Entwurf REST-Webservice

Inhaltsverzeichnis

[Einleitung 2](#_Toc467526065)

[Ressourcen 2](#_Toc467526066)

[Was muss der REST-Service potenziell bereitstellen können? 2](#_Toc467526067)

[Interaktion mit Ressourcen 2](#_Toc467526068)

[Entwurf der URIs 3](#_Toc467526069)

[Anforderungen: 3](#_Toc467526070)

[Fehlerbehandlung 3](#_Toc467526071)

[/benutzer 5](#_Toc467526072)

[POST 5](#_Toc467526073)

[/tag 6](#_Toc467526074)

[GET 6](#_Toc467526075)

[POST 6](#_Toc467526076)

[/tag/{id} 6](#_Toc467526077)

[DELETE 6](#_Toc467526078)

[/freundesliste 7](#_Toc467526079)

[GET 7](#_Toc467526080)

[/raum 8](#_Toc467526081)

[GET 8](#_Toc467526082)

[/raum/{id} 9](#_Toc467526083)

[GET 9](#_Toc467526084)

[PUT 9](#_Toc467526085)

[Notizen für Später / Nützliches 10](#_Toc467526086)

[Offene 10](#_Toc467526087)

# Einleitung

Interaktion mit dem Webservice erfolgt nur per JSON!

# 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 |
| Freundesliste | Freundeslisten |
| Freundschaftsanfrage | Freundschaftsanfragen |
| Benutzer | Benutzer |
| Sitzung | Sitzungen |
| Veranstaltung | Veranstaltungen |
| Raum | Räume |
| Tag | Tags |
| Konfiguration | Konfigurationen |

Es wurde im Team abgestimmt, dass wir für die Beschreibung der Ressourcen einheitlich das 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 <<Entscheidung>> bereitgestellt werden.

TODO: Aggregation mit einem Verlinken darstellen, oder direkt integrieren???

# Interaktion mit Ressourcen

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP-Methode | CRUD-Operation | Beschreibung | Idem potent |
| POST | CREATE |  | (NEIN) am besten |
| GET | READ |  | JA |
| PUT | UPDATE |  | JA |
| DELETE | DELETE |  | JA |

Das Einfügen von Ressourcen soll Idem potent implementiert werden! (Insert or Update)

# Entwurf der URIs

|  |  |  |
| --- | --- | --- |
| URI | HTTP-Methode | Ergebnis |
| /freundesliste | GET | Freundesliste |
|  |  |  |
| /freundschaftsanfrage | GET | Liste Freundschaftsanfragen |
| /freundschaftsanfrage | POST | Neue Freundschaftsanfrage |
| /freundschaftsanfrage/{id} | GET | Freundschaftsanfrage |
| /freundschaftsanfrage/{id} | PUT | Freundschaftsanfrage ändern |
| /freundschaftsanfrage/{id} | DELETE | Freundschaftsanfrage löschen |
|  |  |  |
| /benutzer | GET | Liste von Benutzern |
| /benutzer | POST | Neuer Benutzer / Registrieren |
|  |  |  |
| /sitzung | GET | Aktive Sitzung des Fragenden |
| /sitzung | POST | Neue Sitzung |
| /sitzung/{id} | PUT | Sitzung ändern |
| /sitzung/{id} | DELETE | Sitzung löschen |
|  |  |  |
| /veranstaltung | GET | Liste der Veranstaltungen |
| /veranstaltung | POST | Neue Veranstaltung |
| /veranstaltung/{id} | GET | Veranstaltungsdetails |
| /veranstaltung/{id} | PUT | Veranstaltung ändern |
| /veranstaltung/{id} | DELETE | Veranstaltung löschen |
|  |  |  |
| /raum | GET | Liste der Räume |
| /raum/{id} | GET | Raumdetails |
| /raum/{id} | PUT | TAG SETZEN ???? |
|  |  |  |
| /tag | GET | Liste der Tags |
| /tag | POST | Neuer Tag |
| /tag/{id} | DELETE | Tag löschen |
|  |  |  |
| /konfiguration | GET |  |
| /konfiguration | PUT | Ändern der Konfiguration |

# Anforderungen:

1. JEDE Anfrage muss die Benutzer-Identifikation und ein Token enthalten, diese Angaben werden im Body der Anfrage übergeben.
2. Der Client erhält den Token als Anwort 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!

# Fehlerbehandlung

Es sollen keine eigenen Fehlercodes generiert werden, anstatt sollen die Standart http-Fehlercodes genutzt werden!

|  |  |
| --- | --- |
| Statuscode | Fehlerbeschreibung |
| 200 | OK |
| 201 | Created |
|  |  |
| 403 | Forbidden |
| 404 | Not found |
|  |  |
| 500 | Internalt Server Error |
| 501 | Not implemented |
| … |  |

Status 404 Not found:

1. {
2. "fehlermeldung": " Die angeforderte Ressource wurde nicht gefunden",
3. }

Status 403 Forbidden:

Zum Beispiel, wenn schon ein Tag gesetzt ist!

