Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Современные платформы программирования (СПП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Программное средство «Ez shop»

БГУИР КП I–40 01 01 28 ПЗ

Студент: гр. 751003 Стубеда В.Д.

Проверил: Видничук В.Н.

Минск, 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2020г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Стубеде Вадиму Дмитрипвичу* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Программное средство «Ez shop»*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи законченной работы *08.05.2020г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Языки прогрраммирования Javascript. Программная платформу Ndoe.js. Фреймворк Vue.js.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_*

*5 Руководство по установке и использованию программного средства\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Видничук В.Н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *08.02.2020г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования

(с обозначением сроков выполнения и процентом от общего обьема работы):

*Раздел 1. Введение к 20.03.2020г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 01.03.2020г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 20.03.2020г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 15.04.2020г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5. Заключение. Приложения к 20.04.2020г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 05.05.2020г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 04.05.2020г. по 15.05.2020г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Видничук В.Н.*

*(подпись)*

Задание принял к исполнению *Стубеда В.Д. 08.02.2020г.*

*(дата и подпись студента)*

Содержание

[Введение 5](#_Toc41014152)

[1 Анализ литературных источников 6](#_Toc41014153)

[1.1 Анализ существущих приложений. 15](#_Toc41014154)

[2 Постановка задачи 16](#_Toc41014155)

[3 Разработка программного средства 17](#_Toc41014156)

[3.1 Описание алгоритмов 17](#_Toc41014157)

[3.2 Структура данных 17](#_Toc41014158)

[3.2.1 Структура типов основного алгоритма 17](#_Toc41014159)

[3.2.2 Структура данных основного алгоритма 18](#_Toc41014160)

[3.2.3 Структура данных алгоритма get\_all\_products 18](#_Toc41014161)

[3.2.4 Структура данных алгоритма products\_delete\_product 19](#_Toc41014162)

[3.2.5 Структура данных алгоритма poroducts\_create\_product 19](#_Toc41014163)

[3.2.6 Структура данных алгоритма orders\_create\_order 20](#_Toc41014164)

[3.3 Схемы алгоритмов решения задачи 21](#_Toc41014165)

[4 Тестирование и проверка работоспособности программного средства 25](#_Toc41014166)

[5 Руководство по установке и использованию программного средства 28](#_Toc41014167)

[5.1 Шаг 1. Установка 28](#_Toc41014168)

[5.2 Шаг 2. Регистрация нового пользователя 28](#_Toc41014169)

[5.3 Шаг 3. Добавление заказа 29](#_Toc41014170)

[5.4 Шаг 4. Список заказов 29](#_Toc41014171)

[Заключение 31](#_Toc41014172)

[Список использованной литературы 32](#_Toc41014173)

[Приложение А 33](#_Toc41014174)

Введение

По мнению ведущих аналитических ресурсов [Data Insight](http://www.datainsight.ru/), [e-data.by](https://e-data.by/), [Hootsuite](https://hootsuite.com/) на сегодняшний день, ситуация на рынке интернет-торговли книгами в мире, Беларуси, России и других странах СНГ показывает следующие результаты.

Наблюдается стабильный рост электронной торговли, как в количестве онлайн-покупателей, так и в выручке и количестве заказов. Так за 2018 год число беларусов, совершающих покупки книг в интернете, выросло примерно до 45% (около 4,3 млн человек). При этом 16% совершают большинство покупок по Сети.

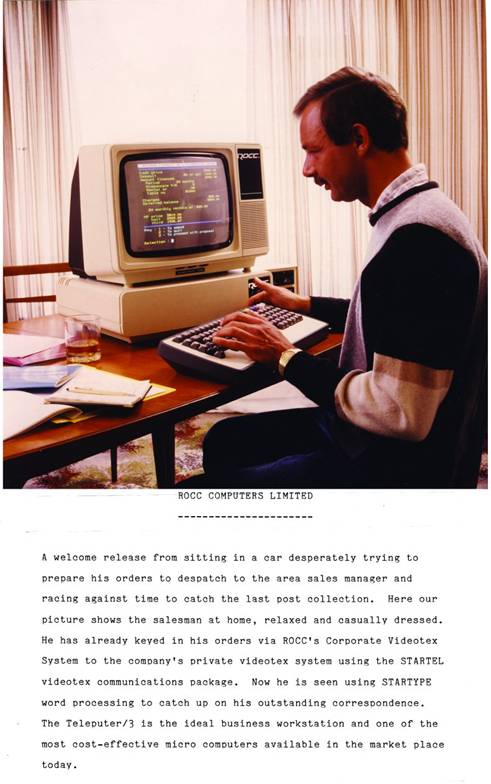
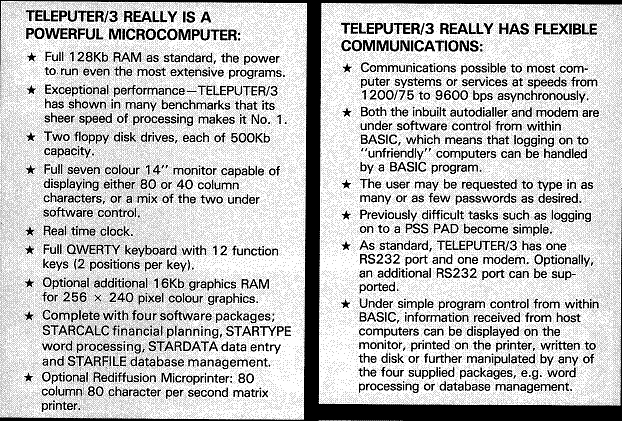
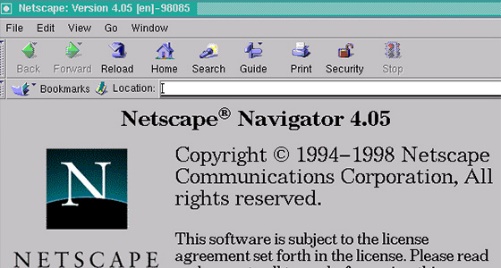
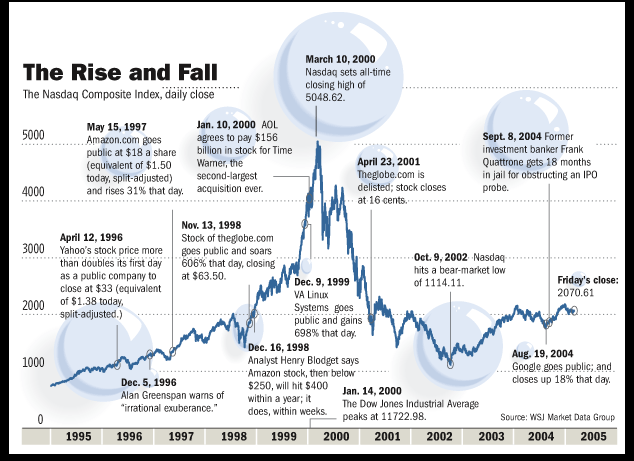
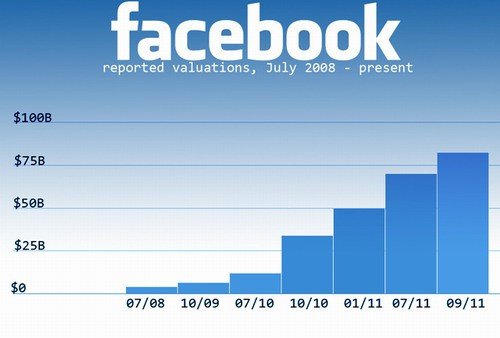
Интернет-магазины книг постепенно вытесняют оффлайн-магазины. Согласно данным опросов 45% респондентов покупают как онлайн, так и оффлайн, при этом четко зная, какой конкретно товар и интернет-сайт им необходим. Поисковыми системами Google и Яндекс для выбора необходимых книг, сравнения цен и условий покупки в различных интернет-магазинах пользуются порядка 31% беларуских пользователей Сети.

За последний год в странах СНГ кредит доверия покупателей к рынку онлайн-торговли вырос, что положительно сказалось на росте интернет-продаж. Это стало возможным благодаря расширению возможностей продавцов в плане доставки товаров в небольшие населенные пункты, появлению новых удобных способов оплаты покупок онлайн.

.

