Administracija pomoću rola

Role

- U cilju lakše administracije korisnici se grupišu u role
- Umesto autorizacije pojedinačnih korisnika radi se autorizacija role
- Biblioteka Microsoft.AspNetCore.Identity sadrži klasu **IdentityRole** kojom se predstavlja rola
- Svaka rola ima svoj Id i svoje ime koje se predstavlja posredstvom svojstva **Name**
- Potrebno je registrovati servis za rad sa rolama

Klasa RoleManager

- Klasa RoleManager omogući će rad sa rolama, kreiranje, editovanje i brisanje role
- Metoda **RoleExistsAsync** proverava da li je rola sačuvana u tabeli AspNetRoles i vraća true ako je tačno
- Svojstvo **Roles** objekta klase RoleManager vraća listu IdentityRole objekata koji odgovaraju rolama sačuvanim u bazi podataka
- Metoda **CreateAsync** služi za kreiranje nove role (upis reda u tabelu AspMetRoles) i vraća rezultat koji je referenca tipa IdentityResult
- Metoda **FindByIdAsync** proverava da li u bazi postoji rola sa datim Id-om i ako postoji vraća objekat klase IdentityRole na osnovu podataka iz baze
- Metoda **FindByNameAsync** proverava da li u bazi postoji rola sa datim imenom(svojstvo Name) i ako postoji vraća objekat klase IdentityRole na osnovu podataka iz baze
- Metoda **UpdateAsync** kao ulazni parametar očekuje objekat klase IdentityRole na osnovu koga će biti promenjena rola u bazi sa istim Id atributom kao i prosleđeni objekat
- Metoda **DeleteAsync** kao ulazni parametar očekuje objekat klase IdentityRole na osnovu koga će biti obrisana rola u bazi sa istim Id atributom kao i prosleđeni objekat

Klasa UserManager

- Klasa UserManager osim što služi za kreiranje korisnika aplikacije ima i metode za rad sa rolama
- Klasa **UserManager** ima metodu **IsInRoleAsync** kojom se proverava da li se neki korisnik nalazi u zahtevanoj roli
- Metoda **GetRolesAsync** kao ulazni parametar zahteva objekat klase ApplicationUser i vraća listu rola za datog korisnika

Polazne osnove

- Pretpostavlja se da je realizovana kastomizacija sistema za logovanje kao u prethodnom predavanju
- Korisnik se predstavlja modelom klase **ApplicationUser**
- Kreirani su modeli za logovanje i registraciju: **LoginModel** i **RegisterModel**
- Kreirana je baza koja će sadržati tabele Identity sistema (pomoću code-first migracija)
- Kreirana je klasa **AccountController** sa metodama:
  - **Login** (GET i POST)
  - **Register** (GET i POST)
  - **SignOut** za uništavanje autentifikacionog kolačića

Konfiguracija Identity sistema sa ulogama

```
// Podesavanje opcija za lozinke
var identityOptions = new Action<IdentityOptions>(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;
});

// Registracija Identity sistema sa korisničkim modelom i rolama
builder.Services.AddIdentity<ApplicationUser, IdentityRole>(identityOptions)
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultTokenProviders();

// Dodavanje kontrolera i pogleda
builder.Services.AddControllersWithViews();

// Podesavanje opcija za autentifikaciju kolačića
var cookieOptions = new Action<CookieAuthenticationOptions>(options =>
{
    // Podesavanje kolačića
    options.Cookie.HttpOnly = true;
    options.ExpireTimeSpan = TimeSpan.FromMinutes(5);
    options.LoginPath = "/Account/Login";
    options.AccessDeniedPath = "/Account/AccessDenied";
    options.SlidingExpiration = true;
});

// Konfiguracija kolačića u aplikaciji
builder.Services.ConfigureApplicationCookie(cookieOptions);
```

Konfiguracija Identity sistema sa ulogama

- Promenljiva **identityOptions** omogućava podešavanje sigurnosnih pravila za lozinke i korisničke podatke
- Metoda **AddIdentity** registruje servis za rad sa korisnicima i ulogama, koristeći ApplicationUser kao korisnički model i IdentityRole za uloge, što omogućava kreiranje korisnika i dodeljivanje uloga
- Promenljiva **cookieOptions** omogućava podešavanje autentifikacionih kolačića
- Metoda **ConfigureApplicationCookie** konfiguriše autentifikacione kolačiće

Kontroler za inicijalizaciju baze - 1

```
public class DbInitController : Controller
{
    private readonly UserManager<ApplicationUser> _userManager;
    private readonly RoleManager<IdentityRole> _roleManager;

    public DbInitController(UserManager<ApplicationUser> userManager,
    RoleManager<IdentityRole> roleManager)
    {
        _userManager = userManager;
        _roleManager = roleManager;
    }
...
}
```

Kontroler za inicijalizaciju baze - 2

```
// Kreiranje role admin ako ne postoji
private async Task CreateRoleIfNotExists(string roleName)
{
    var roleExist = await _roleManager.RoleExistsAsync(roleName);
    if (!roleExist)
    {
        IdentityRole role = new IdentityRole(roleName);
        await _roleManager.CreateAsync(role);
    }
}
```

Kontroler za inicijalizaciju baze - 3

```
// Kreiranje admin korisnika ako ne postoji
private async Task CreateAdminUserIfNotExists()
{
    var user = await _userManager.FindByNameAsync("admin");

    if (user == null)
    {
        ApplicationUser adminUser = new ApplicationUser
        {
            UserName = "admin",
            FirstName = "Marko",
            LastName = "Markovic"
        };
        var password = "123";  // Jednostavna lozinka za testiranje

        var result = await _userManager.CreateAsync(adminUser, password);
        if (!result.Succeeded)
        {
            // Možete obraditi grešku ovde, ako kreiranje korisnika ne uspe
        }
    }
}
```

Kontroler za inicijalizaciju baze - 4

```
// Dodeljivanje korisnika roli
private async Task AssignUserToRole(string userName, string roleName)
{
    var user = await _userManager.FindByNameAsync(userName);
    if (user != null)
    {
        var result = await _userManager.AddToRoleAsync(user, roleName);
        if (!result.Succeeded)
        {
            // Obrada greške pri dodeljivanju uloge
        }
    }
}
```

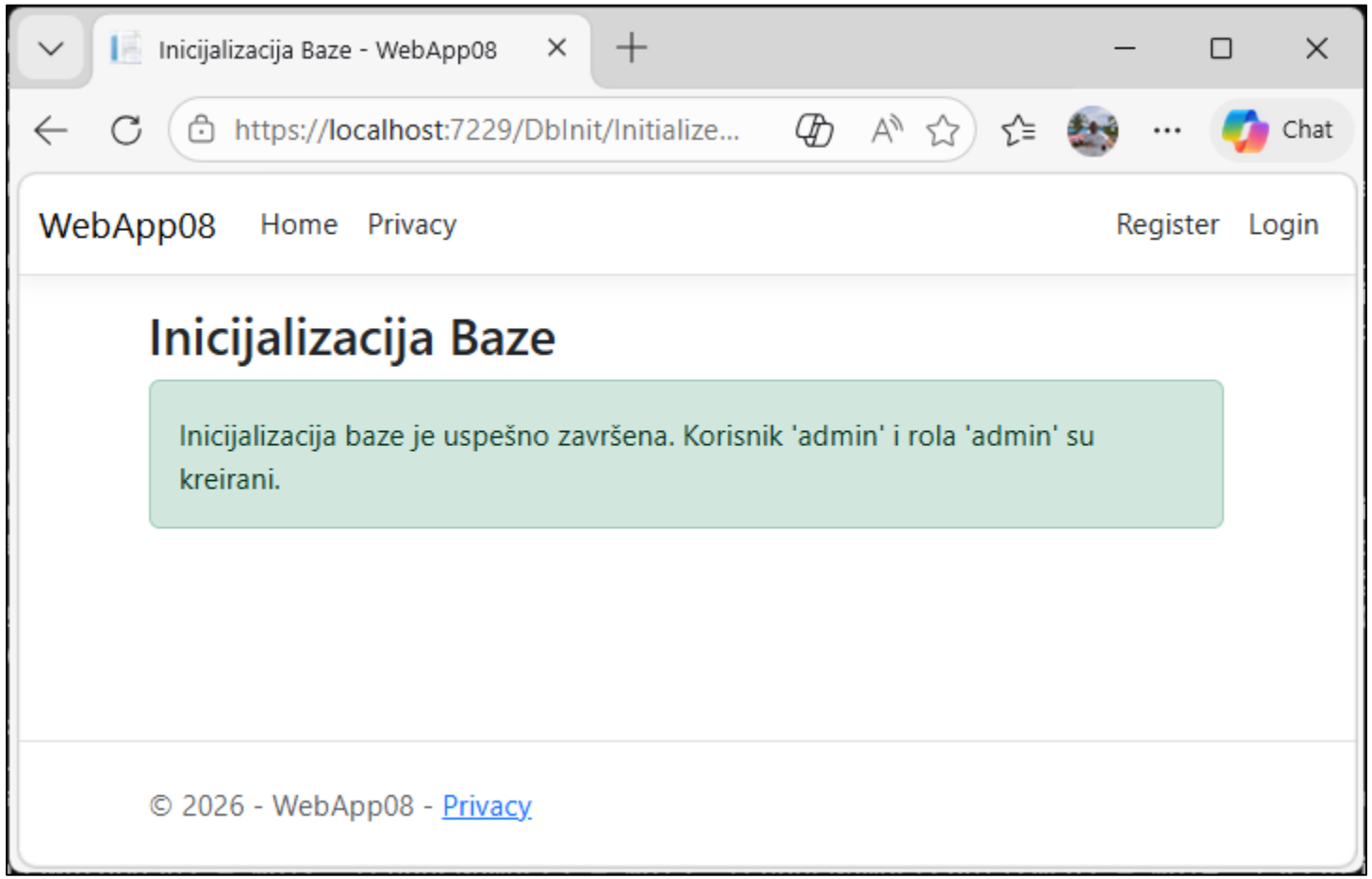
Kontroler za inicijalizaciju baze - 5

```
// Ova metoda inicijalizuje bazu sa korisnicima i rolama
public async Task<IActionResult> InitializeDatabase()
{
    // Kreiraj rolu admin ako ne postoji
    await CreateRoleIfNotExists("admin");

    // Kreiraj korisnika admin ako ne postoji
    await CreateAdminUserIfNotExists();

    // Dodeli admin korisniku rolu admin
    await AssignUserToRole("admin", "admin");

    // Postavljanje poruke koja potvrđuje da je inicijalizacija završena
    ViewBag.Message = "Inicijalizacija baze je uspešno završena.
    Korisnik 'admin' i rola 'admin' su kreirani.";

    return View();
}
```

