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 «Книги»

Преподаватель

подпись, дата

Никитин А.И.

Студент УБ22-08Б, 432210109

подпись, дата

Красноярск 2023

Поляков Н.А.

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc155965500)

[1 Теоретическая часть 4](#_Toc155965501)

[1.1 HTML И CSS 4](#_Toc155965502)

[1.2 JavaScript 5](#_Toc155965503)

[1.3 PHP 7](#_Toc155965504)

[1.4 MySQL 9](#_Toc155965505)

[2 Практическая часть 12](#_Toc155965506)

[2.1 Требования к работе 12](#_Toc155965507)

[2.2 Создание базы данных 13](#_Toc155965508)

[2.3 Регистрация и авторизация 15](#_Toc155965509)

[2.4 Главная страница 20](#_Toc155965510)

[2.5 Личный кабинет 28](#_Toc155965511)

[Заключение 36](#_Toc155965512)

[Список использованных источников 37](#_Toc155965513)

# Введение

В нашем современном информационном обществе веб-приложения стали неотъемлемой частью повседневной жизни, предоставляя удобные и эффективные инструменты для взаимодействия с информацией и выполнения различных задач. Создание веб-приложений становится все более актуальным в контексте быстрого развития технологий интернета и растущей потребности в инновационных решениях.

Целью данной курсовой работы является изучение и реализация процесса создания веб-приложения с использованием современных веб-технологий и базы данных для управления пользователями и внутренними услугами. В ходе работы рассматриваются основные этапы разработки веб-приложения, а также обзор ключевых инструментов и технологий, таких как HTML, CSS, JavaScript, PHP и MySQL.

Акцент делается на понимании архитектурных принципов веб-приложений, аспектах первичной безопасности и удобства использования для конечного пользователя.

Работа направлена на формирование навыков разработки веб-приложений, анализа требований пользователей и применения передовых подходов к созданию современных веб-приложений. В конечном итоге ожидается получение знаний о процессе создания веб-приложений, а также навыков, необходимых для успешной реализации подобных проектов в практической сфере.

Объект исследования: Объектом исследования является само веб-приложение.

Предмет исследования: Предметом исследования является процесс создания веб-приложения и базы данных.

# 1 Теоретическая часть

# 1.1 HTML И CSS

Веб-разработка в значительной степени опирается на два фундаментальных инструмента: HTML (HyperText Markup Language) и CSS (Cascading Style Sheets). Эти технологии служат основой для создания структуры и визуального оформления веб-страниц, обеспечивая браузерам и пользователям четкое и красочное отображение контента.

HTML является языком разметки, предназначенным для структурирования информации на веб-странице. С его помощью создаются различные элементы, такие как заголовки, параграфы, списки, изображения и гиперссылки. HTML предоставляет универсальные теги, каждый из которых определяет определенный тип контента. Например, тег <h1> используется для создания заголовков первого уровня, а тег <p> - для параграфов [2].

CSS, с другой стороны, предоставляет инструменты для стилевого оформления веб-страниц и придания им визуальной привлекательности. Он управляет цветами, шрифтами, отступами, расположением элементов и другими аспектами внешнего вида. CSS также поддерживает концепцию каскадности, что позволяет эффективно управлять стилями и создавать последовательность приоритетов.

Сочетание HTML и CSS позволяет веб-разработчикам создавать динамичные и креативные веб-страницы. Применение семантической разметки с использованием HTML способствует лучшему восприятию контента поисковыми системами, а гибкость CSS дает возможность адаптировать внешний вид страницы под различные устройства и разрешения экранов. Эти инструменты являются основополагающими для веб-разработчика, открывая двери к созданию привлекательных, удобных в использовании и технологически современных веб-приложений.

В современной веб-разработке HTML и CSS также играют ключевую роль в обеспечении доступности и адаптивности веб-приложений. Стандарты разметки HTML5 вводят новые элементы и атрибуты, улучшая структурирование контента и обеспечивая большую гибкость для веб-разработчиков. Новые возможности, такие как мультимедийные элементы <audio> и <video>, а также API для работы с локальным хранилищем, значительно расширяют функционал веб-приложений.

CSS3, в свою очередь, вводит множество новых свойств и селекторов, что позволяет создавать более сложные и интерактивные дизайны. Технологии, такие как Flexbox и Grid Layout, предоставляют разработчикам мощные инструменты для управления расположением элементов на веб-странице [1].

Комбинированный эффект HTML и CSS также проявляется в области анимаций и переходов. CSS предоставляет возможности для создания плавных и эффектных анимаций без необходимости использования дополнительных библиотек или языков программирования.

# 1.2 JavaScript

Третьей важной составляющей для создания веб-приложений является язык программирования JavaScript. Он является клиентским языком сценариев, который выполняется непосредственно в браузере пользователя. Этот язык предоставляет разработчикам мощные инструменты для создания интерактивных и динамичных веб-приложений.

JavaScript был создан для добавления возможности создания динамичных элементов на стороне клиента, что позволяет веб-страницам реагировать на действия пользователя без необходимости обновления всей страницы. Этот язык поддерживает событийно-ориентированное программирование, что означает, что код может реагировать на различные события, такие как нажатия клавиш, клики мыши и изменения веб-элементов.

Одной из ключевых особенностей JavaScript является его способность взаимодействовать с DOM (Document Object Model), представляя собой иерархическое древовидное представление HTML-документа. Это позволяет динамически изменять содержимое страницы, добавлять и удалять элементы, а также изменять их стили.

Современные веб-приложения часто используют JavaScript вместе с библиотеками и фреймворками, такими как React, Angular или Vue.js, чтобы упростить процесс разработки и обеспечить более эффективное управление компонентами приложения. Эти инструменты обеспечивают структурированный и эффективный подход к созданию сложных пользовательских интерфейсов и взаимодействия веб-приложений. JavaScript также включает возможности работы с сервером при помощи технологии AJAX, что позволяет асинхронно обмениваться данными между клиентом и сервером без перезагрузки страницы.

Можно выделить основные преимущества и недостатки данной технологии.

Плюсы JavaScript:

– Интерактивность и Динамичность: JavaScript обеспечивает создание динамичных интерфейсов, где веб-страницы реагируют на действия пользователя без необходимости перезагрузки.

– Всеобъемлющий Клиентский Язык: JavaScript выполняется в браузере пользователя, что упрощает создание клиентской логики, работающей на стороне клиента, и обеспечивает отзывчивость веб-приложений.

– Широкая Поддержка Браузеров: язык имеет высокую степень совместимости с различными браузерами, что позволяет разработчикам создавать кросс-браузерные приложения.

– Множество Библиотек и Фреймворков: существуют множество библиотек и фреймворков, таких как React, Angular, и Vue.js, которые упрощают и ускоряют разработку веб-приложений.

– Современные средства отладки: разработчики имеют доступ к мощным инструментам отладки в интегрированных средах разработки (IDE) и консолях браузеров.

Минусы JavaScript:

– Ограниченные возможности без поддержки браузера: некоторые старые браузеры или браузеры с ограниченной поддержкой могут не корректно отображать или исполнять JavaScript, что может создать проблемы совместимости.

– Безопасность: из-за выполнения на стороне клиента, JavaScript подвержен угрозам безопасности, таким как внедрение межсайтовых сценариев (XSS). Разработчики должны соблюдать меры безопасности для защиты от таких атак.

– Зависимость от Интернета: некоторые функции JavaScript могут зависеть от доступности интернета, что может ограничивать функциональность в офлайн-режиме.

– Использование ресурсов: некорректное использование JavaScript может привести к избыточному использованию ресурсов браузера, что сказывается на производительности.

– Ограниченные возможности доступа к ОС: JavaScript ограничен в своей способности взаимодействовать с операционной системой пользователя из соображений безопасности.

JavaScript является мощным инструментом для создания интерактивных веб-приложений, обеспечивая разработчикам широкие возможности. Однако, для достижения лучших результатов, необходимо соблюдать современные стандарты разработки и учитывать ограничения, связанные с безопасностью и производительностью [3].

# 1.3 PHP

PHP (Hypertext Preprocessor) представляет собой мощный язык программирования, специально предназначенный для веб-разработки. Этот серверный язык широко используется для создания динамических веб-приложений, взаимодействия с базами данных и обработки данных на стороне сервера.

