Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Tim Sielemann

Doodlings: Entwicklung eines location-based Games
mit spielergenerierten Inhalten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Tim Sielemann

Doodlings: Entwicklung eines location-based Games mit spielergenerierten Inhalten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Olaf Zukunft
Zweitgutachter : Prof. Dr. Stefan Sarstedt

Abgegeben am 19.04.2016

Tim Sielemann

Thema der Arbeit
Doodlings: Entwicklung eines location-based Games mit spielergenerierten Inhalten

Stichworte
Location-based Game, Spielentwicklung, User Created Content, Unity, Aktoren, Pokémon

Kurzzusammenfassung
Die Entwicklung des Spielemarktes verlagert sich immer weiter auf Spiele für mobile Endgeräte. Dabei werden häufig die verfügbaren Ressourcen der heutigen Smartphones für die Erzeugung des Spielerlebnisses nicht einbezogen. Prominente Beispiele wie Ingress oder Geocaching zeigen, dass das Spielkonzept der location-based Games durchaus Zuspruch bei Spielern findet.
In dieser Arbeit wird die Konzeption und Entwicklung eines mobilen positionsbasierten Spiels mit der Bezeichnung „Doodlings“ beschrieben, das durch die Spieler erweitert werden kann. Dabei liegt das Hauptaugenmerk auf der evolutionären Entwicklung des Kernsystems und der Spielidee. Für die Umsetzung werden die Game-Engine „Unity“ und das Aktorenframework „Akka“ verwendet. Die Spielidee und Umsetzung werden anschließend durch eine systematische Nutzerbefragung bewertet.

Tim Sielemann

Title of the paper
Doodlings: A location-based Game with player generated Content

Keywords
Location-based Game, Game Development, User Created Content, Unity, Actors, Pokémon

Abstract
The trend of games for mobile devices can be recognized within the games market. Yet obtainable resources of current smartphones are not being included in the gaming experience. Prominent examples like Ingress or Geocaching show that the concept of location-based games are indeed obtaining popularity within the gaming community.
In this thesis, the game “Doodlings” is introduced. It is a mobile location-based game where players can create game extensions. This thesis focusses on the development of the core system and the game concept. The implementation is built upon the Game-Engine “Unity” and the Actor Framework “Akka”. The game concept and the core system is evaluated by a systematic user survey.

Inhaltsverzeichnis

1 Einleitung

Computer- und Mobilegames sind in den letzten Jahren zu einer der häufigsten Freizeitbeschäftigungen von Kindern, Jugendlichen und jungen Erwachsenen geworden [9]. Obwohl es sehr viele Spiele für mobile Endgeräte gibt, werden die zusätzlichen Möglichkeiten, die durch die Hardware von Smartphones zugänglich werden, selten genutzt. Dabei wächst auch das Interesse an Spielen, welche den Spieler in eine virtuelle Realität einbinden oder die Realität um virtuelle Inhalte erweitern [1]. Ein Teilgebiet von diesen bildet das location-based Gaming, also das positionsbasierte spielen. Während das Genre selbst bereits in den 2000er Jahren wissenschaftliche Betrachtung fand, wird der Markt für diese Spiele erst in den letzten Jahren relevant. Spiele wie Ingress oder das während der Anfertigung dieser Arbeit vorgestellte Pokémon GO verdeutlichen ein Interesse von Spielern am location-based Gaming. Zusätzlich werden Spiele wie Super Mario Maker, in denen die Spieler selbstständig die Spielinhalte erzeugen, immer erfolgreicher [22].
In dieser Arbeit wird die Entwicklung des location-based Games „Doodlings“ beschrieben, welches anhand der Geoposition des Spielers die reale Welt um virtuelle Inhalte erweitert. Das Hauptaugenmerk liegt dabei auf der Entwicklung des Spiels für und mit dem Spieler. Der Spieler soll selbstständig Spielinhalte erzeugen und somit das Spiel weiterentwickeln können. Dabei wird das Spielprinzip von dem bekannten Spiel Pokémon inspiriert und erweitert dieses um die oben genannten Aspekte.
Die Arbeit beinhaltet sowohl einen theoretischen als auch einen empirischen Teil. Neben der herangezogenen Literatur wird in dieser Arbeit auch eine qualitative Forschung durchgeführt. Hierbei fungieren Probanden als Testpersonen für das zu entwickelnde Spiel und füllen anschließend einen Fragebogen aus, indem unter anderem das individuelle Spielerlebnis bewertet werden kann und Verbesserungsvorschläge gegeben werden können. Auf diese Art und Weise wird das Spielprinzip, welches den Spieler selbst in den Mittelpunkt der Entwicklung stellt, weitergeführt. Die Spieler können dadurch bereits in der Entwicklungsphase des Spiels aktiv an der Gestaltung von Spielinhalten teilnehmen. Im Auswertungsteil werden die Ergebnisse der Befragungen anschließend dargestellt und analysiert.

1.1 Ziel der Arbeit

Das Ziel der Arbeit ist die genaue Definition der Spielidee für Doodlings sowie die Erzeugung der ersten Spielversionen. Dazu werden zunächst die geplanten Spielmechaniken detailliert erläutert und die Anforderungen an das Spiel aufgrund der Spielmechaniken definiert. Danach wird das Spiel inkrementell entwickelt und mithilfe der empirischen Studien die

Spielidee und die Umsetzung dementsprechend verfeinert, sodass das Spiel den Spielern das bestmögliche Erlebnis bieten kann.

1.2 Gliederung

Die Arbeit ist in neun Kapitel unterteilt. Im Folgenden werden die Themen der Kapitel kurz zusammengefasst:

1. Einleitung – Führt in das Thema und das Ziel der Arbeit ein und gibt einen kurzen Überblick über den aktuellen Stand der Technik.
2. Grundlagen – Beschreibt die Grundlagen auf denen die Arbeit basiert und die das Fundament der Entwicklung bilden.
3. Spielidee – Erläutert die Idee und die grundlegenden Mechaniken des Spiels „Doodlings“.
4. Anforderungen und Vorgehen – Definiert die Anwendungsfälle und Anforderungen die an das Spiel gestellt werden und beschreibt diese detailliert. Erläutert außerdem das Vorgehen bei der Entwicklung.
5. Erstes Inkrement – Betrachtet den Entwurf, die Umsetzung, den Test und die Auswertung der ersten Ausbaustufe des Spiels.
6. Zweites Inkrement – Betrachtet den Entwurf, die Umsetzung, den Test und die Auswertung der zweiten Ausbaustufe des Spiels.
7. Fazit und Ausblick – Fasst die innerhalb der Arbeit erhaltenen Erkenntnisse zusammen, bewertet diese Anhand der Zielsetzung und gibt einen kurzen Ausblick auf die mögliche Weiterentwicklung.

1.3 Themenabgrenzung

In dieser Arbeit wird das Hauptaugenmerk hauptsächlich auf die Entwicklung des Kernsystems und die Evaluation des Spielprinzips gelegt, daher werden Themen wie die Grafik oder die Soundunterlegung nur als Randaspekte betrachtet. Außerdem werden wirtschaftliche Aspekte, wie zum Beispiel die Planung der Monetarisierung oder der Veröffentlichungskanäle aus Gründen des Umfangs nicht in diese Arbeit aufgenommen.

2 Grundlagen

Im Folgenden werden die Grundlagen des zu entwickelnden Spiels beleuchtet. Zunächst wird das Genre von Doodlings genauer beschrieben und zu beachtende Besonderheiten bei der Entwicklung von Spielen dieses Genres herausgestellt. Außerdem wird der Begriff „User Created Content“ erläutert und die dadurch entstehenden Möglichkeiten für die Verbesserung der Skalierbarkeit von location-based Games beleuchtet.
Danach wird die Plattform, für die Doodlings entwickelt wird, näher betrachtet und einige Architekturmuster für die Erstellung des Spiels diskutiert. Als letztes wird ein Einblick in die Spieleserie Pokémon gewährt, welche die Spielidee von Doodlings inspiriert hat.

2.1 Location-based Games

In location-based Games fließt die Position und zum Teil die Fortbewegung des Spielers als wichtiges Spielelement in das Spiel ein. Hierfür wird eine Technologie zum Erkennen der aktuellen Position des Spielers benötigt (zum Beispiel GPS oder netzwerkbasierte Standorterkennung) [25]. Diese Anforderung hat zur Folge, dass moderne location-based Games zumeist Mobile Games sind, also auf einem mobilen Endgerät gespielt werden. Diese werden als Geogames [24] oder Mobile location-based Games [17] bezeichnet. Mobile location-based Games sind wie folgt definiert:

> *„Ein MLBG ist ein ortsbasiertes Spiel, das auf einem mobilen Endgerät lauffähig ist und durch die Nutzung eines Kommunikationskanals Informationen mit einem Spieleserver oder Mitspielern austauschen kann.“*
> [17]

Als erster Vertreter der Kategorie location-based Games gilt das Spiel Geocaching[1] aus dem Jahr 2000. Dabei geht es darum, Verstecke (sogenannte Caches) anderer Spieler zu finden, für die nur die GPS-Positionen angegeben sind [24].
Die kognitiven Anforderungen an den Spieler werden durch location-based Games erweitert. Hierzu werden drei Bezugsräume unterschieden:

[1] www.geocaching.com

- Figurenraum
  - Ist haptisch und visuell erkundbar.
- Blickraum
  - Ist von einem Standpunkt aus visuell erkundbar.
- Umweltraum
  - Ist nur durch Fortbewegung erkundbar.

Während klassische Computerspiele nur den Figurenraum abdecken – z.B. durch die Bedienung eines Controllers – wird mittlerweile auch der Blickraum immer häufiger für Spiele verwendet. Ein Beispiel hierfür sind Spiele mit Unterstützung von Bewegungscontrollern, wie die Kinect oder PlaystationMove. Location-based Games führen diese Entwicklung fort und beziehen auch den Umweltraum als Mittelpunkt in das Spielerlebnis mit ein. Allerdings werden die beiden anderen Bezugsräume für das Spielerlebnis dennoch benötigt (z.B. für das Finden eines versteckten Caches beim Geocaching) [24].
Für das Design des Spiels entstehen durch die Erweiterung um den Umweltraum neue Herausforderungen. Zunächst die Herausforderung der lokomotorischen Exploration, also der Entdeckung der Spielwelt durch körperliche Fortbewegung. Hierbei muss das Spieldesign dem Zeit- und Kraftverbrauch des Spielers Rechnung tragen und Lösungen anbieten, die dem Spieler das Spielen im Rahmen seiner verfügbaren Zeit und Kraft ermöglichen. Des Weiteren ist das Problem der ortsbezogenen Affordanz zu lösen. Mit Affordanz wird die Eignung von räumlichen Objekten für Handlungen bezeichnet. Für das Spieldesign bedeutet dies, dass Spielhandlungen an die verwendeten Lokationen angepasst werden müssen; beispielsweise ist das Begehen eines Baudenkmals nicht überall, sondern nur an bestimmten Orten möglich. Zuletzt bleibt die Herausforderung der Umweltkontingenz, die sich im Gegensatz zu rein virtuellen Spielwelten nicht vorab planen lässt. So könnte beispielsweise einsetzender Regen direkten Einfluss auf bestimmte Spielelemente nehmen. Dies könnte etwa zur Folge haben, dass die Fortbewegungsgeschwindigkeit von Spielern abnimmt oder bestimmte Lokationen nicht mehr erreichbar sind [24].

2.1.1 Pervasive Games

Pervasive Games (deutsch: durchdringende Spiele) sind ein Teilbereich des Pervasive Computing und eine Unterkategorie der location-based Games, welche sich damit beschäftigt, die reale und virtuelle Welt in Spielen zu verbinden. Das heißt, das Spiel kann zu jeder Zeit und unabhängig von der Position des Spielers gespielt werden [25]. Eine Definition von Pervasive Games lautet:

> *"Pervasive gaming implies the construction and enactment of augmented and/or embedded game worlds that reside on the threshold between tangible and immaterial space, which may further include adaptronics, embedded software, and information systems in order to facilitate a "natural" environment for game-play that ensures the explicitness of computational procedures in a postscreen setting."*
> [14]

Aus der Definition geht hervor, dass es in Pervasive Games immer eine Spielwelt gibt, welche die reale Welt um virtuelle Elemente erweitert.
Um diese allgemeine Definition zu erweitern, definiert Kampmann Walther [14] vier Achsen, die zusammen die möglichen Bereiche für Pervasive Games markieren. Diese sind Verteilung, Mobilität, Persistenz und Transmedialität. Verteilung ist im Kontext des Pervasive Gaming deswegen wichtig, weil zumeist mobile oder eingebettete Endgeräte mit einer ubiquitären Netzwerkinfrastruktur verbunden sind und untereinander kommunizieren. Mobilität beschreibt die Mobilität der Geräte und Bestandteile des Systems. Von besonderem Interesse in dieser Arbeit ist die Persistenz, also die zeitliche und räumliche totale Verfügbarkeit der virtuellen Realität. Transmedialität ist die Vermittlung von Inhalten über unterschiedliche Arten von Medien. Sie gibt dem Nutzer die Möglichkeit, über unterschiedliche Wege am Pervasive Gaming teilzunehmen und hilft dadurch, eine Aura des Amüsements zu erzeugen [14].
Kombiniert ergibt sich der sogenannte „Pervasive Gameing Possibility Space“, also der Raum, der das Potential von Pervasive Games kennzeichnet [14].

2.2 User Created Content

Durch die Entwicklung des Internets zum Web 2.0 rückte die Erstellung von Inhalten durch die Benutzer stärker in den Vordergrund. Der Begriff Web 2.0 ist dabei nicht klar definiert und wird häufig auch lediglich als ein Marketing-Schlagwort gesehen. Unter diesem Begriff werden jedoch Internetdienste zusammengefasst, in denen die Nutzer aktiv an der Wertschöpfung des Dienstes teilhaben. Diese Dienste stellen zugleich eine Plattform für die Zusammenarbeit und Interoperabilität bereit, um diese überhaupt erst zu ermöglichen [16]. Die Wertschöpfung der Nutzer wird dementsprechend als User Created Content (UCC) bezeichnet.
Auch für User Created Content gibt es keine allgemein anerkannte Definition, deswegen wird sich in dieser Arbeit auf folgende Definition zu UCC bezogen:

> *‘‘i) content made publicly available over the Internet, ii) which reflects a certain amount of creative effort, and iii) which is created outside of professional routines and practices.”*
> [29]

Für UCC gilt also zunächst, dass die erstellten Inhalte öffentlich durch das Internet zugänglich sein müssen. Der zweite Punkt besagt, dass ein gewisser kreativer Einsatz erforderlich ist. Demnach ist das einfache Hochladen eines Fotos noch kein Erstellen von UCC. Zuletzt wird UCC nicht professionell erzeugt, es steht also kein Monetarisierungsgedanke hinter der Erzeugung von UCC. Der Autor dieser Definition sieht allerdings einen Trend zur Monetarisierung von UCC, wenngleich dies häufig nicht das Hauptargument für die Bereitstellung von UCC ist [29].
Durch die Erweiterung der Internetdienste um UCC werden diese interaktiver, dezentraler und dynamischer [16].

Neben den bekannten Webdiensten, die auf dem Prinzip des UCC beruhen (z.B. Wikipedia, YouTube, Facebook etc.), ist UCC in Spielen ein weit verbreitetes Mittel, um Spielinhalte zu schaffen. Aus diesem Grund existiert mittlerweile eine große Anzahl an Werkzeugen, um Spiele zu erweitern (z.B. den Steam Workshop[2]).
Im Besonderen gilt dies für location-based Games. So finden sich in vielen Spielen Möglichkeiten, für die Spieler Inhalte zu erzeugen, die zum Teil sogar im Spiel selbst integriert sind. Beim oben beschriebenen Geocaching zum Beispiel werden alle Spielinhalte von den Spielern selbst erzeugt. Auch bei dem von NianticLabs entwickelten, aktuell beliebten (zwischen 10 und 50 Millionen Installationen [20]) Spiel Ingress[3] werden die Lokationen für die Spielinhalte von den Spielern selbst gesucht. Dies geschieht direkt aus der Applikation für das Spiel. Neben der bloßen Erstellung von Spielinhalten geht Geocaching noch einen Schritt weiter und überlässt sogar die Überwachung und Pflege dieser Inhalte den Usern.

2.2.1 Skalierbarkeit von location-based Games durch UCC

In diesem Abschnitt werden die Besonderheiten beschrieben, in denen die Skalierbarkeit von location-based Games durch UCC gewährleistet wird. Hierbei wird darauf eingegangen, welche Regeln zu beachten sind, damit das Spiel von Spielern über einen langen Zeitraum gespielt und ein Zuwachs von Spielern skaliert werden kann. Eine Studie von Neustaedter [19], welche das Spiel Geocaching im Bezug auf diese Punkte beleuchtet, wird im Folgenden beschrieben und Schlüsse für die Entwicklung eines skalierbaren location-based Games gezogen. Geocaching ist wie beschrieben, der älteste Vertreter von location-based Games und erfreut sich noch heute großer Beliebtheit und einer großen Community (2.715.026 Caches und 6 Millionen Spieler [11]). Hierzu wird zum einen die Erstellung von neuen Spielinhalten und zum anderen die Überwachung und Orchestrierung von Spielinhalten betrachtet.

2.2.1.1 Erstellung von Spielinhalten

Anhand der Erkenntnisse der Studie wurden für die Erstellung von Spielinhalten folgende Regeln aufgestellt:

1. Es muss für die Spieler von Beginn an leicht sein, neue Spielinhalte zu erzeugen. Hierfür kann es Spielregeln geben, die vom Spieler akzeptiert werden müssen.
2. Es muss Spielern ermöglicht werden, aufwändige Spielinhalte zu erzeugen.
3. Spieler müssen die Bräuche des Spiels verstehen und diese erweitern können, um neue Spielerfahrungen erzeugen zu können.

Die erste Regel sorgt dafür, dass zum einen immer neue Lokationen zum Spielerlebnis hinzugefügt werden und somit erfahrene Spieler vor neue Herausforderungen stellen, zum anderen aber auch neue Spieler direkt eine große Anzahl von Möglichkeiten haben, am Spiel

[2] http://steamcommunity.com/workshop/?l=german
[3] https://www.ingress.com/

teilzunehmen. Außerdem wird Spielern durch die Möglichkeit, Spielinhalte zu erzeugen, das Gefühl gegeben, sie seien ein Teil des Spiels [19]. Des Weiteren wird das beschriebene Problem der ortsbezogenen Affordanz verringert, da der Ersteller der Spielinhalte direkt vor Ort ist und somit die Affordanz häufig besser einschätzen kann als ein entfernter Spielleiter oder gar ein Algorithmus. Ein Nachteil der einfachen Erzeugung von Spielinhalten ist, dass diese Elemente als uninteressant oder sogar störend von anderen Spielern wahrgenommen werden können [19].
Die zweite Regel macht den Reiz für diejenigen Spieler aus, die viel Zeit in das Spiel investieren wollen und die Herausforderung suchen. Dies trifft sowohl auf die Erstellung als auch die Lösung der aufwändigen Spielinhalte zu. Ein Spiel ohne diese Inhalte wird häufig als trivial und langweilig empfunden. Durch das Einbeziehen von Spielern in die Erzeugung dieser Spielinhalte wird dafür gesorgt, dass genügend entsprechende Inhalte vorhanden sind [19].
Die letzte Regel wiederum ist eine sehr stark an Geocaching angelegte Regel. Diese besagt, dass es für Spieler durch Lernen der Bräuche einfacher wird, die Herausforderungen des Spiels zu lösen und damit ein persönliches Erfolgserlebnis und das Gefühl von Verbesserung einhergehen. Als ein Brauch beschreibt Neustaedter dabei z.B. die Caches unter Zweigen zu verstecken oder magnetische Halterungen zu verwenden, wenn sehr viel Metall am Ort des Caches vorhanden ist. Zusätzlich müssen diese Bräuche erweitert werden können, um die Spielerfahrung zu bereichern [19].

2.2.1.2 Überwachung und Orchestrierung von Spielinhalten

Auch für die erstellten Spielinhalte und deren Überwachung wurden in der Studie Regeln aufgestellt:

1. Es muss den Spielern möglich sein, durch einfache Aktionen die Spielinhalte zu überwachen.
2. Spieler müssen einfach miteinander über die zu überwachenden Spielinhalte kommunizieren können.
3. Spieler müssen auch Spielinhalt gefährdende Aktivitäten nicht spielender Menschen überwachen und an andere Spieler weitergeben können.
4. Spielinhalte müssen von den Spielern in Stand gehalten und entfernt werden können.

Alle vier Regeln zielen darauf ab, dass die Aufgabe für das Erhalten, Verbessern und Entfernen der Spielinhalte von der Community selbst übernommen wird und nicht eine externe Instanz (z.B. der Administrator) eingreifen muss. Dabei liegt der Fokus darauf, den Ablauf des Meldens und der Kommunikation einfach zu gestalten und entsprechende Aktionen der Spieler zu belohnen. Somit wird gewährleistet, dass defekte Spielelemente schnell erfasst und entweder repariert oder entfernt werden können [19].

2.3 Smartphones und Android

Mit dem Smartphone ist im letzten Jahrzehnt eine neue Geräteklasse entwickelt worden, welche die Mobilität der Kommunikation stark erweitert hat. Dabei ist der Begriff Smartphone wie folgt definiert:

> *„Mobiltelefon mit erweitertem Funktionsumfang. Dazu zählen neben der Telefonie und Short Message Service (SMS) üblicherweise Zusatzdienste wie Electronic Mail (E-Mail), World Wide Web (WWW), Terminkalender, Navigation sowie Aufnahme und Wiedergabe audiovisueller Inhalte. Auf Smartphones laufen gegenüber herkömmlichen Mobiltelefonen komplexere Betriebssysteme wie etwa Symbian OS, Blackberry OS oder das iPhone OS. Die hierdurch geschaffene Möglichkeit zur Installation weiterer Applikationen durch den Endnutzer verleiht Smartphones einen erweiterbaren und individualisierbaren Funktionsumfang.“*
> [26]

Durch die stark ansteigende Verbreitung von Smartphones in den letzten Jahren (vgl. Abbildung 1) wird einem großen Kreis von Nutzern die Möglichkeit geboten, location-based Services zu nutzen. Benötigte man zuvor häufig mehrere Geräte für die Navigation und Netzwerkverbindung, ist nun alle Funktionalität und Rechenkapazität, die benötigt wird, zugänglich und verfügbar. Location-based Services, also Dienste, die den Standort des Benutzers mit einbeziehen, sind mittlerweile weit verbreitet und gewinnen an Relevanz (vgl. Abbildung 2).
Das von den Marktanteilen stärkste Betriebssystem für Smartphones ist mit großem Abstand Android (vgl. Abbildung 3). Deshalb wird dies die grundlegende Plattform für die Entwicklung des Spiels sein. Android wird von Open Handset Alliance entwickelt und basiert auf einem Linux Kernel. Es wird als freie Software vertrieben und quelloffen entwickelt. Dies ist der Grund, warum es von vielen Smartphone Herstellern als Betriebssystem gewählt wird und somit einen großen Marktanteil hat.
Für die Entwicklung von location-based Games bietet Android viele Vorteile, da es von vielen Game-Engines unterstützt wird und abstrahierte Methoden zum Zugriff auf die Hardware des Smartphones, beispielsweise den GPS-Sensor, bietet.

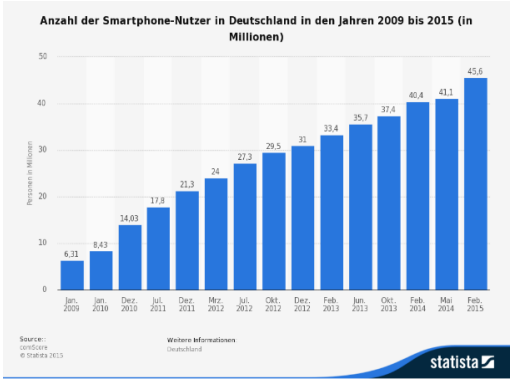

Abbildung 1 Anzahl der Smartphonenutzer

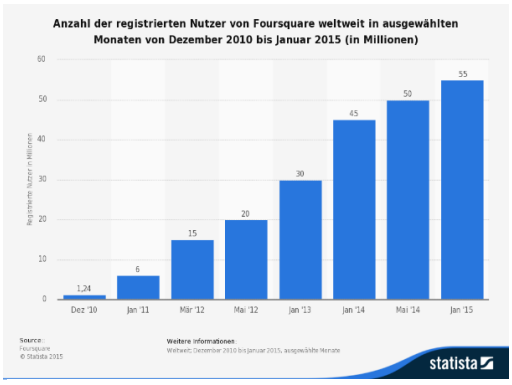

Abbildung 2 Anzahl der Nutzer von Foursquare

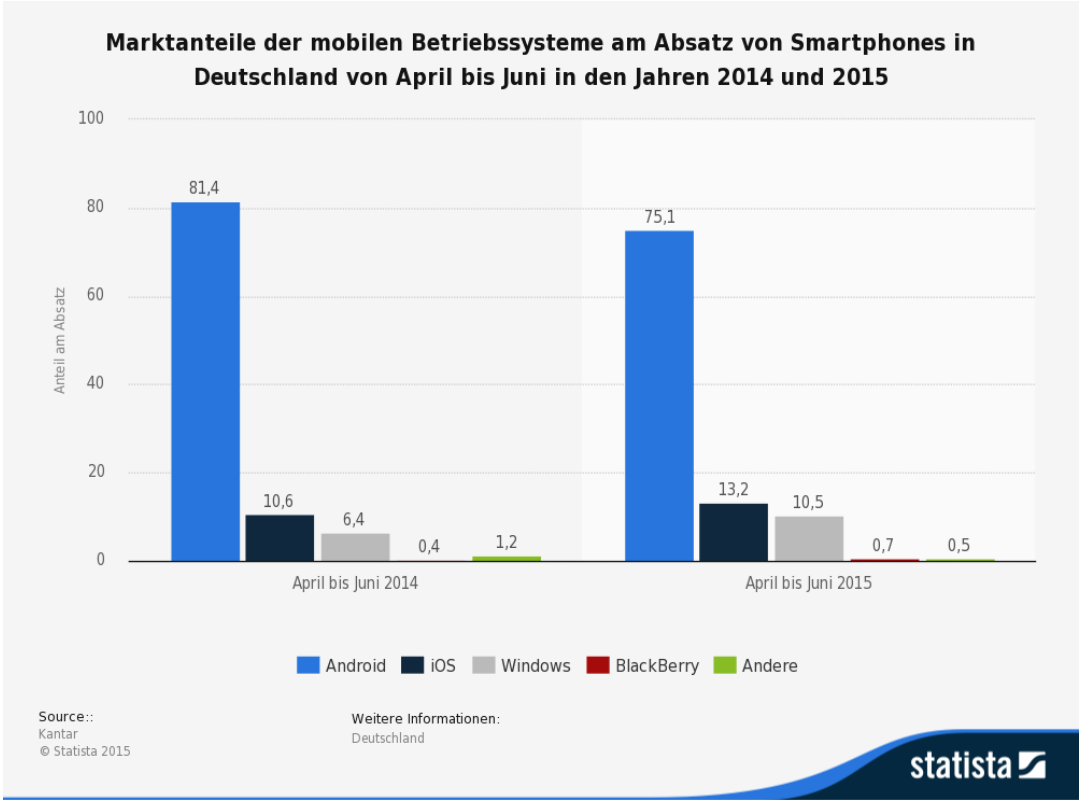

Abbildung 3 Marktanteile mobile Betriebssysteme

2.4 Multiplayer Game Client/Server Architektur

Für die Entwicklung eines Multiplayer Spiels bietet sich eine Mischung aus Client/Server und Peer-To-Peer Architektur an. Hierfür wurden bereits grundlegende Studien getätigt und für Multiplayer Spiele beispielhaft spezifiziert. Die gewonnenen Erfahrungen werden in die Architektur von Doodlings mit einfließen. Zunächst wird das allgemeine Client/Server Modell beschreiben. Danach gehe ich auf die Ausarbeitungen für die Architektur von mobilen Multiplayer Spielen ein.

2.4.1 Client/Server Architektur

Das Client/Server Modell definiert eine Abhängigkeit zwischen einem Dienstnachfragenden (Client) und einem Diensterbringer (dem Server). Der Server als Diensterbringer erwartet Anfragen - sogenannte Requests - von Clients, verarbeitet diese und sendet die Antwort (Reply) auf die Anfrage zurück an den Client. Die Kommunikation zwischen Clients findet dabei allein über den Server und dessen Zustand statt, untereinander gehen Clients keine

Verbindungen zueinander ein [4]. Clients und Server bilden dabei ein verteiltes System, in welchem der Server als Produzent und die Clients als Konsumenten fungieren.
Im Kontext von Multiplayer Spielen hält der Server den aktuellen Zustand des Spiels, während die Clients nur für die Anzeige des Spieles und Aufnahme der Spielereingaben verantwortlich sind [15].

2.4.2 Peer-To-Peer Architektur

Im Peer-To-Peer Modell wird der Zustand des Systems von allen beteiligten Entitäten – den sogenannten Peers - gemeinsam gehalten. Diese kommunizieren untereinander, d.h. eine Änderung des Zustands eines Peers muss an alle anderen Peers weitergegeben werden, um die Konsistenz des Gesamtsystems zu erhalten. Im Vergleich zum Client/Server Modell ist diese Art der Verteilung robuster und skalierbarer. Zum einen, weil der Single-Point-Of-Failure entfällt, zum anderen, weil die einzelnen Peers untereinander kommunizieren und nicht den Kommunikationsweg über den Server gehen müssen. Schwierigkeiten bei dieser Architektur sind die Konsistenzhaltung und die Komplexität der Entwicklungsaufgaben für ein Peer-To-Peer System [12].

2.4.3 Mirrored-Server Architektur

Die Mirrored-Server Architektur ist eine Mischung aus Peer-To-Peer und Client/Server. In der Client/Server Architektur ist der Server ein Single-Point-Of-Failure und skaliert schlecht bei wachsendem Rechenaufwand. Um diese Probleme zu umgehen gibt es die Mirrored-Server Architektur, die eine Duplizierung des Servers vorsieht. Dabei wird der Zustand auf mehrere Server dupliziert und von den Servern untereinander synchronisiert. Es entstehen also mehrere Server-Peers, die alle aus der Sicht des Clients identisch sind. Dieser Ansatz verlangt zusätzlichen Entwicklungs- und Rechenaufwand für das Vermeiden von Inkonsistenzen [12].

2.4.4 Clustered-Server Architektur

Im Gegensatz zur Mirrored-Server Architektur, in der alle Server für den gesamten Systemzustand verantwortlich sind, wird in der Clustered-Server Architektur der Systemzustand in Regionen unterteilt. Jeder Server kennt dann genau seine Region des Zustandes, aber nicht den Gesamtzustand [12]. Dieser Ansatz lässt sich besonders gut auf Spiele abbilden, da der Gesamtzustand meistens die Spielwelt ist und diese häufig in sogenannte Regionen unterteilt ist. Für location-based Games könnte dies beispielsweise bedeuten, dass einer der Clustered-Server für Europa und ein anderer für Asien verantwortlich ist. Die Architektur ist besser skalierbar, da beim Vergrößern des Gesamtzustands (beispielsweise durch Hinzufügen von weiteren Kontinenten) einfach ein neuer Cluster-Server hinzugefügt werden kann. Bei erhöhtem Rechenaufwand (beispielsweise durch erhöhtes Clientaufkommen) können Regionen geteilt und durch Hinzunahme neue Clustered-Server skaliert werden.
In dieser Architektur kommunizieren die Clients immer mit dem Server, der gerade für sie verantwortlich ist. Daraus folgt das Problem, dass wenn ein Client an der Grenze zwischen

zwei Regionen unterwegs ist (beispielsweise zwischen Europa und Asien), der zuständige Server häufig wechselt. Um dieses Problem zu umgehen, gibt es die sogenannten Common-Regions (deutsch: gemeinsame Regionen). Diese Regionen bilden jeweils die Grenzregionen zwischen zwei Server-Regionen. Diese Common-Regions werden von beiden angrenzenden Servern verwaltet und synchronisiert. Erst wenn ein Client diese Region verlässt, wird der Wechsel des zuständigen Servers vollzogen [12].

2.5 Pokémon Spielprinzip

Im Folgenden wird kurz auf das Spielprinzip der Pokémon Spiele eingegangen, da dieses die Spielidee von Doodlings inspiriert hat.
Die ersten Pokémon Spiele wurden bereits 1996 in Japan für den GameBoy von Nintendo veröffentlicht und sind für die Firma Nintendo bis heute eines der wichtigsten Franchises überhaupt. Das meistverkaufte Spiel für den Nintendo 3DS (einem Nachfolger des GameBoy) ist das Spiel Pokémon X bzw. Pokémon Y (vgl. Abbildung 4).
Das Spiel ist laut Hersteller in der Kategorie Rollenspiel und Abenteuer angesiedelt. Das Prinzip des Spiels beschreibt Nintendo wie folgt:

> *„Deine Aufgabe: Sammle alle 150 Pokémon der Inseln, indem du sie mit deinen Pokémon besiegst und fängst. Auf deinem Weg wirst du von zahlreichen Pokémon-Trainern - wie deinem berüchtigten Erzrivalen Gary und vielen anderen - zu Duellen herausgefordert. Du kannst schlecht absagen, also sei auf einiges gefasst. Bedenke, dass Pokémon keine gewöhnlichen Wesen sind. Um deinen Pokédex füllen zu können, musst du die gefangenen Pokémon trainieren, damit sie sich entwickeln. So kannst du einen gefangenen Raupy zu einem Safcon und schließlich zu einem Schmettbo machen. Jede Verwandlung verleiht den Pokémon mehr Kraft - wodurch du andere Pokémon besiegen und fangen kannst.“*
> *[21]*

Die Hauptaufgabe ist also Pokémon zu sammeln und diese zu trainieren. Pokémon sind hierbei fiktive Wesen mit bestimmten Eigenschaften, die mithilfe von Training verbessert werden. Durch diese Verbesserung werden die Pokémon stärker, lernen neue Fähigkeiten, die im Kampf verwendet werden können und verwandeln sich in sogenannte Entwicklungsstufen. Hierzu werden zu den Pokémon sogenannte Level angegeben, je höher der Level desto stärker das Pokémon. Außerdem werden Pokémon in bestimmten Gruppen kategorisiert. Diese Gruppen haben häufig Vor- bzw. Nachteile gegenüber anderen Gruppen, welche im Kampf beispielsweise zu höheren Schadenswerten führen können.
Das Pokémon-Sammeln findet in einer fiktiven Welt statt, in welcher der Spieler einen Spielercharakter, den sogenannten Pokémon Trainer, steuert und entweder zufällig oder in die Geschichte eingebunden auf Pokémon stößt. Diese verwickeln ihn in einen Kampf, an dessen Ende er die sogenannten wilden Pokémon, also solche ohne Besitzer, einfangen kann. Ein Kampf findet rundenbasiert im Modus eins gegen eins statt. Hierbei werden abwechselnd von den Pokémon Fähigkeiten ausgewählt. Diese fügen entweder Schaden am Gegner zu

(Abzug von Lebenspunkten), erhöhen die Stärke des eigenen Pokémon oder schwächen das Pokémon des Kontrahenten. Sobald ein Pokémon verliert, also die Lebenspunkte auf null sinken, scheidet es aus dem Kampf aus. Danach kann ein Trainer bis zu fünf weitere Pokémon einsetzen. Scheiden alle aus dem Kampf aus, ist der Kampf verloren.

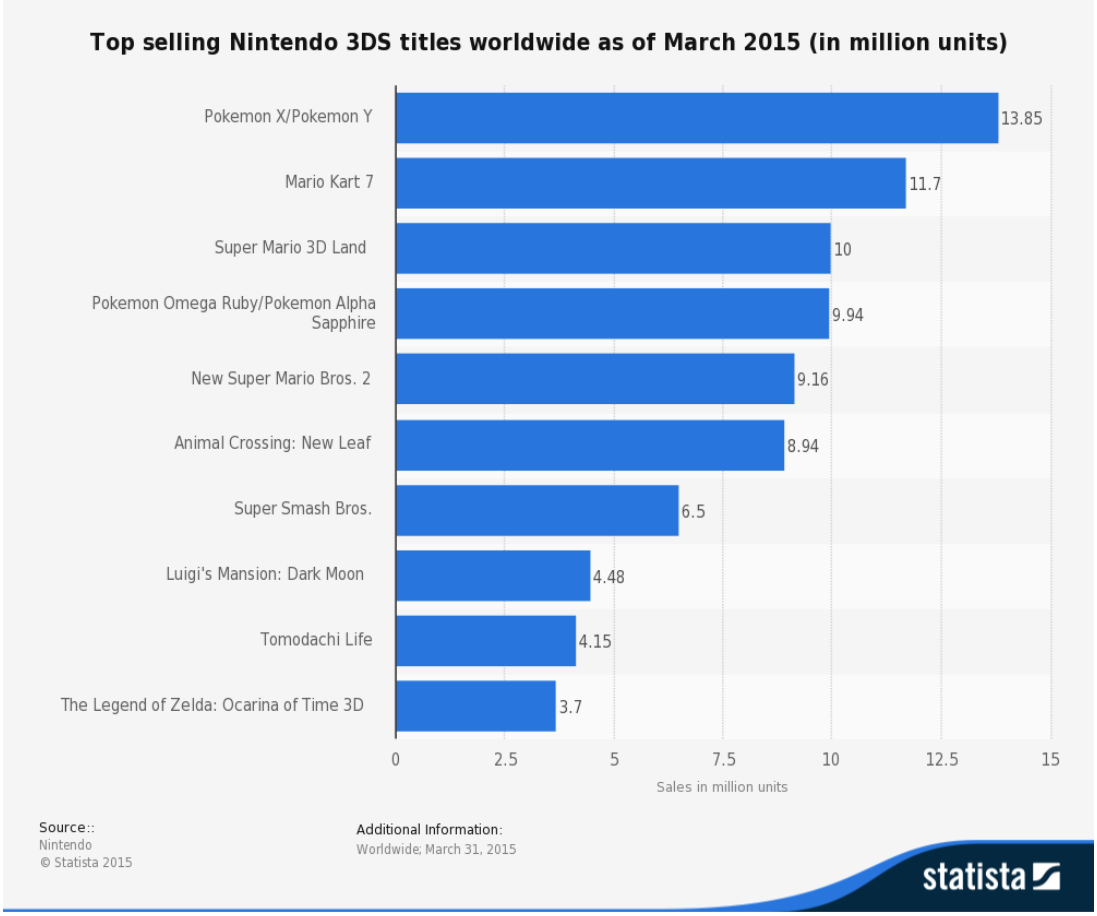

Abbildung 4 Verkaufszahlen Nintendo 3DS Spiele

3 Spielidee

3.1 Beschreibung

Doodlings soll vom Genre eine Mischung aus location-based und Massivly-Multiplayer-Online Spiel sein. Hauptaufgabe des Spielers ist es, durch den Besuch von realen Orten Doodlings zu sammeln, diese zu trainieren, gegen andere Spieler anzutreten oder eigene Herausforderungen für andere Spieler zu erstellen. Das Spiel soll als App auf Smartphones lauffähig sein, sodass keine weiteren Geräte benötigt werden.

