HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

drallo

Teamplay für drallo

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2015

Autoren:	Marcel Amsler, Daniel Kolb, Mathias Vetsch
Betreuung:	Olaf Zimmermann, IFS
Projektpartner:	Ursin Brunner, Challenge Earth AG
Arbeitsumfang:	8 ECTS bzw. 240 Arbeitsstunden pro Student
Arbeitsperiode:	18. September bis 18. Dezember 2015

Inhaltsverzeichnis

1. Abstract

Ziel dieser Arbeit war es, eine Multiplayer-Komponente für die bestehende Plattform “drallo” zu entwickeln. Mit drallo können bereits seit etwa zwei Jahren interaktive Trails und Schnitzeljagden auf dem Web erstellt und in einer App auf verschiedenen Smartphones gespielt werden. drallo wurde in dieser Arbeit so erweitert, dass Benutzer nun auch *miteinander* solche Trails absolvieren, dabei interagieren und Aufgaben gemeinsam lösen können.

Um die einzelnen User, die Teams und die Spielabläufe zu verwalten, wurde eine Serverkomponente entwickelt, die als eigenständiger Service in der Microsoft Azure Cloud läuft. Die Kommunikation zwischen Server und Clients ist bidirektional und asynchron mit Hilfe der SignalR-Library von Microsoft umgesetzt, welche sowohl Websockets als auch Long-Polling unterstützt.

Bei der Umsetzung stand die sinnvolle Integration in das bestehende System im Mittelpunkt. Um die Integration und Wartbarkeit der Anwendung zu vereinfachen, wurden wichtige Komponenten für den Server (ASP.NET) und die App (Xamarin) so umgesetzt, dass sie plattformunabhängig sind und somit untereinander geteilt werden können. Die drallo Plattform wurde ausserdem um einen Multiplayer-Editor erweitert, damit auch Multiplayer Spiele einfach und schnell erstellt werden können.

Als grösste Herausforderungen in der Planung stellten sich die Definition der Kommunikationsabläufe und die Wahl eines ausfallsicheren und zuverlässigen Kommunikationsframeworks heraus. Es hat sich gezeigt, dass einige Cross-Plattform-Lösungen, wie zum Beispiel Websockets, noch nicht ohne Weiteres auf allen Smartphones eingesetzt werden können. Trotz dieser Schwierigkeiten konnten wir alle Anforderungen des Industriepartners erfüllen und eine stabile Multiplayer-Komponente konzipieren, umsetzen und in die bestehende Plattform integrieren.

2. Management Summary

2.1. Ausgangslage

drallo ist eine bestehende Plattform, die interaktive Trails und standortbasierte Schnitzeljagden anbietet. Kunden von drallo können diese selbst erstellen und sie dann auf iOS- und Android-Smartphones zur Verfügung stellen. Bisher gab es auf den Apps keinerlei Interaktion zwischen den Spielern eines Trails, obwohl solche Schnitzeljagden vielfach als Gruppe absolviert werden.

Ziel dieser Arbeit war es deshalb, eine zusätzliche Multiplayer-Komponente zu konzipieren, zu entwickeln und in die bestehende Plattform zu integrieren. Die detaillierten Anforderungen haben wir zu Beginn der Arbeit erarbeitet und durch Gespräche mit dem Industriepartner und bestehenden Kunden von drallo validiert.

2.2. Vorgehen / Technologien

Zu Beginn wurde das Entwickeln eines Kooperationsspiels vom Industriepartner priorisiert, welches das Erreichen eines gemeinsamen Ziels durch das Erfüllen von individuellen Aufgaben vorsah. Zusätzlich sollten aber gewisse Aufgaben bei allen Spielern identisch sein, um beispielsweise alle Teammitglieder anfangs an einen Ort zu lotsen.

Um dieses Verhalten in das bestehende System integrieren zu können, haben wir eine neue Serverkomponente entwickelt, welche unabhängig von den bestehenden Systemen in der Microsoft Azure Cloud läuft. Diese Komponente verwaltet die Kommunikation zwischen den Smartphones und regelt den Spielablauf. Für die Plattform und die Mobile-App konnten bestehende Komponenten wiederverwendet und um die neu benötigte Funktionalität erweitert werden. Somit wurde die Benutzeroberfläche nur minimal verändert, was die Nutzung des neuen Features für bestehende drallo Kunden vereinfacht.

2.3. Ergebnisse

Auf Basis der erarbeiteten Anforderungen haben wir einen Prototyp entwickelt, getestet und dem Industriepartner präsentiert. Er besitzt die vom Industriepartner gewünschte Funktionalität und die nötige Stabilität, um zu Marktanalysezwecken eingesetzt zu werden. Es können nun neue Kombinationen von Aufgaben umgesetzt und sofort auf dem Smartphone getestet werden. Ausserdem bieten sich bereits mehrere Möglichkeiten, mit anderen Spielern zu interagieren. Sie können sich jederzeit auf der Karte finden, sowie dynamische Neuigkeiten über den Spielverlauf und die Aktivitäten anderer Spieler erhalten.

Teil I.

Technischer Bericht

3. Einleitung und Übersicht

Was ist drallo?

> *drallo ist ein Produkt der Challenge Earth AG. Das Technologieunternehmen mit Sitz in Zürich wurde 2012 gegründet. Entwickelt in Zusammenarbeit mit der Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) bietet Challenge Earth mit drallo eine Online-Plattform für Tourismus-, Sport- und Outdoor-Anbieter an. Die Entwicklung der App wurde durch ein Innovationsförderprogramm des Bundes (KTI Projekt Nr. 14277) und durch private Investoren finanziert.* - Challenge Earth AG

Die drallo Plattform bietet aktuell die Möglichkeit interaktive Trails und multimediale Schnitzeljagden, sogenannte drallos, zu erstellen und zu spielen. Bisher gab es auf den Apps keinerlei Interaktion zwischen den Spielern eines Trails, obwohl solche Schnitzeljagden vielfach als Gruppe absolviert werden.

Ziel dieser Arbeit war es deshalb, einen Prototyp für den neuen Multiplayer-Modus zu konzipieren, zu entwickeln und in die bestehende Applikation zu integrieren. Wichtig war es auch, dass die bestehenden Komponenten möglichst nicht verändert werden und die Erweiterung um den Multiplayer-Modus für allfällige Weiterentwicklungen seitens Industriepartner offen ist.

In der weiteren Dokumentation der Arbeit wird konkret auf die Planung der Architektur, die eingesetzten Technologien, die funktionalen und nicht-funktionalen Anforderungen, die aufgetretenen Probleme und deren Lösungsansätze und den daraus entstandenen Prototyp, eingegangen.

4. Ausgangslage

Drallo ist ein bestehendes Produkt. Dieses Kapitel erklärt Funktionen, technische Details sowie das Zusammenspiel der einzelnen Komponenten *vor der Arbeit*. Dieses Kapitel soll das bestehende System von der Arbeit abgrenzen und in die Aufgabenstellung einführen.

Die Abbildung 4.1 zeigt die Zusammenhänge zwischen den verschiedenen Tiers und bietet eine Übersicht der ganzen Plattform. Die in der Abbildung verwendeten Begriffe und Technologien werden in diesem Kapitel erläutert und ausführlich erklärt.

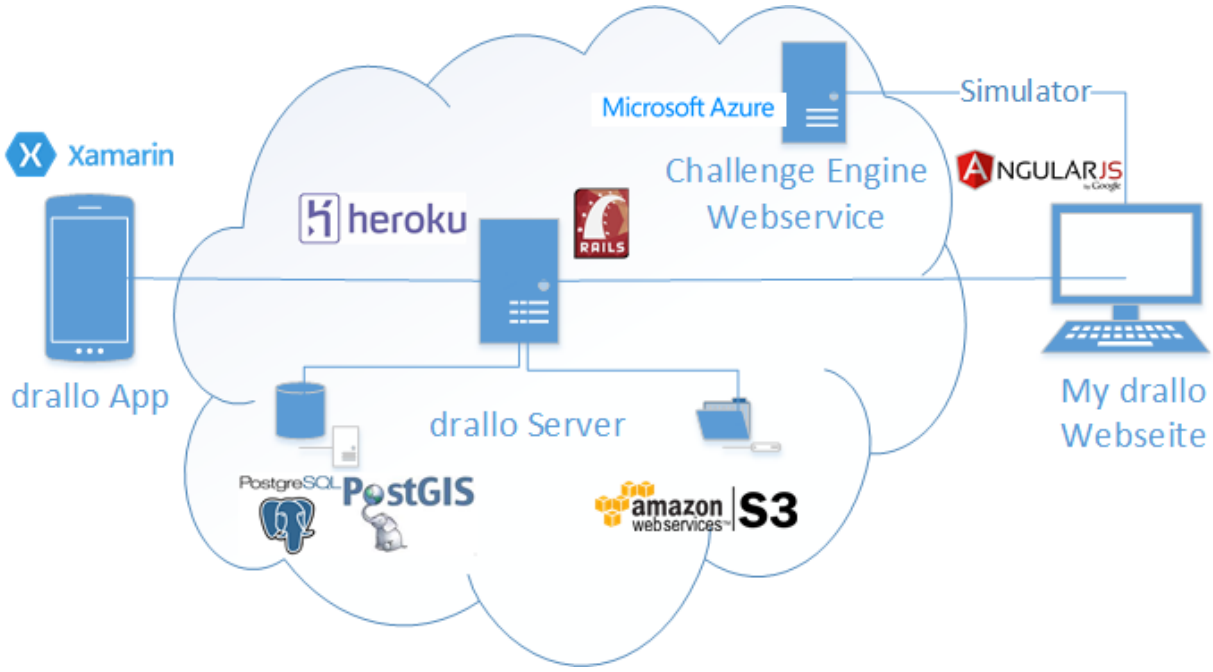

Abbildung 4.1.: Übersicht über die drallo Plattform

4.1. Komponenten

Die drallo App besteht aus verschiedenen Trails oder Schnitzeljagden, welche "drallos" genannt werden. Ein drallo kann aus mehreren Aufgaben bestehen und wird von einem "Promoter" erstellt. Im Gegensatz dazu ist es der "Player", der ein drallo auf seinem Smartphone spielt.

4.1.1. drallo Server

Der drallo Server bietet ein Frontend für Player, eine Verwaltung für Promotoren und eine API, die über RESTful HTTP für die mobilen Clients erreichbar ist.

4.1.2. my.drallo.ch

Die Webseite bietet für die Player Einsicht in das eigene Profil. Es können Informationen zu absolvierten drallos und gemachten Fotos abgerufen werden.

Promoters, sowie Administratoren können hier ihre Accounts verwalten, Teilnahme-Statistiken einsehen, sowie drallos mit dem Editor erstellen.

drallo Editor

Der drallo Editor wurde entwickelt, um den Promoters eine einfache Möglichkeit zu bieten, drallos zu erstellen. Es gibt bereits ein umfangreiches Set an *Conditions*, welche in drallos eingebaut werden können. Eine Condition ist eine Aufgabe innerhalb eines drallos. Es sind bisher folgende Typen verfügbar:

Area
Der Player muss sich zu einer geographischen Fläche begeben.
"Begib Dich nun in die Rapperswiler Altstadt, dort wartet eine spannende Aufgabe auf dich!"

