

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Web-конфигуратор для сборки персональных компьютеров

Студент

(подпись, дата) Анохин Михаил Александрович
(ФИО полностью)

Руководитель

(подпись, дата) Макарычев П.П.

Нормоконтролёр

(подпись, дата) Такташкин Д.В.
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись) Козлов А.Ю.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись) Попова Н.А.
(фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 20__ г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент _____ Анохин Михаил Александрович _____ гр. 21 ВП2 факультета _____ ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы _____ Web-конфигуратор для сборки персональных компьютеров _____

Тема утверждена приказом ПГУ № _____ от “__” _____ 20__

3. Руководитель работы д.т.н., профессор кафедры «МОиПЭВМ» Макарычев П.П.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Разработать web-приложение — конфигуратор для сборки персональных позволяющее пользователю подбирать совместимые комплектующие с учётом характеристик, совместимости и ценовых предпочтений.

2) Среда выполнения клиентской части: современные веб-браузеры (Chrome,

Среда выполнения серверной части: операционная система Windows 10 и

Языковые средства разработки: TypeScript, Python,

Инструментальные средства разработки: PyCharm, PostgreSQL, Swagger

Git

Технологии разработки: MVC, CGI, REST API, ООП, JSON

3) Основные функции web-приложения:

а) выбор компонентов компьютера (процессор, материнская плата, SSD, видеокарта и т. д.);

б) проверка совместимости выбранных комплектующих;

в) отображение общей стоимости сборки;

г) фильтрация комплектующих по параметрам (бренд, цена, характеристики);

д) сохранение и загрузка пользовательских сборок;

е) экспорт конфигурации в PDF или JSON;

ж) администрирование базы комплектующих (для администратора).

5. Перечень подлежащих разработке вопросов

1) *Анализ предметной области и постановка задачи*

2) *Проектирование*

3) *Реализация*

4) *Контроль качества программного обеспечения*

5) *Оформление пояснительной записки*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	<i>Анализ предметной области постановка задачи</i>	10%	12.04.2025– 16.04.2025	
2	<i>Проектирование</i>	20%	17.04.2025– 26.04.2025	
3	<i>Реализация</i>	30%	27.04.2025– 11.05.2025	
4	<i>Контроль качества программного обеспечения</i>	20%	12.05.2025– 21.05.2025	
5	<i>Оформление пояснительной записки</i>	20%	22.05.2025– 01.06.2025	

Дата выдачи задания "22" декабря 2025

Руководитель бакалаврской работы _____

(подпись, дата)

Задание к исполнению принял студент _____

(подпись, дата)

Работу к защите допустить

Декан факультета _____ Л.Р. Фионова

(подпись, дата)

Реферат

Пояснительная записка содержит 241 листов, 66 рисунков, 24 таблиц, 32 использованных источника, 5 приложений.

WEB-КОНФИГУРАТОР, WINDOWS, PYTHON, RUP, SQL, UML, PostgreSQL, КОНФИГУРАЦИЯ, СБОРКА, ВЫБРАТЬ КОМПЛЕКТУЮЩИЕ

Объектом разработки является веб-приложение для конфигурирования персональных компьютеров, позволяющее пользователю подбирать комплектующие, проверять их совместимость, сохранять, редактировать и экспортировать готовые сборки.

Целью работы является создание удобного и функционального инструмента, предназначенного как для опытных пользователей, так и для начинающих, с возможностью динамического обновления данных и управления конфигурациями.

Серверная часть реализована на языке Python с использованием фреймворка FastAPI, клиентская – на TypeScript с использованием Next.js. Для хранения данных используется СУБД PostgreSQL. Разработка велась в среде PyCharm под управлением операционной системы Windows 10.

Приложение функционирует через веб-браузер и совместимо с любыми современными платформами, не требуя установки на устройство пользователя. Разработка велась с учетом модульности архитектуры, что позволяет в будущем легко масштабировать приложение и добавлять новые функции без значительных изменений существующего кода.

.

					ПГУ 09.03.04 – 08БР192.20 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Web-конфигуратор для сборки персональных компьютеров. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Анохин М.А.					4	241
Пров.		Макарычев П.П.						
Реценз.								
Н. контр.		Такташкин Д.В.						
Утв.		Козлов А.Ю.				Группа 21ВП2		

Содержание

Введение.....	7
1 Анализ предметной области и требований.....	9
1.1 Анализ предметной области.....	9
1.2 Анализ программных продуктов аналогичного назначения	10
1.2.1 Обзор web-конфигуратора «PCPartPicker»	11
1.2.2 Обзор web-конфигуратора «HYPERPC».....	11
1.2.3 Обзор web-конфигуратора «DNS».....	13
1.2.4 Сравнение аналогов	14
1.3 Анализ целевой аудитории.....	15
1.4 Анализ функциональных требований	17
1.5 Анализ нефункциональных требований	21
1.6 Планирование разработки и оценка бюджета.....	21
1.7 Постановка задачи и предполагаемое решение	33
2 Проектирование программного обеспечения.....	34
2.1 Выбор технологий и инструментальных средств разработки.....	34
2.2 Проектирование архитектуры программного обеспечения.....	35
2.3 Проектирование базы данных.....	36
2.3.1 Концептуальное проектирование базы данных	36
2.3.2 Логическое проектирование базы данных.....	38
2.4 Проектирование серверной части приложения.....	40

2.5 Проектирование интерфейса пользователя	40
3 Реализация программного обеспечения	48
3.1 Реализация базы данных	48
3.2 Реализация серверной части приложения	53
3.3 Реализация web-конфигуратора.....	55
3.4 Тестирование программного обеспечения	58
3.4.1 Тестирование серверной части приложения	58
3.4.2 Тестирование клиентской части приложения	60
3.4.3 Метрики кода.....	62
Заключение	67
Список использованных источников	68
Приложение А (справочное) Глоссарий	71
Приложение Б (обязательное) Исходный код серверной части приложения.....	73
Приложение В (обязательное) Исходный код web-конфигуратора.....	132
Приложение Г (рекомендуемое) Результаты тестирования серверной части ...	220
Приложение Д (рекомендуемое) Результаты тестирования клиентской части приложения	234

Введение

В современном обществе информационные технологии играют ключевую роль в развитии практически всех сфер жизнедеятельности человека. Информатизация и цифровизация охватывают образование, медицину, промышленность, государственное управление и, конечно, сферу торговли и услуг. Цифровые технологии существенно изменили способы потребления информации и товаров, сделав онлайн-среду привычной и удобной площадкой для большинства повседневных задач.

Одним из направлений цифровизации, получивших широкое распространение, стало развитие онлайн-сервисов, предоставляющих пользователю возможность самостоятельно выполнять сложные действия – от оформления документов до создания индивидуального продукта. К таким сервисам относится и система конфигурации персонального компьютера, предоставляющая пользователю доступ к удобному подбору и сборке совместимых комплектующих в режиме онлайн.

На фоне растущей популярности онлайн-покупок и увеличения числа пользователей, самостоятельно собирающих или модернизирующих свои ПК, всё более востребованными становятся программные средства, позволяющие автоматизировать процесс выбора комплектующих с учетом совместимости и технических требований. Это особенно актуально для начинающих пользователей, не обладающих глубокими техническими знаниями, но желающих собрать персонализированную и сбалансированную конфигурацию своего компьютера.

В рамках государственной программы «Цифровая экономика Российской Федерации» подчёркивается важность развития цифровых платформ и сервисов, ориентированных на потребителя, как инструментов повышения качества жизни граждан и конкурентоспособности страны. Создание веб-приложений, предоставляющих доступ к сложным техническим операциям через интуитивно понятный интерфейс, полностью соответствует этим целям.

Развитие e-commerce, рост интереса к геймингу, увеличивающаяся популярность удалённой работы и образования способствуют росту спроса на индивидуальные компьютерные сборки. Всё это делает актуальной разработку web-конфигуратора, позволяющего пользователю без специальных знаний подобрать компьютер под свои нужды – будь то офисная работа, графика, обучение или игры.

Таким образом, создание веб-приложения – конфигуратора персональных компьютеров – является своевременной задачей, отвечающей вызовам цифровой экономики и удовлетворяющей реальные потребности пользователей.

1 Анализ предметной области и требований

1.1 Анализ предметной области

На современном рынке цифровых технологий пользователи всё чаще сталкиваются с необходимостью самостоятельного выбора и конфигурации персонального компьютера, исходя из собственных требований к производительности, стоимости и назначению. Сборка ПК «под себя» позволяет получить оптимальное сочетание комплектующих и снизить затраты по сравнению с покупкой готовых решений. Однако данный процесс требует от пользователя определённых знаний в области совместимости оборудования и оценки технических характеристик.

Для решения этой задачи всё чаще применяются онлайн-конфигураторы персональных компьютеров – специализированные программные продукты, позволяющие выбрать комплектующие, проверить их совместимость, оценить итоговую стоимость и оформить заказ. Такие системы упрощают процесс конфигурации и делают его доступным даже для пользователей без технического образования.

Разрабатываемое программное обеспечение должно представлять собой удобный и интуитивно понятный веб-сервис или приложение, предоставляющее пользователю следующие возможности:

- выбор и замена основных компонентов компьютера (процессора, материнской платы, видеокарты, оперативной памяти, накопителей, блока питания и т. д.);
- автоматическая проверка совместимости выбранных компонентов;
- отображение актуальных цен и наличия комплектующих;
- расчёт ориентировочной стоимости всей сборки;
- возможность сохранить конфигурацию для дальнейшего редактирования;
- экспорт и отправка конфигурации (например, в виде PDF или ссылки);

- оформление заказа либо перенаправление в интернет-магазин партнёра;
- просмотр ранее сохранённыхборок;
- авторизация и регистрация пользователя для работы с личным кабинетом;
- администрирование каталога компонентов (для модераторов и администраторов системы).

Таким образом, в рамках проекта необходимо реализовать информационную систему, включающую следующие основные функции:

- конфигурация персонального компьютера с выбором комплектующих;
- проверка совместимости компонентов;
- отображение характеристик и стоимости сборки;
- сохранение и загрузка пользовательских конфигураций;
- экспорт конфигурации и оформление заказа;
- администрирование каталога компонентов (для админов).

Функциональность системы должна обеспечить простоту использования для конечного пользователя, а также поддерживать расширение каталога компонентов и масштабируемость проекта в будущем.

1.2 Анализ программных продуктов аналогичного назначения

В настоящее время существует большое количество веб-сервисов и приложений, предназначенных для подбора и конфигурации персональных компьютеров. Они позволяют пользователям выбирать комплектующие, сравнивать цены и совместимость компонентов, а также визуализировать итоговую сборку. Многие из этих сервисов предлагают готовые конфигурации под разные задачи – от офисной работы до профессионального гейминга и монтажа видео. Некоторые платформы также интегрированы с онлайн-магазинами. Для анализа приведены наиболее известные из них.

1.2.1 Обзор web-конфигуратора «PCPartPicker»

PCPartPicker – это один из наиболее популярных международных сервисов для конфигурации персональных компьютеров. Он ориентирован преимущественно на англоязычную аудиторию и включает в себя широкий набор функций: от выбора комплектующих до анализа совместимости и мониторинга цен в магазинах [1].

Главная особенность сервиса – проверка совместимости компонентов и автоматический подбор параметров, таких как необходимая мощность блока питания. Также реализована возможность публикации пользовательских сборок, их комментирования и сравнения.

Достоинства:

- продвинутая система проверки совместимости;
- отслеживание цен в разных магазинах;
- поддержка сохранения и экспорта сборок;
- большое сообщество пользователей.

Недостатки:

- отсутствие русского языка;
- неактуальные цены и магазины для России;
- интерфейс сложен для новичков.

Таким образом, сервис PCPartPicker представляет собой мощный инструмент для опытных пользователей, однако слабо адаптирован под российский рынок.

1.2.2 Обзор web-конфигуратора «HYPERPC»

HYPERPC – российская компания, специализирующаяся на сборке мощных игровых и рабочих компьютеров. На официальном сайте компании доступен интерактивный web-конфигуратор, позволяющий подобрать комплектующие для игровых, рабочих и кастомных ПК. Главная страница конфигуратора представлена на рисунке 1.

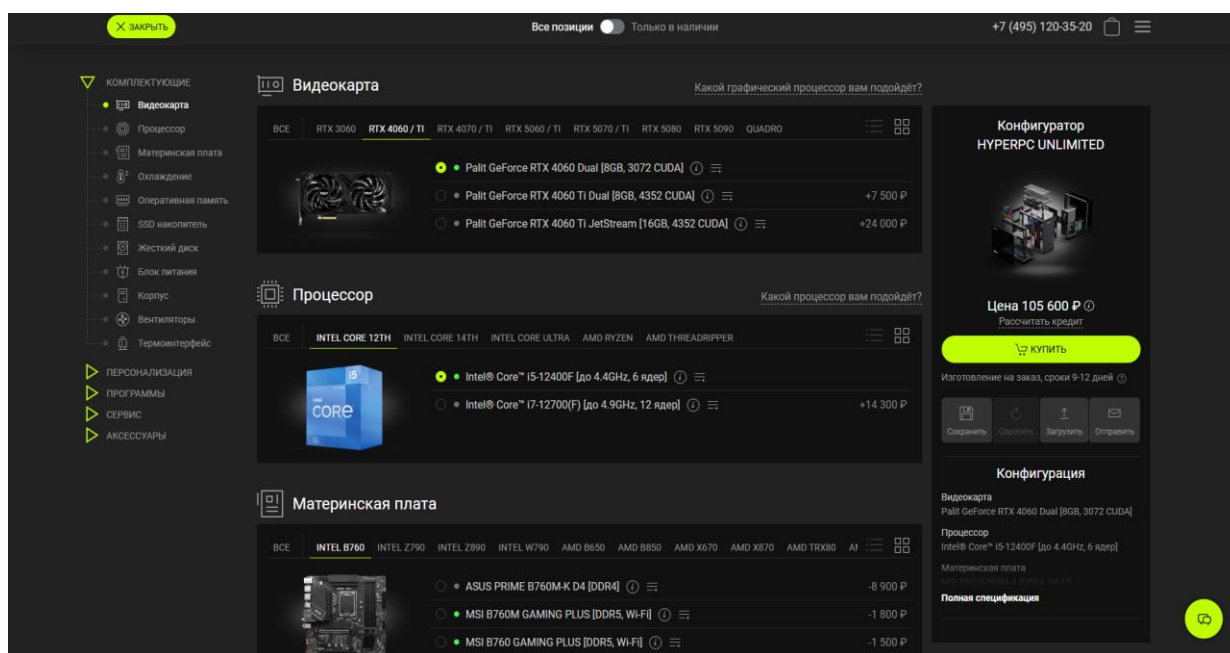


Рисунок 1 – Интерфейс конфигуратора HYPERPC

Интерфейс визуально ориентирован на конечного пользователя: сборка представлена в виде 3D-модели, с возможностью менять подсветку, блоки охлаждения и другие параметры. При выборе компонентов автоматически фильтруются несовместимые товары, что минимизирует вероятность ошибок.

Достоинства:

- современный и наглядный интерфейс;
- проверка совместимости компонентов;
- поддержка визуализации сборки (3D);
- возможность оформления заказа прямо через сайт.

Недостатки:

- ограниченный выбор комплектующих (только из каталога HYPERPC);
- отсутствие экспорта конфигурации;
- высокие цены по сравнению с розничными магазинами.

Таким образом, конфигуратор HYPERPC отличается высокой наглядностью и удобством, однако его функциональность ограничена рамками корпоративного каталога.

1.2.3 Обзор web-конфигуратора «DNS»

Сеть магазинов DNS предоставляет пользователям возможность собрать персональный компьютер с помощью встроенного конфигуратора. Данный инструмент доступен на официальном сайте компании и представляет собой пошаговый мастер выбора комплектующих. Пример интерфейса приведён на рисунке 2.

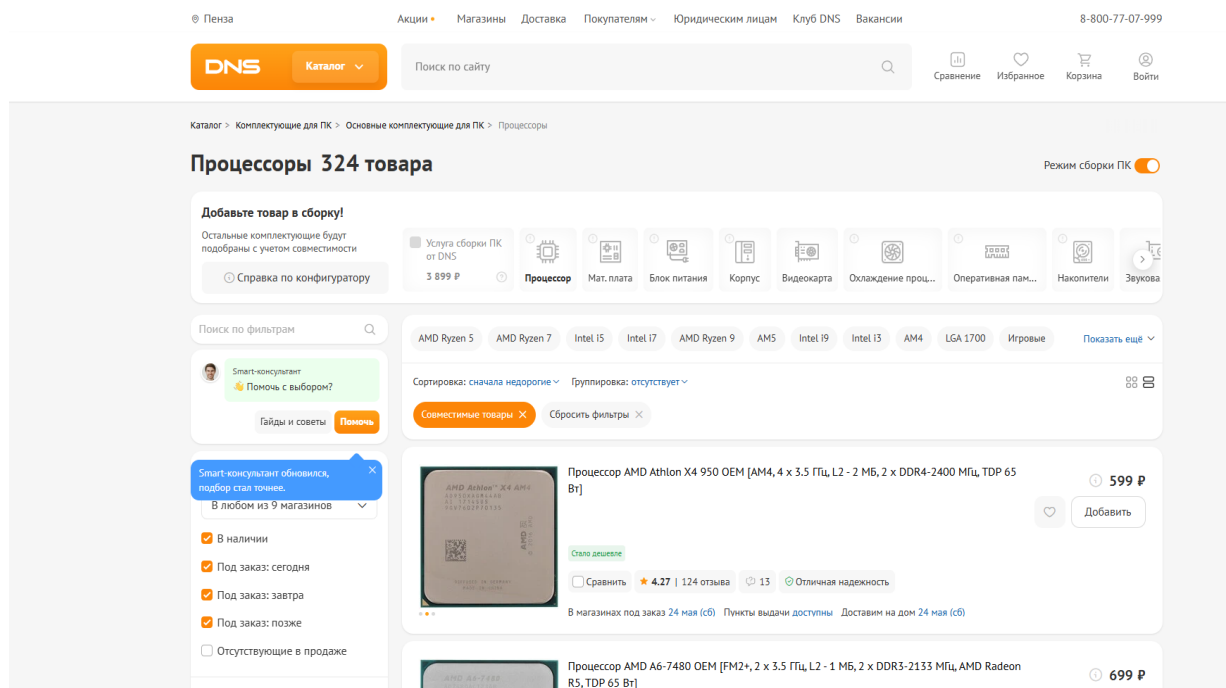


Рисунок 2 –Интерфейс конфигуратора DNS

Пользователь может выбрать базовую конфигурацию и адаптировать её под свои задачи. Программа автоматически проверяет совместимость основных компонентов, таких как процессор и материнская плата, видеокарта и блок питания.

Достоинства:

- простота и понятность интерфейса;
- проверка совместимости основных компонентов;
- возможность оформления заказа в магазине DNS;
- актуальные цены и наличие товаров.

Недостатки:

- ограничения по ассортименту (только товары из DNS);

- отсутствие расширенных функций (например, оценки энергопотребления);
- нет возможности экспортировать сборку во внешние сервисы.

Таким образом, конфигуратор от DNS является удобным решением для широкого круга пользователей, однако обладает ограниченными возможностями по сравнению с специализированными платформами.

1.2.4 Сравнение аналогов

На данном этапе был произведен сравнительный анализ рассмотренных ранее web-конфигураторов. В результате анализа была сформирована таблица 1.

Таблица 1 – Сравнение аналогов

Программный продукт	Проверка совместимости	Расчёт стоимости	Экспорт конфигурации	Оформление заказа
PCPartPicker	Есть	Есть	Есть	Нет
HYPERPC Configurator	Есть	Есть	Нет	Есть
DNS Конфигуратор	Нет	Есть	Нет	Есть
Разрабатываемый конфигуратор	Есть	Есть	Есть	Есть

Из проведённого сравнения видно, что большинство рассмотренных конфигураторов не поддерживают полный набор функций. В частности, функция оформления заказа отсутствует в PCPartPicker, а экспорт конфигурации недоступен в HYPERPC и DNS. Только разрабатываемый конфигуратор включает все ключевые функции, такие как проверка совместимости, расчёт стоимости, экспорт конфигурации и оформление заказа, что делает его более универсальным решением по сравнению с существующими аналогами.

1.3 Анализ целевой аудитории

Целевой аудиторией разрабатываемого web-конфигуратора являются пользователи, заинтересованные в сборке персонального компьютера под собственные задачи – будь то игровое использование, работа с графикой и видео, офисная деятельность или иные цели. Среди таких пользователей можно выделить начинающих сборщиков, энтузиастов, а также консультантов компьютерных магазинов [2].

В последние годы наблюдается устойчивый рост интереса к самостоятельной сборке компьютеров. По данным отчета Statista [3], рынок комплектующих ПК в мире стабильно растёт с 2019 года, несмотря на временные падения из-за кризисов поставок. На рисунке 3 представлена динамика глобальных поставок компьютерных комплектующих с 2015 по 2024 годы, а также прогноз на 2025 год.



Рисунок 3 –Динамика мирового рынка комплектующих для ПК

График показывает рост рынка с заметным скачком в 2020 году (из-за пандемии и роста спроса на ПК), последующим спадом и восстановлением. Прогноз на 2025 год указывает на дальнейшее увеличение спроса, связанное с

развитием игровых технологий, облачных вычислений, ИИ и потребностей пользователей в более производительных системах.

В России отмечается рост интереса к индивидуальной сборке ПК. По данным маркетплейса Ozon, в 2023 году спрос на видеокарты, процессоры и SSD-диски вырос в 1,7 раза по сравнению с 2022 годом. Это подтверждает, что всё больше пользователей предпочитают собирать ПК самостоятельно или прибегают к помощи специализированных онлайн-инструментов.

Существенную роль в развитии таких сервисов играет рост проникновения Интернета и цифровой грамотности. Согласно данным Яндекса, к 2024 году более 86% россиян имеют стабильный доступ в Интернет, а уровень цифровой компетенции пользователей продолжает расти. Это открывает возможности для создания интуитивно понятных и полезных онлайн-сервисов, к числу которых относится и разрабатываемое приложение.

Кроме того, благодаря широкому использованию современных браузеров и развитию JavaScript-фреймворков, веб-приложения могут эффективно заменять нативные программы, предоставляя при этом кроссплатформенную доступность.

С технической точки зрения, web-конфигуратор должен обеспечивать пользователю:

- простой и логичный интерфейс подбора комплектующих;
- проверку совместимости выбранных компонентов;
- расчет суммарной стоимости сборки;
- возможность сохранения или экспорта конфигурации.

Целью разработки является создание удобного веб-сервиса, позволяющего пользователям собрать конфигурацию ПК из доступных компонентов, получить предварительный расчет стоимости, а также сохранить полученную конфигурацию.

Приложение будет работать в среде веб-браузеров, что обеспечит его доступность на любых устройствах – как на ПК, так и на мобильных платформах, без необходимости установки дополнительного ПО.

1.4 Анализ функциональных требований

Основным пользователем разрабатываемого веб-приложения является клиент, заинтересованный в самостоятельной сборке персонального компьютера. Через интерфейс configurатора он должен иметь возможность создания новой конфигурации, выбора комплектующих, проверки их совместимости и получения подсказок в случае несоответствий. Также пользователь может редактировать ранее созданную сборку, просматривать рекомендованные варианты, оценивать итоговую стоимость, сохранять и экспортировать конфигурации, а при необходимости – удалять сохранённые данные. Вся работа с данными осуществляется через взаимодействие с базой данных, которая обеспечивает хранение и обработку информации о комплектующих и пользовательских сборках.

На основе всего вышесказанного была создана диаграмма вариантов использования, представленная на рисунке 4 [4].

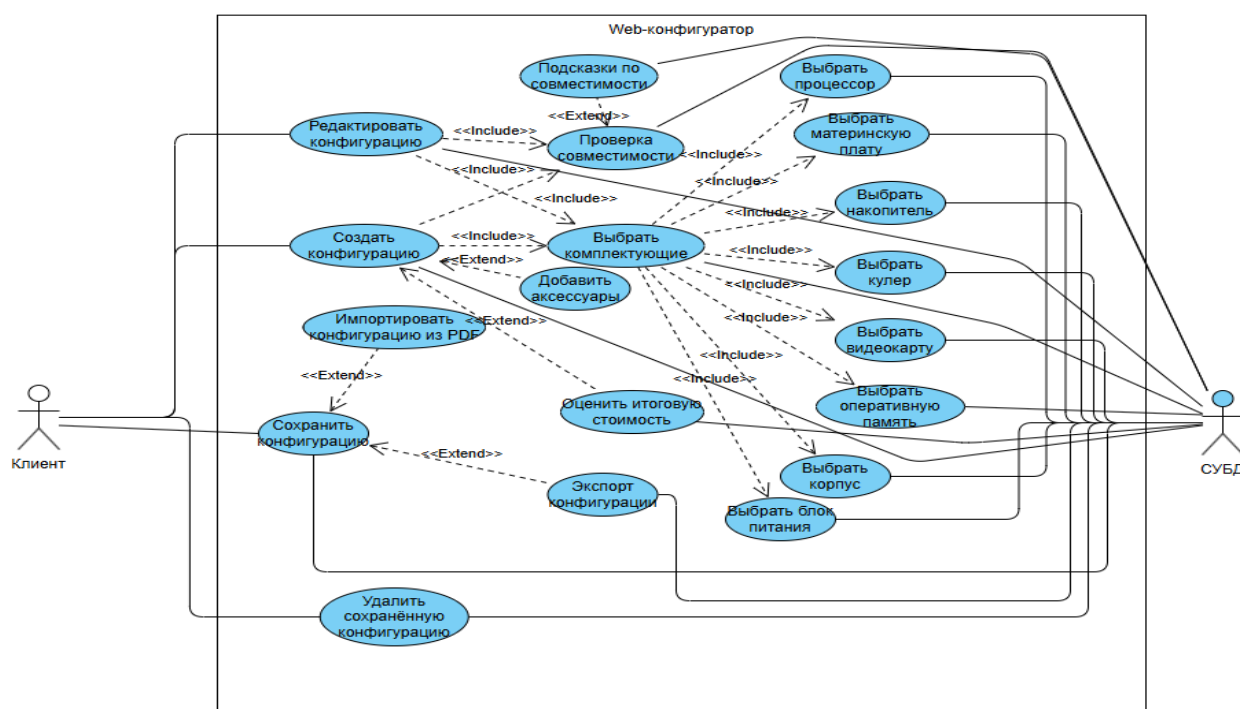


Рисунок 4 – Диаграмма вариантов использования web-конфигуратора

Составим спецификации для основных вариантов использования. «Создать конфигурацию» (рисунок 5).

Наименование: создать конфигурацию
ID: 1
Краткое описание: Клиент создает новую конфигурацию компьютера, выбирая необходимые комплектующие
Действующие лица: клиент, СУБД
Предусловие: Клиент вошел в систему или использует веб-конфигуратор
Основной поток: <ol style="list-style-type: none"> 1. Клиент инициирует создание новой конфигурации; 2. Система предлагает выбор комплектующих (процессор, материнская плата и т. д.); 3. Клиент выбирает необходимые комплектующие; 4. Система включает проверку совместимости между выбранными комплектующими 5. Система рассчитывает итоговую стоимость; 6. Система сохраняет промежуточные данные в памяти или в базе (при необходимости).
Постусловие: Конфигурация создана и готова к сохранению или редактированию.

Рисунок 5 – Спецификация варианта использования «Создать конфигурацию»

«Сохранить конфигурацию» (рисунок 6).

Наименование: сохранить конфигурацию
ID: 2
Краткое описание: клиент сохраняет созданную конфигурацию в системе
Действующие лица: клиент, СУБД
Предусловие: создана конфигурация, доступна для сохранения
Основной поток: <ol style="list-style-type: none"> 1. → Клиент нажимает «Сохранить конфигурацию»; 2. → Система предлагает ввести название или комментарий к конфигурации; 3. → Система предлагает добавить аксессуар к конфигурации. 4. → Система сохраняет конфигурацию в БД; 5. → Система подтверждает успешное сохранение.
Постусловие: Конфигурация сохранена в СУБД и доступна в списке сохраненных.

Рисунок 6 – Спецификация варианта использования «Сохранить конфигурацию»

«Редактировать конфигурацию» (рисунок 7).

Наименование: редактировать конфигурацию
ID: 3
Краткое описание: Клиент изменяет ранее сохраненную конфигурацию.
Действующие лица: клиент, СУБД
Предусловие: существует сохранённая конфигурация
Основной поток: <ol style="list-style-type: none"> 1. Клиент выбирает конфигурацию для редактирования; 2. Система загружает данные конфигурации из БД; 3. Клиент изменяет комплектующие; 4. Система проверяет совместимость и обновляет итоговую стоимость; 5. Клиент сохраняет изменения; 6. Система обновляет данные в БД.
Постусловие: Конфигурация обновлена

Рисунок 7 – Спецификация варианта использования «Редактировать конфигурацию»

Помимо спецификаций для одного прецедента были составлены диаграмма деятельности и диаграмма последовательности. Для варианта использования «Создать конфигурацию» составлена диаграмма деятельности, представленная на рисунке 8.

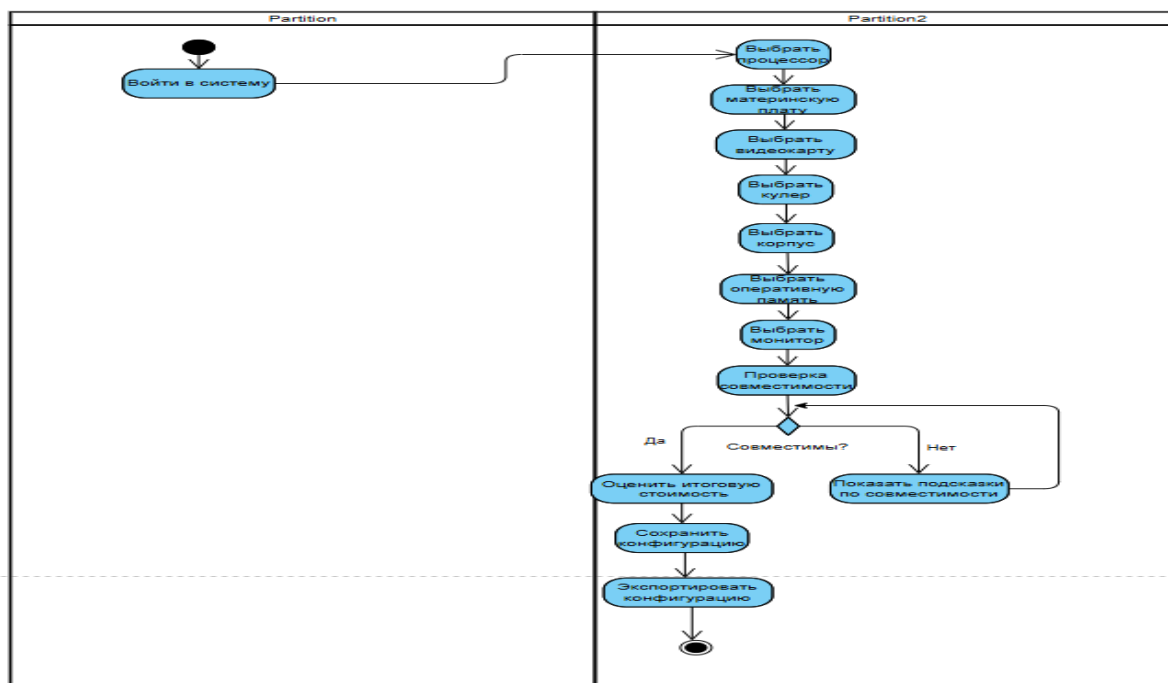


Рисунок 8 – Диаграмма деятельности варианта использования «Создать конфигурацию»

Для того же варианта использования была составлена диаграмма последовательности, приведенная на рисунке 9.

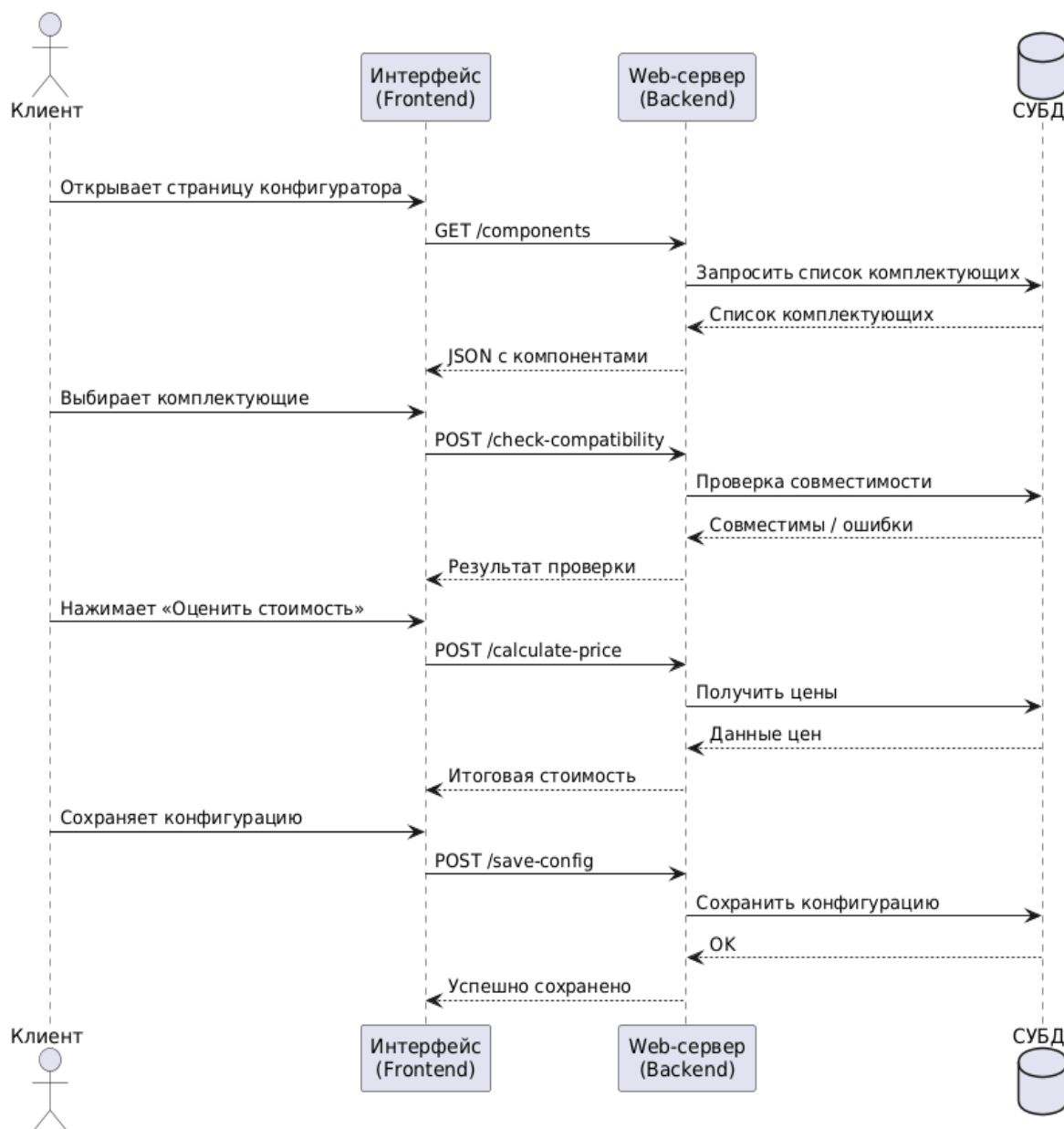


Рисунок 9 – Диаграмма последовательности варианта использования «Создать конфигурацию»

На данном этапе были определены основные функции разрабатываемого web-конфигуратора. При этом для данного приложения были построены диаграммы вариантов использования, деятельности и последовательности. На основе построенных диаграмм был проведен детальный анализ требований к системе и определены основные сценарии взаимодействия пользователей с

функционалом configurator. Полученные результаты легли в основу технического задания и помогут избежать противоречий при дальнейшей разработке системы.

1.5 Анализ нефункциональных требований

На данном этапе были определены нефункциональные требования к разрабатываемому программному обеспечению. Должны быть использованы такие технологии разработки, как RUP, ООП, MVC [5, 6].

Серверная часть приложения должна быть разработана на языке программирования Python в среде программирования PyCharm. Данные на сервере должны храниться в базе данных, реализованной с использованием СУБД PostgreSQL, запросы к базе данных должны осуществляться на языке SQL. Серверная часть должна работать под управлением операционной системы Windows 10 и выше.

Клиентская часть приложения должна быть реализована на языке TypeScript с использованием библиотеки React и фреймвока Next.js. Она должна представлять собой web-конфигуратор с удобным и интуитивно понятным пользовательским интерфейсом. Клиентское приложение должно быть отлажено и должно успешно работать под управлением операционной системы Windows 10 и выше.

1.6 Планирование разработки и оценка бюджета

Без правильного планирования невозможно качественно и в установленные сроки выполнить работы по любому проекту. Это касается и сферы информационных технологий. Планирование предполагает построение графика выполнения работ проекта таким образом, чтобы выполнить все поставленные задачи с приемлемым качеством за минимально возможное время с использованием минимума затрат [7 - 9].

Задачи, представляющие собой этапы работы над проектом, представлены в таблице 2.

Таблица 2 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области и постановка задачи	1 день	1
4	Анализ программных продуктов аналогичного назначения	3 дня	1
5	Анализ функциональных требований	2 дня	3;4
6	Анализ требований к интерфейсу	2 дня	5
7	Анализ целевой аудитории	1 день	5
8	Анализ требований завершен	-	6;7
9	Выбор решения	-	-
10	Анализ существующих технологий разработки ПО	2 дня	8
11	Выбор технологий разработки ПО	1 день	10
12	Выбор решения завершен	-	11
13	Проектирование программных средств	-	-
14	Проектирование архитектуры приложения	5 дней	12
15	Проектирование базы данных	3 дня	12
16	Проектирование серверной части приложения	3 дня	14;15
17	Проектирование web-конфигуратор приложения	4 дня	14
18	Проектирование интерфейса пользователя	2 дня	14
19	Проектирование завершенно	-	16;17;18
20	Реализация программных средств	-	-
21	Реализация информационной системы	5 дней	14;15
22	Реализация серверной части приложения	19 дней	19;21
23	Реализация web-конфигуратора	21 день	19;21
24	Реализация завершенна	-	22;23
25	Тестирование	-	-
26	Изучение основных видов тестирования	1 день	24
27	Функциональное тестирование	2 дня	26
28	Модульное тестирование	3 дня	26
29	Тестирование завершенно	-	27;28
30	Составление и оформление документации	7 дней	29
31	Конец проекта	-	30

Ниже в таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Перечень ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист 1	Трудовой	400 руб/ч	600 руб/ч
Программист 2	Трудовой	350 руб/ч	550 руб/ч
Аналитик	Трудовой	350 руб/ч	550 руб/ч
Дизайнер	Трудовой	300 руб/ч	400 руб/ч
Тестировщик	Трудовой	300 руб/ч	400 руб/ч
Компьютер	Материальный	2000 руб	-
Бумага	Материальный	500 руб	-
Интернет	Затраты	-	-

В таблице 4 приведено распределение ресурсов по задачам.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области и постановка задачи	Аналитик Компьютер Интернет	40% 3 1500 руб
4	Анализ программных продуктов аналогичного назначения	Аналитик	50%
5	Анализ функциональных требований	Программист 1 Программист 2 Аналитик	100% 100% 50%
6	Анализ требований к интерфейсу	Дизайнер Аналитик	100% 30%
7	Анализ целевой аудитории	Аналитик	50%
8	Анализ требований завершен	-	-
9	Выбор решения	-	-
10	Анализ существующих технологий разработки ПО	Программист 2	30%
11	Выбор технологий разработки ПО	Программист 1 Программист 2	50% 50%
12	Выбор решения завершен	-	-
13	Проектирование программных средств	-	-
14	Проектирование архитектуры приложения	Программист 1 Программист 2 Аналитик	100% 100% 50%
15	Проектирование базы данных	Программист 2 Аналитик	70% 50%
16	Проектирование серверной части приложения	Программист 1 Аналитик	100% 30%
17	Проектирование веб-конф приложения	Программист 2 Аналитик	100% 30%
18	Проектирование интерфейса пользователя	Дизайнер Аналитик	60% 50%
19	Проектирование завершено	-	-
20	Реализация программных средств	-	-
21	Реализация информационной системы	Программист 2	100%
22	Реализация серверной части приложения	Программист 1	100%
23	Реализация web-конфигуратора	Программист 2	100%
24	Реализация завершена	-	-
25	Тестирование	-	-
26	Изучение основных видов тестирования	Программист 2	30%
27	Функциональное тестирование	Тестирующий	70%
28	Модульное тестирование	Тестирующий Программист 2	70% 50%
29	Тестирование завершено	-	-
30	Составление и оформление документации	Аналитик Программист 1 Программист 2 Бумага	100% 50% 50% 2
31	Конец проекта	-	-

Таким образом, реализацией проекта будут заниматься программист 1, программист 2, аналитик и дизайнер, которые в процессе работы используют три компьютера и доступ в интернет. Для вычисления длительности реализации проекта и его критического пути был построен сетевой график, представленный на рисунке 10.

Теперь необходимо рассчитать критический путь проекта. Для этого необходимо высчитать раннее и позднее времена начала работ для всех задач проекта, и на основе данных сведений определить резерв времени для каждой из задач.

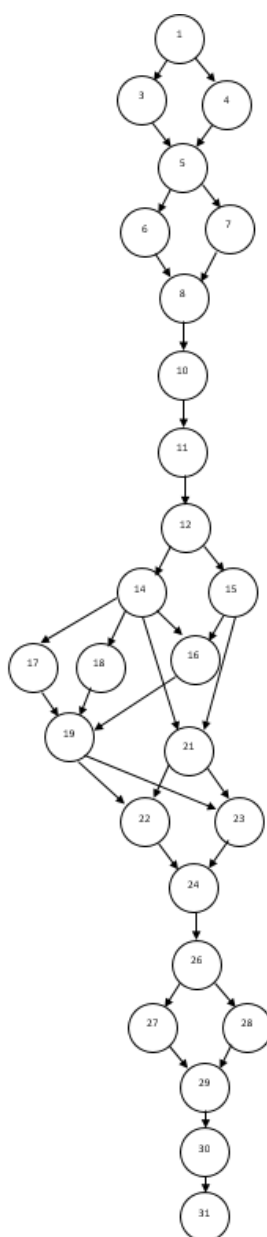


Рисунок 10 – Сетевой график проекта

В таблице 5 представлены расчеты раннего времени начала работ для его задач.

Таблица 5 – Расчеты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	1	0	0	0+0	0
5	3	0	1	Max(0+1;0+3)	3
	4	0	3		
6	5	3	2	3+2	5
7	5	3	2	3+2	5
8	6	5	2	Max(5+2;5+1)	7
	7	5	1		
10	8	7	0	7+0	7
11	10	7	2	7+2	9
12	11	9	1	9+1	10
14	12	10	0	10+0	10
15	12	10	0	10+0	10
16	14	10	5	Max(10+5;10+3)	15
	15	10	3		
17	14	10	5	10+5	15
18	14	10	5	10+5	15
19	16	15	3	Max(15+3;15+4;15+2)	19
	17	15	4		
	18	15	2		
21	14	10	5	Max(10+5;10+3)	15
	15	10	3		
22	19	19	0	Max(19+0;15+5)	20
	21	15	5		
23	19	19	0	Max(19+0;15+5)	20
	21	15	5		
24	22	20	19	Max(20+19;20+21)	41
	23	20	21		
26	24	41	0	41+0	41
27	26	41	1	41+1	42
28	26	41	1	41+1	42
29	27	42	2	Max(42+2;42+3)	45
	28	42	3		
30	29	45	0	45+0	45
31	30	45	7	45+7	52

Также было рассчитано позднее время начала работ для задач проекта (таблица 6). Это позволило оптимизировать график выполнения работ и минимизировать риски задержек на критических этапах разработки.

Таблица 6 – Расчеты позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
31	-	-	-	-	52
30	31	52	7	52 - 7	45
29	30	45	0	45 - 0	45

Продолжение таблицы 6

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
28	29	45	3	45 - 3	42
27	29	45	2	45 - 2	43
26	27 28	43 42	1	Min(43 - 1; 42 - 1)	41
24	26	41	0	41 - 0	41
23	24	41	21	41 - 21	20
22	24	41	19	41 - 19	22
21	22 23	22 20	5	Min(22 - 5; 20 - 5)	15
19	22 23	22 20	0	Min(22 - 0; 20 - 0)	20
18	19	20	2	20 - 2	18
17	19	20	4	20 - 4	16
16	19	20	3	20 - 3	17
15	16 21	17 15	3	Min(17 - 3; 15 - 3)	12
14	16 17 18 21	17 16 18 15	5	Min(17 - 5; 16 - 5; 18 - 5; 15 - 5)	10
12	14 15	10 12	0	Min(10 - 0; 12 - 0)	10
11	12	10	1	10 - 1	9
10	11	9	2	9 - 2	7
8	10	7	0	7 - 0	7
7	8	7	1	7 - 1	6
6	8	7	2	7 - 2	5
5	6 7	5 6	2	Min(5 - 2; 6 - 2)	3
4	5	3	3	3 - 3	0
3	5	3	1	3 - 1	2
1	3 4	2 0	0	Min(2 - 0; 0 - 0)	0

По результатам нахождения раннего и позднего времени начала работ можно найти критический путь проекта. Вычисление критического пути представлено в таблице 7.

Таблица 7 – Вычисление критического пути проекта

Номер работы	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	2	2
4	0	0	0
5	3	3	0
6	5	5	0
7	5	6	1
8	7	7	0
10	7	7	0
11	9	9	0
12	10	10	0
14	10	10	0

Продолжение таблицы 7

Номер работы	Раннее время начала	Позднее время начала	Резерв времени
15	10	12	2
16	15	17	2
17	15	16	1
18	15	18	3
19	19	20	1
21	15	15	0
22	20	22	2
23	20	20	0
24	41	41	0
26	41	41	0
27	42	43	1
28	42	42	0
29	45	45	0
30	45	45	0
31	52	52	0

На основе полученных данных можно сделать вывод, что критический путь проходит через работы с номерами 1, 4, 5, 6, 8, 10, 11, 12, 14, 21, 23, 24, 26, 28, 29, 30, 31, а общее время выполнения проекта составляет 52 дня. Это значение учитывает последовательность задач, которые напрямую влияют на сроки завершения проекта. Оптимизация работ, не входящих в критический путь, может предоставить дополнительные возможности для распределения ресурсов.

Выявленная последовательность работ позволяет сосредоточить внимание на ключевых этапах, требующих особого контроля.

При необходимости сроки могут быть пересмотрены за счет оптимизации отдельных процессов или перераспределения ресурсов. Данный расчет служит основой для дальнейшего мониторинга и управления ходом проекта.

Внедрение регулярного контроля за выполнением этапов поможет своевременно выявлять отклонения и корректировать планы. Использование современных инструментов управления проектами повысит прозрачность процессов и улучшит координацию между командами. Кроме того, важно предусмотреть резервные ресурсы на случай непредвиденных задержек, чтобы минимизировать их влияние на общие сроки проекта.

Граф проекта, где пунктирными линиями указан критический путь, приведен на рисунке 11.

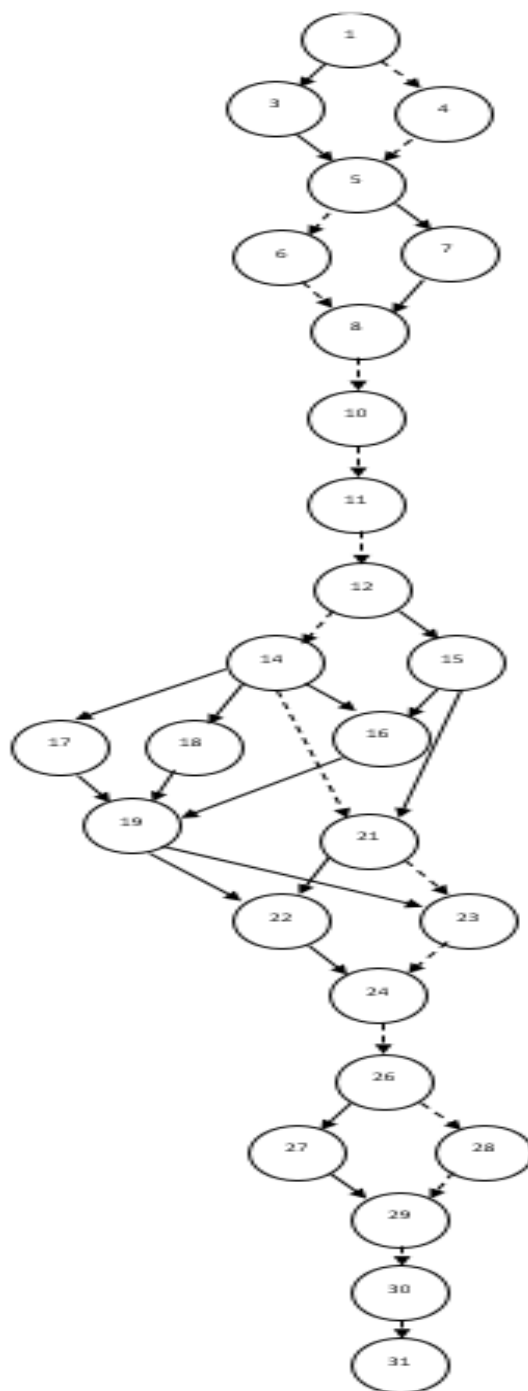


Рисунок 11 – Критический путь проекта

Теперь проверим правильность расчетов продолжительности проекта и вычислим величину его бюджета. Для этого воспользуемся системой управления проектами Microsoft Project 2007, которая позволяет автоматически произвести все необходимые вычисления [24]. Список составленных задач в системе Microsoft Project представлен на рисунке 12. В качестве времени начала проекта была выбрана дата 10.03.2025.

	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало проекта	0 дней	Пн 10.03.15	Пн 10.03.15		
2	- Анализ требований	7 дней	Пн 10.03.15	Пн 20.03.15		
3	Анализ предметной области и постановка задачи	1 день	Пн 10.03.15	Пн 10.03.15	1	Аналитик[40%]; Компьютер[3]; Интернет[1 500,00 Р]
4	Анализ программных продуктов аналогичного назначения	3 дня	Пн 10.03.15	Вт 14.03.15	1	Аналитик[50%]
5	Анализ функциональных требований	2 дня	Ср 15.03.15	Чт 16.03.15	3,4	Программист 1; Программист 2; Аналитик[50%]
6	Анализ требований к интерфейсу	2 дня	Пн 17.03.15	Пн 20.03.15	5	Дизайнер; Аналитик[30%]
7	Анализ целевой аудитории	1 день	Пн 17.03.15	Пн 17.03.15	5	Аналитик[50%]
8	Анализ требований завершения	0 дней	Пн 20.03.15	Пн 20.03.15	6,7	
9	- Выбор решения	3 дня	Вт 23.03.15	Чт 23.03.15		
10	Анализ существующих технологий разработки ПО	2 дня	Вт 23.03.15	Ср 24.03.15	8	Программист 2[30%]
11	Выбор технологий разработки ПО	1 день	Чт 23.03.15	Чт 23.03.15	10	Программист 1[50%]; Программист 2[50%]
12	Выбор решения завершения	0 дней	Чт 23.03.15	Чт 23.03.15	11	
13	- Проектирование программных средств	9 дней	Пн 24.03.15	Ср 05.04.15		
14	Проектирование архитектуры приложений	5 дней	Пн 24.03.15	Чт 30.03.15	12	Программист 1; Программист 2; Аналитик[50%]
15	Проектирование базы данных	3 дня	Пн 24.03.15	Вт 28.03.15	12	Программист 2[70%]; Аналитик[50%]
16	Проектирование серверной части приложения	3 дня	Пн 31.03.15	Вт 04.04.15	14,15	Программист 1; Аналитик[30%]
17	Проектирование веб-конфигуратора	4 дня	Пн 31.03.15	Ср 06.04.15	14	Программист 2; Аналитик[30%]
18	Проектирование интерфейса пользователя	2 дня	Пн 31.03.15	Пн 03.04.15	14	Дизайнер[60%]; Аналитик[50%]
19	Проектирование завершено	0 дней	Ср 06.04.15	Ср 06.04.15	16,17,18	
20	- Реализация программных средств	26 дней	Пн 31.03.15	Ср 10.05.15		
21	Реализация информационной системы	5 дней	Пн 31.03.15	Чт 06.04.15	14,15	Программист 2
22	Реализация серверной части приложения	19 дней	Пн 07.04.15	Чт 04.05.15	19,21	Программист 1
23	Реализация веб-конфигуратора	21 день	Пн 07.04.15	Ср 10.05.15	19,21	Программист 2
24	Реализация завершения	0 дней	Ср 10.05.15	Ср 10.05.15	22,23	
25	- Тестирование	4 дня	Чт 11.05.15	Вт 16.05.15		
26	Изучение основных видов тестирования	1 день	Чт 11.05.15	Чт 11.05.15	24	Программист 2[30%]
27	Функциональное тестирование	2 дня	Пн 12.05.15	Пн 15.05.15	26	Тестировщик[70%]
28	Модульное тестирование	3 дня	Пн 12.05.15	Вт 16.05.15	26	Тестировщик[70%]; Программист 2[50%]
29	Тестирование завершено	0 дней	Вт 16.05.15	Вт 16.05.15	27,28	
30	Составление и оформление документации	7 дней	Ср 17.05.15	Чт 26.05.15	29	Аналитик; Программист 1[50%]; Программист 2[50%]; Бумага[2]
31	Конец проекта	0 дней	Чт 26.05.15	Чт 26.05.15	30	

Рисунок 12 – Список задач проекта в системе Microsoft Project

Построенная системой Microsoft Project диаграмма Ганта приведена на рисунке 13.

