Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт управления бизнес-процессами

Кафедра бизнес-информатики и моделирования бизнес-процессов

Курсовая работа

по дисциплине «Проектирование, разработка и оптимизация WEB-приложений»

Создание web-приложения с применением современных web-технологий по  
тематике «Электроника»

Преподаватель: Никитин А. И.

подпись дата

Студент: УБ23-08Б, 432326828 \_\_\_\_\_\_\_\_\_\_ Бирюков А. А.

подпись дата

Красноярск 2024

**Содержание**

[Введение 2](#_Toc185190636)

[Цель и задачи 4](#_Toc185190637)

[Теоретическая часть 6](#_Toc185190638)

[1. Html (HyperText Markup Language) 7](#_Toc185190639)

[2. CSS (Cascading Style Sheets) 9](#_Toc185190640)

[3. JavaScript 11](#_Toc185190641)

[4. PHP (Hypertext Preprocessor) 13](#_Toc185190642)

[5. MySQL 15](#_Toc185190643)

[6. Brackets 18](#_Toc185190644)

[7. XAMPP 19](#_Toc185190645)

[Практическая часть 19](#_Toc185190646)

[1. Страница регистрации аккаунта 19](#_Toc185190647)

[2. Страница входа 24](#_Toc185190648)

[3. Страница “Панель управления аккаунтом” 29](#_Toc185190649)

[4. Страница “Админ панель” 33](#_Toc185190650)

[5. Страница “Редактор аккаунта пользователя” 38](#_Toc185190651)

[6. Главная страница сайта 43](#_Toc185190652)

[7. Форма для оформления заказа 50](#_Toc185190653)

[8. Страница “Наши партнёры” 55](#_Toc185190654)

[9. Страница с готовыми сборками ПК 59](#_Toc185190655)

[10. База данных для сайта 65](#_Toc185190656)

[Заключение 70](#_Toc185190657)

[Список использованных источников 72](#_Toc185190658)

# **Введение**

В условиях стремительного развития информационных технологий и цифровизации различных сфер деятельности, веб-приложения стали неотъемлемой частью повседневной жизни. Они находят применение в самых разных областях, начиная от электронной коммерции и заканчивая образованием и развлечениями. Особую популярность приобретают веб-сервисы, которые предоставляют пользователям возможность выполнять сложные задачи, такие как подбор комплектующих, настраивание устройств и проектирование решений, не выходя из дома. Тематика электроники, в частности, имеет высокий потенциал для использования таких приложений.

Целью данной курсовой работы является разработка веб-приложения, предназначенного для конфигурирования компьютеров. Такое приложение позволяет пользователям самостоятельно подбирать комплектующие для персонального компьютера, учитывая их совместимость, стоимость и производительность. В рамках проекта была реализована функциональность, обеспечивающая удобный и интуитивно понятный интерфейс, а также надежное взаимодействие с базой данных.

Разработка приложения велась с использованием современных веб-технологий, включая HTML для создания структуры страниц, CSS для оформления интерфейса, JavaScript для реализации интерактивности и проверки данных на клиентской стороне. Серверная часть была написана на языке PHP, а для хранения данных использовалась реляционная база данных MySQL. Такой технологический стек позволяет обеспечить не только стабильную работу приложения, но и его масштабируемость, что особенно важно в условиях растущего числа пользователей и увеличения объема данных.

Работа над данным проектом включала в себя несколько ключевых этапов: анализ требований пользователей, проектирование архитектуры веб-приложения, реализация и тестирование программного обеспечения, а также создание базы данных, которая обеспечивает хранение и управление сведениями о комплектующих. Кроме того, особое внимание уделялось вопросам удобства и безопасности, что является важным фактором для успешной работы любого веб-приложения.

Таким образом, представленная курсовая работа охватывает как технические, так и практические аспекты разработки веб-приложений. Она ориентирована на решение реальной задачи в области электроники, предоставляя пользователям простой и удобный инструмент для выбора и конфигурирования компьютеров.

# **Цель и задачи**

Целью данной курсовой работы является разработка веб-приложения для конфигурирования персональных компьютеров, которое позволит пользователям подбирать комплектующие с учетом их характеристик, совместимости и стоимости. Приложение должно быть функциональным, удобным в использовании и отвечать современным требованиям в области веб-разработки.

Для достижения этой цели поставлены следующие задачи:

1. Анализ предметной области и требований пользователей:

A) Изучить основные подходы к созданию конфигураторов компьютеров.

Б) Выявить ключевые требования к функционалу и пользовательскому интерфейсу.

В) Определить структуру базы данных для хранения информации о комплектующих.

2. Проектирование архитектуры приложения:

A) Разработать архитектурное решение, включающее клиентскую и серверную части.

Б) Спроектировать пользовательский интерфейс с учетом принципов удобства и интуитивности.

3. Реализация клиентской части веб-приложения:

A) Создать структуру страниц с использованием HTML.

Б) Оформить внешний вид с помощью CSS, обеспечив современный и адаптивный дизайн.

В) Реализовать интерактивность с помощью JavaScript (например, фильтрацию комплектующих, проверку совместимости).

4. Реализация серверной части и взаимодействие с базой данных:

A) Настроить серверную логику с использованием PHP.

Б) Разработать и настроить базу данных на основе MySQL для хранения сведений о комплектующих и пользовательских конфигурациях.

В) Обеспечить взаимодействие между клиентом и сервером, используя запросы (AJAX).

5. Тестирование и отладка приложения:

A) Провести тестирование на корректность работы функционала, в том числе проверку совместимости комплектующих.

Б) Устранить ошибки и оптимизировать работу приложения.

6. Обеспечение безопасности приложения:

A) Реализовать защиту от SQL-инъекций и других распространенных уязвимостей.

Б) Организовать корректное хранение данных, соблюдая стандарты безопасности.

7. Подготовка документации и демонстрация результатов:

A) Описать структуру и функционал приложения, а также процесс его разработки.

Б) Подготовить презентацию или демонстрационный материал для защиты работы.

Реализация указанных задач позволит создать удобный и современный инструмент для подбора и конфигурирования компьютеров, востребованный как среди начинающих пользователей, так и среди профессионалов.

# **Теоретическая часть**

Современная разработка веб-приложений основана на использовании разнообразных языков программирования и технологий, которые обеспечивают создание интерактивных, удобных и производительных решений. Каждый компонент веб-приложения, от интерфейса до серверной логики, выполняет важную роль в обеспечении функциональности и удобства использования.

В рамках разработки конфигуратора компьютеров для реализации всех аспектов проекта были использованы следующие технологии:

HTML (HyperText Markup Language) — язык разметки, отвечающий за создание структуры веб-страниц. HTML предоставляет основу, на которой строятся элементы интерфейса, такие как кнопки, таблицы и формы.

CSS (Cascading Style Sheets) — язык описания стилей, применяемый для визуального оформления веб-страниц. CSS используется для создания привлекательного дизайна и обеспечения адаптивности интерфейса под различные устройства.

JavaScript — язык программирования, отвечающий за интерактивность веб-приложения. С его помощью реализуются функции проверки данных, динамическое обновление интерфейса и взаимодействие с сервером через асинхронные запросы (AJAX).

PHP (Hypertext Preprocessor) — серверный язык программирования, обеспечивающий обработку запросов пользователей, управление данными и взаимодействие с базой данных. PHP служит связующим звеном между клиентской частью приложения и сервером.

MySQL — реляционная система управления базами данных, используемая для хранения и управления данными о комплектующих и пользовательских конфигурациях.

Каждая из этих технологий играет ключевую роль в построении веб-приложений. теоретической части подробно рассматриваются принципы работы и основные особенности указанных языков программирования и технологий. Это позволит более глубоко понять, как именно они применяются при разработке веб-приложения и какие задачи они помогают решить.

## **1. Html (HyperText Markup Language)**

HTML (HyperText Markup Language) — это язык разметки гипертекста, который является фундаментом веб-разработки. Он используется для создания структуры веб-страниц и представления их содержимого. HTML определяет элементы страницы, такие как заголовки, параграфы, изображения, ссылки, таблицы и формы, которые формируют основу любого веб-приложения.

HTML был разработан Тимом Бернерсом-Ли в 1991 году как язык для структурирования информации в сети. Первая версия языка поддерживала лишь базовые элементы, но с развитием технологий HTML значительно эволюционировал. Современная версия, HTML5, была представлена в 2014 году и стала значительным шагом вперед, добавив поддержку мультимедиа, улучшив семантическую структуру и сделав возможным создание интерактивных веб-приложений.

Базовая структура HTML-документа включает в себя обязательные элементы, которые обеспечивают правильное отображение страницы браузером. Например, объявление типа документа <!DOCTYPE html> указывает на использование HTML5. Корневой элемент <html> содержит два основных раздела: <head> — для метаинформации о странице, и <body> — для основного содержимого. Пример такой структуры включает заголовки, параграфы и прочие элементы, формирующие интерфейс пользователя.

HTML поддерживает множество элементов для создания различного содержимого. Заголовки <h1>–<h6> применяются для выделения текста разного уровня важности. Абзацы создаются с помощью элемента <p>, а ссылки, позволяющие навигацию между страницами, определяются с использованием <a>. Для отображения изображений используется элемент <img>, который позволяет добавлять графику с указанием источника и описания. Таблицы <table> применяются для представления данных в табличной форме, а формы <form> предоставляют возможность ввода данных пользователем.

С развитием HTML5 были добавлены семантические элементы, такие как <header>, <nav>, <section>, <article> и <footer>. Эти элементы помогают улучшить читаемость кода, структурировать страницы и повысить их доступность для поисковых систем. Например, элемент <nav> используется для создания навигационного меню, а <footer> определяет подвал страницы.

HTML обладает рядом преимуществ. Он прост в изучении, что делает его идеальным для начинающих разработчиков. Благодаря кроссбраузерной совместимости HTML поддерживается всеми современными веб-браузерами. Семантические элементы улучшают доступность и оптимизацию для поисковых систем (SEO), а гибкость языка обеспечивает его интеграцию с другими технологиями, такими как CSS и JavaScript.

## **2. CSS (Cascading Style Sheets)**

CSS (Cascading Style Sheets) — это язык каскадных таблиц стилей, который используется для оформления веб-страниц и управления их визуальным представлением. С помощью CSS разработчики могут задавать внешний вид элементов HTML, включая их размеры, цвета, шрифты, отступы и другие параметры. CSS позволяет отделить структуру документа, созданную на HTML, от его дизайна, что упрощает разработку и поддержку веб-приложений.

История CSS началась в 1996 году, когда была представлена первая версия этого языка. С тех пор CSS постоянно развивался, а его современные возможности значительно расширились. Версия CSS3, представленная в 2012 году, стала важной вехой в развитии веб-дизайна. Она добавила множество новых функций, таких как градиенты, анимации, трансформации и медиа-запросы, что сделало дизайн более гибким и адаптивным.

CSS работает по принципу каскадов: стили применяются к элементам HTML на основе определенных правил, которые могут наследоваться, перекрываться или комбинироваться. Это позволяет создавать сложные и масштабируемые системы оформления. Основной файл стилей может быть подключен к HTML-документу через элемент <link> в разделе <head>: <link rel="stylesheet" href="styles.css">

Основные возможности CSS включают управление шрифтами и цветами, позиционирование элементов, а также настройку внешнего вида различных интерфейсных компонентов. Например, для изменения цвета текста и размера шрифта используется следующий код:

h1 {

color: blue;

font-size: 24px;

}

Для управления расположением элементов применяются такие свойства, как margin, padding и position. Они позволяют задавать отступы, выравнивать содержимое и позиционировать элементы относительно их родительских контейнеров или окна браузера.

Одной из ключевых функций CSS является создание адаптивного дизайна, который автоматически подстраивается под размеры экрана пользователя. Для этого используются медиа-запросы, например:

@media (max-width: 768px) {

body {

font-size: 14px;

}

}

Этот подход особенно важен для разработки современных веб-приложений, которые должны корректно отображаться как на настольных компьютерах, так и на мобильных устройствах.

Современный CSS предоставляет возможности создания визуальных эффектов без использования дополнительных инструментов. С помощью градиентов можно задавать плавные переходы между цветами, а с использованием анимаций и трансформаций — оживлять интерфейс. Пример анимации:

@keyframes example {

from {background-color: red;}

to {background-color: yellow;}

}

div {

animation: example 2s infinite;

}

CSS также позволяет работать с сетками и макетами, что особенно важно при создании сложных веб-приложений. Например, технология Flexbox помогает выравнивать элементы внутри контейнеров, а CSS Grid обеспечивает возможность создания двухмерных макетов с точным размещением элементов.

