Installation

Folgende Tools oder Alternativen sollten bereits installiert sein:

Für die Verbindung zum Server über SSH: **Cyberduck** https://cyberduck.io/download/

Zum Bearbeiten des Source-Codes, und die integrierte Konsole: **VS Code** https://code.visualstudio.com/

Für Version-Control : **Git** for windows: https://git-scm.com/downloads/win

Benötigt **NodeJS** und **npm**: https://nodejs.org/en

Hinweis: Dieses Projekt wurde mit einem Fokus auf Funktionalität und nicht auf maximale Sicherheit entwickelt. Aufgrund begrenzter Zeitressourcen wurden nur grundlegende Sicherheitsmaßnahmen implementiert – beispielsweise ist die API für Einstellungen derzeit nicht geschützt.

Der gesamte Quellcode steht zur freien Verfügung und kann nach Belieben genutzt, verändert oder erweitert werden. Bitte berücksichtige das bei der Verwendung des Codes, insbesondere in produktiven Umgebungen.

Vorbereitung

Step 1: Neuen Uberspace-Asteroid anlegen

- Erstelle einen neuen Account auf https://uberspace.de/de/ oder nutze einen bestehenden.
- Lege einen neuen Asteroid an. Der hier verwendete Name wird auch später in der Domain und als Login- und Datenbankname verwendet.

Step 2: Verbindung über Cyberduck (SSH/SFTP) herstellen

Installiere Cyberduck (https://cyberduck.io/download/)

- Cyberduck öffnen.
- Auf “Neue Verbindung” klicken.
- SFTP (SSH Verbindung) auswählen.
- Folgende Daten eintragen:
  - **Server (Hostname):** Dieser steht auf dem Datenblatt und hat die Form: `columba.uberspace.de`
  - **Username:** Dieser steht ebenfalls auf dem Datenblatt und stimmt mit dem Namen des Asteroids überein.
  - **Passwort:** Dies muss selbst vergeben werden über den Punkt “Zugänge”.

Step 3: Repository klonen und Abhängigkeiten installieren

- Öffne eine Konsole im Zielordner und clone das Repository:

```
git clone https://github.com/Maximan410/MotivierendeToDoListen.git
```

Backend einrichten

Step 4: Backend-Verzeichnis und Umgebungsvariablen konfigurieren

- Erstelle einen neuen Ordner mit Namen “webserver” für das Backend. Dieser sollte im Verzeichnis /home/username/ liegen.
- Sende einen Befehl über Cyberduck zum Auslesen des Datenbankpassworts:
  - Das Fenster lässt sich über den Menüpunkt “Gehe zu” -> “Befehl senden” öffnen.

```
my_print_defaults client
```

2. Erstelle eine `.env`-Datei mit folgendem Inhalt:

```
   MYSQL_HOST=localhost
MYSQL_USER=<USERNAME>
MYSQL_PASSWORD=<READPASSWORD>
MYSQL_PORT=
MYSQL_DB=<DBUSERNAME>
EMAIL_HOST=<'smtp.zoho.eu'>
EMAIL_PORT=<587>
EMAIL_SECURE='false'
EMAIL_USER=<'example@zohomail.eu'>
EMAIL_PASS=<'EMAILPASSWORD'>
```

Um die Erinnerungsfunktion zu nutzen benötigst du einen Account bei einem E-Mailprovider deiner Wahl. Der E-Mailprovider für den wir uns entschieden haben, war Zoho. Dieser erlaubt automatisierten E-Mail Zugriff für das Versenden von E-Mails, was die Einrichtung und Nutzung relativ einfach gestaltet (**Hinweis**: E-Mailprovider wie z.B. Gmail sind für reale User ausgerichtet und nicht ohne weitere Schritte für das Nodemailer-Module verwendbar). In der .env musst du nur noch die EMAIL_HOST an den jeweiligen Provider deiner Wahl anpassen und unter EMAIL_USER und EMAIL_PASS die Logindaten eingeben.

