Homework for applicants:

Description:

1. Implement a C# Web API using .NET using the requirements listed below
2. Showcase the following:
    a. Usage of Dependency Injection
    b. Generics, extension methods
    c. Async programming
    d. Multi-threading
    e. Exceptions handling
    f. JSON serialization/deserialization
3. Benefits:
    a. Add Swagger support
    b. Add Postman support
    c. Attributes creation & usage
    d. Middleware and filter usage
    e. Unit tests

Implement an API capable of generating and returning a subsequence from a sequence of Fibonacci numbers. The API should have a controller with an endpoint accepting the following parameters:

1. The index of the first number in Fibonacci sequence that starts subsequence.
2. The index of the last number in Fibonacci sequence that ends subsequence.
3. A Boolean, which indicates whether it can use cache or not.
4. A time in milliseconds for how long it can run. If generating the first number in subsequence takes longer than that time, the program should return error. Otherwise as many numbers as were generated with extra information indicating the timeout occurred.
5. A maximum amount of memory the program can use. If, during the execution of the request this amount is reached, the execution aborts. The program should return as many generated numbers as possible similarly to the way it does in case if timeout is reached.

To simulate a CPU intensive task there is an artificial requirement to do 500ms delay after a new Fibonacci number is computed. The delay should be applied to the current thread / execution context. This means another thread / execution context can proceed with the next number calculation immediately when the current completes.

The return from the endpoint should be a JSON containing the subsequence from the sequence of Fibonacci numbers that is matching the input indexes.

Please bear in mind, there could be many requests landing simultaneously.

If cache is enabled, you can rely on it to speed up the Fibonacci numbers generation.

The cache shall be invalidated after a period of no use, where the period is defined in configuration.