0x05. AirBnB clone - RESTful API

Python | Back-end | API | Webserver | Flask

Concepts

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

- REST API
- AirBnB clone

Resources

Read or watch:

- REST API concept page
- Introduction to REST API
- Designing a RESTful API with Python and Flask
- HTTP status codes (FR)
- Flask cheatsheet
- Flask Blueprints, exactly!
- Flask
- Modular Applications with Blueprints
- Flask tests
- Postman (FR)

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 is a resource
- What CRUD means
- What is an API
- What is a RESTful API
- How to create a RESTful API
- 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 looked to some up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to steal the "intentions of life or any kind(s)" 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 20.04 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 follow `pep8` style (version 1.7)
- All your files must be executable
- The length of your files will be tested with `wc`
- All your modules should have documentation (`python3 -c 'print(module.__doc__)'`), `pep8` (`python3 -c 'print(module.__doc__)'`)
- All your functions (inside and outside a class) should have documentation (`python3 -c 'print(module.MyClass.my_function.__doc__)'`) and `pep8`
- 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)

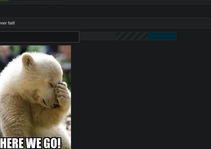
Python Unit Tests

- Allowed editors: `vi`, `vim`, `emacs`
- All your files should end with a new line
- All your files must be executable inside folder `tests`
- You have to use the `unittest` module
- All your test files should prefixed by `test_`
- 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 run tests for a specific module: `python3 -m unittest tests/test_models/test_base_model.py`
- The 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

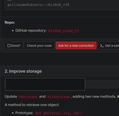


Install Flask

```
$ pip install Flask
```

Video library 2 total

Search by title



Python Flask RestAPI | AirBnB API

Tasks

0. Restart from scratch!

mandatory

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

But once again, let's serve our brain.

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
  - Add a TODO section
- If you fix 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`

Closed | Check your code | QA Review

1. Never fail

mandatory

Since the beginning we've been using the `cmd` module, but do you know why `cmd` exists, are so important? Because when you add a new feature, you sacrifice a piece of code, she... you need to be sure you don't break anything.

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

- `1`,`2`,`3` assertions (as of 08/30/2021)
- `1`,`2`,`3` assertions (as of 01/20/2021)

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

- all current tests must pass (don't delete them...)
- add new tests as you can (this is a very important step for some tasks)

```
guillaumebonnet@ubuntu:~$ python3 -m unittest discover tests 2>&1 | tail -1
OK
guillaumebonnet@ubuntu:~$ DBN_DBH=test DBN_STORE_TYPE=db_storage DBN_HBNB=test_hbnb DBN_HBNB_PW=test_hbnb_pwd DBN_HBNB_HOST=localhost DBN_HBNB_DB=test_hbnb_db DBN_TYPE_STORAGE=db python3 -m unittest discover tests 2>&1 |grep -v -E "^(DEBUG|INFO|ERROR)" | tail -1
OK
guillaumebonnet@ubuntu:~$
```

Repo:

- GitHub repository: `AirBnB_clone_v3`

Closed | Check your code | Ask for a new correction | 1_0_1_unittest | QA Review

2. Improve storage

mandatory

Update `models/engine/file_storage.py`, 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.

```
guillaumebonnet@ubuntu:~$ cat test_get_count.py
#!/usr/bin/python3
"""Test models get() method"""

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)))
guillaumebonnet@ubuntu:~$
guillaumebonnet@ubuntu:~$ DBN_DBH=test DBN_STORE_TYPE=db_storage DBN_HBNB=test_hbnb DBN_HBNB_PW=test_hbnb_pwd DBN_HBNB_HOST=localhost DBN_HBNB_DB=test_hbnb_db DBN_TYPE_STORAGE=db ./test_get_count.py
All objects: 8
State objects: 8
First state: [State] (f8b52b5-3e79-47c5-92a7-8b2d1c2f79c) {'name': 'California', 'updated_at': datetime.datetime(2017, 6, 15, 17, 5), 'id': 'f8b52b5-3e79-47c5-92a7-8b2d1c2f79c', '_sa_instance_state': <sqlalchemy.orm.state.InstanceState object at 0x7f3111ba8e0>, '__pl': 'f8b52b5-3e79-47c5-92a7-8b2d1c2f79c'}
guillaumebonnet@ubuntu:~$
guillaumebonnet@ubuntu:~$ cat test_count.py
All objects: 19
State objects: 8
First state: [State] (f8b52b5-3e79-47c5-92a7-8b2d1c2f79c) {'updated_at': datetime.datetime(2017, 6, 15, 17, 5), 'id': 'f8b52b5-3e79-47c5-92a7-8b2d1c2f79c', 'created_at': datetime.datetime(2017, 6, 15, 17, 5), 'name': 'California'}
guillaumebonnet@ubuntu:~$
```

