Administracija pomoću rola

Role

- Radi 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 model klasom ApplicationUser
- Kreirane su model klase 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 kukija

Konfiguracija identity sistema

Registracija Identity sistema se vrši korišćenjem metode AddIdentity, ne AddDefaultIdentity kao kada se ne koriste Role

```
builder.Services.AddIdentity<ApplicationUser, IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>();
builder.Services.Configure<IdentityOptions>(opcije => {

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

Konfigurisanje protočne obrade zahteva

```
app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

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

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

app.Run();
```

Pošto se koristi kastomizovana varijanta implementacije Identity sistema a
ne web razor biblioteka potrebno je zakomentarisati liniju koda kao što je pokazano

Kontroler za inicijalizaciju baze

```
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;
    }
}
```

Kreiranje korisnika admin

```
public async Task<IActionResult> KreirajKorisnikaAdmin()
{
    ApplicationUser admin = new ApplicationUser
    {
        UserName = "admin",
        FirstName = "Marko",
        LastName = "Markovic"
    };
    string lozinka = "123";

    var rezultat = await _userManager.CreateAsync(admin, lozinka);
    if (rezultat.Succeeded)
    {
        return View(admin);
    }
    else
    {
        return View();
    }

}
```

Pogled KreirajKorisnikaAdmin.cshtml

```
@{
    ViewData["Title"] = "Kreiranje korisnika";
}
@model ApplicationUser

@if (Model!=null)
{
    <p>
        Admin je kreiran. Nemojte vise pozivati ovu metodu
    </p>
    <dl class="row">
    <dt class="col-sm-2">
        First name:
    </dt>
    <dd class="col-sm-10">
        @Model.FirstName
    </dd>
    <dt class="col-sm-2">
        Last name:
    </dt>
    <dd class="col-sm-10">
        @Model.LastName
    </dd>
    <dt class="col-sm-2">
        User name:
    </dt>
    <dd class="col-sm-10">
        @Model.UserName
    </dd>
</dl>
}
else{
    <p>Admin nije kreiran</p>
}
<br />
<a asp-action="Index">Nazad na administraciju</a>
```

Metoda za kreiranje role admin

```
public async Task<IActionResult> KreirajRoluAdmin()
{
    IdentityRole rolaAdmin = new IdentityRole("admin");
    var rezultat = await _roleManager.CreateAsync(rolaAdmin);

    if (rezultat.Succeeded)
    {
        ViewBag.Poruka = "Kreirana rola admin";
    }
    else
    {
        ViewBag.Poruka = "Greska pri kreiranju role admin";
    }
    return View();
}
```

KreirajRoluAdmin.cshtml

```
@{
    ViewData["Title"] = "Kreiranje role admin";
}
<p>@ViewBag.Poruka</p>
<br />
<a asp-action="Index">Nazad na administraciju</a>
```

DodajKorisnikaUrolu

```
public async Task<IActionResult> DodajKorisnikaUrolu()
{
    ApplicationUser admin = await _userManager.FindByNameAsync("admin");
        var rez =   await _userManager.AddToRoleAsync(admin, "admin");
    if (rez.Succeeded)
    {
        ViewBag.Poruka = "Korisnik admin dodat u rolu admin";
    }

    else
    {
        ViewBag.Poruka = "Greska pri dodavanju korisnika u rolu";
    }
    return View();
}
```

DodajKorisnikaUrolu.cshtml

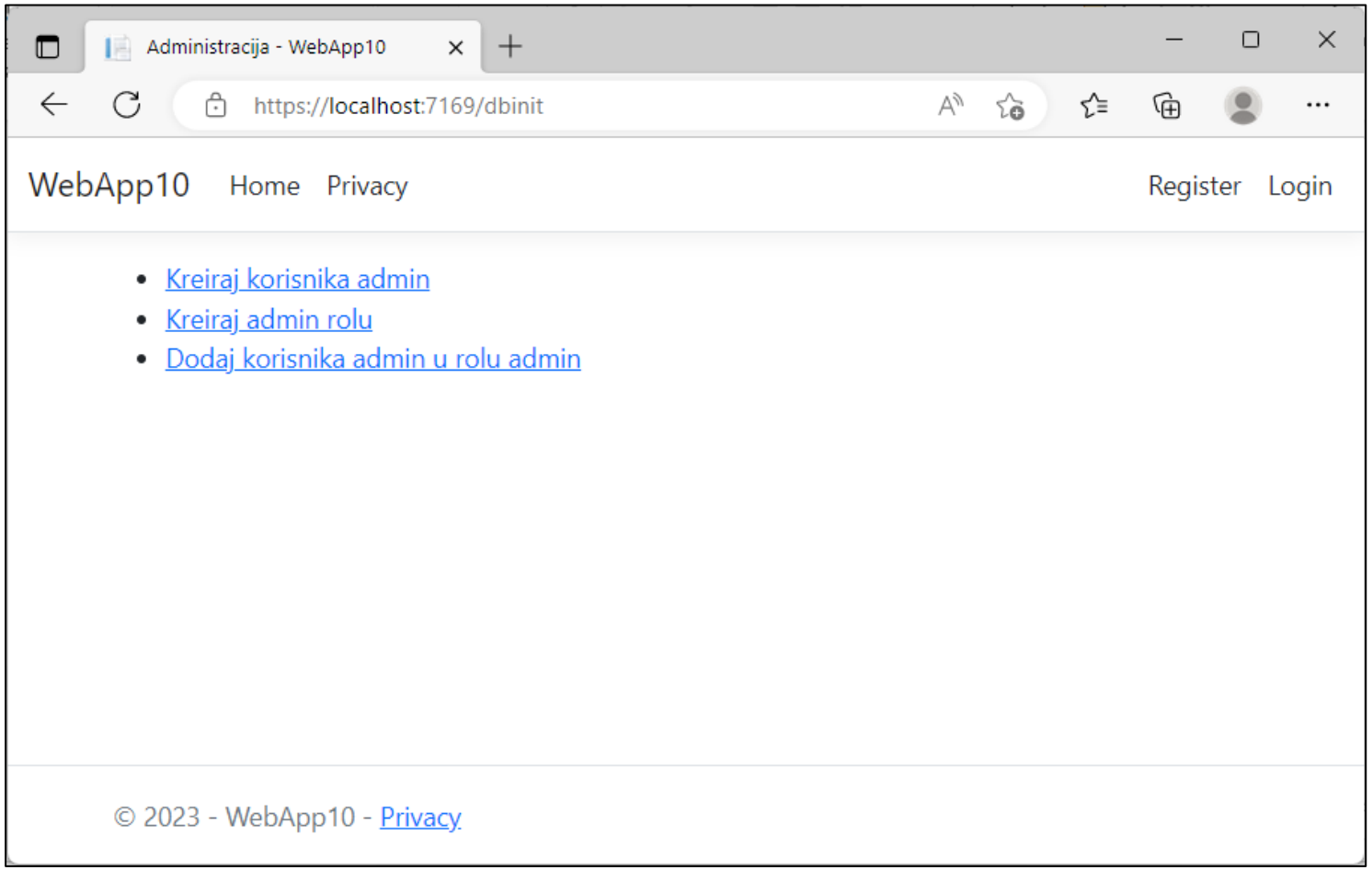
```
@{
    ViewData["Title"] = "Dodavanje korisnika u rolu";
}
<p>@ViewBag.Poruka</p>
<br />
<a asp-action="Index">Nazad na administraciju</a>
```

Index metoda i pogled DbInit kontrolera

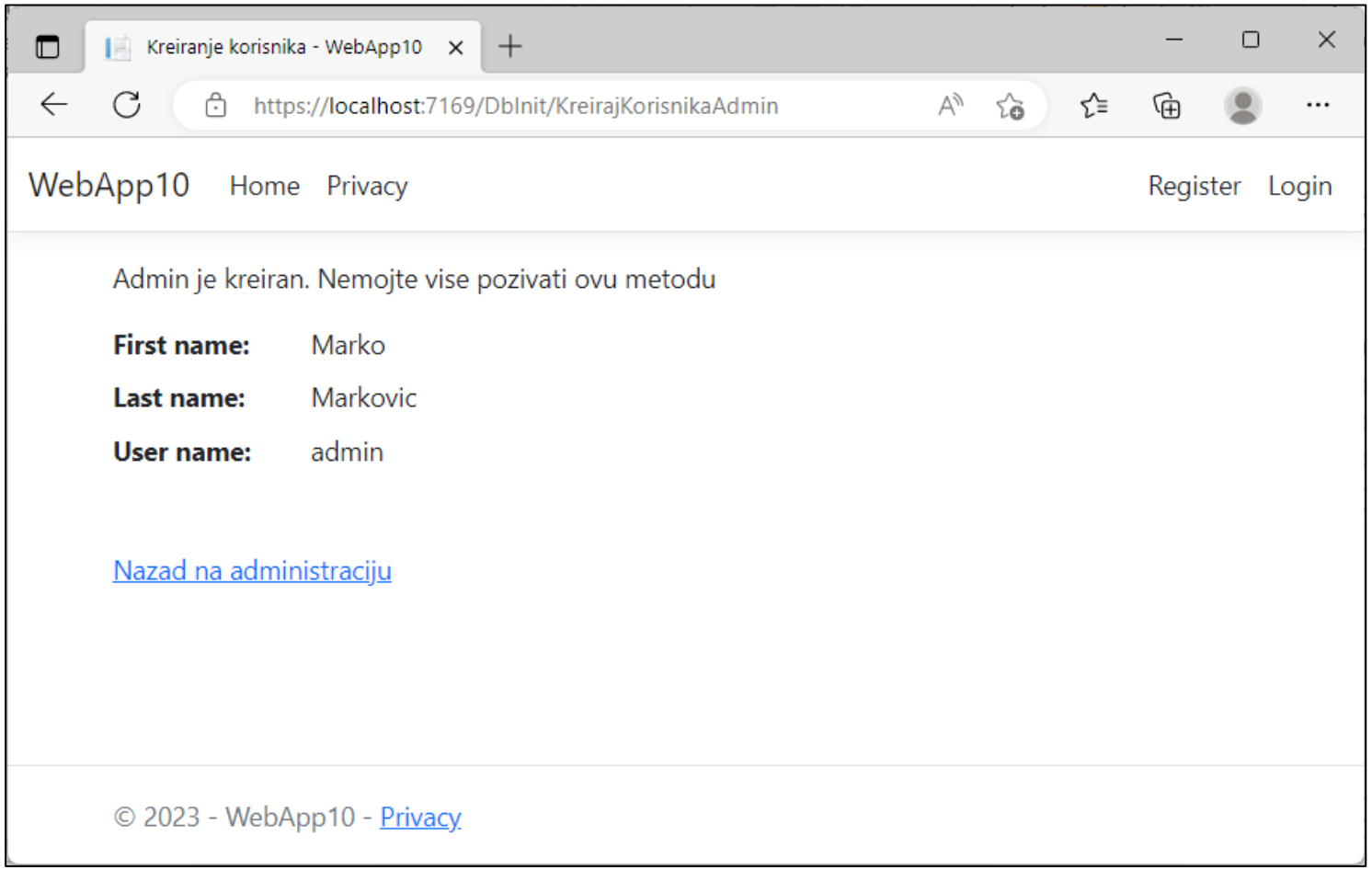
```
public ActionResult Index()
{
    return View();
}
```

