# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **КНІТ**

Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 4

**З дисципліни:** *“Конструювання програмного забезпечення”*

**На тему:** *“Лабораторна WebApi”*

**Лектор:** доц. каф. ПЗ Сердюк П. В.

**Виконав:** ст. гр. ПЗ-35 Хруставчук М.Л.

**Прийняв:** асистент каф. ПЗ Яценко Р. О.

« » 2024 р.

∑= .

Львів – 2024

**Тема роботи:** Лабораторна WebApi.

**Мета роботи:** Навчитися створювати Web Api та використовувати програми Postman та Swagger для відслідковування запитів.

# ЗАВДАННЯ

* Створити Web Api за допомогою ASP.Net. Реалізувати CRUD контролери для роботи з таблицями лабораторної роботи № 9 (Не менше 2 контролерів).
* Продемонструвати роботу контролерів за допомогою програми Postman (обов’язково у форматах JSON).
* Підключити документацію Api за допомогою Swagger. Продемонструвати виконання запитів у Swagger.
* Використати Dependency Injection в контролерах.
* Створити моделі. Створити ViewModels і мапінг до моделей.

Варіант БД: мережа спортивних комплексів.

# ХІД ВИКОНАННЯ

**1. Код розробленої програми**

Program.cs

using AutoMapper;

using Lab4.Abstraction.IRepository;

using Lab4.Abstraction.IServices;

using Lab4.BLL.Services;

using Lab4.DAL.Data;

using Lab4.DAL.Repositories;

using Microsoft.EntityFrameworkCore;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Реєстрація контролерів

builder.Services.AddControllers();

// Реєстрація AutoMapper

var mapperConfig = new MapperConfiguration(mc =>

{

mc.AddProfile(new MappingProfile()); // Використовуйте ваш профіль мапінгу

});

IMapper mapper = mapperConfig.CreateMapper();

builder.Services.AddSingleton(mapper);

//// Реєстрація Swagger

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "SportComplex API",

Version = "v1",

Description = "API для управління спортивним комплексом.",

Contact = new OpenApiContact

{

Name = "Максим Мангуст",

}

});

//var xmlFile = $"{System.Reflection.Assembly.GetExecutingAssembly().GetName().Name}.xml";

//var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);

//c.IncludeXmlComments(xmlPath);

//c.DocumentFilter<SwaggerControllerSorter>();

});

// Реєстрація DbContext

builder.Services.AddDbContext<SportComplexContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddScoped<IClientRepository, ClientRepository>();

builder.Services.AddScoped<IClientService, ClientService>();

builder.Services.AddScoped<IActivityTypeRepository, ActivityTypeRepository>();

builder.Services.AddScoped<IActivityTypeService, ActivityTypeService>();

builder.Services.AddScoped<IPurchaseRepository, PurchaseRepository>();

builder.Services.AddScoped<IPurchaseService, PurchaseService>();

builder.Services.AddScoped<ISubscriptionRepository, SubscriptionRepository>();

builder.Services.AddScoped<ISubscriptionService, SubscriptionService>();

builder.Services.AddScoped<ISubscriptionTermRepository, SubscriptionTermRepository>();

builder.Services.AddScoped<ISubscriptionTermService, SubscriptionTermService>();

builder.Services.AddScoped<IPaymentMethodRepository, PaymentMethodRepository>();

builder.Services.AddScoped<IPaymentMethodService, PaymentMethodService>();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

Purchase.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Lab4.DAL.Models

{

public class Purchase

{

[Key]

public int purchase\_id { get; set; }

[ForeignKey("Client")]

public int client\_id { get; set; }

public Client Client { get; set; }

[ForeignKey("Subscription")]

public int subscription\_id { get; set; }

public Subscription Subscription { get; set; }

[ForeignKey("PaymentMethod")]

public int payment\_method\_id { get; set; }

public PaymentMethod PaymentMethod { get; set; }

[Range(1, int.MaxValue)]

public int purchase\_number { get; set; }

public DateTime purchase\_date { get; set; } = DateTime.Now;

