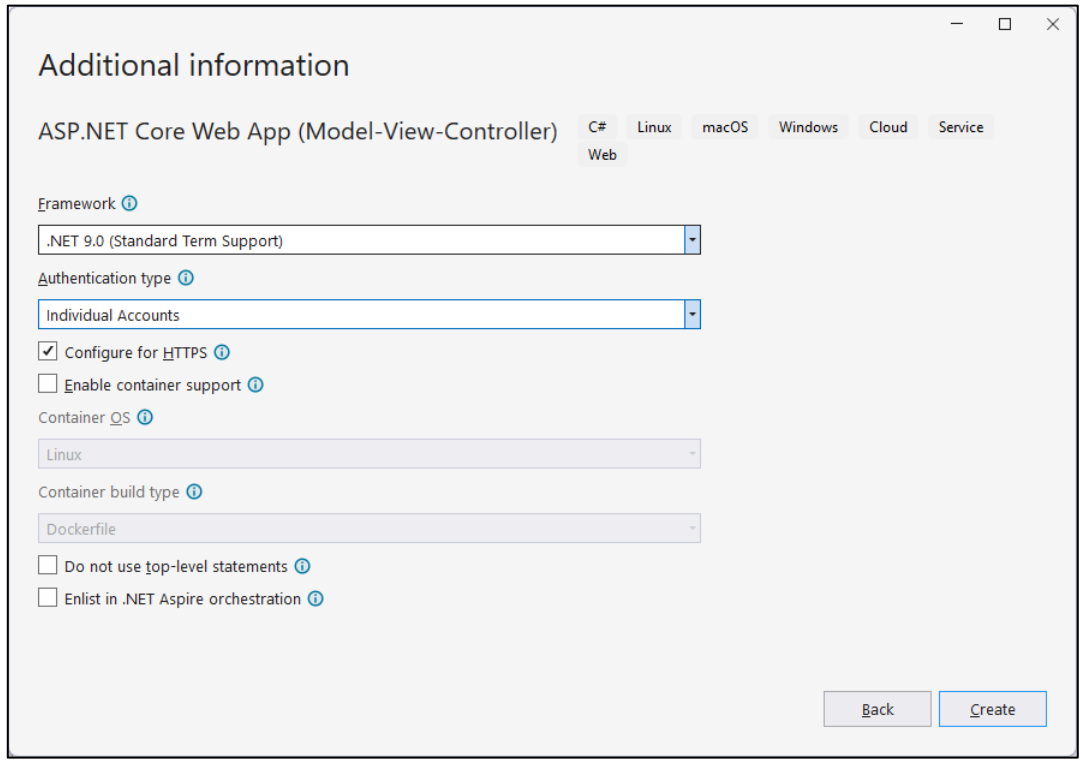
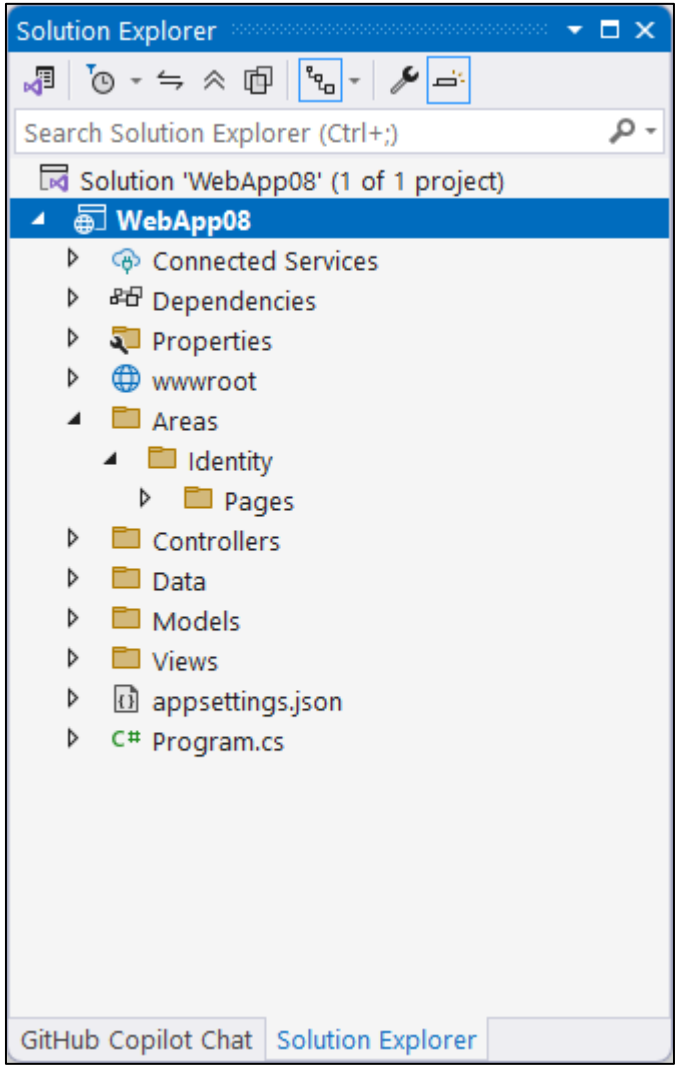
Sigurnost ASP.NET Core MVC web aplikacija

Kreiranje ASP.NET Core web aplikacije



- **Individual User Accounts** koristi ASP.NET Core Identity sistem
- Omogućava registraciju i prijavu korisnika
- Upravljanje lozinkama i sigurnošću naloga
- Podrška za autentifikaciju i autorizaciju
- Kontrola pristupa delovima aplikacije

Aplikacija sa podrškom za logovanje



- Ugrađena ASP.NET Core Identity integracija
- Areas/Identity → login, registracija, logout funkcionalnosti
- Data folder → rad sa bazom i korisničkim podacima
- Gotova autentifikacija bez dodatne implementacije
- Mogućnost proširenja (uloge, prava pristupa)

ASP.NET Core Identity sistem

- Visual Studio 2022 implementira **Identity sistem** za logovanje kao biblioteku baziranu na tehnologiji **Razor Pages** (a ne na ASP.NET Core MVC tehnologiji)
- Implementacija koristi code-first pristup, pri čemu se kroz migracije kreira baza podataka za autentifikaciju
- Koristi se biblioteka **Microsoft.AspNetCore.Identity**
- Korisnik sistema predstavlja se klasom **IdentityUser** (koja se obično proširuje pri prilagođavanju sistema)
- Podrazumevano, korisnik se prijavljuje korišćenjem email adrese (koja ujedno predstavlja korisničko ime) i lozinke
- Prilikom logovanja, korisnik unosi email adresu i lozinku

Kastomizacija Identity sistema

- U cilju uklanjanja Razor Pages pristupa i prelaska na MVC, vrši se kastomizacija ASP.NET Core Identity sistema
- Korisnik veb aplikacije biće predstavljen klasom **ApplicationUser** koja nasleđuje **IdentityUser**
- Klasa za rad sa bazom biće izvedena iz klase **IdentityDbContext<ApplicationUser>**
- Klasa **UserManager<ApplicationUser>** koristi se za upravljanje korisnicima (kreiranje, izmene, pretraga)
- Klasa **SignInManager<ApplicationUser>** koristi se za autentifikaciju korisnika (login/logout)

Fajl appsettings.json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=GORAN-HP;Database=aspnet-WebApp08;
     Integrated Security=True;Encrypt=False"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Konfiguracija baze i Identity servisa

```
var connectionString =
builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

var identityBuilder =
builder.Services.AddDefaultIdentity<IdentityUser>(options =>
    options.SignIn.RequireConfirmedAccount = true);

identityBuilder.AddEntityFrameworkStores<ApplicationDbContext>();

builder.Services.AddControllersWithViews();
```

Konfiguracija baze i Identity servisa

- Klasa **ApplicationDbContext** nasleđuje klasu **IdentityDbContext**<IdentityUser> i predstavlja klasu za pristup bazi podataka
- Klasa **ApplicationDbContext** se registruje u DI kontejneru pomoću metode **AddDbContext** i povezuje se sa SQL Server bazom
- Metodom **AddDefaultIdentity**<IdentityUser>() kreira se objekat tipa **IdentityBuilder** i registruju se osnovni Identity servisi
- Nad objektom identityBuilder poziva se metoda **AddEntityFrameworkStores**<ApplicationDbContext>(), čime se definiše da Identity koristi klasu **ApplicationDbContext** za rad sa bazom
- Opcijom RequireConfirmedAccount = true zahteva se potvrda naloga pre prijave

Modifikacija pravila za lozinku i pravila za korisnika

```
var identityBuilder =
builder.Services.AddDefaultIdentity<IdentityUser>(options =>
{

    // Podesavanje lozinke
    options.Password.RequireDigit = false;
    options.Password.RequiredLength = 3;
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequireUppercase = false;
    options.Password.RequireLowercase = false;
    options.Password.RequiredUniqueChars = 1;
    // Podesavanje korisnika
    options.User.RequireUniqueEmail = false;
});
```

