

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Клиент-серверное приложение для
отслеживания работы электронной коммерции**

Студент Шеянов Николай Николаевич (ФИО полностью)
14.06.19 (подпись, дата)

Руководитель

Трокоз Д.А.
(фамилия, инициалы)

Нормоконтролёр

Кучин А.В. 17.06.19
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 31.05.2019 г. № 38-09-14)

Заведующий кафедрой

Митрохин М.А.
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 19.06.2019, № 5)

Секретарь ГЭК

Гурин Е.И.
(фамилия, инициалы)

Пенза, 2019

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

«__» _____ 20__ г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Шеянов Николай Николаевич гр. 15ВВ1 факультета вычислительной техники

2. Тема работы: «Клиент-серверное приложение для отслеживания работы электронной коммерции»

Тема утверждена приказом ПГУ № 554/0 от "29" апреля 2019г.

3. Руководитель бакалаврской работы к.т.н., доцент каф. ВТ ПГУ Трокоз Д.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки - продажа кондитерских изделий посредством технологий интернет. Система должна предоставлять информацию о фирме, обеспечивать работу с товарами, разграничивать уровень прав доступа пользователей на работников и покупателей, осуществлять работу по оформлению покупок и управлению заказами.

5. Перечень подлежащих разработке вопросов изучения и анализ технологий для разработки клиент-серверного приложения, таких как: Spring Framework, Java, PostgreSQL, pgAdmin4, IntelliJ IDEA, Hibernate, Spring Security, Vue.js, Insomnia, HTML, CSS, JavaScript и SQL.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	10	29.04.2019 - 2.05.2019	<i>Дефюгу</i>
2.	Анализ требований	10	3.05.2019 - 5.05.2019	<i>Дефюгу</i>
3.	Выбор технологий, применяемых при проектировании и разработке	10	6.05.2019 - 8.05.2019	<i>Дефюгу</i>
4.	Разработка БД	15	9.05.2019 - 14.05.2019	<i>Дефюгу</i>
5.	Разработка backend части	20	15.05.2019 - 21.05.2019	<i>Дефюгу</i>
6.	Разработка frontend части	25	22.05.2019 - 28.05.2019	<i>Дефюгу</i>
7.	Оформление пояснительной записки	10	29.05.2019 - 31.05.2019	<i>Дефюгу</i>

Дата выдачи задания "29" апреля 2019г.

Руководитель бакалаврской работы *Дефюгу* Трокоз Д.А.
(подпись, дата)

Задание к исполнению принял студент *Шеянов Н.Н.* 29.04.2019
(подпись, дата)

Работу к защите допустить
Декан факультета *Фионова Л.Р.* 12.06.19
(подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 98 листов, 18 рисунков, 5 таблиц, 16 источников, 4 приложения.

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ ДЛЯ ОТСЛЕЖИВАНИЯ РАБОТЫ ЭЛЕКТРОННОЙ КОММЕРЦИИ, SPRING FRAMEWORK, INTELLIJ IDEA, INSOMNIA, SPRING SECUTITY, HIBERNATE, POSTGRESQL, PGADMIN4, HTML, CSS, JAVA, JAVASCRIPT.

Целью проекта является разработка клиент-серверного приложения для отслеживания работы электронной коммерции фирмы «Торт на заказ «Крендель - Пенза», занимающейся продажей различных кондитерских изделий на заказ. Система должна описывать полную информацию о данной фирме, обеспечивать покупателю осуществление покупки посредством технологии Интернет. Продукт также предоставляет различные уровни доступа к функциям, а также к информации, имеющейся в системе.

В результате проведенной работы создан проект, удовлетворяющий поставленной задаче. А именно, разработано, реализовано и протестировано приложение для отслеживания работы электронной коммерции фирмы «Крендель - Пенза».

Разработка производилась с использованием большого стека технологий и языков программирования, к которым относятся: язык программирования Java, Java Script, язык структурированных запросов SQL, Spring Framework, Spring Security, Hibernate, язык гипертекстовой разметки HTML, язык каскадных таблиц стилей CSS. Для хранения данных выбрана СУБД PostgreSQL, а для её администрирования применялся инструмент pgAdmin4.

					ПГУ 1.09.03.01.29.001 ПЗ				
Изм.	Лист	№ докум	Подпись	Дата					
Разраб.		Шеянов Н.Н.			Клиент-серверное приложение для отслеживания работы электронной коммерции Пояснительная записка	Лит.	Лист	Листов	
Провер.		Трокоз Д.А.					4	98	
						ФВТ гр. 15ВВ1			
Н. Контр.		Кучин А.В.							
Утв.		Митрохин М.А.							

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ	6
ВВЕДЕНИЕ	7
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	9
1.1 Анализ архитектуры системы электронной коммерции	9
1.2 Анализ функциональных блоков системы электронной коммерции	10
2 ТРЕБОВАНИЯ К РАЗРАБОТКЕ ПРИЛОЖЕНИЯ ДЛЯ ОТСЛЕЖИВАНИЯ РАБОТЫ ЭЛЕКТРОННОЙ КОММЕРЦИИ	13
3 ВЫБОР ТЕХНОЛОГИЙ, ЯЗЫКОВ И СРЕДСТВ ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ	15
3.1 Программные средства, используемые при разработке системы	15
3.2 PostgreSQL	16
3.3 pgAdmin 4	17
3.4 IntelliJ IDEA	17
3.5 Spring Framework	17
3.6 Apache Tomcat и Apache Maven	18
3.7 Hibernate	19
3.8 Spring Security	19
3.9 Vue.js	20
3.10 Insomnia	20
3.11 Java	21
3.12 HTML	22
3.13 CSS	22
3.14 JavaScript	23
3.15 SQL	23
3.16 REST API	24
3.17 Взаимодействие рассмотренных средств	25
4 РЕАЛИЗАЦИЯ	27
4.1 Алгоритм работы системы электронной коммерции	27
4.2 Описание структуры и таблиц БД	29
4.3 Основные функции доступные пользователям	34
4.4 Схема оформления заказа	39
4.5 Схема обработки заказа	41
5 МЕТОДЫ ИСПЫТАНИЙ	42
5.1 Регистрация пользователя и вход в профиль	42
5.2 Каталог и корзина	45
ЗАКЛЮЧЕНИЕ	48
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	49
ПРИЛОЖЕНИЕ А	50
ПРИЛОЖЕНИЕ Б	56
ПРИЛОЖЕНИЕ В	90
ПРИЛОЖЕНИЕ Г	97

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

БД - база данных

СУБД - система управления базой данных

ORM - (англ. Object-relational Mapping) - объектно-реляционное отображение

ПО - программное обеспечение

ОС - операционная система

ЯП - язык программирования

CSS - (англ. Cascading Style Sheets) - каскадные таблицы стилей

JS - (англ. JavaScript) - мультипарадигменный язык программирования

JVM - (англ. Java Virtual Machine) виртуальная машина Java

GUI - (англ. Graphical User Interface) - графический интерфейс пользователя

HTML - (англ. HyperTextMarkupLanguage) - язык гипертекстовой разметки

REST - (англ. Representational State Transfer) - передача состояния представления

SQL - (англ. Structured Query Language) - язык структурированных запросов

URL - (англ. Uniform Resource Locator) - единый указатель ресурса

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

ВВЕДЕНИЕ

В настоящее время в России и по всему миру широко развито такое явление, как интернет-торговля. Под интернет-торговлей представляют процесс реализации физических, а также цифровых товаров с использованием специализированных электронных площадок, позволяющих осуществлять заказ из любой точки планеты, где есть Интернет. Постоянно возрастающая популярность интернет-торговли во многом зависит от общего состояния доступности технологий для населения, а также от уровня информационной грамотности.

В современном обществе, где так сильно развиты информационные технологии, каждая фирма, осуществляющая продажу каких-либо товаров или услуг, должна иметь свой собственный сайт, который позволяет легче рекламировать магазин и осуществлять его всестороннюю информационную поддержку. Электронные магазины не вытесняют традиционные, а расширяют сферу и рынок сбыта. В определенных сегментах торговли отсутствие онлайн-магазина является стратегическим упущением. В дополнение к этому сайт магазина является визитной карточкой и справочником товаров и услуг в сети Интернет, дает возможность аудитории быстро и просто получить информацию, расширяет потенциальную базу клиентов и инвесторов и др.

Следовательно, разработка и обслуживание сайтов для фирм оказывается весьма прибыльной и актуальной сферой деятельности. Для осуществления бизнеса интернет-магазины являются удобной площадкой, чтобы рекламировать и продвигать товары и услуги, что, несомненно, увеличивает объем продаж, а также ведёт к росту фирмы.

Интернет-магазины позволяют покупателям экономить свое время, дают возможность осуществить выбор нужного товара. Существуют удобные системы поиска и сравнения по интересующим характеристикам. Хорошо исполненный сайт электронной коммерции не только позволит избежать

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

очереди в магазинах, но и предоставит удобную систему навигацию, поиска, оплаты и доставки, а также связи с производителем и руководством магазина по поводу жалоб и предложений. Все эти факторы говорят о том, что зачастую интернет-магазин может быть не только наравне, но и гораздо удобнее, чем традиционный магазин.

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

При разработке программного продукта необходимо провести анализ существующих вариантов решения задач. В данном разделе рассматриваются общие теоретические вопросы, связанные с проектированием и реализацией клиент-серверного приложения для отслеживания работы электронной коммерции.

Для того чтобы создать программный продукт с заданными параметрами, следует рассмотреть его архитектуру и функциональные блоки, без которых приложение не сможет корректно работать.

1.1 Анализ архитектуры системы электронной коммерции

Архитектура интернет-магазина является трехзвенной архитектурой, которая состоит из трех основных частей:

1. Клиентская часть
2. Серверная часть
3. База данных

Рассмотрим конкретнее, что представляет каждая часть, какова роль каждой в проекте.

Клиентская часть - непосредственно та часть, с которой взаимодействует пользователь. Она представляет собой графический интерфейс, компоненты, которые выводит браузер на страницу в качестве ответов на выполненные запросы от сервера. Взаимодействие пользователя и веб-приложения происходит именно через браузер, с помощью нажатия на кнопки или же перехода по ссылкам.

Серверная часть - программа, которая принимает и обрабатывает запросы от пользователей. Серверную часть можно реализовать с помощью различных технологий и языков программирования. Процесс взаимодействия с клиентской частью состоит из получения запроса от клиента, обработка его

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

на сервере и отправка результата данного запроса обратно на клиентскую сторону.

База данных (БД) - совокупность взаимосвязанных, хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

Система управления базами данных (СУБД) - программное обеспечение, необходимое для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации. При принятии запроса сервером на взаимодействие с какими-либо данными из БД, он в свою очередь посылает SQL-запрос СУБД, где происходит соответствующая операция по добавлению, модификации, получению или удалению данных. После принятия сервером ответа от БД, он посылает результат запроса обратно на клиентскую часть.

1.2 Анализ функциональных блоков системы электронной коммерции

Ни один интернет-магазин не может обойтись без некоторых функциональных блоков, то есть они являются обязательными для любого сайта. К таким важным частям относятся:

- форма регистрации нового пользователя;
- профиль пользователя с данными о нем;
- система поиска товаров;
- каталог товаров;
- форма по добавлению товара;
- пользовательская корзина;
- форма оформления заказа.

Форма регистрации нового пользователя служит для того, чтобы создать личную учетную запись на web-ресурсе, с целью получения доступа к его полному функционалу. Учетная запись хранит информацию о главных

характеристиках клиента, таких как: фамилия, имя, отчество, телефон, e-mail, адрес и др. Данная информация необходима для того, чтобы осуществить заказ в интернет-магазине, а также иметь возможность связаться с администрацией магазина. Данная информация хранится на стороне сервера.

Профиль пользователя с данными о нем необходим, если пользователь захочет изменить какую-либо персональную информацию, например, номер телефона, адрес или электронную почту. Дополнительно профиль пользователя позволяет просмотреть ранее сделанные заказы.

Система поиска товаров - система, которая служит для удобного и быстрого поиска товара по заданным характеристикам. Она также позволяет пользователю убедиться присутствует ли товар в каталоге сайта. Является важной составной частью динамического каталога и располагается на стороне сервера.

Каталог товаров - список товаров, содержащий их полное описание. Он служит пользователю для удобного просмотра товаров, которые предоставляет данный интернет-магазин, а также предлагает ознакомиться с описанием, изображением и другими характеристиками товара.

Форма по добавлению товара необходима для осуществления добавления нового товара в каталог товаров сайта. Должна осуществлять ввод всех требуемых данных о товаре, таких как: название, код товара, цена, описание и изображение.

Пользовательская корзина содержит в себе список товаров, который пользователь добавил, предполагая его дальнейшее приобретение. Содержимое корзины должно отображаться и храниться постоянно. В любой момент времени покупатель может убрать товар из корзины, в случае если он расхотел его покупать.

Форма оформления заказа служит для осуществления непосредственной сделки с интернет-магазином. Форма предоставляет данные обо всех товарах, готовых к приобретению, их количестве и стоимости, а также выводит итоговую сумму за покупку всех товаров в

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

списке. Результат заказа хранится на сервере и является отчетностью для администрации магазина, а также позволяет пользователю в дальнейшем видеть список всех своих заказов в своем профиле.

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

2 ТРЕБОВАНИЯ К РАЗРАБОТКЕ ПРИЛОЖЕНИЯ ДЛЯ ОТСЛЕЖИВАНИЯ РАБОТЫ ЭЛЕКТРОННОЙ КОММЕРЦИИ

Система электронной коммерции должна выполнять список основных функций для осуществления процесса интернет-торговли, к ним относятся:

- со стороны фирмы:

- предоставление полной информации о фирме и её направленности, контактах и режиме работы, товарах и услугах и их характеристиках;
- обработка заказов, хранение данных о заказах.

- со стороны клиента:

- регистрация на сайте и хранение данных о пользователе и заказах пользователя;
- осуществление функции просмотра товаров, возможность их добавления в корзину и дальнейшее приобретение;
- просмотр контактной информации о фирме.

Для администрирования и пользования сайтом пользователям присвоены роли, такие как: покупатель и работник магазина.

При входе на сайт интернет-магазина пользователь имеет минимальные права и ему доступен просмотр информации, расположенной на сайте. После выполнения операции регистрации, где пользователь указывает свои персональные данные, он может войти на сайт как зарегистрированный пользователь и получить статус авторизованный. С этого момента покупатель получает все права по изучению информации, представленной на страницах интернет-магазина, а также по использованию всех функций представленных для пользователя категории "User".

Работник магазина после авторизации имеет возможность работать над содержимым магазина, будь то изменение какой - либо информации, просмотр и сбор статистики по заказам или же расширение ассортимента товаров, а также обслуживания заказов. Подразумевается, что работников

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

интернет-магазина будет несколько, так как в обслуживании такого ресурса необходимо много времени и сил.

Клиент-серверное приложение позволяет улучшить работу фирмы за счет упрощения работы и взаимодействия с покупателем, возможности работы из любого места, автоматизации процессов обработки заказов. Клиентам позволяет экономить время при поиске и покупке интересующего их товара. Приложение должно иметь простой интуитивно понятный интерфейс для работы, осуществлять правильную и быструю обработку запросов, хранить большое количество данных о товарах, пользователях и заказах в рамках одной базы данных.

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

3 ВЫБОР ТЕХНОЛОГИЙ, ЯЗЫКОВ И СРЕДСТВ ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

3.1 Программные средства, используемые при разработке системы

Для реализации программного продукта «Клиент-серверное приложение для отслеживания работы электронной коммерции» были рассмотрены следующие технологии, языки и архитектуры:

- PostgreSQL - СУБД;
- pgAdmin 4 -инструмент администрирования СУБД PostgreSQL;
- IntelliJ IDEA - интегрированная среда разработки;
- Spring Framework - универсальный фреймворк с открытым исходным кодом для Java-платформы;
- Apache Tomcat - встроенный в фреймворк Spring веб-сервер для запуска веб-приложений;
- Apache Maven - фреймворк для автоматизации сборки проектов;
- Hibernate - библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения;
- Spring Security - фреймворк аутентификации и контроля доступа;
- Vue.js - JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов.
- Insomnia - инструмент тестирования REST API.

Языки программирования, использованные при разработке приложения:

- Java - сильно типизированный объектно-ориентированный язык программирования;
- HTML - стандартизированный язык разметки документов;
- CSS - формальный язык описания внешнего вида документа, написанного с использованием языка разметки;
- Java Script - мультипарадигменный язык программирования.

Архитектура:

- REST – это архитектурный стиль взаимодействия компонентов распределенного веб-приложения.

Основными факторами выбора именно этих средств были бесплатное использование, функциональные возможности, популярность, высокая производительность, наличие документации на русском и английском языках.

Компоненты, приведенные выше, образуют необходимый стек технологий для решения поставленной задачи. Все средства были выбраны исходя из отличного взаимодействия между собой, актуальности, наличия большого количества документации и примеров для быстрого обучения. Это позволяет решить достаточно сложные задачи.

3.2 PostgreSQL

PostgreSQL- универсальная объектно-реляционная СУБД, является самой лучшей системой баз данных с открытым исходным кодом. PostgreSQL разработан для работы на UNIX-подобных платформах, но, в свою очередь, она является портативной системой и может работать также и на других платформах как: Mac OS X, Solaris и Windows. Преимущества СУБД Postgre состоят в том, что это бесплатная система с открытым исходным кодом, отличной документацией с примерами использования предложенных технологий, а также имеет большое количество руководств по использованию от других выдающихся программистов. PostgreSQL достаточно легка в освоении и позволяет быстро и просто начать создание своей собственной БД.

