WBA Modellierung und Datenstrukturen

Das Architekturmodell, wie es im Meilenstein 2 dargestellt ist, wurde für den Meilenstein 4 beibehalten. Deshalb wird in diesem Part nur erläutert, welche Module miteinander agieren und welche Daten sie miteinander austauschen.
Für die Erstellung der Prototypen wurden drei Komponenten erstellt. Die PC Anwendung (Website), in der man seine Mannschaft und seine Events (Ziele) administrieren kann, der Server mit seiner Anwendungslogik und die TeamDrive APK, welche die Anwendungslogik für die Fahrer/ Abzuholenden implementiert hat.

Das Gesamtpaket besteht somit aus den folgenden Komponenten:

- PC Anwendung (Website)
- Server (NodeJS und MongoDB)
- TeamDrive APK

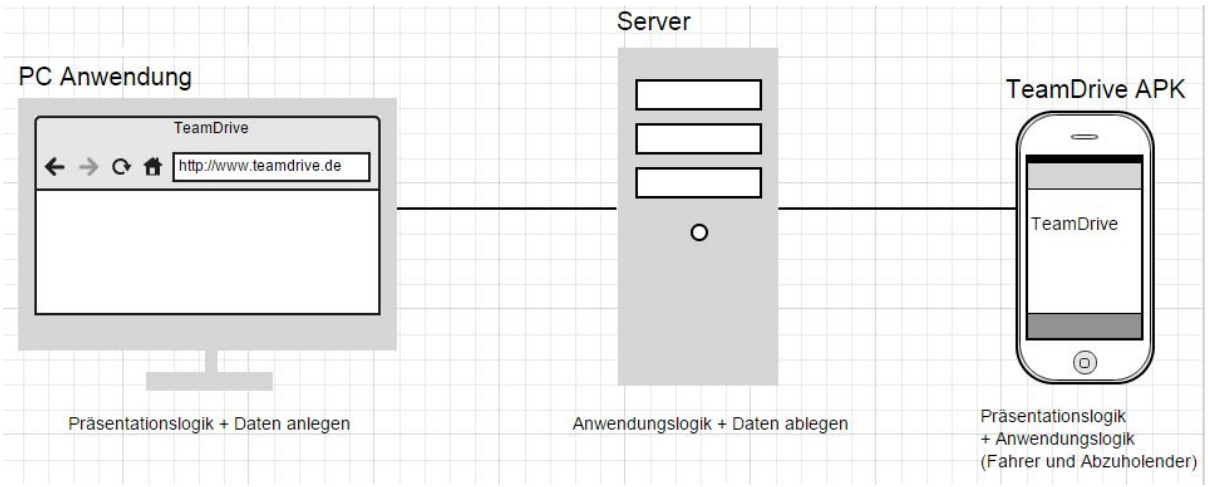

Abbildung 1: Logiken in den Komponenten

1. PC Anwendung (Website)
Diese Komponente wird von dem Trainer oder von einer Person, die die organisatorischen Angelegenheiten der Mannschaft betreuen will, benutzt.
Die PC Anwendung greift auf die Dateien des TeamDrive Servers zu. Diese sind neben dem Präsentationscode die Daten der erstellten Mannschaft (Spieler, Events, etc.)
Da es sich um das Abrufen von HTML Dateien und Abrufen/ Anlegen von Daten handelt, wird für die PC Anwendung keine Anwendungslogik implementiert. Diese Instanz dient einzig allein dem Anlegen und Verwalten der Daten.
Wenn man eine neue Mannschaft, einen Spieler oder einen Ziel/ Event ansehen (erstellen) möchte, so werden folgende Daten von der (in die) Datenbank abgerufen (gespeichert):

Person:

```
{
   "_id" : ObjectId,
   "per_benutzer" : String,
   "per_pw" : String,
   "per_mannschaft" : String,
   "per_name" : String,
   "per_vorname" : String,
   "per_email" : String,
   "per_tel" : String,
   "per_mobil" : String,
   "per_strasse" : "String,
   "per_stadt" : String,
   "per_status" : String,
   "per_auto" : Boolean,
   "per_sitzplaetze" : Number
}
```

