Universität Hamburg
Fachbereich Informatik

Bachelorarbeit

Implementierung und Evaluierung einer innovativen Applikation zur sozialen Interaktion und Kommunikation

Erstgutachter: Dr. Guido Gryczan (Universität Hamburg)
Zweitgutachter: Dr. Paul Drews (Universität Hamburg)

Faisal Rashed
Studiengang Wirtschaftsinformatik
Matr.-Nr. 6143885
7. Fachsemester
faisal_r@msn.com

Inhaltsverzeichnis

1 Einleitung

In den letzten Jahren konnte eine rasante Entwicklung mobiler Geräte festgestellt werden. Diese ermöglichen neben der persönlichen Informationsverwaltung den mobilen Internetzugang und fördern die Erreichbarkeit und Produktivität der Nutzer. Dadurch gewinnen sie immer mehr an Bedeutung für Unternehmen und Privatpersonen. Besonders Smartphones konnten sich durchsetzten und sind ein wesentlicher Bestandteil der Kommunikation, Aufgabenbewältigung und Unterhaltung geworden. Laut der Firma Gartner haben sich die Verkaufszahlen seit 2009 mehr als verdoppelt. Im Jahr 2009 wurden ca. 172 Millionen Smartphones an den Endnutzer verkauft, während es 2012 über 427 Millionen Stück waren. (Pettey [2010, 2012])

Mit der steigenden Beliebtheit mobiler Geräte nahm auch die Bedeutung der Programme die ihre Funktionen erweitern zu. Diese sogenannten Apps werden auf zentralen internetbasierten Plattformen des jeweiligen Betriebssystems angeboten. Bei den Nutzern mobiler Geräte ist ein regelrechter Hype um diese kleinen Programme ausgebrochen. Die Firma Gartner errechnete für das Jahr 2011 einen Umsatz von 15,1 Milliarden US-Dollar, die durch den Verkauf von Applikationen über mobile App-Stores erzielt wurden. Im Vergleich zum Vorjahr hat sich der Umsatz verdreifacht. (Pettey [2011])

Durch diese App-Stores ist ein neuer Markt entstanden, in dem Anwendungen angeboten werden, die den Nutzer Unterhalten oder bei seinen täglichen Aufgaben unterstützen sollen. Applikationen die der Bewältigung von Aufgaben dienen, sollen zweckmäßig eingesetzt werden und Nutzen generieren. Im Bereich der Abendgestaltung, welches den Besuch von Veranstaltungen einschließt, konnten Defizite bestehender Applikationen festgestellt werden. Diese Problemstellung wird im Rahmen dieser Arbeit behandelt und eine Lösung entwickelt.

1.1 Ziel dieser Arbeit

Die Arbeit entsteht mit der Zielsetzung eine innovative Applikation, die der sozialen Interaktion und Kommunikation dient, zu liefern. Dabei ist die Software für einen bestimmten Anwendungsbereich konzipiert. Für die Festlegung der Anforderungen werden die Schwachstellen bestehender Applikationen identifiziert und analysiert. Diese bilden den Leitfaden für die Entwicklung der Anwendung. Des Weiteren soll die Usability und der Nutzen der resultierenden Software evaluiert werden.

1.2 Gliederung

Die Grundlagen zu mobilen Anwendungen und Betriebssystemen werden in Kapitel 2 dieser Arbeit vermittelt. Dabei erfolgt die Darstellung der Variationsmöglichkeit mobiler Anwendungen. Es wird ein Überblick über die am Markt vertretenen Betriebssysteme verschafft. Anschließend wird die Zielplattform der Applikation festgelegt und technische Details des Systems näher erläutert.

In Kapitel 3 erfolgt die Erstellung des Konzepts. Hier werden die Defizite bestehender Anwendungen aufgezeigt und die Grundfunktionalitäten der Applikation festgelegt. Dazu wird ein Szenario beschrieben in welchem der Einsatz der App dargestellt wird. Die erhobenen Funktionen werden abschließend von den bereits angebotenen Anwendungen abgegrenzt.

Die Umsetzung der festgelegten Grundfunktionalitäten wird in Kapitel 4 näher erläutert. Dabei wird zunächst die Architektur der Anwendung aufgezeigt. Anschließend erfolgt die genaue Betrachtung der einzelnen Systemkomponenten und ihrer Funktionsweise.

Das entwickelte System wird in Kapitel 5 evaluiert. Es soll ermittelt werden, ob die erhobenen Funktionen durch die Applikation bereitgestellt werden und ob diese den festgelegten Nutzen generieren. Aus dieser Evaluierung sollen Erkenntnisse erlangt werden, die der Verbesserung der Anwendung dienen.

Abschließend werden die Ergebnisse der Evaluierung gewürdigt. Mögliche Verbesserungen und deren Umsetzung werden in Aussicht gestellt. Der Autor hält Stellung zu den Ergebnissen und dem weiteren Vorgehen zur Einführung der Applikation an den Markt.

2 Grundlagen

2.1 Mobile Anwendungen

Mobile Anwendungen sind Programme, die für mobile Geräte wie Smartphones, Tablet PCs oder Netbooks entwickelt werden. Sie werden auch Applikationen oder kurz Apps genannt und sollen die Funktionalitäten des Gerätes erweitern. Durch die steigende Nachfrage nach mobilen Geräten entstand ein neuer Markt für mobile Anwendungen. Dabei reicht das Angebot von kleinen Werkzeugen und Fun-Apps bis hin zu umfangreichen Softwarepaketen. Derzeit lassen sich Apps in drei verschiedenen Kategorien unterteilen: Native Anwendungen, Mobile Websites und Hybride Apps.

2.1.1 Native Anwendungen

Native Apps sind Applikationen, die Betriebssystem spezifisch entwickelt werden und in der Regel nur auf diesem ausgeführt werden können. Dadurch wird eine optimale Untersetzung durch das System und die bestmögliche Geschwindigkeit erreicht. Des Weiteren werden eine effiziente Hardwareunterstützung und eine maximale Ausreizung der geräte- und softwarespezifischen Eigenschaften ermöglicht. Der Nachteil besteht darin, dass die Applikation für jedes Betriebssystem neu entwickelt bzw. angepasst werden muss. Die Apps werden in Onlineshops des jeweiligen Betriebssystems angeboten und können direkt auf dem Gerät installiert werden. Die Vertriebsplattformen der Betriebssysteme sind:

Anbieter	Store
Google	Google Play
Apple	App Store
Blackberry	BlackBerry App World
Nokia	Nokia Store
Sony Ericsson	PlayNow
Samsung	Samsung Apps
Microsoft	Windows Phone Store
LG	Smart World

Mit über 800.000 Inhalten ist der Google Play Store das größte Vertriebsportal für Applikationen gefolgt vom Apple Store. (Jens [2013]) Den Entwicklern wird dadurch die Möglichkeit gegeben ihre Apps anzubieten. Die Kosten hierfür variieren je Anbieter, ebenso die technische und inhaltliche Prüfung der Anwendung.

2.1.2 Mobile Web-Apps

Der Internetzugriff mit mobilen Geräten ist mittlerweile eine Standardfunktion und ermöglicht das Surfen über Webbrowser. Allerdings sind die Inhalte der Webseiten hochauflösend. Dadurch werden auf kleinen Geräten wie Smartphones nur Teile der Oberfläche angezeigt. Die Lösung hierfür wird durch Web-Apps realisiert. Diese reduzieren den Inhalt

der Webseite auf das Wesentliche und ermöglichen eine ideale Ansicht über den Webbrowser. Diese Anwendungen müssen nicht heruntergeladen werden, können aber auch nicht offline genutzt werden. Die Entwicklung erfolgt mit weitverbreiteten Sprachen wie HTML, XML, JavaScript und CSS. Die Nachteile dieser Anwendungen sind zum einen der Bedarf einer Internetverbindung und zum anderen der fehlende Hardwarezugriff. Ohne Zugang zum Web können diese Apps nicht genutzt werden. Außerdem entstehen lange Wartezeiten bei schlechten Verbindungen oder geringer Datenübertragungsgeschwindigkeit. Die Anwendungen sind webbasiert und können nicht auf Hardware-Komponenten wie der Kamera zugreifen.

2.1.3 Hybride Apps

Eine Möglichkeit die Vorteile der nativen Applikationen und der Web-Apps zu kombinieren wird durch die Hybride Apps geboten. Dabei handelt es sich um Web-Apps, die durch einen Mantel aus nativem Code gehüllt sind. Das bedeutet, dass die Anwendung mit Web-Technologie entwickelt und anschließend in eine native Applikation transformiert wird. Die Anwendungslogik wird in HTML, XML, JavaScript und CSS entwickelt und durch die zusätzliche Verwendung nativen Codes wird die Installation auf den Geräten ermöglicht. Diese Apps können plattformübergreifend eingesetzt werden und trotzdem auf Hardware-Komponenten zugreifen. Außerdem können sie über App-Stores angeboten werden. Obwohl die Anwendung installiert werden muss und sich lokal auf dem Gerät befindet wird der Webbrowser für die Ausführung verwendet, dadurch entstehen Nachteile in der Performance der Anwendung. Des Weiteren können lange Wartezeiten beim Seitenaufbau durch schlechte Internetverbindungen oder geringer Datenübertragungsgeschwindigkeit entstehen. (vgl. Bertram [2010])

2.2 Betriebssysteme mobiler Geräte

Um mobile Geräte für den Nutzer verwendbar zu machen werden entsprechende Betriebssysteme benötigt. Diese bezwecken die Kommunikation mit dem Gerät so angenehm und effizient wie möglich zu gestallten. Außerdem erlauben sie mobilen Anwendungen, die nicht Teil des Betriebssystems sind auf dem Gerät zu arbeiten. Es wird eine Reihe von verschiedenen Systemen angeboten. Abhängig vom Modell und Hersteller des mobilen Gerätes variiert das verwendete Betriebssysteme. Die erfolge die am Markt erzielt werden sind somit beiden Faktoren geschuldet. Während die Nutzer nur geringe Unterschiede zwischen den verschiedenen Betriebssystemen feststellen, stehen Entwickler von Applikationen neuen Dimensionen gegenüber. Diese Weisen gravierende Unterschiede in der verwendeten Programmiersprache und der Entwicklungsumgebungen auf.

Im Folgenden wird der Betriebssystemmarkt näher erläutert. Es erfolgt eine Beschreibung der Entwicklung des relativ jungen Marktes. Anschließend werden relevante Systeme unter die Lupe genommen.

2.2.1 Marktverteilung

Mit der steigenden Nachfrage nach Smartphones ist auch die Anzahl der Betriebssysteme gestiegen. Es konnten sich verschiedene Systeme am Markt etablieren. Im Jahr 2007 stellte die Firma Apple das iPhone mit Betriebssystem iPhone OS[1] vor. Im selben Jahr kündigte Google mit 33 weiteren Mitgliedern der Open Handset Alliance die Entwicklung des Betriebssystems Android an. Weitere Anbieter wie Microsoft folgten kurz darauf. Laut des Jahresreports der Firma Gartner konnte sich das Betriebssystem Android auf dem Smartphone Markt durchsetzen. (Pettey [2012])Ende des dritten Quartals 2012 machte Android 72,4% des Martes aus gefolgt von iOS mit 13,9%. Microsoft erreichte mit dem Windows Phone lediglich 2,4% (siehe Abbildung 1).

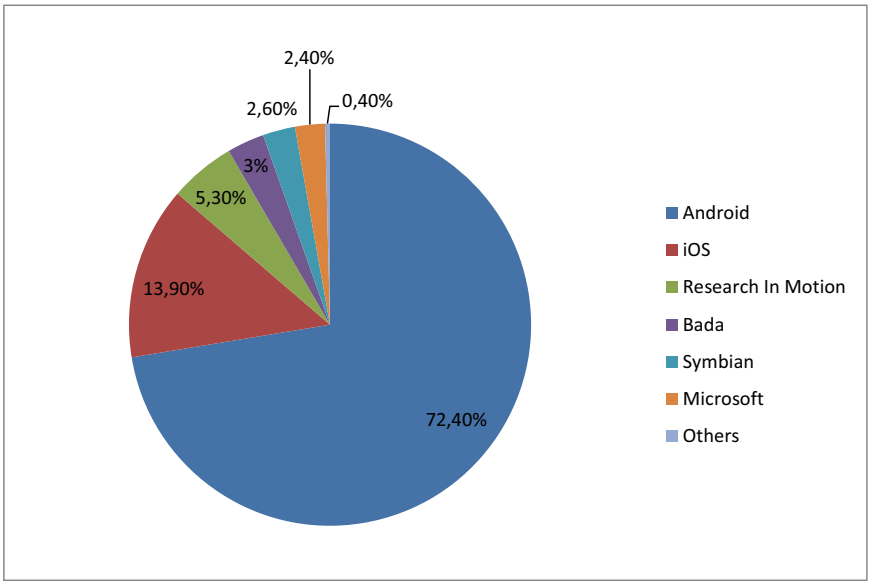

Abbildung 1: Marktanteile Mobiler Smartphone Betriebssysteme Q3 2012
Quelle: Eigene Darstellung basierend auf Pettey [2012]

Auf dem Tablet PC Markt konnten sich die Betriebssysteme von Appel und Google durchsetzen. Ende des dritten Quartals 2012 machte iOS 56,7% des Marktes aus, dicht gefolgt von Android mit 41,3%. Die restlichen 2% wurden von anderen mobilen Betriebssystemen wie Microsoft und weiteren Anbietern ausgemacht (siehe Abbildung 2).

