Handout, 30. Mai 2019

Web-Dashboard

Marvin Gaube[1], Sidney Kuyateh[1], Steffen Walter[1]

Zusammenfassung
Die vorliegende Dokumentation gibt einen kurzen Einblick in das Projekt ”Web-Dashboard” geben, und dabei insbesondere auf folgende Architekturelemente eingehen:

- Allgemeine Struktur
- Anbindung der Openweathermap-API
- Anbindung der Wikipedia-API und der Watson TTS-API
- Einbindung eines RSS-Feeds
- Einbindung eines Mastodon-Feeds inklusive Authentifizierung ggü. der Mastodon-Instanz
- Einbindung der VVS-API

[1] *Studiengang Informationstechnik, Fakultät Technik, Duale Schule Baden-Württemberg, Stuttgart*

Inhaltsverzeichnis

Einleitung 1
1 Grundlegende Architektur 1
2 Aufbau der Oberfläche 1
3 Backend 1
4 Integration von Mastodon 1
4.1 Authentifizierung 1
4.2 Anbindung der Timeline 2
4.3 Schwierigkeiten 2

Einleitung

Aufgabenstellung war, mittels Webtechnologien ein Portal zu entwickeln, dass einige dynamische/interaktive Funktionalitäten - insbesondere unter Einbindung externer Schnittstellen - mit einbindet.

1. Grundlegende Architektur

Das Projekt ist in zwei Module aufgeteilt: **Frontend** und **Backend**.
Das Frontend ist mit HTML, CSS und JavaScript realisiert und läuft komplett Clientseitig, also im Browser. Teilweise kommt das Framework ”Vueßum Einsatz.
Das Backend ist ebenfalls in JavaScript mithilfe der Laufzeitumgebung Node.JS realisiert, und übernimmt die Auslieferung des Frontendes, sowie Authentifizierungs- und Proxyaufgaben. Für die HTTP-Schnittstelle kommt das Node-Modul Express zum Einsatz.

2. Aufbau der Oberfläche

3. Backend

Das Backend ist mittels express.js aufgebaut. Für die Authentifizierung gegenüber Mastodon-Instanzen kommt noch eine Dateibasierte SQLITE-Datenbank mit der Abstraktionsschicht SSequelizeßum Einsatz, die Applikations-ID und Secret speichert. Weiterhin ist der HTTP-POST-Endpunkt /getTTS”vorhanden, der einen Proxy zur IBM-Watson-TTS-API bildet und die Authentifizierung übernimmt.
Die Konfiguration - insbesondere die der API-Schlüssel - erfolgt über Umgebungsvariablen, die mittels dem Modul ”dotenväus dem lokalen .env-File geladen werden. Damit wird eine Trennung von Code/Applikation und Konfigurationsdaten erreicht - eine Beispieldatei für .env (.exampleenv) ist jedoch weiterhin vorhanden.

4. Integration von Mastodon

Mastodon ist ein Twitterähnliches, verteiltes soziales Netzwerk. Verteilt bedeutet, dass es beliebig viele Instanzen gibt, deren Nutzer über das sogenannte ÄctivityPubProtocol miteinander interagieren können. Ziel der Implementierung ist es, dem Nutzer seine persönliche Timeline (die letzten Posts) in einer Kachel anzuzeigen.

4.1 Authentifizierung

Um die nichtöffentlichen Teile der Mastodon-REST-API nutzen zu können, muss sich der Nutzer dem Server gegenüber authentifizieren. Eine zusätzliche Schwierigkeit ist, dass dies nicht gegenüber einem festen Endpunkt erfolgt, sondern davon abhängt, bei welchem Server (auch: Instanz) der Nutzer registriert ist. Beispiele für solche Instanzen sind ”chaos.socialünd ”mastodon.social”.
Der grundlegende Ablauf ist hierbei in Bild 1 zu sehen. Jede Applikation bekommt von der Mastodon-Instanz eine ID und ein Secret, welches vom Backend automatisch bezogen und gespeichert wird. Hierbei unterstützt das Backend beliebig viele Instanzen, es kommt eine sqlite-Datenbank zum Einsatz (siehe: backend/models/appData.js).
Der Client leitet mit Kenntnis der App-ID zur Authentifizierungsseite weiter, bei der der Nutzer unserer App Rechte vergeben oder ablehnen kann. Zurück kommt ein Code, mit dem das Backend (unter Zuhilfenahme des secrets und der ID) einen Token abholen kann. Dieser wird im Browser lokal gespeichert (Cookies).