```
@{
    ViewData["Title"] = "Administracija";
}
<ul>
    <li><a asp-action="KreirajKorisnikaAdmin">Kreiraj
korisnika admin</a></li>
    <li><a asp-action="KreirajRoluAdmin">Kreiraj admin
rolu</a></li>
    <li><a asp-action="DodajKorisnikaUrolu">Dodaj
korisnika admin u rolu admin</a></li>
</ul>
```

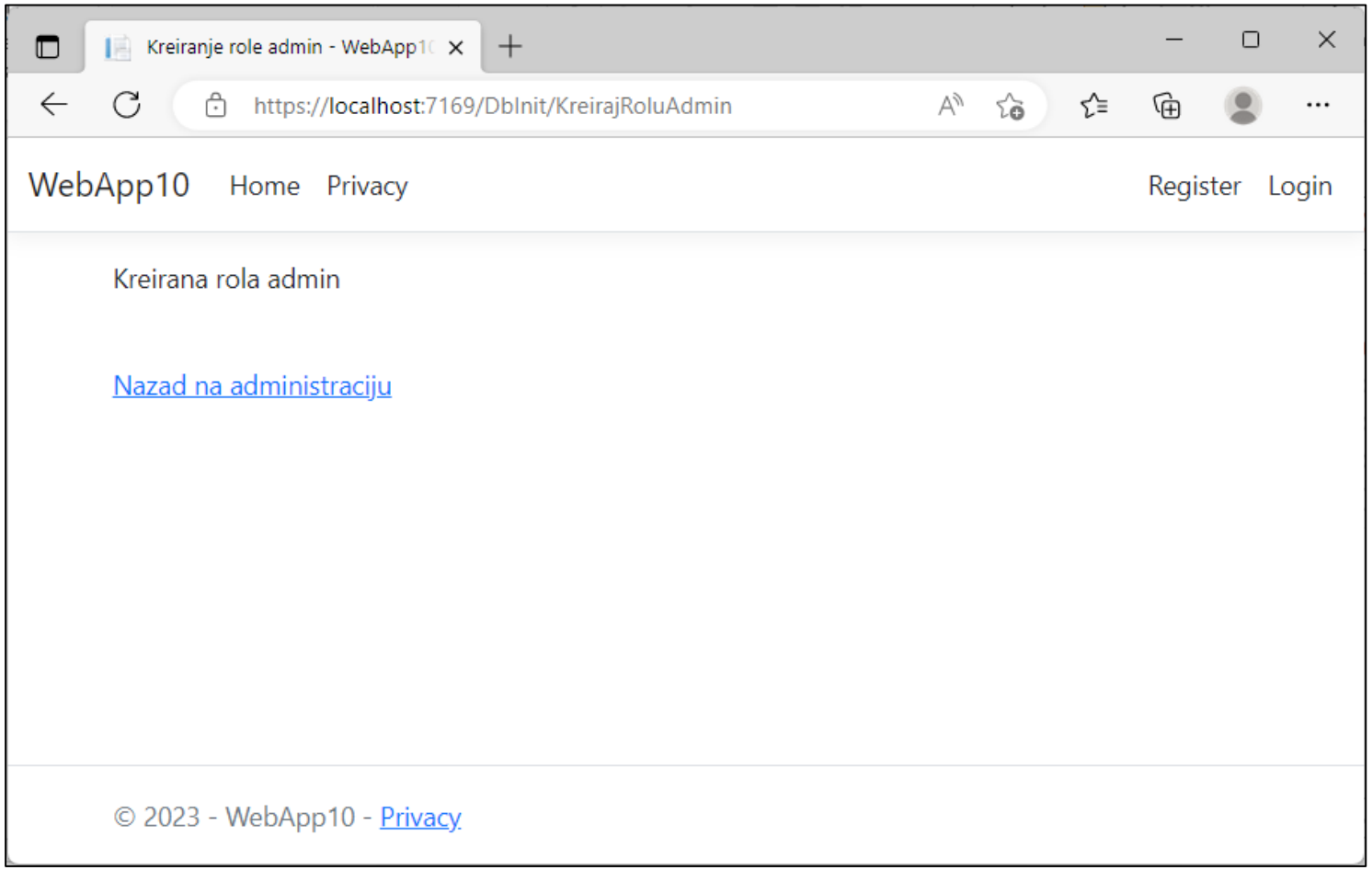
Strana za administraciju



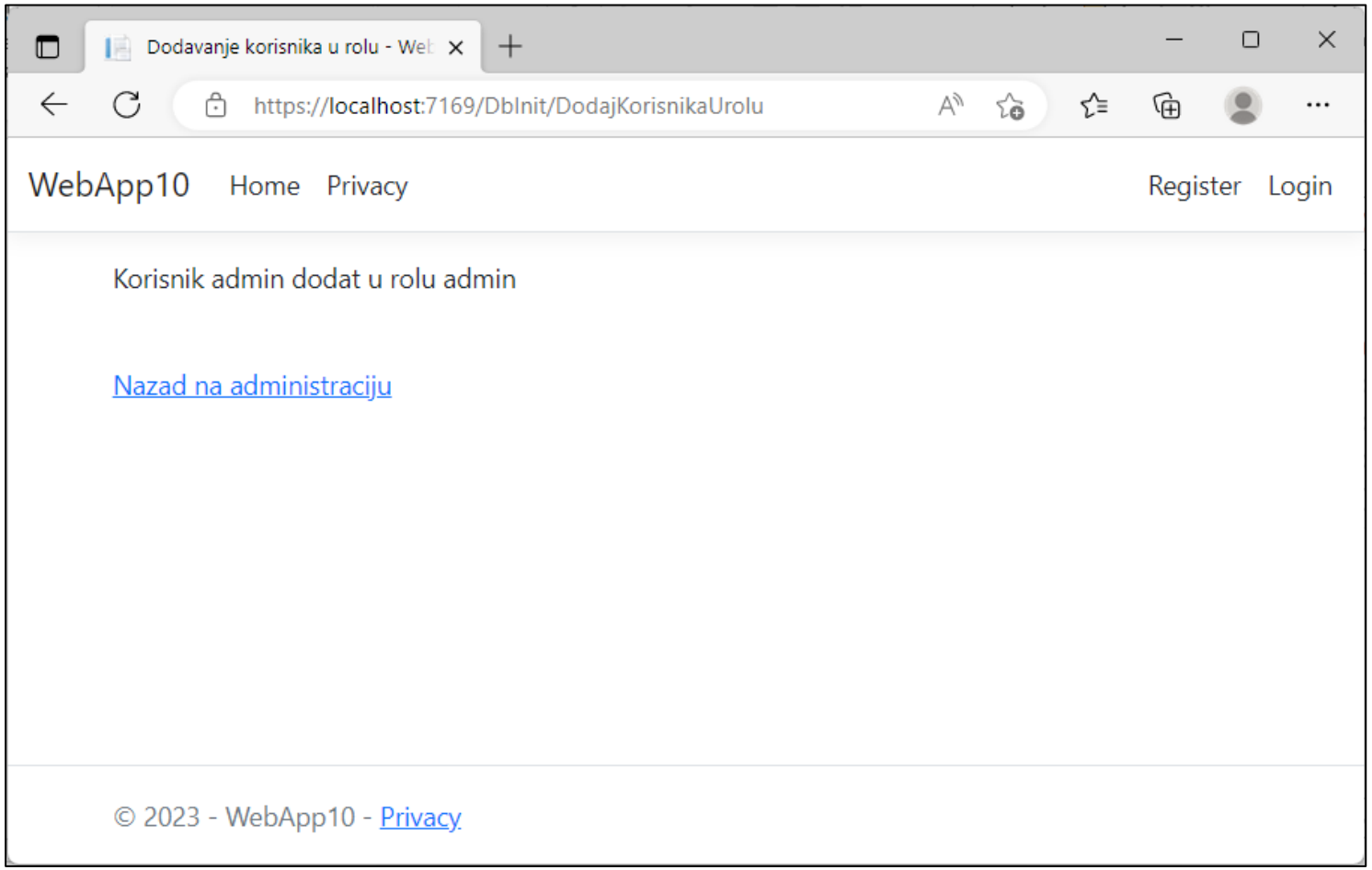
Kreiran korisnik admin



Kreirana rola admin



Korisnik admin dodat u rolu admin



Autorizacija kontrolera

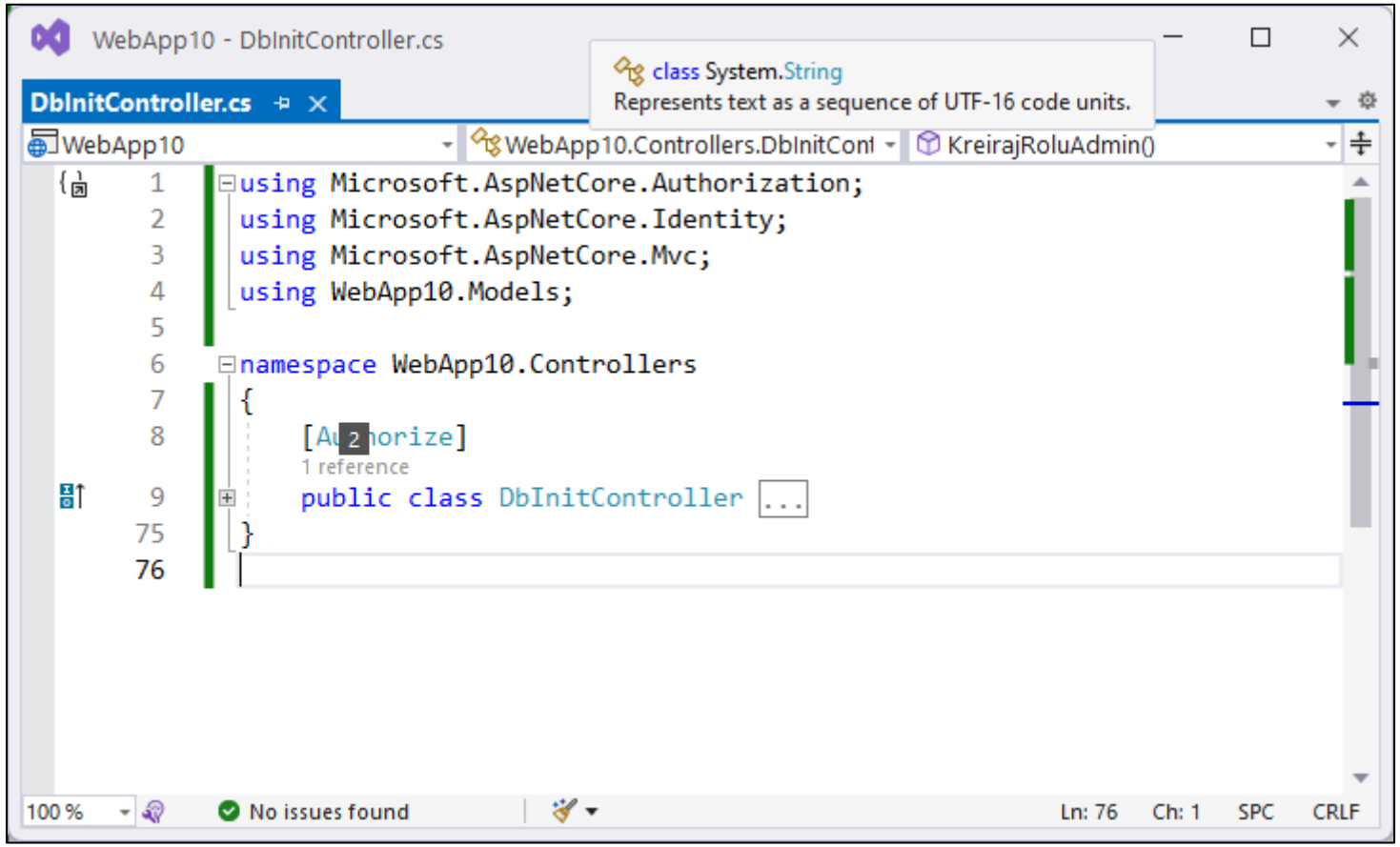


