

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

<i>Факультет</i>	<i>Кафедра</i>
вычислительной техники	МОиПЭВМ
Направление подготовки	09.03.04 «Программная инженерия»
Направленность	Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

Программные средства управления заказами в
приложении электронной коммерции по
продаже 3d-продукции

Студент	_____	<u>Дакнис Андрей Артурасович</u>
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	<u>Балашова И.Ю.</u>
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	<u>Такташкин Д.В.</u>
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	<u>Козлов А.Ю.</u>
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	<u>Попова Н.А.</u>
	(подпись)	(фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«25» декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Дакнис Андрей Артурасович гр. 20ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программные средства управления заказами в приложении электронной
коммерции по продаже 3d-изделий

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

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

1) *Назначение: Программные средства управления заказами в приложении
электронной коммерции по продаже 3d изделий.*

2) *Среда выполнения: веб-браузеры (Google Chrome, Mozilla Firefox и т. д.)*

Языковые средства разработки: TypeScript, Java, SQL, UML

Инструментальные средства: WebStorm, IntelliJ IDEA, PostgreSQL, Draw.io

Технологии разработки: ООП, сервис-ориентированный подход.

3) *Основные функции приложения:*

а) авторизация и регистрация;

б) просмотр каталога 3d-изделий;

в) поиск и сортировка товаров в каталоге товаров;

г) формирование корзины и осуществление типового заказа;

д) оформление индивидуального заказа.

5. Перечень подлежащих разработке вопросов:

1) *Анализ предметной области и требований к системе;*

2) *Проектирование программных средств;*

3) *Реализация программных средств;*

4) Тестирование программных средств;

5) Оформление пояснительной записки.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 79 листов, 22 рисунка, 14 таблиц, 21 использованный источник, 4 приложения.

ВЕБ-ПРИЛОЖЕНИЕ, ПРОДАЖА 3D-ИЗДЕЛИЙ, GOOGLE CHROME, TYPESCRIPT, REACT, БАЗА ДАННЫХ, UML, WEBSTORM, POSTGRESQL

Объектом разработки являются веб-приложение продажи 3d-продукции

Цель работы – разработка программных средств управления заказами в приложении электронной коммерции по продаже 3d-продукции.

Разработка базы данных проводилась в PostgreSQL на языке SQL.

Разработка приложения проводилась на языках TypeScript и Java в средах разработки WebStorm и IntelliJ IDEA в операционной системе Windows 10. Приложение предназначено для использования в веб-браузерах, таких как Google Chrome, Mozilla Firefox, Opera GX, Microsoft Edge и т. д.

					ПГУ 09.03.04 – 08BP201.05 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Программные средства управления заказами в приложении электронной коммерции по продаже 3d-продукции Пояснительная записка	Лит.	Лист	Листов
Разраб.		Дакнис А.А.						
Пров.		Балашова И.Ю.					4	79
Реценз.						Группа 20ВП1		
Н. контр.		Такташкин Д.В						
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ веб-приложений электронной коммерции по продаже 3d-изделий.....	9
1.1 Анализ предметной области и постановка задачи	9
1.2 Обзор программных веб-приложений по продаже 3d-изделий.....	9
1.2.1 Веб-приложение «Asteri-3D»	9
1.2.2 Веб-приложение «VolPrint»	10
1.2.3 Веб-приложение «Lab3DPrint»	11
1.2.4 Веб-приложение «Cubicprints»	12
1.2.5 Сравнение веб-приложений электронной коммерции по продаже 3d-изделий.....	13
1.3 Анализ функциональных требований	15
1.4 Анализ нефункциональных требований	18
2 Проектирование приложения электронной коммерции по продаже 3d-продукции	20
2.1 Архитектура приложения.....	20
2.2 Проектирование структур данных	20
2.2.1 Концептуальная модель базы данных	20
2.2.2 Логическая модель базы данных.....	22
2.3 Проектирование пользовательского интерфейса	22
2.4 Проектирование программных средств.....	25
3 Реализация веб-приложения электронной коммерции по продаже 3d-изделий.....	27
3.1 Реализация базы данных	27
3.2 Реализация серверных компонентов.....	29
4 Контроль качества программного обеспечения.....	34
4.1 Выбор режима тестирования	34
4.2 Функциональная диаграмма и таблица решений	35
4.3 Метрики кода.....	39

4.3.1 Метрика SLOC	39
4.3.2 Цикломатическая сложность	40
4.3.3 Метрика Холстеда.....	42
4.4.4 Метрика Чепина	44
4.5 Планирование разработки и оценка бюджета.....	45
Заключение	47
Список использованных источников	48
Приложение А (справочное) Глоссарий	50
Приложение Б (рекомендуемое) Графическая часть.....	52
Приложение В (рекомендуемое) Результаты тестирования приложения	56
Приложение Г (рекомендуемое) Листинг веб-приложения	59

Введение

3d-печать – это создание детали путем наслоения материала в один слой за один раз. Для печати используют 3d-принтеры. Под разные виды производства существуют разные виды 3d-принтеров. Печатающая головка принтера перемещается по траектории и наносит расплавленный материал наращивая деталь слой за слоем. По мере застывания слои склеиваются между собой. В результате получается цельное изделие. Способ позволяет производить детали практически любой геометрии. В основном для печати используют пластики и металлы. Пластики более популярны, они применяются практически в любой индустрии [1].

За последние годы отрасль 3d-печати проникла во многие сферы производства. Не стала исключением и сфера продажи продуктов 3d-печати. Лучше и удобнее всего реализовывать продажу 3d-изделий через интернет-магазины из-за преимуществ в виде большего охвата аудитории потенциальных покупателей, а также гораздо меньшей стоимости поддержки работы предприятия. Поэтому разработка приложения электронной коммерции для продажи 3d-изделий является актуальной.

Объектом исследования является приложение электронной коммерции.

Предметом исследования являются программные средства управления заказами в приложении электронной коммерции по продаже 3d-продукции.

Целью выпускной квалификационной работы является разработка программных средств управления заказами в приложении электронной коммерции по продаже 3d-продукции.

Для достижения необходимой цели необходимо выполнить следующие задачи:

- провести сравнительный анализ программных продуктов-аналогов и выделить основные результаты;
- провести анализ требований к разрабатываемому программному продукту;

- осуществить проектирование программных средств и структур данных разрабатываемого программного продукта;
- провести планирование и расчет бюджета разрабатываемого программного продукта;
- осуществить разработку программного продукта;
- провести тестирование разработанного программного средства;
- оформить пояснительную записку в соответствии с требованиями, предъявляемыми к выпускной квалификационной работе.

В работе использованы методы сравнительного анализа, а также методы проектирования и разработки программного обеспечения.

1 Анализ веб-приложений электронной коммерции по продаже 3d-изделий

1.1 Анализ предметной области и постановка задачи

Электронная коммерция – это вид бизнеса, связанный с продажей товаров и услуг с помощью интернета. В нее входят все финансовые операции, сделки, бизнес-процессы [2].

В качестве одного из основных преимуществ можно выделить снижение издержек. Продажи онлайн более экономически выгодны, поскольку не требуют наличия офлайн-точек, а, следовательно, аренды и дополнительного персонала – расходов на инфраструктуру.

Кроме того, все виды современной электронной коммерции могут обеспечить круглосуточную доступность. Благодаря этому клиенты имеют возможность совершать покупки и заказывать услуги в любое удобное для них время, независимо от географического местонахождения. Покупатели могут делать покупки, находясь дома или в дороге. Это удобно и экономит время.

Также электронная коммерция обеспечивает расширенные возможности маркетинга. Она активно использует инструменты, позволяющие эффективно продвигать товары и услуги и привлекать новых покупателей.

Кроме того, электронная коммерция позволяет значительно расширить целевую аудиторию и улучшить сервисный опыт для клиентов, предоставляя различные варианты доставки и способы оплаты [3].

Для определения требований к разрабатываемому веб-приложению проведем сравнительный анализ некоторых приложений электронной коммерции.

1.2 Обзор программных веб-приложений по продаже 3d-изделий

1.2.1 Веб-приложение «Asteri-3D»

Приложение «Asteri-3D» является примером простого веб-приложения для продажи 3d-изделий. Интерфейс приложения приведен на рисунке 1.

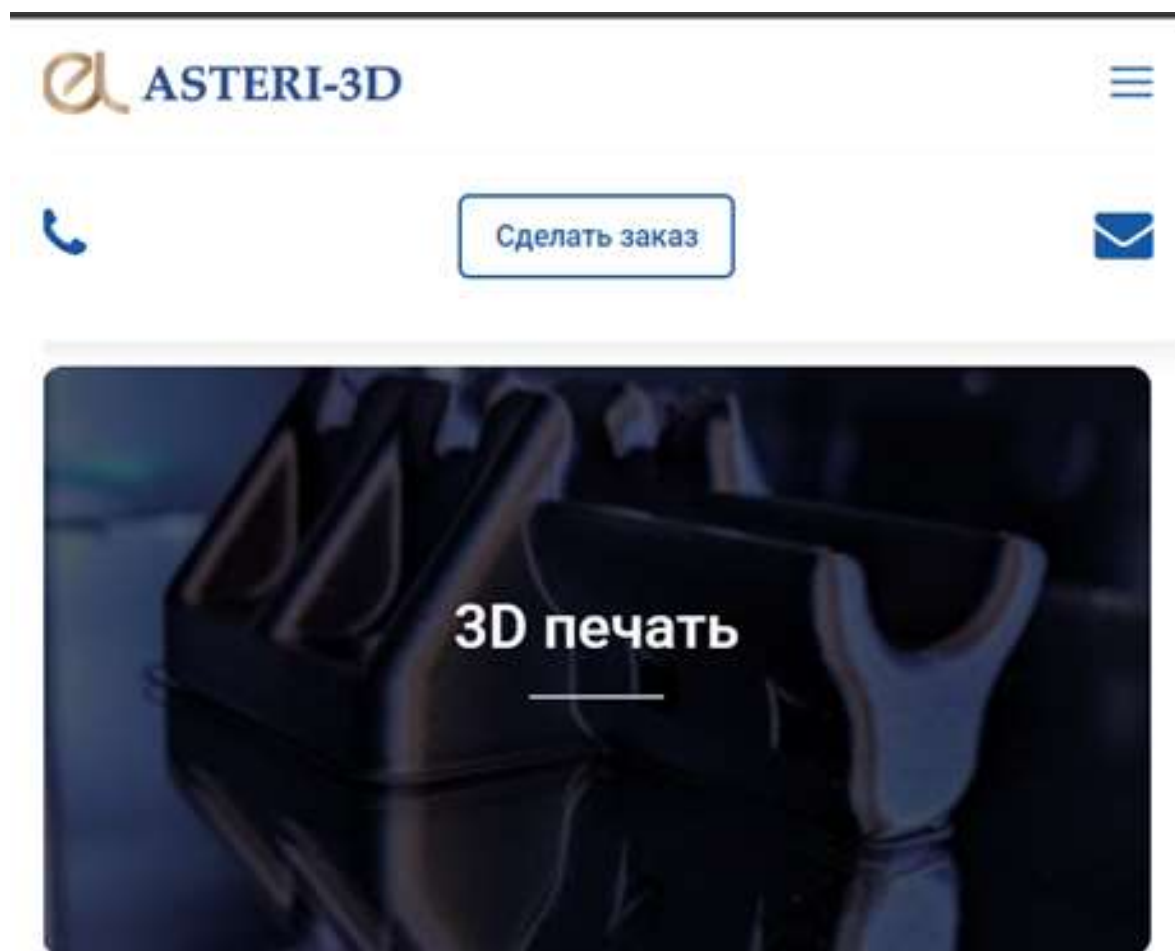


Рисунок 1 – Интерфейс веб-приложения «Asteri-3D»

Среди достоинств данного приложения можно выделить то, что оно предоставляет пользователю возможность заказа 3d-изделия по индивидуальным пожеланиям клиента с выбором типа изделия и условий эксплуатации.

Главным недостатком с точки зрения пользователя можно считать отсутствие системы аккаунтов как таковой, а также корзины.

1.2.2 Веб-приложение «VolPrint»

Компания-обладатель данного веб-приложения оказывает услуги 3d-печати, а также занимается поставками расходных материалов [4]. Данное веб-приложение имеет интерфейс, который представлен на рисунке 2.

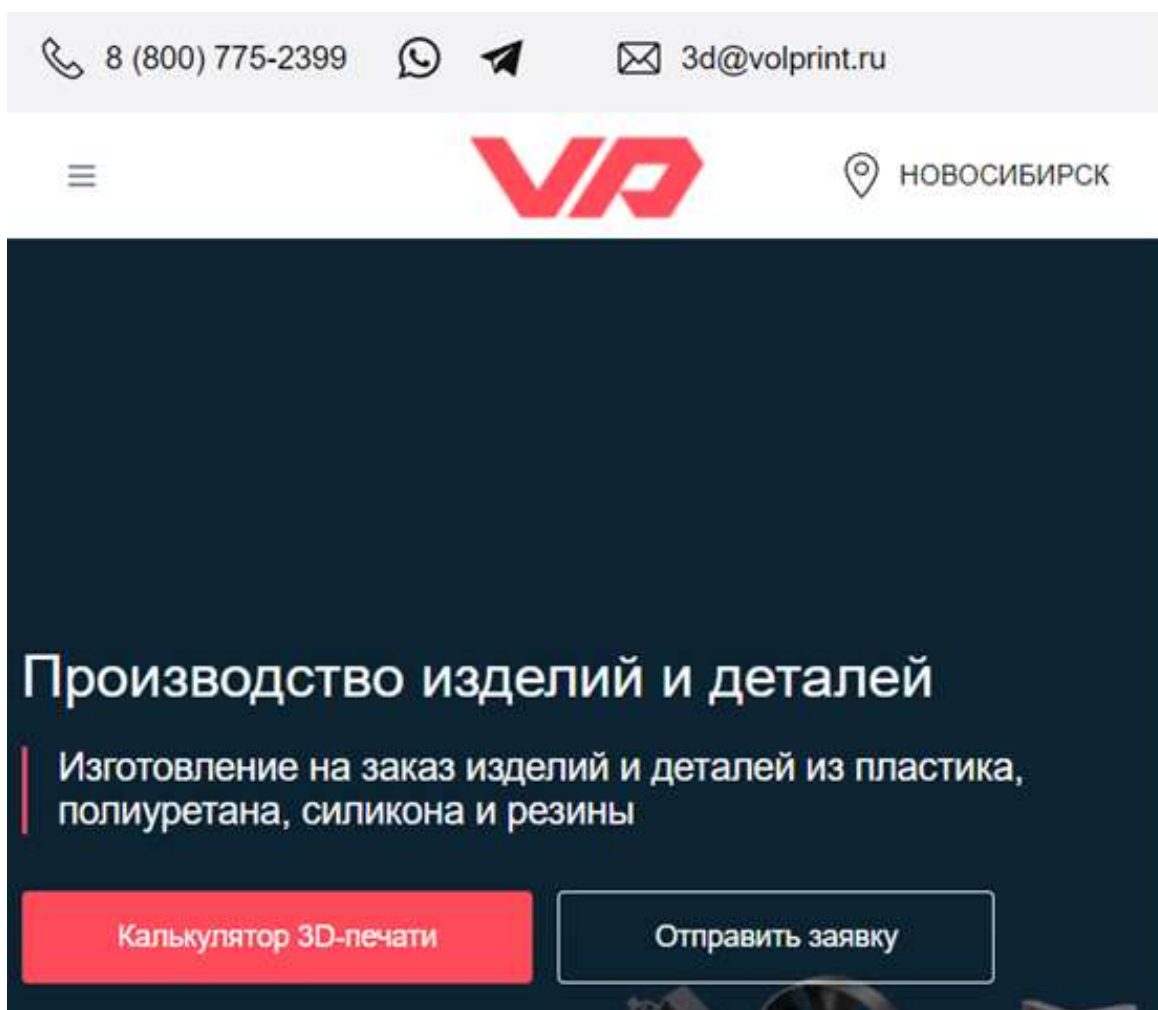


Рисунок 2 – Интерфейс веб-приложения «VolPrint»

Среди достоинств этого приложения можно выделить то, что оно позволяет пользователю работать с аккаунтами, оформлять индивидуальные заказы с возможностью моментально подсчитать цену услуги.

Недостатком этого приложения является отсутствие каталога 3d-изделий, хотя здесь и представлены к продаже компоненты для продажи расходного материала для 3d-печати.

1.2.3 Веб-приложение «Lab3DPrint»

Компания-обладатель данного веб-приложения является центром услуг в сфере инновационных трехмерных технологий [5]. Интерфейс ее веб-приложения представлен на рисунке 3.



Рисунок 3 – Интерфейс веб-приложения «Lab3DPrint»

Главным достоинством веб-приложения является то, что оно позволяет пользователю выбрать множество разных подходов к печати изделия, изготавливаемого на заказ.