[1] 2010 umbenannt in iOS

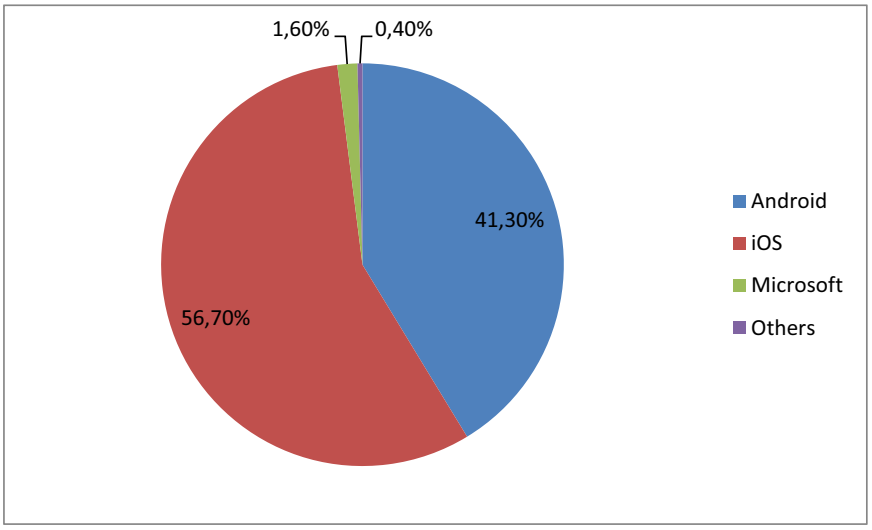

Abbildung 2: Marktanteile Mobiler Tablet PC Betriebssysteme Q3 2012
Quelle: Eigene Darstellung, basierend auf Mawston [2012]

Der Tablet PC Markt ist angesichts drei nennenswerter Betriebssysteme recht übersichtlich. Wobei zwei diesen Markt eindeutig dominieren. Der Betriebssystemmarkt für Smartphones ist bei der großen Menge an angebotenen Systemen etwas schwieriger zu durchdringen. Für ein besseres Verständnis wir die Entwicklung der letzten drei Jahre betrachtet.

Marktanteile mobiler Betriebssysteme für Smartphones

Company	Year*		Point of Change
	2009	2012	
Symbian	46,9	2,6	-44,3
Android	3,9	72,4	68,5
Research In Motion	19,9	5,3	-14,6
iOS	14,4	13,9	-0,5
Microsoft	8,7	2,4	-6,3
Other Oss	6,1	3,4	-2,7

*Angaben für jedes Jahr in Prozent

Abbildung 3: Quelle: Eigene Darstellung, basierend auf Pettey [2010, 2012]

Während Symbian 2009 noch 46,9 Prozent des jungen Marktes ausmachte, waren es 2012 lediglich 2,6 somit verlor der Marktführer seine Position. Apple ging im Laufe der Jahre um 0,5 Punkte zurück und stellt den stabilsten Wert des Marktes dar. Aus der Tabelle wird deutlich, dass nur Android eine positive Entwicklung und einen Wachstum von 68,5 Punkten erzielen konnte. Dadurch konnte sich das von Google entwickelte Betriebssystem,

welches anfänglich nur 3,9 Prozent ausmachte durchsetzten und dominiert derzeit den Markt. Dabei ist der Vorsprung gegenüber den anderen Anbietern sehr groß. Gefolgt wird Android von Apple mit 13,9 Prozent des Marktes, somit liegt Android 58,5 Punkte vor iOS.

2.2.2 Relevante Systeme

Für mobile Geräte wird eine Reihe von verschiedenen Betriebssystemen angeboten. Angesichts der Marktverteilung werden im Rahmen dieser Arbeit die Systeme von Apple, Google und Microsoft näher betrachtet. Diese sind sowohl auf dem Smartphone Markt als auch auf dem Tablet PC Markt vertreten und stellen die führenden Systeme dar. Weitere Betriebssystem wie Symbian, RIM und Baba werden angesichts der schwindenden Anteile am Markt nur der Vollständigkeit halber genannt.

iOS

Das von Apple entwickelte Betriebssystem iOS ist nur für Apple Produkte erhältlich. Es basiert auf Mac OS und überzeugt vor allem durch die Benutzerfreundlichkeit. Die Bedienung der Geräte erfolgt dabei direkt über den Touchscreen. Auf dem System werden die für Apple typischen Anwendungen wie:

- Safari
- iTunes
- App Store
- Apple Mail

angeboten. Diese Standardanwendungen können nicht vom System entfernt werden. Zusätzliche Programme, Spiele und Musik können über den App Store und über iTunes heruntergeladen werden. Für die Entwicklung von Applikationen für iOS werden ein Mac Computer und ein iPhone vorausgesetzt. Die Entwickler müssen sich außerdem mit der Programmiersprache Objective-C auseinandersetzen. Gegen eine jährliche Gebühr von 99 US-Dollar können entwickelte Anwendungen über den App Store angeboten werden. Zusätzlich fällt bei jedem Verkauf eine Transaktionsgebühr von 30 Prozent des Verkaufspreises an. Vor der Veröffentlichung der Applikationen werden diese auf Schadsoftware und Inhalt geprüft. Dabei hält sich Apple das Recht offen bestimmte Inhalte nicht zu veröffentlichen. Kritiker sehen dies als Zensur und innovationshemmend. Zusammenfassend bildet Apple durch seine Produkte ein geschlossenes System. Innerhalb dieses Systems wird auf Benutzerfreundlichkeit und Komplexitätsreduktion gesetzt. Die Anwendungen und Geräte sind dabei optimal aufeinander abgestimmt.(vgl. Inc. [2013])

Android

Das von Google entwickelte Betriebssystem Android ist in vielen mobilen Endgeräten wie Handys, Smartphones, Tablet PCs und Netbooks vertreten. Laut der offiziellen Webseite von Android werden jeden Tag weltweit über eine Millionen neuer Android-Geräte

aktiviert. (Google [2013]) Das Betriebssystem basiert auf einem Linux-Kern und ist frei erhältlich. Google präsentiert Android als Open-Source-Projekt, das durch den offenen Zugang zur Software Punkten möchte. Es sollen keine Einschränkungen und Kontrollen der entwickelten Innovationen zugelassen werden. Diese Quelloffenheit des Systems wurde oftmals kritisiert. In einer EU Studie, die Open-Source-Projekte bewertete, belegte Android den letzten Platz und wurde „das am wenigsten offene Open-Source-Projekt" bezeichnet.(Laffan [2011]) Dennoch steht der Großteil der Plattform unter der Apache Lizenz 2.0[2].

Die Entwicklung von Applikationen erfolgt mit der Programmiersprache Java. Dabei kann auf jedem beliebigen System implementiert werden. Entwickler erfahren große Unterstützung durch kostenlose Werkzeuge und ausführliche Dokumentationen. Die Applikationen können im Google Play Store angeboten werden. Dabei fällt eine einmalige Gebühr von 25 US-Dollar an. (Google [2013])Google beteiligt sich ebenfalls mit 30 Prozent des Kaufpreises an Verkaufstransaktionen. Auf eine explizite Prüfung der Anwendung, wie es bei Apple der Fall ist, wird verzichtet. Die Anwendung wird lediglich auf Malware geprüft.

Windows

Microsoft ist sowohl auf dem Markt der Betriebssysteme für Smartphones als auch für Tablet PCs vertreten. Die derzeit neuste Version für Tablet PCs ist Windows 8 und für Smartphones Windows Phone 8. Mit diesen Systemen versucht Microsoft an die Beliebtheit von Windows als Betriebssystem für PCs und Laptops anzuknüpfen. Der Nutzer erhält ein einheitliches System, welches den Umgang mit den Produkten erleichtern soll. Applikationen werden in der Programmiersprache C# geschrieben. Den Entwicklern steht dabei das kostenlose Werkzeug Windows Phone Developer Tool zur Verfügung. Diese können ihre Anwendungen im Windows Phone Store anbieten. Dabei fällt eine jährliche Gebühr von 99 US-Dollar an. Microsoft beteiligt sich ebenfalls mit 30 Prozent an Verkaufstransaktionen.(Microsoft [2012])

2.3 Zielplattform

Angesichts der vorgestellten varianten mobiler Anwendungen wird im Rahmen dieser Arbeit eine native Applikation entwickelt. Dadurch kann eine effiziente Ausführung der Anwendung sichergestellt werden, da native Anwendungen eine optimale Unterstützung durch das System erfahren. Außerdem ist die offline Verwendung der Applikation sichergestellt. Für die Ausführung wird der Webbrowser nicht benötigt. Des Weiteren können native Apps über die entsprechenden Stores der Betriebssysteme angeboten werden. Dies ermöglicht eine optimale Erreichbarkeit der potenziellen Nutzer. Die Entwicklung einer hybriden App wurde bewusst abgelehnt, da durch die Ausführung der Anwendung über den Webbrowser Performanceprobleme entstehen können. Außerdem sind die für die Entwicklung erforderlichen Werkzeuge mit kostenverbunden, diese werden nur mit der Basisausstattung kostenlos angeboten. (vgl. Bertram [2010])

[2]Eine von der Free Software Foundation anerkannte Lizenz für freie Software

Unter Berücksichtigung der vorgestellten Betriebssysteme für mobile Geräte und deren Marktpositionierung, fiel die Wahl der Zielplattform der nativen Applikation auf Android. Dieses System bietet eine ideale Unterstützung der Entwickler und ihrer Innovationen. Werkzeuge und Dokumentationen werden kostenlos angeboten. Das Veröffentlichen von Applikationen über den App Store dieses Betriebssystems ist mit den geringsten Kosten verbunden. Außerdem bildet der Google Play Store den größten Markt für Apps. Angesichts der derzeitigen Marktverteilung ist Android das beliebteste Betriebssystem. Durch die Entwicklung einer nativen Applikation können somit über 72 Prozent der Smartphone-Nutzer erreicht werden. Die Marktentwicklung seit 2009 lässt außerdem einen weiteren Anstieg vermuten. Zusammenfassend ergeben sich die Vorteile zum einen aus den geringen Kosten und zum anderen kann eine sehr breite Masse erreicht werden. Diese Faktoren sind vor allem bei der Neueinführung einer Applikation, welches ein hohes Risiko mit sich bringt von großer Bedeutung.

Im Folgenden werden einige Grundlagen über Android vermittelt. Dabei werden wichtige Aspekte bei der Entwicklung von Applikationen dargestellt. Anschließend werden technische Details über die Funktionsweise des Systems vermittelt und Tools vorgestellt, die im Laufe der Arbeit verwendet werden.

2.3.1 Google Play

Um den Austausch von Applikationen für mobile Android-Geräte zu ermöglichen wurde 2008 der Android Market von Google zur Verfügung gestellt. Dieser Market war dem App Store von Apple nachempfunden und ermöglichte es Entwicklern ihre Applikationen anzubieten. Durch die Vereinigung des Android Market mit Google Music, Google Books und Google Movies entstand 2012 die Cloud-basierte Plattform Google Play. Sämtliche Programme, Spiele, Bücher und Filme für Android kompatible Geräte werden hier angeboten. Anfang 2013 waren es über 800.000 Inhalte. (vgl. Jens [2013]) Um Applikationen anzubieten muss ein Entwickler-Account eingerichtet werden. Anschließend erfolgt das Hochladen von Apps über die Google Play Developer Console. Entwickler können sich zu ihren Anwendungen die Download-Zahlen und die Anzahl der aktiven Nutzer der App ansehen.

2.3.2 Entwicklungsunterstützung

Anwendungen werden generell in der Programmiersprache Java entwickelt. Java ist eine weitverbreitete Programmiersprache und an vielen Universitäten Teil der Ausbildung. Die Anzahl der potenziellen Entwickler ist daher sehr groß. Um diese zu unterstützen und den Einstieg zu der Entwicklung von Android Applikationen zu erleichtern, bietet Google eine einsteigerfreundliche und ausführliche Dokumentation, Tutorials und Code-Beispiele an. Mit dem Android SDK[3] werden alle Tools zur Verfügung gestellt, die benötigt werden

[3] ein Software Development Kit (SDK) ist eine Sammlung von Werkzeugen und Anwendungen die zur Entwicklung von Software benötigt werden

um Anwendungen für die Android Plattform zu entwickeln. Dazu gehören der Cross-Compiler, der Debugger, der Emulator und die Bibliotheken. (vgl. Saha [2008]) Android baut zum einen auf das bestehende Java Wissen der Entwickler auf und zum anderen kann durch die Eclipse IDE die vorhandene Entwicklungsumgebung weiterverwendet werden. Hierfür müssen lediglich die Android Development Tools (ADT) als Plug-In in Eclipse installiert werden.

2.3.3 Architektur

Android basiert auf einem Linux-Kernel, der die Gerätetreiber enthält und für die Speicher-, Energie- und Prozessverwaltung zuständig ist. Das Herzstück der Laufzeitumgebung ist die Dalvik Virtual Machine (DVM). Sie ähnelt der Java Virtual Machine und führt auch Bytecode aus. Durch den Cross-Complier wird der Java-Bytecode in Dalvik-Bytecode umgewandelt. Für jede Anwendung wird eine DVM-Instanz in einem Betriebssystemprozess gestartet. Dies hat den Vorteil, dass die Anwendungen sich keinen gemeinsamen Speicher teilen und ein sterbender Prozess nur eine Anwendung zerstört. Die in C /C++ geschrieben Bibliothek stellt sämtliche verfügbaren Funktionalitäten, die eine Anwendung haben kann bereit. Dazu gehören unteranderem eine Datenbank (SQLite), ein Webbrowser (WebKit), Audio/Video-Wiedergabe (Mediaframework) und eine 3-D Grafikbibliothek (OpenGL). Um die Hardwarekomponenten den Entwicklern verfügbar zu machen werden verschiedene Manager-Klassen definiert. Soll beispielsweise die Position des Gerätes ermittelt werden kann der Location Manager verwendet werden. Diese Manager-Klassen werden in Java geschrieben und abstrahieren die zugrunde liegenden Hardwarekomponenten und ermöglichen den Entwicklern eine effiziente Nutzung aller Sensoren. Auf der Anwendungssicht befinden sich sämtliche Anwendungen des Systems. Auf dieser Ebene findet die Interaktion zwischen Nutzer und Anwendung, sowie der Anwendungen untereinander statt. Hier werden auch die darunter liegenden Systemkomponenten über die Schnittstellen verwendet. Durch die Losekopplung und das Nutzen von Schnittstellen sind die Anwendungen austauschbar und können einfach gelöscht und installiert werden. Es werden einige Standardanwendungen von Google mitgeliefert. Darüber hinaus können eigene oder über Google Play erworbene Applikationen auf dem mobilen Gerät verwenden werden. (vgl. Becker [2010] S.19 ff.)

