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

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

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,профессор

_____Д.В. Топольский

«___»_____2024 г.

Веб-сервис по подбору комплектующих в сети Интернет для обновления ПК

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2024.405 ПЗ ВКР

Научный руководитель,
кандидат технических наук,
доцент
_____В.А. Парасич

Автор работы,
студент группы КЭ-405
_____Фомин К. И.

Нормоконтролёр,
ст. преп каф. ЭВМ

_____С.В. Сяськов
«___»_____2024 г.

Челябинск-2024

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

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Фомину Константину Игоревичу,
обучающемуся по направлению
09.03.01 Информатика и вычислительная техника»

1. Тема работы: «Веб-сервис по подбору комплектующих в сети Интернет для обновления ПК» утверждена приказом по университету от 22 апреля 2024 г. №764-13/12

2. Срок сдачи студентом законченной работы: 01 июня 2024 г.

3. Исходные данные к работе

3.1. Официальный сайт DNS. [Электронный ресурс]. URL: https://www.dns-shop.ru/configurator/ (дата обращения: 14.04.2024 г.).

3.2. Официальный сайт IronBook. [Электронный ресурс]. URL: https://www.ironbook.ru/ (дата обращения: 14.04.2024 г.).

3.3. Официальный сайт RoyalPC. [Электронный ресурс]. URL: https://royal-computers.ru/configurator (дата обращения: 14.04.2024 г.).

3.4. Metanit.com. | Сайт о программировании [Электронный ресурс] URL:http://metanit.com/. (дата обращения 28.02.2024 г.);

3.5. Django.fun | Официальная документация [Электронный ресурс] URL:https://django.fun/docs/django/5.0/ . (дата обращения 20.03.2024 г.);

3.6. JavaScript документация. [Электронный ресурс]. URL:https://learn.javascript.ru/ (дата обращения: 06.03.2024 г.);
3.7. HTML документация. [Электронный ресурс]. URL:https://hcdev-ru.pages.dev/html/ (дата обращения: 06.03.2024 г.);
3.8. CSS документация. [Электронный ресурс]. URL:https://webref.ru/css (дата обращения: 06.03.2024 г.);
3.9. Git документация. [Электронный ресурс]. URL:https://git-scm.com/doc (дата обращения: 06.03.2024 г.);
3.10. Figma документация. [Электронный ресурс]. URL:https://jestjs.io/docs/getting-started (дата обращения: 06.03.2024 г.).

4. Перечень подлежащих разработке вопросов

4.1 Произвести обзор веб-сервисов и литературы по предметной области.
4.2 Спроектировать веб-сервис.
4.3 Реализовать веб-сервис и базу данных.
4.4 Произвести тестирование веб-сервиса.
4.5 Подготовить текст ВКР.

5. Дата выдачи задания: 1 декабря 2023 г.

Руководитель работы _____/ В.А. Парасич /

Студент _____/ К.И. Фомин /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2024	
Сформулировать цели и задачи, которые должно выполнять веб-приложение. Определить целевую аудиторию, их потребности и ожидания.	07.03.2024	
Исследовать рынок для поиска аналогов приложения, определить преимущества и недостатки.	14.03.2024	
Создать графический прототип веб-приложения, визуализировать и протестировать различные элементы дизайна и опыта пользователя	01.04.2024	
Определить технологии для разработки веб-приложения, которые смогут масштабироваться и разрабатываться в дальнейшем другими разработчиками	3.04.2024	
Проектирование архитектуры	10.04.2024	
Реализовать серверную часть веб-приложения. Подключить базу данных, разработать api – методы, которые будут доступны для использования в приложении	01.05.2024	
Реализовать клиентскую часть веб-приложения. Разработать удобное взаимодействие с калькулятором расчета окон	17.05.2024	
Провести тестирование веб-приложения	20.05.2024	
Компоновка текста работы и сдача на нормоконтроль	24.05.2024	
Подготовка презентации и доклада	30.05.2024	

Руководитель работы _____/ *В.А. Парасич* /

Студент _____/ *К.И. Фомин* /

АННОТАЦИЯ

Фомин К.И. Веб-сервис по подбору комплектующих в сети Интернет для обновления ПК– Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 40 с., библиогр. список – 18 наим.

В ходе выполнения дипломной работы был проведен аналитический обзор современной научно-технической, нормативной и методической литературы в области конфигураторов персональных компьютеров.

Был разработан и реализован веб-сервис для подбора комплектующих ПК в сети Интернет. Для этой цели была создана специализированная база данных, позволяющая осуществлять подбор компонентов и формировать конфигурации ПК на основе заданных пользовательских параметров.

В процессе выполнения тестов была проверена функциональность веб-сервиса и на основании результатов тестирования были получены корректные конфигурации компонентов ПК.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы работы

В современном технологическом мире персональные компьютеры (ПК) стали неотъемлемой частью нашей повседневной жизни. Сфера применения ПК невероятно широка, охватывая все аспекты работы, учебы, развлечений и общения. Однако выбор идеального ПК, отвечающего индивидуальным потребностям, предпочтениям и бюджету, может быть сложной задачей.

Конфигуратор ПК – это ценный инструмент, который упрощает и оптимизирует процесс выбора и сборки ПК. Он предоставляет пользователям интерактивную платформу, где они могут выбирать и настраивать различные компоненты ПК, такие как процессор, материнская плата, оперативная память, видеокарта и устройства хранения данных.

В условиях увеличивающейся конкуренции на рынке компьютеров потребители все более внимательно относятся к соотношению цены и качества. Разработка конфигуратора, учитывающего бюджет, позволяет удовлетворить спрос на более доступные и экономичные варианты ПК.

В этом дипломном проекте будет изучена концепция конфигуратора ПК, его преимущества и ограничения. Будет проведен анализ различных типов конфигураторов ПК, доступных на рынке, и будут исследованы факторы, влияющие на выбор оптимального конфигуратора. Кроме того, будет разработан и реализован собственный веб-сервис по подбору комплектующих ПК с использованием современных технологий.

Исследование позволит получить более глубокое понимание роли конфигураторов ПК в процессе выбора и сборки ПК, а также их потенциала для улучшения пользовательского опыта. Результаты этого проекта будут полезны как для пользователей, ищущих эффективный способ настройки своих ПК, так и для разработчиков, стремящихся создать более совершенные и удобные конфигураторы ПК.

Цели и задачи:

1. Изучение рынка и анализ требований пользователей. Проведение исследования существующих конфигураторов ПК и определение основных функциональных и дизайнерских требований к Вашему проекту.

2. Проектирование интерфейса конфигуратора. Разработка удобного и интуитивно понятного интерфейса, который позволит пользователям легко выбирать компоненты для своего ПК.

3. Разработка базы данных компонентов. Создание базы данных, содержащей информацию о различных компонентах ПК (процессоры, видеокарты, оперативная память и т.д.) с их характеристиками и ценами.

4. Реализация логики выбора компонентов. Написание алгоритмов, которые будут учитывать совместимость компонентов, бюджет пользователя и другие ограничения при создании конфигурации ПК.

5. Тестирование и отладка. Проведение тестирования конфигуратора, выявление и исправление ошибок, а также оптимизация производительности.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор аналогов

На данный момент существует множество различных веб-сервисов по сборке компьютера. Рассмотрим некоторые из них, связанные с темой работы.

1.1.1. Конфигуратор DNS

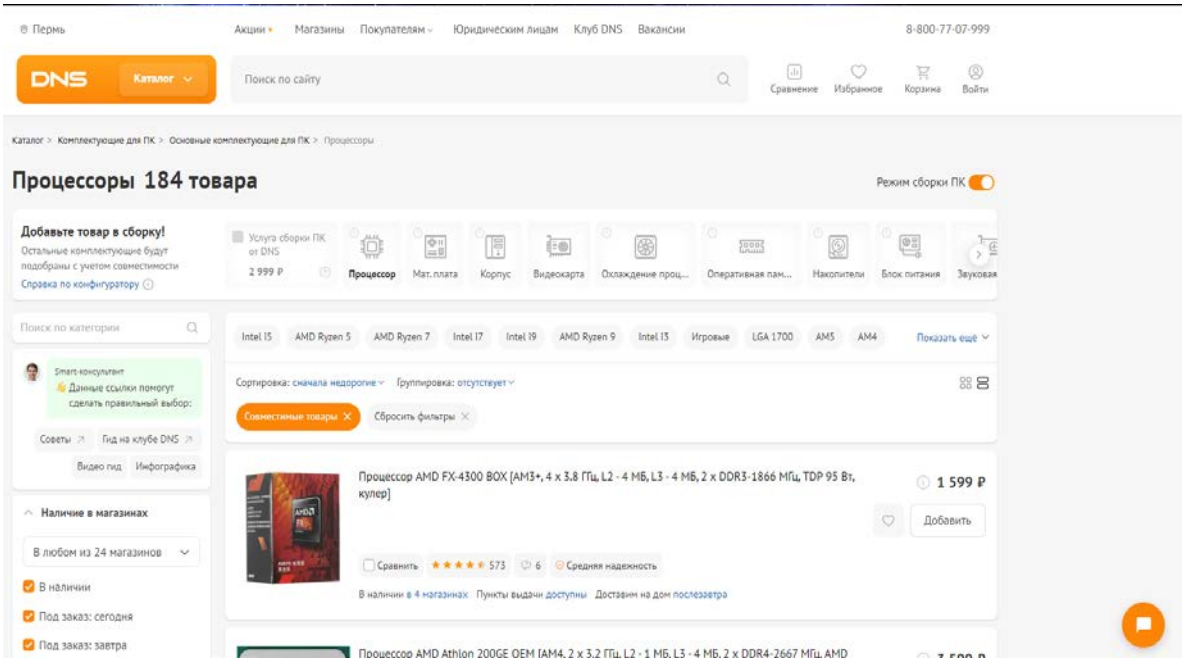

Рисунок 1 – Конфигуратор DNS

На представленном рисунке можно наблюдать блоки выбора комплектующих, с кратким указанием их характеристик, которые включают в себя цену и наличие товара. Указывается общая стоимость всех комплектующих. Также имеется функция сборки в магазине за дополнительную плату.

К плюсам можно отнести привязанность конструктора к одному интернет-магазину. В конкретном случае, плюсом является то, что любые изменения в товарах магазина, будут тут же применены к конфигуратору, таким образом пользователь может не беспокоиться о различиях цен, наличии товара и т.п. Несмотря на то, что данный конфигуратор имеет много положительных сторон, привязанность к конкретному магазину также плохо отражается на его функционале. Большое наличие различных комплектующих не может конкурировать с возможностью выбора определенного товара в

разных магазинах для сравнения цены и качества. Из этого следует отсутствие возможности просмотра сборок, основанных на товаре, которыми данный интернет-магазин не торгует, а также не возможность подбора оптимального по цене и качеству персонального компьютера.

