# TUTORINGTRAIN

# WEBSERVICE

# V1.7.2

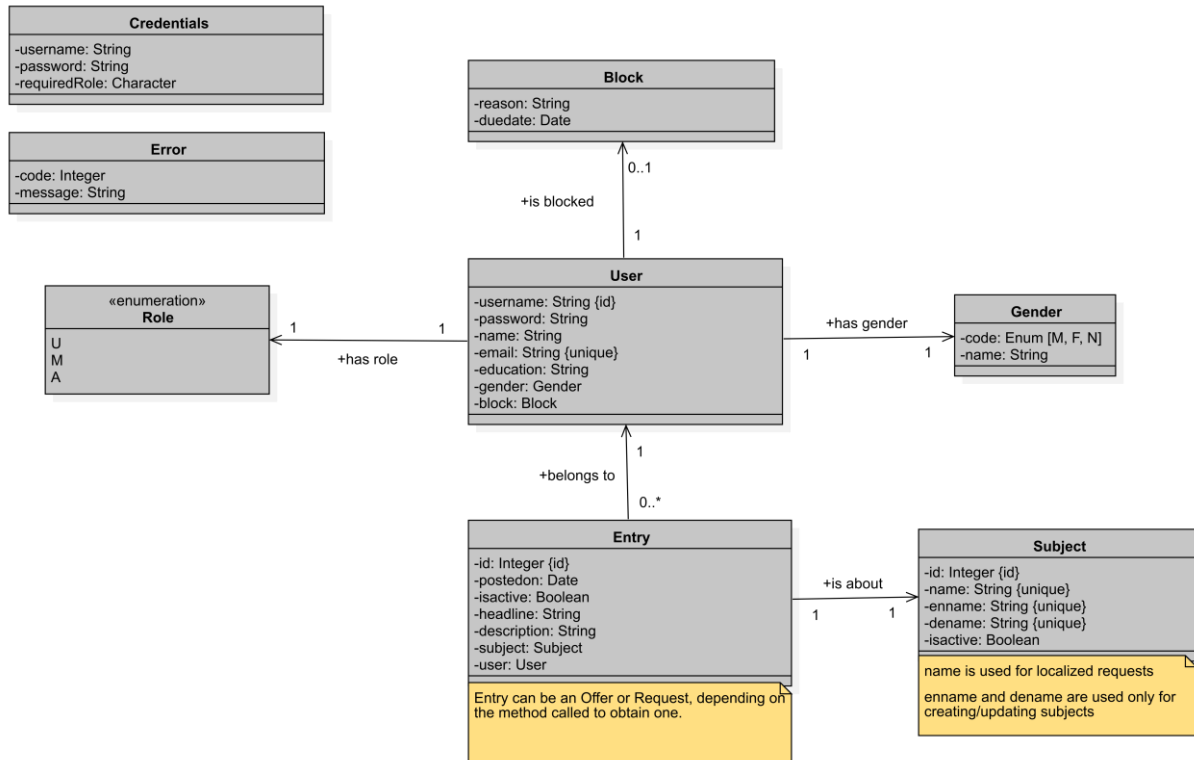REST-DOKU

# TABLE OF CONTENTS

# GENERAL

## Data Model:

**Credentials**
-username: String
-password: String
-requiredRole: Character

**Error**
-code: Integer
-message: String

**Block**
-reason: String
-duedate: Date

0..1

+is blocked

1

**User**
-username: String {id}
-password: String
-name: String
-email: String {unique}
-education: String
-gender: Gender
-block: Block

«enumeration»
**Role**
U
M
A

1        1

+has role

+has gender

1        1

**Gender**
-code: Enum [M, F, N]
-name: String

1

+belongs to

0..*

**Entry**
-id: Integer {id}
-postedon: Date
-isactive: Boolean
-headline: String
-description: String
-subject: Subject
-user: User

+is about

1        1

**Subject**
-id: Integer {id}
-name: String {unique}
-enname: String {unique}
-dename: String {unique}
-isactive: Boolean

name is used for localized requests

enname and dename are used only for creating/updating subjects

Entry can be an Offer or Request, depending on the method called to obtain one.

## 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:                   **2017-09-03T09:45:12+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 | {<br>  "username": "testuser",<br>  "password": "098f6bcd4621d373cade4e832627b4f6",<br>  "requiredRole": "A"<br>} |

**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 | {<br>  "requiredRole": "A"<br>} |

### 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 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 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. |
| 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 deactive) |
| 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 |

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

**Response**

| Status | Response |
|--------|----------|
| 200 | {<br>  "username": "kevin",<br>  "role": "U",<br>  "email": "kevin.paul@gmx.at",<br>  "name": "Kevin Paul",<br>  "education": "HTL",<br>  "gender": "F"<br>} |
| 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.