CSS является универсальным инструментом для создания визуально привлекательных веб-страниц, который продолжает развиваться и расширять свои возможности. Одним из ключевых преимуществ CSS является его способность отделять логику визуального оформления от структуры HTML-документа, что делает код более модульным и легко поддерживаемым. Благодаря широкому набору инструментов, таких как селекторы, псевдоклассы, псевдоэлементы и возможность подключения сторонних библиотек стилей, разработчики могут гибко управлять внешним видом веб-приложений. В совокупности с современными технологиями, такими как SASS и LESS, которые расширяют возможности CSS, язык остается важнейшим компонентом в разработке адаптивных и интерактивных интерфейсов.

## **3. JavaScript**

JavaScript — это высокоуровневый язык программирования, который используется для создания интерактивных и динамичных элементов на веб-страницах. Он был разработан в 1995 году Бренданом Эйхом и изначально предназначался для выполнения простых скриптов прямо в браузере. С тех пор JavaScript значительно эволюционировал и сегодня является неотъемлемой частью веб-разработки, обеспечивая взаимодействие между пользователем и приложением.

Одной из главных особенностей JavaScript является возможность его выполнения на стороне клиента, что позволяет браузеру интерпретировать код и мгновенно реагировать на действия пользователя. Например, JavaScript может использоваться для проверки заполнения форм, динамического изменения содержимого страницы, создания анимации, работы с мультимедиа и взаимодействия с сервером без необходимости обновления страницы (технология AJAX).

JavaScript позволяет добавлять интерактивные элементы на сайт с помощью обработчиков событий. Например, нажатие на кнопку можно обработать следующим образом:

document.querySelector('button').addEventListener('click', function() {

alert('Кнопка нажата!');

});

Современный JavaScript поддерживает множество инструментов для разработки сложных приложений. Например, DOM (Document Object Model) позволяет взаимодействовать с элементами HTML-документа, изменяя их содержимое, стиль или структуру. С помощью API браузера JavaScript предоставляет доступ к более сложным функциям, таким как геолокация, работа с файловой системой, взаимодействие с камерами и микрофонами.

Благодаря появлению платформы Node.js, JavaScript получил возможность выполнения и на стороне сервера. Это сделало его универсальным языком, который можно использовать как для клиентской, так и для серверной разработки. Кроме того, JavaScript активно используется для работы с базами данных, создания REST API и разработки серверной логики.

Одна из важнейших особенностей JavaScript — поддержка асинхронного программирования. Конструкции, такие как Promise и async/await, позволяют обрабатывать операции, которые выполняются в течение неопределенного времени, например, запросы к серверу:

async function fetchData() {

const response = await fetch('https://api.example.com/data');

const data = await response.json();

console.log(data);

}

fetchData();

JavaScript поддерживает объектно-ориентированный и функциональный подходы, что делает его гибким для решения задач разной сложности. С появлением ES6 (ECMAScript 2015) язык обогатился новыми возможностями, такими как стрелочные функции, модули, классы и деструктуризация, которые сделали код более выразительным и компактным.

Визуальные эффекты также занимают важное место в JavaScript. С его помощью можно создавать анимации, переходы, игры и интерактивные графики. Например, библиотеки вроде D3.js и Three.js позволяют визуализировать данные и создавать трехмерные сцены.

JavaScript является основой многих современных веб-технологий. Он тесно интегрирован с CSS и HTML, а также поддерживается всеми браузерами. Более того, многочисленные фреймворки и библиотеки, такие как React, Vue.js и Angular, позволяют упрощать разработку сложных интерфейсов, обеспечивая при этом высокую производительность и удобство использования.

## **4. PHP (Hypertext Preprocessor)**

PHP (Hypertext Preprocessor) — это серверный язык программирования, предназначенный для создания динамических веб-страниц и приложений. Он был разработан в 1994 году Расмусом Лердорфом и с тех пор стал одним из самых популярных инструментов в веб-разработке. PHP позволяет обрабатывать запросы пользователей, взаимодействовать с базами данных, генерировать HTML-код на стороне сервера и отправлять его клиенту.

Одной из ключевых особенностей PHP является его способность интегрироваться с HTML-кодом. Разработчики могут включать PHP-скрипты прямо в HTML-документ, что делает его простым для изучения и использования. Пример базового PHP-кода:

<?php

echo "Привет, мир!";

?>

PHP ориентирован на серверную разработку и обеспечивает выполнение логики приложения перед тем, как страница будет отправлена браузеру. Это позволяет создавать динамические сайты, где содержимое меняется в зависимости от действий пользователя или данных, хранящихся на сервере. Например, с помощью PHP можно обрабатывать формы, авторизовать пользователей, генерировать отчеты или отправлять электронные письма.

Важной составляющей PHP является его тесная интеграция с базами данных. Наиболее популярная связка — это PHP и MySQL, где PHP используется для выполнения SQL-запросов и работы с данными. Пример взаимодействия:

<?php

$mysqli = new mysqli("localhost", "user", "password", "database");

if ($mysqli->connect\_error) {

die("Ошибка подключения: " . $mysqli->connect\_error);

}

$result = $mysqli->query("SELECT \* FROM products");

while ($row = $result->fetch\_assoc()) {

echo "Название: " . $row["name"] . "<br>";

}

?>

PHP поддерживает объектно-ориентированное программирование, что делает его пригодным для разработки сложных приложений. Использование классов и объектов позволяет создавать масштабируемый и поддерживаемый код. Вот пример простого класса:

<?php

class Product {

public $name;

public $price;

public function \_\_construct($name, $price) {

$this->name = $name;

$this->price = $price;

}

public function display() {

echo "Товар: $this->name, Цена: $this->price";

}

}

$product = new Product("Ноутбук", 50000);

$product->display();

?>

PHP предоставляет широкий набор встроенных функций и библиотек, упрощающих работу с файлами, шифрованием, обработкой данных и сетевыми запросами. Это делает его универсальным инструментом для решения различных задач в веб-разработке.

Еще одной сильной стороной PHP является поддержка множества популярных фреймворков, таких как Laravel, Symfony и CodeIgniter, которые упрощают разработку и ускоряют процесс создания приложений. Эти фреймворки предлагают готовую архитектуру и инструменты для работы с маршрутизацией, аутентификацией, шаблонизацией и многими другими аспектами веб-программирования.

Современные версии PHP, начиная с PHP 7, стали значительно быстрее и безопаснее благодаря улучшению механизма обработки данных и снижению нагрузки на сервер. Например, внедрение оператора «коалесцирования» (??) и «космического корабля» (<=>) сделало код более лаконичным и удобным.

PHP также активно используется для разработки API и веб-сервисов, обеспечивая взаимодействие между различными системами. Благодаря поддержке JSON, REST и других технологий PHP стyyал ключевым компонентом в построении современных веб-приложений.

## **5. MySQL**

MySQL — это система управления реляционными базами данных (СУБД), которая использует язык SQL (Structured Query Language) для работы с данными. MySQL была разработана компанией MySQL AB в 1995 году и на сегодняшний день является одной из самых популярных СУБД в мире, особенно в контексте веб-разработки. Она широко используется для хранения, обработки и извлечения данных в веб-приложениях, поддерживает множество различных типов данных и предоставляет мощные инструменты для работы с большими объемами информации.

MySQL является системой управления базами данных с открытым исходным кодом, что позволяет пользователям свободно использовать, изменять и распространять программу. Она обеспечивает высокую производительность, надежность и простоту в использовании, что делает её идеальным выбором для проектов любого масштаба — от небольших веб-сайтов до крупных корпоративных приложений.

Основные возможности MySQL включают создание и управление базами данных, таблицами, индексами и другими объектами, а также поддержку сложных запросов для извлечения, добавления, обновления и удаления данных. Работу с данными можно организовать через SQL-запросы, такие как:

SELECT — для извлечения данных.

INSERT — для добавления новых данных.

UPDATE — для изменения существующих данных.

DELETE — для удаления данных.

Пример простого SQL-запроса для извлечения всех данных из таблицы "products":

SELECT \* FROM products;

MySQL поддерживает такие важные функции, как транзакции, индексация, а также управление правами доступа. Это позволяет обеспечить безопасность и целостность данных в процессе работы с базой. Например, транзакции позволяют выполнять несколько операций как единое целое, гарантируя, что все изменения будут либо применены, либо откатаны в случае ошибки.

Кроме того, MySQL поддерживает разнообразные типы данных, включая числовые, строковые, даты, а также бинарные типы для хранения изображений или других файлов. Также возможна работа с внешними ключами и связями между таблицами, что позволяет проектировать сложные базы данных с нормализованными структурами.

Веб-разработчики часто используют MySQL в паре с серверным языком программирования, таким как PHP, для создания динамических веб-сайтов и приложений. Например, при создании формы на веб-странице, данные, введенные пользователем, могут быть сохранены в MySQL-базе данных с помощью SQL-запроса через PHP. Вот пример такого кода:

<?php

$mysqli = new mysqli("localhost", "user", "password", "database");

if ($mysqli->connect\_error) {

die("Ошибка подключения: " . $mysqli->connect\_error);

}

$name = $\_POST['name'];

$email = $\_POST['email'];

$query = "INSERT INTO users (name, email) VALUES ('$name', '$email')";

$mysqli->query($query);

?>

Для работы с базой данных MySQL можно использовать различные интерфейсы, такие как командная строка, графические панели управления (например, phpMyAdmin), а также библиотеки и фреймворки для облегчения работы с базой, например, PDO (PHP Data Objects) или MySQLi.

Одним из важных преимуществ MySQL является её способность масштабироваться, что позволяет использовать её как для малых, так и для крупных проектов. Она поддерживает репликацию, что позволяет синхронизировать данные между несколькими серверами и обеспечивать отказоустойчивость.

Кроме того, MySQL является высокопроизводительной СУБД, способной обрабатывать большое количество запросов и операций с данными. Эта производительность обеспечивается за счет эффективной оптимизации запросов, индексирования и кэширования, что позволяет ускорить доступ к данным.

С развитием технологий MySQL также получила поддержку таких современных функций, как хранение JSON-данных, полнотекстовый поиск и улучшенные механизмы работы с многозадачностью. Система продолжает активно развиваться и поддерживается большим сообществом разработчиков, что делает её надежным выбором для разработки и управления базами данных.

## **6. Brackets**

Brackets — это бесплатный и открытый текстовый редактор, предназначенный для веб-разработки. Он был разработан Adobe и ориентирован на работу с HTML, CSS и JavaScript. Brackets предоставляет ряд полезных функций, таких как живой предварительный просмотр (Live Preview), который позволяет моментально видеть изменения на веб-странице при редактировании кода, а также интеллектуальное дополнение кода, что ускоряет процесс разработки.

Brackets имеет интуитивно понятный интерфейс, поддержку плагинов для расширения функциональности и возможности для работы с несколькими файлами одновременно. Этот редактор идеально подходит для создания веб-приложений, так как помогает разработчику быстро проверять и оптимизировать свой код.

В процессе разработки веб-приложения для конфигураторов компьютеров я использовал Brackets для написания и редактирования HTML, CSS, JavaScript и PHP-кода. Удобные функции редактора, такие как подсветка синтаксиса и интеграция с браузером, позволили эффективно работать над проектом, ускоряя процесс тестирования и отладки.

## **7. XAMPP**

XAMPP — это популярный и бесплатный программный пакет, который включает в себя сервер Apache, базу данных MySQL, а также интерпретаторы PHP и Perl. XAMPP используется для создания локального веб-сервера на компьютере, что позволяет разработчикам тестировать веб-приложения и сайты на своей машине до их развертывания на удаленном сервере. С помощью XAMPP можно запустить веб-сайт, взаимодействующий с базой данных, без необходимости подключаться к удаленному серверу.

XAMPP предоставляет простой интерфейс для управления сервером, включая старт и остановку Apache и MySQL, а также возможность изменять настройки конфигурации. Веб-сервер Apache и база данных MySQL, включенные в XAMPP, обеспечивают полную поддержку для разработки динамических веб-приложений на PHP и MySQL, что делает его удобным инструментом для локальной разработки.