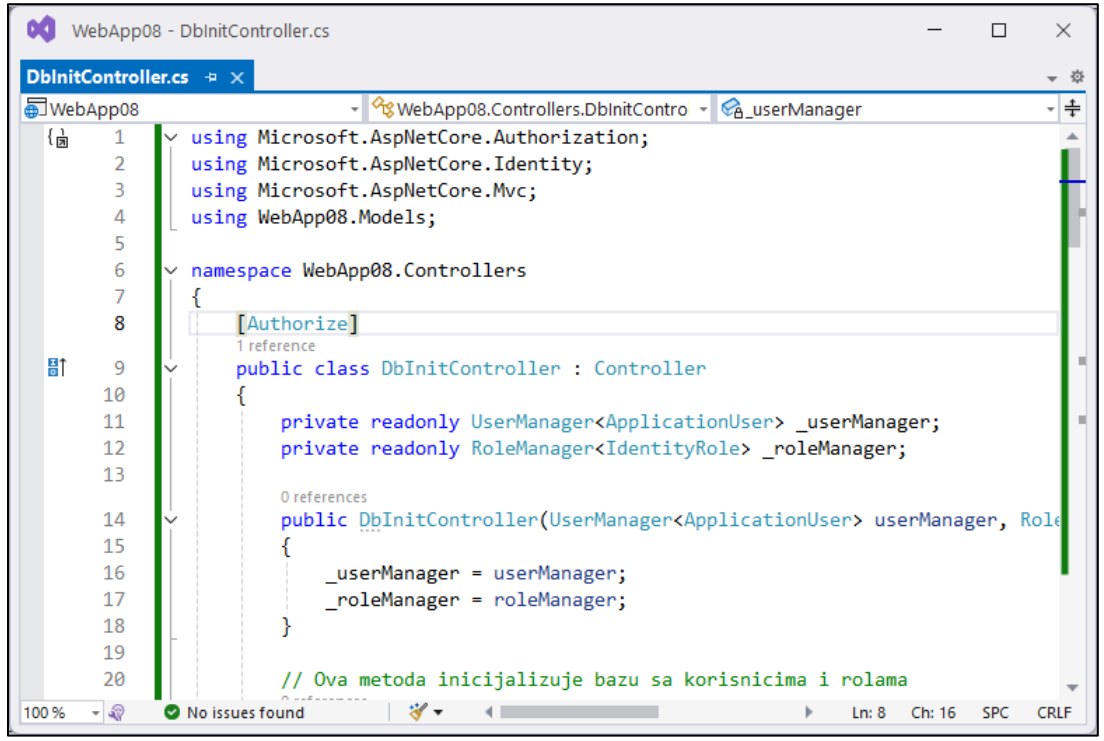
Pogled InitializeDatabase.cshtml

```
@{
    ViewData["Title"] = "Inicijalizacija Baze";
}

<h2>@ViewData["Title"]</h2>

@if (ViewBag.Message != null)
{
    <div class="alert alert-success">
        @ViewBag.Message
    </div>
}
else
{
    <div class="alert alert-warning">
        Inicijalizacija baze nije izvršena.
    </div>
}
```

