API Dokumentation Bustracker

Hochschule Kaiserslautern IMST
Fabian Kalweit,
Johannes Schmitt

Abstract

Die Bustracker API ist eine RestAPI für die Anwendung Bustracker. Die API stellt Funktionen zum Speichern und Laden von Positionsdaten, Benutzerdaten und Daten über die eingesetzten IBeacons bereit.

Diese Dokumentation beschreibt den allgemeinen Aufbau der Bustracker API und genauer die einzelnen Methoden.

Contents

1 Allgemeiner Aufbau

Die RestAPI wurde mit TypeScript und NodeJs umgesetzt. Die Daten werden in einer MariaDb gehalten und über TypeScript-MYSQL abgefragt. In Abbildung 1 ist die Kommunikation der drei Elemente dargestellt.

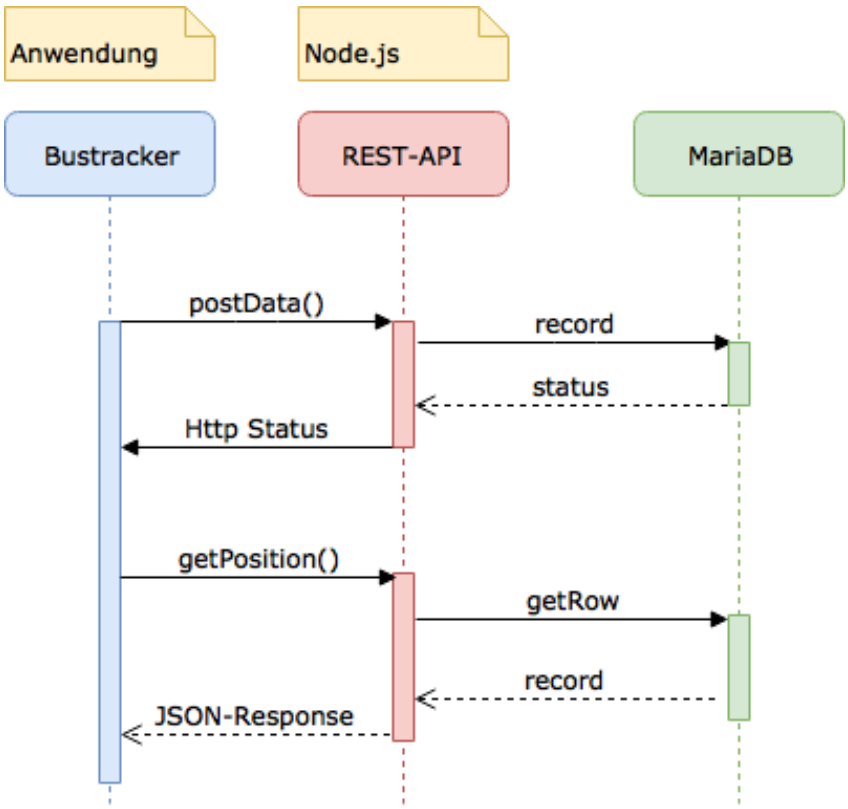

Figure 1: Kommunikation mit der Datenbank und dem Bustracker

1.1 Abhängikeiten der RestAPI

Alle Abhängigkeiten werden mit npm installiert.

devDependencies:

typescript : TypeScript Paket. Sollte global installiert werden.

mocha : Testframework für TypeScript

chai und chai-http : Erweiterte Asserts dür TypeScript.

types/mysql : MYSQL-Types für TypeScript

gulp und gulp-typescript : Ermöglicht das Erstellen von Scripts zum Beispiel zum Erstellen der Anwendung.

ts-node : Anbindung von TypeScript an NodeJS

Dependencies:

express : Express ist ein einfaches und flexibles Node.js-Framework von Webanwendungen, das zahlreiche leistungsfähige Features und Funktionen für Webanwendungen und mobile Anwendungen bereitstellt. Mithilfe unzähliger HTTP-Dienstprogrammmethoden und Middlewarefunktionen gestaltet sich das Erstellen einer leistungsfähigen

API schnell und einfach.
http://expressjs.com/de/

morgan : HTTP request logger middleware for node.js

type-sql : Ein typischerer Querybuilder für SQL erstellt mit TypeScript. Es werden Postgres und MySQL unterstützt.
https://github.com/ggmod/type-sql

2 Bustracker API

Hier werden die von außen sichtbaren Klassen der Bustracker API beschreiben.

2.1 PositionRouter

Der **Position Router** stellt Methoden zum Laden und Speichern von Positionsdaten zur Verfügung. Im Folgenden sind die einzelnen Methoden beschreiben:

getPositionsg : Gibt alle Positionen in der Datenbank zurück. Wird über die Adresse GET **root/api/v1/positions** aufgerufen.
Bei Erfolg gibt die Methode den Status 200, die Nachricht 'Success' und die Daten zurück.
Bei einem internen Fehlschlag gibt die Methode 500, die Nachricht 'Database error' und die Fehlernachricht des DBService.

getUserPosition : Gibt alle Positionen eines bestimmten Benutzers zurück. Wird über die Adresse GET **root/api/v1/positions/:id** aufgerufen. Bei Erfolg gibt die Methode den Status 200, die Nachricht 'Success' und die Daten zurück.
Bei einem internen Fehlschlag gibt die Methode 500, die Nachricht 'Database error' und die Fehlernachricht des DBService.
Wird eine nicht lesbare oder nicht vorhandene User-ID mitgegeben, gibt die Methode den Status 404 un die nachricht 'No position found with the given id.' zurück.

getLastUserPosition : Gibt die letzten x Positionen eines bestimmten Benutzers zurück. Wird über die Adresse GET **root/api/v1/positions/last** aufgerufen.
Parameter: userID und limit →
root/api/v1/positions/last?id=n&limit=x
Bei Erfolg gibt die Methode den Status 200, die Nachricht 'Success' und die Daten zurück.
Bei einem internen Fehlschlag gibt die Methode 500, die Nachricht 'Database error' und die Fehlernachricht des DBService.
Wird eine nicht lesbare oder nicht vorhandene User-ID mitgegeben, gibt die Methode den Status 404 un die nachricht 'No position found with the given id.' zurück.

postPosition : Speichert einen Positionsdatensatz in der Datenbank. Wird über POST **root/api/v1/positions** aufgerufen.
Bei Erfolg gibt die Methode den Status 200, die Nachricht 'Success' zurück.
Bei einem internen Fehlschlag gibt die Methode 500, die Nachricht 'Database error' und die Fehlernachricht des DBService.
Sind die Werte für lat oder lon gleich null, gibt die Methode den Status 400 un die nachricht 'lat or lon = 0.' zurück.

2.2 UserRouter

Der **User Router** stellt Methoden zum Laden und Speichern von Benutzerdaten zur Verfügung. Im Folgenden sind die einzelnen Methoden beschreiben:

getUser : Gibt alle User in der Datenbank zurück. Wird über die Adresse GET **root/api/v1/users** aufgerufen.
Bei Erfolg gibt die Methode den Status 200, die Nachricht 'Success' und die Daten zurück.
Bei einem internen Fehlschlag gibt die Methode 500, die Nachricht 'Database error' und die Fehlernachricht des DBService.