Основные функции PostgreSQL

PostgreSQL содержит большое количество полезных функций, которые облегчают и ускоряют работу с БД, например:

- типы, определяемые пользователем;
- наследование таблиц;
- ссылочная целостность внешних ключей;

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16

- представления, правила и подзапросы;
- вложенные транзакции.

3.3 pgAdmin 4

Инструмент для управления СУБД PostgreSQL pgAdmin 4 является невероятно удобным приложением для администрирования и управления базами данных. Эти возможности представлены за счет удобного пользовательского интерфейса, упрощения множества операций, благодаря созданию грамотной системы управления.

3.4 IntelliJ IDEA

IntelliJ IDEA - программный продукт, разработанный компанией JetBrains, представляет из себя интегрированную среду разработки ПО для многих языков программирования, включая Java и JavaScript.

Среда позволяет оптимизировать работу программиста, предоставляя ему большее время на решение непосредственно функциональных задач, в то время как сама система выполняет большинство однообразных рутинных операций. IntelliJ IDEA предоставляет огромный инструментарий по работе по созданию приложений. Данная среда имеет хорошую совместимость со многими свободными инструментами для разработчиков, например Maven, Apache Tomcat и другие.

3.5 Spring Framework

Spring Framework - универсальный фреймворк с открытым исходным кодом для Java-платформы. Данный фреймворк предоставляет огромные возможности по разработке различных приложений Java-разработчикам.

Хорошо документированные и легкие в использовании средства разработки позволяют создавать программный продукт быстрее в сравнении с аналогичными фреймворками. Spring Framework осуществляет поддержку огромного количества расширений и усовершенствований для разработки веб-приложений на Java Enterprise платформе, что позволяет разработчикам справиться с огромным спектром задач, возникающими при создании

приложения. Также считается, что Spring реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java. Из-за широкой функциональности сложно выделить самые главные блоки, лучше представлять Spring как удобный набор инструментов, который можно подстроить под любой проект и любого разработчика, что позволит оптимизировать процесс разработки, улучшить качество продукта и затратить при этом минимальное количество ресурсов. Все эти доводы позволяют судить о популярности данного фреймворка и постоянном росте его сообщества.

3.6 Apache Tomcat и Apache Maven

Apache Tomcat является самостоятельным веб-сервером с открытым программным кодом, разработанным Apache Software Foundation(ASF).

Apache Maven - это фреймворк для автоматической сборки Java проекта. Он позволяет автоматизировать такие процессы как: компиляция, создание jar и др. Для того чтобы упростить данные операции используются специальные инструменты сборки проектов. Но в простейшем случае проект можно собрать и в командной строке.

Основными преимуществами Maven являются:

- независимость от OS. Проект можно собрать на любой ОС, при чем файл проекта будет всего один;

- управление зависимостями. В большинстве случаев для создания приложения используется множество сторонних библиотек, а в них могут содержаться указания на другие библиотеки. Чтобы в данной ситуации все работало корректно и существует управление версиями, которое позволяет осуществлять управление сложными зависимостями и разрешать конфликты версий, если в этом будет необходимость;

- хорошая интеграция со средами разработки. Наиболее часто используемые среды разработки (IDE) на языке программирования Java без каких-либо проблем могут открыть проекты, собранные с помощью Maven, в большинстве случаев данные проекты даже не требуют дополнительной

настройки и сразу готовы к дальнейшей разработке. Из этого следует, что если проект Maven разрабатывают в различных средах разработки, то количество ошибок и дублирования данных будет гораздо меньше, так как существует удобный способ хранения настроек.

3.7 Hibernate

Hibernate ORM - это инструмент объектно-реляционного отображения для языка программирования Java. Он представляет собой фреймворк для работы с сопоставлением между объектно-ориентированной доменной моделью и реляционной базой данных.

Hibernate решает проблемы несоответствия объектно-реляционного сопоставления, заменяя постоянные обращения к базе данных функциями обработки объектов высокого уровня.

Главной функцией Hibernate является сопоставление Java классов с таблицами базы данных и сопоставление типов данных Java с типами данных SQL. Hibernate также предоставляет возможность запроса и поиска данных. В таком случае, он автоматизирует генерацию SQL-запросов и освобождает разработчика от значительного объёма сравнительно низкоуровневого программирования при работе в объектно-ориентированных средствах в реляционной базе данных.

3.8 Spring Security

Spring Security является невероятно мощным и гибко настраиваемым фреймворком для обеспечения аутентификации и контроля доступа. Он представляет из себя основной стандарт по защите приложений, разработанных на Spring.

Spring Security - это платформа, которая фокусируется на предоставлении механизмов аутентификации и авторизации пользователей для приложений, разработанных с помощью языка Java. Главная особенность

Spring Security, как и других проектов Spring, заключается в том, насколько легко расширить инструментарий, чтобы соответствовать

необходимым требованиям на разработку.

В Spring Security имеются следующие особенности:

- всесторонняя и расширяемая поддержка аутентификации и авторизации;
- защита от атак, таких как фиксация сеанса, кликджекинг, межсайтовая подделка запросов и т. д;
- интеграция API сервлетов;
- дополнительная интеграция с Spring Web MVC;
- и другие.

3.9 Vue.js

Vue - современный и прогрессивный фреймворк, который позволяет разработчикам с меньшими затратами ресурсов и большим удобством осуществлять создание пользовательских интерфейсов. Он в первую очередь позволяет решить задачи уровня представления, данная особенность упрощает взаимодействие с другими библиотеками и уже готовыми проектами. Vue подходит для разработки огромного спектра приложений, что делает его востребованным на рынке. Из этого следует, что знание технологий данного фреймворка и механизмов работы с ним необходимо, чтобы быть конкурентоспособным на рынке.

3.10 Insomnia

Insomnia является одним из лучших REST API клиентов. Данный продукт позволяет быстро и просто протестировать запросы на взаимодействие клиента с сервером, такие основные запросы как: GET (запрос на получение каких-либо ресурсов), POST (запрос на добавление ресурсов), PUT (запрос на модификацию ресурсов) и DELETE (запрос на удаление ресурсов). Благодаря удобному пользовательскому интерфейсу и хорошему функционалу, работать с данным приложением удобно, это экономит большое количество времени и позволяет протестировать необходимые нам запросы на правильность их выполнения.

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

3.11 Java

Java - в современном мире Java является одним из самых известных и используемых объектно-ориентированных языков программирования. Данный ЯП разрабатывался как универсальный для решения различного горизонта задач. Java прошел огромный путь развития, в текущий момент является уже не просто ЯП, а целой платформой, которая объединяет огромное количество различных технологий, библиотек и т.д. Все это позволяет вести разработку на Java серьезных enterprise приложений, а большое количество дополнительных модулей дает возможность подобрать конкретные средства для решения поставленной задачи. Многие разработчики выбирают Java, в результате чего растет количество различных решений для возникающих вопросов, а также увеличивается объем литературы, доступной для изучения.

К особенностям Java относятся:

- Одной из главных особенностей языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. Затем происходит выполнения данного кода специальной виртуальной машиной Java. Данный принцип отличается от других интерпретируемых языков, таких как PHP или Perl, с другой стороны Java не является чисто компилируемым языком как C или C++;

- архитектура языка обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, из этого следует, что программам не нужно перекомпилироваться, чтобы быть используемыми на других платформах, таких как: Windows, Linux, Mac Os и других.

- Java является языком с Си-подобным синтаксисом и близок в этом отношении к C/C++ и C#. Данная особенность позволяет разработчикам, уже имеющих опыт работы с этими языками, быстро приступить к разработке приложений на Java.

- в Java присутствует автоматический сборщик мусора (англ. Garbage Collector), это позволяет разработчику не уделять времени на ручную

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

очистку памяти от ранее использованных объектов, что значительно упрощает и ускоряет работу.

- Java является объектно-ориентированным языком. Объектно-ориентированная модель содержит основные принципы: полиморфизм, наследование, инкапсуляцию. Это позволяет разработчикам на Java создавать сложные, но при этом гибкие, хорошо масштабируемые и расширяемые приложения.

3.12 HTML

Для создания интернет страниц используется специальный язык HTML, он представляет собой язык разметки документа, который иллюстрирует форму отображения данных на мониторе. Языком описывается отображение всевозможных элементов, а также поведение веб-страницы в браузере при взаимодействии с клиентом. Одной из особенностей языка HTML является наличие в его инструментарии огромного количества тегов. Теги - особые инструкции, которые позволяют формировать структуру и описывать оформление гипертекстового документа до самых мельчайших подробностей. Данный язык прост в обучении и имеет огромное количество примеров, что делает его удобным для изучения и использования.

3.13 CSS

CSS - каскадные таблицы стилей предназначены для более глубокого и детального описания внешнего вида разрабатываемого документа, который написан с помощью языка HTML. Данный язык значительно облегчил задачу по созданию и улучшению дизайна документа. Он позволяет:

- определить размер, начертание, цвет и другие параметры шрифта, который будет использоваться для вывода текста.
- расположить блок информации в нужном месте веб-страницы.
- оформить блок информации фоновой картинкой, границей и другими визуальными элементами.

Основные преимущества использования CSS заключаются в:

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

- написании меньшего количества кода, что упрощает работу программиста, код быстрее загружается, а также лучше оптимизирован;
- код пишется в виде модулей, что позволяет использовать один и тот же код многократно, это значительно сокращает время по выполнению однотипных задач, а также способствует общему стилю дизайна всего сайта;
- предоставляет огромные возможности по контролю конкретных блоков и характеристик (позиционирование, размер, поля и др.).

3.14 JavaScript

JavaScript - самый популярный язык для написания клиентских сценариев. Он позволяет ввести обработку взаимодействия клиента с веб-страницей и дает возможность для создания анимаций и переходов. Без JS невозможно было бы создать современного функционала на стороне клиента.

JavaScript - прототипно-ориентированный сценарный язык программирования. Обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.[8] JS - это объектно-ориентированный ЯП, но используемое прототипирование различает его с классическими класс-ориентированными ЯП. В дополнение ко всему этому в JS присутствуют некоторые свойства, характерные для функциональных языков - функции как объекты первого класса, объекты как списки, анонимные функции - все это придаёт языку дополнительную гибкость. Больше информации про язык JS приведено в книгах. [9, 10]

3.15 SQL

SQL - структурированный язык запросов. Он используется для создания, выполнения каких - либо изменений и управления данными в реляционной БД, которая в свою очередь управляется посредством СУБД.

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

В первую очередь язык SQL является информационно-логическим языком, что означает, что он предназначен для описания, модификации и получения данных из реляционных БД. Вначале SQL был главным способом взаимодействия пользователя с БД и позволял ему выполнять такие операции как: создание в БД новых таблиц, добавление в таблицы данных, изменение существующих значений записей, удаление их, получение значений из таблиц в зависимости от условия запроса, модификация структур таблиц.

Со временем SQL стал сложнее и расширился новыми особенностями, поэтому стал приобретать черты, делающие его похожим на языки программирования. Несмотря на все изменения, затронувшие SQL, он является самым распространенным средством для взаимодействия прикладных программ с БД.

3.16 REST API

REST - является архитектурой взаимодействия компонентов распределенного веб-приложения. В общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

В зависимости от ограничений, наложенных на REST-систему, можно рассмотреть следующие свойства:

- масштабируемость, которая необходима для обеспечения расширения компонентов и взаимодействия между ними.
- производительность - с точки зрения пользователя, основным фактором производительности и эффективности системы может являться взаимодействие компонентов системы.

Для проектирования REST-приложений были предложены следующие ограничения, данные ограничения являются обязательными:

- модель клиент-сервер;
- отсутствие состояния;

- кэширование;
- единообразие интерфейса;
- многослойная система;
- код по требованию.

3.17 Взаимодействие рассмотренных средств

Взаимодействие можно рассмотреть в рамках REST модели, где основными блоками являются серверная часть, клиентская часть и база данных. Для разработки серверной части, взаимодействующей с базой данных и частью клиента, а так же обеспечивающей полного функционала интернет-магазина используются следующие средства. Все начинается со среды разработки, так как без нее создание проектного продукта будет гораздо более громоздким и долгим, для этого мы воспользуемся IntelliJ IDEA, вдобавок к ней идут средства для автоматизации сборки проектов maven. Следующим очень мощным инструментом является фреймворк Spring, именно он позволяет нам быстро добавлять интересующие нас зависимости для более быстрой и качественной разработки, такие как Hibernate для работы с БД, Spring Security и других. Java и все выше сказанные средства позволяют описывать работу веб-сервера, те запросы, которые он должен обрабатывать и ответный результат, который он должен формировать. Без сервера приложения просто не будет существовать. Ни одна система электронной коммерции не обойдется без хранения и взаимодействия с данными. Именно для этого в нашем стеке технологий присутствуют такие пункты как: СУБД PostgreSQL, pgAdmin4 и сам язык SQL. Все эти средства позволяют создать структуру БД, а также вести ее грамотное администрирование, дают нам возможность обрабатывать запросы от клиента на получение или модификацию данных. Заключаящей частью является тот блок, который непосредственно видит и использует пользователь - клиентская часть. Она должна соответствовать высоким стандартам не только своими функциональными возможностями по взаимодействию с сервером, но и по оформлению GUI, так как без красивого,

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		25

понятного и удобного интерфейса на вашем ресурсе пользователи просто не смогут разобраться, где и что находится и как с этим работать. Поэтому графический интерфейс также представляет собой очень важный блок, его разработка будет вестись с помощью таких средств как: HTML и CSS для представления разметки и стилей для сайта, а также JavaScript, чтобы позволить пользователю осуществлять взаимодействие с серверной частью и получать ответы от нее.

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

4 РЕАЛИЗАЦИЯ

Прежде чем начинать непосредственную разработку всех частей приложения, необходимо еще раз проанализировать систему, какой функционал необходимо реализовывать, какие сущности должны находиться в базе данных, какие должны быть атрибуты у этих сущностей. Также необходимо рассмотреть, как именно будет взаимодействовать клиент, сервер и БД, какие для этого нужны будут механизмы и т.д.

4.1 Алгоритм работы системы электронной коммерции

Для того чтобы реализовать систему электронной коммерции нужно подготовить клиента, сервера и СУБД. Вся последовательность работы состоит из следующего алгоритма: пользователь своими действиями создает определенные события, которые обрабатываются клиентской частью, в зависимости от того что сделал клиент, серверу посылается запрос на обработку соответствующей задачи. Если во время выполнения запроса возникнет ошибка, то сервер вернет клиенту информацию об этой ошибке и не выполнит действие, если запрос пройдет успешно, то сервер обратится к СУБД для получения данных, запрашиваемых пользователем. Если при обращении к СУБД возникнет ошибка, то сервер соответствующим способом обработает ее и вернет сообщение об ошибке клиенту, в том же случае, если исключительных ситуаций не возникло, то запрос является выполненным и сервер вернет клиенту информацию об успешном выполнении запроса.

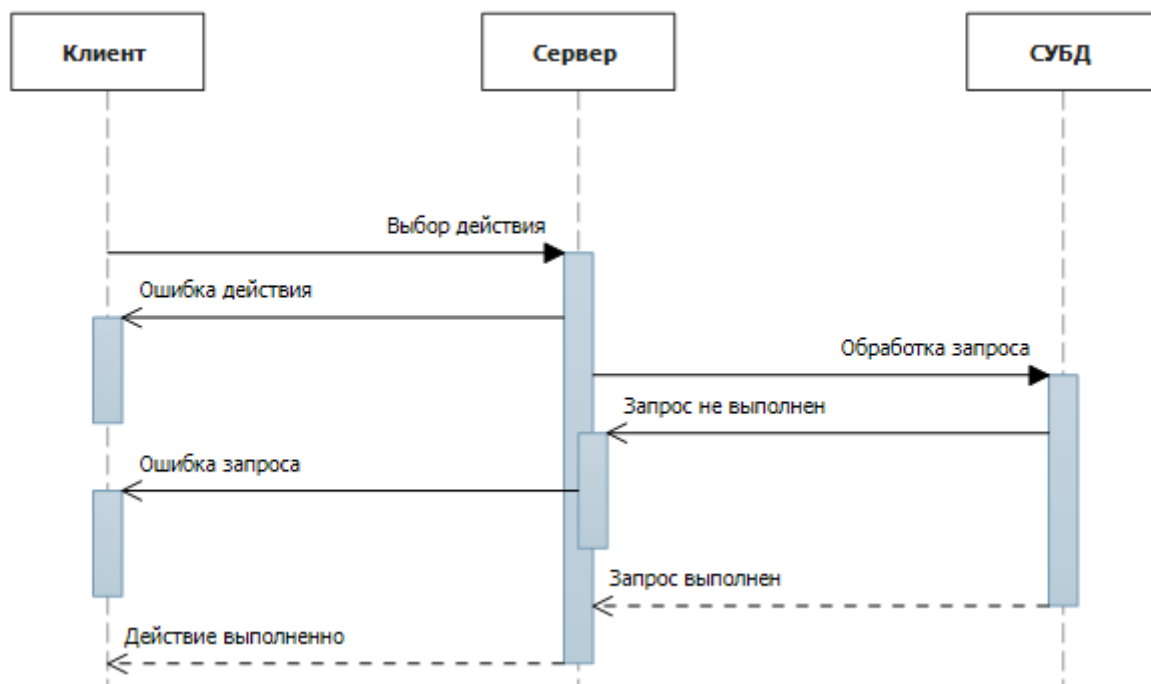


Рисунок 1 – Диаграмма последовательности работы системы электронной коммерции

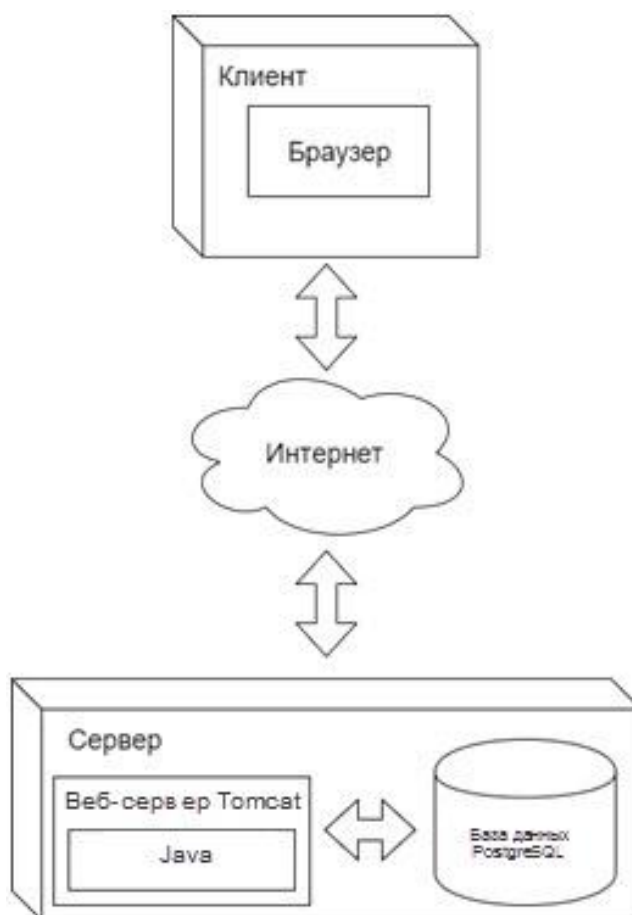


Рисунок 2 – Диаграмма развертывания системы электронной коммерции

4.2 Описание структуры и таблиц БД

Для разработки структуры БД, необходимо проанализировать какие сущности требуют рассмотрения, и какие атрибуты будут им принадлежать. Дополнительно необходимо выявить, как данные сущности будут взаимосвязаны. Все это является работой по созданию ERD (англ. Entity Relation Diagram), т.е. диаграммы взаимодействия сущностей.

