Лабораторна робота №2

**Тема: Робота з ASP.NET MVC. Конфігурація проекту. Відображення даних з БД на веб-сторінках.**

**Мета:** Ознайомитися з архітектурою MVC, набути навичок створення головних компонентів архітектури MVC, навчитися конфігурувати підключення до БД та виконувати запити на читання даних, набути навичок роботи з представленнями (Views).

Тема проекту: Система для прокату транспорту(Варіант 7)

Завдання 1:

Виконав необхідні команди:

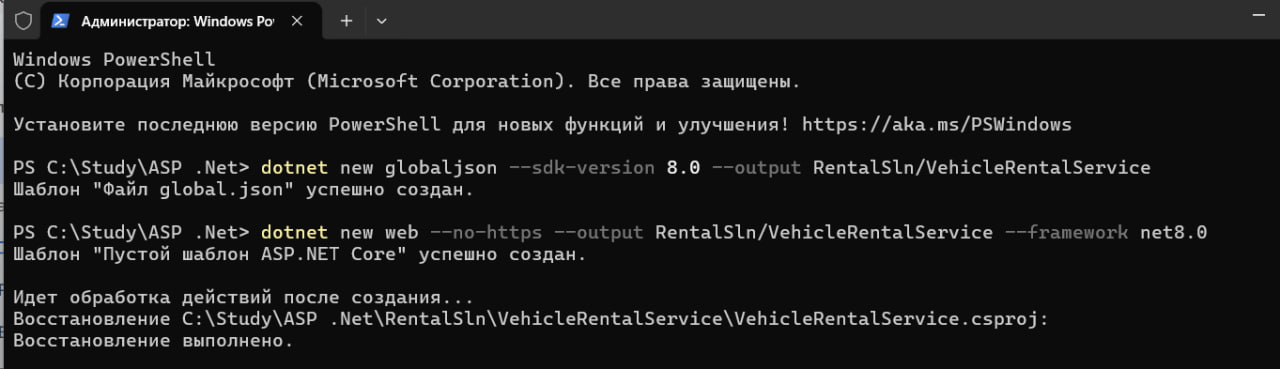


Рис.1.Команди для створення проекту

Структура проекту:

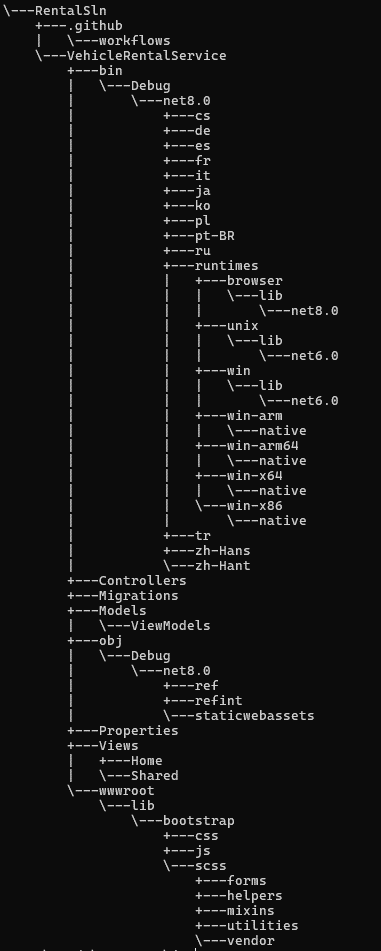


Рис.2.Структура проекту

Лістінг launchsettings.json

{

"$schema": "http://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

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

"sslPort": 0

}

},

"profiles": {

"http": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

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

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

Остаточний лістінг Program.cs

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Migrations;

using VehicleRentalService.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

builder.Services.AddDbContext<ServiceDbContext>(opts =>

{

opts.UseSqlServer(builder.Configuration["ConnectionStrings:VehicleRentalServiceConnection"]);

});

builder.Services.AddScoped<IServiceRepository, EFServiceRepository>();

var app = builder.Build();

app.UseStaticFiles();

app.MapDefaultControllerRoute();

