МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Web-приложение конфигуратор компьютера»

Выполнил студент Песецкий Никита Андреевич

(Ф.И.О.)

Руководитель работы асс. Нистюк Ольга Александровна

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

И.о. зав. кафедрой ст. преп. Блинова Евгения Александровна

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

Курсовая работа защищена с оценкой

Минск 2024

**Содержание**

[Введение 4](#_Toc166253162)

[1 Постановка задачи 5](#_Toc166253163)

[1.1 Аналитический обзор аналогов 5](#_Toc166253164)

[1.1.1 Amd.by 5](#_Toc166253165)

[1.1.2 LeoPanda.by 6](#_Toc166253166)

[1.1.3 npx.by 6](#_Toc166253167)

[1.2 Разработка функциональных требование и вариантов использования 7](#_Toc166253168)

[1.3 Выводы по разделу 8](#_Toc166253169)

[2 Проектирование web-приложения 9](#_Toc166253170)

[2.1 Обобщенная структура управлением приложения 9](#_Toc166253171)

[2.2 Диаграммы UML, взаимосвязь всех компонентов. 9](#_Toc166253172)

[2.4 Выводы по разделу 10](#_Toc166253173)

[3 Разработка web-приложения 11](#_Toc166253174)

[3.1 Разработка клиентской части web-приложения 11](#_Toc166253175)

[3.2 Разработка серверной части web-приложения 15](#_Toc166253176)

[3.3 Выводы по разделу 18](#_Toc166253177)

[4 Тестирование web-приложения 19](#_Toc166253178)

[5 Руководство пользователя 21](#_Toc166253179)

[5.1 Руководство пользователя 21](#_Toc166253180)

[5.2 Установка приложения 27](#_Toc166253181)

[5.3 Выводы по разделу 28](#_Toc166253182)

[Заключение 29](#_Toc166253183)

[Список используемых источников 30](#_Toc166253184)

# Введение

В современном мире компьютеры и цифровые технологии становятся все более важными в повседневной жизни людей. Они широко используются в самых разных сферах, включая работу, образование и развлечения. В рамках данного курсового проекта было принято решение разработать приложение, которое поможет пользователям создавать и настраивать компьютеры по своим потребностям и предпочтениям.

Целью создания данного проекта является обеспечение пользователей возможностью легко и удобно создавать персонализированные компьютеры, учитывая их предпочтения, бюджет и потребности. Приложение будет предоставлять интерфейс для выбора компонентов, их настройки и оценки итоговой конфигурации.

Преимущества данного проекта заключается в том, что пользователи получат возможность создавать свои собственные компьютеры, учитывая их уникальные потребности и предпочтения. Приложение предоставит широкий выбор компонентов, а также инструменты для сравнения различных комплектующих.

Актуальность и необходимость создания данного проекта обусловлены рядом факторов. Во-первых, современные компьютеры становятся все более сложными и разнообразными, и многие пользователи испытывают затруднения при выборе подходящей конфигурации. Во-вторых, с развитием интернет-технологий и электронной коммерции существует растущий спрос на онлайн-сервисы, позволяющие пользователям настраивать и заказывать товары и услуги прямо из дома.

Для разработки серверной части данного веб-приложения был выбран фреймворк Express, который позволит разработать быстрое и многофункционально веб-приложение; а для разработки клиентской части была выбрана библиотека. Для работы с БД будет использоваться ORM mongoose для работы с MongoDB.

В основной части курсового проекта будут рассмотрены все аспекты разработки приложения, включая архитектуру, реализацию функциональности, технические приемы и инструменты. Также будут представлены руководства пользователя и администратора, а также краткая информация о похожих продуктах на рынке.

# 1 Постановка задачи

Целью данного курсового проекта является создание Web-приложения для создания конфигурации компьютера. Для этого необходимо выполнить следующие задачи:

* обеспечивать возможность регистрации и авторизации;
* создать роли администратора и пользователя;
* возможность создавать/удалять/обновлять конфигурации компьютеров;
* возможность создавать и удалять публикации конфигурации;
* возможность оставлять отзывы к публикациям других пользователей;
* возможность просматривать информацию о комплектующих;
* возможность сравнение комплектующих;
* администратор может создавать/удалять/обновлять список комплектующих;
* администратор может удалять публикации пользователей;
* администратор может блокировать пользователей;

## 1.1 Аналитический обзор аналогов

Были проанализированы цели и задачи, поставленные в данном курсовом проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

### 1.1.1 Amd.by

Amd.by это онлайн-гипермаркет, который предоставляет возможность создания конфигураций компьютеров. Он обладает широким ассортиментом комплектующих, которые присутствуют в онлайн-магазине. Интерфейс интернет-ресурса представлен на рисунке 1.1.

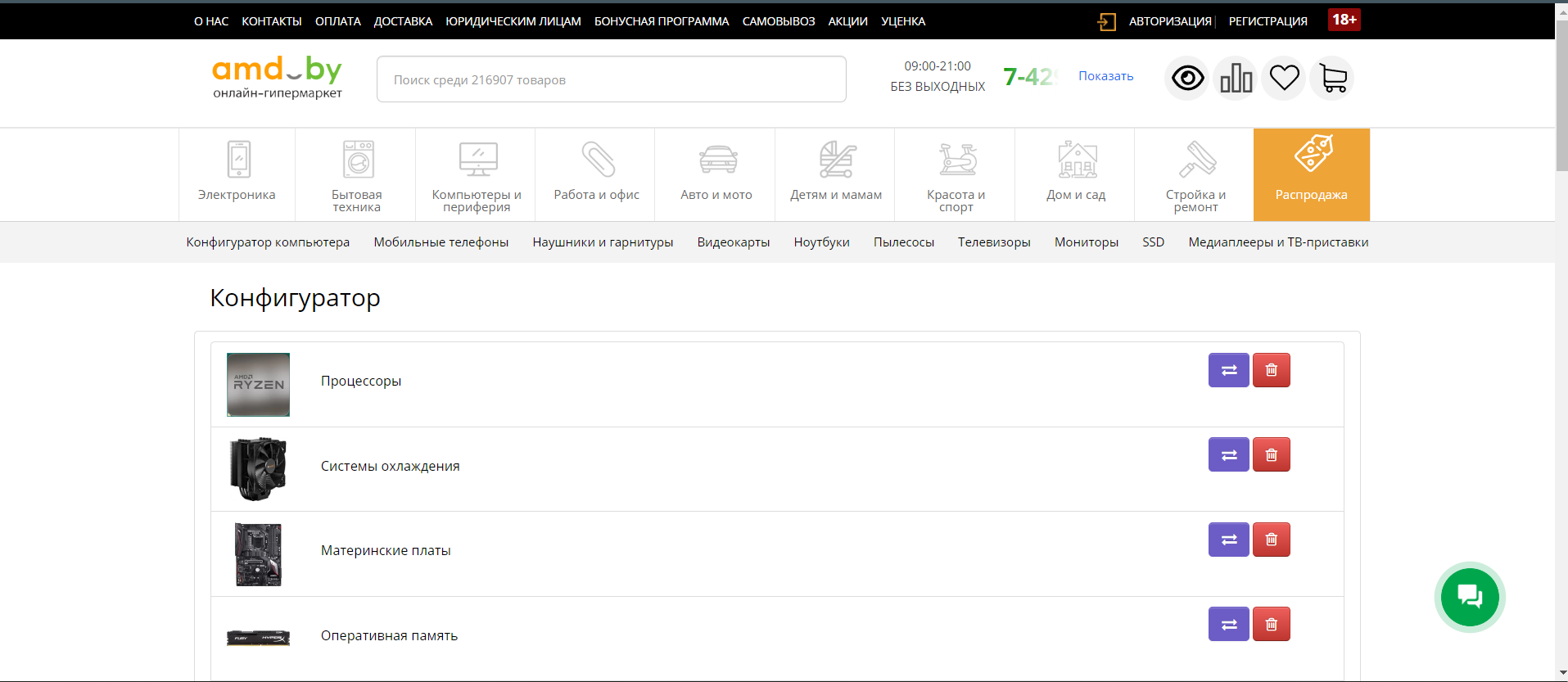


Рисунок 1.1 – Конфигуратор сайта Amd.by

Преимуществами этого конфигуратора можно назвать интуитивно понятный дизайн, возможность узнать цену комплектующих.  
 Недостатками данного приложения является отсутствие тестов производительности и не все возможные компьютерные комплектующие.

### 1.1.2 LeoPanda.by

Сайт LeoPanda.by является онлайн-гипермаркетом, на котором присутствует конфигуратор компьютеров. С первого взгляда может показаться что сайт весьма хороший, но на самом деле конфигуратор достаточно плохой. Так же присутствует раздел готовых сборок.

Конфигуратор плох по нескольким причинам. Первая из них это отсутствие всех возможных комплектующих. В предоставленном выборе присутствуют только новые комплектующие, которые появились на рынке недавно. Конфигуратор перегружен бесполезными разделами такими как роутер, сетевой фильтр и 3 диска. На рисунке 1.2 представлена часть интерфейса данного сайта.

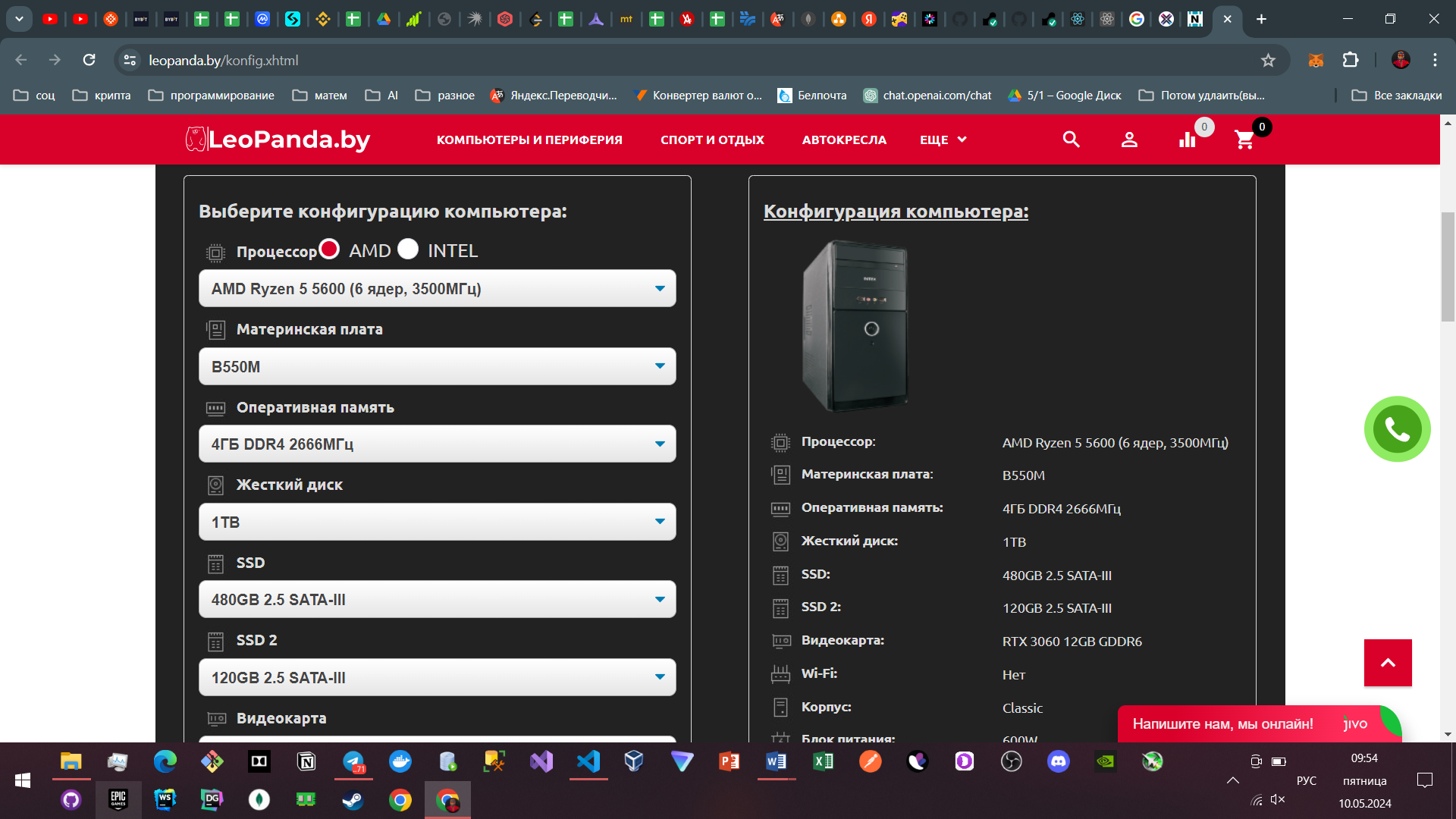


Рисунок 1.2 – Конфигуратор сайта LeoPanda.by

Так при сборе конфигурации не указывается производитель комплектующих, а это часто влияет на производительность компьютера.