Modifikacija pravila za lozinku i pravila za korisnika

- Pravila za lozinku definisana u prethodnom slajdu koriste se isključivo za lakše testiranje sistema
- Namerno su ublaženi bezbednosni zahtevi za lozinku
- Podrazumevano, sistem zahteva da lozinka:
  - ima najmanje 6 karaktera
  - sadrži veliko slovo
  - sadrži malo slovo
  - sadrži specijalan karakter (nije slovo ni broj)

Podešavanje cevovoda obrade zahteva

```
app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapStaticAssets();

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

app.Run();
```

Podešavanje cevovoda obrade zahteva

- Definiše se redosled obrade HTTP zahteva u aplikaciji
- Metodom UseAuthentication () dodaje se komponenta za autentifikaciju
- Metodom UseAuthorization () dodaje se komponenta za autorizaciju

Klasa ApplicationDbContext folder Data

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace WebApp08.Data
{
    public class ApplicationDbContext : IdentityDbContext
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }
    }
}
```

- Klasa **ApplicationDbContext** predstavlja DbContext koji koristi Identity system za rad sa bazom podataka
- Nasleđuje klasu **IdentityDbContext**

Klasa ApplicationUser folder Models

```
namespace WebApp08.Models
{
    public class ApplicationUser : IdentityUser
    {
        [Required]
        [StringLength(30)]
        public string FirstName { get; set; }

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

    }
}
```

- Klasa **ApplicationUser** nasleđuje klasu **IdentityUser** i proširuje je dodatnim atributima
- Dodata su svojstva **FirstName** i **LastName** za čuvanje imena i prezimena korisnika
- Na ovaj način se osnovni **Identity** model prilagođava potrebama aplikacije

Korišćenje klase ApplicationUser u Identity sistemu

```
// Add services to the container.
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

var identityBuilder =
builder.Services.AddDefaultIdentity<ApplicationUser>(options =>
{

    // Podesavanje lozinke
    options.Password.RequireDigit = false;
    options.Password.RequiredLength = 3;
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequireUppercase = false;
    options.Password.RequireLowercase = false;
    options.Password.RequiredUniqueChars = 1;
    // Podesavanje korisnika
    options.User.RequireUniqueEmail = false;
});
```

- Klasa **IdentityUser** zamenjena je klasom **ApplicationUser**
- Omogućeno je proširenje korisničkog modela dodatnim atributima
- Identity sistem sada koristi prilagođenu klasu korisnika

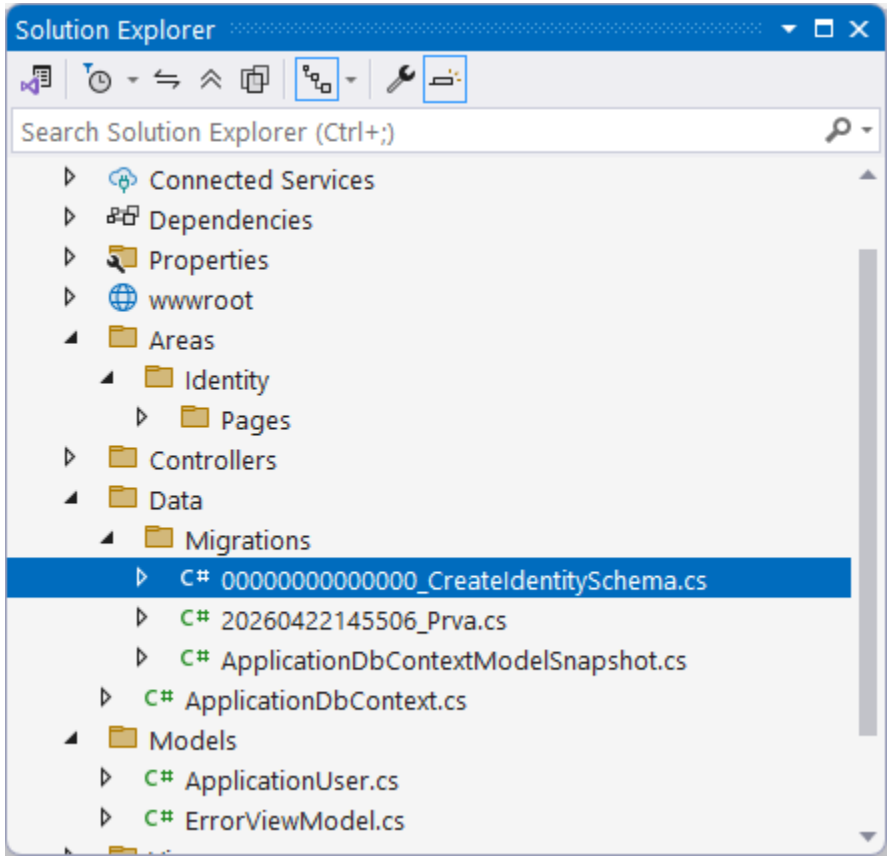
Modifikovani ApplicationDbContext

```
namespace WebApp08.Data
{
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }
    }
}
```

- Klasa ApplicationDbContext sada koristi generičku verziju **IdentityDbContext<ApplicationUser>**
- Identity sistem radi sa prilagođenom klasom **ApplicationUser**
- Omogućeno je čuvanje dodatnih podataka o korisniku u bazi

Dodavanje Migracije

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



- Dodaje se nova migracija komandom **Add-Migration Prva**
- Migracija opisuje izmene u modelu (ApplicationUser)
- Generiše se klasa koja sadrži promene nad bazom
- Izmene se odnose na tabelu **AspNetUsers**
- Migracija se kasnije primenjuje nad bazom pomoću **Update-Database**

Generisana baza podataka

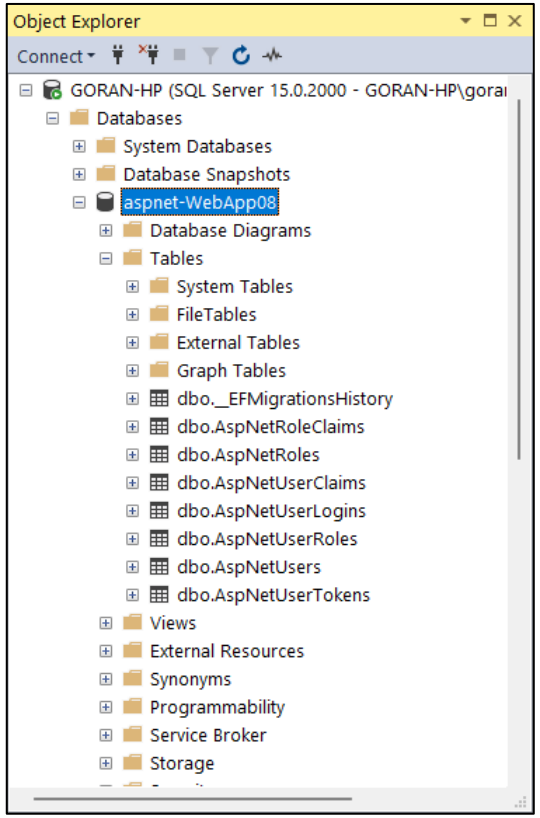
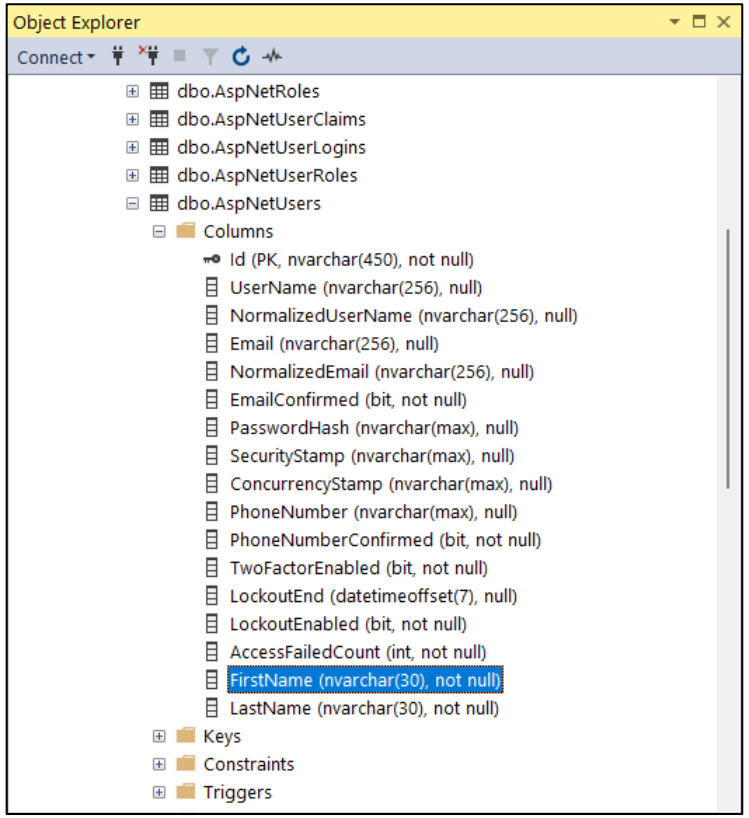


Tabela AspNetUsers



- Tabela **AspNetUsers** čuva podatke o korisnicima u Identity sistemu
- Sadrži osnovna polja kao što su korisničko ime, email i lozinka (hash)
- Automatski se kreira korišćenjem Identity sistema i migracija
- Proširenjem klase ApplicationUser dodata su nova poljaNova polja FirstName i LastName vidljiva su u tabeli

Klasa RegisterModel

