МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
Д. В. Топольский
«___»_____ 2022 г.

РАЗРАБОТКА ИНТЕРНЕТ-МАГАЗИНА ДЛЯ ПРОДАЖИ ИЗДЕЛИЙ РУЧНОЙ РАБОТЫ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-09.03.01.2022.267 ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____М. А. Алтухова
«___»_____2022 г.

Автор работы,
студент группы КЭ-406
_____В. И. Щитов
«___»_____2022 г.

Нормоконтролёр,
к.п.н., доцент каф. ЭВМ
_____ М. А. Алтухова
«___»_____2022 г.

Челябинск-2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____Д. В. Топольский
«___»_____2022 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-406
Щитову Вячеславу Игоревичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Разработка интернет-магазина для продажи изделий ручной работы» утверждена приказом ректора от 12.12.2021 г. № 308/141.
2. **Срок сдачи студентом законченной работы:** 05.06.2022 г.
3. **Исходные данные к работе.**

Пользовательские требования:

– веб-приложение должно предоставлять пользователю «Гость» возможность авторизации или регистрации под ролями «Покупатель» или «Продавец»;

– веб-приложение должно предоставлять пользователю «Гость» возможность просмотра всех страниц, кроме профиля и статистики;

– веб-приложение должно предоставлять пользователю «Покупатель» возможность просмотра всех страниц, кроме статистики;

– веб-приложение должно предоставлять пользователю «Продавец» возможность просмотра всех страниц.

Требование к атрибутам качества:

– веб-приложение должно быть адаптировано для мобильных устройств.

Основные функциональные требования:

– у пользователей должна быть возможность зарегистрироваться в системе;

– у пользователей должна быть возможность просмотра всех товаров;

– у пользователей должна быть возможность просмотра страницы отдельного товара;

– пользователь «Покупатель» должен иметь возможность оформить заказ;

– пользователи должны иметь возможность добавление товара в корзину;

– у зарегистрированного пользователя должна быть возможность просмотра своего профиля;

– пользователи с ролью «Покупатель» должны иметь возможность загрузить файлы и текстовое описание своего индивидуального заказа.

Требования безопасности:

– веб-приложение должно обеспечить защиту от фиктивных запросов, созданных как автоматически, так и вручную;

– веб-приложение должно обеспечить проверку корректности ввода пользовательских данных.

4. **Перечень подлежащих разработке вопросов.**

– анализ предметной области (обзор аналогов и выбор технологических решений);

– определение требований к интернет-магазину;

– проектирование (архитектура, определение ролей пользователей и их доступ к различным функциям, структура, макеты веб-приложения, база данных);

– реализация компонент веб-приложения, которые выполняют поставленные требования;

– тестирование разработанного приложения.

5. **Дата выдачи задания:** 1 декабря 2021 г.

Руководитель работы _____/*М. А. Алтухова*/

Студент _____/*В. И. Щитов*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы: 1) анализ предметной области: 10.03.2022 1.1) обзор аналогов: 05.03.2022 1.2) выбор технологических решений: 10.03.2022	10.03.2022	
Разработка модели, проектирование: 1) определение требований к интернет-магазину: 14.03.2022 2) проектирование: 21.03.2022 2.1) архитектура: 15.03.2022 2.2) определение пользователей и их доступ к различным функциям: 16.03.2022 2.3) структура: 17.03.2022 2.4) макеты веб-сайта: 19.03.2022 2.5) база данных: 21.03.2022	21.03.2022	
Реализация системы: 1) реализация страниц веб-сайта выполняющие поставленные требования: 04.04.2022	04.04.2022	
Тестирование, отладка, эксперименты: 1) тестирование разработанного приложения: 25.04.2022	25.04.2022	
Компоновка текста работы и сдача на нормоконтроль	16.05.2022	
Подготовка презентации и доклада	24.05.2022	

Руководитель работы _____/*М. А. Алтухова*/

Студент _____/*В. И. Щитов*/

АННОТАЦИЯ

В. И. Щитов. Разработка интернет-магазина для продажи изделий ручной работы. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2022, 99 с., 19 ил., библиогр. список – 14 наим.

В рамках выпускной квалификационной работы был разработан интернет-магазин для продажи изделий ручной работы. Были проанализированы существующие системы и выявлены их недостатки. Это обусловило необходимость данной разработки. Рассмотрены основные технологии, применяющиеся в разработке веб-приложений, и выбраны наиболее подходящие для данного проекта.

Были проведены проектирование, разработка и тестирование веб-приложения.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

БД – база данных

Маркетплейс – торговая площадка, которая продаёт товары и услуги разных продавцов через интернет, является следующей ступенью развития интернет-магазина)

ПО – программное обеспечение

СУБД – система управления базами данных

API (от англ. Application programming interface) – программный интерфейс приложения

CSS (от англ. Cascading Style Sheets) – каскадные таблицы стилей

CMS (от англ. Content Management System) – система управления контентом веб-ресурса

Footer – нижняя часть сайта

Header – верхняя часть сайта

HTTP (от англ. Hypertext Transfer Protocol) – протокол прикладного уровня передачи данных

JSON (от англ. JavaScript Object Notation) – текстовый формат обмена данными

JWT (от англ. JSON Web Token) – открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON

MVC (от англ. Model-View-Controller) – схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер

REST (от англ. Representational State Transfer) – протокол передачи данных

SPA (от англ. Single Page Application) – одностраничное приложение

SQL (от англ. Structured Query Language) – язык структурированных запросов

ВВЕДЕНИЕ

Актуальность: с постоянным развитием интернета использование информационных технологий в сфере торговли гарантирует появление интернет-магазинов. Становятся доступны любые товары из любой точки планеты. Одежда, бытовая техника, книги, электроника, продукты и другие товары теперь можно покупать удалённо.

Для большинства продавцов товаров и услуг интернет стал основным способом найти новых поставщиков и клиентов. Появление интернет-магазина, позволит повысить конкурентоспособность умельцев, которые изготавливают различные изделия ручной работы, кроме того, интернет-магазин увеличит охват потенциальных покупателей, предоставит дополнительную возможность свободно распространять свои изделия, размещая их на платформе интернет-магазина.

Интернет-магазин – это дополнительный прирост пользователей и повышение конверсии. Данная разработка позволит продавцам расширить базу клиентов, перейти мастерам на новый уровень торговли, а круглосуточная работа сайта позволит уменьшить время работы сотрудников, в свою очередь клиенту легче самому ознакомится с товаром, прочитать описание, посмотреть отзыв других покупателей, уточнить интересующий вопрос у мастера и сэкономить время выбора покупки товара и доставки.

Целью представленной выпускной квалификационной работы является разработка интернет-магазина для продажи изделий ручной работы.

Для достижения поставленной цели, необходимо решить следующие поставленные задачи:

1) выполнить анализ предметной области;

2) разработать требования к интернет-магазину;

3) спроектировать интернет-магазин;

4) реализовать интернет-магазин для изделий ручной работы;

5) провести тестирование интернет-магазина.

В первой главе представлен обзор аналогов и анализ технологических решений для реализации веб-приложения. Во второй главе определены требования к реализуемому интернет-магазину. Третья глава содержит в себе проектирование архитектуры и макетов, определение структуры и описание базы данных. В четвертой главе представлена реализация интернет-магазина. В пятой главе описано проведенное тестирование.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

Среди интернет-магазинов похожей тематики присутствует конкуренция. Малоизвестным сайтам на начальных этапах тяжело конкурировать с интернет-магазинами, которые уже имеют популярность, отзывы. Большая конкуренция приводит к тому, что разрабатываемому интернет-магазину необходимо выделяться среди аналогичных сайтов. Для того чтобы сайт был актуальным, необходимо провести сравнительный анализ схожих интернет-магазинов, выделить их достоинства и недостатки, и основываясь на данных качествах выделить лучшее. Далее рассмотрим интересующие нас интернет-магазины: «Ярмарка Мастеров», «Hands4U» и «Торговый дом мастеров».

Интернет-магазин «Ярмарка Мастеров» [1] – самый крупный маркетплейс изделий ручной работы и украшений. Конкуренция на сервисе высокая, но платформа имеет большую аудиторию. На рисунке 1 изображена главная страница сайта, на нем расположены каталог товаров, поиск, интерактивные страницы (скидки, мастер-классы, журналы, вебинары, ярмарка талантов, истории).

Достоинства:

1) каталог разделен на подкаталоги по категориям;

2) адаптивная верстка интерфейса;

3) заказ товаров возможен всеми пользователями.

Недостатки:

1) главная страница сайта перегружена элементами;

2) при взаимодействии с мастером нет возможности прикреплять дополнительные материалы;

3) на площадке могут быть размещены товары, не являющиеся изделиями ручной работы;

4) навязчивая реклама;

5) не является SPA приложением.

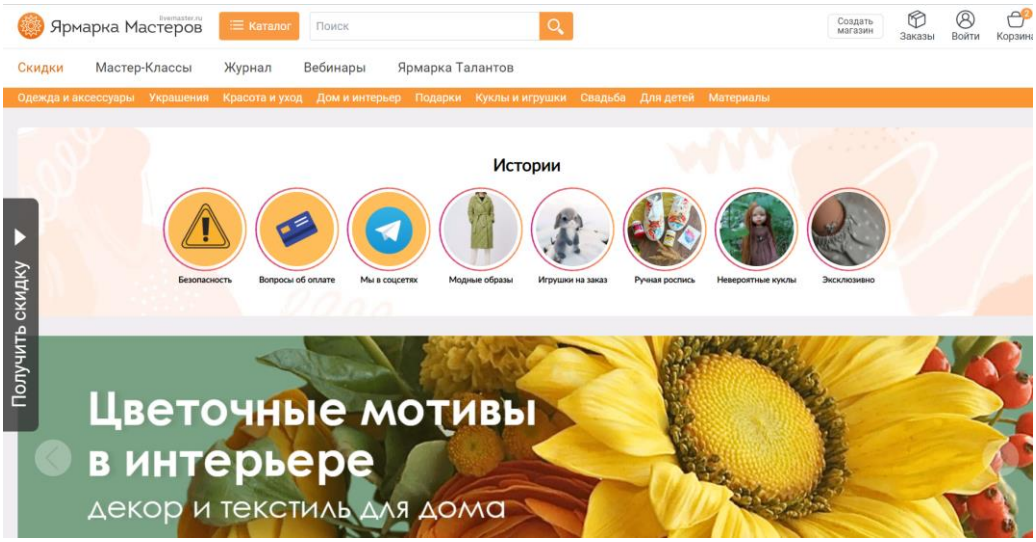

Рисунок 1 – Скриншот главной страницы интернет-магазина «Ярмарка Мастеров»

Интернет-магазин «Hands4U» [2]: на рисунке 2 изображена главная страница сайта, на нем расположены интерактивные блоки с информацией, с популярными категориями и публикациями. Блок категорий вынесен в левую часть.

Достоинства:

1) фильтрация по товарам;

2) адаптивная верстка интерфейса;

3) качественный блок про оплату и доставку.

Недостатки:

1) заказ товаров невозможен всеми пользователями;

2) вкладка меню дублирует информацию подвала;

3) блок с просмотренными товарами не соответствует действительности;

4) не является SPA приложением.

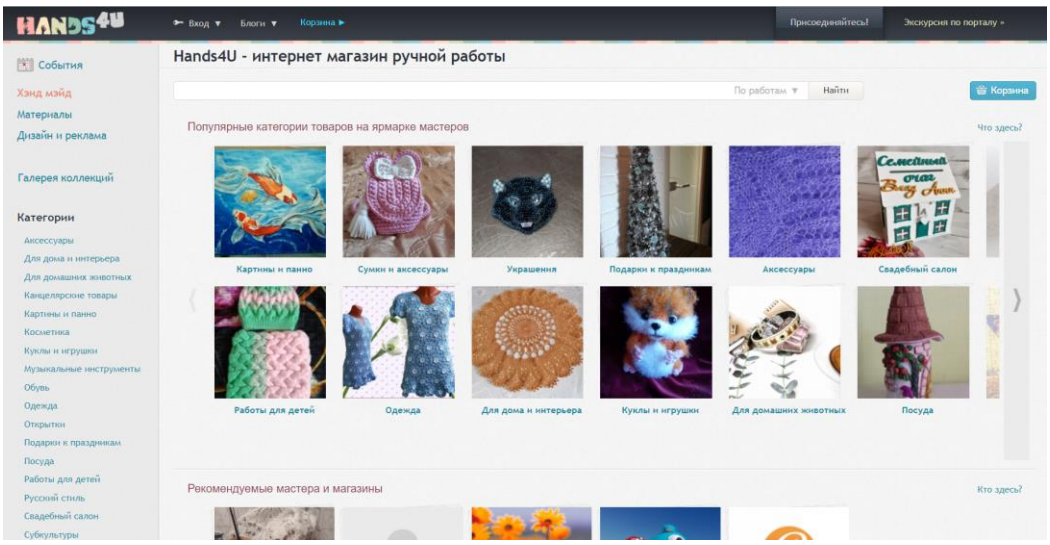

Рисунок 2 – Скриншот главной страницы интернет-магазина «Hands4U»

Интернет-магазин «Торговый дом мастеров» [3]: на рисунке 3 изображена главная страница сайта, на нем расположена навигационная панель, карточки с товарами.

Достоинства:

1) качественный блок про описание товара;

2) каталог разделен на подкаталоги по категориям;

3) простая и удобная сортировка по выбранным параметрам.

Недостатки:

1) отсутствует адаптивная верстка интерфейса;

2) присутствует проблема с CSS стилями (текст на странице накладывается на другой блок);

3) лишняя информация при выборе категорий;

4) заказ товаров невозможен всеми пользователями;

5) не является SPA приложением.

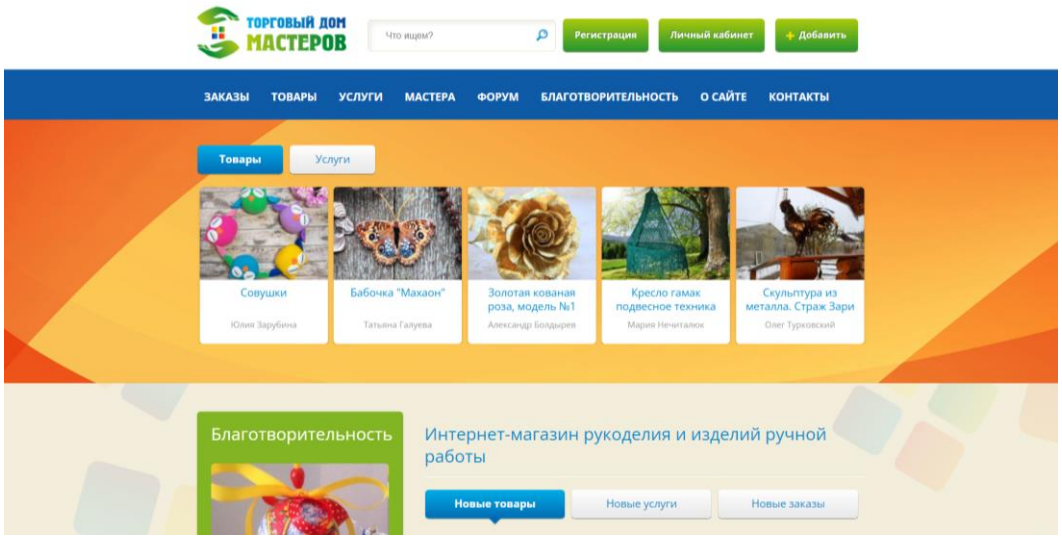

Рисунок 3 – Скриншот главной страницы интернет-магазина «Торговый дом мастеров»

Приняв во внимание все достоинства и недостатки аналогов, для разрабатываемого сайта необходимо реализовать следующие возможности:

1) адаптивная верстка интерфейса;

2) возможность сортировки товаров по параметрам и категориям;

3) оформление заказа всеми пользователями сайта;

4) работа веб-приложения должна осуществляться без перезагрузки страницы, то есть являться SPA приложением.

SPA приложение загружается на одной странице, это реализуется благодаря динамическому обновлению с помощью JavaScript (язык программирования, используются для написания серверной и клиентской части сайтов и мобильных приложений). Во время обновления компонентов сайт не перезагружается, а обновляет только ту часть приложения, где были задействованы компоненты [4].

Следует обратить внимание на то, что один из рассмотренных аналогов реализован с использованием CMS. На рисунке 4 приведен результат анализа веб-сайтов с помощью сервиса для распознавания CMS «be1.ru» [5]. Выяснилось, что интернет-магазин «Торговый дом мастеров» использует CMS систему «OpenCart». Проведя отдельный анализ, интернет-магазин «Ярмарка мастеров» имеет частичное использование CMS системы «OpenCart», которая

используется для реализации дополнительного контента на странице, не задевая основную деятельность.

CMS сайтов

#	Домен	CMS
1	livemaster.ru	-
2	handsforyou.ru	-
3	tdmasterov.ru	OpenCart

Рисунок 4 – Результат анализа сайтов на использование CMS

Разработка веб-приложения будет проходить без использования CMS решений, так как они ограничивают разработчика в реализации функционала непредусмотренной CMS. Приложение будет реализовано по принципам одностраничного сайта (SPA).

1.2 Анализ основных технологических решений

1.2.1 Выбор среды разработки

Для выбора среды разработки необходимо проанализировать языки программирования, подходящие для реализации веб-приложения, такие как JavaScript, Python и PHP.

JavaScript – самый популярный язык разработки для реализации веб-приложения, может быть использован как для разработки Back-end и Front-end части. Имеет среду разработки Node.js, которая предназначена для быстрых и масштабируемых приложений, распространяющаяся бесплатно, обеспечивает управление событийных и асинхронных функций, в результате чего среда разработки становится легкой и эффективной. Как для JavaScript, так и для Node.js существует множество свободно распространяемых библиотек и фреймворков. В таблице 1 приведены достоинства и недостатки JavaScript и Node.js.

Таблица 1 – Достоинства и недостатки JavaScript и Node.js

Достоинства	Недостатки
JavaScript	
Скорость: запускается в браузере клиента	Разные браузеры иногда по-разному интерпретируют код, особенно, если использовать фреймворки, в которых отсутствует поддержка некоторых браузеров
Универсальность: можно реализовать приложение, используя только JavaScript	Не имеет многопроцессорных или многопоточных возможностей
Node.js	
Масштабируемость: легко масштабировать приложение по горизонтали и по вертикали	Несильная поддержка библиотек JavaScript
Одновременная обработка запросов за счет возможности неблокируемых систем ввода-вывода, в результате работает быстрее	API может работать не стабильно из-за частоты изменения API

Python – универсальный язык программирования, имеет встроенные структуры данных и динамическую семантику, можно использовать для реализации веб-приложений. Для создания полнофункциональных веб-приложений чаще всего используют фреймворк Django. В таблице 2 приведены достоинства и недостатки Python и Django.

Таблица 2 – Достоинства и недостатки Python и Django

Достоинства	Недостатки
Python	
Легко читать и поддерживать, за счет удобочитаемости	Низкая эффективность памяти и скорость
Достойная библиотечная система, за счет модулей можно сократить написание кода	Доступность баз данных: присутствуют ограничения доступа к базам данных
Django	
Масштабируемость	Сложность в указании URL регулярными выражениями
Администрирование: встроенное решение, является универсальным и профессиональный	Нельзя использовать для разработки приложений реального времени, где серверы на основе WSGI обрабатывает один запрос

PHP – язык программирования для разработки любых динамических веб-приложений, позволяет создавать полнофункциональные веб-приложения. Используется для разработки серверной части. Сочетается с другими языками программирования, что позволяет расширять функционал. В таблице 3 приведены достоинства и недостатки PHP.

Таблица 3 – Достоинства и недостатки PHP

Достоинства	Недостатки
Скорость: использует собственное пространство памяти, что уменьшает нагрузку на сервер	Снижение производительности при использовании дополнительных функций
Достойная библиотечная система, можно найти необходимые функциональные модули	Некоторые инструменты отладки отсутствуют

В качестве основной среды разработки был выбран Node.js, среда выполнения JavaScript, из-за универсальности и гибкости, документации и фреймворков.

1.2.2 Выбор фреймворков (серверная и клиентская часть)

Для серверной части в Node.js будут рассматриваться следующие фреймворки: Express.js, Koa.js и Meteor.js.

Express.js – среда программирования, работающая на JavaScript, быстрый и минималистичный. Он предназначен для быстрого создания и запуска веб-приложений, обеспечивает высокую производительность, быструю реакцию на запрос пользователя, имеет надежный пользовательский интерфейс, что позволяет реализовать полноценную REST API. REST API – это API, который соответствует архитектурному стилю и ограничениям REST. Системы REST не имеют состояния, масштабируются, кэшируются и имеют единый интерфейс.

Имеет следующие достоинства: открытый исходный код; реализация как статического, так и динамического контента; поддержка большого числа плагинов, поддержка сообществом; удобство работы с HTTP протоколами.

Имеет следующие недостатки: настройка безопасности не предусматривается, следует настраивать отдельно.

Koa.js – работает с промежуточным программным обеспечением. Более модульный и настраиваемый, чем Express, специализируется на веб-сайтах и API. Реализует генераторы ES6, позволяющие избегать обратные вызовы.

Имеет следующие достоинства: качественный обработчик ошибок; отсутствие дополнительного ПО ускоряет работу.

