Laboratorijska Vježba 5.
ASP.NET Core MVC
Uvod i Konfiguracija Okruženja

Cilj vježbe:

- Upoznavanje sa MVC arhitekturalnim paternom za web-aplikacije;
- Kreiranje prve ASP.NET MVC aplikacije.

1. MVC arhitekturalni patern za web-aplikacije

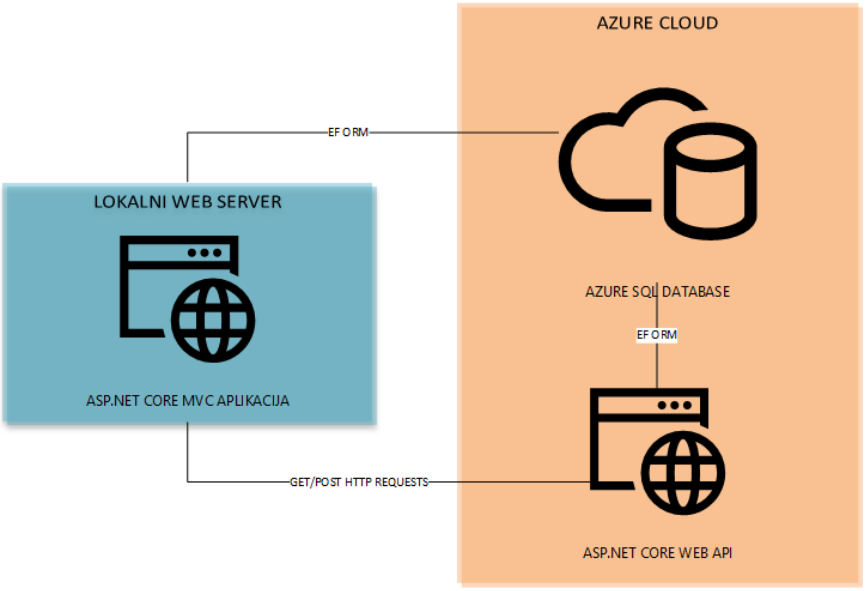
U prethodnim laboratorijskim vježbama prikazan je način kreiranja konzolnih aplikacija i biblioteka klasa. Iako korisne za vježbu, ove aplikacije nisu dovoljne kako bi se uspješno kreirala web-aplikacija za rad sa velikim brojem korisnika koji koriste različite tehnologije za pristup korisničkom interfejsu. Ova vježba predstavlja kratak uvod u **ASP.NET Core MVC** platformu, gdje će se kroz praktičan primjer pokazati osnovne funkcionalnosti ove platforme u kombinaciji sa *Entity Framework Core* metodom za interakciju sa bazom podataka.

U vježbi će biti korišten jednostavan model sa 3 entiteta koji su u vezi jedan-na-više (*one-to-many*). Koristiti će se tzv. *code-first* pristup, što podrazumijeva da se prvo kreira objektni model na osnovu kojeg se generiše šema relacione baze podataka upotrebom *Entity Framework* mapiranja, čime se izbjegava manuelno generisanje šeme (što može biti veoma zahtjevan posao podložan velikom broju grešaka). Nakon toga generisati će se CRUD operacije za rad sa objektima baze podataka (*Create – Read – Update – Delete* akcije), tj. kreirati će se odgovarajući kontroleri (*controllers*) i pogledi (*views*) ASP.NET Core web-aplikacije. Na Slici 1 prikazana je arhitektura aplikacije koja se želi kreirati, koja se sastoji iz sljedećih dijelova:

- **ASP.NET Core MVC web-aplikacija**, koja predstavlja okruženje preko kojeg će se vršiti interakcija sa korisnikom. U okviru ove aplikacije iskoristiti će se MVC patern za razdvajanje modela (klasa mapiranih iz baze podataka) od kontrolera (klasa za obradu podataka, prikaz rezultata i interakciju s korisnikom) i pogleda (klasa koje vrše prikaz informacija korisniku).