4.2 Anzeige der Timeline

Das Frontend führt nach der Authentifizierung alle Kommunikation mit dem Mastodon-Server selbstständig durch. Hierzu wird, entweder das Formular zur Authentifizierung angezeigt, oder per REST die Posts aus der Timeline abgeholt. Diese werden dann, in ein HTML gerendert, in die Kachel geschrieben. Es kommt die fetch-API für HTTP-Anfragen zum Einsatz.
Die Funktion ist in Bild 2, 3 und 4 zu sehen. Die Implementierung erfolgt in der ”mastodon.mjs”.

4.3 Schwierigkeiten und Komplexität

Wirkliche Schwierigkeiten gab es bei der Implementierung nicht. Zu beachten ist natürlich, dass man durch Authentifizierung viel mehr Datenflüsse und auch Zustände hat, als es in einem reinen REST-API-Aufruf der Fall wäre. Insbesondere die Einbindung des Backends ist hier zu erwähnen, da Secrets auf keinen Fall zum Client gelangen dürfen.

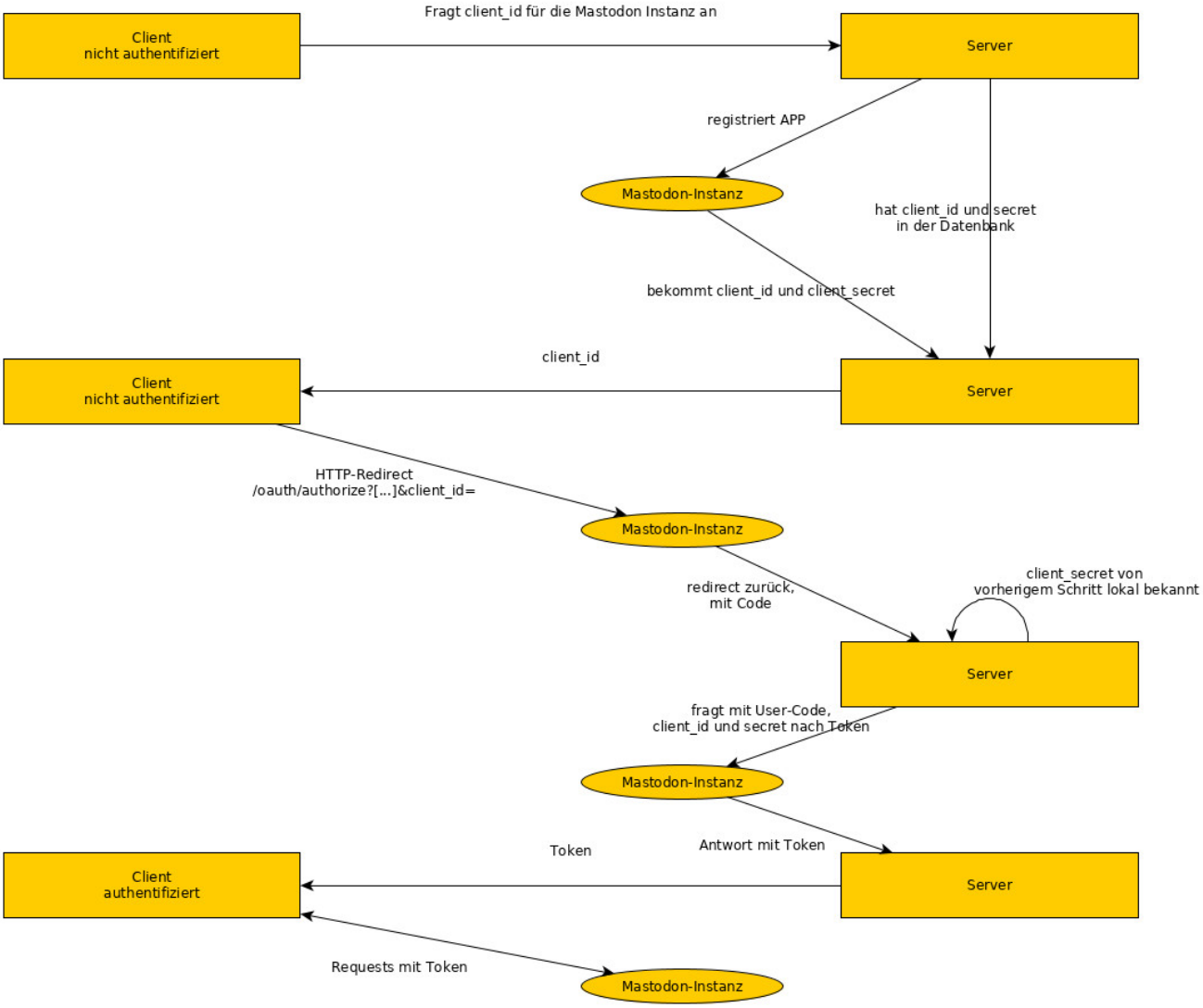

