Proiect Tehnologii Moderne – To Do List

----------PROIECTAREA APLICAȚIEI ȘI USER CASE---------

Am crea o aplicație de tip ToDo List în care utilizatoul își poate insera lucrurile pe care le are de îndeplinim. În plus, am adăugat un o tab de căutare în funcție de locația la care se desfășoară obiectivul respectiv. Toate acestea sunt disponibile doar utilizatorilor înregistrați, având diferite funcționalități, după rol, după cum urmează:

Utilizator neînregistrat:

- Acces la pagina Home;
- Posibilitate de înregistrare, respectiv logare pe site;
- Nu are acces la crearea unui ToDo list până nu se înregistrează/loghează;

Utilizator înregistrat:

- acces la crearea propriului ToDo List cu toate funcționalitățile;

- un tab suplimentare, care apare doar celor înregistrați și logați “Căutare”;

Administrator:

- 2 tab suplimentare, care apare doar celor înregistrați și logați “Căutare”, și “Secret Page”, care pagină fiind disponibilă și vizibilă doar celor cu rolul de Admin;

- ștergerea sarcinilor, vizualizare detalii sarcină;

--------PROIECTAREA ȘI IMPLEMENTAREA BAZEI DE DATE--------

Crearea tabelului pentru ToDo List se va face utilizând Code first approach prin definirea unei clase în folderul Models

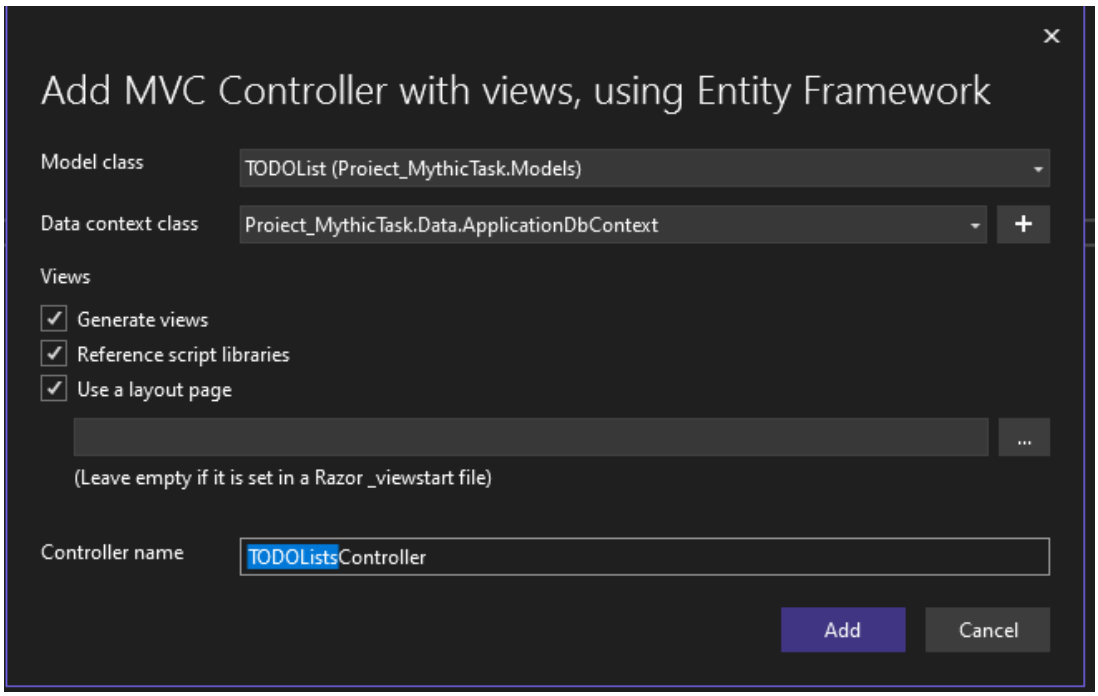
Models -> Class -> Denumim “TODOList” cu următoarele proprietăți:

