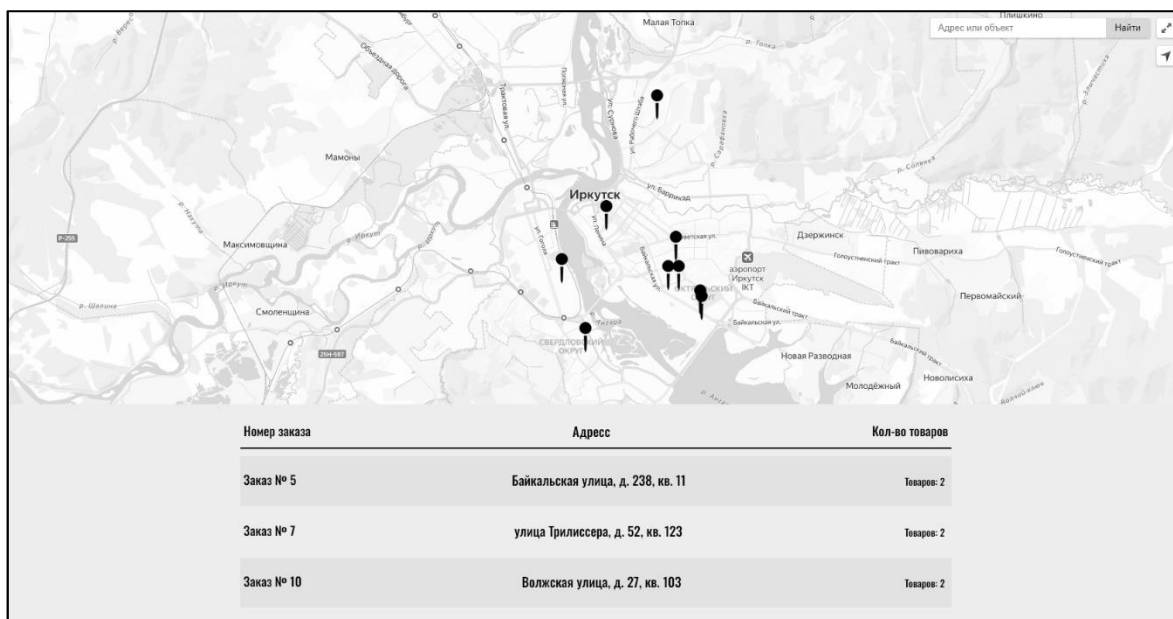


Рисунок 59 – Раздел «Главная»

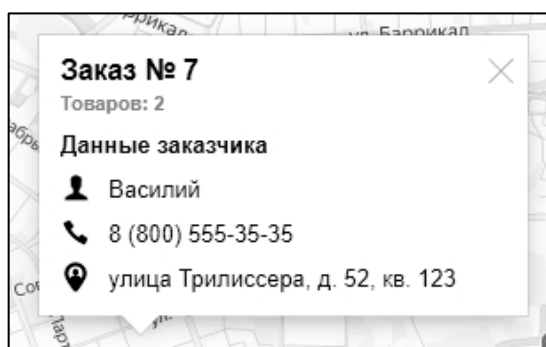


Рисунок 60 – Информация о заказе

Ниже карты находится список заказов и если нажать на один из них, от откроется информация о заказе, карта приблизится и будет показывать только метку выбранного заказа (рисунок 61). Так же появится две кнопки «принять» заказ и «крестик» чтобы вернуться к списку заказов. После принятия заказа, вместо кнопки «принять» отобразится две кнопки «завершить» и «отменить» и нельзя будет вернуться к списку заказов, пока есть активный заказ. После завершения заказа он будет сохранен в историю заказов.

✕



Заказ - № 5

👤 Василий

☎ 8 (800) 555-35-35

📍 Байкальская улица, д. 238, кв. 11

Товары

	Три шоколада	1 шт.
	Ягодный праздник	1 шт.

Принять

Рисунок 61 – Выбранный заказ

Во втором разделе «История» можно посмотреть историю выполненных заказов, а при нажатии на конкретный заказ, можно узнать подробную информацию о заказе.