- **SQL baza podataka**, koja predstavlja odvojen entitet u okviru kojeg su definisane tabele podataka koje logički odgovaraju klasama koje će se koristiti u web-aplikaciji. Ova baza podataka nalazi se na zasebnom serveru i može predstavljati lokalnu bazu podataka (npr. *Microsoft SQL Server* instancu) ili neku *cloud* bazu podataka (npr. *Azure Database*).

- **ASP.NET Core Web Api aplikacija**, koja predstavlja odvojen entitet u okviru kojeg su definisane funkcionalnosti kojima nije moguće pristupiti direktno u okviru web-aplikacije, a neophodne su za ispravan rad sistema. U okviru njih može se podržati korištenje web-zahtjeva putem *Postmana* (bez potrebe za korištenjem web-aplikacije) ili razdvajanje funkcionalnosti koje trebaju veći nivo sigurnosti pa ih je korisno izdvojiti u posebnu aplikaciju. Moguće je i korištenje vanjskih servisa i njihova enkapsulacija u jednostavne zahtjeve zbog lakšeg korištenja u web-aplikaciji.



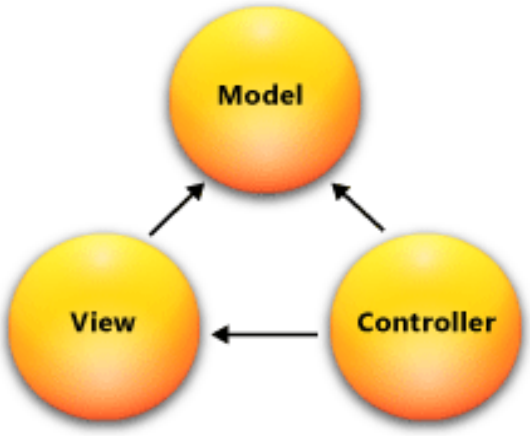
Slika 1. Arhitektura željene ASP.NET aplikacije

MVC je standardni arhitekturalni dizajn patern. ASP.NET MVC je *Microsoft* implementacija MVC paterna za izgradnju web-baziranih aplikacija. Osnovne komponente MVC *framework*-a su (prikazano na Slici 2):

- **Model** – dio aplikacije u kojem se implementira aplikacijska logika u pogledu spremanja podataka. Najčešće se koristi za dobavljanje i spremanje stanja aplikacije u bazu podataka.

- **View** – korisnički interfejs (UI) i komponente koje omogućavaju prikaz. Obično se pogledi kreiraju na osnovu modela podataka i mogu sadržavati elemente poput *text box*, *drop-down list*, *button* i sl. U pogledima treba postojati minimalna programska logika koja omogućava preusmjeravanje rada na kontrolere ili koja direktno ažurira model (vrši manipulaciju nad podacima).

- **Kontroler** – centralni dio sistema, odnosno veza modela sa pogledom (*view*). U kontrolerima se obično nalazi poslovna logika i kompleksne funkcionalnosti za obradu podataka kako bi se model i/ili pogledi mogli ažurirati.