SeedData.EnsurePopulated(app);

app.Run();

HomeController(на момент виконання завдання 1):

﻿using Microsoft.AspNetCore.Mvc;

namespace VehicleRentalService.Controllers

{

public class HomeController:Controller

{

public IActionResult Index() => View();

}

}

Перейдемо до сутностей нашого проекту:

Основна сутність-абстрактний клас Vehicle, який наслідує кожен клас, відповідаючий за основні властивості та методи, яки повинні належати ТЗ:

using System.ComponentModel.DataAnnotations.Schema;

namespace VehicleRentalService.Models

{

public abstract class Vehicle

{

public long? VehicleId { get; set; }

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

public string? Description { get; set; } = string.Empty;

[Column(TypeName = "decimal(8, 2)")]

public decimal PricePerMinute { get; set; }

public bool IsAvailable { get; set; }

public string? Location { get; set; }

}

}

Далі йдуть класи, які відповідають за транспортні засоби, такі як самокати, автомобілі та велосипеди:

Клас Car:

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace VehicleRentalService.Models

{

public enum TransmissionType

{

Manual,

Automatic,

CVT,

SemiAutomatic

}

public class Car : Vehicle

{

private double \_maxFuel;

public double MaxFuel

{

get => \_maxFuel;

set

{

if (value < 1 || value > 70)

throw new ArgumentOutOfRangeException(nameof(MaxFuel), "MaxFuel must be in range from 1 to 70");

\_maxFuel = value;

}

}

private double \_currentFuel;

public double CurrentFuel

{

get => \_currentFuel;

set

{

if (value < 0 || value > MaxFuel)

throw new ArgumentOutOfRangeException(nameof(CurrentFuel), $"CurrentFuel must be in range from 0 to MaxFuel ({MaxFuel})");

\_currentFuel = value;

}

}

public TransmissionType Transmission { get; set; }

}

}

Клас Bike:

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace VehicleRentalService.Models

{

public class Bike:Vehicle

{

}

}

Клас Scooter:

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace VehicleRentalService.Models

{

public class Scooter : Vehicle

{

private int \_charge;

[Range(0, 100)]

public int Charge

{

get => \_charge;

set

{

if (value >= 0 && value <= 100)

{

\_charge = value;

}

}

}

}

}

Також в нас є клас, відповідаючий за поїздки, клас Trip:

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace VehicleRentalService.Models

{

public class Trip

{

public long Id { get; set; }

public long VehicleId { get; set; }

public Vehicle Vehicle { get; set; } = null!;

public DateTime TripBegin { get; set; }

private DateTime \_tripEnd;

public DateTime TripEnd

{

get => \_tripEnd;

set

{

if (value >= TripBegin)

{

\_tripEnd = value;

}

else

{

throw new Exception("Trip end can't be earlier than Trip begin");

}

}

}

[NotMapped]

public TimeSpan Duration => TripEnd - TripBegin;

[NotMapped]

public decimal TripPrice

{

get

{

if (Vehicle == null)

throw new InvalidOperationException("Vehicle is not loaded. Please use include during the request"); //юзать инклюд

var minutes = (decimal)Duration.TotalMinutes;

return Math.Round(minutes \* Vehicle.PricePerMinute, 2);

}

}

}

}

Також в наявності є клас, відповідаючий за збереження новин нашого сервісу оренди:

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace VehicleRentalService.Models

{

public class News

{

public long Id { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public DateTime PublicationDate { get; set; } = DateTime.Now;

}

}

Після цього ми створили файл Index.cshtml, туди додали код для виведення привітання.

@page

@model VehicleRentalService.Views.Home.IndexModel

@{

<h4>Welcome to Vehicle Rental Service!</h4>

}

Завдання 2

Першим кроком було встановлено Entity Framework до проекту та також налаштовано його.

