☆ Django: Github Dataset API

In this challenge, you are part of a team building a git event tracking platform. One requirement is for a REST API service to provide events information using the Python Django framework. You will need to add functionality to add and delete information as well as to perform some queries. You'll be dealing with typical information for git event data like repository, actor, event type, etc. The team has come up with a set of requirements including filtering and ordering requirements, response codes and error messages for the queries you must implement.

The definitions and a detailed requirements list follow. You will be graded on whether your application performs data retrieval and manipulation based on given use cases exactly as described in the requirements.

Each event data is a JSON entry with the following keys:

- `id` : This is the event unique ID.
- `type` : This is the event type.
- `actor` : The actor responsible for the event. The actor itself is a JSON entry consisting of following fields:
  - `id` : This is the actor unique ID.
  - `login` : This is the actor unique login ID.
  - `avatar_url` : This is the actor avatar URL.
- `repo` : The repository to which this event is associated with. The repo itself is a JSON entry consisting of following fields:
  - `id` : This is the repo unique ID.
  - `name` : This is the repo name.
  - `url` : This is the repo URL.
- `created_at` : This is the timestamp for the event creation given in the format `yyyy-MM-dd HH:mm:ss` . The timezone is `UTC +0` .

Sample JSON git event object

```
{
  "id":4055191679,
  "type":"PushEvent",
  "actor":{
    "id":2790311,
    "login":"daniel33",
```

```
      "name":"johnbolton/exercitationem",
      "url":"https://github.com/johnbolton/exercitationem"
    },
    "created_at":"2015-10-03 06:13:31"
  }
```

1

The *REST* service should implement the following functionalities:

1. *Erasing all the events*: The service should be able to erase all the events by the *DELETE* request at `/erase`. The *HTTP* response code should be *200*.
2. *Adding new events*: The service should be able to add a new event by the *POST* request at `/events`. The event *JSON* is sent in the request body. If an event with the same id already exists then the *HTTP* response code should be *400*, otherwise, the response code should be *201*.
3. *Returning all the events*: The service should be able to return the JSON array of all the events by the *GET* request at `/events`. The *HTTP* response code should be *200*. The JSON array should be sorted in ascending order by event ID.
4. *Returning the event records filtered by the actor ID*: The service should be able to return the *JSON* array of all the events which are performed by the actor ID by the *GET* request at `/events/actors/{actorID}`. If the requested actor does not exist then *HTTP* response code should be *404*, otherwise, the response code should be *200*. The JSON array should be sorted in ascending order by event ID.
5. *Updating the avatar URL of the actor*: The service should be able to update the avatar URL of the actor by the *PUT* request at `/actors`. The actor *JSON* is sent in the request body. If the actor with the id does not exist then the response code should be *404*, or if there are other fields being updated for the actor then the *HTTP* response code should be *400*, otherwise, the response code should be *200*.
6. *Returning the actor records ordered by the total number of events*: The service should be able to return the *JSON* array of all the actors sorted by the total number of associated events with each actor in descending order by the *GET* request at `/actors`. If there are more than one actors with the same number of events, then order them by the timestamp of the latest event in the descending order. If more than one actors have the same timestamp for the latest event, then order them by the alphabetical order of login. The *HTTP* response code should be *200*.
7. *Returning the actor records ordered by the maximum streak*: The service should be able to return the *JSON* array of all the actors sorted by the maximum streak (i.e., the total number of consecutive days actor has pushed an event to the system) in descending order by the *GET* request at `/actors/streak`. If there are more than one actors with the same maximum streak, then order them by the timestamp of the latest event in the descending order. If more than one actors have the same timestamp for the latest event, then order them by the alphabetical order of login. The *HTTP* response code should be *200*.

