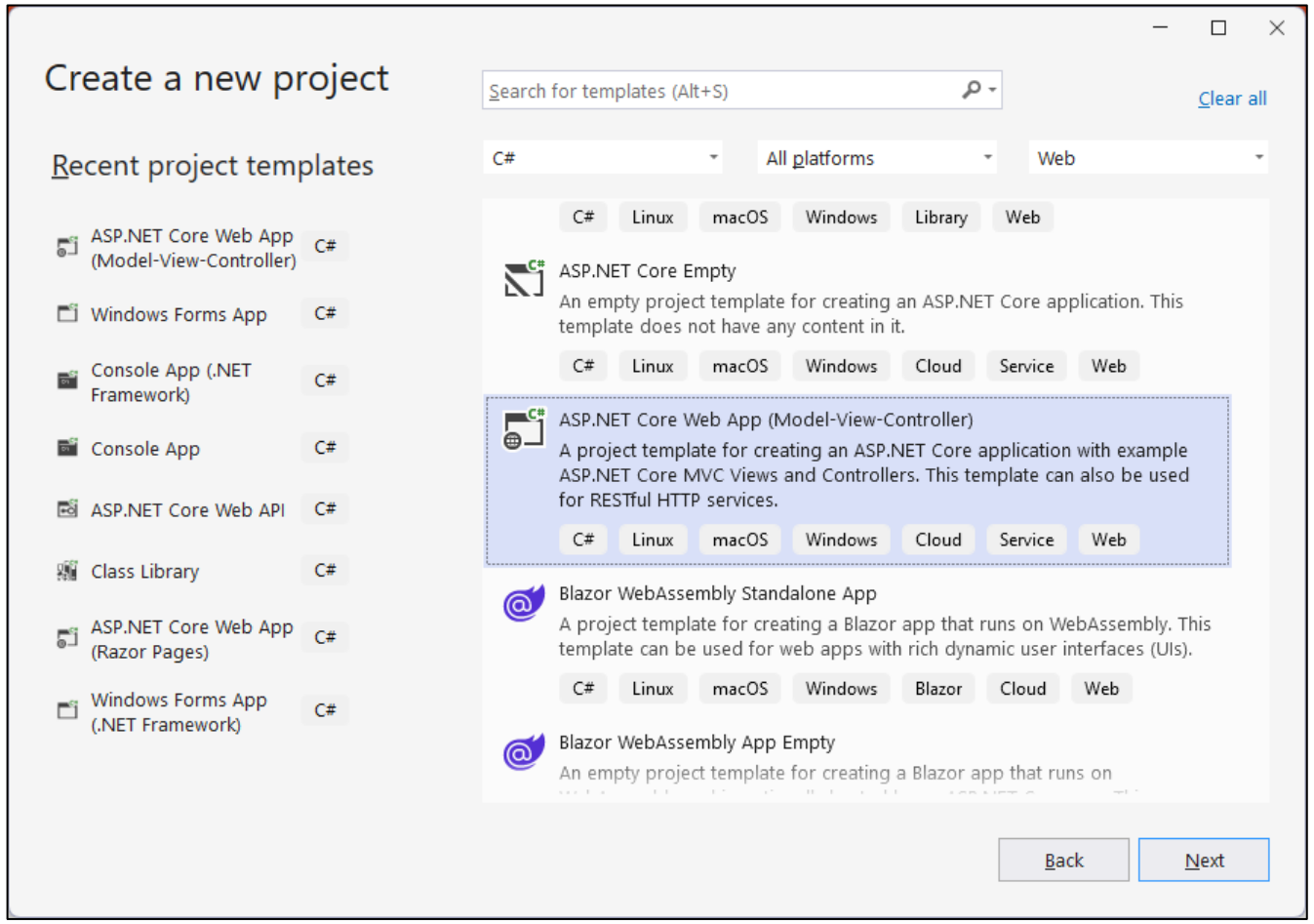
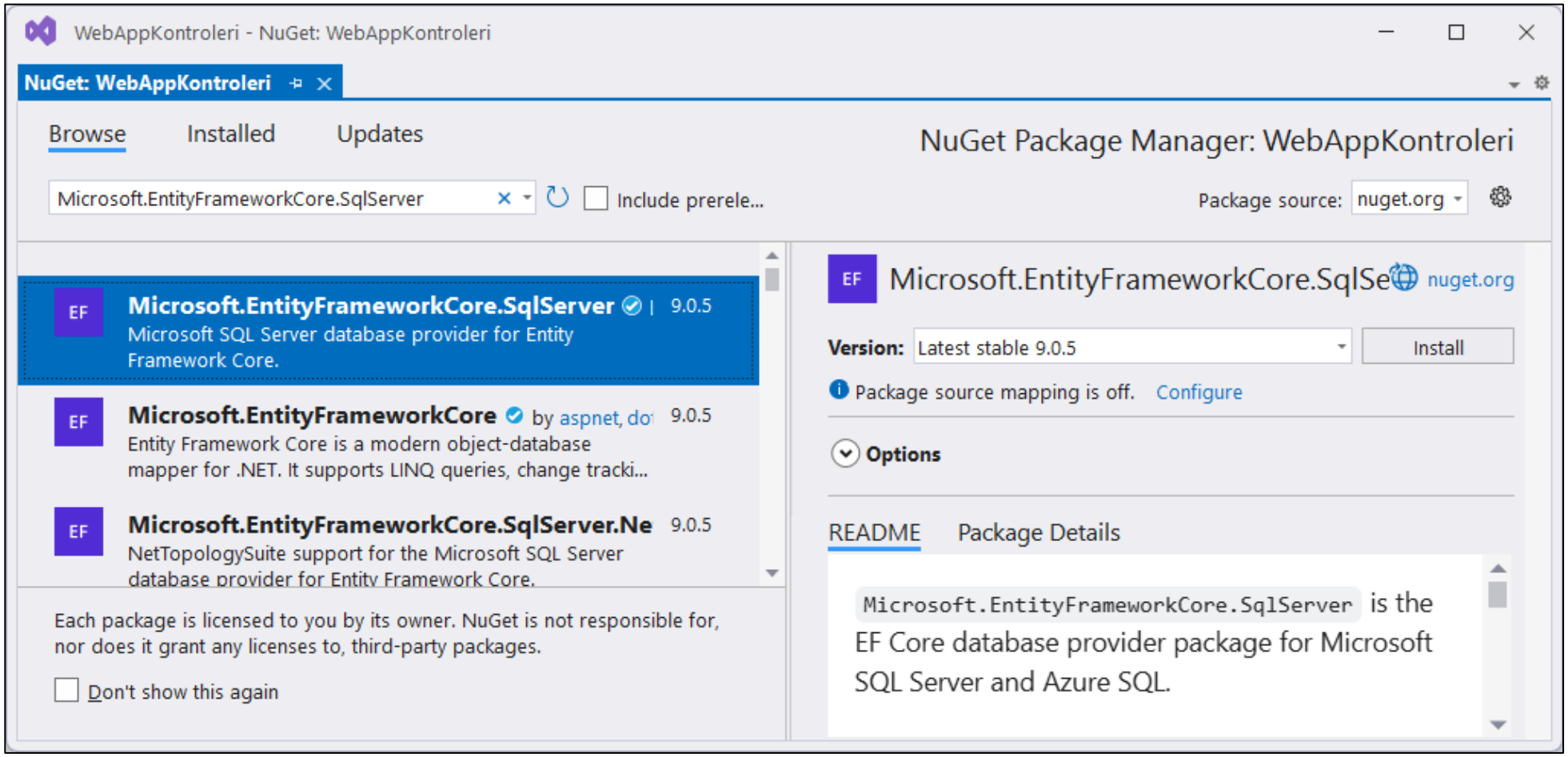
Kreiranje MVC kontrolera

