Wordle: RESTful Service

There are two microservices : Users and Games

1. Users Microservice:

- **Registering a new user (id, username, password)**

  The API is for registering a new user. If all parameters are NOT NULL and passed successfully, we get a success message with 200 status code.
  If any of the parameters are NULL or empty or the user already exists, we get an error status code of 409.
  In case no value has been provided to any parameter then we get back an error status code of 400.

  Endpoint: /register
  Method: POST
  Parameters: id, username, password
  Sample JSON Request :

```
{
        “id”: 2,
        “username”: “Debdyuti”,
        “Password”: “deb@123”
}
```

  Success Response:

```
{
        "message": "Successfully registered!",
         "statusCode": 200

}
```

  Error Response:

```
{
  "error": "409 Conflict: UNIQUE constraint failed: userData.id"
}
```

  Error Response:

```
{
  "error": "409 Conflict: CHECK constraint failed: userData"
}
```

- **Authenticating a user (username, password)**

The API is used for user login. If the username and password match the values in the database, it returns 200 status code and “authenticated: true” response. If the validation returns false, it sends back a response of 401 status code which is an Unauthorised error.
Note: We have used Basic Auth authorization for authenticating the user.

Endpoint: /auth
Method: GET
Parameters: username, password
Authorization: Basic Auth

Success Response:
```
{
        "statusCode": 200,
        "authenticated": "true"
}
```
Error Response:
```
{
        “statusCode": 401,
         "error": "Unauthorized",
        "message": "Login failed !"
}
```

2. **<u>Game Microservice:</u>**

- **Create a new game**
  This API is for creating a new game for a specific user. The client will need to include the user id in the request body. If successful (user exists), the server will send a response that includes the id of the newly created game.

  Endpoint: /game
  Method: POST
  Parameters: none
  Request Body: userId

  Sample Request Body:
  ```
  {
     “userId”: 3,
  }
  ```
  Success Response (201):
  ```
  {
    "gameId": 7,
    "guesses": 6
  }
  ```
  Error Response (404):
  ```
  {
    "error": "Could not find user with this id"
  }
  ```

- **Make a guess to a game**

  This API is for making a guess to an existing game. The client will need to specify the game id in the URL. The client will also need to include the user id and guess word in the request body. If successful (game exists, game belongs to user, valid word), the server will send a response that includes whether the guessed word is correct. If not correct, it will also send the letters in the right place and the letters in the wrong place.

  Endpoint: /game/:gameId
  Method: PATCH
  Parameters: gameId
  Request Body: userId, word

  Sample Request Body:
  ```
  {
     "userId": "3",
     "word": "queue"
  }
  ```
  Success Response (200):
  ```
  {
     "word": {
        "input": "queue",
        "valid": true,
        "correct": false
     },
     "gussesLeft": 3,
     "data": [
        {
           "q": {
              "inSecret": false,
              "wrongSpot": false
           }
        },
        {
           "u": {
              "inSecret": false,
              "wrongSpot": false
           }
        },
        {
           "e": {
              "inSecret": true,
              "wrongSpot": false
           }
  ```

```
        },
        {
            "u": {
                "inSecret": false,
                "wrongSpot": false
            }
        },
        {
            "e": {
                "inSecret": true,
                "wrongSpot": true
            }
        }
    ]
}
```

Sample Error:
Request:

```
{
    "userId": "6",
    "word": "queue"
}
```

Response (400):

```
{
    "error": "This game does not belong to this user"
}
```

Sample Error:
url: /game/10
Request:

```
{
    "userId": "6",
    "word": "queue"
}
```

Response (404):

```
{
    "error": "Could not find a game with this id"
}
```

- **Retrieve game state**
  This API is for retrieving the game state of a specific game. The client will need to provide the game id in the URL. If successful (game exists), the server will send back a response that includes the state of the game.

  Endpoint: /game/:gameId
  Method: GET
  Parameters: gameId
  Request Body: none

  Success Response (200):
  ```
  {
     "guessesLeft": 6,
     "finished": false,
     "gussedWords": []
  }
  ```

  Success Response (200):
  ```
  {
     "guessesLeft": 2,
     "finished": true,
     "gussedWords": [
        {
           "teddy": [
              {
                 "t": {
                    "inSecret": false,
                    "wrongSpot": false
                 }
              },
              {
                 "e": {
                    "inSecret": true,
                    "wrongSpot": true
                 }
              },
           …
  }
  ```

  The response contains a lot of information which can take up pages in this document, so I chose to not include the whole response. However, the response will contain an array of guessed words with each word containing the guess state.

- **Retrieve games for a user**
  This API is for retrieving the games belonging to a specific user. The client will need to provide the user id. If successful (user exists), the server will send a response that includes all the games belonging to this user.

  Endpoint: /users/:userId/games
  Method: GET
  Parameters: userId
  Request Body: none

  Sample Request:
  url: /users/3/games
  Success Response (200):
```
[
   {
      "gameId": 1,
      "guessesLeft": 6,
      "finished": false
   },
   {
      "gameId": 2,
      "guessesLeft": 4,
      "finished": false
   },
   {
      "gameId": 3,
      "guessesLeft": 6,
      "finished": false
   },
   {
      "gameId": 4,
      "guessesLeft": 6,
      "finished": false
   },
   {
      "gameId": 5,
      "guessesLeft": 1,
      "finished": true
   },
   {
      "gameId": 7,
      "guessesLeft": 2,
      "finished": true
   },
    {
        "gameId": 8,
        "guessesLeft": 3,
        "finished": false
    }
]
```

Sample Error:
url: /users/120/games
Error Response (404):

```
{
    "error": "Could not find this user"
}
```