Maersk Offline Technical Challenge

Introduction

This document provides a homework challenge for candidates applying for software engineering positions within Maersk.

The goal is to see:

- How you approach a problem
- How you design, structure, implement, and deliver a complete solution

This will be followed by an on-site discussion with Maersk software engineers.

Handing in your solution

- Your code and any other deliverables must be provided as a link to a private Git repository (Azure DevOps, GitLab, GitHub, or Bitbucket)
- Your solution must only be accessible to you and us; please make sure it is not available for a wider audience, especially not publicly. We would like to reuse the challenge, please help us keeping it fair!

The problem

To simulate a realistic problem, you will be implementing a minimal job scheduling solution exposed via a web API. In short, your solution should make it possible for clients to enqueue an array of numbers to be sorted in the background and to query the state of any previously enqueued job.

Functional requirements

You are asked to develop a web API that supports the following operations:

- The client can enqueue a new job, by providing an unsorted array of numbers as input
- The client can retrieve an overview of all jobs (both pending and completed)
- The client can retrieve a specific job by its ID (see below), including the output (sorted array) if the job has completed

A job takes an unsorted array of numbers as input, sorts the input using an algorithm of your choice, and outputs the sorted array. Apart from the input and output arrays a job should include the following metadata:

- An ID - a unique ID assigned by the application
- A timestamp - when was the job enqueued?
- A duration - how much time did it take to execute the job?
- A status - for example "pending" or "completed"

All jobs should be processed in the background and clients should *not* be forced to wait for jobs to complete. To view the output of a job (the sorted array), the client must query a previously enqueued job.

Your solution

Your delivery should:

- Include a fully functional solution built using C#, ASP.NETCore, and .NET Core
- Expose a web API and implement relevant background job processing
- Include a suite of unit tests covering key components of the solution
- Contain logging functionality
- Include the complete codebase needed to run the solution
- If you use any third-party libraries, these must be referenced via Nuget
- Contain a README explaining how to build, test, and run your solution locally, plus any additional details (e.g. architectural decisions) you might find relevant for us to know

Your solution should be self-contained and all data can be stored **in memory** for the sake of simplicity. No external infrastructure should be needed to run the solution.

The requirements intentionally leave a lot of design decisions up to you, so use your solution to showcase how you would approach the described problem from both a (technical) design and implementation point-of-view.

Good Luck! We look forward to reviewing your solution.