Basic REST API

Overview

The assignment consists of the following parts:

1. Implement a REST API that supports basic create, read, update, delete operations on arbitrary JSON objects. When an object is created, it's assigned a unique identifier. All subsequent operations (read, update, delete) are performed using this unique identifier. The REST API is to be implemented per specification outlined in this document. Write production quality code, along with any automated tests you feel are necessary. You may use any language/stack/technology you like to complete this assignment. Once complete, email us the source code along with instructions on how to get the application running. Be sure to submit your build scripts, automated tests, etc. along with the code.
2. Deploy the resulting application on any public cloud and email us the URL. For example, you can sign up for and use free tier services on Heroku (https://www.heroku.com/) or AWS (https://aws.amazon.com/).
3. Tell us how you think we can further enhance this API. For example, tell us what features you'd add next.

We realize that it might be the first time you're attempting to do anything like this and that you might need to use unfamiliar technologies. A lot of what our team does on a daily basis involves learning new technologies, concepts and development techniques and applying them to our product. This test is designed to demonstrate that you can learn new things and can write great code. Do your best and don't give up.

API Specifications

HTTP POST

Request

HTTP POST to the "api/objects" URL creates an object. Your service must assign a unique identifier to the JSON object and return the original object, along with the unique identifier. Executing a POST request twice should result in creating two object records with distinct uid's but same fields (e.g. firstName, lastName, etc.). The JSON object below is just an example. Your code must support arbitrary JSON objects.

HTTP POST api/objects

```
{
    "firstName": "Andrey",
    "lastName": "Kolmogorov",
    "dob": "25 April 1903"
}
```

Response

Response

```
{
    "uid": "2ae1e37799224920a61dd8e5c3dfde39",
    "firstName": "Andrey",
    "lastName": "Kolmogorov",
    "dob": "25 April 1903"
}
```

HTTP PUT

Request

HTTP PUT to the "api/objects/<uid>" URL updates the object. The update is a full replacement of the original object. The new object doesn't have to look anything like the original object (e.g. new fields may be added, existing fields may be removed). The service returns a copy of the new object. This operation must be idempotent.

HTTP PUT api/objects/2ae1e37799224920a61dd8e5c3dfde39

```
{
    "uid": "2ae1e37799224920a61dd8e5c3dfde39",
    "firstName": "Andrey",
    "lastName": "Kolmogorov",
    "dob": "19030425",
    "dod": "19871020"
}
```

Response

Response

```
{
    "uid": "2ae1e37799224920a61dd8e5c3dfde39",
    "firstName": "Andrey",
    "lastName": "Kolmogorov",
    "dob": "19030425",
    "dod": "19871020"
}
```

HTTP GET

Request

HTTP GET to the "api/objects/<uid>" URL returns the full JSON object.

HTTP GET api/objects/2ae1e37799224920a61dd8e5c3dfde39

Response

Response

```
{
    "uid": "2ae1e37799224920a61dd8e5c3dfde39",
    "firstName": "Andrey",
    "lastName": "Kolmogorov",
    "dob": "19030425",
    "dod": "19871020"
}
```

Request

HTTP GET to the "api/objects/" URL returns a list of unique identifiers of all JSON objects.

HTTP GET api/objects

Response

Response

```
[
   {"url":
"https://myrestapp.cisco.com/api/objects/2ae1e37799224920a61dd8e5c3dfde3
9"},
   {"url":
"https://myrestapp.cisco.com/api/objects/5f52eaa9d06443ff9b67db601c82569
f"},
   {"url":
"https://myrestapp.cisco.com/api/objects/5bbc4d40bdfd4b34ae3404c24a07a7a
6"}
]
```

HTTP DELETE

Request

HTTP DELETE to the "api/objects/<uid>" URL deletes the JSON object. Nothing is returned. This operation must be idempotent.

HTTP DELETE api/objects/2ae1e37799224920a61dd8e5c3dfde39

Response

Response

Errors

Operations can return errors. For example, POSTing a malformed object (not a JSON object) should return an error. Errors must follow the format outlined below. The exact error handling is left as an exercise to the reader.

Response

Response


```
{
   "verb": "POST",
   "url": "https://myrestapp.cisco.com/api/objects/",
   "message": "Not a JSON object"
}
```