# Анализ литературных источников

**Интернет-магазин** (англ. *online shop или e-shop*) — сайт, торгующий товарами посредством сети Интернет. Позволяет пользователям онлайн, в своём браузере или через мобильное приложение, сформировать заказ на покупку, выбрать способ оплаты и доставки заказа, оплатить заказ. При этом продажа товаров осуществляется дистанционным способом и она накладывает ограничения на продаваемые товары. Так, в некоторых странах имеется запрет на интернет-торговлю алкоголем, оружием, ювелирными изделиями и другими товарами (к примеру, в России запрещена дистанционная продажа алкоголя, ювелирных изделий и других товаров, свободная реализация которых запрещена или ограничена).

В 1960 году компании American Airlines и IBM приступили к созданию системы автоматизации процедуры резервирования мест на авиарейсы – SABRE (Semi-Automatic Business Research Environment – полуавтоматическое оборудование для коммерческих исследований). Это был первый опыт создания системы электронной коммерции. Система SABRE сделала воздушные перелеты более доступными для рядовых граждан, помогая им ориентироваться в тарифах и рейсах, число которых постоянно росло. Эта система могла выполнять комплексное управление доходностью, позволяя авиакомпаниям добиваться максимальной прибыли за счет манипуляций с ценами с учетом наличия свободных мест. К 1964 году за один день система могла резервировать места для 26 тысяч пассажиров. Терминалы American Airlines, подключенные к SABRE по телефонным линиям, находились уже более чем в полусотне городов.  
  
По легенде, именно SABRE вдохновила Стенли Кубрика на его Космическую Одиссею 2001 – помните там операционисток на космической станции за терминалами? Система оказалась настолько эффективной, что ее потомки до сих пор используются по всему миру – например, в РЖД или Аэрофлоте. Но по-настоящему розничный электронный магазин появился почти двадцать лет позже. В 1979 году.  
  
  
  
Интернета еще не было, а покупать, не выходя из дома, уже хотелось. Особенно в дождливой Англии, где Майкл Олдрич особенно ненавидел шопинг и решил соединить телевизор, телефон и клавиатуру в почти персональный компьютер, чтобы раз и навсегда положить конец проблеме. Систему назвали VIDEOTEX.  
  
Простота и удобство интерфейса (Полный аналог BBS) обеспечили Видеотексу всемирную популярность. Но Олдрич не остановился на достигнутом и развил телевизионную приставку в полноценный аналог персоналки.  
  
  
  
Рекламная брошюра Teleputer3 производила неизгладимое впечатление для начала восьмидесятых: 128кб памяти, два пятидюймовых дисковода позволяли записывать до 500кб информации, а собственный монитор выводил целых 7 разных цветов. На борту была собственная ОС и набор офисных программ, от табличного процессора до баз данных. При желании можно было серьезно подтянуть и графику, со стандартных 80х40 символов дополнительная видеоплата поднимала изображение до фантастических 256х240 пикселей. Но настоящей гордостью системы, конечно, был скоростной модем на 9600 бит/с. И все это еще до появления IBM PC!  
  
  
  
Videotex имел такой успех у пользователей, что моментально получил признание по всему миру. Уже в 1983 году Videotex работал в Канаде и Австралии, но первые действительно большие деньги компании принес СССР – в 1981 году советское правительство заказывает 247 Телепутеров для обслуживания сибирского газопровода. Правительству система так понравилась, что после Сибири Телепутеры начинают массово производить по лицензии на новом заводе в Польше, сразу с кириллической клавиатурой – таким образом, именно Телепутер стал первым цветным советским ПК.  
  
  
  
А самыми благодарными пользователями были французы. Уже в 1980 французская компания Минитель фактически создает свой протоинтернет. В их модификации видеотекса абонент подключается к телеинформационной системе через терминал небольшого размера, с экраном и клавиатурой. Пользуясь им, можно получить около 13 000 разнообразных услуг, в частности, уточнить расписание пригородных поездов, узнать котировки акций, сделать заказ на определённые товары и услуги, пообщаться с другими пользователями Минителя в чате.  
  
Аппарат выдавался каждому жителю в почтовом отделении бесплатно. За пользование вносится ежемесячная абонентская плата, она зависит от типа аппарата и характера услуг. Спустя пару лет, в качестве эксперимента, руководство сети решило создать каждому пользователю персональный счет, чтобы он мог совершать покупки и оплачивать услуги прямо из своего терминала. И это нововведение обеспечило Минителю абсолютную популярность – фактически, у Франции был полноценный интернет уже в 1980-году, с сайтами, блогами, чатами и интернет-магазинами. Несмотря на ограниченные графические возможности, практически каждый «сайт» в сети отличался оригинальным оформлением.  
  
  
  
  
На своем пике, в 1997 году к минитель одновременно подключались шесть миллионов терминалов, они совершали транзакций на $750 миллионов. Конечно, систему сильно подкосил интернет, но даже в 2009 году за месяц минител регистрировала 10 миллионов подключений к системе. Можете себе представить, сколько негодования вызывает решение Франц Телеком полностью остановить систему 30 июня 2012 года.  
  
Чтобы показать, насколько Минитель опередил свое время, [энтузиасты из США](http://www.minitel.us/) даже смогли присоединить к нему веб-камеру и выйти в скайп:  
  
  
  
В 1990 Тим Бернерс-Ли создает интернет, навсегда решая аппаратную проблему интернет-шоппинга. В отличие от Minitel и Videotex, интернет – это открытая сеть, здесь нет одной компании, контролирующей контент и сами терминалы. Теперь любой пользователь может создавать свои сайты, не спрашивая разрешения у владельца сети.  
  
  
  
В 1994 году Netscape выпускает первый коммерческий браузер Netscape Navigator и добавляет к интернету последний штрих, создав протокол SSL – теперь все покупки и транзакции могут быть надежно защищены шифрованием. Пицца Хат тут же начинает масштабную рекламную кампанию, первой запустив полноценный интернет-магазин. Заказ пиццы по интернету стал сенсацией, но так и не прижился – все же еду оказалось проще заказать по телефону.  
  
  
  
И только в 1995 году начинается интернет-торговля, какой мы ее знаем. Стартует Amazon.com.  
  
  
  
В 1999 году выходит Nokia 7110 – рекламировавшийся как первый телефон с доступом в интернет (что не совсем так), он так же становится одним из символов поп-культуры после того, как его старший брат nokia 8110 попадает в фильм Матрица.  
  
А 10 марта 2000 года наступил «кризис доткомов». Мыльный пузырь доткомов лопается, и многие игроки вынуждены покинуть рынок. Индекс Nasdaq падает в пять раз. Но некоторые интернет-магазины извлекают прибыль и из этой ситуации, начав делать огромные скидки и сбивать цены, Amazon и Ebay не только переживают кризис, но и увеличивают долю.  
  
  
  
Хотя капитализация компании Amazon Inc снизилась практически до первоначального уровня, объем продаж не уменьшился. Выручка к 2008 году составила $15 миллиардов и $476 миллионов чистой прибыли. Неплохо для безумного стартапа одного программиста, который чудом нашел инвестиций на $300 тысяч, потому что умел предсказывать будущее розничной торговли.  
  
К 2011 году индустрия онлайн-шоппинга подошла к концу очередного этапа в развитии. Магазины испробовали все средства конкуренции, и победить сможет лишь радикально инновационное решение. Пользователи больше не ходят на Амазон, чтобы приобрести mp3 или скачать фильм – для этого у них на компьютере уже стоит iTunes и uTorrent.  
  
  
  
Сегодня перед покупкой мы наводим справки о товаре в Гугле и спрашиваем мнение наших друзей в социальных сетях. А компании больше не соревнуются друг с другом за внимание аудитории, теперь их задача быть популярнее забавных котят на фейсбуке. В таких условиях победить может только компания, идущая на шаг впереди конкурентов и стоящая на шаг ближе к пользователям. Как в свое время делал Videotex.  
  
В 2011 году 44% крупных онлайн-магазинов инвестировали в мобильные приложения, чтобы быть ближе к пользователям. Магазины, работающие с социальными сетями, испытывают небывалый рост, только Groupon за год вырос на 468% — желание быть ближе к клиентам это основной тренд в онлайн-шоппинге. Сама индустрия е-коммерции растет быстрее всех остальных розничных секторов в мире, с 4,3% в 2004 до 8,6% в 2011. Но и это не главное – сами магазины вытесняются социальными сетями, только один Facebook планирует к 2015 году занимать 50% розничной торговли.

