МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**РГЗ**

дисциплина:

Технологии Web-программирования

Выполнил: ст. группы ПВ-191

Зимовин Андрей Александрович

Принял:

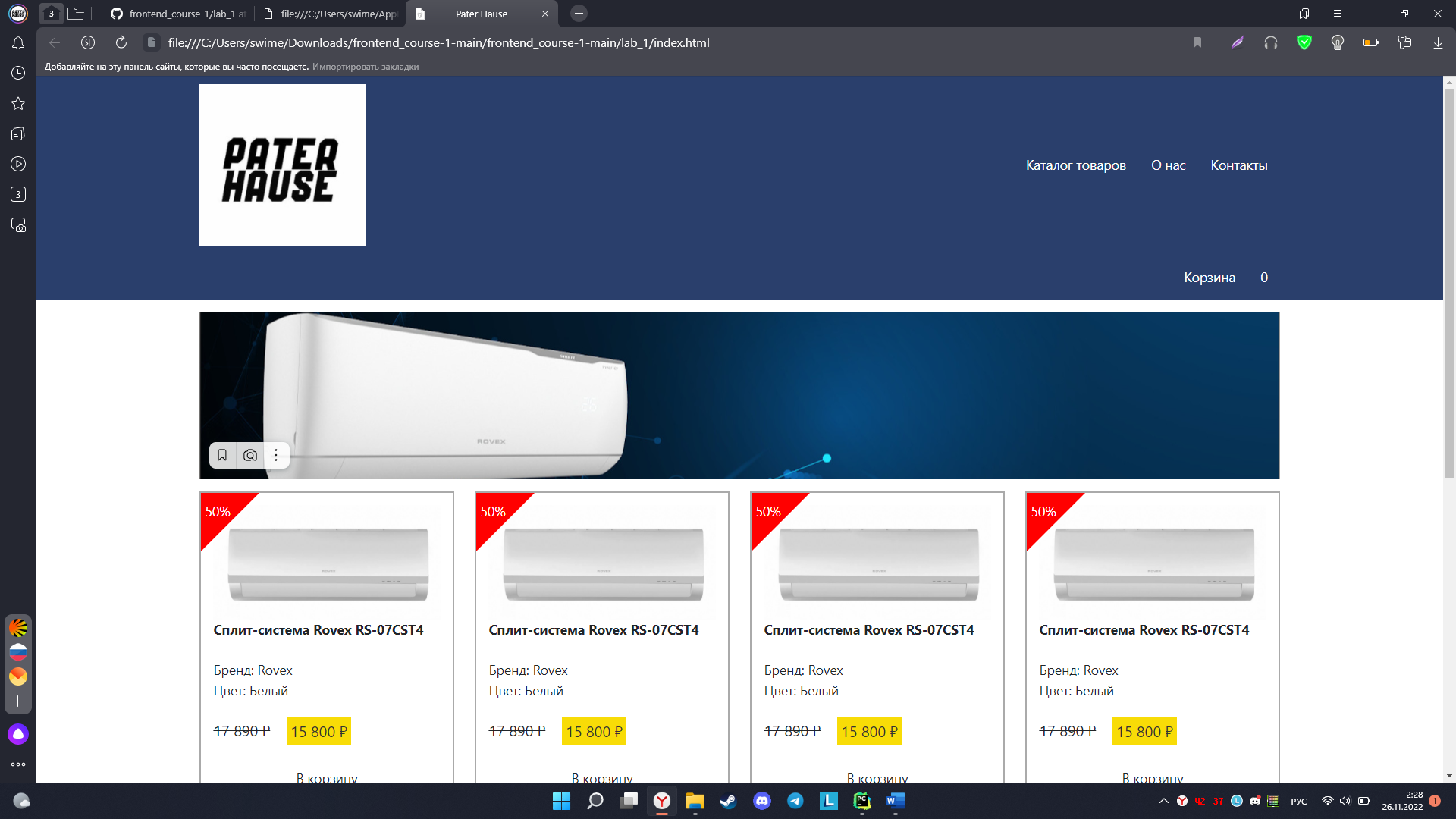
Картамышев Сергей Владимирович

Белгород 2022 г.

