0x05. AirBnB clone - RESTful API

PythonBack-endAPIWebserverFlask

- By: Guillaume, CTO at Holberton School
- Weight: 2
- Project to be done in teams of 2 people (your team: Emmanuel Ahuron, Onyezoba Sarah Chibuzor)
- Project over - took place from Oct 20, 2022 6:00 AM to Oct 25, 2022 6:00 AM
- An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 28.0/267 mandatory & 1.0/52 optional
- **Altogether: 10.69%**
  - Mandatory: 10.49%
  - Optional: 1.92%
  - Calculation: 10.49% + (10.49% * 1.92%) == **10.69%**

Concepts

For this project, we expect you to look at these concepts:

- REST API
- AirBnB clone

Resources

Read or watch:

- **REST API** concept page
- Learn REST: A RESTful Tutorial
- Designing a RESTful API with Python and Flask
- HTTP access control (CORS)
- Flask cheatsheet
- What are Flask Blueprints, exactly?
- Flask
- Modular Applications with Blueprints
- Flask tests
- Flask-CORS
- AirBnB clone - RESTful API

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone, **without the help of Google**:

General

- What REST means
- What API means
- What CORS means
- What is an API
- What is a REST API
- What are other type of APIs
- Which is the HTTP method to retrieve resource(s)
- Which is the HTTP method to create a resource
- Which is the HTTP method to update resource
- Which is the HTTP method to delete resource
- How to request REST API

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

Python Scripts

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted/compiled on Ubuntu 14.04 LTS using `python3` (version 3.4.3)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/python3`
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `PEP 8` style (version 1.7)
- All your files must be executable
- The length of your files will be tested using `wc`
- All your modules should have documentation (`python3 -c 'print(__import__("my_module").__doc__)'`)
- All your classes should have documentation (`python3 -c 'print(__import__("my_module").MyClass.__doc__)'`)
- All your functions (inside and outside a class) should have documentation (`python3 -c 'print(__import__("my_module").my_function.__doc__)'` and `python3 -c 'print(__import__("my_module").MyClass.my_function.__doc__)'`)

- A documentation is not a simple word, it’s a real sentence explaining what’s the purpose of the module, class or method (the length of it will be verified)

Python Unit Tests

- Allowed editors: `vi`, `vim`, `emacs`
- All your files should end with a new line
- All your test files should be inside a folder `tests`
- You have to use the unittest module
- All your test files should be python files (extension: `.py`)
- All your test files and folders should start by `test_`
- Your file organization in the tests folder should be the same as your project: ex: for `models/base_model.py`, unit tests must be in: `tests/test_models/test_base_model.py`
- All your tests should be executed by using this command: `python3 -m unittest discover tests`
- You can also test file by file by using this command: `python3 -m unittest tests/test_models/test_base_model.py`
- We strongly encourage you to work together on test cases, so that you don’t miss any edge cases

GitHub

There should be one project repository per group. If you clone/fork/whatever a project repository with the same name before the second deadline, you risk a 0% score.

More Info

Install Flask

```
$ pip3 install Flask
```

Tasks

0. Restart from scratch!
mandatory

Score: 100.0% (*Checks completed: 100.0%*)

No no no! We are already too far in the project to restart everything.

But once again, let’s work on a new codebase.

For this project you will fork this codebase:

- Update the repository name to `AirBnB_clone_v3`
- Update the `README.md`:
  - Add yourself as an author of the project
  - Add new information about your new contribution
  - Make it better!
- If you're the owner of this codebase, create a new repository called `AirBnB_clone_v3` and copy over all files from `AirBnB_clone_v2`

Repo:

- GitHub repository: `AirBnB_clone_v3`

Done? Help Check your code QA Review

1. Never fail!
mandatory

Score: 69.23% (*Checks completed: 69.23%*)

Since the beginning we've been using the `unittest` module, but do you know why `unittests` are so important? Because when you add a new feature, you refactor a piece of code, etc… you want to be sure you didn't break anything.