## Анализ существущих приложений.

Существует множество подобных приложений. На сегодняшний день самым популярным из них в Беларуси является “Oz.by”. Он был создан в 2010 году и сейчас насчитывает более двух миллионов пользователей.



Рисунок 1.1 – Интерфейс сайта “Oz.by”

Также достаточно популярным является сайт “biblio.by”.

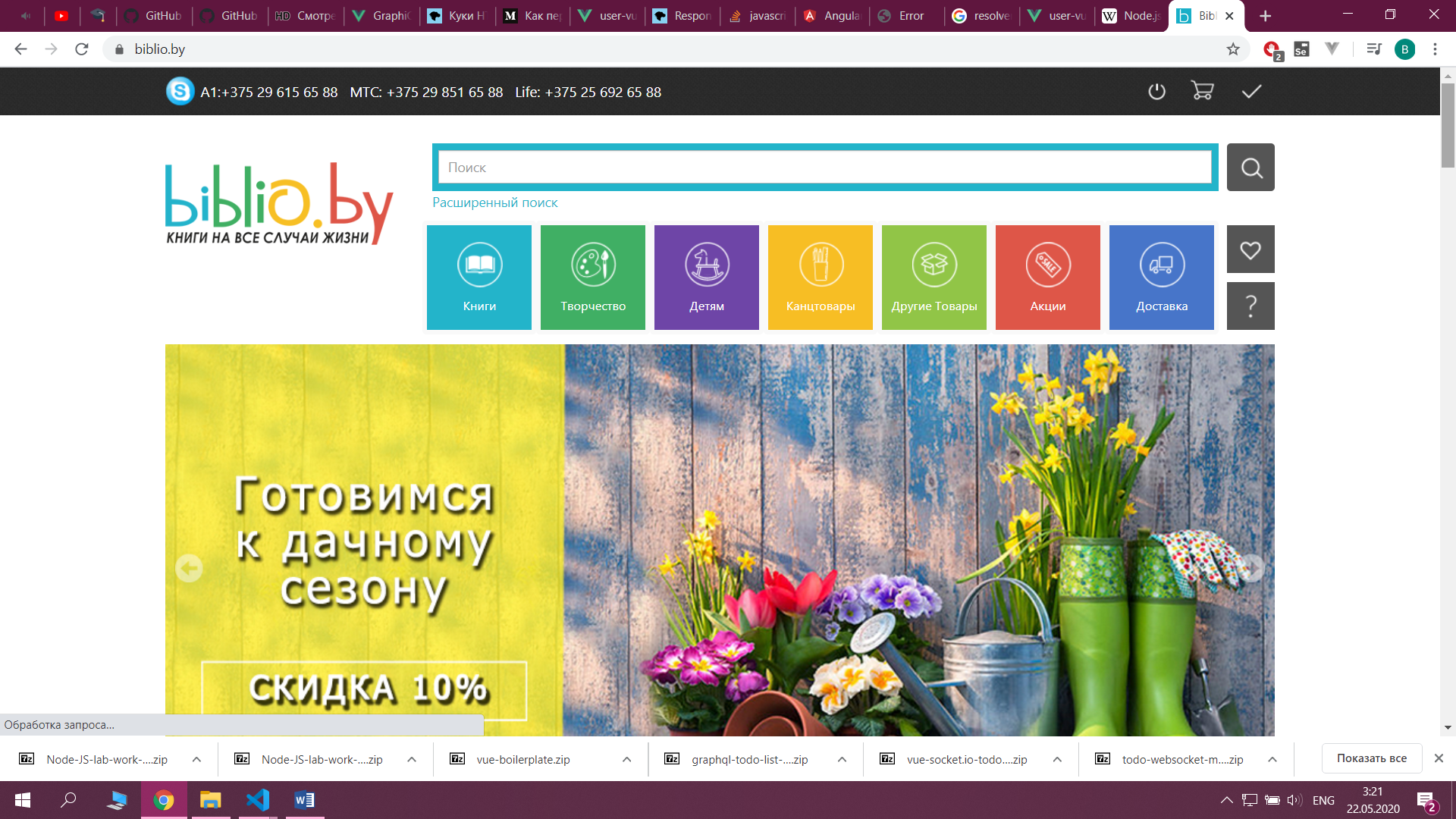


Рисунок 1.1 – Интерфейс приложения “biblio.by”

# 2 Постановка задачи

Разработать веб-сайт “Ez shop”. Необходимо реализовать отображение каталога товаров в бразуере, сохранение данных в базу данных, авторизацию с помощью JWT токена.

При входе на сайт пользователь может просматривать продукты для. Чтобы расширить свои возможности пользователь может зарегистрировать новый аккаунт, или авторизироваться, если у него уже есть аккаунт. После авторизации ему станет доступно делать заказы, просматривать их и. Добавить возможность авторизации в роли администратора. Администратор может смотреть информацию о всех заказах.

Реализовать приложение в виде двух отдельных сервисов: backend, frontend. Frontend служит для отрисовки интерфейса в браузере и шлёт запросы для получения информации к backend. Backend в свою очередь проверяет, авторизован ли пользователь и взаимодействует с базой данных и возвращает результат к frontend. Использовать архитектуру REST API.

# 3 Разработка программного средства

## 3.1 Описание алгоритмов

Таблица 1 – Описание основной программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование алгоритма | Назначение алгоритма | Формальные параметры |
| 1. | check-auth | Проверка токена пользователя | email – email пользователя,  password – пароль пользователя |
| 2. | get\_all\_products | Получить все товары каталога | order – отдельный заказ, отбражаемый на странице |
| 3. | delete\_order | Удаление заказа по его id | \_id – идентификатор заказа, productId – идентификтора продукта |
| 4. | post\_create\_product | Добавление нового товара | productImage –фотография к товару, name – наименование товара, \_id – идентификатор продукта |
| 5. | post\_user\_login | Авторизация пользователя | email – email пользователя,  password – пароль пользователя |

## 3.2 Структура данных

### 3.2.1 Структура типов основного алгоритма

Таблица 2 – Структура типов основного алгоритма

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| UserData | {  “email”,  “password”  } | Хранение данных для аутентификации пользователя |

### 3.2.2 Структура данных основного алгоритма

Таблица 3 – Структура данных основного алгоритма

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| token | String | Сгенерированный токен | Локальная переменная |
| UserData | UserData | Хранение логина и пароля пользователя | Формальный параметр |

### 3.2.3 Структура данных алгоритма get\_all\_products

Таблица 4 – Структура типов алгоритма get\_all\_products

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| product | {  “\_id”,  “name”,  “price”,  “productImage” } | Хранение данных о товаре |

Таблица 5 – Структура данных алгоритма get\_all\_products

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| products | product[] | Массив товаров, который будет возвращён в результате | Локальная переменная |
| orders | order | Заказы пользователей | Формальный параметр |

### 3.2.4 Структура данных алгоритма products\_delete\_product

Таблица 5 – Структура данных алгоритма products\_delete\_product

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| products | product[] | Массив товаров, который будет возвращён в результате | Локальная переменная |
| orders | order | Заказы пользователей | Формальный параметр |
| \_id | String | Идентификатор удаляемого заказа | Локальная переменная |

### 3.2.5 Структура данных алгоритма poroducts\_create\_product

Таблица 6 – Структура данных алгоритма products\_create\_product

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| name | String | Наименование товара | Формальный параметр |
| price | String | Цена одной единицы товара | Формальный параметр |
| productImage | File | Изображение товара | Формальный параметр |

### 3.2.6 Структура данных алгоритма orders\_create\_order

Таблица 7 – Структура данных алгоритма orders\_create\_order

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| productId | Integer | Порядковый номер товара | Формальный параметр |
| quantity | Integer | Количество товара в заказе | Локальная переменная |
| \_id | String | Идентификатор заказа | Локальная переменная |

## 3.3 Схемы алгоритмов решения задачи



Рисунок 3.1 – Схема алгоритма получения товаров



Рисунок 3.2 – Схема алгоритма удаления товара



Рисунок 3.3 – Схема алгоритма products\_create\_product



Рисунок 3.4 – Схема алгоритма добавления заказа

# 4 Тестирование и проверка работоспособности программного средства

В ходе тестирования были рассмотрены в действии все функциональные возможности сайта “Ez shop”. Было изучено поведение приложения при различных действиях пользователя.