**Введение**

В качестве разрабатываемого Web-приложения, была выбрана тема «интернет-магазина». Для выполнения РГЗ проект был разбит на несколько частей:

1. Верстка сайта на HTML+ CSS
2. Добавления JS к сайту
3. Перенос приложения на Фреймворк VUE/React/др.
4. Разработка Backend-приложения и подключение к Frontend
5. Создание docker контейнера приложения
6. **Верстка сайта на HTML+ CSS**



**Файл app.css**

html, body {

padding: 0;

margin: 0;

font-family: 'Open Sans', sans-serif;

}

.wrapper {

min-height: 100vh;

margin-bottom: 30px;

}

header {

background-color: rgb(42, 63, 108);

padding: 0 15px;

}

header .container {

display: flex;

flex-direction: row;

align-items: center;

}

header .logo img {

width: 200px;

padding: 10px 0;

}

header nav {

margin-left: auto;

}

header nav ul {

display: flex;

list-style: none;

color: #fff;

padding: 0;

margin: 0;

}

header nav ul li {

padding: 15px 15px;

}

.products-list .product {

position: relative;

border: 2px solid #aaa;

padding: 15px;

}

.products-list .product img {

width: 100%;

}

.products-list .product .product-title {

font-size: 16px;

}

.price {

font-size: 18px;

}

1. **Добавления JS к сайту**

**Файл style.js**

var counterVal = 0;

function incrementClick() {

updateDisplay(++counterVal);

}

function resetCounter() {

counterVal = 0;

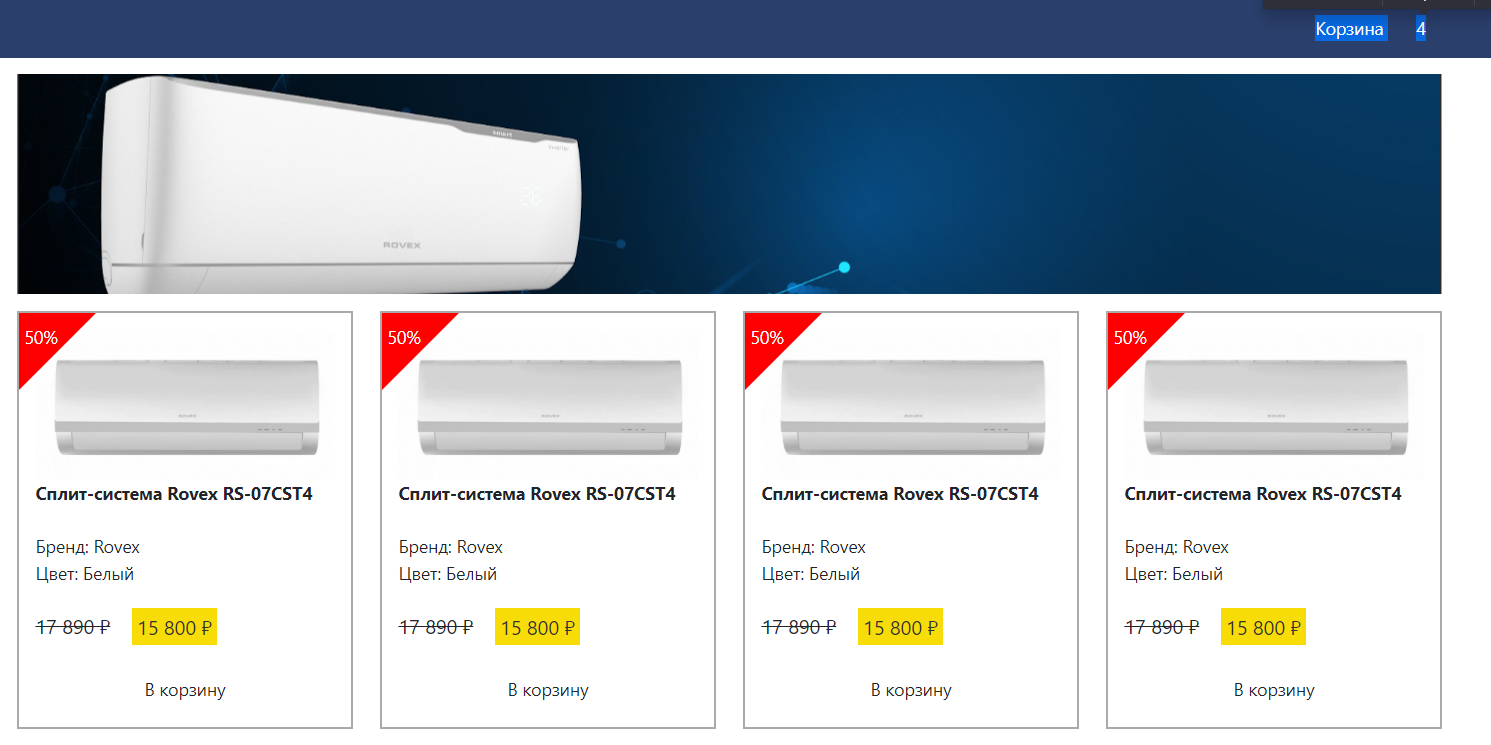
updateDisplay(counterVal);

