

Teil IV.
MEILENSTEIN 4

6.3. Server

Auf dem Server sollen verschiedene Aufgaben durchgeführt werden, unter anderem das Matchmaking. Die Abwägung des Servers fiel auf die Implementierung von NodeJS[11]. NodeJS bietet gegenüber PHP[13] schnellere und performante Ausführung, weniger Arbeitsspeichernutzung und ermöglicht zudem das Verwenden von gemeinsamen Code auf der Client und Serverseite. Dies ist eine Plattform, basierend auf JavaScript-Laufzeitbibliotheken, welches einen schnellen Bau von Netzwerkanwendungen erlaubt. Ein weiterer Aspekt ist die asynchrone Kommunikation bei Ein- und Ausgabe der Daten. Im aktuellen Projekt werden die Daten über die REST-Grundprinzipien[28] versendet und empfangen. Hier kann die Anwendung POST und GET-Anfragen an die URI-Ressourcen des Servers versenden und empfangen. So werden beispielsweise bei der Registrierung die Daten Repräsentationsformat JSON abgelegt und anschließend wird die Client-Anwendung darüber informiert ob der Vorgang erfolgreich war oder nicht.

6.3.1. Datensicherheit und Datenschutz

Leider findet die derzeitige Kommunikation des Prototypen über das HTTP-Protokoll im Klartext statt, hier werden die Daten unverschlüsselt übertragen. In Zukunft sollte man jedoch die verschlüssle Form der Kommunikation durch HTTPS anstreben.

GoogleMaps ist ein Dienst des US-amerikanischen Unternehmens Google Inc und unterliegt nicht den deutschen Datenschutzbestimmungen. So wird beispielsweise die Ermittlung des eigenen Standorts mit der Funktion „Mein Standort“ durch die GoogleMaps API[5] abgefragt, wodurch die Positionsangabe durch Längen- und Breitengrad des GPS bestimmt wird. Leider werden die Informationen über die empfangenen Mobilfunknetze miteinbezogen.

6.3.2. Middleware

Die Middleware stellte eine Komponente dar, die neben der verteilten Anwendung auf verschiedenen Systemen laufen kann. Bei der Middleware fällt die Wahl auf das NodeJS Modul „express“. Dies ist ein kleines und flexibles NodeJS-Webframework, dessen Funktionsumfang sich mit ergänzenden Modulen je nach Bedarf erweitern lässt.

6.3.3. Vorläufiges Matchmaking (Pseudo-code)

```

1 function matchmaking () {
2     if (aktuelleGpsPosition < 10 km) &&
3         (benutzerAlter > -3 || benutzerAlter < 3) {
4             return Benutzer (spielerNamen());
5         } else {
6             return 'keine Spieler gefunden!';
7         }
8 }
```

Listing 1: Vorläufiges Matchmaking (Pseudo-code)

Unter dem Menüpunkt „Herausfordern“ könnte ein Matchmaking-Algorithmus auf dem Server im Hintergrund ausgeführt werden, sobald der Benutzer neue Spieler herausfordert. Es werden als erstes die Spieler angezeigt, die sich in der umliegenden Umgebung befinden, beispielsweise im Radius von 10 km. Anschließend wird das Alter der Benutzer berechnet und geschaut, ob die Benutzer 3 Jahre älter oder junger sind. Danach werden die Spielernamen als Vorschlag ausgegeben und können entsprechend ausgewählt werden. Natürlich sollte man auch weiterhin seine Gegenspieler selbst bestimmen können und beispielsweise in einer Suchliste auswählen.

6.4. Datenmodell

Für die persistente Datenhaltung wird die Datenbank MongoDB[8]“ verwendet. Diese bietet gegenüber den relationalen Datenbank (bspw. MySQL[9]) ein flexibles Datenmodell und erlaubt die Daten in JSON-Format abzuspeichern. Im aktuellen Projekt verwaltet die Datenbank vier Collections und ist über die jeweiligen ObjectID's verknüpft.

