Banking system

# Introduce

This is the basic system of banking with 3 main functions: check balance, deposit and withdraw.

# Technologies

ASP.Net Core, Entity Framework Core, Asynchronous programming, Serilog, Nunit, Shouldly, MS SQL Server.

# How to build code

1. Clone the repository (skip this step if you have the project on your machine)
2. Change ConnectionString in **appsettings.json** to your connection
3. Running EF Migration to create a test database (run this command: **Update-Database -StartupProject BankingApi -Project BankingData** in Package Manager Console tools)
4. Restore package by running **dotnet restore** in cmdat root folder
5. Build solution by running **dotnet build** in cmdat root folder
6. Go to folder .\BankingAPI and run **dotnet run** in cmd for running web api
7. After application started, open other cmd an access to folder.\BankingClient, run **dotnet run** in cmd for running client
8. On this, you can enjoy my app

# Handling Concurrency

For resolved problem with multiple asynchronous requests at the same time. I applied handle concurrency exception with column Version in table AccountDetails.

# Explain business

For each account number which is stored in table AccountInfo will be have multiple account details with difference currency for each account detail. When you call GetBalance and will be receive list of account details belong to this account number.

# Biggest Challenge

* Check concurrency make sure we do not miss any request
* Using Asynchronous for taking larger requests