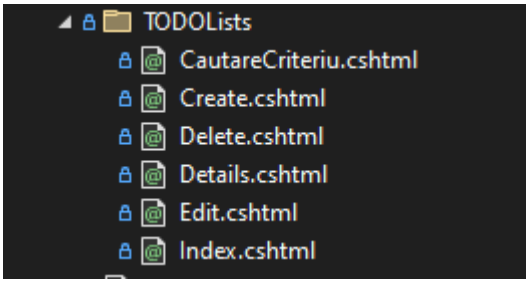
```
namespace Proiect_MythicTask.Models
{
    19 references
    public class TODOList
    {
        [Key]
        11 references
        public int NrCrit { get; set; }
        [Required]
        12 references
        public string Obiectiv { get; set;}

        12 references
        public string Descriere { get; set;}

        13 references
        public string Locație { get; set;}

        12 references
        public DateTime Deadline { get; set; } = DateTime.Now;
```

Denumim “NrCrit” cu atributul **[Key]** pentru a specifica că este cheie primară în baza noastră de date și “Obiectiv” cu **[Required]** pentru a specifica că acea secțiune trebuie să aibă o valoare. Am folosit DateTime pentru “Deadline” pentru a putea selecta manual data la care avem de îndeplinit task-ul respectiv, chiar dacă DateTime.Now este folosit pentru a seta timpul curent.

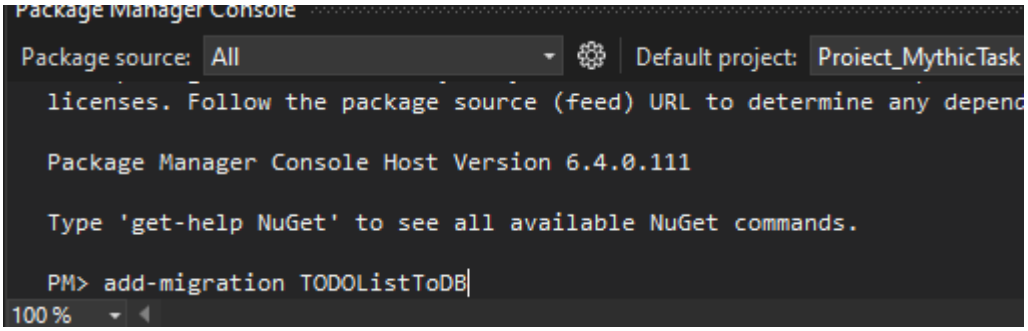
Pentru a crea pagina respectivă pentru TODOList, creem un controler: Controllers -> Add Controller -> MVC Controller with Views, using Entity Framework.

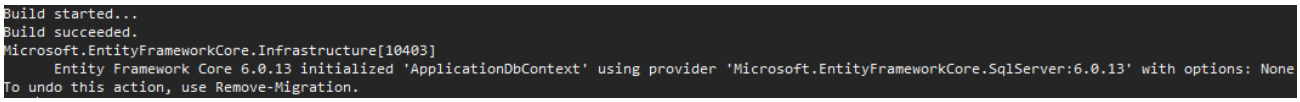
Selectăm la clasa modelului TODOList creat mai sus, iar la data context schimbăm string-ul default cu ApplicationDbContext, după cum urmează:

