**Assignment Wouter**

1. Write a simple C# (Core 3.1) REST web service with two methods:
   1. GUID StartCalculation(some input values) à calculates ‘something’, but takes its time, somewhere between 20 seconds and 1 minute.
   2. StatusObject GetStatus(GUID) à can be called to get a (JSON) status object back. This object should contain the status (i.e. ‘running’, ‘failed’, ‘completed’), progress(10%, 20% etc) and – when completed – the outcome of the calculation.
2. Write a simple C# Web App that consists of a single page where a user can enter some input data for the calculation. It also contains a ‘start’ button. The web app performs both client and server side validation of the user input (are fields empty, is the value numeric). The Start button will only be enabled when the client validation passes. When the user clicks the button, the REST web service’s startCalculation is fed with the user input. The progress of the calculation is visualised (knock yourself out!) and when the calculation is done, the result is displayed.
3. Both client and server app have sufficient unit tests to prove they work.
4. Version control of the code is done through GitHub (you can create a free account there).

Bonus 1: The server and/or client use SQL Server to persist their data

Bonus 2: The calculation(s) is/are written in stored procedures which are also unit tested

Bonus 3: The solution is deployed to an Azure App Service (free account available here: [https://azure.microsoft.com/en-us/free](https://eur03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fazure.microsoft.com%2Fen-us%2Ffree&data=04%7C01%7C%7Cd3618d1826e1463f965f08d8b60ee8ae%7C3a15904d3fd94256a753beb05cdf0c6d%7C0%7C0%7C637459522428957765%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=gBq8dsiXr9ddMibHacHzibxZcIEBwam78RU%2B3HE2Akk%3D&reserved=0) )

# Solution strategy

## Version Control

A GIT repository has been set up on GitHub for version control.

<https://github.com/WouterDeBotWork/InterviewAssignment1>

## REST web service

The purpose of the REST service is to obtain a GUID as well as checking the status of the calculation that is used to obtain the GUID. Two methods will be used to achieve these requirements:

### Method 1: StartCalculation

**Input**: Age, length (cm)

**Calculation**:

1. GUID will have to be generated immediately so it can be passed to the GetStatus method. (Use built in GUID method of C#).
2. Create new Object according to StatusObject model (POST)
   1. GUID = GUID
   2. Status = ‘running’
   3. Progress = 1/10
   4. Result = []
3. For-loop with calculation (i = 2; i <10; i++) (also update StatusObject model)
   1. Thread.sleep(3000ms)
   2. IntermediateResult += Age + length
   3. Update StatusObject (PUT):
      1. GUID = GUID
      2. Status = ‘running’
      3. Progress = i/10
      4. Result = IntermediateResult
4. FinalResult = Result + Age + length
5. Update StatusObject (PUT):
   * 1. GUID = GUID
     2. Status = ‘Completed’
     3. Progress = 100%
     4. Result = IntermediateResult

**Output**: GUID (long)

**Notes:** POSTs and PUTs should be written to memory or a database. If for some reason calculation fails set Status to Failed.

### Method 2: GetStatus

**Input**: GUID (long)

**Output**: StatusObject (JSON)

**Notes:**

From a database (or Memory?) return Object as JSON.

## Web app

## 

### Landing Page

The web application will consist of a single HTML page with input fields and a start button.

After pressing the start button the calculation will start and a progress bar will appear.

Javascript will be used for client-side validation as well as doing server calls. This shall be done through Jquery AJAX calls.

**Input fields:**

Age

Length

**Progress bar:**

Bar that is divided in 10 parts where each part is 10% progress.

**Proces:**

1. Client fills in input fields
2. Client-side Validation whether input is correct
3. Enable start button
4. Start button does API call through JQUERY
5. Server-side validation of input
6. API gives back JSON each loop
7. DIV that contains progress bar is unblocked
8. Update progress bar every time JSON is pushed form server? Or check via JQUERY (Call GetStatus method)

### Validation

**Client side**

Javascript:

* check if value type = numeric
* check if box is not empty
* Length in cm

Give proper error messages.

**Server side**

* check if type = long
* check if var is not empty

Give proper error messages.

## Unit Tests

**Client side:**

1. Happy flow:
   1. Input numeric.
   2. Output: JSON values = JSON that is tested and (manually) verified
2. Input fields:
   1. Input nonnumeric
   2. Check for error message
   3. Done for all input fields
3. Validate models (c# , razor pages)

**Server side:**

1. Happy flow:
   1. Input numeric.
   2. Output: JSON values = JSON that is tested and (manually) verified
   3. Check for multiple iterations
2. Error handling:
   1. Input incorrect types
   2. Check for error message