Tabela AspNet.Roles

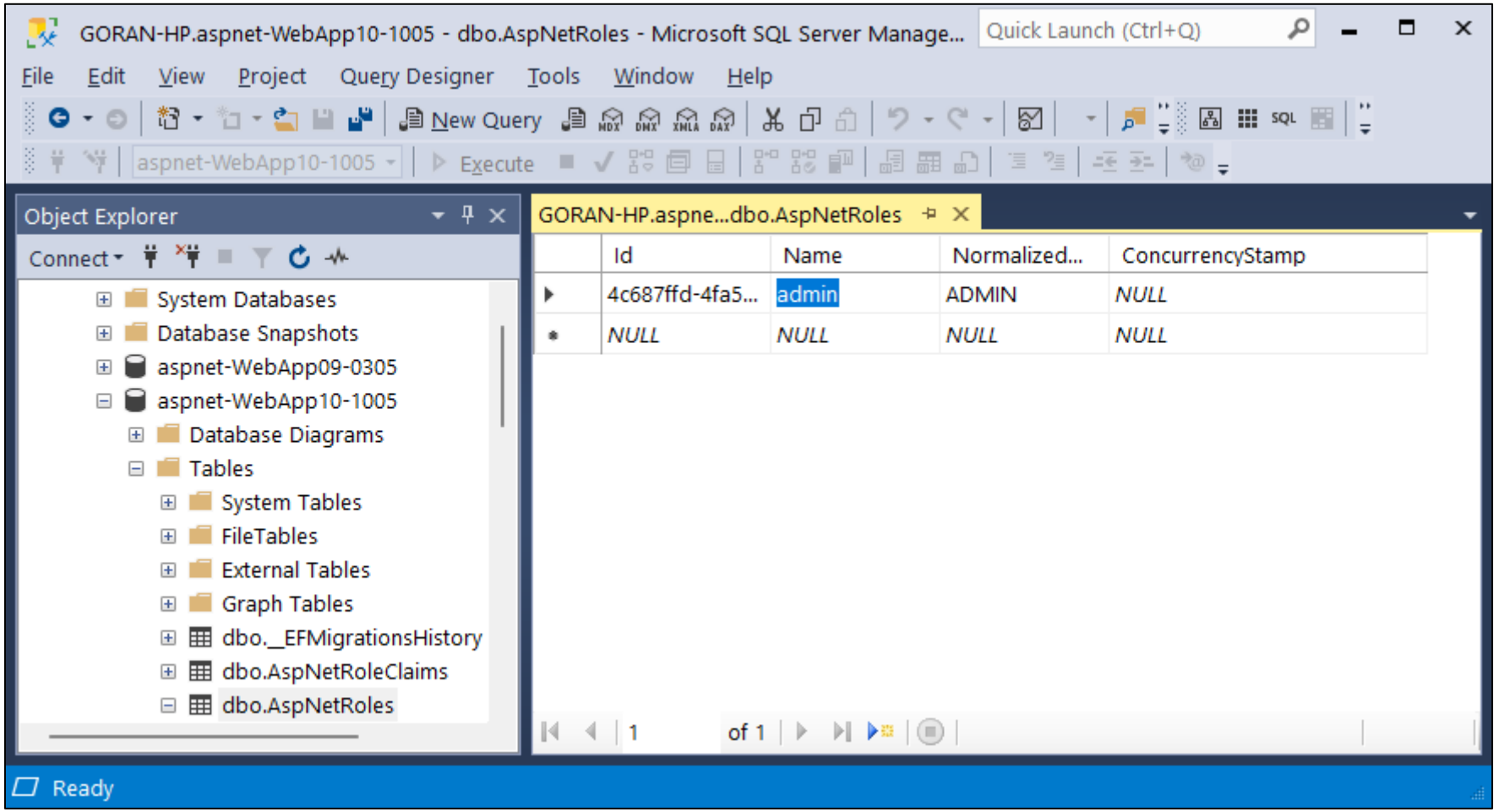
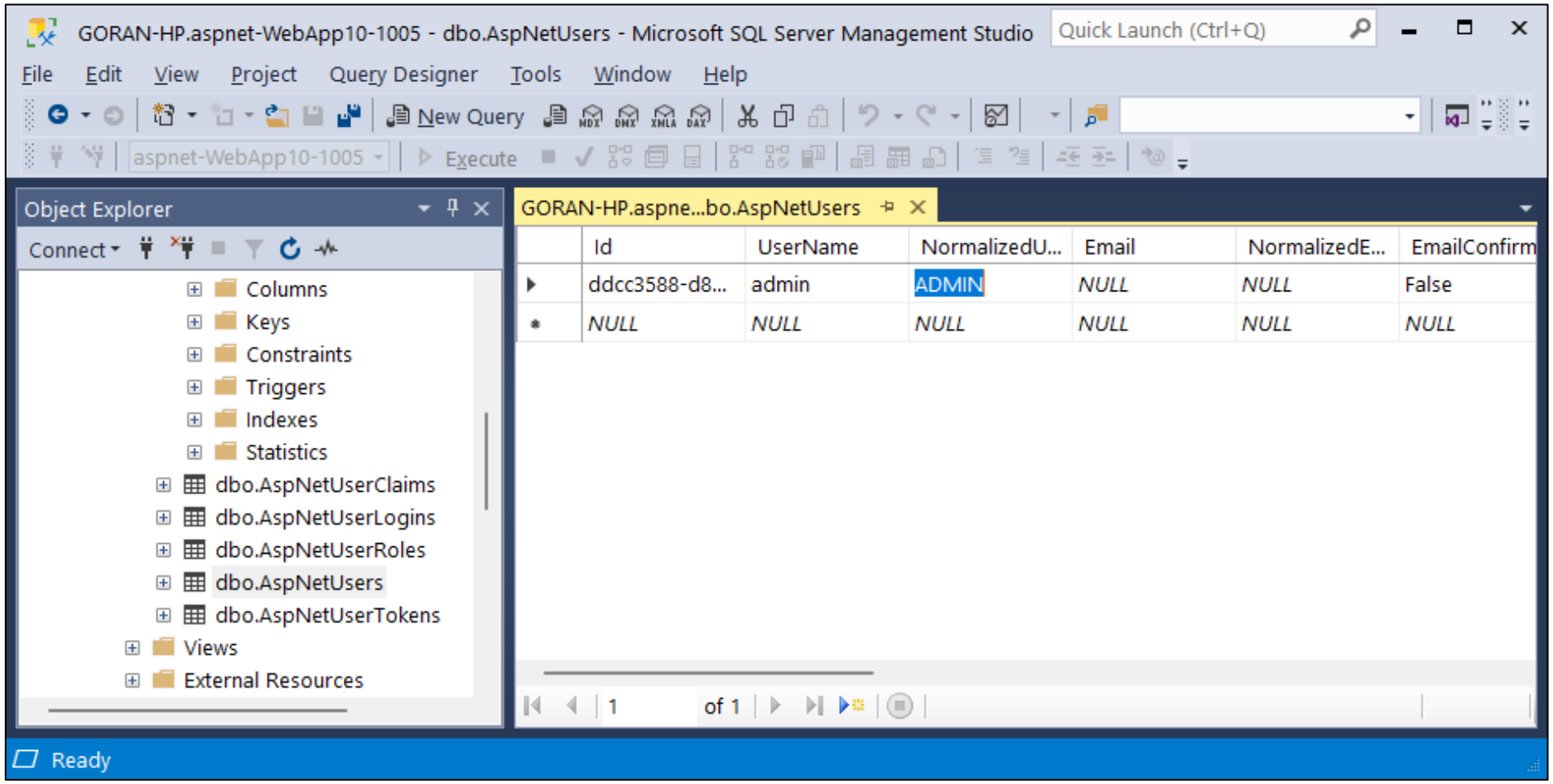
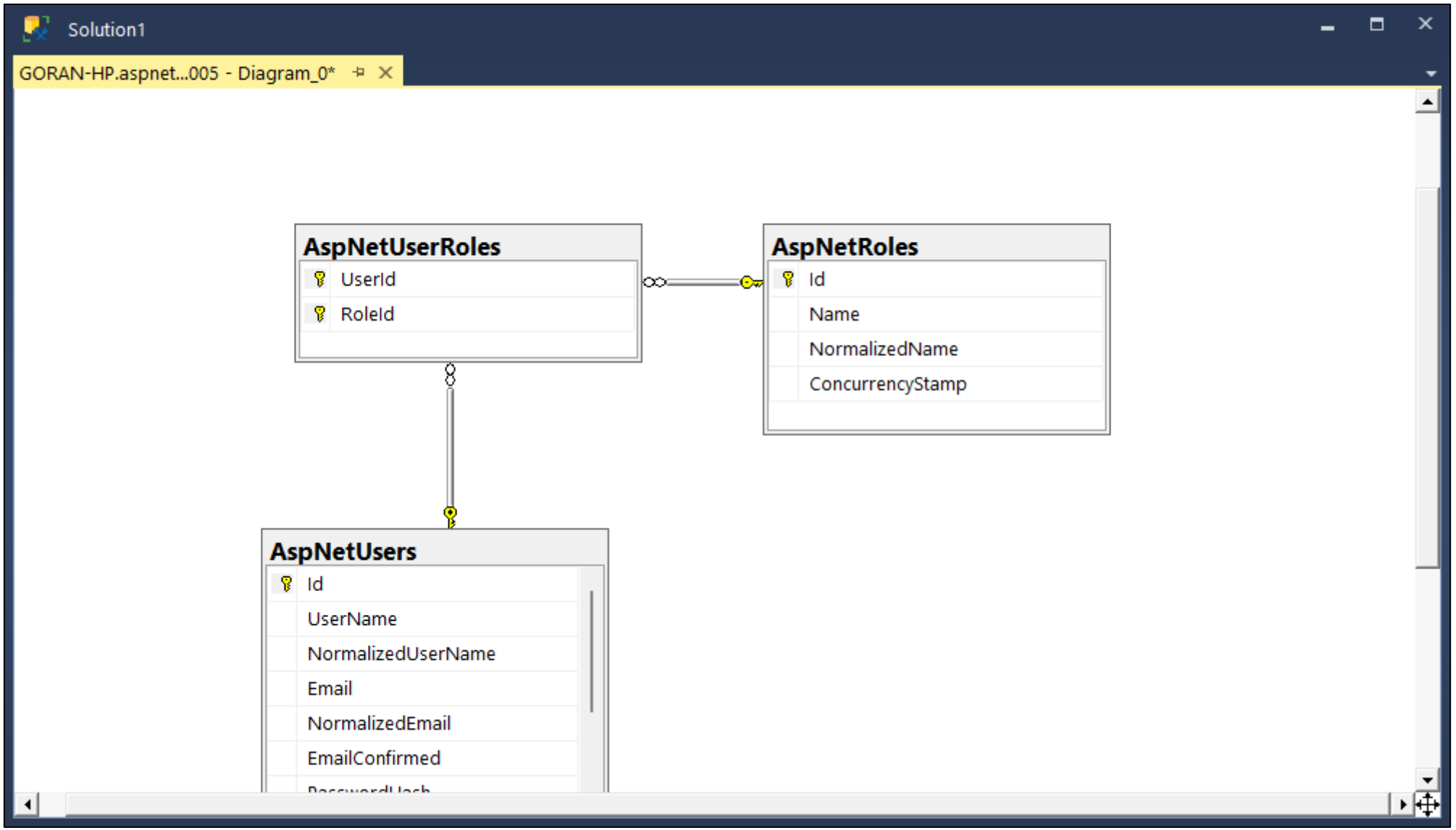


Tabela koja čuva role koje koristi aplikacija

Tabela ASpNetUsers





Veza tabela AspNetRoles i AspNetUsers

Kreiranje kontrolera za rad sa rolama