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/file_storage.py`, `tests/test_models/test_engine/test_file_storage.py`, `tests/test_models/test_engine/test_file_storage.py`

Closed | Check your code | Ask for a new correction | 1_0_1_unittest | QA Review

3. Status of your API

mandatory

It's time to start your API

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

```
guillaumebonnet@ubuntu:~$lsof -i :5000 #Kill process if port 5000 is used
guillaumebonnet@ubuntu:~$ HBNB_MYSQL_USER=hbnb_dev HBNB_MYSQL_PWD=hbnb_dev_pwd HBNB_MYSQL_HOST=localhost HBNB_MYSQL_DB=hbnb_dev_db HBNB_TYPE_STORAGE=db HBNB_API_PORT=5000 HBNB_API_ENV=dev gunicorn -w 4 -p :5001 web_flask.0_hello_route:app --access-logfile - --error-logfile -
[2017-09-25 20:53:31 +0200] [1] [INFO] Starting gunicorn 19.7.1
[2017-09-25 20:53:31 +0200] [1] [INFO] Listening at: http://127.0.0.1:5000 (1)
[2017-09-25 20:53:31 +0200] [1] [INFO] Using worker: sync
[2017-09-25 20:53:31 +0200] [4] [INFO] Booting worker with pid: 4
[2017-09-25 20:53:31 +0200] [5] [INFO] Booting worker with pid: 5
[2017-09-25 20:53:31 +0200] [6] [INFO] Booting worker with pid: 6
[2017-09-25 20:53:31 +0200] [7] [INFO] Booting worker with pid: 7
^C[2017-09-25 20:53:33 +0200] [1] [INFO] Handling signal: int
[2017-09-25 20:53:33 +0200] [4] [INFO] Worker exiting (pid: 4)
[2017-09-25 20:53:33 +0200] [5] [INFO] Worker exiting (pid: 5)
[2017-09-25 20:53:33 +0200] [6] [INFO] Worker exiting (pid: 6)
[2017-09-25 20:53:33 +0200] [7] [INFO] Worker exiting (pid: 7)
[2017-09-25 20:53:33 +0200] [1] [INFO] Shutting down: Master
guillaumebonnet@ubuntu:~$
```

In another terminal:

```
guillaumebonnet@ubuntu:~$ curl -vH "Accept: application/text" http://127.0.0.1:5000/api/v1/status
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 5000 (#0)
> GET /api/v1/status HTTP/1.1
> Host: 127.0.0.1:5000
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: gunicorn/19.7.1
< Content-Type: application/text
< Content-Length: 2
< Connection: close
< Date: Fri, 29 Sep 2017 23:43:14 GMT
<
{
  "status": "OK"
}
* Closing connection 0
guillaumebonnet@ubuntu:~$
guillaumebonnet@ubuntu:~$ curl -vH "Accept: application/text" http://127.0.0.1:5000/api/v1/status
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 5000 (#0)
> GET /api/v1/status HTTP/1.1
> Host: 127.0.0.1:5000
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: gunicorn/19.7.1
< Content-Type: application/text
< Content-Length: 2
< Connection: close
< Date: Fri, 29 Sep 2017 23:43:14 GMT
<
{
  "status": "OK"
}
* Closing connection 0
guillaumebonnet@ubuntu:~$
```

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

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `web_flask/0_hello_route.py`

