**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет Информационных технологий**

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № \_14\_**

**Дисциплина:** \_Backend разработка

**Тема:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Выполнил(а): студент(ка) группы \_\_231-336\_\_**

\_\_\_\_\_\_\_\_Канищев И.М\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О.)

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания:**

**Москва**

**2025**

### 1. Создание базового веб-приложения ASP.NET Core

Для начала создадим простое веб-приложение с API контроллером, который будет возвращать данные.

### 2. Настройка CORS политик

В ASP.NET Core настройка CORS выполняется в классе Program.cs с помощью методов:

* AddCors() - для добавления служб CORS
* UseCors() - для включения middleware CORS

### 3. Реализация различных политик CORS

Создадим несколько политик с разными уровнями доступа:

* Разрешение всех источников, методов и заголовков
* Ограниченная политика с указанием конкретных доменов
* Политика с учетными данными

### 4. Тестирование CORS

Протестируем работу CORS с помощью:

* Postman для отправки запросов
* Простого HTML-файла для симуляции кросс-доменного запроса
* Проверки заголовков ответа

## **Реализация**

### Код приложения

**Program.cs**

using Microsoft.AspNetCore.Cors;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// Configure CORS policies

builder.Services.AddCors(options =>

{

// Policy 1: Allow everything (for development)

options.AddPolicy("AllowAll", policy =>

{

policy.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader();

});

// Policy 2: Restricted policy with specific origins

options.AddPolicy("Restricted", policy =>

{

policy.WithOrigins("https://localhost:3000", "https://example.com")

.WithMethods("GET", "POST")

.WithHeaders("Content-Type", "Authorization")

.AllowCredentials();

});

// Policy 3: Named policy for specific API

options.AddPolicy("ApiPolicy", policy =>

{

policy.WithOrigins("https://api.example.com")

.WithMethods("GET", "POST", "PUT", "DELETE")

.WithHeaders("Content-Type", "Authorization", "X-API-Key");

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

// Use AllowAll policy in development

app.UseCors("AllowAll");

}

else

{

// Use more restrictive policy in production

app.UseCors("Restricted");

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Controllers/UsersController.cs**

using Microsoft.AspNetCore.Cors;

using Microsoft.AspNetCore.Mvc;

namespace CorsWebApp.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class UsersController : ControllerBase

{

private static readonly List<User> \_users = new()

{

new User { Id = 1, Name = "John Doe", Email = "john@example.com" },

new User { Id = 2, Name = "Jane Smith", Email = "jane@example.com" },

new User { Id = 3, Name = "Bob Johnson", Email = "bob@example.com" }

};

// GET: api/users

[HttpGet]

[EnableCors("AllowAll")] // Apply specific policy to this endpoint

public IActionResult GetUsers()

{

return Ok(\_users);

}

// GET: api/users/1

[HttpGet("{id}")]

[EnableCors("Restricted")] // Apply restricted policy

public IActionResult GetUser(int id)

{

var user = \_users.FirstOrDefault(u => u.Id == id);

if (user == null)

return NotFound();

return Ok(user);

}

// POST: api/users

[HttpPost]

[EnableCors("ApiPolicy")] // Apply API policy

public IActionResult CreateUser(User user)

{

if (user == null)

return BadRequest();

user.Id = \_users.Max(u => u.Id) + 1;

\_users.Add(user);

return CreatedAtAction(nameof(GetUser), new { id = user.Id }, user);

}

// PUT: api/users/1

[HttpPut("{id}")]

public IActionResult UpdateUser(int id, User user)

{

var existingUser = \_users.FirstOrDefault(u => u.Id == id);

if (existingUser == null)

return NotFound();

existingUser.Name = user.Name;

existingUser.Email = user.Email;

return Ok(existingUser);

}

}

public class User

{

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public string Email { get; set; } = string.Empty;

}

}

**Properties/launchSettings.json**

{

"profiles": {

"http": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"applicationUrl": "http://localhost:5000",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"https": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"applicationUrl": "https://localhost:7000;http://localhost:5000",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