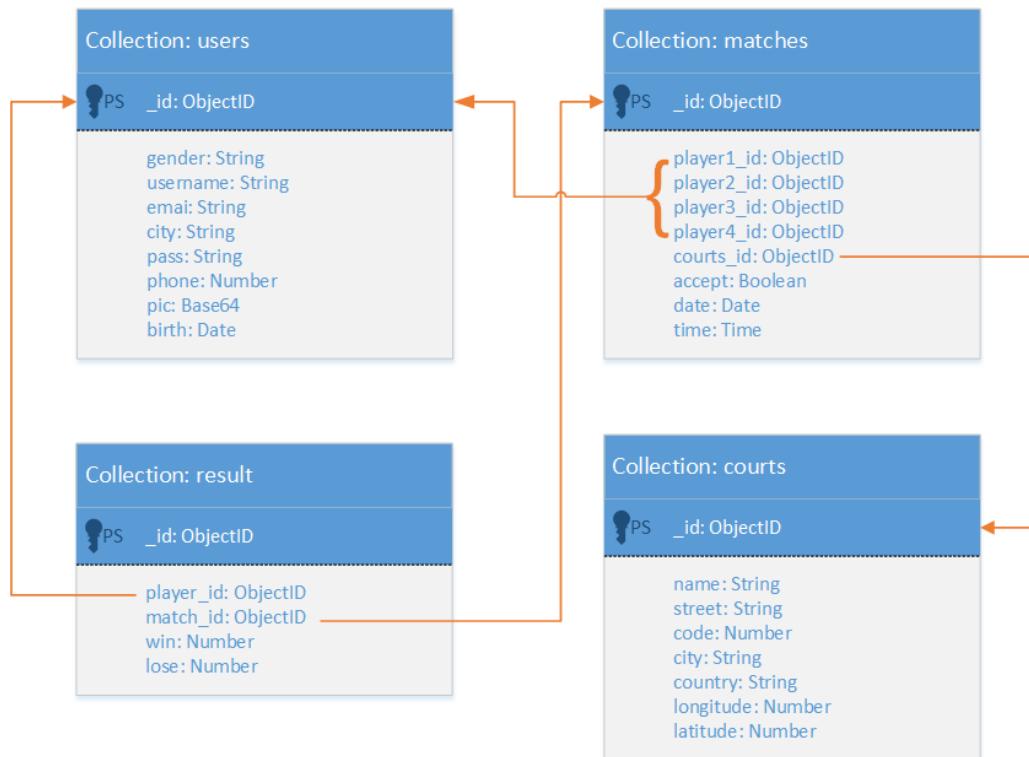


Abbildung 6: MongoDB Datenmodell

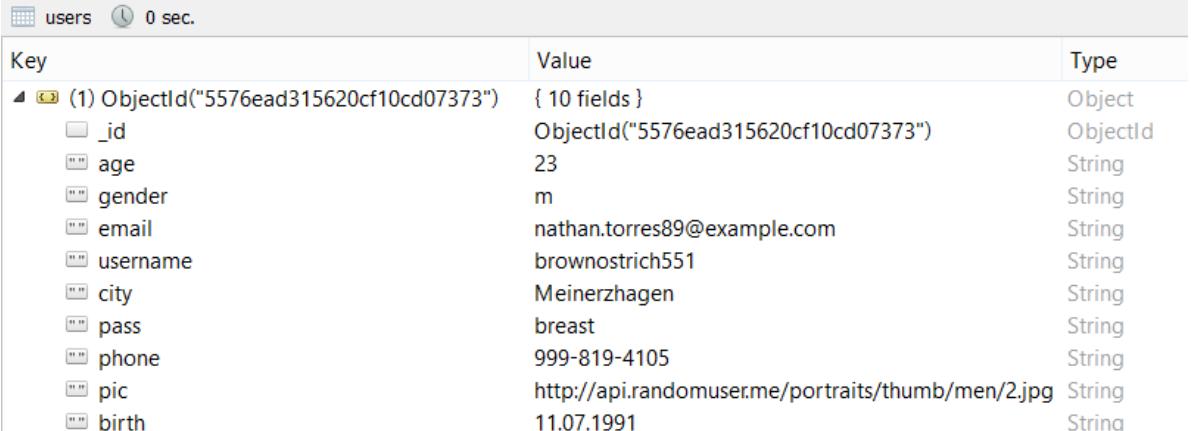
Vorerst wurden die Collections mit Beispieldaten (Mock-Up) gefüllt, damit später verschiedene Berechnungen durchgeführt werden können. Im folgenden werden die einzelnen URI's mit den jeweiligen Verben beschrieben, dabei wird der Payload in Form eines JSON-Objekts übertragen und ist hauptsächlich identisch mit der Datenstrukturen.