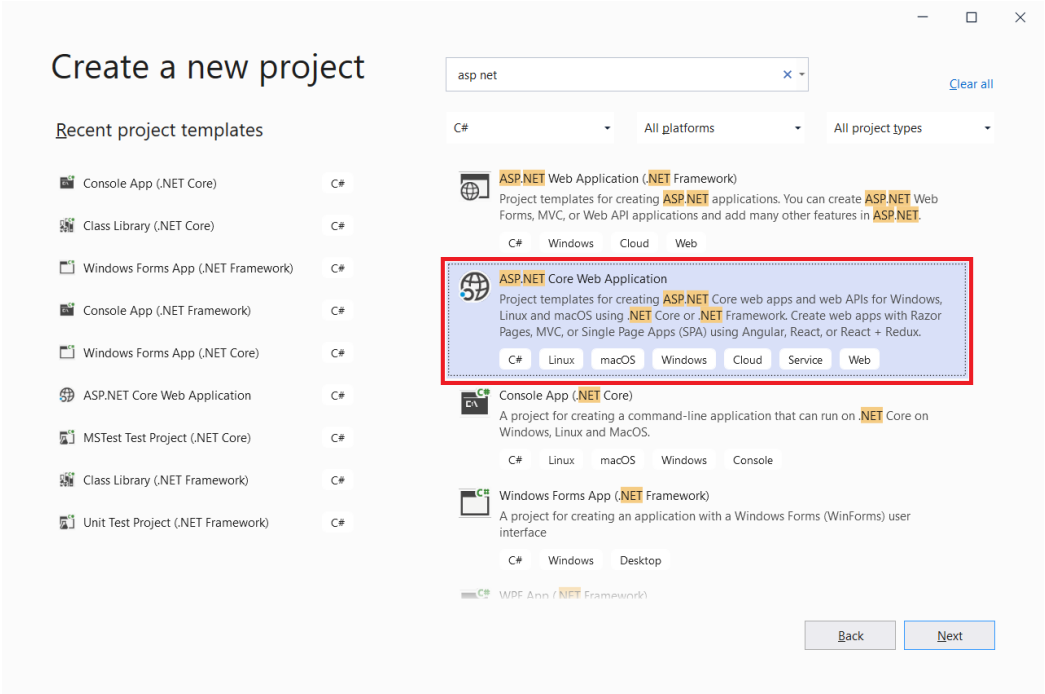


Slika 2. Osnovni prikaz MVC paterna

MVC patern omogućava kreiranje aplikacija koje odvajaju različite aspekte arhitekture aplikacije, tj. ulaznu logiku, poslovnu logiku i UI logiku, pri čemu se omogućava labava veza (slaba međuzavisnost) između navedenih elemenata. To između ostalog omogućava i paralelni razvoj gdje jedan programer može raditi na razvoju jednog sloja, drugi drugog, itd., kao i lokalizaciju grešaka na način da se greške napravljene u okviru jednog dijela sistema ne propagiraju na ostale dijelove. U narednim laboratorijskim vježbama biti će obrađene sve važne mogućnosti koje ASP.NET MVC platforma nudi (od automatskog generisanja koda, principa rutiranja i responzivnog web-dizajna do naprednih mogućnosti za sigurnost aplikacije), a u okviru ove laboratorijske vježbe izvršiti će se konfiguracija okruženja i kreirati prva ASP.NET Core MVC aplikacija.

2. Prva ASP.NET Core MVC aplikacija

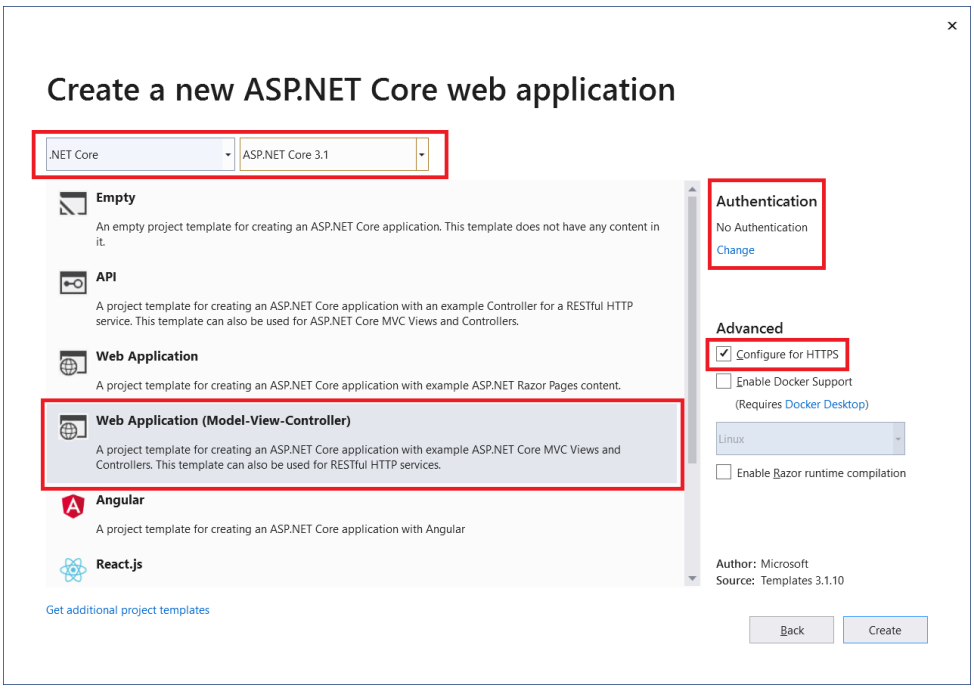
U drugoj laboratorijskoj vježbi izvršena je instalacija svih komponenti neophodnih za kreiranje ASP.NET web-aplikacija. Sada je potrebno, na isti način kao u prethodnim vježbama, pokrenuti okruženje i odabrati opciju za kreiranje novog programskog rješenja. Nakon toga vrši se odabir željene vrste projekta – najlakše je izvršiti pretraživanje za pojam *asp net*, nakon čega je potrebno odabrati tip projekta **ASP.NET Core Web Application**, na način prikazan na Slici 3. Potrebno je voditi računa da se ne odabere *.NET Framework* verzija za ASP.NET aplikacije.



