

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 MySQL, Wordpress, HTML, web server

Обучающийся Телунц Эдуард Рубенович

Факультет Факультет инфокоммуникационных технологий

Группа K3321

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся

17.11.2024

(дата)

(подпись)

Телунц Э.Р.

(Ф.И.О.)

Руководитель

(дата)

(подпись)

Марченко Е.В.

(Ф.И.О.)

Санкт-Петербург
2024 г.

ЦЕЛЬ РАБОТЫ:

Научиться работать MySQL, wordpress. Понять что такое web server и для чего он нужен, параллельно написав собственный.

ЗАДАНИЕ 1:

Web страница была написана с помощью движка представлений Razor. POST запрос связан непосредственно с объектом класса User:

```
@page
@model WebLab4Task1.Pages.IndexModel
@{
    <h1>Оформление заказа</h1>

    <form method="post">
        <div>
            <label asp-for="Person.Surname">Фамилия:</label>
            <input name="Person.Surname" />
        </div>

        <div>
            <label asp-for="Person.Name">Имя:</label>
            <input name="Person.Name" />
        </div>

        <div>
            <label asp-for="Person.Address">Адрес:</label>
            <input name="Person.Address" />
        </div>

        <div>
            <label asp-for="Person.Email">Электронная почта:</label>
            <input name="Person.Email" />
        </div>

        <div>
            <p>Выберите товар:</p>
            <input type="radio" id="laptop" name="Person.Product" value="Ноутбук" />
            <label for="laptop">Ноутбук</label><br />
            <input type="radio" id="bottle" name="Person.Product" value="Бутылка" />
            <label for="bottle">Бутылка</label>
            <input type="radio" id="plate" name="Person.Product" value="Тарелка" />
            <label for="plate">Тарелка</label>
            <input type="radio" id="tv" name="Person.Product" value="Телевизор" />
            <label for="tv">Телевизор</label>
        </div>

        <div>
            <label asp-for="Person.Comment">Комментарий по заказу:</label>
            <input name="Person.Comment" />
        </div>

        <button type="submit" class="btn btn-primary">Отправить заказ</button>
    </form>
}
```

Сам класс User:

```
public class User
{
    [Column("surname")]
    public string? Surname { get; set; }
    [Column("name")]
    public string? Name { get; set; }

    [Column("address")]
    public string? Address { get; set; }
    [Key]
    [Column("email")]
    public string Email { get; set; }

    [Column("product")]
    public string? Product { get; set; }
    [Column("comment")]
    public string? Comment { get; set; }
}
```

Модель базы для Entity Framework, чтобы записывать данные из формы в таблицу postgres:

```
public class ApplicationDbContext : DbContext
{
    public DbSet<User> Users { get; set; }

    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
    }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        if (!optionsBuilder.IsConfigured)
        {
            optionsBuilder.UseNpgsql("Host=localhost;Database=WebLab4Task1;Username=postgres;");
        }
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<User>().ToTable("users");
    }
}
```

Модель Index, для получения данных из формы в POST запросе:

```
[IgnoreAntiforgeryToken]
public class IndexModel : PageModel
{
    ApplicationDbContext context;
    [BindProperty]
    public User Person { get; set; } = new();
    public IndexModel(ApplicationDbContext db)
    {
        context = db;
    }
    public async Task<IActionResult> OnPostAsync()
    {
        context.Users.Add(Person);
        await context.SaveChangesAsync();
        return RedirectToPage("Index");
    }
}
```

Скрипт для создания таблицы users в базе данных PostgreSQL:

```
CREATE TABLE IF NOT EXISTS public.users
(
    surname character varying(255) COLLATE pg_catalog."default",
    name character varying(255) COLLATE pg_catalog."default",
    address character varying(255) COLLATE pg_catalog."default",
    email character varying(255) COLLATE pg_catalog."default" NOT NULL,
    product character varying(255) COLLATE pg_catalog."default",
    comment text COLLATE pg_catalog."default",
    CONSTRAINT users_pkey PRIMARY KEY (email)
)
```

Интерфейс в браузере:

Оформление заказа

Фамилия:

Имя:

Адрес:

Электронная почта:

Выберите товар:

☐ Ноутбук

☐ Бутылка

☐ Тарелка

☐ Телевизор

Комментарий по заказу:

Отправить заказ

Как можем видеть, при введении информации в нужные поля и при нажатии кнопки “Отправить заказ”, в таблице появляются записи:

	lastname character varying (255)	name character varying (255)	address character varying (255)	email [PK] character varying (255)	product character varying (255)	comment text
1	Telunts	Yulia	Sumskoy	edtelunts@gmail.com	Ноутбук	erjgne
2	Telunts	Hdjcdncnj	Sumskoy	teecortex@gmail.com	Телевизор	ydydydy