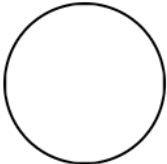
В третьем разделе «Профиль» можно посмотреть и изменить свои личные данные и фото (рисунок 62).

Баланс: 0 Р

Профиль

Редактировать

Активных заказов: 0 Выполненных заказов: 5



👤 **Имя:** Дмитрий **Фамилия:** Лосев **Отчество:** Адреевич

☎ **8 (800) 555-35-33**

🎂 **Дата рождения:** 05-09-1981

📅 **Дата трудоустройства:** 15-11-2023

Рисунок 62 – Профиль

В блоке профиля есть функции редактирования и обновления фото. Чтобы обновить фото достаточно навестись и нажать на фото, и после выбрать файл.

А чтобы редактировать личную информацию нужно нажать на кнопку справа сверху «Редактировать» и в открывшемся окне изменить имеющуюся информацию.

После изменений, для безопасности, нужно будет дополнительно ввести текущий пароль.

Последний четвертый раздел нужен для того, чтобы при каких-то неполадках или проблем во время выполнения заказа можно было обратиться к администраторам, данный раздел можно увидеть на рисунке 63.

В центре раздела находится блок чата, и чтобы создать обращение нужно нажать на центр этого блока.

После создания обращения нужно будет написать свою проблему и через некоторое время получить ответ от администрации.

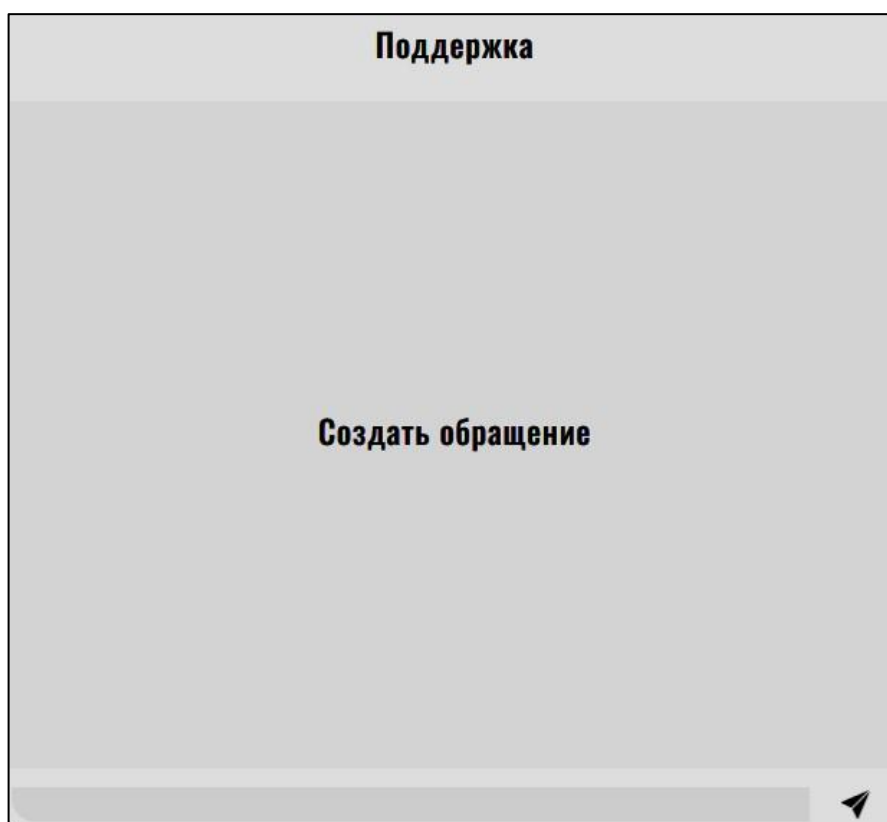


Рисунок 63 – Поддержка

ЗАКЛЮЧЕНИЕ

В ходе курсового проекта была поставлена задача разработать информационную систему «Магазин сладостей». Были определены требования для продукта. Исходя из требований продукта были рассмотрены возможные способы реализации информационной системы 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 таблицами базы данных, а также позволяет удобно хранить множество типов данных в одном документе коллекции.

