Webservice Definition

URL: <http://192.168.43.142:8080/SportVerwaltung_WebServices/webresources>

Im Token wird immer der Account mit dem der User gerade Angemeldet ist mitgegeben. Für alle Methoden muss der User angemeldet sein außer für login, logout, verify and recover.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Url | Methode | Input | Output | Beschreibung | SQL-Anweisung | Angewendet in Activity | Fertig am |
| /login  (Post) | login | String Content (Account) | Response (Status) | die Datenbank wird durchsucht ob der User existiert und wenn ja dann wird er zurückgegeben | SELECT id, email, password FROM account WHERE email = ? and password = ? |  |  |
| /logout  (Post) | logout | String token | Response (Status) | Ein User der Eingeloggt ist und wird hier mittels seinem token ausgeloggt. | Token aus der Liste der Angemeldeten User in der Klasse Authentification löschen. |  |  |
| /verify  (Get) | verifyAccount | String accountToken | Response (temoraryRedirect) | Jeder der sich registriert muss sich auch verifizieren. | UPDATE account SET isVerified = ? where email = ? |  |  |
| /recover  (Post) | recoverAccount | String content (Account) | Response (Status) | Wenn der User sein Passwort vergessen hat wird ihm eine Email mit seinem Passwort geschickt. | SELECT \* FROM account WHERE id = ? |  |  |
| /account  (Get) | getAccount | String token | Response (Account, Status) | Egal ob der User ein Teilnehmer oder Veranstalter ist er bekommt immer einen Account. | SELECT \* FROM account WHERE id = ? |  |  |
| /account  (Put) | updateAccount | String token | Response (Status) | Der User kann seine Daten, welche im Account gespeichert werden verändern. | UPDATE account SET name = ?, password = ? where email = ? |  |  |
| /teilnehmer  (Get) | getTeilnehmer | String token | Response (Teilnehmer) | Gibt die vollständigen Daten des Teilnehmers zurück. | SELECT \* FROM account join teilnehmer on teilnehmer.id\_account = account.id WHERE account.email = ? |  |  |
| teilnehmer  (Post) | registerTeilnehmer | String content (Account) | Response (Status) | Legt einen neuen Account an und einen neuen Teilnehmer | INSERT INTO account (id, name, email, password, isVerified) VALUES(?,?,?,?,?);  INSERT INTO teilnehmer (id\_account) VALUES(?);  insert into Teilnehmer values (actualID, 0);  insert into Veranstalter values (actualID); |  |  |
| /teilnehmer  (Put) | updateScore | String token, String content (Account) | Response (Status) | Der Score des Teilnehmers wird geupdated. | UPDATE teilnehmer SET score = ? where id\_account = ? |  |  |
| /veranstalter  (Get) | getVeranstalter | String token | Response (Account) | Die Daten eines Teilnehmers werden benötigt.. | SELECT \* FROM account join veranstalter on veranstalter.id\_account = account.id WHERE account.email = ? |  |  |
| /veranstalter  (Post) | registerVeranstalter | String content | Response (Status) | Ein neuer Veranstalter wird registriert. | select seq\_account\_id.nextval from dual  INSERT INTO account (id, name, email, password, isVerified) VALUES(?,?,?,?,?)  INSERT INTO veranstalter VALUES(?) |  |  |
| /events  (Get) | getEvents | String token, String filter | Response (List<Event>) | Gibt die Events die es gibt laut Filter zurück. | Datum: SELECT + EVENT\_COLUMNS + FROM veranstaltung inner join account on veranstaltung.id\_veranstalter = account inner join teilnahme on teilnahme.id\_veranstaltung = veranstaltung.id where datetime {comp} ? and id\_teilnehmer = ? order by veranstaltung\_datetime  In Future: SELECT + EVENT\_COLUMNS + FROM veranstaltung inner join account on veranstaltung.id\_veranstalter = account.id where datetime > ? order by veranstaltung\_datetime  Veranstalter: SELECT + EVENT\_COLUMNS + FROM veranstaltung inner join account on veranstaltung.id\_veranstalter = account.id inner join veranstalter on veranstalter.id\_account = account.id where account.id = ?; |  |  |
| /events  (Post) | addEvent | String token, String content (Event) | Response (Status) | erstellt ein neues Event an. | INSERT INTO veranstaltung VALUES(seq\_veranstaltung\_id.nextVal, ?, ?, ?, ?, ?, ?, ?, ?) |  |  |
| /events  (Put) | updateEvent | String token, String content (event) | Response (Status) | Ändert die Daten eines bereits vorhandenen Events | UPDATE veranstaltung SET name = ?, datetime = ?, details = ?, max\_teilnehmer = ?, min\_teilnehmer = ?, sportart = ? where id = ? |  |  |
| /events (Delete) | deleteEvent | String token, int eventid | Response (Status) | Löscht die Teilnahmen an einem Event und dann das Event selbst | select \* from veranstaltung where id = ?  delete from teilnahme where id\_veranstaltung = ?  delete from veranstaltung where id = ? |  |  |
|  |  |  |  | s |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |