



# TUTORINGTRAIN WEBSERVICE V1.9.7

REST-DOKU

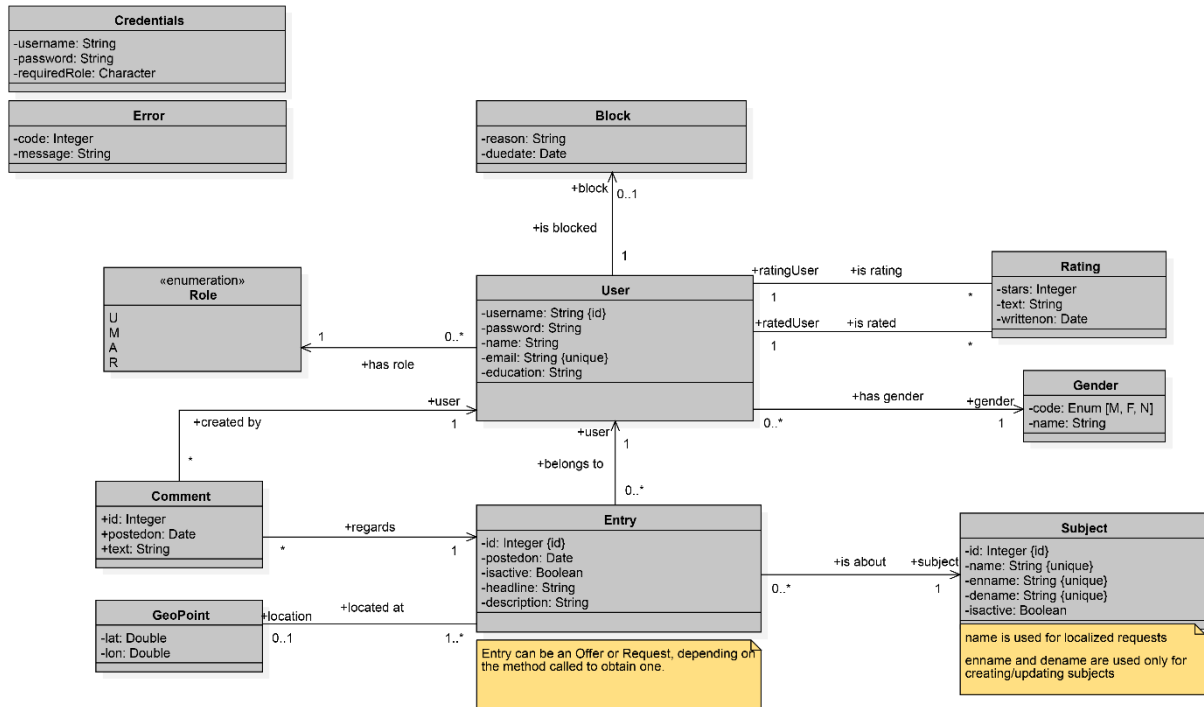
# TABLE OF CONTENTS

General .....	3
Methods .....	4
[1] Security & Authentication .....	4
1. Obtaining a session token .....	4
2. Using the session token.....	5
3. Validating the session token .....	6
4. User roles.....	6
[2] Error Handling and Localization .....	7
[3] User Resource.....	9
1. Register User .....	9
2. Activate account .....	10
3. Get all Users .....	11
4. Get own User.....	12
5. Get single User .....	13
6. Update any User .....	14
7. Update own User.....	15
8. Set Role.....	16
9. Block User .....	17
10. Unblock User .....	18
11. Get all Genders .....	19
12. Get User Count .....	20
13. Get Avatar.....	21
14. Get Own Avatar .....	22
15. Set Avatar .....	23
16. Reset Own Avatar .....	24
17. Reset Avatar .....	25
18. Reset Properties of own User.....	26
19. Reset Properties of any User .....	27
20. Reset Password.....	28
21. Delete User .....	29
22. Rate User .....	30
23. Get Rating of User .....	31
24. Get Rated Users.....	32

