QuizUp Single Player Web

REST URI Guide

Gunnar Marteinsson
Kristinn Júlíusson
Sonja Jónsdóttir
Steinar Pór Árnason

May 13, 2016
Contents

Introduction 3
Login endpoints 4
Topic endpoints 5
User endpoints 11
Games endpoints 15
Introduction

This document is a detailed guide to the REST API developed for QuizUp Single Player Web application.

To generate the documentation, the user will have to navigate to the src/doc folder and run the generate.sh. The script file generate relevant html files that are then served out to http://localhost:4000/api/v1/doc/ if you have the server running.
Login endpoints

Notes

• Sets the users facebookID cookie

POST Login

Logs a user in using Facebook identification

• **URL**
  
  `/`

• **Method**
  
  **POST**

• **Body Params**

  Required:

  ```
  facebookID: string, accessToken: string
  ```

  **Note:** The access token is required because we authenticate that it is valid before logging the user into the application

• **Success Response**

  o **Code:** 200 OK

  ```
  {'Welcome back!'}
  ```

  **Note:** Sets the cookie

• **Error Response**

  o **Code:** 412 PRECONDITION FAILED

  ```
  {}
  ```

  **OR**

  o **Code:** 500 INTERNAL SERVER ERROR

  ```
  {'Internal server error'}
  ```
Topic endpoints

Notes

- If the facebookID is needed it is extracted from the cookie saved when logging in

GET All topics

Gets all topics in the database, doesn't return the questions

- **URL**
  
  `/`

- **Method**
  
  `GET`

- **Success Response**
  
  o **Code:** 200 OK

  ```
  Content: [Object, ..]
  ```

  ```
  name: string,
  description: string,
  category: string,
  tags: [string],
  iconUrl: string,
  leaderboard: [
    {
      facebookId: string,
      name: string,
      score: integer,
      round: integer,
      date: date
    }, ..
  ],
  titles: [
    {
      title: string,
      milestone: integer
    }, ..
  ]
  ```

- **Error Response**

  o **Code:** 500 INTERNAL SERVER ERROR

  ```
  Content: {'Internal server error'}
  ```
POST Validate answer

Used to see if a question has been answered correctly.

- **URL**
  
  `/:slug/answer`

- **Method**
  
  POST

- **URL Params**

  Required:
  
  - `slug`

- **Body Params**

  Required:

  question in this context is the `question._id` of a question and answer is the `answer._id` of an answer to that same question.

  
  `{question, answer}`

- **Success Response**

  - **Code**: 200 OK
    
    Content: `{correct: bool, correctAnswer: string, message: string}`

- **Error Response**

  - **Code**: 404 NOT FOUND
    
    Content: `{'Question not found'}`

  OR

  - **Code**: 500 INTERNAL SERVER ERROR
    
    Content: `{'Internal server error'}` OR `{correct: bool, correctAnswer: string, message: string}`

  **Note:** `'Internal server error'` is returned when the server can’t perform a lookup of the question. Otherwise there was an error removing the Redis user set used to store a person’s game which is what the `message` field is used to indicate.
GET Questions

Gets four questions from the users Redis user set.

- **URL**
  
  `/:slug/questions`

- **Method**
  
  POST

- **URL Params**

  Required:

  - **slug**

- **Success Response**

  - **Code:** 200
    
    **Content:**

    ```
    {{Object, ..}}
    ```

    ```
    text: string,
    _id: string,
    answers: [  
      {  
        text: string,
        _id: string
      }, ..
    ],
    imageUrl: string
    ```

    **Note:** An array of 4 questions, each question has an array of 4 answers.

- **Error Response**

  - **Code:** 500
    
    **Content:** If it fails to get questions from Redis

    ```
    {'relevant error message'}
    ```

  - **OR**

  - **Code:** 404
    
    **Content:** If there are no questions in this topic

    ```
    {'relevant error message'}
    ```
GET Wildcard question

Gets a random question for the wildcard topic.

- **URL**
  
  `/:slug/wildcard/question`

- **Method**
  
  GET

- **URL Params**

  Required:
  
  slug

- **Success Response**

  - Code: 200 OK

    Content: `{text: string, _id: string, imageUrl: string, answers: [id: string, text: string]}`

- **Error Response**

  - Code: 404 NOT FOUND

    Content: `{No question found for topic'}`

  - OR

    - Code: 500 INTERNAL SERVER ERROR

    Content: `{Error getting wildcard question'}`
GET Wildcard topics

Gets 3 random topics, none of them being the current topic being played

- **URL**
  
  `/:slug/wildcard/topics`

- **Method**
  
  `GET`

- **URL Params**
  
  **Required:**
  
  `slug`

- **Success Response**
  
  - **Code:** 200 OK
    
    **Content:** 
    ```json
    {
      "Object",
      "Object",
      "Object"
    }
    ```

    - `name`: string,
    - `category`: string,
    - `iconUrl`: string,
    - `slugUrl`: string

