

HLM-NG | API

Oussama Zgheb & Tobias Zahner

20. April 2015



| Datum | Version | Änderung | Autor |
|----------|---------|--|---------------|
| 11.03.15 | 1.0 | Intial | Oussama Zgheb |
| 13.03.15 | 1.1 | Events + Speaker | Oussama Zgheb |
| 17.03.15 | 1.2 | Media | Oussama Zgheb |
| 23.03.15 | 2.0 | Weitere Items | Oussama Zgheb |
| 24.03.15 | 2.1 | JSON Beispiele | Oussama Zgheb |
| 25.03.15 | 2.2 | Delete Operations / Formatierung | Oussama Zgheb |
| 27.03.15 | 2.3 | Vote Vorgang UML | Oussama Zgheb |
| 30.03.15 | 2.4 | Erneuerungen bei Push | Oussama Zgheb |
| 30.03.15 | 2.5 | Code Formatierung, Qr Code | Oussama Zgheb |
| 31.03.15 | 2.6 | Qr Code Überprüfung / File Upload | Oussama Zgheb |
| 31.03.15 | 2.7 | Null Kennzeichnungen | Oussama Zgheb |
| 01.04.15 | 2.8 | Unique Kennzeichnung / Voting Duration | Oussama Zgheb |
| 09.04.15 | 2.9 | 'Newest' Calls + Spam Schutz | Oussama Zgheb |
| 12.04.15 | 2.91 | Lastupdatetime | Oussama Zgheb |
| 14.04.15 | 3.0 | Informationen / Konventionen gruppiert und erweitert | Oussama Zgheb |
| 17.04.15 | 3.1 | Social Items haben AuthorName Feld | Oussama Zgheb |
| 20.04.15 | 3.2 | Event Active Felder | Oussama Zgheb |

Inhaltsverzeichnis

| | | |
|----------|--|-----------|
| 1 | Einleitung | 3 |
| 2 | API | 3 |
| 2.1 | Informationen und Konventionen | 3 |
| 2.1.1 | Legende | 3 |
| 2.1.2 | Zeit und Daten | 3 |
| 2.1.3 | Diverses | 3 |
| 2.2 | Event | 4 |
| 2.3 | EventItem | 4 |
| 2.4 | EventRoom | 5 |
| 2.5 | Media | 5 |
| 2.6 | News | 6 |
| 2.7 | Presentation | 7 |
| 2.8 | Push | 7 |
| 2.9 | QrCode | 8 |
| 2.9.1 | QrCode Überprüfung | 9 |
| 2.10 | Slider | 10 |
| 2.11 | Social | 10 |
| 2.12 | Speaker | 11 |
| 2.13 | User | 11 |
| 2.14 | Vote | 12 |
| 2.14.1 | Vote Ablauf | 13 |
| 2.15 | Voting | 14 |
| 2.16 | Time | 14 |
| 3 | Fussnoten Index | 15 |

1 Einleitung

Dieses Dokument soll eine ausführliche Übersicht über die Rest API des Backend geben. Dabei sollen alle möglichen »Calls« sowie die zu erwartenden Antworten aufgeführt werden.

2 API

2.1 Informationen und Konventionen

2.1.1 Legende

- **P** = Public (kein Login benötigt)
- **S** = Secure (User Login benötigt)
Kann 401 zurückliefern
- **B** = Backend (Backend Login benötigt)
- **[Null]** kennzeichnet Felder die »null« sein dürfen
- **[!]** kennzeichnet Felder die automatisch gesetzt werden. D.h bei einem POST / PUT müssen diese nicht mitgeliefert werden, falls diese trotzdem mit gegeben werden, werden diese schlichtweg ignoriert.
- **[Unique]** kennzeichnet Felder welche Einzigartig sein müssen, falls etwas hinzugefügt / geändert wird und einen nicht Einzigartigen Wert enthält wird ein Bad Request zurückgeliefert.
- **{varname}** Variablen werden in dieser Dokumentation mit geschweiften Klammern gekennzeichnet

2.1.2 Zeit und Daten

Datum Felder sind in MySQL als String gespeichert, dies da Jersey Probleme mit MySQL »Dates« hat und ein eigener Serialisierer zu umständlich wäre. Das Format ist wie folgt **YYYY-MM-DD**.