Для нормальной работы системы электронной коммерции были разработаны следующие таблицы:

- users - таблица, содержащая основные сведения о пользователе;
- user_role - таблица, которая содержит данные о ролях пользователей, которые необходимы для разграничения прав доступа;
- orders - таблица заказов, отображающая информацию о заказах пользователей, является связующей таблицей между пользователями и продуктами. Хранит в себе данные о дате, статусе и сумме заказа, а также ключи связи с таблицей пользователей и позиций заказов;
- order_item - представляет собой связующее звено между заказом и товаром, фактически хранит в себе одну строку заказа, в которой есть указание на продукт и его количество в текущем заказе;
- products - таблица, хранящая информацию о товарах и их характеристиках.

Таблица 1 - Атрибуты сущности «users» (Пользователи)

Имя столбца	Содержательное описание	Тип данных	Роль атрибута	Пример
1	2	3	4	5
id	Идентификатор пользователя	SERIAL	PK	1
last_name	Фамилия пользователя	VARCHAR(60)		Шеянов
first_name	Имя пользователя	VARCHAR(60)		Николай
middle_name	Отчество пользователя	VARCHAR(60)		Николаевич
phone_number	Номер телефона	VARCHAR(20)		89273952626
e_mail	Адрес электронной почты	VARCHAR(60)		shnn1998@gmail.com
login	Псевдоним пользователя	VARCHAR(60)		shnn1998
password	Зашифрованный пароль пользователя	VARCHAR(255)		\$P\$DAV2YPwBd...
comment_by_admin	Комментарий администратора	VARCHAR(255)		Пользователь надежный, можно доверять
user_role_id	Идентификатор роли пользователя	INTEGER	FK	1
street	Составляющая адреса - улица	VARCHAR(60)		Южная
house	Составляющая адреса - дом	VARCHAR(60)		76
flat	Составляющая адреса - квартира	VARCHAR(60)		1
reg_date	Дата регистрации пользователя	TIMESTAMP WITH TIME ZONE		2019-05-18 11:07:32

Таблица 2 - Атрибуты сущности «user_role» (Список ролей присущих пользователям)

Имя столбца	Содержательное описание	Тип данных	Роль атрибута	Пример
1	2	3	4	5
id	Идентификатор роли	INTEGER	PK	2
role_name	Наименование роли пользователя	VARCHAR(30)		Manager

Таблица 3 - Атрибуты сущности «orders» (Заказы)

Имя столбца	Содержательное описание	Тип данных	Роль атрибута	Пример
1	2	3	4	5
id	Идентификатор заказа	SERIAL	PK	13
user_id	Идентификатор пользователя	INTEGER	FK	1
order_num	Номер заказа	INTEGER		121
status	Статус заказа	VARCHAR (60)		Завершен
order_sum	Общая стоимость заказа	DOUBLE PRECISION		1000.00
order_date	Дата регистрации заказа	TIMESTAMP WITH TIME ZONE		2019-05-22 17:01:12

Таблица 4 – Атрибуты сущности «order_item» (Позиция заказа)

Имя столбца	Содержательное описание	Тип данных	Роль атрибута	Пример
1	2	3	4	5
id	Идентификатор	SERIAL	PK	12
orders_id	Идентификатор заказа	INTEGER	FK	21
products_id	Идентификатор товара	INTEGER	FK	5
count_item	Количество товара в позиции	INTEGER		7

Таблица 5 – Атрибуты сущности «products» (Товары)

Имя столбца	Содержательное описание	Тип данных	Роль атрибута	Пример
1	2	3	4	5
id	Идентификатор	SERIAL	PK	5
name	Цена товара	VARCHAR(255)		Торт бисквитный
code	Количество товара в наличии	VARCHAR(255)		t003
price	Количество товара в обработке на заказ	DOUBLE PRECISION		1300.00
description	Дата создания товара	VARCHAR(255)		Бисквитный торт, лучший десерт на вашем празднике
photo	Дата последнего изменения информации о товаре	BYTEA		Th\0t0oas

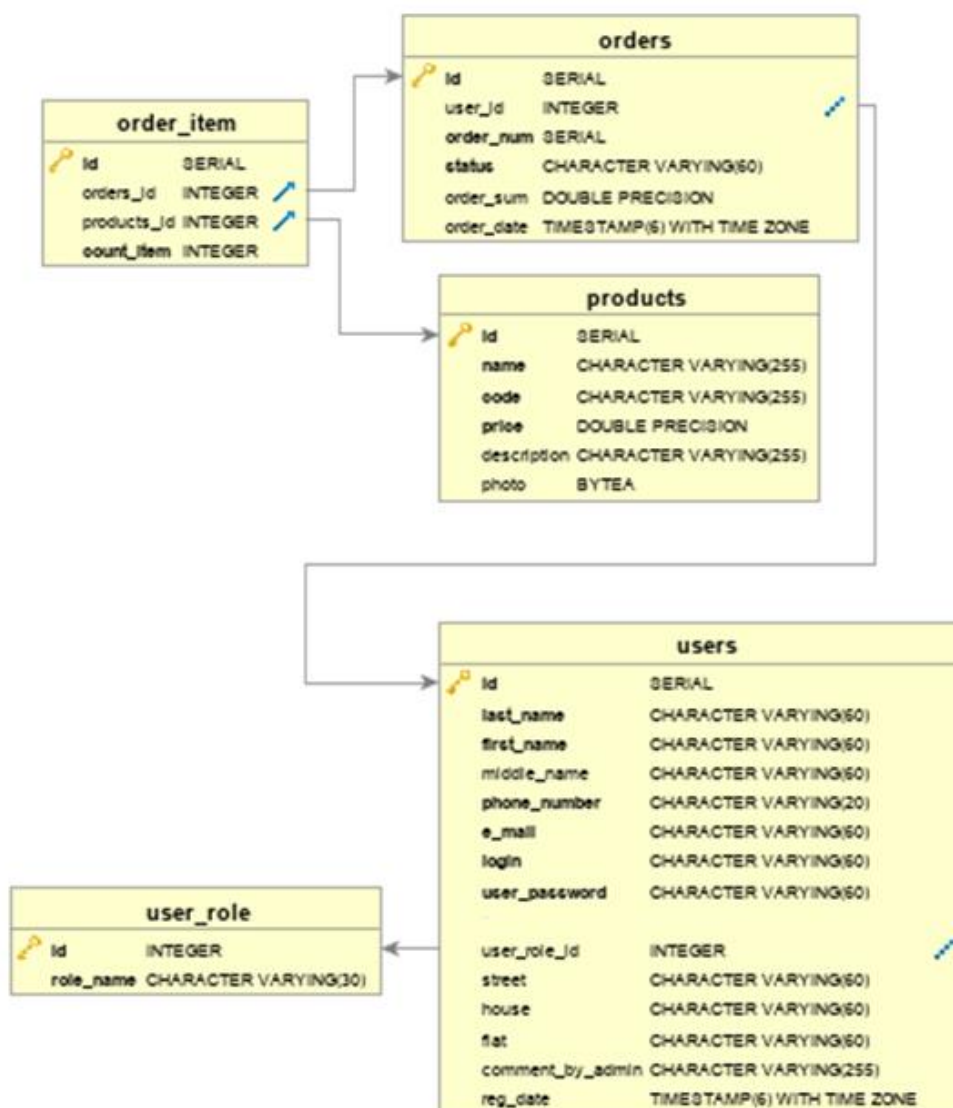


Рисунок 3 – Схема сущностей с атрибутами

4.3 Основные функции доступные пользователям

Система электронной коммерции делится на две группы основных пользователей. К ним относятся клиенты и сотрудники. Клиент в свою очередь может быть авторизован на сайте или не авторизован. Распределение обязанностей сотрудников строится на договоренностях, построенных при личном взаимодействии, что означает, что программно все работники имеют одинаковый уровень доступа и взаимодействия со средой сайта.

Возможности, предоставленные каждой категории пользователей сайта, можно рассмотреть в виде диаграмм вариантов использования. Данные диаграммы наглядно и полно показывают все функции, которые могут выполнять пользователи в рамках прав, которыми они обладают.

Самым низкоуровневым по правам пользователем является неавторизованный пользователь. Ему доступен самый узкий функционал общения с сайтом, в него входит:

- прохождение регистрации на сайте, чтобы получить права авторизованного пользователя;
- просмотр различной информации, например, главной страницы сайта, на которой расположено описание, к какой области относится магазин, контактные данные магазина, каталог товаров, а также способ оплаты заказа. Дополнительно неавторизованный пользователь имеет возможность зайти на страницу товара, где ему будет предоставлена информация о характеристиках выбранного товара, его цене, внешнем виде и другом описании;
- вдобавок к этому пользователь может производить сортировку интересующего товара для более удобного поиска и отображения.

Диаграмма вариантов использования для неавторизованного пользователя представлена на рисунке 4.



Рисунок 4 - Диаграмма вариантов использования неавторизованного пользователя

Следующим по уровню привилегий располагает авторизованный пользователь, чтобы получить данный статус, необходимо всего лишь пройти регистрацию и далее зайти под своим логином и паролем в систему. Авторизованный пользователь включает в себя все возможности, присущие неавторизованному пользователю, но имеет и дополнительные, такие как :

- авторизоваться на сайте и войти в свой профиль;
- изменить персональные данные, таких как: имя, фамилия, отчество, номер телефона, почтовый адрес, физический адрес, представленный в виде улицы, дома и квартиры, данная информация необходима для обеспечения доставки курьером до места;
- авторизованному пользователю разрешено добавлять продукты в корзину для осуществления дальнейшего процесса заказа;
- пользователь может посмотреть все ранее сделанные им заказы.

Диаграмма вариантов использования для авторизованного пользователя представлена на рисунке 5.

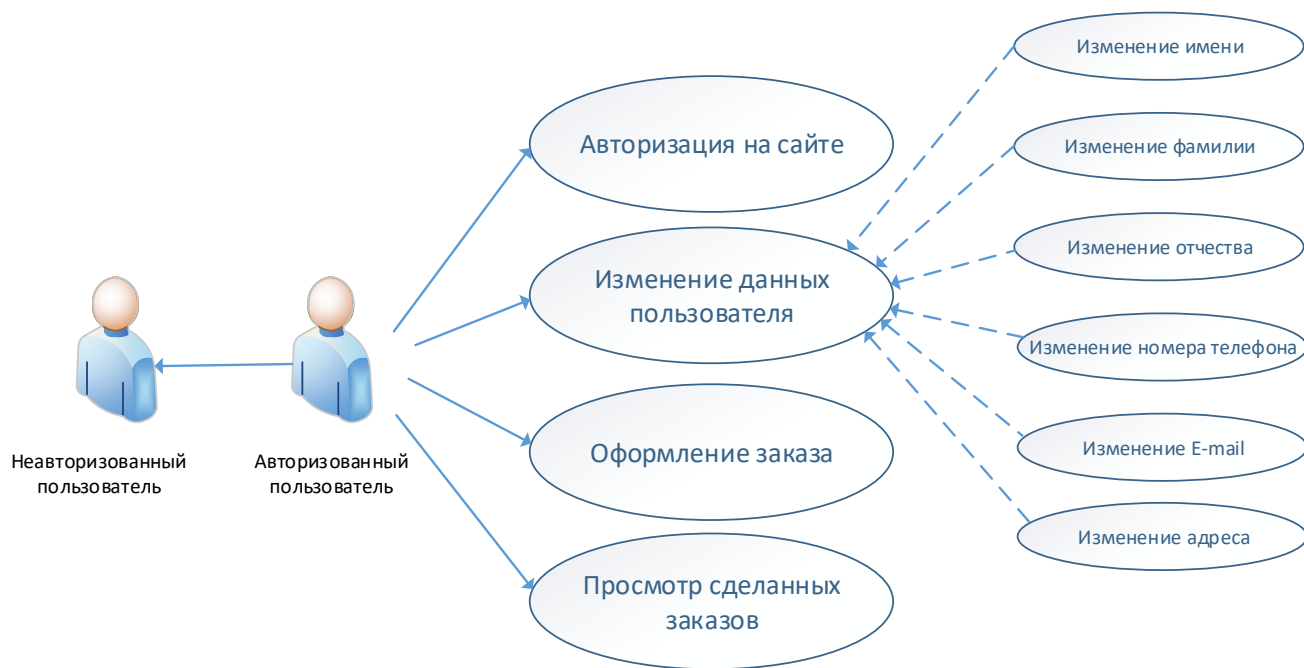


Рисунок 5 - Диаграмма вариантов использования авторизованного пользователя

Пользователем с самым высоким доступом прав является работник сайта. Ему доступно огромное количество функций, поэтому, скорее всего физически всю работу по управлению сайтом будет осуществлять группа человек, разделивших обязанности между собой. При этом каждый из них имеет равный уровень прав.

В деятельности работника можно выделить следующие крупные направления, как:

- работа с пользовательской информацией и пользователями;
- работа с товарами;
- работа с заказами.

В каждой из этих сфер есть множество подфункций, доступных работникам. При работе с пользователем работник в праве:

- оставлять комментарии к профилям клиентов, чтобы указать, насколько этот пользователь благонадежен, не имеет ли он каких-либо просрочек с оплатой заказов и др.

- администрировать списки пользователей, что заключается в просмотре полного списка клиентов, удаление пользователя, просмотр списка заказов конкретного пользователя.

При работе с товарами существует четыре основных операции:

- просмотр товара на актуальность информации о нем;
- добавление нового товара;
- удаление старого товара;
- модификация данных о товаре, который претерпел какие-либо изменения.

При работе с заказами администратор может и должен:

- производить обновления статусов заказов в соответствии с их обработкой;
- просматривать все осуществленные заказы;
- управлять процессом обработки заказов, что заключается в подтверждении, отмене и удалении заказов.

Диаграмма вариантов использования для работника магазина представлена на рисунке 6.

