

УНИВЕРСИТЕТ ИТМО

Факультет «Инфокоммуникационных технологий»
Направление подготовки «Программирование в инфокоммуникационных
системах»

Лабораторная работа №4

Выполнил:
Горлов Игорь Витальевич
Группа №К3321
Проверила:
Марченко Елена Владимировна

Санкт-Петербург
2024

Цель работы:

Разработать функциональный и безопасный веб-сервис для обработки и хранения пользовательских данных, реализации дополнительных методов аутентификации, а также создания собственного web-сервера с настройкой порта для локального доступа.

Задачи:

1. Создать веб-страницу для ввода данных пользователя, включающую поля для фамилии, имени, отчества, адреса, телефона, email, выбора товаров и добавления комментария.
2. Модифицировать существующий PHP-скрипт для авторизации в движке WordPress.
3. Разработать собственный web-сервер с возможностью указания порта.

Ссылка на удаленный репозиторий:

https://github.com/kew2023/WebDevelopment_2024-2025

Ход работы:

Задание 1.

Для выполнения данной задачи создадим в файле *functions.php* нашей темы *шорткод* для создания необходимой формы.

```
function display_order_form(): bool|string {  
    ob_start(); // Начало буфера вывода  
  
    ?>  
    <div class="order-form">  
        <h2 class="order-form-title">Форма заказа</h2>  
        <form action="<?php echo esc_url(admin_url(path: 'admin-post.php')); ?>" method="POST">  
            <input type="hidden" name="action" value="process_order_form">  
  
            <label class="order-form-label">Фамилия:  
                <input class="order-form-input" type="text" name="last_name" required>  
            </label>  
  
            <label class="order-form-label">Имя:  
                <input class="order-form-input" type="text" name="first_name" required>  
            </label>  
  
            <label class="order-form-label">Отчество:  
                <input class="order-form-input" type="text" name="middle_name">  
            </label>  
  
            <label class="order-form-label">Адрес:  
                <textarea class="order-form-textarea" name="address" required></textarea>  
            </label>  
  
            <label class="order-form-label">Телефон:  
                <input class="order-form-input" type="tel" name="phone" required>  
            </label>  
  
            <label class="order-form-label">Электронная почта:  
                <input class="order-form-input" type="email" name="email" required>  
            </label>  
  
            <label class="order-form-label">Выберите товар:  
                <select class="order-form-select" name="product" required>  
                    <option value="Товар 1">Товар 1</option>  
                    <option value="Товар 2">Товар 2</option>  
                    <option value="Товар 3">Товар 3</option>  
                    <option value="Товар 4">Товар 4</option>  
                </select>  
            </label>  
  
            <label class="order-form-label">Комментарий к заказу:  
                <textarea class="order-form-textarea" name="comment"></textarea>  
            </label>  
  
            <button class="order-form-button" type="submit">Отправить заказ</button>  
        </form>  
    }  
}
```

Рисунок 1 – Создание формы

```
// Регистрируем шорткод
add_shortcode(tag: 'order_form', callback: 'display_order_form');
```

Рисунок 2 – Регистрация шорткода

Далее вставим полученный шорткод на страницу в WordPress.

Теперь создадим таблицу в базе данных для обработки формы следующим SQL-запросом.

```
1 CREATE TABLE orders (
2     id INT AUTO_INCREMENT PRIMARY KEY,
3     last_name VARCHAR(50) NOT NULL,
4     first_name VARCHAR(50) NOT NULL,
5     middle_name VARCHAR(50),
6     address TEXT NOT NULL,
7     phone VARCHAR(20) NOT NULL,
8     email VARCHAR(100) NOT NULL,
9     product VARCHAR(100) NOT NULL,
10    comment TEXT,
11    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
12 );
13
```

Рисунок 3 – Создание таблицы в базе данных

И занесем полученные данные в созданную таблицу.

```
// Обработка формы
0 references | Codeium: Refactor | Explain | X
function process_order_form(): never {
    global $wpdb;

    // Получение данных из формы и их обработка
    $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'];

    // Вставка данных в таблицу базы данных
    $wpdb->insert(table: 'orders', data: [
        'last_name' => $last_name,
        'first_name' => $first_name,
        'middle_name' => $middle_name,
        'address' => $address,
        'phone' => $phone,
        'email' => $email,
        'product' => $product,
        'comment' => $comment,
        'created_at' => current_time(type: 'mysql'),
    ]);

    // Перенаправление после отправки формы
    wp_redirect(location: home_url(path: '/'));
    exit;
}

// Регистрируем обработчик формы
add_action(hook_name: 'admin_post_nopriv_process_order_form', callback: 'process_order_form');
add_action(hook_name: 'admin_post_process_order_form', callback: 'process_order_form');
```

