

Coding Test - Part 1

The following API is a public API used to report rainfall from measurement stations around the UK;

 [Rainfall API reference](#)



There are a number of documented endpoints, but you should only need to consume one of them for this part of the exercise. We recommend returning to the documentation once you've understood the requirements.

The take home part of the task is to create a .net core API (minimum .net core 6) which sources data from the public rainfall API and maps the responses into a new contract.

We use OpenAPI specs to design and declare our data contracts. Attached is the OpenAPI spec which the new application must adhere to;

```
openapi: 3.1.0
info:
  title: Rainfall Api
  version: '1.0'
  contact:
    name: Sorted
    url: 'https://www.sorted.com'
  description: An API which provides rainfall reading data
servers:
  - url: 'http://localhost:3000'
    description: Rainfall Api
tags:
  - name: Rainfall
    description: Operations relating to rainfall
paths:
  '/rainfall/id/{stationId}/readings':
    parameters:
      - schema:
          type: string
          name: stationId
          in: path
          required: true
          description: The id of the reading station
      - schema:
          type: number
          minimum: 1
          maximum: 100
          name: count
          in: query
          required: false
          default: 10
```

```

    description: The number of readings to return
get:
  operationId: get-rainfall
  summary: Get rainfall readings by station Id
  description: Retrieve the latest readings for the specified stationId
  tags:
    - Rainfall
  responses:
    '200':
      description: A list of rainfall readings successfully retrieved
      content:
        application/json:
          schema:
            $ref: '#/components/responses/rainfallReadingResponse'
    '400':
      description: Invalid request
      content:
        application/json:
          schema:
            $ref: '#/components/responses/errorResponse'
    '404':
      description: No readings found for the specified stationId
      content:
        application/json:
          schema:
            $ref: '#/components/responses/errorResponse'
    '500':
      description: Internal server error
      content:
        application/json:
          schema:
            $ref: '#/components/responses/errorResponse'
components:
  schemas:
    rainfallReadingResponse:
      title: Rainfall reading response
      type: object
      description: Details of a rainfall reading
      properties:
        readings:
          type: array
          items:
            $ref: '#/components/schemas/rainfallReading'
    rainfallReading:
      title: Rainfall reading
      type: object
      description: Details of a rainfall reading
      properties:
        dateMeasured:
          type: string
          format: date-time
        amountMeasured:
          type: number
          format: decimal
  error:

```

```

    title: Error response
    type: object
    description: Details of a rainfall reading
    properties:
      message:
        type: string
      detail:
        type: array
        items:
          type: array
          $ref: '#/components/schemas/errorDetail'
      additionalProperties: false
    errorDetail:
      type: object
      description: Details of invalid request property
      properties:
        propertyName:
          type: string
        message:
          type: string
        additionalProperties: false
  responses:
    rainfallReadingResponse:
      description: Get rainfall readings response
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/rainfallReadingResponse'
    errorResponse:
      description: An error object returned for failed requests
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/error'

```

We're not prescriptive with the approach or which libraries must be used, but we're looking for code that shows a good understanding of the .net core framework, especially working with REST APIs.

Deliverables

Please push your work to a remote git repository (e.g. GitHub) and share the location and any required credentials with us. Commit as you go to show your working process, rather than just one big commit at the end.

We'll pull down the solution you've put together and expect to see tests passing. We should be able to launch the app from an IDE without additional setup.

Next steps

Upon success at this stage you will be invited to the next stage which will be a pairing exercise with one of our senior engineers. In this session we'll continue to work on your application by tackling one of a variety of new requirements.