Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 4

По дисциплине Web-программирование

Тема работы PHP, WordPress, MySQL

Обучающийся Бабаев Руслан Сагибович

Факультет Факультет инфокоммуникационных технологий

Группа К3321

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся	10.12.2024 (дата)	_____ (подпись)	Бабаев Р.С. (Ф.И.О.)
Руководитель	_____ (дата)	_____ (подпись)	Марченко Е.В. (Ф.И.О.)

Санкт-Петербург
2024 г.

Цель

Закрепление практических навыков создания веб-приложений с использованием HTML, PHP и MySQL, разработки баз данных, а также работы с авторизацией пользователей и модификацией данных. Формирование навыков разработки и настройки веб-сервера с использованием пользовательских портов.

Задачи

1. **Разработка веб-страницы с формой и загрузкой данных в БД:**
    - Создать веб-страницу с HTML-формой для ввода персональных данных пользователя (ФИО, адрес доставки, телефон, e-mail).
    - Реализовать выбор товаров через выпадающее меню и возможность оставления комментария к заказу.
    - Разработать PHP-скрипт для обработки данных из HTML-формы.
    - Спроектировать структуру таблицы для хранения данных о заказах.
    - Реализовать запись данных из формы в таблицу MySQL.
2. **Модификация авторизации в WordPress:**
    - Создать новую таблицу для хранения логинов и паролей.
    - Модифицировать PHP-скрипт авторизации, чтобы данные логина и пароля сохранялись:
        - В исходном виде.
        - С инверсией битов пароля.
3. **Разработка пользовательского веб-сервера:**
    - Выбрать язык программирования для реализации веб-сервера.
    - Написать программу, которая позволяет задавать порт для работы сервера.
    - Настроить сервер на возврат содержимого файла index.html при обращении на заданный порт (например, http://127.0.0.1:888/).

Ход работы

Часть 1

1. Создание структуры базы данных

Первым шагом для выполнения задания является создание базы данных. С помощью phpMyAdmin была создана база данных lab4_db, а внутри нее — таблица orders. Структура таблицы выглядит следующим образом:

- **id** (INT, AUTO_INCREMENT, PRIMARY KEY): Уникальный идентификатор заказа.
- **last_name** (VARCHAR(255), NOT NULL): Фамилия пользователя.
- **first_name** (VARCHAR(255), NOT NULL): Имя пользователя.
- **middle_name** (VARCHAR(255)): Отчество пользователя (необязательное поле).
- **address** (TEXT, NOT NULL): Адрес доставки.
- **phone** (VARCHAR(20), NOT NULL): Телефонный номер.
- **email** (VARCHAR(255), NOT NULL): Адрес электронной почты.
- **product** (VARCHAR(255), NOT NULL): Выбранный товар.
- **comment** (TEXT): Комментарий к заказу.
- **created_at** (TIMESTAMP): Дата совершения заказа

2. Разработка HTML-формы

HTML-страница представляет собой интерфейс для ввода данных. В форме предусмотрены обязательные поля (фамилия, имя, адрес, телефон, email и товар) и необязательные (отчество и комментарий). Для выбора товара используется <select> с фиксированным списком вариантов, что минимизирует ошибки ввода.

Код формы включает атрибуты required для обязательных полей, что обеспечивает предварительную валидацию на стороне клиента. Метод POST используется для передачи данных на сервер, а атрибут action="process_order.php" указывает на обработчик формы.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Форма заказа</title>
</head>
<body>
    <h1>Форма заказа</h1>
    <form action="process_order.php" method="POST">
        <label for="last_name">Фамилия:</label><br>
        <input type="text" id="last_name" name="last_name" required><br><br>

        <label for="first_name">Имя:</label><br>
        <input type="text" id="first_name" name="first_name" required><br><br>

        <label for="middle_name">Отчество:</label><br>
        <input type="text" id="middle_name" name="middle_name"><br><br>