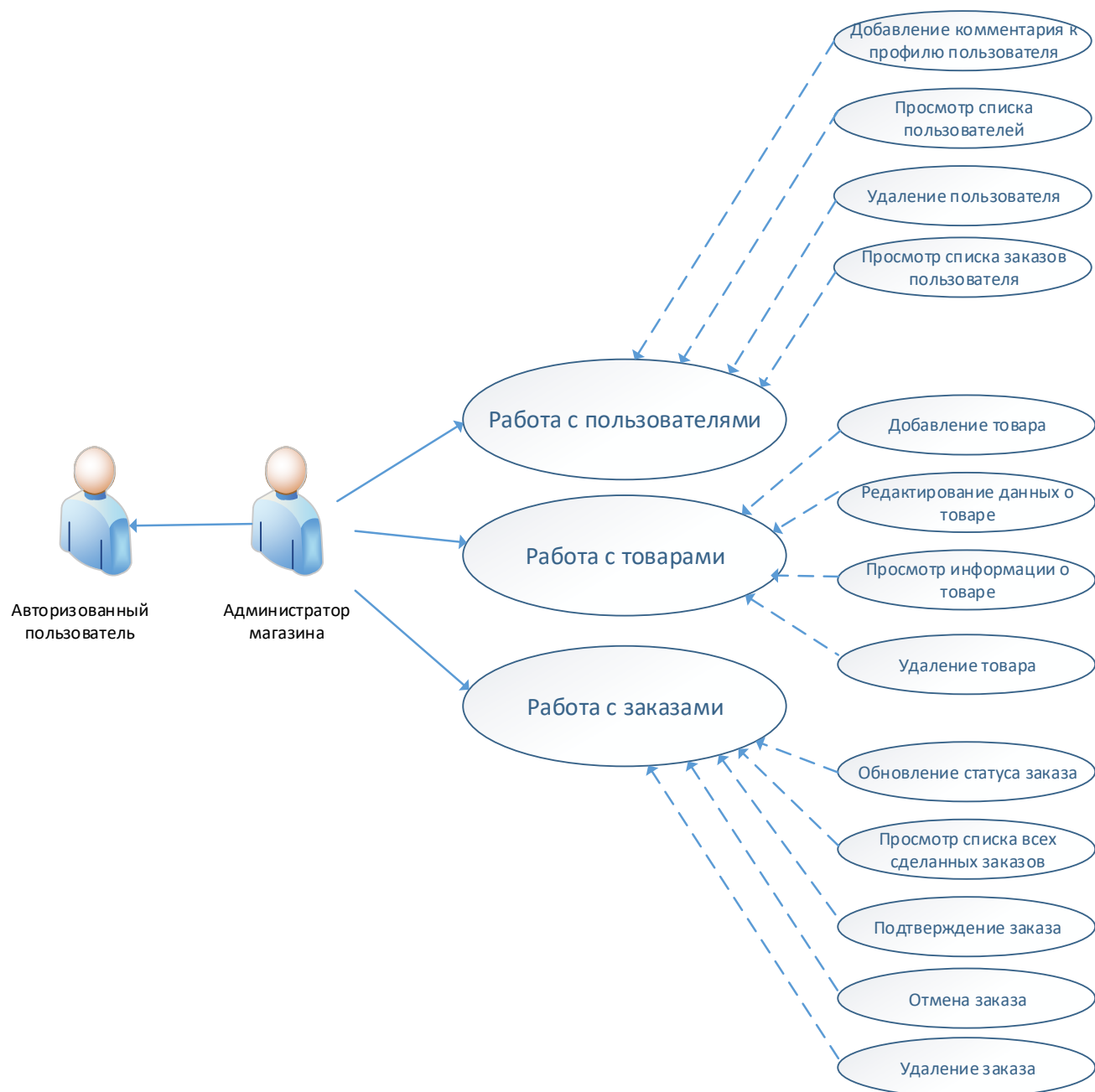


Рисунок 6 - Диаграмма вариантов использования работника

4.4 Схема оформления заказа

Рассмотреть процесс осуществления заказа можно с помощью диаграммы оформления заказа. Он состоит из взаимодействия клиентской и серверной стороны.

Процесс начинается с того, что пользователь зашел в каталог товаров и начал добавлять товары в корзину. После выбора клиентов всех товаров, которые он хочет приобрести, он должен перейти на страницу корзины. В это время на сервере происходит проверка на наличие товаров в корзине. Если корзина пуста, то пользователю отправляется соответствующее сообщение, если в корзине есть товары, то пользователь переходит к редактированию корзины, где он может увеличить или уменьшить количество приобретаемого товара, а также удалить товар из корзины, после чего происходит переход к заполнению контактных данных пользователя для осуществления заказа. Если пользователь авторизован и ввел свои контактные данные, то они берутся с его профиля и указываются в заказе, если пользователь не указал данные, то сейчас необходимо будет ввести их. После этого покупателю необходимо подтвердить осуществление заказа. Запрос отправляется на сервер, где проверяется корректность введенных данных и дальнейшее продолжение процесса заказа. В том случае, когда данные указаны корректно, то система отправляет ответ об успешности выполнения заказа и сохраняет его в системе, а так же информацию о данном заказе пользователь может посмотреть у себя в профиле. На этом процесс оформления заказа является окончен. Диаграмма деятельности оформления заказа представлена на рисунке 7.

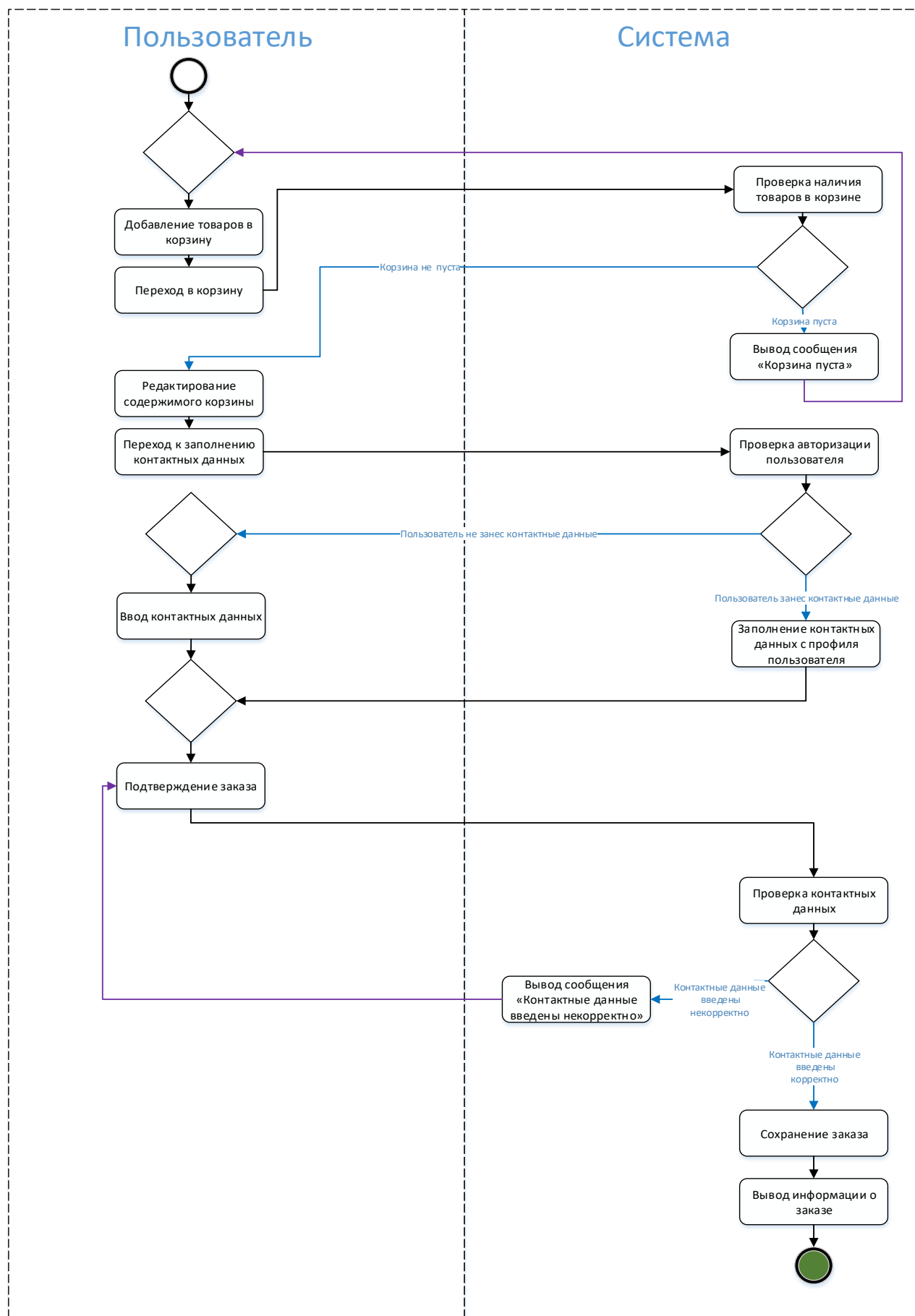


Рисунок 7 - Диаграмма деятельности оформления заказа

4.5 Схема обработки заказа

Работник должен осуществлять своевременное обновление информации о заказах, а также выполнять операции по управлению состоянием заказа. Пользователь с достаточными правами при получении заказа должен выполнить следующие действия с ним: подтвердить, отменить или удалить заказ.

При выполнении одной из операций, заказу присваивается соответствующий статус и на этом дальнейшая обработка заказа либо заканчивается, либо нет. Если заказ подтвержден, то в дальнейшем он должен быть оплачен, после чего он получит статус "Оплачен". После выполнения заказа и получения товара пользователем заказ получает статус "Выполнен". Диаграмма деятельности обработки заказа представлена на рисунке 8.

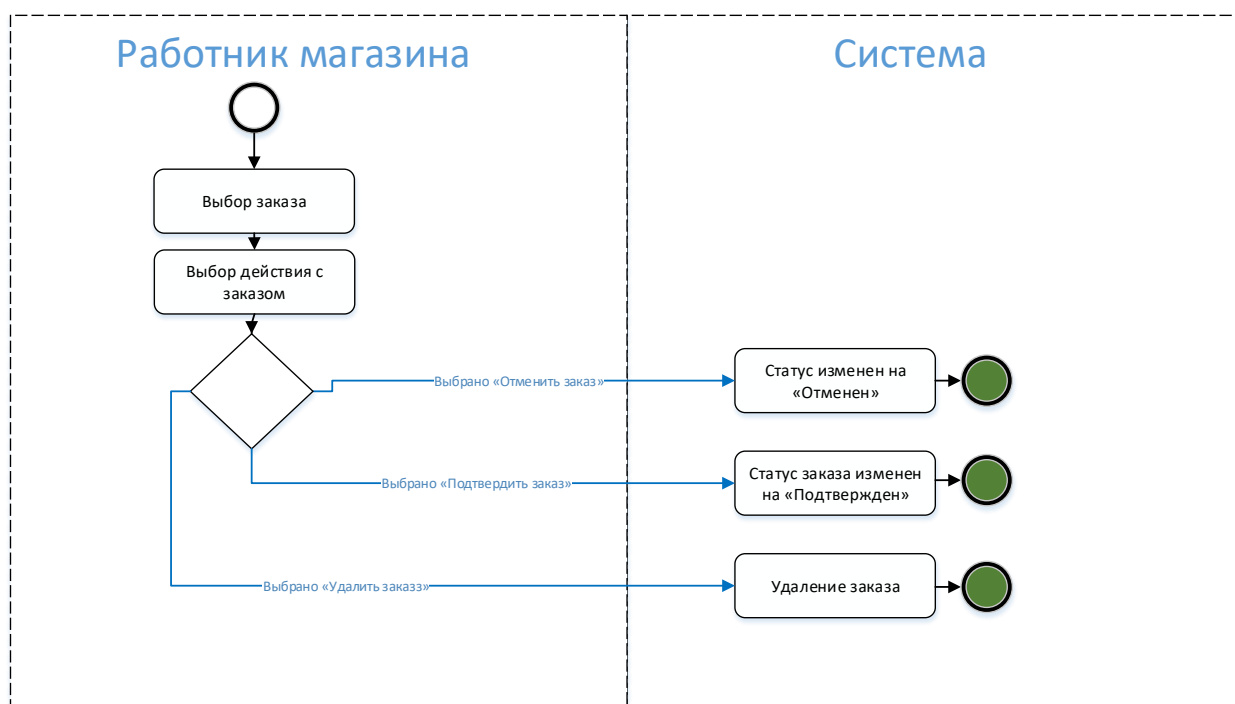


Рисунок 8 - Диаграмма деятельности обработки заказа

5 МЕТОДЫ ИСПЫТАНИЙ

5.1 Регистрация пользователя и вход в профиль

После выполнения работы, необходимо удостовериться в работоспособности сайта, для чего необходимо провести действия по регистрации и входу и, в дальнейшем, по выполнению заказа.

Для проверки работы регистрации необходимо выполнить следующие действия:

- войти на сайт, введя в строке браузера его адрес, главная страница представлена на рисунке 9;



Рисунок 9 - Главная страница сайта "Крендель - Пенза"

- нажать на кнопку «Sign up», она находится в правом верхнем углу сайта (рис. 10);

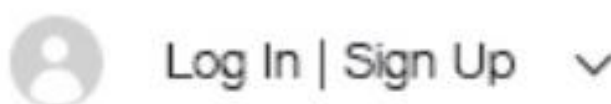


Рисунок 10 - Кнопка «Login/SignUp»

- форма регистрации представлена на рисунке 11;

Регистрация

Имя* Фамилия*

Логин* Пароль*

Эл. почта* Телефон*

Отчество Улица

Дом Квартира

Отправить

Рисунок 11 – Форма «Регистрация»

- заполнить все поля и нажать кнопку «Отправить»

После чего в таблице "users" появятся данные о новом пользователе (рис.12)

	id [PK] integer	last_name character varying (60)	first_name character varying (60)	middle_name character varying (60)	phone_number character varying (20)	e_mail character varying (60)	login charac
5	24	Шеянов	Николай	Николаевич	89273952672	pochta@bk.ru	pochta
6	25	Андреев	Олег	Борисович	89276662539	olegBor@mail.ru	olegBig
7	26	Гроссман	Сергей	Борисович	89993649123	grossS@gmail.com	Grossn

Рисунок 12 - Таблица users

Для проверки входа в личный кабинет необходимо:

-нажать на кнопку «Login», она находится в правом верхнем углу сайта. Открывается страница с формой входа в аккаунт (рис. 13);

Рисунок 13- Форма входа в личный кабинет

- после чего необходимо заполнить данные и нажать кнопку "Войти", после выполнения данной операции пользователь может приступить к оформлению заказа.

5.2 Каталог и корзина

Для просмотра всех товаров на сайте существует каталог. Он позволяет ознакомиться с информацией о товарах, их наименовании, описании и внешнем виде.

Внешний вид каталога представлен на рисунке 14.

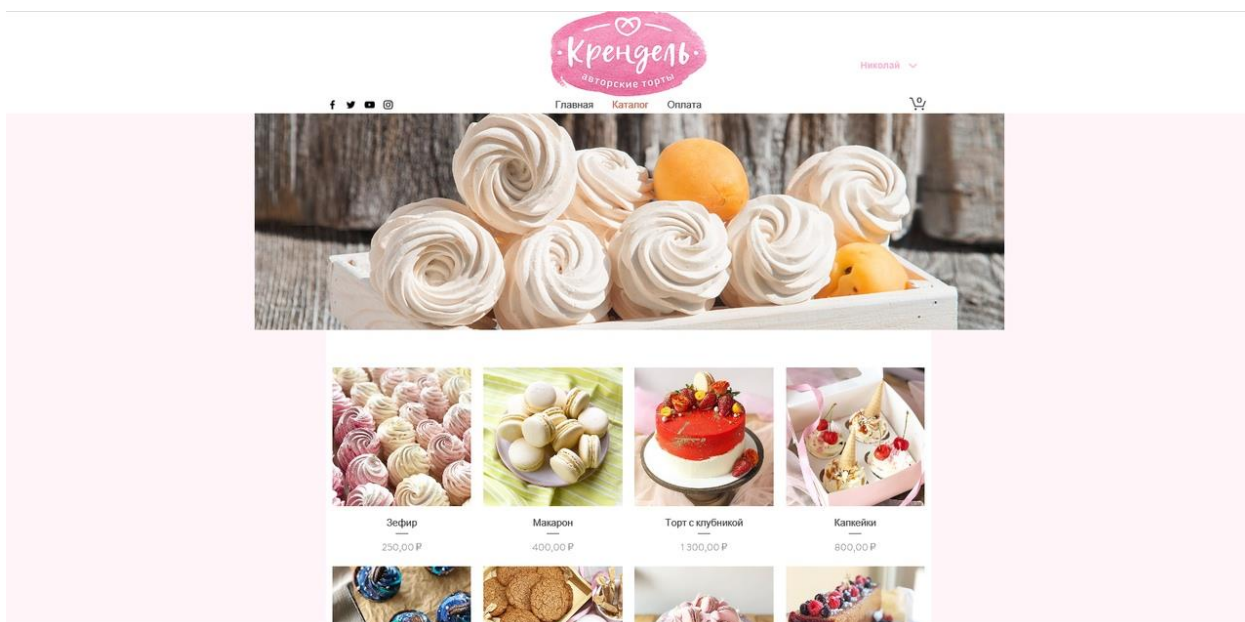


Рисунок 14 - Каталог товаров

В каталоге товаров, мы можем выбрать интересующий нас товар и перейти на страницу с описанием этого товара. Также мы можем добавить выбранное количества товара в корзину (рис. 15).

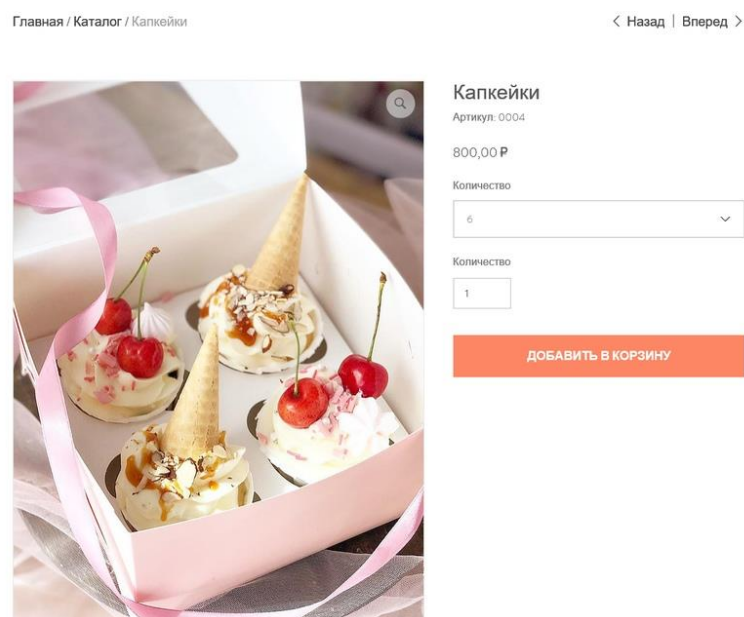


Рисунок 15 - Страница товара "Капкейки"

После того как будут добавлены все необходимые товары, можно перейти на страницу "Корзина". Здесь мы можем просмотреть добавленные товары, удалить их из корзины, изменить количество, а также увидеть цену и итоговую сумму за заказ (рис. 16).