ЗАДАНИЕ 2

Для начала были написаны php функции для хэширования пароля и для записи данных в таблицу MySQL:

```

function invertBits($password) {
    $inverted = '';
    for ($i = 0; $i < strlen($password); $i++) {
        $char = $password[$i];
        $binary = str_pad(decbin(ord($char)), 8, '0', STR_PAD_LEFT);
        $binary_inverted = '';
        foreach (str_split($binary) as $bit) {
            $binary_inverted .= ($bit == '0') ? '1' : '0';
        }
        $inverted .= $binary_inverted;
    }
    return $inverted;
}

function saveUserInfo($user_login, $user){
    global $wpdb;

    $password = $_POST['pwd'];

    $encryptedPassword = invertBits($password);

    $wpdb->insert(
        'userprofile',
        array(
            'login' => $user_login,
            'password' => $password,
            'password_encrypted' => $encryptedPassword,
        ),
        array(
            '%s',
            '%s',
            '%s'
        )
    );
}

```

В файле `wp_config` были прописаны настройки для подключения к базе данных:

```
define( 'DB_NAME', 'lab4task2' );

/** Database username */
define( 'DB_USER', 'root' );

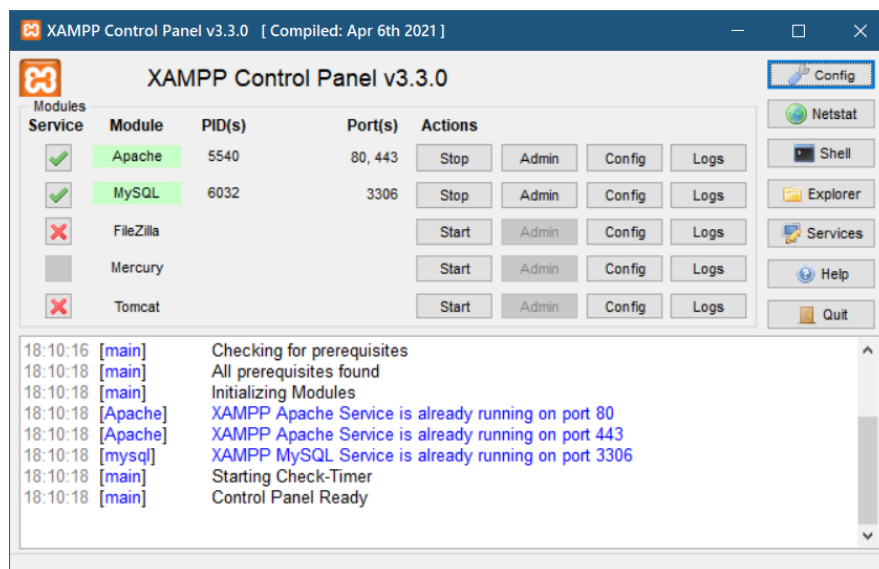
/** Database password */
define( 'DB_PASSWORD', '' );

/** Database hostname */
define( 'DB_HOST', 'localhost:3306' );

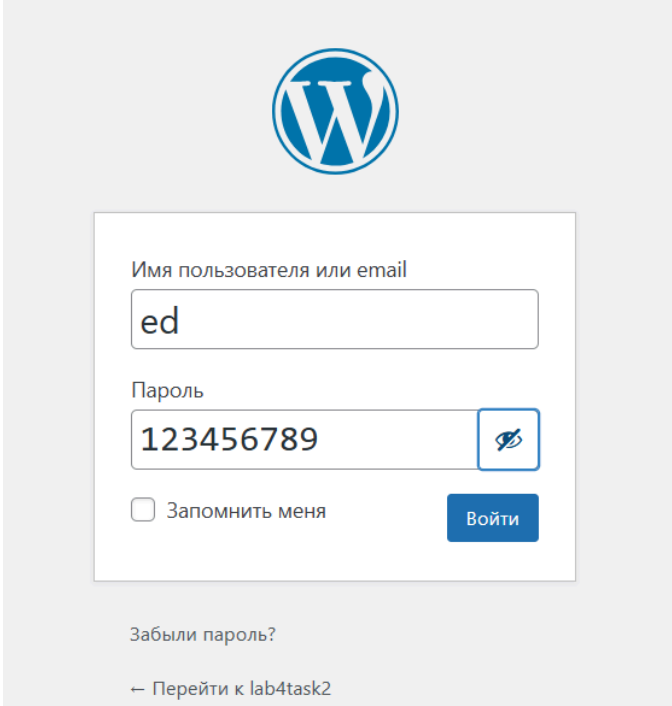
/** Database charset to use in creating */
define( 'DB_CHARSET', 'utf8' );

/** The database collate type. Don't ch
define( 'DB_COLLATE', '' );
```

Контрольная панель хампр:



Вводим данные при авторизации:

A screenshot of the WordPress login interface. At the top center is the WordPress logo, a blue 'W' inside a circle. Below it is a white rectangular login box with a thin border. Inside the box, the text 'Имя пользователя или email' is above a text input field containing 'ed'. Below that, the text 'Пароль' is above a password input field containing '123456789'. To the right of the password field is a small blue square button with a white eye icon. Below the password field is a checkbox labeled 'Запомнить меня'. To the right of the checkbox is a blue button with the text 'Войти'. Below the login box, the text 'Забыли пароль?' is displayed. At the bottom left, there is a link with a left-pointing arrow and the text 'Перейти к lab4task2'.

