Ali Karami (80018), Yannick Söll (80060)

IBS Projekt SS 2021

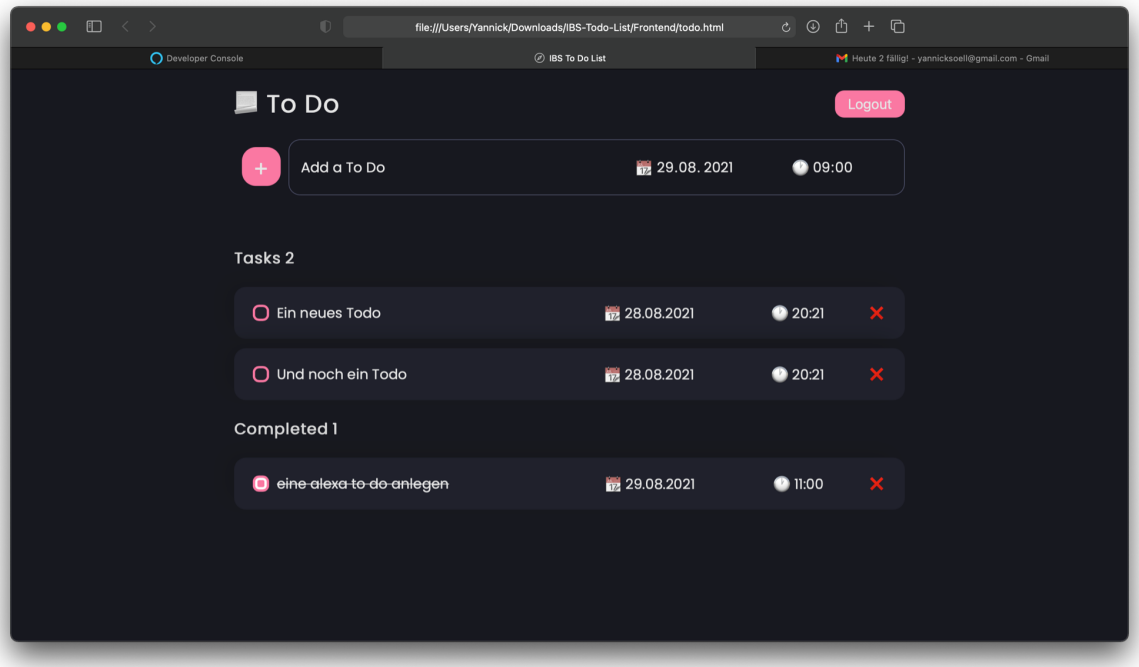
Projektbeschreibung

In unserem Projekt haben wir ein IBS in Form einer Todo-Liste erstellt. Auf diese kann sowohl über eine Web-Interface als auch über die Amazon Alexa zugegriffen werden.

Über das Web-Interface kann zunächst einmal ein Account erstellt werden. Hierfür sind eine email-Adresse sowie ein Passwort erforderlich. Mit diesem kann man sich dann in Zukunft wieder anmelden, falls man sich irgendwann einmal abgemeldet hat.

Wenn man dann angemeldet ist, können neue Todos zur Todo Liste hinzugefügt werden. Diese haben einen Titel/Inhalt sowie einen Fälligkeitszeitpunkt, welcher aus Datum und Uhrzeit besteht.

Todos auf der Liste können nun durch Anklicken als erledigt markiert werden, dadurch wandern sie in die Liste der erledigten Todos, die sich unter der Liste der noch zu erledigenden Todos befindet. Erledigte Todos werden zusätzlich durch einen durchgestrichenen Titel erkenntlich gemacht und können durch erneutes Anklicken wieder als unerledigt markiert werden, wodurch sie wieder in die obere Liste wandern.
Todos beider Listen können durch Klicken auf das X-Symbol am rechten Rand des Todos gelöscht werden.



Neben dem Web-Interface besteht zudem die Möglichkeit, die Todo-Liste mit der Amazon Alexa zu verbinden. Hierfür muss bereits ein Account über das Web-Interface angelegt worden sein, um das für Alexa erforderliche Account-Linking durchzuführen.

Nachdem man das erledigt hat, lässt sich der Skill mit dem Befehl “Öffne Internetbasierte Systeme Todo List” starten. Nun können über die am aktuellen Tag fälligen Todos vorgelesen oder neue Todos hinzugefügt werden.

Um sicher zu stellen, dass keine Aufgaben vergessen werden, werden zudem per Email-Benachrichtigungen für die am aktuellen Tag fälligen Todos versendet.

Technologie

Frontend

Für das Frontend (Web-Interface) der Todo List haben wir HTML, CSS und JavaScript verwendet. Das HTML definiert hierbei die Struktur unseres Web-Interfaces, das Styling wird über das CSS definiert.