Event:

```
{
   "_id" : ObjectId,
   "e_mannschaft" : String,
   "e_eventname" : String,
   "e_gegner" : String,
   "e_strasse" : String,
   "e_stadt" : String,
   "e_datum" : String,
   "e_uhrzeit" : String,
   "e_treffpunkt" : String
}
```

Mannschaft:

```
{
   "_id" : ObjectId,
   "man_name" : String,
   "man_strasse" : String,
   "man_stadt" : String
}
```

2. Server

Diese Komponente ist die Vermittlungsstelle für die beiden anderen Systeme. Sie beinhaltet einen NodeJS Server und eine dokumentenbasierte Datenbank (MongoDB).
Für die PC Anwendung (Website) liefert der Server den Präsentationscode und die gespeicherten Daten.
Für die APK liefert der Server nur die JSON Datenfür weitere weitere Operationen.
Darüber hinaus verwaltet und speichert der Server die Positionsdaten der einzelnen Anwender, damit diese an die Benutzer weitergeleitet werden können.

2.1 Ermittlung der Mitfahrer auf Fahrer

Diese Funktion errechnet automatisch anhand der GPS Daten die nahegelegensten Mitfahrer zu einem Fahrer. Diese orientiert sich an die Abholzeit der jeweiligen Abzuholenden.
Hierfür werden folgende Daten benötigt:

Fahrer:

```
{
   "_id" : ObjectId,
   "p_id" : ObjectId,
   "e_id" : ObjectId,
   "latitude" : Double,
   "longitude" : Double
}
```

Mitfahrer:

```
{
   "_id" : ObjectId,
   "p_id" : ObjectId,
   "e_id" : ObjectId,
   "latitude" : Double,
   "longitude" : Double
}
```

Person:

```
{
   "_id" : ObjectId,
   "per_mannschaft" : String,
   "per_name" : String,
   "per_vorname" : String,
}
```

Event:

```
{
   "_id" : ObjectId
}
```

Wenn alle Daten vorhanden sind, überprüft das System, wer als Fahrer eingetragen ist. Nachdem das ausgelesen wurde, überprüft das System, wie weit die einzelnen Spieler von jedem Fahrer zeitlich entfernt sind. Wenn dies ermittelt wurde, wird der Mitfahrer dem Auto des Fahrers zugeordnet. Hierfür gibt es folgende Tabelle:

Auto:

```
{
   "_id" : ObjectId,
   "f_id" : ObjectId,
   "m_id" : ObjectId,
   "e_id" : ObjectId
}
```

2.2 GPS Daten Verwaltung

Diese Funktion dient der Richtigkeit der Daten und als Backup, falls die GPS Übermittlung eines Abzuholenden nicht mehr funktioniert. Bei jeder GPS Übertragung eines Anwenders überprüft das System, ob die übertragenen GPS Daten vollständig oder leer sind. Damit die gültigen übertragenen GPS Koordinaten nicht von leeren oder korrupten GPS Koordinaten überschrieben werden, werden diese auf Gültigkeit überprüft und dann dem Fahrer übermittelt.
Fallunterscheidung:

- Sind die übermittelten GPS Daten leer oder korrupt, so wird der letzte gespeicherte Punkt an den Fahrer geschickt.
- Sind die übermittelten Daten gleich so werden die Daten nicht an den Fahrer übermittelt. Somit ist gewährleistet, dass das Datenvolumen nicht unnötig verbraucht wird.
- Sind die übermittelten Daten unterschiedlich, so werden die neuen Daten in der Datenbank abgespeichert und dem Fahrer übermittelt.

Die ersten übermittelten Daten werden aber zum Abgleich der neuen Daten immer abgespeichert.
Für die Überprüfung werden die GPS Positionen der Fahrer und Mitfahrer Collection benutzt.