В процессе разработки веб-приложения для конфигуратора компьютера я использовал XAMPP для создания локальной среды. Этот пакет позволил мне легко настроить сервер, работать с базой данных MySQL и тестировать PHP-код без необходимости подключаться к внешнему серверу. XAMPP значительно упростил процесс разработки и отладки приложения.

# **Практическая часть**

## **Страница регистрации аккаунта**

Эта страница представляет собой форму регистрации пользователя на веб-сайте.

A) Подключение к базе данных и начало сессии

В начале скрипта происходит подключение к базе данных через файл db.php. Это позволяет взаимодействовать с базой данных для регистрации нового пользователя. Далее вызывается функция session\_start(), которая начинает сессию для пользователя, что необходимо для сохранения информации о пользователе в процессе его работы с сайтом.

Б) Обработка формы регистрации

Когда пользователь отправляет форму через метод POST, данные (логин, пароль и роль) из формы передаются на сервер. Эти данные сохраняются в переменных $username, $password и $role, соответственно.

Пароль перед сохранением в базе данных хешируется с использованием алгоритма PASSWORD\_BCRYPT для обеспечения безопасности.

B) Проверка наличия существующего логина

После получения данных из формы скрипт выполняет SQL-запрос, чтобы проверить, существует ли уже пользователь с таким же логином. Если логин уже занят, выводится ошибка с просьбой выбрать другой логин.

Г) Добавление нового пользователя в базу данных

Если логин уникален, данные о пользователе (логин, хешированный пароль и роль) записываются в таблицу users в базе данных. После этого происходит редирект на страницу входа, чтобы пользователь мог войти на сайт, используя только что созданный аккаунт.

Д) HTML-разметка

Форма регистрации состоит из полей для ввода логина, пароля и выбора роли. Эти поля обязательны для заполнения. После отправки формы данные отправляются на сервер.

Если произошла ошибка (например, логин уже занят), она отображается в виде красного текста в блоке с классом error.

Е) Дополнительные ссылки

Под формой есть ссылка на страницу входа для тех, кто уже зарегистрирован. Это позволяет пользователю перейти на страницу входа, если он не хочет снова регистрироваться.

Также имеется кнопка для перехода на главную страницу конфигуратора, что позволяет пользователю вернуться на основную страницу сайта.

Ж) Стилизация страницы

Страница имеет темный фон и контрастные элементы, что придает ей современный и стильный вид. Формы и кнопки имеют стили, обеспечивающие удобное взаимодействие. Эффекты при наведении на кнопки и поля ввода улучшают пользовательский опыт. Страница адаптирована под мобильные устройства, что позволяет корректно отображаться на различных экранах.

Итоговый функционал:

Пользователь может зарегистрироваться, введя логин, пароль и роль (например, "Пользователь" или "Администратор").

После успешной регистрации данные сохраняются в базе данных, а пользователь перенаправляется на страницу входа.

Если логин уже существует, отображается ошибка.

Дизайн страницы выполнен с акцентом на простоту и удобство использования, а также с адаптивным дизайном для мобильных устройств. (рисунок 1)

Код страницы:

<?php

include 'db.php';

session\_start();

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$username = $\_POST['username'];

$password = password\_hash($\_POST['password'], PASSWORD\_BCRYPT);

$role = $\_POST['role'];

$stmt = $pdo->prepare("SELECT COUNT(\*) FROM users WHERE username = :username");

$stmt->execute(['username' => $username]);

if ($stmt->fetchColumn() > 0) {

$error = "Логин уже существует.";

} else {

$stmt = $pdo->prepare("INSERT INTO users (username, password, role) VALUES (:username, :password, :role)");

$stmt->execute(['username' => $username, 'password' => $password, 'role' => $role]);

header('Location: login.php');

exit;

}

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Регистрация</title>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@500;600&display=swap" rel="stylesheet">

<style>

body { background-color: #121212; color: white; font-family: 'Poppins', sans-serif; display: flex; justify-content: center; align-items: center; min-height: 100vh; padding: 20px; }

.container { width: 100%; max-width: 400px; background-color: rgba(0, 0, 0, 0.85); padding: 40px; border-radius: 15px; box-shadow: 0 10px 30px rgba(0, 0, 0, 0.6); }

label, input, select, button { width: 100%; padding: 12px; margin: 10px 0; border: 1px solid #fff; background-color: #333; color: white; font-size: 16px; border-radius: 5px; }

button { background-color: #ff7f50; cursor: pointer; }

button:hover { background-color: #e67e22; }

.error { color: #f44336; font-size: 16px; text-align: center; }

.login-link { text-align: center; font-size: 16px; }

.login-link a { color: #ff7f50; text-decoration: none; }

</style>

</head>

<body>

<div class="container">

<h1>Регистрация</h1>

<?php if (isset($error)): ?>

<div class="error"><?= $error ?></div>

<?php endif; ?>

<form method="post">

<label for="username">Логин:</label>

<input type="text" name="username" id="username" placeholder="Введите логин" required>

<label for="password">Пароль:</label>

<input type="password" name="password" id="password" placeholder="Введите пароль" required>

<label for="role">Роль:</label>

<select name="role" id="role" required>

<option value="user">Пользователь</option>

<option value="admin">Администратор</option>

</select>

<button type="submit">Зарегистрироваться</button>

</form>

<div class="login-link">

<p>Уже зарегистрированы? <a href="login.php">Войдите здесь</a></p>

</div>

</div>

</body>

</html>

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 1 — Страница регистрации

## **2. Страница входа**

Эта страница представляет собой форму для входа пользователя в систему. Давайте разберем её функционал поэтапно.

A) Подключение к базе данных и проверка данных

В начале скрипта происходит подключение к базе данных через файл db.php. Это необходимо для выполнения SQL-запроса для проверки введённых пользователем данных.

Когда форма отправляется методом POST, данные (логин и пароль) из формы обрабатываются. Скрипт пытается найти пользователя с введённым логином в базе данных с помощью SQL-запроса. Если пользователь найден, проверяется, совпадает ли введённый пароль с тем, который хранится в базе (с помощью функции password\_verify).

Б) Проверка правильности данных

Если логин существует и введённый пароль соответствует тому, что сохранено в базе данных, то создается сессия для пользователя, и он перенаправляется на страницу dashboard.php. Это значит, что пользователь успешно вошел в систему.

Если данные неверны (неправильный логин или пароль), выводится сообщение об ошибке, информируя пользователя о том, что логин или пароль введены неверно.

B) HTML-разметка

Форма состоит из двух обязательных полей: одно для ввода логина и одно для пароля. Есть также кнопка для отправки формы. Если вход не удался, на экране появится сообщение об ошибке в красном цвете, предупреждающее пользователя о неверных данных.

Г) Дополнительные ссылки

Под формой есть ссылка на страницу регистрации, которая позволяет пользователю создать новый аккаунт, если у него его ещё нет. Также есть кнопка для перехода на главную страницу конфигуратора. Эта кнопка ведет на страницу configurator.php и позволяет пользователю вернуться на основную страницу сайта, не входя в систему.

Д) Стилизация страницы

Страница имеет темный фон с яркими элементами для контраста, чтобы подчеркнуть стильный и современный дизайн. Форма и кнопки стилизованы таким образом, чтобы обеспечить удобное взаимодействие с пользователем. Кнопки и поля ввода изменяют свой цвет при наведении для улучшения пользовательского опыта. Страница адаптируется под мобильные устройства, благодаря чему она выглядит корректно на разных экранах.

Итоговый функционал:

Пользователь может войти в систему, введя логин и пароль. Если данные верны, пользователь перенаправляется на страницу с личным кабинетом. В случае ошибки появляется сообщение, информирующее о неверном логине или пароле. Дизайн страницы выполнен с учётом удобства для пользователя и адаптивности под различные устройства. (рисунок 2)

Код страницы:

<?php

session\_start();

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

include 'db.php';

$username = $\_POST['username'];

$password = $\_POST['password'];

$stmt = $pdo->prepare("SELECT \* FROM users WHERE username = :username");

$stmt->execute(['username' => $username]);

$user = $stmt->fetch();

if ($user && password\_verify($password, $user['password'])) {

$\_SESSION['user'] = $user;

header('Location: dashboard.php');

exit;

} else {

echo "<div class='error'>Неверный логин или пароль!</div>";

}

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Вход</title>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@500;600&display=swap" rel="stylesheet">

<style>

body { background-color: #121212; color: white; font-family: 'Poppins', sans-serif; display: flex; justify-content: center; align-items: center; min-height: 100vh; }

form { width: 100%; max-width: 400px; background-color: rgba(0, 0, 0, 0.85); padding: 40px; border-radius: 15px; }

input, button { width: 100%; padding: 12px; margin: 10px 0; border: 1px solid #fff; background-color: #333; color: white; }

button { background-color: #ff7f50; cursor: pointer; }

.error { color: #f44336; font-size: 16px; margin-bottom: 20px; text-align: center; }

.register-link a { color: #ff7f50; text-decoration: none; }

</style>

</head>

<body>

<div>

<h1>Вход</h1>

<form method="post">

<input type="text" name="username" placeholder="Логин" required><br>

<input type="password" name="password" placeholder="Пароль" required><br>

<button type="submit">Войти</button>

</form>

<div class="register-link">

<p>Нет аккаунта? <a href="register.php">Зарегистрируйтесь здесь</a></p>

</div>

</div>

</body>

</html>

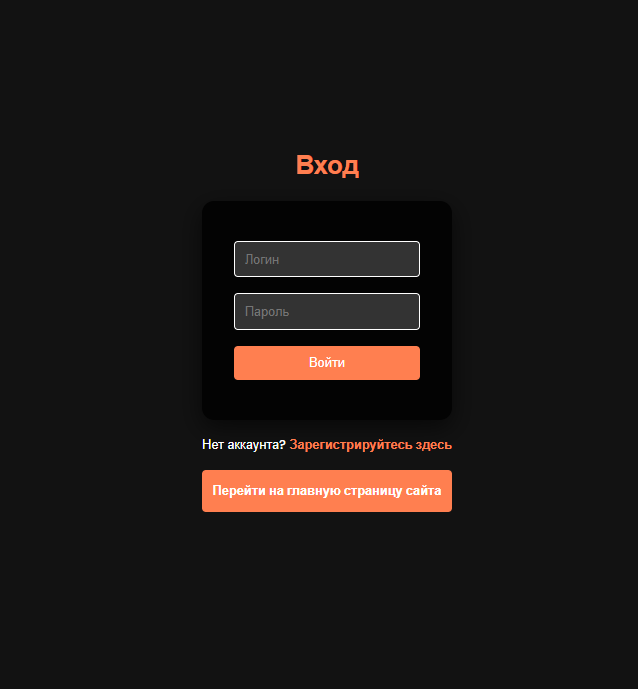


Рисунок 2 — Страница входа в аккаунт

## **Страница “Панель управления аккаунтом”**

Эта страница представляет собой панель управления для пользователя после его успешного входа в систему. Рассмотрим её функционал и основные элементы.

А) Проверка авторизации

На первой стадии скрипт проверяет, вошел ли пользователь в систему. Если сессия не установлена, пользователь перенаправляется на страницу входа. Это необходимо, чтобы только авторизованные пользователи могли получить доступ к панели управления.

Б) Приветствие пользователя

После успешной авторизации на странице выводится персонализированное приветствие, используя имя пользователя, которое извлекается из сессии. Это позволяет пользователю видеть, что он вошел в свою учетную запись.

В) Отображение ссылок в зависимости от роли  
В зависимости от роли пользователя отображаются разные ссылки. Например: если пользователь является администратором, то появляется ссылка на админ-панель. Если пользователь не администратор, он видит ссылку на редактирование своего профиля. Это позволяет предоставить разные функции в зависимости от прав пользователя.

Г) Общие ссылки

Независимо от роли пользователя, ему доступна ссылка для перехода на сайт по конфигурации ПК. Эта ссылка всегда видна на панели управления.

Д) Выход из системы

В нижней части страницы всегда доступна кнопка для выхода из учетной записи. При нажатии на кнопку пользователь будет разлогинен и перенаправлен на страницу входа. Это позволяет пользователю выйти из системы, когда это необходимо.

Е) Стилизация страницы

Страница использует темный фон с контрастными элементами, что придает современный и стильный вид. Внешний вид элементов страницы регулируется с помощью CSS, что делает страницу более привлекательной и улучшает пользовательский опыт.

