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

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

1. Grundlegende Architektur

Das Projekt ist in zwei Module aufgeteilt: **Frontend** und **Backend**.

Das Frontend ist in HTML, CSS und JavaScript realisiert und läuft komplett Clientseitig, also im Browser. Teilweise kommt das Framework „Vue“ [5] zum Einsatz.

Das Backend ist ebenfalls in JavaScript mithilfe der Laufzeitumgebung Node.js [2] realisiert und übernimmt die Auslieferung des Frontends sowie Authentifizierungs- und Proxyaufgaben. Für die HTTP-Schnittstelle kommt das Node-Modul Express [4] zum Einsatz.

2. Aufbau der Oberfläche

Die Idee des Frontends ist eine Art Dashboard, worüber verschiedene Dienste auf einem Blick verwendet werden können. Es sind mehrere Kacheln implementiert, jede Kachel beinhaltet genau einen Dienst.

2.1 Wetterkachel

Die Wetterkachel beinhaltet das aktuelle Wetter mit Temperatur, Luftfeuchtigkeit und Wolkendichte. Die Daten hierfür werden von OpenWeatherMap [**owm**] bezogen und aufbereitet.

2.2 Wikipediakachel

Die Wikipediakachel ermöglicht es, die Einleitung eines Artikels anzeigen und vorlesen zu lassen. Die Daten zum Anzeigen werden von der Wikipedia, die zum Vorlesen von der IBM-Watson-API bezogen und aufbereitet.

2.3 RSS-Feed-Kachel

Die RSS-Feed-Kachel beinhaltet die neuesten Nachrichten von Heise online [**heise**], dafür wird der RSS-Feed bezogen und aufbereitet.

2.4 Wettervorhersagekachel

Die Kachel für die Wettervorhersage zeigt die Minimal- und Maximaltemperaturen für die nächsten 5 Tage an. Die Daten werden ebenfalls von OpenWeatherMap [**owm**] bezogen. Zur Darstellung wird das Framework „Chart.js“ [**chartjs**] verwendet.

2.5 VVS-Kachel

Die VVS-Kachel zeigt die nächsten Abfahrten von der Haltestelle Stadtmitte an. Hierfür werden die Daten vom VVS bezogen und aufbereitet.

2.6 Mastodonkachel

Falls ein Mastodon-Account vorhanden ist, kann hier die eigene Mastodon-Timeline angezeigt werden. Die Daten werden vom jeweiligen Mostodonserver bezogen und aufbereitet.

3. Backend

Das Backend ist mittels express.js [4] aufgebaut. Für die Authentifizierung gegenüber Mastodon-Instanzen kommt noch eine Dateibasierte SQLITE-Datenbank mit der Abstraktionsschicht „Sequelize“ [1] zum 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 „dotenv“ [3] aus der lokalen Datei `.env` 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 „ActivityPub“-Protocol 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.social`“ und „`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 als Cookie gespeichert.

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`“.

5. Kurzanleitung

Um die Software zu erhalten müssen zuerst die beiden Git Repositories für das Frontend und das Backend heruntergeladen werden. Dies kann zum Beispiel durch den diese Befehle erfolgen:

```
git clone https://github.com/activitypub
    ↪ -frontend/backend.git
git submodule update --init --recursive
```

Mit dem zweiten Befehl wird das Frontend Repository in den gewünschten Ordner im Backend platziert. Um alle nötigen Abhängigkeiten herunter zu laden muss folgendes Kommando im Wurzelverzeichnis des Backend Ordners ausgeführt werden:

```
npm install
```

Als nächstes muss die die Datei .exampleenv in .env umbenannt werden und es müssen die entsprechenden API Token und Zugangsdaten eingegeben werden um die externen Dienste ansprechen zu können. Mit einem der Folgenden Kommandos kann dann das Backend gestartet werden:

```
npm run debug
node server.js
```

Nun ist der Server unter localhost auf Port 5000 erreichbar und das Frontend kann intuitiv bedient werden. Anmerkung: Die Mastodon OAUTH Funktionalität funktioniert möglicherweise

nur im dashboard.tinf17.in-deployment, da eine Weiterleitung nötig ist.

