Teil III.
MEILENSTEIN 3

5. Spezifikation der PoCs

Bei Proof of Concept (Machbarkeitsnachweis) werden die Funktionalitäten des Systems geprüft und konkrete Bedingungen sowie konkrete Alternativen beschreiben und anschließen prototypisch umgesetzt.

5.1. Testen der Netzwerkstruktur

Risiko: Asynchrone Datenverarbeitung zwischen der Android-Anwendung und NodeJS-Server. Anschießend sollten die Daten in die Datenbank gespeichert werden, beispielsweise ein einfaches Login.
Ziel: Es sollte eine Android-Anwendung prototypisch geschrieben werden, welche mittels POST-Methode die JSON-Daten von der Anwendung an den Server sendet und diese dort in der Datenbank vergleicht, beispielsweise ein Login.
Folgende Schritte sollten ergriffen werden:

- Eine Android-Anwendung sollte erstellt werden.
- Die Anwendung sollte auf einem Testgerät ausführbar sein.
- Der Server sollte JSON-Daten empfangen und diese persistent speichern können.
- Allgemein: das Erlernen der Android Entwicklungsumgebung.

Exit / Fail Kriterium: Nur wenn der Ausbildungs- und und Lernaufwand in den Projektrahmen minimal gehalten werden und somit ein passables Ergebnis erzielt wird, gilt die Prüfung als bestanden.
Alternative Fallback: Leider führt hier kein Weg vorbei, man sollte sich die Zeit nehmen und sich mit den verschiedenen Entwicklungstools beschäftigen. Als Informatiker sollte man sich mit den neuen Technologien aneignen und sich weiter entwickeln.
Status: Die Entwicklungsumgebung „Android Studio“ wurde eingerichtet. Die ersten Schritte wurden ergriffen und es ist gelungen die Kommunikation zwischen den NodeJS-Server und der MongoDB-Datenbank herzustellen inklusive Datenaustausch. Anschließend wurde eine prototypische Android-Anwendung geschrieben und auf dem Testgerät erfolgreich getestet.

5.2. Nachrichtenaustausch (Chat Applikation)

Risiko: Es besteht zur Zeit keine Möglichkeit mit den Benutzern in Verbindung zu treten um unvorhersehbare Ereignisse zu diskutieren oder den Benutzer kennenzulernen. Man sollte die Netzwerkverbindung zwischen zwei Clients aufbauen.
Ziel: Schaffung offener Verbindung, die zum Nachrichtenaustausch zwischen den Benutzern dient, beispielsweise durch einen Chat.

Exit / Fail Kriterium: Nur wenn eine ständig offenen Verbindung zwischen den Client und einem anderen Client über den Server besteht und diese miteinander kommunizieren können, so gilt die Prüfung als bestanden. Ausnahme ist, wenn der Client nicht kommuniziert wie gewünscht mit dem Server und es findet kein Nachrichtenaustausch statt.
Alternative Fallback: Eine mögliche Alternative wäre eine Handynummer im Profil des Benutzers zu hinterlegen und somit die persönliche Kommunikation zwischen den Benutzern erlauben. Dies sollte dann als Pflichtfeldeingabe im Datenmodell benutzt werden.
Status: Es wurde eine Chat Anwendung mithilfe von NodeJS und Socket.io realisiert. Später sollte die Funktion in die Android-Anwendung implementiert werden.

5.3. Standort

Risiko: Die Benutzer verlieren das Interesse an dem System, wenn schon zu Beginn viele Abfragen gestellt werden, beispielsweise über den aktuellen Standort.
Ziel: Um die Benutzer nicht unnötig mit Standortfrage zu belästigen, sollte eine die Lokalisierung so einfach wie möglich gehalten werden. Mittels GPS sollte der Standort des Benutzers ermittelt werden.
Exit / Fail Kriterium: Nur wenn die GPS Position mittels der Android-Anwendung bestimmt werden kann, so gilt die Prüfung als bestanden, ansonsten greift man auf die Alternative zurück.
Alternative Fallback: Die Alternative sieht vor eine manuelle Eingabe des Standorts.
Status: Die GPS Position kann mithilfe des Android Gerätes ermittelt werden und steht für weitere Entwicklung zur Verfügung.