Slika 3. Odabir vrste projekta za ASP.NET Core MVC aplikaciju

Nakon toga specificiraju se osnovne informacije o programskom rješenju, što se radi na isti način kao u okviru prethodnih laboratorijskih vježbi. Nakon toga pojavljuje se novi ekran u kojem je potrebno odabrati **vrstu web-aplikacije** koja se želi kreirati. Postoji mnogo različitih opcija, kao npr. kreiranje samo komponenti pogleda (*Angular* i *React.js* aplikacije) ili kreiranje web API servisa (koji ne sadrži poglede), o čemu će biti riječi u nekoj od narednih laboratorijskih vježbi. Kako se sada želi kreirati web-aplikacija sa svim komponentama MVC arhitekturalnog pogleda, potrebno je odabrati isključivo opciju **Web Application (Model-**

View-Controller). U zaglavlju je potrebno provjeriti da je odabrana platforma **.NET Core** i njegova najnovija verzija. Kako se u ovoj vježbi neće koristiti principi sigurnosti, *Authentication* opciju koja se nalazi s desne strane nije potrebno mijenjati, već je potrebno ostaviti na **No Authentication**. Konfiguracija za sigurnu konekciju koristeći HTTPS treba biti omogućena. Sve postavke konfiguracije nove ASP.NET aplikacije prikazane su na Slici 4.



Slika 4. Konfiguracija nove ASP.NET aplikacije

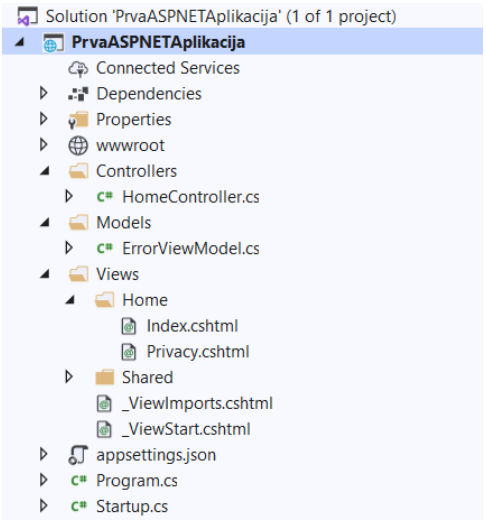
Ovime je proces konfiguracije nove ASP.NET aplikacije završen i sada se može početi programirati. Prvenstveno je potrebno moći se navigirati kroz okruženje, pa će se prvo izvršiti analiza strukture samog programskog rješenja. Kao što je vidljivo na Slici 4, programsko rješenje sadrži tri foldera:

- **Controllers folder**, koji inicijalno ima definisan jedan *file* imena *HomeController*. Ovaj kontroler omogućava navigaciju kroz početne forme aplikacije i prikaz osnovnih informacija o predefinisanoj aplikaciji, što će biti analizirano u nastavku. U ovaj folder se dodaju svi kontroleri koji definišu programsku logiku aplikacije.