```
public class RegisterModel
{
    [Required(ErrorMessage = "Korisničko ime je obavezno")]
    [StringLength(256, ErrorMessage = "Korisničko ime može imati najviše {1} karaktera")]
    public string UserName { get; set; }

    [Required(ErrorMessage = "Ime je obavezno")]
    [StringLength(30, ErrorMessage = "Ime može imati najviše {1} karaktera")]
    public string FirstName { get; set; }

    [Required(ErrorMessage = "Prezime je obavezno")]
    [StringLength(30, ErrorMessage = "Prezime može imati najviše {1} karaktera")]
    public string LastName { get; set; }

    [Required(ErrorMessage = "Lozinka je obavezna")]
    [StringLength(100, ErrorMessage = "Lozinka mora imati između {2} i {1} karaktera", MinimumLength =
3)]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Compare("Password", ErrorMessage = "Lozinka i potvrda lozinke se ne poklapaju")]
    public string ConfirmPassword { get; set; }
}
```

Klasa RegisterModel

- Klasa **RegisterModel** sadrži svojstva za unos podataka o novom korisniku
- Koristi se za prenos podataka sa forme za registraciju ka kontroleru
- Objekat ove klase se kreira na osnovu podataka koje korisnik unese u formu
- Kontroler koristi ove podatke za kreiranje novog korisničkog naloga

Atributi za autorizaciju pristupa

- Nalaze se u biblioteci Microsoft.AspNetCore.Authorization
- Koriste se za kontrolu pristupa kontrolerima i akcijama
- Atribut [**Authorize**]
  - ograničava pristup samo prijavljenim korisnicima
  - može se primeniti na kontroler ili akciju
- Atribut [**AllowAnonymous**]
  - omogućava pristup neprijavljenim korisnicima
  - koristi se kao izuzetak kada je na kontroleru postavljen [Authorize]

Kontroler za autentifikaciju

```
[Authorize]
public class AccountController : Controller
{
    private readonly UserManager<ApplicationUser> _userManager;
    private readonly SignInManager<ApplicationUser> _signInManager;
    public AccountController(UserManager<ApplicationUser> userManager,
        SignInManager<ApplicationUser> signInManager)
    {

        _userManager = userManager;
        _signInManager = signInManager;
    }
}
```

- Atribut [**Authorize**] ograničava pristup kontroleru samo na autentifikovane korisnike
- Klasa **AccountController** služi za implementaciju autentifikacije korisnika
- Kroz konstruktor se ubacuju servisi **UserManager<ApplicationUser>** i **SignInManager<ApplicationUser>**
- UserManager se koristi za rad sa korisnicima (kreiranje, pretraga, izmene)
- SignInManager se koristi za prijavu i odjavu korisnika

Kontroler za autentifikaciju

- Sadrži privatno readonly polje _signInManager tipa **SignInManager<ApplicationUser>**
- Sadrži privatno readonly polje_userManager tipa
  UserManager< ApplicationUser >
- Klase SignInManager i UserManager nalaze se u biblioteci Microsoft.AspNetCore.Identity
- Klasa **SignInManager** služi da proveri postojanje korisnika u bazi i da na osnovu toga napravi autentifikacioni kolačić(metode **SignInAsync** i **PasswordSignInAsync**)
- Klasa **SignInManager** ima metodu za uništavanje autentifikacionog kolačića (metoda SignOutAsync)
- Klasa **UserManager** služi za komunikaciju sa bazom podataka i ima metodu za kreiranje novog korisnika (metoda CreateAsync)

Register GET metoda

```
[AllowAnonymous]
public ActionResult Register()
{
    return View();
}
```

- GET metoda Register prikazuje formu za registraciju korisnika
- Atribut [**AllowAnonymous**] omogućava pristup neprijavljenim korisnicima
- Metoda vraća odgovarajući View za unos podataka
- Ne vrši obradu podataka, već samo prikaz korisničkog interfejsa

Register POST metoda

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser
        {
            UserName = model.UserName,
            FirstName = model.FirstName,
            LastName = model.LastName
        };

        var result = await _userManager.CreateAsync(user, model.Password);

        if (result.Succeeded)
        {
            await _signInManager.SignInAsync(user, isPersistent: false);
            return RedirectToAction("Index", "Home");
        }

        foreach (var error in result.Errors)
        {
            ModelState.TryAddModelError(string.Empty, error.Description);
        }
    }

    return View(model);
}
```

Register POST metoda

- POST metoda Register obrađuje podatke sa forme za registraciju
- Proverava se validnost podataka (ModelState.IsValid)
- Kreira se objekat klase **ApplicationUser** na osnovu unetih podataka
- Metodom **CreateAsync** kreira se korisnički nalog u bazi
- Ako je kreiranje uspešno, korisnik se prijavljuje u sistem
- Metodom **SignInAsync** kreira se autentifikacioni kolačić
- Nakon prijave vrši se redirekcija na početnu stranicu
- U slučaju greške, poruke se dodaju u **ModelState** i forma se ponovo prikazuje

Pogled za registraciju -1

```
@model RegisterModel
@{
    ViewData["Title"] = "Register";
}
```

```
<div class="row">
    <div class="col-md-4">
        <form asp-action="Register">
            <div asp-validation-summary="All" class="text-danger"></div>

            <div class="form-group">
                <label asp-for="UserName" class="control-label"></label>
                <input asp-for="UserName" class="form-control" />
                <span asp-validation-for="UserName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="FirstName" class="control-label"></label>
                <input asp-for="FirstName" class="form-control" />
                <span asp-validation-for="FirstName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="LastName" class="control-label"></label>
                <input asp-for="LastName" class="form-control" />
                <span asp-validation-for="LastName" class="text-danger"></span>
            </div>
```

Pogled za registraciju -2

```
<div class="form-group">
                <label asp-for="Password" class="control-label"></label>
                <input asp-for="Password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ConfirmPassword" class="control-label"></label>
                <input asp-for="ConfirmPassword" class="form-control" />
                <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Klasa LoginModel

```
public class LoginModel
{
    [Required(ErrorMessage = "Korisničko ime je obavezno")]
    public string UserName { get; set; }

    [Required(ErrorMessage = "Lozinka je obavezna")]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    [Display(Name = "Zapamti me")]
    public bool RememberMe { get; set; }
}
```

Objekat klase LoginModel će se kreirati od strane Account kontrolera na osnovu podataka koji se prosleđuju interfejsom za logovanje.

GET metoda za logovanje

```
[AllowAnonymous]
public IActionResult Login(string returnUrl = null)
{
    ViewData["ReturnUrl"] = returnUrl;
    return View();
}
```

- GET metoda Login prikazuje formu za prijavu korisnika
- Atribut [AllowAnonymous] omogućava pristup neprijavljenim korisnicima
- Parametar returnUrl čuva adresu na koju se korisnik vraća nakon prijave
- Vrednost se prosleđuje u ViewData i koristi u view-u

Metoda RedirectToLocal

```
private ActionResult RedirectToLocal(string returnUrl)
{
    if (Url.IsLocalUrl(returnUrl))
    {
        return Redirect(returnUrl);
    }
    return RedirectToAction("Index", "Home");
}
```

- Metoda RedirectToLocal vrši redirekciju korisnika nakon prijave
- Proverava da li je prosleđeni URL lokalni (Url.IsLocalUrl)
- Ako jeste, korisnik se preusmerava na tu adresu
- Ako nije, vrši se redirekcija na početnu stranicu (Home/Index)
- Sprečava se preusmeravanje na spoljne (potencijalno zlonamerne) adrese

POST metoda za logovanje