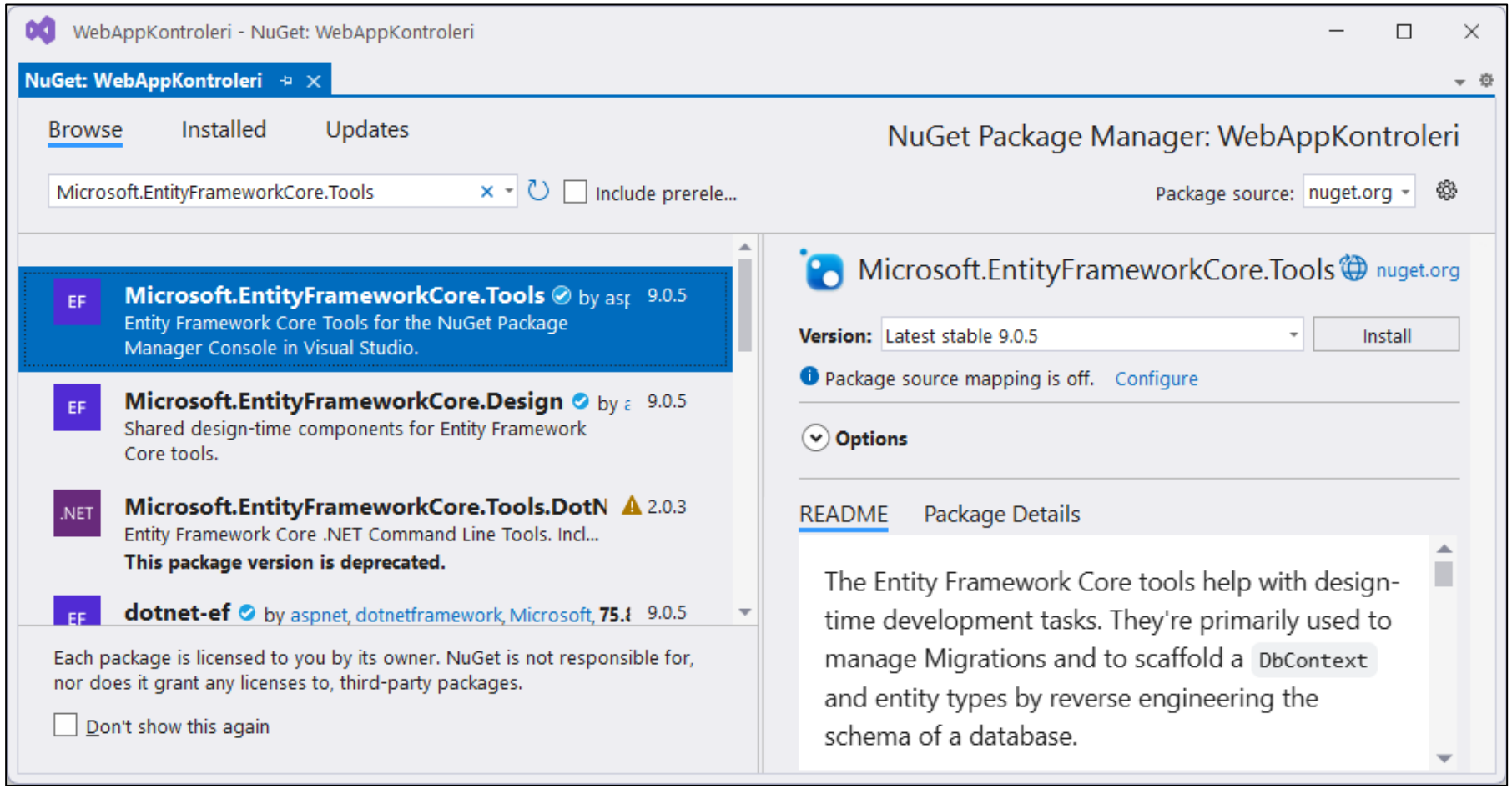
Kreiranje ASP.NET Core Web aplikacije



Microsoft.EntityFrameworkCore.SqlServer



Microsoft.EntityFrameworkCore.Tools



Pojam kontrolera

- Kontroler je C# klasa čije su javne metode odgovorne za obradu HTTP zahteva kod ASP.NET Core MVC web aplikacija
- Ime kontrolerske klase se završava sa *Controller*
- Kontroler je klasa koja je obično izvedena iz bazne klase *Controller*
- Metoda kontrolera koja obrađuje HTTP zahtev naziva se akciona metoda ili akcija
- Svaki kontroler sadrži jednu ili više akcionih metoda
- Akciona metoda kontrolera mora biti javna, ne može biti statička, i ne može se overload-ovati
- Akciona metoda može vratiti objekat bilo kog tipa
- Uobičajeno je da akciona metoda vraća objekat koji implementira *IActionResult* interfejs
- Akcija kontrolera prosleđuje podatke odgovarajućem pogledu

Rutiranje

- Rute opisuju kako se URL adrese zahteva mapiraju na akcije kontrolera
- Rute se definišu unutar middleware komponente za rutiranje u fajlu Program.cs
- Metoda **MapControllerRoute** klase WebApplication koristi se za kreiranje podrazumevane rute
- Ako želimo da pozovemo Details akcionu metodu kontrolera OsobaController i prosledimo joj parametar id = 3, potrebno je uneti sledeću adresu u browser: http://<app>/osoba/details/3

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

HomeController

```
public class HomeController : Controller
{

    public IActionResult Index()
    {
        return View();
    }

}
```

Interfejs IActionResult

- Ovim interfejsom definiše se ugovor koji predstavlja rezultat izvršavanja akcione metode
- Akcione metode obično vraćaju objekat koji implementira **IActionResult**
- Metoda **View**() bazne klase **Controller** vraća instancu klase **ViewResult**
- Metoda **Json**() vraća objekat klase *JsonResult* i koristi se kada akcija treba da generiše JSON

Model klasa Osoba i DbContext klasa OsobaContext

```
[Table("Osoba")]
public class Osoba
{
    public int OsobaId { get; set; }

    [Required]
    [StringLength(30)]
    public required string Ime { get; set; }

    [Required]
    [StringLength(30)]
    public required string Prezime { get; set; }

    [Required]
    [StringLength(100)]
    public required string Adresa { get; set; }
}
```

```
public class OsobaContext : DbContext
{
    public DbSet<Osoba> Osobe { get; set; }
}
```

Definisanje konekcionog stringa u appsettings.json fajlu

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=GORAN-HP;
     Initial Catalog=OsobaDb15052025;Integrated Security=True;Encrypt=False"
  }
}
```

Registracija DbContext servisa

```
public static void Main(string[] args)
{
    var builder = WebApplication.CreateBuilder(args);
    var connectionString =
    builder.Configuration.GetConnectionString("DefaultConnection");

    // Add services to the container.
    builder.Services.AddDbContext<OsobaContext>(options =>
                    options.UseSqlServer(connectionString));
    builder.Services.AddControllersWithViews();

    var app = builder.Build();
    ....

}
```

Dodavanje konstruktora u DbContext klasu

```
using Microsoft.EntityFrameworkCore;
```

```
public class OsobaContext : DbContext
{

    public OsobaContext(DbContextOptions<OsobaContext> opcije) : base(opcije)
    {
    }

    public DbSet<Osoba> Osobe { get; set; }

}
```

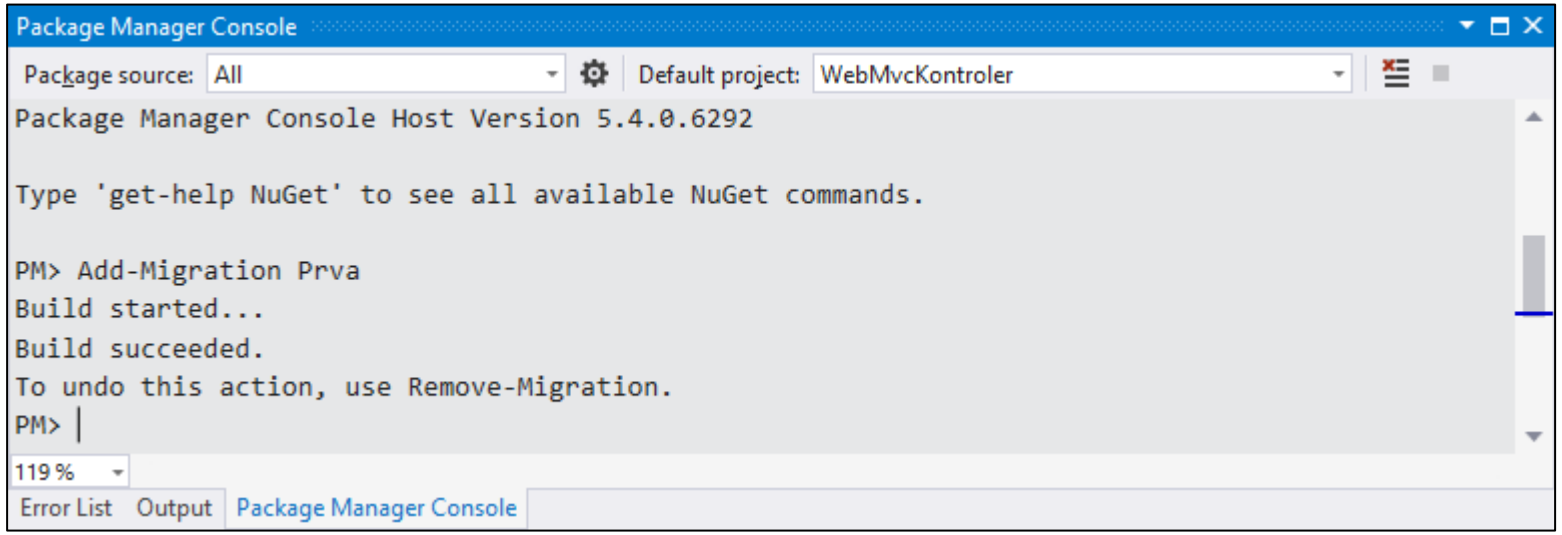
Novi stil definisanja konstruktora C# 12, primary constructor

```
public class OsobaContext(DbContextOptions<OsobaContext> opcije) : DbContext(opcije)
{
    public DbSet<Osoba> Osobe { get; set; }
}
```

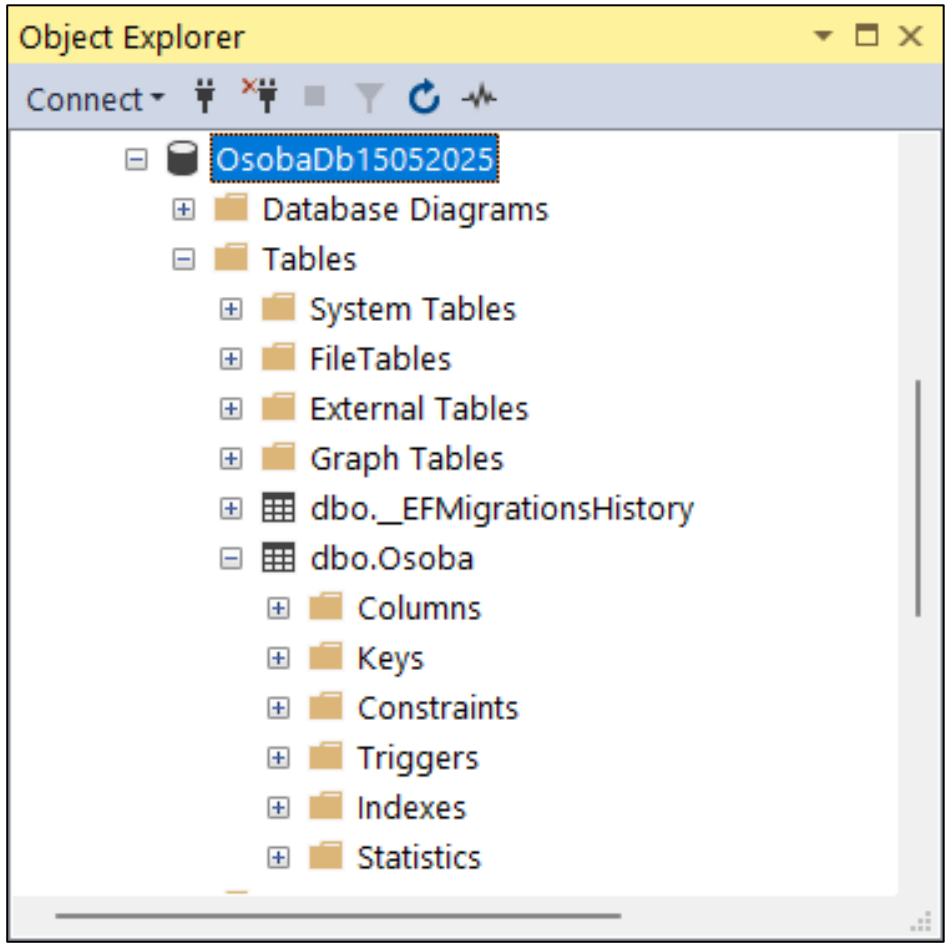
Kreiranje baze podataka primenom migracija