Uspešna inicijalizacija baze



Autorazacija kontrolera DbInitController



Dijagram veze tabele korisnika i tabele sa ulogama

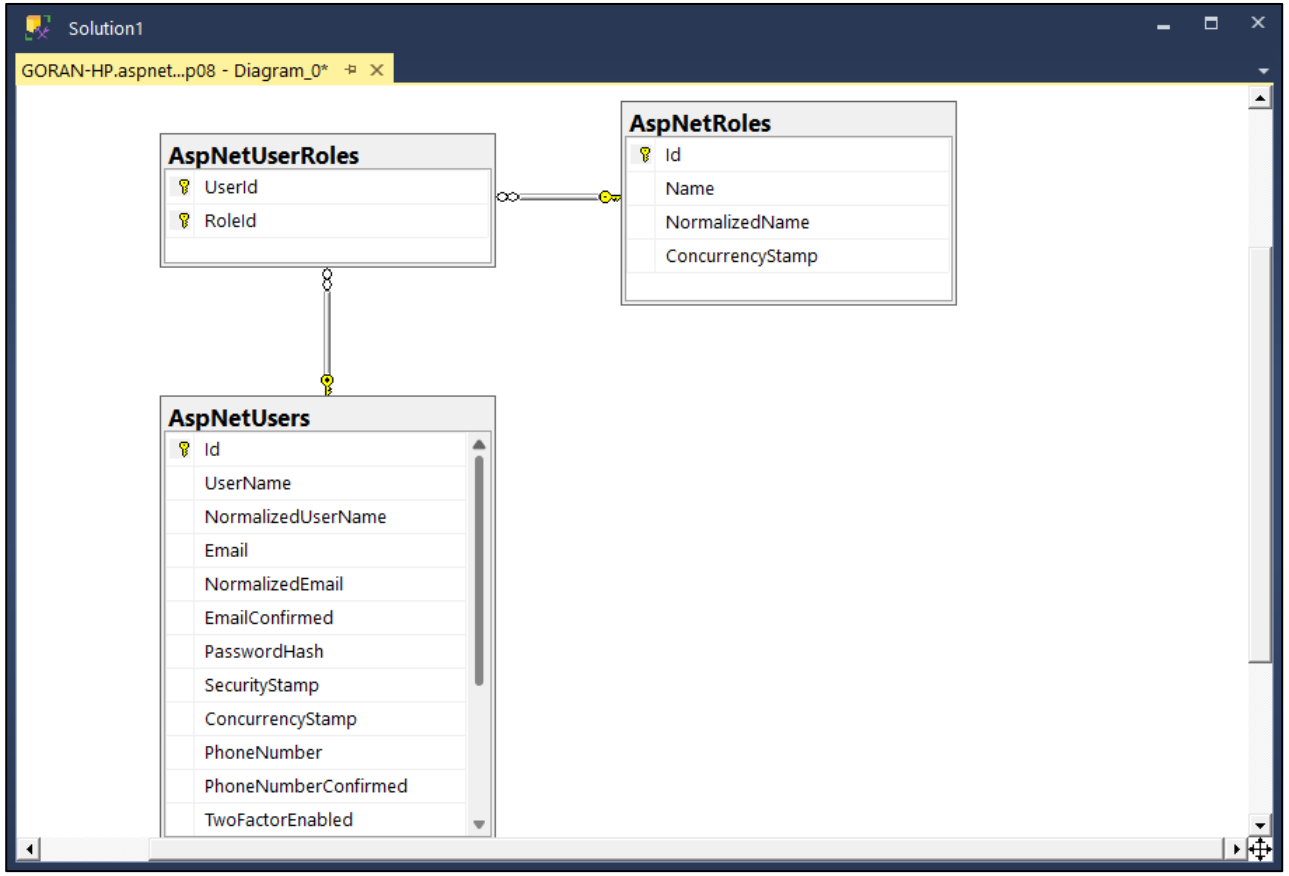


Tabela AspNet.Roles

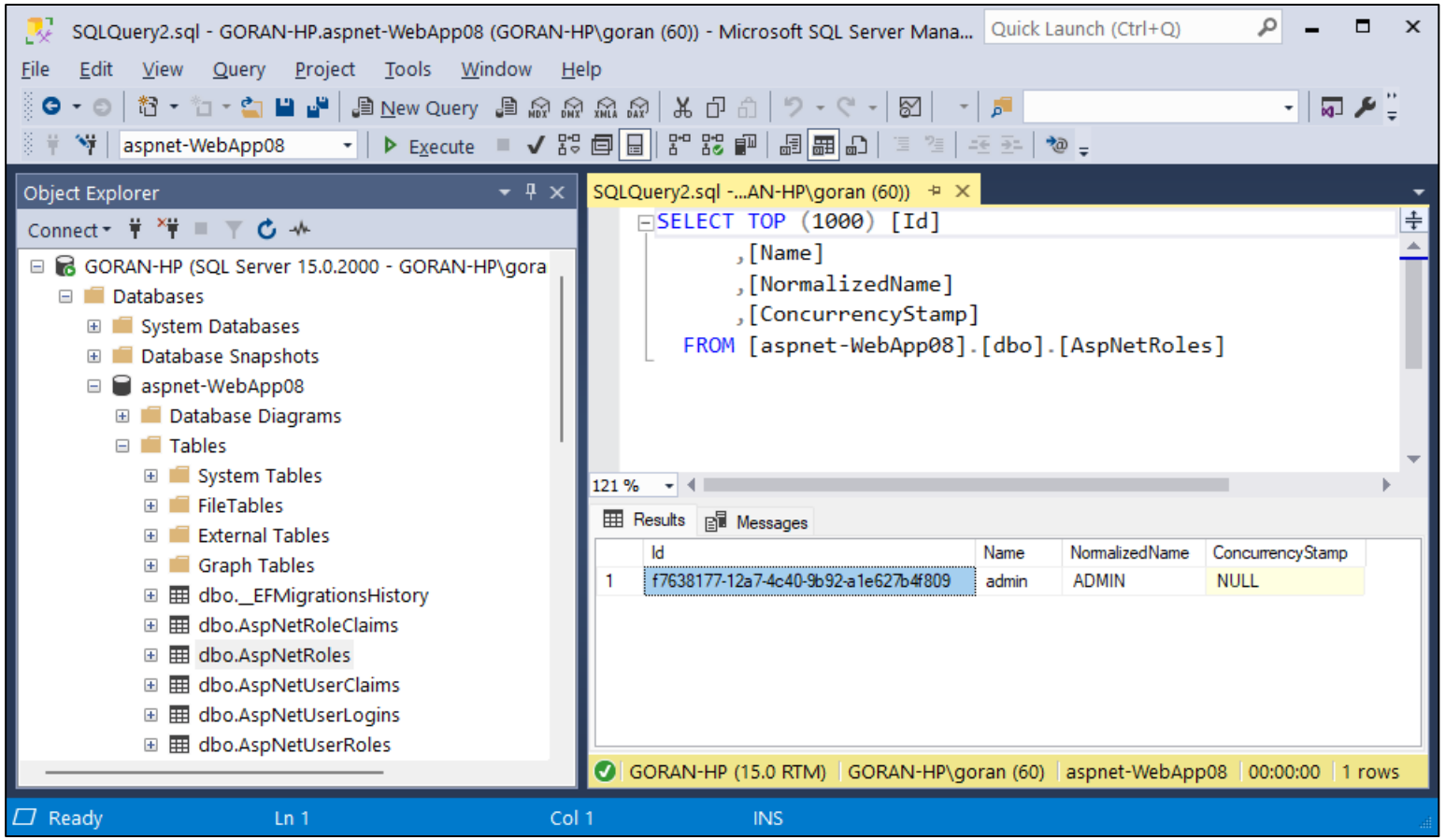


Tabela ASpNetUsers

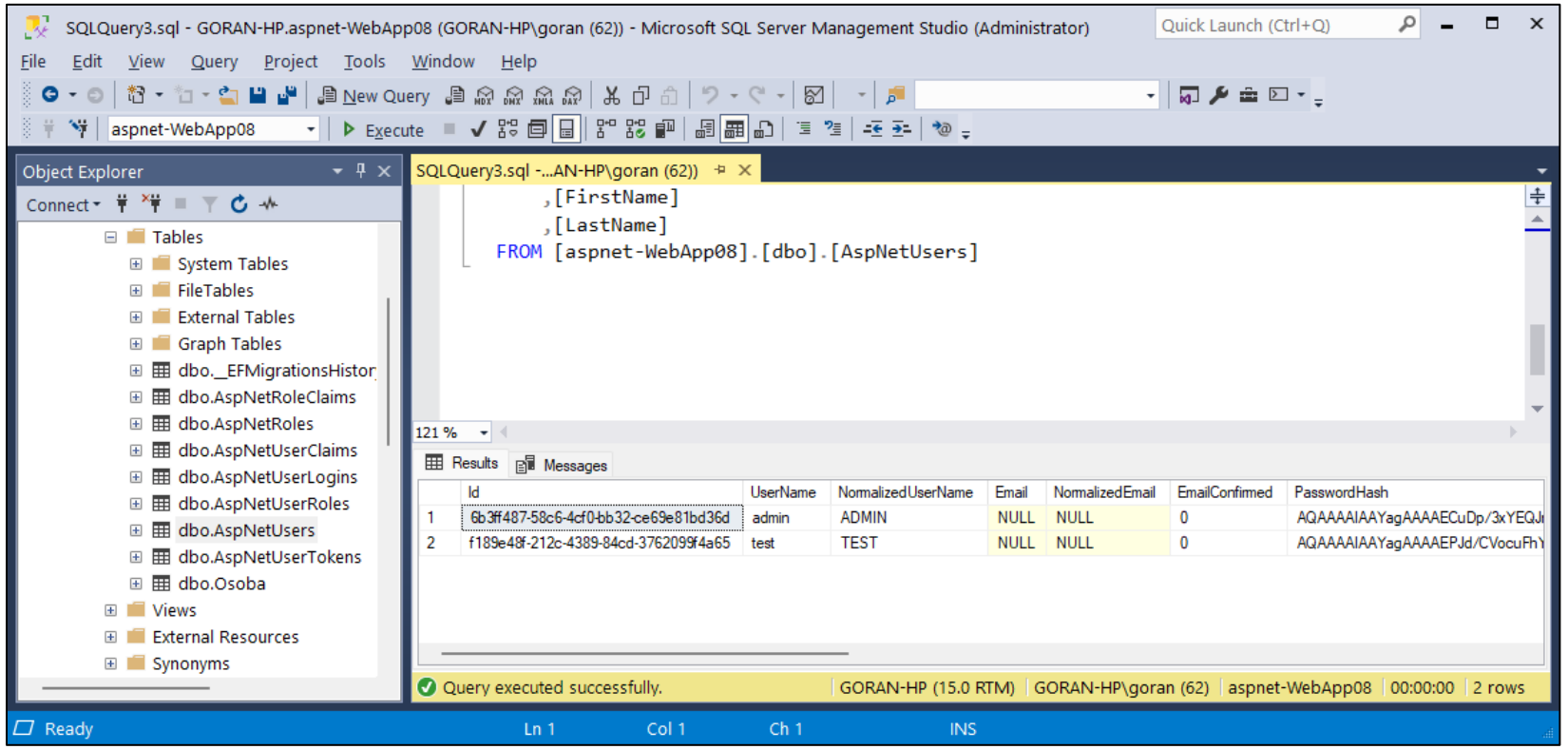
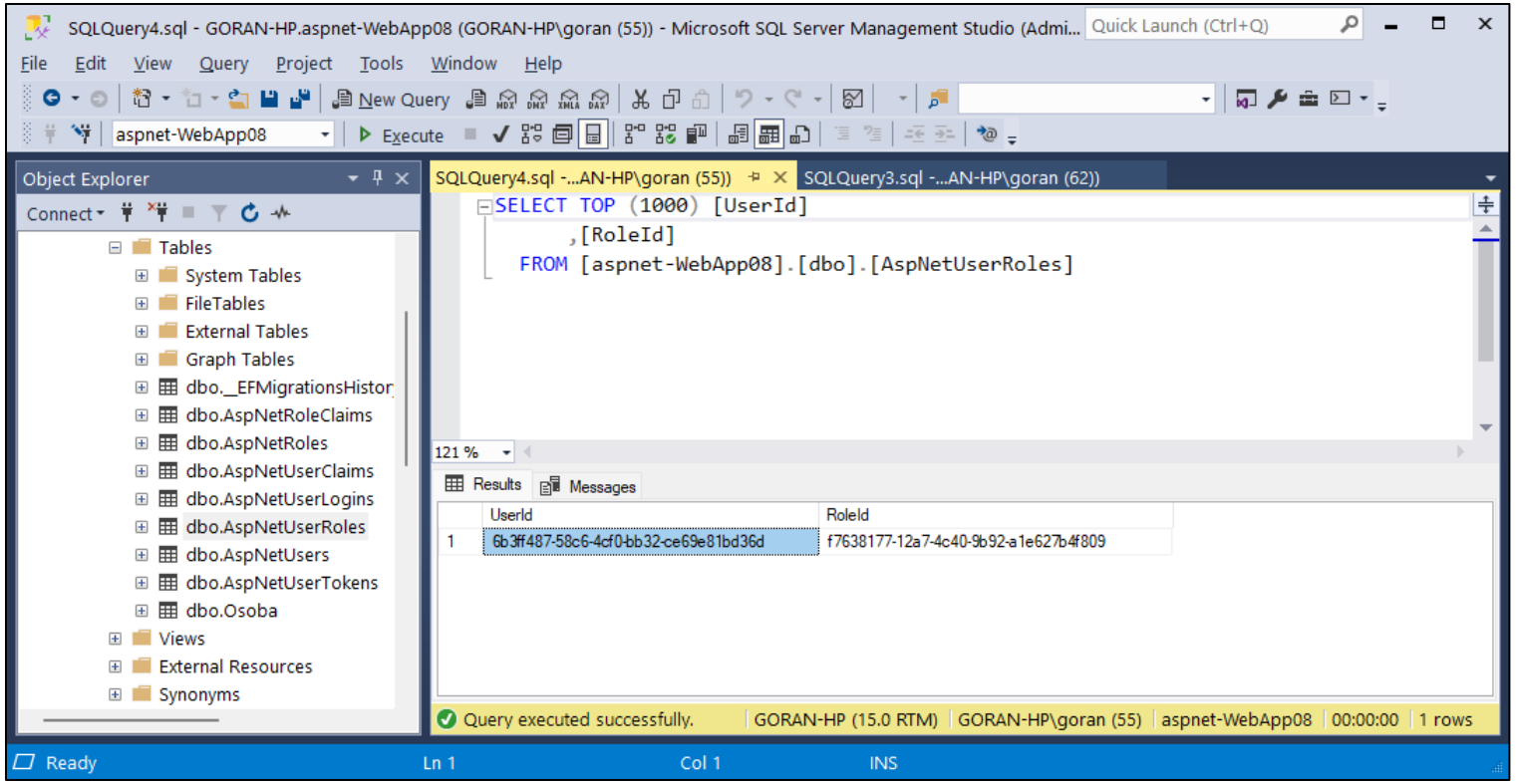