1.1.2. Конфигуратор “RoyalPC”

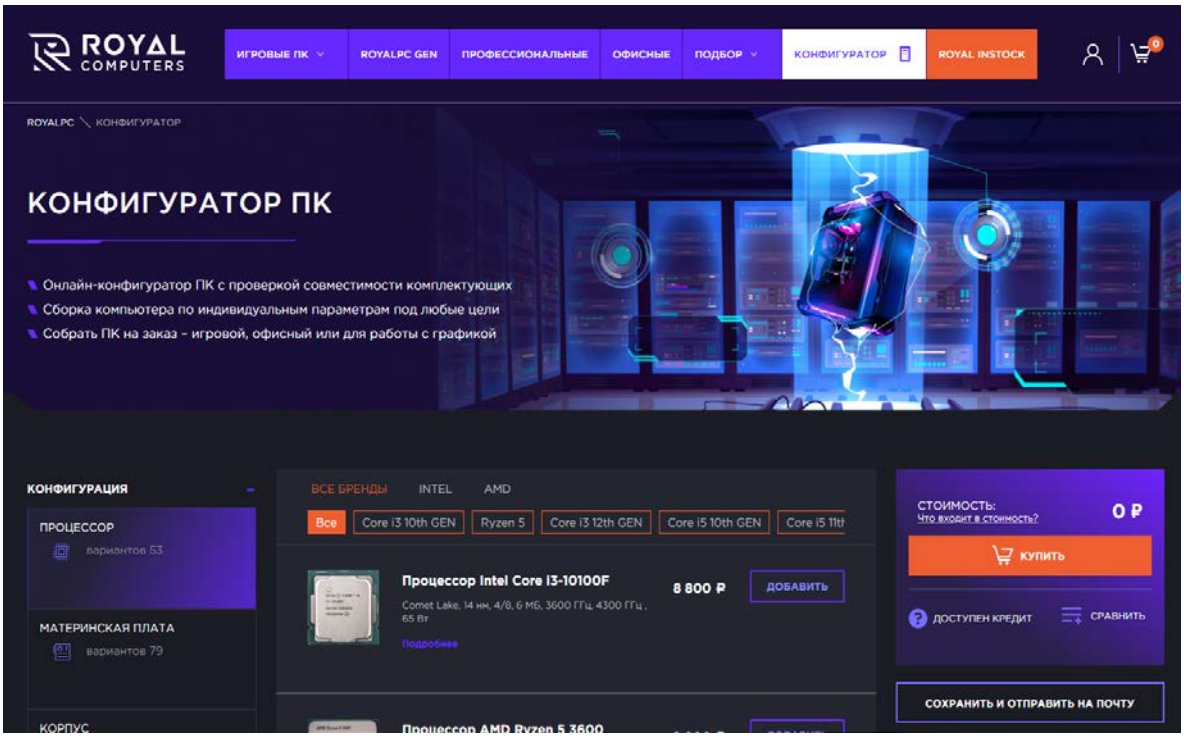

Рисунок 2 – Конфигуратор "RoyalPC"

«RoyalPC» ресурсоемкое приложение с большими вариантами кастомизации ПК. Он предлагает обширные обзоры компонентов ПК, включая процессоры, видеокарты, материнские платы и периферийные устройства. Каждый обзор содержит технические характеристики, сравнения производительности и рекомендации по выбору. Конфигуратор «RoyalPC» обладает сводкой данных в таблице с количеством кадров в секунду из представленного перечня игр. Помогая визуализировать поведение ПК в играх и, что не мало важно, облегчить выбор на той конфигурации, которая будет соответствовать ожиданиям покупателя. RoyalPC поддерживает активное сообщество, где пользователи могут обсуждать проблемы, делиться опытом и задавать вопросы. Это делает сайт не только источником информации, но и площадкой для обмена мнениями. На сайте присутствует раздел советов и

руководств, где пользователи могут найти полезные советы по настройке и улучшению производительности своих компьютеров, а также подробные инструкции по уходу за ними. Скриншот конфигуратора проиллюстрирован на рисунке.

1.1.3. IronBook

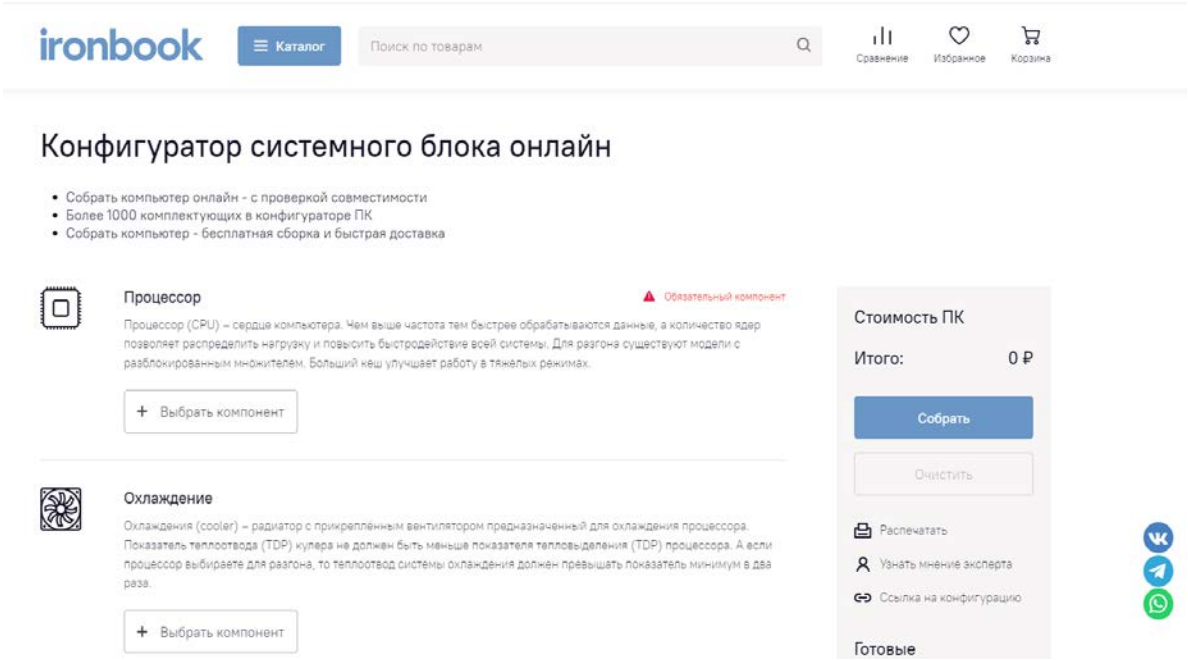

Рисунок 3 – Конфигуратор "IronBook"

IronBook.ru – это веб-сервис, который предоставляет пользователям возможность подбора комплектующих для персональных компьютеров.

Конфигуратор на сайте IronBook также обладает простым и интуитивно понятным интерфейсом, с легкой навигацией по различным категориям товаров. Сайт обеспечивает доступ к широкому ассортименту комплектующих от ведущих производителей. Категории товаров охватывают все основные компоненты ПК, а также периферийные устройства и аксессуары. Конфигуратор на IronBook.ru схож по своим возможностям с такими платформами, как DNS и RoyalPC, что подчеркивает его эффективность и востребованность на рынке компьютерных комплектующих и сборки ПК.

Вывод по первой главе

Таблица 1 – Сравнительная таблица аналогов

Характеристика	DNS	RoyalPC	IronBook
Интерфейс	Прост и интуитивно понятен	Современный и интуитивно понятный дизайн	Прост и интуитивно понятен
Функциональность	Широкий выбор комплектующих Готовые конфигурации ПК	Широкий выбор комплектующих, готовые конфигурации	Широкий выбор комплектующих
Адаптивность под моб. устройства	Поддерживает мобильный формат экрана	Поддерживает мобильный формат экрана	Поддерживает мобильный формат экрана
Интеграция с БД	Интеграция с базами данных	Интеграция с базами данных	Интеграция с базами данных

В свете общего сходства, выявленного в функциональности, предлагаемой конфигураторами DNS, RoyalPC и IronBook, следует отметить, что они демонстрируют аналогичные парадигмы в предоставлении услуг. Спектр функций, включающий разнообразие комплектующих, готовые конфигурации ПК, интуитивно понятный интерфейс и гарантии безопасной покупки, выступает как общий знаменатель для всех трех платформ. В то же время, разнообразия можно обнаружить в мельчайших деталях пользовательского интерфейса и в многообразии дополнительных сервисов, которые могут варьироваться в каждом из указанных конфигураторов. Все вышеупомянутые факторы неизменно подтверждают общую тенденцию к единому подходу в предоставлении услуг со стороны данных платформ.

2. ПРОЕКТИРОВАНИЕ

Для успешной разработки любого проекта критически важно четко определить требования, которые подразделяются на две основные категории: функциональные и нефункциональные.

Первые нужны для ответа на вопрос «что система делает?» , то есть какие функции и возможности должна предоставлять система для достижения бизнес-целей и удовлетворения потребностей пользователей. Эти требования ответственны за то, что система должна делать. Например, для веб-приложения по конфигурации персональных компьютеров функциональные требования могут включать возможности регистрации пользователей, выбора и сборки комплектующих, оформления заказа, а также управления аккаунтом пользователя.

Вторая группа требований отвечает на вопрос «как система это делает?», ответом на который она описывает какими свойствами будет обладать разработка и как пройдет процесс реализации. Примеры нефункциональных требований включают производительность системы (например, время отклика, максимальное число одновременных пользователей), надежность (например, уровень доступности, обработка ошибок), безопасность (например, защита данных, управление доступом), удобство использования (например, интуитивный интерфейс, поддержка адаптивного дизайна для разных устройств и браузеров), а также масштабируемость и совместимость с другими системами.

2.1. Функциональные требования

Программа должна соответствовать следующим функциональным требованиям:

1) выбор комплектующих: Пользователи должны иметь возможность выбирать комплектующие для персонального компьютера из широкого ассортимента, включая процессоры, видеокарты, материнские платы, оперативную память, накопители и другие;

2) информация о совместимости: Пользователям должна быть доступна информация о совместимости выбранных комплектующих между собой, чтобы избежать конфликтов и ошибок при сборке;

3) сохранение конфигураций: Пользователи должны иметь возможность сохранять созданные конфигурации для последующего просмотра, редактирования или покупки;

4) мобильная совместимость: Конфигуратор должен быть адаптирован для работы на мобильных устройствах, чтобы пользователи могли создавать и редактировать конфигурации с любого устройства.

2.2. Нефункциональные требования

Программа должна соответствовать следующим нефункциональным требования:

1) производительность: Веб-сервис должен быть быстрым и отзывчивым, чтобы пользователи могли легко создавать и редактировать конфигурации ПК;

2) безопасность: Должны быть приняты меры безопасности для защиты личной информации пользователей и обеспечения целостности данных;

3) доступность: Сервис должен быть доступен для использования в любое время суток, с минимальным временем простоя для обслуживания;

4) надежность: Сервис должен быть надежным и стабильным, чтобы минимизировать возможность сбоев и простоев;

5) удобный интерфейс: Интерфейс веб-сервиса должен быть интуитивно понятным и привлекательным для пользователей всех уровней навыков.