Time Felder werden durch **HH:MM** repräsentiert

2.1.3 Diverses

- Eine fixe Reihenfolge der Properties ist bei JSON (standardmässig) nicht gewährleistet ¹
- Der Befehl »POST«
 - Liefert (falls erfolgreich) das eingefügt Element als JSON zurück
- Der Befehl »PUT«
 - Updatet lediglich - Es wird kein neues Element erstellt falls unter die definierte ID nicht existiert
 - Ignoriert eine abweichende ID
- Login
 - Bei den als **B/S** gekennzeichneten Calls wird ein Basic **HTTP-Auth** erwartet²
Das Passwort bei den S Logins ist dabei die DeviceID.
 - Bei erfolgreicher Authentifizierung wird die eigentliche Aktion durchgeführt
 - Bei fehlgeschlagener Authentifizierung wegen falschen Anmeldedaten wird erneut ein HTTP-Auth geschickt
 - Bei einem Fehler bei dem Anmeldedaten Parser wird ein Bad Request geschickt

¹4.3.3 Object - <http://goo.gl/eAjLb>

²https://de.wikipedia.org/wiki/HTTP-Authentifizierung#Basic_Authentication

- Spam Schutz
Bei folgenden Aktionen wird beim überschreiten einer Threshold für eine gewisse Zeit zu jeder Anfrage ein 429 zurückgeliefert (too many requests).
Die genauen Einstellungen sind »FileSettings« zu entnehmen.
 - Bei allen Anfragen die eine Benutzerauthentisierung fordern
 - Erstellung oder Update eines Benutzerprofils
- Last Update Time
Bei jeder Resource gibt es einen call Namens »/lastupdateimte« welcher den Zeitpunkt (in Unixzeit ³) angibt, an welchem irgend ein Element geändert wurde. So kann der Client erfahren (durch vergleichen des letzten Zeitpunktes) ob sein Datenbestand noch aktuell ist.
- Alle Pfade werden relativ angegeben

2.2 Event

eventID - Die eindeutige ID des Events

name - Der Name des Events

description - Ein kurzer Info Text zum Event

startDate - Datum beginn des Events

endDate - Datum letzter Tag des Events (kann gleich wie »from« Feld sein)

active - Gibt an ob der Event aktiv sein soll. Falls ein Event nicht aktiv ist, wird er im Frontend nicht angezeigt

```
{
  "eventID": 1,
  "name": "test",
  "description": "desc",
  "from": "2015-06-10",
  "to": "2015-06-11",
  "active": true
}
```

/event

GET - P - Liefert alle Events

POST - B - Erstellt neuen Event

/event/count

GET - P - Liefert die Anzahl Events

/event/{id}

GET - P - Liefert den besagten Event

PUT - B - Updatet den besagten Event

DELETE - B - Löscht den besagten Event

/event/{id}/eventrooms

GET - P - Liefert alle Event Rooms welche zu Event {id} gehören

event/{id}/eventitems

GET - P - Liefert alle Event Items welche zu Event {id} gehören

2.3 EventItem

eventItemID - Die eindeutige ID des Event Item

name - Der Name des Item

description - Die Beschreibung des Item

date - Das Datum an welchem das Item stattfindet

startTime - Der Beginn des Item

³<https://de.wikipedia.org/wiki/Unixzeit>

endTime - Das Ende des Item

roomIDFK - Ein FK zu dem Raum in welchem das Item stattfindet

eventIDFK - Ein FK zu dem Event in welchem das Item stattfindet

```
{
  "eventItemID": 1,
  "name": "Advanced Cryptography",
  "description": "description of item..",
  "date": "2015-06-15",
  "startTime": "13:00",
  "endTime": "14:15",
  "roomIDFK": 1,
  "eventIDFK": 1
}
```

/eventitem

GET - P - Liefert alle Event Items

POST - B - Erstellt neues Event Item

/eventitem/{id}

GET - P - Liefert besagtes Event Item

PUT - B - Updatet besagtes Event Item

DELETE - B - Löscht besagtes

/eventitem/eventroom

GET - P - Liefert den Raum in welchem das Event Item stattfindet

2.4 EventRoom

eventRoomID - Die eindeutige ID des Raumes

name - Die Raumbezeichnung

location - [Null] Eine genauere Bezeichnung zum Raum

eventIDFK - Ein FK zu dem Event in welchem der Raum verwendet wird

```
{
  "eventRoomID": 1,
  "name": "1.206",
  "location": "1",
  "eventIDFK": 1
}
```

