Project Development Roadmap

Note: In order to open the links, you must download and open the pdf file with a browser.

Note: For database, you can either use SQL Server LocalDB if you use Windows and Visual Studio Community, or SQLite (https://www.sqlite.org/) if you use an operating system other than Windows, or SQL Server with Docker if you use an operating system other than Windows.

1. Visual Studio Community installation for Windows:
   https://need4code.com/DotNet/Home/Index?path=.NET\00_Files\Visual%20Studio%20Community\Installation.pdf

2. Rider for MAC:
   https://www.jetbrains.com/rider

3. Docker Desktop installation for MAC:
   https://need4code.com/DotNet?path=.NET%5C00_Files%5CDocker%5CDocker%20Microsoft%20SQL%20Server.pdf

4. Create a .NET Aspire Empty App project.

5. Give Solution name as your project name and if you want change the solution folder in Location, Create in new folder option must be checked.

6. Select .NET 8.0 as the Framework, check Configure for HTTPS and select .NET Aspire version as 8.2.

7. In Solution Explorer create a new project called CORE under your solution, right click the Solution then Add -> New Project, then search for Class Library and select it, give the project name CORE then select .NET 8.0 as the Framework.

8. Create the folders and classes under the CORE project as in:
   https://github.com/cagilalsac/PMSCTIS/tree/master/CORE

   You can right click on CORE project then Add -> New Folder to create a folder,
   you can right click on the folder then Add -> Class to create a new C# Class File with the extension .cs.

   The folder and file structure must be as below:
   APP\Domain\Entity.cs
   APP\Features\Handler.cs
   APP\Features\Request.cs
   APP\Features\Response.cs

9. Create a new project under your solution as Class Library and give the name APP.Projects.

10. Create the Tag entity class under the Domain folder as in:
    https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Domain/Tag.cs

11. Create the ProjectsDb DbContext class under the Domain folder as in:
    https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Domain/ProjectsDb.cs

12. Right click on your APP.Projects project then Manage NuGet Packages, then from Browse tab search for Microsoft.EntityFrameworkCore.SqlServer package, select the latest version starting with 8 and install.

13. If you want to use SQLite database instead of SQL Server LocalDB, right click on your APP.Projects project then Manage NuGet Packages, then from Browse tab search for System.Data.SQLite.Core package, select the latest version and install, then from the Browse tab search for Microsoft.EntityFrameworkCore.Sqlite package, select the latest version starting with 8 and install.

14. Create a new project under the solution named API.Projects, search for web api and select ASP.NET Core Web API template, select .NET 8 as the Framework, Authentication type as None, check Configure for HTTPS, check Enable OpenAPI support, Use controllers and Enlist in .NET Aspire orchestration.

15. Right click on API.Projects then Manage NuGet Packages, then from Browse tab search for Microsoft.EntityFrameworkCore.Tools package, select the latest version starting with 8 and install. Right click API.Projects in Solution Explorer then click Set as Startup Project.

16. To create your database under SQL Server (localdb)\MSSQLLocalDB or SQLite database file, from Visual Studio menu, click Tools -> NuGet Package Manager -> Package Manager Console, select the project that has the DbContext, which is APP.Projects as the Default project, run “add-migration v1” (v1 must be a unique name) in the console then after the operation is completed run “update-database”.

    For Rider you can use the UI for these operations as explained in:
    https://www.jetbrains.com/help/rider/Visual_interface_for_EF_Core_commands.html

    You can follow the sub topics such as Entity Framework Core: Add Migration and Entity Framework Core: Update Database. You must select the project which has the DbContext as explained in the Visual Studio database create operations above.

17. You can see your created database under Visual Studio menu -> View -> SQL Server Object Explorer. Expand SQL Server then (localdb)\MSSQLLocalDB. Under Databases our database called PMSCTISProjectsDB can be found. When necessarry, data can be viewed by right clicking the table under the database PMSCTISProjectsDB then clicking View Data. Data can also be inserted, updated or deleted from this window.

    If you use SQLite database, you can find your database file under API.Projects.

18. Right click on APP.Projects then Manage NuGet Packages, then from Browse tab search for MediatR package, select the latest version and install.

