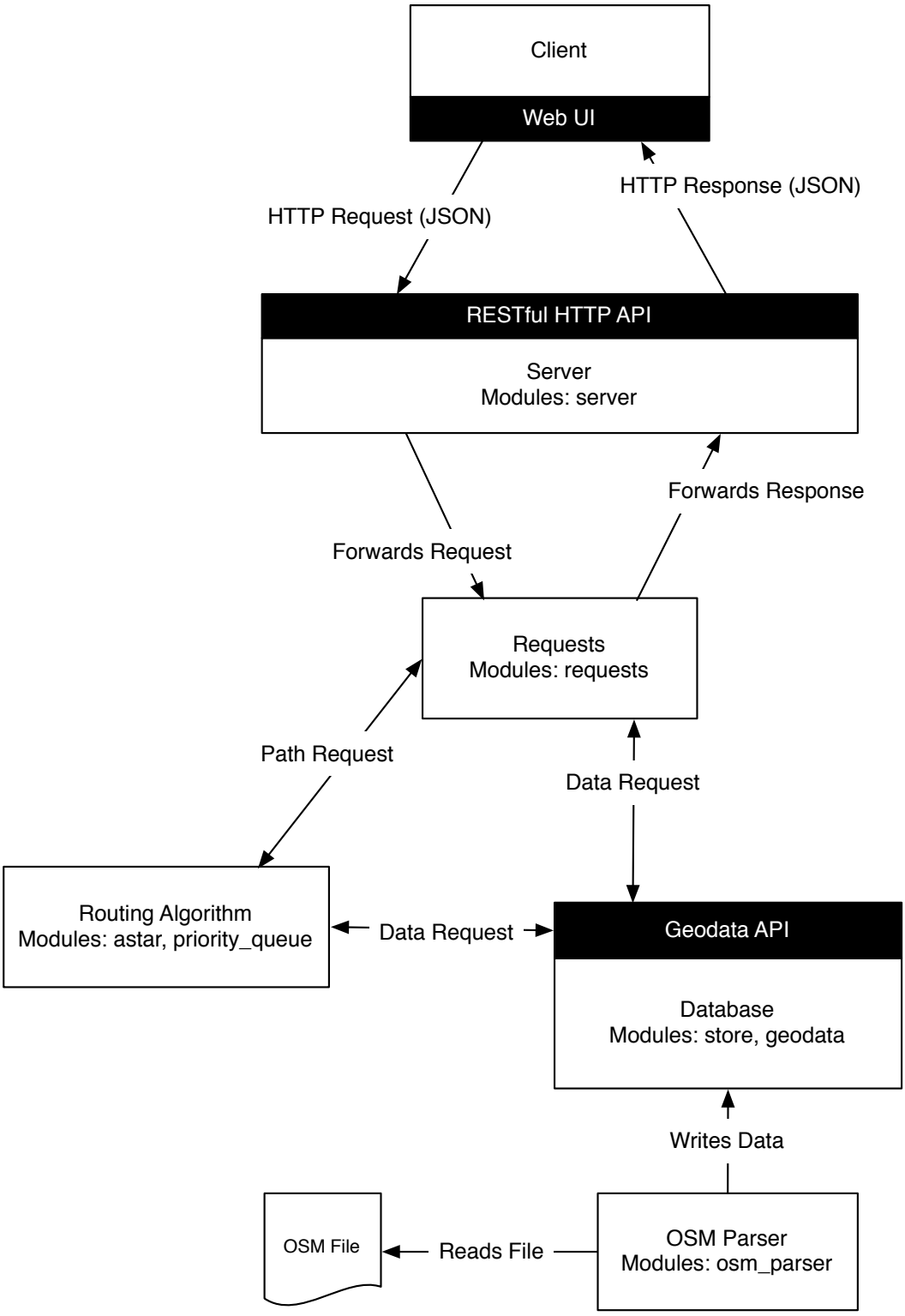
OSM Routing

Projektziel

Ziel des Projektes ist die Schaffung der technischen Infrastruktur, welche die Entwicklung eines Navigationssystems für körperlich behinderte Menschen ermöglicht.
Es soll möglich sein, den kürzesten Weg zwischen zwei Nodes eines OpenStreetMap Datensets zu ermitteln.
Der ermittelte Weg soll von einem Menschen mit körperlicher Behinderung komplett begehbar sein.
Das System soll ausserdem eine Wegbeschreibung des ermittelten Weges generieren.