URI	Verben	Beschreibung
/users	GET	Benutzer abrufen
/users	POST	Benutzer erstellen
/users/:id	GET	Informationen zu einem Benutzer anzeigen
/users/:id	PUT	Benutzer bearbeiten
/users/:id	DELETE	Benutzer löschen
/users/playerresult	GET	Gesamtergebnisse zu allen Benutzern
/users/playserresult/:id	GET	Ergebnisse zu einem Benutzer
/users/login	POST	Benutzer Anmeldung
/users/chat	GET	Benutzer Chat (Textnachrichten)
/courts	GET	Basketballplätze abrufen
/courts	POST	Basketballplätze erstellen
/courts/:id	GET	Einen Basketballplatz abrufen
/courts/:id	PUT	Einen Basketballplatz bearbeiten
/courts/:id	DELETE	Einen Basketballplatz löschen
/matches	GET	Matches abrufen
/matches	POST	Match erstellen
/matches/:id	GET	Einen Match abrufen
/matches/:id	PUT	Einen Match bearbeiten
/matches/:id	DELETE	Einen Match löschen

Tabelle 6: Ressourcen mit HTTP-Verben

Users

Die Collection „users“ liefert die wichtigen Informationen zu den Benutzern.



Key	Value	Type
« 1 ObjectID("5576ead315620cf10cd07373")	{ 10 fields }	Object
_id	ObjectID("5576ead315620cf10cd07373")	ObjectID
age	23	String
gender	m	String
email	nathan.torres89@example.com	String
username	brownstrich551	String
city	Meinerzhagen	String
pass	breast	String
phone	999-819-4105	String
pic	http://api.randomuser.me/portraits/thumb/men/2.jpg	String
birth	11.07.1991	String

Abbildung 7: Datenstruktur der Benutzer

Courts

Basketballplätze, beinhaltetet Informationen über die Plätze und deren Standort.

Key	Value	Type
« ③ (1) ObjectId("5576f84815620cf10cd0737c")	{ 8 fields }	Object
└ _id	ObjectId("5576f84815620cf10cd0737c")	ObjectId
└ name	Lindengymnasium	String
└ street	Moltkestraße 50	String
└ code	51643	String
└ city	Gummerbach	String
└ country	Deutschland	String
└ longitude	7.570.996	String
└ latitude	51.032.045	String

Abbildung 8: Datenstruktur der Basketballplätze

Matches

Spielergebnisse, beinhaltet die gespielten Spiele mit den dazugehörigen Benutzern, Basketballplätzen sowie Datum und Uhrzeit.

Key	Value	Type
« ③ (1) ObjectId("5576fae315620cf10cd0738b")	{ 8 fields }	Object
└ _id	ObjectId("5576fae315620cf10cd0738b")	ObjectId
└ player1_id	5576ead315620cf10cd07373	String
└ player2_id	5576ead315620cf10cd07379	String
└ player3_id	5576ead315620cf10cd0731f	String
└ player4_id		String
└ courts_id	5576f84815620cf10cd0737c	String
└ date	09.06.2015	String
└ time	09:58	String

Abbildung 9: Datenstruktur der Matches

Result

Ist das Ergebnis der einzelnen Benutzer, dort stehen die Siege und die Niederlagen der einzelnen Matches.

Key	Value	Type
« ③ (1) ObjectId("557739f115620cf10cd073ee")	{ 5 fields }	Object
└ _id	ObjectId("557739f115620cf10cd073ee")	ObjectId
└ player_id	5576ead315620cf10cd07373	String
└ match_id	5576fae315620cf10cd0738b	String
└ win	2	String
└ lose	1	String

Abbildung 10: Datenstruktur der Ergebnisse

6.5. Endgeräte (Client)

Hierbei stellt sich die Frage zwischen den verschiedenen Betriebssystemen. Ein der beliebtesten auf dem Markt ist das Android OS von Google. Es hat zurzeit den größten Marktanteil gegenüber den anderen Systemen und wird deshalb für das Projekt priorisiert.

6.5.1. Android Anwendung

Man sollte so flexibel wie möglich sein und relative Größen definieren, um die verschiedenen Geräte mit verschiedenen Auflösungen abzudecken. Die Größen werden in dp (dip per inch) definiert, dies ist wichtig, da man die Elemente mit dem Finger anklickt und nicht wie etwa auf einem Computer mit der Maus. Finger haben einen bestimmten Durchmesser, deshalb gibt es seit neuesten Richtlinien[1] für die Metrics und Grids aus dem Hause Google.

6.5.2. Drittanbieter

Die Wetterinformationen werden von einem Drittanbieter „openweathermap.org“ abgerufen. Die Anfragen an die Schnittstelle erfolgen über ein einfaches REST Format. Als Ausgabeformat wird JSON mit geringen Overhead und damit geringen Übertragungszeiten unterstützt. Die Systemkommunikation zwischen den Drittanbieter soll von dem Server ausgehen, dass die Anwendungslogik zum Abrufen der Wetterinformationen auf der Benutzerseite stattfindet.