На сайте присутствует раздел профиля и корзины, можно создать свою конфигурацию и добавить её в корзину.

Исходя из ранее написанного можно сказать, что технически конфигуратор плох, но интерфейс интуитивно понятен.

### 1.1.3 npx.by

Сайт npx.by является онлайн магазином, направленным только на продажу компьютерных комплектующих и готовых персональных компьютеров. Данный сайт является лучшим среди всех рассматриваемых.

У данного сайта отсутствует проблема с неуказанными производителями комплектующих, но присутствует проблема с присутствием бесполезных разделов. Интерфейс данного сайта интуитивно понятен.

Интерфейс приложения представлен на рисунке 1.3.

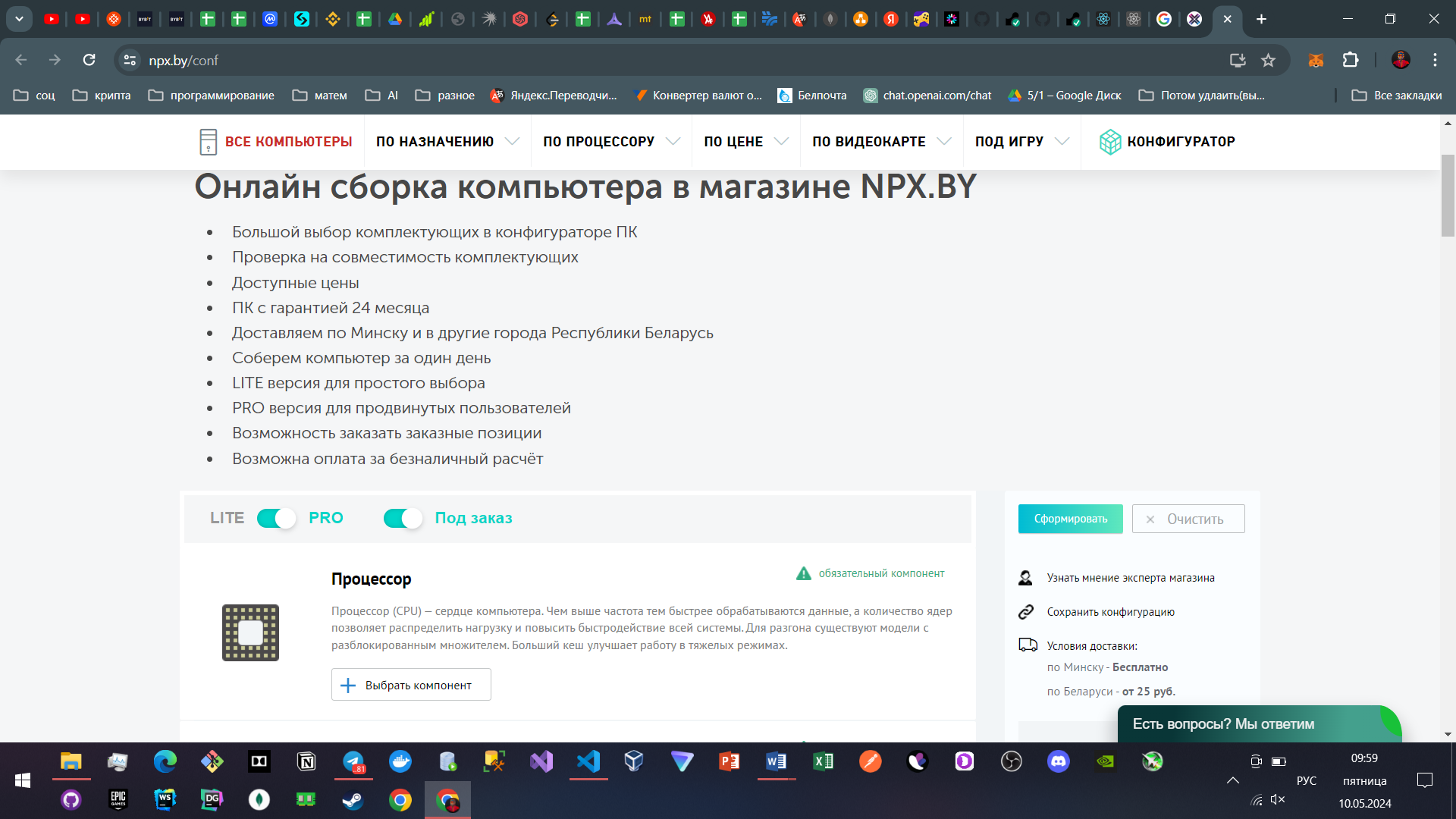


Рисунок 1.3 – Конфигуратор сайта npx.by

На сайте присутствует чат с поддержкой, это может помочь неопытным пользователям при возникновении проблемы.

Исходя из выше перечисленного можно определить плюсы и минусы сайта.  
Плюсами данного сайта является интуитивно понятный интерфейс, хороший выбор комплектующих. Минусами является перегруженный интерфейс конфигурации компьютера.

### 1.1.4 ironbook.ru

Сайт ironbook.ru является онлайн магазином. На данном сайте мы можем увидеть интерфейс, похожий на интерфейс сайтов из прошлых примеров. Проанализировав этот сайт можно вывести типовые ошибки таких сайтов. Сайт представлен на рисунке 1.4.

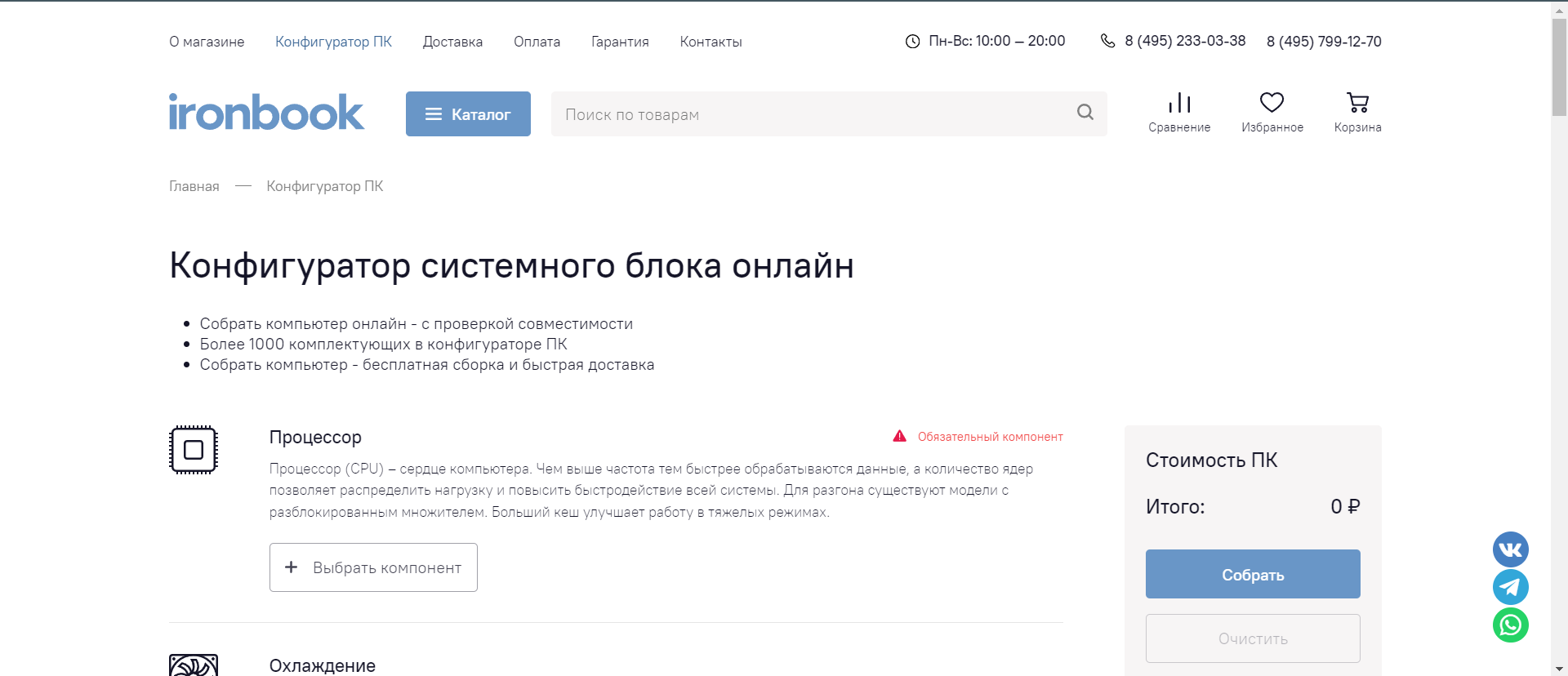


Рисунок 1.4 – Конфигуратор сайта ironbook.ru

Первая из типовых ошибок это перегруженный интерфейс. На данном сайте при сборе компьютера в конфигурацию предлагают добавить 3 диска, платную установку операционной системы, дисковод.

Плюсами данного сайта можно назвать яркий дизайн, понятный интерфейс.