Tabela AspNetUserRoles



Kreiranje kontrolera za rad sa rolama

```
// Kontroler sa autorizacijom samo za admin korisnike
[Authorize(Roles = "admin")]
public class RoleController : Controller
{
    private readonly ApplicationDbContext db;
    private readonly RoleManager<IdentityRole> rm;
    private readonly UserManager<ApplicationUser> um;

    // Konstruktor za injektovanje zavisnosti
    public RoleController(ApplicationDbContext _db, RoleManager<IdentityRole> _rm,
    UserManager<ApplicationUser> _um)
    {
        db = _db;
        rm = _rm;
        um = _um;
    }
    ...
}
```

Kontroler može pozivati samo autentifikovani korisnik koji je u roli admin

_ViewImports.cshtml

```
@using WebApp08
@using WebApp08.Models
@using  Microsoft.AspNetCore.Identity
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

@using Microsoft.AspNetCore.Identity direktiva je potrebna u **_ViewImports.cshtml** fajlu jer omogućava **.cshtml** fajlu da prepozna klase kao što su **IdentityRole**, **UserManager<ApplicationUser>**, i **RoleManager<IdentityRole>**

Index akcija kontrolera Role

```
// Akcija za prikaz svih rola
public IActionResult Index(string poruka)
{
    ViewBag.Poruka = poruka;
    return View(rm.Roles.ToList());
}
```

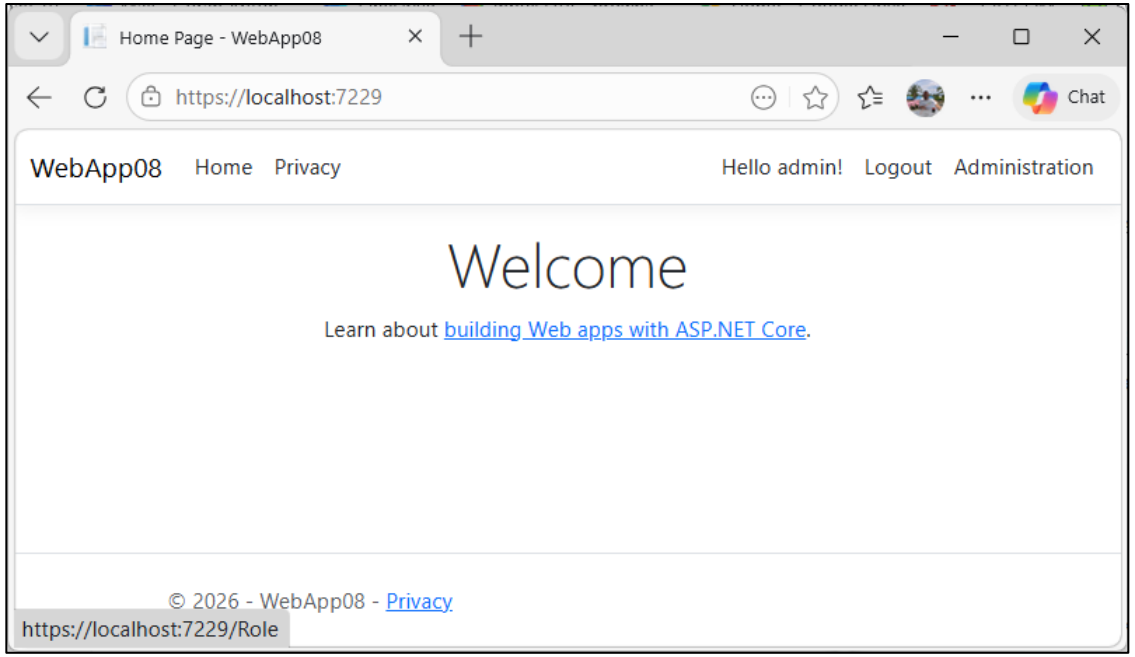
Index pogled kontrolera Role

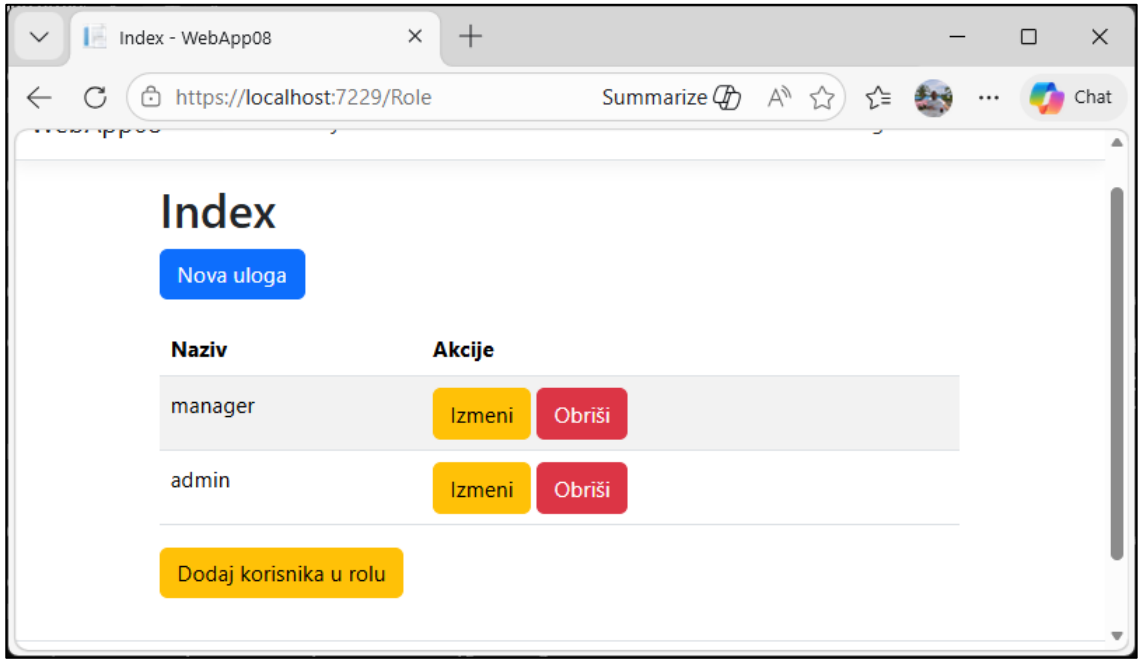
```
<h1>@ViewData["Title"]</h1>