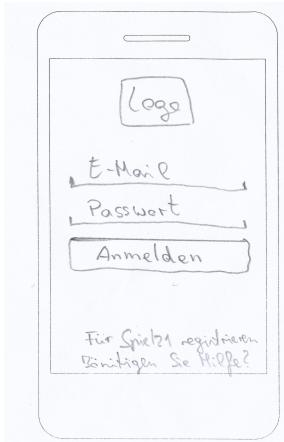
Das Kartenmaterial kommt von einem Drittanbieter GoogleMaps API[5]. Hier kann man für Endgeräte die Standordienste und Karten hinzufügen. Dies basiert auf einer JavaScript API und ist von Google gut auf der Webseite dokumentiert. Durch die GPS-Funktion wird der Standort ermittelt und auf Wunsch können Routen berechnet werden.

7. Prototypen UI

Nach dem die Anforderungen für das System erstellt wurden, folgen nun die vorläufigen Prototypen für das System.

7.1. Papierbasierte Prototypen

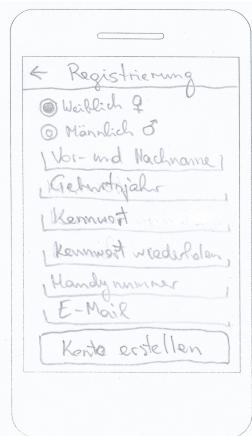
Bei einem papierbasierten Prototyp wird mit einfachen Mitteln nur grob eine Skizze gezeichnet und dient bei der Entwicklung neuer Produktkonzepte. Einer der wichtigen Vorteile ist, dass es keine Grenzen bei der Kreativität durch Technik oder sonstiges gesetzt werden. Hier wird der Fokus auf die vereinfachte Darstellung und leichte Zugänglichkeit für den Benutzer gelegt, so ist beispielsweise die beste Navigation die, die man nicht wahrnimmt. Die einzelnen Anforderungen, sowie die Normen werden hier in Abhängigkeit gezogen und in Iteration umgesetzt.



Start der Anwendung

Oben sollte das Logo des Spiels erscheinen, um speziell die Domäne zu erkennen. Zwei Optionen sollten auf dem Startbildschirm zu Verfügung gestellt werden. Die Anmeldung (Use Case 1) mit E-Mail und Passwort, dafür gibt es einen Button. Nach erfolgreichen Einloggen sollte der Benutzer zum Menü weiter geleitet werden. Ein anderer Punkt ist die Registrierung (Use Case 2), dafür gibt es einen Text zum Anklicken.

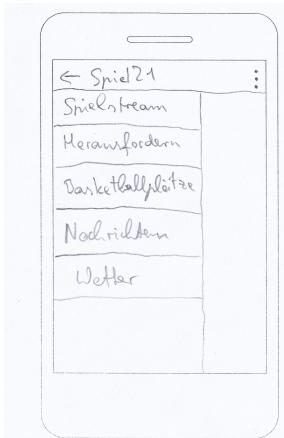
Abbildung 11: UC1



Registrierung des Benutzers

Bei der Registrierung werden die wichtigen Daten von den Benutzer abgefragt. Anschließend nach dem Ausfüllen und Validation aller Felder die Daten persistent in der Datenbank gespeichert. Um das Konto zu erstellen gibt es einen entsprechenden Button dafür. Nach erfolgreicher Registrierung sollte der Benutzer eingeloggt werden (Use Case 1) und zum Menü (Use Case 3) weiter geleitet werden.

Abbildung 12: UC2



Navigation und Menü

Nach dem erfolgreichen Einloggen sollte der Benutzer das Navigationsmenü zu sehen bekommen und darüber die wichtigen Menüpunkte (Use Case 4, 5, 7, 8, 9 und 12) ansteuern können.

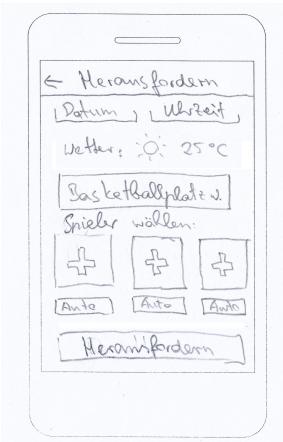
Abbildung 13: UC3



Spielstream

Im dem Menüpunkt Spielstreams hat der Benutzer die Ansicht mit den gespielten Spielen. Hier sind hierarchisch die gespielten Matches zu sehen zwischen den einzelnen Benutzer mit dem entsprechenden Spielergebnis und den Basketballplatz. Des Weiteren sollte oben eine ActionBar zu sehen sein um somit die Navigation zu erleichtern und um weitere Einstellungen (Use Case 10) vornehmen zu können