Closed | Check your code | 1_0_1_unittest | QA Review

4. Some stats?

mandatory

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

- URL: `/api/v1/stats`
- Route: `strict_slashes=False`
- You must use the newly added `count()` method from `models.storage`.

```
guillaumebonnet@ubuntu:~$lsof -i :5000 #Kill process if port 5000 is used
guillaumebonnet@ubuntu:~$ HBNB_MYSQL_USER=hbnb_dev HBNB_MYSQL_PWD=hbnb_dev_pwd HBNB_MYSQL_HOST=localhost HBNB_MYSQL_DB=hbnb_dev_db HBNB_TYPE_STORAGE=db HBNB_API_PORT=5000 HBNB_API_ENV=dev gunicorn -w 4 -p :5001 web_flask.0_hello_route:app --access-logfile - --error-logfile -
[2017-09-25 20:53:31 +0200] [1] [INFO] Starting gunicorn 19.7.1
[2017-09-25 20:53:31 +0200] [1] [INFO] Listening at: http://127.0.0.1:5000 (1)
[2017-09-25 20:53:31 +0200] [1] [INFO] Using worker: sync
[2017-09-25 20:53:31 +0200] [4] [INFO] Booting worker with pid: 4
[2017-09-25 20:53:31 +0200] [5] [INFO] Booting worker with pid: 5
[2017-09-25 20:53:31 +0200] [6] [INFO] Booting worker with pid: 6
[2017-09-25 20:53:31 +0200] [7] [INFO] Booting worker with pid: 7
^C[2017-09-25 20:53:33 +0200] [1] [INFO] Handling signal: int
[2017-09-25 20:53:33 +0200] [4] [INFO] Worker exiting (pid: 4)
[2017-09-25 20:53:33 +0200] [5] [INFO] Worker exiting (pid: 5)
[2017-09-25 20:53:33 +0200] [6] [INFO] Worker exiting (pid: 6)
[2017-09-25 20:53:33 +0200] [7] [INFO] Worker exiting (pid: 7)
[2017-09-25 20:53:33 +0200] [1] [INFO] Shutting down: Master
guillaumebonnet@ubuntu:~$
guillaumebonnet@ubuntu:~$ curl -vH "Accept: application/text" http://127.0.0.1:5000/api/v1/stats
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 5000 (#0)
> GET /api/v1/stats HTTP/1.1
> Host: 127.0.0.1:5000
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: gunicorn/19.7.1
< Content-Type: application/text
< Content-Length: 2
< Connection: close
< Date: Fri, 29 Sep 2017 23:43:14 GMT
<
{
  "amenities": 47,
  "cities": 14,
  "places": 24,
  "reviews": 114,
  "states": 7,
  "users": 7,
}
guillaumebonnet@ubuntu:~$
```

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

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `web_flask/1_hbnb_route.py`

Closed | Check your code | 1_0_1_unittest | QA Review

5. Not found

mandatory

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 `web_flask/0_hello_route.py`, create a handler for `404` errors that returns a JSON-formatted `404` status code response. The content should be: `{"error": "Not found"}`

```
guillaumebonnet@ubuntu:~$lsof -i :5000 #Kill process if port 5000 is used
guillaumebonnet@ubuntu:~$ HBNB_MYSQL_USER=hbnb_dev HBNB_MYSQL_PWD=hbnb_dev_pwd HBNB_MYSQL_HOST=localhost HBNB_MYSQL_DB=hbnb_dev_db HBNB_TYPE_STORAGE=db HBNB_API_PORT=5000 HBNB_API_ENV=dev gunicorn -w 4 -p :5001 web_flask.0_hello_route:app --access-logfile - --error-logfile -
[2017-09-25 20:53:31 +0200] [1] [INFO] Starting gunicorn 19.7.1
[2017-09-25 20:53:31 +0200] [1] [INFO] Listening at: http://127.0.0.1:5000 (1)
[2017-09-25 20:53:31 +0200] [1] [INFO] Using worker: sync
[2017-09-25 20:53:31 +0200] [4] [INFO] Booting worker with pid: 4
[2017-09-25 20:53:31 +0200] [5] [INFO] Booting worker with pid: 5
[2017-09-25 20:53:31 +0200] [6] [INFO] Booting worker with pid: 6
[2017-09-25 20:53:31 +0200] [7] [INFO] Booting worker with pid: 7
^C[2017-09-25 20:53:33 +0200] [1] [INFO] Handling signal: int
[2017-09-25 20:53:33 +0200] [4] [INFO] Worker exiting (pid: 4)
[2017-09-25 20:53:33 +0200] [5] [INFO] Worker exiting (pid: 5)
[2017-09-25 20:53:33 +0200] [6] [INFO] Worker exiting (pid: 6)
[2017-09-25 20:53:33 +0200] [7] [INFO] Worker exiting (pid: 7)
[2017-09-25 20:53:33 +0200] [1] [INFO] Shutting down: Master
guillaumebonnet@ubuntu:~$
guillaumebonnet@ubuntu:~$ curl -vH "Accept: application/text" http://127.0.0.1:5000/api/v1/nop
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 5000 (#0)
> GET /api/v1/nop HTTP/1.1
> Host: 127.0.0.1:5000
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 404 NOT FOUND
< Server: gunicorn/19.7.1
< Content-Type: application/text
< Content-Length: 21
< Connection: close
< Date: Fri, 29 Sep 2017 23:43:14 GMT
<
{
  "error": "Not found"
}
* Closing connection 0
guillaumebonnet@ubuntu:~$
```