Ж) Адаптивность

Страница адаптируется под мобильные устройства, чтобы обеспечить комфортное использование на различных экранах. Это делает страницу удобной для использования на разных устройствах.

Итоговый функционал страницы:

Приветствует пользователя, отображая его имя.

В зависимости от роли пользователя (администратор или обычный пользователь) показываются различные ссылки.

Все пользователи могут перейти на сайт по конфигурации ПК и выйти из своей учетной записи.

Страница имеет стильный и современный дизайн с адаптацией для мобильных устройств. (рисунок 3)

Код страницы:

<?php

session\_start();

if (!isset($\_SESSION['user'])) {

header('Location: login.php');

exit;

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Панель управления</title>

<style>

body {

background-color: black;

color: white;

font-family: 'Poppins', sans-serif;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

text-align: center;

}

.container {

background-color: rgba(0, 0, 0, 0.85);

padding: 40px;

border-radius: 15px;

box-shadow: 0 10px 30px rgba(0, 0, 0, 0.6);

}

h1 { color: #ff7f50; font-size: 32px; margin-bottom: 20px; }

p { color: #ccc; font-size: 18px; margin-bottom: 30px; }

a {

display: inline-block;

padding: 15px 30px;

color: black;

background-color: white;

border-radius: 8px;

text-decoration: none;

}

.logout { background-color: red; color: white; padding: 15px 30px; }

</style>

</head>

<body>

<div class="container">

<h1>Панель управления</h1>

<p>Добро пожаловать, <?php echo $\_SESSION['user']['username']; ?>!</p>

<?php if ($\_SESSION['user']['role'] === 'admin'): ?>

<a href="admin.php">Админ панель</a>

<?php else: ?>

<a href="edit.php">Редактировать профиль</a>

<?php endif; ?>

<a href="configurator.php">Перейти на сайт по конфигурации ПК</a>

<a href="logout.php" class="logout">Выйти</a>

</div>

</body>

</html>

**Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание**

Рисунок 3 — Панель управления аккаунтом

## **Страница “Админ панель”**

Админ панель — это интерфейс для администратора, который позволяет управлять пользователями и контролировать доступ к различным функциям сайта. В этом коде админ панель реализована для управления пользователями с возможностью их редактирования и удаления.

Описание функционала:

А) Авторизация: Администратор должен быть авторизован и иметь роль "admin". Если пользователь не авторизован или не является администратором, он не получит доступ к панели и будет перенаправлен или получит сообщение о запрете доступа.

Б) Просмотр списка пользователей: В админ панели отображается таблица со списком всех пользователей, которая включает следующие столбцы:

Логин: отображает имя пользователя.

Роль: показывает роль пользователя (например, "user" или "admin").

Действия: предоставляет ссылки для редактирования или удаления пользователя.

В) Добавление нового пользователя: есть кнопка "Добавить нового пользователя", которая ведет на страницу для создания нового пользователя.

Г) Редактирование пользователя: для каждого пользователя в таблице есть кнопка "Редактировать", которая открывает страницу редактирования данных пользователя.

Д) Удаление пользователя: Каждому пользователю на странице присваивается кнопка "Удалить". При нажатии на нее появляется подтверждение действия, что предотвращает случайные удаления.

Е) Переход на сайт конфигуратора ПК: есть кнопка для перехода на внешний сайт, связанный с конфигуратором ПК. Это может быть полезно, если администратор хочет провести пользователей через этот инструмент.

Ж) Выход: внизу страницы расположена кнопка для выхода из системы, что завершает сессию администратора и перенаправляет его на страницу логина.

Основные элементы интерфейса:

А) Таблица с пользователями: отображает данные всех пользователей в виде строк с логинами и ролями, а также действиями для редактирования и удаления.

Б) Кнопки для взаимодействия: кнопки для добавления, редактирования, удаления пользователей и выхода из системы. В) Ссылки и кнопки с анимациями: при наведении на кнопки происходит изменение фона и анимация, улучшая пользовательский опыт.

Особенности:

А) Безопасность: Администратор должен быть авторизован для доступа к панели, и любые действия, такие как удаление пользователей, защищены подтверждением.

Б) Управление пользователями: администратор может управлять ролью и доступом пользователей, редактируя их данные или удаляя учетные записи.

Таким образом, админ панель предоставляет удобный и безопасный способ для администрирования пользователей на сайте, обеспечивая гибкость в управлении и контроле за доступом. (рисунок 4)

Код страницы:

<?php

include 'db.php';

session\_start();

// Проверка, если пользователь админ

if (!isset($\_SESSION['user']) || $\_SESSION['user']['role'] !== 'admin') {

die("Доступ запрещён!");

}

// Получаем список пользователей

$stmt = $pdo->query("SELECT \* FROM users");

$users = $stmt->fetchAll();

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Админ панель</title>

<style>

body { background-color: #121212; color: white; font-family: 'Poppins', sans-serif; display: flex; justify-content: center; align-items: center; min-height: 100vh; }

.container { max-width: 900px; background-color: rgba(0, 0, 0, 0.8); padding: 40px; border-radius: 15px; box-shadow: 0 10px 30px rgba(0, 0, 0, 0.6); }

h1 { font-size: 36px; color: #ff7f50; text-align: center; }

table { width: 100%; border-collapse: collapse; }

th, td { padding: 20px; border: 1px solid #444; text-align: left; }

th { background-color: #333; color: white; }

td { background-color: #444; color: #ddd; }

a { color: black; background-color: white; padding: 12px 20px; border-radius: 8px; text-decoration: none; transition: background-color 0.3s ease; }

a:hover { background-color: #ddd; }

.logout { background-color: red; color: white; padding: 12px 20px; margin-top: 30px; text-align: center; width: 100%; }

</style>

</head>

<body>

<div class="container">

<h1>Админ панель</h1>

<a href="adduser.php">Добавить нового пользователя</a>

<table>

<thead>

<tr>

<th>Логин</th>

<th>Роль</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

<?php foreach ($users as $user): ?>

<tr>

<td><?php echo htmlspecialchars($user['username']); ?></td>

<td><?php echo htmlspecialchars($user['role']); ?></td>

<td>

<a href="edit.php?id=<?php echo $user['id']; ?>">Редактировать</a>

<a href="delete.php?id=<?php echo $user['id']; ?>" onclick="return confirm('Вы уверены?');">Удалить</a>

</td>

</tr>

<?php endforeach; ?>

</tbody>

</table>

<a href="configurator.php">Перейти на сайт конфигуратора ПК</a>

<a href="logout.php" class="logout">Выйти</a>

</div>

</body>

</html>

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание**

Рисунок 4 — Админ панель

## **Страница “Редактор аккаунта пользователя”**

Эта страница позволяет редактировать профиль пользователя. Важные моменты, которые следует учитывать:

А) Проверка прав доступа

Если пользователь не авторизован, или если он не является администратором и пытается редактировать профиль другого пользователя, он будет перенаправлен с сообщением "Доступ запрещён!".

Если id пользователя передан в URL, проверяется, может ли текущий пользователь редактировать данные этого пользователя. Если это не так, отображается сообщение об ошибке.

Б) Получение и отображение данных пользователя

Данные пользователя (логин и роль) извлекаются из базы данных, если пользователь найден.

Страница отображает форму с полями для ввода логина и пароля.

Если роль пользователя — админ, добавляется выпадающий список для выбора роли (пользователь или администратор).

В) Обработка формы

При отправке формы:

Логин проверяется на уникальность (если был изменён).

Пароль может быть изменён, если пользователь ввёл новый пароль. Если пароль не изменён, сохраняется текущий.

Роль пользователя также может быть изменена (если текущий пользователь является администратором).

Г) Сообщения об успехе или ошибке

Если изменения сохранены, выводится сообщение о том, что пароль был изменён (если он был изменён).

Если возникла ошибка (например, логин уже занят), отображается соответствующее сообщение об ошибке.

Д) Дополнительные элементы интерфейса

Внизу страницы есть ссылка на страницу пользователя для возврата в панель управления.

Все элементы формы стилизованы для красивого и удобного интерфейса с использованием простых анимаций (например, изменение фона при наведении на кнопки).

Эта страница предоставляет интерфейс для безопасного редактирования пользовательских данных с необходимыми проверками и удобным интерфейсом. (рисунок 5)

Код страницы:

<?php

include 'db.php';

session\_start();

// Проверка доступа

if (!isset($\_SESSION['user'])) {

die("Доступ запрещён!");

}

$id = isset($\_GET['id']) ? (int)$\_GET['id'] : $\_SESSION['user']['id'];

if ($\_SESSION['user']['role'] !== 'admin' && $\_SESSION['user']['id'] !== $id) {

die("У вас нет прав для редактирования этого пользователя!");

}

// Получение данных пользователя

$stmt = $pdo->prepare("SELECT \* FROM users WHERE id = :id");

$stmt->execute(['id' => $id]);

$user = $stmt->fetch();

if (!$user) {

die("Пользователь не найден!");

}

// Обработка изменений

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$username = $\_POST['username'];

$password = !empty($\_POST['password']) ? password\_hash($\_POST['password'], PASSWORD\_BCRYPT) : $user['password'];

$role = $\_POST['role'] ?? $user['role'];

// Проверка на уникальность логина

if ($username !== $user['username']) {

$checkStmt = $pdo->prepare("SELECT COUNT(\*) FROM users WHERE username = :username AND id != :id");

$checkStmt->execute(['username' => $username, 'id' => $id]);

if ($checkStmt->fetchColumn() > 0) {

$errorMessage = "Логин уже занят!";

} else {

$stmt = $pdo->prepare("UPDATE users SET username = :username, password = :password, role = :role WHERE id = :id");

$stmt->execute(['username' => $username, 'password' => $password, 'role' => $role, 'id' => $id]);

}

} else {

$stmt = $pdo->prepare("UPDATE users SET username = :username, password = :password, role = :role WHERE id = :id");

$stmt->execute(['username' => $username, 'password' => $password, 'role' => $role, 'id' => $id]);

}

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Редактирование пользователя</title>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@500;600&display=swap" rel="stylesheet">

<style>

body { background-color: #121212; color: white; font-family: 'Poppins', sans-serif; display: flex; justify-content: center; align-items: center; min-height: 100vh; padding: 20px; }

h1 { font-size: 32px; color: #ff7f50; margin-bottom: 20px; text-align: center; }

.container { width: 100%; max-width: 400px; background-color: rgba(0, 0, 0, 0.85); padding: 40px; border-radius: 15px; box-shadow: 0 10px 30px rgba(0, 0, 0, 0.6); }

input, select, button { width: 100%; padding: 12px; margin: 10px 0; border: 1px solid #fff; background-color: #333; color: white; font-size: 16px; border-radius: 5px; }

button { background-color: #ff7f50; cursor: pointer; }

button:hover { background-color: #e67e22; }

.success-message { color: #4CAF50; font-size: 16px; margin-bottom: 20px; text-align: center; }

.error-message { color: #f44336; font-size: 16px; margin-bottom: 20px; text-align: center; }

</style>

</head>

<body>

<div class="container">

<h1>Редактирование пользователя</h1>

<?php if (isset($errorMessage)): ?>

<div class="error-message"><?= $errorMessage ?></div>

<?php endif; ?>

<form method="post">

<label for="username">Логин:</label>

<input type="text" name="username" id="username" value="<?= htmlspecialchars($user['username']) ?>" required><br>

<label for="password">Пароль:</label>

<input type="password" name="password" id="password"><br>

<?php if ($\_SESSION['user']['role'] === 'admin'): ?>

<label for="role">Роль:</label>

<select name="role" id="role">

<option value="user" <?= $user['role'] === 'user' ? 'selected' : '' ?>>User</option>

<option value="admin" <?= $user['role'] === 'admin' ? 'selected' : '' ?>>Admin</option>