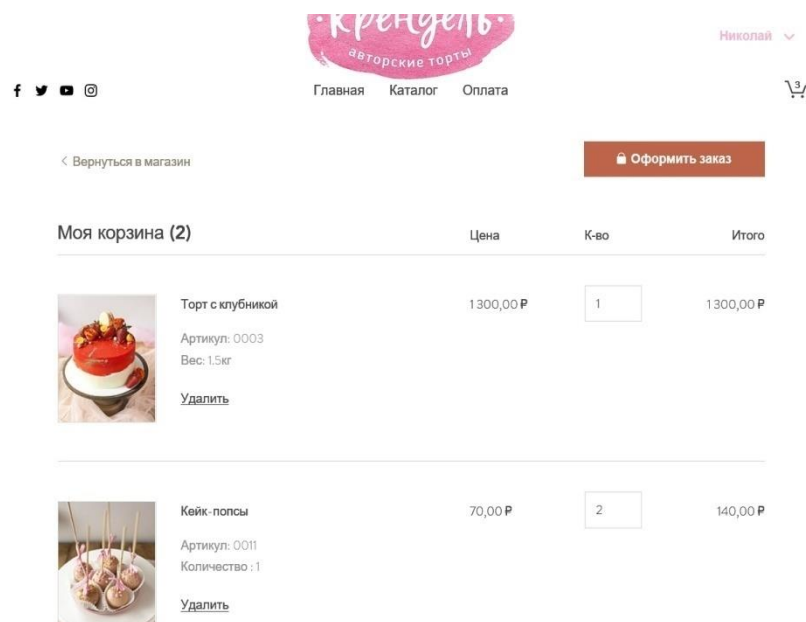


Рисунок 16 - Пользовательская корзина

После того, как список товаров настроен, для оформления заказа необходимо нажать клавишу "Оформить заказ". В результате в базе данных появится новая запись заказа (см. рисунок 17). Теперь остается только ждать звонка от сотрудника магазина.

	id [PK] integer	user_id integer	order_num integer	status character varying (60)	order_sum double precision	order_date timestamp w
2	14	20	2	Подтвержден	1440	2019-06-14 0

Рисунок 17 - Таблица orders

Для того чтобы посмотреть способ оплаты, необходимо вернуться на главную страницу и с нее перейти на страницу оплаты, где расположена вся информация по данному процессу (рис. 18).

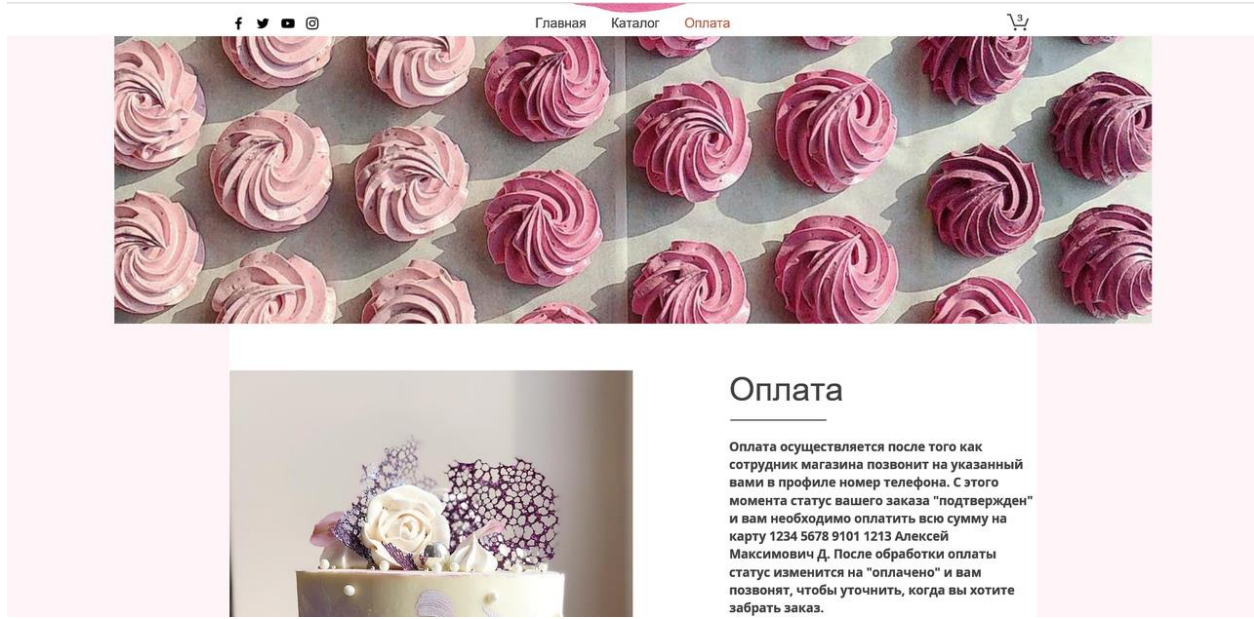


Рисунок 18 - Страница вариантов оплаты заказов

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной бакалаврской работы был разработан алгоритм решения задачи, создан и отлажен сайт, оформлена документация. Проведенные испытания показали, что клиент-серверное приложение для отслеживания работы электронной коммерции соответствует всем поставленным требованиям и выполняет свою работу корректно.

За время работы с данным проектом было прочитано большое количество технической документации, изучены новые способы и механизмы программирования и проектирования систем. Значительно развит навык разработки веб-приложений, администрирования баз данных. Были рассмотрены многие популярные современные средства для разработки и тестирования, такие как: Spring Framework, Java, Insomnia, pgAdmin4, PostgreSQL, Vue.js и другие. Опыт, полученный в ходе выполнения этого задания, однозначно поможет в дальнейшем трудоустройстве и развитии себя.

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

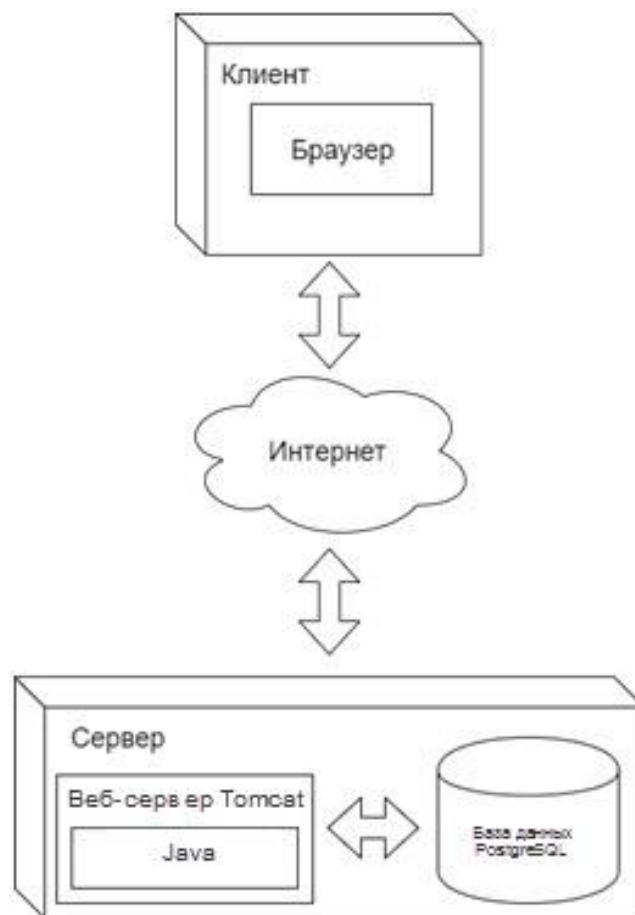
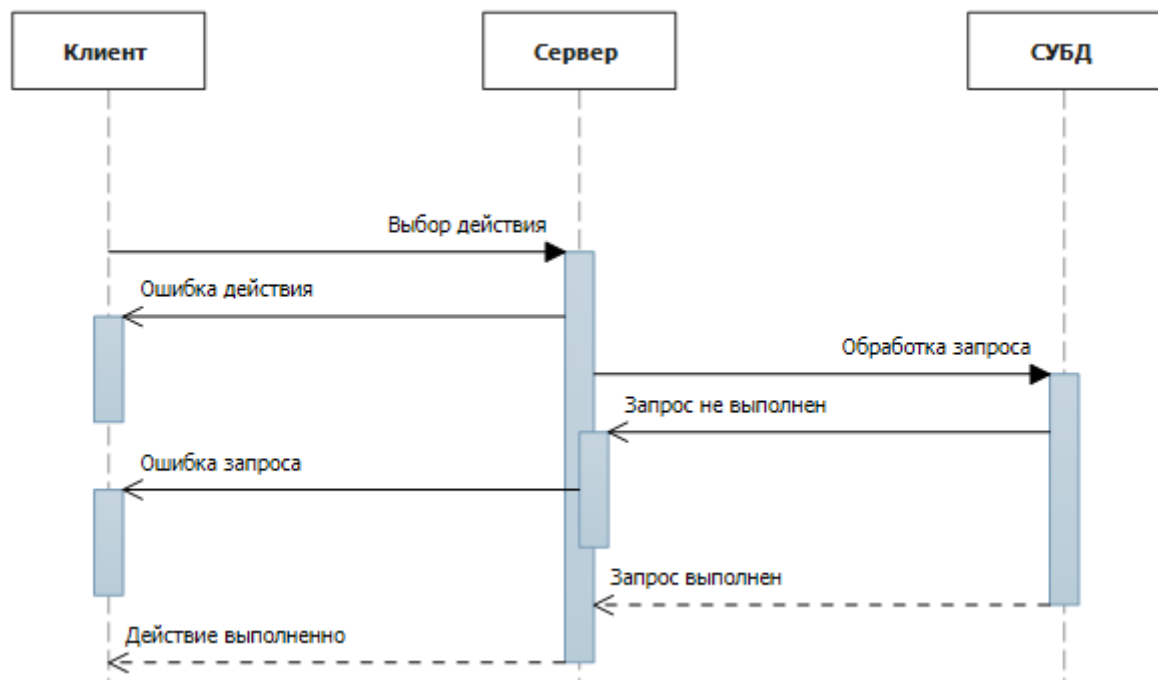
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шилдт Герберт. Java 8: руководство для начинающих, 6-е изд. : Пер. с англ. - М.ООО "И.Д. Вильяме", 2015. - 720 с.: ил. - Парал. тит. англ.
2. Флэнаган Д. JavaScript. Подробное руководство. – Пер. с англ. – СПб: Символ_Плюс, 2008. – 992 с.
3. Ржеуцкая С.Ю. Базы данных. Язык SQL: учеб. пособие / С.Ю. Ржеуцкая. – Вологда: ВоГТУ, 2010. – 159 с.
4. Бейли Л. Изучаем SQL. –СПб.: Питер, 2012. –592 с.
5. Сысолетин Е. Г. Проектирование интернет-приложений : учеб.-метод. пособие / Е. Г. Сысолетин, С. Д. Ростунцев. — Екатеринбург : Изд-во Урал.ун-та, 2015. — 92 с.
6. James J. Ye. Building Applications with Spring 5 and Vue.js 2
7. Роббинс Дж. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дженнифер Роббинс; 4-е издание. М. : Эксмо, 2014. — 528 с.
8. Д.Румянцев. Сам себе Web-программист. Практикум создания качественного Web-сайта. – М.: ИНФРА-М, 2001.
9. Фримен Э., Робсон Э. Изучаем программирование на JavaScript. – СПб.: Питер, 2015. – 640 с.
10. Закас Н. JavaScript для профессиональных веб-разработчиков / [Пер. с англ. А. Лютича]. – СПб.: Питер, 2015. – 960 с.
11. Craig Walls. Spring in Action Fifth Edition. Covers Spring 5.0
12. Erik Wilde, Cesare Pautasso. REST: From Research to Practice. — «Springer Science & Business Media», 2011. — С. 52-340.
13. Клименко Р. А. Веб-мастеринг на 100%. –СПб.: Питер, 2013. –512 с.
14. Herbert Schildt. Java 8. The Complete Reference, 9th Edition.
15. Роберт Седжвик, Кевин Уэйн "Алгоритмы на Java", 4-е издание
16. Spring Framework Reference Documentation - интернет ресурс. URL - <https://docs.spring.io/spring-framework/docs/4.2.x/spring-framework-reference/html/>

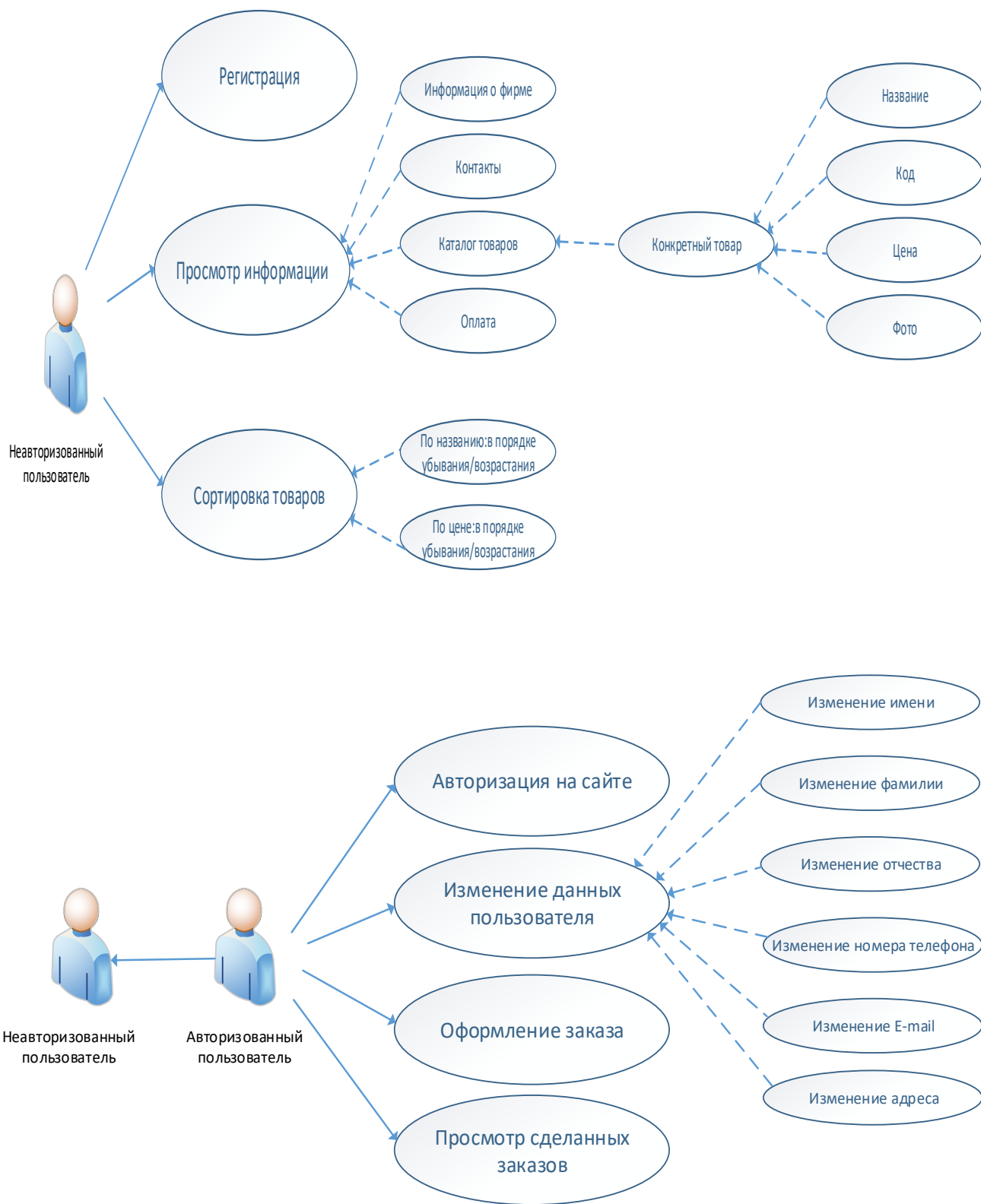
ПРИЛОЖЕНИЕ А

(Графическая часть)