25.	Searching .....	33
26.	XMPP – Add User.....	35
27.	XMPP – Get Contact Type.....	36
28.	Get Newsletter Template .....	37
29.	Send Newsletter .....	38
[4]	Entry Resources .....	40
1.	Create Entry.....	40
2.	Update Entry .....	41
3.	Delete Entry.....	42
4.	Reset Properties of Entry .....	43
5.	Get newest Entries .....	44
6.	Get newest Entries of User.....	45
7.	Get Entry Count.....	46
8.	Get Comments of Entries .....	47
9.	Create Comment .....	48
10.	Update Comment .....	49
11.	Delete Comment .....	50
12.	Searching .....	51
13.	Simple Search .....	54
[5]	Subject Resource .....	55
1.	Create Subject .....	55
2.	Update Subject .....	56
3.	Delete Subject .....	57
4.	Get all Subjects .....	58
5.	Get all active Subjects .....	59
6.	Get all inactive Subjects .....	60
7.	Get Subject Count.....	61
[6]	Examples.....	62
1.	Java code snippet to upload Avatar: .....	62

# GENERAL

## Data Model:



**Only JSON format is supported.**

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

Date format is: **"yyyy-MM-dd'T'HH:mm:ssZ"**  
Example: **2017-09-03T19:45:12+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	<pre>{   "username": "testuser",   "password": "098f6bcd4621d373cade4e832627b4f6",   "requiredRole": "A" }</pre>

#### 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	<pre>{   "requiredRole": "A" }</pre>

#### 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 four roles:

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

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.

The Root can perform all actions, including degrading Admins.

Both the role Admin and Root are considered to be "Admins".

## [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	Description
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 of user not found – The offer with given id doesn’t exist or wasn’t created by given user
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	Blocked – The user is blocked from using the application
26	
27	
28	
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.
33	Start < 0 – The start parameter provided in the query string is < 0
34	Page Size <= 0 – The pageSize parameter provided in the query string is <= 0
35	Offer not found – The offer with given id doesn’t exist
36	Subject is used – The subject is used and hence cannot be deleted. (Use Update to deactivate)
37	Illegal Role – A illegal role was specified (not U, M or A)
38	Degrade Admin – Cannot degrade another Admin
39	Degrade Own User – Cannot set the role of the own user
40	Request not found – The request with given id doesn’t exist
41	Request of user not found – The request with given id doesn’t exist or wasn’t created by given user
42	XMPP Error – An error occurred during some operation (create user, ...) on the Chat Server.
43	Cannot set Root role – This role can only be set directly in the Database.



44	Comment not found – The comment with the given ID does not exist
45	Comment of user not found – The comment with the given ID does not exist or was not created by calling user
46	Invalid Value – An invalid value was provided (e.g. alphanumeric characters for number) NOTE: This does <b>not</b> apply for <b>JSON</b> -Strings, only for other parameters (e.g. ID in as pathparam)
47	Entry not found – The Entry with the given ID was not found (no difference made between Offer and Request)
48	Cannot delete admins – The given user is an admin and can therefore not be deleted (Root (R) can bypass this)
49	Cannot delete user – The given user has created entries, comments or given / received ratings and can therefore not be deleted.
50	Cannot rate own user – You cannot rate your own user
51	Search string null – The search string must not be null (Simple Search)
52	User not activated – The user (account) has not yet been activated

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

#### Response

Status	Response
200	<pre>{   "username": "kevin",   "role": "U",   "email": "kevin.paul@gmx.at",   "name": "Kevin Paul",   "education": "HTL",   "gender": "F",   "averagerating": "4.7" }</pre>
409	Username already exists.

**Note:** If no password is given, the user is assigned a random 8-character long alphanumeric password. This password is send in the welcome email to the given email address.

The user is required to activate his/her account by clicking on the link in this email before they may use any other service.

## 2. Activate account

### Request

Method	URL	Role
GET	url/user/activate	None

### Parameters

Type	Name	Description
Query	key	Activation key sent to user by email

### Response

Status	Response
200	HTML page with message
400	Invalid Query String parameters

**Note:** This method is meant to only be called by clicking the activation link in the welcome email.

### 3. Get all Users

#### Request

Method	URL	Role
GET	url/user/all	User

#### Parameters

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

#### Response

Status	Response
200	<pre>[   {     "username": "kevin",     "role": "U",     "email": "kevin.paul@gmx.at",     "name": "Kevin Paul",     "education": "HTL",     "gender": "F",     "averagerating": "Not yet rated"   },   {     "username": "test",     "role": "U",     "email": "santnere@edu.htl-villach.at",     "name": "wacco",     "education": "HTL",     "gender": "M",     "averagerating": "3.9"     "block": {       "reason": "test"     }   },   {     "username": "wacco",     "role": "A",     "email": "santnere@edu.htl-villach.at",     "name": "wacco",     "education": "HTL",     "gender": "N",     "averagerating": "Not yet rated"   } ]</pre>
455	Invalid Query String parameters

