API-Dokumentation der Nutzerverwaltung

Ein Request an den Server wird entweder per POST oder per GET gesendet.
Er ist immer der Form:

```
action=.....
...
```

Per GET werden Informationen abgefragt, per POST werden Informationen gesendet und Aktionen ausgeführt.

Der Server liefert einen JSON String zurück, der immer mindestens die Variable `status` enthält, in welcher entweder `true` oder `false` steht. Im Fehlerfall enthält die Rückgabe noch eine Variable `error` mit einer Fehlermeldung. Je nach Anfrage sind weitere Variablen enthalten.

Beispiel

GET-Anfragen

action=get_data

Liest die in der Datenbank gespeicherten daten eines Nutzers aus.

Variablen

- `username`: Nutzer, dessen Daten geholt werden sollen. Wenn leer werden die Daten des aktuell eingeloggten Benutzers zurückgegeben

Rückgabe

`{"action": "get_data", "status": true, "data": "Hier stehen Daten"}`

Berechtigung

Benutzer der Rolle `admin` können die Daten jedes Nutzers auslesen.
Ansonsten hat man nur Zugriff auf seine eigenen Daten

action=get_login_data

Gibt das JSON-Unterobjekt `login` aus den Daten eines Nutzers zurück.

Variablen

- `username`

Rückgabe

`{"action": "get_login_data", "status": true, "data": "Hier stehen Daten"}`

Berechtigung Nur Benutzer der Rollen `admin` und `evaluation` können diese Funktion benutzen.

action=check_user

Überprüft, ob ein Nutzer schon existiert.

Variablen

- `username`: Nutzer, dessen Existenz geprüft werden soll.

Rückgabe

`{"action": "check_user", "status": true, "user_exists": true/false}`

action=get_role

Gibt die Rolle des aktuell eingeloggten Benutzers zurück. Mögliche Rollen sind `admin`, `user` und `proofreader`

Variablen

- `username`: Nutzer, dessen Rolle geholt werden soll.

Rückgabe

```
{"action": "get_role", "status": true, "username": "aktueller
Nutzername", "role": "aktuelle Rolle"}
```

Berechtigung

Jeder Nutzer kann seine eigene Rolle abfragen. Admins können die Rolle aller Nutzer abfragen.

action=get_username

Gibt den Nutzernamen des aktuell eingeloggten Benutzers zurück.

Variablen

- keine

Rückgabe

```
{"action": "get_username", "status": true, "username": "aktueller
Nutzername"}
```

POST-Anfragen

action=login

Variablen

- `username`
- `password`

Rückgabe

```
{"action": "login", "status": true, "username": "eingeloggter
Nutzername", "role": "Rolle des eingeloggten Nutzers"}
```

action=add_user

Variablen

- `username`
- `password`
- `role`: Rolle des neu angelegten Benutzers (optional, standard ist `user`)

Rückgabe

```
{"action": "add_user", "status": true, "username": "name des
angelegten Nutzers", "role": "Rolle des angelegten Nutzers"}
```

action=del_user

Löscht einen Nutzer.

Variablen

- `username`

Rückgabe

```
{"action": "add_user", "status": true}
```

action=change_pwd

Ändert das Passwort eines Nutzers.

Variablen

- `username`

- `old_password`: Das aktuelle Passwort
- `password`: Das neue Passwort

Rückgabe

```
{"action": "change_pwd", "status": true}
```

Berechtigung

Ein normaler Nutzer kann immer nur sein eigenes Passwort ändern. Admins können das Passwort jedes Nutzers ändern, ohne das alte Passwort zu kennen.

action=change_role

Ändert die Rolle eines Nutzers.

Variablen

- `username`
- `role`

Rückgabe

```
{"action": "change_role", "status": true}
```

Berechtigung

Nur Admins können die Rolle eines Nutzers ändern.

action=write_data

Schreibt dem Nutzer zugeordnete Daten in die Datenbank.

Variablen

- `username`
- `data`: String, der in der Datenbank gespeichert werden soll. Der vorherige Datenbankeintrag wird überschrieben.

Rückgabe

```
{"action": "write_data", "status": true}
```

Berechtigung

Normale Nutzer dürfen ihre eigenen Daten überschreiben. Admins haben Schreibzugriff auf die Daten aller Nutzer.

action=logout

Schreibt dem Nutzer zugeordnete Daten in die Datenbank.

Variablen

- keine

Rückgabe

```
{"action": "logout", "status": true}
```