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `web_flask/0_hello_route.py`

Closed | Check your code | 1_0_1_unittest | QA Review

6. State

mandatory

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

- In the file `web_flask/v1/views/__init__.py`
- You must use `from models import storage` to retrieve an object into a valid JSON
- Update `web_flask/v1/views/index.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>`

- You must use `models.storage.delete()`. 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 `state_id` is not linked to any `State` object, raise a `404`, error with the message `Not found`
- Return the new `State` with status code `200`

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

- You must use `models.storage.delete()`. 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 `state_id` is not linked to any `State` object, raise a `404`, error with the message `Not found`
- Return the new `State` with status code `200`

Repo:

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

Closed | Check your code | 1_0_1_unittest | QA Review

```
curl -X GET http://0.0.0.0:5000/api/v1/states/
curl -X GET http://0.0.0.0:5000/api/v1/states/d3f1bb6a-c98a-45d0-87d0-d6072a8e2
2b8
curl -X GET http://0.0.0.0:5000/api/v1/states/d3f1bb6a-c98a-45d0-87d0-d6072a8e2
2b8/cities
curl -X POST http://0.0.0.0:5000/api/v1/states/ -H "Content-Type: application/j
son" -d '{"name": "California"}'
{"id": "d3f1bb6a-c98a-45d0-87d0-d6072a8e2b8",
"name": "California",
"created_at": "2017-06-14T00:00:00.000000",
"updated_at": "2017-06-14T00:00:00.000000"}
curl -X PUT http://0.0.0.0:5000/api/v1/states/d3f1bb6a-c98a-45d0-87d0-d6072a8e
2b8 -H "Content-Type: application/json" -d '{"name": "California is on coke"}'
{"id": "d3f1bb6a-c98a-45d0-87d0-d6072a8e2b8",
"name": "California",
"created_at": "2017-06-14T00:00:00.000000",
"updated_at": "2017-06-14T00:00:00.000000"}
curl -X DELETE http://0.0.0.0:5000/api/v1/states/d3f1bb6a-c98a-45d0-87d0-d6072
a8e2b8
{}
curl -X GET http://0.0.0.0:5000/api/v1/states/d3f1bb6a-c98a-45d0-87d0-d6072a8e
2b8
{
"error": "Not Found"
}
curl -X GET http://0.0.0.0:5000/api/v1/states/
```

Repo:

- GitHub repository: **AirBnB_clone_v3**
- File: **models/state/test_api/test_index.py**

Clone | Check your code | Ask a mentor | Get Review

7. City

Mandatory

Same as **states**, create a new new file **cities** objects that handles all default RESTFul API actions:

- In the file **api/v1/views/cities.py**
- You must use **jsonify** 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: **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** object: **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 body request 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**, **created_at** and **updated_at**
- Returns the **City** object with the status code **200**

```
curl -X GET http://0.0.0.0:5000/api/v1/states/fed6a7b-9e8c-4514-9f8c-825f3f86d
ec1/cities
{"error": "Not found"}
curl -X POST http://0.0.0.0:5000/api/v1/states/fed6a7b-9e8c-4514-9f8c-825f3f86d
ec1/cities -H "Content-Type: application/json" -d '{"name": "Fremont"}'
{"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Fremont",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"}
curl -X GET http://0.0.0.0:5000/api/v1/states/fed6a7b-9e8c-4514-9f8c-825f3f86d
ec1/cities
[
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Fremont",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
},
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Marin County",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
},
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Marin County",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
}
]
curl -X GET http://0.0.0.0:5000/api/v1/states/2bc0d7f-9e8e-4f32-a7b1-9a84a7f8
4ef/cities -H "Content-Type: application/json" -d '{"name": "Alameda"}' <<<
{"error": "Not found"}
curl -X GET -v http://0.0.0.0:5000/api/v1/states/fed6a7b-9e8c-4514-9f8c-825f3
f86dec1/cities
* 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/fed6a7b-9e8c-4514-9f8c-825f3f86dec1/cities HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.58.0
> Accept: */*
< HTTP/1.0, assume close after body
< HTTP/1.1 200 OK
< Content-Type: application/json
< Content-Length: 2
< Server: Werkzeug/0.11.2 Python/3.4.3
< Date: Sat, 18 Mar 2017 01:19:07 GMT
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Fremont",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
},
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Marin County",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
},
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Marin County",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
}
]
curl -X PUT -v http://0.0.0.0:5000/api/v1/cities/7b4e4d7-4c4e-4f32-a7b1-9a84a
7f84ef -H "Content-Type: application/json" -d '{"name": "Fremont"}' <<<
{"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Fremont",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"}
curl -X DELETE http://0.0.0.0:5000/api/v1/cities/7b4e4d7-4c4e-4f32-a7b1-9a84a
7f84ef
{}
curl -X GET http://0.0.0.0:5000/api/v1/cities/7b4e4d7-4c4e-4f32-a7b1-9a84a7f8
4ef
{
"error": "Not found"
}
```

Repo:

- GitHub repository: **AirBnB_clone_v3**
- File: **models/state/test_api/test_index.py**

Clone | Check your code | Ask a mentor | Get Review

8. Amenity

Mandatory

Create a new new file **amenities** objects that handles all default RESTFul API actions:

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

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** object: **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**

```
curl -X GET -v http://0.0.0.0:5000/api/v1/states/2bc0d7f-9e8e-4f32-a7b1-9a84a7
84f1c/cities -H "Content-Type: application/json" -d '{"name": "Alameda"}' <<<
{"error": "Not found"}
curl -X GET -v http://0.0.0.0:5000/api/v1/states/2bc0d7f-9e8e-4f32-a7b1-9a84a7
84f1c/cities
* 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/2bc0d7f-9e8e-4f32-a7b1-9a84a7f84ef/cities HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.58.0
> Accept: */*
< HTTP/1.0, assume close after body
< HTTP/1.1 200 OK
< Content-Type: application/json
< Content-Length: 2
< Server: Werkzeug/0.11.2 Python/3.4.3
< Date: Sat, 18 Mar 2017 01:19:07 GMT
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Fremont",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
},
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Marin County",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
}
]
curl -X PUT -v http://0.0.0.0:5000/api/v1/cities/7b4e4d7-4c4e-4f32-a7b1-9a84a
7f84ef -H "Content-Type: application/json" -d '{"name": "Fremont"}' <<<
{"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Fremont",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"}
curl -X DELETE -v http://0.0.0.0:5000/api/v1/cities/7b4e4d7-4c4e-4f32-a7b1-9a
84a7f84ef
{}
curl -X GET -v http://0.0.0.0:5000/api/v1/cities/7b4e4d7-4c4e-4f32-a7b1-9a84a
7f84ef
{
"error": "Not found"
}
```

Repo:

- GitHub repository: **AirBnB_clone_v3**
- File: **models/state/test_api/test_index.py**

Clone | Check your code | Ask a mentor | Get Review

9. User

Mandatory

Create a new new file **users** objects that handles all default RESTFul API actions:

- In the file **api/v1/views/users.py**
- You must use **jsonify** to serialize 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** object: **POST /api/v1/users/**

- You must use **request.get_json** from Flask to transform the HTTP 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**, **created_at** and **updated_at**
- Returns the **User** object with the status code **200**

Repo:

- GitHub repository: **AirBnB_clone_v3**
- File: **models/amenity/test_api/test_index.py**

Clone | Check your code | Ask a mentor | Get Review

10. Place

Mandatory

Create a new new file **places** objects that handles all default RESTFul API actions:

- In the file **api/v1/views/places.py**
- You must use **jsonify** to serialize an object into a valid JSON
- Update **api/v1/views/__init__.py** to import this new file

