C#/.NET

**creare il db, se il db è gia creato o comunque nn si dispone delle autorizzazioni ecco come fare**

**capire come si chiama la nostra utenza:**

USE [nomedb];

GO

SELECT name AS DatabaseName, suser\_sname(owner\_sid) AS OwnerName

FROM sys.databases

WHERE name = 'nome db';

GO

**CAMBIARE LE AUTORIZZAZIONI:**

USE [nomedb];

GO

ALTER AUTHORIZATION ON DATABASE::[nomedb] TO [nomeutenza];

GO

**ADESSO SI PASSA A C# CON .NET 8**

**-CREARE IL MODELLO app web asp.net core(Model-View-Controller)**

**NELLA CREAZIONE DEL MODELLO RICORDARSI DI FARE LE SEGUENTI COSE:**

1. **mettere il tipo di autenticazione: account individuali**
2. **mettere la spunta su configura https e non usare istruzioni di primo livello(molto probabilmente ci sono gia)**

SCARICARE I PACCHETTI NUGGET, CERCARE NEI PACCHETTI MICROSOFT ENTITY FRAMEWORK CORE E SCARICARE:

-Microsoft.EntityFrameworkCore.sqlserver

-Microsoft.EntityFrameworkCore.tools

-Microsoft.EntityFrameworkCore.design

-Microsoft.EntityFrameworkCore

-Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore

-Microsoft.AspNetCore.Identity.EntityFrameworkCore

-Microsoft.AspNetCore.Identity.UI

-Microsoft.AspNetCore.Mvc

**si possono anche scaricare tramite console dei pacchetti nuget:**

Install-Package Microsoft.EntityFrameworkCore.SqlServer

Install-Package Microsoft.EntityFrameworkCore.Tools

Install-Package Microsoft.EntityFrameworkCore.Design

Install-Package Microsoft.EntityFrameworkCore

Install-Package Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore

Install-Package Microsoft.AspNetCore.Identity.EntityFrameworkCore

Install-Package Microsoft.AspNetCore.Identity.UI

**METTERE IL FOOTER CON NOME E COGNOME:**

**inserire questa parte sotto il title del file layout.cshtml:**

**<style>**

**.footer-link {**

**margin-left: 20px;**

**margin-right: 20px;**

**}**

**.separator {**

**margin-left: 20px;**

**margin-right: 20px;**

**}**

**.footer-text {**

**margin-left: 20px;**

**}**

**</style>**

**INSERIRE QUESTA PARTE NEL FOOTER:**

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2024 - BikeStores -

<a class="footer-link" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

<span class="separator">|</span>

<span class="footer-text">Realizzato da Lele Adani, Corso: Faccccccciamo calcio</span>

</div>

</footer>

**ORA SI PASSA A FARE LO SCAFFOLD DEL DB APRIRE LA CONSOLE DEI PACCHETTI NUGGETS E INSERIRE QUESTA STRINGA DI CONNESSIONE:**

scaffold-dbcontext "Server=localhost;Database=nomeDB;Trusted\_Connection=True;"Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

**problema che alcune volte si connette altre volte no bisogna capire in base alle autorizzazioni che si hanno sul db se ci da un problema con SSl allora usare questa stringa:**

Scaffold-DbContext "Server=localhost;Database=Nomedb;Trusted\_Connection=True;TrustServerCertificate=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

**(CONTROLLARE IL SERVER DEL DB SE NON FUNZIONA)**

**N.B nel Models controllare che tutti i nomi delle classi siano al singolare tranne il context, se da errore andare nel program e cambiare il namespace**

**ADESSO BISOGNA PASSARE A CREARE IL/I CONTROLLER**

**SE NON è PRESENTE LA CARTELLA Controllers crearla e poi creare un nuovo controller che usa mvc con entity framework**

**//VEDERE DI METTERE DEGLI ESEMPI DI RICERCA/ FILTRO PER I CONTROLLER**

**PER METTERE UN LINK CHE RICHIAMA UNA ALTRO INDEX APPARTENENTE A UN'ALTRA VISTA:**

**<a asp-controller="Stores" asp-action="Index" asp-route-id="@item.ProductId">Disponibilità in negozio</a>**

**NEL CONTROLLER LE OPERAZIONI CRUD VERRANNO CREATE AUTOMATICAMENTE BISOGNERÀ POI AGGIUNGERE LE FUNZIONI DI RICERCA o comunque cosa viene richiesto dal prof:**

