

Eingereicht von

Gruppe 3:

Matthias Winkler

Thomas Weberndorfer

Dominik Stadler

Angefertigt am

Service Engineering –

Software Engineering

Abgabe

08.05.2017

UE1/2-AUFGABEN: WEB- ANWENDUNG MIT AJAX- KOMMUNIKATION



MTD – Events

Gruppe 3: Veranstaltungs-Service

Hausübung

Inhaltsverzeichnis

1.	Einleitung	3
1.1.	Gruppe 3 – Veranstaltungs-Service	3
1.2.	MTD – Events	3
2.	Verwendete Tools	5
2.1.	MEAN-Stack	5
2.2.	MongoDB	5
2.3.	Node.js	5
2.4.	Express.js	6
2.5.	AngularJS	6
2.6.	Bootstrap	6
2.7.	W3School	6
3.	Planung	7
4.	Implementierungsschicht	7
4.1.	Backend	8
4.2.	Frontend	9
5.	Benutzeransicht/Webanwendung	11
6.	Probleme	14
6.1.	Datenbank	14
6.2.	Javascript / NodeJS	14
6.3.	Verständnis von Angular (MVC)	14
6.4.	Ranking	14
7.	Zusammenfassung	14

1. Einleitung

Ziel der Übung war es eine webbasierte Anwendung (eine oder wenige Views) mit einer asynchronen Client-Server-Kommunikation zu realisieren. Auf Client-seitig musste eine Bibliothek verwendet werden. Empfohlen wurden React, Angular oder JQuery. Server-seitig sollte ebenfalls eine JavaScript Lösung auf Basis von NodeJS realisiert werden. Bei der Datenhaltung konnte entschieden werden zwischen noSQL, mySQL oder Datei-basiert.

1.1. Gruppe 3 – Veranstaltungs-Service

Unsere Aufgabe war es eine dynamische Website zu entwickeln, die Veranstaltungen verwaltet. Es sollte möglich sein, neue Veranstaltungen zu erstellen zu posten, zu bearbeiten, etc. Bei den Veranstaltungen sollte es sich um sportliche Ereignisse handeln, wie z.B. Fußball oder Ski fahren. Die Ergebnisse der einzelnen Veranstaltungen sollten dann in Tabellen veranschaulicht werden.

1.2. MTD – Events

Der Name MTD – Events besteht aus unseren Vornamen (Michael, Matthias und Dominik) und dem Wort Events. Nach einer gemeinsamen Internetrecherche haben wir uns entschieden, sich dem MEAN Stack Konzept zu orientieren. Als Entwicklungsumgebung haben wir uns für WebStorm entschieden, ausgetauscht haben wir uns via Facebook und TeamView und als gemeinsame Codeverwaltung haben wir sich für Git/Github entschieden.

■ Client (Frontend)

- ☐ AngularJS v1.6.4
- ☐ HTML5
- ☐ CSS3
- ☐ Bootstrap v3.3.7

■ Server (Backend)

- ☐ nodeJS
- ☐ Express Framework (für nodeJS)

■ Datenhaltung

- ☐ mongoDB

■ IDE

☐ WebStorm

■ Codesourceverwaltung

☐ Git

☐ Github

■ Kommunikation

☐ Facebook

☐ TeamView

2. Verwendete Tools

In diesem Kapitel wird kurz auf die oben genannten Tools eingegangen und erklärt.

2.1. MEAN-Stack

Der MEAN Stack ist eine freie und Open-Source verfügbarer Software Stack bestehend aus diesen vier Komponenten:

MongoDB

Express.js

AngularJS

Node.js

Der MEAN Stack wird verwendet um dynamische Website Applikation zu erstellen. Das Besondere an dem Stack ist, dass für alle vier Libraries JavaScript verwendet werden kann und, dass JavaScript für Frontend und für Backend verwendet werden kann.

2.2. MongoDB

MongoDB ist eine Schema-freie, dokumentenorientierte NoSQL-Datenbank und ist einer der beliebtesten NoSQL-Datenbank. Die Datenbank speichert Daten im BSON-Format (Binary JSON), das dem JSON-Format (JavaScript Object Notation) nachempfunden ist.

Der Grund, warum wir uns für MongoDB entschieden haben, ist einerseits aufgrund der freien Verfügbarkeit und andererseits man mit MongoDB sehr schnell dynamische Webseiten erstellen kann und von vielen Online Guides empfohlen wird.

2.3. Node.js

Node.js ist eine serverseitige Plattform zum Betrieb von Webservices. Wird das Projekt ausgeführt wird ein Seite mit der lokalen Adresse auf Port 3000 (localhost:3000) zur Verfügung gestellt, auf dem der Veranstaltungsserver läuft.