- **Models folder**, koji inicijalno ima definisan jedan *file* imena *ErrorViewModel*. Ovaj model omogućava prikaz grešaka u web-zahtjevima. U ovaj folder se dodaju sve modelske klase kojima se mapiraju entiteti iz baze podataka.

- **Views folder**, koji inicijalno ima definisane poglede definisane u folderu *Home* i zajedničke postavke za poglede definisane u folderu *Shared*. Kao što se može primijetiti, ime foldera koji sadrži poglede odgovara imenu definisanog kontrolera, čime se omogućava njihova međusobna komunikacija. U ovaj folder dodaju se svi folderi sa formama koje se prikazuju korisniku i sve stilske postavke koje se odnose na neke ili sve forme aplikacije.

Osim ovih foldera, aplikacija posjeduje i zasebne *file*-ove koji sadrže važne postavke. **appsettings.json** sadrži sve konfiguracijske elemente aplikacije i koristi se za dodavanje statičkih stringova koji se koriste na više mjesta u aplikaciji (poput npr. stringa za konekciju na bazu podataka). **Program** je klasa koja definiše šta se dešava pri pokretanju aplikacije, što može biti veoma važno kada je potrebno izvršiti migraciju na bazu podataka ili poslati neki zahtjev pri početku rada aplikacije. **Startup** klasa definiše sve servise koji se koriste u aplikaciji. Ukoliko se neki servis ne registruje u ovoj klasi, neće ga biti moguće koristiti pri radu aplikacije.



Slika 5. Struktura programskog rješenja

Sada je potrebno analizirati elemente od kojih se aplikacija sastoji. Aplikacija ne posjeduje model podataka niti je povezana na bazu podataka. Posjeduje dva pogleda – **Index** i **Privacy**, a pristup istima moguć je kroz **Home** kontroler. Struktura ovog kontrolera prikazana je u isječku koda ispod, odakle je vidljivo da ovaj kontroler posjeduje tri metode – *Index*, *Privacy* i *Error*. Sve tri metode vraćaju poglede, čime se zapravo vrši interakcija između kontrolera i pogleda. U narednim laboratorijskim vježbama biti će objašnjen način na koji se kreiraju metode u kontrolerima koje vraćaju poglede i prikazuju ih korisniku, koje primaju podatke od korisnika i vrše njihovu obradu, kao i koje vrše prebacivanje na neki drugi kontroler.

```
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;

    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
    }

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

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

