# **Stand der MotionTrace-App**

## Neue Features & Erweiterungen seit dem 03.07.2023

* **Allgemeines**
  + Die App sendet (ggf. an MQTT) und speichert nun alle Daten des Location-Services.
  + 🡪 Neben *latitude*, *longitude*, *accuracy* und *timestamp* nun auch **altitude**, **speed** und **heading** (hierdurch werden weniger serverseitige Berechnungen notwendig)
  + **Bugfix**: Rückwärtsinkompatibilität nach Erweiterung der Datenspeicherung
  + **Bugfix**: Aufnahmestart vor Location-Update
    - Bugfix: Start der Aufnahme im Stillstand blockiert (in Version von Herrn Brückner noch vorhanden)
* **Einstellungen** (Seite)
  + MQTT-Einstellungen (IP, Port, Topic, Username) wurden um die Möglichkeit erweitert sich per Passwort gegenüber dem MQTT-Broker zu authentifizieren

# **Stand des MotionTrace-Backend**

## MQTT-Broker

Online und erreichbar! Allerdings aktuell nur für authentifizierte Nutzer (Username + Passwort), die explizit von mir angelegt werden müssen.

**IP:** 3.73.42.240 (leider instabil bei Neustart der AWS-Instanz)

**Port:** 1883

**Topic:** HSCO/5GKC/MotionTrace

## CassandraDB

Eingerichtet und im Einsatz. 🡪 Primär ausgewählt, da CaDB eine sehr gute Write-Performance aufweist.

**Past-Tabelle**: Persistente Speicherung vergangener Aufnahmen zur späteren Offline-Nutzung. Eindeutig partitioniert nach Nutzer-ID und Startzeitpunkt der Aufnahme. Innerhalb der Aufnahme werden Datenpunkte über einen Laufindex indiziert.

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

Automatisch generierte Beschreibung

**Ein Bild, das Text, Screenshot, Reihe, Schrift enthält.

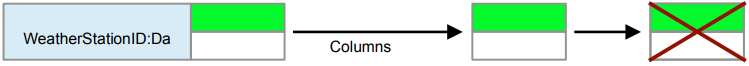
Automatisch generierte Beschreibung**

Nutzer-ID & Startzeit

**Current-Tabelle**: Inhalt und Partitionierung prinzipiell identisch zur Past-Tabelle, allerdings nur zur temporären Speicherung von Aufnahmen für die Online-Nutzung. Writes sind so konfiguriert, dass Datenpunkte nach 2min wieder aus der Tabelle entfernt werden. Außerdem werden Datenpunkte in umgekehrter Reihenfolge gespeichert, um einen möglichst schnellen Zugriff auf die aktuellen Daten zu gewährleisten.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung



Nutzer-ID & Startzeit

## MQTT-Client/Listener 🡪 Schnittstelle Broker & Datenbank

Umgesetzt als Python-Skript, welches auf der Instanz als Service läuft. Überwacht, sobald der MQTT-Broker online ist, dass oben genannte Topic und speist alle eintreffenden Daten im nachfolgenden JSON-Format in die beiden Datenbanktabellen ein. Zuvor werden jedoch noch die fehlenden Werte für **acceleration** und **angular\_speed** (auf Basis der eintreffenden und der zuletzt in der Current-Tabelle gespeicherten Daten – sofern vorhanden –) berechnet.

{

"client": "l\_koepper\_op7tpro",

"recording\_start": 1690359930741,

"index": 0,

"label": 3,

"location": {

"lat": 50.3935511,

"lon": 11.1194291,

"accuracy": 17.69499969482422,

"altitude": 529.9000244140625,

"speed": 0.09203340113162994,

"heading": 0,

"timestamp": 1690359930741

}

}

## Web-API 🡪 Schnittstelle Datenbank und Webanwendung

Umgesetzt mit **Python-Flask**. Online und erreichbar (wenn nicht getunnelt nur lokal auf dem Server) unter [**http://localhost:5000**](http://localhost:5000)**.**

**Noch in Entwicklung!**

**GET /motiontrace/past/recording\_keys**

Erlaubt Abfrage aller Partitionsschlüssel (Nutzer-ID & Aufnahmezeitpunkt) für die Aufnahmen in der Past-Tabelle. Basis für spätere Abfrage einzelner Aufnahmen.