**Note:** Email is only returned if user is an Admin.

#### 4. Get own User

##### Request

Method	URL	Role
GET	url/user	User

##### Response

Status	Response
200	<pre>{   "username": "wacco",   "role": "A",   "email": "santnere@edu.htl-villach.at",   "name": "wacco",   "education": "HTL",   "gender": "M",   "averagerating": "Not yet rated" }</pre>

## 5. Get single User

### Request

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

### Parameters

Type	Name	Description
Path	username	Username of user to get

### Response

Status	Response
200	{ "username": "wacco", "role": "A", "email": "santnere@edu.htl-villach.at", "name": "wacco", "education": "HTL", "gender": "M", "averagerating": "2.5" }
453	User not found

**Note:** Email is only returned if user is an Admin.

## 6. Update any User

### Request

Method	URL	Role
PUT	url/user/update	Admin

MediaType	Sample Request Body
application/json	<pre>{   "username": "kevin",   "name": "Paul Kevin",   "email": "coolerkev@gmx.at",   "password": "9d5e3ecdeb4cdb7acfd63075ae046672",   "education": "HTL",   "gender": "N" }</pre>

### Response

Status	Response
200	
453	User not found

## 7. Update own User

### Request

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

MediaType	Sample Request Body
application/json	<pre>{   "name": "Elias Santner",   "email": "santnere@edu.htl-villach.at",   "password": "098f6bcd4621d373cade4e832627b4f6",   "education": "HTL",   "gender": "N" }</pre>

### Response

Status	Response
200	



## 8. Set Role

### Request

Method	URL	Role
PUT	url/user/role/{username}	Admin

### Parameters

Type	Name	Description
Path	username	Username of user to set role of

MediaType	Sample Request Body
application/json	{ "role": "U" }

### Response

Status	Response
200	
401	Unauthorized, cannot degrade other admins or own user
453	User not found

**Note:** The Role Root (**R**) cannot be set via this method.

## 9. Block User

### Request

Method	URL	Role
POST	url/user/block	Admin

MediaType	Sample Request Body
application/json	<pre>{   "username": "test",   "reason": "test",   "duedate": "2017-09-03T09:45:12+0200" }</pre>

### Response

Status	Response
200	
456	Error blocking user. (Further details may be provided via body)
453	User not found

## 10. 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	
453	User not found

## 11. *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" } ]

## 12. *Get User Count*

### Request

Method	URL	Role
GET	url/user/count	User

### Response

Status	Response
200	10

### 13. *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
453	User not found

## 14. *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

## 15. 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



## 16. *Reset Own Avatar*

### Request

Method	URL	Role
DELETE	url/user/avatar	User

### Response

Status	Response
200	

## 17. *Reset Avatar*

### Request

Method	URL	Role
DELETE	url/user/avatar/{username}	Admin

### Response

Status	Response
200	
453	User not found

## 18. *Reset Properties of own User*

### Request

Method	URL	Role
POST	url/user/reset	User

MediaType	Sample Request Body
application/json	["education", "name"]

### Possible Properties

Education, Name

### Response

Status	Response
200	
451	Invalid property provided

## 19. *Reset Properties of any User*

### Request

Method	URL	Role
POST	url/user/reset/{username}	Admin

MediaType	Sample Request Body
application/json	["education", "name"]

### Possible Properties

Education, Name

### Response

Status	Response
200	
451	Invalid property provided
453	User not found

## 20. *Reset Password*

### Request

Method	URL	Role
PUT	url/user/resetPw/{username}	Admin

### Response

Status	Response
200	
451	Invalid property provided
453	User not found

**Note:** This method sets the selected user's password to a random 8-character-long string. The new password is sent to the user's email address.

## 21. Delete User

### Request

Method	URL	Role
DELETE	url/user/{username}	Admin

### Parameters

Type	Name	Description
Query	force	True ... Delete user even if he has entries, comments or ratings

### Response

Status	Response
200	
460	User has entries, comments or ratings (given and/or received)
453	User not found

**Note:** This method deletes the given user completely. Without the parameter “force=true” this only works if said user has created no entries, no comments and hasn’t given or received any ratings.