5.4. Wetter

Risiko: Es ist den Spielern auf den Basketballplatz nicht bewusst, dass es im Laufe des Tages regnen könnte.
Ziel: Wetterinformationen sollen die Benutzer vor Regen warnen.
Exit / Fail Kriterium: Wenn die Wetterinformationen bei den Benutzern in seiner Umgebung (manuelle Standorteingabe) korrekt als Warnungen ausgegeben werden können. Ausnahme ist eventuell die Anbindung der Schnittstelle bei den Drittanbietern.
Alternative Fallback: Es gibt mehrere Dienstleister am Markt die, die Wetterinformationen und die passenden APIs zur Verfügung stellen. Sollten die Wetterwarnungen nicht funktionieren kann man einfach die Wettervorhersagen für die nächsten zwei Wochen zur Verfügung stellen ohne die Warnungen.
Status: Das Vorgehen hat gezeigt, das die Daten durch einen HTTP-Request durchgeführt werden können. Hier wird auf den „openweathermap[12]“ Wetterdienst zugegriffen und nicht wie zuvor beschrieben auf „wetter.com“, da man sonst zwei Wochen warten müsste,

bis ein Antrag für die Schnittstelle bewilligt wird. Außerdem wird hier über XML und nicht über JSON kommuniziert. Die Warnungen im Regenfall konnten leider noch nicht umgesetzt werden, deshalb gilt der PoC als fehlgeschlagen. Als Alternative wurde die Ausgabe von den nächsten zwei Wochen zur Verfügung gestellt.

5.5. Matchmaking

Risiko: Ohne Matchmaking Algorithmus wird das System für den Benutzer uninteressant und bietet daher kaum einen Mehrwert in der Domäne.
Ziel: Implementierung der Bewertungsgrundlagen anhand der Spielergebnisse schaffen, dann mit Pseudodaten anreichern und testen. Matchmaking verknüpfen mit den Wetterinformationen und Basketballplätzen.
Exit / Fail Kriterium: Nur wenn der Algorithmus einwandfrei funktioniert und die Funktionalitäten der Benutzer bestmöglich erfüllt, so gilt es als erfolgreich. Ausnahme ist, wenn der Algorithmus für jeden Benutzer, der die gleichen Ergebnisse liefert oder der Algorithmus bezieht alle Ergebnisse der Benutzer mit ein, auch diejenigen die nicht in der Umgebung leben.
Alternative Fallback: Es gibt kaum eine Alternative dafür, außer man macht es in einer kleinen Domäne, wo jeder Jeden kennt.
Status: Das Matchmaking wird zu diesen Zeitpunkt nur als Pseudocode entwickelt, da hier das Testen der Netzwerkstruktur Vorrang hat und muss zuerst als erfolgreich abgeschlossen werden um später die Implementierung im System zu gestatten.

6. WBA-Modellierungen

6.1. Kommunikationsmodell

Die Anwendung wirkt als verteiltes System mit verschiedenen Teilkomponenten. Im Folgenden wird der Ablauf der Kommunikation genauer erläutert, ohne auf die technischen Details einzugehen.

6.1.1. Deskriptives Modell

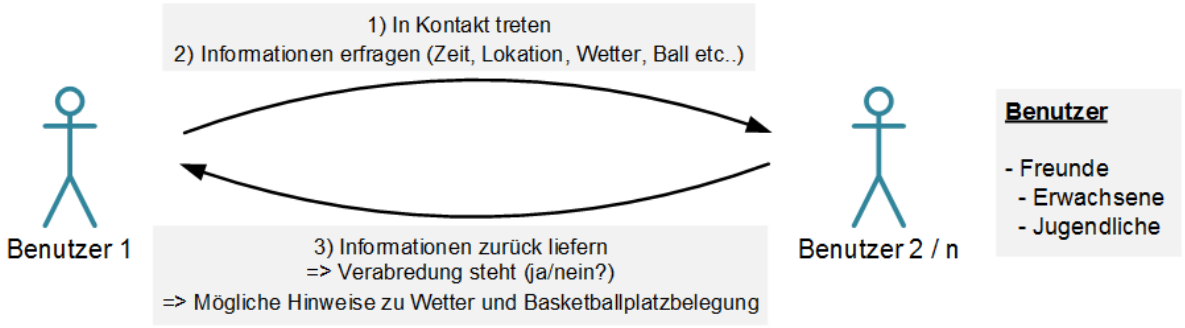

Abbildung 3: deskriptives Kommunikationsmodell

Die Abbildung 3 zeigt die momentane Situation zwischen den Benutzern. Derzeit verläuft die Kommunikation zum größten Teil über die Sprachebene, indem der Benutzer 1 mit dem Benutzer 2 in Kontakt tritt. Sie tauschen verschiedene Informationen untereinander aus. Beispielsweise wird erfragt:

- Wann und wo wird sich getroffen?
- Eventuell wird das momentane Wetter geprüft, regnet es?
- Wer bringt den Basketball zu den verabredeten Platz mit?