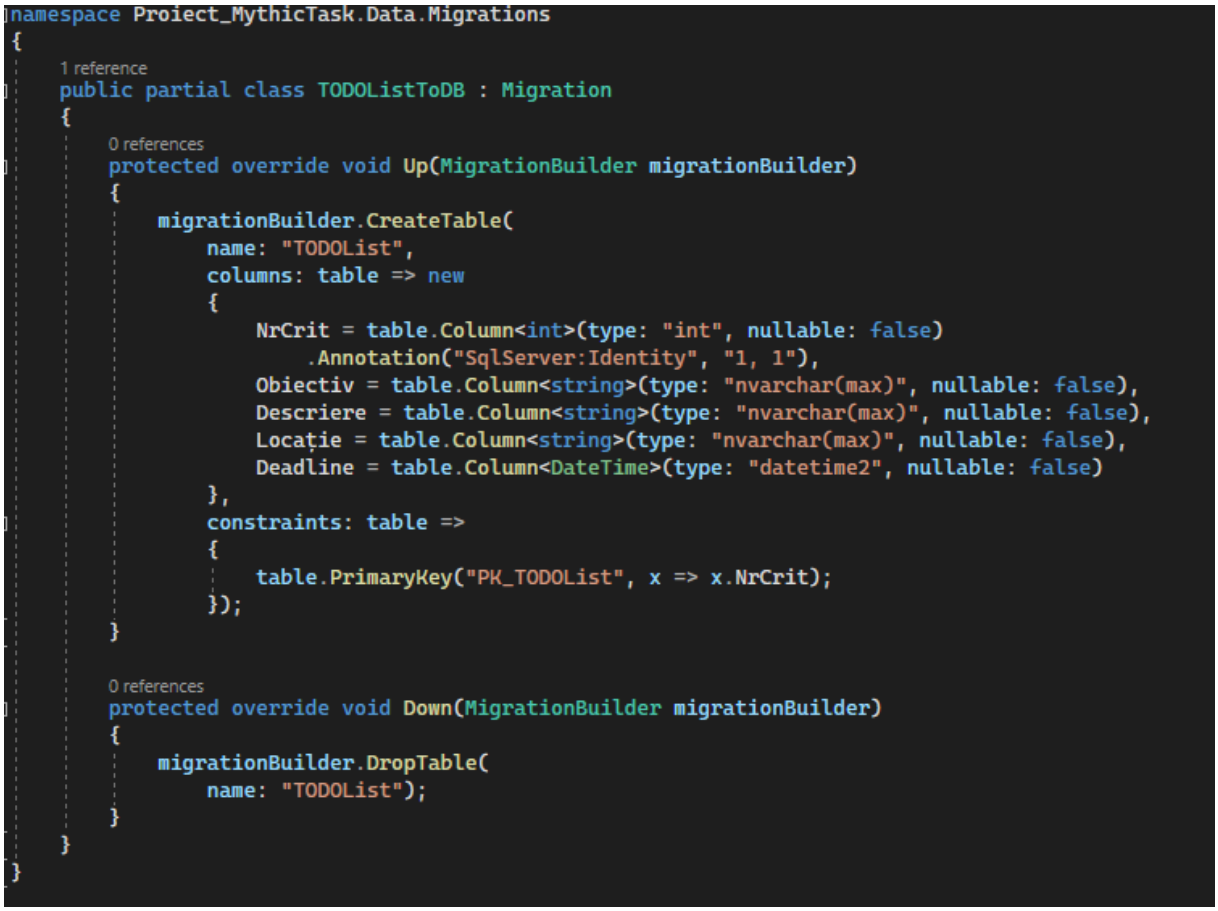




Pentru a stoca și insera datele, va să schimbăm diagrama bazei de date, astfel încât să corespundă cu proprietățile din în clasa TODOList. Pentru asta vom folosit Add-migration pe care o vom numi TODOListToDB.







```
namespace Proiect_MythicTask.Data.Migrations
{
    1 reference
    public partial class TODOListToDB : Migration
    {
        0 references
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "TODOList",
                columns: table => new
                {
                    NrCrit = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    Obiectiv = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    Descriere = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    Locație = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    Deadline = table.Column<DateTime>(type: "datetime2", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_TODOList", x => x.NrCrit);
                });
        }

        0 references
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "TODOList");
        }
    }
}
```

După ce am verifica că totul este ok, folosim comanda update-database pentru a crea tabelule; unul pentru ToDo list, iar celălalte pentru utilizatori fiindcă, la crearea proiectul, am ales, la “Authentification Type’ -> Individual Account pentru a putea folosi funcția de înregistrare.