2.3. Варианты использования

Для создания наглядной и понятной демонстрации функциональных требований веб-приложения используется диаграмма вариантов использования. Это инструмент позволяющий систематизировать взаимодействие актеров – пользователей и администраторов – с функциями приложения.

Актеры в данном контексте представлены двумя основными ролями: администраторами и пользователями. Администраторы являются сотрудниками магазина, ответственными за управление товарами и обслуживание системы. Их задачи включают внедрение новых товаров, удаление устаревших позиций, редактирование ассортимента в конфигураторе и обработку заявок от пользователей. С другой стороны, пользователи - клиенты магазина – имеют возможность пользоваться функционалом приложения для подбора комплектующих в конфигураторе или создания готовых конфигураций ПК из предложенного ассортимента.

Диаграмма вариантов использования позволяет систематизировать все возможные сценарии взаимодействия между актерами и системой. Каждый вариант использования определяет конкретный сценарий работы, начиная от инициации действия актера до достижения цели. Например, вариант использования "Внедрение нового товара" для администратора магазина включает шаги по добавлению нового продукта в систему, его описание, ассоциирование с категориями и установку цен.

Кроме того, каждый вариант использования включает алгоритм действий, который подробно описывает последовательность шагов для достижения поставленной цели. Например, вариант использования "Создание конфигурации ПК" для пользователя может включать выбор основных компонентов (процессор, видеокарта, оперативная память и т.д.), их согласование по совместимости и генерацию окончательного списка для заказа.

Таким образом, диаграмма вариантов использования является эффективным инструментом для визуализации и документирования функциональных требований веб-приложения, позволяя легко увидеть взаимодействие между актерами и системой на разных этапах их взаимодействия.

Рисунок 4 – Диаграмма вариантов использования

Администратор – сотрудник магазина, который внедряет новый товар под реализацию, удаляет потерявший актуальность товар, редактирует ассортимент комплектующих в конфигураторе и отвечает на заявки пользователей.

Пользователь – клиент магазина, который имеет возможность подобрать нужные себе комплектующие в конфигураторе или собрать готовый ПК из предложенного ассортимента.

Описание диаграммы вариантов использования:

1. Создать аккаунт – возможность для пользователя, в случае отсутствия учетной записи, необходимо ввести свои данные для регистрации (Имя пользователя, пароль).

2. Авторизация – возможность для пользователя ввести свои данные для входа, чтобы просматривать и пользоваться полным функционалом сайта.

3. Добавление комплектующих в корзину – возможность пользователя добавлять в корзину, интересующие его комплектующие.

4. Удаление комплектующих из корзины - возможность пользователя удалять комплектующие из корзины.

5. Указать бюджет – возможность пользователя указать бюджет, на которую будет подбирать комплектующие.

6. Удаление комплектующих из БД – возможность администратора удалять комплектующие из БД.

7. Добавление комплектующих в БД – возможность администратора добавлять комплектующие в БД.

8. Удаление пользователей – возможность администратора удалять пользователей из БД.

9. Редактирование данных в БД комплектующих – возможность администратора редактировать данные в БД комплектующих.

2.4. Проектирование базы данных

На рисунках 5,6,7,8 представлена структура базы данных веб-сервиса. На ней изображены комплектующие которые содержат необходимые поля для разработки и выстраивание зависимости между друг другом. А также несколько дополнительных таблиц, которые нужны для проверки совместимости комплектующих.

