Systemarchitektur

Die folgende Ausarbeitung beinhaltet zwei der geforderten Artefakte:

- Architekturdiagramm
- Architekturbegründung

Alle hier ausgearbeiteten Konzeptideen können während der anderen Meilensteine noch abgeändert werden.

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 selbst auf dem Server liegen, oder auch 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 (z.B. Response Nachrichten an die Client Applikation).

1.3 Google Dienste

Damit mit den ermittelten GPS Koordinaten gearbeitet werden kann und die Fahrer zu den Mitfahrer navigiert werden können, wird die Google Map API benutzt. Diese ist leicht zu implementieren und ist darüber hinaus ein mächtiges Tool, mit der viele Implementierungen vereinfacht werden.

2 Paradigmenwahl

Es werden folgende Paradigmen für die Erstellung der Applikation festgelegt:

- Client-Server Paradigma (einfache Datenübertragung und Datenerhaltung)

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

3 Plattformen

3.1 Client

Die Clientkomponenten Fahrer und Mitfahrer 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, sowie 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 Servertechnologie muss folgende Punkte erfüllen:
Der Server muss..

- eine schnelle Performance liefern.
- Ereignisse selbstständig abarbeiten können und Änderungen an der Datenbank durchführen können
- eine hohe Anzahlen an Request gleichzeitig handhaben können
- skalierbar sein

Zusätzlich zu den oben genannten Punkten, sollte der Verwaltungsaufwand des Servers die Projektzeit nicht sprengen.

Nach momentanem Stand der Dinge ist der geeigneteste Kandidat für die Serverimplementation das NodeJS Modul. NodeJS ist eine ressourcenschonende Möglichkeit, Serverdienste mit hohen Request bereitzustellen. Darüber hinaus bietet es ein ereignisgesteuertes Input/Output Modell zu Verfügung, welches nur einen Aufruf benötigt.
Außerdem bietet es für die Datenhaltung eine breite Unterstützung von NoSQL Datenbanken. Mittels des „node package managers“ ist es möglich, weitere benötigte Module nachzuladen, was den Verwaltungs- und Wartungsaufwand minimiert.

3.3 Datenhaltung

Für eine persistente Datenhaltung verwenden wir die dokumentenbasierte Datenbank “MongoDB”. Diese eignet sich bestens mit dem Umgang von JSON Objekten und kann in eine JavaScript Serverumgebung eingebettet werden. Darüber

---

[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]

hinaus bietet die “MongoDB” uneingeschränkte Skalierbarkeit, Flexibilität bei der Datenwahl und eine hohe Performance, was bei der Entwicklung vorteilhaft ist.
Ein weiterer wichtiger Grund für die Wahl der “MongoDB” ist, dass aufgrund des hohen Datendurchsatzes der Nutzeranfragen eine möglichst performante und zuverlässige Technologie benötigt wird.

3.4 Datenformat

Da die Datenbank eine “MongoDB” Datenbank sein wird, fällt die Wahl auf das Datenformat JSON. Da die GPS Koordinaten ständig übermittelt werden, muss darauf geachtet werden, dass das Datenvolumen möglichst gering gehalten wird. Zu einem sollte das Datenvolumen des Benutzer nicht großartig verbraucht werden und anderseits sollte das Datenformat den Server bei hohen Serveranfragen nicht überlasten.
Als Alternative zu dem JSON Format wurde das XML gefunden. Jedoch erweißt es sich nicht als wirkliche Alternative, da es bei gleichem Informationsgehalt wie JSON ein höheres Datenvolumen aufweist.
Da es für die Programmiersprache Java einen JSON Parser gibt, ist somit gewährleistet, dass Server und Clientsystem mit dem Datenformat arbeiten können.

4. Architekturdiagramm

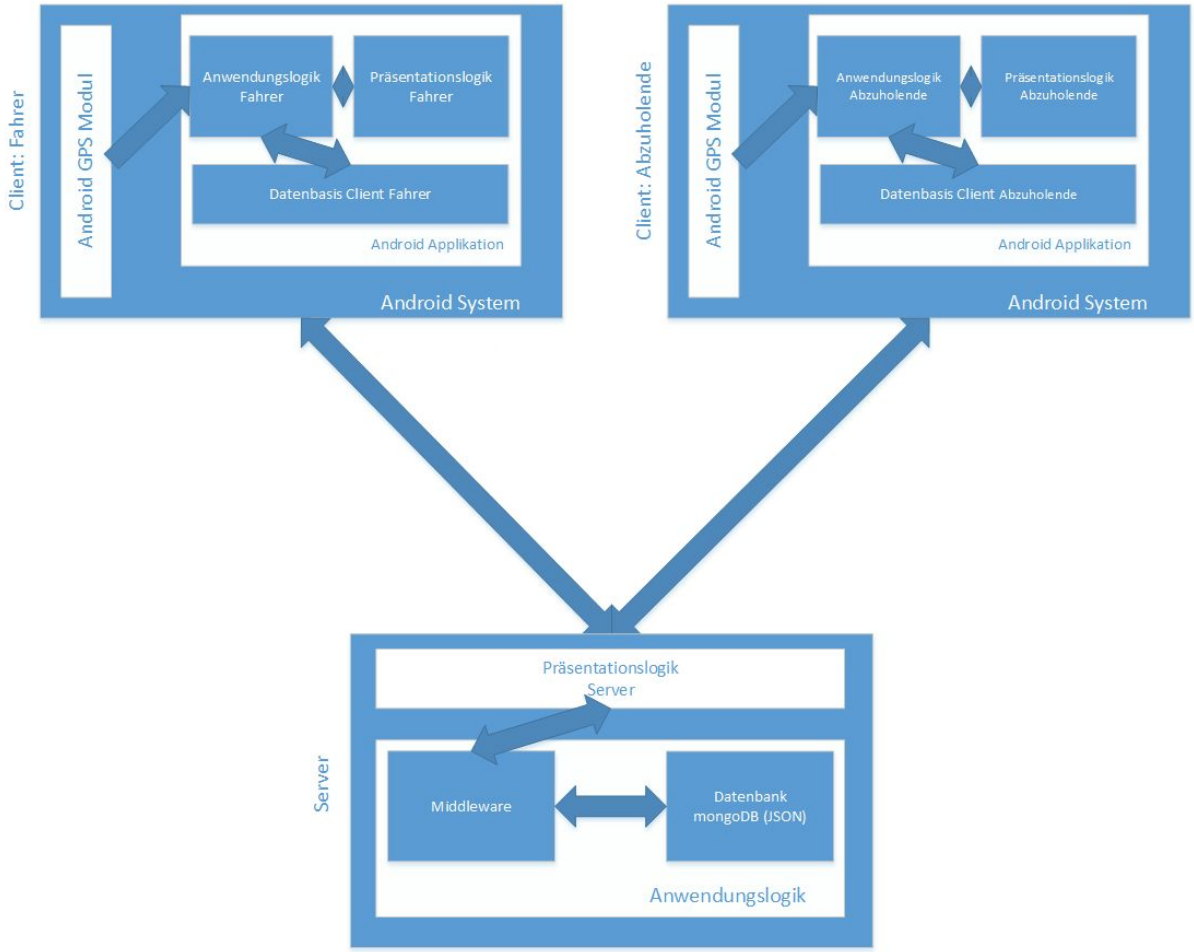

Abbildung 1: Architekturdiagramm

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.
Weitere Anwendungslogiken können bei der Entwicklung hinzukommen. Sie sind somit mit endgültig.