Architektur

Im folgenden wird eine Übersicht über die Architektur des Systems gegeben. Dazu gehören ein Diagramm über die einzelnen Softwaremodule, eine Beschreibung des Ablaufs einer Anfrage sowie eine Übersicht über die verwendeten Erlang Module.



Bearbeitung einer Anfrage an das System

1. HTTP Anfrage wird gesendet.

2. Der Server erhält die Anfrage. Die Bearbeitung der Anfrage übernimmt das

Erlang System.

3. server:request() wird aufgerufen. Diese Funktion dient der Erkennung der Art der Anfrage und zur Weiterleitung an die entsprechenden Submodule.

4. requests:route() wird aufgerufen. Diese Funktion berechnet die Route der Anfrage. Dies geschieht mithilfe der Funktionen des astar Erlang moduls.

5. Die Funtionen aus dem astar Modul greifen zur Berechnung der Route auf das geodata Erlang Modul zu. Das geodata Modul dient als High-Level API für die Datenbank. Alle Anfrage des geodata Moduls an die Datenbank laufen über das store Modul, welches als Low-Level API für diese dient.

6. Nachdem die Route berechnet ist wird die HTTP Antwort in JSON enkodiert. Anschliesend sendet die Funktion server:request() die HTTP Antwort.

Erlang Module

astar
Dieses Modul implementiert einen A* Algorithmus, um den kürzesten Weg zwischen zwei Knoten zu finden.

geodata
Diese Modul dient als High-Level API für die Datenbank. Alle Zugriffe auf die Datenbank laufen über dieses Modul.

osm_parser
Mithilfe dieses Moduls ist es mögich eine .osm Datei auszulesen und deren Inhalt in der Datenbank zu sichern. Die Daten werden beim Einlesen über generische Regeln gefiltert. So lassen sich leicht für Behinderte ungeignete Wege aus dem Datenset entfernen.

priority_queue
Dieses Modul implementiert eine Prioritätswarteschlange, welche vom Routing-Algorithmus benötigt wird.

requests

Dieses Modul implementiert die HTTP Antworten des HTTP Servers.

routing
Dieses Modul dient als Schnittstelle für das komplette System. Andere Erlang Systeme sollten über diese Schnittstelle mit dem Routingsystem kommunizieren.

server
Dieses Modul implementiert die HTTP API des Servers. Es ist nur die Serverlogik und die JSON Serialisierung enthalten.

store
Dieses Modul dient als API der ets Tabellen. Die ets Tabellen sollten nur über diese API angespochen werden.

Softwarekomponenten

Programmiersprache

Die Wahl der Programmiersprache des Backends viel auf Erlang.

PRO:

- Virtuelle Maschine: Erlang läuft auf einer VM. Damit ist es sehr leicht möglich ein Programm im laufenden Betrieb zu ändern, ohne es komplett neu zu kompilieren und neu zu starten. Dies erhöht die Entwickler-Produktivität.
- Referenzielle Transparenz: Da der einzige Effekt einer nebeneffektfreien Funktion ihr Rückgabewert ist, ist Sie leichter zu verstehen als eine Prozedur.
- Deklarative Sprache: Erlang ist eine Deklarative Sprache. Im vergleich zu imperativen Sprachen führt das zu kürzeren, leichter verständlichen Programmen.
- Leichtgewichtige Prozesse: Prozesse sind Teil der Erlang VM und nicht Teil des Betriebssystems. Dadurch sind Sie wesentlich leichtgewichtiger als OS Prozesse oder sogar Threads. Auf einem handelsüblichen Rechner ist es möglich mehrere hundertausend Erlang Prozesse laufen zu lassen. Damit ist Erlang optimal für ein asynchrones und ausfallsicheres System geeignet.

KONTRA:

- Performancekritischer, sequentieller Code: Durch die Indirektion der VM geht ein Teil der Performance der Hardware verloren. Obwohl der grösste Teil des Systems durch IO Operationen dominiert wird, gibt es kleine, performancekritische, sequentielle Codestücke (z.B. der routing Algorithmus). Hier könnte man durch eine Hardwarenahe Sprache wie C eine verbesserte Antwortzeit erreichen.

Datenbank

Als Datenbank werden mehrere ets Tabellen verwendet.