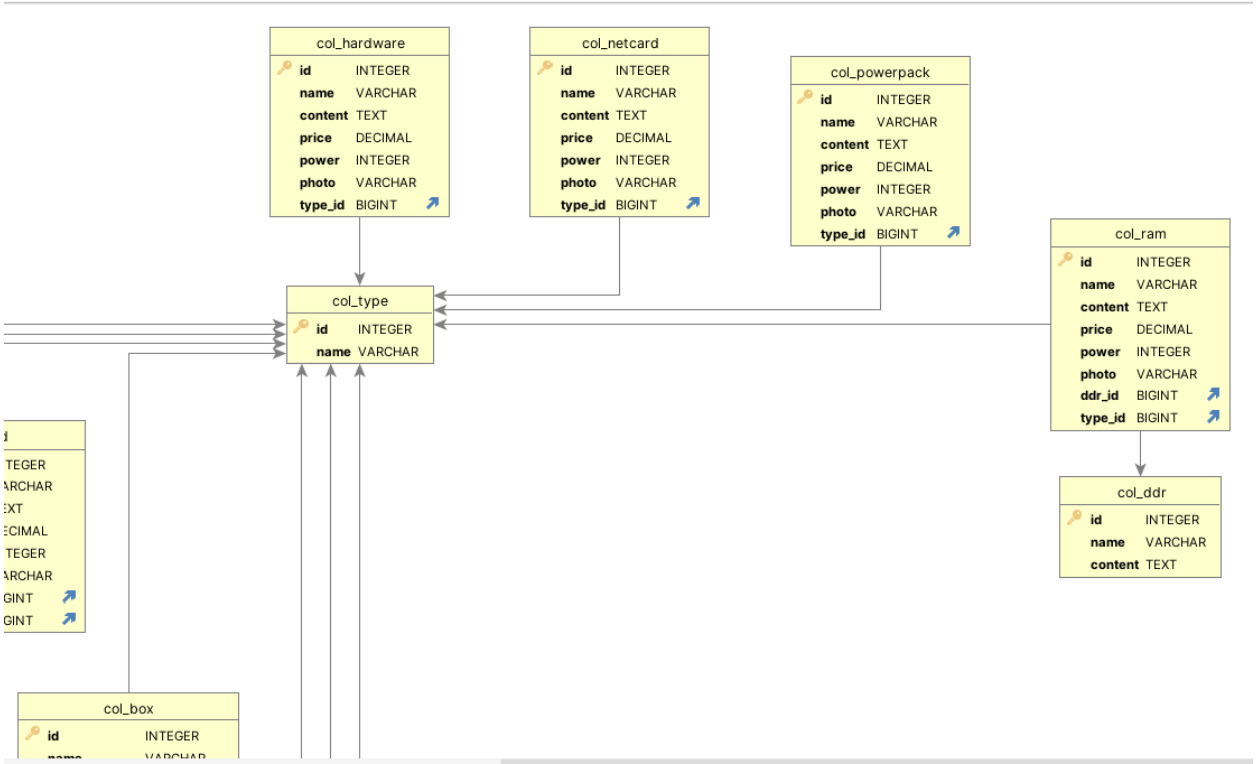

Рисунок 5 – Структура базы данных

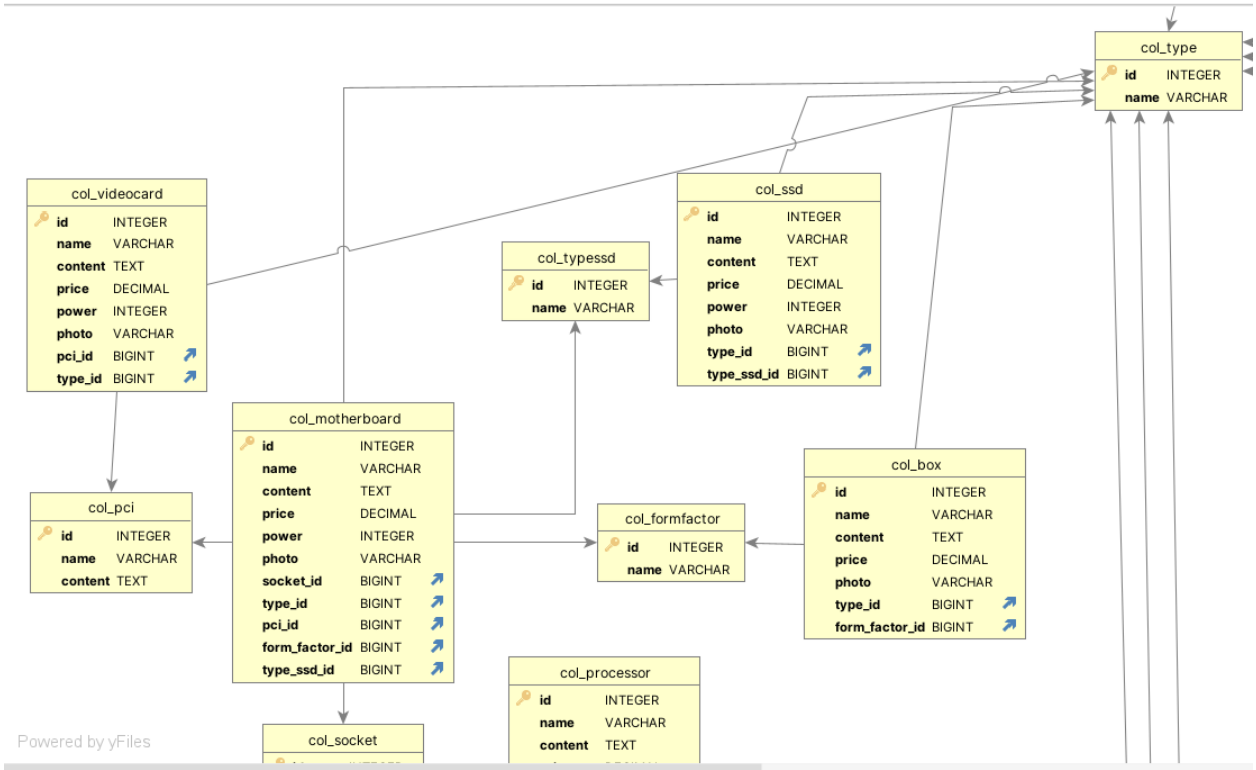

Рисунок 6 – Структура базы данных

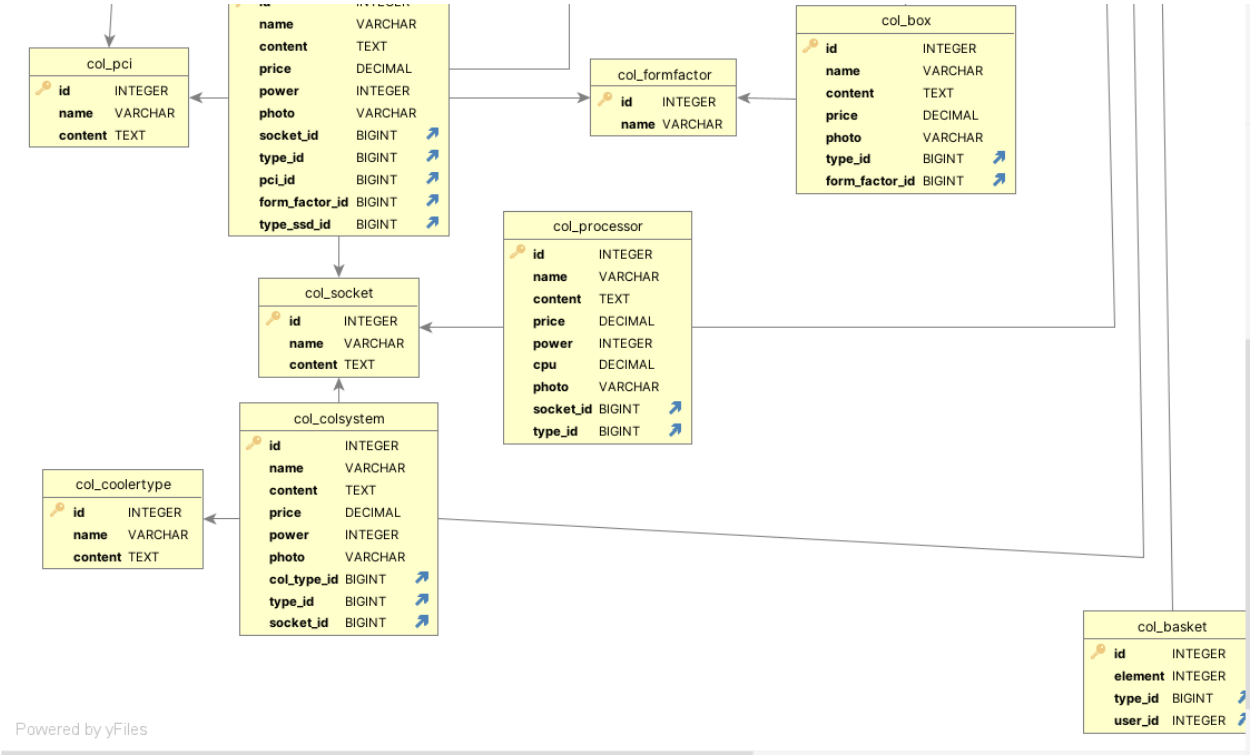

Рисунок 7 – Структура базы данных

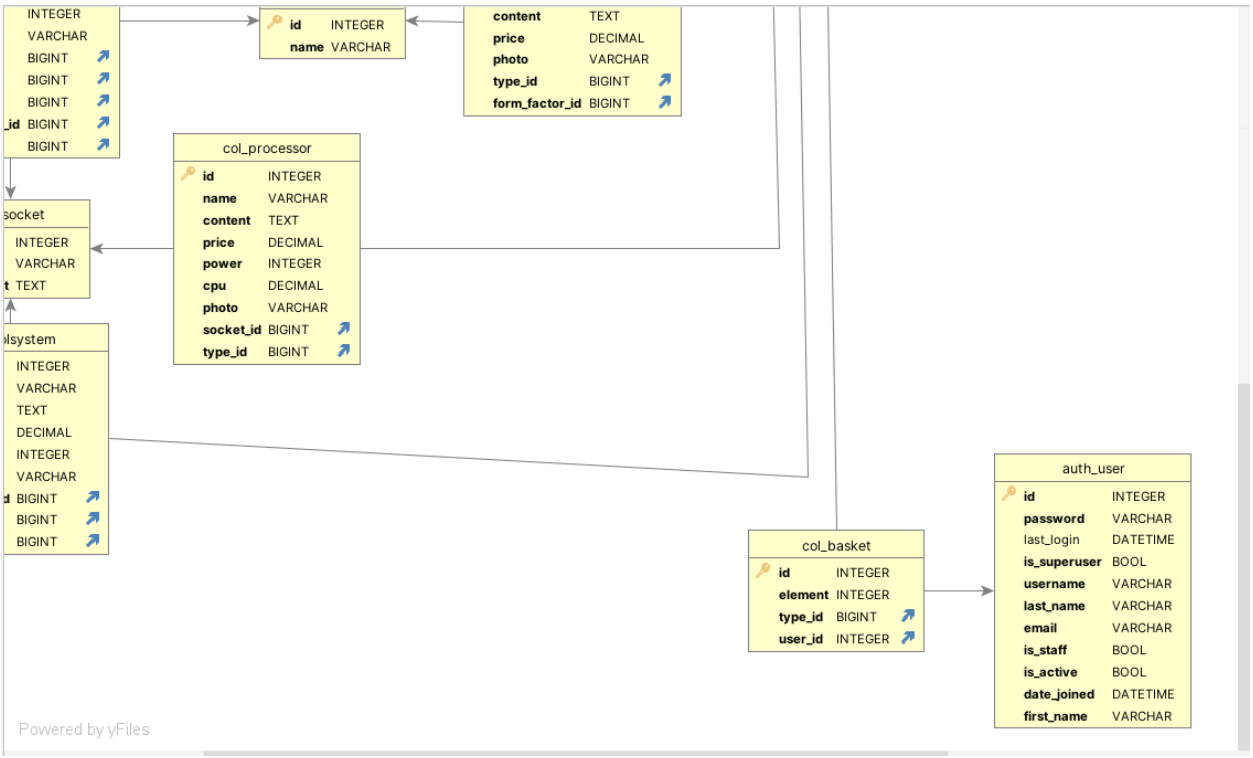

Рисунок 8 – Структура базы данных

Необходимость базы данных при создании любого веб-приложения обуславливается хранением всех необходимых данных в одном месте, для

дальнейшей обработки и хранения информации, доступ к которой можно получить, сохранив ее ранее.

Таблица 2 – Описание базы данных

Название	Описание
auth_user	Информация о пользователях
Col_basket	Информация о корзинах
Col_type	Информация о типе комплектующего
Col_ddr	Информация о DDR
Col_hardware	Информация о жестких дисках
Col_netcard	Информация о сетевых картах
Col_processor	Информация о процессорах
Col_ram	Информация об оперативной памяти
Col_powerpack	Информация о блоках питания
Col_colsystem	Информация о системах охлаждения
Col_coolertype	Информация о типе системы охлаждения
Col_ssd	Информация о твердотельных накопителях
Col_typessd	Информация о типах твердотельных накопителей
Col_videocard	Информация о видеокартах
Col_box	Информация о корпусах для ПК
Col_motherboard	Информация о материнских платах
Col_socket	Информация о сокетах
Col_pci	Информация об PCIe
Col_formfactor	Информация об форм-факторах

В таблице 2 указана информация с описанием той или иной таблицы, содержащейся в базе данных, в соответствии с тем, за что каждая из них отвечает. Такая таблица как socket отвечает за соответствие такому важному параметру, как сокет. Данный параметр фигурирует в таблицах processor, motherboard и colsystem. Его реализация необходима в первую очередь для

неопытных пользователей, а именно для защиты от выбора неподходящих друг другу материнских плат, процессоров и систем охлаждения процессоров, соответственно.

Вывод по данной главе

В данной главе были проанализированы и выдвинуты требования к системе, реализована диаграмма вариантов использования, отображена и описана схема базы данных.

3. РЕАЛИЗАЦИЯ

3.1. Архитектура приложения

Приложения написанные с помощью фреймворка Django обычно следуют принципам Model–View–Controller или его вариации Model-Template–View, которая является более характерной для Django. Вот основные компоненты стандартной архитектуры веб-приложения Django:

1) модели (Models): Модели представляют собой структуру данных и бизнес-логику вашего приложения. Они определяют структуру таблиц в базе данных и взаимосвязи между ними;

2) шаблоны (Templates): Шаблоны представляют HTML-файлы с вставками переменных и блоков шаблонного кода Django (шаблонный движок). Они отображают информацию пользователю в виде веб-страниц;

3) представления (Views): Представления обрабатывают запросы от клиентов и формируют ответы. Они взаимодействуют с моделями данных для получения необходимых данных и передают их в шаблоны для отображения;

4) URL-адреса (URLs): URL-адреса определяют, какие представления должны быть вызваны при обращении к определенным URL-адресам. Они маршрутизируют запросы от клиентов к соответствующим представлениям;

5) настройки проекта (Settings): Файл настроек проекта содержит все конфигурационные параметры вашего приложения, такие как настройки базы данных, маршрутизации URL-адресов, настройки безопасности и другие;

6) административный интерфейс (Admin): Django поставляется с встроенным административным интерфейсом, который автоматически

создается на основе ваших моделей данных. Он позволяет администраторам управлять данными приложения через веб-интерфейс.

Выше перечислены основные компоненты архитектуры веб-приложения Django. Они взаимодействуют друг с другом для обеспечения функциональности, безопасности и эффективной работы веб-сервиса.

3.2. Реализация компонентов

Первым шагом в реализации проекта является создание основы с помощью Django. Для этого была использована команда *django-admin startproject*, которая создает базовую структуру проекта. Полученный каталог проекта содержит основные файлы управления и настройки Django, такие как manage.py, settings.py, urls.py, и другие.

Далее необходимо реализовать на главной странице аутентификацию пользователей и администраторов. На листинге 1 изображен листинг кода аутентификации, где в зависимости от процесса аутентификации идет переключение на страницу пользователя или администратора рис 9.

Листинг 1 – Код аутентификации между пользователем и администратором

```
#функиця для аутентификации
def auth(request):
#проверка данных пользователя с данными из бд
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('collector')
        else:
            return HttpResponse('Ошибка: неверное имя пользователя или 
пароль.', status=401)
    else:
        return HttpResponse('Для начала зарегестриуйся!.', status=405)
```

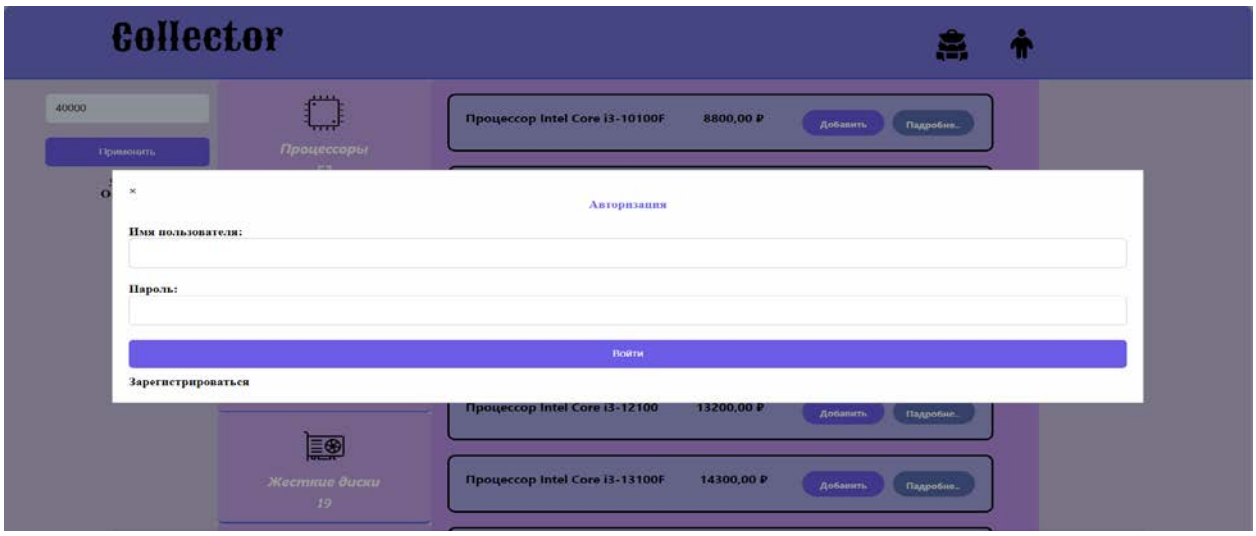

Рисунок 9 – Окно аутентификации пользователя

Если пользователь отсутствует в базе данных пользователей, то он может пройти процедуру регистрации. В листинге 2 изображен листинг кода регистрации.