Im JavaScript (Datei app.js) findet sich dann der eigentliche Hauptteil des Frontends. Hier wird die komplette Darstellung der Todos durch dynamische DOM-Manipulation erledigt sowie deren Speicherung, Änderung oder Löschung von Todos im Backend veranlasst.

Das JavaScript kommuniziert mit Hilfe von Axios über unsere REST API mit dem Backend. Axios ist eine promise-basierte HTTP-Client-Bibliothek um eine HTTP-Anfrage zu senden und ihre Antwort zu verarbeiten und stellt eine Alternative zu JavaScript eigenen fetch()-Funktion dar.

Zudem haben wir für das Frontend auch die mqttws31.js-Bibliothek sowie eine leicht angepasste Version der Konfiguration (mqtt-fetch.js) von Testat4 verwendet, um über MQTT die Realtime-Darstellung eines über Alexa erstellten Todos im Web-Interface zu ermöglichen.

Backend

Das Backend wird mit Hilfe von NodeJS und Express erstellt und stellt seine Funktionen über eine REST-API für das Frontend bereit. Sowohl Todos als auch Benutzer werden in einer MongoDB-Datenbank gespeichert, welche über Mongo-DB-Atlas gehostet wird.

Im Zuge dessen haben wir Mongoose verwendet, um Mongo-DB Backend-intern schemabaisert verwenden zu können. Die Schemata hierfür befinden sich im Models-Ordner.

Zudem befinden sich im Backend zwei MQTT-Verbindungen: Eine mit unserem lokalen MQTT-Server auf der Internetbasierten-Systeme-VM, über welche Frontend und Backend

verbunden sind, die zweite mit dem Ostalbradar-Server, um die Verbindung zwischen Backend und Alexa zu ermöglichen.

Um eine ordentliche Struktur zu erreichen, haben wir einen zwischen Routes, Controllern und Middleware unterschieden:

Im Route-Ordner finden sich lediglich die Definitionen der API-Adressen.
Der Controller-Ordner beinhaltet nun die zu den jeweiligen gehörende Logik.
Die Middleware stellt eine Funktion zum Schützen gewisser Routes (z.B. die Todo-Routes) gegen unautorisierten Zugriff bereit: Sie überprüft, ob ein korrekter auth-token (jsonwebtoken) im auth-header vorhanden ist und stellt nur in diesem Fall die geforderte Funktion zu Verfügung, ansonsten wird der Fehlercode 401 (unauthorized) sowie eine Fehlermeldung zurückgegeben.

Zuzüglich zu den oben genannten haben wir drei weitere Bibliotheken verwendet: bcrypt, memory-cache und jsonwebtoken.

Bcrypt ist eine sehr sichere Methode zum Hashing der Passwörter (nur die Hashes werden in der MongoDB gespeichert).

Memory-Cache ermöglicht das Speichern der User-Session bei der Verwendung von Alexa. Die User-Session besteht aus dem aktuellen Zustand des Zustandsautomaten, der User-Id und der Session-Id. Dadurch wird ermöglicht, dass viele User gleichzeitig den Alexa-Skill benutzen.

Jsonwebtoken wird zur Verwendung eines solchen als Access-Token benötigt. Bei Erfolgreicher Anmeldung des Users wird mit Hilfe der User-Id eines geheimen Schlüssels ein solches Token erstellt, was dann als Response auf den Login-Request an das Frontend gesendet und dort im Locale-Storage des Browsers gespeichert wird. Dies reicht bei allen weiteren Anfragen des Users zu vollständigen Authentifizierung und ermöglicht so eine Zustandslose Backend-Architektur, in der keine User-Sitzungen gespeichert werden müssen.

Alexa

Der Alexa Skill ist wie bereits erwähnt mithilfe des Ostalbradar-Servers mit dem Backend verbunden. Die für Alexa benötigten Dateien finden sich im Backend-Ordner unter Alexa.

Für Alexa ist Account-Linking erforderlich, da jeder User seinen eigenen Account mit seiner eigenen Todo-Liste hat. Dies ist mit Hilfe von Implicit Grant umgesetzt. Hierbei wird ein Access-Token erstellt, mithilfe dessen die Alexa des Users sich dann bei unserem Backend im Namen des Users authentifizieren und dessen Todo-Liste verwalten kann.

Die Login-Seite für das Account-Linking wird per PUG generiert. Um sie Alexa zu Verfügung zu stellen, haben wir Ngrok verwendet.

Email

Für die Email-Funktion haben wir das Paket nodemailer verwendet und zudem einen eigenen Google-Account (ibstodo@gmail.com) erstellt, über den die Mails versendet werden.

Hiermit werden Emails bei jedem Neustart des Servers oder ab dann in 24-Stunden-Abschnitten versendet, falls ein User am jeweiligen Tag ein oder mehrere fällige Todos hat.