**CERCARE PER PROVINCIA:**

**[Authorize]**

**[HttpGet]**

**public IActionResult SearchByProvincia()**

**{**

**return View();**

**}**

**[Authorize]**

**[HttpPost]**

**public async Task<IActionResult> SearchByProvincia(string provincia)**

**{**

**var farmacie = await \_context.Farmacies**

**.Include(f => f.IdComuneNavigation)**

**.ThenInclude(c => c.IdProvinciaNavigation)**

**.ThenInclude(p => p.IdRegioneNavigation)**

**.Where(f => f.IdComuneNavigation.IdProvinciaNavigation.Denominazione.Contains(provincia))**

**.ToListAsync();**

**return View("Index", farmacie);**

**}**

**CERCARE PER COMUNE**

**[Authorize]**

**[HttpGet]**

**public IActionResult SearchByComune()**

**{**

**return View();**

**}**

**[Authorize]**

**[HttpPost]**

**public async Task<IActionResult> SearchByComune(string comune)**

**{**

**var farmacie = await \_context.Farmacies**

**.Include(f => f.IdComuneNavigation)**

**.ThenInclude(c => c.IdProvinciaNavigation)**

**.ThenInclude(p => p.IdRegioneNavigation)**

**.Where(f => f.IdComuneNavigation.Denominazione.Contains(comune))**

**.ToListAsync();**

**return View("Index", farmacie);**

**}**

**RICERCA PER REGIONE**

**[Authorize]**

**[HttpGet]**

**public IActionResult SearchByRegione()**

**{**

**return View();**

**}**

**[Authorize]**

**[HttpPost]**

**public async Task<IActionResult> SearchByRegione(string regione)**

**{**

**if (string.IsNullOrEmpty(regione))**

**{**

**// Se la stringa della regione è vuota o nulla, restituisce una lista vuota o tutte le farmacie a seconda dei requisiti.**