}

function updateDisplay(val) {

document.getElementById("counter-label").innerHTML = val;

}



1. **Перенос приложения на Фреймворк VUE/React/др.**

**Изображение выглядит как текст, дерево, внешний, гора

Автоматически созданное описание**

**Изображение выглядит как текст

Автоматически созданное описание**

**Файл ProductCard.vue:**

<template>

<div class="product">

<div>

<img src="../../../public/catalog\_images/rovex.jpeg" />

</div>

<div class="product-title"><b>Rovex Trend MDX1 09</b></div>

<br>

<div>Бренд: Rovex</div>

<div>Сезон: 2022</div>

<br>

<div>

<span class="price">45 890 ₽</span>

<span class="discount">22 945 ₽</span>

</div>

<div class="triangle"></div>

<div class="discount-percent">50%</div>

<br>

<button class="btn btn-cart">В корзину</button>

</div>

</template>

<script>

export default {

name: "ProductCard"

}

</script>

<style>

.product {

position: relative;

border: 1px solid #ddd;

padding: 15px;

}

.product .product-title {

font-size: 16px;

}

.product img {

width: 100%;

}

.price {

font-size: 18px;

}

.discount {

background-color: #f8dc07;

color: #333;

padding: 5px;

font-size: 18px;

margin-left: 15px;

}

.triangle {

position: absolute;

left: 0;

top: 0;

width: 0;

height: 0;

border-style: solid;

border-width: 70px 70px 0 0;

border-color: red transparent transparent transparent;

}

.discount-percent {

position: absolute;

top: 10px;

left: 5px;

color: #fff;

font-size: 16px;

}

.btn-cart {

width: 100%;

background-color: rgb(42, 63, 108);

color: #fff;

border-radius: 4px;

}

.btn.btn-cart:hover {

background-color: rgb(19, 24, 47);

