CMPUT 401: Assignment

(Winter 2024)

GitHub Classroom assignment invitation link: https://classroom.github.com/a/_8ovbHol
Click the link above to create a repository that you will use for this assignment.

The goal of this assignment is to get you acquainted with the technical aspects of the web app development process.

In this assignment, you will build a simple REST API and deploy it to a Cybera Rapid Access Cloud instance.

This assignment will also get you acquainted with GitHub basics.

You are free to choose **any** programming language, and use **any** frameworks, packages, modules, libraries, and **any** database management system (SQL or NoSQL).

You will develop a REST API that manages employee information with the following structure of endpoints.

Endpoints

You must strictly follow the request and response structure for all endpoints.

Pay careful attention to the spelling of keys and endpoint addresses.

`id` can be an integer or a string (so, if you use MongoDB, you may use its id field).

List all Employees

`http://[ipv6 address]/employees (GET)`

Sample response structure (**Employees are sorted by `years_with_company` in descending order**).

```
{
  "total": 3,
  "average_years_with_company": 9.33,
  "average_salary": 100000,
  "employees": [
    {
```

```
      "id": 1,
      "name": "Bob Doe",
      "years_with_company": 9.5,
      "department": "HR",
      "salary": 100000
    },
    {
      "id": 2,
      "name": "Leeroy Jenkins",
      "years_with_company": 9.5,
      "department": "IT",
      "salary": 100000
    },
    {
      "id": 3,
      "name": "Rick Astley",
      "years_with_company": 9.0,
      "department": "Marketing",
      "salary": 100000
    }
  ]
}
```

Retrieve individual employee details

`http://[ipv6 address]/employees/{id} (GET)`

Sample response structure (for `http://[ipv6 address]/employees/2`):

```
{
    "id": 2,
    "name": "Bob Doe",
    "job_title": "Recruitment Manager",
    "department": "HR",
    "years_with_company": 9.5,
    "salary": 100000
}
```

Add an Employee

`http://[ipv6 address]/employees (POST)`

Sample request structure (all fields are mandatory):

```
{
    "name": "Bob Doe",
    "job_title": "Recruitment Manager",
    "department": "HR",
    "years_with_company": 9.5,
    "salary": 100000
}
```

Response structure (if successful), where `{id}` is the id of the new employee:

```
{
  "status": 200,
  "message": "New employee added",
  "id": {id}
}
```

(Pay careful attention to the message spelling and text-case)

Update an Employee

`http://[ipv6 address]/employees/{id} (PUT)`

Sample request structure (**all** fields must be supplied):

```
{
    "name": "Bob Doe",
    "job_title": "Chief Hiring Manager",
    "department": "Executive",
    "years_with_company": 20,
    "salary": 200000
}
```

Response structure (if successful):

```
{
  "status": 200,
  "message": "Employee updated"
}
```

(Pay careful attention to the message spelling and text-case)

Modify an Employee

`http://[ipv6 address]/employees/{id} (PATCH)`

Sample request structure (**any** field(s) of the PUT method can be supplied):

```
{
   "salary": 0
}
```

Response structure (if successful):

```
{
  "status": 200,
  "message": "Employee modified"
}
```

(Pay careful attention to the message spelling and text-case)

Delete an Employee

`http://[ipv6 address]/employees/{id} (DELETE)`

Request: empty

Response structure (if successful):

```
{
  "status": 200,
  "message": "Employee deleted"
}
```

(Pay careful attention to the message spelling and text-case)

HTTP Status Codes

All endpoints must return correct HTTP response codes, including at least the following codes:

- **200 (OK)** if a request was successfully completed
- **404 (Not Found)** if the requested resource was not found
- **405 (Method Not Allowed)** if the HTTP method was not supported by the resource

Authentication

No authentication is needed.

API Documentation

Use your framework/library of choice to auto-generate API documentation using the OpenAPI standard.

The API documentation must be available at **`http://[ipv6 address]/docs`**.

The API documentation must be valid OpenAPI Spec.

Deployment

The system **must** be deployed on your personal Cybera RAC instance (**not** the instance you have for your group project).

Many web frameworks include some kind of development server that you can use to run your app during development quickly. For example, in Django, you can easily run your app using

```
python manage.py runserver
```

Do **NOT** use a development server for deployment (e.g., if you use Django REST Framework, you can **not** run it on your server using the command above). This way is great for development purposes, but it is a horrible idea to do it in production for several reasons:

1. It is not secure
2. It can not handle multiple requests well (i.e., it's VERY slow)
3. You would need to manually restart your server if something happens (your operating system can't take care of this),

You need to **deploy** your API. There are many ways to do it, and it depends on the framework you choose. Use a web server like **Nginx**, or **Apache**. You should find a tutorial for your framework.

Your server must be listening on port 80.