Команди для встановлення Entity Framework

dotnet add package Microsoft.EntityFrameworkCore.Design

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet tool install --global dotnet-ef

Далі йде файл налаштуваннь ORM appsettings.json:

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"VehicleRentalServiceConnection": "Server=(localdb)\\MSSQLLocalDB;Database=VehicleRentalService;MultipleActiveResultSets=true"

}

}

Тут ми налаштували БД. Під’єднали її.

Далі розробили клас ServiceDBContext, щоб у EF був доступ до БД.

using Microsoft.EntityFrameworkCore;

namespace VehicleRentalService.Models

{

public class ServiceDbContext : DbContext

{

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

public DbSet<Car> Cars => Set<Car>();

public DbSet<Scooter> Scooters => Set<Scooter>();

public DbSet<Bike> Bikes => Set<Bike>();

public DbSet<News> News => Set<News>();

public DbSet<Trip> Trips => Set<Trip>();

}

}

Також розробили інтерфейс IServiceRepository, щоб реалізувати певний паттерн

namespace VehicleRentalService.Models

{

public interface IServiceRepository

{

IQueryable<Car> Cars { get; }

IQueryable<Bike> Bikes { get; }

IQueryable<Scooter> Scooters { get; }

IQueryable<Trip> Trips { get; }

IQueryable<News> News { get; }

}

}

Та реалізація інтерфейсу EFServiceRepository

namespace VehicleRentalService.Models

{

public class EFServiceRepository : IServiceRepository

{

private ServiceDbContext context;

public EFServiceRepository(ServiceDbContext ctx)

{

context = ctx;

}

public IQueryable<Car> Cars => context.Cars;

public IQueryable<Bike> Bikes => context.Bikes;

public IQueryable<Scooter> Scooters => context.Scooters;

public IQueryable<News> News => context.News;

public IQueryable<Trip> Trips => context.Trips;

}

}

Після цього виконаємо міграцію. Міграція створює необхідні таблиці у БД

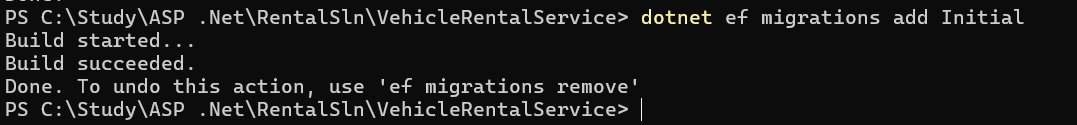


Рис.3.Міграція БД

Після цього наповнимо нашу БД даними. Цей процес називається Seeding.

using Microsoft.EntityFrameworkCore;

using VehicleRentalService.Models;

using static System.Net.Mime.MediaTypeNames;

namespace VehicleRentalService.Models

{

public static class SeedData

{

public static void EnsurePopulated(IApplicationBuilder app)

{

ServiceDbContext context = app.ApplicationServices

.CreateScope().ServiceProvider

.GetRequiredService<ServiceDbContext>();

if (context.Database.GetPendingMigrations().Any())

{

context.Database.Migrate();

}

// ------------------ Cars ------------------

if (!context.Cars.Any())

{

context.Cars.AddRange(

new Car

{

Name = "Toyota Corolla",

Description = "Reliable compact car",

PricePerMinute = 0.50m,

IsAvailable = true,

Location = "Shevchenko St 12, 10000 Zhytomyr",

MaxFuel = 50,

CurrentFuel = 50,

Transmission = TransmissionType.Automatic

},

new Car

{

Name = "Ford Mustang",

Description = "Sporty coupe",

PricePerMinute = 1.20m,

IsAvailable = true,

Location = "Khmelnytskyi St 45, 10001 Zhytomyr",

MaxFuel = 60,

CurrentFuel = 40,

Transmission = TransmissionType.Manual

},

new Car

{

Name = "Tesla Model 3",

Description = "Electric sedan",

PricePerMinute = 2.00m,

IsAvailable = true,

Location = "Kotsiubynskoho St 1, 10002 Zhytomyr",

MaxFuel = 70,

CurrentFuel = 70,

Transmission = TransmissionType.CVT

},

new Car

{

Name = "Daewoo Matiz",

Description = "Best car of the World",

PricePerMinute = 10.00m,

IsAvailable = true,

Location = "Lisna 7, 10001 Zhytomyr",

MaxFuel = 70,

CurrentFuel = 70,

Transmission = TransmissionType.Automatic

}

);