At Holberton, we have a lot of tests, and they all pass! Just for the Intranet itself, there are:

- `5,213` assertions *(as of 08/20/2018)*
- `13,061` assertions *(as of 01/25/2021)*

The following requirements **must** be met for your project:

- all current tests must pass (don't delete them…)
- add new tests as much as you can (tests are mandatory for some tasks)

```
guillaume@ubuntu:~/AirBnB_v3$ python3 -m unittest discover tests 2>&1 | tail -1

OK

guillaume@ubuntu:~/AirBnB_v3$ HBNB_ENV=test HBNB_MYSQL_USER=hbnb_test HBNB_MYSQL_PWD=
hbnb_test_pwd HBNB_MYSQL_HOST=localhost HBNB_MYSQL_DB=hbnb_test_db HBNB_TYPE_STORAGE=
db python3 -m unittest discover tests 2>&1 /dev/null | tail -n 1

OK

guillaume@ubuntu:~/AirBnB_v3$
```

Repo:

- GitHub repository: `AirBnB_clone_v3`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

2. Improve storage

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Update `DBStorage` and `FileStorage`, adding two new methods. **All changes should be done in the branch `storage_get_count`**:

A method to retrieve one object:

- Prototype: `def get(self, cls, id):`
  - `cls`: class
  - `id`: string representing the object ID
- Returns the object based on the class and its ID, or `None` if not found

A method to count the number of objects in storage:

- Prototype: `def count(self, cls=None):`
  - `cls`: class (optional)
- Returns the number of objects in storage matching the given class. If no class is passed, returns the count of all objects in storage.

Don’t forget to add new tests for these 2 methods on each storage engine.

```
guillaume@ubuntu:~/AirBnB_v3$ cat test_get_count.py
#!/usr/bin/python3
""" Test .get() and .count() methods
"""
from models import storage
from models.state import State


print("All objects: {}".format(storage.count()))
print("State objects: {}".format(storage.count(State)))


first_state_id = list(storage.all(State).values())[0].id
print("First state: {}".format(storage.get(State, first_state_id)))


guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ HBNB_MYSQL_USER=hbnb_dev HBNB_MYSQL_PWD=hbnb_dev_pwd HB
NB_MYSQL_HOST=localhost HBNB_MYSQL_DB=hbnb_dev_db HBNB_TYPE_STORAGE=db ./test_get_cou
nt.py
```

```
All objects: 1013

State objects: 27

First state: [State] (f8d21261-3e79-4f5c-829a-99d7452cd73c) {'name': 'Colorado', 'upd
ated_at': datetime.datetime(2017, 3, 25, 2, 17, 6), 'created_at': datetime.datetime(2
017, 3, 25, 2, 17, 6), '_sa_instance_state': <sqlalchemy.orm.state.InstanceState obje
ct at 0x7fc0103a8e80>, 'id': 'f8d21261-3e79-4f5c-829a-99d7452cd73c'}

guillaume@ubuntu:~/AirBnB_v3$

guillaume@ubuntu:~/AirBnB_v3$ ./test_get_count.py

All objects: 19

State objects: 5

First state: [State] (af14c85b-172f-4474-8a30-d4ec21f9795e) {'updated_at': datetime.d
atetime(2017, 4, 13, 17, 10, 22, 378824), 'name': 'Arizona', 'id': 'af14c85b-172f-447
4-8a30-d4ec21f9795e', 'created_at': datetime.datetime(2017, 4, 13, 17, 10, 22, 378763
)}

guillaume@ubuntu:~/AirBnB_v3$
```

For this task, you **must** make a pull request on GitHub.com, and ask at least one of your peer to review and merge it.

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `models/engine/db_storage.py, models/engine/file_storage.py, tests/test_models/test_engine/test_db_storage.py, tests/test_models/test_engine/test_file_storage.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

3. Status of your API

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

It’s time to start your API!