```
PM> Add-Migration Prva
```

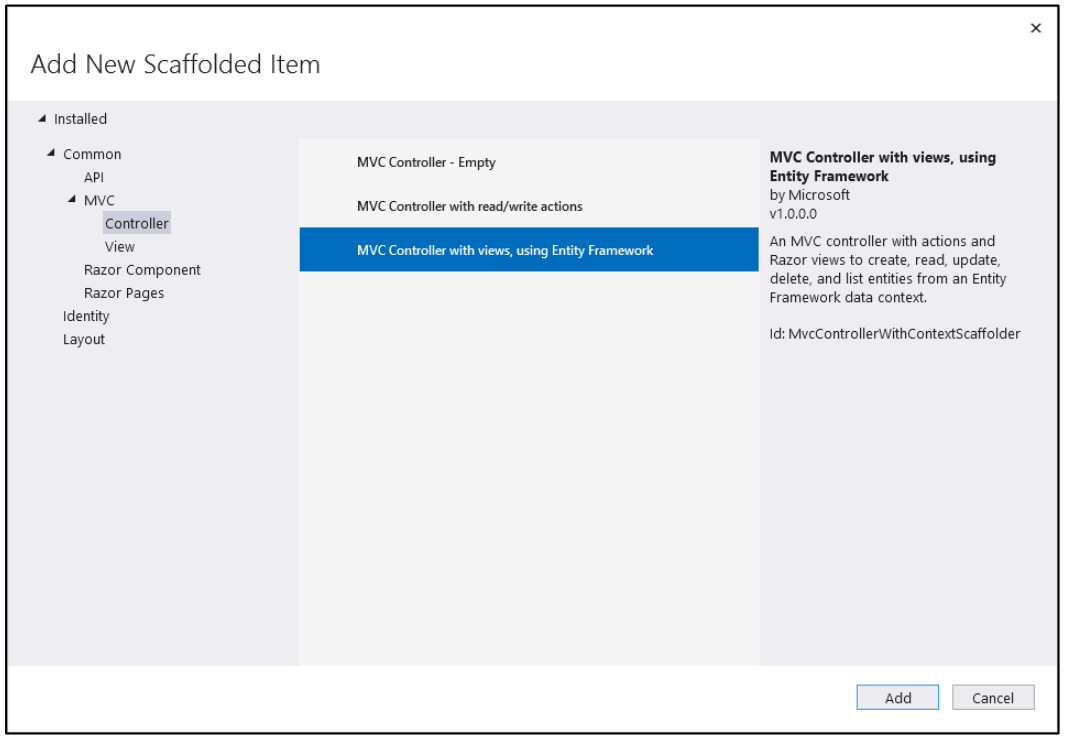
```
PM> Update-Database
```

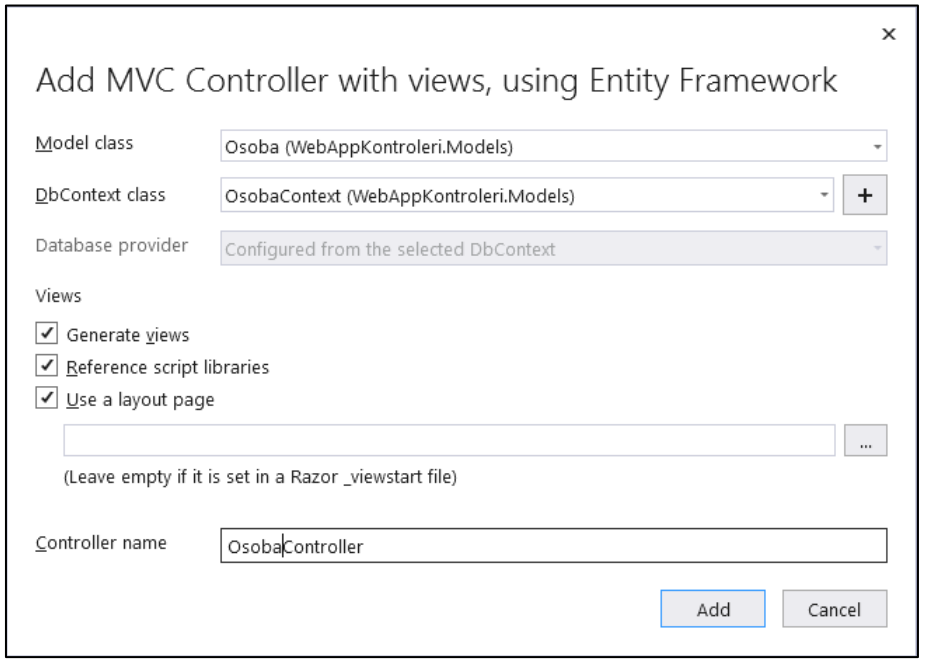


Kreirana baza podataka



Kreiranje kontrolera





Klasa OsobaController

```
public class OsobaController : Controller
{
    private readonly OsobaContext _context;

    public OsobaController(OsobaContext context)
    {
        _context = context;
    }
}
```

- Klasa OsobaController nasleđuje baznu klasu Controller
- Koristi servis OsobaContext za pristup bazi podataka
- OsobaContext se prosleđuje kroz konstruktor (Dependency Injection)
- Polje _context služi za rad sa podacima

Akcija Index asinhrona verzija

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebMvcKontroler.Models;
using Microsoft.EntityFrameworkCore;
```

```
public async Task<IActionResult> Index()
{
    return View(await _context.Osobe.ToListAsync());
}
```

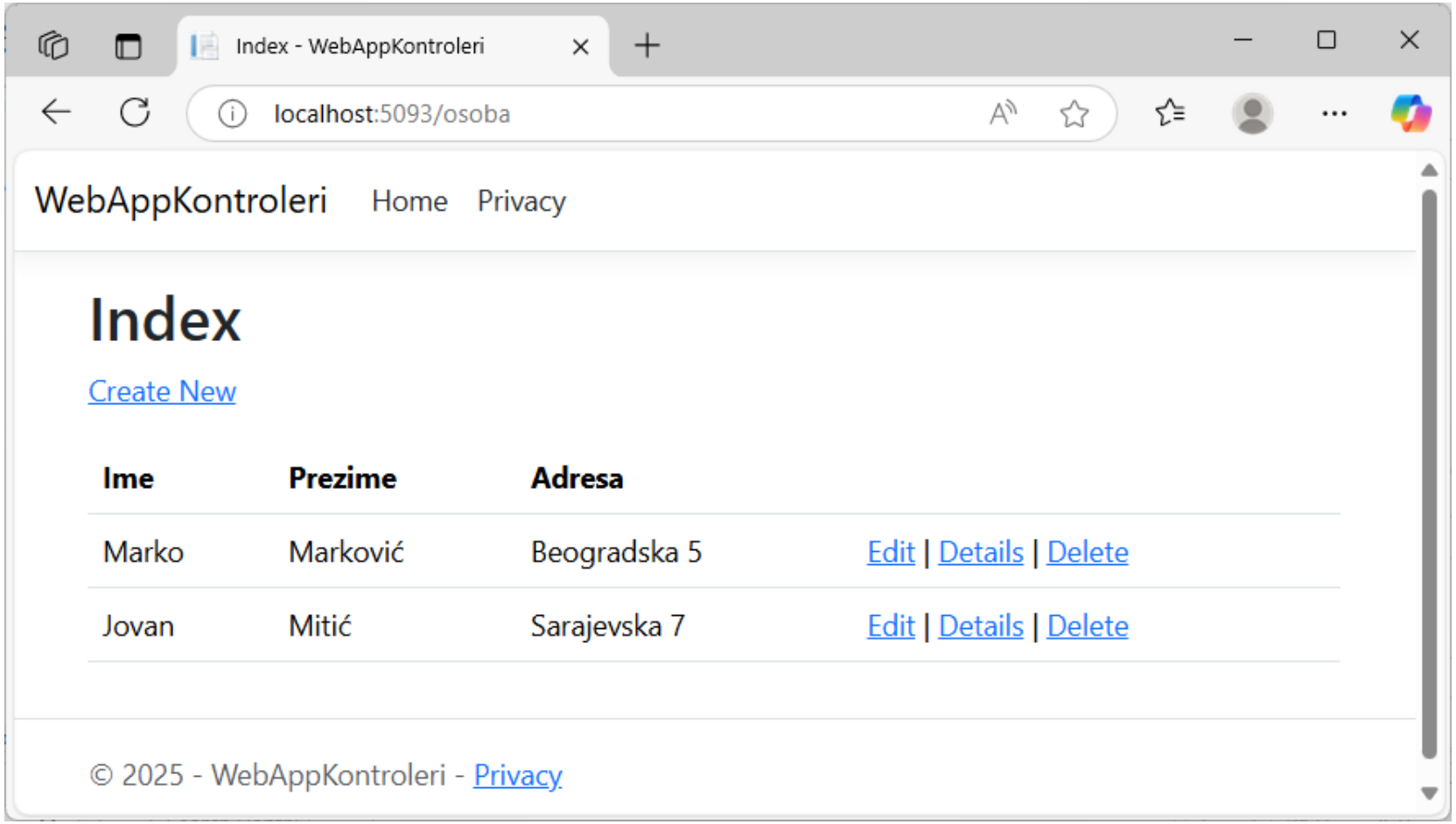
- Akciona metoda je deklarisana kao async
- Vraća Task<IActionResult>
- Koristi await za asinhrono izvršavanje upita
- Metoda ToListAsync() asinhrono učitava podatke iz baze
- Dobijeni podaci se prosleđuju pogledu pomoću View()

Akcija Index sinhrona verzija

```
// GET: Osoba
public IActionResult Index()
{
    return View(_context.Osobe.ToList());
}
```

- Akciona metoda vraća IActionResult
- Podaci se učitavaju sinhrono pomoću ToList()
- Rezultat se prosleđuje pogledu pomoću View()

