ptyhon-microservice

Framework: Python Flask/Bottle

Servers: Users Server + Posts Server + API Gateway and Load Balance Server + Direct Message Server + Search Engine Cache Server + Message Queue Async Server.

Database: SQLite3, DynamoDB, Redis, Redis-Queue

Tools: Sandman2, Datasette, NoSQL Workbench, Apache Benchmark

Diagram: https://drive.google.com/file/d/1y3z-DT1PL2G4VL8Yw7wAcBpvwZqLpXM7/view?usp=sharing

This project includes API gateway, load balace, basic authentication for micro blog services. DynamoDB for directly messages services. Redis for inverted-index search engine to search posts contents.

Initialize:

1: `foreman start -m gateway=1,users=1,timelines=3,user-queries=1,timeline-queries=1,direct-messages=1,search-engine=1,message-queue=1,dynamoDB=1,redis=1,worker=1` (for load balance)

2: `make` (use a separate terminal to pre-load and initial databases)

optional: run `make clean` ahead of `make` in case of pre-exist table errors

Authentication:

example of search a post with username = ProfAvert and password = password:

```
http -a ProfAvery:password -v GET 'localhost:5000/search-engine/search-any?q=profavery+tuffy'
```

Test API

message queue:

post_tweet() + inverted_index() asynchronously

```
http -a ProfAvery:password POST 'localhost:5000/posts-async/' username="ProfAvery" text='This is a newwordtotest test'
```

report sync & async

```
ab -p post_data.json -T application/json -A 'ProfAvery:password' -n 250 -c 5 http://127.0.0.1:5000/posts-async/ > async.txt &
```

```
ab -p post_data.json -T application/json -A 'ProfAvery:password' -n 250 -c 5 http://127.0.0.1:5000/posts/ > sync.txt &
```

search engine:

index(postId, text)

```
http -a ProfAvery:password -v POST 'localhost:5000/search-engine/inverted-index/' post_id="777" text="add newinput keyword to test inverted index search"
```

(to test this function, run `http -a ProfAvery:password -v GET 'localhost:5000/search-engine/search-any?q=newinput'` before and after to compare results)

search(keyword)

```
http -a ProfAvery:password -v GET 'localhost:5000/search-engine/search-any?q=profavery'
```

(`profavery` will return "6","11","5","10")

(`tuffy` will return "3","1","10","11","5","4")

(`test` will return "11","15","9")

(`one` will return "14","15","16")

any(keywordList)

```
http -a ProfAvery:password -v GET 'localhost:5000/search-engine/search-any?q=profavery+tuffy'
```

(`profavery OR tuffy` will return "3","1","10","6","11","5","4") (`test OR one` will return "15","16","9","14","11")

all(keywordList)

```
http -a ProfAvery:password -v GET 'localhost:5000/search-engine/search-all?q=profavery+tuffy'
```

(`profavery AND tuffy` will return "5","11","10")

exclude(includeList, excludeList)

```
http -a ProfAvery:password -v GET 'localhost:5000/search-engine/search-exclude?q=profavery+tuffy&exclude=test+one'
```

(`profavery OR tuffy` AND `test OR one` will return "3","1","10","6","5","4")

direct messages

Note: if Authentication is on, 'username' should be the same as 'sender'.

sendDirectMessage(inputs)

```
http -a ProfAvery:password -v POST localhost:5000/dm/ sender='ProfAvery'
receiver='tester' content='this is http POST' quick-
replies:='{"1":"red","2":"white","3":"black","4":"yellow"}'
```

```
http -a ProfAvery:password -v POST localhost:5000/dm/ sender='NotProfAvery'
receiver='tester' content='this is http POST' quick-
replies:='{"1":"red","2":"white","3":"black","4":"yellow"}'
```

(note: this will cause 401, since user != sender)

replyToDirectMessage(inputs)

```
http -a ProfAvery:password -v POST localhost:5000/dm/ chat_id='tester#0005'
is_text='False' sender='ProfAvery' receiver='tester' content='2'
```

```
http -a ProfAvery:password -v POST localhost:5000/dm/ chat_id='tester#0005'
is_text='True' sender='ProfAvery' receiver='tester' content='i want to get
green color'
```

```
http -a ProfAvery:password -v POST localhost:5000/dm/ chat_id='tester#0007'
is_text='False' sender='ProfAvery' receiver='tester' content='3'
```

(note: this will cause error because the latest message of chat_id='tester#0007' & sender='tester' don not contain quick_replies fields')

listDirectMessagesForThe(username)

```
http -a ProfAvery:password -v GET
'localhost:5000/dm/username/ProfAvery/receiver'
```

listRepliesTo(messageId)

```
http -a ProfAvery:password -v GET
'localhost:5000/dm/username/ProfAvery/message/0001'
```

users

createUser(username, email, password)

```
http -a ProfAvery:password -v POST localhost:5000/users/ username=tester
email=test@example.com password=testing
```

authenticateUser(username, password)

```
http -a ProfAvery:password -v GET 'localhost:5000/users/?
username=ProfAvery&password=password'
```

addFollower(username, usernameToFollow)

```
http -a ProfAvery:password -v POST localhost:5000/followers/ follower_id=4
following_id=2
```

removeFollower(username, usernameToRemove)

```
http -a ProfAvery:password -v DELETE localhost:5000/followers/4
```

timelines

getUserTimeline(username)

```
http -a ProfAvery:password -v GET 'localhost:5000/posts/?
username=ProfAvery&sort=-timestamp'
```

getPublicTimeline()

```
http -a ProfAvery:password -v GET 'localhost:5000/posts/?sort=-timestamp'
```

getHomeTimeline(username)

```
http -a ProfAvery:password -v GET "http://localhost:5000/home/ProfAvery"
```

postTweet(username, text)

```
http -a ProfAvery:password -v POST localhost:5000/posts/ username=tester
text='This is a test'
```