Sample Series of Requests

Requests are received in the following order and are provided in the test file *http00.json*:

1

POST `/events`

Consider the following *POST* requests (these are performed in the ascending order of event id):

1.
```
{
  "id":4055191679,
  "type":"PushEvent",
  "actor":{
    "id":2790311,
    "login":"daniel33",
    "avatar_url":"https://avatars.com/2790311"
  },
  "repo":{
    "id":352806,
    "name":"johnbolton/exercitationem",
    "url":"https://github.com/johnbolton/exercitationem"
  },
  "created_at":"2015-10-03 06:13:31"
}
```


2.
```
{
  "id":2712153979,
  "type":"PushEvent",
  "actor":{
    "id":2907782,
    "login":"eric66",
    "avatar_url":"https://avatars.com/2907782"
  },
  "repo":{
    "id":426482,
    "name":"pestrada/voluptatem",
    "url":"https://github.com/pestrada/voluptatem"
  },
  "created_at":"2014-07-13 08:13:31"
}
```


3.
```
{
  "id":4633249595,
  "type":"PushEvent",
  "actor":{
    "id":4276597,
    "login":"iholloway",
```

```
      "name":"iholloway/aperiam-consectetur",
      "url":"https://github.com/iholloway/aperiam-consectetur"
    },
    "created_at":"2016-04-18 00:13:31"
  }
```


4.

```
  {
    "id":1514531484,
    "type":"PushEvent",
    "actor":{
      "id":3698252,
      "login":"daniel51",
      "avatar_url":"https://avatars.com/3698252"
    },
    "repo":{
      "id":451024,
      "name":"daniel51/quo-tempore-dolor",
      "url":"https://github.com/daniel51/quo-tempore-dolor"
    },
    "created_at":"2013-06-16 02:13:31"
  }
```


5.

```
  {
    "id":1838493121,
    "type":"PushEvent",
    "actor":{
      "id":4864659,
      "login":"katrinaallen",
      "avatar_url":"https://avatars.com/4864659"
    },
    "repo":{
      "id":275832,
      "name":"elizabethbailey/error-quod-a",
      "url":"https://github.com/elizabethbailey/error-quod-a"
    },
    "created_at":"2013-09-28 01:13:31"
  }
```


6.

```
  {
    "id":1979554031,
    "type":"PushEvent",
    "actor":{
      "id":3648056,
      "login":"ysims",
      "avatar_url":"https://avatars.com/3648056"
    },
    "repo":{
      "id":292520,
```

```
}
```

7.
```
{
  "id":1536363444,
  "type":"PushEvent",
  "actor":{
    "id":4949434,
    "login":"millerlarry",
    "avatar_url":"https://avatars.com/4949434"
  },
  "repo":{
    "id":310964,
    "name":"brownphilip/rerum-quidem",
    "url":"https://github.com/brownphilip/rerum-quidem"
  },
  "created_at":"2013-06-23 08:13:31"
}
```

8.
```
{
  "id":4501280090,
  "type":"PushEvent",
  "actor":{
    "id":2917996,
    "login":"oscarschmidt",
    "avatar_url":"https://avatars.com/2917996"
  },
  "repo":{
    "id":301227,
    "name":"oscarschmidt/doloremque-expedita",
    "url":"https://github.com/oscarschmidt/doloremque-expedita"
  },
  "created_at":"2016-03-05 10:13:31"
}
```

9.
```
{
  "id":3822562012,
  "type":"PushEvent",
  "actor":{
    "id":2222918,
    "login":"xnguyen",
    "avatar_url":"https://avatars.com/2222918"
  },
  "repo":{
    "id":425512,
    "name":"cohenjacqueline/quam-autem-suscipit",
    "url":"https://github.com/cohenjacqueline/quam-autem-
suscipit"
  },
```

```
  {
    "id":1319379787,
    "type":"PushEvent",
    "actor":{
      "id":3466404,
      "login":"khunt",
      "avatar_url":"https://avatars.com/3466404"
    },
    "repo":{
      "id":478747,
      "name":"ngriffin/rerum-aliquam-cum",
      "url":"https://github.com/ngriffin/rerum-aliquam-cum"
    },
    "created_at":"2013-04-17 04:13:31"
  }
```


GET `/events/actors/2222918`

The response of the *GET* request is the following *JSON* array with the *HTTP* response code *200*:


```
[
  {
    "id":3822562012,
    "type":"PushEvent",
    "actor":{
      "id":2222918,
      "login":"xnguyen",
      "avatar_url":"https://avatars.com/2222918"
    },
    "repo":{
      "id":425512,
      "name":"cohenjacqueline/quam-autem-suscipit",
      "url":"https://github.com/cohenjacqueline/quam-autem-suscipit"
    },
    "created_at":"2015-07-15 15:13:31"
  }
]
```