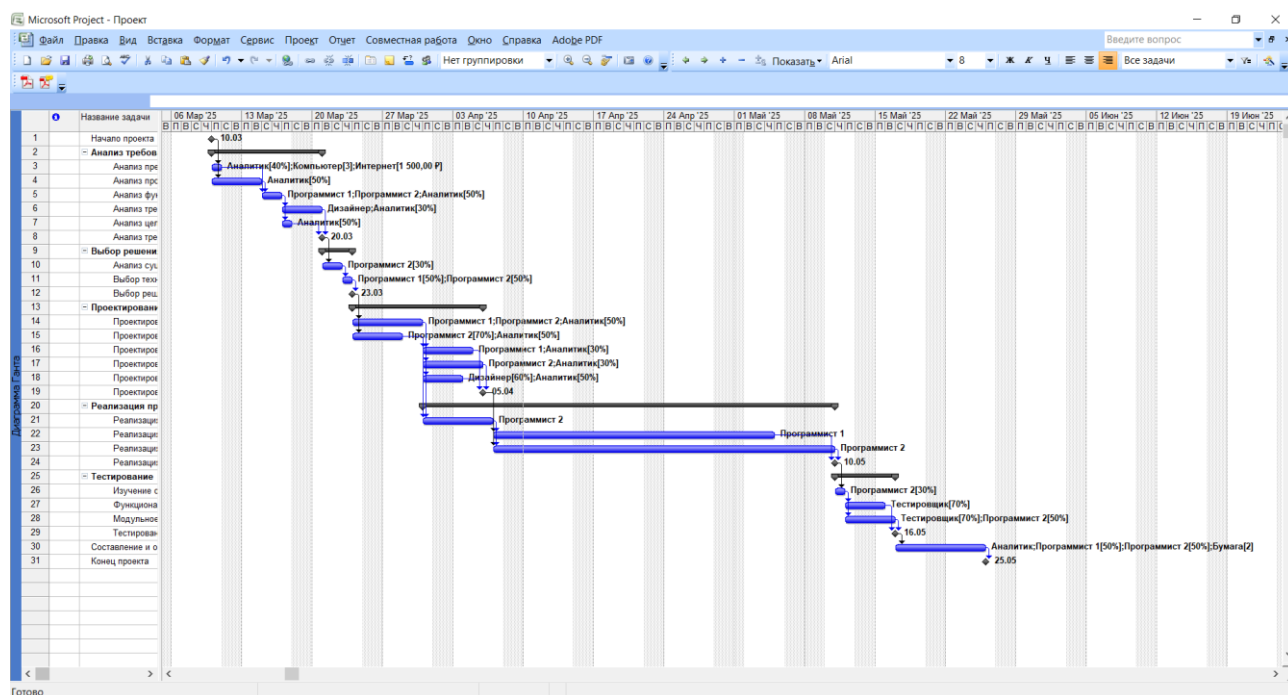


Рисунок 13 – Диаграмма Ганта

График загрузки ресурса «Программист 1» представлен на рисунке 14.

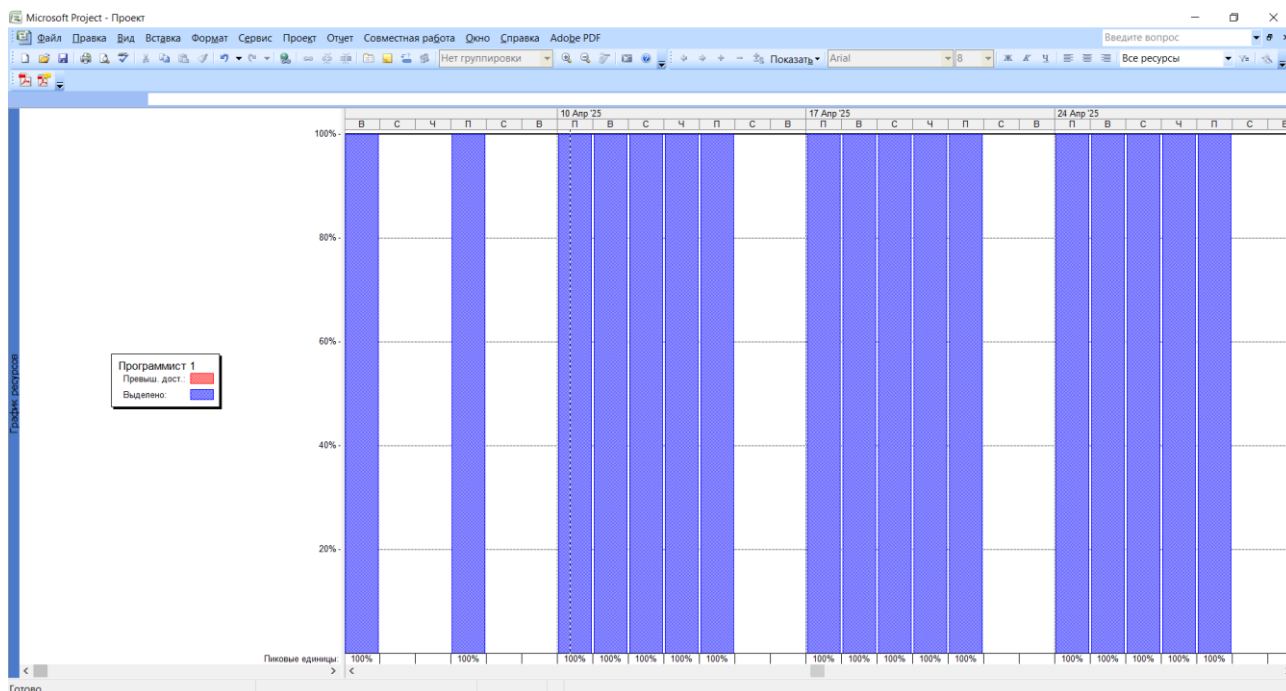


Рисунок 14 – График загрузки ресурса «Программист 1»

График загрузки ресурса «Программист 2» представлен на рисунке 15.

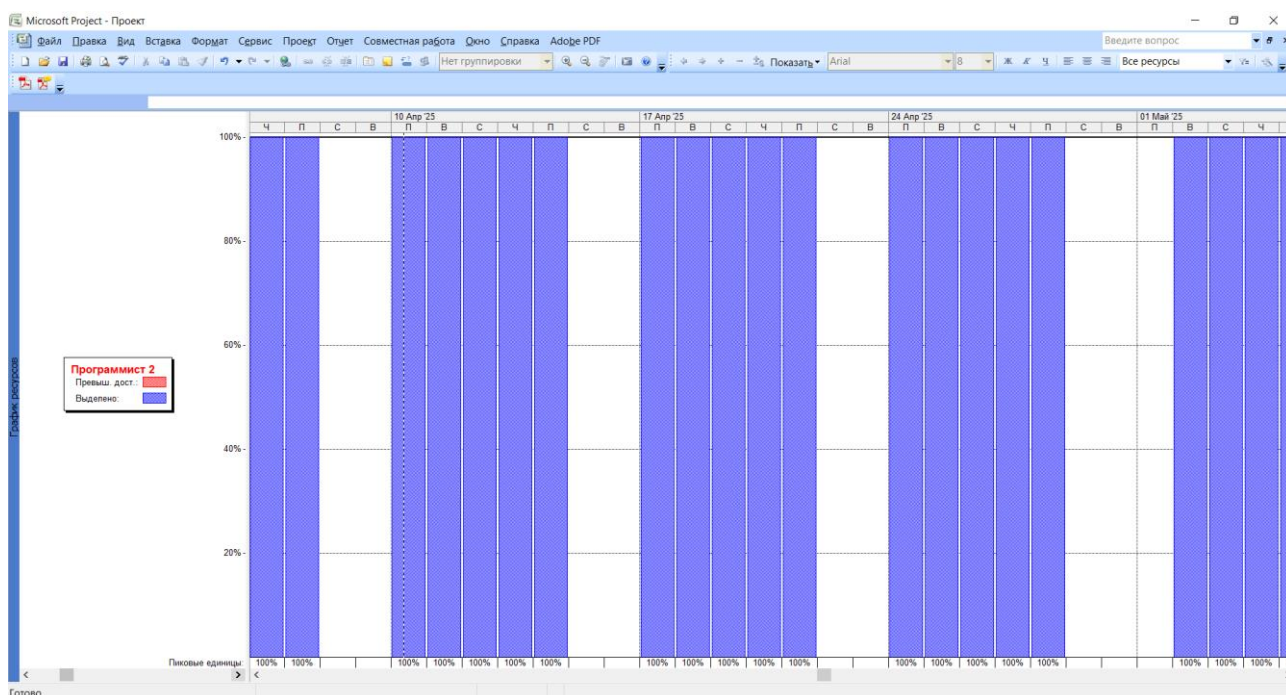


Рисунок 15 – График загрузки ресурса «Программист 2»

График загрузки ресурса «Аналитик» представлен на рисунке 16.

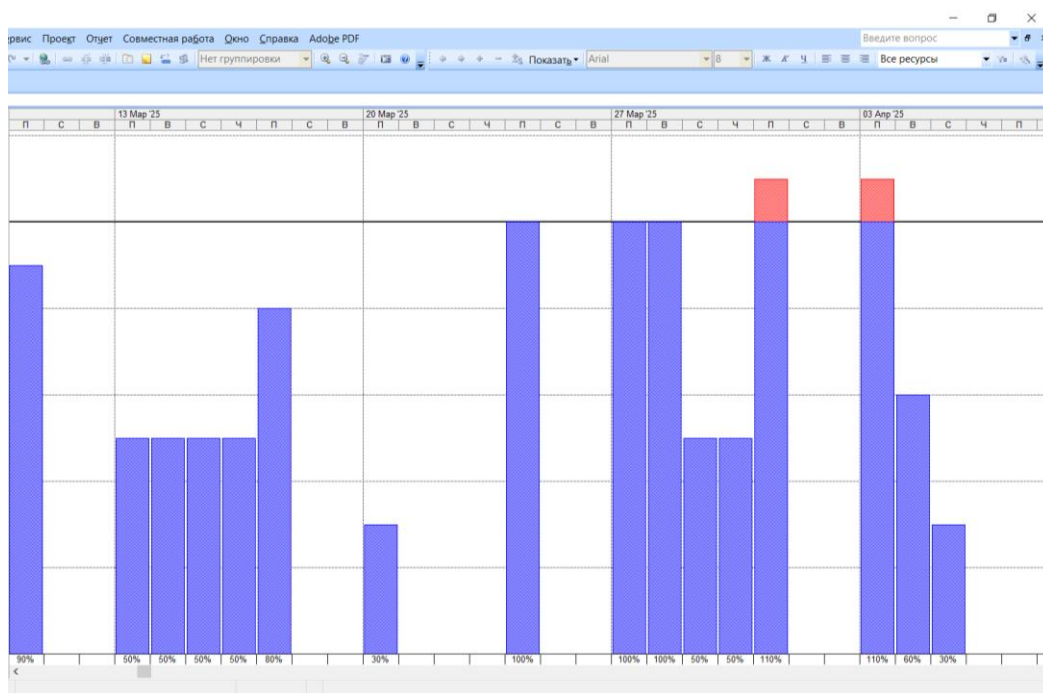


Рисунок 16 – График загрузки ресурса «Аналитик»

График загрузки ресурса «Дизайнер» представлен на рисунке 17.

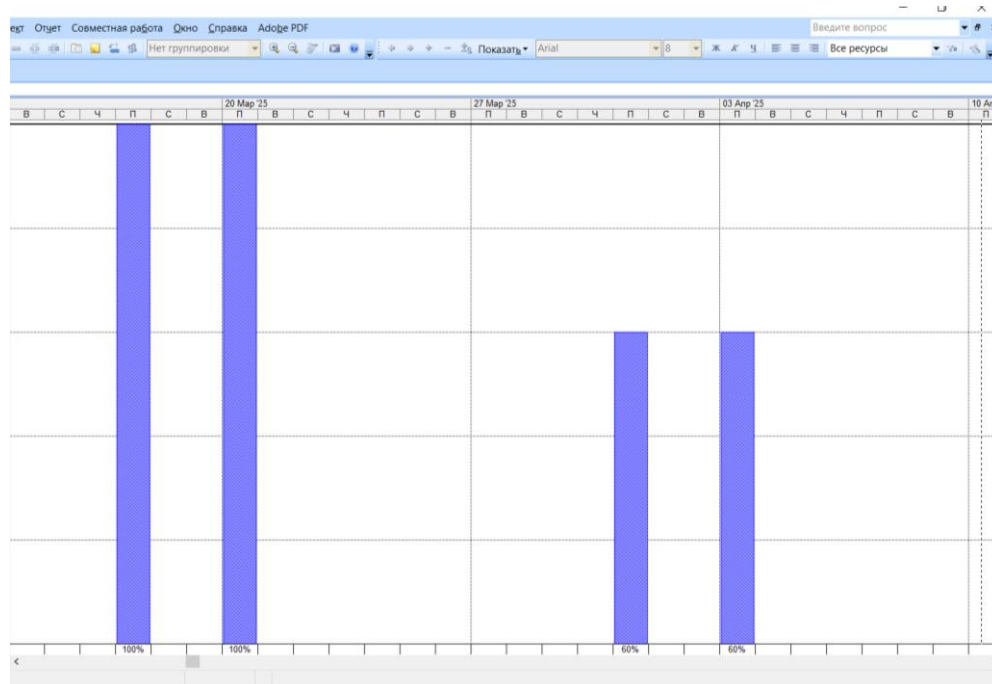


Рисунок 17 – График загрузки ресурса «Дизайнер»

График загрузки ресурса «Тестировщик» представлен на рисунке 18.

На рисунке 19 представлен обзорный отчет проекта, составленный системой Microsoft Project. Из рисунка видно, что общая длительность проекта составляет 52 дня (совпадает со значением, вычисленным вручную), а предполагаемые затраты – 303760 рублей.

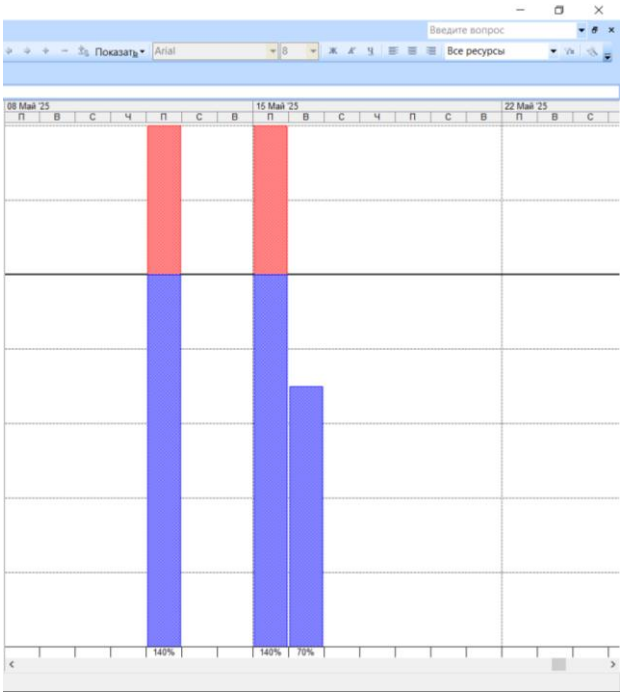


Рисунок 18 – График загрузки ресурса «Тестировщик»

Даты			
Начало:	Пт 10.03.25	Окончание:	Чт 25.05.25
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начало:	0 дней	Отклон. окончание:	0 дней
Длительность			
Текущий план:	52 дней	Оставшаяся:	52 дней
Базовый план:	0 дней?	Фактически:	0 дней
Отклонение:	52 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	826,4 ч	Оставшиеся:	826,4 ч
Базовый план:	0 ч	Фактически:	0 ч
Отклонение:	826,4 ч	Процент завершения:	0%
Затраты			
Текущий план:	303 760,00 Р	Оставшиеся:	303 760,00 Р
Базовый план:	0,00 Р	Фактически:	0,00 Р
Отклонение:	303 760,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начаты задачи	31	Трудовые ресурсы:	3
Выполняющиеся задачи	0	Трудовые ресурсы с пер. дост.	3
Завершенные задачи	0	Материальные ресурсы:	2
Итого задач:	31	Итого ресурсов:	8

Рисунок 19 – Обзорный отчет проекта

На основе произведенных оценок стоимости и сроков проекта можно сделать вывод о его экономической целесообразности.

1.7 Постановка задачи и предполагаемое решение

Собрать компьютер под собственные нужды – задача, с которой сталкиваются как обычные пользователи, так и энтузиасты. Однако даже базовая сборка требует понимания совместимости комплектующих, а также внимания к стоимости и техническим характеристикам. Покупатели нередко теряются в большом количестве деталей и опций, что может привести к ошибкам или неоптимальному выбору.

Разрабатываемый web-конфигуратор призван решить эту проблему. Его основная задача – помочь пользователю быстро и удобно собрать совместимую конфигурацию ПК, получить расчёт стоимости, сохранить или экспортировать сборку, а при необходимости – просмотреть рекомендованные варианты. Система автоматически проверяет компоненты на совместимость, снижая риск ошибок и упрощая процесс выбора.

Таким образом, результатом проекта станет доступный и интуитивно понятный инструмент, который облегчает процесс подбора комплектующих и делает сборку компьютера доступной даже для неопытных пользователей.

2 Проектирование программного обеспечения

2.1 Выбор технологий и инструментальных средств разработки

В результате проектирования веб-приложения для конфигурации персонального компьютера было принято решение использовать одну из самых известных методологий разработки – Rational Unified Process (RUP). Эта методология описывает общий подход, которым должна руководствоваться команда разработчиков для создания качественного программного продукта. Одним из основных принципов RUP является итеративная разработка, при которой весь процесс делится на четыре этапа, каждый из которых состоит из одной или нескольких итераций продолжительностью от двух до шести недель:

- начальная стадия – определение целей проекта, сбор и анализ требований, обоснование целесообразности разработки;
- стадия уточнения – детализация требований, проектирование архитектуры, обновление оценки рисков;
- стадия построения – реализация ключевой функциональности веб-приложения;
- стадия внедрения – завершение разработки, тестирование, публикация и развёртывание приложения.

Выбор технологии RUP обоснован её гибкостью: в случае изменения требований или выявления ошибок можно оперативно внести коррективы. Также одним из преимуществ является ранняя оценка рисков, что позволяет завершить проект на начальной стадии при наличии существенных проблем. В отличие от методологии eXtreme Programming (XP), постоянное взаимодействие с заказчиком не является обязательным на всех этапах.

Для анализа требований была использована методика прецедентов использования (use case), что обуславливает применение диаграмм UML. Визуальное моделирование проводилось с использованием инструмента ArgoUML, в котором были построены диаграммы прецедентов, активности и классов.

В качестве системы управления базами данных была выбрана PostgreSQL, как одна из самых популярных и надёжных СУБД с открытым исходным кодом. Серверная часть веб-приложения реализована на языке Python с использованием фреймворка FastAPI, который обеспечивает высокую производительность, поддержку асинхронных запросов и удобное построение REST API. Взаимодействие с базой данных осуществляется через библиотеку SQLAlchemy, реализующую объектно-реляционное отображение (ORM), позволяющее работать с данными как с Python-объектами.

Клиентская часть приложения разработана с использованием Next.js современной JavaScript-библиотеки для построения пользовательских интерфейсов. Next.js обеспечивает высокую производительность, модульность и поддержку компонентного подхода, что делает его оптимальным выбором для реализации интерактивного и удобного интерфейса web-конфигуратора.

2.2 Проектирование архитектуры программного обеспечения

В данной работе было принято решение использовать трехуровневую архитектуру «клиент-сервер». Ее схема представлена на рисунке 20.

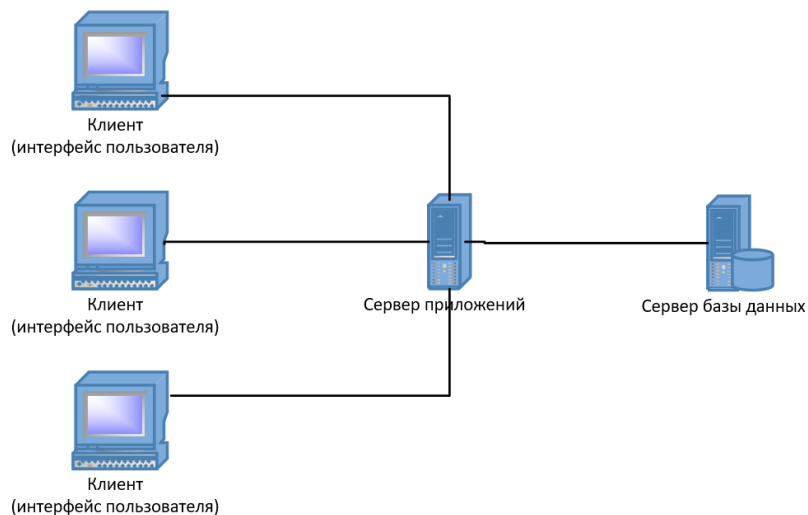


Рисунок 20 – Трехуровневая архитектура «клиент-сервер»

В данной архитектуре на сервере базы данных осуществляется управлением данными. Сервер приложений реализует функционал всего проекта и отвечает за взаимодействие с базой данных. Клиентская же часть, в

первую очередь, обеспечивает интерфейс, позволяющий просматривать информацию, полученную с сервера приложений, и взаимодействовать с данными путем удаленного обращения к серверу. В данной работе в качестве клиентов будут выступать персональные компьютеры имеющие веб-браузеры.

Основным преимуществом трехуровневой архитектуры «клиент-сервер» является то, что каждый уровень в ней независим и может разрабатываться параллельно с разработкой других уровней. Также клиент в данной архитектуре не работает напрямую с базой данных, в связи с чем повышается безопасность данных.

2.3 Проектирование базы данных

2.3.1 Концептуальное проектирование базы данных

На основе анализа предметной области была построена концептуальная модель базы данных. Графическое представление концептуальной модели БД в виде диаграммы «сущность-связь» представлено на рисунке 21.

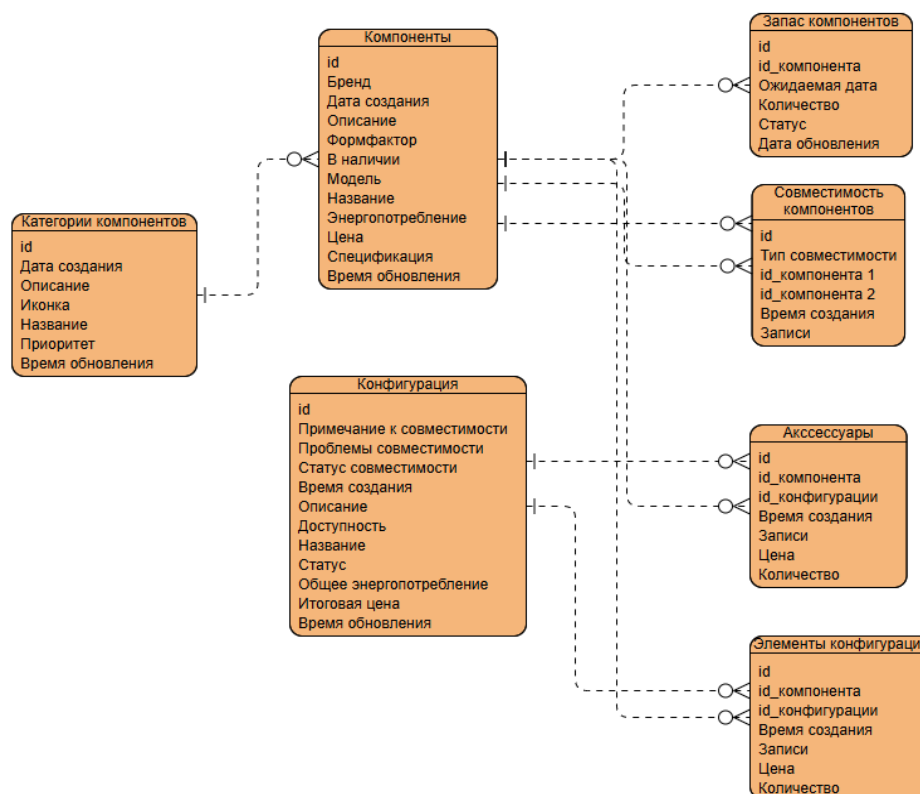


Рисунок 21 – Графическое представление концептуальной модели базы данных

Сущности концептуальной модели данных представлены в таблице 8.

Таблица 8 – Сущности концептуальной модели

№	Название	Описание	Количество экземпляров	Режим обновления
1	Категории компонентов	Содержит информацию о категориях компонентов (иконка, приоритет и т. д.)	1 таблица	По мере добавления категорий
2	Компоненты	Описывает характеристики компонентов (бренд, модель, цена и т. д.)	Множество записей	При изменении ассортимента
3	Запас компонентов	Учитывает наличие и статус компонентов на складе	Зависит от компонентов	При изменении количества
4	Совместимость компонентов	Определяет совместимость между разными компонентами	Зависит от связей	При добавлении новых правил
5	Конфигурация	Описывает сборки/конфигурации из компонентов	Множество вариантов	При создании/изменении сборок
6	Аксессуары	Содержит дополнительные элементы для конфигураций	Зависит от комплектов	При обновлении ассортимента
7	Элементы конфигураций	Связывает компоненты с конкретными конфигурациями	Зависит от сборок	При изменении состава

В таблице 9 представлено описание связей между сущностями.

Таблица 9 – Описание связей между сущностями концептуальной модели

№	Связь	Описание	Мощность	Обязательность
1	Категории компонентов → Компоненты	Категория может содержать множество компонентов	Один-ко-многим	Обязательная
2	Компоненты → Запас компонентов	Каждый компонент может иметь несколько записей о запасе	Один-ко-многим	Необязательная
3	Компоненты → Совместимость компонентов	Компонент может быть связан с другими компонентами через совместимость	Многие-ко-многим	Обязательная
4	Компоненты → Элементы конфигураций	Компонент может входить в несколько конфигураций	Многие-ко-многим	Обязательная
5	Компоненты → Аксессуары	Компонент может быть связан с аксессуарами	Один-ко-многим	Необязательная

Продолжение таблицы 9

№	Связь	Описание	Мощность	Обязательность
6	Конфигурация → Элементы конфигураций	Конфигурация включает в себя несколько компонентов	Один-ко- многим	Обязательная
7	Конфигурация → Аксессуары	Конфигурация может включать в себя аксессуары	Один-ко- многим	Необязательная

Таким образом, были спроектированы основные сущности с атрибутами и связями между ними, без включения атрибутов первичных и внешних ключей.

2.3.2 Логическое проектирование базы данных

На основании концептуальной модели была спроектирована логическая модель базы данных, представленная на рисунке 22.

На данной логической модели структура базы данных отображена уже более подробно. Здесь определены первичные и внешние ключи, ограничения атрибутов сущностей, а также уточнен их состав и связи между ними.

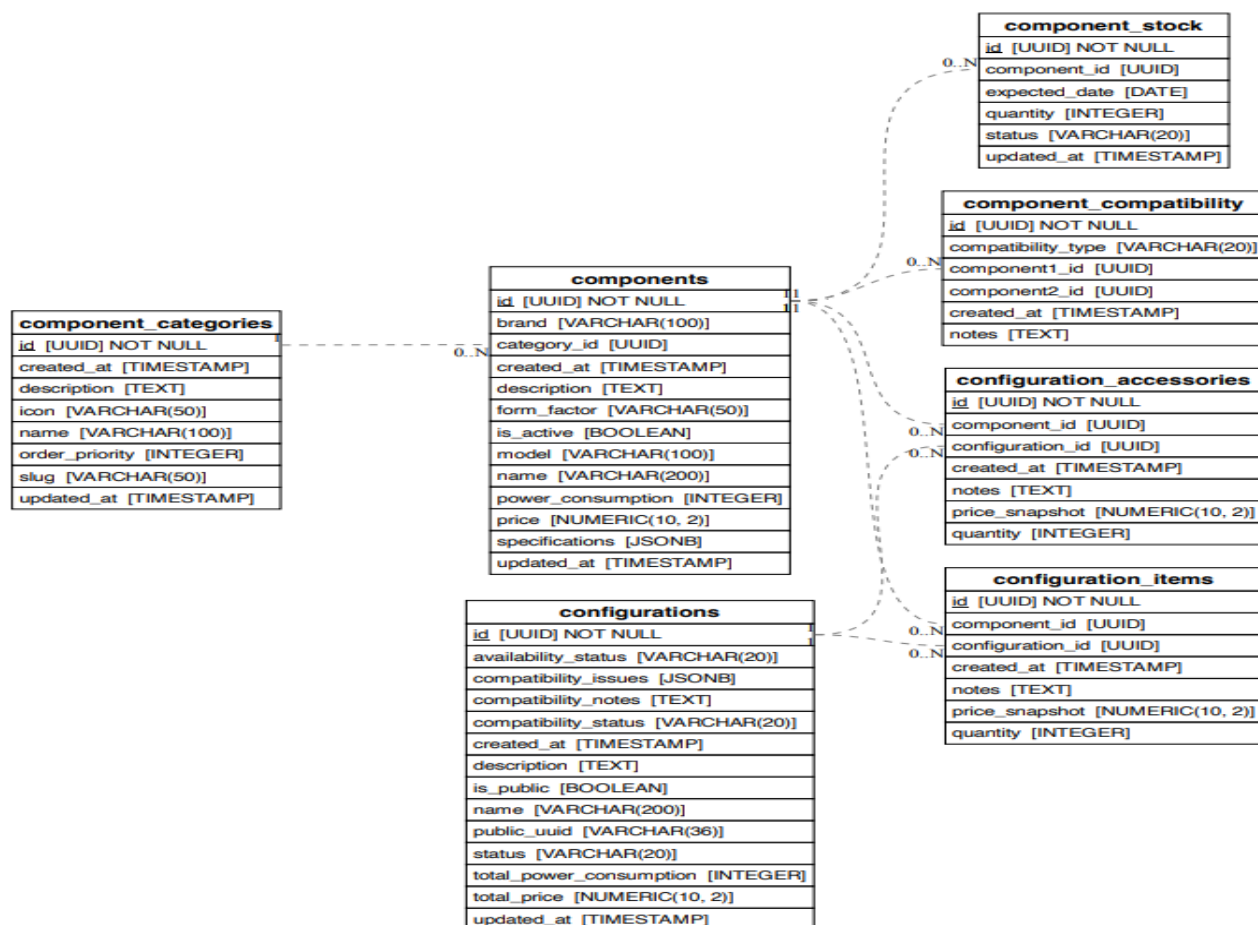


Рисунок 22 – Графическое представление логической модели базы данных

Сущности логической модели данных, не определенные в концептуальной модели, представлены в таблице 10.

Таблица 10 – Сущности логической модели

№	Название	Атрибуты	Количество экземпляров (примерно)
1	component_categories	id, created_at, description, icon, name, order_priority, slug, updated_at	10–50 (категории комплектующих)
2	components	id, brand, category_id, created_at, description, form_factor, is_active, model, name, power_consumption, price, specifications, updated_at	100–10,000 (разные комплектующие)
3	configurations	id, availability_status, compatibility_issues, compatibility_notes, compatibility_status, created_at, description, is_public, name, public_uuid, status, total_power_consumption, total_price, updated_at	100–1,000 (готовые сборки)
4	component_stock	id, component_id, expected_date, quantity, status, updated_at	500–5,000 (записи о запасах)
5	component_compatibility	id, compatibility_type, component_id (2x), created_at, notes	1,000–50,000 (связи совместимости)
6	configuration_accessories	id, component_id, configuration_id, created_at, notes, price_snapshot, quantity	200–2,000 (аксессуары в сборках)
7	configuration_items	id, component_id, configuration_id, created_at, notes, price_snapshot, quantity	1,000–10,000 (компоненты в сборках)

Также в таблице 11 представлено описание связей между сущностями логической модели, не определенных в результате концептуального моделирования.

Несмотря на достаточную содержательность данной модели, сущности все еще представлены в абстрактной форме, без привязки к конкретной реализации базы данных.

2.4 Проектирование серверной части приложения

Серверная часть приложения выполнена на основе архитектуры REST API. В данном случае взаимодействие с сервером осуществляется по протоколу HTTP при помощи стандартных методов, таких как GET (получение ресурса), PUT (обновление существующего ресурса), POST (создание нового ресурса) и DELETE (удаление ресурса). При этом обращение к ресурсам сервера происходит по их уникальным идентификаторам, называемым URI.

Исходя из выбранных технологий разработки серверной части приложения, для ее реализации было решено использовать шаблон MVC (Model-View-Controller). Схема данного шаблона приведена на рисунке 23.

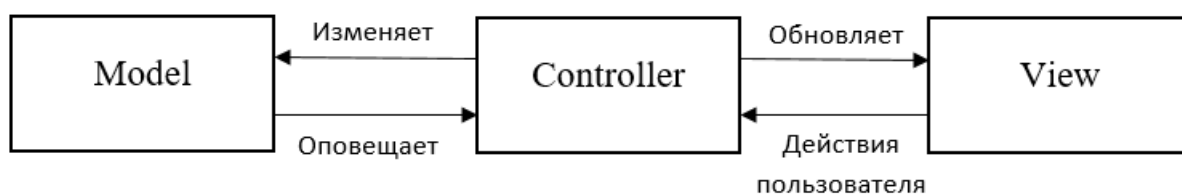


Рисунок 23 – Схема шаблона MVC

Как видно из данной схемы, в системе имеется 3 компонента: модель, представление и контроллер. Классы уровня модели отвечают за организацию доступа к данным (за непосредственное взаимодействие с базой данных). Классы контроллера отвечают за организацию взаимодействия пользователя с клиентом. Они обрабатывают HTTP-запросы клиента и формируют ответы на них. Уровень представления отвечает за взаимодействие с пользователем.

Выбор данного шаблона обусловлен использованием в работе технологии ASP.NET, которая нацелена на применение данного шаблона.

2.5 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был спроектирован пользовательский интерфейс веб-приложения для конфигурирования персонального компьютера. Интерфейс включает 5 основных вкладок, каждая из которых обеспечивает доступ к ключевым функциям приложения:

- «Конфигуратор» – основная страница, предназначенная для сборки пользовательской конфигурации ПК путём выбора доступных компонентов и аксессуаров;
- «Компоненты» – предоставляет список основных аппаратных частей (процессоры, материнские платы, видеокарты и др.), доступных для выбора, редактирования или добавления в сборку;
- «Аксессуары» – отображает дополнительные устройства и периферийное оборудование, такие как клавиатуры, наушники, веб-камеры и мониторы;
- «Импорт» – вкладка для загрузки ранее сохранённых конфигураций из файлов или баз данных, а также для редактирования импортированных сборок;
- «О программе» – содержит справочную информацию о приложении, его назначении, версии и авторах проекта.

Так же есть вкладка оформления заказа, возникающая при нажатии на кнопку «Перейти к покупке» на странице «Конфигуратор».

Для визуализации структуры интерфейса и взаимодействия с пользователем был разработан прототип интерфейса. Он включает макет страниц и основные их компоненты, что позволило протестировать пользовательский опыт до начала реализации и обеспечить согласованность с функциональными требованиями.

Для разработки прототипа использовался современный инструмент проектирования Figma, что позволило создать интерактивные макеты с возможностью навигации между экранами.

Дизайн интерфейса выполнен в соответствии с принципами удобства использования, обеспечивая интуитивную понятность и комфортное взаимодействие для пользователей.

Прототип интерфейса страницы «Конфигуратор» представлен на рисунке 24.

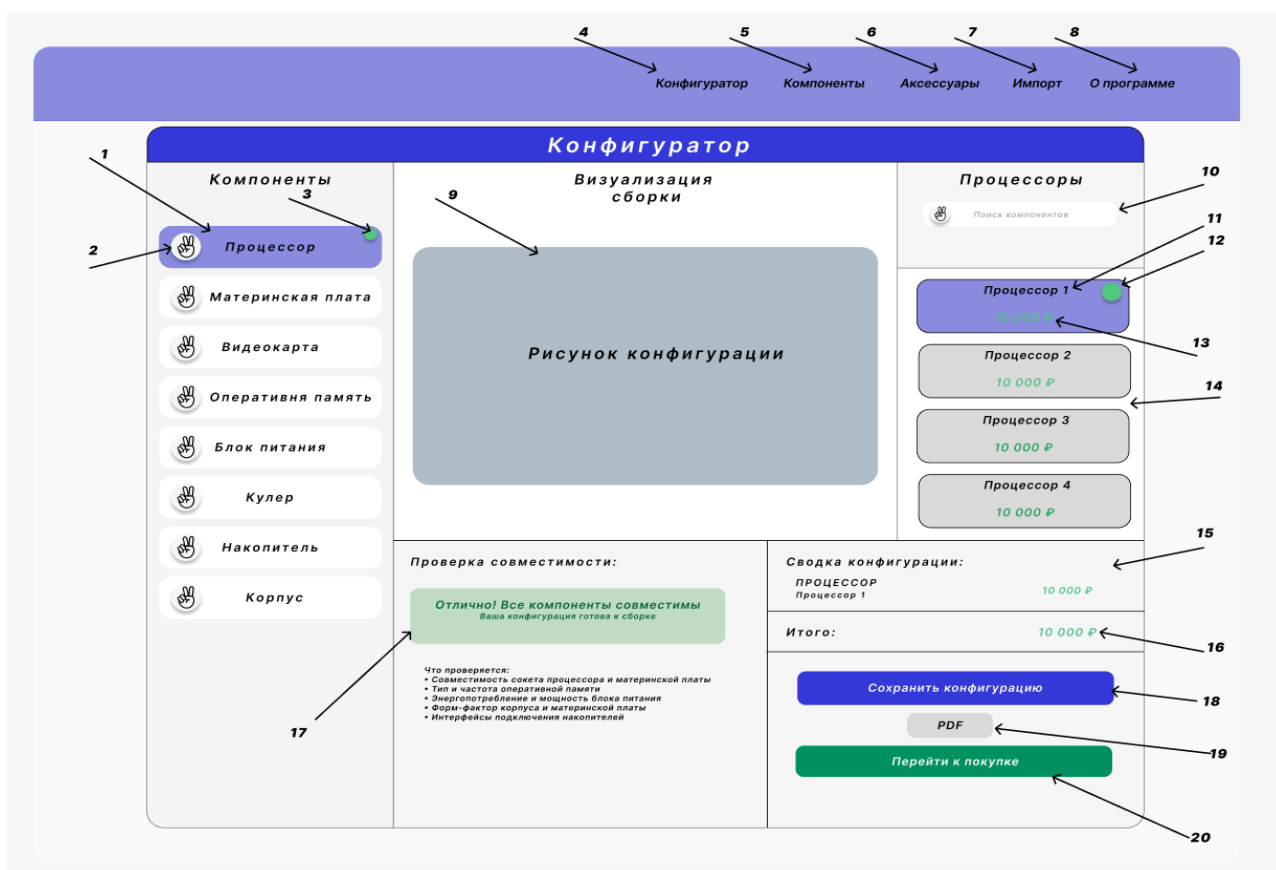


Рисунок 24 – Страница «Конфигуратор»

На интерфейсе страницы «Конфигуратор» имеются следующие элементы:

- 1 – кнопка выбора детали компьютера;
- 2 – иконка детали компьютера;
- 3 – индикатор выбора детали;
- 4 – кнопка перехода на страницу «конфигуратора»;
- 5 – кнопка перехода на страницу «компоненты»;
- 6 – кнопка перехода на страницу «аксессуары»;
- 7 – кнопка перехода на страницу «импорт»;
- 8 – кнопка перехода на страницу «о программе»;
- 9 – визуальное отображение конфигурации;
- 10 – поисковая строка компонентов;
- 11 – название процессора;
- 12 – индикатор выбора процессора;
- 13 – стоимость процессора;

- 14 – список процессоров;
- 15 – окно сводки конфигурации;
- 16 – итоговая стоимость;
- 17 – сообщение с результатом проверки совместимости;
- 18 – кнопка сохранения конфигурации;
- 19 – кнопка экспорта конфигурации в pdf;
- 20 – кнопка перехода к покупке.

Прототип интерфейса страницы «Компоненты» представлен на рисунке 25.

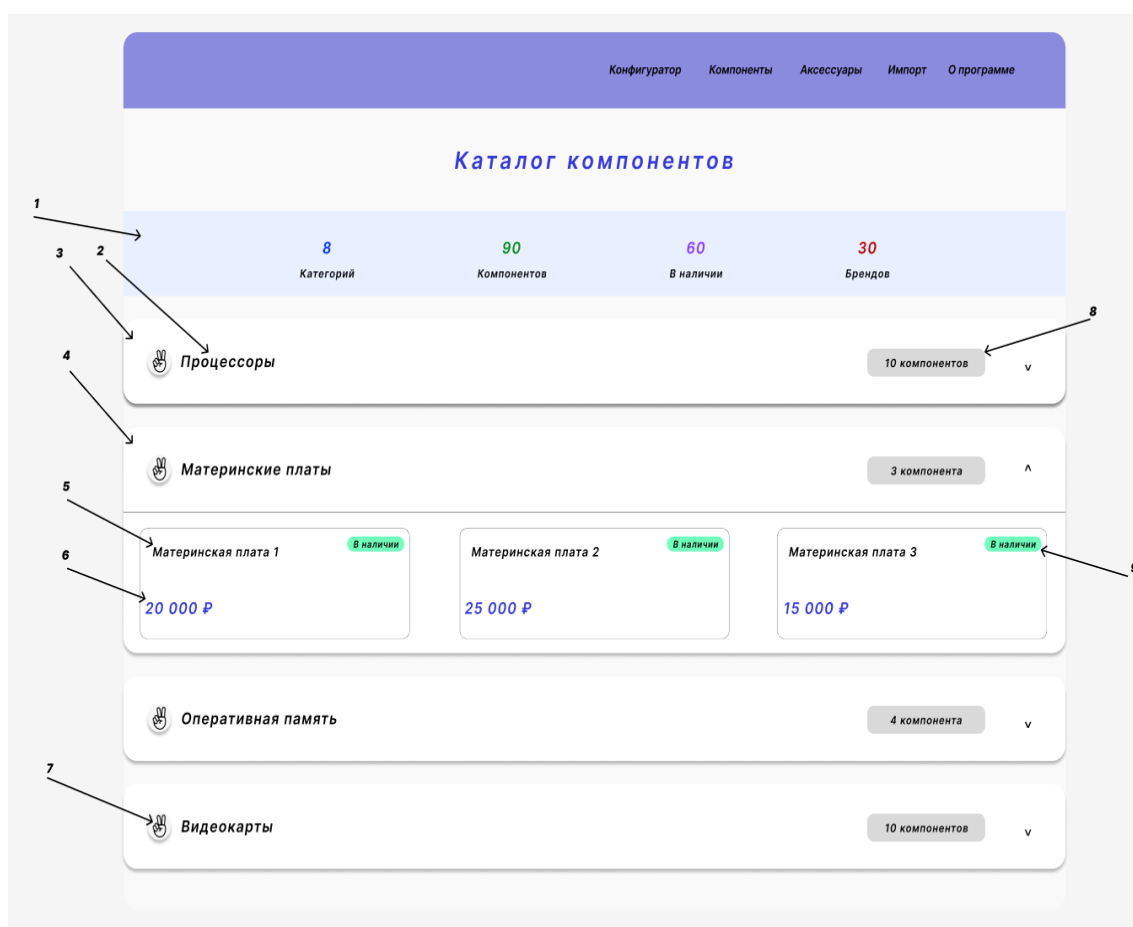


Рисунок 25 – Страница «Компоненты»

На интерфейсе страницы «Компоненты» имеются следующие элементы:

- 1 – информация о комплектующих;
- 2 – название категории компонентов;
- 3 – вкладка категории компонентов;
- 4 – раскрытая вкладка компонентов;

5 – название компонента;

6 – цена компонента;

7 – иконка категории компонентов;

8 – количество компонентов;

9 – индикатор, показывающий есть ли в наличии компонент.

Прототип интерфейса страницы «Аксессуары» представлен на рисунке 26.

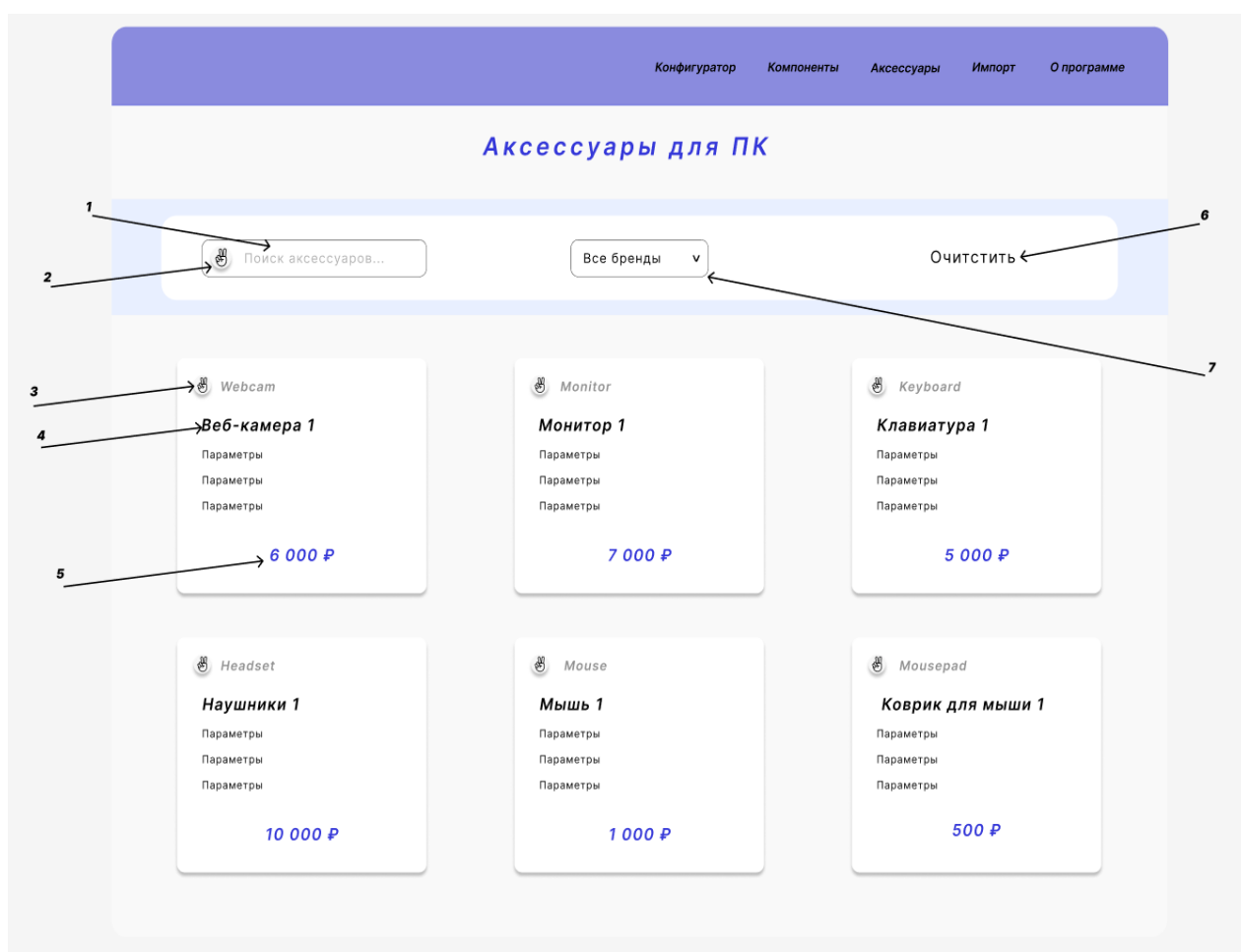


Рисунок 26 – Страница «Аксессуары»

На интерфейсе страницы «Аксессуары» имеются следующие элементы:

1 – поисковая строка аксессуаров;

2 – иконка аксессуара;

3 – иконка типа аксессуара;

4 – название аксессуара;

5 – цена аксессуара;

6 – выпадающий список фильтрации;

7 – кнопка, позволяющая очистить ранее введенные данные.

Прототип интерфейса страницы «Импорт» представлен на рисунке 27.

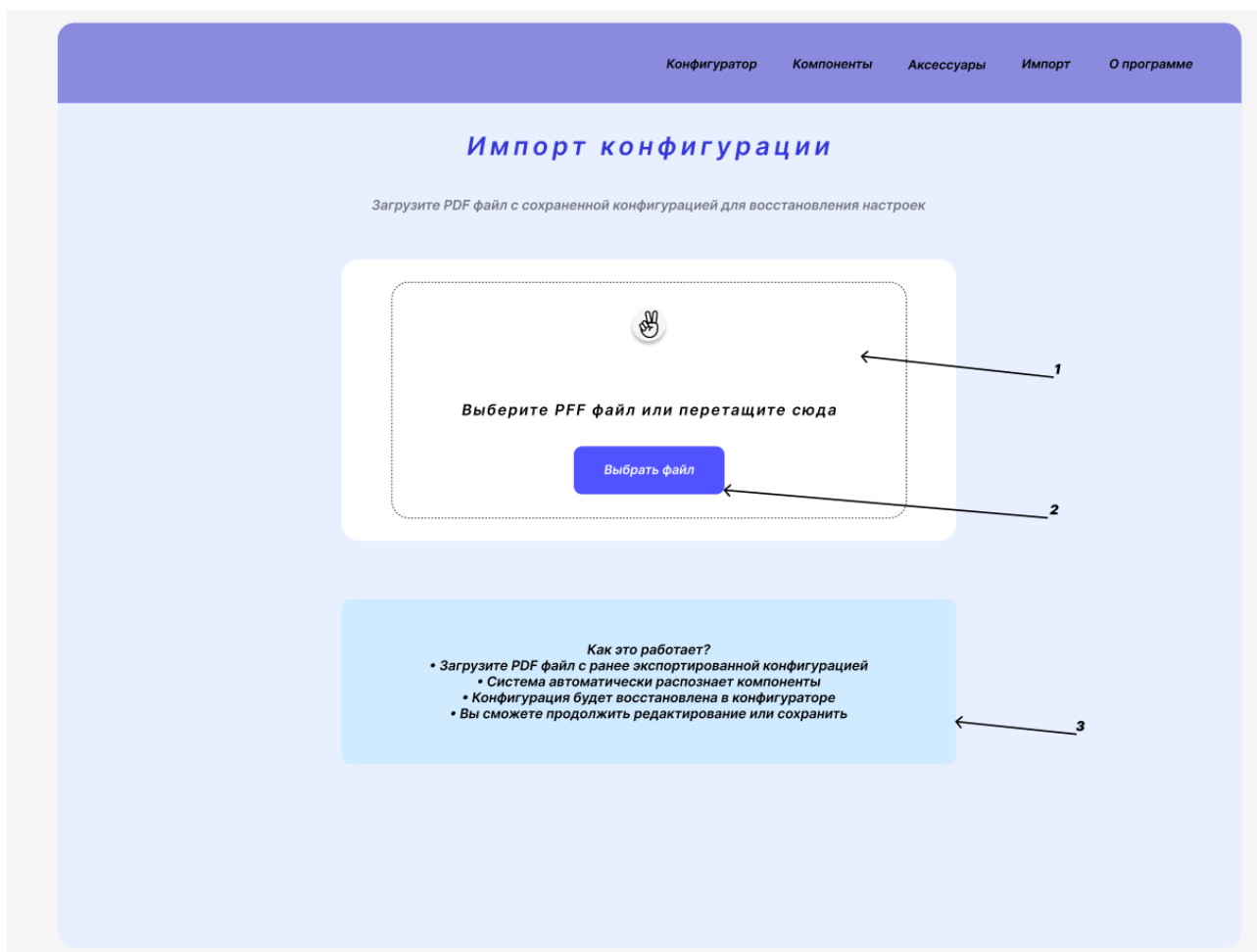


Рисунок 27 – Страница «Импорт»

На интерфейсе страницы «Аксессуары» имеются следующие элементы:

1 – поле куда можно перетащить файл;

2 – кнопка для выбора файла с компьютера;

3 – справка.

Страница «О проекте» планируется статической и должна отображать общую информацию о web-конфигураторе. Для удобства пользователей информация будет представлена в лаконичном и структурированном виде, с использованием заголовков и иконок для визуального выделения ключевых особенностей. Прототип интерфейса страницы «О проекте» представлен на рисунке 28.