[

        {

            "user\_id": "l\_koepper\_op7tpro",

            "start\_time": 1690286253788

        },

...

    ]

**GET /motiontrace/past/recording**

Erlaubt Abfrage einer einzelnen gesamten Aufnahme aus der Past-Tabelle.

{

        "user\_id": "l\_koepper\_op7tpro",

        "start\_time": 1690286253788,

        "vehicle": 3,

        "locations": [

            {

                "lat": 50.3638,

                "lon": 11.1465,

                "accuracy": 6.78,

                "timestamp": 1690286253788,

                "altitude": 455.0,

                "velocity": 13.78,

                "acceleration": 1.2,

                "heading": 68.23,

                "angular\_speed": 12.54

            },

...

        ]

    }

**GET /motiontrace/current/recorders**

Erlaubt die Abfrage aller aktuell aktiven Aufnehmenden entsprechend der Current-Tabelle.

[

        {

            "user\_id": "l\_koepper\_op7tpro",

            "real\_type": 3,

            "predicted\_type": 3, // Vorhersage durch 2s\_2min-RNN-Modell entsprechend der

der Werte in der Current-Tabelle

            "prediction": [0.01, 0.19, 0.30, 0.50], // zug. Vorhersagewahrscheinlichkeiten

            "location": { // letzte Position

                "lat": 50.3638,

                "lon": 11.1465,

                "accuracy": 6.78,

                "timestamp": 1690286253788,

                "altitude": 455.0,

                "velocity": 13.78,

                "acceleration": 1.2,

                "heading": 68.23,

                "angular\_speed": 12.54

            }

        },

...

]

## MotionTrace-Dashboard-Webanwendung

Wird mit **Angular** umgesetzt und befindet sich aktuell **noch in Entwicklung**. Siehe Prototyp.