</select><br>

<?php endif; ?>

<button type="submit">Сохранить изменения</button>

</form>

</div>

</body>

</html>

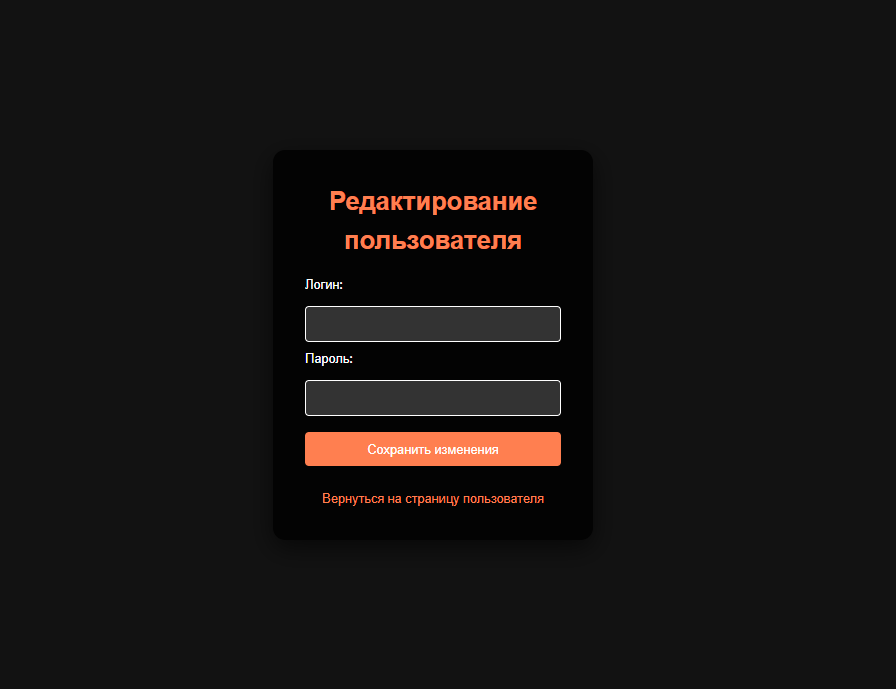


Рисунок 5 — Редактор аккаунта пользователя

## **Главная страница сайта**

Эта страница представляет собой главную страницу конфигуратора ПК. На ней пользователи могут собирать компьютер, выбирая компоненты из различных категорий, таких как процессор, оперативная память, видеокарта и другие, а также видеть итоговую стоимость выбранных компонентов. Ниже приведено более подробное описание элементов страницы:

1. Шапка страницы

Аккаунт: В верхнем правом углу находится ссылка на аккаунт пользователя. Если пользователь авторизован, он может перейти в личный кабинет.

Наши партнёры: Внизу страницы присутствует ссылка на раздел с партнерами компании, который располагается на внешней странице.

Заголовок: Заголовок страницы — "Конфигуратор ПК", который выделен ярким оранжевым цветом (#ff7f50) и помогает пользователю понять, что это основная страница для настройки ПК.

2. Основная часть (Категории комплектующих)

Страница разделена на несколько секций, каждая из которых представляет собой отдельную категорию комплектующих для ПК. В каждой секции присутствует:

Иконка: Каждая категория компонентов имеет свою иконку, которая помогает визуально идентифицировать компоненты.

Название категории: например, "Процессор", "Оперативная память", "Видеокарта" и т.д.

Выпадающий список: Пользователи могут выбрать компоненты из выпадающего списка. Каждое оборудование имеет цену, которая отображается в рублях. Когда пользователь выбирает компонент, его стоимость автоматически обновляется на странице.

Категории:

Процессор: включает варианты процессоров Intel и AMD с ценами от 10,000 до 32,000 ₽.

Оперативная память: Варианты по объему и брендам (например, Kingston, Corsair, G.Skill) с ценами от 4,000 до 16,000 ₽.

Видеокарта: Варианты от NVIDIA и AMD, цены от 20,000 до 70,000 ₽.

Материнская плата: Разнообразие материнских плат с ценами от 5,000 до 15,000 ₽.

Накопитель: SSD и HDD, начиная с 3,000 ₽.

Блок питания: Блоки питания с мощностью от 450W до 850W, начиная с 3,000 ₽.

Корпус: Корпуса для ПК от простых до премиум-класса, с ценами от 2,000 до 8,000 ₽.

3. Вывод информации о стоимости

Внизу страницы есть блок с общей суммой, которая зависит от выбранных компонентов.

Общая стоимость: это сумма всех выбранных компонентов, которая обновляется динамически при изменении выбора. Она отображается в отдельном блоке, и цена обновляется в реальном времени.

Список выбранных компонентов: слева от общей стоимости отображается список выбранных компонентов с их ценами. Если компонент выбран, он добавляется в список, если нет — он удаляется. Если пользователь не выбрал компоненты, отображается текст "Выбранные компоненты отсутствуют".

Ссылка на оформление заказа: внизу блока с суммой и списком компонентов находится кнопка "Оформить заказ". Эта ссылка ведет на страницу оформления заказа, где пользователь может подать заявку на покупку конфигурируемого ПК.

4. Интерактивность с помощью JavaScript

Страница использует JavaScript для динамического обновления стоимости. Каждый раз, когда пользователь выбирает компонент в выпадающем списке, функция updateTotal обновляет общую стоимость и список выбранных компонентов.

Функция updateTotal: Она:

Получает выбранную цену из выпадающего списка.

Обновляет объект selectedComponents, который хранит информацию о выбранных компонентах.

Пересчитывает общую стоимость всех выбранных компонентов.

Обновляет блок с общей стоимостью и список выбранных компонентов.

5. Ссылки

Популярные сборки: Ссылка на страницу, где представлены популярные сборки ПК. Эта ссылка направляет пользователя на другую страницу с заранее подготовленными сборками, что упрощает процесс для тех, кто не хочет собирать ПК вручную.

6. Адаптивность и стиль

Гибкость макета: Страница адаптируется под разные размеры экранов благодаря использованию flexbox для организации блоков. Элементы, такие как категории комплектующих, располагаются в строках и оборачиваются на мобильных устройствах.

Темный фон: Стиль выполнен в темной цветовой гамме, что помогает выделить контент и создать приятный визуальный опыт для пользователя.

Тени и анимации: используются плавные анимации и тени для выделения компонентов при наведении, что добавляет интерактивности и привлекательности интерфейсу.

7. Оформление заказа

Внизу страницы пользователю предлагается перейти к оформлению заказа. Это позволяет пользователю, после того как он собрал свою конфигурацию, приступить к оформлению покупки. (рисунок 6)

Код страницы:

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Конфигуратор ПК</title>

<style>

body {

background-color: #1a1a1a;

color: white;

font-family: 'Poppins', sans-serif;

padding: 20px;

display: flex;

flex-direction: column;

align-items: center;

}

.category {

width: 300px;

background-color: rgba(0, 0, 0, 0.85);

padding: 20px;

border-radius: 15px;

box-shadow: 0 10px 30px rgba(0, 0, 0, 0.6);

text-align: center;

}

.category select {

width: 100%;

padding: 10px;

border-radius: 8px;

background-color: rgba(0, 0, 0, 0.7);

color: white;

}

h1 { color: #ff7f50; margin-bottom: 20px; font-size: 32px; }

.summary { background-color: rgba(30, 30, 30, 0.95); padding: 20px; border-radius: 15px; }

.total { font-size: 20px; text-align: center; }

.selected-items { font-size: 16px; background-color: rgba(50, 50, 50, 0.8); padding: 10px; border-radius: 8px; max-height: 200px; overflow-y: auto; }

.order-link { background-color: #ff7f50; color: white; padding: 10px 20px; border-radius: 8px; text-decoration: none; }

</style>

</head>

<body>

<h1>Конфигуратор ПК</h1>

<div class="category">

<h3>Процессор</h3>

<select onchange="updateTotal(this, 'Процессор')">

<option value="0">Не выбрано</option>

<option value="10000">Intel Core i3 - 10,000 ₽</option>

<option value="15000">Intel Core i5 - 15,000 ₽</option>

</select>

</div>

<div class="category">

<h3>Оперативная память</h3>

<select onchange="updateTotal(this, 'Оперативная память')">

<option value="0">Не выбрано</option>

<option value="4000">Kingston Fury 16GB - 4,000 ₽</option>

<option value="8000">Corsair Vengeance 32GB - 8,000 ₽</option>

</select>

</div>

<div class="category">

<h3>Видеокарта</h3>

<select onchange="updateTotal(this, 'Видеокарта')">

<option value="0">Не выбрано</option>

<option value="20000">NVIDIA GTX 1660 - 20,000 ₽</option>

<option value="40000">NVIDIA RTX 3060 - 40,000 ₽</option>

</select>

</div>

<div class="summary">

<h2>Общая стоимость</h2>

<div id="total" class="total">0 ₽</div>

<div class="selected-items" id="selectedItems">Выбранные компоненты отсутствуют</div>

<a href="forma.php" class="order-link">Оформить заказ</a>

</div>

<script>

const selectedComponents = {};

function updateTotal(selectElement, category) {

const price = parseInt(selectElement.value);

const componentName = selectElement.parentElement.querySelector('h3').innerText;

if (price > 0) {

selectedComponents[componentName] = { price: price };

} else {

delete selectedComponents[componentName];

}

let total = 0;

Object.values(selectedComponents).forEach(component => { total += component.price; });

document.getElementById('total').innerText = total + ' ₽';

const selectedItems = Object.keys(selectedComponents).map(key => `${key} - ${selectedComponents[key].price} ₽`).join('<br>');

document.getElementById('selectedItems').innerHTML = selectedItems || 'Выбранные компоненты отсутствуют';

}

</script>

</body>

</html>

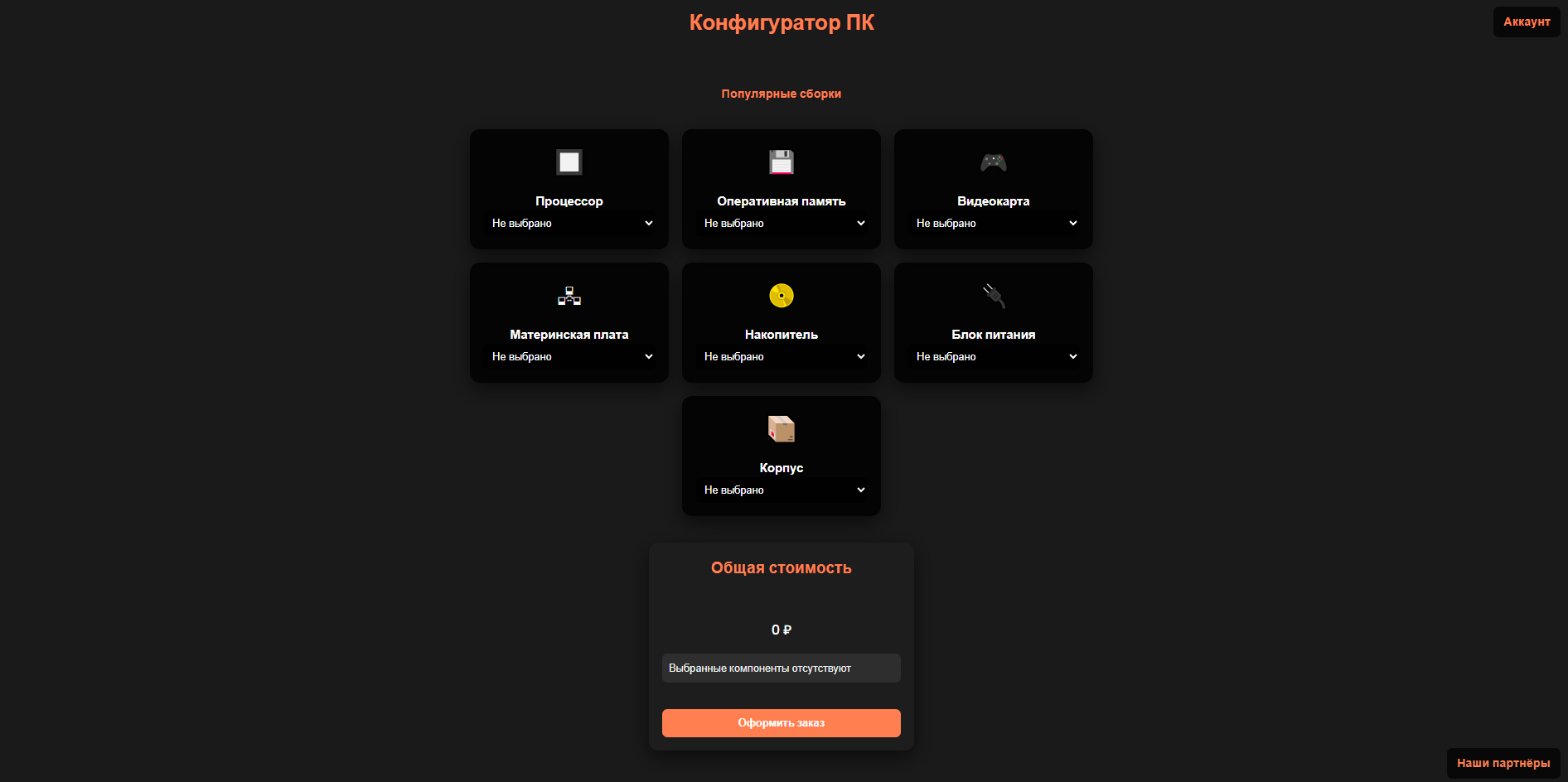


Рисунок 6 — Главная страница сайта

## **Форма для оформления заказа**

Эта страница представляет собой форму для оформления заказа.