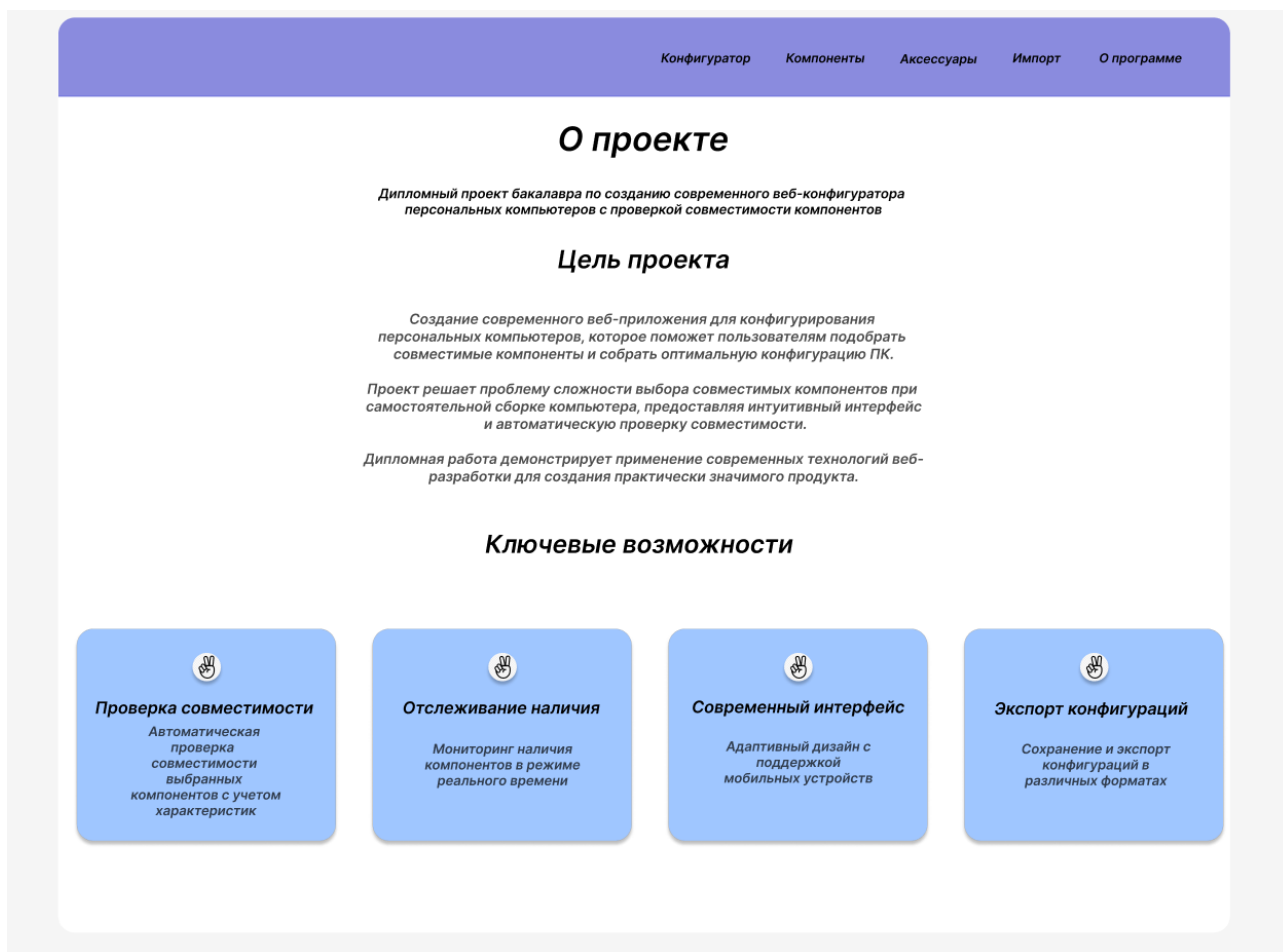


Рисунок 28 – Страница «О проекте»

На интерфейсе страницы «О проекте» отображается общая информация о web-конфигураторе

Макет интерфейса страницы оформления заказа представлен на рисунке 29.

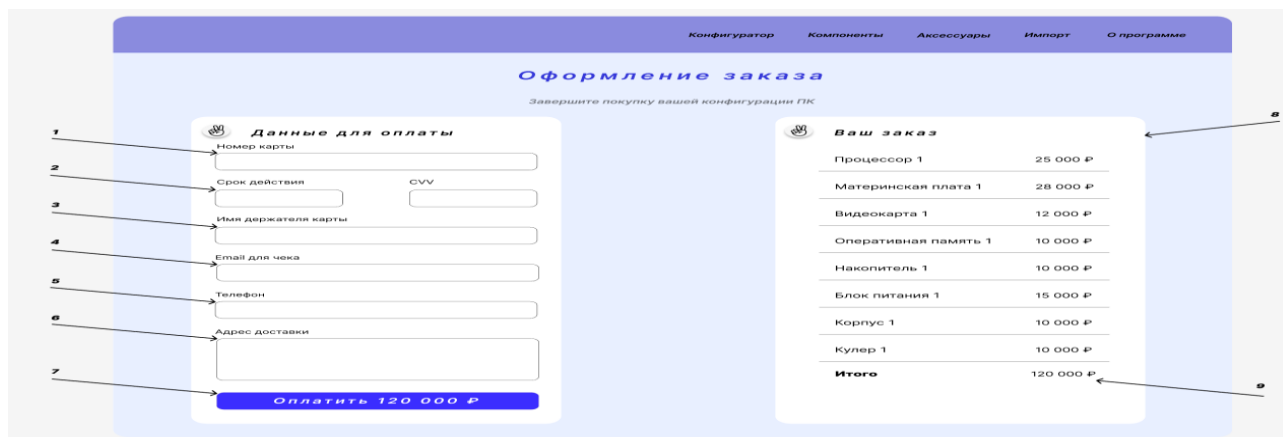


Рисунок 29 – Страница «Оформление заказа»

На макете страницы добавления нового пассажира имеются следующие элементы:

- 1 – поле для ввода номера карты;
- 2 – поле для ввода срока действия;
- 3 – поле для ввода cvv;
- 4 – поле для ввода имени держателя карты;
- 5 – поле для ввода email;
- 6 – поле для ввода номера телефона;
- 7 – поле для ввода адреса доставки;
- 8 – кнопка для оплаты;
- 9 – выбранная конфигурация;
- 10 – итоговая стоимость.

Таким образом, в рамках проектирования пользовательского интерфейса было разработано удобное и интуитивно понятное представление основных функций приложения. Структура интерфейса охватывает все ключевые аспекты работы пользователя – от выбора компонентов и аксессуаров до импорта конфигураций и получения справочной информации. Разработка прототипа в Figma позволила заранее визуализировать логику взаимодействия, учесть возможные сценарии использования и обеспечить удобство при дальнейшей реализации проекта.

3 Реализация программного обеспечения

3.1 Реализация базы данных

На основании концептуальной и логической модели для была разработана физическая модель базы данных [10, 11] (рисунок 30).

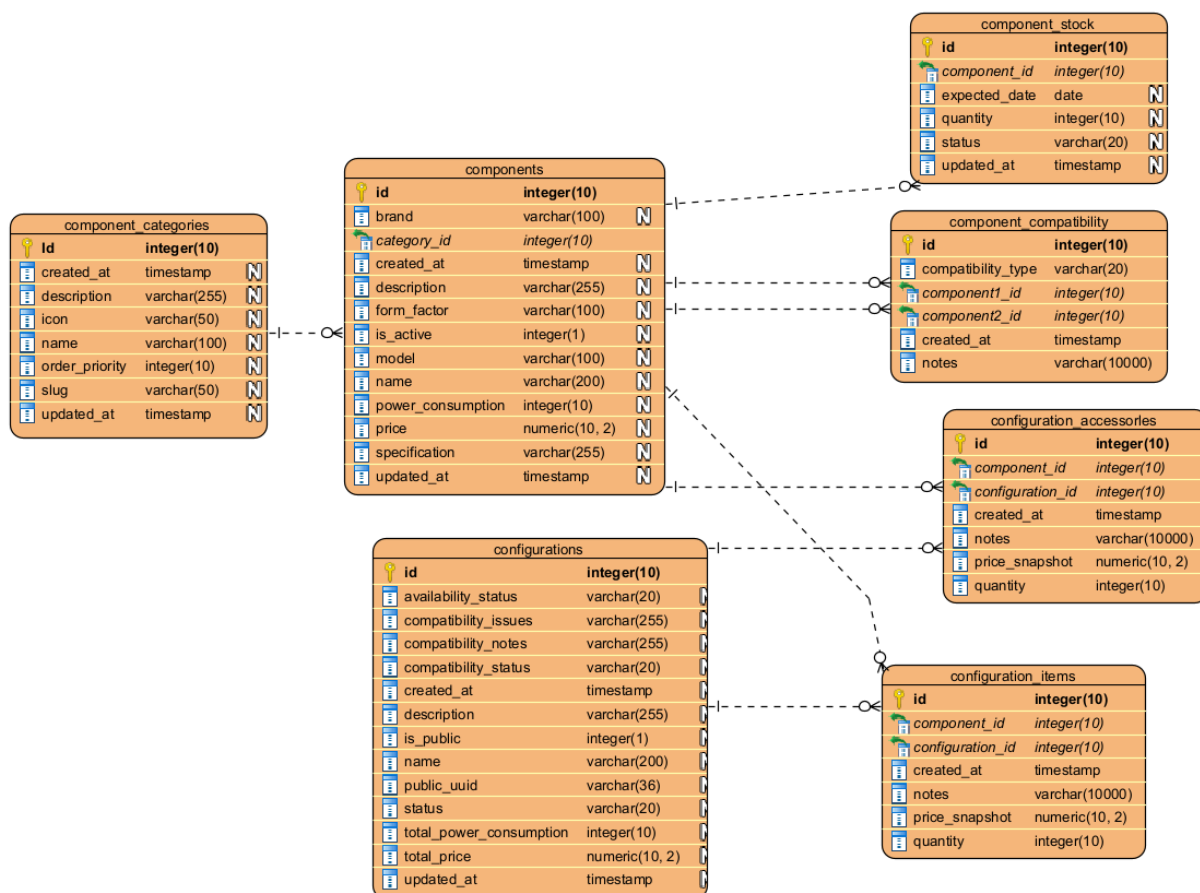


Рисунок 30 – Графическое представление физической модели базы данных

Описание атрибутов сущности «component_categories» представлено в таблице 11.

Таблица 11 – Описание атрибутов сущности «component_categories»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id	Идентификатор	integer(10)	Нет	Да	PK
created_at	Дата создания	timestamp	Нет	Да	-
description	Описание	varchar(255)	Да	Да	-
icon	Иконка категории	varchar(50)	Да	Нет	-

Продолжение таблицы 11

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
name	Название	varchar(100)	Нет	Да	-
order_priority	Приоритет	integer(10)	Нет	Нет	-
slug	Ссылка	varchar(50)	Нет	Нет	-
updated_at	Дата обновления	timestamp	Да	Нет	-

Описание атрибутов сущности «components» представлено в таблице 12.

Таблица 12 – Описание атрибутов сущности «components »

Наименование	Описание	Тип данных	NULL	Unique	Роль
id	Уникальный идентификатор компонента	integer	Нет	Да	PK
brand	Производитель компонента	varchar(100)	Нет	Нет	-
category_id	ID категории	integer(10)	Нет	Нет	FK
created_at	Дата и время создания записи	timestamp	Нет	Нет	-
description	Подробное описание компонента	varchar(255)	Нет	Нет	-
form_factor	Стандарт/размер компонента	varchar(100)	Нет	Нет	-
is_active	Флаг доступности для выбора	integer(1)	Да	Нет	-
model	Модельное название компонента	varchar(100)	Нет	Нет	-
name	Название для отображения	varchar(200)	Нет	Нет	-
power_consumption	Потребляемая мощность (Вт)	integer(10)	Нет	Нет	-
price	Цена компонента	numeric(10,2)	Нет	Нет	-
specification	Ключевые характеристики	varchar(255)	Нет	Нет	-
updated_at	Дата и время обновления записи	timestamp	Нет	Нет	-

Описание атрибутов сущности «configurations» представлено в таблице 13.

Таблица 13 – Описание атрибутов сущности «configurations»

id	Уникальный идентификатор сборки	integer(10)	Нет	Да	PK
availability_status	Статус доступности сборки	varchar(20)	Нет	Нет	-
compatibility_issues	Ошибки совместимости компонентов	varchar(255)	Да	Нет	-
compatibility_notes	Примечания по совместимости	varchar(255)	Да	Нет	-
compatibility_status	Общий статус совместимости	varchar(20)	Нет	Нет	-
created_at	Дата и время создания сборки	timestamp	Нет	Нет	-
description	Описание сборки	varchar(255)	Да	Нет	-
is_public	Флаг публичности сборки	integer(1)	Нет	Нет	-
name	Название сборки	varchar(200)	Нет	Нет	-
public_uuid	Уникальный UUID для публичных ссылок	varchar(36)	Да	Да	-
status	Текущий статус сборки	varchar(20)	Нет	Нет	-
total_power_consumption	Суммарное энергопотребление (Вт)	integer(10)	Нет	Нет	-
total_price	Итоговая стоимость сборки	numeric(10,2)	Нет	Нет	-
updated_at	Дата и время обновления сборки	timestamp	Нет	Нет	-

Описание атрибутов сущности «Город» представлено в таблице 14.

Таблица 14 – Описание атрибутов сущности «component_stock»

Наименование	Описание	Тип данных	NULL	Unique	Роль
id	Уникальный идентификатор записи	integer(10)	Нет	Да	PK

Продолжение таблицы 14

Наименование	Описание	Тип данных	NULL	Unique	Роль
component_id	Ссылка на компонент из таблицы components	integer(10)	Нет	Нет	FK
expected_date	Ожидаемая дата поставки	date	Да	Нет	-
quantity	Доступное количество на складе	integer(10)	Нет	Нет	-
status	Статус наличия (в наличии, под заказ и т. д.)	varchar(20)	Нет	Нет	-
updated_at	Дата и время обновления информации	timestamp	Нет	Нет	-

Описание атрибутов сущности «Аэропорт» представлено в таблице 15.

Таблица 15 – Описание атрибутов сущности «component_compatibility»

Наименование	Описание	Тип данных	NULL	Unique	Роль
id	Уникальный идентификатор связи	integer(10)	Нет	Да	PK
compatibility_type	Тип совместимости	varchar(20)	Нет	Нет	-
component1_id	ID первого компонента	integer(10)	Нет	Нет	FK (components)
component2_id	ID второго компонента	integer(10)	Нет	Нет	FK (components)
created_at	Дата создания записи	timestamp	Нет	Нет	-

Описание атрибутов сущности «Самолет» представлено в таблице 16.

Таблица 16 – Описание атрибутов сущности configuration_accessories»

Наименование	Описание	Тип данных	NULL	Unique	Роль
id	Уникальный идентификатор связи	integer(10)	Нет	Да	PK
component_id	ID компонента-аксессуара	integer(10)	Нет	Нет	FK (components)

Продолжение таблицы 16

Наименование	Описание	Тип данных	NULL	Unique	Роль
configuration_id	ID сборки	integer(10)	Нет	Нет	FK (configurations)
created_at	Дата добавления аксессуара	timestamp	Нет	Нет	-
notes	Примечания по аксессуару	varchar(10000)	Да	Нет	-
price_snapshot	Цена на момент добавления	numeric(10,2)	Нет	Нет	-
quantity	Количество аксессуаров	integer(10)	Нет	Нет	-

Описание атрибутов сущности «Маршрут» представлено в таблице 17.

Таблица 17 – Описание атрибутов сущности «configuration_items»

Наименование	Описание	Тип данных	NULL	Unique	Роль
id	Уникальный идентификатор записи	integer(10)	Нет	Да	PK
component_id	ID основного компонента	integer(10)	Нет	Нет	FK (components)
configuration_id	ID сборки	integer(10)	Нет	Нет	FK (configurations)
created_at	Дата добавления компонента	timestamp	Нет	Нет	-
notes	Примечания по компоненту	varchar(10000)	Да	Нет	-
price_snapshot	Цена на момент добавления	numeric(10,2)	Нет	Нет	-
quantity	Количество компонентов	integer(10)	Нет	Нет	-

На данной физической модели структура базы данных отображена с учетом спецификации типов СУБД Microsoft SQL Server 2019.

На основе физической модели БД был сгенерирован скрипт на языке SQL для создания базы данных. Его код представлен в приложении Б. Развертывание спроектированной базы данных было выполнено путем запуска данного скрипта в СУБД Microsoft SQL Server 2019.

3.2 Реализация серверной части приложения

Реализация серверной части приложения была выполнена на языке программирования Python с использованием фреймворка FastAPI и технологии RESTful API [12, 13]. В качестве инструмента для взаимодействия с базой данных применялась объектно-реляционная система отображения данных SQLAlchemy, обеспечивающая удобную работу с таблицами и связями между ними на уровне классов Python.

Архитектура приложения построена на принципах многослойного подхода. Все классы серверной части разделены на два основных слоя: слой моделей, отвечающий за описание структуры базы данных и взаимодействие с ней, и слой сервисов (контроллеров), реализующий бизнес-логику и обработку входящих HTTP-запросов.

Диаграмма классов слоя моделей представлена на рисунке 31. На ней отражены классы, соответствующие структурам таблиц в базе данных: Component, Configuration, ComponentCategory, ComponentStock, ConfigurationItem, ConfigurationExport, CompatibilityCheck и CompatibilityIssue. Эти классы описывают такие сущности, как компоненты ПК, пользовательские конфигурации, остатки на складе и результаты проверки совместимости.

Модель Component содержит информацию о конкретном комплектующем, включая его характеристики, категорию и состояние на складе. Конфигурации пользователей описываются с помощью модели Configuration, которая включает в себя список компонентов (ConfigurationItem) и связанные данные об экспортируемых версиях (ConfigurationExport) и совместимости (CompatibilityCheck, CompatibilityIssue). Такая структура позволяет систематизировать хранение и обработку информации о сборках, а также автоматизировать проверку их технической совместимости.

Диаграмма классов слоя контроллеров представлена на рисунке 32. На ней показаны основные сервисы, реализующие бизнес-логику приложения: ConfigurationService, ComponentService, CompatibilityService и PDFService. Каждый из этих классов инкапсулирует функции, связанные с определённой

задачей. Так, ConfigurationService управляет конфигурациями, добавляет и удаляет компоненты, обновляет данные о совместимости и рассчитывает итоговые параметры сборки. ComponentService отвечает за выборку компонентов из базы, фильтрацию и получение допустимых комбинаций. ComratibilityService выполняет проверку совместимости компонентов в конфигурации, а PDFService обеспечивает генерацию экспортируемых PDF-документов.

Такой подход обеспечивает модульность, расширяемость и чёткое разделение ответственности между слоями, что упрощает сопровождение и развитие приложения.

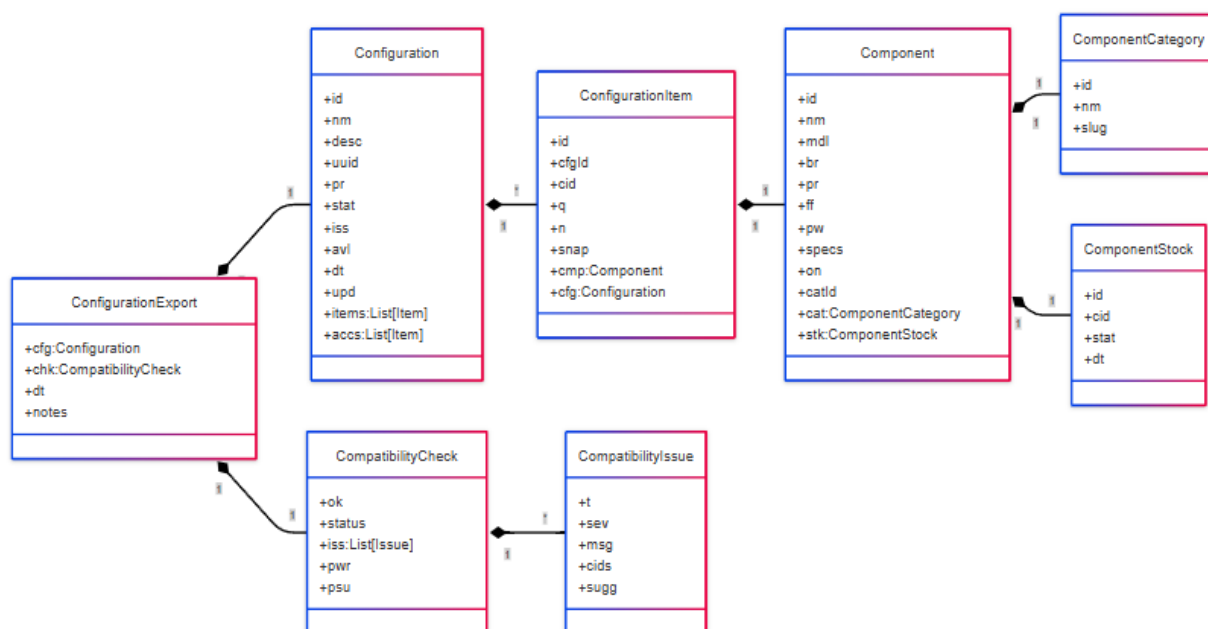


Рисунок 31 – Диаграмма классов серверной части слоя модели

Диаграмма классов слоя контроллеров представлена на рисунке 32.

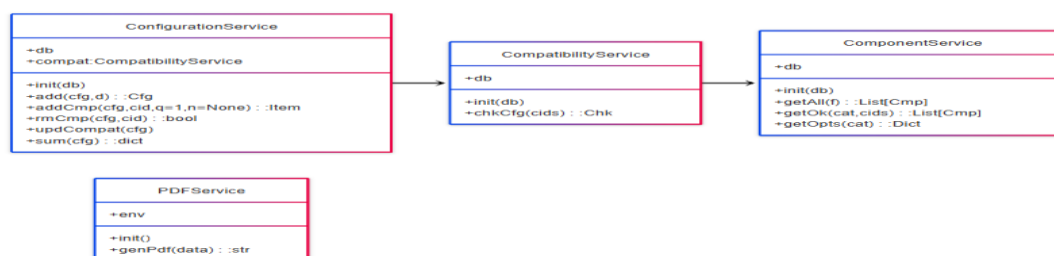


Рисунок 32 – Диаграмма классов серверной части слоя контроллеров

Описание классов слоя контроллеров приведено в таблице 18.

Таблица 18 – Описание классов слоя контроллеров

Класс	Назначение
ConfigurationService	Управление конфигурациями ПК: создание, добавление/удаление компонентов, обновление совместимости, расчёт итоговой стоимости и энергопотребления.
CompatibilityService	Проверка совместимости компонентов (CPU-MB, RAM-MB, корпус-MB, БП), формирование списка проблем и рекомендаций.
ComponentService	Работа с компонентами: фильтрация по брендам/категориям/цене, поиск совместимых деталей, получение характеристик.
PDFService	Генерация PDF-отчётов с конфигурациями.

Исходный код серверной части приведен в приложении Б.

3.3 Реализация web-конфигуратора

На данном этапе было реализован web-конфигуратор на языке Python [14 - 21] в среде PyCharm. Исходный код представлен в приложении В. Диаграмма компонентов, отражающая структуру программы, представлена на рисунке 33.

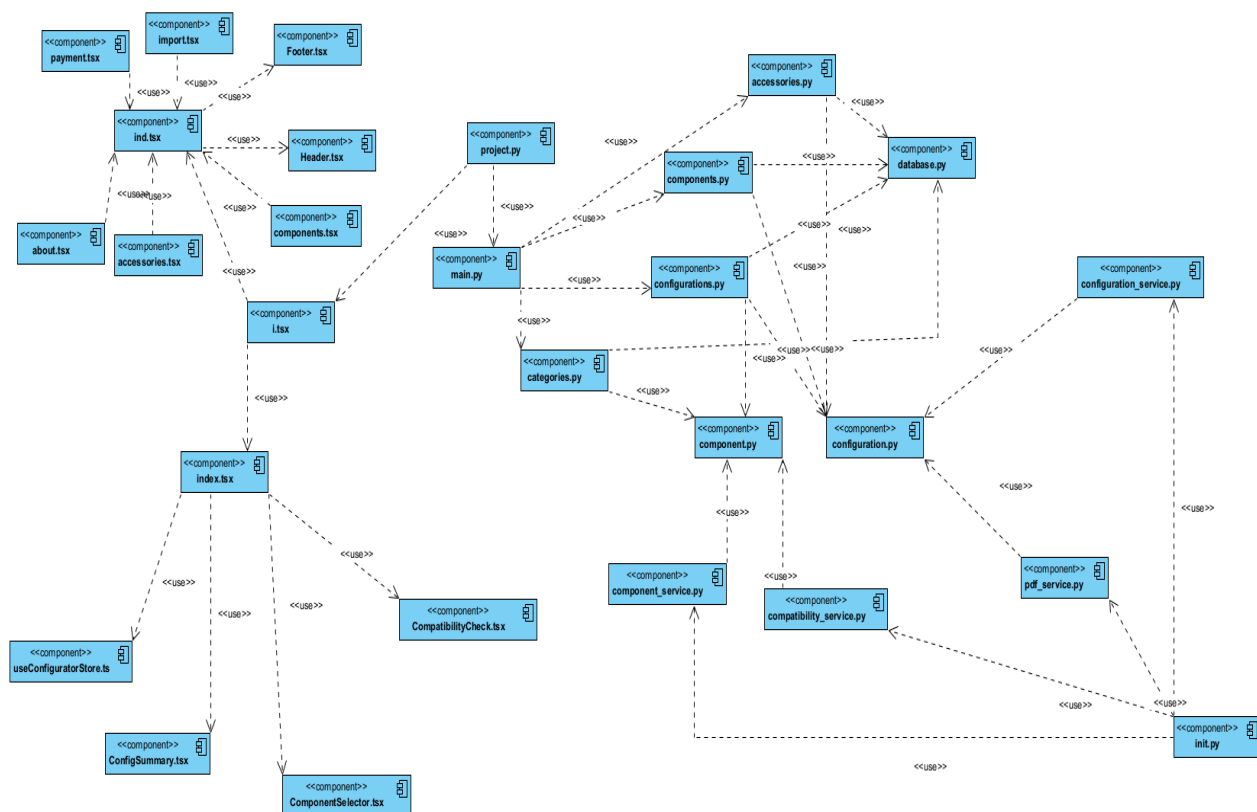


Рисунок 33 – Диаграмма компонентов web-конфигуратора

Описание компонентов представлено в таблице 19.

Таблица 19 – Описание компонентов web-конфигуратора приложения

Компонент	Назначение
project.py	Основной файл проекта, содержащий точку входа или управление основными процессами.
main.py	Главный скрипт для запуска приложения или сервиса.
categories.py	Управление категориями компонентов
component.py	Модель или логика работы с отдельными компонентами системы.
compatibility_service.py	Сервис проверки совместимости компонентов между собой.
database.py	Работа с базой данных: подключение, запросы, управление данными.
configuration_service.py	Сервис для управления конфигурациями системы или пользовательскими настройками.
configuration.py	Модель или логика работы с конфигурациями.
pdf_service.py	Генерация PDF-документов (например, отчетов или конфигураций).
init.py	Инициализация модулей или пакетов
component_service.py	Содержит бизнес-логику для работы с компонентами ПК (создание, чтение, обновление, удаление). Взаимодействует с database.py для хранения/получения данных.
accessories.py	Управляет аксессуарами для ПК (мыши, клавиатуры и т. д.). Включает функции проверки совместимости с другими компонентами через compatibility_service.py.
database.py	Обеспечивает подключение к БД и базовые операции (CRUD). Содержит настройки соединения, модели данных и ORM-запросы. Используется всеми сервисами.
configuration_service.py	Отвечает за сборку конфигураций ПК: проверяет совместимость компонентов (через compatibility_service.py), рассчитывает итоговую стоимость и формирует готовые сборки.
payment.tsx	Отвечает за обработку платежей и отображение платежных форм.
import.tsx	Предназначен для импорта данных или файлов в приложение.
ind.tsx	Вероятно, главная страница или точка входа (возможно, опечатка в названии).
Header.tsx	Содержит шапку сайта с навигацией и логотипом.
about.tsx	Отображает информацию о проекте или компании.
accessories.tsx	Показывает аксессуары или дополнительные компоненты.
components.tsx	Основной файл для сборки или экспорта компонентов.
compatibilitycheck.tsx	Проверяет совместимость выбранных компонентов или устройств.
index.tsx	Главный файл для рендеринга приложения или страницы.
useConfiguratorStore.ts	Хранит состояние конфигуратора (например, Redux или Zustand store).
configSummary.tsx	Отображает сводку выбранной конфигурации.
componentsSelector.tsx	Специализированный компонент для лектора (возможно, часть образовательной системы).
Footer.tsx	Компонент подвала сайта, содержащий логотип, описание проекта, навигационные ссылки (Конфигуратор, О проекте, Аксессуары, Программы), информацию о дипломной работе

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 34.

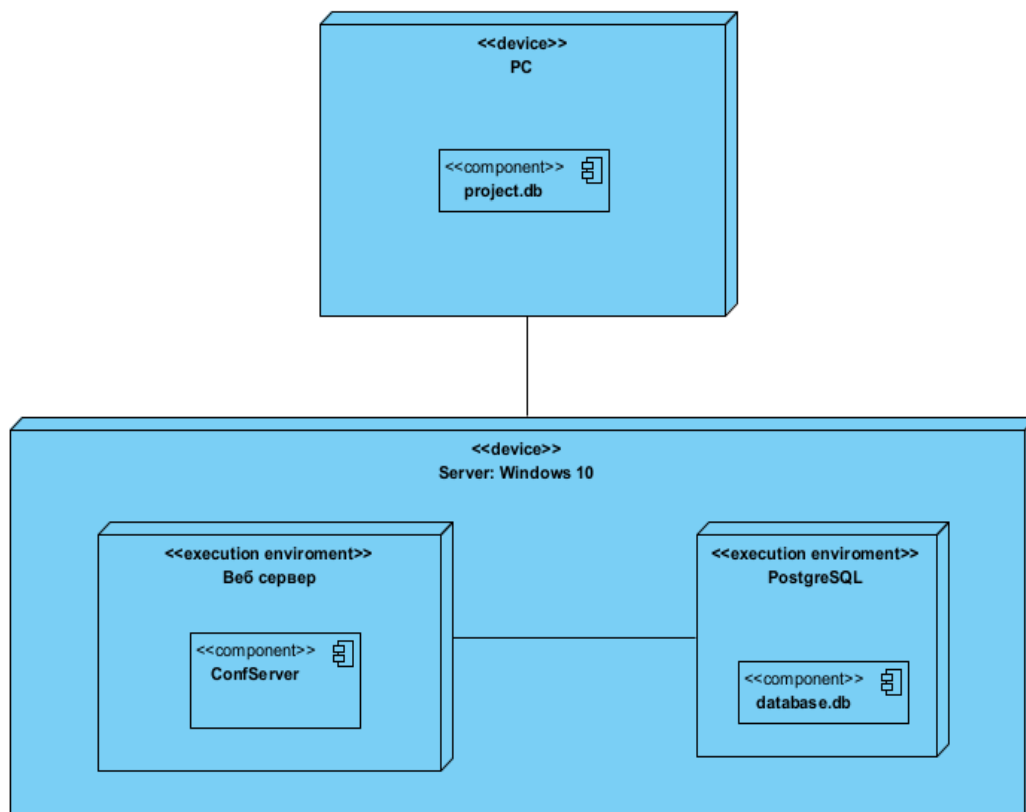


Рисунок 34 – Диаграмма развертывания программного обеспечения

Из представленной диаграммы можно увидеть, что программное обеспечение реализовано по трёхуровневой архитектуре, включающей клиентское приложение, серверную часть и подсистему управления базой данных. Все компоненты развёрнуты на устройствах под управлением операционной системы Windows 10. Такая архитектура обеспечивает четкое разделение логики приложения, что повышает его масштабируемость и удобство поддержки. Клиентский уровень отвечает за взаимодействие с пользователем, серверный — за обработку бизнес-логики, а уровень базы данных — за хранение и управление информацией. Подобная структура также позволяет при необходимости модернизировать отдельные компоненты без значительного влияния на работу системы в целом.

3.4 Тестирование программного обеспечения

3.4.1 Тестирование серверной части приложения

3.4.1.1 Функциональное тестирование

На данном этапе было выполнено функциональное тестирование [22-31] в целях проверки правильности работы серверной части приложения. Результаты функционального тестирования серверной части приложения представлены в таблице 20.

Таблица 20 – Результаты функционального тестирования серверной части

№	Описание теста	Структура запроса	Результат
1	Получить все категории компонентов	Метод: GET URL: /api/v1/categories	Тест пройден. (рисунок Г.1).
2	Получить категорию по ID	Метод: GET URL: /api/v1/categories/{category_id}	Тест пройден. Информация получена (рисунок Г.2).
3	Создать новую категорию	Метод: POST URL: /api/v1/categories	Тест пройден. Список получен (рисунок Г.3).
4	Получение компонентов с фильтрацией	Метод: GET URL: /api/v1/categories	Тест пройден. Список получен (рисунок Г.4).
5	Получить компонент по ID	Метод: GET URL: /api/v1/components/{component_id}	Тест пройден. Список получен (рисунок Г.5).
6	Получить компоненты категории с фильтрацией и поиском	Метод: GET URL: /api/v1/components/category/{category_slug}	Тест пройден. Информация получена (рисунок Г.6).
7	Проверить совместимость набора компонентов	Метод: POST URL: /api/v1/components/check-compatibility	Тест пройден. Данные изменены (рисунок Г.7).
8	Получить все ID компонентов для отладки	Метод: GET URL: /api/v1/components/debug/ids	Тест пройден. Пассажир добавлен (рисунок Г.8).

Продолжение таблицы 20

№	Описание теста	Структура запроса	Результат
9	Получить доступные варианты для фильтров	Метод: GET URL: /api/v1/components/filters/options	Тест пройден. Бронь добавлена (рисунок Г.9).
10	Создать новую конфигурацию	Метод: POST URL: /api/v1/configurations	Тест пройден. Бронь выкуплена (рисунок Г.10).
11	Получить список конфигураций	Метод: GET URL: /api/v1/configurations	Тест пройден. (рисунок Г.11).
12	Получить конфигурацию по ID	Метод: GET URL: /api/v1/configurations/{config_id}	Тест пройден. Информация получена (рисунок Г.12).
13	Обновить конфигурацию	Метод: PUT URL: /api/v1/configurations/{config_id}	Тест пройден. Список получен (рисунок Г.13).
14	Удалить конфигурацию	Метод: DELETE URL: /api/v1/configurations/{config_id}	Тест пройден. Конфигурация удалена (рисунок Г.14).
15	Получить конфигурацию по публичному UUID	Метод: GET URL: /api/v1/configurations/uuid/{public_uuid}	Тест пройден. Конфигурация получена (рисунок Г.15).
16	Добавить компонент в конфигурацию	Метод: POST URL: /api/v1/configurations/{config_id}/items	Тест пройден. Компонент добавлен (рисунок Г.16).
17	Добавить аксессуар в конфигурацию	Метод: POST URL: /api/v1/configurations/{config_id}/accessories	Тест пройден. Аксессуар добавлен (рисунок Г.17).
18	Удалить аксессуар из конфигурации	Метод: DELETE URL: /api/v1/configurations/{config_id}/accessories/{accessory_id}	Тест пройден. Аксессуар удалён (рисунок Г.18).

Все тесты тестирования были пройдены. Значит можно говорить о правильности работы серверной части приложения.

3.4.2 Тестирование клиентской части приложения

Также было произведено функциональное тестирование web-конфигуратора, результаты которого приведены в таблице 21.

Таблица 21 – Результаты функционального тестирования web-конфигуратора

Номер теста	Тестируемая ситуация	Описание теста	Ожидаемый результат	Наблюдаемый результат	Результат теста
1	Редактирование конфигурации	Проверить возможность редактирования существующей конфигурации.	Конфигурация открывается для редактирования, изменения сохраняются.	Рисунок Д.1	Тест пройден
2	Проверка совместимости компонентов	Добавить несовместимые компоненты и проверить реакцию системы.	Система выводит предупреждение о несовместимости.	Рисунок Д.2	Тест пройден
3	Выбор процессора	Выбрать процессор из списка доступных.	Процессор добавляется в конфигурацию, отображается в списке выбранных.	Рисунок Д.3	Тест пройден
4	Выбор материнской платы	Выбрать материнскую плату, совместимую с выбранным процессором.	Материнская плата добавляется в конфигурацию, отображается в списке выбранных.	Рисунок Д.4	Тест пройден
15	Сохранение конфигурации	Нажать кнопку «Сохранить конфигурацию» после её создания.	Конфигурация сохраняется, отображается сообщение об успешном сохранении.	Рисунок Д.5	Тест пройден
6	Выбор комплектующих	Добавить несколько комплектующих (ОЗУ, видеокарта, корпус).	Все комплектующие отображаются в конфигурации, проверка совместимости пройдена.	Рисунок Д.6	Тест пройден

Продолжение таблицы 21

Номер теста	Тестируемая ситуация	Описание теста	Ожидаемый результат	Наблюдаемый результат	Результат теста
7	Выбор накопителя	Выбрать накопитель из списка доступных.	Накопитель добавляется в конфигурацию, отображается в списке выбранных.	Рисунок Д.7	Тест пройден
8	Выбор кулера	Выбрать кулер, совместимый с выбранным процессором.	Кулер добавляется в конфигурацию, отображается в списке выбранных.	Рисунок Д.8	Тест пройден
9	Импорт конфигурации из PDF	Загрузить файл PDF с корректной конфигурацией.	Конфигурация успешно импортирована, данные отображаются в интерфейсе.	Рисунок Д.9	Тест пройден
10	Добавление аксессуаров	Выбрать аксессуар (мышь, клавиатура) и добавить их в конфигурацию.	Аксессуары отображаются в конфигурации, их можно редактировать или удалять.	Рисунок Д.10	Тест пройден
11	Выбор видеокарты	Выбрать видеокарту, совместимую с материнской платой.	Видеокарта добавляется в конфигурацию, отображается в списке выбранных.	Рисунок Д.11	Тест пройден
12	Выбор оперативной памяти	Выбрать ОЗУ, совместимую с материнской платой.	ОЗУ добавляется в конфигурацию, отображается в списке выбранных.	Рисунок Д.12	Тест пройден
13	Выбор корпуса	Выбрать корпус, совместимый с выбранными компонентами.	Корпус добавляется в конфигурацию, отображается в списке выбранных.	Рисунок Д.13	Тест пройден
14	Удаление сохранённой конфигурации	Удалить ранее сохранённую конфигурацию.	Конфигурация удаляется.	Рисунок Д.14	Тест пройден

Все тесты были выполнены успешно, что подтверждает стабильную и корректную работу web-конфигуратора. Результаты тестирования демонстрируют полное соответствие заявленным функциональным требованиям и отсутствие критических ошибок в работе системы.

3.4.3 Метрики кода

Метрики кода позволяют получить количественные величины, характеризующее программное обеспечение по тем или иным свойствам. Существует большое количество разнообразных метрик [32]. Одной из самых простых метрик кода является количество строк кода (SLOC-метрики). Также одной из самых распространенных метрик, основанных на подсчете некоторых единиц в коде программы, относят метрики Холстеда.

Рассмотрим метод класса `CompabilityService`, отвечающего за проверку совместимости корпуса и материнской платы, код метода приведён в листинге 1.

Листинг 1 – Коды методов класса `CompabilityService`

```
def _check_case_motherboard_compatibility(self,
components_by_category: Dict[str, List[Component]]) ->
List[CompatibilityIssue]:
    """Проверка совместимости корпуса и материнской платы"""
    issues = [] # Список для хранения выявленных проблем
    совместимости

    # Получаем список выбранных корпусов и материнских плат из
    словаря по категориям
    cases = components_by_category.get('case', [])
    motherboards = components_by_category.get('motherboard', [])

    # Проверяем, что выбран только один корпус и одна материнская
    плата
    if len(cases) == 1 and len(motherboards) == 1:
        case = cases[0] # Получаем единственный выбранный корпус
        motherboard = motherboards[0] # Получаем единственную
        выбранную материнскую плату

        # Получаем список поддерживаемых форм-факторов от корпуса
        case_form_factors =
case.specifications.get('supported_form_factors', [])
        # Получаем форм-фактор материнской платы
        mb_form_factor = motherboard.form_factor
```

```

        # Если форм-фактор материнской платы не поддерживается
        корпусом – добавляем проблему совместимости
        if mb_form_factor and case_form_factors and mb_form_factor
        not in case_form_factors:
            issues.append(CompatibilityIssue(
                type="form_factor_mismatch", # Тип проблемы:
                # несовпадение форм-факторов
                severity="error", # Уровень серьезности: ошибка
                message=f"Форм-фактор материнской платы
                {mb_form_factor} не поддерживается корпусом",
                component_ids=[case.id, motherboard.id], # ID
                # компонентов, между которыми проблема
                suggestions=[f"Выберите корпус с поддержкой
                {mb_form_factor}"] # Рекомендация по устранению
            ))

```

```

return issues # Возвращаем список проблем совместимости

```

Данный код находится в серверной части приложения и отвечает за проверку совместимости компонентов. При попытке создать конфигурацию этот метод проверяет на совместимость материнскую плату и корпус. Это позволяет уберечь пользователя от создания несовместимой конфигурации. В случае обнаружения несоответствия форм-фактора, метод возвращает информацию об ошибке с описанием проблемы и рекомендацией по её устранению. Такая проверка способствует повышению надёжности и удобства использования веб-конфигуратора.

Перед проведением расчетов необходимо определить исходные параметры программного кода, включая количество операторов и операндов. Эти данные послужат основой для последующего анализа метрик.

Для данных методов будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

3.4.3.1 Метрики размера программы

Данная группа метрик используется для измерения объема программы с помощью подсчета количества строк в тексте исходного кода. При этом часто выделяют методику подсчета физических строк кода и строк комментариев. Вычислим метрики размера программы для кода, приведенного в листинге 1 (таблица 22).

Таблица 22 – Метрики размера программы

Метрика	Количество
Количество строк кода	42
Количество строк комментариев	9

Теперь вычислим процент комментариев. Он равен $(9/42)*100\%=21.4\%$. Это означает, что количество комментариев в коде достаточно для его хорошего понимания, и при этом не слишком велико и не переполняет программу.

3.4.3.2 Метрики Холстеда

Метрика Холстеда вычисляется на основе анализа числа строк и синтаксических элементов исходного кода программы, а именно используются характеристики: число уникальных операторов, число уникальных операндов, общее число операторов и общее число операндов [20 – 34]. На основе этих характеристик рассчитываются такие показатели, как объём программы, её сложность и уровень понимания, что позволяет количественно оценить качество и трудоёмкость программного кода. Для измерения данных характеристик необходимо составить словарь операторов (таблица 23) и словарь операндов (таблица 24).

Таблица 23 – Словарь операторов

Оператор	Число вхождений
def	1
return	1
=	6
.	6
:	6
[]	5
()	6
if	1
and	2
not in	1
append()	1
get()	2
len()	2
f'''	1

Таблица 24 – Словарь операндов

Операнд	Число вхождений
components_by_category	3
'case'	1
'motherboard'	1
cases	2
motherboards	2
case	2
motherboard	2
case.specifications	1
'supported_form_factors'	1
motherboard.form_factor	1
mb_form_factor	2
case_form_factors	2
issues	2
CompatibilityIssue	1
"form_factor_mismatch"	1
"error"	1
case.id	1
motherboard.id	1

Из вышеописанных таблиц получаем следующие данные: число уникальных операторов программы $n_1 = 14$, число уникальных операндов программы $n_2 = 18$, общее число операторов в программе $N_1 = 42$, общее число операндов в программе $N_2 = 29$.

Вычислим некоторые из существующих характеристик на основании полученных данных:

- длина программы (N):

$$N = N_1 + N_2 = 42 + 29 = 71;$$

- словарь программы (n):

$$n = n_1 + n_2 = 14 + 18 = 32;$$

- теоретический объем программы (V):

$$V = N \cdot \log_2 N \approx 355;$$

- сложность программы (D):

$$D = \left(\frac{n_1}{2}\right) \cdot \left(\frac{N_2}{n_2}\right) \approx 11.28;$$

- оценка интеллектуальных усилий (E):

$$E = D \cdot V = 4004.4.$$

Метрики показывают, что функция имеет умеренную сложность, не слишком большой объём, а ожидаемое число ошибок близко к нулю. что указывает на хорошее качество и небольшую логическую нагрузку кода.

Заключение

В ходе выполнения выпускной квалификационной работы было разработано веб-приложение – конфигуратор персонального компьютера, позволяющий пользователям подбирать совместимые комплектующие с учётом своих требований.

На этапе анализа предметной области были изучены существующие аналоги, выявлены их сильные и слабые стороны, а также определены целевая аудитория и основные требования к функциональности и интерфейсу приложения.

В рамках проектирования была разработана архитектура системы, спроектирована база данных и определены технологии реализации. В качестве инструментов были выбраны FastAPI для серверной части, SQLAlchemy для работы с базой данных и Next.js для клиентской части.

На этапе реализации была создана база данных в среде MySQL Workbench, разработаны серверные маршруты и логика работы с конфигурациями, а также реализован пользовательский интерфейс, обеспечивающий удобную навигацию и визуализацию конфигурации ПК.

В перспективе планируется дополнить функционал системой рекомендаций и интеграцией с API поставщиков комплектующих.

Таким образом, поставленные цели и задачи были успешно достигнуты, и система web-конфигуратора была реализована в полном объёме.

Список использованных источников

1. PCPartPicker API Documentation [Электронный ресурс]. – Режим доступа: (дата обращения: 01.06.2025).
2. Демура, Н. А. Цифровизация: сущность и роль в развитии национальной экономики / Н. А. Демура, Н. П. Путинцева // Научный результат. Экономические исследования. – 2021. – № 1. – С. 22–30.
3. Статистика рынка комплектующих для ПК [Электронный ресурс] // Statista. – Режим доступа: <https://www.statista.com> (дата обращения: 01.06.2025).
4. Программа «Цифровая экономика Российской Федерации»: утверждена распоряжением Правительства РФ от 28 июля 2017 г. № 1632-р [Электронный ресурс]. – Режим доступа: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 03.04.2025).
5. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка: пер. с англ. / Дж. Рамбо, М. Блаха. – 2-е изд. – СПб. : Питер, 2007. – 544 с.
6. Крачтен, Ф. Введение в Rational Unified Process / Ф. Крачтен. – 2-е изд. – М. : Вильямс, 2002. – 240 с.
7. Беркун, С. Искусство управления IT-проектами / С. Беркун. – СПб. : Питер, 2007. – 400 с.
8. Гвоздев, В. Е. Управление программными проектами: учебное пособие для вузов / В. Е. Гвоздев. – М. : Юрайт, 2023. – 167 с.
9. Иванов, П. В. Управление проектами: учебное пособие / П. В. Иванов. – Ростов н/Д : Феникс, 2018. – 102 с.
10. Документация по MySQL [Электронный ресурс]. – Режим доступа: <https://dev.mysql.com/doc/> (дата обращения: 11.04.2025).
11. Дьяконов, В. Ю. MySQL 8.0: справочник администратора / В. Ю. Дьяконов. – М. : Солон-Пресс, 2021. – 512 с.

12. Документация по ASP.NET [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/aspnet/core/> (дата обращения: 11.04.2025).
13. FastAPI Tutorial [Электронный ресурс]. – Режим доступа: <https://fastapi.tiangolo.com/> (дата обращения: 01.06.2025).
14. Маттис, М. Изучаем Python / М. Маттис. – 5-е изд. – СПб.: Питер, 2022. – 560 с.
15. Свейгарт, А. Автоматизация рутинных задач с помощью Python / А. Свейгарт. – М.: ДМК Пресс, 2021. – 592 с.
16. Грон, Л. Изучаем программирование на Python / Л. Грон. – СПб.: Питер, 2020. – 400 с.
17. McKinney, W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython / W. McKinney. – 3rd ed. – Sebastopol: O'Reilly Media, 2022. – 544 p.
18. Документация по FastAPI. [Электронный ресурс]. URL: <https://fastapi.tiangolo.com/ru/> (дата обращения: 15.06.2025).
19. Документация по SQLAlchemy. [Электронный ресурс]. URL: <https://docs.sqlalchemy.org/> (дата обращения: 15.06.2025).
20. Документация PyCharm. [Электронный ресурс]. URL: <https://www.jetbrains.com/pycharm/documentation/> (дата обращения: 15.06.2025).
21. Лутц, М. Изучаем Python. Том 1. Основы / М. Лутц. – 5-е изд. – СПб.: Питер, 2023. – 800 с.
22. Савин, А. Тестирование программных систем: учебное пособие / А. Савин. – 5-е изд. – СПб.: Питер, 2020. – 448 с.
23. Канер, С. Тестирование программного обеспечения / С. Канер, Дж. Фолк, Х. К. Нгуен. – М.: ДМК Пресс, 2022. – 560 с.
24. Гленфорд, Майерс. Искусство тестирования программ / Г. Майерс, К. Сандлер, Т. Баджетт. – М.: Вильямс, 2019. – 368 с.
25. Лапшин, А. В. Тестирование программного обеспечения: системный подход / А. В. Лапшин. – М.: БХВ-Петербург, 2018. – 304 с.

26. Куликов, А. В. Практика тестирования программного обеспечения / А. В. Куликов. – М.: ДМК Пресс, 2021. – 272 с.
27. Источников, В. В. Основы тестирования программного обеспечения / В. В. Источников. – СПб.: Питер, 2020. – 320 с.
28. Черников, С. И. Основы тестирования ПО: учебное пособие / С. И. Черников. – М.: Инфра-М, 2022. – 210 с.
29. IEEE Std 829-2008. IEEE Standard for Software and System Test Documentation. – New York: IEEE, 2008. – 54 p.
30. ISTQB Foundation Level Certification Syllabus, Version 4.0. – International Software Testing Qualifications Board, 2023. – [Электронный ресурс]. URL: <https://www.istqb.org/certification-path-root/foundation-level> (дата обращения: 15.06.2025).
31. Documentation – Functional Testing Basics. [Электронный ресурс]. URL: <https://www.softwaretestinghelp.com/functional-testing/> (дата обращения: 15.06.2025).
32. Программный код и его метрики [Электронный ресурс] // Habr. – Режим доступа: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 05.05.2025).

Приложение А
(справочное)
Глоссарий

UML (Unified Modeling Language) – это стандартизированный язык графического моделирования, используемый для визуализации, проектирования и документирования объектно-ориентированных систем в разработке ПО, бизнес-анализе и проектировании сложных систем.

RUP (Rational Unified Process) – это гибкая методология разработки программного обеспечения, основанная на итеративном подходе, которая включает лучшие практики управления проектами и инженерии требований.

СУБД (Система Управления Базой Данных) – это программное обеспечение, предназначенное для создания, управления и взаимодействия с базами данных через специализированные интерфейсы.

SQL (Structured Query Language) – это язык структурированных запросов, предназначенный для работы с реляционными базами данных: создания, изменения, удаления и выборки информации.

HTTP (HyperText Transfer Protocol) – это протокол передачи данных в сети, работающий по принципу «клиент-сервер» и используемый для обмена информацией в веб-приложениях.

REST API – это архитектурный подход к проектированию веб-сервисов, обеспечивающий стандартизированное взаимодействие между клиентом и сервером через HTTP-запросы.