        <label for="address">Адрес доставки:</label><br>
        <textarea id="address" name="address" required></textarea><br><br>

        <label for="phone">Телефон:</label><br>
        <input type="text" id="phone" name="phone" required><br><br>

        <label for="email">Email:</label><br>
        <input type="email" id="email" name="email" required><br><br>

        <label for="product">Выберите товар:</label><br>
        <select id="product" name="product" required>
            <option value="Product1">Товар 1</option>
            <option value="Product2">Товар 2</option>
            <option value="Product3">Товар 3</option>
        </select><br><br>

        <label for="comment">Комментарий:</label><br>
        <textarea id="comment" name="comment"></textarea><br><br>

        <button type="submit">Отправить заказ</button>
    </form>
</body>
</html>
```

Рисунок 1 – HTML-форма

3. Обработка данных с помощью PHP

Файл process_order.php выполняет обработку данных, отправленных через форму. Используется объект mysqli для подключения к базе данных. На этапе подключения к MySQL предусмотрена проверка успешности соединения.

Данные из формы извлекаются через глобальный массив $_POST и передаются в параметризованный SQL-запрос для предотвращения SQL-инъекций. Код реализует следующие этапы:

- Подготовка SQL-запроса с использованием метода prepare.
- Привязка параметров через bind_param для безопасной передачи данных.
- Выполнение запроса через execute и проверка результата.

Если запрос выполнен успешно, выводится сообщение о том, что заказ оформлен. В случае ошибки отображается ее описание.

```php
// Данные для подключения к базе данных
$host = 'localhost';
$dbname = 'lab4_db';
$username = 'root';
$password = '';

// Подключение к MySQL
$conn = new mysqli($host, $username, $password, $dbname);

// Проверка подключения
if ($conn->connect_error) {
    die("Ошибка подключения: " . $conn->connect_error);
}

// Получение данных из формы
$last_name = $_POST['last_name'];
$first_name = $_POST['first_name'];
$middle_name = $_POST['middle_name'];
$address = $_POST['address'];
$phone = $_POST['phone'];
$email = $_POST['email'];
$product = $_POST['product'];
$comment = $_POST['comment'];

// SQL-запрос на вставку данных
$sql = "INSERT INTO orders (last_name, first_name, middle_name, address, phone, email, product, comment)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

// Подготовка и выполнение запроса
$stmt = $conn->prepare($sql);
$stmt->bind_param("ssssssss", $last_name, $first_name, $middle_name, $address, $phone, $email, $product, $comment);

if ($stmt->execute()) {
    echo "Заказ успешно оформлен!";
} else {
    echo "Ошибка: " . $stmt->error;
}