Основной Функционал PHP:

– Интеграция с HTML: PHP позволяет встраивать свой код непосредственно в HTML-документы. Это делает возможным создание динамических веб-страниц, где содержимое генерируется на сервере в зависимости от условий и данных.

– Работа с базами данных: язык обеспечивает удобные средства взаимодействия с различными системами управления базами данных (например, MySQL, PostgreSQL). Это позволяет создавать приложения, эффективно хранящие и обрабатывающие данные.

– Обработка форм и данных: PHP является отличным инструментом для обработки данных, отправляемых с веб-форм. Это важно для создания интерактивных форм и взаимодействия с пользователями.

– Создание сессий и куки: язык обеспечивает механизмы для создания сессий, что позволяет хранить информацию о состоянии пользователя между запросами. Куки используются для хранения информации на стороне клиента.

– Работа с файлами: PHP предоставляет средства для работы с файловой системой сервера, что полезно для загрузки, скачивания и обработки файлов.

Актуальность использования PHP в веб-приложениях:

– Широкое распространение: язык широко поддерживается хостинг-провайдерами и работает на большинстве серверов. Это делает его доступным для разработчиков различных уровней.

– Сообщество и ресурсы: PHP обладает активным сообществом разработчиков, множеством библиотек и фреймворков, что упрощает процесс разработки и поддержки проектов.

– Простота и гибкость: язык предоставляет простой синтаксис, что делает его доступным для новичков, но при этом обладает достаточной гибкостью для удовлетворения потребностей опытных разработчиков.

– Обширный инструментарий: с PHP легко интегрировать другие технологии и инструменты, что делает его универсальным решением для создания различных веб-приложений.

Минусы PHP:

– Глобальная интерпретация: PHP интерпретируется на каждом запросе, что может сказаться на производительности по сравнению с некоторыми языками, такими как Java.

– Отсутствие строгой типизации: отсутствие строгой типизации может привести к непредсказуемым результатам в определенных сценариях, требуя внимательного контроля типов данных.

– Ограниченные возможности асинхронности: в сравнении с некоторыми современными языками, PHP имеет ограниченные возможности для эффективной работы в асинхронном режиме.

– Менее подходит для сложных интерфейсов: в сравнении с фронтенд-ориентированными языками, такими как JavaScript, PHP может быть менее эффективным для создания сложных интерфейсов.

– Большая история и изменение языка: некоторые старые черты PHP могут считаться устаревшими, и изменения в языке могут потребовать внимательного обновления существующего кода [5].

PHP остается одним из наиболее популярных языков для серверной веб-разработки благодаря своей простоте использования и мощным возможностям. Однако разработчики должны быть в курсе его ограничений и избегать старых практик в пользу более современных и безопасных методов.

# 1.4 MySQL

MySQL является одной из самых популярных открытых реляционных систем управления базами данных (СУБД). Рассмотрим, что представляет собой MySQL его функционал, основные плюсы и минусы.

Основы MySQL:

– Реляционная база данных: MySQL предоставляет мощный механизм для хранения и управления данными в виде таблиц с отношениями между ними. Это обеспечивает структурированное и эффективное хранение информации.

– Язык SQL: взаимодействие с MySQL осуществляется с использованием структурированного языка запросов SQL. Он позволяет выполнять операции чтения, записи, обновления и удаления данных, а также управлять структурой базы данных.

– Поддержка транзакций: MySQL обеспечивает поддержку транзакций, что гарантирует целостность данных при выполнении серии операций. Это особенно важно в приложениях, где важна надежность хранения данных.

– Масштабируемость и производительность: MySQL спроектирован с учетом возможности масштабирования и обеспечивает высокую производительность при работе с большим объемом данных.

– Открытость и бесплатность: MySQL распространяется под лицензией GPL и является бесплатным для использования.

– Большое сообщество и поддержка: MySQL обладает обширным сообществом разработчиков, что обеспечивает наличие множества ресурсов и поддержки.

– Сообщество и документация: огромное сообщество пользователей MySQL активно обсуждает проблемы, предоставляет решения и дополнительные материалы в интернете. Официальная документация MySQL также предоставляет обширные сведения по настройке, использованию и оптимизации.

– Широкий спектр интеграции: MySQL легко интегрируется с различными языками программирования и фреймворками, такими как PHP, Python, Ruby, Node.js, что делает его универсальным выбором для многих приложений.

– Инструменты управления: для удобства работы с MySQL существует множество инструментов управления базой данных, таких как MySQL Workbench, phpMyAdmin и другие, предоставляющие графический интерфейс для выполнения различных задач.

– Безопасность и аутентификация: MySQL обеспечивает механизмы аутентификации и авторизации пользователей, а также поддерживает шифрование данных, что делает его безопасным выбором для хранения чувствительной информации.

К минусам можно отнести:

– Ограниченные возможности для некоторых видов запросов: для некоторых специфических видов запросов MySQL может быть менее подходящим в сравнении с другими СУБД.

– Нет встроенной полноценной поддержки JSON: в некоторых случаях может возникнуть необходимость в более продвинутой обработке JSON-данных [4].

MySQL остается востребованным инструментом для разработчиков и системных архитекторов, предоставляя надежное и эффективное решение для управления большим объемом данных в различных веб-приложениях.

# 2 Практическая часть

# 2.1 Требования к работе

Необходимо разработать web-приложения с применением современных web-технологий по тематике «Книги», используя технологии HTML, CSS, JavaScript, PHP и MySQL. Кроме того для своего проекта необходимо разработать базу данных для управления пользователями. Реализовать добавление, редактирование, просмотр и удаление пользователей. Также предусмотреть функции авторизации и аутентификации.

Описание реализации:

– неавторизированные пользователи должны иметь возможность регистрироваться. (форма добавления пользователя)

– авторизированные пользователи разделить на 2 группы: администраторы и авторизированные пользователи.

– администраторы имеют право просматривать, добавлять, редактировать и удалять пользователей.

– авторизированные пользователи имеют право просматривать и редактировать только свои данные.

– при регистрации пользователю присваивается статус авторизированного пользователя.

Требования к проекту:

– должна быть предусмотрена авторизация и регистрация пользователей;

– зарегистрированный пользователь имеет право редактировать свои данные и просматривать свои услуги;

– администраторы имеют право редактировать пользователей и добавлять услуги;

– неавторизированный пользователи имеют право просматривать список услуг;

– сайт должен быть выдержан в определенном стиле и корректно открываться на любом разрешении;

– общий функционал может быть продуман самостоятельно;

– важно продемонстрировать не большой объем функционала, а минимальные требования к проекту – он может быть максимально простым, но при этом содержать все необходимые функции;

– отдельные бонусы предусмотрены за доп. функционал и красивый дизайн.

В качестве инструментов в работе будет использован текстовый редактор VSCode, приложение для создания локального сервера MAMP, в котором также можно найти веб-приложение phpMyAdmin для администрирования СУБД MySQL.

# 2.2 Создание базы данных

Перед началом работы необходимо запустить приложение MAMP и подождать, пока автоматически не включатся необходимые серверы: apache Server и MySQL Server (рис.1).

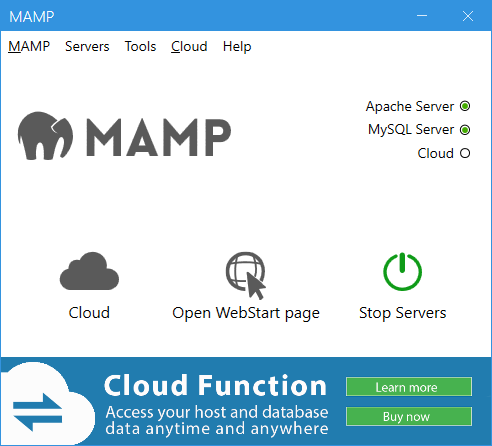


Рисунок 1 – запуск серверов через MAMP

Далее следует открыть веб-приложение phpMyAdmin (рис.2), в котором можно создать базу данных через вспомогательные кнопки или напрямую через консоль.