Your first endpoint (route) will be to return the status of your API:

```
guillaume@ubuntu:~/AirBnB_v3$ HBNB_MYSQL_USER=hbnb_dev HBNB_MYSQL_PWD=hbnb_dev_pwd HB
NB_MYSQL_HOST=localhost HBNB_MYSQL_DB=hbnb_dev_db HBNB_TYPE_STORAGE=db HBNB_API_HOST=
0.0.0.0 HBNB_API_PORT=5000 python3 -m api.v1.app

 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

...
```

In another terminal:

```
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/status
```

```
{
  "status": "OK"
}
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET -s http://0.0.0.0:5000/api/v1/status -vvv 2
>&1 | grep Content-Type
< Content-Type: application/json
guillaume@ubuntu:~/AirBnB_v3$
```

Magic right? (No need to have a pretty rendered output, it's a JSON, only the structure is important)

Ok, let starts:

- Create a folder `api` at the root of the project with an empty file `__init__.py`
- Create a folder `v1` inside `api`:
  - create an empty file `__init__.py`
  - create a file `app.py`:
    - create a variable `app`, instance of `Flask`
    - import `storage` from `models`
    - import `app_views` from `api.v1.views`
    - register the blueprint `app_views` to your Flask instance `app`
    - declare a method to handle `@app.teardown_appcontext` that calls `storage.close()`
    - inside `if __name__ == "__main__":`, run your Flask server (variable `app`) with:
      - host = environment variable `HBNB_API_HOST` or `0.0.0.0` if not defined
      - port = environment variable `HBNB_API_PORT` or `5000` if not defined
      - `threaded=True`
- Create a folder `views` inside `v1`:
  - create a file `__init__.py`:
    - import `Blueprint` from `flask` doc
    - create a variable `app_views` which is an instance of `Blueprint` (url prefix must be `/api/v1`)
    - wildcard import of everything in the package `api.v1.views.index` => PEP8 will complain about it, don't worry, it's normal and this file (`v1/views/__init__.py`) won't be check.
  - create a file `index.py`
    - import `app_views` from `api.v1.views`
    - create a route `/status` on the object `app_views` that returns a JSON: `"status": "OK"` (see example)

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/__init__.py, api/v1/__init__.py, api/v1/views/__init__.py, api/v1/views/index.py, api/v1/app.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

4. Some stats?

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Create an endpoint that retrieves the number of each objects by type:

- In `api/v1/views/index.py`
- Route: `/api/v1/stats`
- You must use the newly added `count()` method from `storage`

```
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/stats
{
  "amenities": 47,
  "cities": 36,
  "places": 154,
  "reviews": 718,
  "states": 27,
  "users": 31
}
guillaume@ubuntu:~/AirBnB_v3$
```

(No need to have a pretty rendered output, it's a JSON, only the structure is important)

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/v1/views/index.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

5. Not found

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Designers are really creative when they have to design a "404 page", a "Not found"… 34 brilliantly designed 404 error pages

Today it's different, because you won't use HTML and CSS, but JSON!

In `api/v1/app.py`, create a handler for `404` errors that returns a JSON-formatted `404` status code response. The content should be: `"error": "Not found"`

```
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/nop
{
  "error": "Not found"
}
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/nop -vvv
*   Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> GET /api/v1/nop HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.51.0
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 404 NOT FOUND
< Content-Type: application/json
< Content-Length: 27
< Server: Werkzeug/0.12.1 Python/3.4.3
< Date: Fri, 14 Apr 2017 23:43:24 GMT
<
{
  "error": "Not found"
}
guillaume@ubuntu:~/AirBnB_v3$
```

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/v1/app.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

6. State
mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Create a new view for `State` objects that handles all default RESTFul API actions:

- In the file `api/v1/views/states.py`
- You must use `to_dict()` to retrieve an object into a valid JSON
- Update `api/v1/views/__init__.py` to import this new file