3.1.1 Doodling

Doodlings (von „to doodle“; englisch für kritzeln) sind die virtuellen Haustiere des Spielers und das Hauptelement des Spiels. Wie in den Pokémon Spielen können sie trainiert werden und gegen andere Doodlings antreten. Es gibt unterschiedliche Arten von Doodlings (z.B. Feuer, Wasser, Erde, Luft etc.). Dem Doodling stehen dann, entsprechend der Klasse, Fähigkeiten zur Verfügung. Außerdem gibt es weitere Eigenschaften, die alle Klassen von Doodlings besitzen (Stärke, Agilität, Intelligenz etc.). Es soll möglich sein, jeden Doodling in jeder dieser Eigenschaften individuell zu verbessern. Dabei hat jedes Doodling individuelle Merkmale, die den Lernerfolg erleichtern oder erschweren. Diese Merkmale werden für jedes Doodling am Anfang in einem für jeden Doodlingtyp bestimmten Rahmen zufällig ausgewählt, sodass sich auch Doodlings des gleichen Typs unterscheiden können.

3.1.2 Sammeln der Doodlings

Ein Doodling kann von jedem Spieler gesammelt werden. Der Spieler bekommt angezeigt, in welcher Richtung seiner derzeitigen Position sich Doodlings befinden. Dabei wird keine Karte, sondern nur die Richtung und die grobe Entfernung angezeigt. Dies hat den Grund, dass der Entdeckersinn der Spieler geweckt werden soll. Wenn der Spieler an dem Punkt angekommen ist, an dem sich das Doodling aufhält, muss er dort eine definierte Zeitspanne verharren, um das Doodling einzufangen. Sollten in dieser Zeit andere Spieler dazukommen, die das Doodling ebenfalls einsammeln wollen, gibt es ein Duell um dieses Doodling. Wenn mehr als zwei Spieler das Doodling einsammeln wollen, gibt es ein Turnier in dem sich immer zwei Spieler duellieren, wobei der Verlierer ausgeschieden, der Sieger weiter im Turnier ist. Der Sieger gewinnt das Doodling. Wichtig hierbei ist, dass ab dem Zeitpunkt, an dem der erste Spieler das Doodling fangen will, die restlichen Spieler nur eine Minute Zeit haben, am Ort des Geschehens einzutreffen. Ein Spieler kann so viele unterschiedliche Doodlings sammeln

wie er will, allerdings immer nur eins pro Art. Hierbei steht es dem Spieler frei, Doodlings zu tauschen, freizulassen oder zu verschenken.

3.1.3 Kämpfe

Kämpfe finden in Echtzeit statt, indem die Spieler gegeneinander kämpfen. Dabei werden die Doodlings nacheinander eingesetzt. Von den Doodlings, die dabei zur Verfügung stehen, werden vor den Kämpfen von den Spielern jeweils sechs aus ihrer Sammlung ausgewählt. Es soll möglich sein, über das Internet, aber auch persönlich, gegeneinander anzutreten. Dabei gibt es besondere Boni (Erfahrungsbonus o.Ä.), sofern persönlich gekämpft wird. Dies soll dazu führen, dass es mehr Kontakt zwischen den Spielern gibt. Während der Kämpfe können Spieler die Aktionen ihrer Doodlings auswählen. Je nach bestimmten Werten der Doodlings verändert sich das Kampfverhalten (schnelle Doodlings dürfen mehr Aktionen ausführen, starke Doodlings verursachen mehr Schaden, etc.). Dazu haben Doodlings Lebenspunkte. Sobald diese auf 0 sinken, ist das Doodling aus dem Spiel ausgeschieden. Am Ende des Kampfes werden Erfahrungspunkte an die Doodlings verteilt, die davon abhängig sind, wie lange sie am Kampf teilgenommen haben.

3.1.4 Events

Besonders seltene Doodlings werden über sogenannte Events verteilt. Wenn ein Event startet, werden alle Spieler in einem gewissen Umkreis um das Event herum benachrichtigt. Dies soll dazu führen, dass es um diese Doodlings ein „großes“ Turnier gibt, weil dort viele Spieler gleichzeitig eintreffen werden. Man könnte dies noch fördern, indem man die Zeit für das Sammeln der Doodlings von einer Minute auf beispielsweise fünf Minuten erhöht.

3.1.5 Zusätzliche location-based Features

Neben den Doodlings gibt es auch die Möglichkeit, besondere Schätze zu finden. Hierbei könnte es beispielsweise um neue Attacken, Gegenstände, um Doodlings zeitweise zu stärken oder ähnliches gehen. Dabei läuft das Suchen entsprechend dem Sammeln der Doodlings ab. Kombinieren kann man diese Suche mit Kämpfen gegen besonders starke Doodlings.

3.1.6 Doodling als Tamagotchi

Da viele Spieler ein Interesse daran haben könnten, ihre Doodlings (oder nur eines davon) als eine Art virtuelles Haustier zu halten, sollen auch die üblichen Tamagotchi-Funktionen implementiert werden. Diese können durch den Spieler aber auch ausgeschaltet werden, da einige Spieler dies als zu zeitraubend einschätzen könnten.

3.2 Player Created Content

Großes Augenmerk soll bei der Entwicklung des Spiels auf Player Created Content gelegt werden. So trägt, wie in den Grundlagen ausgearbeitet, ein Einbeziehen der Spieler in das Erstellen und Überwachen der Spielinhalte, grundlegend zur Skalierbarkeit des Spiels bei.
Für Spieler selbst soll es möglich sein, Herausforderungen für andere Spieler zu erstellen. Das heißt, dass Spieler an selbstgewählten Orten eine solche Herausforderung starten und anschließend ihre Doodlings zum Kampf gegen andere Spieler zur Verfügung stellen können. Dazu müssen sie dann einen Preis für die erfolgreiche Bewältigung aus ihrem eigenen Inventar ausschreiben. Sind alle Preise vergeben, wird die Herausforderung automatisch gestoppt, bis wieder neue Preise vorhanden sind. Mögliche Arten der Herausforderung sind einfache Kämpfe bis hin zu einer Schnitzeljagd über mehrere Punkte oder Rätsel des Spielers, der die Herausforderung erstellt hat.
Zusätzlich zu den location-based Features soll es für Spieler auch möglich sein, Vorschläge für neue Doodlings einzusenden, die dann über einen Abstimmungsmechanismus in das Spiel übernommen werden. Hierzu werden alle Parameter (Name, Art, Startwerte der Eigenschaften), die für ein Doodling benötigt werden, vom erstellenden Spieler aus vorgegebenen Werten ausgewählt.
Zusätzlich zum Erstellen der Inhalte soll es Spielern möglich sein, die location-based Spielinhalte zu überwachen. Hierbei sollen sie zum Beispiel über einfache Werkzeuge mitteilen können, ob ein Spielinhalt defekt ist oder Überarbeitung benötigt.

3.3 Bewegungsförderung

Durch das positionsbezogene Spielprinzip wird der Spieler dazu ermutigt, sich in der realen Welt zu bewegen. Diese Ermutigung soll zusätzlich gesteigert werden, indem der Trend zu bewegungsaufzeichnenden Applikationen (zum Beispiel Google Fit[4]) ausgenutzt wird [10]. Doodlings soll von diesen Apps Aufzeichnungen importieren können (Beispielsweise zurückgelegte Schritte pro Tag) und für bestimmte Leistungen den Spieler, durch zusätzliche Erfahrungspunkte, die er seinen Doodlings hinzufügen kann, belohnen.

3.4 Zielgruppe

Die Zielgruppe des Spiels sind Jugendliche und junge Erwachsene im Alter von 12 bis 35 Jahren, die in Ihrer Kindheit/Jugend mit den Pokémon Spielen aufgewachsen sind. Durch die Selbstständigkeit und Komplexität, die das Auffinden der positionsbasierten Inhalte verlangen, ist eine jüngere Zielgruppe nicht sinnvoll. Für die Verfeinerung und Anpassung der Zielgruppe werden während der Entwicklung Tests durchgeführt.

[4] https://fit.google.com/

3.5 Abgrenzung

In diesem Abschnitt wird Doodlings von ähnlichen Spielen abgegrenzt und im Bereich der location-based Games kategorisiert.

Wie bereits erwähnt, wurde die Spielidee von Doodlings von dem Spielprinzip der Pokémon Spiele inspiriert. Sie wurde allerdings weitergeführt, indem die Rollenspiel- und Abenteuerelemente des Ursprungsspiels durch location-based Features ersetzt werden. Durch diese Ersetzung wird das Spiel pervasiv und mobil. Durch die Vermischung von realen und virtuellen Inhalten soll die Motivation der Spieler erhöht werden, da ein klarer Fortschritt in der virtuellen Welt durch Aktionen in der realen Welt erkannt werden kann. Zusätzlich werden durch den Fortschritt in der virtuellen Welt neue Möglichkeiten in der realen Welt bereitgestellt. Hier bietet das Spiel mehr als die bereits erwähnten Geocaching und Ingress. Während Geocaching nur realen Inhalt bietet, der virtuell durch bestimmte Webdienste unterstützt wird, werden bei Ingress außer die Positionsdaten des Spielers keine weiteren Eigenschaften der realen Welt benötigt.

Zeitgleich mit der Ausarbeitung dieser Arbeit stellten die Entwickler der Pokémon Spiele und die Entwickler von Ingress ein Spiel namens PokémonGO[5] vor. Die bisher veröffentlichten Details zum Spielprinzip sind in vielen Punkten ähnlich zu denen von Doodlings. Allerdings setzt Doodlings stark auf von User Created Content, während diese in PokémonGO nach bisherigem Kenntnisstand nicht geplant sind. Außerdem geht Doodlings auch bei der Individualität der Doodlings bzw. Pokémon weiter.

Zusätzlich zu den bereits aufgezählten ähnlichen Spielen gibt es eine große Anzahl an location-based Games, die in wissenschaftlichen Arbeiten beschrieben werden. In der in den Grundlagen erwähnten Studie zum Thema Skalierbarkeit von location-based Games werden diese anhand von vier Eigenschaften kategorisiert und diese Kategorisierung bei den, in der Literatur meist referenzierten location-based Games, angewendet [19].

Bei den Eigenschaften handelt es sich um „type of play“, „type of content“, „player interaction“ und „who creates content“. „Type of play“ ist die Art der Spiels. Dieses kann entweder zeitlich und räumlich fixiert (fixed) oder unbeschränkt (continual) spielbar sein. Die Eigenschaft „type of content“ beleuchtet die Komplexität der Erstellung der Spielinhalte. Diese können entweder leichtgewichtig, umfangreich oder komplex sein. „Player interaction“ beschreibt das Zusammenwirken der Spieler während des Spiels. Hierbei gibt es die Möglichkeiten der synchronen und asynchronen Interaktion. Bei synchroner Interaktion müssen die Spieler das Spiel zur gleichen Zeit gemeinsam spielen. Bei der Eigenschaft „who creates content“ wird geklärt, wie die Spielinhalte erzeugt werden. Diese werden entweder technisch, vom Administrator des Spiels oder von den Spielern selbst erzeugt [19].

Die Tabelle 1 [19] zeigt die Erkenntnisse der Studie. Diese wurde um die Spiele Ingress und PokémonGO (soweit Informationen verfügbar), sowie das in dieser Arbeit beschriebene Spiel Doodlings erweitert. Hierbei steht das „X“ für die Eigenschaft des Spiels und „u“ für „zur Zeit noch unbekannt“.

[5] http://www.pokemon.com/us/pokemon-video-games/pokemon-go/

Tabelle 1 Eigenschaften ausgewählter location-based Games

Spiel	Type of play		Types of content			Player interaction		Who creates content		
	Fixed	Cont.	Light	Elaborate	Complex	Sync	Async	Admin	Player	Auto
Ambient Wood	X				X	X		X		
Savannah	X				X	X		X		
Treasure	X		X			X		X		
Feeding Yoshi		X	X				X			X
Uncle Roy	X							X		
CYSMN?	X		X							X
EyeSpy		X	X				X		X	
Anywhere	X		X					X		
Blowtooth		X	X							X
Geocaching		X	X	X		X	X		X	
Ingress		X	X				X		X	
PokémonGo		X	u	u	u	X	X	u	u	u
Doodlings		X	X	X		X	X	X	X	X

4 Anforderungen und Vorgehen

In diesem Abschnitt werden die Anforderungen für Doodlings herausgestellt. Diese werden im späteren Verlauf, nach der Umsetzung des Spiels, herangezogen und deren Einhaltung überprüft. Die Anforderungen werden dabei in zwei Bereiche unterteilt, zum einen die funktionalen und zum anderen die nicht funktionalen Anforderungen. Für die theoretischen Grundlagen der Anforderungsspezifikation wird das Buch Software Engineering von Ludewig und Lichter herangezogen [18].

4.1 Akteure

Zunächst werden die Akteure, die mit dem zu entwickelnden System interagieren, kurz beschrieben. Akteure sind dabei ein Benutzer oder ein externes System und nicht Teil des Systems [18].

- Spieler

  Wie der Name schon sagt ist der Spieler eine Person, die das Spiel spielt. Es handelt sich hierbei um Personen, die sich in der fachlichen Domäne des Spiels auskennen, aber keine speziellen technischen Hintergrundinformationen haben.
- Administrator

  Der Administrator verwaltet das Spiel. Er hat sowohl technische als auch fachliche Hintergrundinformationen.
- Authentifizierungsdienst

  Da es spezielle Benutzerdatensätze und bestimmte Berechtigungen innerhalb des Systems gibt, muss es eine Möglichkeit geben, Benutzer zu authentifizieren. Für die Authentifizierung der Spieler werden Dienste von externen Anbietern verwendet.

4.2 Anwendungsfälle

Für Beschreibung der funktionalen Anforderungen aus der Außensicht werden Anwendungsfälle verwendet. Bei einem Anwendungsfall sind immer das zu entwickelnde System und mindestens ein Akteur beteiligt. Er wird durch ein spezielles Ereignis von einem Akteur ausgelöst, der damit ein bestimmtes Ziel erreichen möchte. Dabei beschreibt der Anwendungsfall alle Interaktionen zwischen beteiligten Akteuren und dem System, die benötigt werden, um das Ziel zu erreichen und endet mit ebendiesem oder der Feststellung, dass dieses nicht erreicht werden kann [18].

Für die Notation von Anwendungsfällen wird das „Fully Dressed Use Case Template“ von Cockburn verwendet [5]. Dieses ist wie folgt aufgebaut:

> *“<the name should be the goal as a short active verb phrase>*
> *Context of use: <a longer statement of the goal, if needed, its normal occurrence conditions>*
> *Scope: <design scope, what system is being considered black-box under design>*
> *Level: <one of: Summary, User-goal, Subfunction>*
> *Primary Actor: <a role name for the primary actor, or description>*
> *Stakeholders & Interests: <list of stakeholders and key interests in the use case>*
> *Precondition: <what we expect is already the state of the world>*
> *Minimal Guarantees: <how the interests are protected under all exits>*
> *Success Guarantees: <the state of the world if goal succeeds>*
> *Trigger: <what starts the use case, may be time event>*
> ***Main Success Scenario:***
> *<put here the steps of the scenario from trigger to goal delivery, and any cleanup after>*
> *<step #> <action description>*
> ***Extensions***
> *<put here there extensions, one at a time, each referring to the step of the main scenario>*
> *<step altered> <condition>: <action or sub-use case>*
> *<step altered> <condition>: <action or sub-use case>*
> ***Technology and Data Variations List***
> *<put here the variations that will cause eventual bifurcation in the scenario>*
> *<step or variation # > <list of variations>*
> *<step or variation # > <list of variations>*
> ***Related Information***
> *<whatever your project needs for additional information>”*
>
> [5]

Anwendungsfälle werden in drei Detailierungsstufen, den sogenannten Level, eingeteilt. Die sogenannten „User-Goals“ beschreiben den genauen Ablauf eines Anwendungsfalls, während „Summarys“ diesen eher zusammenfassen. Durch das Level „Subfunction“ wird gezeigt, dass es sich dabei um einen Teil eines Anwendungsfalls handelt.
Da in dieser Arbeit aufgrund des beschränken Rahmens nicht das Gesamtsystem entwickelt werden kann, werden nur die Anwendungsfälle aufgelistet, die für das beschriebene Teilsystem relevant sind. Die Anwendungsfälle und deren Zusammenhänge sind im Anwendungsfalldiagramm (Abbildung 5 Anwendungsfalldiagramm) grafisch dargestellt.

1: Location-based Spielinhalte anzeigen
Kontext: Spielinhalte in der unmittelbaren Umgebung des Spielers anzeigen.
Hauptakteur: Spieler
Level: User-goal
Stakeholder & Interessen:

- Spieler – Anzeige der location-based Spielinhalte in der Umgebung

Vorbedingungen:

- Spieler ist angemeldet
- Spieler hat Internetverbindung

Nachbedingung bei Erfolg:

- Spieler sieht die location-based Spielinhalte in der näheren Umgebung auf seinem Device

Trigger: Spieler navigiert zur Anzeigekomponente für die location-based Spielinhalte
Erfolgsszenario:

1 Spieler navigiert zur Anzeigekomponente für die location-based Spielinhalte
2 System liest die aktuelle Position des Spielers über das Betriebssystem aus
3 System berechnet anhand der Position die Spielinhalte in einem konfigurierbaren Radius und zeigt Wartemeldung
4 System bewegt die dynamischen Spielinhalte anhand ihrer Geschwindigkeit
5 System zeigt dem Spieler die entsprechenden Spielinhalte an

Erweiterungen:

2.1. Aktuelle Position kann nicht ausgelesen werden
2.1.1. System zeigt die Nachricht „Position konnte nicht ermittelt werden“ an
2.1.2. System überprüft, ob Lokalisierungsdienste des Betriebssystems eingeschaltet sind
2.1.3.a Wenn Nein: System zeigt Aufforderung zum Einschalten
3.1. Spielinhalte können nicht berechnet werden, weil der Spieler außerhalb der erfassten Spielwelt ist
3.1.1. System zeigt Nachricht „Sie befinden sich außerhalb der Spielwelt“
3.2. Keine dynamischen Spielinhalte innerhalb des Radius vorhanden
3.2.1. System erzeugt neue dynamische Spielinhalte innerhalb dieses Radius

Zusatzinformation:
Mit location-based Spielinhalten sind alle Inhalte gemeint, mit denen der Spieler in der realen Welt interagieren kann. Also zum Beispiel Doodlings, die er sammelt, oder Schätze, die er finden kann, aber auch die von anderen Spielern erstellten Spielinhalte.

2: Fangen eines Doodlings
Hauptakteur: Spieler
Level: User-goal
Stakeholder & Interessen:

- Spieler – Will ein bestimmtes Doodling fangen

Vorbedingungen:

- Spieler ist angemeldet
- Spieler hat Internetverbindung
- Spieler hat den Anwendungsfall 1 erfolgreich durchgeführt

Nachbedingung bei Erfolg:

- Doodling ist an den Spieler gebunden
- Anderen Spielern wird dieses Doodling nicht mehr angezeigt

Trigger: Spieler entscheidet sich ein Doodling zu Fangen

Erfolgsszenario:

1. Spieler entscheidet sich ein Doodling zu Fangen
2. Spieler entscheidet sich für ein konkretes Doodling, das ihm durch Anwendungsfall 1 angezeigt wird
3. Spieler bewegt sich in Richtung des Doodlings
4. Betriebssystem informiert das System nach einer Distanz von 10 Metern oder einer Zeit von 10 Sekunden. Das System aktualisiert die Anzeige und zeigt die Richtung an, in die sich der Spieler bewegt
5. Spieler befindet sich innerhalb einer konfigurierbaren Entfernung zum Doodling
6. System blendet Anzeige zum „Doodling fangen“ ein
7. Spieler aktiviert die Anzeige „Doodling fangen“
8. System startet Timer von einer Minute
9. Timer läuft aus und benachrichtigt das System
10. System bindet das Doodling an den Spieler
11. System blendet Erfolgsmeldung „Doodling gefangen“ ein
12. System entfernt den Spielinhalt für andere Spieler

Erweiterungen:

4.1. Lokalisierungsdienst des Betriebssystems nicht verfügbar
  4.1.1. System zeigt die Nachricht „Position konnte nicht ermittelt werden“ an
  4.1.2. System überprüft, ob Lokalisierungsdienste des Betriebssystems eingeschaltet sind
  4.1.3.a Wenn Nein: System zeigt Aufforderung zum Einschalten

4.2. System erkennt die Richtung nicht in die sich der Spieler bewegt
  4.2.1. Anzeige für die Richtung wird ausgeschaltet
  4.2.2. Weiter bei Schritt 5

5.1. Genauigkeit der Positionserkennung ist unzureichend (mehr als 10 Meter Messungenauigkeit)
  5.1.1. System zeigt Meldung „Positionsdaten ungenau“
  5.1.2. System deaktiviert „Doodling fangen“ Anzeige

9.1. Anderer Spieler aktiviert die Anzeige „Doodling fangen“ bevor der Timer ausgelaufen ist
  9.1.1. System zeigt Meldung „Mehrere Spieler wollen dieses Doodling fangen, das kann nur ein Turnier entscheiden!“
  9.1.2. System startet Anwendungsfall 4 Schritt 5

Zusatzinformation:

3: Anzeige der Doodlings im Besitz des Spielers
Kontext: -
Hauptakteur: Spieler
Level: User-goal
Stakeholder & Interessen:
- Spieler – Will den Status seiner Doodlings abfragen

Vorbedingungen:
- Spieler ist angemeldet
- Spieler hat Internetverbindung

Nachbedingung bei Erfolg:
- Spieler sieht die in seinem Besitz befindlichen Doodlings inklusive ihrer Werte

Trigger: Spieler navigiert zum Hauptbildschirm des Spiels
Erfolgsszenario:
1. Spieler navigiert zum Hauptbildschirm des Spiels
2. System lädt die Doodlings des Spielers
3. System zeigt die Doodlings und deren Werte dem Spieler in einer Auflistung an

Erweiterungen:
Zusatzinformation:

4: Kämpfen gegen andere Spieler
Kontext: -
Hauptakteur: Spieler
Level: User-goal
Stakeholder & Interessen:
- Spieler – Will einen Kampf gegen einen anderen Spieler bestreiten
- Gegnerischer Spieler - Will einen Kampf gegen einen anderen Spieler bestreiten

Vorbedingungen:
- Spieler ist angemeldet
- Spieler hat Internetverbindung
- Spieler besitzt mindestens ein Doodling

Nachbedingung bei Erfolg:
- Doodlings des Spielers haben an Erfahrung gewonnen
- Statistik des Spielers wurde um den Kampf erweitert

Trigger: Spieler betätigt „Gegner suchen“ Schaltfläche
Erfolgsszenario:
1. Spieler betätigt „Gegner suchen“ Schaltfläche
2. System bewertet die Erfahrung des Spielers, um einen passenden Gegner auswählen zu können
3. System reiht den Spieler in die Warteschlange ein
4. System findet passenden gegnerischen Spieler
5. System zeigt Anzeige für die Kampfkonfiguration
6. System wählt die Anzahl der eingesetzten Doodlings (1 bis 6) aus

7. Spieler wählen aus ihren Doodlings diejenigen aus, die am Kampf teilnehmen sollen
8. System zeigt beiden Spielern Kampfbildschirm an
9. Spieler und gegnerischer Spieler wählen nächste Kampfaktion aus
10. System wartet bis Kampfaktion ausgeführt werden kann, basierend auf den Werten der eingesetzten Doodlings
11. System führt Kampfaktionen aus und berechnet das Ergebnis
12. Spieler und System wiederholen die Schritte neun bis elf bis ein Spieler verloren hat
13. System gibt je nach Ausgang die Meldung „Sieg“ oder „Niederlage“ aus
14. System berechnet die Erfahrungspunkte für die Doodlings
15. System zeigt die Veränderung der Erfahrungspunkte an
16. Spieler verlässt den Kampfbildschirm

Erweiterungen:

8.1. Gegnerischer Spieler beendet das Spiel oder verliert die Verbindung
  8.1.1. System zeigt Meldung „Gegner hat das Spiel verlassen“
  8.1.2. Weiter bei Schritt 13 mit Meldung „Sieg“
9.1. Gegnerischer Spieler beendet das Spiel oder verliert die Verbindung
  9.1.1. Weiter bei Schritt 8.1.1
10.1. Gegnerischer Spieler beendet das Spiel oder verliert die Verbindung
  10.1.1. Weiter bei Schritt 8.1.1
11.1. Gegnerischer Spieler beendet das Spiel oder verliert die Verbindung
  11.1.1. Weiter bei Schritt 8.1.1

Zusatzinformation:
Ein Spieler hat verloren wenn alle Doodlings dieses Spielers besiegt wurden. Wurde ein Doodling besiegt, kann der Spieler das nächste einzusetzende Doodling auswählen.

5: Erstellen eines Quests
Kontext: Ein Spieler will der virtuellen Welt von Doodlings einen Spielinhalt, genannt Quest (Aufgabe), der von anderen Spielern gesehen und erledigt werden kann, hinzufügen.
Hauptakteur: Spieler
Level: User-goal
Stakeholder & Interessen:

- Spieler – Will einen Quest für andere Spieler erstellen

Vorbedingungen:

- Spieler ist angemeldet
- Spieler hat Internetverbindung

Nachbedingung bei Erfolg:

- Erstellter Quest ist für die anderen Spieler sichtbar

Trigger: Spieler betätigt auf der Anzeigekomponente für location-based Spielinhalte die Schaltfläche „Quest erstellen“
Erfolgsszenario:

1. Spieler betätigt auf der Anzeigekomponente für location-based Spielinhalte die Schaltfläche „Quest erstellen“

2. System liest die Position des Spielers über das Betriebssystem aus
3. System überprüft, ob innerhalb eines gewissen Radius bereits andere Quests vorhanden sind
4. System zeigt Questtyp Auswahl
5. Spieler wählt Questtyp aus
6. Je nach ausgewähltem Questtyp werden die entsprechenden Anwendungsfälle ausgeführt
7. System zeigt Quest Zusammenfassung an
8. Spieler betätigt „Quest veröffentlichen“ Schaltfläche
9. System überprüft die übermittelten Daten auf Korrektheit
10. System speichert den Quest persistent in der Questdatenbank
11. System zeigt Meldung „Quest erfolgreich erstellt“

Erweiterungen:

7.1. Daten sind nicht Korrekt

9.1.1 System zeigt dem Spieler die Nachricht „Die eingegeben Daten xy sind nicht korrekt, bitte korrigiere sie.“

Zusatzinformation:
Folgende Questtypen sind zunächst geplant: Schnitzeljagt, Rätsel, Kampf. Dabei soll aber nur der Questtyp Rätsel innerhalb dieser Arbeit umgesetzt werden.

6: Erstellen eines Rätsels
Kontext: Spieler will einen Quest des Typs Rätsel erstellen
Hauptakteur: Spieler
Level: Subfunction
Stakeholder & Interessen:

- Spieler – Will ein Rätsel für andere Spieler erstellen

Vorbedingungen:

- Spieler ist angemeldet
- Spieler hat Internetverbindung

Trigger: Spieler wählt auf der Anzeige zum Quest erstellen den Questtyp Rätsel aus.
Erfolgsszenario:

1. System zeigt Maske zum Erzeugen eines Rätsels
2. Spieler gibt das Rätsel und die Antwortmöglichkeiten in Textfelder ein
3. Spieler wählt die richtige Antwort aus
4. Spieler wählt Preis für die Belohnung von korrekten Antworten aus
5. Spieler bestätigt die Eingaben

Erweiterungen:
Zusatzinformation:

7: Spielen eines Rätsels
Kontext: Spieler will einen Quest des Typs Rätsel spielen
Hauptakteur: Spieler

Level: User-goal
Stakeholder & Interessen:

- Spieler – Will ein Rätsel lösen

Vorbedingungen:

- Spieler ist angemeldet
- Spieler hat Internetverbindung
- Spieler hat Anwendungsfall 1 erfolgreich durchgeführt

Trigger: Spieler entscheidet sich einen Rätselquest zu lösen
Erfolgsszenario:

1. Spieler bewegt sich zu Geoposition des Rätsels
2. Betriebssystem informiert das System nach einer Distanz von 10 Metern oder einer Zeit von 10 Sekunden. Das System aktualisiert die Anzeige und zeigt die Richtung an, in die sich der Spieler bewegt
3. Spieler befindet sich innerhalb einer konfigurierbaren Entfernung zum Rätsel
4. System blendet Schaltfläche „Rätsel lösen“ ein
5. Spieler aktiviert Schaltfläche
6. System zeigt Rätsel und Antwortmöglichkeiten
7. Spieler wählt eine Antwortmöglichkeit
8. System deaktiviert das Rätsel für diesen Spieler
9. System ordnet dem Spieler die Belohnung zu
10. System zeigt Erfolgs oder Misserfolgsmeldung

Erweiterungen:

2.1. Lokalisierungsdienst des Betriebssystems nicht verfügbar
  2.1.1. System zeigt die Nachricht „Position konnte nicht ermittelt werden“ an
  2.1.2. System überprüft, ob Lokalisierungsdienste des Betriebssystems eingeschaltet sind
  2.1.3.a Wenn Nein: System zeigt Aufforderung zum Einschalten

2.2. System erkennt die Richtung nicht in die sich der Spieler bewegt
  2.2.1. Anzeige für die Richtung wird ausgeschaltet
  2.2.2. Weiter bei Schritt 5

3.1. Genauigkeit der Positionserkennung ist unzureichend (mehr als 10 Meter Messungenauigkeit)
  3.1.1. System zeigt Meldung „Positionsdaten ungenau“
  3.1.2. System deaktiviert „Doodling fangen“ Anzeige

Zusatzinformation:

8: Spieler meldet ein fehlerhaftes Spielelement
Kontext: Wenn zum Beispiel ein erstellter Quest unlösbar ist, kann ein Spieler diesen als Fehlerhaft melden
Hauptakteur: Spieler
Level: Summary
Stakeholder & Interessen:

- Spieler – Will, dass ein scheinbar fehlerhaftes Spielelement aus dem Spiel entfernt wird
- Administrator – Will über fehlerhafte Spielelemente informiert werden

Vorbedingungen:

- Spieler ist angemeldet
- Spieler hat aktive Internetverbindung

Nachbedingung bei Erfolg:

- System hat das Spielelement als eventuell fehlerhaft gekennzeichnet

Trigger: Spieler erkennt, dass ein Spielelement scheinbar fehlerhaft ist
Erfolgsszenario:

1. Spieler betätigt Schaltfläche „als Fehlerhaft melden“
2. System kennzeichnet das Spielelement in der Datenbasis als fehlerhaft
3. System blendet Meldung „Danke für die Mithilfe ein“

Erweiterungen:
Zusatzinformation:

9: Anzeigen von Meldungen über fehlerhafte Spielelemente
Kontext: Ein Administrator will sich von Spielern gemeldete fehlerhafte Spielelemente anzeigen lassen, um die Fehlerhaftigkeit zu überprüfen.
Hauptakteur: Administrator
Level: Summary
Stakeholder & Interessen:

- Administrator – Will Informationen über von Spielern gemeldete fehlerhafte Spielelemente

Vorbedingungen:

- Administrator ist am System angemeldet

Nachbedingung bei Erfolg:

- Liste der Fehlerhaften Spielelemente liegt dem Administrator vor

Trigger: Administrator navigiert zur Ansichtsmaske für Fehlerhafte Spielelemente
Erfolgsszenario:

1. System fragt die Datenbasis nach von Spielern gemeldeten Spielelementen ab
2. System bereitet die erhaltenen Daten zur Ansicht auf
3. System zeigt die Daten in einer Liste

Erweiterungen:
Zusatzinformation:

10: Entfernen von Spielelementen
Kontext: Ein Administrator will ein Spielelement aus dem Spiel entfernen.
Hauptakteur: Administrator
Level: Summary
Stakeholder & Interessen:

- Administrator – Will ein Spielelement löschen

Vorbedingungen:
- Administrator ist am System angemeldet
- Administrator hat Anwendungsfall 10 ausgeführt

Nachbedingung bei Erfolg:
- Spielelement ist im System nicht mehr vorhanden

Trigger: Administrator navigiert zur Löschmaske für das Entfernen von Spielelementen
Erfolgsszenario:
1. Administrator gibt die identifizierenden Informationen für das zu löschende Spielelement ein
2. Administrator betätigt die löschen Funktion
3. System entfernt Spielelement aus der Datenbasis
4. System zeigt Erfolgsmeldung

Erweiterungen:
Zusatzinformation:

11: Erzeugen von neuen Doodlingtypen
Kontext: Ein Spieler will einen neuen Doodlingtypen erstellen.
Hauptakteur: Spieler
Level: User-Goal
Stakeholder & Interessen:
- Spieler – Will einen neuen Doodlingtypen erzeugen

Vorbedingungen:
- Spieler ist am System angemeldet
- Spieler ist sind die Gegenstände „Papier“ und „Stifte“ zugeordnet

Nachbedingung bei Erfolg:
- Neuer Doodlingtyp kann vom System verwendet werden

Trigger: Spieler navigiert zur Maske zum Erstellen von Doodlingtypen
Erfolgsszenario:
1. Spieler gibt den Namen, eine Beschreibung sowie die Attributwerte für Stärke, Beweglichkeit und Intelligenz für den neuen Doodlingtyp an
2. Spieler navigiert auf die Oberfläche zum Malen des Anzeigebildes für den neuen Doodlingtyp
3. Spieler malt den Doodlingtyp
4. Spieler bestätigt seine Eingaben
5. System überprüft die eingegebenen Daten
6. System speichert den neuen Doodlingtyp
7. System zeigt dem Spieler Erfolgsmeldung

Erweiterungen:
5.1. Eingegebene Daten nicht korrekt
    5.1.1. System zeigt dem Spieler die Nachricht „Die eingegeben Daten xy sind nicht korrekt, bitte korrigiere sie.“

Zusatzinformation:

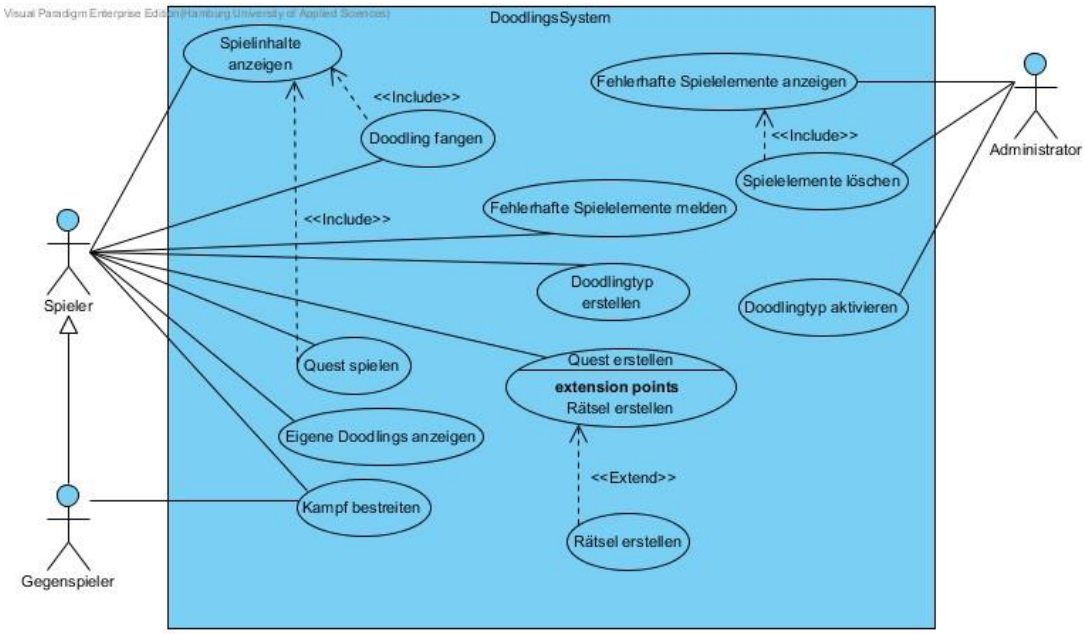

Abbildung 5 Anwendungsfalldiagramm

4.3 Funktionale Anforderungen

Aus den Anwendungsfällen können die funktionalen Anforderungen an das System abgeleitet werden. Funktionale Anforderungen spezifizieren den Nutzen des Programms [18]. In „Tabelle 2 Funktionale Anforderungen“ werden die funktionalen Anforderungen des Systems definiert und beschrieben. Es werden jeder Anforderung eine Identifikation, ein Name und eine Beschreibung zugeordnet. Zusätzlich wird in der Spalte „Anwendungsfall“ vermerkt, aus welchen Anwendungsfällen diese funktionale Anforderung abgeleitet wurde.