Distance
Der Player muss eine bestimmte Distanz zurücklegen.
"Du wirst verfolgt! Geh Richtung Schloss Rapperswil, wo du Schutz finden kannst. Ich sag' dir, wann du weit genug weg bist."

Altitude
Eine gewisse Höhendifferenz muss überwunden werden.
"Die Stadt ist belagert. Um in Sicherheit zu sein, solltest du nun einige Meter den Schlosshügel hoch gehen."

Photo
Der Player macht ein Photo von einem bestimmten Objekt.
"Gut gemacht, geniesse jetzt die Aussicht und mach doch ein Foto für dein Fotoalbum als Erinnerung."

Speed
Die Geschwindigkeit des Players muss einen bestimmten Wert übersteigen.
"Leider ist es hier auch nicht mehr sicher. Renn so schnell du kannst wieder nach unten."

Location
Der Player muss sich zu einem geographischen Punkt begeben.
"Geschafft! Um die nächste Aufgabe lösen zu können, finde jetzt den Löwenbrunnen auf dem Fischmarktplatz in Rapperswil. Von dort aus geht's ruhiger weiter.'

QR-Tag
Ein QR-Tag[1] muss gescannt werden.
"Finde den QR Code beim Eingang des Museums Rapperswil."

Beacon

[1] Ein QR-Code (Quick Response Code) ist ein meist quadratisches Muster mit Feldern, die schwarz oder weiss sein können. So kann man z.B. eine URL codieren. QR-Codes gelten als robust, da eine Fehlerkorrektur beim Scannen möglich ist.

Der Player muss sich in die Nähe eines Bluetooth Beacons[2] begeben.
"Finde das berühmte Gemälde von Vincent van Gogh!"

Timer
Dies verzögert den Ablauf und kann genutzt werden um nach einer gewissen Zeit einen Tipp zu geben.
"Nicht gefunden? Ich gebe dir einen Tipp: Du findest es im Ausstellungsraum "Zeugen der Zeit.""

Information
Der Player konsumiert eine Information, er muss nichts beitragen.
"Dieses Bild wurde im Juni 1889 von Vincent van Gogh gemalt und heisst: The Starry Night. Den Wert dieses Bildes kann man nicht bestimmen, da es auf dem Markt nicht käuflich ist. "

Question
Eine Frage muss beantwortet werden.
"Nachdem ich dir nun von Vincent van Gogh erzählt habe: Wann hat er sein berühmtestes Werk gemalt? "

Condition-Kombinationsmöglichkeiten

Sequence
Die Conditions werden nacheinander abgearbeitet.

Circular Sequence
Eine Menge an Conditions, welche nacheinander erledigt werden müssen. Der Einstiegspunkt ist aber nicht definiert.

Parallel
Eine Menge von Conditions, welche in beliebiger Reihenfolge erledigt werden können.

Technologie

Die Webseite `my.drallo.ch` nutzt die folgenden Technologien:

- Ruby on Rails als Web-Framework
- AngularJS als JavaScript Framework
- CoffeeScript als Skriptsprache, welche in JavaScript transkompiliert wird
- Bootstrap als Layout Framework

4.1.3. drallo App

Applikation

Die Applikation bietet dem Player die Möglichkeit, alle drallos anzusehen, einzelne herunterzuladen und dann zu spielen.

[2] Beacons (dt. Leuchtfeuer) basieren auf einem Sender-/Empfänger-Prinzip. Ist ein Beacon in Betrieb genommen, sendet er regelmässig Signale. Begibt sich z.B. ein Smartphone mit einer Applikation, die Beacons verwendet, in diese Region, kann sie den Beacon an dessen UUID identifizieren und dementsprechend handeln.

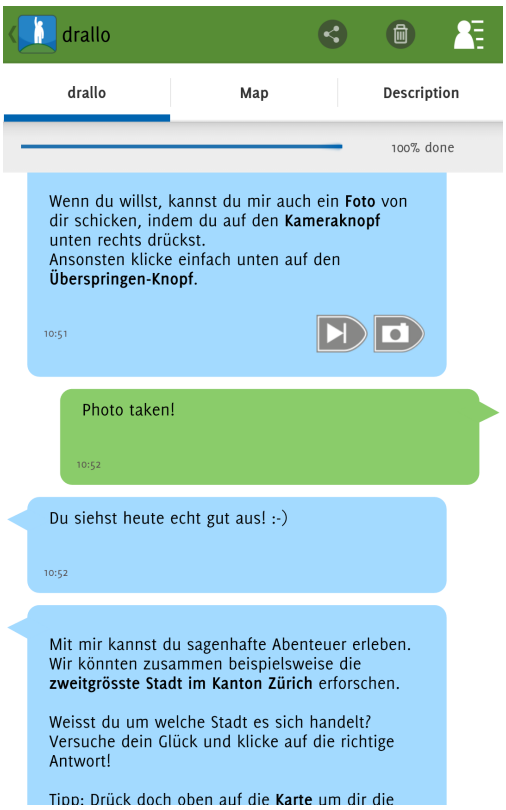

Abbildung 4.2.: drallo Timeline im Singleplayer Spiel

4.1.4. Simulator

Der Simulator ist ein Werkzeug für Promoters, um ihre drallos laufend testen zu können. Er simuliert einen mobilen Client im Browser. Die Challenge Engine läuft dafür hinter einem Webservice, welcher vom Simulator-Client angesprochen werden kann.

Auf Abbildung 4.3 ist zu sehen, dass eine Karte zur Verfügung steht um Positionsdaten zu generieren.

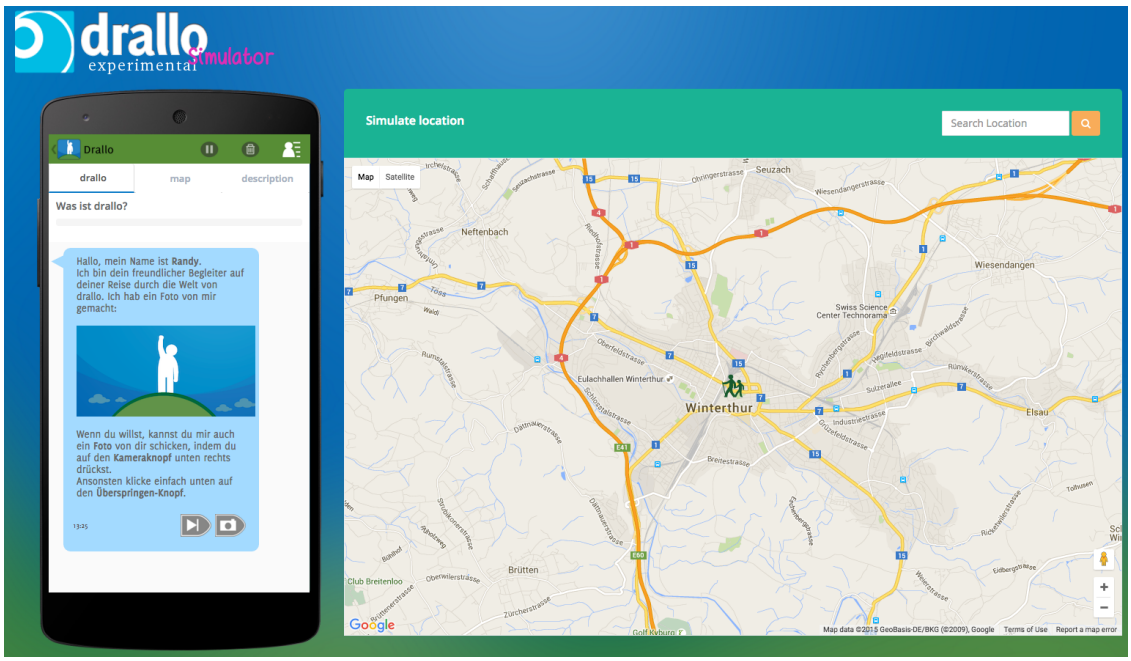

Abbildung 4.3.: drallo Simulator im Einsatz

4.2. System Kontext

drallo hat drei verschiedene Nutzer: Players, Promoters und Administratoren. Das System verwendet die Amazon S3 Image Storage um Bilder der Benutzer oder drallos anzubieten. Es gibt keine weiteren externen Schnittstellen. Diese Zusammenhänge werden in Abbildung 4.4 visualisiert.

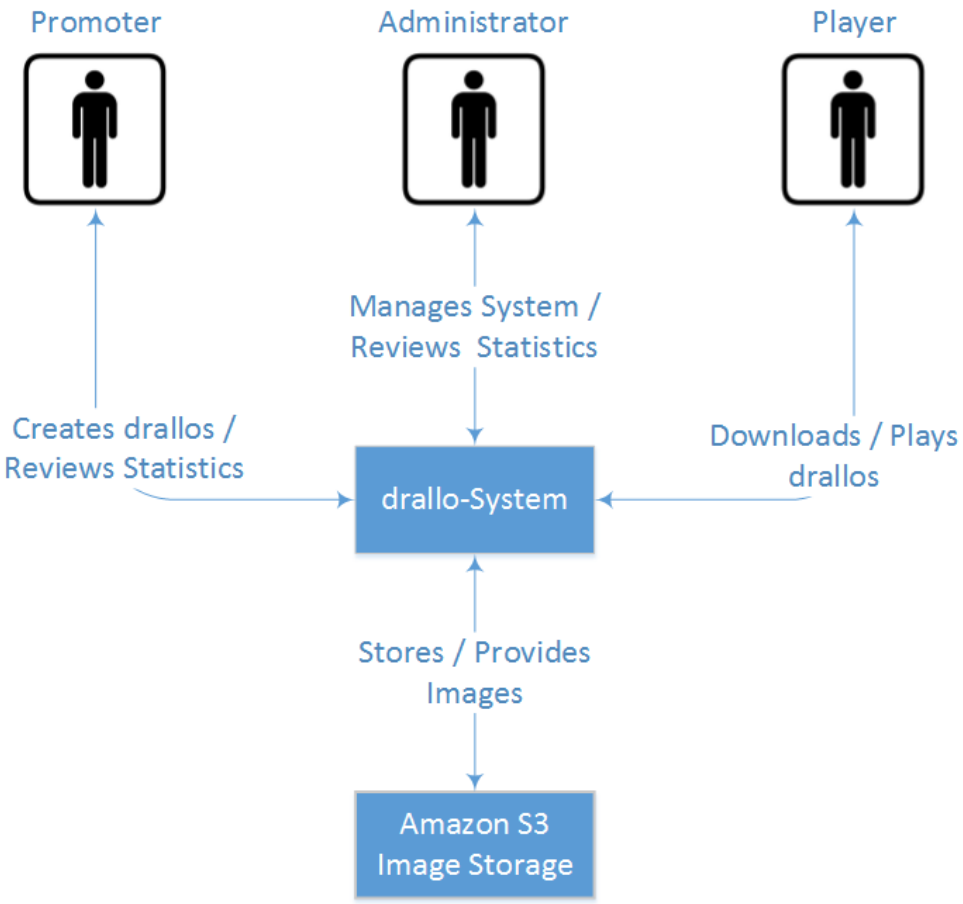

Abbildung 4.4.: System Kontext Diagramm der externen Schnittstellen

5. Anforderungen

5.1. Benutzer und Personas

Benutzer von drallo teilen sich in zwei Gruppen auf, welche für die Anforderungen in dieser Arbeit berücksichtigt werden.