## 1.2 Разработка функциональных требование и вариантов использования

Обзор вышеперечисленных известных аналогов позволяет проанализировать все преимущества и недостатки альтернативных возможностей и позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте. Программное средство должно обеспечивать возможность выполнения перечисленных ниже функций:

* Регистрация и авторизация пользователя;
* Обеспечение безопасного соединения посредством SSL;
* Возможность создания конфигурации компьютера;
* Сохранение конфигурации;
* Возможность оставлять и просматривать отзывы;
* Получение онлайн-консультации;

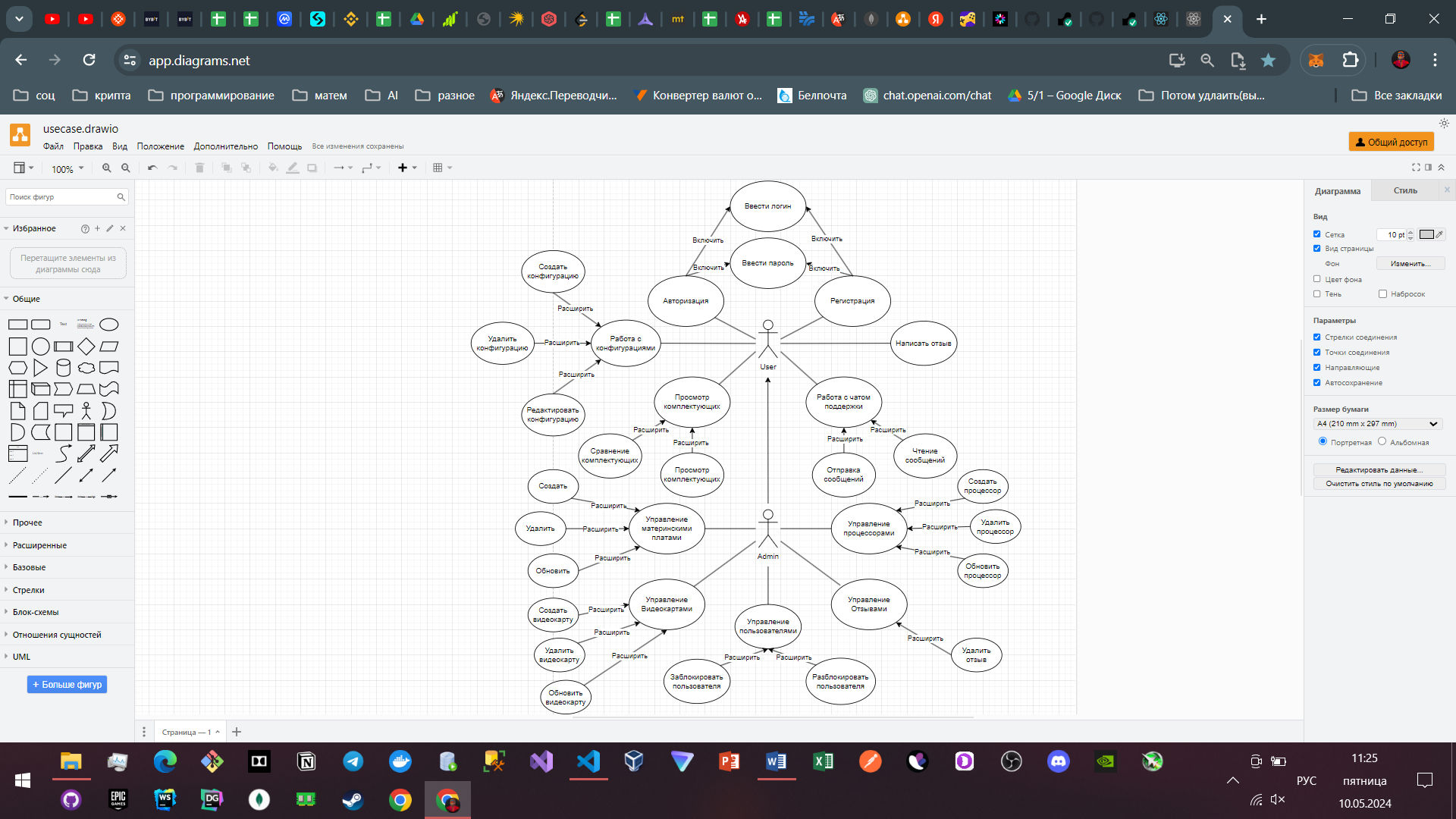


Рисунок 1.5 – Диаграмма вариантов использования

Диаграмма UML вариантов использования позволяет понять, что доступно каждой роли, доступной в данном веб-приложении.

## 1.3 Выводы по разделу

В данном разделе мы рассмотрели аналогичные веб-приложения. Исходя из этих аналогов можно взять пару идей для своего веб-приложения. Например, данный конфигуратор компьютера должен обеспечивать удобный интерфейс, предоставлять безопасное соединение.

Также в данном разделе рассмотрели весь основной функционал, который должен присутствовать в конечном приложении. Данный функционал рассчитан на то, что он позволит довольно-таки просто начать пользоваться этим веб-приложением, а также что этот функционал позволит создавать конфигурации компьютера разным людям. Была приведена UML диаграмма, на которой все и приведено.

Кроме анализа аналогичных приложений, важно учитывать не только интерфейс и безопасность, но и основной функционал, необходимый для удовлетворения потребностей пользователей. Этот функционал должен быть интуитивно понятен и удобен в использовании для широкого круга пользователей. Например, конфигуратор компьютера должен предлагать выбор компонентов, возможность сохранять конфигурации, сравнивать различные варианты, а также быть гибким для адаптации к различным запросам пользователей.

Важно также учесть аспекты безопасности, чтобы обеспечить защиту данных пользователей и обеспечить надежное соединение при работе с приложением. Это включает в себя использование шифрования данных и применение соответствующих методов защиты от угроз безопасности.

Помимо этого, конечное веб-приложение должно быть гибким и масштабируемым, чтобы соответствовать потребностям разных категорий пользователей. Для этого важно адекватно структурировать основной функционал, основываясь на анализе требований пользователей и бизнес-процессов.

Анализ UML диаграммы позволяет визуализировать структуру приложения и взаимосвязи между его компонентами. Это помогает лучше понять функциональные возможности приложения и оптимизировать его проектирование и разработку.

Таким образом, основываясь на анализе аналогичных приложений и основных функциональных требованиях, можно сформировать четкое видение конечного веб-приложения, которое будет удовлетворять потребности пользователей и обеспечивать безопасное и удобное использование.

# 2 Проектирование web-приложения

## 2.1 Обобщенная структура управлением приложения

Для обеспечения управления приложением с использованием базы данных необходимо разработать удобный и интуитивно понятный интерфейс, который позволит пользователю взаимодействовать с базой данных и эффективно управлять данными. Это может включать в себя разработку оптимизированных запросов для вставки, обновления и удаления данных, а также разработку механизмов для извлечения и обработки информации из базы данных.

В функциональность приложения должны входить функции для удобного поиска конфигураций и комплектующих в приложении, функции создания конфигураций и другие подобные функции.

## 2.2 Диаграммы UML, взаимосвязь всех компонентов.

Диаграмма базы данных таблиц (Database Table Diagram) – это визуальное представление структуры базы данных и отношений между таблицами, которые хранятся в этой базе данных. Диаграмма базы данных представлена на рисунке 2.1.

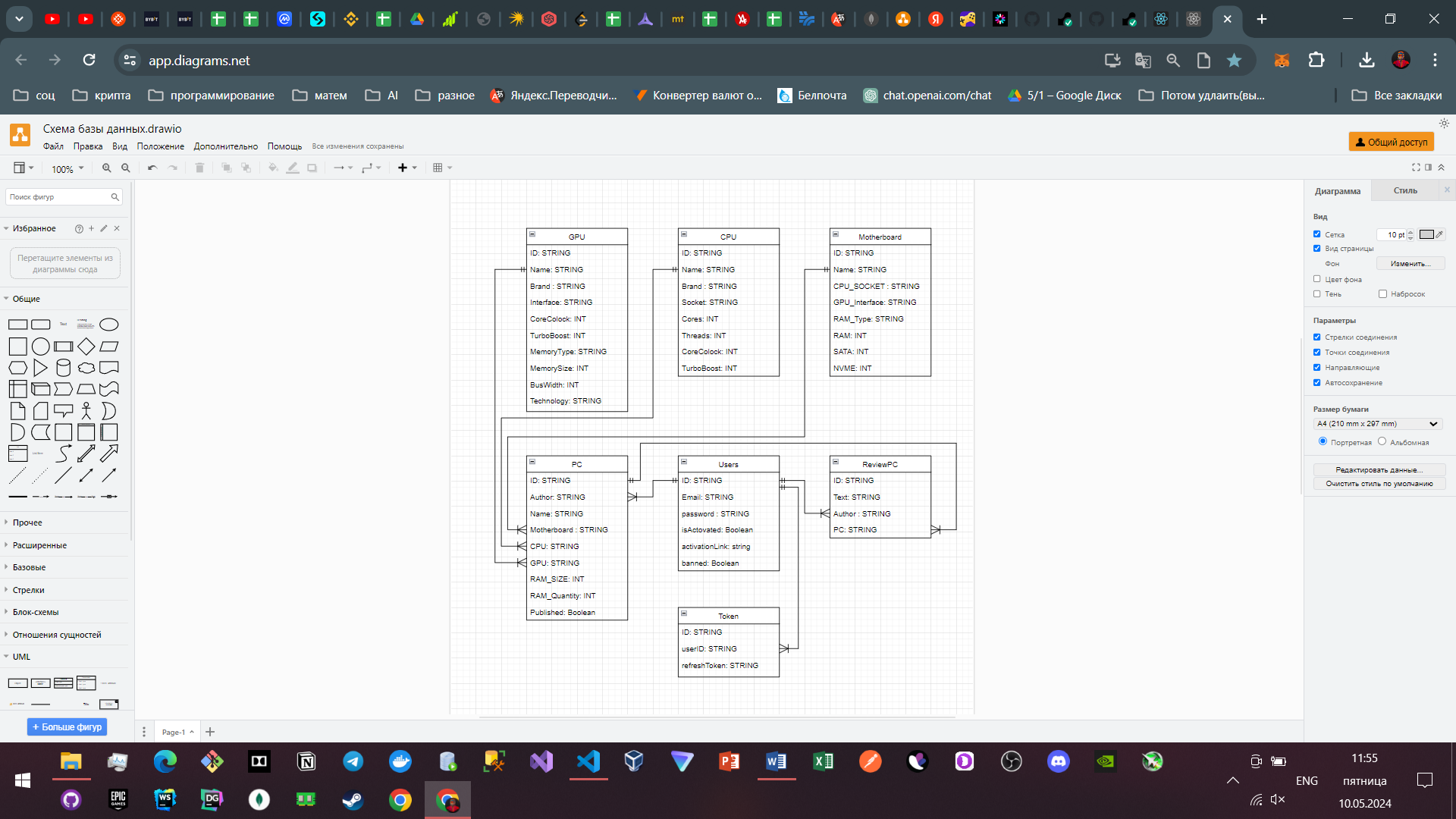


Рисунок 2.1 – Логическая схема базы данных

Таким образом, диаграмма показывает связи между таблицами и полями, а также отношения между ними, такие как связи «один-ко-многим. Например, таблица Users связана с таблицами PC, Token, ReviewPC через внешние ключи.

Для того понимания структуры данного приложения и как его разворачивать нам может помочь диаграмма развертывания. Данная диаграмма показана на рисунке 2.2.