Имеет следующие недостатки: не такая большая поддержка, как у Express; плохая совместимость с дополнительным ПО.

Meteor.js – среда MVC, позволяет создавать веб-приложение в реальном времени. Имеется возможность повторного использования как для серверной части, так и для клиентской. Между клиентом и сервером реализуется передача данных, а не HTML-код.

Имеет следующие достоинства: возможность использования одной команды для подключения нескольких функций; обновление системы, не прерывая сеансов.

Имеет следующие недостатки: плохая поддержка реляционных БД.

Необходим надежный набор функций, с которыми не будет проблем, с возможностью создания API, с поддержкой дополнительных решений, исходя из этого следует выбрать Express.js.

Для клиентской части имеются 2 популярных фреймворка и 1 библиотека, это AngularJS, Vue.js и ReactJS соответственно.

AngularJS – это JavaScript-фреймворк на основе TypeScript, представляет собой фреймворк MVC, создан для работы с большими приложениями, чаще используется для динамических страниц.

Имеет следующие достоинства: использование TypeScript обеспечивает поддержку типов; хорошая документация, зависимость от компонентов, завязанных с модулями.

Имеет следующие недостатки: сложная структура (Injectables, Components, Pipes, Modules и др.), усложняющая разработку.

ReactJS – библиотека JavaScript для создания уникального пользовательского интерфейса веб-приложения, берет акцент на производительность. Эффективно отображает только нужные компоненты при изменении данных.

Имеет следующие достоинства: качественная документация; поддержка модулей, повторное использование кода; поддержка виртуального React DOM; компоненты можно использовать несколько раз; возможность работы с большой нагрузкой.

Имеет следующие недостатки: актуальность документации в связи с частотой обновления и свободной поддержки.

Vue.js – фреймворк JavaScript, подходит для создания легко адаптируемых пользовательских интерфейсов. Предназначен для решения задач уровня представления, хорошо внедряемый.

Имеет следующие достоинства: качественная документация; генератор проектов, где присутствуют различные шаблоны.

Имеет следующие недостатки: при интеграции могут возникать проблемы.

Необходим надежный стабильный набор функций, гибкий, с возможностью реализации SPA приложения, исходя из этого следует выбрать React.

1.2.3 Выбор СУБД

Для полного функционирования веб-приложения необходимо хранить информацию о пользователях и товарах, следовательно, требуется БД. Данные будут точными, надежными и простыми в использовании, если будет использована БД.

БД управляется при помощи СУБД – это программное обеспечение, которое позволяет создавать, изменять и управлять, определять, хранить, манипулировать и извлекать данные из БД.

Далее рассмотрим два распространённых типа БД: реляционная и нереляционная.

Реляционная база данных – отношения между данными являются реляционными, и данные хранятся в табличной форме столбцов и строк. Каждый столбец таблицы представляет атрибут, в свою очередь строка

таблицы представляет собой запись. В таблице 4 приведены достоинства и недостатки реляционной БД.

Таблица 4 – Достоинства и недостатки реляционной базы данных

Достоинства	Недостатки
Простота в использовании: просмотр информации в виде таблиц, проще для понимания	Техническое обслуживание: со временем из-за увеличения объема данных обслуживание усложняется
ACID: Атомарность: транзакции будут выполнены полностью или не будет выполнены совсем Согласованность: данные будут действительны в соответствии со всеми определенными правилами Изолированность: параллельно выполняющиеся транзакции не влияют на результат Надежность: если выполнение транзакции подтверждено, то транзакция выполнена	Отсутствие масштабируемости: проблемы в управлении на нескольких серверах, с увеличением набора данных и распределением, структура нарушается, а использование нескольких серверов влияет на производительность
Гибкость: легко расширять, обновлять и удалять данные	Сложность в структуре: хранение данных осуществляется в табличной форме, что затрудняет представление сложных отношений между объектами

Следует заметить, что для связи с данными, хранящимися в системе управления реляционными базами данных используется язык SQL.

Нереляционная БД (NoSQL от «Not Only SQL») является альтернативой, которая не используют табличную схему данных и SQL в качестве основного языка доступа к данным. Данный тип БД полезен для больших распределенных наборов данных. БД NoSQL поддерживают различные модели данных, включая форматы, основанные на «ключ-значение», «документ», «широкий столбец» и «график», еще встречается «объект». В таблице 5 приведены достоинства и недостатки нереляционной БД.

Таблица 5 – Достоинства и недостатки нереляционной БД

Достоинства	Недостатки
Горизонтальная масштабируемость: с увеличением набора данных и распределением, структура не нарушается, а использование нескольких серверов не влияет на производительность	Большинство из БД не поддерживают транзакции ACID с несколькими записями Не имеют функций надежности, которые есть у реляционных баз данных

Окончание таблицы 5

Гибкая модель данных: гибкая схема позволяет легко вносить изменения в базу данных по мере изменения проекта	Отсутствие универсального языка запросов
Репликация: поддерживается автоматическая репликация БД для обеспечения доступности в случае сбоев	

Разработка интернет-магазина будет базироваться на реляционной БД, так как необходима связь данных у пользователя и товаров, поддержка ACID (atomicity – атомарность, consistency – согласованность, isolation – изолированность и durability – надежность) важна, горизонтальная масштабируемость не планируется.

Рассмотрим основные технологические решения для СУБД. Самыми популярными СУБД являются MySQL, Microsoft SQL Server, PostgreSQL, Oracle Database. Из них не будут рассмотрены Microsoft SQL Server и Oracle Database, так как имеют коммерческую лицензию и закрытый исходный код, что усложнит дальнейшую разработку.

MySQL – реляционная СУБД с открытым исходным кодом, разрабатывается и поддерживается Oracle, распространяется как под GNU (General Public License), так и под коммерческой лицензией.

Используется для различных целей, имеет простой пользовательский интерфейс, дополнительный пакет команд, позволяющий обрабатывать большой объем данных. Поддерживает системы хранения данных MyISAM и InnoDB. В таблице 6 приведены достоинства и недостатки СУБД MySQL.

Таблица 6 – Достоинства и недостатки MySQL

Достоинства	**Недостатки**
Производительность: обеспечивает многопоточность для достижения более оптимизированной производительности	Присутствуют проблемы со стабильностью
Безопасность данных: данные защищены паролем, который хранится в зашифрованном виде, за счет сложных алгоритмов шифрования	Не предназначен для данных большого размера: снижение производительности при увеличении размера данных, заметно снижение производительности
Имеет клиент-серверную архитектуру: любое количество пользователей могут взаимодействовать с БД	Не соответствует стандарту SQL: имеются функциональные ограничения

PostgreSQL – это система управления реляционными базами данных клиент-серверного типа с многопроцессорной архитектурой, с открытым исходным кодом, ориентированная на расширяемость и соответствие стандартам. Поддерживает запросы SQL (реляционные) и JSON (нереляционные). Имеет множество расширенных функций, которые предлагают другие системы управления базами данных корпоративного класса. Не просто реляционная, а объектно-реляционная СУБД, поддерживает и объектный, и реляционный подход, поддерживающая хранение пользовательских объектов. В таблице 7 приведены достоинства и недостатки СУБД PostgreSQL.

Таблица 7 – Достоинства и недостатки PostgreSQL

Достоинства	**Недостатки**
Структуры и типы данных: большое число поддерживаемых типов данных	Требует дополнительных изменений для повышения скорости
Расширяемость: возможность расширения функционала и настройки	Менее популярен чем MySQL
Соответствие SQL: стремится строго придерживаться стандартов SQL	Может возникнуть трудности при установке и настройки

В качестве основной СУБД для реализации интернет-магазина был выбран PostgreSQL, так как необходима СУБД с открытым исходным кодом, распространяемая свободно, с высокой расширяемостью и надежностью.

1.3 Вывод по анализу предметной области

На основе анализа предметной области были определены подлежащие к реализации основные возможности: адаптивность, сортировка товаров, оформление заказа всеми пользователями. Из анализа был сделан вывод, что необходимо реализовать SPA приложение без использования CMS средств. Далее был определен основной набор технических решений, которые будут использованы в работе: Node.js – среда выполнения JavaScript [6]; Express.js в качестве фреймворка для серверной части [7]; ReactJS в качестве библиотеки клиентской части [8]; PostgreSQL в качестве СУБД [9].

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К ИНТЕРНЕТ-МАГАЗИНУ

Необходимо разработать интернет-магазин, в котором будет реализована логика взаимодействия пользователей с системой. С помощью данного веб-приложения пользователи получат возможность выбирать товары по категориям, получать всю основную информацию о товарах, иметь связь с продавцом напрямую и заказывать индивидуальные работы.

2.1 Функциональные требования

Далее будут определены функциональные требования к интернет-магазину:

1) пользователи должны иметь возможность авторизироваться и зарегистрироваться;

2) пользователь «Покупатель» должен иметь возможность просматривать каталог товаров;

3) пользователь «Покупатель» должен иметь возможность просмотреть отдельный товар и его подробное описание;

4) пользователь «Покупатель» должен иметь возможность поставить оценку товару или оставить отзыв о товаре;

5) пользователь «Покупатель» должен иметь возможность загрузить файлы и текстовое описание своего индивидуального заказа для продавца;

6) пользователь «Покупатель» должен иметь возможность добавить товар в корзину;

7) пользователь «Покупатель» должен иметь возможность просмотра корзины товаров;

8) пользователь «Покупатель» должен иметь возможность удалить товар из корзины;

9) пользователь «Покупатель» должен иметь возможность оформить заказ;

10) пользователь «Покупатель» должен иметь возможность просмотра

профиля;

11) пользователь «Продавец» должен иметь возможность просмотра всех своих товаров.

12) пользователь «Продавец» должен иметь возможность добавления, изменения, редактирования своих товаров;

13) пользователь «Продавец» должен иметь возможность просмотра статистики заказов.

2.2 Нефункциональные требования

Далее будут определены нефункциональные требования к интернет-магазину:

1) веб-приложение должно быть кроссбраузерным, т.е. приложение должно одинаково отображаться в различных браузерах;

2) интерфейс веб-приложения должен быть минималистичным и интуитивно понятным;

3) элементы навигации по приложению должны быть быстродоступными.

2.3 Требования безопасности

Далее будут определены требования безопасности к интернет-магазину:

1) веб-приложение должно обеспечить защиту от фиктивных запросов, созданных как автоматически, так и вручную;

2) веб-приложение должно обеспечить проверку корректности ввода пользовательских данных;

3) веб-приложение должно обеспечить шифрование паролей пользователя.

3 ПРОЕКТИРОВАНИЕ

3.1 Архитектура предлагаемого решения

Интернет-магазин будет строится на клиент-серверной архитектуре, в котором клиентом выступает браузер, а сервером – веб-сервер.

Основная часть веб-приложения находится на стороне веб-сервера, который обрабатывает полученные от пользователя действия (запросы) и формирует ответ, который получит пользователь. Затем свою работу начинает выполнять браузер, его задача преобразовывать полученный ответ со стороны сервера отображая его в виде графический интерфейс, понятный пользователю [10].

Проектируемый интернет-магазин будет использовать двухуровневую архитектуру. Двухзвенная архитектура состоит из трех компонентов: пользовательский системный интерфейс, управление сервером и управление БД. Пользовательский системный интерфейс – это компонент системы поддержки принятия решений, обеспечивает связь для выполнения запросов к серверу, отвечает за отображение, сеансы, ввод текста и диалог с сервером. Управление сервером включает в себя основные процессы, логика взаимодействий процессов и мониторинг. Управление базой данных включает службы базы данных и файлов. Диаграмма компонентов системы представлена на рисунке 5.

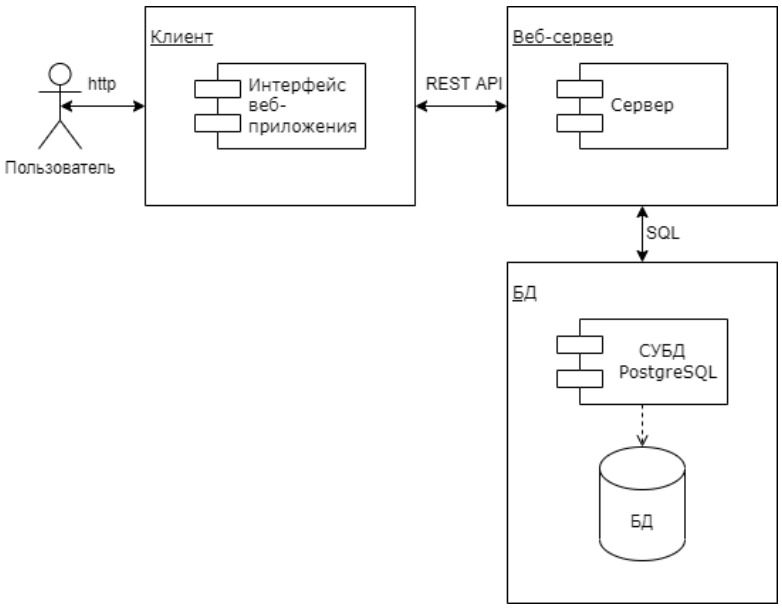

Рисунок 5 – Диаграмма компонентов системы

Клиент – это браузер пользователя. Чтобы пользователь увидел графический интерфейс интернет-магазина в окне браузера, последний обрабатывает полученный ответ веб-сервера, в котором будет содержаться необходимая информация, получаемая в результате применения JS, CSS и HTML.

Веб-сервер – это сервер, который принимает запросы от клиентов и выдает им ответы. Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает.

БД фактически не является частью веб-сервера, но большинство приложений просто не могут выполнять все возложенные на них функции без нее, так как именно в базе данных хранится вся динамическая информация веб-сайта.

3.2 Варианты использования

В ходе анализа проектируемой системы с помощью языка графического моделирования UML [11] выявлены основные варианты использования, которые представлены на рисунке 6.

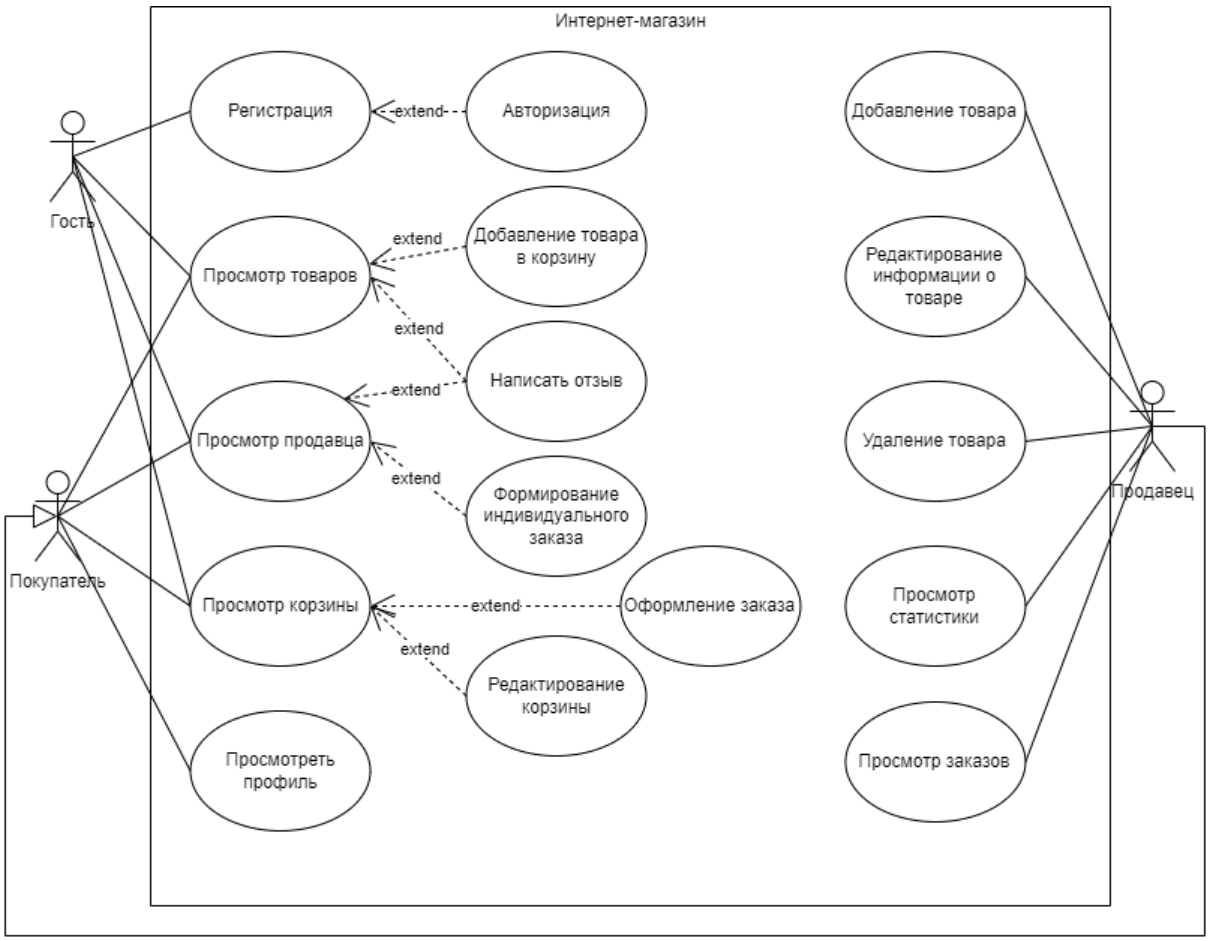

Рисунок 6 – Диаграмма вариантов использования

С данной системой взаимодействует три актера:

1) «Гость» – это неавторизированный пользователь сайта, который может выбрать товары из каталога, добавить их в корзину и сделать заказ с указанием дополнительной информации и пройти процедуру регистрации/авторизации;

2) «Покупатель» – это авторизированный пользователь сайта, который может выбрать товары из каталога, добавить их в корзину и сделать заказ, просмотреть профиль;

3) «Продавец» – пользователь, прошедший авторизацию, имеет право на добавление товара, категории товаров, просмотр статистики. Продавец наследует все возможности покупателя сайта.

3.3 Структура интернет-магазина

Структура веб-приложения относится к тому, как построен сайт, как все отдельные подстраницы в нем связаны друг с другом и какой иерархии придерживаются. Хорошая структура сделает поиск информации интуитивно понятным и простым в использовании, навигация по странице будет понятней.

На рисунке 7 представлена структура проектируемого сайта. Зайдя на сайт пользователя встречает главная страница, где пользователь может ознакомится с основной информацией и перейти на интересующие его разделы, такие как страница продавца, в которой он может ознакомится об определенном продавце и о его товарах, кроме того он может сделать индивидуальный заказ, кроме того пользователь может зайти на страницу товара, прочитать основную информацию и прочитать или оставить отзыв о товаре, пользователь может просмотреть корзину и в дальнейшем оформить заказ, пользователь может посетить раздел профиль с его персональной карточкой покупателя, всем неавторизированным пользователям доступна авторизация.

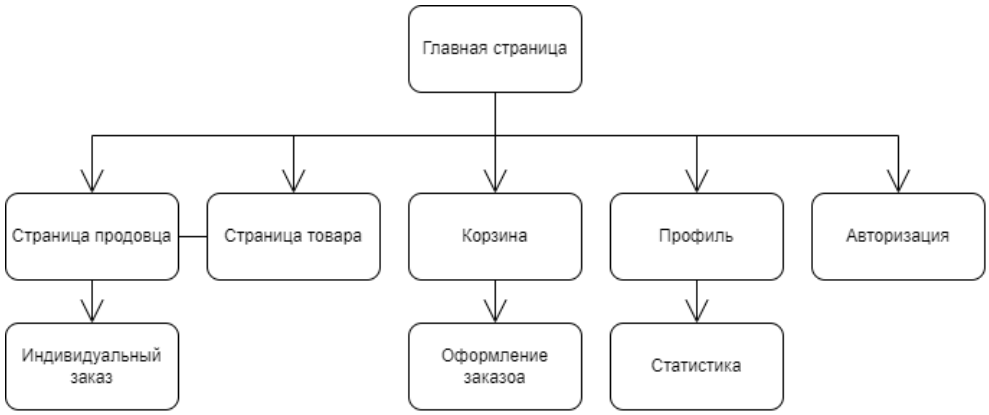

Рисунок 7 – Структура интернет-магазина

Следует сделать важное замечание, не всем пользователям доступны все разделы сайта.

Пользователю «Гость» не доступны следующие разделы «Профиль» и «Статистика».

Пользователю «Покупатель» не доступен следующий раздел «Статистика».

3.4 Проектирование макетов сайта

Спроектируем макет веб-приложения, он будет определять структуру интернет-магазина, поможет структурировать информацию, улучшить навигацию по страницам, выделить важную информацию. Кроме того, макет будет определять, что пользователь будет видеть в первую очередь. Макет главной страницы изображен на рисунке 8.