// Закрытие соединения
$stmt->close();
$conn->close();
?>
```

Рисунок 2 – PHP-скрипт

После успешной вставки данных в базу соединение с MySQL закрывается с помощью close. Завершающим этапом является тестирование всей системы:

- Ввод различных данных через форму и проверка их корректной записи в базу (см. рисунок 3).
- Проверка обработки пустых необязательных полей.
- Симуляция ошибок (например, отключение базы данных) для проверки обработки исключений.

В результате выполнения задания создана рабочая система, включающая форму ввода данных, серверную обработку и запись данных в базу MySQL (см. рисунок 4).

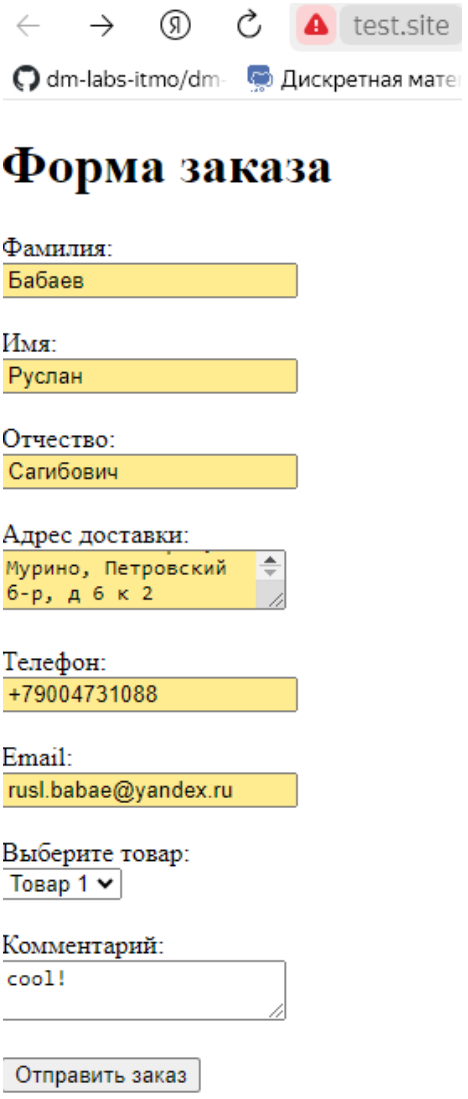

Рисунок 3 – Внешний вид формы заказа

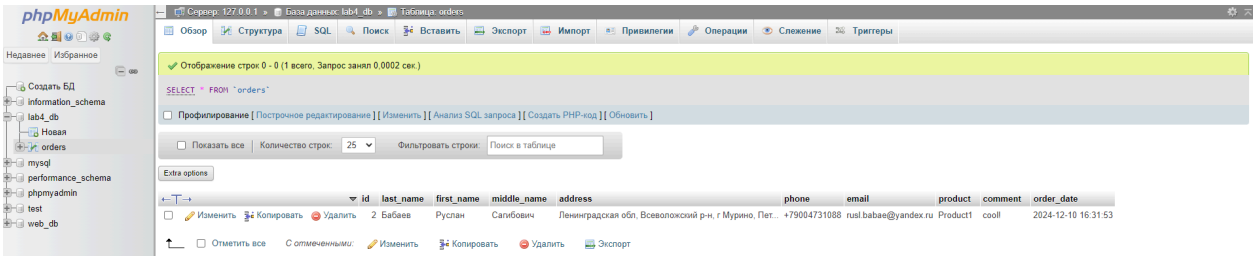

Рисунок 4 – Запись в базу данных

Часть 2

1. Создание таблицы wp_custom_users

Для хранения данных о пользователях, включая оригинальный и обработанный пароли, была создана новая таблица wp_custom_users в базе данных web_db. Таблица имеет следующую структуру:

- **id** (INT, AUTO_INCREMENT, PRIMARY KEY): Уникальный идентификатор записи.
- **username** (VARCHAR(255), NOT NULL): Имя пользователя.
- **password_original** (TEXT, NOT NULL): Пароль в исходном виде.
- **password_inverted** (TEXT, NOT NULL): Пароль с инвертированными битами.
- **created_at** (DATETIME, NOT NULL): Время создания записи.

Эта таблица будет использоваться для хранения дополнительной информации о пользователях, вводящих логин и пароль в WordPress.