Poziv Index akcije kontrolera Osoba



Akciona metoda Details asinrona verzija

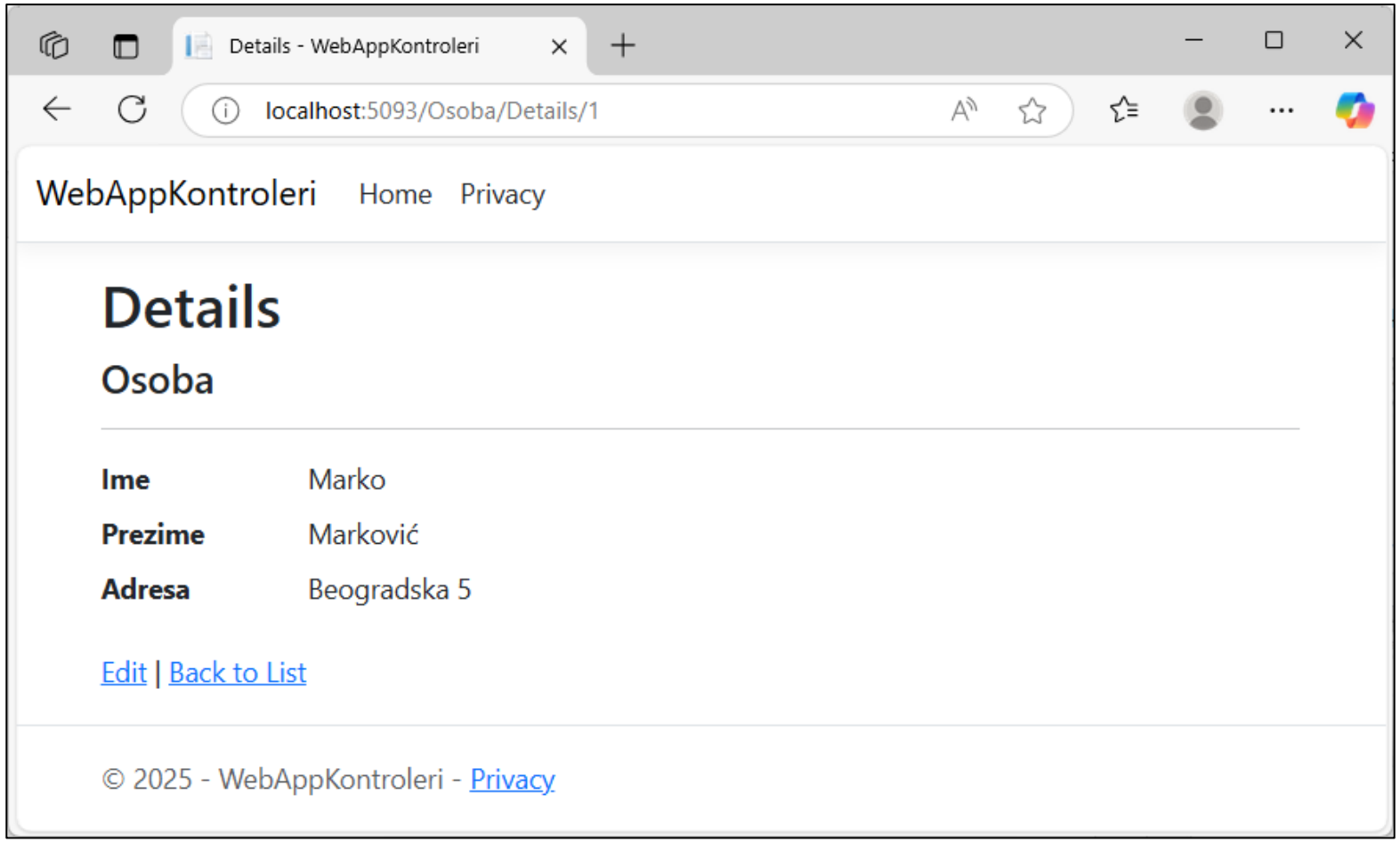
```
// GET: Osoba/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }
    var osoba = await _context.Osobe
        .FirstOrDefaultAsync(m => m.OsobaId == id);
    if (osoba == null)
    {
        return NotFound();
    }
    return View(osoba);
}
```

- Metoda NotFound() kreira objekat klase NotFoundResult
- NotFound() vraća HTTP odgovor klijentu (browseru)
- Status je 404 – Not Found

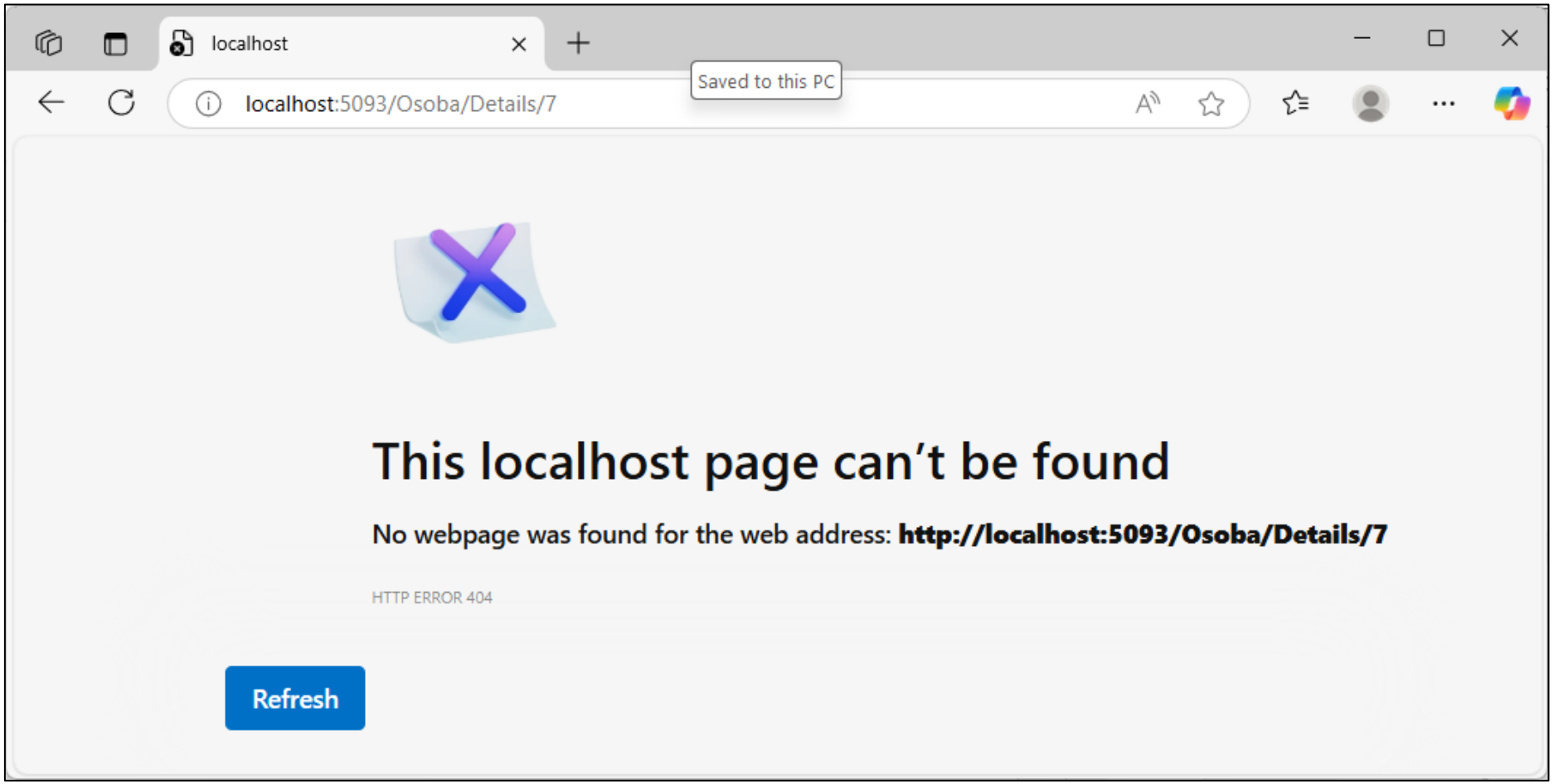
Metoda Details sinhrona verzija

```
public IActionResult Details(int? id)
{
    if (id == null || _context.Osobe == null)
    {
        return NotFound();
    }

    var osoba =  _context.Osobe
        .FirstOrDefault(m => m.OsobaId == id);
    if (osoba == null)
    {
        return NotFound();
    }

    return View(osoba);
}
```

Poziv Details akcije kontrolera Osoba



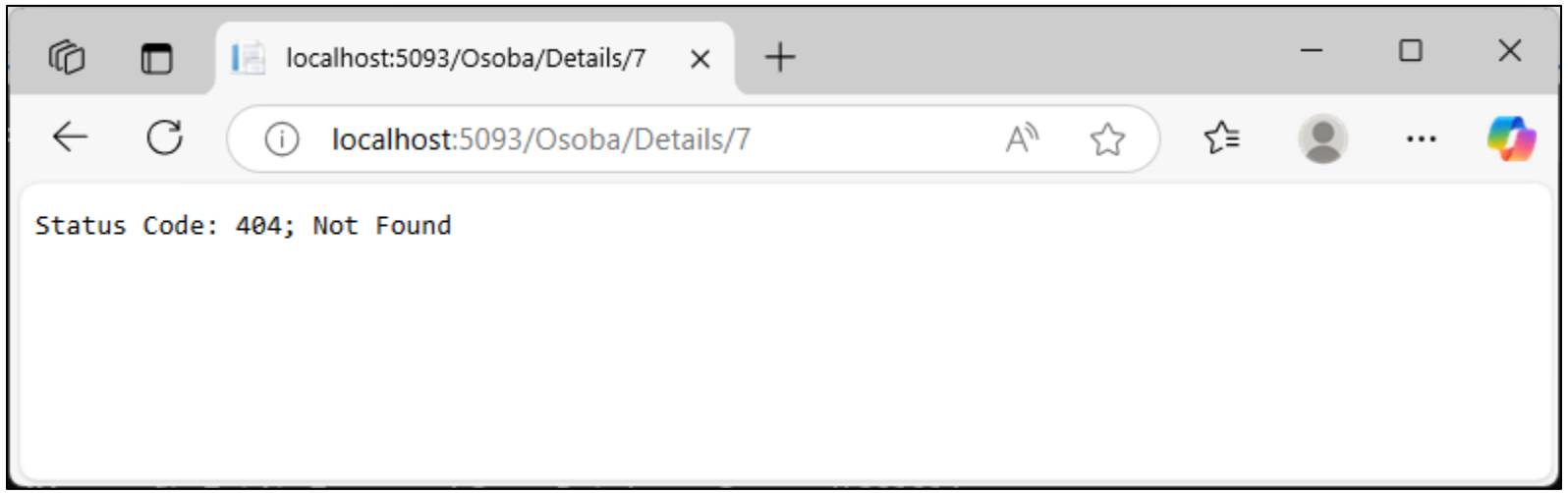
Prosleđivanje nepostojećeg id-a



Dodavanje middleware komponente za obradu odgovara sa kodnim statusom od 400 do 599

```
app.UseStatusCodePages();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

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

app.Run();
```

Loš poziv Details akcione metode



Redirekcija na korisnički kreiranu stranu greške

```
public IActionResult Details(int? id)
{
    if (id == null || _context.Osobe == null)
    {
        return RedirectToAction("Error");
    }

    var osoba =  _context.Osobe
        .FirstOrDefault(m => m.OsobaId == id);
    if (osoba == null)
    {
        return  RedirectToAction("Error");
    }

    return View(osoba);
}
```