Основным недостатком является то, что отсутствует функционал, который касается каталога. Также можно выделить в этот пункт и то, что веб-приложение не имеет системы аккаунтов.

1.2.4 Веб-приложение «Cubicprints»

«Cubicprints» – веб-приложение для изготовления изделий на заказ. Его интерфейс представлен на рисунке 4.

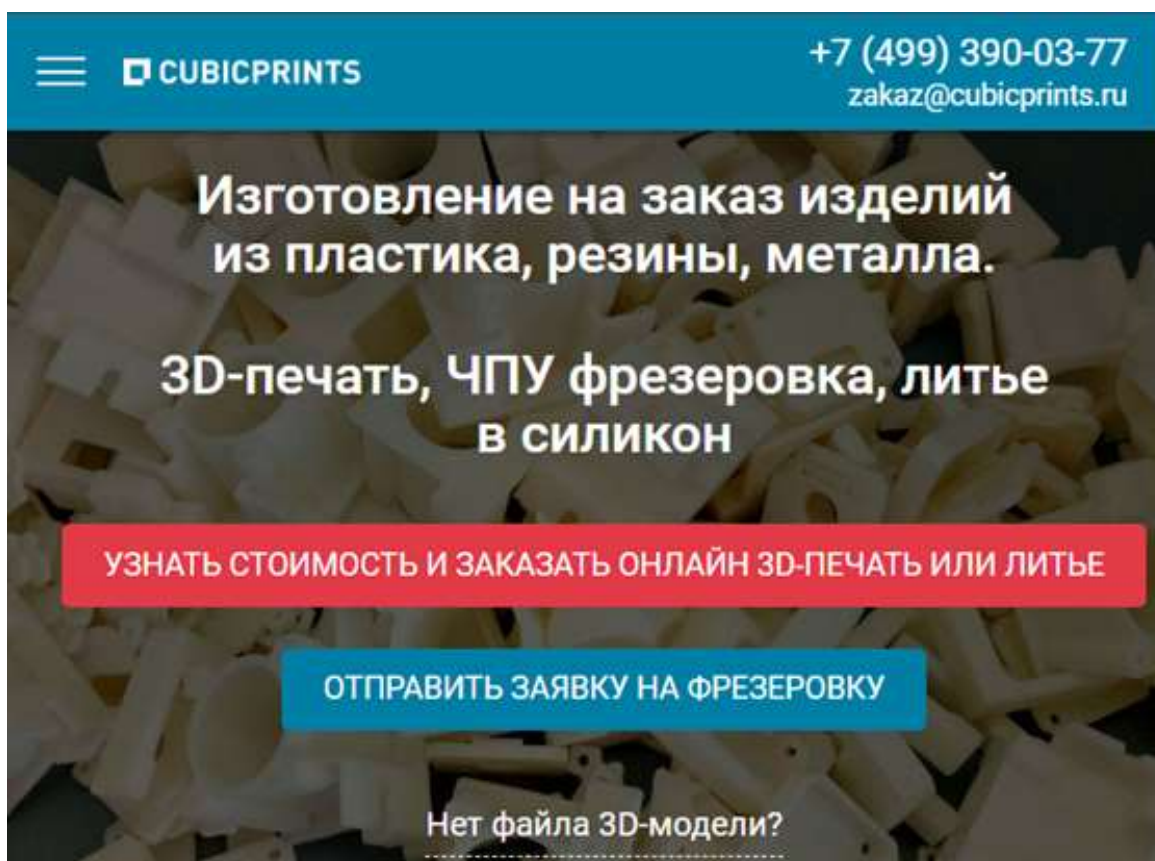


Рисунок 4 – Интерфейс приложения «cubicprints»

Ресурс позволяет пользователю заказать изготовление 3d-изделия по его модели с выбором состава и типа пластика.

К недостаткам можно отнести отсутствие функционала продажи готовых товаров, а также неспособность оформить заказ без предоставления файлов желаемой модели.

1.2.5 Сравнение веб-приложений электронной коммерции по продаже 3d-изделий

Проведем сравнительный анализ рассмотренных выше веб-приложений. Для оценки приложений будет использоваться трехбалльная шкала оценки, в которой 0 означает «полное отсутствие реализации», 1 – «частичную реализацию», а 2 – «полную реализацию» рассматриваемой функции.

Критерии были выбраны на основании дружелюбности к пользователю, а также удобства интерфейса.

Результаты сравнения приложений электронной коммерции по продаже 3d-изделий представлены в таблице 1.

Таблица 1 – Сравнение программных средств электронной коммерции по продаже 3d-изделий

Критерий \ Приложение	Asteri-3D	VolPrint	Lab3DPrint	cubicprints
Наличие регистрации и авторизации	1	2	0	0
Наличие каталога готовых 3d-изделий	0	0	0	0
Возможность фильтрации и поиска в каталоге изделий	0	0	0	0
Возможность изменения данных пользователя	0	2	0	0
Наличие корзины	0	2	0	0
Управление содержимым корзины	0	2	0	0
Возможность создания товара под заказ	2	2	2	2
Возможность учета пожеланий пользователя при создании товара под заказ	2	2	2	0
Итого	5	12	4	2

Подводя итог сравнения можно сделать вывод, что ни одно из рассмотренных приложений не соответствует объявленным критериям в полной мере. Среди всех продуктов-аналогов большее количество баллов

набрал сервис компании «VolPrint», однако у них отсутствует каталог для продажи 3d-изделий.

В результате отсутствия такого элемента пользователю не будет предоставлен необходимый функционал для удобного и быстрого заказа желаемого товара, что в свою очередь может повлечь за собой отток потенциальных покупателей из веб-приложения.

В результате принято решение разработать собственное программное средство, которое будет включать в себя лучшие решения от рассмотренных приложений и будет лишено наиболее существенных недостатков.

Исходя из данной таблицы, к разрабатываемому приложению были выдвинуты следующие дополнительные требования:

- приложение должно обладать каталогом готовой 3d-продукции с дополнительным функционалом;
- приложение должно предоставлять возможность входа, авторизации и изменения данных;
- приложение должно предоставлять пользователям возможность создания товара под заказ при учете их пожеланий.

1.3 Анализ функциональных требований

Обобщив сказанное выше, к разрабатываемому веб-приложению были выделены следующие функциональные требования:

- обеспечение работы каталога;
- обеспечение работы поиска и сортировки в каталоге;
- оформление корзины с возможностью ее редактирования;
- обеспечение оформления заказа 3d-изделий из каталога пользователем;
- обеспечение оформления индивидуального заказа пользователем;
- обеспечение просмотра активных заказов к выполнению мастером 3d-печати;
- обеспечение отправки активных заказов мастеру 3d-печати.

В соответствии с выявленными требованиями была разработана диаграмма вариантов использования UML (use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей [6]. Диаграмма вариантов использования для разрабатываемого веб-приложения представлена на рисунке 8.

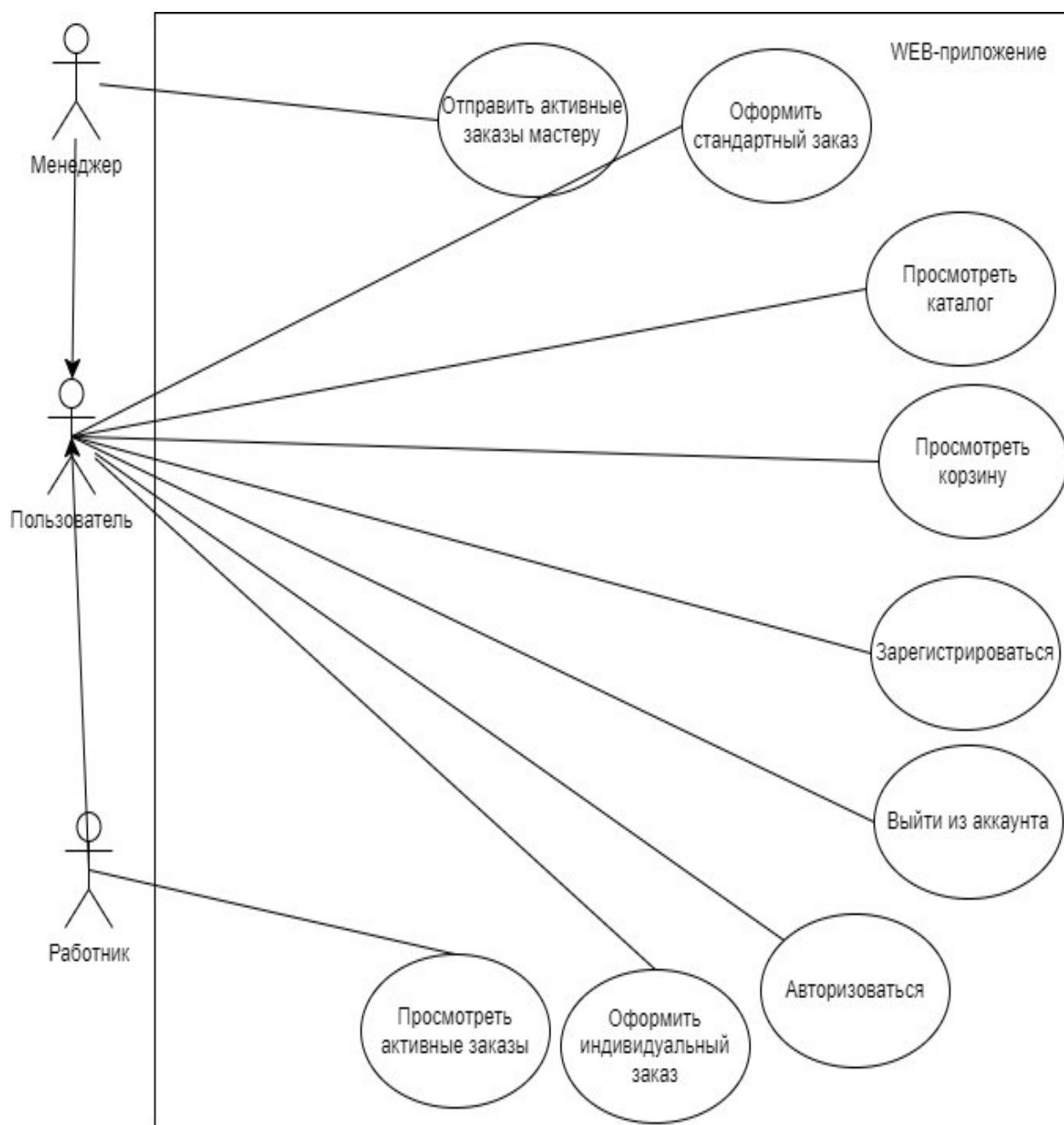


Рисунок 8 – Диаграмма вариантов использования

Сценарий варианта использования «Оформить стандартный заказ» представлен на рисунке 9.

Прецедент: Оформить стандартный заказ
ID: 1
Краткое описание: Пользователь оформляет стандартный заказ
Главные актеры: Пользователь
Предусловие 1. Пользователь вошел в аккаунт 2. Пользователь выбрал товары к покупке
Основной поток: 1. Пользователь заходит на страницу корзины и нажимает кнопку «Оформить заказ». 2. Посылается запрос отправки заказа 3. Система сохраняет заказ в базу данных
Постусловие Отсутствует

Рисунок 9 – Сценарий варианта использования «Оформить стандартный заказ»

Сценарий варианта использования «Войти в аккаунт» изображен на рисунке 10.

Прецедент: Войти в аккаунт
ID: 2
Краткое описание: Пользователь входит в аккаунт
Главные актеры: Пользователь
Предусловие 1. Имеется созданный аккаунт
Основной поток: 1. Если пользователь вводит неправильный номер телефона или пароль 1.1. Приложение выдает предупреждение 2. Иначе 2.1. Пользователь входит в свой аккаунт
Постусловие Отсутствует

Рисунок 10 – Сценарий варианта использования «Войти в аккаунт»

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования были представлены их спецификации, которые позволяют отобразить последовательность действий пользователя и системы.

1.4 Анализ нефункциональных требований

Разработка базы данных будет проводиться с использованием PostgreSQL на языке SQL. База данных будет размещена на удаленном сервере.

PostgreSQL – это объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных [7].

Клиентская часть веб-приложения будет написана на языке TypeScript с использованием фреймворка React в интегрированной среде разработки JetBrains WebStorm.

TypeScript – это строго типизированный язык программирования, построенный на JavaScript. Разработчики добавили в него дополнительные возможности, такие как статическая типизация, классы и модули, чтобы создавать более надежные и поддерживаемые программы.

TypeScript позволяет выявлять ошибки на этапе разработки, облегчает совместную работу в команде и улучшает производительность разработки в больших проектах. Он преобразуется в обычный JavaScript, поэтому код, написанный на TypeScript, может выполняться в любом современном браузере или окружении, поддерживающем JavaScript [8].

React.js – это библиотека для языка программирования JavaScript с открытым исходным кодом для разработки пользовательских интерфейсов. Она помогает быстро и легко реализовать реактивность – явление, когда в ответ на изменение одного элемента меняется все остальное [9].

JetBrains WebStorm – интегрированная среда разработки на JavaScript, CSS & HTML от компании JetBrains, разработанная на основе платформы

IntelliJ IDEA. WebStorm обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями [10].

Серверная часть будет написана на языке Java при использовании фреймворка Spring в интегрированной среде разработки JetBrains IntelliJ IDEA.

Java – язык программирования общего назначения. Он используется для написания кода, который может выполняться на разных платформах: компьютерах, мобильных устройствах и серверах. Java входит в тройку самых популярных языков программирования. Java опережает по популярности PHP, C#, C++, TypeScript и другие востребованные языки, а уступает только JavaScript и Python [11].

Spring – это фреймворк с открытым исходным кодом для языка программирования Java. Он был создан для упрощения разработки и поддержки масштабируемых, слабосвязанных и повторно используемых приложений. Фреймворк нужен, чтобы разработчикам было легче проектировать и создавать приложения. Spring не связан с конкретной парадигмой или моделью программирования, поэтому его могут использовать как каркас для разных видов приложений [12].

IntelliJ IDEA – интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains [13].

Веб-приложение должно быть отлажено и успешно работать под управлением любого современного браузера (Google Chrome, Mozilla Firefox, Opera GX, Microsoft Edge и т. д.).

2 Проектирование приложения электронной коммерции по продаже 3d-продукции

2.1 Архитектура приложения

В данном проекте будет использована клиент-серверная архитектура, схема которой представлена на рисунке 11.



Рисунок 11 – Архитектура «клиент-сервер»

На сервере будет расположена база данных. Клиентское приложение будет производить обращения к серверу и предоставлять пользователю удобный интерфейс для просмотра и взаимодействия с данными из базы [14].

2.2 Проектирование структур данных

Для взаимодействия между компонентами веб-приложения необходима следующая информация о конкретном 3d-изделии:

- название;
- рейтинг товара;
- изображение;
- цена;
- цвет;
- код цвета.

Также необходимо дополнительное поле – количество товаров. Оно необходимо для группировки одинаковых изделий.

2.2.1 Концептуальная модель базы данных

На основе анализа предметной области и результатов моделирования бизнес-процессов была составлена концептуальная модель базы данных в виде

диаграммы «сущность-связь». Концептуальная модель представлена на рисунке 12.



Рисунок 12 – Концептуальная модель базы данных предметной области
«Приложение электронной коммерции по продаже 3d-изделий»

На основе концептуальной модели базы данных были описаны сущности. Результаты приведены в таблице 2.

Таблица 2 – Описание сущностей

Наименование сущности	Идентификатор сущности	Назначение сущности
Пользователь	User	Содержит сведения о пользователях приложения, а также о работниках
Заказ	Order	Содержит сведения о заказах, совершенных пользователями
Заказанные товары	OrderedItems	Содержит сведения о заказанных пользователями товарах
Предмет заказа	Item	Содержит сведения о предмете заказа

Продолжение таблицы 2

Наименование сущности	Идентификатор сущности	Назначение сущности
Дополнительные данные о заказе	Item	Содержит дополнительные данные о заказе
Цвет товара	Colors	Содержит данные о цвете товара
Пластик	Plastic	Содержит данные о пластике
Предмет каталога	Items	Содержит сведения о лоте каталога
Статистика заказа	Orders	Содержит статистику заказа

Концептуальная модель была построена в среде draw.io в нотации IDEF1X (Integration DEFinition for information modeling) [15].

2.2.2 Логическая модель базы данных

Логическая модель данных – это расширение концептуальной модели данных. Она включает в себя все сущности, атрибуты, ключи и взаимосвязи, которые представляют бизнес-информацию и определяют бизнес-правила.

На основании концептуальной модели была сконструирована логическая модель базы данных изображенная на рисунке Б.1 в приложении Б. Логическая модель построена в среде draw.io в нотации IDEF1X.

