Entwurf REST-API

Inhaltsverzeichnis

[Einleitung 3](#_Toc467604253)

[Ressourcen 3](#_Toc467604254)

[Was muss der REST-Service potenziell bereitstellen können? 3](#_Toc467604255)

[Interaktion mit Ressourcen 3](#_Toc467604256)

[Fehlerbehandlung 4](#_Toc467604257)

[Entwurf der URIs 6](#_Toc467604258)

[Anforderungen: 6](#_Toc467604259)

[/benutzer 7](#_Toc467604260)

[POST 7](#_Toc467604261)

[PUT: 8](#_Toc467604262)

[/tag 9](#_Toc467604263)

[GET 9](#_Toc467604264)

[POST 9](#_Toc467604265)

[/tag/{id} 10](#_Toc467604266)

[DELETE 10](#_Toc467604267)

[/freundschaft 10](#_Toc467604268)

[GET 10](#_Toc467604269)

[POST 11](#_Toc467604270)

[/freundschaft/{id} 11](#_Toc467604271)

[PUT 11](#_Toc467604272)

[DELETE 12](#_Toc467604273)

[/raum 12](#_Toc467604274)

[GET 12](#_Toc467604275)

[/raum/{id} 13](#_Toc467604276)

[GET 13](#_Toc467604277)

[PUT 13](#_Toc467604278)

[/veranstaltung 14](#_Toc467604279)

[GET 14](#_Toc467604280)

[POST 15](#_Toc467604281)

[/veranstaltung/{id} 16](#_Toc467604282)

[GET 16](#_Toc467604283)

[PUT 16](#_Toc467604284)

[DELETE 17](#_Toc467604285)

[/sitzung 18](#_Toc467604286)

[GET 18](#_Toc467604287)

[POST 18](#_Toc467604288)

[/sitzung/{id} 19](#_Toc467604289)

[PUT 19](#_Toc467604290)

[DELETE 19](#_Toc467604291)

[/konfiguration 20](#_Toc467604292)

[GET 20](#_Toc467604293)

[PUT 20](#_Toc467604294)

[/karte 20](#_Toc467604295)

[Notizen für Später / Nützliches 20](#_Toc467604296)

# 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 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 |  | (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. "error\_code":400
3. "description":"BAD REQUEST"
4. "error\_msg": "In Ihrer Anfrage liegt ein Fehler vor.",
5. }

Status 401 UNAUTHORIZED

Wird benutzt, wenn der Token ungültig ist

1. {
2. "error\_code":401
3. "description":"UNAUTHORIZED"
4. "error\_msg": "Token nicht gültig",
5. }

Status 403 FORBIDDEN:

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

1. {
2. "error\_code":403
3. "description":"FORBIDDEN"
4. "error\_msg":"Dazu haben sie keine Berechtigung",
5. }

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 beschrieben Antwort zu senden!

1. {
2. "error\_code":404
3. "description":"NOT FOUND"
4. "error\_msg": "Die angeforderte Ressource wurde nicht gefunden",
5. }

Status 500 – INTERNAL SERVER ERROR

Wird bei Fehlern in der Verarbeitung (Bsp. Datenbank-Fehler) zurückgegeben

1. {
2. "error\_code":500
3. "description":"INTERNAL SERVER ERROR"
4. "error\_msg": "Interner Serverfehler",
5. }

Status 501 – NOT IMPLEMENTED

Wird bei nicht implementierten Methoden geliefert!

1. {
2. "error\_code":501
3. "description":"NOT IMPLEMENTED"
4. "error\_msg": "Bisher nicht implementiert",
5. }

Status 900 – NO ACTIVE SESSION

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

1. {
2. "error\_code":900
3. "description":"NO ACTIVE SESSION"
4. "error\_msg": "Keine aktive Sitzung",
5. }

Status 910 – ROOM BLOCKED

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

1. {
2. "error\_code":910
3. "description":"ROOM BLOCKED"
4. "error\_msg": "Raum bereits blockiert",
5. }

# 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, 403, 400 |
|  |  |  |  |
| /veranstaltung | GET | Liste der Veranstaltungen | 200, 403 |
| /veranstaltung | POST | Neue Veranstaltung | 201, 403, 910 |
| /veranstaltung/{id} | GET | Veranstaltungsdetails | 200, 403, 400 |
| /veranstaltung/{id} | PUT | Veranstaltung ändern | 200, 403, 400, 910 |
| /veranstaltung/{id} | DELETE | Veranstaltung löschen | 200, 403, 400 |
|  |  |  |  |
| /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 |  |

# Anforderungen:

1. JEDE Anfrage 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.

1. Der Client erhält den Token als Antwort auf seine Registrierung (POST: [/benutzer](#_POST))
2. Der Client erhält Daten nur nach seiner Benutzer-Identifikation gefiltert  
   Beispiel: Benutzer A erhält nur seine Freunde in der Freundesliste!
3. Jede Anfrage an den Server liefert auch eine Antwort.
4. Jede Antwort vom Server beinhaltet einen passenden [HTTP-Status-Code](#_Fehlerbehandlung)
5. Zeitstempel und Zeitangaben werden nur im [UNIX-Format](https://de.wikipedia.org/wiki/Unixzeit) geliefert und gesendet!

