МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

Специализация 1-40 01 01 10 «Программное обеспечение информационных технологий (программирование интернет-приложений)»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

Web-приложение «Магазин компьютерных комплектующих»

Выполнил студент Гришин Иван Сергеевич

(Ф.И.О.)

Руководитель проекта асс. Нистюк О. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты асс. Нистюк О. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер асс. Нистюк О. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2022

Содержание

[Введение 3](#_Toc104303390)

[1 Постановка задачи 4](#_Toc104303391)

[2 Обзор аналогов 5](#_Toc104303392)

[3 Проектирование приложения 7](#_Toc104303393)

[4 Разработка приложения 10](#_Toc104303394)

[4.1 Разработка серверной части 10](#_Toc104303395)

[4.2 Разработка клиентской части 13](#_Toc104303396)

[4.3 Реализация основного функционала приложения 15](#_Toc104303397)

[4.4 Реализация Web-сокета 15](#_Toc104303398)

[5 Тестирование 16](#_Toc104303399)

[6 Руководство пользователя 20](#_Toc104303400)

[6.1 Руководство для неавторизованного пользователя 20](#_Toc104303401)

[6.2 Руководство для администратора 20](#_Toc104303402)

[6.3 Руководство клиента 24](#_Toc104303403)

[Заключение 27](#_Toc104303404)

[Список литературы 28](#_Toc104303405)

[Приложение А 29](#_Toc104303406)

# Введение

Web-приложение «Магазин компьютерных комплектующих» – это приложение, позволяющее приобрести запчасти для компьютера. Программный продукт позволяет ознакомиться с техническими характеристиками компьютерных комплектующих, производить поиск необходимых деталей, оформлять заказ на покупку деталей.

Данное Web-приложение реализовано на платформе Node JS [1]. Серверная часть приложения написана с помощью фреймворка Express JS [2]. Для пользователя доступно React-приложение с асинхронным UI [3]. Управление сайтом интуитивно, понятно и удобно.

В качестве СУБД для базы данных была выбрана MySql[4], в связи с ее простотой, производительностью и надежностью.

Node.js — программная платформа для разработки серверных web-приложений на языке JS/V8. Данная платформа применяется преимущественно на сервере, выполняя роль web-сервера. Node.js легко обрабатывает большое количество запросов одновременно и обеспечивает быстродействие приложения. Поэтому серверный JavaScript часто используют для создания SPA — одностраничных web-приложений, в которых рендеринг выполняется на стороне клиента.

Npm – менеджер пакетов, входящий в состав Node.js, позволяющий расширить возможности приложения.

MySQL — [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) [реляционная система управления базами данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94). Разработку и поддержку MySQL осуществляет корпорация [Oracle](https://ru.wikipedia.org/wiki/Oracle). MySQL является решением для малых и средних приложений. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа [MyISAM](https://ru.wikipedia.org/wiki/MyISAM" \o "MyISAM), поддерживающие полнотекстовый поиск, так и таблицы [InnoDB](https://ru.wikipedia.org/wiki/InnoDB" \o "InnoDB), поддерживающие транзакции на уровне отдельных записей.

# 1 Постановка задачи

Для выполнения данного программного средства были поставлены следующие задачи:

1. Обзор литературы по теме, примеры похожих программных средств, анализ их с критической стороны.
2. Проектирование, создание логических схем работы программного средства.
3. Программирование, отладка модулей проекта, сборка и комплексная отладка. Описание контроллеров и моделей.
4. Проведение тестирования программного средства.
5. Разработка руководства по установке и использованию приложения.

Решение каждой из задач отображено соответственно в главах:

1. Обзор аналогов.
2. Проектирование приложения.
3. Разработка приложения.
4. Тестирование
5. Руководство пользователя.

Помимо этого, программное средство должно иметь HTTPS-сервер и WebSocket-сервер. Иметь реляционную базу данных с достаточным количеством таблиц, чтобы полностью реализовать функционал интернет-магазина. Также должны быть регистрация, аутентификация, авторизация.

**2 Обзор аналогов**

«5 Элемент» — белорусская сеть магазинов электроники и бытовой техники. Основана в 1994 году. По состоянию на ноябрь 2020 года сеть насчитывает 70 розничных магазинов. Первый магазин открылся в Минске и был известен под названием BOSCH. После открытия четвертого магазина в Могилеве сеть приобретает известное всем название «5 Элемент».

«5 Элемент» предоставляет возможность оформления заказов, покупки электроники и бытовой техники. Рассмотрим пример каталога товаров на рисунке 2.1.

Для неавторизованного пользователя функционал web-приложения ничем не отличается от авторизированного пользователя и представляет все те же возможности. Приложение имеет огромное количество категорий, товаров. Помимо этого, в наличии имеет огромное количество товаров разных брэндов и комплектации. Поддерживается онлайн чат с консультантом магазина, который может помочь с поиском или выбором необходимого товара.

Для зарегистрированных пользователей существует опция добавления товаров в избранное, в корзину или же в список для сравнения товаров. Сайт имеет понятный интерфейс, однако огромное число кнопок может существенно повлиять на скорость поиска необходимых товаров у неопытных пользователей сети интернет.

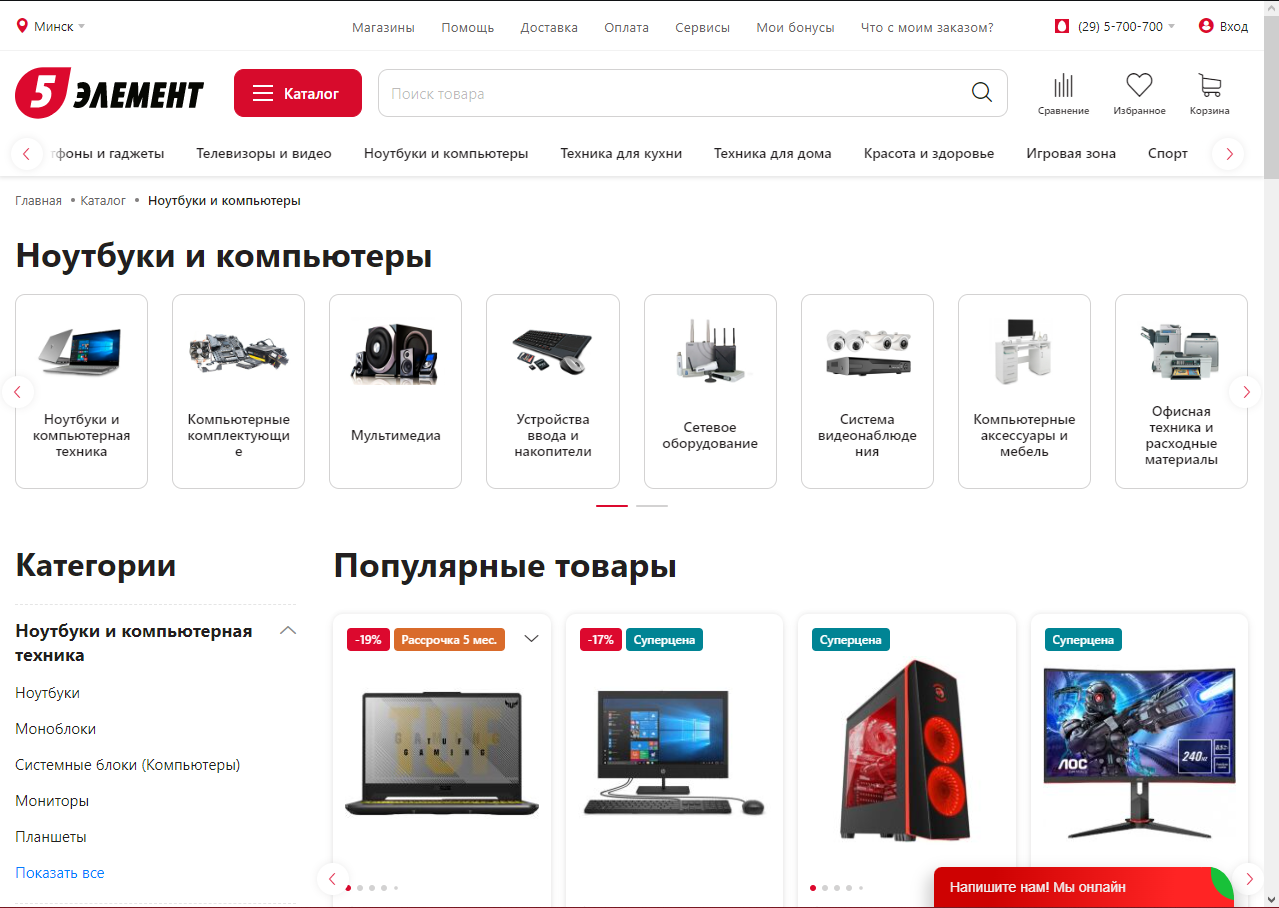


Рисунок 2.1 – Интерфейс сайта «5 элемент»

Следующий в очереди из аналогов web-приложения выступает интернет-магазин «21vek».

«21vek» — интернет-гипермаркет, специализирующийся на продаже электроники и бытовой техники, товаров для дома и прочего. Он является одним из лидеров интернет-порталов в Республике Беларусь. Бизнес-модель данного магазина характеризуется как интернет-гипермаркет с более чем 450 тысячами позиций в каталоге. У данной компании нет магазинов в Беларуси, а поступление товара к клиенту осуществляется посредством доставки до дома клиента или же доставкой до пункта самовывоза, где заказ хранится некоторое время перед тем, как его заберут.

Для авторизированных пользователей сайт позволяет выбирать и покупать товары. Ставить оценку и оставлять отзывы для конкретных продуктов. Есть корзина, списки для сравнения, личный кабинет пользователя, а также есть онлайн-чат с консультантами. В приложении удобно и визуально красиво оформлена фильтрация товаров и категорий, брэндов и характеристик. Пример каталога представлен на рисунке 2.2.

