

Uniwersytet Bielsko-Bialski

LABORATORIUM

Programowanie dla Internetu w technologii ASP.NET

Sprawozdanie nr 8

System ról

Cel ćwiczenia

Celem ćwiczenia było stworzenie i przypisanie ról użytkownikom.

Wprowadzenie

Współczesne aplikacje internetowe często wymagają zarządzania różnorodnymi uprawnieniami użytkowników, aby zapewnić bezpieczeństwo oraz odpowiednie funkcjonowanie systemu. Jednym z kluczowych aspektów zarządzania uprawnieniami jest implementacja systemu ról. W systemie tym użytkownicy są przypisywani do określonych ról, które definiują ich prawa dostępu do różnych funkcjonalności aplikacji.

W ramach niniejszego ćwiczenia naszym celem było stworzenie i przypisanie ról użytkownikom w aplikacji internetowej opartej na technologii ASP.NET. Dzięki temu rozwiązaniu możemy kontrolować, które zasoby i operacje są dostępne dla poszczególnych użytkowników na podstawie przypisanych im ról.

Stworzenie ról

Dodanie ról do DbInitializera

```
var Roles = new IdentityRole[]
{
    new IdentityRole { Name = "Admin", NormalizedName = "ADMIN", Id = "1" },
    new IdentityRole { Name = "Manager", NormalizedName = "MANAGER", Id = "2"},
    new IdentityRole { Name = "Member", NormalizedName = "MEMBER", Id = "3"}
};
foreach (IdentityRole r in Roles)
{
    context.Roles.Add(r);
}
context.SaveChanges();
```

Efekt przy tworzeniu bazy danych :

	Id	Name	NormalizedNa...	ConcurrencySt...
▶	1	Admin	ADMIN	NULL
	2	Manager	MANAGER	NULL
	3	Member	MEMBER	NULL
⚙	NULL	NULL	NULL	NULL

Przypisanie ról użytkownikom :

	UserId	RoleId
▶	fa1-3bcd5c15eb8	1
	e4fc416e-b20f-...	1
	385a2b22-21c3...	3
⚙	NULL	NULL

Stworzenie widoków dla Admina do zarządzania rolami użytkowników:

```
@model List<Microsoft.AspNetCore.Identity.IdentityUser>

<h1>Users</h1>

<table class="table">
  <thead>
    <tr>
      <th>User Name</th>
      <th>Email</th>
      <th>Actions</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var user in Model)
    {
      <tr>
        <td>@user.UserName</td>
        <td>@user.Email</td>
        <td>
          <a href="@Url.Action("ManageRoles", new { userId = user.Id })"
class="btn btn-primary">Manage Roles</a>
        </td>
      </tr>
    }
  </tbody>
</table>
```

Jak to wygląda :

Wycieczki Rezerwacje Klienci Home ManageRoles

Hello kacper@gmail.com! Logout

Users

User Name	Email	Actions
Menager	menager@gmail.com	<button>Manage Roles</button>
Member	Member@gmail.com	<button>Manage Roles</button>
Admin	admin@gmail.com	<button>Manage Roles</button>
kacper@gmail.com	kacper@gmail.com	<button>Manage Roles</button>

© 2024 - Trins - [Privacy](#)

Kod dla samego "ManageRoles"

```
@model TripsS.ViewModel.ManageRolesViewModel

<h1>Manage Roles for @Model.UserId</h1>

<form asp-action="UpdateRoles" method="post">
  <input type="hidden" name="userId" value="@Model.UserId" />
  <div class="form-group">
    @foreach (var role in Model.AllRoles)
    {
      <div class="form-check">
        <input class="form-check-input" type="checkbox" name="roles"
value="@role.Name" id="role_@role.Name" @(Model.UserRoles.Contains(role.Name) ?
"checked" : "")>
        <label class="form-check-label"
for="role_@role.Name">@role.Name</label>
      </div>
    }
  </div>
  <button type="submit" class="btn btn-primary">Update Roles</button>
</form>
```

Sam widok:

Manage Roles for e4fc416e-b20f-4628-a1d9-5690f04621ec

☒ Admin

☐ Manager

☐ Member

Update Roles

© 2024 - Trips - [Privacy](#)

Przykładowe wykorzystanie roli:

WycieczkiRezerwacjeKlienciHome

Hello admin123@gmail.com!Logout

Witaj w aplikacji Wycieczki!