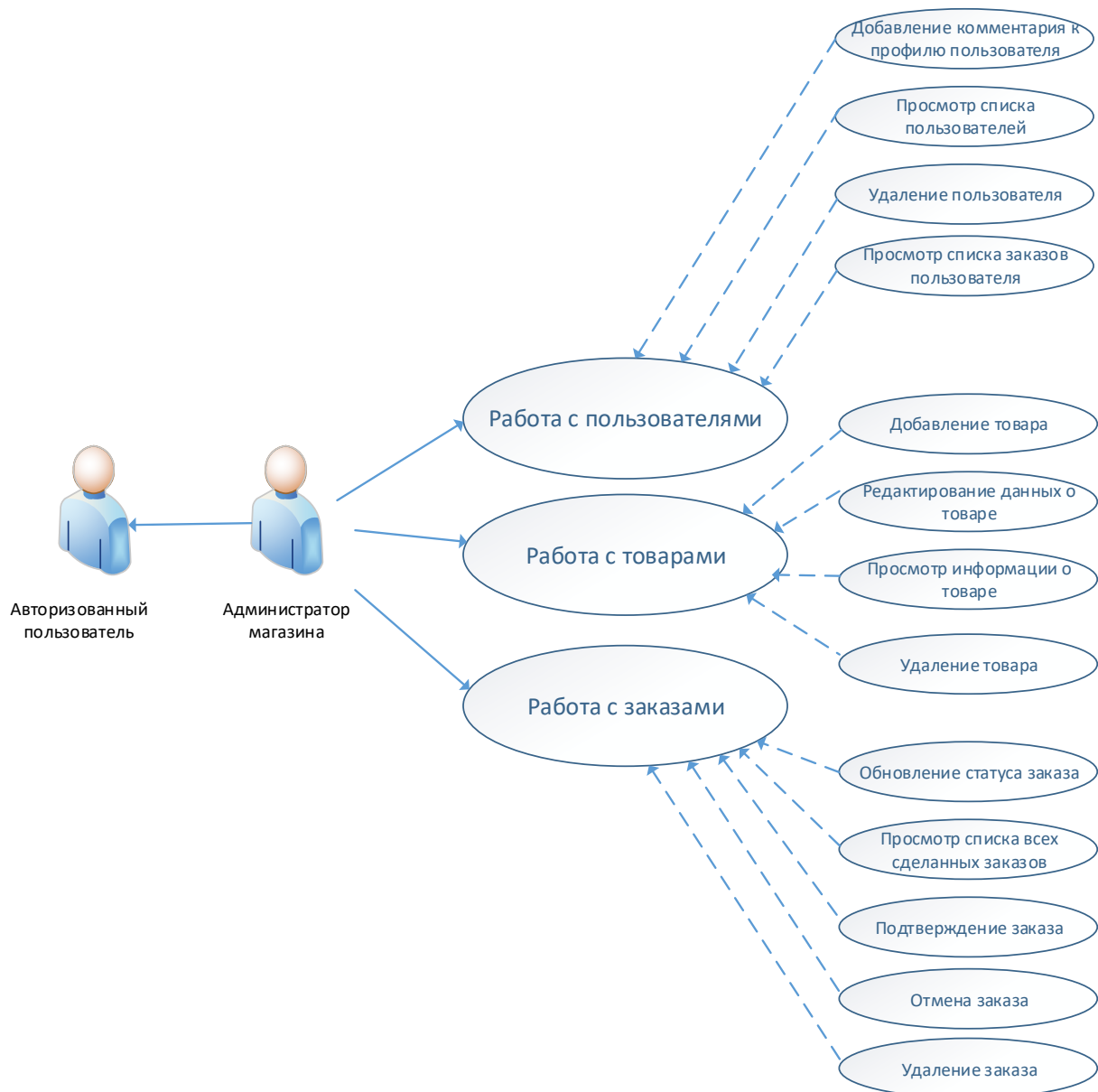
					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		



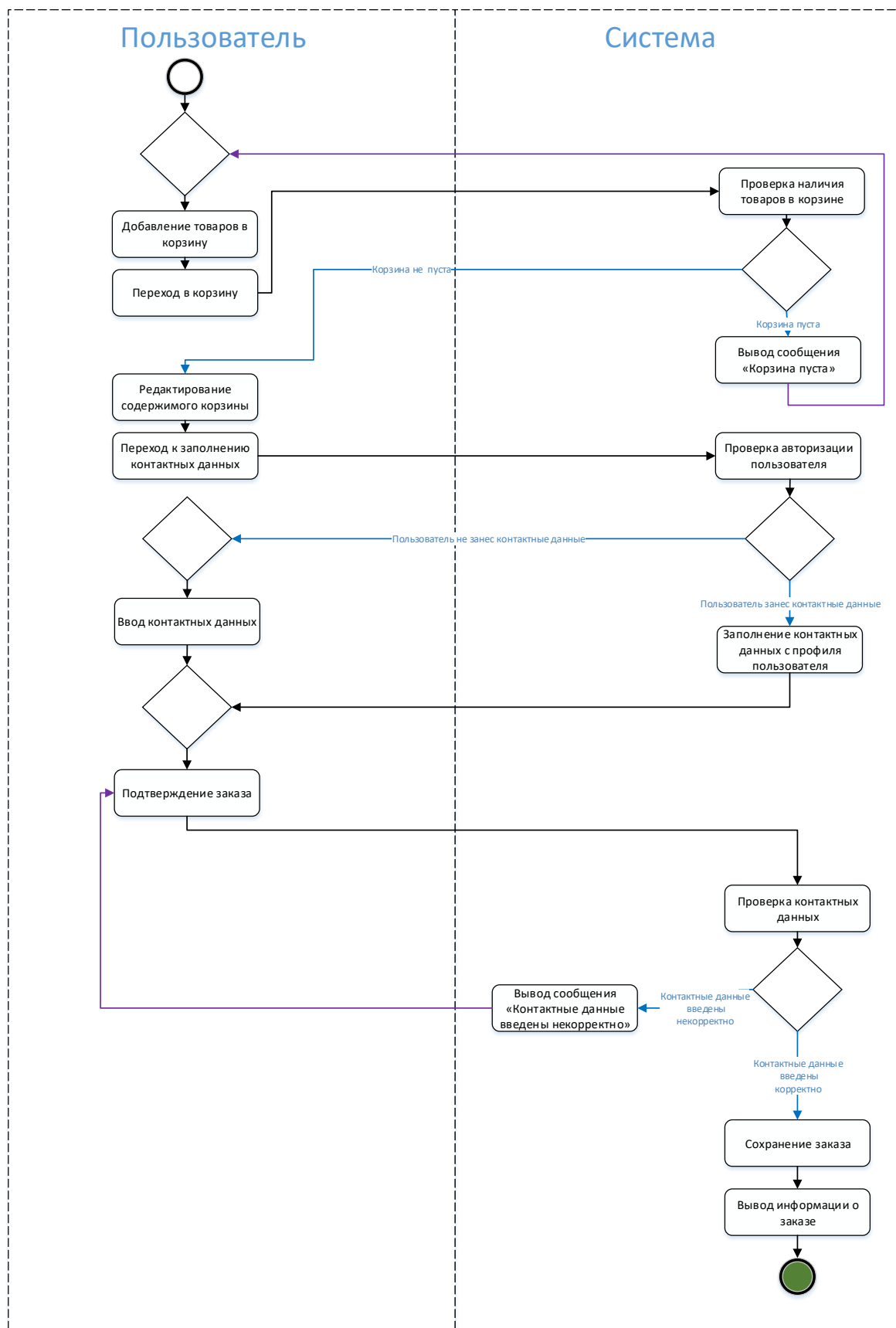
					ПГУ 1.09.03.01.29.002 Д1			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Шеянов Н.Н.			Клиент-серверное приложение для отслеживания работы электронной коммерции Структура взаимодействия компонентов системы	Лит.	Лист	Листов
Провер.		Трокоз Д.А.					1	1
Н. Контр.		Кучин А.В.				ФВТ гр. 15BB1		
Утв.		Митрохин М.А.						



					ПГУ 1.09.03.01.29.003 Д2			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Шеянов Н.Н.			Клиент-серверное приложение для отслеживания работы электронной коммерции Неавторизованный пользователь, авторизованный пользователь UML диаграммы вариантов использования	Лит.	Лист	Листов
Провер.		Трокоз Д.А.					1	1
Н. Контр.		Кучин А.В.				ФВТ гр. 15ВВ1		
Утв.		Митрохин М.А.						



					ПГУ 1.09.03.01.29.004 Д2		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Шеянов Н.Н.			Клиент-серверное приложение для отслеживания работы электронной коммерции Администратор магазина UML диаграмма вариантов использования	Лит.	Лист
Провер.		Трокоз Д.А.					1
						ФВТ гр. 15ВВ1	
Н. Контр.		Кучин А.В.					
Утв.		Митрохин М.А.					
						Листов	1



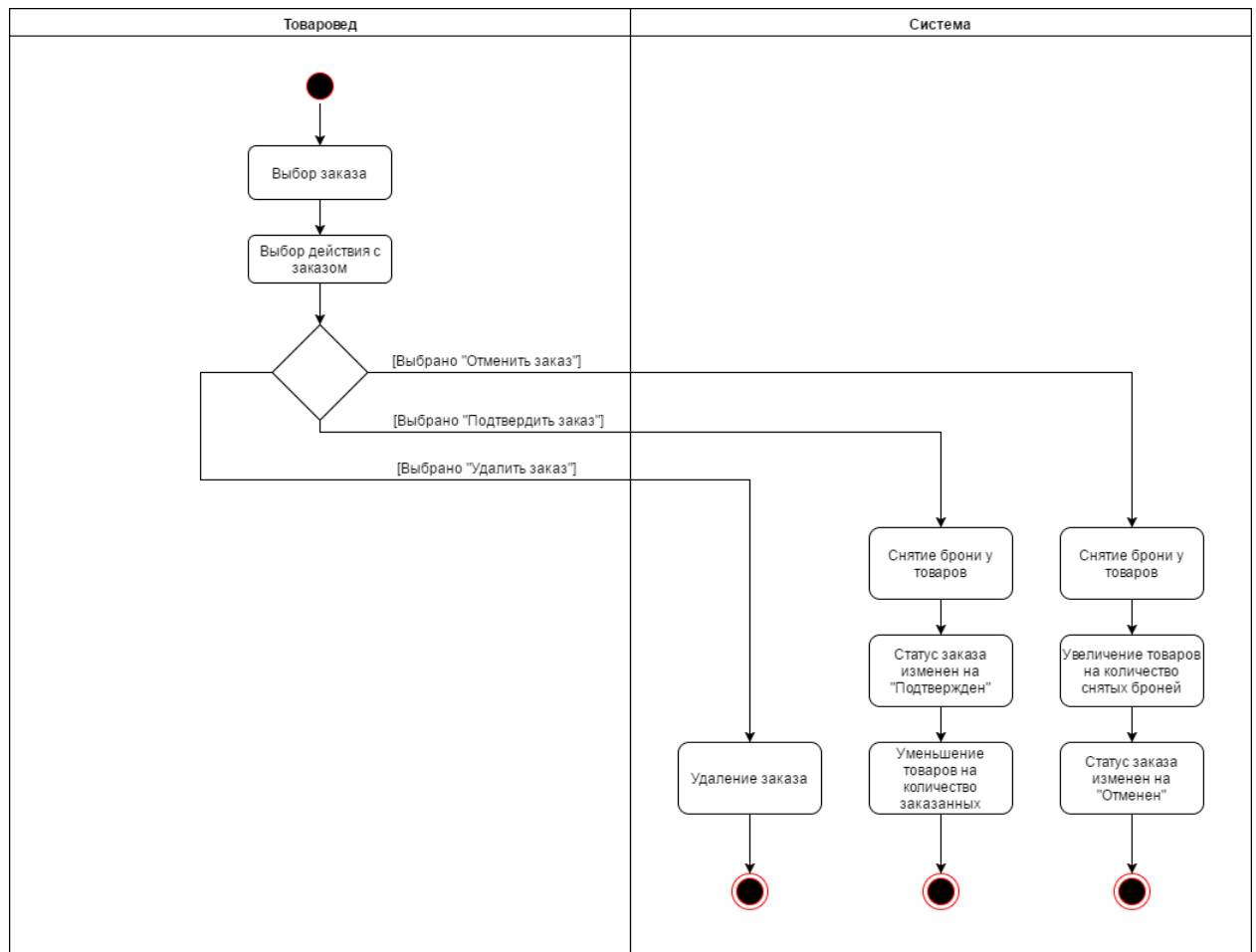
Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Шеянов Н.Н.		
Провер.		Трокоз Д.А.		
Н. Контр.		Кучин А.В.		
Утв.		Митрохин М.А.		

ПГУ 1.09.03.01.29.005 ДЗ

Клиент-серверное приложение для
отслеживания работы электронной
коммерции
Оформление заказа
UML диаграмма деятельности

Лит.	Лист	Листов
	1	1

ФВТ
гр. 15BB1



ПРИЛОЖЕНИЕ Б
(Листинги программы)

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение Б.1 Файл Order.java

```
@Entity
@Data
@Table(name="orders")
@NamedEntityGraphs({
    @NamedEntityGraph(name = "graph.order_items.products",
        attributeNodes = @NamedAttributeNode(value =
"order_items", subgraph = "order_items"),
        subgraphs = @NamedSubgraph(name = "order_items",
attributeNodes = @NamedAttributeNode("product"))))})
public class Order {    @GenericGenerator(
        name = "orders_seq",
        strategy =
"org.hibernate.id.enhanced.SequenceStyleGenerator",
        parameters = {
            @Parameter(name = "sequence_name", value =
"orders_seq"),
            @Parameter(name = "INCREMENT", value = "1"),
            @Parameter(name = "MINVALUE", value = "1"),
            @Parameter(name = "MAXVALUE", value =
"2147483647"),
            @Parameter(name = "CACHE", value = "1")
        }
    )
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"orders_seq")
    @Column(name = "id", nullable = false, updatable = false)
    private Integer id;
    //связь М-1 от таблицы Заказ к таблице Пользователь
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "user_id", insertable = false, updatable = false
)
    @JsonInclude(NON_NULL)
    private User user;
    @Column(name = "user_id")
    private Integer user_id;
    //@GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "order_num")//,unique=true)
```

```

private Integer order_num;
@Column(name = "status", nullable = false)
private String status;
@Column(name = "order_sum", nullable = false)
private Double order_sum;
@Column(name = "order_date", updatable = false)
@JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-mm-dd
hh:mm:ss")
private LocalDateTime order_date;
//связь 1-М от таблицы Заказ к таблице Позиция заказа
@JsonIgnore
@OneToMany(mappedBy = "order", fetch = FetchType.LAZY,
orphanRemoval = true)
private List<Order_item> order_items = new
ArrayList<Order_item>();}

```

Приложение Б.2 Файл Order_item.java

```

@Entity
@Data
@Table(name="order_item")
public class Order_item {
    @GenericGenerator(
        name = "order_item_seq",
        strategy =
"org.hibernate.id.enhanced.SequenceStyleGenerator",
        parameters = {
            @Parameter(name = "sequence_name", value =
"order_item_seq"),
            @Parameter(name = "INCREMENT", value = "1"),
            @Parameter(name = "MINVALUE", value = "1"),
            @Parameter(name = "MAXVALUE", value =
"2147483647"),
            @Parameter(name = "CACHE", value = "1")
        }
    )
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"order_item_seq")
    @Column(name = "id", nullable = false, updatable = false)
    private Integer id;
}

```

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

```

//связь М-1 от таблицы Позиция заказа к таблице Заказ
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "orders_id", insertable = false, updatable =
false)
@JsonInclude(NON_NULL)
private Order order;
@Column(name = "orders_id")
private Integer orders_id;
//связь М-1 от таблицы Позиция заказа к таблице Товар
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "products_id", insertable = false, updatable =
false)
@JsonInclude(NON_NULL)
private Product product;
@Column(name = "products_id")
private Integer products_id;
@Column(name = "count_item", nullable = false)
private Integer count_item;}

```

Приложение Б.3 Файл Product.java

```

@Entity
@Data
@Table(name = "products")
public class Product {
    @GenericGenerator(
        name = "product_seq",
        strategy
        "org.hibernate.id.enhanced.SequenceStyleGenerator",
        parameters = {
            @Parameter(name = "sequence_name", value =
"product_seq"),
            @Parameter(name = "INCREMENT", value = "1"),
            @Parameter(name = "MINVALUE", value = "1"),
            @Parameter(name = "MAXVALUE", value =
"2147483647"),
            @Parameter(name = "CACHE", value = "1")
        }
    )
    @Id

```

```

    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"product_seq")

    @Column(name = "id", nullable = false, updatable = false)
    private Integer id;
    @Column(name = "name", nullable = false)
    private String name;
    @Column(name = "code", nullable = false)
    private String code;
    @Column(name = "price", nullable = false)
    private Double price;
    @Column(name = "description")
    private String description;
    @Column(name = "photo")
    private byte[] photo;
    //связь 1-М от таблицы Товар к таблице Позиция заказа
    @JsonIgnore
    @OneToMany(mappedBy = "product", fetch = FetchType.LAZY)
    private List<Order_item> order_items = new
ArrayList<Order_item>();}

```

Приложение Б.4 Файл User.java

```

@Entity
@Data
@Table(name="users")
@NamedEntityGraphs({
    @NamedEntityGraph(name = "graph.orders",
        attributeNodes = {
            @NamedAttributeNode("orders"),
        }
    ))
public class User {
    @GenericGenerator(
        name = "users_seq",
        strategy =
"org.hibernate.id.enhanced.SequenceStyleGenerator",
        parameters = {
            @Parameter(name = "sequence_name", value =
"users_seq"),
            @Parameter(name = "INCREMENT", value = "1"),

```

```

        @Parameter(name = "MINVALUE", value = "1"),
        @Parameter(name = "MAXVALUE", value =
"2147483647"),
        @Parameter(name = "CACHE", value = "1")
    }

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"users_seq")
    @Column(name = "id", nullable = false, updatable = false)
    private Integer id;
    @Column(name = "last_name", nullable = false)
    private String last_name;
    @Column(name = "first_name", nullable = false)
    private String first_name;
    @Column(name = "middle_name")
    private String middle_name;
    @Column(name = "phone_number", nullable = false)
    private String phone_number;
    @Column(name = "e_mail", nullable = false)
    private String e_mail;
    @Column(name = "login", nullable = false)
    private String login;
    @Column(name = "user_password", nullable = false)
    private String user_password;
    //связь М-1 от таблицы Пользователь к таблице Роль_пользователя
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "user_role_id", insertable = false, updatable
= false)
    @JsonInclude(NON_NULL)
    private User_role user_role;
    @Column(name = "user_role_id")
    private Integer user_role_id;
    @Column(name = "street")
    private String street;

    @Column(name = "house")