```
[AllowAnonymous]
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Login(LoginModel model, string returnUrl = null)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    var result = await _signInManager.PasswordSignInAsync(model.UserName, model.Password,
    model.RememberMe, false);
    if (result.Succeeded)
    {
        return RedirectToLocal(returnUrl);
    }
    else
    {
        ModelState.AddModelError("", "Neispravno korisničko ime ili lozinka");
        return View();
    }
}
```

POST metoda za logovanje

- POST metoda Login obrađuje podatke sa forme za prijavu
- Atribut [**AllowAnonymous**] omogućava pristup neprijavljenim korisnicima
- Proverava se validnost podataka (ModelState.IsValid)
- Metodom **PasswordSignInAsync** vrši se provera korisničkog imena i lozinke
- Ako je prijava uspešna, korisnik se preusmerava na traženu stranicu (**returnUrl**)
- U suprotnom, prikazuje se poruka o grešci i forma za prijavu

Pogled za Logovanje – Login.cshtml

```
@model LoginModel
<h1>Login</h1>
<div class="row">
    <div class="col-md-4">
        <form asp-action="Login" asp-route-returnUrl="@ViewData["ReturnUrl"]">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="UserName" class="control-label"></label>
                <input asp-for="UserName" class="form-control" />
                <span asp-validation-for="UserName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Password" class="control-label"></label>
                <input asp-for="Password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="form-group form-check">
                <label class="form-check-label">
                    <input class="form-check-input" asp-for="RememberMe" />
                    @Html.DisplayNameFor(model => model.RememberMe)
                </label>
            </div>
            <div class="form-group">
                <input type="submit" value="Log in" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Pogled za Logovanje – Login.cshtml

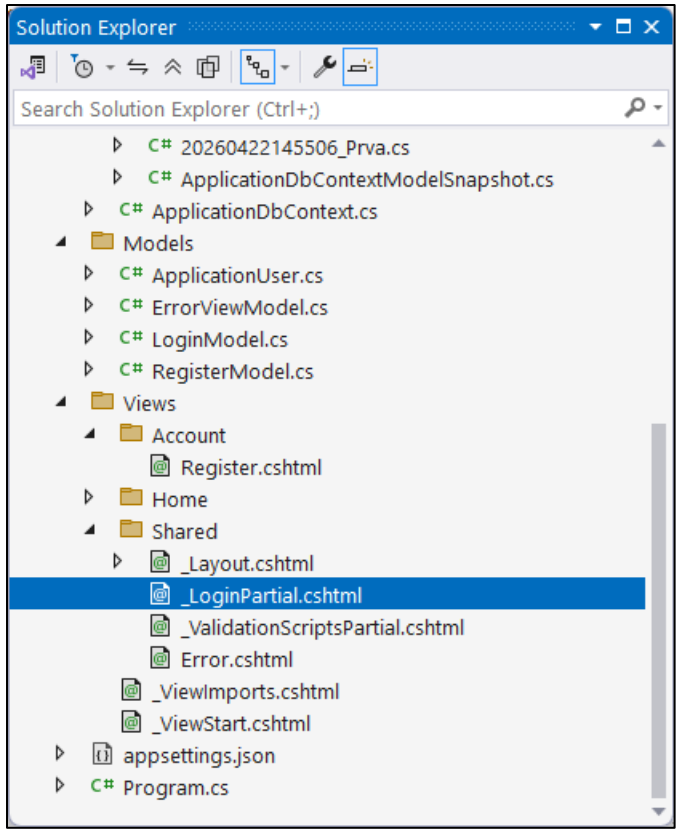
- Forma se prikazuje nakog poziva Login **GET** metode
- Uneti podaci se prosleđuju kontroleru prilikom slanja forme
- Polja za unos (UserName, Password) odgovaraju parametrima modela
- Opcija RememberMe omogućava pamćenje korisnika kroz perzistentni kolačić
- Nakon slanja forme poziva se POST metoda Login

Konfigurisanje autentifikacionog kolačića

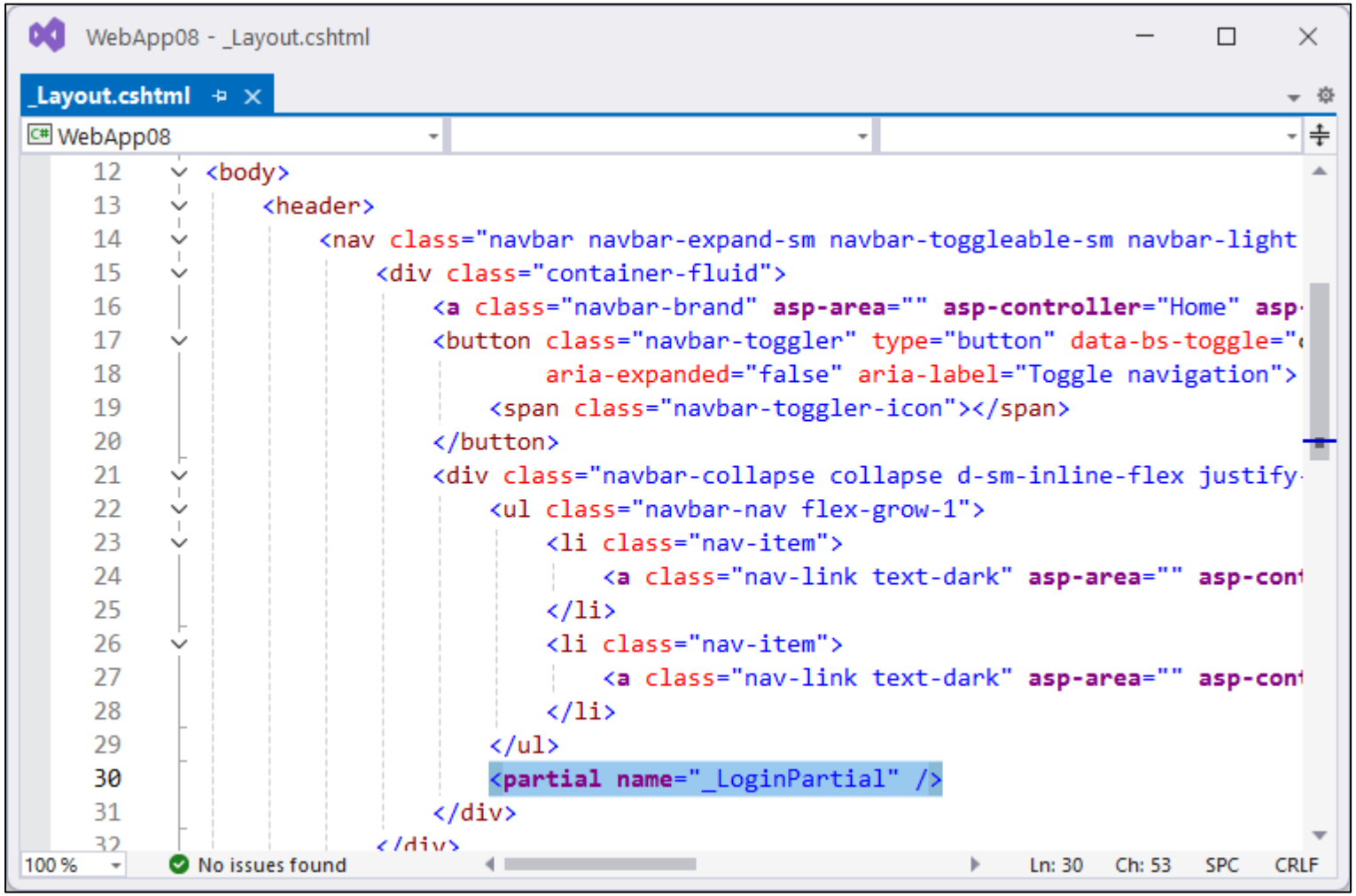
```
builder.Services.ConfigureApplicationCookie(opcije =>
{
    // Cookie settings
    opcije.Cookie.HttpOnly = true;
    opcije.ExpireTimeSpan = TimeSpan.FromMinutes(5);

    opcije.LoginPath = "/Account/Login";
    opcije.AccessDeniedPath = "/Account/AccessDenied";
    opcije.SlidingExpiration = true;
});
```

- Konfiguriše se autentifikacioni kolačić u fajlu Program.cs
- Svojstvo HttpOnly sprečava pristup kolačiću iz JavaScript-a
- ExpireTimeSpan definiše vreme važenja kolačića
- **LoginPath** određuje rutu za prijavu korisnika
- **AccessDeniedPath** određuje rutu u slučaju zabrane pristupa
- SlidingExpiration produžava trajanje kolačića pri aktivnom korišćenju aplikacije