Putea afișarea listei, ca un item în bara de navigare, pe care îl putem selecta mergem în Views -> Shared -> _Layout.cshtml și adăugăm un nou item. La „asp-controller” folosim cel creat mai sus TODOLists, iar la asp-action vom utiliza “Index”, care ne va oferi posibilitatea de a vizualiza toate task-urile.

```
<li class="nav-item">
<a class="nav-link text-dark" asp-area="" asp-controller="TODOLists" asp-
action="Index">ToDo</a>
</li>
```

Dacă mergem în controller TODOLists, avem următoarea metodă

```
// GET: TODOLists
        public async Task<IActionResult> Index()
        {
              return View(await _context.TODOList.ToListAsync());
        }
```

care ne va oferi posibilitatea de a vizualiza toate task-urile, în cazul de față **Index().**

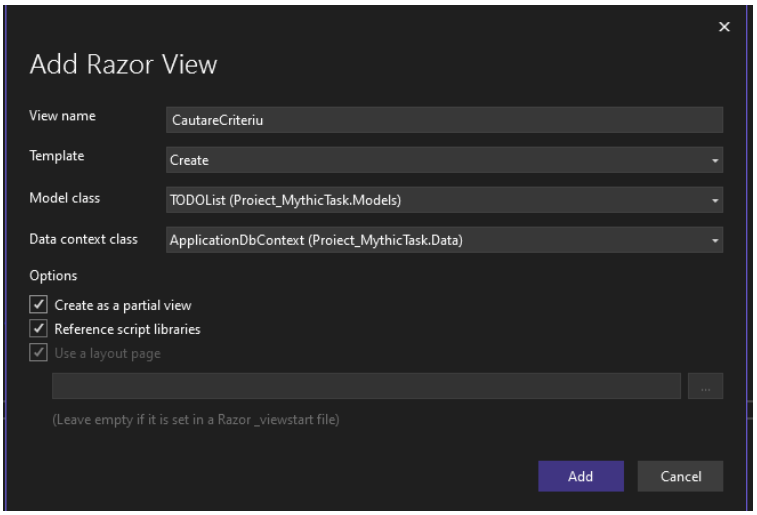
----------CREAREA TAB-ULUI “Căutare”---------

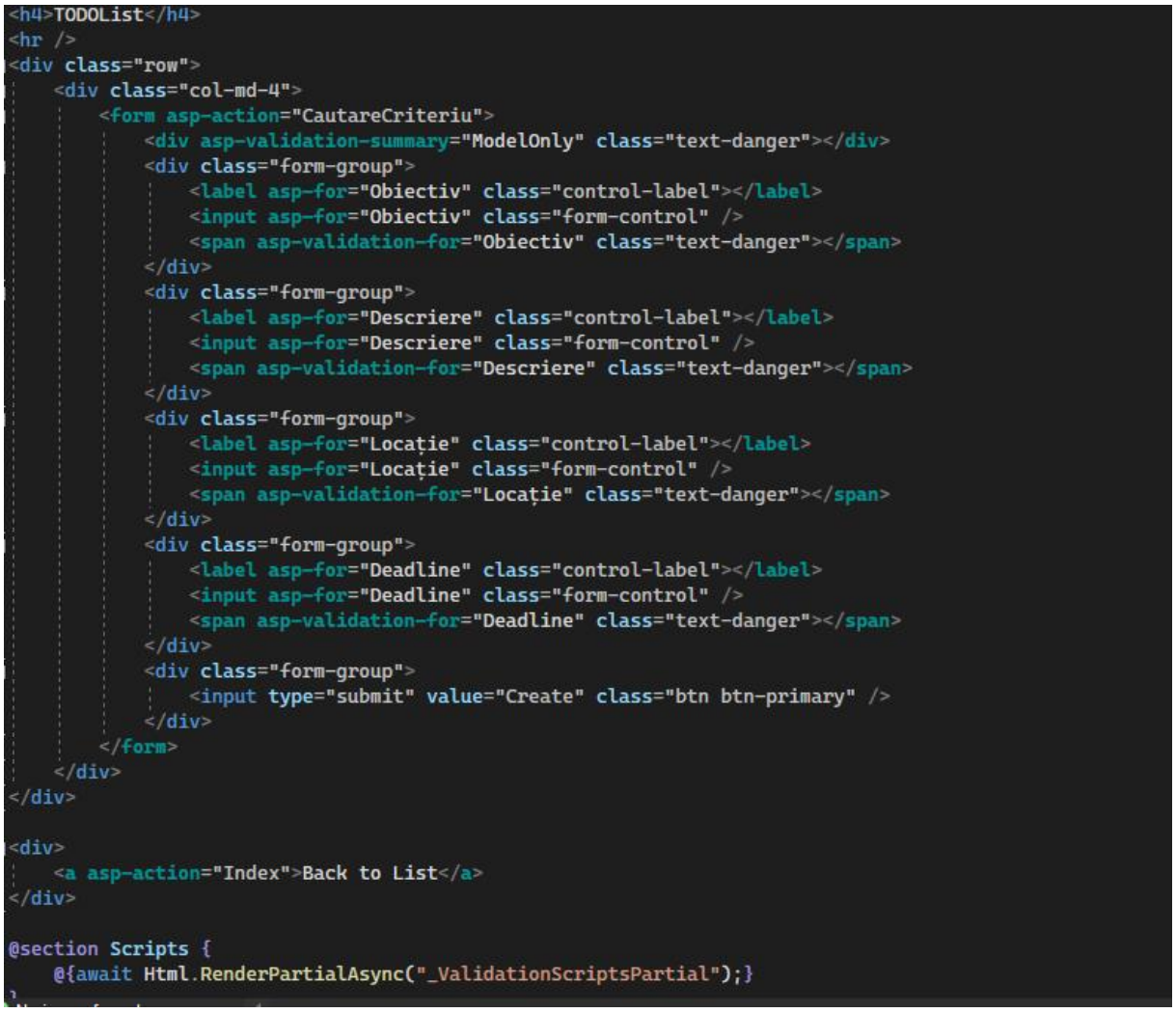
În crearea tab-ului **“Căutare”** vom proceda în felul următor:

- Creăm o metoda în controlerrul TODOLists numită `CautareCriteriu`, care returnează un form atunci când utilizatorul accesează link-ul respectiv

```
// GET: TODOLists // CautareCriteriu
public async Task<IActionResult> CautareCriteriu()
{
    return View("CautareCriteriu");
}
```

- Click dreapta pe `CautareCriteriu()` -> AddViews -> RazorView -> selectam Create Template (deoarece dorim să un form în care să inserăm termenul după care dorim să căutăm) -> selectăm TODOLists la Model Class (ca baza de creare a formului) -> bifăm Create as aPartial View deoarece vreau să facă parte din template-ul selectat.





```
<h4>TODOList</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="CautareCriteriu">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Obiectiv" class="control-label"></label>
                <input asp-for="Obiectiv" class="form-control" />
                <span asp-validation-for="Obiectiv" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Descriere" class="control-label"></label>
                <input asp-for="Descriere" class="form-control" />
                <span asp-validation-for="Descriere" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Locație" class="control-label"></label>
                <input asp-for="Locație" class="form-control" />
                <span asp-validation-for="Locație" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Deadline" class="control-label"></label>
                <input asp-for="Deadline" class="form-control" />
                <span asp-validation-for="Deadline" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

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

Form-ul este funcțional, dar necesită îmbunătățiri adiționale. În cazul vreau să fac căutarea după “Locație”, așa că voi strege “Obiectiv”, ”Descriere” și “Deadline”.

- Ștergem următoarele linii de cod deoarece nu dorim să creăm un nou task, respectiv validare.

```
@model Proiect_MythicTask.Models.TODOList

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

- Modificăm label, input-ul din form-group și asp-validation-for, secțiunea “Locație”, fiindcă nu dorim să asociem form-ul cu “Locație” cu

`CautareDupaLocatie`. Așadar, în vom transforma într-un basic HTML form. Totodată cu ștergerea `@modeProiect_MythicTask.Models.TODOList` acestea sunt irelevante; nu avem un ToDo list nou aici.

- Adăugăm mici modificări, cum ar fi denumirea titlului și a butoanelor.