Retrieves the list of all `State` objects: `GET /api/v1/states`

Retrieves a `State` object: `GET /api/v1/states/<state_id>`

- If the `state_id` is not linked to any `State` object, raise a `404` error

Deletes a `State` object:: `DELETE /api/v1/states/<state_id>`

- If the `state_id` is not linked to any `State` object, raise a `404` error
- Returns an empty dictionary with the status code `200`

Creates a `State`: `POST /api/v1/states`

- You must use `request.get_json` from Flask to transform the HTTP body request to a dictionary
- If the HTTP body request is not valid JSON, raise a `400` error with the message `Not a JSON`
- If the dictionary doesn't contain the key `name`, raise a `400` error with the message `Missing name`
- Returns the new `State` with the status code `201`

Updates a `State` object: `PUT /api/v1/states/<state_id>`

- If the `state_id` is not linked to any `State` object, raise a `404` error
- You must use `request.get_json` from Flask to transform the HTTP body request to a dictionary
- If the HTTP body request is not valid JSON, raise a `400` error with the message `Not a JSON`
- Update the `State` object with all key-value pairs of the dictionary.
- Ignore keys: `id`, `created_at` and `updated_at`
- Returns the `State` object with the status code `200`

```
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/states/
[
  {
    "__class__": "State",
    "created_at": "2017-04-14T00:00:02",
    "id": "8f165686-c98d-46d9-87d9-d6059ade2d99",
```

```
    "name": "Louisiana",
    "updated_at": "2017-04-14T00:00:02"
  },
  {
    "__class__": "State",
    "created_at": "2017-04-14T16:21:42",
    "id": "1a9c29c7-e39c-4840-b5f9-74310b34f269",
    "name": "Arizona",
    "updated_at": "2017-04-14T16:21:42"
  },
...
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/states/8f165686-
c98d-46d9-87d9-d6059ade2d99
 {
  "__class__": "State",
  "created_at": "2017-04-14T00:00:02",
  "id": "8f165686-c98d-46d9-87d9-d6059ade2d99",
  "name": "Louisiana",
  "updated_at": "2017-04-14T00:00:02"
}
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X POST http://0.0.0.0:5000/api/v1/states/ -H "Con
tent-Type: application/json" -d '{"name": "California"}' -vvv
*   Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> POST /api/v1/states/ HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.51.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 22
>
```

```
* upload completely sent off: 22 out of 22 bytes
* HTTP 1.0, assume close after body
< HTTP/1.0 201 CREATED
< Content-Type: application/json
< Content-Length: 195
< Server: Werkzeug/0.12.1 Python/3.4.3
< Date: Sat, 15 Apr 2017 01:30:27 GMT
<
{
  "__class__": "State",
  "created_at": "2017-04-15T01:30:27.557877",
  "id": "feadaa73-9e4b-4514-905b-8253f36b46f6",
  "name": "California",
  "updated_at": "2017-04-15T01:30:27.558081"
}
* Curl_http_done: called premature == 0
* Closing connection 0
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X PUT http://0.0.0.0:5000/api/v1/states/feadaa73-9e4b-4514-905b-8253f36b46f6 -H "Content-Type: application/json" -d '{"name": "California is so cool"}'
{
  "__class__": "State",
  "created_at": "2017-04-15T01:30:28",
  "id": "feadaa73-9e4b-4514-905b-8253f36b46f6",
  "name": "California is so cool",
  "updated_at": "2017-04-15T01:51:08.044996"
}
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/states/feadaa73-9e4b-4514-905b-8253f36b46f6
{
  "__class__": "State",
  "created_at": "2017-04-15T01:30:28",
```

```
  "id": "feadaa73-9e4b-4514-905b-8253f36b46f6",
  "name": "California is so cool",
  "updated_at": "2017-04-15T01:51:08"
}
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X DELETE http://0.0.0.0:5000/api/v1/states/feadaa73-9e4b-4514-905b-8253f36b46f6
{}
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/states/feadaa73-9e4b-4514-905b-8253f36b46f6
{
  "error": "Not found"
}
guillaume@ubuntu:~/AirBnB_v3$
```