## /benutzer

### POST

Client:

1. {
2. "benutzer":
3. {
4. "id": "32423423jok4n23oasibf23ri2b3ufb2",
5. "email":"thomas.gorgels@googlemail.com",
6. "name":"Gorgels",
7. "vorname":"Thomas",
8. "foto":"http:://google.com/profilbild"
9. }
10. }

Antworten:

Status 200 OK

Wenn der Benutzer wiederkehrend ist

1. {
2. "benutzer":
3. {
4. "id": "32423423jok4n23oasibf23ri2b3ufb2",
5. "email":"thomas.gorgels@googlemail.com",
6. "name":"Gorgels",
7. "vorname":"Thomas",
8. "foto":"http:://google.com/profilbild",
9. "token":" asdjkbaws923enadnh9q3qdn398z23e",
10. "professor": "false"
11. }
12. }

Status 201 CREATED

Wenn der Benutzer neu ist

1. {
2. "benutzer":
3. {
4. "id": "32423423jok4n23oasibf23ri2b3ufb2",
5. "email":"thomas.gorgels@googlemail.com",
6. "name":"Gorgels",
7. "vorname":"Thomas",
8. "foto":"http:://google.com/profilbild",
9. "token":" asdjkbaws923enadnh9q3qdn398z23e",
10. "professor": "false"
11. }
12. }

## PUT:

Die Ressource wird benutzt, wenn das Masterpasswort eingegeben wurde

Client:

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

Antworten:

Status 200 OK:

Masterpasswort korrekt

1. {
2. "benutzer":
3. {
4. "id": "32423423jok4n23oasibf23ri2b3ufb2",
5. "email":"thomas.gorgels@googlemail.com",
6. "name":"Gorgels",
7. "vorname":"thomas",
8. "foto":"http:://google.com/profilbild",
9. "token":" asdjkbaws923enadnh9q3qdn398z23e",
10. "professor": "true"
11. }
12. }

Status 403 FORBIDDEN

Masterpasswort falsch

## /tag

### GET

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

Client:

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

Antworten:

Status 200 OK

1. {
2. "tags:
3. [
4. {
5. "id": "4711",
6. "name":"Präsentation"
7. },
8. {
9. "id": "4712",
10. "name":"Ruhe"
11. }
12. ]
13. }

### POST

Die Ressource wird benutzt, um einen neuen Tag hinzuzufügen

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e ",
4. "tag":
5. {
6. "name": "Gruppenarbeit"
7. }
8. }

Antworten:

Status 201 CREATED

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:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e "
4. "tag":
5. {
6. "id":"4711"
7. }
8. }

Antworten:  
Status 200 OK

Status 400 BAD REQUEST

## /freundschaft

### GET

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

Client:

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

Antworten:

Status 200 OK

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

## POST

Die Ressource wird benutzt, um einen neuen Freund hinzuzufügen.

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e ",
4. "freundschaft":
5. {
6. "benutzer":
7. {
8. email: [thomas.gorgels@googlemail.com](mailto:thomas.gorgels@googlemail.com)
9. }
10. }
11. }

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

Client:

{id} = Benutzeridentifikation des hinzuzufügenden Freundes

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

Antworten:

Status 200 OK

### DELETE

Die Ressource wird benutzt, um eine Freundschaft zu bestätigen

Client:

{id} = Benutzeridentifikation des zu löschenden Freundes

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

Antworten:

Status 200 –OK

## /raum

### GET

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

Client:

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

Antworten:

Status 200 OK

1. {
2. "raeume":
3. [
4. {
5. "id": "1234",
6. "raumnummer": "G101",
7. "teilnehmer\_max":"32",
8. "teilnehmer\_aktuell": "15",
9. "status": "rot/gelb/grün/grau" // enum??,
10. "foto": "http://<server>/raum/{id}/foto",
11. "tag":
12. {
13. "id": "4711",
14. "name":"Präsentation",
15. }
16. },
17. {
18. "id": "1235",
19. "raumnummer": "G102",
20. "teilnehmer\_max":"32",
21. "teilnehmer\_aktuell": "17",
22. "status": "rot/gelb/grün",
23. "foto": "http://<server>/raum/{id}/foto",
24. "tag":
25. {
26. "id": "4711",
27. "name":"Präsentation",
28. }
29. }
30. ]
31. }