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 курсовом проекте были выполнены все главные цели, разработана удобная и понятная для пользователей информационная система, которая подходит под все критерии технического задания и соответствует задумке самого курсового проекта.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Devhints.io – Справочник React – URL: <https://devhints.io/react> (дата обращения: 20.09.2023). – Текст: электронный.

2 Expressjs.com – Официальная документация Express.js – URL: <https://expressjs.com/ru/guide/routing.html> (дата обращения: 20.09.2023). – Текст: электронный.

3 GitHub.com – Официальная документация MERN – URL: <https://github.com/vasansr/pro-mern-stack> (дата обращения: 20.09.2023). – Текст: электронный.

4 HTML5CSS.ru – JavaScript учебник – URL: <https://html5css.ru/js/default.php> (дата обращения: 22.09.2023). – Текст: электронный.

5 JavaScript.ru – JavaScript учебник – URL: <https://learn.javascript.ru> (дата обращения: 22.09.2023). – Текст: электронный.

6 Mongodb.com – Официальная документация MongoDB – URL: <https://www.mongodb.com/docs/> (дата обращения: 23.09.2023). – Текст: электронный.

7 Mongoosejs.com – Официальная документация Mongoose – URL: <https://mongoosejs.com/docs/> (дата обращения: 23.09.2023). – Текст: электронный.

8 Nodejs.org – Официальная документация Node.js – URL: <https://nodejs.org/ru/docs> (дата обращения: 23.09.2023). – Текст: электронный.

9 React.dev – Официальная документация React – URL: react.dev/learn (дата обращения: 25.09.2023). – Текст: электронный.

10 Reactrouter.com – Официальная документация React-router – URL: <https://reactrouter.com/en/main/start/tutorial> (дата обращения: 26.09.2023). – Текст: электронный.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А – Техническое задание

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

ИНФОРМАЦИОННАЯ СИСТЕМА «МАГАЗИН СЛАДОСТЕЙ»

Руководитель: _____ (А.С.Александрова)
(подпись, дата)

Студент: _____ (В.О. Никифоров)
(подпись, дата)

Иркутск 2023

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

1 Общие сведения

Наименование работы: информационная система «Магазин сладостей».

Исполнитель: студент иркутского авиационного техникума, группы Веб-20-1 Никифоров В.О.

Разработка информационной системы проходит в рамках курсового проекта по МДК.05.02 Разработка кода информационных систем, на основании приказа № 39-у от 14 сентября 2023 года

Сроки разработки информационной системы с 14.09.2022 по 17.11.2023 года.

2 Назначение и цели создания информационной системы

Назначение информационной системы «Магазин сладостей» заключается в продаже сладостей. Для главного администратора организации необходимы такие функции, как добавление товара, изменение товаров и регистрация новых сотрудников. Для курьеров – принятие заказов, доставка клиенту.

3 Требования к информационной системе в целом

3.1 Требования к структуре и функционированию информационной системы

Функции информационной системы:

1. раздел «Сладости»;
- 1.1. отображение карточек товара;
- 1.2. добавить товар в корзину;
- 1.3. добавить товар в избранное;
- 1.4. управление товарами (для администратора);
- 1.4.1. редактирование товаров;
- 1.4.2. удаление товаров;