- http://www.erlang.org/doc/man/ets.html

PRO:

- Einfache API: Da ets Tabellen einfache Key-Value Stores sind, sind Sie sehr einfach zu benutzen.
- Erlang Modul: ets Tabellen sind Teil des Erlang Systems. Damit sind Sie direkt über Erlang Syntax verwendbar. Darüber hinaus werden die Abhängigkeiten des Systems nicht unnötig erweitert.
- In-Memory Datenbank: ets Tabellen werden komplett im Hauptspeicher gehalten. Zugriffe auf die Tabellen sind damit sehr schnell.

KONTRA:

- Speicherverbrauch: Da alle Tabellen komplett im Hautspeicher gehalten werden, verbrauchen diese viel Platz. Es ist somit nicht möglich extrem grosse .osm Dateien einzulesen.

Algorithmus

Zur Berechnung der kürzesten Route zwischen zwei Knoten wird der A-Star Algorithmus verwendet.

PRO:

- Einfach A-Star ist leicht zu implementieren. Sämtliche in vergleichbaren Systemen implementierten Algorithmen basieren auf A-Star mit zusätzlichen Optimierungen.

KONTRA:

- Speicherbedarf Da alle besuchten Knoten in einer Liste gahelten werden, benötigt A-Star im worst-case sehr viel Speicher.

HTTP API

Das gesamte System wird über eine HTTP Schnittstelle angesprochen.

PRO:

- Universelle API: Sowohl lokale Anfragen als auch Netzwerkanfragen können über eine API getätigt werden.
- Einfache API: HTTP Anfragen sind sehr einfach zu stellen. Es existiert zu jeder populären Sprache ein entsprechendes Modul.

KONTRA:

- Nichts (bei der Art des Systems).

Performanz-Tests

Ausgewertet wurden verschiedene Metriken von drei Routen unterschiedlicher Länge:

Metrik	Rohrbach - Uni	OMZ Uni - Mensa Uni	OMZ Uni - Mannheim
Zeit	82 ms	190 ms	1520 ms
Nodes in Ergebnis	20	25	71
Pfadlänge	460 m	4151 m	17 866 m
Besuchte Nodes in			

Algorithmus	451	2550	15816
Speicherbedarf	44 k words	220 k words	1338 k words

Weiterentwicklung

Geochouch

Es bietet sich an, das Routing-System an GeoCouch anzuknüpfen. GeoCouch ermöglicht die Berechnung von Bounding-Box Anfragen, welche für eine weitere Verbesserung der Wegbeschreibung und einen fortgeschrittenen Name-Server nötig sind. Da GeoCouch/CouchDB ebenfalls in Erlang implementiert wurde, ließe sich das Routing System sogar zu einem CouchDB-Plugin entwickeln. Das hätte den Vorteil einer schnelleren Daten-Zugriffszeit und würde das Deployment erleichtern. Ein mit GeoCouch gebundeltes Routing System bietet beste Vorraussetzungen zur Entwicklung von Routing-Webapps. Der Entwickler dieser Apps müsste lediglich JavaScript und HTML beherrschen, da die komplette Backend-Funktionalität von GeoCouch abgedeckt werden kann.

- https://github.com/couchbase/geocouch
- http://www.couchbase.org/

GeoJson

Sämtliche APIs des Routing-Systems sollten auf GeoJson standardisiert werden. GeoJson ist ein leichtgewichtiger Standard für Geo-Daten, der sich wachsender Beliebtheit in der Webentwicklung erfreut. Die Standardisierung würde die Verwendung vor allem von UI-libraries deutlich erleichtern.

- http://geojson.org/

Polymaps

Polymaps ist eine JavaScript library, die das Erstellen stark angepasster und dynamischer Karten ermöglicht. Als karthografische Ressourcen können OpenStreetMap, Google Maps, Bing und weitere Anbieter eingebunden werden. Die Anpassung und das Anzeigen zusätzlicher Informationen (Pfade, Lables, Statistiken, …) auf der Karte erfolgt über SVG und GeoJson. Es eignet

sich besonders Polymaps zusammen mit d3.js zu verwenden - eine weitere Javascript library, die das Erstellen datengetriebener Dokumente ermöglicht. Hinter Polymaps und d3.js stehen herraussragende Entwickler der Stanford Visualization Group und eine weitere Wartung der libraries ist sicher.

- http://polymaps.org/
- http://mbostock.github.com/d3/

Mitwirkende

- Johannes Auer
- Haykuhi Jaghinyan
- Mirko Kiefer