Entwurf REST-API

Inhaltsverzeichnis

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

```
{
        //Objekt je nach verwendeter Methode
}
```

Status 400 BAD REQUEST

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

```
{
 "status":
 {
        "code":400,
        "description":"BAD REQUEST"
 }
}
```

Status 401 UNAUTHORIZED

Wird benutzt, wenn der Token ungültig ist

```
{
 "status":
 {
        "code":401,
        "description":"UNAUTHORIZED"
 }
}
```

Status 403 FORBIDDEN:

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

```
{
 "status":
 {
        "code":403,
        "description":"FORBIDDEN"
 }
}
```

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!

```
{
 "status":
 {
        "code":404,
        "description":"NOT FOUND"
 }
}
```

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!

```
{
 "status":
 {
        "code":500,
        "description":"INTERNAL SERVER ERROR"
 }
}
```

Status 501 – NOT IMPLEMENTED

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

```
{
 "status":
 {
        "code":501,
        "description":"NOT IMPLEMENTED"
 }
}
```

Status 900 – NO ACTIVE SESSION

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

```
{
 "status":
 {
       "code":900,
       "description":"NO ACTIVE SESSION"
 }
}
```

Status 910 – ROOM BLOCKED

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

```
{
 "status":
 {
       "code":910,
       "description":"ROOM BLOCKED"
 }
}
```

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 aktzeptiert 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:

```
{
        "id":"32423423jok4n23oasibf23ri2b3ufb2",
        "email":"thomas.gorgels@googlemail.com",
        "name":"Gorgels",
        "vorname":"Thomas",
        "fotoURL":"http:://google.com/profilbild"
}
```

Antworten:

Status 200 OK

Wenn der Benutzer wiederkehrend ist

```
{
        "id":"32423423jok4n23oasibf23ri2b3ufb2",
        "email":"thomas.gorgels@googlemail.com",
        "name":"Gorgels",
        "vorname":"Thomas",
        "fotoURL":"http:://google.com/profilbild",
        "token":"asdjkbaws923enadnh9q3qdn398z23e",
        "istProfessor": "false"
        "istAnonym": "false"
        "istPush" : 1

}
```

Status 201 CREATED

Wenn der Benutzer neu ist

```
{
        "id":"32423423jok4n23oasibf23ri2b3ufb2",
        "email":"thomas.gorgels@googlemail.com",
        "name":"Gorgels",
        "vorname":"Thomas",
        "fotoURL":"http:://google.com/profilbild",
        "token":"asdjkbaws923enadnh9q3qdn398z23e",
        "professor": "false"
        "istAnonym": "false"
        "istPush" : 1

}
```

PUT:

Die Ressource wird benutzt, wenn das Masterpasswort eingegeben wurde

Client:

```
{
"masterpasswort": "MatheIstToll"
"istAnonym" : 0
"istPush" : 1
}
```

Antworten:

Status 200 OK:

Masterpasswort korrekt

```
{
        "id":"32423423jok4n23oasibf23ri2b3ufb2",
        "email":"thomas.gorgels@googlemail.com",
        "name":"Gorgels",
        "vorname":"thomas",
        "foto":"http:://google.com/profilbild",
        "token":"asdjkbaws923enadnh9q3qdn398z23e",
        "istProfessor": "true"
        "istAnonym": "false"
        "istPush" : 1
}
```

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

```
[
        {
                "id": "4711",
                "name":"Präsentation"
        },
        {
                "id": "4712",
                "name":"Ruhe"
        }
]
```

POST

Die Ressource wird benutzt, um einen neuen Tag hinzuzufügen

Client:

```
{
        "name": "Gruppenarbeit"
}
```

Antworten:

Status 201CREATED

```
{
        "id":"4713",
        "name":"Gruppenarbeit"
}
```

/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

```
[
{
        "benutzer":
        {
                //Benutzer-Objekt
        },
        "status":"freund" //Boolean
        "raum":
        {
                //Raumobjekt, wenn vorhanden
        }
},
{
        "benutzer":
        {
                //Benutzer-Objekt
        },
        "status":"anfrage" //Boolean
        "raum":NULL //leer
}
]
```

POST

Die Ressource wird benutzt, um einen neuen Freund hinzuzufügen.