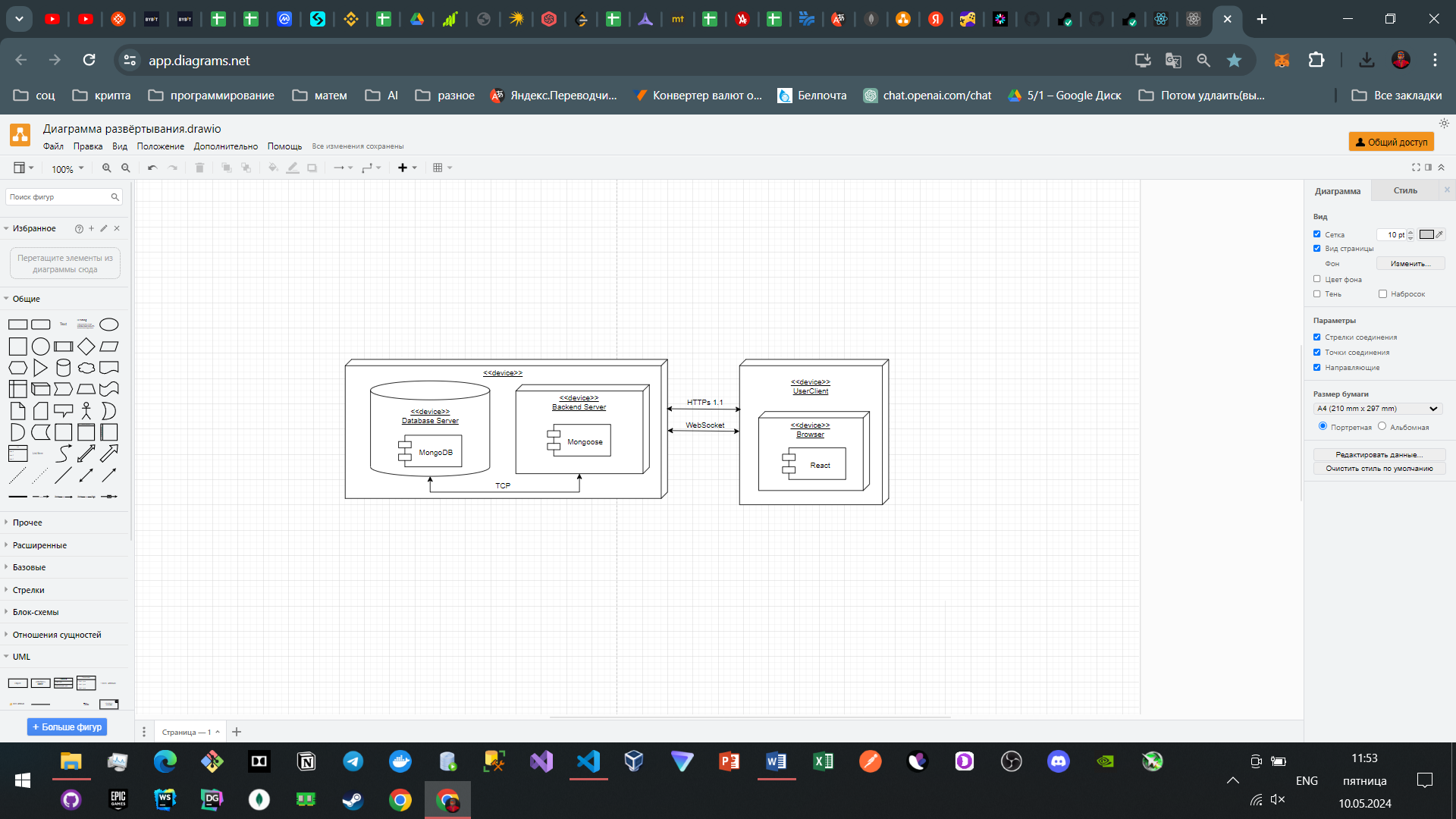


Рисунок 2.2 – Диаграмма развертывания

По данной диаграмме развертывания видно, что веб-приложение состоит из трех основных частей. Первая часть – это клиентская часть нашего приложения. Эта часть реализована при помощи одноименной библиотеки для языка программирования JavaScript ReactJS. Данная библиотека поможет создать динамически SPA приложение, которое будет делать запросы по протоколу HTTP к серверной части приложения. Также оно будет отправлять и получать сообщения по протоколу WebSocket. Вторая часть данного веб-приложения – это серверная часть, разработанная на платформе NodeJS при помощи фреймворка Express, который позволяет построить структуру из всех основных модулей нашего приложения. Данная часть приложения будет обращаться к базе данных по протоколу TCP. Обращение к MongoDb будет благодаря ORM модулю mongoose, который легко позволяет работать с сущностями, хранимыми внутри данной базы данных. Третья часть – это MongoDb-сервер, где и будут хранится все данные.

## 2.4 Выводы по разделу

Разработка архитектуры проекта необходима для определения структуры и функциональности приложения. Обобщенная структура управления приложения позволяет определить, какие компоненты необходимы для реализации приложения и как они должны взаимодействовать между собой.

В данном разделе рассмотрели структуру базы данных MongoDB, ее основные сущности и как они связаны. Также рассмотрели, какие есть основные части нашего веб-приложения, точнее, во что оно развертывается.

# 3 Разработка web-приложения

## 3.1 Разработка клиентской части web-приложения

Клиентская часть – главная часть приложения. Она должна быть интуитивно понятна и интерактивна. Клиентская часть будет состоять из двух основных частей: библиотека React и WebSocket клиент.

Часть React состоит из девяти страниц. Когда пользователь заходит первый раз в веб-приложение ему доступна лишь одна страница – страница авторизации. В библиотеки React для маршрутизации используется компонент AppRouter. Пример использования данной компоненты показан в листинге 3.1.

<Routes>.isAuth && store.user.banned) ? (

bannedRoutes.map(({ path, Component }) => (

<Route key={path} path={path} element={<Component />} />

))

) : null}

{(store.user.banned) ? (

<Route path="\*" element={<Navigate to="/ban" replace />} />

) : null}

{(store.isAuth && store.user.role == "admin") ? (

adminRoutes.map(({ path, Component }) => (

<Route key={path} path={path} element={<Component />} />

))) : null}</Routes>

Листинг 3.1 – Изначальные настройки маршрутизации

По данному коду видно, что если пользователь авторизован, то его будет пересылать на MainPage, где будет происходить другая маршрутизация, о которой будет сказано чутка позже. Если же пользователь не авторизован, то ему будет доступны страница из PublicRoutes, в которых также есть своя маршрутизация.

Объект authRoutes содержит в себе пять маршрутизации. Первая маршрутизация – это по URL равном «/login». Она будет маршрутизировать на компонент profile, который будет содержать разметку профиля. Вторая маршрутизация – это по URL равно «/pc». Она будет маршрутизировать на компонент компьютерных конфигураций, который будет содержать разметку публикаций конфигураций. Третья маршрутизация отвечает за вкладку для сравнения комплектующих. Четвертая маршрутизация отвечает за базу дынных комплектующих. Пятая маршрутизация отвечает за связь с поддержкой. Также есть шестая маршрутизация. Данная маршрутизация будет отвечать за все остальные URL, которые не были описаны ранее. Ее задача будет, если авторизованный пользователь введет не поддерживаемый URL, перенаправлять на «/login», как компонент на случай ненайденной страницы. Объект authRoutes представлена в листинге 3.2.

export const authRoutes = [

{

path: PCCONFIGS\_ROUTE,

Component: PcStorePage

},

{

path: CURRENTPC\_ROUTE,

Component: PcPage

},

{

path: COMPARE\_ROUTE,

Component: Compare

},

{

path: CHAT\_ROUTE,

Component: Chat

},

{

path: STORE\_ROUTE,

Component: ComponentsStore

}

]

Листинг 3.2 – Объект authRoutes

Объект BannedRoutes отвечает за маршрутизацию для авторизованного пользователя. Пользователю доступна одна страницы, это компонент BannedUser. Данный компонент отвечает за страницу с сообщениями о блокировке аккаунта. Так же на странице находится кнопка для выхода из аккаунта. Объект BannedRoutes показан в листинге 3.3.

{(store.user.banned) ? (

<Route path="\*" element={<Navigate to="/ban" replace />} />

) : null}  
  
export const bannedRoutes = [

{

path: BANNED\_ROUTE,

Component: BannedUser

}

]

Листинг 3.3 – Объект BannedRoutes

Объект adminRoutes отвечает за маршрутизацию на одну страницу. Эта страница панели управления администратора. На этой странице реализован весь функционал администратора. Объект BannedRoutes показан в листинге 3.4.

export const adminRoutes = [

{

path: ADMIN\_PANEL\_ROUTE,

Component: adminPanel

}

]

Листинг 3.4 – Объект adminRoutes

Компонент AdminPanel отвечает за страницу панели администратора. Она состоит из компонентов CPU\_FORM, GPU\_FORM, MOTHERBOARD\_FORM, USER\_FORM. В этих компонентах реализован функционал создания, удаления, обновления данных из основных таблиц сайта. Реализация компоненты находится в листинге 3.5.

function AdminPanel(){

return(

<div className="AdminPanel">

<CPUForm/>

<GPUForm/>

<MotherboardForm/>

<UsersForm/>

</div>

)

}

Листинг 3.5 – Объект AdminPanel

Объект publicRoutes отвечает за маршрутизацию на одну страницу. Эта страница панели управления администратора. На этой странице реализован весь функционал администратора. Объект BannedRoutes показан в листинге 3.4.

Компонент ProfilePage отвечает за страницу профиля авторизованного пользователя. Реализация компонента находится в листинге 3.5. На данной странице можно создать конфигурацию, просмотреть список конфигурации, посмотреть список отзывов.

function ProfilePage(){

const {store} = useContext(Context);

const [reviews,setReviews] = useState([]);

const [revPage, setRevPage] = useState(1);

const [userPCs,setuserPCs] = useState([]);

const [pcPage, setPcPage] = useState(1);

async function getUserPCs(){ }

async function getReviews(){ }

useEffect(()=>{

getUserPCs();

getReviews();

}, [])

return( )

}export default observer(ProfilePage);

Листинг 3.6 – Компонент ProfilePage

За хранение состояния на клиентской части отвечает объект store и хуки CreateContext, useContext; Они позволяетют хранить состояние и делать запросы к серверу с помощью хуков в React. Реализация store показано в листинге 3.7.

import {IUser} from '../models/IUser'

import {makeAutoObservable} from 'mobx' //012441

import AuthService from '../service/AuthService';

import axios from 'axios';

import { API\_URL } from '../http';

export default class Store{

user = new IUser();

isAuth = false;

isLoading = false;

constructor(){ }

setAuth(bool){ }

setUser(user){ }

async login(email, password){}

async registration(email, password){ }

async logout(){ }

setLoading(bool){ }

async checkAuth(){}

}

Листинг 3.7 – Объект store

Хранилище будет хранить состояния об авторизации. К этому состоянию будет относиться JWT-токен авторизации. Также хранилище будет хранить информацию о пользователях. К этой информации относится объект с пользователем, который на данный момент пользуется приложением.

Компонент Chat отвечает за чат поддержки. Для связи клиента с сервером в этом компоненте используется WebSocket. Взаимодействие происходит по WSS (WebSockets Secure). Клиент может отправлять запросы перечисленный в команде help или отправлять баг репорты. Реализация компоненты находится в листинге 3.8.

import { observer } from 'mobx-react-lite';

import { useState, useEffect } from 'react';

function Chat() {

const [ws, setWs] = useState(null);

const [message, setMessage] = useState('');

const [chatHistory, setChatHistory] = useState('Введите /Help\n');

useEffect(() => {

const websocket = new WebSocket('wss://localhost:5000');

websocket.onopen = () => {

};

websocket.onmessage = (event) => {

};

websocket.onclose = () => {

};

websocket.onerror = (error) => {

};

// Закрытие соединения при размонтировании компонента

return () => {

if (websocket) {

websocket.close();

}

};

}, []); // Выполняется только при монтировании

const bugReport = () => {}

const requestSupportChat = () => {};

return (}

export default observer(Chat);

Листинг 3.7 – Компонент Chat

Хук useEffecct выполняется один раз при запуске страницы. При закрытии страницы выполняется закрытие соединения.

## 3.2 Разработка серверной части web-приложения

Серверная часть написана на фреймворке Express для платформы NodeJS. Данный фреймворк позволил разработать множество модулей.

Первый модуль – это Index – самый главный модуль, который является управляющим файлом сервера. Этот модуль занимается тем, что подключает роутеры, содержит реализацию веб-сокета, настраивает политику cors, устанавливает соединение с базой данных и запускает сервер.

Сервер состоит из следующих компонентов: модели базы данных, сервисы для управления моделями базой данных, контроллеры для обработки запросов и передачи информации в сервисы, middlewares для обработки доступа запросов и обработки ошибок, роуты для маршрутизации запросов, компонент содержащий dto классы для преобразования данных.

Модуль Mongoose – это динамический модуль, который позволяет создавать подключение к NoSql базе данных MongoDB. Для MongoDB было разработано шесть схем: CPUShema, GPUShema, MotherboardShema, PCShema, ReviewShema, TokenShema, UserShema. Основные схемы без которых сайт не будет работать это UserShema и TokenShema их реализацию можно увидеть в листинге 3.8 и 3.9 соответственно.

const {Schema, model} = require('mongoose');

const UserSchema = new Schema({

email:{type:String, unique:true, required: true},

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

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

isActivated:{type:Boolean, default:false},

activationLink:{type:String},

banned:{type:Boolean, default:false}

})

module.exports = model("User", UserSchema)

Листинг 3.7 – Схема UserShema

В данной схеме есть поля email и password отвечающие за авторизацию пользователя. Поле banned отвечает за блокировку пользователя. isActivated и actvationLink за активацию аккаунта.

const {Schema, model} = require('mongoose');

const TokenSchema = new Schema({

user:{type:Schema.Types.ObjectId, ref: 'User'},

refreshToken:{type:String, required: true}

})

module.exports = model("Token", TokenSchema)

Листинг 3.7 – Схема TokenShema

В данной схеме мы храним идентификатор пользователя и его refreshToken, таким способом реализован белый список токенов.