Parcijalni pogled za logovanje



Ubacivanje parcijalnog pogleda za logovanje u Layout



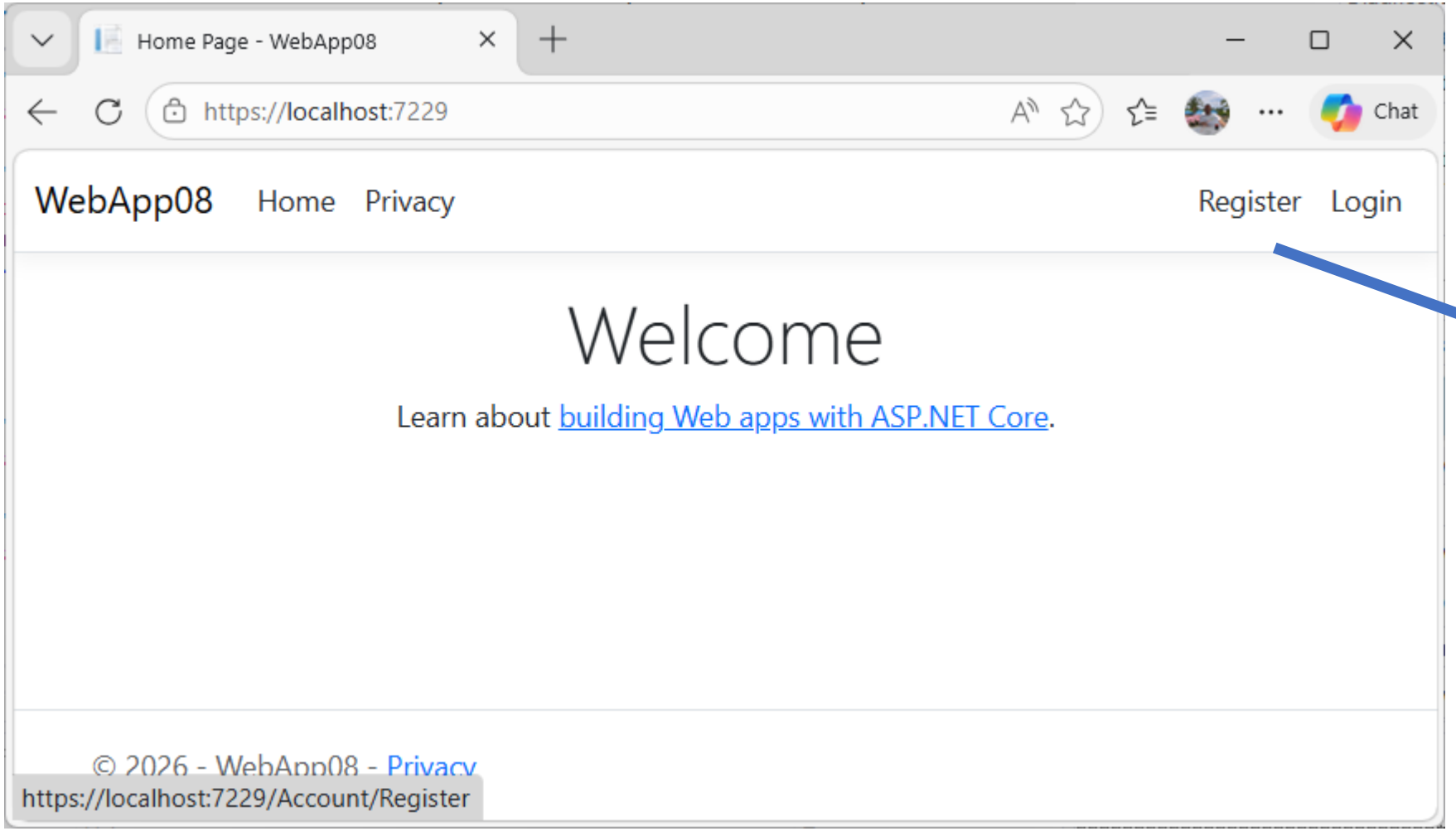
Parcijalni pogled za logovanje _LoginPartial.cshtml

```
@using Microsoft.AspNetCore.Identity
@inject SignInManager<ApplicationUser> SignInManager
@inject UserManager<ApplicationUser> UserManager

<ul class="navbar-nav">
    @if (SignInManager.IsSignedIn(User))
    {
        <li class="nav-item">
            <a class="nav-link text-dark">Hello @User.Identity?.Name!</a>
        </li>
        <li class="nav-item">
            <form class="form-inline" asp-controller="Account" asp-action="Logout" asp-route-
returnUrl="@Url.Action("Index", "Home", new { area = "" })">
                <button type="submit" class="nav-link btn btn-link text-dark">Logout</button>
            </form>
        </li>
    }
    else
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Account" asp-action="Register">Register</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Account" asp-action="Login">Login</a>
        </li>
    }
</ul>
```

Parcijalni pogled za logovanje _LoginPartial.cshtml

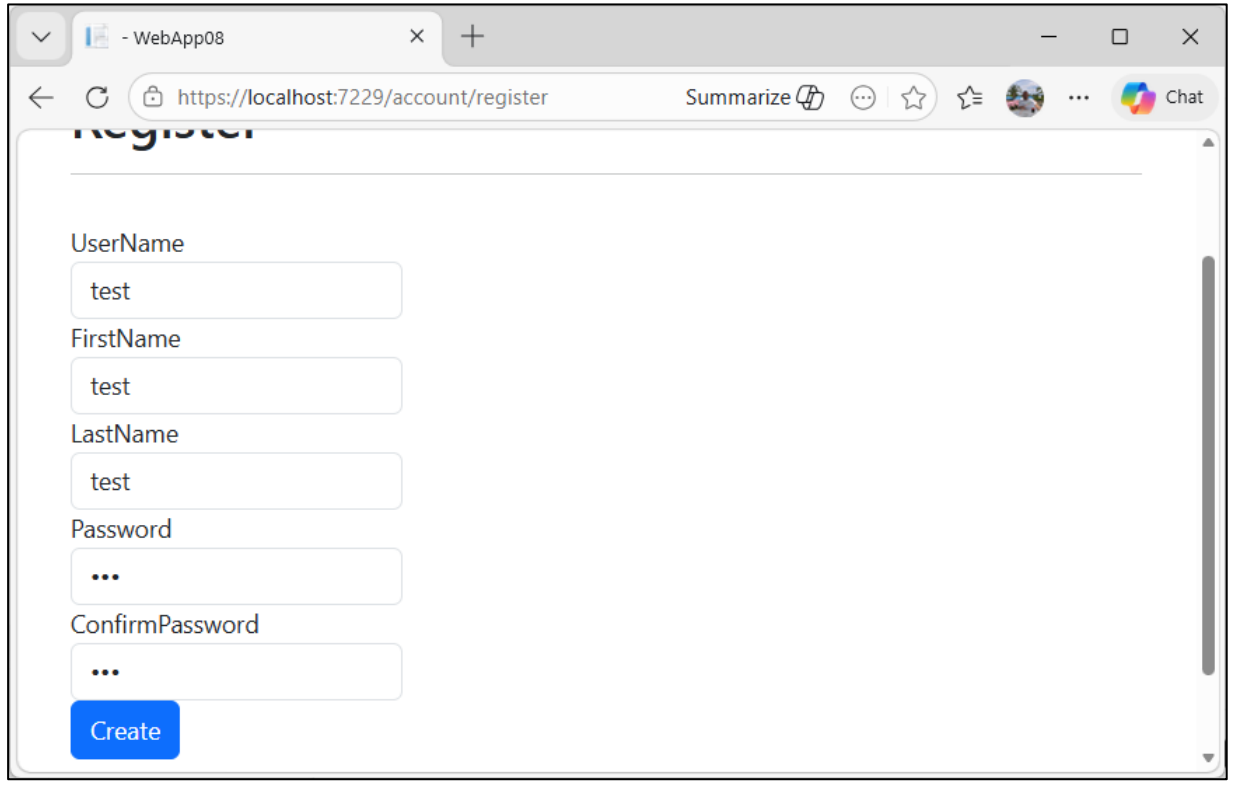
- Parcijalni pogled _LoginPartial.cshtml je izmenjen za rad u MVC aplikaciji
- Uklonjena je zavisnost od Razor Pages pristupa
- Linkovi koriste MVC rutiranje (asp-controller, asp-action)
- Umesto klase IdentityUser koristi se proširena klasa ApplicationUser
- Servisi su tipizirani kao SignInManager<ApplicationUser> i UserManager<ApplicationUser>
- Prikaz se menja u zavisnosti od toga da li je korisnik prijavljen
- Ako je prijavljen, prikazuje se korisničko ime i opcija za odjavu
- Ako nije prijavljen, prikazuju se linkovi za registraciju i prijavu