3. TeamDrive APK

Diese Komponente wird von allen Anwendern verwendet - auch von dem Trainer oder dem organisatorischen Helfer. Sie ermöglicht es, sich am TeamDrive System anzumelden, sich als Fahrer oder Abzuholender für das nächste Event einzutragen und sich zu den einzelnen Spielern navigieren zu lassen und Nachrichten an die Fahrer zu schicken, die ihm vorgelesen werden.
Nur die oben genannten Funktionen sind in der APK realisiert. Die organisatorischen Funktionen, wie die an der PC Anwendung (Website), sind nicht mit implementiert.

3.1 Fahrer/ Mitfahrer Auswahl

Nach der Anmeldung am System ruft die APK die Personen und die Event Tabelle ab und speichert die angemeldete Person und das nächste Event auf dem Smartphone. Dies dient dazu, dass nicht ständig die Daten vom Server abgerufen werden müssen.
Somit werden folgende Daten auf dem Smartphone gespeichert:

Person:

```
{
   "_id" : ObjectId,
   "per_benutzer" : String,
   "per_pw" : String,
   "per_mannschaft" : String,
   "per_name" : String,
   "per_vorname" : String,
   "per_email" : String,
   "per_tel" : String,
   "per_mobil" : String,
   "per_strasse" : "String,
   "per_stadt" : String,
   "per_status" : String,
   "per_auto" : Boolean,
   "per_sitzplaetze" : Number
}
```

Event:

```
{
   "_id" : ObjectId,
   "e_mannschaft" : String,
   "e_eventname" : String,
   "e_gegner" : String,
   "e_strasse" : String,
   "e_stadt" : String,
```

```
    "e_datum" : String,
    "e_uhrzeit" : String,
    "e_treffpunkt" : String
}
```

Die Anwender müssen sich in dem Menü als Fahrer oder Mitfahrer eintragen. Dies passiert mittels eines Buttons. Nach einer Bestätigungsnachricht wird der ausgewählte Status der Person zu geteilt und dem Server übermittelt, welcher den Schritt in Punkt 2.1 "Ermittlung der Mitfahrer auf Fahrer" ausführt. Danach wird auf dem Home Screen der Applikation der Button für den ausgewählten Status aktiviert (Fahrer- oder Mitfahrermenü).
Ist aber die Person im System eingetragen, dass diese kein Auto besitzt, so wird diese Person von Anfang an als Mitfahrer eingetragen und die Option zur Auswahl, ob man Fahrer oder Mitfahrer ist, ist ausgegraut.
Wenn die Auswahl getroffen wurde, werden folgende Daten an den Server geschickt (abhängig ob Fahrer oder Mitfahrer):

Fahrer / Mitfahrer:

```
{
    "_id" : ObjectId,
    "p_id" : ObjectId,
    "e_id" : ObjectId
}
```

3.2 GPS Ermittlung
Da das System auf GPS Koordinaten basiert, müssen diese stets an den Server übermittelt werden. In diesem Fall ist es wichtig, dass die Daten so klein wie möglich gehalten werden. Es gibt drei Fallunterscheidungen:

- Sind die ermittelteten GPS Koordinaten ungleich den aktuellen, so werden die Daten geschickt.
- Ist die Aktualisierung der Daten länger als eine Minute, so werden die Daten ermittelt und an den Server geschickt.

Die erste Ermittlung wird aber in jedem Fall an den Server geschickt, damit dieser die Funktion der GPS Daten Verwaltung in Punkt 2.2 ausführen kann.
Es werden somit folgende Daten an den Server übermittelt:

Fahrer / Mitfahrer:

```
{
   "_id" : ObjectId,
   "p_id" : ObjectId,
   "e_id" : ObjectId,
   "latitude" : Double,
   "longitude" : Double
}
```

3.3 Vorlese Funktion Fahrer

Da im Straßenverkehr das Führen eines Handy gesetzlich nicht erlaubt ist, gibt es für den Fahrer die Funktion, dass alle gesendeten Nachrichten, die über die TeamDrive APK im Mitfahrermenü verschickt werden, dem Fahrer vorgelesen werden. Es wird die Android Funktion Text-To-Speech aufgerufen. Diese Funktion liest den Namen der Sender vor, so wie die Nachricht. Tippt der Fahrer eine Nachricht an, so kann diese wieder vorgelesen werden. Eine Sprachsteuerung konnte zu dem Zeitpunkt nicht realisiert werden.
Folgende Daten werden für diese Funktion benötigt:

Nachricht:

```
{
   "_id" : ObjectId,
   "f_id" : ObjectId,
   "m_id" : ObjectId,
   "e_id" : ObjectId,
   "msg" : String
}
```

3.4 Nachricht an Fahrer

Falls der Abzuholende eine Nachricht an den Fahrer verschicken möchte, weil dieser sich an einer bestimmten Stelle befindet, so kann er eine Nachricht an den Fahrer schicken. Hierfür muss er nur die Nachricht in dem vordefinierten Formular schreiben und auf den Senden Button klicken. Der Fahrer ist im Vorfeld eingetragen. Es werden folgende Daten benötigt:

Nachricht:

```
{
   "_id" : ObjectId,
   "f_id" : ObjectId,
   "m_id" : ObjectId,
   "e_id" : ObjectId,
   "msg" : String
}
```

Weitere Anwendungslogiken entnehmen Sie der Kommentierung im Code oder der Implementationsabgabe.

4 Datenstrukturen

Aus der Modellierung der Anwendungslogik gehen somit folgende Daten hervor:

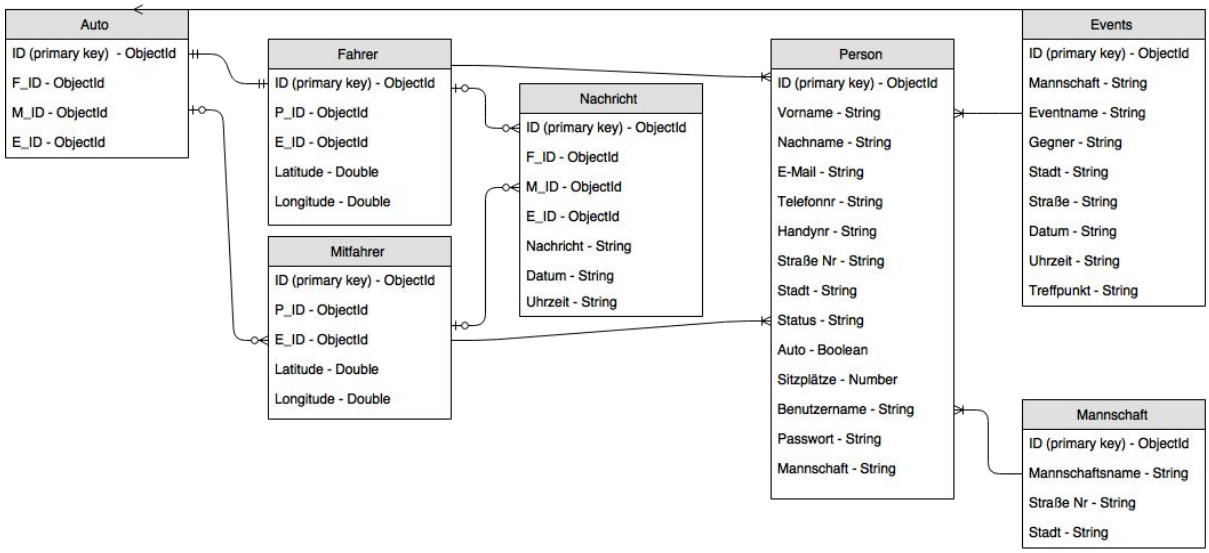

Abbildung 2: Datenstruktur TeamDrive

Sämtliche Daten können als atomare Datentypen gespeichert werden. Alle Dateninhalte werden als “string”, Nummern (GPS-Koordinaten) als “double”, IDs als “ObjectId” und Status als “boolean” gespeichert. Als Datenformat wird das dem JSON[1] ähnlichem BSON[2] gewählt, da die NO-SQL Datenbank MongoDB mit diesem Format arbeitet.

[1] The JavaScript Object Notation (JSON) Data Interchange Formt http://tools.ietf.org/html/rfc7159 [29.05.15]

[2] Universal Binary JSON Specification http://ubjson.org/ [29.05.15]; BSON http://bsonspec.org/ [29.05.15]