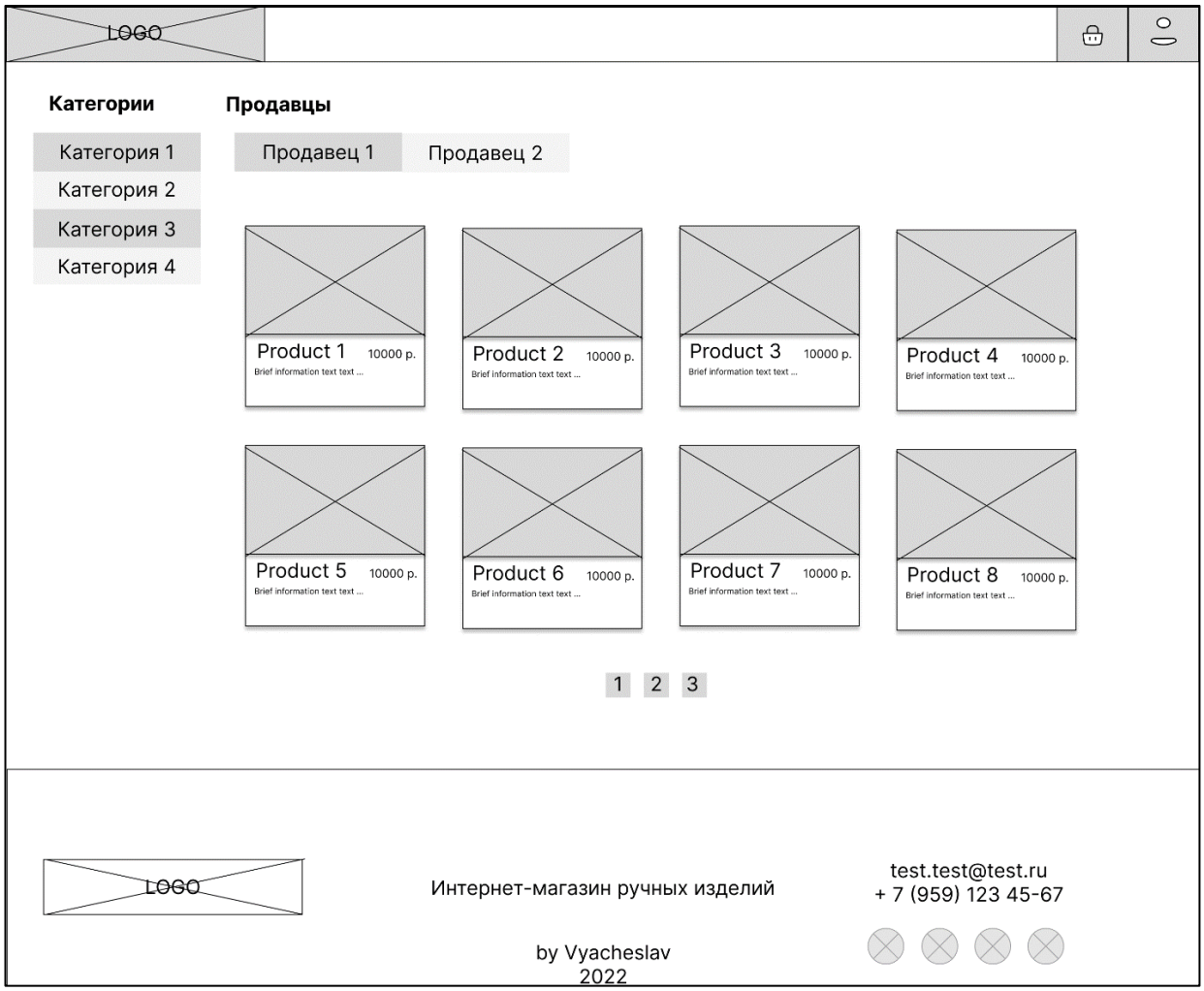

Рисунок 8 – Макет главной страницы

Основными объектами главной страницы является поиск товара по категориям и по продавцам.

Все проектируемые страницы содержат два одинаковых блока: header и footer. В шапке сайта расположено навигационное меню, логотип, корзина и профиль пользователя. В подвале сайта представлены контактная информация, ссылки на социальные сети. Информация между шапкой и

подвалом отображает тот или иной раздел сайта в зависимости от выбора пользователя.

Макет страницы товара изображен на рисунке 9.

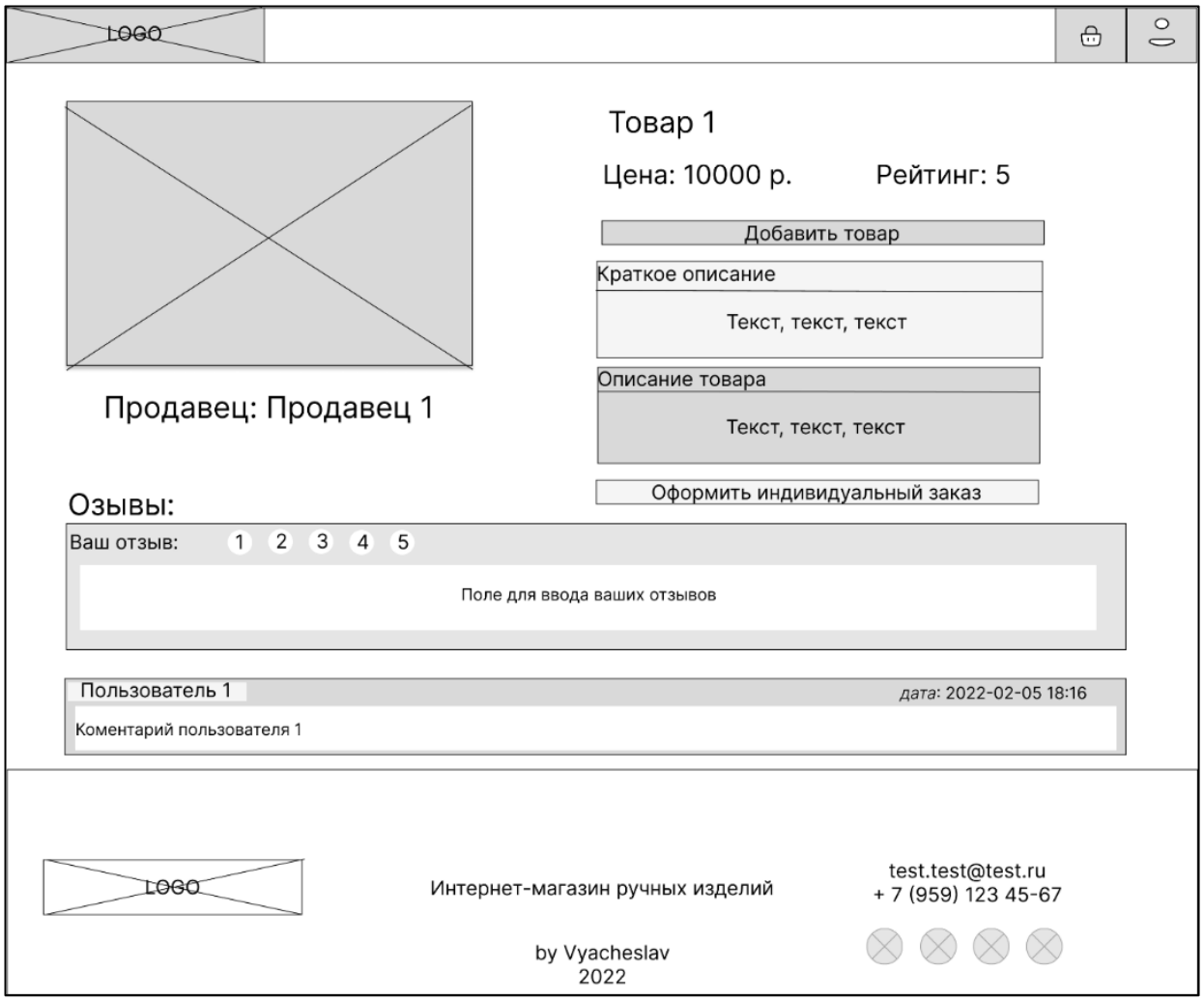

Рисунок 9 – Макет страницы товара

3.5 Описание данных

Схема спроектированной БД представлена на рисунке 10.

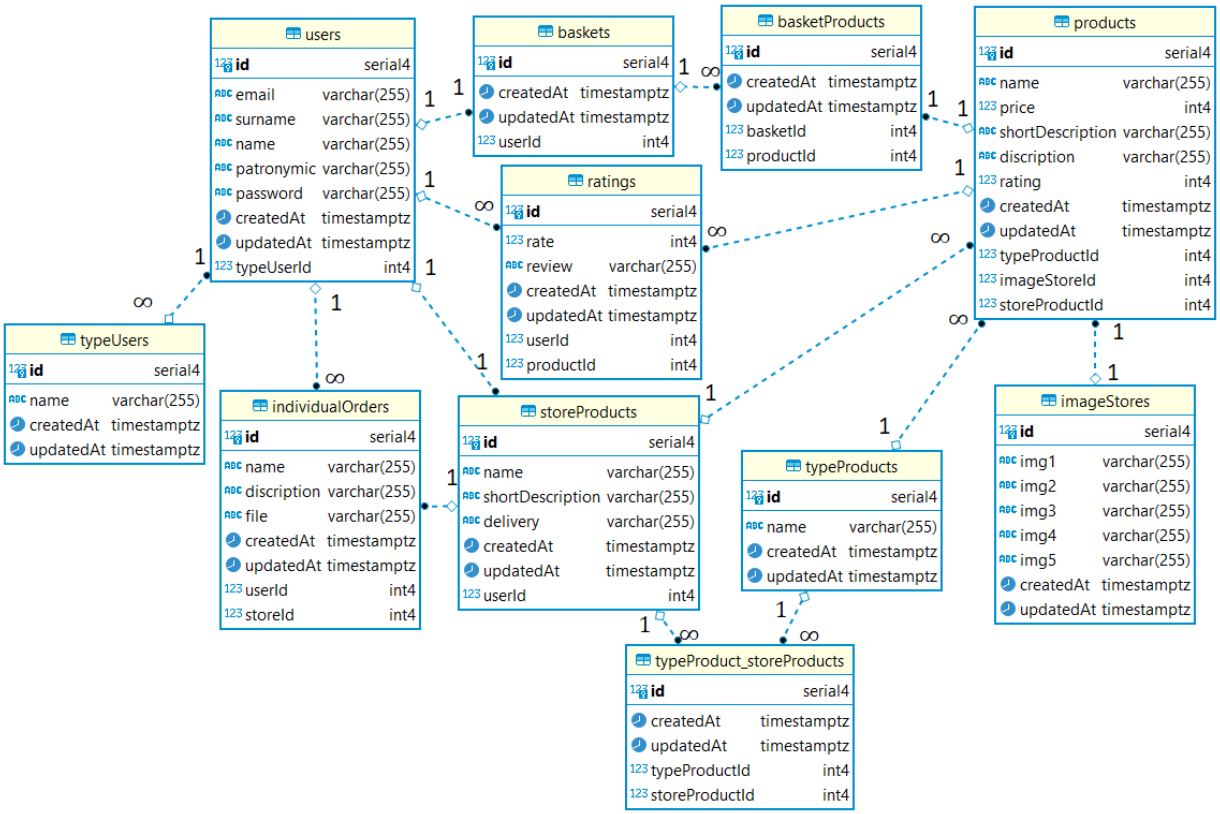

Рисунок 10 – Схема базы данных

В таблице 8 предоставлено описание полей таблицы «users», которая содержит информацию об авторизованном пользователе.

Таблица 8 – Таблица «users»

Название поля	Тип данных	Описание
id	INT	Идентификатор пользователя, первичный ключ
email	VARCHAR	Почта пользователя
surname	VARCHAR	Фамилия пользователя
name	VARCHAR	Имя пользователя
patronymic	VARCHAR	Отчество пользователя
password	VARCHAR	Зашифрованный пароль пользователя
typeUserId	INT	Идентификатор роли, внешний ключ из таблицы «typeUser»

В таблице 9 предоставлено описание полей таблицы «basket», которая содержит информацию об корзине пользователя.

Таблица 9 – Таблица «basket»

Название поля	Тип данных	Описание
id	INT	Идентификатор корзины, первичный ключ
userId	INT	Идентификатор пользователя, внешний ключ из таблицы «user»

В таблице 10 предоставлено описание полей таблицы «typeUser», которая содержит информацию о ролях пользователей доступных в системе.

Таблица 10 – Таблица «typeUser»

Название поля	Тип данных	Описание
id	INT	Идентификатор роли пользователя, первичный ключ
name	VARCHAR	Роль пользователя

В таблице 11 предоставлено описание полей таблицы «basketProduct», которая содержит информацию об товарах в корзине пользователя.

Таблица 11 – Таблица «basketProduct»

Название поля	Тип данных	Описание
id	INT	Идентификатор товаров в корзине, первичный ключ
productId	INT	Идентификатор товара, внешний ключ из таблицы «product»
basketId	INT	Идентификатор корзины пользователя, внешний ключ из таблицы «basket»

В таблице 12 предоставлено описание полей таблицы «rating», которая содержит информацию об пользователе, который указал рейтинг товара.

Таблица 12 – Таблица «rating»

Название поля	Тип данных	Описание
Id	INT	Идентификатор рейтинга на товар от пользователя, первичный ключ
userId	INT	Идентификатор пользователя, внешний ключ из таблицы «user»
productId	INT	Идентификатор товара, внешний ключ из таблицы «product»
rate	INT	Рейтинг товара
review	VARCHAR	Отзыв товара

В таблице 13 предоставлено описание полей таблицы «typeProduct», которая содержит информацию об типе изделия.

Таблица 13 – Таблица «typeProduct»

Название поля	Тип данных	Описание
id	INT	Идентификатор типа изделия, первичный ключ
name	VARCHAR	Имя типа продукта

В таблице 14 предоставлено описание полей таблицы «storeProduct», которая содержит информацию об магазине продавца.

Таблица 14 – Таблица «storeProduct»

Название поля	Тип данных	Описание
id	INT	Идентификатор магазина продавца, первичный ключ
name	VARCHAR	Имя магазина
shortDescription	VARCHAR	Краткая информация о магазине
userId	INT	Идентификатор пользователя, внешний ключ из таблицы «user»
delivery	VARCHAR	Информация о доставке

В таблице 15 предоставлено описание полей таблицы «product», которая содержит информацию об товаре.

Таблица 15 – Таблица «product»

Название поля	Тип данных	Описание
id	INT	Идентификатор товара, первичный ключ
name	VARCHAR	Название изделия
price	INT	Информация о стоимости изделия
shortDescription	VARCHAR	Краткая информация о товаре
description	VARCHAR	Полное описание товара
rating	INT	Рейтинг товара
imageStoreId	INT	Идентификатор хранилища фотографий, внешний ключ из таблицы «imageStore»
typeProductId	INT	Идентификатор типе товара, внешний ключ из таблицы «typeProduct»
storeProductId	INT	Идентификатор магазина продавца, внешний ключ из таблицы «storeProduct»

В таблице 16 предоставлено описание полей таблицы «tepeProduct_storeProducts», которая является дополнительной таблицей, реализую связь «многие ко многим» у таблиц «typeProduct» и «storeProduct».

Таблица 16 – Таблица «individualOrder»

Название поля	Тип данных	Описание
id	INT	Идентификатор индивидуального заказа, первичный ключ
typeProductId	INT	Идентификатор типа продукта, внешний ключ из таблицы «typeProduct»
storeProductId	INT	Идентификатор магазина продавца, внешний ключ из таблицы «storeProduct»

В таблице 17 предоставлено описание полей таблицы «individualOrder», которая содержит информацию о индивидуальном заказе.

Таблица 17 – Таблица «individualOrder»

Название поля	Тип данных	Описание
id	INT	Идентификатор индивидуального заказа, первичный ключ
name	VARCHAR	Название заказа
description	VARCHAR	Описание заказа
file	VARCHAR	Название прикрепляемого файла
userId	INT	Идентификатор пользователя, внешний ключ из таблицы «user»
storeId	INT	Идентификатор магазина продавца, внешний ключ из таблицы «storeProduct»

В таблице 18 предоставлено описание полей таблицы «imageStore», которая содержит информацию о фотографиях товара.

Таблица 18 – Таблица «imageStore»

Название поля	Тип данных	Описание
id	INT	Идентификатор хранилища фотографий, первичный ключ
img1	VARCHAR	Название файла фотографии товара №1
img2	VARCHAR	Название файла фотографии товара №2
img3	VARCHAR	Название файла фотографии товара №3
img4	VARCHAR	Название файла фотографии товара №4
img5	VARCHAR	Название файла фотографии товара №5

Кроме того, всем таблицам присущи поля «createdAt» и «updatedAt» с типом данных timestamptz, для определения создания и обновления записи соответственно по дате и времени с часовым поясом [12].

4 РЕАЛИЗАЦИЯ

4.1 Форма регистрации и авторизации

Страница форм авторизации и регистрации реализованы в одном компоненте Authorization.jsx, который отвечает за авторизацию и регистрацию пользователя. Формы представлены на рисунках 11 и 12 соответственно.

На данную страницу можно попасть через навигационную панель приложения. Она доступна пользователю с ролью «Гость». После авторизации или регистрации пользователя перешлет на главную страницу.

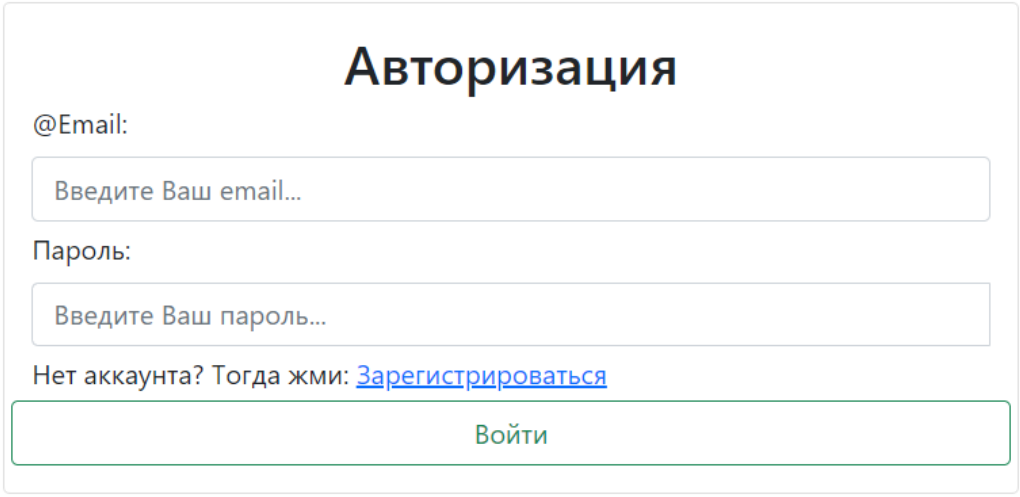

Рисунок 11 – Форма авторизации

При авторизации пользователь должен ввести email и пароль, который он указывал при регистрации. После ввода данных происходит валидация (процесс обеспечения точности и качества данных) со стороны клиента. Если валидация не пройдена, на форме отобразятся подсказки по заполнению форм. При успешном прохождении данной проверки отправляется запрос на сервер, который проверит данные пользователя на наличие в БД и присвоит пользователю токен. Авторизация и регистрация осуществлена по JWT токену [13]. Он состоит из трех основных частей: заголовка (header), нагрузки (payload) и подписи (signature). Header содержит служебную часть о типе и алгоритме шифрования, payload содержит данные, которые пользователь ввел, signature содержит закодированную с помощью секретного ключа, хранящегося на сервере, часть header и payload. На стороне сервера

происходит проверка на наличие пользователя в системе, в случае успеха происходит сравнивание пароля, указанного пользователем, с хранимым в зашифрованном виде паролем в БД. Если пароль пользователя совпал, то генерируется JWT токен, который в дальнейшем допускает пользователя к функционалу, доступному определенному типу пользователей.