За взаимодействие с таблицей пользователя, авторизацию, регистрацию, обновление токена, выход из аккаунта, активацию аккаунта отвечает UserService. Реализация этого класса показана в листинге 3.8.

const UserModel = require('../models/user-model')

const bcrypt = require('bcrypt')

const uuid = require('uuid')

const mailService = require('./mail-service')

const tokenService = require('./token-service')

const UserDto = require('../dtos/user.dto')

const ApiError = require('../exceptions/api-error');

class UserService{

async registration(email,password, role){

}

async activate(activationLink){

}

async login(email, password){

}

async logout(refreshToken){

}

async refresh(refreshToken){

}

async getAllUsers(){

}

async banStatusUser(\_id, banned){

}

}

module.exports = new UserService();

Листинг 3.8 – Класс UserService

Функция banStatusUser отвечает за блокировку и разблокировку пользователя. С помощью функций login, refresh выполняется авторизация и обновление токена пользователя. Этот класс использует функции из класса token-service, который отвечает за проверку валидности refresh и access токенов, сохранения токена в базу данных, генерацию токенов, удаления токена и его поиск.

Ограничения доступа к ресурсу выполняется при помощи Middlewares auth-middleware и role-middleware. Первый отвечает за проверку валидности токена, второй за проверку уровня доступа.

Пример контроллера для обработки запросов об компьютерных конфигурациях показан в листинге 3.9. В этом контроллере реализован функционал создания, удаления, публикации и обновления конфигурации.

const ApiError = require('../exceptions/api-error');

const {PCService, PC} = require('../service/pc-service');

const pcService = new PCService();

class PCController{

async createPC(req, res, next){ }

async getPCs(req, res, next){

}

async getPublishedPCs(req, res, next){

try{

const PCs = await pcService.readPublishedPCs();

return res.json(PCs);

}

catch(e){

next(e);

}

}

async getCurrentPC(req, res, next){

const {id} = req.params;

try{

const PCs = await pcService.readCurrentPC(id);

return res.json(PCs);

}

catch(e){

next(e);

}

}

async getUserPCs(req, res, next){

}

async updatePC(req, res, next){

}

async deletePC(req, res, next){

}

}

module.exports = new PCController();

Листинг 3.9 – Контроллер pc-controller

В этом листинге можно увидеть примеры реализации получения всех конфигурации или конфигураций пренадлежащих конкретному пользователю и т.д.

## 3.3 Выводы по разделу

В данном разделе мы рассмотрели основные момент разработки веб-приложения. Рассмотрели, как построено взаимодействие клиента с сервером по протоколу HTTP и по протоколу WebSocket. Рассмотрели, как хранится информация, в каких БД она хранится и как отображается на клиенте. Рассмотрели, как построена безопасность данного веб-приложения.

# 4 Тестирование web-приложения

Вначале нужно протестировать форму регистрации, введя неверные данные, а именно в поле логин или пароль укажем пробелы. Ожидаем неудачную попытку регистрации 4.1.

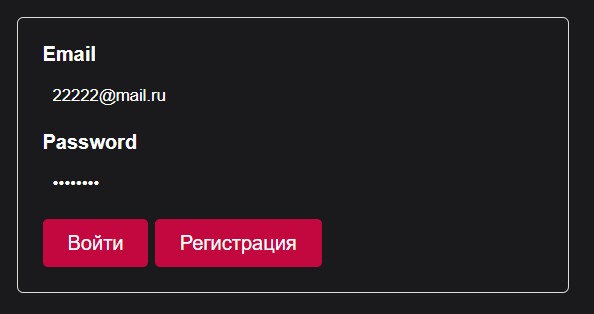


Рисунок 4.1 – Результат проделывания первого теста

По картинке 4.1 видно, что система отработала правильно и не дала зарегистрировать аккаунт.

Второй тест будет связан с созданием пустой конфигурации компьютера. Если отправить пустую конфигурацию, она не должна создаться. Результат теста показан на рисунке 4.2.

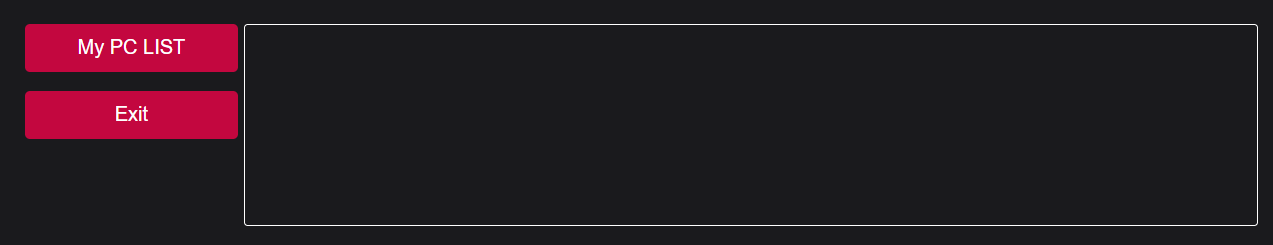


Рисунок 4.2 – Результат проделывания второго теста

По результатам второго теста, видно, что приложение отработало верно.

Третий тест будет заключатся в проверке сервера. Мы воспользуемся токеном авторизации для пользователя, который не является администратором и попробуем получить список всех пользователей. Ожидаемый результат – это ошибка «403» неавторизованный доступ. Результат проделывания теста три показан на рисунке 4.3.

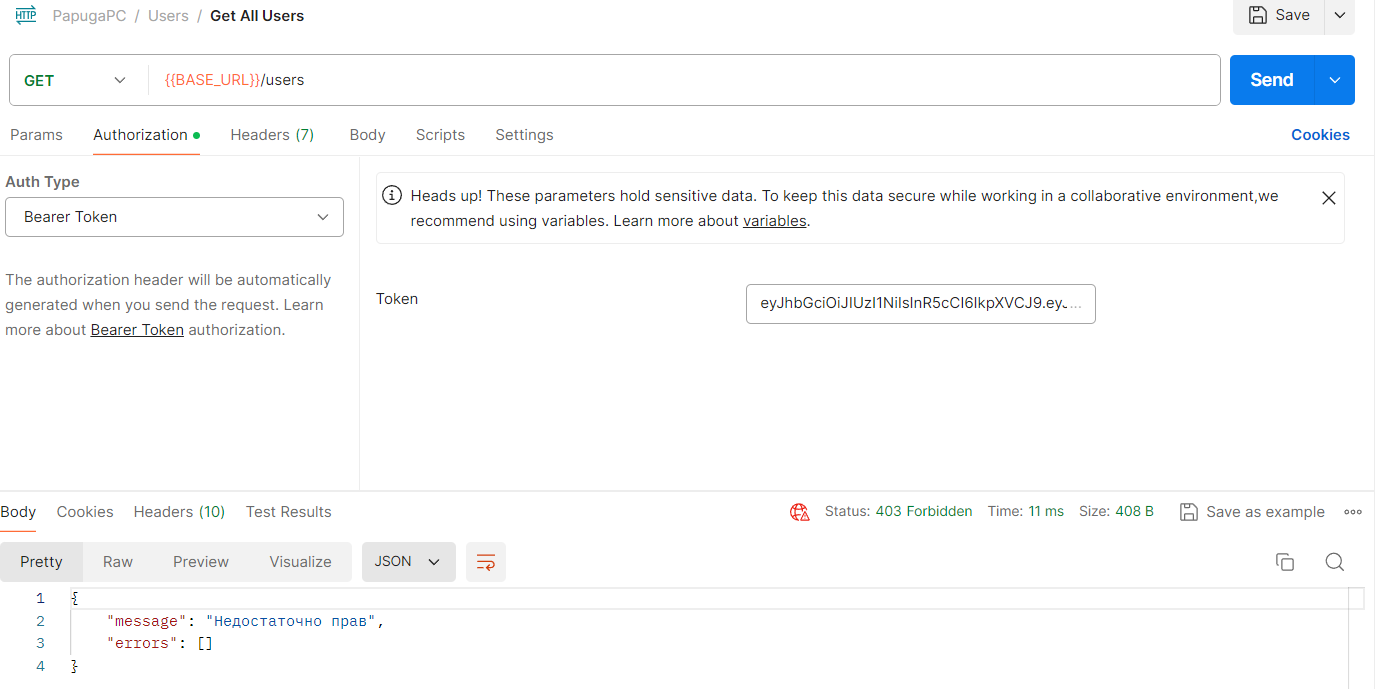


Рисунок 4.3 – Результат проделывания третьего теста

По результатам третьего теста можно сказать, что role-middleware полностью отрабатывает и, что приложение «конфигуратор компьютеров» защищено от неавторизированного доступа.

Четвертый тест будет заключатся в том, что мы попробуем изменить существующую конфигурацию на пустую строку. Ожидаемый результат – редактирование не пройдет. Результат данного тест показан на рисунке 4.4.

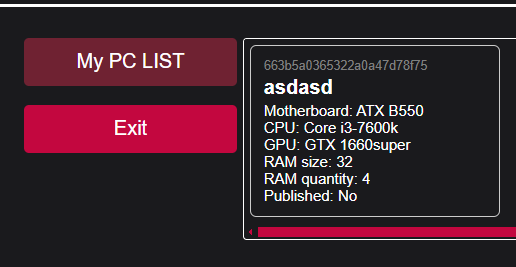


Рисунок 4.4 – Результат проделования четвертого теста

Результат четвертого теста соответствует ожидаемому результату.

Пятый тест заключается в том, что мы отправим пустой отзыв на конфигурацию. Ожидаемый результат – отзыв не появится. Результат данного теста будет показан на рисунке 4.5.

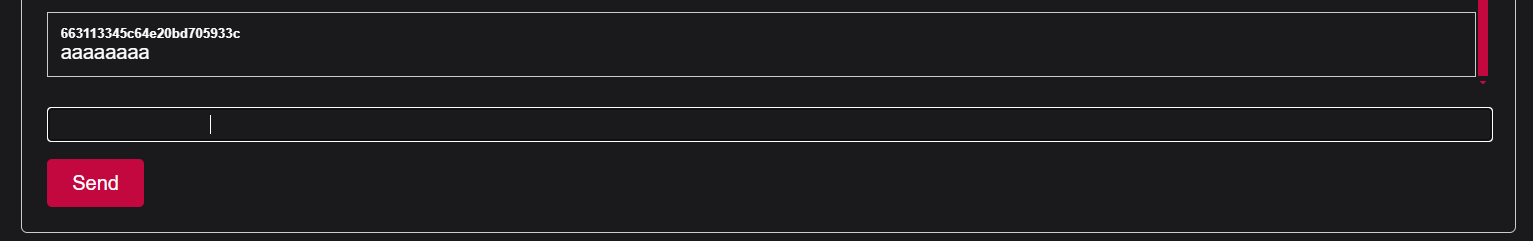


Рисунок 4.5 – Результат проделования пятого теста

Результат пятого теста, как и результаты всех остальных тестов, показал, что приложение функционирует правильно и безопасно. Данное приложение готово к публикации и работе в сети Интернет.

# 5 Руководство пользователя

## 5.1 Руководство пользователя

При открытии веб-приложения «Конфигуратор компьютеров» вас будет встречать форма авторизации, в которой вы можете ввести данные и зайти под определенным пользователем или же вы можете нажать на кнопку «Зарегистрироваться» и зарегистрировать аккаунт. Рисунок формы авторизации представлен на рисунке 5.1.

