Table of Contents

[High Level API Diagram 2](#_Toc106926633)

[Repositories 2](#_Toc106926634)

[Interfaces 2](#_Toc106926635)

[Repositories 2](#_Toc106926636)

[Business Logic 2](#_Toc106926637)

[Interfaces 2](#_Toc106926638)

[BusinessLogic 2](#_Toc106926639)

[Dependency Injection 3](#_Toc106926640)

[ServiceExtensions 3](#_Toc106926641)

[Register Services for IoC in Programs.cs 3](#_Toc106926642)

[Controllers 3](#_Toc106926643)

# High Level API Diagram

Diagram

Description automatically generated

# Repositories

## Interfaces

1. IUsersRepository

## Repositories

1. UsersRepository and implement IUsersRepository. Inject DataContext

# Business Logic

## Interfaces

1. IUsersBusinessLogic

## BusinessLogic

1. UsersBusinessLogic and implement IUsersBusinessLogic. Inject UsersRepository

# Dependency Injection

## ServiceExtensions

Create an extension to add instances of Repositories and BusinessLogic to the container

All repositories and business logic will get added to this extension. Following code only showing Repository and BusinessLogic.

using Microsoft.Extensions.DependencyInjection;

using MSC.Api.Core.BusinessLogic;

using MSC.Api.Core.Repositories;

namespace MSC.Api.Core.Extensions;

public static class ServiceExtensions

{

    public static void RegisterRepos(this IServiceCollection collection)

    {

        //repositories

        collection.AddTransient<IUsersRepository, UsersRepository>();

        //business logic

        collection.AddTransient<IUsersBusinessLogic, UsersBusinessLogic>();

    }

}

## Register Services for IoC in Programs.cs

Inside the CUSTOM comment section, towards the end add

builder.Services.RegisterRepos();

# Controllers

Following controllers will be created. Check the source code for more details.

1. UsersController inject UsersBusinessLogic