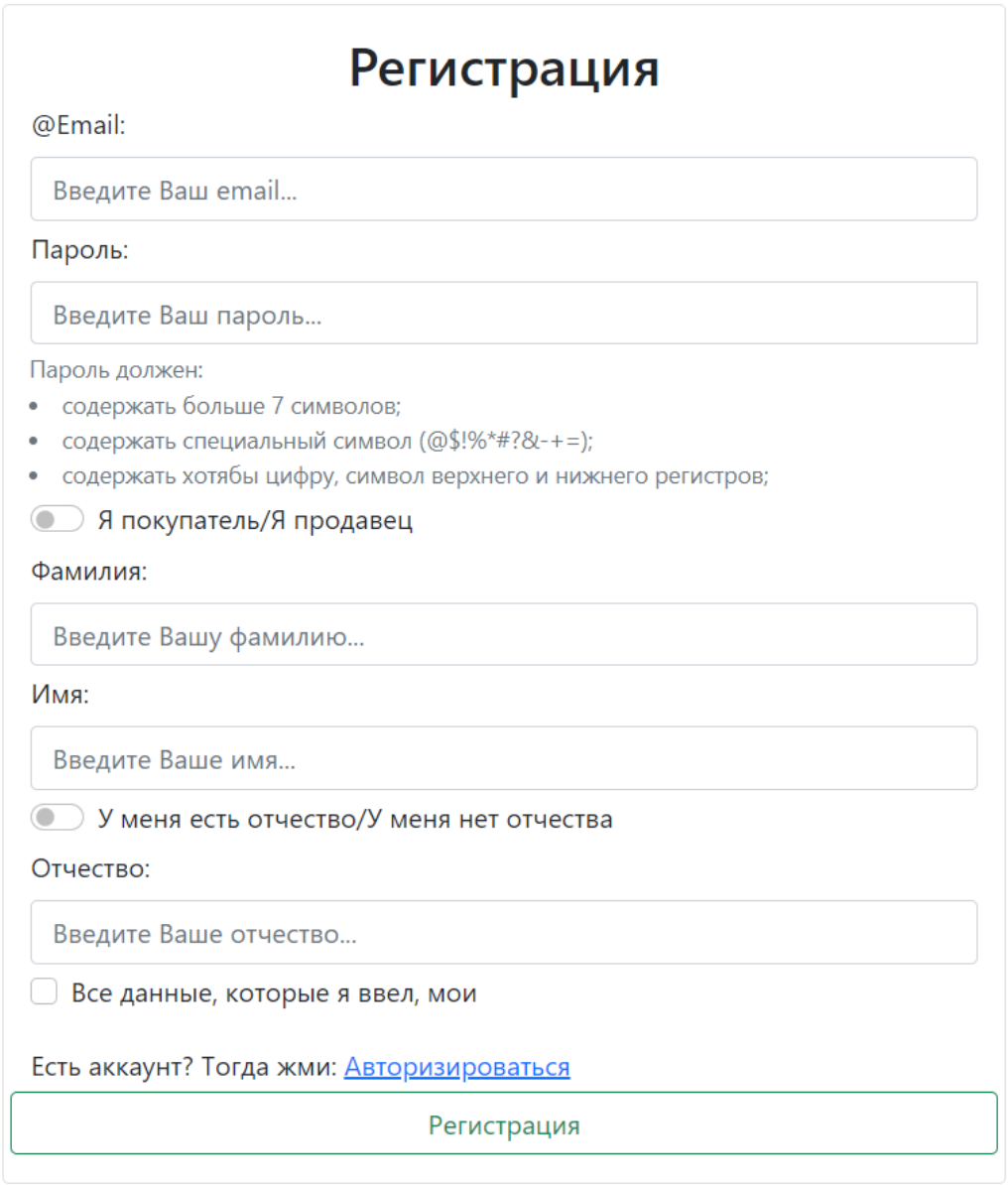

Рисунок 12 – Форма регистрации

При регистрации пользователь должен ввести обязательные поля: email, пароль и имя, кроме этого, должен установить чек-бокс (от англ. check box, галочка – элемент графического пользовательского интерфейса) для подтверждения того, что его данные указаны верно. Данный чек-бокс при

дальнейшей разработке будет изменен на согласие пользователя с политикой конфиденциальности.

В компоненте userController.js описан класс UserController, который содержит методы взаимодействия веб-приложения с сервером: registration (post), authorization (post) и ifAuthorization (get). В листинге 1 приведен один из методов взаимодействия (registration).

Листинг 1 – Метод registration в класс UserController

```
class UserController {
    async registration(req, res, next) {
        const {email, password, name, surname, patronymic, typeUserId}
= req.body
        if (!email || !password) {
            return next(APIError.badRequest('Некорректный email или
пароль'))
        }
        const candidate = await User.findOne({where: {email}})
        if (candidate) {
            return next(APIError.badRequest('Пользователь с таким
email уже существует'))
        }
        const hashPassword = await bcrypt.hash(password, 5)
        const user = await User.create({email, name, surname,
patronymic, typeUserId, password: hashPassword})
        const basket = await Basket.create({userId: user.id})
        const token = generateJwt(user.id, user.email,
user.typeUserId)
        return res.json({token})
    }
/*   */
}
```

Рассмотрим метод registration. Для него необходимо в теле запроса передавать все имеющиеся у пользователя поля на форме регистрации. Если email или пароль не были переданы или отправленный email уже существует в системе, то будет возвращен код статус 400 и соответствующее сообщение об ошибке. При успешной проверке пароль пользователя хешируется, и в БД создается запись с данными пользователя, генерируется токен для дальнейшего использования.

4.2 Основная страница

Все страницы имеют навигационную панель, на которой в зависимости от роли меняется наполнение. Так пользователь с ролью «Гость» имеет

возможность перейти на страницу авторизации или регистрации, а авторизированным пользователям доступна ссылка на страницу профиля и кнопка для выхода из системы. Кнопка на страницу корзина доступно всем пользователям.

Главная страница представляет собой каталог товаров, где можно выбрать тип товара и продавца, так же можно перейти на страницу товара, кликнув на сам товар. В случае если пользователь введет несуществующий URL, его перенаправят на главную страницу. На рисунке 13 представлена главная страница интернет-магазина.

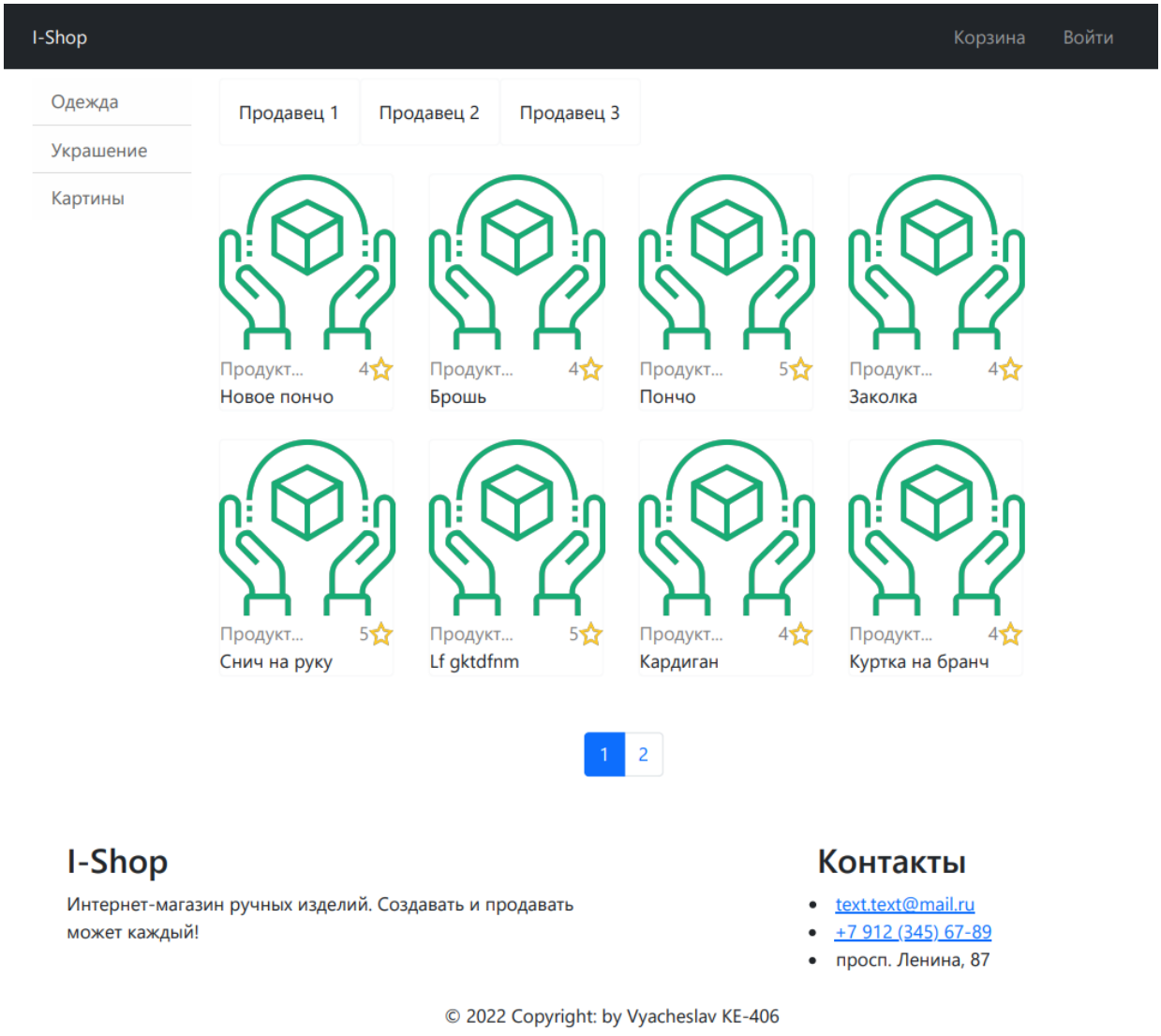

Рисунок 13 – Главная страница

Главная страница представляет собой 3 основных компонента: TypeBar.jsx, SellerProductBar.jsx и ProductList.jsx, которые отвечают за

отрисовку фильтра типов и продавцов и карточек товара. Первый компонент продемонстрирован в листинге 2.

Листинг 2 – Компонент TypeBar

```
const TypeBar = observer( () => {
    const {product} = useContext(Context)

    return (
        <ListGroup variant="flush">
            {product.typeProducts.map(typeProduct =>
                <ListGroup.Item
                    style={{cursor: 'pointer'}}
                    active={typeProduct.id ===
product.selectedTypeProduct.id}
                    onClick={() =>
product.setSelectedTypeProduct(typeProduct)}
                    key={typeProduct.id}
                >
                    {typeProduct.name}
                </ListGroup.Item>

            )}
        </ListGroup>
    );
});
```

Для управления состоянием приложения используется библиотека MobX. Компонент TypeBar получает данные из глобального состояния при помощи компонента высшего порядка (HOC – это продвинутая технология в React для повторного использования логики компонента) observer, который оборачивает react компонент. Observer наблюдает за изменением объектов в состоянии, в данном компоненте – это типы товаров. Это делается для того, чтобы компонента автоматически перерисовывалась при изменении наблюдаемых объектов (typeProducts). По аналогии реализуется компонент SellerProductBar, но взаимодействует с продавцами.

Получение данных на клиент происходит путем отправки get запросов для получения всех данных по продавцам, типам и товарам. Отдельный get запрос на получение списка товаров будет выглядеть следующим образом: *http://localhost:4060/api/product?typeProductId=2&storeProductId=3&limit=8&page=1*. Где в теле параметрами передаются id типа и продавца, лимит отображения карточек и номер страницы. Пример тела ответа в формате JSON представлен в листинге 3.

Листинг 3 – Ответ на GET запрос, для получения данных


```
{
    "count": 1,
    "rows": [
        {
            "id": 5,
            "name": "Снич на руку",
            "price": 20000,
            "shortDescription": "Для любителей Гари Потера",
            "discription": " Выиграй кубок",
            "rating": 5,
            "img": "d883201c-d9c1-4ce7-ac7c-3408cd2a7c15.jpg",
            "createdAt": "2022-05-29T12:28:26.896Z",
            "updatedAt": "2022-05-29T12:28:26.896Z",
            "typeProductId": 2,
            "imageStoreId": 5,
            "storeProductId": 3
        }
    ]
}
```


На стороне сервера класс ProductController отвечает за взаимодействие с базой данных товаров. Код метода getAll, который возвращает полный список товаров по параметрам, представлен в листинге 4.

Листинг 4 – Метод getAll в классе ProductController

```
class ProductController {
/*   */
    async getAll(req, res, next) {
        try {
            let {typeProductId, storeProductId, limit, page} =
req.query
            page = page || 1
            limit = limit || 9
            let offset = page * limit - limit
            let products;

            if (!storeProductId && !typeProductId) {
                products = await Product.findAndCountAll({limit,
offset: offset})
            }
            if (storeProductId && !typeProductId) {
                products = await Product.findAndCountAll({where:
{storeProductId}, limit, offset: offset})
            }
            if (!storeProductId && typeProductId) {
                products = await Product.findAndCountAll({where:
{typeProductId}, limit, offset: offset})
            }
            if (storeProductId && typeProductId) {
                products = await Product.findAndCountAll({where:
{storeProductId, typeProductId}, limit, offset: offset})
            }

            return res.json(products)
```

```
        }
        catch (e) {
            next(APIError.notFound(e.message))
        }
    }
/*   */
}
```

Асинхронная функция может принять несколько параметров, параметры могут быть как указаны, так и нет, для этого предусмотрены несколько случаев. Для параметра номера страницы и лимита назначены значения по умолчанию, а для типов товара и продавцов прописаны условия для каждого из состояний [14]. Когда будет отправлен запрос на данные позиции товаров вернется ответ в формате JSON, который продемонстрирован в листинге 3.

4.3 Страница товара

При переходе на страницу конкретного товара посетитель может ознакомиться с информацией о товаре, добавить его в корзину или прочитать отзывы о нем. Страницу товара приведена на рисунке 14.

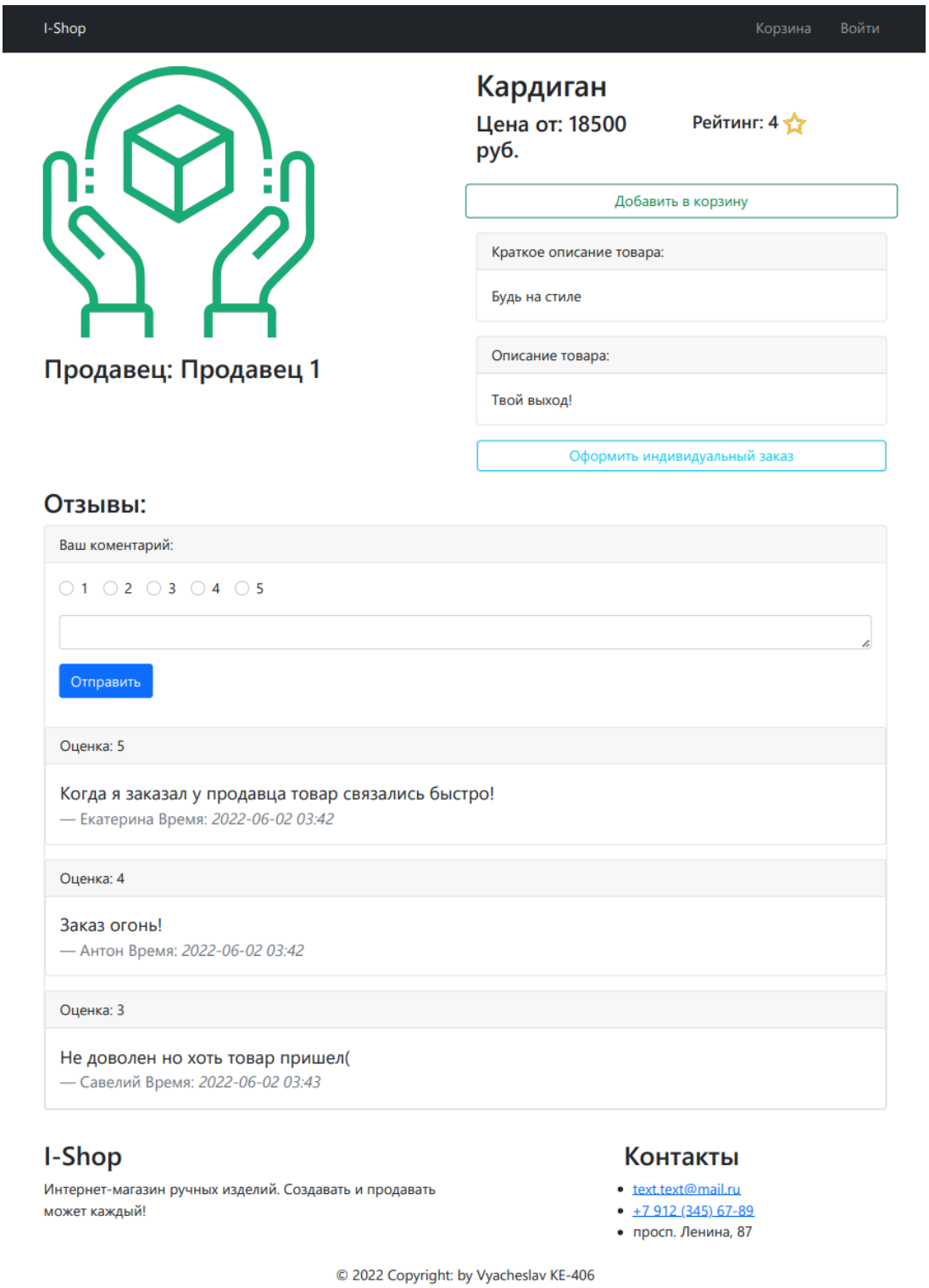

Рисунок 14 – Страница товара

Страница представляет собой два компонента ProductInfo.jsx и RetingList.jsx. Первый компонент отвечает за отрисовку информации о товаре, а второй за отзывы к товару. Для получения данных о товаре отправляется GET запрос с идентификатором данного товара. FetchOneProduct – функция, которая со стороны клиента отправляет запрос на сервер, а потом сохраняет

ответ от сервера в переменную. Функция fetchOneProduct приведена в листинге 5.

Листинг 5 – Функция fetchOneProduct

```
export const fetchOneProduct = async (id) => {
    const {data} = await $host.get('api/product/' + id)
    return data
}
```

4.4 Страница профиля

В зависимости от роли пользователя («Покупатель» или «Продавец») страница будет иметь уникальный вид. Так пользователю «Покупатель» будет доступна информация о нем, а для пользователя «Продавец» будут доступны все возможности профиля. Профиль «Продавца» продемонстрирован на рисунке 15.

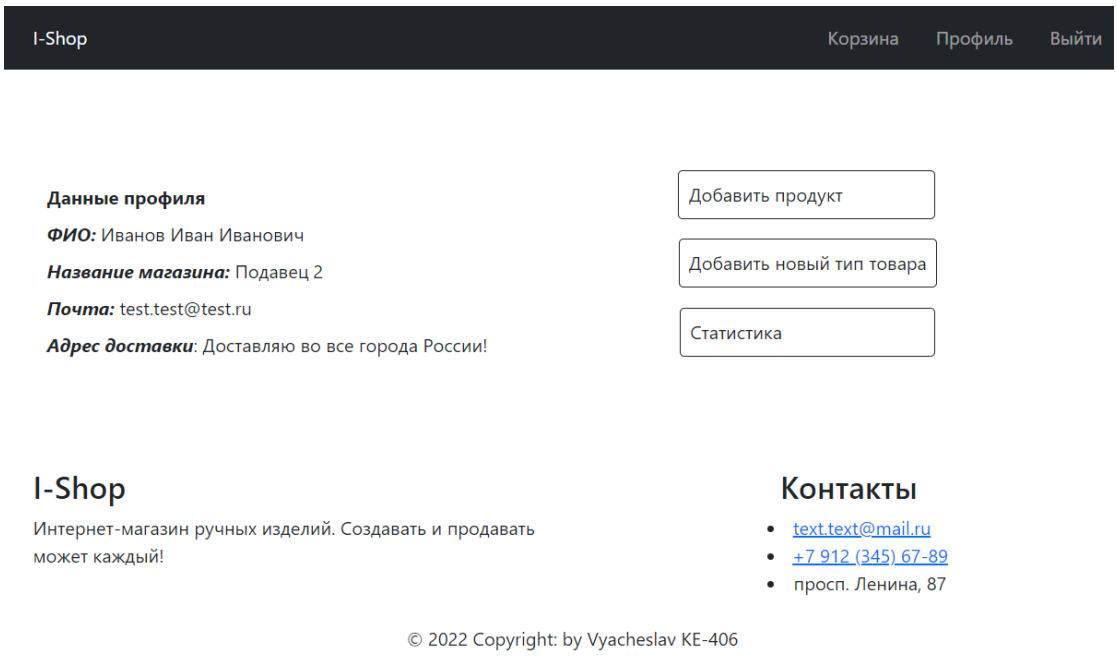

Рисунок 15 – Страница профиль, пользователя «продавец»

4.5 Корзина

Страница корзины доступна всем пользователям. Для пользователей «Продавец» и «Покупатель» данные их корзины хранятся в БД, при каждом добавлении товара в корзину отправляется запрос на добавление информации в таблицу basket. А для пользователя «Гость» информация хранится на

клиенте, и при обновлении страницы данные корзины исчезнут. Но так как интернет-магазин является SPA приложением, то при навигации по сайту данные корзины будут продолжать храниться в состоянии приложения. Корзина продемонстрирована на рисунке 16.