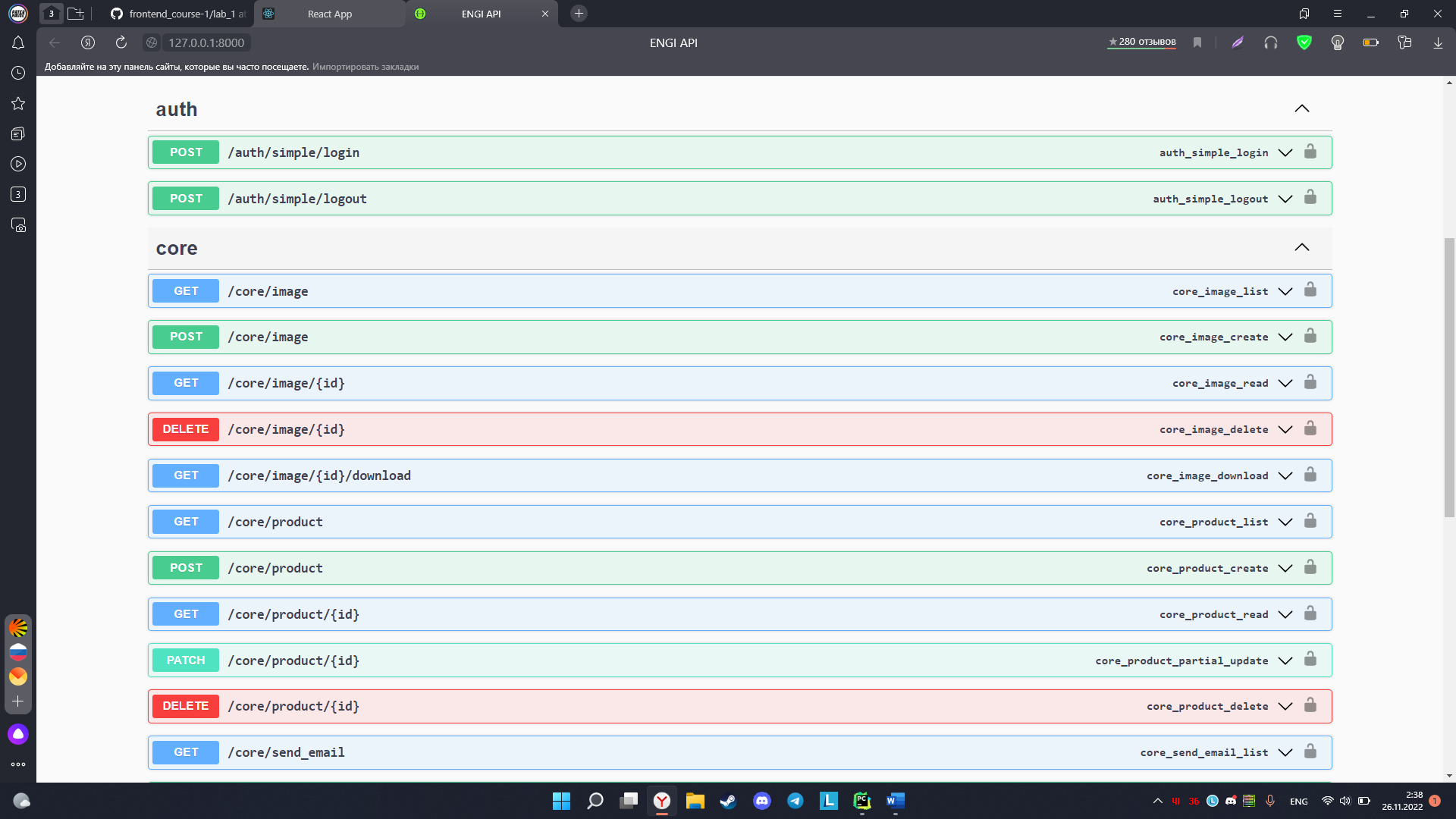
color: #fff;

}

</style>

1. **Разработка Backend-приложения и подключение к Frontend**

Для разработки backend-а был выбран Фреймворк Django3. Он написан на языке Python и содержит в себе огромное кол-во модулей. Данный фреймворк хорошо взаимодействует с различными веб-фреймворками. Было написанно следующее API:



Изображение выглядит как текст

Автоматически созданное описание

Файл, содержащий описания моделей

**core.py:**

*from* django.db *import* models  
  
*from* core.models.dicts *import* DictProduct  
  
  
*class* Product(models.Model):  
 name = models.CharField("Название", max\_length=250)  
 price = models.IntegerField("Цена", default=0)  
 description = models.CharField("Описание товара", max\_length=500, null=*True*)  
 type\_product = models.ForeignKey(DictProduct, on\_delete=models.CASCADE)  
  
  
*class* Image(models.Model):  
 name = models.CharField("Имя", max\_length=200)  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE, related\_name='image')  
 path = models.ImageField(upload\_to='media')  
 date\_upload = models.DateTimeField("Дата загрузки", auto\_now=*True*)  
  
*class* FeedBack(models.Model):  
 email = models.CharField("Email", max\_length=100)  
 text = models.CharField("Текст", max\_length=200)  
 date = models.DateTimeField("Дата получения обращения", auto\_now=*True*)

**dicts.py:**

*from* django.db *import* models  
  
  
*class* DictProduct(models.Model):  
 TYPES = (  
 (0, 'Бытовая техника'),  
 (1, 'Промышленное оборудование'),  
 (2, 'Инструменты'),  
 (3, 'Дом, декор и кухня'),  
 (4, 'Строительство')  
 )  
 type = models.IntegerField("Тип", choices=TYPES, primary\_key=*True*)  
  
 *def* \_\_str\_\_(self):  
 *return* self.get\_type\_display()