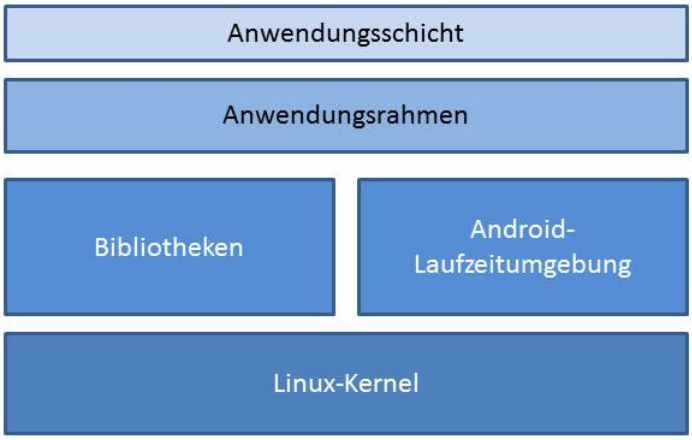

Abbildung 4: SystemarchitekturQuelle: Eigene Darstellung, basierend auf Becker [2010]

2.3.4 Komponenten

Die starke Modularisierung des Systemaufbaus ermöglicht es einzelne Komponenten einfach auszutauschen und weiterzuentwickeln. Diese Flexibilität zeichnet sich auch bei dem Aufbau der Programme ab. Neben den Java-Klassen setzt sich eine Applikation aus verschiedenen Android spezifischen Komponenten zusammen:

- Activities
- Services
- Content Provider
- Broadcast Receiver
- Android-Manifest
- Ressourcen

Activities

Die grafische Oberfläche einer Anwendung wird hauptsächlich in XML-Layoutfiles (Views) definiert. Diese enthalten die Elemente, die für den User sichtbar sind und mit denen er interagiert. Für die Verwaltung und Darstellung dieser Oberflächenelemente sind Activities verantwortlich. Diese reagieren auf die Eingaben des Users und enthalten die Programmlogik. Durch die Verknüpfung von Activities können komplexe Anwendungen erzeugt werden. Die Kommunikation zwischen Activities erfolgt durch Intents, die den Nachrichtenaustausch auf Anwendungsebene darstellen. Dadurch wird die Interaktion zwischen den Activites ermöglicht, der Programmfluss kann gestaltet und Informationen

ausgetauscht werden. Man unterscheidet zwei Arten von Intents: Ist der Empfänger bereits bei der Implementierung bekannt so spricht man von einem expliziten Intent. Wird der Empfänger erst durch die angegeben Parameter bestimmt so handelt es sich um einen impliziten Intent.

Services

Neben den Activities die den Programmfluss enthalten kann die Programmlogik auf Services ausgelagert werden. Dies sind Prozesse, die im Hintergrund laufen und keine direkte Interaktion mit dem User anbieten. Im Gegensatz zu Activities können sie einen längeren Lebenszyklus haben und trotz Schließung der Oberfläche im Hintergrund weiterlaufen. Dadurch können Teile der Anwendung, die viel Zeit beanspruchen oder aktiv bleiben sollen, realisiert werden ohne die Performance zu beinträchtigen. Die Trennung von Activities und Services ermöglicht es die Benutzeroberfläche ständig verfügbar zu machen und trotz langwieriger Hintergrundprozesse auf Eingaben des Users zu reagieren. Services können dabei andere Komponenten verwenden, Intents verschicken oder über Content Provider auf Daten zugreifen.

Content Provider

Wie in Abschnitt 2.3.3 erwähnt, wird für jede Anwendung eine eigene DVM-Instanz gestartet und für eine separate Datenhaltung gesorgt. Um die Interaktion und den Informationsaustausch zwischen den Anwendungen zu gewähren werden sogenannte Content Provider eingesetzt. Da jede Applikation einen eigenen Speicher erhält können andere Anwendungen nicht auf diese Daten zugreifen. Um dies trotzdem zu gewährleisten und beispielsweise den Zugriff auf die Kontakte oder Termine zu ermöglichen werden Content Provider angeboten. Diese bilden eine Datenschnittstelle und können über einen URI (Uniform Resource Identifier) eindeutig identifiziert werden. Die enthaltenen Daten werden auch eindeutig adressiert und dadurch verfügbar gemacht. Diese Adressen werden als Extension der Basisadresse des Content Providers realisiert.

Broadcast Receiver

Intents werden nicht nur bei der Kommunikation zwischen Activities und Services verwendet, die Plattform selber kann auch Intents verschicken. Dadurch können Anwendungen über den Systemstatus benachrichtigt werden. Dies könnte unter anderem eine Nachricht über den Batteriestatus sein. Um diese Intents erhalten zu können wird der Broadcast Receiver angeboten. Dieser ähnelt einem Listener und lauert auf Benachrichtigungen des Systems. Die Implementierung des Broadcast Receivers erfolgt entweder im Android-Manifest oder in bestimmten Activities bzw. Services. Bei der ersten Variante wird der Broadcast Receiver bei der Installation der Anwendung mit dem System verbunden und ist bereit für das Empfangen von Intents. Dadurch kann die Applikation durch eine Systemnachricht gestartet werden. Bei der zweiten Variante ist der Receiver an den Lebenszyklus der Komponente, in der die Implementierung erfolgt, gebunden d.h. Systembenachrichtigungen können nur in dieser Zeitspanne empfangen werden.

Android-Manifest

Ein wesentlicher Bestandteil einer Android Applikation ist das Android-Manifest. Hierbei handelt es sich um eine XML-Datei, die alle wichtigen Informationen enthält, die das System benötigt um die Anwendung auszuführen. In dieser Datei werden unteranderem:

- alle Komponenten der Anwendung aufgelistet
- sämtliche Berechtigungen vergeben die benötigt werden um bestimmte Teile der API zu verwenden
- Berechtigungsanforderungen definiert um Teile der Anwendung von außen zu verwenden
- das benötigte API Level angegeben
- und es wird bestimmt mit welcher Activity die Anwendung gestartet werden soll.

Das Manifest ist für das Ausführen der Applikation fundamental und gibt den Entwicklern einen groben Überblick über die Komponenten der Anwendung und ihrer Interaktion.

Ressourcen

Das Komponentenmodell von Android spiegelt sich auch bei der Verwaltung quelltextfremder Elementen wieder. Sämtliche Daten, die kein Java-Quellcode sind, werden in dem Ordern Ressourcen abgelegt. Hierzu gehören unteranderem die XML-Layouts (Views), sämtliche Grafiken, Texte und Styles. Durch diese Separierung wird eine effizientere Gestaltung der Benutzeroberfläche ermöglicht. Eine Bilddatei kann in verschiedenen Auflösungen als Ressource abgelegt werden und je nach Bildschirmgröße des Gerätes verwendet werden. Auch Texte können in verschieden Sprachen abgelegt werden. Dies ermöglicht den multilingualen Einsatz der Applikation. Änderungen an der Oberfläche können vorgenommen werden ohne den Quellcode zu bearbeiten.

2.3.5 GCM

Mit dem kostenlosen Dienst Android Cloud to Device Messaging (C2DM) ermöglichte Google Android-Entwicklern Nachrichten in der Größe von 1 Kilobyte an ihre Applikationen zu senden. Dies könnte beispielsweise eine Mitteilung über ein verfügbares Update sein. Diese kostenlose Alternative zu bekannten Push-Nachrichten wie SMS stieß auf große Nachfrage seitens der Entwickler. Im Juni 2012 wurde das C2DM durch Google Cloud Messaging for Android (GCM) ersetzt, welches das Versenden von Daten in der Größe von 4 Kilobyte ermöglicht. Dies wird durch eine bestehende Verbindung des Android-Gerätes mit einem Google-Konto realisiert. Das Gerät wird dabei ständig mit dem Google-Konto synchronisiert und ist dadurch erreichbar. Nachrichten können von einem Server über die GCM-Technologie direkt an die Applikation gesendet werden. Um die Push-Nachricht zu erhalten muss die Anwendung nicht gestartet sein. Das Android-System weckt die Applikation über einen Broadcast Intent. (vgl. Google [2012])

Anforderungen

Um den Push-Nachrichtenservice von Google nutzen zu können wird ein Google-Account benötigt. Unter diesem Account muss ein neues Google API Projekt angelegt werden. Dadurch wird von Google eine Projektnummer (Sender-ID) und ein API-Schlüssel vergeben. Diese werden verwendet um mit dem Google Server zu kommunizieren. Des Weiteren muss ein Server aufgesetzt werden von dem die zu übermittelnden Daten versendet werden. Android-Geräte auf denen die Applikation ausgeführt werden soll, benötigen die Android Version 2.2 oder höher und die Google Play Store Applikation, die standardmäßig installiert ist. Dadurch wird eine Verbindung mit dem Google-Konto hergestellt und das Gerät für den Nachrichtenservice verfügbar gemacht.

Architektur

Das Android-Gerät auf dem die Anwendung ausgeführt wird beantragt zunächst eine Registration-ID beim GCM Server. Dieser antwortet umgehend mit einer eindeutigen ID. Als nächstes muss diese ID dem Server, der die Daten an die Geräte versendet, mitgeteilt werden. Dadurch sind alle Geräte über eine eindeutige Adresse erreichbar. Jedoch können sie nicht direkt angesteuert werden, sondern über den GCM Server. Dieser erhält vom Applikationsserver die ID und die zu übermittelnden Daten und leitet sie an die Geräte weiter.

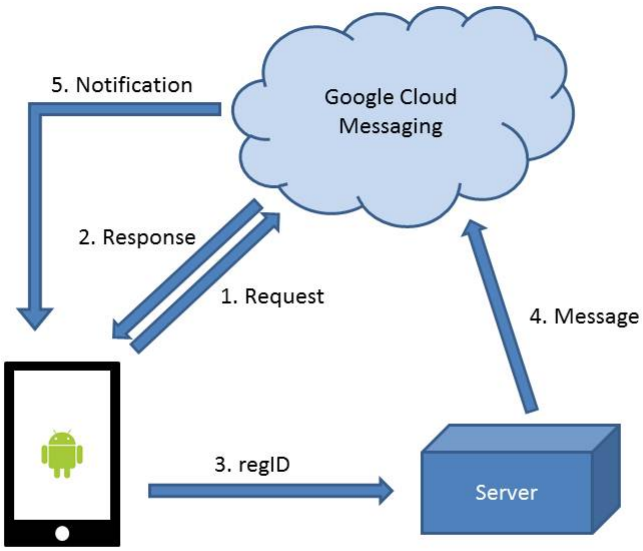

Abbildung 5: GCM ArchitekturQuelle: Eigene Darstellung

2.3.6 SQLite

Im Lieferumfang von Android ist das Datenbanksystem SQLite inbegriffen. Dieses ermöglicht die Aufbewahrung von Daten, die über die Lebensdauer einer Anwendung hinaus gespeichert werden sollen. Des Weiteren wird die Zwischenspeicherung externer Daten ermöglicht, wodurch mehrfach Anforderungen innerhalb der Anwendung vermieden werden können. SQLite erfordert keine weitere Installation. Es handelt sich dabei um eine Programmbibliothek, die ein relationales Datenbanksystem enthält, welches für den Einsatz auf mobilen Geräten optimiert ist. Dabei können ein Großteil der Befehle der Datenbanksprache SQL verwendet werden. Durch das Anbinden dieser Bibliothek wird eine Anwendung um Datenbankfunktionalitäten erweitert. SQLite bedarf keiner Client-Server-Architektur und wird als serverloses Datenbanksystem bezeichnet. Die Datenbank wird als Datei gespeichert und benötigt nur sehr wenig Speicher (höchstens 250 kB). In dieser Datei wird die gesamte Datenbank samt Tabellen, Indizes und Views gespeichert. Der Zugriff auf eine Datenbank kann aus mehreren Anwendungen gleichzeitig erfolgen. Es bestehen Einschränkungen bei der Vergabe von Berechtigungen. Diese können nur als Zugriffsrecht auf die Datenbank-Datei vergeben werden d.h. die Zugriffsberechtigung auf die Datenbank wird einer Anwendung entweder komplett oder gar nicht vergeben. (vgl. Becker [2010] S.225 ff.)

3 Konzept

Dieses Kapitel widmet sich dem Konzept, der in dieser Arbeit entwickelten Applikation. Es wird zunächst ein Szenario beschrieben, dass die mögliche Umsetzung der Nutzung dieser Applikation darstellt. Dabei sollen die Funktionalitäten des finalen Produktes nur grob umschrieben werden. Im Anschluss werden bestehende Applikationen dieses Anwendungsbereiches näher betrachtet und ihre Schwachstellen herausgearbeitet. Hieraus ergibt sich der Leitfaden für die Entwicklung der Anwendung. Abschließend werden die Grundfunktionalitäten festgelegt und von bereits existierenden Apps auf dem Markt abgegrenzt.

3.1 Szenario

Die Applikation soll dem Nutzer die Möglichkeit bieten einen Überblick über die eingetragenen Informationen zu erhalten. Bei diesen Informationen wird es sich um Veranstaltungsdaten handeln. Dabei ist das Ziel, dass der Nutzer sich über Veranstaltungen informiert und gegebenenfalls sein Interesse geweckt wird. Ist dies der Fall können weitere Details zu dem betreffenden Event eingesehen werden. Entscheidet sich der Nutzer für dieses Event können mit nur wenigen eingaben Freunde benachrichtigt werden. Diese werden ebenfalls in einen Gruppenchat eingeladen, in der weitere Details besprochen werden können. Ein mögliches Szenario könnte wie folgt aussehen:

Die Woche neigt sich dem Ende und Peter möchte mit seinen Kommilitonen ausgehen. Er weiß nur noch nicht wohin. Unter Zuhilfenahme der Applikation schaut er sich anstehenden Veranstaltungen an. Dabei hat ihn ein Event überzeugt, der Eintritt ist akzeptabel und es gibt eine „Happy Hour“. Dies wird auch seinen Kommilitonen gefallen also beschließt er ihnen Bescheid zugeben. Aus der Liste seiner Kontakte kann er diese aussuchen und benachrichtigen. Dabei wird ein Gruppenchat, der die Informationen zu dem Event enthält erstellt und die entsprechenden Kontakte eingeladen. Hier wird nun die Diskussion über die Abendgestaltung angestoßen. Seine Kommilitonen können die Informationen zu dem Event einsehen und sich dazu äußern. Nach dem alle mit der Veranstaltung einverstanden sind schlägt Peter vor gemeinsam hinzufahren und die Bahn zu einer bestimmten Uhrzeit zu nehmen. Nach einer weiteren Diskussion legen sie sich auf 22:00 Uhr fest. Lisa, die den ganzen Tag an der Universität verbrachte und die Einladung erst spät am Abend erhält kann trotz des langen Chatverlaufes die Informationen zu der Veranstaltung schnell einsehen und beschließt, mitzukommen.

In diesem Szenario wird der mögliche Einsatz der Applikation dargestellt. Es werden die grundlegenden Funktionalitäten beschrieben und deren Nutzen aufgezeigt.