}

// ------------------ Bikes ------------------

if (!context.Bikes.Any())

{

context.Bikes.AddRange(

new Bike

{

Name = "Mountain Bike",

Description = "Off-road bike",

PricePerMinute = 0.15m,

IsAvailable = true,

Location = "Sadova St 7, 10003 Zhytomyr"

},

new Bike

{

Name = "City Bike",

Description = "Comfortable urban bike",

PricePerMinute = 0.10m,

IsAvailable = true,

Location = "Bohdana Khmelnytskoho St 10, 10004 Zhytomyr"

},

new Bike

{

Name = "Racing Bike",

Description = "Lightweight bike for speed",

PricePerMinute = 0.25m,

IsAvailable = true,

Location = "Akademichna St 3, 10005 Zhytomyr"

}

);

}

// ------------------ Scooters ------------------

if (!context.Scooters.Any())

{

context.Scooters.AddRange(

new Scooter

{

Name = "Xiaomi M365",

Description = "Electric scooter",

PricePerMinute = 0.30m,

IsAvailable = true,

Location = "Tupikova St 18, 10006 Zhytomyr",

Charge = 90

},

new Scooter

{

Name = "Segway Ninebot",

Description = "Self-balancing scooter",

PricePerMinute = 0.35m,

IsAvailable = true,

Location = "Svobody St 22, 10007 Zhytomyr",

Charge = 75

},

new Scooter

{

Name = "Boosted Rev",

Description = "High-performance electric scooter",

PricePerMinute = 0.50m,

IsAvailable = true,

Location = "Mira St 99, 10008 Zhytomyr",

Charge = 100

}

);

}

context.SaveChanges();

// ------------------ Trips ------------------

if (!context.Trips.Any())

{

var ids = context.Cars.Select(c => c.VehicleId!.Value)

.Concat(context.Bikes.Select(b => b.VehicleId!.Value))

.Concat(context.Scooters.Select(s => s.VehicleId!.Value))

.ToList();

if (ids.Count >= 3)

{

context.Trips.AddRange(

new Trip

{

VehicleId = ids[0],

TripBegin = DateTime.Now.AddHours(-2),

TripEnd = DateTime.Now.AddHours(-1)

},

new Trip

{

VehicleId = ids[1],

TripBegin = DateTime.Now.AddHours(-1.5),

TripEnd = DateTime.Now.AddHours(-0.5)

},

new Trip

{

VehicleId = ids[2],

TripBegin = DateTime.Now.AddHours(-3),

TripEnd = DateTime.Now.AddHours(-2)

}

);

}

}

// ------------------ News ------------------

if (!context.News.Any())

{

context.News.AddRange(

new News

{

Title = "New Cars Available",

Description = "Check out our latest car models.",

PublicationDate = DateTime.Now.AddDays(-2)

},

new News

{

Title = "Bike Discounts",

Description = "Get up to 20% off on bikes this week.",

PublicationDate = DateTime.Now.AddDays(-1)

},

new News

{

Title = "Scooter Safety Tips",

Description = "Always wear a helmet when riding.",

PublicationDate = DateTime.Now

}

);

}

context.SaveChanges();

}

}

}

Завдання 3 та 4

Після цього зробимо виведення даних на головної сторінці.