5.1.1. Player

Players verwenden hauptsächlich die drallo-App für Smartphones. Sie spielen darauf meistens ein einzelnes drallo, welches an ihrem derzeitigen Standort verfügbar ist.

Bei den Player-Personas wurde bewusst nur eine ausgearbeitet, da diese bereits den anspruchsvollsten Benutzer verkörpert, den drallo ansprechen kann.

Persona Robert

Er ist **45 Jahre alt** und Wohnt mit seiner Familie in Uster. Hat einen 15-jährigen Sohn und eine 12-jährige Tochter.

Robert liest gerne und geht mindestens vier Mal pro Jahr in ein Museum. Er unternimmt am Wochenende gerne etwas in der Schweiz mit seiner Frau und/oder seinen Kindern.

Er arbeitet als Buchhalter in einem Treuhandbüro in Thalwil.

Technisches Verhalten Arbeitet täglich acht Stunden mit dem PC. Nutzt gerne neue Technologien. Besitzt ein iPad und iPhone. Probiert gerne neue Apps aus, wenn sie ihm einen Zusatznutzen bringen. Im Durchschnitt installiert er ein neues App im Monat. Er verwendet zum Beispiel regelmässig die Kindle App, WhatsApp und Runkeeper. Wenn er nicht weiterkommt, fragt er jeweils seinen 15-jährigen Sohn. Mittlerweile kommt er mit den Apple Produkten zurecht, allerdings verwirrt es ihn, wenn in einem App nicht die **Standard-Steuerelemente** von iOS verwendet werden, wie z.B. Spinner. Die **iOS Updates macht er immer sofort**, da er die Aufforderungen mittlerweile kennt.

Kommunikationsverhalten Nutzt hauptsächlich Email, WhatsApp und Telefon. WhatsApp hat SMS fast komplett ersetzt, da es sehr einfach zu bedienen ist und es die meisten Bekannten nutzen. Die Telefonfunktion in WhatsApp verwendet er immer wieder aus Versehen, da das Hörersymbol für ihn irreführend ist.

Ziele Möchte bei seinen Unternehmungen möglichst viel Neues erfahren und erleben. Möchte jederzeit eine Führung durch ein Museum oder eine Stadt machen können, welche zusätzliche Informationen vermittelt. Möchte sich messen mit Freunden. Will Fokus auf Content, nicht auf App Features.

5.1.2. Promoters

Promotoren sind die zahlende Kundschaft von drallo, wie zum Beispiel lokale Tourismusbüros oder Museumsbetreiber. Sie verwenden die drallo-Plattform hauptsächlich, um neue drallos zu erstellen und Teilnahmestatistiken anzusehen.

Persona Carmen

Carmen ist **35 Jahre alt** und lebt alleine in Zürich.

Sie unternimmt viel mit Freunden und reist gerne. Sie ist auf dem Land aufgewachsen und wohnt jetzt in Zürich.

Carmen arbeitet 80 Prozent bei Zürich-Tourismus und hat Tourismus an einer höheren Fachschule studiert.

Technisches Verhalten Sie nutzt den PC täglich bei der Arbeit, vor allem Excel für Auswertungen und ein Buchhaltungsprogramm. Ausserdem ist sie zu Recherchezwecken viel im Internet unterwegs. Sie hat keinen Lieblingsbrowser, nutzt aber immer Chrome, da dieser im Geschäft vom Informatiker empfohlen wurde. Carmen nutzt Technologie in der Freizeit nur wo sie nützlich ist, deshalb installiert sie nur selten neue Apps (ca. alle drei Monate eine neue App). Seit vier Jahren besitzt sie ein iPhone 4s. Ihr privates Notebook ist fünf Jahre alt. Sie überlegt sich nun ein neues Telefon zu kaufen und tendiert zu einem Android, da ihr dies von Freunden empfohlen wurde.

Kommunikationsverhalten Täglich nutzt sie mehrere Stunden WhatsApp und Facebook, um mit ihren Bekannten im In- und Ausland in Kontakt zu bleiben.

Ziele Sie möchte mit neuen Ideen ihre Destination für Touristen spannender machen, will den Menschen Zürich näher bringen und spannendes Wissen vermitteln. Möchte spezielle Anlässe mit neuen Medien unterstützen können. Möchte sich mit innovativen Projekten von der Konkurrenz abgrenzen. Möchte möglichst wenig Zeit im Editor verbringen, da es nur eine Aufgabe von Vielen ist.

5.2. User-Stories

Die funktionalen Anforderungen wurden in Absprache mit dem Industriepartner und einem Kunden von drallo im Laufe des Projekts erarbeitet. Es ergab sich eine Liste an User-Stories pro Persona. Es wurde absichtlich die Kurzform ohne Nutzen gewählt um die User-Stories zu beschreiben.

5.2.1. Promoter

- Ich als Promoter kann eine Multiplayer Challenge auf der drallo Plattform anlegen und mit einer Singleplayer Challenge verknüpfen.
- Ich als Promoter kann Titel, Mindest- und Maximalanzahl an Player für ein Multiplayer drallo definieren.
- Ich als Promoter kann Aufgaben für alle Teammitglieder definieren.
- Ich als Promoter kann Aufgaben für einzelne Spieler definieren.
- Ich als Promoter kann textuelles und multimediales Feedback für den ausführenden Player und den Rest definieren.
- Ich als Promoter kann Objekte auf der Karte erstellen und diese bei Conditions ein- oder ausblenden.
- Ich als Promoter kann ein bestehendes drallo verändern.
- Ich als Promoter kann meine erstellten drallos finden.

5.2.2. Player

- Ich als Player werde im Singleplayer-Modus zu einem Multiplayer Spiel eingeladen und kann entscheiden, ob ich einem Team beitreten möchte.
- Ich als Player erhalte nach dem Spielstart Aufgaben im Multiplayer-Modus.
- Ich als Player habe die Möglichkeit Aufgaben im gleichen Stil wie im Singleplayer-Modus zu erfüllen.
- Ich als Player sehe Objekte auf der Karte, wenn diese für die Erfüllung meiner Aufgabe notwendig sind.
- Ich als Player erhalte eine Nachricht, wenn Teammitglieder eine Aufgabe erledigt haben.
- Ich als Player werde benachrichtigt, wenn ein Teammitglied das Spiel verlässt.
- Ich als Player erhalte am Ende des Spiels eine Punkteauswertung.
- Ich als Player kann während eines Multiplayer Spiels meine Mitspieler auf der Karte finden.
- Ich als Player kann ein Multiplayer Spiel verlassen.

5.3. Nichtfunktionale Anforderungen

5.3.1. iPhone Kompatibilität

Synopsis	Die iOS App muss iOS ab Version 8.0 und ab iPhone 5 ohne Einschränkungen lauffähig sein. Die App muss mit 64-Bit lauffähig sein.
Relevante QA	Compatibility
Messbarkeit	Die Software lässt sich auf der angegebenen Plattform (iPhone 5, iOS 8.0) installieren, starten und spielen.
offene Probleme	Kompatibilität aller neu verwendeten Komponenten muss gewährleistet sein.

5.3.2. Android Kompatibilität

Synopsis	Android App muss ab Android 4.4 ohne Einschränkungen lauffähig sein.
Relevante QA	Compatibility
Messbarkeit	Die Software lässt sich auf der angegebenen Plattform (Android 4.4) installieren, starten und spielen.
offene Probleme	Kompatibilität aller neu verwendeten Komponenten muss gewährleistet sein.

5.3.3. Fehlertoleranz

Synopsis	Der Singleplayer Teil eines drallos muss weiterhin funktionieren, auch wenn ein oder mehrere Spieler die Verbindung verlieren. Bei zwei oder mehr verbleibenden Spielern muss auch das Multiplayer Spiel weiterlaufen.
Relevante QA	Robustness
Messbarkeit	In der Messung wird ein drallo mit einem vierer Team gestartet. Zwei Spieler schalten das Gerät mitten im Spiel aus. Die Verbleibenden müssen das Spiel fertig spielen können.
offene Probleme	Was passiert beim Wiedereintritt eines ausgefallenen Spielers?

5.3.4. Erweiterbarkeit

Synopsis	Die Teamplay Komponente wird so gebaut, dass weitere Teamplay Spielarten mit vergleichbaren Anforderungen einfach auf der bestehenden Komponente aufgebaut werden können.
Relevante QA	Extensibility
Messbarkeit	Der Implementationsaufwand weiterer Spielarten soll erheblich kleiner sein als der, der ersten Teamplay-Variante.
offene Probleme	Kann im Rahmen der Studienarbeit nicht gemessen werden, da die zu Verfügung stehende Zeit nicht ausreicht.

5.3.5. Performance

Synopsis	Die Multiplayer-Komponente funktioniert mit vier Spielern in einem drallo. Wir testen mit vier Playern, weil diese Anzahl, auch durch Absprache mit dem Industriepartner, die am ehesten erwartete Teamgrösse sein wird.
Relevante QA	Performance, Capacity
Messbarkeit	Ein Teamplay-drallo mit vier Spielern wird vollständig durchgespielt. Es treten keine unerwarteten Wartezeiten (grösser als drei Sekunden) auf.
Offene Probleme	Die Wartezeiten können auch durch den Netzwerkanbieter auftreten. Diese können wir weder messen noch verhindern.

5.3.6. Mobile Datennutzung

Synopsis	Das Teamplay-drallo verursacht maximal 100 Kilo Byte Datenverkehr in der Minute. Ausgenommen sind Spielsituationen, welche die Übertragung von Medien (Audio, Bilder, Videos oder Kartenmaterial) erfordern.
Relevante QA	User Costs, Data Usage
Messbarkeit	Die Datennutzung kann auf dem Mobilgerät gemessen werden. Es soll ein Szenario mit zwei Spielern und mit vier Spielern getestet werden.
Offene Probleme	Der Overhead des zugrunde liegenden Kommunikationsprotokolls ist nicht bekannt

5.3.7. Team Erstellung

Synopsis	Die Erstellung eines Teams um ein drallo zu spielen soll einfach sein und wenig Zeit in Anspruch nehmen. Die Entscheidung für die Verfügbarkeit eines Multiplayer drallos und die Teamerstellung soll deshalb automatisch geschehen.
Relevante QA	Performance, Simplicity
Messbarkeit	Wenn ein Multiplayer drallo verfügbar ist, soll der Player von der Applikation benachrichtigt werden. Er soll so einem automatisch gebildeten Team beitreten können oder die Einladung ablehnen können und im Singleplayer-Modus weiter spielen können. Die Applikation soll mit Popups arbeiten, um zusätzliche Navigation über Activities zu verhindern, wenn kein Multiplayer drallo gespielt werden will.

