Guide API REST User Service

User API

Table of Contents

NOTE	This document contains the REST API documentation to consume the user service

This documentation is for version {project-version}.

Overview

Chapter 1. HTTP verbs

The services developed for the project try to comply with the HTTP and REST conventions in the use of verbs HTTP.

Verb	Usage
`GET`	Used to retrieve a resource
`POST`	Used to create a new resource
`PUT`	Used to update a resource

Chapter 2. HTTP status codes

The services developed for the project try to comply with the HTTP and REST conventions in the use of codes of HTTP status.

Status code	Usage
`200 OK`	The request completed successfully
`201 Created`	A new resource has been created successfully. The resource's URI is available from the response's `Location` header
`204 No Content`	An update to an existing resource has been applied successfully
`400 Bad Request`	The request was malformed. The response body will include an error providing further information
`401 Unauthorized`	The request resource is restricted. The requested resource is restricted and requires authentication
`404 Not Found`	The requested resource did not exist
`405 Method Not Allowed`	The requested method not supported
`415 Unsupported Media Type`	The Content type is empty or invalid
`422 Unprocessable Entity`	The resource already exist
`5XX Server Error`	Unable to process the request

Chapter 3. Headers

Every response has the following header(s):

Name	Description
`Content-Type`	The Content-Type of the payload, e.g. application/json

Chapter 4. Errors

Whenever an error response (status code >= 400) is returned, the body will contain a JSON object that describes the problem. The error object has the following structure:

Path	Type	Description
`timestamp`	Long	The time, in milliseconds, at which the error occurred
`status`	Number	The HTTP status code, e.g. 400
`exception`	String	The HTTP error that occurred, e.g. Bad Request
`messages`	String[]	A description of the cause of the error(s)
`path`	String	The path to which the request was made

For example, a request that attempts to apply a non-existent tag to a note will produce a 400 Bad Request response:

```
HTTP/1.1 404 Not found
Content-Type: application/json;charset=UTF-8
Content-Length: 208

{
    "timestamp": 1588028874150,
    "status": 404,
    "exception": "NOT_FOUND",
    "messages": [
        "User found"
    ],
    "path": "/user/1"
}
```

Resources

Chapter 5. User

The user resource is used to find information from a user

5.1. Get User By Identifier

A `GET` request will get user Info

5.2. Path Parameters

Table 1. /users/{id}

Parameter	Description
`id`	User name

5.2.1. Example curl request

```
$ curl 'http://localhost:8080/users/1' -i -X GET \
    -H 'Content-Type: application/json'
```

5.2.2. Request

```
GET /users/1 HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

5.2.3. Information Response Fields

Path	Type	Optional	Description
id	String	false	User name
name	String	false	User name
lastname	String	false	User name
age	String	false	User name
email	String	false	User name

5.2.4. Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 113

{
  "id" : 1,
  "name" : "David",
  "lastname" : "Bernal",
  "age" : 31,
  "email" : "leo.bernal1946@gmail.com"
}
```

5.3. Get User Accounts

A `GET` request will get user accounts Info

5.4. Path Parameters

Table 2. /users/{id}/accounts

Parameter	Description
`id`	User name

5.4.1. Example curl request

```
$ curl 'http://localhost:8080/users/1/accounts' -i -X GET \
    -H 'Content-Type: application/json'
```

5.4.2. Request

```
GET /users/1/accounts HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

5.4.3. Information Response Fields

Path	Type	Optional	Description
[].id	String	false	User name
[].type	String	false	User name
[].number	String	false	User name

5.4.4. Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 67

[ {
  "id" : 1,
  "type" : "DEBIT",
  "number" : "435332443242"
} ]
```

5.5. Post Create User

A `POST` request will create a new user

5.5.1. Request Fields

Path	Type	Optional	Description	Constraints	asd
name	String	false	User name	Must not be null	
lastname	String	false	User name	Must not be null	
age	Integer	false	User name	Must not be null	
email	String	false	User name	Must be a well-formed email address- Must not be null	