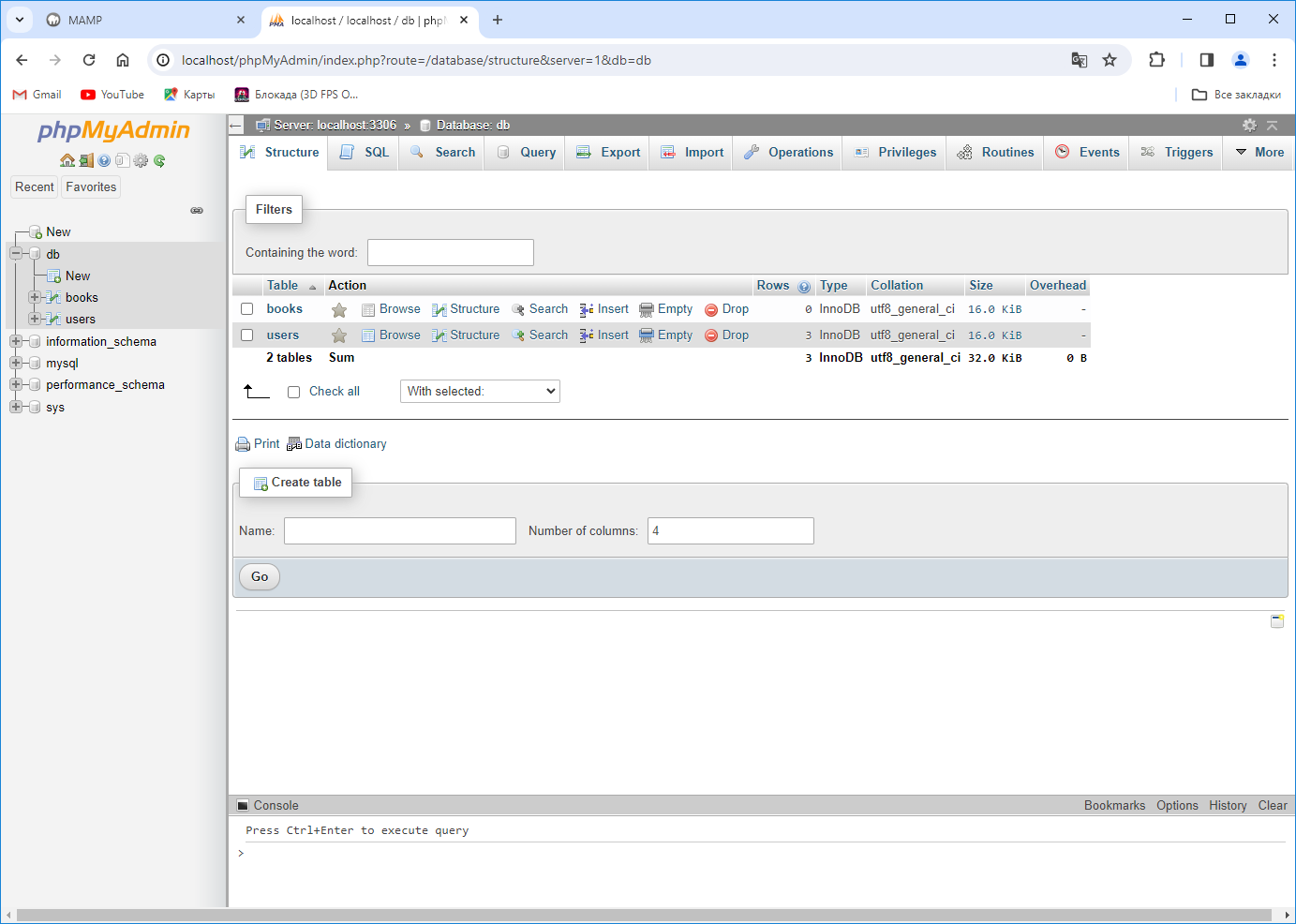


Рисунок 2 – приложение phpMyAdmin

Можно выбрать второй вариант, тогда необходимо прописать две команды:

* CREATE DATABASE IF NOT EXISTS db;

USE db;

CREATE TABLE IF NOT EXISTS users (

id INT(10) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL,

role ENUM('admin', 'user') NOT NULL DEFAULT 'user',

tel VARCHAR(255) NOT NULL

);

– CREATE TABLE IF NOT EXISTS books (

id INT(11) AUTO\_INCREMENT PRIMARY KEY,

book VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

imagepath VARCHAR(255) NOT NULL,

description VARCHAR(255) NOT NULL,

price VARCHAR(255) NOT NULL

);

Первая команда создает базу данных с названием db, если она уже не создана, а также таблицу с данными, где указано имя пользователя, пароль, роль (администратор или обычный пользователь) и номер телефона. Вторая команда создает таблицу для хранения информации о книгах (название, автор, путь к изображению, описание, цена). Вся база данных основана на кодировке utf8\_general\_ci.

После создания базы данных следует проверить корректность ее подключения. Для этого можно использовать серверный язык php. В текстовом редакторе достаточно ввести следующий код:

<?php

$user = 'root';

$password = 'root';

$db = 'db';

$host = 'localhost';

$port = 3306;

$dbreg = new mysqli($host, $user, $password, $db, $port);

if ($dbreg->connect\_error)

{

    exit("Ошибка соединения: " . $dbreg->connect\_error);

}

?>

Теперь, при ошибке, связанной с подключением базы данных, выведется соответствующее сообщение.

# 2.3 Регистрация и авторизация

Для получения форм регистрации и авторизации был создан HTML-документ openFirst.html:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <link rel="stylesheet" href="normalize.css">

    <link rel="stylesheet" href="styles.css">

    <title>Регистрация/Авторизация пользователя</title>

</head>

<body>

    <div class="container">

        <h2>Авторизация</h2>

        <form action="myLogIn.php" method="post">

            <label for="loginUsername">Имя пользователя:</label>

            <input type="text" id="loginUsername" name="loginUsername" required>

            <label for="loginPassword">Пароль:</label>

            <input type="password" id="loginPassword" name="loginPassword" required>

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

        </form>

        <h2>Регистрация</h2>

        <form action="registration.php" method="post" onsubmit="return normPassword()">

            <label for="tel">Контактный телефон:</label>

            <input type="text" id="tel" name="tel" required>

            <label for="username">Имя пользователя:</label>

            <input type="text" id="username" name="username" required>

            <label for="password">Пароль:</label>

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

            <button type="submit" onclick="normPassword()">Зарегистрироваться</button>

        </form>

    </div>

    <script>

        function normPassword() {

            var password = document.getElementById('password').value;

            if (password.length < 5) {

                alert("Пароль должен содержать не менее 5 символов");

                return false;

            }

            var phone= document.getElementById('tel').value;

            var str = /^8\d{10}$/;

            if(!str.test(phone)){alert("Введите корректный телефон в формате: 88005553535");return false;}

            return true;

        }

    </script>

</body>

</html>

В самом теле документа используется тег <form>. Именно он позволяет создать форму и ссылаться на определенный файл после нажатия кнопки в этой форме посредством атрибута action. Кроме того используется тег <script> для связи HTML-документа с языком JavaScript. Функция normPassword() регулирует длину установленного пароля (не менее 5 символов), а также корректность указанного при регистрации телефона. В заголовке используется тег <link> для связи со стилями normalize.css и styles.css. Первый отвечает за стандартизацию всего документа, чтобы в разных браузерах открывшийся файл выглядел идентично. Второй основан на личных предпочтениях в отображении веб-приложения:

body {

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    background-color: #f0f0f0;

}

.container {

    max-width: 500px;

    margin: 50px auto;

    padding: 30px;

    background-color: #fff;

    border-radius: 8px;

    box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);

}

form {

    margin-top: 20px;

}

input {

    width: calc(100% - 16px);

    padding: 10px;

    margin-bottom: 15px;

    border: 1px solid #ccc;

    border-radius: 5px;

}

button {

    background-color: #3498db;

    color: #fff;

    padding: 12px;

    border: none;

    border-radius: 3px;

    cursor: pointer;

}

.admin-panel {

    margin-top: 30px;

    border: 2px solid #ddd;

    padding: 20px;

    text-align: center;

}

.admin-panel form {

    margin-top: 15px;

}

.admin-panel ul {

    list-style-type: none;

    padding: 0;

}

.admin-panel li {

    margin-bottom: 10px;

}

.admin-panel button {

    cursor: pointer;

}

table {

    border-collapse: collapse;

    width: 400px;

}

td {

    padding: 10px;

}

tr:nth-child(even) {

    background-color: #f2f2f2;

}

.flex{

    display: flex;

    align-items: center;

    justify-content: center;

}

Теперь стоит обратить внимание на два документа, к которым ссылается форма регистрации и форма авторизации: registration.php и myLogIn.php.

Первый документ позволяет зарегистрировать нового пользователя:

<?php

require('dataBase.php');

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

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

{

    $select = $dbreg->query("SELECT id FROM users WHERE username='$username'");

    if ($select->num\_rows > 0)

    {

        echo "Такой пользователь уже существует!";

    }

    else

    {

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

        $phone=$\_POST['tel'];

        if ($dbreg->query("INSERT INTO users (username, password, role, tel) VALUES ('$username', '$password', 'user', $phone)"))

        {

            header("Location: openFirst.html");

        }

        else

        {

            echo "Ошибка: " . $dbreg->error;

        }

    }

}

$dbreg->close();

?>

Сначала проверяется введенное имя пользователя на тот случай, если оно уже занято в базе данных. Затем указанный пароль хешируется для большей безопасности и, как все указанные данные, вписывается в таблицу users.

Второй документ позволяет авторизоваться под своим аккаунтом:

<?php

session\_start();

require('dataBase.php');

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

{

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

    $select = $dbreg->query("SELECT \* FROM users WHERE username='$loginUsername'");

    if ($select->num\_rows > 0)

    {

        $array = $select->fetch\_assoc();

        if (password\_verify($\_POST['loginPassword'], $array['password']))

        {

            $\_SESSION['user\_id'] = $array['id'];

            header("Location: books.php");

        }

        else

        {

            echo "Неверный пароль!";

        }

    }

    else

    {

        echo "Ошибка: такого пользователя нет.<br> Но вы всегда можете зарегистрироваться!";

    }

}

$dbreg->close();

?>

При корректном вводе логина и пароля пользователя перенаправит на главную страницу сайта (файл books.php). В результате создания HTML-документа с помощью различных стилей получаются минималистичные, но при этом приятные глазу формы (рис.3).

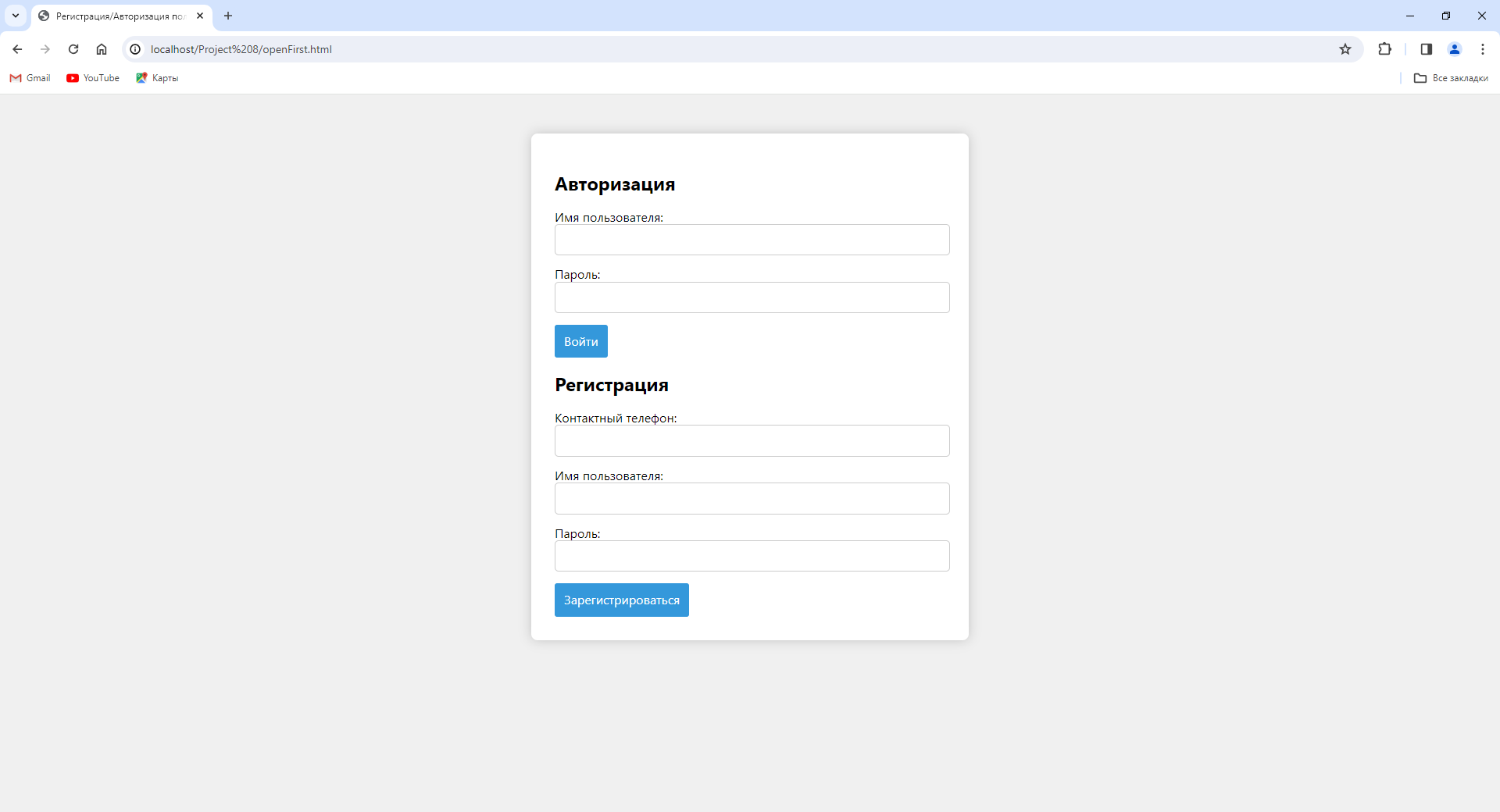


Рисунок 3 – формы регистрации и авторизации

# 2.4 Главная страница

Попасть на главную страницу могут как авторизованные так и неавторизованные пользователи. Однако функционал, предоставляемый сайтом, будет разниться. Сам документ называется books.php и содержит весь основной контент об интернет-магазине BuyBooks (название выбрано произвольно, любые совпадения случайны), используемом в качестве примера для курсовой работы:

<?php

session\_start();

require('dataBase.php');

if (isset($\_SESSION['user\_id']))

{

    $user\_id=$\_SESSION['user\_id'];

    $name=$dbreg->query("SELECT username FROM users WHERE id=$user\_id");

    if ($name->num\_rows > 0) {

        $username = $name->fetch\_assoc()['username'];

    }

    $dbreg->close();

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <link rel="stylesheet" href="normalize.css">

    <link rel="stylesheet" href="forBooks.css">

    <title>Интернет-магазин книг BuyBooks</title>

</head>

<body>

<header>

    <button class="account-button" onclick="account()"><?php $noname="Пользователь"; if (isset($\_SESSION['user\_id'])) {echo $username;} else{echo $noname;} ?></button>

    <button class="cart-button" onclick="showCart()">Корзина<span id="cartCount" class="cart-count">0</span></button>

</header>

<br><br><br>

<span style="text-align:center;color: aqua;"><h1>Главная страница BuyBooks</h1></span>

<div class="book-container">

 <div class="cart-menu" id="cartMenu">

        <div class="cart-content">

            <span style="text-align:center;"><h2>Корзина</h2></span>

            <ul id="cartList"></ul>

            <button class="buy-all-button" onclick="buyAll()">Купить</button>

            <button class="close-button" onclick="hideCart()">Закрыть</button>

            <button class="delete-button" onclick="deleteAll()">Очистить</button>

        </div>

    </div>

    <script src="books.js">

    </script>

    <?php

        require('dataBase.php');

        $book = $dbreg->query("SELECT \* FROM books");

        if ($book->num\_rows>0)

        {

            while ($eachBook = $book->fetch\_assoc())

            {

                echo '<div id="book" class="book"><img src='.$eachBook['imagepath'].'><h2>'.$eachBook['book'].'</h2> <p>Автор:'.$eachBook['author'].'</p> <p>Описание: '.$eachBook['description'].'</p>';

                if (isset($\_SESSION['user\_id']))

                {

                echo '<button class="buy-button" onclick=';echo"'addToCart";echo'("'.$eachBook['book'].'",'. $eachBook['price'] .')';echo"'>Купить</button>";

                echo '<p class="book-price">' . $eachBook['price'] . ' руб.</p>';

                } else {echo "<a href='openFirst.html'>Необходимо зарегистрироваться для покупки!</a>";

                }

                echo "</div>";

            }

        }

        $dbreg->close();

    ?>

</div>

</body>

</html>

В первую очередь идет проверка, является ли пользователь авторизованным. Если это так, тогда в правом верхнем углу будет надпись с именем, выбранном при регистрации, в противном случае будет надпись пользователь. Кроме того все книги, представленные на сайте, можно приобрести только авторизованным пользователям, иначе выведется ссылка на документ openFirst.html с надписью о том, что покупать книги можно только после авторизации.

