**Angular SPA project**

**public API**

Version: 0.1, 5-Nov 2015

Оглавление

[API 3](#__RefHeading__35816_1441106636)

[Implemented services 3](#__RefHeading__35506_1441106636)

[User service 4](#__RefHeading__35918_1441106636)

[Get user 4](#__RefHeading__35920_1441106636)

[Get users 6](#__RefHeading__36022_1441106636)

[Create user 8](#__RefHeading__36024_1441106636)

[Update user 10](#__RefHeading__36026_1441106636)

[Delete user 12](#__RefHeading__36028_1441106636)

[Permission service 13](#__RefHeading__1559_1441132061)

[Get user permissions 14](#__RefHeading__1561_1441132061)

[Set user permissions 15](#__RefHeading__1563_1441132061)

1. API

Rest services supported http headers

1. Accept: application/json

2. Content-Type: **application/json**

Both headers are required.

Response Format

Response format is json.

For services with GET method instead of using http headers can be used parameter fmt set to json. Just add“fmt=json” to your url.

|  |
| --- |
| * + 1. Implemented services |
| User service |

1. User service
   * 1. Get user

**Endpoint:** [GET] /user/{ID}

**Description:** Returns user data.

**Base user fields:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **Example value** |
| operation | Internal operation name | «get user» |
| status | Status result («ok», «error») | «ok» |
| error | Description of the error if it happens. The value could be null if status=ok. | null |
| data | User data | JSON |
| data.\_id | The id of the user | 563b3f77e8d64c3f52ad008e |
| data.name | The name of the user | user |
| data.password | The hash of the user password (currently it displays a real password) | user |
| data.email | The email of the user | user@sample.com |
| data.published | The flag of user visibility. Only published users can access the website | true |
| data.created | The date the the user has been created | 2015-11-05T11:37:27.203Z |
| data.updated | The date the the user has been updated last time | 2015-11-05T11:37:27.203Z |

**Example request:**

<http://localhost:3000/user/563b3f77e8d64c3f52ad008e>

**Example response (data found):**

{

"operation": "get user",

"status": "ok",

"error": null,

"data": {

"\_id": "563b3f77e8d64c3f52ad008e",

"name": "user",

"password": "user",

"email": "user@sample.com",

"\_\_v": 0,

"published": true,

"updated": "2015-11-05T11:37:27.203Z",

"created": "2015-11-05T11:37:27.203Z"

}

}

**Example responce (error):**

{

"operation": "get user",

"status": "error",

"error": {

// … Stacktrace

},

"data": {}

}

* + 1. Get users

**Endpoint:** [GET] /users

**Description:** Returns users data.

**Fields:** The same like in Get user with one difference. Field «data» consists of array list of all users.

**Example request:**

<http://localhost:3000/users>

**Example response (data found):**

{

"operation": "list users",

"status": "ok",

"error": null,

"data": [

{

"\_id": "563a02626e1f1c3f13bcc253",

"name": "test1",

"password": "",

"email": "sample@sample.com",

"\_\_v": 0,

"published": true,

"updated": "2015-11-04T13:04:34.771Z",

"created": "2015-11-04T13:04:34.770Z"

},

//...,

{

"\_id": "563b4018e8d64c3f52ad008f",

"name": "admin",

"password": "admin",

"email": "admin@sample.com",

"\_\_v": 0,

"published": true,

"updated": "2015-11-05T11:40:08.318Z",

"created": "2015-11-05T11:40:08.317Z"

}

]

}

**Example responce (error):**

{

"operation": "list users",

"status": "error",

"error": {

// … Stacktrace

},

"data": {}

}

* + 1. Create user

**Endpoint:** [PUT] /user/create

**Description:** Creates a new user.

**Request fields:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **Default value** |
| name (mandatory) | The name of the user |  |
| password (mandatory) | The hash of the user password (currently it displays a real password) |  |
| email (mandatory) | The email of the user |  |
| published | The flag of user visibility. Only published users can access the website | true |
| created (fills automatically) | The date the the user has been created | Date.now() |
| updated (fills automatically) | The date the the user has been updated last time | Date.now() |

**Example request:**

<http://localhost:3000/user/create>

**Example request fields:**

{

"name": "admin",

"password" : "admin",

"email": "admin@sample.com"