5.5.2. Example curl request

```
$ curl 'http://localhost:8080/users' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "name" : "David",
  "lastname" : "Bernal",
  "age" : 31,
  "email" : "leo.bernal1946@gmail.com"
}'
```

5.5.3. Example http request

```
POST /users HTTP/1.1
Content-Type: application/json
Content-Length: 101
Host: localhost:8080

{
  "name" : "David",
  "lastname" : "Bernal",
  "age" : 31,
  "email" : "leo.bernal1946@gmail.com"
}
```

5.5.4. Request Body

```
{
  "name" : "David",
  "lastname" : "Bernal",
  "age" : 31,
  "email" : "leo.bernal1946@gmail.com"
}
```

5.5.5. Information Response Fields

Path	Type	Optional	Description
id	String	false	User name
name	String	false	User name
lastname	String	false	User name
age	String	false	User name
email	String	false	User name

5.5.6. Http Response

```
HTTP/1.1 201 Created
Location: http://localhost:8080/users/2
Content-Type: application/json
Content-Length: 113

{
  "id" : 2,
  "name" : "David",
  "lastname" : "Bernal",
  "age" : 31,
  "email" : "leo.bernal1946@gmail.com"
}
```

5.5.7. Response Body

```
{
  "id" : 2,
  "name" : "David",
  "lastname" : "Bernal",
  "age" : 31,
  "email" : "leo.bernal1946@gmail.com"
}
```

5.6. Put Update User

A `PUT` request will update an existing user

5.7. Path Parameters

Unresolved directive in api-guide.adoc - include::/Users/leobernal/Documents/apps/java/spring-rest-docts/target/generated-snippets/update-user/path-parameters.adoc[]

5.7.1. Request Fields

Unresolved directive in api-guide.adoc - include::/Users/leobernal/Documents/apps/java/spring-rest-docts/target/generated-snippets/update-user/request-fields.adoc[]

5.7.2. Example curl request

Unresolved directive in api-guide.adoc - include::/Users/leobernal/Documents/apps/java/spring-rest-docts/target/generated-snippets/update-user/curl-request.adoc[]

5.7.3. Example http request

Unresolved directive in api-guide.adoc - include::/Users/leobernal/Documents/apps/java/spring-rest-docts/target/generated-snippets/update-user/http-request.adoc[]

5.7.4. Information Response Body

Unresolved directive in api-guide.adoc - include::/Users/leobernal/Documents/apps/java/spring-rest-docts/target/generated-snippets/update-user/response-body.adoc[]

5.8. Delete User

A `DELETE` request will delete an existing user

5.9. Path Parameters

Table 3. /users/{id}

Parameter	Description
`id`	User name

5.9.1. Example curl request

```
$ curl 'http://localhost:8080/users/1' -i -X DELETE \
    -H 'Content-Type: application/json'
```

5.9.2. Example http request

```
DELETE /users/1 HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

5.10. Get Users

A `GET` request will get all existing users

5.10.1. Example curl request

```
$ curl 'http://localhost:8080/users' -i -X GET \
    -H 'Content-Type: application/json'
```

5.10.2. Example http request

```
GET /users HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

5.10.3. Information Response Fields

Path	Type	Optional	Description
[].id	String	false	User name
[].name	String	false	User name
[].lastname	String	false	User name
[].age	String	false	User name
[].email	String	false	User name

5.10.4. Http Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 117

[ {
  "id" : 1,
  "name" : "David",
  "lastname" : "Bernal",
  "age" : 31,
  "email" : "leo.bernal1946@gmail.com"
} ]
```

5.10.5. Response Body

```
[ {
  "id" : 1,
  "name" : "David",
  "lastname" : "Bernal",
  "age" : 31,
  "email" : "leo.bernal1946@gmail.com"
} ]
```