Таблица 9 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Авторизация с правильными данными |
| Ожидаемый результат: | Сообщение о успешной авторизации |
| Фактический результат: |  |

Таблица 10 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация | Регистрация пользователя |
| Ожидаемый результат: | Сообщение об успешной регистрации нового пользователя |
| Фактический результат: |  |

Таблица 11 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация: | Добавление нового заказа |
| Ожидаемый результат: | Новый заказ появился в списке заказов |
| Фактический результат: |  |

Таблица 12 – Тест 4

|  |  |
| --- | --- |
| Тестовая ситуация: | Создание существующего аккаунта |
| Ожидаемый результат: | Сообщение о занятом email |
| Фактический результат: |  |

# 5 Руководство по установке и использованию программного средства

## 5.1 Шаг 1. Установка

Для запуска приложения необходимо наличиеподключения к интернету для доступа к MongoDB, запустить backend и frontend. После запуска всех сервисов открыть браузер и перейти по адресу localhost:8080. Появится главная страница стайта со списком товаров.

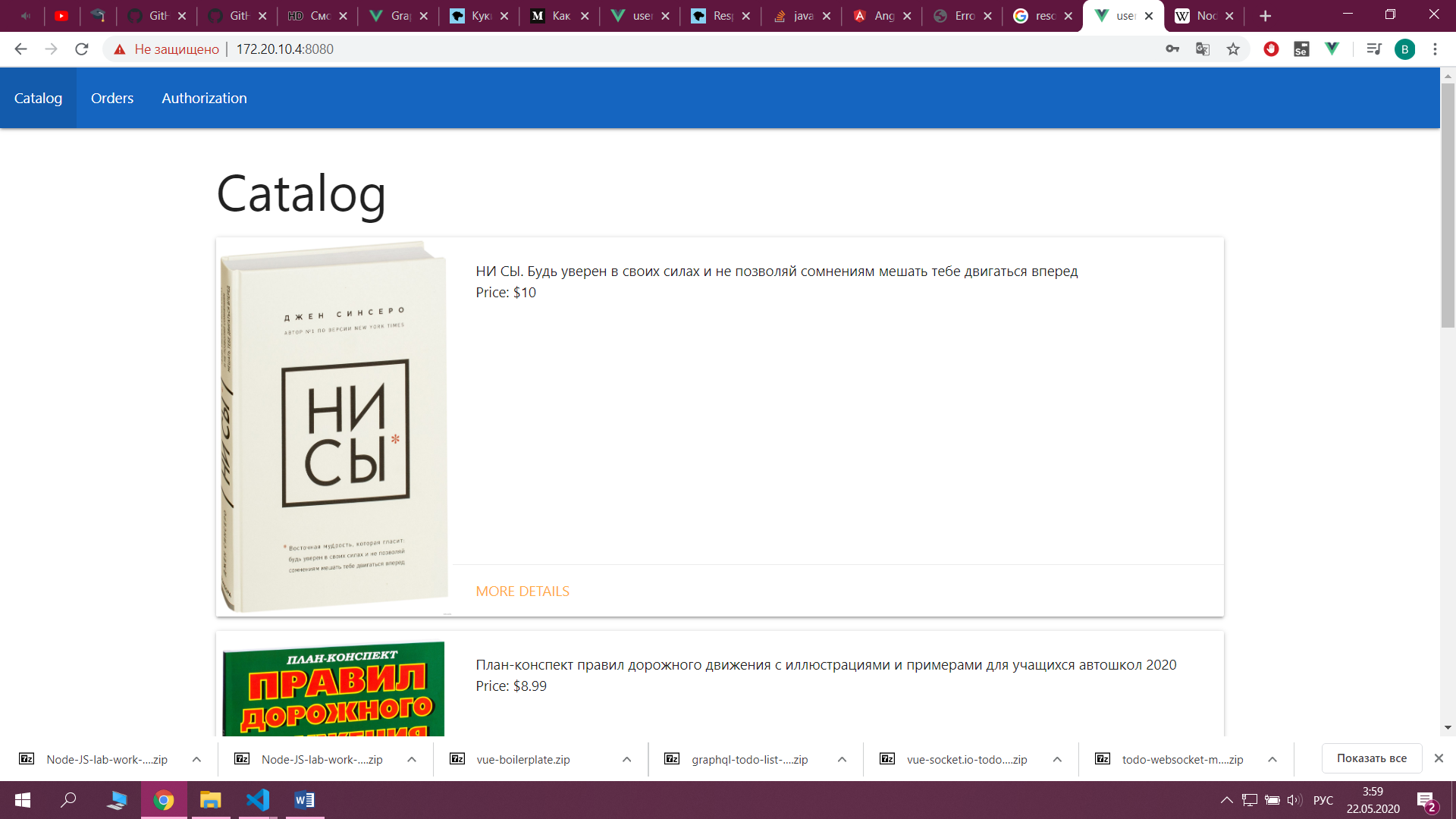


Рисунок 5.1 – Главная страница сайта

## 5.2 Шаг 2. Регистрация нового пользователя

После нажатия на кнопку “Authorization” будет открыта страница с формой для регистрации. Там пользователь может ввсети свои личные данные и зарегистрировать новый аккаунт.