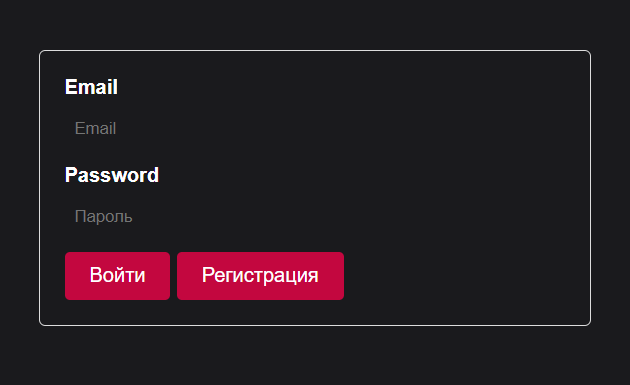


Рисунок 5.1 – Форма авторизации

После входа под определенным пользователем вам откроется главная страница приложения, где вы сможете увидеть профиль пользователя и меню сайта. Также здесь видно, отзывы которые отправлял пользователь и его конфигурации. В правом нижнем углу присутствует форма для создания конфигурации компьютера. Главная страница представлена на рисунке 5.2.

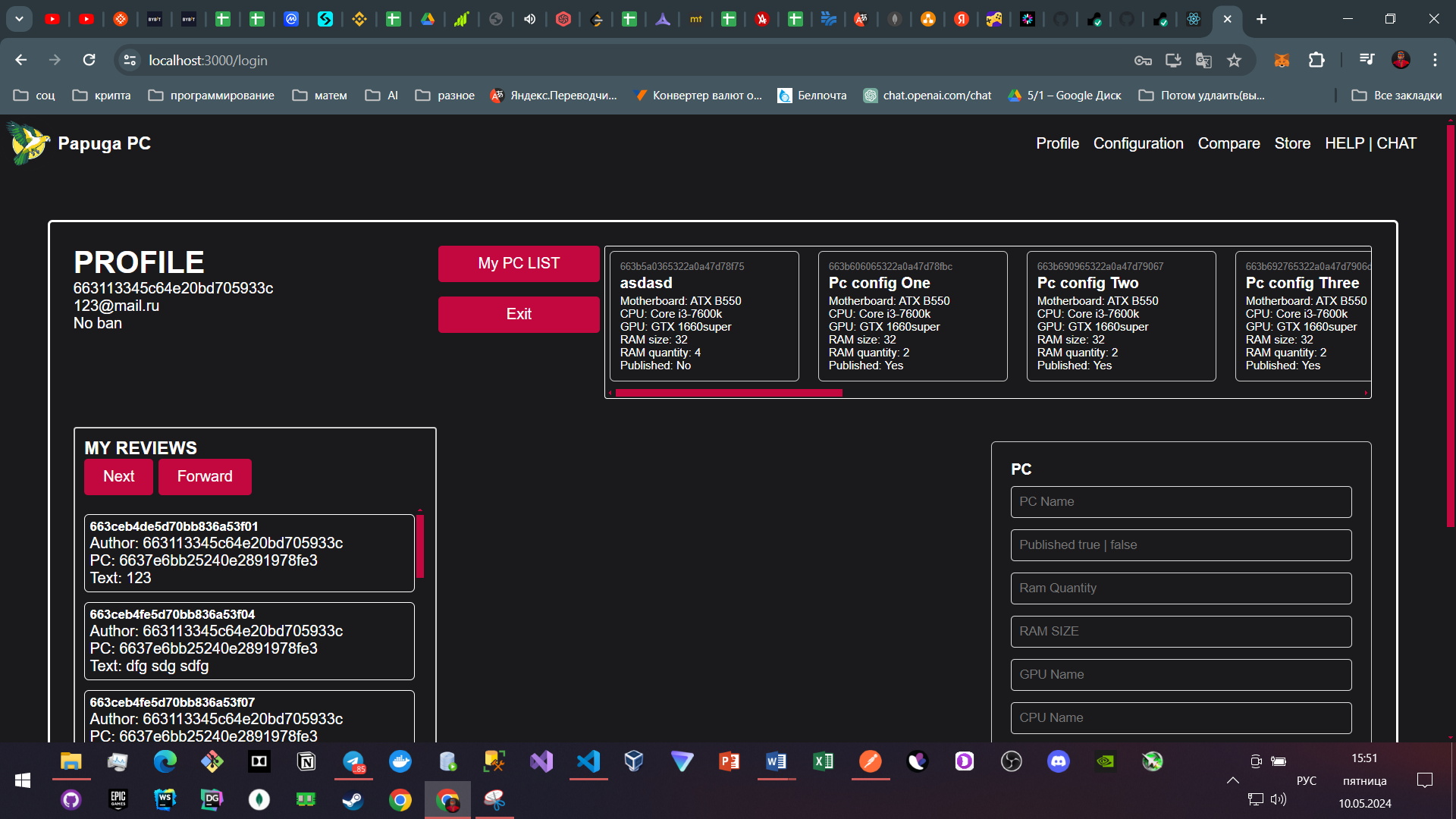


Рисунок 5.2 – Главная страница

Пользователь может создать новый аккаунт. Для этого ему надо перейти на страницу авторизации, где ему будет доступна интуитивно понятная форма. Рисунок формы регистрации представлен на рисунке 5.3.

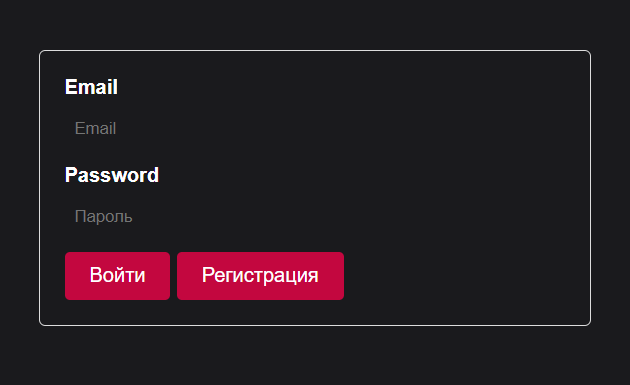


Рисунок 5.3 – Страница регистрации

Пользователь может сравнивать комплектующие во вкладке compare, страница показана на рисунке 5.4.

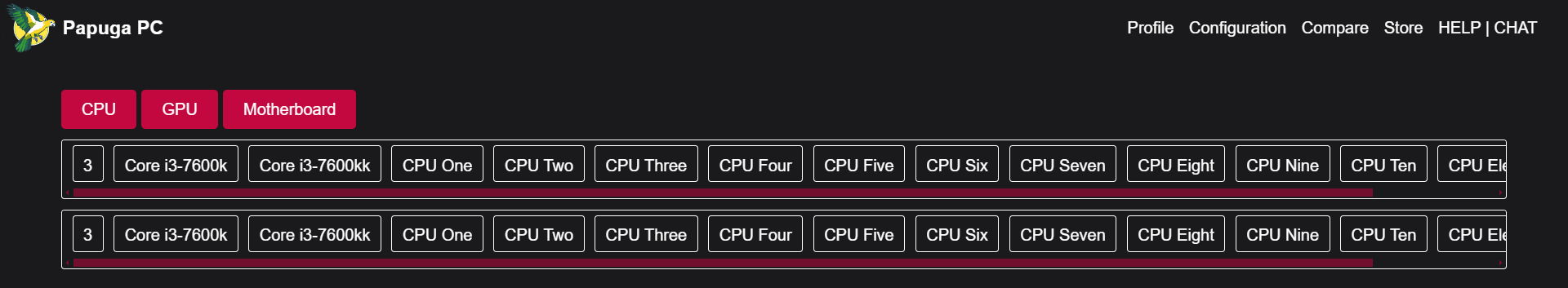


Рисунок 5.4 – Меню выбора комплектующих для сравнения

Для сравнения комплектующих надо на элемент из первого списка и на элемент из второго списка, страница показана на рисунке 5.5.

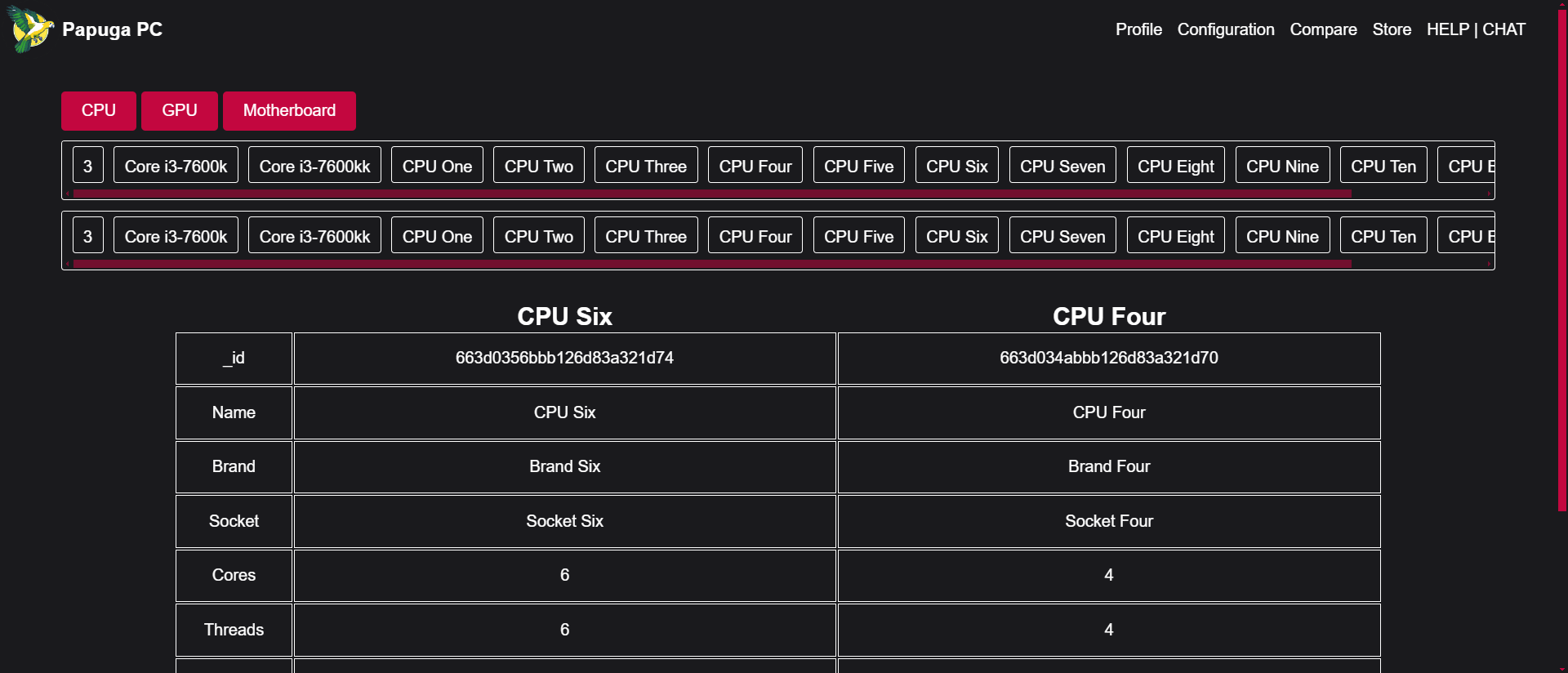


Рисунок 5.5 – Страница сравненяи комплектующих

Пользователь может посмотреть конфигурации компьютера во вкладке Configuration, страница показана на рисунке 5.6.