Листинг 2 – Код регистрации

```
#функиця для регистрации пользователя
def register(request):
#вызов формы и регистрация
if request.method == 'POST':
#вызов формы
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
        form = UserCreationForm()
    return redirect('collector')
```

На рисунке 10 изображено окно для регистрации пользователей.

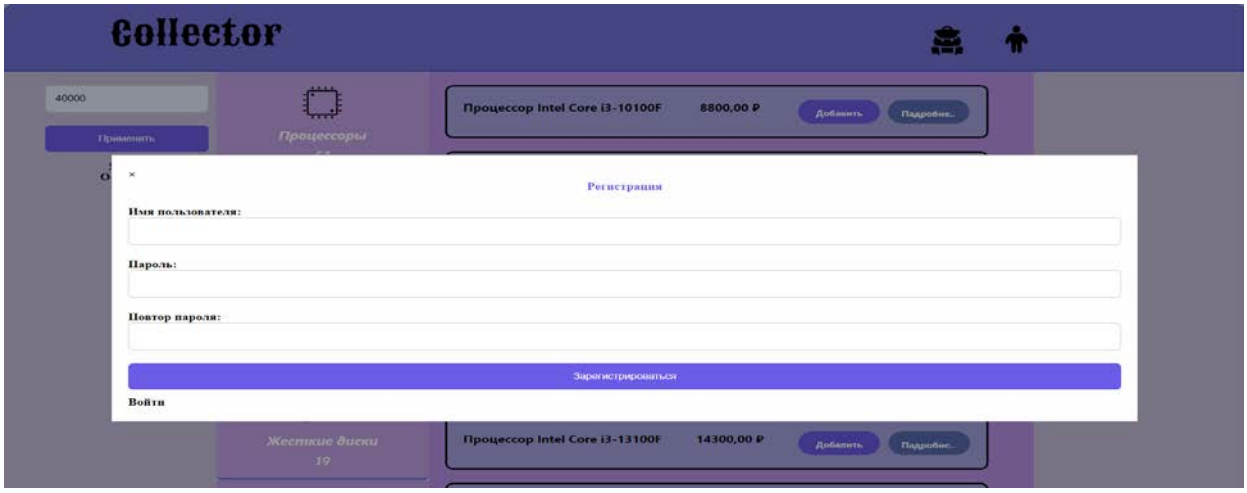

Рисунок 10 – Окно регистрации пользователей

После процедуры аутентификации необходимо перевести пользователя, либо администратора на соответствующие им личные кабинеты. На листинге 3 изображен фрагмент кода реализации главной страницы конфигуратора рисунок 11.

Листинг 3 – Код реализации главной страницы конфигуратора ПК

```
#функиця для отображения главного экрана
def index(request):
    current_basket = get_basket_items(request)
#Вызов функции get_data_to_load_view
    data = get_data_to_load_view(request)
#Сумма всех товаров в корзине
    basket_sum = sum(item.price for item in current_basket)
    if request.GET and 'is_description_response' in request.GET:
        data = {}
        item = basket_to_modal_objects(request.GET['id'],
request.GET['typeId'])
        print(item)
        data['desc'] = item.content
        data['name'] = item.name
        data['photo'] = item.photo.url
        data['price'] = item.price
        return JsonResponse({"data_to_description": data})
    if request.GET and 'price' in request.GET and
request.GET['price'].isdigit():
        price = request.GET['price']
    else:
        price = request.COOKIES.get('price')

    isinpr = None
    register_form = UserCreationCustomForm
    login_form = LoginUserForm
#сравнение текущей суммы товаров из корзины и бюджета
    if price and decimal.Decimal(price) < basket_sum:
        isinpr = True
    group_names = ['Процессоры', 'Материнские платы', 'Видеокарты', 'Жесткие
диски', 'Твердотельные накопители',
                   'Оперативная память', 'Сетевые карты', 'Корпуса',
'Охлаждение', 'Блоки питания']
    context = {'data': data, 'currentBasket': current_basket,
'register_form': register_form, 'login_form': login_form,
               'price': price, 'group_names': group_names, 'basket_sum':
basket_sum, 'is_incorrect_price': isinpr,
               'rest_money': get_rest_money(request)}
    return render(request, 'collector/index.html',
                  context=context)
```

Для отображения всех добавленных пользователем компонентов, а также для обработки запросов на их удаление и добавление.

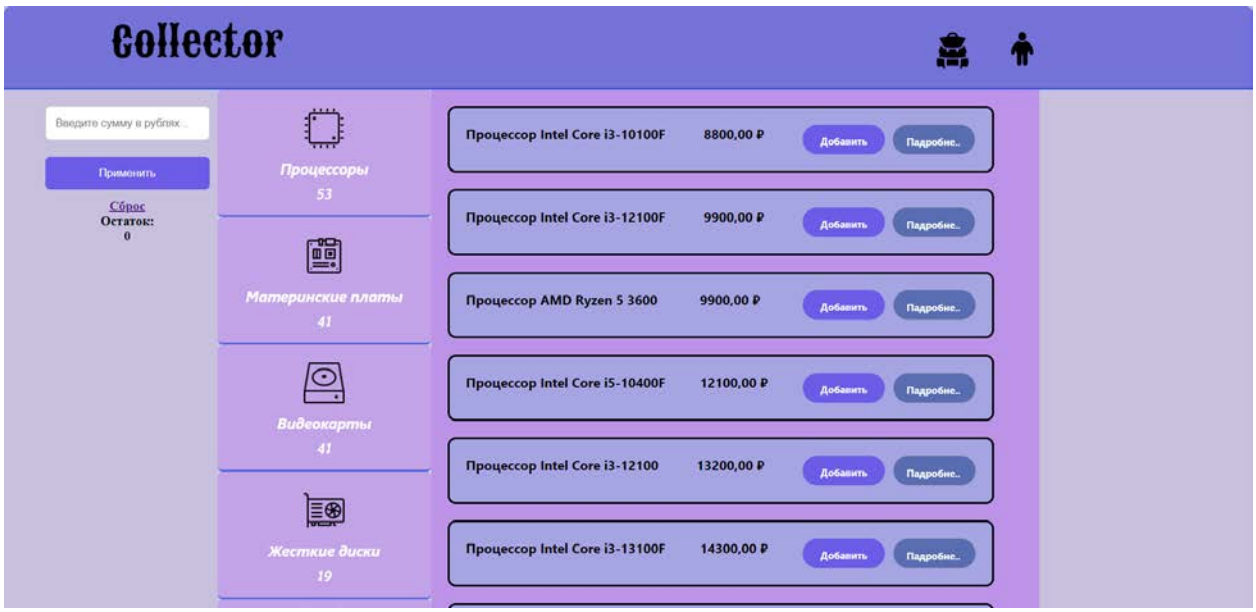

Рисунок 11 – Главная страница сайта

На листинге 4 изображен фрагмент кода для отображения страницы корзины рисунок 12. Функция проверяет наличие данных в БД и выводит их на экран.

Листинг 4 – Код корзины

```
#функиця для отображения корзины
def basket(request):
    if request.POST:
        type = Type.objects.get(pk=request.headers['typeId'])
        if request.user.is_authenticated:
            user = User.objects.get(pk=request.user.id)
            if request.headers['switchKey'] == '1':
                same_items =
Basket.objects.filter(type=type).filter(user=user)
                if same_items is not None:
                    same_items.delete()
                Basket.objects.create(user=user, type=type,
element=request.headers['id'])
            else:
                basket_el = Basket.objects.filter(user=user, type=type,
element=request.headers['id'])
                basket_el.delete()
        return JsonResponse({"data_to_reload_view":
get_data_to_reload_view(request)})
    context = {'basket_list': get_basket_items(request)}
    return render(request, 'collector/basket.html', context=context)
```

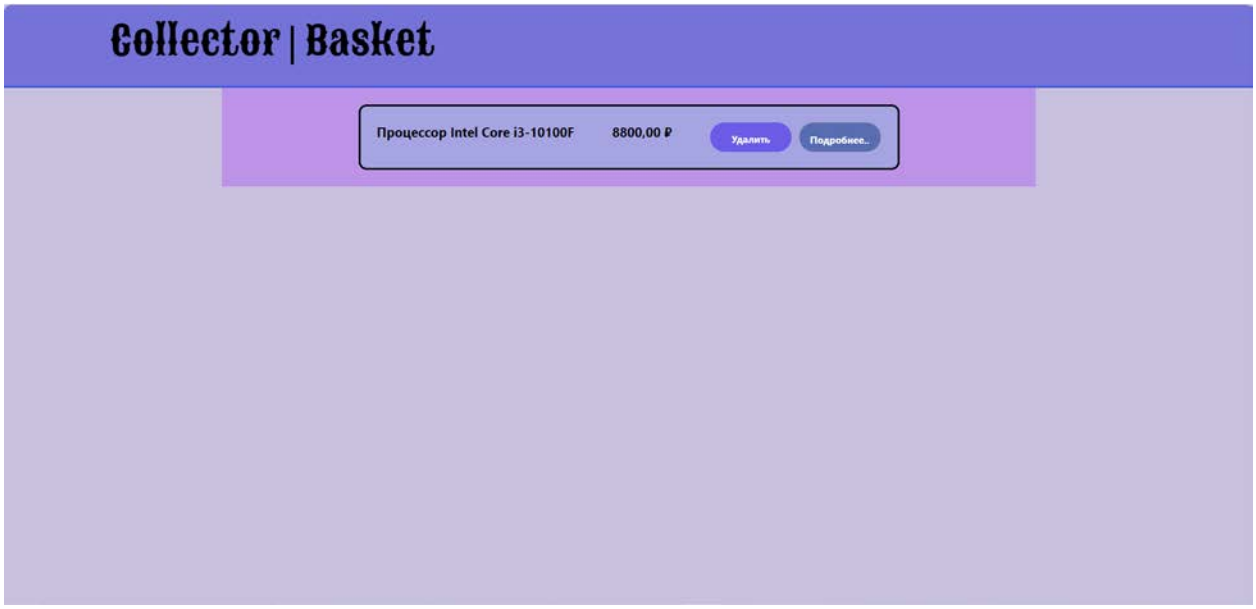

Рисунок 12 – Страница Корзины

У материнской платы *motherboard*, как и у других комплектующих есть также свои внутренние характеристики. На листинге 5 изображен код с реализацией параметров материнской платы. Поле name отвечает за наименование материнской платы, content – ее описание, price – цена, power хранит в себе информацию о потребляемой мощности, socket – информация о сокете материнской платы, photo – без комментариев, type – тип комплектующего, pci – информация об pci материнской платы, form_factor – хранит информацию о форм-факторе платы, type_ssd – хранит в себе информацию о типах твердотельных накопителей, которые она поддерживает.

