Тестовое задание:

1. Реализовать скрип/консольное приложение/сервис, загружающий исторические данные из http://www.cbr.ru/scripts/XML_daily.asp?date_req=01/01/2021
2. Реализовать REST API сервис аналогичный http://www.cbr.ru/scripts/XML_daily.asp.
3. Реализовать REST API сервис, выдающий историю изменения курса.
   3.1. История курсов валют должна быть на каждый день, без пропусков.
   3.2. Объяснить, что нужно добавить или изменить, чтобы сервис мог выдержать 1500 запросов в секунду.
4. Сервисы должны быть закрыты авторизацией.

Решение:

п.1. Загрузка исторических данных.

Опыт моей работы с XML из консольного приложения можно увидеть:
https://github.com/Constantine-SRV/ServiceLogonMultifactor/blob/master/ServiceLogonMultifactor/Configs/Services/Generic/ConfigReader.cs
https://github.com/Constantine-SRV/ServiceLogonMultifactor/blob/master/ServiceLogonMultifactor/Configs/Services/Generic/ConfigWriter.cs
Я решил сделать внесение данных из сервиса ЦБ в базу средствами MSSQL в хранимой процедуре [dbo].[usp_fillDyliCBR]
https://github.com/Constantine-908/TestTask2022/blob/master/Database_testDB/dbo/Stored%20Procedures/usp_fillDyliCBR.sql
Из сложностей, которые сразу были незаметны - оказалось, что при запросе на дату, когда курс не устанавливался, ЦБ возвращает данные на предыдущую установленную дату. Пришлось ввести таблицу [dbo].[tbl_noRatesDates] с двумя полями: дата на которую нет курса и дата используемая вместо запрашиваемой.
Процедура [dbo].[usp_selectCBRRates]
https://github.com/Constantine-908/TestTask2022/blob/master/Database_testDB/dbo/Stored%20Procedures/usp_selectCBRRates.sql
проверяет, есть ли данные за запрашиваемую дату и, если нет, то вызывает [dbo].[usp_fillDyliCBR] для заполнения. Это решение позволяет отказаться от сервиса по заполнению данных. В случае необходимости можно добавить вызов ежедневного заполнения данных в таймер.
https://github.com/Constantine-908/TestTask2022/blob/master/WebApiWebLoad/Services/TimedHostedService.cs

п.2. REST API сервис аналогичный http://www.cbr.ru/scripts/XML_daily.asp

Контроллер UspSelectCBRRatesController и вышеуказанная SP
https://github.com/Constantine-908/TestTask2022/blob/master/WebApiWebLoad/Controllers/UspSelectCBRRatesController.cs
возвращают данные аналогично сервису ЦБ. Выбор типа ответа осуществляется заголовком запроса.

```
curl -X GET "https://srv-2022/load/UspSelectCbrRates?date_req=2021-03-01" -H  "accept: application/xml"
```

п.3. Реализовать REST API сервис выдающий историю изменения курса.

Сервис UspSelectCBRhistoryControllerService
https://github.com/Constantine-908/TestTask2022/blob/master/WebApiWebLoad/Controllers/UspSelectCBRhistoryController.cs

```
curl -X GET "https://srv-2022/load/UspSelectCBRhistory?dtStartString=2022-01-01&charCode=USD&days=60" -H
"accept: application/json"
```

3.1 Хранимая процедура [dbo].[usp_selectCBRhistory] возвращает курс как за рабочие дни, так и за выходные.
3.2 1500 запросов выдерживает, подробнее о тестах на быстродействие ниже. Дополнительно ускорить можно денормализовав хранение данных или повысив производительность VM.

п.4 Сервисы должны быть закрыты авторизацией:

В рамках этого проекта была использована встроенная авторизация Windows/AD с разграничением и проверкой прав доступа на MSSQL сервере.

