TutoringTrain Webservice

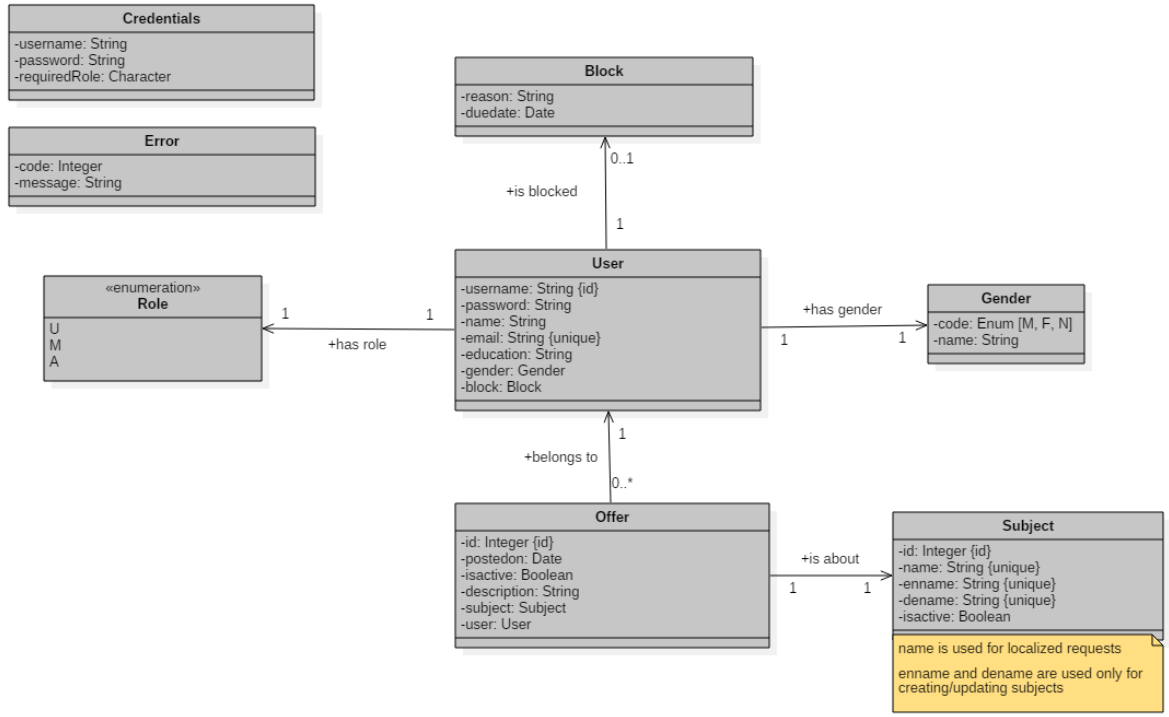
V1.5

REST-DOKU

TABLE OF CONTENTS

GENERAL

Data Model:



Only JSON format is supported.

All JSON attributes marked in **THIS COLOR** are optional.

Dates format is: "**yyyy-MM-dd'T'hh:mm:ssZ**"
Example: **2017-09-03T09:45:12+0200**

METHODS

[1] SECURITY & AUTHENTICATION

Authentication is managed via session tokens. This is a randomly generated 32-character string which is used to identify the logged in user.

1. Obtaining a session token

Note: A new session token is generated every time this method is called with valid credentials.

A token expires 24 hours after its creation.

Request

Method	URL	Role
POST	url/authentication	None

MediaType	Sample Request Body
application/json	{ "username": "testuser", "password": "098f6bcd4621d373cade4e832627b4f6", "requiredRole": "A" }

Response

Status	Response
200	654d4e57c9a868dd87759268b225ae44
401	Password and/or username are invalid.
403	User cannot authenticate himself in current context. (eg. normal user tries to log in with admin tool → set requiredRole attribute for this to work)
450	User is currently blocked. (Reason is given in response body)

2. Using the session token

Once a session token has been acquired it can be used to call all other methods. An exception is registering a User as this method does not require any authentication.

For every request the token has to be placed in the “Authorization”-Header in the following format:

“Bearer <token>”

Example:

Key	Value
Authorization	Bearer 4d8068152f82f0ae9ed117365db71cec

A method called without a session token returns **401** HTTP status.