2.3 Проектирование пользовательского интерфейса

На основе выявленных функциональных требований разработан интерфейс веб-приложения. Он включает в себя страницы входа, регистрации, каталога, корзины, а также список активных заказов к выполнению. Самые главные интерфейсы представлены ниже.

Страница каталога представлена на рисунке 13. На ней имеются следующие элементы:

- 1 – поле поиска товаров;
- 2 – поле указания сортировки;
- 3 – поле самого товара;
- 4 – кнопка выбора цвета товара;
- 5 – кнопка добавления товара в корзину;
- 6 – блок навигации по приложению.

При нажатии на кнопку «Купить» система проверяет, сколько товаров этого наименования пользователь уже добавил в корзину. Если такого товара добавлено не было, то появится соответствующий элемент корзины. Иначе счетчик увеличится на 1.

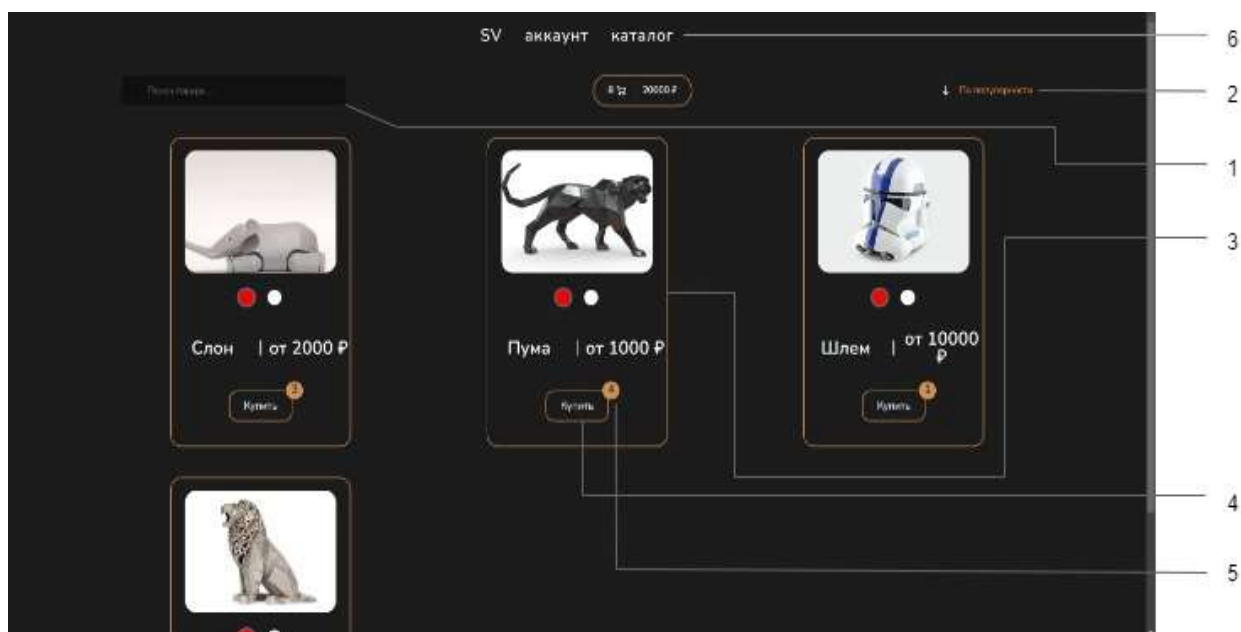


Рисунок 13 – Страница каталога

Страница корзины представлена на рисунке 14. На ней имеются следующие элементы:

- 1 – блок навигации по приложению;
- 2 – поле очистки корзины;
- 3 – поле добавленного товара;
- 4 – кнопка убавления количества товара на 1;

- 5 – кнопка прибавления количества товара на 1;
- 6 – кнопка удаления данного добавленного товара;
- 7 – кнопка «Вернуться в каталог»;
- 8 – кнопка «Оформить заказ».

При нажатии кнопки убавления количества товара на 1 система проверяет, есть ли в поле товара более одного предмета. Если есть, то при нажатии на кнопку количество снизится на 1. Иначе кнопка заблокируется и взаимодействие с ней станет невозможным.

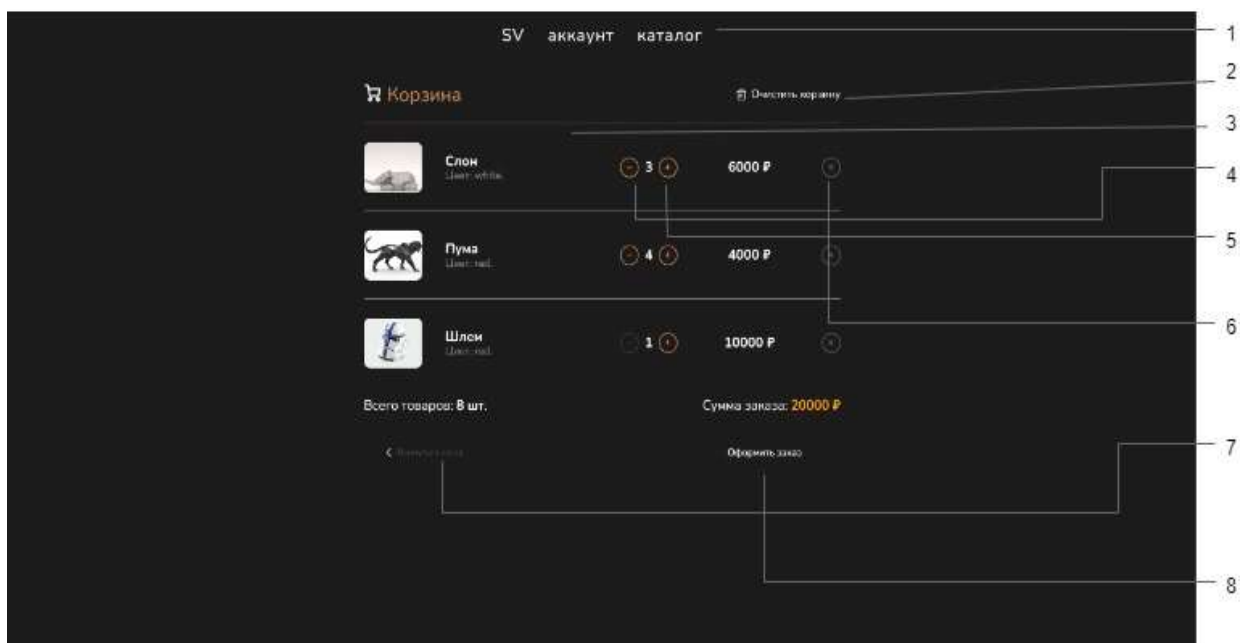


Рисунок 14 – Страница корзины

Страница управления заказами представлена на рисунке 15. На ней имеются следующие элементы:

- 1 – блок навигации по приложению;
- 2 – поле данных аккаунта;
- 3 – поле данных аккаунта;
- 4 – кнопка подтверждения заказа;
- 5 – кнопка отклонения заказа.

При нажатии кнопки подтверждения заказа система изменяет статус заказа на «Активен» и удаляет из списка заказов для рассмотрения. При

нажатии на кнопку «Отклонить заказ» система изменяет статус заказа на «Отклонен» и удаляет из списка заказов для рассмотрения.

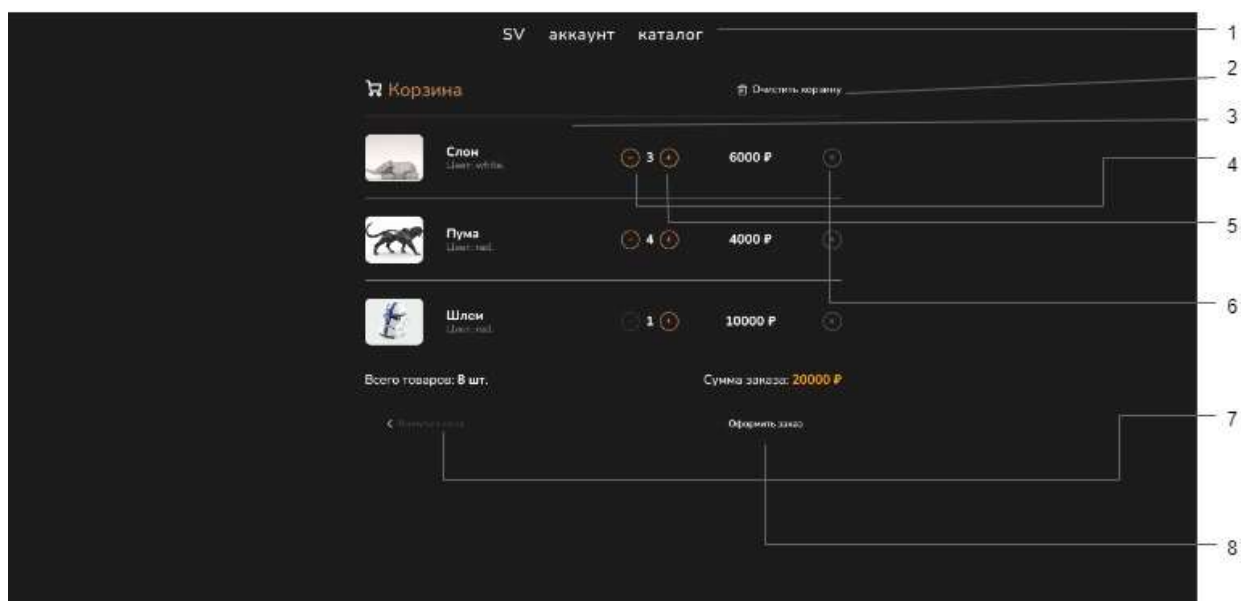


Рисунок 15 – Страница управления заказами

В соответствии с требованиями к заданному интерфейсу были реализованы страницы, код создания которых расположен в приложении Г.

2.4 Проектирование программных средств

На основе требований была спроектирована диаграмма классов одного из основных микросервисов – микросервиса оформления заказов, компоненты которой представлены на таблице 3.

Таблица 3 – Компоненты диаграммы классов

Класс	Назначение
UserEntity	Описание таблицы пользователя БД с помощью библиотеки ORM
CheckEntity	Описание таблицы чека БД с помощью библиотеки ORM
OrderedItem	Описание таблицы заказанного товара БД с помощью библиотеки ORM

Продолжение таблицы 3

Класс	Назначение
ItemsEntity	Описание таблицы товара БД с помощью библиотеки ORM
OrderedItemRepository	Класс для работы с заказанным товаром
ItemsRepository	Класс для работы с товаром
ResourceController	Класс, содержащий конечные точки обращения к сервису
Application	Основная точка входа
SecurityConfig	Класс, содержащий в себе цепочку фильтров запроса
CheckService	Класс–сервис для работы с заказами
CheckRepository	Промежуточный класс для работы с заказами
CheckDTO	Слой–представление данных о чеке
ItemsDTO	Слой–представление данных о предмете каталога
Items	Класс, содержащий информацию о предметах каталога
UserRepository	Класс для работы с предметом каталога

В ходе разработки данного микросервиса будет создано пятнадцать классов, четыре из которых отвечают за описания таблиц, еще четыре – за работу с сущностями, а другие отвечают за обозначения структуры запросов и обработку данных при их выполнении.

3 Реализация веб-приложения электронной коммерции по продаже 3d-изделий

3.1 Реализация базы данных

На основе построенной ранее логической модели базы данных при помощи draw.io была построена ее физическая модель, представленная на рисунке 16.

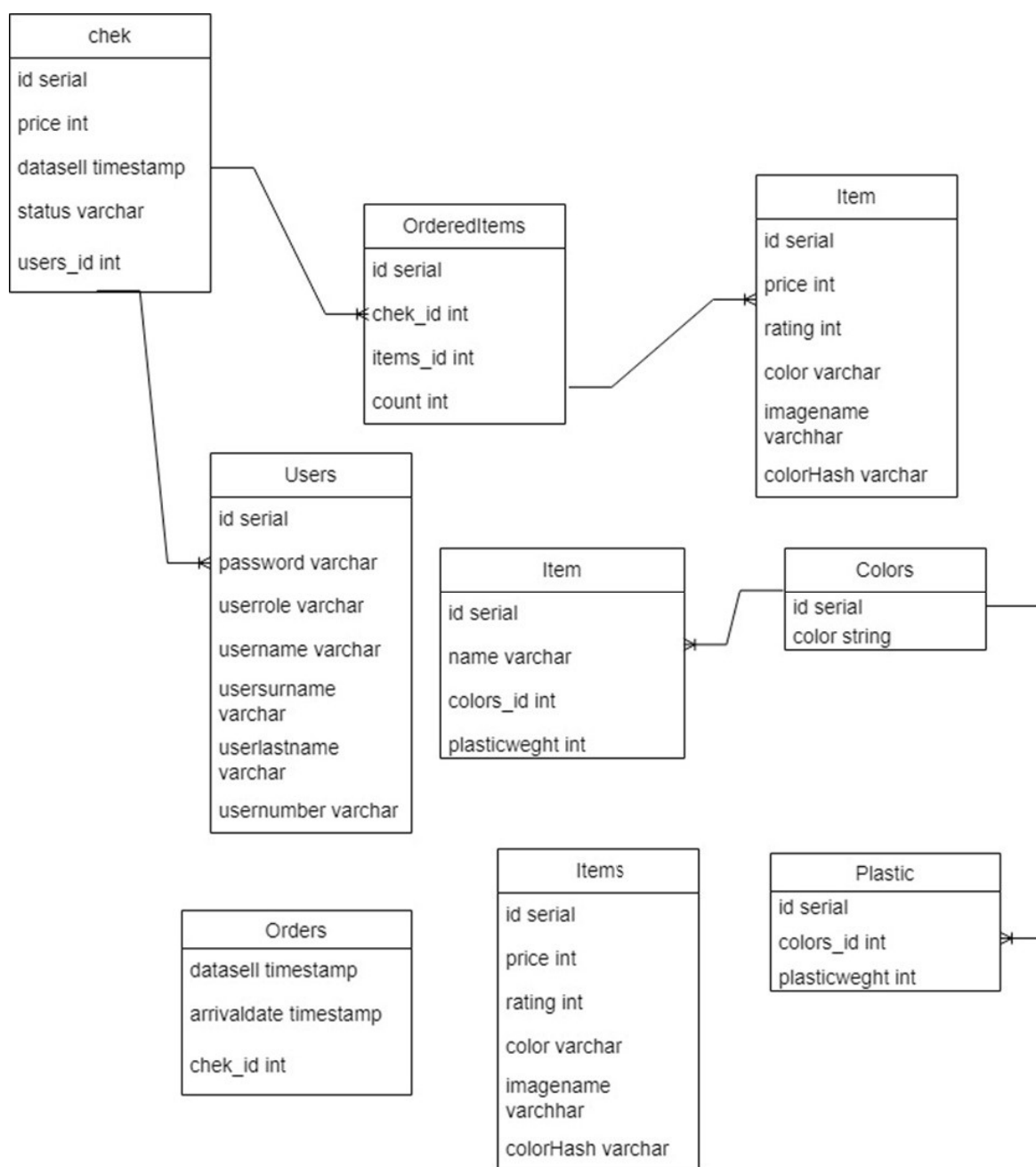


Рисунок 16 – Физическая модель базы данных

В таблице 4 представлена спецификация сущности «Пользователь».

Таблица 4 – Описание атрибутов сущности «Пользователь»

Наименование	Описание	Тип данных	Размер (в символах)	Роль	Обязательный	Ограничения значений	Значение по умолчанию
id	Идентификатор	целый	3	РК	да	1 – 999	
userrole	Роль пользователя	символьный	100		да	Буквы 'A-z' и цифры	
password	Пароль для входа в систему	символьный	100		да	Буквы 'A-z' и цифры	
username	Имя	символьный	100		да	Буквы 'A-я'	
usersecond name	Фамилия	символьный	100		да	Буквы 'A-я'	
usersurname	Отчество	символьный	100		да	Буквы 'A-я'	
usernumber	Номер телефона	символьный	12		да	Цифры и символ «+»	

В таблице 5 представлена спецификация сущности «Заказанные товары».

Таблица 5 – Описание атрибутов сущности «Заказанные товары»

Наименование	Описание	Тип данных	Размер (в символах)	Роль	Обязательный	Ограничения значений	Значение по умолчанию
id	Идентификатор	целый	5	РК	да	1 – 99999	
chek_id	Идентификатор заказа	целый	4	FK	да	1 – 9999	
items_id	Идентификатор предмета заказа	целый	2	FK	да	1 – 99	
count	Счет одинаковых товаров	целый	2		да	1 – -99	

В таблице 6 представлена спецификация сущности «Заказ».

