**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ**

**ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным и практическим работам**

**МДК 11.01**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студент | ИСПП-35 | |  |  | |  | | --- | | *Волков Н.А.* | |
|  | (Группа) | | (Подпись) | (Дата) | (И.О. Фамилия) |
| Преподаватель | | |  |  | *Маломан Ю. С.* |
|  | |  | (Подпись) | (Дата) | (И.О. Фамилия) |

Архангельск 2024

# **Практическая работа №4**

**Разработка веб-клиента**

1. **Цель работы** 
   1. Научиться разрабатывать клиентское веб-приложение для доступа к БД.
   2. Научиться проверять работоспособность RESTful API в клиентском приложении.
2. **Контрольные вопросы**

Вопрос: 4.1 Как выполнить HTTP-запрос к API с помощью Fetch API?

Для выполнения HTTP-запроса к API с помощью Fetch API можно использовать следующий код:

fetch('https://api.example.com/data', {

method: 'GET',

headers: {

'Content-Type': 'application/json',

'Authorization': 'Bearer token'

}

})

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Ошибка:', error));

Ключевые моменты:

- Используется метод fetch()

- Указывается URL API и опции запроса в объекте второго аргумента

- Метод then() используется для обработки ответа

- Метод catch() для обработки ошибок

Вопрос: Как обрабатывать ошибки при вызове API в JavaScript?

Для обработки ошибок можно использовать следующий код:

fetch('https://api.example.com/data')

.then(response => {

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

}

return response.json();

})

.then(data => console.log(data))

.catch(error => console.error('Ошибка:', error));

Ключевые моменты:

- Проверка статуса ответа с помощью response.ok

- Выбрасывание исключения при неудачном статусе

- Обработка ошибок с помощью catch()

Вопрос: Что такое асинхронные функции и как они связаны с вызовом API?

Асинхронные функции позволяют выполнять операции без блокировки основного потока выполнения. Они связаны с вызовом API тем, что многие операции с API (например, сетевые запросы) являются асинхронными.

Пример асинхронной функции:

async function fetchData() {

try {

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

const data = await response.json();

console.log(data);

} catch (error) {

console.error('Ошибка:', error);

}

}

fetchData();

Ключевые моменты:

- Использование ключевого слова async

- Оператор await для ожидания асинхронных операций

- Обработка ошибок с помощью try/catch

Вопрос: Что такое Razor Pages?

Razor Pages - это модель разработки в ASP.NET Core, которая позволяет создавать веб-страницы с использованием C# и HTML. Она предоставляет простой способ создания веб-страниц без необходимости использования контроллеров.

Основные характеристики:

- Использование C# для логики и HTML для представления

- Автоматическое маршрутизирование на основе URL

- Встроенная поддержка AJAX

Вопрос: Как создать и настроить новую страницу Razor в ASP.NET Core?

Для создания новой страницы Razor:

1. Создайте новый класс PageModel, наследующий от PageModel:

public class NewPageModel : PageModel

{

public void OnGet()

{

// Логика загрузки страницы

}

public IActionResult OnPost()

{

// Логика обработки POST-запроса

return Page();

}

}

2. Создайте файл Razor (.cshtml) в папке Pages с соответствующим именем:

@page

@model YourNamespace.NewPageModel

<h1>Новая страница</h1>

<form method="post">

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

</form>

3. Настройте маршрутизацию в Startup.cs:

app.MapRazorPages();

Ключевые моменты:

- Использование @page директивы в файле Razor

- Наследование от PageModel

- Определение методов OnGet() и OnPost() для обработки запросов

- Простота создания и структурирования кода

Этот ответ охватывает основные аспекты работы с API в JavaScript, обработки ошибок, асинхронных функций, Razor Pages и создания новых страниц Razor в ASP.NET Core.

1. **Вывод**
   1. В ходе лабораторной работы мы научились разрабатывать клиентское веб-приложение для доступа к БД.
   2. Научились проверять работоспособность RESTful API в клиентском приложении.

.