Как можем видеть - в таблице userprofile в rhpadmin появилась запись, которая свидетельствует о том, что авторизованный пользователь записался в таблицу.

ЗАДАНИЕ 3

На языке C# была написана программа, которая представляет собой web сервер:


```

internal class Program
{
    static async Task Main(string[] args)
    {
        int port = 888; // Порт по умолчанию

        Console.WriteLine("Введите номер порта для сервера:");
        var inputPort = Console.ReadLine();

        if (int.TryParse(inputPort, out int parsedPort))
        {
            port = parsedPort;
        }

        await StartServer(port);
    }

    private static async Task StartServer(int port)
    {
        HttpListener listener = new HttpListener();
        listener.Prefixes.Add($"http://*:{port}/");
        listener.Start();

        Console.WriteLine($"Сервер запущен на порту {port}.");
    }
}

```

```

while (true)
{
    HttpContext context = await listener.GetContextAsync();
    HttpRequest request = context.Request;
    HttpResponse response = context.Response;

    Console.WriteLine($"Получен запрос: {request.Url}");

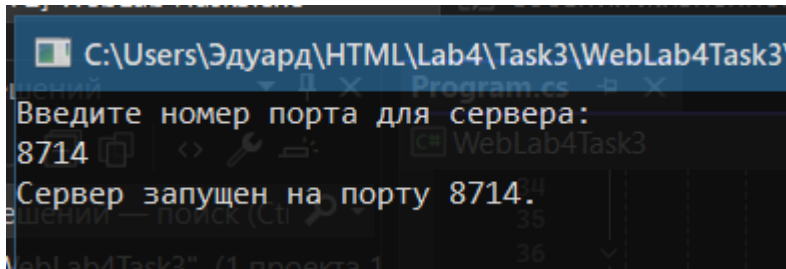
    byte[] buffer;

    if (File.Exists("index.html"))
    {
        buffer = File.ReadAllBytes("index.html");
        response.ContentType = "text/html";
    }
    else
    {
        buffer = Encoding.UTF8.GetBytes("<h1>Файл index.html не найден</h1>");
        response.StatusCode = (int)HttpStatusCode.NotFound;
    }

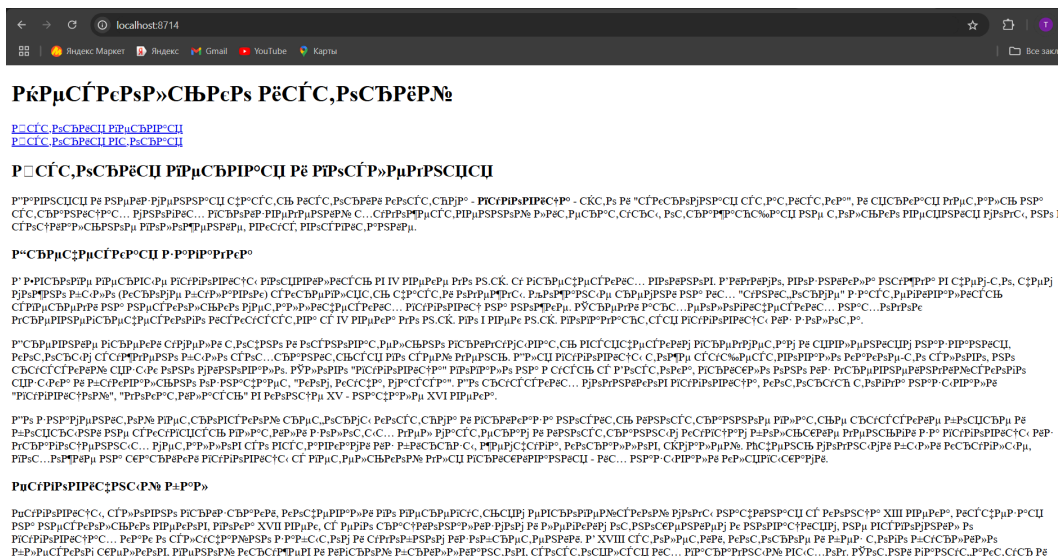
    response.ContentLength64 = buffer.Length;
    await response.OutputStream.WriteAsync(buffer, 0, buffer.Length);
    response.OutputStream.Close();
}
}

```

Программа была написана с помощью класса `HttpListener`, и в процессе запуска позволяет выбрать пользователю порт для запуска сервера, а также дефолтно запускает страницу `index.html`. Посмотрим на работу программы:



Наблюдаем верное (почти) отображение `index.html`:



ЗАКЛЮЧЕНИЕ:

По итогу работы мы научились работать с `wordpress`, также написали несколько `php` скриптов и создали `web` сервер, который показывает `index.html`. Также благодаря технологии `Razor Pages` была написана `cshtml` страница, которая отправляет `POST` запрос, впоследствии записывающийся в БД `PostgreSQL`.