Таблица 6 – Описание атрибутов сущности «Заказ»

Наименование	Описание	Тип данных	Размер (в символах)	Роль	Обязательный	Ограничения значений	Значение по умолчанию
id	Идентификатор	целый	4	РК	да	1 – 9999	
users_id	Идентификатор пользователя	целый	3	РК	да	1 – 999	
price	Цена заказа	целый	7		да	1 – 999999	
datasell	Дата продажи	timestamp	18		да	2022-12-01 00:00:00 – 2030-12-31 00:00:00	
status	Статус	символьный	100		да	Буквы 'A-z' и цифры	

В таблице 7 представлена спецификация сущности «Предмет заказа».

Таблица 7 – Описание атрибутов сущности «Предмет заказа»

Наименование	Описание	Тип данных	Размер (в символах)	Роль	Обязательный	Ограничения значений	Значение по умолчанию
id	Идентификатор	целый	2	РК	да	1 – 99	
title	Название предмета	символьный	100		да	Буквы 'A-я' и пробел	
rait	Рейтинг	целый	2		да	0 – 10	
price	Цена	целый	6		да	1 – 999999	
colors	Цвет товара	символьный	100		да	Буквы 'A-z'	
colors_hash	Хеш кода товара	символьный	7		да	Буквы 'A-F', «#» и цифры	
imagename	Путь к изображению	символьный	100		да	Буквы 'A-z' и др. спецсимволы	

На основе физической модели базы данных, а также спецификаций атрибутов сущностей была построена база данных.

3.2 Реализация серверных компонентов

Серверная часть была разработана в среде IntelliJ Idea с использованием языка Java и фреймворка Spring. Взаимодействие с базой данной производилось посредством Java Hibernate.

Серверная часть включает в себя микросервисы каталога, менеджера и мастера 3d-печати. Они предназначены для эффективного разделения задач.

Также был реализован прокси сервер – дополнительный шлюз, участвующий в интернет-соединении. Он используется как посредник между клиентом и защищенным сервисом, на который он хочет перейти. При подключения через проху-server происходит замена IP-адреса, что позволяет ускорить интернет-соединение [16]. В данном случае прокси сервер также обеспечивает разделение ролей и блокирует несанкционированный доступ. Конечные точки прокси сервера описаны в таблице 8.

Таблица 8 – Спецификация конечных точек веб-приложения

№	Назначение сервиса	HTTP метод	URI сервис	Класс контроллера
1	Получение истории заказов пользователя	GET	/user/orders	RecourseController
2	Оформление корзины	POST	/user/order	RecourseController
3	Получение всех заказов, ожидающих подтверждения	GET	/manager/getallorders	ManagerController
4	Принятие или отклонение заказа	POST	/manager/getallorders	ManagerController

Продолжение таблицы 8

№	Назначение сервиса	HTTP метод	URI сервис	Класс контроллера
5	Получение всех заказов, ожидающих выполнения	GET	/worker/getallorders	WorkerController
6	Завершение заказа	POST	/worker/getallorders	WorkerController

На рисунке 17 показана диаграмма компонентов серверной части приложения.



Рисунок 17 – Диаграмма компонентов

Рассмотрим каждый компонент:

- клиент предназначен для предоставления данных пользователю, а также для запроса данных на сервер;

- прокси сервер содержит элементы, предоставляющие механизм ролей, а также предотвращающие несанкционированный доступ;
- микросервис менеджера отвечает за реализацию функционала менеджера в системе;
- микросервис мастера 3d-печати отвечает за реализацию функционала мастера 3d-печати в системе;
- микросервис оформления заказа предоставляет реализацию оформления товаров корзины, а также получения истории заказов.

На рисунке 18 представлена диаграмма разворачивания приложения, отражающая основное содержимое системы.

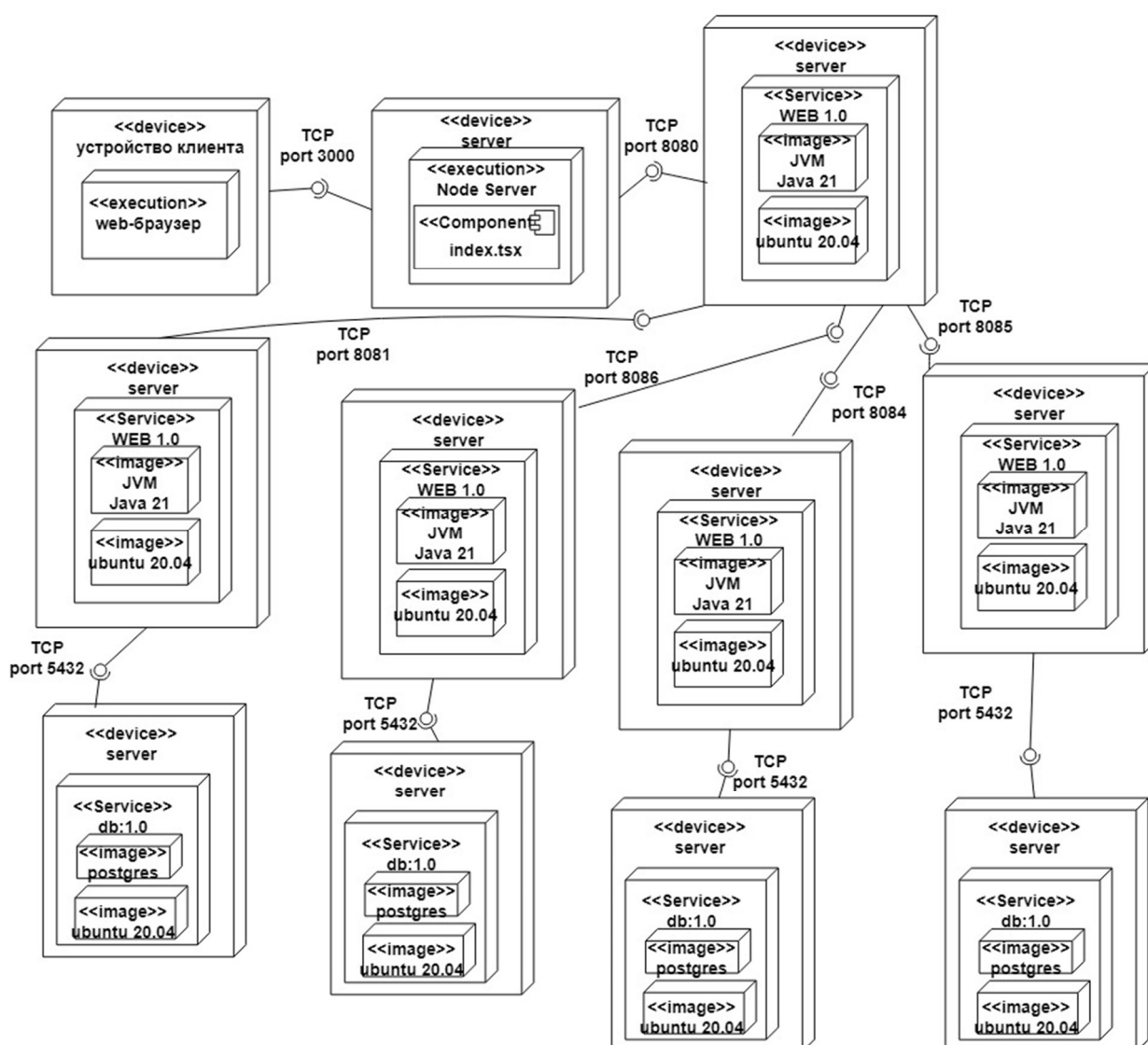


Рисунок 18 – Диаграмма разворачивания

По диаграмме можно заметить, что приложение соответствует трехуровневой архитектуре. Сервера баз данных и сервера приложений могут быть обернуты в docker-контейнеры. Это позволяет в дальнейшем с легкостью разворачивать приложение на другие сервера с платформами, которые поддерживают docker.

Docker позволяет хранить приложение и зависимости в контейнере и запускать его из изолированной среды контейнера. На данной диаграмме показаны 4 микросервиса, обращение по которым идет по 4 разным портам. Они в свою очередь обращаются к своим базам данных по порту 5432. Для взаимодействия с клиентом был открыт порт 8080. Клиентская часть веб-приложения запускается в браузере на 3000 порту.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Существует множество видов тестирования. Самыми распространенными из них являются:

- функциональное тестирование;
- модульное тестирование;
- интеграционное тестирование.

Функциональное тестирование представляет собой комплекс ключевых мероприятий по проверке программного обеспечения, по результатам которых устанавливается соответствие ПО исходным требованиям [17]. Функциональное тестирование делает акцент на бизнес-требованиях к приложению, а проверяется только результат без промежуточных состояний системы в процессе выполнения тестирования.

Юнит-тестирование применяется к изолированным фрагментам кода, поэтому может выявить только ошибки проверяемого модуля. Оно не способно показать баги, возникающие при интеграции модуля с другими компонентами приложения. Также юнит-тестирование не способно выявить системные ошибки продукта в целом [18].

Интеграционное тестирование проверяет совместную работу используемых приложением модулей и сервисов. К данному виду тестирования относится проверка взаимодействия базы данных и приложения. Этот вид тестирования требует запуска различных компонентов приложения. Именно поэтому интеграционное тестирование – самое затратное из всех.

При тестировании веб-приложений необходимо учесть, что разные модели и виды устройств имеют разные разрешения экранов. Необходимо узнать, какие модели и виды устройств на данный момент наиболее популярны среди пользователей, и проводить тестирование в соответствии с этими параметрами.

Кроме того, необходимо провести тестирование на самой старой и самой новой версии браузера. Пользователь не может корректно взаимодействовать

с веб-приложением, если последнее не поддерживает требуемую версию браузера. Разработчик может поставить ограничения и предупреждения исходя из рекомендуемых версий браузера к использованию.

Необходимо учесть все возможные сценарии взаимодействия пользователя с веб-приложением. Для этого пишутся тест-кейсы с описанием действий в подробностях для проверки определенной элемента программы. Тест-кейсы основаны на пользовательских схемах того, как он взаимодействует с приложением, и что может помешать этому.

Также нужно проверить взаимодействие пользователя с интерфейсом приложения, безопасность и удобство использования, производительность приложения и учесть все детали.

Было принято решение реализовать функциональное тестирование программы. Этот выбор обоснован тем, что к программе были предъявлены в основном функциональные требования, выполнение которых нужно проверить.

4.2 Функциональная диаграмма и таблица решений

Произведем функциональное тестирование процесса оформления заказа пользователем. Пользователь должен иметь возможность оформить заказ при условии наличия товаров в корзине, а также при условии того, что пользователь вошел в систему.

Для того, чтобы построить тесты, воспользуемся методом функциональных диаграмм. При построении тестов с использованием данного метода в спецификации определяются причины и следствия. Причина – входное условие, а следствие – выход системы. Далее эти данные преобразуется в управляющий граф.

Входные данные и выводы системы представлены ниже. В таблице 9 представлены причины и следствия при условии выполнения заданных условий.

Таблица 9 – Причины и следствия

П1	Корректный токен доступа
П2	Пользователь вошел в систему
П3	Наличие товаров в корзине
П4	Наличие корректных данных в запросе

В таблице 10 представлены следствия при невыполнении условий.

Таблица 10 – Следствия

С1	Сообщение об ошибке: «Некорректный токен доступа»
С2	Перенаправление на страницу входа в систему
С3	Вывод страницы: «Корзина пуста»
С4	Сообщение об ошибке: «Отсутствие корректных данных в запросе»
С5	Запрос на добавление заказа прошел успешно

Для процесса создания заказа пользователем была построена функциональная диаграмма, представленная на рисунке 19.

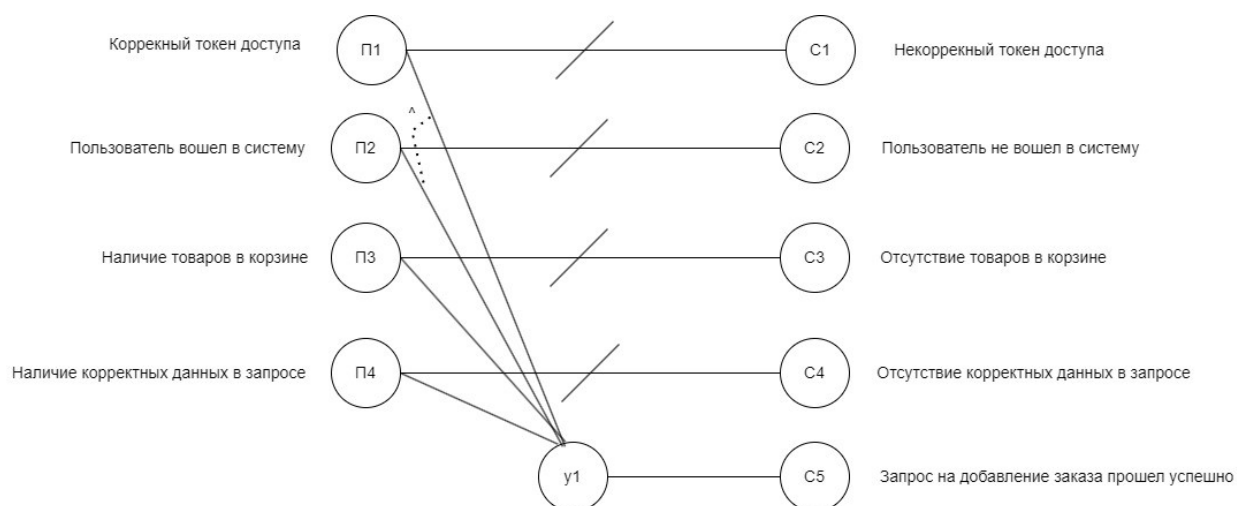


Рисунок 19 – Функциональная диаграмма

На основании функциональной диаграммы была построена таблица основных решений, представленная в таблице 11.

Таблица 11 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12
П1	1	1	1	1	0	0	0	1	1	1	0	0
П2	1	1	1	0	1	0	1	1	0	0	0	0
П3	1	1	0	1	1	1	1	0	0	0	0	0
П4	1	0	1	1	1	1	0	0	1	0	1	0
С1	0	0	0	0	1	0	0	0	0	0	0	0
С2	0	0	0	1	1	1	1	0	1	1	0	0
С3	0	0	1	0	0	0	0	1	0	1	1	1
С4	0	1	0	0	0	0	0	1	0	1	0	0
С5	1	0	0	0	0	0	0	0	0	1	0	0

В таблице 12 представлен набор тестов, составленные на основе таблицы решений.

Таблица 12 – Набор функциональных тестов

№	Входные данные	Ожидаемый результат	Результат
1	Cart: { items: [{ id:3, title:«Пума», price:1000, imageURL: «puma», colorName: «red», colorHash: «#ff0000», count:5}], totalPrice:15000} access_token: eyJraWQiOiI5ZjYyOWQyMy1lNWF... Пользователь вошел в систему	Запрос на добавление заказа прошел успешно	Тест пройден (рисунок В.1)
2	Cart: { items: [{id:3, title: «Пума», price:1000, imageURL: «puma», colorName: «red», colorHash: «#ff0000», count:5}]} access_token: eyJraWQiOiI5ZjYyOWQyMy1lNWF... Пользователь вошел в систему	Сообщение об ошибке: «Отсутствие корректных данных в запросе»	Тест пройден (рисунок В.2)
3	Cart: {} access_token: eyJraWQiOiI5ZjYyOWQyMy1lNWF... Пользователь вошел в систему	Вывод страницы: «Корзина пуста»	Тест пройден (рисунок В.3)

Продолжение таблицы 12

№	Входные данные	Ожидаемый результат	Результат
4	Cart: { items: [id:3, title: «Пума», price:1000, imageUrl: «puma», colorName: «red», colorHash: «#ff0000», count:5}], totalPrice:15000 } access_token: eyJraWQiOiI5ZjYyOWQyMy1lNWF... Пользователь не вошел в систему	Перенаправление на страницу входа в систему	Тест пройден (рисунок В.4)
5	Cart: { items: [{id:3, title: «Пума», price:1000, imageUrl: «puma», colorName: «red», colorHash: «#ff0000», count:5}], totalPrice:15000 } access_token: AAAeyJraWQiOiI5ZjYyOWQyMy1lNWF... Пользователь вошел в систему	Сообщение об ошибке: «Некорректный токен доступа»	Тест пройден (рисунок В.5)
6	Cart: { items: [{id:3, title: «Пума», price:1000, imageUrl: «puma», colorName: «red», colorHash: «#ff0000», count:5}], totalPrice:15000 } access_token: AAAeyJraWQiOiI5ZjYyOWQyMy1lNWF... Пользователь не вошел в систему	Перенаправление на страницу входа в систему	Тест пройден (рисунок В.4)
7	Cart: {} access_token: AAAeyJraWQiOiI5ZjYyOWQyMy1lNWF... Пользователь не вошел в систему	Перенаправление на страницу входа в систему	Тест пройден (рисунок В.4)
8	Cart: {} access_token: AAAeyJraWQiOiI5ZjYyOWQyMy1lNWF... Пользователь не вошел в систему	Перенаправление на страницу входа в систему	Тест пройден (рисунок В.4)

Функциональное тестирование показало, что веб-приложение при добавлении заказа пользователем работает корректно

4.3 Метрики кода

4.3.1 Метрика SLOC

Количество строк кода (англ. Source Lines of Code – SLOC) – это метрика программного обеспечения, используемая для измерения его объёма с помощью подсчёта количества строк в тексте исходного кода [19].