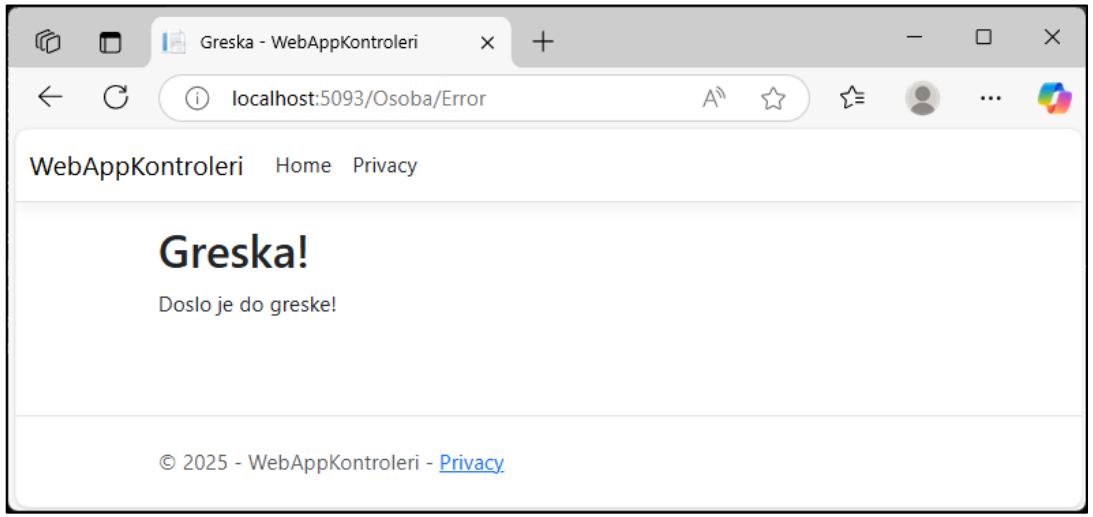
RedirectToAction("Error") → preusmerava na akciju Error (stranicu greške)

Akcija Error()

```
public IActionResult Error()
{
    return View();
}
```

Pogled Error.cshtml

```
@{
    ViewData["Title"] = "Greska";
}

<h1>Greska!</h1>
<p>Doslo je do greske!</p>
```



GET i POST akciona metoda Create

```
// GET: Osoba/Create
public IActionResult Create()
{
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("OsobaId,Ime,Prezime,Adresa")] Osoba osoba)
{
    if (ModelState.IsValid)
    {
        _context.Add(osoba);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(osoba);
}
```

Akciona metoda Create sinhrona verzija

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create([Bind("OsobaId,Ime,Prezime,Adresa")] Osoba osoba)
{
    if (ModelState.IsValid)
    {
        _context.Add(osoba);
        _context.SaveChanges();
        return RedirectToAction(nameof(Index));
    }
    return View(osoba);
}
```

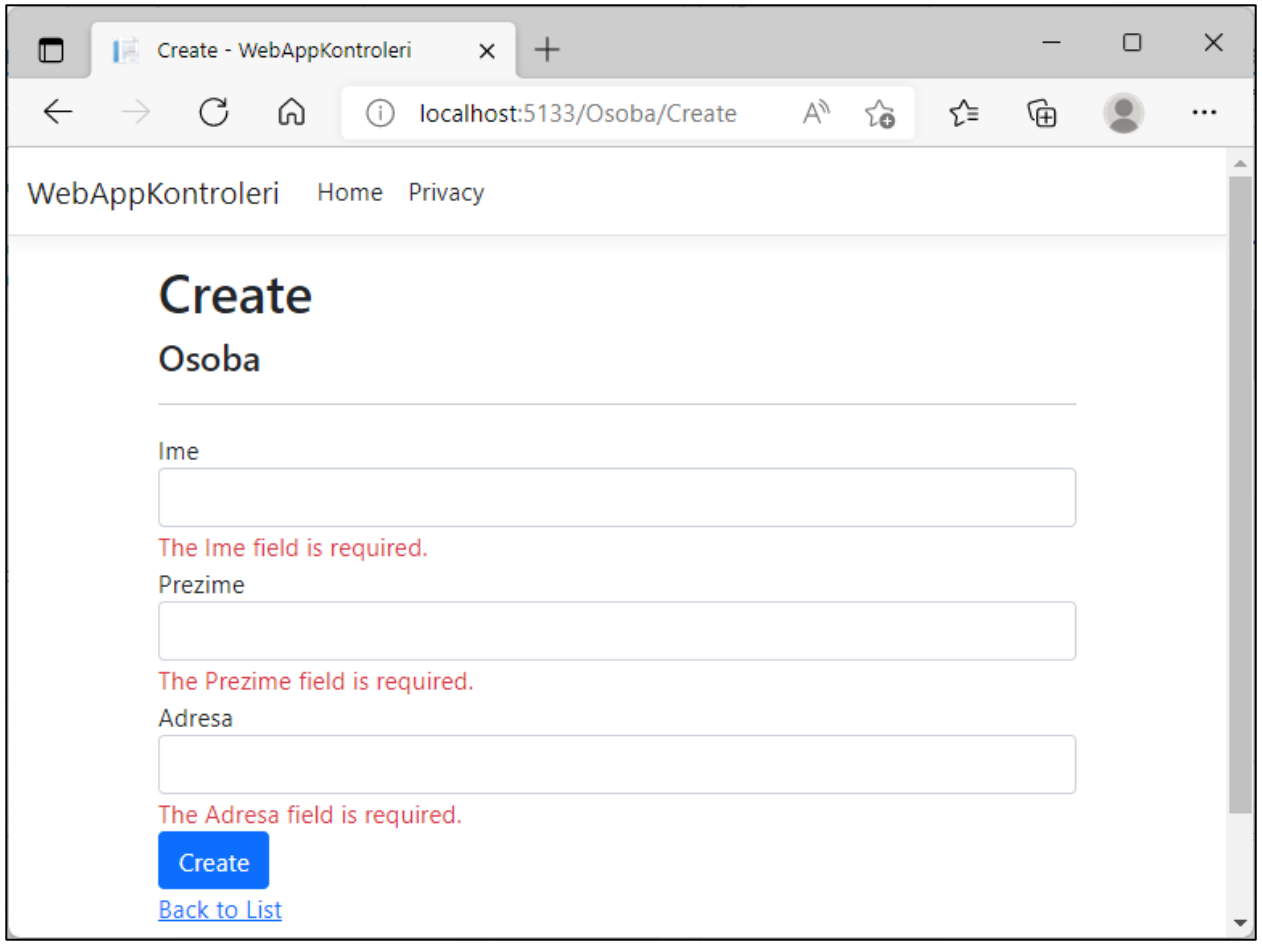
Model binding

- Model binding je proces kreiranja .NET objekata korišćenjem podataka iz HTTP zahteva
- Kreirani objekti se prosleđuju akcionim metodama kontrolera
- **Model binder** je komponenta MVC aplikacije koja obezbeđuje podatke za akcione metode iz različitih delova HTTP zahteva
- Model binder koristi tri izvora podataka da bi mapirao HTTP zahteve u podatke neophodne akcionim metodama
  - vrednosti polja forme
  - vrednosti parametara iz rute
  - vrednosti parametara iz query stringova

Svojstva i atributske klase koje koristi metoda Create

- **ValidateAntiForgeryToken** predstavlja atributsku klasu koja sprečava falsifikovanje zahteva
- **ModelState** je property kontrolera i sadrži kolekciju (name,value) parova koju su prosleđeni u POST zahtevu do servera, takođe sadrži i kolekciju grešaka za svaku vrednost poslatu do servera
- **ModelState.IsValid** ukazuje na to da li model sadrži greške (false ako postoje greške)
- Pomoću atributske klase **Bind** se specificiraraju koja svojstva model klase se koriste u kreiranju .NET objekata od strane model bindera
- Metoda **RedirectToAction** poziva akciju kontrolera koja se specificira kao string parameter metode
- Ključna reč **nameof** se koristi da se dobije ime promenljive, tipa ili metode

Poziv akcione metode Create kontrolera Osoba



Edit GET metoda – asinhrona verzija

```
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var osoba = await _context.Osobe.FindAsync(id);
    if (osoba == null)
    {
        return NotFound();
    }
    return View(osoba);
}
```

Edit GET metoda – sinhrona verzija

```
// GET: Osoba/Edit/5
public  IActionResult Edit(int? id)
{
    if (id == null || _context.Osobe == null)
    {
        return RedirectToAction("Error");
    }