@if (ViewBag.Poruka != null)
{
    <div class="alert alert-success">@ViewBag.Poruka</div>
}

<p><a class="btn btn-primary" asp-action="Create">Nova uloga</a></p>

<table class="table table-striped">
    <thead>
        <tr><th>Naziv</th><th>Akcije</th></tr>
    </thead>
    <tbody>
        @foreach (IdentityRole rola in Model)
        {
            <tr>
                <td>@rola.Name</td>
                <td>
                    <a class="btn btn-warning" asp-action="Edit" asp-route-roleName="@rola.Name">Izmeni</a>
                    <a class="btn btn-danger" asp-action="Delete" asp-route-imeRole="@rola.Name">Obriši</a>
                </td>
            </tr>
        }
    </tbody>
</table>

<p><a class="btn btn-warning" asp-action="DodajKorisnikaUrolu">Dodaj korisnika u rolu</a></p>
```

Modifikacija parcijalnog pogleda za logovanje

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

<ul class="navbar-nav">

    @if (SignInManager.IsSignedIn(User))
    {
        ApplicationUser au = await UserManager.FindByNameAsync(User.Identity.Name);
         bool isAdmin = await UserManager.IsInRoleAsync(au, "admin");

        <li class="nav-item">
            <a class="nav-link text-dark">Hello @au.UserName!</a>
        </li>
        <li class="nav-item">
            <form class="form-inline" asp-controller="Account" asp-action="Logout"
            asp-route-returnUrl="@Url.Action("Index", "Home")">
                <button type="submit" class="nav-link btn btn-link text-dark">Logout</button>
            </form>
        </li>

        @if (isAdmin)
         {
             <li class="nav-item">
                 <a class="nav-link text-dark" asp-controller="Role"
                 asp-action="Index">Administration</a>
             </li>
         }
    }
```

Administratorski meni i prikaz rola





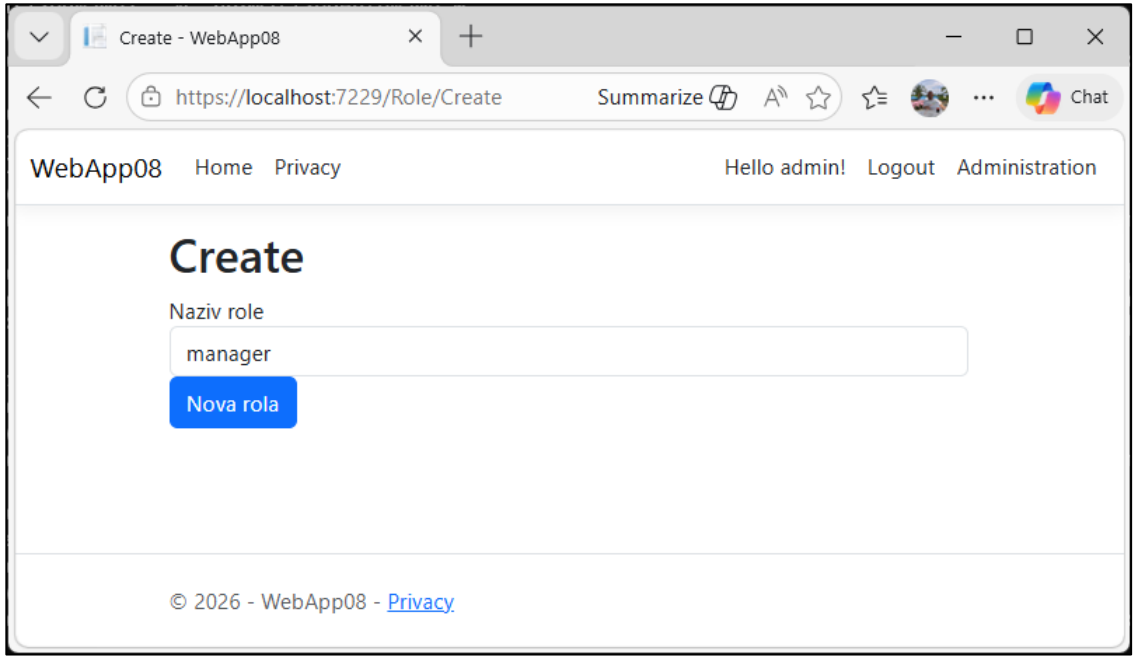
GET i POST metoda CREATE

```
// Akcija za kreiranje nove role
public IActionResult Create()
{
    return View();
}
// Kreiranje nove role (POST)
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create(IdentityRole ir)
{
    // Provera da li rola već postoji
    var rolaPostoji = await rm.RoleExistsAsync(ir.Name);
    if (rolaPostoji)
    {
        ViewBag.Poruka = "Rola sa tim imenom već postoji.";
        return View(ir);
    }
    // Kreiranje nove role
    var rezultat = await rm.CreateAsync(ir);
    if (rezultat.Succeeded)
    {
        ViewBag.Poruka = $"Kreirana rola: {ir.Name}";
        return RedirectToAction("Index");
    }
    else
    {
        ViewBag.Poruka = "Greška pri kreiranju role.";
        return View(ir);
    }
}
```

Pogled Create

```
@model IdentityRole

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

<h1>@ViewData["Title"]</h1>

<div class="row">
    <div class="col-md-6">
        <form asp-action="Create" method="post">
            <div class="form-group">
                <label for="Name">Naziv role</label>
                <input asp-for="Name" class="form-control" data-val="true" data-val-required="Unesite naziv role" />
                <span asp-validation-for="Name"></span>
            </div>
            <button type="submit" class="btn btn-primary">Nova rola</button>
        </form>
        <br />
        <span>
            @ViewBag.Poruka
        </span>
    </div>
</div>

@section Scripts {
    <script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>
    <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>
}
```