5.3.8. Kommunikation

Synopsis	Um einen Spielablauf zu gewährleisten, muss die Kommunikationsmethode folgende Anforderungen erfüllen: • Übertragen von 1000 Echo Nachrichten • Übertragen einer Nachricht mit einer Grösse von 10kB • Erkennen eines Connection Timeouts • Erkennen eines Transport Timeouts • Erkennen von inaktiven Clients • Aufrechterhalten der Verbindung für eine Stunde ohne Aktivität
Relevante QA	Performance, Reliability
Messbarkeit	Die Messbarkeit wird durch einen Prototyp gewährleistet, der die oben genannten Funktionen direkt anbietet.

6. Architektur

6.1. Ziele

Um Multiplayer Spiele zu ermöglichen, müssen die Clients miteinander kommunizieren können. Es muss ausserdem möglich sein, Teams zu bilden. Eine Multiplayer-Komponente muss aber auch die Möglichkeit bieten, an ein Team Aufgaben zu stellen und deren Erfüllung zu validieren.

Die Multiplayer-Komponente wird fortan mit **drallo Multiplayer Manager** (dMM) bezeichnet.

6.2. drallo Multiplayer Manager (dMM)

6.2.1. Anforderungen

Aus den funktionalen und nicht funktionalen Anforderungen für das Gesamtsystem ergeben sich für den dMM folgende Anforderungen:

- Empfängt Nachrichten von Clients
- Kann Nachrichten von Clients auswerten
- Kann Nachrichten basierend auf Antworten an einen oder mehrere Clients liefern
- Leitet Nachrichten an Teammitglieder weiter
- Steuert den Spielablauf während eines Multiplayer Spiels
- Stellt eine Auswertung der erreichten Punkte pro Spieler zur Verfügung
- Verwaltet Teams
- Verwaltet Erstellung und Auflösung eines Teams
- Verwaltet Beitritt der Player zu Teams
- Verwaltet alle Spiele

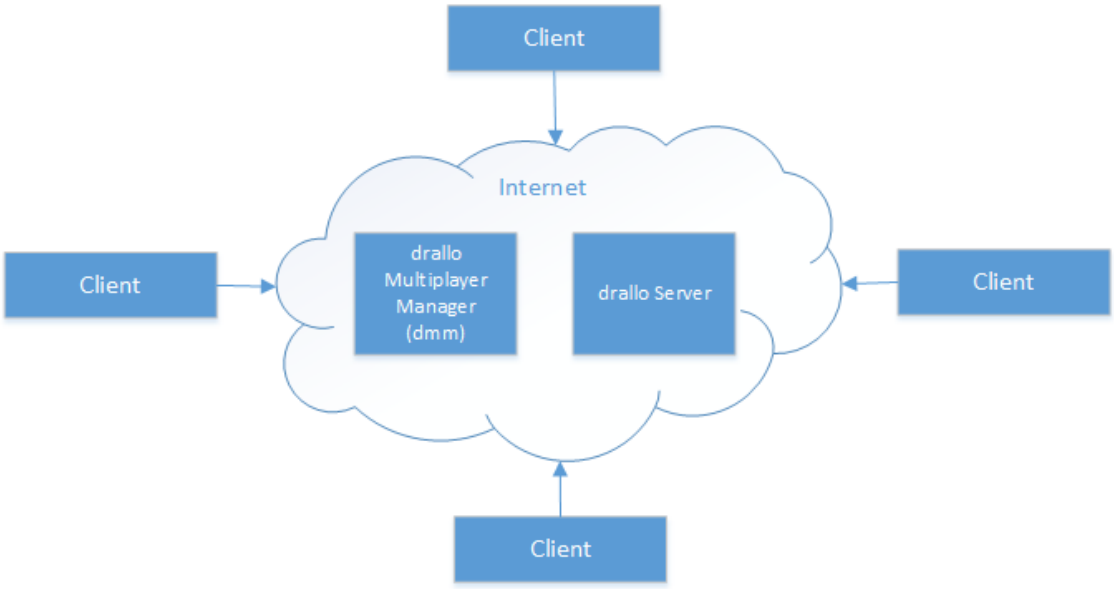

Abbildung 6.1.: drallo Multiplayer Manager und drallo Server im Internet als einzelne Services verfügbar

Um möglichst bestehende Technologien nutzen zu können, soll der **dMM mit .NET-Technologien** umgesetzt werden, da dies die Möglichkeit bietet, bestehende Spiellogik-Komponenten (Challenge Engine) einzubinden.

6.2.2. Kommunikationsarchitektur

Peer-to-Peer

Die erste Möglichkeit die oben genannten Ziele zu erreichen, wäre ein Peer-to-Peer Verbindung zwischen allen Clients (Abb. 6.2).

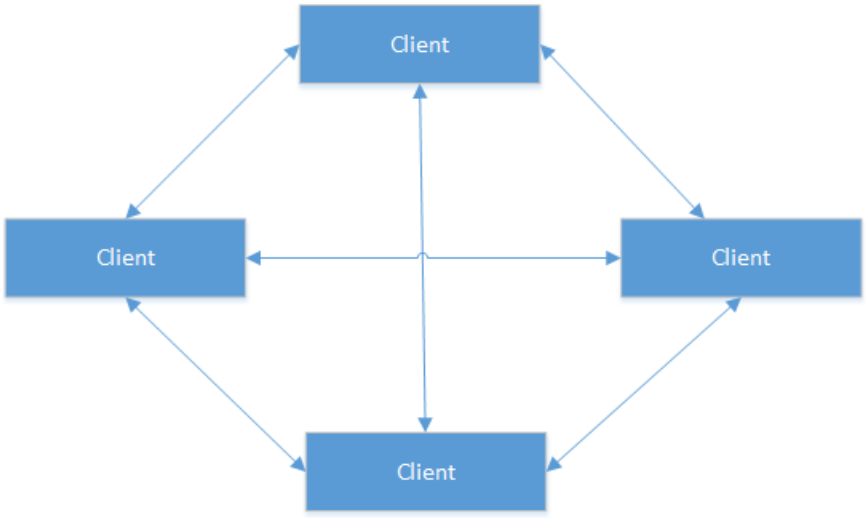

Abbildung 6.2.: Verbindungen zwischen allen Clients, die sich für ein Spiel verbinden möchten

Dabei gibt es aber einige **Nachteile**:

- Viele Verbindungen müssen aufgebaut werden

- Nachrichten müssen immer direkt an alle geschickt werden, was zu höherem Datenverbrauch führt (Entscheidend bei mobilem Internet)
- Dezentrale Verwaltung kann schwierig werden, vor allem wenn Disconnects oder Reconnects gehandelt werden müssen
- Logik zur Spielsteuerung müsste auf jedem Client laufen

Zentralisierte Kommunikation

Aus den oben erwähnten Nachteilen ergibt sich schnell eine bessere Lösung: **Eine zentrale Komponente**, die die Kommunikation zwischen den Clients steuert und über das Internet erreichbar ist (Abb. 6.3).

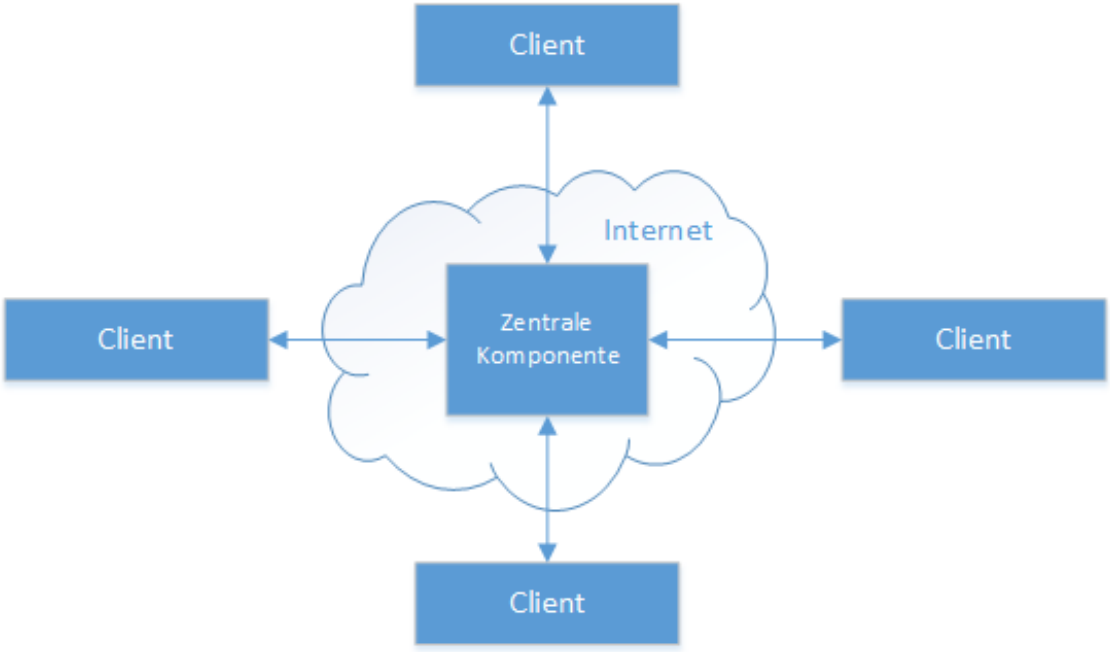

Abbildung 6.3.: Eine über das Internet erreichbare, zentrale Komponente

6.2.3. Kommunikations-Technologie

Für die Verbindung zwischen den Clients und dem dMM muss eine Technologie gefunden werden, die den Anforderungen aus den funktionalen und nicht-funktionalen Anforderungen entspricht. Es ist ausserdem zentral, dass sie sich gut in die Xamarin-Apps integrieren lässt und auch keine hohen oder unvorhersehbaren Kosten verursacht.

Um die Anforderungen an die Kommunikationstechnologie zu evaluieren und herauszufinden, welche MEP (Message Exchange Pattern) oder bestehenden Angebote genutzt werden können, müssen zuerst die verschiedenen Szenarien analysiert werden.

Szenario 1

Ein Client möchte sich für ein Multiplayerspiel anmelden und registriert sich dafür beim Server. Der Server sendet ihm, falls ein Multiplayer drallo möglich ist, eine Einladung für den Beitritt zu einem Team zurück.

Dies könnte mit HTTP realisiert werden.

Szenario 2

Ein Client erhielt gerade die Einladung zum Team, währenddessen bricht die Verbindung zu einem anderen Client ab. Der nicht mehr verfügbare Client soll nun nicht mehr im Teamplay integriert oder aufgelistet sein. Zusätzlich muss der Server den **Verbindungsabbruch des Players feststellen** können.

HTTP kann diesen Fall in der Standardausführung nicht erfüllen, da in HTTP die Kommunikation per Definition vom Client initiiert wird und sofort geantwortet wird. Der Server benötigt also eine Möglichkeit, jederzeit Nachrichten an den Player zu senden. Das wäre auch mit HTTP Long-Polling [2] zu erreichen.

