A7: High-level architecture. Privileges. Web resources specification

This document catalogues the resources used by *SegFault*, and identifies their properties. These include references to the graphical interfaces, and the format of JSON responses. Furthermore, this artifact presents a documentation for the web application, including all operations over data: create, read, update, and delete.

1. Overview

This section presents an overview of the web application to implement, identifies the application's modules and briefly describes them. The web resources associated with each module are detailed in the individual documentation of each module.

M01: Authentication	Web resources associated with user authentication, includes the following system features: login/logout and registration.
M02: Individual Profile	Web resources associated with individual profile management, includes the following system features: view and edit personal profile information, view personal notifications and favorite questions.
M03: Messages	Web resources associated with questions, answers and their comments, includes the following system features: add, view, vote, report and delete questions, add, view, vote, report and delete answers and add, view, vote, report and delete comments.
M04: Categories	Web resources associated with categories, includes the following system features: list categories and search categories
M05: Static Pages	Web resources with static content are associated with this module: about.

2. Permissions

This section defines the permissions used in the modules to establish the conditions of access to resources, in increasing order of restrictiveness.

PUB	Public	Group of users without privileges.
USR	User	Group of authenticated users.

OWN	Owner	Group of users that can update their profiles and have privileges regarding their messages (extension over the *USR* permissions).
MOD	Moderator	Group of Moderators that can manage any message (extension over the *USR* permissions).

3. Modules

This section documents each web resource of each module, indicating the URL, HTTP methods, the possible parameters, interfaces with the user (referring to the A3 artefact) — or JSON responses in the event of APIs.

> Web resources organized by module Document and describe the web resources associated with each module, indicating the URL, HTTP method, request parameters and response. Follow the RESTful resource naming At the end of this page is presented some usual descriptors to document the web resources.

3.1 Module M01: Authentication

Endpoints of module *Authentication*

These are the endpoints available in the Authentication module. * R101: Login Form `/login` * R102: Login Action `/login` * R103: Logout Action `/logout` * R104: Register Form `/register` * R105: Register Action `/register`

R101: Login Form

URL	`/login`
Description	Drop down form to login to a user account.
Method	GET
UI	All pages allow for login, from the *navbar*. (UI01-UI08)
Submit	R102
Permissions	PUB

R102: Login Action

URL	`/login`
Description	This web resource logs the user into the system.
Request Body	+username: string (account username) +password: string (account password)
Method	POST
Redirects	Reloads the page on success or failure.

Permissions	PUB

R103: Logout Action

URL	`/logout`
Description	This web resource logs out the authenticated user (or moderator).
Method	POST
Redirects	Reloads the page.
Permissions	USR

R104: Register Form

URL	`/register`
Description	Page with a form to register a new user account.
Method	GET
UI	All pages allow registration, from the *navbar*. (UI01-UI08)
Submit	R105
Permissions	PUB

R105: Register Action

URL	`/register`
Description	This web resource inserts the user into the system.
Method	POST
Redirects	Redirects to the main page on success.
Request Body	+username: string (account username) +email: string (account email) +password: string (account password)
Permissions	PUB

3.2 Module M02: Individual Profile

Endpoints of module *Individual Profile*

These are the endpoints available in the Individual Profile module. * … > TODO Timon

3.3 Module M03: Messages

Endpoints of module *Messages*

- R301: Search Questions Page `/questions`
- R302: Get recent questions `/questions/recent`
- R303: Get hot questions `/questions/hot`
- R304: Get highly voted questions `/questions/highly-voted`
- R305: Get active questions `/questions/active`
- R306: Get question's details `/questions/{id}`
- R307: Add a new question - Form `/questions/create`
- R308: Add a new question - Action `/questions`
- R309: Edit a question - Form `/questions/{id}/edit`
- R310: Edit a question - Action `/questions/{id}`
- R311: Delete a question `/questions/{id}`
- R312: Get question's answers `/questions/{id}/answers`
- R313: Get answer's details `/questions/{id}/answers/{answer_id}`
- R314: Add a new answer `/questions/{id}/answers`
- R315: Edit an answer `/questions/{id}/answers/{answer_id}`
- R316: Delete an answer `/questions/{id}/answers/{answer_id}`
- R317: Get question's comments `/questions/{id}/comments`
- R318: Add a new comment to a question `/questions/{id}/comments`
- R319: Edit a question's comment `/questions/{id}/comments/{comment_id}`
- R320: Delete a question's comment `/questions/{id}/comments/{comment_id}`
- R321: Get answer's comments `/questions/{id}/answers/{answer_id}/comments`
- R322: Add a new comment to an answer `/questions/{id}/answers/{answer_id}/comments`
- R323: Edit an answer's comment `/questions/{id}/answers/{answer_id}/comments/{comment_id}`
- R324: Delete an answer's comment `/questions/{id}/answers/{answer_id}/comments/{comment_id}`

  TODO Bayard R311-317 TODO Ed R318-R324

R301: Search Questions Page

URL	`/questions`
Description	*Navbar* on the homepage allows searching for a question by it's title.
Method	GET
Parameters	+query: string (String field to search for in questions). ?categories: string[] (Optionally, filter questions by categories).
UI	UI01
Response Body	JSON01
Permissions	PUB