Retrieves the list of all **Place** objects: **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** object: **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 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 **city_id** is not linked to any **City** 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 body request 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: **models/place/test_api/test_index.py**

Clone | Check your code | Ask a mentor | Get Review

11. Reviews

Mandatory

Create a new new file **reviews** object that handles all default RESTFul API actions:

- In the file **api/v1/views/reviews.py**
- You must use **jsonify** to serialize an object into JSON
- Update **api/v1/views/__init__.py** to import this new file

Retrieves the list of all **Review** objects: **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** object: **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 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 **place_id** is not linked to any **Place** 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 body request 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: **models/review/test_api/test_index.py**

Clone | Check your code | Ask a mentor | Get Review

12. HTTP access control (CORS)

Mandatory

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 we are using Flask allowing a web client to make requests your API, if your API doesn't have a correct CORS setup, your client will not be able to communicate with it.

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

```
$ sudo pip3 install flask_cors
```

Update **api/v1/app.py** to add the **Access-Control-Allow-Origin** header for **0.0.0.0**

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

Now you can test the HTTP Response Header: **Access-Control-Allow-Origin: 0.0.0.0**

```
curl -X GET -v http://0.0.0.0:5000/api/v1/states/
curl -X GET -v http://0.0.0.0:5000/api/v1/states/2bc0d7f-9e8e-4f32-a7b1-9a84a7
84f1c/cities
* 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/2bc0d7f-9e8e-4f32-a7b1-9a84a7f84ef/cities HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.58.0
> Accept: */*
< HTTP/1.0, assume close after body
< HTTP/1.1 200 OK
< Content-Type: application/json
< Access-Control-Allow-Origin: 0.0.0.0
< Content-Length: 2
< Server: Werkzeug/0.11.2 Python/3.4.3
< Date: Sat, 18 Mar 2017 01:19:07 GMT
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Fremont",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
},
{
"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Marin County",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"
}
]
curl -X PUT -v http://0.0.0.0:5000/api/v1/cities/7b4e4d7-4c4e-4f32-a7b1-9a84a
7f84ef -H "Content-Type: application/json" -d '{"name": "Fremont"}' <<<
{"id": "7b4e4d7-4c4e-4f32-a7b1-9a84a7f84ef",
"state_id": "fed6a7b-9e8c-4514-9f8c-825f3f86dec1",
"name": "Fremont",
"created_at": "2017-03-29T02:11:17.000000",
"updated_at": "2017-03-29T02:11:17.000000"}
curl -X DELETE -v http://0.0.0.0:5000/api/v1/cities/7b4e4d7-4c4e-4f32-a7b1-9a
84a7f84ef
{}
curl -X GET -v http://0.0.0.0:5000/api/v1/cities/7b4e4d7-4c4e-4f32-a7b1-9a84a
7f84ef
{
"error": "Not found"
}
```

Repo:

- GitHub repository: **AirBnB_clone_v3**
- File: **models/state/test_api/test_index.py**

Clone | Check your code | Ask a mentor | Get Review

```
  "_id": "5f0a0a0a0a0a0a0a0a0a0a0a",
  "name": "Nice Cozy Apartment",
  "type": "Apartment",
  "room_id": "5f0a0a0f-0a0e-0010-a71c-7e8b1a79b6b",
  "updated_at": "2017-11-07T00:00:00.000Z"
}

* Connection #0 to host demo.restfocus.com left intact
* Closing connection 0
pollicom@ubuntu:~/Bin$$ vi
```

Repos:

- GitHub repository: `AirBnB_clone_v3`
- File: `models/amenity.py`

Close | Check your code | Get a random | Ok Review

13. Place - Amenity

Advanced

Create a new view for the link between `Place` objects and `Amenity` objects that handles all default RESTful API actions:

- In the file `models/views/places_amenities.py`
- You must use `request.method` to retrieve an object into a valid JSON
- Update `app/views/__init__.py` to import this new file
- Depending of the storage:
  - `DBStorage`: list, create and delete `Amenity` objects from `Amenities` relationship
  - `FileStorage`: list, add and remove `Amenity` ID in the list `amenity_ids` of a `Place` object

Retrieves the list of all `Amenity` objects of a `Place` link `/places/<place_id>/amenities`