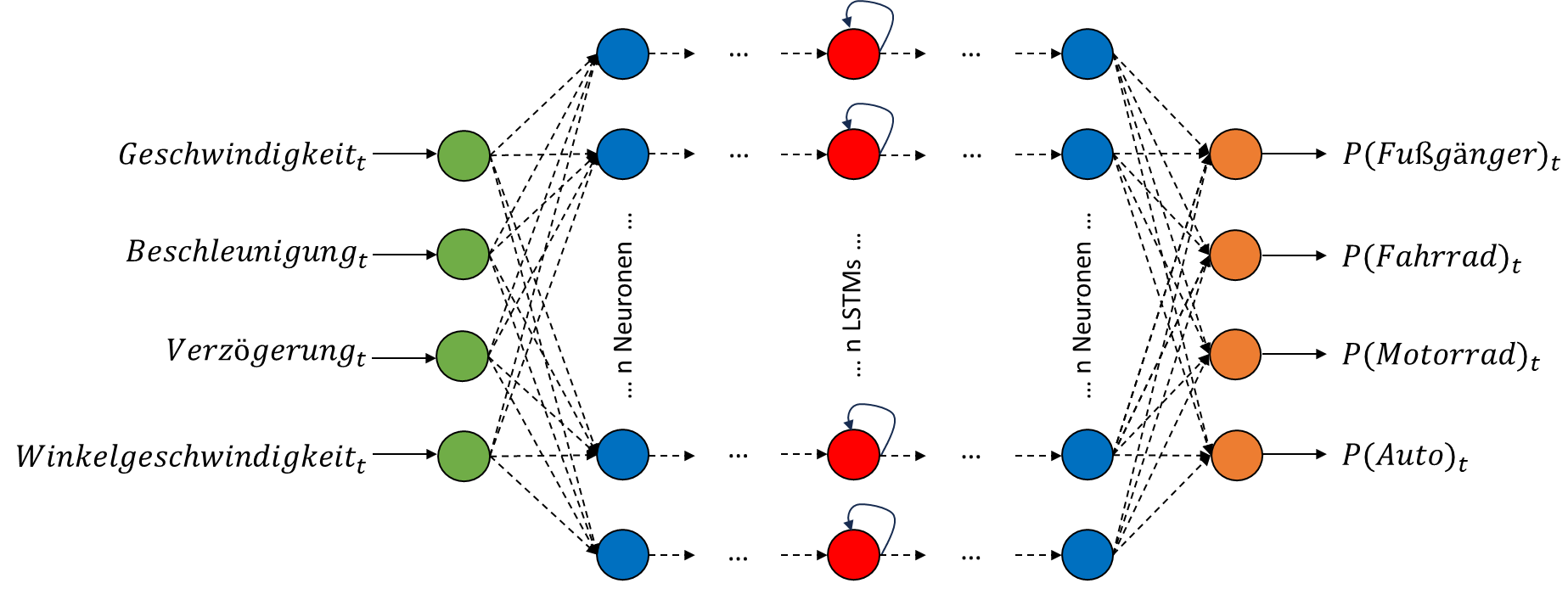
# **Datensammlung mit der App**

Ein Bild, das Text, Screenshot, Rechteck, Diagramm enthält.

Automatisch generierte Beschreibung

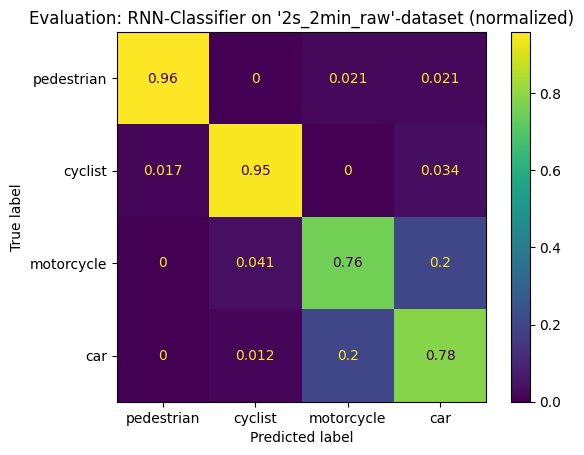
# **Klassifikation mittels RNNs**

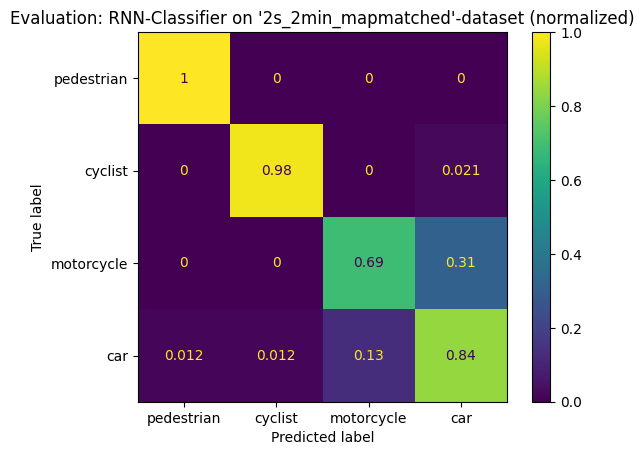
## Basisarchitektur



## Zwischenergebnis Hyperparameteroptimierung: Beste Modelle

2s-2min-Datensatz. RNN mit 128 Neuronen pro Schicht, 4-LSTM-Schichten und 4 vor- bzw. nachverarbeitenden Schichten (AF: tanh).



2s-2min-Datensatz (map-matched). RNN mit 128 Neuronen pro Schicht, 4-LSTM-Schichten und 2 vor- bzw. nachverarbeitenden Schichten (AF: tanh).

2s-4min-Datensatz. RNN mit 128 Neuronen pro Schicht, 2-LSTM-Schichten und 2 vor- bzw. nachverarbeitenden Schichten (AF: tanh).

Ein Bild, das Text, Screenshot, Diagramm, Quadrat enthält.

Automatisch generierte Beschreibung