API Documentation for EEPPI

3.6 **Task Property**
- **POST /rest/api/1/taskProperty** (Creates a new Task Property.)
- **GET /rest/api/1/taskProperty** (Reads all Task Properties.)
- **POST /rest/api/1/taskProperty/<id>** (Updates an existing Task Property with new data.)
- **GET /rest/api/1/taskProperty/<id>** (Reads a Task Property.)
- **DELETE /rest/api/1/taskProperty/<id>** (Deletes a Task Property.)

3.7 **Task Template**
- **POST /rest/api/1/taskTemplate** (Creates a new Task Template of which (concrete) Tasks then can be generated.)
- **GET /rest/api/1/taskTemplate** (Reads all Task Templates.)
- **POST /rest/api/1/taskTemplate/<id>** (Updates an existing Task Template with new data.)
- **GET /rest/api/1/taskTemplate/<id>** (Reads a Task Template.)
- **DELETE /rest/api/1/taskTemplate/<id>** (Deletes a Task Template.)
- **POST /rest/api/1/taskTemplate/<id>/addProperty** (Adds a new property to an existing Task Template.)
- **POST /rest/api/1/taskTemplate/<id>/properties/<taskTemplate>** (Updates a task property value.)
- **DELETE /rest/api/1/taskTemplate/<id>/properties/<taskTemplate>** (Deletes a task property value.)

3.8 **PPTAccount**
- **POST /rest/api/1/user/pptAccount** (Stores a new login information for a Project Planning Tool.)
- **GET /rest/api/1/user/pptAccount** (Returns all login information (but the password) for the currently logged in user for Project Planning Tools.)
- **POST /rest/api/1/user/pptAccount/<id>** (Updates login information for a Project Planning Tool on the server.)
- **GET /rest/api/1/user/pptAccount/<id>** (Returns one login information (but the password) for the currently logged in user for Project Planning Tools.)
- **DELETE /rest/api/1/user/pptAccount/<id>** (Deletes login information for a Project Planning Tool on the server.)

3.9 **Project**
- **GET /rest/api/1/project** (Returns all Projects.)
- **GET /rest/api/1/project/<id>** (Returns one Project.)

3.10 **User**
- **POST /rest/api/1/user/changePassword** (This changes the password of an EEPPI-user.)
- **POST /rest/api/1/user/login** (Checks the login information for the user and if the login is successful a cookie is set.)
- **GET /rest/api/1/user/loginStatus** (Returns the login status for the currently logged in user and a Json representation of it.)
- **POST /rest/api/1/user/logout** (Does log out the currently logged in user by removing the cookie.)
- **POST /rest/api/1/user/register** (This creates a new EEPPI-user.)

1 Authentication

Some methods need authentication. The authentication can be made by providing a cookie generated by /rest/api/1/user/login or by providing HTTP Basic Authentication. The HTTP Basic Authentication could be used with something like http://username:password@localhost:9000/rest/api/1/user/pptAccount?basicAuth=true. The GET parameter "basicAuth=true" lets the server enable HTTP Basic Authentication. A response with status code 401 (Unauthorized) is returned, when authentication information are required but are not provided.

2 Parameters

All method parameters can be passed in two ways:

- Normal POST-data (this is also used in this documentation):

```
curl --request POST --data "theKey=theValue&theKey2=theValue2"
http://localhost:9000/...
```

- As a JSON-object as binary data:

```
curl -H 'Content-Type: application/json;charset=UTF-8' --data-binary
'{"theKey":"theValue","theKey2":"theValue2"}' http://localhost:9000/...
```

  If it's passed as a JSON-object and some parameter reference another object, the reference can be passed in two ways:

  - As a numeric ID of the referenced object:

```
curl -H 'Content-Type: application/json;charset=UTF-8' --data-binary
'{"id":61,"name":"Example processor","project":63,"code":"function(num) {
return num*num; }"}' http://localhost:9000/rest/api/1/processor
```

  - As a full object containing at least an ID-property:

```
curl -H 'Content-Type: application/json;charset=UTF-8' --data-binary
'{"id":61,"name":"Example processor","project":{"id":63,"name":"The Example
Project"},"code":"function(num) { return num*num; }"}' http://localhost:9000
/rest/api/1/processor
```

    If it's passed as a full object and the method returns the referenced object again, it's returned as it got passed. However, it's not updated on the server.

3 Resources

3.1 Decision Knowledge System

POST /rest/api/1/dks

Stores a new DKS (Decision Knowledge System).

Authentication is required to use this method.

Parameters

Name	Type	Description
name	String	the new name of the new DKS
url	String	the URL where the DKS is

Responses

Status	Description
400	If the DKS could not be stored.
200	If the DKS was stored. It is also returned in Json form.

Example 1

```
curl --request POST --data "name=The DKS&url=" http://localhost:9000/rest/api
/1/dks
```

The previous command **did** return with status code **500 (Internal Server Error)** and yielded the following:

```
This is not implemented yet, it is a known limitation of this version.
```

Example 2

```
curl --request POST --data "name=The DKS&url=http://the-dks.ch"
http://localhost:9000/rest/api/1/dks
```

The previous command **did** return with status code **500 (Internal Server Error)** and yielded the following:

```
This is not implemented yet, it is a known limitation of this version.
```

GET /rest/api/1/dks

Returns all DKSs (Decision Knowledge Systems).

Authentication is required to use this method.

Responses

Status	Description
200	It's always a list returned containing all (but if there is none also none) entities.

Example 1

```
curl http://localhost:9000/rest/api/1/dks
```

The previous command **did** return with status code **200 (OK)** and yielded the following:


```
{"items":[{"id":1000000000000000071,"name":"Example DKS","url":"http://an-
example-dks.com"},{"id":1000000000000000073,"name":"Example
DKS","url":"http://an-example-dks.com"},
{"id":1000000000000000074,"name":"Example DKS","url":"http://an-example-
dks.com"},{"id":1000000000000000075,"name":"Example DKS","url":"http://an-
example-dks.com"}]}
```


POST /rest/api/1/dks/<id>

Updates a DKS (Decision Knowledge System).