Szenario 3

Ein Player tritt einem Team mit drei anderen Player bei. Das Spiel wird darauf hin gestartet. Alle Teammitglieder erhalten eine Nachricht, um sie über den Start des Spiels zu informieren.

Dafür muss ein Publish / Subscribe Mechanismus vorhanden sein, auf welchem alle Teammitglieder eingeschrieben und erreichbar sind.

Szenario 4

Während eines Spiels vollendet ein Player eine Aufgabe. Der Server sendet nun an alle **anderen Clients** im Team eine Nachricht.

Mit einem Publish / Subscribe Konzept funktioniert dies bereits nicht mehr, da hier nicht alle Clients in einem Team die Nachricht erhalten sollen. Es wird also **eine separate Verbindung zu jedem Client** benötigt. Der Server selbst entscheidet, an welche Clients die Nachrichten geschickt werden müssen.

Zusammenfassung

Diese Szenarien zeigen auf, dass für jeden Client eine einzelne Verbindung mit dem Server benötigt wird und es deshalb weniger sinnvoll ist ein System mit einem Publish / Subscribe - Mechanismus zu nutzen. Wichtig ist ebenfalls, dass Verbindungsabbrüche vom Server sofort bemerkt werden und je nach Status dieses Clients andere Aktionen ausgelöst werden können (siehe NFRs).

Aus den Anforderungen ergab sich die Entscheidung für Websockets als Kommunikations-Technologie.

Da im Team bereits Erfahrung mit Websockets vorhanden war und wir uns sowieso selbst um das Connection Management und das Verwalten von Clients und Zielgruppen kümmern müssen, fiel die Entscheidung gegen Messaging-Queues und Cloud Angebote wie Microsoft Azure Service Bus und Google Cloud Messaging aus der fehlenden Nützlichkeit der zusätzlichen Features, die diese Dienste bieten.

Websockets können grundsätzlich alle Anforderungen erfüllen, ohne dass sie zusätzliche Kosten verursachen oder die Implementation verkomplizieren. Deshalb wurde entschieden einen Prototyp mit Websockets zu bauen.

6.2.4. Kommunikations-Prototyp

Der Websocket-Prototyp sollte folgende Eigenschaften prüfen:

- Server Stellt Verbindungsabbruch fest
- Client kann nach Verbindungsabbruch die Verbindung wiederherstellen
- Server lehnt keine Nachrichten ab, auch wenn er überlastet ist. Es muss garantiert sein, dass alle Nachrichten ankommen, solange die Verbindung besteht.

Nach intensivem Testen und Erproben der Szenarien wurde festgestellt, dass die momentan verfügbaren Libraries für Websocket-Kommunikation in Xamarin einige Schwierigkeiten mit sich bringen.

- Abbruch der Verbindung bei gleichzeitigem Senden und Empfangen unabhängig von der Menge an Nachrichten
- Abbruch der Verbindung beim Empfangen von vielen Nachrichten (zwischen 50 und 1000)
- Timeout Handling aufwändig, da das Vorgesehene nicht zuverlässig ist

Der Abbruch der Verbindung konnte nicht nachvollzogen werden, da keine Exceptions geworfen wurden und somit auch keine Stacktraces verfügbar waren. Es war zudem kein Muster zu erkennen, nach wie vielen Nachrichten die Verbindung abbricht. Normalerweise erst nach ein paar Hundert, jedoch wurden auch Fälle getestet, wo sie nach nur ein paar Dutzend abbrach. Die wenigen verfügbaren Informationen in den Error Nachrichten, wiesen auf Fehler im zugrunde liegenden Frameworks (Xamarin) hin (SIGABRT).

Aufgrund dieser Erfahrungen entschieden wir uns für SignalR [3], eine Kommunikations-Library von Microsoft, welche mit Long-Polling statt Websockets arbeitet, jedoch zuverlässiger funktioniert als die getesteten Websocket-Libraries. Microsoft hat hier bereits angekündigt, dass es in naher Zukunft eine Implementierung für Websockets mit SignalR geben soll.

6.2.5. Bezug: Message Exchange Patterns (MEP)

Um die Kommunikation nach bekannten, erfolgreichen Mustern zu gestalten und damit zu vergleichen, wurden folgende Message Exchange Patterns untersucht.

Publish / Subscribe

Definition: Send the event on a Publish-Subscribe Channel, which delivers a copy of a particular event to each receiver. [5]
Um den Ansprüchen an die Kommunikation der Applikation gerecht zu werden, reicht ein Publish/-Subscribe Mechanismus alleine nicht aus. Nachrichten sollen von Server, sowie Client initiiert werden können. Man müsste zudem mehrere Topics führen, da ja die Nachrichten innerhalb des Teams unterschiedlich sind. Eine weitere Schwierigkeit besteht darin, dass der dGM (drallo Game Manager) die Identitäten der Subscriber kennen soll, was gemäss Definition des Patterns nicht der Fall ist.
Microsoft: "*How can an application in an integration architecture only send messages to the applications that are interested in receiving the messages* **without knowing the identities of the receivers?**" [4]

Request / Reply (Half-Duplex)

Definition: Send a pair of Request-Reply messages, each on its own channel. [5]
Die Applikation sendet häufig ohne eine direkte Antwort zu erwarten. Zum Beispiel wird eine Register-

Message zum Server gesendet, ohne dass dieser diese Nachricht bestätigt, oder direkt darauf reagiert. Ein Beispiel, wo dieses Pattern in der drallo Multiplayer Kommunikation eingesetzt wird ist die Join-Message, welche eine JoinAcceptedMessage oder JoinRejectedMessage als Antwort erwartet.

Message Selector

Definition: Make the consumer a Selective Consumer, one that filters the messages delivered by its channel so that it only receives the ones that match its criteria. [5]
Gegen diesen Ansatz spricht hauptsächlich der zusätzlich generierte Datenverkehr, die Logik auf dem Client, sowie, dass der Server die Kontrolle über die Consumer dem Client abgeben müsste. Man möchte dem Client nur die Nachrichten senden, die er tatsächlich anzeigen wird.

Duplex

Definition: The duplex MEP allows an arbitrary number of messages to be sent by a client and received in any order. The duplex MEP is like a phone conversation, where each word being spoken is a message.[4]
Die in diesem Projekt implementierte Lösung erinnert stark an das Duplex Message Exchange Pattern. Jeder Teilnehmer kann jederzeit Nachrichten unabhängig voneinander initiieren und entgegennehmen. Nachrichten können synchron, als auch asynchron versendet werden.

Zusammenfassung

Da die eingesetzte Library SignalR mit HTTP Long-Polling arbeitet, werden dem Entwickler keine Channels, wie bei einem Messaging System, zur Verfügung gestellt. Das zugrundeliegende Protokoll HTTP baut auf dem Prinzip des “Request / Reply”-Patterns auf. Durch das Verwenden von zwei HTTP-Verbindungen, wird ein “Duplex”-Verhalten erreicht. Das Verfahren könnte deshalb auch mit “Duplex over Half-Duplex” beschrieben werden.

6.3. Erfahrungen mit Cloud Anbietern

6.3.1. Microsoft Azure

Die zentrale Komponente für die Multiplayer-Spiele, der drallo Multiplayer Manager ist ein ASP.NET Projekt. Microsoft bietet mit Webapps eine sehr einfache Möglichkeit um diese zu veröffentlichen.

Deployment

In der Entwicklungsumgebung (Visual Studio Enterprise 2015) kann sich der Benutzer gleich zu Beginn mit dem persönlichen Microsoft-Konto anmelden. Dies ermöglicht es ein Projekt zu erzeugen, welches direkt auf die Microsoft Azure Cloud deployed werden kann. Beim initialen Deployment sind folgende Parameter zu konfigurieren:

- **Name** z.B. drallomultiplayermanager.azurewebsites.net
- **Service Plan** Vor allem Kosten / Leistungen
- **Ressource Group** Gruppieren von Instanzen
- **Region** Ort des Data Centers z.B. West Europe

Das Deployment erforderte von unserer Seite keinerlei Aufwand und funktionierte zuverlässig. Der Ablauf erfordert keinerlei Einarbeitungszeit. In der Mitte des Projekts haben wir die Applikation zum Konto des Industriepartners transferiert. Dazu gibt es leider keine Funktionalität. Im ersten Schritt muss die bestehende App gelöscht werden, damit der Name nicht mehr besetzt ist. Danach gestaltet sich das Deployment auf das neue Konto wieder sehr einfach.

Azure Portal

Das Azure Portal bietet benutzerfreundlichen Einblick in das Geschehen auf dem Server.

Überraschenderweise hat sich das komplette User Interface dieses Portals drei Wochen vor Projektende geändert. Der Zugriff auf die wichtigsten Funktionen waren auch nachher gut auffindbar.

Logdateien

Etwas umständlich ist die Handhabung der Logdateien. Die Logs des Webservers können auf einer speziellen Seite im Portal eingesehen werden. Neue Logeinträge werden nicht automatisch auf der Seite eingeblendet. Die Logdateien der eigenen Applikation können auf dem Dateisystem des Webservers gespeichert werden und von dort via FTP heruntergeladen werden. Wenn man sich an die einfache und praktische Handhabung der Logdateien auf Unix Webservern gewohnt ist, ist man hier schnell frustriert.

6.3.2. Heroku

Heroku war bereits als PaaS (Platform as a Service) -Anbieter für den drallo Server (Ruby on Rails) im Einsatz. Für die Dauer dieser Arbeit konnte eine zusätzliche Instanz (Dyno) verwendet werden, um Multiplayer drallos zu erstellen und diese zu ändern.

Das Deployment ist sehr simpel. Der veränderte Code kann per GIT [1] auf das Remote Repository bei Heroku geladen werden und wird dann innert Minuten installiert und zur Verfügung gestellt.

Dadurch, dass die Installation bereits vollständig vorhanden war, gab es keinerlei Aufwand oder Probleme im Zusammenhang mit Heroku.

7. Beispielanwendung

Dieses Kapitel ist eine User Guide für Promoters, wir zeigen die Erstellung und das Spiel anhand eines einfachen Multiplayer drallos. Es hilft auch zur Veranschaulichung der Ergebnisse dieser Arbeit.

7.1. Erstellung eines drallos im Editor

Für die Erstellung eines Multiplayer drallos wird ein Promotoren-Account benötigt, welcher die Möglichkeit bietet, alle eigenen drallos zu verwalten.

Dort kann ein neues drallo erstellt werden, was den in der Arbeit erstellten Editor (Abb. 7.1) öffnet.