Parcijalni pogled za logovanje

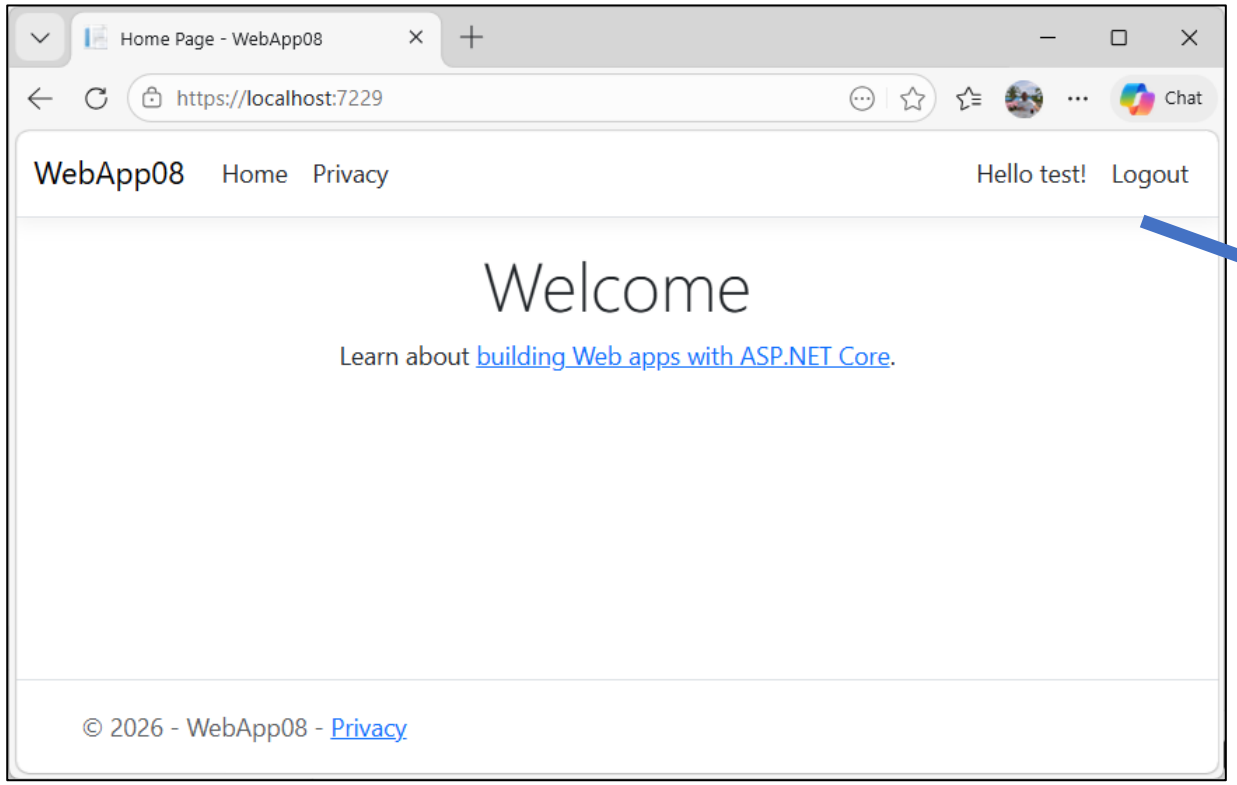


parcijalni pogled

Registracija novog korisnika

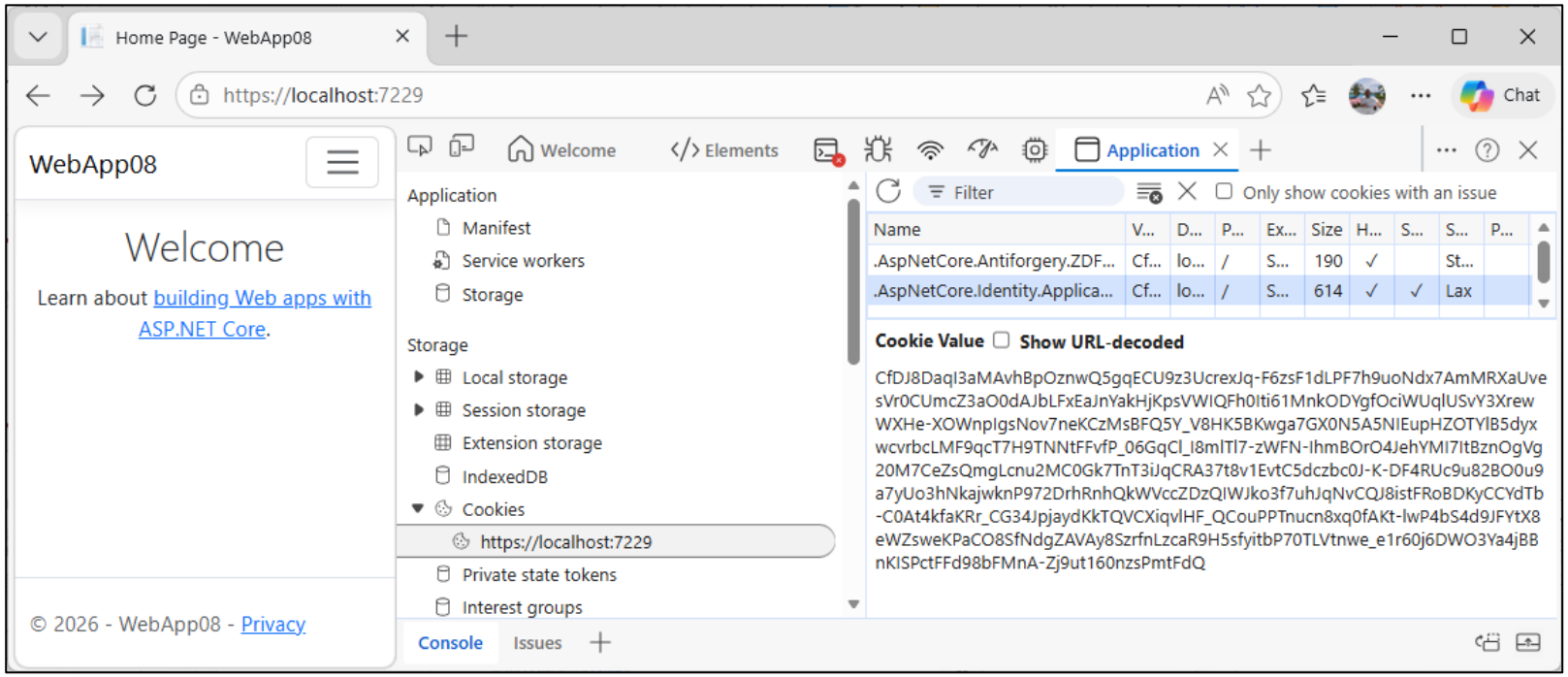


Parcijalni pogled za logovanje



parcijalni pogled

Autentifikacioni kolačić

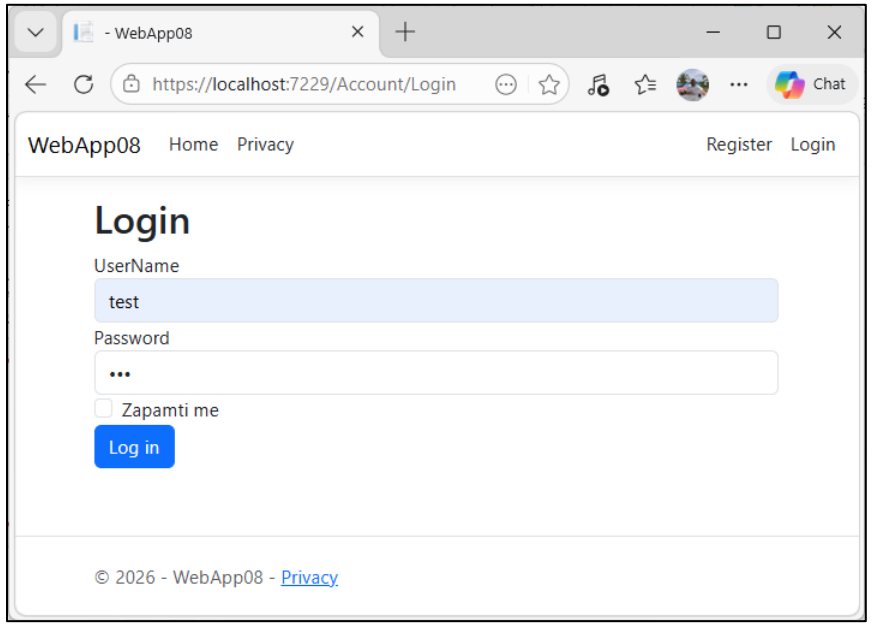


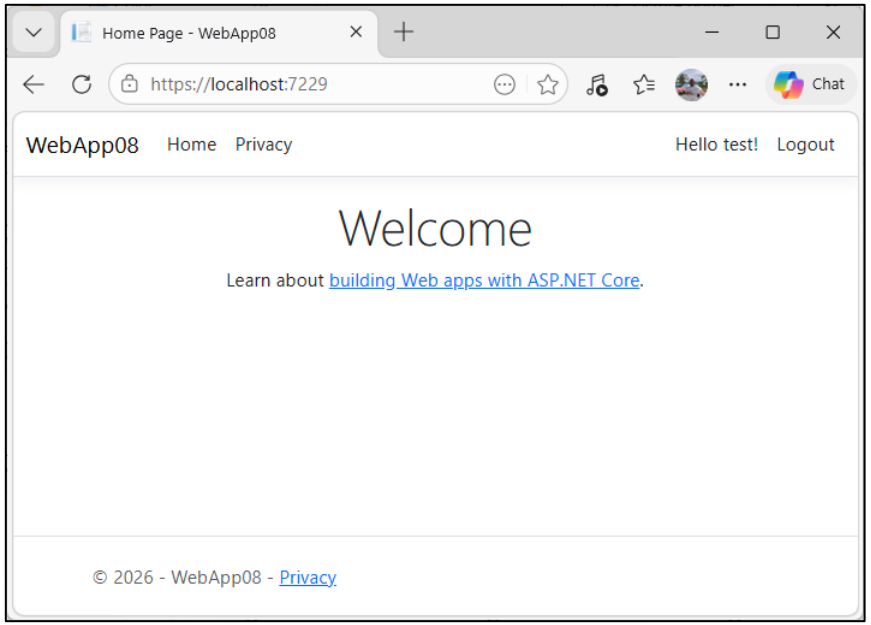
Uništavanje autentifikacionog kolačića

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Logout()
{
    await _signInManager.SignOutAsync();