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the mapping to update
name	String	the new name of the new DKS to update
url	String	the URL where the DKS is

Responses

Status	Description
404	If no entity with the given ID exists.
400	If the request parameter contain errors.
200	The new created entity is returned

Example 1

```
curl --request POST --data "name=Example DKS&url=http://a-dks.com"
http://localhost:9000/rest/api/1/dks/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find DecisionKnowledgeSystem with id 9999."
```

Example 2

```
curl --request POST --data "name=Example DKS&url=http://an-example-dks.com"
http://localhost:9000/rest/api/1/dks/1000000000000000073
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000073,"name":"Example DKS","url":"http://an-example-
dks.com"}
```

Example 3

```
curl --request POST --data "name=Example DKS&url=" http://localhost:9000
/rest/api/1/dks/1000000000000000074
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"url":["This field is required"]}
```

GET /rest/api/1/dks/<id>

Returns one DKS (Decision Knowledge System).

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to get

Responses

Status	Description
404	If no entity with the given ID exists.
200	If it's found, it's returned.

Example 1

```
curl http://localhost:9000/rest/api/1/dks/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find DecisionKnowledgeSystem with id 9999."
```

Example 2

```
curl http://localhost:9000/rest/api/1/dks/1000000000000000071
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000071,"name":"Example DKS","url":"http://an-example-
dks.com"}
```

DELETE /rest/api/1/dks/<id>

Deletes a DKS (Decision Knowledge System).

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to delete

Responses

Status	Description
404	If the DKS to delete could not be found.
204	If the DKS was deleted.

Example 1

```
curl --request DELETE http://localhost:9000/rest/api/1/dks/9999
```

The previous command **did** return with status code **500 (Internal Server Error)** and yielded the following:

```
This is not implemented yet, it is a known limitation of this version.
```

Example 2

```
curl --request DELETE http://localhost:9000/rest/api/1/dks/1000000000000000075
```

The previous command **did** return with status code **500 (Internal Server Error)** and yielded the following:

```
This is not implemented yet, it is a known limitation of this version.
```

GET /rest/api/1/dks/getFromDKS?url=<url>

Redirects a request to a remote server using GET to avoid restrictions with Cross Origin Requests.

Authentication is required to use this method.

Parameters

Name	Type	Description
url	String	The full URL of the remote server to GET from.

Responses

Status	Description
400	If there is an error during preparation of the request for the remote server.
-	The return value from the remote server is returned.

Example 1

```
curl http://localhost:9000/rest/api/1/dks/getFromDKS?url=http:
//headers.jsontest.com/
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"Host":"headers.jsontest.com","User-Agent":"NING/1.0","Accept":"*/*"}
```

Example 2

```
curl http://localhost:9000/rest/api/1/dks/getFromDKS?url=hatetepe?__no-valid-url
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
Could not load hatetepe?__no-valid-url
```

3.2 Decision Knowledge System Mapping

POST /rest/api/1/dksMapping

Creates a new Mapping for a DKS Node and a Task Template.

Authentication is required to use this method.

Parameters

Name	Type	Description
taskTemplate	String	The Task Template to map to a DKS Node
dksNode	String	The DKS Node to map to a Task Template

Responses

Status	Description
400	If the request parameter contain errors.
200	The new created entity is returned.

Example 1

```
curl --request POST --data "taskTemplate=10000000000000000033&dksNode=80"
http://localhost:9000/rest/api/1/dksMapping
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":3353,"taskTemplate":{"id":100000000000000033,"properties":
[],"parent":null,"name":"My example Task Template"},"dksNode":"80"}
```

Example 2

```
curl --request POST --data "taskTemplate=9999&dksNode=87" http://localhost:9000
/rest/api/1/dksMapping
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"taskTemplate":["Invalid value"]}
```

GET /rest/api/1/dksMapping

Reads all Mappings for DKS Nodes and Task Templates.

Authentication is required to use this method.

Responses

Status	Description
200	It's always a list returned containing all (but if there is none also none) entities.

Example 1

```
curl http://localhost:9000/rest/api/1/dksMapping
```

The previous command **did** return with status code **200 (OK)** and yielded the following:


```
{"items":[{"id":3308,"taskTemplate":{"id":3307,"properties":
[],"parent":null,"name":"My example Task Template
7"},"dksNode":"1000000000000000064"},{"id":3353,"taskTemplate":
{"id":1000000000000000033,"properties":[],"parent":null,"name":"My example Task
Template"},"dksNode":"80"},{"id":1000000000000000035,"taskTemplate":
{"id":3303,"properties":[],"parent":null,"name":"My example Task Template
2"},"dksNode":"87"},{"id":1000000000000000037,"taskTemplate":
{"id":3301,"properties":[],"parent":null,"name":"My example Task Template
2"},"dksNode":"87"},{"id":1000000000000000039,"taskTemplate":
{"id":3305,"properties":[],"parent":null,"name":"My example Task Template
2"},"dksNode":"87"}]}
```


POST /rest/api/1/dksMapping/<id>

Updates an existing Mapping for a DKS Node and a Task Template.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the Mapping to update
taskTemplate	String	The Task Template to map to a DKS Node
dksNode	String	The DKS Node to map to a Task Template

Responses

Status	Description
404	If no entity with the given ID exists.
400	If the request parameter contain errors.