**return View("Index", new List<Farmacia>());**

**}**

**var farmacie = await \_context.Farmacies**

**.Include(f => f.IdComuneNavigation)**

**.ThenInclude(c => c.IdProvinciaNavigation)**

**.ThenInclude(p => p.IdRegioneNavigation)**

**.Where(f => f.IdComuneNavigation.IdProvinciaNavigation.IdRegioneNavigation.Denominazione.Contains(regione))**

**.ToListAsync();**

**return View("Index", farmacie);**

**}**

**CERCA PER NOME**

**[Authorize]**

**[HttpGet]**

**public IActionResult SearchByDenominazione()**

**{**

**return View();**

**}**

**[Authorize]**

**[HttpPost]**

**public async Task<IActionResult> SearchByDenominazione(string denominazione)**

**{**

**var farmacie = await \_context.Farmacies**

**.Include(f => f.IdComuneNavigation)**

**.ThenInclude(c => c.IdProvinciaNavigation)**

**.ThenInclude(p => p.IdRegioneNavigation)**

**.Where(f => f.Descrizionefarmacia.Contains(denominazione))**

**.ToListAsync();**

**return View("Index", farmacie);**

**}**

**QUERY PER VISUALIZZARE TUTTO IL DB ALL’INTERNO DELLA TABELLA PRINCIPALE:**

**public async Task<IActionResult> Index()**

**{**

**var farmacie = await \_context.Farmacies**

**.Include(f => f.IdComuneNavigation)**

**.ThenInclude(c => c.IdProvinciaNavigation)**

**.ThenInclude(p => p.IdRegioneNavigation)**

**.Include(f => f.IdComuneNavigation.Fraziones)**

**.ToListAsync();**

**return View(farmacie);**

**}**

**// GET: Farmacie/Details/5**

**[Authorize]**

**public async Task<IActionResult> Details(int? id)**

**{**

**if (id == null)**

**{**

**return NotFound();**

**}**

**var farmacie = await \_context.Farmacies**

**.Include(f => f.IdComuneNavigation)**

**.ThenInclude(c => c.IdProvinciaNavigation)**

**.ThenInclude(p => p.IdRegioneNavigation)**

**.Include(f => f.IdComuneNavigation.Fraziones)**

**.FirstOrDefaultAsync(m => m.Id == id);**

**if (farmacie == null)**

**{**

**return NotFound();**

**}**

**return View(farmacie);**

**}**

**RICORDARSI POI DI MODIFICARE LA VISTA INDEX.CSHTML INN MODO CHE COMPAIANO TUTTE LE COSE DEL DB:**

**<td>**

**@Html.DisplayFor(modelItem => item.IdComuneNavigation.Denominazione)**

**</td>**

**<td>**

**@foreach (var frazione in item.IdComuneNavigation.Fraziones)**

**{**

**@frazione.Denominazione**

**<br />**

**}**

**</td>**

**<td>**

**@Html.DisplayFor(modelItem => item.IdComuneNavigation.IdProvinciaNavigation.Denominazione)**

**</td>**

**<td>**

**@Html.DisplayFor(modelItem => item.IdComuneNavigation.IdProvinciaNavigation.Sigla)**

**</td>**

**<td>**

**@Html.DisplayFor(modelItem => item.IdComuneNavigation.IdProvinciaNavigation.IdRegioneNavigation.Denominazione)**

**</td>**

**UNA VOLTA FINITO IL CONTROLLER SI PASSA ALLA CREAZIONE DELLE VISTE**

**per prima cosa creare una vista vuota chiamata Farmacie nella cartella Vievs/Home**

**@{**

**ViewData["Title"] = "Farmacie";**

**}**

**<div class="text-center">**

**<h1 class="display-4">Farmacie</h1>**

**</div>**

**ORA PASSARE ALLA CREAZIONE DELLE VISTE NELLA CARTELLA Views/farmacie:**

**VISTA SEARCH BY COMUNE:**

**@{**

**ViewData["Title"] = "Ricerca per Comune";**

**}**

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

**<form asp-action="SearchByComune" method="post">**

**<div class="form-group">**

**<label for="comune">Comune:</label>**

**<input type="text" class="form-control" id="comune" name="comune" />**

**</div>**

**<button type="submit" class="btn btn-primary">Cerca</button>**

**</form>**

**VISTA SEARCH BY DENOMINAZIONE**

**@{**

**ViewData["Title"] = "Ricerca per Denominazione";**

**}**

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

**<form asp-action="SearchByDenominazione" method="post">**

**<div class="form-group">**

**<label for="denominazione">Denominazione:</label>**

**<input type="text" class="form-control" id="denominazione" name="denominazione" />**

**</div>**

**<button type="submit" class="btn btn-primary">Cerca</button>**

**</form>**

**VISTA SEARCH BY PROVINCIA:**

**@{**

**ViewData["Title"] = "Ricerca per Provincia";**

**}**

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

**<form asp-action="SearchByProvincia" method="post">**

**<div class="form-group">**

**<label for="provincia">Provincia:</label>**

**<input type="text" class="form-control" id="provincia" name="provincia" />**

**</div>**

**<button type="submit" class="btn btn-primary">Cerca</button>**

**</form>**

**VISTA SEARCH BY REGIONE**

**@{**

**ViewData["Title"] = "Ricerca per Regione";**

**}**

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

**<form asp-action="SearchByRegione" method="post">**

**<div class="form-group">**

**<label for="regione">Regione:</label>**

**<input type="text" class="form-control" id="regione" name="regione" />**

**</div>**

**<button type="submit" class="btn btn-primary">Cerca</button>**

**</form>**

**SUCCESSIVAMENTE BIOSGNA MODIFICARE IL LAYOUT.CSHTML**

**IN QUESTA PARTE SI RICHIAMANO LE VISTE EI CONTROLLER IN MODO DA FARLO VEDERE A SCHERMO**

**<li class="nav-item">**

**<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>**

**</li>**

**<li class="nav-item">**

**<a class="nav-link text-dark" asp-area="" asp-controller="Farmacie" asp-action="Index">Farmacie</a>**

**</li>**

**<li class="nav-item">**

**<a class="nav-link text-dark" asp-area="" asp-controller="Farmacie" asp-action="SearchByRegione">Ricerca per Regione</a>**