```
[Authorize(Roles = "admin")]
public class RoleController : Controller
{
    private UserManager<ApplicationUser> um;
    private RoleManager<IdentityRole> rm;


    public RoleController(UserManager<ApplicationUser> _um, RoleManager<IdentityRole> _rm)
    {
        um = _um;
        rm = _rm;
    }

 }
```

Kontroler može pozivati samo autentifikovani korisnik koji je u roli admin

Index akcija kontrolera Role

```
public IActionResult Index()
{
    List<IdentityRole> uloge = rm.Roles.ToList();
    return View(uloge);
}
```

_ViewImports.cshtml

```
@using WebApp10
@using WebApp10.Models
@using Microsoft.AspNetCore.Identity
@addTagHelper *,
Microsoft.AspNetCore.Mvc.TagHelpers
```

Index pogled kontrolera Role

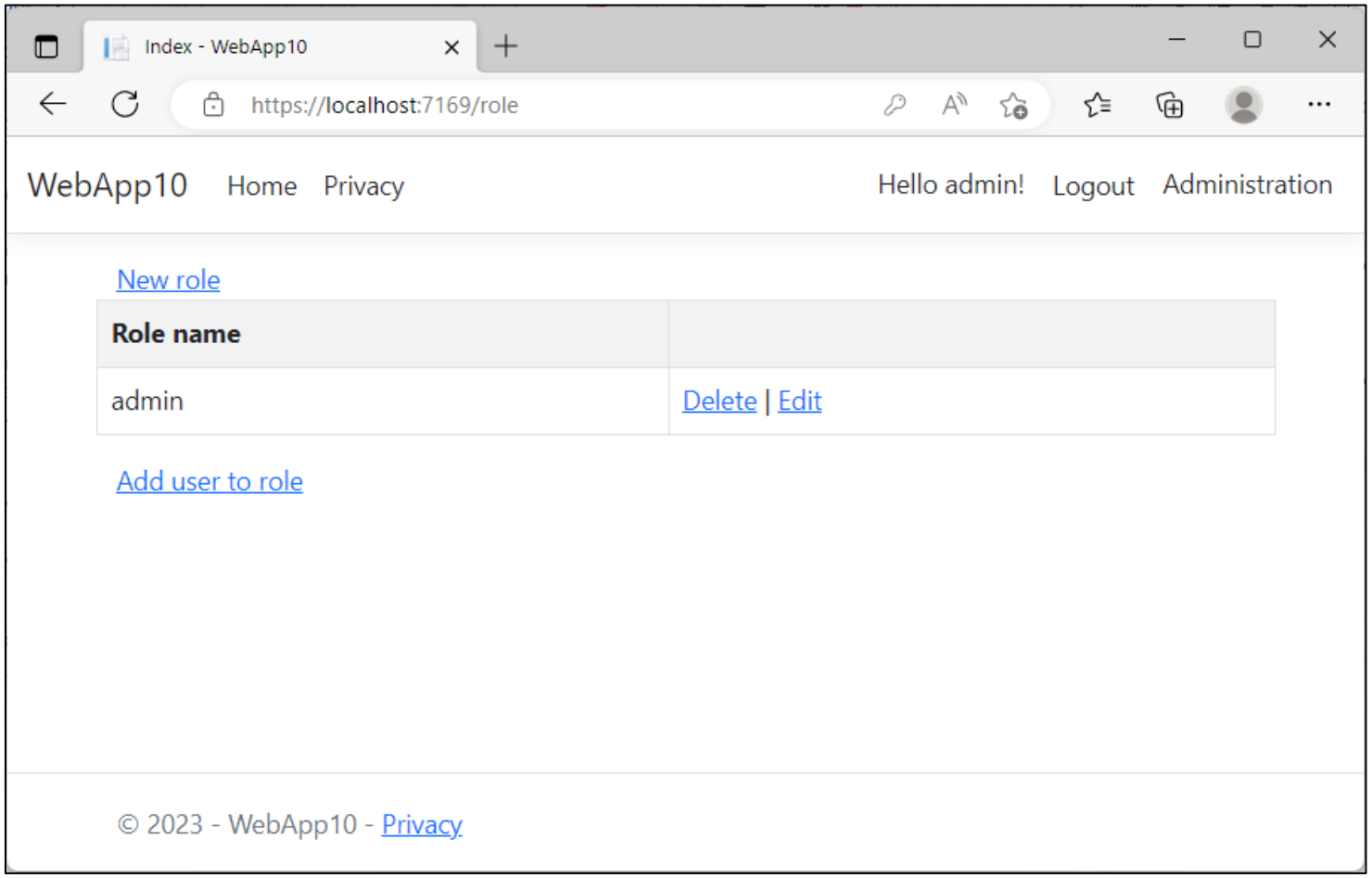
```
@model IEnumerable<IdentityRole>
@{
    ViewData["Title"] = "Index";
}
<div class="row">
    <a asp-action="Create">New role</a>
</div>

<div class="row">
<table class="table table-bordered table-striped">
    <tr>
        <th>Role name</th>
        <th></th>
    </tr>

    @foreach (var r in Model)
    {
        <tr>
            <td>@r.Name</td>
            <td>
                <a asp-action="Delete" asp-route-roleName="@r.Name">Delete</a> |
                <a asp-action="Edit" asp-route-roleName="@r.Name">Edit</a>
            </td>
        </tr>
    }
</table>
</div>
<div class="row">
    <a asp-action="AddUserToRole">Add user to role</a>
</div>
<h3>@ViewBag.Message</h3>
```

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>
            }



        }
```

Dodaje se nova stavka menija ukoliko je korisnik koji poziva pogled u roli admin

Administratorski meni i prikaz rola



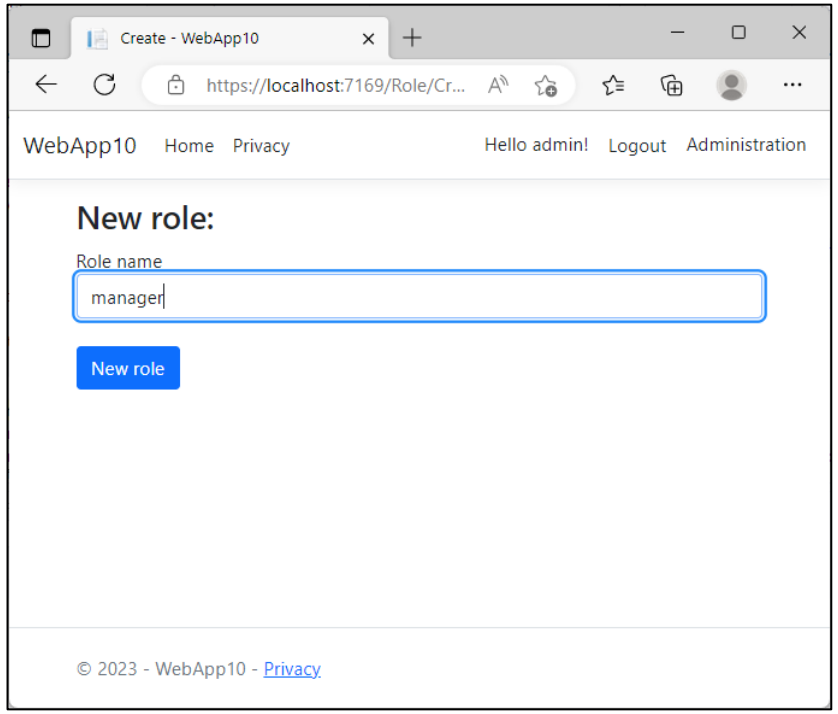
GET i POST metoda CREATE

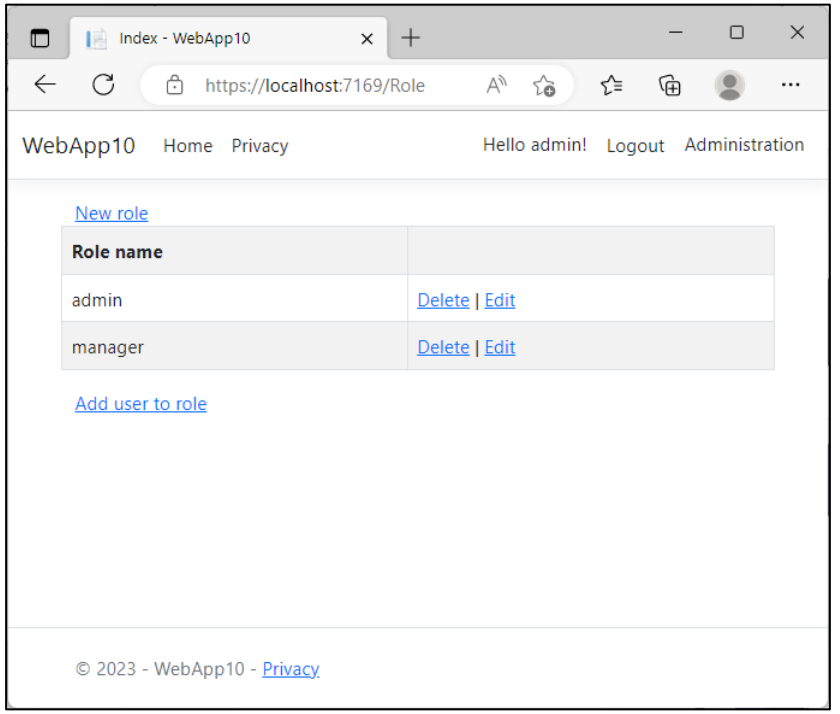
```
public ActionResult Create()
    {
        return View();
    }


    [HttpPost]
    [ValidateAntiForgeryToken]

    public async Task<IActionResult> Create(IdentityRole role)
    {
        var rezultat = await rm.CreateAsync(role);
        if (rezultat.Succeeded)
        {
            return RedirectToAction("Index");
        }
        else
        {
            return View(role);
        }
    }
```