200	The new created entity is returned

Example 1

```
curl --request POST --data "taskTemplate=10000000000000033&dksNode=87"
http://localhost:9000/rest/api/1/dksMapping/100000000000000037
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":100000000000000037,"taskTemplate":{"id":100000000000000033,"properties":
[],"parent":null,"name":"My example Task Template"},"dksNode":"87"}
```

Example 2

```
curl --request POST --data "taskTemplate=9999&dksNode=87" http://localhost:9000
/rest/api/1/dksMapping/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Decision Knowledge System Mapping with id 9999."
```

GET /rest/api/1/dksMapping/<id>

Reads a Mapping for a DKS Node and a Task Template.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to get

Responses

Status	Description
404	If no entity with the given ID exists.
200	If it's found, it's returned.

Example 1

```
curl http://localhost:9000/rest/api/1/dksMapping/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the

following:

```
"Could not find Decision Knowledge System Mapping with id 9999."
```

Example 2

```
curl http://localhost:9000/rest/api/1/dksMapping/1000000000000000035
```

The previous command **did** return with status code **200 (OK)** and yielded the following:


```
{"id":1000000000000000035,"taskTemplate":{"id":3303,"properties":
[],"parent":null,"name":"My example Task Template 2"},"dksNode":"87"}
```

DELETE /rest/api/1/dksMapping/<id>

Deletes a Mapping for a DKS Node and a Task Template.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to delete

Responses

Status	Description
404	If no entity with the given ID exists.
409	If the entity could not be deleted.
204	If the entity is successfully deleted.

Example 1

```
curl --request DELETE http://localhost:9000/rest/api/1/dksMapping/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Decision Knowledge System Mapping with id 9999."
```

Example 2

```
curl --request DELETE http://localhost:9000/rest/api/1/dksMapping
/1000000000000000039
```

The previous command **did** return with status code **204 (No Content)** and yielded the following:

```

```

GET /rest/api/1/dksMapping/byDKSNode/<dksNode>

Reads all Mappings for a given DKS Node.

Authentication is required to use this method.

Parameters

Name	Type	Description
dksNode	String	The id of the DKS Node to get the mappings for

Responses

Status	Description
200	A list of all Mappings is returned, if no mapping could be found, an empty list is returned.

Example 1

```
curl http://localhost:9000/rest/api/1/dksMapping/byDKSNode/9999
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"items":[]}
```

Example 2

```
curl http://localhost:9000/rest/api/1/dksMapping/byDKSNode/1000000000000000064
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"items":[{"id":3308,"taskTemplate":{"id":3307,"properties":
[],"parent":null,"name":"My example Task Template
7"},"dksNode":"1000000000000000064"}]}
```

3.3 Processor

POST /rest/api/1/processor

Persists the given processor

Authentication is required to use this method.

Parameters

Name	Type	Description
name	String	a speaking name to identify the processor
project	String	a reference (ID) to the Project

code	String	JavaScript code of the processor

Responses

Status	Description
400	If the processor could not be stored.
200	If the processor was stored successfully. The processor is also returned as JSON object.

Example 1

```
curl --request POST --data "name=Example processor&project=9999&
code=function(num) { return num*num; }" http://localhost:9000/rest/api
/1/processor
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"project":["Invalid value"]}
```

Example 2

```
curl --request POST --data "name=Example processor&project=10000000000000042&
code=function(num) { return num*num; }" http://localhost:9000/rest/api
/1/processor
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":3354,"name":"Example processor","project":
{"id":10000000000000042,"name":"The Example Project"},"code":"function(num) {
return num*num; }"}
```

GET /rest/api/1/processor

Returns all processors.

Authentication is required to use this method.

Responses

Status	Description
200	It's always a list returned containing all (but if there is none also none) entities.

Example 1

```
curl http://localhost:9000/rest/api/1/processor
```

The previous command **did** return with status code **200 (OK)** and yielded the following:


```
{"items":[{"id":3354,"name":"Example processor","project":
{"id":1000000000000000042,"name":"The Example Project"},"code":"function(num) {
return num*num; }"},{"id":1000000000000000054,"name":"Example
Processor","project":{"id":3315,"name":"Example project"},"code":"function(a) {
return a+'.'+a; }"},{"id":1000000000000000055,"name":"Example
Processor","project":{"id":3310,"name":"Example project"},"code":"function(a) {
return a+'.'+a; }"},{"id":1000000000000000056,"name":"Example
Processor","project":{"id":3312,"name":"Example project"},"code":"function(a) {
return a+'.'+a; }"},{"id":1000000000000000057,"name":"Example
Processor","project":{"id":3317,"name":"Example project"},"code":"function(a) {
return a+'.'+a; }"}]}
```


POST /rest/api/1/processor/<id>

Updates a processor.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the processor to update
name	String	a speaking name to identify the processor
project	String	a reference (ID) to the Project
code	String	JavaScript code of the processor

Responses

Status	Description
404	If no entity with the given ID exists.
400	If the request parameter contain errors.
200	The new created entity is returned

Example 1

```
curl --request POST --data "name=Example processor&project=9998&code=function()
{}" http://localhost:9000/rest/api/1/processor/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Processor with id 9999."
```

Example 2

```
curl --request POST --data "name=Example processor 2&project=9898&
code=function() {}" http://localhost:9000/rest/api/1/processor
/1000000000000000055
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"project":["Invalid value"]}
```

Example 3

```
curl --request POST --data "name=Example processor&project=10000000000000000043&
code=function(a) { return a+'.'+a; }" http://localhost:9000/rest/api/1/processor
/1000000000000000056
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000056,"name":"Example processor","project":
{"id":1000000000000000043,"name":"The Example Project"},"code":"function(a) {
return a+'.'+a; }"}
```

GET /rest/api/1/processor/<id>

Returns one processor.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to get

Responses

Status	Description
404	If no entity with the given ID exists.
200	If it's found, it's returned.

Example 1

```
curl http://localhost:9000/rest/api/1/processor/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Processor with id 9999."
```

Example 2

