**ЛАБОРАТОРНА РОБОТА № 4**

**Знайомство з принципами створення веб-додатків за допомогою ASP.NET Core MVC.**

***Мета***: ознайомитися з принципами побудови веб-додатків за допомогою патерну MVC та навчитися використовувати ASP.NET Core MVC для виконання CRUD-операцій.

**Хід роботи:**

**Завдання 2:** Створити у Visual Studio **ASP.NET Core Empty**.

Назва рішення: **AspDotNetLabs**

Назва проекту: **AspDotNetLab4**

При створенні проекту виберіть версію Framework: .NET 5.0 (Current) Якщо у Вас не доступна версія .NET 5.0, то встановіть відповідний компонент

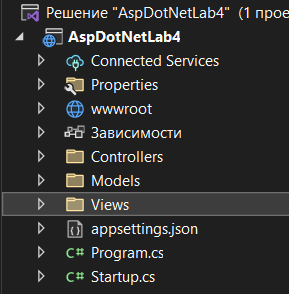


Рис. 2.0.0 Оглядач рішень з відкритим проектом

**Частина 1.** Використовуючи підхід Code First, створити дві моделі, що відповідатимуть двом таблицям бази даних зі зв’язком “один до багатьох”.

Лістинг моделі Artist.cs

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace AspDotNetLab4.Models

{

public class Artist

{

[Key]

public int ArtistId { get; set; }

[Required]

public string ArtistName { get; set; }

public ICollection<Album> Albums { set; get; }

}

}

Лістинг моделі Album.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace AspDotNetLab4.Models

{

public class Album

{

[Key]

public int AlbumId { get; set; }

[Required]

public string AlbumName { get; set; }

[ForeignKey("Artist")]

public int ArtistId { set; get; }

public Artist Artist { set; get; }

}

}

Лістинг контексту MusicContext.cs

using Microsoft.EntityFrameworkCore;

namespace AspDotNetLab4.Models

{

public class MusicContext : DbContext

{

public DbSet<Artist> Artists { set; get; }

public DbSet<Album> Albums { set; get; }

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

{

Database.EnsureCreated();

}

}

}

Лістинг класу Program.cs

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace AspDotNetLab4

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

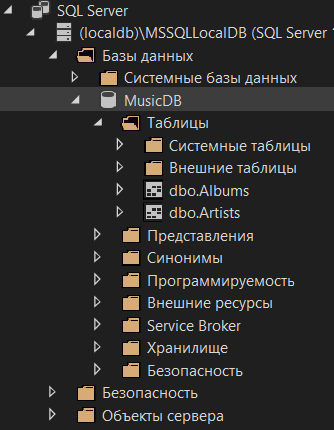
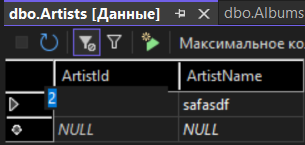


Рис. 2.1.1 Створена база та таблиці.



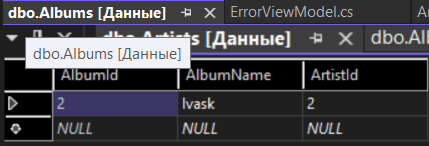


Рис. 2.1.2 – 2.1.3 Данні в таблицях після виконання програми.

**Частина 2:** За допомогою Entity Framework створити CRUD-операції для забезпечення можливості додавання, перегляду, редагування та видалення даних у цих двох таблицях.