```
        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore
= true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
```

Index i *Privacy* pogledi koji se vraćaju kao rezultat metoda kontrolera imaju svoj predefinisani oblik. Pogledi se kreiraju koristeći kombinaciju HTML *markup* koda za definiciju elemenata forme, CSS koda za stil elemenata i C# programskog jezika. Deklaracija ovih pogleda veoma je jednostavna – cijela stranica *Index* prikazana je u isječku koda ispod, a u okviru narednih laboratorijskih vježbi biti će objašnjen način kreiranja kompleksnih formi sa različitim elementima za prikaz informacija i prihvatanje korisničkog unosa.

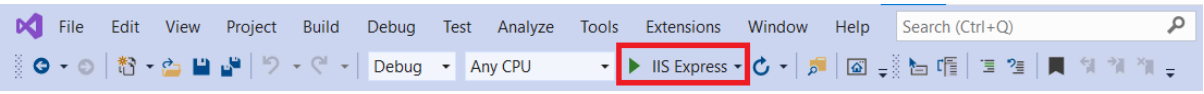
```
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web
apps with ASP.NET Core</a>.</p>
</div>
```

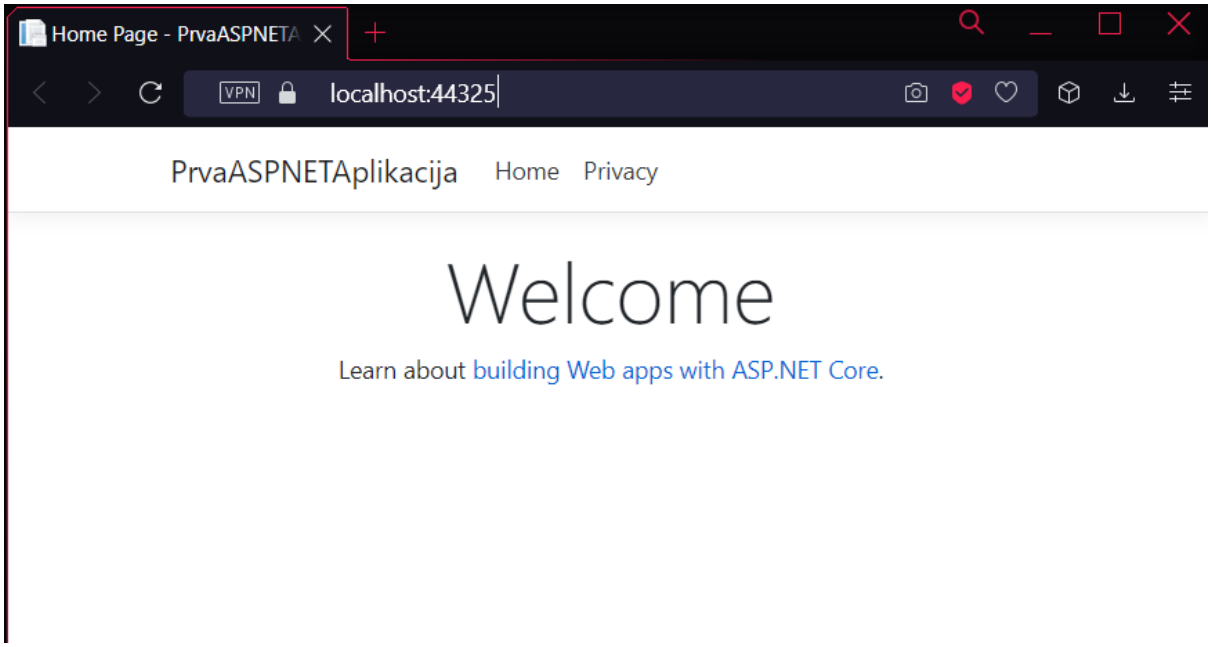
U klasi **Startup** nalazi se definicija *default* rute na koju se stranica naviguje kada se prvi put pokrene, kao što je prikazano u isječku koda ispod. Predefinisana ruta je: **{controller=Home}/{action=Index}/{id?}**, što zapravo znači da će se aplikacija navigirati na rutu **web-adresa/Home/Index**, pri čemu je ruta Index podrazumijevana ukoliko se ne navede (i ruta **web-adresa/Home** ili samo ruta **web-adresa** vrši navigaciju na Home kontroler i Index stranicu). Kako web-aplikacija nema nikakav *deployment* i pokreće se lokalno, njena *default* web-adresa uvijek će biti **localhost:port**, pri čemu je *port* predefinisani broj koji omogućava pristup aplikaciji. Pri prvom pokretanju aplikacija će se automatski otvoriti na predefinisanom portu.

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

Predefinisanu aplikaciju moguće je pokrenuti bez obzira na činjenicu što ne posjeduje konekciju na bazu podataka niti definisane modele podataka. Aplikacija se pokreće na isti način kao i sve ostale vrste aplikacija, odabirom opcije **IIS Express** u *Visual Studio* okruženju, na način prikazan na Slici 6. Nakon nekoliko sekundi, aplikacija se automatski pokreće i prikazuje se *default* ruta koja je definisana u *Startup* klasi. U ovom slučaju ta ruta vrši pristup *Home* kontroleru i njegovoj *Index* metodi, koja kao rezultat vraća *Index* pogled sa definicijom HTML elemenata. Kao što je vidljivo na Slici 7, aplikacija je pokrenuta na web-adresi **localhost:44325**, što je *default* adresa za ovu web-aplikaciju pri svakom pokretanju. Prikazana je osnovna stranica koja posjeduje nekoliko kontrola – meni za navigaciju i naslov sa opisom.