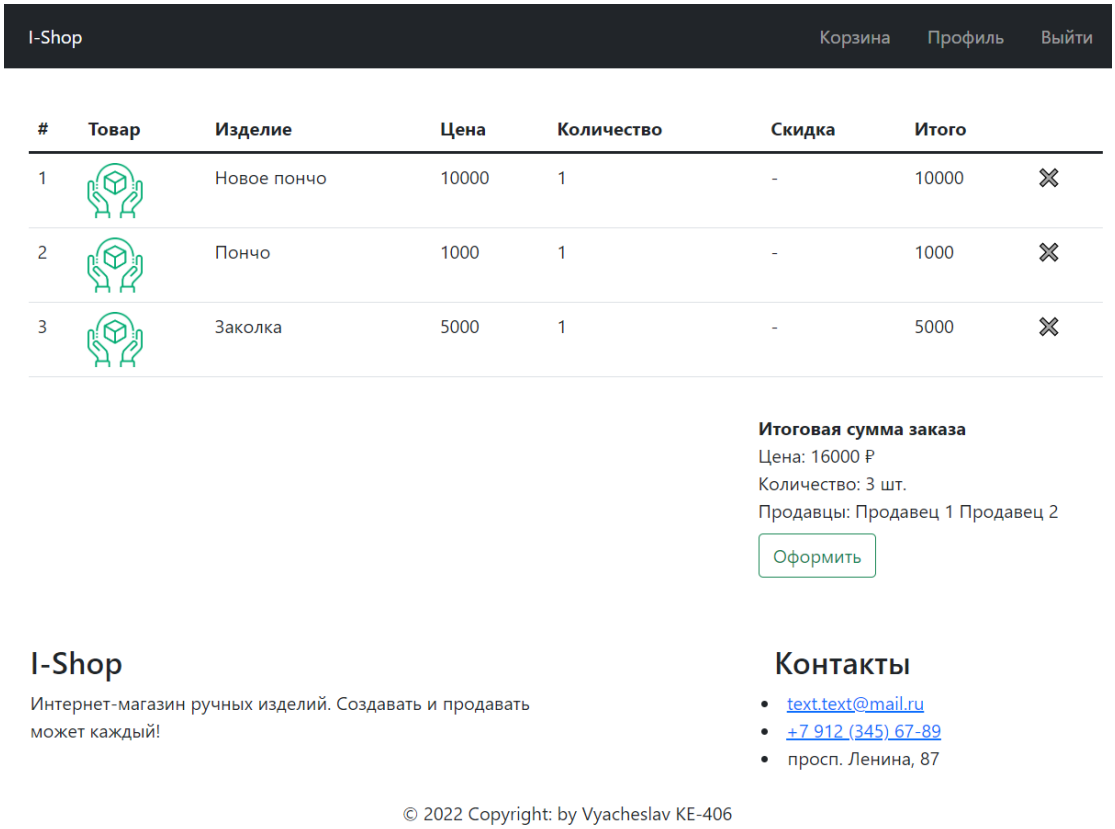

Рисунок 16 – Страница корзина

5 ТЕСТИРОВАНИЕ

5.1 Функциональное тестирование

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям.

Результаты функционального тестирования приведены в таблице 19 и совпадают с ожидаемыми. Функциональное тестирование пройдено успешно.

Таблица 19 – Протокол тестирования

№	Название теста	Шаги выполнения	Ожидаемый результат	Тест пройден?
1	Просмотр товаров	Зайти на главную страницу сайта	Вывод всех товаров на экран	Да
2	Просмотр отдельной карточки товара	1) Зайти на главную страницу сайта 2) Нажать на карточку любого товара	Вывод на экран информации о выбранном товаре	Да
3	Авторизация пользователя в системе	1) Зайти на главную страницу сайта под ролью «Гость» 2) Нажать на кнопку «Войти» 3) Заполнить поля формы 4) Нажать кнопку «Войти»	При успешной авторизации пользователю показывают главную страницу	Да
4	Валидация полей формы авторизации	1) Зайти на главную страницу сайта под ролью «Гость» 2) Нажать на кнопку «Войти» 3) Заполнить поля формы с некорректными данными 4) Нажать кнопку «Войти»	На странице авторизации поля формы, которые не прошли валидацию, будут подсвечены красным, а снизу будет указан текст ошибки	Да
5	Регистрация пользователя в системе	1) Зайти на главную страницу сайта под ролью «Гость» 2) Нажать на кнопку «Войти» 3) Нажать на кнопку «Регистрация» 4) Заполнить поля формы 5) Нажать кнопку «Регистрация»	При успешной регистрации пользователю показывают главную страницу	Да

Продолжение таблицы 19

№	Название теста	Шаги выполнения	Ожидаемый результат	Тест пройден?
6	Валидация полей формы регистрации	1) Зайти на главную страницу сайта под ролью «Гость» 2) Нажать на кнопку «Войти» 3) Нажать на кнопку «Регистрация» 4) Заполнить поля формы с некорректными данными 5) Нажать кнопку «Регистрация»	На странице регистрации поля формы, которые не прошли валидацию, будут подсвечены красным, а снизу будет указан текст ошибки	Да
7	Просмотр товаров отдельной категории	1) Зайти на главную страницу сайта 2) Выбрать любую категорию товара	Вывод всех товаров на экран выбранной категории	Да
8	Просмотр товаров отдельного продавца	1) Зайти на главную страницу сайта 2) Выбрать любого продавца	Вывод всех товаров на экран выбранного продавца	Да
9	Добавление товара в корзину	1) Зайти на главную страницу сайта 2) Нажать на карточку любого товара 3) Нажать на кнопку «Добавить товар»	На странице товара кнопка «Добавить товар» изменяется на «Товар уже в корзине». В корзине добавляется выбранный товар	Да
10	Просмотр товаров в корзине	1) Зайти на главную страницу сайта 2) Нажать на кнопку «Корзина»	На странице корзина должно отобразиться все добавленные в корзину товары	Да
11	Удаление товара из корзины	1) Зайти на главную страницу сайта 2) Нажать на кнопку «Корзина» 3) Нажать на кнопку удалить товар (крест) у любого товара	На странице корзина должно пропасть отображение удаленного товара	Да
12	Написать отзыв к товару авторизированным пользователем	1) Зайти на главную страницу сайта 2) Нажать на карточку любого товара 3) Нажать на одну из цифр оценки 4) Написать в форму текстовое сообщение 5) Нажать на кнопку «Отправить»	На странице товара должен появиться отзыв с поставленной оценкой. Оценка товара должна измениться или остаться неизменной.	Да
13	Написать отзыв к товару не авторизированным пользователем		На странице товара отобразиться сообщение «Оставить отзыв могут авторизированные пользователи»	Да

Продолжение таблицы 19

№	Название теста	Шаги выполнения	Ожидаемый результат	Тест пройден?
14	Просмотр профиля пользователем с ролью «Покупатель»	1) Зайти на главную страницу сайта 2) Нажать на кнопку «Профиль»	На странице профиля должна отобразиться информация о пользователе (email, Фамилия и Имя, Отчество если есть и роль пользователя (Покупатель))	Да
15	Просмотр профиля пользователем с ролью «Продавец»	1) Зайти на главную страницу сайта 2) Нажать на кнопку «Профиль»	На странице профиля должна отобразиться информация о пользователе (email, Фамилия и Имя, Отчество если есть, название магазина, адрес доставки) и кнопки: «Статистика», «Добавить новый товар» и «Статистика»	Да
16	Оформление индивидуального заказа	1) Зайти на главную страницу сайта 2) Нажать на карточку любого товара 3) Нажать на кнопку «Оформить индивидуальный заказ» 4) Заполнить модальное (название, выбор продавца, текст к заказу, файл и email 5) Нажать на кнопку «Отправить заказ на обработку»	На странице товара должно отобразиться сообщение «Ваш заказ принят на обработку»	Да
17	Добавление товара авторизированным пользователем с ролью «Продавец»	1) Зайти на главную страницу сайта 2) Нажать на кнопку «Профиль» 3) Нажать на кнопку «Добавить продукт» 4) Заполнить форму в модальном окне (поля: название, краткое описание, подробное описание товара, стоимость товара и прикрепить фотографии) 5) Нажать на кнопку «Добавить»	На странице «Профиль» должно отобразится сообщение «Ваш товар был успешно добавлен». На главной странице должен отобразится добавленный товар.	Да

Окончание таблицы 19

№	Название теста	Шаги выполнения	Ожидаемый результат	Тест пройден?
18	Добавление нового типа товара авторизированным пользователем с ролью «Продавец»	1) Зайти на главную страницу сайта 2) Нажать на кнопку «Профиль» 3) Нажать на кнопку «Добавить новый тип товара» 4) Заполнить форму в модальном окне (поле: название типа) 5) Нажать на кнопку «Добавить»	На странице «Профиль» должно отобразится сообщение «Добавлен новый тип товара». На главной странице должен отобразится добавленный новый тип товара.	Да

5.2 Тестирование верстки

Было проведено тестирование интерфейса веб-сайта. Данный веб-сайт был протестирован в следующих браузерах:

1) Google Chrome версии 50 и выше;
2) Яндекс.Браузер версии 16.9 и выше
3) Mozilla Firefox версии 50.2. и выше;
4) Safari версии 5 и выше;
5) Opera версии 40 и выше;
6) Microsoft Edge версии 38 и выше;
7) Mobile Chrome версии 30 и выше.

5.3 Тестирование адаптивности

Веб-приложение адаптируется под размер экрана устройства за счет дополнительных стилей, прописанных для компонент, что позволяет положению подстраиваться под любые расширения у пользователя. Непомещающиеся элементы навигации будут сворачиваться, доступ к ним можно получить, нажав на появившуюся кнопку, которая вызовет меню. На рисунках 17 и 18 представлены примеры главной страница для планшета (iPad Air с расширением 1180x820 pi) и телефона (iPhone SE с расширением 375x667

pi) соответственно. На рисунке 19 демонстрируется пример корзины для телефона.

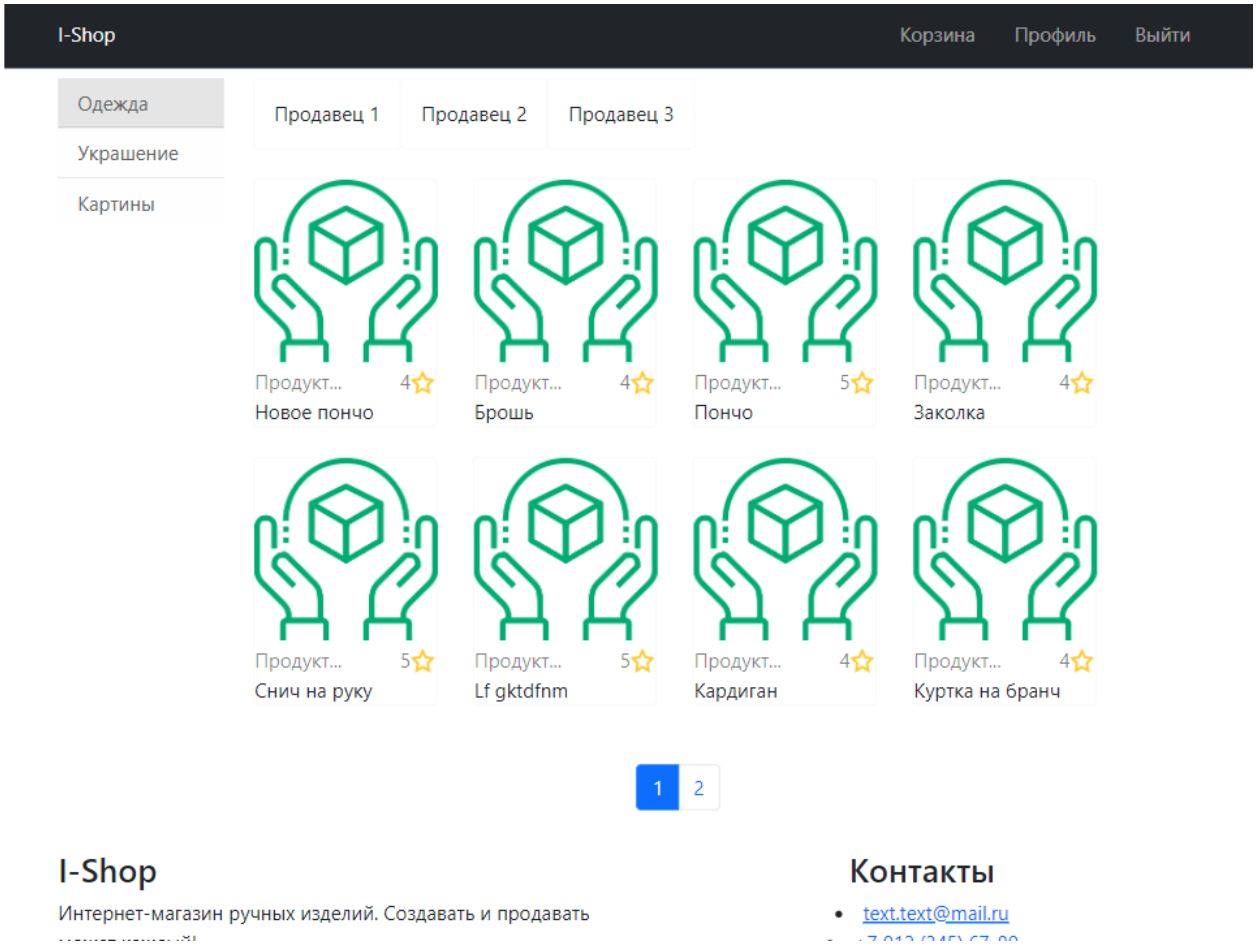

Рисунок 17 – Главная страница (планшет)

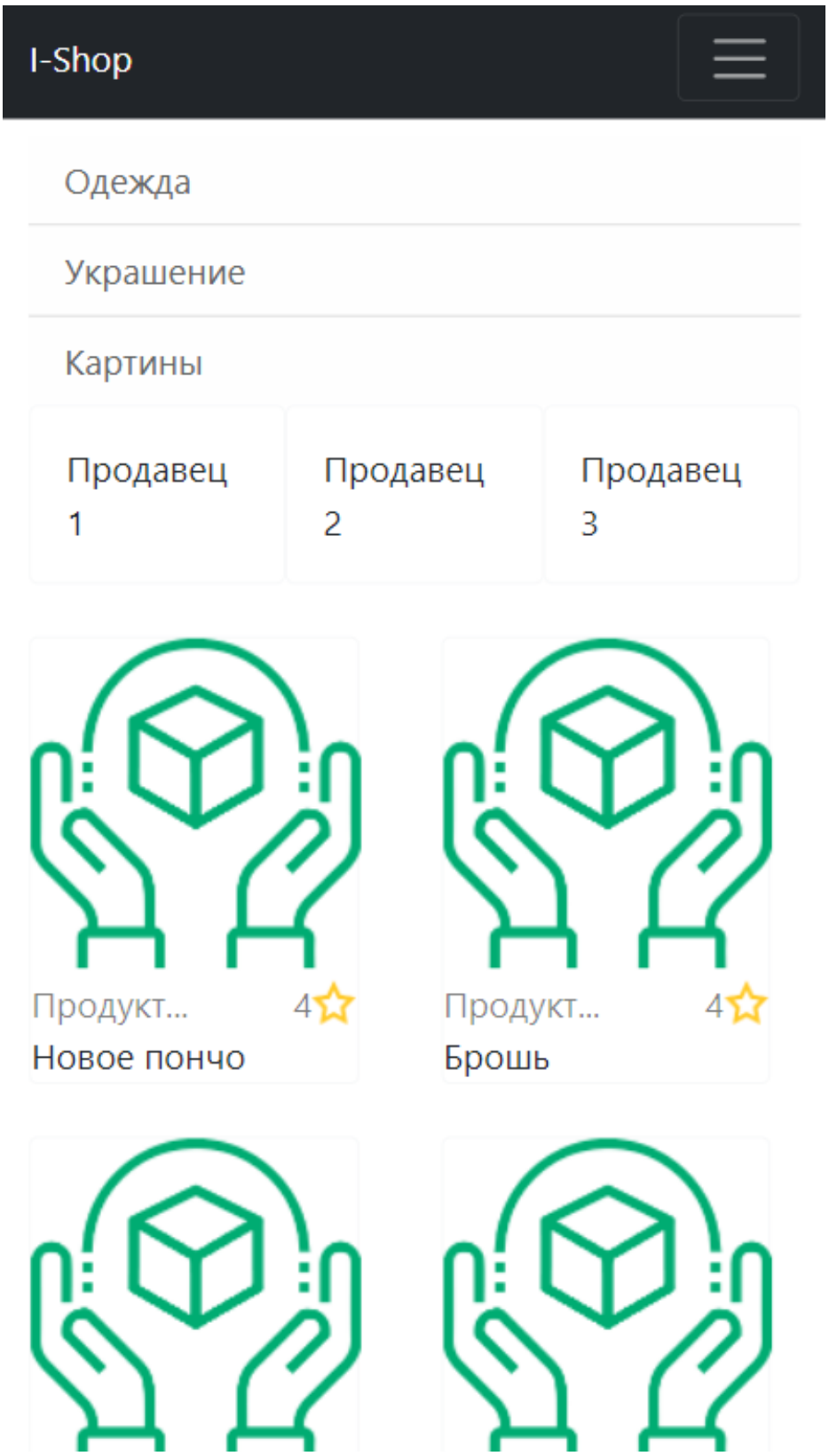

Рисунок 18 – Главная страница (телефон)

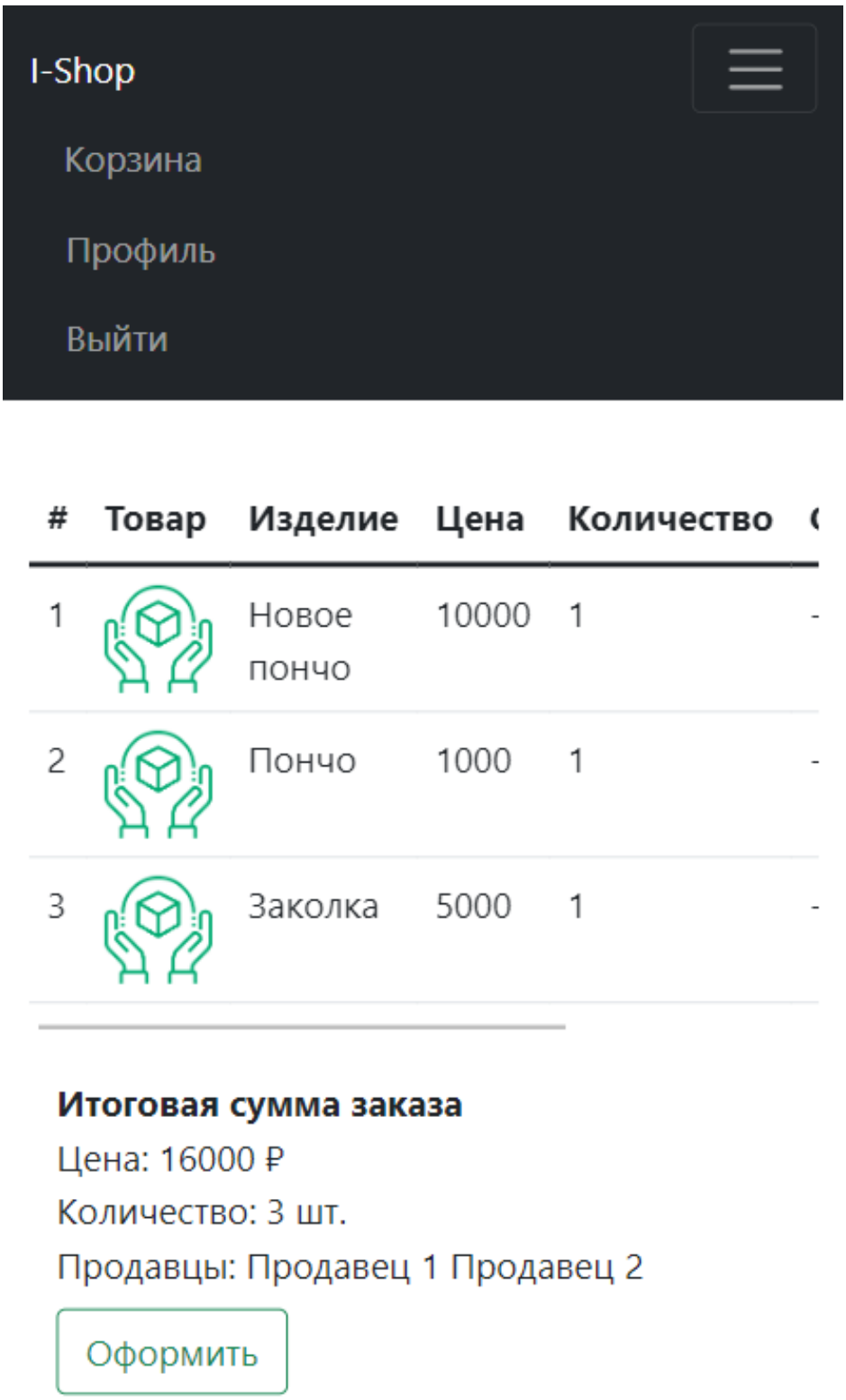

Рисунок 19 – Страница корзины (телефон)

5.4 Вывод по тестированию

Для разрабатываемой системы на основе функциональных требований был сформирован набор тестов, с помощью которых было проведено функциональное тестирование, тестирование верстки и тестирование адаптивности. Все тесты из набора были выполнены успешно. По результатам тестирования можно сделать выводы о том, что разработанная система соответствует требованиям.

ЗАКЛЮЧЕНИЕ

Целью данной работы является разработка интернет-магазина для продажи изделий ручной работы.

В ходе выполнения выпускной квалификационной работы бакалавра были решены следующие задачи:

1) выполнен анализ предметной области;

2) разработаны требования к интернет-магазину;

3) спроектирован интернет-магазин;

4) реализован интернет-магазин для изделий ручной работы;

5) проведено тестирование интернет-магазина.

В результате выполнения выпускной квалификационной работы были решены все поставленные задачи, следовательно, достигнута цель данной работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Интернет-магазин «Ярмарка Мастеров». – Текст. Изображение : электронные // Ярмарка Мастеров : [официальный сайт]. – URL: https://www.livemaster.ru (дата обращения: 24.04.2022).

2. Интернет-магазин «Hands4U». – Текст. Изображение : электронные // Ярмарка мастеров Hands4U : [официальный сайт]. – URL: https://handsforyou.ru (дата обращения: 24.04.2022).