**</li>**

**<li class="nav-item">**

**<a class="nav-link text-dark" asp-area="" asp-controller="Farmacie" asp-action="SearchByProvincia">Ricerca per Provincia</a>**

**</li>**

**<li class="nav-item">**

**<a class="nav-link text-dark" asp-area="" asp-controller="Farmacie" asp-action="SearchByComune">Ricerca per Comune</a>**

**</li>**

**<li class="nav-item">**

**<a class="nav-link text-dark" asp-area="" asp-controller="Farmacie" asp-action="SearchByDenominazione">Ricerca per Denominazione</a>**

**</li>**

**<li class="nav-item">**

**<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>**

**</li>**

**ADESSO PER MISURA PREVENTIVA CONTROLLARE CHE IL FILE appsettings.json**

**CI SIA L'EFFETTIVO NOME DEL DB**

{

"ConnectionStrings": {

"DefaultConnection": "Server=localhost;database=nomedb;Trusted\_Connection=true;Encrypt=false"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**OK ADESSI SI PASSA AL PROGRAM.CS**

**MODIFICARE LA DICHIARAZIONE DEL MAIN IN MODO ASINCRONO:**

**public static async Task Main(string[] args)**

**METTERE QUESTO PER LA CONNESSIONE AL DB:**

// Add services to the container.

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ?? throw new InvalidOperationException("Connection string 'DefaultConnection' not found.");

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(connectionString));

builder.Services.AddDbContext<FarmaciaContext>(options =>

options.UseSqlServer(connectionString));

builder.Services.AddDatabaseDeveloperPageExceptionFilter();

**AGGIUNGERE QUESTO PER PERMETTERE L’AGGIUNTA DEI RUOLI:**

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

.AddRoles<IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>();

builder.Services.AddControllersWithViews();

**ADESSO AVVIARE L’APPLICAZIONE E REGISTRARSI, UNA VOLTA REGISTRATO L’APPLICAZIONE TI CHIEDERÀ DI FARE UNA MIGRAZIONE DEL DB PER AGGIUNGERE LE TABELLE DEGLI UTENTI.**

**SE DOVESSE DARE ERRORE INSERIRE I SEGUENTI COMANDI NELLA CONSOLE DEI PACCHETTI NUGET**

**-se lo si vuole installare**: dotnet tool install --global dotnet-ef

**-se lo si vuole aggiornare**: dotnet tool update --global dotnet-ef

**-per eseguire la migrazione:**  dotnet ef migrations add NomeDellaMigrazione

**SPEGNERE L'APP E SEGUIRE I PROSSIMI PASSI**

**AGGIUNGERE QUESTO PER CREARE I RUOLI ADMIN, MANAGER, USER:**

using (var scope = app.Services.CreateScope())

{

var roleManager = scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>();

var roles = new[] { "Admin", "Manager", "User" };

foreach (var role in roles)

{

if (!await roleManager.RoleExistsAsync(role))

await roleManager.CreateAsync(new IdentityRole(role));