Tabelle 2 Funktionale Anforderungen

Id	Name	Anwendungsfall	Beschreibung
FR1	Abruf von Spielinhalten	1	Das System muss anhand einer übermittelten Position eines Spielers berechnen, welche Spielinhalte in einem konfigurierbaren Radius verfügbar sind. Diese müssen dem Spieler nach der Berechnung angezeigt werden. Werden keine Spielinhalte gefunden, müssen automatisiert Spielinhalte erzeugt werden. Es wird zwischen zwei Typen von Spielinhalten unterschieden, dynamischen und statischen. Dynamische Spielinhalte

			bewegen sich mit einer konfigurierbaren Geschwindigkeit, statische halten ihre Position. Dynamische Spielinhalte sind: Doodlings. Statische Spielinhalte sind: Quests.
FR2	Fangen von Doodlings	2	Spieler müssen Doodlings fangen können. Dafür muss der Spieler einen konfigurierbaren maximalen Abstand zu dem Doodling haben. Wenn ein Doodling von einem Spieler gefangen wurde, muss es diesem dauerhaft zugeordnet sein. Es können mehrere Spieler innerhalb einer konfigurierbaren Zeit das gleiche Doodling fangen. Nach Ablauf der Zeit wird dann von den Spielern um das Doodling gekämpft (siehe Anforderung 3). Das Doodling wird dann dem Gewinner zugeordnet.
FR3	Anzeige eigener Doodlings	3	System zeigt die dem Spieler zugeordneten Doodlings an.
FR4	Kämpfe zwischen Spielern	4	Spieler müssen mit ihren Doodlings gegeneinander kämpfen können. Das System sucht zunächst automatisiert einen etwa gleich starken Gegner. Der Kampf läuft in Echtzeit ab. Die Aktionen, die für die Spieler möglich sind, werden anhand der Werte der eingesetzten Doodlings berechnet und angezeigt. Ein Spieler gewinnt, sobald der Gegnerische Spieler keine einsetzbaren Doodlings (Lebenspunkte > 0) mehr zur Verfügung hat. Nach dem Kampf werden Erfahrungspunkte je nach Kampfteilnahme an die Doodlings verteilt und als Ergebnis angezeigt.
FR5	Erstellen von Quests	5, 6	Spieler müssen Quests erstellen können. Für die Erstellung eines Quests übermittelt der Spieler die Position des Quests sowie die Aufgabe, die der Quest beinhaltet (Schnitzeljagd, Rätsel oder Kampf) an das System. Dieses erzeugt einen statischen

			Spielinhalt für diesen Quest und zeigt ihn anderen Spielern in der Umgebung an.
FR6	Melden eines fehlerhaften Spielinhalts	8	Spieler müssen einen Fehlerhaften Spielinhalt melden können. Durch das Melden ist der entsprechende Spielinhalt als fehlerhaft markiert. Diese Markierung ist für andere Spieler sichtbar.
FR7	Administration der Spielinhalte	9, 10	Der Administrator muss die als fehlerhaft markierten Spielinhalte überprüfen und bei Bedarf löschen können. Ein gelöschter Spielinhalt darf den Spielern nicht mehr angezeigt werden.
FR8	Erstellen von Doodlingtypen durch Spieler	11	Spieler müssen Doodlingtypen erstellen können. Dafür müssen die Spieler ein Bild hochladen und die relevanten Daten (Name, Stärke, Beweglichkeit, Intelligenz) in ein Formular eingeben. Als Voraussetzung müssen zunächst Materialien gesammelt werden. Nach dem Erstellen des Doodlingtyps muss das System Doodlings dieses Typs erstellen und wie unter FR1 beschrieben anzeigen können.
FR9	Spielen eines Quests		Der Spieler muss die von anderen Spielern erstellten Quests angezeigt bekommen und die darin enthaltenen Aufgaben lösen können. Bei erfolgreicher Durchführung muss dem Spieler eine Belohnung zugewiesen werden. Bereits durchgeführte Quests dürfen dem Spieler nicht mehr angezeigt werden.

4.4 Nicht funktionale Anforderungen

Neben den funktionalen Anforderungen an das System, die das Systemverhalten beschreiben, gibt es auch nicht funktionale Anforderungen, die Bedingungen an das System stellen, wie Wartbarkeit, Performance, Bedienbarkeit etc. Diese sind häufig weich und vage. Das heißt, ihre Erfüllung ist nicht direkt mit Ja oder Nein zu beantworten und sie sind schwer objektivierbar [18]. Dies erschwert die spätere Evaluierung der Anforderungen. Nachfolgend werden in „Tabelle 3 Nicht funktionale Anforderungen“ die nicht funktionalen Anforderungen an das System so weit wie möglich hart und objektivierbar beschrieben.

Tabelle 3 Nicht funktionale Anforderungen

Id	Name	Beschreibung
NFR1	Skalierbarkeit	Das System sollte insofern skalierbar sein, als dass es für die frühen Phasen der Entwicklung auf einem Entwickler PC laufen kann und eine Spieleranzahl von 100 Spielern deutschlandweit verarbeiten kann. Da die Spieleranzahl mit der Veröffentlichung des Spiels wahrscheinlich steigen wird und das Spiel auf der ganzen Welt gespielt werden kann, ist es wichtig, dass das System auch verteilt über mehrere Server und das Internet lauffähig ist.
NFR2	Performanz	Die Antwortzeit des Systems auf alle Anfragen des Spielers sollte maximal drei Sekunden benötigen. Der Wert wird auf drei Sekunden festgelegt, da 36% der Benutzer eine längere Wartezeit für die Anzeige von mobilen Webseiten nicht tolerieren [8]. Da es sich bei der Anzeige der Spielelemente ebenfalls um mobilen Content handelt, wurde dieser Wert übernommen.
NFR3	Nutzbarkeit	Das Spiel soll ohne weitere Dokumentation von jedem Spieler der Zielgruppe gespielt und verstanden werden können. Dabei sollen vor allem der Spielablauf ohne externe Anleitungen oder ähnliches verstanden werden.
NFR4	Ressourcenverbrauch	Da mobile Endgeräte sowohl in der Rechenkapazität als auch in der Batteriekapazität eingeschränkt sind, soll das Spiel so ressourcenschonend entwickelt werden, dass es auf einem zwei Jahre alten Mittelklasse Smartphone gespielt werden kann und pro Stunde nicht mehr als 20% Akku benötigt. Ebenfalls zu beachten ist die möglichst geringe Netzwerklast, da die übertragenen Daten den Spieler Geld kosten könnten. Hier soll bei einer Spieldauer von einer Stunde nicht mehr als 10 MB durchschnittlich verbraucht werden.
NFR5	Erweiterbarkeit	Da das Spiel ständig weiterentwickelt werden soll, müssen die Entwickelten Komponenten besonders erweiterbar und unabhängig voneinander sein. Dabei soll die Implementierung der einzelnen Komponenten austauschbar sein, ohne andere Komponenten anpassen zu müssen.
NFR6	Sicherheit	Die Kommunikation zwischen Server und Client soll verschlüsselt sein. Sensible Daten der Spieler sollen so geschützt werden, dass ein Angreifer diese nicht in

		relevanter Zeit und mit relevanten Aufwand erreichen kann. Relevante Zeit bzw. relevanter Aufwand ist dabei auf die jeweiligen Daten bezogen.

4.5 Vorgehensmodell

Die Entwicklung von Doodlings findet inkrementell statt. Das heißt, dass im ersten Schritt ein Kernsystem entwickelt wird, welches Schritt für Schritt erweitert wird. Dabei ist das Ergebnis jedes Schrittes in einem Zustand, der die Auslieferung und den Einsatz ermöglichen würde[18].
Den Begriff der inkrementellen Software Entwicklung beschreiben Ludewig und Lichter wie folgt:

> *"Das zu entwickelnde System bleibt in seinem Gesamtumfang offen; es wird in Ausbaustufen realisiert. Die erste Stufe ist das Kernsystem. Jede Ausbaustufe erweitert das vorhandene System und wird in einem eigenen Projekt erstellt. Mit der Bereitstellung einer Erweiterung ist in aller Regel auch (wie bei der iterativen Entwicklung) eine Verbesserung der alten Komponenten verbunden."*
> *[18]*

Die Definition beschreibt die Vorteile dieses Systems. Zum einen wird das Projekt in kleinere, besser planbare Teilprojekte zerlegt, zum anderen bietet die Fertigstellung eines Inkrements die Chance, diese zu evaluieren, da alle Inkremente sich in einem auslieferbaren Zustand befinden [18].
Auch in der Spieleentwicklung ist ein schleifenbasiertes System für die Entwicklung von entscheidender Bedeutung. Jesse Schell definierte dafür die Schleifenregel:

> *„Je öfter Sie Ihr Design testen und verbessern, desto besser wird Ihr Spiel werden."*
> (Schell, 2012)

In der inkrementellen Softwareentwicklung wird jedes Inkrement am Ende evaluiert und bewertet, um nachfolgende Inkremente zu verbessern. Dies schließt auch die Verbesserung der alten Inkremente mit ein. Deswegen wiederspricht dieser Ansatz nicht der Schleifenregel.
Für das Projekt Doodlings bietet sich dieses Vorgehensmodell an, da erstens die Entwicklungszeit durch die Zeit zur Ausfertigung dieser Arbeit beschränkt ist und zweitens nicht sichergestellt ist, ob die Spielidee der ausgewählten Zielgruppe überhaupt gefällt beziehungsweise ob Änderungen vorgenommen werden müssen, damit das Spiel der Zielgruppe besser gefällt.

4.6 Ausbaustufen

Die vorläufige Planung der einzelnen Ausbaustufen der Entwicklung von Doodlings wird in Tabelle 4 aufgelistet. Inhalt dieser Arbeit sind lediglich die erste und die zweite Ausbaustufe. Die Ausbaustufen wurden so gewählt, dass nach der Fertigstellung der ersten Ausbaustufe die grundlegenden Elemente der Spielidee evaluiert werden können. Damit eine Spielidee frühzeitig evaluiert werden kann, sollte das Spielzeug zuerst entwickelt werden. Der Unterschied zwischen einem Spiel und einem Spielzeug ist, dass das Spielzeug aus sich selbst heraus Spaß macht, während ein Spiel zielgetrieben Spaß machen soll (Schell, 2012). Für die Auswahl des Spielzeugs gibt es innerhalb der Spielidee zwei Alternativen. Zum einen das Fangen der Doodlings, zum anderen der Kampf gegen andere Spieler. Da es für letzteres genügend aktuelle Spiele gibt, die zeigen, dass das Spielprinzip des Kampfes von Spielern gegen Spieler mit trainierbaren Monstern funktioniert (siehe Abschnitt Pokémon Spielprinzip), wird Ersteres im ersten Inkrement umgesetzt.
In der zweiten Ausbaustufe werden dann einfache Elemente für die Inhaltsgenerierung durch Spieler und das Kampfsystem für Kämpfe zwischen Spielern hinzugefügt.
Die nicht funktionalen Anforderungen sollen in allen Ausbaustufen erfüllt sein, da diese unabhängig vom Umfang des Spiels gelten müssen. Eine Ausnahme stellt dabei die Anforderung an die Sicherheit (NFR6) dar, die erst im dritten Inkrement relevant werden soll, um den Umfang der ersten beiden Inkremente zu verkleinern und somit eine Evaluation des Spielprinzips zu beschleunigen.

Tabelle 4 Ausbaustufen vorläufige Planung

Ausbaustufe	Umfang
1 (Kernsystem)	• FR1 • FR2 • FR3 • Architektur des Kernsystems
2	• FR4 • FR5
3	• Übrige location-based Features (inklusive Player-Created-Content) • Kämpfe zwischen Doodlings (KI gegen Spieler) • Events • NFR6
4	• Trainieren der Doodlings • Doodlings als Tamagotchi • Einlesen der Bewegungsdaten aus externen Fitnessapps

5 Erstes Inkrement

In diesem Kapitel wird die Entwicklung und Auswertung der ersten Ausbaustufe von Doodlings beschrieben. In der inkrementellen Softwareentwicklung muss im ersten Inkrement sichergestellt sein, dass ein zentraler und funktional einsetzbarer Ausschnitt des Gesamtsystems realisiert wird. Dieser entspricht dem sogenannten Kernsystem [18].
Dieses wird im Folgenden zunächst entworfen, dann entwickelt und zuletzt eingesetzt sowie bewertet.

5.1 Entwurf

In diesem Abschnitt wird das Kernsystem von Doodlings entworfen. Dabei wird zunächst die Architektur herausgestellt und später die einzelnen Komponenten und deren Schnittstellen detailliert beschrieben.

5.1.1 Systemkontext

Zur Übersicht über die Nachbarsysteme und die Stakeholder von Doodlings wird im Folgenden auf den Systemkontext eingegangen. Der Systemkontext wird dabei in „Abbildung 6 Systemkontext“ dargestellt. Zu diesem Zeitpunkt der Entwicklung sind nur zwei externe Systeme geplant, mit denen Doodlings kommunizieren muss. Zum einen ein externer Authentifizierungsdienst für die Authentifizierung der Spieler, zum anderen das Betriebssystem des Endgerätes, dass Funktionalitäten zur Ermittlung der Position und der Himmelsrichtung bieten muss. Die Stakeholder sind die Spieler, welche das Spiel spielen wollen, also in der ersten Ausbaustufe Doodlings finden und sammeln; die Administratoren, welche über den Zustand der beteiligten Geräte informiert werden müssen und diese wenn nötig administrieren, und die Entwickler, welche das System erweitern und warten.

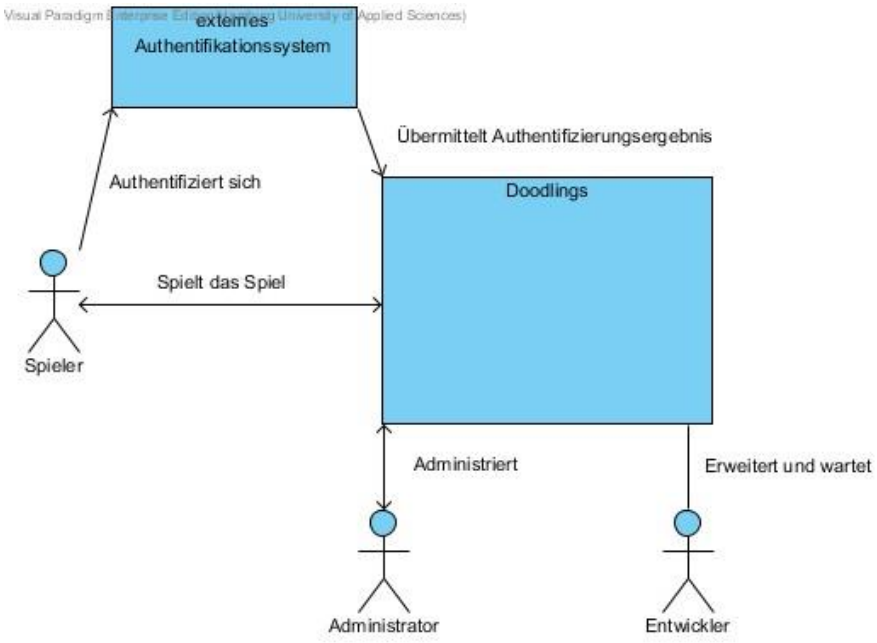

Abbildung 6 Systemkontext

5.1.2 Architektur

Durch die Anforderungen an Doodlings steht fest, dass es sich um ein Multiplayerspiel mit einer konsistenten Welt für alle Spieler handelt. Das heißt, dass der Spielzustand für alle Spieler einheitlich sein muss. Hier bieten sich die in den Grundlagen vorgestellten Architekturmuster „Client-Server" und „Peer-To-Peer" an. Dabei ist die Aufteilung in Client und Server im Fall von Doodlings deutlich vorteilhafter als eine Architektur aus gleichberechtigten Peers. Dies liegt daran, dass bei der „Peer-To-Peer" Architektur immer sichergestellt werden muss, dass der komplette Spielzustand zu jeder Zeit abrufbar und auch über alle Peers konsistent ist. Neben dem erhöhten Rechenaufwand für die Peers wäre auch ein erhöhter Netzwerkaufwand für jeden Spieler nötig. Dies ist jedoch durch die nicht funktionale Anforderung an den Ressourcenverbrauch nicht erwünscht. Außerdem ist ein verteilen des Spielzustandes auf die Geräte der Spieler unerwünscht, damit der Spieler diesen nicht unrechtmäßig verändern kann.

Durch eine „Client-Server" Architektur ist der Spielzustand in einer zentralisierten Infrastruktur gebündelt und kann leichter konsistent gehalten werden. Allerdings bilden die Server in dieser Architektur einen Bottleneck (deutsch: „Flaschenhals"), da sie die gesamte Funktionalität zum Abfragen des Spielzustands für die Clients bereitstellen müssen. Deswegen ist es wichtig, eine Architektur zu entwickeln, die diese Problematik bestmöglich abschwächt. Da es bereits viele ausgearbeitete Architekturen für location-based Games gibt, soll eine von diesen verwendet werden. Dies beschleunigt zum einen den Entwicklungsprozess zum anderen wird das Risiko abgeschwächt, dass die Architektur den

Anforderungen nicht genügt. Als Architektur für Doodlings wird die „Geobashing MMMOG Architektur für location-based Games“ [6] verwendet, da sie sowohl die nicht funktionale Anforderung der Skalierbarkeit (NFR1) erfüllt als auch eine für die Spieler konsistente Sicht auf die positionsbasierten Spielinhalte ermöglicht, die essentiell für die Erfüllung der funktionalen Anforderungen (FR1, FR2, FR5) sind.

5.1.2.1 Geobashing Architektur

Die „Geobashing MMMOG Architektur für location-based Games“ beschreibt eine serverseitige Architektur von location-based Multiplayerspielen. Sie verwendet einen ähnlichen Ansatz wie das in den Grundlagen beschriebene Architekturmuster der „Clustered Server Architektur“ und setzt besonderes Augenmerk auf die schnelle Verarbeitung von Positionsdaten. Die Architektur besteht aus zwei Layern (Webserver Layer und Applicationserver Layer (siehe Abbildung 7)) [6].

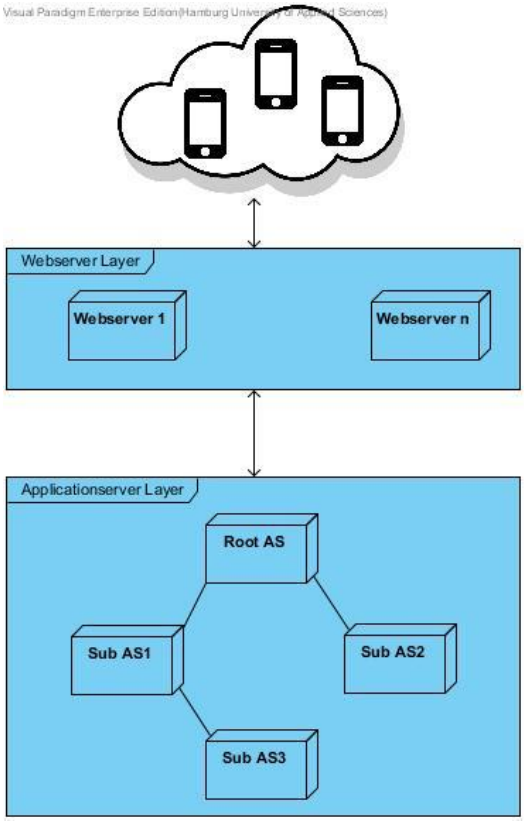

Abbildung 7 Geobashing Architektur

Der Webserver Layer besteht aus gleichförmigen Webservern die über load balancing Mechanismen (z.B. DNS-basiertes load balancing) von den Clients angesprochen werden können. Die Nutzung von Standardtechnologien wie den Webservern vereinfacht und beschleunigt die Anbindung von Clients, da diese Technologien bereits in vielen Frameworks unterstützt werden. Die Webserver merken sich für jeden Nutzer den zugewiesenen Applikationsserver, sodass sie Anfragen direkt weiterleiten können [6].

Die Applikationsserver des Applicationserver Layer implementieren die Spiellogik und halten den Spielzustand. Diese sind in einer Baumhierarchie angeordnet. Dabei ist jeder Applikationsserver für ein spezielles geographisches Gebiet zuständig. Das Gebiet jedes Applikationsservers muss komplett innerhalb des übergeordneten Applikationsservers liegen und darf sich nicht mit den Gebieten anderer Geschwister-Applikationsserver überschneiden. Jeder Applikationsserver kennt seine Kind-Applikationsserver. Diese Anordnung der Applikationsserver verbessert die Skalierbarkeit des Gesamtsystems, da bei starker Belastung eines Applikationsservers dieser durch Erzeugung eines Kind-Applikationsservers entlastet werden kann [6].

Die Anfragen der Clients werden von den Webservern direkt an die Applikationsserver weitergeleitet. Ist die Zuordnung eines Clients zu einem Applikationsserver unbekannt, wird die Anfrage an den Root-Applikationsserver gestellt. Die Bearbeitung jeder Anfrage findet in zwei Schritten statt. Zunächst wird überprüft, ob der Applikationsserver für die Anfrage zuständig ist. Ist dies der Fall, wird diese verarbeitet, wenn nicht, wird überprüft, ob die Anfrage in den Bereich der Kind-Applikationsserver fällt. Ist dieser verantwortlich, wird die Anfrage weitergeleitet. Sind auch diese nicht zuständig, wird eine Nachricht zurück an die Webserver geschickt, die eine neue Zuständigkeitssuche über den Root-Applikationsserver anstoßen. Ein besonderer Anfragetyp ist die sogenannte „in-range“ Anfrage. Diese fragt positionsbasierte Inhalte innerhalb eines Radius an. Dieser Radius könnte sich über die Gebiete mehrerer Applikationsserver erstrecken. In diesem Fall ist derjenige Applikationsserver zuständig, in dessen Gebiet das abgefragte Gebiet fällt. Dieser leitet, wenn nötig, dann die Abfrage an die Kind-Applikationsserver weiter und aggregiert die Ergebnisse. Ändert sich der zuständige Applikationsserver für einen Spieler, wird die Zuordnung für diesen Spieler in den Webservern aktualisiert. Zusätzlich zu den positionsbezogenen Anfragen kann jeder Applikationsserver jede nicht positionsbezogene Anfrage beantworten, also Beispielsweise bestimmte Nutzerdaten [6].

5.1.2.2 Anpassungen der Geobashing Architektur

Doodlings soll serverseitig auf Basis der Geobashing Architektur entwickelt werden. Hierzu wird die Architektur etwas verändert, um den Anforderungen von Doodlings besser gerecht zu werden. Aufgrund der Anzeige der positionsbasierten Inhalte (FR1) werden sehr häufig die oben beschriebenen „in-range“-Anfragen an die Applikationsserver gestellt. Deswegen ist es wichtig, dass die Zuständigkeit der Applikationsserver erst wechselt, wenn die mitgesendete Position inklusive des Radius komplett in das Zuständigkeitsgebiet desjenigen Applikationsservers fällt. Dies verhindert häufige neue Suchoperationen ausgehend vom Rootserver, wenn der Applikationsserver nicht den kompletten Radius abdeckt.
Des Weiteren sollen die Kind-Applikationsserver ebenfalls ihre Elternserver kennen, damit dynamische Spielinhalte, also Inhalte, die ihre Position mit der Zeit verändern, ohne Belastung des Root-Applikationsservers übergeben werden können.

5.1.2.3 Client Architektur

Da die Geobashing Architektur nur den serverseitigen Teil der Architektur eines location-based Games berücksichtigt, wird in diesem Abschnitt auf die Architektur des Clients eingegangen. Der Client hat bei Doodlings lediglich die Aufgabe der Präsentation des Spielinhalts und der Aufnahme und Weiterleitung der Nutzerinteraktion an den Server. Für die Präsentation bietet sich eine sogenannte Game Engine an, die von der Grafikschnittstelle des Betriebssystems abstrahiert und komplexere Funktionalität vereinfacht zur Verfügung stellt. Zusätzlich dazu bieten viele Game Engines auch eine abstrahierte Schnittstelle für andere Betriebssystemfunktionalitäten, wie beispielsweise der Positionsfindung, an. Aufgrund dieser Vorteile wird der Client von Doodlings mithilfe einer Game Engine

entwickelt. Für die Abfrage der Spielinhalte wird die vom Server bereitgestellte Webserver Schnittstelle verwendet.

5.1.3 Bausteinsicht

In der Bausteinsicht wird die statische Sicht der Architektur verdeutlicht. Sie zeigt die einzelnen Komponenten und welche Schnittstellen diese bereitstellen oder nutzen [3]. Hierbei wird zwischen fachlichen und technischen Komponenten unterschieden. Die Aufgabe der fachlichen Komponenten ist die Erfüllung der funktionalen Anforderungen. Technische Komponenten stehen zwischen fachlichen Komponenten und stellen unterstützende Dienste, wie zum Beispiel für die Kommunikation, zur Verfügung. Sie erweitern den fachlichen Umfang der Funktionalität des Systems nicht. In Abbildung 8 ist die Bausteinsicht in einem Komponentendiagramm dargestellt.
Das System ist, wie von der Architektur vorgesehen, in drei Unterkomponenten aufgeteilt: Client, Webserver und Applikationsserver.

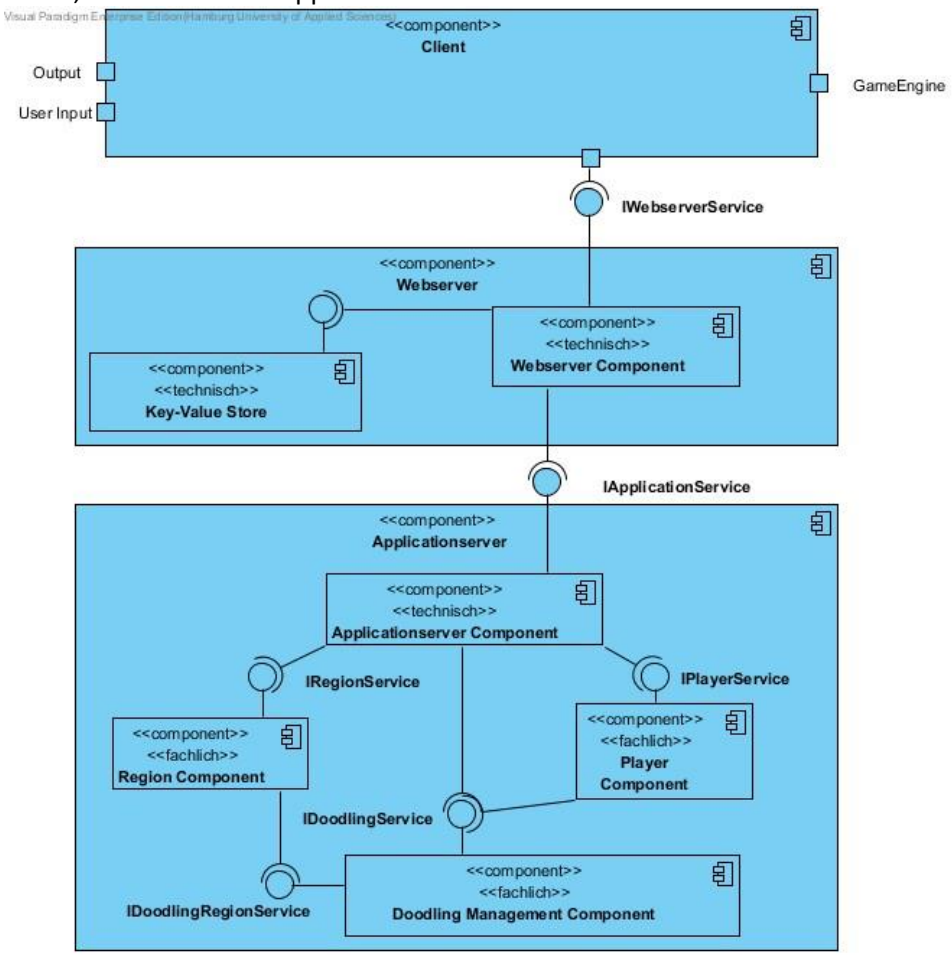

Abbildung 8 Bausteinsicht

Diese wiederum sind in Unterkomponenten aufgeteilt. Auf diese wird im weiteren Verlauf des Kapitels eingegangen. Nur die Applikationsserver stellen Fachlichkeit zur Verfügung.
Dies stellt sicher, dass der Spielzustand gekapselt bleibt und von Spielern nicht unrechtmäßig verändert werden kann (sogenanntes Cheating).
Die Kommunikation zwischen Client und Applikationsserver findet ausschließlich über die Schnittstelle des Webservers statt, dieser wiederum hat Zugriff auf die Schnittstelle des Applikationsservers.
Die Schnittstellen stellen nur Zugriff auf Daten Transfer Objekte (DTO) und nicht auf die Entitäten zur Verfügung. Diese DTO kapseln den aktuellen Zustand einer Entität ohne die Funktionalität dieser bereitzustellen. Somit wird sichergestellt, dass die Entitäten nur von denjenigen Komponenten verändert werden können, die auch für sie verantwortlich sind.

5.1.4 Komponenten

Im folgenden Abschnitt werden die Komponenten des Systems detailliert entworfen. Zunächst werden die fachlichen, danach die technischen Komponenten beschrieben. Dabei werden alle Komponenten in die Schichten Zugriff, Logik und Daten unterteilt. Die Trennung der Schichten vereinfacht die Erweiterung, da der Zugriff auf andere Komponenten nur über die Zugriffsschicht erfolgt und von der restlichen Implementierung der Komponente unabhängig ist. Dies wird aufgrund der nicht funktionalen Anforderung an die Erweiterbarkeit (NFR5) durchgeführt. Zusätzlich werden durch diese Architektur auch die Logik und die Daten voneinander entkoppelt und das Auffinden der Zuständigkeit einzelner Programmteile verbessert. Weitere Einzelheiten zu der Architektur sind in Quasar 3.0 [7] ausgearbeitet.

5.1.4.1 Region

Die Regionkomponente hält den Zustand des Gebietes, welches von dem Applikationsserver verwaltet wird. Hierzu überwacht es alle positionsbezogenen Inhalte innerhalb dieses Gebietes. Es stellt eine Schnittstelle für Anfragen an diese Inhalte zur Verfügung. Außerdem ist es für die Neuerstellung von Doodlings und anderen positionsbezogenen Spielinhalten und das bewegen der dynamischen Inhalte verantwortlich. Ihre Funktionalität muss die Anforderungen FR1, FR2, FR5 und FR6 abdecken.
Die Regionkomponente verwendet die Schnittstelle der Doodling-Verwaltungskomponente, um Spielinhalte, wie zum Beispiel gefangene Doodlings, an den Spieler zu binden und um neue Doodlings zu erstellen.

5.1.4.1.1. Innensicht

Die Innensicht der Komponente ist in Abbildung 9 dargestellt. Der RegionService ist dabei die Fassade der Komponente. Alle Anfragen anderer Komponenten müssen an den RegionService gesendet werden.

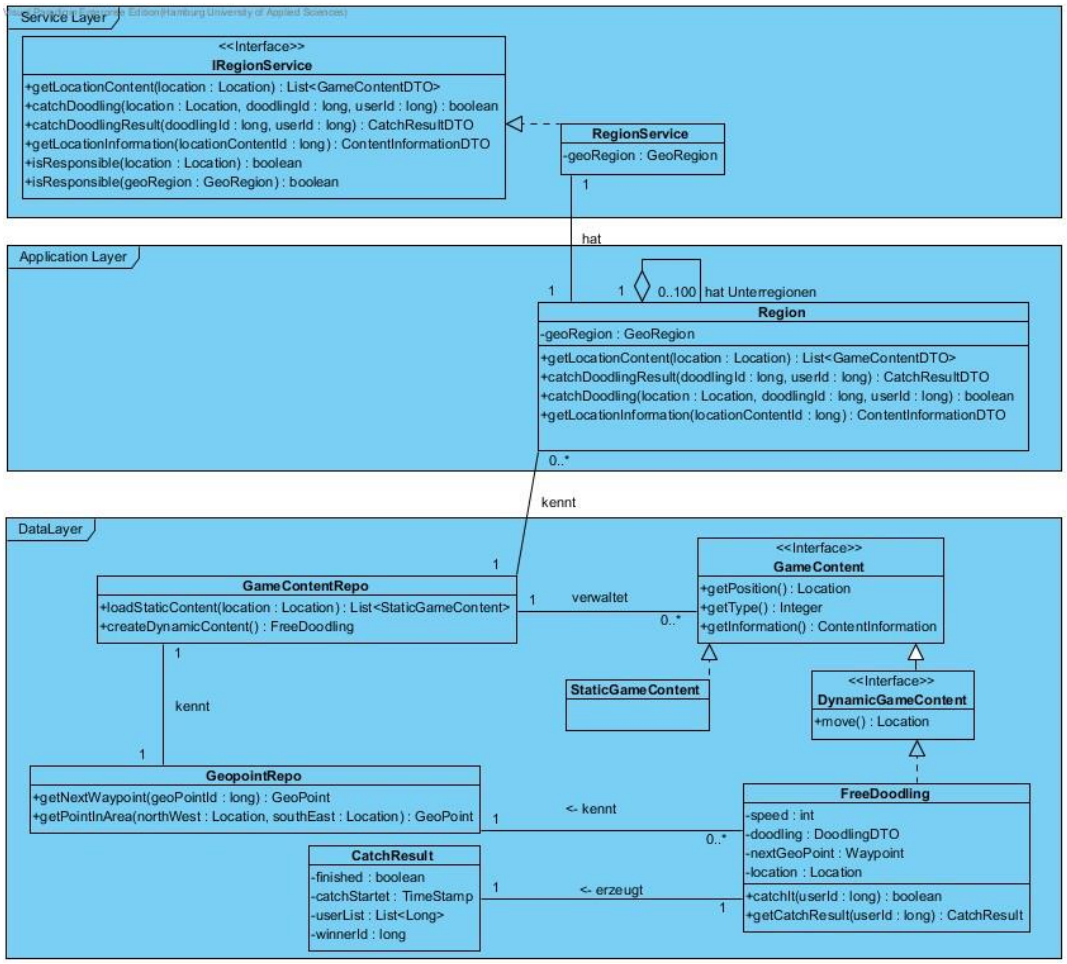

Abbildung 9 Innensicht Region Komponente

5.1.4.1.2. Anfragen an den Spielzustand

Durch die Schnittstellenmethode „getLocationContent“ werden Anfragen an den Spielzustand gestellt. Zurückgegeben werden alle positionsbezogenen Spielinhalte in einem bestimmten Umkreis (Wert sollte konfigurierbar sein). Für die Suche der positionsbezogenen Spielinhalte in der Umgebung einer Position wird ein Suchbaum mit drei Ebenen aufgebaut. Hierzu wird das Gebiet des Applikationsservers anhand der Breiten- und Längengrade seiner „Bounding Box“ in kleinere Teilgebiete aufgeteilt. Die „Bounding Box“ ist dabei die Fläche, die von der Koordinate mit dem nördlichsten und westlichsten Längen- bzw. Breitengrad und der Koordinate mit dem südlichsten und östlichsten Längen- bzw. Breitengrad aufgespannt wird. Die Teilgebiete der ersten Ebene haben immer genau 1° Kantenlänge, also beispielsweise 53°Nord 9°Ost bis 52°Nord 10°Ost. Die weiteren Teilgebiete sind jeweils um den Faktor 10 kleiner, also auf Ebene Zwei 0,1° Kantenlänge. Für Anfragen an den Spielzustand wird dann innerhalb des Suchbaums das Teilgebiet gesucht, welches die Anfrage bearbeiten kann. Dabei werden die Regionen „lazy“ erzeugt um nicht unnötigen Speicher zu belegen. Die Erzeugung einer Unterregion findet zum Zeitpunkt der ersten Abfrage statt. Bei der Erzeugung der Unterregionen werden die statischen positionsbezogen Spielinhalte für diese Region über das GameContentRepo geladen, so dass keine Datenbankabfrage bei Anfragen an den Spielzustand nötig sind. Wird der Spielzustand durch eine Anfrage verändert, wird dies persistent in der Datenbank gespeichert.

Abbildung 10 Ebenensuchbaum Hamburg Jungfernstieg, Kartendaten © 2015 GeoBasis-DE/BKG (©2009), Google

Zur Verdeutlichung kann der Suchbaum für eine Anfrage am Hamburger Jungfernstieg in Abbildung 10 nachvollzogen werden. Hierbei ist die Ebene 1 rot, die Ebene 2 grün, und die Ebene 3 blau gekennzeichnet. Wird vom System erkannt, dass zu wenig automatisiert generierte Inhalte vorhanden sind, werden diese erzeugt (siehe FR1).

5.1.4.1.3. Dynamische positionsbezogene Inhalte

Die dynamischen positionsbezogenen Inhalte benötigen eine besondere Verarbeitung, da sich die Positionen dieser Inhalte verändern sollen (Systeminterne Anforderung 1). In der

ersten Ausbaustufe fallen nur freie Doodlings, also solche, die von Spielern gefangen werden können, in diese Kategorie. Die Bewegung der freien Doodlings wird nur berechnet, wenn diese auch angefragt wird (siehe oben). Sobald eine Anfrage stattfindet, wird zunächst die neue Position berechnet und diese dann zurückgegeben. Die Berechnung der Position findet anhand der Geschwindigkeit (durch das Attribut „speed“ symbolisiert) und des nächsten Wegpunktes („nextGeoPoint“) statt. Dabei wird die Geschwindigkeit in Metern pro Minute angegeben. Der genaue Ablauf der Bewegung ist im nachfolgenden Aktivitätsdiagramm beschrieben (siehe Abbildung 11). Für die Wegpunkte wird ein Straßengraph verwendet. Dies hat zum Vorteil, dass die Doodlings sich auf jeden Fall auf erreichbarem Terrain bewegen. Der Straßengraph wird durch die von OpenStreetMap[6] bereitgestellten Kartendaten erzeugt. Die Kartendaten bestehen aus drei Elementen: Nodes, Ways und Relations. Für die Ermittlung des Straßengraphen für Doodlings werden nur Nodes und Ways benötigt. Ways werden durch eine Aneinanderreihung von Nodes beschrieben, welche, wenn die jeweils benachbarten Nodes durch eine Gerade verbunden werden, die symbolisierte Straße ergeben. Aus dem Ursprungsdatensatz werden diejenigen Straßen (Ways) entfernt, die per se unsicher für die Spieler sein könnten, beispielsweise Autobahnen, Bundes-, Hauptverkehrs-, und Überlandstraßen. Das Ergebnis ist ein nicht zusammenhängender Graph, bei dem die Ecken die Wegpunkte im Straßenverkehrsnetz markieren und die Kanten diese Verbinden. . Bei der Ermittlung eines neuen Wegpunktes für ein freies Doodling müssen also nur die verbundenen Wegpunkte des alten Wegpunktes abgefragt und davon ein Zufälliger ausgewählt werden. Dabei ist wichtig, dass der Wegpunkt, den das Doodling vor dem aktuellen Wegpunkt besucht hat, berücksichtigt wird und nur dann ausgewählt wird, wenn kein Anderer zur Verfügung steht. Für die Abfrage neuer Wegpunkte ist das GeopointRepo verantwortlich.

Nach dem Bewegen des Doodlings muss von der Region, in der sich das Doodling vor der Bewegung befand, überprüft werden, ob es sich immer noch in dieser befindet oder ob sich die Zuständigkeit verändert hat. Ist dies der Fall, wird das in der Rückgabe der Anfrage vermerkt und die im Suchbaum jeweils höherliegenden Regionen müssen überprüfen, ob sie noch für dieses Doodling verantwortlich sind und gegebenenfalls dieses an entsprechende Unterregionen übergeben. Falls sich das Doodling durch seine Bewegung aus dem Bereich des aktuellen Applikationsservers bewegt hat, muss es über den Root-Applikationsserver dem dann zuständigen Server übergeben werden.

[6] http://www.openstreetmap.org/

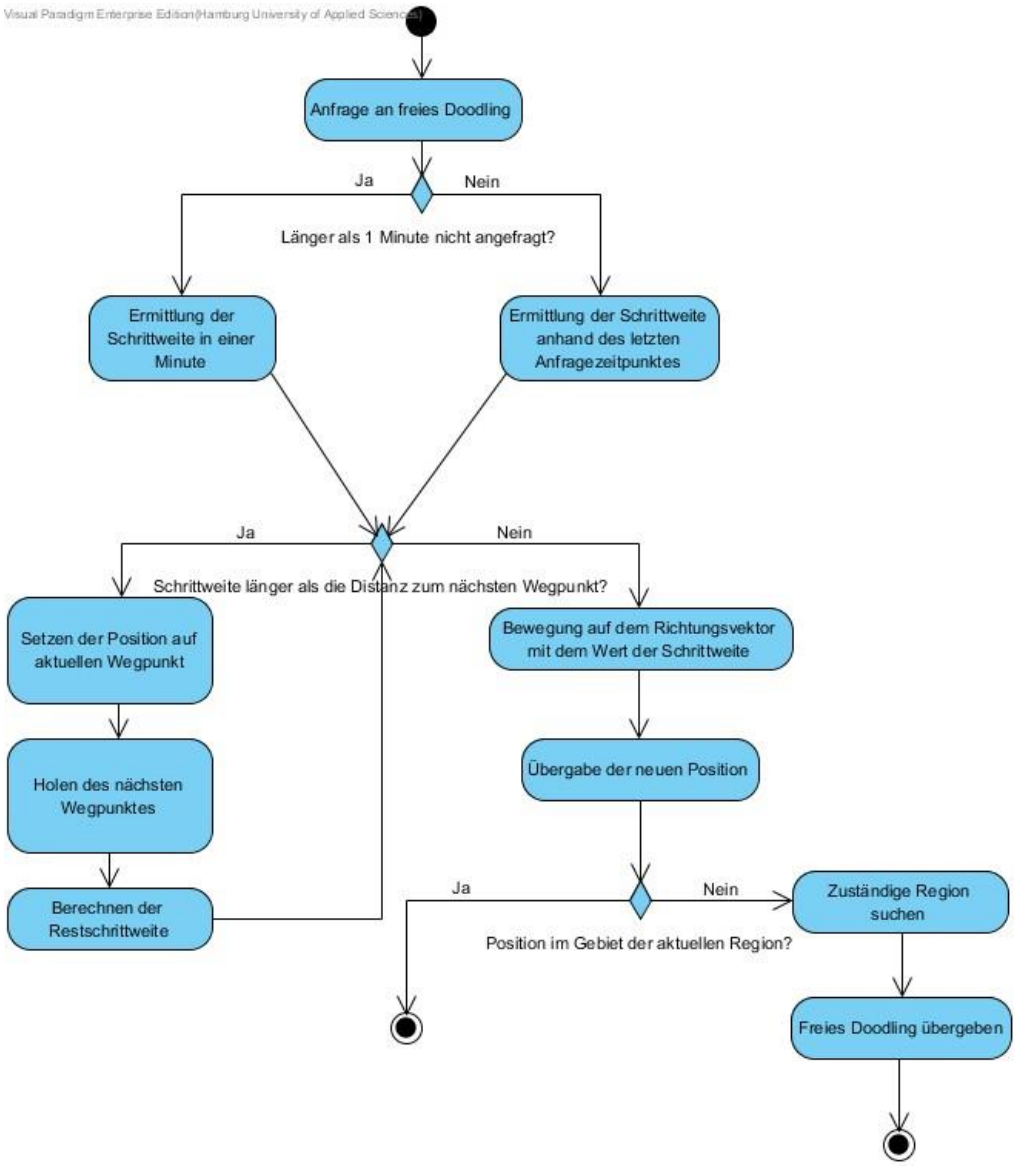

Abbildung 11 Bewegung eines freien Doodlings

5.1.4.1.4. Fangen eines Doodlings

Die Schnittstellenoperationen für das Fangen eines Doodlings sind „catchDoodling“ und „catchDoodlingResult“. Beim Aufruf von catchDoodling wird zunächst überprüft, ob die übergebene Lokation im unmittelbaren Radius der Lokation des Doodlings ist. Der Radius ist konfigurierbar. Danach wird beim ersten Aufruf der „catchDoodling“-Methode ein Timer gestartet. Solange der Timer läuft werden alle weiteren Aufrufer dieser Funktion gespeichert; die Zeitspanne ist dabei ebenfalls konfigurierbar. Nach dem Ende der Zeitspanne wird das Doodling entweder direkt zugeordnet – wenn nur ein Spieler die catchDoodling Methode

aufgerufen hat – oder es findet ein Turnier um das Doodling statt. Da für ein Turnier das Kampfsystem implementiert sein muss, dieses aber nicht in der ersten Ausbaustufe implementiert wird, wird es zunächst immer dem ersten Aufrufer zugeordnet. Über die Methode „catchDoodlingResult“ kann der aktuelle Status des Fangvorgangs abgefragt werden. Nachdem die Zuordnung des Doodlings zu einem Aufrufer stattgefunden hat, wird das freie Doodling aus dem Spielzustand entfernt.

5.1.4.2 Player

Die Spielerkomponente ist für alle Dienste verantwortlich, die den Spieler betreffen. In dieser Ausbaustufe ist das nur die Zuordnung des Spielernamens zu einer Identifikation. Hierfür stellt sie eine Schnittstelle zum Abfragen (IPlayerService) zur Verfügung. Für die Abfrage der Doodlings zu einem Spieler wird die Schnittstelle der Doodling-Verwaltungskomponente verwendet.

5.1.4.2.1. Innensicht

Die Innensicht der Spielerkomponente ist in Abbildung 12 grafisch dargestellt. Es handelt sich hierbei um eine sehr einfache Komponente, da in der ersten Ausbaustufe noch sehr wenig spielerspezifische Funktionalität angeboten wird. Sie besteht lediglich aus einem Service und einem Repository, welches die persistent gespeicherten Spieler verwaltet, sowie einer Player Entität, die den Namen eines Spielers mit einer Identifikation verbindet.

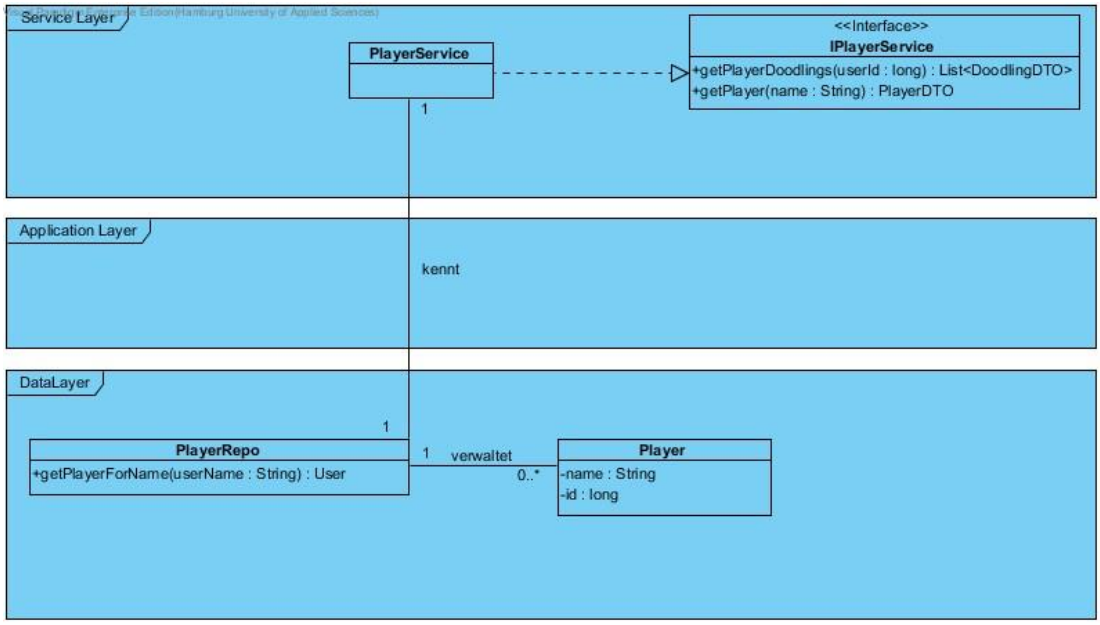

Abbildung 12 Innensicht Spielerkomponente

5.1.4.3 Doodling-Verwaltung

Die Doodling-Verwaltung Komponente verwaltet die Doodling Entitäten. Sie ist verantwortlich für die Erzeugung neuer Doodlings, die Verwaltung der persistierten Doodlings und der Zuordnung der Doodlings zu den Spielern. Die Anforderungen an diese Komponente sind durch FR3 und FR2 definiert.

5.1.4.3.1. Innensicht

Die Komponente ist ähnlich aufgebaut wie die Spielerkomponente, allerdings ist der Entitätstyp Doodling wesentlich komplexer als die Entität Player. Die Innensicht ist in Abbildung 13 dargestellt. Da die Doodling Entität in dieser Ausbaustufe nur angezeigt werden soll, bietet sie keinerlei Funktionalität. Wie aus der Spielidee-Beschreibung hervorgeht hat jedes Doodling einen Typ, dieser wird durch die Klasse DoodlingType beschrieben. Jedes Doodling dieses Typs hat also bestimmte Standardwerte. Jeder DoodlingType wiederum hat ein Element, dem er zugeordnet ist. Dieses entscheidet, gegen welche Elemente es Vor- oder Nachteile hat. Die individuellen Fähigkeiten der Doodlings werden durch die DoodlingAbility Klasse symbolisiert. Doodling und DoodlingAbility haben jeweils ein Level, auf dem sie sich befinden und anhand dessen sich die Werte verändern. Ein Doodling hat immer genau einen Spieler, dem es zugeordnet ist. Dabei ist das oben erwähnte freie Doodling eine Ausnahme, dieses wird einem Spieler zugeordnet, nachdem es gefangen wurde. Doodling und Spieler werden lose über die Identifikationsnummer des Spielers gekoppelt, da sie von unterschiedlichen Komponenten verwaltet werden.

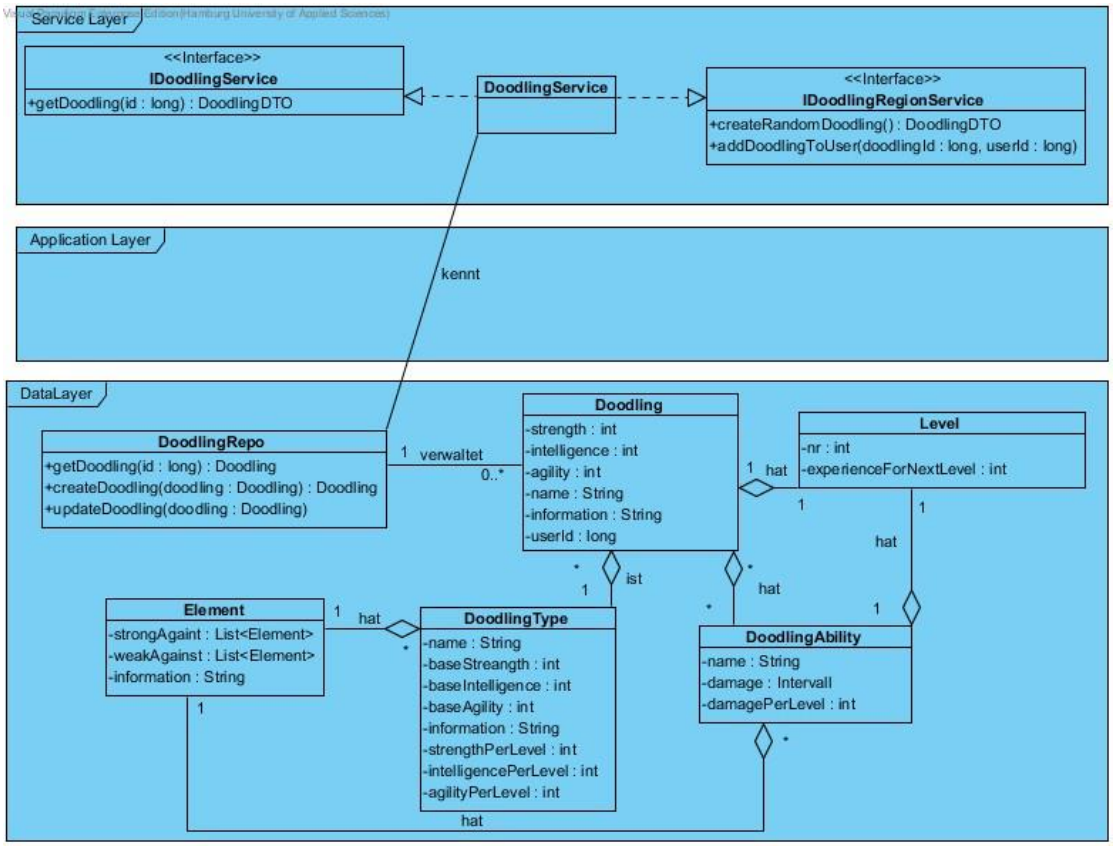

Abbildung 13 Innensicht Doodling-Verwaltung

5.1.4.4 Applikationsserver

Die Applikationsserverkomponente stellt die für die Webserver nutzbare Schnittstelle bereit. Die Kommunikation findet dabei über Nachrichten statt, da der Webserver und der Applikationsserver verteilt lauffähig sein sollen. Für die Kommunikation über Nachrichten bietet sich das Aktor-Modell an. Ein Aktor kapselt dabei den Zustand eines Programmabschnitts, in diesem Fall des Applikationsservers. Die Veränderung dieses Zustands von außen kann nur über Nachrichten an den Aktor verändert werden. Zusätzlich erleichtert das Aktorenmodell die Verteilung des Systems, da lokale und entfernte Kommunikation gleichartig abläuft. Dabei ist ein Aktor ein leichtgewichtiger Thread, sodass sehr viele Akteure von einer einzelnen Maschine verwaltet werden können (bis zu 2.5 Millionen pro GB Arbeitsspeicher mit dem Akka Framework [27]). Für die Entwicklung der Applikationsserver wird das Akka Framework[7] verwendet. Dabei besteht die Applikationsserverkomponente nur aus einem Aktor, der die Aufrufe an die zuständigen

[7] http://akka.io

fachlichen Komponenten (Doodling-, Spieler- und Regionkomponente) delegiert. Weitere Informationen zum Aktorenmodell findet man zum Beispiel in [2].
Neben den fachlichen Schnittstellen werden vom Applikationsserver ebenfalls die Kind-Applikationsserver und die Verbindung zum Eltern-Applikationsserver verwaltet. Diese benötigt der Applikationsserver für die Suche nach einem geeigneten Applikationsserver für einen Spieler und für die Übergabe von dynamischen Spielinhalten an andere Applikationsserver. Außerdem ist der Applikationsserver für die Rücksendung der Antwortnachrichten zuständig. In Abbildung 14 ist der Kontext der Applikationsserverkomponente dargestellt.

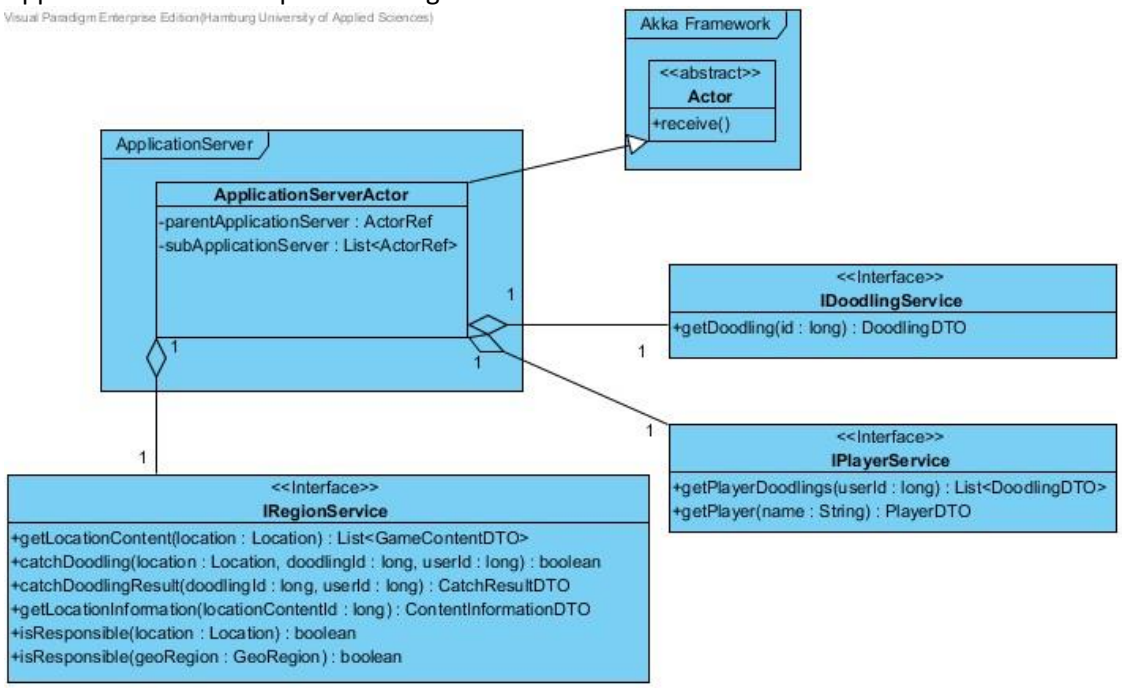

Abbildung 14 Kontext Applikationsserver

5.1.4.4.1. Schnittstelle

Für die Anfrage und Rückgabe an den Applikationsserver werden jeweils serialisierbare Anfrage- und Rückgabeklassen erzeugt, die vom ApplicationServerActor erkannt, verarbeitet und entsprechend beantwortet werden. Dabei stellt der ApplicationServerActor für alle öffentlichen Schnittstellenmethoden der fachlichen Komponenten entsprechende Klassen bereit. Die Parameter werden ebenfalls in den Anfrageklassen übergeben. Es ist wichtig, dass alle Parameter und Rückgabewerte der Services serialisierbar sind, damit sie übertragen werden können. Bei Ankunft jeder Anfragenachricht überprüft der Applikationsserver zunächst, ob er für diese Anfrage zuständig ist, wie in der Architektur beschrieben.

5.1.4.5 Webserver

Die Webserverkomponente stellt den für den Client ansprechbaren Teil der Schnittstelle bereit. Er fungiert als sogenannter Broker, das heißt, er bietet die Funktionalität des Gesamtsystems gekapselt an, sodass die Clients über den Verteilungsgrad des Servers keine Informationen haben müssen. Zusätzlich kann durch den Broker die Server-Architektur verändert werden, ohne dass die Clients angepasst werden müssen. Der Nachteil des Broker Entwurfsmusters ist, dass der Broker zu einem „Single-Point-Of-Failure“ wird. Dies bedeutet, dass ein Ausfall des Brokers das Gesamtsystem ausfallen lassen würde [4]. In der Geobashing-Architektur wird dies verhindert, indem es mehrere Webserver gibt, die von den Clients angesprochen werden können.

Die Webserverkomponente baut wie die Applikationsserverkomponente auf dem Aktormodell auf. Da sich vom Webserver für jeden Client der zuständige Applikationsserver gemerkt werden muss, wird für jeden Benutzer ein Aktor auf dem Webserver erzeugt. Dieser erhält die Anfragen des Clients, für den er zuständig ist und leitet diese an den zuständigen Applikationsserver weiter. Außerdem erhält er die Antwort des Applikationsservers, wandelt diese in das entsprechende Rückgabeformat um und leitet sie weiter an den Client.

Zusätzlich zu den Abfragen an des dynamischen Contents des Applikationsservers bietet der Webserver auch statischen Content an. In dieser Ausbaustufe von Doodlings betrifft das nur die Bilder der Doodlings, damit Doodlings zum Spiel hinzugefügt werden können ohne die Clients aktualisieren zu müssen.

Für die Erstellung des Webservers wird die Bibliothek Spray[8] verwendet. Sie basiert auf dem Akka Framework und bietet Funktionalitäten zum Erzeugen eines Webservers, welcher das Aktorenmodell und „RESTful web services“ unterstützt. Durch die Unterstützung des Aktorenmodells wird es einfacher, die Nebenläufigkeit der Clientanfragen sicherzustellen. Außerdem wird die Kommunikation mit dem Applikationsserver, der auch auf dem Aktorenmodell basiert, vereinfacht. „RESTful web services“ bieten eine ressourcenorientierte Schnittstelle, die mit HTTP-Anfragen angesprochen werden kann. Dabei wird der URI (Uniform Resource Identifier) als Definition des Anwendungsbereichs verwendet, also zum Beispiel der Methode, die aufgerufen werden soll [23]. HTTP hat sich in den letzten Jahrzehnten als Standardtechnologie durchgesetzt. Dies hat zum Vorteil, dass in nahezu jeder Programmiersprache oder jedem Framework – beispielsweise Game Engines – entsprechende Konstrukte zum Erzeugen von HTTP-Anfragen bereits vorhanden sind. Dies beschleunigt die Entwicklung sowohl des Clients als auch des Webservers [6]. Die Kontextsicht der Webserverkomponente ist in Abbildung 15 dargestellt.

Außerdem muss für das Testen der Architektur und die Bereitstellung für den Test durch mehrere Benutzer ein rudimentärer Authentifikationsmechanismus implementiert werden. Da dieser nicht für den Produktivbetrieb verwendet werden soll, kann dabei zunächst auf Sicherheitsmechanismen verzichtet werden.

[8] http://www.spray.io

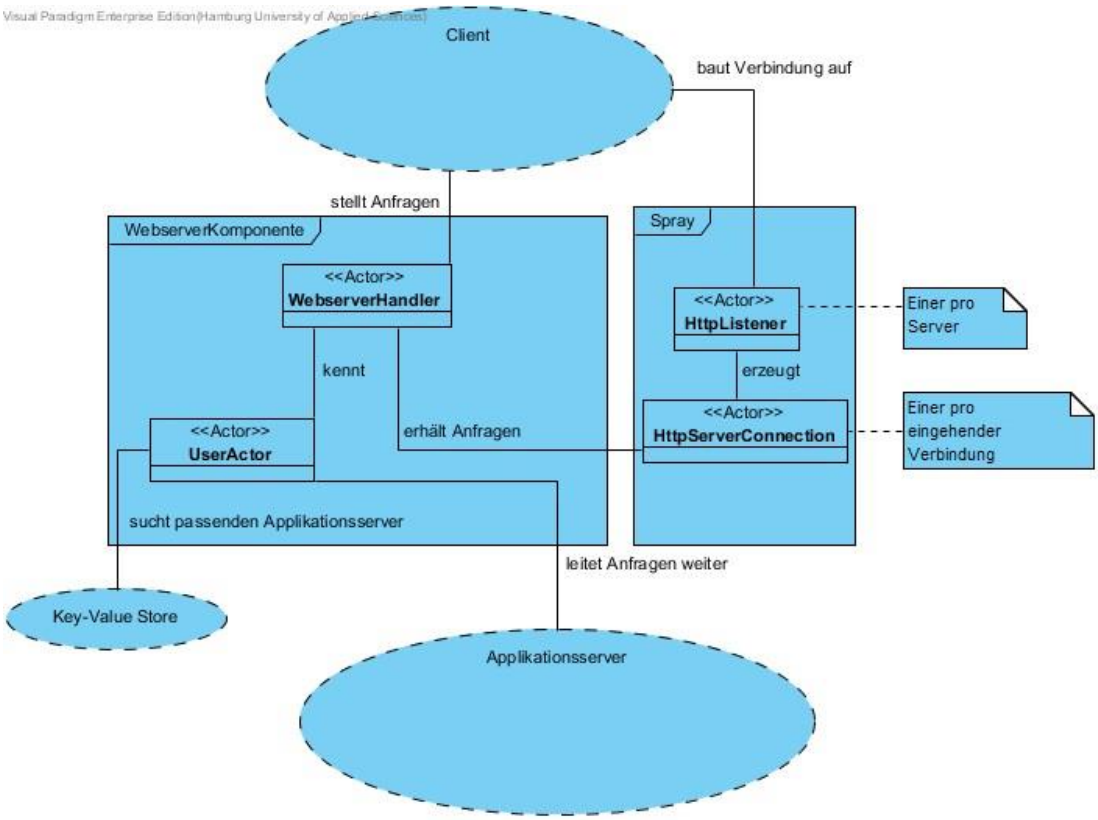

Abbildung 15 Kontext Webserver

5.1.4.5.1. Schnittstelle

Die REST-Schnittstelle des Webservers ist wie folgt definiert:

Anfrage der Positionsbasierten Spielinhalte
URI: /locationContent
Methode: GET
Authentifiziert: Ja
Parameter:

- lat – Breitengrad der aktuellen Spielerposition
- long – Längengrad der aktuellen Spielerposition

Anfrage der Detailinformationen zu einem bestimmten Spielinhalt:
URI: /locationContentInformation
Methode: GET
Authentifiziert: Ja
Parameter:

- lat – Breitengrad der aktuellen Spielerposition
- long – Längengrad der aktuellen Spielerposition
- typeid – Typidentifikation des Spielinhalts
- id – Identifikation des Spielinhalts

Anfrage der Doodlings eines Spielers
URI: /doodlingsofplayer
Methode: GET
Authentifiziert: Ja
Parameter: -

Anfrage für das Fangen eines Doodlings
URI: /catch
Methode: GET
Authentifiziert: Ja
Parameter:

- lat – Breitengrad der aktuellen Spielerposition
- long – Längengrad der aktuellen Spielerposition
- id – Identifikation des Doodlings

Anfrage des Ergebnisses des Fangens eines Doodlings
URI: /catchResult
Methode: GET
Authentifiziert: Ja
Parameter:

- id – Identifikation des Doodlings

Authentifizierung
URI: /auth
Methode: GET
Authentifiziert: Ja
Parameter: -

Anfrage der Statischen Inhalte (z.B. die Bilder der Doodlings)
URI: /bundles/*
Methode: GET
Authentifiziert: Nein
Parameter: -

5.1.4.6 Game Engine

Der Client wird mithilfe einer Game Engine entwickelt. Es gibt eine Vielzahl von Game Engines für mobile Endgeräte. Die Auswahl der richtigen Game Engine zu Beginn der Entwicklung ist wichtig, da ein Wechsel der Game Engine nach Beginn der Entwicklung erheblichen Aufwand bedeuten würde. Um dieses Risiko abzuschwächen, werden zunächst Game Engines auf ihre Nutzbarkeit für Doodlings überprüft. Dafür wird ein Prototyp entwickelt, der bestimmte Grundlagen überprüft und einen ersten Einblick in die Entwicklung mit der jeweiligen Game Engine bietet. Aufgrund des begrenzten Umfangs dieser Arbeit wurde eine Vorauswahl der Game Engines, die evaluiert werden sollen, getroffen. Es werden die Game Engines Unity[9] und LibGdx[10] überprüft. Unity ist eine weit verbreitete Game Engine, die eine eigene IDE mit Grafikwerkzeugen und ein eigenes Designmuster mitbringt. LibGdx wiederum ist eine Opensource Game Engine, die sich an die für Softwareentwickler bekannten Strukturen hält. Sie bietet keine eigene IDE und keine Grafikwerkzeuge. Die Vorauswahl der Game Engines wurde aufgrund der für Doodlings in jedem Fall benötigten Anforderungen, die erfüllt werden müssen, getroffen. Die Anforderungen sind:

- 2D Kompatibilität
- GPS-Service des Gerätes muss angesprochen werden können
- Keine Kosten für die Entwicklung und Veröffentlichung
- Android muss unterstützt werden

Zusätzlich soll in der Evaluation auch auf optionale aber den Entwicklungsprozess stark vereinfachende oder das Spielerlebnis verbessernde Merkmale eingegangen werden. Diese sind:

- Dem Entwickler bekannte Sprache
- Unterstützung des Kompass
- Ressourcenschonend
- Gut dokumentiert
- Plattformunabhängig (Android, iOS, WindowsPhone)

5.1.4.6.1. Prototyp

Der Prototyp soll innerhalb eines Tages entwickelt werden können. Er soll die grundlegenden Eigenschaften, die für den Client von Doodlings benötigt werden, abdecken. Deswegen wird der Prototyp aus einer Anzeigekomponente bestehen. Diese zeigt die aktuelle Position des

[9] https://unity3d.com
[10] https://libgdx.badlogicgames.com/

Spielers, tätigt eine HTTP-Abfrage an einen Webserver und stellt das Ergebnis grafisch dar. Die HTTP-Abfrage soll als Rückgabe ein JSON Objekt erhalten, welches eine Liste von Geopositionen (Längen- und Breitengraden) enthält. Zusätzlich soll auf der Mitte des Bildschirms ein Pfeil die Himmelsrichtung, in die das Gerät gerade ausgerichtet ist, anzeigen. Des Weiteren soll eine grafische Benutzungsoberfläche entwickelt werden, welche die aktuelle Position des Benutzers anzeigt und eine HTTP-Abfrage starten kann. In Abbildung 16 ist ein Wireframe des Prototypen abgebildet.

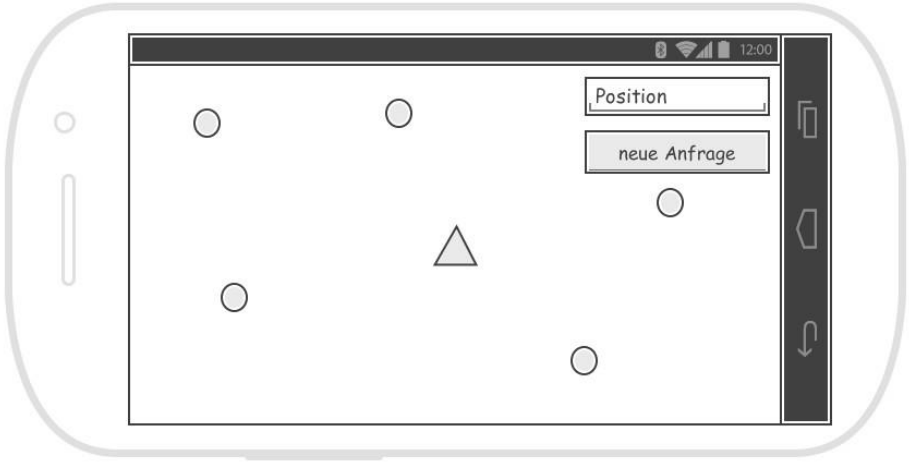

Abbildung 16 Prototyp Mock

5.1.4.6.2. Evaluation

Die folgende Tabelle (siehe Tabelle 5) zeigt die Erkenntnisse, die durch die Entwicklung des Prototypen für die Entscheidungskriterien gewonnen wurden. Sie stellt dabei die beiden betrachteten Game-Engines gegenüber. Für die Bestimmung des Ressourcenverbrauchs wurde der Prototyp eine Stunde auf einem Samsung Galaxy S4 ausgeführt und danach der in den Android Einstellungen angezeigte Akkuverbrauch notiert.

Bei der Entwicklung des Prototyps ist vor allem die sehr ausführliche Dokumentation der Unity Engine positiv aufgefallen. Durch diese und die von der Unity IDE bereitgestellten Entwicklungswerkzeuge war es ohne Vorkenntnisse möglich, die Entwicklung des Prototypen fast in der gleichen Zeit wie in der bereits bekannten LibGDX-Engine abzuschließen. Ein zusätzlicher Vorteil der Unity-Engine ist die GPS-Kompatibilität ohne platformspezifischen Code. Außerdem ist bei der Entwicklung des Userinterfaces des Prototypen aufgefallen, dass die Skalierung der Interfacekomponenten bei LibGDX sehr aufwändig ist. Die Skalierung muss anhand der DPI Implementiert werden, da die Standardskalierung für Smartphone Bildschirme viel zu klein angezeigt wird. Dies wird bei Unity von der Engine übernommen.

Tabelle 5 Prototyp Ergebnis

Merkmal	Unity	LibGDX
Umsetzungsdauer des Prototypen	4 Stunden	3,5 Stunden
2D Kompatibel	Ja	Ja
GPS-Kompatibel	Ja	Ja, allerdings muss direkt auf die Android Implementierung des LocationServices zugegriffen werden. Dies verhindert eine Plattformunabhängigkeit, ohne dass weitere Entwicklungsarbeit notwendig ist.
Kosten	Gratis	Gratis
Android Unterstützung	Ja	Ja
Sprache	C# (bekannt)	Java (bekannt)
Kompass unterstützt	Ja	Ja
Ressourcenverbrauch	7% Akku/Stunde	6% Akku/Stunde
Dokumentation	Sehr ausführlich • (Tiefgehende) Tutorials (Video und Text) • Live Trainings • Beispielprojekte • API • Communityforum	Ausreichend • (Einfache) Tutorials • Beispielprojekte • (unvollständiges) Wiki • API • Communityforum
Plattformunabhängig	Ja. Auslieferung möglich für Android, IOS und WindowsPhone.	Ja. Auslieferung möglich für Android und IOS. Durch die nicht in der Engine verankerte Unterstützung von GPS allerdings zusätzlicher Aufwand nötig.

5.1.4.6.3. Fazit

Unity bietet eine gute Dokumentation und eine Entwicklungsumgebung. Diese beschleunigen den Entwicklungsprozess des Spiels ungemein. Zusätzlich ist die GPS-Kompatibilität plattformunabhängig und die Userinterface Gestaltung deutlich einfacher. Diese Vorteile wiegen den marginalen Unterschied des Akkuverbrauchs und des Wissensvorsprungs bei LibGDX auf. Aus diesem Grund wird für die Entwicklung des Clients von Doodlings die Unity-Engine zum Einsatz kommen.