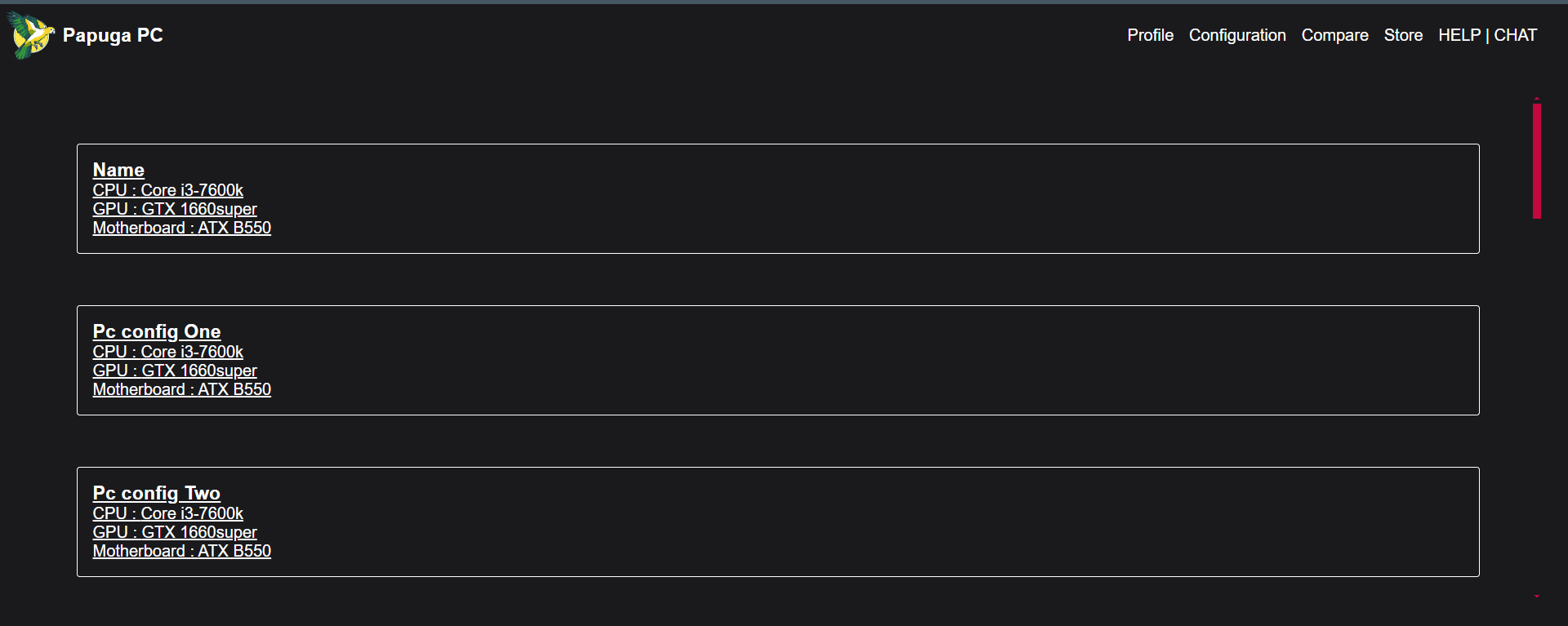


Рисунок 5.6 – Список конфигураций

Пользователю доступны функции создания отзыва на конфигурацию. При выборе конкретной конфигурации в списке конфигурации открывается страница с выбранной пользователем сборкой. Пример можно увидеть на рисунке 5.7.



Рисунок 5.7 – Страница с отзывами

В разделе Store пользователь может просматривать все доступные комплектующие на сайте. При нажатии на кнопки открывается определённы раздел с комплектующими. В списке комплектующих можно ознакомится с характеристиками комплектующих. Пример можно увидеть на рисунке 5.8.

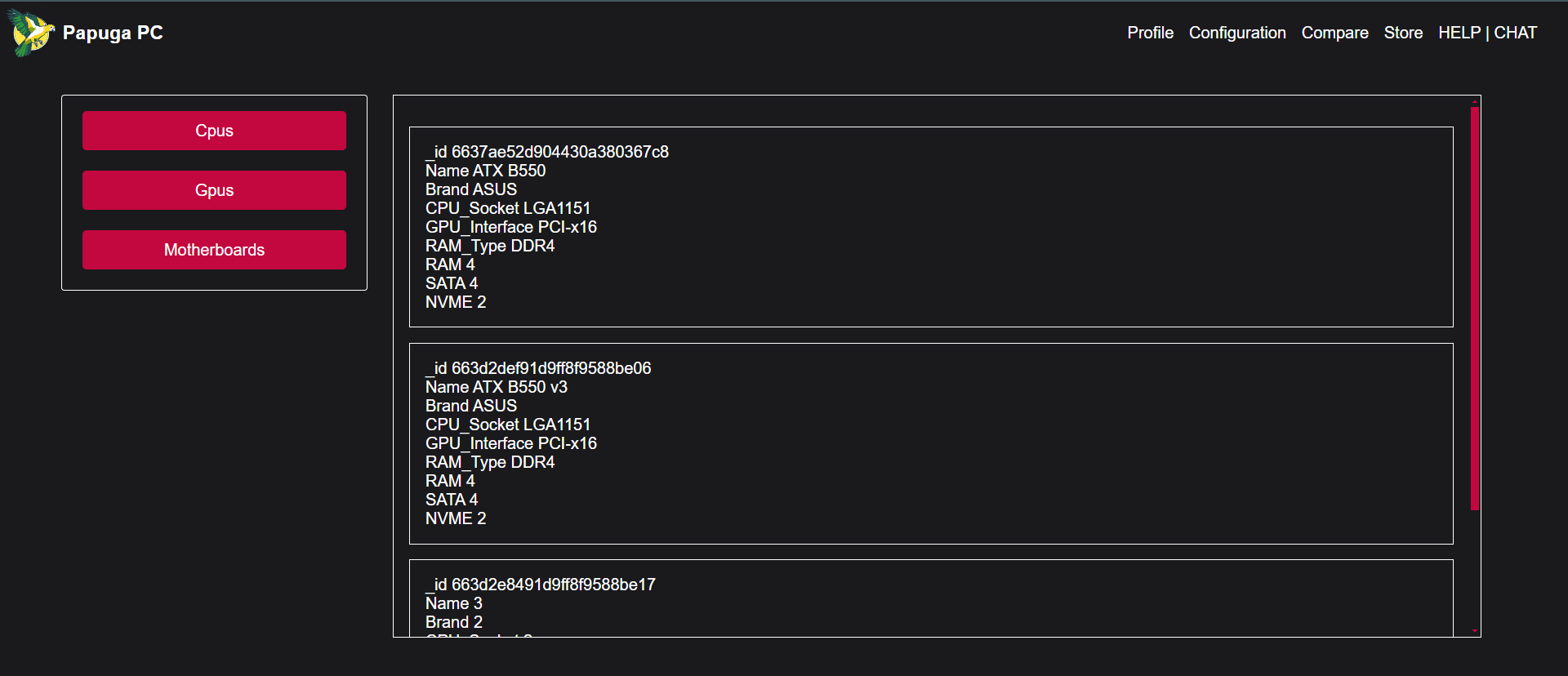


Рисунок 5.8 – Страница Store

Если пользователь захочет создать конфигурацию компьютера, ему нужно будет перейти в раздел profile. Заполнить форму и нажать на кнопку добавить. Пример представлен на рисунке 5.9.

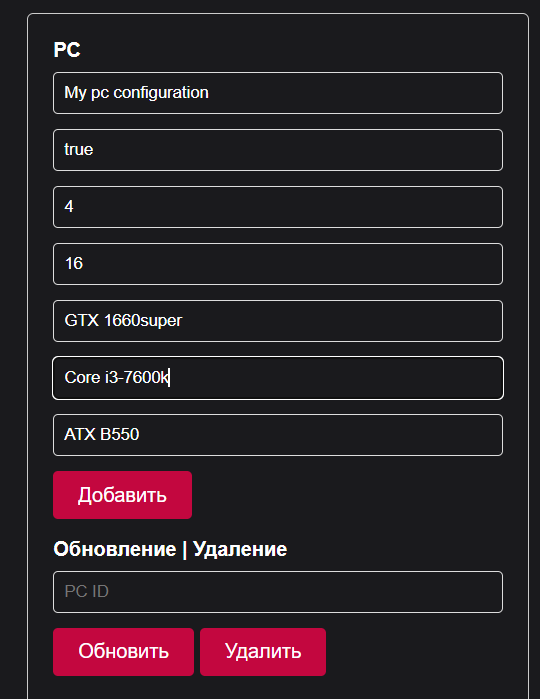


Рисунок 5.9 – Форма редактирования

Если пользователь захочет удалить конфигурацию ему надо будет в списке своих конфигураций найти идентификатор конфигурации, пример на рисунке 5.10. Ввести этот идентификатор в поле PC ID и нажать на кнопку удалить, пример на рисунке 5.11.

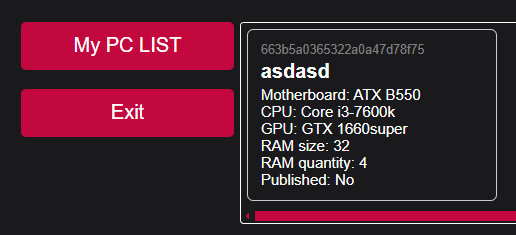


Рисунок 5.10 – Блок с конфигурациями пользователя

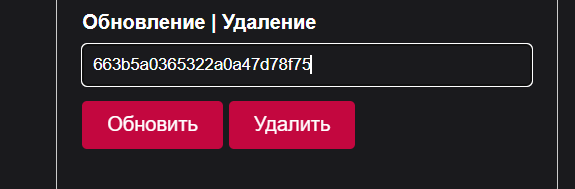


Рисунок 5.11 – Блок с удаление конфигурации

Пользователь может изменить данные о публикации в той же форме или управлять публикацией конфигурации. Для обновления конфигурации пользователю требуется ввести новые данные в поля и нажать кнопку обновить. Пример представлен на рисунке 5.12.

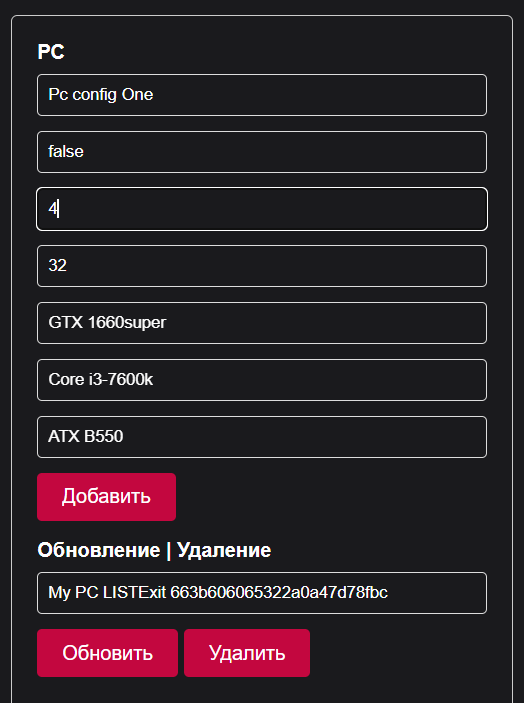


Рисунок 5.12 – Блок с обновлением конфигурации

Для выхода из аккаунта требуется перейти на страницу профиля и нажать на кнопку Exit. Пример показан на рисунке 5.13.

