**Exchange Rate System - .Net WEB API**

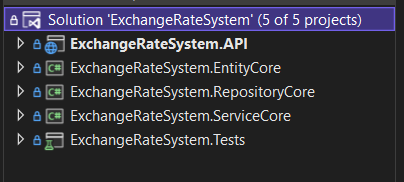
This application (API) is developed with technologies such as**: .Net6**, **Entity FrameworkCore**, **SqlServer**, **Rest Services.** Also, in this projet are included: **Cahcing**, **Logging** and **Unit Testing** with Xunit.

Notice: You can get project files from <https://github.com/GranitKrasniqi20/ExchangeRateSystem>

Below, I have described the structure of the project.

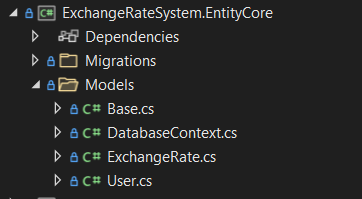
**Structure of project:**

Structure of application is **Clean Architecture.** Thesolution of application contains these projects (image below):



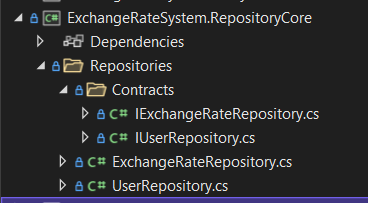
**ExchangeRateSystem.EntityCore** which contains**:**

Folder Models which include all models or entities of application and **DatabaseContext.cs** class in which is it implemented **Code First Approach.** Also, folder Migration which include all migrations of database.



**ExchangeRateSystem.RepositoryCore** or Data Acces Layer of application which contains:

Folder Repositories for Repositories User and ExchangeRate and folder Contracts for Interfaces of these Repositories.

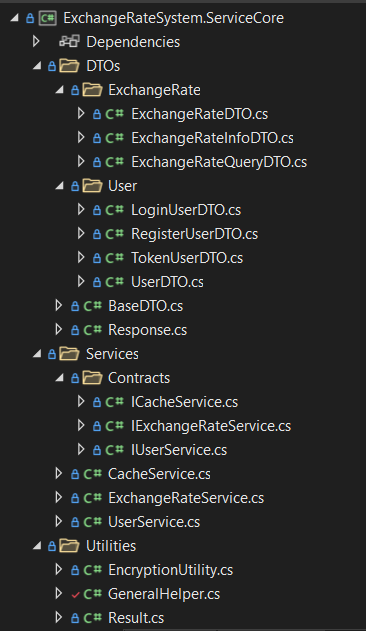


Notice: Repositories communicate with DatabaseContext at Entities for access to database and sent the data to Services.