Структура:

Поля формы: Имя, email, телефон, адрес доставки, способы доставки и оплаты.

Кнопка отправки формы. После отправки показывается сообщение о подтверждении заказа.

Дизайн (CSS):

Темная тема: Фон страницы черный с белым текстом, элементы формы имеют закругленные углы и оранжевые акценты.

Анимация: Сообщение об успешном заказе появляется с плавной анимацией.

Функционал (JavaScript):

Обработка отправки формы: при отправке форма не перезагружает страницу, показывается сообщение об успешном оформлении заказа, которое исчезает через 4 секунды.

Обработка данных (PHP):

Безопасность: Данные из формы обрабатываются с помощью htmlspecialchars для предотвращения XSS-атак.

Данные передаются в серверный скрипт для дальнейшей обработки.

Эта страница предоставляет простой и стильный интерфейс для оформления заказа с базовой валидацией и интерактивными элементами. (рисунок 7)

Код страницы:

<?php

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$name = htmlspecialchars($\_POST['name']);

$email = htmlspecialchars($\_POST['email']);

$phone = htmlspecialchars($\_POST['phone']);

$address = htmlspecialchars($\_POST['address']);

$payment\_method = $\_POST['payment\_method'];

$delivery\_method = $\_POST['delivery\_method'];

}

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Оформление заказа</title>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@500;600&display=swap" rel="stylesheet">

<style>

body { background-color: #1a1a1a; color: white; font-family: 'Poppins', sans-serif; padding: 40px; display: flex; flex-direction: column; gap: 20px; align-items: center; }

h1 { color: #ff7f50; font-size: 32px; margin-bottom: 40px; text-align: center; }

.form-container { width: 100%; max-width: 800px; background-color: rgba(30, 30, 30, 0.95); padding: 20px; border-radius: 15px; box-shadow: 0 10px 30px rgba(0, 0, 0, 0.6); }

.form-group { margin-bottom: 20px; }

.form-group label { color: #ff7f50; font-size: 16px; margin-bottom: 10px; }

.form-group input, .form-group select, .form-group textarea { width: 100%; padding: 10px; border-radius: 8px; border: none; font-size: 16px; background-color: rgba(255, 255, 255, 0.1); color: white; }

.form-group button { background-color: #ff7f50; color: white; padding: 10px 20px; border-radius: 8px; font-size: 18px; cursor: pointer; width: 100%; }

.form-group button:hover { background-color: #e06f39; }

.order-success { display: none; padding: 20px; background-color: #28a745; color: white; font-size: 18px; border-radius: 10px; margin-top: 20px; }

.back-link { color: #ff7f50; font-size: 18px; text-decoration: none; text-align: center; margin-top: 20px; }

.back-link:hover { text-decoration: underline; }

</style>

</head>

<body>

<h1>Оформление заказа</h1>

<div class="form-container">

<form action="forma.php" method="POST">

<div class="form-group">

<label for="name">Ваше имя</label>

<input type="text" id="name" name="name" required placeholder="Введите ваше имя">

</div>

<div class="form-group">

<label for="email">Электронная почта</label>

<input type="email" id="email" name="email" required placeholder="Введите ваш email">

</div>

<div class="form-group">

<label for="phone">Телефон</label>

<input type="tel" id="phone" name="phone" required placeholder="Введите ваш номер телефона">

</div>

<div class="form-group">

<label for="address">Адрес доставки</label>

<textarea id="address" name="address" required placeholder="Введите ваш адрес"></textarea>

</div>

<div class="form-group">

<label for="delivery\_method">Способ доставки</label>

<select id="delivery\_method" name="delivery\_method" required>

<option value="курьер">Курьером</option>

<option value="пункт самовывоза">Пункт самовывоза</option>

<option value="почта">Почта</option>

</select>

</div>

<div class="form-group">

<label for="payment\_method">Способ оплаты</label>

<select id="payment\_method" name="payment\_method" required>

<option value="наличными">Наличными при получении</option>

<option value="картой">Картой при доставке</option>

<option value="онлайн">Онлайн оплата</option>

</select>

</div>

<div class="form-group">

<button type="submit">Оформить заказ</button>

</div>

</form>

<div id="success-message" class="order-success">Заказ оформлен!</div>

<a href="configurator.php" class="back-link">Вернуться в конфигуратор</a>

</div>

<script>

document.querySelector('form').addEventListener('submit', function(event) {

event.preventDefault();

document.getElementById('success-message').style.display = 'block';

setTimeout(() => document.getElementById('success-message').style.display = 'none', 4000);

});

</script>

</body>

</html>

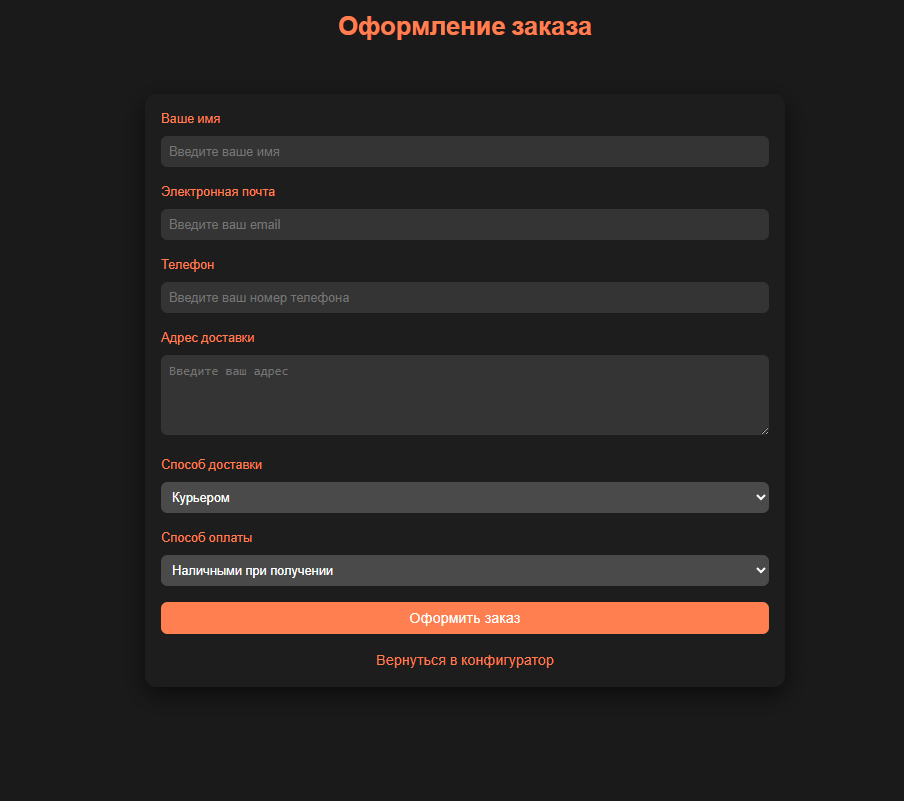


Рисунок 7 — Форма оформления заказа

## **Страница “Наши партнёры”**

Страница предназначена для демонстрации партнеров компании и содержит ссылки на их официальные сайты. Основные элементы:

1. Общий стиль

Темный фон с яркими акцентами оранжевого цвета (#ff7f50) для заголовков, ссылок и иконок.

Шрифт Poppins создает современный вид.

Минималистичный и адаптивный дизайн.

2. Структура

Заголовок: "Наши партнёры" выделен крупным шрифтом.

Сетка партнеров:

Карточки партнеров, оформленные с помощью CSS Grid.

Каждая карточка содержит:

Иконку (Font Awesome).

Название партнера.

Ссылку для перехода на сайт компании.

Кнопка возврата: позволяет вернуться в конфигуратор.

3. Анимация

Наведение на карточку увеличивает её, меняет цвет фона и активирует подчеркивание.

Адаптивность позволяет комфортно использовать страницу на любых устройствах.

4. Функциональность

Переход на сайт партнера в новой вкладке.

Современный, легкий интерфейс для быстрой навигации.

Страница проста, функциональна и подходит для представления партнеров. (рисунок 8)

Код страницы:

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Наши партнёры</title>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@500;600&display=swap" rel="stylesheet">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">

<style>

body {

background: #1a1a1a; color: white; font-family: 'Poppins', sans-serif; text-align: center; padding: 20px;

}

h1 { color: #ff7f50; margin-bottom: 20px; }

.partners-list {

display: grid; grid-template-columns: repeat(auto-fill, minmax(150px, 1fr)); gap: 20px;

}

.partner {

background: rgba(0, 0, 0, 0.7); padding: 20px; border-radius: 10px; transition: 0.3s;

}

.partner:hover { transform: scale(1.05); background: rgba(0, 0, 0, 0.9); }

.partner i { font-size: 40px; color: #ff7f50; margin-bottom: 10px; }

.partner h3 { color: #ff7f50; font-size: 18px; }

.partner a { color: #ff7f50; text-decoration: none; font-size: 14px; }

.back-link { color: #ff7f50; text-decoration: none; margin-top: 20px; display: inline-block; }