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/v1/views/states.py, api/v1/views/__init__.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

7. City

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Same as `State`, create a new view for `City` objects that handles all default RESTFul API actions:

- In the file `api/v1/views/cities.py`
- You must use `to_dict()` to serialize an object into valid JSON
- Update `api/v1/views/__init__.py` to import this new file

Retrieves the list of all `City` objects of a `State`: `GET /api/v1/states/<state_id>/cities`

- If the `state_id` is not linked to any `State` object, raise a `404` error

Retrieves a `City` object. : `GET /api/v1/cities/<city_id>`

- If the `city_id` is not linked to any `City` object, raise a `404` error

Deletes a `City` object: `DELETE /api/v1/cities/<city_id>`

- If the `city_id` is not linked to any `City` object, raise a `404` error
- Returns an empty dictionary with the status code `200`

Creates a `City`: `POST /api/v1/states/<state_id>/cities`

- You must use `request.get_json` from Flask to transform the HTTP body request to a dictionary
- If the `state_id` is not linked to any `State` object, raise a `404` error
- If the HTTP body request is not a valid JSON, raise a `400` error with the message `Not a JSON`
- If the dictionary doesn't contain the key `name`, raise a `400` error with the message `Missing name`
- Returns the new `City` with the status code `201`

Updates a `City` object: `PUT /api/v1/cities/<city_id>`

- If the `city_id` is not linked to any `City` object, raise a `404` error
- You must use `request.get_json` from Flask to transform the HTTP body request to a dictionary
- If the HTTP request body is not valid JSON, raise a `400` error with the message `Not a JSON`
- Update the `City` object with all key-value pairs of the dictionary
- Ignore keys: `id`, `state_id`, `created_at` and `updated_at`
- Returns the `City` object with the status code `200`

```
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/states/not_an_id
/cities/

{

  "error": "Not found"

}

guillaume@ubuntu:~/AirBnB_v3$

guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/states/2b9a4627-
8a9e-4f32-a752-9a84fa7f4efd/cities

[

  {

    "__class__": "City",

    "created_at": "2017-03-25T02:17:06",

    "id": "1da255c0-f023-4779-8134-2b1b40f87683",

    "name": "New Orleans",

    "state_id": "2b9a4627-8a9e-4f32-a752-9a84fa7f4efd",

    "updated_at": "2017-03-25T02:17:06"

  },

  {
```

```
    "__class__": "City",
    "created_at": "2017-03-25T02:17:06",
    "id": "45903748-fa39-4cd0-8a0b-c62bfe471702",
    "name": "Lafayette",
    "state_id": "2b9a4627-8a9e-4f32-a752-9a84fa7f4efd",
    "updated_at": "2017-03-25T02:17:06"
  },
  {
    "__class__": "City",
    "created_at": "2017-03-25T02:17:06",
    "id": "e4e40a6e-59ff-4b4f-ab72-d6d100201588",
    "name": "Baton rouge",
    "state_id": "2b9a4627-8a9e-4f32-a752-9a84fa7f4efd",
    "updated_at": "2017-03-25T02:17:06"
  }
]
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/cities/1da255c0-f023-4779-8134-2b1b40f87683
{
  "__class__": "City",
  "created_at": "2017-03-25T02:17:06",
  "id": "1da255c0-f023-4779-8134-2b1b40f87683",
  "name": "New Orleans",
  "state_id": "2b9a4627-8a9e-4f32-a752-9a84fa7f4efd",
  "updated_at": "2017-03-25T02:17:06"
}
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X POST http://0.0.0.0:5000/api/v1/states/2b9a4627-8a9e-4f32-a752-9a84fa7f4efd/cities -H "Content-Type: application/json" -d '{"name": "Alexandria"}' -vvv
*   Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
```

```