Был проведен подсчет данной метрики в методе Create класса CheckService, который проверяет наличие в базе данных пользователя с указанным номером телефона и паролем. Код функции приведен в листинге 1. Листинг 1 – CheckService.java

```
public boolean create(CheckDTO checkDTO, Long id) {
    List<ItemsDTO> e = checkDTO.getItemsDTOS();
    List<Ordereditem> ordereditems = new ArrayList<>();
    Ordereditem ordereditem;
    Date date = new Date();
    UserEntity userEntity =
userRepository.getReferenceById(id);
    CheckEntity checkEntity = new CheckEntity();
    checkEntity.setDataseil(date);
    checkEntity.setUser(userEntity);
    checkEntity.setStatus("ACTIVE");
    checkEntity.setPrice((long)
checkDTO.getTotalPrice());

    for (ItemsDTO itemsDTO: e) {
        ItemsEntity itemsEntity =
itemsRepository.getItemsEntitiesByColorsAndTitle(itemsD
TO.getColorName(), itemsDTO.getTitle());
        if(itemsEntity!=null){
            ordereditem = new Ordereditem();
            ordereditem.setChek(checkEntity);
```

```

        ordereditem.setItems(itemsEntity);
        ordereditem.setCount(itemsDTO.getCount());
        ordereditems.add(ordereditem);
    }
} checkRepository.save(checkEntity);
orderedItemsRepository.saveAll(ordereditems);
return true;
}

```

Физическими строками считаются все непустые строки текстового файла.

Измеряя логические строки кода, предпринимается попытка посчитать количество собственно операторов в программе, но, разумеется, их определение зависит от конкретного языка программирования.

В данной функции имеем:

- 25 физических строк кода;
- 22 логических строк кода;
- 0 строк комментариев.

Значение метрики SLOC равняется 25 физическим строкам кода и 22 логическим. Это нормальные значения для одной функции.

4.3.2 Цикломатическая сложность

Цикломатическая сложность – показатель сложности исходного кода программы, который связан с вероятностью возникновения ошибок в программе. Показатель цикломатической сложности вычисляется через граф потока управления (Control flow graph, CFG), который отображает количество линейно-независимых путей выполнения в программе.

Большая цикломатическая сложность программы (модуля) как правило означает запутанность кода, и повышение риска ошибок при модификации кода [20].

При вычислении цикломатической сложности используется граф потока управления программы. Узлы графа соответствуют неделимым группам

команд программы, они соединены ориентированными рёбрами, если группа команд, соответствующая второму узлу, может быть выполнена непосредственно после группы команд первого узла.

Граф потока управления программы для функции, представленной в листинге 1, приведен на рисунке 20.

Для вычисления цикломатической сложности $V(G)$ применяется следующая формула:

$$V(G) = e - n + 2p$$

где e – количество дуг;

n – количество вершин;

p – число компонент связности.

Граф содержит девять вершин, тринадцать дуг и один компонент связности.

$$V(G) = 13 - 9 + 2 = 6.$$

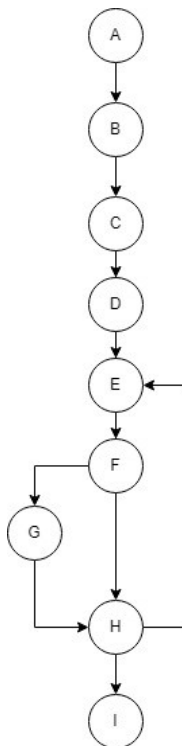


Рисунок 20 – Граф потока управления программы

Таким образом, значение цикломатической сложности равно 6. Это значение не превышает 10, следовательно, функция не является структурно ложной.

4.3.3 Метрика Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

- число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
- число уникальных операндов программы (словарь операндов);
- общее число операторов в программе;
- общее число операндов в программе.

Операторы метода create класса CheckService приведены в таблице 13.

Таблица 13 – Операторы

Операторы	Число вхождений
=	4
.	15
!=	1
New	4
setDataSell	1
setUser	1
setStatus	1
setPrice	1
setChek	1
setItems	1
setCount	1
add	1
;	19
,	2
:	1
{}	6
n1 = 16	N1 = 60

Операнды метода create класса CheckService приведены в таблице 14.

Таблица 14 – Операнды

Операнды	Число вхождений
checkEntity	6
ordereditem	6
e	2
itemsEntity	2
checkDTO	2
date	2
itemsDTO	3
$n2 = 7$	$N2 = 23$

Длина программы рассчитывается по формуле:

$$N = N1 + N2 = 60 + 23 = 83,$$

где N – длина программы;

$N1$ – общее количество вхождений операторов;

$N2$ – общее количество вхождений операндов.

Словарь программы рассчитывается по формуле:

$$n = n1 + n2 = 16 + 7 = 23.$$

где n – словарь программы;

$n1$ – количество различных операторов;

$n2$ – количество различных операндов.

Объем программы рассчитывается по формуле:

$$HV = N \times \log_2(n) = 83 \times 4.5236 = 375,$$

где N – длина программы;

n – словарь программы.

Потенциальный объем программы:

$$V^* = (2 + n^*) \times \log_2(2 + n^*) = 4 \times \log_2(4) = 8,$$

где n – количество уникальных входных и выходных параметров.

Теоретический уровень программы рассчитывается по формуле:

$$L^* = \frac{V^*}{HV} = \frac{8}{375} = 0.021,$$

где V^* – потенциальный объем;

NV – объем программы.

Сложность программы рассчитывается по формуле:

$$HD = \frac{n1}{2} \times \frac{N2}{n2} = \frac{16}{2} \times \frac{23}{7} = 26,$$

где $n1$ – количество различных операторов;

$n2$ – количество различных операндов;

$N2$ – общее количество вхождений операндов.

Значение уровня программы согласно метрике Холстеда равняется 0.021.

4.4.4 Метрика Чепина

Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода.

Все множество переменных, составляющих список ввода-вывода, разбивается на четыре функциональные группы:

- вводимые переменные для расчетов и для обеспечения вывода;
- модифицируемые, или создаваемые внутри программы переменные;
- переменные, участвующие в управлении работой программного модуля (управляющие переменные);
- не используемые в программе («паразитные») переменные;

Каждая переменная может выполнять несколько функций одновременно, поэтому необходимо учитывать ее в каждой соответствующей функциональной группе. Метрика Чепина рассчитывается следующим образом:

$$Q = a1 \times P + a2 \times M + a3 \times C + a4 \times T,$$

где $a1, a2, a3, a4$ – весовые коэффициенты.

Для отражения различного влияния на сложность программы каждой группы использованы весовые коэффициенты. Наибольший вес, равный 3, по мнению Чепина, имеет функциональная группа C, так как она влияет на поток

управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: $a_1=1$, $a_2=2$, $a_4=0.5$. Неиспользуемые переменные имеют ненулевой коэффициент. Несмотря на то, что они не увеличивают сложность программы, иногда они затрудняют ее понимание. В представленной в листинге 1 функции $P = 3$, $M = 10$, $C = 0$, $T = 0$. Учитывая весовые коэффициенты:

$$Q = 3 + 2 \times 10 + 3 \times 0 + 0.5 \times 0 = 23.$$

Значение метрики Чепина равняется 23.

Если в результате изменения рассматриваемых функций значение метрики значительно увеличиться, это будет означать низкое качество изменений.

4.5 Планирование разработки и оценка бюджета

При помощи программы Microsoft Project 2021 было проведены процедуры планирования и оценки бюджета. Проект был начат 8 апреля 2024 года.

Статистика затрат для проекта приведена на рисунке 21.

Статистика проекта для 'Проект1'			
	Начало		Окончание
Текущее	Пн 08.04.24		Пн 03.06.24
Базовое	НД		НД
Фактическое	НД		НД
Отклонение	0 дней		0 дней
	Длительность	Трудозатраты	Затраты
Текущие	40 дней	784 ч	196 000,00 Р
Базовые	0 дней	0 ч	0,00 Р
Фактические	0 дней	0 ч	0,00 Р
Оставшиеся	40 дней	784 ч	196 000,00 Р
Процент завершения			
Длительность: 0%		Трудозатраты: 0%	
			Заккрыть

Рисунок 21 – Статистика проекта

При планировании разработки было выделено 22 задачи. Список задач и их описание представлено на рисунке 22.

Диаграмма Ганта, построенная на основе введенных данных в среде Microsoft Project 2021 приведена на рисунке Б.3 в приложении Б.

Начало реализации проекта	0 дней	Пн 08.04.24	Пн 08.04.24		
➤ Анализ требований	6 дней	Вт 09.04.24	Вт 16.04.24		
Анализ предметной области	2 дней	Вт 09.04.24	Ср 10.04.24	1	Программист1;Программист2
Анализ приложений электронной коммерции	2 дней	Чт 11.04.24	Пт 12.04.24	3	Программист2;Программист1
Постановка задачи	1 день	Пн 15.04.24	Пн 15.04.24	4	Программист1;Программист2
Анализ функциональных требований	1 день	Вт 16.04.24	Вт 16.04.24	5	Программист2;Программист1
Анализ требований завершен	0 дней	Вт 16.04.24	Вт 16.04.24	6	Программист1;Программист2
➤ Проектирование программных средств	9 дней	Ср 17.04.24	Пн 29.04.24		
Проектирование базы данных	3 дней	Ср 17.04.24	Пт 19.04.24	7	Программист1
Проектирование архитектуры приложения	3 дней	Пн 22.04.24	Ср 24.04.24	9	Программист2
Проектирование интерфейса приложения	3 дней	Чт 25.04.24	Пн 29.04.24	10	Программист1
Проектирование программных средств завершено	0 дней	Пн 29.04.24	Пн 29.04.24	11	Программист2;Программист1
➤ Разработка программных средств	23 дней	Вт 30.04.24	Чт 30.05.24		Программист1;Программист2
Разработка базы данных	3 дней	Вт 30.04.24	Чт 02.05.24	12	Программист1
Разработка клиентского приложения	10 дней	Пт 03.05.24	Чт 16.05.24	14	Программист2
Разработка серверной части	10 дней	Пт 17.05.24	Чт 30.05.24	15	Программист1
Разработка программных средств завершена	0 дней	Чт 30.05.24	Чт 30.05.24	16	Программист1;Программист2
➤ Тестирование	2 дней	Пт 31.05.24	Пн 03.06.24		Программист1;Программист2
Тестирование базы данных	1 день	Пт 31.05.24	Пт 31.05.24	17	Программист1;Программист2
Тестирование работы приложения	1 день	Пн 03.06.24	Пн 03.06.24	19	Программист1;Программист2
Тестирование завершено	0 дней	Пн 03.06.24	Пн 03.06.24	20	Программист1;Программист2
Конец проекта	0 дней	Пн 03.06.24	Пн 03.06.24	21	Программист1;Программист2

Рисунок 22 – Список задач проекта

Из полученных данных по проекту наблюдаем, что затраты на его реализацию составляют 196 000 рублей, а общая длительность составляет 40 дней.

Заключение

В ходе выполнения выпускной квалификационной работы разработаны программные средства управления заказами в приложении электронной коммерции по продаже 3d-продукции.

Проведен сравнительный анализ приложений электронной коммерции по продаже 3d-изделий, были обнаружены их преимущества и недостатки. Исходя из него были сформированы требования к разрабатываемому веб-приложению, описаны основные функции. Разработаны диаграмма вариантов использования и диаграмма классов, а также построены концептуальная, логическая и физическая модели базы данных. Успешно проведено функциональное тестирование.

Проведено планирование и расчет бюджета разрабатываемого программного средства при помощи системы управления проектами Microsoft Project 2021.

Разработка программного обеспечения проводилась в среде разработки WebStorm и IntelliJ Idea на языке программирования TypeScript и Java в операционной системе Windows 10 с использованием библиотек React и Spring.

Таким образом, задача, поставленная по разработке средств управления заказами в приложении электронной коммерции по продаже 3d-продукции была выполнена.

Список использованных источников

1. Что такое 3D-печать [Электронный ресурс]. URL: <https://i3d.ru/blog/detail/chto-takoe-3d-pechat-nedostatki-i-preimushchestva-3d-pechat/> (дата обращения: 15.04.2024).
2. Электронная коммерция [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/elektronnaya-kommercziya/> (дата обращения: 15.04.2024).
3. Электронная коммерция: особенности, преимущества и недостатки [Электронный ресурс]. <https://www.uiscom.ru/blog/elektronnaya-kommertsiya-osobennosti-preimushchestva-i-nedostatki/>
4. Volprint [Электронный ресурс]. URL: <https://volprint.shop/> (дата обращения: 15.04.2024).
5. lab3print [Электронный ресурс]. URL: <https://lab3dprint.ru/> (дата обращения: 15.04.2024).
6. Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс]. URL: <https://habr.com/ru/articles/566218/> (дата обращения: 15.04.2024).
7. PostgreSQL [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/postgresql/> (дата обращения: 15.04.2024).
8. Что такое TypeScript и зачем он нужен [Электронный ресурс]. URL: <https://ru.hexlet.io/blog/posts/vse-chto-nuzhno-znat-novichku-o-typescript-ischerpyvayuschiy-gayd> (дата обращения: 15.04.2024).
9. React [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/react/> (дата обращения: 15.04.2024).
10. Оценка возможностей интегрированной среды разработки JetBrains WebStorm [Электронный ресурс]. URL: <https://scienceforum.ru/2018/article/2018003314> (дата обращения: 15.04.2024).
11. Язык программирования Java: что нужно знать новичку [Электронный ресурс]. URL: <https://ru.hexlet.io/blog/posts/yazyk->

programmirovaniya-java-osobennosti-populyarnost-situatsiya-na-rynke-truda (дата обращения: 15.04.2024).

12. Spring [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/spring/> (дата обращения: 15.04.2024).

13. Наши любимые приложения-помощники в тестировании back-end [Электронный ресурс]. URL: <https://habr.com/ru/articles/714700/> (дата обращения: 15.04.2024).

14. Фаулер М. Архитектура корпоративных программных приложений. – Москва.: Вильямс, 2006. – ISBN 5-8459-0579-6.

15. Краткий путеводитель по методологиям и нотациям описания и моделирования бизнес-процессов. Часть 1 [Электронный ресурс]. URL: <https://infostart.ru/pm/1426878/> (дата обращения: 15.04.2024).

16. Что такое прокси-сервер [Электронный ресурс]. URL: <https://help.reg.ru/support/vydelennyye-servery-i-dc/administrirovaniye-vydelennykh-serverov/chto-takoye-proksi-server> (дата обращения: 15.04.2024).

17. Понятие функционального тестирования программного обеспечения [Электронный ресурс]. URL: <https://gb.ru/blog/funktsionalnoe-testirovanie-po/> (дата обращения: 15.04.2024).

18. Тестирование [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/unit-testirovanie> (дата обращения: 15.04.2024).

19. Количество строк кода [Электронный ресурс]. URL: <https://dic.academic.ru/dic.nsf/ruwiki/647191> (дата обращения: 15.04.2024).

20. Что такое цикломатическая сложность? [Электронный ресурс]. URL: <https://testengineer.ru/cyclomatic-complexity/> (дата обращения: 15.04.2024).

Приложение А
(справочное)
Глоссарий

База данных (БД) – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

СУБД – это система управления базами данных. Так называют сложное программное обеспечение, которое требуется, чтобы создавать базы данных, изменять их, получать из них информацию и контролировать версии.

Unified Modeling Language (UML) – графический унифицированный язык моделирования, использующий объектно-ориентированный подход для моделирования в сфере разработки программного обеспечения.

ORM – технология, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

API – Application Programming Interface (интерфейс программирования приложений) – это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

Hibernate Framework – это фреймворк для языка Java, предназначенный для работы с базами данных. Он реализует объектно-реляционную модель – технологию, которая «соединяет» программные сущности и соответствующие записи в базе. Иногда его называют библиотекой – оба названия в принципе справедливы.

