Para implementar **Entity Framework Core** e **Identity Framework** en tu proyecto **ASP.NET 8 MVC con SQL Server**, sigue estos pasos en el orden adecuado:

1. **Implementar Entity Framework Core** para que genere el contexto y los modelos basados en la base de datos existente.
2. **Implementar Identity Framework** para manejar autenticación y autorización con la estructura de **Areas**.

Paso 1: **Configurar Entity Framework Core (EF Core)**

Primero, debes agregar EF Core y configurar el contexto basado en tu base de datos existente.

1.1 **Instalar los paquetes de EF Core**

Ejecuta en la Consola del Administrador de Paquetes (PM) o en la terminal:

- *dotnet add package Microsoft.EntityFrameworkCore.SqlServer*
- *dotnet add package Microsoft.EntityFrameworkCore.Tools*

Si usarás migraciones, también instala:

- *dotnet add package Microsoft.EntityFrameworkCore.Design*

1.2 **Generar el DbContext y Modelos desde la BD**

Si ya tienes la base de datos en SQL Server, usa Scaffold-DbContext para generar el contexto y las entidades.

Ejecuta en la Consola del Administrador de Paquetes:

- *Scaffold-DbContext "Server=TU_SERVIDOR;Database=TU_BASE_DATOS;Trusted_Connection=True;TrustServerCertificate=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Context AppDbContext -Force*

Si usas dotnet CLI, ejecuta:

- *dotnet ef dbcontext scaffold "Server=TU_SERVIDOR;Database=TU_BASE_DATOS;Trusted_Connection=True;TrustServerCertificate=True;" Microsoft.EntityFrameworkCore.SqlServer -o Models -c AppDbContext –force*

Esto generará:

- **AppDbContext.cs** en la carpeta Models.
- **Las clases modelo** que representan las tablas de la base de datos.

1.3 **Registrar el DbContext en el proyecto**

Abre Program.cs y agrega el DbContext en los servicios:

```
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Configurar Entity Framework Core con SQL Server

builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();
```

Y en appsettings.json, define la conexión:

```
"DefaultConnection": "Server=192.168.11.194;Initial Catalog=TU_DB;User Id=TU_USER;Password=TU_PSW;Persist Security Info=true;Encrypt=false;"
```

Líneas que Debes Eliminar de AppDbContext.cs

Elimina este bloque de código dentro de AppDbContext.cs:

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

  if (!optionsBuilder.IsConfigured)

  {
optionsBuilder.UseSqlServer("Server=TU_SERVIDOR;Database=TU_BASE_DATOS;Trusted_Connection=True;TrustServerCertificate=True;");

  }

}
```

Paso 2: Implementar Identity Framework

Identity Framework se usa para autenticación y gestión de usuarios. Lo integraremos con EF Core.

2.1 Instalar los paquetes de Identity

Ejecuta:

- dotnet add package Microsoft.AspNetCore.Identity.EntityFrameworkCore
- dotnet add package Microsoft.AspNetCore.Identity.UI
- dotnet add package Microsoft.EntityFrameworkCore.SqlServer
- dotnet add package Microsoft.EntityFrameworkCore.Design

2.2 **Modificar el AppDbContext para Identity**

Edita AppDbContext.cs para que herede de IdentityDbContext:

```
using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

public class AppDbContext : IdentityDbContext<IdentityUser>

{

    public AppDbContext(DbContextOptions<AppDbContext>
    options) : base(options) { }

}
```

2.3 Configurar Identity en Program.cs

Modifica Program.cs para agregar Identity:

```
using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);
```

```
// Configurar Entity Framework Core
builder.Services.AddDbContext<AppDbContext>(options
=>options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnect
ion")));
// Configurar Identity con Entity Framework
builder.Services.AddDefaultIdentity<IdentityUser>(options =>
options.SignIn.RequireConfirmedAccount =
true).AddEntityFrameworkStores<AppDbContext>();
// Agregar servicios MVC
builder.Services.AddControllersWithViews();
var app = builder.Build();
app.UseAuthentication();
app.UseAuthorization();
app.MapRazorPages();
app.MapControllerRoute(
  name: "default",
  pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
```

2.4 Agregar Identity con scaffold (Generar UI y lógica de autenticación)

Ejecuta el siguiente comando para generar la estructura de **Areas/Identity**:

- *dotnet aspnet-codegenerator identity -dc AppDbContext*

Esto generará los archivos de autenticación dentro de **Areas/Identity**, incluyendo:

- Login, Register, Logout
- Administración de cuentas
- Restablecimiento de contraseñas

2.5 Aplicar Migraciones y Actualizar la Base de Datos

Si la base de datos no tiene las tablas de Identity, aplica las migraciones:

- *dotnet ef migrations add IdentitySetup*
- *dotnet ef database update*

Esto generará las tablas necesarias para manejar usuarios y roles.

Paso 3: Configurar Identity UI

Para que el sistema de autenticación funcione correctamente:

1. Verifica que app.UseAuthentication() y app.UseAuthorization() estén en Program.cs.
2. Habilita las páginas de Identity en Program.cs:
   - *app.MapRazorPages();*
3. Protege controladores con [Authorize]. Por ejemplo:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
[Authorize]
public class DashboardController : Controller
{
  public IActionResult Index()
  {
    return View();
  }
}
```

Paso 4: Probar el Sistema

Resumen del Orden de Implementación

1. **Configurar Entity Framework Core** (conexión a SQL Server, scaffold de modelos).
2. **Configurar Identity Framework** (agregar identidad a AppDbContext, configurar servicios en Program.cs).
3. **Generar la estructura de Identity (Areas/Identity)**.
4. **Crear y aplicar migraciones** para las tablas de Identity.
5. **Probar autenticación y autorización**.