**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 № \_6\_**

**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 проекта и структуры конфигурации**

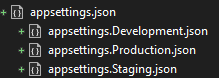
### Структура файлов конфигурации:

appsettings.json (базовая конфигурация)

appsettings.Development.json (для среды разработки)

appsettings.Staging.json (для тестовой среды)

appsettings.Production.json (для продуктивной среды)



### appsettings.json (базовый файл):

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ApplicationSettings": {

"AppName": "My ASP.NET Core App",

"Version": "1.0.0",

"MaxItemsPerPage": 10

},

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\\\mssqllocaldb;Database=AppDb;Trusted\_Connection=true"

},

"FeatureFlags": {

"EnableSwagger": true,

"EnableCaching": false

}

}

### appsettings.Development.json:

{

"Logging": {

"LogLevel": {

"Default": "Debug",

"Microsoft.AspNetCore": "Information"

}

},

"ApplicationSettings": {

"Environment": "Development",

"ApiUrl": "https://localhost:7001",

"DebugMode": true

},

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\\\mssqllocaldb;Database=AppDb\_Dev;Trusted\_Connection=true"

},

"FeatureFlags": {

"EnableSwagger": true,

"EnableCaching": false,

"EnableDetailedErrors": true

}

}

### appsettings.Staging.json:

{

"ApplicationSettings": {

"Environment": "Staging",

"ApiUrl": "https://staging-api.myapp.com",

"DebugMode": false

},

"ConnectionStrings": {

"DefaultConnection": "Server=staging-db;Database=AppDb\_Staging;User Id=appuser;Password=StagingPass123;"

},

"FeatureFlags": {

"EnableSwagger": false,

"EnableCaching": true,

"EnableDetailedErrors": false

}

}

### appsettings.Production.json:

{

"Logging": {

"LogLevel": {

"Default": "Warning",

"Microsoft.AspNetCore": "Error"

}

},

"ApplicationSettings": {

"Environment": "Production",

"ApiUrl": "https://api.myapp.com",

"DebugMode": false

},

"ConnectionStrings": {

"DefaultConnection": "Server=prod-db;Database=AppDb\_Prod;User Id=produser;Password=ProdPass456;"

},

"FeatureFlags": {

"EnableSwagger": false,

"EnableCaching": true,

"EnableDetailedErrors": false

}

}

## **2. Код приложения с использованием конфигурации**

### Program.cs

using Microsoft.AspNetCore.Mvc;

using WebApplicationConfig.Configuration;

using WebApplicationConfig.Services;

var builder = WebApplication.CreateBuilder(args);

// Настройка конфигурации из различных источников

builder.Configuration

.AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)

.AddJsonFile($"appsettings.{builder.Environment.EnvironmentName}.json", optional: true, reloadOnChange: true)

.AddEnvironmentVariables() // Добавляем переменные среды

.AddCommandLine(args); // Добавляем аргументы командной строки

// Регистрация сервисов

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

// Конфигурация Swagger на основе FeatureFlags

var featureFlags = builder.Configuration.GetSection("FeatureFlags").Get<FeatureFlags>();

if (featureFlags?.EnableSwagger == true)

{

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new() { Title = "WebApplicationConfig API", Version = "v1" });

});

}

// Регистрация кастомных сервисов

builder.Services.Configure<ApplicationSettings>(builder.Configuration.GetSection("ApplicationSettings"));

builder.Services.Configure<FeatureFlags>(builder.Configuration.GetSection("FeatureFlags"));

builder.Services.AddSingleton<IConfigurationService, ConfigurationService>();

var app = builder.Build();

// Конфигурация pipeline в зависимости от среды

if (app.Environment.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

if (featureFlags?.EnableSwagger == true)

{

app.UseSwagger();

app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "WebApplicationConfig v1"));

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

// Endpoint для отображения текущей конфигурации

app.MapGet("/config", (IConfiguration config, IConfigurationService configService) =>

{

return configService.GetConfigurationInfo();

});

app.Run();

### Модели конфигурации (Configuration/ApplicationSettings.cs)

namespace WebApplicationConfig.Configuration;

public class ApplicationSettings

{

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

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

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

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

public bool DebugMode { get; set; }

public int MaxItemsPerPage { get; set; } = 10;

}

public class FeatureFlags

{

public bool EnableSwagger { get; set; }

public bool EnableCaching { get; set; }

public bool EnableDetailedErrors { get; set; }

}

### Сервис для работы с конфигурацией (Services/ConfigurationService.cs)

using Microsoft.Extensions.Options;

using WebApplicationConfig.Configuration;

namespace WebApplicationConfig.Services;

public interface IConfigurationService

{

string GetConfigurationInfo();

ApplicationSettings GetApplicationSettings();

FeatureFlags GetFeatureFlags();

}

public class ConfigurationService : IConfigurationService

{

private readonly ApplicationSettings \_appSettings;

private readonly FeatureFlags \_featureFlags;

private readonly IConfiguration \_configuration;

public ConfigurationService(

IOptions<ApplicationSettings> appSettings,

IOptions<FeatureFlags> featureFlags,

IConfiguration configuration)

{

\_appSettings = appSettings.Value;

\_featureFlags = featureFlags.Value;