2.4. Express.js

Das serverseitige, auf JavaScript basierende Web-Applikation-Framework erweitert Node.js und vereinfacht das Entwickeln von Webanwendungen

2.5. AngularJS

AngularJS ist ein client-seitiges JavaScript Framework und wurde von Google entwickelt. Mit dem Framework können Single Page Webanwendungen (SPAs) entwickelt werden nach dem MVC Konzept. Angular ist frei verfügbar und kann mittels einem HTML Tag eingebunden werden. (es gibt mehrere Möglichkeiten AngularJS in ein Projekt einzubinden)

```
<script  
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script  
>
```

Die Strukturierung eines AngularJS Webclients erfolgt auf Basis von Modulen, View-Templates, Controllern, Scopes, Filtern und Providern bzw. Services.

2.6. Bootstrap

Bootstrap ist ein Open Source Framework welches auf CSS und HTML basiert. Mit Bootstrap kann verschieden Elemente wie Formulare, Tabellen und Container Objekte mit vorgefertigten Layout einfügen. Bootstrap ermöglicht somit eine schnelle Entwicklung des Frontenddesigns. Außerdem ist es sehr leicht anwendbar und es gibt unzählige Tutorials dazu.

2.7. W3School

Das Design der Webseite wurde von einem der frei-verfügbaren Templates von W3School übernommen.

Link: https://www.w3schools.com/bootstrap/bootstrap_templates.asp

3. Planung

Es wurde mit der Skizzierung des Frontends begonnen.

Aufbauend auf diese Skizze wurde das Datenmodell erstellt. Durch die Skizze konnten wir uns besser vorstellen welche Daten benötigt werden. Somit haben wir aus der Skizze unser Datenmodell erstellt.

```
name: String,
eventDate: { type: Date, default: Date.now },
type: String,
info: String,

matches:
[
  {
    team1: String,
    team2: String,
    result1: Number,
    result2: Number,
  }
],

points:
[
  {
    team: String,
    points: Number
  }
]
```

Danach wurden Frontend und Backend immer miteinander entwickelt. Dies war für uns ein großer Vorteil da das Backend sofort auf neue Anforderungen des Frontends angepasst werden konnte.

4. Implementierungsschicht

Der Quellcode gliedert sich in Frontend (AngularJS, HTML5) und Backend (NodeJS). Das Backend befindet sich in unserer Datenstruktur unter routes und das Datenmodell mit der Datenbankanbindung unter model. Das Frontend befindet sich im public Ordner. Hier befindet sich die Index Datei und die

app.js. In der app.js befinden sich die Abfragen an das Backend und ein Großteil der Programmlogik des Frontends.

4.1. Backend

Der Mittelteil zwischen Backend und Frontend befindet sich in der Datei app.js. Diese Datei enthält die Controller die aufgerufen werden, sobald auf eine Seite zugegriffen wird.

Im folgenden Code-Ausschnitt ist ein Beispiel zum Hinzufügen eines neuen Matches abgebildet. Über den Scope werden die Daten aus der View mitübertragen, in ein JSON umgewandelt und an das Backend per Post Request übertragen.

```
myApp.controller('controller_addMatch', ['$scope', '$http', 'myApp_Service', '$routeParams', '$location',
function($scope, $http, myApp_Service, $routeParams, $location) {

    // saves the id of the selected event into $scope.param
    $scope.param = $routeParams.param;

    $scope.team1 = "";
    $scope.team2 = "";
    $scope.result1 = "";
    $scope.result2 = "";
    $scope.error="";

    $scope.saveMatch = function () {

        console.log("Save new event: 1");
        var jsonTest = JSON.stringify({ team1: $scope.match.team1,
            team2: $scope.match.team2, result1: $scope.match.result1, result2: $scope.match.result2});
        console.log(jsonTest);

        $http.post('/events/' + $scope.param + '/matches', jsonTest).then(function (response) {
            console.log("Save new event: 2");
            console.log('RegSuc: ' + response.data);
            $location.path('showDetails/' + $scope.param);
        });
    };
}]);
```

Abbildung 1: app.js

Das Backend übernimmt dann die Weiterleitung der Daten an die Datenbank MongoDB mittels Mongoose. Mongoose stellt wiederum Funktionen wie push oder findOneAndUpdate zur Verfügung, um die Daten direkt zu übertragen. Folgendes Beispiel zeigt das verändern eines Matches mittels findOneAndUpdate:


```
mongoose.model('Event').findOneAndUpdate(
  {
    'matches._id': req.id,
  },
  {
    $set: {
      'matches.$.team1': team1,
      'matches.$.team2': team2,
      'matches.$.result1': result1,
      'matches.$.result2': result2
    }
  },
  function (err, event) {
    event.save(function (err) {
      if (err) {
        console.error('error: ' + err)
      } else {
        //Event has been created
        console.log('the sub-doc was removed');
      }
    });
  }
);
```

Abbildung 2: findOneAndUpdate

4.2. Frontend

Das Frontend besteht aus HTML-Dateien welche die Darstellung der Website erledigen. In diese HTML-Dateien wurde das Bootstrap Framework eingebunden.