5.1.4.7 Client

Im Folgenden wird der Client entworfen. Wie im vorherigen Kapitel beschrieben, wird dieser mithilfe der Unity-Engine entwickelt. Ein Spiel der Unity-Engine besteht aus sogenannten Szenen. Diese enthalten Spielobjekte (sogenannte GameObjects) und stehen für einen Abschnitt des Spiels (in klassischen Computerspielen zum Beispiel Level). Die erste Ausbaustufe von Doodlings wird zunächst in drei Szenen unterteilt:

- Kompass
- Überblick
- Login

Innerhalb der Kompassszene werden die positionsbasierten Spielinhalte angezeigt. Sie soll sich optisch an dem Aufbau des Prototypen orientieren (siehe Abbildung 16) und wie ein Kompass aufgebaut sein. Zusätzlich soll eine Navigation zur Überblickszene möglich sein. In der Mitte des Bildschirms soll eine Kompassnadel angezeigt werden, welche in die Richtung zeigt, in die sich der Spieler aktuell bewegt. Durch das Tippen auf einen der positionsbasierten Spielinhalte soll eine Detailansicht für diesen geladen werden. Diese soll ein Bild, die Entfernung, den Namen, eine Beschreibung sowie die Attribute Stärke, Intelligenz und Agilität des Spielinhalts anzeigen. Zusätzlich wird innerhalb dieser Detailansicht die Schaltfläche zum Fangen der Doodlings angezeigt.
Die Überblicksszene ist für die Anzeige der Doodlings, die der Spieler bereits gefangen hat (siehe Abbildung 17). Dabei wird ein Slider für die Navigation durch die Liste der Doodlings verwendet. Auch hier werden die Detailinformationen zu den Doodlings angezeigt. Diese sind: Der Name, ein Bild, ein Bild für das Element, die Attribute sowie das Level, auf dem sich das Doodling aktuell befindet (initial immer 1). Zusätzlich kann über eine Schaltfläche die Kompassszene geladen werden. Die Bilder und Informationen der Doodlings werden immer dynamisch nachgeladen, sodass bei Veränderung eines Bildes oder dem Hinzufügen neuer Doodlings der Client nicht verändert werden muss. Dies wird in Unity über Asset-Bundles realisiert. Unity stellt eine Funktion zum Laden und Cachen von Asset-Bundles über HTTP-Requests zur Verfügung, sodass unveränderte Bilder nicht immer wieder runtergeladen werden müssen.
Die Loginszene fungiert als Platzhalter und wird nur für den Test benötigt. Sie besteht aus einem Textfeld für den Benutzernamen, einem Textfeld für das Passwort sowie einer Schaltfläche für die Durchführung der Authentifikation.
Beim Starten des Spiels wird zunächst die Loginszene angezeigt. Nach einer erfolgreichen Authentifikation wird der Spieler dann auf die Überblicksszene weitergeleitet.

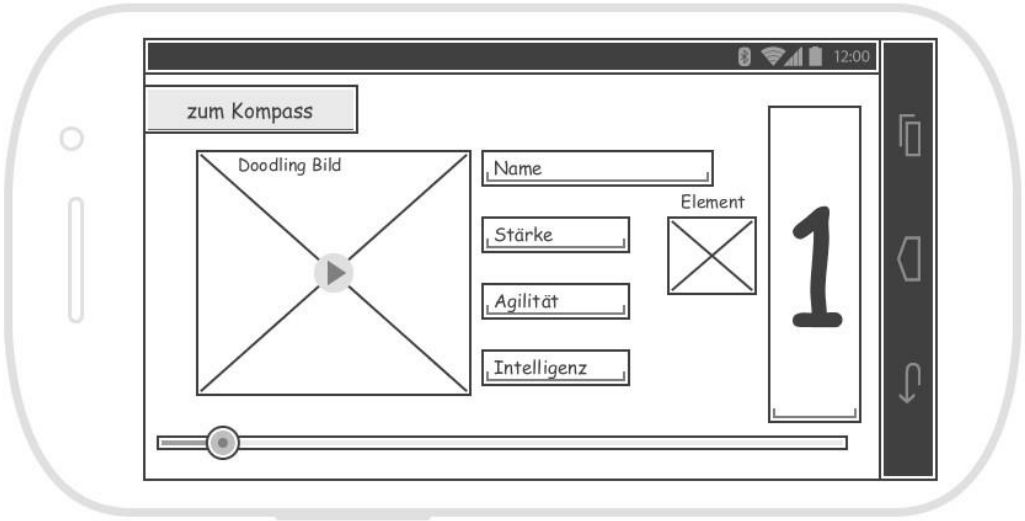

Abbildung 17 Übersichtsszene

5.1.5 Laufzeitsicht

Im folgenden Abschnitt wird die Kommunikation zwischen den einzelnen Komponenten zur Laufzeit entworfen. Da alle Anfragen vom Client an den Webserver, von diesem an den Applikationsserver und die entsprechende Komponente identisch ablaufen, wird nur die Laufzeitsicht für die Abfrage der positionsbasierten Spielinhalte beschrieben. Ein Sequenzdiagramm zur Verdeutlichung des Ablaufs ist in Abbildung 18 dargestellt. Wie in den Anwendungsfällen beschrieben, werden die Abfragen an das Programm vom Spieler angestoßen; im beschriebenen Fall implizit durch Navigation auf die Kompassszene. Diese wiederum stellt eine http-Anfrage an den Webserver. Dieser leitet diese Anfrage weiter an den Applikationsserver, wie in der Architektur beschrieben, entweder dem Root-Server oder dem den Benutzer zugewiesenen Applikationsserver.

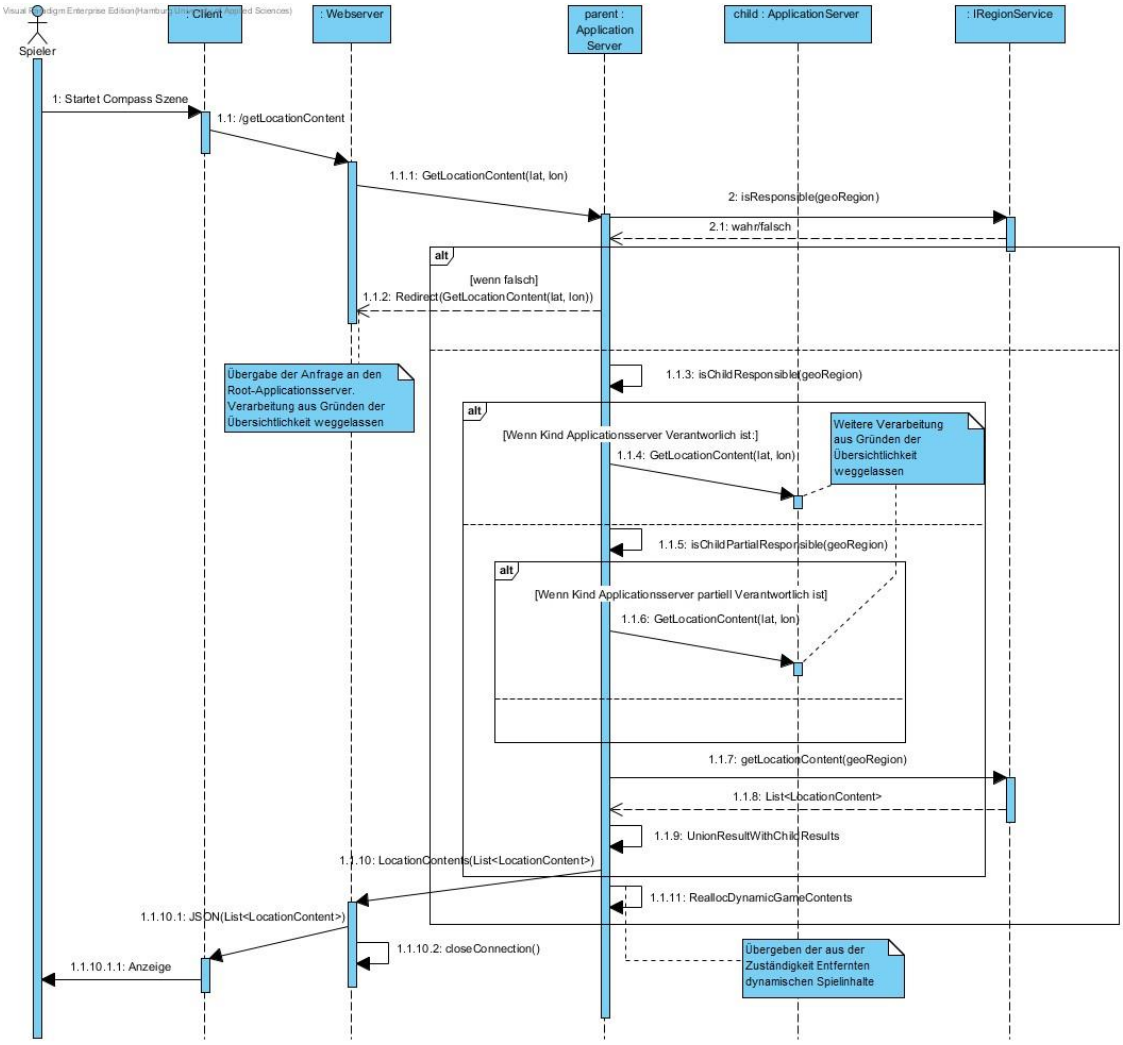

Abbildung 18 Laufzeitsicht

Der Applikationsserver überprüft zunächst die Zuständigkeit für die Anfrage. Dabei gibt es drei verschiedene Möglichkeiten der Zuständigkeit:
Der Applikationsserver ist

- nicht zuständig
- nicht zuständig, aber ein Kind-Applikationsserver ist zuständig
- teilweise oder vollständig zuständig

Im ersten Fall teilt er dies durch eine Redirect-Nachricht dem Webserver mit. Dieser startet dann eine neue Anfrage, allerdings an den Root-Applikationsserver. Dieser ist immer zuständig. Im zweiten Fall wird die Anfrage-Nachricht an den entsprechenden Kind-Applikationsserver weitergeleitet. Dieser kommuniziert die Antwort dann direkt an den Webserver. Im dritten Fall wird das Ergebnis aus den eigenen Komponenten ermittelt und mit dem Ergebnis von eventuell teilzuständigen Kind-Applikationsservern vereinigt. Bei Anfragen an den Applikationsserver, die keinen positionsabhängigen Kontext haben, wird die Ermittlung der Zuständigkeit weggelassen. Diese Art der Anfrage muss, wie in der Architektur beschrieben, von jedem Applikationsserver beantwortet werden können.

5.2 Umsetzung

Im folgenden Kapitel wird auf die Umsetzung des Doodling-Systems eingegangen. Zunächst werden die verwendeten Bibliotheken erläutert, danach der Implementierungsvorgang besprochen.
Für die Umsetzung des Projektes werden folgende Bibliotheken und Frameworks verwendet:

- Unity3D Game-Engine (www.unity3d.com)
- Spray (www.spray.io)
- Akka (www.akka.io)
- Hibernate (www.hibernate.org)
- Jackson (http://wiki.fasterxml.com/JacksonHome)
- MySQL (www.mysql.de)
- SLF4J (www.slf4j.org)
- Log4J (http://logging.apache.org/log4j/2.x/)
- Redis (www.redis.io)

Da bereits im Entwurf detailliert auf Unity3D, Spray und Akka eingegangen wurde, wird hier auf eine nähere Erläuterung des Einsatzes verzichtet.
Hibernate ist eine Bibliothek, die Funktionalitäten für das Überführen von Objekten auf relationale Datenbanken bereitstellt. Sie ist frei verfügbar und wird für das Doodling-System in Verbindung mit der ebenfalls frei verfügbaren relationalen Datenbank MySQL für die persistente Speicherung der Entitäten des Systems auf Serverseite verwendet.
Jackson ist eine Bibliothek für die Erzeugung von JSON-Objekten aus Java-Objekten. Sie wird für die Serialisierung der Antwort-Nachrichten des Webservers verwendet.
SLF4J (Simple Logging Facade for Java) stellt eine Fassade für das Logging bereit, damit Logging-Bibliotheken ausgetauscht werden können, ohne die Anwendung zu verändern. Diese wird zunächst mit der Logging-Bibliothek Log4J verwendet.

Redis wird für den Webserver als Key-Value Store verwendet. Redis ist eine NoSQL-Datenbank, die für die Verarbeitung von einfachen Datenstrukturen (zum Beispiel Strings) in Key-Value Paaren konzipiert ist. Sie arbeitet In-Memory und ist deswegen besonders schnell zugreifbar. Allerdings werden die Einträge dadurch auch nicht persistent gespeichert. Dies führt zu erhöhtem Aufwand nach einem Neustart des Servers, da die zuständigen Applikationsserver wieder neu gefunden werden müssen. Da der Neustart eines Servers nur in Ausnahmefällen durchgeführt werden soll, ist die durch die schnellere Zugriffzeit verminderte Last des Webservers als wichtiger zu bewertet worden.
Für das Buildmanagement und die Abhängigkeitsverwaltung wird auf der Seite des Servers Maven verwendet. Die Projektstruktur ist so aufgebaut, dass es für jede Komponente zwei Projekte gibt. Ein sogenanntes API-Projekt (Application Programming Interface) und ein Implementierungs-Projekt, welches die API implementiert (siehe Abbildung 19). Der Vorteil der Aufteilung ist, dass die Implementierung der Komponenten ausgetauscht werden kann, ohne dass die anderen Komponenten davon beeinträchtigt werden. Beim Client übernimmt Unity die Buildmanagement und Abhängigkeitsverwaltung. Die direkte Abhängigkeit des „application“ Projektes zu den Projekten „region“, „doodling“ und „player“ liegt daran, dass die Implementierung dieser Komponenten für das Starten des „application“ Projektes initialisiert werden muss. Außer der Initialisierung werden innerhalb des „application“ Projektes nur die durch die API-Projekte angebotenen Schnittstellen verwendet.
Für die Versionsverwaltung der Komponenten wird Git verwendet. Dabei gibt es zwei Repositorien, jeweils eins für Server und Client.

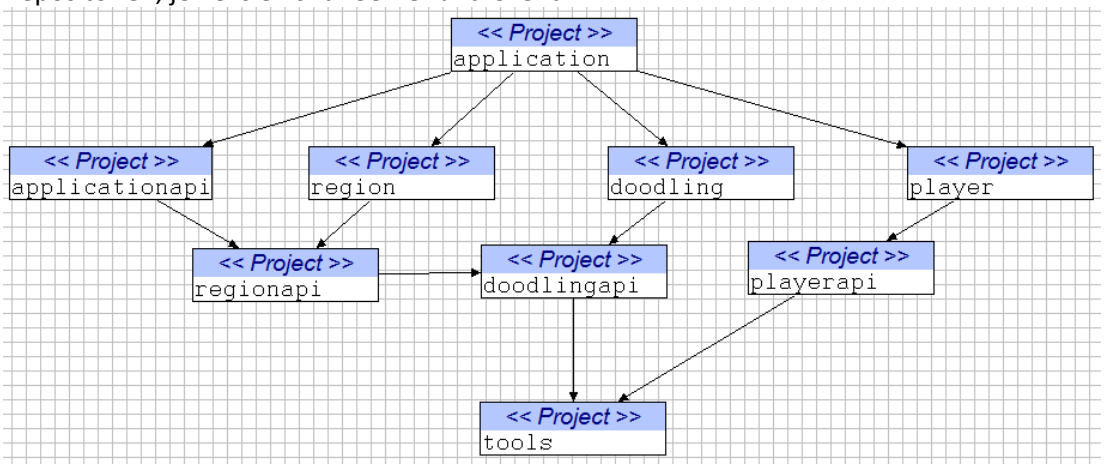

Abbildung 19 Projektabhängigkeiten

5.2.1 Grafische Benutzungsoberfläche

Für die Erzeugung der Zeichnungen und Animationen wurde die Software GIMP 2.8 (www.gimp.com) verwendet. In Abbildung 20, Abbildung 21 und Abbildung 22 sind die Szenen dargestellt. Hierbei wird auf der Kompasssicht zusätzlich das Detailfenster für ein Doodling angezeigt.

Abbildung 20 Kompassszene

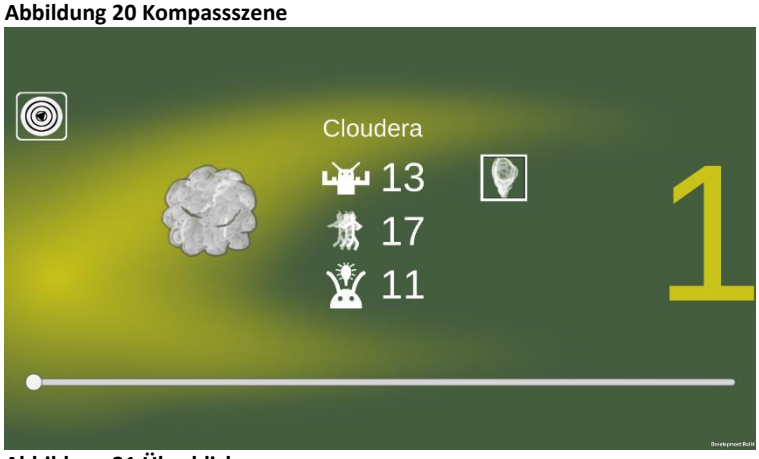

Abbildung 21 Überblickszene

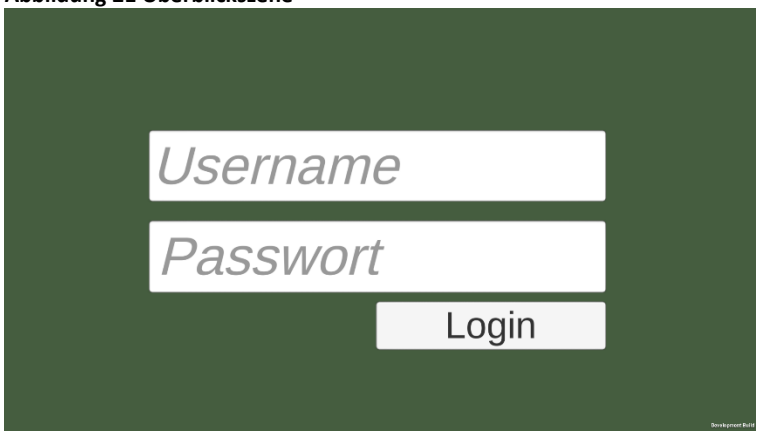

Abbildung 22 Loginszene

5.2.2 Änderungen zum Entwurf

Während der Umsetzung des Systems sind an verschiedenen Punkten Probleme aufgetreten, die eine im Vergleich zum Entwurf veränderte Umsetzung erzwungen haben.

Zunächst musste die Schnittstelle des Webservers insofern geändert werden, als dass alle authentifizierten Aufrufe als HTTP-Methode POST verwenden müssen, da die Unity interne Umsetzung von HTTP-Requests eine Veränderung der Headerfelder für die Methode GET nicht erlaubt.
Außerdem wurde die Zuständigkeit für das Auslesen der zu einem Spieler gehörenden Doodlings in die Doodling-Verwaltung Komponente verlegt, da einem Spieler beliebig viele Doodlings zugeordnet werden können, aber jedem Doodling nur ein Spieler. Deshalb kann die Abfrage ohne Join getätigt werden und ist dadurch wesentlich performanter.
Des Weiteren wurde die Schnittstelle der Region Komponente (IRegionService) angepasst. Die Methode „getLocationContent“ gibt jetzt ein Ergebnisobjekt zurück, in dem sowohl die ursprünglich geplante Liste als auch eine Liste der aus dem Zuständigkeitsbereich der Region Komponente bewegten dynamischen Spielinhalte übergeben werden. Dies war notwendig, da diese in der Hierarchie nach oben gegeben werden müssen, um den zuständigen Applikationsserver zu finden. Zusätzlich wurde die Rückgabe der Schnittstellenmethode „isResponsible(geoRegion: GeoRegion)“ geändert. Diese gibt nun keinen Boolean mehr zurück sondern einen Integer, da es drei Möglichkeiten der Zuständigkeit gibt: Vollständig, teilweise oder gar nicht zuständig.

5.3 Test

In diesem Abschnitt wird erläutert welche Testverfahren für die Sicherstellung der Korrektheit des Systems verwendet werden und welche Anforderungen mit welchen Methoden überprüft werden. Dabei werden zwei unterschiedliche Herangehensweisen eingesetzt, zum einen automatisierte Tests, zum anderen manuelle Tests. Die manuellen Tests werden von einer Gruppe projektfremder Probanden ausgeführt.

5.3.1 Automatisierte Tests

Für jede Komponente des Systems werden automatisierte Unit Tests angelegt. Diese werden mithilfe des Testframeworks JUnit bzw. ScalaTestKit erzeugt. Sie werden verwendet, um die korrekte Fachlichkeit der einzelnen Komponenten sicherzustellen. Zusätzlich wird das Server System ebenfalls als Gesamtkonstrukt automatisiert über die Restschnittstelle getestet. Die Tests werden anhand der Anwendungsfälle erstellt. Dies nennt sich „anwendungsfallbasiertes Testen“ [18]. Zusätzlich wird der Server mit dem Testframework JMeter[11] auf seine Performance und Antwortzeit getestet.

5.3.2 Manuelle Tests

Zusätzlich zu den automatisierten Tests wird ein Test mit Probanden durchgeführt, welche die erste Ausbaustufe des Spiels spielen sollten. Dabei werden 15 Probanden der zuvor definierten Zielgruppe eingesetzt. Danach wurden die Probanden dazu angehalten, einen Fragebogen auszufüllen (siehe Abbildung 51 und Abbildung 52). Der Test wird als Blackbox-

[11] http://jmeter.apache.org/

Test durchgeführt, das heißt, die Probanden haben keinerlei Informationen über den inneren Aufbau des Doodling-Systems [18]. Dabei soll der Proband mindestens ein Doodling fangen und das Erlebnis mit Hilfe des Fragebogens evaluieren. Im Umfang des Tests sollten somit von jedem Probanden die Anwendungsfälle 1, 2 und 3 durchgeführt werden.
Da ein automatisierter Test der grafischen Benutzungsoberfläche innerhalb Unitys nicht möglich ist und es eine große Vielfalt an unterschiedlichen Android-Smartphones gibt, wurde der manuelle Test auch für die Überprüfung der Korrektheit des Clients verwendet. Zusätzlich dazu wurde das Spielprinzip evaluiert, damit die Wünsche der Spieler in die weitere Entwicklung mit einfließen können. Zuletzt wurden die nicht funktionalen Anforderungen NFR3 und NFR4 überprüft, die automatisiert ebenfalls nicht getestet werden können.

5.4 Auswertung

In diesem Kapitel werden die Ergebnisse der Tests der Umsetzung ausgewertet und Schlüsse gezogen, welche Inhalte, Verbesserungen und Erweiterungen die zweite Ausbaustufe des Systems enthalten sollte. Zunächst werden die Testergebnisse der automatisierten Tests, dann die des manuellen Tests evaluiert.

5.4.1 Auswertung der automatisierten Tests

Wie bereits erläutert, wurden die automatisierten Tests anhand der Anwendungsfälle entworfen. Diese überprüfen die korrekte Implementierung der Funktionalität. Neben der Korrektheit der Testfälle ist es wichtig, dass auch die gesamte Implementierung getestet wird. Dies wird Coverage oder auch Anweisungsüberdeckung genannt, also die Abdeckung des Source-Codes durch die Testfälle [18]. In Tabelle 6 wird die Coverage der einzelnen Komponenten durch ihre Komponententests gezeigt. Dabei wäre es wünschenswert, wenn die Testfälle die gesamte Codebasis abdecken, also einen Wert von 100% erreichen würden.

Tabelle 6 Coverage Analyse automatisierte Tests

Komponente	**Coverage in %**
Applikation	83,5
Doodling-Verwaltung	64,6
Player	77,2
Region	75,1
Tools	95,4

In realer Software ist ein Coverage Wert von über 85% meist unrealistisch, da bestimmte Anweisungen (zum Beispiel Anweisungen die als Fehlerbehandlung ausgeführt werden) schwer durch einen Testfall angesteuert werden können [18]. Bis auf die Doodling-Verwaltung Komponente ist der Coverage-Wert bei jeder Komponente also ausreichend. Die nachträgliche Überprüfung der ausgeführten Anweisungen ergab, dass die nicht ausgeführten Anweisungen zumeist Fehler und Getter/Setter Anweisungen waren. Diese haben in der Doodling-Verwaltung Komponente einen großen Anteil an der Gesamtzahl der

Anweisungen (vgl. „Abbildung 13 Innensicht Doodling-Verwaltung“). Dies erklärt den geringen Coverage-Wert von 64,6. Für die Ermittlung der Coverage wurde das Eclipse Plugin ECLEmma[12] verwendet.
Die Überprüfung der Antwortzeit und Performance des Webservers mit JMeter ergab Werte, welche die nicht funktionalen Anforderungen zur Performance (NFR2) vollständig erfüllen. Der zur Überprüfung verwendete Server verfügte über 6 GB Arbeitsspeicher, sowie zwei CPU Kernen der Bauart „Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz“.
Für die Überprüfung wurden 100 Spieler simuliert. Ein simulierter Spieler ruft dabei alle fünf Sekunden die Spielinhalte in seiner Umgebung ab, da dies auch im Spiel so konfiguriert ist. Dabei wurden mithilfe von JMeter die Zeiten zwischen dem Senden der Anfrage bis zum Erhalt der Antwortnachricht protokolliert. Der Verlauf der Nachrichtenlaufzeiten ist in Abbildung 23 nachzuvollziehen. Die durchschnittliche Nachrichtenlaufzeit betrug 59 Millisekunden. Für die Berechnung der tatsächlichen Bearbeitungszeit pro Anfrage wird die Zeit des Verbindungsaufbaus aus der Nachrichtenlaufzeit entfernt. Dies ergibt eine durchschnittliche Bearbeitungszeit von 31 Millisekunden. Die ermittelten Daten können im Anhang eingesehen werden.

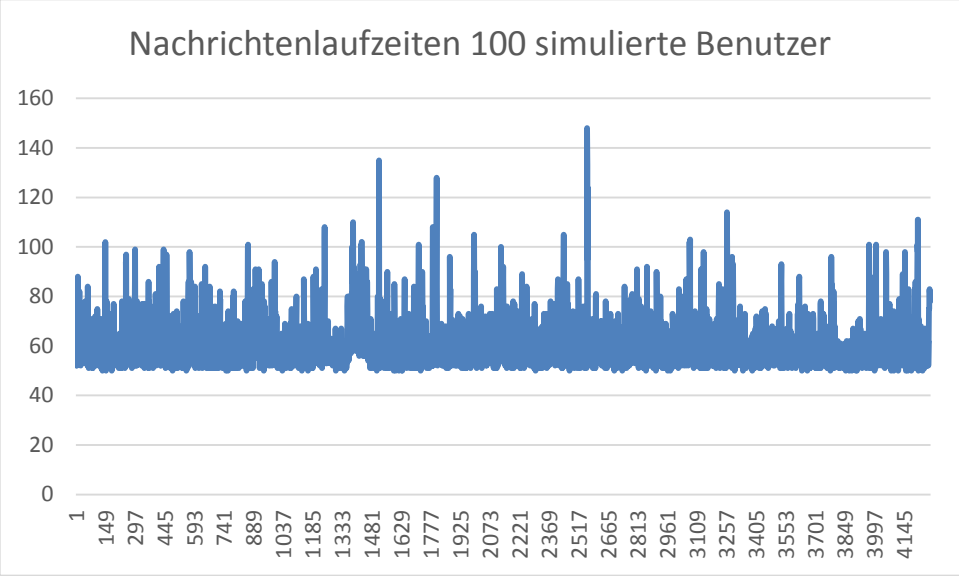

Abbildung 23 Nachrichtenlaufzeiten

[12] http://eclemma.org/

5.4.2 Auswertung der manuellen Tests

Die Auswertung des manuellen Tests ist für die weitere Entwicklung des Spiels von großer Bedeutung, da diese sich an den Wünschen der Spieler orientieren soll, um somit ein nachhaltiges Erlebnis generieren zu können.

Von den 15 Probanden, die den manuellen Test durchführen sollten, haben zehn den Fragebogen ausgefüllt. Die detaillierten Antworten aller Probanden sind im Anhang zu finden. Leider ist eine geschlechtsneutrale Auswertung der Ergebnisse nicht möglich, da nur 20% der Teilnehmer weiblich waren (siehe Abbildung 24). Die Hälfte der Probanden haben mehr Doodlings gefangen, als sie aufgrund der Aufgabe fangen sollten (siehe Abbildung 25). Dies könnte bedeuten, dass die Probanden die Aufgabe als interessant empfanden. Belegt wird die Annahme durch die Antworten zu Frage 6 (siehe Abbildung 26), indem dem Spielprinzip im Mittel der Spaßfaktor „hoch“ zugeordnet wurde. Dabei ist auffällig, dass niemand eine Wertung von niedrig und darunter angegeben hat. Die Probanden bewerteten das Spielerlebnis im Mittel als „innovativ“ bis „sehr innovativ“. Das zeigt, dass die Art des Spiels den meisten Probanden noch nicht bekannt war und Spiele dieser Art, obwohl es einige Vertreter des Genres bereits lange gibt, immer noch als neu und innovativ empfunden werden. Die Bewertung der Schwierigkeit des Spiels zeigt, dass die Schwierigkeit ausgewogen ist und es weder als zu schwer, noch als zu leicht empfunden wurde.

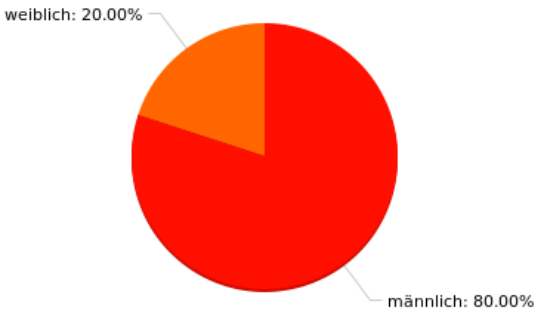

Abbildung 24 Probanden nach Geschlecht

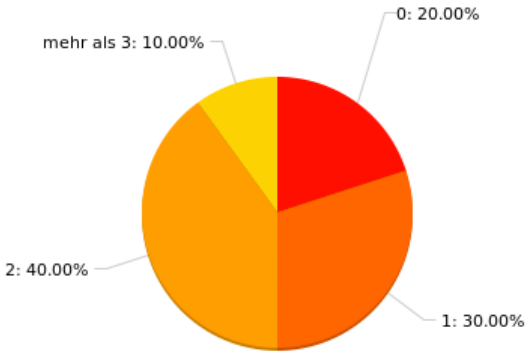

Abbildung 25 Gefangene Doodlings pro Proband

	sehr hoch (1)		hoch (2)		mittel (3)		niedrig (4)		sehr niedrig (5)		keine Angabe (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Spaßfaktor	2x	20,00	5x	50,00	3x	30,00	-	-	-	-	-	2,10	0,74
Innovativität	4x	40,00	4x	40,00	2x	20,00	-	-	-	-	-	1,80	0,79
Übersichtlichkeit	1x	10,00	5x	50,00	3x	30,00	1x	10,00	-	-	-	2,40	0,84
Schwierigkeit	-	-	1x	10,00	6x	60,00	3x	30,00	-	-	-	3,20	0,63

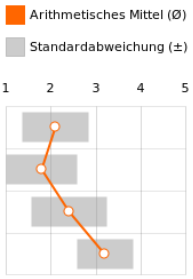

Abbildung 26 Auswertung Spielbewertung

Ein Problem scheint die Übersicht des Spiels darzustellen. Diese wurde im Mittel zwar zwischen „hoch“ und „mittel“ bewertet, aber eine Einzelbewertung hat die Übersichtlichkeit sogar als niedrig eingestuft. Da die Übersichtlichkeit direkte Einwirkungen auf die nicht funktionale Anforderung der Nutzbarkeit (siehe NFR3) hat, muss diese im hohen Maß gewährleistet sein. Zur Überprüfung, warum die Probanden die Übersichtlichkeit als mittel oder niedrig einstufen, werden die Verbesserungsvorschläge dieser Probanden überprüft. Diese sind (siehe Abbildung 52 Frage 12):

- Das Spiel sollte auch im Hochformat spielbar sein
- Nutzername sollte speicherbar sein und nicht jedes Mal neu eingegeben werden
- Die Doodlings, je nach Rasse, unterschiedlich auf der Karte darstellen

Diese Anpassungen am Spiel sollten in der nächsten Ausbaustufe unbedingt umgesetzt werden, da alle Punkte in ähnlicher Form auch von anderen Probanden als Verbesserungsvorschläge genannt werden (siehe Abbildung 52 Frage 12). Außerdem wird in den Verbesserungsvorschlägen die Kompassnadel als wenig intuitiv bezeichnet und ein drehen des kompletten Bildschirms in die Blickrichtung des Spielers von mehreren Probanden gewünscht. Auch dies sollte in die nächste Ausbaustufe aufgenommen werden.
Bei der Durchführung des Tests sind 50% der Spieler Fehler aufgefallen. Diese stehen zum größten Teil mit der Lokalisierung durch das GPS Modul im Zusammenhang. Folgende Fehler wurden von den Spielern gemeldet (siehe Abbildung 51 Frage 8):

- *„Die Entfernungsangaben scheinen ungenau zu sein.“*
- *„Es sagt, dass mein Passwort falsch ist, wenn nur die Internetverbindung fehlschlägt“*
- *„Position ungenau wird häufig angezeigt, ein Fangen von Doodlings ist dann nicht möglich. Ursache wird im fehlerhaften GPS Modul des verwendeten Handy vermutet“*
- *„Anscheinend umher springende Doodlings. 50-100m Distanz Veränderung innerhalb von ein paar Sekunden“*
- *„Pfeilspitze des Radars relativ schwer die Richtung zu erkennen und dieser schlägt die ganze Zeit aus vielleicht etwas korrelieren.“*

Die erste, dritte und vierte Meldung sind sehr deutlich auf Ungenauigkeiten des GPS Moduls zurückzuführen. Die Genauigkeit der Positionsdaten des GPS-Moduls der getesteten Smartphones weisen häufig eine horizontale Genauigkeit von mehr als 45 Metern auf. Das bedeutet, dass die tatsächliche Position des Empfangsgerätes zu 66,6% innerhalb eines Radius von 45 Metern von der zurückgegebenen Position liegt. Der Wert, ab dem eine Position innerhalb des Doodlings-Systems als „ausreichend genau“ gewertet wird, wurde für die Testphase auf 45 Meter gesetzt, die gewünschte Genauigkeit, die dem GPS-Modul übermittelt wird, liegt bei 10 Metern. Durch die Erhöhung dieser Grenze könnte man verhindern, dass die Meldung „Position ungenau“ angezeigt wird, allerdings erhöht dies auch die „Sprünge“, die ein Doodling macht. Als Kompromiss könnte eine Approximation der

aktuellen Position anhand der Geschwindigkeit und der vorhergehenden Positionsdaten gemacht werden. Für die nächste Ausbaustufe sollte Überprüft werden, ob diese Möglichkeit umsetzbar ist. Die beiden anderen gemeldeten Fehler sollten als Korrekturen in der nächsten Ausbaustufe behandelt werden.
Neben der Betrachtung des Ist-Zustandes des Spiels sollten die Probanden ebenfalls eine Einschätzung der Wichtigkeit zukünftiger Features abgeben. Das Ergebnis der Umfrage ist in Tabelle 7 dargestellt. Dabei geben die Zeilen die Bewertungen der einzelnen Probanden an. In der letzten Zeile ist die Durchschnittsbewertung des jeweiligen Features eingetragen. Bei der Bewertung sollten die Features vom Probanden in eine Reihenfolge gebracht werden, so dass „1“ das wichtigste Feature und „6“ das unwichtigste Feature kennzeichnen.

Tabelle 7 Bewertung zukünftige Features

Erstellen neuer Doodlings durch Spieler	Erstellen von Aufgaben und Rätseln durch Spieler	Kämpfe zwischen Spielern mit Doodlings	Gemeinsame Kämpfe von Spielern gegen besonders starke Doodlings	Gegenstände zum leichteren Einfangen/Verbessern von Doodlings	Doodlings als Tamagotchi
3	2	5	4	1	6
5	6	1	3	2	4
2	3	1	4	5	6
1	5	2	3	4	6
3	4	2	5	6	1
4	5	1	3	2	6
1	5	2	3	4	6
5	3	1	4	6	2
5	1	2	3	4	6
5	4	1	2	3	6
3,4	3,8	1,8	3,4	3,7	4,9

Mit großem Abstand werden Kämpfe zwischen Spielern als wichtigstes Feature bewertet. Dieses Feature ist auch bereits in der Planung für Ausbaustufe 2 vorgesehen und sollte implementiert werden. Das weitere für Ausbaustufe 2 geplante Feature, nämlich das Erstellen von Aufgaben und Rätseln durch Spieler, wird hingegen von den Probanden als eher unwichtig empfunden. Deswegen sollte in der nächsten Ausbaustufe entweder das Erstellen von Doodlings durch Spieler oder die gemeinsamen Kämpfe mit aufgenommen werden. Beide Features werden im Mittel als gleichwichtig empfunden. Aufgrund dessen, dass auch das Einbeziehen der Spieler ein elementarer Bestandteil der Spielidee ist und die Features

„gemeinsamer Kampf“ und „Spieler gegen Spielerkampf“ sehr ähnlich sind, sollte das Erstellen neuer Doodlings durch Spieler in die nächste Ausbaustufe aufgenommen werden.
Als letztes werden die Antworten auf die Fragen 9 und 10 (siehe Fragebogen im Anhang) ausgewertet. Diese zielen auf die Überprüfung der nicht funktionalen Anforderung zum Ressourcenverbrauch (NFR 4). Durch die erwähnte große Vielfalt an unterschiedlichen Android Geräten ist es wichtig, dass die Anforderung flächendeckend und nicht nur für einen Gerätetyp erfüllt ist. In Tabelle 8 sind die Angaben der Probanden zum Akkuverbrauch und der Netzwerknutzung zusammengefasst und auf den Verbrauch innerhalb einer Stunde normalisiert. Während die Netzwerknutzung bei jedem Probanden weit unter der durch NFR4 definierten Schranke von 10 MB pro Stunde liegt, sind die Werte für den Akkuverbrauch in einem Fall über dem definierten Wert von 20% Akkuverbrauch pro Stunde. Da die Werte für die Spieldauer von den einzelnen Probanden nur Schätzungen sind, sind die Werte der Tabelle nicht komplett verlässlich, außerdem ist nicht klar, ob Werte die mit 0 angegeben sind tatsächliche Antworten sind (zum Beispiel bei der Netzwerknutzung bei weniger als 500 KB abgerundet wurden) oder ob der Proband den entsprechenden Wert nicht herausfinden konnte. Letzterer Fall ist vor allem bei den Angaben zum Akkuverbrauch wahrscheinlich, da die anderen Werte deutlich über einem Prozent liegen. Für die Entwicklung der weiteren Ausbaustufen kann aus den Ergebnissen geschlossen werden, dass bei der Netzwerknutzung Spielraum für weitere Funktionen besteht, während bei der Implementierung neuer Funktionen ein besonderes Augenmerk auf die Auswirkung auf den Akkuverbrauch gelegt werden sollte.