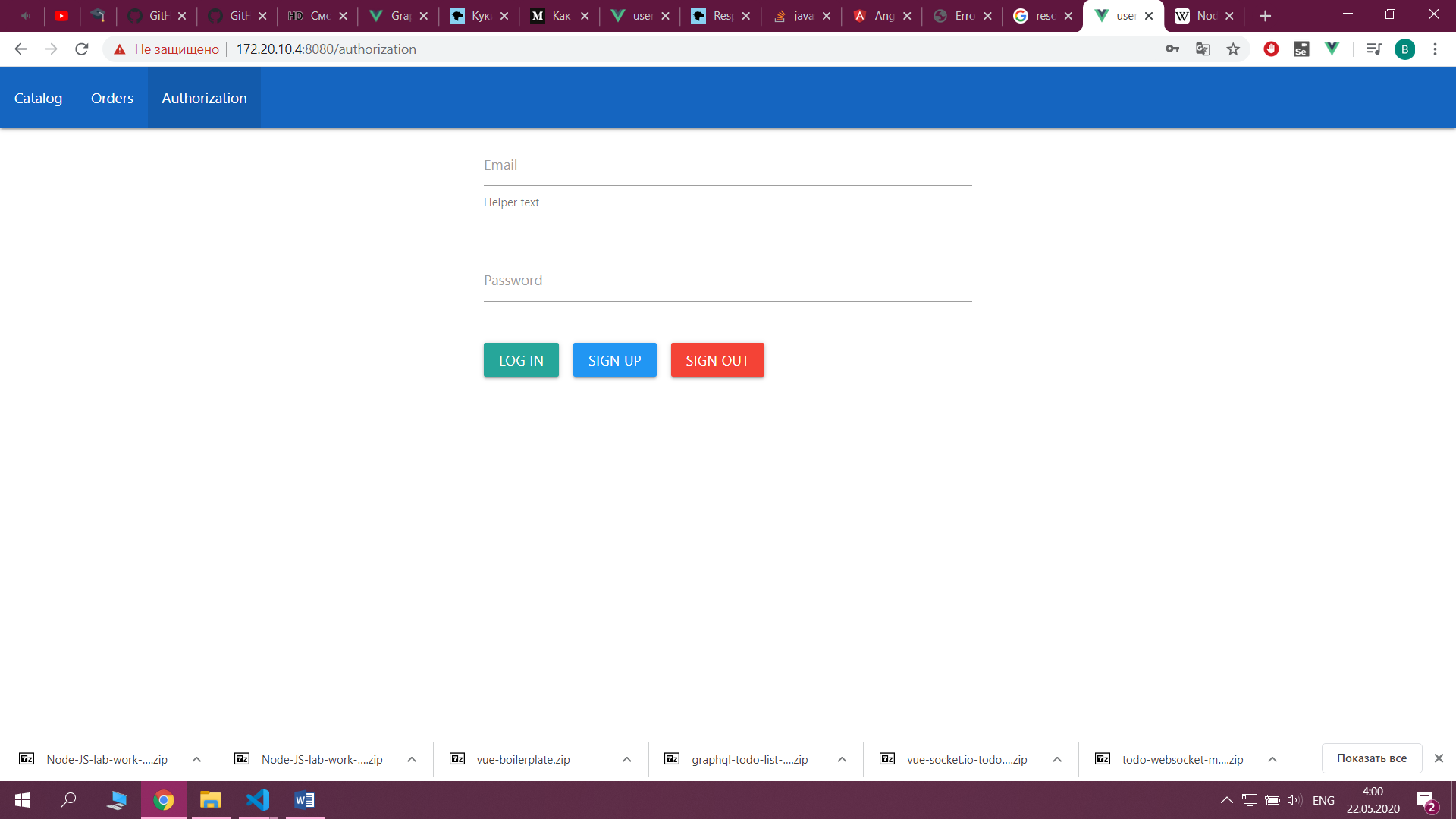


Рисунок 5.2 – Страница регистрации

## 5.3 Шаг 3. Добавление заказа

После регистрации и авторизации пользователь может сделать заказ. Для этого необходимо нажать на кнопку “Catalog”. Откроется каталог товаров. Там нужно выбрать товар и нажать на нем на кнопку “More details”, после чего будет открыта страница подробностей о товаре. На странице добавления заказа пользователь может выбрать количество нужного товара. Для создания заказа необходимо нажать на кнопку “Buy”.

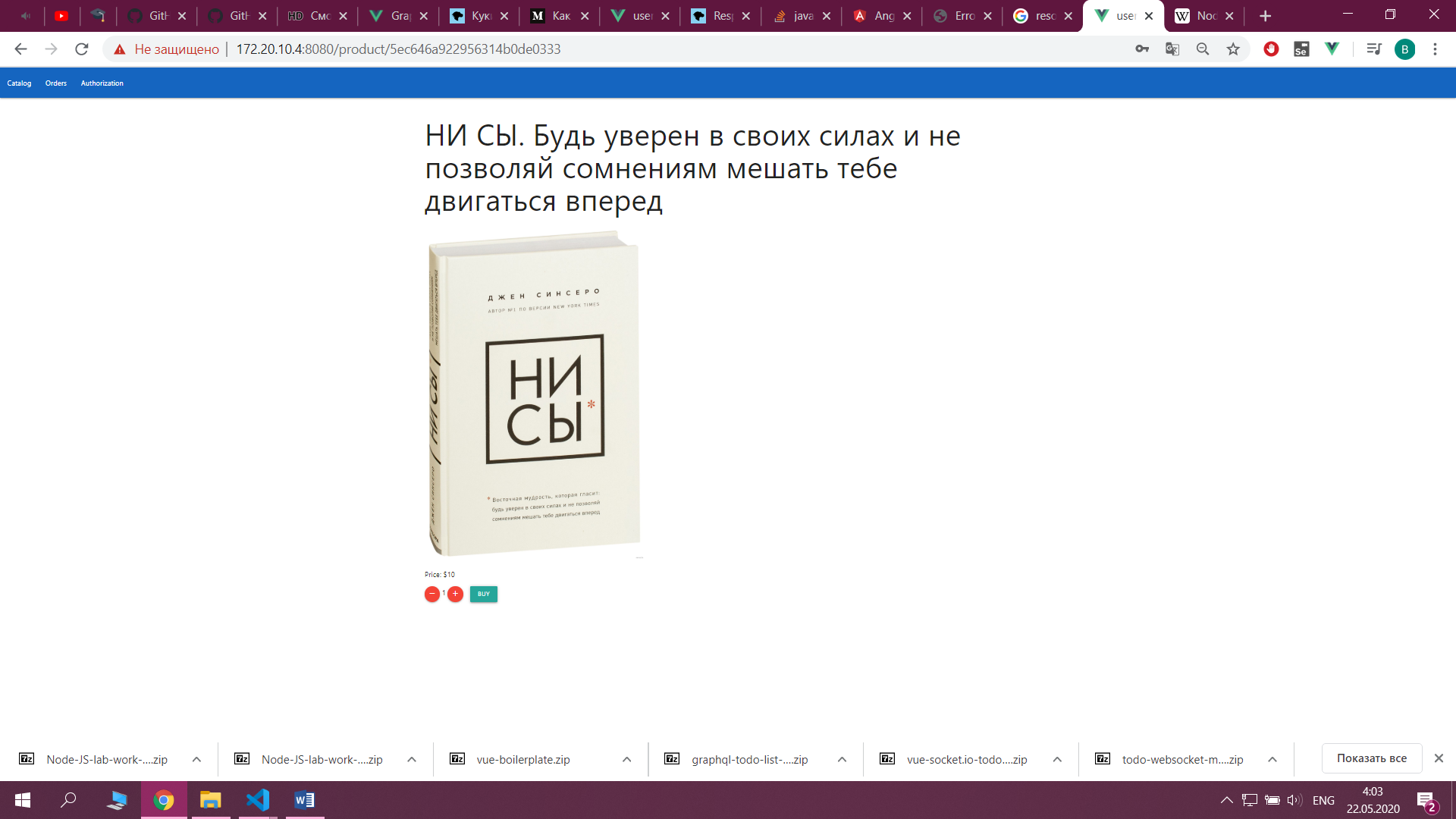


Рисунок 5.3 – Страница добавления заказа

## 5.4 Шаг 4. Список заказов

Авторизованный пользователь может просматривать заказы. Для этого необходимо открыть страницу “Orders” приложения. Также пользователь может удалять заказы. Для этого необходимо выбрать заказ и нажать на кнопку “Delete”.