    var osoba =  _context.Osobe.Find(id);
    if (osoba == null)
    {
        return RedirectToAction("Error");
    }
    return View(osoba);
}
```

Edit POST metoda - asinhrona verzija

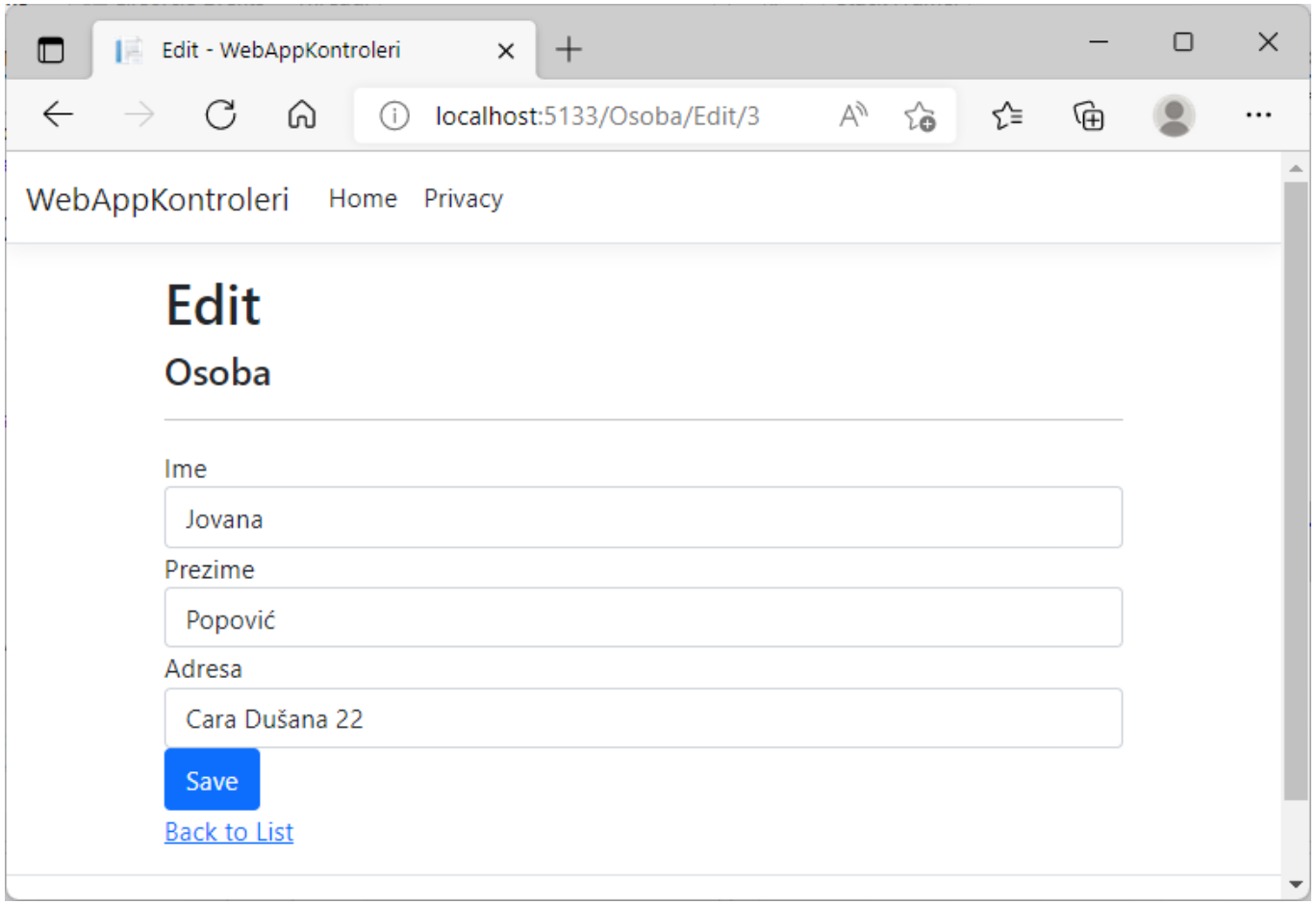
```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("OsobaId,Ime,Prezime,Adresa")]
Osoba osoba)
{
    if (id != osoba.OsobaId)
    {
        return NotFound();
    }

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

```
private bool OsobaExists(int id)
{
    return _context.Osobe.Any(e => e.OsobaId == id);
}
```

Edit POST metoda - sinhrona verzija

```
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Edit(int id, [Bind("OsobaId,Ime,Prezime,Adresa")] Osoba osoba)
{
    if (id != osoba.OsobaId)
    {
        return RedirectToAction("Error");
    }

    try
    {
        if (ModelState.IsValid)
        {
            _context.Update(osoba);
            _context.SaveChanges();
            return RedirectToAction(nameof(Index));
        }
        return View(osoba);
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!OsobaExists(osoba.OsobaId))
        {
            return RedirectToAction("Error");
        }
        return RedirectToAction("Error");
    }
    catch (Exception)
    {
        // Log the exception or do something else with it.
        return RedirectToAction("Error");
    }
}
```

Poziv Edit GET metode kontrolera Osoba



Delete GET metoda – asinhrona verzija

```
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var osoba = await _context.Osobe
        .FirstOrDefaultAsync(m => m.OsobaId == id);
    if (osoba == null)
    {
        return NotFound();
    }

    return View(osoba);
}
```

Delete GET metoda – sinhrona verzija

```
public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.Osobe == null)
    {
        return RedirectToAction("Error");
    }

    var osoba = await _context.Osobe
        .FirstOrDefaultAsync(m => m.OsobaId == id);
    if (osoba == null)
    {
        return RedirectToAction("Error");
    }

    return View(osoba);
}
```

Delete POST metoda – asinhrona verzija

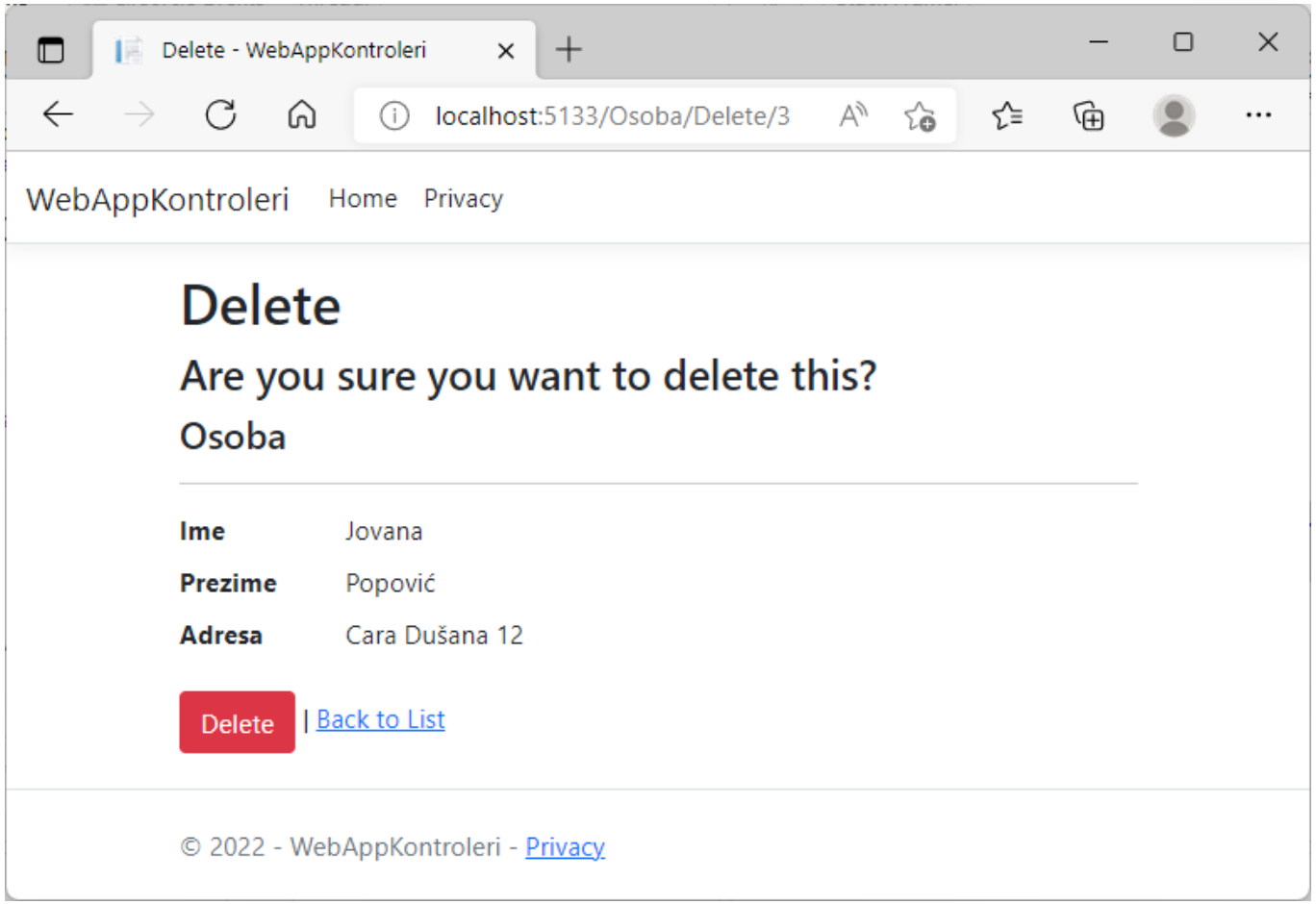
ActionName klasa ukazuje da se radi o akciji Delete

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    if (_context.Osobe == null)
    {
        return RedirectToAction("Error");
    }
    var osoba = await _context.Osobe.FindAsync(id);
    if (osoba != null)
    {
        _context.Osobe.Remove(osoba);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

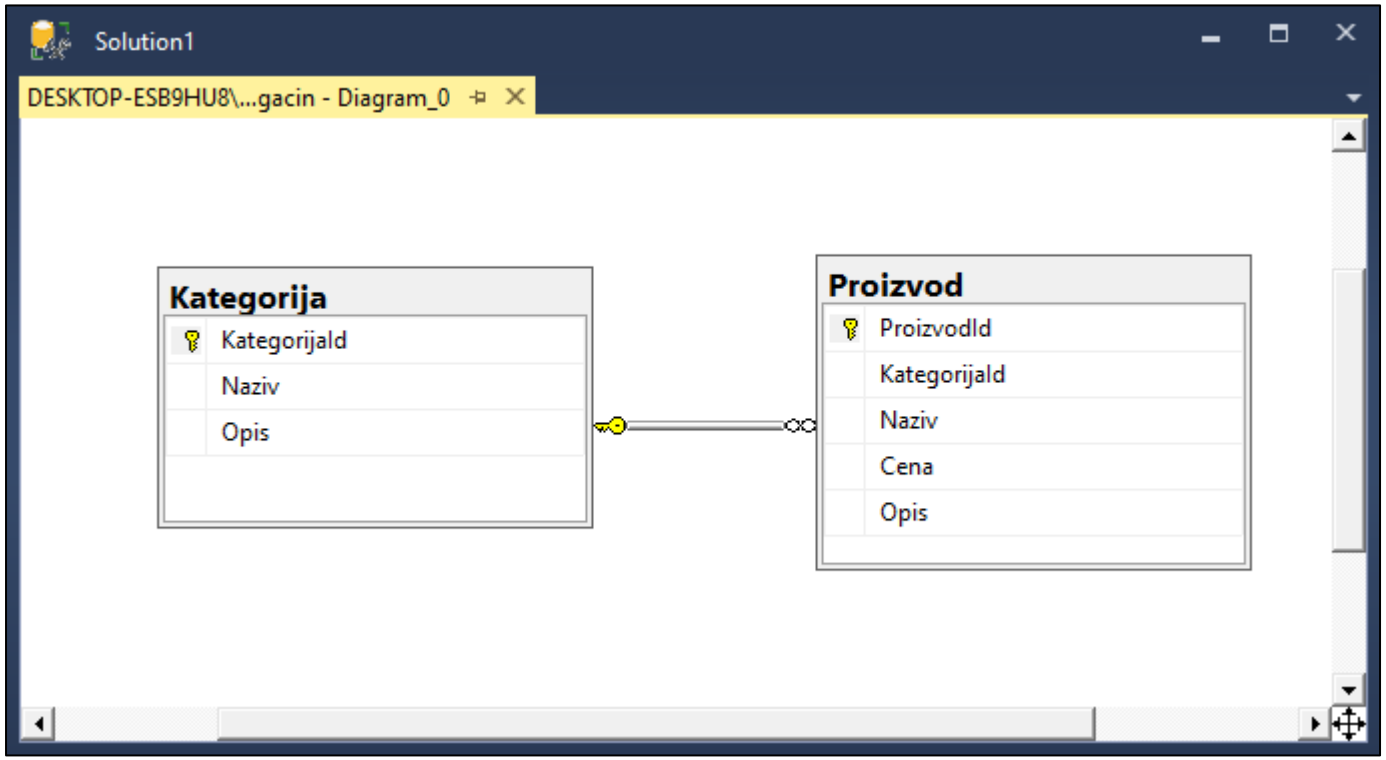
Delete POST metoda – sinhrona verzija

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public IActionResult DeleteConfirmed(int id)
{
    if (_context.Osobe == null)
    {
        return RedirectToAction("Error");
    }
    var osoba =  _context.Osobe.Find(id);
    if (osoba != null)
    {
        _context.Osobe.Remove(osoba);
    }