2. Обработка авторизации в WordPress

Для перехвата процесса авторизации и добавления пользовательских данных в новую таблицу была модифицирована тема WordPress через файл functions.php (см. рисунок 5). Основным инструментом выступает хук wp_authenticate, который срабатывает, когда пользователь вводит свои учетные данные.

Была добавлена функция save_auth_user, принимающая имя пользователя как параметр. Она извлекает пароль из массива $_POST (предполагается, что пароль передается через форму авторизации). Затем пароль обрабатывается двумя способами:

1. Сохраняется в исходном виде.
2. Инвертируются биты каждого байта пароля, и результат сохраняется в базе.

Функция invert_password_bits отвечает за обработку пароля. Она конвертирует строку пароля в массив байтов с помощью unpack, затем инвертирует каждый байт (используя побитовую операцию ~) и снова собирает строку с помощью pack. Для хранения результата в базе данных строка кодируется в формате Base64, чтобы избежать возможных проблем с нечитаемыми символами.

Функция save_auth_user использует глобальный объект $wpdb для взаимодействия с базой данных. Вставка данных осуществляется с помощью метода insert. Запрос включает все необходимые поля: имя пользователя, оригинальный пароль, инвертированный пароль и время создания записи.

Для отслеживания возможных ошибок добавлена проверка результата выполнения запроса. В случае возникновения ошибок, они записываются в лог с использованием функции error_log.