Pogled Create

```
@model IdentityRole
@{
    ViewData["Title"] = "Create";
}
<h2>New role:</h2>
<form asp-action="Create">
    <div class="form-group">
        <label asp-for="Name">Role name</label>
        <input asp-for="Name" class="form-control" data-val="true" data-val-required="Enter
role name" />
        <span asp-validation-for="Name"></span>
    </div>
    <br />
    <button class="btn btn-primary" type="submit">New role</button>
</form>

@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

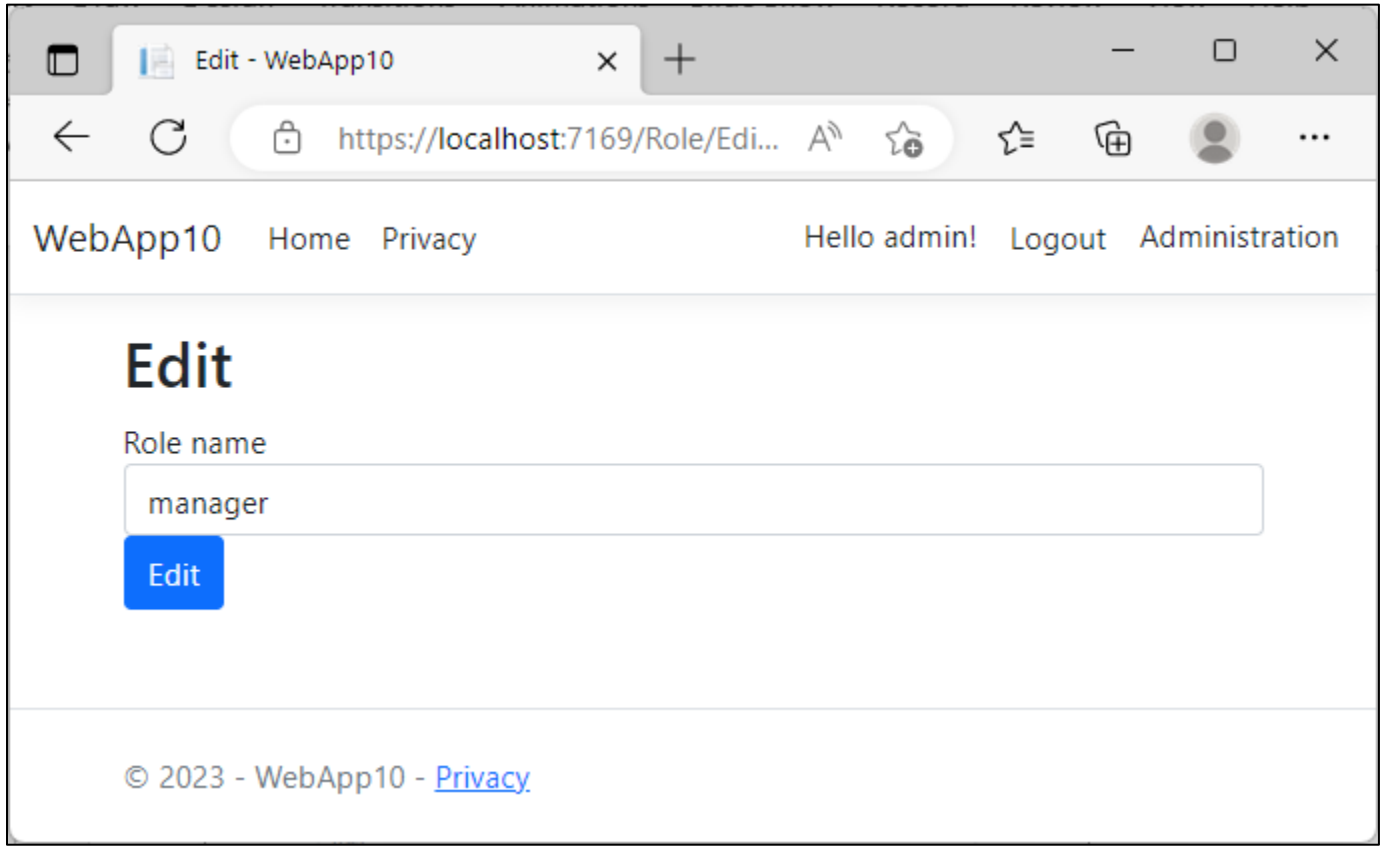
```
public async Task<ActionResult> Edit(string roleName)
{
    IdentityRole role = await rm.FindByNameAsync(roleName);

    return  View(role);
}
```

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Edit(IdentityRole role)
{
    IdentityRole r = await rm.FindByIdAsync(role.Id);
    try
    {

        r.Name = role.Name;
        await rm.UpdateAsync(r);
        return RedirectToAction("Index");
    }
    catch (Exception)
    {

        return View(r);
    }
}
```

Pogled za editovanje role

```
@model IdentityRole
@{
    ViewData["Title"] = "Edit";
}

<h1>Edit</h1>


<form asp-action="Edit">
    <div class="form-group">
        <input asp-for="Id" type="hidden" />
        <label asp-for="Name">Role name</label>
        <input asp-for="Name" class="form-control" data-val="true" data-val-required="Enter role name" />
        <span asp-validation-for="Name"></span>
    </div>
    <button class="btn btn-primary" type="submit">Edit</button>
</form>

@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



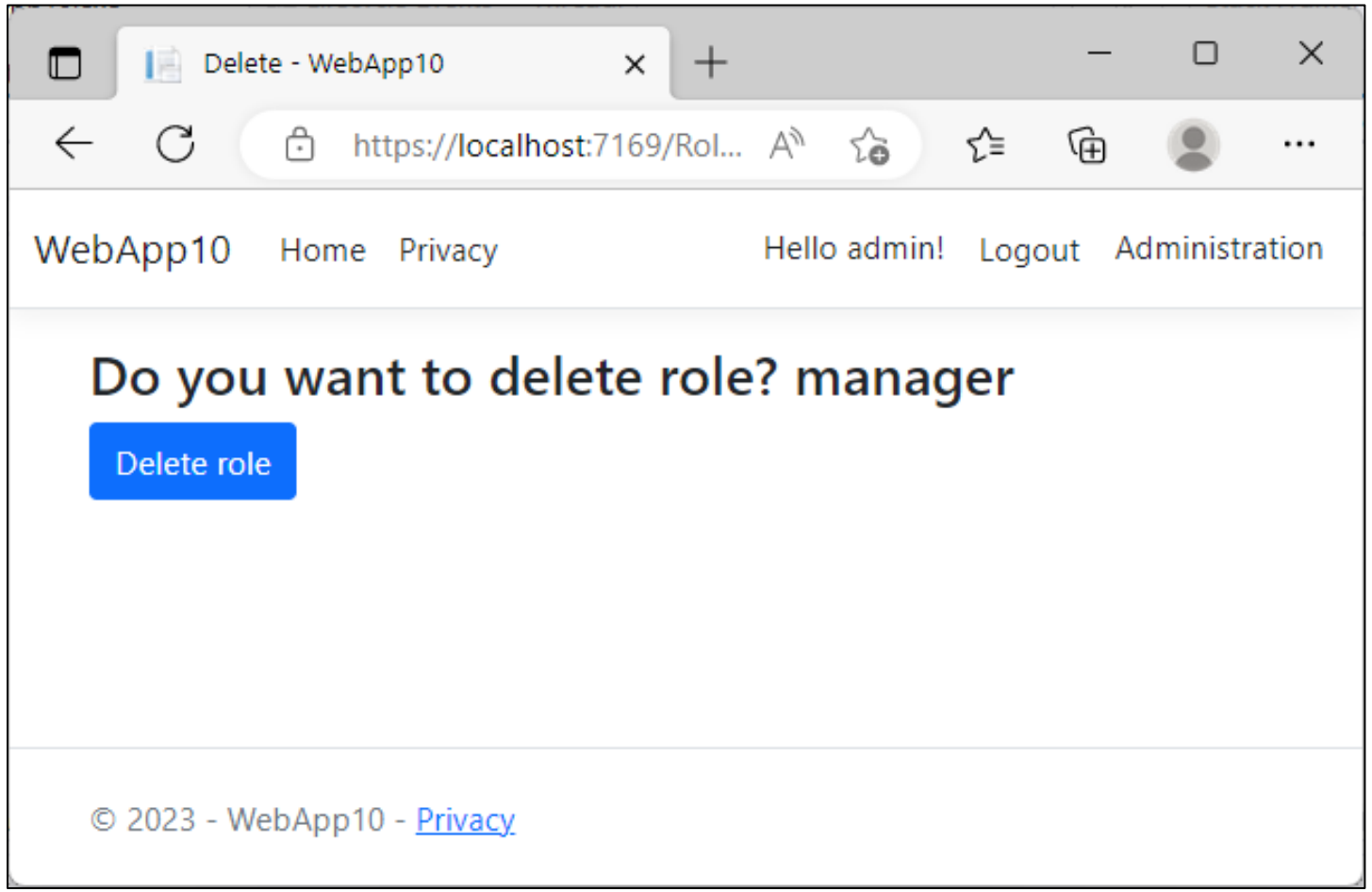
Metode za brisanje role

```
public async Task<IActionResult> Delete(string roleName)
{
    IdentityRole role = await rm.FindByNameAsync(roleName);

    return View(role);
}


[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(string roleName)
{
    IdentityRole role = await rm.FindByNameAsync(roleName);

    var rez = await rm.DeleteAsync(role);


    if (rez.Succeeded)
    {
        return RedirectToAction("Index");
    }

    return View(role);
}
```

Pogled za brisanje role

```
@model IdentityRole
@{
    ViewData["Title"] = "Delete";
}

<h2>Do you want to delete role? @Model?.Name</h2>

<div class="row">
    <form asp-action="Delete">
        <input type="hidden" name="roleName" value="@Model?.Name" />
        <button class="btn btn-primary">Delete role</button>
    </form>