}

**Example response (data created):**

{

"operation": "create user",

"status": "ok",

"error": null,

"data": {

"\_\_v": 0,

"name": "dshghsfh",

"password": "325fghd",

"email": "admin@sample.com",

"\_id": "563b553a8027e8605880aced",

"published": true,

"updated": "2015-11-05T13:10:18.343Z",

"created": "2015-11-05T13:10:18.343Z"

}

}

**Example response (data not created):**

{

"operation": "create user",

"status": "error",

"error": {

//... Stack trace

},

"data": null

}

* + 1. Update user

**Endpoint:** [PUT] /user/{ID}/update

**Description:** Creates a new user.

**Request fields:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **Default value** |
| name (mandatory) | The name of the user |  |
| password (mandatory) | The hash of the user password (currently it displays a real password) |  |
| email (mandatory) | The email of the user |  |
| published | The flag of user visibility. Only published users can access the website | true |
| created | The date the the user has been created |  |
| updated (fills automatically) | The date the the user has been updated last time. This field fills forced by current time | Date.now() |

**Example request:**

<http://localhost:3000/user/563b145051dadda3493474e0/update>

**Example request:**

{

"name": "admin33aaa",

"password" : "325fghd",

"email": "admin@sample.com"

}

**Example response (data created):**

{

"operation": "update user",

"status": "ok",

"error": null,

"data": {

"\_id": "563b145051dadda3493474e0",

"name": "admin33aaa",

"password": "325fghd",

"email": "admin@sample.com",

"\_\_v": 1,

"published": true,

"updated": "2015-11-05T13:05:15.979Z",

"created": "2015-11-05T08:33:20.923Z"

}

}

**Example response (data not created):**

{

"operation": "create user",

"status": "error",

"error": {

//... Stack trace

},

"data": null

}

* + 1. Delete user

**Endpoint:** [DELETE] /user/{ID}/delete

**Description:** Deletes user

**Example request:**

<http://localhost:3000/user/563b145051dadda3493474e0/delete>

**Example response (user deleted):**

{

"operation": "delete user",

"status": "ok",

"error": null,

"data": null

}

**Example response (user has no deleted):**

{

"operation": "delete user",

"status": "error",

"error": {

//... Stack trace

},

"data": null

}

Permission service

Base permissions:

1. VIEW\_ARTICLE — allows to read articles
2. USE\_SEARCH — allows to access search and see results
3. VIEW\_OWN\_CART — allows to access cart of the current user
4. VIEW\_ANY\_CART — allows to access cart of any user
5. EDIT\_OWN\_CART — allows to have full access for cart of the current user
6. EDIT\_ANY\_CART — allows to have full access for cart of any user
7. VIEW\_OWN\_PROFILE — allows to access current profile for viewing
8. VIEW\_ANY\_PROFILE — allows to access any profile for viewing
9. EDIT\_OWN\_PROFILE — allows to have full access for editing current profile
10. EDIT\_ANY\_PROFILE — allows to have full access for editing any profile
11. ACCESS\_ADMIN\_PANEL — allows to open any administrative page

Get user permissions

**Endpoint:** [GET] /role/get/{USER\_ID}

**Description:** Returns permissions for user USER\_ID

**Example request:** [http://localhost:3000/role/123456789](http://localhost:3000/role/set/123456789)

**Example response (role found):**

{

"operation": "get user permissions",

"status": "ok",

"error": null,

"data": [

"VIEW\_ARTICLE",

"USE\_SEARCH"

]

}

**Example response (role has not found):**

{

"operation": "delete user",

"status": "error",

"error": {

//... Stack trace

},

"data": null

}

Set user permissions

**Endpoint:** [POST] /role/set/{USER\_ID}

**Description:** Sets permissions for user USER\_ID. Implements create/update features. If permission for USER\_ID already exists it updates.

**Example request:** <http://localhost:3000/role/set/123456789>

**Example request fields:**

{

"name": "Sample permissions",

"permissions": [

"VIEW\_ARTICLE",

"USE\_SEARCH"

]

}

**Example response (permission set):**

{

"operation": "set user permissions",

"status": "ok",

"error": null,

"data": {

"\_\_v": 0,

"name": "Sample permissions",

"userId": "123456789",

"\_id": "563b67c24077fd4b5cacff38",

"permissions": [

"VIEW\_ARTICLE",

"USE\_SEARCH"

]

}

}

**Example response (permission not set):**

{

"operation": "delete user",

"status": "error",

"error": {

//... Stack trace

},

"data": null

}