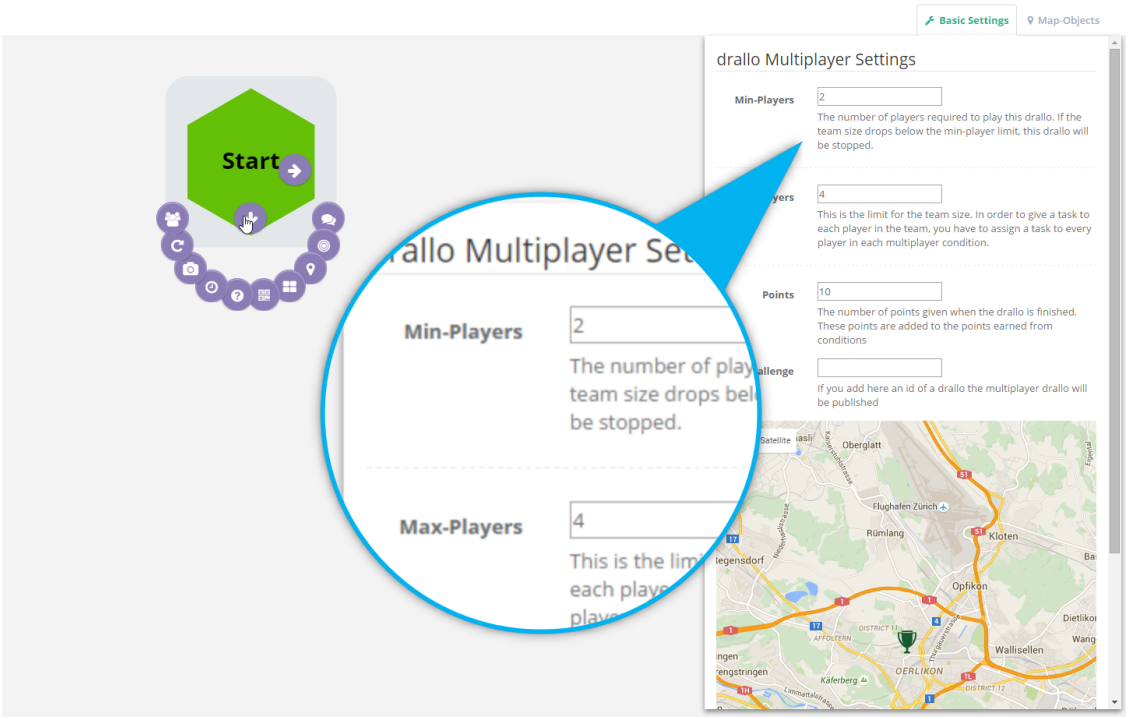

Abbildung 7.1.: Multiplayer Editor im Startzustand

Um nun Conditions zum Multiplayer-drallo hinzuzufügen, kann wie gehabt auf der Startfläche der Pfeil nach unten betätigt um die verfügbaren Conditions anzuzeigen. Wie in Abbildung 7.2 gezeigt wird, wird hier neben den Conditions neu zusätzlich ein Team-Container angeboten.

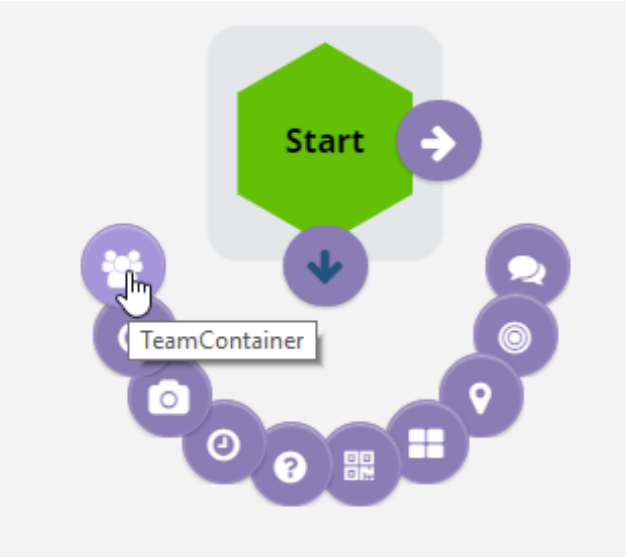

Abbildung 7.2.: Erstellung Teamcontainer im Multiplayer Editor

Es wird nun ein leerer Team-Container (Abb. 7.3) gezeigt. In diesem kann man für jeden Player unterschiedliche Conditions erstellen.

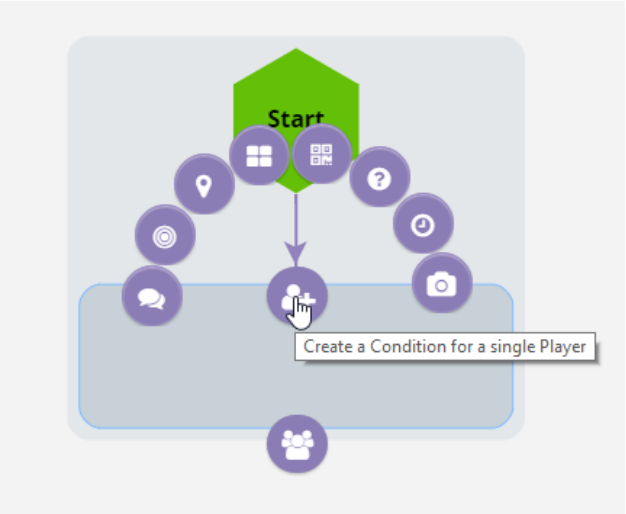

Abbildung 7.3.: Erstellung Condition für einzelnen Player im Multiplayer Editor

In Abbildung 7.4 wird gezeigt, wie innerhalb einem Team-Container den drei Players, hier als P1 bis P2 gekennzeichnet, unterschiedliche Conditions zugeteilt werden können.

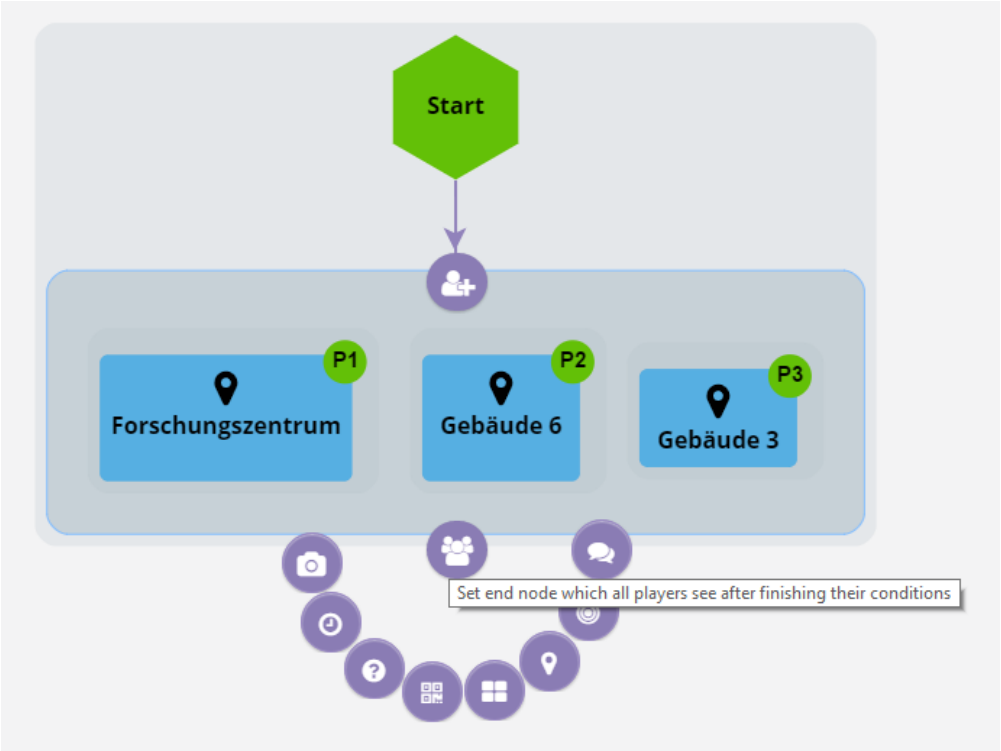

Abbildung 7.4.: Erstellung verschiedener Conditions für drei Players (P1, P2, P3) im Multiplayer Editor

In folgender Abbildung 7.5 wird am Beispiel eines kurzen, aber dennoch kompletten Multiplayer-drallos aufgezeigt wie die Struktur aussehen könnte. Dieser Spielablauf wird im nächsten Kapitel als Grundlage verwendet.

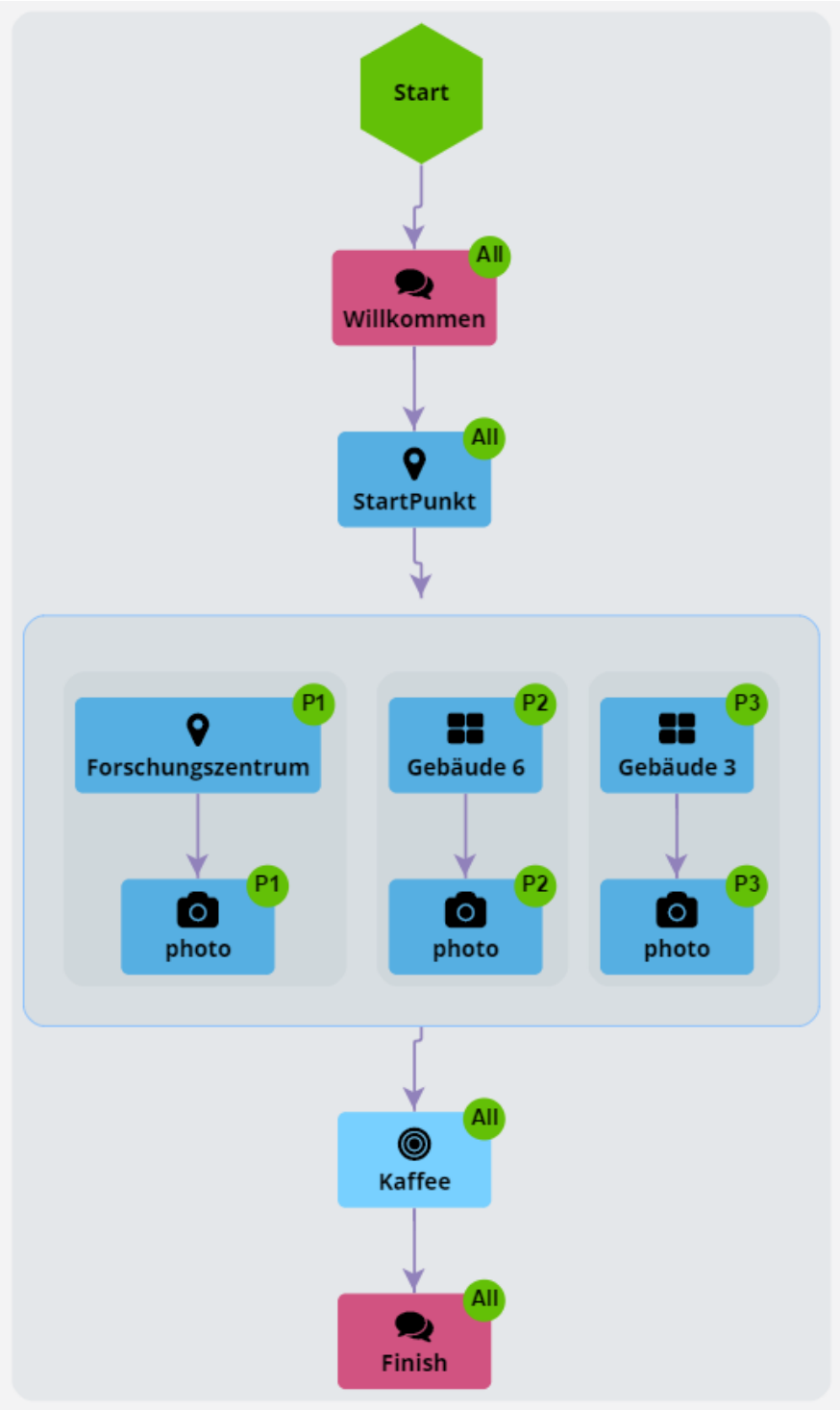

Abbildung 7.5.: Übersicht komplettes Multiplayer-drallo im Multiplayer Editor

7.2. Spielen eines drallos auf dem Smartphone

Um ein Multiplayer-drallo spielen zu können, muss ein Singleplayer-drallo existieren, in welches dann, wie im Editor definiert, ein oder mehrere Team-Container integriert wurden. Spielt der Player also gerade das Singleplayer-drallo wird er, sobald das Spiel Aufgaben für mehrere Player vorsieht und auch

ausreichend Player vorhanden sind, gefragt ob er bei einem Multiplayer-drallo mitspielen möchte (Abb. 7.6). Um seine Entscheidung zu fällen hat er eine gewisse Zeit, die konfiguriert werden kann.

