VIMOND

Node Test Case

The current exercise aims to evaluate your knowledge of NodeJS as a backend language in a pretty common use-case for Vimond: create a simple Rest API that works as a middle-layer between a client and another Rest API.

You can install all the npm modules you need, but we will look more favourable on a solution that uses as many native Node packages as possible. The way you structure your program will also be considered (we love readable and meaningful folder trees :-).

Think also about incorrect cases: Incorrect type in query parameters, null values, XSS injection, etc., as if your program will be in production tomorrow.

The case can be delivered via your GitHub account or any other delivery method of your choice, including the package.json, sources of your application and README (so excluding node_modules).

The exercise

1. Create a concurrent HTTP server that listens on port 8040.

2. Create an endpoint “GET /ping” that returns “pong!”. This is the health-check.

3. Create an endpoint “GET /version” that returns the Node version in use.

4. Create an endpoint “GET /images” that fetches the data from this other endpoint: https://jsonplaceholder.typicode.com/photos

    4.1. Create a query parameter “?size=<number>” that lets a user specify how many objects will be returned.

    4.2. Create a query parameter “?offset=<number>” that lets a user select an offset of the previous size (i.e. “?size=2&offset=5” would show images 11 and 12). By default, the offset is 0.

5. Create an endpoint “GET /Nicholas” where you return an aggregation of the data for userId 8 along with all his posts, wrapping them in “user” and “posts” respectively. You get this data from https://jsonplaceholder.typicode.com/users and https://jsonplaceholder.typicode.com/posts.

6. Create an endpoint “GET /Romaguera” where you return all the posts created by users that work for Romaguera group (tip: there’s more than one company in the group).

7. Create an endpoint “POST /todo” that creates a new TODO. You will receive JSON from the client and you will forward it to the API https://jsonplaceholder.typicode.com/todos in UTF-8 format with the following body:

```
{
      “userId”: <number>,
      “title”: <string>,
      “completed”: <boolean>
}
```

   If your POST is correct, you’ll be getting the same JSON you sent as response with the id of the new TODO (201).

   Don’t expect the TODO to be visible in https://jsonplaceholder.typicode.com/todos -API after you POST, simply assume it is correct if you get the response mentioned above.

8. Create an endpoint “GET /sorted-users” where you return the users on https://jsonplaceholder.typicode.com/users sorted alphabetically by their cities, filtering those which their websites domains are not “.com”, “.net” or “.org”.

9. Return a graceful 404 message when trying to access your API outside the previously described endpoints.

10. Document your endpoint in a README file so other developers can make use of it.

BONUS:

11. Add 10 seconds cache to all the previous GET endpoints.

12. Store the TODOs of question number 7 in the filesystem / in-memory, making them accessible as JSON via “GET /new-todos”.

If you have access issues with Typicode API, please contact Kine Lunde at kine@vimond.com / +47-47383588 (sms).