## 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"
  }
]``` |
| 455 | Invalid Query String parameters |

## *3. Get own User*

### Request

| Method | URL | Role |
|--------|-----|------|
| GET | url/user | User |

### Response

| Status | Response |
|--------|----------|
| **200** | {<br>  "username": "wacco",<br>  "role": "A",<br>  "email": "santnere@edu.htl-villach.at",<br>  "name": "wacco",<br>  "education": "HTL",<br>  "gender": "M"<br>} |

## 4. Update any User

**Request**

| Method | URL | Role |
|--------|-----|------|
| PUT | url/user/update | Admin |

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

**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 | {<br>  "name": "Elias Santner",<br>  "email": "santnere@edu.htl-villach.at",<br>  "password": "098f6bcd4621d373cade4e832627b4f6",<br>  "education": "HTL",<br>  "gender": "N"<br>} |

**Response**

| Status | Response |
|--------|----------|
| 200 | |

## *6. 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 | {<br>  "role": "U"<br>} |

## Response

| Status | Response |
|--------|----------|
| 200 | |
| 401 | Unauthorized, cannot degrade other admins or own user |

## *7. Block User*

**Request**

| Method | URL | Role |
|--------|-----|------|
| POST | url/user/block | Admin |

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

**Response**

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

## *8. 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** | |

## *9. Get all Genders*

**Request**

| Method | URL | Role |
|--------|-----|------|
| GET | url/user/gender | User |

**Response**

| Status | Response |
|--------|----------|
| 200 | ```[``` <br> ```  {``` <br> ```    "code": "F",``` <br> ```    "name": "Female"``` <br> ```  },``` <br> ```  {``` <br> ```    "code": "M",``` <br> ```    "name": "Male"``` <br> ```  },``` <br> ```  {``` <br> ```    "code": "N"``` <br> ```    "name": "Unspecified"``` <br> ```  }``` <br> ```]``` |

## *10.     Get User Count*

**Request**

| Method | URL | Role |
|--------|-----|------|
| GET | url/user/count | User |

**Response**

| Status | Response |
|--------|----------|
| **200** | 10 |

## *11.    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 |

## *12.     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 |

## *13.    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 |

## *14.    Reset Own Avatar*

**Request**

| Method | URL | Role |
|--------|-----|------|
| DELETE | url/user/avatar | User |

**Response**

| Status | Response |
|--------|----------|
| **200** | |

## *15.    Reset Avatar*

**Request**

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

**Response**

| Status | Response |
|--------|----------|
| **200** | |

## *16.    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 |

## *17.     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 |

## *18.    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>{<br>  "criteria": [<br>   {<br>     "key": "USERNAME",<br>     "value": "Adm",<br>     "operation": "CONTAINS",<br>     "ignoreCase": true<br>   },<br>   {<br>     "key": "USERNAME",<br>     "value": "in",<br>     "not": true,<br>     "operation": "ENDS_WITH",<br>     "ignoreCase": true<br>   },<br>   {<br>     "key": "GENDER",<br>     "value": "U",<br>     "not": true,<br>     "operation": "EQ"<br>   }<br>  ],<br>  "order": [<br>   {<br>     "prop": "USERNAME",<br>     "direction": "ASC"<br>   }<br>  ]<br>}</pre> |

**Response**

| Status | Response |
|--------|----------|
| 200 | A list of all found users is provide in the response body, similar to "Get All Users" |
| 451 | Invalid property 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 |

## Operations

| Type | Operations |
|---|---|
| STRING | EQ<br>CONTAINS<br>STARTS_WITH<br>ENDS_WITH |
| CHAR | EQ |

## Order Directions

| Direction |
|---|
| ASC |
| DESC |

# [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 | {<br>  "description": "This is just a test offer",<br>  "headline": "Test offer",<br>  "subject": {<br>   "id": 3<br>  }<br>} |

**Response**

| Status | Response |
|--------|----------|
| **200** | {<br>  "id": 1,<br>  "postedon": "2017-10-04T07:18:39+0200",<br>  "isactive": true,<br>  "description": "This is just a test offer",<br>  "headline": "Test offer",<br>  "subject": {<br>   "id": 1,<br>   "name": "Englisch",<br>   "isactive": true<br>  },<br>  "user": {<br>   "username": "wacco"<br>  }<br>} |
| **452** | Subject not found |
| **457** | Subject is not active |

## *2. Update Entry*

**Request**

| Method | URL | Role |
|--------|-----|------|
| PUT | url/<entryType> | User |

| MediaType | Sample Request Body |
|-----------|---------------------|
| application/json | { <br> "id": 1, <br> "isactive": false, <br> "description": "This is just a test offer", <br> "subject": { <br> "id": 1 <br> }, <br> "headline": "Test offer" <br> } |

**Response**

| Status | Response |
|--------|----------|
| 200 | |
| 452 | Subject not found |
| 454 | Entry not found |
| 457 | 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. *Reset Properties of Entry*