Для работы с корзиной покупок используется код JavaScript и соответствующий документ books.js:

let cartCount = 0;

let cartItems = [];

function addToCart(bookTitle,cost) {

    cartCount++;

    cartItems.push(bookTitle+" "+cost+" руб.");

    updateCartCount();

}

function updateCartCount() {

    document.getElementById("cartCount").textContent = cartCount;

}

function showCart() {

    document.getElementById("cartList").innerHTML = "";

    cartItems.forEach((item) => {

        const li = document.createElement("li");

        li.textContent = item;

        document.getElementById("cartList").appendChild(li);

    });

    document.getElementById("cartMenu").style.display = "flex";

}

function hideCart() {

    document.getElementById("cartMenu").style.display = "none";

}

function deleteAll() {

    cartCount = 0;

    cartItems = [];

    updateCartCount();

    hideCart();

}

function buyAll() {

    if (cartCount == 0) {

        alert("Сначала выберите книгу(и) для покупки!")

    }

    else { alert("Мы свяжемся с вами по телефону для подтверждения покупки."); }

    cartCount = 0;

    cartItems = [];

    updateCartCount();

    hideCart();

}

function account() {

    var addButton = document.createElement("button");

    addButton.type = "submit";

    var form = document.createElement("form");

    form.method = "post";

    form.action = "page.php";

    form.appendChild(addButton);

    document.body.appendChild(form);

    form.submit();

}

Этот код помогает отобразить изменение счетчика покупок при нажатии на кнопку «купить». Также при нажатии на корзину пользователь увидит все выбранные для покупки книги, их стоимость, а также три кнопки «Купить», «Закрыть» и «Очистить». Первая кнопка проверяет, выбрана ли хотя бы одна книга и отправляет сообщение: «Мы свяжемся с вами по телефону для подтверждения покупки», вторая позволяет скрыть корзину, а третья ­– очистить весь выбор. Кроме того если нажать на кнопку с именем пользователя, произойдет переход на страницу с личным кабинетом, если он авторизован, или переход на страницу openFirst.html для регистрации в противном случае.

Также стоит обратить внимание, что для данной страницы используются другие стили forBooks.css:

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    background: url('https://oir.mobi/uploads/posts/2021-03/thumbs/1616569357\_8-p-obichnii-fon-9.jpg') no-repeat center center fixed;

    background-size: cover;

    color: #333;

}

header {

    text-align: right;

    padding: 5px;

    background-color: #333;

    color: white;

    position: fixed;

    top: 0;

    right: 0;

    left: 0;

    z-index: 1000;

}

.book-container {

    max-width: 800px;

    margin: 20px auto;

    display: grid;

    grid-template-columns: repeat(2, 1fr);

    gap: 20px;

}

.book {

    background-color: #fff;

    padding: 15px;

    border-radius: 8px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    transition: transform 0.3s ease-in-out;

}

.book:hover {

    transform: scale(1.05);

}

.book img {

    max-width: 250px;

    height: auto;

    border-radius: 8px;

    margin-bottom: 10px;

}

h2 {

    color: #333;

    font-size: 1.5em;

    margin-bottom: 5px;

}

p {

    color: #666;

    margin-bottom: 10px;

}

.buy-button {

    background-color: #4caf50;

    color: white;

    border: none;

    padding: 8px 16px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 14px;

    margin-top: 10px;

    cursor: pointer;

    border-radius: 4px;

}

.book-price {

    font-size: 16px;

    color: #007bff;

    margin-top: 10px;

}

.cart-button {

    background-color: #2196F3;

    color: white;

    border: none;

    padding: 8px 16px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 14px;

    margin-top: 10px;

    cursor: pointer;

    border-radius: 4px;

}

.account-button {

    background-color:#333;

    color: white;

    border: none;

    padding: 8px 16px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 14px;

    margin-top: 10px;

    cursor: pointer;

    border-radius: 4px;

}

.cart-count {

    margin-left: 5px;

    font-weight: bold;

}

.cart-menu {

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background: rgba(0, 0, 0, 0.5);

    display: none;

    align-items: center;

    justify-content: center;

}

.cart-content {

    background-color: #fff;

    padding: 20px;

    border-radius: 8px;

    max-width: 400px;

    text-align: left;

}

.close-button {

    background-color: grey;

    color: white;

    border: none;

    padding: 8px 16px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 14px;

    margin-top: 10px;

    cursor: pointer;

    border-radius: 4px;

}

.delete-button {

    background-color: #ff4444;

    color: white;

    border: none;

    padding: 8px 16px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 14px;

    margin-top: 10px;

    cursor: pointer;

    border-radius: 4px;

}

.buy-all-button {

    background-color: #4caf50;

    color: white;

    border: none;

    padding: 8px 16px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 14px;

    margin-top: 10px;

    cursor: pointer;

    border-radius: 4px;

}

Результат отображения сайта представлен на рисунках 4-5.

