
gjjg-backend-challenge

Release 1.0.0

Deniz Gokcin

Aug 08, 2020

CONTENTS:

1	gfg-backend-challenge	1
1.1	Requirements	1
1.2	Used Containers	1
1.3	Documentation	1
1.4	Building	1
1.5	Deployment	2
1.6	Testing the endpoints	2
1.7	Notes:	2
1.8	Future Work:	2
2	handlers package	3
2.1	Submodules	3
2.2	handlers. leaderboard module	3
2.3	handlers. score module	3
2.4	handlers. users module	4
2.5	Module contents	4
3	Indices and tables	5
	Python Module Index	7
	Index	9

GJG-BACKEND-CHALLENGE

A REST API endpoint, that manages a game which uses a leaderboard with players submitting new scores from around the world.

1.1 Requirements

- docker
- docker-compose

1.2 Used Containers

```
flask: Contains the Flask application and uWSGI application server.  
nginx: Contains the Nginx web server.  
redis: Stores information about users & handles leaderboard interactions.
```

- The containers can be found under my [docker-hub account](#).

1.3 Documentation

- The documentation can be found [here](#).

1.4 Building

```
docker-compose up -d
```

1.5 Deployment

- The API is deployed to my Docker Swarm running on my Digital Ocean Droplets. The application is distributed on 3 nodes. The main page can be reached from [this](#) link.

```
docker stack deploy -c docker-compose-swarm.yml gjg
```

1.6 Testing the endpoints

- The leaderboard is already filled with 50k random users.
- You can create more random data by using the following snippet in (<https://www.json-generator.com>).

```
[
  '{{repeat(50000)}}',
  {
    user_id: '{{guid()}}',
    rank: '{{integer(1, 10000000)}}',
    country: '{{random("tr", "fr", "us", "uk", "it")}}',
    display_name: '{{firstName()}}',
    points: '{{integer(1, 100000000)}}'
  }
]
```

- You can post your [sample-data.json](#) to <http://178.62.26.184/user/create>.
- You can also add individual users using the same endpoint.
- You can get the leaderboard from <http://178.62.26.184/leaderboard>
- You can update a users score by posting to <http://178.62.26.184/score/submit> following the syntax in [this-document](#)

1.7 Notes:

- GitHub Actions are used for automatically running pytest and deploying to DockerHub.

1.8 Future Work:

- Although there are multiple worker nodes, the response time could be improved if more powerful droplets are used.
- Getting the leaderboard for all the players takes short time but updating in with display name, country takes a long time since it depends on the number of players in the leaderboard. Could be improved but I do not think it is necessary since you only need to see detailed information about the first few players on a leade

HANDLERS PACKAGE

2.1 Submodules

2.2 handlers.leaderboard module

`handlers.leaderboard.generate_all_leaderboard(r)`

Generates the global leaderboard for all the players. Due to the use of sorted sets, as the data structure for the leaderboard, the time complexity of obtaining the leaderboard takes $O(\log(N)+M)$ with N being the number of elements in the sorted set and M the number of elements returned.

Parameters: `r` (RedisClient): Redis Client

Returns: `leaderboard` (list): The leaderboard as a list of dicts

`handlers.leaderboard.generate_leaderboard(r)`

Generates the top 10 players in the global leaderboard. Due to the use sorted sets, as the data structure for the leaderboard, the time complexity of obtaining the leaderboard takes $O(\log(N)+M)$ with N being the number of elements in the sorted set and M the number of elements returned.

Parameters: `r` (RedisClient): Redis Client

Returns: `leaderboard` (list): The leaderboard as a list of dicts

`handlers.leaderboard.generate_leaderboard_by_country(r, iso)`

Generates the leaderboard and filters it by iso code.

Parameters: `r` (RedisClient): Redis Client `iso` (str): Country iso code

Returns: `leaderboard` (list): The leaderboard as a list of dicts

2.3 handlers.score module

`handlers.score.update_user_score(r, user_id, score_worth)`

Increments the score of a given player by `score_worth`. Time complexity of incrementing the score: $O(\log(N))$ where N is the number of elements in the sorted set. Time complexity of updating user profile: $O(1)$.

Parameters: `r` (RedisClient): Redis Client `user_id` (guid): guid `score_worth` (float): Score to increment

2.4 handlers.users module

`handlers.users.get_rank_of_user(r, guid)`

Returns the rank of a specific user in $O(\log(N))$, due to the use of a sorted set.

Parameters: `r` (RedisClient): Redis Client `user_id` (guid): guid

Returns: rank (int): The rank of the given user

`handlers.users.get_user_profile(r, guid)`

Returns detailed information about a given user.

Parameters: `r` (RedisClient): Redis Client `guid` (guid): guid

Returns: user (dict): The user object as a dict.

`handlers.users.register_user(r, user_id, display_name, points, rank, country)`

Stores the json fields of user data in a redis hash. Stores the country iso code of a user in a redis set Adds the user to the leaderboard using `player:<guid>` as the key and the points as the value.

Parameters: `r` (RedisClient): Redis Client `user_id` (guid): guid `display_name` (str): Display Name
`points` (float): Initial points `rank` (int): Initial rank, will be overridden once added to the leader-
board. `country` (str): Country iso code

2.5 Module contents

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

h

- `handlers`, 4
- `handlers.leaderboard`, 3
- `handlers.score`, 3
- `handlers.users`, 4

INDEX

G

`generate_all_leaderboard()` (*in module `handlers.ledgerboard`*), 3
`generate_leaderboard()` (*in module `handlers.ledgerboard`*), 3
`generate_leaderboard_by_country()` (*in module `handlers.ledgerboard`*), 3
`get_rank_of_user()` (*in module `handlers.users`*), 4
`get_user_profile()` (*in module `handlers.users`*), 4

H

`handlers`
 module, 4
`handlers.ledgerboard`
 module, 3
`handlers.score`
 module, 3
`handlers.users`
 module, 4

M

`module`
 `handlers`, 4
 `handlers.ledgerboard`, 3
 `handlers.score`, 3
 `handlers.users`, 4

R

`register_user()` (*in module `handlers.users`*), 4

U

`update_user_score()` (*in module `handlers.score`*),
 3