```
curl http://localhost:9000/rest/api/1/processor/1000000000000000054
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000054,"name":"Example Processor","project":
{"id":3315,"name":"Example project"},"code":"function(a) { return a+'.'+a; }"}
```

DELETE /rest/api/1/processor/<id>

Deletes a processor.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to delete

Responses

Status	Description
404	If the processor to delete could not be found.
204	If the processor was deleted.

Example 1

```
curl --request DELETE http://localhost:9000/rest/api/1/processor/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Processor with id 9999."
```

Example 2

```
curl --request DELETE http://localhost:9000/rest/api/1/processor
/1000000000000000057
```

The previous command **did** return with status code **204 (No Content)** and yielded the following:

3.4 Project Planning Tool

GET /rest/api/1/ppt

Returns all Project Planning Tools.

Authentication is required to use this method.

Responses

Status	Description
200	It's always a list returned containing all (but if there is none also none) entities.

Example 1

```
curl http://localhost:9000/rest/api/1/ppt
```

The previous command **did** return with status code **200 (OK)** and yielded the following:


```
{"items":[{"id":3321,"name":"Example PPT"},{"id":3324,"name":"Example PPT"},
{"id":3327,"name":"Example PPT"},{"id":3330,"name":"Example PPT"},
{"id":3348,"name":"Example Project Planning Tool"},{"id":3350,"name":"Example
Project Planning Tool"},{"id":1000000000000000005,"name":"Example Jira"},
{"id":1000000000000000041,"name":"Example Jira"},
{"id":1000000000000000051,"name":"Example Jira"}]}
```


GET /rest/api/1/ppt/<id>

Returns one Project Planning Tool.

Authentication is required to use this method.

Parameters

Name	Type	Description

id	Long	The id of the entity to get

Responses

Status	Description
404	If no entity with the given ID exists.
200	If it's found, it's returned.

Example 1

```
curl http://localhost:9000/rest/api/1/ppt/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Project Planning Tool with id 9999."
```

Example 2

```
curl http://localhost:9000/rest/api/1/ppt/100000000000000051
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":100000000000000051,"name":"Example Jira"}
```

POST /rest/api/1/ppt/createPPTTask

Creates a Task on a remote Project Planning Tool Server and stores the creation on the server.

Authentication is required to use this method.

Parameters

Name	Type	Description
account	Long	The id of an EEPPI-account of the currently logged in user.
path	String	The path on the remote server beginning with a '/'
content	JsonNode	Json-data to be sent to the remote server
taskTemplate	Long	The id of a TaskTemplate of which this Task is created.

taskProperties[]	String	This parameter can be passed multiple times. It represents a Task Property Value for this Task. The Format is "{ID of the Task Property}-{The Value of it}".
project	Long	The id of a Project in which this Task is created.

Responses

Status	Description
400	If there is an error during preparation of the request for the remote server.
502	If the remote server could not be found.
504	If the remote server did not respond.
-	The return value from the remote server is returned (the Json if it's a Json or a Json containing the type and the content as a simple Json-Object).

Example 1

```
curl --request POST --data "account=100&path=/rest/api/2/issue/&content={
    "fields": {
       "project":
       {
          "key": "PRV"
       },
       "summary": "My generated issue",
       "description": "This is an issue, which is created by EEPPI over the
API",
       "issuetype": {
          "name": "Task"
       }
   }
}&taskTemplate=51&taskProperties[]=53-Example Value&project=55"
http://localhost:9000/rest/api/1/ppt/createPPTTask
```

The previous command **would probably** return with status code **201** and yield the following:

```
{
    "id": "10000",
    "key": "PRV-24",
    "self": "http://jira.example.ch/jira/rest/api/2/issue/10000"
}
```

Example 2

```
curl --request POST --data "account=100&path=/index.html&content=
{}&taskTemplate=51&taskProperties[]=53-Example Value&project=55"
http://localhost:9000/rest/api/1/ppt/createPPTTask
```

The previous command **would probably** return with status code **200** and yield the following:

```
{
    "content": "<html><head></head><body>...</body></html>",
    "type": "text/html; charset=utf-8"
}
```

Example 3

```
curl --request POST --data "account=not a number&path=not a path&content=no
Json&taskTemplate=wrongFormat&taskProperties[]=9999" http://localhost:9000
/rest/api/1/ppt/createPPTTask
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"taskProperties[0]":["Invalid value"],"project":["This field is
required"],"taskTemplate":["Invalid value"],"content":["Invalid
value"],"account":["Invalid value"]}
```

3.5 Request Template

POST /rest/api/1/requestTemplate

Stores a new Request Template for sending a Task to a Project Planning Tool.

Authentication is required to use this method.

Parameters

Name	Type	Description
name	String	the new name of the new Request Template
ppt	String	a reference (ID) to the Project Planning Tool
project	String	a reference (ID) to the Project
url	String	the URL to call on the Project Planning Tool (only the part after the domain and port, beginning with a "/")
requestTemplate	String	the template for the request to be performed (including markup for usages of Processors)

Responses

Status	Description
400	If the Request Template could not be stored.
200	If the Request Template was stored. It is also returned in Json form.

Example 1

```
curl --request POST --data "name=Request Template name&ppt=9999&project=9998&
url=/some/target&requestTemplate={}" http://localhost:9000/rest/api
/1/requestTemplate
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"ppt":["Invalid value"],"project":["Invalid value"]}
```

Example 2

```
curl --request POST --data "name=Request Template name&ppt=1000000000000000041&
project=1000000000000000042&url=/example/target&requestTemplate={}"
http://localhost:9000/rest/api/1/requestTemplate
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":3355,"ppt":{"id":1000000000000000041,"name":"Example Jira"},"project":
{"id":1000000000000000042,"name":"The Example Project"},"name":"Request
Template name","url":"/example/target","requestBodyTemplate":null}
```

GET /rest/api/1/requestTemplate

Returns all Request Templates for sending a Task to a Project Planning Tool.

Authentication is required to use this method.

Responses

Status	Description
200	It's always a list returned containing all (but if there is none also none) entities.

Example 1