În final, avem următorul form:

```
<h4>Caută un obiectiv după locație</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="RezultateCautareCriteriu">

            <div class="form-group">
                <label for="CautareDupaLocatie" class="control-label"></label>
                <input name="CautareDupaLocatie" class="form-control" />
            </div>
            <div class="form-group">
                <input type="submit" value="Cautare" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Înapoi la listă</a>
</div>
```

Odată cu crearea `CautareDupaLocatie`, va trebui să mergem în folderul Controller -> TODOLists pentru a crea acea metodă și de a o afișa în pagină.

```
// POST: TODOLists // RezultateCautareCriteriu
        public async Task<IActionResult> RezultateCautareCriteriu(string
CautareDupaLocatie)
        {
            return View("Index", await _context.TODOList.Where(j =>
j.Locație.Contains(CautareDupaLocatie)).ToListAsync());
        }
```

Metoda `RezultateCautareCriteriu` i-a un parametru `CautareDupaLocatie`. Metoda returnează „Index” și o listă a elementelor inserate in tab-ul TODO. Se generează articolele

TODO Lista în care proprietatea „Locație” conține valoarea parametrului „CautareDupaLocatie” prin interogarea bazei de date.

- În final, Creăm, în bara de navigare, tab-ului Căutare, cu acțiunea `CautareCriteriu`
-

```
<li class="nav-item">
<a class="nav-link text-dark" asp-area="" asp-controller="TODOLists"
asp-action="CautareCriteriu">Căutare</a>
</li>
```

----------AUTORIZAREA, AUTENTIFICAREA ȘI CREAREA ROLURILOR----------

La crearea proiectului opțiunea de autentificare Individual User Accounts pentru a beneficia de framework-ul de autorizare si autentificare oferit de MVC.

Ne asigurăm că ApplicationDbContext moștenește IdentityDbContext

```
namespace Proiect_MythicTask.Data
{
    9 references
    public class ApplicationDbContext : IdentityDbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }
        12 references
        public DbSet<Proiect_MythicTask.Models.TODOList> TODOList { get; set; }
    }
}
```

Apăsăm pe buton register și vom înregistra 2 utilizatori: student și admin, care vor avea rolurile de utilizator, respectiv administrator.

Dupa înregistrare, navigăm în baza de date, și observăm că s-au generat noi înregistrări în tabela AspNetUsers. Dacă vom da click dreapta pe tabela AspnetUsers > Show Table Data vom vedea stocat în db noi utilizatori

	Id	UserName	NormalizedUs...	Email
▶	ae03-a66b2134f4cf	admin@admin....	ADMIN@ADMI...	admin@admin....
	c767e5b8-0a70-...	student@stude...	STUDENT@STU...	student@stude...
	NULL	*NULL*	*NULL*	*NULL*

Deschidem AspNetRoles pentru a crea rolurile.

	Id	Name	NormalizedNa...	ConcurrencySt...
▷	1	Administrator	Admin	*NULL*
	2	Utilizator	User	*NULL*
◆	*NULL*	*NULL*	*NULL*	*NULL*

Deschidem AspNetUserRoles pentru asignarea RoleID cu UserId-urile înregistrate. În cazul de față, Username-ul admin de mai sus, are ID-ul b62dabc6-e8c0-4bc2-ae03-a66b2134f4cf, are asignat RoleID 1, care are numele de Administrator. În funcţie de rolul pe care îl are utilizatorul logat vom afişa în view funcţionalităţile la care are acesta are acces.

	UserId	RoleId
▷	ae03-a66b2134f4cf	1
	c767e5b8-0a70-...	2
◆	*NULL*	*NULL*

Pentru ca rolurile să funționeze cum trebuie, mergem în Program.cs și adăugam „AddRoles” pentru a adăuga suport pentru autorizarea bazată pe roluri în aplicație. Metoda i-a tipul clasei de rol, care este „IdentityRole”.

```
builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>();
```

---------EXEMPLE DE AUTORIZARE---------

Implementarea autorizarea la nivel de controller cu Authorize:

```
// GET: TODOLists
[Authorize]
3 references
public async Task<IActionResult> Index()
{
    return View(await _context.TODOList.ToListAsync());
}
```