6. Fazit

Das vorliegende Projekt hat uns sowohl auf inhaltlicher als auch auf persönlicher Ebene weiter gebracht und wir konnten unsere Erfahrungen im Bereich Webtechnologien stark ausbauen.

6.1 Projektverlauf

Das Projekt lief neben einigen wenigen Stolpersteinen ohne größere Hindernisse ab. Wie bei den meisten Gruppenarbeiten war es auch bei dieser nicht immer leicht, eine faire Aufteilung der Arbeitspakete zu gewährleisten. Für die Zukunft nehmen wir mit, dass wir zu einem früheren Zeitpunkt Methoden aus dem Projektmanagement einsetzen wollen, um einen reibungslosen Ablauf, faire Arbeitsteilung und die Vermeidung von doppelter Arbeit zu gewährleisten.

6.2 Technologien

Vor allem die Arbeit mit dem JavaScript-Framework „Vue“ [5] war für uns alle neu. Das Framework hat uns aber durch den geringen Overhead und die einfache Handhabe sehr überzeugt. Die Arbeit mit „Vue“ hat uns weitergebracht und uns einen neuen Horizont in der Webentwicklung eröffnet. Auch war die Arbeit mit der Mastodon API neu und CSS wurde auch noch von keinem von uns in der Tiefe wie in diesem Projekt verwendet. Alles in allem kann gesagt werden, dass wir einige neue Technologien lernen und schätzen gelernt haben und einige Alte noch besser verstanden haben.

6.3 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.

Literatur

[1] Sascha Depold. *Sequelize*. 2014–present. URL: `http://docs.sequelizejs.com/`.

[2] Node.js Foundation. *Node.js*. URL: `https://nodejs.org/en/`.

[3] Scott Motte. *dotenv*. URL: `https://github.com/motdotla/dotenv`.

[4] StrongLoop, IBM und other expressjs.com contributors. *Express*. 2017. URL: `https://expressjs.com/`.

[5] Evan You. *Vue.js*. 2014–2019. URL: `https://vuejs.org/`.

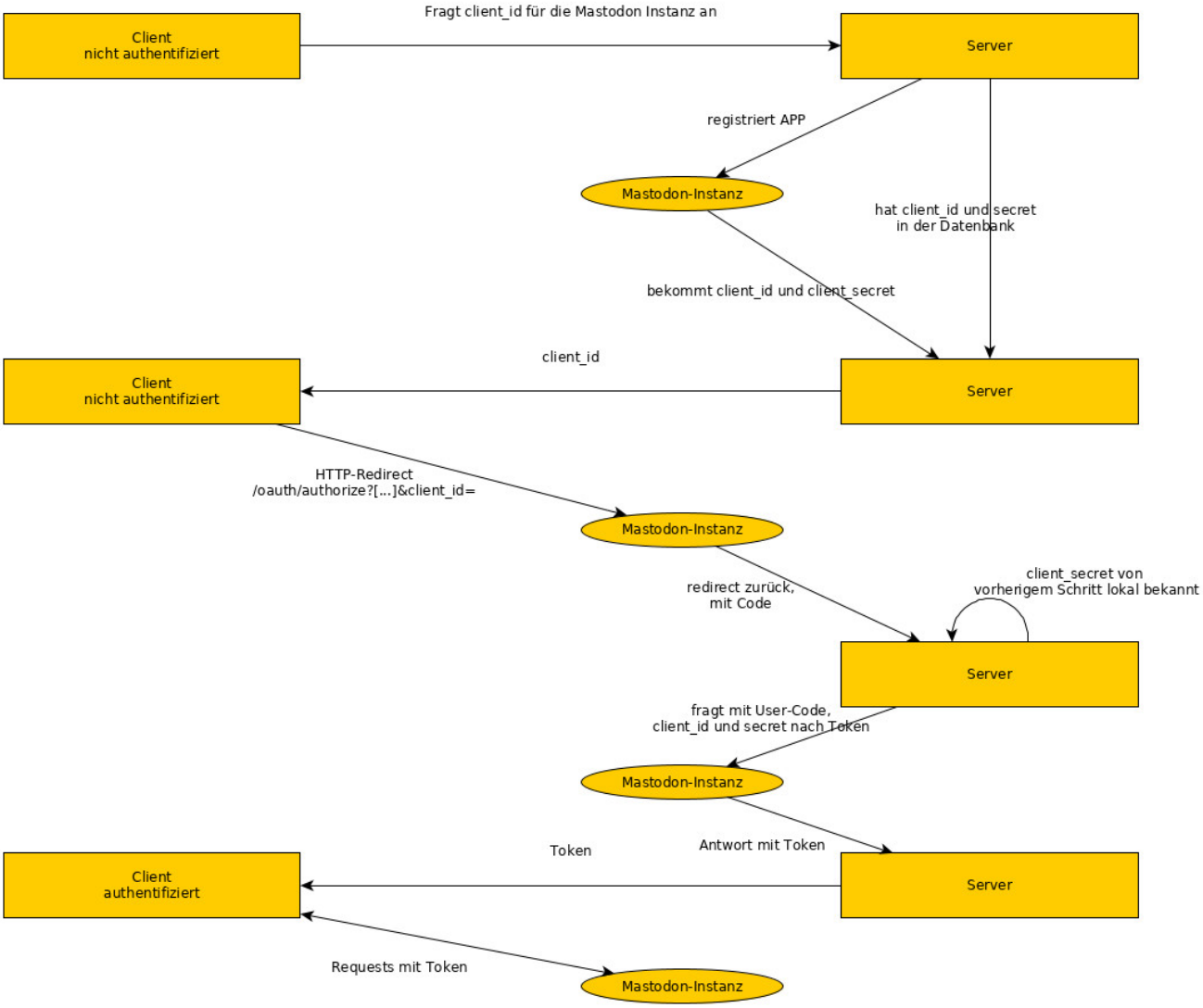

Abbildung 1. Mastodon Authentication

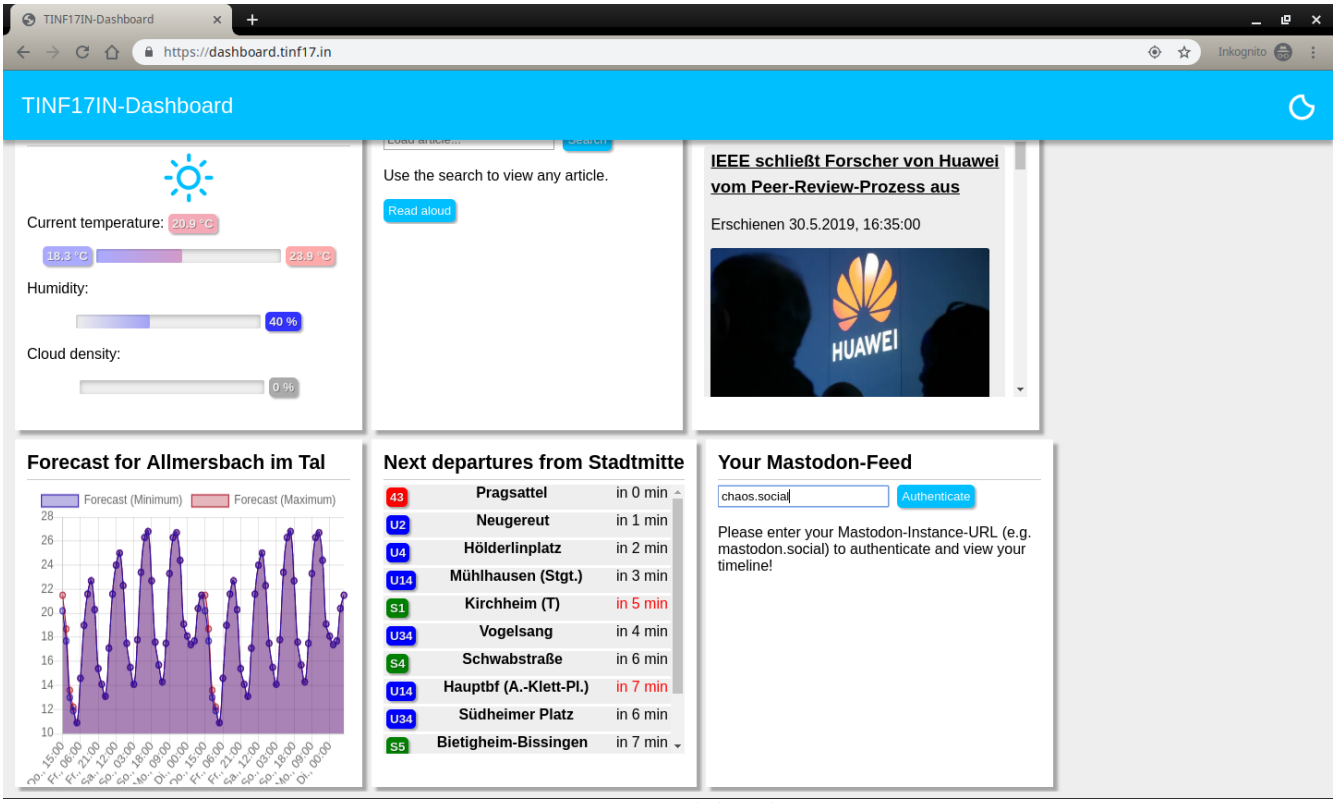