Приложение Б
(обязательное)

Исходный код серверной части приложения

Листинг Б.1 – Файл config.py

```

from pydantic_settings import BaseSettings
from pydantic import field_validator
from typing import Optional, List, Union
import json

class Settings(BaseSettings):
    DATABASE_URL: str =
"postgresql://postgres:postgres@localhost:5432/pc_configurator"
    DATABASE_HOST: Optional[str] = "localhost"
    DATABASE_PORT: Optional[int] = 5432
    DATABASE_USER: Optional[str] = "postgres"
    DATABASE_PASSWORD: Optional[str] = "postgres"
    DATABASE_NAME: Optional[str] = "pc_configurator"

    JWT_SECRET_KEY: str = "your-secret-key-here"
    JWT_ALGORITHM: str = "HS256"
    ACCESS_TOKEN_EXPIRE_MINUTES: int = 30

    CORS_ORIGINS: Union[str, List[str]] = [
        "http://localhost:3000",
        "http://127.0.0.1:3000",
    ]

    ENVIRONMENT: str = "development"
    DEBUG: bool = True

    PDF_TEMP_PATH: str = "/tmp/pc_configs"

    @field_validator("CORS_ORIGINS", mode="before")
    @classmethod
    def split_cors(cls, v):
        if isinstance(v, str):
            try:
                return json.loads(v) # JSON-строка?
            except json.JSONDecodeError:
                return [i.strip() for i in v.split(",")]
        return v

    class Config:
        env_file = ".env"
        extra = "allow"

settings = Settings()

```

Листинг Б.2 – Файл database.py

```

from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
from .config import settings

engine = create_engine(settings.DATABASE_URL)

```

```

SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)

Base = declarative_base()

def get_db():
    """Dependency для получения сессии БД"""
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

```

Листинг Б.3 – Файл main.py

```

from fastapi import FastAPI, Request
from fastapi.middleware.cors import CORSMiddleware
from fastapi.staticfiles import StaticFiles
from fastapi.responses import JSONResponse
from .config import settings
from .database import engine, Base
from .routers import components, configurations, categories,
accessories
import os
import logging
import time
import traceback

# Настройка логирования
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

app = FastAPI(
    title="Конфигуратор ПК API",
    description="API для веб-конфигуратора персональных
компьютеров",
    version="1.0.0",
    docs_url="/docs",
    redoc_url="/redoc"
)

# Настройка CORS
app.add_middleware(
    CORSMiddleware,
    allow_origins=settings.CORS_ORIGINS,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Middleware для логирования запросов
@app.middleware("http")
async def log_requests(request: Request, call_next):
    start_time = time.time()

```

```

# Логируем входящий запрос
logger.info(f"● {request.method} {request.url}")

# Выполняем запрос
try:
    response = await call_next(request)
    process_time = time.time() - start_time

    # Логируем ответ
    if response.status_code >= 400:
        logger.error(f"● {request.method} {request.url} - {response.status_code} ({process_time:.3f}s)")
    else:
        logger.info(f"□ {request.method} {request.url} - {response.status_code} ({process_time:.3f}s)")

    return response
except Exception as e:
    process_time = time.time() - start_time
    logger.error(f"✱ {request.method} {request.url} - ERROR ({process_time:.3f}s): {str(e)}")
    logger.error(f"Traceback: {traceback.format_exc()}")
    return JSONResponse(
        status_code=500,
        content={"detail": f"Внутренняя ошибка сервера: {str(e)}"}
    )

# Создание директории для PDF файлов
os.makedirs(settings.PDF_TEMP_PATH, exist_ok=True)

# Подключение статических файлов
app.mount("/static", StaticFiles(directory="static"),
name="static")

# Подключение роутеров
app.include_router(categories.router, prefix="/api/v1",
tags=["Категории"])
app.include_router(components.router, prefix="/api/v1",
tags=["Компоненты"])
app.include_router(configurations.router, prefix="/api/v1",
tags=["Конфигурации"])
app.include_router(accessories.router, prefix="/api/v1",
tags=["Аксессуары"])

# Добавляем роутеры без префикса для совместимости с фронтом
app.include_router(categories.router, tags=["Категории (без префикса)"])
app.include_router(components.router, tags=["Компоненты (без префикса)"])

```

```

app.include_router(configurations.router, tags=["Конфигурации (без
префикса)"])
app.include_router(accessories.router, tags=["Аксессуары (без
префикса)"])

@app.get("/")
async def root():
    """Главная страница API"""
    return {
        "message": "Конфигуратор ПК API",
        "version": "1.0.0",
        "docs": "/docs",
        "redoc": "/redoc"
    }

@app.get("/health")
async def health_check():
    """Проверка здоровья сервиса"""
    return {"status": "ok", "environment": settings.ENVIRONMENT}

```

Листинг Б.4 – Файл compatibility_service.py

```

from typing import List, Dict, Any
from uuid import UUID
from sqlalchemy.orm import Session
from ..models import Component
from ..schemas.configuration import CompatibilityCheck,
CompatibilityIssue, CompatibilityStatus

class CompatibilityService:
    """Сервис для проверки совместимости компонентов ПК"""

    def __init__(self, db: Session):
        self.db = db

    def check_configuration_compatibility(self, component_ids:
List[UUID]) -> CompatibilityCheck:
        """Проверка совместимости конфигурации"""
        components =
self.db.query(Component).filter(Component.id.in_(component_ids)).a
ll()

        if not components:
            return CompatibilityCheck(
                is_compatible=False,
                status=CompatibilityStatus.UNKNOWN,
                issues=[],
                total_power_consumption=None,
                recommended_psu_wattage=None
            )

        # Группируем компоненты по категориям
        components_by_category = {}
        for comp in components:

```

```

        category_slug = comp.category.slug
        if category_slug not in components_by_category:
            components_by_category[category_slug] = []
        components_by_category[category_slug].append(comp)

    issues = []
    total_power = 0

    # Проверяем основные совместимости

    issues.extend(self._check_cpu_motherboard_compatibility(components_by_category))

    issues.extend(self._check_ram_motherboard_compatibility(components_by_category))

    issues.extend(self._check_case_motherboard_compatibility(components_by_category))

    # Считаем энергопотребление
    total_power =
self._calculate_total_power_consumption(components)
    power_issues =
self._check_power_supply_compatibility(components_by_category,
total_power)
    issues.extend(power_issues)

    # Определяем общий статус
    status = self._determine_compatibility_status(issues)
    is_compatible = status == CompatibilityStatus.COMPATIBLE

    # Рекомендуемая мощность БП (с запасом 20%)
    recommended_psu = int(total_power * 1.2) if total_power >
0 else None

    return CompatibilityCheck(
        is_compatible=is_compatible,
        status=status,
        issues=issues,
        total_power_consumption=total_power,
        recommended_psu_wattage=recommended_psu
    )

    def _check_cpu_motherboard_compatibility(self,
components_by_category: Dict[str, List[Component]]) ->
List[CompatibilityIssue]:
    """Проверка совместимости процессора и материнской
платы"""
    issues = []

    cpus = components_by_category.get('cpu', [])
    motherboards = components_by_category.get('motherboard',
[])
```

```

        if len(cpus) == 1 and len(motherboards) == 1:
            cpu = cpus[0]
            motherboard = motherboards[0]

            cpu_socket = cpu.specifications.get('socket')
            mb_socket = motherboard.specifications.get('socket')

            if cpu_socket and mb_socket and cpu_socket !=
mb_socket:
                issues.append(CompatibilityIssue(
                    type="socket_mismatch",
                    severity="error",
                    message=f"Несовместимые сокет: процессор
{cpu_socket}, материнская плата {mb_socket}",
                    component_ids=[cpu.id, motherboard.id],
                    suggestions=[
                        f"Выберите процессор с сокетом
{mb_socket}",
                        f"Выберите материнскую плату с сокетом
{cpu_socket}"
                    ]
                ))

        return issues

    def _check_ram_motherboard_compatibility(self,
components_by_category: Dict[str, List[Component]]) ->
List[CompatibilityIssue]:
        """Проверка совместимости RAM и материнской платы"""
        issues = []

        ram_modules = components_by_category.get('ram', [])
        motherboards = components_by_category.get('motherboard',
[[]])

        if ram_modules and len(motherboards) == 1:
            motherboard = motherboards[0]
            mb_memory_type =
motherboard.specifications.get('memory_type', [])
            mb_max_memory =
motherboard.specifications.get('max_memory_gb', 0)
            mb_memory_slots =
motherboard.specifications.get('memory_slots', 0)

            total_ram_gb = 0
            for ram in ram_modules:
                ram_type = ram.specifications.get('memory_type')
                ram_capacity =
ram.specifications.get('capacity_gb', 0)
                total_ram_gb += ram_capacity

            # Проверяем тип памяти

```

```

        if ram_type and mb_memory_type and ram_type not in
mb_memory_type:
            issues.append(CompatibilityIssue(
                type="memory_type_mismatch",
                severity="error",
                message=f"Тип памяти {ram_type} не
поддерживается материнской платой",
                component_ids=[ram.id, motherboard.id],
                suggestions=[f"Выберите память типа {'
'.join(mb_memory_type)}"]
            ))

        # Проверяем максимальный объем
        if mb_max_memory > 0 and total_ram_gb > mb_max_memory:
            issues.append(CompatibilityIssue(
                type="memory_capacity_exceeded",
                severity="warning",
                message=f"Объем памяти ({total_ram_gb}ГБ)
превышает максимум материнской платы ({mb_max_memory}ГБ)",
                component_ids=[ram.id for ram in ram_modules]
+ [motherboard.id],
                suggestions=[f"Уменьшите объем памяти до
{mb_max_memory}ГБ"]
            ))

        # Проверяем количество слотов
        if mb_memory_slots > 0 and len(ram_modules) >
mb_memory_slots:
            issues.append(CompatibilityIssue(
                type="memory_slots_exceeded",
                severity="error",
                message=f"Количество модулей памяти
({len(ram_modules)}) превышает количество слотов
({mb_memory_slots})",
                component_ids=[ram.id for ram in ram_modules]
+ [motherboard.id],
                suggestions=[f"Используйте не более
{mb_memory_slots} модулей памяти"]
            ))

        return issues

    def _check_case_motherboard_compatibility(self,
components_by_category: Dict[str, List[Component]]) ->
List[CompatibilityIssue]:
        """Проверка совместимости корпуса и материнской платы"""
        issues = []

        cases = components_by_category.get('case', [])
        motherboards = components_by_category.get('motherboard',
[])

        if len(cases) == 1 and len(motherboards) == 1:

```

```

        case = cases[0]
        motherboard = motherboards[0]

        case_form_factors =
case.specifications.get('supported_form_factors', [])
        mb_form_factor = motherboard.form_factor

        if mb_form_factor and case_form_factors and
mb_form_factor not in case_form_factors:
            issues.append(CompatibilityIssue(
                type="form_factor_mismatch",
                severity="error",
                message=f"Форм-фактор материнской платы
{mb_form_factor} не поддерживается корпусом",
                component_ids=[case.id, motherboard.id],
                suggestions=[f"Выберите корпус с поддержкой
{mb_form_factor}"]
            ))

        return issues

    def _check_power_supply_compatibility(self,
components_by_category: Dict[str, List[Component]], total_power:
int) -> List[CompatibilityIssue]:
        """Проверка совместимости блока питания"""
        issues = []

        psus = components_by_category.get('psu', [])

        if len(psus) == 1 and total_power > 0:
            psu = psus[0]
            psu_wattage = psu.specifications.get('wattage', 0)

            # Рекомендуем запас в 20%
            recommended_wattage = int(total_power * 1.2)

            if psu_wattage < total_power:
                issues.append(CompatibilityIssue(
                    type="power_insufficient",
                    severity="error",
                    message=f"Мощность БП ({psu_wattage}Вт)
недостаточна для системы ({total_power}Вт)",
                    component_ids=[psu.id],
                    suggestions=[f"Выберите БП мощностью от
{recommended_wattage}Вт"]
                ))
            elif psu_wattage < recommended_wattage:
                issues.append(CompatibilityIssue(
                    type="power_warning",
                    severity="warning",
                    message=f"Рекомендуется БП большей мощности
для надежности ({recommended_wattage}Вт+)",
                    component_ids=[psu.id],

```

```

        suggestions=[f"Рассмотрите БП мощностью
{recommended_wattage}Вт+ для большего запаса"]
    ))

    return issues

    def _calculate_total_power_consumption(self, components:
List[Component]) -> int:
        """Расчет общего энергопотребления"""
        total_power = 0

        for component in components:
            if component.power_consumption:
                total_power += component.power_consumption

        return total_power

    def _determine_compatibility_status(self, issues:
List[CompatibilityIssue]) -> CompatibilityStatus:
        """Определение общего статуса совместимости"""
        if not issues:
            return CompatibilityStatus.COMPATIBLE

        has_errors = any(issue.severity == "error" for issue in
issues)
        has_warnings = any(issue.severity == "warning" for issue
in issues)

        if has_errors:
            return CompatibilityStatus.INCOMPATIBLE
        elif has_warnings:
            return CompatibilityStatus.WARNING
        else:
            return CompatibilityStatus.COMPATIBLE

```

Листинг Б.5 – Файл component_service.py

```

from typing import List, Dict, Any, Optional
from sqlalchemy.orm import Session, joinedload
from sqlalchemy import and_, or_
from ..models import Component, ComponentCategory, ComponentStock
from ..schemas.component import ComponentCreate, ComponentFilter

class ComponentService:
    """Сервис для работы с компонентами"""

    def __init__(self, db: Session):
        self.db = db

    def get_components_with_filters(self, filters:
ComponentFilter) -> List[Component]:
        """Получить компоненты с применением фильтров"""

        query = self.db.query(Component).options(

```

```

        joinedload(Component.category),
        joinedload(Component.stock)
    ).filter(Component.is_active == True)

    # Применяем фильтры
    query = self._apply_filters(query, filters)

    # Пагинация
    offset = (filters.page - 1) * filters.limit
    components =
query.offset(offset).limit(filters.limit).all()

    return components

def get_compatible_components(
    self,
    category_slug: str,
    compatible_with_ids: List[int]
) -> List[Component]:
    """Получить компоненты категории, совместимые с
указанными"""

    # Получаем категорию
    category = self.db.query(ComponentCategory).filter(
        ComponentCategory.slug == category_slug
    ).first()

    if not category:
        return []

    # Получаем все компоненты категории
    components = self.db.query(Component).options(
        joinedload(Component.category),
        joinedload(Component.stock)
    ).filter(
        Component.category_id == category.id,
        Component.is_active == True
    ).all()

    # Если нет базовых компонентов для проверки, возвращаем
все
    if not compatible_with_ids:
        return components

    # Фильтруем по совместимости
    compatible_components = []
    existing_components =
self._get_components_by_ids(compatible_with_ids)

    for component in components:
        if self._is_component_compatible(component,
existing_components):
            compatible_components.append(component)

```

```

        return compatible_components

    def get_filter_options(self, category_slug: Optional[str] =
None) -> Dict[str, Any]:
        """Получить доступные варианты для фильтров"""

        query =
self.db.query(Component).filter(Component.is_active == True)

        if category_slug:
            query = query.join(ComponentCategory).filter(
                ComponentCategory.slug == category_slug
            )

        components = query.all()

        return self._extract_filter_options(components)

    def _apply_filters(self, query, filters: ComponentFilter):
        """Применить фильтры к запросу"""

        # Фильтр по категории
        if filters.category_slug:
            query = query.join(ComponentCategory).filter(
                ComponentCategory.slug == filters.category_slug
            )

        # Фильтр по бренду
        if filters.brand:
            query =
query.filter(Component.brand.in_(filters.brand))

        # Фильтр по цене
        if filters.price_min is not None:
            query = query.filter(Component.price >=
filters.price_min)
        if filters.price_max is not None:
            query = query.filter(Component.price <=
filters.price_max)

        # Фильтр по наличию
        if filters.only_in_stock:
            query = query.join(ComponentStock).filter(
                ComponentStock.status == "in_stock"
            )

        # Фильтр по форм-фактору
        if filters.form_factor:
            query =
query.filter(Component.form_factor.in_(filters.form_factor))

        # Фильтр по энергопотреблению

```

```

        if filters.power_max is not None:
            query = query.filter(
                or_(
                    Component.power_consumption <=
filters.power_max,
                    Component.power_consumption.is_(None)
                )
            )

        # Поиск по тексту
        if filters.search:
            search_filter = or_(
                Component.name.ilike(f"%{filters.search}%"),
                Component.model.ilike(f"%{filters.search}%"),
                Component.brand.ilike(f"%{filters.search}%")
            )
            query = query.filter(search_filter)

        # Специфичные фильтры
        if filters.socket:
            query = query.filter(
                Component.specifications["socket"].astext.in_(filters.socket)
            )

        if filters.memory_type:
            query = query.filter(
                Component.specifications["memory_type"].astext.in_(filters.memory_
type)
            )

        if filters.interface:
            query = query.filter(
                Component.specifications["interface"].astext.in_(filters.interface
)
            )

        return query

    def _get_components_by_ids(self, component_ids: List[int]) ->
List[Component]:
        """Получить компоненты по ID"""
        return self.db.query(Component).filter(
            Component.id.in_(component_ids)
        ).all()

    def _is_component_compatible(
        self,
        component: Component,
        existing_components: List[Component]
    ) -> bool:

```

```

        """Проверить совместимость компонента с существующими"""

        # Простая проверка совместимости по основным параметрам
        for existing in existing_components:
            if not self._check_basic_compatibility(component,
existing):
                return False

        return True

    def _check_basic_compatibility(
        self,
        component1: Component,
        component2: Component
    ) -> bool:
        """Базовая проверка совместимости между двумя
компонентами"""

        # Процессор + Материнская плата: проверка сокета
        if (component1.category.slug == "cpu" and
component2.category.slug == "motherboard") or \
            (component1.category.slug == "motherboard" and
component2.category.slug == "cpu"):

            socket1 = component1.specifications.get("socket")
            socket2 = component2.specifications.get("socket")

            if socket1 and socket2 and socket1 != socket2:
                return False

        # RAM + Материнская плата: проверка типа памяти
        if (component1.category.slug == "ram" and
component2.category.slug == "motherboard") or \
            (component1.category.slug == "motherboard" and
component2.category.slug == "ram"):

            ram_type = None
            mb_supported_types = None

            if component1.category.slug == "ram":
                ram_type =
component1.specifications.get("memory_type")
                mb_supported_types =
component2.specifications.get("memory_type", [])
            else:
                ram_type =
component2.specifications.get("memory_type")
                mb_supported_types =
component1.specifications.get("memory_type", [])

            if ram_type and mb_supported_types:
                if isinstance(mb_supported_types, str):
                    mb_supported_types = [mb_supported_types]

```

```

        if ram_type not in mb_supported_types:
            return False

        # Корпус + Материнская плата: проверка форм-фактора
        if (component1.category.slug == "case" and
            component2.category.slug == "motherboard") or \
            (component1.category.slug == "motherboard" and
            component2.category.slug == "case"):

            case_supported = None
            mb_form_factor = None

            if component1.category.slug == "case":
                case_supported =
component1.specifications.get("supported_form_factors", [])
                mb_form_factor = component2.form_factor
            else:
                case_supported =
component2.specifications.get("supported_form_factors", [])
                mb_form_factor = component1.form_factor

            if case_supported and mb_form_factor and
mb_form_factor not in case_supported:
                return False

        return True

    def _extract_filter_options(self, components: List[Component])
-> Dict[str, Any]:
        """Извлечь варианты для фильтров из списка компонентов"""

        brands = set()
        form_factors = set()
        sockets = set()
        memory_types = set()
        interfaces = set()
        prices = []

        for comp in components:
            if comp.brand:
                brands.add(comp.brand)

            if comp.form_factor:
                form_factors.add(comp.form_factor)

            prices.append(comp.price)

            # Извлекаем из specifications
            specs = comp.specifications or {}

            if "socket" in specs:
                sockets.add(specs["socket"])

```

```

    if "memory_type" in specs:
        mem_type = specs["memory_type"]
        if isinstance(mem_type, list):
            memory_types.update(mem_type)
        else:
            memory_types.add(mem_type)

    if "interface" in specs:
        interfaces.add(specs["interface"])

    return {
        "brands": sorted(list(brands)),
        "form_factors": sorted(list(form_factors)),
        "sockets": sorted(list(sockets)),
        "memory_types": sorted(list(memory_types)),
        "interfaces": sorted(list(interfaces)),
        "price_range": {
            "min": min(prices) if prices else 0,
            "max": max(prices) if prices else 0
        }
    }
}

```

Листинг Б.6 – Файл configuration_service.py

```

from typing import List, Optional
from sqlalchemy.orm import Session, joinedload
from datetime import datetime
import uuid
from ..models import Configuration, ConfigurationItem, Component,
ComponentStock
from ..schemas.configuration import ConfigurationCreate,
AvailabilityStatus
from .compatibility_service import CompatibilityService

class ConfigurationService:
    """Сервис для работы с конфигурациями ПК"""

    def __init__(self, db: Session):
        self.db = db
        self.compatibility_service = CompatibilityService(db)

    def create_configuration(self, config_data:
ConfigurationCreate) -> Configuration:
        """Создать новую конфигурацию"""

        db_config = Configuration(
            name=config_data.name,
            description=config_data.description,
            public_uuid=str(uuid.uuid4())
        )

        self.db.add(db_config)
        self.db.commit()
        self.db.refresh(db_config)

```

```

        return db_config

def add_component_to_configuration(
    self,
    config_id: int,
    component_id: int,
    quantity: int = 1,
    notes: Optional[str] = None
) -> ConfigurationItem:
    """Добавить компонент в конфигурацию"""

    # Проверяем существование конфигурации и компонента
    config =
self.db.query(Configuration).filter(Configuration.id ==
config_id).first()
    if not config:
        raise ValueError("Конфигурация не найдена")

    component = self.db.query(Component).filter(Component.id
== component_id).first()
    if not component:
        raise ValueError("Компонент не найден")

    # Проверяем, нет ли уже такого компонента
    existing_item = self.db.query(ConfigurationItem).filter(
        ConfigurationItem.configuration_id == config_id,
        ConfigurationItem.component_id == component_id
    ).first()

    if existing_item:
        # Обновляем существующий
        existing_item.quantity = quantity
        existing_item.notes = notes
        self.db.commit()
        self.db.refresh(existing_item)

        # Обновляем общие показатели
        self._update_configuration_totals(config_id)

        return existing_item
    else:
        # Создаем новый
        db_item = ConfigurationItem(
            configuration_id=config_id,
            component_id=component_id,
            quantity=quantity,
            notes=notes,
            price_snapshot=component.price
        )

        self.db.add(db_item)
        self.db.commit()

```

```

        self.db.refresh(db_item)

        # Обновляем общие показатели
        self._update_configuration_totals(config_id)

        return db_item

    def remove_component_from_configuration(self, config_id: int,
component_id: int) -> bool:
        """Удалить компонент из конфигурации"""

        item = self.db.query(ConfigurationItem).filter(
            ConfigurationItem.configuration_id == config_id,
            ConfigurationItem.component_id == component_id
        ).first()

        if not item:
            return False

        self.db.delete(item)
        self.db.commit()

        # Обновляем общие показатели
        self._update_configuration_totals(config_id)

        return True

    def update_configuration_compatibility(self, config_id: int) -
> None:
        """Обновить статус совместимости конфигурации"""

        config =
self.db.query(Configuration).filter(Configuration.id ==
config_id).first()
        if not config:
            return

        # Получаем все компоненты конфигурации
        items = self.db.query(ConfigurationItem).filter(
            ConfigurationItem.configuration_id == config_id
        ).all()

        if not items:
            config.compatibility_status = "unknown"
            config.compatibility_issues = []
            self.db.commit()
            return

        # Проверяем совместимость
        component_ids = [item.component_id for item in items]
        compatibility_result =
self.compatibility_service.check_configuration_compatibility(compo
nent_ids)

```

```

        # Обновляем конфигурацию
        config.compatibility_status =
compatibility_result.status.value
        config.compatibility_issues = [issue.dict() for issue in
compatibility_result.issues]
        config.updated_at = datetime.now()

        self.db.commit()

    def get_configuration_summary(self, config_id: int) -> dict:
        """Получить сводку по конфигурации"""

        config = self.db.query(Configuration).options(

joinedload(Configuration.items).joinedload(ConfigurationItem.compo
nent).joinedload(Component.category),

joinedload(Configuration.items).joinedload(ConfigurationItem.compo
nent).joinedload(Component.stock)
        ).filter(Configuration.id == config_id).first()

        if not config:
            return {}

        # Группируем компоненты по категориям
        components_by_category = {}
        total_power = 0
        total_price = 0

        for item in config.items:
            category = item.component.category.name
            if category not in components_by_category:
                components_by_category[category] = []

            components_by_category[category].append({
                "component": item.component,
                "quantity": item.quantity,
                "price": item.price_snapshot or
item.component.price
            })

            # Считаем общее энергопотребление
            if item.component.power_consumption:
                total_power += item.component.power_consumption *
item.quantity

            # Считаем общую стоимость
            price = item.price_snapshot or item.component.price
            total_price += price * item.quantity

        # Проверяем совместимость

```

```

        component_ids = [item.component_id for item in
config.items]
        compatibility =
self.compatibility_service.check_configuration_compatibility(compo
nent_ids)

        return {
            "configuration": config,
            "components_by_category": components_by_category,
            "total_power_consumption": total_power,
            "total_price": total_price,
            "compatibility": compatibility,
            "missing_categories":
self._get_missing_categories(config.items)
        }

    def _update_configuration_totals(self, config_id: int) ->
None:
        """Обновить общие показатели конфигурации"""

        config =
self.db.query(Configuration).filter(Configuration.id ==
config_id).first()
        if not config:
            return

        # Получаем все элементы конфигурации с компонентами
        items = self.db.query(ConfigurationItem).options(

joinedload(ConfigurationItem.component).joinedload(Component.stock
)

        ).filter(ConfigurationItem.configuration_id ==
config_id).all()

        if not items:
            config.total_price = 0.0
            config.availability_status =
AvailabilityStatus.UNKNOWN.value
            config.expected_delivery_date = None
            self.db.commit()
            return

        # Считаем общую стоимость
        total_price = 0.0
        for item in items:
            price = item.price_snapshot or item.component.price
            total_price += price * item.quantity

        # Определяем статус наличия
        availability_status =
self._calculate_availability_status(items)

        # Определяем ожидаемую дату поставки

```

```

        expected_date =
self._calculate_expected_delivery_date(items)

        # Обновляем конфигурацию
        config.total_price = total_price
        config.availability_status = availability_status.value
        config.expected_delivery_date = expected_date
        config.updated_at = datetime.now()

        self.db.commit()

        # Обновляем совместимость
        self.update_configuration_compatibility(config_id)

    def _calculate_availability_status(self, items:
List[ConfigurationItem]) -> AvailabilityStatus:
        """Рассчитать статус наличия конфигурации"""

        stock_statuses = []
        for item in items:
            if item.component.stock:
                stock_statuses.append(item.component.stock.status)
            else:
                stock_statuses.append("out_of_stock")

        if all(status == "in_stock" for status in stock_statuses):
            return AvailabilityStatus.AVAILABLE
        elif any(status == "in_stock" for status in
stock_statuses):
            return AvailabilityStatus.PARTIAL
        elif any(status == "expected" for status in
stock_statuses):
            return AvailabilityStatus.PARTIAL
        else:
            return AvailabilityStatus.UNAVAILABLE

    def _calculate_expected_delivery_date(self, items:
List[ConfigurationItem]) -> Optional[datetime]:
        """Рассчитать ожидаемую дату поставки"""

        expected_dates = []
        for item in items:
            if item.component.stock and
item.component.stock.expected_date:

expected_dates.append(item.component.stock.expected_date)

        if expected_dates:
            # Возвращаем самую позднюю дату
            return max(expected_dates)

        return None

```

```

def _get_missing_categories(self, items:
List[ConfigurationItem]) -> List[str]:
    """Получить список отсутствующих категорий"""

    # Основные категории для полной сборки ПК
    essential_categories = [
        "cpu", "motherboard", "ram", "storage",
        "gpu", "psu", "case"
    ]

    # Получаем категории, которые уже есть в конфигурации
    existing_categories = set()
    for item in items:
        existing_categories.add(item.component.category.slug)

    # Находим недостающие
    missing = []
    for category in essential_categories:
        if category not in existing_categories:
            # Получаем человекочитаемое название категории
            cat_obj = self.db.query(ComponentCategory).filter(
                ComponentCategory.slug == category
            ).first()
            if cat_obj:
                missing.append(cat_obj.name)

    return missing

```

Листинг Б.7 – Файл pdf_service.py

```

import os
import tempfile
import logging
from datetime import datetime
from typing import Optional
from reportlab.lib.pagesizes import letter, A4
from reportlab.platypus import SimpleDocTemplate, Paragraph,
Spacer, Table, TableStyle, PageBreak, Image
from reportlab.lib.styles import getSampleStyleSheet,
ParagraphStyle
from reportlab.lib.units import inch, cm
from reportlab.lib import colors
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont
from reportlab.lib.enums import TA_CENTER, TA_LEFT, TA_RIGHT
from jinja2 import Environment, FileSystemLoader
from ..schemas.configuration import ConfigurationExport
from ..config import settings

# Настройка логирования
logger = logging.getLogger(__name__)

class PDFService:
    """Сервис для генерации PDF отчетов"""

```

```

def __init__(self):
    # Настройка Jinja2 для шаблонов
    template_dir = os.path.join(os.path.dirname(__file__),
    '..', 'templates')
    self.jinja_env =
Environment(loader=FileSystemLoader(template_dir))

    # Регистрируем русские шрифты
    self._register_fonts()

def _register_fonts(self):
    """Регистрация шрифтов для поддержки русского языка"""
    try:
        # Пытаемся использовать системные шрифты
        font_paths = [
            '/System/Library/Fonts/Arial.ttf', # macOS
            '/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf',
Linux
            '/Windows/Fonts/arial.ttf', # Windows
            '/usr/share/fonts/TTF/DejaVuSans.ttf', # Arch
Linux
        ]

        font_registered = False
        for font_path in font_paths:
            if os.path.exists(font_path):
                try:

pdfmetrics.registerFont(TTFont('DejaVuSans', font_path))
                    font_registered = True
                    logger.info(f"Зарегистрирован шрифт:
{font_path}")

                    break
                except Exception as e:
                    logger.warning(f"Не удалось
зарегистрировать шрифт {font_path}: {e}")

            if not font_registered:
                logger.warning("Не удалось найти подходящий шрифт,
будет использован стандартный")

        except Exception as e:
            logger.error(f"Ошибка регистрации шрифтов: {e}")

    def generate_configuration_pdf(self, export_data:
ConfigurationExport) -> str:
        """Генерация PDF отчета конфигурации"""

        logger.info(f"Начинаю генерацию PDF для конфигурации
{export_data.configuration.id}")

        # Создаем временный файл для PDF

```

```

        pdf_filename =
f"config_{export_data.configuration.id}_{datetime.now().strftime('%Y%m%d_%H%M%S')}.pdf"
        pdf_path = os.path.join(settings.PDF_TEMP_PATH,
pdf_filename)

        # Создаем директорию если её нет
        os.makedirs(settings.PDF_TEMP_PATH, exist_ok=True)
        logger.info(f"Создана директория:
{settings.PDF_TEMP_PATH}")

    try:
        # Генерируем PDF с помощью reportlab
        logger.info("Создаю PDF с помощью reportlab")
        self._generate_modern_pdf(export_data, pdf_path)

        # Проверяем, что файл действительно создан
        if os.path.exists(pdf_path):
            file_size = os.path.getsize(pdf_path)
            logger.info(f"PDF файл создан, размер: {file_size}
байт")

            return pdf_path
        else:
            raise Exception("PDF файл не был создан")

    except Exception as e:
        # В случае ошибки генерируем HTML отчет с помощью
Jinja2 шаблона
        logger.error(f"Ошибка генерации PDF: {e}",
exc_info=True)

        # Создаем HTML файл с помощью Jinja2 шаблона
        html_filename =
f"config_{export_data.configuration.id}_{datetime.now().strftime('%Y%m%d_%H%M%S')}.html"
        html_path = os.path.join(settings.PDF_TEMP_PATH,
html_filename)

        logger.info("Создаю HTML файл с помощью Jinja2
шаблона")

        html_content =
self._generate_html_from_template(export_data)

        with open(html_path, 'w', encoding='utf-8') as f:
            f.write(html_content)

        logger.info(f"HTML файл создан: {html_path}")
        return html_path

    def _generate_html_from_template(self, export_data:
ConfigurationExport) -> str:
        """Генерация HTML отчета с помощью Jinja2 шаблона"""

```

```

config = export_data.configuration
compatibility = export_data.compatibility_check

# Подготавливаем данные для шаблона
components_total = sum((item.price_snapshot or
item.component.price) * item.quantity for item in config.items)
accessories_total = sum((acc.price_snapshot or
acc.component.price) * acc.quantity for acc in config.accessories)
if config.accessories else 0
total_price = components_total + accessories_total

# Группируем компоненты по категориям
components_by_category = {}
for item in config.items:
    category = item.component.category.name
    if category not in components_by_category:
        components_by_category[category] = []

    component_data = {
        'brand': item.component.brand,
        'name': item.component.name,
        'model': item.component.model,
        'quantity': item.quantity,
        'price': item.price_snapshot or
item.component.price,
        'total': (item.price_snapshot or
item.component.price) * item.quantity,
        'stock_status':
self._get_stock_status_text(item.component.stock.status if
item.component.stock else 'unknown'),
        'specifications':
self._format_specifications(item.component.specifications) if
item.component.specifications else None
    }

components_by_category[category].append(component_data)

# Подготавливаем данные аксессуаров
accessories = []
if config.accessories:
    for acc in config.accessories:
        accessories.append({
            'component': acc.component,
            'quantity': acc.quantity,
            'price_snapshot': acc.price_snapshot,
            'notes': acc.notes
        })

# Подготавливаем данные для шаблона
template_data = {
    'config': config,
    'export_date':
export_data.export_date.strftime('%d.%m.%Y %H:%M'),

```

```

        'components_total': components_total,
        'accessories_total': accessories_total,
        'total_price': total_price,
        'total_power': compatibility.total_power_consumption,
        'compatibility_status':
self._get_compatibility_status_text(compatibility.status),
        'components_by_category': components_by_category,
        'accessories': accessories,
        'compatibility_issues': compatibility.issues,
        'recommended_psu':
compatibility.recommended_psu_wattage,
        'notes': export_data.notes
    }

    # Рендерим шаблон
    template =
self.jinja_env.get_template('configuration_pdf.html')
    return template.render(**template_data)

def _get_compatibility_status_text(self, status: str) -> str:
    """Получить текстовое описание статуса совместимости"""
    status_map = {
        "compatible": "Совместимо",
        "incompatible": "Несовместимо",
        "warning": "Предупреждения",
        "unknown": "Неизвестно"
    }
    return status_map.get(status.lower(), "Неизвестно")

def _get_stock_status_text(self, status: str) -> str:
    """Преобразовать статус наличия в текст"""
    status_map = {
        "in_stock": "В наличии",
        "expected": "Ожидается",
        "out_of_stock": "Нет в наличии"
    }
    return status_map.get(status, "Неизвестно")

def _get_issue_type_text(self, issue_type: str) -> str:
    """Преобразовать тип проблемы в текст"""
    type_map = {
        "socket_mismatch": "Несовместимость сокетов",
        "memory_type_mismatch": "Несовместимость типа памяти",
        "memory_capacity_exceeded": "Превышение объема
памяти",
        "memory_slots_exceeded": "Превышение количества
слотов",
        "form_factor_mismatch": "Несовместимость форм-
фактора",
        "power_insufficient": "Недостаточная мощность БП",
        "power_warning": "Предупреждение о мощности БП"
    }
    return type_map.get(issue_type, issue_type)

```

```

def _get_severity_text(self, severity: str) -> str:
    """Преобразовать серьезность проблемы в текст"""
    severity_map = {
        "error": "Ошибка",
        "warning": "Предупреждение",
        "info": "Информация"
    }
    return severity_map.get(severity, severity)

def _format_specifications(self, specs: dict) -> str:
    """Форматировать технические характеристики"""
    if not specs:
        return ""

    formatted = []
    for key, value in specs.items():
        if isinstance(value, list):
            value = ", ".join(str(v) for v in value)
        formatted.append(f"{key}: {value}")

    return " | ".join(formatted)

def _generate_modern_pdf(self, export_data:
ConfigurationExport, pdf_path: str):
    """Генерация современного PDF отчета в стиле фронтенда"""

    config = export_data.configuration
    compatibility = export_data.compatibility_check

    # Создаем документ с отступами
    doc = SimpleDocTemplate(
        pdf_path,
        pagesize=A4,
        rightMargin=2*cm,
        leftMargin=2*cm,
        topMargin=2*cm,
        bottomMargin=2*cm
    )
    story = []

    # Определяем шрифт
    font_name = 'DejaVuSans' if 'DejaVuSans' in
pdfmetrics.getRegisteredFontNames() else 'Helvetica'

    # Создаем стили в современном минималистичном дизайне
    styles = self._create_modern_styles(font_name)

    # === ЗАГОЛОВОК ===
    story.append(Paragraph("КОНФИГУРАЦИЯ ПК",
styles['title']))
    story.append(Spacer(1, 0.3*cm))

```

```

        # Название конфигурации
        story.append(Paragraph(config.name,
styles['config_name']))
        if config.description:
            story.append(Spacer(1, 0.2*cm))
            story.append(Paragraph(config.description,
styles['description']))

        story.append(Spacer(1, 0.5*cm))

        # Дата экспорта
        export_date_str =
export_data.export_date.strftime('%d.%m.%Y в %H:%M')
        story.append(Paragraph(f"Дата создания отчета:
{export_date_str}", styles['meta']))

        story.append(Spacer(1, 1*cm))

        # === ОБЩАЯ ИНФОРМАЦИЯ ===
        components_total = sum((item.price_snapshot or
item.component.price) * item.quantity for item in config.items)
        accessories_total = sum((acc.price_snapshot or
acc.component.price) * acc.quantity for acc in config.accessories)
        if config.accessories else 0
        total_price = components_total + accessories_total

        # Создаем таблицу с общей информацией
        info_data = [
            ['СТОИМОСТЬ КОМПОНЕНТОВ', f'{components_total:,.0f}
₽'.replace(',', ' ')],
        ]

        if accessories_total > 0:
            info_data.append(['СТОИМОСТЬ АКССЕССУАРОВ',
f'{accessories_total:,.0f} ₽'.replace(',', ' ')])

        info_data.extend([
            ['ОБЩАЯ СТОИМОСТЬ', f'{total_price:,.0f}
₽'.replace(',', ' ')],
            ['СТАТУС СОВМЕСТИМОСТИ',
self._get_compatibility_status_text(compatibility.status)],
        ])

        if compatibility.total_power_consumption:
            info_data.append(['ЭНЕРГОПОТРЕБЛЕНИЕ',
f'{compatibility.total_power_consumption} Вт'])
        if compatibility.recommended_psu_wattage:
            info_data.append(['РЕКОМЕНДУЕМЫЙ БП',
f'{compatibility.recommended_psu_wattage} Вт'])

        info_table = Table(info_data, colWidths=[6*cm, 6*cm])
        info_table.setStyle(TableStyle([
            ('FONTNAME', (0, 0), (-1, -1), font_name),

```

```

        ('FONTSIZE', (0, 0), (-1, -1), 11),
        ('FONTNAME', (0, 0), (0, -1), font_name), # Жирный
для левой колонки
        ('TEXTCOLOR', (0, 0), (0, -1),
colors.HexColor('#374151')),
        ('TEXTCOLOR', (1, 0), (1, -1),
colors.HexColor('#1f2937')),
        ('ALIGN', (0, 0), (-1, -1), 'LEFT'),
        ('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),
        ('BOTTOPPADDING', (0, 0), (-1, -1), 8),
        ('TOPPADDING', (0, 0), (-1, -1), 8),
        ('LINEBELOW', (0, 0), (-1, 0), 2,
colors.HexColor('#2563eb')), # Синяя линия под заголовком
        # Выделяем строку с общей стоимостью
        ('BACKGROUND', (0, -4 if accessories_total > 0 else -
3), (-1, -4 if accessories_total > 0 else -3),
colors.HexColor('#e0f2fe')),
        ('TEXTCOLOR', (1, -4 if accessories_total > 0 else -
3), (1, -4 if accessories_total > 0 else -3),
colors.HexColor('#01579b')),
        ('FONTSIZE', (0, -4 if accessories_total > 0 else -3),
(-1, -4 if accessories_total > 0 else -3), 13),
    ]))

    story.append(info_table)
    story.append(Spacer(1, 1*cm))

    # === КОМПОНЕНТЫ ===
    story.append(Paragraph("КОМПОНЕНТЫ",
styles['section_header']))
    story.append(Spacer(1, 0.5*cm))

    # Группируем компоненты по категориям
    components_by_category = {}
    for item in config.items:
        category = item.component.category.name
        if category not in components_by_category:
            components_by_category[category] = []
        components_by_category[category].append(item)

    for category, items in components_by_category.items():
        # Заголовок категории
        story.append(Paragraph(category.upper(),
styles['category_header']))
        story.append(Spacer(1, 0.3*cm))

        # Компоненты в категории
        for item in items:
            price = item.price_snapshot or
item.component.price

            # Создаем таблицу для каждого компонента
            component_data = [

```

```

        [f"{item.component.brand}
{item.component.name}", f'{price:,.0f} ₽'.replace(',', ' ')],
        [f"Модель: {item.component.model}",
f"Количество: {item.quantity}"]
    ]

    component_table = Table(component_data,
colWidths=[10*cm, 4*cm])
    component_table.setStyle(TableStyle([
        ('FONTNAME', (0, 0), (-1, -1), font_name),
        ('FONTSIZE', (0, 0), (1, 0), 12), # Название
компонента
        ('FONTSIZE', (0, 1), (1, 1), 10), # Детали
        ('FONTNAME', (0, 0), (0, 0), font_name), #
Жирный для названия
        ('FONTNAME', (1, 0), (1, 0), font_name), #
Жирный для цены
        ('TEXTCOLOR', (0, 0), (0, 0),
colors.HexColor('#1f2937')),
        ('TEXTCOLOR', (1, 0), (1, 0),
colors.HexColor('#059669')), # Зеленый для цены
        ('TEXTCOLOR', (0, 1), (1, 1),
colors.HexColor('#6b7280')),
        ('ALIGN', (0, 0), (0, -1), 'LEFT'),
        ('ALIGN', (1, 0), (1, -1), 'RIGHT'),
        ('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),
        ('BOTTPADDING', (0, 0), (-1, -1), 6),
        ('TOPPADDING', (0, 0), (-1, -1), 6),
        ('BACKGROUND', (0, 0), (-1, -1),
colors.HexColor('#f9fafb')),
        ('BOX', (0, 0), (-1, -1), 1,
colors.HexColor('#e5e7eb')),
    ]))

    story.append(component_table)
    story.append(Spacer(1, 0.4*cm))

    # === АКЦЕССУАРЫ ===
    if config.accessories:
        story.append(Spacer(1, 0.5*cm))
        story.append(Paragraph("АКЦЕССУАРЫ",
styles['section_header']))
        story.append(Spacer(1, 0.5*cm))

        for accessory in config.accessories:
            price = accessory.price_snapshot or
accessory.component.price

            # Создаем таблицу для каждого аксессуара
            accessory_data = [
                [f"{accessory.component.brand}
{accessory.component.name}", f'{price:,.0f} ₽'.replace(',', ' ')],

```

```

        [f"Модель: {accessory.component.model}",
        f"Количество: {accessory.quantity}"]
    ]

    accessory_table = Table(accessory_data,
colWidths=[10*cm, 4*cm])
    accessory_table.setStyle(TableStyle([
        ('FONTNAME', (0, 0), (-1, -1), font_name),
        ('FONTSIZE', (0, 0), (1, 0), 12), # Название
аксессуара
        ('FONTSIZE', (0, 1), (1, 1), 10), # Детали
        ('FONTNAME', (0, 0), (0, 0), font_name), #
Жирный для названия
        ('FONTNAME', (1, 0), (1, 0), font_name), #
Жирный для цены
        ('TEXTCOLOR', (0, 0), (0, 0),
colors.HexColor('#1f2937')),
        ('TEXTCOLOR', (1, 0), (1, 0),
colors.HexColor('#059669')), # Зеленый для цены
        ('TEXTCOLOR', (0, 1), (1, 1),
colors.HexColor('#6b7280')),
        ('ALIGN', (0, 0), (0, -1), 'LEFT'),
        ('ALIGN', (1, 0), (1, -1), 'RIGHT'),
        ('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),
        ('BOTTPADDING', (0, 0), (-1, -1), 6),
        ('TOPPADDING', (0, 0), (-1, -1), 6),
        ('BACKGROUND', (0, 0), (-1, -1),
colors.HexColor('#f0f9ff')), # Слегка другой цвет для аксессуаров
        ('BOX', (0, 0), (-1, -1), 1,
colors.HexColor('#e5e7eb')),
    ]))

    story.append(accessory_table)
    story.append(Spacer(1, 0.4*cm))

    # === СОВМЕСТИМОСТЬ ===
    if compatibility.issues:
        story.append(Spacer(1, 0.5*cm))
        story.append(Paragraph("ПРОБЛЕМЫ СОВМЕСТИМОСТИ",
styles['section_header']))
        story.append(Spacer(1, 0.5*cm))

    for issue in compatibility.issues:
        # Определяем цвет по серьезности
        severity_colors = {
            'error': colors.HexColor('#dc2626'),
            'warning': colors.HexColor('#d97706'),
            'info': colors.HexColor('#2563eb')
        }
        severity_color =
severity_colors.get(issue.severity, colors.HexColor('#6b7280'))

```

```

        issue_text =
f"<b>{self._get_issue_type_text(issue.type)}</b><br/>{issue.message}"
        if issue.suggestions:
            issue_text += f"<br/><i>Рекомендации: {'',
'.join(issue.suggestions)}</i>"

        issue_para = Paragraph(issue_text,
styles['issue'])

        # Создаем таблицу для выделения проблемы
        issue_table = Table([[issue_para]],
colWidths=[14*cm])
        issue_table.setStyle(TableStyle([
            ('BACKGROUND', (0, 0), (-1, -1),
colors.HexColor('#fef2f2')),
            ('LEFTPADDING', (0, 0), (-1, -1), 12),
            ('RIGHTPADDING', (0, 0), (-1, -1), 12),
            ('TOPPADDING', (0, 0), (-1, -1), 10),
            ('BOTTOMPADDING', (0, 0), (-1, -1), 10),
            ('BOX', (0, 0), (-1, -1), 2, severity_color),
]))

        story.append(issue_table)
        story.append(Spacer(1, 0.3*cm))

        # === ФУТЕР ===
        story.append(Spacer(1, 1*cm))

        # Линия разделитель
        line_table = Table(['']*1, colWidths=[14*cm])
        line_table.setStyle(TableStyle([
            ('LINEABOVE', (0, 0), (-1, -1), 1,
colors.HexColor('#e5e7eb')),
]))
        story.append(line_table)
        story.append(Spacer(1, 0.5*cm))

        # Информация о системе
        footer_text = "Конфигурация создана с помощью веб-
конфигуратора ПК"
        if config.public_uuid:
            footer_text += f"<br/>UUID: {config.public_uuid}"

        story.append(Paragraph(footer_text, styles['footer']))

        # Генерируем PDF
        doc.build(story)

def _create_modern_styles(self, font_name: str):
    """Создание современных стилей для PDF"""
    styles = {}

```

```

# Заголовок документа
styles['title'] = ParagraphStyle(
    'Title',
    fontName=font_name,
    fontSize=24,
    textColor=colors.HexColor('#1f2937'),
    alignment=TA_CENTER,
    spaceAfter=0,
    spaceBefore=0,
)

# Название конфигурации
styles['config_name'] = ParagraphStyle(
    'ConfigName',
    fontName=font_name,
    fontSize=18,
    textColor=colors.HexColor('#2563eb'),
    alignment=TA_CENTER,
    spaceAfter=0,
    spaceBefore=0,
)

# Описание
styles['description'] = ParagraphStyle(
    'Description',
    fontName=font_name,
    fontSize=12,
    textColor=colors.HexColor('#6b7280'),
    alignment=TA_CENTER,
    spaceAfter=0,
    spaceBefore=0,
)

# Метаинформация
styles['meta'] = ParagraphStyle(
    'Meta',
    fontName=font_name,
    fontSize=10,
    textColor=colors.HexColor('#9ca3af'),
    alignment=TA_CENTER,
    spaceAfter=0,
    spaceBefore=0,
)

# Заголовки разделов
styles['section_header'] = ParagraphStyle(
    'SectionHeader',
    fontName=font_name,
    fontSize=16,
    textColor=colors.HexColor('#1f2937'),
    alignment=TA_LEFT,
    spaceAfter=0,
    spaceBefore=0,
)

```

```

    )

    # Заголовки категорий
    styles['category_header'] = ParagraphStyle(
        'CategoryHeader',
        fontName=font_name,
        fontSize=13,
        textColor=colors.HexColor('#374151'),
        alignment=TA_LEFT,
        spaceAfter=0,
        spaceBefore=0,
    )

    # Проблемы совместимости
    styles['issue'] = ParagraphStyle(
        'Issue',
        fontName=font_name,
        fontSize=11,
        textColor=colors.HexColor('#1f2937'),
        alignment=TA_LEFT,
        spaceAfter=0,
        spaceBefore=0,
    )

    # Футер
    styles['footer'] = ParagraphStyle(
        'Footer',
        fontName=font_name,
        fontSize=9,
        textColor=colors.HexColor('#9ca3af'),
        alignment=TA_CENTER,
        spaceAfter=0,
        spaceBefore=0,
    )

    return styles

```

Листинг Б.8 – Файл configuration.py

```

from pydantic import BaseModel, Field
from typing import Optional, List, Dict, Any
from datetime import datetime
from enum import Enum
from uuid import UUID
from .component import ComponentResponse

class ConfigurationStatus(str, Enum):
    DRAFT = "draft"
    COMPLETED = "completed"
    EXPORTED = "exported"

class CompatibilityStatus(str, Enum):
    COMPATIBLE = "compatible"
    INCOMPATIBLE = "incompatible"

```

```

WARNING = "warning"
UNKNOWN = "unknown"

class AvailabilityStatus(str, Enum):
    AVAILABLE = "available"
    PARTIAL = "partial"
    UNAVAILABLE = "unavailable"
    UNKNOWN = "unknown"

class ConfigurationCreate(BaseModel):
    name: str = Field(..., max_length=200)
    description: Optional[str] = None

class ConfigurationItemCreate(BaseModel):
    component_id: UUID
    quantity: int = Field(1, ge=1)
    notes: Optional[str] = None

class ConfigurationAccessoryCreate(BaseModel):
    component_id: UUID
    quantity: int = Field(1, ge=1)
    notes: Optional[str] = None

class ConfigurationItemResponse(BaseModel):
    id: UUID
    component: ComponentResponse
    quantity: int
    price_snapshot: Optional[float]
    notes: Optional[str]
    created_at: datetime

    class Config:
        from_attributes = True

class ConfigurationAccessoryResponse(BaseModel):
    id: UUID
    component: ComponentResponse
    quantity: int
    price_snapshot: Optional[float]
    notes: Optional[str]
    created_at: datetime

    class Config:
        from_attributes = True

class ConfigurationResponse(BaseModel):
    id: UUID
    name: str
    description: Optional[str]
    public_uuid: Optional[str]
    total_price: float
    total_power_consumption: Optional[int]
    compatibility_status: CompatibilityStatus

```

```

    compatibility_notes: Optional[str]
    status: ConfigurationStatus
    is_public: bool
    created_at: datetime
    updated_at: datetime
    items: List[ConfigurationItemResponse]
    accessories: List[ConfigurationAccessoryResponse] = []

    class Config:
        from_attributes = True

class CompatibilityIssue(BaseModel):
    """Проблема совместимости"""
    type: str # "socket_mismatch", "power_insufficient",
    "form_factor_conflict"
    severity: str # "error", "warning", "info"
    message: str
    component_ids: List[UUID]
    suggestions: Optional[List[str]] = None

class CompatibilityCheck(BaseModel):
    """Результат проверки совместимости"""
    is_compatible: bool
    status: CompatibilityStatus
    issues: List[CompatibilityIssue]
    total_power_consumption: Optional[int]
    recommended_psu_wattage: Optional[int]

class ConfigurationExport(BaseModel):
    """Данные для экспорта конфигурации в PDF"""
    configuration: ConfigurationResponse
    compatibility_check: CompatibilityCheck
    export_date: datetime
    notes: Optional[str] = None

```

Листинг Б.9 – Файл component.py

```

from pydantic import BaseModel, Field
from typing import Optional, Dict, Any, List
from datetime import datetime, date
from enum import Enum
from uuid import UUID

class StockStatus(str, Enum):
    IN_STOCK = "in_stock"
    EXPECTED = "expected"
    OUT_OF_STOCK = "out_of_stock"

class ComponentCategoryCreate(BaseModel):
    name: str = Field(..., max_length=100)
    slug: str = Field(..., max_length=50)
    description: Optional[str] = None
    order_priority: int = 0
    icon: Optional[str] = None

```

```

class ComponentCategoryResponse(BaseModel):
    id: UUID
    name: str
    slug: str
    description: Optional[str]
    order_priority: int
    icon: Optional[str]
    created_at: datetime
    updated_at: datetime

    class Config:
        from_attributes = True

class ComponentStockResponse(BaseModel):
    status: StockStatus
    quantity: int
    expected_date: Optional[date]
    updated_at: datetime

    class Config:
        from_attributes = True

class ComponentCreate(BaseModel):
    name: str = Field(..., max_length=200)
    brand: str = Field(..., max_length=100)
    model: str = Field(..., max_length=100)
    description: Optional[str] = None
    price: float = Field(..., gt=0)
    category_id: UUID
    specifications: Dict[str, Any] = {}
    form_factor: Optional[str] = None
    power_consumption: Optional[int] = None

class ComponentResponse(BaseModel):
    id: UUID
    name: str
    brand: str
    model: str
    description: Optional[str]
    price: float
    category: ComponentCategoryResponse
    specifications: Dict[str, Any]
    form_factor: Optional[str]
    power_consumption: Optional[int]
    created_at: datetime
    updated_at: datetime
    is_active: bool
    stock: Optional[ComponentStockResponse]

    class Config:
        from_attributes = True

```

```
class ComponentFilter(BaseModel):
```

Листинг Б.10 – Файл configurations.py

```
from fastapi import APIRouter, Depends, HTTPException, Response
from fastapi.responses import FileResponse
from sqlalchemy.orm import Session, joinedload
from typing import List
import uuid
from datetime import datetime
from uuid import UUID
from ..database import get_db
from ..models import Configuration, ConfigurationItem,
ConfigurationAccessory, Component
from ..schemas.configuration import (
    ConfigurationCreate, ConfigurationResponse,
    ConfigurationItemCreate, ConfigurationAccessoryCreate,
    CompatibilityCheck,
    ConfigurationExport
)
from ..services.compatibility_service import CompatibilityService
from ..services.pdf_service import PDFService

router = APIRouter()

@router.post("/configurations",
response_model=ConfigurationResponse)
async def create_configuration(
    config_data: ConfigurationCreate,
    db: Session = Depends(get_db)
):
    """Создать новую конфигурацию"""
    db_config = Configuration(
        name=config_data.name,
        description=config_data.description,
        public_uuid=str(uuid.uuid4())
    )

    db.add(db_config)
    db.commit()
    db.refresh(db_config)

    return db_config

@router.get("/configurations",
response_model=List[ConfigurationResponse])
async def get_configurations(
    skip: int = 0,
    limit: int = 100,
    db: Session = Depends(get_db)
):
    """Получить список конфигураций"""
    configs = db.query(Configuration).options(
```

```

joinedload(Configuration.items).joinedload(ConfigurationItem.component),

joinedload(Configuration.accessories).joinedload(ConfigurationAccessory.component)
    ).offset(skip).limit(limit).all()

    return configs

@router.get("/configurations/{config_id}",
response_model=ConfigurationResponse)
async def get_configuration(config_id: UUID, db: Session =
Depends(get_db)):
    """Получить конфигурацию по ID"""
    config = db.query(Configuration).options(

joinedload(Configuration.items).joinedload(ConfigurationItem.component)
).joinedload(Component.category),

joinedload(Configuration.items).joinedload(ConfigurationItem.component)
).joinedload(Component.stock),

joinedload(Configuration.accessories).joinedload(ConfigurationAccessory.component)
).joinedload(Component.category),

joinedload(Configuration.accessories).joinedload(ConfigurationAccessory.component)
).joinedload(Component.stock)
    ).filter(Configuration.id == config_id).first()

    if not config:
        raise HTTPException(status_code=404, detail="Конфигурация
не найдена")

    return config

@router.get("/configurations/uuid/{public_uuid}",
response_model=ConfigurationResponse)
async def get_configuration_by_uuid(public_uuid: str, db: Session
= Depends(get_db)):
    """Получить конфигурацию по публичному UUID"""
    config = db.query(Configuration).options(

joinedload(Configuration.items).joinedload(ConfigurationItem.component)
).joinedload(Component.category),

joinedload(Configuration.items).joinedload(ConfigurationItem.component)
).joinedload(Component.stock),

joinedload(Configuration.accessories).joinedload(ConfigurationAccessory.component)
).joinedload(Component.category),