/eventroom

GET - P - Liefert alle Event Rooms

POST - B - 401 - Erstellt neuen Event Room

/eventroom/{id}

GET - P - Liefert den besagten Event Room

PUT - B - Updatet den besagten Event Room

DELETE - B - Löscht den besagten Event Room

2.5 Media

mediaID - Die eindeutige ID jedes Medien Objektes

type - Der Kennzeichner des Medien Objektes. »jpeg«, »png« (TODO weitere Typen wie Video etc.)

link - Ein absoluter Link zur Ressource

```
{
  "mediaID": 1,
  "type": "jpg",
}
```

```
    "link": "http://localhost:8080/hlmng/rest/media/jpeg/1.jpg",
  }
```

/media

GET - P - Liefert alle Medien

/media/{id}

GET - P - Liefert das besagte Medien Objekt

DELETE - B - Löscht die verlinkte Medien Datei, der Eintrag in der Datenbank wird **nicht gelöscht**.

Dies da auf das Medien Objekt verlinkende Objekte weiterhin den Link benutzen können und dieser dann ein 404 zurückliefert.

/media/{typ}/{filename}

GET - P - Liefert die gewünschte Medien Ressource

z.B. `http://.../rest/media/image/png/image.png`

/media/upload

POST - P - Ermöglicht einen Upload von Media Dateien mittels Multipart-Data ⁴.

Die möglichen Return Codes sind (nebst den üblichen Authorization needed / Bad Request).

- 500 - Server konnte das File nicht speichern
- 415 - Falscher / fehlender / unbekannter Mimetype
- 422 - Datei mit gleichem Namen existiert bereits
- 413- Datei zu gross
- 200 - Hochgeladen, Objekt wird als JSON geliefert, sodass sich der Client die ID speichern kann

/media/uploadbackend

POST - B - Für Backendlogins, ansonsten gleich wie »/upload«.

2.6 News

newsID - Die eindeutige ID zu dem News Eintrag

title - Der Titel des News Eintrag

text - Der Text des News Eintrag

media - Der direkte Link zur angehängten Medien Datei

author - Der Autor (frei wählbarer String)

mediaIDFK - [Null] Ein FK zu der angehängten Medien Datei

eventIDFK - Ein FK zu dem Event zu welchem die News gehören

```
{
  "newsID": 1,
  "title": "Breaking news!",
  "text": "Team Switzerland takes the lead!",
  "media": "http://localhost:8080/hlmng/rest/media/jpg/1.jpg",
  "author": "Max Muster",
  "mediaIDFK": 1,
  "eventIDFK": 1
}
```

/news

GET - P - Liefert alle News Einträge

POST - B - Erstellt neuen News Eintrag

/news/newest

GET - P - Liefert nur die neusten (höchste ID zuerst) 15 (Standardwert) Einträge. Es wird empfohlen diesen Call zu werden, ausser der User wünscht explizit alle Einträge.

/news/{id}

⁴<https://stackoverflow.com/questions/913626>

GET - P - Liefert den besagten News Eintrag
 PUT - B - Updatet den besagten News Eintrag
 DELETE - B - Löscht den besagten News Eintrag

/news/{id}/media

GET - P - Liefert das Medien Objekt zu besagten News Eintrag

2.7 Presentation

presentationID - Die eindeutige ID zu der Präsentation
name - Der durch das Team gewählte Name der Präsentation
teamName - Der Teamname
date - Das Datum an welchem die Präsentation durchgeführt wurde
duration - [!] [Null] Die Dauer der Präsentation

```
{
  "presentationID": 1,
  "name": "presentation 1",
  "teamName": "1336+1",
  "date": "2015-05-10",
  "duration": "00:12:35"
}
```

/presentation

GET - P - Liefert alle Präsentationen
 POST - B - Erstellt neue Präsentation

/presentation/{id}

GET - P - Liefert die besagte Präsentation
 PUT - B - Updatet die besagte Präsentation
 DELETE - B - Löscht die besagte Präsentation