Step 5: Abhängigkeiten und Services konfigurieren

1.Lade die Dateien .env, main.js, packe-lock.json und package.json aus dem Ordner “server” des Repositorys über Cyberduck in den Ordner “webserver”.

- Öffne erneut das Befehlfenster und sende den Befehl

  ```
  npm install --build-from-source --prefix ~/webserver
  ```

  um die nötigen Dependencies zu installieren. Das nimmt kurze Zeit in Anspruch.

2. Öffne die `backend.ini`-Datei und ändere den Pfad (directory=/home/USERNAME/webserver) mit dem von dir gewählten Namen und lade sie in den Ordner `~/etc/services.d/` hoch.

- Dienste neu laden und starten:

  ```
  supervisorctl reread
  supervisorctl update
  ```

—

Backend erreichbar machen

Step 6: API-URL auf Uberspace setzen

1. Standardmäßig soll das Backend unter folgender URL erreichbar sein:

   ```
   https://username.uber.space/api
   ```

   Falls ein anderer URL-Suffix gewünscht ist, muss dieser angepasst werden:
2. Um die API über Uberspace zu registrieren, um den Zugriff zu gewährleisten, muss folgender Befehl ausgeführt werden:

   ```
   uberspace web backend set /api --http --port 5000 --remove-prefix
   ```

—

Frontend deployen

Step 7: API-URL im Frontend anpassen

- In `client/src/App.tsx` die API-URL setzen mit dem gewählten Benutzernamen:

```
export const API_URL = "https://username.uber.space/api";
```

Step 8: Build-Prozess und Deployment

1. Ins Client-Verzeichnis wechseln und die Webseite builden:

```
cd client
npm install
npm run build
```

2. Lösche die alten Dateien im `html`-Verzeichnis deines Asteroids und lade die Dateien aus dem Ordner "client/dist" hoch.

Nutzer registrieren und devtools nutzen

Um Nutzer registrieren zu können, navigiere in den Ordner "devtools" des Repositorys. Ändere auch in `devtools/src/App.tsx` die API-URL mit dem gewählten Benutzernamen:

```
export const API_URL = "https://username.uber.space/api";
```

Führe folgende Befehle aus :

```
npm install
npm run dev
```

Dies startet lokal eine Seite mit verschiedenen Tools.

- **Register:**
  - Hier können neue Nutzer mit einem Namen und einem Passwort registriert werden.
- **Show Users:**
  - Dies zeigt bereits registrierte Nutzer an, sowie die Anzahl, wie oft sich diese Nutzer an bestimmten Tagen eingeloggt haben.
- **Manage Rewards:**
  - Bietet die Möglichkeit die Anzahl an Pixeln zu setzen, die es für eine erledigte Aufgabe gibt.
  - Unter dem Punkt "Update Reward" wird die Setting ID als die Anzahl der erledigten Aufgabe interpretiert und die Setting Value als die Anzahl der Pixel, die ein Nutzer bekommt, sollte er diese Menge and Aufgaben erledigt haben.
  - Unter dem Punkt update mode können zwischen den Modi 0, 1 und einem größeren Wert gewechselt werden. ´Hierbei ist 0 die Standardeinstellung und schaltet nur die Basis-ToDo-Listen App frei, 1 schaltet die Erweiterung durch die Pixelwall frei und größere Werte blockieren den standardmäßigen Zugang für alle Nutzer, ohne diese löschen zu müssen.
- **Block user:**
  - Dies erlaubt es einzelne User zu blockieren ohne diese löschen zu müssen.
    - Hierführ ist die userId notwendig. Diese kann aus der Datenbank ausgelesen werden.
    - Die ID-Zuweisung beginnt bei 1 und wird für jeden neuen User einfach erhöht.
- **GodModePwall:**
  - Erlaubt es so viele Pixel auf der Pixelwall zu setzen wie gewünscht ohne diese vorher verdienen zu müssen.
- **SpeedViewPwall:**
  - Erlaubt es den Verlauf der gesetzten Pixel zu verfolgen.