### Request

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

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

### Possible Properties

Duedate

### 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.

## 4. 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** | ```[
  {
   "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"
   }
  }
]``` |
| **455** | Query parameters were not given |

## 5. *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** | ```[``` <br> ```  {``` <br> ```    "id": 2,``` <br> ```    "postedon": "2017-10-04T07:27:58+0200",``` <br> ```    "isactive": true,``` <br> ```    "description": "Test offer 2",``` <br> ```    "subject": {``` <br> ```      "id": 1,``` <br> ```      "name": "Englisch",``` <br> ```      "isactive": true``` <br> ```    },``` <br> ```    "user": {``` <br> ```      "username": "wacco"``` <br> ```    }``` <br> ```  },``` <br> ```  {``` <br> ```    "id": 1,``` <br> ```    "postedon": "2017-10-04T07:18:39+0200",``` <br> ```    "isactive": false,``` <br> ```    "description": "This is just a test offer",``` <br> ```    "subject": {``` <br> ```      "id": 1,``` <br> ```      "name": "Englisch",``` <br> ```      "isactive": true``` <br> ```    },``` <br> ```    "user": {``` <br> ```      "username": "wacco"``` <br> ```    }``` <br> ```  }``` <br> ```]``` |
| **453** | User not found |
| **455** | Query parameters were not given |

## 6. Get Entry Count

**Request**

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

**Response**

| Status | Response |
|--------|----------|
| **200** | 10 |

## *7. Searching*

### Request

| Method | URL | Role |
|--------|-----|------|
| POST | url/&lt;entryType&gt;/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 | ```{
  "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"
   }
  ],
  "order": [
   {
     "key": "POSTEDON",
     "direction": "DESC"
   }
  ]
}``` |

### 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 |

## Operations

| Type | Operations |
|------|-----------|
| STRING | EQ<br>CONTAINS<br>STARTS_WITH<br>ENDS_WITH |
| DATE | EQ<br>BEFORE (does NOT include date)<br>AFTER (does NOT include date)<br>BEFORE_EQ (does include date)<br>AFTER_EQ (does include date) |
| BOOLEAN | EQ |
| NUMBER | EQ<br>GT<br>LT<br>GTE<br>LTE |

## Order Directions

| Direction |
|-----------|
| ASC |
| DESC |

# [5]  SUBJECT RESOURCE

## 1. Create Subject

**Request**

| Method | URL | Role |
|---|---|---|
| POST | url/subject | User |

| MediaType | Sample Request Body |
|---|---|
| application/json | {<br>  "dename": "Programmieren",<br>  "enname": "Programming"<br><br>} |

**Response**

| Status | Response |
|---|---|
| **200** | {<br>  "id": 2,<br>  "name": "Programmieren"<br>} |
| **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 | {<br>  "id": 4,<br>  "dename": "Mathe",<br>  "enname": "Maths",<br>  "isactive": true<br>} |

**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 "isactive" to false with Update Subject.

## *4. Get all Subjects*

**Request**

| Method | URL | Role |
|--------|-----|------|
| GET | url/subject/all | User |

**Response**

| Status | Response |
|--------|----------|
| **200** | <pre>[<br>  {<br>    "id": 2,<br>    "name": "Deutsch",<br>    "isactive": false<br>  },<br>  {<br>    "id": 1,<br>    "name": "Englisch",<br>    "isactive": true<br>  },<br>  {<br>    "id": 3,<br>    "name": "Mathe",<br>    "isactive": true<br>  }<br>]</pre> |

## 5. *Get all active Subjects*

**Request**

| Method | URL | Role |
|--------|-----|------|
| GET | url/subject | User |

**Response**

| Status | Response |
|--------|----------|
| **200** | <pre>[<br>  {<br>    "id": 2,<br>    "name": "Deutsch",<br>    "isactive": true<br>  },<br>  {<br>    "id": 1,<br>    "name": "Englisch",<br>    "isactive": true<br>  },<br>  {<br>    "id": 3,<br>    "name": "Mathe",<br>    "isactive": true<br>  }<br>]</pre> |

## 6. Get all inactive Subjects

**Request**

| Method | URL | Role |
|--------|-----|------|
| GET | url/subject/inactive | Admin |

**Response**

| Status | Response |
|--------|----------|
| **200** | ```[``` <br> ```{``` <br> ```"id": 2,``` <br> ```"name": "Deutsch",``` <br> ```"isactive": false``` <br> ```},``` <br> ```{``` <br> ```"id": 1,``` <br> ```"name": "Englisch",``` <br> ```"isactive": false``` <br> ```},``` <br> ```{``` <br> ```"id": 3,``` <br> ```"name": "Mathe",``` <br> ```"isactive": false``` <br> ```}``` <br> ```]``` |

## 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:

```java
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();
```