Representational State Transfer Application Programming Interface

For programs or applications to manipulate, query, update, delete, data in a structural format.

API with bunch of endpoints in commands or requests. Returns response based on request was given when being hit.

python -m venv .venv # build virtual environment for using pip

requirement.txt

aniso8601==8.0.0

click==7.1.2

Flask==1.1.2

Flask-RESTful==0.3.8

Flask-SQLAlchemy==2.4.3

itsdangerous==1.1.0

Jinja2==2.11.2

MarkupSafe==1.1.1

pytz==2020.1

six==1.15.0

SQLAlchemy==1.3.18

Werkzeug==1.0.1

pip install -r requirements.txt

pip freeze OR pip show Flask # check packages versions in current environment

pip list -local # If in a virtualenv that has global access, do not list # globally-installed packages

app = Flask(\_\_name\_\_) # create app by Flask

api = Api(app) # wrap app in API

if \_\_name\_\_ == "\_\_main\_\_"; # start server & flask application in debug

    app.run(debug=True) # mode (output & log, only in dev env.)

class HelloWorld(Resource): # Build a class HelloWorld inherited

    def get(self): # Resource. Override the get method and return

        return{"data": "Hello World"}

# “Hello World” when get request received

# should be return in json format

\*Every time we want to return some type of information from API, information must be serializable

api.add\_resource(HelloWorld, "/helloworld") # Put the recourse HelloWorld into

# api with request URL suffix as

# /helloworld

import requests # simple request-script to acquire

# data from class HelloWorld

BASE = "http://127.0.0.1:5000/"

response = requests.get(BASE + "helloworld")

print(response.json())

def post(self): # define post function in class

        return{"data": "Posted"} # HelloWorld to receive post # requests. Data should also be in

# json format (dict. In python)

Main.py

class HelloWorld(Resource):

    def get(self, name, test):

        return{"name": name, "test": test}

api.add\_resource(HelloWorld, "/helloworld/<string:name>/<int:test>")

Test.py

response = requests.get(BASE + "helloworld/marco/31")

print(response.json())

# Class Is modified to receive 2 variables from input URL (name in string and test in int respectively). When requests.get is assigned in test.py with correct URL, name=marco and test=31 will be acquired by the API and will be presented in the console in json format.

names = {

            "tim"   : {"age": 19, "gender": "male"},

            "bill"  : {"age": 70, "gender": "male"}

        }

# With a dictionary being built in main.py, the get request with specific name as input will return the exact name information according to the dictionary.

Main.py

from flask import Flask, request

videos = { # dict. To store data from put requests

        }

class Video(Resource): # Class video to handle requests from

    def get(self, video\_id ): # URL link. Get function output data

        return videos[video\_id] # from dict. videos

    def put(self, video\_id, ): # Put request store data into videos

        print(request.form['likes']) # in dict. format

        return{}

api.add\_resource(Video, "/video/<int:video\_id>") # URL suffix changes to int

test.py

response = requests.put(BASE + "video/1", {"likes": 10})

Assign a put request to input data “likes” with key value 10 in dict. Format