**ExchangeRateSystem.Services** or *Business Logic Layer* of application contains:

**DTOs** folder which contains all DTO (Data transfer object) classes for interaction with the user.

**Utilities** folder which contains static classes and methods for general reasons.

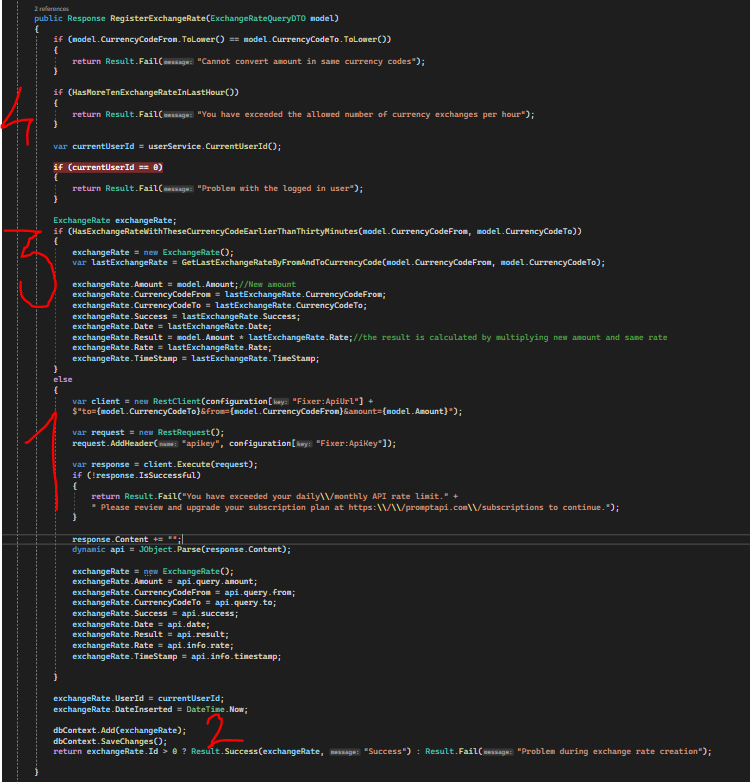


**Services** folder or main folder for business logic of application which contains Services in folder Services and Interfaces for these services in folder Contracts.

**CacheService.cs** contains some methods for the implementation of caching.

**UserService.cs contains** methods (register and login) for responses to enpoints of UserController.cs at PresentationLayer and other private methods.

**ExchangeRateService.cs** contains method for register/convert exchange rate which is main method for task requirements (image below) and other private methods. The red numbers in image indicate the implementation of the task requirements according to the numbers.



Notice: **Services** get the data from **Repositories**, develop business logic and response to **Controllers**.

**ExchangeRateSystem.API** or *Presentation Layer* of application contains:

**Program.cs** which is main class of project, contains all necessary configurations including **Bootsrapper.cs** class which contains all microservices of project.

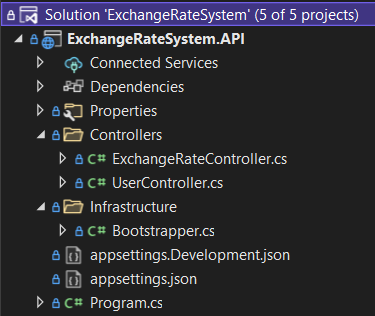
Controllers folder which contains all controllers of project:

**UserController.cs** contains two endpoints:

* **RegisterUser:** for add new User.
* **LoginUser:** for login with e-mail and password.

**ExcangeRateController.cs** contains only one endpoint.

* **RegisterExhangeRate:** for convert new ExchangeRate.



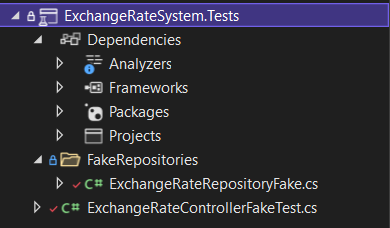
Notice: **Controllers** get data from **Services** (Busines logic Layer) and respond to client. Also, I don’t use **Try** and **Catch** on endpoints because I have created the Response and Result Class for responses and exceptions handling.

**ExchangeRateSystem.Tests** contains all unit test of application, for testing I I used the Xunit package. This project contains FakeControllers and FakeRepositories such as:

**ExchanRateRepositoryFakeTest.cs** containsstatic list of ExchangeRate objects.

**ExchanRateControllerFakeTest.cs** is a class which contains *Test Scenarios* and *Test Cases*:

* + **RegisteExchangeRate\_MustBeAddedNewRecord** which when executed should add a new record, calling the API (fixer.io) or without calling that API.
  + **RegisteExchangeRate\_MustBeAddedNewRecordWithoutApiCall** which when executed should add a new record, but without calling that API (fixer.io).



Notice: *FakeRepositories* contains static data same as *Repositories* and implement business logic same as *Services*. *FakeControllers* are same as *Controllers*, but get data and send request to *Fake* *Repositories.*

Below, I have described some general details for application:

* Password constraint are: At least 8 character, 1 uppercase leter and 1 special character.
* If API fixer.io is called over 100 times, you should replace ApiKey at appsettings.json file in ExchangeRateSystem.API project.