Лістинг контролера ArtistsController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using AspDotNetLab4.Models;

namespace AspDotNetLab4.Controllers

{

public class ArtistsController : Controller

{

private readonly MusicContext \_context;

public ArtistsController(MusicContext context)

{

\_context = context;

}

// GET: Artists

public async Task<IActionResult> Index()

{

return View(await \_context.Artists.ToListAsync());

}

// GET: Artists/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var artist = await \_context.Artists

.FirstOrDefaultAsync(m => m.ArtistId == id);

if (artist == null)

{

return NotFound();

}

return View(artist);

}

// GET: Artists/Create

public IActionResult Create()

{

return View();

}

// POST: Artists/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("ArtistId,ArtistName")] Artist artist)

{

if (ModelState.IsValid)

{

\_context.Add(artist);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(artist);

}

// GET: Artists/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var artist = await \_context.Artists.FindAsync(id);

if (artist == null)

{

return NotFound();

}

return View(artist);

}

// POST: Artists/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("ArtistId,ArtistName")] Artist artist)

{

if (id != artist.ArtistId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(artist);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ArtistExists(artist.ArtistId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(artist);

}

// GET: Artists/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var artist = await \_context.Artists

.FirstOrDefaultAsync(m => m.ArtistId == id);

if (artist == null)

{

return NotFound();

}

return View(artist);

}

// POST: Artists/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var artist = await \_context.Artists.FindAsync(id);

\_context.Artists.Remove(artist);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool ArtistExists(int id)

{

return \_context.Artists.Any(e => e.ArtistId == id);

}

}

}

Лістинг контролера AlbumsController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using AspDotNetLab4.Models;

namespace AspDotNetLab4.Controllers

{

public class AlbumsController : Controller

{

private readonly MusicContext \_context;

public AlbumsController(MusicContext context)

{

\_context = context;

}

// GET: Albums

public async Task<IActionResult> Index()

{

var musicContext = \_context.Albums.Include(a => a.Artist);

return View(await musicContext.ToListAsync());

}

// GET: Albums/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var album = await \_context.Albums

.Include(a => a.Artist)

.FirstOrDefaultAsync(m => m.AlbumId == id);

if (album == null)

{

return NotFound();

}

return View(album);

}

// GET: Albums/Create

public IActionResult Create()

{

ViewData["ArtistId"] = new SelectList(\_context.Artists, "ArtistId", "ArtistName");

return View();

}

// POST: Albums/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("AlbumId,AlbumName,ArtistId")] Album album)

{

if (ModelState.IsValid)

{

\_context.Add(album);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["ArtistId"] = new SelectList(\_context.Artists, "ArtistId", "ArtistName", album.ArtistId);

return View(album);

}

// GET: Albums/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var album = await \_context.Albums.FindAsync(id);

if (album == null)

{

return NotFound();

}

ViewData["ArtistId"] = new SelectList(\_context.Artists, "ArtistId", "ArtistName", album.ArtistId);

return View(album);

}

// POST: Albums/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("AlbumId,AlbumName,ArtistId")] Album album)

{

if (id != album.AlbumId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(album);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!AlbumExists(album.AlbumId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["ArtistId"] = new SelectList(\_context.Artists, "ArtistId", "ArtistName", album.ArtistId);

return View(album);

}

// GET: Albums/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var album = await \_context.Albums

.Include(a => a.Artist)

.FirstOrDefaultAsync(m => m.AlbumId == id);

if (album == null)

{

return NotFound();

}

return View(album);

}

// POST: Albums/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var album = await \_context.Albums.FindAsync(id);

\_context.Albums.Remove(album);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool AlbumExists(int id)

{

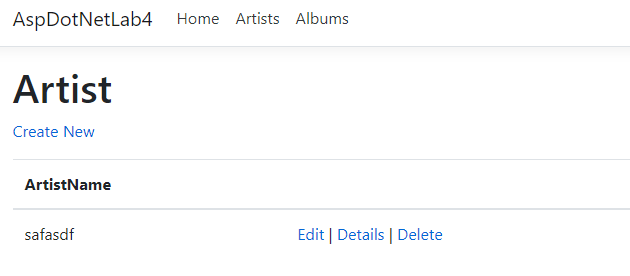
return \_context.Albums.Any(e => e.AlbumId == id);