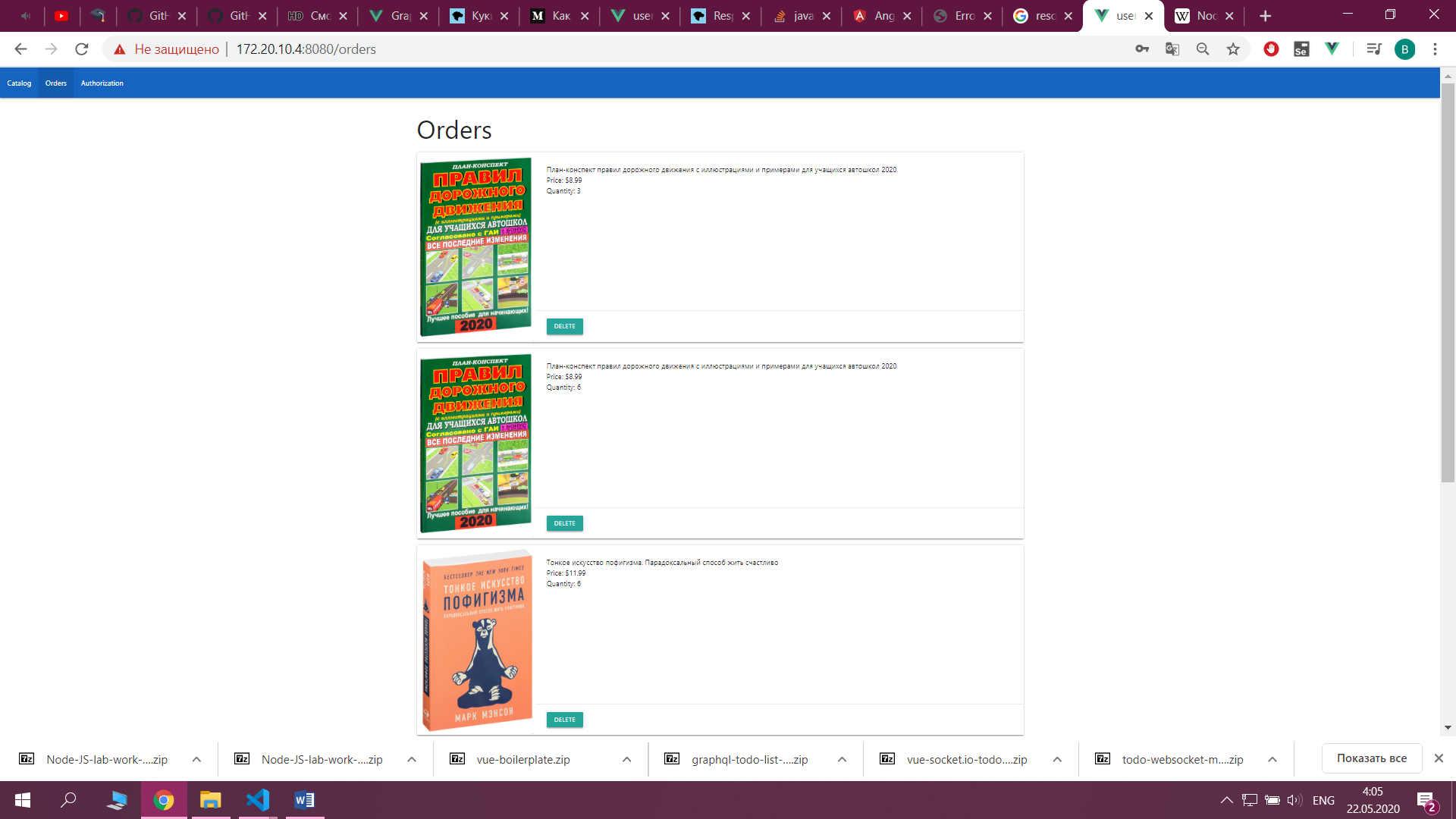


Рисунок 5.4 – Список заказов

Заключение

В ходе выполнения курсовой работы мною было разработано веб-приложение “Ez shop”. В процессе разработки испльзовались языки программирования JavaScript, HTML и CSS, а также фреймворк Vue. В качестве СУБД использовалсь MongoDb. Приложение создано в соответствии с архитектурой Rest API.

В разработанном приложении пользователь может зарегистрироваться и затем авторизироваться. Авторизация реализована с помощью JWT токенов. В случае успешной авторизации сервер генерирует токени отправляет пользователю. Пользователь в свою очередь сохраняет полученный токен и добавляет его в качестве заголовка авторизации в каждый запрос на сервер. Также пользователь может добавлять свои заказы. Если у пользователя роль “Администратор”, то ему доступны расширенные возможности. Администраторы могут удалять любые заказы, просматривать информацию о пользователях и редактировать товары.

Итогами выполнения данной работы является опыт разработки на языках программировани JavaScript, опыт работы с базами данных, изучение и применение архитектуры Rest API.

Список использованной литературы

[1] В. Салд – JavaScript. Руководство для начинающих, 2019г. – 816с.

[2] Ф. Окель – Язык JavaScript , 2019г. – 1168с.

[3] MongoDB Framework Documentation [Электронный ресурс] – Режим доступа: https://docs.mongodb/mongodb/docs/current/mongodb-framework-reference/index.html – Дата доступа: 10.03.2020.

[4] Vue Documentation [Электронный ресурс] – Режим доступа: https://vue.io/docs. – Дата доступа: 20.03.2020.

Приложение А

(обязательное)

Исходный код программы

controllers/orders.js

const mongoose = require('mongoose');

const Order = require('../models/order');

const Product = require('../models/product');

exports.orders\_get\_all = (req, res, next) => {

    Order

        .find()

        .select('product quantity \_id')

        .populate('product', 'name productImage price')

        .exec()

        .then(docs => {

            res.status(200).json({

                count: docs.length,

                orders: docs.map(doc => {

                    return {

                        \_id: doc.\_id,

                        product: doc.product,

                        quantity: doc.quantity,

                        request: {

                            type: 'GET',

                            url: 'http://localhost:3000/orders/' + doc.\_id

                        }

                    }

                })

            });

        })

        .catch(err => {

            res.status(500).json({

                error: err

            });

        });

}

exports.orders\_create\_order = (req, res, next) => {

    Product

        .findById(req.body.productId)

        .then(product => {

            if (!product) {

                return res.status(404).json({

                    message: "Product not found"

                });

            };

            const order = new Order({

                \_id: mongoose.Types.ObjectId(),

                quantity: req.body.quantity,

                product: req.body.productId

            });

            return order.save()

        })

        .then(result => {

            console.log(result);

            res.status(201).json({

                message: 'Order stored',

                createdOrder: {

                    \_id: result.\_id,

                    product: result.product,

                    quantity: result.quantity

                },

                request: {

                    type: 'GET',

                    url: 'http://localhost:3000/orders/' + result.\_id

                }

            });

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({

                error: err

            });

        });

}

exports.orders\_get\_order = (req, res, next) => {

    Order

        .findById(req.params.orderId)

        .populate('product')

        .exec()

        .then(order => {

            if (!order) {

                return res.status(404).json({

                    message: 'Order not found'

                });

            };

            res.status(200).json({

                order: order,

                request: {

                    type: 'GET',

                    url: 'http://localhost:3000/orders/'

                }

            });

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({

                error: err

            });

        });

}

exports.orders\_delete\_order = (req, res, next) => {

    Order

        .remove({ \_id: req.params.orderId })

        .exec()

        .then(order => {

            res.status(200).json({

                message: 'Order deleted',

                request: {

                    type: 'GET',

                    url: 'http://localhost:3000/orders/',

                    body: { productId: 'ID', quantity: 'Number' }

                }

            });

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({

                error: err

            });

        });

}

controllers/products.js

const Product = require('../models/product');

const mongoose = require('mongoose');

exports.products\_get\_all = (req, res, next) => {

    Product.find()

    .select('name price \_id productImage')

        .exec()

        .then(docs => {

            const response = {

                count: docs.length,

                products: docs.map(doc => {

                    return {

                        name: doc.name,

                        price: doc.price,

                        productImage: doc.productImage,

                        \_id: doc.\_id,

                        request: {

                            type: 'GET',

                            url: 'http://localhost:3000/products/' + doc.\_id

                        }

                    }

                })

            };

            //if (docs.length >= 0) {

                res.cookie('token', 'token', {

                    maxAge: 86\_400\_000,

                    httpOnly: true

                    });

                res.status(200).json(response);

            //} else {

            //    res.status(404).json({

            //        message: 'No entries found'

            //    });

            //

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({

                error: err

            });

        });

}

exports.products\_create\_product = (req, res, next) => {

    const product = new Product({

        \_id: new mongoose.Types.ObjectId(),

        name: req.body.name,

        price: req.body.price,

        productImage: 'http://localhost:3000/' + req.file.path.replace('\\','/')

    });

    product

        .save()

        .then(result =>  {

            console.log(result);

            res.status(201).json({

                message: 'Created product successfully',

                createdProduct: {

                    name: result.name,

                    price: result.price,

                    productImage: result.productImage,

                    \_id: result.\_id,

                    request: {

                        type: 'GET',

                        url: 'http://localhost:3000/products/' + result.\_id

                    }

                }

            });

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({

                error: err

            });

        });

}

exports.products\_get\_product = (req, res, next) => {

    const id = req.params.productId;

    Product.findById(id)

        .select('name price \_id productImage')

        .exec()

        .then(doc => {

            console.log("From database", doc);

            if (doc) {

                res.status(200).json({

                    product: doc,

                    request: {

                        type: 'GET',

                        url: 'http://localhost:3000/products'

                    }

                });

            } else {

                res.status(404).json({message: 'No valid entry found for provided ID'});

            }

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({error: err});

        });

}

exports.products\_update\_product = (req, res, next) => {

    const id = req.params.productId;

    const updateOps = {};

    for (const ops of req.body) {

        updateOps[ops.propName] = ops.value;

    }

    Product.update({ \_id: id }, { $set: updateOps })

        .exec()

        .then(result => {

            res.status(200).json({

                message: 'Product updated',

                request: {

                    type: 'GET',

                    url: 'http://localhost:3000/products/' + id

                }

            });

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({

                error: err

            });

        });

}

exports.products\_delete\_product = (req, res, next) => {

    const id = req.params.productId;

    Product.remove({ \_id: id })

        .exec()

        .then(result => {

            res.status(200).json({

                message: 'Product deleted',

                request: {

                    type: 'POST',

                    url: 'http://localhost:3000/products',

                    body: { name: 'String', price: 'Number' }

                }

            });

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({

                error: err

            });

        });

}

controllers/user.js

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

const User = require('../models/user')

exports.user\_signup = (req, res, next) => {

    User.find({ email: req.body.email })

    .exec()

    .then(user => {

        if (user.length >= 1) {

            return res.status(409).json({

                message: 'Mail exists'

            });

        } else {

            bcrypt.hash(req.body.password, 10, (err, hash) => {

                if (err) {

                    return res.status(500).json({

                        error: err

                    });

                } else {

                    const user = new User({

                        \_id: new mongoose.Types.ObjectId(),

                        email: req.body.email,

                        password: hash

                        });

