Šta je .NET i čemu služi?

- .NET je platforma za razvoj softvera koju je razvio Microsoft i koja pruža alate i biblioteke za izradu različitih vrsta aplikacija, uključujući veb, desktop, mobilne i aplikacije u oblaku. Sastoji se od okruženja za izvršenje koje se naziva Common Language Runtime (CLR) i bogate biblioteke klasa, zajedno sa razvojnim okvirima, jezicima i alatima.
- Svrha .NET-a je da programerima pruži jedinstvenu platformu za kreiranje aplikacija koje mogu raditi na različitim platformama i uređajima. Uprošćava proces razvoja pružajući konzistentan set alata, biblioteka i okvira za različite tipove aplikacija.

Korišćenje .NET-a sa Angularom

1. Razvoj Backend-a sa .NET-om

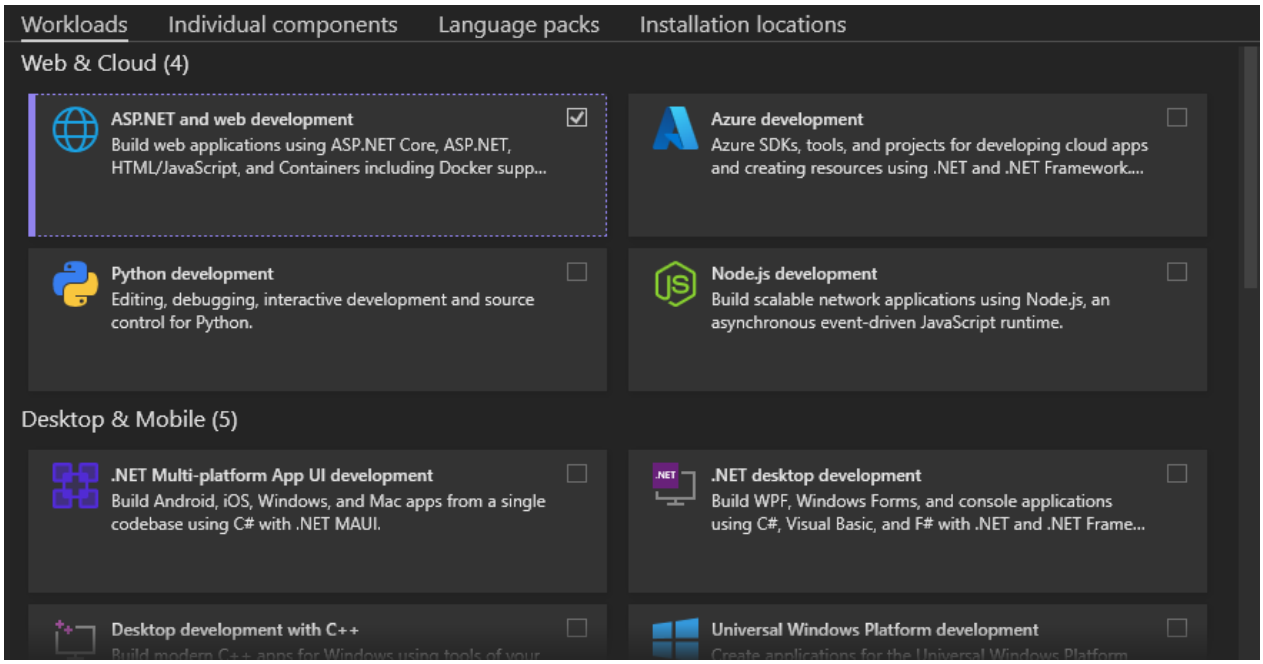
- .NET se često koristi za razvoj backend-a kako bi se izgradili RESTful API-ji, rukovalo poslovnom logikom i komuniciralo sa bazom podataka.
- ASP.NET Core, deo .NET ekosistema, često se koristi za kreiranje backend servisa i API-ja.
- .NET pruža funkcije poput Entity Framework-a za pristup bazi podataka, autentifikaciju, autorizaciju i sigurnosne mehanizme.

