

Entwurf REST-API

Inhaltsverzeichnis

| | |
|--|----|
| Einleitung..... | 3 |
| Ressourcen | 3 |
| Was muss der REST-Service potenziell bereitstellen können? | 3 |
| Fehlerbehandlung | 4 |
| Entwurf der URIs | 7 |
| Anforderungen: | 8 |
| /benutzer..... | 8 |
| POST | 8 |
| PUT: | 10 |
| /tag | 11 |
| GET..... | 11 |
| POST | 11 |
| /tag/{id}..... | 12 |
| DELETE | 12 |
| /freundschaft..... | 12 |
| GET..... | 12 |
| POST | 13 |
| /freundschaft/{id}..... | 13 |
| PUT | 13 |
| DELETE | 13 |
| /raum..... | 14 |
| GET..... | 14 |
| /raum/{id}..... | 15 |
| GET..... | 15 |
| PUT | 16 |
| /veranstaltung..... | 17 |
| GET..... | 17 |
| POST | 17 |
| /veranstaltung/{id}..... | 18 |
| GET..... | 18 |
| PUT | 19 |
| DELETE | 20 |
| /sitzung..... | 21 |

| | |
|---------------------------------------|----|
| GET..... | 21 |
| POST | 22 |
| /sitzung/{id}..... | 23 |
| PUT | 23 |
| DELETE | 23 |
| /konfiguration | 24 |
| GET..... | 24 |
| PUT | 24 |
| /karte..... | 24 |
| Notizen für Später / Nützliches | 24 |

Einleitung

Interaktion mit dem Webservice erfolgt nur per JSON!

Das bedeutet es müssen ACCEPT und CONTENT im HTTP-Header gesetzt sein, ansonsten werden entsprechende Fehlermeldungen erzeugt!

Ressourcen

Der Webservice soll folgende Ressourcen repräsentieren.

Für den Bau der URI ist es wichtig, sich auf eine Bezeichnung der Ressource zu einigen (Singular / Plural). Abstimmung!

Eine einheitliche URI ist ein Maß für die Qualität des Rest-Services!

| Singular | Plural |
|---------------|-----------------|
| Freundschaft | Freundschaften |
| Benutzer | Benutzer |
| Sitzung | Sitzungen |
| Veranstaltung | Veranstaltungen |
| Raum | Räume |
| Tag | Tags |
| Konfiguration | Konfigurationen |
| Karte | Karten |

Es wurde im Team abgestimmt, dass wir für die Beschreibung der Ressourcen einheitlich den Singular benutzen!

Was muss der REST-Service potenziell bereitstellen können?

Der REST-Service muss potenziell alle Instanzen von Ressourcen und Listen dieser bereitstellen können.

Verlinkte Objekte bzw. Aggregationen sollen als Integration bereitgestellt werden.

Interaktion mit Ressourcen

| HTTP-Methode | CRUD-Operation | Beschreibung | Idempotent |
|--------------|----------------|--------------|-----------------------|
| POST | CREATE | | (JA) nach Möglichkeit |
| GET | READ | | JA |
| PUT | UPDATE | | JA |
| DELETE | DELETE | | JA |

Das Einfügen von Ressourcen soll nach Möglichkeit Idempotent implementiert werden! (Insert or Update)

Fehlerbehandlung

Einige Fehlercodes (u.A. 401, 404, 500) können bei jeder Ressource auftreten!

| Statuscode | Fehlerbeschreibung |
|------------|-----------------------|
| 200 | OK |
| 201 | CREATED |
| | |
| 400 | BAD REQUEST |
| 401 | UNAUTHORIZED |
| 403 | FORBIDDEN |
| 404 | NOT FOUND |
| | |
| 500 | INTERNAL SERVER ERROR |
| 501 | NOT IMPLEMENTED |
| | |
| 900 | NO ACTIVE SESSION |
| 910 | ROOM BLOCKED |

Status 200/201 OK/CREATED:

Wird bei korrekter Ausführung zurückgegeben

```
1. {  
2.     //Objekt je nach verwendeter Methode  
3. }
```