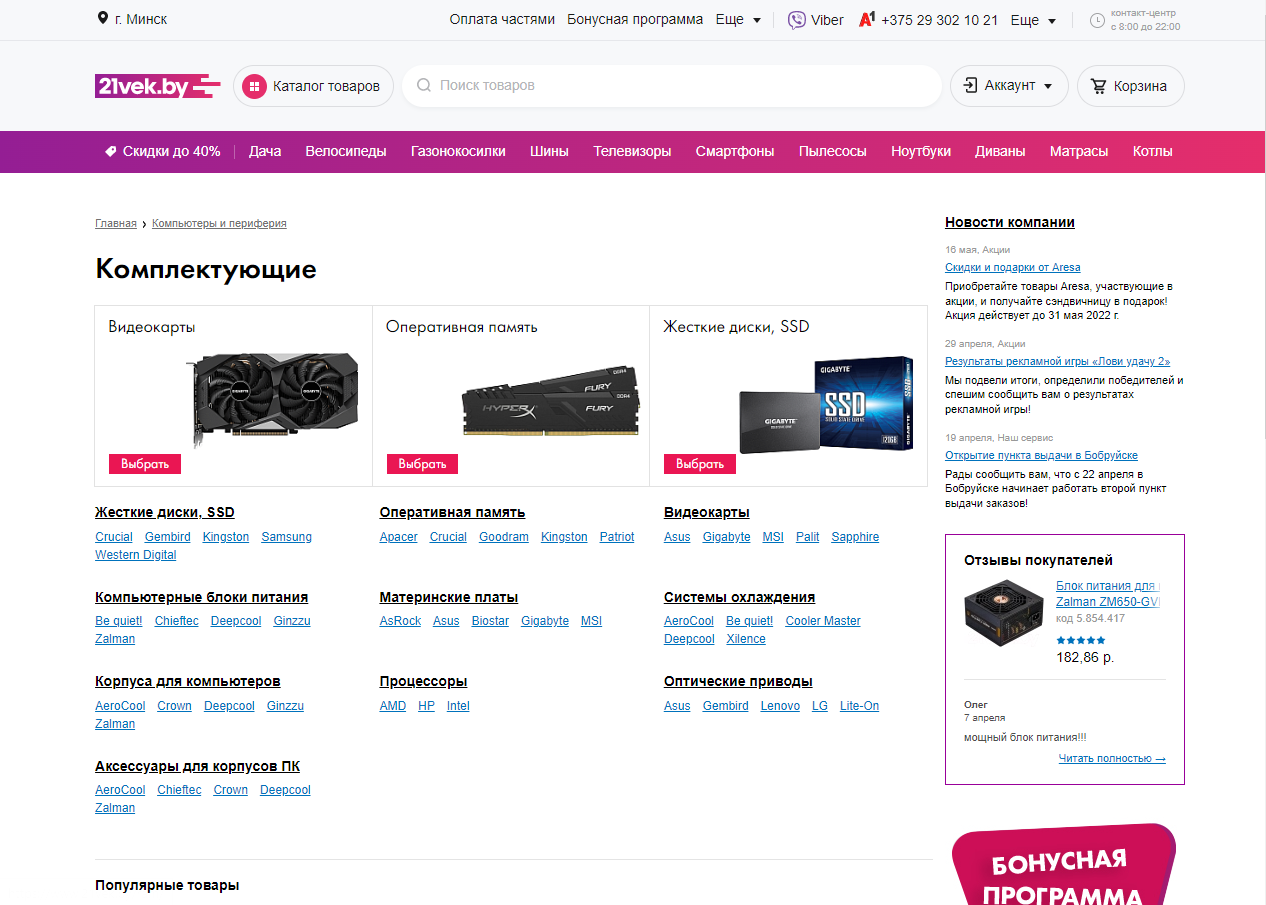


Рисунок 2.2 – Интерфейс сайта «21vek»

Проанализировав программы на схожую тематику, были разработаны технические требования к разработке web-приложения. Приложение должно:

* Иметь 2 типа пользователей (администратор и клиент);
* иметь интуитивно понятное и простое управление;
* обращаться к базе данных, успешно извлекать, изменять и дополнять данные, осуществлять поиск по базе и автоматическую запись некоторых полей;
* иметь различный функционал в зависимости от роли;
* выглядеть аккуратно, не иметь лишней информации;
* быстро отвечать на запросы.

# 3 Проектирование приложения

Для реализации разрабатываемого проекта была выбрана технология Node JS, платформа Express и систему управления базами данных MySql.

Взаимодействие серверной части приложения с реляционной базой данных осуществляется с помощью ORM-библиотеки Sequelize[5]. Благодаря данному подходу появилась возможность сопоставлять таблицы в базе данных и отношения между ними с классами.

Схема базы данных для разрабатываемого программного средства отображена в приложении А. Рассмотрим её подробнее.

Схема отображает таблицы базы данных и связи между ними.

Как видно из схемы, в приложении используются 13 таблиц:

Таблица Users содержит данные пользователя для его аутентификации, авторизации и идентификации. Она включает в себя следующие поля:

* ID – уникальный идентификатор записи;
* TYPE – тип пользователя;
* LOGIN – логин пользователя в приложении;
* PHONE – номер телефона пользователя;
* PASSWORD – пароль пользователя;
* FULL\_NAME – имя пользователя.

Таблица TOKENS содержит в себе токены для входа:

* ID – уникальный идентификатор токена;
* REFRESH\_TOKEN – обновленный токен;
* USER\_ID – идентификатор пользователя.

Таблица ATTRIBUTES содержит в себе все данные об атрибутах продуктов:

* ID – уникальный идентификатор атрибута;
* NAME – имя атрибута;
* ATRIBUTE\_ID – идентификатор атрибута из таблиц с атрибутами.

Таблица ATTRIBUTE\_OPTIONS является дополнением к таблице атрибутов:

* ID – уникальный идентификатор связи;
* PRODUCT\_ID – идентификатор продукта из таблицы PRODUCTS;
* ATTRIBUTE\_ID – идентификатор атрибута из таблицы ATTRIBUTES.

Таблица ATTRIBUTE\_SELECTS содержит данные конкретного типа атрибутов:

* ID – уникальный идентификатор атрибута;
* PRODUCT\_ID – идентификатор продукта из таблицы PRODUCTS;
* OPTION\_ID – идентификатор связи из таблицы ATTRIBUTE\_OPTIONS.

Таблица ATTRIBUTE\_CHECKBOXES содержит данные конкретного типа атрибута:

* ID – уникальный идентификатор связи;
* VALUE – значение атрибута;
* PRODUCT\_ID – идентификатор продукта из таблицы PRODUCTS;
* ATTRIBUTED\_ID – идентификатор атрибута из таблицы ATTRIBUTES.

Таблица ATTRIBUTE\_NUMBERS содержит данные конкретного типа атрибута:

* ID – уникальный идентификатор связи;
* VALUE – значение атрибута;
* PRODUCT\_ID – идентификатор продукта из таблицы PRODUCTS;
* ATTRIBUTE\_ID – идентификатор атрибута из таблицы ATTRIBUTES.

Таблица ATTRIBUTE\_TEXTS содержит данные конкретного типа атрибута:

* ID – уникальный идентификатор связи;
* VALUE – значение атрибута;
* PRODUCT\_ID – идентификатор продукта из таблицы PRODUCTS;
* ATTRIBUTE\_ID – идентификатор продукта из таблицы ATTRIBUTES.

Таблица BRANDS содержит в себе все данные о брендах, а именно:

* ID – уникальный идентификатор бренда;
* NAME – название бренда;
* DELETED – проверка, удалён ли бренд.

Таблица CATEGORIES содержит в себе все данные о категориях, а именно:

* ID – уникальный идентификатор категории;
* NAME – название категории;
* DELETED – проверка, удалена ли категория.

Таблица PRODUCTS содержит в себе все данные о продуктах, а именно:

* ID – уникальный идентификатор продукта;
* NAME – название продукта;
* PRICE – цена продукта;
* IMAGE – изображение продукта;
* DELETED – проверка, удалён ли продукт.
* CATEGORY\_ID – идентификатор категории из таблицы CATEGORIES.
* BRAND\_ID – идентификатор бренд из таблицы BRANDS.

Таблица PRODUCT\_ORDERS является связующей таблицей для заказов и продуктов и позволяет реализовать связь многих ко многим. Содержит в себе все данные товаров в заказе, а именно:

* ID – уникальный идентификатор заказа;
* PRICE – цена;
* QUANTITY – количество товаров в заказе;
* PRODUCT\_ID – идентификатор продукта из таблицы PRODUCTS.
* ORDER\_ID – идентификатор заказа из таблицы ORDERS.

Таблица ORDERS содержит в себе все данные о заказах, а именно:

* ID – уникальный идентификатор заказа;
* NUMBER – номер заказа;
* STATUS – статус заказа;
* USER\_ID – идентификатор пользователя из таблицы USERS.

Помимо этого, в каждой таблице есть столбца CREATED\_AT и UPDATED\_AT для отслеживания даты и времени изменение информации. Также, в таблицах PRODUCTS, CATEGORIES и BRANDS используется столбец DELETED для того, чтобы при удалении соответствующих брендов, категории или продукта из продажи, не пропадали уже имеющиеся заказы, находящиеся в статусе ожидания.

В разрабатываемом программном средстве существует возможность добавления, изменения и удаления записей в базе данных. Разработано REST API благодаря которому можно взаимодействовать с базой данных посредством HTTPS запросов.

База данных к приложению подключается c помощью параметров, приведенных в листинге 3.1. Const sequelize вызывает метод Sequelize с параметрами, содержащими имя базы данных, имя пользователя и пароль.

const sequlize = new Sequelize(‘online\_store’,’root’,’dfyef2001’,{

host: ’localhost’,

dialect: ’MySQL’,

pool: {

max:10,

min:0,

acquire:30000,

idle:10000,

},

logging: false,

});

Листинг 3.1 – Подключение к базе данных

После подключения к базе данных, открывается пул соединений. Пулы соединений являются кэшом соединений с базой данных, который поддерживается для повторного использования. Они используются для повышения производительности выполнения команд в базе данных, что крайне необходимо при разработке крупных проектов или же в тех с случаях, когда данные в базе данных имеют большой вес и могут сильно ограничивать скорость выполнения соответствующих запросов.

# 4 Разработка приложения

## 4.1 Разработка серверной части

На рисунке 4.1 отображена структура серверной части приложения

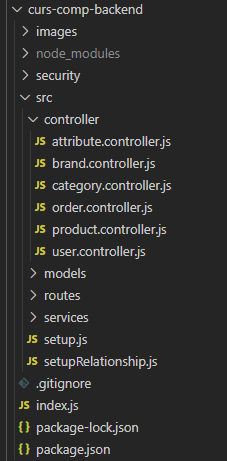


Рисунок 4.1 – Структура сервера

Проект поделен на 4 раздела: «controller», «models», «routes», «services».

В директории «controllers» содержатся контроллеры. Контроллеры обеспечивают взаимодействие пользователя и системы, обрабатывая и проверяя запросы.

Для удобство все методы были разбиты по файлам в зависимости от таблицы в базе данных, над которыми выполняются операции. Однако в данном директории отсутствуют контроллеры для всех типов атрибутов, а также для таблицы, содержащей токены для входа. Все методы, затрагивающие токены, а именно: авторизация – представлены в контроллере для пользователей. Обработка запросов для всех видов атрибутов представлена в одном общем файле. Пример одного из методов контроллера, а именно регистрация пользователя, представленная в файле «user.controller.js», приведен в листинге 4.2.

async registerUser(req, res) {

try {

const { login, phone, password, fullName } = req.body;

const findUser = await User.findOne({ where: { login } });

if (findUser) {

throw new Error('Такой пользователь уже существует');

}

const hashPassword = await bcrypt.hash(password, 3);

const user = await User.create({ login, phone, fullName, password: hashPassword, type: 'client' });

const accessToken = jwt.sign({ id: user.id, login: user.login, phone: user.phone, fullName: user.fullName, type: user.type }, 'access-secret', { expiresIn: '24h' });

return res.json({ token: accessToken });

} catch (error) {

res.status(500).send({

message: error.message || 'Непредвиденная ошибка',

});

}

Листинг 4.2 –Регистрация пользователя

После регистрации происходит авторизация пользователя. Проверка пользователей происходит посредством использования JWT-токенов. Для этого используется модуль «jsonwebtoken». Реализация токенов представлена в листинге 4.3.

const jwt = require('jsonwebtoken');

async checkAuth(req, res) {

try {

const token = req.headers.authorization.split(' ')[1];

if (!token) {

throw new Error('Пользователь не авторизован');

}

const decodedData = jwt.verify(token, 'access-secret');

res.json({ message: 'Success', login: decodedData.login, id: decodedData.id, phone: decodedData.phone, fullName: decodedData.fullName, type: decodedData.type });

} catch (error) {

res.status(500).send({

message: error.message || 'Непредвиденная ошибка',});}}

Листинг 4.3 – Jwt-токен

Директорий «security» содержит ключ и сертификат для поддержки HTTPS-запросов. Подключение ключа и сертификата представлено в листинге 4.4. Сертификат был сгенерирован на основе ключа. После чего был зарегистрирован в доверенном корневом центре сертификации. А также были добавлены псевдонимы для запросов, посредством протокола Https.

const https = require('https');

const options = {

key: fs.readFileSync("security/RS-GIS.key").toString(),

cert: fs.readFileSync("security/RS-GIS.crt").toString(),

};

https.createServer(options, app).listen(PORT, () => {

console.log(`Server is running on port ${PORT}.`);

});

Листинг 4.4 – HTTPS-сервер

После запуска Https-сервера появляется возможность использовать https-запросы к серверу, используя в URL вместо «localhost» объявленные в сертификате псевдонимы.

Директория «models» содержит файлы с описанием всех таблиц и связей между ними в реляционной базе данных. Аналогично другим директориями, описание каждой таблицы происходит в отдельных файлах. Пример описания одной из таблиц представлен в листинге 4.5.

module.exports = (sequelize, Sequelize) => {

const Product = sequelize.define('product', {

name: {

type: Sequelize.STRING,

unique: true,

allowNull: false,

},

price: {

type: Sequelize.FLOAT,

allowNull: false,

defaultValue: 0,

},

image: {

type: Sequelize.STRING,

},

deleted: {

type: Sequelize.BOOLEAN,

defaultValue: false,

},

});

return Product;

};

Листинг 4.5 – Описание таблицы «PRODUCTS»

Директория «routes» представляет собой файлы с маршрутизацией запросов. Все запросы разделены по соответствующим таблицам, над которыми выполняются операции. Пример маршрутизатора приведен в листинге 4.6.

const Router = require('express');

const router = new Router();

const brandController = require('../controller/brand.controller');

router.post('/brand/create', brandController.createBrand);

router.post('/brand/update', brandController.updateBrand);

router.post('/brand/delete', brandController.deleteBrand);

router.get('/brand/list', brandController.getBrands);

module.exports = router;

Листинг 4.6 – Маршрутизатор запросов к таблице «BRANDS»

## Разработка клиентской части

Клиентская часть проекта написана с использованием фреймворка «React». Фреймворк позволяем удобно сочетать язык JavaScript и декларативный подход при написании разметки и стилей. На рисунке 4.7 отображена структура проекта клиентской части приложения.

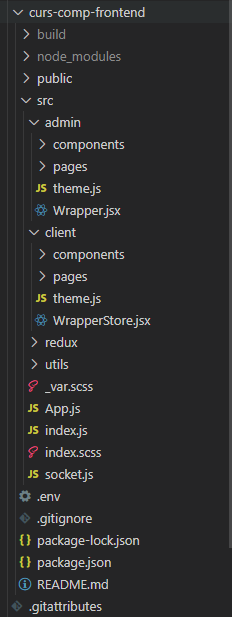


Рисунок 4.7 – Структура клиентской части приложения

Проект поделен на 2 раздела: «client», «admin», которые предоставляют разный функционал в зависимости от тип пользователя. Каждый из разделов содержит 2 основных директория: «components», «pages», содержащие react-компоненты и страницы приложения соответственно.

Клиентская часть реализует асинхронный подход к реализации пользовательского интерфейса.

Важным файлов в клиентской части приложения является файл «app.js», содержащий в себе список путей для доступа к соответствующим страницам приложения. Данная часть является по сути аналогом маршрутизатора для клиентской части приложения.

Помимо этого, в структуре проекта есть ещё 2 репозитория: «utils», «redux». В них находятся https-запросов с клиентской стороны.

На рисунке 4.8 представлена структура раздела администратора, содержащий страницы и компонента для пользователя, авторизированного как администратор web-приложения. Аналогичные компоненты и страницы есть и для клиента, с той лишь разницей, что ему недоступны некоторые страницы, позволяющие производить добавление, изменение и удаление товаров, брэндов, категорий, а также отсутствует возможность просмотра заказов всех пользователей.

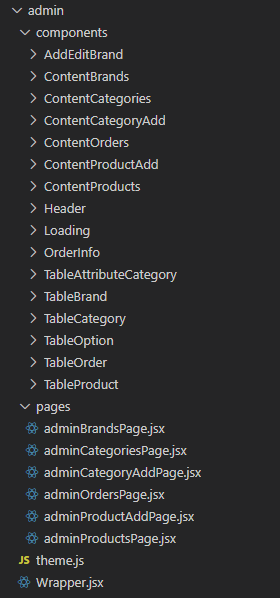


Рисунок 4.8 – Структура раздела администратора

## Реализация основного функционала приложения

Каждый слой приложения реализуют асинхронную систему вызовов. Слой работы с базой данных вызывает асинхронные методы для выполнения sql-команд, а верхние слои дожидаются их исполнения. Данная модель позволяет параллельную работу нескольких пользователей без ожидания завершения операций других пользователей.

Для обеспечения защиты данных во время процесса пользования приложением был использован протокол https. Для этого был создан и зарегистрирован сертификат, который впоследствии станет основой для https сервера.

В ходе реализации приложения было разработано REST API для предоставления гибкого интерфейса взаимодействия с приложением. Таким образом работа сервера не привязана к клиентской части. Клиентская часть может быть реализована на любом языке, любой платформе и любым желающим. Благодаря такой архитектуре можно реализовать свое собственное приложение и через него полноценно получать информацию о компьютерных комплектующих.

## Реализация Web-сокета

Сервер web-сокета находится в главном модуле приложения. Для работы с ним был выбран модуль «socket.io»[6]. Первым делом осуществляется подписка на событие «connection», чтобы пользователи смогли подключиться. Реализация серверной части web-сокета и обрабатываемое событие представлено в листинге 4.9.

const io = require('socket.io')(server, {

cors: {

origin: '\*',

},

});

io.on('connection', (socket) => {

console.log('New client connected');

socket.on('CREATE\_ORDER', (sockett) => {

console.log('CREATE ORDER ');

io.emit('REFRESH\_ORDERS');

});

socket.on('disconnect', () => {

console.log('Client disconnected');

clearInterval(interval);

});

});

Листинг 4.9 – Серверный web-сокет и обрабатываемое событие

# Тестирование

В разработанном программном средстве предусматриваются многие негативные сценарии.

Создавая новую учетную запись, предназначенную для помещения в базу данных, при незаполненных обязательных полях кнопка регистрации будет заблокирована, пример на рисунке 5.1.

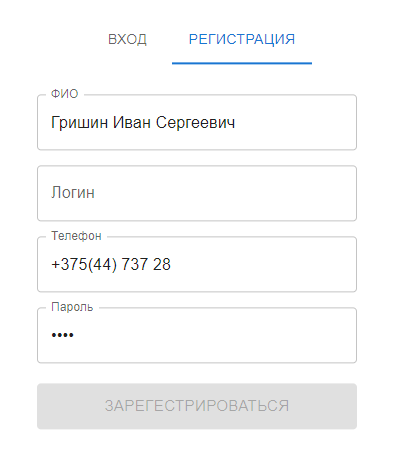


Рисунок 5.1 – Блокировка кнопки регистрации

Так же существует ограничение, накладываемое на поле для ввода номера телефона. Здесь автоматически ввод номер начинается с префикса, а именно 2 цифры идущие после кода страны. Код страны вводится автоматически при вводе первой цифры номера телефона. Используется код Республики Беларусь, что исключает возможно регистрации, используя номера телефонов других стран. На данное поле действует ограничение по вводу количества цифры, ограничение по количеству цифр в префиксе номера и по типу данных, что означает, что ввод букв в данное поле невозможен.

При попытке зарегистрировать пользователя с логином, который уже имеется в базе данных, выводится сообщение, отображенное на рисунке 5.2. Остальные же поля не имеют сильного влияния на аккаунт поскольку не являются уникальными идентификатором для пользователей, а служат дополнительной информацией о клиентах.

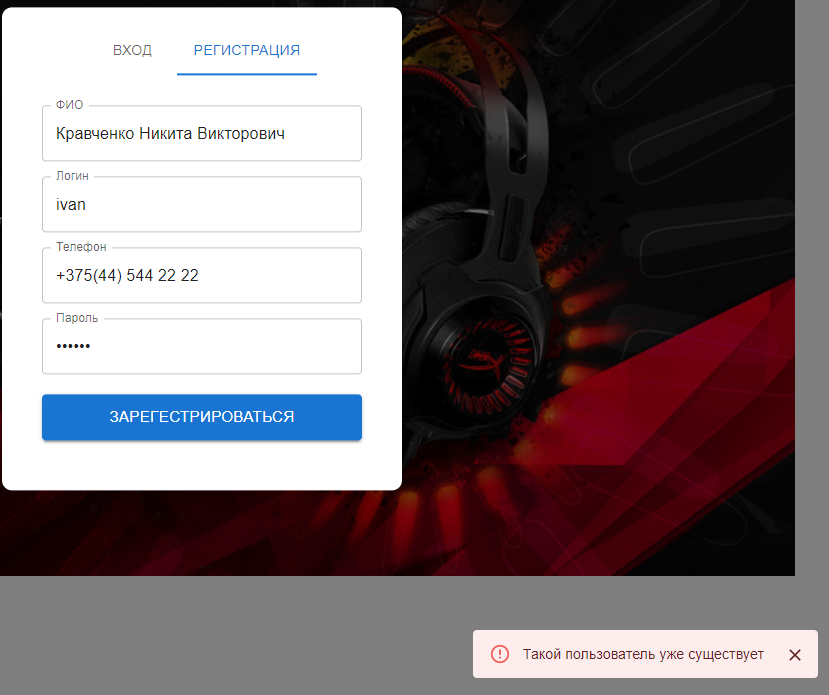


Рисунок 5.2 – Сообщение о существующем пользователе

Если пользователь уже зарегистрирован, он может нажать на кнопку «вход». После нажатия откроется форма авторизации в приложение, показанная на рисунке 5.3, где он может ввести свой логин и пароль для получения доступа к дополнительным возможностям приложения.

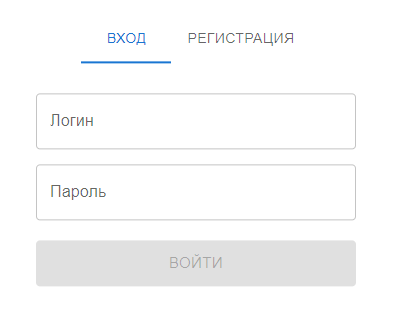


Рисунок 5.3 – Форма авторизации в приложение

Если при входе в систему указываются неверные данные, выводится сообщение об ошибке. Пример сообщения продемонстрирован на рисунке 5.4.

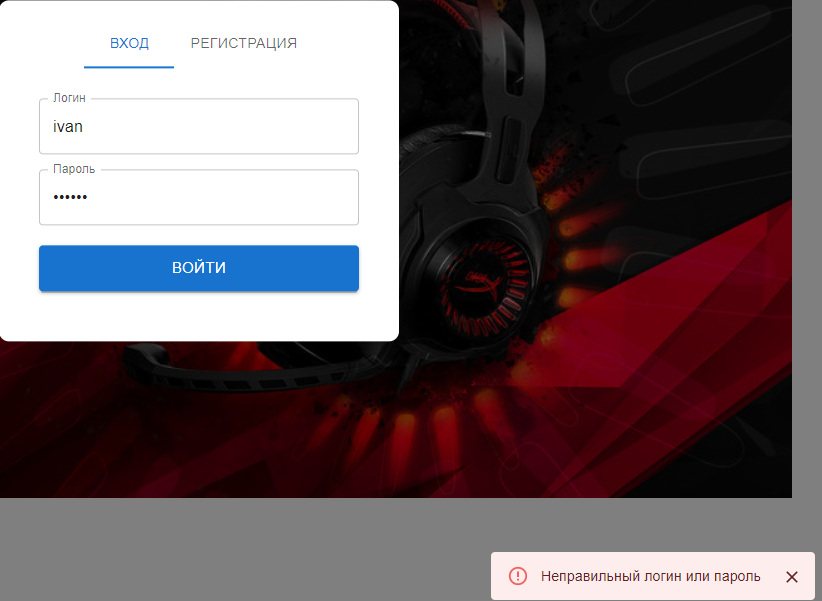


Рисунок 5.4 – Сообщение о неправильно введенных данных пользователя

Помимо этого, при попытке оформления заказа, указав нулевое или отрицательное значение количества товаров, его значение автоматически изменится до минимального – единицы. Пример оформления заказа нуля видеокарт представлен на рисунке 5.5.

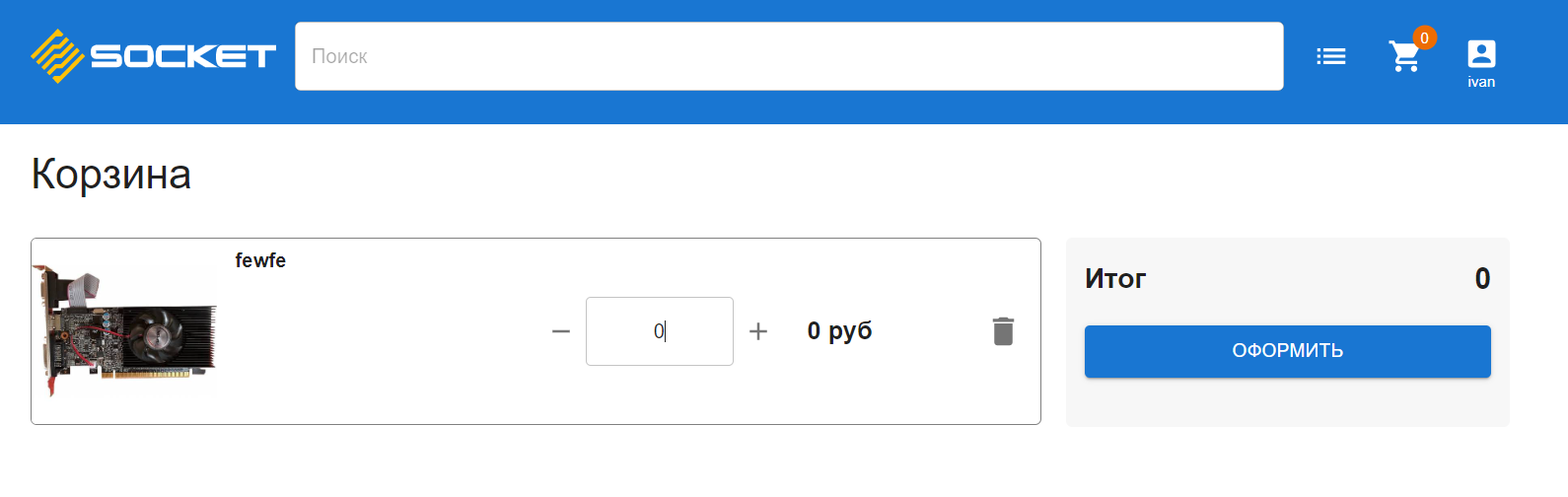


Рисунок 5.5 – Заказ нуля товаров

Подобный вид ограничения также накладывается на добавления новых товаров в режиме администратора, запрещающий добавление товара с отрицательной или нулевой ценой. Однако в режиме администратора, в случае нарушения ограничений, накладываемые на поля, описывающие характеристики товара, будет блокироваться кнопка добавления нового товара в базу данных. Пример добавления нового товара с нулевой ценой представлен на рисунке 5.6.

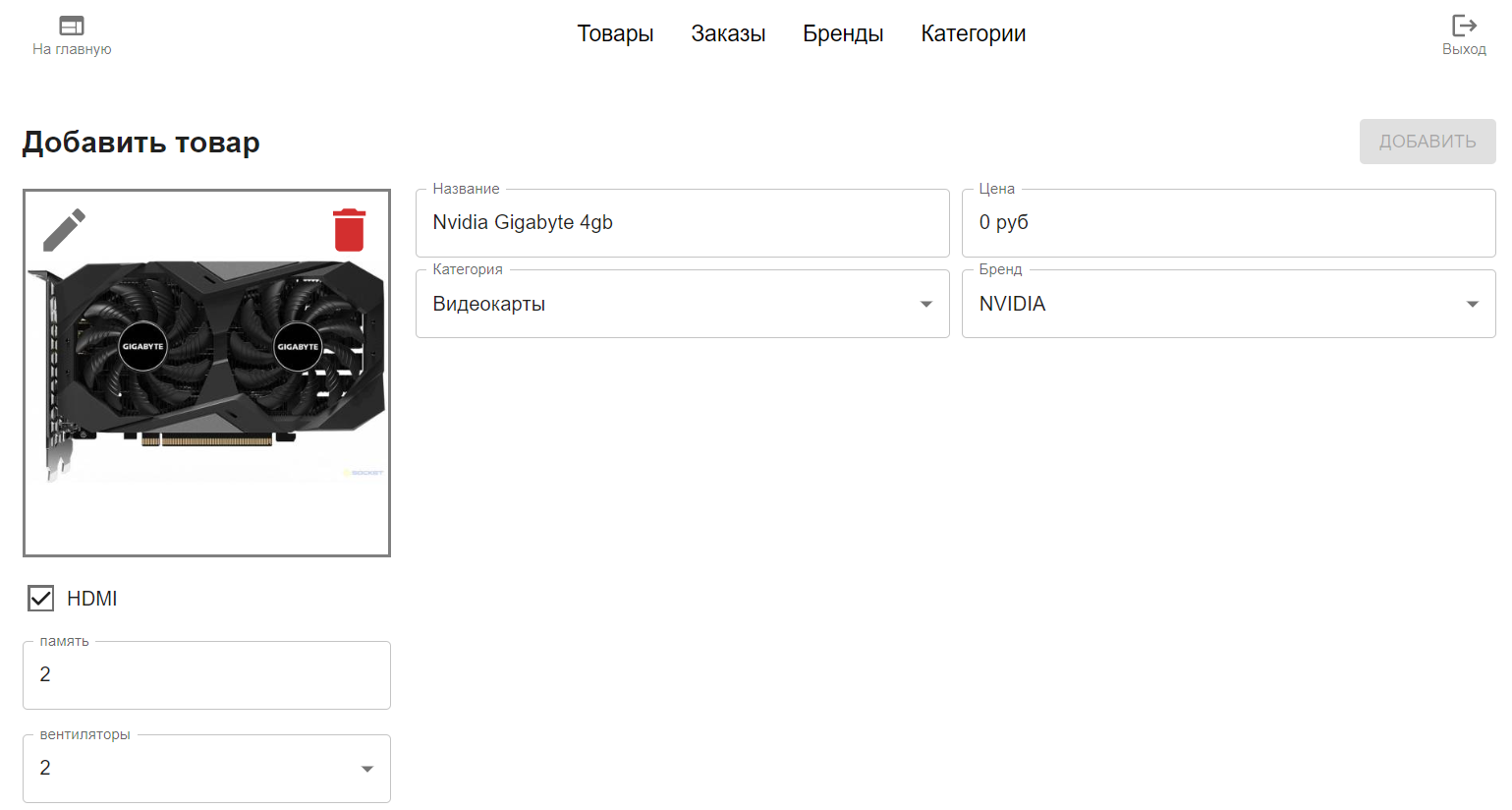


Рисунок 5.6 – Блокировка кнопки добавления товара

Помимо этого, существует ещё ряд ограничении, накладываемые на действия администратора с целью предотвращения возможных ошибок и неточностей при добавлении новых товаров, брэндов или категорий, а также при их изменении. Одно из таких ограничений представляет собой блокировку изменения категории, представленную на рисунке 5.7.

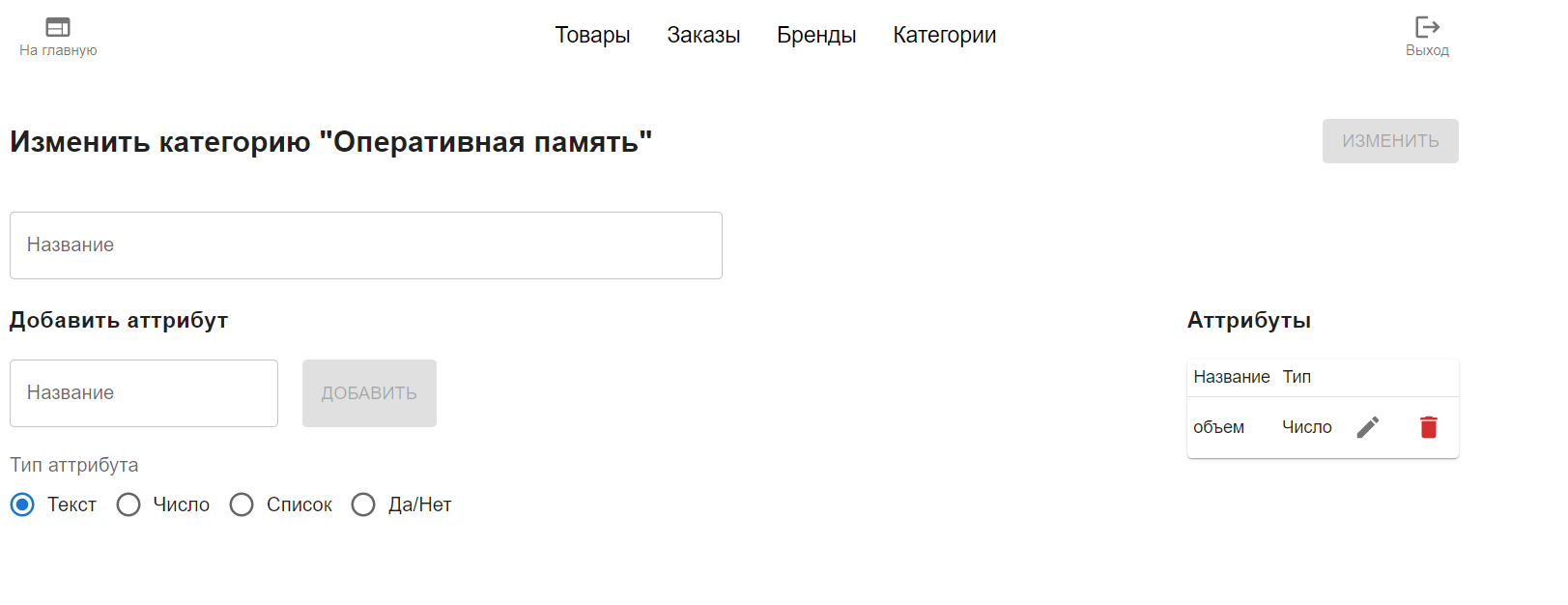


Рисунок 5.7 – Блокировка кнопки изменения категории

На данном примере видно, что администратор не может изменить существующую категорию, оставив название пустым.

# Руководство пользователя

Для корректной работы приложения необходимо наличие компьютера со всеми необходимыми для его работы компонентами (процессором, видеокартой, оперативной памятью, монитором и т.д.), браузер-клиент, а также подключение к сети Internet.

Фактически, установка для разработанного в данном курсовом проекте приложения не требуется. Если необходимо запустить приложение в локальной сети, нужно установить конфигурацию приложения, в частности скачать все используемые пакеты из npm. Также необходимо установить MySQL и создать пользователя с базой данных. Все данные, необходимые для работы базы данных создаются автоматически при запуске серверной части приложения.

Чтобы иметь возможность пользоваться основным функционалом приложения, необходимо создать аккаунт администратора вручную используя скрипты в MySQL.

При входе в аккаунт, необходимо ввести данные, указанные при регистрации.

## Руководство для неавторизованного пользователя

В разработанном web-приложении пользователь является неавторизованным, если токен, хранящийся в localstorage браузера, не является действующим, или же если токена вообще нет.

Функционал неавторизованного пользователя представляет собой авторизацию, регистрацию, выбор товаров, добавление в корзину. При оформлении заказа у неавторизированного пользователя появится окно для входа в аккаунт или создания нового. Руководство по поиску и оформлению заказа представлено в главе 6.3.

Для создания аккаунта необходимо ввести свои имя, фамилию, отчество, номер телефона, логин и пароль.

Пользователь также может войти в созданный аккаунт, заполнив поля на форме аутентификации. Чтобы аутентификация прошла успешно, пользователю необходимо указать логин и пароль, которые были указаны при регистрации.

## Руководство для администратора

После аутентификации, у администратора появляется возможность войти в режим администрирования. Для этого необходимо зайти в личный кабинет, нажав на соответствующую иконку в верхнем правом углу экрана.

После чего появляется возможность добавления новых категорий, брендов и товаров в магазине. Демонстрация процесса создания новой категории представлен на рисунке 6.1.

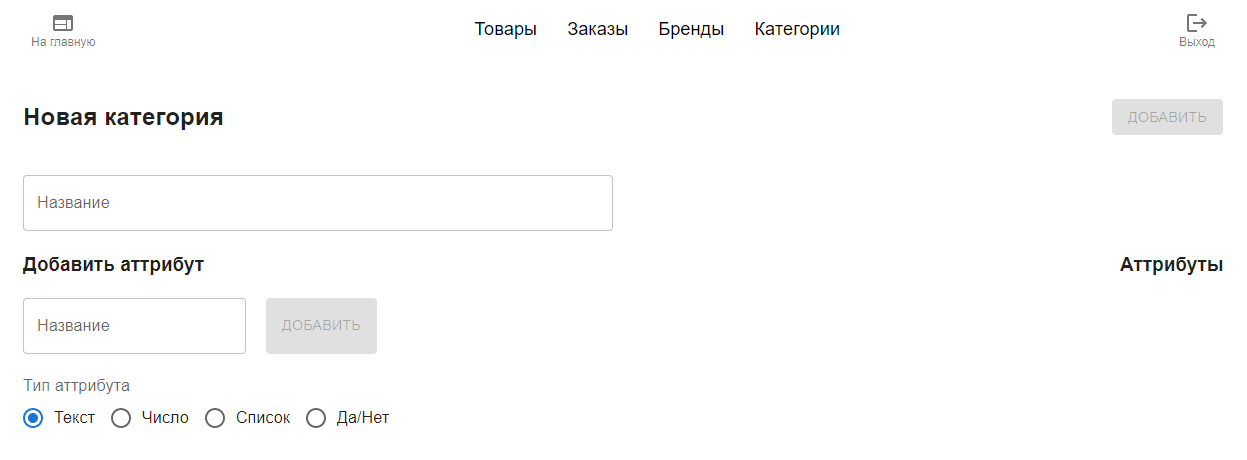


Рисунок 6.1 – Создание новой категории

Здесь администратор выбирает название новой категории, а также необходимые для неё атрибуты. После создание категории, её также можно изменить. Демонстрация на рисунке 6.2.

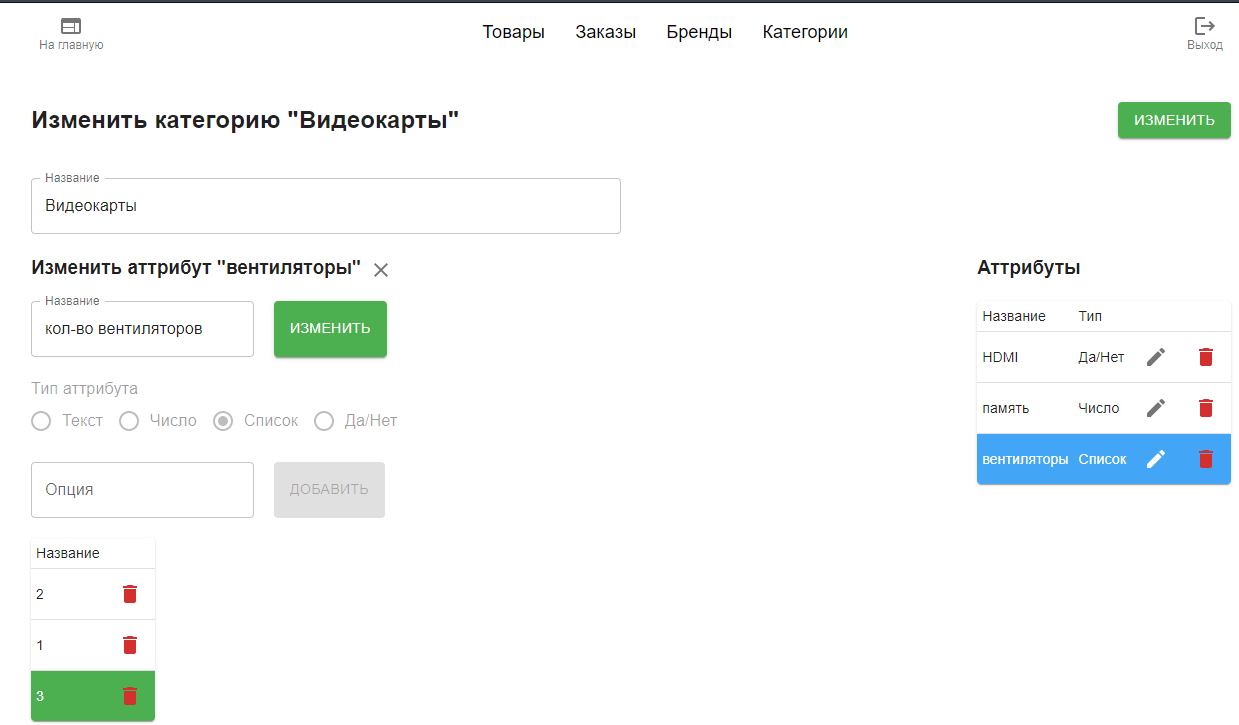


Рисунок 6.2 – Редактирование категории

Далее администратор может создавать новые бренды. Единственным параметром для бренда является его имя, пример представлен на рисунке 6.3. После создания нового брэнда, его можно будет изменить на той же странице, нажав на иконку карандаша рядом с соответствующим брэндом.

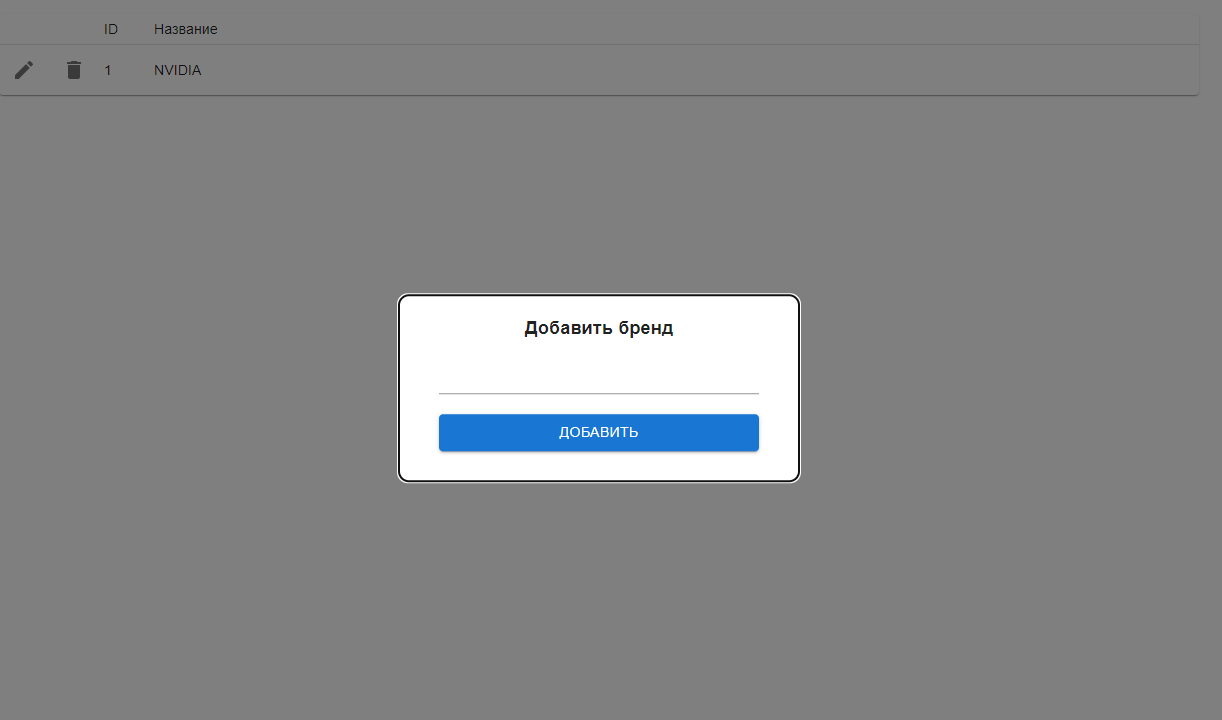


Рисунок 6.3 – Добавление нового бренда

Имея категорию и бренд, администратор можешь добавить новые товары для магазина. Добавление товаров приведено на рисунке 6.4.

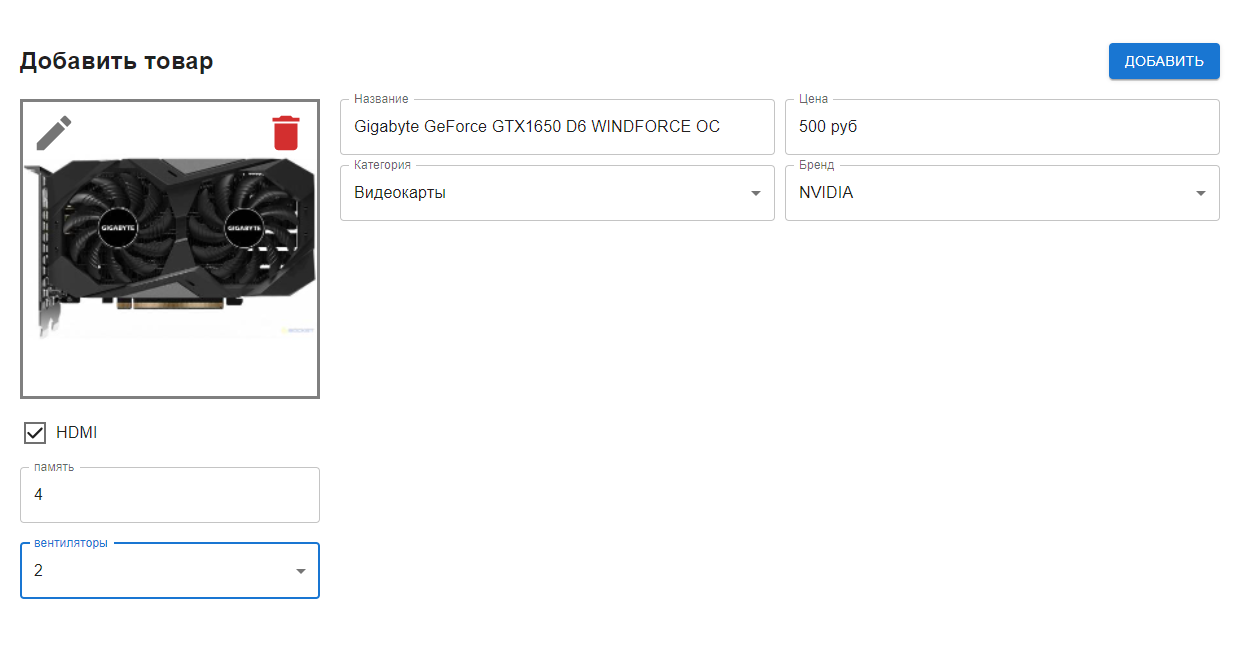


Рисунок 6.4 – Меню добавления товара

Здесь происходит выбор изображения товара, его название, цена в рублях, категория, бренд и атрибуты, созданные для конкретной категории.

Также администратор может изменять только что созданный или уже имеющийся товар. Изменение товара приведено на рисунке 6.5. При изменении товара администратор может оставить некоторые значения полей, описывающих товар, неизмененными. Однако здесь есть ограничения, не позволяющие ввести заведомо неправильную информацию о товаре, например установка нулевой цены.

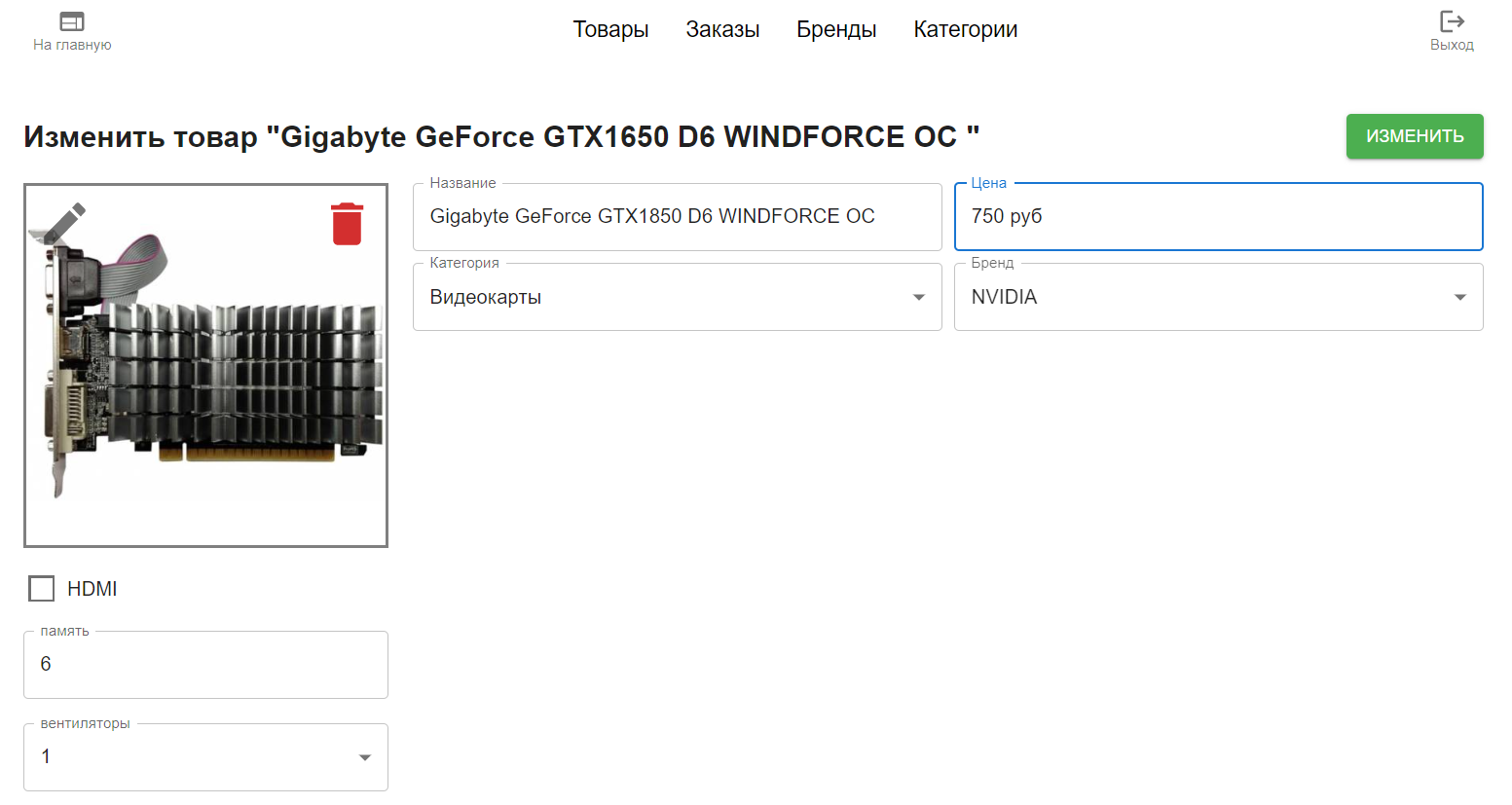


Рисунок 6.5 – Меню изменения товара

Здесь происходит выбор изображения товара, его название, цена в рублях, категория, бренд и атрибуты

Помимо этого, администратор можешь отслеживать все заказы, а также менять статус конкретного заказа. Изменение статусу заказа приведено на рисунке 6.6.

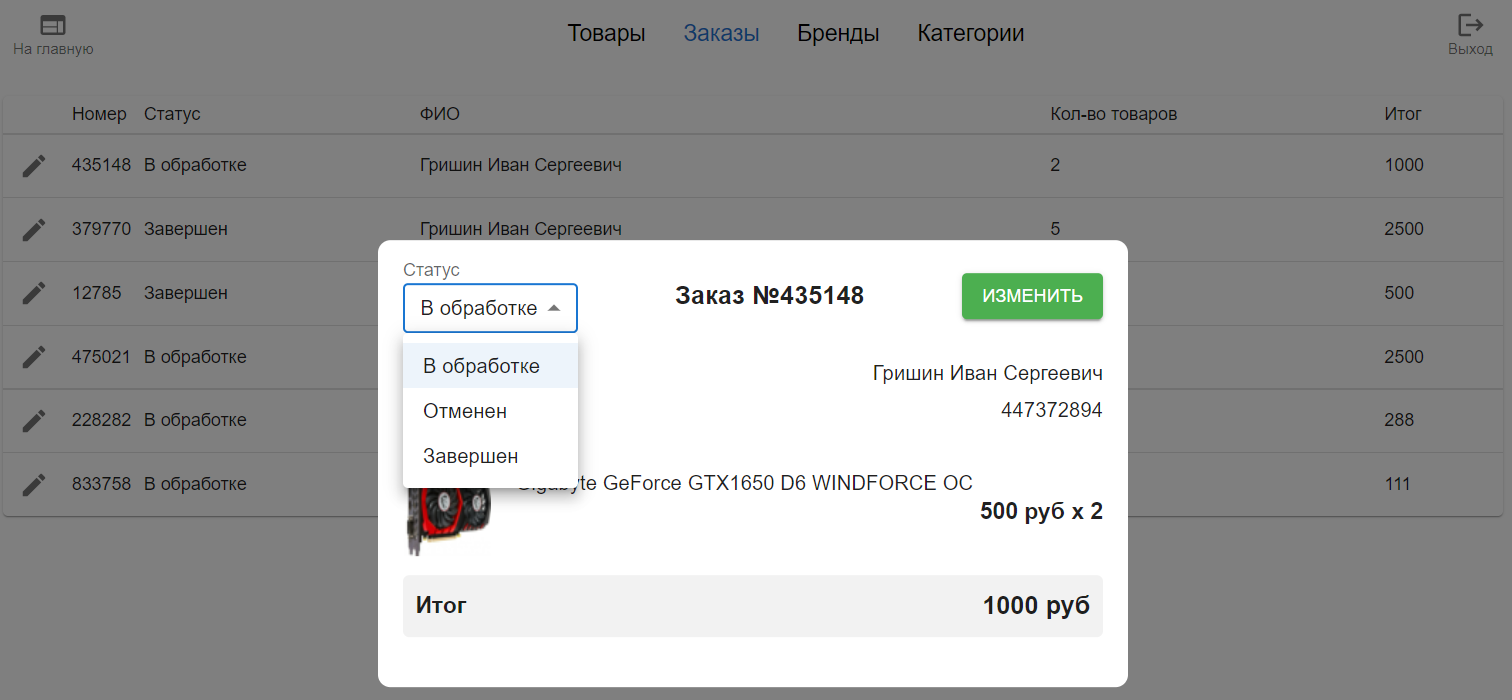


Рисунок 6.6 – Изменение статуса заказа

Также администратор имеет возможность изменять бренды, нажав на соответствующую иконку рядом с необходимым брендом. Изменение происходит аналогично процессу создания нового брэнда.

* 1. **Руководство клиента**

После аутентификации, у пользователя появляется возможность оформлять заказы из корзины.

Первоначально, после аутентификации клиента перенаправляет на основную страницу сайта, где он видит рекламу и недавно добавленные товары. Для быстрого и удобного поиска необходимого товара клиент имеет возможность воспользоваться поисковой строкой. Пример использования поисковой строки представлен на рисунке 6.7.

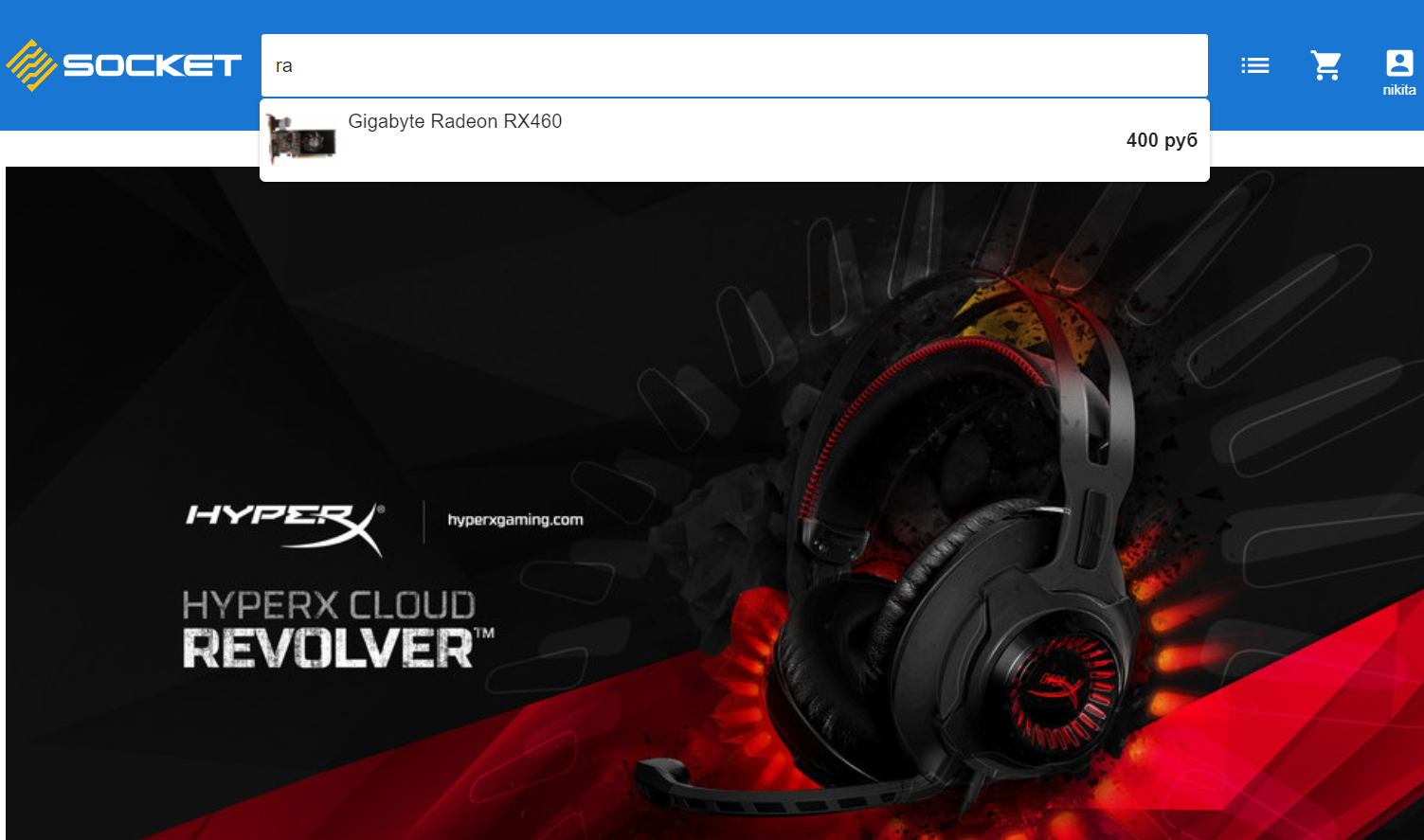


Рисунок 6.7 – Работа с поисковой строкой

Если же клиент не знает название нужного товара, он может перейти в раздел категории, выбрав соответствующую иконку в верхнем правом углу. Раздел категорий товаров представлен на рисунке 6.8.

Здесь отображаются все доступные категории компьютерных комплектующих. В случае, если товары в данной категории отсутствуют, категория всё равно будет отображаться у клиента.

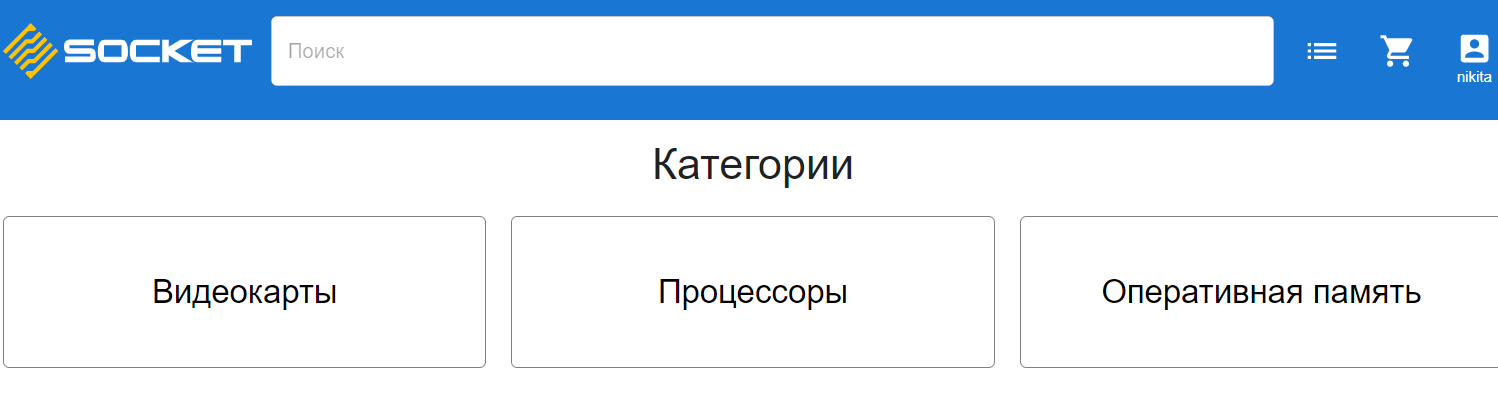


Рисунок 6.8 – Список категорий

Клиент может сразу добавить товар в корзину для последующего оформления заказа. Для получения дополнительной информации о товаре, необходимо нажать на его название. Здесь отображаются все атрибуты товара, его цена, бренд, категория, а также есть возможность выбора необходимого количества товара. Пример характеристик товара представлен на рисунке 6.9.

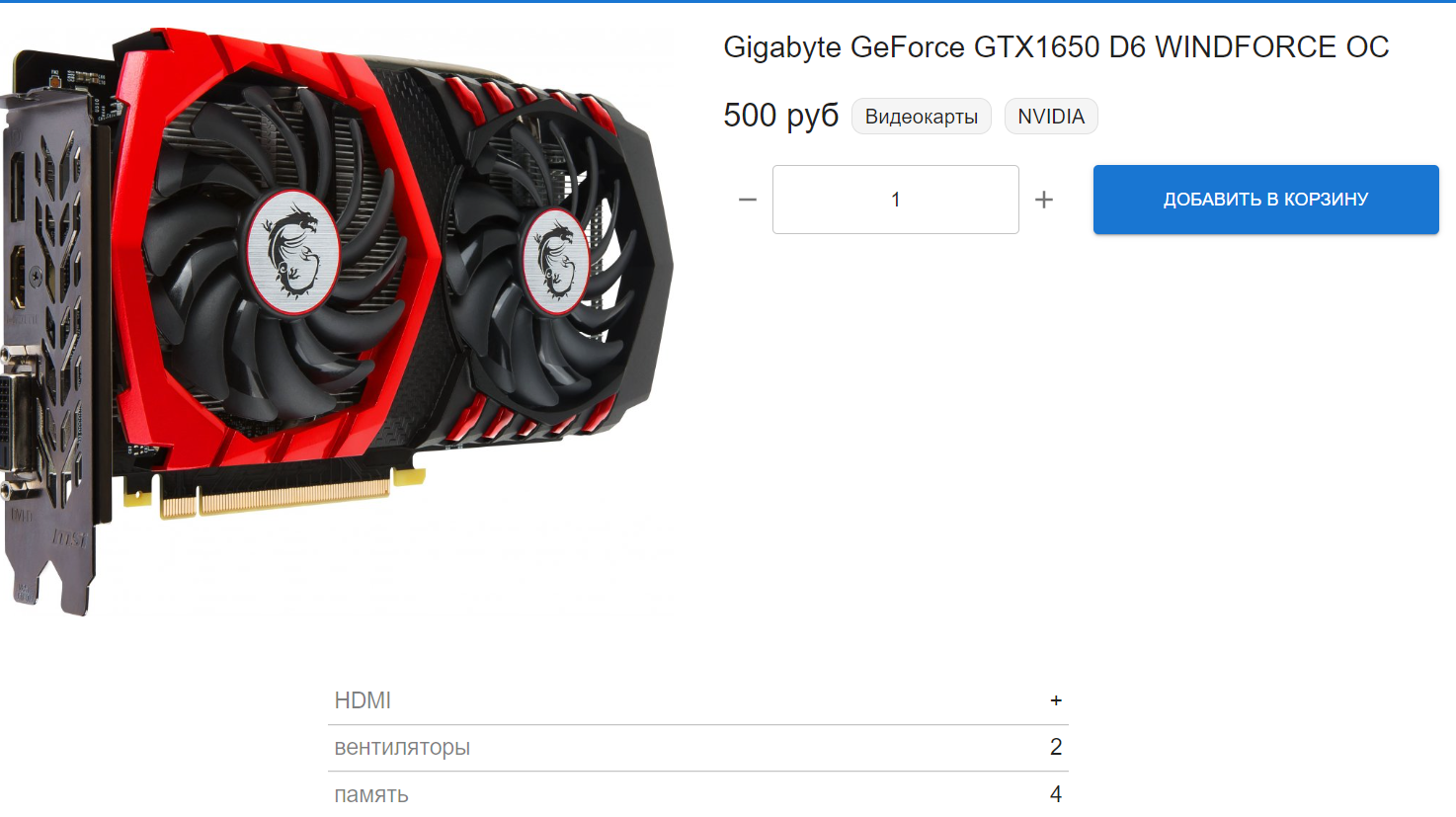


Рисунок 6.9 – Характеристика товара

После выбора товаров, все они помещаются в корзину, после чего она подсвечивается количеством добавленных товаров. Для перехода в корзину, нужно нажать на иконку корзины в верхнем правом углу экрана. Здесь пользователь может в последний раз изменить количество товаров или удалить лишние. При выходе из корзины все товары в ней сохраняются, что позволяет добавить новых товаров к заказу. Страница корзины приведена на рисунке 6.10.

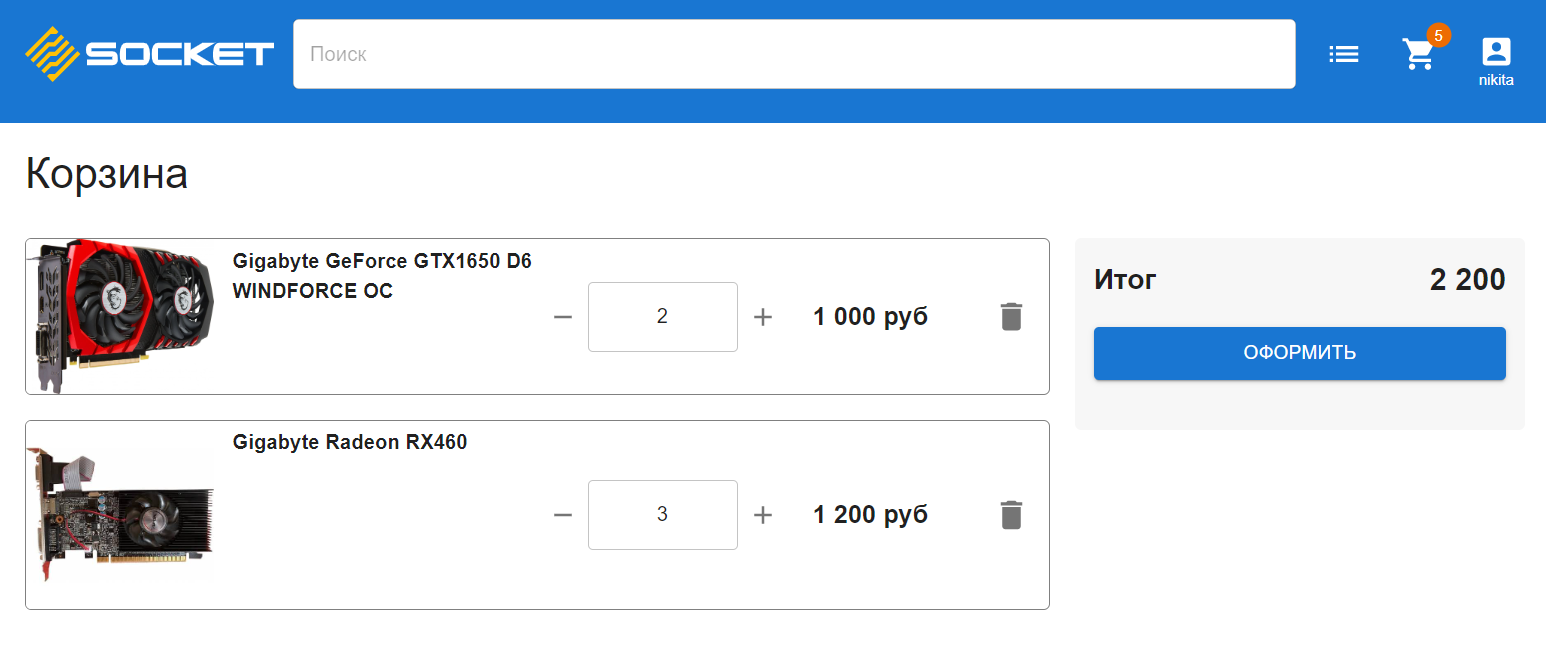


Рисунок 6.10 – Меню корзины

За этим следует оформления заказа. После чего все заказы будут отображаться в личном кабинете, с указанием статуса заказа. Подробную информацию о конкретном заказе можно узнать, щелкнув по нему. Личный кабинет со списком заказов пользователя представлен на рисунке 6.11.

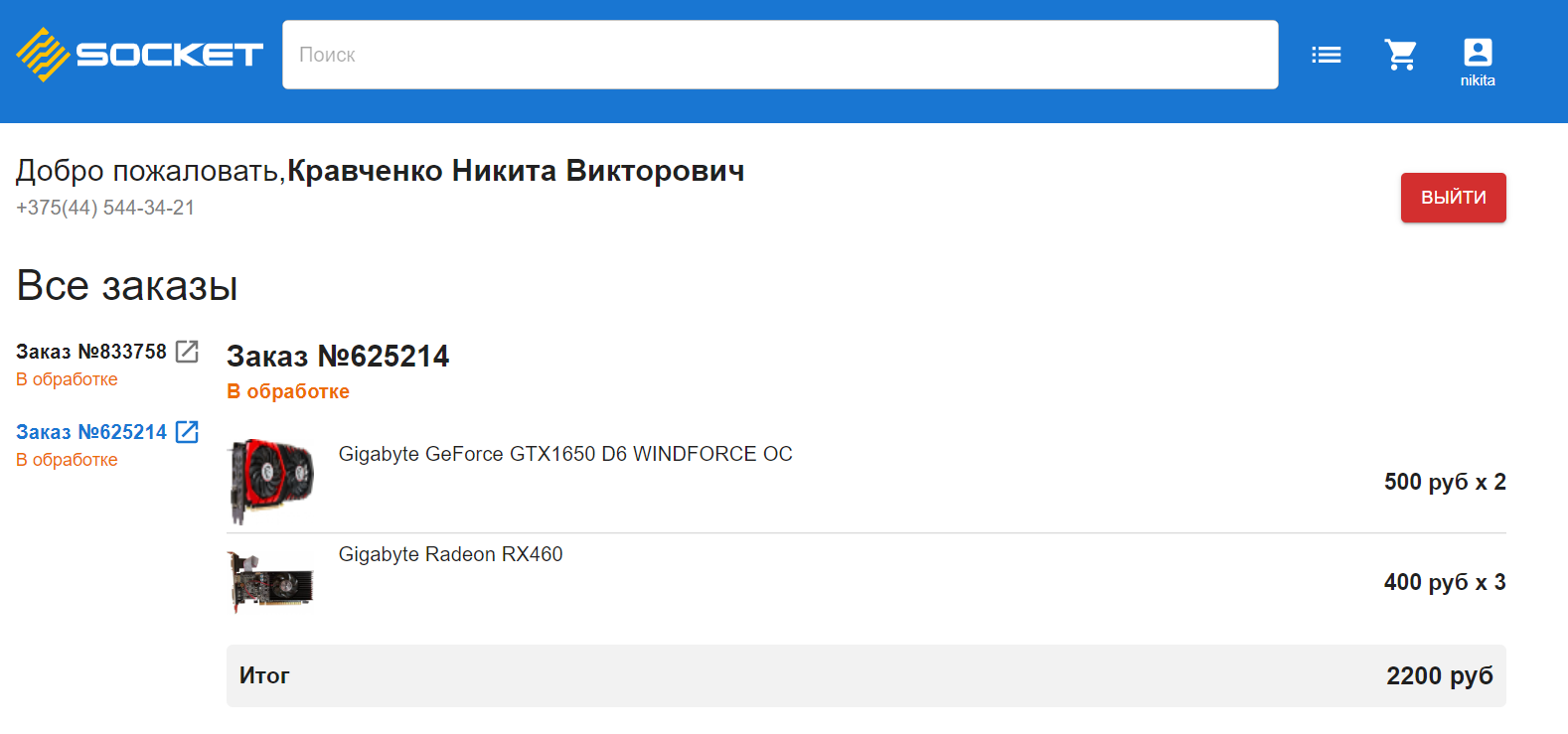


Рисунок 6.11 – Личный кабинет

Остается только кнопка выхода, находящаяся в верхнем правом углу личного кабинета пользователя. На этом функционал клиента логически завершен.

# Заключение

В результате курсовой работы было разработано Web-приложение «Магазин компьютерных комплектующих». Основой приложения стал язык Node.js. Серверная часть написана с помощью фреймворка Express, а клиентская с помощью фреймворка React. Обе технологии поддерживают асинхронное выполнение операций, что позволяет добиться высокой производительности приложения. В качестве базы данных выступила MySQL от компании Oracle.

Перед началом разработки был произведен аналитический обзор прототипов приложений подобной тематики и определение функциональных возможностей разрабатываемого приложения.

Программное средство выполняет следующие функции:

* имеет интуитивно понятный и просто интерфейс;
* обращается к базе данных, успешно извлекает и вносит новые данные, осуществляет поиск по базе данных;
* имеет различный функционал в зависимости от типа пользователя;
* быстро обрабатывает запросы;
* функционирует по протоколам https и wss;
* выглядит аккуратно, не содержит лишнюю информацию.

Разработанное программное средство удовлетворяет всем требованиям технического задания.

Приложение было успешно протестировано. В результате проведённых тестов ошибок обнаружено не было. Требования технического задания реализованы в полном объёме. Данный программный продукт является завершённым и готов к использованию.

# Список литературы

1. Node.js v11.15.0 Documentation [Электронный ресурс] / OpenJS Foundation. – Режим доступа: https://nodejs.org/docs/latest-v11.x/api. – Дата доступа:10.05.2022;
2. Express [Электронный ресурс] – Режим доступа: <https://expressjs.com/ru/> – Дата доступа: 10.05.2022;
3. React [Электронный ресурс] –Режим доступа: https://ru.reactjs.org/docs/getting-started.html – Дата доступа: 10.05.2022.
4. MySQL [Электронный ресурс] – Режим доступа: https://dev.mysql.com/doc/– Дата доступа: 10.05.2022.
5. Sequelize [Электронный ресурс] – Режим доступа: <https://sequelize.org>/ – Дата доступа: 10.05.2022.
6. Web-socket [Электронный ресурс] – Режим доступа: https://socket.io/docs/v4/ – Дата доступа: 10.05.2022.

# Приложение А