Для цього потрібно розробити 2 речі:

1. Виведення даних(як логічну так і візуальну частину)
2. Пейджинг

Почнемо. Розробимо клас VehicleListViewModel:

namespace VehicleRentalService.Models.ViewModels

{

public class VehicleListViewModel

{

public IEnumerable<Vehicle> Vehicles { get; set; }

public PagingInfo PagingInfo { get; set; }

}

}

Він зберігає список ТЗ, та інформацію о пейджинге

Сам клас для пейджингу:

namespace VehicleRentalService.Models.ViewModels

{

public class PagingInfo

{

public int CurrentPage { get; set; }

public int ItemsPerPage { get; set; }

public int TotalItems { get; set; }

public int TotalPages => (int)Math.Ceiling((decimal)TotalItems / ItemsPerPage);

}

}

Він зберігає інформацію, необхідну для реалізації пейджингу

Далі йде HomeController, який і обробляє логіку виведення ТЗ:

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore.Migrations;

using VehicleRentalService.Models;

using VehicleRentalService.Models.ViewModels;

namespace VehicleRentalService.Controllers

{

public class HomeController : Controller

{

private IServiceRepository \_repository;

private int \_pageSize = 5;

public HomeController(IServiceRepository repo)

{

\_repository = repo;

}

public IActionResult Index(int page = 1)

{

var AllVehicles = new List<Vehicle>();

AllVehicles.AddRange(\_repository.Cars);

AllVehicles.AddRange(\_repository.Bikes);

AllVehicles.AddRange(\_repository.Scooters);

var ViewModelVh = new VehicleListViewModel

{

Vehicles = AllVehicles

.OrderBy(v => v.VehicleId)

.Skip((page - 1) \* \_pageSize)

.Take(\_pageSize),

PagingInfo = new PagingInfo

{

CurrentPage = page,

ItemsPerPage = \_pageSize,

TotalItems = AllVehicles.Count()

}

};

return View(ViewModelVh);

}

}

}

Завдяки поліморфізму, ми апкастимо класи ТЗ(такі як Bicycle, Car, Scooter) до класу Vehicle, завдяки чому можемо зберігати їх у одному списку.

Далі розберемось із візуальною частиною

\_Layout.cshtml відповідає за те, щоби не повторювати один ї тей ж код на сторінках. Зазвичай, як і у нашому випадку, там зберігається стиль хедеру та футеру.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>@ViewData["Title"] - VehicleRentalService</title>

<**link** rel="stylesheet" href="~/lib/bootstrap/css/bootstrap.min.css" />

<**link** rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<header class="bg-dark text-white p-3">

<div class="container">

<h1>Vehicle Rental Service</h1>

</div>

</header>

<main class="container my-4">

@RenderBody()

</main>

<footer class="bg-light text-center p-3">

<div class="container">

&copy; 2025 Vehicle Rental Service

</div>

</footer>

<**script** src="~/lib/bootstrap/js/bootstrap.bundle.min.js"></**script**>

</body>

</html>

І далі в нас йде файл Index.cshtml, у якому і е вся візуальна та частинка логіки виведення ТЗ. Також реалізован Пейджинг:

@model VehicleRentalService.Models.ViewModels.VehicleListViewModel

@foreach(var v in Model.Vehicles)

{

<div>

<h3>@v.Name</h3>

<h4>@v.Description</h4>

<h4>@v.PricePerMinute.ToString("c", System.Globalization.CultureInfo.GetCultureInfo("de-DE"))/minute</h4>

<br/>

</div>

}

<div>

Page @Model.PagingInfo.CurrentPage / @Model.PagingInfo.TotalPages

</div>

<div>

@for (int i = 1; i <= Model.PagingInfo.TotalPages; i++)

{

if (i == Model.PagingInfo.CurrentPage)