3.2 Eignung bestehender Anwendungen

Durch die in dieser Arbeit entwickelten Applikation soll ein Informations-und Kommunikationssystem realisiert werden. Für den geplanten Anwendungsbereich existieren be-

reits diverse Systeme, die sich auf die reine Informationsbereitstellung spezialisiert haben. Ebenso konnten sich einige Anwendungen im Bereich der Kommunikation durchsetzen. Im Folgenden werden diese Applikationen näher betrachtet und ihre Schwachstellen herausgearbeitet. Dabei wurden speziell die Anwendungen im Google Play Store betrachtet, welches den größten Markt für Applikationen bildet (siehe 2.3.1) und den App Store der Zielplattform darstellt. Diese Recherche spiegelt den Stand am 28.01.2013 wieder und bezieht sich auf das Angebot für Deutschland. Es wird zunächst der Anwendungsbereich grob umschrieben und anschließend die Ergebnisse der Recherche dargestellt.

3.2.1 Anwendungsbereich

Der Einsatz der Applikation soll der sozialen Interaktion dienen. Dabei soll speziell der Prozess der Abendgestaltung, der mit dem zusammenfinden sozialer Kontakte verbunden ist unterschützt werden. Hierfür bedarf es der Auskunft zukünftiger Veranstaltungen. Der Nutzer soll die Möglichkeit erhalten sich über diese zu informieren und bei der weiteren Planung unterstützt werden. Weitere Schritte wären dabei, das Teilen besagter Informationen mit sozialen Kontakten und der Austausch bezüglich weiterer Einzelheiten. Der Aspekt der Kommunikation steht im Vordergrund. Speziell bei der Interaktion mit mehreren Individuen ($\geq$3) muss für einen effizienten Informationsaustausch gesorgt werden.

3.2.2 Ergebnisse der Recherche

In Anbetracht des Anwendungsgebietes konnten die relevanten Applikationen des Google Play Stores in zwei Kategorien eingeteilt werden.

Informationssysteme

Auf dem Applikationsmarkt von Android konnten sich verschieden Anwendungen, die der Bereitstellung von Veranstaltungsinformationen dienen, etablieren. Diese sollen den Nutzer dabei unterstützen interessante Partys und Events zu finden. Die Anzahl der angebotenen Applikationen ist sehr hoch, diese unterscheiden sich hauptsächlich in der Art und der geographischen Lage der angebotenen Veranstaltungen. Zusätzlich bieten einige Applikationen weitere Features an. Im Folgenden werden vier der beliebtesten Anwendungen dieser Kategorie näher betrachtet.

Die von der BlindAd UG entwickelte App „Events und Partys“ ist ein reines Informationsportal, welches deutschlandweite Veranstaltungen auflistet.

Die Applikation „Virtual Nights” wirbt mit dem Slogan „ dein Nightlife-Guide für unterwegs“, neben der Bereitstellung von Informationsdaten zu Veranstaltungen werden hier auch Fotos der Events publiziert. Der Aspekt der Kommunikation wird nicht berücksichtigt.

Zwei weitere Vertreter dieser Kategorie von Software sind „Smobber“ und „Eventphant“.

Diese stellen ebenfalls Informationen zu Events bereit. Darüber hinaus wird das Teilen von Events auf Facebook und Twitter ermöglichen. Die Ausgewählte Veranstaltung wird gepostet und ist allen Kontakten ersichtlich. Eine direkte Anfrage an ausgewählte Personen ist dabei nicht vorgesehen. Weitere Details der Gestaltung des Eventbesuches wären hier über das Kommentieren des besagten Posts möglich. Dies wäre auch für alle Kontakte des sozialen Netzwerkes sichtbar. Angesichts der stetig wachsenden Community dieser Netzwerke ist die öffentliche Planung der Abendgestaltung fraglich.

Kommunikationssysteme

Die zweite Kategorie der Applikationen die für den Anwendungsbereich in Frage kommen würden sind die Kommunikationssysteme. Diese können die beteiligten Personen für den Austausch und der Abstimmung weiterer Einzelheiten verwenden. Hierfür eignen sich verschieden Systeme. Im Rahmen dieser Arbeit wird ein Vertreter sozialer Netzwerke und eine Chat-App näher betrachtet.

Das soziale Netzwerk Facebook bietet dem Nutzer viele verschieden Möglichkeiten zu kommunizieren und zu interagieren. Hierzu gehören unter anderem der Austausch von Nachrichten und das Posten von Inhalten. Des Weiteren gibt es die Möglichkeit Events zu erstellen. Zu diesen Events können Kontakte eingeladen werden. Im Rahmen dieses Personenkreises kann anschließend über Nachrichtenaustausch kommuniziert werden. Nachteile dieser Applikation ergeben sich zum einen aus dem manuellen Erfassungsaufwand, der mit dem Erstellen von Events verbunden ist und zum anderen mit dem Bedarf weiterer Anwendungen um Veranstaltungsinformationen zu erhalten. Auf Facebook werden auch öffentliche Events angeboten, hierbei handelt es sich um bereits erstellte Events die entsprechende Informationen zu einer Veranstaltung enthalten. Jedoch findet der Nachrichtenaustausch ebenfalls öffentlich statt und ist für alle sichtbar. Eine private Eventplanung ist hier nicht möglich. Zusammenfassend wird die in dem Anwendungsbereich geforderte Leistung durch diese Applikation nicht erfüllt.

Die Applikation „Whatsapp Messenger“ ist eine Chat Anwendung und ermöglicht neben dem Nachrichten-, Bild- und Videoaustausch auch den Gruppenchat. Dabei kann der Nutzer aus seinem Kontaktbuch Personen zu Einzel- oder Gruppenchats einladen. Die Kommunikation findet dabei live statt, das bedeutet eine abgesendete Nachricht wird den Empfängern in der Regel sofort zugestellt und diese können direkt darauf reagieren. Für den beschriebenen Anwendungsbereich stellt diese Art der Online-Kommunikation die ideale Lösung dar. Der Nachteil dieser Software besteht darin, dass der Aspekt der Informationsbereitstellung nicht berücksichtigt wird. Es werden keine Veranstaltungen publiziert, daher dient diese Applikation der reinen Kommunikation.

Im Rahmen der Recherche konnte keine Anwendung im Google Play Store gefunden werden, die der im Anwendungsbereich beschriebenen Anforderungen gerecht wird. Durch die vorhandenen Applikationen wird lediglich ein Workaround ermöglicht. Dies schließt den Einsatz mehrerer Anwendungen und manuellen Erfassungsaufwand seitens des Nut-

zers ein.

3.3 Grundfunktionen

Um die Grundfunktionalitäten der in dieser Arbeit entwickelten Applikation festzulegen, sollten die Schwachstellen bereits angebotener Anwendungen und das vorher beschriebene Szenario berücksichtigt werden.

Durch die im Google Play Store angebotenen Applikationen konnten die im Anwendungsbereich beschriebenen Anforderungen nicht erfüllt werden(siehe 3.2.2). Diese Apps dienen entweder der Informationsbereitstellung oder der Kommunikation. Im Rahmen der Rechercheauswertung wurden die Anwendungen daher in Informationssysteme und Kommunikationssysteme kategorisiert. Durch die erstgenannten Systeme erhält der Nutzer die Möglichkeit sich über Veranstaltungen zu Informieren. Einige der Anwendungen bieten auch weitere Funktionalitäten an, die jedoch für das Anwendungsgebiet irrelevant sind. Der Aspekt der Kommunikation wird von diesen Applikationen nicht berücksichtigt. Dies wird speziell von der zweiten Kategorie der in diesem Kontext relevanten Anwendungen unterstützt. Dabei reicht das Angebot von textuellem Nachrichtenaustausch, Sprachanrufen, Videoanrufen bis hin zu Chats. Diese Applikationen bieten dem Nutzer jedoch nicht die Möglichkeit sich über Veranstaltungen zu informieren und diese mit sozialen Kontakten zu teilen.

Für die Erfüllung der geforderten Anforderungen bedarf es einer Applikation, die neben der Bereitstellung von Veranstaltungsdaten auch eine effiziente Kommunikation der Nutzer gewährleistet. Dabei sollen die Vorteile beider Systeme vereint und ein Informations- und Kommunikationssystem entwickelt werden. Dem Nutzer wird dabei die Auskunft über Veranstaltungen ermöglicht, welche er direkt mit seinen Freunden teilen kann und über die sie sich sofort austauschen können. Besonderes Augenmerk liegt hier auf der Kommunikation, diese soll effizient unterstütz werden, vor allem bei steigender Anzahl der beteiligten Personen. Zusammengefasst sollte die Applikation folgende Grundfunktionen bereitstellen:

- Informationsbereitstellung
- Teilen dieser Information mit sozialen Kontakten
- Kommunikationsmöglichkeit via Chat

Die Abgrenzung der festgelegten Funktionalitäten von den bereits angebotenen Applikationen wird in Abbildung4 dargestellt. Dabei wurden die Ergebnisse der Recherche verwendet. Es werden hier ebenfalls die bereits benannten Applikationen als Vertreter der verschiedenen Systeme betrachtet. Diese werden in der Tabelle links aufgeführt, die für das Anwendungsgebiet bzw. dem Szenario geforderten Funktionalitäten stehen oben. Wird eine Funktion von einer Anwendung angeboten, ist an entsprechender Stelle der Tabelle ein „X“ aufgeführt. Aus der Tabelle geht hervor, dass keines der Applikationen alle Funktionen der zu entwickelnden Anwendung anbietet.

Anwendungen	Event Anzeige	Teilen von Events	Kommunikation mit Freunden
Virtual Nights	X		
Eventphant	X	X	
Smobber	X	X	
Events und Partys	X		
Whatsapp			X
Facebook		X	X

Abbildung 6: Abgrenzung der Funktionen

Durch die Integrierung der aufgeführten Funktionalitäten wird in zweierlei Hinsicht Nutzen generiert:

1. Der User wird bei der Planung des Besuches von Veranstaltungen unterstützt, wobei diese Unterstützung aus der Informationsbereitstellung und der Ermöglichung der Kommunikation mit sozialen Kontakten besteht. Dies erhält besondere Bedeutung wenn davon ausgegangen wird das der Veranstaltungsbesuch nicht alleine stattfinden soll. Daher ist die Leistung bestehender Systeme die ein reines Informationsportal darstellen ungenügend.

2. Durch die Bereitstellung von Veranstaltungsinformationen und der damit verbundenen Erreichbarkeit potenzieller Besucher, wird durch die Applikation Werbefläche angeboten. Veranstalter können ihre Events präsentieren und dadurch Kunden akquirieren.

Für die grobe Beschreibung des Grundgerüsts des zu entwickelnden Systems wurde ein Mock-up (siehe Abbildung 7&8) erstellt. Dieses stellt graphisch eine mögliche Umsetzung der Applikation dar. Die Verfügbaren Events werden in einer Liste dargestellt (siehe Abblidung 7). Durch auswählen der Elemente dieser Liste können weitere Details zu dem entsprechenden Event eingesehen werden. Dabei wird eine neue Ansicht mit der detaillierten Beschreibung der Veranstaltung geöffnet. Der Nutzer hat die Möglichkeit zurück zur Eventliste zu gelangen oder das entsprechende Event direkt mit seinen Freunden zu teilen. Hierfür wird eine Liste mit den verfügbaren Kontakten angezeigt. Aus dieser Liste können die gewünschten Personen ausgewählt werden. Nach Bestätigung der Eingabe wird ein Chatraum mit diesen Kontakten geöffnet, hier findet nun die Kommunikation statt. Des Weiteren hat der Nutzer die Möglichkeit die Eventdaten einzusehen oder sich die Liste mit den eröffneten Chatkonversationen (siehe Abbildung 8) anzeigen zu lassen.

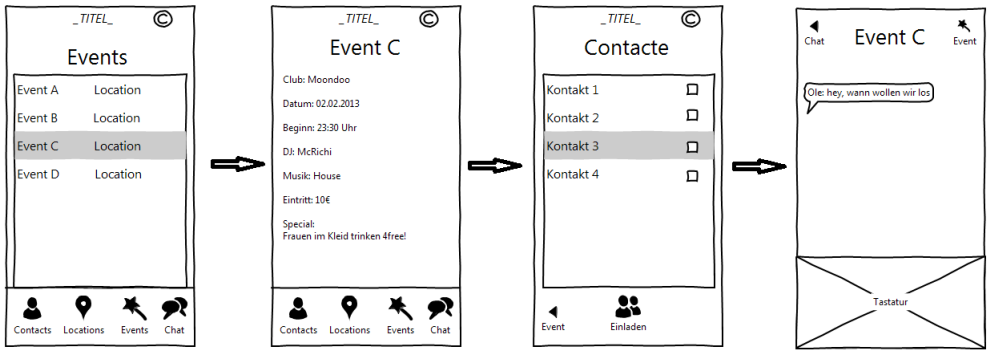

Abbildung 7: Mock-up Eventauswahl

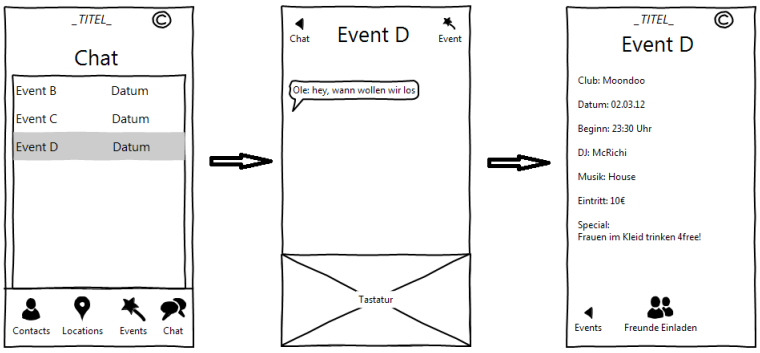

Abbildung 8: Mock-up Chat

4 Umsetzung

Um eine geeignete Umsetzung für eine Applikation zur Informationsbereitstellung und Kommunikation zu erreichen, sollten die vorher festgelegten Anforderungen(siehe Kapitel 3) und die notwendigen Grundlagen(siehe Kapitel 2) berücksichtigt werden.

Die Wahl fiel auf die Entwicklung einer nativen Applikation für das Betriebssystem Android (siehe Kapitel2). Für die Umsetzung der Kommunikation soll ein Chat integriert werden. Dadurch wird dem Nutzer die Möglichkeit einer Echtzeit-Konversation gegeben. Im Folgenden wird ein Überblick über die Programmarchitektur vermittelt, anschließend werden die Komponenten des Systems näher erläutert.

4.1 Programmarchitektur

Unter Berücksichtigung der in Kapitel 3 gestellten Anforderungen wird der Aufbau des Systems durch eine Client-Server-Architektur realisiert. Abbildung 9 stellt die wesentlichen Komponenten der Anwendung dar. Auf Clientseite werden die Daten durch das lokale Datenbanksystem SQLite gehalten. Der Server integriert ebenfalls eine Datenbank und stellt die Aufbewahrung globaler Daten sicher. Client und Server stehen miteinander in Verbindung und können Daten austauschen. Der Chat wird durch den kostenlosen Dienst Google Cloud Messaging(GCM) umgesetzt. Sowohl der Server als auch der Client sind mit dem Google-Server, der diesen Dienst bereitstellt verbunden.

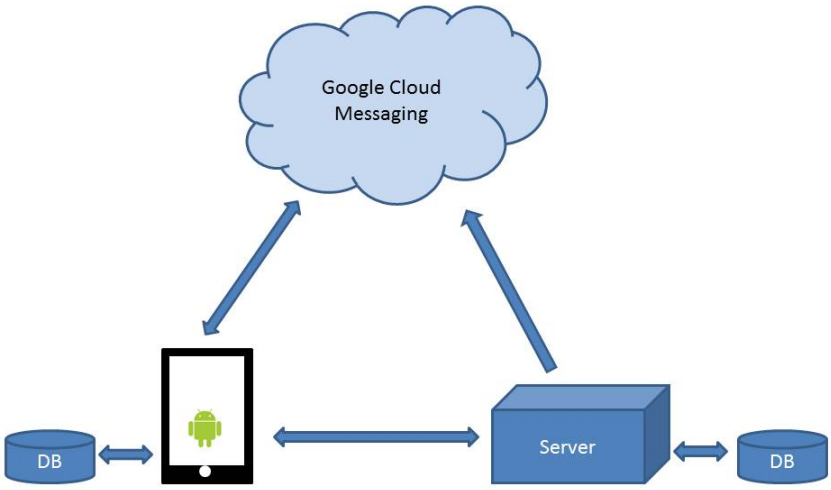

Abbildung 9: Anwendungsarchitektur

4.2 Chat Implementierung mittels GCM

Der von Google angebotene GCM-Dienst bietet die Möglichkeit Daten in der Größe von 4 Kilobyte an mobile Geräte zu verschicken. In dieser Arbeit wird dieser Dienst für die Umsetzung eines Chats verwendet.

Durch die Verwendung von GCM kann für die Nachrichtenübermittlung eine bestehende Verbindung zum Google-Server verwendet werden. Auf den Einsatz von jWebSocketes[4] kann verzichtet werden. Dadurch können zeitaufwendige Verbindungen zum Internet vermieden und Systemressourcen geschont werden. Die Nachrichten werden von GCM über ein Push-Verfahren zugestellt, somit wird die permanente Anfrage des Clients nach neuen Daten vermieden. Des Weiteren bietet das Push-System eine effiziente Nachrichtenqueue und stellt eine optimale Zustellung der Daten sicher. Dabei ist die Anzahl der Zustellversuche beliebig, d.h. ist das Zielgerät inaktiv, werden mehrere Zustellversuche unternommen. Im Folgenden wird die Implementierung von GCM näher betrachtet. Anschließend wird die Umsetzung eines Chats mittels dieser Technologie erläutert.

4.2.1 GCM Implementierung

Um Applikationen aus dem Google Play Store downloaden zu können wird ein Google-Account benötigt. Der Einsatz des GCM der einen solchen Account für die Verbindung mit dem Google-Server benötigt, wird daher als ermöglicht betrachtet. Für die Implementierung ist die GCM-Bibliothek (gcm.jar) erforderlich, diese muss auf Client- und Serverseite in den Classpath der Applikation eingefügt werden. Außerdem müssen einige Änderungen im Android-Manifest vorgenommen werden. Hier werden unteranderem die Zugriffsrechte der Anwendung festgelegt. Für die Verwendung von GCM werden Folgende Berechtigungen benötigt:

- Internetzugriffsberechtigung
- Empfangsberechtigung von GCM-Nachrichten
- Zugriffrecht auf den Google-Account
- Weckrecht des Prozessors wenn eine Nachricht empfangen wird

Die Nachrichten vom Google-Server werden vom Client durch einen Broadcast Receiver empfangen. Dieser muss ebenfalls im Manifest aufgeführt werden. Innerhalb des Intent-Filters werden die Nachrichten, auf die der Receiver reagiert, festgelegt. Dies sind entweder Benachrichtigungen vom Server oder die Registration-ID. Des Weiteren muss der GCM-Intent-Service dem Manifest bekannt gegeben werden. Um Nachrichten vom Server zu empfangen ruft der Broadcast Receiver den Intent-Service auf.

[4] http://jwebsocket.org/

Nach der Bearbeitung des Android-Manifestes können die einzelnen Komponenten verknüpft werden. Dazu muss die abstrakte Klasse `GCMBaseIntentService`, die in der GCM-Bibliothek enthalten ist implementiert werden. Im Konstruktor muss die Sender-ID, welche durch die Projektnummer dargestellt wird, der Methode `super()` übergeben werden. Die abstrakten Methoden, die implementiert werden müssen, sind folgende:

- `onRegistered(Context context, String regId)`
- `onUnregistered(Context context, String regId)`
- `onMessage(Context context, Intent intent)`
- `onError(Context context, String errorId)`
- `onRecoverableError(Context context, String errorId)`

Die Methode `onRegistered` erhält die Registration-ID, welche bei der Verbindung mit GCM vergeben wird. Bei der Abmeldung eines Gerätes von GCM wird die Methode `onUnregistered` aufgerufen. Dieses Gerät ist hiernach für keine weiteren Nachrichten erreichbar. Besondere Bedeutung erhält die Methode `onRegistered`onMessage, hier werden die versendeten Daten durch einen Intent empfangen und können weiter verarbeitet werden. Für die Fehlerbehandlung sind die Methoden `onError` und `onRecoverableError` zuständig. (vgl. Google [2012])

Um die Registration-ID vom Google-Server zu erhalten und das Gerät somit beim GCM zu registrieren muss dies in einer Activity angestoßen werden. Durch Import des GCM-Registrar kann mittels der Methode `register()` eine Registration-ID beantragt werden. Eine mögliche Umsetzung könnte wie folgt aussehen:

```
GCMRegistrar.register(this, SENDERID)
```

Anschließen muss diese Registration-ID dem Applikationsserver übermittelt werden. Dieser kann mittels der GCM-Technologie Daten an den Client senden. Hierfür bietet die GCM-Bibliothek die Methode `send()` an. Dabei wird neben der Nachricht eine Empfängerliste übergeben. Dadurch können Nachrichten an verschiedene Geräte zeitgleich versendet werden.

```
MulticastResult result = sender.send(message, devices, 5)
```

4.2.2 GCM als Chat

Wie im vorhergehenden Abschnitt beschrieben, können nun Nachtrichten vom Server an alle mobilen Geräte, auf denen die Applikation installiert ist, versendet werden. Für die Umsetzung des Chats mittels dieser Technologie müssen die Registration-IDs aller Geräte auf dem Server gespeichert werden. Außerdem werden die zu versendeten Nachrichten vom Client erstellt, d.h. der Server dient hier nur der Vermittlung zwischen dem Client

und dem Google-Server. Die in der Datenbank des Servers gespeicherten Registration-IDs müssen dem Client verfügbar gemacht werden. Dabei sollen nur die verfügbaren Kontakte des Nutzers angezeigt werden. Beim Versenden von Nachrichten wird neben der Mitteilung, eine Liste von Kontakten übergeben die diese Benachrichtigung erhalten sollen. Beides wird vom Server empfangen und mittels der in Abschnitt 4.2.1 beschriebene Methode `send()` an den Google-Server weitergeleitet. Durch die Übergabe einer List von Registration-IDs womit mehrere Personen zeitgleich benachrichtigt werden können, wird ein Gruppenchat ermöglicht.

Die versendeten Nachrichten müssen anschließend durch den Broadcast Receiver empfangen werden. Dies erfolgt in der von `GCMBaseIntentService` erbenden Klasse `GCMIntentService`. Hier wird die Nachricht durch die Methode `onMessage()` in Form eines Intents empfangen. Daraufhin wird die Nachricht im Chatverlauf eingetragen und der Nutzer durch entsprechende Noticikations auf die Neuigkeit aufmerksam gemacht.

4.3 Client

Die Clientseite wird durch die mobilen Geräte auf denen die Applikation ausgeführt wird dargestellt. Diese bilden die Schnittstelle zum User und enthalten die Benutzeroberfläche. Da Android derzeit in verschiedenen Versionen[5] verfügbar ist, welche oft nicht aufwärts kompatible sind, fiel die Wahl auf Android 2.2 und dem API-Level 8. Durch diesen tief gewählten Build Target werden die erforderten Tools noch bereitgestellt und sichergestellt das die Applikation auf möglichst viele Geräten lauffähig ist.

Android Anwendungen setzen sich aus betriebssystemspezifischen Komponenten und Java-Klassen zusammen. Um einen Überblick über die Clientseite der Applikation zugeben, werden diese Bestandteile näher betrachtet. Zuvor wird der Verbindungsaufbau zum Server beschrieben.

4.3.1 Verbindung zum Server

Socket

Für die Verbindung zu einem Server bietet Java die beiden Klassen `java.net.Socket` und `java.net.ServerSocket` an. Die erst genannte Klasse wird vom Client importiert, dadurch wird die Erzeugung eines neuen Sockets ermöglicht. Diesem Socket wird eine IP-Adresse und ein Port übergeben, wodurch die Verbindung zum Server aufgebaut wird. Auf Serverseite müssen beide Klassen importiert werden. Es wird ein neuer `ServerSocket` erzeugt, dieser soll den Socket des Clients empfangen. Hierfür muss dem `ServerSocket` der gleiche Port übergeben werden. Dieser ist nun für einen Verbindungsaufbau bereit. Durch die Methode `accept()` blockiert der `ServerSocket` bis sich ein Client angemeldet hat. Nach dem Verbindungsaufbau können Daten zwischen Client und Server ausgetauscht werden.

[5] neuste Version Android 4.2 JellyBean

Der Begriff Socket stammt aus dem Englischen und bedeutet Steckdose. Diese Benennung kann auf die Funktionsweise des Nachrichtenaustauschs mittels des Sockets zurückgeführt werden. Ähnlich wie bei dem Stecker der in die Steckdose eingeführt wird gibt es zwei Kanäle, einen für das Empfangen und einen für das Abgeben von Daten bzw. Elektrizität. Durch die Instanziierung eines `BufferedReader` wird der Datenempfang ermöglicht. Dieser blockiert solange bis die Nachricht empfangen wurden.

```
BufferedReader in = new BufferedReader(new InputStreamReader(
socket. getInputStream()));
```

Mittels der Methode `readLine()` können einzelne Zeilen des Inputs ausgelesen werden. Diese können anschließend als String weiter verwendet werden. Für das Versenden von Daten wird der `PrintWriter` eingesetzt

```
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
out.println("Nachricht");
```

Nach der Instanziierung kann unter Verwendung der Methode `println()` der Datenversand erfolgen. Bei diesen Daten handelt es sich ebenfalls um Strings. Es können Daten vom Client mittels des `PrintWriters` versendet und vom Server über den `BufferedReader` empfangen werden. Auf dieselbe Weise kann der Server antworten. Dadurch wird ein Datenfluss ermöglicht. Die Kommunikation zwischen Server und Client findet über diese beiden Kanäle statt. Abschließend kann der Server über die Methode `close()` beendet werden.

```
serverSocket.close();
```

AsyncTask

In der entwickelten Applikation stellen die Android-Geräte die Clientseite dar. Der Verbindungsaufbau wird auf den mobilen Endgeräten gestartet. Da solch eine Verbindung und der damit verbundene Datenaustausch viel Zeit in Anspruch nimmt, sollte dies nicht direkt in einer Activity ausgeführt werden. Dies hätte unangenehme Pausen für den Nutzer zur Folge, da die Applikation in dieser Zeit keine weiteren Aktionen ausführt.

Zeitintensive Aktionen wie die Kommunikation mit dem Server sollten daher nebenläufig und unabhängig vom Ablauf der Activity durchgeführt werden. Dazu kann der UI-Thread, auf dem die Activity ausgeführt wird, verschiedene Worker-Threads aufrufen. Auf diesen erfolgt die Ausführung nebenläufiger Aktionen. Um den Umgang mit Threads zu erleichtern bietet Android die Klasse `AsyncTask` an. In dieser Klasse können zeitintensive Aktionen einer Aktivity ausgelagert werden. Die Ausführung dieser Klasse

erfolgt asynchron zu dem der Aktivität. (Hamburg [2012])

In dieser Arbeit erfolgt die Kommunikation mit dem Server über Socket-Verbindungen die asynchron ausgeführt werden. Dazu wird in den entsprechenden Activities ein Objekt vom Typ `AsyncTask` erzeugt. Dies geschieht durch eine innere Klasse innerhalb der Activity. Das Versenden einer Nachricht innerhalb des `AsyncTask` sieht dabei wie folgt aus:

```
new AsyncTask<JMessage, Void, Void>(){

	@Override
	protected Void doInBackground(JMessage... message) {
		try{
			Gson gson = new Gson();
			String json = gson.toJson(message[0]);

			ClientSocket socket = new ClientSocket();
			socket.Message(json);

			}
		catch(Exception e){
			e.printStackTrace();
			}
		return null;
	}
}.execute(new JMessage[]{message});
```

Der neuen Instanz von AsyncTask können dabei drei generische Typen übergeben werden. Dies wären:

- Params, Typ der Parameter die der aynchronen Aufgabe mitgegeben werden
- Progress, Typ der Fortschritteinheiten
- Result, Wiedergabetyp

In der Methode `doInBackround()` wird die Aufgabe, die nebenläufig ausgeführt werden soll, definiert. Hier findet die Kommunikation mit dem Server statt. Ist dies abgeschlossen, übergibt die Methode `onPostExecute()` den Rückgabewert der ausgeführten Aufgabe an die Activity. (vgl.Google [2012])

4.3.2 Java Klassen

ClientSocket

Der Verbindungsaufbau zum Server erfolgt in der Klasse `ClientSocket`. Hier werden verschiedene Methoden angeboten. Diese können aus den entsprechenden Activities aufgerufen werden. Der Aufruf erfolgt dabei aus dem `AsyncTask` der Aktivität. Die angebotenen Methoden liefern entweder Daten vom Server oder übergeben diesem neue Inhalte. Dazu wird ein neuer Socket geöffnet, die benötigte IP und der Port sind als globale Variablen deklariert. Für den Datenaustausch mit dem Server wird die Java-Bibliothek Gson verwendet. Diese ermöglicht es Java Objekte in JSON Strings zu serialisieren und umgekehrt

zu deserialisieren. Zwischen Client und Server werden diese JSON Strings ausgetauscht. Hierfür muss zunächst eine neue Instanz von Gson erzeugt werden. Durch die Methode `toJson()` wird die Serialisierung von Objekten ermöglicht, diese werden der Methode übergeben. Der hierdurch entstandene String kann über den `PrintWriter` an den Server versendet werden. Die Deserialisierung erfolgt durch die ebenfalls von Gson angebotene Methode `fromJson()`. Diese benötigt den zu deserialisierenden JSON String und die entsprechende Java-Klasse, in die der String transformiert wird. Bei diesen Java-Klassen handelt es sich im Rahmen der entwickelten Applikation um `JavaBeans`. (vgl. Wilson)

JavaBeans

Für den Austausch von Daten mit dem Server werden Datencontainer benötigt. Diese werden durch `JavaBeans` realisiert. Bei diesen Software-Komponenten handelt es sich um Behälter für die Datenübertragung. Um beispielsweise Eventdaten vom Server zu erhalten, werden die serialisierten Informationen empfangen und durch die Methode `fromJson()`, unter Angabe des `JavaBeans JEvents` deserialisiert. Umgesetzt werden diese Klassen durch die Implementierung des Interface `java.io.Serializable`. Außerdem haben diese Klassen folgende Eigenschaften:

- Private deklarierte Attribute
- Leerer Konstruktor
- Public deklarierte Getter-/Setter-Methoden

Dadurch wird zum einen die Übertragung und zum anderen die effiziente Verarbeitung der Daten sichergestellt. Die Instanziierung neuer Objekte dieser Beans ermöglicht das Füllen von Informationen über die Setter-Methoden und das Abfragen dieser durch die Getter-Methoden. Die Übertragung mehrerer Events ist anschließend durch die Implementierung einer `ArrayList` möglich. Dies sieht wie folgt aus:

```
public class JEventlist extends ArrayList<JEvents>
```

4.3.3 LocalDB

Für die lokale Datenhaltung wird eine SQLite Datenbank benötigt. Diese soll unter anderem Angaben zum Nutzer und den Chatverlauf speichern. Dadurch wird die Aufbewahrung der Daten über die Lebensdauer der Anwendung ermöglicht. Android bietet hierzu die abstrakte Klasse SQLiteOpenHelper an. Diese gibt zwei abstrakte Methoden vor:

- `onCreate`
- `onUpdate`

In der `onCreate` Methode, die eine SQLiteDatenbank übergeben bekommt, können über SQL-Befehle Tabellen erzeugt oder gelöscht werden. Für das Ausführen dieser Befehle wird die Methode `execSQL()` angeboten. Um Änderungen an Tabellen vorzunehmen

kann `onUpdate` verwendet werden. In dieser Methode wird die alte Version der Tabelle gelöscht und die bearbeitete Version tritt an deren Stelle. Die Klasse `ClubcheckerLocalDB` erbt von der abstrakten Klasse `SQLiteOpenHelper`, durch die Implementierung derer Methoden wird das Datenbankschema festgelegt. Anschließend ist die Bearbeitung der Datenbank durch Erzeugung einer neuen Instanz der `ClubcheckerLocalDB` möglich. Die Daten können dabei entweder nur ausgelesen oder auch bearbeitet werden. Der erstgenannte Zugriff erfolgt über die Methode `getReadableDatabase()` und der zweite über `getWriteableDatabase()`. Daraufhin können SQL-Befehle über SQLiteStatements ausgeführt werden.

4.3.4 GCMIntentService

Um Nachrichten über den GCM-Dienst erhalten zu können, muss wie in Abschnitt 4.2.2 beschrieben, eine Subklasse von `GCMBaseIntentService` implementiert werden. Die Daten werden in der `onMessage()` Methode empfangen und entsprechend verarbeitet. Hier wird beispielsweise der Chatverlauf gepflegt. Dazu wird die als JSON String empfangene Nachricht deserialisiert, dies erfolgt unter Angabe des entsprechenden `JavaBean` `JMessage`. Anschließend muss eine Verbindung zur lokalen DB aufgebaut und die empfangenen Daten eingetragen werden.

4.3.5 DiscussArrayAdapter

Der durch den GCM-Dienst empfangene und in der lokalen Datenbank gespeicherte Chatverlauf muss anschließend dem Nutzer geeignet präsentiert werden. Hierfür wurde die Klasse `DiscussArrayAdapter` entwickelt. Diese erbt von der abstrakten Klasse `ArrayAdapter`, welche über `Arrays` operiert und deren Inhalte mit entsprechenden Views darstellt. Bei diesen Elementen handelt es sich um Chatnachrichten, die durch die Klasse `TextInhalt` erzeugt werden. Hier werden die Nachricht, der Name des Absenders und die Ausrichtung im Verlauf angegeben:

```
public TextInhalt(boolean left, String comment, String name) {
      super();
      this.left = left;
      this.comment = comment;
      this.name = name;
}
```

Der `DiscussArrayAdapter` stellt anschließend die Verknüpfung dieser Angaben mit den entsprechenden Views des XML-Layouts sicher. Dazu wird eine `ArrayList` erstellt, in der die Elemente durch die Klasse `TextInhalt` dargestellt werden. Über die Methode `add()` können dieser Liste Elemente hinzugefügt werden. Die Inhalte dieser Liste sollen in einem Layout angezeigt werden. Hierzu dient das Layout `listitem`. Dieses schließt in einem `LinearLayout` zwei `TextViews` ein. Durch den `DiscussArrayAdapter` können die Inhalte den entsprechenden `TextViews` zugeordnet werden. Der boolean Wert der Klasse `TextInhalt` ermöglicht es empfangene Nachrichten von selbst geschriebenen zu

differenzieren. Dies wird im Verlauf durch die Verwendung unterschiedlicher Hintergrundgraphiken und die Ausrichtung des Textes verdeutlicht.

4.3.6 Activities

Für die Verwaltung der Benutzeroberfläche und die Programmlogik sind Activities zuständig. Um die Funktionsweise der in dieser Arbeit entwickelten Activities zu erläutern, wird zunächst Ihr Zusammenspiel umrissen, anschließend werden wichtige Quellcodestellen näher betrachtet.

Gestartet wird die Applikation mit der `EventActivity`, bei erstmaligem Aufruf der Anwendung werden der Name und die Telefonnummer des Nutzers aufgenommen und in der globalen Datenbank gespeichert. In dieser Activity erfolgt ebenfalls die Anzeige der vorhandenen Events. Dies wir durch eine Datenbankabfrage umgesetzt, deren Ergebnisse werden mittels eines Adapters an die Views der angezeigten Liste gebunden. Die Elemente dieser Liste können ausgewählt werden und führen zur Detailansicht des entsprechenden Events. Die Daten bezüglich des Events werden dabei über einen Intent der aufgerufenen Activity übergeben.

Die `EventDetailActivity` ist nun für die Anzeige der Detaillierten Ansicht zuständig. Hierfür werden die erhaltenen Daten an die zugehörigen Views gebunden. Über einen Button können Kontakte zu dieser Veranstaltung eingeladen werden. Dazu wird eine Liste mit den verfügbaren Kontakten angezeigt, diese können ausgewählt und benachrichtigt werden. Die Benachrichtigung erfolgt dabei über den GCM-Dienst.

Die verfügbaren Kontakte werden in der `ContactActivity` angezeigt. Da die Applikation ohne aufwendige Anmeldung und zusätzlicher Kontaktverwaltung verwendbar sein soll muss bei der Anzeige verfügbarer Kontakte für eine entsprechende alternative gesorgt werden. Dem User soll aus der gesamten Menge der Personen die diese Anwendung nutzen, nur diejenigen Angezeigt werden die Ihm bekannt sind. Hierzu wird eine Liste mit den Kontakten die auf dem Gerät gespeichert sind an den Server geschickt. Dieser gleicht die Telefonnummern mit denen der Datenbank ab und antwortet mit einer Liste der Übereinstimmungen.

Die gestarteten Chatunterhaltungen werden in der `ChatListActivity` angezeigt. Die einzelnen Chats werden hier in einer Liste aufgeführt. Die Informationen zu den Chaträumen werden lokal in der SQLite Datenbank gespeichert. Um zu dem Verlauf zu gelangen und Nachrichten zu versenden können die Elemente dieser Liste ausgewählt werden. Daraufhin öffnet sich die Chatansicht, diese wird durch die `ChatActivity` realisiert. Der Verlauf der Unterhaltung wird ebenfalls in der lokalen Datenbank gespeichert. Im Folgenden werden die `ChatActivity` und die `EventActivity` näher erläutert. In diesen Aktivitäten erfolgt die Umsetzung der Schlüsselfunktionen dieser Applikation.

EventActivity

In der `onCreate`-Methode erfolgt zunächst die Prüfung der Registration-ID des GCM-Dienstes. Hierfür wird die von `GCMRegistrar` angebotene Methode `getRegistrationId()` verwendet. Dadurch erhält der Google-Server eine Anfrage nach der Registration-ID des Gerätes. Liegt keine ID vor antwortet der Google-Server mit einem leeren String. In diesem Fall wird über die Methode `register()` eine neue ID beantragt. Dabei müssen der Kontext und die Sender-ID des GCM-Projektes übergeben werden.

```
//GCM properties
String SENDER_ID = "";
String TAG = "GCMRegistrar";
regId = GCMRegistrar.getRegistrationId(this);

GCMRegistrar.checkDevice(this);
GCMRegistrar.checkManifest(this);

     if (regId.equals("")) {

           GCMRegistrar.register(this, SENDER_ID);
           setUser();

        } else {
          Log.v(TAG, "Already registered");
        }
```

Des Weiteren erfolgt die Eintragung des Users in die Datenbank, da es sich um eine neue Anmeldung handelt. Hierfür wird die `setUser()` Methode verwendet. Diese übergibt den Programmfluss an eine Hilfs-Activity, welche eine Eingabemaske für die Userdaten bereitstellt. Nach der Eingabe erhält die `EventActivity` die Steuerung und die Daten in der `onActivityResult()` Methode. Hier wird in einem `AsyncTask` die Verbindung zum Server aufgebaut um die Userdaten zu übergeben. Diese werden anschließend in die Datenbank aufgenommen. Die Anmeldung erfolgt nur bei der ersten Ausführung der Applikation, bei wiederholten starten antwortet der Google-Server mit der entsprechenden Registration-ID.

Nach dem `if`-Block in der die Anmeldung geprüft und ausgeführt wird müssen die verfügbaren Events angezeigt werden. Dies erfolgt über die `getEventlist()` und `setEventlist()` Methoden. In der erstgenannten Methode wird in einem `AsyncTask` die Verbindung zum Server aufgebaut und die verfügbaren Eventdaten importiert. Anschließend wird die `setEventlist()` Methode ausgeführt. Diese erhält die eingeholten Daten und stellt sie über einen `SimpleCursorAdapter` in einer Liste dar. Über einen `OnItemClickListener()` können die Elemente der Liste ausgewählt werden. Die anschließend ausgeführte EventDetailActivity erhält die Informationen zu der gewählten Veranstaltung über einen Intent.

ChatActivity

Die einzelnen Chaträume werden in einer Liste der `ChatListActivity` angezeigt. Durch das Auswählen eines der Elemente wird die `ChatActivity` geöffnet. Diese ist für die Anzeige des entsprechenden Chatverlaufs und für das Senden von Nachrichten zuständig. Die

empfangenen und gesendeten Kommentare einer Unterhaltung werden lokal gespeichert. Für die Bearbeitung dieser Daten muss der Zugriff auf die lokale Datenbank eingerichtet werden. Hierfür muss eine neue Instanz der `ClubcheckerLocalDB` erzeugt und ein Lese- und Schreibzugriff erstellt werden. Dies erfolgt über die in Abschnitt 4.3.3 beschriebenen Methoden. Anschließend kann der Verlauf über entsprechende SQL-Befehle abgefragt werden. Die dadurch erhaltenen Daten müssen anschließend in einer geeigneten Weise dargestellt werden. Hierfür wird der `DiscussArrayAdapter` verwendet (siehe Abschnitt 4.3.5). Dieser ermöglicht die Anzeige der Daten in Form eines Chatverlaufes. Dabei wird zwischen empfangenen und gesendeten Nachrichten unterschieden. Der Name des Senders wird ebenfalls angezeigt. Die Daten werden in einer Liste angezeigt, deren Elemente durch das Layout `listitem` definiert sind. Die Bindung der Daten an die zugehörigen Views erfolgt über den Adapter.

Das Senden von Nachrichten ist zum einen über den Send-Button und zum anderen über die Entertaste möglich. Für letzteres wurde ein `onKeyListener` implementiert, der auf Eingaben in Verbindung mit der Entertaste reagiert. Anschließend wird die Methode `sendMessage()` ausgeführt. Diese nutzt die bereits geöffnete Verbindung zur lokalen Datenbank und trägt die neue Nachricht in den Verlauf ein. Außerdem wird der Name des Users, welcher in einer anderen Tabelle der Datenbank gespeichert ist abgefragt. Über einen `AsyncTask` erfolgt der Aufbau einer Verbindung zum Server. Es wird eine neune Instanz von `JMessage` erzeugt, diese erhält die Nachricht, den Namen des Absenders und die Liste der Kontakte, die benachrichtigt werden sollen. Um diese nun dem Server zu schicken wird die `JMessage`-Instanz in einen JSON String transformiert. Letztendlich wird der String über den `ClientSocket` an den Server geschickt, welcher diesen an den GCM-Dienst weiterleitet.

4.4 Server

Der Applikationsserver stellt die globalen Daten für den Client bereit. Außerdem ist dieser mit dem Google-Server verbunden und ermöglicht den Nachrichtenaustausch. Für die Speicherung der Daten wurde eine Datenbank aufgesetzt. Um die Funktionsweise des Servers näher zu erläutern werden die folgenden beiden Klassen betrachtet:

- `MySQLConnector`
- `RequestHandler`

Dabei werden die Informationen aus den vorhergehenden Abschnitten vorausgesetzt.