Client:

```
{
        email:thomas.gorgels@googlemail.com
}

```

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

```
[
{
        "id":"1234",
        "raumnummer":"G101",
        "teilnehmer_max":"32",
        "teilnehmer_aktuell":"15",
        "status":"rot/gelb/grün/grau" // enum??,
        "fotoURL":"http://<server>/raum/{id}/foto",
        "tag":
        {
                "id":"4711",
                "name":"Präsentation",
        }
},
{
        "id":"1235",
        "raumnummer":"G102",
        "teilnehmer_max":"32",
        "teilnehmer_aktuell":"17",
        "status":"rot/gelb/grün",
        "fotoURL":"http://<server>/raum/{id}/foto",
        "tag":
        {
                "id":"4711",
                "name":"Präsentation",
        }
}
]
```

/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

```
{
        "id":"1235",
        "raumnummer":"G102",
        "teilnehmer_max":"32",
        "teilnehmer_aktuell":"17",
        "status":"rot/gelb/grün",
        "fotoURL":"http://<server>/raum/{id}/foto",
        "tag":
        {
                "id":"4711",
                "name":"Präsentation",
        }
        "benutzer":
        [
        {Benutzer-Objekt1},
        {Benutzer-Objekt2}
        ] //nur nicht anonyme
}
```

PUT

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

Client: Siehe Authentifizierung!

```
{
        "tag": "4712"
}
```

Antworten:

Status 200 OK

```
{
        "raumnummer":"G101",
        "teilnehmer_max":"32",
        "teilnehmer_aktuell":"17",
        "status":"rot/gelb/grün",
        "fotoURL":"http://<server>/raum/{id}/foto",
        "tag":
        {
                "id":"4712",
                "name":"Ruhe",
        }
}
```

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

```
[
```

```
        {Veranstaltungs-Objekt},
        {Veranstaltungs-Objekt}
        ]
```

Status 403 FORBIDDEN

POST

Die Ressource wird benutzt, um eine neue Veranstaltung zu erstellen.
Client: Siehe Authentifizierung!

```
{
        "name": "hallo123"
        "von": "11.11.2016 13:00",   //UNIX Timestamp
        "bis": "11.11.2016 15:00",   //UNIX Timestamp
        "raum": "4711"
}
```

Antworten:
Status 201CREATED:

```
{
        "id":"1188",
        "name":"SWE VL"
        "dozent":
        {
                //Benutzer
        },
        "von":"11.11.2016 13:00", //UNIX Timestamp
        "bis":"11.11.2016 15:00", //UNIX Timestamp
        "raum":
        {
                //Raum der Veranstaltung
        }
}
```

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

```
{
        "id":"1188",
        "name":"SWE VL"
        "dozent":
        {
                //Benutzer
        },
        "von":"11.11.2016 13:00", //UNIX Timestamp
        "bis":"11.11.2016 15:00", //UNIX Timestamp
        "raum":
        {
                //Raum der Veranstaltung
        }
}
```

PUT

Client: Siehe Authentifizierung!

Die Ressource wird benutzt, um eine Veranstaltung zu bearbeiten

```
{
        "name":"SWE VL"
        "von":"11.11.2016 13:00", //UNIX Timestamp
        "bis":"11.11.2016 15:00", //UNIX Timestamp
        "raum":
        {
                //Raum der Veranstaltung
        }
}
```

Antwort

Status 200 OK

```
{
        "id":"1188",
        "name":"SWE VL"
        "benutzer":
        {
                //Benutzer
        },
        "von":"11.11.2016 13:00", //UNIX Timestamp
        "bis":"11.11.2016 15:00", //UNIX Timestamp
        "raum":
        {
                // Raum der Veranstaltung
        }
}
```

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

```
{
        "id":"32423423jok4n23oasibf23ri2b3ufb2",
        "raum":
        {
                //Raum-Objekt
        },
        "myTag": "true/false",
        "endzeit": "2016-11-21 13:44" //Unix Timestamp
}
```

Status 900 NO ACTIVE SESSION

```
{
        "räume":
        [
        {Raum-Objekt1},
        {Raum-objekt2}
        ],
        "karte":
        {
                //Karte-Objekt
        },
}
```

POST

Diese Ressource wird benutzt, wenn eine Sitzung gestartet wird.

Client: Siehe Authentifizierung!

```
"sitzung":
{
        "raum": "4555455"
}
}
```

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“

```
{
        "id": "4711",
        "raum":
        {
                //Raum-Objekt
        },
        "myTag": "true/false",
        "endzeit": "2016-11-21 13:44" //Unix timestamp
}
```

/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

```
{
        "id": "4711",
        "raum":
        {
                //Raum-Objekt
        },
        "myTag": "true/false",
        "endzeit": "2016-11-21 13:44" //Unix timestamp
}
```

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