## 22. Rate User

### Request

Method	URL	Role
POST	url/user/rating/{username}	User

MediaType	Sample Request Body
application/json	<pre>{   "stars": 2,   "text": "Your service is too expensive, IMO" }</pre>

### Response

Status	Response
200	<pre>{   "text": "Your service is too expensive, IMO",   "stars": 2,   "writtenon": "2018-02-22T04:27:13+0100",   "ratedUser": {     "username": "user4"   },   "ratingUser": {     "username": "admin"   } }</pre>
460	Cannot rate own user
453	User not found

**Note:** “stars” must be an integer value between 1 and 5.

When a user rates the same user again his previous rating is overwritten.

### 23. *Get Rating of User*

#### Request

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

#### Response

Status	Response
200	[ { "text": "TEST RATING", "stars": 2, "writtenon": "2018-02-23T06:27:13+0100", "ratedUser": { "username": "user4" }, "ratingUser": { "username": "admin" } }, { "text": "Very nice lessons", "stars": 5, "writtenon": "2017-02-23T06:27:13+0100", "ratedUser": { "username": "user4" }, "ratingUser": { "username": "gunther" } } ]
453	User not found



## 24. *Get Rated Users*

### Request

Method	URL	Role
GET	url/user/rating/{username}/given	User

### Response

Status	Response
200	<pre>[   {     "stars": 4,     "writtenon": "2018-02-23T06:27:13+0100",     "ratedUser": {       "username": "NativeSpeaker69"     },     "ratingUser": {       "username": "user4"     }   },   {     "text": "OK maths lessons",     "stars": 3,     "writtenon": "2017-02-23T06:27:13+0100",     "ratedUser": {       "username": "MethChamp"     },     "ratingUser": {       "username": "user4"     }   } ]</pre>
453	User not found

## 25. Searching

### Request

Method	URL	Role
POST	url/user/search	Admin

### Parameters

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

MediaType	Sample Request Body
application/json	<pre>{   "criteria": [     {       "key": "USERNAME",       "value": "Adm",       "operation": "CONTAINS",       "ignoreCase": true     },     {       "key": "USERNAME",       "value": "in",       "not": true,       "operation": "ENDS_WITH",       "ignoreCase": true     },     {       "key": "GENDER",       "value": "U",       "not": true,       "operation": "EQ"     }   ],   "order": [     {       "key": "USERNAME",       "direction": "ASC"     }   ] }</pre>

### Response

Status	Response
200	A list of all found users is provide in the response body, similar to “Get All Users”
451	Invalid key provided, e.g. Key ‘XY’ or Operation ‘adfgsdg’
455	Invalid Query String parameters

All given criteria must be met for a user to be returned.

## Keys

Name	Type
USERNAME	STRING
NAME	STRING
EDUCATION	STRING
ROLE	CHAR
GENDER	CHAR
RATING	NUMBER

## Operations

Type	Operations
STRING	EQ CONTAINS STARTS_WITH ENDS_WITH
CHAR	EQ
NUMBER	EQ GT LT GTE LTE

## Order Directions

Direction
ASC
DESC

## 26. XMPP – Add User

### Request

Method	URL	Role
POST	url/xmpp/add/{username}	User

### Parameters

Type	Name	Description
Path	username	User to add

### Response

Status	Response
200	
453	User not found

## 27. XMPP – Get Contact Type

### Request

Method	URL	Role
POST	url/xmpp/myRoster/{username}	User

### Parameters

Type	Name	Description
Path	username	User to get contact type of

### Response

Status	Response
200	
453	User not found

Contact Types:

- NONE: User is not my contact
- REQUESTED: Contact request has already been sent to User
- ACCEPTED: User is my contact

## 28. *Get Newsletter Template*

### Request

Method	URL	Role
GET	url/user/newsletter	Admin

### Response

Status	Response
200	{HTML of Newsletter}

All attributes described in chapter 29. are included in this template with the following syntax:

**{attribute\_name}**

## 29. Send Newsletter

### Request

Method	URL	Role
POST	url/user/newsletter	Admin

MediaType	Sample Request Body
application/json	<pre>{   "subject": "Email Subject",   "title": "NEW FEATURES",   "text_top": "Line1&lt;br&gt;Line2",   "text_bottom": "xyz",   "url_img_top": "https://avatars1.githubusercontent.com/u/100",   "url_img_bottom": "https://avatars1.githubusercontent.com/u/101",   "targets": [     "U",     "M"   ] }</pre>

### Response

Status	Response
200	

This method sends a Newsletter to all users having the given “target-roles” (U, M, A, R).