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

Deletes a `Amenity` object to a `Place` DELETE /places/<place_id>/amenities/<amenity_id>

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

Link a `Amenity` object to a `Place` POST /places/<place_id>/amenities/<amenity_id>

- No HTTP body needed
- If the `place_id` is not linked to any `Place` object, raise a `404` error
- If the `amenity_id` is not linked to any `Amenity` object, raise a `404` error
- If the `Amenity` is already linked to the `Place`, return the `Amenity` with the status code `200`
- Returns the `Amenity` with the status code `201`

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `models/views/places_amenities.py, app/views/__init__.py`

Close | Check your code | Get a random | Ok Review

14. Security improvements!

Advanced

Currently, the `User` object is designed to store the user password in cleartext.

It's crystal bad!

To avoid that, improve the `User` object:

- Update the method `to_dict()` of `models/user` to remove the `password` key **except when it's used by `FileStorage`** to **save data to disk**. Two default parameters:
- Each time a new `User` object is created or password updated, the password is hashed to a `MD5` value
- In the database for `DBStorage`, the password stored is now hashed to a MD5 value
- In the file for `FileStorage`, the password stored is now hashed to a MD5 value

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `models/base_model.py, models/user.py`

Close | Check your code | Get a random | Ok Review

15. Search

Advanced

For the moment, the only way to list `State` objects is via `GET /api/v1/states/?limit_100:0&page_1`

Good, but not enough...

Update `/api/v1/places_search` to add a new endpoint `POST /api/v1/places_search` that retrieves all `Place` objects depending of the JSON in the body of the request.

The JSON can contain 3 optional keys:

- `states`: list of `State` IDs
- `cities`: list of `City` IDs
- `amenities`: list of `Amenity` IDs

Search rules:

- If the HTTP request body is not valid JSON, raise a `400` error with the message `Not a JSON`
- If the JSON body is empty or each list of all keys are empty: return all `Place` objects
- If `states` list is not empty, results should include all `Place` objects for each `State` id listed
- If `cities` list is not empty, results should include all `Place` objects for each `City` id listed
- Keys `states` and `cities` are inclusive. Search results should include all `Place` objects in storage related to each `State` id and `City` id listed, plus every `Place` linked individually to each `amenities` id listed. `Place` was already included by `states`/`cities`:
  - Example:
    - States A has 3 cities A1 and A2
    - States B has 3 cities B1, B2 and B3
    - 1 place in A1
    - 2 places in A2
    - 4 in A3
    - 5 in B1
    - 6 in B2
    - 7 in B3
    - 8 in B4
    - 9 in B5

