SMG GM Data Team - Data Engineer Python API Exercise

Maurizio Idini
08/05/2023

1) Introduction

This is a simple document that briefly describes the project.

The project is written in **Python 3.8**, using **Docker** container.
The api is written using **Flask** and the API documentation is written using **Flask-restx**.
The code is documented using **Docstring** and tested using **pytest**.

2) Project Description

The folder structure is

```
smg_business_case/
    requirements.txt
    Dockerfile
    app.py
    docker-compose.yml
    bin/
        test.up.sh
        down.sh
        exec.sh
        test.sh
        up.sh
    lib/
        dataencryption/
            DataEncryptor_test.py
            DataEncryptor.py
        cipher/
            Rot13Cipher_test.py
            Rot13Cipher.py
        storage/
            FakeStorageManager.py
            BigqueryStorageManager.py
            StorageManager.py
            FakeStorageManager_test.py
        data/
            sentences.json
```

The main folder contains

- `requirements.txt` with the libraries used in the project
- `Dockerfile` and `docker-compose.yml` for the Docker container
- `lib` that contains the code and unit tests
- `bin` folder with bash script useful to run docker environments
- `app.py` that contains Flask API code based on `openapi definition`

The `lib` code is composed by

- `dataencryption` code that perform read/write operations useful in the api
- `cipher` that perform encrypt/decrypt operations using `Rot13` Algorithm
- `storage` that performs read/write operations on storage
- `data` that contains, for project purpose, the json data file

The `storage` folder contains `BigqueryStorageManager` to perform read/write operations on Google BigQuery but, for the project purpose and to avoid to share a GCP account and project_id, there is also a `FakeStorageManager`, used inside API, that simply load the `sentences.json` file and read/write on it.

3) Run the code

You can run the code in two ways:

- using `python app.py`
- using Docker, running `./bin/up.sh`

You can also access to `test_env` Docker container, running `./bin/test.up.sh`