LudusCristaltec Fullstack Challenge

This challenge consists of two exercises that target different important aspects of a good API/Web Service.

GENERAL GUIDELINES:

Every call must be validated (check if the required arguments are present and if their value is in the expected domain)
All calls should send a json body when appropriate and should return the correct http code matching the operation result as defined by the REST pattern (https://blog.stoplight.io/api-design-patterns-for-rest-web-services).
The response’s body is open to be defined by you.

1. CAT API PROXY

In the first one you will set up a nodejs (you can pick javascript or typescript) server application that should be able to receive external calls and fetch information from an external api - https://cataas.com/ - check the Get JSON data table.

The service will have 3 endpoints:

● GET - /api/v1/tags
This endpoint should fetch the tags from the target api and answer them. it works as a pure request proxy.
● GET - /api/v1/cats/filter?tag={{filtertag}}&omit={{number}}&total={{number}}
This endpoint should return information matching the filter tag.
all three fields for the url query parameters must be enforced and validated (requests that do not feature **tag**, **omit** and **total** parameters should return the appropriate error and reference the error).
● GET - api/v1/cats/match?string={{substr}}
This endpoint should return information about cats in a structured format. The string key is mandatory.
The substr value will be used to find tag names that have the substring in their name.
Example: if substr = br, information about cats that have bread,brazil,brown,abroad… should be compiled in the response. you can decide the structure of the response.
In order to test your solution, a postman collection should be provided as well as instructions to launch your project (https://www.postman.com/downloads/)
Bonus: The project should contain a Dockerfile in order to generate a docker image of your project. Please provide a sh or bat script to perform the build and launch the container.

2. CAT AS A SERVICE FRONTEND

In the second exercise you will set up a frontend application using VueJS that will use the API build in the first exercise; the application is a Cat Image Search Page featuring with two main sections.

Search Section. A search input where you can type your search query. Auto complete with tags is recommended to have valid results.

Results Section. Where the cats matching the query will be displayed. This section must be paginated with up to 10 cat images per page. On the top of Results Section display suggested alternate searches based on the tags present in the results found.

Your application will use an DBMS (MySQL or MariaDB) to store search information. Raw SQL queries are discouraged: you should try to use an ORM (sequelize is advisable). The goal is to store search queries and search results.

The database structure should be generated using the migrations mechanism provided by your ORM of choice and, if needed, a seeder mechanism too.

Create a Statistical page where the top search queries of the Application and the top found categories of cats are shown. Clicking any saved search query will display the current search results. Clicking any category will display cats matching that category.

Bonus: Clicking any saved search must show the difference, if any, of results found now versus saved results. Newfound cats that were not present in the saved results must highlight that they are new. Cats that are no longer available are to be aggregated and shown in a dedicated section. You can manipulate the stored results in order to achieve this bonus; hint use a doctored seed to achieve this.

Bonus: The project should contain a Dockerfile in order to generate a docker image of your project. Please provide a sh or bat script to perform the build and launch the container.

TIPS

- Consider using .env files to keep your app configuration outside your code.
- Although not necessary, it is highly advisable to implement at least unit tests.
- You can start by using docker but since it's not mandatory, you should focus first on the coding part of the challenge.
- Structuring your code usually follows one of two principles:
    - You can structure your code around functional tasks (controller, router, model).
    - You can structure your code around the model (a folder for cats and inside it, a file for a router, a controller and a model).
    - Both approaches are valid, figure the one that fits you better and makes sense according to the requirements.

For any doubt or inquiries please reach out to:
pedro.reis@ludustechnology.com