If the attribute “targets” is absent, the Newsletter is sent to all users.

The attributes “title”, “text\_top” and “text\_bottom” are displayed in the Newsletter form.

In addition to these three attributes, there are placeholders which are replaced by the individual user’s values:

- @name: Replaced with the name of a user.
- @username: Replaced with the username of a user.

To separate lines, use the “<br>”-tag.

Furthermore, the two images on the Newsletter can be customized using URLs (“url\_img\_top” and “url\_img\_bottom”). If one of these attributes is left out, a default image is used.

The layout of the Newsletter is shown on the next page.

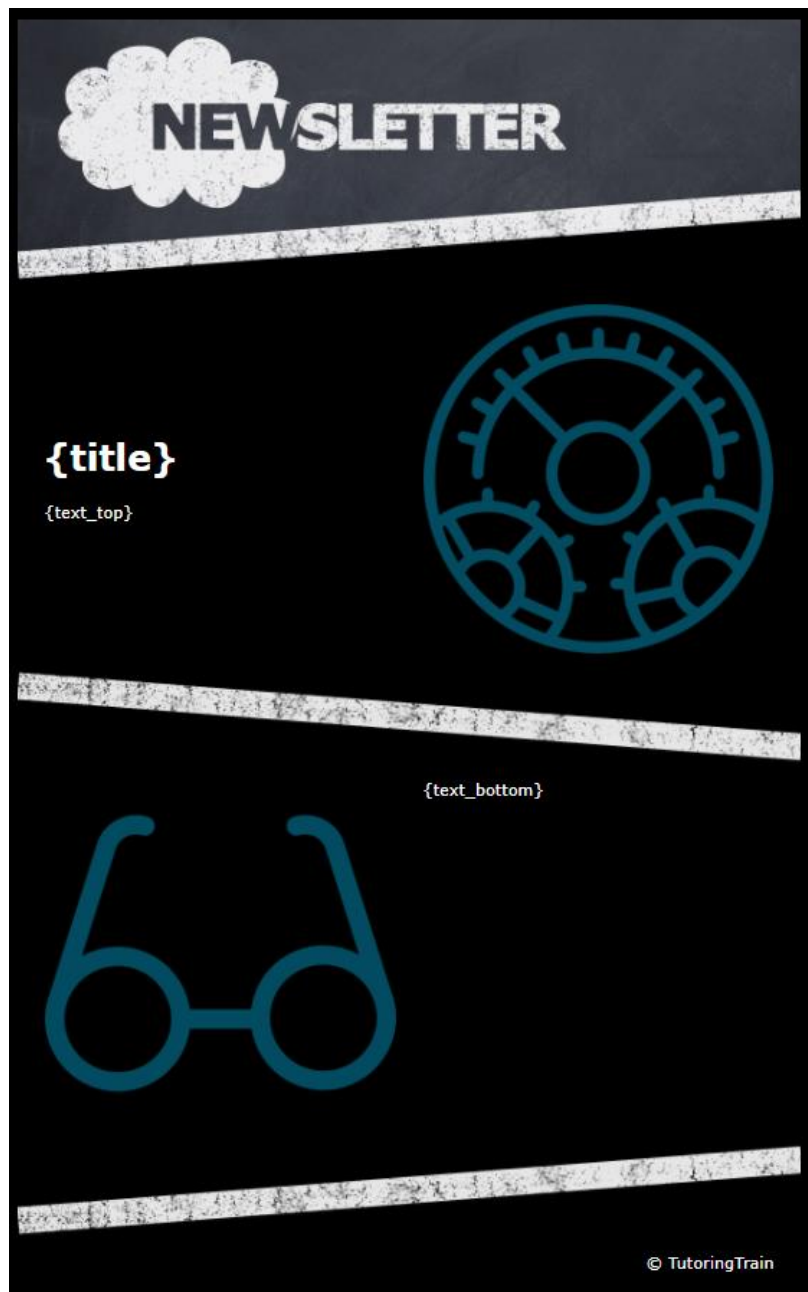


Figure 1: Newsletter layout



## [4] ENTRY RESOURCES

Offers and Requests are both entries, and since this service offers the same methods for both types of entries they are only described once.