    return RedirectToAction("Index", "Home");
}
```

- Metodom SignOutAsync objekta klase SignInManager uklanja se autentifikacioni kolačić
- Korisnik se odjavljuje iz sistema
- Nakon odjave vrši se redirekcija na početnu stranicu (Home/Index)

Logovanje na aplikaciju





Model klasa Osoba

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

    [Required(ErrorMessage = "Ime je obavezno")]
    [StringLength(30, ErrorMessage = "Ime može imati najviše 30 karaktera")]
    public string Ime { get; set; }

    [Required(ErrorMessage = "Prezime je obavezno")]
    [StringLength(30, ErrorMessage = "Prezime može imati najviše 30 karaktera")]
    public string Prezime { get; set; }

    [Required(ErrorMessage = "Telefon je obavezan")]
    [StringLength(20, ErrorMessage = "Telefon može imati najviše 20 karaktera")]
    public string Telefon { get; set; }
}
```

DbContext klasa

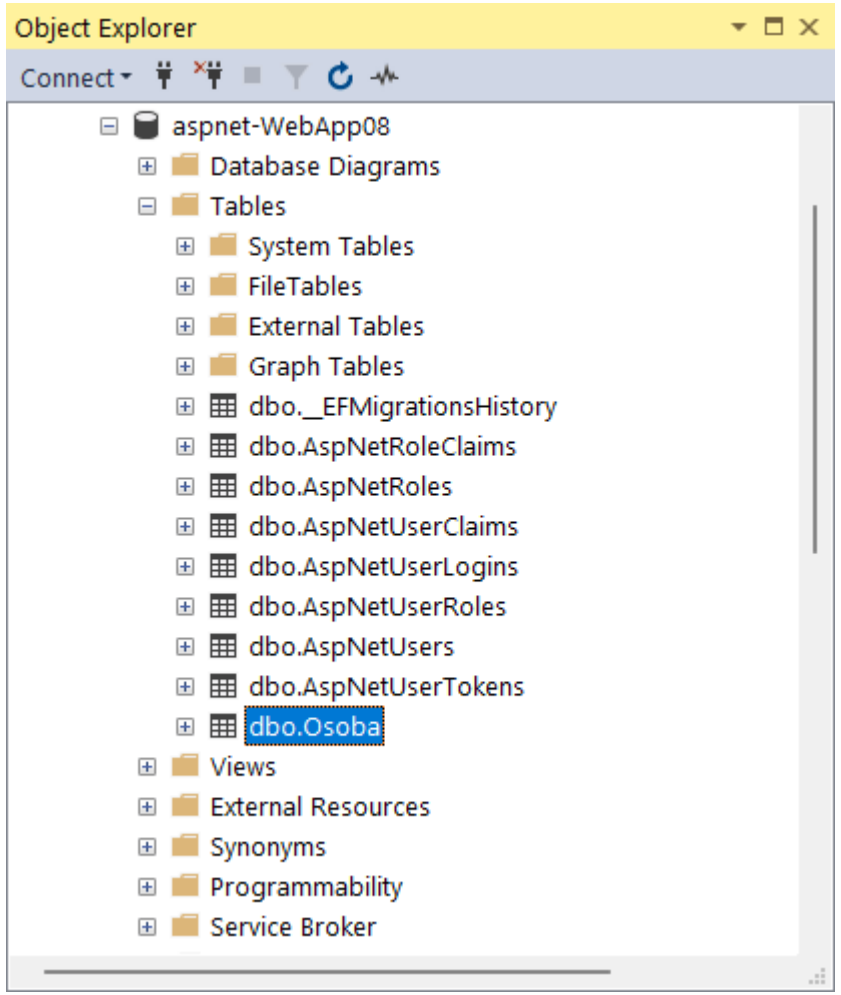
```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
    }

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

- Klasa ApplicationDbContext je proširena dodavanjem entiteta Osoba
- Definisano je svojstvo **DbSet<Osoba> Osobe**
- Svojstvo Osobe predstavlja skup entiteta tipa Osoba
- Omogućava pristup podacima iz tabele Osoba u bazi
- Preko DbSet<Osoba> vrše se operacije nad podacima (dodavanje, čitanje, izmena, brisanje)

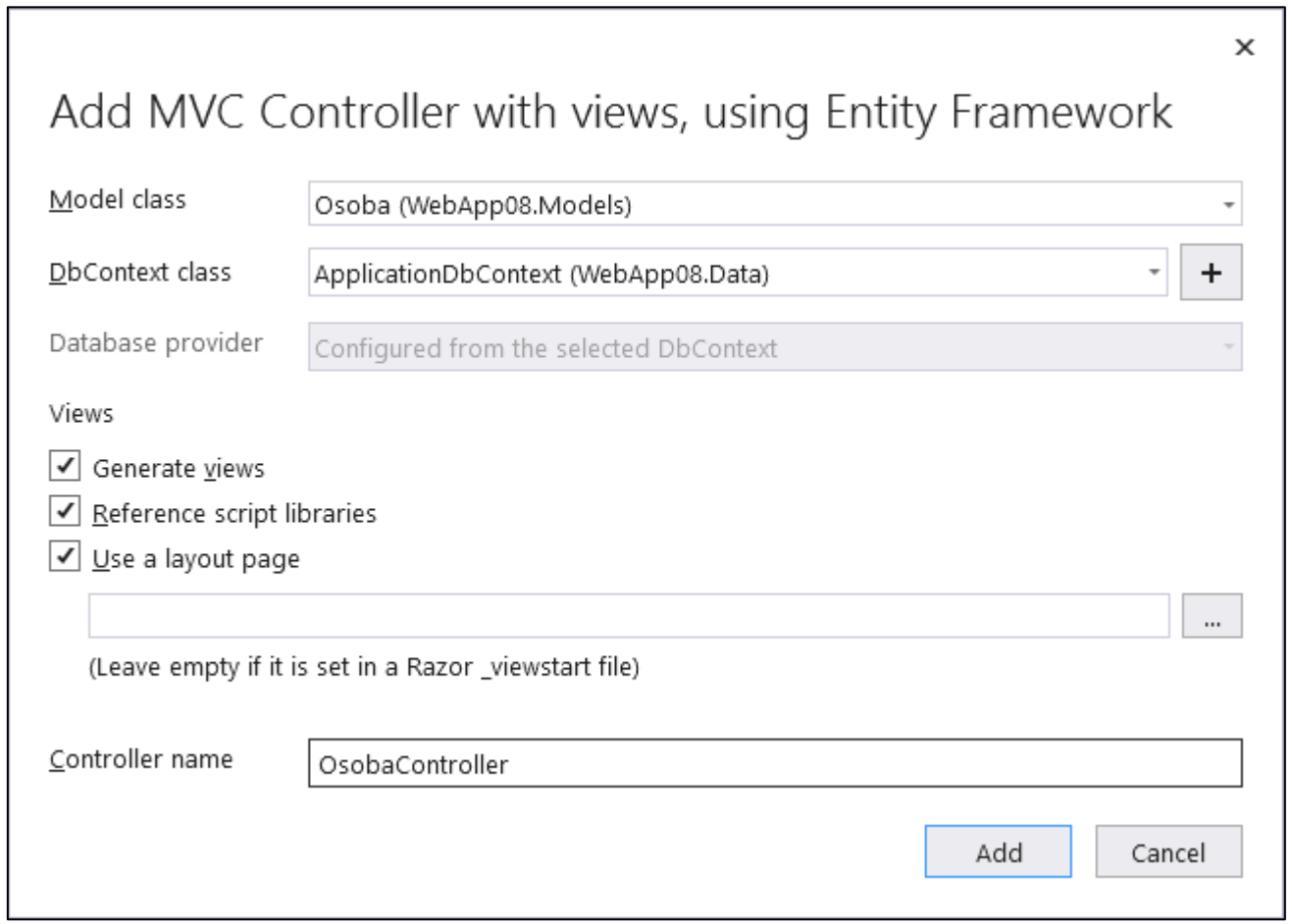
Migracije baze