```

```

    private String house;
    @Column(name = "flat")
    private String flat;
    @Column(name = "comment_by_admin")
    private String comment_by_admin;
    @Column(name = "reg_date", updatable = false)
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-mm-dd
hh:mm:ss")
    private LocalDateTime reg_date;
    //связь 1-М от таблицы Пользователь к таблице Заказ
    @JsonIgnore
    @OneToMany(mappedBy = "user", fetch = FetchType.LAZY,
orphanRemoval = true)
    private List<Order> orders = new ArrayList<Order>(); }

```

Приложение Б.5 Файл User_role.java

```

@Entity
@Data
@Table(name = "user_role")
@NamedEntityGraphs({
    @NamedEntityGraph(name = "graph.users",
        attributeNodes = {
            @NamedAttributeNode("users"),
        }
    )
})
public class User_role {
    @Id
    @Column(name = "id", nullable = false, updatable = false)
    private Integer id;
    @Column(name = "role_name", nullable = false)
    private String role_name;
    //связь 1-М от таблицы Роль_пользователя к таблице Пользователь
    @JsonIgnore
    @OneToMany(mappedBy = "user_role", orphanRemoval = true, fetch =
FetchType.LAZY)
    private List<User> users = new ArrayList<>();}

```

Приложение Б.6 Файл OrderRepo.java

```

@Repository

```

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

```

public interface OrderRepo extends JpaRepository<Order, Integer> {

    @EntityGraph(value = "graph.order_items.products", type =
EntityGraph.EntityGraphType.LOAD)

    @Query("select r from Order r where r.id=:id")

    Optional<Order> findOrderById(@Param("id") Integer id)}

```

Приложение Б.7 Файл Order_itemRepo.java

```

@Repository

public interface Order_itemRepo extends JpaRepository<Order_item,
Integer> {}

```

Приложение Б.8 Файл ProductRepo.java

```

@Repository

public interface ProductRepo extends JpaRepository<Product, Integer> {

    Product findByCode(String code);

    List<Product> findAllByName(String name);

    Product deleteByCode(String code);

    boolean existsByCode(String code);}

```

Приложение Б.9 Файл User.java

```

@Repository

public interface UserRepo extends JpaRepository<User, Integer> {

    @EntityGraph(value = "graph.orders", type =
EntityGraph.EntityGraphType.LOAD)

    @Query("select r from User r where r.id=:id")

    Optional<User> findUserById(@Param("id") Integer id);

    void deleteById(Integer id);

    boolean existsByLogin(String login);}

```

Приложение Б.10 Файл User_roleRepo.java

```

@Repository

public interface User_roleRepo extends JpaRepository<User_role,
Integer> {

    @EntityGraph(value = "graph.users", type =
EntityGraph.EntityGraphType.LOAD)

    @Query("select r from User_role r where r.id=:id")

    Optional<User_role> findUser_roleById(@Param("id") Integer id);}

```

Приложение Б.11 Файл UserService.java

```

@Service

public class UserService {

```

```

private final UserRepo repo;
private final User_roleRepo user_roleRepo;
public UserService(final UserRepo repo,
                   final User_roleRepo user_roleRepo) {
    this.repo = repo;
    this.user_roleRepo = user_roleRepo;
}
@Transactional
public User create(UserCreateReq req) {
    User user = new User();
    if (req.getLogin() == "" || req.getUser_password() == "") {
        throw new ApplicationException("Parameters can not be
null");
    }
    user.setLast_name(req.getLast_name());
    user.setFirst_name(req.getFirst_name());
    user.setMiddle_name(req.getMiddle_name());
    user.setPhone_number(req.getPhone_number());
    user.setE_mail(req.getE_mail());
    user.setLogin(req.getLogin());
    user.setUser_password(req.getUser_password());
    user.setSalt(req.getSalt());
    user.setUser_role_id(req.getUser_role_id());
    user.setStreet(req.getStreet());
    user.setHouse(req.getHouse());
    user.setFlat(req.getFlat());
    user.setComment_by_admin(req.getComment_by_admin());
    user.setReg_date(LocalDateTime.now());
    if (repo.existsByLogin(user.getLogin())) {
        throw new ApplicationException("Login already exists");
    }
    return repo.save(user);
}
@Transactional
public User update(Integer id, UserUpdateReq req) {
    if (!repo.existsById(id)) {
        throw new
ApplicationException("Not found");
    }
    User user = getById(id);
    user.setLast_name(req.getLast_name());

```

```

        user.setFirst_name(req.getFirst_name());
        user.setMiddle_name(req.getMiddle_name());
        user.setPhone_number(req.getPhone_number());
        user.setE_mail(req.getE_mail());
        user.setStreet(req.getStreet());
        user.setHouse(req.getHouse());
        user.setFlat(req.getFlat());
        user.setComment_by_admin(req.getComment_by_admin());
        return repo.save(user);    }

@Transactional
public void delete(Integer id) {
    repo.deleteById(id);    }

@Transactional
public void deleteAll() {
    repo.deleteAll();    }

public User getById(Integer id) {
    return repo.findById(id).orElseThrow(() -> new
ApplicationException("Not found"));    }

public List<User> getAll() {
    List<User> user = repo.findAll();
    if (user.isEmpty()) {
        throw new ApplicationException("Not found");    }
    return user;    }}

```

Приложение Б.12 Файл ProductService.java

```

@Service
public class ProductService {
    private final ProductRepo repo;
    public ProductService(final ProductRepo repo) {
        this.repo = repo;    }
    public Product getById(Integer id) {
        return repo.findById(id).orElseThrow(() -> new
ApplicationException("Not found"));    }
    public Product getByCode(String code) {
        Product product = repo.findByCode(code);
        if (product == null) {

```

```

        throw new ApplicationException("Not found");    }
    return product;    }

    public List<Product> getAllProductsByName(String name) {
        List<Product> product = repo.findAllByName(name);
        if (product.isEmpty()) {
            throw new ApplicationException("Not found");    }
        return product;    }

    public List<Product> getAll() {
        List<Product> product = repo.findAll();
        if (product.isEmpty()) {
            throw new ApplicationException("Not found");    }
        return product;    }

    @Transactional
    public Product create(ProductCreateReq req) {
        Product product = new Product();
        if (req.getName() == "" || req.getCode() == "") {
            throw new ApplicationException("Parameters can not be
null");    }
        product.setName(req.getName());
        product.setCode(req.getCode());
        product.setPrice(req.getPrice());
        product.setDescription(req.getDescription());
        if (repo.existsByCode(product.getCode())) {
            throw new ApplicationException("Code already exists");
        }

        return repo.save(product);    }

    @Transactional
    public Product update(Integer id, ProductUpdateReq req) {
        if (!repo.existsById(id)) {
            throw new ApplicationException("Not found");    }
        Product product = getById(id);
        product.setName(req.getName());
        product.setPrice(req.getPrice());
        product.setDescription(req.getDescription());
        return repo.save(product);    }

    @Transactional

```

```

public Product updateByCode(String code, ProductUpdateReq req) {
    if (!repo.existsByCode(code)) {
        throw new ApplicationException("Not found");
    }
    Product product = getByCode(code);
    product.setName(req.getName());
    product.setPrice(req.getPrice());
    product.setDescription(req.getDescription());
    return repo.save(product);
}

@Transactional
public void delete(Integer id) {
    if (!repo.existsById(id)) {
        throw new ApplicationException("Not found");
    }
    repo.deleteById(id);
}

@Transactional
public void deleteByCode(String code) {
    if (!repo.existsByCode(code)) {
        throw new ApplicationException("Not found element");
    }
    repo.deleteByCode(code);
}

```

Приложение Б.13 Файл OrderService.java

```

@Service
public class OrderService {
    private final OrderRepo repo;
    private final Order_itemRepo itemRepo;

    public OrderService(OrderRepo repo, Order_itemRepo itemRepo) {
        this.repo = repo;
        this.itemRepo = itemRepo;
    }

    @Transactional
    public Order_item createOrderItem(Order_itemCreateReq
itemCreateReq) {
        Order_item order_item = new Order_item();
        order_item.setOrders_id(itemCreateReq.getOrders_id());
        order_item.setProducts_id(itemCreateReq.getProducts_id());
        order_item.setCount_item(itemCreateReq.getCount_item());
        return itemRepo.save(order_item);
    }

    @Transactional

```

```

public Order create(OrderCreateReq req) {
    Order order = new Order();
    order.setUser_id(req.getUser_id());
    order.setOrder_num(req.getOrder_num());
    order.setStatus(req.getStatus());
    order.setOrder_sum((double)1440);
    order.setOrder_date(LocalDateTime.now());
    return repo.save(order);    }

public List<Order> getAll() {
    List<Order> orders = repo.findAll();
    if (orders.isEmpty()) {
        throw new ApplicationException("Not found");    }
    return orders;    }}

```

Приложение Б.14 Файл OrderController.java

```

@RestController
@RequestMapping("/orders")

public class OrderController {

    private final OrderRepo orderRepo;
    private final OrderService orderService;

    public OrderController(OrderRepo orderRepo, OrderService
orderService) {

        this.orderRepo = orderRepo;
        this.orderService = orderService;    }

    @PostMapping("/create")
    @ResponseStatus(HttpStatus.CREATED)
    @ResponseBody
    public Order create(@RequestBody OrderCreateReq req) {
        return orderService.create(req);    }

    @GetMapping
    @ResponseBody
    public List<Order> getAll() {
        return orderService.getAll();    }

    @GetMapping("/id:{id}")
    @ResponseBody
    public OrderItemProductResp getById(@PathVariable Integer id) {

```

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

```

OrderItemProductResp roleResp = new OrderItemProductResp();
Order order = orderRepo.findOrderById(id).get();
roleResp.setId(order.getId());
roleResp.setOrder_num(order.getOrder_num());
roleResp.setStatus(order.getStatus());
roleResp.setOrder_sum(order.getOrder_sum());
roleResp.setOrder_date(order.getOrder_date());
roleResp.setOrder_items(order.getOrder_items());
return roleResp;    }}

```

Приложение Б.15 Файл UserController.java

```

@RestController
@RequestMapping("/users")
public class UserController {
    private final UserService service;
    private final UserRepo userRepo;

    public UserController(final UserService service, UserRepo
userRepo) {
        this.service = service;
        this.userRepo = userRepo;    }

    @PostMapping("/create")
    @ResponseStatus(HttpStatus.CREATED)
    @ResponseBody
    public User create(@RequestBody UserCreateReq req) {
        return service.create(req);    }

    @PutMapping("/updateById:{id}")
    public User updateUser(@PathVariable Integer id, @RequestBody
UserUpdateReq req) {
        return service.update(id, req);    }

    @DeleteMapping("/deleteById:{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteUserById(@PathVariable Integer id) {
        service.delete(id);    }

    @DeleteMapping("/deleteAll")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteAllUsers() {

```

```

        service.deleteAll();    }

@GetMapping("/id:{id}")
@ResponseBody
public UserOrderResp getById(@PathVariable Integer id) {
    UserOrderResp roleResp = new UserOrderResp();
    User user = userRepo.findUserById(id).get();
    roleResp.setId(user.getId());
    roleResp.setLast_name(user.getLast_name());
    roleResp.setFirst_name(user.getFirst_name());
    roleResp.setMiddle_name(user.getMiddle_name());
    roleResp.setOrders(user.getOrders());
    return roleResp;    }

@GetMapping
@ResponseBody
public List<User> getAll() {
    return service.getAll();    }}

```

Приложение Б.16 Файл User_roleController.java

```

@RestController
@RequestMapping("/user_role")
public class User_roleController {
    private final User_roleService service;
    private final User_roleRepo user_roleRepo;

    public User_roleController(final User_roleService service,
        User_roleRepo user_roleRepo) {
        this.service = service;
        this.user_roleRepo = user_roleRepo;    }

    @GetMapping("/id:{id}")
    @ResponseBody
    public User_roleResp getById(@PathVariable Integer id) {
        User_roleResp roleResp = new User_roleResp();
        User_role role = user_roleRepo.findUser_roleById(id).get();
        roleResp.setId(role.getId());
        roleResp.setRole_name(role.getRole_name());
        roleResp.setUsers(role.getUsers());
        return roleResp;    }
}

```

```

@GetMapping
@ResponseBody
public List<User_roleResp> getAll() {
    List<User_role> user_roles = user_roleRepo.findAll();
    List<User_roleResp> user_roleResps = new ArrayList<>();
    for (User_role user_role: user_roles) {
        User_roleResp roleResp = new User_roleResp();
        User_role role =
user_roleRepo.findUser_roleById(user_role.getId()).get();
        roleResp.setId(role.getId());
        roleResp.setRole_name(role.getRole_name());
        roleResp.setUsers(role.getUsers());
        user_roleResps.add(roleResp);
    }
    return user_roleResps;
}

```

Приложение Б.17 Файл ProductController.java

```

@RestController
@RequestMapping("/product")
public class ProductController {
    private final ProductService service;
    public ProductController(final ProductService service) {
        this.service = service;
    }
    @PostMapping("/create")
    @ResponseStatus(HttpStatus.CREATED)
    @ResponseBody
    public Product create(@RequestBody ProductCreateReq req) {
        return service.create(req);
    }
    @GetMapping
    @ResponseBody
    public List<Product> getAll() {
        return service.getAll();
    }
    @GetMapping("/id:{id}")
    @ResponseBody
    public Product getById(@PathVariable Integer id) {
        return service.getById(id);
    }
    @GetMapping("/name:{name}")

```

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

```

    public List<Product> getAllProductsByName(@PathVariable String
name) {

        return service.getAllProductsByName(name);    }

    @GetMapping("/code:{code}")
    @ResponseBody
    public Product getByCode(@PathVariable String code) {

        return service.getByCode(code);    }

    @PutMapping("/updateById:{id}")

    public Product updateProduct(@PathVariable Integer id,
@RequestBody ProductUpdateReq req) {

        return service.update(id, req);    }

    @PutMapping("/updateByCode:{code}")

    public Product updateProductByCode(@PathVariable String code,
@RequestBody ProductUpdateReq req) {

        return service.updateByCode(code, req);    }

    @DeleteMapping("/deleteById:{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteProductById(@PathVariable Integer id) {

        service.delete(id);    }

    @DeleteMapping("/deleteByCode:{code}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteProductByCode(@PathVariable String code) {

        service.deleteByCode(code);    }}

```

Приложение Б.18 Файл Main.js

```

Vue.use(VueAwesomeSwiper)
Vue.config.productionTip = false
router.beforeResolve((to, from, next) => {
    if (to.name) {    NProgress.start()    }
    next()})
router.afterEach(() => {
    NProgress.done()})
// eslint-disable-next-line no-unused-vars
let vm = new Vue({    router,    store,    render: h =>
h(App)}).$mount('#app')
if (process.env.NODE_ENV === 'production') {
    register(`${process.env.BASE_URL}service-worker.js`, {

```

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

```

ready() {
    console.log(
        'App is being served from cache by a service
worker.\n' +
        'For more details, visit https://goo.gl/AFskqB'
    ),
    cached() {
        console.log('Content has been cached for offline use.')
    },
    updated() {
        console.log('New content is available; please refresh.')
        infoToaster('Please refresh', 'New content is available')
    },
    offline() {
        console.log('No internet connection found. App is running
in offline mode.')
    },
    error(error) {
        console.error('Error during service worker registration:',
error)
        errorToaster('Error during service worker registration',
error)
    }
})
export default new Router({
    mode: 'hash',
    base: process.env.BASE_URL,
    routes: [{
        path: '/',
        name: 'home',
        component: Home
    }, {
        path: '/about',
        name: 'about',
        component: () => import(/* webpackChunkName: "about" */
'./views/About.vue')
    }, {
        path: '/products',
        name: 'products',
        component: () => import('./views/Products.vue')
    }, {
        path: '/products/:id',
        name: 'productDetails',

```

```

        component: () =>
import('./components/products/ProductDetail.vue') },

    {
        path: '/cart',
        name: 'cart',
        component: () =>
import('./components/products/cart/CartProducts.vue'),
        beforeEnter: (to, from, next) => {
            if (isLoggedIn()) {
                next()
            } else {
                next({
                    name: 'login',
                    query: {
                        from: to.name } }) }, {
        path: '/checkout',
        name: 'checkout',
        component: () =>
import('./components/products/cart/Checkout.vue'),
        beforeEnter: (to, from, next) => {
            if (isLoggedIn()) {
                next()
            } else {
                next({
                    name: 'login',
                    query: {
                        from: to.name
                    } }) }, {
        path: '/login',
        name: 'login',
        component: () => import('./views/login.vue') ]})

var mongoose = require('mongoose')
var Schema = mongoose.Schema
var UserSchema = new Schema({
    firstName: String,
    middleName:String,
    lastName: String,