Tabelle 8 Akkuverbrauch und Netzwerknutzung

	Akkuverbrauch in % pro Stunde	Netzwerknutzung in MB pro Stunde
	20	2,4
	0	0,196
	23	1,1
	2,5	0,2765
	10	0
	11	1,4
	6	1,4
	0	0
	2	0
	14	0
Durchschnitt	8,85	0,67725

6 Zweites Inkrement

Das zweite Inkrement erweitert und verbessert das Doodling-System, dass mit dem ersten Inkrement entwickelt wurde. In diesem Abschnitt wird zunächst der Umfang des zweiten Inkrements definiert, danach wird es entworfen, umgesetzt, getestet und abschließend ausgewertet.

6.1 Umfang

Im zweiten Inkrement sollen, laut Planung, zwei neue Spielmechaniken in das Spiel integriert werden. Anders als unter „Tabelle 4 Ausbaustufen vorläufige Planung“ geplant, hat die Auswertung des ersten Inkrements ergeben, dass anstatt der Anforderung FR5 „erstellen von Quests“ die Anforderung FR8 „erstellen von Doodlingtypen durch Spieler“ in diesem Inkrement umgesetzt werden soll, da sie von den Spielern als interessanter bewertet wurde. Außerdem sollen, wie geplant und durch die Auswertung bestätigt, Kämpfe zwischen Spielern implementiert werden.
Des Weiteren sollen Folgende, von den Probanden vorgeschlagene Verbesserungen und Korrekturen umgesetzt werden:

- Das Spiel sollte auch im Hochformat spielbar sein
- Nutzername sollte speicherbar sein und nicht jedes Mal neu eingegeben werden
- Die Doodlings je nach Rasse unterschiedlich auf der Karte darstellen
- Anstatt der Kompassnadel den kompletten Bildschirm drehen
- Genauere Fehlermeldungen beim Login

Diese betreffen direkt bereits implementierte Funktionalitäten. Im Fall der Spielbarkeit im Hochformat ist eine frühe Umsetzung von Vorteil, damit die Benutzungsoberfläche gleich an diese Anforderung angepasst werden kann und nicht im Nachhinein aufwendig verändert werden muss.

6.2 Entwurf

Im Folgenden werden die Systemerweiterungen für die beiden zu entwickelnden Spielmechaniken entworfen. Am Ende wird kurz auf die nötigen Schritte zur Verbesserung der im Test des letzten Inkrements aufgefallenen Fehler eingegangen.

6.2.1 Kampf zwischen Spielern

Der Kampf zwischen zwei Spielern (definiert durch die funktionale Anforderung FR4) besteht aus drei Phasen, die entworfen werden müssen:

- Matchmaking
- Kampf
- Auswertung

Beim Matchmaking werden zunächst geeignete Opponenten ausgewählt, die den Kampf bestreiten sollen. Dabei spielen sowohl technische Parameter, wie die Qualität der Verbindung zwischen den Opponenten als auch fachliche Parameter, wie die Erfahrung der einzelnen Spieler [28], eine Rolle.
Im Kampf kontrollieren die Opponenten die Aktionen ihrer Doodlings in Echtzeit, bis ein Spieler den Kampf gewonnen hat.
In der Ergebnisphase wird der Kampf ausgewertet und die Belohnungen, in Form von Erfahrungspunkten, den Doodlings zugeordnet. Dem Spieler muss diese Auswertung für sich angezeigt werden.

6.2.1.1 Erweiterung der Architektur

Für die Erweiterung der Funktionalität um den Kampf zwischen zwei Spielern muss die Architektur erweitert werden. Dies ist notwendig, da ein Kampf zwischen Spielern unterschiedlicher Regionen stattfinden kann. Da die für den Spieler zuständigen Applikationsserver durch die letzte Position bestimmt werden, sind diese für die Verarbeitung der Kämpfe ungeeignet. Außerdem wird durch die Aufteilung die Last auf den Applikationsservern verringert und die Verteilung vereinfacht. Es werden jeweils Server für das Matchmaking und für die Verarbeitung des Kampfes entworfen. Dabei greifen die Matchmaking-Server auf die Kampf-Server zu und erstellen, wenn geeignete Opponenten gefunden wurden, einen neuen Kampf für diese. Die erweiterte Architektur ist in Abbildung 27 dargestellt.

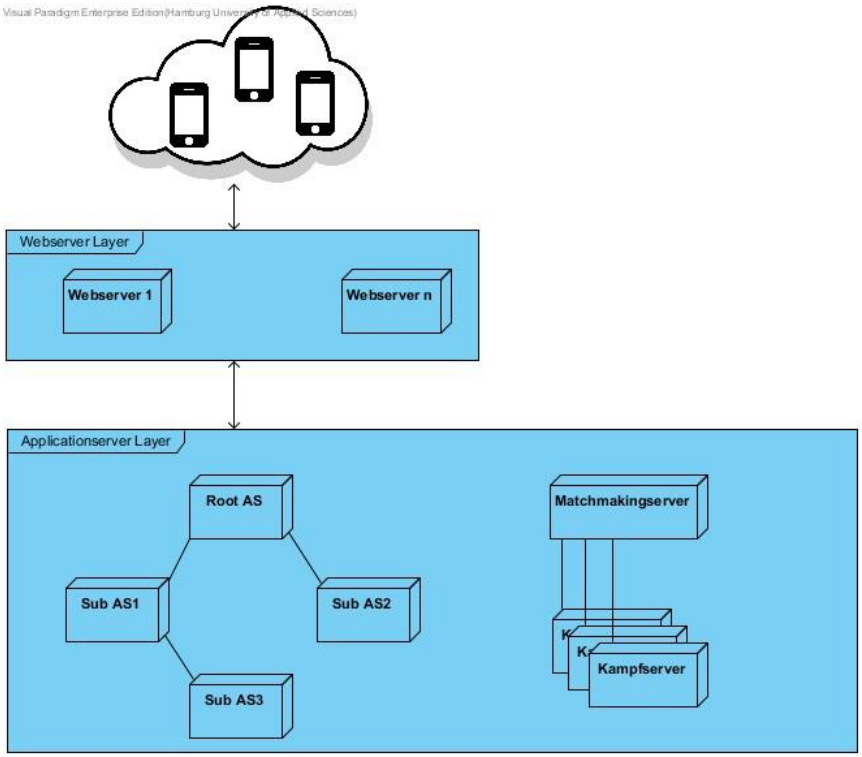

Abbildung 27 Erweiterte Architektur

6.2.1.2 Matchmakingserver

Der Matchmakingserver ist für das Matchmaking der Opponenten zuständig. Spieler, die einen Kampf bestreiten müssen, senden über den Webserver eine Anfrage. Danach wird automatisiert ein passender Gegenspieler gesucht. Die Bausteinsicht des Matchmakingservers ist in Abbildung 28 dargestellt. Die Schnittstellen der Doodling- und Playerkomponente werden für die Bedürfnisse des Matchmakingservers erweitert.
Ein passender Gegner wird anhand von zwei Eigenschaften ausgewählt. Zum einen muss die Erfahrung der Opponenten ähnlich hoch sein. Diese wird durch den Level einer konfigurierbaren Anzahl der Doodlings des Spielers, die das höchste Level (Attribut Levelnummer) haben, berechnet. Die Formel für die Berechnung ist:

$$W_{Spieler} = höchstes\ Level + 2.\ höchstes\ Level * \frac{1}{2} + n.höchstes\ Level * \frac{1}{n}$$

Die Abschwächung wird deshalb vorgenommen, damit ein Spieler mit nur einem Doodling, das ein sehr hohes Level hat, nicht gegen einen schwächeren Spieler mit vielen Doodlings auf niedrigem Level antreten kann. Für die Berechnung des Wertes muss die Playerkomponente um diese Funktionalität erweitert werden.

Zum anderen müssen auch technische Merkmale, wie die Latenz zwischen den beiden Spielern, berücksichtigt werden. Da Doodlings ein positionsbasiertes Spiel ist, bietet es sich hier an, die Position des Spielers als Approximation für die Latenz zu nutzen. Das heißt, Opponenten die geographisch nah beieinander liegen, werden als besseres Match gewertet. Beim Matchmaking wird dann der Radius für die Suche nach Opponenten in konfigurierbaren Zeitintervallen bis zu einem Maximalwert erweitert. Nachdem dieser Maximalwert erreicht ist, werden auch Opponenten gematched, deren Erfahrungen weit auseinander liegen. Auch der Wert der Erfahrungsabweichung muss Schrittweise erhöht werden und einem Maximum unterliegen. Formal dargestellt ergeben sich also folgende Beziehungen zwischen dem Erfahrungswertmatching W, dem Distanzmatching D und dem Gesamtmatching M:

$$W \Leftrightarrow \left(abs(W_{Spieler1} - W_{Spieler2}) < Max_W * \frac{Min(Zeit_{Spieler1}, Zeit_{Spieler2}, Zeit_{bisMaxW}) - Zeit_{bisMaxDistanz}}{Zeit_{bisMaxW} - Zeit_{bisMaxDistanz}} \right)$$

Wenn $Zeit_{Spieler1}$ und $Zeit_{Spieler2} > Zeit_{bisMaxDistanz}$ sonst:

$$W \Leftrightarrow \left(abs(W_{Spieler1} - W_{Spieler2}) < 1 \right)$$

$$D \Leftrightarrow \left(Distanz_{Spieler1,Spieler2} < Max_{Distanz} * \left(\frac{Min(Zeit_{Spieler1}, Zeit_{Spieler2}, Zeit_{bisMaxDistanz})}{Zeit_{bisMaxDistanz}} \right) \right)$$

$$M \Leftrightarrow W \wedge D$$

Nachdem das Matchmaking für zwei Opponenten abgeschlossen ist, erstellt der Matchmakingserver einen Kampf über die vom Kampfserver angebotene Schnittstelle und gibt ein entsprechendes Ticket für diesen Kampf über den Webserver an die Opponenten zurück. Das Ticket enthält die Aktorreferenz für den erstellten Kampf sowie die Spierleridentifikation des Spielers, für den es ausgestellt wurde. So kann die Kommunikation des Spielers mit dem Webserver stateless ablaufen.

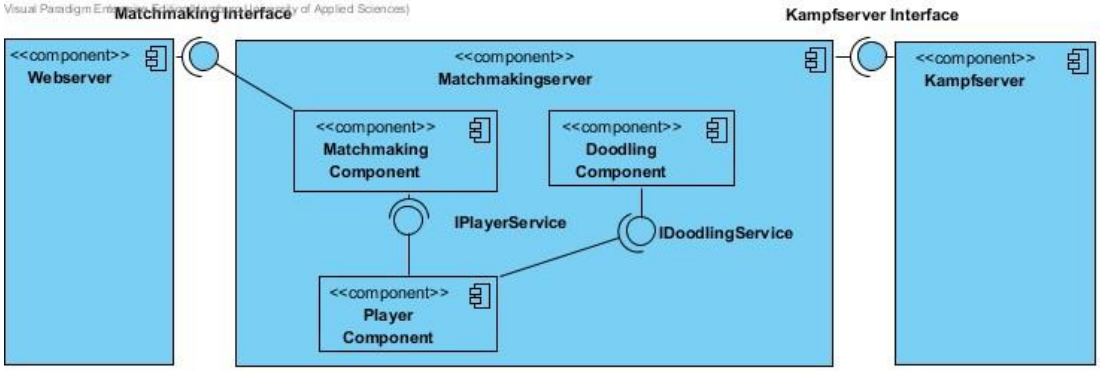

Abbildung 28 Matchmakingserver Bausteinsicht

6.2.1.3 Kampfserver

Der Kampfserver hat die Aufgabe, die Kämpfe der Spieler zu leiten. Dazu melden sich die Opponenten mit dem durch das Matchmaking erhaltene Ticket für einen Kampf an. Danach wird der Kampf in zwei Phasen ausgeführt:

- Auswahlphase
- Kampfphase

In der Auswahlphase werden von jedem Spieler die eingesetzten Doodlings ausgewählt und in eine Reihenfolge gebracht, in der sie beim Kampf eingesetzt werden sollen. Dabei werden bis zu sechs Doodlings gewählt. Sind mindestens einem Spieler weniger als sechs Doodlings zugeordnet, werden so viele Doodlings gewählt, wie der Spieler mit den wenigsten zugeordneten Doodlings zur Verfügung hat.
Nachdem die Auswahl getroffen wurde, wird der Kampf gestartet. Der Kampf findet in Echtzeit statt, das heißt es wird nicht abwechselnd eine Aktion ausgewählt, sondern gleichzeitig. Dabei liegt zwischen jeder ausgeführten Aktion eine Zeitspanne, die sogenannte Abklingzeit. Diese soll konfigurierbar sein und zunächst bei fünf Sekunden liegen. Die Aktionen, die ein Spieler ausführen kann, ergeben sich aus den „DoodlingAbility“ (siehe Abbildung 13), die dem Doodling zugeordnet sind, welches gerade aktiv am Kampf teilnimmt. Der Ablauf eines Kampfes ist in Abbildung 30 grafisch dargestellt.
Pro Kampf wird vom Kampfserver jeweils ein Aktor erzeugt, der den Zustand des Kampfes kapselt. Der Aufbau und die Abhängigkeiten des Kampfservers werden in Abbildung 29 gezeigt.

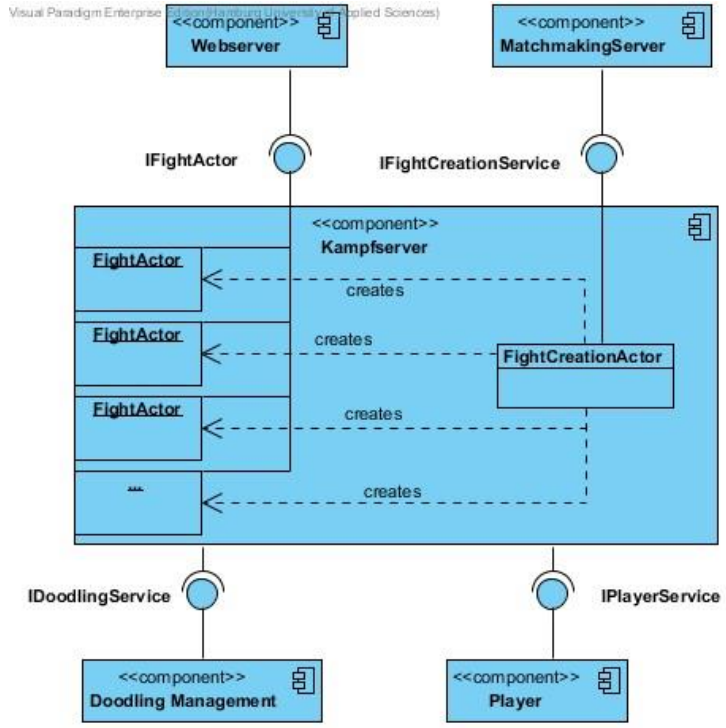

Abbildung 29 Innensicht Kampfserver

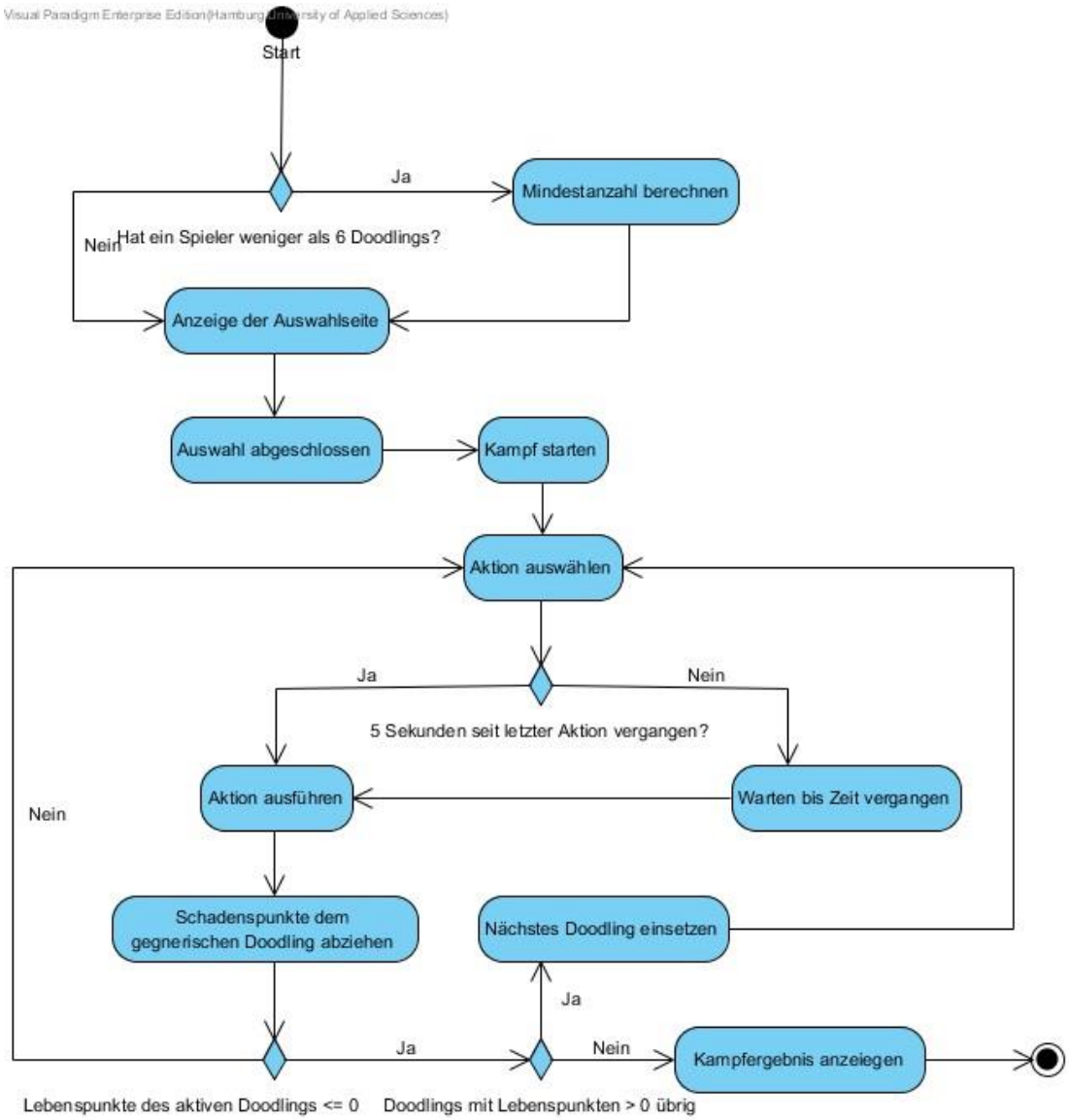

Abbildung 30 Kampfablauf

6.2.1.3.1. Beschreibung der Spielmechanik

Damit sich unterschiedliche Doodlings unterschiedlich im Kampf verhalten werden die Attribute der Doodlings im Kampf mit berücksichtigt. Die Doodlings werden dabei durch die „Doodling“ Entität, der Doodling-Verwaltung Komponente symbolisiert (siehe Abbildung 13). Dabei haben die Attribute unterschiedlichen Einfluss. Der Einfluss der Attribute kann in Tabelle 9 nachvollzogen werden. Die Anzahl an Stärke-, Intelligenz- bzw. Beweglichkeitspunkten wird anhand der Zahlenwerte in der Doodling Entität und dessen

DoodlingType Entität berechnet. Dabei wird folgende Formel verwendet (hier für Stärkepunkte):

$$SP = DS + DTS + DTSPL * LN$$

Dabei steht DS für das Attribut „strength“ der Entität Doodling, DTS für das Attribut „baseStrength“ der Entität DoodlingType, DTSPL für das Attribut „strengthPerLevel“ der Entität DoodlingType und LN für das Attribut „nr“ der Level Entität, die dem Doodling zugeordnet ist. Für die Intelligenz- bzw. Beweglichkeitspunkte ist die Berechnung äquivalent mit den Attributen „intelligence“ bzw. „agility“.
Nachdem eine Aktion ausgewählt wurde, werden die Schadenspunkte, die diese Aktion verursacht hat, von den Lebenspunkten des gegnerischen Doodlings abgezogen. Die Schadenspunkte (SchP) berechnen sich durch folgende Formel:

$$\mathrm{SchP} = (AS + DS) * ED$$

AS steht dabei für einen Wert aus dem Intervall des Attributes „damage“ der eingesetzten „DoodlingAbility“. Welcher Wert aus dem Intervall gewählt wird, wird dabei zufällig bestimmt. DS steht für das Attribut „strength“ der Doodling Entität. Der Wert für ED wird durch das der „DoodlingAbility“ und das „Doodling“ zugewiesene Element ermittelt. Ist das Element des „Doodling“ in der Liste „strongAgainst“ bzw. „weakAgainst“ des Elementes der „DoodlingAbility“, wird der Wert ED auf 1,25 bzw. 0,75 gesetzt. Trifft keine der beiden Möglichkeiten zu, wird der Wert ED auf 1 festgelegt. Dies führt dazu, dass einige Aktionen mehr Schadenspunkte verursachen als andere und erhöht die Taktikvielfalt innerhalb der Kämpfe. Daraus ergibt sich diese formale Berechnungsformel für den Ausgang eines Kampfes G für einen Kampf zwischen zwei Doodlings:

$$G \Leftrightarrow (\sum_{i}^{n} \mathrm{SchP}_i * Min(IntValue\left(\frac{Random_i(0{,}1)}{0{,}025 * BP_{Gegner}}\right), 1)) > \left(SP_{Gegner} * 25\right)$$

Dabei ist n die Anzahl an Attacken, die während des Kampfes von dem zu bestimmenden Doodling ausgeführt worden sind. BP sind die Beweglichkeitspunkte des Doodlings.
Nachdem der Kampf beendet ist, wird dieser ausgewertet. Dabei werden den Doodlings Erfahrungspunkte zugeordnet, die den Level des Doodlings erhöhen können. Dabei werden die Erfahrungspunkte (EP) für die einzelnen Doodlings wie folgt berechnet:

$$EP = (WL * GE * GEP) * KT$$

Dabei steht WL für den Sieg-Niederlage-Multiplikator. Dieser soll bei einem Sieg auf 1 und bei einer Niederlage auf einen konfigurierbaren Wert gesetzt werden (z.B. 0,5. Dies würde bedeuten, dass der Verlierer 50% der Erfahrungspunkte, die der Gewinner bekommt, erhält). GE steht für die im Matchmaking berechnete Erfahrung des gegnerischen Spielers. GEP ist

ein konfigurierbarer Wert für die Grunderfahrung, die gewährt werden soll. Der Wert von KT wird anhand des prozentualen Anteils der Zeit des Kampfes berechnet, in dem das entsprechende Doodling, für das die Erfahrung berechnet wird, aktiv am Kampf teilgenommen hat.
Mithilfe der Erfahrung und der „Level“ Entität des Doodlings wird anhand des Attributes „experienceForNextLevel“ berechnet, ob das Doodling einen Level aufsteigt. Dies ist der Fall wenn die erhaltene Erfahrung größer ist als der Wert von „experienceForNextLevel“. Ist dies nicht der Fall oder bleiben Erfahrungspunkte übrig, werden diese für das Doodling gespeichert und nach dem nächsten Kampf hinzugefügt.

Tabelle 9 Attributabhängigkeiten

Attribut	Abhängige Werte
Stärke	• Definiert die Lebenspunkte. Ein Stärkepunkt ergibt dabei 25 Lebenspunkte. • Erhöht die Schadenswerte der Attacken pro Stärkepunkt um einen Schadenspunkt
Intelligenz	• Erhöht die Chance, dass das Doodling eine Attacke außerhalb der Abklingzeit ausführt. Jeder Intelligenzpunkt erhöht die Chance um 1%, dass innerhalb der Abklingzeit eine weitere zufällige Aktion ausgeführt wird
Beweglichkeit	• Verringert die Abklingzeit zwischen zwei Aktionen um 10 Millisekunden pro Beweglichkeitspunkt • Erhöht die Chance, einer Attacke auszuweichen um 0.25% pro Beweglichkeitspunkt

6.2.1.3.2. Kommunikation

Die Kommunikation zwischen den Opponenten des Kampfes findet über den Kampfserver statt. Dabei ist der Zustand des Kampfes im Kampfserver gekapselt und die Opponenten übermitteln die gewünschten nächsten Aktionen. Dabei ist das Problem, dass bei einer Änderung des Systemzustands eigentlich der Server die Kommunikation mit den Opponenten anstoßen muss, beziehungsweise es eine Verbindung geben muss, die beidseitig genutzt werden kann. Dies ist mit der aktuellen Architektur nicht möglich. Eine geeignete Lösung wären Websockets. Diese beruhen auf dem TCP-Protokoll und unterstützen die beidseitige Kommunikation von Client und Webserver [14]. Sie werden allerdings nicht von der kostenlosen Unity-Version unterstützt. Deswegen werden von den Opponenten jeweils zwei Kommunikationskanäle erzeugt. Einer für die Synchronisation mit dem Spielzustand und einer zur Übermittlung der gewählten Aktionen. Für jeden Kampf wird ein Aktor erzeugt, der den Zustand des Kampfes hält und über seine Identifikation direkt vom Webserver angesprochen werden kann.

6.2.1.3.3. Erweiterung Webserver

Aufgrund der neuen Funktionalität muss die Schnittstelle des Webservers erweitert werden. Dabei werden die Verbindungen zum Synchronisieren des Kampfzustandes solange gehalten bis der Kampf vorbei ist. Die Synchronisationsnachrichten werden dann als Datenstrom zur Verfügung gestellt. Auch die Anmeldung zum Matchmaking funktioniert nach diesem Prinzip. Folgende Pfade werden der Schnittstelle hinzugefügt:

Anmeldung am Matchmaking
URI: /findMatch
Methode: GET
Authentifiziert: Ja
Parameter:

- lat – Breitengrad der Spielerposition
- long – Längengrad der Spielerposition

Rückgabe:

- Ticket für den Kampf
- Name des Gegnerischen Spielers

Synchronisation des Zustands des Kampfes
URI: /getSynct
Methode: GET
Authentifiziert: Implizit durch Ticket
Parameter:

- Ticket – Durch den Matchmakingserver erstelltes Ticket

Rückgabe:

- Änderungen des Kampfzustands als Stream
- Ergebnis des Kampfes

Übermittlung einer Aktion eines Spielers
URI: /setAction
Methode: POST
Authentifiziert: Implizit durch Ticket
Parameter:

- Ticket – Durch den Matchmakingserver erstelltes Ticket
- Action – Klassifikation der Aktion (z.B. Attacke ausgewählt)
- ActionId – Id der Aktion in der entsprechenden Klassifikation

Übermittlung der Bereitschaft eines Spielers
URI: /ready
Methode: POST
Authentifiziert: Implizit durch Ticket
Parameter:

- Ticket – Durch den Matchmakingserver erstelltes Ticket

Übermittlung der Liste der für den Kampf ausgewählten Doodlings
URI: /doodlingList
Methode: POST
Authentifiziert: Implizit durch Ticket
Parameter:

- Ticket – Durch den Matchmakingserver erstelltes Ticket
- DoodlingList – Semikolonseparierte Liste der Doodlingidentifikationen

6.2.1.4 Erweiterung Client

Um den Spielern die neue Funktionalität zur Verfügung zu stellen, muss der Client erweitert werden. Es sollen 4 neue Szenen erstellt werden:

- Matchmaking
- Auswahl
- Kampf
- Ergebnis

Die Matchmakingszene soll über eine Schaltfläche von der Überblicksszene erreichbar sein. Sie zeigt dem Spieler den Fortschritt beim Matchmaking an. Nachdem das Matchmaking erfolgreich ist, soll automatisch die Auswahlszene geladen werden, die dem Spieler die Auswahl seiner Doodlings für den Kampf ermöglicht. Nach einer Bestätigung der Fertigstellung der Auswahl durch alle Spieler, die am Kampf teilnehmen, soll die Kampfszene geladen werden. Diese soll den Kampf darstellen und Schaltflächen zur Auswahl der nächsten Aktion bieten. Außerdem soll sie die aktuell aktiven Doodlings und deren Werte anzeigen. Ein Mock der Kampfansicht ist in Abbildung 31 dargestellt. Nachdem ein Kampf beendet wurde, soll automatisch die Ergebnisszene geladen werden, auf der das Ergebnis des Kampfes und besonders die Verteilung der Erfahrungspunkte auf die Doodlings angezeigt wird. Diese kann der Spieler über eine Schaltfläche verlassen, woraufhin die Überblicksszene geladen wird.
Neben der Anzeige sind die Szenen ebenfalls für die Übermittlung der Benutzereingaben an den Server sowie die Verarbeitung der Antworten des Servers verantwortlich. Hierbei soll, wie bei den bisherigen Anfragen auch, ebenfalls das HTTP-Protokoll verwendet werden.

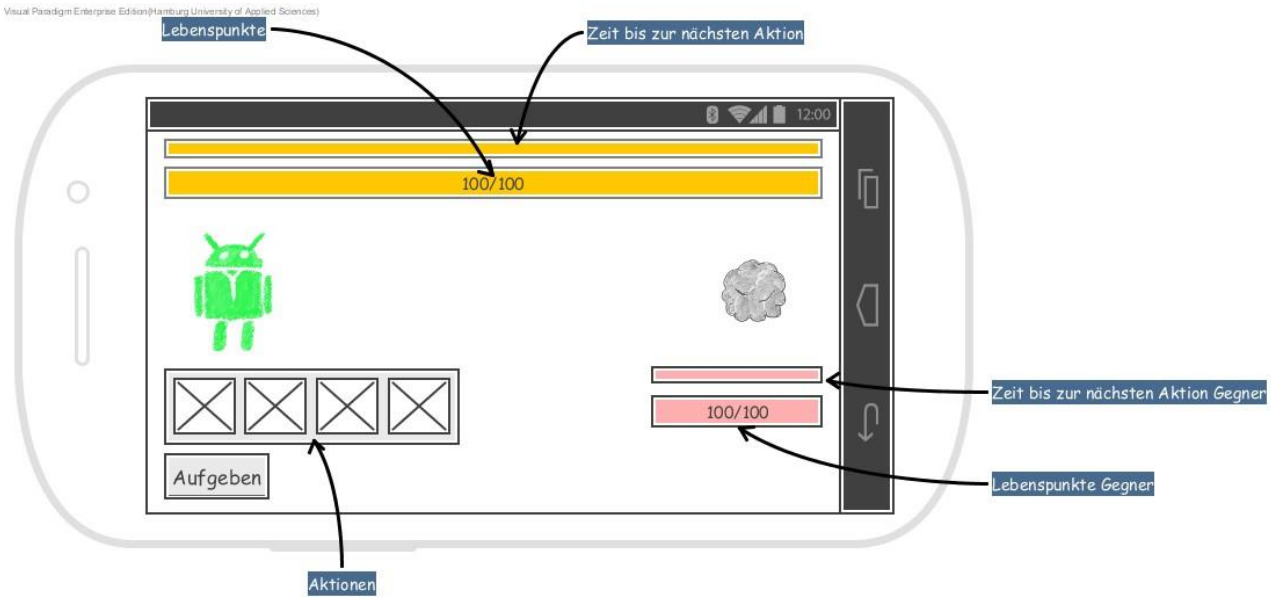

Abbildung 31 Kampfanzeige

6.2.2 Erstellen von Doodlingtypen durch Spieler

Die Auswertung der Spielerumfrage des letzten Inkrements ergab den Wunsch nach der Umsetzung der funktionalen Anforderung FR8. Diese wird in diesem Kapitel entworfen.
Für die Entwicklung dieser Funktionalität muss die Doodling-Verwaltung Komponente angepasst werden, da neben dem Abrufen von Doodlings und Doodlingtypen diese jetzt auch erzeugt werden müssen. Zusätzlich muss die Schnittstelle des Webservers und der Clients angepasst werden.

6.2.2.1 Beschreibung der Spielmechanik

Das Erstellen eines Doodlingtypen soll dem Spieler eine kreative Möglichkeit bieten, das Spiel mitzugestalten. Diese Funktionalität soll laut Beschreibung das Vorhandensein von Materialien erfordern. Materialien kann der Spieler sammeln. Dies funktioniert äquivalent zu dem Fangen der Doodlings. Dabei sind die Materialien jedoch statisch und bewegen sich nicht. Für die Erzeugung eines Doodlingtyps benötigt der Spieler zwei Materialien, nämlich Papier und Stifte. Nachdem der Spieler diese beiden Materialien gesammelt hat, kann er einen Doodlingtyp erzeugen, indem er die Werte für die Attribute der Entität „DoodlingType“ (siehe Abbildung 13) festlegt und ein Bild auswählt. Dabei dürfen die Summen der Attribute einen konfigurierbaren Wert nicht überschreiten, damit alle Doodlingtypen ungefähr gleich stark sind. Nachdem der Spieler die Erstellung abgeschlossen hat, wird das Doodling zunächst zur Überprüfung freigegeben und dann in das Spiel eingefügt. Die Überprüfung ist notwendig, damit keine unpassenden oder beleidigenden Inhalte in das Spiel überführt werden. Die Überprüfung soll zunächst von einem Administrator durchgeführt werden, da eine technische Erkennung der Merkmale zu komplex für den Umfang dieser Arbeit wäre.

6.2.2.2 Erweiterung Client

Im Client muss eine neue Szene zur Erstellung des Doodlingtypen hinzugefügt werden. Diese soll über eine Schaltfläche auf der Überblicksszene erreichbar sein. Ein Mock der Anzeigeelemente der Szene ist in Abbildung 32 dargestellt. Dabei soll der Spieler das Bild über einen Dialog vom internen Speicher seines Smartphones auswählen können oder selbst ein Bild innerhalb des Spiels zeichnen können. Der Wert für Stärke, Intelligenz und Beweglichkeit in der „Start“-Spalte, darf nicht kleiner als 0 sein, damit ein Attribut nicht einen Wert bis zu unendlich zu Lasten eines anderen Wertes annehmen kann. Nachdem der Benutzer die „Fertig“-Schaltfläche betätigt hat, werden die Daten an den Server übertragen und eine Erfolgs- bzw. Fehlermeldung angezeigt. Dabei werden die Bilddaten im PNG-Format übertragen.

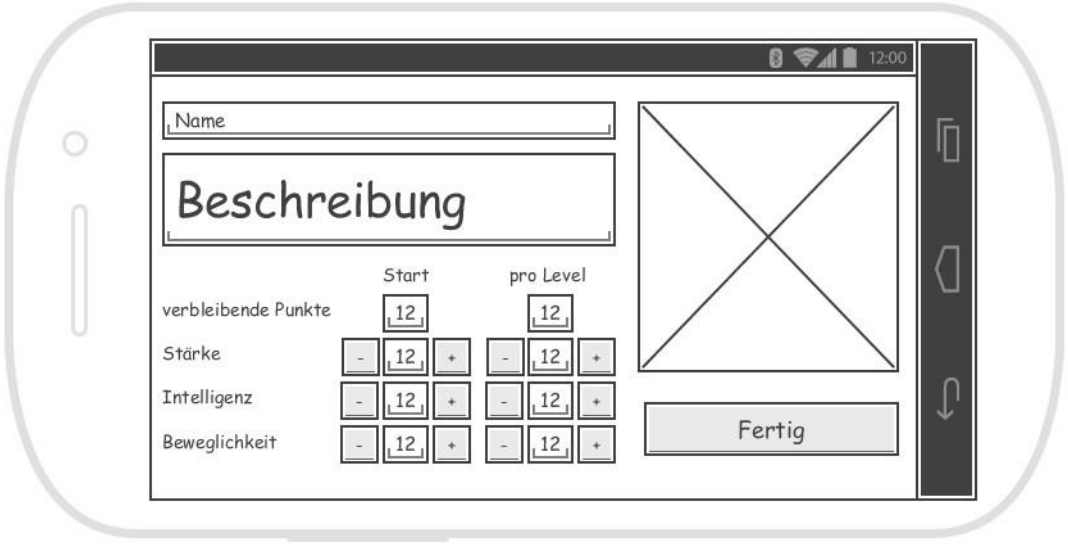

Abbildung 32 Szene für das Erstellen von Doodlingtypen

6.2.2.3 Erweiterung Doodling-Verwaltung

Für die Verarbeitung und Erzeugung der neuen Doodlingtypen ist die Doodling-Verwaltung Komponente verantwortlich. Sie überprüft zunächst, ob der neue Doodlingtyp valide ist und erzeugt danach eine neue DoodlingType-Entität. Die DoodlingType-Entität wird um ein Flag erweitert, welches anzeigt, ob diese bereits von einem Administrator aktiviert wurde. Für die restliche Funktionalität der Doodling-Management Komponente gilt, dass nur solche Doodlingtypen verarbeitet werden, bei denen dieses Flag gesetzt wurde.
Neben der Funktionalität für die Erzeugung müssen auch Funktionalitäten für die Freigabe und das Entfernen durch den Administrator implementiert werden.

6.2.2.4 Erweiterung Webserver

Für die Funktionalität muss die Schnittstelle des Webservers lediglich um eine weitere Funktion erweitert werden:

Erzeugung eines neuen Doodlingtyps
URI:/createDoodlingType
Methode: POST
Authentifiziert: Ja
Parameter:

- doodlingTypeDTO – Datenkapselung für neuen Doodlingtyp
- image – Anzeigebild für den neuen Doodlingtyp

Rückgabe

- „Ok“ oder Fehlernachricht

6.2.3 Umsetzung der Verbesserungsvorschläge

Die von den Probanden übermittelten Fehler und Verbesserungsvorschläge betreffen allesamt die Benutzungsoberfläche des Clients. Die Änderungen für die Umsetzung sind marginal, da viele Änderungen durch Funktionen der Unity-Engine abgedeckt werden. Für das Speichern des Nutzernamens werden die Playersettings verwendet. Diese speichern Daten auf dem Clientdevice persistent. Für das Spielen des Spiels im Hochformat werden die UI Elemente mit sogenannten Verankerungen ausgestattet, die diese unabhängig vom Format in der gleichen Bildschirmregion darstellen. Für das Mitdrehen des gesamten Bildschirms wird das Script, was bisher für das drehen der „Kompassnadel“ verantwortlich ist, einfach dem Kameraobjekt, welches für die Anzeige der Scene verantwortlich ist, zugeordnet.
Für die Verbesserung der Fehleranzeige beim Login werden die vom Server übermittelten HTTP-Statuscodes ausgewertet und eine dementsprechende Nachricht ausgegeben.
Zuletzt sollen innerhalb der Kompassszene die einzelnen Doodlingtypen unterscheidbar sein. Hierfür werden anstatt des Standardbildes jetzt die entsprechenden Bilder der Doodlingtypen vom Server geladen. Dabei wird genau wie beim Laden der Bilder auf der Übersichtsseite vorgegangen (siehe 5.1.4.7).

6.3 Umsetzung

Im folgenden Abschnitt wird die Umsetzung des zweiten Inkrements beschrieben. Zunächst wird die eingesetzte Technologie und die Struktur der Umsetzung beschrieben. Danach wird ein Einblick in die grafische Benutzungsoberfläche gegeben und die Unterschiede im Vergleich zum Entwurf herausgestellt.
Neben den bereits eingesetzten Bibliotheken wurden zwei weitere für die Entwicklung des zweiten Inkrements eingesetzt:

- Mobile Paint[13]
- C3P0[14]

Mobile Paint bietet Scripts und Bausteine zum Erzeugen eines einfachen Malprogramms in Unity. Dies wurde für die Umsetzung der Szene zur Erstellung neuer Doodlingtypen verwendet. C3P0 ist eine Connection- und Statementpooling Bibliothek, welche für die nebenläufige Ausführung von JDBC-Statements verwendet wird. Im Fall von Doodlings wird C3P0 als Connectionpool für Hibernate verwendet, da während der Entwicklung ein Fehler bei der Verwendung des in Hibernate integrierten Connectionpools aufgetreten ist.
Auch die Projektstruktur der neuen Serverkomponenten wurde aus dem ersten Inkrement übernommen. Auch hier bestehen die Abhängigkeiten zu den Implementierungsprojekten („doodling“ und „player“) nur aufgrund der Initialisierung. Nachdem die einzelnen Services initialisiert wurden, wird nur auf die in den API Projekten deklarierten Schnittstellen zugegriffen. Der Abhängigkeitsbaum der beiden neuen Serverkomponenten ist in Abbildung 33 und Abbildung 34 dargestellt.

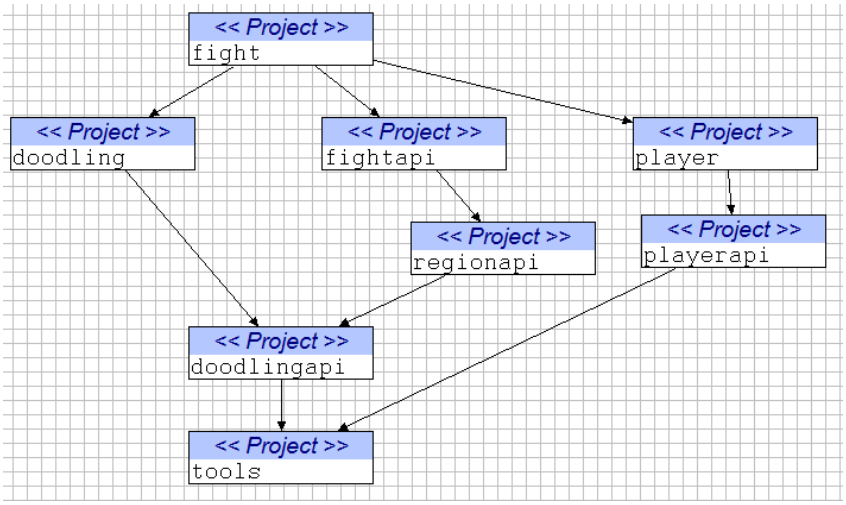

Abbildung 33 Abhängigkeitsbaum Kampfserver

[13] https://www.assetstore.unity3d.com/en/#!/content/19803
[14] https://github.com/swaldman/c3p0

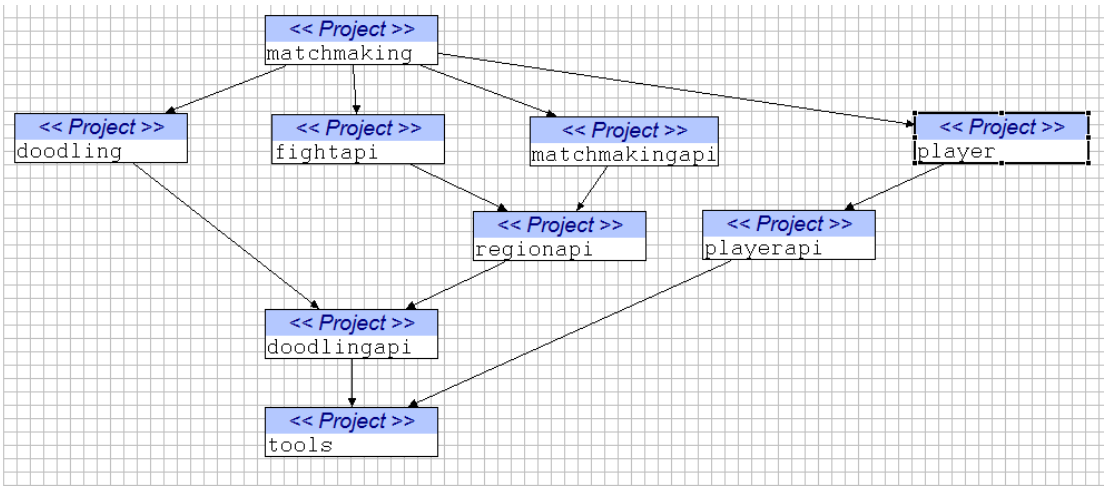

Abbildung 34 Abhängigkeitsbaum Matchmakingserver

6.3.1 Grafische Benutzungsoberfläche

Durch die neuen Funktionalitäten und die Verbesserungen aus dem vorangegangenen Inkrement, wurde die Oberfläche zum Teil stark verändert und erweitert. Aufgrund der von den Probanden gemeldeten Verbesserungsvorschläge der Kompassszene wurde diese grundlegend überarbeitet. Außerdem wurde eine Übersicht für die aktuelle Erfahrung der Doodlings in der Übersichtsszene hinzugefügt, da diese für die Kampf-Spielmechanik relevant ist. Einen Überblick über die Benutzungsoberfläche bieten die Abbildungen 35 bis 41.

Abbildung 35 Übersichtsszene

Abbildung 36 Kompassszene

Abbildung 37 Malszene

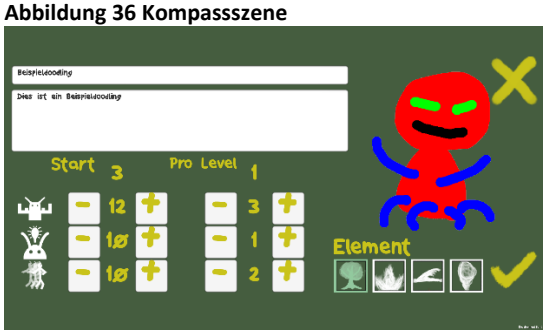

Abbildung 38 Doodlingtyperstellungsszene

Abbildung 39 Kampfauswahlszene

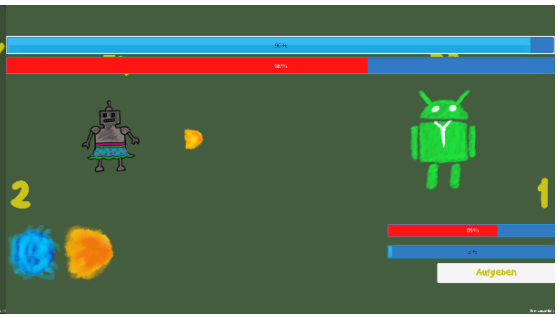

Abbildung 40 Kampfszene

Abbildung 41 Kampfergebnisszene

6.4 Test

Wie bereits im vorangegangen Inkrement, werden auch diesmal wieder sowohl automatisierte Tests als auch eine empirische Studie zur Evaluation der Umsetzung eingesetzt.

6.4.1 Automatisierte Test

Für die neu implementierten Funktionalitäten werden ebenfalls anwendungsfallbasierte Whitebox-Tests geschrieben. Dabei werden sowohl Komponententests als auch Integrationstests entwickelt, um die Korrektheit der Implementierung zu überprüfen. Für die Automatisierung der Tests kommen wieder die Frameworks JUnit und die Scala Test Suite zum Einsatz.
Des Weiteren wird die Performance des Kampfservers mithilfe eines Programms evaluiert, welches Kämpfe zwischen echten Spielern simuliert. Dafür wird eine einfache KI implementiert, die entsprechende Nachrichten an den Kampfserver schickt.

6.4.2 Manuelle Tests

Äquivalent zum ersten Inkrement wird auch für die Bewertung dieses Inkrements eine empirische Studie durchgeführt. Diese basiert auf der Studie des ersten Inkrements und erweitert diese um einige Fragen bezüglich der neuen Spielmechaniken (siehe Anhang Abbildung 53 und Abbildung 54). Für den Test werden 30 Probanden ausgewählt, die jeweils

die für dieses Inkrement relevanten Anwendungsfälle durchführen sollen. Der Test wird als Blackbox-Test durchgeführt.

6.5 Auswertung

Im Folgenden wird die Umsetzung des zweiten Inkrements anhand der Tests analysiert und bewertet. Zunächst wird die technische Seite der automatisierten Tests beleuchtet, danach die empirische Studie ausgewertet.

6.5.1 Auswertung der automatisierten Tests

Wie bereits im ersten Inkrement wurden auch diesmal die Testfälle auf ihre Anweisungsüberdeckung analysiert. Tabelle 10 zeigt die ermittelten Werte für das zweite Inkrement. Dabei fällt auf, dass im Zuge der Weiterentwicklung der Doodling-Verwaltung Komponente die Anweisungsüberdeckung stark verbessert wurde. Außerdem liegen die neu entwickelten Komponenten „Kampf“ und „Matchmaking“ im akzeptablen Bereich. Der Coverage-Wert der anderen Komponenten hat sich im Vergleich zum ersten Inkrement nicht maßgeblich verändert, da nur marginale Änderungen an diesen Komponenten vorgenommen wurden.

Tabelle 10 Coverageanalyse 2. Inkrement

Komponente	Coverage in %
Applikation	82,5
Kampf	82,2
Matchmaking	91,1
Doodling-Verwaltung	78,3
Player	77,5
Region	75,1
Tools	95,4

Neben der Analyse der Korrektheit der Anwendung ist auch die Performance von entscheidender Bedeutung (NFR 2). Dafür ist es wichtig zu ermitteln, wie viele Kämpfe von einem Kampf-Server gleichzeitig verarbeitet werden können, um die Serverinfrastruktur für das Spiel planen zu können. Die entwickelte künstliche Intelligenz ermöglicht eine ungefähre Abschätzung des Aufwandes, der für die Verarbeitung von einem Kampf auftritt. Für die Ermittlung des Maximalaufwandes, den ein bestimmter Server verarbeiten kann, wurde die Anzahl der simulierten Kämpfe schrittweise erhöht und mithilfe des Programmes Java Visual VM[15] die Auslastung der CPU sowie der benötigte Arbeitsspeicher gemessen. Der für den Test verwendete Server hat 8 Gigabyte Arbeitsspeicher, von denen 2 Gigabyte für das Programm zur Verfügung standen und einen Quadcore Prozessor mit der Bezeichnung „AMD Ahtlon X4 860K at 3.70 GHz“. Die Ergebnisse der Messung sind in Abbildung 42 bis 47 visualisiert.

[15] https://visualvm.java.net/

Anhand der gemessenen Werte ist eindeutig zu sehen, dass die durch die nichtfunktionale Anforderung NFR1 vorgegebene Nutzerzahl von 100 Spielern, die durch den Entwicklungsrechner gleichzeitig verarbeitet werden sollen, eingehalten wird. Die gleichzeitige Verarbeitung von 100 Kämpfen benötigt in der Spitze 20% der CPU und circa 70 Megabyte Arbeitsspeicher. Die beiden weiteren Testreihen zeigen, dass sogar bis zu 10.000 Kämpfe gleichzeitig verarbeitet werden könnten. Dabei fällt auf, dass besonders das Erzeugen neuer Kämpfe für hohe Auslastungen der CPU sorgt (abzulesen an den Maxima der Auslastungen zu Beginn der Messung). In realen Situationen werden die Kämpfe allerdings fortlaufend erstellt und nicht auf einmal. Das bedeutet, dass die Last, die im Testfall auf dem Anfang der Messung liegt, real auf den Gesamtzeitraum verteilt werden würde. Deswegen ist eine Verarbeitung von 10.000 gleichzeitigen Kämpfen auf einem Server mit der oben angegebenen Konfiguration durchaus realistisch.

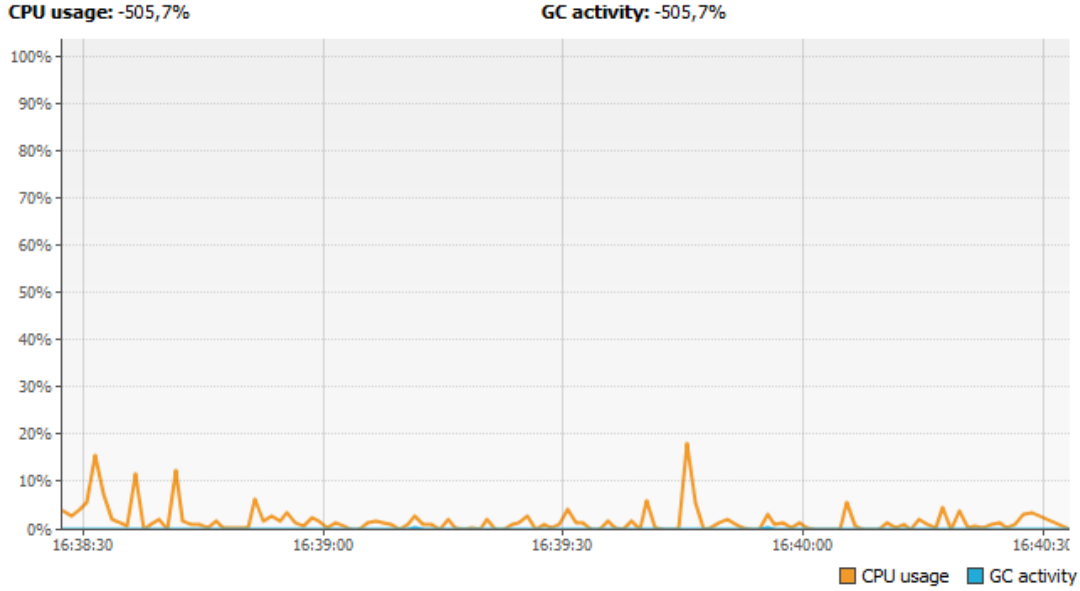

Abbildung 42 CPU Auslastung 100 Kämpfe

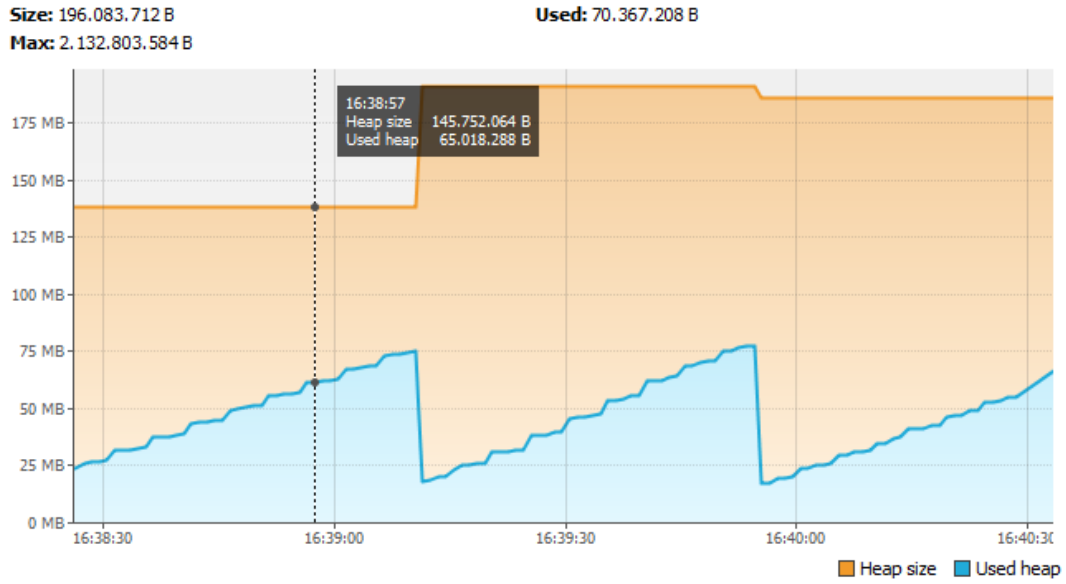

Abbildung 43 Arbeitsspeicher 100 Kämpfe

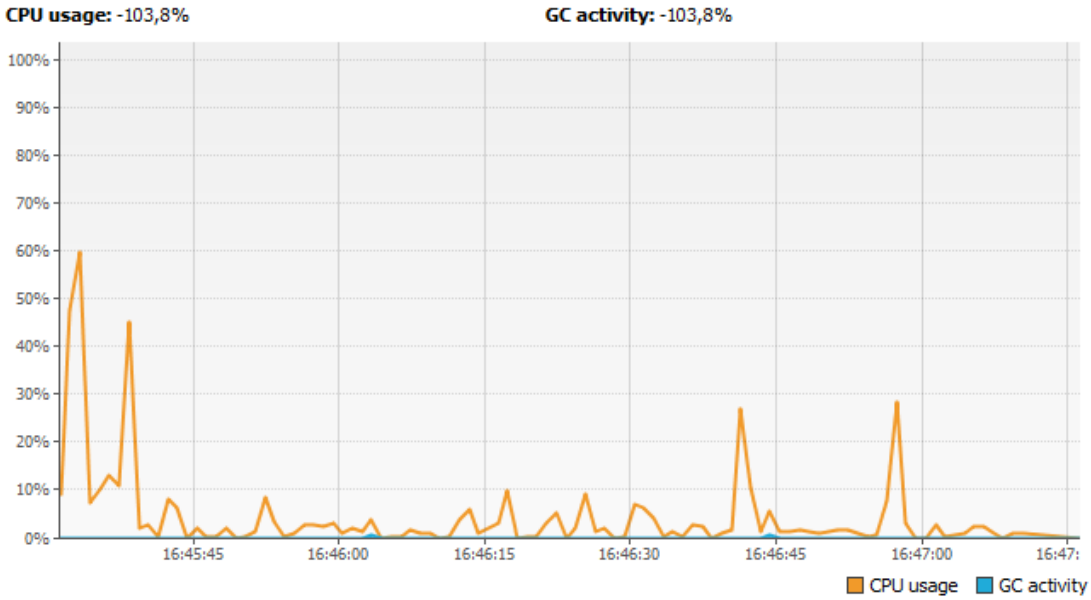

Abbildung 44 CPU-Auslastung 1.000 Kämpfe

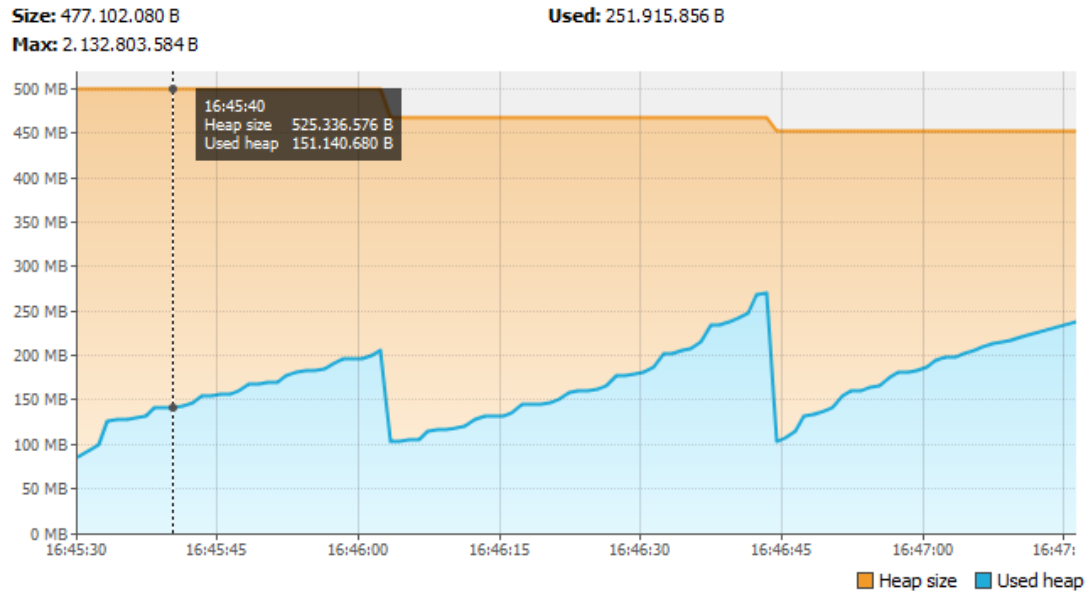

Abbildung 45 Arbeitsspeicher 1.000 Kämpfe

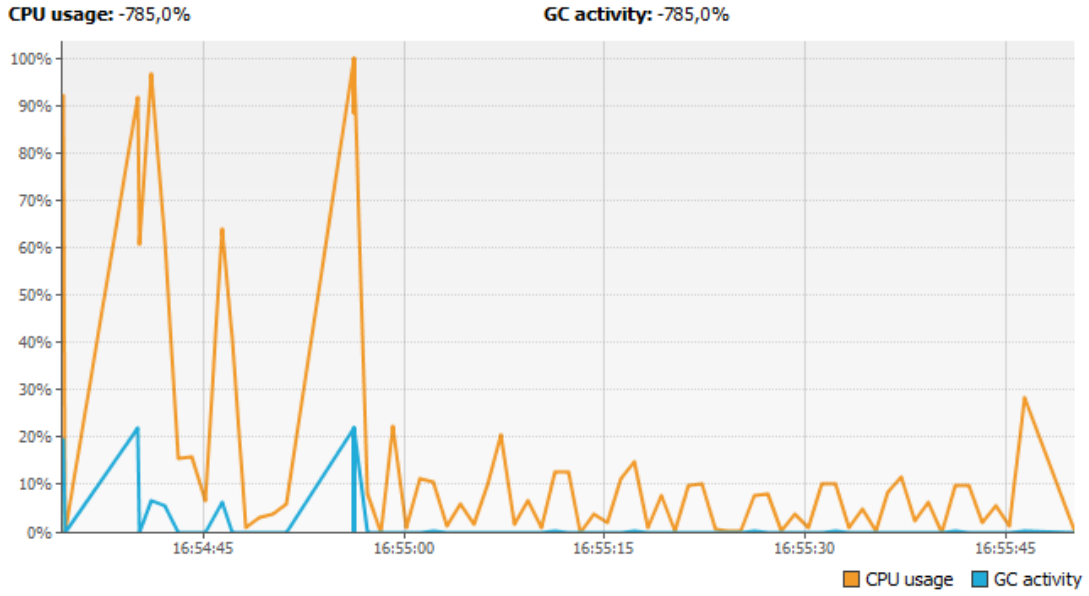

Abbildung 46 CPU Auslastung 10.000 Kämpfe

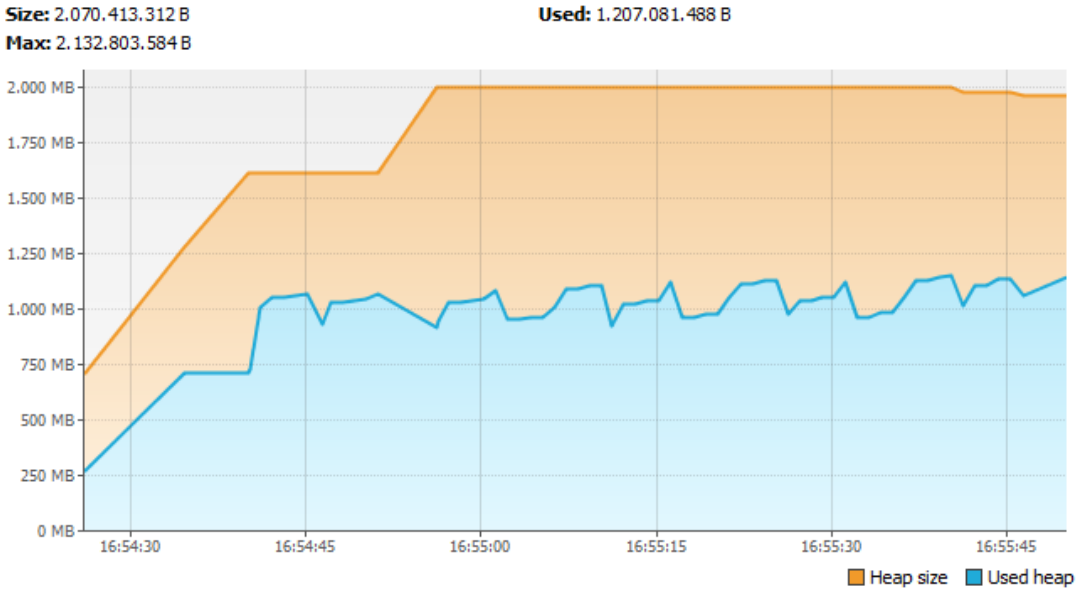

Abbildung 47 Arbeitsspeicher 10.000 Kämpfe

6.5.2 Auswertung der manuellen Tests

Auch nach dem zweiten Inkrement wurde eine empirische Studie durchgeführt, um die Umsetzung zu überprüfen und zu bewerten. Von den 30 ausgewählten Probanden haben 15 die Umfrage abgeschlossen. Die gesamte Umfrage sowie alle Ergebnisse sind im Anhang dieser Arbeit zu finden. Wie bereits in der ersten Studie ist eine geschlechtsneutrale Überprüfung der Ergebnisse nicht möglich, da im Verhältnis auch dieses mal 80% der Probanden männlich waren. Die Spielbewertung anhand der Kriterien Spaßfaktor, Innovativität, Übersichtlichkeit und Schwierigkeit ist nahezu konstant im Vergleich zur ersten Umfrage (siehe Abbildung 48). Dabei sind besonders die Innovativität und der Spaßfaktor im akzeptablen Bereich für die Weiterentwicklung des Spiels. Zusätzlich wurde nach dem Kriterium Kreativität gefragt, da durch die Spielmechanik zur Erstellung neuer Doodlings ein Feature hinzugekommen ist, was besonders die Kreativität der Spieler fordert. Da auch dieses Kriterium als hoch bis sehr hoch bewertet wurde, ist die Intention wohl von den Spielern verstanden und angenommen worden. Dies wird auch dadurch belegt, dass 93,33% der Spieler angaben, es interessant zu finden, aktiv Spielinhalte mitgestalten zu können.
Da im Vergleich zum ersten Inkrement in dieser Version mehr als eine Spielmechanik zur Verfügung stehen, wurde auch der Spaßfaktor der einzelnen Spielmechaniken erfragt (siehe Abbildung 49). Das Ergebnis zeigt, dass der Spaß, den die Spieler während des Kampfes empfunden haben, deutlich gegenüber den anderen beiden Spielmechaniken abfällt. Dies könnte zum einen daran liegen, dass in diesem Bereich relativ viele Fehler aufgetreten sind (siehe Tabelle 11), zum anderen daran, dass das Kämpfen, im Vergleich zu anderen Spielen, relativ wenig Abwechslung bietet. Dies belegt folgender Verbesserungsvorschlag:

> *„Im Kampf könnten nicht nur Gegenstände eingebunden werden, sondern auch so etwas wie Magie, welche durch Level / Gegenstände verbessert werden könnte. Dies könnte jedoch auch später in das Spiel implementiert werden und ist für die grundlegende Spielmechanik kein elementares Kriterium, sondern optional, um die Spieltiefe / den Umfang zu vergrößern. Hierbei könnte man sich an Kämpfen von rundenbasierten Rollenspielen orientieren (z.B. Final Fantasy/ Persona / Trails in the Sky etc.).“*

Innerhalb der nächsten Ausbaustufe sollten also die Kämpfe interessanter und vielseitiger gestaltet und die dort aufgetretenen Fehler analysiert und beseitigt werden.
Neben den beim Kämpfen aufgetretenen Fehlern ist außerdem innerhalb der Kompassszene ein Fehler aufgetreten. Es wurden keine Doodlings auf dem Radar geladen (siehe Tabelle 11). Da dieser Fehler allerdings nur bei einem Probanden aufgetreten ist, könnte es sich dabei auch um eine langsame oder nicht vorhandene mobile Internetverbindung gehandelt haben; auch die Zeitspanne, die der Szenenwechsel gedauert haben soll, ist untypisch und wurde ansonsten von keinem Probanden gemeldet. Auch dieser Fehler benötigt zunächst eine genauere Analyse, die in der nächsten Ausbaustufe durchgeführt werden sollte. Dennoch zeigt die Auswertung der einzelnen Spielinhalte auch, dass die beiden anderen Spielmechaniken als sehr spaßig eingestuft werden. Dies verdeutlicht, dass sowohl das location-based gaming als auch das Generieren von Spielinhalten durch Spieler durchaus Potential für ein vom Spieler gewolltes Spielerlebnis haben. Da das Spiel zusätzlich als sehr innovativ bewertet wird, ist davon auszugehen, dass die Spieler ein derartiges Spiel vorher noch nicht gespielt haben, es aber dennoch überzeugen konnte.

	sehr hoch (1)		hoch (2)		mittel (3)		niedrig (4)		sehr niedrig (5)		keine Angabe (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Spaßfaktor	3x	20,00	9x	60,00	2x	13,33	1x	6,67	-	-	-	2,07	0,80
Innovativität	6x	40,00	7x	46,67	2x	13,33	-	-	-	-	-	1,73	0,70
Übersichtlichkeit	3x	20,00	5x	33,33	6x	40,00	-	-	1x	6,67	-	2,40	1,06
Schwierigkeit	-	-	2x	13,33	7x	46,67	5x	33,33	-	-	1x	3,21	0,70
Kreativität	6x	40,00	6x	40,00	3x	20,00	-	-	-	-	-	1,80	0,77

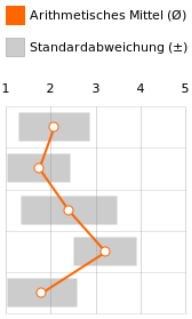

Abbildung 48 Auswertung Spielbewertung

	sehr hoch (1)		hoch (2)		mittel (3)		niedrig (4)		sehr niedrig (5)		keine Angabe (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Fangen der Doodlings	5x	33,33	7x	46,67	1x	6,67	1x	6,67	-	-	1x	1,86	0,86
Kampf	1x	6,67	2x	13,33	7x	46,67	3x	20,00	-	-	2x	2,92	0,86
Erstellen neuer Doodli...	5x	33,33	4x	26,67	3x	20,00	-	-	-	-	3x	1,83	0,83

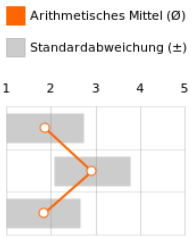

Abbildung 49 Auswertung Spielmechanikbewertung

Tabelle 11 Gemeldete Fehler

- Nach dem Fang des neuen Doodlings hing das spiel im Kampfmodus bei "warte auf andere Spieler".
- Auf der Suche nach dem Doodling war die Position des Doodlings relativ ungenau, weil der Maßstab des Radars relativ groß ist.
- Nach mehreren Spiel starten war nie ein Doodling in meiner direkten nähe. Also in weniger als <50 m Entfernung.
- Obwohl ich keine Doodlings hatte konnte ich Gegner zum Kampf heraus fordern. Allerdings befand ich mich dann in einer Endlos Wartezone, von welcher ich aber auch zurück zum Hauptmenü wechseln konnte.
- Das Spiel ist beim kämpfen abgestützt
- Wenn man vor dem Kampf nur einen Doodling auswählt, hängt das Spiel bei “warte auf andere Spieler“.
- Es wurden keine Doodlings auf dem Radar angezeigt, beim klicken des "zurück" button auf dem radar screen hat es 7 sekunden gedauert um zurück ins menü zu kommen - dadurch war man versucht mehrfach zu drücken und wurde dadurch sofort wieder zurück auf den Radar screen geleitet (vielleicht die position des "play" buttons auf dem Menü screen nicht gleich wie den "zurück" Button)
- Ein Computer Gegner wurde auch nach 3 Minuten nicht geladen
- Der erstellte Doodling steht beim Kampf nicht zur Verfügung. Muss das so?
- Nach 60 Sekunden wurde ein automatischer Kampf initialisiert, beim zweiten Mal hat dies jedoch nicht mehr funktioniert und der Timer lief weiter (vermutlich nur für die Testphase von Bedeutung).

Des Weiteren sind mehrere Fehler in der Nutzbarkeit aufgetreten. Von einem Probanden wurde folgender Fehler gemeldet:

> *„Auf der Suche nach dem Doodling war die Position des Doodlings relativ ungenau, weil der Maßstab des Radars relativ groß ist.“*

Das Problem bei der Behandlung dieses Fehlers ist, dass bei Verringerung des Maßstabs des Radars (bzw. Kompasses) auch weniger Doodlings angezeigt werden würden. Dies könnte

dazu führen, dass vor allem in weniger bevölkerten Gebieten überhaupt keine Doodlings auf dem Kompass angezeigt werden würden. Ein Kompromiss wäre, den Kompass zoombar zu machen und so dem Spieler die Entscheidung zu überlassen. Neben diesem Fehler sind auch zwei weitere Fehler aufgetreten, welche die Nutzbarkeit des Spiels betreffen:

> *„Der erstellte Doodling steht beim Kampf nicht zur Verfügung. Muss das so?“*

> *„Nach mehreren Spiel starten war nie ein Doodling in meiner direkten nähe. Also in weniger als <50 m Entfernung.“*

An beiden Fehlermeldungen ist zu erkennen, dass der betreffende Proband die Spielmechanik nicht komplett verstanden hat. Betrachtet man dazu die Verbesserungsvorschläge, wird deutlich, dass der Spieler innerhalb des Spiels zu wenig eingewiesen wird:

> *„Es fehlt ein Tutorial, das Spiel sollte starten und einem in einer kleinen interaktiven Anleitung das Spielziel und die Mechaniken erläutern.“*

> *„Es sollte eine Art Tutorial für das Spiel geben, dabei würden Info-Fenster bereits genügen, welche man durch nicht mehr anzeigen dauerhaft verbergen kann (sollte in den Einstellungen zurückgesetzt bzw. neu gestartet werden können). Eines sollte auf dem "Radarbildschirm" die grundlegende Spielmechanik des Fangens der Doodlings erläutern (was bedeutet der Pfeil, in welche Richtung bewege ich mich als Spieler, wie fange ich Doodlings?), vor allem im Kampf ist jedoch ohne einen Infobildschirm mit einer Erläuterung der Grafiken (z.B. was machen Feuer / Wasser?) nicht unmittelbar klar, was zu tun ist - die Einbindung von Erläuterungen würde den Einstieg in Doodlings erleichtern und könnte so neue potentielle Spieler eher dazu bewegen, sich mit dem Spiel zu beschäftigen, bevor die Wahl auf ein anderes Spiel fällt.“*

> *„Eine Beschreibung der Attribute wäre Hilfreich.“*

In der nicht funktionalen Anforderung an die Nutzbarkeit (NFR3) wird definiert, dass der Spieler das Spiel ohne weitere Anleitungen verstehen soll. Diese Anforderung wird scheinbar im zweiten Inkrement nicht erfüllt und muss in der nächsten Ausbaustufe umgesetzt werden. Innerhalb der Verbesserungsvorschläge wird deutlich, dass sich vor allem ein optionales Tutorial sowie Tooltips, gewünscht werden. Diese sollten dem Spiel hinzugefügt werden.
Neben den akuten Fehlern und Verbesserungsvorschlägen wurden auch aufwendige Vorschläge für zusätzliche oder erweiterte Spielmechaniken gemacht, die für die Aufnahme in das Spiel zu einem späteren Zeitpunkt diskutiert werden müssen. So könnte man die Kompassszene durch eine Szene ersetzen, die das Bild der Kamera zeigt und den Kompass über dieses legen, so dass der Blick nicht zwischen Umgebung und Handy wechseln muss. Außerdem wurde ein Vorschlag gemacht, dass bestimmte Doodlings nur in bestimmten

Gebieten auftauchen (zum Beispiel Pflanzendoodlings im Wald). Zuletzt wurde der Wunsch nach der Anzeige der Kartendaten auf dem Kompass geäußert. Alle diese Verbesserungsvorschläge sind aufwendig in der Umsetzung und müssen auch zunächst auf die Zustimmung der Zielgruppe und Machbarkeit überprüft werden und sollten daher nicht für eine Umsetzung in der nächsten Ausbaustufe geplant werden.
Neben der Bewertung des Ist-Zustands sollten die Probanden auch ihre Meinung zu den geplanten nächsten Spielmechaniken abgeben. Dabei sollten die Probanden die Wichtigkeit der Umsetzung der Spielmechaniken mit 1 (am wichtigsten) bis 7 (am unwichtigsten) bewerten. Hierbei wurde durchschnittlich die Mechanik „gemeinsame Kämpfe gegen besonders starke Doodlings“ als am wichtigsten bewertet und sollte als nächstes umgesetzt werden. Interessant ist ebenfalls die Bewertung von „Verbesserung des Doodlings durch vom Spieler ausgeführte sportliche Betätigung (Einbindung von z.B. GoogleFit)“. Diese wird entweder als wichtig oder besonders unwichtig bewertet. Dabei fällt auf, dass die Probanden, die diese Mechanik als besonders unwichtig bewertet haben auch bei der Bewertung des Spielspaßes eine wesentlich geringere Einschätzung hatten als der Durchschnitt. Dies könnte bedeuten, dass dieses Feature für die Zielgruppe dennoch von gehobenem Interesse ist und eine mögliche Umsetzung weiter geprüft werden sollte.

	1.		2.		3.		4.		5.		6.		7.			
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Ø	±
Erstellen von Aufgabe...	2x	13,33	3x	20,00	3x	20,00	4x	26,67	-	-	3x	20,00	-	-	3,40	1,68
Gemeinsame Kämpfe ...	6x	40,00	2x	13,33	1x	6,67	3x	20,00	1x	6,67	1x	6,67	1x	6,67	2,87	2,03
Gegenstände zum leic...	2x	13,33	-	-	3x	20,00	3x	20,00	4x	26,67	1x	6,67	2x	13,33	4,20	1,82
Doodlings als Tamagot...	1x	6,67	1x	6,67	2x	13,33	1x	6,67	1x	6,67	4x	26,67	5x	33,33	5,13	2,03
Gegenstände zur Unte...	1x	6,67	3x	20,00	2x	13,33	1x	6,67	5x	33,33	2x	13,33	1x	6,67	4,07	1,79
Automatisch generiert...	1x	6,67	-	-	3x	20,00	3x	20,00	2x	13,33	4x	26,67	2x	13,33	4,67	1,72
Verbesserung des Doo...	2x	13,33	6x	40,00	1x	6,67	-	-	2x	13,33	-	-	4x	26,67	3,67	2,38

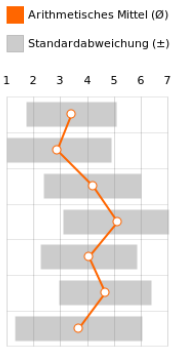

Abbildung 50 Bewertung neue Features

Neben den funktionalen Tests wurden auch diesmal wieder die nicht funktionalen Anforderungen an den Ressourcenverbrauch (NFR4) durch zwei Fragen überprüft. In Tabelle 12 sind die Daten zum Akkuverbrauch und der Datennutzung der einzelnen Probanden aufgelistet. Während die durchschnittlichen Werte weit unter den durch NFR4 vorgegebenen Werten liegen, sind sie im Vergleich zu Messung des ersten Inkrements signifikant gestiegen (Akkuverbrauch etwa um 3%, Datennutzung etwa um 0,5 MB). Dies könnte sich durch die dauerhafte Netzwerkverbindung während des Kampfes und das Übermitteln der gemalten Doodlings erklären. Da aber auch die Einzelwerte bis auf einen Ausreißer alle innerhalb der Anforderungen liegen, ist kein akuter Handlungsbedarf zur Verbesserung gegeben. Der Proband, dessen Akkuverbrauch mit 28% pro Stunde angegeben ist, gab an, eine halbe Stunde gespielt und dabei 14 % Akku verbraucht zu haben. Bei dieser kurzen Spieldauer wirken sich Schätzungsfehler des Probanden, was die genaue Spieldauer angeht, besonders

stark aus. Dies könnte ein Grund für die starke Abweichung im Vergleich zu den anderen Probanden sein.

Tabelle 12 Messung Akkuverbrauch und Datennutzung

	Akkuverbrauch in % pro Stunde	Datennutzung in MB pro Stunde
	7.33	0.4
	8	0.83
	6.67	2.67
	7	1.32
	3	1.06
	k.A.	0.45
	k.A.	k.A.
	8.33	0.31
	10	1.27
	10	1.27
	20	0.45
	k.A.	k.A.
	10	3
	28	1.6
	16	0.92
Durchschnitt	11.19	1.2

7 Fazit und Ausblick

In diesem Kapitel wird zunächst eine methodische Abstraktion durchgeführt, die die eingesetzten Methoden evaluiert. Danach wird ein abschließendes Fazit gezogen und ein kurzer Ausblick in die weiteren Entwicklungsschritte gewährt.

7.1 Methodische Abstraktion

Das inkrementelle Vorgehen war von Vorteil, da damit die richtigen Entscheidungen bezüglich der Weiterentwicklung des Projektes getroffen werden konnten. Dies wurde aufgrund der Bewertung am Ende jedes Inkrements gewährleistet. Dadurch wurde es möglich, Erfahrungen des vorangegangen Inkrements in das neue einfließen zu lassen und somit zu verbessern. Zusätzlich konnten durch die Stückelung der Gesamtentwicklung die Teilentwicklungen besser und klarer geplant werden. Allerdings wird durch die Aufteilung in Inkremente die Schwierigkeit erhöht, das Gesamtsystem von Beginn an zu überblicken, da zunächst nur diejenigen Funktionalitäten detailliert entworfen werden, die für das Inkrement relevant sind. Der praktische Einsatz des Entwurfsmusters zeigt aber, dass eine von Beginn an erfolgende Berücksichtigung der Erweiterbarkeit des Systems dieses Problem zu umgehen vermag.
Weiterhin wurde das Durchführen einer empirischen Studie mit Probanden als sehr hilfreich empfunden. Dadurch konnten relevante Entscheidungen zur Weiterentwicklung des Projektes getroffen werden und Fehler gefunden werden, die durch die automatisierten Tests nicht gefunden wurden. Demgegenüber steht der große Aufwand, den das Durchführen und Auswerten der Studie mit sich bringt. Besonders das Motivieren der Probanden, den Test durchzuführen, war sehr aufwändig, da diese der Studie vermutlich keine besondere Wichtigkeit beigemessen haben. Der zusätzliche Aufwand ist aber durch die beschleunigte Entscheidungsfindung und die erhöhte Sicherheit, das richtige Spiel für die Spieler zu entwickeln, gerechtfertigt und besonders am Anfang der Entwicklung lohnenswert.

7.2 Fazit

Das Ziel dieser Arbeit war die Entwicklung des Kernsystems und die Evaluation der grundlegenden Spielidee von Doodlings. Zunächst wurde die Spielidee ausgearbeitet und die Anforderungen an das zugrundeliegende Softwaresystem analysiert. Aufgrund der Schleifenregel wurde sich für einen inkrementellen Entwicklungsansatz entschieden und die ersten beiden Inkremente entworfen und umgesetzt. Dabei beruhte die Planung des zweiten Inkrements auf der Auswertung des ersten Inkrements. Um genau das Spiel zu entwickeln,

dass dem Spieler gefällt, wurde in der Bewertung jedes Inkrements eine empirische Studie mit den Spielern durchgeführt.
Die Auswertung der empirischen Studien zeigt, dass die Spielidee sowohl als innovativ als auch als spaßig empfunden wurde. Vor allem die so erhaltenen Verbesserungsvorschläge haben die Entwicklung des Spiels stark beeinflusst und zu einem besseren Gesamterlebnis geführt. Außerdem wurde das Spielerlebnis in Bezug auf die Aufnahme neuer Spielmechaniken durch das Einbeziehen der Spieler verbessert, da die laut Studie spaßigste Spielmechanik erst durch den Wunsch der Spieler in die Planung des zweiten Inkrements aufgenommen wurde. Dennoch zeigt die Auswertung des zweiten Inkrements, dass durchaus noch die Notwendigkeit zur Verbesserung und Erweiterung von Doodlings besteht. Insgesamt wurde der Entwicklungsprozess durch die empirischen Studien vereinfacht, da diese konkret zur Entscheidungsfindung innerhalb der Planung beigetragen haben.
Eine Betrachtung der technischen Seite des Systems zeigt, dass die Geobashing Architektur mitsamt der entwickelten Erweiterungen die Anforderungen an das System erfüllt, sodass alle bis zu diesem Zeitpunkt relevanten funktionalen und nicht funktionalen Anforderungen erfüllt werden. Außerdem ist eine Wiederverwendbarkeit der einzelnen Systemkomponenten gegeben. Dies zeigt das zweite Inkrement, welches für die neuen Kampf und Matchmaking Server Systemkomponenten verwendet, die für den Applikationsserver entwickelt wurden. Besonders die auf Aktoren basierte Entwicklung hat sich ausgezahlt, da die Verteilung und Kommunikation der einzelnen Server so stark vereinfacht wurde. Dies hat zur Folge, dass Applikationsserver bei erhöhter Spieleranzahl sehr leicht für kleinere Gebiete konfiguriert und entfernt gestartet werden können, was die Anwendung wiederum skalierbar macht. Zusätzlich wurde die Entwicklung des Kampfservers durch die Aktoren stark vereinfacht, da die Nebenläufigkeit der einzelnen Kämpfe durch die Aktoren implizit gewährleistet wird.
Neben der Serverarchitektur war auch die Entscheidung für die Game Engine Unity von zentraler Bedeutung. Durch die von Unity bereitgestellten Werkzeuge und Dokumentationen wurde die Entwicklung beschleunigt, obwohl keine Vorerfahrung vorhanden war. Allerdings traten auch einige Probleme auf, zum Beispiel mit den von Unity bereitgestellten Web Klassen, da diese nicht den gesamten Funktionsumfang bereitstellten, der für die Entwicklung benötigt wurde.
Abschließend lässt sich feststellen, dass location-based gaming in Verbindung mit von Spielern generierten Inhalten, ein neuartiges Erlebnis bieten kann und somit gerade diese Verbindung dabei den Spielspaß ausmacht.

7.3 Ausblick

Insgesamt kann der Start der Entwicklung von Doodlings als Erfolg und die Weiterentwicklung aufgrund der Spielerumfragen als lohnenswert bezeichnet werden. Dennoch sind viele Anforderungen an Doodlings noch nicht umgesetzt und einige der Umsetzungen müssen überarbeitet werden. Für die weitere Entwicklung kann auf die Erkenntnisse der Auswertung des zweiten Inkrements zurückgegriffen werden. Weiterhin müssen für die Veröffentlichung des Spiels auch die in dieser Arbeit nicht behandelten Punkte wie die Monetarisierung sowie

eine Professionalisierung der Grafik und des Sounds geplant und umgesetzt werden. Für die Monetarisierung könnte beispielsweise eine Crowdfunding Kampagne gestartet werden.
Zusätzlich muss das Spiel auch technisch weiterentwickelt werden, so wurde in der bisherigen Entwicklung beispielsweise keine Priorität auf die Sicherheit der Daten gelegt. Es werden aber sicherheitsrelevante Daten wie Passwörter und auch private Daten wie Positionsdaten übertragen. Diese müssen im Produktivbetrieb vor Angriffen geschützt werden. Außerdem wurde auch die Möglichkeit des sogenannten Cheatings noch nicht betrachtet. So könnten beispielsweise Spieler falsche Positionsdaten übermitteln und so einen Vorteil gegenüber anderen Spielern erhalten. Auch dies muss vor der Veröffentlichung noch geprüft und verhindert werden.

Literatur

[1]
Adobe Systems, 2015. *Haben Sie in den letzten drei Monaten eine Augmented-Reality-App genutzt?.* [Online]
Available at: http://de.statista.com/statistik/daten/studie/425536/umfrage/nutzung-von-augmented-reality-apps-nach-laendern/
[Zugriff am 9 April 2016].

[2]
Agha, G., 1985. *Actors: A Model of Concurrent Computation in Distributed Systems.* Cambridge: MIT Press.

[3]
Balzert, H., 2011. *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb.* Ausgabe 3 Hrsg. Heidelberg: Spektrum Akademischer Verlag.

[4]
Bengel, G., 2014. *Grundkurs Verteilte Systeme.* 4. Auflage Hrsg. Mannheim: Springer Fachmedien Wiesbaden.

[5]
Cockburn, A., 2001. *Writing Effective Use Cases.* Boston(MA): Addison Wesley.

[6]
Dieber, B., Grassauer, T., Mayring, J. & Rinner, B., 2010. The Geobashing Architecture for Location-Based Mobile Massive Multiplayer Online Games. *Next Generation Mobile Applications, Services and Technologies (NGMAST), 2010 Fourth International Conference on*, 27-29 July, pp. 13-18.

[7]
Engels, G., Kremer, M. & Nötzold, T., 2012. *http://web.de.capgemini.com/.* [Online]
Available at: http://web.de.capgemini.com/doc/Quasar3/Quasar3_external_V1.1paper.pdf
[Zugriff am 6 March 2016].

[8]
Equation Research, 2011. *www.de.statista.com.* [Online] Available at: http://de.statista.com/statistik/daten/studie/202650/umfrage/wartezeit-bis-zum-verlassen-einer-mobilen-website/
[Zugriff am 3 11 2015].

[9]
GfK Enigma, 2015. *Durchschnittliche tägliche Nutzungszeit von Computer-, Konsolen-, Online-, Tablet-, Handyspielen durch Jugendliche im Jahr 2015 (in Minuten).* [Online] Available at: http://de.statista.com/statistik/daten/studie/29441/umfrage/taegliche-nutzungsdauer-von-games-pc-und-konsole-durch-jugendliche/
[Zugriff am 05 April 2016].

[10]
GfK, 2015. *Absatz von Smartwatches und Gesundheits- und Fitness-Trackern weltweit in den Jahren 2014 und 2015 (in Millionen Stück).* [Online] Available at: http://de.statista.com/statistik/daten/studie/421270/umfrage/absatz-von-smartwatches-und-gesundheits-und-fitness-trackern-weltweit/
[Zugriff am 9 April 2016].

[11]
Groundspeak, 2015. *Geocaching.com.* [Online] Available at: https://www.geocaching.com/?guest=1
[Zugriff am 15 September 2015].

[12]
Hsu, C.-c., Ling, J. & Li, Q., 2003. On the Design of Multiplayer Online Video Game Systems. *Multimedia Systems and Applications VI ITCom 2003*, 7-11 September, Band 5241.

[13]
Internet Engineering Task Force, 2011. *The WebSocket Protocol.* [Online] Available at: http://tools.ietf.org/html/rfc6455
[Zugriff am 1 April 2016].

[14]
Kampmann Walther, B., 2005. Atomic actions -- molecular experience: theory of pervasice gaming. *Computers in Entertainment (CIE) - Theoretical and Practical Computer Applications in Entertainment,* 3(3), p. 4.

[15]
Khan, A. M., Arsov, I. & Preda, M., 2010. *Adaptable client-server architecture for mobile multiplayer games.* Brüssel, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) ICST.

[16]
Kilian, T., Hass, B. H. & Walsh, G., 2008. Grundlagen des Web 2.0. In: B. H. Hass, G. Walsh & T. Kilian, Hrsg. *Web 2.0.* s.l.:Springer-Verlag Berlin Heidelberg, pp. 3-21.

[17]
Lonthoff, J. & Leiber, T., 2005. *Mobile Location Based Gaming als Wegbereiter für Location Based Services (LBS).* [Online]
Available at: http://cs.emis.de/LNI/Proceedings/Proceedings73/GI-Proceedings.73-22.pdf
[Zugriff am 22 September 2015].

[18]
Ludewig, J. & Lichter, H., 2013. *Software Engineering.* 3. Auflage Hrsg. Heidelberg: dpunkt.verlag GmbH.

[19]
Neustaedter, C., Tang, A. & Judge, T. K., 2013. Creating scalable location-based games: lessons from Geocaching. *Personal and Ubiquitous Computing*, Februar, 17(2), pp. 335-349.

[20]
Niantic Labs, 2015. *Ingress Googleplay.* [Online]
Available at: https://play.google.com/store/apps/details?id=com.nianticproject.ingress
[Zugriff am 29 September 2015].

[21]
Nintendo, 2015. *Nintendo.de.* [Online]
Available at: http://www.nintendo.de/Spiele/Game-Boy/Pokemon-Rote-Edition-266109.html
[Zugriff am 22 September 2015].

[22]
Nintendo, 2015. *Super Mario Maker überspringt die Marke von 1 Million verkauften Exemplaren.* [Online]
Available at: https://www.nintendo.de/News/2015/September/Super-Mario-Maker-uberspringt-die-Marke-von-1-Million-verkauften-Exemplaren-1054662.html
[Zugriff am 9 April 2016].

[23]
Richardson, L. & Ruby, S., 2008. *RESTful Web Services,* s.l.: O'Reilly Media.
Schell, J., 2012. *Die Kunst des Game Designs - Bessere Games konzipieren und entwickeln.* 1. Auflage Hrsg. Heidelberg: mitp.

[24]
Schlieder, C., 2014. Geogames – Gestaltungsaufgaben und geoinformatische Lösungsansätze. *Informatik-Spektrum*, Dezember, 37(6), pp. 567-574.

[25]
Schlieder, C., Kiefer, P. & Matyas, S., 2006. Geogames: Designing Location-Based Games from Classic Board Games. *Intelligent Systems, IEEE*, 2 Oktober, pp. 40 - 46.

[26]
Sjurts, I., 2015. *Gabler Wirtschaftslexikon, Stichwort: Smartphone.* [Online] Available at: http://wirtschaftslexikon.gabler.de/Archiv/569824/smartphone-v1.html [Zugriff am 31 August 2015].

[27]
Typesafe Inc., 2015. *akka.io/.* [Online] Available at: http://akka.io/ [Zugriff am 15 November 2015].

[28]
Véron, M., Marin, O. & Monnet, S., 2014. Matchmaking in multi-player on-line games: studying user traces to improve the user experience. *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*.

[29]
Vickery, G. & Wunsch-Vincent, S., 2007. *Participative Web and User-Created Content.* s.l.:OECD.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

a) Empirische Studie erstes Inkrement

Doodlings Alphatest

Seite 1

Bitte geben Sie Ihr Alter an *

Bitte geben Sie Ihr Geschlecht an *

Bitte wählen...

Bitte geben Sie an wie lange Sie Doodlings gespielt haben *

Stunden

Wie viele Doodlings haben Sie gefangen? *

- 0
- 1
- 2
- 3
- mehr als 3

Bitte geben Sie Marke und Typ Ihres Smartphones an *

Bitte bewerten Sie das Spiel anhand der nachfolgenden Kriterien *

	sehr hoch	hoch	mittel	niedrig	sehr niedrig	keine Angabe
Spaßfaktor	○	○	○	○	○	○
Innovativität	○	○	○	○	○	○
Übersichtlichkeit	○	○	○	○	○	○
Schwierigkeit	○	○	○	○	○	○

Sind während des Spielens Fehler aufgetreten?

- ja
- nein

Wenn Fehler aufgetreten sind, erläutern Sie diese bitte

Wie viel Akku hat Doodlings verbraucht?

Sie finden den Akkuverbrauch in den Einstellungen Ihres Smartphones unter Akku

%

Abbildung 51 Fragebogen 1 Seite 1

Abbildung 52 Fragebogen 1 Seite 2

Tabelle 13 Antworten Fragebogen 1

Frage	Antworten
Bitte geben Sie Ihr Alter an	Jahre Anzahl 22 1 24 1 26 2 27 1 30 3 32 2
Bitte geben Sie Ihr Geschlecht an	8 – männlich 2 – weiblich

<table>
<tr><td>Bitte geben Sie an, wie lange Sie Doodlings gespielt haben (in Std)</td><td>0,5
2
1
2
3
1
0,5
2
1
0,5</td></tr>
<tr><td>Wie viele Doodlings haben Sie gefangen?</td><td>Anzahl Doodlings Antworten
0 2
1 3
2 4
3 0
Mehr als 3 1</td></tr>
<tr><td>Bitte geben Sie Marke und Typ Ihres Smartphones an</td><td>Fairphone, Fairphone 2
Google Nexus 5
Samsung Galaxy s4
Sony Xperia T
Lg g3
Galaxy Nexus
Xperia Z5 Compact
Sony Xperia z5
Sony Xperia Z1 Compact
Gigaset qv1030</td></tr>
<tr><td>Bitte bewerten Sie das Spiel anhand der nachfolgenden Kriterien</td><td><table>
<tr><td></td><td>sehr hoch</td><td>hoch</td><td>mittel</td><td>niedrig</td><td>sehr niedrig</td></tr>
<tr><td>Spaßfaktor</td><td>2</td><td>5</td><td>3</td><td>-</td><td>-</td></tr>
<tr><td>Innovativität</td><td>4</td><td>4</td><td>2</td><td>-</td><td>-</td></tr>
<tr><td>Übersichtlichkeit</td><td>1</td><td>5</td><td>3</td><td>1</td><td>-</td></tr>
<tr><td>Schwierigkeit</td><td>-</td><td>1</td><td>6</td><td>3</td><td>-</td></tr>
</table></td></tr>
<tr><td>Sind während des Spielens Fehler aufgetreten?</td><td>5 – Ja
5 – Nein</td></tr>
</table>

Wenn Fehler aufgetreten sind, erläutern Sie diese bitte	• Die Entfernungsangaben scheinen ungenau zu sein. • Es sagt, dass mein Passwort falsch ist wenn nur die Internetverbindung fehlschlägt • Position ungenau wird häufig angezeigt, ein Fangen von Doodlings ist dann nicht möglich. Ursache wird im fehlerhaften GPS Modul des verwendeten Handy vermutet • Anscheinend umher springende doodlings. 50-100m Distanz Veränderung innerhalb von ein paar Sekunden • Beim anmelden erst falsch eingeben... danach versucht mit richtigem login anzumelder und zu bestätigen... passierte aber nix. nach neustart wieder io. vlt auch fehlermeldung einbauen bei falschem pw o uname. pfeil spitze des radars relativ schwer die richtung zu erkennen und dieser schlägt die ganze zeit aus vlt etwas korrelieren.
Wie viel Akku hat Doodlings verbraucht?	10 0 23 5 30 11 3 2 7
Wieviel Netzwerkdaten hat Doodlings gesendet und empfangen?	1,2 0,392 1,1 0,553 0 1,4 0,7 0 0

<table>
<tr><td rowspan="8">Bitte bewerten Sie, welche neuen Spielmechaniken in das Spiel übernommen werden sollen</td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr>
<tr><td>Erstellen neuer Doodlings durch Spieler</td><td>2</td><td>1</td><td>2</td><td>1</td><td>4</td><td>-</td></tr>
<tr><td>Erstellen von Aufgaben und Rätseln durch Spieler</td><td>1</td><td>1</td><td>2</td><td>2</td><td>3</td><td>1</td></tr>
<tr><td>Kämpfe zwischen Spielern mit Doodlings</td><td>5</td><td>4</td><td>-</td><td>-</td><td>1</td><td>-</td></tr>
<tr><td>Gemeinsame Kämpfe von Spielern gegen besonders starke Doodlings</td><td>-</td><td>1</td><td>5</td><td>3</td><td>1</td><td>-</td></tr>
<tr><td>Gegenstände zum leichteren Einfangen/Verbessern von Doodlings</td><td>1</td><td>2</td><td>1</td><td>3</td><td>1</td><td>2</td></tr>
<tr><td>Doodlings als Tamagotchi</td><td>1</td><td>1</td><td>-</td><td>1</td><td>-</td><td>7</td></tr>
<tr><td>Verbesserungs-vorschläge</td><td colspan="7">Hochformat, drehbare Karte
auch im Hochformat spielbar sein
Nutzername sollte speicherbar sein und nicht jedes mal neu eingegeben werden
die Doodlings je nach Rasse unterschiedlich in der Karte darstellen
Den Pfeil der Blickrichtung intuitiver machen. So das besser erkannt wird in welche Richtung man schaut. Oder (noch besser) die Umgebung drehen. So das der Pfeil immer nach oben zeigt.
Anstatt 60 Sekunden zu warten für das fangen, feedback geben oder "kämpfen".
Mehrfach den gleichen doodlings fangbar machen oder vorher diejenigen ausblenden/besonders markieren, die man bereits hat.
Bewegungsgeschwindigkeit der doodlings reduzieren.
Mehr Doodlings
Countdown beim Doodlingfangen erweckt den Eindruck als müsste man sich am gleichen Ort aufhalten</td></tr>
</table>

b) Empirische Studie zweites Inkrement

Doodlings Alphatest

Seite 1

Bitte geben Sie Ihr Alter an *

Bitte geben Sie Ihr Geschlecht an *

Bitte wählen...

Bitte geben Sie an wie lange Sie Doodlings gespielt haben *

Stunden

Wie viele Doodlings haben Sie gefangen? *

- 0
- 1
- 2
- 3
- mehr als 3

Bitte geben Sie Marke und Typ Ihres Smartphones an *

Bitte bewerten Sie das Spiel anhand der nachfolgenden Kriterien *

	sehr hoch	hoch	mittel	niedrig	sehr niedrig	keine Angabe
Spaßfaktor	○	○	○	○	○	○
Innovativität	○	○	○	○	○	○
Übersichtlichkeit	○	○	○	○	○	○
Schwierigkeit	○	○	○	○	○	○
Kreativität	○	○	○	○	○	○

Bitte bewerten Sie die einzelnen Spielmechaniken anhand ihres Spaßfaktors *

	sehr hoch	hoch	mittel	niedrig	sehr niedrig	keine Angabe
Fangen der Doodlings	○	○	○	○	○	○
Kampf	○	○	○	○	○	○
Erstellen neuer Doodlingtypen	○	○	○	○	○	○

Finden Sie die Möglichkeit aktiv Spielinhalte eines Spiels mitgestalten zu können interessant? *

- ja
- nein

Abbildung 53 Fragebogen 2 Seite 1

Sind während des Spielens Fehler aufgetreten?

- ja
- nein

Wenn Fehler aufgetreten sind, erläutern Sie diese bitte

Wie viel Akku hat Doodlings verbraucht?

Sie finden den Akkuverbrauch in den Einstellungen Ihres Smartphones unter Akku

%

Wieviel Netzwerkdaten hat Doodlings gesendet und empfangen?

Sie finden die Datennutzung in den Einstellungen Ihres Smartphones unter Datennutzung

MB

Bitte bewerten Sie, welche neuen Spielmechaniken in das Spiel übernommen werden sollen *

von 1 (am interessantesten) bis 7 (am uninteressantesten). Alle Zahlen dürfen nur einmalig vergeben werden

- Erstellen von Aufgaben und Rätseln durch Spieler
- Gemeinsame Kämpfe von Spielern gegen besonders starke Doodlings
- Gegenstände zum leichteren Einfangen/Verbessern von Doodlings
- Doodlings als Tamagotchi
- Gegenstände zur Unterstützung der Doodlings im Kampf
- Automatisch generierte Turniere an bestimmten Geopositionen
- Verbesserung des Doodlings durch vom Spieler ausgeführte sportliche Betätigung (Einbindung von z.B. GoogleFit)

Verbesserungsvorschläge

Wenn Sie sonstige Verbesserungsvorschläge oder Ideen für neue Spielmechaniken haben tragen Sie sie bitte hier ein

Abbildung 54 Fragebogen 2 Seite 2

Tabelle 14 Antworten Fragebogen 2

Frage	**Antworten**
Bitte geben Sie Ihr Alter an	Jahre Anzahl 19 1 22 1 24 1 27 2 28 4 29 2 30 2 33 1 35 1
Bitte geben Sie Ihr Geschlecht an	12 – männlich 3 – weiblich
Bitte geben Sie an, wie lange Sie Doodlings gespielt haben (in Std)	1,5 3 1,5 1 2 2 1 3 2 2 0,5 1 5 0,5 0.5
Wie viele Doodlings haben Sie gefangen?	Anzahl Doodlings Antworten 0 3 1 4 2 4 3 1 Mehr als 3 3

<table>
<tr><td>Bitte geben Sie Marke und Typ Ihres Smartphones an</td><td>Samsung Galaxy S4
Sony Xperia Z5
Samsung Galaxy S6
One Plus Two
Samsung Galaxy S3
Nexus 5
Lg G3
Huawei Ascend
Samsung Galaxy s 5
Samsung Galaxy S 5
LG Fino
Fairphone 2
Samsung Galaxy SII
Samsung Galaxy S3 Neo (Android 4.4.2)
Sony Xperia T</td></tr>
<tr><td>Bitte bewerten Sie das Spiel anhand der nachfolgenden Kriterien</td><td><table>
<tr><td></td><td>sehr hoch</td><td>hoch</td><td>mittel</td><td>niedrig</td><td>sehr niedrig</td></tr>
<tr><td>Spaßfaktor</td><td>3</td><td>9</td><td>2</td><td>1</td><td>-</td></tr>
<tr><td>Innovativität</td><td>6</td><td>7</td><td>2</td><td>-</td><td>-</td></tr>
<tr><td>Übersichtlichkeit</td><td>3</td><td>5</td><td>6</td><td>-</td><td>1</td></tr>
<tr><td>Schwierigkeit</td><td>-</td><td>2</td><td>7</td><td>5</td><td>-</td></tr>
<tr><td>Kreativität</td><td>6</td><td>4</td><td>3</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Bitte bewerten Sie die einzelnen Spielmechaniken anhand ihres Spaßfaktors</td><td><table>
<tr><td></td><td>sehr hoch</td><td>hoch</td><td>mittel</td><td>niedrig</td><td>sehr niedrig</td></tr>
<tr><td>Fangen der Doodlings</td><td>5</td><td>7</td><td>1</td><td>1</td><td>-</td></tr>
<tr><td>Kampf</td><td>1</td><td>2</td><td>7</td><td>3</td><td>-</td></tr>
<tr><td>Erstellen neuer Doodlingtypen</td><td>5</td><td>4</td><td>3</td><td>-</td><td>-</td></tr>
</table></td></tr>
<tr><td>Finden Sie die Möglichkeit aktiv Spielinhalte eines Spiels mitgestalten zu können interessant?</td><td>14 – Ja
1 – Nein</td></tr>
<tr><td>Sind während des Spielens Fehler aufgetreten?</td><td>7 – Ja
8 – Nein</td></tr>
</table>

Wenn Fehler aufgetreten sind, erläutern Sie diese bitte	• Nach dem Fang des neuen Doodlings hing das spiel im Kampfmodus bei "warte auf andere Spieler". Auf der Suche nach dem Doodling war die Position des Doodlings relativ ungenau, weil der Maßstab des Radars relativ groß ist. • Nach mehreren Spiel starten war nie ein Doodling in meiner direkten nähe. Also in weniger als <50 m Entfernung. • Obwohl ich keine Doodlings hatte konnte ich Gegner zum Kampf heraus fordern. Allerdings befand ich mich dann in einer Endlos Wartezone, von welcher ich aber auch zurück zum Hauptmenü wechseln konnte. • Das Spiel ist Bein kämpfen abgestützt • Wenn man vor dem Kampf nur einen Doodling auswählt, hängt das Spiel bei “warte auf andere Spieler“. • Es wurden keine Doodlings auf dem Radar angezeigt, beim klicken des "zurück" button auf dem radar screen hat es 7 sekunden gedauert um zurück ins menü zu kommen - dadurch war man versucht mehrfach zu drücken und wurde dadurch sofort wieder zurück auf den Radar screen geleitet (vielleicht die position des "play" buttons auf dem Menü screen nicht gleich wie den "zurück" Button) Ein Computer Gegner wurde auch nach 3 Minuten nicht geladen • Der erstellte Doodling steht beim Kampf nicht zur Verfügung. Muss das so? • Nach 60 Sekunden wurde ein automatischer Kampf initialisiert, beim zweiten Mal hat dies jedoch nicht mehr funktioniert und der Timer lief weiter (vermutlich nur für die Testphase von Bedeutung).
Wie viel Akku hat Doodlings verbraucht?	11 24 10 7 6 25 20 20 10 50 14 8

<table>
<tr><td>Wieviel Netzwerkdaten hat Doodlings gesendet und empfangen?</td><td>0.6
2,5
4
1,32
2,12
0,9
0,93
2,54
2,54
0,225
15
0,8
0,46</td></tr>
<tr><td>Bitte bewerten Sie, welche neuen Spielmechaniken in das Spiel übernommen werden sollen</td><td>
<table>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>Erstellen von Aufgaben und Rätseln durch Spieler</td><td>2</td><td>3</td><td>3</td><td>4</td><td>-</td><td>3</td><td>-</td></tr>
<tr><td>Gemeinsame Kämpfe von Spielern gegen besonders starke Doodlings</td><td>6</td><td>2</td><td>1</td><td>3</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>Gegenstände zum leichteren Einfangen/Verbessern von Doodlings</td><td>2</td><td>-</td><td>3</td><td>3</td><td>4</td><td>1</td><td>2</td></tr>
<tr><td>Doodlings als Tamagotchi</td><td>1</td><td>1</td><td>2</td><td>1</td><td>1</td><td>4</td><td>5</td></tr>
<tr><td>Gegenstände zur Unterstützung der Doodlings im Kampf</td><td>1</td><td>3</td><td>2</td><td>1</td><td>5</td><td>2</td><td>1</td></tr>
<tr><td>Automatisch generierte Turniere an bestimmten Geopositionen</td><td>1</td><td>-</td><td>3</td><td>3</td><td>2</td><td>4</td><td>2</td></tr>
<tr><td>Verbesserung des Doodlings durch vom Spieler ausgeführte sportliche Betätigung (Einbindung von z.B. GoogleFit)</td><td>2</td><td>6</td><td>1</td><td>-</td><td>2</td><td>-</td><td>4</td></tr>
</table>
</td></tr>
<tr><td>Verbesserungs-vorschläge</td><td>• Während des Kampfes könnte die ausgewählte Attacke visuell kenntlich gemacht werden. (Bsp. Aufleuchten). Der Maßstab des Radars könnte verringert werden. Ungefähr auf die Hälfte nach meinem Empfinden.</td></tr>
</table>

	Im Kampf könnten die Namen der Spieler angezeigt werden. Eine Beschreibung der Attribute wäre Hilfreich. • Die Doodlings sollten sich vom Typ her verändern, wenn man in einem speziellen Gelände ist z.B. Urban, Wald, Wasser, Mount Everest usw. • Bewegende Animaion für die Doodlings • Das fangen dauert sehr lange • Anzeigen der Spielernamen oder der KI im Kampf • Einen Button zum Verlassen des Spiels • Es fehlt ein Tutorial, das Spiel sollte starten und einem in einer kleinen interaktiven Anleitung das Spielziel und die Mechaniken erläutern. Das designen eigener Doodlings ist zu kompliziert, Vorschläge wären fertige Formen anbieten, Augen, Nasen, Ohren und den Charakter per Drag and Drop zusammensetzen. Anstelle der Malstifte zur Farbauswahl eine klassische Palette benutzen, ein Assistent der die Schritte nach und nach Abfragt sorgt für mehr platz auf dem Bildschirm so das keine Überfrachtung dort entsteht. Ein Vorschlag wäre auch, die Kamera mit zu benutzen und das Radar quasi interaktiv über die aktuelle Umgebung zu legen, da man ich mit dem Handy in der Hand viel bewegen muss, wechselt man ständig zwischen Handy und Umgebung hin und her. So wäre das ganze etwas interaktiver. Generell fehlen noch Tooltips. • 1.: Es sollte eine Art Tutorial für das Spiel geben, dabei würden Info-Fenster bereits genügen, welche man durch nicht mehr anzeigen dauerhaft verbergen kann (sollte in den Einstellungen zurückgesetzt bzw. neu gestartet werden können). Eines sollte auf dem "Radarbildschirm" die grundlegende Spielmechanik des Fangens der Doodlings erläutern (was bedeutet der Pfeil, in welche Richtung bewege ich mich als Spieler, wie fange ich Doodlings?), vor allem im Kampf ist jedoch ohne einen Infobildschirm mit einer Erläuterung der Grafiken (z.B. was machen Feuer / Wasser?) nicht unmittelbar klar, was zu tun ist - die Einbindung von Erläuterungen würde den Einstieg in Doodlings erleichtern und könnte so neue potentielle Spieler eher dazu bewegen, sich mit dem Spiel zu beschäftigen, bevor die Wahl auf ein anderes Spiel fällt. 2.: Das Antippen der Doodlings auf dem Radarbildschirm ist

	momentan ein wenig zu senisbel, sodass auf kleineren Bildschirmen <5 Zoll Toucheingaben oft nicht übernommen werden. Helfen würde zB ein größeres unsichtbares Feld, welches die Doodlings umschließt, bei dessen betätigung entsprechende Infos angezeigt werden. 3.: Im Kampf könnten nicht nur Gegenstände eingebunden werden, sondern auch so etwas wie Magie, welche durch Leveln / Gegenstände verbessert werden könnte. Dies könnte jedoch auch später in das Spiel impelemtiert werden und ist für die grundlegende Spielmechanik kein elementares Kriterium, sondern optional, um die Spieltiefe / den Umfang zu vergrößern. Hierbei könnte man sich an Kämpfen von rundenbasierten Rollenspielen orientieren (z.B. Final Fantasy/ Persona / Trails in the Sky etc.). 4.: Die Einbindung echter Karten z.B. von Open Street Map könnte die Immersion in das Spiel erhöhen, indem wirklich das Gefühl entsteht, dass die Kämpfe "in der eigenen Nachbarschaft" ausgetragen werden.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den _____ _____