Lista Studentów: [Pokaż szczegóły](#)

Jan Kowalski

Anna Nowak

Piotr Kowalczyk

Jak widać użytkownik admin123 nie widzi nawet zarządzania użytkownikami, a spowodowane jest to następującym kodem :

```
@if (User.IsInRole("Admin"))
{
    <a class="navbar-brand" asp-area="" asp-controller="Admin" asp-
action="Index">ManageRoles</a>
}
```

Teraz przypiszemy temu użytkownikowi rolę "Manager"

admin123@gmail.com

admin123@gmail.com

Manage Roles

Manage Roles for ea940d85-6f24-4b04-bc61-a04a4005976f

☐ Admin

☒ Manager

☐ Member

Update Roles

Dzięki tej roli ma dostęp do np. zarządzania klientami :

Index

[Create New](#)

FirstName	LastName	Pesel	Email	Phone
Jan	Kowalski	123456789	askdlaskd@gmail.com	Edit Details Delete
Anna	Nowak	987654321	aSDASDAS@gmail.com	Edit Details Delete
Piotr	Kowalczyk	123123123	kakaka@gmail.com	Edit Details Delete

Dzieje się to dzięki :

```
[Authorize(Roles = "Manager,Admin")]
public class ClientsController : Controller
```

Do tej klasy i jej metod mają dostęp tylko Manager i Admin

Manager może tworzyć i edytować rezerwacje:

```
[Authorize(Roles = "Manager,Admin")]
public async Task<IActionResult> Edit(Guid id,
[Bind("IdReservation,IdClient,IdTrip,AmountOfPeople,ReservationDate,Status")]
ReservationViewModel reservationViewModel)
```

```
[Authorize(Roles = "Manager,Admin")]
public async Task<IActionResult>
Create([Bind("IdReservation,IdClient,IdTrip,AmountOfPeople,ReservationDate,Status"
)] ReservationViewModel reservationViewModel)
```

Create

Reservation

IdClient

IdTrip

AmountOfPeople

ReservationDate

dd.mm.yyyy --:--

Status

Create

Back to List

Edit

Reservation

IdClient

1

IdTrip

2

AmountOfPeople

3

ReservationDate

25.05.2024 10:46

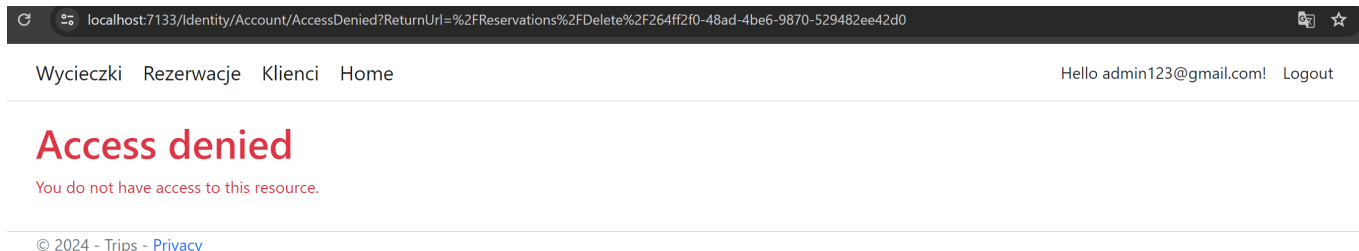
Status

Save

Back to List

Ale nie może ich usuwać :

```
[Authorize(Roles = "Admin")]  
public async Task<IActionResult> Delete(Guid? id)
```



Wnioski

Dzięki temu ćwiczeniu dowiedzieliśmy się jak stworzyć role i przypisać je do użytkowników, a także jak wykorzystać role w naszym projekcie. Dzięki temu możemy przydzielać różne uprawnienia różnym użytkownikom i w zależności od tego wyświetlać im różne widoki.