Twitter Web Project

Golang & Echo for server
Postgresql/Gorm database

Open Endpoints

Open endpoints require no Authentication.

- **Signup**: `POST /signup`
- **Login**: `POST /login`
- **Search a special username**: `GET /search@/{username}`

Endpoints that require Authentication

Closed endpoints require a valid Token to be included in the body of the request. A Token can be acquired from the Login or Signup view above.

Current user related

Each endpoint manipulates or displays information related to the user whose Token is provided with the request:

- **Timeline:** `POST /timeline`
- **Search a special username:** `POST /search@/{username}`

Account related

Endpoints for viewing and manipulating the Accounts that the Authenticated user has permissions to access:

- **Tweet:** `POST /tweet`
- **Delete Tweet:** `DELETE /deletetweet`
- **Edit Profile:** `PUT /editprofile`
- **Follow:** `POST /follow/{username}`
- **Unfollow:** `DELETE /unfollow/{username}`
- **Like a Tweet:** `POST /like`
- **Logout:** `POST /logout`

Signup

Used to store the new user information to the database and give the user a Token.

URL: `/signup`
Method: `POST`
Auth required: NO

Data constraints:

```
{
      "username": “[String with minimum 3 chars]”
      "email": “[Valid Email address]”
      "password": “[String with minimum 8 chars]”
}
```

Data Example:

```
{
      "username": “John”
      "email": “John@gmail.com”
      "password": “12345678”
}
```

Success Response:
Code: CREATED
Content example:

```
{
      "Username": “John”
      "Email": “John@gmail.com”
      "Image": nil
      "Token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Error Response:
Condition: if username already exists.
Code: BAD REQUEST
Content:

```
{
      "Username John already exists!"
}
```

Login

Used to give the user a Token.

URL: `/login`

Method: `POST`

Auth required: NO

Data constraints:

```
{
      "username": “[String with minimum 3 chars]”
      "password": “[String with minimum 8 chars]”
}
```

Data Example:

```
{
      "username": “John”
      "password": “12345678”
}
```

Success Response:

Code: OK

Content example:

```
{
      "username": “John”
      “email": “John@gmail.com”
      "image": nil
      "token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Error Response:

Condition: if password is wrong

Code: FORBIDDEN

Content:

```
{
      "Password is Wrong!"
}
```

Search a special username

Used to find a user by its username.

URL: `/search@/{username}`
Method: `GET`
Auth required: NO

There is no message body for this request.
Request url example: `/search@/john`

Success Response:
Code: OK
Content example:

```
{
      "username": “John”
      "image": nil
}
```

Error Response:
Condition: if username does not exist
Code: NOT FOUND
Content:

```
{
      "Username John does not exist!"
}
```

Timeline

Display all the tweets created by users that the current user follows and the user’s own tweets, all sorted by creation time.

URL: `/timeline`
Method: `POST`
Auth required: YES

Data Example:

```
{
      "token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Success Response:
Code: OK
Content example:

```
{
      "timeline":
       [
            {
                  “time”: 2021-21-1 18:15:11
                  “text”: “This is my first tweet”
                  “liked”: true
                  “likesCount”: 3
                  “retweeted”: false
                  “retweetsCount”: 1
                  “owner”:
                  {
                        “username”: “john”
                        “image”: nil
                        “following”: true
                  }
             }
      ]
}
```

Search a special username(2)

Used to find a user by its username when current user logged in.

URL: `/search@/{username}`
Method: `POST`
Auth required: YES

Request url example: `/search@/john`

Data Example:

```
{
      "token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Success Response:
Code: OK
Content example:

```
{
      "username": “John”
      "image": nil
      “following”: true
}
```

Error Response:
Condition: if username does not exist
Code: NOT FOUND
Content:

```
{
      "Username John does not exist!"
}
```

Tweet

Tweet something...

URL: `/tweet`
Method: `POST`
Auth required: YES

Data constraints:

```
{
     "text": "[String with minimum 1 and maximum 250 chars]"
     "token": "[String]"
}
```

Data Example:

```
{
     "text": "This is a test tweet"
     "token": "ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845"
}
```

Success Response:
Code: CREATED
Content example:

```
{


     "time": 2021-21-1 18:15:11
     "text": "This is my first tweet"
     "liked": true
     "likesCount": 3
     "retweeted": false
     "retweetsCount": 1
     "owner":
     {
          "username": "john"
          "image": nil
          "following": true
     }


}
```

Delete Tweet

Users can delete their own tweets.

URL: `/deletetweet`
Method: `DELETE`
Auth required: YES

Data constraints:

```
{
      "text": “[String with minimum 1 and maximum 250 chars]”
      "token": “[String]”
}
```

Data Example:

```
{
      “text”: “This is a test tweet”
      "token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Success Response:
Code: OK
Content example:

```
{
      "Tweet successfully deleted!"
}
```

Error Response:
Condition: if tweet not found for current user
Code: NOT FOUND
Content:

```
{
      "Tweet Not found for user"
}
```

Edit Profile

Users can change their username and image in this part.

URL: `/editprofile`
Method: `PUT`
Auth required: YES

Data Example:

```
{
      “username”: “John”
      “image”: nil
      "token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Success Response:
Code: OK
Content example:

```
{
      “username”: “Joni”
      “email”: “John@gmail.com”
      “image”: nil
}
```

Error Response:
Condition: if new username already exists.
Code: BAD REQUEST
Content:

```
{
      "Username Joni already exists!"
}
```

Follow

The user is able to follow anyone who wants.

URL: `/follow/{username}`

Method: `POST`

Auth required: YES

Request url example: `/follow/John`

Data Example:

```
{
      "token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Success Response:

Code: OK

Content example:

```
{
      “username”: “John”
      “image”: nil
      “following”: true
}
```

Error Response:

Condition: if given username does not exist.

Code: NOT FOUND

Content:

```
{
      "Username John does not exist!"
}
```

Unfollow

The user is able to unfollow anyone who followed before.

URL: `/unfollow/{username}`
Method: `DELETE`
Auth required: YES

Request url example: `/unfollow/John`

Data Example:

```
{
      "token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Success Response:
Code: OK
Content example:

```
{
      “username”: “John”
      “image”: nil
      “following”: false
}
```

Error Response:
Condition: if the user wants to unfollow the one who did not follow him/her.
Code: BAD REQUEST
Content:

```
{
      "You did not follow user John"
}
```

Like a Tweet

React to someone’s tweet by liking it.

URL: `/like`
Method: `POST`
Auth required: YES

Data Example:

```
{
      “text”: “This is the test tweet”
      “ownerusername”: “John”
      "token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Success Response:
Code: OK
Content example:

```
{


      “time”: 2021-21-1 18:15:11
      “text”: “This is my first tweet”
      “liked”: true
      “likesCount”: 4
      “retweeted”: false
      “retweetsCount”: 1
      “owner”:
      {
            “username”: “john”
            “image”: nil
            “following”: true
      }


}
```

Logout

Logout from current user.

URL: `/logout`

Method: `POST`

Auth required: YES

Data Example:

```
{
      "token": “ejkjsdslhsl.hskjfjajsfdasdfrjsksdfrtldjfi845”
}
```

Success Response:

Code: OK

Content example:

```
{
      “You logged out successfully!”
}
```