GET `/actors/streak`

The response of the *GET* request is the following *JSON* array with the *HTTP* response code *200*:


```
[
  {
    "id":4276597,
    "login":"iholloway",
    "avatar_url":"https://avatars.com/4276597"
```

```
    "avatar_url":"https://avatars.com/2917996"
  },
  {
    "id":2790311,
    "login":"daniel33",
    "avatar_url":"https://avatars.com/2790311"
  },
  {
    "id":2222918,
    "login":"xnguyen",
    "avatar_url":"https://avatars.com/2222918"
  },
  {
    "id":2907782,
    "login":"eric66",
    "avatar_url":"https://avatars.com/2907782"
  },
  {
    "id":3648056,
    "login":"ysims",
    "avatar_url":"https://avatars.com/3648056"
  },
  {
    "id":4864659,
    "login":"katrinaallen",
    "avatar_url":"https://avatars.com/4864659"
  },
  {
    "id":4949434,
    "login":"millerlarry",
    "avatar_url":"https://avatars.com/4949434"
  },
  {
    "id":3698252,
    "login":"daniel51",
    "avatar_url":"https://avatars.com/3698252"
  },
  {
    "id":3466404,
    "login":"khunt",
    "avatar_url":"https://avatars.com/3466404"
  }
]
```


PUT /actors

The request is sent with the following body. Response should be an empty body wi
a status code of *200*

```
    "avatar_url":"https://avatars.com/modified2"
  }
```

GET /events

The response of the *GET* request is the following *JSON* array with the *HTTP* response code *200*:

```
[
  {
    "id":1319379787,
    "type":"PushEvent",
    "actor":{
      "id":3466404,
      "login":"khunt",
      "avatar_url":"https://avatars.com/3466404"
    },
    "repo":{
      "id":478747,
      "name":"ngriffin/rerum-aliquam-cum",
      "url":"https://github.com/ngriffin/rerum-aliquam-cum"
    },
    "created_at":"2013-04-17 04:13:31"
  },
  {
    "id":1514531484,
    "type":"PushEvent",
    "actor":{
      "id":3698252,
      "login":"daniel51",
      "avatar_url":"https://avatars.com/3698252"
    },
    "repo":{
      "id":451024,
      "name":"daniel51/quo-tempore-dolor",
      "url":"https://github.com/daniel51/quo-tempore-dolor"
    },
    "created_at":"2013-06-16 02:13:31"
  },
  {
    "id":1536363444,
    "type":"PushEvent",
    "actor":{
      "id":4949434,
      "login":"millerlarry",
      "avatar_url":"https://avatars.com/4949434"
    },
    "repo":{
      "id":310964,
```

```
  },
  {
    "id":1838493121,
    "type":"PushEvent",
    "actor":{
      "id":4864659,
      "login":"katrinaallen",
      "avatar_url":"https://avatars.com/4864659"
    },
    "repo":{
      "id":275832,
      "name":"elizabethbailey/error-quod-a",
      "url":"https://github.com/elizabethbailey/error-quod-a"
    },
    "created_at":"2013-09-28 01:13:31"
  },
  {
    "id":1979554031,
    "type":"PushEvent",
    "actor":{
      "id":3648056,
      "login":"ysims",
      "avatar_url":"https://avatars.com/modified2"
    },
    "repo":{
      "id":292520,
      "name":"svazquez/dolores-quidem",
      "url":"https://github.com/svazquez/dolores-quidem"
    },
    "created_at":"2013-11-11 17:13:31"
  },
  {
    "id":2712153979,
    "type":"PushEvent",
    "actor":{
      "id":2907782,
      "login":"eric66",
      "avatar_url":"https://avatars.com/2907782"
    },
    "repo":{
      "id":426482,
      "name":"pestrada/voluptatem",
      "url":"https://github.com/pestrada/voluptatem"
    },
    "created_at":"2014-07-13 08:13:31"
  },
  {
    "id":3822562012,
    "type":"PushEvent",
```

```
    },
    "repo":{
      "id":425512,
      "name":"cohenjacqueline/quam-autem-suscipit",
      "url":"https://github.com/cohenjacqueline/quam-autem-suscipit"
    },
    "created_at":"2015-07-15 15:13:31"
  },
  {
    "id":4055191679,
    "type":"PushEvent",
    "actor":{
      "id":2790311,
      "login":"daniel33",
      "avatar_url":"https://avatars.com/2790311"
    },
    "repo":{
      "id":352806,
      "name":"johnbolton/exercitationem",
      "url":"https://github.com/johnbolton/exercitationem"
    },
    "created_at":"2015-10-03 06:13:31"
  },
  {
    "id":4501280090,
    "type":"PushEvent",
    "actor":{
      "id":2917996,
      "login":"oscarschmidt",
      "avatar_url":"https://avatars.com/2917996"
    },
    "repo":{
      "id":301227,
      "name":"oscarschmidt/doloremque-expedita",
      "url":"https://github.com/oscarschmidt/doloremque-expedita"
    },
    "created_at":"2016-03-05 10:13:31"
  },
  {
    "id":4633249595,
    "type":"PushEvent",
    "actor":{
      "id":4276597,
      "login":"iholloway",
      "avatar_url":"https://avatars.com/4276597"
    },
    "repo":{
      "id":269910,
      "name":"iholloway/aperiam-consectetur",
```