### Тестовый клиент

**test-client.html**

<!DOCTYPE html>

<html>

<head>

<title>CORS Test Client</title>

<style>

body { font-family: Arial, sans-serif; margin: 20px; }

.container { max-width: 800px; margin: 0 auto; }

.button {

background: #007bff;

color: white;

border: none;

padding: 10px 20px;

margin: 5px;

cursor: pointer;

border-radius: 4px;

}

.button:hover { background: #0056b3; }

.result {

background: #f8f9fa;

border: 1px solid #dee2e6;

padding: 15px;

margin: 10px 0;

border-radius: 4px;

white-space: pre-wrap;

}

.error { color: #dc3545; }

.success { color: #28a745; }

</style>

</head>

<body>

<div class="container">

<h1>CORS Test Client</h1>

<div>

<h3>Test CORS Policies</h3>

<button class="button" onclick="testAllowAll()">Test AllowAll Policy</button>

<button class="button" onclick="testRestricted()">Test Restricted Policy</button>

<button class="button" onclick="testWithoutCors()">Test Without CORS</button>

</div>

<div id="result"></div>

</div>

<script>

const API\_BASE\_URL = 'https://localhost:7000/api';

function displayResult(message, isError = false) {

const resultDiv = document.getElementById('result');

resultDiv.innerHTML = `<div class="result ${isError ? 'error' : 'success'}">${message}</div>`;

}

async function testAllowAll() {

try {

const response = await fetch(`${API\_BASE\_URL}/users`);

const data = await response.json();

let result = `Status: ${response.status}\n`;

result += `CORS Headers:\n`;

result += `- Access-Control-Allow-Origin: ${response.headers.get('Access-Control-Allow-Origin') || 'Not set'}\n`;

result += `- Access-Control-Allow-Methods: ${response.headers.get('Access-Control-Allow-Methods') || 'Not set'}\n`;

result += `Data: ${JSON.stringify(data, null, 2)}`;

displayResult(result);

} catch (error) {

displayResult(`Error: ${error.message}`, true);

}

}

async function testRestricted() {

try {

const response = await fetch(`${API\_BASE\_URL}/users/1`);

const data = await response.json();

let result = `Status: ${response.status}\n`;

result += `CORS Headers:\n`;

result += `- Access-Control-Allow-Origin: ${response.headers.get('Access-Control-Allow-Origin') || 'Not set'}\n`;

result += `Data: ${JSON.stringify(data, null, 2)}`;

displayResult(result);

} catch (error) {

displayResult(`Error: ${error.message}`, true);

}

}

async function testWithoutCors() {

try {

const response = await fetch(`${API\_BASE\_URL}/users/2`);

const data = await response.json();

let result = `Status: ${response.status}\n`;

result += `CORS Headers:\n`;

result += `- Access-Control-Allow-Origin: ${response.headers.get('Access-Control-Allow-Origin') || 'Not set'}\n`;

result += `Data: ${JSON.stringify(data, null, 2)}`;

displayResult(result);

} catch (error) {

displayResult(`Error: ${error.message}`, true);

}

}

</script>

</body>

</html>

## **Тестирование работы приложения**

### 1. Тестирование с помощью Postman

**Запрос GET /api/users (AllowAll policy):**

GET https://localhost:7000/api/users

**Ответ:**

[

{

"id": 1,

"name": "John Doe",

"email": "john@example.com"

},

{

"id": 2,

"name": "Jane Smith",

"email": "jane@example.com"

},

{

"id": 3,

"name": "Bob Johnson",

"email": "bob@example.com"

}

]