Des Weiteren bleibt jedoch die Problematik bestehen, ob ein Basketballplatz belegt ist oder nicht. Die Benutzer können verschiedene Personen sein, sind jedoch in den meisten Fällen nur Freunde aus der Umgebung die sich gut kennen.

6.1.2. Präskriptives Modell

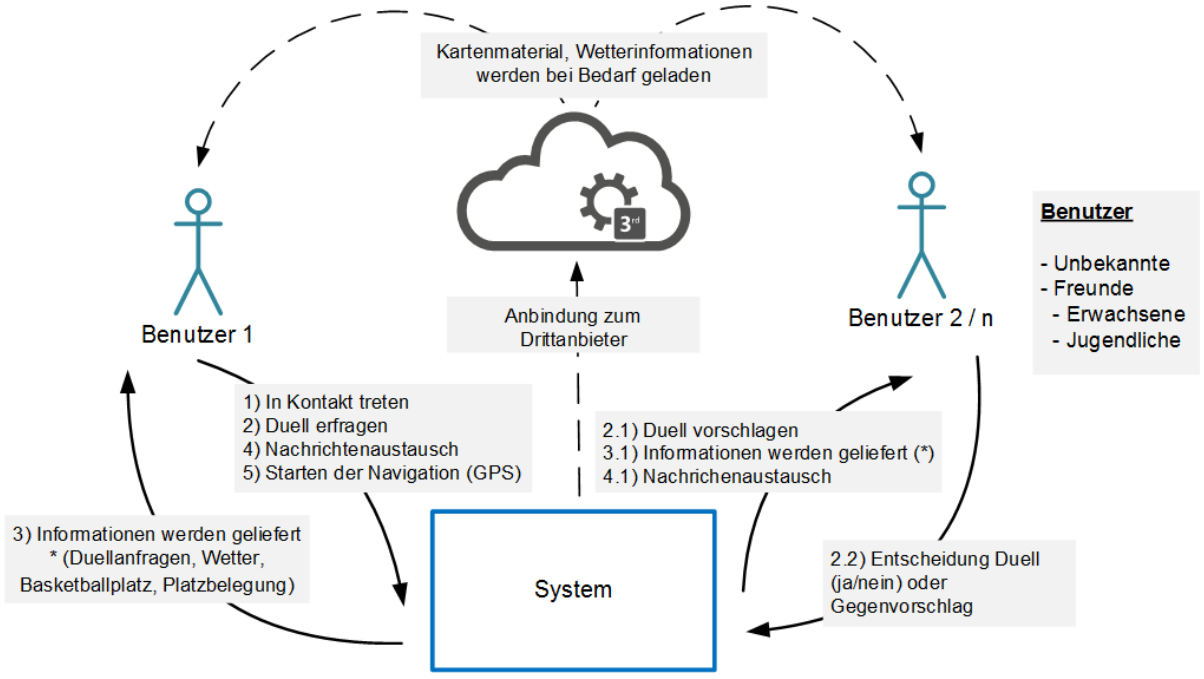

Abbildung 4: präskriptives Kommunikationsmodell

In der Zukunft sollte möglichst der größte Teil der Kommunikation über das System erfolgen (Abb. 4). Ein Benutzer kann in Kontakt mit dem System treten und nach einen Match anfragen. Daraufhin werden die möglichen Informationen, wie beispielsweise andere Matchanfragen, Wetterinformationen, Basketballplatz geliefert. Anhand der Informationen kann der Benutzer 1 mit anderen Benutzer(n), die sich vorher im System registriert haben in Kontakt treten. Hierbei findet ein Nachrichtenaustausch statt.
Sollte sich ein Match ergeben, so werden den Benutzer 2 möglichen Matches vorgeschlagen und die passenden Informationen geliefert, wie beispielsweise Datum, Uhrzeit, Ort, etc. Der Benutzer kann anhand dieser Informationen selbst entscheiden, ob man mit dem Vorschlag einverstanden ist oder einen Gegenvorschlag unterbreiten möchte.
Hierbei kann die Kommunikation zwischen den verschiedenen Benutzern in verschiedene Richtungen laufen. Das präskriptive Kommunikationsmodell sollte nur einen Fallbeispiel zeigen, wie so was in der Anwendung ablaufen könnte. Die Benutzer können verschiedene Personen sein, die man eventuell aus der Umgebung nicht kennt. Dadurch könnten sich neue Freundschaften und Gemeinschaften bilden.
Drittanbieter liefern die Wetterdaten und das Kartenmaterial, um Wettervorhersagen und Navigationsfunktionalitäten zur Verfügung zu stellen.