Status 400 BAD REQUEST

Wird benutzt, wenn ungültige Daten gesendet werden (Beispielsweise ungültige ID)

```
1. {  
2.   "status":  
3.   {  
4.     "code":400,  
5.     "description":"BAD REQUEST"  
6.   }  
7. }
```

Status 401 UNAUTHORIZED

Wird benutzt, wenn der Token ungültig ist

```
1. {  
2.   "status":  
3.   {  
4.     "code":401,  
5.     "description":"UNAUTHORIZED"  
6.   }  
7. }
```

Status 403 FORBIDDEN:

Wird zurückgegeben, wenn der Benutzer nicht genügend Rechte hat um die gewünschte Operation auszuführen.

```
1. {
2.   "status":
3.   {
4.     "code":403,
5.     "description":"FORBIDDEN"
6.   }
7. }
```

Status 404 NOT FOUND:

Wird benutzt, wenn eine unbekannte Ressource angefordert wird.

Sollte Grizzly bereits automatisch machen, eventuell muss die Standardantwort von Grizzly geändert werden, um die unten beschriebene Antwort zu senden!

```
1. {
2.   "status":
3.   {
4.     "code":404,
5.     "description":"NOT FOUND"
6.   }
7. }
```

Status 500 – INTERNAL SERVER ERROR

Wird bei Fehlern in der Verarbeitung (Bsp. Datenbank-Fehler) zurückgegeben. Bitte darauf achten, dass jeder noch so kleine Fehler zu Status 500 führt!

```
1. {
2.   "status":
3.   {
4.     "code":500,
5.     "description":"INTERNAL SERVER ERROR"
6.   }
7. }
```

Status 501 – NOT IMPLEMENTED

Wird bei nicht implementierten Methoden zurückgegeben. Bei Ressourcen die oben beschrieben sind, sollte nie 404 kommen!

```
1. {
2.   "status":
3.   {
4.     "code":501,
5.     "description":"NOT IMPLEMENTED"
6.   }
7. }
```

Status 900 – NO ACTIVE SESSION

Wird benutzt um zu zeigen, ob eine aktive Sitzung angezeigt werden soll, oder der Homescreen

```
1. {  
2.   "status":  
3.   {  
4.     "code":900,  
5.     "description":"NO ACTIVE SESSION"  
6.   }  
7. }
```

Status 910 – ROOM BLOCKED

Wird benutzt um zu zeigen, dass ein Raum zur gewählten Zeit bereits blockiert ist.

```
1. {  
2.   "status":  
3.   {  
4.     "code":910,  
5.     "description":"ROOM BLOCKED"  
6.   }  
7. }
```

Entwurf der URIs

| URI | HTTP-Methode | Ergebnis | Status |
|-------------------------|--------------|-------------------------------|--------------------|
| /benutzer | POST | Neuer Benutzer / Registrieren | 200,201 |
| /benutzer | PUT | Masterpasswort senden | 201,403 |
| | | | |
| /tag | GET | Liste der Tags | 200 |
| /tag | POST | Neuer Tag | 201 |
| /tag/{id} | DELETE | Tag löschen | 200, 400 |
| | | | |
| /freundschaft | GET | Freundschaften (alle Status) | 200 |
| /freundschaft | POST | Neue Freundschaftsanfrage | 200 |
| /freundschaft/{id} | PUT | Freundschafts-Status ändern | 200, 400 |
| /freundschaft/{id} | DELETE | Freundschaft löschen | 200, 400 |
| | | | |
| /raum | GET | Liste der Räume | 200 |
| /raum/{id} | GET | Raumdetails | 200, 400 |
| /raum/{id} | PUT | TAG SETZEN | 200, 400, 403 |
| | | | |
| /veranstaltung | GET | Liste der Veranstaltungen | 200, 403 |
| /veranstaltung | POST | Neue Veranstaltung | 201, 403, 910 |
| /veranstaltung/{id} | GET | Veranstaltungsdetails | 200, 400, 403 |
| /veranstaltung/{id} | PUT | Veranstaltung ändern | 200, 400, 403, 910 |
| /veranstaltung/{id} | DELETE | Veranstaltung löschen | 200, 400, 403 |
| | | | |
| /sitzung | GET | Aktive Sitzung des Fragenden | 200, 900 |
| /sitzung | POST | Neue Sitzung | 201 |
| /sitzung/{id} | PUT | Sitzung ändern / verlängern | 200, 400, |
| /sitzung/{id} | DELETE | Sitzung löschen | 200, 400, 403 |
| | | | |
| TODO: | | | |
| /konfiguration | GET | Anfordern der Konfiguration | |
| /konfiguration | PUT | Ändern der Konfiguration | |
| | | | |
| /karte | GET | Karte | |
| /karte?start=XX?ziel=YY | GET | Karte mit Weglinien | |
| | | | |
| /raum/{id}/foto | GET | Foto von angefragtem Raum | 200, 400, 404 |