Kreiranje nove role



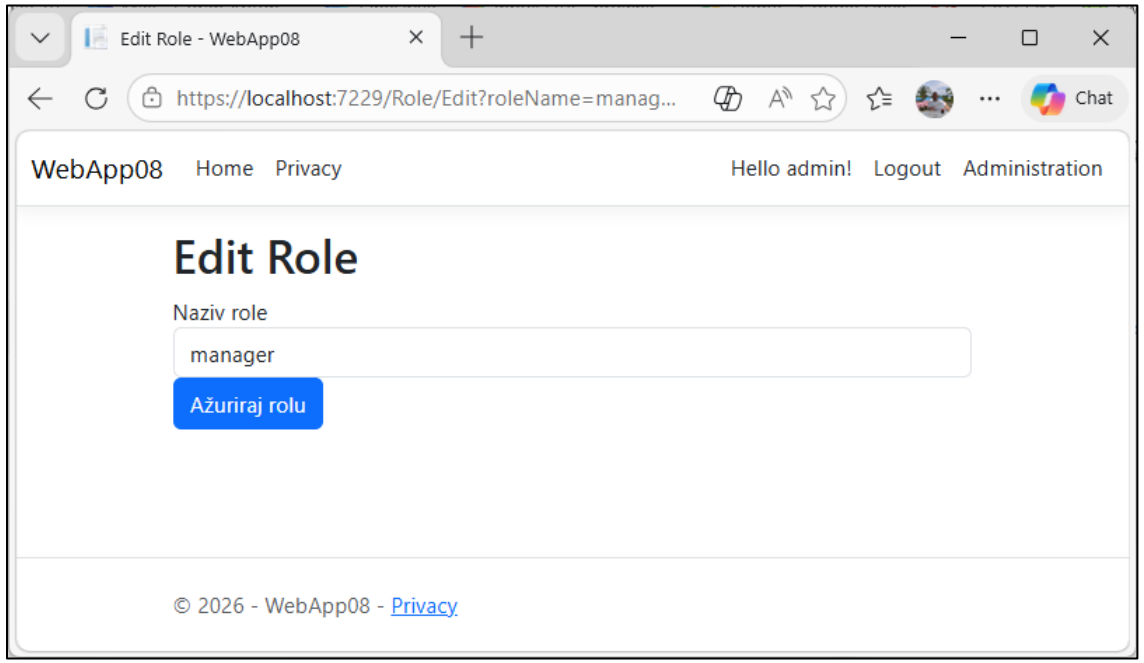
Metode za editovanje role

```
public async Task<ActionResult> Edit(string roleName)
{
    // Pronađi rolu po imenu
    IdentityRole role = await rm.FindByNameAsync(roleName);

    // Vraćanje role u View
    return View(role);
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Edit(IdentityRole role)
{
    try
    {
        // Pronađi rolu po Id-u
        IdentityRole r = await rm.FindByIdAsync(role.Id);

        // Ažuriranje imena role
        r.Name = role.Name;

        // Spremanje promjena
        await rm.UpdateAsync(r);

        // Preusmeravanje na Index
        return RedirectToAction("Index");
    }
    catch (Exception)
    {
        // U slučaju greške, vraćanje na View sa rolom
        return View(role);
    }
}
```

Pogled za editovanje role

```
@model IdentityRole

@{
    ViewData["Title"] = "Edit Role";
}

<h1>@ViewData["Title"]</h1>

<!-- Forma za editovanje role -->
<form asp-action="Edit" method="post">
    <div class="form-group">
        <label for="Name">Naziv role</label>
        <input asp-for="Name" class="form-control" data-val="true" data-val-required="Unesite naziv role" />
        <span asp-validation-for="Name"></span>
    </div>
    <button type="submit" class="btn btn-primary">Ažuriraj rolu</button>
</form>

<!-- Poruka o grešci ili uspehu -->
<br />
@if (ViewBag.Poruka != null)
{
    <div class="alert alert-danger">
        @ViewBag.Poruka
    </div>
}

@section Scripts {
    <script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>
    <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>
}
```

Editovanje role



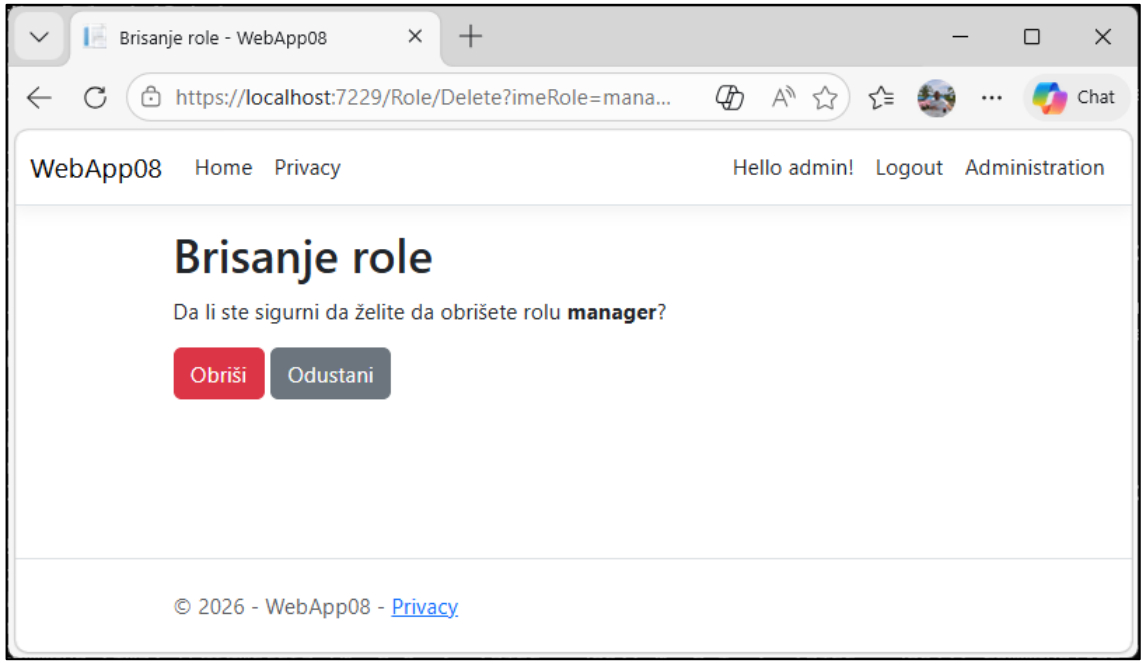
Metode za brisanje role

```
// Akcija za prikazanje role koja se briše
public async Task<IActionResult> Delete(string imeRole)
{
    IdentityRole rola = await rm.FindByNameAsync(imeRole);
    if (rola == null)
    {
        ViewBag.Poruka = "Rola nije pronađena.";
        return RedirectToAction("Index");
    }
    return View(rola);
}

// Brisanje role (POST)
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(string imeRole)
{
    IdentityRole rola = await rm.FindByNameAsync(imeRole);
    if (rola == null)
    {
        ViewBag.Poruka = "Rola nije pronađena.";
        return RedirectToAction("Index");
    }

    // Brisanje role iz baze
    db.Roles.Remove(rola);
    await db.SaveChangesAsync();

    ViewBag.Poruka = "Rola uspešno obrisana.";
    return RedirectToAction("Index");
}
```

Pogled za brisanje role

```
@model IdentityRole

@{
    ViewData["Title"] = "Brisanje role";
}

<h1>@ViewData["Title"]</h1>

<p>Da li ste sigurni da želite da obrišete rolu <strong>@Model.Name</strong>?</p>

<form method="post">
    <button type="submit" class="btn btn-danger">Obriši</button>
    <a href="@Url.Action("Index")" class="btn btn-secondary">Odustani</a>
</form>
```

Brisanje role



Metoda AddUserToRole - GET

```
// Akcija za dodeljivanje korisnika u rolu
public IActionResult DodajKorisnikaUrolu()
{
    ViewBag.Korisnici = new SelectList(db.Users, "UserName", "UserName");
    ViewBag.Uloge = new SelectList(db.Roles, "Name", "Name");
    return View();
}
```

```
public SelectList (System.Collections.IEnumerable items, string dataValueField,
string dataTextField);
```

Klasa SelectList služi za popunjavanje select elementa automatski generisanim option elementima.
Instanca klase SelectList prosleđuje se pogledu posredstvom ViewBag polja.
SelectTagHelper pomoću atributa asp-items povezuje select element sa ViewBag poljem.

```
<select name="korisnik" asp-items="ViewBag.Korisnici" class="form-
control">
```