Приложение Б
(рекомендуемое)
Графическая часть

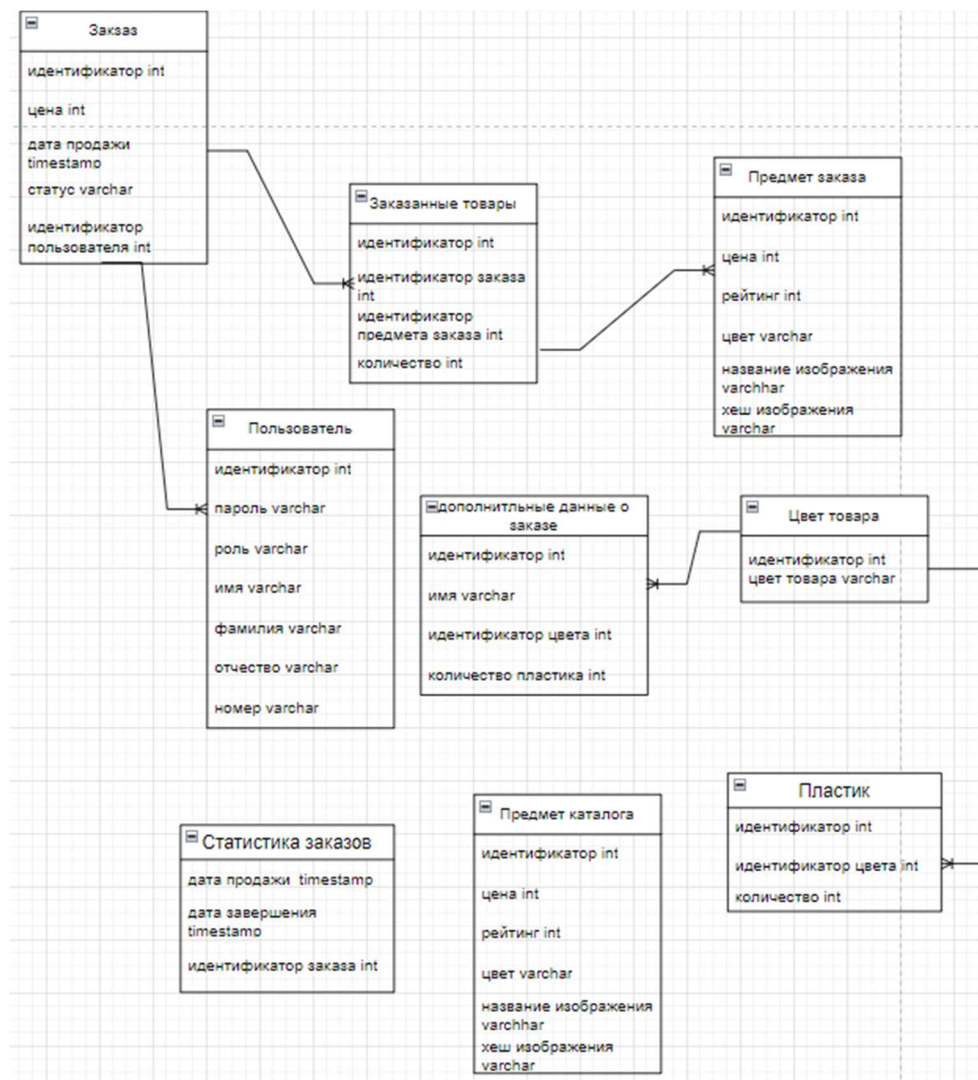


Рисунок Б.1 – Логическая модель базы данных предметной области «Приложение электронной коммерции по продаже 3d изделий»