```

```

joinedload(Configuration.accessories).joinedload(ConfigurationAccess
sory.component).joinedload(Component.stock)
    ).filter(Configuration.public_uuid == public_uuid).first()

    if not config:
        raise HTTPException(status_code=404, detail="Конфигурация
не найдена")

    return config

@router.post("/configurations/{config_id}/items")
async def add_component_to_configuration(
    config_id: UUID,
    item_data: ConfigurationItemCreate,
    db: Session = Depends(get_db)
):
    """Добавить компонент в конфигурацию"""

    # Проверяем существование конфигурации
    config = db.query(Configuration).filter(Configuration.id ==
config_id).first()
    if not config:
        raise HTTPException(status_code=404, detail="Конфигурация
не найдена")

    # Проверяем существование компонента с подробной информацией
    component = db.query(Component).filter(Component.id ==
item_data.component_id).first()
    if not component:
        # Получаем дополнительную информацию для отладки
        total_components = db.query(Component).count()
        active_components =
db.query(Component).filter(Component.is_active == True).count()

        raise HTTPException(
            status_code=404,
            detail={
                "message": "Компонент не найден",
                "requested_id": str(item_data.component_id),
                "total_components_in_db": total_components,
                "active_components_in_db": active_components
            }
        )

    # Проверяем, нет ли уже такого компонента в конфигурации
    existing_item = db.query(ConfigurationItem).filter(
        ConfigurationItem.configuration_id == config_id,
        ConfigurationItem.component_id == item_data.component_id
    ).first()

    if existing_item:
        # Обновляем количество

```

```

existing_item.quantity = item_data.quantity
existing_item.notes = item_data.notes
db.commit()
db.refresh(existing_item)

# Обновляем общую стоимость конфигурации
await _update_configuration_totals(config_id, db)

return {"message": "Компонент обновлен в конфигурации"}
else:
    # Создаем новый элемент
    db_item = ConfigurationItem(
        configuration_id=config_id,
        component_id=item_data.component_id,
        quantity=item_data.quantity,
        notes=item_data.notes,
        price_snapshot=component.price
    )

    db.add(db_item)
    db.commit()

    # Обновляем общую стоимость конфигурации
    await _update_configuration_totals(config_id, db)

    return {"message": "Компонент добавлен в конфигурацию"}

@router.post("/configurations/{config_id}/accessories")
async def add_accessory_to_configuration(
    config_id: UUID,
    accessory_data: ConfigurationAccessoryCreate,
    db: Session = Depends(get_db)
):
    """Добавить аксессуар в конфигурацию"""

    # Проверяем существование конфигурации
    config = db.query(Configuration).filter(Configuration.id ==
config_id).first()
    if not config:
        raise HTTPException(status_code=404, detail="Конфигурация
не найдена")

    # Проверяем существование компонента и что это аксессуар
    component = db.query(Component).options(
        joinedload(Component.category)
    ).filter(Component.id == accessory_data.component_id).first()

    if not component:
        raise HTTPException(status_code=404, detail="Компонент не
найден")

    if component.category.slug != "accessories":

```

```

        raise HTTPException(status_code=400, detail="Компонент не
является аксессуаром")

    # Проверяем, нет ли уже такого аксессуара в конфигурации
    existing_accessory = db.query(ConfigurationAccessory).filter(
        ConfigurationAccessory.configuration_id == config_id,
        ConfigurationAccessory.component_id ==
accessory_data.component_id
    ).first()

    if existing_accessory:
        # Обновляем количество
        existing_accessory.quantity = accessory_data.quantity
        existing_accessory.notes = accessory_data.notes
        db.commit()
        db.refresh(existing_accessory)

        return {"message": "Аксессуар обновлен в конфигурации"}
    else:
        # Создаем новый аксессуар
        db_accessory = ConfigurationAccessory(
            configuration_id=config_id,
            component_id=accessory_data.component_id,
            quantity=accessory_data.quantity,
            notes=accessory_data.notes,
            price_snapshot=component.price
        )

        db.add(db_accessory)
        db.commit()

        return {"message": "Аксессуар добавлен в конфигурацию"}

@router.delete("/configurations/{config_id}/accessories/{accessory
_id}")
async def remove_accessory_from_configuration(
    config_id: UUID,
    accessory_id: UUID,
    db: Session = Depends(get_db)
):
    """Удалить аксессуар из конфигурации"""

    accessory = db.query(ConfigurationAccessory).filter(
        ConfigurationAccessory.id == accessory_id,
        ConfigurationAccessory.configuration_id == config_id
    ).first()

    if not accessory:
        raise HTTPException(status_code=404, detail="Аксессуар не
найден в конфигурации")

    db.delete(accessory)
    db.commit()

```

```

        return {"message": "Аксессуар удален из конфигурации"}

@router.delete("/configurations/{config_id}/items/{item_id}")
async def remove_component_from_configuration(
    config_id: UUID,
    item_id: UUID,
    db: Session = Depends(get_db)
):
    """Удалить компонент из конфигурации"""

    item = db.query(ConfigurationItem).filter(
        ConfigurationItem.id == item_id,
        ConfigurationItem.configuration_id == config_id
    ).first()

    if not item:
        raise HTTPException(status_code=404, detail="Элемент
конфигурации не найден")

    db.delete(item)
    db.commit()

    # Обновляем общую стоимость конфигурации
    await _update_configuration_totals(config_id, db)

    return {"message": "Компонент удален из конфигурации"}

@router.post("/configurations/{config_id}/check-compatibility",
response_model=CompatibilityCheck)
async def check_configuration_compatibility(config_id: UUID, db:
Session = Depends(get_db)):
    """Проверить совместимость конфигурации"""

    # Получаем все компоненты конфигурации
    config_items = db.query(ConfigurationItem).filter(
        ConfigurationItem.configuration_id == config_id
    ).all()

    if not config_items:
        raise HTTPException(status_code=400, detail="Конфигурация
пуста")

    component_ids = [item.component_id for item in config_items]

    compatibility_service = CompatibilityService(db)
    result =
compatibility_service.check_configuration_compatibility(component_
ids)

    # Обновляем статус совместимости в конфигурации
    config = db.query(Configuration).filter(Configuration.id ==
config_id).first()

```

```

    if config:
        config.compatibility_status = result.status.value
        config.compatibility_issues = [issue.dict() for issue in
result.issues]
        db.commit()

    return result

@router.get("/configurations/{config_id}/export/pdf")
async def export_configuration_pdf(config_id: UUID, db: Session =
Depends(get_db)):
    """Экспортировать конфигурацию в PDF"""

    # Получаем конфигурацию с аксессуарами
    config = db.query(Configuration).options(

joinedload(Configuration.items).joinedload(ConfigurationItem.compo
nent).joinedload(Component.category),

joinedload(Configuration.items).joinedload(ConfigurationItem.compo
nent).joinedload(Component.stock),

joinedload(Configuration.accessories).joinedload(ConfigurationAcce
ssory.component).joinedload(Component.category),

joinedload(Configuration.accessories).joinedload(ConfigurationAcce
ssory.component).joinedload(Component.stock)
    ).filter(Configuration.id == config_id).first()

    if not config:
        raise HTTPException(status_code=404, detail="Конфигурация
не найдена")

    if not config.items:
        raise HTTPException(status_code=400, detail="Конфигурация
пуста")

    # Проверяем совместимость
    component_ids = [item.component_id for item in config.items]
    compatibility_service = CompatibilityService(db)
    compatibility_check =
compatibility_service.check_configuration_compatibility(component_
ids)

    # Создаем данные для экспорта
    export_data = ConfigurationExport(
        configuration=config,
        compatibility_check=compatibility_check,
        export_date=datetime.now(),
        notes="Конфигурация создана с помощью веб-конфигуратора
ПК"
    )

```

```

    # Генерируем PDF файл
    pdf_service = PDFService()
    file_path =
pdf_service.generate_configuration_pdf(export_data)

    # Обновляем статус конфигурации
    config.status = "exported"
    db.commit()

    # Определяем media_type и имя файла на основе расширения
    if file_path.endswith('.pdf'):
        media_type = "application/pdf"
        filename = f"конфигурация_{config.name}.pdf"
    else:
        media_type = "text/html"
        filename = f"конфигурация_{config.name}.html"

    return FileResponse(
        file_path,
        media_type=media_type,
        filename=filename
    )

@router.put("/configurations/{config_id}",
response_model=ConfigurationResponse)
async def update_configuration(
    config_id: UUID,
    config_data: ConfigurationCreate,
    db: Session = Depends(get_db)
):
    """Обновить конфигурацию"""

    config = db.query(Configuration).filter(Configuration.id ==
config_id).first()
    if not config:
        raise HTTPException(status_code=404, detail="Конфигурация
не найдена")

    config.name = config_data.name
    config.description = config_data.description
    config.updated_at = datetime.now()

    db.commit()
    db.refresh(config)

    return config

@router.delete("/configurations/{config_id}")
async def delete_configuration(config_id: UUID, db: Session =
Depends(get_db)):
    """Удалить конфигурацию"""

```

```

        config = db.query(Configuration).filter(Configuration.id ==
config_id).first()
        if not config:
            raise HTTPException(status_code=404, detail="Конфигурация
не найдена")

        db.delete(config)
        db.commit()

        return {"message": "Конфигурация удалена"}

async def _update_configuration_totals(config_id: UUID, db:
Session):
    """Обновить общие показатели конфигурации"""

    config = db.query(Configuration).filter(Configuration.id ==
config_id).first()
    if not config:
        return

    # Получаем все элементы конфигурации
    items = db.query(ConfigurationItem).options(
joinedload(ConfigurationItem.component).joinedload(Component.stock
)
    ).filter(ConfigurationItem.configuration_id ==
config_id).all()

    # Считаем общую стоимость
    total_price = sum(
        (item.price_snapshot or item.component.price) *
item.quantity
        for item in items
    )

    # Определяем статус наличия
    availability_statuses = []
    for item in items:
        if item.component.stock:

availability_statuses.append(item.component.stock.status)
        else:
            availability_statuses.append("out_of_stock")

    if all(status == "in_stock" for status in
availability_statuses):
        availability_status = "available"
    elif any(status == "in_stock" for status in
availability_statuses):
        availability_status = "partial"
    else:
        availability_status = "unavailable"

```

```
# Обновляем конфигурацию
config.total_price = total_price
config.availability_status = availability_status
config.updated_at = datetime.now()

db.commit()
```

Листинг Б.11 – Файл components.py

```
from fastapi import APIRouter, Depends, HTTPException, Query
from sqlalchemy.orm import Session, joinedload
from sqlalchemy import and_, or_
from typing import List, Optional
from ..database import get_db
from ..models import Component, ComponentCategory, ComponentStock
from ..schemas.component import ComponentResponse, ComponentFilter
from ..schemas.configuration import CompatibilityCheck
from ..services.compatibility_service import CompatibilityService
import uuid

router = APIRouter()

@router.get("/components", response_model=List[ComponentResponse])
async def get_components(
    category_slug: Optional[str] = Query(None, description="Фильтр по категории"),
    brand: Optional[List[str]] = Query(None, description="Фильтр по брендам"),
    price_min: Optional[float] = Query(None, description="Минимальная цена"),
    price_max: Optional[float] = Query(None, description="Максимальная цена"),
    only_in_stock: bool = Query(False, description="Только товары в наличии"),
    form_factor: Optional[List[str]] = Query(None, description="Фильтр по форм-фактору"),
    power_max: Optional[int] = Query(None, description="Максимальное энергопотребление"),
    search: Optional[str] = Query(None, description="Поиск по названию/модели"),
    socket: Optional[List[str]] = Query(None, description="Фильтр по сокету"),
    memory_type: Optional[List[str]] = Query(None, description="Фильтр по типу памяти"),
    interface: Optional[List[str]] = Query(None, description="Фильтр по интерфейсу"),
    page: int = Query(1, ge=1, description="Номер страницы"),
    limit: int = Query(20, ge=1, le=100, description="Количество на странице"),
    db: Session = Depends(get_db)
):
    """Получить компоненты с фильтрацией"""

    query = db.query(Component).options(
```

```

        joinedload(Component.category),
        joinedload(Component.stock)
    ).filter(Component.is_active == True)

    # Фильтр по категории
    if category_slug:
        query =
query.join(ComponentCategory).filter(ComponentCategory.slug ==
category_slug)

    # Фильтр по бренду
    if brand:
        query = query.filter(Component.brand.in_(brand))

    # Фильтр по цене
    if price_min is not None:
        query = query.filter(Component.price >= price_min)
    if price_max is not None:
        query = query.filter(Component.price <= price_max)

    # Фильтр по наличию
    if only_in_stock:
        query =
query.join(ComponentStock).filter(ComponentStock.status ==
"in_stock")

    # Фильтр по форм-фактору
    if form_factor:
        query =
query.filter(Component.form_factor.in_(form_factor))

    # Фильтр по энергопотреблению
    if power_max is not None:
        query = query.filter(
            or_(
                Component.power_consumption <= power_max,
                Component.power_consumption.is_(None)
            )
        )

    # Поиск по тексту
    if search:
        search_filter = or_(
            Component.name.ilike(f"%{search}%"),
            Component.model.ilike(f"%{search}%"),
            Component.brand.ilike(f"%{search}%")
        )
        query = query.filter(search_filter)

    # Фильтр по сокету (для процессоров/материнок)
    if socket:

```

```

        query =
query.filter(Component.specifications["socket"].astext.in_(socket)
)

        # Фильтр по типу памяти
        if memory_type:
            query =
query.filter(Component.specifications["memory_type"].astext.in_(me
memory_type))

        # Фильтр по интерфейсу
        if interface:
            query =
query.filter(Component.specifications["interface"].astext.in_(inte
rface))

        # Пагинация
        offset = (page - 1) * limit
        components = query.offset(offset).limit(limit).all()

        return components

@router.get("/components/{component_id}",
response_model=ComponentResponse)
async def get_component(component_id: str, db: Session =
Depends(get_db)):
    """Получить компонент по ID"""
    component = db.query(Component).options(
        joinedload(Component.category),
        joinedload(Component.stock)
    ).filter(Component.id == component_id).first()

    if not component:
        raise HTTPException(status_code=404, detail="Компонент не
найден")

    return component

@router.get("/components/category/{category_slug}",
response_model=List[ComponentResponse])
async def get_components_by_category(
    category_slug: str,
    brand: Optional[List[str]] = Query(None, description="Фильтр
по брендам"),
    price_min: Optional[float] = Query(None,
description="Минимальная цена"),
    price_max: Optional[float] = Query(None,
description="Максимальная цена"),
    only_in_stock: bool = Query(False, description="Только товары
в наличии"),
    form_factor: Optional[List[str]] = Query(None,
description="Фильтр по форм-фактору"),

```

```

        power_max: Optional[int] = Query(None,
description="Максимальное энергопотребление"),
        search: Optional[str] = Query(None, description="Поиск по
названию/модели"),
        socket: Optional[List[str]] = Query(None, description="Фильтр
по сокету"),
        memory_type: Optional[List[str]] = Query(None,
description="Фильтр по типу памяти"),
        interface: Optional[List[str]] = Query(None,
description="Фильтр по интерфейсу"),
        page: int = Query(1, ge=1, description="Номер страницы"),
        limit: int = Query(20, ge=1, le=100, description="Количество
на странице"),
        compatible_with: Optional[List[str]] = Query(None,
description="ID компонентов для проверки совместимости"),
        db: Session = Depends(get_db)
):
    """Получить компоненты категории с фильтрацией и поиском"""

    # Проверяем существование категории
    category =
db.query(ComponentCategory).filter(ComponentCategory.slug ==
category_slug).first()
    if not category:
        raise HTTPException(status_code=404, detail="Категория не
найдена")

    query = db.query(Component).options(
        joinedload(Component.category),
        joinedload(Component.stock)
    ).filter(
        Component.category_id == category.id,
        Component.is_active == True
    )

    # Фильтр по бренду
    if brand:
        query = query.filter(Component.brand.in_(brand))

    # Фильтр по цене
    if price_min is not None:
        query = query.filter(Component.price >= price_min)
    if price_max is not None:
        query = query.filter(Component.price <= price_max)

    # Фильтр по наличию
    if only_in_stock:
        query =
query.join(ComponentStock).filter(ComponentStock.status ==
"in_stock")

    # Фильтр по форм-фактору
    if form_factor:

```

```

        query =
query.filter(Component.form_factor.in_(form_factor))

# Фильтр по энергопотреблению
if power_max is not None:
    query = query.filter(
        or_(
            Component.power_consumption <= power_max,
            Component.power_consumption.is_(None)
        )
    )

# Поиск по тексту
if search:
    search_filter = or_(
        Component.name.ilike(f"%{search}%"),
        Component.model.ilike(f"%{search}%"),
        Component.brand.ilike(f"%{search}%")
    )
    query = query.filter(search_filter)

# Фильтр по сокету (для процессоров/материнок)
if socket:
    query =
query.filter(Component.specifications["socket"].astext.in_(socket)
)

# Фильтр по типу памяти
if memory_type:
    query =
query.filter(Component.specifications["memory_type"].astext.in_(me
memory_type))

# Фильтр по интерфейсу
if interface:
    query =
query.filter(Component.specifications["interface"].astext.in_(inte
rface))

# Пагинация
offset = (page - 1) * limit
components = query.offset(offset).limit(limit).all()

# Если нужна проверка совместимости
if compatible_with:
    compatibility_service = CompatibilityService(db)
    compatible_components = []

    for component in components:
        test_config = compatible_with + [str(component.id)]
        compatibility =
compatibility_service.check_configuration_compatibility(test_conf
ig)

```

```

        # Добавляем только совместимые компоненты или с
        предупреждениями
        if compatibility.status in ["compatible", "warning"]:
            compatible_components.append(component)

        components = compatible_components

    return components

@router.post("/components/check-compatibility",
response_model=CompatibilityCheck)
async def check_components_compatibility(
    component_ids: List[str],
    db: Session = Depends(get_db)
):
    """Проверить совместимость набора компонентов"""
    if not component_ids:
        raise HTTPException(status_code=400, detail="Список
компонентов не может быть пустым")

    # Валидация UUID формата
    invalid_uuids = []
    for comp_id in component_ids:
        try:
            uuid.UUID(comp_id)
        except ValueError:
            invalid_uuids.append(comp_id)

    if invalid_uuids:
        raise HTTPException(
            status_code=400,
            detail=f"Некорректный формат UUID: {invalid_uuids}"
        )

    # Проверяем существование всех компонентов
    existing_components =
db.query(Component.id).filter(Component.id.in_(component_ids)).all
()
    existing_ids = {str(comp.id) for comp in existing_components}
    missing_ids = set(component_ids) - existing_ids

    if missing_ids:
        # Получаем информацию о существующих компонентах для
отладки
        existing_count = len(existing_ids)
        total_components = db.query(Component).count()

        raise HTTPException(
            status_code=404,
            detail={
                "message": "Компоненты не найдены",
                "missing_ids": list(missing_ids),

```

```

        "existing_count": existing_count,
        "total_components_in_db": total_components,
        "requested_ids": component_ids
    }
)

compatibility_service = CompatibilityService(db)
result =
compatibility_service.check_configuration_compatibility(component_
ids)

return result

@router.get("/components/debug/ids")
async def get_all_component_ids(db: Session = Depends(get_db)):
    """Получить все ID компонентов для отладки"""
    components = db.query(Component.id, Component.name,
Component.brand, Component.is_active).all()

    return {
        "total_count": len(components),
        "active_count": len([c for c in components if
c.is_active]),
        "components": [
            {
                "id": str(comp.id),
                "name": comp.name,
                "brand": comp.brand,
                "is_active": comp.is_active
            }
            for comp in components
        ]
    }

@router.get("/components/filters/options")
async def get_filter_options(
    category_slug: Optional[str] = Query(None,
description="Категория для фильтров"),
    db: Session = Depends(get_db)
):
    """Получить доступные варианты для фильтров"""

    query = db.query(Component).filter(Component.is_active ==
True)

    if category_slug:
        query =
query.join(ComponentCategory).filter(ComponentCategory.slug ==
category_slug)

    components = query.all()

    # Собираем уникальные значения для фильтров

```

```

    brands = list(set(comp.brand for comp in components if
comp.brand))
    form_factors = list(set(comp.form_factor for comp in
components if comp.form_factor))

    # Специфичные фильтры из specifications
    sockets = set()
    memory_types = set()
    interfaces = set()

    for comp in components:
        specs = comp.specifications or {}

        if "socket" in specs:
            sockets.add(specs["socket"])
        if "memory_type" in specs:
            if isinstance(specs["memory_type"], list):
                memory_types.update(specs["memory_type"])
            else:
                memory_types.add(specs["memory_type"])
        if "interface" in specs:
            interfaces.add(specs["interface"])

    return {
        "brands": sorted(brands),
        "form_factors": sorted(form_factors),
        "sockets": sorted(list(sockets)),
        "memory_types": sorted(list(memory_types)),
        "interfaces": sorted(list(interfaces)),
        "price_range": {
            "min": min((comp.price for comp in components),
default=0),
            "max": max((comp.price for comp in components),
default=0)
        }
    }

```

Листинг Б.12 – Файл categories.py

```

from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.orm import Session
from typing import List
from ..database import get_db
from ..models import ComponentCategory
from ..schemas.component import ComponentCategoryCreate,
ComponentCategoryResponse

router = APIRouter()

@router.get("/categories",
response_model=List[ComponentCategoryResponse])
async def get_categories(db: Session = Depends(get_db)):
    """Получить все категории компонентов"""

```

```

        categories =
db.query(ComponentCategory).order_by(ComponentCategory.order_prior
ity).all()
        return categories

@router.get("/categories/{category_id}",
response_model=ComponentCategoryResponse)
async def get_category(category_id: int, db: Session =
Depends(get_db)):
    """Получить категорию по ID"""
    category =
db.query(ComponentCategory).filter(ComponentCategory.id ==
category_id).first()
    if not category:
        raise HTTPException(status_code=404, detail="Категория не
найдена")
    return category

@router.get("/categories/slug/{slug}",
response_model=ComponentCategoryResponse)
async def get_category_by_slug(slug: str, db: Session =
Depends(get_db)):
    """Получить категорию по slug"""
    category =
db.query(ComponentCategory).filter(ComponentCategory.slug ==
slug).first()
    if not category:
        raise HTTPException(status_code=404, detail="Категория не
найдена")
    return category

@router.post("/categories",
response_model=ComponentCategoryResponse)
async def create_category(category: ComponentCategoryCreate, db:
Session = Depends(get_db)):
    """Создать новую категорию"""
    # Проверяем уникальность slug
    existing =
db.query(ComponentCategory).filter(ComponentCategory.slug ==
category.slug).first()
    if existing:
        raise HTTPException(status_code=400, detail="Категория с
таким slug уже существует")

    db_category = ComponentCategory(**category.dict())
    db.add(db_category)
    db.commit()
    db.refresh(db_category)
    return db_category

```

Листинг Б.13 – Файл accessories.py

```

from fastapi import APIRouter, Depends, HTTPException, Query
from sqlalchemy.orm import Session, joinedload

```

```

from sqlalchemy import and_, or_
from typing import List, Optional
from ..database import get_db
from ..models import Component, ComponentCategory, ComponentStock
from ..schemas.component import ComponentResponse, ComponentFilter

router = APIRouter()

@router.get("/accessories",
response_model=List[ComponentResponse])
async def get_accessories(
    type_filter: Optional[List[str]] = Query(None,
description="Фильтр по типу аксессуара"),
    brand: Optional[List[str]] = Query(None, description="Фильтр
по брендам"),
    price_min: Optional[float] = Query(None,
description="Минимальная цена"),
    price_max: Optional[float] = Query(None,
description="Максимальная цена"),
    only_in_stock: bool = Query(False, description="Только товары
в наличии"),
    connection: Optional[List[str]] = Query(None,
description="Фильтр по типу подключения"),
    search: Optional[str] = Query(None, description="Поиск по
названию/модели"),
    page: int = Query(1, ge=1, description="Номер страницы"),
    limit: int = Query(20, ge=1, le=100, description="Количество
на странице"),
    db: Session = Depends(get_db)
):
    """Получить аксессуары с фильтрацией"""

    # Получаем категорию аксессуаров
    accessories_category = db.query(ComponentCategory).filter(
        ComponentCategory.slug == "accessories"
    ).first()

    if not accessories_category:
        raise HTTPException(status_code=404, detail="Категория
аксессуаров не найдена")

    query = db.query(Component).options(
        joinedload(Component.category),
        joinedload(Component.stock)
    ).filter(
        Component.category_id == accessories_category.id,
        Component.is_active == True
    )

    # Фильтр по типу аксессуара
    if type_filter:

```

```

        query =
query.filter(Component.specifications["type"].astext.in_(type_filt
er))

    # Фильтр по бренду
    if brand:
        query = query.filter(Component.brand.in_(brand))

    # Фильтр по цене
    if price_min is not None:
        query = query.filter(Component.price >= price_min)
    if price_max is not None:
        query = query.filter(Component.price <= price_max)

    # Фильтр по наличию
    if only_in_stock:
        query =
query.join(ComponentStock).filter(ComponentStock.status ==
"in_stock")

    # Фильтр по типу подключения
    if connection:
        query =
query.filter(Component.specifications["connection"].astext.in_(con
nection))

    # Поиск по тексту
    if search:
        search_filter = or_(
            Component.name.ilike(f"%{search}%"),
            Component.model.ilike(f"%{search}%"),
            Component.brand.ilike(f"%{search}%")
        )
        query = query.filter(search_filter)

    # Пагинация
    offset = (page - 1) * limit
    accessories = query.offset(offset).limit(limit).all()

    return accessories

@router.get("/accessories/{accessory_id}",
response_model=ComponentResponse)
async def get_accessory(accessory_id: str, db: Session =
Depends(get_db)):
    """Получить аксессуар по ID"""
    accessory = db.query(Component).options(
        joinedload(Component.category),
        joinedload(Component.stock)
    ).filter(Component.id == accessory_id).first()

    if not accessory:

```

```

        raise HTTPException(status_code=404, detail="Аксессуар не
найден")

    # Проверяем, что это действительно аксессуар
    if accessory.category.slug != "accessories":
        raise HTTPException(status_code=400, detail="Компонент не
является аксессуаром")

    return accessory

@router.get("/accessories/filters/options")
async def get_accessories_filter_options(db: Session =
Depends(get_db)):
    """Получить доступные опции для фильтров аксессуаров"""

    # Получаем категорию аксессуаров
    accessories_category = db.query(ComponentCategory).filter(
        ComponentCategory.slug == "accessories"
    ).first()

    if not accessories_category:
        raise HTTPException(status_code=404, detail="Категория
аксессуаров не найдена")

    # Получаем все аксессуары для анализа
    accessories = db.query(Component).filter(
        Component.category_id == accessories_category.id,
        Component.is_active == True
    ).all()

    # Собираем уникальные значения для фильтров
    brands = list(set(acc.brand for acc in accessories))
    types = list(set(acc.specifications.get("type") for acc in
accessories if acc.specifications.get("type")))
    connections = list(set(acc.specifications.get("connection")
for acc in accessories if acc.specifications.get("connection")))

    # Ценовой диапазон
    prices = [acc.price for acc in accessories]
    price_range = {
        "min": min(prices) if prices else 0,
        "max": max(prices) if prices else 0
    }

    return {
        "brands": sorted(brands),
        "types": sorted(types),
        "connections": sorted(connections),
        "price_range": price_range
    }

@router.get("/accessories/categories")

```

```

async def get_accessories_categories(db: Session =
Depends(get_db)):
    """Получить категории аксессуаров (типы)"""

    # Получаем категорию аксессуаров
    accessories_category = db.query(ComponentCategory).filter(
        ComponentCategory.slug == "accessories"
    ).first()

    if not accessories_category:
        raise HTTPException(status_code=404, detail="Категория
аксессуаров не найдена")

    # Получаем все типы аксессуаров
    accessories = db.query(Component).filter(
        Component.category_id == accessories_category.id,
        Component.is_active == True
    ).all()

    # Группируем по типам
    type_counts = {}
    for acc in accessories:
        acc_type = acc.specifications.get("type")
        if acc_type:
            type_counts[acc_type] = type_counts.get(acc_type, 0) +
1

    # Создаем список категорий с русскими названиями
    type_names = {
        "mouse": "Мыши",
        "keyboard": "Клавиатуры",
        "monitor": "Мониторы",
        "headset": "Гарнитуры",
        "headphones": "Наушники",
        "webcam": "Веб-камеры",
        "mousepad": "Коврики для мыши"
    }

    categories = []
    for acc_type, count in type_counts.items():
        categories.append({
            "slug": acc_type,
            "name": type_names.get(acc_type, acc_type.title()),
            "count": count
        })

    return sorted(categories, key=lambda x: x["name"])
}

```

Приложение В
(обязательное)
Исходный код web-конфигуратора

Листинг В.1 – Файл AccessorySelector.tsx

```
import React, { useState, useEffect } from 'react';
import { motion, AnimatePresence } from 'framer-motion';
import { X, Plus, Minus, ShoppingCart } from 'lucide-react';
import { Component } from '../types';
import { formatPrice } from '../lib/api';

interface AccessorySelectorProps {
  isOpen: boolean;
  onClose: () => void;
  onConfirm: (selectedAccessories: SelectedAccessory[]) => void;
}

interface SelectedAccessory {
  component: Component;
  quantity: number;
}

export default function AccessorySelector({ isOpen, onClose,
onConfirm }: AccessorySelectorProps) {
  const [accessories, setAccessories] = useState<Component[]>([]);
  const [selectedAccessories, setSelectedAccessories] =
useState<SelectedAccessory[]>([]);
  const [loading, setLoading] = useState(false);
  const [searchTerm, setSearchTerm] = useState('');
  const [selectedCategory, setSelectedCategory] =
useState<string>('all');

  useEffect(() => {
    if (isOpen) {
      loadAccessories();
    }
  }, [isOpen]);

  const loadAccessories = async () => {
    setLoading(true);
    try {
      const response = await
fetch(`${process.env.NEXT_PUBLIC_API_URL} ||
'http://localhost:8000'/accessories?limit=50`);
      const data = await response.json();
      setAccessories(data);
    } catch (error) {
      console.error('Ошибка загрузки аксессуаров:', error);
    } finally {
      setLoading(false);
    }
  };

  const filteredAccessories = accessories.filter(accessory => {
    const matchesSearch =
accessory.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
```

```

    accessory.brand.toLowerCase().includes(searchTerm.toLowerCase());
    const matchesCategory = selectedCategory === 'all' ||
        accessory.specifications?.type ===
selectedCategory;
    return matchesSearch && matchesCategory;
});

const categories = [
    { value: 'all', label: 'Все категории' },
    { value: 'mouse', label: 'Мыши' },
    { value: 'keyboard', label: 'Клавиатуры' },
    { value: 'monitor', label: 'Мониторы' },
    { value: 'headset', label: 'Гарнитуры' },
    { value: 'webcam', label: 'Веб-камеры' },
    { value: 'mousepad', label: 'Коврики' },
];

const addAccessory = (accessory: Component) => {
    const existing = selectedAccessories.find(item =>
item.component.id === accessory.id);
    if (existing) {
        setSelectedAccessories(prev =>
            prev.map(item =>
                item.component.id === accessory.id
                    ? { ...item, quantity: item.quantity + 1 }
                    : item
            )
        );
    } else {
        setSelectedAccessories(prev => [...prev, { component:
accessory, quantity: 1 }]);
    }
};

const removeAccessory = (accessoryId: string) => {
    setSelectedAccessories(prev => prev.filter(item =>
item.component.id !== accessoryId));
};

const updateQuantity = (accessoryId: string, quantity: number)
=> {
    if (quantity <= 0) {
        removeAccessory(accessoryId);
        return;
    }

    setSelectedAccessories(prev =>
        prev.map(item =>
            item.component.id === accessoryId
                ? { ...item, quantity }
                : item
        )
    );
};

```

```

    );
  };

  const totalPrice = selectedAccessories.reduce(
    (sum, item) => sum + (item.component.price * item.quantity),
    0
  );

  const handleConfirm = () => {
    onConfirm(selectedAccessories);
    onClose();
  };

  if (!isOpen) return null;

  return (
    <AnimatePresence>
      <motion.div
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        exit={{ opacity: 0 }}
        className="fixed inset-0 bg-black bg-opacity-50 flex
items-center justify-center z-50 p-4"
        onClick={onClose}
      >
        <motion.div
          initial={{ scale: 0.9, opacity: 0 }}
          animate={{ scale: 1, opacity: 1 }}
          exit={{ scale: 0.9, opacity: 0 }}
          className="bg-white rounded-2xl shadow-xl max-w-6xl w-
full max-h-[90vh] overflow-hidden"
          onClick={(e) => e.stopPropagation()}
        >
          {/* Заголовок */}
          <div className="p-6 border-b bg-gradient-to-r from-blue-
600 to-purple-600 text-white">
            <div className="flex items-center justify-between">
              <div>
                <h2 className="text-2xl font-bold">Выберите
аксессуары</h2>
                <p className="text-blue-100 mt-1">Дополните свою
конфигурацию полезными аксессуарами</p>
              </div>
              <button
                onClick={onClose}
                className="p-2 hover:bg-white/20 rounded-lg
transition-colors"
              >
                <X className="w-6 h-6" />
              </button>
            </div>
          </div>
        </motion.div>
      </AnimatePresence>
    );
  };

```

```

<div className="flex h-[calc(90vh-200px)]">
  /* Левая панель - каталог аксессуаров */
  <div className="flex-1 p-6 overflow-y-auto">
    /* Фильтры */
    <div className="mb-6 space-y-4">
      <input
        type="text"
        placeholder="Поиск аксессуаров..."
        value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)}
        className="input w-full"
      />

      <select
        value={selectedCategory}
        onChange={(e) =>
setSelectedCategory(e.target.value)}
        className="input w-full"
      >
        {categories.map(category => (
          <option key={category.value}
value={category.value}>
            {category.label}
          </option>
        ))}
      </select>
    </div>

    /* Список аксессуаров */
    {loading ? (
      <div className="text-center py-8">
        <div className="animate-spin rounded-full h-8 w-
8 border-b-2 border-blue-600 mx-auto"></div>
        <p className="mt-2 text-gray-600">Загрузка
аксессуаров...</p>
      </div>
    ) : (
      <div className="grid grid-cols-1 md:grid-cols-2
gap-4">
        {filteredAccessories.map(accessory => (
          <div key={accessory.id} className="border
rounded-lg p-4 hover:shadow-md transition-shadow">
            <div className="flex justify-between items-
start mb-2">
              <h4 className="font-medium text-gray-900
line-clamp-2">{accessory.name}</h4>
              <span className="text-sm text-gray-500 ml-
2">
                {accessory.specifications?.type ||
'Аксессуар'}
              </span>
            </div>
          </div>
        ))}
      </div>
    )}
  </div>
</div>

```

```

        <p className="text-sm text-gray-600 mb-2">
          {accessory.brand} • {accessory.model}
        </p>

        <div className="flex justify-between items-
center">
          <span className="text-lg font-bold text-
blue-600">
            {formatPrice(accessory.price)}
          </span>

          <button
            onClick={() => addAccessory(accessory)}
            className="btn-primary flex items-center
space-x-1"
            >
            <Plus className="w-4 h-4" />
            <span>Добавить</span>
          </button>
        </div>
      </div>
    )))
  </div>
) }
</div>

{/* Правая панель - выбранные аксессуары */}
<div className="w-80 border-l bg-gray-50 p-6 overflow-
y-auto">
  <h3 className="text-lg font-semibold text-gray-900
mb-4">
    Выбранные аксессуары
    ({selectedAccessories.length})
  </h3>

  {selectedAccessories.length === 0 ? (
    <div className="text-center py-8">
      <ShoppingCart className="w-12 h-12 text-gray-400
mx-auto mb-2" />
      <p className="text-gray-600">Аксессуары не
выбраны</p>
    </div>
  ) : (
    <div className="space-y-3">
      {selectedAccessories.map(item => (
        <div key={item.component.id} className="bg-
white rounded-lg p-3 border">
          <div className="flex justify-between items-
start mb-2">
            <h5 className="font-medium text-sm line-
clamp-2">{item.component.name}</h5>
            <button

```

```

                                onClick={() =>
removeAccessory(item.component.id)}
                                className="text-red-500 hover:text-red-
700 ml-2"
                                >
                                <X className="w-4 h-4" />
                                </button>
                                </div>

                                <div className="flex items-center justify-
between">
                                <div className="flex items-center space-x-
2">
                                <button
                                onClick={() =>
updateQuantity(item.component.id, item.quantity - 1)}
                                className="w-6 h-6 rounded border flex
items-center justify-center hover:bg-gray-100"
                                >
                                <Minus className="w-3 h-3" />
                                </button>
                                <span className="w-8 text-
center">{item.quantity}</span>
                                <button
                                onClick={() =>
updateQuantity(item.component.id, item.quantity + 1)}
                                className="w-6 h-6 rounded border flex
items-center justify-center hover:bg-gray-100"
                                >
                                <Plus className="w-3 h-3" />
                                </button>
                                </div>

                                <span className="text-sm font-medium text-
blue-600">
                                {formatPrice(item.component.price *
item.quantity)}
                                </span>
                                </div>
                                </div>
                                )}}
                                </div>
                                )}

                                {/* Итого */}
                                {selectedAccessories.length > 0 && (
                                <div className="mt-6 pt-4 border-t">
                                <div className="flex justify-between items-
center text-lg font-bold">
                                <span>Итого:</span>
                                <span className="text-blue-
600">{formatPrice(totalPrice)}</span>
                                </div>

```

```

        </div>
      )}
    </div>
  </div>

  {/* Футер */}
  <div className="p-6 border-t bg-gray-50 flex justify-
between items-center">
    <div className="text-sm text-gray-600">
      {selectedAccessories.length > 0 && (
        <span>Выбрано аксессуаров:
{selectedAccessories.length}</span>
      )}
    </div>

    <div className="flex space-x-3">
      <button
        onClick={onClose}
        className="btn-outline"
      >
        Отмена
      </button>
      <button
        onClick={handleConfirm}
        className="btn-primary"
      >
        Продолжить с выбранными аксессуарами
      </button>
    </div>
  </div>
</motion.div>
</motion.div>
</AnimatePresence>
  );
}

```

Листинг В.2 – Файл CompatibilityCheck.tsx

```

import React from 'react';
import { motion } from 'framer-motion';
import { CheckCircle, AlertTriangle, XCircle, Info, Zap } from
' lucide-react';
import { Configuration, CompatibilityIssue } from '../types';

interface CompatibilityCheckProps {
  configuration?: Configuration;
}

export default function CompatibilityCheck({ configuration }:
CompatibilityCheckProps) {
  if (!configuration || !configuration.items ||
configuration.items.length < 2) {
    return (
      <div className="text-center py-8">

```

```

        <Info className="w-12 h-12 text-gray-400 mx-auto mb-4" />
        <p className="text-gray-600">
            Добавьте компоненты для проверки совместимости
        </p>
        <p className="text-sm text-gray-500 mt-2">
            Минимум 2 компонента для анализа
        </p>
    </div>
    );
}

const getStatusIcon = (status: string) => {
    switch (status) {
        case 'compatible':
            return <CheckCircle className="w-6 h-6 text-green-500" />;
        case 'warning':
            return <AlertTriangle className="w-6 h-6 text-yellow-500"
/>;
        case 'incompatible':
            return <XCircle className="w-6 h-6 text-red-500" />;
        default:
            return <Info className="w-6 h-6 text-gray-500" />;
    }
};

const getStatusColor = (status: string) => {
    switch (status) {
        case 'compatible':
            return 'bg-green-50 border-green-200 text-green-700';
        case 'warning':
            return 'bg-yellow-50 border-yellow-200 text-yellow-700';
        case 'incompatible':
            return 'bg-red-50 border-red-200 text-red-700';
        default:
            return 'bg-gray-50 border-gray-200 text-gray-700';
    }
};

const getStatusText = (status: string) => {
    switch (status) {
        case 'compatible':
            return 'Все компоненты совместимы';
        case 'warning':
            return 'Есть предупреждения';
        case 'incompatible':
            return 'Обнаружены проблемы';
        default:
            return 'Проверка совместимости';
    }
};

const getIssueSeverityIcon = (severity: string) => {
    switch (severity) {

```

```

    case 'error':
      return <XCircle className="w-5 h-5 text-red-500" />;
    case 'warning':
      return <AlertTriangle className="w-5 h-5 text-yellow-500"
/;>;
    case 'info':
      return <Info className="w-5 h-5 text-blue-500" />;
    default:
      return <Info className="w-5 h-5 text-gray-500" />;
  }
};

const issues = configuration.compatibility_issues || [];

return (
  <div className="space-y-4">
    <h3 className="text-lg font-semibold text-gray-900 mb-4">
      Проверка совместимости
    </h3>

    { /* Общий статус */ }
    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      className={`p-4 rounded-lg border
${getStatusColor(configuration.compatibility_status)} `
    >
      <div className="flex items-center space-x-3">
        {getStatusIcon(configuration.compatibility_status)}
        <div className="flex-1">
          <h4 className="font-medium">
            {getStatusText(configuration.compatibility_status)}
          </h4>
          <p className="text-sm opacity-75">
            Проверено {configuration.items.length} компонентов
          </p>
        </div>
      </div>
    </motion.div>

    { /* Энергопотребление */ }
    {configuration.compatibility_issues?.some(issue =>
issue.type === 'power') && (
      <div className="p-4 bg-blue-50 border border-blue-200
rounded-lg">
        <div className="flex items-center space-x-2 mb-2">
          <Zap className="w-5 h-5 text-blue-600" />
          <span className="font-medium text-blue-
700">Энергопотребление</span>
        </div>
        <div className="text-sm text-blue-600">
          { /* Здесь можно добавить расчет общего
энергопотребления */ }

```

```

        <p>Рекомендуемая мощность БП:
{configuration.compatibility_issues.find(i => i.type ===
'power')?.suggestions?.[0] || 'Проверьте характеристики'}</p>
    </div>
  </div>
)}

{/* Список проблем */}
{issues.length > 0 && (
  <div className="space-y-3">
    <h4 className="font-medium text-gray-900">Обнаруженные
проблемы:</h4>
    {issues.map((issue, index) => (
      <motion.div
        key={index}
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        transition={{ delay: index * 0.1 }}
        className="p-3 bg-white border rounded-lg"
      >
        <div className="flex items-start space-x-3">
          {getIssueSeverityIcon(issue.severity)}
          <div className="flex-1">
            <p className="text-sm font-medium text-gray-
900">
              {issue.message}
            </p>
            {issue.suggestions && issue.suggestions.length >
0 && (
              <div className="mt-2">
                <p className="text-xs text-gray-600 mb-
1">Рекомендации:</p>
                <ul className="text-xs text-gray-600 space-
y-1">
                  {issue.suggestions.map((suggestion,
suggestionIndex) => (
                    <li key={suggestionIndex}
className="flex items-start space-x-1">
                      <span>•</span>
                      <span>{suggestion}</span>
                    </li>
                  )))}
                </ul>
              </div>
            </div>
          </div>
        </div>
      </motion.div>
    ))}
  </div>
)}

{/* Если нет проблем */}

```

```

    {issues.length === 0 && configuration.compatibility_status
=== 'compatible' && (
    <div className="text-center py-4">
      <CheckCircle className="w-12 h-12 text-green-500 mx-auto
mb-2" />
      <p className="text-green-700 font-medium">
        Отлично! Все компоненты совместимы
      </p>
      <p className="text-sm text-green-600 mt-1">
        Ваша конфигурация готова к сборке
      </p>
    </div>
  )}

  {/* Дополнительная информация */}
  <div className="mt-6 p-3 bg-gray-50 rounded-lg">
    <h5 className="text-sm font-medium text-gray-700 mb-2">
      Что проверяется:
    </h5>
    <ul className="text-xs text-gray-600 space-y-1">
      <li>• Совместимость сокета процессора и материнской
платы</li>
      <li>• Тип и частота оперативной памяти</li>
      <li>• Энергопотребление и мощность блока питания</li>
      <li>• Форм-фактор корпуса и материнской платы</li>
      <li>• Интерфейсы подключения накопителей</li>
    </ul>
  </div>
</div>
);
}

```

Листинг В.3 – Файл Service.cs

```

import React, { useState, useEffect } from 'react';
import { useQuery } from 'react-query';
import { motion } from 'framer-motion';
import { Search, Filter, X, Check, AlertCircle, Package } from
' lucide-react';
import { Component, ComponentFilter } from '../.. /types';
import { componentsApi, formatPrice, formatStock } from
'../.. /lib/api';

interface ComponentSelectorProps {
  category: string;
  selectedComponent?: Component;
  onComponentSelect: (component: Component) => void;
  onComponentRemove: () => void;
}

export default function ComponentSelector({
  category,
  selectedComponent,
  onComponentSelect,

```

```

    onComponentRemove,
  }: ComponentSelectorProps) {
    const [filters, setFilters] = useState<ComponentFilter>({
      category_slug: category,
      only_in_stock: false,
      page: 1,
      limit: 20,
    });

    const [searchTerm, setSearchTerm] = useState('');
    const [showFilters, setShowFilters] = useState(false);

    // Загрузка компонентов
    const {
      data: componentsData,
      isLoading,
      error,
      refetch,
    } = useQuery(
      ['components', category, filters],
      () => componentsApi.getByCategory(category, filters),
      {
        enabled: !!category,
        keepPreviousData: true,
      }
    );

    // Обновление фильтров при смене категории
    useEffect(() => {
      setFilters(prev => ({
        ...prev,
        category_slug: category,
        page: 1,
      }));
      setSearchTerm('');
    }, [category]);

    // Обновление поискового запроса
    useEffect(() => {
      const debounceTimer = setTimeout(() => {
        setFilters(prev => ({
          ...prev,
          search: searchTerm || undefined,
          page: 1,
        }));
      }, 500);

      return () => clearTimeout(debounceTimer);
    }, [searchTerm]);

    const handleFilterChange = (key: keyof ComponentFilter, value:
    any) => {
      setFilters(prev => ({

```

```

        ...prev,
        [key]: value,
        page: 1,
    }));
};

const getCategoryName = (slug: string) => {
    const names: Record<string, string> = {
        cpu: 'Процессоры',
        motherboard: 'Материнские платы',
        ram: 'Оперативная память',
        gpu: 'Видеокарты',
        storage: 'Накопители',
        psu: 'Блоки питания',
        case: 'Корпуса',
        cooler: 'Охлаждение',
    };
    return names[slug] || 'Компоненты';
};

const getStockColor = (status: string) => {
    switch (status) {
        case 'in_stock':
            return 'text-green-600 bg-green-50';
        case 'expected':
            return 'text-yellow-600 bg-yellow-50';
        case 'out_of_stock':
            return 'text-red-600 bg-red-50';
        default:
            return 'text-gray-600 bg-gray-50';
    }
};

if (isLoading) {
    return (
        <div className="p-6 h-full flex items-center justify-center">
            <div className="text-center">
                <div className="loading-dots mb-4">
                    <div className="loading-dot"></div>
                    <div className="loading-dot"></div>
                    <div className="loading-dot"></div>
                </div>
                <p className="text-gray-600">Загрузка компонентов...</p>
            </div>
        </div>
    );
}

if (error) {
    return (
        <div className="p-6 h-full flex items-center justify-center">

```

```

        <div className="text-center">
            <AlertCircle className="w-12 h-12 text-red-500 mx-auto
mb-4" />
            <p className="text-red-600 mb-4">Ошибка загрузки
компонентов</p>
            <button onClick={() => refetch()} className="btn-
primary">
                Попробовать снова
            </button>
        </div>
    </div>
);
}

const components = componentsData || [];

return (
    <div className="h-full flex flex-col">
        {/* Заголовок */}
        <div className="p-4 border-b bg-gray-50">
            <div className="flex items-center justify-between mb-4">
                <div>
                    <h3 className="text-lg font-semibold text-gray-900">
                        {getCategoryName(category)}
                    </h3>
                    <{!isLoading} && (
                        <p className="text-sm text-gray-500">
                            Найдено: {components.length} компонентов
                        </p>
                    </>
                </div>
                <div>
                    <div>
                        {selectedComponent} && (
                            <button
                                onClick={onComponentRemove}
                                className="text-red-600 hover:text-red-700 flex
items-center space-x-1"
                            >
                                <X className="w-4 h-4" />
                                <span className="text-sm">Убрать</span>
                            </button>
                        </>
                    </div>
                </div>
            </div>
            {/* Поиск */}
            <div className="relative mb-4">
                <Search className="absolute left-3 top-1/2 transform -
translate-y-1/2 text-gray-400 w-5 h-5" />
                <input
                    type="text"
                    placeholder="Поиск компонентов..."
                    value={searchTerm}
                    onChange={(e) => setSearchTerm(e.target.value)}
                    className="input pl-10 pr-10 w-full"
                />
            </div>
        </div>
    </div>
);

```

```

    />
    {isLoading && (
      <div className="absolute right-3 top-1/2 transform -
translate-y-1/2">
        <div className="animate-spin rounded-full h-4 w-4
border-b-2 border-blue-600"></div>
      </div>
    )}
  </div>

  {/* Фильтры */}
  <div className="flex items-center justify-between">
    <button
      onClick={() => setShowFilters(!showFilters)}
      className="btn-outline flex items-center space-x-2"
    >
      <Filter className="w-4 h-4" />
      <span>Фильтры</span>
    </button>

    <label className="flex items-center space-x-2">
      <input
        type="checkbox"
        checked={filters.only_in_stock || false}
        onChange={(e) => handleFilterChange('only_in_stock',
e.target.checked)}
        className="rounded border-gray-300"
      />
      <span className="text-sm text-gray-600">Только в
наличии</span>
    </label>
  </div>

  {/* Расширенные фильтры */}
  {showFilters && (
    <motion.div
      initial={{ opacity: 0, height: 0 }}
      animate={{ opacity: 1, height: 'auto' }}
      exit={{ opacity: 0, height: 0 }}
      className="mt-4 p-4 bg-white rounded-lg border"
    >
      <div className="grid grid-cols-1 sm:grid-cols-2 gap-
4">
        <div>
          <label className="block text-sm font-medium text-
gray-700 mb-2">
            Цена от
          </label>
          <input
            type="number"
            placeholder="0"
            value={filters.price_min || ''}

```

```

        onChange={ (e) => handleFilterChange('price_min',
parseInt(e.target.value) || undefined)}
        className="input w-full"
      />
    </div>
    <div>
      <label className="block text-sm font-medium text-
gray-700 mb-2">
        Цена до
      </label>
      <input
        type="number"
        placeholder="100000"
        value={filters.price_max || ''}
        onChange={ (e) => handleFilterChange('price_max',
parseInt(e.target.value) || undefined)}
        className="input w-full"
      />
    </div>
  </div>
</motion.div>
)}
</div>

{/* ВЫБРАННЫЙ КОМПОНЕНТ */}
{selectedComponent && (
  <div className="p-4 bg-blue-50 border-b">
    <div className="flex items-start space-x-3">
      <div className="w-8 h-8 bg-green-500 rounded-full flex
items-center justify-center">
        <Check className="w-5 h-5 text-white" />
      </div>
      <div className="flex-1">
        <h4 className="font-medium text-gray-
900">{selectedComponent.name}</h4>
        <p className="text-sm text-gray-
600">{selectedComponent.brand}</p>
        <p className="text-sm font-medium text-green-600">
          {formatPrice(selectedComponent.price)}
        </p>
      </div>
    </div>
  </div>
)}

{/* Список компонентов */}
<div className="flex-1 overflow-y-auto custom-scrollbar"
style={{ maxHeight: 'calc(100vh - 400px)', minHeight: '300px' }}>
  {components.length === 0 ? (
    <div className="p-8 text-center">
      <Package className="w-12 h-12 text-gray-400 mx-auto
mb-4" />
      <p className="text-gray-600">Компоненты не найдены</p>

```

```

        <p className="text-sm text-gray-500 mt-2">
          Попробуйте изменить фильтры или поисковый запрос
        </p>
      </div>
    ) : (
      <div className="p-4 space-y-3">
        {components.map((component) => (
          <motion.div
            key={component.id}
            className={`component-card ${
              selectedComponent?.id === component.id ?
'selected' : ''
            } ${component.stock?.status === 'out_of_stock' ?
'unavailable' : ''}`}
            whileHover={{ scale: 1.02 }}
            whileTap={{ scale: 0.98 }}
            onClick={() => {
              if (component.stock?.status !== 'out_of_stock')
{
                  onComponentSelect(component);
                }
            }}
          >
            <div className="flex justify-between items-start">
              <div className="flex-1">
                <h4 className="font-medium text-gray-900 mb-
1">
                  {component.name}
                </h4>
                <p className="text-sm text-gray-600 mb-2">
                  {component.brand} • {component.model}
                </p>

                {/* Основные характеристики */}
                {component.specifications && (
                  <div className="text-xs text-gray-500 mb-2">
                    {Object.entries(component.specifications)
                      .slice(0, 2)
                      .map(([key, value]) => (
                        <span key={key} className="mr-2">
                          {key}: {value}
                        </span>
                      ))}
                  </div>
                )}

                {/* Цена */}
                <div className="flex items-center justify-
between">
                  <span className="text-lg font-bold text-
gray-900">
                    {formatPrice(component.price)}
                  </span>