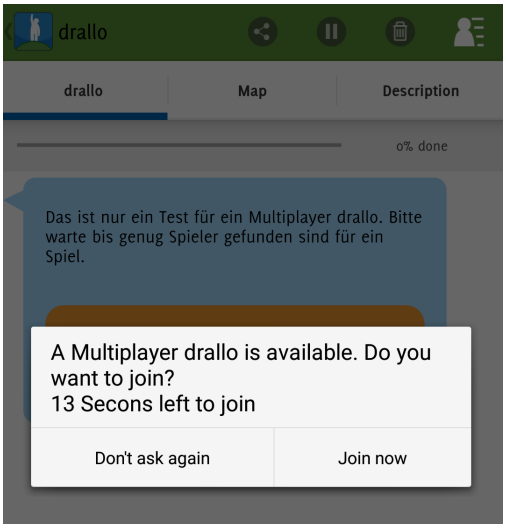

Abbildung 7.6.: Aufforderung zum Multiplayer Spiel auf dem Mobiltelefon

Entscheidet sich nun der Player für das Multiplayer-drallo wird er dem Team hinzugefügt. Es wird jetzt noch ein wenig gewartet, um anderen Playern auch noch die Zeit zu geben, sich zu entscheiden (Abb. 7.7).

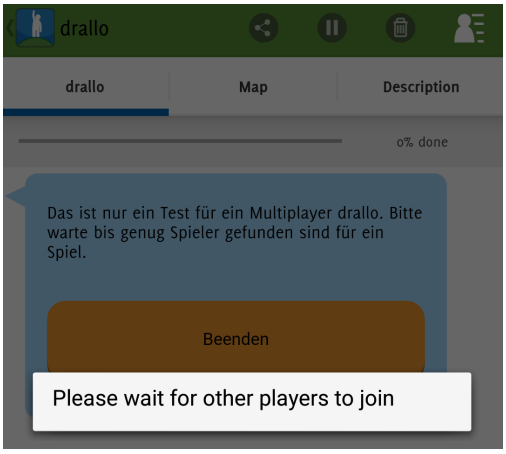

Abbildung 7.7.: Warten auf weitere Teammitglieder

Sollte nun der Fall auftreten, dass zu wenige Player die Einladung akzeptieren und somit kein ausreichend grosses Team gebildet werden kann, wird der Player informiert (Abb. 7.8). Ausserdem wird ihm mitgeteilt, dass er später nochmals gefragt werden wird.

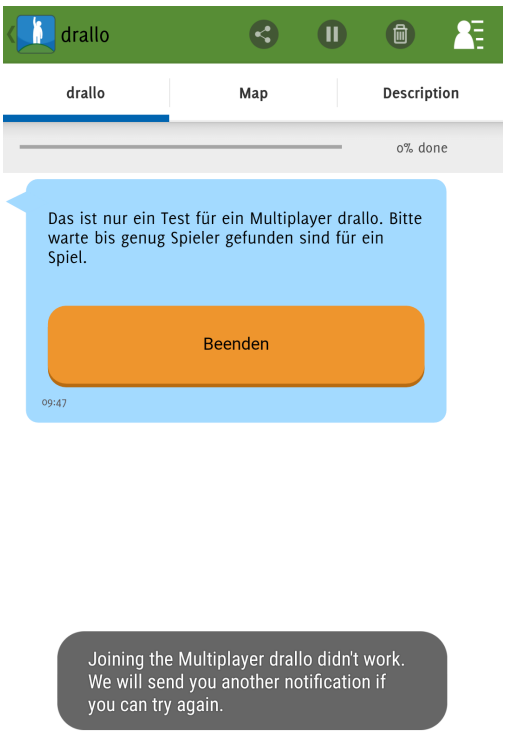

Abbildung 7.8.: Information für Player wenn Teambildung nicht möglich

Akzeptieren jedoch alle notwendigen Player die Einladung, startet das Multiplayer-drallo. Die folgenden Abbildungen zeigen den Spielablauf für Player zwei (P2), so wie das drallo im Editor erstellt wurde. Hier wird allen Playern dieselbe erste Aufgabe gestellt (Abb. 7.9, damit sie sich finden können. Um dem Player das Finden des Ortes zu erleichtern, wird ihm eine Karte (Abb. 7.10) angeboten. Dort sieht er neben seiner Position und dem Ziel auch die anderen Player.

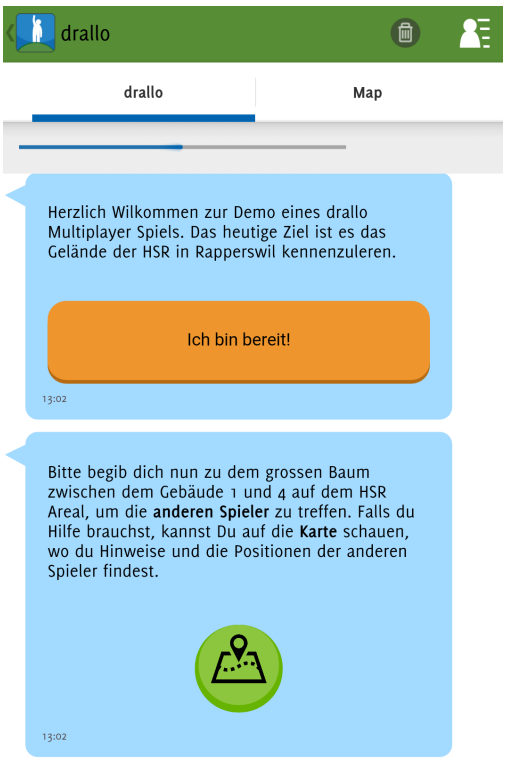

Abbildung 7.9.: Spielstart Multiplayer-drallo

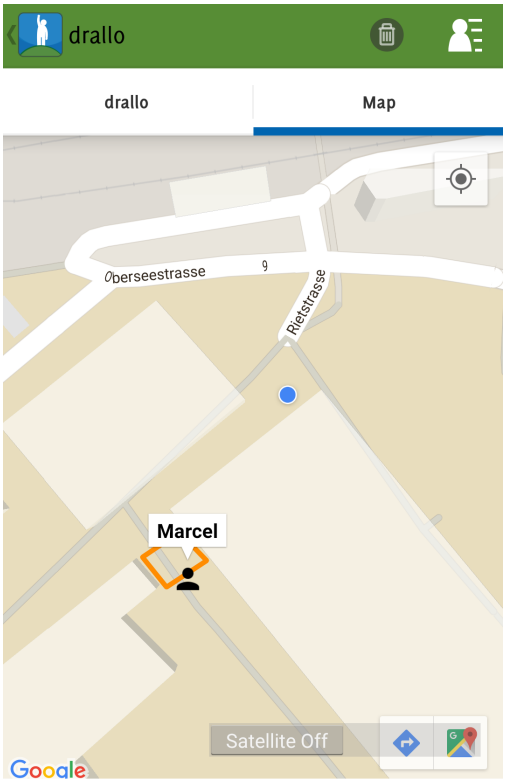

Abbildung 7.10.: Visualisierung auf der Karte von eigener Position, anderen Players und Ziel

Im gestarteten Testspiel können nun die Spieler gemeinsam sowie einzeln benachrichtigt werden und

individuell Aufgaben erhalten. In Abbildung 7.11 wird dem Player nach dem Finden der ersten Lokation mitgeteilt, dass er noch auf die anderen Player warten soll. Sobald diese angekommen sind, werden die individuellen Aufgaben verteilt. Der Player soll also das Gebäude 6 entdecken gehen.

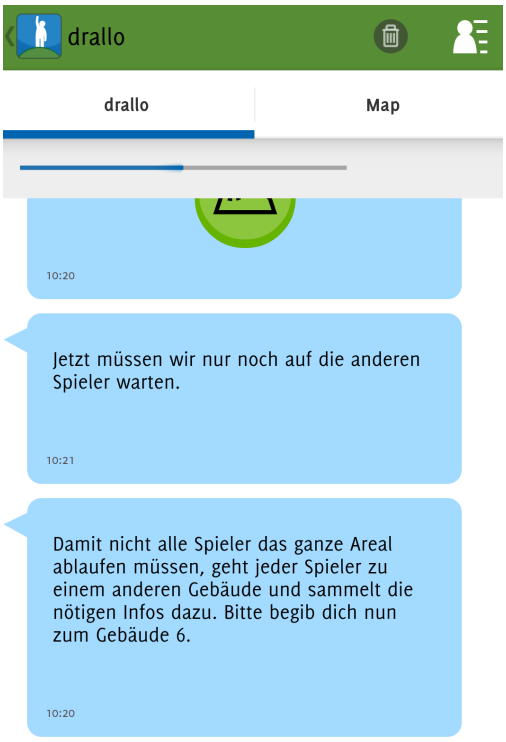

Abbildung 7.11.: Gemeinsame und einzelne Aufgabe

Abbildung 7.12 zeigt, wie dem Player eine individuelle Erfolgsmeldung gezeigt und gleich die nächste Aufgabe erteilt wird.