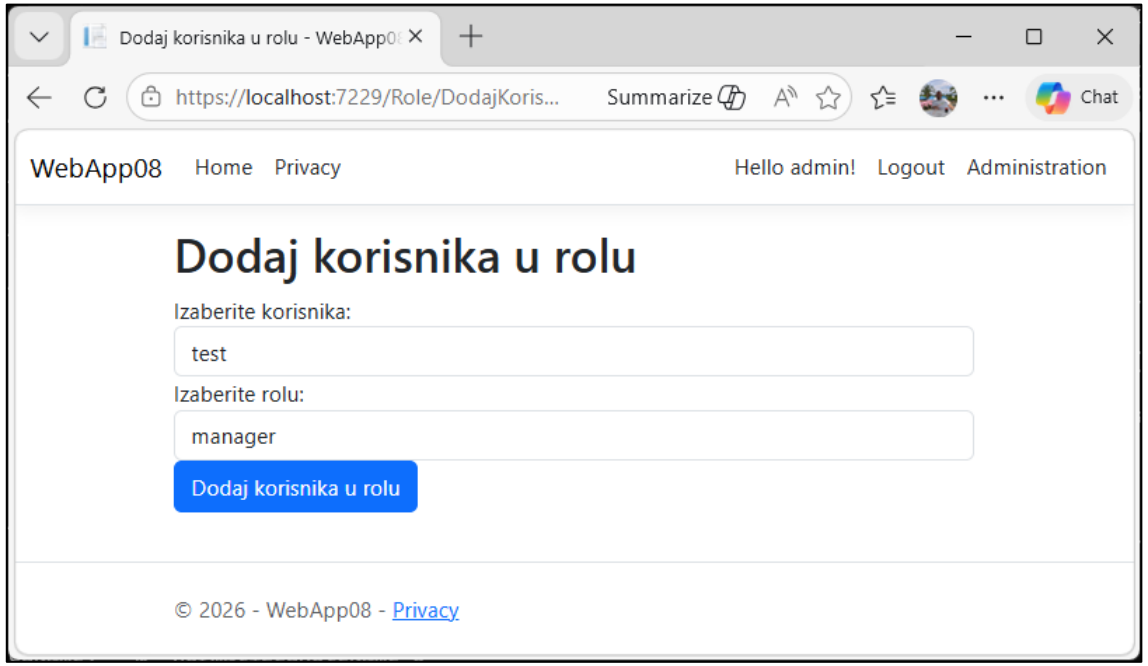
Metoda AddUserToRole - POST

```
// Dodavanje korisnika u rolu (POST)
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DodajKorisnikaUrolu(string korisnik, string rola)
{
    ViewBag.Korisnici = new SelectList(db.Users, "UserName", "UserName");
    ViewBag.Uloge = new SelectList(db.Roles, "Name", "Name");

    ApplicationUser user = await um.FindByNameAsync(korisnik);

    if (user == null)
    {
        ViewBag.Poruka = $"Korisnik {korisnik} nije pronađen.";
        return View();
    }

    // Provera da li je korisnik već u roli
    bool uRoliJe = await um.IsInRoleAsync(user, rola);
    if (uRoliJe)
    {
        ViewBag.Poruka = $"Korisnik {korisnik} je već u roli {rola}.";
        return View();
    }

    // Dodavanje korisnika u rolu
    var dodaj = await um.AddToRoleAsync(user, rola);
    if (dodaj.Succeeded)
    {
        ViewBag.Poruka = $"Korisnik {korisnik} je uspešno dodat u rolu {rola}.";
        return RedirectToAction("Index");
    }
    else
    {
        ViewBag.Poruka = $"Greška pri dodavanju korisnika {korisnik} u rolu {rola}.";
        return View();
    }
}
```

DodajKorisnikaUrolu.cshtml

```
@{
    ViewData["Title"] = "Dodaj korisnika u rolu";
}

<h1>@ViewData["Title"]</h1>

<!-- Forma za dodeljivanje korisnika u rolu -->
<form method="post">
    <div class="form-group">
        <label for="korisnik">Izaberite korisnika:</label>
        <select asp-items="ViewBag.Korisnici" name="korisnik" class="form-control">
            <option value="0">Odaberi korisnika</option>
        </select>
    </div>

    <div class="form-group">
        <label for="rola">Izaberite rolu:</label>
        <select asp-items="ViewBag.Uloge" name="rola" class="form-control">
            <option value="0">Izaberite rolu</option>
        </select>
    </div>

    <button type="submit" class="btn btn-primary">Dodaj korisnika u rolu</button>
</form>

@if (ViewBag.Poruka != null)
{
    <div class="alert alert-success mt-2">
        @ViewBag.Poruka
    </div>
}
```

Dodavanje korisnika u ulogu



Autorizacija metode bazirano na rolama

```
[Authorize(Roles = "admin")]
public ActionResult OnlyAdmin()
{
    return View("AdminOrManager");
}

[Authorize(Roles = "admin, manager")]
public IActionResult AdminOrManager()
{

    return View();
}
```

Metode kontrolera Home

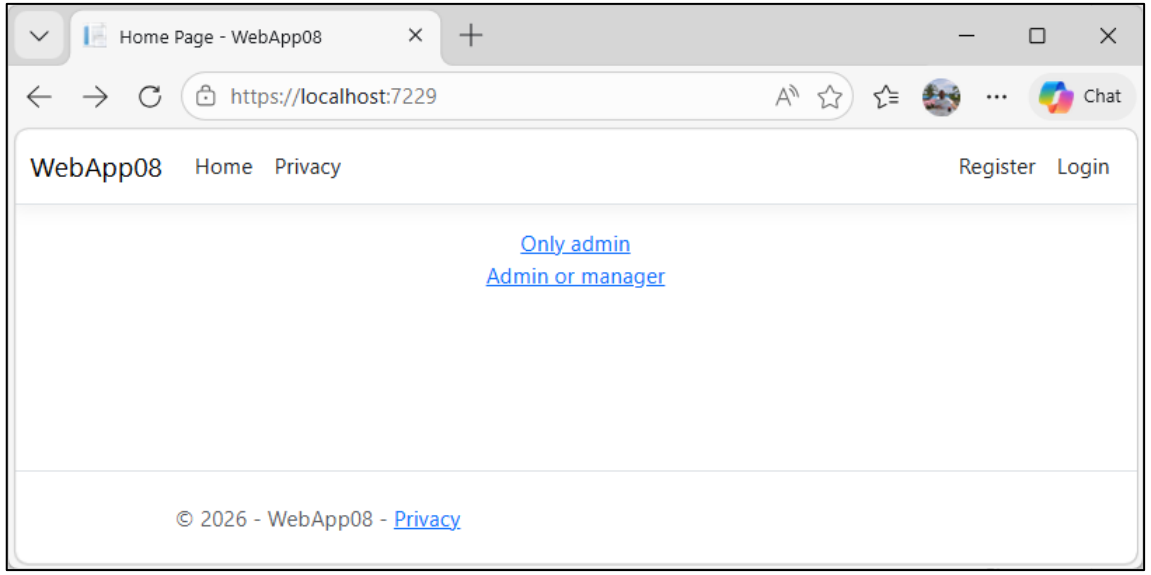
Metodu OnlyAdmin može da poziva samo korisnik u roli admin.
Metodu AdminOrManager može da poziva samo korisnik koji je u roli admin ili manager

```
@inject UserManager<ApplicationUser> um

@{
    ViewData["Title"] = "Admin or Manager";

    ApplicationUser user = await um.GetUserAsync(User);

    IList<string> roles = await um.GetRolesAsync(user);
}

<h2>@ViewBag.Poruka</h2>

<h2>Administracija</h2>

<dl class="row">
    <dt class="col-sm-2">First Name</dt>
    <dd class="col-sm-10">@user.FirstName</dd>

    <dt class="col-sm-2">Last Name</dt>
    <dd class="col-sm-10">@user.LastName</dd>
</dl>
    Role
<ul>
    @foreach (var role in roles)
    {
        <li>@role</li>
    }
</ul>

<a asp-action="Index">Index</a>
```

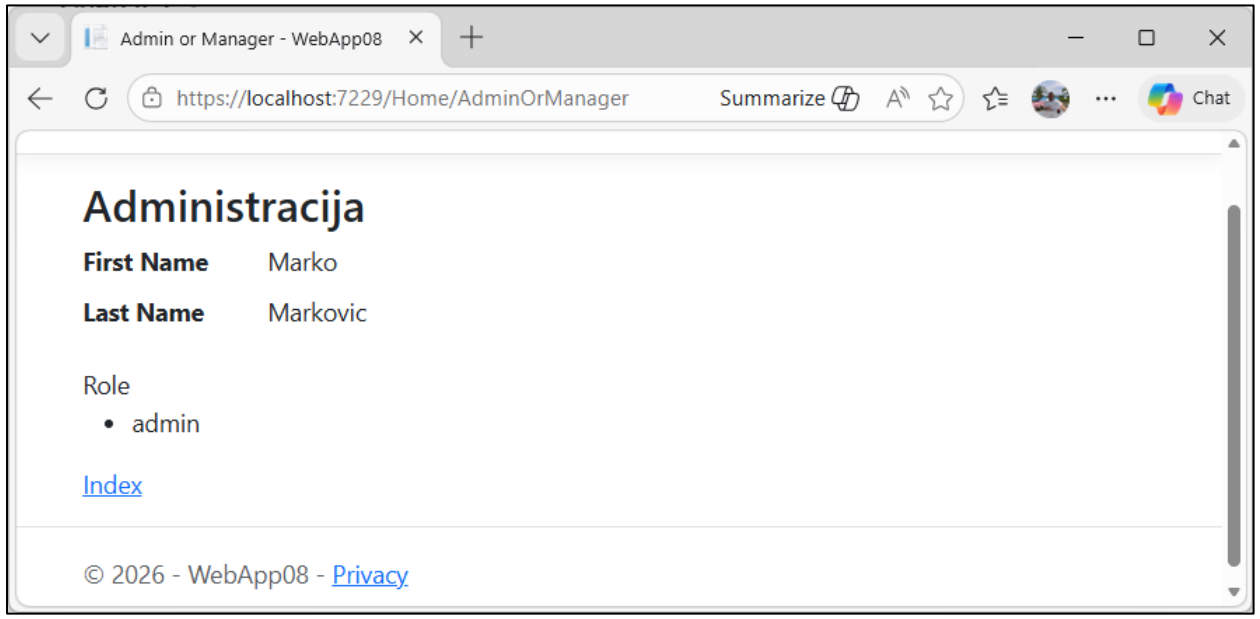
Svojstvo User unutar cshtml fajla daje informacije o logovanom korisniku

