fleetport.io dev guide

fleetport ist ein Tool, das helfen soll, den Spritverbrauch von Autos darzustellen und Anhaltspunkte für Analysen bieten soll. In diesem Dev Guide wird die Struktur des Projektes erklärt.

Es gibt hier 3 Hauptakteure: Eine verbaute CAN-Tracking-Box in einem KFZ, einen fleetport-server und einen traccar-server. Die Can-Box sendet Daten an traccar, eine Open-Source "Modern GPS Tracking Platform", die das Decoding der Nachrichten übernimmt und in ein einheitliches Format konvertiert. Diese Daten sind die Basis, auf der fleetport agiert. Der fleetport-server kann, muss aber nicht auf der selben Maschine wie traccar laufen.

Fleetport läuft auf einem node.js Stack. Hier gibt es 2 große Teile. Einer ist für die Datenbeschaffung von traccar zuständig, der andere stellt die API bereit, auf die unsere Weboberfläche zugreift.

Das Polling von traccar-Daten läuft einmal alle 10 Minuten ab. Diese Daten werden dann untereinander verknüpft in einer SQLite Datenbank gespeichert. Von dort kann die API dann Daten an das Frontend liefern. Das Frontend selbst ist in Vue.JS geschrieben und mit webpack kompiliert.

Ordnerstruktur Backend:

```
├──certs // SSL - server.crt, server.key
├──config // default.json
├──log
│   ├──api // API request log
│   ├──frontend-server // frontend delivery log
│   └──node
└──src // source code
    └──schema // JSON Schema to check incoming requests
```

Der Haupteinstiegspunkt ist *app.js*. Hier laufen alle Module zusammen. *database-setup.js* wird einmal ausgeführt und stellt sicher, dass die SQLite3-Datenbank in der richtigen Form vorhanden ist. *express-server.js* initialisiert den express-Server. *persistence.js* ist für den Datenbankzugriff zuständig. Außerdem befinden sich hier die routes für die API-Calls des fleetport-frontends. *traccar-requests.js* behinhält alle API-Calls an traccar selbst. Unter /schema finden sich die JSON-Schema Dateieen, die prüfen, ob einkommene Requests die richtige Form haben.

Das Polling der Daten vom traccar-Server findet alle 10 Minuten statt.

Erweiterungsmöglichkeiten finden sich vor Allem in *persistence.js*, wo man die API mit neuen Funktionen ausstatten und diese dann im Frontend verwenden kann. Komplexere Datenbank-Queries und Multi-User-Unterstützung sind Beispiele für neue Features.

API Endpoints:

```
/absoluteConsumption
PARAMS: START, END
returns float

/averageConsumption
PARAMS: START, END
returns float

/mileage
PARAMS: START, END
returns float

/allTrips
PARAMS: START, END
returns object

/singleTrip
PARAMS: ID
returns object
```

fleetport user guide

Getting started

Hi, willkommen bei fleetport.io. Diese Dokumentation soll durch eine reibungslose Installation von allen benötigten Tools und Programmen sowie der Hardware führen.

fleetport.io ist eine Lösung zur Erfassung von Emissionsdaten zum eigenen Fahrverhalten zur Verfügung um dieses reflektieren und verbessern zu können. Mittels einer im Fahrzeug verbauten Hardwarelösung können Verbrauchsdaten aus dem Fahrzeug abgegriffen werden. Diese Daten werden dann auf einer Website sinnvoll dargestellt.