Abbildung 14: UC4



Herausforderungen

Die Überschrift zeigt immer wo sich zur Zeit der Benutzer befindet. Hier kann der Benutzer andere Spieler zu einem Match herausfordern, dabei kann er das Datum und die Uhrzeit bestimmen, daraufhin wird automatisch das passende Wetter für den spielenden Tag angezeigt. Weiterhin wählt er einen Basketballplatz (Use Case 7) und weitere Spieler (Use Case 6) aus. Das System kann die Spieler automatisch vorschlagen durch das Matchmaking.

Abbildung 15: UC5



Gegenspieler wählen

Im Menü Herausfordern kann der Benutzer manuell seinen Gegenspieler wählen. Durch das Plus-Symbol wird in einer Itemliste vorgeschlagen mit den registrierten Spielern im System, dort kann er seinen Gegenspieler auswählen und kehrt anschließend wieder in das Herausforderungen Menü (Use Case 5) zurück.

Abbildung 16: UC6



Basketballplatz auswählen

Im Menü Herausforderungen (Use Case 5) kann man auf einer Karte einen Basketballplatz auswählen, seinen eigenen Standort bestimmen, sich zu den Basketballplatz navigieren lassen und einen neuen hinzufügen. So kann hier beispielsweise ein Basketballplatz gewählt werden und man kehrt anschließend wieder in das Herausforderungen Menü (Use Case 5) zurück. Die Kartenansicht sollte über das Menü (Use Case 3) direkt ansteuerbar sein.

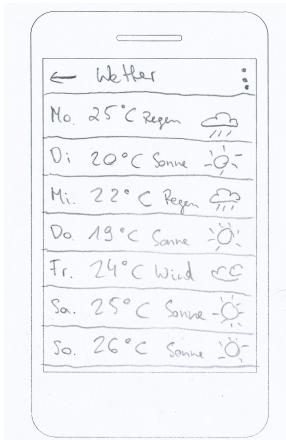
Abbildung 17: UC7



Nachrichten

Im Menü Nachrichten können die Benutzer verschiedenen Textinformationen austauschen und andere Benutzer kennen lernen. Hier kann beispielsweise ausgemacht werden, wer den Basketball zum Platz bringen soll.

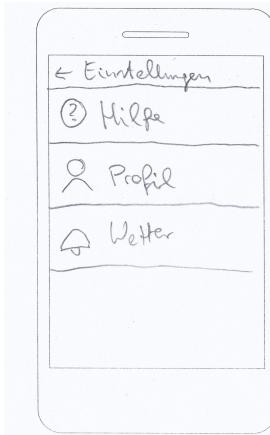
Abbildung 18: UC8



Wetter

Einer wohl der wichtigsten Nutzungsprobleme bei dem Spiel ist wohl die Wetterlage. Im Menü Wetter wird eine Vorhersage des Wetters bekannt gegeben und der Benutzer kann sich über die Wetterlage informieren und kann somit besser mit anderen Benutzern verabreden.

Abbildung 19: UC9



Einstellungen

Die Option Einstellungen wird nach den erfolgreichen Einloggen sichtbar und beinhaltet Auskunftsinformationen über das Spiel unter dem Punkt „Hilfe“, dort kann der Benutzer sich beispielsweise über die Regeln des Spiels informieren. Unten den Punkt „Profil“, kann der Benutzer seine registrierten Informationen im System einsehen. Letzter Punkt ist das „Wetter“, dort kann der Benutzer den Standort manuell bestimmen oder durch das GPS automatisch bestimmen lassen.

Abbildung 20: UC10



Popup Benachrichtigung

Die Popup Benachrichtigung taucht dann auf, wenn ein Benutzer einen anderen zum Spiel herausfordert (Use Case 5). Hier werden die Grundlegende Informationen (Name, Datum, Uhrzeit, Wetter) für den Benutzer gezeigt. Hier kann der Benutzer selbst die Entscheidung treffen, ob er die Herausforderung annimmt oder ablehnt.

Abbildung 21: UC11



Spielergebnis eintragen

Sollte es zu einer Herausforderung (Use Case 5) kommen und diese wurde angenommen (Use Case 11). So kann der Benutzer nach dem gespielten Spiel sein Spielergebnis eintragen, der Gegenspieler muss das gleiche tun. Das System verifiziert dann die Ergebnisse und zeigt diese später in den Spielstream (Use Case 3) an.

Abbildung 22: UC12