3. Интернет-магазин «Торговый дом мастеров». – Текст. Изображение : электронные // Торговый Дом Мастеров : [официальный сайт]. – URL: https://tdmasterov.ru (дата обращения: 24.04.2022).

4. Что такое SPA или одностраничный портал. – Текст : электронный // Calabonga SOFT : [сайт]. – 26.04.2022. – URL: https://www.calabonga.net/blog/post/chto-takoe-spa-ili-odnostranichnyi-portal (дата обращения: 27.04.2022).

5. Определение CMS сайта. – Текст : электронный // be1.ru : [сайт]. – URL: https://be1.ru/cms/ (дата обращения: 24.04.2022).

6. Платформа Node.js : [официальный сайт] / JavaScript-окружение построенное на движке Chrome V8. – URL: https://nodejs.org/en/ (дата обращения: 29.04.2022). – Текст : электронный.

7. Веб-фреймворк Express : [официальный сайт] / Быстрый, гибкий, минималистичный веб-фреймворк для приложений Node.js. – URL: https://expressjs.com (дата обращения: 29.04.2022). – Текст : электронный.

8. Библиотека React : [официальный сайт] / JavaScript-библиотека для создания пользовательских интерфейсов. – URL: https://reactjs.org (дата обращения: 29.04.2022). – Текст : электронный.

9. СУБД PostgreSQL : [официальный сайт] / Самая передовая в мире реляционная база данных с открытым исходным кодом. – URL: https://www.postgresql.org (дата обращения: 29.04.2022). – Текст : электронный.

10. Гайнанова, Р. Ш. Создание клиент-серверных приложений / Р. Ш. Гайнанова, О. А. Широкова. – Текст : электронный // Вестник Казанского технологического университета. – 2017. – № 9. – URL: https://cyberleninka.ru/article/n/sozdanie-klient-servernyh-prilozheniy (дата обращения: 02.06.2022).

11. Унифицированный язык моделирования UML : [официальный сайт] / Unified Modeling Language. – URL: https://www.uml.org (дата обращения: 29.04.2022). – Текст : электронный.

12. Брешенков, А. В. Методика проектирования реляционных баз данных / А. В. Брешенков. – Текст : электронный // Инженерный журнал: наука и инновации. – 2013. – № 11 (23). – URL: https://cyberleninka.ru/article/n/metodika-proektirovaniya-relyatsionnyh-baz-dannyh (дата обращения: 03.06.2022).

13. Пять простых шагов для понимания JSON Web Tokens (JWT). – Текст. Изображение (неподвижное ; двухмерное) : электронные // Habr : [сайт]. – 16 октября 2017. – URL: https://habr.com/ru/post/340146/ (дата обращения: 28.05.2022).

14. Ивонин, О. А. Создание клиент-серверного приложения на основе restful api архитектуры / О. А. Ивонин, А. Г. Гречихин, И. Р. Гильматдинов. – Текст : электронный // Молодой ученый. – 2021. – № 29 (371). – С. 18-20. – URL: https://moluch.ru/archive/371/83265/ (дата обращения: 01.06.2022).

ПРИЛОЖЕНИЕ А

Серверная часть разработанного интернет-магазина

Листинг А.1 – Инициализация серверной части в файле index.js

```
require('dotenv').config()
const express = require('express')
const sequelize = require('./bd')
const models = require('./models/models')
const cors = require('cors')
const fileUpload = require('express-fileupload')
const router = require('./routes/index')
const path = require('path')
const errorHandler = require ('./midlewere/ErrorHandlingMiddelewere')
const PORT = process.env.PORT || 5000
const app = express()

app.use(cors())
app.use(express.json())
app.use(express.static(path.resolve(__dirname, 'static')))
app.use(fileUpload({}))
app.use('/api', router)
app.use(errorHandler)

const start = async () => {
    try {
        await sequelize.authenticate()
        await sequelize.sync()
        app.listen(PORT, () => console.log(`Server started on port
${PORT}`))
    } catch (e) {
        console.log(e)
    }
}
start()
```

Листинг А.2 – Модель базы данных для взаимодействия с системой в файле madels.js

```
const sequelize = require('../bd')
const {DataTypes} = require('sequelize')
const User = sequelize.define('user', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
    email: {type: DataTypes.STRING, unique:true},
    surname: {type: DataTypes.STRING},
    name: {type: DataTypes.STRING},
    patronymic: {type: DataTypes.STRING},
    password: {type: DataTypes.STRING},
})

const Basket = sequelize.define('basket', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
})
```

```
const BasketProduct = sequelize.define('basketProduct', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
})

const Product = sequelize.define('product', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
    name: {type: DataTypes.STRING, unique:true, allowNull: false},
    price: {type: DataTypes.INTEGER, allowNull: false},
    shortDescription: {type: DataTypes.STRING},
    discription: {type: DataTypes.STRING},
    rating: {type: DataTypes.INTEGER, defaultValue: 0},
    img: {type: DataTypes.STRING, allowNull: false},
})

const TypeProduct = sequelize.define('typeProduct', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
    name: {type: DataTypes.STRING, unique: true,}
})

const StoreProduct = sequelize.define('storeProduct', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
    name: {type: DataTypes.STRING, unique: true, allowNull: false},
    shortDescription: {type: DataTypes.STRING},
    delivery: {type: DataTypes.STRING},
})

const Rating = sequelize.define('rating', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
    rate: {type: DataTypes.INTEGER, allowNull: false},
    review: {type: DataTypes.STRING},
})

const IndividualOrder = sequelize.define('individualOrder', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
    name: {type: DataTypes.STRING, allowNull: false},
    description: {type: DataTypes.STRING, allowNull: false},
    file: {type: DataTypes.STRING},
})

const ImageStore = sequelize.define('imageStore', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
    img1: {type: DataTypes.STRING, allowNull: false},
    img2: {type: DataTypes.STRING},
    img3: {type: DataTypes.STRING},
    img4: {type: DataTypes.STRING},
    img5: {type: DataTypes.STRING},
})
```

```
const TypeUser = sequelize.define('typeUser', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
    name: {type: DataTypes.STRING, unique: true,},
})

const TypeProduct_StoreProduct =
sequelize.define('typeProduct_storeProduct', {
    id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement:true},
})

User.hasOne(Basket)
Basket.belongsTo(User)

User.hasMany(Rating)
Rating.belongsTo(User)

User.hasMany(IndividualOrder)
IndividualOrder.belongsTo(User)

User.hasOne(StoreProduct)
StoreProduct.belongsTo(User)

Basket.hasMany(BasketProduct)
BasketProduct.belongsTo(Basket)

Product.hasOne(BasketProduct)
BasketProduct.belongsTo(Product)

TypeUser.hasMany(User)
User.belongsTo(TypeUser)

Product.hasMany(Rating)
Rating.belongsTo(Product)

TypeProduct.hasMany(Product)
Product.belongsTo(TypeProduct)

ImageStore.hasOne(Product)
Product.belongsTo(ImageStore)

StoreProduct.hasMany(Product)
Product.belongsTo(StoreProduct)

TypeProduct.belongsToMany(StoreProduct, {through:
TypeProduct_StoreProduct})
StoreProduct.belongsToMany(TypeProduct, {through:
TypeProduct_StoreProduct})

StoreProduct.hasMany(IndividualOrder)
IndividualOrder.belongsTo(StoreProduct)

module.exports = {
    User,
```

```
    Basket,
    BasketProduct,
    Product,
    Rating,
    StoreProduct,
    TypeProduct,
    ImageStore,
    IndividualOrder,
    TypeUser,
    TypeProduct_StoreProduct
}
```

Листинг А.3 – Middleware проверки авторизации пользователя в файле checkTypeUserMiddleware.js

```
const jwt = require('jsonwebtoken')

module.exports = function (req, res, next) {
    if (req.method === "OPTIONS") {
        next()
    }
    try {
        const token = req.headers.authorization.split(' ')[1]
        if (!token) {
            return res.status(401).json({message: "Не авторизован"})
        }
        const decoded = jwt.verify(token, process.env.SECRET_KEY)
        req.user = decoded
        next()
    } catch (e) {
        res.status(401).json({message: "Не авторизован"})
    }
}
```

Листинг А.4 – Middleware проверки роли пользователя в файле authMiddleware.js

```
const jwt = require('jsonwebtoken')

module.exports = function(typeUserId) {
    return function (req, res, next) {
        if (req.method === "OPTIONS") {
            next()
        }
        try {
            const token = req.headers.authorization.split(' ')[1]
            if (!token) {
                return res.status(401).json({message: "Не
авторизован"})
            }
            const decoded = jwt.verify(token, process.env.SECRET_KEY)
            if (decoded.typeUserId !== typeUserId) {
                return res.status(403).json({message: "Нет доступа"})
            }
            req.user = decoded;
```

```
            next()
        } catch (e) {
            res.status(401).json({message: "Не авторизован"})
        }
    };
}
```

Листинг А.5 – Middleware проверки критической ошибки в файле ErrorHandlingMiddelewere.js

```
const APIError = require('../errors/APIError')

module.exports = function (err, req, res, next) {
    if (err instanceof APIError) {
        return res.status(err.status).json({message: err.message})
    }
    return res.status(500).json({message: 'Ошибка со стороны сервера,
если вы сделали все правильно, то обратитесь в поддержку.'})
}
```

Листинг А.6 – Класс ошибок в файле APIError.js

```
class APIError extends Error{
    constructor(status, message) {
        super()
        this.status = status
        this.message = message
    }

    static badRequest(message){
        return new APIError(400, message)
    }
    static unauthorized(message){
        return new APIError(401, message)
    }
    static paymentRequired(message){
        return new APIError(402, message)
    }
    static forbidden(message){
        return new APIError(403, message)
    }
    static notFound(message){
        return new APIError(404, message)
    }
    static conflict(message){
        return new APIError(409, message)
    }
    static internalServerError(message){
        return new APIError(500, message)
    }
    static serviceUnavailable(message){
        return new APIError(503, message)
    }
}

module.exports = APIError
```

Листинг А.7 – Инициализация роутеров (маршрутов) в файле index.js

```
const Router = require('express')
const router = new Router()
const userRouters = require('./userRouters')
const storeProductRouters = require('./storeProductRouters')
const productRouters = require('./productRouters')
const typeRouters = require('./typeRouters')
const individualOrderRouters = require('./individualOrderRouters')

router.use('/user', userRouters)
router.use('/typeProduct', typeRouters)
router.use('/storeProduct', storeProductRouters)
router.use('/product', productRouters)
router.use('/individualOrder', individualOrderRouters)

module.exports = router
```

Листинг А.8 – Роутер индивидуального заказа в файле IndividualOrderRouters.js

```
const Router = require('express')
const router = new Router()
const individualOrderController =
require('../controllers/individualOrderController')

router.post('/',individualOrderController.create)

module.exports = router
```

Листинг А.9 – Роутер товаров в файле productRouters.js

```
const Router = require('express')
const router = new Router()
const productController = require('../controllers/productController')
const checkTypeUser = require("../midlewere/checkTypeUserMiddleware");

router.post('/', checkTypeUser(4),productController.create)
router.get('/', productController.getAll)
router.get('/:id', productController.getOne)
router.put('/', checkTypeUser(4), productController.update)
router.delete('/:id', checkTypeUser(4),
productController.deleteProduct)

module.exports = router
```

Листинг А.10 – Роутер продавцов в файле storeProductRouters.js

```
const Router = require('express')
const router = new Router()
const storeProductController = require
('../controllers/storeProductController')
const checkTypeUser = require("../midlewere/checkTypeUserMiddleware");
```

```
router.post('/',checkTypeUser(4), storeProductController.create)
router.get('/', storeProductController.getAll)
router.get('/:id', storeProductController.getOne)
router.put('/', checkTypeUser(4), storeProductController.update)
router.delete('/:id', checkTypeUser(4),
storeProductController.deleteStore)

module.exports = router
```

Листинг А.11 – Роутер типов товара в файле typeRouters.js

```
const Router = require('express')
const router = new Router()
const typeProductController = require
('../controllers/typeController')
const checkTypeUser = require('../midlewere/checkTypeUserMiddleware')

router.post('/', checkTypeUser(4), typeProductController.create)
router.get('/', typeProductController.getAll)
router.get('/:id', storeProductController.getOne)
router.put('/', checkTypeUser(4), typeProductController.update)
router.delete('/:id', checkTypeUser(4),
typeProductController.deleteType)

module.exports = router
```

Листинг А.11 – Роутер пользователей в файле userRouters.js

```
const Router = require('express')
const router = new Router()
const userController = require('../controllers/userController')
const authMiddleware = require('../midlewere/authMiddleware')

router.post('/registration', userController.registration)
router.post('/authorization', userController.authorization)
router.get('/ifAuthorization', authMiddleware,
userController.ifAuthorization)
router.get('/me', userController.ifAuthorization)

module.exports = router
```

Листинг А.12 – Контроллер для роутера индивидуального заказа в файле individualOrderController.js

```
const uuid = require('uuid')
const path = require('path')
const {IndividualOrder} = require('../models/models')
const APIError = require('../errors/APIError')

class IndividualOrderProduct {
    async create(req, res, next) {
        try {
            let {name, description, userId, storeId} = req.body
            const {file} = req.files
```

```
            let fileName = uuid.v4() + ".jpg"
            file.mv(path.resolve(__dirname, '..', 'static', fileName))

            const individualOrder = await IndividualOrder.create({
                name,
                description,
                userId,
                storeId,
                file: fileName
            })
            return res.json(individualOrder)
        } catch (e) {
            next(APIError.notFound(e.message))
        }
    }
}

module.exports = new IndividualOrderProduct()
```

Листинг А.13 – Контроллер для роутера товаров в файле productController.js

```
const uuid = require('uuid')
const path = require('path')
const {Product, ImageStore} = require('../models/models')
const APIError = require('../errors/APIError')

class ProductController {
    async create(req, res, next) {
        try {
            let {name, price, shortDescription, description, rating,
typeProductId, storeProductId} = req.body
            const {img} = req.files
            let fileName = uuid.v4() + ".jpg"
            img.mv(path.resolve(__dirname, '..', 'static', fileName))

            const product = await Product.create({
                name,
                price,
                shortDescription,
                description,
                rating,
                typeProductId,
                storeProductId,
                img: fileName
            })
            if(imgS) {
                imgS = JSON.parse(imgS)
                imgS.forEach(i =>
                    ImageStore.create({
                        title: i.title,
                        description: i.description,
                        productID: product.id
                    })
                )
            }
```

```
            return res.json(product)
        } catch (e) {
            next(APIError.notFound(e.message))
        }
    }

    async getAll(req, res, next) {
        try {
                    let {typeProductId, storeProductId, limit, page} =
req.query
            page = page || 1
            limit = limit || 9
            let offset = page * limit - limit
            let products;

            if (!storeProductId && !typeProductId) {
                products = await Product.findAndCountAll({limit,
offset: offset})
            }
            if (storeProductId && !typeProductId) {
                products = await Product.findAndCountAll({where:
{storeProductId}, limit, offset: offset})
            }
            if (!storeProductId && typeProductId) {
                products = await Product.findAndCountAll({where:
{typeProductId}, limit, offset: offset})

            }
            if (storeProductId && typeProductId) {
                products = await Product.findAndCountAll({where:
{storeProductId, typeProductId}, limit, offset: offset})
            }

            return res.json(products)
        }
        catch (e) {
            next(APIError.notFound(e.message))
        }
    }

    async getOne(req, res) {
        const {id} = req.params
        const product = await Product.findOne(
            {
                where: {id},
                include:[{model: ImageStore, as: 'imgS'}]
            },
        )
        return res.json(product)
    }

    async deleteProduct(req, res) {
        const {id} = req.params
        const product = await Product.findOne(
            {
```

```
                where: {id},
                include:[{model: ImageStore, as: 'imgS'}]
            },
        )
        return res.json(product)
    }

    async update(req, res, next) {
        try {
               let {name, price, shortDescription, discription, rating,
typeProductId, storeProductId} = req.body
            const  {img} = req.files
            let fileName = uuid.v4() + ".jpg"
            img.mv(path.resolve(__dirname, '..', 'static', fileName))
            const product = req.body

            if (!product.id) next(APIError.badRequest(e.message))
            const updateProduct = await Product.update(product,
{where: {id}})
            return res.json(updateProduct)
        } catch (e) {
            next(APIError.notFound(e.message))
        }
    }

}

module.exports = new ProductController()
```

Листинг А.14 – Контроллер для роутера продавцов в файле storeProductController.js

```
const {TypeProduct, Product} = require('../models/models')
const APIError = require ('../errors/APIError')

class TypeController {
    async create(req, res) {
        const {name} = req.body
        const typeProduct = await TypeProduct.create({name})
        return res.json(typeProduct)
    }

    async getAll(req, res) {
        const typesProduct = await TypeProduct.findAll()
        return res.json(typesProduct)
    }
    async deleteType(req, res) {
        const {id} = req.params
        const typeProduct = await TypeProduct.findOne(
            {
                where: {id},
            },
        )
        return res.json(typeProduct)
```

```
    }

    async update(req, res, next) {
        try {
            const typeProduct = req.body

            if (!product.id) next(APIError.badRequest(e.message))
            const updatetypeProduct = await
TypeProduct.update(product, {where: {id}})
            return res.json(updatetypeProduct)
        } catch (e) {
            next(APIError.notFound(e.message))
        }
    }
}

module.exports = new TypeController()
```

Листинг А.15 – Контроллер для роутера типов в файле typeController.js

```
const {TypeProduct, Product} = require('../models/models')
const APIError = require ('../errors/APIError')

class TypeController {
    async create(req, res) {
        const {name} = req.body
        const typeProduct = await TypeProduct.create({name})
        return res.json(typeProduct)
    }

    async getAll(req, res) {
        const typesProduct = await TypeProduct.findAll()
        return res.json(typesProduct)
    }

    async deleteType(req, res) {
        const {id} = req.params
        const typesProduct = await TypesProduct.findOne(
            {
                where: {id},
            },
        )
        return res.json(typesProduct)
    }

    async update(req, res, next) {
        try {
            const typesProduct = req.body

            if (!typesProduct.id) next(APIError.badRequest(e.message))
            const updatetypesProduct = await
TypesProduct.update(product, {where: {id}})
            return res.json(updatetypesProduct)
        } catch (e) {
            next(APIError.notFound(e.message))
```

```
        }
    }
}

module.exports = new TypeController()
```

Листинг А.16 – Контроллер для роутера пользователей в файле userController.js

```
const APIError = require('../errors/APIError')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')
const {User, Basket, Product} = require ('../models/models')

const generateJwt = (id, email, typeUserId) => {
    return jwt.sign(
        {id, email, typeUserId},
        process.env.SECRET_KEY,
        {expiresIn: '24h'}
    )
}

class UserController {
    async registration(req, res, next) {
        const {email, password, name, surname, patronymic, typeUserId}
= req.body
        if (!email || !password) {
            return next(APIError.badRequest('Некорректный email или
пароль'))
        }
        const candidate = await User.findOne({where: {email}})
        if (candidate) {
            return next(APIError.badRequest('Пользователь с таким
email уже существует'))
        }
        const hashPassword = await bcrypt.hash(password, 5)
        const user = await User.create({email, name, surname,
patronymic, typeUserId, password: hashPassword})
        const basket = await Basket.create({userId: user.id})
        const token = generateJwt(user.id, user.email,
user.typeUserId)
        return res.json({token})
    }

    async authorization(req, res, next) {
        const {email, password} = req.body

        const user = await User.findOne({where: {email}})
        if (!user) {
            return next(APIError.badRequest('Пользователь не найден'))
        }
        let comparePassword = bcrypt.compareSync(password,
user.password)
        if (!comparePassword) {
```

```
            return next(APIError.badRequest('Указан некорректный email
или пароль'))
        }
        const token = generateJwt(user.id, user.email,
user.typeUserId)
        return res.json({token})
    }

    async ifAuthorization(req, res) {
        const token = generateJwt(req.user.id, req.user.email,
req.user.typeUserId)
        return res.json({token})
    }

    async getOne(req, res) {
        const {email} = req.params
        const product = await Product.findOne(
            {
                where: {email},
            },
        )
        return res.json(product)
    }
}

module.exports = new UserController()
```

ПРИЛОЖЕНИЕ Б
Клиентская часть разработанного интернет-магазина

Листинг Б.1 – Инициализация приложение в файле App.jsx

```
import React, {useContext, useEffect, useState} from "react";
import {BrowserRouter} from "react-router-dom";
import {AppRouter} from "./components/AppRoyter";
import NavBar from "./components/NavBar";
import {observer} from "mobx-react-lite";
import {Context} from "./index";
import {check} from "./http/userAPI";
import {Loader} from "./components/Loader";
import FooterBar from "./components/FooterBar";

const App = observer(() => {
    const {userStore} = useContext(Context)
    const [loading, setLoading] = useState(true)

    useEffect(() => {
        check().then(data => {
            userStore.setUser(true)
        }).finally(() => setLoading(false))
    }, [])

    if (loading) {
        return <Loader/>
    }

    return (
        <BrowserRouter>
            <div className="content">
                <NavBar/>
                <AppRouter/>
            </div>

            <div className="footer">
                <FooterBar/>
            </div>
        </BrowserRouter>
    );
});
export default App;
```

Листинг Б.2 – Запуск приложение в файле index.jsx

```
import React, {createContext} from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import UserStore from "./store/UserStore";
import ProductStore from "./store/ProductStore";
import "./styles.css";

const root = ReactDOM.createRoot(document.getElementById("root"));
export const Context = createContext(null)

root.render(
```

```
    <Context.Provider value={{
        userStore: new UserStore(),
        productStore: new ProductStore(),
    }}>
        <App />
    </Context.Provider>,

);
```

Листинг Б.3 – Роутеры клиентской части в файле routes.js

```
import SellerPage from "./pages/SellerPage";
import Profile from "./pages/Profile";
import Shop from "./pages/Shop";
import Authorization from "./pages/Authorization";
import Basket from "./pages/Basket";
import ProductPage from "./pages/ProductPage";
import SellerPanel from "./pages/SellerPanel";
import {
    BASKET_ROUTE,
    LOGIN_ROUTE,
    PROFILE_ROUTE,
    REGISTRATION_ROUTE,
    SELLER_PAGE_ROUTE, SELLER_PANEL_ROUTE,
    SHOP_ROUTE,
    STATISTICS_ROUTE,
    PRODUCT_ROUTE,
    ORDER_ROUTE
} from "./urls/consts";

export const authSellerRoutes = [
    {
        path: STATISTICS_ROUTE,
        Component: <SellerPanel/>
    },
    {
        path: SELLER_PANEL_ROUTE,
        Component: <SellerPanel/>
    },
    {
        path: PROFILE_ROUTE,
        Component: <Profile/>
    },
]

export const authBuyerRoutes = [
        {
        path: PROFILE_ROUTE,
        Component: <Profile/>
    },
]

export const publicRoutes = [
    {
        path: SHOP_ROUTE,
```

```
        Component: <Shop/>
    },
    {
        path: LOGIN_ROUTE,
        Component: <Authorization/>
    },
    {
        path: REGISTRATION_ROUTE,
        Component: <Authorization/>
    },
    {
        path: BASKET_ROUTE,
        Component: <Basket/>
    },
    {
        path: PRODUCT_ROUTE + '/:id',
        Component: <ProductPage/>
    },
    {
        path: SELLER_PAGE_ROUTE + '/:id',
        Component: <SellerPage/>
    },
    {
        path: ORDER_ROUTE,
        Component: <Basket/>
    },
]
```

Листинг Б.4 – Пользовательские стили в файле styles.css

```
html, body {
    height: 100%;
}

#root {
    height: 100%;
    display: flex;
    /*height: 100vh;*/
    flex-direction: column;
}

.content {
    flex: 1 0 auto;
}

.footer {
    flex-shrink: 0;
}

.auth {
    height: 100%;
}

.reg {
    padding: 2rem 0 3rem;
}
```

```
.logo {
    color: white;
    text-decoration: none;
}

.logo:hover {
    color: #1ddc83;
}

.nav-item {
    margin-left: 1rem;
}

.nav-item:first-child {
    margin-left: 0;
}

.center {
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
}

.main {
    margin: 2rem 0 3rem;
}

.basket-img {
    width: 3rem;
}

.cursor {
    cursor: pointer;
}

.basket-total {
    margin: 1rem 0;
}

.seller-btn {
    display: flex;
    justify-content: space-evenly;
}
```

Листинг Б.5 – Хранилище пользовательской информации в файле UserStore.js

```
import {makeAutoObservable} from "mobx";

export default class UserStore {
    constructor() {
        this._isAuthSeller = false
        this._isAuthBuyer = false
        this._user = {}
        makeAutoObservable(this)
    }
```

```
    setIsAuthSeller(bool) {
        this._isAuthSeller = bool
    }
    setIsAuthBuyer(bool) {
        this._isAuthBuyer = bool
    }
    setUser(user) {
        this._user = user
    }
    get isAuthSeller() {
        return this._isAuthSeller
    }
    get isAuthBuyer() {
        return this._isAuthBuyer
    }
    get user() {
        return this._user
    }
}
```

Листинг Б.6 – Хранилище информации и все что связано с товарами в файле ProductStore.js

```
import {makeAutoObservable} from "mobx";

export default class ProductStore {
    constructor() {
        this._typeProducts = []
        this._storeProducts = []
        this._cart = []
        this._products = []
        this._product = {}

        this._ratings = []
        this._selectedTypeProduct = {}
        this._selectedStoreProduct = {}
        this._page = 1
        this._totalCount = 0
        this._limit = 8

        makeAutoObservable(this)
    }
    setTypeProducts(typeProducts) {
        this._typeProducts = typeProducts
    }
    setStoreProducts(storeProducts) {
        this._storeProducts = storeProducts
    }
    setProducts(products) {
        this._products = products
    }
    addItemInCart(product) {
        this._cart = [...this._cart, product]
    }
    removeItemInCard(productId) {
```

```
        this._products = this._cart.filter(p => p.id === productId)
    }

    setProduct(product) {
        this._product = product
    }

    setRatings(ratings) {
        this._ratings = ratings
    }

    setSelectedTypeProduct(typeProduct) {
        this._selectedTypeProduct = typeProduct
    }
    setSelectedStoreProduct(storeProduct) {
        this._selectedStoreProduct = storeProduct
    }
    setPage(page) {
        this._page = page
    }
    setTotalCount(count) {
        this._totalCount = count
    }

    get typeProducts() {
        return this._typeProducts
    }
    get storeProducts() {
        return this._storeProducts
    }
    get products() {
        return this._products
    }
    get product() {
        return this._product
    }
    get cart() {
        return this._cart
    }
    get ratings() {
        return this._ratings
    }

    get selectedTypeProduct() {
        return this._selectedTypeProduct
    }
    get selectedStoreProduct() {
        return this._selectedStoreProduct
    }
    get totalCount() {
        return this._totalCount
    }
    get page() {
        return this._page
    }
```

```
    get limit() {
        return this._limit
    }
}
```

Листинг Б.7 – Страница авторизация и регистрация в файле Authorization.jsx

```
import React, {useContext, useState} from "react";
import {Button, Card, Col, Container, Form, InputGroup, Row} from
"react-bootstrap";
import {NavLink, useLocation, useNavigate} from "react-router-dom";
import {LOGIN_ROUTE, REGISTRATION_ROUTE, SHOP_ROUTE} from
"../urls/consts";
import {login, registration} from "../http/userAPI";
import {observer} from "mobx-react-lite";
import {Context} from "../index";

const Authorization = observer(() => {
    const {userStore} = useContext(Context)
    const navigate = useNavigate()
    const location = useLocation()
    const isLogin = location.pathname === LOGIN_ROUTE
    const[email,setEmail] = useState("");
    const[password,setPassword] = useState("");
    const[name,setName] = useState("");
    const[surname,setSurname] = useState("");
    const[patronymic,setPatronymic] = useState("");
    const[isSeller,setIsSeller] = useState(false);
    const[isError, setIsError] = useState(false);
    const[isMiddleName,setIsMiddleName] = useState(false);

    const handleLoginOrReg = async () => {
        try {
            let typeUserId = 10;
            if (isSeller) typeUserId = 4;

            let data;
            if (isLogin) {
                data = await login(email, password);
            } else {
                data = await registration(email, password, name,
surname, patronymic, typeUserId)
            }
            debugger
            // userStore.setUser(user)

            if (isSeller || data.typeUserId === 4) {
                userStore.setIsAuthSeller(true)
                userStore.setIsAuthBuyer(true)
            }
            else {
                userStore.setIsAuthSeller(false)
                userStore.setIsAuthBuyer(true)
            }
            navigate(SHOP_ROUTE)
        }
```

```
        catch (e) {
            if (e?.response?.data?.message) {
                alert(e.response.data.message)
            } else {
                alert("Произошла ошибка")
            }
        }
    }

    const handleSubmit = (event) => {
        event.preventDefault();
        const form = event.currentTarget;
        if (form.checkValidity() === false) {
            setIsError(true);
        } else {
            setIsError(false);
            handleLoginOrReg();
        }
    };
    return (
        <Container
            className={`d-flex justify-content-center align-items-
center ${isLogin ? "auth" : "reg"}`}
        >
            <Card style={{width: 600}} className="p-3">
                <h2 className="m-auto">{isLogin ? 'Авторизация' :
"Регистрация"}</h2>
                <Form noValidate validated={isError}
onSubmit={handleSubmit} className="d-flex flex-column">
                    <Form.Group as={Col} md="12"
controlId="validationCustom01">
                        <Form.Label className="mt-
1">@Email:</Form.Label>
                        <Form.Control
                            required
                            type="email"
                            placeholder="Введите Ваш email..."
                            pattern="^([a-zA-Z0-9_.+-])+@(([a-zA-Z0-9-
])+\.)+([a-zA-Z]{2,4})+$"
                            value={email}
                            onChange={e => setEmail(e.target.value)}
                        />
                        <Form.Control.Feedback>Email
введен!</Form.Control.Feedback>
                        <Form.Control.Feedback
type="invalid">Пожалуйста, введите email.</Form.Control.Feedback>
                    </Form.Group>
                    <Form.Group as={Col} md="12"
controlId="validationCustom02">
                        <Form.Label className="mt-
1">Пароль:</Form.Label>
                        <InputGroup hasValidation>
                            <Form.Control
                                placeholder="Введите Ваш пароль..."
                                required
```

```
                                value={password}
                                pattern="(?=.*\d)(?=.*[@$!%*#?&-
+=])(?=.*[a-z])(?=.*[A-Z]).{8,}"
                                onChange={e =>
setPassword(e.target.value)}
                                type="password"
                                style={{width: "100%"}}
                            />
                            <Form.Control.Feedback>Пароль
введен!</Form.Control.Feedback>
                            <Form.Control.Feedback
type="invalid">Пожалуйста, введите пароль.</Form.Control.Feedback>
                            {!isLogin ?
                            <Form.Text className="text-muted">
                                Пароль должен:
                                <li>содержать больше 7 символов;</li>
                                <li>содержать специальный символ
(@$!%*#?&-+=);</li>
                                <li>содержать хотябы цифру, символ
верхнего и нижнего регистров;</li>
                            </Form.Text>
                                : null}
                        </InputGroup>
                    </Form.Group>
                    {isLogin ? null : <div>
                        <Form.Check
                            className="mt-1"
                            type="switch"
                            checked={isSeller}
                            onChange={e =>
setIsSeller(e.target.checked)}
                            label="Я покупатель/Я продавец"
                        />
                    <Form.Group as={Col} md="12"
controlId="validationCustom03">
                        <Form.Label className="mt-
1">Фамилия:</Form.Label>
                        <Form.Control
                            type="text"
                            placeholder="Введите Вашу фамилию..."
                            required
                            pattern="[А-Я]{1}[а-я]{1,40}|[а-я]{2,40}"
                            value={surname}
                            onChange={e => setSurname(e.target.value)}
                        />
                        <Form.Control.Feedback>Фамилия
введена!</Form.Control.Feedback>
                        <Form.Control.Feedback
type="invalid">Пожалуйста, введите фамилию.</Form.Control.Feedback>
                    </Form.Group>
                    <Form.Group as={Col} md="12"
controlId="validationCustom04">
                        <Form.Label className="mt-1">Имя:</Form.Label>
                        <Form.Control
                            type="text"
```

```
                            placeholder="Введите Ваше имя..."
                            required
                            pattern="[А-Я]{1}[а-я]{1,40}|[а-я]{2,40}"
                            value={name}
                            onChange={e => setName(e.target.value)}
                        />
                        <Form.Control.Feedback>Имя
введено!</Form.Control.Feedback>
                        <Form.Control.Feedback
type="invalid">Пожалуйста, введите имя.</Form.Control.Feedback>
                    </Form.Group>
                        <Form.Check
                            className="mt-1"
                            type="switch"
                            checked={isMiddleName}
                            onChange={e =>
setIsMiddleName(e.target.checked)}
                            label="У меня есть отчество/У меня нет
отчества"
                        />
                        {!isMiddleName &&
                    <Form.Group as={Col} md="12"
controlId="validationCustom05">
                        <Form.Label className="mt-
1">Отчество:</Form.Label>
                        <Form.Control
                            type="text"
                            placeholder="Введите Ваше отчество..."
                            required
                            value={patronymic}
                            pattern="[А-Я]{1}[а-я]{1,40}|[а-я]{2,40}"
                            onChange={e =>
setPatronymic(e.target.value)}/>
                        <Form.Control.Feedback>Отчество
введено!</Form.Control.Feedback>
                        <Form.Control.Feedback
type="invalid">Пожалуйста, введите отчество.</Form.Control.Feedback>
                    </Form.Group>
                        }
                    <Form.Group className="mb-3 mt-1">
                        <Form.Check
                            required
                            pattern="[А-Я]{1}[а-я]{2,40}|[а-я]{3,40}"
                            label="Все данные, которые я ввел, мои"
                            feedback="Вы должны подтвердить, что это
ваши данные!"
                            feedbackType="invalid"
                        />
                    </Form.Group>
                    </div>}
                    <Row className="d-flex justify-content-between mt-
1 pl-3 pr-3">
                        {isLogin ?
                            <div>
                                Нет аккаунта? Тогда жми: <NavLink
```

```
to={REGISTRATION_ROUTE}>Зарегистрироваться</NavLink>
                                </div>
                                :
                                <div>
                                    Есть аккаунт? Тогда жми: <NavLink
to={LOGIN_ROUTE}>Авторизироваться</NavLink>
                                </div>
                            }
                            <Button className="mt-1" type="submit"
variant={"outline-success"}>
                                {isLogin ? 'Войти' : 'Регистрация'}
                            </Button>
                        </Row>
                    </Form>
                </Card>
            </Container>
        );
});

export default Authorization;
```

Листинг Б.8 – Страница корзины в файле Basket.jsx

```
import React, {useContext} from "react";
import {Button, Col, Container, Row, Table} from "react-bootstrap";
import {observer} from "mobx-react-lite";
import {Context} from "../index";
import {login} from "../http/userAPI";

const temp = [1, 2, 3, 6, 8];

const Basket = observer(() => {
    const {productStore} = useContext(Context)
    if (temp.length < 1) {
        return <div className="center">
            Корзина пуста!
        </div>
    }

    function sum (x){
        var s = 0;
        for (let i = 0; i < x.length; i++){
            s += x[i]
        }
        return s
    }

    const arrPrice = productStore.cart.map(item=> item.price)
    console.log(sum(arrPrice))
    console.log('flgkfgkjrjk')
    productStore.cart.map(cart => console.log(cart))
    return (
        <div className="main">
            <Container>
                <Table responsive>
                    <thead>
```

```
                    <tr>
                        <th>#</th>
                        <th>Товар</th>
                        <th>Изделие</th>
                        <th>Цена</th>
                        <th>Количество</th>
                        <th>Скидка</th>
                        <th>Итого</th>
                        <th/>
                    </tr>
                    </thead>

                    <tbody>
                    {productStore.cart.map((item, index) => {
                        console.log(item)
                        console.log(item.img)
                        console.log(item.id)
                        return <tr key={item.id}>
                            <td>{index + 1}</td>
                            <td>
                                <img src={"http://localhost:4060/" +
item.img}
                                     alt="Картинка товара"
                                     className="basket-img"
                                />
                            </td>
                            <td>{item.name}</td>
                            <td>{item.price}</td>
                            <td>1</td>
                            <td> -</td>
                            <td>{item.price}</td>
                            <td className="cursor"
                                onClick={() => console.log("Удаление
товара")}>
                                &#10006;
                            </td>
                        </tr>
                    })}
                    </tbody>
                </Table>

                {productStore.cart.length > 0 && <Row
className="basket-total">
                    <Col xs={12} md={{span: 4, offset: 8}}>
                        <b>Итоговая сумма заказа</b>
                        <div>Цена: {sum(arrPrice)} ₽</div>
                        <div>Количество: {productStore.cart.length}
шт.</div>
                        <div>Продавцы: Продавец 1 Продавец 2</div>
                        <Button
                            variant={"outline-success"}
                            className="mt-2"
                            onClick={() => console.log("Оформить")}
                        >
                            Оформить
```

```
                        </Button>
                    </Col>
                </Row>}
            </Container>
        </div>
    );
});

export default Basket;
```

Листинг Б.9 – Страница товара в файле ProductPage.jsx

```
import React, {useContext, useEffect, useState} from 'react';
import {Button, Card, Col, Container, Form, Image, Row} from "react-
bootstrap";
import star from '../assets/img/army.png'
import RatingList from "../components/RatingList";
import {useParams} from 'react-router-dom'
import {fetchOneProduct} from "../http/productAPI";
import {observer} from "mobx-react-lite";
import {Context} from "../index";
import CreateStoreProduct from
"../components/modals/CreateStoreProduct";
import IndividualOrderProduct from
"../components/modals/IndividualOrderProduct";

const ProductPage = observer(() => {
    const [orderVisible, setOrderVisible] = useState(false)
    const {id} = useParams()
    const {productStore} = useContext(Context)

    useEffect(() => {
        fetchOneProduct(id).then(data => {
            console.log(data)
            productStore.setProduct(data)
        })
    }, [])

    const handleAddInCart = () => {
        productStore.addItemInCart(productStore.product)
    }

    // productStore.products.some(p=> p.id === productStore.id)
// checks whether an element is even
    const arr = productStore.cart.map(item => productStore.product.id
=== item.id)
    const even = (element) => (element === true);
    const isExistsInCart = arr.some(even)
    return (
        <Container className="mt-3">
            {id !== productStore.product.id && <></>}

            {productStore.product && <>
                <Row>
                    <Col xs={8} md={6}>
```

```
                        <Image width={300} height={300}
src={'http://localhost:4060/' + productStore.product.img}/>
                        <h3 className={"mt-3"}>Продавец: Продавец
1</h3>
                    </Col>
                    <Col md={6}>
                        <Row>
                            <h2>{productStore.product.name}</h2>
                            <Col>
                                <h4>Цена от:
{productStore.product.price} руб.</h4>
                            </Col>
                            <Col>
                                <h5>Рейтинг:
{productStore.product.rating} <Image width={25} height={25}
src={star}/>
                                </h5>
                            </Col>
                            <Button disabled={isExistsInCart}
className={"mt-3"} variant={"outline-success"}
                                    onClick={handleAddInCart}>
                                {isExistsInCart ? "Товар уже в
корзине" : "Добавить в корзину"}
                            </Button>
                        </Row>
                        <Card className={"mt-3 d-flex"}>
                            <Card.Header>Краткое описание
товара:</Card.Header>
                            <Card.Body>
                                <Card.Text>

{productStore.product.shortDescription}
                                </Card.Text>
                            </Card.Body>
                        </Card>
                        <Card className={"mt-3 d-flex"}>
                            <Card.Header>Описание
товара:</Card.Header>
                            <Card.Body>
                                <Card.Text>
                                    {productStore.product.discription}
                                </Card.Text>
                            </Card.Body>

                        </Card>
                        <Card className={"mt-3"}>
                        <Button
                            xs={8}
                            md={6}
                            variant="outline-info"
                            className="text-center p-1 d-grid  "
                            onClick={() => setOrderVisible(true)}
                        >
                            Оформить индивидуальный заказ
                        </Button>
```

```
                        </Card>
                        <IndividualOrderProduct show={orderVisible}
onHide={() => setOrderVisible(false)}/>
                    </Col>
                </Row>

                <h3 className={"mt-3"}>Отзывы:</h3>
                <Card className={"mt-3, d-flex"}>
                    <Card.Header>Ваш коментарий:</Card.Header>
                    <Card.Body>
                        <Card.Text>
                            <Form>
                                {['radio'].map((type) => (
                                    <div key={`inline-${type}`}
className="mb-3">
                                        <Form.Check
                                            inline
                                            label="1"
                                            name="group1"
                                            type={type}
                                            id={`inline-${type}-1`}
                                        />
                                        <Form.Check
                                            inline
                                            label="2"
                                            name="group1"
                                            type={type}
                                            id={`inline-${type}-2`}
                                        />
                                        <Form.Check
                                            inline
                                            label="3"
                                            name="group1"
                                            type={type}
                                            id={`inline-${type}-3`}
                                        />
                                        <Form.Check
                                            inline
                                            label="4"
                                            name="group1"
                                            type={type}
                                            id={`inline-${type}-4`}
                                        />
                                        <Form.Check
                                            inline
                                            label="5"
                                            name="group1"
                                            type={type}
                                            id={`inline-${type}-5`}
                                        />
                                    </div>
                                ))}
                                <Form.Group className="mb-3"
controlId="exampleForm.ControlTextarea1">
                                    <Form.Control as="textarea"
```

```
rows={3}/>
                                    </Form.Group>
                                    <Button variant="primary"
type="submit">
                                        Отправить
                                    </Button>
                                </Form>
                            </Card.Text>
                        </Card.Body>
                        <RatingList/>
                    </Card>
                </>}
            </Container>
        );
});

export default ProductPage;
```

Листинг Б.10 – Страница профиля в файле Profile.jsx

```
import React, {useContext} from 'react';
import {Col, Container, Row} from "react-bootstrap";
import {Context} from "../index";

const Profile = () => {
    const {userStore} = useContext(Context)
    return (
        <Container className="d-flex flex-column">
            <Row className="main">
                <b>Данные профиля</b>
                <Col xs={12} md={6}>
                    <div className="mt-2">
                        <b><i>ФИО:</i></b> {userStore.user.surname}
{userStore.user.name} {userStore.user.patronymic}
                    </div>
                    <div className="mt-2">
                        <b><i>Почта:</i></b> {userStore.user.email}
                    </div>
                </Col>
            </Row>
        </Container>
    );
};

export default Profile;
```

Листинг Б.11 – Страница профиля продавца в файле SellerPanel.jsx

```
import React, {useContext, useState} from 'react';
import {Button, Col, Container, Row} from "react-bootstrap";
import CreateProduct from "../components/modals/CreateProduct";
import CreateStoreProduct from
"../components/modals/CreateStoreProduct";
import CreateTypeProduct from
"../components/modals/CreateTypeProduct";
import {observer} from "mobx-react-lite";
import {Context} from "../index";
```

```
const SellerPanel = observer(() => {
    const {userStore} = useContext(Context)
    const {productStore} = useContext(Context)

    const [typeProductVisible, setTypeProductVisible] =
useState(false)
    const [statisticsVisible, setStatisticsVisible] = useState(false)
    const [productVisible, setProductVisible] = useState(false)

    return (
        <Container className="d-flex flex-column">
            <Row>
                <Col xs={12} className="seller-btn">
                    {userStore.isAuthSeller === true && <>
                        <Button
                            variant="outline-dark"
                            className="mt-4 p-2"
                            onClick={() => setStatisticsVisible(true)}
                        >
                            Статистика
                        </Button>

                        <Button
                            variant="outline-dark"
                            className="mt-4 p-2"
                            onClick={() =>
setTypeProductVisible(true)}
                        >
                            Добавить тип товара
                        </Button>

                        <Button
                            variant="outline-dark"
                            className="mt-4 p-2"
                            onClick={() => setProductVisible(true)}
                        >
                            Добавить продукт
                        </Button>

                        <CreateStoreProduct show={typeProductVisible}
onHide={() => setTypeProductVisible(false)}/>
                        <CreateProduct show={productVisible}
onHide={() => setProductVisible(false)}/>
                        <CreateTypeProduct show={statisticsVisible}
onHide={() => setStatisticsVisible(false)}/>
                    </>}
                </Col>
            </Row>

            <Row className="main">
                <b>Данные профиля</b>
                <Col xs={12} md={6}>
                    <div className="mt-2">
                        <b><i>ФИО:</i></b> {userStore.user.surname}
{userStore.user.name} {userStore.user.patronymic}
```

```
                    </div>
                    <div className="mt-2">
                        <b><i>Название магазина:</i></b>
{userStore.user.typeUserId === 10 &&
productStore.storeProducts.some((storeProduct, key={id}) =>
                    <>{storeProduct.name}</>)
                    }
                    </div>
                    <div className="mt-2">
                        <b><i>Почта:</i></b> {userStore.user.email}
                    </div>
                    <div className="mt-2">
                        <b><i>Адрес доставки</i></b>:
{userStore.user.typeUserId === 10 &&
productStore.storeProducts.some((storeProduct, key={id}) =>
                        <>{storeProduct.delivery}</>)
                    }
                    </div>
                </Col>
            </Row>
        </Container>
    );
});

export default SellerPanel;
```

Листинг Б.12 – Главная страница в файле Shop.jsx

```
import React, {useContext, useEffect} from 'react';
import {Col, Container, Row} from "react-bootstrap";
import TypeBar from "../components/TypeBar";
import SellerProductBar from "../components/SellerProductBar";
import ProductList from "../components/ProductList";
import {observer} from "mobx-react-lite";
import {Context} from "../index";
import {fetchProducts, fetchStoreProducts, fetchTypes} from
"../http/productAPI";
import Pages from "../components/Pages";

const Shop = observer(() => {
    const {productStore} = useContext(Context)

    useEffect(() => {

        fetchTypes().then(data => productStore.setTypeProducts(data))
        fetchStoreProducts().then(data =>
productStore.setStoreProducts(data))
        fetchProducts(null, null, 1, 8).then(data => {
            productStore.setProducts(data.rows)
            productStore.setTotalCount(data.count)
        })
    }, [])

    useEffect(() => {
        fetchProducts(productStore.selectedTypeProduct.id,
productStore.selectedStoreProduct.id, productStore.page, 8).then(data
```

```
=> {
            productStore.setProducts(data.rows)
            productStore.setTotalCount(data.count)
        })
    }, [productStore.page, productStore.selectedTypeProduct,
productStore.selectedStoreProduct])

    return (
        <Container>
            <Row className="mt-2">
                <Col md={2}>
                    <TypeBar/>
                </Col>

                <Col md={9}>
                    <SellerProductBar/>
                    <ProductList/>
                    <Pages/>
                </Col>
            </Row>
        </Container>
    );
});

export default Shop;
```

Листинг Б.13 – Компонент модального окна создания товара в файле CreateProduct.jsx

```
import React, {useContext, useState} from 'react';
import {Button, Dropdown, Form, Modal} from "react-bootstrap";
import {Context} from "../../index";
import {createProduct} from "../../http/productAPI";

const CreateProduct = ({show, onHide}) => {
    const [value, setValue] = useState('')
    const {productStore} = useContext(Context)

    const addProduct = () => {
        createProduct({name: value}).then(data => {
            setValue('')
            onHide()
        })
    }

    return (
        <Modal
            show={show}
            onHide={onHide}
            centered
        >
            <Modal.Header closeButton>
                <Modal.Title id="contained-modal-title-vcenter">
                    Добавить товар
                </Modal.Title>
```

```
            </Modal.Header>
            <Modal.Body>
                <Form>
                    <Form.Control
                        value={value}
                        onChange={e => setValue(e.target.value)}
                        placeholder={"Введите уникальное название
товара"}
                    />
                    <Dropdown className="mt-2 mb-2">
                        <Dropdown.Toggle>Введите тип</Dropdown.Toggle>

<Dropdown.Menu>{productStore.typeProducts.map(typeProduct =>
                            <Dropdown.Item
key={typeProduct.id}>{typeProduct.name}</Dropdown.Item>
                        )}</Dropdown.Menu>
                    </Dropdown>
                    <Dropdown className="mt-2 mb-2">
                        <Dropdown.Toggle>Введите
продавца</Dropdown.Toggle>

<Dropdown.Menu>{productStore.storeProducts.map(storeProduct =>
                            <Dropdown.Item
key={storeProduct.id}>{storeProduct.name}</Dropdown.Item>
                        )}</Dropdown.Menu>
                    </Dropdown>
                    <Form.Control
                        className="mt-3"
                        placeholder={"Введите краткое описание
товара"}
                    />
                    <Form.Control
                        className="mt-3"
                        placeholder={"Введите подробное описание
товара"}
                    />
                    <Form.Control
                        className="mt-3"
                        placeholder={"Введите стоимость товара"}
                        type="number"
                    />
                    <Form.Control
                        className="mt-3"
                        type="file"
                    />

                </Form>
            </Modal.Body>
            <Modal.Footer>
                <Button variant="outline-danger"
onClick={onHide}>Закрыть</Button>
                <Button variant="outline-success"
onClick={addProduct}>Добавить</Button>
            </Modal.Footer>
        </Modal>
```

```
    );
};

export default CreateProduct;
```

Листинг Б.14 – Компонент модального окна создания типа товара в файле CreateTypeProduct.jsx

```
import React, {useState} from 'react';
import {Button, Form, Modal} from "react-bootstrap";
import {createTypeProduct} from "../../http/productAPI";

const CreateTypeProduct = ({show, onHide}) => {
    const [value, setValue] = useState('')

    const addTypeProduct = () => {
        createTypeProduct({name: value}).then(data => {
            setValue('')
            onHide()
        })
    }

    return (
        <Modal
            show={show}
            onHide={onHide}
            centered
        >
            <Modal.Header closeButton>
                <Modal.Title id="contained-modal-title-vcenter">
                    Добавить тип
                </Modal.Title>
            </Modal.Header>
            <Modal.Body>
                <Form>
                    <Form.Control
                        value={value}
                        onChange={e => setValue(e.target.value)}
                        placeholder={"Введите название типа"}
                    />
                </Form>
            </Modal.Body>
            <Modal.Footer>
                <Button variant="outline-danger"
onClick={onHide}>Закрыть</Button>
                <Button variant="outline-success"
onClick={addTypeProduct}>Добавить</Button>
            </Modal.Footer>
        </Modal>
    );
};

export default CreateTypeProduct;
```

Листинг Б.15 – Компонент роутера клиентского приложения в файле AppRoyter.jsx

```
import {Route, Routes} from "react-router-dom";
import {authBuyerRoutes, authSellerRoutes, publicRoutes} from
'../routes';
import Shop from "../pages/Shop";
import {useContext} from "react";
import {Context} from "../index";
import {observer} from "mobx-react-lite";

const AppRouter = observer(() => {
    const {userStore} = useContext(Context)

    return (
        <div style={{height: "100%"}}>

            <Routes>
                {userStore.isAuthSeller === true &&
userStore.isAuthBuyer === true
                    && authSellerRoutes.map(({path, Component}) =>
                        <Route key={path} path={path}
element={Component}/>
                    )
                }

                {userStore.isAuthSeller === false &&
userStore.isAuthBuyer === true
                    && authBuyerRoutes.map(({path, Component}) =>
                        <Route key={path} path={path}
element={Component}/>
                    )
                }

                {publicRoutes.map(({path, Component}) =>
                    <Route key={path} path={path}
element={Component}/>
                )}

                {/* Defaults url */}
                <Route path="*" element={<Shop/>}/>
            </Routes>
        </div>
    );
});

export {AppRouter};
```

Листинг Б.16 – Компонент футера в файле FooterBar.jsx

```
import React from 'react';
import {Container, Row} from "react-bootstrap";

const FooterBar = () => {
```

```
    return (
        <Container className="mt-3 background-color:#bec1c54d">
            <div className="row">
                <div className="col-md-6 item text">
                    <h3>I-Shop</h3>
                    <p>Интернет-магазин ручных изделий. Создавать и
продавать может каждый!</p>
                </div>
                <div className="col-sm-6 col-md-6 text-center">
                    <Row>
                        <h3>Контакты</h3>
                        <div>
                            <li><a href="mailto:text.text@mail.ru"
className="contact-link">text.text@mail.ru</a></li>
                            <li><a href="tel:99123456789"
className="contact-link">+7 912 (345) 67-89</a></li>
                            <li><a>просп. Ленина, 87</a></li>
                        </div>
                    </Row>

                </div>
                <div className="text-center p-4">
                    © 2022 Copyright: by Vyacheslav KE-406
                </div>
            </div>
        </Container>
    );
};

export default FooterBar;
```

Листинг Б.17 – Компонент навигационной панели в файле NavBar.jsx

```
import React, {useContext} from 'react';
import {Context} from "../index";
import {Button, Container, Nav, Navbar} from "react-bootstrap";
import {BASKET_ROUTE, LOGIN_ROUTE, PROFILE_ROUTE, SELLER_PANEL_ROUTE,
SHOP_ROUTE} from "../urls/consts";
import {observer} from "mobx-react-lite";
import {Link, useNavigate} from "react-router-dom";

const NavBar = observer(() => {
    const {userStore} = useContext(Context)
    const navigate = useNavigate();

    const logOut = () => {
        userStore.setUser({})
        userStore.setIsAuthSeller(false)
        userStore.setIsAuthBuyer(false)
        localStorage.removeItem('token')
    }

    return (
        <Navbar collapseOnSelect expand="lg" bg="dark" variant="dark">
            <Container>
                <Link to={SHOP_ROUTE}
```

```
                    className="logo"
                >
                    I-Shop
                </Link>

                <Navbar.Toggle aria-controls="responsive-navbar-nav"
/>
                <Navbar.Collapse id="responsive-navbar-nav"
className="justify-content-end">
                    <Nav>
                        <Nav.Item>
                           <Nav.Link variant={"outline-light"}
onClick={() => navigate(BASKET_ROUTE)}>Корзина</Nav.Link>
                        </Nav.Item>

                        {userStore.isAuthSeller &&
userStore.isAuthBuyer && <>
                            <Nav.Item>
                                <Nav.Link variant={"outline-light"}
onClick={() => navigate(SELLER_PANEL_ROUTE)}>
                                    Профиль
                                </Nav.Link>
                            </Nav.Item>

                            <Nav.Item>
                                <Nav.Link variant={"outline-light"}
onClick={() => logOut()}>
                                    Выйти
                                </Nav.Link>
                            </Nav.Item>
                        </>}

                        {userStore.isAuthBuyer &&
!userStore.isAuthSeller && <>
                            <Nav.Item>
                                <Nav.Link variant={"outline-light"}
onClick={() => navigate(PROFILE_ROUTE)}>
                                    Профиль
                                </Nav.Link>
                            </Nav.Item>

                            <Nav.Item>
                                <Nav.Link variant={"outline-light"}
onClick={() => logOut()}>
                                    Выйти
                                </Nav.Link>
                            </Nav.Item>
                        </>}

                        {!userStore.isAuthBuyer &&
!userStore.isAuthSeller && <>
                            <Nav.Item>
                                <Nav.Link variant={"outline-light"}
onClick={() => navigate(LOGIN_ROUTE)}>
                                    Войти
```

```
                                    </Nav.Link>
                                </Nav.Item>
                            </>}
                        </Nav>
                    </Navbar.Collapse>
                </Container>
            </Navbar>
    );
});

export default NavBar;
```

Листинг Б.18 – Компонент переключения страниц в файле Pages.jsx

```
import React, {useContext} from 'react';
import {observer} from "mobx-react-lite";
import {Context} from "../index";
import {Pagination} from "react-bootstrap";

const Pages = observer(() => {
    const {productStore} = useContext(Context)
    const pageCount = Math.ceil(productStore.totalCount /
productStore.limit)
    const pages = []

    for (let i=0; i < pageCount; i++) {
        pages.push(i+1)
    }
    return (
        <Pagination className="justify-content-center mt-5">
            {pages.map(page =>
            <Pagination.Item
                key={page}
                active={productStore.page === page}
                onClick={()=> productStore.setPage(page)}
            >{page}</Pagination.Item>
            )}
        </Pagination>
    );
});

export default Pages;
```

Листинг Б.19 – Компонент карточки товара в файле ProductItem.jsx

```
import React from 'react';
import {Card, Col, Image} from "react-bootstrap";
import star from '../assets/img/army.png'
import {PRODUCT_ROUTE} from "../urls/consts";
import { useNavigate } from 'react-router-dom';

const ProductItem = ({product}) => {
    const navigate = useNavigate ()
    return (
        <Col  md={4} lg={3} xs={6} className={"mt-4"} onClick={()
```

```
=>navigate(PRODUCT_ROUTE + '/' + product.id)}>
            <Card style={{width: 150, cursor: 'pointer'}}
border={"light"}>
                <Image width={150} height={150}
src={'http://localhost:4060/' + product.img}/>
                <div className="text-black-50 mt-1 d-flex justify-
content-between align-items-center">
                    <div>Продукт...</div>
                    <div className="d-flex align-items-center">
                        <div>{product.rating}</div>
                    <Image width={20} height={20} src={star}/>
                    </div>
                </div>
                <div>{product.name}</div>
            </Card>
        </Col>
    );
};

export default ProductItem;
```

Листинг Б.20 – Компонент списка карточек товаров в файле ProductList.jsx

```
import React, {useContext} from 'react';
import {observer} from "mobx-react-lite";
import {Context} from "../index";
import ProductItem from "./ProductItem";
import {Row} from "react-bootstrap";

const ProductList = observer(() => {
    const {productStore} = useContext(Context)

    return (
        <Row className="d-flex">
            {productStore.products.map(product =>
                <ProductItem key={product.id} product={product}/>
            )}

            {productStore.products.length < 1 &&
                <div className="m-4">
                    Товаров не найдено!
                </div>
            }
        </Row>
    );
});

export default ProductList;
```

Листинг Б.21 – Компонент листа отзывов в файле RatingList.jsx

```
import React, {useContext} from 'react';
import {Card, Row} from "react-bootstrap";
import {Context} from "../index";
import {observer} from "mobx-react-lite";
```

```
const RatingList = observer(() => {
    const {productStore} = useContext(Context)

    return (
        <>
            {productStore.ratings.map(rating => {
                return <Card
                    className={"mt-3 d-flex"}
                    key={rating.id}
                >
                    <Card.Header>Оценка: {rating.rate}</Card.Header>
                    <Card.Body>
                        <blockquote className="blockquote mb-0">
                            <p>
                                {' '}
                                {rating.review}{' '}
                            </p>
                            <footer className="blockquote-footer">
                                {rating.userId} Время: <cite
title="Source Title">{rating.createdAt}</cite>
                            </footer>
                        </blockquote>
                    </Card.Body>
                </Card>
            })}
        </>
    );
});

export default RatingList;
```

Листинг Б.22 – Компонент панели продавцов в файле SellerProductBar.jsx

```
import React, {useContext} from 'react';
import {observer} from "mobx-react-lite";
import {Context} from "../index";
import {Card, Col, Row} from "react-bootstrap";

const SellerProductBar = observer(() => {
    const {productStore} = useContext(Context)

    return (
        <Row className="d-flex">
            <Col md={12} className="d-flex">
                {productStore.storeProducts.map(storeProduct =>
                    <Card
                        action variant="light"
                        style={{cursor: 'pointer'}}
                        key={storeProduct.id}
                        className="p-3"
                        onClick={() => {storeProduct.id ===
productStore.selectedStoreProduct.id
                            ? productStore.setSelectedStoreProduct({})
                            :
productStore.setSelectedStoreProduct(storeProduct)
```

```
                            }
                        }
                        size="sm"
                        border={storeProduct.id ===
productStore.selectedStoreProduct.id ? 'success' : 'light'}
                    >
                        {storeProduct.name}
                    </Card>
                )}
            </Col>
        </Row>
    );
});

export default SellerProductBar;
```

Листинг Б.23 – Компонент панели типов товаров в файле TypeBar.jsx

```
import React, {useContext} from 'react';
import {observer} from "mobx-react-lite";
import {Context} from "../index";
import {ListGroup} from "react-bootstrap";

const TypeBar = observer( () => {
    const {productStore} = useContext(Context)

    return (
        <ListGroup variant="flush">
            {productStore.typeProducts.map(typeProduct =>
                <ListGroup.Item
                    key={typeProduct.id}
                    action variant="light"
                    style={{cursor: 'pointer'}}
                    active={typeProduct.id ===
productStore.selectedTypeProduct.id}

                    onClick={() => {typeProduct.id ===
productStore.selectedTypeProduct.id
                        ? productStore.setSelectedTypeProduct({})
                        :
productStore.setSelectedTypeProduct(typeProduct)
                    }
                }
                >
                    {typeProduct.name}
                </ListGroup.Item>

            )}
        </ListGroup>
    );
});

export default TypeBar;
```

Листинг Б.24 – Инициализация связи взаимодействия серверной части с клиентской в файле idex.js

```
import axios from "axios";

const $host = axios.create({
    baseURL: process.env.REACT_APP_API_URL
})

const $authHost = axios.create({
    baseURL: process.env.REACT_APP_API_URL
})

const authInterceptor = config => {
    config.headers.authorization = `Bearer
${localStorage.getItem('token')}`
    return config
}

$authHost.interceptors.request.use(authInterceptor)

export {
    $host,
    $authHost
}
```

Листинг Б.25 – API товаров в файле productAPI.js

```
import {$authHost, $host} from "./index";

export const createTypeProduct = async (type) => {
    const {data} = await $authHost.post('api/typeProduct', type)
    return data
}

export const fetchTypes = async () => {
    const {data} = await $host.get('api/typeProduct')
    return data
}

export const createStoreProduct = async (storeProduct) => {
    const {data} = await $authHost.post('api/storeProduct',
storeProduct)
    return data
}

export const fetchStoreProducts = async () => {
    const {data} = await $host.get('api/storeProduct', )
    return data
}

export const createProduct = async (product) => {
    const {data} = await $authHost.post('api/product', product)
    return data
}
```

```
export const fetchProducts = async (typeProductId, storeProductId,
page, limit= 8) => {
    const {data} = await $host.get('api/product', {params: {
            typeProductId, storeProductId, page, limit
        }})
    return data
}

export const fetchOneProduct = async (id) => {
    const {data} = await $host.get('api/product/' + id)
    return data
}

export const createIndividualOrder = async (individualOreder) => {
    const {data} = await $host.post('api/individualOrder',
individualOreder)
    return data
}
```

Листинг Б.26 – API потльзователей в файле userAPI.js

```
import {$authHost, $host} from "./index";
import jwt_decode from "jwt-decode";

export const registration = async (email, password, name, surname,
patronymic, typeUserId) => {
    const {data} = await $host.post('api/user/registration', {email,
password, name, surname, patronymic, typeUserId})
    localStorage.setItem('token', data.token)
    return jwt_decode(data.token)
}

export const login = async (email, password) => {
    const {data} = await $host.post('api/user/authorization', {email,
password})
    localStorage.setItem('token', data.token)
    return jwt_decode(data.token)
}

export const check = async () => {
    const {data} = await $authHost.get('api/user/ifAuthorization' )
    localStorage.setItem('token', data.token)
    return jwt_decode(data.token)
}

export const myProfile = async () => {
    const {data} = await $authHost.get('api/user/me' )
    localStorage.setItem('token', data.token)
    return jwt_decode(data.token)
}
```