> POST /api/v1/states/2b9a4627-8a9e-4f32-a752-9a84fa7f4efd/cities/ HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.51.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 22
>
* upload completely sent off: 22 out of 22 bytes
* HTTP 1.0, assume close after body
< HTTP/1.0 201 CREATED
< Content-Type: application/json
< Content-Length: 249
< Server: Werkzeug/0.12.1 Python/3.4.3
< Date: Sun, 16 Apr 2017 03:14:05 GMT
<
{
  "__class__": "City",
  "created_at": "2017-04-16T03:14:05.655490",
  "id": "b75ae104-a8a3-475e-bf74-ab0a066ca2af",
  "name": "Alexandria",
  "state_id": "2b9a4627-8a9e-4f32-a752-9a84fa7f4efd",
  "updated_at": "2017-04-16T03:14:05.655748"
}
* Curl_http_done: called premature == 0
* Closing connection 0
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X PUT http://0.0.0.0:5000/api/v1/cities/b75ae104-a8a3-475e-bf74-ab0a066ca2af -H "Content-Type: application/json" -d '{"name": "Bossier City"}'
{
  "__class__": "City",
  "created_at": "2017-04-16T03:14:06",
  "id": "b75ae104-a8a3-475e-bf74-ab0a066ca2af",
  "name": "Bossier City",
```

```
  "state_id": "2b9a4627-8a9e-4f32-a752-9a84fa7f4efd",
  "updated_at": "2017-04-16T03:15:12.895894"
}
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/cities/b75ae104-a8a3-475e-bf74-ab0a066ca2af
{
  "__class__": "City",
  "created_at": "2017-04-16T03:14:06",
  "id": "b75ae104-a8a3-475e-bf74-ab0a066ca2af",
  "name": "Bossier City",
  "state_id": "2b9a4627-8a9e-4f32-a752-9a84fa7f4efd",
  "updated_at": "2017-04-16T03:15:13"
}
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X DELETE http://0.0.0.0:5000/api/v1/cities/b75ae104-a8a3-475e-bf74-ab0a066ca2af
{}
guillaume@ubuntu:~/AirBnB_v3$
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/cities/b75ae104-a8a3-475e-bf74-ab0a066ca2af
{
  "error": "Not found"
}
guillaume@ubuntu:~/AirBnB_v3$
```

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/v1/views/cities.py, api/v1/views/__init__.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

8. Amenity

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Create a new view for `Amenity` objects that handles all default RESTFul API actions:

- In the file `api/v1/views/amenities.py`
- You must use `to_dict()` to serialize an object into valid JSON
- Update `api/v1/views/__init__.py` to import this new file

Retrieves the list of all `Amenity` objects: `GET /api/v1/amenities`

Retrieves a `Amenity` object: `GET /api/v1/amenities/<amenity_id>`

- If the `amenity_id` is not linked to any `Amenity` object, raise a `404` error

Deletes a `Amenity` object:: `DELETE /api/v1/amenities/<amenity_id>`

- If the `amenity_id` is not linked to any `Amenity` object, raise a `404` error
- Returns an empty dictionary with the status code `200`

Creates a `Amenity`: `POST /api/v1/amenities`

- You must use `request.get_json` from Flask to transform the HTTP request to a dictionary
- If the HTTP request body is not valid JSON, raise a `400` error with the message `Not a JSON`
- If the dictionary doesn't contain the key `name`, raise a `400` error with the message `Missing name`
- Returns the new `Amenity` with the status code `201`

Updates a `Amenity` object: `PUT /api/v1/amenities/<amenity_id>`

- If the `amenity_id` is not linked to any `Amenity` object, raise a `404` error
- You must use `request.get_json` from Flask to transform the HTTP request to a dictionary
- If the HTTP request body is not valid JSON, raise a `400` error with the message `Not a JSON`
- Update the `Amenity` object with all key-value pairs of the dictionary
- Ignore keys: `id`, `created_at` and `updated_at`
- Returns the `Amenity` object with the status code `200`