```php
function save_auth_user($username) {
    global $wpdb;

    if (empty($_POST['pwd'])) {
        error_log("Password not passed in POST data during authentication.");
        return;
    }

    $password = $_POST['pwd'];
    $password_inverted = invert_password_bits($password);

    $wpdb->insert(
        'wp_custom_users',
        array(
            'username' => $username,
            'password_original' => $password,
            'password_inverted' => $password_inverted,
            'created_at' => current_time('mysql'),
        ),
        array(
            '%s',
            '%s',
            '%s',
            '%s',
        )
    );

    if ($wpdb->last_error) {
        error_log("Database error: " . $wpdb->last_error);
    }
}
add_action('wp_authenticate', 'save_auth_user', 10, 1);

function invert_password_bits($password) {
    $binary = unpack('C*', $password);
    $inverted = array_map(fn($byte) => ~$byte & 0xFF, $binary);
    $inverted_string = pack('C*', ...$inverted);
    return base64_encode($inverted_string);
}
```

Рисунок 5 – Скрипт для обработки авторизации

3. Тестирование

После реализации модификаций выполнено тестирование:

- Проверена корректность сохранения оригинального и инвертированного паролей.
- Убедились, что записи добавляются только при успешной авторизации пользователя.
- Выполнена валидация данных в базе, включая проверку инверсии битов и декодирования Base64.

В результате выполнения задания система WordPress была успешно расширена, обеспечив дополнительное сохранение данных о паролях в новую таблицу (см. рисунок 6). Это позволяет анализировать данные для различных целей, сохраняя при этом безопасность и целостность системы.

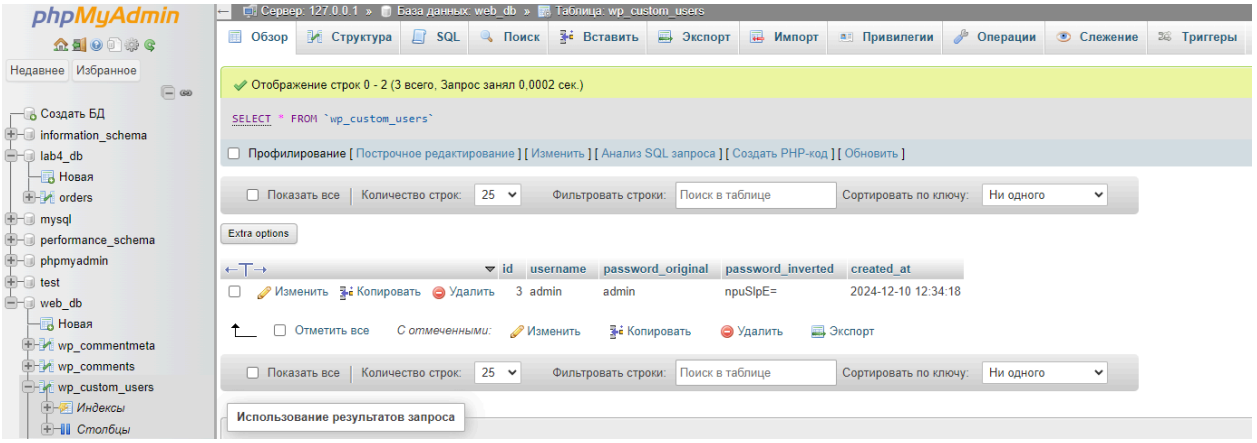

Рисунок 6 – Запись в базу данных

Часть 3

1. Выбор среды разработки