\_configuration = configuration;

}

public string GetConfigurationInfo()

{

return $"""

Application: {\_appSettings.AppName}

Version: {\_appSettings.Version}

Environment: {\_appSettings.Environment}

API URL: {\_appSettings.ApiUrl}

Debug Mode: {\_appSettings.DebugMode}

Max Items Per Page: {\_appSettings.MaxItemsPerPage}

Features:

- Swagger: {\_featureFlags.EnableSwagger}

- Caching: {\_featureFlags.EnableCaching}

- Detailed Errors: {\_featureFlags.EnableDetailedErrors}

Connection String: {\_configuration.GetConnectionString("DefaultConnection")}

""";

}

public ApplicationSettings GetApplicationSettings()

{

return \_appSettings;

}

public FeatureFlags GetFeatureFlags()

{

return \_featureFlags;

}

}

### Контроллер (Controllers/WeatherForecastController.cs)

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Options;

using WebApplicationConfig.Configuration;

using WebApplicationConfig.Services;

namespace WebApplicationConfig.Controllers;

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

private readonly ApplicationSettings \_appSettings;

private readonly FeatureFlags \_featureFlags;

private readonly IConfigurationService \_configService;

public WeatherForecastController(

ILogger<WeatherForecastController> logger,

IOptions<ApplicationSettings> appSettings,

IOptions<FeatureFlags> featureFlags,

IConfigurationService configService)

{

\_logger = logger;

\_appSettings = appSettings.Value;

\_featureFlags = featureFlags.Value;

\_configService = configService;

}

[HttpGet]

public IActionResult Get()

{

// Логирование в зависимости от конфигурации

if (\_appSettings.DebugMode)

{

\_logger.LogInformation("WeatherForecast endpoint called in debug mode");

}

var forecasts = Enumerable.Range(1, \_appSettings.MaxItemsPerPage).Select(index => new WeatherForecast

{

Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = Summaries[Random.Shared.Next(Summaries.Length)]

})

.ToArray();

// Добавляем информацию о конфигурации в ответ

var response = new

{

Environment = \_appSettings.Environment,

AppName = \_appSettings.AppName,

MaxItems = \_appSettings.MaxItemsPerPage,

Forecasts = forecasts

};

return Ok(response);

}

[HttpGet("config")]

public IActionResult GetConfig()

{

var configInfo = \_configService.GetConfigurationInfo();

return Ok(configInfo);

}

[HttpGet("features")]

public IActionResult GetFeatures()

{

return Ok(\_featureFlags);

}

}

public record WeatherForecast

{

public DateOnly Date { get; set; }

public int TemperatureC { get; set; }

public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);

public string? Summary { get; set; }

}

## **3. Тестирование приложения**

## Настройка launchSettings.json

Файл Properties/launchSettings.json определяет профили запуска для разных сред:

{

"profiles": {

"WebApplicationConfig (Development)": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"applicationUrl": "https://localhost:7001;http://localhost:5001",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development",

"Custom\_\_ApiKey": "dev-key-123",

"ConnectionStrings\_\_DefaultConnection": "Server=local-dev;Database=AppDb\_Dev"

}

},

"WebApplicationConfig (Staging)": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"applicationUrl": "https://localhost:7002;http://localhost:5002",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Staging",

"Custom\_\_ApiKey": "staging-key-456",

"ConnectionStrings\_\_DefaultConnection": "Server=staging-sql;Database=AppDb\_Staging"

}

},

"WebApplicationConfig (Production)": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"applicationUrl": "https://localhost:7003;http://localhost:5003",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Production",

"Custom\_\_ApiKey": "prod-key-789",

"ConnectionStrings\_\_DefaultConnection": "Server=prod-sql;Database=AppDb\_Production"

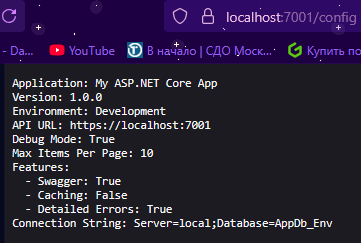
}

}

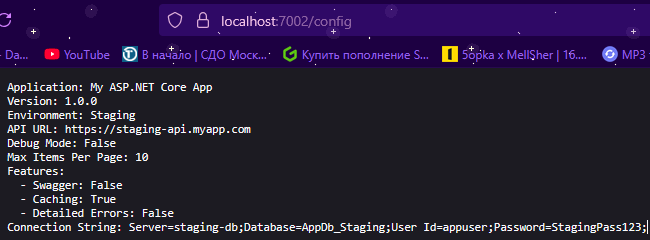
}

}

1. **Запустите в Development среде:**
   * Выберите "WebApplicationConfig (Development)"
   * Нажмите F5
   * Откройте в браузере: https://localhost:7001/config
   * Проверьте, что отображаются настройки Development



1. **Запустите в Staging среде:**
   * Выберите "WebApplicationConfig (Staging)"
   * Нажмите F5
   * Откройте: https://localhost:7002/config
   * Убедитесь, что настройки изменились на Staging



1. **Запустите в Production среде:**
   * Выберите "WebApplicationConfig (Production)"
   * Нажмите F5
   * Откройте: https://localhost:7003/config
   * Проверьте Production настройки

### 