Рисунок 4 – Обработка формы

Результат работы представлен далее

Форма для получения
товара

Форма заказа

Фамилия:

Горлов

Имя:

Игорь

Отчество:

Витальевич

Адрес:

НОВОИЗМАЙЛОВСКИЙ ПР-КТ,16,8,46

Телефон:

89523096126

Электронная почта:

igorlov2604@gmail.com

Выберите товар:

Товар 3

Комментарий к заказу:

МСГ

Отправить заказ

Рисунок 5 – Полученная форма

После отправки формы в таблице появляется новая запись

SELECT * FROM `orders`

Профилирование [Построчное редактирование] [Изменить] [Анализ SQL запроса] [Создать PHP-код] [Обновить]

Показать все

Количество строк: 25

Фильтровать строки: Поиск в таблице

Extra options

	id	last_name	first_name	middle_name	address	phone	email	product	comment	created_at
<div><div><div></div><div>Изменить</div><div>Копировать</div><div>Удалить</div></div><div>1</div></div>		Горлов	Игорь	Витальевич	НОВОИЗМАЙЛОВСКИЙ ПР-КТ,16,8,46	89523096126	igorlov2604@gmail.com	Товар 3	МСГ	2024-11-06 10:18:57

Рисунок 6 – Обновление в базе данных

Задание 2.

Для выполнения данного задания создадим новую таблицу в базе данных.

```
1 CREATE TABLE custom_user_logins (  
2     id INT NOT NULL AUTO_INCREMENT,  
3     user_login VARCHAR(60) NOT NULL,  
4     password_original VARCHAR(255) NOT NULL,  
5     password_inverted VARCHAR(255) NOT NULL,  
6     PRIMARY KEY (id)  
7 );  
8 |
```

Рисунок 7 – Создание таблицы в базе данных

Необходимо разрешить регистрацию любым пользователям.

Далее создадим функцию, которая при помощи хука *user_register* будет перехватывать регистрацию пользователя. Также создадим функцию для инвертирования битов пароля.

```
// Функция для инвертирования битов пароля  
1 reference | Codeium: Refactor | Explain | X  
function invert_password_bits($password): string {  
    $binary_password = '';  
    for ($i = 0; $i < strlen(string: $password); $i++) {  
        $char = $password[$i];  
        $binary = decbin(num: ord(character: $char));  
        $inverted_binary = '';  
        for ($j = 0; $j < strlen(string: $binary); $j++) {  
            $inverted_binary .= $binary[$j] === '0' ? '1' : '0';  
        }  
        $inverted_char = chr(codepoint: bindec(binary_string: $inverted_binary));  
        $binary_password .= $inverted_char;  
    }  
    return $binary_password;  
}
```

Рисунок 8 – Инвертирование пароля

Так как созданный пользователь сразу получает пароль – получим его с помощью *get_userdata()*. Получив из функции данные о логине и пароле, сохраним также инвертированный пароль в базе данных.

```
// Обработчик регистрации пользователя
0 references | Codeium: Refactor | Explain | X
function save_user_login_to_custom_table($user_id): void {
    global $wpdb;

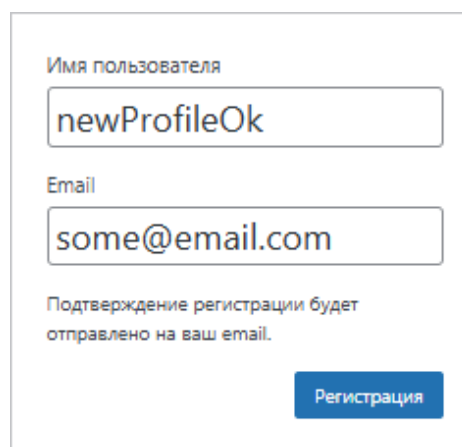
    // Получаем данные пользователя
    $user_data = get_userdata(user_id: $user_id);
    $user_login = $user_data->user_login;

    // Хешированный пароль
    $password_hash = $user_data->user_pass;

    // Инвертируем хеш пароля
    $password_inverted = invert_password_bits(password: $password_hash);

    // Записываем данные в новую таблицу
    $wpdb->insert(
        table: 'custom_user_logins',
        data: array(
            'user_login' => $user_login,
            'password_original' => $password_hash,
            'password_inverted' => $password_inverted,
        ),
        format: array(
            '%s',
            '%s',
            '%s'
        )
    );
}
add_action(hook_name: 'user_register', callback: 'save_user_login_to_custom_table');
```

Рисунок 9 – Запуск сервера для отправки форм
Теперь создадим нового пользователя.