- **Error Response**
  
  - **Code:** 404 NOT FOUND
    
    **Content:** 
    ```json
    {'No topics found in database'}
    ```

  - **OR**
    
    - **Code:** 500 INTERNAL SERVER ERROR
      
      **Content:** 
      ```json
      {'Internal server error'}
      ```
GET Topic

Gets a single topic

- **URL**
  
  `/:slug`

- **Method**

  `GET`

- **URL Params**

  **Required:**

  `slug`

- **Success Response**

  - **Code:** 200 OK

    **Content:** 

    ```json
    {Object}
    name: string,
    description: string,
    category: string,
    tags: [string],
    iconUrl: string,
    leaderboard: [
      {
        facebookId: string,
        name: string,
        score: integer,
        round: integer,
        date: date
      }, ..
    ],
    titles: [
      {
        title: string,
        milestone: integer
      }, ..
    ]
    ```

- **Error Response**

  - **Code:** 400 BAD REQUEST

    **Content:** 

    ```json
    {'Regex invalid'}
    ```

    **Note:** This means that the slug provided contained illegal characters

  - **OR**

    - **Code:** 404 NOT FOUND

      **Content:** 

      ```json
      {'Topic not found'}
      ```

  - **OR**

    - **Code:** 500 INTERNAL SERVER ERROR

      **Content:** 

      ```json
      {'Internal server error'}
      ```
User endpoints

Notes

• If the facebookID is needed it is extracted from the cookie saved when logging in

GET User

Gets user information.

• URL

/ 

• Method

GET

• Success Response

  o Code: 200 OK

  Content: 

  Note: Returns the whole User object.

• Error Response

  o Code: 404 NOT FOUND

  Content: 

  Note: An error occurred during

  OR

  o Code: 500 INTERNAL SERVER ERROR

  Content: 

  Note: An error occurred during
PUT User

Updates user information.

- **URL**
  
  `/`

- **Method**
  
  `PUT`

- **Body Params**

  Optional:

  displayName: `string`, title: `string`, titles: `[strings]`, nationality: `string`,

  dob: `date`, gender: `string`, about: `string`

- **Success Response**

  o **Code**: 200 OK

    Content: `{'User updated'}`

- **Error Response**

  o **Code**: 404 NOT FOUND

    Content: `{'User not found'}`

  OR

  o **Code**: 412 PRECONDITION FAILED

    Content: `{'Data invalid'}`

  OR

  o **Code**: 500 INTERNAL SERVER ERROR

    Content: `{'Internal server error'}`
POST favorite topic

Adds a favorite topic to the user

- **URL**
  
  /favorite/:slug

- **Method**
  
  POST

- **URL Params**

  Required:
  
  slug

- **Success Response**

  o **Code**: 200 OK
    
    Content: 
    
    `{"Favorite topic added successfully"}`

- **Error Response**

  o **Code**: 404 NOT FOUND
    
    Content: 
    
    `{"Error trying to add a favorite topic (User not found)"}` OR `{"Error trying to add a favorite topic (Topic not found)"}`

  OR

  o **Code**: 412 PRECONDITION FAILED
    
    Content: 
    
    `{"This topic is already in your favorites"}`

  OR

  o **Code**: 500 NOT FOUND
    
    Content: 
    
    `{"Internal server error"}`
DELETE favorite topic

Removes a favorite topic from a user

- **URL**
  
  /favorite/:slug

- **Method**
  
  DELETE

- **URL Params**
  
  Required:

  slug

- **Success Response**

  - Code: 200 OK
    
    Content: {'Favorite topic removed successfully'}

- **Error Response**

  - Code: 404 NOT FOUND
    
    Content: {'Error trying to remove a favorite topic (User not found)'} OR {'Error trying to remove a favorite topic (Topic not found)'}

  - OR

    - Code: 412 PRECONDITION FAILED
      
      Content: {'This topic is not in your favorites'}

  - OR

    - Code: 500 NOT FOUND
      
      Content: {'Internal server error'}
Games endpoints

Notes

• If the facebookID is needed it is extracted from the cookie saved when logging in

POST Game

Send various details to the server to be updated. Updates the topic leaderboard (Topic DB, Topic.leaderboard) and the topic the user was playing (User DB, User.topics)

• URL
  
  /

• Method
  
  POST

• Body Params

  Required:

  name: string, slugUrl: string, game: {answered: integer, round: integer}, score: integer, round: integer

• Success Response

  o  Code: 200 OK

  Content: {Object}

  updateTopic: {
    code: integer,
    message: string
  },
  updateUser: {
    code: integer,
    message: string
  }

• Notes: This endpoint always returns a 200 OK status (unless the server isn’t reached). It returns an object with two objects that have to be checked to make sure that all data was posted/updated correctly. We chose to call one endpoint instead of multiple and have it return a general overview object because the endpoint does two separate things.

updateTopic can return the following:

  o  Success Response

    •  Code: 200 OK

    Content: [{[title: string]}]
- **Error Response**
  - **Code:** 404 NOT FOUND
    - **Content:** `{'User not found'}`
  - OR
    - **Code:** 404 NOT FOUND
      - **Content:** `{'Topic not found'}`
  - OR
    - **Code:** 500 INTERNAL SERVER ERROR
      - **Content:** `{'Server error'}` OR `{'Failed to save user'}`

- **updateUser can return the following:**
  - **Success Response**
    - **Code:** 200 OK
      - **Content:** `{'Topic leaderboard updated'}`
  - OR
    - **Code:** 304 NOT MODIFIED
      - **Content:** `{'Topic not modified'}`
  - OR
    - **Code:** 500 NOT FOUND
      - **Content:** `{'Internal server error'}` OR `{'Failed to save topic'}`