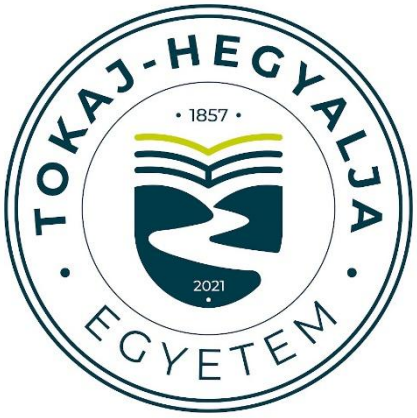


# Programozási technológiák és keretrendszerek jegyzőkönyv

Készítette: Piszró Dániel

Neptun kód: ELJUVY

2025.12.30.

# Bevezetés:

Az elkészített feladat egy egyszerű jármű nyilvántartó oldal, melyre felvihetjük az egyes járművek adatait (rendszám, márka, típus, tulaj neve). Ezeket az adatokat a főoldal kilistázza, ahol tudjuk a tételeket szerkeszteni, valamint törölni. Az alkalmazás localhost-on fut, így nem igényel internetkapcsolatot.

A következő technológiákra épül:

**C#** - az alkalmazás programozási nyelve

**ASP.NET core MVC** - a webes keretrendszer

**Entity framework core** - adatbázis-kezelés

**SQLite** - fájl alapú relációs adatbázis

# Kódrészek:

## Program.cs

Felel az alkalmazás indításáért, a szükséges szolgáltatások és az adatbázis regisztrálásáért, valamint a webes kérés feldolgozás és útvonal kezelés beállításáért.

```
using Microsoft.EntityFrameworkCore;
using JarmuMvcApp.Data;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

// SQLite connection
builder.Services.AddDbContext<AppDbContext>(options =>
    options.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection") ?? "Data Source=jarmu.db"));

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Vehicles}/{action=Index}/{id?}");

app.Run();
```

## VehiclesController.cs

Az alkalmazás vezérlő kódrésze. Feladata a járművekhez kapcsolódó kérések kezelése, valamint kapcsolatot tart a felhasználói felület és az adatbázis között.

Index: Lekéri az összes járművet az adatbázisból, majd rendszám szerint rendezve kiíratja azokat.

```
public async Task<IActionResult> Index()
{
    var list = await _context.Vehicles.OrderBy(v => v.LicensePlate).ToListAsync();
    return View(list);
}
```

Create: Megjeleníti az új jármű felvételére szolgáló űrlapot, felveszi onnan az adatokat és ellenőrzi őket. Végül menti az adatbázisba.

```
// GET: Vehicles/Create
0 references
public IActionResult Create()
{
    return View();
}

// POST: Vehicles/Create
[HttpPost]
0 references
public async Task<IActionResult> Create([Bind("LicensePlate,Owner,Name,Type")] Vehicle vehicle)
{
    if (ModelState.IsValid)
    {
        _context.Add(vehicle);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(vehicle);
}
```

Edit: Betölti a kiválasztott jármű adatait, majd megjeleníti a szerkesztő oldalt. Feldolgozza a bevitt adatokat és frissíti az adatbázisban.

```
// GET: Vehicles/Edit
0 references
public async Task<IActionResult> Edit(int? id)
{
    if (id == null) return NotFound();

    var vehicle = await _context.Vehicles.FindAsync(id);
    if (vehicle == null) return NotFound();
    return View(vehicle);
}

// POST: Vehicles/Edit
[HttpPost]
0 references
public async Task<IActionResult> Edit(int id, [Bind("Id,LicensePlate,Owner,Name,Type")] Vehicle vehicle)
{
    if (id != vehicle.Id) return NotFound();

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(vehicle);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!VehicleExists(vehicle.Id)) return NotFound();
            else throw;
        }
        return RedirectToAction(nameof(Index));
    }
    return View(vehicle);
}
```

Delete: Megjeleníti a törlést megerősítő oldalt és választás szerint véglegesen törli azt az adatbázisból.

```
// GET: Vehicles/Delete/5
0 references
public async Task<IActionResult> Delete(int? id)
{
    if (id == null) return NotFound();

    var vehicle = await _context.Vehicles.FirstOrDefaultAsync(m => m.Id == id);
    if (vehicle == null) return NotFound();

    return View(vehicle);
}

// POST: Vehicles/Delete/5
[HttpPost, ActionName("Delete")]
0 references
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var vehicle = await _context.Vehicles.FindAsync(id);
    if (vehicle != null)
    {
        _context.Vehicles.Remove(vehicle);
        await _context.SaveChangesAsync();
    }
    return RedirectToAction(nameof(Index));
}
```

## AppDbContext.cs

Az alkalmazás adatbázis kezelője, az Entity framework core segítségével biztosítja a kapcsolatot C# objektumok és az adatbázis között

```
using Microsoft.EntityFrameworkCore;
using JarmuMvcApp.Models;

namespace JarmuMvcApp.Data
{
    7 references
    public class AppDbContext : DbContext
    {
        0 references
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

        7 references
        public DbSet<Vehicle> Vehicles { get; set; } = null!;

        0 references
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<Vehicle>(entity =>
            {
                entity.HasKey(e => e.Id);
                entity.Property(e => e.LicensePlate).IsRequired().HasMaxLength(20);
                entity.Property(e => e.Owner).HasMaxLength(100);
                entity.Property(e => e.Name).HasMaxLength(100);
                entity.Property(e => e.Type).HasMaxLength(50);
            });
        }
    }
}
```

## Vehicles.cs

Az alkalmazás adatmodellje, egy jármű adatait adja meg. Az osztály megfeleltethető az adatbázis egy sorának.

```
using System.ComponentModel.DataAnnotations;

namespace JarmuMvcApp.Models
{
    14 references
    public class Vehicle
    {
        12 references
        public int Id { get; set; }

        [Required]
        [Display(Name = "Rendszám")]
        6 references
        public string LicensePlate { get; set; } = string.Empty;

        [Display(Name = "Tulaj")]
        5 references
        public string? Owner { get; set; }

        [Display(Name = "Név")]
        5 references
        public string? Name { get; set; }

        [Display(Name = "Típus")]
        5 references
        public string? Type { get; set; }
    }
```