    _context.SaveChanges();
    return RedirectToAction(nameof(Index));
}
```

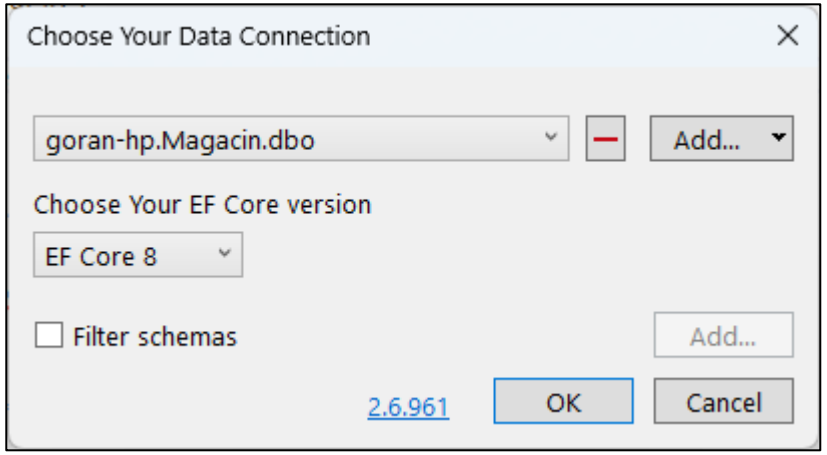
Poziv akcije Delete kontrolera Osoba

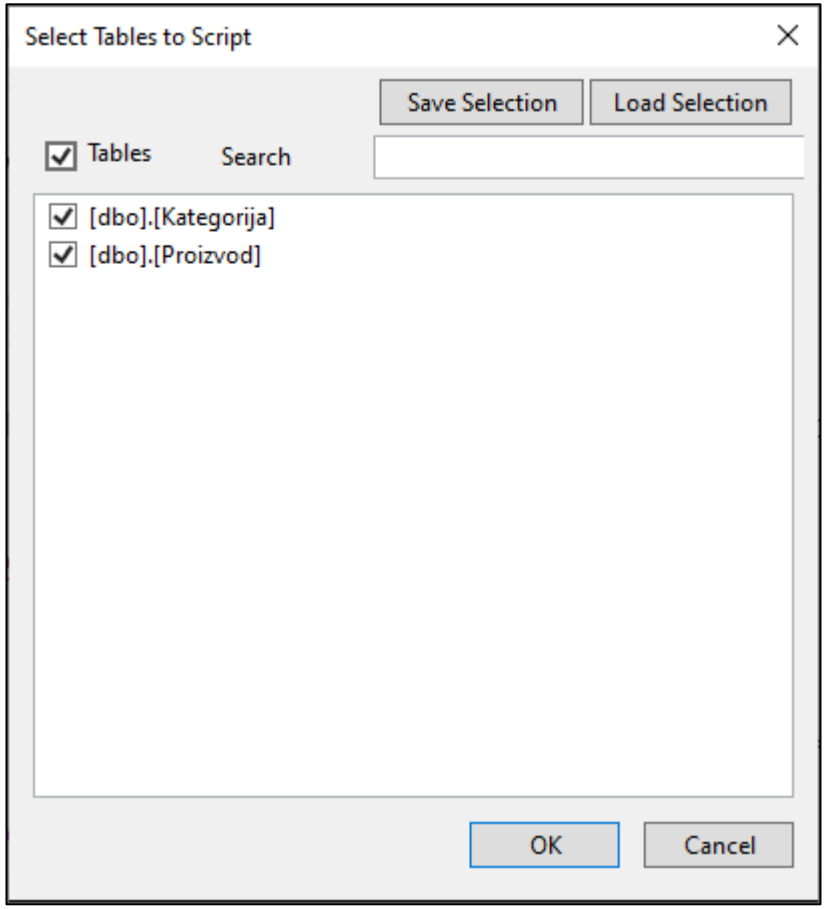


Primer filtriranja podataka – baza podataka

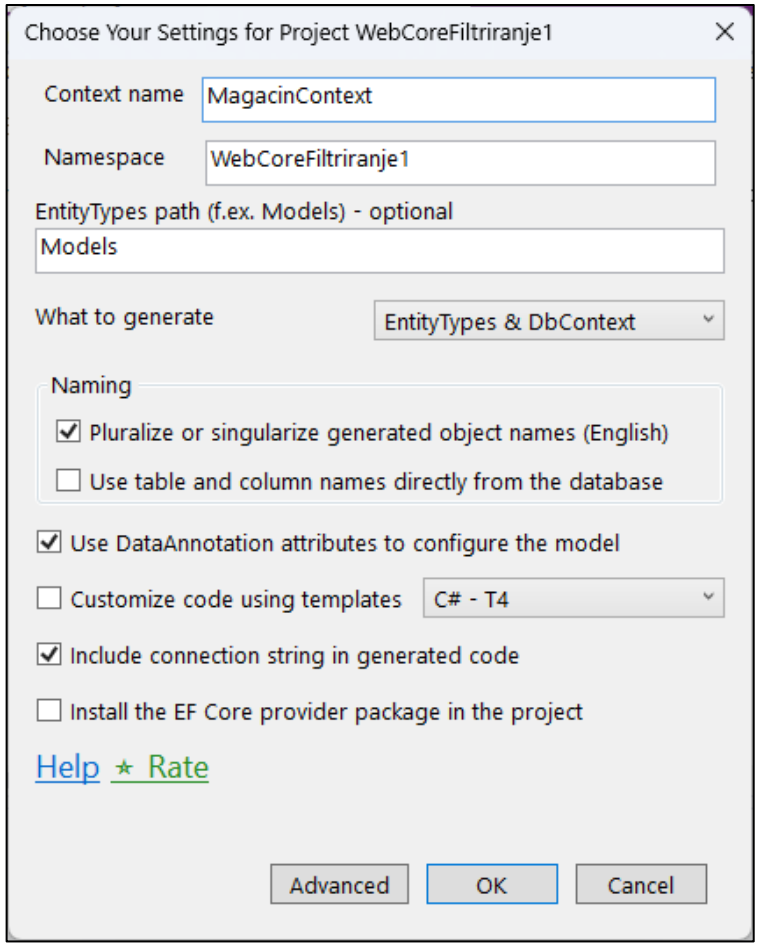


EF Core Power Tools





EF Core Power Tools



Model klasa Kategorija

```
[Table("Kategorija")]
public partial class Kategorija
{
    [Key]
    public int KategorijaId { get; set; }

    [Required]
    [StringLength(70)]
    public required string Naziv { get; set; }

    [StringLength(120)]
    public string Opis { get; set; }

    [InverseProperty("Kategorija")]
    public virtual ICollection<Proizvod> Proizvodi { get; set; } = new List<Proizvod>();
}
```

Model klasa Proizvod

```
[Table("Proizvod")]

public partial class Proizvod
{
    [Key]
    public int ProizvodId { get; set; }

    public int KategorijaId { get; set; }

    [Required]
    [StringLength(120)]
    public required string Naziv { get; set; }

    [Column(TypeName = "decimal(10, 2)")]
    public required decimal Cena { get; set; }

    [StringLength(120)]
    public string Opis { get; set; }

    [ForeignKey("KategorijaId")]
    [InverseProperty("Proizvodi")]
    public virtual Kategorija Kategorija { get; set; }
}
```

DbContext klasa

```
public partial class MagacinContext : DbContext
    {

        public MagacinContext(DbContextOptions<MagacinContext> opcije)
            : base(opcije)
        {
        }

        public virtual DbSet<Kategorija> Kategorije { get; set; }
        public virtual DbSet<Proizvod> Proizvodi { get; set; }

    }
```

appSettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=GORAN-HP;Initial Catalog=Magacin;
    Integrated Security=True;Encrypt=False"
  }
}
```

Registracija DbContext servisa

```
public static void Main(string[] args)
{
    var builder = WebApplication.CreateBuilder(args);
    var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");

    // Add services to the container.
    builder.Services.AddDbContext<MagacinContext>(options =>
                    options.UseSqlServer(connectionString));
    builder.Services.AddControllersWithViews();

    var app = builder.Build();
     ......

}
```