Anforderungen:

1. JEDE Anfrage, außer POST: [/benutzer](#), muss die Benutzer-Identifikation und ein Token enthalten, diese Angaben werden im Body der Anfrage übergeben und serverseitig geprüft. Bei falschem Token wird [Status 401 UNAUTHORIZED](#) zurückgegeben.
2. Der Client erhält den Token als Antwort auf seine Registrierung(POST: [/benutzer](#))
3. Der Client erhält Daten nur nach seiner Benutzer-Identifikation gefiltert
Beispiel: Benutzer A erhält nur seine Freunde in der Freundesliste!
4. Jede Anfrage an den Server liefert auch eine Antwort.
5. Jede Antwort vom Server beinhaltet einen passenden [HTTP-Status-Code](#)
6. Zeitstempel und Zeitangaben werden nur im [UNIX-Format](#) geliefert und gesendet!

Authentifizierung:

Als Authentifizierung wird Basic Auth von HTTP genutzt.

Wie funktioniert Basic Auth:

In dem HTTP-Header (PAKET NICHT PER GET!) wird ein Feld namens Authorization übertragen!

Dieses Feld muss zwingend folgende Werte enthalten.

Der Server akzeptiert nur Pakete, die folgendes Format einhalten, andere Pakete werden auf Server-Seite verworfen!

HTTP-Header:

Authorization: Basic Base64(<username>:<token>)

Base64 macht aus dem String eine 64Bit codierte Zeichenkette, dieser String muss auf dem Client generiert werden!

[/benutzer](#)

POST

Diese Ressource wird benutzt, wenn der Benutzer auf einloggen klickt oder die App öffnet

Client:

```
1. {  
2.     "id": "32423423jok4n23oasibf23ri2b3ufb2",  
3.     "email": "thomas.gorgels@googlemail.com",  
4.     "name": "Gorgels",  
5.     "vorname": "Thomas",  
6.     "fotoURL": "http://google.com/profilbild"  
7. }
```


Antworten:

Status 200 OK

Wenn der Benutzer wiederkehrend ist

```
1. {  
2.     "id": "32423423jok4n23oasibf23ri2b3ufb2",  
3.     "email": "thomas.gorgels@googlemail.com",  
4.     "name": "Gorgels",  
5.     "vorname": "Thomas",  
6.     "fotoURL": "http://google.com/profilbild",  
7.     "token": "asdjkbaws923enadnh9q3qdn398z23e",  
8.     "istProfessor": "false"  
9.     "istAnonym": "false"  
10. }
```

Status 201 CREATED

Wenn der Benutzer neu ist

```
1. {
2.     "id": "32423423jok4n23oasibf23ri2b3ufb2",
3.     "email": "thomas.gorgels@googlemail.com",
4.     "name": "Gorgels",
5.     "vorname": "Thomas",
6.     "fotoURL": "http://google.com/profilbild",
7.     "token": "asdjkbaws923enadnh9q3qdn398z23e",
8.     "professor": "false"
9.     "istAnonym": "false"
10. }
```

PUT:

Die Ressource wird benutzt, wenn das Masterpasswort eingegeben wurde

Client:

```
1. {
2.     "masterpasswort": "MatheIstToll"
3. }
```

Antworten:

Status 200 OK:

Masterpasswort korrekt

```
1. {
2.     "id": "32423423jok4n23oasibf23ri2b3ufb2",
3.     "email": "thomas.gorgels@googlemail.com",
4.     "name": "Gorgels",
5.     "vorname": "thomas",
6.     "foto": "http://google.com/profilbild",
7.     "token": "asdjkbaws923enadnh9q3qdn398z23e",
8.     "istProfessor": "true"
9.     "istAnonym": "false"
10. }
```

Status 403 FORBIDDEN

Masterpasswort falsch

/tag

GET

Die Ressource wird benutzt, um eine Liste von Tags zu erhalten

Client: Siehe Authentifizierung!

Antworten:

Status 200 OK

```
1. [
2.     {
3.         "id": "4711",
4.         "name": "Präsentation"
5.     },
6.     {
7.         "id": "4712",
8.         "name": "Ruhe"
9.     }
10. ]
```

POST

Die Ressource wird benutzt, um einen neuen Tag hinzuzufügen

Client:

```
1. {
2.     "name": "Gruppenarbeit"
3. }
```

Antworten:

Status 201CREATED

```
1. {
2.     "id": "4713",
3.     "name": "Gruppenarbeit"
4. }
```

/tag/{id}

DELETE

Die Ressource wird benutzt, um einen Tag zu löschen

{id}= ID des zu löschenden Tags

Client: Siehe Authentifizierung!

Antworten:

Status 200 OK

Status 400BAD REQUEST

/freundschaft

GET

Die Ressource wird benutzt um eine Liste von allen Freundschaften (d.h. Freunde und Freundschaftsanfragen) zu erhalten.

Client: Siehe Authentifizierung!

Antworten:

Status 200 OK

```
1.  [
2.  {
3.      "benutzer":
4.      {
5.          //Benutzer-Objekt
6.      },
7.      "status":"freund" //Boolean
8.      "raum":
9.      {
10.         //Raumobjekt, wenn vorhanden
11.     }
12. },
13. {
14.     "benutzer":
15.     {
16.         //Benutzer-Objekt
17.     },
18.     "status":"anfrage" //Boolean
19.     "raum":NULL //leer
20. }
21. ]
```

POST

Die Ressource wird benutzt, um einen neuen Freund hinzuzufügen.

Client:

```
1. {  
2.     email: thomas.gorgels@googlemail.com  
3. }  
4.
```

Antworten:

Status 200 OK

- Der Server erstellt eine neue Freundschaft mit dem Status „anfrage“
- Besteht schon eine Freundschaft zwischen den Benutzern, wird trotzdem 200 OK ohne Änderungen am Datenbestand zurückgegeben
- Gibt es keinen Benutzer mit der angegebenen Email-Adresse, wird trotzdem 200 OK zurückgegeben.

/freundschaft/{id}

PUT

Die Ressource wird benutzt, um eine Freundschaftsanfrage zu bestätigen.

Es soll geprüft werden, ob die Freundschaftsanfrage auch wirklich von dem User ist, der die Ressource konsumiert!

Client: Siehe Authentifizierung!

{id} = Benutzeridentifikation des hinzuzufügenden Freundes

Antworten:

Status 200 OK

DELETE

Die Ressource wird benutzt, um eine Freundschaft zu bestätigen

Client: Siehe Authentifizierung!

{id} = Benutzeridentifikation des zu löschenden Freundes

Antworten:

Status 200 –OK

[/raum](#)

GET

Die Ressource wird benutzt, um eine Liste aller Räume zu bekommen

Client: Siehe Authentifizierung!