</style>

</head>

<body>

<h1>Наши партнёры</h1>

<div class="partners-list">

<div class="partner">

<i class="fab fa-asus"></i>

<h3>Asus</h3>

<a href="https://www.asus.com" target="\_blank">Перейти на сайт</a>

</div>

<div class="partner">

<i class="fab fa-msi"></i>

<h3>MSI</h3>

<a href="https://www.msi.com" target="\_blank">Перейти на сайт</a>

</div>

<div class="partner">

<i class="fab fa-nvidia"></i>

<h3>NVIDIA</h3>

<a href="https://www.nvidia.com" target="\_blank">Перейти на сайт</a>

</div>

<div class="partner">

<i class="fas fa-microchip"></i>

<h3>Intel</h3>

<a href="https://www.intel.com" target="\_blank">Перейти на сайт</a>

</div>

</div>

<a href="configurator.php" class="back-link">Вернуться в конфигуратор</a>

</body>

</html>

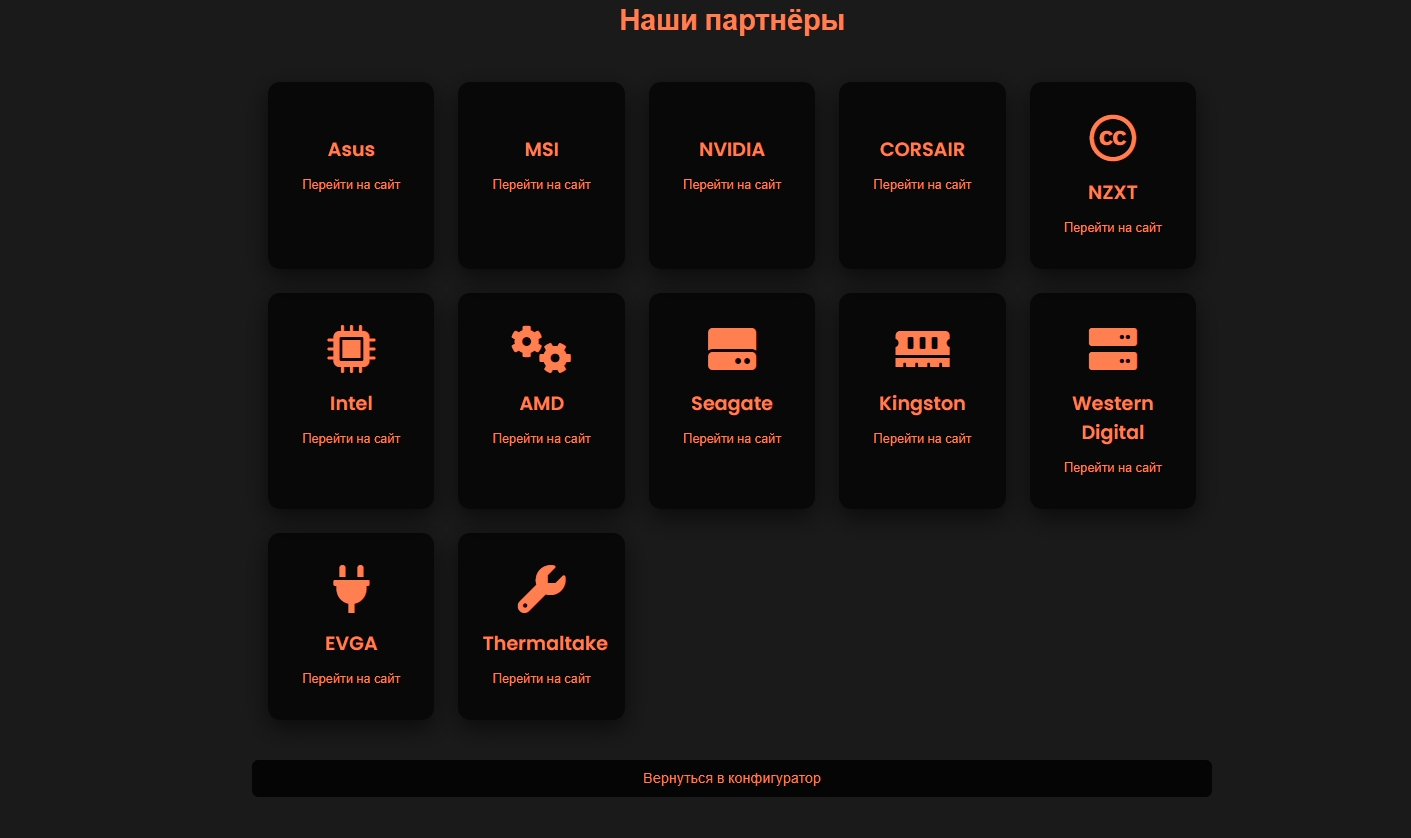


Рисунок 8 — Страница “Наши партнёры”

## **Страница с готовыми сборками ПК**

Эта страница представляет собой веб-интерфейс для работы с готовыми сборками ПК. Она позволяет пользователям просматривать доступные сборки, а администраторам — добавлять или удалять сборки. Рассмотрим основные аспекты реализации.

1. Функционал

Администратор:

Может добавлять новые сборки (форма для добавления с полями: название, цена, описание). Удалять существующие сборки через кнопку "Удалить".

Пользователь:

Видит только активные сборки. Может оформить заказ через кнопку "Оформить заказ".

1. Действия с базой данных:

Добавление сборки: данные передаются через форму методом POST и сохраняются в таблицу popular\_builds. Удаление сборки: используется идентификатор сборки (ID), чтобы удалить запись из базы данных. Отображение сборок: извлечение данных из таблицы с учетом роли пользователя:

Для администратора: отображаются все сборки.

Для пользователя: только активные.

1. Реализация на PHP

Проверка сессии:

При старте страницы выполняется проверка, авторизован ли пользователь через $\_SESSION['user']. Роль пользователя (admin или user) определяет доступный функционал.

Работа с базой данных:

Подключение к базе через require\_once 'db\_connection.php'. Подготовленные SQL-запросы (prepare) защищают от SQL-инъекций.

Выполнение запросов:

Добавление: INSERT INTO popular\_builds. Удаление: DELETE FROM popular\_builds. Получение данных: выборка всех или только активных сборок.

Обработка форм:

Все действия (add, delete) обрабатываются в зависимости от метода запроса (POST) и переданного значения поля action.

1. Сообщения:

Пользователь получает уведомления о выполнении действий (успех или ошибка). Интерфейс пользователя (HTML + CSS)

1. Стиль и оформление:

Фон: темный (#1a1a1a) для удобного восприятия. Основной акцент (#ff7f50) выделяет заголовки и кнопки. Анимации при наведении добавляют интерактивности.

1. Разметка:

Заголовок h1 указывает на страницу "Готовые сборки ПК".

Блоки сборок:

Название, цена, описание и статус.

Кнопки действия: "Оформить заказ" для пользователей и "Удалить" для администраторов.

Форма добавления сборки (для администратора): включает поля ввода и кнопку "Добавить сборку".

1. Адаптивность:

Элементы страницы структурированы так, чтобы корректно отображаться на устройствах разного размера.

1. Особенности

Роли пользователей: логика на основе роли позволяет ограничивать функционал для разных категорий пользователей.

Защита данных: использование подготовленных запросов ($stmt->bind\_param) минимизирует риски SQL-инъекций.

Динамичное отображение: контент (сборки) отображается в зависимости от данных в базе.

1. Пример сценария использования

Пользователь:

Заходит на страницу и видит список доступных сборок. Выбирает интересующую сборку и нажимает "Оформить заказ", чтобы перейти к форме заказа. (рисунок 9)

Администратор:

Авторизуется и получает доступ к форме добавления новых сборок. Добавляет новую сборку, указывая название, цену и описание. При необходимости удаляет устаревшие сборки через соответствующую кнопку. (рисунок 10)

Код страницы:

<?php

session\_start();

require\_once 'db\_connection.php';

$user\_role = $\_SESSION['user']['role'] ?? 'user';

// Добавление сборки (только для администратора)

if ($user\_role === 'admin' && $\_SERVER['REQUEST\_METHOD'] === 'POST' && $\_POST['action'] === 'add') {

$name = $\_POST['name'] ?? '';

$price = $\_POST['price'] ?? '';

$description = $\_POST['description'] ?? '';

if ($name && $price && $description) {

$stmt = $conn->prepare("INSERT INTO popular\_builds (name, price, description, is\_active) VALUES (?, ?, ?, 1)");

$stmt->bind\_param("sds", $name, $price, $description);

$message = $stmt->execute() ? "Сборка добавлена!" : "Ошибка: " . $stmt->error;

$stmt->close();

} else {

$message = "Заполните все поля.";

}

}

// Удаление сборки (только для администратора)

if ($user\_role === 'admin' && $\_SERVER['REQUEST\_METHOD'] === 'POST' && $\_POST['action'] === 'delete') {

$build\_id = intval($\_POST['build\_id']);

$stmt = $conn->prepare("DELETE FROM popular\_builds WHERE id = ?");

$stmt->bind\_param("i", $build\_id);

$message = $stmt->execute() ? "Сборка удалена!" : "Ошибка: " . $stmt->error;

$stmt->close();

}

// Получение списка сборок

$sql = $user\_role === 'admin' ? "SELECT \* FROM popular\_builds" : "SELECT \* FROM popular\_builds WHERE is\_active = 1";

$builds = $conn->query($sql)->fetch\_all(MYSQLI\_ASSOC);

$conn->close();

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Готовые сборки ПК</title>

</head>

<body>

<h1>Готовые сборки ПК</h1>

<?php if (!empty($message)): ?><p><?= htmlspecialchars($message) ?></p><?php endif; ?>

<?php if ($user\_role === 'admin'): ?>

<form method="POST">

<input type="hidden" name="action" value="add">

<input type="text" name="name" placeholder="Название" required>

<input type="number" name="price" placeholder="Цена" required>

<textarea name="description" placeholder="Описание" required></textarea>

<button type="submit">Добавить</button>

</form>

<?php endif; ?>

<?php foreach ($builds as $build): ?>

<div>

<h2><?= htmlspecialchars($build['name']) ?></h2>

<p>Цена: <?= htmlspecialchars($build['price']) ?> ₽</p>

<p><?= htmlspecialchars($build['description']) ?></p>

<?php if ($user\_role === 'user'): ?>

<a href="forma.php?id=<?= $build['id'] ?>">Оформить заказ</a>

<?php elseif ($user\_role === 'admin'): ?>

<form method="POST">

<input type="hidden" name="action" value="delete">

<input type="hidden" name="build\_id" value="<?= $build['id'] ?>">

<button type="submit">Удалить</button>

</form>

<?php endif; ?>

</div>

<?php endforeach; ?>

</body>

</html>

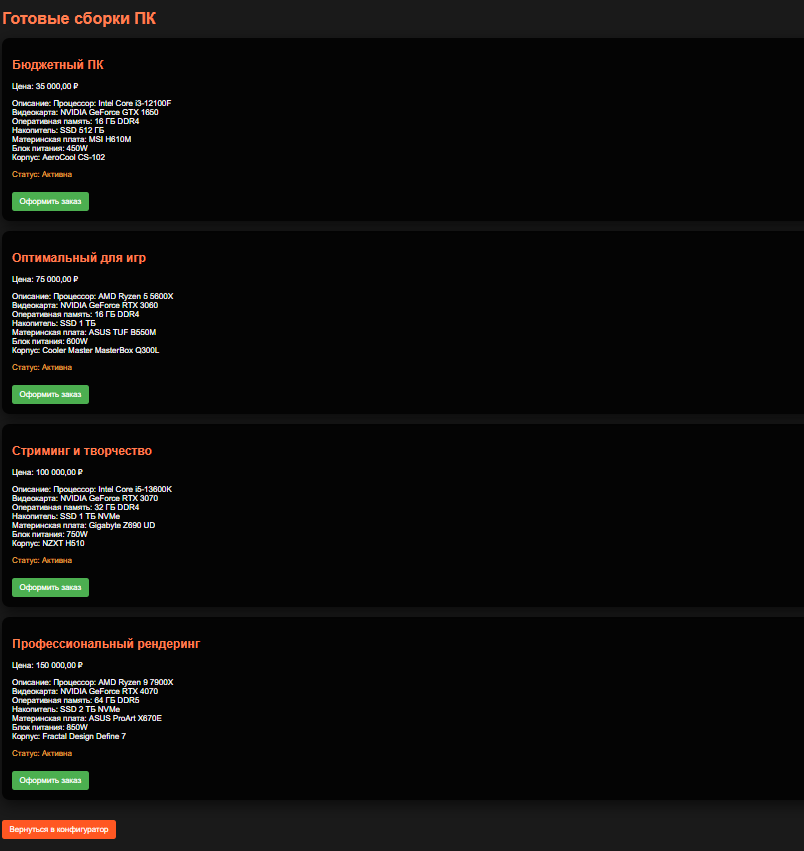


Рисунок 9 — Страница “Готовые сборки ПК” для обычного пользователя

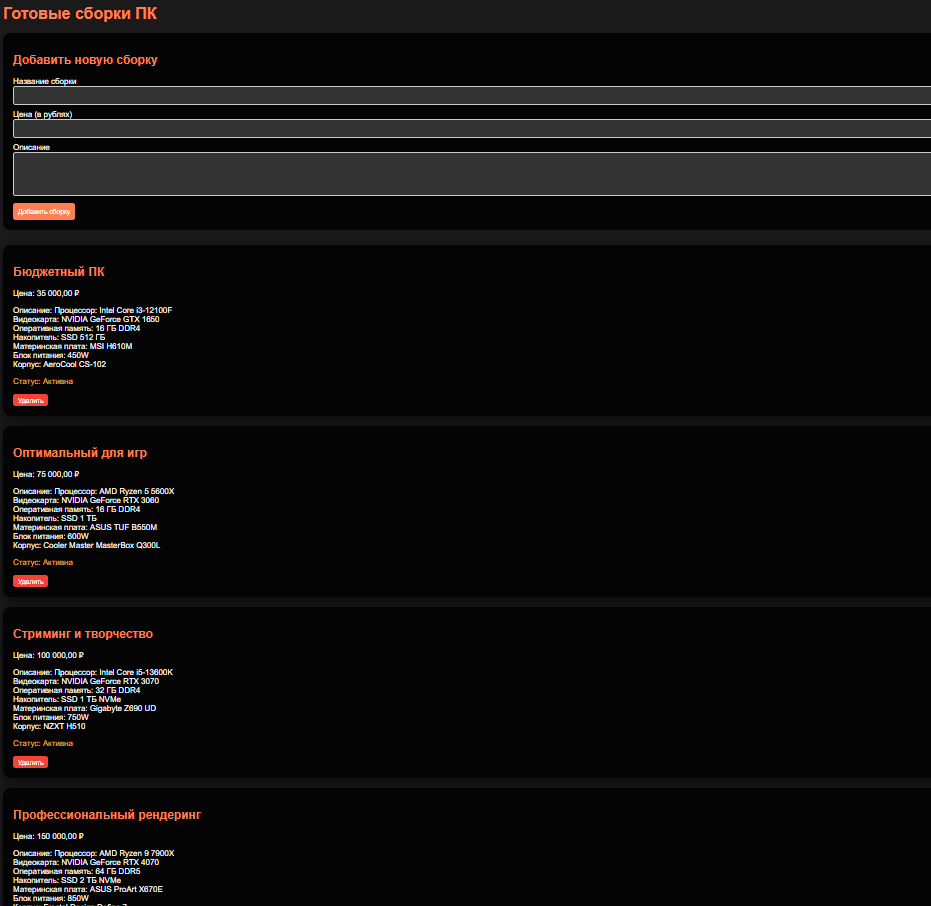


Рисунок 10 — Страница “Готовые сборки ПК” для администратора

## **База данных для сайта**

Оба кода представляют собой SQL-скрипты, которые работают с двумя таблицами в базе данных для управления пользователями и популярными сборками ПК.