Klasa HomeController

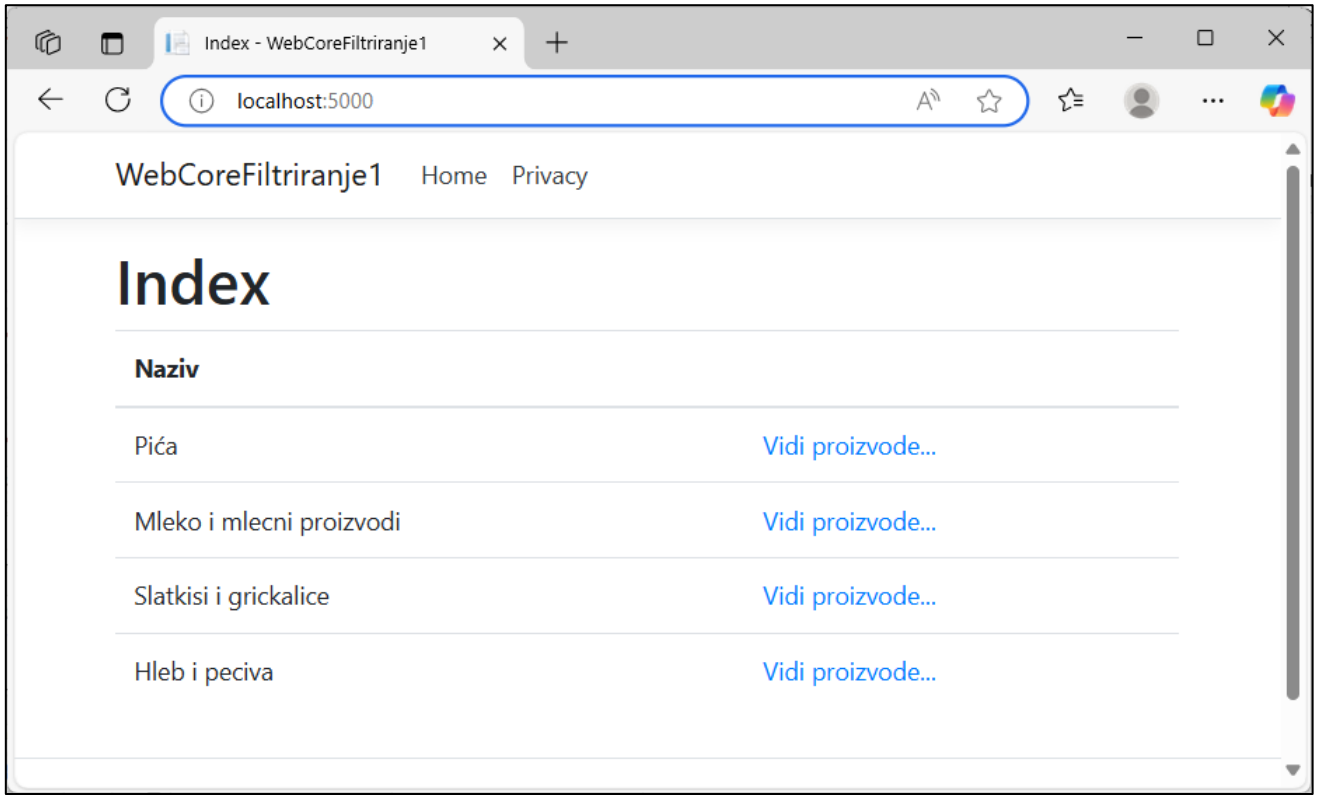
```
public class HomeController(MagacinContext _db) : Controller
{
    public IActionResult Index()
    {
        return View(_db.Kategorije.ToList());
    }
}
```

```
public class HomeController : Controller
{
    private readonly MagacinContext _db;

    public HomeController(MagacinContext db)
    {
        _db = db;
    }
    public IActionResult Index()
    {
        return View(_db.Kategorije.ToList());
    }
```

Index.cshtml

```
@model IEnumerable<Kategorija>

@{
    ViewData["Title"] = "Index";
}

<table class="table">
    <thead>
        <tr>
            <th>Naziv kategorije</th>
            <th></th>
        </tr>
  </thead>
  <tbody>
        @foreach(Kategorija k in Model)
        {
            <tr>
                <td>
                    @k.Naziv
                </td>
                <td>
                    <a href="@($"/Home/VratiProizvode/{@k.KategorijaId}")">Vidi proizvode...</a>
                </td>
            </tr>
        }
 </tbody>
</table>
```

Index pogled



Metoda za filtriranje

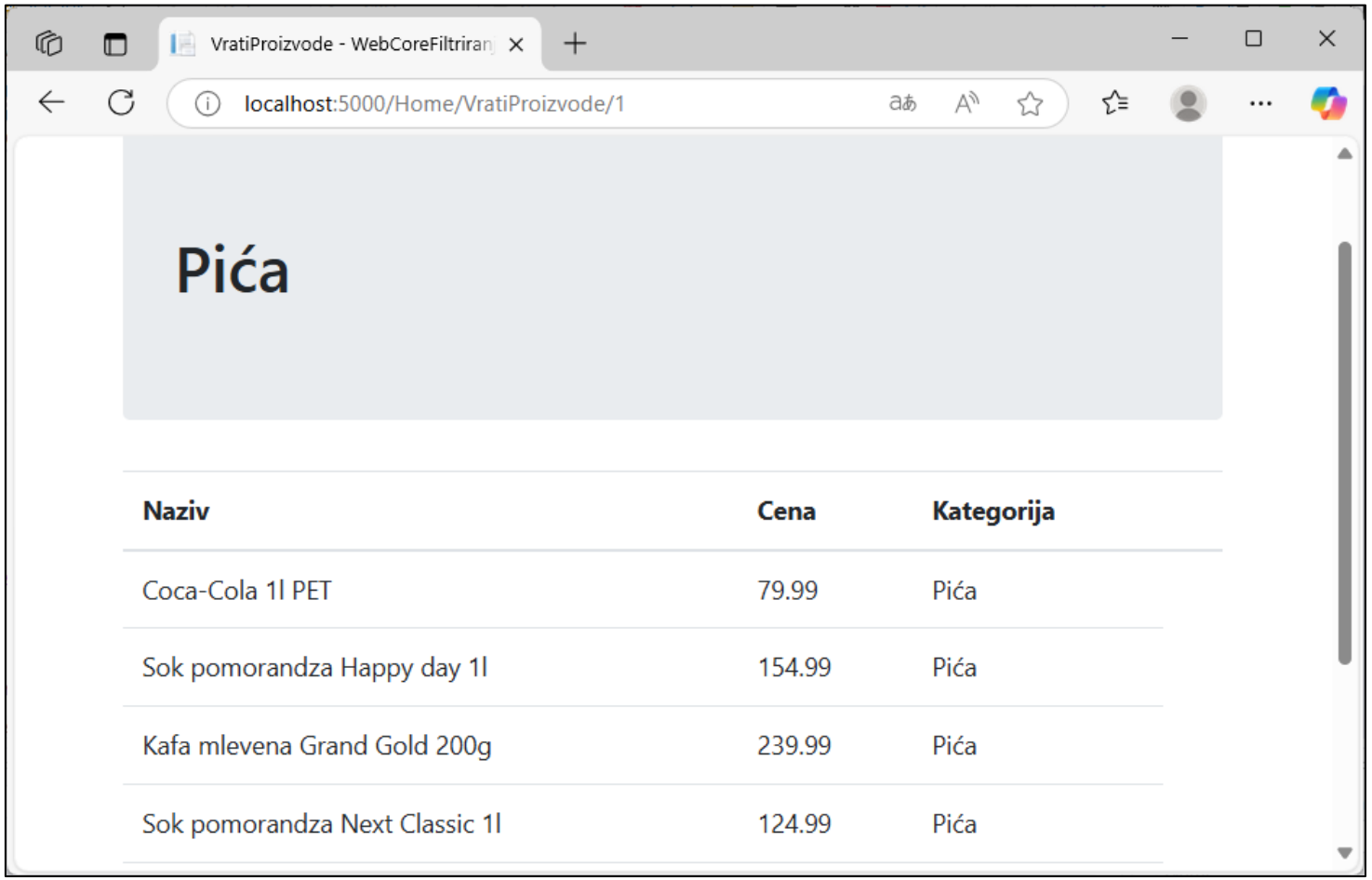
```
public IActionResult VratiProizvode(int id=0)
{
    IEnumerable<Proizvod> listaProizvoda = _db.Proizvodi;
    if (id != 0)
    {
        Kategorija? k1 = _db.Kategorije.Find(id);
        if (k1 != null)
        {
            ViewBag.Kategorija = k1.Naziv;
            listaProizvoda = listaProizvoda.Where(p => p.KategorijaId == id);
        }
        else
        {
            ViewBag.Kategorija = "";
            return View();
        }
    }
    else
    {
        ViewBag.Kategorija = "Svi proizvodi";
    }

    return View(listaProizvoda);
}
```

VratiProizvode.cshtml

```
@model IEnumerable<WebAppFiltriranje01.Models.Proizvod>

@{
    ViewData["Title"] = "VratiProizvode";
}

<div class="mt-4 p-5 bg-primary text-white rounded">
    <h1>
        @ViewBag.Kategorija
    </h1>
</div>


@if (ViewBag.Kategorija != "")
{
    <table class="table">
        <thead>
            <tr>
                <th>
                   Naziv
                </th>
                <th>
                   Cena
                </th>
           </tr>
        </thead>
```

```
 <tbody>
            @foreach (Proizvod p in Model)
            {
                <tr>
                    <td>
                        @p.Naziv
                    </td>
                    <td>
                        @p.Cena
                    </td>

                </tr>
            }
        </tbody>
    </table>
}
else
{
    <p>Ne postoji kategorija</p>
}
<br>
<a href="/Home/Index">Prikazi kategorije</a>
```

Prikaz proizvoda



Pitanje 1

ASP.NET Core MVC web aplikacija koristi podrazumevano rutiranje i izvršava se na adresi http://localhost:5000. Koju URL adresu treba uneti u browseru da bi se pozvala akcija Details kontrolera OsobaController sa prosleđenom vrednošću parametra id = 3?

a. http://localhost:5000/OsobaController/details/3
b. http://localhost:5000/Osoba/Details/3
c. http://localhost:5000/Osoba/details/?=3

Odgovor: b

Pitanje 2

Koja komponenta ASP.NET Core MVC web aplikacije je odgovorna za prijem i obradu HTTP zahteva?

a. model
b. pogled
c. kontroler

Odgovor: c

Pitanje 3

Zaokruži netačno tvrđenje. Kontrolerska klasa:

a. Ima ime koje se završava sufiksomController
b. Može biti izvedena iz klase Controller
c. Mora biti opisana atributskom klasom [Authorize]

Odgovor: c

Pitanje 4

Koji interfejs definiše povratni tip akcione metode kontrolera?

a. IDbContext
b. IActionResult
c. IJson

Odgovor: b

Pitanje 5

Ako akciona metoda kontrolera nije označena atributom koji definiše HTTP metodu, kojim HTTP zahtevom se može pozvati?

a. POST zahtev
b. GET zatev
c. I GET i POST zahtevom

Odgovor: c

Pitanje 6

Kako se naziva proces kreiranja .NET objekata na osnovu podataka iz HTTP zahteva u akcionim metodama kontrolera?

a. Controller binding
b. Model binding
c. Pogled binding

Odgovor: b

Pitanje 7

Koja je uloga GET verzije akcione metode Create u ASP.NET Core MVC aplikaciji?

a. Prikazuje korisnički interfejs (formu) za unos podataka
b. Obrađuje unesene podatke i čuva ih u bazi
c. Ažurira postojeće podatke u bazi

Odgovor: a

Pitanje 8

Kako se proverava validnost podataka prosleđenih akcionoj metodi kontrolera?

a. ModelState.IsValid
b. Model.IsValid
c. ModeBinding.IsValid

Odgovor: a