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/v1/views/amenities.py, api/v1/views/__init__.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

9. User
mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Create a new view for `User` object that handles all default RESTFul API actions:

- In the file `api/v1/views/users.py`
- You must use `to_dict()` to retrieve an object into a valid JSON

- Update `api/v1/views/__init__.py` to import this new file

Retrieves the list of all `User` objects: `GET /api/v1/users`

Retrieves a `User` object: `GET /api/v1/users/<user_id>`

- If the `user_id` is not linked to any `User` object, raise a `404` error

Deletes a `User` object:: `DELETE /api/v1/users/<user_id>`

- If the `user_id` is not linked to any `User` object, raise a `404` error
- Returns an empty dictionary with the status code `200`

Creates a `User`: `POST /api/v1/users`

- You must use `request.get_json` from Flask to transform the HTTP body request to a dictionary
- If the HTTP body request is not valid JSON, raise a `400` error with the message `Not a JSON`
- If the dictionary doesn't contain the key `email`, raise a `400` error with the message `Missing email`
- If the dictionary doesn't contain the key `password`, raise a `400` error with the message `Missing password`
- Returns the new `User` with the status code `201`

Updates a `User` object: `PUT /api/v1/users/<user_id>`

- If the `user_id` is not linked to any `User` object, raise a `404` error
- You must use `request.get_json` from Flask to transform the HTTP body request to a dictionary
- If the HTTP body request is not valid JSON, raise a `400` error with the message `Not a JSON`
- Update the `User` object with all key-value pairs of the dictionary
- Ignore keys: `id`, `email`, `created_at` and `updated_at`
- Returns the `User` object with the status code `200`

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/v1/views/users.py, api/v1/views/__init__.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

10. Place

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Create a new view for `Place` objects that handles all default RESTFul API actions:

- In the file `api/v1/views/places.py`
- You must use `to_dict()` to retrieve an object into a valid JSON
- Update `api/v1/views/__init__.py` to import this new file

Retrieves the list of all `Place` objects of a `City`: `GET /api/v1/cities/<city_id>/places`

- If the `city_id` is not linked to any `City` object, raise a `404` error

Retrieves a `Place` object. : `GET /api/v1/places/<place_id>`

- If the `place_id` is not linked to any `Place` object, raise a `404` error

Deletes a `Place` object: `DELETE /api/v1/places/<place_id>`

- If the `place_id` is not linked to any `Place` object, raise a `404` error
- Returns an empty dictionary with the status code `200`

Creates a `Place`: `POST /api/v1/cities/<city_id>/places`

- You must use `request.get_json` from Flask to transform the HTTP request to a dictionary
- If the `city_id` is not linked to any `City` object, raise a `404` error
- If the HTTP request body is not valid JSON, raise a `400` error with the message `Not a JSON`
- If the dictionary doesn't contain the key `user_id`, raise a `400` error with the message `Missing user_id`
- If the `user_id` is not linked to any `User` object, raise a `404` error
- If the dictionary doesn't contain the key `name`, raise a `400` error with the message `Missing name`
- Returns the new `Place` with the status code `201`

Updates a `Place` object: `PUT /api/v1/places/<place_id>`

- If the `place_id` is not linked to any `Place` object, raise a `404` error
- You must use `request.get_json` from Flask to transform the HTTP request to a dictionary
- If the HTTP request body is not valid JSON, raise a `400` error with the message `Not a JSON`
- Update the `Place` object with all key-value pairs of the dictionary
- Ignore keys: `id`, `user_id`, `city_id`, `created_at` and `updated_at`
- Returns the `Place` object with the status code `200`

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/v1/views/places.py, api/v1/views/__init__.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

