USAR AUTENTICACIONES, ROLES Y PERMISOS

FRONTEND

1. Agregar las siguientes líneas en el program.cs

```
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(option =>
    {
        option.LoginPath = "/Acceso/Index";
        option.ExpireTimeSpan = TimeSpan.FromMinutes(20);
        option.AccessDeniedPath = "/Home/Privacy";
    });
```

```
app.UseAuthentication();

app.UseAuthorization();
```

2. Crear una clase con la entidad Usuario

```
namespace FrontEnd.Models
{
    10 referencias
    public class Usuario
    {
        4 referencias
        public string Nombre { get; set; }
        9 referencias
        public string Correo { get; set; }
        8 referencias
        public string Clave { get; set; }
        4 referencias
        public string[] Roles { get; set; }
    }
}
```

3. Crear un controlador para el acceso de usuarios

```
using Microsoft.AspNetCore.Mvc;
using FrontEnd.Data;
using FrontEnd.Models;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authentication;
using System.Security.Claims;

namespace FrontEnd.Controllers
{
    0 referencias
    public class AccesoController : Controller
    {
        0 referencias
        public IActionResult Index()
        {
            return View();
        }
```

```
        [HttpPost]
        0 referencias
        public async Task<IActionResult> Index(Usuario _usuario)
        {
            DA_Logica da_usuario = new DA_Logica();
            var usuario = da_usuario.validarUsuario(_usuario.Correo, _usuario.Clave);

            if(usuario!=null)
            {
                var claims = new List<Claim>
                {
                    new Claim(ClaimTypes.Name , usuario.Nombre),
                    new Claim("Correo", usuario.Correo)
                };

                foreach(string rol in usuario.Roles)
                {
                    claims.Add(new Claim(ClaimTypes.Role, rol));
                }

                var claimsIdentity = new ClaimsIdentity(claims,CookieAuthenticationDefaults.AuthenticationScheme);

                await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIdentity));


                return RedirectToAction("Index", "Empleado");
            }
            else
            {
                return View();
            }

        }
        0 referencias
        public async Task<IActionResult> Salir()
        {
            await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
            return RedirectToAction("Index", "Acceso");
        }
    }
}
```

4. Para bloquear las vistas nos vamos al controlador y podemos bloquear todo con la palabra [Authorize]

```
using Microsoft.AspNetCore.Authorization;

namespace FrontEnd.Controllers
{
    [Authorize]
```

Si queremos solo bloquear cierta vista a ciertos roles nos colocamos encima de ella y colocamos

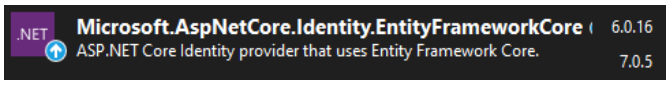
```
[Authorize(Roles = "Administrador")]
0 referencias
public async Task<IActionResult> Editar(int id)
{
```

5. Si queremos bloquear ciertas partes de una página por html. Agregamos

```
@using Microsoft.AspNetCore.Identity;
@if (User.IsInRole("Administrado"))
{
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
    </li>
}
```

BACKEND

1. Instalamos



2. Vamos a ApplicationDbContext y cambiamos de donde hereda la clase a IdentityDbContext y agregamos el método OnModelCreating

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

namespace Industria.Utilidades
{
    7 referencias
    public class ApplicationDbContext : IdentityDbContext
    {
        0 referencias
        public ApplicationDbContext(DbContextOptions options) : base(options)
        {
        }
        0 referencias
        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);
        }
```

3. Vamos al Control de administrador de paquetes y escribimos. Con esto se crea la BD de Usuarios

```
PM> add-migration SistemaDeUsuarios
PM> Update-database
```

4. Vamos al program.cs y agregamos