Abbildung 2. Dashboard mit inaktivem Mastodon

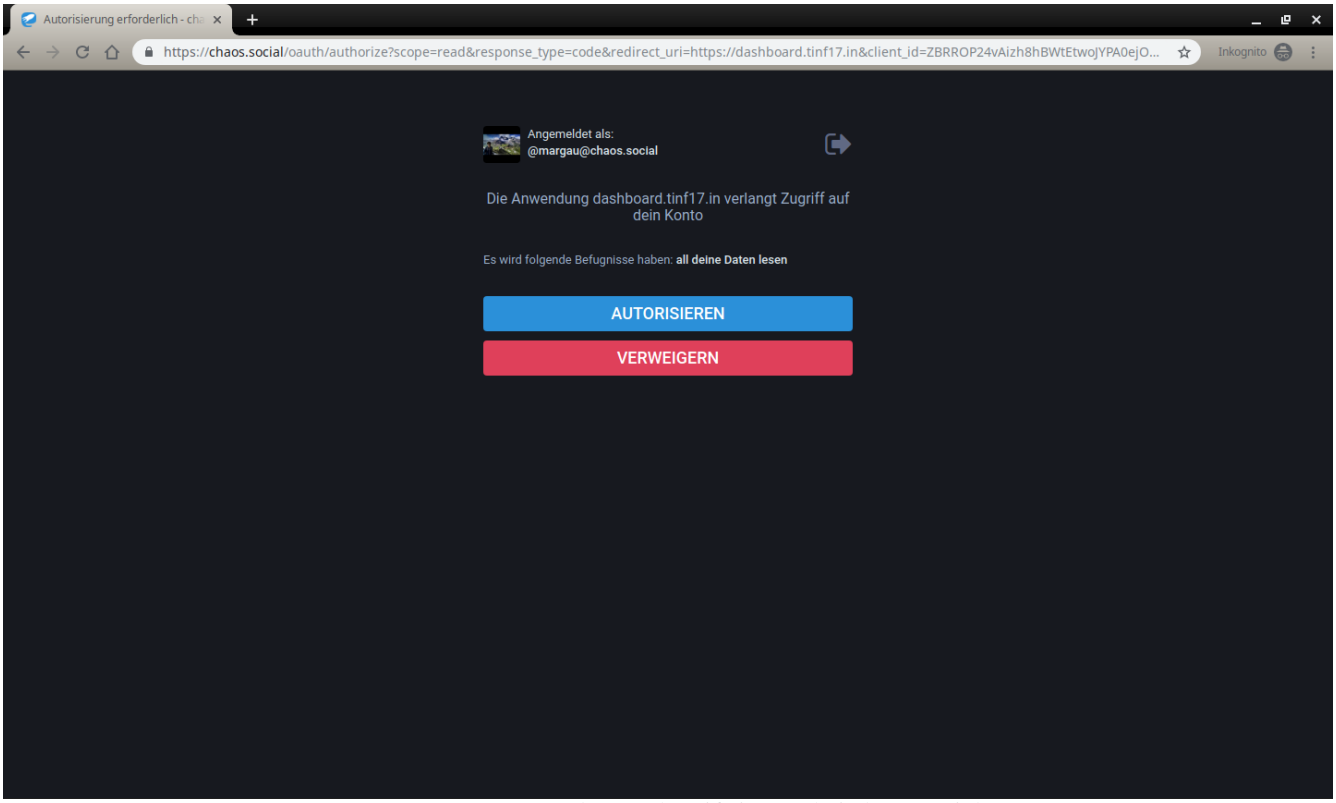

Abbildung 3. Mastodon