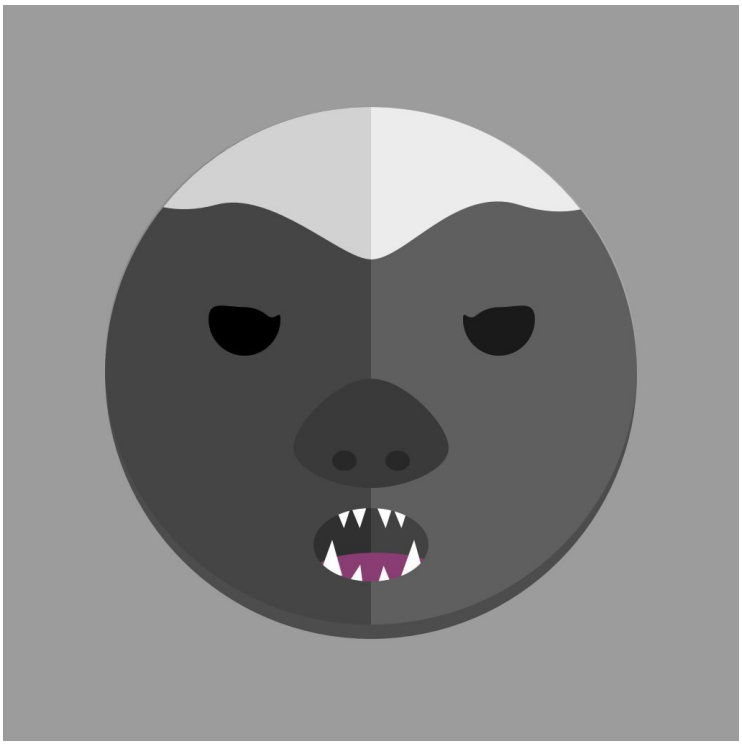
Technische Informationen



HoneyBadger

26. August, 2018

Team Members

Raphael Antonietti
Nick Gerber
Jonas Niestroj
Manuel Gysin

Contents

Contents 2
1 Ausgangslage 4
1.1 Technische Entscheide 4
2 Technischer Aufbau 5
2.1 Verwendete Komponenten 5
2.2 Schnittstellen 6
2.2.1 TDI 6
2.2.2 REST 6
2.2.3 Websocket 6
2.2.4 CSV 6
3 Implementation 7
3.1 Warum ein Backend 7
3.2 Yay, that rocks! 7
3.3 Visuelle Darstellung 8
4 Abgrenzung und offene Punkte 9

Revisionen

Version	Date	Name	Description
1	25/08/18	Manuel Gysin	Initial Dokument

1 Ausgangslage

Unser Ziel ist es, dass die BernMobil App einen Mehrwert zu den bestehenden Anwendungen auf dem Markt bietet. Daher gehend haben wir den Fokus auf einmalige Features gesetzt, welche die Endbenutzer neugierig machen und an die App fesseln.

Dies erreichen wir mit der Gamification der kompletten Anwendung, womit wir die Neugierde, sowie den Sammel- und Entwicklungstrieb des Endbenutzers ansprechen.

Zusätzlich soll die Anwendung allgemeine Betriebsinformationen enthalten, welche die Navigation innerhalb des BernMobil Netzes unterstützten und dem Endbenutzer interessante Details näherbringen.

1.1 Technische Entscheide

Uns war wichtig auf offene und erprobte Technologie zusetzen, welche plattformneutral eingesetzt werden kann, ob Backend oder Frontend.

Asp .Net Core

- Plattform unabhängig
- Gerüstet für die Zukunft dank Docker-Integration und Cloud-optimized runtime
- Opensource

ReactJS

- Ergiebige Auswahl an Third Party Modulen
- Runtime Performance
- Enterprise optimiert

OpenStreetMap

- Selfhosting möglich
- Copyleft Datenbasis, keine Lizenzgebühren
- Community erarbeitet Daten

2 Technischer Aufbau

Die Anwendung verfügt über zwei Hauptkomponenten. Dabei handelt es sich um das Frontend und das Backend. Diese kommunizieren per REST und WebSockets und tauschen damit die Nutzdaten aus.

Für die Websockets wurde sich bewusst ausfolgenden Gründen entschieden:

- Reduzierung der Calls auf die TDI-API
- Streaming der Livedaten an das Frontend
- Flexible Push Benachrichtigung ohne Polling

2.1 Verwendete Komponenten

Folgende Komponenten werden Verwendet:

Komponente	Version	Lizenz	Typ	Verwendungszweck
.Net Core	2.1.401	MIT/Apache 2	Backend	Bibliotheken zur Entwicklung
ASP.Net Core	2.1.1	MIT/Apache 2	Backend	Web-API-Abbildung
protobuf-net	2.3.17	BSD	Backend	Protobuf De/Serialisierung
ReactJs	16.4.2	MIT	Frontend	Grafische Darstellung Frontend
React-Leaflet	2.0.0	MIT	Frontend	OpenStreetMap Integration

2.2 Schnittstellen

Für die Aufbereitung und Präsentation der Daten wurden verschiedene APIs angebunden. Diese werden weiter unten detaillierter aufgeführt.

2.2.1 TDI

Die Traveller Data Interface bietet Real Time Informationen über die Fahrzeuge von BernMobil. Die API wird von BernMobil zur Verfügung gestellt.

Besonderheiten

- **Protocol-Buffer wird verwendet**
- **Ca. alle 2 Sekunden neue Livedaten**

2.2.2 REST

Das Backend bietet dem Frontend eine REST-API, welche Detailinformationen für die Fahrzeuge zur Verfügung stellt. Die Daten werden als JSON-Format gesendet und Empfangen. Für spätere Funktionalität werden über diese Schnittstelle weitere Nutzdaten ausgetauscht, welche nicht Real Time sind.

Besonderheiten

- Keine

2.2.3 Websocket

Der Websocket versorgt alle aktiven Clients mit den Real Time Daten, welche über die TDI Schnittstelle gewonnen werden und mit der CSV Schnittstelle angereichert werden. Der Websocket dient als Broadcast Channel für alle Clients und soll die API-Calls auf die TDI Schnittstelle minimieren.

Besonderheiten

- Keine

2.2.4 CSV

Die CSV Schnittstelle dient dazu die Daten der TDI-Schnittstelle anzureichern.

Besonderheiten

- CSV wurde nach JSON konvertiert

3 Implementation

3.1 Warum ein Backend

Bei der Betrachtung der Schnittstelle und der Komplexität der Daten und durch die Separation der Zusatzdaten als CSV, haben wir schnell den Entschluss gefasst, dass wir einen Backend zum Frontend entwickeln.

Durch die Gegebenheit, dass die App Benutzerdaten verwalten muss und auch der BernMobil Trackingdaten zur Verfügung stellen soll, war ein Backend unumgänglich.

In der jetzigen Implementierung bietet das Backend schon mehrere Vorteile, weile wie folgt sind:

- Konsolidierung der Livedaten und Anreicherung mit Details
- Entlastung der TDI API durch das Streaming per Websocket
- Skalierung per Cloud ohne weiteres Möglich