4.4.1 Verbindung zur Datenbank

Für die Speicherung der globalen Daten der Applikation wurde eine MySQL-Datenbank aufgesetzt. Um auf diese zugreifen zu können, wird ein entsprechender Treiber benötigt. Hierfür stellt Java die Datenbankschnittstelle JDBC (Java Database Connectivity) zur Verfügung. Diese ermöglicht einen effizienten Zugriff auf die Datenbank. Dazu wird die Java-Bibliothek `mysql-connector.jar` angeboten. Durch das Einfügen dieser Library in

den Java Build Path des Servers kann der Datenbanktreiber verwendet werden. Dieser wird in der `Main`-Methode der `MySQLConnector` Klasse geladen:

```
// Open a connection
  Class.forName("com.mysql.jdbc.Driver");
  conn = DriverManager.getConnection(url,user,pass);

// ServerSocket
   ccSocket();

// Clean environment
   conn.close();
```

Die Methode `getConnection()` erhält dabei die benötigten Serverdaten. Für eine entsprechende Fehlerbehandlung muss gesorgt werden ebenso für die Schließung der Verbindung zur Datenbank. Nach dem Verbindungsaufbau wird die Methode `ccSocket()` aufgerufen. Hier wird der vom Client gesendete Socket empfangen. Dazu wird ein neuer `ServerSocket` erstellt, der solange blockiert bis sich ein Client anmeldet. Der Server ist nun für Anfragen des Client erreichbar und wartet auf den Eingang eines Sockets. Tritt dies ein wird die Klasse `RequestHandler` gestartet, diese wird im folgenden Abschnitt näher erläutert.

4.4.2 Verarbeitung der Anfragen

Nach dem Aufbau der Verbindung zur Datenbank und der Implementierung des ServerSockets, steht der Server für Anfragen des Client bereit. Für die Verarbeitung dieser Anfragen wurde die Klasse `RequestHandler` implementiert. Diese wird nach dem Eingang eines Client-Sockets aufgerufen. Dabei soll die Bearbeitung mehrerer Anfragen nebenläufig möglich sein, da der Server mehreren Clients zur Verfügung steht. Ermöglicht wird dies durch die von Java bereitgestellten Klasse `Thread`. Der `RequestHandler` erbt von dieser Klasse. Dabei muss die Methode `public void run() {}` implementiert werden, welche die nebenläufig auszuführende Verarbeitungslogik enthält. Für jeden eingehenden Aufruf des Servers wird eine neue Instanz des `RequestHandlers` erzeugt, welche sich um die Bearbeitung genau dieser Anfrage kümmert. Durch das Erben der Klasse `Thread`, wird jede Instanz auf einem neuen Thread nebenläufig ausgeführt. Beim Eingang einer Anfrage wird der empfangene Socket und die geöffnete Datenbankverbindung einer neuen Instanz des `RequestHandlers` übergeben.

```
// Start RequestHandler
      while(listening){
            new RequestHandler(serverSocket.accept(), conn).start();
```

Die vom Client versendete Anfrage wird durch den Socket gehalten. Die Auslesung dieser Daten wird durch einen `BufferedReader` realisiert. Anschließend erfolgt eine Fallunterscheidung um welche Art Anfrage es sich handelt. Dabei ist der empfangene String bei der Wahl der auszuführenden Operation auschlaggebend. Durch die ebenfalls erhaltene Datenbankverbindung conn können Datenbankabfragen vorgenommen werden. Dies erfolgt über herkömmliche SQL-Statements.

Neben Datenbankoperationen findet hier auch die Kommunikation mit dem Google-Server statt. Dabei wurde die in 4.2 beschriebene Technologie verwendet. Für das Versenden von Chatnachrichten wird nach der Fallunterscheidung der JSON String deserialisiert. Dies erfolgt unter Angabe der Klasse `JMessage`. Der Nachrichtentext und die Liste der Kontakte, die diese Nachricht erhalten sollen, sind nun über die Methoden der genannten Klasse zugreifbar. Anschließend wird eine neue Instanz der von GCM bereitgestellten Klasse `Sender` erzeugt. Diese erhält den API-Schlüssel zur Authentifizierung und ermöglicht das Versenden der Nachricht durch die Methode `send()`.

Letztendlich wird abhängig von der Anfrage eine Antwort über den `PrintWriter` geschrieben und der Socket und die Verbindung zur Datenbank geschlossen.

5 Evaluierung

Gegen Ende der Arbeit sollen die in Kapitel 3 festgelegten Grundfunktionalitäten durch eine Befragung evaluiert werden. Dabei soll festgestellt werden, ob die entwickelte Applikation ihren Zweck erfüllt und an welchen Stellen Verbesserungsbedarf besteht. Im Folgenden wird diese Evaluierung näher erläutert. Zunächst erfolgt eine Beschreibung der Testpersonen, anschließend werden die Befragung und der verwendete Fragebogen unter die Lupe genommen.

5.1 Testpersonen

Um produktive Erkenntnisse aus der Evaluierung zu erlangen, sollten die ausgewählten Testpersonen der Zielgruppe entsprechen. Da die Anwendung auf das Planen und Organisieren von Veranstaltungsbesuchen ausgerichtet ist und die Datenbank zunächst nur Informationen über Partys in Clubs und Bars enthält, wird von einer Zielgruppe zwischen 18 und 28 Jahren ausgegangen. Daher wird die Gruppe der Testpersonen von Studenten und Jugendlichen dieser Altersklasse gebildet. Genauere Angaben bezüglich der Befragten Person können aus dem Fragebogen entnommen werden (Anhang B).

5.2 Gestaltung der Befragung

Im Rahmen der Evaluierung wird den Testpersonen ein Szenario beschrieben und Aufgaben gestellt, die den Einsatz der entwickelten Applikation erfordert. Ausgehend von diesem Szenario sollen die Aufgaben mittels der Anwendung bearbeitet werden. Anschließend füllen die Testpersonen einen Fragebogen aus. Anhand der gestellten Fragen soll die Usability und der Nutzen der Applikation ermittelt werden. Der Fragebogen ist an der ISO-Norm 9241/10 zur Beurteilung von Software gelehnt. (Prümper [1993]) Diese Vorlage bietet eine ideale Unterstützung der Gestaltung des Evaluierungsbogens. Anhand der vorgeschlagenen Kriterien und Fragen wurde ein Fragebogen der den Anforderungen dieser Arbeit entspricht entwickelt (siehe Anhang A2). Dabei wird auch die Relevanz des Anwendungsbereiches für den Nutzer gemessen. Außerdem erfolgt ein Vergleich der entwickelten Applikation zu bestehenden Anwendungen.

Es nehmen 12 Personen an der Befragung Teil. Diese werden in vier Gruppen mit jeweils 3 Probanden eingeteilt. Das Testen der Applikation erfolgt gruppenweise. Die Anwendung ist dabei auf drei verschiedenen Smartphones installiert. In jeder Gruppe erhält ein Teilnehmer die Aufgabe eine Veranstaltung aus zu suchen und die anderen beiden Testpersonen einzuladen. Diese haben die Aufgabe sich über das vorgeschlagene Event zu informieren und anschließend auszutauschen. Die Teilnehmer haben die Möglichkeit auf eine andere Veranstaltung auszuweichen. Das übergreifende Ziel ist es sich zu informieren und zu kommunizieren. Der Ausgangspunkt dabei ist ein Szenario, in dem die Teilnehmer gemeinsam am Wochenende eine Veranstaltung besuchen möchten.

Nach der Bearbeitung der Aufgaben erhalten die Testpersonen jeweils einen Fragebogen.

Bei der Beantwortung der Fragen können die Probanden weiterhin auf die Applikation zugreifen und ihre Bewertung abgeben. Der Evaluationsbogen besteht aus zwei Teilen, wobei der erste sich auf die Usability der Anwendung bezieht. Hier werden folgende Kriterien bewertet:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Lernförderlichkeit

(Prümper [1993])
Im letzteren Teil soll die Relevanz des Anwendungsfalls, für den die Applikation entwickelt wurde, gemessen werden. Des Weiteren erfolgt eine Abgrenzung der App von bestehender Software für diesen Anwendungsbereich.

5.3 Ergebnisse

Bei der Bewertung der Usability der Applikation haben die Probanden jede Ausprägung mit einer von sieben möglichen Werten beurteilt. Dabei reicht die Skala von einer sehr negativen Bewertung bis hin zu einer sehr positiven Bewertung. Für die Darstellung der Ergebnisse wurde die aufgeführte Grafik erstellt. Diese soll das Gesamtergebnis der Usability-Evaluierung präsentieren.

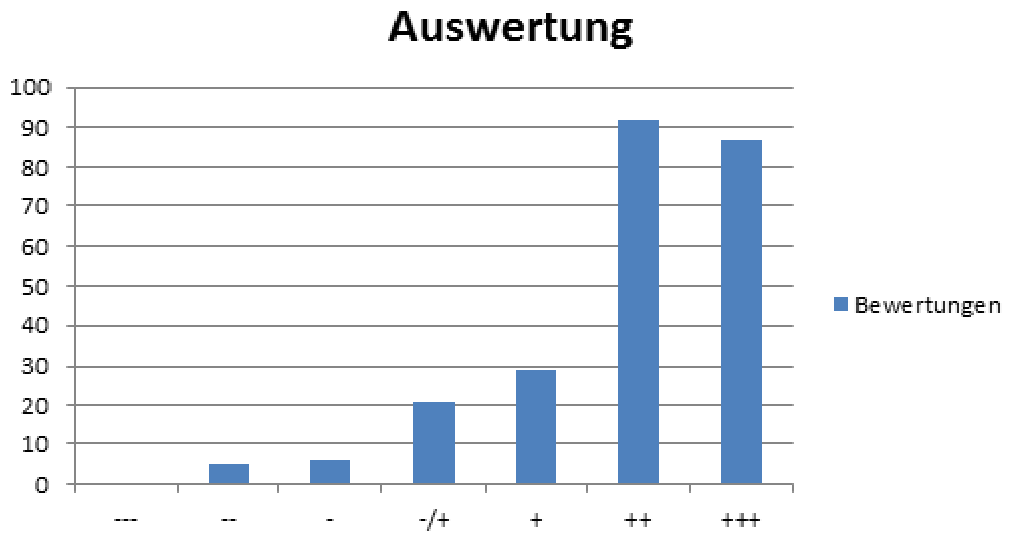

Abbildung 10: Auswertung der Befragung

Im Folgenden werden die Werte der Beurteilung mit Punkten auf einer Skala von eins bis sieben gleichgesetzt. Eine sehr negative Bewertung ist dabei eine eins und eine sehr positive eine sieben. Die Bewertung fiel überwiegend positiv aus. Die Testpersonen haben 20 Ausprägungen bewertet. Von der Gesamtmenge der Bewertungen sind 36 Prozent sehr positiv ausgefallen, weitere 38 Prozent wurden mit sechs Punkten bewertet. 12 Prozent der Bewertungen fiel mit 5 Punkten aus. Nur 14 Prozent der Bewertungen fiel mit 4 Punkten oder schlechter aus. Es wurde keine Ausprägung sehr negativ, also mit nur einem Punkt bewertet. Die durchschnittliche Bewertung liegt bei 5,9 Punkten. Die einzelnen Kriterien wurden wie folgt bewertet:

Kriterium	Durchschnittliche Bewertung (Punkte)
Erwartungskonformität	5,88
Lernförderlichkeit	6,00
Steuerbarkeit	5,02
Selbstbeschreibungsfähigkeit	6,03
Aufgabenangemessenheit	6,63

Aus der Tabelle geht hervor, dass vor allem die Aufgabenangemessenheit besonders gut bewertet wurde. Ebenso kann festgestellt werden, dass die Kriterien der Erwartungskonformität und Steuerbarkeit im Vergleich zu den anderen Kriterien schlechter ausgefallen sind.

Die weiteren Ergebnisse des Fragebogens sollen die Relevanz des Anwendungsbereiches, für den die Applikation entwickelt wurde, verdeutlichen. Die Probanden sollen angeben wie oft sie der Aufgabe der Organisation eines Veranstaltungsbesuches gegenüber stehen. Dabei wurde von sechs Teilnehmern einmal die Woche angegeben. Bei weiteren drei Probanden war es zwei- bis dreimal im Monat. Zwei Testpersonen gaben an lediglich einmal im Monat dieser Aufgabe gegenüber zu stehen und eine Person niemals.

Des Weiteren sollen die Probanden angeben welche Applikationen sie gewöhnlich für den Anwendungsbereich verwenden. Dabei wurden überwiegend mehrere Anwendungen benannt. Lediglich vier Testpersonen gaben an, gewöhnlich, nur eine Applikation zu verwenden. Die Anwendung „Whatsapp“ wurde am häufigsten genannt. Die Apps „Eventphant“ und „Smobber“ wurden nicht angegeben. Fünf Personen gaben die Verwendung sonstiger Medien an.

Außerdem wurde ermittelt, ob die vorgestellte Applikation eine bessere Lösung für den beschrieben Anwendungsbereich darstellt. Zehn Teilnehmer beantworteten diese Frage mit „Ja“ und eine Person mit „Nein“. Bei einem Fragebogen wurde diese Frage nicht beantwortet. Abschließend sollte festgestellt werden, ob die Probanden in Zukunft die vorgestellte Applikation für diesen Anwendungsbereich verwenden würden. Dabei bejahten acht Person diese Frage und drei verneinten. Diese Frage wurde ebenfalls von einer Person nicht beantwortet.

6 Fazit und Ausblick

In diesem abschließenden Kapitel werden die Ergebnisse der Arbeit gewürdigt. Dies erfolgt mit Blick auf die festgelegten Ziele und den Herausforderungen, die sich zu deren Erreichung ergeben haben. Des Weiteren wird ein Ausblick über das zukünftige Vorgehen zur Optimierung der Applikation vermittelt.

6.1 Ergebnisse

Die in der Zielsetzung festgelegten Anforderungen an diese Arbeit wurden schrittweise umgesetzt. In Kapitel 3 wurden mittels einer Recherche, die für den Anwendungsbereich relevanten Applikationen identifiziert und analysiert. Dabei wurden speziell die Schwachstellen herausgearbeitet. Es konnte festgestellt werden, dass keine der am Markt angebotenen Anwendungen den erforderten Nutzen erbringt. Für das Bewältigen der beschriebenen Aufgaben musste auf mehrere Applikationen zurückgegriffen werden die ein Workaround darstellen. Die Ergebnisse der Befragung haben diese Annahme bestärkt. Acht von zwölf Testpersonen gaben an gewöhnlich mehr als eine Applikation für diese Aufgaben zu benötigen. Außerdem konnte festgestellt werden, dass neun der befragten Personen mehrmals im Monat der Aufgabe der Planung und Organisierung von Veranstaltungsbesuchen in Clubs und Bars gegenüber stehen. Dies verdeutlicht die Relevanz des Anwendungsbereiches. Durch das Kriterium der Aufgabenangemessenheit des Evaluierungsbogens konnte die Effizienz der Applikation zur Aufgabenbewältigung ermittelt werden. Diese fiel mit einer durchschnittlichen Bewertung von 6,63 Punkten von insgesamt 7 möglichen Punkten sehr gut aus. Des Weiteren konnte die Ausbaufähigkeit der Steuerbarkeit und der Erwartungskonformität festgestellt werden.

Bei der Umsetzung der festgelegten Funktionalitäten bestand die größte Herausforderung in der Implementierung des Chat. Dieser sollte effizient und kostenlos ausführbar gestaltet werden. Dabei war der kostenlose Dienst von Google, das Google Cloud Messaging von großem Vorteil. Durch den entsprechenden Einsatz konnte ein effizientes Kommunikationsmittel entwickelt werden. Des Weiteren wird durch die Architektur und Funktionsweise der Applikation das Nutzen der Software ohne langwierige Anmeldungen und Freundesverwaltungen ermöglicht.

6.2 Ausblick

Die in dieser Arbeit entwickelte Applikation bildet den Grundstein für ein neues Konzept der Planung und Organisierung des Zusammenfindens sozialer Kontakte. Derzeit bietet die Datenbank Informationen bezüglich Veranstaltungen, diese können geteilt und verwendet werden um den Besuch dieser Veranstaltung zu planen. Der Vorteil besteht darin, dass dies zweckorientiert geschieht, die Daten werden geteilt und es wird direkt eine Möglichkeit der Kommunikation bezüglich dieser Veranstaltung geboten. Dieser Gedanke ist ausbaufähig, es können viele verschiedene Informationen geteilt werden, die der Zusammenfindung sozialer Kontakte dienen. Dies könnten auch Locations sein. Orte

und ihre Information, die direkt geteilt werden können und anschließend die Basis einer Unterhaltung, bezüglich eines Treffens bilden. Mit steigender Zahl der Personen, die teilnehmen sollen, wächst auch die Bedeutung dieser Applikation.

Der Nutzen, der durch die entwickelte Applikation generiert wird, ermöglicht die Einführung der Anwendung an den Markt. Die User erhalten die Möglichkeit eins kostenlosen mobilen Chats, außerdem können sie sich über Veranstaltungen informieren. Für Veranstalter besteht die Möglichkeit gezielt Werbung zu schalten und potenzielle Besucher zu akquirieren. Die Applikation kann somit für den Nutzer kostenlos angeboten werden, ermöglicht jedoch den wirtschaftlichen Einsatz durch die Bereitstellung von Werbefläche.

Bei der ersten Ausführung der Applikation muss der Nutzer seine Telefonnummer und seinen Namen eingeben, diese werden in der Datenbank gespeichert. Dadurch können langwierige Anmeldungen und Freundesverwaltungen vermieden werden. Für die Authentifizierung des Nutzers kann eine Bestätigungs-SMS an die angegebene Nummer gesendet werden, da dies mit Kosten verbunden ist, wird derzeit darauf verzichtet. Dies wird bei der Einführung am Markt implementiert. Des Weiteren kann die Applikation auch für Tablet PCs angeboten werden, da Android auch auf diesem Markt vertreten ist.

Abbildungsverzeichnis

Literatur

Marcus Pant & Arno Becker. *Android 2*. dpunkt.verlag, 2010.

Alexander Bertram. Nativ versus web. 2010. URL http://stud.fh-wedel.de/ inf7337/hybride-app-entwicklung.html. Zugriff: 2013.

Google. Google cloud messaging for android. 2012. URL http://developer.android.com/google/gcm/index.html. Zugriff: 2013.

Google. Android, the world's most popular mobile platform. 2013. URL http://developer.android.com/about/index.html. Zugriff: 2013.

VSIS Uni Hamburg. Mobile computing. 2012.

Apple Inc. ios developer program. 2013. URL https://developer.apple.com/programs/ios/. Zugriff: 2013.

Jens. Google play überholt apples app store. 2013. URL http://www.googlewatchblog.de/2013/01/apps-google-play-apples/. Zugriff: 2013.

Liz Laffan. The open governance index. 2011. URL http://www.visionmobile.com/blog/2011/07/the-open-governance-index-measuring-openness-from-android-to-webkit/. Zugriff: 2013.

Neil Mawston. Tablet os market share. 2012. http://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=7885 Zugriff: 2013.

Microsoft. Windows phone dev center. 2012. URL http://dev.windowsphone.com/en-us/join. Zugriff: 2013.

Christy Pettey. Mobile phone sales. 2010. URL http://www.gartner.com/newsroom/id/1306513. Gartner Zugriff: 2013.

Christy Pettey. Mobile application store revenue. 2011. URL http://www.gartner.com/newsroom/id/1529214. Gartner Zugriff: 2013.

Christy Pettey. Market share. 2012. URL http://www.gartner.com/newsroom/id/2237315. Gartner Zugriff: 2013.

Michael Anft & Jochen Prümper. Beurteilung von software auf grundlage der internationalen ergonomie-norm iso 9241/10. 1993.

Amit Kumar Saha. A developer' s first look at android. 2008.

Inderjeet Singh & Joel Leitch & Jesse Wilson. Gson user guide. URL https://sites.google.com/site/gson/gson-user-guide. Zugriff: 2013.

Anhang A

1 Aufgabenstellung

Ausgangspunkt:

Für die Bearbeitung der folgenden Aufgaben ist der Ausgangspunkt ein Szenario in dem Sie mit Ihren Freunden am Wochenende ausgehen möchten. Dabei soll der Besuch einer Veranstaltung geplant werden. Ihre Freunde werden in diesem Szenario von den anderen beiden Testpersonen dargestellt.

Aufgabe:

- Testperson 1: Suchen Sie unter Zuhilfenahme der Applikation eine Veranstaltung aus. Anschließend laden Sie die anderen Testpersonen zu dieser Veranstaltung ein und planen den Besuch und die Organisation des Abends.
- Testperson 2&3: Informieren Sie sich über das Event zu dem Sie eingeladen wurden. Tauschen Sie sich diesbezüglich mit den anderen Testpersonen aus. Sie haben ebenfalls die Möglichkeit eine andere Veranstaltung aus zu suchen und die anderen Teilnehmer einzuladen.

Anweisung

(Bitte unbedingt lesen!)

Im folgenden geht es um die Beurteilung von Softwaresystemen auf Grundlage der Internationalen Norm ISO 9241/10.

Das Ziel dieser Beurteilung ist es, Schwachstellen bei Softwaresystemen aufzudecken und konkrete Verbesserungsvorschläge zu entwickeln.

Um dies zu bewerkstelligen, ist Ihr Urteil als Kenner des Softwaresystems von entscheidender Bedeutung! Grundlage Ihrer Bewertung sind Ihre individuellen Erfahrungen mit dem Software-Programm, das Sie beurteilen möchten.

Dabei geht es nicht um eine Beurteilung Ihrer Person, sondern um Ihre persönliche Bewertung der Software mit der Sie arbeiten.

Am besten bearbeiten Sie den Beurteilungsbogen, während Sie das zu bewertende Softwaresystem vor sich am Bildschirm haben. Dadurch haben Sie die Möglichkeit, bei der Beantwortung der einzelnen Fragen die ein oder andere Sache noch einmal zu überprüfen.

Noch ein Hinweis zur Beantwortung des Beurteilungsbogens:

Die einzelnen Normen werden über Beschreibungen konkretisiert. Diese Beschreibungen weisen immer folgende Form auf.

Beispiel Nr.1:

Die Software ...	−−−	−−	−	−/+	+	++	+++	*Die Software ...*
ist schlecht.	☐	☐	☐	☐	☐	☒	☐	ist gut.

Beispiel 1

Im ersten Beispiel wird danach gefragt, wie gut, bzw. wie schlecht die Software ist.
Der Benutzer beurteilt in diesem Fall die Software zwar als gut, sieht jedoch noch Verbesserungsmöglichkeiten.

Beispiel Nr.2:

Die Software ...	−−−	−−	−	−/+	+	++	+++	*Die Software ...*
ist langsam.	☐	☒	☐	☐	☐	☐	☐	ist schnell.

Beispiel 2

Im zweiten Beispiel beurteilt der Benutzer die Software als ziemlich langsam.

Füllen Sie bitte den Beurteilungsbogen äußerst sorgfältig aus und lassen Sie keine der Fragen aus!

Die Auswertung der Daten erfolgt anonym.

Aufgabenangemessenheit

Unterstützt die Software die Erledigung Ihrer Arbeitsaufgaben, ohne Sie als Benutzer unnötig zu belasten?

Die Software ...	−−−	−−	−	−/+	+	++	+++	*Die Software ...*
ist kompliziert zu bedienen.	☐	☐	☐	☐	☐	☐	☐	ist unkompliziert zu bedienen.
bietet nicht alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.	☐	☐	☐	☐	☐	☐	☐	bietet alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.
erfordert überflüssige Eingaben.	☐	☐	☐	☐	☐	☐	☐	erfordert keine überflüssigen Eingaben.
ist schlecht auf die Anforderungen der Arbeit zugeschnitten.	☐	☐	☐	☐	☐	☐	☐	ist gut auf die Anforderungen der Arbeit zugeschnitten.

Selbstbeschreibungsfähigkeit

Gibt Ihnen die Software genügend Erläuterungen und ist sie in ausreichendem Maße verständlich?

Die Software ...	−−−	−−	−	−/+	+	++	+++	*Die Software ...*
bietet einen schlechten Überblick über ihr Funktionsangebot.	☐	☐	☐	☐	☐	☐	☐	bietet einen guten Überblick über ihr Funktionsangebot.
verwendet schlecht verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.	☐	☐	☐	☐	☐	☐	☐	verwendet gut verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.
liefert in unzureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.	☐	☐	☐	☐	☐	☐	☐	liefert in zureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.

Steuerbarkeit

Können Sie als Benutzer die Art und Weise, wie Sie mit der Software arbeiten, beeinflussen?

Die Software ...	−−−	−−	−	−/+	+	++	+++	*Die Software ...*
bietet keine Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen.	☐	☐	☐	☐	☐	☐	☐	bietet die Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen.
erzwingt eine unnötig starre Einhaltung von Bearbeitungsschritten.	☐	☐	☐	☐	☐	☐	☐	erzwingt keine unnötig starre Einhaltung von Bearbeitungsschritten.
ermöglicht keinen leichten Wechsel zwischen einzelnen Menüs oder Masken.	☐	☐	☐	☐	☐	☐	☐	ermöglicht einen leichten Wechsel zwischen einzelnen Menüs oder Masken.
erzwingt unnötige Unterbrechungen der Arbeit.	☐	☐	☐	☐	☐	☐	☐	erzwingt keine unnötigen Unterbrechungen der Arbeit.

Erwartungskonformität

Kommt die Software durch eine einheitliche und verständliche Gestaltung Ihren Erwartungen und Gewohnheiten entgegen?

Die Software ...	−−−	−−	−	−/+	+	++	+++	*Die Software ...*
erschwert die Orientierung, durch eine uneinheitliche Gestaltung.	☐	☐	☐	☐	☐	☐	☐	erleichtert die Orientierung, durch eine einheitliche Gestaltung.
läßt einen im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.	☐	☐	☐	☐	☐	☐	☐	läßt einen nicht im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.
informiert in unzureichendem Maße über das, was sie gerade macht.	☐	☐	☐	☐	☐	☐	☐	informiert in ausreichendem Maße über das, was sie gerade macht.
läßt sich nicht durchgehend nach einem einheitlichen Prinzip bedienen.	☐	☐	☐	☐	☐	☐	☐	läßt sich durchgehend nach einem einheitlichen Prinzip bedienen.

Lernförderlichkeit

Ist die Software so gestaltet, daß Sie sich ohne großen Aufwand in sie einarbeiten konnten und bietet sie auch dann Unterstützung, wenn Sie neue Funktionen lernen möchten?

Die Software ...	−−−	−−	−	−/+	+	++	+++	*Die Software ...*
erfordert viel Zeit zum Erlernen.	☐	☐	☐	☐	☐	☐	☐	erfordert wenig Zeit zum Erlernen.
ermutigt nicht dazu, auch neue Funktionen auszuprobieren.	☐	☐	☐	☐	☐	☐	☐	ermutigt dazu, auch neue Funktionen auszuprobieren.
erfordert, daß man sich viele Details merken muß.	☐	☐	☐	☐	☐	☐	☐	erfordert nicht, daß man sich viele Details merken muß.
ist so gestaltet, daß sich einmal Gelerntes schlecht einprägt.	☐	☐	☐	☐	☐	☐	☐	ist so gestaltet, daß sich einmal Gelerntes gut einprägt.
ist schlecht ohne fremde Hilfe oder Handbuch erlernbar.	☐	☐	☐	☐	☐	☐	☐	ist gut ohne fremde Hilfe oder Handbuch erlernbar.

Relevanz und Vergleich

Wie oft stehen Sie der beschriebenen Aufgabe gegenüber?

☐ 2 mal die Woche oder öfter ☐ 2-3 mal im Monat ☐ Nie
☐ 1 mal die Woche ☐ 1 mal im Monat

Welche Applikation(en) verwenden Sie gewöhnlich für den beschriebenen Anwendungsfall?

☐ Facebook ☐ Virtualnights ☐ Smobber
☐ Whatsapp ☐ Eventphant ☐ Sonstige

Eignet sich die vorgestellte Applikation besser für den Anwendungsbereich?

☐ Ja ☐ Nein

Würden Sie in Zukunft die vorgestellte Applikation für diesen Anwendungsfall verwenden?

☐ Ja ☐ Nein

Zum Schluss

Zum Schluß bitten wir Sie, noch folgende Fragen zu beantworten:

Was ist Ihr Beruf?	.....		JOB
Wie alt sind Sie?	.....	Jahre	AGE
Ihr Geschlecht?	.....	m/w	SEX

Anhang B

Digitale Version auf CD-ROM

Inhalt dieser beigefügten CD-ROM sind die digitale Version dieser Bachelorarbeit sowie die Ergebnisse der Evaluierung.

Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Diese Versicherung bezieht sich auch auf alle in der Arbeit enthaltenen Zeichnungen, Skizzen, bildlichen Darstellungen und dergleichen.

Des Weiteren bin ich mit einer Einstellung der Arbeit in die Bibliothek nicht einverstanden.

(Ort, Datum) (Unterschrift)