R302: Get Recent Questions

URL	`/questions/recent`
Description	Show the 25 most recent questions.
Method	GET
UI	UI01
Response Body	JSON01
Permissions	PUB

R303: Get Hot Questions

URL	`/questions/hot`
Description	Show the 25 *hottest* questions (highly voted questions with most answers but no correct answer).
Method	GET
UI	UI01
Response Body	JSON01
Permissions	PUB

R304: Get Highly Voted Questions

URL	`/questions/highly-voted`
Description	Show the 25 questions with highest score.
Method	GET
UI	UI01
Response Body	JSON01
Permissions	PUB

R305: Get Active Questions

URL	`/questions/active`
Description	Show 25 unanswered questions.
Method	GET
UI	UI01
Response Body	JSON01
Permissions	PUB

R306: Get Question's Details

URL	`/questions/{id}`
Description	The details and contents of a question.
Method	GET
UI	UI03
Parameters	+id: integer (The question's *id*)
Response Body	JSON02
Permissions	PUB

R307: Add a New Question - Form

URL	`/questions/create`
Description	Page with a form for creating a new question.
Method	GET
UI	UI04
SUBMIT	R308
Permissions	USR

R308: Add a New Question - Action

URL	`/questions`
Description	Web resource that creates a new question based on the input received. Redirects to the new question page on success, and back to new question form on failure.
Method	POST
Request Body	+title: string (The question's title) +content: string (The question's contents) +author: integer (The question author's id) +categories: string[] (The question's categories)
Redirects	R306 - SUCCESS R307 - FAILURE
Permissions	USR, OWN

R309: Edit a Question - Form

URL	`/questions/edit`
Description	Page with a form for editing a question.
Method	GET
UI	UI04
SUBMIT	R310
Permissions	OWN

R310: Edit a Question - Action

URL	`/questions`
Description	Web resource that edits a previous question based on the input received. Redirects to the new question page on success, and back to edit question form on failure.
Method	PUT
Request Body	+content: string (The question's contents) +author: integer (The question author's id) ?categories: string[] (The question's categories)
Redirects	R306 - SUCCESS R309 - FAILURE
Permissions	OWN

3.4 Module M04: Categories

Endpoints of module *Categories*

TODO Ed

3.5 Module M05: Static Pages

Endpoints of module *Static Pages*

TODO Timon

4. JSON/XML Types

JSON01: Questions {question}[]

JSON response containing a list of questions and their contents.

```
{
  "question": [
    {
      "id": 137
      "title": "Is there multiple inheritance in JAVA?",
      "content": "I'm a C++ developer and have always been intrigued by the
        fact that some languages allow for multiple inheritance and others
        don't.\nIs there a deeper thought process behind this?",
      "author": "sudomakemeasandwich",
      "score": 120,
      "correct_answer": 43,
      "creation_time": "2018-04-07 14:11",
      "category": [
        "JAVA",
        "OOP"
      ]
    },
    {
      "id": 12
      "title": "Is javascript dynamically typed?",
      "content": "I was wondering if javascript was a dynamically typed language?",
      "author": "sudormrf",
      "score": 10,
      "correct_answer": 435,
      "creation_time": "2018-03-01 10:03",
      "category": [
        "javascript"
      ]
    },
    ...
  ]
}
```

JSON02: Question {question}

JSON response containing a question's details and contents, as well as its previous versions, in order to show if it was edited and by whom.

```
{
  "question":
    {
      "id": 137
      "title": "Is there multiple inheritance in JAVA?",
      "author": "sudomakemeasandwich",
      "score": 120,
      "correct_answer": 43,
      "category": [
        "JAVA",
        "OOP"
      ],
      "content": [
        {
          "version": "I'm a C++ developer and have always been intrigued by the
          fact that some languages allow for multiple inheritance and others
          don't.\nIs there a deeper thought process behind this?",
          "creation_time": "2018-04-07 14:11",
          "author": "sudomakemeasandwich",
        },
        {
          "version": "I'm a C developer and have always been i",
          "creation_time": "2018-04-07 14:03",
          "author": "sudomakemeasandwich",
        },
      ]
    },
}
```

Document the JSON or XML responses that will be used by the web resources.

Web resources descriptors - Do not include on the final artefact

TODO delete * URL - Resource identifier, following the RESTful resource naming conventions * Description - Describe the resource, when it's used and why * UI - Reference to the A3 user interface used by the resource * SUBMIT - Reference to the actions/requests integrated with the resource * Method - HTTP request Method * Parameters - Information that is sent through the URL, by a query string or path * Request Body - Data associated and transmitted with each request * Returns - HTTP code returned from a request * Response Body - Data sent from the server, in response to a given request * Permissions - Required permissions to access the resource

Revision history

GROUP1763, 10/04/2018

André Cruz, up201503776@fe.up.pt
Daniel Marques, up201503822@fe.up.pt
Edgar Carneiro, up201503784@fe.up.pt
João Carvalho, up201504875@fe.up.pt