2. Povezivanje .NET Backend-a sa Angular Frontend-om

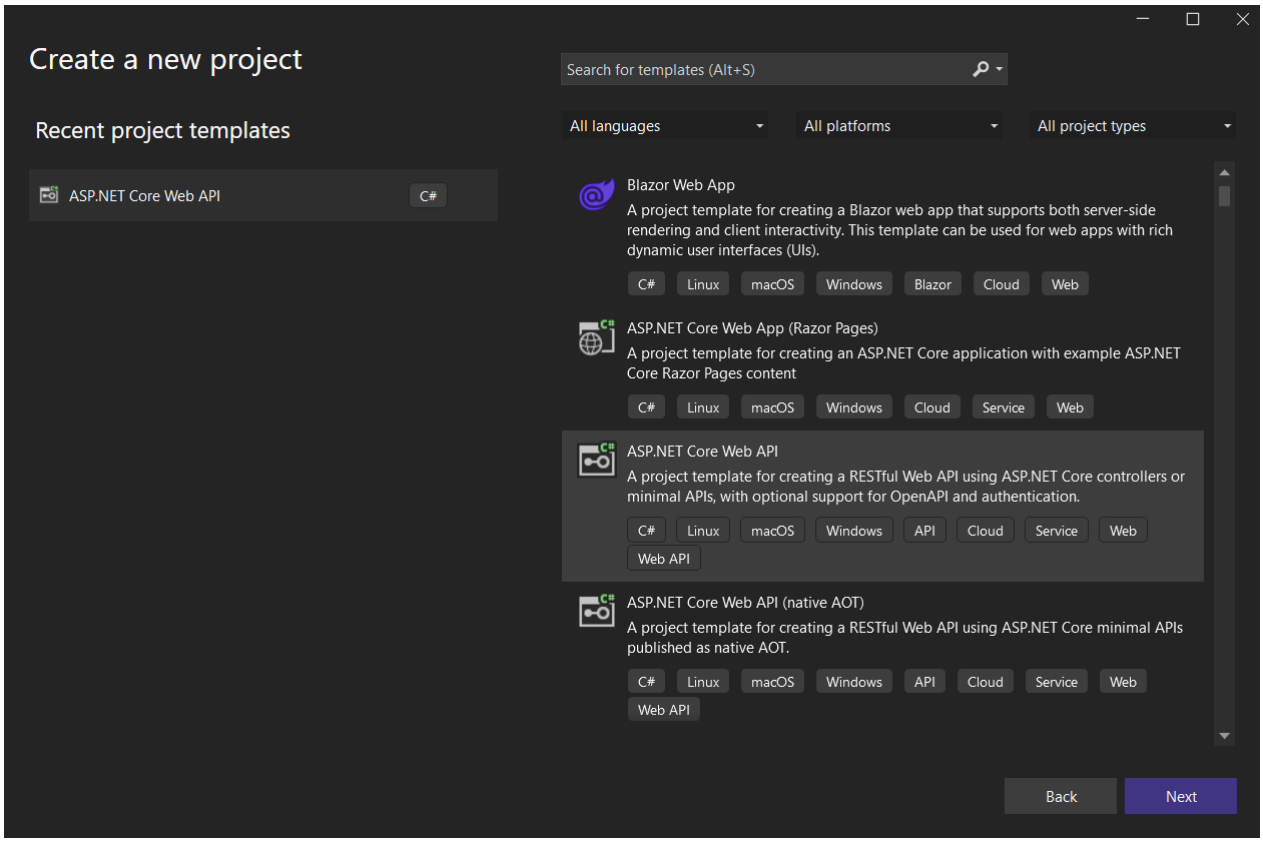
- Angular aplikacije komuniciraju sa .NET backend-om putem HTTP zahteva.
- Angular-ov HttpClient modul se koristi za slanje HTTP zahteva ka API endpoint-ima backend-a.
- Backend obrađuje ove zahteve, obavlja neophodne operacije i vraća podatke Angular aplikaciji u JSON formatu.
- CORS(Cross-Origin Resource Sharing) politike se mogu konfigurisati na backend-u kako bi se omogućili zahtevi sa Angular front-enda.

Instalacije

- Visual Studio, integrisano razvojno okruženje za .NET razvoj.
  Link: https://visualstudio.microsoft.com/downloads/

- 

- .NET SDK —Proverite da li imate .NET SDK instaliran na svom računaru. SDK uključuje sve što vam je potrebno za izgradnju i pokretanje .NET aplikacija.