**Заголовки ответа:**

Access-Control-Allow-Origin: \*

Access-Control-Allow-Methods: \*

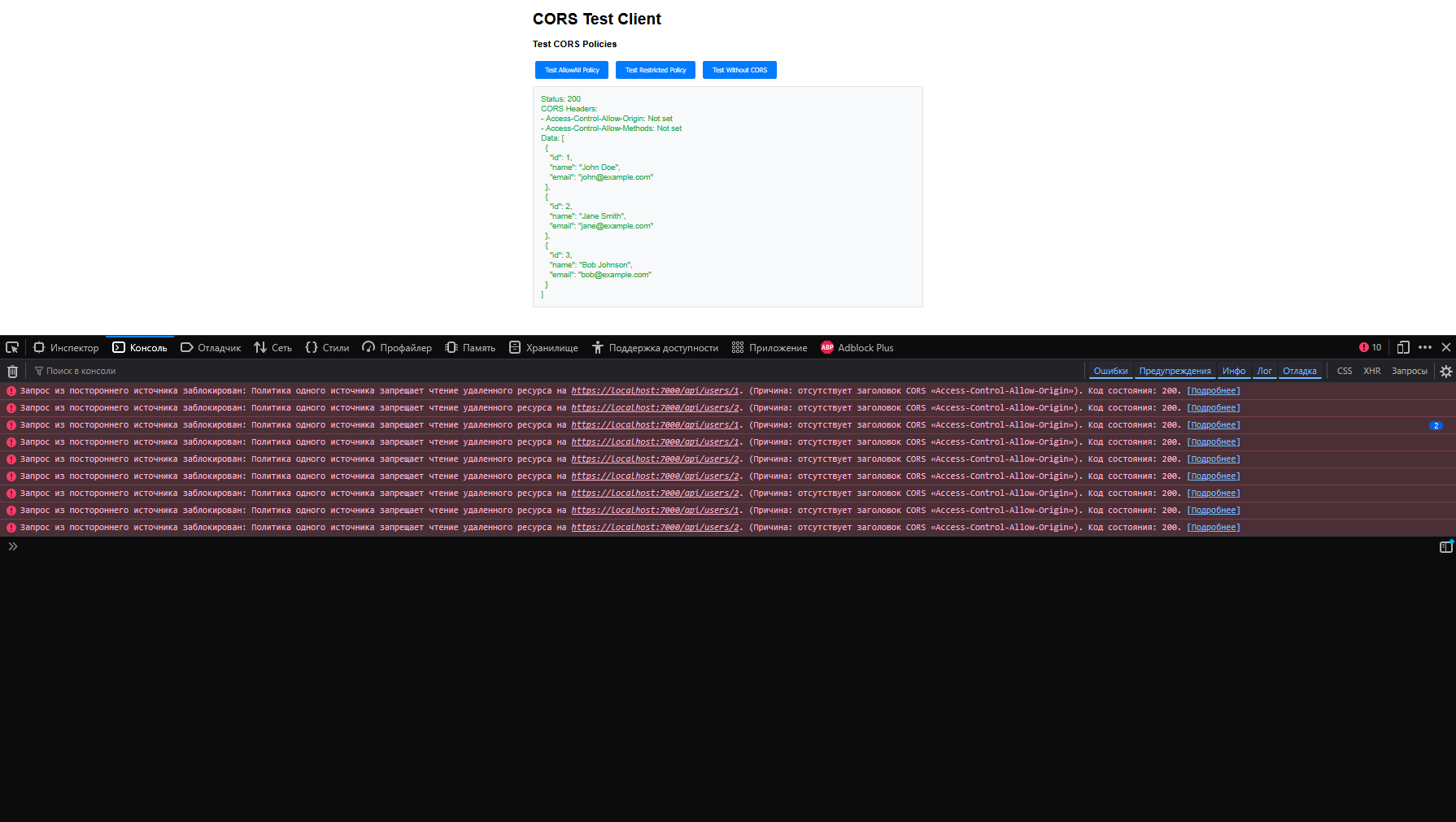
Access-Control-Allow-Headers: \*

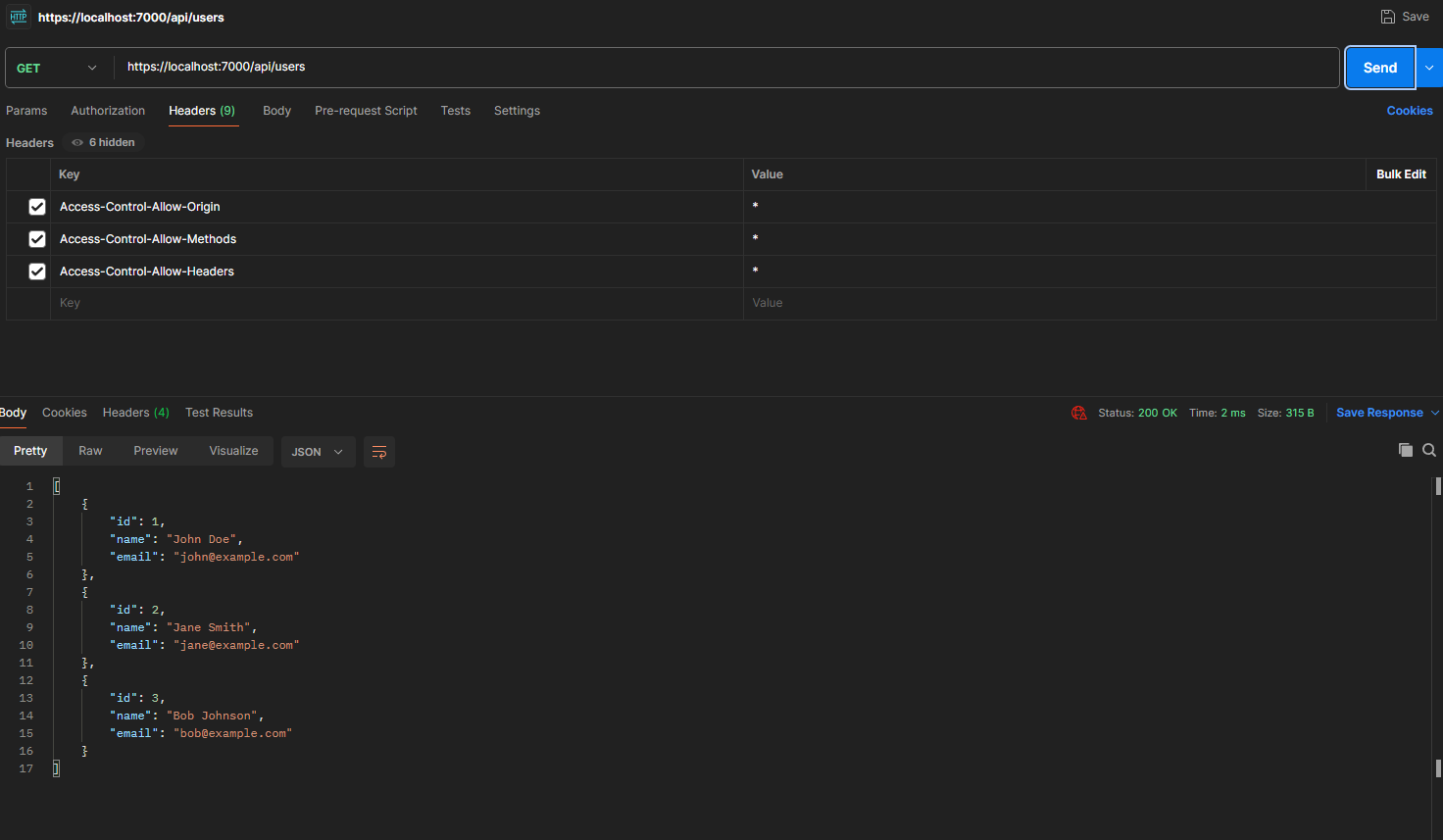
### 2. Тестирование с помощью браузера

При открытии test-client.html и нажатии кнопок можно наблюдать:

* **AllowAll Policy**: Успешный запрос с заголовком Access-Control-Allow-Origin: \*
* **Restricted Policy**: Успешный запрос с конкретным origin
* **Без CORS**: Ошибка CORS при запросе с другого домена

### Скриншоты работы приложения

  
Результаты тестирования CORS политик

  
Тестирование CORS в Postman