</div>
```

Brisanje role



Metoda AddUserToRole - GET

```
public IActionResult AddUserToRole()
{
    ViewBag.Users = new SelectList(um.Users, "UserName", "UserName");
    ViewBag.Roles = new SelectList(rm.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="user" asp-items="ViewBag.Users" class="form-control">
```

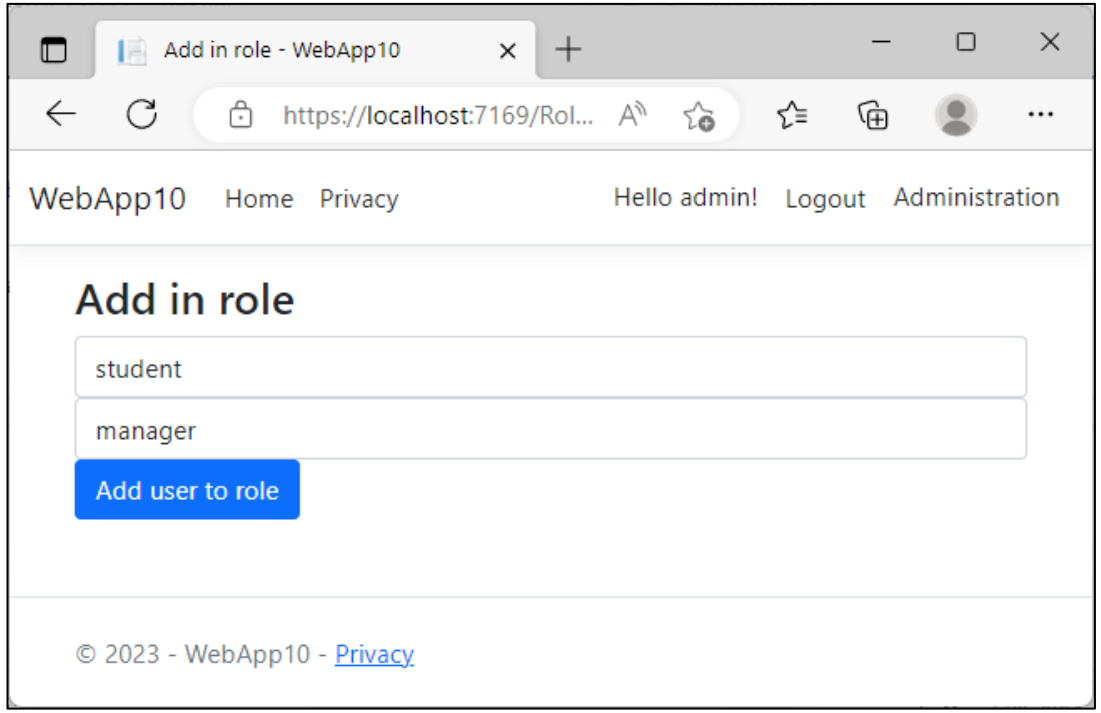
Metoda AddUserToRole - POST

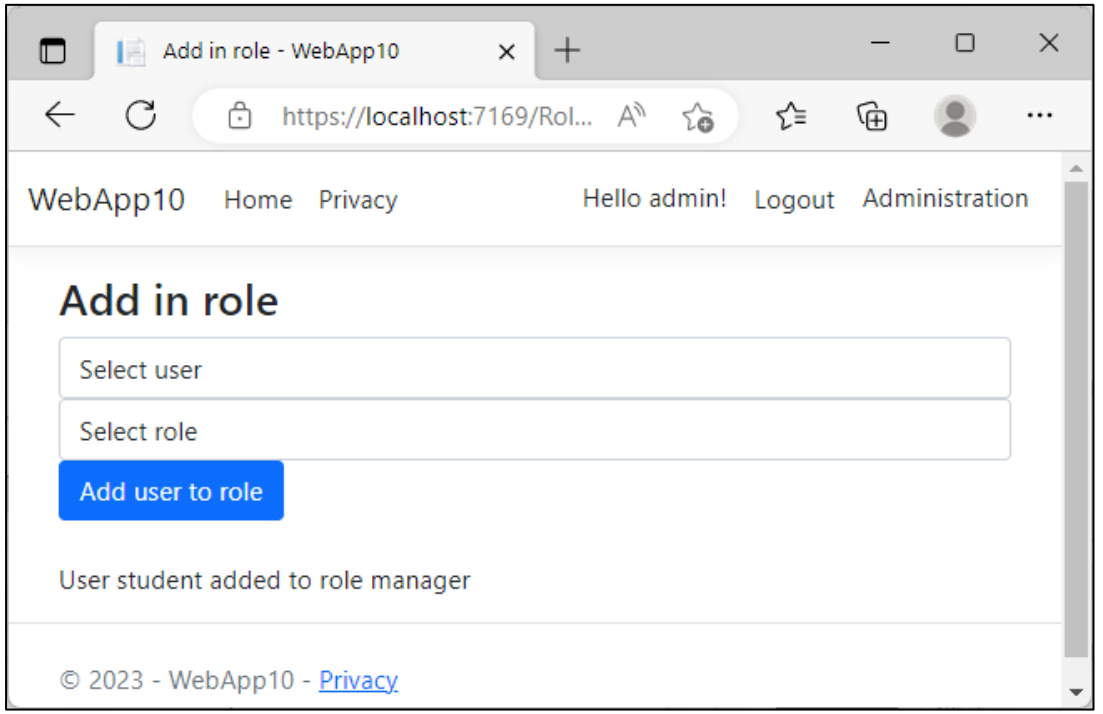
```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> AddUserToRole(string user, string role)
{
    ViewBag.Users = new SelectList(um.Users, "UserName", "UserName");
    ViewBag.Roles = new SelectList(rm.Roles, "Name", "Name");
    ApplicationUser au = await um.FindByNameAsync(user);

    bool rez1 = await um.IsInRoleAsync(au, role);

    if (rez1)
    {
        ViewBag.Message = $"User {user} is already in {role} role";
        return View();
    }

    var rez2 = await um.AddToRoleAsync(au, role);

    if (rez2.Succeeded)
    {
        ViewBag.Message = $"User {user} added to role {role}";
    }
    else
    {
        ViewBag.Poruka = "Error adding user in role";
    }

    return View();
}
```

AddUserToRole.cshtml

```
@{
    ViewData["Title"] = "Add in role";
}
<h2>Add in role</h2>
<div class="row">
    <div class="col-md-4">
        <form asp-action="AddUserToRole">
            <div class="form-group">
                <select name="user" asp-items="ViewBag.Users" class="form-control">
                    <option>Select user </option>
                </select>
            </div>
            <div class="form-group">
                <select name="role" asp-items="ViewBag.Roles" class="form-control">
                    <option>Select role</option>
                </select>
            </div>
            <button class="btn btn-primary">Add user to role</button>
        </form>
        <br />
        <span>@ViewBag.Message</span>
    </div>
</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

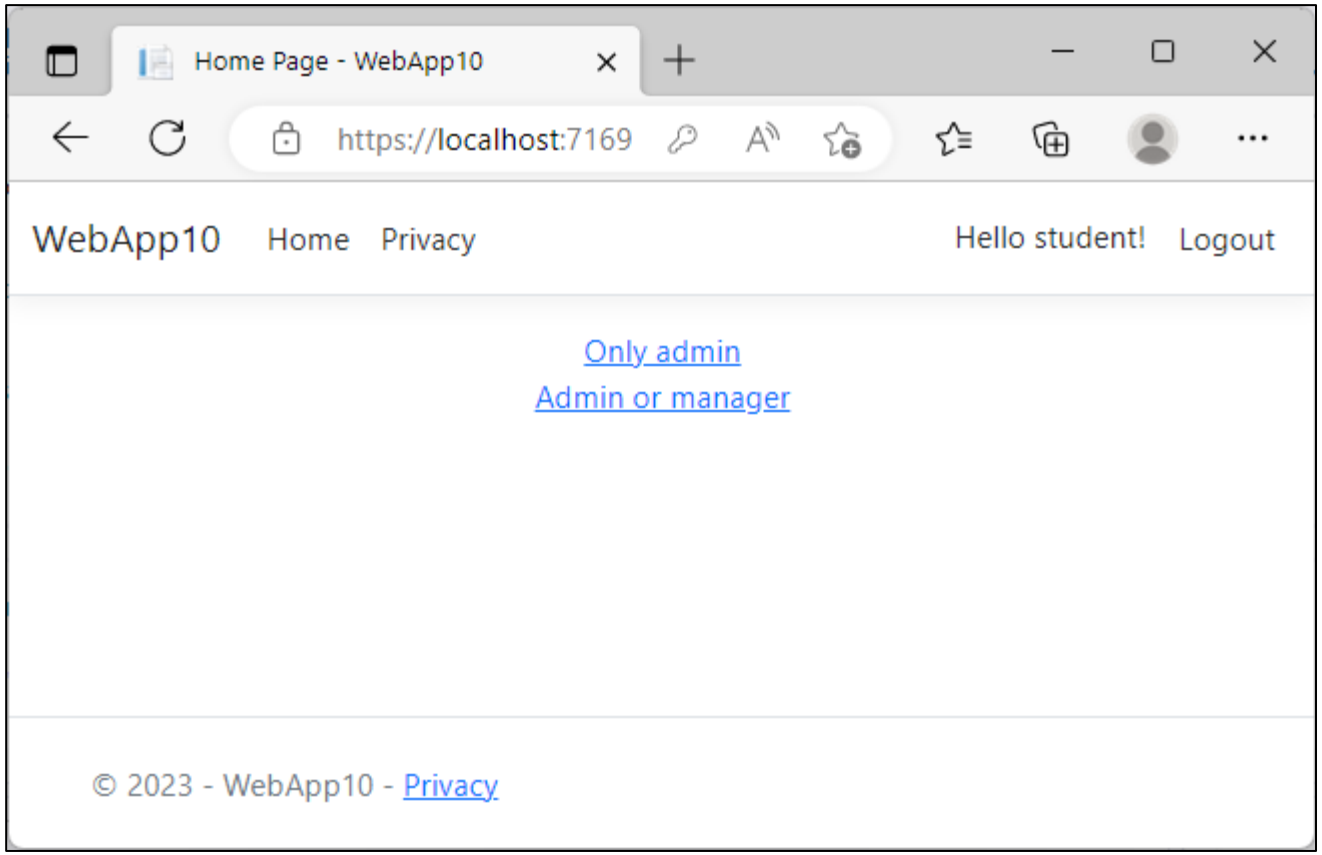
Metodu OnlyAdmin može da poziva samo korisnik u roli admin.
Metodu AdminOrManager može da poziva samo korisnik koji je u roli admin ili manager

```
@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>
```