```

```

        { /* Статус наличия */ }
        { component.stock && (
            <span className={`px-2 py-1 text-xs
rounded-full ${getStockColor(component.stock.status)}`}>
                {formatStock(component.stock)}
            </span>
        ) }
    </div>
</div>

    { /* Изображение */ }
    { component.image_url && (
        <div className="ml-4 w-16 h-16 bg-gray-100
rounded-lg overflow-hidden">
            <img
                src={component.image_url}
                alt={component.name}
                className="w-full h-full object-cover"
            />
        </div>
    ) }
</div>

    { /* Индикатор выбранного */ }
    { selectedComponent?.id === component.id && (
        <div className="absolute top-2 right-2 w-6 h-6
bg-green-500 rounded-full flex items-center justify-center">
            <Check className="w-4 h-4 text-white" />
        </div>
    ) }
</motion.div>
    ) ) }
</div>
    ) }
</div>
</div>
);
}

```

Листинг В.4 – Файл ConfigSummary.tsx

```

import React, { useState } from 'react';
import { motion } from 'framer-motion';
import { Download, Save, Share2, ShoppingCart, Calculator, Package
} from 'lucide-react';
import { Component, Configuration } from '../types';
import { formatPrice } from '../lib/api';
import { useConfiguratorStore } from
'../hooks/useConfiguratorStore';
import AccessorySelector from './AccessorySelector';
import { useRouter } from 'next/router';

interface ConfigSummaryProps {

```

```

    selectedComponents: Record<string, Component>;
    configuration?: Configuration;
}

export default function ConfigSummary({ selectedComponents,
configuration }: ConfigSummaryProps) {
    const [isExporting, setIsExporting] = useState(false);
    const [showSaveDialog, setShowSaveDialog] = useState(false);
    const [showAccessorySelector, setShowAccessorySelector] =
useState(false);
    const [configName, setConfigName] = useState('');
    const [configDescription, setConfigDescription] = useState('');

    const { saveConfiguration, exportToPdf,
addAccessoriesToConfiguration, isLoading } =
useConfiguratorStore();
    const router = useRouter();

    const componentList = Object.values(selectedComponents);
    const totalPrice = componentList.reduce((sum, component) => sum
+ component.price, 0);
    const componentCount = componentList.length;

    const handleSave = async () => {
        if (!configName.trim()) return;

        try {
            await saveConfiguration(configName, configDescription);
            setShowSaveDialog(false);
            setConfigName('');
            setConfigDescription('');
        } catch (error) {
            console.error('Ошибка сохранения:', error);
        }
    };

    const handleExportPDF = async () => {
        if (!isConfigurationSaved) {
            return;
        }

        // Показываем селектор аксессуаров перед экспортом
        setShowAccessorySelector(true);
    };

    const handleAccessoriesSelected = async (selectedAccessories:
Array<{component: Component, quantity: number}>) => {
        setIsExporting(true);
        try {
            // Добавляем выбранные аксессуары в конфигурацию
            if (selectedAccessories.length > 0) {
                await addAccessoriesToConfiguration(selectedAccessories);
            }
        }
    };
}

```

```

    // Экспортируем в PDF
    const pdfBlob = await exportToPdf();
    if (pdfBlob) {
        // Создаем ссылку для скачивания
        const url = window.URL.createObjectURL(pdfBlob);
        const link = document.createElement('a');
        link.href = url;
        link.download = `конфигурация_${configuration?.name ||
'config'}.pdf`;
        document.body.appendChild(link);
        link.click();
        document.body.removeChild(link);
        window.URL.revokeObjectURL(url);
    }
    } catch (error) {
        console.error('Ошибка экспорта:', error);
    } finally {
        setIsExporting(false);
    }
};

const handleShare = () => {
    if (configuration?.public_uuid) {
        const shareUrl =
`${window.location.origin}/config/${configuration.public_uuid}`;
        navigator.clipboard.writeText(shareUrl);
        // Можно добавить уведомление о копировании
    }
};

const isConfigurationSaved = configuration?.id &&
configuration.id !== '0';

console.log('ConfigSummary - configuration:', configuration);
console.log('ConfigSummary - isConfigurationSaved:',
isConfigurationSaved);

const getCategoryName = (slug: string) => {
    const names: Record<string, string> = {
        cpu: 'Процессор',
        motherboard: 'Материнская плата',
        ram: 'Оперативная память',
        gpu: 'Видеокарта',
        storage: 'Накопитель',
        psu: 'Блок питания',
        case: 'Корпус',
        cooler: 'Охлаждение',
    };
    return names[slug] || 'Компонент';
};

if (componentCount === 0) {

```

```

return (
  <div className="text-center py-8">
    <Package className="w-12 h-12 text-gray-400 mx-auto mb-4"
  />
    <p className="text-gray-600">
      Конфигурация пуста
    </p>
    <p className="text-sm text-gray-500 mt-2">
      Добавьте компоненты для просмотра сводки
    </p>
  </div>
);
}

return (
  <div className="space-y-4">
    <div className="flex items-center justify-between">
      <div className="flex items-center space-x-3">
        <h3 className="text-lg font-semibold text-gray-900">
          Сводка конфигурации
        </h3>
        {isConfigurationSaved && (
          <span className="px-2 py-1 text-xs bg-green-100 text-green-800 rounded-full">
            Сохранена
          </span>
        )}
      </div>
      <div className="flex items-center space-x-2 text-sm text-gray-600">
        <Calculator className="w-4 h-4" />
        <span>{componentCount} компонентов</span>
      </div>
    </div>

    { /* Список выбранных компонентов */ }
    <div className="space-y-2 max-h-64 overflow-y-auto">
      {componentList.map((component) => (
        <motion.div
          key={component.id}
          initial={{ opacity: 0, x: 20 }}
          animate={{ opacity: 1, x: 0 }}
          className="flex items-center justify-between p-3 bg-gray-50 rounded-lg"
        >
          <div className="flex-1">
            <p className="text-sm font-medium text-gray-900">
              {getCategoryName(component.category.slug)}
            </p>
            <p className="text-xs text-gray-600 truncate">
              {component.brand} {component.name}
            </p>
          </div>
        </div>
      )}
    </div>
  </div>
);

```

```

        <div className="text-sm font-medium text-gray-900">
          {formatPrice(component.price)}
        </div>
      </motion.div>
    )})
  </div>

  {/* Итоговая цена */}
  <div className="border-t pt-4">
    <div className="flex items-center justify-between">
      <span className="text-lg font-semibold text-gray-900">
        Итого:
      </span>
      <span className="text-2xl font-bold text-green-600">
        {formatPrice(totalPrice)}
      </span>
    </div>

    {/* Дополнительная информация */}
    <div className="mt-2 text-sm text-gray-600">
      <p>• Цены указаны без учета доставки</p>
      <p>• Наличие компонентов может изменяться</p>
    </div>
  </div>

  {/* Действия */}
  <div className="space-y-3 pt-4 border-t">
    {/* Сохранение конфигурации */}
    <button
      onClick={() => setShowSaveDialog(true)}
      disabled={componentCount === 0}
      className="btn-primary w-full flex items-center justify-
center space-x-2"
    >
      <Save className="w-4 h-4" />
      <span>Сохранить конфигурацию</span>
    </button>

    {/* Дополнительные действия */}
    <div className="grid grid-cols-2 gap-2">
      <button
        onClick={handleExportPDF}
        disabled={!isConfigurationSaved || isExporting}
        className="btn-outline flex items-center justify-
center space-x-2"
      >
        <Download className="w-4 h-4" />
        <span>{isExporting ? 'Экспорт...' : 'PDF'}</span>
      </button>

      <button
        onClick={handleShare}
        disabled={!isConfigurationSaved}

```

```

        className="btn-outline flex items-center justify-
center space-x-2"
      >
        <Share2 className="w-4 h-4" />
        <span>Поделиться</span>
      </button>
    </div>

    { /* Покупка (заглушка) */ }
    <button
      onClick={() => router.push('/payment')}
      disabled={componentCount === 0}
      className="btn-secondary w-full flex items-center
justify-center space-x-2"
    >
      <ShoppingCart className="w-4 h-4" />
      <span>Перейти к покупке</span>
    </button>
  </div>

  { /* Диалог сохранения */ }
  { showSaveDialog && (
    <motion.div
      initial={{ opacity: 0 }}
      animate={{ opacity: 1 }}
      className="fixed inset-0 bg-black bg-opacity-50 flex
items-center justify-center z-50"
      onClick={() => setShowSaveDialog(false)}
    >
      <motion.div
        initial={{ scale: 0.9, opacity: 0 }}
        animate={{ scale: 1, opacity: 1 }}
        className="bg-white rounded-lg p-6 max-w-md w-full mx-
4"
        onClick={(e) => e.stopPropagation()}
      >
        <h4 className="text-lg font-semibold text-gray-900 mb-
4">
          Сохранить конфигурацию
        </h4>

        <div className="space-y-4">
          <div>
            <label className="block text-sm font-medium text-
gray-700 mb-2">
              Название конфигурации *
            </label>
            <input
              type="text"
              value={configName}
              onChange={(e) => setConfigName(e.target.value)}
              placeholder="Мой игровой ПК"
              className="input w-full"

```

```

        autoFocus
      />
    </div>

    <div>
      <label className="block text-sm font-medium text-gray-700 mb-2">
        Описание
      </label>
      <textarea
        value={configDescription}
        onChange={ (e) =>
setConfigDescription(e.target.value) }
        placeholder="Описание сборки (необязательно)"
        className="input w-full h-20 resize-none"
      />
    </div>
  </div>

  <div className="flex space-x-3 mt-6">
    <button
      onClick={() => setShowSaveDialog(false)}
      className="btn-outline flex-1"
    >
      Отмена
    </button>
    <button
      onClick={handleSave}
      disabled={!configName.trim()}
      className="btn-primary flex-1"
    >
      Сохранить
    </button>
  </div>
</motion.div>
</motion.div>
)}

{/* Селектор аксессуаров */}
<AccessorySelector
  isOpen={showAccessorySelector}
  onClose={() => setShowAccessorySelector(false)}
  onConfirm={handleAccessoriesSelected}
/>
</div>
);
}

```

Листинг В.5 – Файл index.tsx

```

import React, { useState } from 'react';
import { motion } from 'framer-motion';
import PCModel from './PCModel';
import ComponentSelector from './ComponentSelector';

```

```

import ConfigSummary from './ConfigSummary';
import CompatibilityCheck from './CompatibilityCheck';
import { useConfiguratorStore } from
'../../hooks/useConfiguratorStore';
import { Component, ComponentCategory } from '../../types';

export default function Configurator() {
  const [selectedCategory, setSelectedCategory] =
useState<string>('cpu');

  const {
    selectedComponents,
    currentConfiguration: configuration,
    addComponent,
    removeComponent,
    checkCompatibility,
    isLoading,
    error
  } = useConfiguratorStore();

  const categories: ComponentCategory[] = [
    { id: "1", name: 'Процессор', slug: 'cpu', order_priority: 1,
icon: '■' },
    { id: "2", name: 'Материнская плата', slug: 'motherboard',
order_priority: 2, icon: '🔧' },
    { id: "3", name: 'Оперативная память', slug: 'ram',
order_priority: 3, icon: '📁' },
    { id: "4", name: 'Видеокарта', slug: 'gpu', order_priority: 4,
icon: '🎮' },
    { id: "5", name: 'Накопитель', slug: 'storage',
order_priority: 5, icon: '🕸' },
    { id: "6", name: 'Блок питания', slug: 'psu', order_priority:
6, icon: '⚡' },
    { id: "7", name: 'Корпус', slug: 'case', order_priority: 7,
icon: '📦' },
    { id: "8", name: 'Охлаждение', slug: 'cooler', order_priority:
8, icon: '❄' },
  ];

  const handleComponentSelect = (component: Component) => {
    addComponent(component);
    checkCompatibility();
  };

  const handleComponentRemove = (categorySlug: string) => {
    removeComponent(categorySlug);
    checkCompatibility();
  };

  return (

```

```

    <div className="bg-white rounded-2xl shadow-xl overflow-
hidden">
    { /* Заголовок configurатора */ }
    <div className="bg-gradient-to-r from-blue-600 to-purple-600
text-white p-6">
    <h2 className="text-2xl md:text-3xl font-bold mb-2">
    Конфигуратор ПК
    </h2>
    <p className="text-blue-100">
    Выберите компоненты для вашего идеального компьютера
    </p>
    </div>

    { /* Основной контент */ }
    <div className="grid grid-cols-1 lg:grid-cols-12 min-h-
[800px]">
    { /* Левая панель - Категории */ }
    <div className="lg:col-span-3 bg-gray-50 border-r">
    <div className="p-4">
    <h3 className="text-lg font-semibold text-gray-900 mb-
4">
    Компоненты
    </h3>
    <div className="space-y-2">
    {categories.map((category) => (
    <button
    key={category.slug}
    onClick={() =>
setSelectedCategory(category.slug)}
    className={`w-full text-left p-3 rounded-lg
transition-colors duration-200 flex items-center space-x-3 ${
    selectedCategory === category.slug
    ? 'bg-blue-600 text-white'
    : 'bg-white hover:bg-gray-100 text-gray-700'
    }`}
    >
    <span className="text-lg">{category.icon}</span>
    <div className="flex-1">
    <span className="font-
medium">{category.name}</span>
    {selectedComponents[category.slug] && (
    <div className="text-xs opacity-75
truncate">
    {selectedComponents[category.slug].name}
    </div>
    )}
    </div>
    {selectedComponents[category.slug] && (
    <div className="w-2 h-2 bg-green-400 rounded-
full"></div>
    )}
    </button>
    )}
    )}

```

```

        </div>
    </div>
</div>

{ /* Центральная часть - Модель ПК */ }
<div className="lg:col-span-5 relative">
    <div className="sticky top-20">
        <div className="p-6 h-full flex flex-col">
            <div className="flex items-center justify-between
mb-4">
                <h3 className="text-lg font-semibold text-gray-
900">
                    Визуализация сборки
                </h3>

            </div>

            <div className={`flex-1 bg-gradient-to-br from-
slate-100 to-slate-200 rounded-xl ${
                false ? 'fixed inset-0 z-50 p-8' : ''
            }`} >
                <PCModel
                    selectedComponents={selectedComponents}
                    selectedCategory={selectedCategory}
                    onComponentClick={setSelectedCategory}
                    expanded={false}
                    onClose={() => {}}
                />
            </div>
        </div>
    </div>
</div>
</div>

{ /* Правая панель - Селектор компонентов */ }
<div className="lg:col-span-4 border-l">
    <div className="h-full flex flex-col">
        <ComponentSelector
            category={selectedCategory}

selectedComponent={selectedComponents[selectedCategory]}
            onComponentSelect={handleComponentSelect}
            onComponentRemove={() =>
handleComponentRemove(selectedCategory)}
        />
    </div>
</div>
</div>

{ /* Нижняя панель - Сводка и совместимость */ }
<div className="border-t bg-gray-50">
    <div className="grid grid-cols-1 lg:grid-cols-2">
        <div className="p-6 border-r">
            <CompatibilityCheck configuration={configuration} />

```

```

    </div>
    <div className="p-6">
      <ConfigSummary
        selectedComponents={selectedComponents}
        configuration={configuration}
      />
    </div>
  </div>
</div>

{/* Ошибки */}
{error && (
  <motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    className="bg-red-50 border-l-4 border-red-400 p-4 m-4"
  >
    <div className="flex">
      <div className="ml-3">
        <p className="text-sm text-red-700">{error}</p>
      </div>
    </div>
  </motion.div>
)}
</div>
);
}

```

Листинг В.6 – PCModel.tsx

```

import React from 'react';
import { motion } from 'framer-motion';
import { X, Maximize2, Cpu, HardDrive, Zap, Fan, MemoryStick,
Monitor, Boxes } from 'lucide-react';
import { Component } from '../types';

interface PCModelProps {
  selectedComponents: Record<string, Component>;
  selectedCategory: string;
  onComponentClick: (category: string) => void;
  expanded: boolean;
  onClose: () => void;
}

interface ComponentPosition {
  x: number;
  y: number;
  width: number;
  height: number;
  label: string;
  category: string;
  icon: React.ReactNode;
  depth?: number;
}

```

```

    shape?: 'motherboard' | 'cpu' | 'ram' | 'gpu' | 'storage' |
    'psu' | 'cooler';
  }

export default function PCModel({
  selectedComponents,
  selectedCategory,
  onComponentClick,
  expanded,
  onClose,
}: PCModelProps) {
  // Позиции компонентов в 3D модели ПК (в процентах) -
  // схематичное расположение
  const componentPositions: ComponentPosition[] = [
    {
      x: 10, y: 25, width: 65, height: 50,
      label: 'Материнская плата', category: 'motherboard',
      icon: <Boxes className="w-6 h-6" />,
      depth: 1,
      shape: 'motherboard'
    },
    {
      x: 15, y: 30, width: 15, height: 15,
      label: 'CPU', category: 'cpu',
      icon: <Cpu className="w-5 h-5" />,
      depth: 2,
      shape: 'cpu'
    },
    {
      x: 40, y: 28, width: 4, height: 22,
      label: 'RAM', category: 'ram',
      icon: <MemoryStick className="w-4 h-4" />,
      depth: 3,
      shape: 'ram'
    },
    {
      x: 46, y: 28, width: 4, height: 22,
      label: 'RAM', category: 'ram',
      icon: <MemoryStick className="w-4 h-4" />,
      depth: 3,
      shape: 'ram'
    },
    {
      x: 12, y: 52, width: 45, height: 20,
      label: 'GPU', category: 'gpu',
      icon: <Monitor className="w-6 h-6" />,
      depth: 4,
      shape: 'gpu'
    },
    {
      x: 78, y: 35, width: 15, height: 8,
      label: 'SSD', category: 'storage',
      icon: <HardDrive className="w-4 h-4" />,

```

```

        depth: 2,
        shape: 'storage'
      },
      {
        x: 78, y: 70, width: 18, height: 18,
        label: 'PSU', category: 'psu',
        icon: <Zap className="w-5 h-5" />,
        depth: 1,
        shape: 'psu'
      },
      {
        x: 35, y: 10, width: 12, height: 12,
        label: 'Охлаждение', category: 'cooler',
        icon: <Fan className="w-5 h-5" />,
        depth: 6,
        shape: 'cooler'
      },
    ],
  ];

  const [hoveredComponent, setHoveredComponent] =
    React.useState<string | null>(null);

  const getComponentStatus = (category: string) => {
    if (selectedComponents[category]) {
      return 'selected';
    }
    if (selectedCategory === category) {
      return 'highlighted';
    }
    return 'default';
  };

  // Проверяем, находится ли компонент на материнской плате
  const isOnMotherboard = (category: string) => {
    return ['cpu', 'ram', 'gpu', 'cooler'].includes(category);
  };

  // Определяем, можно ли кликнуть на компонент
  const isClickable = (category: string) => {
    // Все компоненты всегда кликабельны
    return true;
  };

  // Определяем, можно ли применить hover эффект
  const isHoverable = (category: string) => {
    // Материнская плата не увеличивается при наведении
    if (category === 'motherboard') {
      return false;
    }
    return isClickable(category);
  };

```

```

const getSchematicStyle = (shape: string, status: string, depth:
number = 1) => {
  const baseStyles = {
    borderWidth: '2px',
    borderStyle: 'solid',
    cursor: 'pointer',
    transition: 'all 0.3s ease',
  } as React.CSSProperties;

  const statusStyles = {
    selected: {
      borderColor: '#22c55e',
      boxShadow: '0 0 20px rgba(34, 197, 94, 0.4), inset 0 1px
3px rgba(255,255,255,0.1)',
    },
    highlighted: {
      borderColor: '#3b82f6',
      boxShadow: '0 0 20px rgba(59, 130, 246, 0.4), inset 0 1px
3px rgba(255,255,255,0.1)',
      transform: 'scale(1.05)',
    },
    default: {
      borderColor: '#64748b',
      boxShadow: 'inset 0 1px 3px rgba(0,0,0,0.2)',
    }
  };

  const shapeStyles = {
    motherboard: {
      background: 'linear-gradient(145deg, #1e293b, #334155)',
      borderRadius: '8px',
    },
    cpu: {
      background: 'linear-gradient(145deg, #374151, #4b5563)',
      borderRadius: '4px',
    },
    ram: {
      background: 'linear-gradient(145deg, #059669, #047857)',
      borderRadius: '2px',
    },
    gpu: {
      background: 'linear-gradient(145deg, #7c2d12, #9a3412)',
      borderRadius: '6px',
    },
    storage: {
      background: 'linear-gradient(145deg, #1f2937, #374151)',
      borderRadius: '3px',
    },
    psu: {
      background: 'linear-gradient(145deg, #18181b, #27272a)',
      borderRadius: '6px',
    },
    cooler: {

```

```

        background: 'radial-gradient(circle, #6366f1, #4f46e5)',
        borderRadius: '50%',
    }
};

return {
    ...baseStyles,
    ...shapeStyles[shape as keyof typeof shapeStyles],
    ...statusStyles[status as keyof typeof statusStyles],
    transform: `translateZ(${depth * 2}px) ${status ===
'highlighted' ? 'scale(1.05)' : 'scale(1)'}\``,
    } as React.CSSProperties;
};

const renderComponentDetails = (position: ComponentPosition) =>
{
    const { shape } = position;

    switch (shape) {
        case 'motherboard':
            return (
                <div className="absolute inset-0 pointer-events-none">
                    { /* Слоты и разъемы на материнской плате */ }
                    <div className="absolute top-2 left-2 w-4 h-1 bg-
yellow-500 rounded opacity-60"></div>
                    <div className="absolute top-4 left-2 w-4 h-1 bg-
yellow-500 rounded opacity-60"></div>
                    <div className="absolute bottom-2 right-2 w-3 h-3 bg-
blue-500 rounded opacity-60"></div>
                    <div className="absolute bottom-2 left-8 w-8 h-1 bg-
gray-400 rounded opacity-60"></div>
                    { /* Чипсет */ }
                    <div className="absolute bottom-4 left-4 w-6 h-6 bg-
gray-600 rounded border border-gray-500"></div>
                </div>
            );
        case 'cpu':
            return (
                <div className="absolute inset-0 pointer-events-none">
                    { /* Контакты процессора */ }
                    <div className="absolute inset-1 grid grid-cols-6 gap-
0.5 opacity-40">
                        {Array.from({ length: 36 }).map((_, i) => (
                            <div key={i} className="w-0.5 h-0.5 bg-yellow-400
rounded-full"></div>
                        ))}
                    </div>
                    { /* Крышка процессора */ }
                    <div className="absolute inset-2 bg-gray-500 rounded
border border-gray-400 opacity-80"></div>
                </div>
            );
        case 'ram':

```

```

    return (
        <div className="absolute inset-0 pointer-events-none">
            {/* Чипы памяти */}
            <div className="absolute top-1 left-0.5 right-0.5 h-1
bg-black rounded opacity-60"></div>
            <div className="absolute top-3 left-0.5 right-0.5 h-1
bg-black rounded opacity-60"></div>
            <div className="absolute bottom-3 left-0.5 right-0.5
h-1 bg-black rounded opacity-60"></div>
            <div className="absolute bottom-1 left-0.5 right-0.5
h-1 bg-black rounded opacity-60"></div>
            {/* Разъем */}
            <div className="absolute bottom-0 left-1/2 transform -
translate-x-1/2 w-1 h-1 bg-yellow-500"></div>
        </div>
    );
    case 'gpu':
        return (
            <div className="absolute inset-0 pointer-events-none">
                {/* Кулеры видеокарты */}
                <div className="absolute top-2 left-4 w-6 h-6 border-2
border-gray-400 rounded-full opacity-60">
                    <div className="absolute inset-1 border border-gray-
400 rounded-full"></div>
                </div>
                <div className="absolute top-2 right-8 w-6 h-6 border-
2 border-gray-400 rounded-full opacity-60">
                    <div className="absolute inset-1 border border-gray-
400 rounded-full"></div>
                </div>
                {/* Разъемы питания */}
                <div className="absolute top-1 right-2 w-2 h-3 bg-
yellow-500 rounded opacity-80"></div>
                {/* Выходы */}
                <div className="absolute bottom-1 right-1 w-6 h-2 bg-
blue-600 rounded opacity-80"></div>
            </div>
        );
    case 'storage':
        return (
            <div className="absolute inset-0 pointer-events-none">
                {/* Разъем SATA */}
                <div className="absolute bottom-0 left-1 w-4 h-1 bg-
red-500 rounded opacity-80"></div>
                {/* Контроллер */}
                <div className="absolute top-1 left-1 right-1 h-2 bg-
gray-600 rounded opacity-60"></div>
            </div>
        );
    case 'psu':
        return (
            <div className="absolute inset-0 pointer-events-none">
                {/* Вентилятор БП */}

```

```

        <div className="absolute inset-2 border-2 border-gray-
500 rounded-full opacity-60">
            <div className="absolute inset-1 border border-gray-
500 rounded-full"></div>
            <div className="absolute top-1/2 left-1/2 transform
-translate-x-1/2 -translate-y-1/2 w-2 h-2 bg-gray-600 rounded-
full"></div>
        </div>
        { /* Кабели */ }
        <div className="absolute top-0 left-2 w-1 h-2 bg-red-
500 rounded opacity-80"></div>
        <div className="absolute top-0 left-4 w-1 h-2 bg-
yellow-500 rounded opacity-80"></div>
    </div>
    );
    case 'cooler':
        return (
            <div className="absolute inset-0 pointer-events-none">
                { /* Лопастни кулера */ }
                <div className="absolute inset-1 border-2 border-white
rounded-full opacity-60">
                    <div className="absolute top-1/2 left-1/2 transform
-translate-x-1/2 -translate-y-1/2 w-2 h-2 bg-white rounded-
full"></div>
                    <div className="absolute top-1/2 left-1/2 transform
-translate-x-1/2 -translate-y-1/2 rotate-45">
                        <div className="w-0.5 h-4 bg-white opacity-
80"></div>
                        <div className="w-4 h-0.5 bg-white opacity-80 -mt-
2"></div>
                    </div>
                </div>
            </div>
        );
    default:
        return null;
    }
};

const getTotalPrice = () => {
    return Object.values(selectedComponents).reduce((sum,
component) => sum + component.price, 0);
};

const formatPrice = (price: number) => {
    return new Intl.NumberFormat('ru-RU', {
        style: 'currency',
        currency: 'RUB',
        minimumFractionDigits: 0,
    }).format(price);
};

return (

```

```

    <div className={`relative w-full h-full ${expanded ? 'bg-gray-
50' : ''}`}>
      {expanded && (
        <div className="absolute top-4 right-4 z-10">
          <button
            onClick={onClose}
            className="p-2 bg-white rounded-full shadow-lg
            hover:bg-gray-50 transition-colors"
          >
            <X className="w-6 h-6" />
          </button>
        </div>
      )}

      <div className="w-full h-full flex items-start justify-
center p-6">
        <div className="relative w-full max-w-4xl aspect-[16/10]">
          {/* Основной корпус компьютера */}
          <div className="relative w-full h-full bg-gradient-to-br
from-gray-900 via-gray-800 to-black rounded-3xl shadow-2xl border
border-gray-700 overflow-hidden"
            style={{
              background: 'linear-gradient(145deg, #1f2937,
#111827)',
              boxShadow: 'inset 0 2px 4px
              rgba(255,255,255,0.1), 0 25px 50px -12px rgba(0,0,0,0.25)'
            }}>

            {/* Боковые панели для создания 3D эффекта */}
            <div className="absolute inset-0 rounded-3xl" style={{
              background: 'linear-gradient(90deg, rgba(0,0,0,0.3)
              0%, transparent 10%, transparent 90%, rgba(0,0,0,0.2) 100%)'
            }}></div>

            {/* Передняя панель с вентиляционными отверстиями */}
            <div className="absolute left-4 top-6 bottom-6 w-3 bg-
            gradient-to-b from-gray-600 to-gray-800 rounded-lg shadow-
            inner"></div>

            {/* Световая полоса RGB */}
            <motion.div
              className="absolute bottom-4 left-8 right-8 h-2
              rounded-full opacity-80"
              style={{
                background: 'linear-gradient(90deg, #ff0080,
                #ff8000, #00ff80, #0080ff, #8000ff, #ff0080)',
                backgroundSize: '300% 100%'
              }}
              animate={{
                backgroundPosition: ['0% 50%', '100% 50%']
              }}
              transition={{
                duration: 3,

```

```

        repeat: Infinity,
        ease: 'linear'
      }}
    />

    { /* Вентиляционная решетка */ }
    <div className="absolute top-6 right-6 grid grid-cols-
6 gap-1 opacity-60">
      {Array.from({ length: 24 }).map((_, i) => (
        <div key={i} className="w-1.5 h-1.5 bg-gray-500
rounded-full shadow-inner"></div>
      ))}
    </div>

    { /* Схематичные компоненты */ }
    <div className="absolute inset-0 perspective-1000">
      {componentPositions.map((position, index) => {
        const status =
getComponentStatus(position.category);
        const component =
selectedComponents[position.category];
        const componentStyle =
getSchematicStyle(position.shape || 'motherboard', status,
position.depth);
        const clickable = isClickable(position.category);
        const hoverable = isHoverable(position.category);

        return (
          <motion.div
            key={` ${position.category}-${index}`}
            className="absolute"
            style={{
              left: ` ${position.x}%`,
              top: ` ${position.y}%`,
              width: ` ${position.width}%`,
              height: ` ${position.height}%`,
              ...componentStyle,
              pointerEvents: clickable ? 'auto' : 'none',
              zIndex: position.depth,
            }}
            whileHover={hoverable ? {
              scale: 1.1,
              zIndex: 20,
              transition: { duration: 0.2 }
            } : {}}
            whileTap={clickable ? { scale: 0.95 } : {}}
            onClick={clickable ? () =>
onComponentClick(position.category) : undefined}
            onMouseEnter={() =>
setHoveredComponent(position.category)}
            onMouseLeave={() => setHoveredComponent(null)}
            animate={status === 'highlighted' ? {
              scale: [1, 1.05, 1],

```

```

        transition: { duration: 1, repeat: Infinity
    }
    } : {} }
  >
  { /* Схематичные детали компонента */ }
  { renderComponentDetails(position) }

  { /* Иконка и название компонента - адаптивные
размеры */ }
    <div className="absolute top-1/2 left-1/2
transform -translate-x-1/2 -translate-y-1/2 text-white text-center
pointer-events-none z-10">
      <div className="flex flex-col items-center
justify-center">
        <div className={ `mb-0.5 opacity-90 drop-
shadow-lg ${
          expanded ? 'text-base' : 'text-xs'
        }` }>
          { React.cloneElement(position.icon as
React.ReactElement, {
            className: expanded ? 'w-6 h-6' :
position.width < 10 ? 'w-3 h-3' : 'w-4 h-4'
          }) }
        </div>
        <span className={ `font-semibold leading-
tight drop-shadow-lg ${
          expanded ? 'text-sm' : position.width <
10 ? 'text-[8px]' : 'text-xs'
        }` }>
          { expanded ? position.label :
position.label.length > 8 ?
            position.label.substring(0, 6) + '...'
          : position.label }
        </span>
      </div>
    </div>

    { /* Индикатор выбранного компонента */ }
    { component && (
      <>
        <motion.div
          className={ `absolute -top-1 -right-1 bg-
green-400 rounded-full border-2 border-white shadow-lg z-20 ${
            expanded ? 'w-4 h-4' : 'w-3 h-3'
          }` }
          animate={{ scale: [1, 1.2, 1] }}
          transition={{ duration: 2, repeat:
Infinity }}
        </>
        <div className="absolute inset-0 rounded
border-2 border-green-300 animate-pulse z-15"></div>
      </>
    ) }
  }

```

```

        { /* Информация о компоненте при наведении */ }
        { component && clickable && expanded && (
          <div className="absolute bottom-full left-
1/2 transform -translate-x-1/2 mb-2 px-3 py-2 bg-black/90 text-
white text-xs rounded-lg shadow-lg opacity-0 hover:opacity-100
transition-opacity duration-200 whitespace-nowrap z-30">
            <div className="font-
medium">{component.name}</div>
            <div className="text-gray-
300">{formatPrice(component.price)}</div>
            <div className="absolute top-full left-1/2
transform -translate-x-1/2 border-4 border-transparent border-t-
black/90"></div>
          </div>
        ) }

        { /* Подсветка при наведении на пустые слоты
*/ }
        { !component && status === 'highlighted' &&
clickable && (
          <div className="absolute inset-0 rounded
border-2 border-blue-400 animate-pulse z-15"></div>
        ) }

        { /* Затемнение некликабельных компонентов */ }
        { !clickable && (
          <div className="absolute inset-0 bg-black/30
rounded z-15 pointer-events-none"></div>
        ) }
      </motion.div>
    );
  }}}
</div>

{ /* Эффект питания */ }
{ Object.keys(selectedComponents).length > 0 && (
  <>
    { /* Пульсирующий свет */ }
    <motion.div
      className="absolute inset-0 pointer-events-none
rounded-3xl"
      animate={{
        boxShadow: [
          '0 0 20px rgba(59, 130, 246, 0.1)',
          '0 0 40px rgba(59, 130, 246, 0.2)',
          '0 0 20px rgba(59, 130, 246, 0.1)'
        ]
      }}
      transition={{ duration: 3, repeat: Infinity }}
    />

    { /* LED индикатор питания */ }

```

```

        <motion.div
            className="absolute top-6 left-6 w-3 h-3 bg-
green-400 rounded-full shadow-lg"
            animate={{ opacity: [0.5, 1, 0.5] }}
            transition={{ duration: 2, repeat: Infinity }}
        />
    </>
    )}
</div>

    {/* Подсказка для текущего компонента */}
    {!expanded && (
        <div className="absolute -top-12 left-0 right-0 text-
center">
            <p className="text-sm text-gray-600 bg-white px-4
py-2 rounded-lg shadow-sm inline-block">
                {selectedComponents[selectedCategory]}
                ? `✓
${selectedComponents[selectedCategory].name}`
                : `Выберите
${getCategoryDisplayName(selectedCategory)}`
            </p>
        </div>
    )}
</div>
</div>

    {/* Информационная панель - перенесена ниже */}
    <div className="p-4 mb-4 bg-white rounded-2xl shadow-xl
border p-4">
        <div className="flex justify-between items-center">
            <div className="flex items-center space-x-4">
                <div className="text-sm">
                    <span className="text-gray-600">Компонентов:</span>
                    <span className="font-bold text-gray-900 ml-2">
                        {Object.keys(selectedComponents).length}/7
                    </span>
                </div>
                <div className="w-px h-6 bg-gray-300"></div>
                <div className="text-sm">
                    <span className="text-gray-600">Общая
стоимость:</span>
                    <span className="font-bold text-green-600 ml-2">
                        {formatPrice(getTotalPrice())}
                    </span>
                </div>
            </div>
        </div>

        {/* Индикаторы статуса */}
        <div className="flex items-center space-x-2">
            <div className={`flex items-center space-x-1 px-3 py-1
rounded-full text-xs font-medium ${`

```

```

        Object.keys(selectedComponents).length >= 5
        ? 'bg-green-100 text-green-800'
        : 'bg-yellow-100 text-yellow-800'
    }`}>
    <div className={`w-2 h-2 rounded-full ${
        Object.keys(selectedComponents).length >= 5 ? 'bg-
green-400' : 'bg-yellow-400'
    }`}></div>
    {Object.keys(selectedComponents).length >= 5 ?
'Готов к сборке' : 'Требуется компоненты'}
    </div>
</div>
</div>

    {/* Прогресс-бар */}
    <div className="mt-3 w-full bg-gray-200 rounded-full h-2
overflow-hidden">
        <motion.div
            className="h-full bg-gradient-to-r from-blue-400 to-
green-400 rounded-full"
            initial={{ width: 0 }}
            animate={{ width:
`${(Object.keys(selectedComponents).length / 7) * 100}%` }}
            transition={{ duration: 0.5 }}
        />
    </div>
</div>
</div>
);
}

function getCategoryDisplayName(category: string): string {
    const names: Record<string, string> = {
        cpu: 'процессор',
        motherboard: 'материнскую плату',
        ram: 'оперативную память',
        gpu: 'видеокарту',
        storage: 'накопитель',
        psu: 'блок питания',
        cooler: 'охлаждение',
        case: 'корпус'
    };
    return names[category] || 'компонент';
}

```

Листинг В.7 – Файл Footer.tsx

```

import React from 'react';
import { Monitor } from 'lucide-react';

export default function Footer() {
    return (
        <footer className="bg-gray-900 text-white">
            <div className="container py-12">

```

```

<div className="grid grid-cols-1 md:grid-cols-4 gap-8">
  {/* Логотип и описание */}
  <div className="col-span-1 md:col-span-2">
    <div className="flex items-center space-x-2 mb-4">
      <Monitor className="h-8 w-8 text-blue-400" />
      <span className="text-xl font-bold">PC
Builder</span>
    </div>
    <p className="text-gray-400 max-w-md">
      Современный configurator персональных компьютеров с
проверкой
      совместимости компонентов и отслеживанием наличия.
    </p>
  </div>

  {/* Быстрые ссылки */}
  <div>
    <h3 className="text-lg font-semibold mb-
4">Навигация</h3>
    <ul className="space-y-2">
      <li>
        <a href="/" className="text-gray-400 hover:text-
white transition-colors">
          Конфигуратор
        </a>
      </li>
      <li>
        <a href="/about" className="text-gray-400
hover:text-white transition-colors">
          О проекте
        </a>
      </li>
      <li>
        <a href="/accessories" className="text-gray-400
hover:text-white transition-colors">
          Аксессуары
        </a>
      </li>
      <li>
        <a href="/software" className="text-gray-400
hover:text-white transition-colors">
          Программы
        </a>
      </li>
    </ul>
  </div>

  {/* Информация */}
  <div>
    <h3 className="text-lg font-semibold mb-4">Проект</h3>
    <ul className="space-y-2">
      <li>

```

```

        <span className="text-gray-400">Дипломная
работа</span>
      </li>
      <li>
        <span className="text-gray-400">Бакалавриат</span>
      </li>
      <li>
        <span className="text-gray-400">2025 год</span>
      </li>
    </ul>
  </div>
</div>

  { /* Нижняя часть */ }
  <div className="border-t border-gray-800 mt-8 pt-8">
    <div className="flex flex-col md:flex-row justify-
between items-center">
      <p className="text-gray-400 text-sm">
        © 2025 PC Builder. Дипломный проект.
      </p>
      <p className="text-gray-400 text-sm mt-2 md:mt-0">
        Создано с использованием Next.js и FastAPI
      </p>
    </div>
  </div>
</div>
</footer>
);
}

```

Листинг В.8 – Файл Header.tsx

```

import React from 'react';
import Link from 'next/link';
import { Monitor, Menu, X } from 'lucide-react';
import { useState } from 'react';

export default function Header() {
  const [isMenuOpen, setIsMenuOpen] = useState(false);

  const navigation = [
    { name: 'Конфигуратор', href: '/' },
    { name: 'Компоненты', href: '/components' },
    { name: 'Аксессуары', href: '/accessories' },
    { name: 'Импорт', href: '/import' },
    { name: 'О проекте', href: '/about' },
  ];

  return (
    <header className="bg-white/95 backdrop-blur-sm border-b
border-gray-200 sticky top-0 z-50">
      <div className="container">
        <div className="flex items-center justify-between h-16">
          { /* Логотип */ }

```

```

        <Link href="/" className="flex items-center space-x-2">
            <Monitor className="h-8 w-8 text-blue-600" />
            <span className="text-xl font-bold text-gray-900">PC
Builder</span>
        </Link>

        {/* Навигация для десктопа */}
        <nav className="hidden md:flex items-center space-x-8">
            {navigation.map((item) => (
                <Link
                    key={item.name}
                    href={item.href}
                    className="text-gray-600 hover:text-blue-600
transition-colors duration-200 font-medium"
                >
                    {item.name}
                </Link>
            ))}
        </nav>

        {/* Мобильное меню кнопка */}
        <button
            onClick={() => setIsMenuOpen(!isMenuOpen)}
            className="md:hidden p-2 rounded-md text-gray-600
hover:text-blue-600 hover:bg-gray-100"
        >
            {isMenuOpen ? <X className="h-6 w-6" /> : <Menu
className="h-6 w-6" />}
        </button>
    </div>

    {/* Мобильная навигация */}
    {isMenuOpen && (
        <div className="md:hidden py-4 border-t border-gray-
200">
            <nav className="flex flex-col space-y-2">
                {navigation.map((item) => (
                    <Link
                        key={item.name}
                        href={item.href}
                        className="px-4 py-2 text-gray-600 hover:text-
blue-600 hover:bg-gray-50 rounded-md transition-colors duration-
200"
                        onClick={() => setIsMenuOpen(false)}
                    >
                        {item.name}
                    </Link>
                ))}
            </nav>
        </div>
    )}
</div>
</header>

```

```
);
}
```

Листинг В.9 – Файл index.tsx

```
import React from 'react';
import Header from './Header';
import Footer from './Footer';

interface LayoutProps {
  children: React.ReactNode;
}

export default function Layout({ children }: LayoutProps) {
  return (
    <div className="min-h-screen flex flex-col">
      <Header />
      <main className="flex-1">
        {children}
      </main>
      <Footer />
    </div>
  );
}
```

Листинг В.10 – Файл useConfiguratorStore.ts

```
import { create } from 'zustand';
import { Component, Configuration, ConfiguratorState,
CompatibilityCheck } from '../types';
import { checkCompatibilityAPI, configurationsApi } from
'../lib/api';

interface ConfiguratorStore extends ConfiguratorState {
  // Actions
  addComponent: (component: Component) => void;
  removeComponent: (categorySlug: string) => void;
  clearConfiguration: () => void;
  checkCompatibility: () => Promise<void>;
  saveConfiguration: (name: string, description?: string) =>
Promise<void>;
  addAccessoriesToConfiguration: (accessories: Array<{component:
Component, quantity: number}>)> => Promise<void>;
  exportToPdf: () => Promise<Blob | null>;
  setLoading: (loading: boolean) => void;
  setError: (error: string | null) => void;
}

export const useConfiguratorStore =
create<ConfiguratorStore>((set, get) => ({
  // Initial state
  selectedComponents: {},
  currentConfiguration: undefined,
  isLoading: false,
```

```

error: undefined,

// Actions
addComponent: (component: Component) => {
  const state = get();
  const newSelectedComponents = {
    ...state.selectedComponents,
    [component.category.slug]: component,
  };

  set({
    selectedComponents: newSelectedComponents,
    error: undefined,
  });
},

removeComponent: (categorySlug: string) => {
  const state = get();
  const newSelectedComponents = { ...state.selectedComponents };
  delete newSelectedComponents[categorySlug];

  set({
    selectedComponents: newSelectedComponents,
    error: undefined,
  });
},

clearConfiguration: () => {
  set({
    selectedComponents: {},
    currentConfiguration: undefined,
    error: undefined,
  });
},

checkCompatibility: async () => {
  const state = get();
  const componentIds =
Object.values(state.selectedComponents).map(c => c.id);

  if (componentIds.length < 2) {
    return; // Не проверяем совместимость для менее чем 2
компонентов
  }

  set({ isLoading: true });

  try {
    const compatibilityResult = await
checkCompatibilityAPI(componentIds);

    // Обновляем конфигурацию с результатами проверки
совместимости

```

```

    const newConfiguration: Configuration = {
      id: '0',
      name: 'Текущая конфигурация',
      description: '',
      total_price:
Object.values(state.selectedComponents).reduce((sum, c) => sum +
c.price, 0),
      currency: 'RUB',
      status: 'draft',
      compatibility_status: compatibilityResult.status,
      compatibility_issues: compatibilityResult.issues,
      availability_status: 'unknown',
      created_at: new Date().toISOString(),
      items:
Object.values(state.selectedComponents).map((component, index) =>
({
      id: `temp-${index}`,
      component,
      quantity: 1,
      price_snapshot: component.price,
      order_priority: index,
      added_at: new Date().toISOString(),
    })),
    };

    set({
      currentConfiguration: newConfiguration,
      isLoading: false,
      error: undefined,
    });
  } catch (error) {
    set({
      isLoading: false,
      error: error instanceof Error ? error.message : 'Ошибка
проверки совместимости',
    });
  }
},

saveConfiguration: async (name: string, description?: string) =>
{
  const state = get();
  const components = Object.values(state.selectedComponents);

  if (components.length === 0) {
    set({ error: 'Нет компонентов для сохранения' });
    return;
  }

  set({ isLoading: true });

  try {
    // 1. Создаем базовую конфигурацию

```

```

const savedConfig = await configurationsApi.create({
  name,
  description,
});

// 2. Добавляем каждый компонент в конфигурацию
for (const component of components) {
  await configurationsApi.addComponent(savedConfig.id,
component.id, 1);
}

// 3. Проверяем совместимость сохраненной конфигурации
const compatibilityResult = await
configurationsApi.checkCompatibility(savedConfig.id);

// 4. Получаем обновленную конфигурацию
const updatedConfig = await
configurationsApi.getById(savedConfig.id);

set({
  currentConfiguration: updatedConfig,
  isLoading: false,
  error: undefined,
});
} catch (error) {
  set({
    isLoading: false,
    error: error instanceof Error ? error.message : 'Ошибка
сохранения конфигурации',
  });
}
},

  addAccessoriesToConfiguration: async (accessories:
Array<{component: Component, quantity: number}>) => {
    const state = get();

    if (!state.currentConfiguration?.id ||
state.currentConfiguration.id === '0') {
      set({ error: 'Сначала необходимо сохранить конфигурацию' });
      return;
    }

    set({ isLoading: true });

    try {
      for (const { component, quantity } of accessories) {
        await
configurationsApi.addAccessory(state.currentConfiguration.id,
component.id, quantity);
      }

      // Получаем обновленную конфигурацию

```

```

    const updatedConfig = await
configurationsApi.getById(state.currentConfiguration.id);

    set({
      currentConfiguration: updatedConfig,
      isLoading: false,
      error: undefined,
    });
  } catch (error) {
    set({
      isLoading: false,
      error: error instanceof Error ? error.message : 'Ошибка
добавления аксессуаров',
    });
  }
},

exportToPdf: async () => {
  const state = get();

  if (!state.currentConfiguration?.id ||
state.currentConfiguration.id === '0') {
    set({ error: 'Сначала необходимо сохранить конфигурацию' });
    return null;
  }

  set({ isLoading: true });

  try {
    const pdfBlob = await
configurationsApi.exportToPdf(state.currentConfiguration.id);

    set({
      isLoading: false,
      error: undefined,
    });

    return pdfBlob;
  } catch (error) {
    set({
      isLoading: false,
      error: error instanceof Error ? error.message : 'Ошибка
экспорта в PDF',
    });
    return null;
  }
},

setLoading: (loading: boolean) => {
  set({ isLoading: loading });
},

setError: (error: string | null) => {

```

```

    set({ error: error || undefined });
  },
  ));

```

Листинг В.11 – Файл _app.tsx

```

import React, { useState } from 'react';
import type { AppProps } from 'next/app';
import { QueryClient, QueryClientProvider } from 'react-query';
import '../styles/globals.css';

export default function App({ Component, pageProps }: AppProps) {
  const [queryClient] = useState(() => new QueryClient({
    defaultOptions: {
      queries: {
        refetchOnWindowFocus: false,
        retry: 1,
        staleTime: 5 * 60 * 1000, // 5 минут
      },
    },
  }));

  return (
    <QueryClientProvider client={queryClient}>
      <Component {...pageProps} />
    </QueryClientProvider>
  );
}

```

Листинг В.12 – Файл about.tsx

```

import React from 'react';
import Head from 'next/head';
import Layout from '../components/Layout';
import {
  Monitor,
  Code,
  Database,
  Palette,
  CheckCircle,
  Users,
  Zap,
  Shield,
  Globe,
  Settings
} from 'lucide-react';

export default function About() {
  const technologies = [
    {
      name: 'Frontend',
      items: [
        { name: 'Next.js 14', description: 'React фреймворк с SSR'

```

```

        { name: 'TypeScript', description: 'Статическая типизация'
    },
    { name: 'Tailwind CSS', description: 'Utility-first CSS
    фреймворк' },
    { name: 'Framer Motion', description: 'Анимации и
    интерактивность' },
    { name: 'React Query', description: 'Управление состоянием
    данных' },
    { name: 'Zustand', description: 'Глобальное состояние
    приложения' },
    ],
    icon: <Monitor className="w-6 h-6" />,
    color: 'from-blue-500 to-cyan-500',
  },
  {
    name: 'Backend',
    items: [
      { name: 'Python 3.11+', description: 'Основной язык
    разработки' },
      { name: 'FastAPI', description: 'Современный веб-
    фреймворк' },
      { name: 'PostgreSQL', description: 'Реляционная база
    данных' },
      { name: 'SQLAlchemy', description: 'ORM для работы с БД'
    },
      { name: 'Pydantic', description: 'Валидация данных' },
      { name: 'Alembic', description: 'Миграции базы данных' },
    ],
    icon: <Database className="w-6 h-6" />,
    color: 'from-green-500 to-emerald-500',
  },
  {
    name: 'DevOps & Tools',
    items: [
      { name: 'Docker', description: 'Контейнеризация
    приложения' },
      { name: 'Git', description: 'Система контроля версий' },
      { name: 'ESLint', description: 'Статический анализ кода'
    },
      { name: 'Prettier', description: 'Форматирование кода' },
      { name: 'Swagger/OpenAPI', description: 'Документация API'
    },
    ],
    icon: <Settings className="w-6 h-6" />,
    color: 'from-purple-500 to-pink-500',
  },
];

const features = [
  {
    icon: <Zap className="w-8 h-8 text-yellow-500" />,
    title: 'Проверка совместимости',
  },
];

```

```

        description: 'Автоматическая проверка совместимости
выбранных компонентов с учетом характеристик',
    },
    {
        icon: <Shield className="w-8 h-8 text-green-500" />,
        title: 'Отслеживание наличия',
        description: 'Мониторинг наличия компонентов в режиме
реального времени',
    },
    {
        icon: <Globe className="w-8 h-8 text-blue-500" />,
        title: 'Современный интерфейс',
        description: 'Адаптивный дизайн с поддержкой мобильных
устройств',
    },
    {
        icon: <Code className="w-8 h-8 text-purple-500" />,
        title: 'Экспорт конфигураций',
        description: 'Сохранение и экспорт конфигураций в различных
форматах',
    },
    ],
];

return (
    <>
        <Head>
            <title>0 проекте - PC Configurator</title>
            <meta name="description" content="Информация о дипломном
проекте веб-конфигуратора персональных компьютеров" />
        </Head>

        <Layout>
            <div className="min-h-screen bg-gradient-to-br from-slate-
50 via-white to-slate-100">
                {/* Hero секция */}
                <section className="pt-20 pb-16">
                    <div className="container">
                        <div className="text-center mb-16">
                            <h1 className="text-4xl md:text-6xl font-bold
text-gray-900 mb-6">
                                0 проекте
                                <span className="text-gradient"> PC
Builder</span>
                            </h1>
                            <p className="text-xl text-gray-600 max-w-3xl mx-
auto">
                                Дипломный проект бакалавра по созданию
современного веб-конфигуратора
                                персональных компьютеров с проверкой
совместимости компонентов
                            </p>
                        </div>
                    </div>
                </div>
            </div>
        </>
    );

```

```

</section>

{/* Описание проекта */}
<section className="py-16">
  <div className="container">
    <div className="grid grid-cols-1 lg:grid-cols-2 gap-
12 items-center">
      <div>
        <h2 className="text-3xl font-bold text-gray-900
mb-6">
          Цель проекта
        </h2>
        <div className="space-y-4 text-gray-600">
          <p>
            Создание современного веб-приложения для
конфигурирования персональных
компьютеров, которое поможет пользователям
подобрать совместимые
компоненты и собрать оптимальную
конфигурацию ПК.
          </p>
          <p>
            Проект решает проблему сложности выбора
совместимых компонентов при
самостоятельной сборке компьютера,
предоставляя интуитивный интерфейс
и автоматическую проверку совместимости.
          </p>
          <p>
            Дипломная работа демонстрирует применение
современных технологий
веб-разработки для создания практически
значимого продукта.
          </p>
        </div>
      </div>
      <div className="bg-white p-8 rounded-2xl shadow-
xl">
        <div className="grid grid-cols-2 gap-6">
          <div className="text-center">
            <div className="text-3xl font-bold text-
blue-600">2025</div>
            <div className="text-sm text-gray-600">Год
разработки</div>
          </div>
          <div className="text-center">
            <div className="text-3xl font-bold text-
green-600">8+</div>
            <div className="text-sm text-gray-
600">Категорий компонентов</div>
          </div>
          <div className="text-center">

```



```

</section>

{/* Технологии */}
<section className="py-16">
  <div className="container">
    <div className="text-center mb-12">
      <h2 className="text-3xl font-bold text-gray-900
mb-4">
        Технологический стек
      </h2>
      <p className="text-xl text-gray-600">
        Современные технологии для создания
качественного продукта
      </p>
    </div>

    <div className="grid grid-cols-1 lg:grid-cols-3 gap-
8">
      {technologies.map((tech, index) => (
        <div key={index} className="card p-6">
          <div className="flex items-center mb-6">
            <div className={`p-3 rounded-lg bg-gradient-
to-r ${tech.color} text-white mr-4`}>
              {tech.icon}
            </div>
            <h3 className="text-xl font-bold text-gray-
900">
              {tech.name}
            </h3>
          </div>

          <div className="space-y-3">
            {tech.items.map((item, itemIndex) => (
              <div key={itemIndex} className="flex
items-start space-x-3">
                <CheckCircle className="w-5 h-5 text-
green-500 mt-0.5 flex-shrink-0" />
                <div>
                  <div className="font-medium text-gray-
900">
                    {item.name}
                  </div>
                  <div className="text-sm text-gray-
600">
                    {item.description}
                  </div>
                </div>
              </div>
            )})}
          </div>
        </div>
      )})}
    </div>
  </div>