Имя пользователя

Email

Подтверждение регистрации будет отправлено на ваш email.

Регистрация

Рисунок 10 – Создание пользователя

После регистрации пользователя в созданной ранее таблице
появляются необходимые данные

SELECT * FROM `custom_user_logins`

☐ Профилирование

[Построчное редактирование]

[Изменить]

[Анализ SQL запроса]

[Создать PHP-код]

[Обновить]

☐ Показать все

Количество строк: 25

Фильтровать строки:

Extra options

	id	user_login	password_original	password_inverted
<input type="checkbox"/>				<div><div>□/□=(□)64</div><div>□0□ □</div><div>□:*</div><div>□□ %+□8*□□</div></div>
<input type="checkbox"/>	2	newProfileOk	\$P\$BW7VIKrzOc6q2cEUXrdl63ZT8GU3sc0	

Рисунок 11 – Данные пользователя

Задание 3.

В данном задании сервер будет создаваться на NodeJS.

```
const http = require('http');
const fs = require('fs');
const path = require('path');

// Функция для запуска сервера на указанном порту
Codeium: Refactor | Explain | X
function startServer (port) {
  const server = http.createServer((req, res) => {
    if (req.url === '/') {
      // Путь к файлу index.html
      const filePath = path.join(__dirname, 'index.html');

      // Чтение файла index.html
      fs.readFile(filePath, (err, data) => {
        if (err) {
          res.writeHead(500, { 'Content-Type': 'text/plain' });
          res.end('500 Internal Server Error');
        } else {
          // Отправка содержимого файла index.html
          res.writeHead(200, { 'Content-Type': 'text/html' });
          res.end(data);
        }
      });
    } else {
      // Если запрос не на корневой URL
      res.writeHead(404, { 'Content-Type': 'text/plain' });
      res.end('404 Not Found');
    }
  });

  // Запуск сервера на указанном порту
  server.listen(port, () => {
    console.log(`Server is running on http://127.0.0.1:${port}`);
  });
}

// Ввод порта от пользователя через командную строку
const port = process.argv[2] || 3000; // По умолчанию порт 3000
startServer(port);
```

Рисунок 12 – Сервер на NodeJS

Данный код работает следующим образом.

Модули: *http* используется для создания сервера, *fs* — для чтения файлов, *path* — для работы с путями.

Функция *startServer(port)*:

- Создаёт *HTTP*-сервер, который обрабатывает запросы.
- При запросе на корневой *URL* (/), сервер читает файл *index.html* и отправляет его содержимое как ответ.
- Если файл не найден или запрос не является корневым, сервер возвращает 404 Not Found.

И запуск скрипта получает в аргументе порт сервера, а если его нет, то запускает сервер на 3000-ом порте.

Файл index.html находится в той же директории и выводит простейший заголовок.



Hello from NodeJs

Рисунок 13 – Результат вывода

Вывод: В ходе выполнения лабораторной работы были успешно выполнены поставленные задачи. Создана веб-страница, которая позволяет пользователю вводить личные данные и выбирать товары из предложенного списка. Введенные данные обрабатываются с помощью PHP-скрипта и сохраняются в базе данных MySQL, что подтверждает успешное подключение веб-интерфейса к серверной части и базе данных.

Также был модифицирован скрипт авторизации для движка WordPress. Введенные пользователем логин и пароль теперь записываются в дополнительную таблицу MySQL двумя способами: в исходном виде и с инверсией битов пароля. Эта модификация расширяет возможности хранения данных для анализа или экспериментов с шифрованием.

Кроме того, был разработан простой веб-сервер, который поддерживает указание порта для подключения. Сервер корректно возвращает содержимое файла index.html при обращении по адресу и порту, заданным пользователем. Данный опыт подтвердил навыки работы с сетевыми протоколами и настройкой локальных серверов, что полезно для создания серверного ПО на различных языках программирования.

Таким образом, все задачи были выполнены в полном объеме, что свидетельствует о приобретении практических навыков в разработке серверных приложений, работе с базами данных и обработке данных пользователей в веб-приложениях.