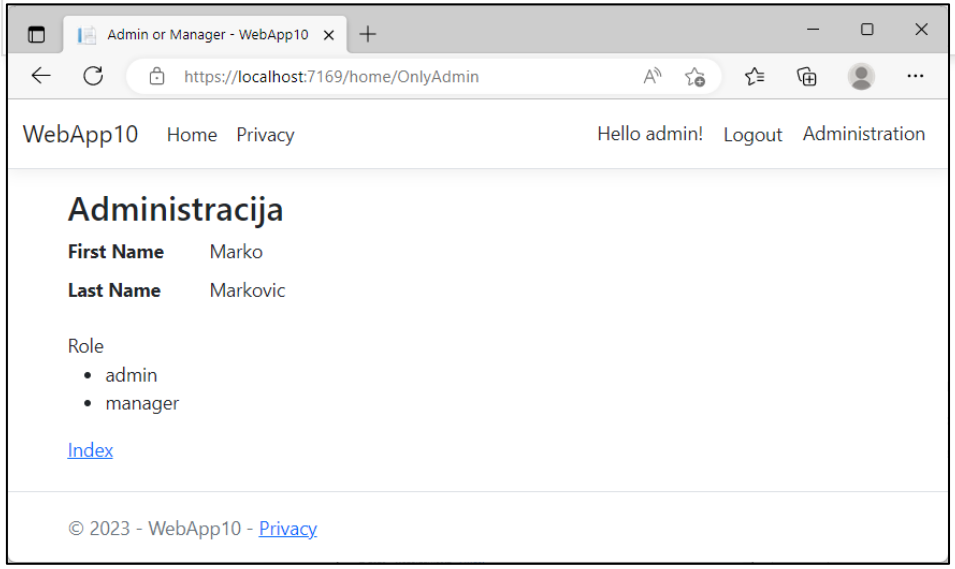
Pogled AdminOrManager.cshtml

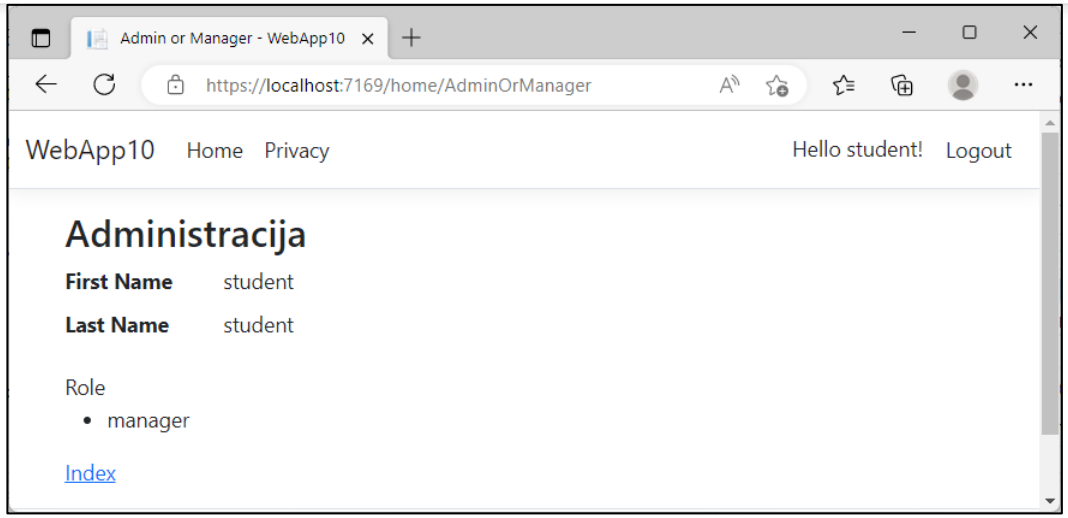
Svojstvo User unutar cshtml fajla daje informacije o logovanom korisniku

Početna strana aplikacije



Primer korisnika i njima dodeljenih uloga





Korisnik admin je u rolama admin i manager

Korisnik student je u roli manager

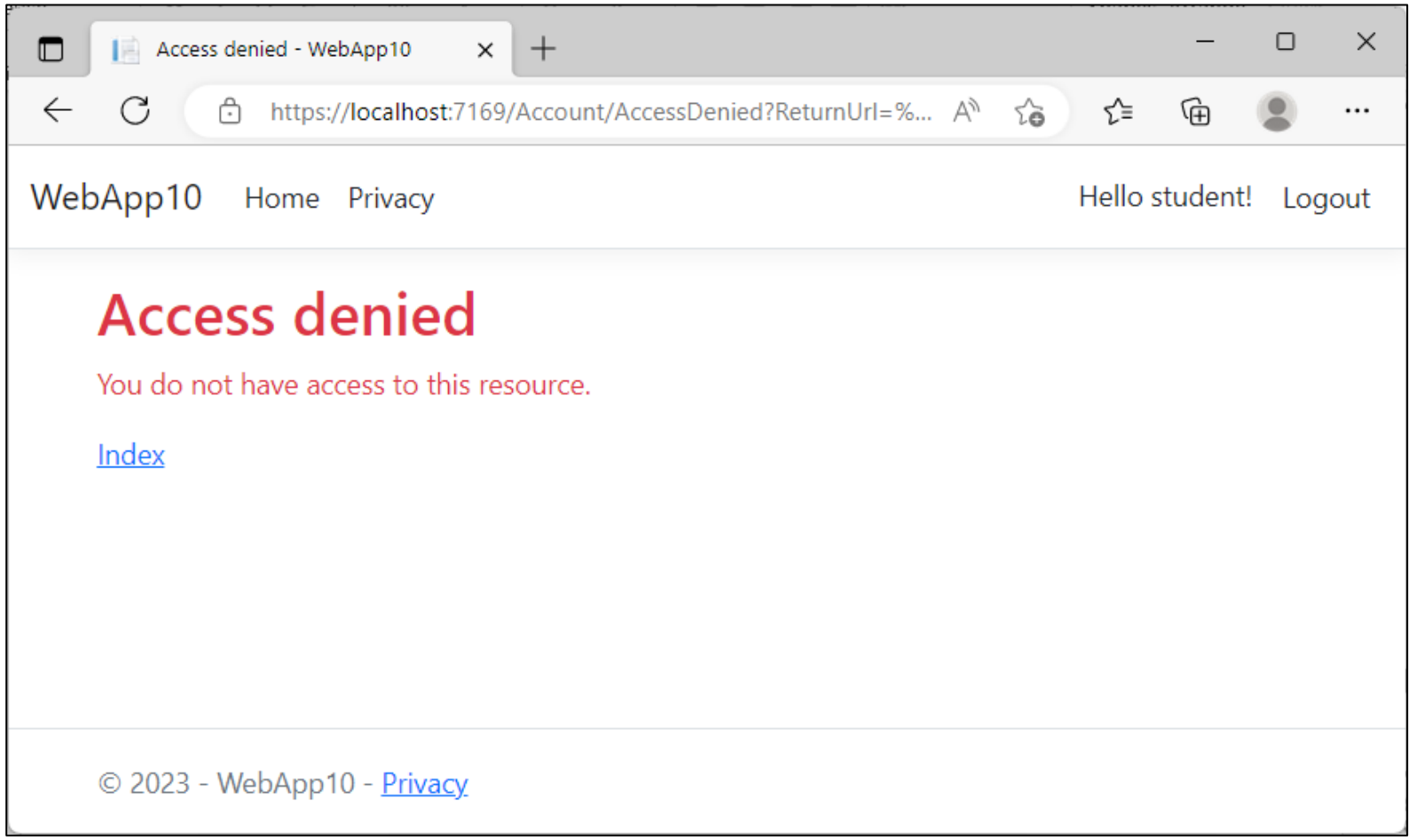
AccountController metoda AccessDenied

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

Pogled AccessDenied.cshtml

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

<header>
    <h1 class="text-danger">@ViewData["Title"]</h1>
    <p class="text-danger">You do not have access to this resource.</p>
</header>
<a asp-controller="Home" asp-action="Index">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 se svojstvo koristi pri definisanja autentifikacionog kolačića da bi se specificirala metoda koja se poziva kada korisnik pokuša da pozove zabranjenu metodu:

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