Systemarchitektur

Die folgende Ausarbeitung beinhaltet zwei der geforderten Artefakte:

- Architekturdiagramm
- Architekturbegründung

Da diese im Zusammenspiel sind, ist es unnötig diese als zwei Artefakte anzusehen, da die Architekturbegründung das Architekturdiagramm erläutert.
Alle hier ausgearbeiteten Konzeptideen können während der anderen Meilensteine noch abgeändert werden. Dies hängt davon ab, ob alles in der angegeben Zeit realisierbar ist.

1 Systemkomponenten

1.1 Client

Die Client Applikation beinhaltet eine Anwendungs- und Präsentationslogik. GPS Koordinaten werden auf dem Client abgerufen und zu einer Route berechnet. Über eine HTTP-Schnittstelle werden die ermittelten Daten an den Anwendungsserver für weitere Verarbeitungsroutinen gesendet.
Die bereitgestellten Serverdaten werden für die Präsentation verwendet.

1.2 Server

Der zu erstellende Server wird mit einer Datenbank verbunden. Diese kann, muss aber nicht, selbst auf dem Server liegen, sondern kann wegen Sicherheitsmaßnahmen dezentralisiert werden. Die dort abgelegten Daten enthalten, Namen, Adressen, Materialien, Mannschaften etc.. Diese können für weitere Zwecke weiterverarbeitet und an die Clients geschickt werden (Push Notification an die Client Applikation).

2 Paradigmenwahl

Es werden folgende Paradigmen für die Erstellung der Applikation festgelegt:

- Client-Server Paradigma [einfache Datenübertragung und Datenerhaltung]
- Publish / Subscribe Paradigma [dient der Kommunikation zwischen den Stellen]

Wenn sich im Entwicklungsprozess weitere Erfordernisse herauskristallisieren, kann sich die Paradigmenwahl gegebenenfalls ändern.

3 Plattformen

3.1 Client

Die Clientkomponenten Fahrer und Abzuholde werden als eine mobile Android Applikation entwickelt, da diese aktuell den größten Marktanteil im mobilen Bereich beanspruchen[1].

Um Eingaben und Zugehörigkeiten zu einer Mannschaft, so wie weitere Verknüpfungen zu einander leichter zu realisieren, wird die Desktopanwendung in HTML und JavaScript umgesetzt. So ist eine Plattformunabhängig für die Verwaltungsstelle gewährleistet.

3.2 Server

Die Serverseitige Implementierung wird mit NodeJS realisiert. NodeJS ist eine Open Source Entwicklung und basiert auf JavaScript. Zu brauchende Daten werden im JSON Format erzeugt und abgelegt. Dies ist geeignet, da somit JSON basierte REST Dienste verwendet werden können. Ein weiterer Aspekt der für die Verwendung für NodeJS spricht, ist die Einbindung von Modulen, die weitere Funktionalität einbringen können, wie beispielsweise asynchrone , nicht blockierende Übermittlung von Daten.

3.3 Middleware

Das Modul “express”, welches auf einen NodeJS Server installiert wird, dient als Vermittlungsschicht. Im Gegensatz zu dem Modul “connect”, welches statt “express” installiert werden könnte, beherbergt “express” weitaus mehrere Funktionen, die die Entwicklung komfortabler macht.

3.4 Datenhaltung

Für eine persistente Datenhaltung verwenden wir die dokumentenbasierte Datenbank mongoDB. Diese eignet sich bestens mit dem Umgang von JSON Objekten. Darüber hinaus bietet die mongoDB uneingeschränkte Skalierbarkeit, was bei der Entwicklung vorteilhafter ist.