                        user

                            .save()

                            .then(result => {

                                console.log(result);

                                res.status(201).json({

                                    message: 'User created'

                                });

                            })

                            .catch(err => {

                                console.log(err);

                                res.status(500).json({

                                    error: err

                                });

                            });

                    }

                });

        }

    })

}

exports.user\_login = (req, res, next) => {

    User

        .find({ email: req.body.email })

        .exec()

        .then(user => {

            if (user.length < 1) {

                return res.status(404).json({

                    message: 'Auth failed'

                });

            }

            bcrypt.compare(req.body.password, user[0].password, (err, result) => {

                if (err) {

                    return res.status(404).json({

                        message: 'Auth failed'

                    });

                }

                if (result) {

                    const token = jwt.sign(

                        {

                            email: user[0].email,

                            userId: user[0].\_id

                        },

                        process.env.JWT\_KEY,

                        {

                            expiresIn: "1h"

                        }

                    );

                    return res.status(200).json({

                        message: 'Auth successful',

                        token: token

                    });

                }

                res.status(401).json({

                    message: 'Auth failed'

                });

            });

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({

                error: err

            });

        });

}

exports.user\_delete = (req, res, next) => {

    User.remove({ email: req.body.email})

        .exec()

        .then(result => {

            res.status(200).json({

                message: "User deleted"

            });

        })

        .catch(err => {

            console.log(err);

            res.status(500).json({

                error: err

            });

        });

}

middleware/check-auth.js

const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {

    try {

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

        const decoded = jwt.verify(token, process.env.JWT\_KEY);

        req.userData = decoded;

        next();

    } catch (error) {

        return res.status(401).json({

            message: 'Auth failed'

        });

    }

};

models/orders.js

const mongoose = require('mongoose');

const orderSchema = mongoose.Schema({

    \_id: mongoose.Schema.Types.ObjectId,

    product: { type: mongoose.Schema.Types.ObjectId, ref: 'Product', required: true },

    quantity: { type: Number, default: 1 }

});

module.exports = mongoose.model('Order', orderSchema);

models/product.js

const mongoose = require('mongoose');

const productSchema = mongoose.Schema({

    \_id: mongoose.Schema.Types.ObjectId,

    name: { type: String, required: true },

    price: { type: Number, required: true },

    productImage: { type: String, required: true }

});

module.exports = mongoose.model('Product', productSchema);

models/user.js

const mongoose = require('mongoose');

const userSchema = mongoose.Schema({

    \_id: mongoose.Schema.Types.ObjectId,

    email: {

        type: String,

        required: true,

        unique: true,

        match: /^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$/

    },

    password: { type: String, required: true }

});

module.exports = mongoose.model('User', userSchema);

}

routes/orders.js

const express = require('express');

const router = express.Router();

const checkAuth = require('../middleware/check-auth');

const OrdersController = require('../controllers/orders');

// Handle incoming GET requests to /orders

router.get('/', checkAuth,  OrdersController.orders\_get\_all);

router.post('/', checkAuth, OrdersController.orders\_create\_order);

router.get('/:orderId', checkAuth, OrdersController.orders\_get\_order);

router.delete('/:orderId', checkAuth, OrdersController.orders\_delete\_order);

module.exports = router;

routes/product.js

const express = require('express');

const router = express.Router();

const multer = require('multer');

const checkAuth = require('../middleware/check-auth');

const ProductController = require('../controllers/products');

const storage = multer.diskStorage({

    destination: function(req, file, cb) {

        cb(null, './uploads/');

    },

    filename: function(req, file, cb) {

        cb(null, file.originalname);

    }

});

const fileFilter = (req, file, cb) => {

    // reject a file

    if (file.mimetype === 'image/jpeg' || file.mimetype === 'image/png') {

        cb(null, true);

    } else {

        cb(null, false);

    }

}

const upload = multer({

    storage: storage,

    limits: {

        fileSize: 1024 \* 1024 \* 5

    },

    fileFilter: fileFilter

});

router.get('/', ProductController.products\_get\_all);

router.post('/', checkAuth, upload.single('productImage'), ProductController.products\_create\_product);

router.get('/:productId', ProductController.products\_get\_product);

router.patch('/:productId', checkAuth, ProductController.products\_update\_product);

router.delete('/:productId', checkAuth, ProductController.products\_delete\_product);

module.exports = router;

routes/user.js

const express = require('express');

const router = express.Router();

const UserController = require('../controllers/user');

const checkAuth = require('../middleware/check-auth');

router.post('/signup', UserController.user\_signup);

router.post('/login', UserController.user\_login);

router.delete('/:email', checkAuth, UserController.user\_delete);

module.exports = router;

server.js

const http = require('http');

const app = require('./app');

const port = process.env.PORT || 3000;

const server = http.createServer(app);

server.listen(port);

app.js

const express = require('express');

const app = express();

const morgan = require('morgan');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const cookieParser = require('cookie-parser');

mongoose.set('useCreateIndex', true);

const productRoutes = require('./api/routes/products');

const orderRoutes = require('./api/routes/orders');

const userRoutes = require('./api/routes/user');

mongoose.connect(

    "mongodb+srv://CTY6A:" +

      process.env.MONGO\_ATLAS\_PW +

      "@node-rest-shop-ez-beyb5.mongodb.net/test?retryWrites=true&w=majority",

    {

        useNewUrlParser: true,

        useUnifiedTopology: true

    }

);

mongoose.Promise = global.Promise;

app.use(cookieParser());

app.use(morgan('dev'));

app.use('/uploads', express.static('uploads'));

app.use(bodyParser.urlencoded({extended: false}));

app.use(bodyParser.json());

app.use((req, res, next) => {

    res.header("Access-Control-Allow-Origin", "\*");

    res.header(

        "Access-Control-Allow-Headers",

        "Origin, X-Requested-With, Content-Type, Accept, Authorization"

    );

    if (req.method === 'OPTIONS') {

        res.header("Access-Control-Allow-Methods", "PUT, POST, PATCH, DELETE, GET");

        return res.status(200).json({});

    }

    next();

});

app.use ((req, res, next) => {

    res.cookie('JWT', 'JWT', {

        maxAge: 900000

    });

    next();

});

// Routes which should handle requests

app.use('/products', productRoutes);

app.use('/orders', orderRoutes);

app.use('/user', userRoutes);

app.use((req, res, next) => {

    const error = new Error('Not found');

    error.status = 404;

    next(error);

})

app.use((error, req, res, next) => {

    res.status(error.status || 500);

    res.json({

        error: {

            message: error.message

        }

    });

});

module.exports = app;

components/Navbar.vue

<template>

    <div>

        <nav class="blue darken-3">

            <div class="nav-wrapper">

            <ul>

                <router-link

                    tag="li"

                    to="/"

                    exact

                    active-class="active"

                >

                    <a href="#">Catalog</a>

                </router-link>

                <router-link

                    tag="li"

                    to="/orders"

                    active-class="active"

                >

                    <a href="#">Orders</a>

                </router-link>

                <router-link

                    tag="li"

                    to="/authorization"

                    active-class="active"

                >

                    <a href="#">Authorization</a>

                </router-link>

            </ul>

            </div>

        </nav>

    </div>

</template>

<script>

export default {

}

</script>

<style lang="scss" scoped>

</style>

router/index.js

import Vue from 'vue'

import VueRouter from 'vue-router'

Vue.use(VueRouter)

  const routes = [

  {

    path: '/' || '/products',

    name: 'Catalog',

    component: () => import('../views/Catalog.vue')

  },

  {

    path: '/orders',

    name: 'Orders',

    component: () => import('../views/Orders.vue')

  },

  {

    path: '/product/:id',

    name: 'Product',

    component: () => import('../views/Product.vue')

  },

  {

    path: '/authorization',

    name: 'Authorization',

    component: () => import('../views/Authorization.vue')

  }

]

const router = new VueRouter({

  mode: 'history',

  base: process.env.BASE\_URL,

  routes

})

export default router

store/index.js

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

export default new Vuex.Store({

  state: {

    tasks: JSON.parse(localStorage.getItem('tasks') || '[]').map(task => {

      if (new Date(task.date) < new Date()) {

        task.status = 'outdated'

      }

      return task

    })