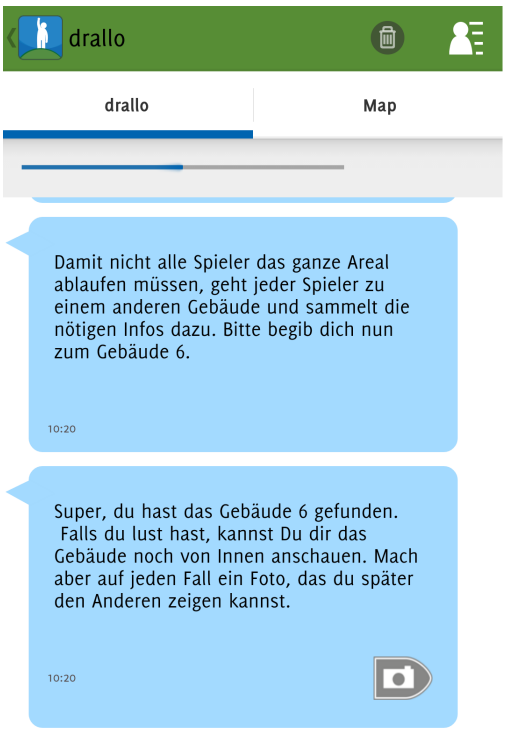

Abbildung 7.12.: Individuelle Erfolgsmeldung und neue Aufgabe für Player

Offenbar hat auch ein anderer Player seine Aufgabe erledigt. Der Player wird jetzt darüber informiert (Abb. 7.13).

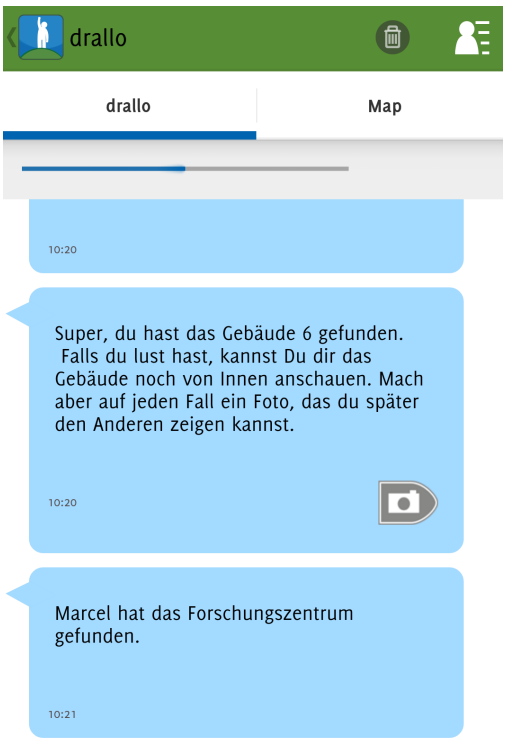

Abbildung 7.13.: Information über Spielstatus anderer Player

Der Player hat nun das Foto gemacht. Somit sind die individuellen Aufgaben für diesen Player erfüllt.

Die nächste Aufgabe (Abb. 7.14) ist wieder für alle Player dieselbe, damit sie sich auch am Ende des Spiels nochmals treffen.

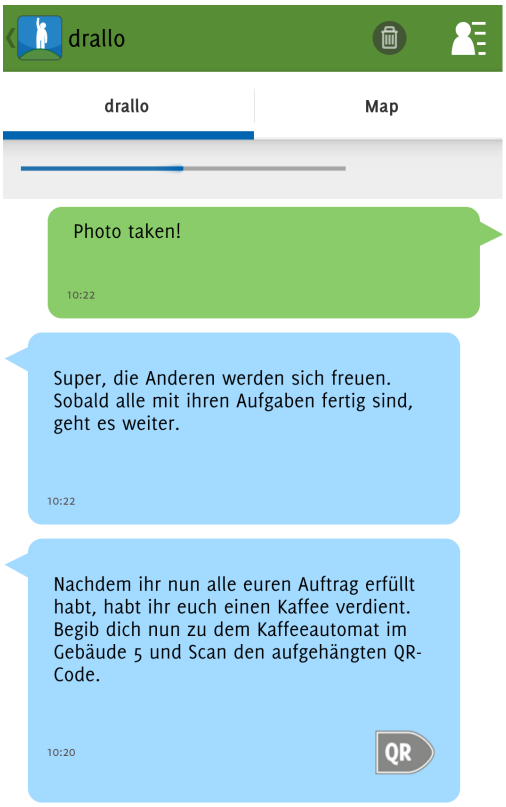

Abbildung 7.14.: Individuelle Status- und Erfolgsmeldung und neue gemeinsame Aufgabe

Am vorgegebenen Ort angekommen, scannen nun alle Player den dort platzierten QR-Code (Abb. 7.15). Diese eignen sich vor allem gut, um sehr präzise innerhalb eines Gebäudes festzustellen, wo sich die Player befinden.

Abbildung 7.15.: QR-Code scannen als gemeinsame Aufgabe um Players zusammenzuführen

Auch nach dieser Aufgabe wird eine Erfolgsmeldung ausgegeben. Da diese eine gemeinsame Aufgabe

ist, wird auf alle Teammitglieder gewartet und dann eine Auswertung angezeigt (Abb. 7.16). Das Multiplayer-drallo ist somit beendet.

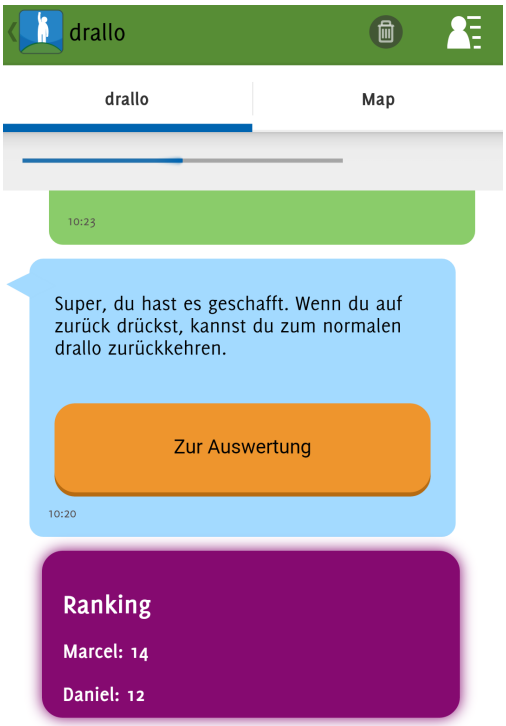

Abbildung 7.16.: Erfolgsmeldung und Auswertung nach Vollendung der gemeinsamen Aufgabe

8. Zusammenfassung und Ausblick

8.1. Zusammenfassung der Ergebnisse

Der entwickelte Multiplayer-Modus für drallo wurde nach dem Vorbild des Singleplayer-Modus umgesetzt und enthält auf Plattformebene eine sehr ähnliche Funktionalität. Alle nachfolgend erwähnten Conditions und Features können durch Promoters im Editor selbst zusammengestellt, bearbeitet und an einen oder mehrere Player verteilt und auf der App getestet werden.

8.1.1. Umgesetzte Features

Übernommene Features aus dem Singleplayermodus

Es können alle bestehenden Aufgabentypen verwendet werden. Namentlich sind das alle GeoConditions sowie EventConditions (Photo, QR Code, Question, iBeacon, usw.). Diese unterstützen auch die gewohnten Möglichkeiten, die aus dem Singleplayer drallos bekannt sind:

- Bilder, Videos, Audio-Dateien einbinden.
- Aufgabe Überspringen
- Objekte auf der Karte anzeigen/ausblenden
- Zeitlimit für eine Aufgabe setzen
- Texte definieren für erfolgreiches / fehlgeschlagenes Beenden einer Aufgabe

Ausserdem funktionieren die wichtigsten Conditions die andere Conditions beinhalten (Sequence, Parallel). Einzig die CircularSequence wurde nicht implementiert.

Neue Features für den Multiplayermodus

Um das Kooperationsspiel zu ermöglichen, wurden neue Features konzipiert und umgesetzt:

- Einfache Teambildung
- Synchronisationsbarrieren für ein Team
- Ranking am Ende eines Multiplayer Spiels
- Separate Teilaufgaben für Teammitglieder
- Teilaufgaben für alle Teammitglieder
- Finden von Mitspielern auf der Karte
- Nachrichten für Teammitglieder, über den Spielstatus der einzelnen Player

Die bestehende Infrastruktur stellt eine stabile Basis, welche weitere Erweiterungen im Multiplayermodus aller Art zulässt.

8.2. Schlussfolgerung

Während dieser Arbeit haben wir die Vision des Industriepartners in den Anforderungen formalisiert und das gestellte Problem in seinem Sinne adäquat gelöst. Der Prototyp, den wir konzipiert, entwickelt und integriert haben, erfüllt alle erarbeiteten funktionalen und nicht-funktionalen Anforderungen.

Obwohl während der Entwicklung Schwierigkeiten mit den Kommunikationstechnologien aufgetreten sind, konnten wir das Projekt pünktlich und vollständig fertigstellen. Dabei wurde die Erweiterbarkeit erzielt, indem keine bestehenden Komponenten geändert wurden. Diese konnten entweder wiederverwendet oder über neue Bibliotheken zwischen den Tiers geteilt werden.

Das erste Feedback des Industriepartners war durchwegs positiv und er plant bereits einen Test mit einem bestehenden Kunden.

Teil II.

Anhänge

A. Glossar

Administrator Mitarbeiter von Challenge Earth AG

Aufgabe Teilaufgabe eines drallos, drallo condition (z.B Location, Ibeacon, Frage, Photo)

Challenge Editor Werkzeug um drallos zu erstellen, verändern

Condition Teilaufgabe einer Challenge

dMM drallo Multiplayer Manager

drallo Name der App, auch eine Challenge innerhalb der Applikation

PCL Portable Class Library; plattformunabhängige Library

Player Spieler eines drallos mit dem Smartphone

Promoter Ersteller von drallos

Team Temporäre Gruppe von drallo Playern

Timeline Spielansicht welche die Nachrichten chronologisch zeigt.

Timeline Entry Eintrag in der Spiel-History der App

UI User Interface; die grafische Benutzeroberfläche auf dem Mobile Phone / im Simulator

UUID Universally Unique Identifier; globale, eindeutige Identifikationsnummer

B. Literatur

[1] Scott Chacon und Ben Straub. *Pro Git*. https://github.com/progit/progit2/tree/master/book. Letzter Zugriff am 15. Dezember 2015.

[2] Atlassian (basierend auf IETF RFC 6455). *HTTP Long-Polling*. https://eucalyptus.atlassian.net/wiki/display/ARCH/HTTP+Long-Polling, RFC 6455: https://tools.ietf.org/html/rfc6455. Letzter Zugriff am 15. Dezember 2015.

[3] Xamarin Inc. *SignalR - Incredibly simple real-time web for .NET*. https://components.xamarin.com/view/signalr. Letzter Zugriff am 15. Dezember 2015.

[4] Martin Fowler Rachel Reinitz und Mark Weitzel Kyle Brown John Crupi. *Patterns and Best Practices for Enterprise Integration*. http://www.enterpriseintegrationpatterns.com/. Letzter Zugriff am 13. Dezember 2015.

[5] Microsoft. *Definitionen der Message Exchange Patterns*. https://msdn.microsoft.com/. Letzter Zugriff am 13. Dezember 2015.

Abbildungsverzeichnis