11. Reviews

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Create a new view for `Review` object that handles all default RESTFul API actions:

- In the file `api/v1/views/places_reviews.py`
- You must use `to_dict()` to retrieve an object into valid JSON
- Update `api/v1/views/__init__.py` to import this new file

Retrieves the list of all `Review` objects of a `Place`: `GET /api/v1/places/<place_id>/reviews`

- If the `place_id` is not linked to any `Place` object, raise a `404` error

Retrieves a `Review` object. : `GET /api/v1/reviews/<review_id>`

- If the `review_id` is not linked to any `Review` object, raise a `404` error

Deletes a `Review` object: `DELETE /api/v1/reviews/<review_id>`

- If the `review_id` is not linked to any `Review` object, raise a `404` error
- Returns an empty dictionary with the status code `200`

Creates a `Review`: `POST /api/v1/places/<place_id>/reviews`

- You must use `request.get_json` from Flask to transform the HTTP request to a dictionary
- If the `place_id` is not linked to any `Place` object, raise a `404` error
- If the HTTP body request is not valid JSON, raise a `400` error with the message `Not a JSON`
- If the dictionary doesn't contain the key `user_id`, raise a `400` error with the message `Missing user_id`
- If the `user_id` is not linked to any `User` object, raise a `404` error
- If the dictionary doesn't contain the key `text`, raise a `400` error with the message `Missing text`
- Returns the new `Review` with the status code `201`

Updates a `Review` object: `PUT /api/v1/reviews/<review_id>`

- If the `review_id` is not linked to any `Review` object, raise a `404` error
- You must use `request.get_json` from Flask to transform the HTTP request to a dictionary
- If the HTTP request body is not valid JSON, raise a `400` error with the message `Not a JSON`
- Update the `Review` object with all key-value pairs of the dictionary
- Ignore keys: `id`, `user_id`, `place_id`, `created_at` and `updated_at`
- Returns the `Review` object with the status code `200`

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/v1/views/places_reviews.py, api/v1/views/__init__.py`

Done? Help Check your code Ask for a new correction Get a sandbox QA Review

12. HTTP access control (CORS)

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

A resource makes a cross-origin HTTP request when it requests a resource from a different domain, or port, than the one the first resource itself serves.

Read the full definition here

Why do we need this?

Because you will soon start allowing a web client to make requests your API. If your API doesn't have a correct CORS setup, your web client won't be able to access your data.

With Flask, it's really easy, you will use the class `CORS` of the module `flask_cors`.

How to install it: `$ pip3 install flask_cors`

Update `api/v1/app.py` to create a `CORS` instance allowing: `/*` for `0.0.0.0`

You will update it later when you will deploy your API to production.

Now you can see this HTTP Response Header: `< Access-Control-Allow-Origin: 0.0.0.0`

```
guillaume@ubuntu:~/AirBnB_v3$ curl -X GET http://0.0.0.0:5000/api/v1/cities/1da255c0-f023-4779-8134-2b1b40f87683 -vvv
*   Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> GET /api/v1/states/2b9a4627-8a9e-4f32-a752-9a84fa7f4efd/cities/1da255c0-f023-4779-8134-2b1b40f87683 HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.51.0
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: application/json
< Access-Control-Allow-Origin: 0.0.0.0
< Content-Length: 236
< Server: Werkzeug/0.12.1 Python/3.4.3
< Date: Sun, 16 Apr 2017 04:20:13 GMT
<
```

```
{
  "__class__": "City",
  "created_at": "2017-03-25T02:17:06",
  "id": "1da255c0-f023-4779-8134-2b1b40f87683",
  "name": "New Orleans",
  "state_id": "2b9a4627-8a9e-4f32-a752-9a84fa7f4efd",
  "updated_at": "2017-03-25T02:17:06"
}
* Curl_http_done: called premature == 0
* Closing connection 0
guillaume@ubuntu:~/AirBnB_v3$
```

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `api/v1/app.py`