Pogled AdminOrManager.cshtml

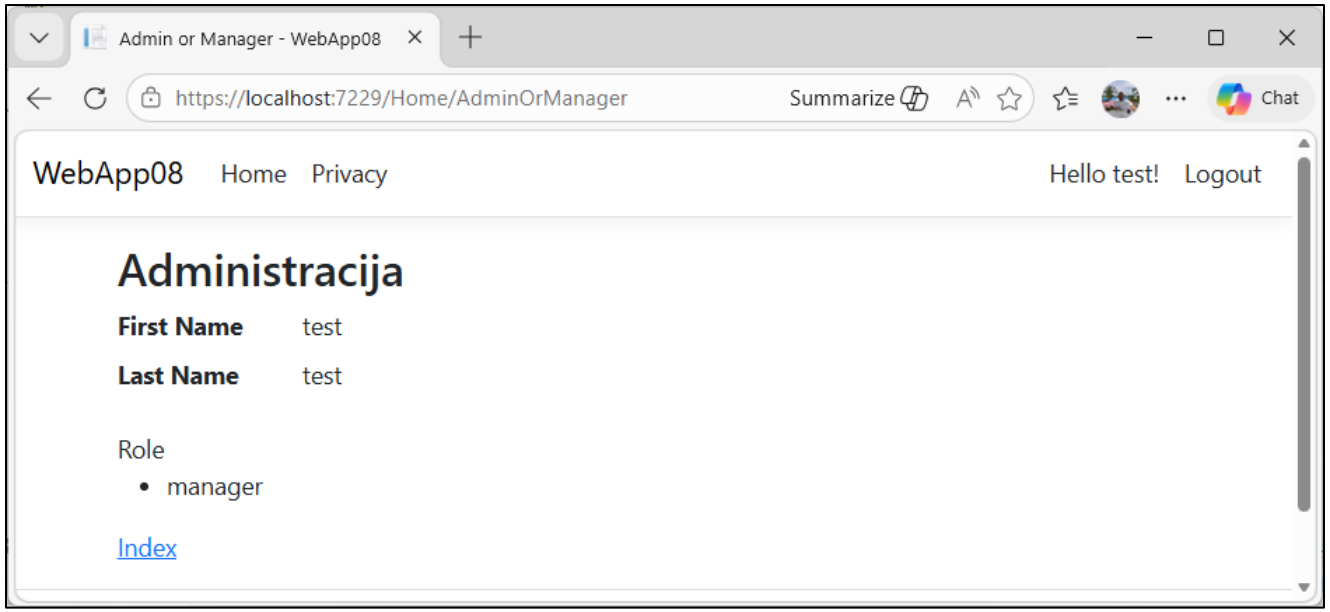
Početna strana aplikacije



Admin rola


Admin or Manager - WebApp08
https://localhost:7229/Home/AdminOrManager
Summarize
Chat
Administracija
First Name Marko
Last Name Markovic
Role
admin
Index
© 2026 - WebApp08 - Privacy

Manager rola

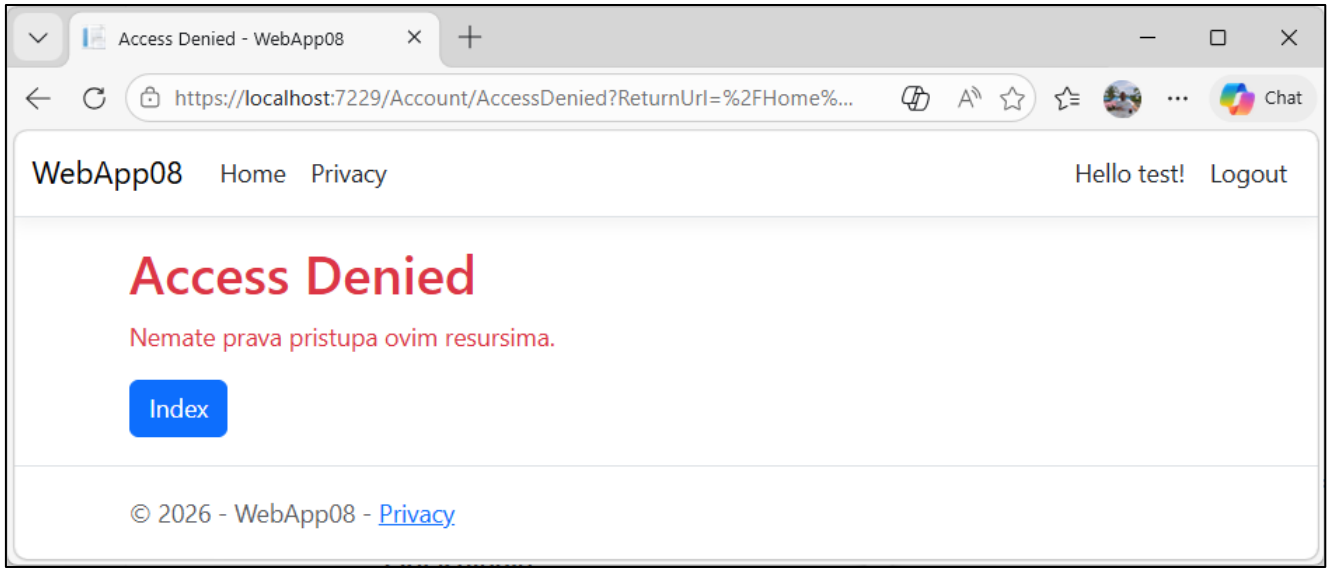


AccountController metoda AccessDenied

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

```
@{
    ViewData["Title"] = "Access Denied";
}

<header>
    <h1 class="text-danger">@ViewData["Title"]</h1>
    <p class="text-danger">Nemate prava pristupa ovim resursima.</p>
</header>

<!-- Link za povratak na početnu stranicu -->
<a asp-controller="Home" asp-action="Index" class="btn btn-primary">Index</a>
```

Korisnik poziva metodu za koju nema dozvolu



Pitanje 1

Identity system kao model klasu za rad sa rolama koristi klasu baziranu na baznoj klasi?

a. UserManager
b. IdentityRole
c. ApplicationUser

Odgovor: b

Pitanje 2

Neka je ApplicationUser klasa izvedena iz klase IdentityUser. Metoda AddIdentity<ApplicationUser, IdentityRole>() služi za:

a. Definisanje role ApplicationUser
b. Podešavanje middleware komponente za autorizaciju
c. Registraciju servisa koji omogućavaju korišćenje identity sistema za autentifikaciju uključujući i rad sa rolama

Odgovor: c

Pitanje 3

Biblioteka Microsoft.AspNetCore.Identity za kreiranje role definiše klasu:

a. Roles
b. RoleManager
c. ApplicationRole

Odgovor: b

Pitanje 4

Metoda **IsInRoleAsync** kojom se proverava da li se korisnik nalazi u roli, pripada sledećoj klasi iz biblioteke Microsoft.AspNetCore.Identity:

a. UserManager
b. RoleManager
c. ApplicationRole

Odgovor: a

Pitanje 5

Koje svojstvo se koristi u konfiguraciji autentifikacionog kolačića kako bi se definisala stranica na koju korisnik bude preusmeren ako pokuša da pristupi zabranjenim metodama ili resursima?

a. AccessDisabled
b. AccessDeniedPath
c. AccessPath

Odgovor: b

Pitanje 6

Potrebno je dozvoliti da akcionu metodu kontrolera mogu pozivati samo korisnici koju su u roli admin. To postižemo upotrebom sledećeg koda:

a. [Authorize(Roles(admin)]
b. [Authorize(Roles : admin)]
c. [Authorize(Roles = "admin")]

Odgovor: c