  },

  mutations: {

    createTask(state, task) {

      state.tasks.push(task)

      localStorage.setItem('tasks', JSON.stringify(state.tasks))

    },

    updateTask(state, {id, description, date}) {

      const tasks = state.tasks.concat()

      const idx = tasks.findIndex(t => t.id ===id)

      const task = tasks[idx]

      const status = new Date(date) > new Date() ? 'active' : 'outdated'

      tasks[idx] = {...task, date, description, status}

      state.tasks = tasks

      localStorage.setItem('tasks', JSON.stringify(state.tasks))

    },

    completeTask(state, id) {

      const idx = state.tasks.findIndex(t => t.id === id)

      state.tasks[idx].status = 'completed'

      localStorage.setItem('tasks', JSON.stringify(state.tasks))

    }

  },

  actions: {

    createTask({commit}, task) {

      commit('createTask', task)

    },

    updateTask({commit}, task) {

      commit('updateTask', task)

    },

    completeTask({commit}, id) {

      commit('completeTask', id)

    }

  },

  getters: {

    tasks: s => s.tasks,

    taskById: s => id => s.tasks.find(t => t.id === id)

  },

  modules: {

  }

})

views/Authorizatio.vue

<template>

<div class="row">

    <form class="col s6 offset-s3" @submit.prevent="submitHandler">

      <div class="row">

        <div class="input-field col s12">

          <input v-model="email" id="email\_inline" type="email" class="validate">

            <label for="email\_inline">Email</label>

            <span class="helper-text" data-error="wrong" data-success="right">Helper text</span>

        </div>

      </div>

      <div class="row">

        <div class="input-field col s12">

          <input v-model="password" id="password" type="password" class="validate">

          <label for="password">Password</label>

        </div>

      </div>

      <button class="btn" type="submit" style="margin-right: 1rem;">Log In</button>

      <button class="btn blue" type="button" @click="signUp" style="margin-right: 1rem;">Sign Up</button>

      <button class="btn red" type="button" @click="signOut">Sign Out</button>

    </form>

  </div>

</template>

<script>

export default {

  methods: {

    submitHandler() {

      this

        .$http

        .post

        (

          'http://localhost:3000/user/login',

          {

            email: this.email,

            password: this.password

          }

        )

        .then(

          (res) => {

            alert(res.body.message)

            localStorage.setItem('token', res.body.token)

          },

          (res) =>

          {

            alert(res.body.message)

          })

    },

    signUp() {

      this

        .$http

        .post

        (

          'http://localhost:3000/user/signup',

          {

            email: this.email,

            password: this.password

          }

        )

        .then(

          (res) => {

            alert(res.body.message)

          },

          (res) =>

          {

            alert(res.body.message)

          })

    },

    signOut() {

      localStorage.removeItem('token');

      alert('Sign Out successful')

    }

  }

}

</script>

<style lang="scss" scoped>

</style>

views/Catalog.vue

<template>

<div class="col s12 m7">

    <h2 class="header">Catalog</h2>

    <div class="card horizontal"

      v-for="(product) of products"

            :key="product.\_id"

    >

      <div class="card-image small">

        <img style="width: 250px; height: 400px;" v-bind:src="product.productImage">

      </div>

      <div class="card-stacked">

        <div class="card-content">

          <p>{{product.name}}</p>

          <p>{{'Price: \$' + product.price}}</p>

        </div>

        <div class="card-action">

          <router-link :to="'/product/' + product.\_id">More details</router-link>

        </div>

      </div>

    </div>

  </div>

</template>

<script>

export default {

  data: () => ({

    products: ''

  }),

  created() {

    this.$http.get('http://localhost:3000/products').then((res) => {

      this.products = res.body.products;

      console.log(res);

    })

  }

}

</script>

views/Orders.vue

<template>

  <div class="col s12 m7">

    <h2 class="header"

      v-if="orders.length > 0"

      >Orders</h2>

    <h2 class="header"

      v-else

      >Orders Not Found</h2>

    <div class="card horizontal"

      v-for="(order) of orders"

            :key="order.\_id"

    >

      <div class="card-image small">

        <img style="width: 250px; height: 400px;" v-bind:src="order.product.productImage">

      </div>

      <div class="card-stacked">

        <div class="card-content">

          <p>{{order.product.name}}</p>

          <p>{{'Price: \$' + order.product.price}}</p>

          <p>{{'Quantity: ' + order.quantity}}</p>

        </div>

        <div class="card-action">

      <button class="btn btn-small" v-on:click="btnDelete(order.\_id)">Delete</button>

        </div>

      </div>

    </div>

  </div>

</template>

<script>

export default {

  data: () => ({

    products: '',

    orders: '',

    srcImage: '',

    price: ''

  }),

  methods: {

    btnDelete(id) {

      this.$http.delete(

        'http://localhost:3000/orders/' + id,

        {

          headers:

          {

            'Authorization': 'Bearer ' + localStorage.getItem('token')

          }

        }

        )

        .then(

          (res) => {

            alert(res.body.message);

          }, (res) =>

          {

            alert(res.body.message);

            if (res.status == '401')

            {

              this.$router.push('/authorization');

            }

          }

        )

    }

  },

  created() {

    this.$http.get(

      'http://localhost:3000/orders/',

      {

        headers:

        {

          'Authorization': 'Bearer ' + localStorage.getItem('token')

        }

      }

      )

      .then(

        (res) => {

          this.orders = res.body.orders;

        }, (res) =>

        {

          alert(res.body.message);

          if (res.status == '401')

          {

            this.$router.push('/authorization');

          }

        }

      )

  }

}

</script>

<style lang="scss" scoped>

</style>

views/Product.vue

<template>

  <div class="row">

      <h1>{{this.product.name}}</h1>

      <img style="weight: 500px; height: 700px" v-bind:src="product.productImage" >

      <p>{{'Price: \$' + product.price}}</p>

      <a class="btn-floating btn-small waves-effect waves-light red" v-on:click="quantity > 1 ? quantity-- : quantity"><i class="material-icons">remove</i></a> {{quantity}}

      <a class="btn-floating btn-small waves-effect waves-light red" style="margin-right: 1rem;" v-on:click="quantity++"><i class="material-icons">add</i></a>

      <button class="btn btn-small" v-on:click="btnBuy">Buy</button>

  </div>

</template>

<script>

export default {

  data: () => ({

    product: '',

    order: '',

    quantity: ''

  }),

  created() {

    this.quantity = 1;

    this

      .$http

      .get(

        'http://localhost:3000/products/'+this.$route.params.id

        )

      .then(

        (res) =>

        {

          this.product = res.body.product;

          console.log(res);

        },

        (err) =>

        {

          alert(err.message);

          console.log(err);

        }

        )

  },

  methods: {

    btnBuy() {

      this

        .$http

        .post(

          'http://localhost:3000/orders/',

          {

            'productId': this.product.\_id,

            'quantity': this.quantity

          },

          {

            headers:

            {

              'Authorization': 'Bearer ' + localStorage.getItem('token')

            }

          })

        .then(

          (res) => {

            this.$router.push('/orders')

          },

          (err) =>

          {

            if (err.status == '401')

            {

              alert(err.message);

              this.$router.push('/authorization');

            }

          }

          )

    }

  }

}

</script>

<style lang="scss" scoped>

</style>

App.vue

<template>

  <div id="app">

  <Navbar />

    <div class="container">

      <router-view/>

    </div>

  </div>

</template>

<script lang="ts">

import Navbar from '@/components/Navbar'

export default {

  components: {Navbar}

}

</script>

<style lang="scss">

  @import "~materialize-css/dist/css/materialize.min.css"

</style>

main.js

import Vue from 'vue'

import VueResource from 'vue-resource'

import App from './App.vue'

import 'materialize-css/dist/js/materialize.min'

import './registerServiceWorker'

import router from './router'

import store from './store'

Vue.use(VueResource)

Vue.config.productionTip = false

new Vue({

  router,

  store,

  render: h => h(App)

}).$mount('#app')

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 028 ПЗ | | | | Пояснительная записка | | | | 56 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР 751003 028 СП | | | | Алгоритм работы метода user\_login Схема алгоритма | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 028 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Веб-приложение “Ez shop”  Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Стубеда В.Д.. |  |  |  |  |  | |  | 64 | 64 |
| Пров. | | Видничук В.Н. |  |  |  | Кафедра ПОИТ  гр. 751003 | | | | | |