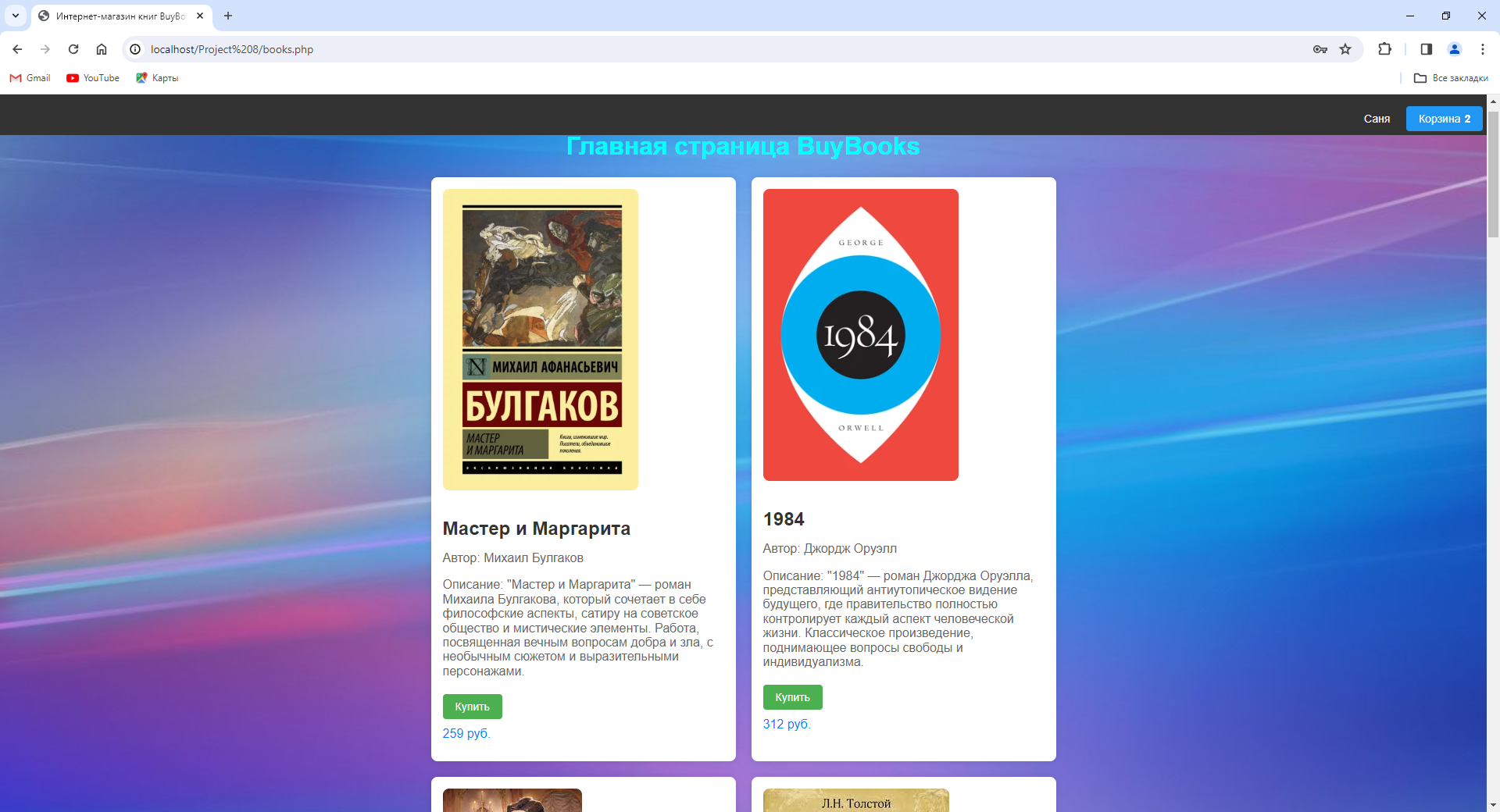


Рисунок 4 – главная страница интернет-магазина

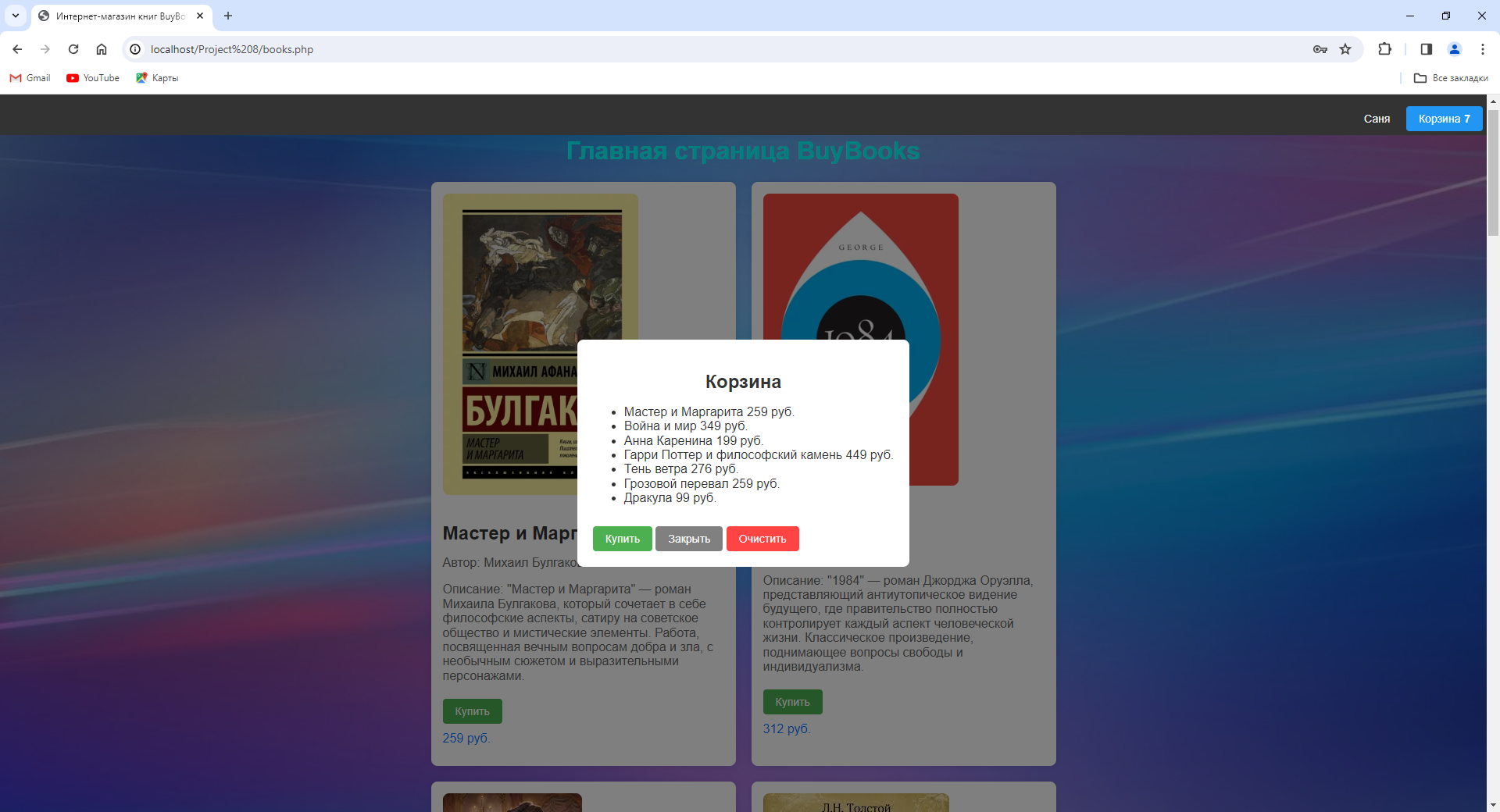


Рисунок 5 – отображение корзины покупок

# 2.5 Личный кабинет

При переходе в личный кабинет (файл page.php) перед пользователем представлена возможность изменить все необходимые данные (телефон, логин, пароль). Также как и для регистрации, здесь используется функция, помогающая определить корректность ввода пароля и телефона. Имеется возможность выхода из аккаунта или полного его удаления. Сама страница представлена на рисунке 6.