Authentifizierung bei chaos.social

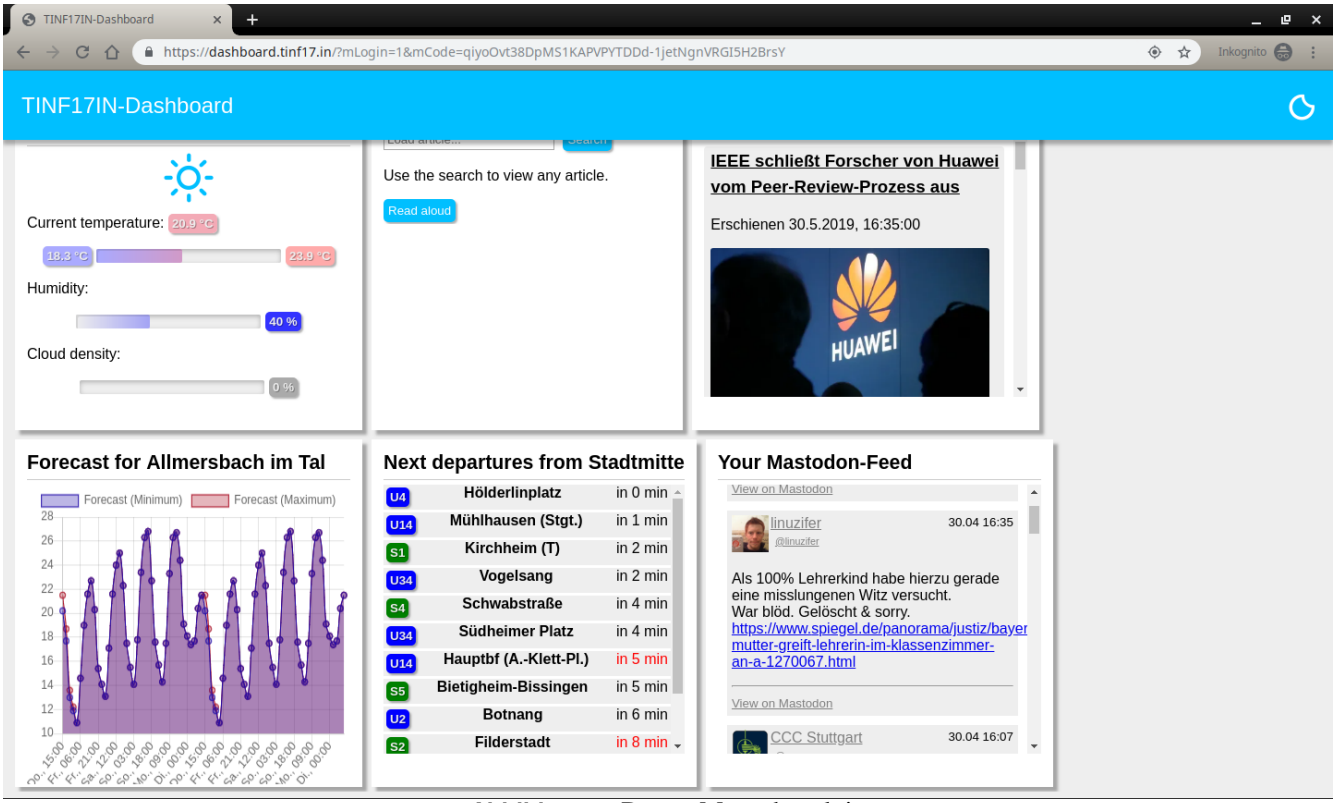

Abbildung 4. Demo, Mastodon aktiv