**Forum web services API specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version History** | | | |
| **version** | **description** | **author** | **date** |
| 1.01 | Initial version | Joshgun Huseynov | 18.12.2018 |
| 1.02 | Resolved Date/Time issue | Vugar Nesirov | 22.12.2018 |

**Table of contents**

1. Introduction………........................................................ 3
2. Operations………………………………………………………………. 3
   1. Topics………………………………………………………… 3
      1. getAllTopics……………………………………….. 3
      2. getTopicById………………………………………. 5
      3. getPopularTopics………………………………..
      4. getSimilarTopics………………………………….
      5. getCommentsByTopicId………………………
      6. addTopic……………………………………………. 6
      7. updateTopicViewCount……………………….
   2. Comments…………………………………………………..
      1. addComment……………………………………..
   3. Users…………………………………………………………..
      1. getTopicByUserId……………………………….
      2. getUserByEmail………………………………….
      3. addUser……………………………………………..
   4. Roles…………………………………………………………..
      1. getActionsByRoleId……………………………
3. Error codes and messages………………………………………. 8
4. Resource entities……………………………………………………. 8

**1.Introduction**

This document describes specifications about Forum API.

Base url for all methods: [http://localhost:8080/api/\*\*](http://localhost:8080/api/**)

All requests and responses are JSON format. All responses contain 2 main fields – status and body.

In case of success response status field is “success” and body field is appropriate requested data.

In case of error response status field is “error” and body field is Error entity containing error code and error message.

**2.Operations**

All possible operations are described below

**2.1.Topics**

**2.1.1.getAllTopics**

Uri: /topics

Method: GET

Example response on success:

{

"body": [

  {

"commentList": [

  {

"desc": "Test description.",

"id": 17,

"writeDate": "2018-12-20T23:39:27"

},

  {

"desc": "Test description.",

"id": 18,

"writeDate": "2018-12-22T10:51:15"

}

],

"desc": "Test description.",

"id": 11,

"shareDate": "2018-12-15T05:30:00",

"status": 1,

"title": "Test Title.",

"user": {

"email": "fatime@gmail.com",

"firstName": "Fatime",

"id": 4,

"imagePath": "default.png",

"lastName": "Gurbanova"

},

"viewCount": 21

},

  {

"commentList": [],

"desc": "hshshshshshhshs",

"id": 10,

"shareDate": "2018-12-08T12:15:29",

"status": 1,

"title": "Maven",

"user": {

"email": "vnesirov@gmail.com",

"firstName": "Vugar",

"id": 7,

"imagePath": "vnesirov@gmail.com\p1.jpg",

"lastName": "Nasirov"

},

"viewCount": 0

}

],

"status": "success"

}

Example response on error:

{

"body": {

"code": 300,

"message": "Internal error!"

},

"status": "error"

}

(For all error codes and messages please see chapter 3)

**2.1.1.getTopicById**

Uri: /topics/{id}

Method: GET

Path parameters: id – id of the topic

Example response on success:

{

"body": {

"commentList": [],

"desc": "Some popular websites send email/push notifications to their users if they notice unusual or suspicious login activity. Examples include Google, Facebook, Mozilla, Booking.com. How to implement this feature in Java/Spring application?",

"id": 3,

"shareDate": "2018-09-15T05:30:00",

"title": "Unusual Sign In notification with Spring Security",

"user": {

"firstName": "Fatime",

"id": 4,

"imagePath": "default.png",

"lastName": "Gurbanova"

},

"viewCount": 92

},

"status": "success"

}

Example response on error:

{

"body": {

"code": 200,

"message": "Topic with given id not found"

},

"status": "error"

}

(For all error codes and messages please see chapter 3)

…

**2.1.6.addTopic**

Uri: /topics

Method: POST

Example request:

{

"desc": "Description of topic",

"shareDate": "2018-09-15T05:30:00",

"title": "Title of topic",

"user": {

"id": 4

},

"viewCount": 0,

"status": 1

}

Example response on success:

{

"body": {

"commentList": [],

"desc": "Description of topic",

"id": 12,

"shareDate": "2018-09-15T05:30:00",

"status": 1,

"title": "Title of topic",

"user": {

"id": 4

},

"viewCount": 0

},

"status": "success"

}

Example response on error:

{

"body": {

"code": 300,

"message": "Internal error!"

},

"status": "error"

}

…

**3.Error codes and messages**

|  |  |  |
| --- | --- | --- |
| **Code** | **Message** | **Description** |
| 100 | Email is already taken | Returns when user email is already taken |
| 101 | User not found | User not found |
| 200 | Topic with given id not found | Topic with given id not found |
| 300 | Internal error | Internal error |

**4.Resource entities**

|  |  |  |
| --- | --- | --- |
| **Topic entity** | | |
| **Field type** | **Field name** | **description** |
| int | id | Id of the topic |
| string | title | Title of the topic |
| string | desc | Description of the topic |
| timestamp | shareDate | Share date of the topic. Example format: “yyyy-MM-dd’T’hh:mm:ss” |
| int | viewCount | View count of the topic |
| User entity | user | User who shared the topic |
| Array of Comment entity | commentList | List of the comments written to the topic |
| int | commentCount | Count of the comments. Used only getPopularTopics method |
| int | status | Status of the topic.  Possible statuses:  1 – Active  2 - Inactive |