}

}

PurchasesController.cs

using Lab4.Abstraction.IServices;

using Lab4.Abstraction.ViewModels;

using Microsoft.AspNetCore.Mvc;

namespace TodoApi.Controllers

{

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

[ApiController]

public class PurchasesController : ControllerBase

{

private readonly IPurchaseService \_service;

public PurchasesController(IPurchaseService service)

{

\_service = service;

}

[HttpGet]

public async Task<ActionResult<IEnumerable<PurchaseViewModel>>> GetPurchases()

{

return Ok(await \_service.GetAllPurchasesAsync());

}

[HttpGet("{id}")]

public async Task<ActionResult<PurchaseViewModel>> GetPurchase(int id)

{

var purchase = await \_service.GetPurchaseByIdAsync(id);

if (purchase == null)

{

return NotFound();

}

return Ok(purchase);

}

[HttpPost]

public async Task<ActionResult> PostPurchase(PurchaseViewModel purchaseViewModel)

{

await \_service.AddPurchaseAsync(purchaseViewModel);

return CreatedAtAction(nameof(GetPurchase), new { id = purchaseViewModel.Id }, purchaseViewModel);

}

[HttpPut("{id}")]

public async Task<IActionResult> PutPurchase(int id, PurchaseViewModel purchaseViewModel)

{

if (id != purchaseViewModel.Id)

{

return BadRequest();

}

await \_service.UpdatePurchaseAsync(purchaseViewModel);

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeletePurchase(int id)

{

await \_service.DeletePurchaseAsync(id);

return NoContent();

}

}

}

PurchaseRepository.cs

using Lab4.Abstraction.IRepository;

using Lab4.Abstraction.ViewModels;

using Lab4.DAL.Data;

using Lab4.DAL.Models;

using Microsoft.EntityFrameworkCore;

namespace Lab4.DAL.Repositories

{

public class PurchaseRepository : IPurchaseRepository

{

private readonly SportComplexContext \_context;

public PurchaseRepository(SportComplexContext context)

{

\_context = context;

}

public async Task<IEnumerable<PurchaseViewModel>> GetAllAsync()

{

return await \_context.Purchases

.Include(p => p.Client)

.Include(p => p.PaymentMethod)

.Select(p => new PurchaseViewModel

{

Id = p.purchase\_id,

ClientFullName = p.Client.client\_full\_name,

SubscriptionId = p.subscription\_id,

PaymentMethodName = p.PaymentMethod.payment\_method,

PurchaseNumber = p.purchase\_number,

PurchaseDate = p.purchase\_date

})

.ToListAsync();

}

public async Task<PurchaseViewModel> GetByIdAsync(int id)

{

var purchase = await \_context.Purchases

.Include(p => p.Client)

.Include(p => p.PaymentMethod)

.FirstOrDefaultAsync(p => p.purchase\_id == id);

if (purchase == null) return null;

return new PurchaseViewModel

{

Id = purchase.purchase\_id,

ClientFullName = purchase.Client.client\_full\_name,

SubscriptionId = purchase.subscription\_id,

PaymentMethodName = purchase.PaymentMethod.payment\_method,

PurchaseNumber = purchase.purchase\_number,

PurchaseDate = purchase.purchase\_date

};

}

public async Task AddAsync(PurchaseViewModel purchaseViewModel)

{

var purchase = new Purchase

{

subscription\_id = purchaseViewModel.SubscriptionId,

purchase\_number = purchaseViewModel.PurchaseNumber,

purchase\_date = purchaseViewModel.PurchaseDate

};

\_context.Purchases.Add(purchase);

await \_context.SaveChangesAsync();

}

public async Task UpdateAsync(PurchaseViewModel purchaseViewModel)

{

var purchase = await \_context.Purchases.FindAsync(purchaseViewModel.Id);

if (purchase == null) return;

purchase.subscription\_id = purchaseViewModel.SubscriptionId;

purchase.purchase\_number = purchaseViewModel.PurchaseNumber;

purchase.purchase\_date = purchaseViewModel.PurchaseDate;