To call a method for offers / requests, use the provided URL and replace the **<entryType>** placeholder with **offer** / **request**.

### 1. Create Entry

#### Request

Method	URL	Role
POST	url/<entryType>	User

MediaType	Sample Request Body
application/json	<pre>{   "description": "This is just a test offer",   "headline": "Test offer",   "subject": {     "id": 3   },   "location": {     "lon": -45.6545,     "lat": 5.7894   } }</pre>

#### Response

Status	Response
200	<pre>{   "id": 1,   "postedon": "2017-10-04T07:18:39+0200",   "isactive": true,   "description": "This is just a test offer",   "headline": "Test offer",   "subject": {     "id": 1,     "name": "Englisch",     "isactive": true   },   "user": {     "username": "wacco"   },   "location": {     "lon": 13.85062,     "lat": 46.60856,     "name": "Villach, 9500, Carinthia, Austria"   } }</pre>
452	Subject not found

<b>457</b>	Subject is not active
------------	-----------------------

## 2. Update Entry

### Request

Method	URL	Role
PUT	url/<entryType>	User

MediaType	Sample Request Body
application/json	<pre>{   "id": 1,   "isactive": false,   "description": "This is just a test offer",   "subject": {     "id": 1   },   "headline": "Test offer",   "location": {     "lon": -45.6545,     "lat": 5.7894   } }</pre>

### Response

Status	Response
<b>200</b>	
<b>452</b>	Subject not found
<b>454</b>	Entry not found
<b>457</b>	Subject is not active

**Note:** Only Admins can update entries of other users. If a Moderator or normal User tries to update an entry not created by himself, an error is returned.

### 3. Delete Entry

#### Request

Method	URL	Role
DELETE	url/<entryType>/<entryID>	User

#### Response

Status	Response
200	
404	Offer/Request not found

**Note:** Admins can delete any Entry, non-admin users can only delete entries created by themselves.

#### 4. Reset Properties of Entry

##### Request

Method	URL	Role
POST	url/<entryType>/reset/{id}	Admin

MediaType	Sample Request Body
application/json	["duedate"]

##### Possible Properties

Duedate, Location

##### Response

Status	Response
200	
451	Invalid property provided
454	Entry not found

**Note:** Only Admins can reset properties of entries other users created. If a Moderator or normal User tries this, an error is returned.

## 5. Get newest Entries

### Request

Method	URL	Role
GET	url/<entryType>/new	User

### Parameters

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

### Response

Status	Response
200	<pre>[   {     "id": 2,     "postedon": "2017-10-04T07:27:58+0200",     "isactive": true,     "description": "This is just a test offer",     "headline": "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",     "headline": "Test",     "subject": {       "id": 1,       "name": "Englisch",       "isactive": true     },     "user": {       "username": "wacco"     },     "location": {       "lon": 13.85062,       "lat": 46.60856,       "name": "Villach, 9500, Carinthia, Austria"     }   } ]</pre>
455	Query parameters were not given

## 6. Get newest Entries of User

### Request

Method	URL	Role
GET	url/<entryType>/new/{username}	User

### Parameters

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

### 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" }, "location": { "lon": 13.85062, "lat": 46.60856, "name": "Villach, 9500, Carinthia, Austria" } }, { "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

## 7. Get Entry Count

### Request

Method	URL	Role
GET	url/<entryType>/count	User

### Response

Status	Response
200	10

## 8. Get Comments of Entries

### Request

Method	URL	Role
GET	url/<entryType>/{entryID}/comments	User

### Parameters

Type	Name	Description
Path	entryID	ID of the Offer/Request to get the comments of

### Response

Status	Response
200	[ { "id": 2, "postedon": "2017-12-22T12:06:28+0100", "text": "test comment", "user": { "username": "admin" } }, { "id": 4, "postedon": "2017-12-22T14:16:28+0100", "text": "another comment", "user": { "username": "someuser" } } ]
404	Entry not found