If **401** is returned when a session token was provided it can be assumed that the token has expired or is invalid. In this case you can try to acquire a new token.

If **403** is returned the user associated with the session key has insufficient privileges to perform that method.

If **450** is returned the user associated with the session key has been blocked.

If **451** is returned, the provided JSON-text is invalid. Further details might be given in the returned Error.

If **422** is returned, a provided value violated a constraint. Further details might be given in the returned Error.

If **500** is returned an unknown error happened. Further details might be given in the returned Error.

These response codes apply to all methods described below but are not mentioned there. Only method-specific response codes are listed.

3. Validating the session token

A session token can be validated using the method described below. This is useful since clients can use this feature to check whether a previously obtained, cached token is still valid.
(Set token in Authorization header for this check)

Request

Method	URL	Role
POST	url/authentication/check	User

MediaType	Sample Request Body

application/json	{
	"requiredRole": "A"
	}

Response

Status	Response
200	The token is still valid
401	The token is not valid (see returned Error for further information)
403	The user belonging to the session token does not have the required role (set requiredRole in request body for this to work)

4. User roles

This System currently supports three roles:

User (**U**), Moderator (**M**) and Admin (**A**)

These roles extend the privileges of their inferior roles, this means a Moderator can perform all actions a User can, and an Admin can perform all actions a Moderator can.

[2] ERROR HANDLING AND LOCALIZATION

In the event of an error / exception, an Error object of the following structure is returned:

```
{
   "code": <error code>,
   "message": <error message>
}
```

The following error codes exist:

Code	Descrption
1	Unknown error – further details might be given in message
2	Admin only – happens when user without Admin role tries to authenticate when “requiredRole” is set to ‘A’
3	Token invalid – the authentication token does not exist
4	Token expired – the authentication token has expired
5	Insufficient privileges – the user does not have to required role to perform this action
6	Username is null – No username was given
7	Name is null – No name was given (e.g. Create Subject)
8	User is null – No user was given
9	Offer is null – No offer was given
10	Gender is null – No gender was given
11	Subject is null – No subject was given
12	User not found – The user was not found
13	Subject not found – The subject was not found
14	Offer not found – The offer was not found
15	Gender not found – The gender was not found
16	Invalid email – The given email is of an invalid format
17	Query parameters missing – “start” and “pageSize” query parameters were not provided
18	Subject name conflict – The subject name already exists
19	Username conflict – The username is not available (already in use)
20	Email conflict – The email is not available (already in use)
21	Cannot block own user
22	Cannot unblock own user
23	Cannot block other admins
24	Authentication failed – Invalid credentials were provided
25 26 27 28	Blocked – The user is blocked from using the application
29	Invalid JSON – The provided JSON-Request is malformed
30	Constraint violation – As provided parameter violates a constraint (see message for more specific information)
31	Unsupported Media Type – The provided media type is not PNG or JPG/JPEG.
32	Subject not active – The provided subject is not active and hence cannot be used.

While the “code” identifies the error, the “message” attribute provides a more detailed description of the error.

To receive a **localized error message**, set the “Accept-Language”-HTTP Header to the desired language. Currently only English (**en**) and German (**de**) are supported.
If no “Accept-Language” – Header is provided, English is used by default.

[3] User Resource

1. Register User

Request

Method	URL	Role
POST	url/user/register	None

MediaType	Sample Request Body

application/json	{
	"name": "Kevin Paul",
	"username": "kevin",
	"email": "kevin.paul@gmx.at",
	"password": "098f6bcd4621d373cade4e832627b4f6",
	"education": "HTL",
	"gender": "F"
	}

Response

Status	Response

200	{
	"username": "kevin",
	"role": "U",
	"email": "kevin.paul@gmx.at",
	"name": "Kevin Paul",
	"education": "HTL",
	"gender": "F"
	}

409	Username already exists.

2. Get all Users

Request

Method	URL	Role
GET	url/user/all	Admin

Parameters

Type	Name	Description
Query	start	Start index of users to get
Query	pageSize	Maximum number of offers

Response

Status	Response

200	[
	{
	"username": "kevin",
	"role": "U",
	"email": "kevin.paul@gmx.at",
	"name": "Kevin Paul",
	"education": "HTL",
	"gender": "F"
	},
	{
	"username": "test",
	"role": "U",
	"email": "santnere@edu.htl-villach.at",
	"name": "wacco",
	"education": "HTL",
	"gender": "M",
	"block": {
	"reason": "test"
	}
	},
	{
	"username": "wacco",
	"role": "A",
	"email": "santnere@edu.htl-villach.at",
	"name": "wacco",
	"education": "HTL",
	"gender": "N"
	}
	]