Листинг 5 – Код материнской платы

```
#модель материнской платы
class Motherboard(models.Model):
    name = models.CharField(max_length=50)
    content = models.TextField()
    price = models.DecimalField(max_digits=12, decimal_places=2)
    power = models.IntegerField()
    socket = models.ForeignKey(Socket, on_delete=models.PROTECT)
    photo = models.ImageField(upload_to="photo/%V/")
    type = models.ForeignKey(Type, on_delete=models.PROTECT)
    pci = models.ForeignKey(PCI, on_delete=models.PROTECT, default=1)
    form_factor = models.ForeignKey(FormFactor, on_delete=models.PROTECT,
default=1)
    type_ssd = models.ForeignKey(TypeSSD, on_delete=models.PROTECT,
default=1)
```

Функция get_data_to_load_view для защиты пользователя от выбора неподходящих друг другу материнских плат, процессоров, систем охлаждения и др., а также для подготовки данных на сторону клиента. На листинге 6 изображен код реализации этой функции.

Листинг 6 – Код реализации сбора и фильтрации данных

```
#функиця для сбора и фильтрации данных
def get_data_to_load_view(request):
    result = get_products_by_price(request)
    current_basket = get_basket_items(request)
    power_sum = 100
#проверка совместимости комплектующих
    for element in current_basket:
        if isinstance(element, Motherboard):
            result['processors'] =
result['processors'].filter(socket=element.socket)
            result['boxes'] =
result['boxes'].filter(form_factor=element.form_factor)
            result['ssd'] = result['ssd'].filter(type_ssd=element.type_ssd)
            result['videocards'] =
result['videocards'].filter(pci=element.pci)
        if isinstance(element, Processor):
            result['motherboards'] =
result['motherboards'].filter(socket=element.socket)
            power_sum += element.power
        if isinstance(element, Box):
            result['motherboards'] =
result['motherboards'].filter(form_factor=element.form_factor)
        if isinstance(element, SSD):
            result['motherboards'] =
result['motherboards'].filter(type_ssd=element.type_ssd)
        if isinstance(element, VideoCard):
            result['motherboards'] =
result['motherboards'].filter(pci=element.pci)
            power_sum += element.power
    power_sum *= 1.2
    result['powerpacks'] = result['powerpacks'].filter(power__gte=power_sum *
1.2)
    return result
```

Для получения всех комплектующих из корзины. Для авторизированных пользователей сбор данных происходит путем обращения к базе данных, а для неавторизированных пользователей данные берутся из Cookies браузера. На листинге 7 реализован этот метод.

Листинг 7 – Код сбора комплектующих из текущей корзины

```
#функиця для сбора данных из корзины
def get_basket_items(request):
    basket_els = []
#проверка на авторизацию пользователя
    if request.user.is_authenticated:
        baskets = Basket.objects.filter(user=request.user.id)
        for el in baskets:
            basket_els.append(basket_to_modal_objects(el.element,
el.type_id))
    else:
        types = Type.objects.all()
        for type in types:
            group_id = str(type.pk)
            if group_id in request.COOKIES and request.COOKIES.get(group_id)
!= '':
                group_id = request.COOKIES.get(group_id)
                basket_els.append(basket_to_modal_objects(group_id, type.pk))

    return basket_els
```

Функция get_products_by_price для отображения комплектующих, которые удовлетворяют бюджету пользователя изображена на листинге 8.

Листинг 8 – Код реализации отбора комплектующих по бюджету

```
#функиця для подбора комплектующих
def get_products_by_price(request):
#словарь имя/данные
    result = {'processors': Processor.objects.all(),
              'motherboards': Motherboard.objects.all(),
              'videocards': VideoCard.objects.all(),
              'hardwares': Hardware.objects.all(),
              'ssd': SSD.objects.all(),
              'ram': RAM.objects.all(),
              'netcards': NetCard.objects.all(),
              'boxes': Box.objects.all(),
              'colsystems': ColSystem.objects.all(),
              'powerpacks': PowerPack.objects.all()}
    basket = get_basket_items(request)
    basket_sum = sum(item.price for item in basket)
    if request.GET and 'price' in request.GET and
request.GET['price'].isdigit() or request.COOKIES.get('price'):
        if request.GET and 'price' in request.GET and
request.GET['price'].isdigit():
            price = decimal.Decimal(request.GET['price']) - basket_sum
        else:
            price = decimal.Decimal(request.COOKIES.get('price')) -
basket_sum
        for key in result.keys():
            if result[key].first().type.pk in list(int(i.type.pk) for i in
basket):
                index = list(int(i.type.pk) for i in
basket).index(result[key].first().type.pk)
                result[key] = result[key].filter(price__lte=price +
basket[index].price)
            else:
                result[key] = result[key].filter(price__lte=price)
    return result
```

На рисунке 13 изображен результат изменения количества комплектующих подходящих под бюджет пользователя.

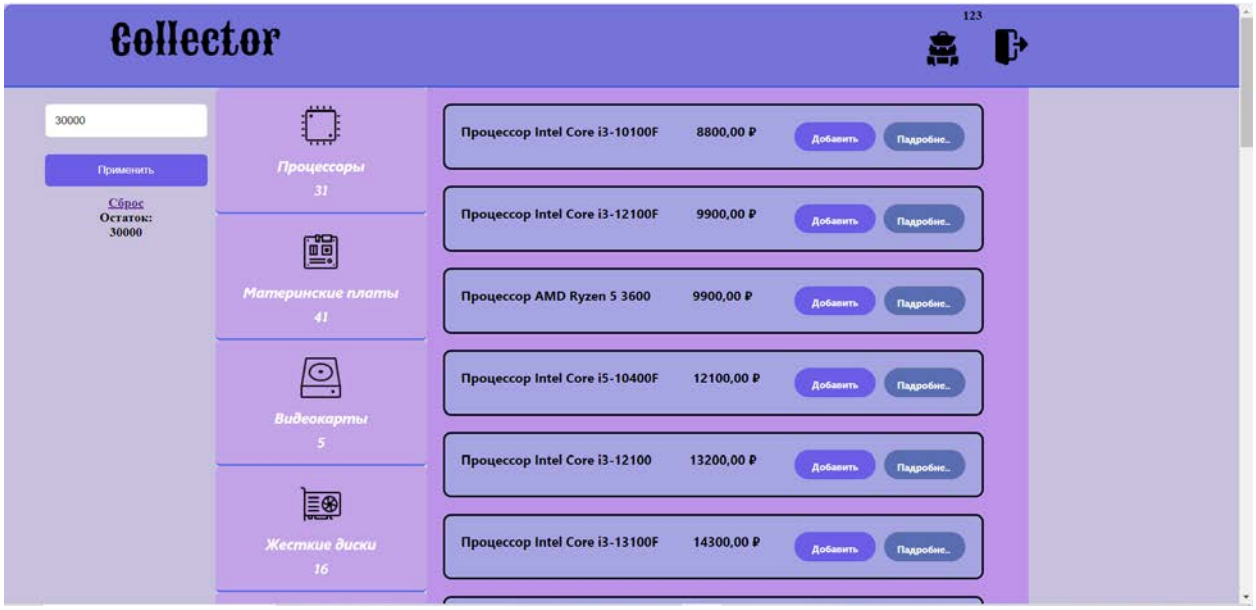

Рисунок 13 – Результат работы отбора по указанному бюджету

На листинге 9 изображена функция для подсчета остатка бюджета пользователя.

Листинг 9 – Код функции для подсчета остатка бюджета пользователя

```
#функциця для подсчета остатка
def get_rest_money(request):
    rest_money = 0
    current_basket = get_basket_items(request)
    basket_sum = sum(item.price for item in current_basket)
    if request.GET and 'price' in request.GET and
request.GET['price'].isdigit():
        price = request.GET['price']
        rest_money = decimal.Decimal(price) - basket_sum
    else:
        price = request.COOKIES.get('price')
        if price:
            rest_money = decimal.Decimal(price) - basket_sum
    return rest_money
```

Листинг 10 – Код для отображения окна с подробной информацией о товаре.

```
#функциця для отображения окна с информацией о товаре
<div id="description_modal" class="desc_modal">
    <div class="descr_modal_content">
        <img width="200px" height="200px" id='content_img' src="">
        <div class="desc-cont">
            <div class="desc_name" id="desc_name"></div>
            <div class="desc_desc" id="desc_desc"></div>
            <div class="desc_price">
                <div>Цена:</div>
                <div id="desc_price"></div>
```

```
        </div>
        </div>

         <span class="close_descr">&times;</span>
    </div>
</div>
```

На рисунке 14 изображено окно с подробной информацией.

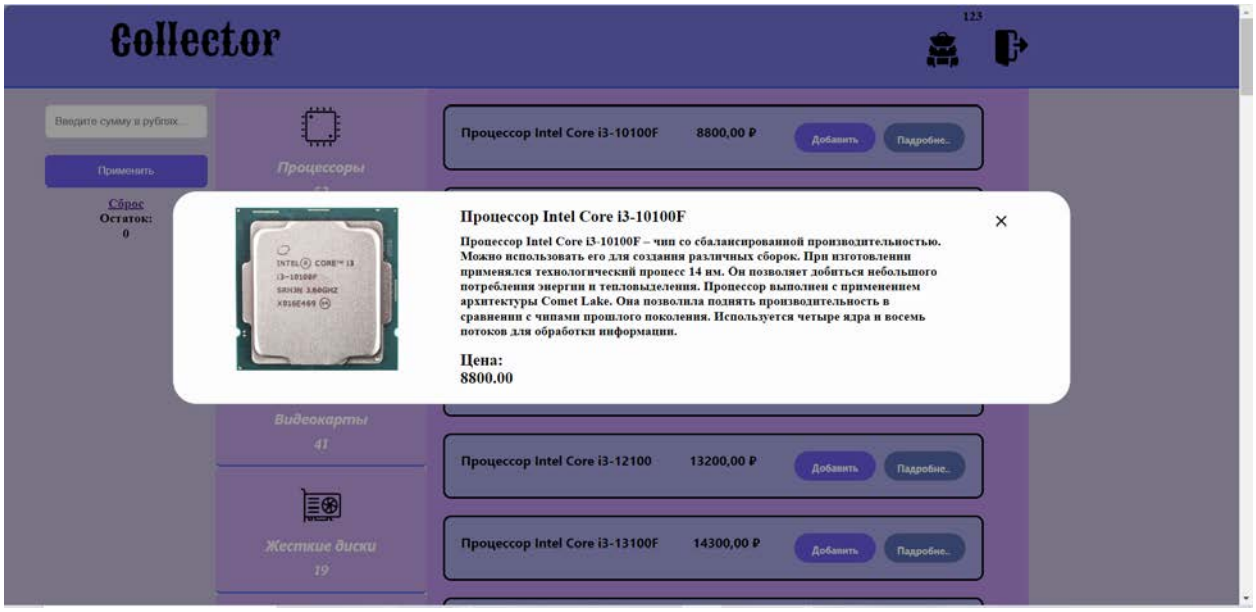

Рисунок 14 – Окно с подробной информацией о товаре

Вывод по данной главе

В ходе данной главы были освещены ключевые аспекты разработки, начиная с создания базовой структуры проекта с помощью Django и реализации основных функций, таких как аутентификация пользователей, регистрация, отображение и управление компонентами ПК.

