**EF Core**

1. Create a new Empty .NET Core Web Application.

2. Install the following NuGet packages:

Microsoft.EntityFrameworkCore.SqlServer v7.0.20

Microsoft.EntityFrameworkCore.Tools v7.0.20

Microsoft.VisualStudio.Web.CodeGeneration.Design v7.0.12

3. Create DbContext.cs with it's constructor inside Data folder.

4. Add your connection string to the appsettings.json file

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=TunifyDB;Trusted\_Connection=true;TrustServerCertificate=True;MultipleActiveResultSets=true"

},

5. Configure DbContext and connection string in Program.cs

- string ConnectionStringVar = builder.Configuration.GetConnectionString("DefaultConnection");

- builder.Services.AddDbContext<TunifyDbContext>(optionsX => optionsX.UseSqlServer(ConnectionStringVar));

=================================================

6. Create Model folder for the ERD entities:

- Each FK need a navigator with it to access the object easily.

e.g:

class User

{

int subscription\_id ; //FK

Subscription subscription ;

}

## for the other table

- One to One relation: add a navigator with EntityClass datatype.

e.g: Class Subscription {User user;}

- One to Many relation: add a navigator with datatype of Icollection<EntityClass>

e.g: Class User {ICollection<Playlist> Playlists;}

=================================================

7. Add Entity DbSet to DbContext file:

- public DbSet<User> User { get; set; }

8. in the DbContext >> in OnModelCreating method:

- define composite key if there is any:

- modelBuilder.Entity<PlaylistSong>().HasKey(ps => new { ps.Playlist\_ID, ps.Song\_ID });

- configure the entity relationships: start with the entity that have the FK. (will have>> .HasOne()).

- add seeded data if needed

9. Create the interface and the service

10. Create the controller

11. Update Program.cs:

- builder.Services.AddControllers();

- // Configure JSON options to handle object cycles

builder.Services.AddControllers()

.AddJsonOptions(options =>

{

options.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.Preserve;

});

- // Register the repository

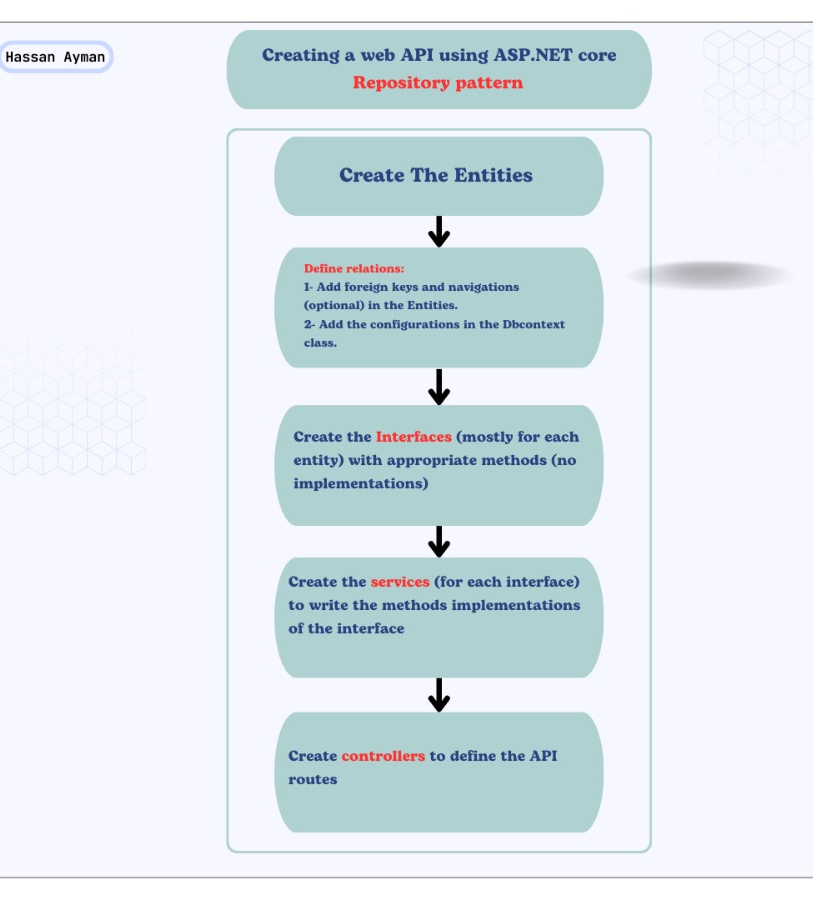
builder.Services.AddScoped<IPlaylist, PlaylistService>();

- app.MapControllers();

12. Run Migrations: From Tools --> Nuget --> console:

- Add-Migration MigName

- Update-Database



**Identity and DTO**

What is the Identity?

- It's a membership system that adds login functionality in your application.

- It's provides features like: Roles, Permissions...

Why to use?

- Security: Protecting your sensetive data and preventing the unautheriazed access

- Ensure that only authorized user can access certain parts of your application

- Managing your personal data across the application

What is the DTO?

Are simple objects used to transfer data between layers across the application (Containers of data).

Why to use?

- Isoliting data from business logic

- Protecting sensetive data by limiting exposed information

- Optimizing data transfer bu including only the needed data

Steps:

1- Install the "Microsoft.AspNetCore.Identity.EntityFrameworkCore" package.

2- Extend AppDbContext:

- Ensure AppDbContext extends IdentityDbContext<IdentityUser>.

3- Create DTOs:

- Create Data Transfer Objects (DTOs) for user management, such as RegisterDto and LoginDto.

4- Create the Interface:

- Define an interface (e.g., IAccount) for account-related operations.

5- Create the Service

- Implement the service (e.g., IdentityAccountService : IAccount) that adheres to

the IAccount interface and handles authentication logic.

6- Create the Controller: (Ex: AccountController)

- Implement a controller that handles registration, login, and optionally password management.

7- Update "Program.cs":

- Add the following services and configuration:

- builder.Services.AddScoped<IAccount, IdentityAccountService>();//Before app

- builder.Services.AddIdentity<IdentityUser, IdentityRole>()

.AddEntityFrameworkStores<AppDbContext>();//Before app

- app.UseAuthentication();//After app

8- Run Migrations: From Tools --> Nuget --> console:

- Add-Migration MigName

- Update-Database

**JWT Token**

A computer screen shot

Description automatically generated