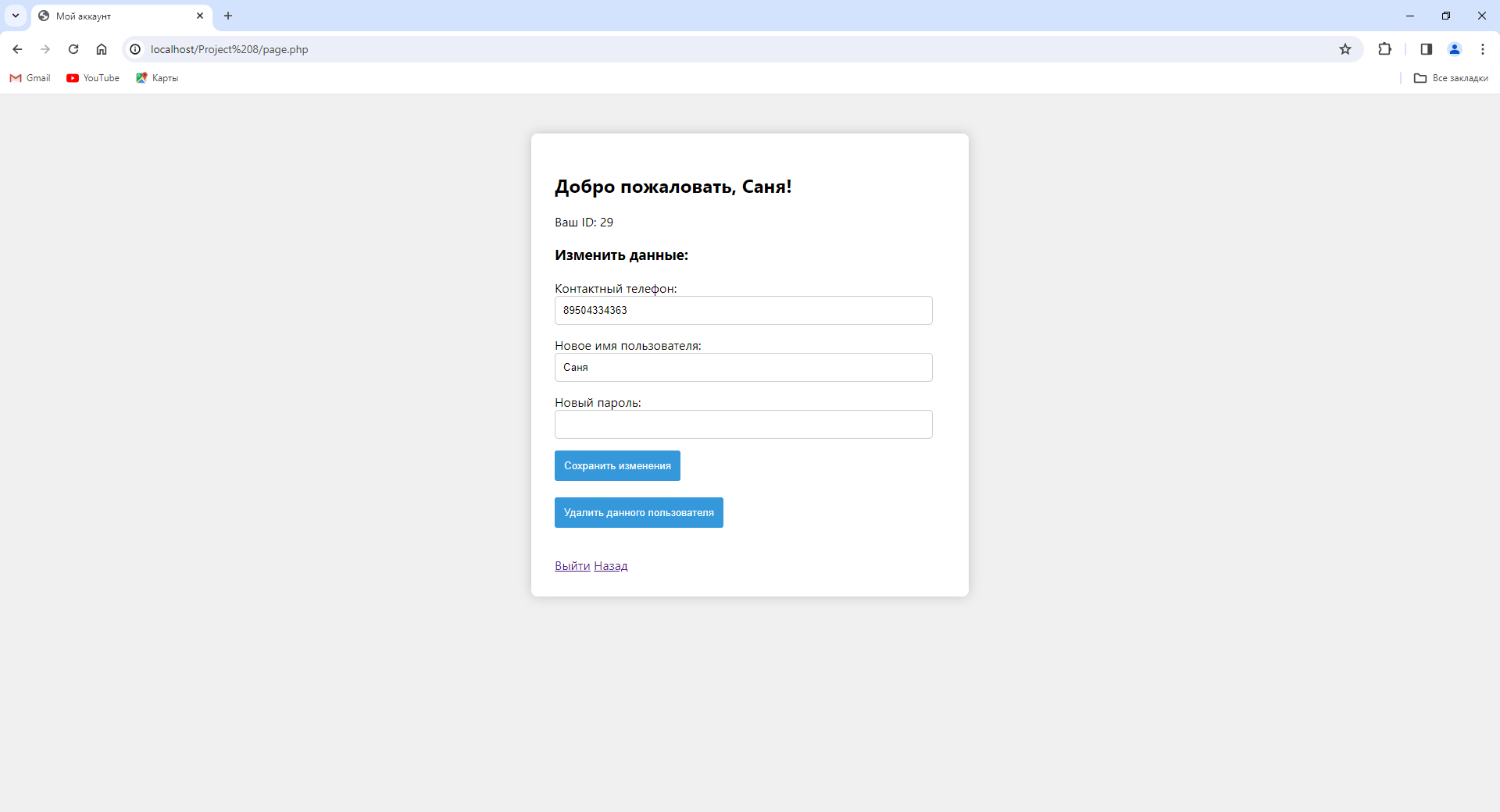


Рисунок 6 – личный кабинет

Однако представленный кабинет относится только к обычным пользователям. Для администраторов имеется дополнительная функция, позволяющая работать с панелью администратора (рис.7).

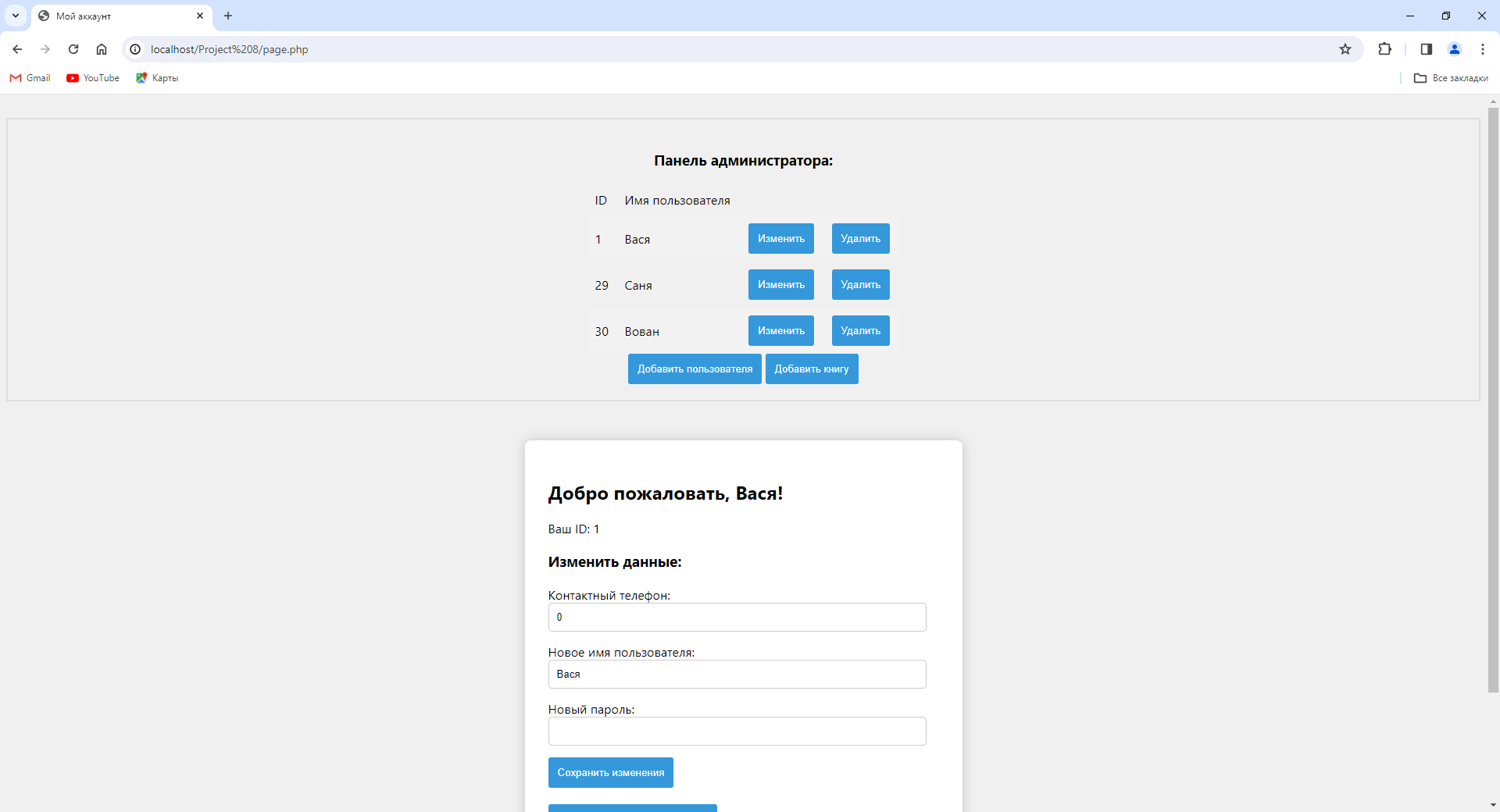


Рисунок 7 – панель администратора

Для объяснения принципа работы всей страницы стоит ознакомиться с кодом:

<?php

session\_start();

require('dataBase.php');

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

{

    header("Location: openFirst.html");

    exit();

}

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

$role = $dbreg->query("SELECT role FROM users WHERE id=$user\_id");

if ($role->num\_rows > 0) {

    $user\_role = $role->fetch\_assoc()['role'];

    if ($user\_role == 'admin')

    {

        $users = $dbreg->query("SELECT id, username FROM users");

        if ($users->num\_rows > 0)

        {

            echo "<div class='admin-panel'>";

            echo "<h3>Панель администратора:</h3>";

            echo "<div class='flex'><table>";

            echo "<tr> <td>ID</td>  <td>Имя пользователя</td></tr>";

            while ($eachUser = $users->fetch\_assoc())

            {

                echo "<tr><td>".$eachUser['id'] . "</td><td>" . $eachUser['username'] . "</td>   <td><button onclick='editUserData(" . $eachUser['id'] . ")'>Изменить</button></td>     <td><button onclick='deleteData(" . $eachUser['id'] . ")'>Удалить</button></td>";

            }

            echo "</table></div>";

            echo "<button onclick='addUser()'>Добавить пользователя</button> ";

            echo " <button onclick='addBook()'>Добавить книгу</button>";

            echo "</div>";

        }

        else

        {

            echo "Зарегистрированных пользователей еще нет.";

        }

    }

    $result = $dbreg->query("SELECT \* FROM users WHERE id=$user\_id");

    if ($result->num\_rows > 0)

    {

        $user = $result->fetch\_assoc();

        echo "<h2> </h2>";

    }

    else

    {

        echo "Пользователь не найден!";

    }

}

else

{

    header("Location: openFirst.html");

    exit();

}

$dbreg->close();

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <link rel="stylesheet" href="styles.css">

    <title>Мой аккаунт</title>

</head>

<body>

    <div class="container">

        <h2>Добро пожаловать, <?php echo $user['username']; ?>!</h2>

        <p>Ваш ID: <?php echo $user['id']; ?></p>

        <h3>Изменить данные:</h3>

        <form action="editUser.php" method="post" onsubmit="return corPassword()">

            <label for="tel">Контактный телефон:</label>

            <input type="text" id="new\_tel" name="new\_tel" value="<?php echo $user['tel'];?>" required>

            <input type="hidden" name="edit\_user\_id" id="edit\_user\_id" value="<?php echo $user['id'];?>">

            <label for="new\_username">Новое имя пользователя:</label>

            <input type="text" id="new\_username" name="new\_username" value="<?php echo $user['username'];?>" required>