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<link rel="stylesheet" href="stylesheets/style.css">
```

Abbildung 3: Frontend Bootstrap

In die Index Datei wird je nach Aufruf der richtige Inhalt geladen. Der Inhalt befindet sich im Ordner Pages. Standardmäßig wird allEvents.html auf der Index Seite angezeigt. Hier werden die Events dynamisch in die Website geladen.

```
<tr ng-repeat="event in events">
  <td><div ng-click="showDetailsPage(event._id)"></div></td>
  <td>{{event.name}}</td>
  <td>{{event.type}}</td>
  <td>{{event.eventDate}}</td>
  <td>{{event.info}}</td>
```

Abbildung 4: Frontend All Events

Durch ng-repeat (AngularJS) wird die Tabelle mit Events befüllt. Somit wird die Tabelle genau in dem gewünschten Ausmaß (Anzahl der Events) erstellt.

Ein weiterer Vorteil der durch AngularJS erzielt wird ist das je nach dem welches Event angezeigt werden soll können diverser Teile der Website aus bzw. eingeblendet werden.

```
<table ng-if="detailed_type == 'Fußball'">
  <tr>
    <th>Rang </th>
    <th>Team </th>
    <th>Punkte</th>
  </tr>
  <tr><td>nbsp</td></tr>
  <tr ng-repeat="point in detailed_points | orderBy:'-points'" >
    <td>{{index +1}}</td>
    <td>{{point.team}}</td>
    <td>{{point.points}}</td>
  </tr>
</table>