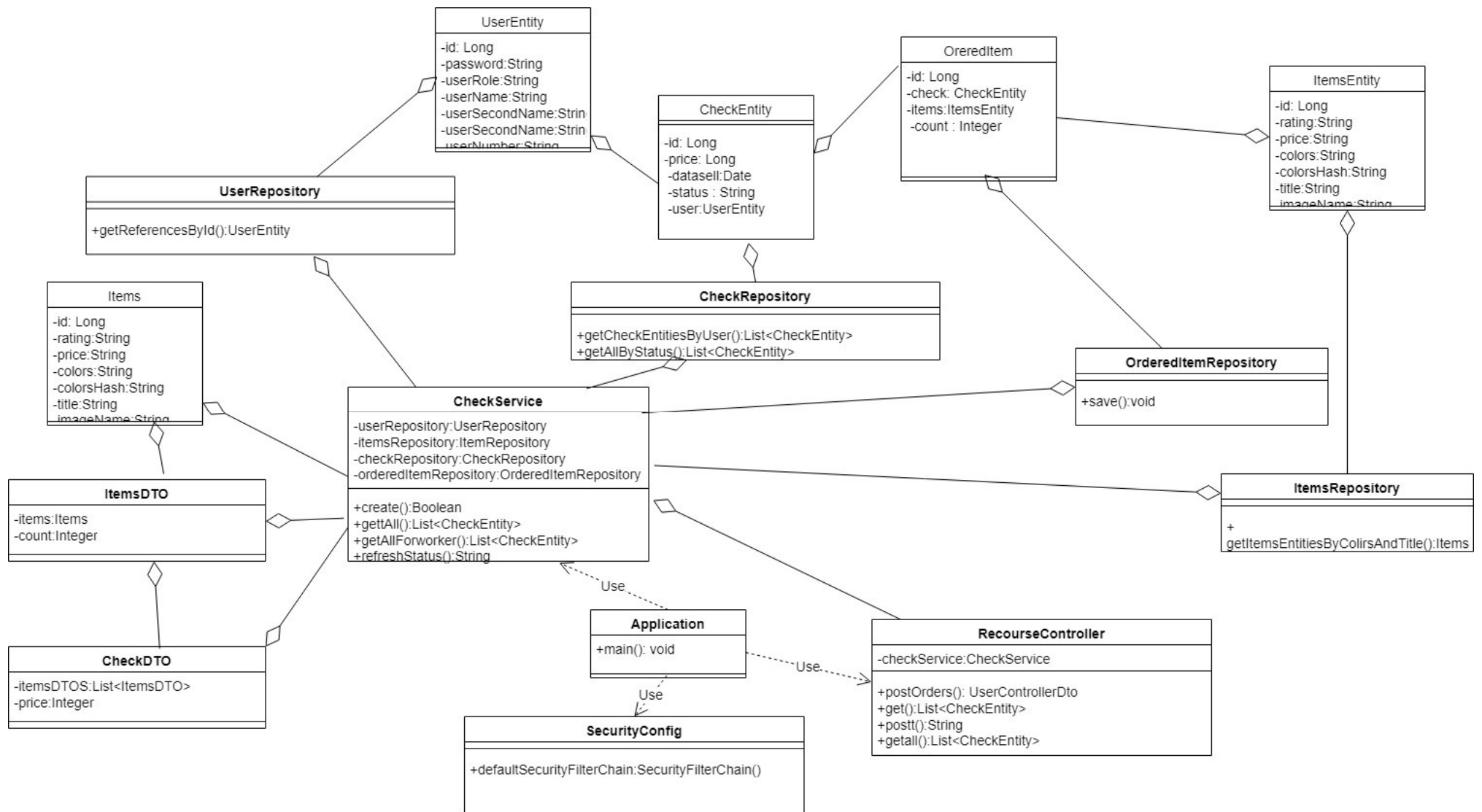


Рисунок Б.2 – Диаграмма классов микросервиса оформления заказов

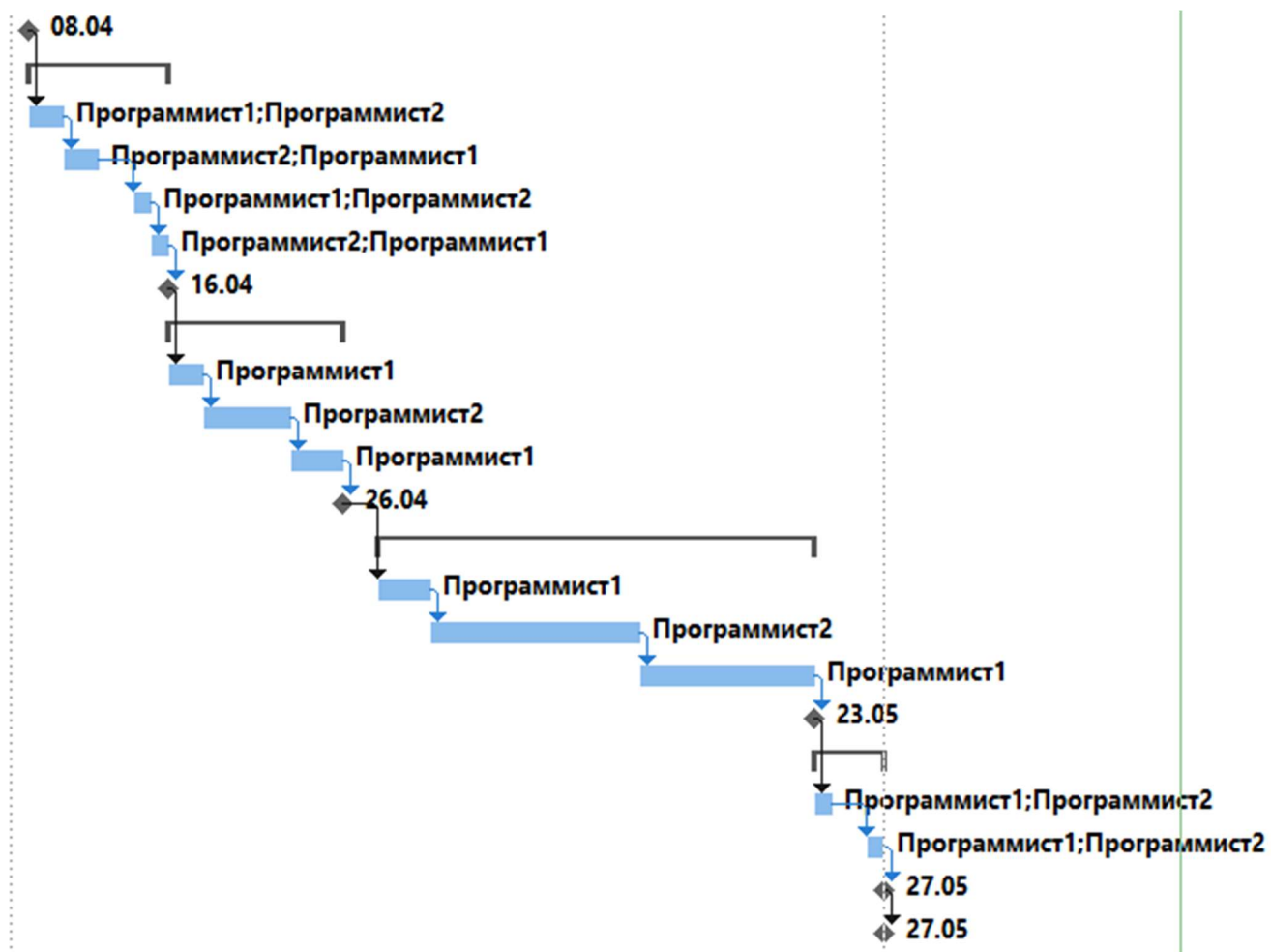


Рисунок Б.3 – Диаграмма Ганта

Приложение В
(рекомендуемое)
Результаты тестирования приложения

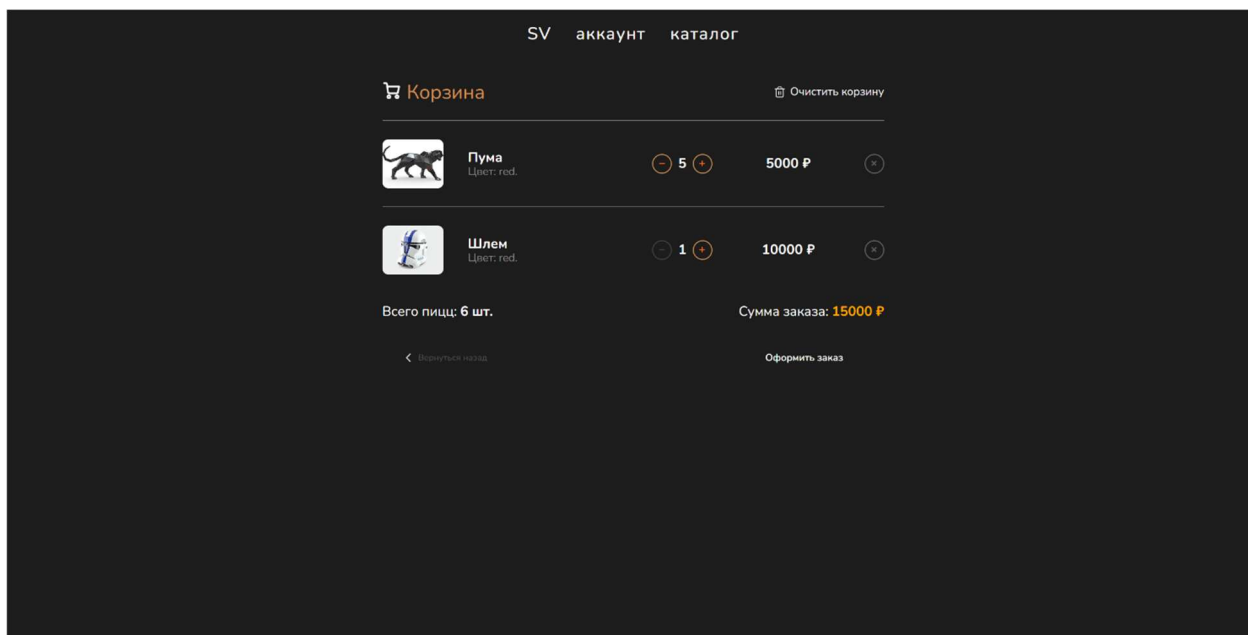


Рисунок В.1 – Успешное добавление

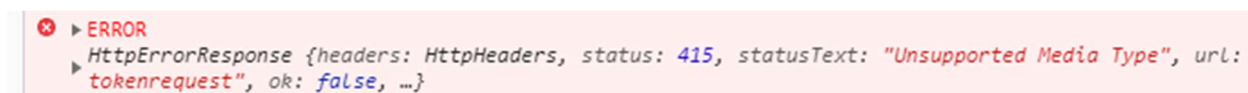


Рисунок В.2 – Ошибка о некорректном запросе

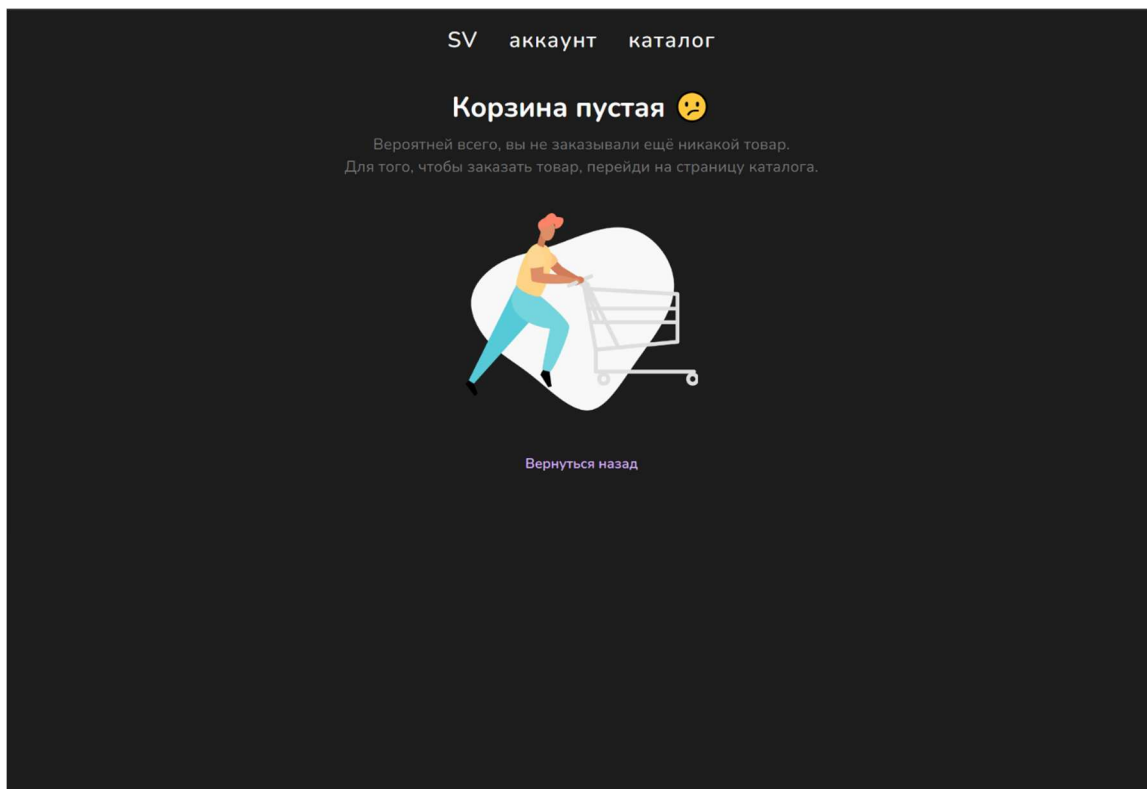


Рисунок В.3 – Страница пустой корзины

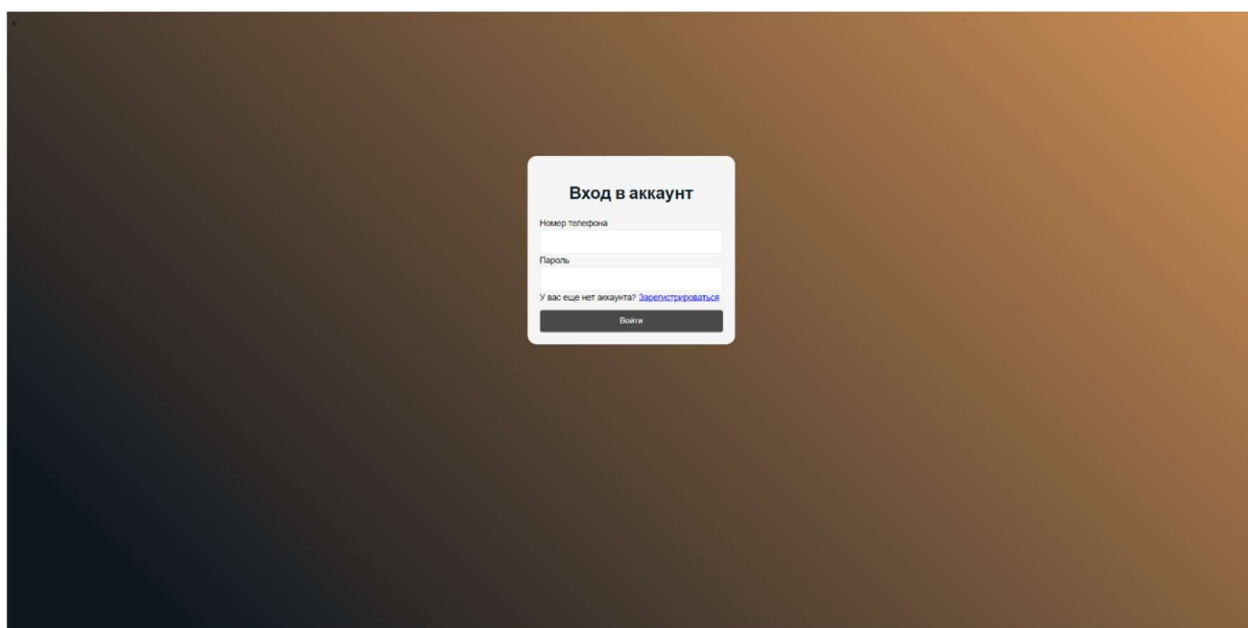


Рисунок В.4 – Перенаправление на страницу входа



Рисунок В.5 – Некорректный токен доступа

Приложение Г
(рекомендуемое)
Листинг веб-приложения

Листинг Г.1 – Файл SecurityConfig.java

```
package com.sv.katalog.config;

@EnableWebSecurity
@Configuration
public class SecurityConfig implements WebMvcConfigurer {

    @Bean
    public SecurityFilterChain
    securityFilterChain(HttpSecurity http) throws Exception {
        return http

        .sessionManagement(httpSecuritySessionManagementConfigurer ->
httpSecuritySessionManagementConfigurer

        .sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .build();
    }

    @Bean
    public FilterRegistrationBean<CorsFilter>
    corsFilterFilterRegistrationBean() {
        UrlBasedCorsConfigurationSource source = new
        UrlBasedCorsConfigurationSource();
        CorsConfiguration config = new CorsConfiguration();

        config.setAllowCredentials(true);

        // Указываем список адресов для которых разрешены кросс-
        доменные запросы

        config.addAllowedOrigin("http://127.0.0.1:3000,http://localhost:
        3000");
        config.addAllowedHeader(CorsConfiguration.ALL);
        config.addExposedHeader(CorsConfiguration.ALL);
        config.addAllowedMethod(CorsConfiguration.ALL);

        source.registerCorsConfiguration("/**", config);
        FilterRegistrationBean<CorsFilter> bean = new
        FilterRegistrationBean<>(new CorsFilter(source));
        bean.setOrder(Ordered.HIGHEST_PRECEDENCE);
        return bean;
    }
}
```

Листинг Г.2 – Файл ItemsController.java

```
package com.sv.katalog.Controller;

@RestController
public class ItemsController {
    final ItemsService itemsService;
    public ItemsController(ItemsService itemsService) {
```

```

        this.itemsService = itemsService;
    }

    @GetMapping("/items")
    // @CrossOrigin(origins = "http://localhost:3000")
    public List<ItemsDTO>
    getItems(@RequestParam("search")Optional<String> search,
    @RequestParam("sortBy")String sort){
        if(search.isPresent()) return
        itemsService.getByNames(search.get());
        return itemsService.getItemsForSort(sort);
    }

}

```

Листинг Г.3 – Файл ItemsDTO.java

```

package com.sv.katalog.dto;

@Data
public class ItemsDTO {
    private Long id;
    private String title;
    private int rating;
    private int price;
    private String colorsName;
    private String colorsHash;
    private String imageURL;
}

```

Листинг Г.4 – Файл ItemsMapper.java

```

package com.sv.katalog.dto;
@Service
public class ItemsMapper {
    public Items toItems(ItemsDTO itemsDTO){
        Items items = new Items();
        items.setId(itemsDTO.getId());
        items.setName(itemsDTO.getTitle());
        items.setColors(itemsDTO.getColorsName());
        items.setPrice(itemsDTO.getPrice());
        items.setColors_hash(itemsDTO.getColorsHash());
        items.setImageName(itemsDTO.getImageURL());
        items.setRating(itemsDTO.getRating());
        return items;
    }

    public ItemsDTO toItemDto(Items items){
        if(items!= null) {
            ItemsDTO itemsDto = new ItemsDTO();
            itemsDto.setId(items.getId());
            itemsDto.setTitle(items.getName());
            itemsDto.setColorsName(items.getColors());
            itemsDto.setPrice(items.getPrice());
        }
    }
}

```

```

        itemsDto.setColorsHash(items.getColors_hash());
        itemsDto.setImageURL(items.getImageName());
        itemsDto.setRating(items.getRating());
        return itemsDto;
    }
    return new ItemsDTO();
}
}

```

Листинг Г.5 – Файл Items.java

```

package com.sv.katalog.entity;
@Table("items")
@Data
public class Items {
    @Id
    Long id;
    String name;
    int rating;
    int price;
    String colors;
    String colors_hash;
    @Column("imagename")
    String imageName;

}

package com.sv.katalog.entity;
@Table("items")
@Data
public class Items {
    @Id
    Long id;
    String name;
    int rating;
    int price;
    String colors;
    String colors_hash;
    @Column("imagename")
    String imageName;

}

```

Листинг Г.6 – Файл ItemsServiceImpl.java

```

package com.sv.katalog.service;
@Service
public class ItemsServiceImpl implements ItemsService{
    private final ItemsMapper itemsMapper;
    private final ItemsRepository itemsRepository;

    public ItemsServiceImpl(ItemsMapper itemsMapper,
ItemsRepository itemsRepository) {
        this.itemsMapper = itemsMapper;
    }
}

```

```

        this.itemsRepository = itemsRepository;
    }

    @Override
    public List<ItemsDTO> getByName(String name) {
        Items items = itemsRepository.findItemsByName(name);
        if(items != null){
            ItemsDTO itemsDTO = itemsMapper.toItemsDto(items);
            List<ItemsDTO> itemsDTOList = new ArrayList<>();
            itemsDTOList.add(itemsDTO);
            return itemsDTOList;
        }
        return new ArrayList<>();
    }

    @Override
    public List<ItemsDTO> getItemsForSort(String sort) {

        if (sort.equals("-price")) {
            return itemsRepository.findAll().stream() //создали
из листа стираим
                .map(itemsMapper::toItemsDto)
                .sorted((a, b) -> b.getPrice() -
a.getPrice())//оператором из streamAPI map
                .collect(Collectors.toList());
        }
        if (sort.equals("price")) {
            return itemsRepository.findAll().stream() //создали
из листа стираим
                .map(itemsMapper::toItemsDto)

                .sorted(Comparator.comparingInt(ItemsDTO::getPrice))//оператором
из streamAPI map
                .collect(Collectors.toList());
        }
        if (sort.equals("rating")) {
            return itemsRepository.findAll().stream() //создали
из листа стираим
                .map(itemsMapper::toItemsDto)

                .sorted(Comparator.comparingInt(ItemsDTO::getRating))//операторо
м из streamAPI map
                .collect(Collectors.toList());
        }
        if (sort.equals("-rating")) {
            return itemsRepository.findAll().stream() //создали
из листа стираим
                .map(itemsMapper::toItemsDto)
                .sorted((a, b) -> b.getRating() -
a.getRating())//оператором из streamAPI map
                .collect(Collectors.toList());
        }
    }

```

```
        return new ArrayList<>();
    }
}
```

Листинг Г.7 – Файл SecConfig.java

```
package com.sv.manager.config;
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecConfig implements WebMvcConfigurer {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity
http) throws Exception{
        http
        .authorizeHttpRequests(authorizationManagerRequestMatcherRegistr
y -> authorizationManagerRequestMatcherRegistry
            .anyRequest().hasRole("MANAGER"))

        .oauth2ResourceServer(httpSecurityOAuth2ResourceServerConfigurer
-> httpSecurityOAuth2ResourceServerConfigurer
            .jwt(Customizer.withDefaults()));

        http.sessionManagement(httpSecuritySessionManagementConfigurer -
> httpSecuritySessionManagementConfigurer

        .sessionCreationPolicy(SessionCreationPolicy.STATELESS));

        return http.build();
    }

    @Bean
    public JwtAuthenticationConverter
jwtAuthenticationConverter() {
        JwtGrantedAuthoritiesConverter
grantedAuthoritiesConverter = new
JwtGrantedAuthoritiesConverter();

        grantedAuthoritiesConverter.setAuthoritiesClaimName("roles");
        grantedAuthoritiesConverter.setAuthorityPrefix("");
        JwtAuthenticationConverter jwtAuthenticationConverter =
new JwtAuthenticationConverter();

        jwtAuthenticationConverter.setJwtGrantedAuthoritiesConverter(gra
ntedAuthoritiesConverter);
        return jwtAuthenticationConverter;
    }

    @Bean
    public FilterRegistrationBean<CorsFilter>
corsFilterFilterRegistrationBean() {
        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
    }
}
```

```

        CorsConfiguration config = new CorsConfiguration();

        config.setAllowCredentials(true);

        // Указываем список адресов для которых разрешены кросс-
        // доменные запросы

        config.addAllowedOrigin("http://127.0.0.1:3000,http://localhost:
        3000");
        config.addAllowedHeader(CorsConfiguration.ALL);
        config.addExposedHeader(CorsConfiguration.ALL);
        config.addAllowedMethod(CorsConfiguration.ALL);

        source.registerCorsConfiguration("/**", config);
        FilterRegistrationBean<CorsFilter> bean = new
        FilterRegistrationBean<>(new CorsFilter(source));
        bean.setOrder(Ordered.HIGHEST_PRECEDENCE);
        return bean;
    }
}

```

Листинг Г.8 – Файл ManagerController.java

```

package com.sv.manager.controller;
@RestController
@RequestMapping("/manager")
public class ManagerController {

    final ManagerService managerService;

    public ManagerController(ManagerService managerService) {
        this.managerService = managerService;
    }

    @GetMapping("/getallorders")
    public List<CheckDTO> managers(@AuthenticationPrincipal Jwt
    jwt) {
        return managerService.getAll(jwt.getTokenValue());
    }

    @PostMapping("/getallorders")
    public ResponseEntity<String> setStatus(@RequestBody ReqBody
    reqBody, @AuthenticationPrincipal Jwt jwt){
        managerService.setNewStatus(reqBody,
        jwt.getTokenValue());
        return new ResponseEntity<>(HttpStatus.CREATED);
    }
}

```

Листинг Г.9 – Файл CheckDTO.java

```

package com.sv.manager.DTO;
@Data
public class CheckDTO implements Serializable {
    Long id;
    Long price;
}

```

```

        Date datasell;
        String status;
        boolean acc;
        UserDTO user;
    }

```

Листинг Г.10 – Файл OrderDTO.java

```

package com.sv.manager.DTO;
@Data
public class OrderDTO implements Serializable {
    Long id;
    Long chek_id;
    Long user_id;
    String usersecondname;
    String username;
    String usersurname;
    String usernumber;
    String status;
    Date datasell;
    Long price;
    String title;
    String colors;
}

```

Листинг Г.11 – Файл UserDTO.java

```

package com.sv.manager.DTO;
@Data
public class UserDTO implements Serializable {
    private Long id;
    private String userName;
    private String userSecondName;
    private String userSurName;
    private String userNumber;
}

```

Листинг Г.12 – Файл ManagerService.java

```

package com.sv.manager.service;
@Service
public class ManagerService {
    private final RestClient restClient;
    private final PlasticRepository plasticRepository;
    private final ItemRepository itemRepository;

    private final ColorRepository colorRepository;
    public ManagerService(RestClient.Builder restClientbuilder,
        PlasticRepository plasticRepository, ItemRepository
        itemRepository, ColorRepository colorRepository) {
        this.restClient =
restClientbuilder.baseUrl("http://localhost:8085").build();
        this.plasticRepository = plasticRepository;
        this.itemRepository = itemRepository;
        this.colorRepository = colorRepository;
    }
}

```

```

@Transactional
public List<CheckDTO> gettAll(String token) {
    List<OrderDTO> orderDTOS =
restClient.method(HttpMethod.GET).uri("manager/allorders").headers(httpHeaders ->
httpHeaders.setBearerAuth(token)).retrieve().body(new
ParameterizedTypeReference<>() {
    });
    List<CheckDTO> checkDTOS = new ArrayList<>();
    assert orderDTOS != null;
    for ( OrderDTO orderDTO: orderDTOS){
        UserDTO userDTO = new UserDTO();
        userDTO.setId(orderDTO.getUser_id());
        userDTO.setUserSecondName (orderDTO.getUsersecondname());
        userDTO.setUsername (orderDTO.getUsername());
        userDTO.setUserSurName (orderDTO.getUsersurname());
        userDTO.setUserNumber (orderDTO.getUsernumber());
        CheckDTO checkDTO = new CheckDTO();
        checkDTO.setId(orderDTO.getChek_id());
        checkDTO.setStatus (orderDTO.getStatus());
        checkDTO.setPrice (orderDTO.getPrice());
        checkDTO.setDatasell (orderDTO.getDatasell());
        checkDTO.setUser (userDTO);
        Color color =
colorRepository.getColorByColor (orderDTO.getColors());
        Item item =
itemRepository.getItemByNameAndColors (orderDTO.getTitle(),
color);
        Plastic plastic =
plasticRepository.getPlasticByColor (color);
        if (plastic.getPlasticweight() -
item.getPlasticweight() >= 0){
            checkDTO.setAcc (true);
        }
        else {
            checkDTO.setAcc (false);
        }
        checkDTOS.add (checkDTO);
    }
    return checkDTOS;
}

```

```

@Transactional
public void setNewStatus(ReqBody reqBody, String token){

restClient.post().uri("manager/allorders").headers(httpHeaders -
>
httpHeaders.setBearerAuth(token)).body(reqBody).retrieve().toBod
ilessEntity();
}