Antworten:

Status 200 OK

```
1.  [
2.  {
3.      "id": "1234",
4.      "raumnummer": "G101",
5.      "teilnehmer_max": "32",
6.      "teilnehmer_aktuell": "15",
7.      "status": "rot/gelb/grün/grau" // enum?!,
8.      "fotoURL": "http://<server>/raum/{id}/foto",
9.      "tag":
10.     {
11.         "id": "4711",
12.         "name": "Präsentation",
13.     }
14. },
15. {
16.     "id": "1235",
17.     "raumnummer": "G102",
18.     "teilnehmer_max": "32",
19.     "teilnehmer_aktuell": "17",
20.     "status": "rot/gelb/grün",
21.     "fotoURL": "http://<server>/raum/{id}/foto",
22.     "tag":
23.     {
24.         "id": "4711",
25.         "name": "Präsentation",
26.     }
27. }
28. ]
```

/raum/{id}

GET

Die Ressource wird benutzt, um die Raumdetails für einen Raum zu erhalten

{id}: ID des Raums zu dem Details erwünscht sind

Client: Siehe Authentifizierung!

Antwort:

Status 200 OK

```
1. {  
2.     "id": "1235",  
3.     "raumnummer": "G102",  
4.     "teilnehmer_max": "32",  
5.     "teilnehmer_aktuell": "17",  
6.     "status": "rot/gelb/grün",  
7.     "fotoURL": "http://<server>/raum/{id}/foto",  
8.     "tag":  
9.     {  
10.         "id": "4711",  
11.         "name": "Präsentation",  
12.     }  
13.     "benutzer":  
14.     [  
15.         {Benutzer-Objekt1},  
16.         {Benutzer-Objekt2}  
17.     ] //nur nicht anonyme  
18. }
```

PUT

Die Ressource wird benutzt, um einen Tag in einem Raum zu setzen

Client: Siehe Authentifizierung!

```
1. {  
2.     "tag": "4712"  
3. }
```

Antworten:

Status 200 OK

```
1. {  
2.     "raumnummer": "G101",  
3.     "teilnehmer_max": "32",  
4.     "teilnehmer_aktuell": "17",  
5.     "status": "rot/gelb/grün",  
6.     "fotoURL": "http://<server>/raum/{id}/foto",  
7.     "tag":  
8.     {  
9.         "id": "4712",  
10.        "name": "Ruhe",  
11.    }  
12. }
```

Status 403 FORBIDDEN

Wenn schon ein Tag gesetzt wurde, bevor der Client den Tag gesetzt hat. Antwort enthält trotzdem Raumobjekt

/veranstaltung

ACHTUNG: DIESE RESSOURCE KANN NUR VON PROFESSOREN GENUTZT WERDEN!

GET

Die Ressource wird benutzt, um eine Liste aller von diesem Benutzer erstellten Veranstaltungen zu bekommen

Client: Siehe Authentifizierung!

Antworten:

Status 200 OK

```
[
```

```
1.     {Veranstaltungs-Objekt},
2.     {Veranstaltungs-Objekt}
3. ]
4.
```

Status 403 FORBIDDEN

POST

Die Ressource wird benutzt, um eine neue Veranstaltung zu erstellen.

Client: Siehe Authentifizierung!

```
1. {
2.     "name": "hallo123"
3.     "von": "11.11.2016 13:00",    //UNIX Timestamp
4.     "bis": "11.11.2016 15:00",   //UNIX Timestamp
5.     "raum": "4711"
6. }
7.
```

Antworten:

Status 201 CREATED:

```
1. {
2.     "id": "1188",
3.     "name": "SWE VL"
4.     "dozent":
5.     {
6.         //Benutzer
7.     },
8.     "von": "11.11.2016 13:00", //UNIX Timestamp
9.     "bis": "11.11.2016 15:00", //UNIX Timestamp
10.    "raum":
11.    {
12.        //Raum der Veranstaltung
13.    }
14. }
```

Status 403 FORBIDDEN

Status 910 ROOM BLOCKED

[/veranstaltung/{id}](#)

{id}=ID der Veranstaltung

GET

Die Ressource wird benutzt, um Informationen über eine einzelne Veranstaltung zu bekommen

Client: Siehe Authentifizierung!

Antworten:

Status 200 OK

```
15. {
16.     "id": "1188",
17.     "name": "SWE VL"
18.     "dozent":
19.     {
20.         //Benutzer
21.     },
22.     "von": "11.11.2016 13:00", //UNIX Timestamp
23.     "bis": "11.11.2016 15:00", //UNIX Timestamp
24.    "raum":
25.    {
26.        //Raum der Veranstaltung
27.    }
28. }
```

PUT

Client: Siehe Authentifizierung!