1. **Создание docker контейнера приложения**

**Dockerfile:**

ARG PYTHON\_VERSION=3.10  
  
FROM python:${PYTHON\_VERSION}  
  
WORKDIR /app  
  
COPY requirements.txt /app/  
RUN \  
 pip install --force-reinstall --ignore-installed --no-cache-dir \  
 --use-feature=in-tree-build -r requirements.txt gunicorn  
  
COPY . /app

Docker-compose.yml:

version: '3'  
  
services:  
 db:  
 container\_name: postgres\_en  
 image: postgres  
 hostname: db  
 env\_file:  
 - .env  
 volumes:  
 - postgres\_data:/var/lib/postgresql/data  
 restart: always  
  
 backend:  
 container\_name: backend\_en  
 image: app:latest  
 build:  
 context: online-shop  
 dockerfile: Dockerfile  
 command: sh -c "./wait-for-it.sh db:5432 -- python manage.py migrate &&  
 gunicorn --bind 0.0.0.0:8000 online-shop.wsgi:application  
 --log-level=info --log-file=./logs/gunicorn.log  
 --access-logfile=./logs/gunicorn-access.log"  
 environment:  
 DJANGO\_SETTINGS\_MODULE: "online-shop.settings.settings\_server"  
 env\_file:  
 - .env  
 depends\_on:  
 - db  
 restart: on-failure  
  
 frontend:  
 container\_name: frontend\_en  
 build:  
 context: frontend  
 dockerfile: Dockerfile  
 ports:  
 - "8088:8088"  
 depends\_on:  
 - backend  
 restart: unless-stopped  
volumes:  
 postgres\_data:

1. **Привязка API к Frontend**

**Contacts.js:**

class Contacts extends Component {  
 constructor(*props*) {  
 super(*props*);  
 this.state = {  
 textArea: "",  
 email: "",  
 send\_me: false  
 };  
 this.handleChangeEmail = this.handleChangeEmail.bind(this);  
 this.handleSubmit = this.handleSubmit.bind(this);  
 this.handleChangeText = this.handleChangeText.bind(this);  
 this.handleSubmitSend = this.handleSubmitSend.bind(this);  
 this.sender = this.sender.bind(this);  
 }  
  
 handleChangeEmail(*event*) {  
 this.setState({email: *event*.target.value});  
 }  
  
 handleChangeText(*event*) {  
 this.setState({textArea: *event*.target.value});  
 }  
  
 handleSubmitSend(*event*){  
 this.setState({send\_me: *event*.target.checked});  
 }  
  
 handleSubmit(*event*) {  
 *console*.log('Почта: ' + this.state.email);  
 *console*.log('Текст:'+ this.state.textArea);  
 *console*.log('Отправить мне?', this.state.send\_me);  
 *event*.preventDefault();  
 }  
  
 sender() {  
 const requestOptions = {  
 method: 'POST',  
 headers: { 'Content-Type': 'application/json' },  
 body: *JSON*.stringify({'email': this.state.email,  
 'text': this.state.textArea, 'send\_me': this.state.send\_me})  
 };  
 fetch('http://127.0.0.1:8000/api/v1/core/send\_email', requestOptions)  
 .then(*response* => *response*.json())  
 .then(  
 (*error*) => {  
 *console*.log(*error*);  
 }  
 );  
 alert("Ваше обращение отправлено!\nОтвет будет получен в ближайшее время.")  
 }

**Catalog.js:**

export default class Catalog extends Component {  
 constructor(*props*) {  
 super(*props*);  
 this.state = {  
 viewCompleted: false,  
 activeItem: {  
 id: 0,  
 name: "",  
 price: 0,  
 description: "",  
 type\_product: 0  
 },  
 products: []  
 };  
 }  
  
 componentDidMount() {  
 *axios* .get("http://127.0.0.1:8000/api/v1/core/product")  
 .then(  
 (*result*) =>{  
 this.setState({  
 isLoaded: true,  
 products: *result*.data  
 });  
 },  
 (*error*) => {  
 this.setState({  
 isLoaded: true,  
 *error* });  
 }  
 )  
 }

**Заключение**

В ходе выполнения РГЗ мною были изучены фреймворки Vue и React JS, а также подкреплены умения владения фреймворком Django3. Так же была продела работа с Git и Docker. В результате был написан интернет-магазин, взаимодействующий с API backend-а. Пройденной дисциплиной доволен!