Maersk Offline Exercise

Introduction

Software development is an essential part of all our engineering roles, and we are very curious to see how you approach a programming task. In this exercise, we ask you to implement a small feature on top of an existing codebase, to simulate a semi-realistic work situation.

The existing codebase

As a starting point, we have attached a small C#/.NET Core solution which exposes a simple API, allowing clients to sort a list of numbers. In order to not force clients to wait for the results, we want to sort the numbers in a background job, and make it possible to query these jobs at a later point. This is not the case at the moment, however, and in the current implementation values are sorted synchronously before a response is returned to the client. Your task will be to implement this background processing, and replace the current synchronous API with an asynchronous one.

The current solution only has a minimal amount of structure, and you are welcome to change anything in the solution as you see fit.

Your task

Your task is to implement all the logic necessary to process the jobs asynchronously. This includes implementing the following three controller actions in the `SortController`:

- `EnqueueJob`: An endpoint to which clients can post a list of numbers to be sorted. The endpoint returns immediately, without requiring clients to wait for the jobs to complete.
- `GetJob`: Returns the current state of a specific job.
- `GetJobs`: Return the current state of all jobs (both pending and completed).

To keep things simple, the solution should be fully self-contained and not depend on any external services (but it is of course okay to reference packages from nuget as needed). All data can be stored in memory. Finally, we do not expect you to write any integration or unit tests for the solution.

Example requests

Once the asynchronous endpoints have been implemented, the following could be an example of how a session could look:

```
> curl --request GET "http://localhost:5000/sort"
[]

> curl --request POST
       --header "Content-Type: application/json"
       --data "[2, 3, 1, 5, 3, 1, -20, 2]"
       "http://localhost:5000/sort"
{
  "id":"fcdffff4-1017-410c-9aa9-44c04e6aac6f",
  "status":"Pending",
  "duration":null,
  "input":[2,3,1,5,3,1,-20,2],
  "output":null
}

> curl --request GET
       "http://localhost:5000/sort/fcdffff4-1017-410c-9aa9-44c04e6aac6f"
{
  "id":"fcdffff4-1017-410c-9aa9-44c04e6aac6f",
  "status":"Completed",
  "duration":"00:00:05.0134826",
  "input":[2,3,1,5,3,1,-20,2],
  "output":[-20,1,1,2,2,3,3,5]
}

> curl --request GET "http://localhost:5000/sort"
[{
  "id":"fcdffff4-1017-410c-9aa9-44c04e6aac6f",
  "status":"Completed",
  "duration":"00:00:05.0134826",
  "input":[2,3,1,5,3,1,-20,2],
  "output":[-20,1,1,2,2,3,3,5]
}]
```

Handing in the solution

Once you have implemented the above described changes, you should zip your full solution and send it to us, after which we will review it and get back to you as quickly as possible.

Good luck! We look forward to reviewing your solution.