4. Тестирование веб-сервиса

4.1 Функциональное тестирование

Функциональное тестирование программного обеспечения направлено на проверку того, способна ли система выполнять необходимые функциональные требования.

Используя методологию функционального тестирования, проверим работу веб-сервиса.

Тест №1. Корректное отображение главной страницы и всех комплектующих.

Цель: проверить корректность главной страницы и всех комплектующих.

В боковой части экрана должны отображаться все виды комплектующих, которые получаются с сервера. В центре экрана должен располагаться список товаров, выбранного типа, который также получается с сервера.

Результат: тест пройден.

Тест №2. Корректная работа фильтра товаров по бюджету.

Входные данные: веб-сервис открыт на экране общего каталога и бюджет.

Цель: проверить корректность работы фильтра.

При указании бюджета пользователя, фильтр должен корректно выполнять выборку товаров по заданному значению .

Ход проведения:

1) в поле указать нужный бюджет;

2) нажать на кнопку **Применить;**

3) выбрать несколько товаров.

Результат: тест пройден, ход тестирования и результат изображены на рисунке 1,2,3.

Тест №3. Регистрация пользователя.

Входные данные: веб-сервис открыт на экране регистрации пользователя, данные пользователя.

Цель: проверить работу регистрации новых пользователей.

Ход проведения:

1) открыть окно регистрации пользователей;

2) ввести данные пользователя;

3) нажать на кнопку **Зарегистрироваться.**

Результат: тест пройден.

Тест №4. Аутентификация пользователя.

Входные данные: веб-сервис открыт на экране аутентификации пользователя, данные пользователя.

Цель: проверить работоспособность аутентификации пользователя.

Ход проведения:

1) открыть окно аутентификации пользователя;

2) заполнить необходимые поля;

3) нажать на кнопку **Войти.**

Результат: тест пройден.

Тест №5. Корректность работы добавления товара в корзину.

Входные данные: веб-сервис открыт на главном экране.

Цель: проверить работоспособность добавления комплектующих.

Ход проведения:

1) открыть главный экран;

2) в списке типов комплектующих выбрать нужный;

3) в списке товаров по выбранному типу, выбрать товар;

4) нажать кнопку **Добавить**.

Результат: Тест пройден, ход проведения и результат тестирования изображен в приложении на рисунке 4,5,6.

Тест №6. Корректность работы удаления товара из корзины.

Входные данные: веб-сервис открыт на главном экране, добавлен хотя бы один элемент.

Цель: проверить работоспособность удаления комплектующих.

Ход проведения:

1) открыть главный экран;

2) в списке типов комплектующих выбрать нужный;

3) в списке товаров по выбранному типу, найти добавленный товар;

4) нажать кнопку **Удалить.**

Результат: Тест пройден, ход проведения и результат тестирования изображен в приложении на рисунке 7,8.

Тест №7. Корректность работы кнопки **Подробнее**.

Входные данные: веб-сервис открыт на главном экране.

Цель: проверить работоспособность кнопки **Подробнее**.

Ход проведения:

1) открыть главный экран;

2) в списке типов комплектующих выбрать нужный;

3) в списке товаров по выбранному типу, выбрать товар;

4) нажать кнопку **Подробнее**.

Результат: Тест пройден, ход проведения и результат тестирования изображен в приложении на рисунке 9,10.

ЗАКЛЮЧЕНИЕ

Конфигураторы ПК являются важным инструментом для современных потребителей, позволяя создавать индивидуальные компьютерные системы, которые полностью подходят для потребителя. В ходе данного исследования был разработан и реализован конфигуратор ПК, основанный на технологиях веб-разработки и интеграции с базами данных. Основная его цель заключалась в предоставлении пользователям возможности выбирать комплектующие, адаптированные к их специфическим требованиям и бюджету, для создания персонализированных конфигурации ПК. Для достижения данной цели были решены следующие задачи:

1) произведена постановка задачи;

2) произведен обзор аналогов;

3) спроектирован веб-сервис;

4) реализован веб-сервис;

5) протестирован веб-сервис.

Все поставленные задачи были решены. Разработанный веб-сервис имеет перспективы дальнейшего развития.

В перспективе планируется реализовать следующие возможности:

1) добавление возможности заказывать комплектующие;

2) добавление возможности оставлять отзывы;

3) добавление статистики по продажам;

4) добавление статистики кадров в зависимости от выбора комплектующих.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Официальный сайт DNS. [Электронный ресурс]. URL: https://www.dns-shop.ru/configurator/ (дата обращения: 14.04.2024 г.).

2. Официальный сайт IronBook. [Электронный ресурс]. URL: https://www.ironbook.ru/ (дата обращения: 14.04.2024 г.).

3. Официальный сайт RoyalPC. [Электронный ресурс]. URL: https://royal-computers.ru/configurator (дата обращения: 14.04.2024 г.).

4. Metanit.com. | Сайт о программировании [Электронный ресурс] URL:http://metanit.com/. (дата обращения 28.02.2024 г.);

5. Django.fun | Официальная документация [Электронный ресурс] URL:https://django.fun/docs/django/5.0/ . (дата обращения 20.03.2024 г.);

6. React документация. [Электронный ресурс]. URL: https://reactjs.org/docs/getting-started.html (дата обращения: 14.04.2024 г.).

7. TypeScript документация. [Электронный ресурс]. URL: https://www.typescriptlang.org/docs/ (дата обращения: 14.04.2024 г.).

8. Webpack документация. [Электронный ресурс]. URL: https://webpack.js.org/concepts/ (дата обращения: 14.04.2024 г.).

9. Redux документация. [Электронный ресурс]. URL: https://redux.js.org/introduction/getting-started (дата обращения: 14.04.2024 г.).

10. JavaScript документация. [Электронный ресурс]. URL: https://learn.javascript.ru/ (дата обращения: 14.04.2024 г.).

11. HTML документация. [Электронный ресурс]. URL: https://hcdev-ru.pages.dev/html/ (дата обращения: 14.04.2024 г.).

12. CSS документация. [Электронный ресурс]. URL: https://webref.ru/css (дата обращения: 14.04.2024 г.).

13. Git документация. [Электронный ресурс]. URL: https://git-scm.com/doc (дата обращения: 14.04.2024 г.).

14. RTK query документация. [Электронный ресурс]. URL: https://redux.js.org/introduction/getting-started (дата обращения: 14.04.2024 г.).

15. FSD архитектура документация. [Электронный ресурс]. URL: https://feature-sliced.design/docs/ (дата обращения: 14.04.2024 г.).

16. Jest документация. [Электронный ресурс]. URL: https://jestjs.io/docs/getting-started (дата обращения: 14.04.2024 г.).

17. Storybook документация. [Электронный ресурс]. URL: https://storybook.js.org/docs/get-started (дата обращения: 14.04.2024 г.).