---

[1] Statistik mobiler Betriebssysteme
http://de.statista.com/statistik/daten/studie/184332/umfrage/marktanteil-der-mobilen-betriebssysteme-in-deutschland-seit-2009/ [abgerufen am 24.04.2014]

4. Architekturdiagramm

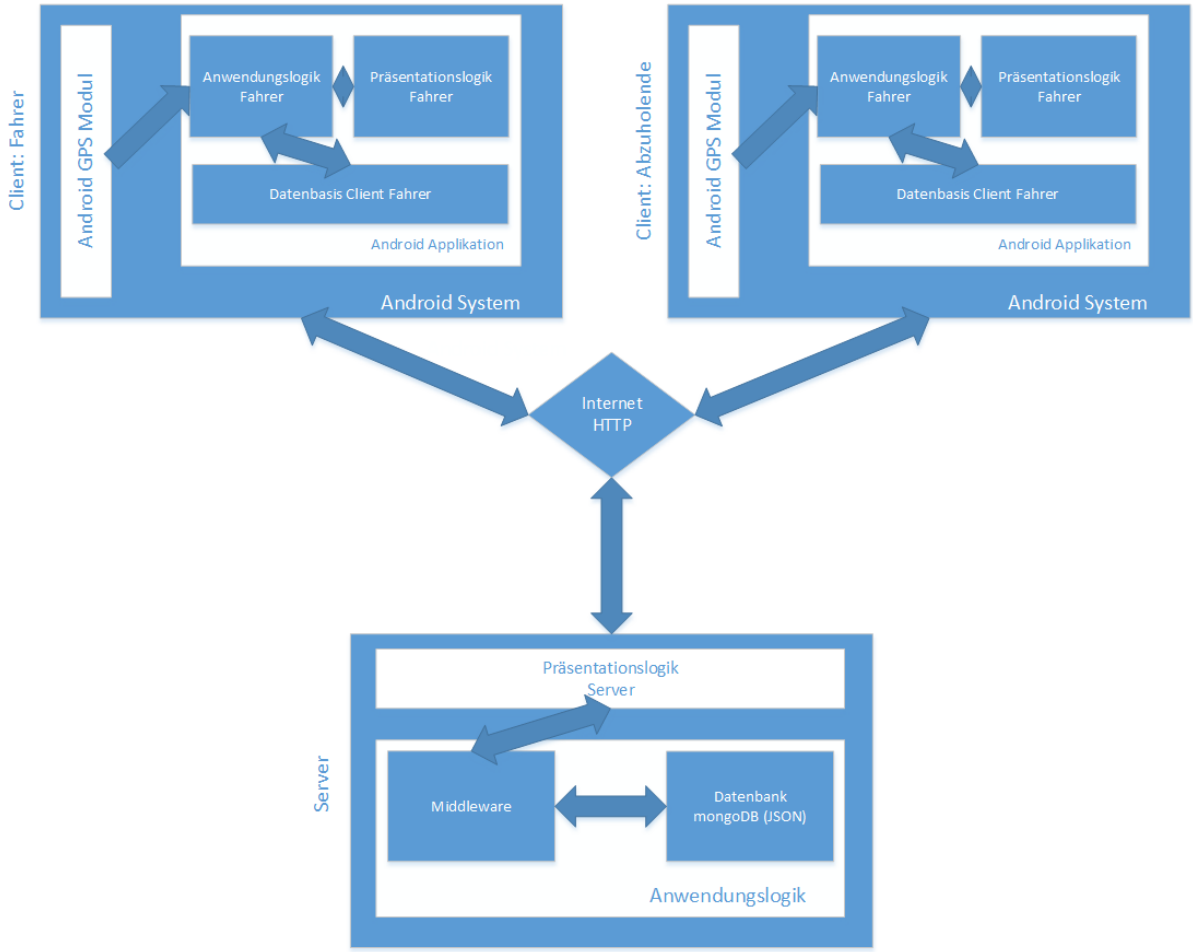

Abbildung 1: Architekturdiagramm

Die direktionalen Pfeile bilden die Kommunikation zwischen dem GPS Modul und der Anwendungslogik. Hier werden die GPS Koordinaten weitergeleitet.
Bei den bidirektionalen Pfeilen kommunizieren die Komponenten untereinander. Der Datenfluss geht somit in beide Richtungen.

Die Anwendungslogik “Fahrer” ermittelt die Position des Fahrers und berechnet eine Route.
Die Anwendungslogik “Abzuholender” ermittelt sich die Position des Abzuholenden und versendet sie.
Die Anwendungslogik “Server” errechnet und benachrichtigt die einzelnen Clients.