```
curl http://localhost:9000/rest/api/1/requestTemplate
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"items":[{"id":3355,"ppt":{"id":1000000000000000041,"name":"Example
Jira"},"project":{"id":1000000000000000042,"name":"The Example
Project"},"name":"Request Template name","url":"/example
/target","requestBodyTemplate":null},{"id":1000000000000000045,"ppt":
{"id":3327,"name":"Example PPT"},"project":{"id":3328,"name":"Example
Project"},"name":"My Request Template","url":"/example
/endpoint","requestBodyTemplate":"{\"name\":\"${title}\"}"},
{"id":1000000000000000047,"ppt":{"id":3321,"name":"Example PPT"},"project":
{"id":3322,"name":"Example Project"},"name":"My Request
Template","url":"/example/endpoint","requestBodyTemplate":"{\"name\":\"${title}
```

```
\"}"},{"id":1000000000000000048,"ppt":{"id":3330,"name":"Example
PPT"},"project":{"id":3331,"name":"Example Project"},"name":"My Request
Template","url":"/example/endpoint","requestBodyTemplate":"{\"name\":\"${title}
\"}"},{"id":1000000000000000049,"ppt":{"id":3324,"name":"Example
PPT"},"project":{"id":3325,"name":"Example Project"},"name":"My Request
Template","url":"/example/endpoint","requestBodyTemplate":"{\"name\":\"${title}
\"}"}]}
```


POST /rest/api/1/requestTemplate/<id>

Updates a Request Template for sending a Task to a Project Planning Tool.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the Request Template to update
name	String	the new name of the Request Template to update
ppt	String	a reference (ID) to the Project Planning Tool
project	String	a reference (ID) to the Project
url	String	the URL to call on the Project Planning Tool (only the part after the domain and port, beginning with a "/")
requestTemplate	String	the template for the request to be performed (including markup for usages of Processors)

Responses

Status	Description
404	If no entity with the given ID exists.
400	If the request parameter contain errors.
200	The new created entity is returned

Example 1

```
curl --request POST --data "name=Request Template name&ppt=9999&project=9998&
url=/asdf&requestTemplate={}" http://localhost:9000/rest/api
/1/requestTemplate/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Request Template with id 9999."
```

Example 2

```
curl --request POST --data "name=Request Template name&ppt=9988&project=9977&
url=/example/target&requestTemplate={}" http://localhost:9000/rest/api
/1/requestTemplate/1000000000000000047
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"ppt":["Invalid value"],"project":["Invalid value"]}
```

Example 3

```
curl --request POST --data "name=Request Template name&ppt=10000000000000041&
project=10000000000000042&url=/example/target&requestTemplate={}"
http://localhost:9000/rest/api/1/requestTemplate/1000000000000000049
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000049,"ppt":{"id":10000000000000041,"name":"Example
Jira"},"project":{"id":10000000000000042,"name":"The Example
Project"},"name":"Request Template name","url":"/example
/target","requestBodyTemplate":null}
```

GET /rest/api/1/requestTemplate/<id>

Returns one Request Template for sending a Task to a Project Planning Tool.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to get

Responses

Status	Description
404	If no entity with the given ID exists.
200	If it's found, it's returned.

Example 1

```
curl http://localhost:9000/rest/api/1/requestTemplate/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Request Template with id 9999."
```

Example 2

```
curl http://localhost:9000/rest/api/1/requestTemplate/1000000000000000045
```

The previous command **did** return with status code **200 (OK)** and yielded the following:


```
{"id":1000000000000000045,"ppt":{"id":3327,"name":"Example PPT"},"project":
{"id":3328,"name":"Example Project"},"name":"My Request
Template","url":"/example/endpoint","requestBodyTemplate":"{\"name
\":\"${title}\"}"}
```


DELETE /rest/api/1/requestTemplate/<id>

Deletes a Request Template for sending a Task to a Project Planning Tool.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to delete

Responses

Status	Description
404	If the Request Template to delete could not be found.
204	If the Request Template was deleted.

Example 1

```
curl --request DELETE http://localhost:9000/rest/api/1/requestTemplate/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Request Template with id 9999."
```

Example 2

```
curl --request DELETE http://localhost:9000/rest/api/1/requestTemplate
/1000000000000000048
```

The previous command **did** return with status code **204 (No Content)** and yielded the following:

```
```

3.6 Task Property

POST /rest/api/1/taskProperty

Creates a new Task Property.

Authentication is required to use this method.

Parameters

Name	Type	Description
name	String	The name of the new Task Property

Responses

Status	Description
400	If the request parameter contain errors.
200	The new created entity is returned.

Example 1

```
curl --request POST --data "name=A new Task Property" http://localhost:9000
/rest/api/1/taskProperty
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":3356,"name":"A new Task Property"}
```

GET /rest/api/1/taskProperty

Reads all Task Properties.

Authentication is required to use this method.

Responses

Status	Description
200	It's always a list returned containing all (but if there is none also none) entities.

Example 1

```
curl http://localhost:9000/rest/api/1/taskProperty
```

The previous command **did** return with status code **200 (OK)** and yielded the following:


```
{"items":[{"id":3340,"name":"My example Task Property 2"},{"id":3343,"name":"My
example Task Property 2"},{"id":3356,"name":"A new Task Property"},
{"id":1000000000000000008,"name":"My example Task Property"},
{"id":1000000000000000013,"name":"My example Task Property"},
{"id":1000000000000000023,"name":"My example Task Property"},
{"id":1000000000000000027,"name":"My example Task Property"}]}
```


POST /rest/api/1/taskProperty/<id>

Updates an existing Task Property with new data.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the Task Property to update
name	String	The new name of the Task Property

Responses

Status	Description
404	If no entity with the given ID exists.
400	If the request parameter contain errors.
200	The new created entity is returned

Example 1

```
curl --request POST --data "name=My beautiful task property"
http://localhost:9000/rest/api/1/taskProperty/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the

following:

```
"Could not find Task Property with id 9999."
```

Example 2

```
curl --request POST --data "name=My example Task Property" http://localhost:9000
/rest/api/1/taskProperty/1000000000000000008
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000008,"name":"My example Task Property"}
```

GET /rest/api/1/taskProperty/<id>

Reads a Task Property.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to get

Responses

Status	Description
404	If no entity with the given ID exists.
200	If it's found, it's returned.