- 

- Kreiranje projekta

- Entity Framework Core (opciono) — je okvir za mapiranje objekata na relacijske baze podataka koji omogućava developerima da rade sa bazama podataka koristeći .NET objekte. Iako nije neophodan, Entity Framework Core može znatno da pojednostavi interakciju sa bazama podataka u vašem .NET backend-u. Deo je NuGet paketa (osnovni deo ekosistema .NET-a, koriste se za upravljanje zavisnostima u projektima Visual Studio-a).

Osnovni koncepti rada u .NET-u

Modeli

Modeli u .NET-u su klase ili strukture koje se koriste za definisanje podataka ili entiteta unutar aplikacije. Ove klase služe za definisanje modela podataka koji se prenose između frontend i backend delova aplikacije. Kada se radi sa bazama podataka, modalne klase često služe kao "mapiranje" na tabele u bazi podataka i omogućavaju manipulaciju podacima putem ORM alata kao što je Entity Framework. Obično se u okviru Solution Explorer-a kreira novi folder u okviru projekta za smeštanje modalnih klasa.

Primer modalne klase:

```
public class Korisnik
{
    public int KorisnikId { get; set; }
    public string KorisničkoIme { get; set; }
    public string Email { get; set; }
    public string Lozinka { get; set; }
}
```

Kontroleri

Kontroleri u .NET-u, služe kao posrednici između komponenti frontend-a i backend-a aplikacije. Oni obrađuju dolazne zahteve od klijenta (Angular frontend) i formiraju odgovarajuće odgovore koji često uključuju interakciju sa modelima, servisima i drugim backend komponentama.

Uloge kontrolera:

1. Definicija tačaka (Endpoints) — Kontroleri definišu tačke sa kojima Angular frontend može da interaguje. Ove tačke obično predstavljaju metode unutar klase kontrolera.
2. Obrada zahteva — Kada backend primi zahtev, odgovarajuća metoda kontrolera obrađuje zahtev npr. preuzimanje podataka iz baze ili interakciju sa drugim servisima.
3. Generisanje odgovora — Nakon obrade zahteva, metoda kontrolera generiše odgovarajući odgovor koji se šalje klijentu (Angular frontend-u). Odgovor najčešće sadrži podatke koji treba da budu prikazani na frontend-u ili statusne kodove koji ukazuju na ishod operacije.

Kontroleri u .NET-u se zapisuju kao klase koje sadrže metode koji odgovaraju različitim HTTP zahtevima (GET, POST, PUT, DELETE).

Metode:

1. GET — Preuzima podatke sa servera. Definisane unutar klase kontrolera korišćenjem atributa [HttpGet].
2. POST — Šalje podatke radi obrade na određeni resurs. ([HttpPost])
3. PUT — Ažurira podatke na serveru. ([HttpPut])
4. DELETE — Briše podatke sa servera. ([HttpDelete])

Ove metode odgovaraju standardnim CRUD(Create, Read, Update, Delete) operacijama i koriste se za interakciju sa resursima na serveru.

Primer GET metode u kodu:

```
namespace DemoAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CitizensController : Controller
    {
        private readonly DemoDbContext _demoDbContext;

        public CitizensController(DemoDbContext demoDbContext)
        {
            _demoDbContext = demoDbContext;
        }
        [HttpGet]
        public async Task<IActionResult> GetAllCitizens()
        {
            var citizens = await _demoDbContext.Citizens.ToListAsync();

            return Ok(citizens);
        }

     }

}
```

Metoda GetAllCitizens() preuzima sve zapise građana iz baze podataka i vraća ih kao odgovor na HTTP GET zahtev poslat na endpoint api/citizens. Ona vrši upit ka tabeli Citizens u DemoDbContext kontekstu baze podataka koristeći Entity Framework Core, konvertuje rezultat u listu asinhrono(bez blokiranja, omogućava aplikaciji da izvršava druge zadatke tokom operacije) i vraća je kao telo OK(200) odgovora. Kada klijent pošalje zahtev ka API endpoint-u koji obrađuje ovu metodu, server odgovara porukom koja signalizuje da je zahtev uspešno izvršen (statusni kod 200).