```
PM> Add-Migration Druga
```

```
PM> Update-Database
```



Kreiranje kontrolera



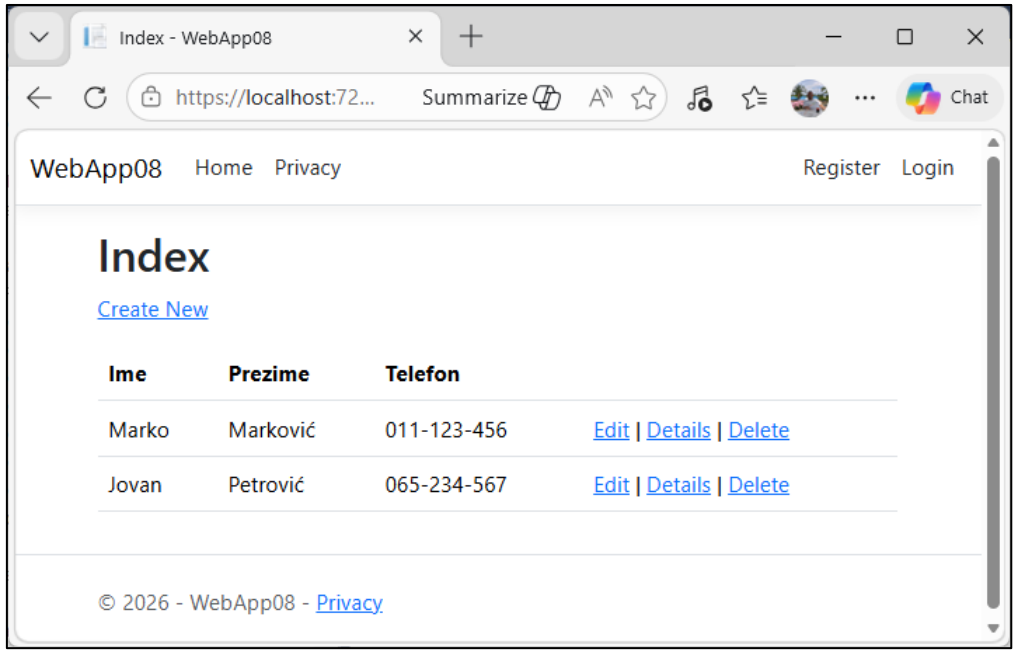
Autorizacija kontrolera

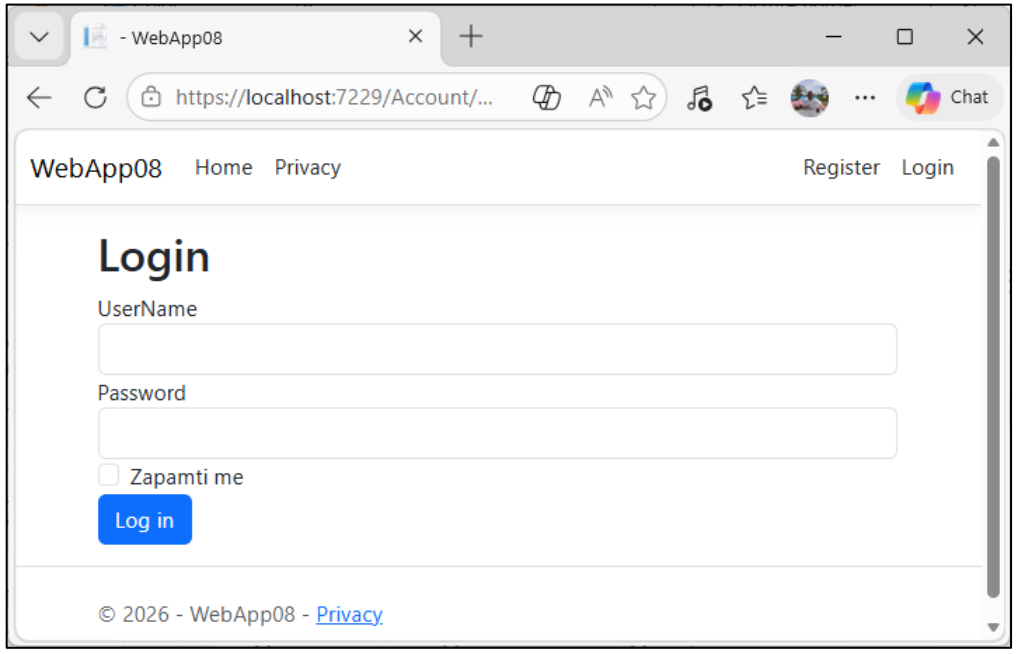
```
12  namespace WebApp08.Controllers
13  {
14      [Authorize]
        1 reference
15      public class OsobaController : Controller
16      {
17          private readonly ApplicationDbContext _context;
18
            0 references
19          public OsobaController(ApplicationDbContext context)
20          {
21              _context = context;
22          }
23
```

Omogućavanje pristupa Index akciji anonimnim korisnicima

```
[AllowAnonymous]
// GET: Osoba
public async Task<IActionResult> Index()
{
    return View(await _context.Osobe.ToListAsync());
}
```

Pokušaj editovanja podataka od strane anonimnog korisnika





Pitanje 1

Atributska klasa za opis akcione metode kontrolera pomoću koje se u autorizovanim kontrolerima dozvoljava pristup toj akcionoj metodi anonimnim korisnicima je:

a. [UnAuthorize]
b. [AllowAnonymous]
c. [NoAuthorize]

Odgovor: b

Pitanje 2

Na koji način se zabranjuje neregistrovanim korisnicima ASP.NET Core MVC web aplikacije, da pozivaju akcione metode nekog kontrolera?

a. Kontroler se opsije atributskom klasom [NoAnonymous]
b. Kontroler se opisuje atributskom klasom [Authenticate]
c. Kontroler se opisuje atributskom klasom [Authorize]

Odgovor: c

Pitanje 3

Koju baznu klasu ASP.NET Core Identity sistem koristi za predstavljanje korisnika?

a. IdentityUser
b. User
c. LoggedUser

Odgovor: a

Pitanje 4

Registracija Identity servisa u ASP.NET Core aplikaciji vrši se u okviru fajla Program.cs pozivom metode:

a. builder.services.AddDefaultIdentity<IdentityUser>()
b. builderservices.AddAutentitication<IdentityUser>()
c. builderservices.Add<IdentityUser>()

Odgovor: a

Pitanje 5

Zaokruži opciju u kojoj je ispravno postavljen redosled middleware komponenti za autentifikaciju i autorizaciju u ASP.NET Core MVC aplikaciji:

a.
```
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
```

b.
```
app.UseAuthentication();
app.UseRouting();
app.UseAuthorization();
```

c.
```
app.UseRouting();
app.UseAuthorization();
app.UseAuthentication();
```

Odgovor: a

Pitanje 6

Kastomizovani korisnik Identity sistema predstavljen je klasom ApplicationUser. Koja generička klasa se koristi za kreiranje novog korisnika u ASP.NET Core MVC aplikaciji?

a. UserManager<ApplicationUser>
b. SignInManager<ApplicationUser>
c. User <ApplicationUser>

Odgovor: a

Pitanje 7

Konfigurisanje autentifikacionog kolačića u ASP.NET Core aplikaciji vrši se pozivom koje metode?

a. builder.Services. CookieOptions()
b. builder.Services.Cookie()
c. builder.Services.ConfigureApplicationCookie()

Odgovor: c

Pitanje 8

Koja metoda klase UserManager se koristi za kreiranje novog korisnika u ASP.NET Core Identity sistemu?

a. NewUserAsync()
b. CreateAsync()
c. CreateUser()

Odgovor: b

Pitanje 9

Kreiranje autentifikacionog kolačića nakon uspešne prijave korisnika kod ASP.NET Core Identity sistema vrši se korišćenjem koje metode?

a. Metode SignInAsync() objekta klase SignInManager
b. Metode SignInAsync() objekta klase UserManager
c. Metode CreateCookie() objekta klase SignInManager

Odgovor: a

Pitanje 10

Uništavanje autentifikacionog kolačića kod ASP.NET Core Identity sistema vrši se korišćenjem:

a. Metode SignOutAsync objekta klase SignInManager
b. Metode SignOutAsync objekta klase UserManager
c. Metode SignOut objekta klase Cookie

Odgovor: a