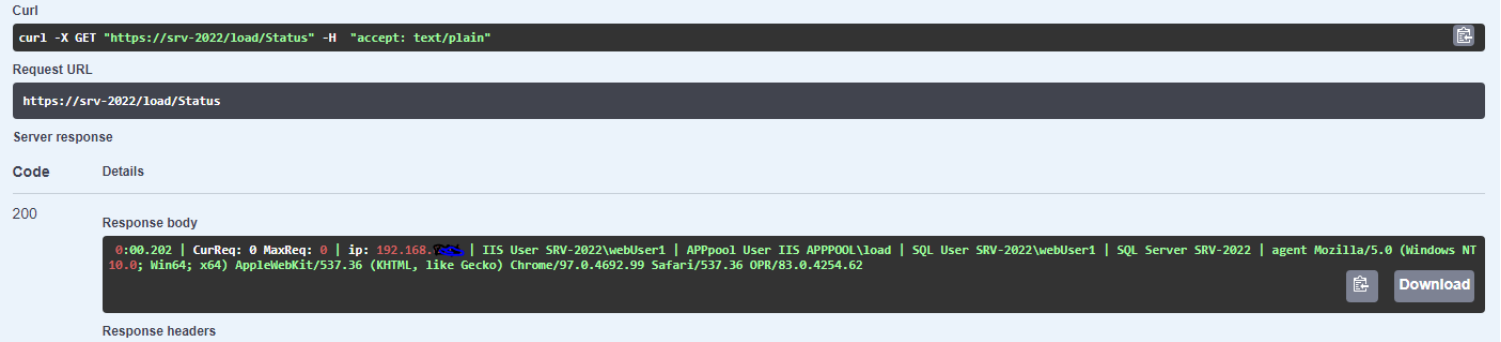
Данное решение имеет как плюсы, так и минусы. И хотя это решение для задач, реализованных в рамках данного задания, не совсем оптимально, для систем, с которыми работают внутренние пользователи, это наилучшее решение.

Реализация:

1. Приложение в IIS настроено с Basic Autentification.
2. Все запросы пользователей выполняются к базе данных в контексте пользователя

```
WindowsIdentity.RunImpersonatedAsync(((WindowsIdentity) HttpContext.User.Identity).AccessToken
```

3. В базе создана роль cbrReader, которой даны права на выполнение необходимых хранимых процедур.
4. Пользователи добавлены в эту роль.
   https://github.com/Constantine-908/TestTask2022/blob/master/Database_testDB/Security/cbrReader.sql
5. У учетной записи, из-под которой работает Application Pool приложения, доступа к MS SQL server нет.
6. Прямого доступа к таблицам у пользователей нет.
7. Для проверки прав доступа и демонстрации, как это работает, служит контроллер
   https://github.com/Constantine-908/TestTask2022/blob/master/WebApiWebLoad/Controllers/StatusController.cs
   и хранимая процедура
   https://github.com/Constantine-908/TestTask2022/blob/master/Database_testDB/dbo/Stored%20Procedures/usp_status.sql



8. В базу данных добавлена функция логирования вызовов хранимых процедур. В таблице
   https://github.com/Constantine-908/TestTask2022/blob/master/Database_testDB/dbo/Tables/tbl_log.sql
   при вызове хранимых процедур сохраняется:
   8.1. ИД процедуры.
   8.2. ИД пользователя.
   8.3. Время вызова.
   8.4. Длительность.
   8.5. Количество строк.

При необходимости можно добавить параметры, но это повлияет на производительность.

Дополнительно добавлены заготовки сервисов:

- Статистика количества запросов: https://github.com/Constantine-908/TestTask2022/blob/master/WebApiWebLoad/Services/ErrorLogService.cs
- Таймер: https://github.com/Constantine-908/TestTask2022/blob/master/WebApiWebLoad/Services/TimedHostedService.cs
- Обработка ошибок: https://github.com/Constantine-908/TestTask2022/blob/master/WebApiWebLoad/Services/ErrorLogService.cs

Тестирование производительности:

Для тестирования производительности добавлено консольное приложение https://github.com/Constantine-908/TestTask2022/tree/master/ConsoleWebLoadMuktitask , которое позволяет сымитировать требуемые 1500 запросов в секунду.
Примеры конфигурационных файлов для тестирования в папке
https://github.com/Constantine-908/TestTask2022/tree/master/ConsoleWebLoadMuktitask/ConfigsExamples
Результаты:
На виртуальной машине (i5-6600, 6Gb, простой SSD) были получены следующие результаты:

- Количество вызовов хранимых процедур 122 843 в минуту.
- HTTP Request Arrival Rate - 1571 в секунду

Для отображения счетчика количества одновременных транзакций инструкция select обернута командами начала и подтверждения транзакции, в реале они не нужны.

```
8  select count(logID) as RequestsCount,datepart(day,dt) as dd ,datepart(hour,dt) as hh ,datepart(minute,dt) as mm
9  from [tbl_log] (nolock)
10 group by datepart(day,dt), datepart(hour,dt)  ,datepart(minute,dt)
11 order by  count(logID) desc
12
```

100 %

Results | Messages

	RequestsCount	dd	hh	mm
1	122843	20	15	38
2	98287	20	12	29
3	95859	20	12	4
4	95613	20	15	37
5	91714	20	12	3