2.8 Push

pushID - Die eindeutige ID des Push
title - Der Titel der Push Nachricht
author - Der Autor (frei wählbarer String)
date - Das Datum an dem die Nachricht gesendet wurde
time - Der Zeitpunkt an dem die Nachricht gesendet wurde
receivedCounter - Die Anzahl Mobile Apps welche die Nachricht erhalten haben
failedCounter - Die Anzahl Mobile Apps an welche die Nachricht nicht gesendet werden konnte (Details siehe Log)

```
{
  "pushID": 1,
  "title": "push title",
  "author": "mmuster",
  "date": "2015-05-20",
  "time": "14:15",
  "receivedCounter": 10,
  "failedCounter": 2
}
```

/push

GET - P - Liefert alle Push Einträge
 POST - B Erstellt neuen Push Eintrag
 Der Post sollte etwa wie folgt aussehen:

```
{
  "author": "mmuster",
```

```

    "text": "it works",
    "title": "unbelievable"
  }

```

/push/{id}

GET - P - Liefert besagten Push Eintrag

PUT - B - Updatet besagten Push Eintrag *

DELETE - B - Löscht den besagten Push Eintrag *

* Die Push Notifaction auf dem Mobile App kann logischerweise nicht nachträglich geändert / gelöscht werden.

2.9 QRCode

qrCodeID - Die eindeutige ID des QR Code

createdAt - [!] Der Erstellungszeitpunkt des QR Code

claimedAt - [!] [Null] Der Zeitpunkt an dem der QR Code eingelöst wurde

payload - [Unique] Die Payload des QR Code.

role - Die Rolle welche der QR Code legitimiert [jury,author]

userIDFK - [Null] Ein FK zu dem User welcher den QR Code »geclaimt« hat

eventIDFK - Ein FK zu dem Event in welchem der QR Code gültig ist

```

{
    "qrCodeID": 1,
    "createdAt": "2015-06-21 22:59:59",
    "claimedAt": "2015-06-21 23:09:13",
    "payload": "jdi7.di-n23i*msdgz?db*p",
    "role": "jury",
    "userIDFK": 1,
    "eventIDFK": 1
}