            <label for="new\_password">Новый пароль:</label>

            <input type="password" id="new\_password" name="new\_password" required>

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

            <button onclick="deleteData(<?php echo $user['id']?>)">Удалить данного пользователя</button>

        </form>

        <br>

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

        <a href="books.php">Назад</a>

    </div>

    <script src="admin.js">

    </script>

</body>

</html>

Первоначально используется язык php для проверки установленного в сессии id пользователя. Если id не найден, тогда пользователя отправит на страницу openFirst.html. Далее с помощью базы данных определяется роль пользователя, и, в зависимости от того, является ли он администратором или нет, выводится соответствующий личный кабинет. Панель администратора позволяет изменять данные пользователя, удалять и добавлять аккаунты, а также добавлять новые книги для главной страницы books.php. Для того чтобы обычный пользователь смог изменить данные, используется все тот же тег <form> с атрибутом action, позволяющим перейти к файлу editUser.php:

<?php

require('dataBase.php');

session\_start();

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

    $editUserId = $\_POST['edit\_user\_id'];

    $newUsername = $\_POST['new\_username'];

    $newPassword = password\_hash($\_POST['new\_password'], PASSWORD\_DEFAULT);

    $newPhone=$\_POST['new\_tel'];

    if ($dbreg->query("UPDATE users SET username='$newUsername', password='$newPassword', tel='$newPhone' WHERE id=$editUserId"))

    {

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

        $role=$dbreg->query("SELECT role FROM users WHERE id=$userId");

        $userRole=$role->fetch\_assoc()['role'];

        if ($userRole =='admin')

        {

            header("Location: page.php");

            exit();

        }

        else

        {

            header("Location: openFirst.html");

            exit();

        }

    }

    else

    {

        echo "При обновлении данных возникла ошибка: " . $dbreg->error;

    }

}

else

{

    echo "Некорректный запрос";

}

$dbreg->close();

?>

Здесь важно отметить лишь одну деталь: обычный пользователь после изменения данных будет отправлен на первичную страницу openFirst.html, в то время как администратор останется в личном кабинете.

Кроме того при нажатии на кнопку «выйти» сработает переход по ссылке на документ logout.php:

<?php

session\_start();

session\_destroy();

header("Location: openFirst.html");

?>

По сути своей он заканчивает всю сессию, сбрасывая id пользователя, и переносит на странницу openFirst.html.

Для того, чтобы настроить функционал администратора, используется файл admin.js:

function editUserData(userId) {

    document.getElementById("edit\_user\_id").value = userId;

    document.getElementById("new\_username").value = prompt("Enter new username:", "");

    document.getElementById("new\_password").value = prompt("Enter new password:", "");

    document.getElementById("new\_tel").value = prompt("Enter new phone:", "");

    document.forms[0].submit();

}

function deleteData(userId) {

    var confirmDelete = confirm("Вы уверены, что хотите удалить данного пользователя?");

    if (confirmDelete) {

        var deleteForm = document.createElement("form");

        deleteForm.method = "post";

        deleteForm.action = "deleteUsers.php";

        var input = document.createElement("input");

        input.type = "hidden";

        input.name = "user\_id";

        input.value = userId;

        deleteForm.appendChild(input);

        document.body.appendChild(deleteForm);

        deleteForm.submit();

    }

}

function addUser() {

    var new\_username = prompt("Enter new username:", "");

    var new\_password = prompt("Enter new password:", "");

    var new\_tel = prompt("Enter new phone:", "");

    var usernameInput = document.createElement("input");

    usernameInput.type = "text";

    usernameInput.name = "new\_username";

    usernameInput.value = new\_username;

    var passwordInput = document.createElement("input");

    passwordInput.type = "password";

    passwordInput.name = "new\_password";

    passwordInput.value = new\_password;

    var telInput = document.createElement("input");

    telInput.type = "text";

    telInput.name = "new\_tel";

    telInput.value = new\_tel;

    var addButton = document.createElement("button");

    addButton.type = "submit";

    var form = document.createElement("form");

    form.method = "post";

    form.action = "addUsers.php";

    form.appendChild(usernameInput);

    form.appendChild(passwordInput);

    form.appendChild(telInput);

    form.appendChild(addButton);

    document.body.appendChild(form);

    form.submit();

}

function addBook(){

    var book = prompt("Enter new book:", "");

    var author = prompt("Enter author:", "");

    var image = prompt("Enter imagepath:", "");

    var desc = prompt("Enter description:", "");

    var price = prompt("Enter price:", "");

    var priceInput =  document.createElement("input");

    priceInput.type = "text";

    priceInput.name = "price";

    priceInput.value = price;

    var bookInput =  document.createElement("input");

    bookInput.type = "text";

    bookInput.name = "new\_book";

    bookInput.value = book;

    var authorInput =  document.createElement("input");

    authorInput.type = "text";

    authorInput.name = "author";

    authorInput.value = author;

    var imageInput =  document.createElement("input");

    imageInput.type = "text";

    imageInput.name = "image";

    imageInput.value = image;

    var descInput =  document.createElement("input");

    descInput.type = "text";

    descInput.name = "desc";

    descInput.value = desc;

    var addButton = document.createElement("button");

    addButton.type = "submit";

    var form = document.createElement("form");

    form.method = "post";

    form.action = "addBooks.php";

    form.appendChild(bookInput);

    form.appendChild(authorInput);

    form.appendChild(imageInput);

    form.appendChild(descInput);

    form.appendChild(priceInput);

    form.appendChild(addButton);

    document.body.appendChild(form);

    form.submit();

}

function corPassword() {

    var password = document.getElementById('new\_password').value;

    if (password.length < 5) {

        alert("Пароль должен содержать не менее 5 символов");

        return false;

    }

    var phone= document.getElementById('new\_tel').value;

    var str = /^8\d{10}$/;

    if(!str.test(phone)){alert("Введите корректный телефон в формате: 88005553535");return false;}

    return true;

}

В качестве стилей выбран файл styles.css, используемый для форм регистрации и авторизации.

# Заключение

В результате можно отметить, что цель данной курсовой работы была выполнена. Создано веб-приложение с использованием современных веб-технологий (HTML, CSS, JavaScript, PHP и MySQL) и база данных для управления пользователями и внутренними услугами. Проект получился достаточно минималистичным, однако он обладает всем необходимым функционалом.

Представлены формы регистрации и авторизации: использованы простейшие алгоритмы, связанные с безопасностью, например, хешированное хранение пароля и проверка его минимальной длины.

Разработана основная страница интернет-магазина книг: функционал определяется для авторизованных и неавторизованных пользователей по-разному; для авторизованных пользователей имеется возможность увидеть основную информацию по книге (от ее названия до цены) и приобрести ее, нажав на соответствующую кнопку «купить», в то время как неавторизованные пользователи смогут увидеть лишь представленные товары; реализована функция, позволяющая увеличивать счетчик покупок; используются функции для работы с корзиной: ее можно очистить, закрыть или использовать для покупки всех выбранных книг; в целом использован интуитивно понятный и приятный глазу интерфейс.

Создан личный кабинет: вход доступен только для авторизованных пользователей; имеется возможность изменить все необходимые данные (телефон, логин, пароль); имеется возможность выхода из аккаунта или полного его удаления; если пользователь обладает ролью администратора, тогда будет отображена специальная панель, с помощью которой можно изменять данные пользователя, удалять и добавлять аккаунты, а также добавлять новые книги для главной страницы интернет-магазина.

# Список использованных источников

1. Основы CSS: mdn web docs [Электронный источник]. – URL: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/CSS\_basics [дата доступа 11.01.2024].

2. Основы HTML: mdn web docs [Электронный источник]. – URL: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/HTML\_basics [дата доступа 11.01.2024].

3. Основы JavaScript: mdn web docs [Электронный источник].– URL: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/JavaScript\_basics [дата доступа 11.01.2024].

4. Что такое MySQL: Макхост [Электронный источник]. – URL: https://mchost.ru/articles/chto-takoe-mysql/ [дата доступа 12.01.2024].

5. PHP: википедия [Электронный источник]. – URL: https://ru.wikipedia.org/wiki/PHP [дата доступа 12.01.2024].