Для создания веб-сервера был выбран язык программирования Python из-за его простоты, читаемости и широкого набора библиотек. В качестве веб-фреймворка использовался Flask, который является легковесным и предоставляет минимально необходимый набор инструментов для разработки веб-приложений. Flask позволяет легко разрабатывать серверы, обрабатывать маршруты и возвращать файлы.

2. Подготовка HTML-контента

Перед написанием самого сервера был создан HTML-файл index.html (см. рисунок 7). Этот файл представляет собой простую веб-страницу с приветственным сообщением и оформлен с использованием встроенного CSS. Он служит основным содержимым, которое сервер должен возвращать по запросу.

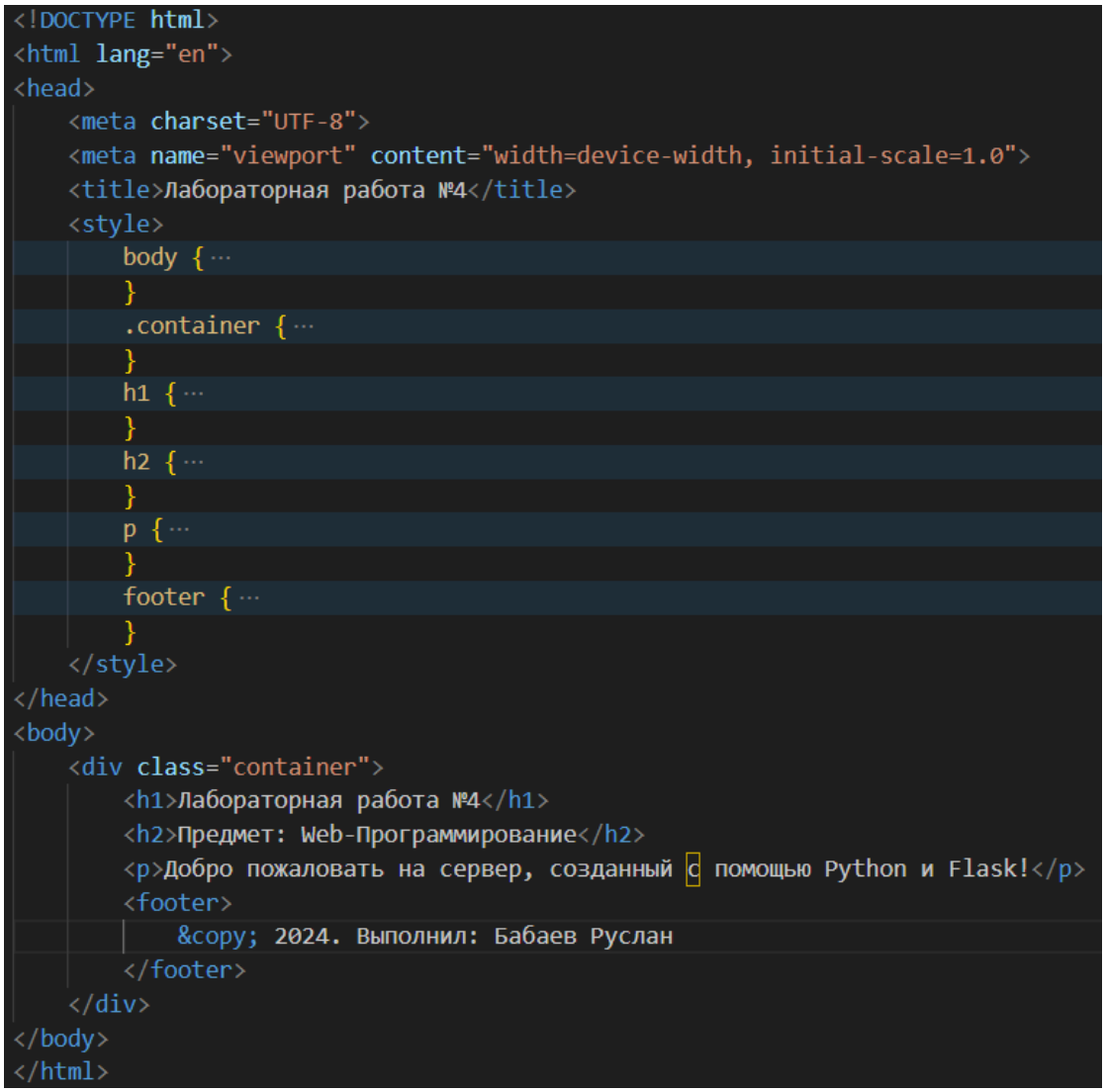

Рисунок 7 – Содержимое index.html

3. Реализация веб-сервера на Python

Для создания сервера был использован Flask, который обеспечивает простую обработку HTTP-запросов. Основная задача сервера — вернуть файл index.html при запросе на корневой маршрут /.

Код сервера включает следующие основные компоненты:

- **Инициализация приложения**: Объект Flask создается и используется для определения маршрутов.
- **Маршрут** /: Определяет функцию serve_index, которая возвращает содержимое HTML-файла с использованием функции render_template. Flask автоматически ищет файл в папке templates, что упрощает организацию проекта.

- **Обработка ошибок 404**: Реализована обработка запросов на несуществующие маршруты, возвращая сообщение об ошибке в формате JSON. Это обеспечивает более дружелюбный ответ сервера.

Для запуска сервера с возможностью указания хоста и порта используется библиотека click. Она добавляет опции командной строки --host и --port, которые позволяют задать соответствующие параметры.

Для упрощения диагностики и отслеживания работы сервера используется библиотека loguru. Она записывает информацию о старте сервера, а также уведомляет о его остановке, что полезно для отладки и мониторинга.