```

/qrcode

GET - B - Liefert alle QR Codes

POST - B - Erstellt neuen QR Code

/qrcode/{id}

GET - B - Liefert den besagten QR Code

PUT - B - Updatet den besagten QR Code

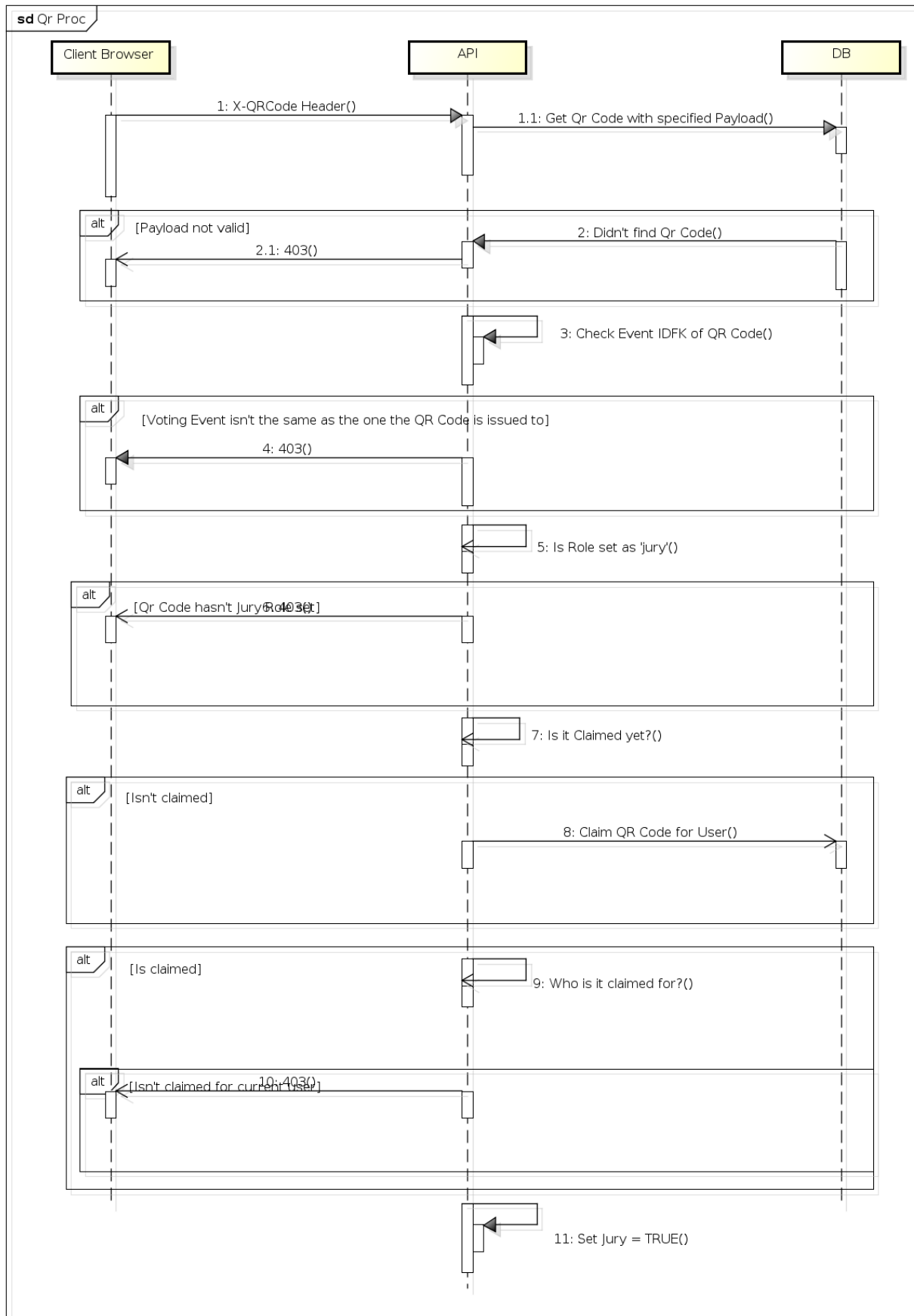
DELETE - B - Löscht den besagten QR Code

/qrcode/{id}/render

GET - B - Liefert den QR Code als Bild zurück

2.9.1 QrCode Überprüfung

Folgendes Diagramm soll den Ablauf und die möglichen Return Codes aufzeigen,



2.10 Slider

sliderID - Die eindeutige ID des Slider

name - Die Bezeichnung des Slider

weight - Die Gewichtung des Sliders

votingIDFK - Ein FK zu dem dazugehörigen Voting

```
{
    "sliderID": 1,
    "name": "Language",
    "weight": 2,
    "votingIDFK": 1
}
```

/slider

GET - P - Liefert alle Slider

POST - B - Erstellt neuen Slider

/slider/{id}

GET - P - Liefert den besagten Slider

POST - B - Updatet den besagten Slider

DELETE - B - Löscht den besagten Slider

2.11 Social

socialID - Die eindeutige ID des Social Eintrages

text - Der Text des Eintrages

status - Der Moderationsstatus [pending,accepted,rejected]

authorName - Der Username des Social Items, falls der User nicht gefunden wird ist das Feld »unknown«

media - [Null] Der direkte Link zu der angehängten Medien Datei (falls vorhanden)

userIDFK - Ein FK zu dem User der den Social Eintrag erstellt hat

mediaIDFK - [Null] Ein FK zu dem Medien Objekt welches angehängt sein kann

eventIDFK - Ein FK zu dem Event in welchem der Social Eintrag erstellt wurde

```
{
    "socialID": 1,
    "text": "Nice presentation!",
    "status": "pending",
    "authorName": "Harrison Ford",
    "media": "http://localhost:8080/hlmng/rest/media/jpg/1.jpg",
    "userIDFK": 1,
    "mediaIDFK": 1,
    "eventIDFK": 1
}
```

/social

GET - P - Liefert alle Social Einträge

POST - S - Erstellt neuen Social Eintrag

/social/newest

GET - P - Liefert nur die neusten (höchste ID zuerst) 15 (Standardwert) Einträge. Es wird empfohlen diesen Call zu verwenden, ausser der User wünscht explizit alle Einträge.