}

}

**AGGIUNGERE QUESTO PER CREARE I RUOLI ADMIN E MANAGER:**

**using (var scope = app.Services.CreateScope())**

**{**

**var userManager = scope.ServiceProvider.GetRequiredService<UserManager<IdentityUser>>();**

**string email = "admin@admin.com";**

**string password = "Tango.gay15,";**

**if (await userManager.FindByEmailAsync(email) == null)**

**{**

**var user = new IdentityUser();**

**user.UserName = email;**

**user.Email = email;**

**user.EmailConfirmed = true;**

**await userManager.CreateAsync(user, password);**

**await userManager.AddToRoleAsync(user, "Admin");**

**}**

**}**

**using (var scope = app.Services.CreateScope())**

**{**

**var userManager = scope.ServiceProvider.GetRequiredService<UserManager<IdentityUser>>();**

**string email = "manager@manager.com";**

**string password = "Tango.gay15,";**

**if (await userManager.FindByEmailAsync(email) == null)**

**{**

**var user = new IdentityUser();**

**user.UserName = email;**

**user.Email = email;**

**user.EmailConfirmed = true;**

**await userManager.CreateAsync(user, password);**

**await userManager.AddToRoleAsync(user, "Manager");**

**}**

**}**

**using (var scope = app.Services.CreateScope())**

**{**

**var userManager = scope.ServiceProvider.GetRequiredService<UserManager<IdentityUser>>();**

**string email = "admin@admin.com";**

**string password = "Tango.gay15,";**

**if (await userManager.FindByEmailAsync(email) == null)**

**{**

**var user = new IdentityUser();**

**user.UserName = email;**

**user.Email = email;**

**user.EmailConfirmed = true;**

**await userManager.CreateAsync(user, password);**

**await userManager.AddToRoleAsync(user, "Admin");**

**}**

**}**

**using (var scope = app.Services.CreateScope())**

**{**

**var userManager = scope.ServiceProvider.GetRequiredService<UserManager<IdentityUser>>();**

**string email = "manager@manager.com";**

**string password = "Tango.gay15,";**

**if (await userManager.FindByEmailAsync(email) == null)**

**{**

**var user = new IdentityUser();**

**user.UserName = email;**

**user.Email = email;**

**user.EmailConfirmed = true;**

**await userManager.CreateAsync(user, password);**

**await userManager.AddToRoleAsync(user, "Manager");**

**}**

**}**

**CREARE UN MENU A TENDINA NELL’INDEX:**

**CONTROLLER:**

**public async Task<IActionResult> Index(int? brandId, string sortOrder)**

**{**

**// Impostazione dell'ordinamento predefinito per il nome del prodotto**

**ViewData["NameSortParam"] = String.IsNullOrEmpty(sortOrder) ? "name\_desc" : "";**

**// Query per ottenere l'elenco delle marche per il menu a tendina**

**var brands = await \_context.Brands.OrderBy(b => b.BrandName).ToListAsync();**

**ViewBag.BrandId = new SelectList(brands, "BrandId", "BrandName", brandId);**

**// Query per ottenere i prodotti**

**var productsQuery = \_context.Products**

**.Include(p => p.Brand)**

**.Include(p => p.Category)**

**.AsQueryable();**

**// Applica l'ordinamento in base alla scelta dell'utente**

**switch (sortOrder)**

**{**

**case "name\_desc":**

**productsQuery = productsQuery.OrderByDescending(p => p.ProductName);**

**break;**

**default:**

**productsQuery = productsQuery.OrderBy(p => p.ProductName);**

**break;**

**}**

**// Filtra per marca se specificato**

**if (brandId.HasValue)**

**{**

**productsQuery = productsQuery.Where(p => p.BrandId == brandId);**

**}**

**var products = await productsQuery.ToListAsync();**

**return View(products);**

**}**

**VIEW**

**@using (Html.BeginForm("Index", "Products", FormMethod.Get))**

**{**

**<div class="form-group">**

**<label for="brandId">Brand:</label>**

**@Html.DropDownList("brandId", ViewBag.BrandId as SelectList, "All Brands", new { @class = "form-control" })**

**<button type="submit" class="btn btn-primary">Filter by Brand</button>**

**</div>**

**}**

**<table class="table">**

**<thead>**

**<tr>**

**<th>**

**<a asp-action="Index" asp-route-sortOrder="@ViewData["NameSortParam"]">@Html.DisplayNameFor(model => model.ProductName)</a>**

**</th>**