Software Developer Technical Test

Submission format

Upload your artifacts to a public **GitHub** repository and provide us with its URL. The repository should contain all the necessary source code and files required to make your solution work.

Include a `README.md` file at the root of your repository. This file should provide any special instructions required to run the project.

Tools

Unless explicitly specified otherwise, utilize any tools that you deem appropriate for this task, but be sure to document a comprehensive list of all tools in the `README.md` file.

Questions

If you have any questions regarding the test or the position, please do not hesitate to contact Alvaro Ruano at alvaro@relishiq.com

Part 1: **The API Endpoint**

You've been hired by a startup called **MetaPhoto** to help on the development of their new website and mobile app which helps photographers to organize their photo libraries.

The backend team has provided you with three internal API endpoints which let you `GET` the raw data they have stored for their users:

- https://jsonplaceholder.typicode.com/users
- https://jsonplaceholder.typicode.com/albums
- https://jsonplaceholder.typicode.com/photos

Each of those internal endpoints also support individual `GET` operations, for example you can consume https://jsonplaceholder.typicode.com/users/1 to get the information where `userID` = 1.

Problem 1.1: **The data enrichment** *(20 points)*

Complete this section using **JavaScript** or **TypeScript**.

Given the latency existing on mobile networks, doing multiple API calls from the mobile app is not an option, so you've been tasked with the creation of an external facing API endpoint which will provide the photo information but will enrich it with the information of the album it belongs to, and the User, all in a single API call.

For example, if you receive a request for `/externalapi/photos/1`, the expectation is to return the below **JSON** which includes all the information as shown below:


```
{
  "id": 1,
  "title": "accusamus beatae ad facilis cum similique qui sunt",
  "url": "https://via.placeholder.com/600/92c952",
  "thumbnailUrl": "https://via.placeholder.com/150/92c952",
  "album": {
    "id": 1,
    "title": "quidem molestiae enim",
    "user": {
      "id": 1,
      "name": "Leanne Graham",
      "username": "Bret",
      "email": "Sincere@april.biz",
      "address": {
        "street": "Kulas Light",
        "suite": "Apt. 556",
        "city": "Gwenborough",
        "zipcode": "92998-3874",
        "geo": {
          "lat": "-37.3159",
          "lng": "81.1496"
        }
      },
      "phone": "1-770-736-8031 x56442",
      "website": "hildegard.org",
      "company": {
        "name": "Romaguera-Crona",
        "catchPhrase": "Multi-layered client-server neural-net",
        "bs": "harness real-time e-markets"
      }
    }
  }
}
```


Problem 1.2: **The filtering** *(35 points)*

In addition to the data enrichment, we also need to be able to filter the photos information, so we want to include the below three filtering options:

- `title` should be **contains**
- `album.title` should be **contains**
- `album.user.email` should be an **equals**

First example, if you receive a request for `/externalapi/photos?title=repudiandae%20iusto`, the expectation is to return the 4 photos that contain **repudiandae iusto** in its `title`, which are the ones with IDs 13, 260, 318, 577.

Second example, if you receive a request for `/externalapi/photos?album.title=quidem`, the expectation is to return the 100 photos that belong to albums 1 and 79, because those are the only ones containing the word **quidem** in their `album.title`.

Third example, if you receive a request for `/externalapi/photos?album.user.email=Sincere@april.biz`, the expectation is to return the 500 photos that belong to albums 1 to 10, which are related to the user where `album.user.email` is **Sincere@april.biz**.

Fourth example, more than one filter can be applied in a single request, so you can receive a request like `/externalapi/photos?album.title=quidem&title=repudiandae%20iusto` and you should only return photo ID 13 which is the only record that matches both filters.

For all the scenarios above, they should still follow the data enrichment requested on **Section 1.1**.

Problem 1.3: **The pagination** *(10 points)*

Exposing a collection of resources through a single endpoint can lead to applications fetching large amounts of data when only a subset of the information is required. Clearly this process is highly inefficient, it wastes network bandwidth and processing power on the server hosting the web API.

You should modify the external API to limit the amount of data returned by any single request. Consider supporting query strings that specify the maximum number of items to retrieve (`limit`) and a starting offset into the collection (`offset`).

For example, if you receive a request `/externalapi/photos?album.title=quidem&limit=10&offset=50` you should only return 10 records starting from the item in position 50.

In case the `limit` parameter is not received, you should default it to `25`, and in case the `offset` parameter is not received, you should default it to `0`.

Part 2: **The Web Application** *(35 points)*

Now that you've completed the creation of the external API endpoints, your next task is to help the team to create the web site for **MetaPhoto**. The development team has heard about the concept of Single Page Applications (SPA) but has no experience building them, so they have asked you to create a Minimum Viable Product (MVP) using a SPA framework.

You need to create an application that allows the user to visualize the photos and its existing metadata (album and user information) and navigate through them on a paginated fashion using the (`offset`) implemented on **Section 1.3**.

The user should be allowed to apply any of the filtering options created in **Section 1.2** and be able to change the page size (`limit`) implemented in **Section 1.3**.

Our focus is not the look and feel of the application, we'll prioritize the correct implementation of the requirements above over a good looking UI.

We recommend the use of React but you can use any other SPA framework you feel comfortable with, like Angular or Vue.

Part 3: **Extra Points**

We'll grant some extra points if you complete some of the below tasks:

- Publish your project to a cloud provider of your preference and provide us with a URL to test the web application *(20 points)*
- Implement a CI/CD pipeline to deploy your project to the cloud provider *(20 points)*