/slider/{id}

GET - P - Liefert den besagten Slider

POST - B - Updatet den besagten Slider

DELETE - B - Löscht den besagten Slider

2.12 Speaker

speakerID - Die eindeutige ID des Speaker

name - Der Vor- und Nachname des Speaker

description - Ein kurzer Info Text zum Speaker

media - **[Null]** Der direkte Link zu der angehängten Medien Datei (falls vorhanden)

nationality - Das Kürzel zu dem Heimatland des Speaker. **Achtung:** Die Nationality wird immer in toUpperCase umgewandelt.

title - **[Null]** Der akademische Titel des Speaker, falls vorhanden

mediaIDFK - Ein FK zu dem Speaker Bild

```
{
  "speakerID": 1,
  "name": "Richard Stallman",
  "title": "PhD MIT",
  "media": "http://localhost:8080/hlmng/rest/media/jpg/1.jpg",
  "description": ".. freedom activist and computer programmer ..",
  "nationality": "USA"
  "mediaIDFK": 1
}
```

/speaker

GET - P - Liefert alle Speaker

POST - B - Erstellt neuen Speaker

/speaker/{id}

GET - P - Liefert den besagten Speaker

POST - B - Updatet den besagten Speaker

DELETE - B - Löscht den besagten Speaker

2.13 User

userID - Die eindeutige ID welche den Benutzer identifiziert

name - Der Benutzername

deviceID - Die eindeutige ID jedes Android Gerätes

regID - Die Registrations ID bei dem Google Cloud Messaging Dienst

```
{
  "userID": 3,
  "name": "hmuster",
  "deviceID": "1234567890987654321",
  "regID": "1aa23cd45ef6789fg098hk765432ff1..."
}
```

/user

GET - P - Liefert alle user

POST - P - Erstellt neuen user. Falls der Username bereits existiert wird 422 (unprocessable entity / exists) zurückgeliefert.

/user/{id}

GET - P - Liefert den besagten User

PUT - B - Updaten den besagten User

DELETE - B - Löscht den besagten

{id}/changeregid

PUT - S - Updatet die »regID« des Users welcher sich anmeldet. Ein im JSON abweichender Username wird zurückgewiesen. Alle Variablen ausser »regID« werden **nicht** geändert.

2.14 Vote

voteID - Die eindeutige ID des Vote (= eine Stimme)

score - Die abgebende Bewertung, beginnend bei 0 bis zu sliderMaxValue von Voting

isJury - Sagt aus ob die Vote von einem Jury Mitglied gemacht wurde oder nicht

sliderIDFK - Ein FK zu dem Slider über den die Bewertung abgegeben wurde

userIDFK - Ein FK zu dem Benutzer welcher »gevotes« hat

```
{  
    "voteID": 1,  
    "score": 7,  
    "isJury": true,  
    "sliderIDFK": 1,  
    "userIDFK": 1  
}
```