```

```

    phone:String,
    email: String,
    role: String,
    login: String,
    password: String,
    createdOn: String,
    street: String,
    house: String,
    flat: String,
    commentByAdmin: String}))

module.exports = mongoose.model('User', UserSchema)

var mongoose = require('mongoose')
var Schema = mongoose.Schema
var ProductSchema = new Schema({
    productName: String,
    productDescription: String,
    productPrice: String,
    productImage: String,
})

module.exports = mongoose.model('Product', ProductSchema)

const express = require('express')
const router = express.Router()

const UserModel = require('../mongo/model/user')

router.route('/login').post((req, res) => {
    UserModel.find({
        email: req.body.email,
        password: req.body.password
    }, function (err, user) {
        if (err) {
            res.send(err)
        }
        if (user.length === 0) {
            res.status(401).json({
                status: 401,
                message:'Unauthorized credentials mismatch' })
        } else {
            res.json(user)
        }
    })
})

```

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

```

module.exports = router
const express = require('express')
const router = express.Router()
const ProductModel = require('../mongo/model/product')
router.route('/products')
    // Create a new Product
    .post((req, res) => {
        const product = new ProductModel() // create a new instance of
        the Product model
        product.productName = req.body.productName
        product.productDescription = req.body.productDescription
        product.productPrice = req.body.productPrice
        product.productImage = req.body.productImage
        product.save(product, (err, product) => {
            if (err) { es.send(err) }
            console.log(product)
            res.send(product) }) })
    // Get All products
    .get((req, res) => {
        ProductModel.find((err, data) => {
            if (err) {
                res.send(err) }
            console.log('data', data)
            res.json(data) }) })
router.route('/products/similarProduct')
    .get(function (req, res) {
        ProductModel.find({
            productSeller: req.query['productSeller']
        }, function (err, product) {
            if (err) {
                res.send(err)
            }
            console.log('data', product)
            res.json(product)
        })).limit(4)
    })

```

```

    })

    router.route('/:product_id')

    // get the product with that id (accessed at GET
    http://localhost:8080/api/products/:product_id)

    .get(function (req, res) {

        ProductModel.findById(req.params.product_id, function (err,
        data) {

            if (err) {

                res.send(err) }

            console.log('data', data)

            res.json(data)

        })

    })

    // update the product with this id (accessed at PUT
    http://localhost:8080/api/products/:product_id)

    .put(function (req, res) {

        // use our product model to find the product we want

        ProductModel.findById(req.params.product_id, function (err,
        product) {

            if (err) {

                res.send(err)

            }

            product.productName = req.body.productName
            product.productDescription = req.body.productDescription
            product.productPrice = req.body.productPrice
            product.productImage = req.body.productImage

            product.save(function (err, data) {

                if (err) {

                    res.send(err)

                }

                console.log('Updating product', data)

                res.send(data)

            })

        })

    })

})

```

```

    // delete the product with this id (accessed at DELETE
    http://localhost:8080/api/products/:product_id)

    .delete(function (req, res) {
        ProductModel.remove({
            _id: req.params.product_id
        },
        function (err, product) {
            if (err) {
                res.send(err)
            }
            res.send(product)
        })
    })

module.exports = router
const express = require('express')
const router = express.Router()
const UserModel = require('../mongo/model/user')
/* GET api listing. */
router.get('/', (req, res) => {
    res.send('api works')
})
router.route('/users')
    // Create a new User
    .post((req, res) => {
        const user = new UserModel() // create a new instance of the
        User model
        user.firstName = req.body.firstName
        user.lastName = req.body.lastName
        user.fullName = req.body.firstName + " " + req.body.lastName
        user.password = req.body.password
        user.email = req.body.email
        user.isAdmin = false
        user.createdOn = new Date().toLocaleString()
        // save the bear and check for errors
        user.save(user, (err, user) => {
            if (err) {

```

```

        res.send(err)
    }
    res.send(user)
  })
})

// Get All Users
.get((req, res) => {
  UserModel.find((err, data) => {
    if (err) {
      res.send(err)
    }
    res.json(data)
  })
})

router.route('/users/:user_id')
  // get the user with that id (accessed at GET
  http://localhost:8080/api/users/:user_id)
  .get(function (req, res) {
    UserModel.findById(req.params.user_id, function (err, data) {
      if (err) {
        res.send(err)
      }
      res.json(data)
    })
  })

  // update the user with this id (accessed at PUT
  http://localhost:8080/api/users/:user_id)
  .put(function (req, res) {
    // use our user model to find the user we want
    UserModel.findById(req.params.user_id, function (err, user) {
      if (err) {
        res.send(err)
      }
      user.name = req.body.name
      user.password = req.body.password
      user.email = req.body.email
    })
  })
})

```

```

        user.isAdmin = false
        user.user_avatar = req.body.user_avatar
        user.phoneNumber = req.body.phoneNumber
        user.createdOn = req.body.createdOn
        user.save(function (err, data) {
            if (err) {
                res.send(err)
            }
            res.send(data)
        })
    })
})

module.exports = router

```

Приложение Б.19 Файл Index.html

```

<html>

  <head>

    <meta charset = "utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Крендель</title>

    <!--[if lt IE 9]>

    <script
    src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script
    >

    <script
    src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

    <![endif]-->

    <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">

  </head>

  <link rel="stylesheet" type="text/css" href="css.css">

  <body>

    <table width=100% height=5% border="0" align=center cellpadding="0">

      <tr>

        <th>

          <a class="menu">Главная |</a>

```

```

<a href="cat.html" class="menu"> Каталог |</a>
<a href="pay.html" class="menu"> Оплата </a>
</th>
</tr>
</table>

<table width=100% height=20% border="0" align=center
cellpadding="0">
<tr>
<td wight=40% class="td1"height=70%>
<h1 class="gh1">Что такое «Крендель»?</h1>
<p class="pp1">Это нестерпимое желание угостить Вас вкусным
тортом!</p>
<p class="pp1">Это радость от Ваших восторженных отзывов!</p>
<p class="pp1">Это Ваши удивленные глаза при виде съедобного
искусства!</p>
<p class="pp1">Это натуральные ингредиенты, много любви и щепотка
юмора!</p>
</td>
<td wight=60% class="td2" height=70%> </td>
</tr>
</table>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js
"></script>

<script src="bootstrap/js/bootstrap.min.js"></script>

<script>
$(function() {
});
</script>
</body>
</html>

```

Приложение Б.20 Файл Pay.html

```

<html>
<head>
<meta charset = "utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Онлара</title>
<!--[if lt IE 9]>

<script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script
>

<script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
<![endif]-->

    <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
</head>

<link rel="stylesheet" type="text/css" href="css.css">
<body>

    <table width=100% height=5% border="0" align=center cellpadding="0">
        <tr>
            <th>
                <a href="index.html" class="menu">Главная |</a>
                <a href="cat.html" class="menu">Каталог |</a>
                <a class="menu">Оплата </a>
            </th>
        </tr>
    </table>

    <table width=100% height=100% border="0" align=center
cellpadding="0" class="bgpay">
        
        <tr>
            <td width=50%>
                
            </td>
            <td>
                <h1 class="gh2"><u>Оплата</u></h1>
                <p class="ppay">Оплата осуществляется после того, как сотрудник
магазина позвонит на указанный вами в профиле
номер телефона. С этого момента статус вашего заказа "подтвержден" и
вам необходимо оплатить всю сумму

```

на карту 1234 5678 9101 1112 Алексей Максимович Д. После обработки оплаты статус изменится

на "оплачено" и вам позвонят, чтобы уточнить, когда вы хотите забрать заказ.</p>

<p></p>

</td>

</tr>

</table>

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js
"></script>
```

```
<script src="bootstrap/js/bootstrap.min.js"></script>
```

```
<script>
```

```
$(function() {
```

```
});
```

```
</script>
```

```
</body>
```

```
</html>
```

Приложение Б.21 Файл Cart.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
<meta name="viewport" content="width=device-width, initial-
scale=1">
```

```
<title>Каталог товаров</title>
```

```
<link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">
```

```
<link rel="stylesheet" href="css.css">
```

```
<!--[if lt IE 9]>
```

```
<script
src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script
>
```

```
<script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
```

```
<![endif]-->
```

```
</head>
```

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						83
Изм.	Лист	№ докум.	Подпись	Дата		

```

<body>

  <table width=100% height=5% border="0" align=center cellpadding="0">
    <tr>
      <th>
        <a href="index.html" class="menu">Главная |</a>
        <a class="menu"> Каталог |</a>
        <a href="pay.html" class="menu"> Оплата </a>
      </th>
    </tr>
  </table>

<div class="container content">
  <div class="row">
    <div class="col-md-3">

    </div>
    <div class="col-md-12 products">
      <div class="row">
        <div class="col-sm-3">
          <div class="product">
            <div class="product-img">
              <a href="#"><img
src={{product.img}} alt=""></a>
            </div>
            <p class="product-title">
              <a href="#">{{product.name}}</a>
            </p>
            <p class="product-
price">{{product.price}}</p>
          </div>
        </div>
        <div class="col-sm-3">
          <div class="product">
            <div class="product-img">

```

```

src={{product.img}} alt=""></a>
</div>
<p class="product-title">
    <a href="#">{{product.name}}</a>
</p>
<p class="product-
price">{{product.price}}</p>
</div>
</div>
<div class="col-sm-3">
    <div class="product">
        <div class="product-img">
            <a href="#"><img
src={{product.img}} alt=""></a>
        </div>
        <p class="product-title">
            <a href="#">{{product.name}}</a>
        </p>
        <p class="product-
price">{{product.price}}</p>
        </div>
    </div>
    <div class="col-sm-3">
        <div class="product">
            <div class="product-img">
                <a href="#"><img
src={{product.img}} alt=""></a>
            </div>
            <p class="product-title">
                <a href="#">{{product.name}}</a>
            </p>
            <p class="product-
price">{{product.price}}</p>
            </div>
        </div>
    </div>
</div>

```

```

        </div>
    </div>
    <div class="row">
        <div class="col-sm-3">
            <div class="product">
                <div class="product-img">
                    <a href="#"><img
src={{product.img}} alt=""></a>
                </div>
                <p class="product-title">
                    <a href="#">{{product.name}}</a>
                </p>
                <p class="product-
price">{{product.price}}</p>
            </div>
        </div>
        <div class="col-sm-3">
            <div class="product">
                <div class="product-img">
                    <a href="#"><img
src={{product.img}} alt=""></a>
                </div>
                <p class="product-title">
                    <a href="#">{{product.name}}</a>
                </p>
                <p class="product-
price">{{product.price}}</p>
            </div>
        </div>
        <div class="col-sm-3">
            <div class="product">
                <div class="product-img">
                    <a href="#"><img
src={{product.img}} alt=""></a>
                </div>
                <p class="product-title">
                    <a href="#">{{product.name}}</a>

```

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						86
Изм.	Лист	№ докум.	Подпись	Дата		


```

        background-color:white;
        color:#A9A9A9;}
.tdg{vertical-align: top;
    text-align: left;
    border-collapse:collapse;
    color: white;
    border-collapse:collapse;
    border: 1px solid #FAB4D2;

}
.td1{background-color:#FAB4D2;
    padding: 15px 5px 22px 10px;
}
.td2{background-color:#FAB4D2;
    text-align: right;
}
.img0{ display: block;
    margin: 0 auto;
    text-align:center;}
.img1{ text-align:right;}
.gh1 {font-family: "Monotype Corsiva"; padding: 10px 1px 20px
25px;}
.pp1{font-family: " Monotype Corsiva"; font-size: 20px;
    line-height: 1.5;}
.content{ margin-top: 10px;}
.product { background: #fff none repeat scroll 0 0;
    border: 1px solid white;
    height: 390px;
    overflow: hidden;
    padding: 25px 15px;
    position: relative;
    text-align: center;
    transition: all 0.5s ease 0s;
    margin-bottom: 20px;}
.product:hover {box-shadow: 0 0 16px rgba(0, 0, 0, 0.5);}

```

```

.product-img { height: 200px;}
.product-title a {
    bottom: 60px;
    left: 0;
    position: absolute;
    width: 100%;
    color: #C0C0C0;
    font-size: 18px;
    font-weight: 500;
}.product-price {
    bottom: 15px;
    left: 0;
    position: absolute;
    width: 100%;
    color: #C0C0C0;
    font-size: 18px;
    font-weight: 500;
}
img {height: auto;
    max-width: 100%;}
.menu{    color:#A9A9A9;
    font-size: 20px;}.imgpay{
    text-align:left;
    padding: 1px 1px 10px 200px;
}
.gh2 {    text-align:left;
    color: black;
    padding: 1px 1px 1px 100px;}.ppay{
    color: black;
    padding: 10px 60px 10px 100px;
    font-size:18px;
}.bgpay {background-color:#fff0f5;
    border: 1px solid #fff0f5;
}

```

ПРИЛОЖЕНИЕ В

(Презентация)

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
КАФЕДРА «ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ ДЛЯ ОТСЛЕЖИВАНИЯ РАБОТЫ ЭЛЕКТРОННОЙ КОММЕРЦИИ



Выполнил студент группы 15ВВ1
Шеянов Н.Н.
Руководитель к.т.н., доцент каф. ВТ
Трокоз Д.А.

ПЕНЗА
2019

Актуальность работы

Технологии интернет проникли во все сферы жизни, в том числе в экономическую. В связи с этим для сохранения конкурентоспособного положения ни одна торговая фирма не может обойтись без интернет-магазина, из-за чего их количество возрастает с каждым годом.



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.03.01.29.001 ПЗ

Лист

91

Цели и задачи

Цель работы – создание клиент-серверного приложения для отслеживания работы электронной коммерции.

Задачи:

- изучить предметную область;
- проанализировать предъявляемые требования;
- выбрать средства и технологии для разработки;
- разработать программные модули;
- протестировать и отладить их работу.

Анализ требований к системе

- Ведение каталога товаров
- Процесс покупки
- Просмотр информации о фирме
- Хранение данных



Изм.	Лист	№ докум.	Подпись	Дата

Выбор языков, средств и технологий

Языки:

- Java
- JavaScript
- HTML
- CSS
- SQL

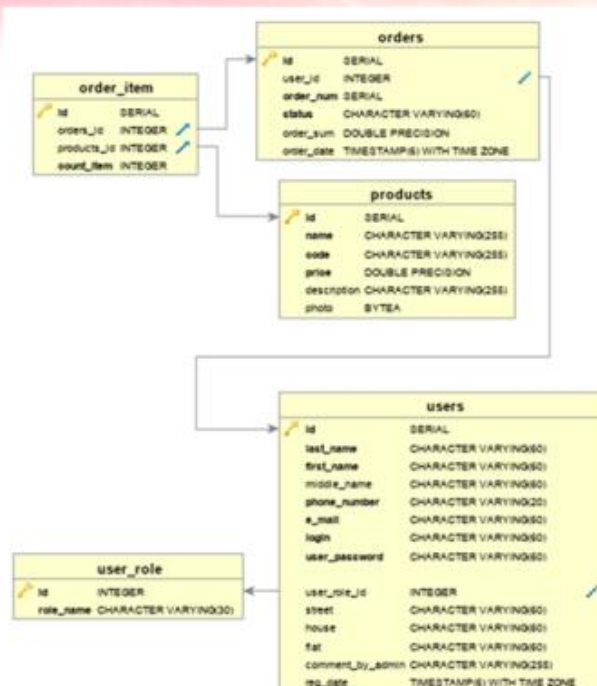


Средства и технологии:

- PostgreSQL
- pgAdmin4
- IntelliJ IDEA
- Spring Framework
- Hibernate
- Spring Security
- Vue.js
- Insomnia



Проектирование базы данных



Разработка серверной части



Разработка клиентской части



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.09.03.01.29.001 ПЗ

Лист

94

Тестирование

- Тестирование отдельных модулей
- Комплексное тестирование взаимодействия модулей
- Исправление обнаруженных ошибок
- Повторная отладка



Выводы

В результате проделанной работы:

- Разработано и отлажено клиент-серверное приложение;
- Составлены и реализованы алгоритмы работы интернет-магазина;
- Создано разграничение прав доступа;
- Оформлена соответствующая документация.

✎ ✎ ✎ ✎ ✎



Спасибо за внимание.

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						96
Изм.	Лист	№ докум.	Подпись	Дата		

ПРИЛОЖЕНИЕ Г

(Ведомость выпускной квалификационной работы)

					ПГУ 1.09.03.01.29.001 ПЗ	Лист
						97
Изм.	Лист	№ докум.	Подпись	Дата		

Формат	Обозначение	Наименование	Кол. лист.
		Документация текстовая	
A4	ПГУ 1.09.03.01.29.001 ПЗ	Пояснительная записка.	98
		Документация графическая	
A4	ПГУ 1.09.03.01. 29.002 Д1	Структура взаимодействия компонентов системы UML. Диаграмма последовательности.	1
A4	ПГУ 1.09.03.01. 29.003 Д2	Неавторизованный пользователь, авторизованный пользователь. UML диаграммы вариантов использования	1
A4	ПГУ 1.09.03.01. 29.004 Д2	Администратор магазина. UML диаграмма вариантов использования.	1
A4	ПГУ 1.09.03.01. 29.005 Д3	Оформление заказа. UML диаграмма деятельности.	1
A4	ПГУ 1.09.03.01. 29.006 Д4	Обработка заказа. UML диаграмма деятельности.	1

					ПГУ 1.09.03.01.29.001 ВП				
Изм.	Лист	№ докум.	Подпись	Дата					
Разраб.		Шеянов Н.Н.			Клиент-серверное приложение для отслеживания работы электронной коммерции Ведомость выпускной квалификационной работы	Лит.	Лист	Листов	
Провер.		Трокоз Д.А.					1	1	
						ФВТ гр. 15ВВ1			
Н. Контр.		Кучин А.В.							
Утв.		Митрохин М.А.							