## /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:

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

Antwort:

Status 200 OK

1. {
2. "raum":
3. {
4. "id": "1235",
5. "raumnummer": "G102",
6. "teilnehmer\_max":"32",
7. "teilnehmer\_aktuell": "17",
8. "status": "rot/gelb/grün",
9. "foto": "http://<server>/raum/{id}/foto",
10. "tag":
11. {
12. "id": "4711",
13. "name":"Präsentation",
14. }
15. "benutzer":
16. [
17. {Benutzer-Objekt1},
18. {Benutzer-Objekt2}
19. ] //nur nicht anonyme
20. }
21. }

### PUT

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

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":"asdjkbaws923enadnh9q3qdn398z23e",
4. "raum":
5. {
6. "tag": "4712"
7. }
8. }

Antworten:

Status 200 OK

1. {
2. "raum":
3. {
4. "raumnummer": "G101",
5. "teilnehmer\_max":"32",
6. "teilnehmer\_aktuell": "17",
7. "status": "rot/gelb/grün",
8. "foto": "http://<server>/raum/{id}/foto",
9. "tag":
10. {
11. "id": "4712",
12. "name":"Ruhe",
13. }
14. }
15. }

Status 403 FORBIDDEN

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

## /veranstaltung

### GET

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

Client:

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

Antworten:

Status 200 OK

1. {
2. "verantaltungen":
3. [
4. {Veranstaltungs-Objekt},
5. {Veranstaltungs-Objekt}
6. ]
7. }

Status 403 FORBIDDEN

### POST

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

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e ",
4. "veranstaltung": {
5. "datum": "11.11.2016",
6. "von": "13:00"
7. "bis": "15:00",
8. "raum": "4711"
9. }
10. }

Antworten:

Status 201 CREATED:

1. "veranstaltung":{
2. "id": "1188",
3. "benutzer":{Benutzer-Objekt}, //Professor
4. "datum": "11.11.2016",
5. "von": "13:00"
6. "bis": "15:00",
7. "raum":{} // Raum der Veranstaltung
8. }

Status 403 FORBIDDEN

Status 910 ROOM BLOCKED

## /veranstaltung/{id}

{id}=ID der Veranstaltung

### GET

Die Ressource wird benutzt, um ???

Client:

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

Antworten:

Status 200 OK

1. {
2. "veranstaltung":
3. {
4. "id": "1188",
5. "benutzer":
6. {
7. //Professor
8. },
9. "datum": "11.11.2016",
10. "von": "13:00"
11. "bis": "15:00",
12. "raum":
13. {
14. // Raum der Veranstaltung
15. }
16. }
17. }

### PUT

Client

Die Ressource wird benutzt, um eine Veranstaltung zu bearbeiten

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

Antwort

Status 200 OK

1. {
2. "veranstaltung":
3. {
4. "id": "1188",
5. "benutzer":
6. {
7. //Professor
8. },
9. "datum": "11.11.2016",
10. "von": "13:00"
11. "bis": "15:00",
12. "raum":
13. {
14. // Raum der Veranstaltung
15. }
16. }
17. }

Status 403 FORBIDDEN

Status 910 ROOM BLOCKED

### DELETE

Diese Ressource wird benutzt, um eine Veranstaltung zu löschen

Client

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

Antworten:

Status 200 OK

Status 403 FORBIDDEN

Status 400 BAD 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

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

Antworten:

Status 200 OK

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

Status 900 NO ACTIVE SESSION

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

### POST

Diese Ressource wird benutzt, wenn eine Sitzung gestartet wird.

Client:

1. {
2. "id": "32423423jok4n23oasibf23ri2b3ufb2",
3. "token":" asdjkbaws923enadnh9q3qdn398z23e",
4. "sitzung":
5. {
6. "raum": "4555455"
7. },
8. }

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. "sitzung":
3. {
4. "id": "4711",
5. "raum":
6. {
7. Raum-Objekt
8. },
9. "myTag": "true/false",
10. "endzeit": "2016-11-21 13:44" //Unix timestamp
11. }
12. }

## /sitzung/{id}

{id}=Sitzungs-ID

### PUT

Diese Ressource wird benutzt, um eine Sitzung zu verlängern

Client:

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

Antworten

Status 200 OK

Server setzt Endzeit auf aktuellen Zeitpunkt + SITZUNGSINTERVALL

1. {
2. "sitzung":
3. {
4. "id": "32423423jok4n23oasibf23ri2b3ufb2",
5. "raum":
6. {
7. Raum-Objekt
8. },
9. "myTag": "true/false",
10. "endzeit": "2016-11-21 13:44" //Unix timestamp
11. }
12. }

400 BAD REQUEST

### DELETE

Client

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

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