/vote

GET - B - Liefert alle Votes

POST - S - Erstellt neuen Vote

/vote/{id}

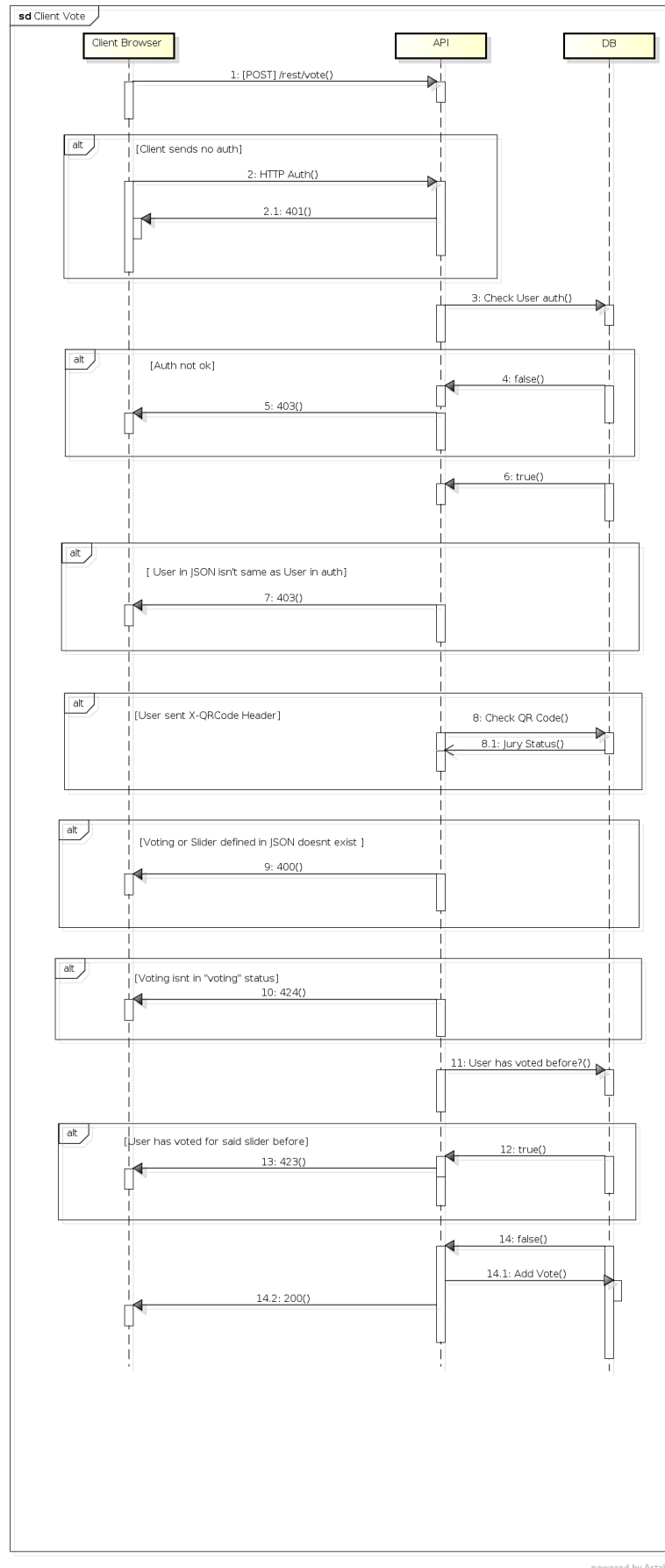
GET - B - Liefert den besagten Vote

/vote/slider/{id}

GET - B - Liefert alle Votes zu dem Slider {id}

2.14.1 Vote Ablauf

Folgendes Diagramm soll den Ablauf und die möglichen Return Codes aufzeigen,



2.15 Voting

votingID - Die eindeutige ID des Votings

name - Der Name des Votings

juryCount - Die Anzahl anwesender Jury Mitglieder

status - Der Status des Voting (pre_presentation,presentation,presentation_end,voting,voting_end)

sliderMaxValue - Der maximale Wert der Slider

votingStarted - [Null] Der Zeitpunkt an welchem das Voting gestartet wurde (also status=voting)

votingDuration - Die Zeit in der, nach dem erhaltenen Push, abgestimmt werden darf

arethmeticMode - Die Auswahl der mathematischen Funktion zur Auswertung (median,todo)

round - Die Runde **TODO**

presentationIDFK - Ein FK zu der Präsentation über welche abgestimmt wird

```
{
  "votingID": 1,
  "name": "Voting 1",
  "juryCount": 15,
  "status": "running",
  "votingStarted": "14:04:27",
  "votingDuration": "00:00:50",
  "sliderMaxValue": 10,
  "arethmeticMode": "median",
  "round": 1,
  "presentationIDFK": 1
}
```

/voting

GET - P - Liefert alle Votings

POST - B - Erstellt neues Voting

/voting/{id}

GET - P - Liefert besagtes Votings

PUT - B - Updatet besagtes

/voting/{id}/slider

GET - P - Liefert alle Slider des besagten Voting

2.16 Time

Liefert die aktuelle Serverzeit im Format HH-mm-ss.SSS.

3 Fussnoten Index

| | | |
|---|---|---|
| 1 | 4.3.3 Object - http://goo.gl/eAjLb | 3 |
| 2 | https://de.wikipedia.org/wiki/HTTP-Authentifizierung#Basic_Authentication | 3 |
| 3 | https://de.wikipedia.org/wiki/Unixzeit | 4 |
| 4 | https://stackoverflow.com/questions/913626 | 6 |