Example 1

```
curl http://localhost:9000/rest/api/1/taskProperty/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Task Property with id 9999."
```

Example 2

```
curl http://localhost:9000/rest/api/1/taskProperty/1000000000000000013
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000013,"name":"My example Task Property"}
```

DELETE /rest/api/1/taskProperty/<id>

Deletes a Task Property.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to delete

Responses

Status	Description
404	If no entity with the given ID exists.
409	If the entity could not be deleted.
204	If the entity is successfully deleted.

Example 1

```
curl --request DELETE http://localhost:9000/rest/api/1/taskProperty/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Task Property with id 9999."
```

Example 2

```
curl --request DELETE http://localhost:9000/rest/api/1/taskProperty
/1000000000000000023
```

The previous command **did** return with status code **204 (No Content)** and yielded the following:

```
```

3.7 Task Template

POST /rest/api/1/taskTemplate

Creates a new Task Template of which (concrete) Tasks then can be generated.

Authentication is required to use this method.

Parameters

Name	Type	Description
name	String	The name of the new Task Template

Responses

Status	Description
400	If the request parameter contain errors.
200	The new created entity is returned.

Example 1

```
curl --request POST --data "name=A new Task Template" http://localhost:9000
/rest/api/1/taskTemplate
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":3357,"properties":[],"parent":null,"name":"A new Task Template"}
```

GET /rest/api/1/taskTemplate

Reads all Task Templates.

Authentication is required to use this method.

Responses

Status	Description
200	It's always a list returned containing all (but if there is none also none) entities.

Example 1

```
curl http://localhost:9000/rest/api/1/taskTemplate
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"items":[{"id":3357,"properties":[],"parent":null,"name":"A new Task
```

```
Template"},{"id":1000000000000000008,"properties":[],"parent":null,"name":"My
example Task Template"},{"id":1000000000000000013,"properties":
[],"parent":null,"name":"My example Task Template"},
{"id":1000000000000000025,"properties":[],"parent":null,"name":"My example Task
Template"},{"id":1000000000000000031,"properties":[],"parent":null,"name":"My
example Task Template"},{"id":1000000000000000033,"properties":
[],"parent":null,"name":"My example Task Template"},{"id":3301,"properties":
[],"parent":null,"name":"My example Task Template 2"},{"id":3303,"properties":
[],"parent":null,"name":"My example Task Template 2"},{"id":3305,"properties":
[],"parent":null,"name":"My example Task Template 2"},{"id":3341,"properties":
[{"id":1000000000000000029,"property":{"id":3340,"name":"My example Task
Property 2"},"value":"My example Value"}],"parent":null,"name":"My example Task
Template 2"},{"id":3344,"properties":[{"id":1000000000000000030,"property":
{"id":3343,"name":"My example Task Property 2"},"value":"My example
Value"}],"parent":null,"name":"My example Task Template
2"},{"id":3307,"properties":[],"parent":null,"name":"My example Task Template
7"}]}
```


POST /rest/api/1/taskTemplate/<id>

Updates an existing Task Template with new data.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the Task Template to update
name	String	The new name of the Task Template

Responses

Status	Description
404	If no entity with the given ID exists.
400	If the request parameter contain errors.
200	The new created entity is returned

Example 1

```
curl --request POST --data "name=My beautiful task template"
http://localhost:9000/rest/api/1/taskTemplate/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:


```
"Could not find Task Template with id 9999."
```

Example 2

```
curl --request POST --data "name=My example Task Template" http://localhost:9000
/rest/api/1/taskTemplate/1000000000000000008
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000008,"properties":[],"parent":null,"name":"My example Task
Template"}
```

GET /rest/api/1/taskTemplate/<id>

Reads a Task Template.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to get

Responses

Status	Description
404	If no entity with the given ID exists.
200	If it's found, it's returned.

Example 1

```
curl http://localhost:9000/rest/api/1/taskTemplate/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Task Template with id 9999."
```

Example 2

```
curl http://localhost:9000/rest/api/1/taskTemplate/1000000000000000008
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000008,"properties":[],"parent":null,"name":"My example Task
Template"}
```

DELETE /rest/api/1/taskTemplate/<id>

Deletes a Task Template.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to delete

Responses

Status	Description
404	If no entity with the given ID exists.
409	If the entity could not be deleted.
204	If the entity is successfully deleted.

Example 1

```
curl --request DELETE http://localhost:9000/rest/api/1/taskTemplate/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Task Template with id 9999."
```

Example 2

```
curl --request DELETE http://localhost:9000/rest/api/1/taskTemplate
/1000000000000000013
```

The previous command **did** return with status code **204 (No Content)** and yielded the following:

```

```

POST /rest/api/1/taskTemplate/<id>/addProperty

Adds a new property to an existing Task Template.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the Task Template
property	String	The id of the Task Property
value	String	The value of the Property

Responses

Status	Description
404	If no Task Template with the given ID exists
400	If the request parameters contain errors.
200	The Task Template containing the new Property is returned

Example 1