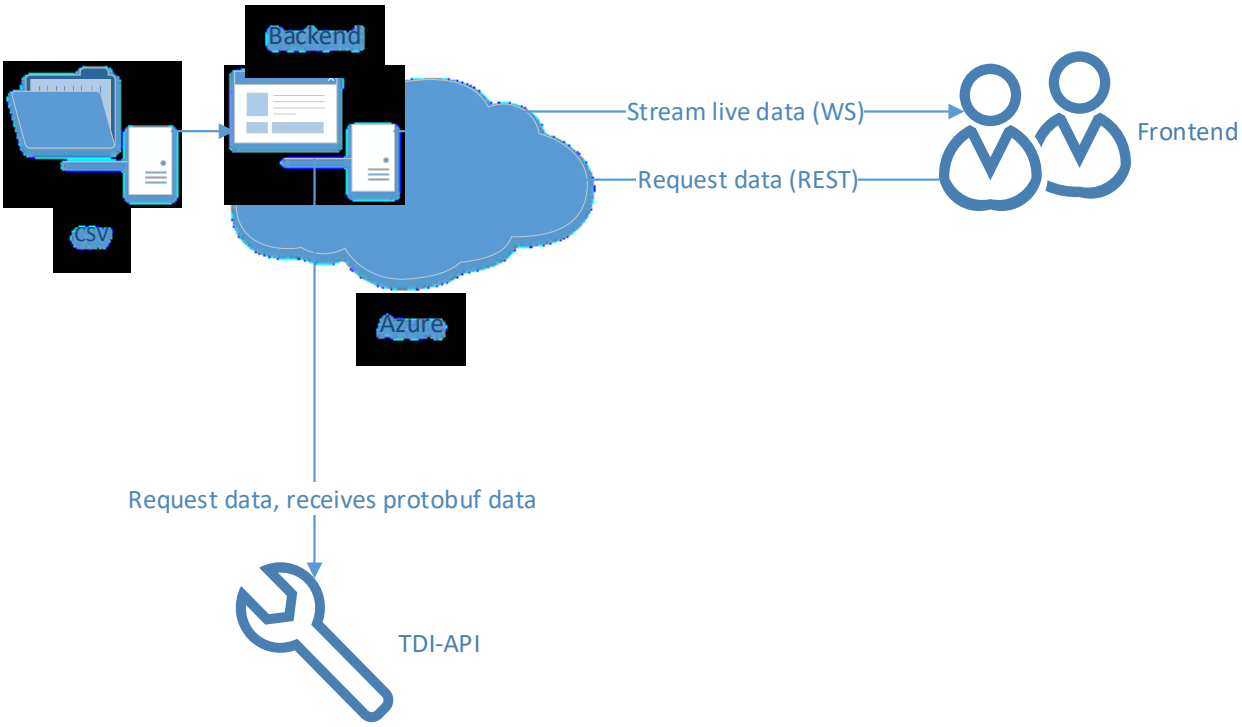
3.2 Yay, that rocks!

Besonders stolz sind wir auf die Live-Map und das Streaming der Livedaten. Wir können eine optimierte und hoch performante Datenverteilung gewährleisten, welche die TDI-Schnittstelle bestmöglich schont und konsistente Daten zeitgleich an alle Clients verteilt. Mit dieser Schnittstelle kann im einem weiteren Schritt auch ein gezieltes Auswerten von Trackinginformationen erfolgen und das in Echtzeit. (Wer ist auf der Tram 9, etc.)

Video mit der Funktionalität: https://github.com/CPlusPlus17/HoneyBadger-BERNMOBIL/blob/master/01_Dokumentation/Demovideo.mp4

3.3 Visuelle Darstellung

Stark vereinfachte Darstellung der Systemarchitektur.



4 Abgrenzung und offene Punkte

Ziel war es die Idee und die Grundfunktionalität der App als Prototype abzubilden. Alle Benutzerbezogenen Daten werden nicht gesammelt und verwertet. Die TDI-API konnte erfolgreich eingebunden werden und legt damit die Basis für die Implementierung der weiteren Features.

Umgesetzte Features:

- Anbindung TDI-API mit Protobuf Messages
- Live-Streaming der BERNMOBIL Fahrzeuge
- Spezifisches Fahrzeug finden
- Detailansicht eines Fahrzeugs

Folgende angedachten Features wurde nicht implementiert aus zeitlichen Gründen:

- Fahrplan
- Usertracking
- Störungsmeldungen
- Alternativrouten
- Routentipps für Touristen
- Smart Ticketing mit NFC
- Gamification Features

5 Sourcen

Jeglicher Code und Designelemente befinden sich auf Github:

https://github.com/CPlusPlus17/HoneyBadger-BERNMOBIL