{

<span>@i</span>

}

else

{

<a href="@Url.Action("Index", new { page = i })">@i</a>

}

}

</div>

<h4>Welcome to Vehicle Rental Service</h4>

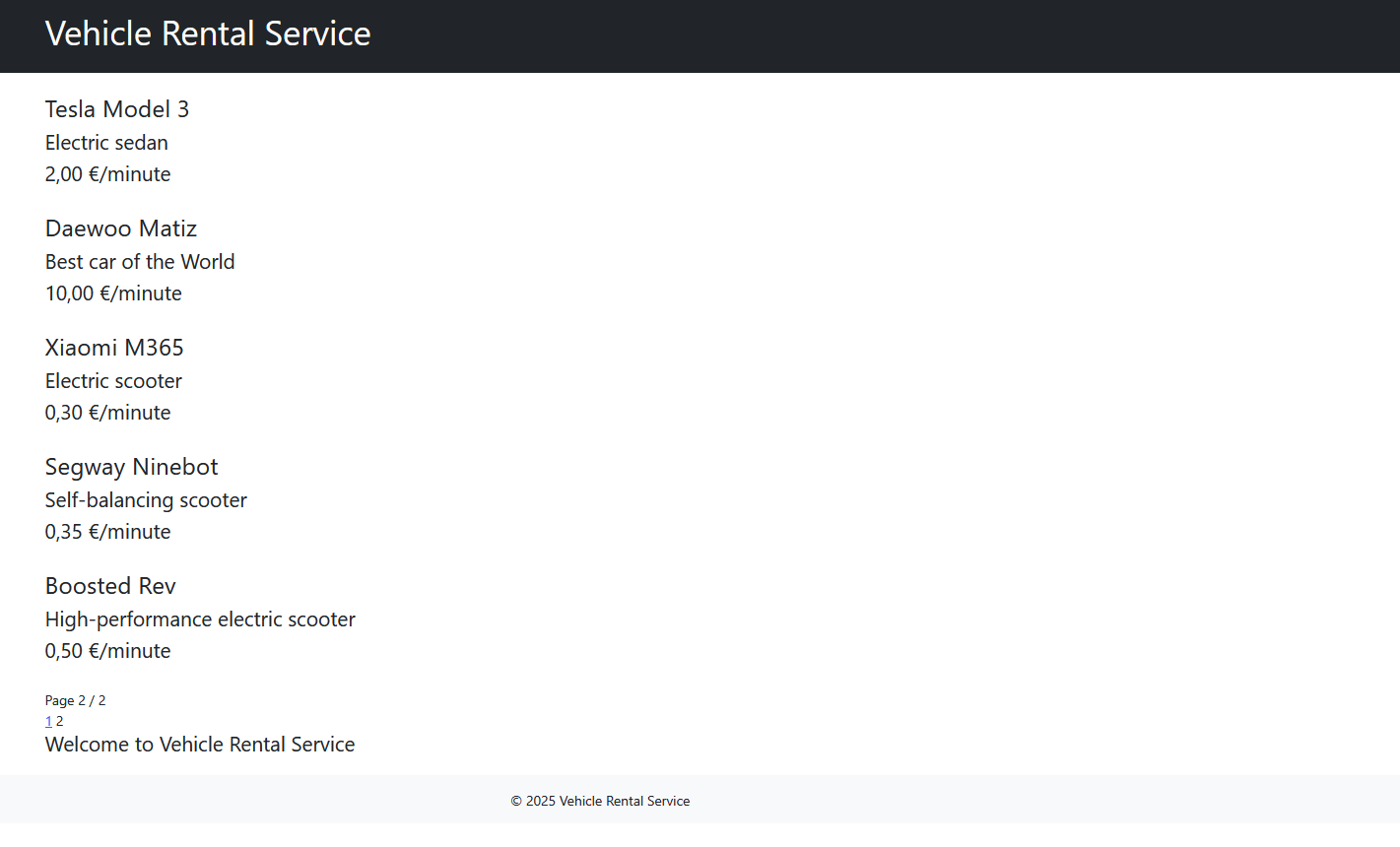


Рис.4. Головна сторінка нашого додатку

Висновки: під час виконання Л/Р 2 було ознайомлено з архітектурою MVC, набуто навичок створення головних компонентів архітектури MVC, навчився конфігурувати підключення до БД та виконувати запити на читання даних, набув навичок роботи з представленнями (Views).