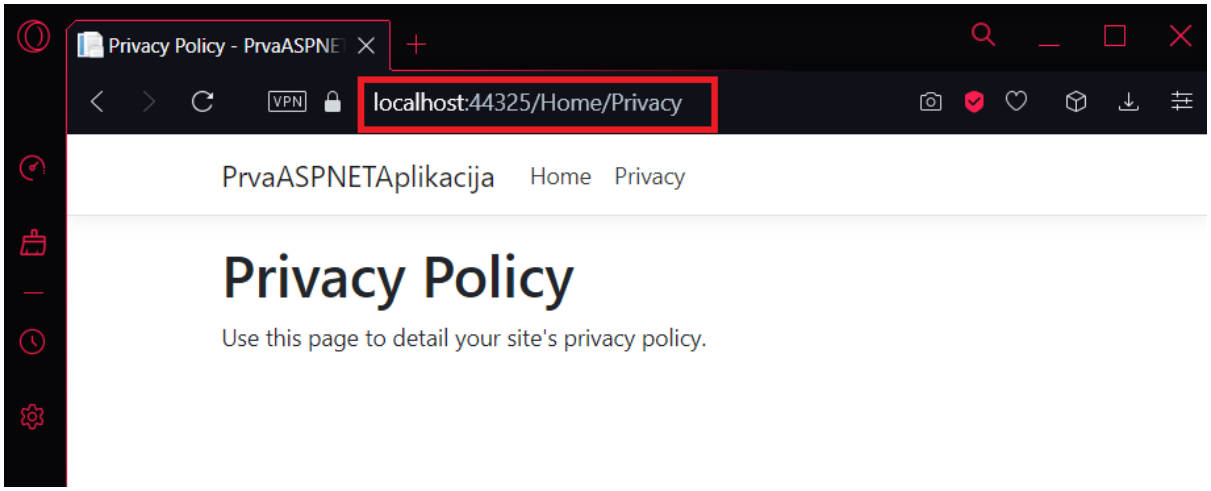


Slika 6. Način pokretanja ASP.NET aplikacije



Slika 7. Prikaz predefinisane Index stranice

Kako kontroler posjeduje definisanu metodu *Privacy* koja vraća pogled *Privacy* koji je definisan, moguće je izvršiti navigaciju (preko forme ili manuelnim unosom web-adrese) na rutu **localhost:44325/Home/Privacy**, čime se kao rezultat dobiva prikaz ove stranice, na način prikazan na Slici 7. I ova forma veoma je jednostavna i posjeduje definiciju menija, naslova i opisa.



Slika 8. Prikaz predefinisane Privacy stranice

Ovako definisana aplikacija predstavlja osnov koji je moguće mijenjati i nadograđivati prema želji korisnika. *Home* kontroler lako je proširiti tako da posjeduje još metoda koje vraćaju druge poglede, kao i metode kojima se vrši obrada unosa podataka sa neke od ovih formi. Također je moguće definisati druge kontrolere i jednostavno vršiti navigaciju između njih. Naposlijetku je potrebno kreirati i model podataka nad kojim će se vršiti obrada kao i prikaz. Sve ovo biti će obrađeno u okviru narednih laboratorijskih vježbi.

3. Zadaci za samostalni rad

Programski kod aplikacije koja je kreirana u vježbi dostupan je u repozitoriju na sljedećem linku: https://github.com/ehlymana/OOADVjezbe, na *branchu* **lv5**.

1. Istražiti MVC arhitekturalni patern izvan upotrebe u okviru ASP.NET aplikacija. Da li se komunikacija između različitih komponenti vrši na isti način? U čemu je glavna razlika?

2. Kreirati prvu ASP.NET Core MVC aplikaciju na način opisan u vježbi. Uspješno pokrenuti aplikaciju i izvršiti navigaciju kroz predefinisane poglede.

3. Promijeniti *default* rutu na Home/Privacy. Da li je vršenje promjene rute jednostavno?

4. Dodati novu formu *Information* sa proizvoljnim sadržajem. Šta je potrebno uraditi kako bi se ova forma mogla uspješno prikazivati u aplikaciji? Gdje je potrebno smjestiti *file* koji sadrži definiciju ovog pogleda?