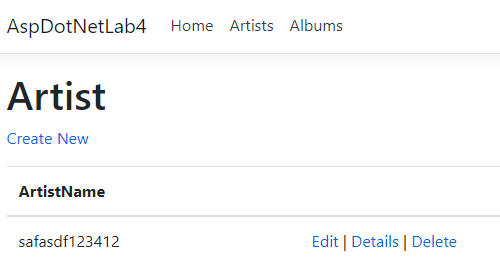
}

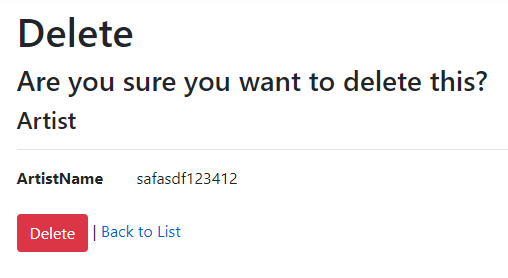
}

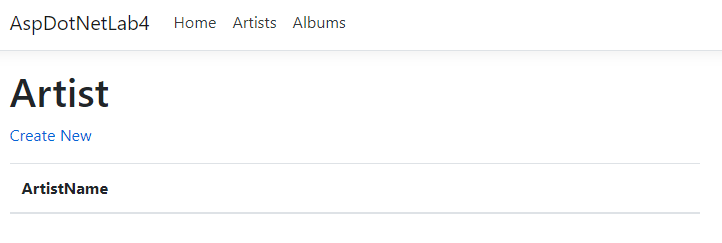
}

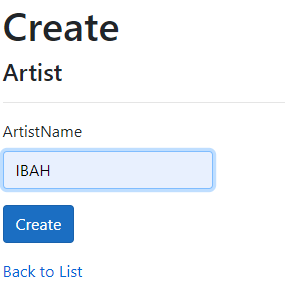
Результат виконання:











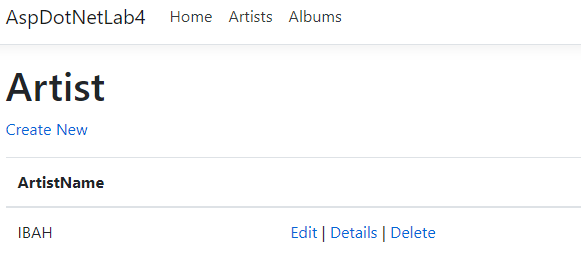
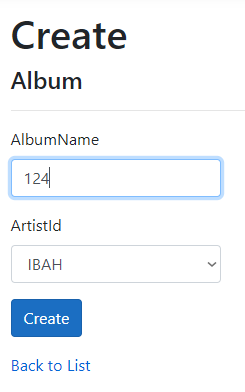
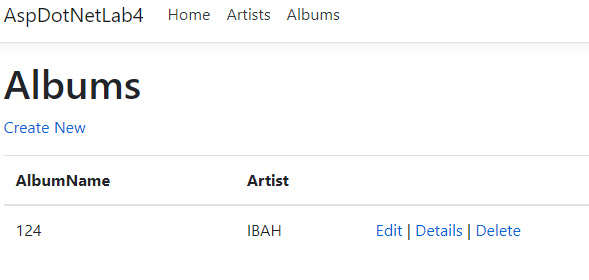
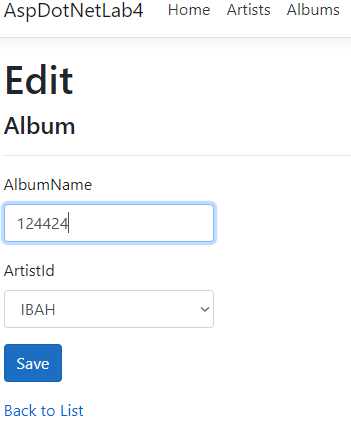
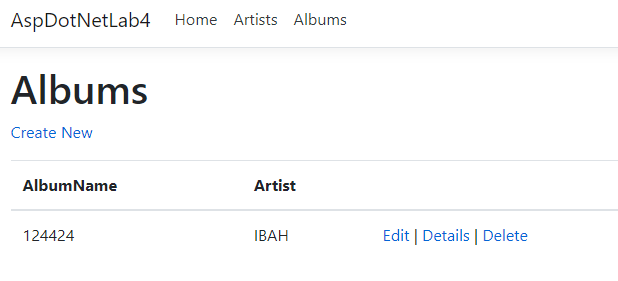
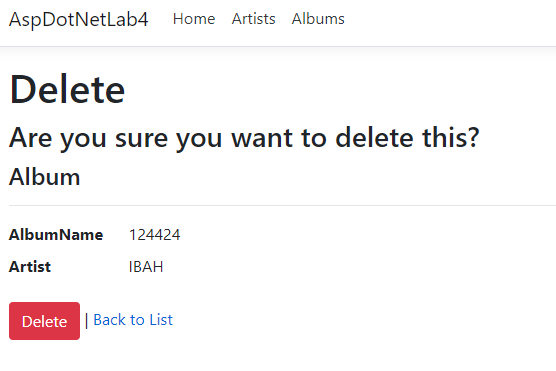