]

GET /actors

The response of the *GET* request is the following *JSON* array with the *HTTP* response code *200*:


```
[
  {
    "id":4276597,
    "login":"iholloway",
    "avatar_url":"https://avatars.com/4276597"
  },
  {
    "id":2917996,
    "login":"oscarschmidt",
    "avatar_url":"https://avatars.com/2917996"
  },
  {
    "id":2790311,
    "login":"daniel33",
    "avatar_url":"https://avatars.com/2790311"
  },
  {
    "id":2222918,
    "login":"xnguyen",
    "avatar_url":"https://avatars.com/2222918"
  },
  {
    "id":2907782,
    "login":"eric66",
    "avatar_url":"https://avatars.com/2907782"
  },
  {
    "id":3648056,
    "login":"ysims",
    "avatar_url":"https://avatars.com/modified2"
  },
  {
    "id":4864659,
    "login":"katrinaallen",
    "avatar_url":"https://avatars.com/4864659"
  },
  {
    "id":4949434,
    "login":"millerlarry",
    "avatar_url":"https://avatars.com/4949434"
  },
  {
```

```
  {
    "id":3466404,
    "login":"khunt",
    "avatar_url":"https://avatars.com/3466404"
  }
]
```

DELETE `/erase`

This request deletes all events and returns an empty body in the response with status code as *200*.

Online IDE Work Offline

automatically update the code inside the editor.

- Push your updates at regular intervals, to ensure all your changes are on the server before timer runs out.
- Only work with the master branch.
- Git is only used for syncing. For scoring, we'll use the latest code in the IDE.
- Avoid editing files offline, and in the online IDE at the same time.
- Avoid force pushing to master.
- Remember to run the "Run Tests" button to ensure you run all test cases (if present) after pushing your code.

Clone git repository

Copy

```
git clone https://git-rba.hackerrank.com/git/bbb14b70-b8d6-43ea-b287-e502eebc788d django--github-dataset-api-bt82lb754an
```

Commands

Use the following commands to work with this project

- Run

```
pip install --user -r requirements.txt; python manage.py makemigrations && python manage.py migrate --run-syncdb && python manage.py runserver 0.0.0.0:8000
```

- Insta

```
pip install --user -r requirements.txt
```

- Test

```
pip install --user -r requirements.txt; python manage.py makemigrations && python manage.py migrate --run-syncdb && python manage.py test
```

247e06fab9b4115c272d237d3b43ede89d2696

Author: Git Server Admin
Date: Mon Feb 11 2019 18:18:24 GMT+0530 (India Standard Time)

Add initial repository

Run Tests

Submit answer & continue

You can change your submission

1