```

```
}
```

Листинг Г.13 – Файл CheckService.java

```
package com.sv.recorce.Service;
@Service
public class CheckService {
    @Autowired
    ItemsRepository itemsRepository;
    @Autowired
    CheckRepository checkRepository;
    @Autowired
    UserRepository userRepository;
    @Autowired
    OrderedItemsRepository orderedItemsRepository;
    @Autowired
    OrdersRepository ordersRepository;

    @Transactional
    public boolean create(CheckDTO checkDTO, Long id){
        List<ItemsDTO> e = checkDTO.getItemsDTOS();
        List<Ordereditem> ordereditems = new ArrayList<>();
        Ordereditem ordereditem;
        Date date = new Date();
        UserEntity userEntity =
userRepository.getReferenceById(id);
        CheckEntity checkEntity = new CheckEntity();
        checkEntity.setDataseil(date);
        checkEntity.setUser(userEntity);
        checkEntity.setStatus("ACTIVE");
        checkEntity.setPrice((long) checkDTO.getTotalPrice());

        for (ItemsDTO itemsDTO: e) {
            ItemsEntity itemsEntity =
itemsRepository.getItemsEntitiesByColorsAndTitle(itemsDTO.getCol
orName(), itemsDTO.getTitle());
            if(itemsEntity!=null){
                ordereditem = new Ordereditem();
                ordereditem.setChek(checkEntity);
                ordereditem.setItems(itemsEntity);
                ordereditem.setCount(itemsDTO.getCount());
                ordereditems.add(ordereditem);
            }
        }
        checkRepository.save(checkEntity);
        orderedItemsRepository.saveAll(ordereditems);
        return true;
    }

    public CheckEntity getByOne(Long id){
        return checkRepository.getCheckEntityById(id);
    }
    public CheckEntity getByOneM(Long id){
        return checkRepository.getCheckEntityById(id);
    }
}
```

```

    }

    public List<CheckEntity> gettall(Long id){
        return
checkRepository.getCheckEntitiesByUser(userRepository.getReferen
ceById(id));
    }
    public List<CheckEntity> gettallforworker(){
        return checkRepository.getAllByStatus("PENDING");
    }
    public List<OrderView> gettallforManager(){
        return ordersRepository.getAll();
    }

    @Transactional
    public String refreshCheckStatus(Long id){
        CheckEntity checkEntity =
checkRepository.getReferenceById(id);
        checkEntity.setStatus("DONE");
        checkRepository.save(checkEntity);
        return "done";
    }

    @Transactional
    public String setNewStatus(Long id, String status){
        CheckEntity checkEntity =
checkRepository.getCheckEntityById(id);
        if (checkEntity == null) {
            return "not found";
        }
        if(status.equals("PENDING")){
            checkEntity.setStatus("PENDING");
            return "true-pending";
        }

        if(status.equals("REJECTED")){
            checkEntity.setStatus("REJECTED");
            return "true-rejected";
        }
        return "false";
    }
}

```

Листинг Г.14 – Файл ResourceController.java

```

package com.sv.recorce;
@RestController
public class ResoruceController {

    @GetMapping("/manager/allorders")
    public List<OrderView> getAllForManager(){

```

```

        return checkService.gettallforManager();
    }
    @PostMapping("/manager/allorders")
    public String postOrdersForManager(@RequestBody RequestBodyy
requestBody){;
        return checkService.setNewStatus(requestBody.getId(),
requestBody.getStatus());
    }
    @GetMapping("/manager/order/{id}")
    public CheckEntity getOneM(@PathVariable Long id){
        return checkService.getByOne(id);
    }
    @GetMapping("/worker/order/{id}")
    public CheckEntity getOne(@PathVariable Long id){
        return checkService.getByOne(id);
    }
    @GetMapping("/worker/allorders")
    public List<CheckEntity> getall(){
        return checkService.gettallforworker();
    }
    @PostMapping("/worker/allorders")
    public String postOrders(@RequestBody RequestBodies
requestBodies){
        return
checkService.refreshCheckStatus(requestBodies.getId());
    }

    @Autowired
    CheckService checkService;

    @GetMapping("/user/orders")
    public List<CheckEntity> gett(@AuthenticationPrincipal Jwt
jwt){
        Long id =(long)jwt.getClaims().get("id");
        return checkService.gettall(id);
    }

    @PostMapping("/user/order")
    public String postt(@RequestBody CheckDTO checkDTO,
@AuthenticationPrincipal Jwt jwt){
        Long id =(long)jwt.getClaims().get("id");
        return checkService.create(checkDTO, id) ? "true"
:"false";
    }
}

```

Листинг Г.15 – Файл SecurityConfig.java

```

package com.sv.recorce;
@Configuration
@EnableWebSecurity

```

```

public class SecurityConfig implements WebMvcConfigurer {
    @Bean
    SecurityFilterChain securityFilterChain(HttpSecurity http)
    throws Exception {
        http
            .authorizeHttpRequests(authorize -> authorize

            .requestMatchers("/worker/**").hasRole("WORKER")

            .requestMatchers("/worker/order/**").hasRole("MANAGER")

            .requestMatchers("/manager/**").hasRole("MANAGER")
                .anyRequest().authenticated()
                .sessionManagement(ss ->
ss.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
                .oauth2ResourceServer((oauth2) ->
oauth2.jwt(jwtConfigurer ->
                jwtConfigurer

            .jwtAuthenticationConverter(jwtAuthenticationConverter())
                ));

        return http.build();
    }

    @Bean
    public JwtAuthenticationConverter
jwtAuthenticationConverter() {
        JwtGrantedAuthoritiesConverter
grantedAuthoritiesConverter = new
JwtGrantedAuthoritiesConverter();

grantedAuthoritiesConverter.setAuthoritiesClaimName("roles");
grantedAuthoritiesConverter.setAuthorityPrefix("");
        JwtAuthenticationConverter jwtAuthenticationConverter =
new JwtAuthenticationConverter();

jwtAuthenticationConverter.setJwtGrantedAuthoritiesConverter(gra
ntedAuthoritiesConverter);
        return jwtAuthenticationConverter;
    }

    @Bean
    public FilterRegistrationBean<CorsFilter>
corsFilterFilterRegistrationBean() {
        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        CorsConfiguration config = new CorsConfiguration();

        config.setAllowCredentials(true);

        // Указываем список адресов для которых разрешены кросс-
доменные запросы

```

```

config.addAllowedOrigin("http://127.0.0.1:3000,http://localhost:3000");
    config.addAllowedHeader(CorsConfiguration.ALL);
    config.addExposedHeader(CorsConfiguration.ALL);
    config.addAllowedMethod(CorsConfiguration.ALL);

    source.registerCorsConfiguration("/**", config);
    FilterRegistrationBean<CorsFilter> bean = new
FilterRegistrationBean<>(new CorsFilter(source));
    bean.setOrder(Ordered.HIGHEST_PRECEDENCE);
    return bean;
}
}

```

Листинг Г.16 – Файл SecurityConfig.java

```

package com.example.worker.config;
@EnableWebSecurity
@Configuration
public class SecurityConfig {

    @Bean
    public SecurityFilterChain webFilterChain(HttpSecurity http)
throws Exception {
        http.cors(Customizer.withDefaults())

.sessionManagement(httpSecuritySessionManagementConfigurer ->

httpSecuritySessionManagementConfigurer.sessionCreationPolicy(Se
ssionCreationPolicy.STATELESS))
        .oauth2ResourceServer((oauth2) ->
oauth2.jwt(jwtConfigurer ->

jwtConfigurer.jwtAuthenticationConverter(jwtAuthenticationConver
ter()))))
        .authorizeHttpRequests(
authorizationManagerRequestMatcherRegistry ->
authorizationManagerRequestMatcherRegistry.anyRequest().hasRole(
"WORKER"));
        return http.build();
    }
    @Bean
    public JwtAuthenticationConverter
jwtAuthenticationConverter() {
        JwtGrantedAuthoritiesConverter
grantedAuthoritiesConverter = new
JwtGrantedAuthoritiesConverter();

grantedAuthoritiesConverter.setAuthoritiesClaimName("roles");
grantedAuthoritiesConverter.setAuthorityPrefix("");
        JwtAuthenticationConverter jwtAuthenticationConverter =
new JwtAuthenticationConverter();

```

```

jwtAuthenticationConverter.setJwtGrantedAuthoritiesConverter(gra
ntedAuthoritiesConverter);
    return jwtAuthenticationConverter;
}
}

```

Листинг Г.17 – Файл WorkerController.java

```

package com.example.worker.controller;
@RestController
@RequestMapping("/worker")
public class WorkerController {
    private final WorkerService workerService;

    public WorkerController(WorkerService workerService) {
        this.workerService = workerService;
    }

    @RequestMapping(value = "/getallorders", method =
RequestMethod.GET)
    public List<CheckDTO> gettOrders(@AuthenticationPrincipal
Jwt jwt){
        return workerService.gettAll(jwt.getTokenValue());
    }

    @RequestMapping(value = "/getallorders", method =
RequestMethod.POST)
    public String setOrders(@RequestBody RequestBodyyy
requestBodyyy, @AuthenticationPrincipal Jwt jwt){
        return
workerService.setOrder(requestBodyyy.getId(), jwt.getTokenValue())
;
    }

}

```

Листинг Г.18 – Файл WorkerService.java

```

package com.example.worker.service;
@Service
public class WorkerService {
    private final RestClient restClient;
    private final OrderRepository orderRepository;
    public WorkerService(RestClient.Builder restClientbuilder,
OrderRepository orderRepository) {
        this.restClient =
restClientbuilder.baseUrl("http://localhost:8085").build();
        this.orderRepository = orderRepository;
    }
    @Transactional
    public String setOrder(Long id, String token){
        CheckDTO checkDTO =
restClient.method(HttpMethod.GET).uri("worker/order/"+id).header
s(httpHeaders ->
httpHeaders.setBearerAuth(token)).retrieve().body(CheckDTO.class
);
        assert checkDTO != null;
    }
}

```

```

        Order order = new Order();
        order.setTimeComp(new Date());
        order.setTimeAcc(checkDTO.getDatasell());
        order.setCheks_id(checkDTO.getId());
        orderRepository.save(order);

restClient.post().uri("worker/allorders").headers(httpHeaders ->
httpHeaders.setBearerAuth(token)).body("id: " +
id).retrieve().toBodilessEntity();
        return "true";
    }

    public List<CheckDTO> gettAll(String token) {
        List<CheckDTO> checkDTOS =
restClient.method(HttpMethod.GET).uri("worker/allorders").header
s(httpHeaders ->
httpHeaders.setBearerAuth(token)).retrieve().body(new
ParameterizedTypeReference<>() {
            });
        return checkDTOS;
    }
}

```

Листинг Г.19 – Файл Home.tsx

```

export interface IResponseToken{
    access_token: string
    expires_in: number
    refresh_token:string
    token_type: string
}

const Home: FC = () => {

    const navigate = useNavigate();
    useEffect(()=>{
        onStartAccount()
    },[])

    const onStartAccount = async () =>{
        await requestTokens()
    }

    const requestTokens = async () => {
        const queryString = window.location.search;
        const urlParams = new URLSearchParams(queryString);
        const codeParam = urlParams.get('code');

        const username = 'gateway';
        const password = 'secret';
        const encoded = Buffer.from(username + ':' +
password).toString('base64');
        if(codeParam) {
            try{

```

```

        let payload = new FormData()
        payload.append('grant_type',
'authorization_code')
        payload.append('code', codeParam)
        payload.append('redirect_uri',
'http://localhost:3000/')
        payload.append('client_id', 'gateway')
        const res = await
axios.post<IResponseToken>('http://localhost:9000/oauth2/token',
payload, {
            headers: { 'content-type':
'application/x-www-form-urlencoded', 'Authorization': 'Basic ' +
encoded},
            withCredentials:true,
        }
    )
    console.log(res.data.refresh_token)
    window.localStorage.setItem("access_token",
res.data.access_token)
    window.localStorage.setItem("refresh_token",
res.data.refresh_token)
    window.location.search = ''
    } catch (e) {
        console.log(e)
    }
}

const packOrder = async () => {

    if (window.confirm('хотите оформить индивидуальный
заказ?')) {
        try {
            const res = await
$api.post<IAccount>('http://localhost:8080/user/order', {
                itemsDTOS: [],
                price: 0,
            }, {
                headers: { 'Content-Type':
'application/json'},
                withCredentials:true,
            }
        )
        alert('Благодарим за покупку! Скоро с вами
свяжется менеджер для обсуждения дальнейших деталей')
    } catch (e) {
        console.log(e)
    }
}

return (
    <>

```

```

<div className="intro">
  <div className="logo">
    <span className="logo__upper">
      <span>S</span><span>uper</span>
      <span>V</span><span>ision</span>
    </span>
    <div className="logo__lower">
      <span>Prod</span><span>uction</span>
    </div>
  </div>

  <div className="title__circle" onClick={() =>
packOrder()}>
    оформить заказ
  </div>
</div>

<div className="description">
  <div className="title">3D - ЭТО
    <div> ИСКУССТВО</div>
  </div>
  <div className="visualization">
    Визуализация невидимого. Воплощение
неосвязаемого.
  </div>
  <div className="our__team">Наша команда не
боится сложных задач. Воплощение самых смелых решений -
    <span> НАША ЖИЗНЬ.</span>
  </div>
  <div className="our__experience">Наш опыт для
    <div> ВАШИХ ПОТРЕБНОСТЕЙ.</div>
  </div>

  <div className="icons__block">
    <div className="icon__block">
      <div className="icon__description">
        <div className="modeling">3D
МОДЕЛИРОВАНИЕ</div>
        <span className="icon__circle">
          <img className="icon_block_img"
src={cubeLogo} alt="3D modeling" />
        </span>
      </div>
      <div className="icon__line"></div>
    </div>
    <div className="icon__block">
      <div className="icon__description">
        <div className="print">3D
ПЕЧАТЬ</div>
        <span className="icon__circle">
          <img className="icon_block_img"
src={printLogo} alt="3D print" />
        </span>
      </div>
    </div>
  </div>
</div>

```

```

        </div>
        <div className="icon__line"></div>
    </div>
</div>

<div className="map">
    <div className="map__div">
        <a className="map__a__first"
href="https://yandex.ru/maps/49/penza/?utm_medium=mapframe&utm_s
ource=maps">
            Пенза
        </a>
        <a className="map__a__second"
href="https://yandex.ru/maps/49/penza/house/ternopol'skaya_ulitsa
_7/YE0YfwVkSE0GQFtpfX5zd3prYw==/?ll=44.924182%2C53.226687&utm_me
dium=mapframe&utm_source=maps&z=17">
            Тернопольская улица, 7 — Яндекс Карты
        </a>
        <iframe className="map__iframe"
src="https://yandex.ru/map-
widget/v1/?ll=44.924182%2C53.226687&mode=search&ol=geo&ouri=ymap
sbml%3A%2F%2Fgeo%3Fdata%3DCgglNzIyNDIxNRJC0KDQvtGB0YHQuNGPLCDQn9
C10L3Qt9CwLCDQotC10YDQvdC-
0L_QvtC70YzRgdC60LDRjyDRg9C70LjRhtCwLCA3IgoNXLIzQhUh6FRC&z=17"
width="100%" height="500px">
        </iframe>
    </div>
</div>

<div className="contacts">
    <div className="contact__us">
        <div
className="contact__us__title">Связаться с нами</div>
        <div className="contact__us__icons">
            <img className="contact__us__icon"
src={emailLogo} alt="Email logo"/>
            <img className="contact__us__icon"
src={vkLogo} alt="VK logo"/>
            <img className="contact__us__icon"
src={telegramLogo} alt="Telegram logo"/>
            <img className="contact__us__icon"
src={pinterestLogo} alt="Pinterest logo"/>
        </div>
    </div>
    <div className="contact__address">
        <div
className="contact__address__text">Контакты для
сотрудничества</div>
        <div className="contact__address__text">г.
Пенза ул. Плеханова д. 19</div>
        <div
className="contact__address__text">+7 (000) 000-00-00</div>
    </div>
</div>

```

```

        </div>
    </div>

    <footer>
        <div className="footer__line"></div>
        <div className="footer__description">
            <div className="footer__title">2023
SUPERVISION PRODUCTION ©</div>
            <div className="footer__signature">
                <div
className="footer__signature__first">S</div>
                <div
className="footer__signature__second">V</div>
            </div>
        </div>
    </footer>
</>
)
}

```

```
export default Home
```

Листинг Г.20 – Файл Catalog.tsx

```

const Catalog: FC = () => {
    const dispatch = useAppDispatch()
    const {searchValue, activeSort} =
useSelector(selectFilterData)
    const {items, status} = useSelector(selectCatalogData)

    const.getItems = async () => {
        const clearSearchValue = (searchValue !== '') ?
`&search=${searchValue}` : ''
        const sortCategory = activeSort.sortCategory
        await
dispatch(fetchItems({sortCategory, clearSearchValue}))
        window.scrollTo(0, 0)
    }

    useEffect(() => {
        const.getItems()
    }, [activeSort, searchValue])

    const catalogItems = items.map(obj => <CatalogItem
key={obj.id} {... obj}/>)

    const onSortChange = useCallback((i: ISortValue) =>
{dispatch(setActiveSort(i))}, [])

    return (
        <div className="container">
            <div className="catalog__top">
                <Search/>
                <CartButton/>
            </div>
        </div>
    )
}

```

```

        <Sort value={activeSort}
onChange={onSortChange}/>
      </div>
      {
        status === 'error' ?
          <div className="catalog__error-info">
            <h2>Произошла ошибка
<span>😞</span></h2>
            <p>
              К сожалению, не удалось получить
элементы каталога<br/>
              Повторите попытку позже
            </p>
          </div> :
          <>
            <div className="catalog__items">
              {status === 'success' ? catalogItems
: <></>}
            </div>
          </>
        }
      </div>
    )
  }
}

export default Catalog

```