\_context.Entry(purchase).State = EntityState.Modified;

await \_context.SaveChangesAsync();

}

public async Task DeleteAsync(int id)

{

var purchase = await \_context.Purchases.FindAsync(id);

if (purchase != null)

{

\_context.Purchases.Remove(purchase);

await \_context.SaveChangesAsync();

}

}

}

}

SportComplexContext.cs

using Microsoft.EntityFrameworkCore;

using Lab4.DAL.Models;

namespace Lab4.DAL.Data

{

public class SportComplexContext : DbContext

{

public SportComplexContext(DbContextOptions<SportComplexContext> options) : base(options)

{

}

public DbSet<PaymentMethod> PaymentMethods { get; set; }

public DbSet<City> Cities { get; set; }

public DbSet<SubscriptionTerm> SubscriptionTerms { get; set; }

public DbSet<SubscriptionVisitTime> SubscriptionVisitTimes { get; set; }

public DbSet<SportComplex> SportComplexes { get; set; }

public DbSet<Gym> Gyms { get; set; }

public DbSet<Trainer> Trainers { get; set; }

public DbSet<Client> Clients { get; set; }

public DbSet<Subscription> Subscriptions { get; set; }

public DbSet<Purchase> Purchases { get; set; }

public DbSet<ActivityType> ActivityTypes { get; set; }

public DbSet<Training> Trainings { get; set; }

public DbSet<TrainerActivity> TrainerActivities { get; set; }

public DbSet<SubscriptionActivityType> SubscriptionActivityTypes { get; set; }

}

}

PurchaseService.cs

using Lab4.Abstraction.IServices;

using Lab4.Abstraction.IRepository;

using Lab4.Abstraction.ViewModels;

using Lab4.DAL.Repositories;

namespace Lab4.BLL.Services

{

public class PurchaseService : IPurchaseService

{

private readonly IPurchaseRepository \_repository;

public PurchaseService(IPurchaseRepository repository)

{

\_repository = repository;

}

public async Task<IEnumerable<PurchaseViewModel>> GetAllPurchasesAsync()

{

return await \_repository.GetAllAsync();

}

public async Task<PurchaseViewModel> GetPurchaseByIdAsync(int id)

{

return await \_repository.GetByIdAsync(id);

}

public async Task AddPurchaseAsync(PurchaseViewModel purchaseViewModel)

{

await \_repository.AddAsync(purchaseViewModel);

}

public async Task UpdatePurchaseAsync(PurchaseViewModel purchaseViewModel)

{

await \_repository.UpdateAsync(purchaseViewModel);

}

public async Task DeletePurchaseAsync(int id)

{

await \_repository.DeleteAsync(id);