```

```

        </div>
    </section>

    {/* Архитектура */}
    <section className="py-16 bg-gray-50">
        <div className="container">
            <div className="text-center mb-12">
                <h2 className="text-3xl font-bold text-gray-900
mb-4">
                    Архитектура приложения
                </h2>
                <p className="text-xl text-gray-600">
                    Современная микросервисная архитектура с
разделением frontend и backend
                </p>
            </div>

            <div className="bg-white rounded-2xl p-8 shadow-lg">
                <div className="grid grid-cols-1 md:grid-cols-3
gap-8">
                    <div className="text-center">
                        <div className="w-20 h-20 bg-blue-100 rounded-
full flex items-center justify-center mx-auto mb-4">
                            <Monitor className="w-10 h-10 text-blue-600"
/>
                        </div>
                        <h3 className="text-lg font-semibold text-
gray-900 mb-2">
                            Frontend (Next.js)
                        </h3>
                        <p className="text-gray-600 text-sm">
                            Пользовательский интерфейс с серверным
рендерингом,
                            современными компонентами и адаптивным
дизайном
                        </p>
                    </div>

                    <div className="text-center">
                        <div className="w-20 h-20 bg-green-100
rounded-full flex items-center justify-center mx-auto mb-4">
                            <Database className="w-10 h-10 text-green-
600" />
                        </div>
                        <h3 className="text-lg font-semibold text-
gray-900 mb-2">
                            Backend (FastAPI)
                        </h3>
                        <p className="text-gray-600 text-sm">
                            REST API с автоматической документацией,
валидацией данных
                            и логикой проверки совместимости
                        </p>
                    </div>
                </div>
            </div>
        </div>
    </section>

```

```

</div>

<div className="text-center">
  <div className="w-20 h-20 bg-purple-100
rounded-full flex items-center justify-center mx-auto mb-4">
    <Settings className="w-10 h-10 text-purple-
600" />
  </div>
  <h3 className="text-lg font-semibold text-
gray-900 mb-2">
    Database (PostgreSQL)
  </h3>
  <p className="text-gray-600 text-sm">
    Реляционная база данных для хранения
информации о компонентах,
    совместимости и конфигурациях
  </p>
</div>
</div>
</div>
</div>
</div>
</section>

{/* О разработчике */}
<section className="py-16">
  <div className="container">
    <div className="text-center mb-12">
      <h2 className="text-3xl font-bold text-gray-900
mb-4">
        О разработчике
      </h2>
    </div>

    <div className="max-w-4xl mx-auto">
      <div className="card p-8 text-center">
        <div className="w-32 h-32 bg-gradient-to-r from-
blue-500 to-purple-500 rounded-full flex items-center justify-
center mx-auto mb-6">
          <Users className="w-16 h-16 text-white" />
        </div>

        <h3 className="text-2xl font-bold text-gray-900
mb-4">
          Дипломный проект Анохина М.А.
        </h3>

        <div className="text-gray-600 space-y-2 mb-6">
          <p>Специальность: Программная инженерия</p>
          <p>Год защиты: 2025</p>
        </div>

        <div className="flex justify-center space-x-4">

```



```

    };
    stock?: {
      status: 'in_stock' | 'expected' | 'out_of_stock';
      quantity: number;
      expected_date?: string;
    };
  }

interface AccessoryCategory {
  slug: string;
  name: string;
  count: number;
}

interface FilterOptions {
  brands: string[];
  types: string[];
  connections: string[];
  price_range: {
    min: number;
    max: number;
  };
}

export default function Accessories() {
  const [accessories, setAccessories] = useState<Accessory[]>([]);
  const [categories, setCategories] =
    useState<AccessoryCategory[]>([]);
  const [filterOptions, setFilterOptions] = useState<FilterOptions
  | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  // Фильтры
  const [selectedType, setSelectedType] = useState<string>('');
  const [selectedBrand, setSelectedBrand] = useState<string>('');
  const [selectedConnection, setSelectedConnection] =
    useState<string>('');
  const [priceRange, setPriceRange] = useState<[number,
  number]>([0, 100000]);
  const [onlyInStock, setOnlyInStock] = useState(false);
  const [searchQuery, setSearchQuery] = useState('');

  // Загрузка данных
  useEffect(() => {
    loadAccessories();
    loadCategories();
    loadFilterOptions();
  }, []);

  // Применение фильтров
  useEffect(() => {
    loadAccessories();

```

```

    }, [selectedType, selectedBrand, selectedConnection, priceRange,
onlyInStock, searchQuery]);

const loadAccessories = async () => {
  try {
    setLoading(true);
    const params = new URLSearchParams();

    if (selectedType) params.append('type_filter',
selectedType);
    if (selectedBrand) params.append('brand', selectedBrand);
    if (selectedConnection) params.append('connection',
selectedConnection);
    if (priceRange[0] > 0) params.append('price_min',
priceRange[0].toString());
    if (priceRange[1] < 100000) params.append('price_max',
priceRange[1].toString());
    if (onlyInStock) params.append('only_in_stock', 'true');
    if (searchQuery) params.append('search', searchQuery);
    params.append('limit', '50');

    const response = await
fetch(`${process.env.NEXT_PUBLIC_API_URL} ||
'http://localhost:8000'/accessories?${params}`);
    if (!response.ok) throw new Error('Ошибка загрузки
аксесуаров');

    const data = await response.json();
    setAccessories(data);
  } catch (err) {
    setError(err instanceof Error ? err.message : 'Неизвестная
ошибка');
  } finally {
    setLoading(false);
  }
};

const loadCategories = async () => {
  try {
    const response = await
fetch(`${process.env.NEXT_PUBLIC_API_URL} ||
'http://localhost:8000'/accessories/categories`);
    if (!response.ok) throw new Error('Ошибка загрузки
категорий');

    const data = await response.json();
    setCategories(data);
  } catch (err) {
    console.error('Ошибка загрузки категорий:', err);
  }
};

const loadFilterOptions = async () => {

```

```

    try {
      const response = await
fetch(`${process.env.NEXT_PUBLIC_API_URL} ||
'http://localhost:8000'/accessories/filters/options`);
      if (!response.ok) throw new Error('Ошибка загрузки опций
фильтров');

      const data = await response.json();
      setFilterOptions(data);
      setPriceRange([data.price_range.min, data.price_range.max]);
    } catch (err) {
      console.error('Ошибка загрузки опций фильтров:', err);
    }
  };

const getTypeIcon = (type: string) => {
  switch (type) {
    case 'mouse': return <Mouse className="w-5 h-5" />;
    case 'keyboard': return <Keyboard className="w-5 h-5" />;
    case 'monitor': return <Monitor className="w-5 h-5" />;
    case 'headset':
    case 'headphones': return <Headphones className="w-5 h-5"
/>;
    case 'webcam': return <Camera className="w-5 h-5" />;
    case 'mousepad': return <Gamepad2 className="w-5 h-5" />;
    default: return <Gamepad2 className="w-5 h-5" />;
  }
};

const getStockStatus = (stock: Accessory['stock']) => {
  if (!stock) return { icon: <AlertTriangle className="w-4 h-4
text-gray-400" />, text: 'Нет данных', color: 'text-gray-400' };

  switch (stock.status) {
    case 'in_stock':
      return { icon: <Check className="w-4 h-4 text-green-500"
/>, text: 'В наличии', color: 'text-green-500' };
    case 'expected':
      return { icon: <Clock className="w-4 h-4 text-yellow-500"
/>, text: 'Ожидается', color: 'text-yellow-500' };
    case 'out_of_stock':
      return { icon: <AlertTriangle className="w-4 h-4 text-red-
500" />, text: 'Нет в наличии', color: 'text-red-500' };
    default:
      return { icon: <AlertTriangle className="w-4 h-4 text-
gray-400" />, text: 'Неизвестно', color: 'text-gray-400' };
  }
};

const clearFilters = () => {
  setSelectedType('');
  setSelectedBrand('');
  setSelectedConnection('');
};

```

```

    setPriceRange(filterOptions ? [filterOptions.price_range.min,
filterOptions.price_range.max] : [0, 100000]);
    setOnlyInStock(false);
    setSearchQuery('');
  };

  return (
    <>
      <Head>
        <title>Аксессуары - PC Configurator</title>
        <meta name="description" content="Каталог аксессуаров для
ПК: мыши, клавиатуры, мониторы, наушники и другие периферийные
устройства" />
      </Head>

      <Layout>
        <div className="min-h-screen bg-gradient-to-br from-slate-
50 via-white to-slate-100">
          {/* Hero секция */}
          <section className="pt-20 pb-16">
            <div className="container">
              <div className="text-center mb-16">
                <h1 className="text-4xl md:text-6xl font-bold
text-gray-900 mb-6">
                  Аксессуары
                  <span className="text-gradient"> для ПК</span>
                </h1>
                <p className="text-xl text-gray-600 max-w-3xl mx-
auto">
                  Широкий выбор периферийных устройств: мыши,
клавиатуры, мониторы,
                  наушники и другие аксессуары для комфортной
работы и игр
                </p>
              </div>
            </div>
          </section>

          {/* Категории */}
          <section className="py-8 bg-white">
            <div className="container">
              <div className="flex flex-wrap gap-4 justify-
center">
                <button
                  onClick={() => setSelectedType('')}
                  className={`px-6 py-3 rounded-full transition-
all duration-200 ${
                    selectedType === ''
                      ? 'bg-blue-600 text-white shadow-lg'
                      : 'bg-gray-100 text-gray-700 hover:bg-gray-
200'
                  }`>
                </button>
              </div>
            </div>
          </section>
        </div>
      </>
    )
  );

```

```

        Все категории
    </button>
    {categories.map((category) => (
        <button
            key={category.slug}
            onClick={() => setSelectedType(category.slug)}
            className={`px-6 py-3 rounded-full transition-
all duration-200 flex items-center space-x-2 ${
                selectedType === category.slug
                ? 'bg-blue-600 text-white shadow-lg'
                : 'bg-gray-100 text-gray-700 hover:bg-
gray-200'
            }}
        >
            {getTypeIcon(category.slug)}
            <span>{category.name}</span>
            <span className="text-xs opacity-
75">({category.count})</span>
        </button>
    ))}
</div>
</div>
</section>

{/* Фильтры и поиск */}
<section className="py-8">
    <div className="container">
        <div className="bg-white rounded-2xl p-6 shadow-lg
mb-8">
            <div className="flex flex-wrap gap-4 items-
center">
                {/* Поиск */}
                <div className="flex-1 min-w-64">
                    <div className="relative">
                        <Search className="absolute left-3 top-1/2
transform -translate-y-1/2 text-gray-400 w-5 h-5" />
                        <input
                            type="text"
                            placeholder="Поиск аксессуаров..."
                            value={searchQuery}
                            onChange={(e) =>
setSearchQuery(e.target.value)}
                            className="w-full pl-10 pr-4 py-2 border
border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-500
focus:border-transparent"
                        />
                    </div>
                </div>
                </div>

                {/* Фильтр по бренду */}
                {filterOptions && (
                    <select
                        value={selectedBrand}

```

```

        onChange={ (e) =>
setSelectedBrand(e.target.value) }
        className="px-4 py-2 border border-gray-300
rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-
transparent"
      >
        <option value="">Все бренды</option>
        {filterOptions.brands.map((brand) => (
          <option key={brand}
value={brand}>{brand}</option>
        ))}
      </select>
    )}

    {/* Фильтр по подключению */}
    {filterOptions && (
      <select
        value={selectedConnection}
        onChange={ (e) =>
setSelectedConnection(e.target.value) }
        className="px-4 py-2 border border-gray-300
rounded-lg focus:ring-2 focus:ring-blue-500 focus:border-
transparent"
      >
        <option value="">Все типы
подключения</option>
        {filterOptions.connections.map((connection)
=> (
          <option key={connection}
value={connection}>
            {connection === 'wired' ? 'Проводное' :
connection === 'wireless' ? 'Беспроводное' : connection}
          </option>
        ))}
      </select>
    )}

    {/* Только в наличии */}
    <label className="flex items-center space-x-2
cursor-pointer">
      <input
        type="checkbox"
        checked={onlyInStock}
        onChange={ (e) =>
setOnlyInStock(e.target.checked) }
        className="w-4 h-4 text-blue-600 border-
gray-300 rounded focus:ring-blue-500"
      />
      <span className="text-sm text-gray-700">Только
в наличии</span>
    </label>

    {/* Очистить фильтры */}

```

```

        <button
            onClick={clearFilters}
            className="px-4 py-2 text-gray-600 hover:text-
blue-600 transition-colors duration-200"
        >
            ОЧИСТИТЬ
        </button>
    </div>
</div>
</div>
</section>

{ /* Список аксессуаров */ }
<section className="pb-20">
    <div className="container">
        {loading ? (
            <div className="text-center py-12">
                <div className="animate-spin rounded-full h-12
w-12 border-b-2 border-blue-600 mx-auto"></div>
                <p className="mt-4 text-gray-600">Загрузка
аксессуаров...</p>
            </div>
        ) : error ? (
            <div className="text-center py-12">
                <AlertTriangle className="w-12 h-12 text-red-500
mx-auto mb-4" />
                <p className="text-red-600">{error}</p>
            </div>
        ) : accessories.length === 0 ? (
            <div className="text-center py-12">
                <Filter className="w-12 h-12 text-gray-400 mx-
auto mb-4" />
                <p className="text-gray-600">Аксессуары не
найжены</p>
                <p className="text-sm text-gray-500 mt-
2">Попробуйте изменить фильтры поиска</p>
            </div>
        ) : (
            <div className="grid grid-cols-1 md:grid-cols-2
lg:grid-cols-3 xl:grid-cols-4 gap-6">
                {accessories.map((accessory) => {
                    const stockStatus =
getStockStatus(accessory.stock);
                    return (
                        <div key={accessory.id} className="card p-6
hover:shadow-lg transition-shadow duration-300">
                            <div className="flex items-start justify-
between mb-4">
                                <div className="flex items-center space-
x-2">
                                    {getTypeIcon(accessory.specifications.type)}

```

```

capitalize">
        <span className="text-sm text-gray-500
            {accessory.specifications.type}
        </span>
    </div>
    <div className={`flex items-center
space-x-1 ${stockStatus.color}`}>
        {stockStatus.icon}
        <span className="text-
xs">{stockStatus.text}</span>
    </div>
</div>

    <h3 className="text-lg font-semibold text-
gray-900 mb-2 line-clamp-2">
        {accessory.name}
    </h3>

    <p className="text-sm text-gray-600 mb-2">
        {accessory.brand} • {accessory.model}
    </p>

    <p className="text-sm text-gray-500 mb-4
line-clamp-2">
        {accessory.description}
    </p>

    {/* Характеристики */}
    <div className="space-y-1 mb-4 text-xs
text-gray-600">
        {accessory.specifications.connection &&
        (
            <div>Подключение:
{accessory.specifications.connection === 'wired' ? 'Проводное' :
'Беспроводное'}</div>
        )}
        {accessory.specifications.dpi && (
            <div>DPI:
{accessory.specifications.dpi.toLocaleString()}</div>
        )}
        {accessory.specifications.size_inch && (
            <div>Размер:
{accessory.specifications.size_inch}"</div>
        )}
        {accessory.specifications.resolution &&
        (
            <div>Разрешение:
{accessory.specifications.resolution}</div>
        )}
    </div>

    <div className="flex items-center justify-
between">

```

```

blue-600">
    {accessory.price.toLocaleString()} ₸
  </div>
  <button
    className="btn-secondary text-sm"
    disabled={accessory.stock?.status ===
'out_of_stock'}
  >
    <ShoppingCart className="w-4 h-4 mr-1"
  />
    {accessory.stock?.status ===
'out_of_stock' ? 'Нет в наличии' : 'В корзину'}
  </button>
</div>
</div>
);
}}}
</div>
)}
</div>
</section>
</div>
</Layout>
</>
);
}

```

Листинг В.14 – Файл components.tsx

```

import React, { useState, useEffect } from 'react';
import Head from 'next/head';
import Layout from '../components/Layout';
import {
  Cpu,
  HardDrive,
  Monitor,
  Zap,
  Box,
  Fan,
  MemoryStick,
  Gamepad2,
  ChevronDown,
  ChevronUp,
  Package
} from 'lucide-react';

interface Component {
  id: string;
  name: string;
  brand: string;
  model: string;
  description: string;
  price: number;

```

```

    category: {
      name: string;
      slug: string;
    };
    specifications: any;
    stock?: {
      status: 'in_stock' | 'expected' | 'out_of_stock';
      quantity: number;
    };
  }

interface Category {
  id: string;
  name: string;
  slug: string;
  description: string;
  icon: string;
}

export default function Components() {
  const [components, setComponents] = useState<Component[]>([]);
  const [categories, setCategories] = useState<Category[]>([]);
  const [loading, setLoading] = useState(true);
  const [expandedCategories, setExpandedCategories] =
    useState<Set<string>>(new Set());

  useEffect(() => {
    loadData();
  }, []);

  const loadData = async () => {
    try {
      setLoading(true);

      // Загружаем категории
      const categoriesResponse = await
fetch(`${process.env.NEXT_PUBLIC_API_URL} ||
'http://localhost:8000'/categories`);
      const categoriesData = await categoriesResponse.json();
      setCategories(categoriesData);

      // Загружаем все компоненты
      const componentsResponse = await
fetch(`${process.env.NEXT_PUBLIC_API_URL} ||
'http://localhost:8000'/components?limit=100`);
      const componentsData = await componentsResponse.json();
      setComponents(componentsData);

    } catch (err) {
      console.error('Ошибка загрузки данных:', err);
    } finally {
      setLoading(false);
    }
  }
}

```

```

};

const getCategoryIcon = (iconName: string) => {
  switch (iconName) {
    case 'cpu': return <Cpu className="w-6 h-6" />;
    case 'motherboard': return <Package className="w-6 h-6" />;
    case 'memory': return <MemoryStick className="w-6 h-6" />;
    case 'gpu': return <Monitor className="w-6 h-6" />;
    case 'storage': return <HardDrive className="w-6 h-6" />;
    case 'power': return <Zap className="w-6 h-6" />;
    case 'case': return <Box className="w-6 h-6" />;
    case 'fan': return <Fan className="w-6 h-6" />;
    case 'accessories': return <Gamepad2 className="w-6 h-6" />;
    default: return <Package className="w-6 h-6" />;
  }
};

const getComponentsByCategory = (categorySlug: string) => {
  return components.filter(component => component.category.slug
=== categorySlug);
};

const toggleCategory = (categorySlug: string) => {
  const newExpanded = new Set(expandedCategories);
  if (newExpanded.has(categorySlug)) {
    newExpanded.delete(categorySlug);
  } else {
    newExpanded.add(categorySlug);
  }
  setExpandedCategories(newExpanded);
};

const getStockBadge = (stock: Component['stock']) => {
  if (!stock) return <span className="px-2 py-1 text-xs bg-gray-
100 text-gray-600 rounded">Нет данных</span>;

  switch (stock.status) {
    case 'in_stock':
      return <span className="px-2 py-1 text-xs bg-green-100
text-green-700 rounded">В наличии ({stock.quantity})</span>;
    case 'expected':
      return <span className="px-2 py-1 text-xs bg-yellow-100
text-yellow-700 rounded">Ожидается</span>;
    case 'out_of_stock':
      return <span className="px-2 py-1 text-xs bg-red-100 text-
red-700 rounded">Нет в наличии</span>;
    default:
      return <span className="px-2 py-1 text-xs bg-gray-100
text-gray-600 rounded">Неизвестно</span>;
  }
};

if (loading) {

```

```

    return (
      <Layout>
        <div className="min-h-screen bg-gradient-to-br from-slate-
50 via-white to-slate-100 flex items-center justify-center">
          <div className="text-center">
            <div className="animate-spin rounded-full h-12 w-12
border-b-2 border-blue-600 mx-auto"></div>
            <p className="mt-4 text-gray-600">Загрузка
компонентов...</p>
          </div>
        </div>
      </Layout>
    );
  }

  return (
    <>
      <Head>
        <title>Все компоненты - PC Configurator</title>
        <meta name="description" content="Полный каталог
компонентов для сборки ПК" />
      </Head>

      <Layout>
        <div className="min-h-screen bg-gradient-to-br from-slate-
50 via-white to-slate-100">
          {/* Hero секция */}
          <section className="pt-20 pb-16">
            <div className="container">
              <div className="text-center mb-16">
                <h1 className="text-4xl md:text-6xl font-bold
text-gray-900 mb-6">
                  Каталог
                  <span className="text-gradient">
компонентов</span>
                </h1>
                <p className="text-xl text-gray-600 max-w-3xl mx-
auto">
                  Полный список всех доступных компонентов для
сборки персональных компьютеров
                </p>
              </div>
            </div>
          </section>

          {/* Статистика */}
          <section className="py-8 bg-white">
            <div className="container">
              <div className="grid grid-cols-2 md:grid-cols-4 gap-
6">
                <div className="text-center">
                  <div className="text-3xl font-bold text-blue-
600">{categories.length}</div>

```

```

        <div className="text-sm text-gray-
600">Категорий</div>
        </div>
        <div className="text-center">
            <div className="text-3xl font-bold text-green-
600">{components.length}</div>
            <div className="text-sm text-gray-
600">Компонентов</div>
            </div>
            <div className="text-center">
                <div className="text-3xl font-bold text-purple-
600">
                    {components.filter(c => c.stock?.status ===
'in_stock').length}
                </div>
                <div className="text-sm text-gray-600">В
наличии</div>
            </div>
            <div className="text-center">
                <div className="text-3xl font-bold text-orange-
600">
                    {new Set(components.map(c => c.brand)).size}
                </div>
                <div className="text-sm text-gray-
600">Брендов</div>
            </div>
        </div>
    </div>
</section>

    {/* Список категорий и компонентов */}
    <section className="py-16">
        <div className="container">
            <div className="space-y-6">
                {categories.map((category) => {
                    const categoryComponents =
getComponentsByCategory(category.slug);
                    const isExpanded =
expandedCategories.has(category.slug);

                    return (
                        <div key={category.id} className="bg-white
rounded-2xl shadow-lg overflow-hidden">
                            <button
                                onClick={() =>
toggleCategory(category.slug)}
                                className="w-full p-6 flex items-center
justify-between hover:bg-gray-50 transition-colors duration-200"
                            >
                                <div className="flex items-center space-x-
4">
                                    <div className="p-3 bg-blue-100 rounded-
lg text-blue-600">

```

```

        {getCategoryIcon(category.icon)}
      </div>
      <div className="text-left">
        <h3 className="text-xl font-semibold
text-gray-900">{category.name}</h3>
        <p className="text-gray-
600">{category.description}</p>
      </div>
    </div>
    <div className="flex items-center space-x-
4">
      <span className="px-3 py-1 bg-gray-100
text-gray-700 rounded-full text-sm">
        {categoryComponents.length}
        КОМПОНЕНТОВ
      </span>
      {isExpanded ? (
        <ChevronUp className="w-5 h-5 text-
gray-400" />
      ) : (
        <ChevronDown className="w-5 h-5 text-
gray-400" />
      )}
    </div>
  </button>

  {isExpanded && (
    <div className="border-t border-gray-200">
      <div className="p-6">
        <div className="grid grid-cols-1
md:grid-cols-2 lg:grid-cols-3 gap-4">
          {categoryComponents.map((component)
=> (
            <div key={component.id}
className="border border-gray-200 rounded-lg p-4 hover:shadow-md
transition-shadow duration-200">
              <div className="flex justify-
between items-start mb-2">
                <h4 className="font-semibold
text-gray-900 text-sm line-clamp-2">
                  {component.name}
                </h4>
                {getStockBadge(component.stock)}
              </div>
              <p className="text-xs text-gray-
600 mb-2">
                {component.brand} •
                {component.model}
              </p>
            </div>
          )}
        </div>
      </div>
    </div>
  )}

```



```

        setFile(selectedFile);
        setUploadStatus('idle');
        setErrorMessage('');
    } else {
        setErrorMessage('Пожалуйста, выберите PDF файл');
        setUploadStatus('error');
    }
}
};

const handleDrop = (event: React.DragEvent<HTMLDivElement>) => {
    event.preventDefault();
    const droppedFile = event.dataTransfer.files[0];
    if (droppedFile) {
        if (droppedFile.type === 'application/pdf') {
            setFile(droppedFile);
            setUploadStatus('idle');
            setErrorMessage('');
        } else {
            setErrorMessage('Пожалуйста, выберите PDF файл');
            setUploadStatus('error');
        }
    }
};

const handleDragOver = (event: React.DragEvent<HTMLDivElement>)
=> {
    event.preventDefault();
};

const handleImport = async () => {
    if (!file) return;

    setIsUploading(true);
    setUploadStatus('idle');

    try {
        // Имитация импорта (в реальности здесь был бы API вызов)
        await new Promise(resolve => setTimeout(resolve, 2000));

        // Имитация успешного импорта
        const mockConfig = {
            name: 'Импортированная конфигурация',
            components: [
                { category: 'Процессор', name: 'Intel Core i7-13700K',
price: 35990 },
                { category: 'Материнская плата', name: 'ASUS ROG STRIX
Z790-E', price: 28990 },
                { category: 'Оперативная память', name: 'Corsair
Vengeance LPX 32GB', price: 12990 },
                { category: 'Видеокарта', name: 'NVIDIA RTX 4070 Ti',
price: 89990 },
            ],

```



```

        onDrop={handleDrop}
        onDragOver={handleDragOver}
        className={`border-2 border-dashed rounded-xl
p-8 text-center transition-colors ${
          uploadStatus === 'error'
            ? 'border-red-300 bg-red-50'
            : 'border-gray-300 hover:border-blue-400
hover:bg-blue-50'
        }}
      >
        <input
          type="file"
          accept=".pdf"
          onChange={handleFileSelect}
          className="hidden"
          id="file-upload"
        />

        <label htmlFor="file-upload"
className="cursor-pointer">
          <div className="flex flex-col items-center
space-y-4">
            {uploadStatus === 'error' ? (
              <AlertCircle className="w-16 h-16 text-
red-500" />
            ) : (
              <Upload className="w-16 h-16 text-gray-
400" />
            )}

            <div>
              <p className="text-lg font-medium text-
gray-900 mb-2">
                {file ? file.name : 'Выберите PDF файл
или перетащите сюда'}
              </p>
              <p className="text-sm text-gray-500">
                Поддерживаются только PDF файлы
конфигураций
              </p>
            </div>

            {!file && (
              <button className="btn-primary">
                Выбрать файл
              </button>
            )}
          </div>
        </label>
      </div>

      {/* Сообщение об ошибке */}
      {uploadStatus === 'error' && (

```

```

        <motion.div
          initial={{ opacity: 0, y: 10 }}
          animate={{ opacity: 1, y: 0 }}
          className="mt-4 p-4 bg-red-50 border border-
red-200 rounded-lg"
        >
          <div className="flex items-center space-x-
2">
            <AlertCircle className="w-5 h-5 text-red-
500" />
            <p className="text-red-
700">{errorMessage}</p>
          </div>
        </motion.div>
      )}

      {/* Кнопка импорта */}
      {file && uploadStatus !== 'error' && (
        <motion.div
          initial={{ opacity: 0, y: 10 }}
          animate={{ opacity: 1, y: 0 }}
          className="mt-6 text-center"
        >
          <button
            onClick={handleImport}
            disabled={isUploading}
            className="btn-primary px-8 py-3 text-lg"
          >
            {isUploading ? (
              <div className="flex items-center space-
x-2">
                <div className="animate-spin rounded-
full h-5 w-5 border-b-2 border-white"></div>
                <span>Импортируем...</span>
              </div>
            ) : (
              <div className="flex items-center space-
x-2">
                <FileText className="w-5 h-5" />
                <span>Импортировать
конфигурацию</span>
              </div>
            )}
          </button>
        </motion.div>
      )}
    </>
  ) : (
    {/* Результат импорта */}
    <motion.div
      initial={{ opacity: 0, scale: 0.95 }}
      animate={{ opacity: 1, scale: 1 }}
      className="text-center"
    >

```

```

        >
        <CheckCircle className="w-16 h-16 text-green-500
mx-auto mb-4" />
        <h3 className="text-2xl font-bold text-gray-900
mb-4">
            Конфигурация успешно импортирована!
        </h3>

        {importedConfig && (
            <div className="bg-gray-50 rounded-lg p-6 mb-6
text-left">
                <h4 className="font-semibold text-gray-900
mb-3">
                    {importedConfig.name}
                </h4>
                <div className="space-y-2">
                    {importedConfig.components.map((component:
any, index: number) => (
                        <div key={index} className="flex
justify-between items-center">
                            <span className="text-sm text-gray-
600">
                                {component.category}:
                            {component.name}
                            </span>
                            <span className="text-sm font-medium">
                                {component.price.toLocaleString()} ₰
                            </span>
                        </div>
                    )))}
                </div>
                <div className="border-t mt-3 pt-3">
                    <div className="flex justify-between
items-center">
                        <span className="font-
semibold">Итого:</span>
                        <span className="text-lg font-bold text-
green-600">
                            {importedConfig.totalPrice.toLocaleString()} ₰
                        </span>
                    </div>
                </div>
            </div>
        )}

        <div className="flex space-x-4">
            <button
                onClick={() => {
                    setFile(null);
                    setUploadStatus('idle');
                    setImportedConfig(null);
                }}
            </button>
        </div>
    </div>

```

```

        className="btn-outline flex-1"
      >
        Импортировать другую
      </button>
      <button
        onClick={handleUseConfiguration}
        className="btn-primary flex-1 flex items-
center justify-center space-x-2"
      >
        <span>Использовать конфигурацию</span>
        <ArrowRight className="w-4 h-4" />
      </button>
    </div>
  </motion.div>
)}
</motion.div>

{/* Дополнительная информация */}
<motion.div
  initial={{ opacity: 0, y: 20 }}
  animate={{ opacity: 1, y: 0 }}
  transition={{ delay: 0.2 }}
  className="mt-8 text-center"
>
  <div className="bg-blue-50 rounded-lg p-6">
    <h4 className="font-semibold text-blue-900 mb-2">
      Как это работает?
    </h4>
    <div className="text-sm text-blue-700 space-y-1">
      <p>• Загрузите PDF файл с ранее экспортированной
конфигурацией</p>
      <p>• Система автоматически распознает
компоненты</p>
      <p>• Конфигурация будет восстановлена в
конфигураторе</p>
      <p>• Вы сможете продолжить редактирование или
сохранить</p>
    </div>
  </div>
</motion.div>
</motion.div>
</div>
</div>
</Layout>
);
}

```

Листинг В.16 – Файл i.tsx

```

import React from 'react';
import Head from 'next/head';
import Layout from '../components/Layout';
import Configurator from '../components/Configurator';

```

```

export default function Home() {
  return (
    <>
      <Head>
        <title>PC Configurator - Создайте свой идеальный
компьютер</title>
        <meta name="description" content="Конфигуратор
персональных компьютеров с проверкой совместимости компонентов" />
        <meta name="viewport" content="width=device-width,
initial-scale=1" />
        <link rel="icon" href="/favicon.ico" />
      </Head>

      <Layout>
        <div className="min-h-screen bg-gradient-to-br from-slate-
50 via-white to-slate-100">
          {/* Hero секция */}
          <section className="pt-20 pb-16">
            <div className="container">
              <div className="text-center mb-12">
                <h1 className="text-4xl md:text-6xl font-bold
text-gray-900 mb-6">
                  Создайте свой
                  <span className="text-gradient"> идеальный
</span>
                    компьютер
                </h1>
                <p className="text-xl text-gray-600 max-w-3xl mx-
auto">
                  Профессиональный configurator ПК с проверкой
совместимости компонентов,
                  отслеживанием наличия и современным интерфейсом
                </p>
              </div>
            </div>
          </section>

          {/* Основной configurator */}
          <section className="pb-20">
            <div className="container">
              <Configurator />
            </div>
          </section>
        </div>
      </Layout>
    </>
  );
}

```

Листинг В.17 – Файл payment.tsx

```

import React, { useState, useEffect } from 'react';
import { motion } from 'framer-motion';

```

```

import { CreditCard, Shield, CheckCircle, ArrowLeft, Package,
Truck } from 'lucide-react';
import Layout from '../components/Layout';
import { useRouter } from 'next/router';

export default function PaymentPage() {
  const [step, setStep] = useState<'form' | 'processing' |
'success'>('form');
  const [formData, setFormData] = useState({
    cardNumber: '',
    expiryDate: '',
    cvv: '',
    cardHolder: '',
    email: '',
    phone: '',
    address: '',
  });
  const router = useRouter();

  // Имитация данных заказа
  const orderData = {
    items: [
      { name: 'Intel Core i7-13700K', price: 35990 },
      { name: 'ASUS ROG STRIX Z790-E', price: 28990 },
      { name: 'Corsair Vengeance LPX 32GB', price: 12990 },
      { name: 'NVIDIA RTX 4070 Ti', price: 89990 },
    ],
    total: 167960,
    delivery: 1500,
    grandTotal: 169460
  };

  const handleInputChange = (field: string, value: string) => {
    setFormData(prev => ({ ...prev, [field]: value }));
  };

  const formatCardNumber = (value: string) => {
    const v = value.replace(/\s+/g, '').replace(/[^\d0-9]/gi, '');
    const matches = v.match(/\d{4,16}/g);
    const match = matches && matches[0] || '';
    const parts = [];
    for (let i = 0, len = match.length; i < len; i += 4) {
      parts.push(match.substring(i, i + 4));
    }
    if (parts.length) {
      return parts.join(' ');
    } else {
      return v;
    }
  };

  const formatExpiryDate = (value: string) => {
    const v = value.replace(/\s+/g, '').replace(/[^\d0-9]/gi, '');

```

```

    if (v.length >= 2) {
      return v.substring(0, 2) + '/' + v.substring(2, 4);
    }
    return v;
  };

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setStep('processing');

  // Имитация обработки платежа
  await new Promise(resolve => setTimeout(resolve, 3000));
  setStep('success');
};

const isFormValid = formData.cardNumber.length >= 19 &&
  formData.expiryDate.length >= 5 &&
  formData.cvv.length >= 3 &&
  formData.cardHolder.length >= 2 &&
  formData.email.length >= 5 &&
  formData.phone.length >= 10;

return (
  <Layout>
    <div className="min-h-screen bg-gradient-to-br from-green-50
to-blue-50 py-12">
      <div className="container mx-auto px-4">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="max-w-4xl mx-auto"
        >
          {step === 'form' && (
            <>
              {/* Заголовок */}
              <div className="text-center mb-8">
                <h1 className="text-3xl font-bold text-gray-900
mb-4">
                  Оформление заказа
                </h1>
                <p className="text-gray-600">
                  Завершите покупку вашей конфигурации ПК
                </p>
              </div>

              <div className="grid grid-cols-1 lg:grid-cols-2
gap-8">
                {/* Форма оплаты */}
                <motion.div
                  initial={{ opacity: 0, x: -20 }}
                  animate={{ opacity: 1, x: 0 }}
                  className="bg-white rounded-2xl shadow-xl p-8"
                >

```

```

        <div className="flex items-center space-x-3
mb-6">
            <CreditCard className="w-6 h-6 text-blue-
600" />
            <h2 className="text-xl font-semibold text-
gray-900">
                Данные для оплаты
            </h2>
        </div>

        <form onSubmit={handleSubmit}
className="space-y-6">
            {/* Номер карты */}
            <div>
                <label className="block text-sm font-
medium text-gray-700 mb-2">
                    Номер карты
                </label>
                <input
                    type="text"
                    value={formData.cardNumber}
                    onChange={ (e) =>
handleInputChange('cardNumber', formatCardNumber(e.target.value))}
                    placeholder="1234 5678 9012 3456"
                    maxLength={19}
                    className="input w-full"
                    required
                />
            </div>

            <div className="grid grid-cols-2 gap-4">
                {/* Срок действия */}
                <div>
                    <label className="block text-sm font-
medium text-gray-700 mb-2">
                        Срок действия
                    </label>
                    <input
                        type="text"
                        value={formData.expiryDate}
                        onChange={ (e) =>
handleInputChange('expiryDate', formatExpiryDate(e.target.value))}
                        placeholder="MM/YY"
                        maxLength={5}
                        className="input w-full"
                        required
                    />
                </div>

                {/* CVV */}
                <div>
                    <label className="block text-sm font-
medium text-gray-700 mb-2">

```

```

        CVV
      </label>
      <input
        type="text"
        value={formData.cvv}
        onChange={ (e) =>
handleInputChange('cvv', e.target.value.replace(/\D/g, ''))}
        placeholder="123"
        maxLength={3}
        className="input w-full"
        required
      />
    </div>
  </div>

  { /* Имя держателя карты */ }
  <div>
    <label className="block text-sm font-
medium text-gray-700 mb-2">
      Имя держателя карты
    </label>
    <input
      type="text"
      value={formData.cardHolder}
      onChange={ (e) =>
handleInputChange('cardHolder', e.target.value)}
      placeholder="IVAN PETROV"
      className="input w-full"
      required
    />
  </div>

  { /* Email */ }
  <div>
    <label className="block text-sm font-
medium text-gray-700 mb-2">
      Email для чека
    </label>
    <input
      type="email"
      value={formData.email}
      onChange={ (e) =>
handleInputChange('email', e.target.value)}
      placeholder="ivan@example.com"
      className="input w-full"
      required
    />
  </div>

  { /* Телефон */ }
  <div>
    <label className="block text-sm font-
medium text-gray-700 mb-2">

```

```

        Телефон
      </label>
      <input
        type="tel"
        value={formData.phone}
        onChange={ (e) =>
handleInputChange('phone', e.target.value)}
        placeholder="+7 (999) 123-45-67"
        className="input w-full"
        required
      />
    </div>

    { /* Адрес доставки */ }
    <div>
      <label className="block text-sm font-
medium text-gray-700 mb-2">
        Адрес доставки
      </label>
      <textarea
        value={formData.address}
        onChange={ (e) =>
handleInputChange('address', e.target.value)}
        placeholder="Москва, ул. Примерная, д.
123, кв. 45"
        className="input w-full h-20 resize-
none"
        required
      />
    </div>

    { /* Безопасность */ }
    <div className="flex items-center space-x-2
text-sm text-gray-600 bg-gray-50 p-3 rounded-lg">
      <Shield className="w-4 h-4 text-green-600"
/ >
      <span>Ваши данные защищены SSL-
шифрованием</span>
    </div>

    { /* Кнопка оплаты */ }
    <button
      type="submit"
      disabled={!isFormValid}
      className="btn-primary w-full py-4 text-
lg"
    >
      Оплатить
      {orderData.grandTotal.toLocaleString()} Р
    </button>
  </form>
</motion.div>

```

```

        { /* Сводка заказа */ }
        <motion.div
          initial={{ opacity: 0, x: 20 }}
          animate={{ opacity: 1, x: 0 }}
          className="bg-white rounded-2xl shadow-xl p-8"
        >
          <h2 className="text-xl font-semibold text-
gray-900 mb-6">
            Ваш заказ
          </h2>

          <div className="space-y-4">
            {orderData.items.map((item, index) => (
              <div key={index} className="flex justify-
between items-center py-2 border-b">
                <span className="text-gray-
700">{item.name}</span>
                <span className="font-
medium">{item.price.toLocaleString()} ₸</span>
              </div>
            ))}

            <div className="flex justify-between items-
center py-2">
              <span className="text-gray-
700">Доставка</span>
              <span className="font-
medium">{orderData.delivery.toLocaleString()} ₸</span>
            </div>

            <div className="flex justify-between items-
center py-3 border-t-2 text-lg font-bold">
              <span>Итого</span>
              <span className="text-green-
600">{orderData.grandTotal.toLocaleString()} ₸</span>
            </div>
          </div>

          { /* Информация о доставке */ }
          <div className="mt-6 p-4 bg-blue-50 rounded-
lg">
            <div className="flex items-center space-x-2
mb-2">
              <Truck className="w-5 h-5 text-blue-600"
/>
              <span className="font-medium text-blue-
900">Доставка</span>
            </div>
            <p className="text-sm text-blue-700">
              Бесплатная доставка при заказе от 150 000
₸
            </p>
            <p className="text-sm text-blue-700">

```

```

        Срок доставки: 2-3 рабочих дня
      </p>
    </div>
  </motion.div>
</div>
</>
)}

{step === 'processing' && (
  <motion.div
    initial={{ opacity: 0, scale: 0.9 }}
    animate={{ opacity: 1, scale: 1 }}
    className="text-center py-16"
  >
    <div className="bg-white rounded-2xl shadow-xl p-
12 max-w-md mx-auto">
      <div className="animate-spin rounded-full h-16
w-16 border-b-4 border-blue-600 mx-auto mb-6"></div>
      <h2 className="text-2xl font-bold text-gray-900
mb-4">
        Обработка платежа
      </h2>
      <p className="text-gray-600">
        Пожалуйста, подождите. Не закрывайте страницу.
      </p>
    </div>
  </motion.div>
)}

{step === 'success' && (
  <motion.div
    initial={{ opacity: 0, scale: 0.9 }}
    animate={{ opacity: 1, scale: 1 }}
    className="text-center py-16"
  >
    <div className="bg-white rounded-2xl shadow-xl p-
12 max-w-2xl mx-auto">
      <CheckCircle className="w-20 h-20 text-green-500
mx-auto mb-6" />
      <h2 className="text-3xl font-bold text-gray-900
mb-4">
        Заказ успешно оформлен!
      </h2>
      <p className="text-gray-600 mb-8">
        Номер заказа: <span className="font-mono font-
bold">#PC-2024-001</span>
      </p>

      <div className="bg-gray-50 rounded-lg p-6 mb-8
text-left">
        <h3 className="font-semibold text-gray-900 mb-
4">Что дальше?</h3>
        <div className="space-y-3">

```

```

        <div className="flex items-center space-x-
3">
            <div className="w-6 h-6 bg-green-500
rounded-full flex items-center justify-center">
                <span className="text-white text-sm
font-bold">1</span>
            </div>
            <span className="text-gray-700">Мы
отправили подтверждение на ваш email</span>
        </div>
        <div className="flex items-center space-x-
3">
            <div className="w-6 h-6 bg-blue-500
rounded-full flex items-center justify-center">
                <span className="text-white text-sm
font-bold">2</span>
            </div>
            <span className="text-gray-700">Сборка и
тестирование займет 1-2 дня</span>
        </div>
        <div className="flex items-center space-x-
3">
            <div className="w-6 h-6 bg-purple-500
rounded-full flex items-center justify-center">
                <span className="text-white text-sm
font-bold">3</span>
            </div>
            <span className="text-gray-700">Доставка
курьером в течение 2-3 дней</span>
        </div>
        </div>
        <div className="flex space-x-4">
            <button
                onClick={() => router.push('/') }
                className="btn-outline flex-1 flex items-
center justify-center space-x-2"
            >
                <ArrowLeft className="w-4 h-4" />
                <span>На главную</span>
            </button>
        </div>
    </div>
</motion.div>
    )}
</motion.div>
</div>
</div>
</Layout>
);
}

```

Приложение Г
(рекомендуемое)
Результаты тестирования серверной части

GET /api/v1/categories Get Categories

Получить все категории компонентов

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	Successful Response	No links

Media type:

Controls Accept header.

Example Value | Schema

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "name": "string",
  "slug": "string",
  "description": "string",
  "order_priority": 0,
  "icon": "string",
  "created_at": "2025-06-08T21:33:20.879Z",
  "updated_at": "2025-06-08T21:33:20.879Z"
}
```

Рисунок Г.1 – Получить все категории компонентов

GET /api/v1/categories/{category_id} Get Category

Получить категорию по ID

Parameters Try it out

Name	Description
category_id * required integer (path)	<input type="text" value="10"/>

Responses

Code	Description	Links
200	Successful Response	No links

Media type:

Controls Accept header.

Example Value | Schema

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "name": "string",
  "slug": "string",
  "description": "string",
  "order_priority": 0,
  "icon": "string",
  "created_at": "2025-06-08T21:36:50.958Z",
  "updated_at": "2025-06-08T21:36:50.958Z"
}
```

Рисунок Г.2 – Получить категорию по ID

POST

/api/v1/categories Create Category

Создать новую категорию

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{
  "name": "string",
  "slug": "string",
  "description": "string",
  "order_priority": 0,
  "icon": "string"
}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/api/v1/categories' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "string",
    "slug": "string",
    "description": "string",
    "order_priority": 0,
    "icon": "string"
  }'
```

Request URL

http://localhost:8080/api/v1/categories

Server response

Code

Details

200

Response body

```
{
  "id": "f78f1e6c-7c5e-4dc5-86df-eb68575970f",
  "name": "string",
  "slug": "string",
  "description": "string",
  "order_priority": 0,
  "icon": "string"
}
```

Рисунок Г.3 – Создать новую категорию

GET /api/v1/components Get Components

Получить компоненты с фильтрацией

Parameters
Try it out

Name	Description
category_slug <small>(query)</small>	Фильтр по категории category_slug
brand <small>(query)</small>	Фильтр по брендам brand
price_min <small>(query)</small>	Минимальная цена price_min
price_max <small>(query)</small>	Максимальная цена price_max
only_in_stock boolean <small>(query)</small>	Только товары в наличии Default value : false false
form_factor <small>(query)</small>	Фильтр по форм-фактору form_factor
power_max <small>(query)</small>	Максимальное энергопотребление power_max
search <small>(query)</small>	Поиск по названию/модели search
socket <small>(query)</small>	Фильтр по сокету socket
memory_type <small>(query)</small>	Фильтр по типу памяти memory_type
interface <small>(query)</small>	Фильтр по интерфейсу interface
page integer <small>(query)</small> minimum: 1	Номер страницы Default value : 1 1
limit integer <small>(query)</small>	Количество на странице Default value : 20 20

Рисунок Г.4 – Получение компонентов с фильтрацией

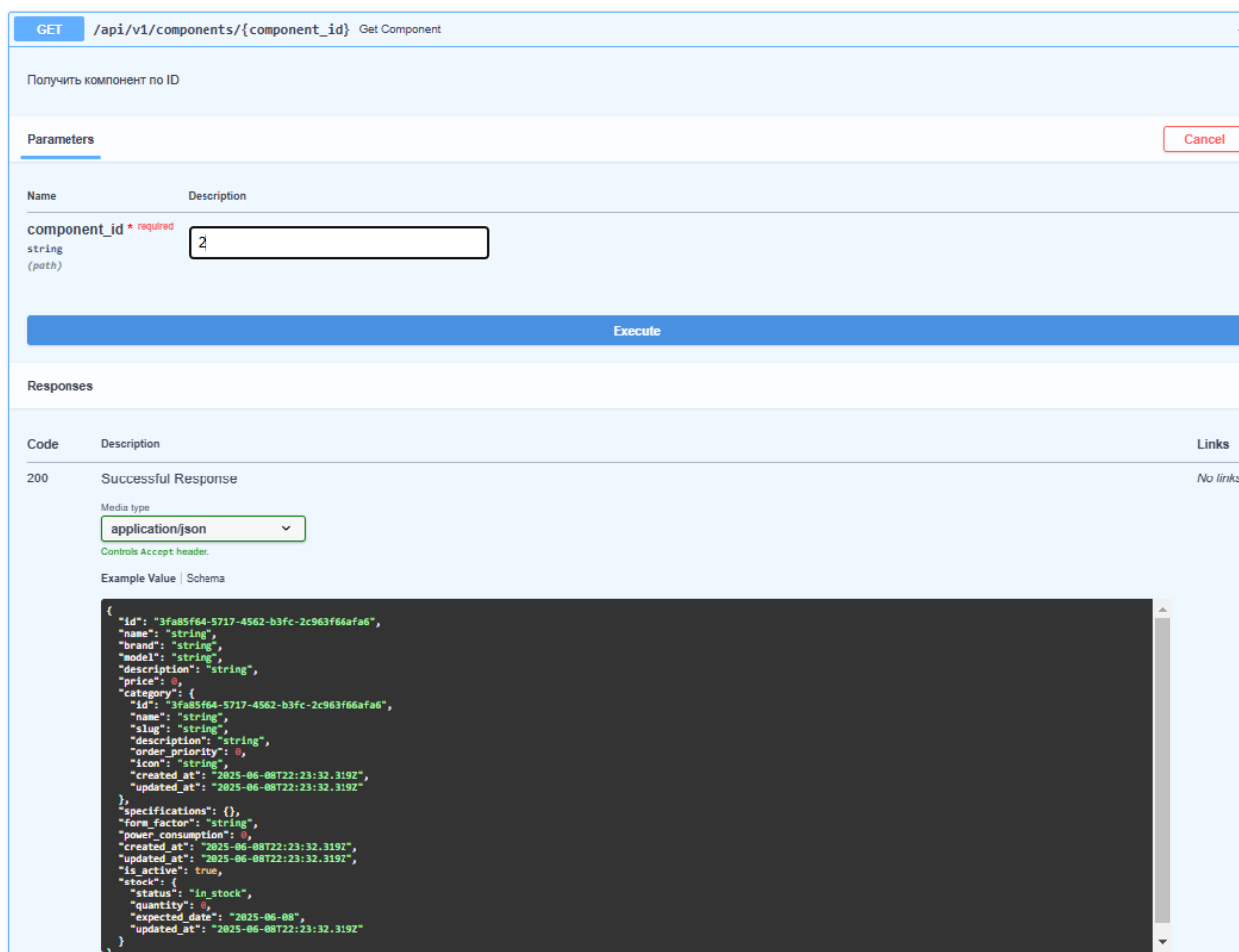


Рисунок Г.5 – Получить компонент по ID

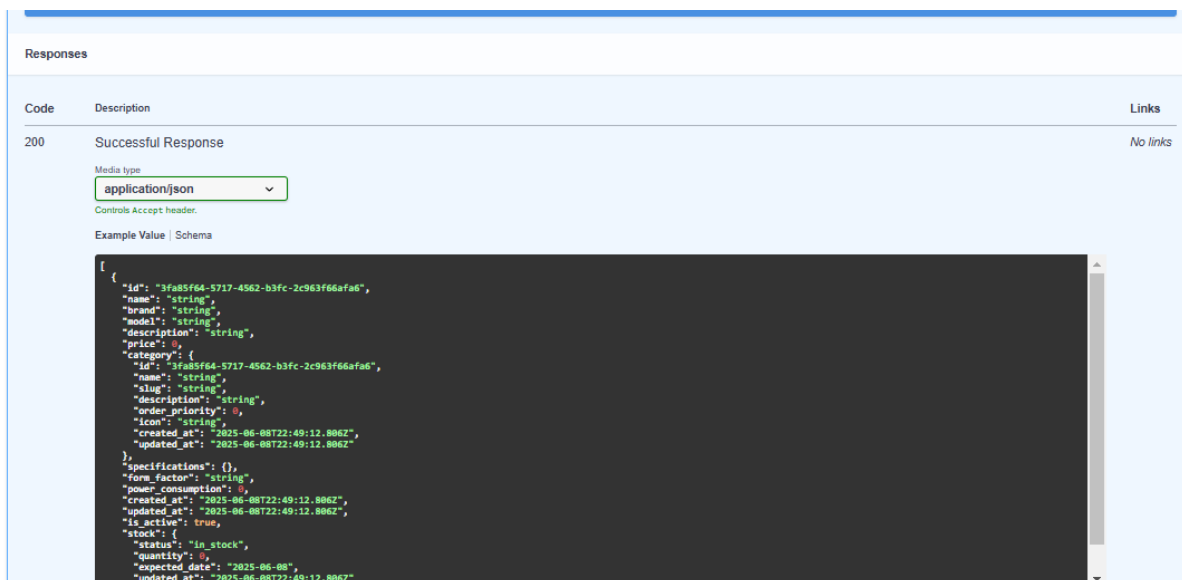


Рисунок Г.6 – Получить компоненты категории с фильтрацией и поиском

boolean
(query)

--

connection

Фильтр по типу подключения

(query)

connection

search

Поиск по названию/модели

(query)

search

page

Номер страницы

integer
(query)
minimum: 1

1

limit

Количество на странице

integer
(query)
maximum: 100
minimum: 1

20

Execute

Responses

Code	Description	Links
200	Successful Response	No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
[
  {
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "name": "string",
    "brand": "string",
    "model": "string",
    "description": "string",
    "price": 0,
    "category": {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "name": "string",
      "slug": "string",
      "description": "string",
      "order_priority": 0,
      "icon": "string",
      "created_at": "2025-06-08T22:47:25.612Z",
      "updated_at": "2025-06-08T22:47:25.612Z"
    },
    "specifications": {},
    "form_factor": "string",
    "power_consumption": 0,
    "created_at": "2025-06-08T22:47:25.612Z",
    "updated_at": "2025-06-08T22:47:25.612Z",
    "is_active": true,
    "stock": {
      "status": "in_stock",
      "quantity": 0,
      "expected_date": "2025-06-08",
      "updated_at": "2025-06-08T22:47:25.612Z"
    }
  }
]
```

Рисунок Г.7 – Получить аксессуары с фильтрацией

POST /api/v1/components/check-compatibility Check Components Compatibility

Проверить совместимость набора компонентов

Parameters Try it out

No parameters

Request body *required* application/json

Example Value | Schema

```
[
  "string"
]
```

Responses

Code	Description	Links
200	Successful Response	No links

Media type application/json

Controls Accept header.