<table ng-if="detailed_type == 'Skifahren' || detailed_type == 'Formell'">
  <tr>
    <th>Rang </th>
    <td>nbsp</td>
    <th>Fahrer </th>
    <th>Zeit</th>
  </tr>
  <tr><td>nbsp</td></tr>
  <tr ng-repeat="match in detailed_matches | orderBy:'+team2'">
    <td>{{index +1}}</td>
    <td>nbsp</td>
    <td>{{match.team1}}</td>
    <td>nbsp</td>
    <td>{{match.team2}}</td>
  </tr>
</table>
```

Abbildung 5: Frontend ng-if

Ng-if ermöglicht es je nachdem welche Daten angezeigt werden wollen die diversen Teile aus und ein zu blenden. Somit können verschiedene Sportarten wie Fußball und Skifahren auf einer Seite angezeigt werden.

5. Benutzeransicht/Webanwendung

Die Startseite unsere Webanwendung zeigt einen Überblick über alle Events die erstellt wurden. Durch Klicken auf den Info Button erhält man die nötigen Details (Startreihenfolge, Matches) des einzelnen Events.

Logo




Home

Neues Event

Info

MTD - All Events

Neues Event

	Eventname	Sportart	Datum	Info
	Hobbyturnier Schiedberg	Fußball	2017-07-02T10:00:00.000Z	Hobbyturnier für Jedermann
	Skirennen Hochwurzen	Skifahren	2017-12-12T09:00:00.000Z	Skirennen U17
	Formel1 am Spielbergring	Formel1	2017-08-12T07:00:00.000Z	Das Legendäre Rennen

Impressum

Abbildung 6 Startseite All Events

Die Seite Event Details ist in 3 Teilbereiche gegliedert.

1. Linker Bereich – Event Details, Event bearbeiten, Event löschen

Hier werden nochmal alle Details des Events angezeigt. Das Event kann gelöscht bzw. die Daten können hier nochmal verändert werden.

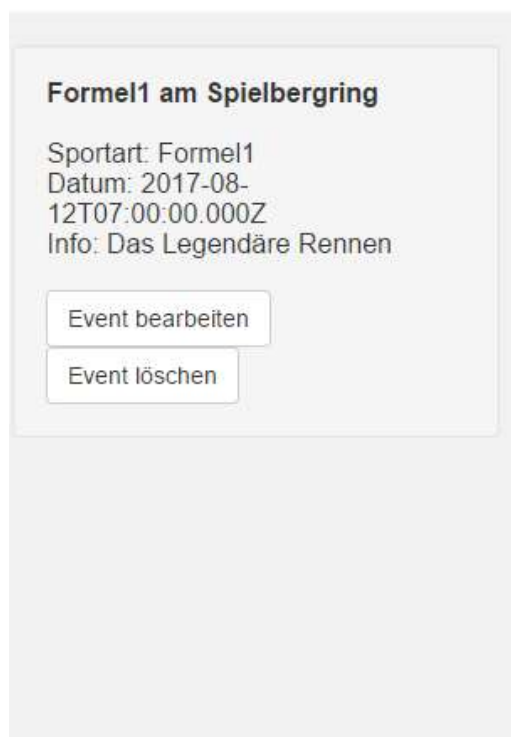


Abbildung 7: Überblick Event

2. Mittel Bereich – Übersicht der Matches/Teilnehmer

Im Mittelbereich werden die Fahrer hinzugefügt und angezeigt. Sie werden in Reihenfolge des Hinzufügens angezeigt.

MTD - Event Details

Fahrer	Startnummer	Fahrtzeit	
Sebastian Vettel	1	15,51	<button>Fahrer löschen</button>
Jensen Button	2	15,99	<button>Fahrer löschen</button>
Fernando Alonso	3	14,99	<button>Fahrer löschen</button>
<button>Fahrer hinzufügen</button>			

Abbildung 8: Überblick Fahrer

3. Rechter Bereich – Resultat

Hier erhält man einen Überblick über die Teilnehmer/Mannschaften sowie die Ergebnisse. Bei Sportarten wie Skifahren oder Formel1 werden die Ränge angezeigt bei Fußball die Tabelle mit den Punkte.

Überblick Fahrer		
Sebastian Vettel Jensen Button Fernando Alonso		
Rang	Fahrer	Zeit
1	Fernando Alonso	14,99
2	Sebastian Vettel	15,51
3	Jensen Button	15,99

Abbildung 9: Überblick Ergebnis

Neue Events können über den Button neue Events angelegt werden. Bei der Eventerstellung können nur Sportarten ausgewählt werden die implementiert wurden.

MTD - Create Event

Event Name:

Event Name

Sportart:

Datum:

tt.mm.jjjj --:--

Info:

Event Info

Event erzeugen

Abbildung 10: Create Event

6. Probleme

6.1. Datenbank

MongoDb verursachte einige Schwierigkeiten, weil der Datenzugriff im Vergleich zu klassischen Datenbanken anders erfolgt. Die Aufteilung von Daten in Dokumente anstatt von Tabellen macht redundante Daten notwendig. Beispielsweise werden die Teamnamen in den Matches sowie in den Punkten gespeichert.

Zusätzlich sind die Befehle von Mongoose in Javaskript für oft nicht ausreichend beschrieben.

6.2. Javascript / NodeJS

Die Konzepte von Javascript mit Post, Get, Delete, Put führt zu Verständnisproblemen bei Entwicklern, die Java gewöhnt sind. Put und Post können beide neue Daten hinzufügen. Der Aufruf dieser Operationen ist ungewohnt, da man mittels route eine ID sowie einen Zusatz an den Request anhängen muss, der als „String“ übergeben wird; anders als in Java, wo ein Code nicht ausgeführt werden kann, wenn es eine Operation nicht gibt.

6.3. Verständnis von Angular (MVC)

Das Problem bei AngularJS war das wir es zuerst wie normales JavaScript betrachtet haben. Leider haben wir dann die Weiterleitung falsch realisiert. Nach dem wir herausgefunden haben, dass das MVC Pattern bei Angular verwendet wird, viel die Implementierung Client Seitige Logik um einiges leichter. Seiten müssen registriert werden und es muss angegeben werden welcher Controller dafür zuständig ist.

6.4. Ranking

Die Event-Management soll verschiedene Sportarten erstellen können, die aber unterschiedliche Punkte- oder Platzierungsvergaben haben. Es ist eine Herausforderung mit einem Datenmodell mehrere verschiedene Events zu verarbeiten, ohne Ausnahmelogiken einbauen zu müssen.

7. Zusammenfassung

Die verwendeten Tools eignen sich gut für die Lösung dieses Projektes. Anfangs ist es schwierig von klassischen Programmen auf Services umzusteigen und den MEAN-Stack anzuwenden.

Die Gruppe zerfiel leider trotz erneuter Aufteilung durch die LVA-Leiter von ehemals fünf auf nur mehr drei Mitglieder. Dadurch stieg der Aufwand pro Person erheblich. Die drei verbleibenden Mitglieder haben sich durch Online-Medien und Kollaborationsplattformen gut abgestimmt und mit ihren individuellen Fähigkeiten gut ergänzt.

Das Projekt ist funktionsfähig. Die wenigen Anforderungen wurden umgesetzt und um Features, wie ein automatisches Ranking-System, ergänzt.

Abbildungsverzeichnis

Abbildung 1: app.js	8
Abbildung 2: findOneAndUpdate	9
Abbildung 3: Frontend Bootstrap	9
Abbildung 4: Frontend All Events	9
Abbildung 5: Frontend ng-if	10
Abbildung 6 Startseite All Events	11
Abbildung 7: Überblick Event	11
Abbildung 8: Überblick Fahrer	12
Abbildung 9: Überblick Ergebnis	12
Abbildung 10: Create Event	13