    Curl POST states A and cities = B2

    Result: all 4 places from the city B2 and the place from the city A1 and the 2 places of the city A2 (because they are part of State A) in 7 places returned
- If `amenities` list is not empty, limit search results to only `Place` objects having all `Amenity` IDs listed
- The key `amenities` must be ignoring as a filter on the results generated by `states` and `cities` or on all `Place` if `states` and `cities` are both empty or missing
- Results will only include `Place` objects having all listed `amenities`. If a `Place` doesn't have even one of these `amenities`, it won't be retrieved.

```
pollicom@ubuntu:~/Bin$$ vi curl -X POST http://0.0.0.0:5001/api/v1/places_search -d "{\"states\":[\"bb64df7c-b67a-4618-8ad9-f7d778b7fb9d\"], \"cities\":[\"84160a17-fd1-4f49-aae1-95d09fd18bcf\"]}"
[
  {
    "__class__": "Place",
    "created_at": "2017-03-25T02:17:14",
    "id": "1987f1a0-0f2d-4f2d-a7f2-f9f2d2d2d2d2",
    "name": "The Lake House",
    "city_id": "84160a17-fd1-4f49-aae1-95d09fd18bcf",
    "updated_at": "2017-03-25T02:17:14",
    "description": "Our place is 2 blocks from Stata Park (Farmer's Market), Historic Maumee Boulevard, and an ad 2 miles from Old Orchard where there is shopping, dining, and site seeing. We offer continental breakfast. You get the quiet life with great beaches and garden views. This is a 100+ year old city home which has been fully restored. Our place is 2 blocks from Stata Park (Farmer's Market), Historic Maumee Boulevard, and an ad 2 miles from Old Orchard where there is shopping, dining, and site seeing. We offer continental breakfast. Rooms are as little in a bedroom as have from 5am until 7am where we opt to turn. ONE TO TWO rooms some has two bedrooms, one king and one queen. There are 2 bedrooms, 1 (2017) stab tub with shower and 1 with shower in the hallway. Our place is 2 blocks from Stata Park (Farmer's Market), Historic Maumee Boulevard, and an ad 2 miles from Old Orchard where there is shopping, dining, and site seeing. We offer continental breakfast. You get the quiet life with great beaches and garden views. This is a 100+ year old city home which has been fully restored. Our place is 2 blocks from Stata Park (Farmer's Market), Historic Maumee Boulevard, and an ad 2 miles from Old Orchard where there is shopping, dining, and site seeing. We offer continental breakfast. Rooms are as little in a bedroom as have from 5am until 7am where we opt to turn. ONE TO TWO rooms some has two bedrooms, one king and one queen. There are 2 bedrooms, 1 (2017) stab tub with shower and 1 with shower in the hallway. Our place is 2 blocks from Stata Park (Farmer's Market), Historic Maumee Boulevard, and an ad 2 miles from Old Orchard where there is shopping, dining, and site seeing. We offer continental breakfast. Rooms are as little in a bedroom as have from 5am until 7am where we opt to turn. ONE TO TWO rooms some has two bedrooms, one king and one queen. There are 2 bedrooms, 1 (2017) stab tub with shower and 1 with shower in the hallway. Our place is 2 blocks from Stata Park (Farmer's Market), Historic Maumee Boulevard, and an ad 2 miles from Old Orchard where there is shopping, dining, and site seeing. We offer continental breakfast. Rooms are as little in a bedroom as have from 5am until 7am where we opt to turn. ONE TO TWO rooms some has two bedrooms, one king and one queen. There are 2 bedrooms, 1 (2017) stab tub with shower and 1 with shower in the hallway.",
    "number_rooms": 4,
    "number_bathrooms": 3,
    "max_guest": 8,
    "price_by_night": 42,
    "latitude": 41.3983,
    "longitude": -105.5747,
    "updated_at": "2017-03-25T02:17:14"
  },
  {
    "__class__": "Place",
    "created_at": "2017-03-25T02:17:14",
    "id": "1987f1a0-0f2d-4f2d-a7f2-f9f2d2d2d2d2",
    "name": "Cozy Apartment",
    "city_id": "84160a17-fd1-4f49-aae1-95d09fd18bcf",
    "updated_at": "2017-03-25T02:17:14",
    "description": "Our place is 2 blocks from Stata Park (Farmer's Market), Historic Maumee Boulevard, and an ad 2 miles from Old Orchard where there is shopping, dining, and site seeing. We offer continental breakfast. You get the quiet life with great beaches and garden views. This is a 100+ year old city home which has been fully restored. Our place is 2 blocks from Stata Park (Farmer's Market), Historic Maumee Boulevard, and an ad 2 miles from Old Orchard where there is shopping, dining, and site seeing. We offer continental breakfast. Rooms are as little in a bedroom as have from 5am until 7am where we opt to turn. ONE TO TWO rooms some has two bedrooms, one king and one queen. There are 2 bedrooms, 1 (2017) stab tub with shower and 1 with shower in the hallway. Our place is 2 blocks from Stata Park (Farmer's Market), Historic Maumee Boulevard, and an ad 2 miles from Old Orchard where there is shopping, dining, and site seeing. We offer continental breakfast. Rooms are as little in a bedroom as have from 5am until 7am where we opt to turn. ONE TO TWO rooms some has two bedrooms, one king and one queen. There are 2 bedrooms, 1 (2017) stab tub with shower and 1 with shower in the hallway.",
    "number_rooms": 1,
    "number_bathrooms": 1,
    "max_guest": 2,
    "price_by_night": 42,
    "latitude": 41.3983,
    "longitude": -84.0753,
    "updated_at": "2017-03-25T02:17:14"
  }
]
pollicom@ubuntu:~/Bin$$ vi
```

Repo:

- GitHub repository: `AirBnB_clone_v3`
- File: `models/places_search.py`

Close | Check your code | Get a random | Ok Review

Copyright © 2021 ALX. All rights reserved.