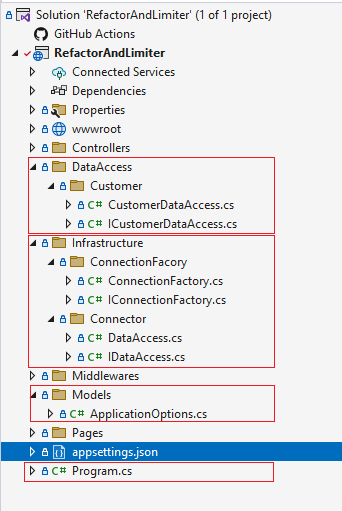
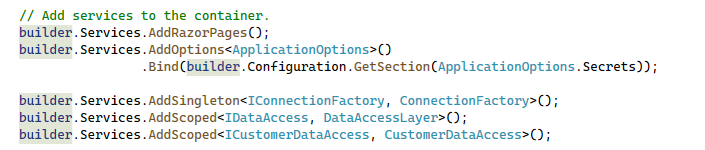
1. **Code Refactor**

The Code Refactor included the folders highlighted in red. The Code is split into Each Section named

1. Data Access – Which deals customers Insert , Update , Create
2. Infrastructure
   1. Connection Factory – to handle connection string
   2. Connected – Generic implementation for handling database operations
3. Model – IOption binding class to store Connection string to be injected in Connection Factory
4. Program.cs – All dependency Injection with IOptions Bindings



All Dependency Injection handled as scoped. Options Pattern for binding Configuration from app settings.



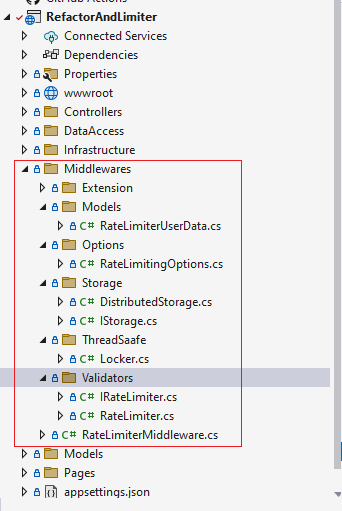
1. **Rate Limiter**

Rate Limiter implemented with the help of Middleware feature under the below circumstances

1. Application to be run on windows authentication to take user info from context
   1. If run on anonymous authentication, IP address of the sender will be used as user Id
2. The implemented rate limiter will block all the API for the user if limit reached. The current implementation does not check for individual API for the user.
3. Distributed Cache used.
   1. Redis cache can be used for better implementation on enterprise level
4. Locking Mechanism with semaphore and based on user key
   1. Implementation take from outside

Logic

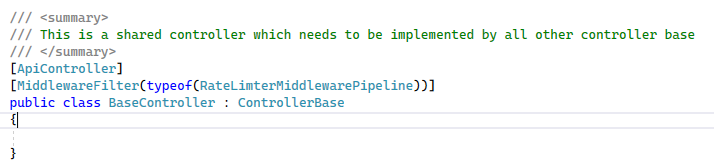
1. Any user hitting api .. first user info extracted
2. Checks initially whitelisted user, if yes skips all other validations
3. Checks user Is allowed to hit the api
   1. Forms a unique key for user
   2. Check in cache if user data exists
      1. If not exist
         1. Creates a cache entry with current time and initial count
      2. If exist
         1. Checks if time in cache crosses’ limit or max limit reach
            1. The throw “too many request status code”
         2. Else overwrite cache with incremented count



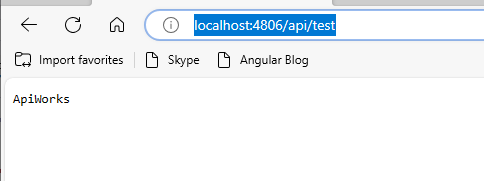
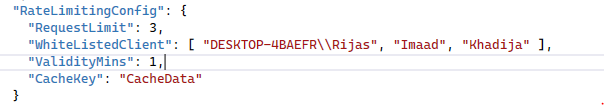
The below highlighted are are the files used to develop the middleware

1. Extension – The middleware extension for the register in the pipeline and register in application builder
2. Models – to store the configurations used middleware. The options to dependency inject in the classes to be used
3. Storage – Interface and implementation for to hold distributed cache
4. Thread Safe – locking implementation from stack overflow
5. Validators – Two interface implementations for whitelist and allow check with rate limiter logic

Middleware implemented on api controller which needs to be inheriting Base Contoller



How to test

1. Run the Project in “Windows Authentication”
2. In browser call api “[localhost:4806/api/test](http://localhost:4806/api/test)”
3. 
4. Any configuration changes can be done in appsettings
5. 

All dependency injected as singleton