19. In APP.Projects, create a new folder named Features. Under Features create a new class called ProjectsDbHandler as in below:
    https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Features/ProjectsDbHandler.cs

20. Under Features folder in APP.Projects, create a new folder named Tags and create the class TagQueryHandler under this folder as below:
    https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Features/Tags/TagQueryHandler.cs

21. Then create the file containing TagCreateHandler and TagCreateRequest classes under APP.Projects -> Features -> Tags folder as below:
    https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Features/Tags/TagCreateHandler.cs

22. Then create the file containing TagUpdateHandler and TagUpdateRequest classes under APP.Projects -> Features -> Tags folder as below:
    https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Features/Tags/TagUpdateHandler.cs

23. Then create the file containing TagDeleteHandler and TagDeleteRequest classes under APP.Projects -> Features -> Tags folder as below:
    https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Features/Tags/TagDeleteHandler.cs

24. In API.Projects, open Program.cs to configure the Inversion of Control for the ProjectsDb and IMediator dependency injections as below in the IoC section:
    https://github.com/cagilalsac/PMSCTIS/blob/master/API.Projects/Program.cs

25. Instead of using the connection string we defined in ProjectsDb class, we will use the connection string from our project's configuration file which is appsettings.json. So open the ProjectsDb class and comment or delete the OnConfiguring method we overrode before as below:
    https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Domain/ProjectsDb.cs

Db classes inherited from the base DbContext class must have a constructor with a DbContextOptions instance parameter in order the use the connection string defined in appsettings.json.

Then add the connection string to appsettings.json as below:
https://github.com/cagilalsac/PMSCTIS/blob/master/API.Projects/appsettings.json

26. In API.Projects, create an empty API Controller called TagsController then modify it as below:
https://github.com/cagilalsac/PMSCTIS/blob/master/API.Projects/Controllers/TagsController.cs

27. In APP.Projects, create the Project and ProjectTag entity classes under the Domain folder as in:
https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Domain/Project.cs
https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Domain/ProjectTag.cs

    Also add the navigation property ProjectTags in the Tag entity:
    ```
    public List<ProjectTag> ProjectTags { get; set; } = new List<ProjectTag>();
    ```

28. In APP.Projects, add the Projects and ProjectTags DbSets in the ProjectsDb class in the Domain folder:
    ```
    public DbSet<Project> Projects { get; set; }
    public DbSet<ProjectTag> ProjectTags { get; set; }
    ```

29. In Package Manager Console, select the project that has the DbContext (ProjectsDb), which is APP.Projects as the Default project, run “add-migration v2” in the console then after the operation is completed run “update-database” to add the Projects and ProjectTags tables to the database.

30. In APP.Projects add ProjectQueryHandler class under the Features folder with ProjectQueryRequest and ProjectQueryResponse classes as below:
https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Features/Projects/ProjectQueryHandler.cs

31. In APP.Projects add ProjectCreateHandler class under the Features folder with ProjectCreateRequest class as below:
https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Features/Projects/ProjectCreateHandler.cs

32. In APP.Projects add ProjectUpdateHandler class under the Features folder with ProjectUpdateRequest class as below:
https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Features/Projects/ProjectUpdateHandler.cs

33. In APP.Projects add ProjectDeleteHandler class under the Features folder with ProjectDeleteRequest class as below: https://github.com/cagilalsac/PMSCTIS/blob/master/APP.Projects/Features/Projects/ProjectDeleteHandler.cs

34. From now on we will use the Scaffolding Templates that the Scaffolding Mechanism of Visual Studio uses, which generates code for our API Controllers. In order to use these templates, go to https://need4code.com/DotNet/Home/Index?path=.NET\00_Files\Scaffolding%20Templates and extract the Templates folder inside the compressed Templates.7z file under your API.Projects folder.

35. Right click the Controllers folder in your API.Projects, Add -> Controller. Then select API from the left menu, then API Controller with actions, using Entity Framework. Select Project entity as the Model class, ProjectsDb for the DbContext class and give the Controller name ProjectsController. After the Scaffolding operation, your controller will be created as below: https://github.com/cagilalsac/PMSCTIS/blob/master/API.Projects/Controllers/ProjectsController.cs