3. Get own User

Request

Method	URL	Role
GET	url/user	User

Response

Status	Response

200	{
	"username": "wacco",
	"role": "A",
	"email": "santnere@edu.htl-villach.at",
	"name": "wacco",
	"education": "HTL",
	"gender": "M"
	}

4. Update any User

Request

Method	URL	Role
PUT	url/user/update	Admin

MediaType	Sample Request Body

application/json	{
	"username": "kevin",
	"name": "Paul Kevin",
	"email": "coolerkev@gmx.at",
	"password": "9d5e3ecdeb4cdb7acfd63075ae046672",
	"education": "HTL",
	"gender": "N"
	}

Response

Status	Response
200	
453	User not found

5. Update own User

Request

Method	URL	Role
PUT	url/user/update/own	User

MediaType	Sample Request Body

application/json	{
	"name": "Elias Santner",
	"email": "santnere@edu.htl-villach.at",
	"password": "098f6bcd4621d373cade4e832627b4f6",
	"education": "HTL",
	"gender": "N"
	}

Response

Status	Response
200	

6. Block User

Request

Method	URL	Role
POST	url/user/block	Admin

MediaType	Sample Request Body

application/json	{
	"username": "test",
	"reason": "test",
	"duedate": "2017-09-03T09:45:12+0200"
	}

Response

Status	Response
200	
456	Error blocking user. (Further details may be provided via body)

7. Unblock User

Request

Method	URL	Role
GET	url/user/unblock/{username}	Admin

Parameters

Type	Name	Description
Path	Username	Username of user to unblock

Response

Status	Response
200	

8. Get all Genders

Request

Method	URL	Role
GET	url/user/gender	User

Response

Status	Response

200	[
	{
	"code": "F",
	"name": "Female"
	},
	{
	"code": "M",
	"name": "Male"
	},
	{
	"code": "N"
	"name": "Unspecified"
	}
	]

9. Get User Count

Request

Method	URL	Role
GET	url/user/count	User

Response

Status	Response
200	10

10. Get Avatar

Request

Method	URL	Role
GET	url/user/avatar/{username}	User

Response

Status	Response
200	Avatar is returned with format "image/jpg"
204	User has no avatar

11. Get Own Avatar

Request

Method	URL	Role
GET	url/user/avatar	User

Response

Status	Response
200	Avatar is returned with format "image/jpg"
204	User has no avatar

12. Set Avatar

Request

Method	URL	Role
POST	url/user/avatar	User

MediaType	Sample Request Body
multipart form data	See example Java program (6.1)

Response

Status	Response
200	Avatar is returned with format "image/jpg"
204	User has no avatar

[4] Offer Resource

1. Create Offer

Request

Method	URL	Role
POST	url/offer	User

MediaType	Sample Request Body

application/json	{
	"description": "This is just a test offer",
	"subject": {
	"id": 3
	}
	}

Response

Status	Response

200	{
	"id": 1,
	"postedon": "2017-10-04T07:18:39+0200",
	"isactive": true,
	"description": "This is just a test offer",
	"subject": {
	"id": 1,
	"name": "Englisch",
	"isactive": true
	},
	"user": {
	"username": "wacco"
	}
	}

452	Subject not found
457	Subject is not active

2. Update Offer

Request

Method	URL	Role
PUT	url/offer	User

MediaType	Sample Request Body

application/json	{
	"id": 1,
	"isactive": false,
	"description": "This is just a test offer",
	"subject": {
	"id": 1
	}
	}

Response

Status	Response
200	
452	Subject not found
454	Offer not found
457	Subject is not active

3. Get newest Offers

Request

Method	URL	Role
GET	url/offer/new	User

Parameters

Type	Name	Description
Query	start	Start index of offers to get
Query	pageSize	Maximum number of offers