Die Ressource wird benutzt, um eine Veranstaltung zu bearbeiten

```
1. {
2.     "name": "SWE VL"
3.     "von": "11.11.2016 13:00", //UNIX Timestamp
4.     "bis": "11.11.2016 15:00", //UNIX Timestamp
5.     "raum":
6.     {
7.         //Raum der Veranstaltung
8.     }
9. }
```

Antwort

Status 200 OK

```
1. {
2.     "id": "1188",
3.     "name": "SWE VL"
4.     "benutzer":
5.     {
6.         //Benutzer
7.     },
8.     "von": "11.11.2016 13:00", //UNIX Timestamp
9.     "bis": "11.11.2016 15:00", //UNIX Timestamp
10.    "raum":
11.    {
12.        // Raum der Veranstaltung
13.    }
14. }
```

Status 403 FORBIDDEN

Status 910 ROOM BLOCKED

DELETE

Diese Ressource wird benutzt, um eine Veranstaltung zu löschen

Client: Siehe Authentifizierung!

Antworten:

Status 200 OK

Status 403 FORBIDDEN

Status 400BAD REQUEST

/sitzung

GET

Diese Ressource wird benutzt, um an Infos für den Homescreen zu gelangen.

Es werden entweder Infos zur aktuellen Sitzung, oder die notwendigen Infos zur allgemeinen Homescreenanzeige angezeigt.

Client: Siehe Authentifizierung!

Antworten:

Status 200 OK

```
1. {
2.     "id": "32423423jok4n23oasibf23ri2b3ufb2",
3.     "raum":
4.     {
5.         //Raum-Objekt
6.     },
7.     "myTag": "true/false",
8.     "endzeit": "2016-11-21 13:44" //Unix Timestamp
9. }
```

Status 900 NO ACTIVE SESSION

```
1. {
2.     "räume":
3.     [
4.         {Raum-Objekt1},
5.         {Raum-objekt2}
6.     ],
7.     "karte":
8.     {
9.         //Karte-Objekt
10.    },
11. }
```

POST

Diese Ressource wird benutzt, wenn eine Sitzung gestartet wird.

Client: Siehe Authentifizierung!

```
1. "sitzung":  
2. {  
3.     "raum": "4555455"  
4. }  
5. }
```

Antworten:

Status 201 CREATED

Besteht bereits eine aktive Sitzung des Benutzers, wird diese überschrieben und trotzdem 201 zurückgegeben! Hier greift also nicht „insert und update“

```
1. {  
2.     "id": "4711",  
3.     "raum":  
4.     {  
5.         //Raum-Objekt  
6.     },  
7.     "myTag": "true/false",  
8.     "endzeit": "2016-11-21 13:44" //Unix timestamp  
9. }
```

/sitzung/{id}
{id}=Sitzungs-ID

PUT

Diese Ressource wird benutzt, um eine Sitzung zu verlängern

Client: Siehe Authentifizierung!

Antworten

Status 200 OK

Server setzt Endzeit auf aktuellen Zeitpunkt + SITZUNGSINTERVALL

```
1. {  
2.     "id": "4711",  
3.     "raum":  
4.     {  
5.         //Raum-Objekt  
6.     },  
7.     "myTag": "true/false",  
8.     "endzeit": "2016-11-21 13:44" //Unix timestamp  
9. }
```

400 BAD REQUEST

DELETE

Client: Siehe Authentifizierung!

Antwort

Status 200 OK

Status 400 BAD REQUEST

Status 403 FORBIDDEN

/konfiguration

GET

PUT

/karte

TODO: Konzeptionelle arbeit leisten!

Notizen für Später / Nützliches

- Für das Sortieren der Informationen / Listen kann URI-Parameter **sort** genutzt werden, dem eine kommaseparierte Liste von Attributen der Ressource mit jeweils führendem + (aufsteigende Sortierung), oder – (absteigende Sortierung) zugewiesen wird