CORS(Cross-origin Resource Sharing)

CORS je bezbednosna funkcionalnost implementirana u veb pregledačima kako bi se sprečilo da veb stranice prave zahteve ka drugim domenima, nego sa onog odakle su potekle. Omogućava serverima da specificiraju koji izvori imaju dozvolu da pristupe njihovim resursima. U .NET-u, CORS se implementira kako bi se kontrolisao pristup API resursima sa različitih izvora.

```
var  MyAllowSpecificOrigins = "_myAllowSpecificOrigins";

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddCors(options =>
{
    options.AddPolicy(name: MyAllowSpecificOrigins,
                      policy  =>
                      {
                          policy.WithOrigins("http://example.com",
                                             "http://www.contoso.com");
                      });
});

// services.AddResponseCaching();

builder.Services.AddControllers();

var app = builder.Build();
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();

app.UseCors(MyAllowSpecificOrigins);

app.UseAuthorization();

app.MapControllers();

app.Run();
```

- Pišemo u fajlu Startup.cs
- U policy.WithOrigins() unosimo url-ove sa kojih je dozvoljeno primanje zahteva

Rad sa bazom podataka

- Prvo treba odabrati pružaoca baze podataka koji podržava .NET okruženje. Neki od najpoznatijih su: SQL Server, MySQL, SQLite, Oracle itd.
- Instalacija Entity Framework Core-a (opciono)
- Veza sa bazom podataka se uspostavlja korišćenjem ConnectionString-a koji sadrži informacije o adresi servera i ostalim parametrima neophodnim za uspostavljanje veze. ConnectionString se najčešće upisuje u konfiguracioni fajl.

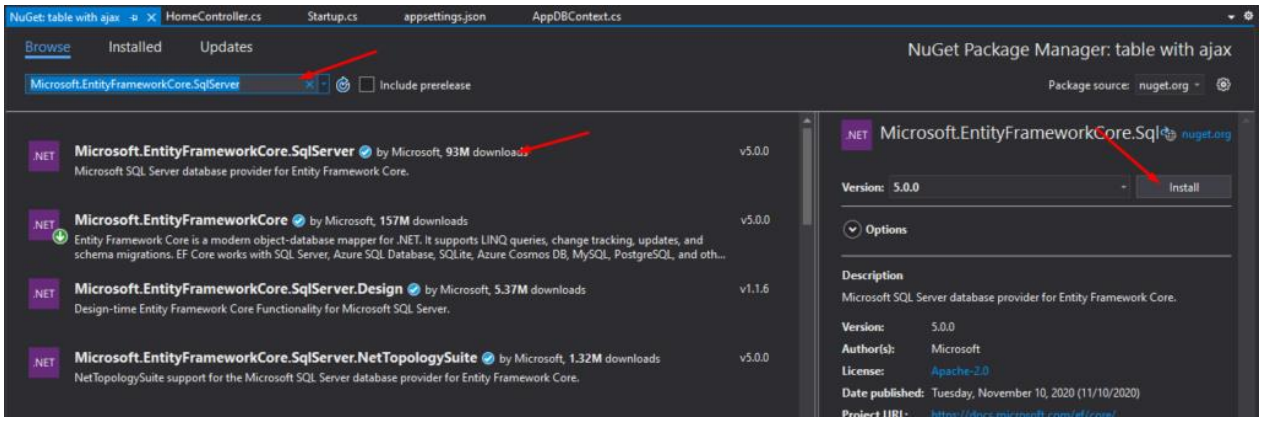
  Primer upotrebe ConnectionString-a za SQLite
  appsettings.json:

  ```
  "ConnectionStrings": {
            "DefaultConnection": "Data Source=mydatabase.db"}
  ```

  Startup.cs:

  ```
  builder.Services.AddDbContext<DemoDbContext>(option =>
  option.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"
  )));
  ```

- Pomoću EF Core migracija može da se ažurira šema baze podataka. Migracije omogućavaju definisanje promena u šemi baze podataka putem koda.

- 

- Primeri instalacije EF Core paketa u Visual Studio-u (NuGet paketi). Postoji i Microsoft.EntityFrameworkCore.Sqlite podrška za SQLite provajdera baze podataka. Microsoft.EntityFrameworkCore.Tools paket sadrži komande za upravljanje migrcijama baze podataka i generisanje koda na osnovu postojeće šeme baze podataka.