Abbildung 1. Mastodon Authentication

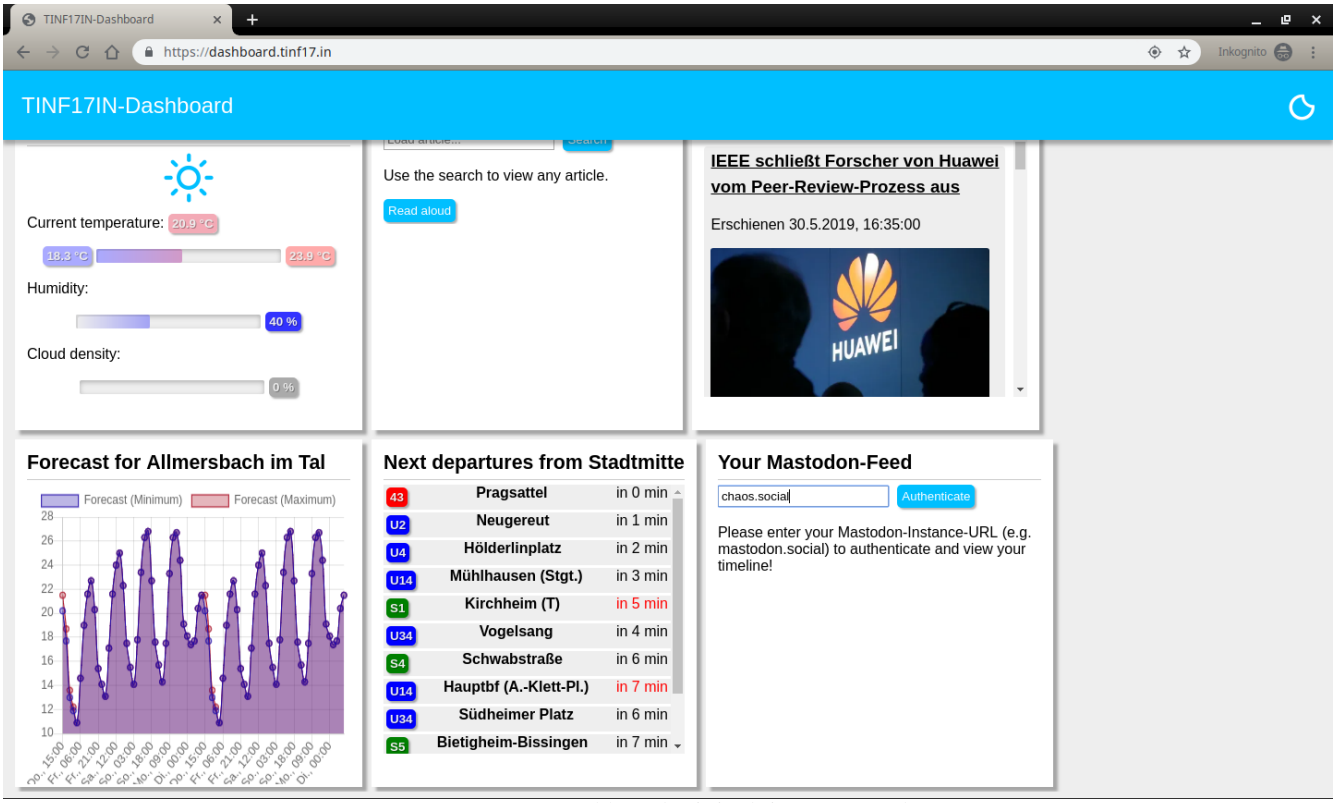

Abbildung 2. Dashboard mit inaktivem Mastodon

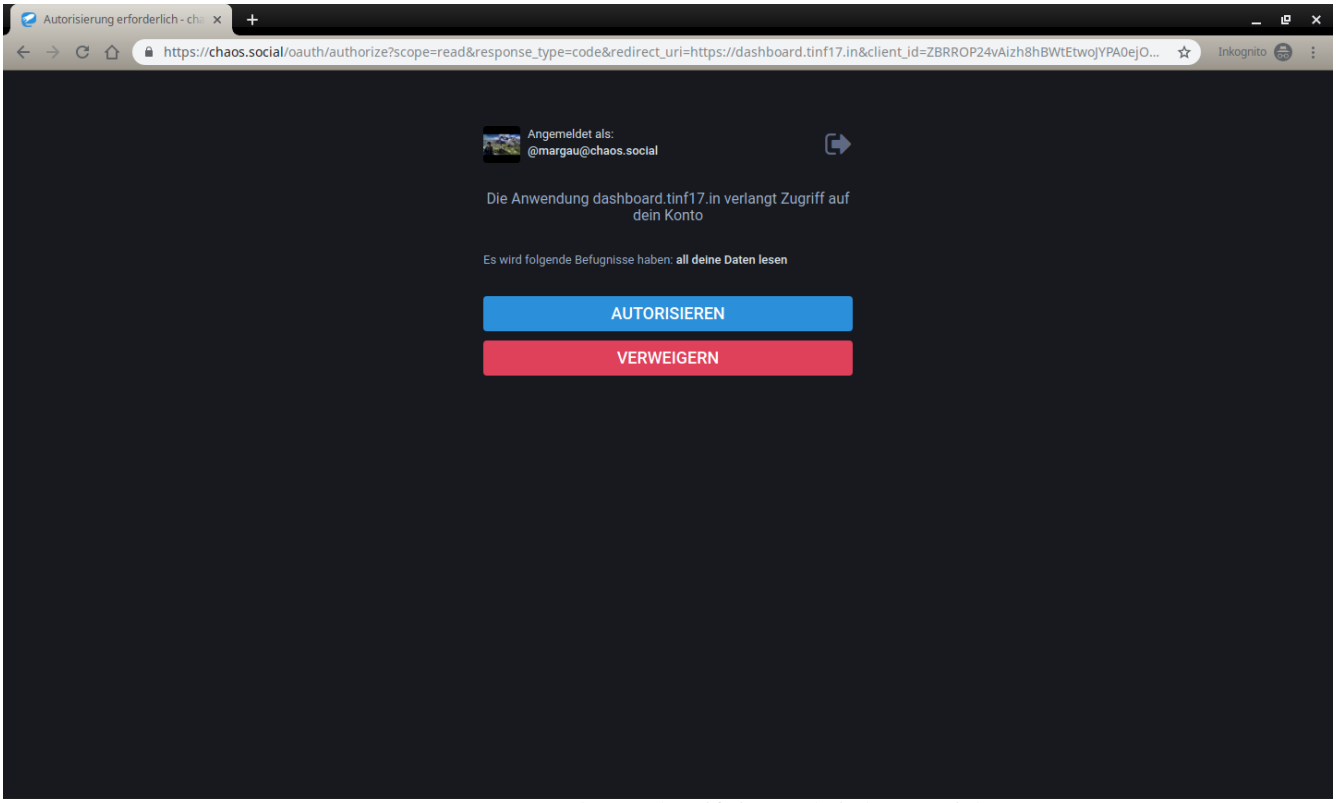

Abbildung 3. Mastodon Authentifizierung bei chaos.social

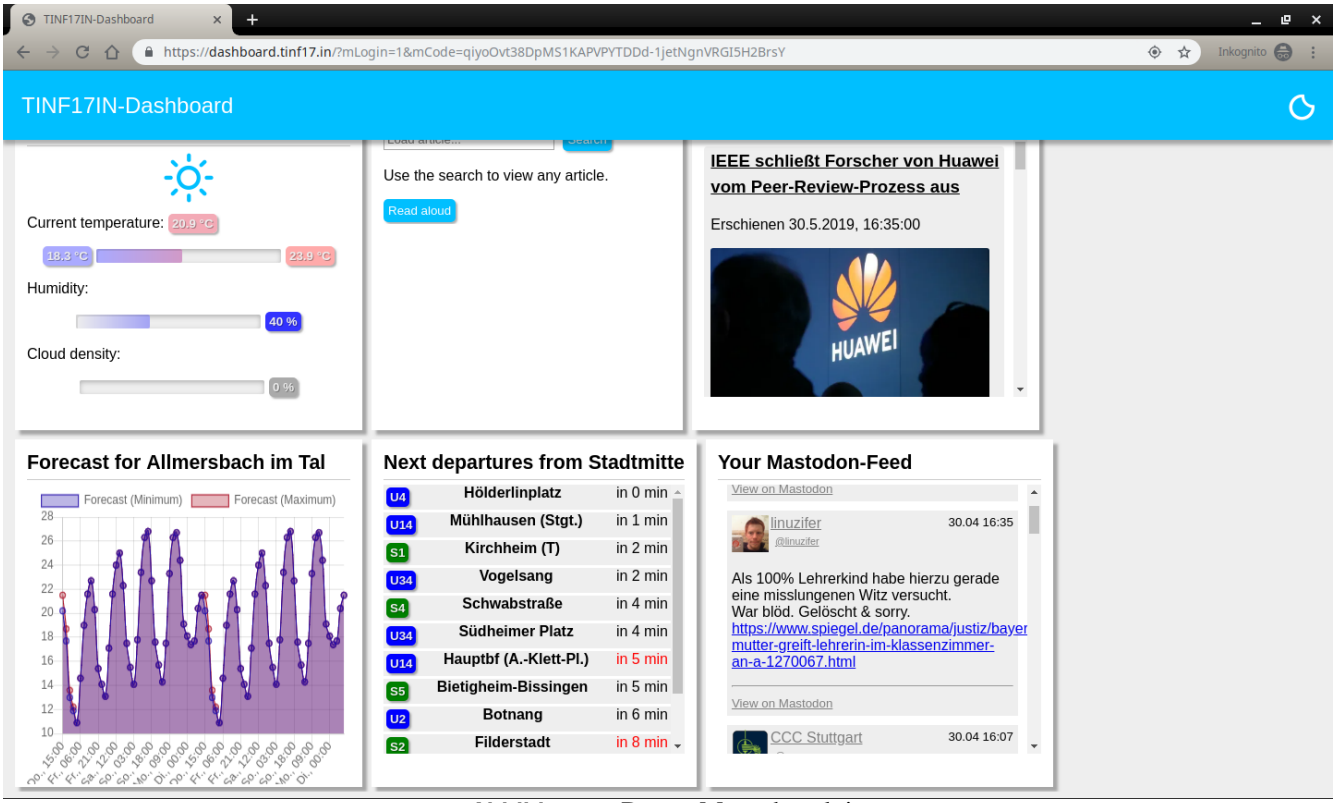

Abbildung 4. Demo, Mastodon aktiv