```
curl --request POST --data "property=8888&value=My beautiful task value"
http://localhost:9000/rest/api/1/taskTemplate/9999/addProperty
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Task Template with id 9999."
```

Example 2

```
curl --request POST --data "property=1000000000000000027&value=My example Task
Value" http://localhost:9000/rest/api/1/taskTemplate/1000000000000000025
/addProperty
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000025,"properties":[{"id":3358,"property":
{"id":1000000000000000027,"name":"My example Task Property"},"value":"My
example Task Value"}],"parent":null,"name":"My example Task Template"}
```

POST /rest/api/1/taskTemplate/<id>/properties/<taskTemplate>

Updates a task property value.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the Task Template Value
taskTemplate	Long	The id of the Task Template
property	String	The id of the Task Property
value	String	The value of the Property

Responses

Status	Description
404	If no Task Template or Task Property Value with the given ID exists
400	If the request parameters contain errors.
200	The Task Template containing the updated Property is returned

Example 1

```
curl --request POST --data "property=8888&value=My beautiful task template"
http://localhost:9000/rest/api/1/taskTemplate/9999/properties/7777
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Task Template with id 7777."
```

Example 2

```
curl --request POST --data "property=1000000000000000027&value=My example
Value" http://localhost:9000/rest/api/1/taskTemplate/1000000000000000029
/properties/100000000000000025
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Task Template with id 100000000000000025."
```

DELETE /rest/api/1/taskTemplate/<id>/properties/<taskTemplate>

Deletes a task property value.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the Task Template Value
taskTemplate	Long	The id of the Task Template

Responses

Status	Description
404	If no Task Template or Task Property Value with the given ID exists
200	The Task Template containing the removed Property is returned

Example 1

```
curl --request DELETE http://localhost:9000/rest/api/1/taskTemplate
/9999/properties/7777
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Task Template with id 7777."
```

Example 2

```
curl --request DELETE http://localhost:9000/rest/api/1/taskTemplate
/1000000000000000030/properties/1000000000000000031
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Task Template with id 1000000000000000031."
```

3.8 PPTAccount

POST /rest/api/1/user/pptAccount

Stores a new login information for a Project Planning Tool.

Authentication is required to use this method.

Parameters

Name	Type	Description

ppt	String	a reference (ID) to the Project Planning Tool
urlUrl	String	the URL to the Project Planning Tool
pptUsername	String	the username for the user for the Project Planning Tool
pptPassword	String	the password for the user for the Project Planning Tool

Responses

Status	Description
400	If the login information could not be stored.
200	If the login information were stored. They are also returned in Json form.

Example 1

```
curl --request POST --data "ppt=9999&urlUrl=no url&pptUsername=name&
pptPassword=1234" http://localhost:9000/rest/api/1/user/pptAccount
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"pptUrl":["This field is required"],"ppt":["Invalid value"]}
```

Example 2

```
curl --request POST --data "ppt=1000000000000000005&
urlUrl=http.jira.example.com&pptUsername=admin&pptPassword=12345678"
http://localhost:9000/rest/api/1/user/pptAccount
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"pptUrl":["This field is required"]}
```

GET /rest/api/1/user/pptAccount

Returns all login information (but the password) for the currently logged in user for Project Planning Tools.

Authentication is required to use this method.

Responses

Status	Description

200	It's always a list returned containing all (but if there is none also none) entities.

Example 1

```
curl http://localhost:9000/rest/api/1/user/pptAccount
```

The previous command **did** return with status code **200 (OK)** and yielded the following:


```
{"items":[{"id":1000000000000000003,"ppt":{"id":3348,"name":"Example Project
Planning Tool"},"pptUrl":"https://ppt.example.com","user":
{"id":3295,"name":"user0"},"pptUsername":"tbucher"},
{"id":1000000000000000007,"ppt":{"id":3350,"name":"Example Project Planning
Tool"},"pptUrl":"https://ppt.example.com","user":3295,"pptUsername":"tbucher"}]}
```


POST /rest/api/1/user/pptAccount/<id>

Updates login information for a Project Planning Tool on the server.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the login information to update
ppt	String	a reference (ID) to the Project Planning Tool
pptUrl	String	the URL to the Project Planning Tool
pptUsername	String	the username for the user for the Project Planning Tool
pptPassword	String	the password for the user for the Project Planning Tool (optional)

Responses

Status	Description
404	If no entity with the given ID exists.
400	If the request parameter contain errors.
200	The new created entity is returned

Example 1

```
curl --request POST --data "ppt=1&pptUrl=no url&pptUsername=name&
pptPassword=1234" http://localhost:9000/rest/api/1/user/pptAccount/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Projectplanningtool Account with id 9999."
```

Example 2

```
curl --request POST --data "ppt=9999&pptUrl=no url&pptUsername=ozander&
pptPassword=pMuE2ekiDa" http://localhost:9000/rest/api/1/user/pptAccount
/1000000000000000003
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"ppt":["Invalid value"]}
```

Example 3

```
curl --request POST --data "ppt=1&pptUrl=https://ppt.example.com&
pptUsername=tbucher&pptPassword=7YqupNxN9v" http://localhost:9000/rest/api
/1/user/pptAccount/1000000000000000003
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"ppt":["Invalid value"]}
```

GET /rest/api/1/user/pptAccount/<id>

Returns one login information (but the password) for the currently logged in user for Project Planning Tools.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to get

Responses

Status	Description
404	If no entity with the given ID exists.

200	If it's found, it's returned.

Example 1

```
curl http://localhost:9000/rest/api/1/user/pptAccount/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Projectplanningtool Account with id 9999."
```

Example 2

```
curl http://localhost:9000/rest/api/1/user/pptAccount/1000000000000000003
```

The previous command **did** return with status code **200 (OK)** and yielded the following:


```
{"id":1000000000000000003,"ppt":{"id":3348,"name":"Example Project Planning
Tool"},"pptUrl":"https://ppt.example.com","user":
{"id":3295,"name":"user0"},"pptUsername":"tbucher"}
```


DELETE /rest/api/1/user/pptAccount/<id>

Deletes login information for a Project Planning Tool on the server.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to delete

Responses

Status	Description
404	If the login information to delete could not be found.
204	If the login information were deleted.

Example 1

```
curl --request DELETE http://localhost:9000/rest/api/1/user/pptAccount/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the

following:

```
"Could not find Projectplanningtool Account with id 9999."
```

Example 2

```
curl --request DELETE http://localhost:9000/rest/api/1/user/pptAccount
/1000000000000000007
```

The previous command **did** return with status code **204 (No Content)** and yielded the following:

```
```

3.9 Project

GET /rest/api/1/project

Returns all Projects.

Authentication is required to use this method.

Responses

Status	Description
200	It's always a list returned containing all (but if there is none also none) entities.

Example 1

```
curl http://localhost:9000/rest/api/1/project
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"items":[{"id":3310,"name":"Example project"},{"id":3312,"name":"Example
project"},{"id":3315,"name":"Example project"},{"id":3317,"name":"Example
project"},{"id":3322,"name":"Example Project"},{"id":3325,"name":"Example
Project"},{"id":3328,"name":"Example Project"},{"id":3331,"name":"Example
Project"},{"id":1000000000000000042,"name":"The Example Project"},
{"id":1000000000000000043,"name":"The Example Project"},
{"id":1000000000000000059,"name":"The Example Project"}]}
```

GET /rest/api/1/project/<id>

Returns one Project.

Authentication is required to use this method.

Parameters

Name	Type	Description
id	Long	The id of the entity to get

Responses

Status	Description
404	If no entity with the given ID exists.
200	If it's found, it's returned.

Example 1

```
curl http://localhost:9000/rest/api/1/project/9999
```

The previous command **did** return with status code **404 (Not Found)** and yielded the following:

```
"Could not find Project with id 9999."
```

Example 2

```
curl http://localhost:9000/rest/api/1/project/1000000000000000059
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":1000000000000000059,"name":"The Example Project"}
```

3.10 User

POST /rest/api/1/user/changePassword

This changes the password of an EEPPI-user.

Authentication is required to use this method.

Parameters

Name	Type	Description
oldPassword	String	the current password for the user
newPassword	String	the new password for the user

newPasswordRepeat	String	the new password for the user (repetition, to guarantee the user didn't make a typo)

Responses

Status	Description
400	If the password could not be changed.
200	If the password was changed.

Example 1

```
curl --request POST --data "oldPassword=demo&newPassword=1234&
newPasswordRepeat=1234" http://localhost:9000/rest/api/1/user/changePassword
```

The previous command **would probably** return with status code **200** and yield the following:

```
```

Example 2

```
curl --request POST --data "oldPassword=demo&newPassword=1234&
newPasswordRepeat=another password" http://localhost:9000/rest/api/1/user
/changePassword
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"":["The two passwords do not match"]}
```

POST /rest/api/1/user/login

Checks the login information for the user and if the login is successful a cookie is set.

This method is public and can be used without authentication.

Parameters

Name	Type	Description
name	String	username
password	String	the password for the user

Responses

Status	Description
400	If the user could not be logged in.
200	If the user could be logged in, and a cookie is set.

Example 1

```
curl --request POST --data "name=demo&password=demo" http://localhost:9000
/rest/api/1/user/login
```

The previous command **would probably** return with status code **200** and yield the following:

```
{"id":1,"name":"demo"}
```

Example 2

```
curl --request POST --data "name=demo&password=invalid password"
http://localhost:9000/rest/api/1/user/login
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
Username or Password wrong
```

Example 3

```
curl --request POST --data "name=demo&password=" http://localhost:9000/rest/api
/1/user/login
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"password":["This field is required"]}
```

GET /rest/api/1/user/loginStatus

Returns the login status for the currently logged in user and a Json representation of it.

This method is public and can be used without authentication.

Responses

Status	Description
200	Is always returned, and a Json containing either nothing (if no user is logged in) or the user.

Example 1

```
curl http://localhost:9000/rest/api/1/user/loginStatus
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":3295,"name":"user0"}
```

Example 2

```
curl http://localhost:9000/rest/api/1/user/loginStatus
```

The previous command **would probably** return with status code **200** and yield the following:

```
{

    "id": 1,
    "name": "demo"

}
```

POST /rest/api/1/user/logout

Does log out the currently logged in user by removing the cookie.

This method is public and can be used without authentication.

Responses

Status	Description
204	Is always returned, and the login cookie is being removed.

Example 1

```
curl --request POST http://localhost:9000/rest/api/1/user/logout
```

The previous command **did** return with status code **204 (No Content)** and yielded the following:

```

```

POST /rest/api/1/user/register

This creates a new EEPPI-user.

This method is public and can be used without authentication.

Parameters

Name	Type	Description
name	String	username
password	String	the new password for the user
passwordRepeat	String	the new password for the user (repetition, to guarantee the user didn't make a typo)

Responses

Status	Description
400	If the user could not be created.
200	If the user could be created, it is returned.

Example 1

```
curl --request POST --data "name=New Username 1&password=1234&
passwordRepeat=1234" http://localhost:9000/rest/api/1/user/register
```

The previous command **did** return with status code **200 (OK)** and yielded the following:

```
{"id":3359,"name":"New Username 1"}
```

Example 2

```
curl --request POST --data "name=New Username 2&password=1234&
passwordRepeat=another password" http://localhost:9000/rest/api/1/user/register
```

The previous command **did** return with status code **400 (Bad Request)** and yielded the following:

```
{"":["The two passwords do not match"]}
```