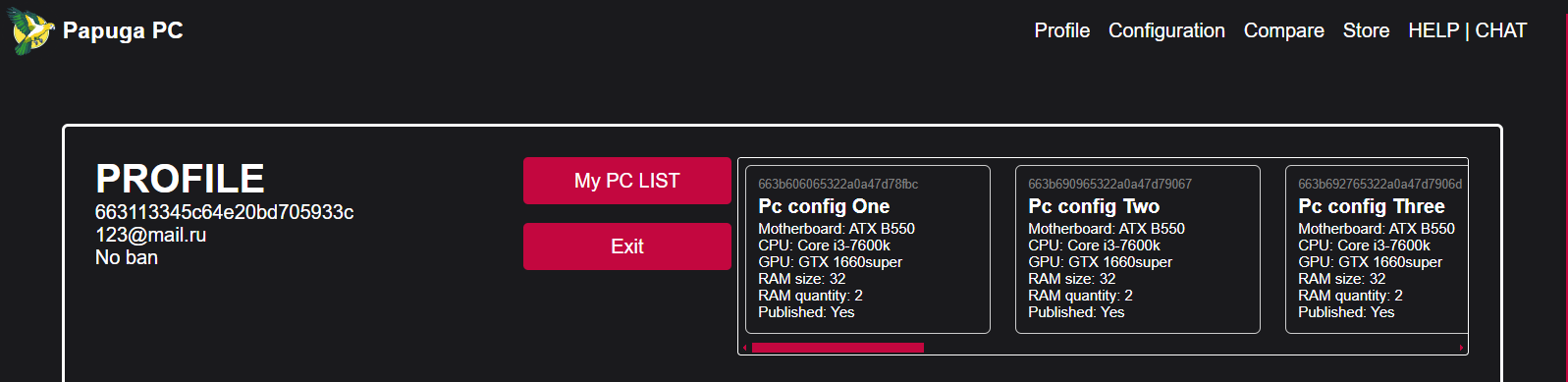


Рисунок 5.13 – Блок с кнопкой выход

Рассмотрим функционал администратора. При авторизации в аккаунт от пользователя с ролью администратора отличия от интерфейса будут в одной вкладке. Вид навигационно панели от лица администратора можно увидеть на рисунке 5.14.

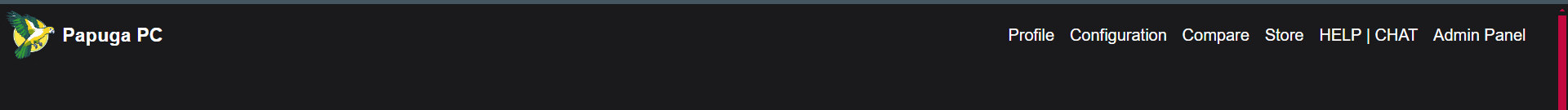


Рисунок 5.14 – Навигационная панель от лица администарота

Для добавления новых комплектующих или удаления старых администратор должен перейти во вкладку Admin Panel. Для управления комплектующими у администратора есть три формы. Фома для управление процессорами, форма для управления видеокартами и форма для управления материнскими платами. Таблицы представлены на рисунке 5.15.

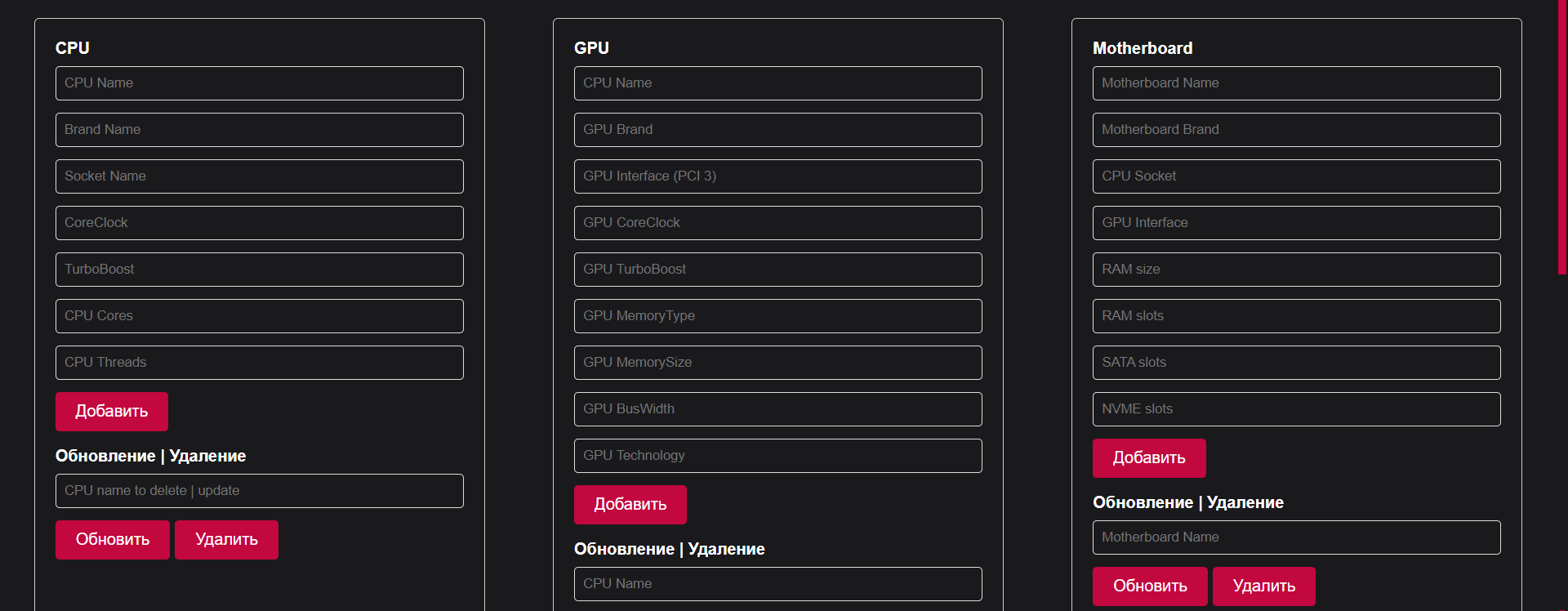


Рисунок 5.15 – Формы для управления комплектующими

Для блокировки или разблокировки аккаунта пользователя администратору требуется идентификатор пользователя. На странице Admin Panel в форму User, представленную на рисунке 5.16, администратор вводит идентификатор пользователя и выполняет блокировку или разблокировку пользователя.

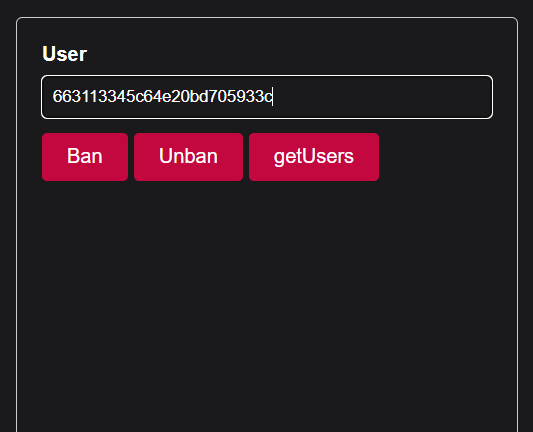


Рисунок 5.16 – Формы блокировки пользователя

Если заблокированный пользователь попробует зайти на сайт он увидит страницу, пример представлен на рисунке 5.17.

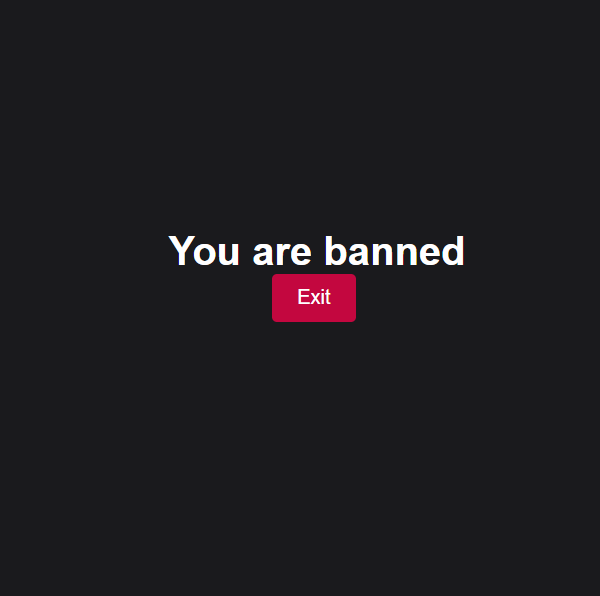


Рисунок 5.17 – Страница заблокированного пользователя

## 5.2 Установка приложения

Для запуска приложения необходимо выполнить следующие шаги:

1. Запустить серверную часть приложения, которая соединяет базу данных и React приложение. Для этого необходимо запустить скрипт, который настроит соединение с базой данных и запустит сервер.

2. Запустить React приложение, которое будет обрабатывать пользовательские запросы и взаимодействовать с сервером. Для этого необходимо запустить команду для сборки и запуска React приложения.

После выполнения этих шагов приложение будет полностью готово к работе и пользователь сможет начать использовать его функционал.

## 5.3 Выводы по разделу

В данном разделе мы рассмотрели, как пользоваться приложением "Конфигуратор компьютера" с различными уровнями доступа: обычным пользователем и администратором. Обычный пользователь имеет определенный функционал, позволяющий создавать и сохранять конфигурации компьютера, а также просматривать доступные комплектующие.

Администратор, в свою очередь, обладает расширенными возможностями, такими как добавление новых комплектующих в базу данных, управление пользователями (включая блокировку и разблокировку учетных записей), удаление пользовательских конфигураций или управление их публикацией.

Добавление новых комплектующих позволяет расширить ассортимент предлагаемых компонентов для конфигурации компьютера, что делает приложение более гибким и актуальным для пользователей.

Управление пользователями (блокировка, разблокировка) дает возможность администратору контролировать доступ и поведение пользователей в приложении, обеспечивая безопасность и соблюдение правил использования.

Удаление пользовательских конфигураций или управление их публикацией позволяет поддерживать порядок в системе и обеспечивать качество контента, предоставляемого пользователями.

Таким образом, различные функции для обычных пользователей и администраторов приложения "Конфигуратор компьютера" обеспечивают удобство использования, контроль и безопасность работы с приложением, делая его полезным инструментом как для конечных пользователей, так и для администраторов системы.

# Заключение

Приложение «Конфигуратор компьютера» – полностью функциональное приложения для создания конфигурации компьютера. Оно позволяет пользователями создавать конфигурации, сравнивать комплектующие, обращаться в чат поддержки и просматривать список существующих комплектующих.

За серверную часть отвечала платформа NodeJS, а именно фреймворк Express, который позволил реализовать быструю асинхронную серверную часть. За счет паттерна проектирования «MVC» структура серверной части стала выглядеть как дерево, что позволило легко соединять модули, делать их независимыми от других модулей.

Клиентская часть была написана при помощи библиотеки ReactJS для языка JavaScript.

База данных MongoDB позволила создать структуру, которая позволила эффективно хранить взаимосвязанные данные. База данных содержит функционал для работы с ней, также в ней описаны роли, которые ограничивают доступ к данным.

Тестирование проекта показало, что приложение функционирует правильно и все поставленные тесты прошло. В ходе тестирования ошибок не было найдено.

Этот функционал делает приложение более гибким и удобным как для обычных пользователей, так и для администраторов, обеспечивая эффективное управление и персонализированный опыт использования. Такой подход позволяет создать платформу, которая соответствует разнообразным потребностям и требованиям пользователей, обеспечивая им удобство и функциональность в рамках конфигурирования компьютеров.

Данной приложение готово к эксплуатации и использованию в сети Интернет.

# Список используемых источников

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com – Дата доступа: 15.04.2023.

3. React.js документация [Электронный ресурс] / Режим доступа: https://react.dev/learn – Дата доступа: 26.03.2023.

4. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 08.03.2023

5. Продвинутый BACKEND на Node.js и Nest.js from Zero [Электронный ресурс] / Режим доступа: https://www.youtube.com/watch?v=dDeWWQWMM-Y&list=PLvQU2QK9kDm6qqI7s9WYTuj1JQpbWNhHq&index=2