}

}

}

**2. Результати виконання розробленої програми**

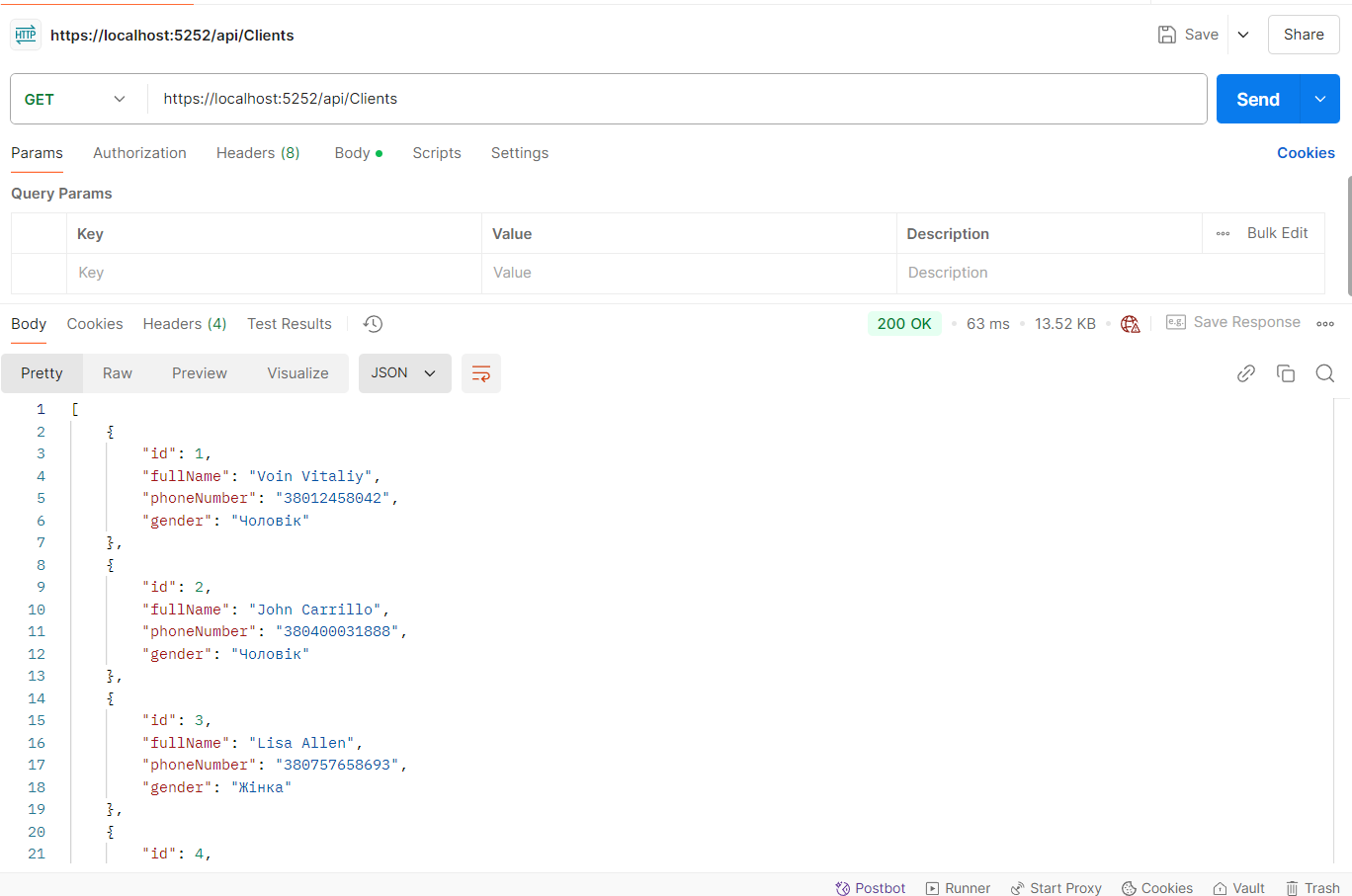
****

Рис. 1. Демонстрація GET-запиту

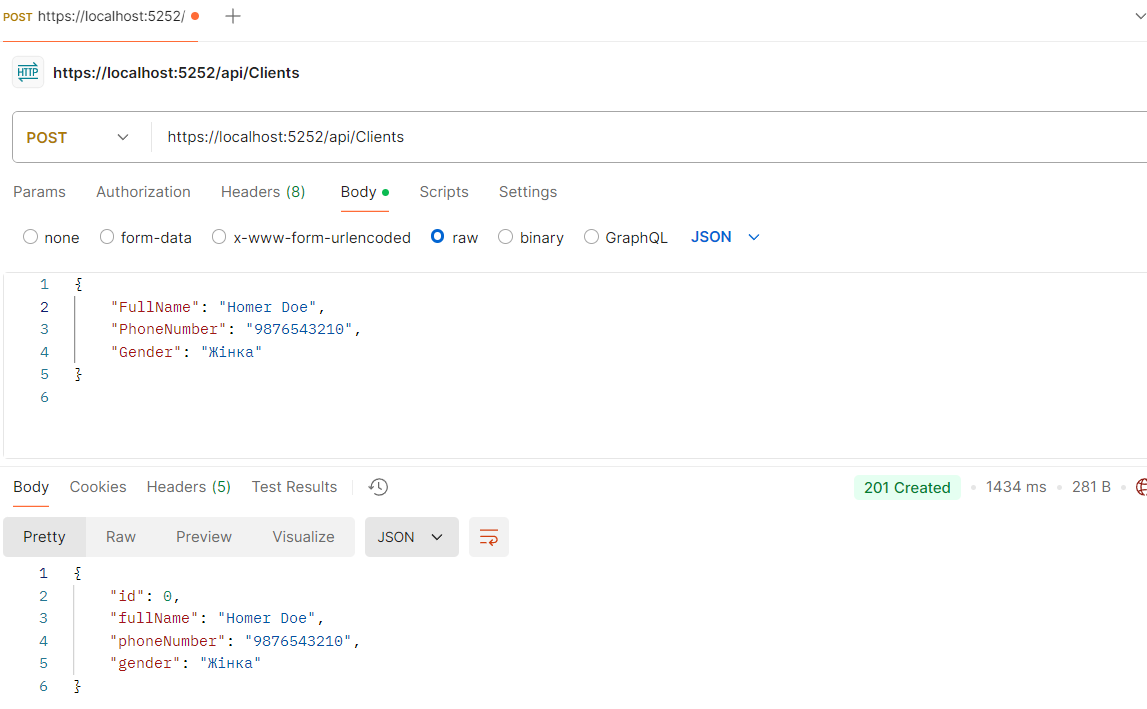
****

Рис. 2. Демонстрація POST-запиту

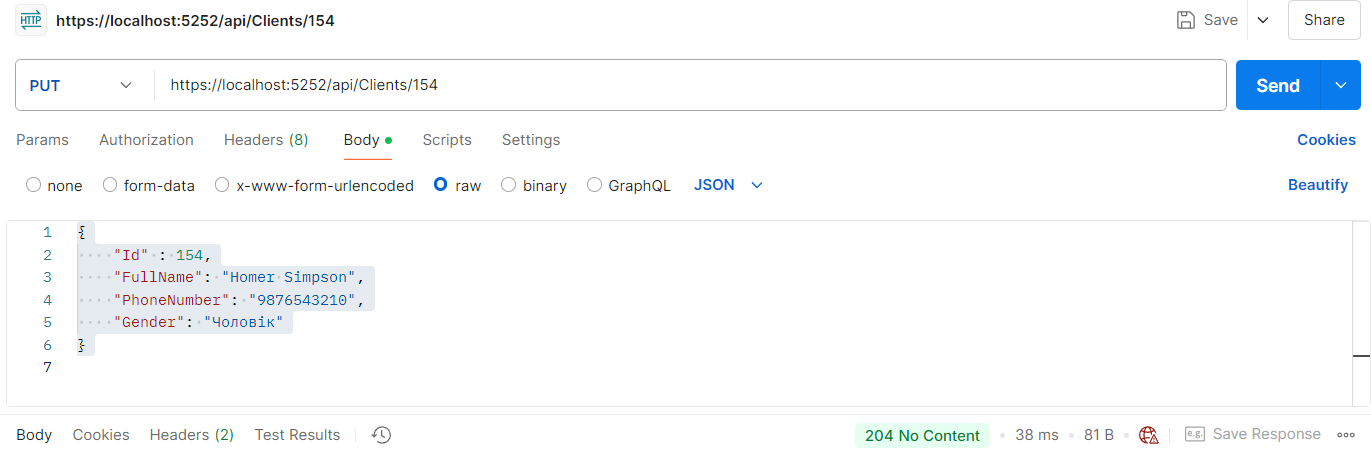


Рис. 3. Демонстрація PUT-запиту