Response

Status	Response

200	[
	{
	"id": 2,
	"postedon": "2017-10-04T07:27:58+0200",
	"isactive": true,
	"description": "Test offer 2",
	"subject": {
	"id": 1,
	"name": "Englisch",
	"isactive": true
	},
	"user": {
	"username": "wacco"
	}
	},
	{
	"id": 1,
	"postedon": "2017-10-04T07:18:39+0200",
	"isactive": false,
	"description": "This is just a test offer",
	"subject": {
	"id": 1,
	"name": "Englisch",
	"isactive": true
	},
	"user": {
	"username": "wacco"
	}
	}
	]

455	Query parameters were not given

4. Get newest Offers of User

Request

Method	URL	Role
GET	url/offer/new/{username}	User

Parameters

Type	Name	Description
Path	Username	Username of user to get newest offers of
Query	start	Start index of offers to get
Query	pageSize	Maximum number of offers

Response

Status	Response

200	[
	{
	"id": 2,
	"postedon": "2017-10-04T07:27:58+0200",
	"isactive": true,
	"description": "Test offer 2",
	"subject": {
	"id": 1,
	"name": "Englisch",
	"isactive": true
	},
	"user": {
	"username": "wacco"
	}
	},
	{
	"id": 1,
	"postedon": "2017-10-04T07:18:39+0200",
	"isactive": false,
	"description": "This is just a test offer",
	"subject": {
	"id": 1,
	"name": "Englisch",
	"isactive": true
	},
	"user": {
	"username": "wacco"
	}
	}
	]

453	User not found
455	Query parameters were not given

5. *Get User Count*

Request

Method	URL	Role
GET	url/offer/count	User

Response

Status	Response
200	10

[5] SUBJECT RESOURCE

1. Create Subject

Request

Method	URL	Role
POST	url/subject	User

MediaType	Sample Request Body

application/json	{
	"dename": "Programmieren",
	"enname": "Programming"
	}

Response

Status	Response

200	{
	"id": 2,
	"name": "Programmieren"
	}

409	Subject name already exists

Note: If an Admin creates a Subject, it will be active. If a User or Moderator creates a Subject it will be inactive until activated by an Admin via the “Update Subject” method.

2. Update Subject

Request

Method	URL	Role
PUT	url/subject	Admin

MediaType	Sample Request Body

application/json	{
	"id": 4,
	"dename": "Mathe",
	"enname": "Maths",
	"isactive": true
	}

Response

Status	Response
200	
452	Subject not found
409	Subject name already exists

3. Delete Subject

Request

Method	URL	Role
DELETE	url/subject/{id}	Admin

Parameters

Type	Name	Description
Path	Id	Id of subject to delete

Response

Status	Response
200	
452	Subject not found

4. Get all active Subjects

Request

Method	URL	Role
GET	url/subject	User

Response

Status	Response

200	[
	{
	"id": 2,
	"name": "Deutsch",
	"isactive": true
	},
	{
	"id": 1,
	"name": "Englisch",
	"isactive": true
	},
	{
	"id": 3,
	"name": "Mathe",
	"isactive": true
	}
	]

5. *Get all inactive Subjects*

Request

Method	URL	Role
GET	url/subject/inactive	Admin

Response

Status	Response

200	[
	{
	"id": 2,
	"name": "Deutsch",
	"isactive": false
	},
	{
	"id": 1,
	"name": "Englisch",
	"isactive": false
	},
	{
	"id": 3,
	"name": "Mathe",
	"isactive": false
	}
	]

6. *Get Subject Count*

Request

Method	URL	Role
GET	url/subject/count	User

Response

Status	Response
200	3

[6] EXAMPLES

1. Java code snippet to upload Avatar:

```
File file2upload = new File("D:\\Elias\\Desktop\\1612131143-Mini360-
Sokrates\\20241720090151.jpg");

    RequestBody requestBody = new MultipartBody.Builder()
       .setType(MultipartBody.FORM)
       .addPart(
          Headers.of("Content-Disposition", "form-data; name=\"name\""),
          RequestBody.create(null, file2upload.getName()))
       .addPart(
          Headers.of("Content-Disposition", "form-data; name=\"file\""),
          RequestBody.create(MEDIA_TYPE_PNG, file2upload))
       .build();

    Request request = new Request.Builder()
       .header("Authorization", "Bearer f14a84d2958f26e400121c36f11ff1d")
       .url("http://localhost:51246/TutoringTrainWebservice/services/user/avatar")
       .post(requestBody)
       .build();

    Response response = client.newCall(request).execute();
```