Deliverables

We are using GitHub Classroom for this assignment. You must have your code in the GitHub repository associated with the assignment.

Mandatory Files

In addition to your code, you must create four files (pay attention to file text-case and spelling) in your root directory.

1. **`cybera.txt`**, containing the IPv6 address of your personal Cybera instance with the deployed API. NOTE THAT THERE IS NO HTTP(S), it is purely the IPV6 address. Don’t include anything else in this file.

Example of the text file contents:

```
Unset
2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

2. **`README.md`**, containing brief information about the framework, programming language and DBMS that you used, each on a separate line (there should be 3 lines, don't include anything else in this file):
`Framework`
`Language`
`DBMS`

Example of **`README.md`:**

```
Unset
Rocket
Rust
MongoDB
```

`3.` **`LICENSE`**, containing an appropriate license for your code. Refer to these instructions: https://help.github.com/articles/adding-a-license-to-a-repository/

4. **REFERENCES.md** which contains links and text of the resources you used to build your project. The format should be a "*" for each reference point.

Example of **REFERENCES.md**:

```
Unset
* https://stackoverflow.com
* I prompted ChatGPT to ask "how do I open a GitHub issue"
```

How GitHub should Be Used

You will create an issue titled "Create an Employee API" (pay careful attention to spelling and the text of the tile).

In the issue you will mention “k—-n” with a message saying that you read the assignment specifications (https://github.blog/2011-03-23-mention-somebody-they-re-notified/) . THIS IS NOT THE SAME AS ASSIGNING SOMEBODY TO AN ISSUE.

You must work off a “dev” branch and merge your dev branch into your “main” branch with a pull request.

You must merge the pull request you created into your “main” branch with a phrase that closes the “Create an Employee API” issue you created (https://help.github.com/articles/closing-issues-using-keywords/). DO NOT delete the “dev” branch after merging the pull request and closing the issue.

Testing

You should emulate the example endpoints and use diff checking tools to test the API.

Marking Scheme

We will use automated tests to mark your submission. You are allowed to fix your issues and request 1 remarking following the due date within 10 days of the assignment due date.

22 marks total

- Passing of test cases **(10 marks [A1]-[A10])**
- Auto-generated API documentation accessible from **`http://[ipv6 address]/docs` (1 mark [A11])**
    - The source of the OpenAPI docs is accessible from is **`http://[ipv6 address]/api-spec.yaml`** and is valid OpenAPI Spec **(1 mark [A12])**
- Able to access Cybera server **(1 mark [A13])**
- Application is deployed on a web server such as Apache or Nginx **(1 mark [A14])**
- Create a GitHub issue titled “Create an Employee API” **(1 mark [A15])**
- Mention “k—-n” with a message saying that you read the assignment specifications (https://github.blog/2011-03-23-mention-somebody-they-re-notified/) in the issue **(1 mark [A16])**
- Merge the pull request you created into your “main” branch with a phrase that closes the “Create an Employee API” issue you created (https://help.github.com/articles/closing-issues-using-keywords/) **(1 mark [A17])**
- Create a github branch with the exact name “dev” **(1 mark [A18])**
- Create a **README.md** in the main branch according to specifications **(1 mark [A19])**
- Create a **cybera.txt** according to the specifications **(1 mark [A20])r**
- Create a an MIT or Apache 2.0 license file according to these instructions: https://help.github.com/articles/adding-a-license-to-a-repository/ **(1 mark [A21])**
- Create a **REFERENCES.md** according to the specifications **(1 mark [A22])**

Your REST API must be correctly deployed on Cybera for our tests to run. You will receive a 0 if your API is not deployed properly.

Your GitHub account must be linked to the GitHub classroom (choose your username when signing up).

Academic Integrity

This is an individual assignment. Please **do not** share your work with your classmates. We may use a plagiarism detection system to check for plagiarism.

You **may** (and should!) use online tutorials, Stack Overflow posts and other publicly available resources. You **may** (and should!) use AI tools like GitHub Copilot or Amazon CodeWhisperer. However, you should cite your sources in **REFERENCES.md**

Frameworks to Consider

Here are some suggestions if you're unsure which framework to use.

- Django REST Framework (Python)
- FastAPI (Python)
- Flask RESTful (Python)
- Express (Node.js / JavaScript)
- Laravel (PHP)
- ASP.NET Web API (C#)
- Spring and Hibernate (Java)
- Rocket (Rust)
- Rails (Ruby)
- Gin (Golang)

DBMS to Consider

- SQLite (SQL)
- MySQL / MariaDB (SQL)
- PostgreSQL (SQL)
- MongoDB (NoSQL)

Resources

You are encouraged to find tutorials online for your framework of choice.

For Cybera Rapid Access Cloud, please use the following:

- Cybera Tutorial (CMPUT 401, Winter 2024)
- Cybera Screencast

- Official Cybera RAC documentation