Example Value | Schema

```
{
  "is_compatible": true,
  "status": "compatible",
  "issues": [
    {
      "type": "string",
      "severity": "string",
      "message": "string",
      "component_ids": [
        "3fa85f64-5717-4562-b3fc-2c963f66afa6"
      ],
      "suggestions": [
        "string"
      ]
    }
  ],
  "total_power_consumption": 0,
  "recommended_psu_wattage": 0
}
```

Рисунок Г.8 – Проверить совместимость набора компонентов

GET /api/v1/components/debug/ids Get All Component Ids

Получить все ID компонентов для отладки

Parameters Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  http://localhost:8000/api/v1/components/debug/ids' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/api/v1/components/debug/ids
```

Server response

Code	Details
200	Response body

```
{
  "total_count": 90,
  "active_count": 90,
  "components": [
    {
      "id": "026ffd10-daa8-4588-84ea-286abed253ae",
      "name": "Intel Core i5-13600K",
      "brand": "Intel",
      "is_active": true
    },
    {
      "id": "96147e78-0fa8-487e-a364-63a1311b92bf",
      "name": "AMD Ryzen 5 7600X",
      "brand": "AMD",
      "is_active": true
    },
    {
      "id": "84adec59-753f-4c41-b1b0-ac54cb577da0",
      "name": "Intel Core i7-13700K",
      "brand": "Intel",
      "is_active": true
    },
    {
      "id": "f40cfd05-99e3-4264-aa1f-5323f6594829",
      "name": "AMD Ryzen 7 7700X",
      "brand": "AMD",
      "is_active": true
    },
    {
      "id": "5b7c4718-abb4-456a-81e3-6300a00a7c5a"
    }
  ]
}
```

Download

Рисунок Г.9 – Получить все ID компонентов для отладки

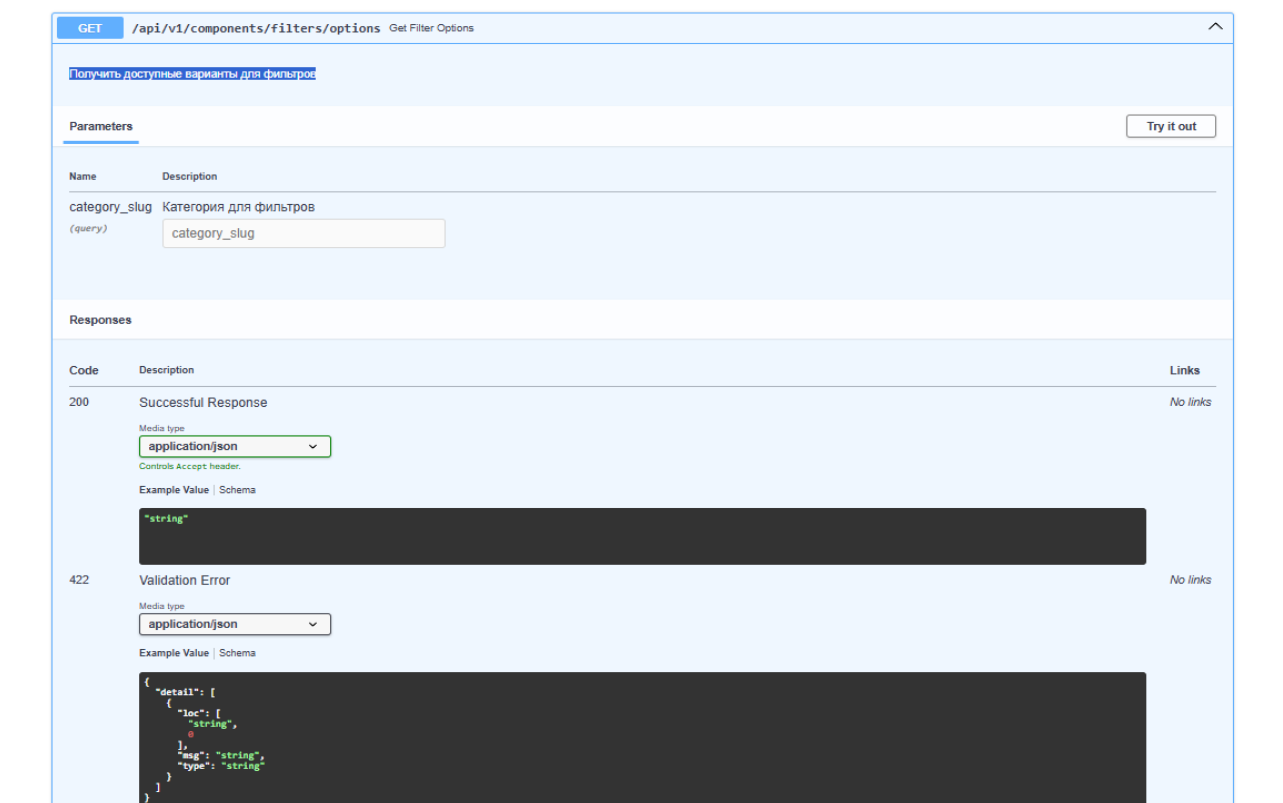


Рисунок Г.10 – Получить доступные варианты для фильтров

Конфигурации

POST /api/v1/configurations Create Configuration

Создать новую конфигурацию

Parameters
Cancel

No parameters

Request body required
application/json

```
{
  "name": "string",
  "description": "string"
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8000/api/v1/configurations' \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
    "name": "string",
    "description": "string"
  }'
```

Request URL

```
http://localhost:8000/api/v1/configurations
```

Server response

Code Details

200
Response body

```
{
  "id": "9165ef80-b942-4341-911f-b100e34fd4b9",
  "name": "string",
  "description": "string",
  "public_uuid": "a09158f3-8dc8-4d5e-926b-bac1227bdf52",
  "total_price": 0
}
```

Рисунок Г.11 – Создать новую конфигурацию

GET /api/v1/configurations Get Configurations

Получить список конфигураций

Parameters
Cancel

Name	Description
Skip integer (query)	<input type="text" value="0"/>
limit integer (query)	<input type="text" value="100"/>

Execute
Clear

Responses

Curl

```
curl -X 'GET' \
  "http://localhost:8000/api/v1/configurations?skip=0&limit=100" \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/api/v1/configurations?skip=0&limit=100
```

Server response

Code	Details
200	<div> Response body <pre>{ "id": "916sef09-b9d2-4341-911f-b100e34fd4b9", "name": "string", "description": "string", "public_uid": "4b915bf3-0dc8-4d5e-926b-bac1227bdf32", "total_price": 0, "total_power_consumption": null, "compatibility_status": "unknown", "compatibility_notes": null, "status": "draft", "is_public": false, "created_at": "2025-06-08T22:35:00.450595Z", "updated_at": "2025-06-08T22:35:00.450595Z", "items": [], "accessories": [] }</pre> Download </div> <div> Response headers <pre>content-length: 397 content-type: application/json date: Sun, 08 Jun 2025 22:35:39 GMT server: uvicorn</pre> </div>

Responses

Code	Description	Links
------	-------------	-------

Рисунок Г.12 – Получить список конфигураций

GET /api/v1/configurations/{config_id} Get Configuration

Получить конфигурацию по ID

Parameters Try it out

Name	Description
config_id required string(\$uuid) (path)	config_id

Responses

Code	Description	Links
200	Successful Response	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afaf",
  "name": "string",
  "description": "string",
  "public_uuid": "string",
  "total_price": 0,
  "total_power_consumption": 0,
  "compatibility_states": "compatible",
  "compatibility_notes": "string",
  "status": "draft",
  "is_public": true,
  "created_at": "2025-06-08T22:37:55.210Z",
  "updated_at": "2025-06-08T22:37:55.210Z",
  "items": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afaf",
      "component": {
        "id": "3fa85f64-5717-4562-b3fc-2c963f66afaf",
        "name": "string",
        "brand": "string",
        "model": "string",
        "description": "string",
        "price": 0,
        "category": {
          "id": "3fa85f64-5717-4562-b3fc-2c963f66afaf",
          "name": "string",
          "slug": "string",
          "description": "string",
          "order_priority": 0,
          "is_public": true
        }
      }
    }
  ]
}
```

422 Validation Error No links

Рисунок Г.13 – Получить конфигурацию по ID

PUT

/api/v1/configurations/{config_id} Update Configuration

Обновить конфигурацию

Parameters

Try it out

Name

Description

config_id * required

string(\$uuid)

(path)

config_id

Request body required

application/json

Example Value | Schema

```
{
  "name": "string",
  "description": "string"
}
```

Responses

Code

Description

Links

200

Successful Response

No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "name": "string",
  "description": "string",
  "public_guid": "string",
  "total_price": 0,
  "total_power_consumption": 0,
  "compatibility_status": "compatible",
  "compatibility_notes": "string",
  "status": "draft",
  "is_public": true,
  "created_at": "2025-06-08T22:38:46.281Z",
  "updated_at": "2025-06-08T22:38:46.281Z",
  "items": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "component": {
        "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
        "name": "string",
        "brand": "string",
        "model": "string",
        "description": "string",
        "price": 0,
        "category": {
          "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
          "name": "string",
          "slug": "string",
          "description": "string",
          "order_priority": 0,

```

Рисунок Г.14 – Обновить конфигурацию

DELETE

/api/v1/configurations/{config_id} Delete Configuration

Удалить конфигурацию

Parameters

Try it out

Name

Description

config_id * required

string(\$uuid)

(path)

config_id

Responses

Code

Description

Links

200

Successful Response

No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

Рисунок Г.15 – Удалить конфигурацию

GET /api/v1/configurations/uuid/{public_uuid} Get Configuration By Uuid

Получить конфигурацию по публичному UUID

Parameters Try it out

Name	Description
public_uuid * required string (path)	12

Responses

Code	Description	Links
200	Successful Response	No links

Media type
application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "name": "string",
  "description": "string",
  "public_uuid": "string",
  "total_price": 0,
  "total_power_consumption": 0,
  "compatibility_status": "compatible",
  "compatibility_notes": "string",
  "status": "draft",
  "is_public": true,
  "created_at": "2025-06-08T22:41:55.110Z",
  "updated_at": "2025-06-08T22:41:55.110Z",
  "items": [
    {
      "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "component": {
        "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
        "name": "string",
        "brand": "string",
        "model": "string",
        "description": "string",
        "price": 0,
        "category": {
          "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
          "name": "string",
          "slug": "string",
          "description": "string",
          "order_priority": 0,
          "status": "string"
        }
      }
    }
  ]
}
```

Рисунок Г.16 – Получить конфигурацию по публичному UUID

POST /api/v1/configurations/{config_id}/items Add Component To Configuration

Добавить компонент в конфигурацию

Parameters Try it out

Name	Description
config_id * required string(\$uuid) (path)	1

Request body * required application/json

Example Value | Schema

```
{
  "component_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "quantity": 1,
  "notes": "string"
}
```

Responses

Code	Description	Links
200	Successful Response	No links

Media type
application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

Рисунок Г.17 – Добавить компонент в конфигурацию

POST

/api/v1/configurations/{config_id}/accessories

Add Accessory To Configuration

^

Добавить аксессуар в конфигурацию

Parameters

Try it out

Name	Description
config_id * required	
string(\$uuid)	
(path)	2

Request body * required

application/json

Example Value | Schema

```
{  "component_id": "3fa85f64-5717-4562-b3fc-2c963f66af6",  "quantity": 1,  "notes": "string"}
```

Responses

Code	Description	Links
200	Successful Response	No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

Рисунок Г.18 – Добавить аксессуар в конфигурацию

Приложение Д
(рекомендуемое)

Результаты тестирования клиентской части приложения

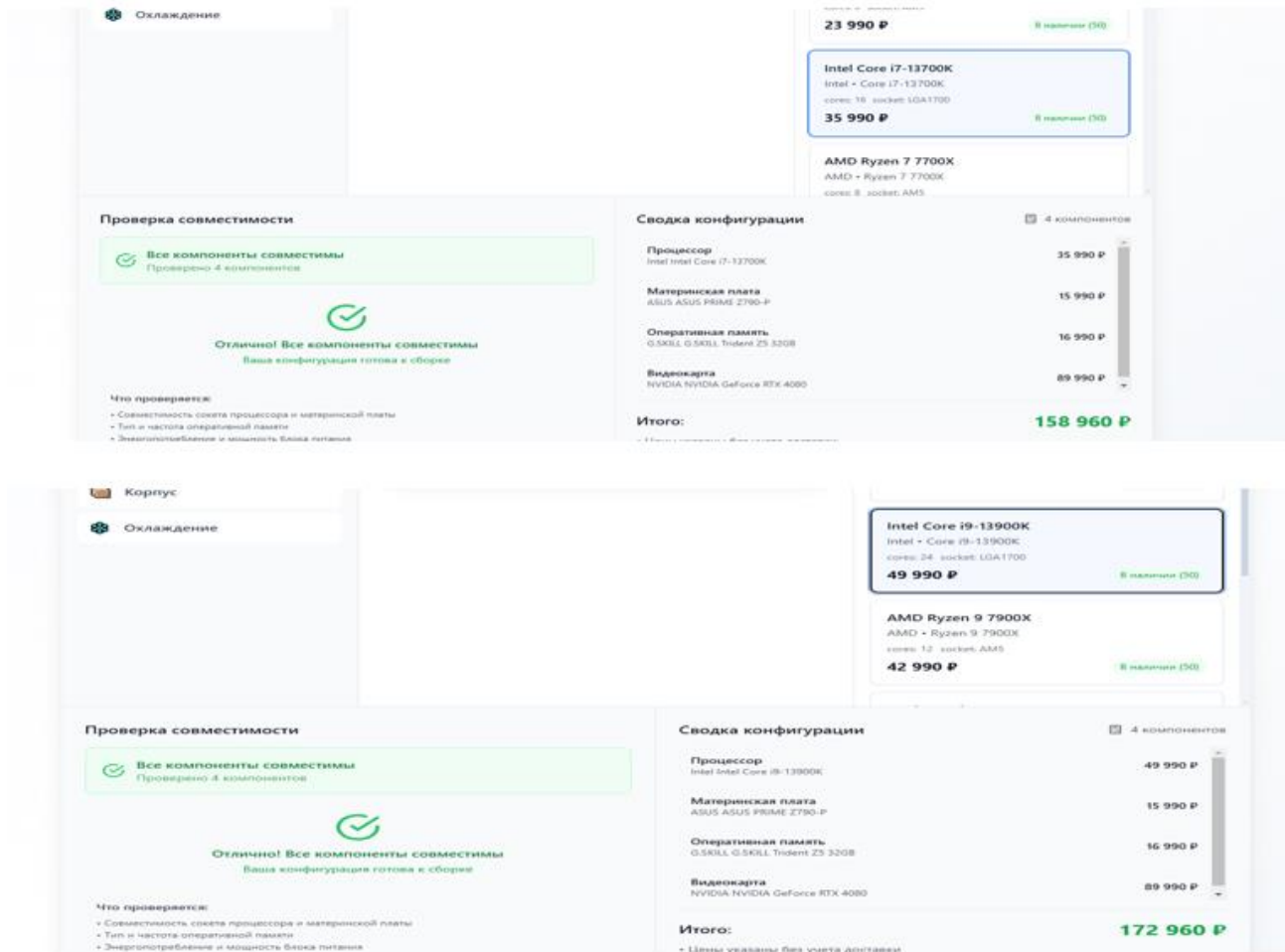


Рисунок Д.1 – Редактирование конфигурации

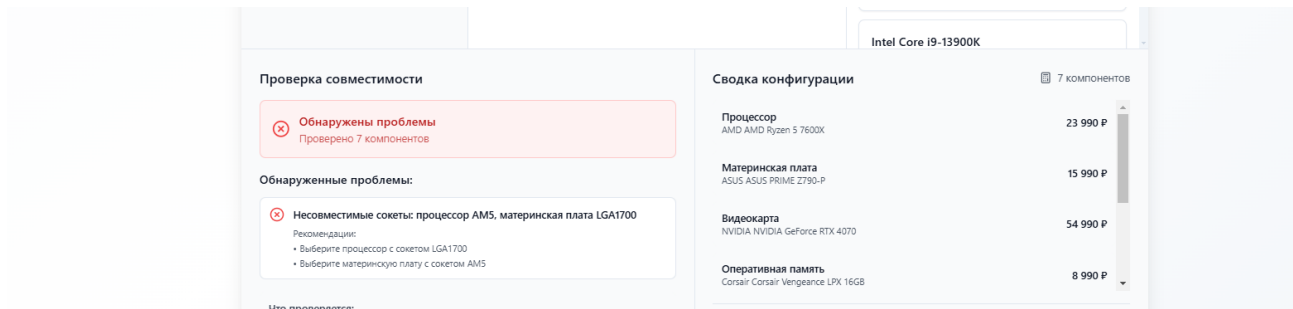


Рисунок Д.2 – Проверка совместимости компонентов

Создайте свой **идеальный** компьютер

Профессиональный конфигуратор ПК с проверкой совместимости компонентов, отслеживанием наличия и современным интерфейсом

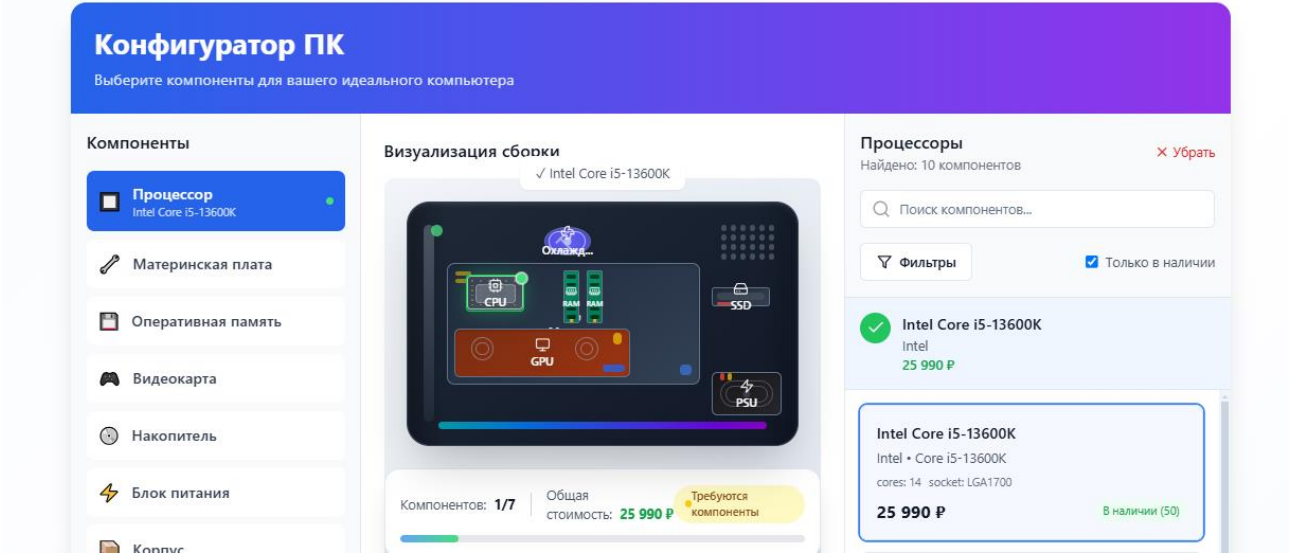


Рисунок Д.3 – Выбор процессора

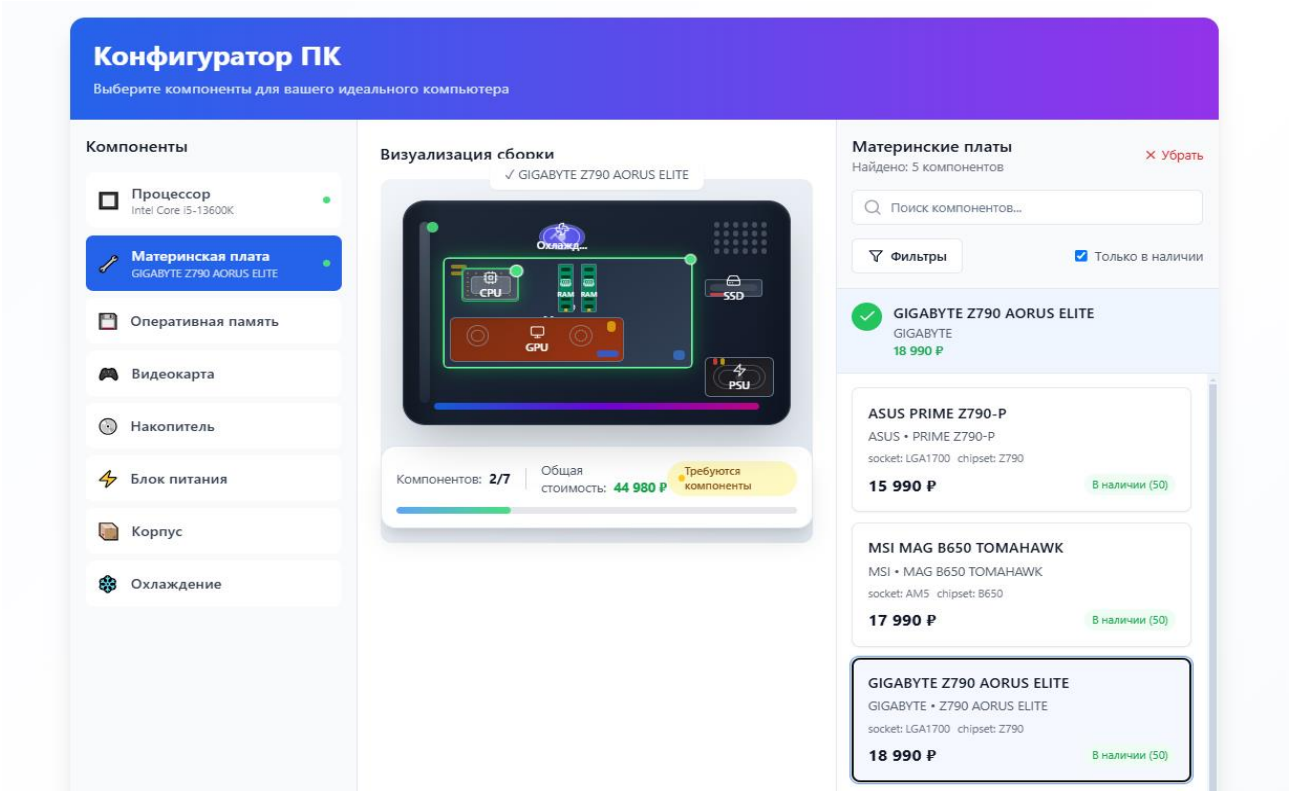


Рисунок Д.4 – Выбор материнской платы

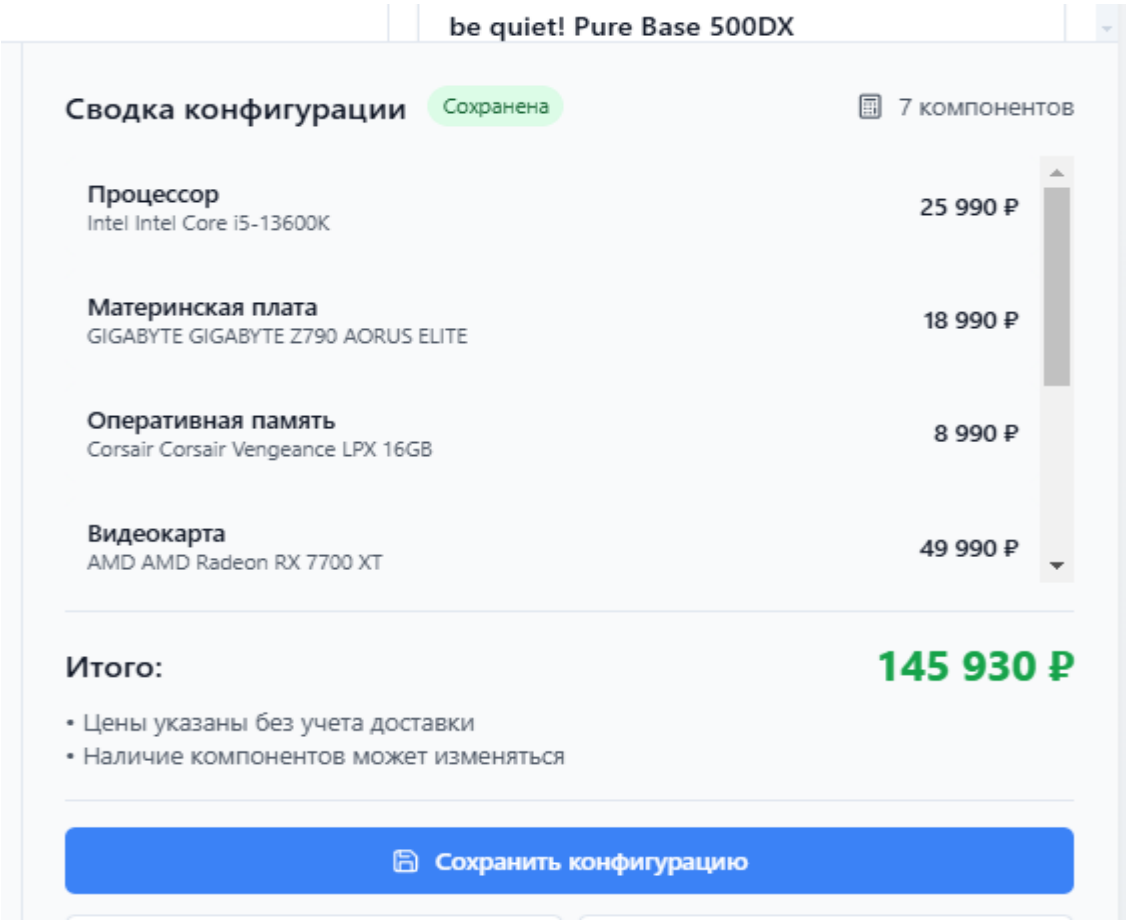


Рисунок Д.5 – Сохранение конфигурации

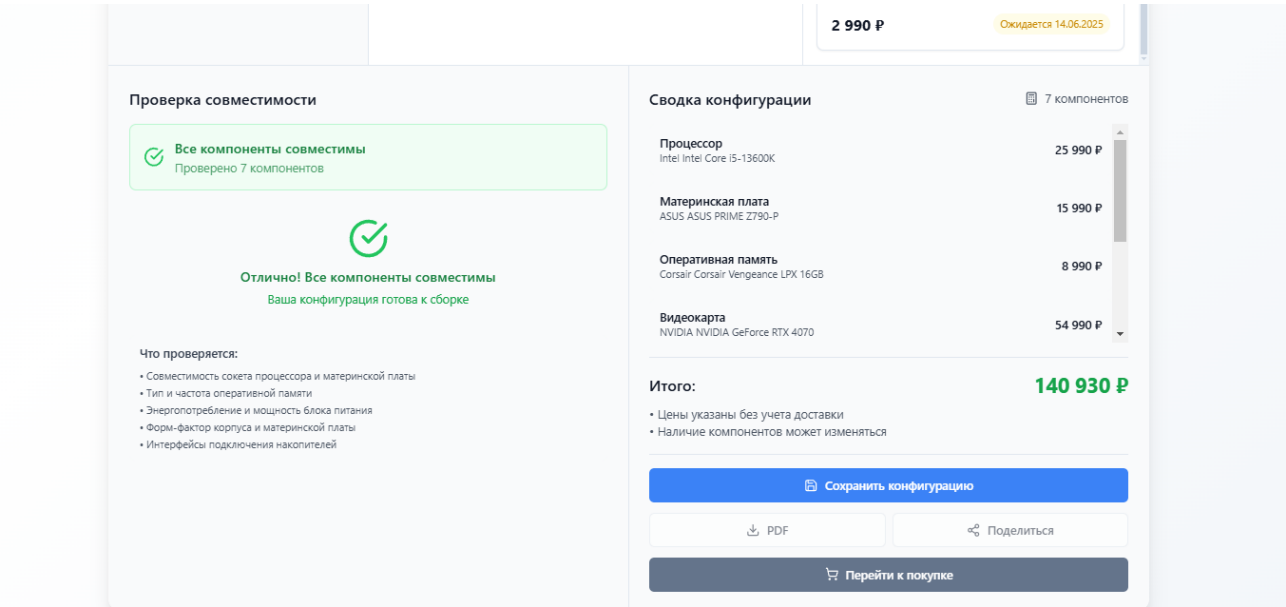


Рисунок Д.6 – Выбор комплектующих

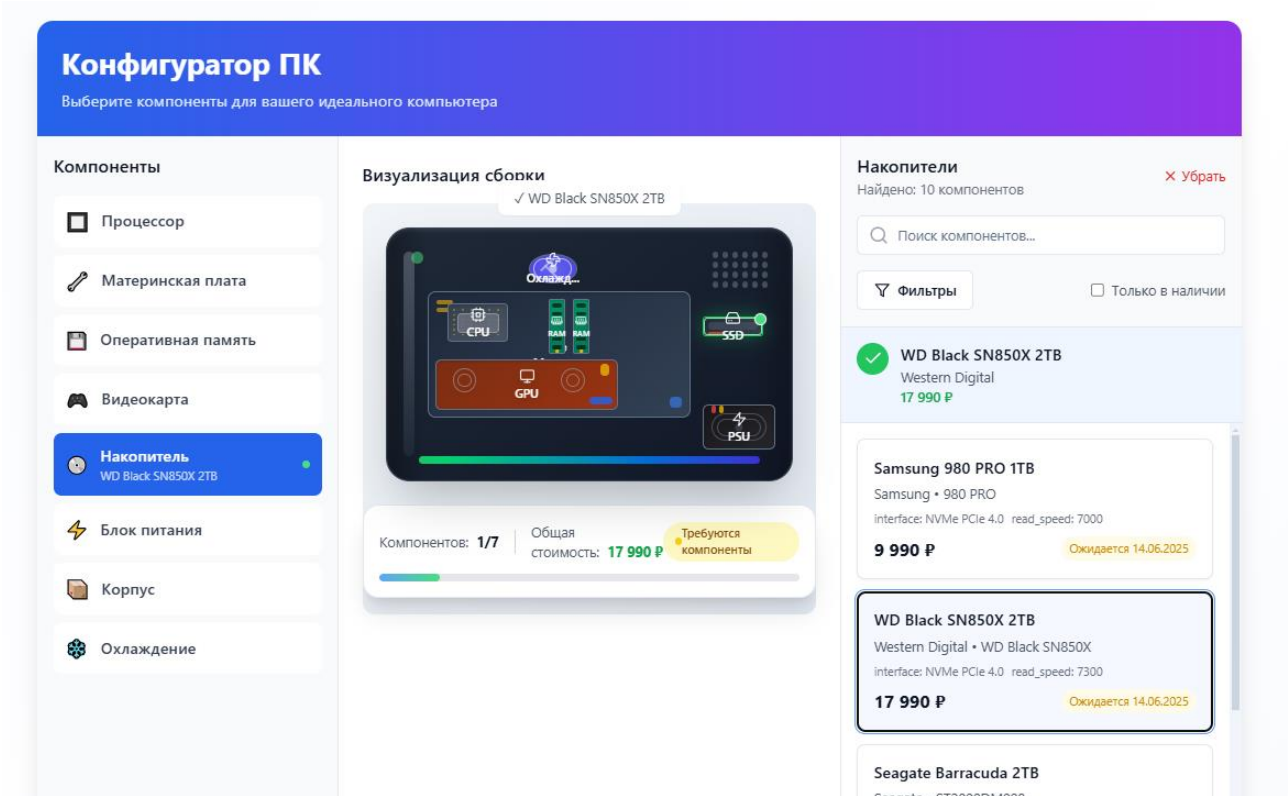


Рисунок Д.7 – Выбор накопителя

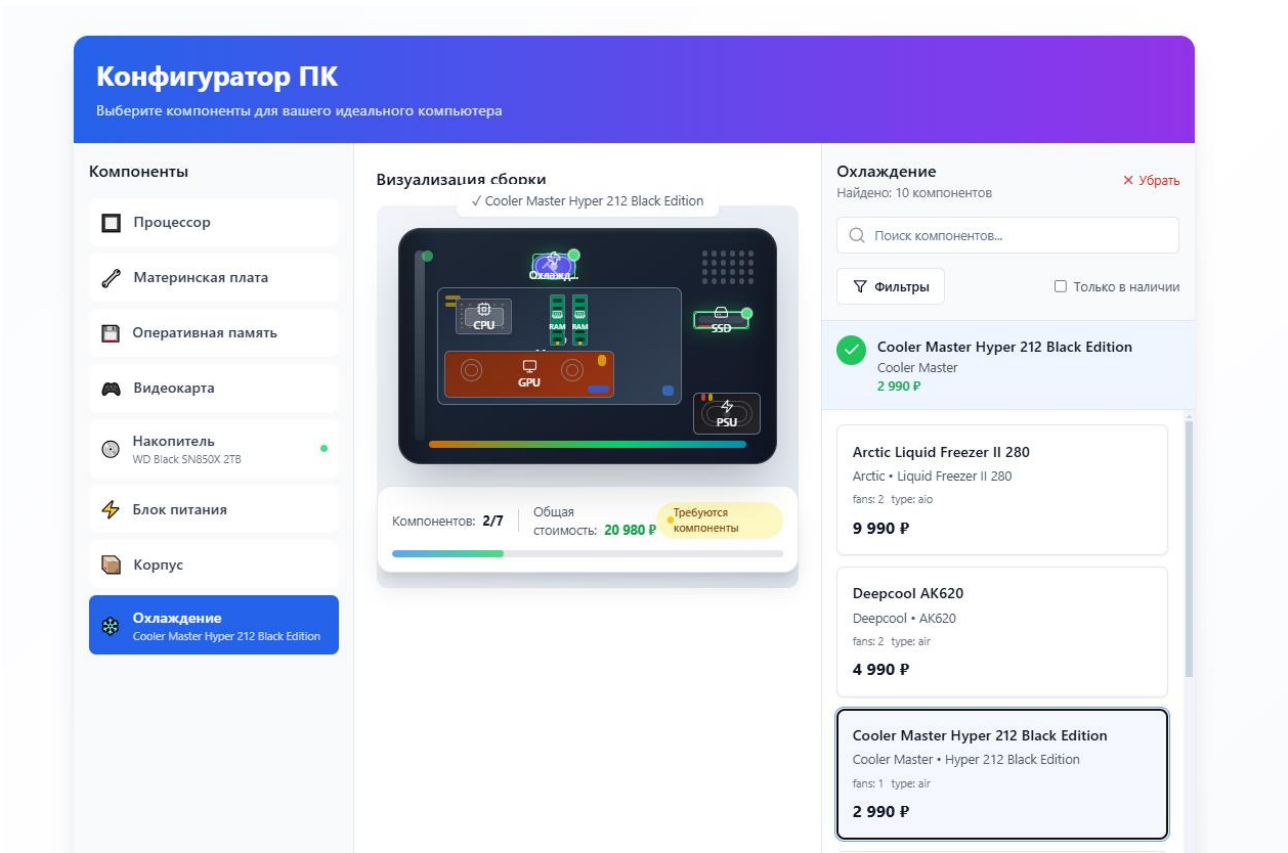


Рисунок Д.8 – Выбор кулера

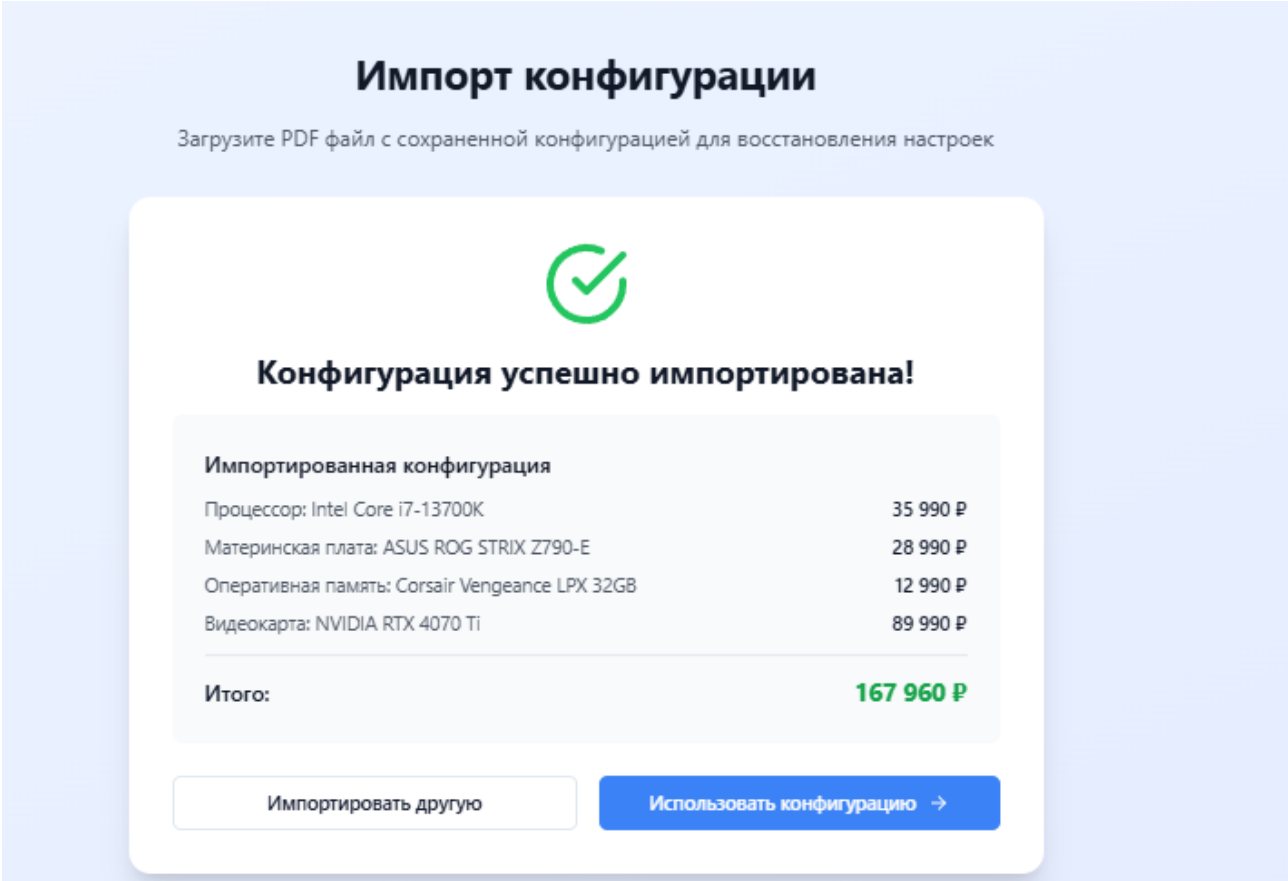


Рисунок Д.9 – Импорт из конфигурации из PDF

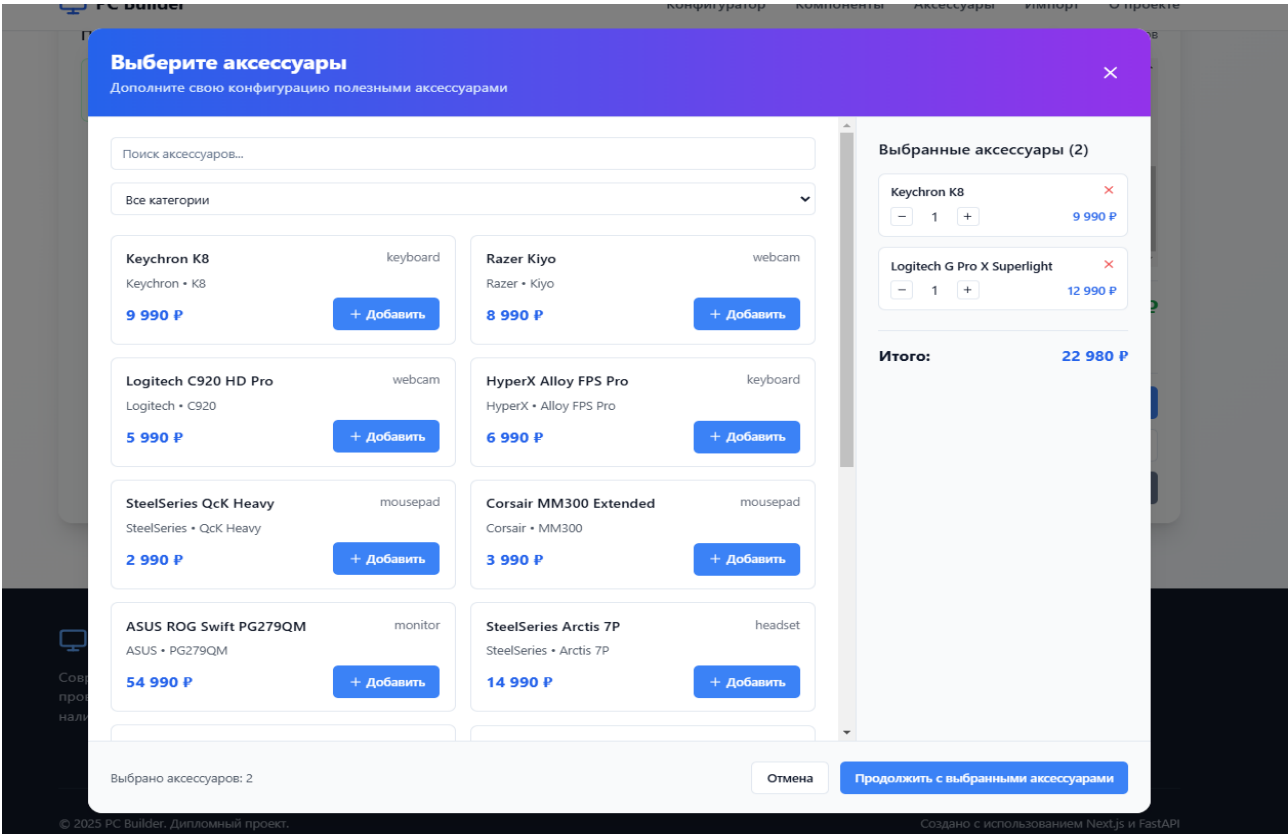


Рисунок Д.10 – Добавление аксессуаров

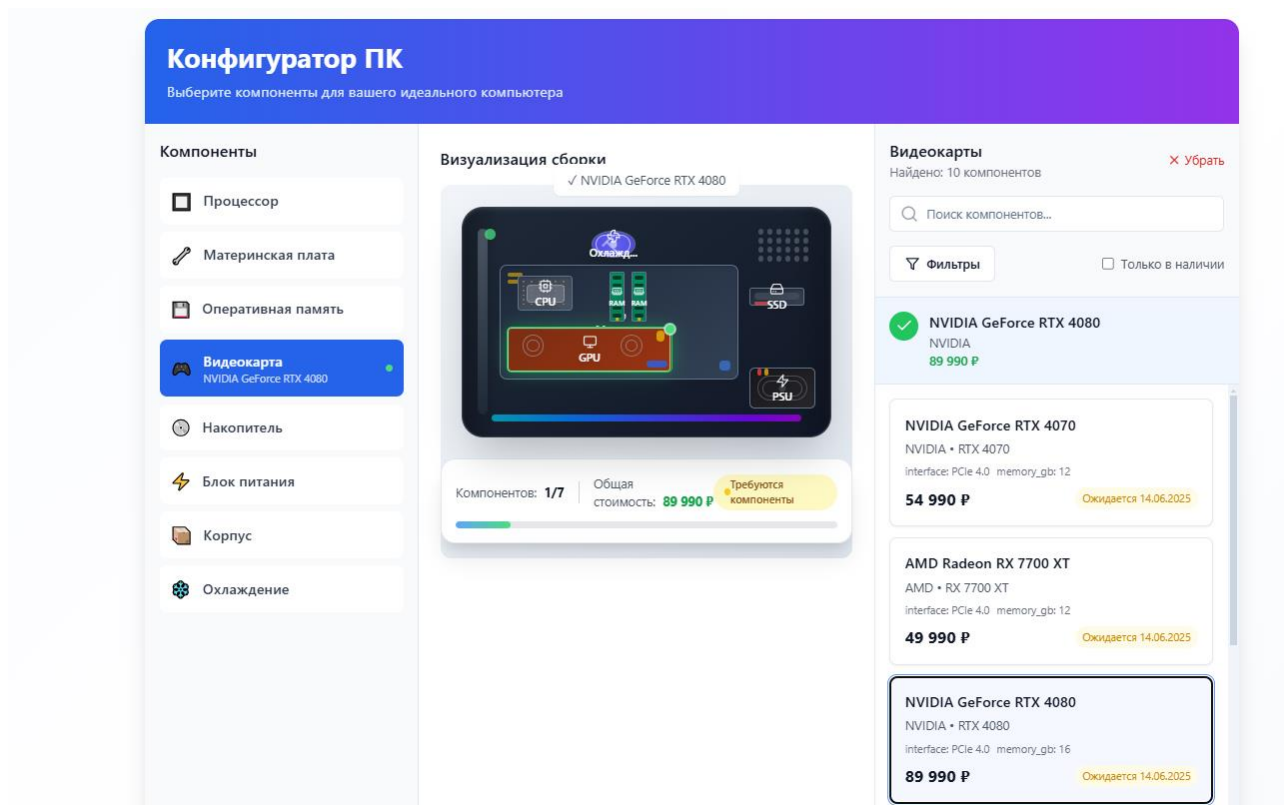


Рисунок Д.11 – Выбор видеокарты

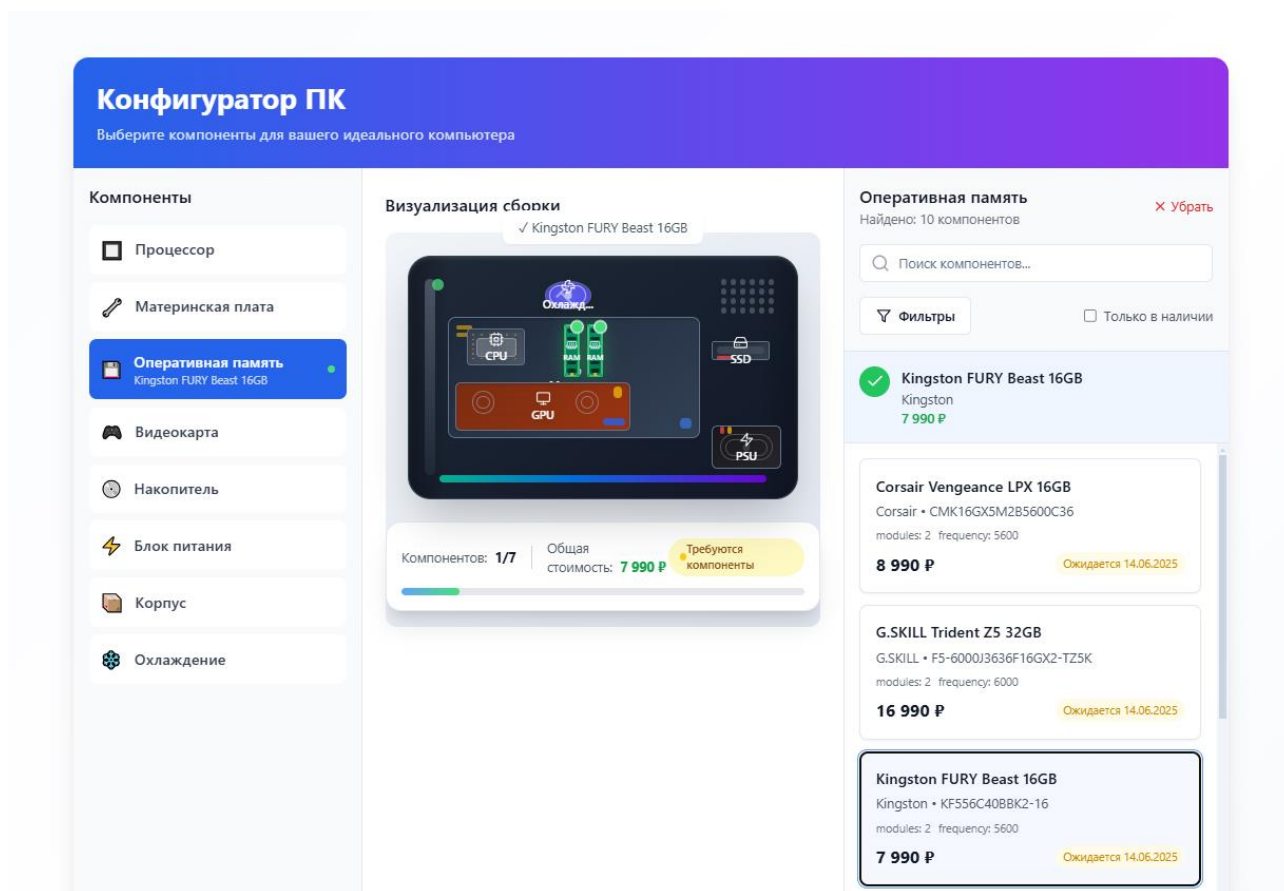


Рисунок Д.12 – Выбор оперативной памяти

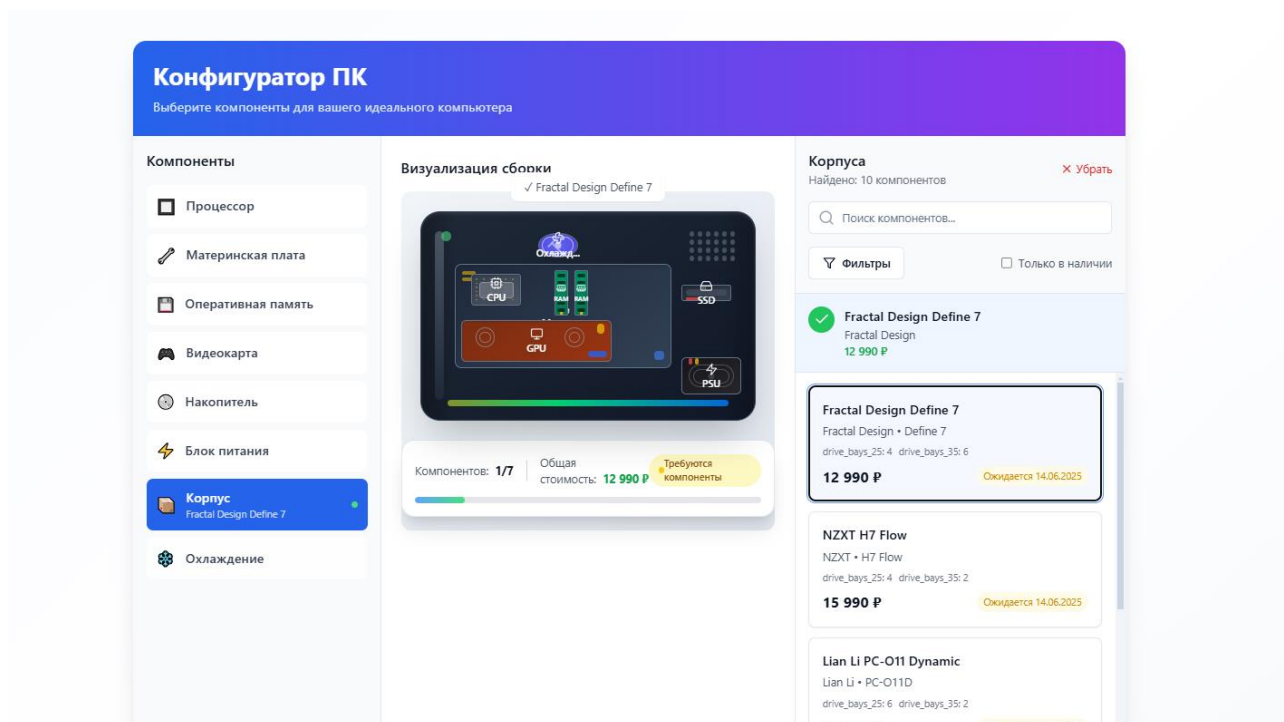


Рисунок Д.13 – Выбор корпуса

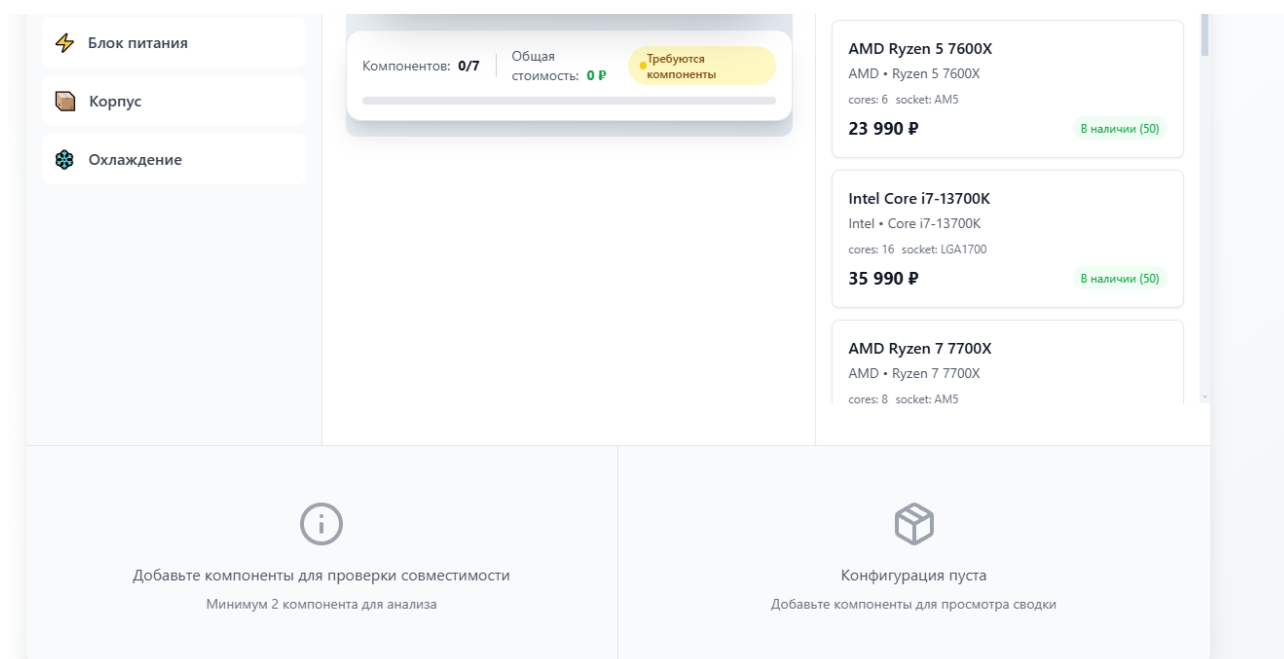


Рисунок Д.14 – Удаление сохранённой конфигурации