1. {
2. "fehlermeldung": " Dazu haben sie keine Berechtigung",
3. }

## /benutzer

### POST

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "email":"thomas.gorgels@googlemail.com",
4. "name":"Gorgels",
5. "vorname":"thomas"
6. }

Antworten:

Status 201 OK - Created

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "email":"thomas.gorgels@googlemail.com",
4. "name":"Gorgels",
5. "vorname":"thomas"
6. "token":" asdjkbaws923enadnh9q3qdn398z23e "
7. }

Status XXX – Benutzer schon vorhanden

1. {
2. "fehlermeldung": " Fehler: Benutzer schon vorhanden!",
3. }

## /tag

### GET

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e "
4. }

Antworten:

Status 200 OK

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

### POST

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e ",
4. "tag":
   1. {
   2. "name": "Gruppenarbeit"
   3. }
5. }

Antworten:

Status 200 OK

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

## /tag/{id}

### DELETE

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e "
4. "tag": {"id":"4711"}
5. }

Antwort:  
Status 200 OK

STATUS 403 FORBIDDEN

Status 404 NOT FOUND

## /freundesliste

### GET

Client

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e "
4. }

Antwort

Status 200 OK

1. {
2. [
3. {
   1. "email":"thomas.gorgels@googlemail.com",
   2. "name":"Gorgels",
   3. "vorname":"thomas"
4. },
5. {
   1. "email":"merlin.dienst@gmail.com",
   2. "name":"Dienst",
   3. "vorname":"Merlin"
6. }
7. }
8. ]
9. }

## /raum

### GET

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":"asdjkbaws923enadnh9q3qdn398z23e"
4. }

Antworten:

Status 200 OK

1. [
2. {
3. "raumnummer": "G101",
4. "teilnehmer\_max":"32",
5. "teilnehmer\_aktuell": "15",
6. "status": "rot/gelb/grün",
7. "foto": "http://<server>/raum/{id}/foto",
8. "tag":{
   1. "id": "4711",
   2. "name":"Präsentation",
   3. }
9. },
10. {
11. "raumnummer": "G102",
12. "teilnehmer\_max":"32",
13. "teilnehmer\_aktuell": "17",
14. "status": "rot/gelb/grün",
15. "foto": "http://<server>/raum/{id}/foto",
16. "tag":{
    1. "id": "4711",
    2. "name":"Präsentation",
    3. }
17. }
18. …
19. ]

## /raum/{id}

### GET

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":"asdjkbaws923enadnh9q3qdn398z23e"
4. }

Antwort:

Status 200 OK

1. {
2. "raumnummer": "G102",
3. "teilnehmer\_max":"32",
4. "teilnehmer\_aktuell": "17",
5. "status": "rot/gelb/grün",
6. "foto": "http://<server>/raum/{id}/foto",
7. "tag":{
   1. "id": "4711",
   2. "name":"Präsentation",
   3. }
8. }

Status 404 Not Found: siehe oben!

### PUT

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":"asdjkbaws923enadnh9q3qdn398z23e",
4. "raumnummer": "G101",
5. "tag":"4712",
6. }

Antwort:

Status 200 OK

1. {
2. "raumnummer": "G101",
3. "teilnehmer\_max":"32",
4. "teilnehmer\_aktuell": "17",
5. "status": "rot/gelb/grün",
6. "foto": "http://<server>/raum/{id}/foto",
7. "tag":{
   1. "id": "4712",
   2. "name":"Ruhe",
   3. }
8. }

Status 403 Forbidden: siehe oben!

* Wenn ein Benutzer keine Berechtigung hat, den Tag zu setzen

Status 404 Not found: siehe oben!

* URL wurde mit einer nicht vorhanden {id} aufgerufen

## /veranstaltung/{id}

### GET

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e "
4. }

Antwort:

Status 200 OK

1. {
2. "id": "1188",
3. "professor":{},
4. "datum": "11.11.2016",
5. "von": "13:00"
6. "bis": "15:00",
7. "raum":{}
8. }

Status 404 Not FOUND

### PUT

Client

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e",
4. "veranstaltung":{
5. "id": "1188",
6. "professor":{},
7. "datum": "11.11.2016",
8. "von": "13:00"
9. "bis": "15:00",
10. "raum":{}
11. }
12. }

Status 200 OK:

1. {
2. "id": "1188",
3. "professor":{},
4. "datum": "11.11.2016",
5. "von": "13:00"
6. "bis": "15:00",
7. "raum":{}
8. }

Status 403 Forbidden

Status 404 NOT FOUND

### DELETE

Client

Client

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e",
4. "veranstaltung":{
5. "id": "1188",
6. }
7. }

Antworten:

Status 200 OK

Status 403 FORBIDDEN

Status 404 NOT FOUND

## /sitzung

## /sitzung/{id}

# 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
* Wie lösen wir das mit der Sicherheit, Jeder kann nur seine Freunde sehen etc.

# Offene