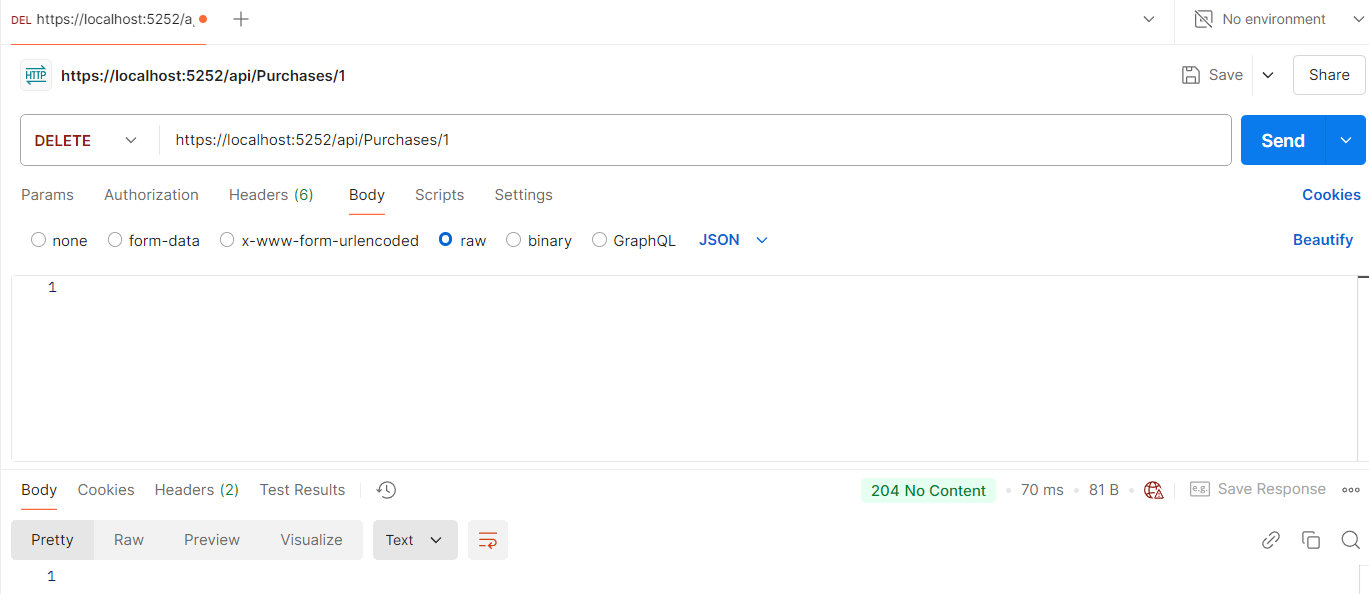


Рис. 4. Демонстрація DELETE-запиту

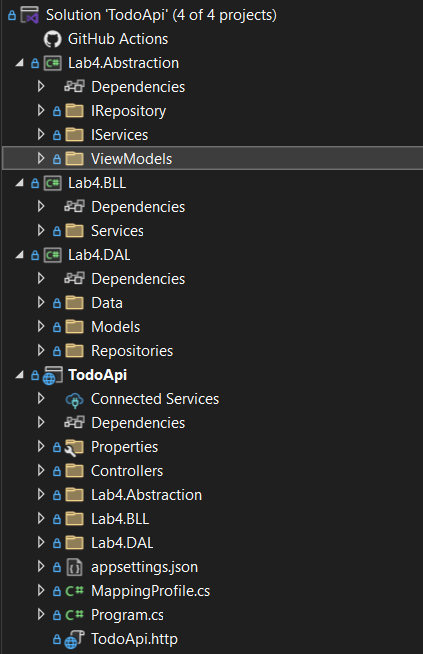
****

Рис. 5. Файлова архітектура проєкту

# ВИСНОВКИ

У цій лабораторній роботі я навчився створювати Web Api та використовувати програми Postman та Swagger для відслідковування запитів. Також реалізував шарову архітектуру для Api, яка складається з чотирьох окремих шарів, що забезпечують чіткий розподіл обов'язків між частинами системи. Це дозволяє легше розширювати функціонал, спрощує тестування та підтримку проєкту.