CPSC 449-01 Project 2 – Microservices

Zulema Perez - Rancid80s@csu.fullerton.edu

Microservice – Users

Description:

Creates a new user given a specific username, email and password. If the username already exists in the database an exception is thrown, and the http status code 409 CONFLICT will be returned. If the username does not exist, the new user will be saved to the database and where the password is hashed before it is also stored in the database.

API Operation: **createUser(username, email, password)**

URL Endpoint: /users/createuser

HTTP Method: POST

HTTP Status Codes:

success - 200 OK

failure – 409 CONFLICT

Request JSON Data Format:

```
{
        ‘username’: ‘String’
        ‘email’: ‘String’
        ‘password’: ‘String’
}
```

Response JSON Data Format:

```
{
        ‘success’: ‘User account created.’
}
```

Description:

Authenticates the user by comparing the username and the hashed password with the username and hashed password stored in the database. If the username and hashed password to not match the http status code 403 FORBIDDIN is returned.

API Operation: **authenticateUser(username, hashed_password)**

URL Endpoint: /users/authenticate

HTTP Method: GET

HTTP Status Codes:

success: 200 OK

Failure: 403 FORBIDDEN

Response JSON Data Format:

```
{
        ‘success’: ‘User account authenticated.’
}
```

Description:

Starts following a new user and stores the specific username of the user being followed in the FollowsUsers table in the database, if the username does not exist in the database the http status code 400 BAD REQUEST is returned.

API Operation: **addFollower(username, usernametoFollow)**

URL Endpoint: /users/follow

HTTP Method: POST

HTTP Status Codes:

success: 200 OK

failure: 400 BAD REQUEST (user doesn’t exist), 409 CONFLICT

Request JSON Data Format:

```
{
        ‘followed_user’: ‘String’
        ‘user_following’: ‘String’
}
```

Response JSON Data Format:

```
{
        ‘success’: ‘ ’ user_following + ‘ is now following  ‘ +  followed_user
}
```

Description:

Stops following a user by removing the specific user from the FollowsUsers table. If the username does not exist in the database or the user is not being followed then the http status code 400 BAD REQUEST is returned.

API Operation: **removeFollower(username, usernameToRemove)**

URL Endpoint: /users/unfollow

HTTP Method: DELETE

HTTP Status Codes:

success: 200 OK

failure: 400 BAD REQUEST

Request JSON Data Format:

```
{
        ‘followed_user’: ‘String’
        ‘user_following’: ‘String’
}
```

Response JSON Data Format:

```
{
        ‘success’: ‘ ’ user_following + ‘ is no longer following ‘ +  followed_user
}
```

Microservice – Timelines

Description:

Returns a specific user’s 25 most recent posts. If the user does not exist in the database, the http status code 400 BAD REQUEST is returned.

API Operation: **getUserTimeline(username)**

URL Endpoint: /usertimeline

HTTP Method: GET

HTTP Status Codes:

success: 200 OK

failure: 400 BAD REQUEST

Response JSON Data Format:

```
{
        {
                ‘postauthor’: ‘String’
                ‘postbody’: ‘String’
                ‘time_stamp’: ‘DATETIME’
        }
        .
        .
        {
                ‘postauthor’: ‘String’
                ‘postbody’: ‘String’
                ‘time_stamp’: ‘DATETIME’
        }
}
```

Description:

The public timeline gets the 25 most recent post from all users. If there are no users in the database nothing gets returned.

API Operation:

getPublicTimeline()

URL Endpoint: /publictimeline

HTTP Method: GET

HTTP Status Codes: success: 200 OK

Response JSON Data Format:

```
{
        {
                ‘postauthor’: ‘String’
                ‘postbody’: ‘String’
                ‘time_stamp’: ‘DATETIME’
        }
        .
        .
        {
                ‘postauthor’: ‘String’
                ‘postbody’: ‘String’
                ‘time_stamp’: ‘DATETIME’
        }
}
```

Description:

Returns the 25 most recent posts from all users the specific user is following. If the user does not exist, the http status code 400 BAD REQUEST will be returned.

API Operation: **getHomeTimeline(username)**

URL Endpoint: /hometimeline

HTTP Method: GET

HTTP Status Codes:

success: 200 OK

failure: 400 BAD REQUEST (user doesn’t exist)

Response JSON Data Format:

```
{
        {
                ‘postauthor’: ‘String’
                ‘postbody’: ‘String’
                ‘time_stamp’: ‘DATETIME’
        }
        .
        .
        {
                ‘postauthor’: ‘String’
                ‘postbody’: ‘String’
                ‘time_stamp’: ‘DATETIME’
        }
}
```

Description:

Posts a new tweet under a specific given username. The tweet is stored in the database in the Posts table. Will return http status code 400 BAD REQUEST if the specific given username does not exist in the database.

API Operation: **postTweet(username, text)**

URL Endpoint: /post/create

HTTP Method: POST

HTTP Status Codes:

success: 200 OK

failure: 400 BAD REQUEST (user doesn’t exist)

Request JSON Data Format:

```
{
        ‘user’: ‘String’
        ‘text’: ‘String’
}
```

Response JSON Data Format:

```
{
        ‘success’: ‘Tweet posted.’
}
```