Рис. 2.2.1- 2.2.6. Перегляд, редагування, видалення, створення користувачів.

При видаленні користувача видаляються і всі його альбоми.



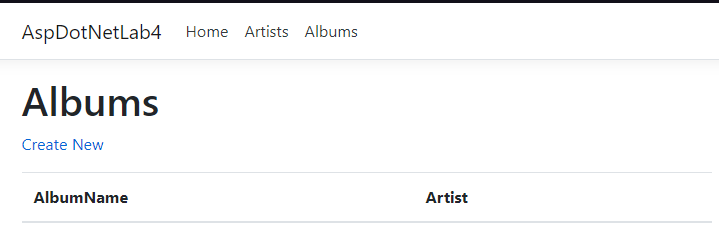


Рис. 2.2.5- 2.2.8. Перегляд, редагування, видалення, створення завданнь.

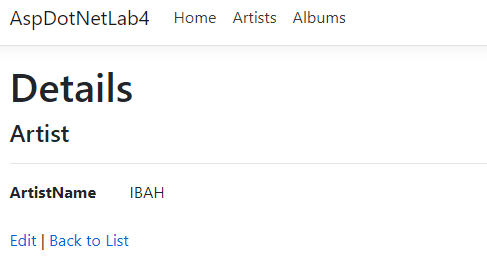
****

Рис. 2.2.9. Перегляд детальної інформації про користувача.

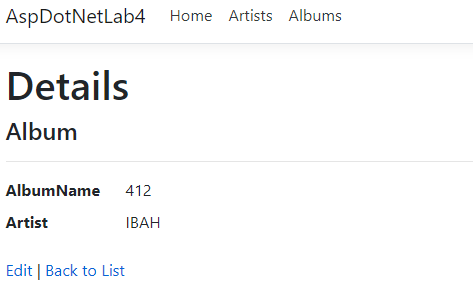
****

Рис. 2.2.10. Перегляд детальної інформації про завдання.

**Завдання 3.**

Запушити виконану роботу у репозиторій на GitHub за назвою AspDotNetLab1 і відкрити доступ для викладачів:

● [morozov@ztu.edu.ua](mailto:morozov@ztu.edu.ua)

● [4ov.ztu@gmail.com](mailto:4ov.ztu@gmail.com)

Посилання на репозиторій:

<https://github.com/IvanPaliy/AspDotNetLab4.git>

***Висновки:*** в ході виконання лабораторної роботи я ознайомився з принципами побудови веб-додатків за допомогою патерну MVC та навчився використовувати ASP.NET Core MVC для виконання CRUD-операцій.