18. Figma документация. [Электронный ресурс]. URL: https://figma.com/ (дата обращения: 14.04.2024 г.).

ПРИЛОЖЕНИЕ

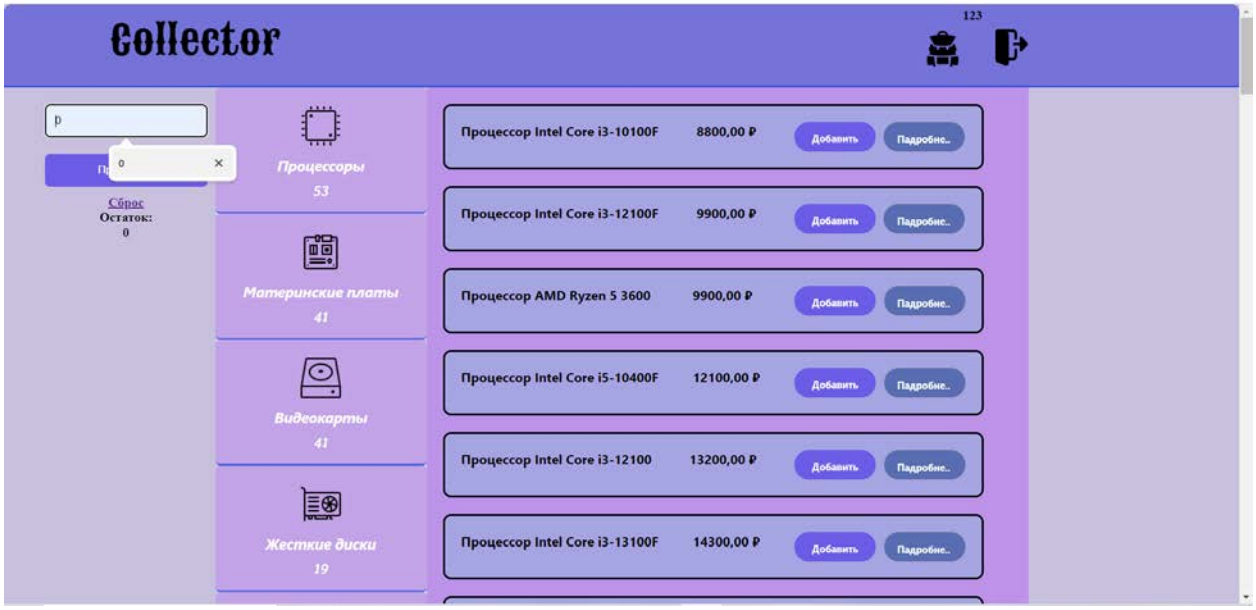

Рисунок 1 – Текстовое поле для бюджета пользователя

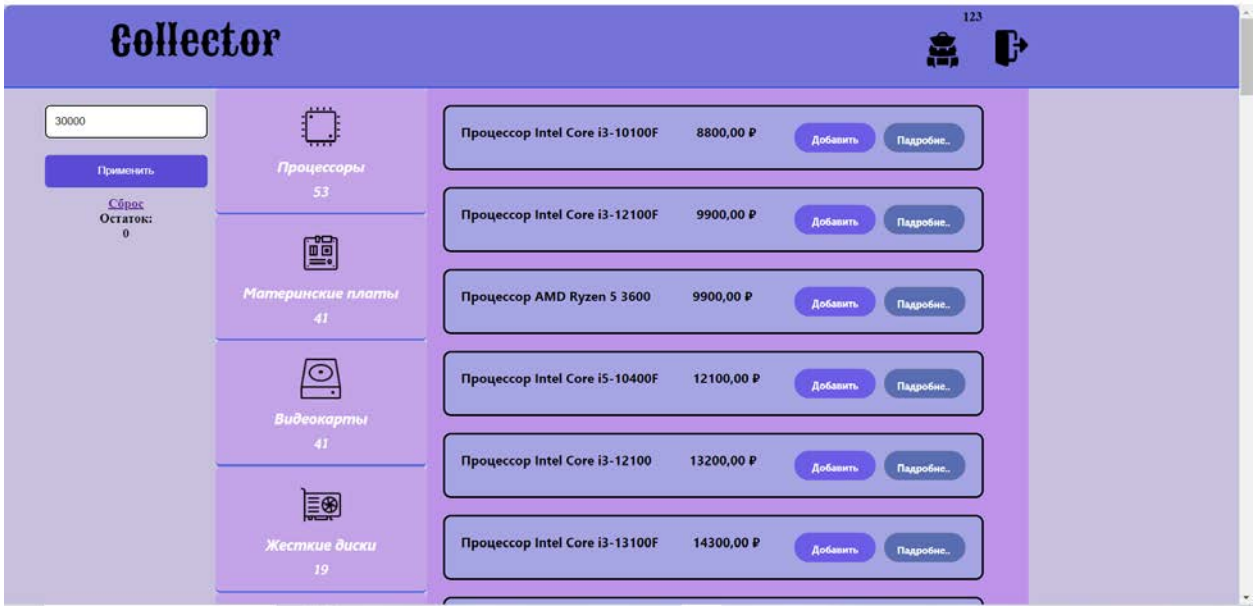

Рисунок 2 – Ввести бюджет

Рисунок 3 – Результат отбора по бюджету

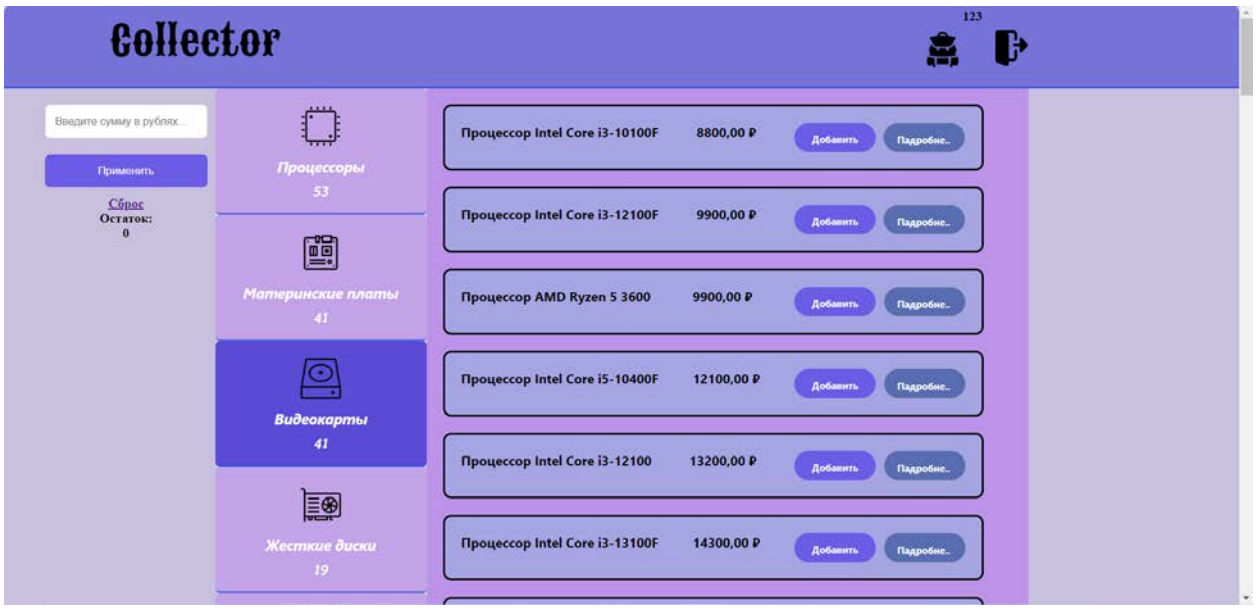

Рисунок 4 – Выбрать категорию комплектующего

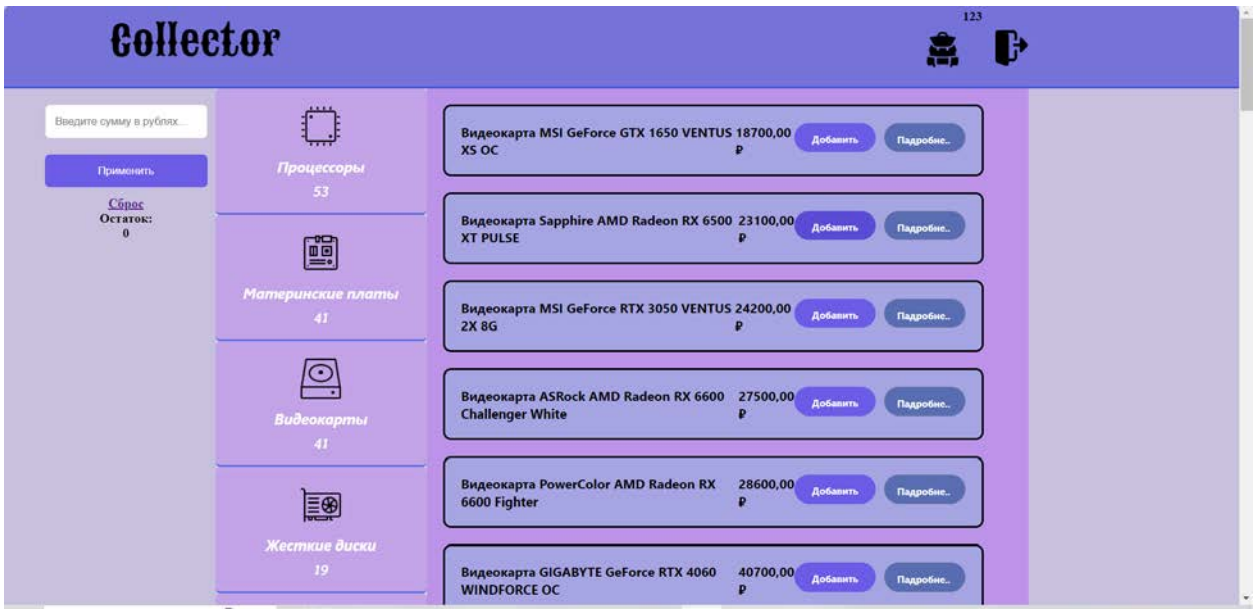

Рисунок 5 – Выбрать товар и нажать кнопку **Добавить**

Рисунок 6 – Результат тестирования

Рисунок 7 – Нажата кнопка **Удалить**

Рисунок 8 – Результат тестирования кнопки **Удалить**

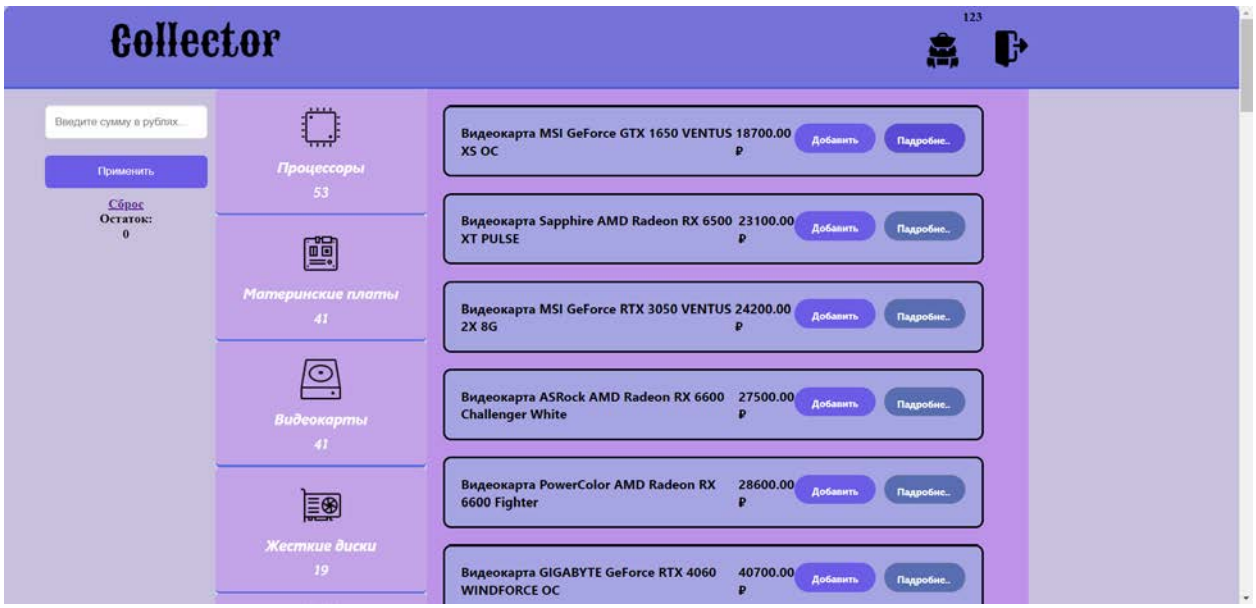

Рисунок 9 – Нажата кнопка **Подробнее**

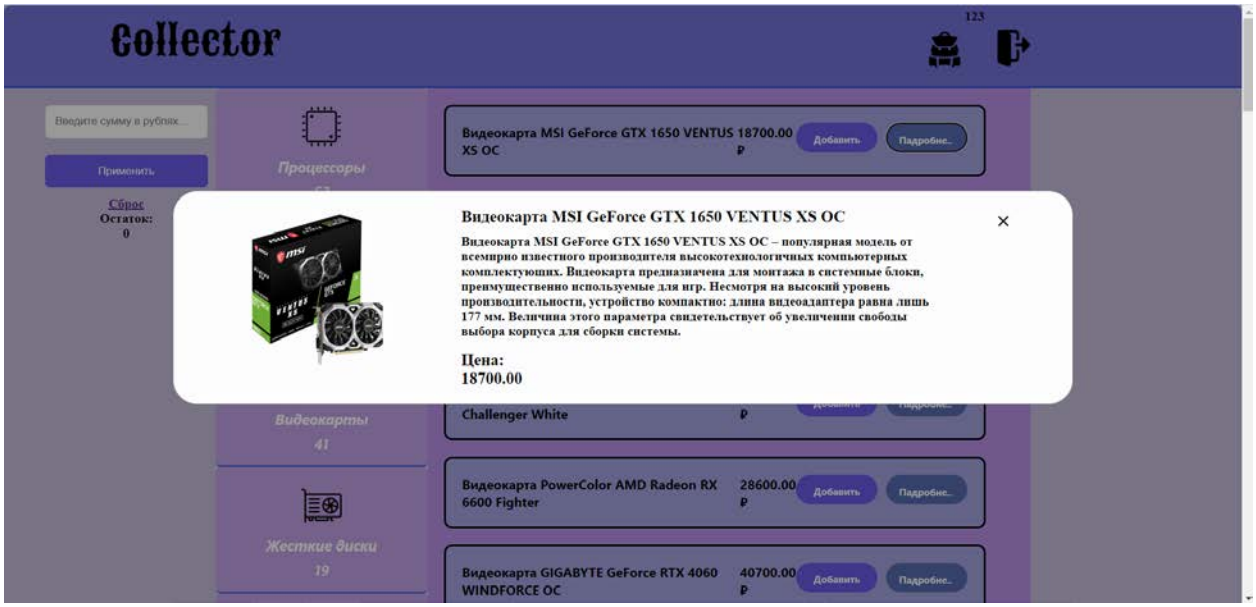

Рисунок 10 – Результат нажатия на кнопку **Подробнее**