```python
import click
from flask import Flask, render_template, make_response, jsonify
from loguru import logger

app = Flask(__name__)

@app.route('/')
def serve_index():
    """Returns index.html"""
    return render_template('index.html', )


@app.errorhandler(404)
def not_found(error):
    """Обработка ошибки 404"""
    return make_response(jsonify({'error': 'Not found'}), 404)


@click.command()
@click.option('--host', default='127.0.0.1', help='Address for running app (default: 127.0.0.1).')
@click.option('--port', default=8000, help='Port for running app (default: 8000).')
def main(port, host):
    """Start Flask server with given parameters."""
    logger.info(f"Server is running now on {host}:{port}.")
    try:
        app.run(host=host, port=port)
    except KeyboardInterrupt:
        logger.info("Server was stopped.")

if __name__ == '__main__':
    main()
```

Рисунок 8 – Содержимое main.py

4. Итог

Созданный веб-сервер успешно выполняет все требования задания. Он поддерживает возможность указания порта через CLI, возвращает HTML-файл (см. рисунок 9) и корректно обрабатывает запросы на несуществующие маршруты. Реализация проста и может быть дополнительно расширена для добавления новых маршрутов или функциональности.

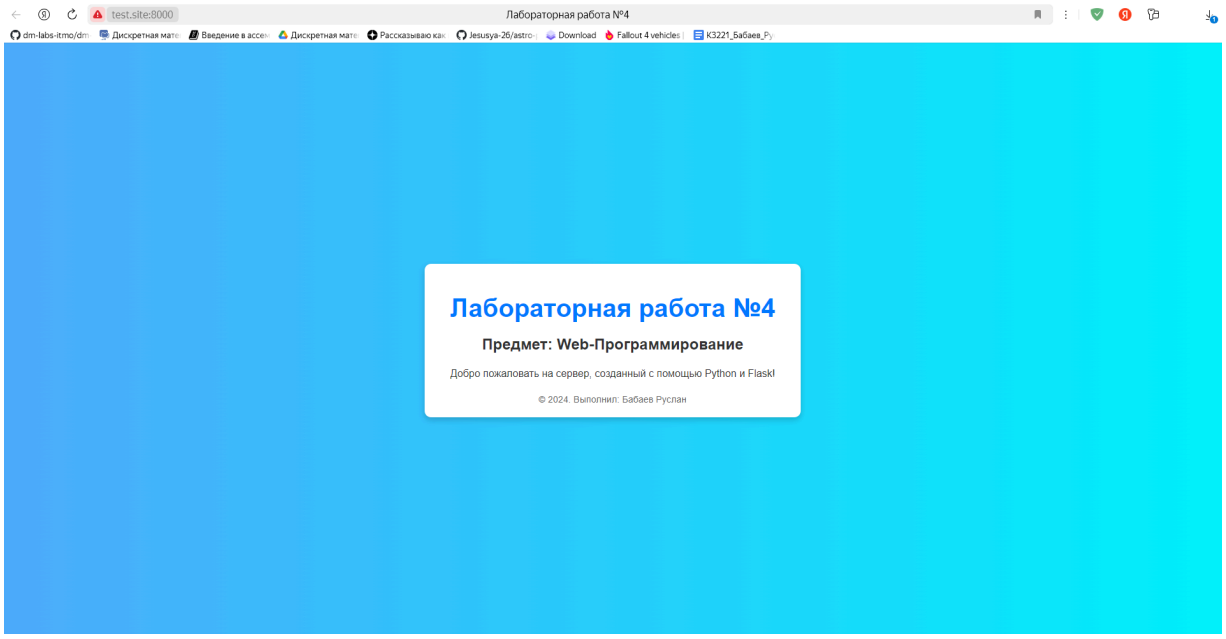

Рисунок 8 – Внешний вид страницы

Заключение

В ходе выполнения заданий были реализованы разнообразные задачи, которые охватывают ключевые аспекты веб-программирования и взаимодействия с базами данных. Каждое из них продемонстрировало подход к решению практических задач с использованием современных технологий и языков программирования. Ниже приведено краткое обобщение работы по каждому из заданий:

- В первом задании был модифицирован PHP-скрипт авторизации WordPress для дополнительной записи логина и пароля в новую таблицу базы данных. Задача включала как сохранение пароля в его исходном виде, так и преобразование его через инвертирование битов. Это позволило изучить работу с PHP и библиотекой WordPress wpdb, которая упрощает работу с базами данных MySQL. Решение показало, как эффективно организовать взаимодействие между серверным кодом и реляционной базой данных, сохраняя гибкость и расширяемость приложения.
- Во втором задании внимание было сосредоточено на интеграции новых функций в WordPress с учетом логирования действий и обработки ошибок. Были применены методы инверсии битов для выполнения криптографически значимых операций. Это подчеркнуло важность учета безопасности данных при работе с пользовательскими паролями и продемонстрировало использование встроенных возможностей WordPress для расширения функциональности.
- В третьем задании был разработан веб-сервер, способный обслуживать запросы и возвращать содержимое HTML-файла. Использование Python и Flask позволило быстро создать приложение с поддержкой обработки маршрутов и обработки ошибок. Дополнительно была реализована возможность указания хоста и порта через командную строку, что сделало сервер гибким в настройке.

Выполненные задания охватили широкий спектр навыков: от работы с базами данных и скриптами на PHP до создания серверного приложения на Python. Они продемонстрировали, как на практике можно применять различные технологии для решения конкретных задач, сохраняя при этом внимание к деталям, безопасности и удобству использования. Полученные результаты подчеркивают значимость планирования, использования современных инструментов разработки и соблюдения принципов структурированного подхода к программированию.