6.1.3. Architekturdiagramm

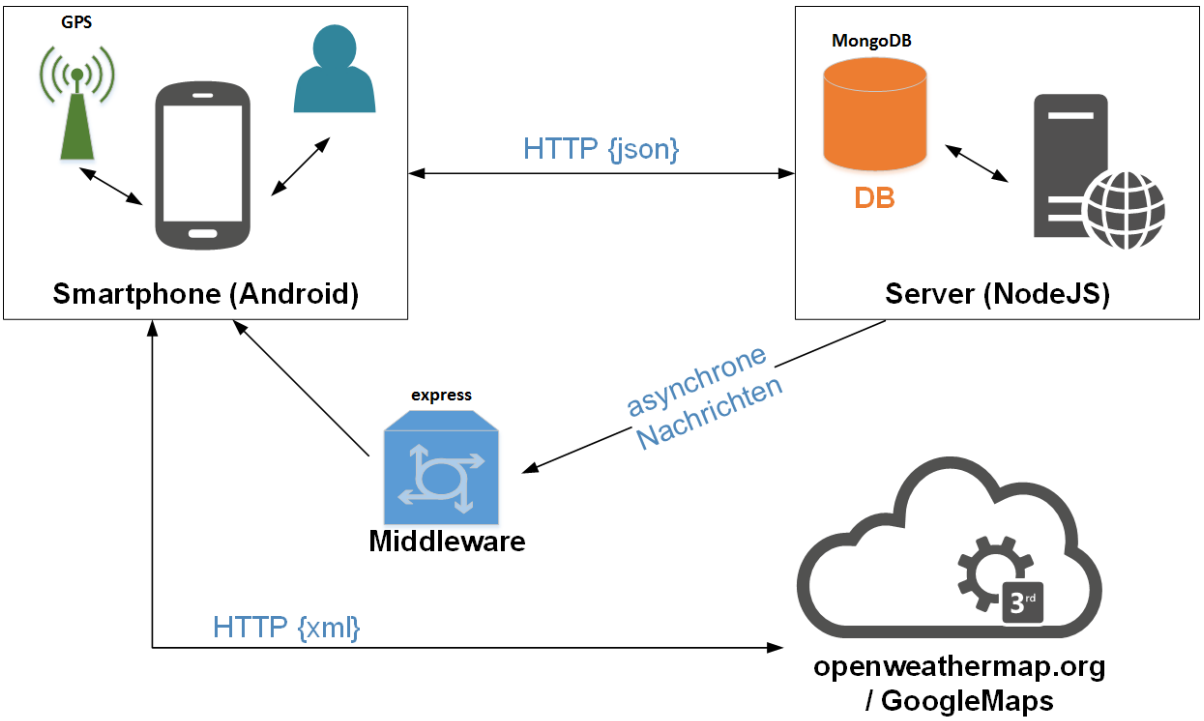

Abbildung 5: Systemarchitektur

6.2. Systemarchitektur

Die Systemarchitektur (Abb. 5) sollte über einen Server mit angebundener Datenbank kommunizieren. Der Client besteht aus einer Präsentations- und Anwendungslogik. Bei dem Endgerät spielt die GPS Funktion eine wichtige Rolle, um beispielsweise die Wetterinformationen für bestimmte Ortschaft abrufen zu können oder sich zum Basketballplatz navigieren zu lassen. Hierbei wird auf die meteorologischen Wetterinformationen von einem Drittanbieter „openweathermap.org[12]“ zugegriffen und über eine XML Struktur abgerufen. Außerdem sollte das System in der Lage sein, die Textnachrichten zwischen den Benutzern in einer asynchronen Verbindung auszutauschen, dies wurde bereits in den PoCs realisiert.

Obwohl es eine Menge voneinander unabhängiger Computer sind, so sollte es für den Benutzer als eine einzelnes zusammenwirkendes System erscheinen. Im weiteren Punkten werden die Abwägungen für die Systeme erläutert.