Implementarea autorizarea la nivel de controller în funcție de rolul utilizătorului:

```
// GET: TODOLists/Create
[Authorize(Roles = "Utilizator")]
0 references
public IActionResult Create()
{
    return View();
}
```

```
// GET: TODOLists/Details/5
[Authorize(Roles = "Administrator, Utilizator")]
0 references
public async Task<IActionResult> Details(int? id)
{
    if (id == null || _context.TODOList == null)
    {
        return NotFound();
    }

    var tODOList = await _context.TODOList
        .FirstOrDefaultAsync(m => m.NrCrit == id);
    if (tODOList == null)
    {
        return NotFound();
    }

    return View(tODOList);
}
```

```
// GET: TODOLists/Delete/5
[Authorize(Roles = "Administrator")]
0 references
public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.TODOList == null)
    {
        return NotFound();
    }

    var tODOList = await _context.TODOList
        .FirstOrDefaultAsync(m => m.NrCrit == id);
    if (tODOList == null)
    {
        return NotFound();
    }

    return View(tODOList);
}
```

Ascunderea tab-ului Căutare în funcție dacă utilizatorul este logat:

```
@if(User.Identity.IsAuthenticated){

<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="TODOLists" asp-action="CautareCriteriu">Căutare</a>
</li>

}
```

Ascunderea tab-ului SECRET PAGE în funcție dacă utilizatorul are rolul de admin:

```
@if (User.IsInRole("Administrator"))
{


<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Secret Page</a>
</li>


}
```

----------CUSTOMIZARE----------

În această secțiune am să prezint alegerile stilsitice personale

1. Ștergerea tab-ului de Copyright din partea de jos a paginii -> ștergem secțiunea footer din Views -> Shared ->_Layout.cshtml

```
<footer class="border-top footer text-muted">
    <div class="container">
        &copy; 2023 - Proiect_MythicTask - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
    </div>
</footer>
<script src="~/lib/jquery/dist/jquery.min.js"></script>
```

2. Ascunderea proprietății “Descriere” atunci când accesăm tab-ul TODO

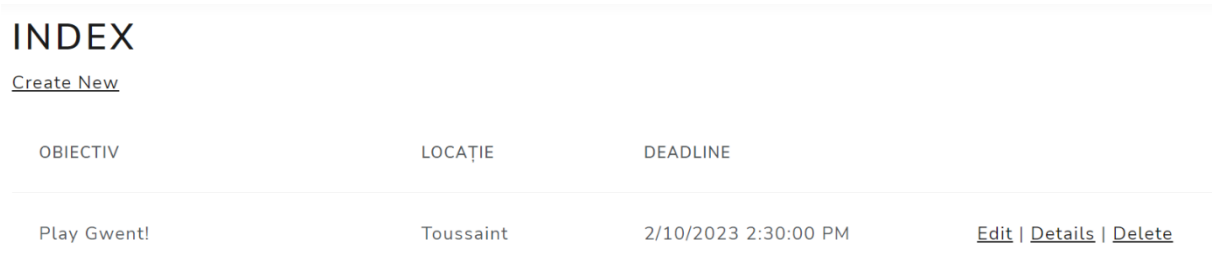
INDEX

Create New

OBIECTIV	DESCRIERE	LOCAȚIE	DEADLINE	
Play Gwent!	While travelling in Toussaint, I learned of the existence of a new deck. It was inspired by the Skellige Isles and their inhabitants.	Toussaint	2/10/2023 2:30:00 PM	Edit | Details | Delete

Mergem în view -> TODOLists -> Index.cshtml și stergem, în clasa table și @foreach:

```
<td>
@Html.DisplayFor(modelItem => item.Descriere)
</td>
```



3. Adăugare Bootswach theme

Mergem pe https://bootswatch.com/, alegem tema dorită și o descărcăm în format bootstrap.css. Copiem conținutul -> mergem în proiect -> wwwroot -> click dreapta pe css -> new styleshee -> copiem conținutul în foaia nou creată.

Ca utilizăm tema, mergem în Views -> Shared ->_Layout.cshtml și adăugăm următoarea linie de cod în secțunea head: `<link rel="stylesheet" href="~/css/StyleSheet.css" />`

4. Adăugare paginii SECRET PAGE

View -> Home -> Modificăm fișierul deja existent Privacy.cshtml.