1. Таблица users (Пользователи)

Первая часть скрипта создает таблицу для хранения информации о пользователях, которая включает:

id: Уникальный идентификатор пользователя, автоинкрементируемое поле.

username: Имя пользователя (строка до 50 символов).

password: Пароль пользователя (хранится в зашифрованном виде).

role: Роль пользователя — может быть admin (администратор) или user (обычный пользователь). По умолчанию — user.

created\_at: Время создания пользователя.

После создания таблицы в нее добавляются три записи:

Пользователь с ролью администратора (admin) с зашифрованным паролем.

Два обычных пользователя (user), также с зашифрованными паролями.

Затем задается первичный ключ по полю id, и поле id делается автоинкрементируемым, что позволяет автоматически увеличивать значение этого поля при добавлении новых пользователей.

2. Таблица popular\_builds (Популярные сборки ПК)

Вторая часть скрипта создает таблицу для хранения информации о популярных сборках ПК:

id: Уникальный идентификатор для каждой сборки (автоинкремент).

name: Название сборки (до 100 символов).

price: Цена сборки с точностью до двух знаков после запятой.

description: Описание сборки (например, ее характеристики).

is\_active: Статус сборки (1 — активна, 0 — неактивна), по умолчанию — 1 (активна).

Данные для четырех популярных сборок (игровая, рабочая станция, бюджетная и для программирования) вставляются в таблицу с ценами и описаниями каждой сборки.

Как это работает в системе:

Таблица users управляет данными о пользователях, их ролями и правами доступа. Пользователи могут быть как администраторами (с правом добавлять или удалять сборки), так и обычными пользователями (с доступом только к просмотру сборок).

Таблица popular\_builds хранит информацию о сборках ПК, которые доступны для покупки или заказа. Администраторы могут добавлять, редактировать или удалять сборки, а обычные пользователи могут их просматривать и оформлять заказы.

Связь между этими таблицами:

Администраторы могут управлять сборками (добавлять, изменять или удалять).

Обычные пользователи могут просматривать список активных сборок и оформлять заказы.

Эти две таблицы взаимодействуют друг с другом, управляя пользователями и их правами, а также обеспечивают возможность добавления, редактирования и отображения популярных сборок ПК на веб-сайте или в приложении. (рисунок 11)

Код базы данных:

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

CREATE TABLE `users` (

`id` int(11) NOT NULL,

`username` varchar(50) NOT NULL,

`password` varchar(255) NOT NULL,

`role` enum('admin','user') DEFAULT 'user',

`created\_at` timestamp NOT NULL DEFAULT current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

INSERT INTO `users` (`id`, `username`, `password`, `role`, `created\_at`) VALUES

(1, '1', '$2y$10$OKEsSUhtxgVwmMTw4R8LFONOFgOMJ.gPQpZYtzE1XM9j6ECCwY/2y', 'admin', '2024-12-08 10:56:13'),

(3, '2', '$2y$10$WD.lqj6rxo6z5tL9nzNt/.XS2ACX.lZ9iEfv1PLLM6xxjVnSdVKtO', 'user', '2024-12-08 10:58:23');

ALTER TABLE `users`

ADD PRIMARY KEY (`id`);

ALTER TABLE `users`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

COMMIT;

CREATE TABLE `popular\_builds` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(100) NOT NULL,

`price` DECIMAL(10,2) NOT NULL,

`description` TEXT NOT NULL,

`is\_active` TINYINT(1) DEFAULT 1,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

-- Вставка данных для сборок

INSERT INTO `popular\_builds` (`name`, `price`, `description`, `is\_active`) VALUES

('Сборка для игр 2024', 45000.00, 'Мощная игровая сборка для современных игр.', 1),

('Рабочая станция', 60000.00, 'Компьютер для работы с графикой и видео.', 1),

('Бюджетная сборка', 25000.00, 'Доступная по цене сборка для повседневных задач.', 1),

('Сборка для программирования', 40000.00, 'Компьютер для разработки программного обеспечения.', 1);

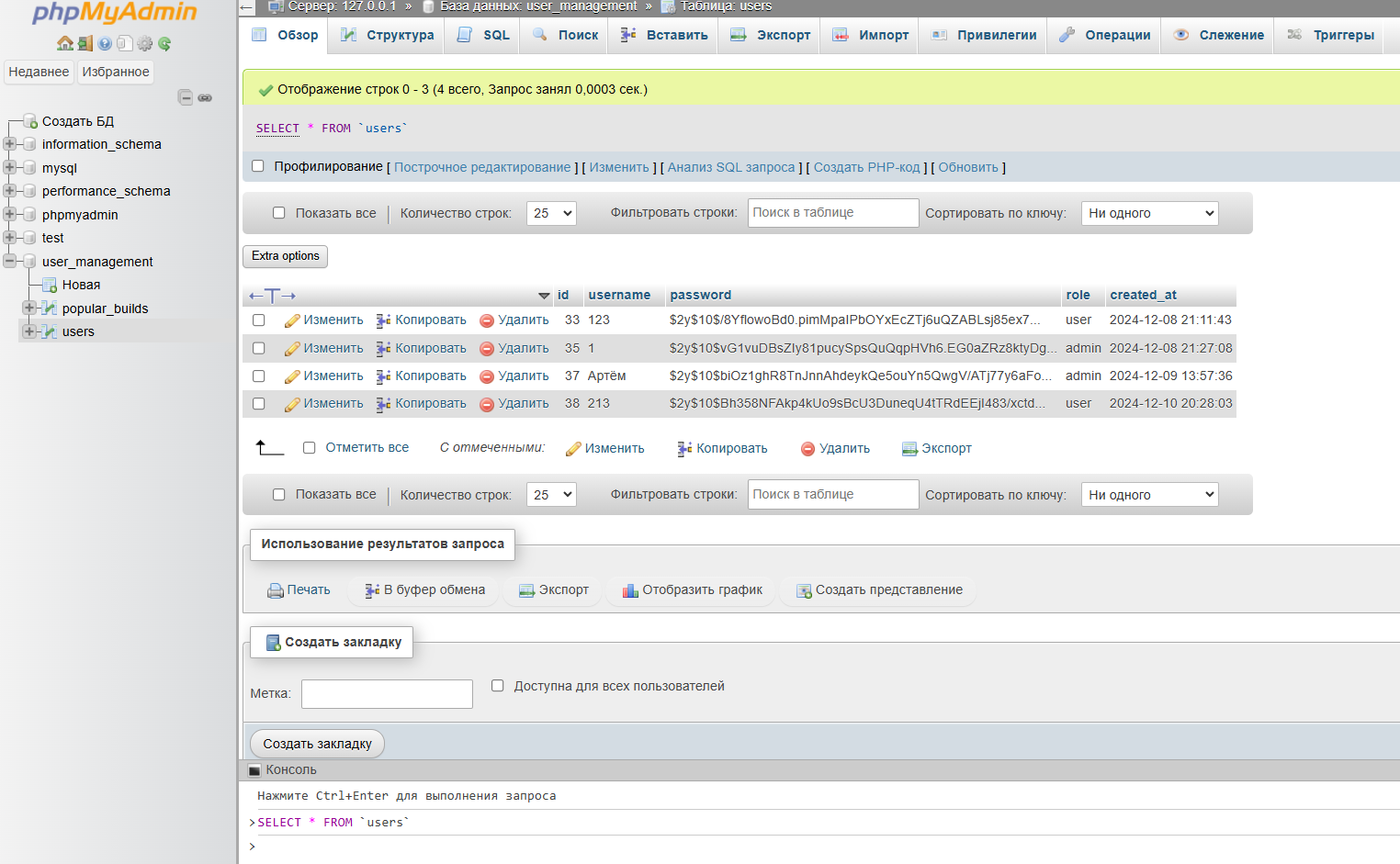


Рисунок 11 — Отображение базы данных в сервисе в phpMyAdmin

# **Заключение**

В условиях быстрого развития информационных технологий и повсеместной цифровизации веб-приложения становятся важным инструментом для решения различных задач в самых разных областях. Особенно актуальными становятся сервисы, которые позволяют пользователям решать сложные задачи, такие как подбор комплектующих для компьютеров, прямо из дома, что имеет большой потенциал в сфере электроники.

Целью данной курсовой работы было создание веб-приложения для конфигурирования персональных компьютеров, которое позволяет пользователям подбирать комплектующие с учетом их характеристик, совместимости, стоимости и производительности. В процессе работы было разработано приложение, обеспечивающее интуитивно понятный интерфейс и эффективное взаимодействие с базой данных.

Приложение реализовано с использованием современных веб-технологий, таких как HTML, CSS, JavaScript, PHP и MySQL. Эти технологии обеспечивают надежную работу, гибкость и масштабируемость проекта. Клиентская часть веб-приложения была разработана с использованием HTML и CSS для создания структуры и оформления интерфейса, а JavaScript использовался для реализации интерактивных функций, таких как фильтрация комплектующих и проверка их совместимости. Серверная часть приложения написана на языке PHP, а для хранения данных о комплектующих и пользовательских конфигурациях использована реляционная база данных MySQL.

В процессе разработки были решены важные задачи, такие как проектирование архитектуры приложения, создание базы данных, тестирование функционала и обеспечение безопасности, включая защиту от SQL-инъекций и других уязвимостей. Также уделено внимание созданию удобного и безопасного интерфейса, который обеспечивает эффективное взаимодействие с пользователем.

Реализованное приложение представляет собой удобный инструмент для выбора и конфигурирования персональных компьютеров, что является важным шагом в развитии современных веб-сервисов в сфере электроники. Дальнейшее развитие проекта может включать расширение функционала, добавление новых категорий комплектующих, улучшение алгоритмов совместимости и персонализированных рекомендаций.

Таким образом, выполненная работа подтверждает эффективность использования современных веб-технологий для создания полезных и востребованных приложений. Разработка конфигуратора компьютеров является практичным и полезным примером применения веб-технологий в реальной сфере, который способствует упрощению выбора и сборки персональных компьютеров для пользователей разных уровней подготовки.

# **Список использованных источников**

1. Основы веб-разработки: HTML, CSS и JavaScript // URL: <https://developer.mozilla.org/ru/docs/Web/HTML> (дата обращения 05.11.2024)
2. PHP и MySQL для начинающих // URL: https://www.php.net/manual/ru/ (дата обращения 08.11.2024)
3. Введение в MySQL // URL: <https://dev.mysql.com/doc/refman/8.0/ru/> (дата обращения 12.11.2024)
4. JavaScript: Современное руководство // URL: <https://developer.mozilla.org/ru/docs/Web/JavaScript> (дата обращения 15.11.2024)
5. Основы веб-разработки с использованием PHP и MySQL // URL: https://www.tutorialrepublic.com/php-tutorial/ (дата обращения 20.11.2024)
6. Технологии создания веб-приложений // URL: https://habr.com/ru/post/355142/ (дата обращения 25.11.2024)
7. Руководство по безопасности веб-приложений // URL: https://owasp.org/www-project-top-ten/ (дата обращения 05.12.2024)
8. Лучшие практики проектирования пользовательского интерфейса // URL: https://www.smashingmagazine.com/guidelines-for-ui-design/ (дата обращения 10.12.2024)