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

2. раздел «Корзина»;
 - 2.1. отображение информации о заказе;
 - 2.2. перечень выбранных товаров;
 - 2.3. удаление или добавление товара;
 - 2.4. выбор способа доставки и оплаты;
3. раздел «История заказов»;
 - 3.1. отображение всех заказов;
 - 3.2. отображение статуса заказов;
 - 3.3. повтор заказа;
 - 3.4. отмена заказа;
4. раздел «Панель администратора» (для администратора);
 - 4.1. просмотр статистики;
 - 4.2. подраздел «Заказы»;
 - 4.2.1. просмотр заказов;
 - 4.2.2. изменение статуса заказа;
 - 4.2.3. отмена заказа;
 - 4.3. подраздел «Товары»;
 - 4.3.1. поиск товара;
 - 4.3.2. создание товара;
 - 4.3.3. изменение товара;
 - 4.3.4. удаление товара;
 - 4.4. подраздел «Аккаунты»;
 - 4.4.1. создание аккаунта;
 - 4.4.2. изменение аккаунта;
 - 4.4.3. удаление аккаунта;
 - 4.5. подраздел «Поддержка»;
 - 4.5.1. просмотр обращений;
 - 4.5.2. просмотр автора обращения;
 - 4.5.3. отмена обращения;

- 5. раздел «Личный кабинет курьера» (для курьеров);
 - 5.1. подраздел «Главная»;
 - 5.1.1. просмотр заказов;
 - 5.1.2. принятие заказа;
 - 5.1.3. подтверждение выполнения заказа;
 - 5.1.4. отклонение заказа;
 - 5.2. подраздел «История»;
 - 5.2.1. просмотр истории заказов;
 - 5.3. подраздел «Профиль»;
 - 5.3.1. просмотр профиля;
 - 5.3.2. редактирование профиля;
 - 5.3.3. обновление фото;
 - 5.4. подраздел «Поддержка»;
 - 5.4.1. создание обращения;
 - 5.4.2. просмотр обращения;
 - 5.4.3. отмена обращения.

3.2 Требования к надежности

Для обеспечения надежности необходимо проверять корректность получаемых данных и реализовать валидность полей. Входные данные поступают в виде значений с клавиатуры. Эти значения отображаются в отдельных полях таблицы.

3.3 Требования к безопасности

Для обеспечения безопасности в информационной системе необходимо реализовать разграничение прав доступа, возможность восстановить данные.

3.4 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов информационной системы

Минимальные системные требования для рабочей станции:

- 1) Процессор: Intel Pentium 4 2.0Ghz / AMD XP 2200+;
- 2) Оперативная память: 4 ГБ;
- 3) Жёсткий диск: 64 ГБ;
- 4) Операционная система: Windows, Linux, MacOS.

Минимальные системные требования для сервера:

- 1) Процессор: Intel Core i3 2100CPU / AMD FX-6300;
- 2) Оперативная память: 4Гб;
- 3) Жёсткий диск: 24Гб
- 4) Операционная система: Windows, Linux, MacOS.

4 Требования к документированию

Основным документам, регламентирующими использование информационной системой является руководство пользователя.

Основным документам, регламентирующими разработку информационной системы является техническое задание.

5 Состав и содержание работ по созданию информационной системы

В таблице 1 представлены плановые сроки начала и окончания работы по созданию информационной системы.

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						66
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 1 – Плановые сроки по созданию информационной системы

№ п/п	Этап разработки ИС	Сроки выполнения	
		Начало	Окончание
1	Предпроектное исследование предметной области	21.09.23	26.09.23
2	Разработка технического задания	26.09.23	10.10.23
3	Проектирование программного обеспечения	10.10.23	10.11.23
4	Оформление пунктов пояснительной записки	10.11.23	15.11.23
5	Разработка и отладка программного продукта	15.11.23	17.11.23
6	Составление программной документации	17.11.23	20.11.23

Приложение Б – Листинг

```
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';
dotenv.config()
mongoose.connect(process.env.DB).then(() => console.log('DB OK'))
.catch((err) => console.log('DB ERROR', err));
const PORT = process.env.PORT
const HOST = process.env.HOST
const app = express();
app.use(express.json());
app.use(cors({ origin: '*' }));
app.use('/', routes);
const server = app.listen(PORT, HOST, (err) => {
  if (err) {
    return console.log(err);
  }
  console.log(`Server running at http://${HOST}:${PORT}/`);
});
Socket(server);
export default server
```

					КП.09.02.07-3.23.201.14. ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		