Vorraussetzungen:

- Webserver oder Computer mit statischer öffentlicher IP-Adresse

- Es werden mindestens 2GB RAM empfohlen. Lauffähig ist die Software auch mit weniger, die Stabilität ist dann aber nicht mehr gewährleistet.
- CAN-Tracker mit der Möglichkeit, den Spritverbrauch auszulesen
- SIM-Karte für CAN-Tracker

Walkthrough:

- Webserver besorgen (zB bei DigitalOcean, Hetzner, etc...)
- Traccar aufsetzen
- CAN-Box in KFZ verbauen
- CAN-Box in traccar hinzufügen & Daten validieren
- fleetport konfigurieren
- fleetport starten & initialen Datentransfer abwarten
- fleetport verwenden 😊

fleetport ist als node.js-Projekt grundsätzlich auf Windows, Linux und MacOS lauffähig. In diesem Beispiel wird ein VPS von digitalocean verwendet. Sobald der Server gestartet und man per SSH verbunden ist, gibt folgendes Tutorial einen guten Startpunkt für die Installation von traccar:

> Install Traccar on DigitalOcean VPS - Traccar
>
> https://www.traccar.org/install-digitalocean/

Anmerkung: In dem Tutorial ist eine alte Version von traccar verlinkt, für optionale Funktionalität sollte immer die aktuellste Version bezogen werden: https://github.com/traccar/traccar/releases

Anmerkung: Es kann auf einigen Linux-Distributionen vorkommen, dass Mysql nicht richtig funktioniert, wenn nach der Installation nicht dieser Command ausgeführt wird:

```
sudo mysql_secure_installation
```

Traccar sollte jetzt über das Webinterface auf Port 80 erreichbar sein. Der nächste Schritt ist die Auswahl passender Hardware und der Einbau in das KFZ. Unter https://www.traccar.org/devices/ ist eine Liste mit kompatiblen Geräten und Herstellerprotokollen verfügbar, auf deren Basis diese Entscheidung getroffen werden kann. Eine Liste mit einer Auswahl an Trackern findet sich hier: https://gpsgate.com/devices. Diese Liste einfach nach "CAN" durchsuchen und mit der traccar-Protokoll Liste abgleichen, bis ein gerät gefunden ist, das den Ansprüchen entspricht.

Einbauanleitungen für das jeweilige Gerät gibt es üblicherweise beim Hersteller selbst, wir haben aber trotzdem einen allgemeinen Leitfaden für den Einbau geschrieben. Dieser erfolgt selbstverständlich auf eigene Gefahr und fleetport.io trägt keinerlei Haftung.

Einbauanleitung CAN-Box

Der Einbau darf ausschließlich durch Fachpersonal erfolgen. Für den Einbau sowie den Abgriff der Stromversorgung und CAN-Signale sind die jeweiligen fahrzeugspezifischen Daten zu beachten. Eine Grundausstattung an verschiedenen Werkzeugen ist erforderlich, zusätzlich sind eine Crimpzange, Abisolierzange, div. Plastikkeile und evtl. ein Heißluftföhn zu empfehlen.

1. Fahrzeugspezifischen Einbauort sowie Abgriff der Spannungsversorgung sind den Dokumenten der CAN-Box sowie dem gültigen Stromlaufplan des Fahrzeuges zu entnehmen.

2. Abbauen der Innenraumverkleidungen nach Reparaturleitfaden des Herstellers.

3. Der Abgriff der Spannungsversorgung sowie der Abgriff des Zündungsplussignals kann auf diverse Arten erfolgen:

- Lötverbindung
- Direkt aus dem Sicherungskasten, hierfür sind diverse Zusatzmaterialien erforderlich (z. B.: Pins für den jeweiligen Sicherungsträger, Sicherungen)
- Abgriff an der Batterie (z. B.: mit Ringöse)
- „Y-Stück“ mit Löt- oder Quetschverbinder
- Abgriff mit Stromdieb

4. CAN-Leser am vorgesehenen Platz/Ort montieren: hierbei ist unbedingt zu beachten, dass die CAN-Leitungen nicht weiter als unbedingt notwendig aufgedrillt werden, zusätzlich muss beachtet werden, dass die CAN-Highleitung und die CAN-Lowleitung sich auf der jeweils richtigen Seite befinden und mit Kabelbindern in der richtigen Position befestigt werden.

5. Einbauplatz der CAN-Box auswählen: hierbei ist darauf zu achten, dass alle Klappen (z.B.: Staufächer oder Handschuhfach) in ihrer Bewegung uneingeschränkt bleiben, keine Scheuerstellen für die Überlängen der Leitungen entstehen und die thermische Belastung (z.B.: Wärmetauscher Heizung) so gering wie möglich gehalten werden.

6. Die Befestigung der CAN-Box ist stark vom Einbauort abhängig, kann z. B.: mit diversen Halterungen, doppelseitigem Klebeband oder Klettverschluss erfolgen. Wichtig ist hierbei, dass sich die Box nicht lösen kann und kein Geräusch oder Funktionseinschränkung (z.B. Blockade Lenksäule/Pedalerie) entsteht.

7. Diverse Leitungen an der CAN-Box welche für die gewünschte Funktion nicht benötigt werden, müssen isoliert und zusammengebunden werden, um Geräusche oder Funktionseinschränkungen zu vermeiden.

8. Verbindung des CAN-Leser mit der CAN-Box herstellen, hierbei ist auch darauf zu achten, dass die Kabeln auf so wenig Strecke wie möglich entdrillt sind, um gegen elektromagnetische Störungsimpulse immun sein zu können. Als Verbindung gibt es diverse Möglichkeiten, siehe Punkt 3.

9. Korrekte Kabelverlegung durchführen: Scheuerstellen vermeiden, Kabeln an fahrzeugseitigen Kabelsträngen mittels Kabelbinder befestigen.

10. Für die Fertigstellung des Fahrzeuges müssen alle abgenommenen Verkleidungen in umgekehrter Reihenfolge wieder befestigt werden, eine Funktionsprüfung des Fahrzeuges ist erforderlich, zum Schluss ist die

Löschung des Fehlerspeichers notwendig.

Die CAN-Box muss üblicherweise mit der Server-IP von traccar konfiguriert werden. Sobald dies getan ist, kann man das Gerät in traccar hinzufügen. Als Kontrolle kann man überprüfen, ob auch wirklich Daten ankommen. Damit ist die Arbeit mit traccar abgeschlossen.

Als nächsten Schritt sollte nodejs installiert werden: https://nodejs.org/en/. fleetport kann von unserem GitHub-Repository heruntergeladen werden: https://github.com/fleetport-io?tab=repositories. Hier müssen beide Repositories geladen werden.

Die benötigten Packages werden in beiden Fällen mit

```
npm install
```

heruntergeladen & installiert.

Im frontend wird dann per

```
npm run prod
```

in den Ordner /dist das Frontend kompiliert. Dieser Output wird dann in die andere repository verschoben und ist die Weboberfläche, die der Server ausliefert.

Im Backend befindet sich im Ordner "config" die Datei default.json, die konfiguriert werden muss:

default.json

```
"host": "IP Adresse des traccar-servers",
"port": Port des traccar-Servers,
"username": "traccar username",
"password": "traccar passwort",
"deviceID": standardmäßg 1, muss nur angepasst werden, wenn in traccar mehrere devices vorha
"SSL": true || false - wenn auf true gesetzt wird https verwendet. In dem Ordner certs müssen sich
"login": {
  "username": "username für frontend",
  "password": "passwort für frontend"
}
```

Für lokales SSL (ausschließlich zu Testzwecken!) dieses Tutorial befolgen:
https://www.freecodecamp.org/news/how-to-get-https-working-on-your-local-development-environment-in-5-minutes-7af615770eec/

Gestartet wird das Backend letztendlich mit

```
npm run start
```

Der erste Start kann länger dauern, wenn schon Daten aus der Vergangenheit in traccar vorhanden sind.

Wenn die Applikation im Hintergrund laufen soll, empfiehlt es sich, Tools wie 'screen', 'nohup' oder 'disown' zu verwenden. Außerdem kann man mit https://github.com/nexe/nexe eine ausführbare Datei erstellen.