## 9. Create Comment

### Request

Method	URL	Role
POST	url/<entryType>/{entryID}/comments	User

MediaType	Sample Request Body
application/json	<pre>{   "text": "Testcomment" }</pre>

### Response

Status	Response
200	<pre>{   "id": 21,   "postedon": "2018-01-12T10:16:37+0100",   "text": "Testcomment",   "user": {     "username": "admin"   } }</pre>
404	Offer/Request not found

## 10. Update Comment

### Request

Method	URL	Role
PUT	url/<entryType>/{entryID}/comments	User

MediaType	Sample Request Body
application/json	<pre>{   "id": 2,   "text": "TEST" }</pre>

### Response

Status	Response
200	
404	Offer/Request not found

**Note:** Admins can update any Comment, non-admin users can only update comments created by themselves.

## 11. *Delete Comment*

### Request

Method	URL	Role
DELETE	url/<entryType>/<entryID>/comments/{commentId}	User

### Response

Status	Response
200	
404	Offer/Request not found

**Note:** Admins can delete any Comment, non-admin users can only delete comments created by themselves.

## 12. Searching

### Request

Method	URL	Role
POST	url/<entryType>/search	User

### Parameters

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

MediaType	Sample Request Body
application/json	<pre>{   "criteria": [     {       "key": "ISACTIVE",       "value": true,       "not": false,       "operation": "EQ"     },     {       "key": "POSTEDON",       "value": "2017-10-20T12:00:00+0200",       "not": false,       "operation": "BEFORE_EQ"     },     {       "key": "DESCRIPTION",       "value": "you",       "not": false,       "operation": "CONTAINS"     },     {       "key": "LOCATION",       "value": {         "lon": -45.6545,         "lat": 5       },       "not": false,       "operation": "INSIDE",       "radius": 100     }   ],   "order": [     {       "key": "POSTEDON",       "direction": "DESC"     }   ] }</pre>

## Response

Status	Response
200	A list of all found entries is provide in the response body, similar to “Get Newest Entries”
451	Invalid property provided, e.g. Key ‘XY’ or Operation ‘adfgsdg’
455	Invalid Query String parameters

All given criteria must be met for an entry to be returned.

## Keys

Name	Type
ID	NUMBER
POSTEDON	DATE
ISACTIVE	BOOLEAN
DESCRIPTION	STRING
HEADLINE	STRING
SUBJECT	NUMBER
USERNAME	STRING
LOCATION	SPATIAL

## Operations

Type	Operations
STRING	EQ CONTAINS STARTS_WITH ENDS_WITH
DATE	EQ (compares by day, not by second) BEFORE (does NOT include date) AFTER (does NOT include date) BEFORE_EQ (does include date) AFTER_EQ (does include date)
BOOLEAN	EQ
NUMBER	EQ GT LT GTE LTE
SPATIAL (see <b>Sample Request Body</b> )	INSIDE (searches for entries inside the given radius) INSIDE_NULL (same as INSIDE, but entries with no location are returned as well)

## Order Directions

Direction
ASC
DESC

### 13. Simple Search

#### Request

Method	URL	Role
POST	url/<entryType>/simpleSearch	User

#### Parameters

Type	Name	Description
Query	start	Start index of entries to get
Query	pageSize	Maximum number of entries
Query	searchStr	Phrase to search for

#### Response

Status	Response
200	A list of all found entries is provide in the response body, similar to “Get Newest Entries”
451	Invalid property provided, e.g. Key ‘XY’ or Operation ‘adfgsdg’
455	Invalid Query String parameters

This method searches all Entries of the specified type. It returns all Entries containing the “searchStr” in **any** of the following properties:

- Headline
- Description
- Subject
  - Enname (English name)
  - Dename (German name)
- Location name

## [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	<pre>{   "id": 4,   "dename": "Mathe",   "enname": "Maths",   "isactive": true }</pre>

### Response

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

**Note:** Setting “isactive” to false disables the use of a subject in Create Offer / Update Offer. Offers already referencing this subject remain untouched.

### 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
458	Subject is in use and cannot be deleted

**Note:** This method completely removes a subject. This only works if the subject is not used anywhere. To disable a used subject, please set “inactive” to false with Update Subject.

#### 4. Get all Subjects

##### Request

Method	URL	Role
GET	url/subject/all	User

##### Response

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

## 5. 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 } ]

## 6. 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 } ]

## 7. *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();
```