Manual for Rest-Server

Johann Haag, Tarik Hosic und Simon Blaha

20.12.2018, Leonding

Project Name	Smart Organizer
Project Leader	Simon Blaha
Version	1.0
Document state	In process

Contents

1 Installation

1.1 Aufsetzen der MongoDB

Eine MongoDB kann standardmäßig auf den Computer installiert werden. Es bietet sich auch die Möglichkeit an, die MongoDB per Docker zu betreiben. Wenn sie die MongoDB auf den Rechner installieren möchten, sehen sie auf folgender Seite nach `https://www.mongodb.com`. Falls sie Docker verwenden möchten, dann sehen sie auf folgender Seite nach `https://hub.docker.com/_/mongo`.

1.2 Installation der benötigten Pakete für den Rest-Server

Wechseln Sie in das Root-Verzeichnis des Rest-Servers und geben sie im Terminal npm i ein. Somit werden alle benötigten Pakete installiert. Informationen zu npm finden Sie auf `https://www.npmjs.com/get-npm`.

2 Konfiguration

In der /database/conifg.json Datei des Servers können Datenbankeinstellungen des Servers geändert werden. Hier kann unter dbURI eine andere URI angegeben werden, falls man keinen Localhost verwendet. Bei der Standardinstallation ist immer Localhost und der standard Port für MongoDB eingetragen. Des Weiteren ist es auch möglich, die Fehlerausgaben zu ändern.

3 Inbetriebnahme des Rest-Servers

Um den Server zu starten gibt es mehrere Wege. Zum einen können sie per CMD in das Root-Verzeichnis des Servers wechseln und node index.js ausführen oder Sie führen das Script run aus. Hierbei wird jedoch die Skriptsprache Bash benötigt. Bei der Inbetriebnahme werden automatisch die benötigten Collections auf der MongoDB erstellt. Falls alles korrekt gestartet wird, wird ausgegeben, dass der Server auf dem Port 8080 lauscht und dass eine Verbindung zur MongoDB hergestellt wurde. Der Server kann einfach mit STRG+C im Terminal beendet werden.

4 Verwendung des Servers

Es wird davon ausgegangen, dass der Server auf Localhost läuft. Falls etwas schiefläuft, wird immer ein JSON-Objekt mit den Attributen error und code zurückgegeben. Code besitzt den Wert des Fehlercodes und im Attribut error steht die Fehlermeldung. Wenn eine Abfrage erfolgreich ist, so wird ein JSON-Objekt mit einem Attribut result zurückgegeben.

4.1 User registrieren

Registriert einen neuen User auf dem Server. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/user/register`. Struktur des JSON-Objekts: {"username":"<username>", "password":"<password>"}

4.2 User login

Einloggen eines Users auf dem Server. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/user/login`. Struktur des JSON-Objekts: {"username":"<username>", "password":"<password>"} Bei erfolgreichen Login wird ein Token zurückgegeben.

4.3 Logout

Ausloggen eines Users auf dem Server. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/user/logout`. Struktur des JSON-Objekts: {"username":"<username>", "token":"<token>"}

4.4 Check if user exists

Überprüfen, ob ein Username bereits vergeben ist. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/user/check`. Struktur des JSON-Objekts: {"username":"<username>"}

4.5 Create Appointment

Erstellen eines Termins für einen User. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/appoitnment/create`. Struktur des JSON-Objects: "appointment": {"username": "<username>", "date": "<date>", "duration": "<number>", "name": "<name>", "token": "<token>"}

4.6 Delete Appointment

Löschen eines Termines eines Users. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/appoitnment/delete`. Struktur des JSON-Objects: {"username": "<username>", "token": "<token>", "appointmentid": "<appointmentid>" }

4.7 Edit Appointment

Editieren eines Termins eines Users. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/appoitnment/delete`. Struktur des JSON-Objects: "appointment": "username": "<username>"[, "date": "<date>"][, "duration": "<number>"][, "name": "<name>"] Die in [] geschriebenen Attribute sind optional!

4.8 Get all appointments

Alle Termine abfragen eines Users. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet http://localhost:8080/appoitnment/get. Struktur des JSON-Objects: "username": "<username>", "token": "<token>"

4.9 Create group

Eine Gruppe erstellen. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/group/create`. Struktur des JSON-Objects: "group": "owner": "<username>", "name": "<groupName>" , "token": "<token>"

4.10 Delete group

Eine Gruppe erstellen. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/group/create`. Struktur des JSON-Objects: "group": "owner": "<username>", "name": "<groupName>" , "token": "<token>"

4.11 Get groups

Alle Gruppen bekommen für den der User der Besitzer ist. Hierfür muss ein JSON-Objekt per Post-Request an den Server geschickt werden. Der Pfad für den Post-Request lautet `http://localhost:8080/group/create`. Struktur des JSON-Objects: {"username": "<username>", "token": "<token>"}