HTL - Perg
Höhere Abteilung für Informatik

Diplomarbeit

AdQuest

Entwicklung und Umsetzung eines neuartigen Marketingkonzeptes

Projektteam: Tobias Friedinger

Sebastian Hatmanstorfer

Lukas Hinterreiter

Michael Leonhartsberger

Harald Mayrhofer

Projektbetreuer: Professor DI Dietmar Wokatsch-Ratzberger

In Zusammenarbeit mit der Kreft Steuerberatungs GmbH

Geschäftsführer Herr Mag. Gerald Kreft

Bearbeitungszeitraum: 03.10.2016 – 05.04.2017

1. Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg, _____ Unterschrift _____

(Tobias Friedinger)

Perg, _____ Unterschrift _____

(Sebastian Hatmanstorfer)

Perg, _____ Unterschrift _____

(Lukas Hinterreiter)

Perg, _____ Unterschrift _____

(Michael Leonhartsberger)

Perg, _____ Unterschrift _____

(Harald Mayrhofer)

2. Danksagung

Das gesamte Diplomarbeitsteam bedankt sich bei sämtlichen Personen und Organisationen, die diese Diplomarbeit ermöglicht haben.

Besonders hervorheben möchten wir hier unseren Diplomarbeitsbetreuer
Herrn DI Dietmar Wokatsch-Ratzberger, der uns während der ganzen Diplomarbeit unterstützt hat, so wie den Geschäftsführer der Kreft Steuerberatungs GmbH,
Herrn Mag. Gerald Kreft, welcher als Auftraggeber und Ansprechpartner fungierte.
Darüber hinaus auch großen Dank an Frau Professor Elke Buchberger-Sigl, welche uns maßgeblich bei der Erstellung unseres Geschäftsmodelles unterstützt hat.

Des Weitern möchten wir uns bei Herrn Prof. Ing. Werner Schöller sowie Herrn Clemens Denzinger bedanken, welche durch die Bereitstellung der Serverumgebung, die technische Basis für diese Diplomarbeit schufen.

3. Impressum

Schule

HTBLA Perg für Informatik

Machlandstraße 48

4320 Perg

Schuljahr

2016/17

Klasse

5AHIF

Projektname

AdQuest

Projektteam

Tobias Friedinger

Sebastian Hatmanstorfer

Lukas Hinterreiter

Michael Leonhartsberger

Harald Mayrhofer

Betreuungslehrer

Prof. DI Dietmar Wokatsch-Ratzberger

Auftraggeber

Kreft Steuerberatungs GmbH

Ghegastraße 30, 4020 Linz

Geschäftsführer und Ansprechpartner: Mag Gerald Kreft

4. Inhaltsverzeichnis

5. Kurzbeschreibung

AdQuest ist das erste System, einer neuen, innovativen Werbegeneration. Es ist sehr auf das Internet und Social-Media-Plattformen wie Facebook ausgerichtet. Es soll vor allem kleinen, regionalen Unternehmen die Möglichkeit günstiger Werbeschaltung bieten.

Kunden können Events, sogenannten "Quests" auf unserer Website anlegen in denen sie Produkte, wie etwa Gutscheine, Schuhe, etc. in Form von GeoCaches in der Natur verstecken können. Quests haben bestimmte, vom Ersteller festgelegte Teilnehmerschwellen, ab denen sie starten und das versteckte Geschenk gefunden werden kann. Um an einer Quest teilnehmen zu können, muss ein User mit seiner AdQuest-App den Werbebeitrag des Quest-Veranstalters auf Facebook teilen, wodurch der sogenannte Ad-Spreading Effekt entsteht und sich die Werbung sehr schnell an sehr viele Facebook-User verbreitet.

Aufgrund diese Effektes und des geringen Aufwandes ist es uns möglich, Werbung regional und zu unschlagbar günstigen Preisen anzubieten.

6. Abstract

AdQuest is a new, innovative and interactive advertisment-system. It is based on the concept of online-ads, especially ads over social media and uses the effect of information-spreading through the sharing-system of facebook. Moreover it turns every advertisement in an exciting searchquest for the customers.
Our clients have the opportnunity of buying so called “quests” on our website. In this quests they can hide free-products everywhere in the nature. With every quest, clients also publish a facebook-post, about their company. Now customers are able to see the quest in their app, but without any information about the location of the hidden products. If they want get this information, they first have to share the post of our clients with all their friends on facebook. This concept is what makes this form of advertisments so effective.
Our clients also have the opportunity to rent a drone. For little more money the can film the search of their quest. The captured raw-material can then be snipped to an advertisement-film, which can further be published.
AdQuest is aimed especially at small, local companies who cannot afford expensive advertisments, such as TV- or radiospots. For this kind of companies AdQuest offers an aimed, extremely effective, and unbelievably cheap option of advertisement.

7. Beteiligte

7.1. Team

7.1.1. Tobias Friedinger (Projektleiter)

Abbildung 1 – Tobias Friedinger

Persönliches:

Geburtsdaten:	18.02.1998
Staatsbürgerschaft:	Österreich
Religion:	evang. A. B.
Eltern:	Alois Martin Friedinger Andrea Friedinger

Schulbildung:

2012 – 2017	HTL-Perg
2008 – 2012	CMC-HS Ried in der Riedmark
2004 – 2008	Volksschule Ried in der Riedmark

Arbeitserfahrung:

Sommer 2016	Engel, CSD
Sommer 2015	Engel, CSD

Kontakt:

Telefon:	+43664 304 46 24
E-Mail:	tobias.friedinger@gmail.com

7.1.2. Sebastian Hatmanstorfer

Abbildung 2 – Sebastian Hatmanstorfer

Persönliches:

Geburtsdaten:	13.05.1998
Staatsbürgerschaft:	Österreich
Religion:	röm.-kath.
Eltern:	Karl Hatmanstorfer Margit Hatmanstorfer

Schulbildung:

2012 – 2017	HTL-Perg
2008 – 2012	CMC-HS Ried in der Riedmark
2004 – 2008	Volksschule Ried in der Riedmark

Arbeitserfahrung:

Sommer 2016	E+E Elektronik, Abteilung IT
Sommer 2015	Land OÖ, Abteilung Softwarentwicklung
Sommer 2014	Land OÖ, Abteilung Rechnungswesen

Kontakt:

Telefon:	+43699 110 457 51
E-Mail:	s.hatmanstorfer@gmail.com

7.1.3. Lukas Hinterreiter

Abbildung 3 – Lukas Hinterreiter

Persönliches:

Geburtsdaten:	10.11.1997
Staatsbürgerschaft:	Österreich
Religion:	röm.-kath.
Eltern:	Johann Hinterreiter Johanna Hinterreiter

Schulbildung:

2012 – 2017	HTL-Perg
2008 – 2012	Hauptschule für Informationstechnologie Grein
2004 – 2008	Volksschule Grein

Arbeitserfahrung:

Sommer 2016	IFE Kematen, Abteilung Warenwirtschaft
Sommer 2014	IFE Kematen, Abteilung Entwicklung
Sommer 2013	IFE Kematen, Abteilung Entwicklung

Kontakt:

Telefon:	+436649150353
E-Mail:	lukas.hinterreiter@gmail.com

7.1.4. Michael Leonhartsberger

Abbildung 4 – Michael Leonhartsberger

Persönliches:

Geburtsdaten:	13.01.1998
Staatsbürgerschaft:	Österreich
Religion:	röm.-kath.
Eltern:	Walter Leonhartsberger Elisabeth Leonhartsberger

Schulbildung:

2012 – 2017	HTL-Perg
2008 – 2012	HS Sankt Georgen am Walde
2004 – 2008	Volksschule Sankt Georgen am Walde

Arbeitserfahrung:

Sommer 2015	CADS, Bereich Softwareentwicklung
Sommer 2014	Biberauer, Abteilung Büro

Kontakt:

Telefon:	+43 6801346185
E-Mail:	leonhartsbergerm@gmail.com

7.1.5. Harald Mayrhofer

Abbildung 5 – Harald Mayrhofer

Persönliches:

Geburtsdaten:	19.09.1998
Staatsbürgerschaft:	Österreich
Religion:	röm.-kath.
Eltern:	Eva Mayrhofer Wolfgang Mayrhofer

Schulbildung:

2012 – 2017	HTL-Perg
2008 – 2012	CMC-HS Ried in der Riedmark
2005 – 2008	Volksschule Ried in der Riedmark

Arbeitserfahrung:

Sommer 2016	STIWA, Abteilung Softwareentwicklung
Sommer 2015	Programmierfabrik
Sommer 2014	SecureGuard

Kontakt:

Telefon:	+43 664 375 80 87
E-Mail:	mayrhofer.ha@gmail.com

7.2. Betreuung

7.2.1. Prof. DI Dietmar Wokatsch-Ratzberger

Abbildung 6 – DI Dietmar Wokatsch-Ratzberger

Betreut wurde die Diplomarbeit von Herrn Prof. DI Dietmar Wokatsch-Ratzberger. Herr Prof. Wokatsch unterrichtet uns seit dem 3. Jahrgang im Fach Datenbanken und Informationssysteme. Aufgrund fundierter Kenntnisse, nicht nur im Bereich Datenbanken, sondern auch im Bereich Projektentwicklung, stand er uns auch bei der Projektplanung, und Konzeptentwicklung, stets zur Seite. Durch seine ebenfalls sehr hohe Kompetenz im Bereich der Rhetorik konnten wir insbesondere bei diversen Präsentationen und Vorträgen, sehr von Herrn Prof. Wokatsch lernen und profitieren.

Kontakt: d.wokatsch@htl-perg.ac.at

7.3. Auftraggeber

7.3.1. Kreft Steuerberatungs GmbH

Abbildung 7 – Kreft_Logo (kreft.at, 2017)

Die Kreft Steuerberatungs GmbH ist eine Linzer Steuerberatungs- und Wirtschaftstreuhandskanzlei. Leiter und Geschäftsführer ist Herr Mag. Gerald Kreft, seines Zeichens auch Auftraggeber und Ansprechpartner der Diplomarbeit. Herr Mag. Kreft Leitet das Unternehmen nun in 3. Generation, mit außerordentlicher Kompetenz und Know-How.

Kontakt:
Herr Mag. Gerald Kreft: office@kreft.at

8. Einleitung

8.1. Motivation

Das Interesse, an der Ausarbeitung und Erstellung eines neuen und innovativen Systems war von Anfang an im gesamten Projektteam sehr hoch. Am Werbemarkt ist derzeit noch kein vergleichbares System im Einsatz, das Werbung in dieser Form, geschweige denn in der niedrigen Preiskategorie anbietet, wie es AdQuest tut. Dieser Faktor, sowie das allgemeine Interesse am Lernen neuer Techniken zur Umsetzung solcher Systeme, schufen bereits von Anfang an ein sehr motiviertes und angenehmes Arbeitsklima.

8.2. Ziele

Ziel und Intention der Diplomarbeit liegen darin, ein flexibles und einsetzbares System zur Umsetzung der Geschäftsidee zu entwickeln. Des Weitern steht auch die Ausarbeitung eines realistischen und wirtschaftlich umsetzbaren Geschäftsmodelles bzw. Businessplanes im Zentrum dieser Arbeit. Das Ergebnis der Diplomarbeit soll eine solide Grundlage für ein neues Start-Up Unternehmen darstellen.

8.3. Zusammenarbeit mit Auftraggeber

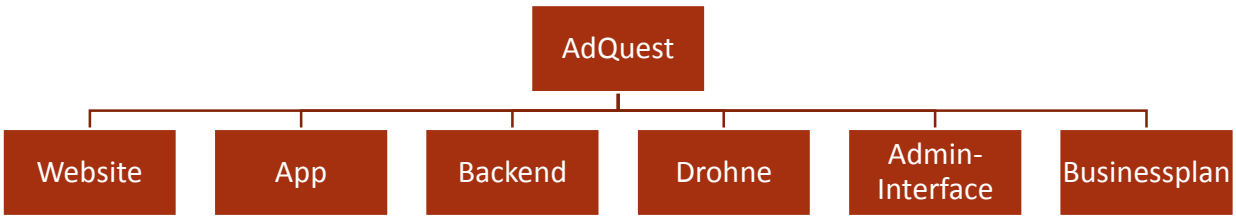
Die Zusammenarbeit mit dem Auftraggeber verlief von Anfang an reibungslos. Herr Mag. Kreft hat in den Jahren in denen er bereits in der Wirtschaft tätig ist, großes Know-How angesammelt und hatte stets Tipps, und Vorschläge zu Verbesserungen des Systems parat. Weiters gab uns Herr Kreft bei der Entwicklung praktisch keine Einschränkungen in Bezug auf das Design bzw. zur Funktionalität, was zusätzliche Motivation im ganzen Projektteam hervorrief.

9. Das Projekt

9.1. Projektbeschreibung

AdQuest ist ein neues, innovatives und interaktives Werbesytem. Es basiert auf dem Konzept von Online-Werbung, im speziellen Werbung über Social-Media und bedient sich hierbei der Verbreitung von Informationen über das Sharing-System von Facebook. Auf der Unternehmensseite kaufen Kunden bei uns Werbeveranstaltungen, sogenannte Quests, in denen sie Produkte, wie etwa Gutscheine oder Schuhe, in Form von GeoCaches überall in der Natur verstecken können. Zusammen mit der Quest, erstellt der Kunde auch einen Facebook-Werbepost, in dem er sein Unternehmen und seine Veranstaltung bewirbt. Auf der Konsumentenseite, steht dem gegenüber die AdQuest-App. In dieser sieht der Konsument die Quests des Kunden, nicht aber wo dieser seine Goodies versteckt hat. Um dazu genauere Infos zu erhalten, sprich einen Suchradius in welchem das Produkt versteckt ist, muss der Konsument zuerst über unsere App den Werbepost des Kunden, mit all seinen Facebook-Freunden teilen. Dadurch entsteht für den Werbetreibenden eine extrem rapide Verbreitung seiner Werbebotschaft – das sogenannte AdSpreading.
Für AdQuest entstehen während all dieser Vorgänge beinahe keine Kosten, wodurch es uns möglich ist unser System so günstig anzubieten.

9.2. Projektbestandteile



10. Allgemeines

10.1. Entwicklungsumgebungen

Im folgenden Kapitel werden Informationen, zu den für die Umsetzung und der Entwicklung nötigen Programmen, dargelegt.

10.1.1. JetBrains WebStorm

Abbildung 8 – WebStorm_Logo (Jetbrains, 2017)

WebStorm ist eine, von der Firma JetBrains veröffentlichte Entwicklungsumgebung für JavaScript, HTML und CSS, also zur Entwicklung von Weboberflächen. Es basiert auf dem IntelliJ IDEA von JetBrains, ist aber nicht auf Java sondern auf JavaScript spezialisiert. WebStorm sticht vor allem durch seine erweiterte Unterstützung des JS Frameworks AngularJS, seiner hervorragenden code-completiton und seiner schnellen code-inspection heraus. Des Weiteren unterstützt es den Benutzer mit diversen Funktionen zur Versionskontrolle. (vgl. Jetbrains, 2017)

10.1.2. IntelliJ

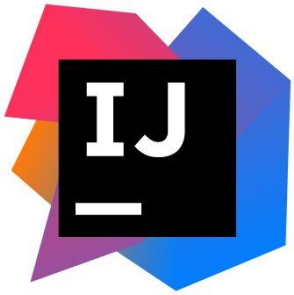


Abbildung 9 – IntelliJ_Logo (Twitter, 2017)

IntelliJ ist eine von JetBrains entwickelte IDE zur Java-Entwicklung. Ebenso wie WebStorm besticht es durch seine hervorragende code-completition, seine Erweiterbarkeit über Plug-Ins und seine Tools zur Versionsverwaltung. Das UI ist wie bei WebStorm sehr übersichtlich gestaltet und erleichtert seinem Benutzer durch eine Vielzahl von Shortcuts die Navigation. In AdQuest wurde IntelliJ zur Entwicklung der Administrationsoberfläche verwendet. (vgl. Jetbrains, 2017)

10.1.3. Android Studio

Abbildung 10 – AS_Logo (Pinterest.com, 2017)

Android Studio ist die offizielle Entwicklungsumgebung von Android, zur Entwicklung von Apps. Es basiert auf dem IntelliJ IDE von JetBrains und bietet schnelle und übersichtliche Entwicklung von Android-Applikationen. Android Studio ist vor allem wegen seines intelligenten Code-Editors sehr benutzerfreundlich. Herausragende code-completition und eine allgemein sehr gute Performance erleichtern dem Benutzer das Programmieren. Außerdem hat man mit dem integrierten Emulator eine breite Palette von schnell laufenden Android-Systemen zur Verfügung und ist nicht ständig auf ein eigenes Gerät angewiesen. (vgl. Google, 2017)

10.1.4. Eclipse

Abbildung 11 – Eclipse_Logo (Seeklogo, 2017)

Eclipse ist eine open-source Entwicklungsumgebung zur Erstellung von Software verschiedenster Art. Hauptsächlich wird es für die Entwicklung von Java genutzt. Eclipse ist besonders durch die große Erweiterbarkeit und dem übersichtlichen UI eine gute Wahl für das Erstellen von Software. (vgl. Wikipedia, 2017)

Bei AdQuest wurde Eclipse mit dem Wildfly Plug-In zur Entwicklung des REST-Servers genutzt.

10.2. Tools

Im folgenden Kapitel werden alle Tools die zur Unterstützung des Entwicklungsprozesses dienten aufgelistet.

10.2.1. Postman

Postman ist eine open-source Software zum Testen der REST-Kommunikation mit Endschnittstellen. Es ermöglicht das Prüfen von GET, POST, PUT und DELETE Requests auf REST-Server. Besonders zum Testen von Funktionalitäten des Servers sind solche REST-Clients wie Postman ein unerlässliches Tool.

Abbildung 12 – Postman (Postman, 2017)

10.2.2. MySQL Workbench

MySQL Workbench ist ein Visualisierungstool für MySQL Datenbanken. Es ermöglicht Datenmodelle zu erstellen und diese direkt über automatisierte Skripterstellung auf der Datenbank auszuführen. Zusätzlich ermöglicht es direkt Daten auf der Datenbank zu verwalten und SQL-Befehle auszuführen. Außerdem bietet es eine einfache Lösung zur Datenbankmigrierung. (vgl. mysql.de, 2017)

Abbildung 13 – MySQLWorkbench_Logo (quintetsolutions.com, 2017)

10.2.3. Firebase

Firebase ist eine mobile und web application platform, welche Tools und Infrastruktur zur Verfügung stellt um Apps zu entwickeln. Es stellt beispielsweise ein Service bereit, welches Push-Notifications an eine Android-App senden kann. (vgl. Wikipedia, 2017), (vgl. Google, 2017)

Abbildung 14 – Firebase_Logo (Google, 2017)

10.2.4. SourceTree

SourceTree ist eine open-source Benutzeroberfläche des Herstellers Atlassian, die das Verwalten von Git-Repositories ermöglicht. Vor allem durch seine erweiterten Funktionen wie etwa Review changesets, stash und seinem übersichtlichen Userinterface überzeugt SourceTree als Verwaltungswerkzeug. (vgl. Atlassian, 2017)

Abbildung 15 – SourcetreeLogo (Atlassian, 2017)

10.2.5. BitBucket

BitBucket ist ein Filehosting-Dienst der das Versionsverwaltungssystem GIT und Mercurial unterstützt. 2010 wurde BitBucket vom Unternehmen Atlassian gekauft. BitBucket ist besonders auf Nutzer ausgerichtet und speichert das Projekt nicht zentral. Es ist außerdem besonders auf Branchen (Erstellen) und Mergen (Vereinigen) von Abspaltungen spezialisiert und unterstützt somit das gleichzeitige Entwickeln mehrerer Projekte. (vgl. Wikipedia, 2017)

10.3. Sonstige Software

10.3.1. Microsoft Office

Office bietet eine Reihe von Standardprogrammen, die man generell bei fast jeder dokumentarischen Tätigkeit im Projekt benötigt.

MS Office Word:

- Dokument- und Textbearbeitungssoftware

MS PowerPoint:

- Präsentationssoftware

MS Excel:

- Tabellenkalkulationsprogramm

Visio:

- Visualisierungsprogramm zur Erstellung von Diagrammen

10.3.2. Ganttproject

Gantt ist ein open-source Programm zur Visualisierung und Erstellung von Zeitplänen.

10.3.3. Affinity Designer

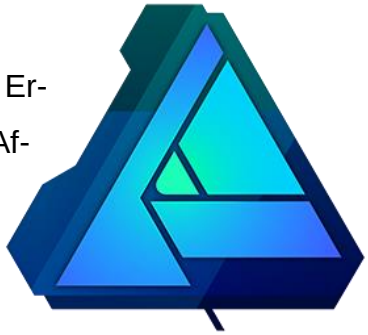


Abbildung 16 – AD_Logo (Affinity, 2017)

Der von Adobe entwickelte Affinity Designer ist ein Programm zur Erstellung von Vektorgrafiken und Grafiken für Werbematerial. Der Affinity Designer bietet vorbildliche Leistungen auf verschiedensten Plattformen und sehr hohe Geschwindigkeit und Präzision bei der Erstellung von Grafiken. Außerdem verspricht er pixelgenaue Steuerung und eine Echtzeit-Pixelvorschau, sowie eine große Reihe von verschiedenen Effekten und Anpassungen. (vgl. Affinity, 2017)

10.4. Verwendete Techniken

10.4.1. Angular2

Abbildung 17 – Angular2_Logo (Habert, 2017)

Angular2 ist ein auf TypeScript aufbauendes Framework zur Entwicklung von Webapplikationen. Dank der Verbindung mit Typescript bringt Angular2 die Möglichkeit der objektorientierten Programmierung in die Webentwicklung (Vergleichbar mit der Syntax von Java 8).
Anders als viele andere Frameworks ist Angular2 kein reines MVC Framework (Model View Controller) sondern viel mehr komponentenbasiert. Eine Angular2 Applikation ist ein Baum aus lose gekoppelten Komponenten.

Neben diversen objektorientierten Features bekommt man mit Angular2 auch die Angular CLI. Diese Commandline-Tools erleichtern dem Benutzer den Umgang mit dem Framework, und bieten viele hilfreiche Kommandos. (vgl. infoq.com, 2017)

10.4.1.1. Typescript

Typescript ist eine von Microsoft entwickelte Programmiersprache, die Sprachkonstrukte wie Klassen, Interfaces, Vererbung usw. bereitstellt. Der Typescript-Compiler ist so geschrieben, dass auch jeder JavaScript Code gültiger TypeScript Code ist. Somit ist eine Verbindung zu gängigen JavaScript-Bibliotheken wie Angular möglich. (vgl. Wikipedia, 2017)

10.4.1.2. BrowserSync

BrowserSync ist ein Tool das gemeinsam mit Angular2 geliefert wird. Es erleichtert dem Benutzer das Entwickeln von Webapplikationen, da es eine dauerhafte Synchronisation des geschriebenen Codes mit dem Browserfenster gewährleistet, d.h. jede größere Änderung im Code bzw. jedes Speichern führt zu einem Reload des Browsers.

10.4.2. Java

Java ist eine objektorientierte und plattformunabhängige Programmiersprache, deren Syntax sich stark an C++ anlehnt. Objektorientiert heißt hier, dass das Programm in Objekte unterteilt wird die verschiedenen Zustände aufweisen können. Objekte können mittels Funktionen verwaltet werden und stehen meist untereinander in Beziehung.

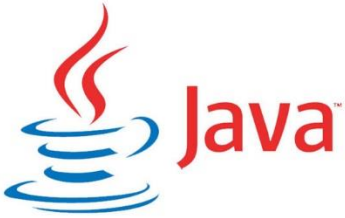


Abbildung 18 – JAVA_Logo (logos.wikia.com, 2017)

Um ein Java-Programm ausführen zu können wird ein JRE (Java Runtime Environment) benötigt. Außerdem braucht man ein JDK (Java Development Kit) welches für die Codierung und Kompilierung benötigt wird.
Java ist plattformunabhängig – d.h. das Java-Programme auf jedem Gerät ausführbar sind, auf dem Java installiert ist. (vgl. itwissen.info, 2017)

10.4.3. Java Persistence API (JPA)

JPA ist ein Standard für das objektrelationale Mapping von Java-Objekten (auf relationale Datenbanken). Mit JPA ist es möglich Klassen bzw. Objekte in relationalen Datenbanken zu persistieren (speichern). (vgl. itwissen.info, 2017)

10.4.4. Java Server Faces (JSF)

JSF ist ein Web-Application-Framework das auf Java basiert. Durch JSF ist es möglich, grafische Benutzeroberflächen, JSF Web-Applikationen, für Serveranwendungen zu erstellen. Es stellt die Verbindung der Komponenten der GUI zu der Datenquelle dar. JSF ergänzt Komponenten von HTML5. (vgl. tutorialspoint.com, 2017)

10.4.5. Wildfly

Wildfly ist ein kostenloser Anwendungsserver der nach dem JPA-Standard und in Java gecodet ist.
In AdQuest ist der Wildfly-Server für das REST-Service zuständig.

10.4.6. Apache-Webserver

Der Apache http Server ist ein open-source Produkt der Apache Software Foundation. Es handelt sich dabei um einen Webserver den man über das Internet erreichen kann und der von jedem gehostet werden kann.
Im Fall von AdQuest ist er zuständig für das Hosten der AdQuest-Website.

10.4.7. Resteasy

Resteasy ist ein Framework mit dem RESTful Web Services in Java implementiert werden können. (vgl. jaxenter.de, 2017)

10.4.8. MySQL

MySQL ist ein relationales open-source Datenbankverwaltungssystem. (vgl. Wikipedia, 2017)

10.4.9. NodeJS

NodeJS ist eine der bekanntesten JavaScript-Runtimes. Sie ist von Google entwickelt und läuft mit der „Chrome V8 JavaScript Engine“. NodeJS wurde für Chrome entwickelt und ist daher sehr ressourcensparend. Was NodeJS ausmacht sind die Module die dafür entwickelt werden. Zur Verwaltung dieser Module gibt es den sogenannten „Node Package Manager“, über den sich laut Stand 2017 bereits über 400.000 Pakete installieren lassen. (vgl. Wikipedia, 2017)

10.5. Design

Die AdQuest Website und App sind im Google Material Design gehalten.

10.5.1. Material Design

Das Material Design ist eine von Google entwickelte Designsprache welche auf dem sogenannten Flat Design basiert. Das Flat Design benutzt zur Darstellung meist kartenähnliche Flächen und Objekte welche allerdings sehr minimalistisch gestaltet sind. Dennoch kommen sehr viele Animationen zum Einsatz und es werden Schatten zum Erzeugen von Tiefeneffekten verwendet. (vgl. Wikipedia, 2017)

11. Das Geschäftsmodell

11.1. Produkte & Dienstleistungen

11.1.1. Angebotene Produkte / Dienstleistungen (Unternehmensseite)

Die Verkaufsstrategie von AdQuest ist im B2B Bereich angelegt und bietet Unternehmenskunden die Möglichkeit der Nutzung des Systems von AdQuest zur Werbeschaltung. AdQuest bietet zwei käufliche Produkte an (in unterschiedlichen Ausführungen): Quest und Events. Diese stellen jedoch nicht den Hauptnutzen für den Unternehmer dar. Sie sind Mittel zum Zweck → Werbung für den Unternehmer über Social-Media. Diese entsteht, wenn ein Konsument, einen vom Unternehmer verfassten Beitrag teilt um an einer Quest/einem Event teilnehmen zu können. Dieser Beitrag kann individuelle Werbung für jedes Unternehmen enthalten. Durch das AdQuest-System wird dieser verbreitet.

11.1.1.1. Quests

Eine Quest kann auf der AdQuest-Website erstellt werden. Es wird eine Benutzerschwelle vergeben, ab deren Überschreiten die Quest startet. Der Unternehmer bestimmt wie viele AdCaches er verstecken möchte (=wie viele Produkte er verlosen will).
Diese AdCaches werden nun versteckt (vom Unternehmer oder von den Mitarbeitern von AdQuest) und die Koordinaten im AdQuest-System gespeichert. Es werden zwei Arten von Quests angeboten:

11.1.1.2. Vorgefertigte Quests (Pakete)

Quest werden, um den Kunden die Bedienung des Systems zu erleichtern und um Basisdaten zur Orientierung für den Unternehmer bereitzustellen, in verschiedenen Paketen angeboten. Sie enthalten eine bereits im Vorfeld definierte Größe, Benutzerschwelle und Beschreibung des Einsatznutzens beinhaltet.

11.1.1.3. Selbstkonfigurierte Quests

AdQuest bietet erfahreneren Kunden auch die Möglichkeit, ihre Quest selbst zu erstellen. Der Kunde kann dabei die Teilnehmerschwelle und die Anzahl der versteckten Caches selbst wählen. Da finanziell für AdQuest dadurch ein gewisser Mehraufwand entsteht wird bei selbstkonfigurierten Paketen ein kleiner Aufschlag verrechnet

11.1.1.4. Event

Grundsätzlich gleich aufgebaut wie ein Quest, allerdings ohne Benutzerschwelle. Kunden geben bei Erstellen eines Events eine maximale Benutzeranzahl ein und ein Datum, an dem das Event unabhängig von der Teilnehmerzahl fix startet. Das Teilen des Facebook-Beitrags ist ebenfalls Pflicht zur Teilnahmen.
Events ermöglichen eine genauere Zeitplanung und erlauben dem Kunden ein Event zu organisieren(zum Beispiel: Gemeinsamer Start der Suche, Catering Service, Show-Auftritte, etc.).

11.1.1.5. Drohnenservice

Zusätzlich bietet AdQuest bei einem Event einen Drohnenservice an. Ein AdQuest-Mitarbeiter filmt das Event mittels einer Drohne und kann durch Benachrichtigungen vom Server gezielt die AdCaches filmen, die am kürzesten vor Ihrer Entdeckung stehen. Dadurch entsteht ein Werbevideo für den Unternehmenskunden.

11.1.2. Produktnutzen – Warum AdQuest?

Da sich die Kosten, die pro Werbeschaltung für AdQuest entstehen, hauptsächlich aus Wartungskosten und Serverkosten bzw. diversen geringen Fixkosten für das Unternehmen zusammensetzten, ist es möglich, den Kunden extrem günstige Preise für die Werbungschaltung anzubieten.
Darüber hinaus besitzt die AdQuest-App auf Konsumentenseite einen Umkreisfilter, der je nach Größe der Quests/Events Konsumenten nur jene in ihrer näheren Umgebung anzeigt. Dadurch lässt sich ein sehr stark regional forcierter Werbeeffekt erzielen.

11.1.3. Innovation

Der Innovationsgrad von AdQuest definiert sich durch eine Reihe von Merkmalen. Die wichtigsten von diesen sind:

- ➔ **Gamification: Neuartige Verbindung von Werbung und Suchabenteuer**
- ➔ **Vorteile für Beide Seiten des Systems**
  - Kunde → Günstige Möglichkeit der Werbeschaltung
  - Konsument → spannendes Suchabenteuer und Gratisprodukte
- ➔ **unglaublich rasche Verbreitung der Werbebotschaft auf Facebook** → Ad-Spreading

11.1.4. Angebotenes Produkt (Konsumentenseite)

11.1.4.1. AdQuest-App

Da AdQuest nur im B2B-Bereich gewinnorientiert arbeitet, hat die für Konsumenten entwickelte AdQuest-App nur indirekt, als Bestandteil des gesamten Systems einen kommerziellen Nutzen. Auf der App werden keine Produkte verkauft und sie ist kostenlos für Konsumenten erhältlich.
Die App bietet dem Konsumenten die Möglichkeit sich mit Facebook anzumelden und zeigt auf einer Maps-Oberfläche bzw. in Listenform, kleinere Quests die in näherer Umgebung stattfinden und große, überregionale Quests an. Um an einer Quest teilnehmen zu können und den Suchradius der in der Quest versteckten Caches zu bekommen, muss der Konsument über die App den Werbebeitrag des Unternehmens, das die Quest erstellt hat, teilen.

11.1.4.2. Vorteile für Konsumenten

Natürlich bietet AdQuest nicht nur für Unternehmen, sondern auch für Konsumenten einige Vorteile. Die Nutzer der AdQuest-App kommen unter anderem in den Genuss folgender Punkte:

- ➔ Konsumenten werden animiert nach draußen in die Natur zu gehen
- ➔ Spannende, interaktive Suchabenteuer
- ➔ Reiz von Gratisprodukten

11.1.5. Wartung- & Weiterentwicklung

AdQuest beschäftigt als Unternehmen Angestellte, die für die Wartung, Weiterentwicklung bzw. für Management, Rechnungswesen und Controlling zuständig sind. AdQuest wird laufend von der Abteilung „Wartung & Entwicklung“ weiterentwickelt und auf dem neuesten Stand gehalten. Sind Erweiterungen geplant werden diese von der Entwicklung umgesetzt.
AdQuest hat darüber hinaus ein Support-System, durch welches Kunden bei Problemen, Anregungen oder Beschwerden direkt Kontakt mit der Abteilung „Wartung & Entwicklung“ bzw. Administration aufnehmen können.
Die Abteilung Marketing und Management ist verantwortlich für das Akquirieren von neuen Kunden und für das Gesamtbild des Unternehmens AdQuest nach außen.

11.1.6. Produktschutz

Grundsätzlich verbleiben alle Rechte an der Idee bei KREFT Steuerberatungs GmbH. Die Rechte am Projekt AdQuest selbst bleiben ebenfalls beim Auftraggeber und sind diesem zu übergeben.

11.1.7. Auswirkung auf Gesellschaft

Durch die Möglichkeit der günstigen Werbeschaltung und der Verbreitung speziell auf regionaler, örtlicher Ebene ist es nun auch kleinen Unternehmen möglich zu leistbaren und fairen Preisen Werbung zu machen. AdQuest arbeitet damit gegen das Aussterben von kleinen Familienbetrieben und unterstützt das Wachstum und die Umsatzsteigerung von neu gegründeten Firmen, die oft noch kein großes Werbebudget aufweisen.
Darüber hinaus animiert AdQuest die Konsumenten, sich in die Natur zu begeben, und mit der Aussicht auf Belohnung in Form von Gratisprodukten, Schätze zu suchen und sich auf Abenteuer zu begeben. AdQuest verwandelt so die sonst störende Werbung, wie etwa TV-Spots während Filmen, in ein Abenteuer das dem Konsumenten auch tatsächlich im Gedächtnis bleibt.

11.2. Markt

11.2.1. Branche

AdQuest ist im Bereich der Online-Werbeanbieter tätig. Es ist nicht ganz zu vergleichen mit einer Werbeagentur, da AdQuest seine Kunden weder berät noch bei der Planung ihrer Werbung unterstütz. AdQuest bietet lediglich ein System zur Vereinfachung und Publizierung der Werbung.

11.2.2. Zielgruppe

AdQuest ist im Rahmen eines sogenannten Two-Sided-Markets tätig. Das heißt AdQuest unterscheidet zwischen zwei Kundengruppen. Auf der einen Seite stehen Unternehmen welche die Hauptkunden darstellen, da sie die Produkte kaufen, auf der anderen Seite stehen die Konsumenten, welche die AdQuest-App benutzen und an den, von den Unternehmen erstellten Quests/Events teilnehmen. Diese zwei Gruppen werden auf der AdQuest-Plattform zusammengeführt.

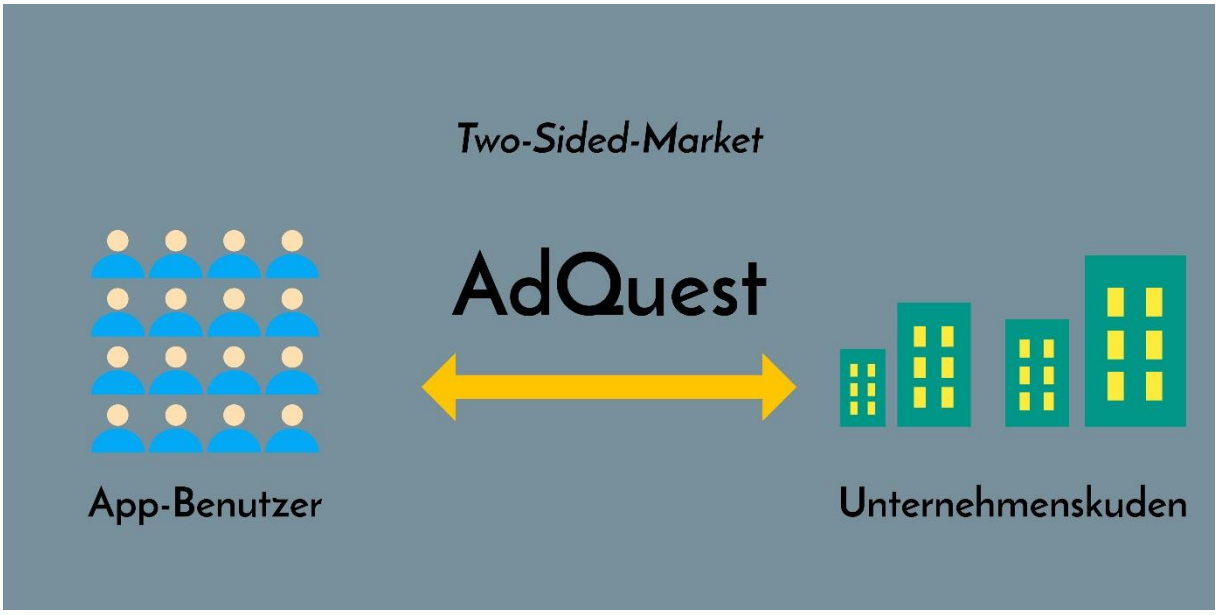


Abbildung 19 – Two-sided-Market

11.2.2.1. Zielgruppe App-Benutzer

Laut Statistiken von ut11.net besitzen 92% aller Mobiltelefon-Nutzer in Österreich ein Smartphone. 70% dieser Smartphone Nutzer, nehmen dabei Online-Werbung bewusst wahr.
Zudem antworten laut Statistiken von statista.com mehr als 30% auf die Frage "Verwenden Sie regelmäßig Coupon-Apps" mit "Ja" und weitere 30% antworteten zumindest mit "Teilweise". Dies zeigt das Gratisprodukte bzw. Vergünstigungen einen hohen Anziehungsfaktor auf den Konsumenten haben.
Das Alter unserer Zielkonsumenten bewegt sich zwischen 16 und 35 Jahren, da diese am Aktivsten an diversen Events und Veranstaltungen teilnehmen und neuen Technologien offener gegenüberstehen.

Zielgruppengröße

Statistiken zu Folge haben 27% der Erwachsenen zwischen 18 und 29 Jahren mehr als 500 Freunde auf Facebook.
Teilen bei einer Quest nur 120 Leute den Beitrag, sehen bereits ca. 21.000 Leute den Werbepost auch wenn 80% der Teilenden nur 250 Freunde haben. – Und diese Zahl steigt Jahr für Jahr.
Der Gesamtdurchschnitt von Facebook-Freunden liegt ca. bei 350 pro Benutzer. (Quellen: faz.net, statista.com).

Sharing-Verhalten

- Nutzer teilen in erster Linie Bilder gefolgt von ihrer persönlichen Meinung und Berichten, was und wie sie etwas gerade tun.
- **49 % teilen Informationen über Produkte, die ihnen gefallen**
- In den USA sind die drei Hauptkategorien für geteilte Inhalte: Humor, Sport, Politik
- Männer teilen in allen Netzwerken mehr als Frauen
- Jeden Tag werden 4,75 Milliarden Elemente auf Facebook geteilt

Der Durchschnittswert für das Teilen von Inhalten in sozialen Netzwerken beträgt 24%. Das bedeutet jeder 4. teilt Links etc. In Deutschland liegt dieser Wert bei 10% - demnach teilt in Deutschland trotzdem noch jeder 10. regelmäßig Links und Posts ohne dafür etwas zu bekommen. (Futurebiz, 2017)

Gründe für das Teilen von Nachrichten

- **42%** der Leute die Beiträge von Marken auf Facebook teilen tun dies um einen Coupon bzw. einen Rabatt zu bekommen.
- **49%** liken die Facebook-Pages von Marken und Unternehmen, weil sie die Marke unterstützen wollen

(Allfacebook.de, 2017)

Beispiel Zielgruppenrechnung

- City-Package – Unternehmen in Perg – Essensgutscheine
- 50 Shares müssen erreicht werden
- 7.620 Einwohner Perg

Ca. 25% der Österreicher sind zwischen 16 und 36 Jahren alt. Der prozentuelle Anteil der Facebook-Benutzer in diesem Alter ist zu vernachlässigen, da er bei ca. 99% liegt. Im absolut schlechtesten Fall teilen ca. 5% den Post.
50 * 20 = 1000 – Anzahl der Personen zwischen 16 und 36 die den Post sehen müssen.
1000 * 4 = 4000 – Anzahl aller Personen in der Region

AdSpreading

Gastronomiebetrieb in Perg ca. 100 Freunde (schlechter Fall) 5% Teilen den Beitrag => 5 Personen mit im Durchschnitt 300 Freunde (Normalfall) => 1500 neue Views und ca. 75 potentielle Teiler. 75 * 300 =22.500.

Konsumentengewinnung

Grundsätzlich werden die Konsumenten vom Reiz von Gratis-Produkte angezogen. Zusätzlich sehen sie auf Facebook den Werbepost der Unternehmen. Weitere Werbung für die Konsumenten ist von Seiten von AdQuest nicht vorgesehen.

11.2.2.2. Zielgruppe Unternehmer

Größeneinteilung

	Mitarbeiter	Umsatz
Kleinstunternehmen	Bis 9	< 2 Mio €
Kleinunternehmen	10 – 49	< 10 Mio. €
Mittlere Unternehmen	50 – 249	< 50 Mio. €
Großunternehmen	Ab 250	> 50 Mio. €

(vgl. wko.at, 2017)

Die Betriebe welche AdQuest hauptsächlich als Kunden gewinnen will bewegen sich im Bereich der Kleinst- bis Mittelunternehmen.
In Österreich gibt es laut dem Bundesministerium für Wissenschaft, Forschung und Wirtschaft ca. 330.000 Kleinst-, Klein- und Mittelunternehmen.

Unternehmensdifferenzierung

Nicht in die Zielgruppe von AdQuest fallen:

- Unternehmen die im B2B-Bereich tätig sind
- Unternehmen die per Auftragsfertigung produzieren/Verkaufen
    - z.B. Tischlereien
- Dienstleistungsunternehmen der Branche für Bau- und Haustechnik
    - z.B. Bauunternehmen, Spenglereien

11.2.3. Marktsegmentierung

AdQuest arbeitet hauptsächlich im B2B-Bereich mit gewinnbringender Absicht. Aufgrund des auf Social-Media basierten Konzeptes muss vor allem das Marktsegment der Online-Werbung bzw. im speziellen die Werbung mittels Facebook betrachtet werden.

11.2.3.1. Marktpriorisierung

- Priorität 1: Online-Werbung – speziell der Social-Media Bereich
- Priorität 2: App- bzw. Smartphone-Sektor – Werbung über die angezeigten Quests in der App

11.2.3.2. Kundensegmentierung

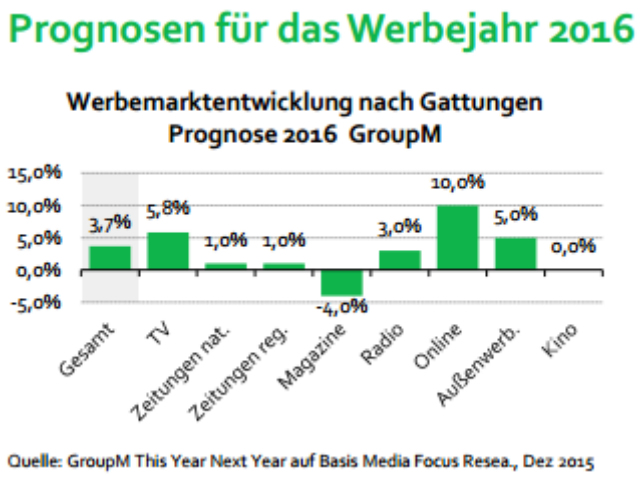
- Priorität 1: lokale Unternehmen die Regional im B2C Bereich operieren
    - als Kleinunternehmen werden hauptsächlich lokal operierende kleine Handels- oder Dienstleistungsunternehmen angesehen
    - Bsp.: Gastronomiebetriebe, kleine Lebensmittelgeschäfte – aber auch Franchise-Betriebe von großen Unternehmen wie etwa Nike-Stores, Hervis-Stores etc.
- Priorität 2: größere Unternehmen die International im B2C Bereich operieren
    - als Großunternehmen werden Unternehmen mit überregionalem Namen und überregionaler Bedeutung angesehen
    - Bsp.: Coca-Cola, BMW, Eristoff

11.2.4. Marktübersicht

11.2.4.1. Marktgröße

Derzeit entfallen in Österreich ca. 20% der Netto-Werbeausgaben auf Online-Werbung, wobei Experten in den nächsten 5-6 Jahren mit einem Wachstum von ca. 24% rechnen da der Online-Werbemarkt sehr im Kommen ist.
Der „Digital in 2016 Report“ von „We are Social“ brachte interessante Erkenntnisse zur Internetnutzung weltweit hervor. So nutzen 3,42 Milliarden Menschen das Netz. In Österreich sind es 7,14 Millionen und damit 83% der Bevölkerung. Während weltweit 31% der Internetuser soziale Netzwerke nutzen, sind es in Österreich 41%. Die ÖWA-Plus-Studie 2016-I besagt, dass nur noch 32,5% aller User in Österreich das Internet ausschließlich mit einem Standrechner verwenden. Die 14- bis 19-Jährigen nutzen das Internet zu 98,1% mobil.

11.2.4.2. Marktwachstum



Für das Jahr 2016 wurde ein Anstieg der Online-Werbung von 10% vorhergesagt. Diese Prognose ist auch eingetreten. Rund 528 Mio. € wurden 2016 in den Online-Werbemarkt investiert, 11,6 % mehr als im Jahr 2015(Quelle www.horizon.at, www.Werbeplanung.at). Besonders immer mehr junge Leute nutzen das Werbemedium Social-Media bzw. allgemein Online-Werbung. Auch in den kommenden Jahren werden von Experten massive Anstiege im Online-Werbemarkt zu verzeichnen sein weshalb dieser Markt eine sehr gute Einstiegsbasis bildet.

11.2.5. Marktbeurteilung

Der Online-Werbemarkt ist in den letzten Jahren immer mehr im Kommen. Insbesondere die junge Generation wird sehr stark von der Online-Werbung angesprochen, außerdem ist auch der Trend zu verzeichnen mit dem Handy nach draußen zu gehen („PokemonGO-Trend“) und Sachen zu unternehmen für die man belohnt wird („Gamification“ des Alltags; bei AdQuest zb. mit dem Finden von Caches).

11.2.6. Zahlen und Fakten

Nimmt man die in den Facebook-Statistiken erhobenen Zahlen her, nach denen jeder 10. Posts und Links auf Facebook teilt und nimmt an das anfangs nur 1 Nutzer mit 300 Freunden den Werbepost teilt, hätte man nach einem Share bereits 30 weitere Nutzer, die bereit wären den Link zu teilen. Es ergibt sich eine mehr als exponentielle Wachstumskurve in den Klicks.
Teilen bei einer Quest 120 Leute den Post von denen 30% mehr als 500 und 70% etwa 250 Freunde haben, sehen bereits ca. 21.000 Leute den Werbepost. - Und diese Zahl steigt Jahr für Jahr.

11.3. Konkurrenz

11.3.1. Konkurrenten

Die wichtigsten Konkurrenten von AdQuest sind prinzipiell alle Unternehmen die Online vor allem Billigwerbung anbieten. Da es aber in diesem Bereich nicht sehr viele Mitkonkurrenten gibt ist die Situation für AdQuest auf diesem Markt sehr gut.
Die nächsten nennenswerten Konkurrenten sind Events und Feste, die durch Sponsoring für lokale Unternehmen gute Werbemöglichkeiten bieten oder lokale Medien wie Regionalzeitungen. Weniger Konkurrenz sind die klassischen Werbeträger wie Radio- oder TV-Anstalten. Aufgrund der großer Reichweite und den daraus resultierenden hohen Kosten für Werbeeinschaltungen sprechen diese andere Kunden an.

11.3.2. Konkurrenzveranstaltungen am Beispiel „Gamsleiten-Kriterium“

Bei diesem Event im Schigebiet Obertauern werden ein Autoschlüssel, sowie viele andere attraktive Preise auf der Piste im Schnee vergraben. Die Leute schwärmen aus und suchen besagte Schätze um sie zu beanspruchen, Hauptpreis dabei ist ein Auto. Veranstaltet von BMW zieht dieses Event jährlich über 1000 Besucher an – entsprechen groß ist dabei der Werbeeffekt von BMW.

11.4. Marketing & Vertrieb

11.4.1. Marketingziele

- Jahr 2017: kleine lokale Unternehmen im Bundesbereich erschließen
  - Ziel: jedes Monat ca. 4-5 Neukunden
- Jahr 2018: Klein- und Mittelbetriebe im Bundesbereich erschließen
- Jahr 2019: Erschließung des Werbegebietes Süddeutschland, evtl. auch internationale Aufträge

11.4.2. Markteinführungsplan

Da AdQuest besonders für Kleinunternehmen eine Möglichkeit der günstigen Werbeschaltung bietet, sind diese am Anfang auch die primäre Zielgruppe. Gleichermaßen mit der Verteilung der Unternehmen die Aufträge an AdQuest erteilen steigt auch die Anzahl bzw. die Verbreitung der AdQuest App was eine exponentielle Steigerung der Bekanntheit und er Einnahmen zur Folge haben wird.

11.4.2.1. Einführungsevent

Startevent für AdQuest wird ein Event des Linzer Steuerberaters Kreft sein. Dieses Unternehmen wird auch der erste Kunde sein, der das AdQuest-System nutzt.
Ein weiterer Fixkunde den das Unternehmen derzeit verzeichnen kann, ist das in Linz ansässige italienische Restaurant „Rosso di aqua el sole“.
Nach Beendigung der ersten Events wird sofort mit der Neukundenerschließung durch die Marketingmanager von AdQuest gestartet.

11.4.2.2. Marktkommunikation

Um den Bekanntheitsgrad insbesondere beim Hauptkundensegment, den Klein- und Mittelbetrieben zu steigern, will AdQuest hauptsächlich das eigene System verwenden. Da AdQuest im Bereich Online-Marketing tätig ist und so, viele Menschen erreichen kann reichen kleine Werbebotschaften in Verbindung mit der Aktivität der Kunden.
Hauptwerbeziele von AdQuest sind:

- Priorität 1: Klein- und Mittelunternehmen
- Priorität2: Konsumenten, die noch nicht AdQuest verwenden um mehr Werbepublikum für Kunden zu erschließen

11.4.2.3. Werbemaßnahmen

Hauptwerbemaßnahme von AdQuest ist der Social-Media Bereich, der auch die Grundlage des Unternehmenskonzeptes bildet.
Hintergrund dabei ist es über die Werbung, die Kunden über das System verbreiten, einen gewissen Anteil von Selbstwerbung einzubringen. Ziel dieser Selbstwerbung ist der Effekt durch den andere Klein- und Mittelunternehmen die auf Facebook aktiv sind, auf das System AdQuest aufmerksam werden.

11.4.2.4. Selbstwerbung

Ziel des Selbstwerbeeffektes von AdQuest ist es, den Bekanntheitsgrad unter anderen, auf Facebook tätigen, potenziellen Kunden zu erhöhen. Jeder Werbepost eines Kunden von AdQuest muss auch einen Verweis auf das Unternehmen AdQuest selbst enthalten. Dadurch erfahren andere Unternehmen von AdQuest und können die Wirkung der Werbebotschaften auf Facebook live miterleben.

11.4.3. Verkauf / Vertrieb

11.4.3.1. Zahlungsart

Abgerechnet wird über PayPal, der Kunde ist somit verpflichtet ein funktionierendes PayPal-Konto vorweisen zu können.
Des Weiteren, kann die Zahlung auch per Rechnung erfolgen falls das Produkt „Event“ gewählt wurde, diese wird am Stichtag (Tag des Event-Startes) zugesandt.

11.4.3.2. Zahlungskonditionen

Bei Zahlung per Rechnung legt AdQuest als Zahlungsziel die **Zahlung binnen 10 Tagen** ab Rechnungsdatum fest. Bei Zahlung innerhalb dieser Frist wird ein **Skonto von 2%** gewährt.

11.4.3.3. Lieferkonditionen

AdQuest liefert dem Kunden eine druckbare PDF-Datei mit den jeweiligen QR-Codes für die einzelnen Caches. Für den weiteren Umgang mit diesen ist der Auftraggeber selbst verantwortlich, ebenso für die Platzierung des Caches. AdQuest übernimmt keine Haftung für unsicher platzierte Caches oder Verletzungen die im Zuge der Cache-Suche entstehen, der Auftraggeber wird diesbezüglich hingewiesen und akzeptiert diese Bedingungen auch bei dem Kauf bzw. bei der Bestätigung der AGBs von AdQuest.

11.4.3.4. Lieferung Werbevideo

Sofern erwünscht filmt eine AdQuest-Drohne die Suche bzw. den Fund der Caches und speichert diese Werbevideo auf den Servern des Unternehmens. Das Werbevideo wird anschließend in Rohform an den Auftraggeber übergeben.

Dies ist jedoch nur bei einem Event möglich und als Zusatzoption bei diesen buchbar. Die Planung erfolgt hierbei mündlich bzw. telefonisch.

11.4.3.5. Lieferung Statistiken

Kunden von AdQuest haben neben den Leistungen des Systems auch das Recht auf alle Statistiken, bzw. Zahlen die im Rahmen einer Quest erhoben wurden. Im Detail sind das etwa:

- tatsächliche Shares
- Views
- Dauer der Quest
- usw.

Diese Statistiken werden dem Kunden nach der Erhebung, am Ende der Quest per Email zugesandt.

11.4.3.6. Rechtliche Situation

Grundsätzlich ist laut Gesetz das Filmen von Personen ohne der Erlaubnis der betreffenden Person verboten. Jeder User der AdQuest App, welche nötig ist um Caches zu finden und gefilmt zu werden, muss beim Herunterladen und erstmaligen Starten der App die AdQuest AGBs bestätigen. Diese beinhalten einen Artikel der AdQuest es bei Bestätigung erlaubt Aufnahmen der bestätigenden Person zu machen. Dies ist jedoch nur während der aktiven Suche, also während die Person die AdQuest-App aktiv benutzt der Fall.

11.4.4. Kundengewinnung

11.4.4.1. Aktives Kundenmanagement

Die Marketingmanager von AdQuest bemühen sich um ein aktives Kundenmanagement, gehen gezielt auf Kunden zu, befragen, werten Feedbacks aus und verfassen konkrete Verbesserungsvorschläge für das gesamte System.

Darüber hinaus steht dem Kunden bei der Buchung des „Big-Package“ Paketes während den Geschäftszeiten ein Kundenmanager zur Verfügung der Fragen klärt und den Kunden bei der Planung und Durchführung seiner Quest zur Seite steht. Dies ist ebenfalls bei einem Event der Fall.

11.4.4.2. Neukundenerschließung

Neben der Selbstwerbung durch den normalen Betrieb des AdQuest-Systems gehen Marketingmanager auch gezielt auf Unternehmen zu die noch nicht Kunde bei AdQuest sind. Sie kontaktieren diese Unternehmen per Email bzw. Telefon oder nehmen persönlich Kontakt auf. Sie verhandeln mit diesen und überzeugen die Kunden von den Vorteilen des AdQuest-Systems.

11.5. Erfolgs & Finanzplanung

11.5.1. Rechtsform

AdQuest wird als GmbH mit dem Namen AdQuest GmbH von Herrn Mag. Gerald Kreft gegründet.
Es liegt derzeit nur eine Beteiligung vor – DI Dietmar Wokatsch-Ratzberger (Inhaber der Idee).

11.5.2. Kostenplan Gründung

Quelle des Kapitels 11.5.2: (vgl. Gründerservice, 2017)

11.5.2.1. Gründungskosten

Für die Gründung einer GmbH fallen eine Reihe von Kosten an, die im Folgenden aufgelistet und anschließend summiert werden.

11.5.2.2. Vertragserrichtung

- Vertragserrichtung Notar (notariatspflichtig): **ca. 2000 €**
    - abhängig vom Umfang der Gesellschaft und des Vertrages

11.5.2.3. Gewerbeanmeldung von reglementierten und freien Gewerben

Für die Anmeldung eines Gewerbes ("Dienstleistungen in der automatischen Datenverarbeitung und Informationstechnik") fallen folgende Kosten/Gebühren an:

- Eingabegebühr: **47,30 €**
- Gebühr für den Gewerberegisterauszug: **7,20 €**
- Beilagen (pro Bogen): **3,90 €**
- Verwaltungsabgaben: **2,10 €**
- Anzeige eines gewerberechtlichen Geschäftsführers: **14,30 €**

11.5.2.4. Firmenbucheintrag

- Firmenbucheintrag einer GmbH **ab 400 €**

11.5.2.5. Kostenbefreiung laut NeuFöG - Neugründungsförderung

Laut NeuFöG liegt eine betriebliche Neugründung unter folgenden Voraussetzungen vor:

- Neueröffnung eines
  - gewerblichen
  - land- und forstwirtschaftlichen oder
  - dem selbstständigen (freiberuflichen) Erwerb dienenden Betriebes durch Schaffung einer bisher nicht vorhandenen betrieblichen Struktur.
- Der Betriebsinhaber hat sich innerhalb der letzten 5 Jahre nicht in vergleichbarer Art (in einer vergleichbaren Branche) sowohl im Inland als auch im Ausland betrieblich betätigt.
- Es handelt sich nicht nur um eine Änderung der Rechtsform.
- Es liegt kein bloßer Wechsel in der Person des Betriebsinhabers vor, egal, ob es sich dabei um eine entgeltliche oder unentgeltliche Betriebsübertragung handelt.
- Die geschaffene betriebliche Struktur darf innerhalb eines Jahres ab Neugründung nicht um bestehende andere Betriebe oder Teilbetriebe erweitert werden.

11.5.2.6. Tatsächliche gründungsbezogenen Kosten

Da AdQuest die vom NeuFöG geforderten Voraussetzungen zur Kostenbefreiung erfüllt, treten folgende gründungsbezogene Kosten auf:

- Vertragserrichtung Notar (notariatspflichtig): **ca. 2000 €**
  - abhängig vom Umfang der Gesellschaft und des Vertrages

11.5.3. Investitionsplan

11.5.3.1. Investitionsplan Gründungsjahr

Investition	Kosten	Anmerkung
EDV-Ausstattung		
1 Drucker/Scanner	300,00 €	
5 Windows PCs + Zubehör	3000,00 €	
1 MacBook Pro + Zubehör	2400,00 €	
Lizenzen und Rechte		
Office-Lizenzen	200,00 €	
Affinity-Designer	50,00 €	
JetBrains Webstorm	206,00 €	
5 Windows-Lizenzen	135,00 €	
Büroausstattung		
5 Schreibtische + Sessel	700,00 €	
div. Büroartikel	200,00 €	
Drohne		
DJI Phantom 4	1300,00 €	
EDV-Netzwerk		
div. Netzwerkkomponenten	120,00 €	
	8611,00 €	

11.5.3.2. Laufende Kosten

Investition	Kosten	Anmerkung
Server		
Mietkosten	7,50 €/Monat	
Domain	1,16 €/Monat	Jährlich 13,60 €
Lizenzen		
BitBucket-Pro Lizenz	10,00 €/Monat	
Büro		
Betriebskosten	250,00 €/Monat	
div. Büroartikel (Papier, etc.)	100,00 €/Monat	
Büromiete	140,00 €/Monat	Mietförderungsmodell Softwarepark Hagenberg *
Drohne		
Drohnenversicherung	11,60 €/Monat	139 €Jährlich zu entrichten
	519,10 €/M	

11.5.3.3. Startkapital

Alle in den vorangegangenen Kapiteln aufgelisteten Kosten zusammengefasst beläuft sich die Summe der Kosten bzw. Investitionen im 1. Monat des Gründungsjahres auf: **10.611 €**

Bezieht man auch die laufenden Kosten mit ein, betragen die Kosten ca. **11.000 €**

Ein realistisches Startkapital wäre aufgrund dieser Berechnung ein Betrag im Bereich von

ca. **10.000 – 15.000 €**

11.5.3.4. Investoren

AdQuest bemüht sich im Rahmen diverser Wettbewerbe und Einführung- bzw. Probeevents so viele Investoren wie möglich von der Geschäftsidee zu überzeugen. Ziel ist es bis 15.05.2017 zumindest 70% des erforderlichen Startkapitals aufgebracht zu haben.

11.5.4. Umsatz & Erfolgsplan

11.5.4.1. Jährliche Kosten

Summiert man die im Monat entstehenden laufenden Kosten auf 12 Monate auf entstehen jährliche Kosten von ca. **4550,00 €**
Bezieht man einen gewissen Betrag (ca. 1000 €) – vorgesehen für außerordentliche Kosten
im Jahr – mit ein liegt die Summe der jährlichen Kosten bei ca. **5550,00 €**

11.5.4.2. Umsatz & Geschäftsziele

	Ziele
Jahr 1	**Umsatzziel: 6000,00 €**
Quartal 1	• Eröffnungsevent • 5-6 Aufträge monatlich • Verluste gering halten
Quartal 2	• 5-6 Aufträge monatlich
Quartal 3	• 7-8 Aufträge monatlich • erste Aufträge von Oberösterreichweiten Unternehmen
Quartal 4	• 8-9 Aufträge monatlich • erste Anfragen von Unternehmen in ganz Österreich
Jahr 2	**Umsatzziel: 35.000 €**
Jahr 3	**Umsatzziel: 100.000 €**

11.5.4.3. Investitionen in Folgejahren

Jahr 2	• keine größeren Investitionen geplant • Gehaltsverzicht einstellen → Auszahlung von Gehältern
Jahr 3	• Einstellung von 2 zusätzlichen Mitarbeitern

11.5.5. Personal - Gründungsjahr

11.5.5.1. Gehälter im Gründungsjahr

Sowohl die Geschäftsführung als auch die Entwickler und Buchhalter werden bis zum 3. Quartal im 2. Geschäftsjahr nur das nötigste Gehalt in Anspruch nehmen bzw. zum Wohle des Unternehmens **komplett auf die Auszahlung des Gehaltes verzichten.**

- **GF**: 1500€/Monat – brutto
- **restliches Unternehmensteam**: 1200€/Monat - brutto

11.5.5.2. Gehaltsverzicht - Grundlagen

Nur beitragsrechtlich gültig, wenn folgende Kriterien erfüllt:

- arbeitsrechtlich zugelassen
- schriftlich vereinbart
- Zusammensetzung und Höhe des Entgeltes muss genauestens schriftlich niedergelegt sein
- Verzicht gilt nur für künftiges Arbeitsentgelt

11.5.5.3. Gehaltsverzicht, steuerliche Besonderheiten

Im Steuerrecht gilt das Zuflussprinzip. Damit unterliegt nur der geminderte Arbeitslohn dem Lohnsteuerabzug (§ 38 Abs. 2 EStG)

11.5.5.4. Gehaltsverzicht – versicherungstechnische Besonderheiten

Die Beiträge werden fällig, wenn der Anspruch des Arbeitnehmers auf das Arbeitsentgelt entstanden ist. Man spricht aus diesem Grund auch von Anspruchsprinzip. Er muss auch für geschuldetes, aber noch nicht gezahltes Entgelt Versicherungsabgaben entrichten.

11.5.5.5. Gehaltsverzicht – Motivation

Alle Mitarbeiter, welche in den ersten 2 Jahren in der AdQuest GmbH tätig sind, zählen zum Gründungsteam. Da es die Intention von Gründern ist, ein neues Unternehmen bzw. eine Idee in relativ geringer Zeit gewinnbringend zu machen, ist dies die Motivation der Mitarbeiter, zum Wohle des Unternehmens auf ihr Gehalt zu verzichten.

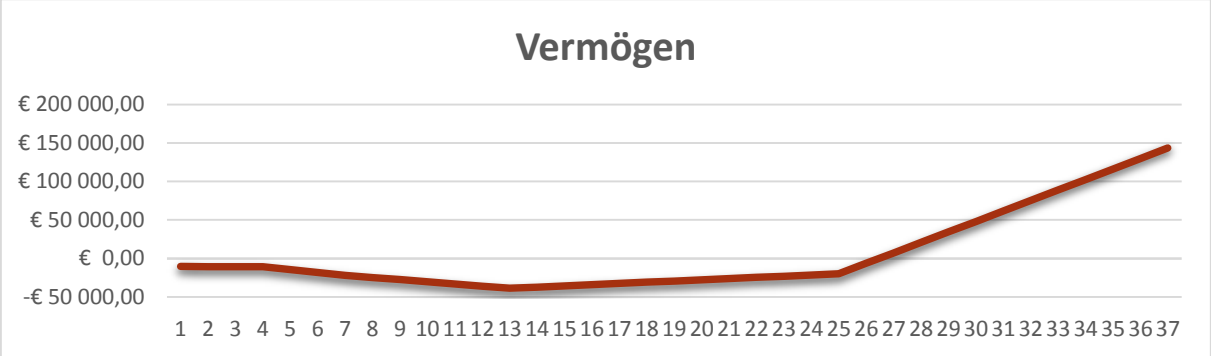
11.5.5.6. Gehälter in Folgejahren

Das Ende des Verpflichtenden Gehaltsverzichtes für die Mitarbeiter endet mit dem Ende des 3. Monats.
Ab diesem Zeitpunkt werden folgende Gehälter ausbezahlt:

- Normalangestellte: 1722 **€/Brutto laut Kollektivvertrag**

(vgl. Lohn-Info.de, 2017)

11.5.5.7. Grober Finanzplan Jahre 1-3



Monat 2: Einnahmen: 6 City Pakete und 0 Medium Pakete

Monat 7: Einnahmen: 7 City Pakete und 1 Medium Pakete

Monat 13: Einnahmen: 16 City Pakete und 5 Medium Pakete

Monat 25: Einnahmen: 50 City Pakete und 15 Medium Pakete

Bis Monat 3 – Gehaltsverzicht.

Wichtige Anmerkung: Bei Folgender Rechnung ist das Vermögen ohne Steuern aufgeführt!

AdQuest	Einnahmen	Ausgaben	Summe (Vermögen)
Startinvestition + Gründung		€ 10 611,00	-€ 10 611,00
1. Monat	€ 594,00	€ 640,00	-€ 10 657,00
2. Monat	€ 594,00	€ 640,00	-€ 10 703,00
7. Monat	€ 1 562,25	€ 4 362,00	-€ 24 852,75
13. Monat	€ 5 930,25	€ 4 362,00	-€ 37 283,25
25. Monat	€ 17 988,75	€ 4 362,00	-€ 6 405,75
36. Monat	€ 17 988,75	€ 4 362,00	€ 143 488,50

11.6. Preispolitik

11.6.1. Grundsätzliche Entstehung des Preises

AdQuest verrechnet seinen Kunden die Werbekosten per Klick + einen Aufschlag per View. Beim Anlegen eines Quests vergibt eine Firma eine Mindestuserzahl ab der der Quest
startet – diese Zahl fungiert als Basiswert für die Pro Klick Verrechnung. Darauf aufgeschlagen werden die Kosten des Pro View Werbeeffekts welcher durch das zwingende Teilen des Beitrages auf Facebook entsteht, durch den Freunde des Users, der die Werbung geteilt hat den Beitrag zumindest sehen.

11.6.1.1. Angebotene Produkte

AdQuest bietet seinen Unternehmenskunden eine Dienstleistung in Form der Bereitstellung des AdQuest-Systems, zur Verwaltung von Quests und Events.
Bei der Preiskalkulation werden dabei 2 verschiede „Produkte“ voneinander unterschieden. Sogenannte „Quests“ und sogenannte „Events“.
Bei Quests wird der Preis aus der vorher festgelegten Benutzerschwelle, die überschritten werden muss um den Quest zu starten, errechnet.
Bei Events erfolgt die Preiskalkulation und Abrechnung erst am festgelegten Start-Datum des Events und wird dem Kunden per Rechnung zugestellt. Der Betrag ergibt sich aus der tatsächlichen Teilnehmerzahl am Abrechnungstag.

11.6.2. Preiskalkulation

11.6.2.1. Fixkosten

Siehe Laufende Kosten 11.5.3.2.
Die Fixkosten pro Monat belaufen sich auf ca. 520,00 €.

11.6.2.2. Variable Kosten

Da in Abhängigkeit der angenommenen Kundenaufträge für AdQuest kein Mehraufwand entsteht werden keine Variablen Kosten in die Preiskalkulation eingerechnet.

11.6.2.3. Abschreibungskosten

Monatlich fallen für AdQuest Abschreibungskosten in der Höhe von: **110,00 €**

11.6.2.4. Preisberechnung – Pakete

AdQuest muss in den ersten 2 Jahren der Gründung somit monatlich Kosten in der Höhe von **ca. 500 €** decken.

Paket „City“: vorgefertigtes Paket mit einem Userminimum von 50 Usern und 1-2 Caches

• Systemnutzungskosten	19,00 €
• 50 shares á 1,20€/Klick	60,00 €
• 50 shares á 0,40€/View	20,00 €
	99,00 €

Paket „Medium-Package“: vorgefertigtes Paket mit einem Userminimum von 500 Usern und 1-5 Caches

• Systemnutzungskosten	115,00 €
• 500 shares á 1,20€/Klick	600,00 €
• 500 shares á 0,40€/View	200,00 €
	915,00 €
• Paketrabatt 5%	- 45,75 €
	869,25 €

Paket „Big-Package“: vorgefertigtes Paket mit einem Userminimum von 2500 Usern und 1-15 Caches

• Systemnutzungskosten	215,00 €
• Persönliche Betreuung	650,00 €
• 2500 shares á 1,20€/Klick	3000,00 €
• 2500 shares á 0,40€/View	1000,00 €
	4865,00 €
• Paketrabatt 5%	- 243,25 €
	4621,75 €

11.6.2.5. Preisberechnung – Events

Events haben im Gegensatz zu Quests kein fix definiertes Userminimum um zu starten. Events haben einen Festen Start- und Endzeitpunkt und werden erst am Stichtag des Startzeitpunktes mit der bis dahin angesammelten Teilnehmerzahl berechnet.

Preisberechnung Event:

• Systemnutzungskosten	425,00 €
• shares am Stichtag á 1,20 €/Klick	X
• shares am Stichtag á 0,40€/View	Y
• Bearbeitungsaufwand	260,00 €
	425,00 + X + Y + 260,00 €

11.6.2.6. Preisberechnung – Custom Quests

Custom Quests oder Custom-Pakete sind für Kunden bestimmt die nicht im normalen Paket-Angebot fündig werden. Bei Custom Quests kann die Userschwelle sowie die Anzahl der Caches vom Kunden selbst Festgelegt werden.

• Systemnutzungskosten	19,00 €
• Userzahl á 1,20€/Klick	X
• Userzahl á 0,40€/View	Y
• Bearbeitungsaufwand	20,00 €
	19,00 + X + Y + 20,00 €

11.6.2.7. Aufschlag bei Drohnenbuchung

Es gibt bei jeder Quest bzw. jedem Event auch die Möglichkeit der Buchung einer Drohne. Diese Drohne filmt die Schatzsuche sobald sich User dem Cache auf eine bestimmte Distanz nähern. Das Werbevideo wird dem Kunden nach Beendigung der Quest bzw. des Events zur Verfügung gestellt. Um Unfälle zu vermeiden und Haftungsfragen zu klären wird das Filmen der Drohnen von einem AdQuest Mitarbeiter übernommen. Dieser kann nur zu einem Event gebucht werden und der Kunde muss stündlich für ihn bezahlen.

• Kosten für Drohnenmiete	**120,00/h €**

11.7. Unternehmen / Management

11.7.1. Unternehmensstandort (Gründungsjahr)

Unternehmensstandort ist der Softwarepark Hagenberg. AdQuest wird das Mietförderungsmodell des Softwareparks Hagenberg für junge Startups in Anspruch nehmen.

11.7.2. Unternehmensstruktur (Gründungsjahr)

11.7.2.1. Stellen - Gründungsjahr

- **Technik und Entwicklung**
  - 2 Stellen | Teilzeit(10-20h)
- **Marketing und Vertrieb**
  - 1 Stellen | 40h
  - 2 Stelle | Teilzeit (10-20h)

11.7.3. SWOT-Analyse

SWOT-Analyse AdQuest	
Strengths	Weaknesses
+ qualifiziertes Personal	– keine Referenzprojekte
+ extrem erweiterbares System	– zeitaufwendige Entwicklung
+ viele Kooperationen möglich	– sehr komplexes System
+ Neuartigkeit	– stark Risikobehaftet
Opportunities	Threats
+ keine Konkurrenz am Markt	– Kunden nehmen System nicht an
+ ganzes Marktsegment steht offen	– schlechtes Feedback
+ branchenunabhängige Software	– unzufriedene Kunden
+ Potenzial für Expansion ins Ausland	– Konsumenten teilen nicht

11.7.4. Geschäft

11.7.4.1. Was verkauft AdQuest?

AdQuest ist als Online-Werbe-Dienstleister tätig. Das Unternehmen verkauft die Nutzung des AdQuest-Systems zur Publizierung von Werbebotschaften auf Facebook. Im Konkreten kaufen Kunden bei AdQuest sogenannte Quests und Events, erstellen einen Werbepost und veröffentlichen diesen.

11.7.4.2. Wertschöpfung

Durch die geringen Kosten die für AdQuest bei Nutzung des Systems anfallen, ist es möglich Kunden sehr günstige Werbeangebote zu schaffen. Die Wertschöpfung liegt im Verkauf der Quests und Events bzw. in der kostenpflichtigen Nutzung des AdQuest-Systems.

11.7.4.3. Kunden

Die Verkaufsstrategie von AdQuest ist fokusiert auf regional operierende Klein- und Mittelnternehmen die im B2C-Bereich tätig sind.

11.7.5. Ziele

11.7.5.1. Geschäftsziele

Ziel von AdQuest ist es im Gründungsjahr zumindest in den ersten Monaten keine Verluste zu erzielen. Im 1. Jahr nach der Gründung nimmt AdQuest es sich zum Ziel mindestens einen Reingewinn von ca. 2000€ monatlich zu erwirtschaften und diesen Betrag Monat für Monat zu erhöhen.

Genauere Beschreibung der Umsatz & Zielplanung erfolgt
im Kapitel **6.4.3 Umsatz & Geschäftsziele**

11.7.5.2. Unternehmensziele

Das wesentlichste Unternehmensziel von AdQuest im Gründungsjahr wird es sein, so viele Kunden wie möglich zu gewinnen und den Name „AdQuest“ bekannt zu machen.
Die Unternehmensziele grob nach Jahren gestaffelt werden folgendermaßen definiert:

- Jahr 1:
  - Gewinnung möglichst vieler Kunden im Bezirk Perg, und später im ganzen Mühlviertel und Linz
  - Bekanntmachung des Names „AdQuest“ im Bereich Perg, Linz, Mühlviertel
- Jahr2:
  - Gewinnung von Kunden im Bereich Oberösterreich bzw. auch Bundeslandübergreifend
  - erste österreichweite Aufträge erlangen
  - erste Langzeitverträge mit kleineren Kunden schließen
- Jahr 3:
  - Verbreitung des Namens „AdQuest“ im Bereich Süddeutschland
  - Erschließung erster Kunden im Ausland – forciert Deutschland

11.8. Risikoanalyse

11.8.1. interne Risiken

- nicht ausreichendes technisches Know-How
- Unternehmensinterne Unstimmigkeiten

11.8.2. externe Risiken

11.8.2.1. Finanzelle Risiken

Da es dem Unternehmen bei der Gründung an Eigenkapital fehlt, ist AdQuest auf Investoren angewiesen.
Sollten sich nicht genügend Investoren finden oder sollten diese von der Geschäftsidee „AdQuest“ nicht überzeugt sein, könnte dies ein Risiko für die gesamte Gründung des Unternehmens darstellen.

11.8.2.2. Produktbezogene Risiken

Sollte das das System „AdQuest“ seinen Nutzen verfehlen und die Konsumenten nicht in der Form ansprechen, in der es das Geschäftskonzept vorsieht, könnten sowohl Kunden als auch Investoren sich von AdQuest zurückziehen und das Unternehmen würde scheitern.

11.8.2.3. Maßnahmen

Treten produktbezogenen Risiken auf führt AdQuest sofort Umfragen durch und wertet Feedback aus, um das Produkt zur Zufriedenheit des Kunden und der Konsumenten zu verbessern, zu erweitern oder umzubauen.
AdQuest wird in den ersten Jahren der Gründung keine Gewinnrücklagen tätigen. Das ist riskant, jedoch aus finanzieller Sicht besser für das Unternehmen.

12. Realisierung

12.1. Vorgehensmodell - SCRUM

Die Diplomarbeit wurde auf Grundlage der agilen Vorgehensmethode „SCRUM“ entwickelt

SCURM bezeichnet ein häufig in der Softwareentwicklung eingesetztes sogenanntes „agiles Vorgehensmodell“. Agil deshalb, da es mit SCRUM möglich ist sehr schnell auf Änderungen des Funktionsumfanges bzw. der Projektstruktur zu reagieren. Des Weitern basiert SCRUM auf dem iterativen Ansatz, d.h. es wird in Zyklen fest definierter Länge gearbeitet, an deren Ende immer ein fertiger Release des Produktes steht.

12.1.1. Abwandlung von SCRUM

Nach einigen Diskussionen hat das Projektteam einstimmig beschlossen, nicht das klassische SCRUM zu verwenden, sondern eine kleine, für das Projekt zurechtgeschnitzte Abwandlung zu schaffen.
Bei AdQuest baut ein Sprint, also die regelmäßigen Zyklen, nicht mehr auf sogenannten „User-Stories“ auf, sondern ist direkt mit Tasks gefüllt, die bis zum Ende des Sprints erledigt sein müssen. Des Weitern muss am Ende eines Sprints nicht unbedingt ein verwendbarer Release des Produktes vorhanden sein, es müssen lediglich alles Tasks belegbar abgeschlossen sein.

12.1.2. Gründe für die Abwandlung

Da seitens des Auftraggebers keine wirklichen Einschränkungen bezüglich der Gestaltung des entwickelten Systems vorgebracht wurden, war der ständige Release-Zwang, zwecks Abgleichung mit den Vorstellungen des Auftraggebers, nicht mehr nötig und wurde somit ausgelassen.

12.2. Meilensteine

19.09.2016	•Kick-Off
30.11.2016	•Benutzeroberfläche fertig •Datenschnittstelle fertig
31.12.2016	•Social Media Shares implementiert •Serveranbindung implementiert
15.01.2017	•Speicherung Werbevideo
31.01.2017	•Design angewendet
15.02.2017	•QR-Code System fertig
16.03.2017	•Zahlungssystem implementiert
16.03.2017	•Drohnensteuerung fertig

13. Server – Backend

13.1. Grundidee

Ein RESTful Webservice dient als zentrale Komponente im AdQuest-System, auf welche sowohl die Website als auch die Android-App zugreift, um Daten zu erhalten bzw. Daten zu speichern. Alle benötigten Daten werden mittels JPA in der MySQL-Datenbank persistiert und gelesen. Damit die Website und App auf die REST-API zugreifen können, wird ein Linux-Server konfiguriert und mit folgenden Komponenten ausgestattet:

- Apache2
    - Webserver, auf welchem sich die AdQuest-Website befindet
- MySQL-Datenbank
- Wildfly
    - Applikationsserver, auf welchem sich das Webservice und Admin-Interface befindet
    - Verbindung zur MySQL-Datenbank

13.2. Architektur

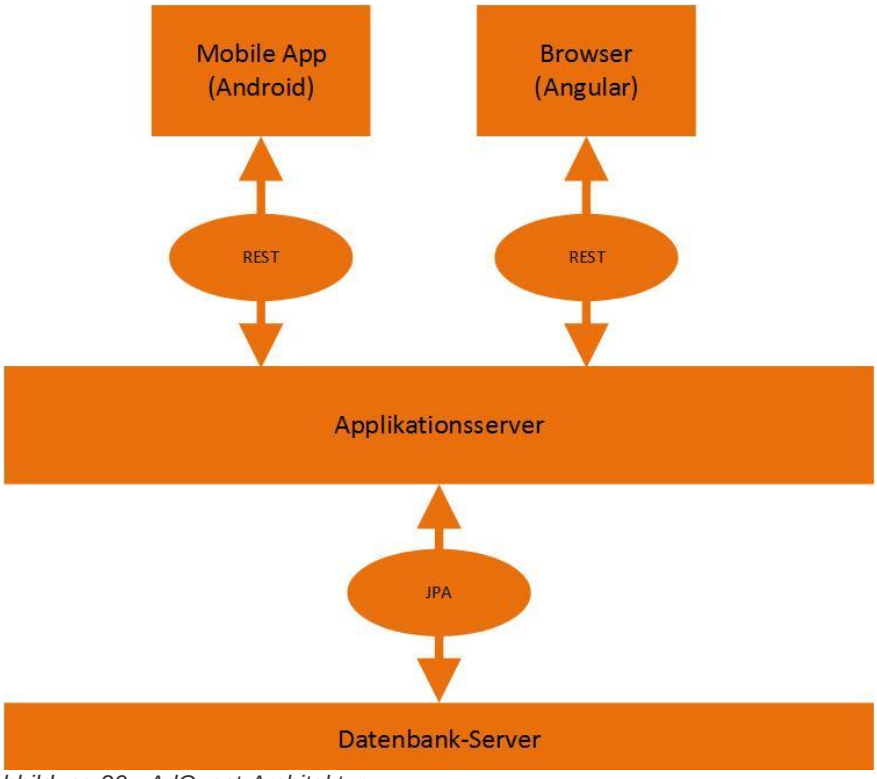


Abbildung 20 - AdQuest-Architektur

Die mobile App und die Website greifen über die REST-API (http-Request) auf den Applikationsserver zu, um Daten anzufordern, zu speichern, sie zu ändern oder zu löschen.

13.3. Angewandte Techniken

13.3.1. Java

Siehe Kapitel 10.4.2 Java

13.3.2. Wildfly

Siehe Kapitel 10.4.5 Wildfly

13.3.3. Resteasy

Siehe Kapitel 10.4.7 Resteasy

13.3.4. JPA

Siehe Kapitel 10.4.3 Java Persistence API (JPA)

13.3.5. MySQL-Datenbank

Siehe Kapitel 0 Resteasy ist ein Framework mit dem RESTful Web Services in Java implementiert werden können. (vgl. jaxenter.de, 2017)

MySQL

13.4. Datenmodell

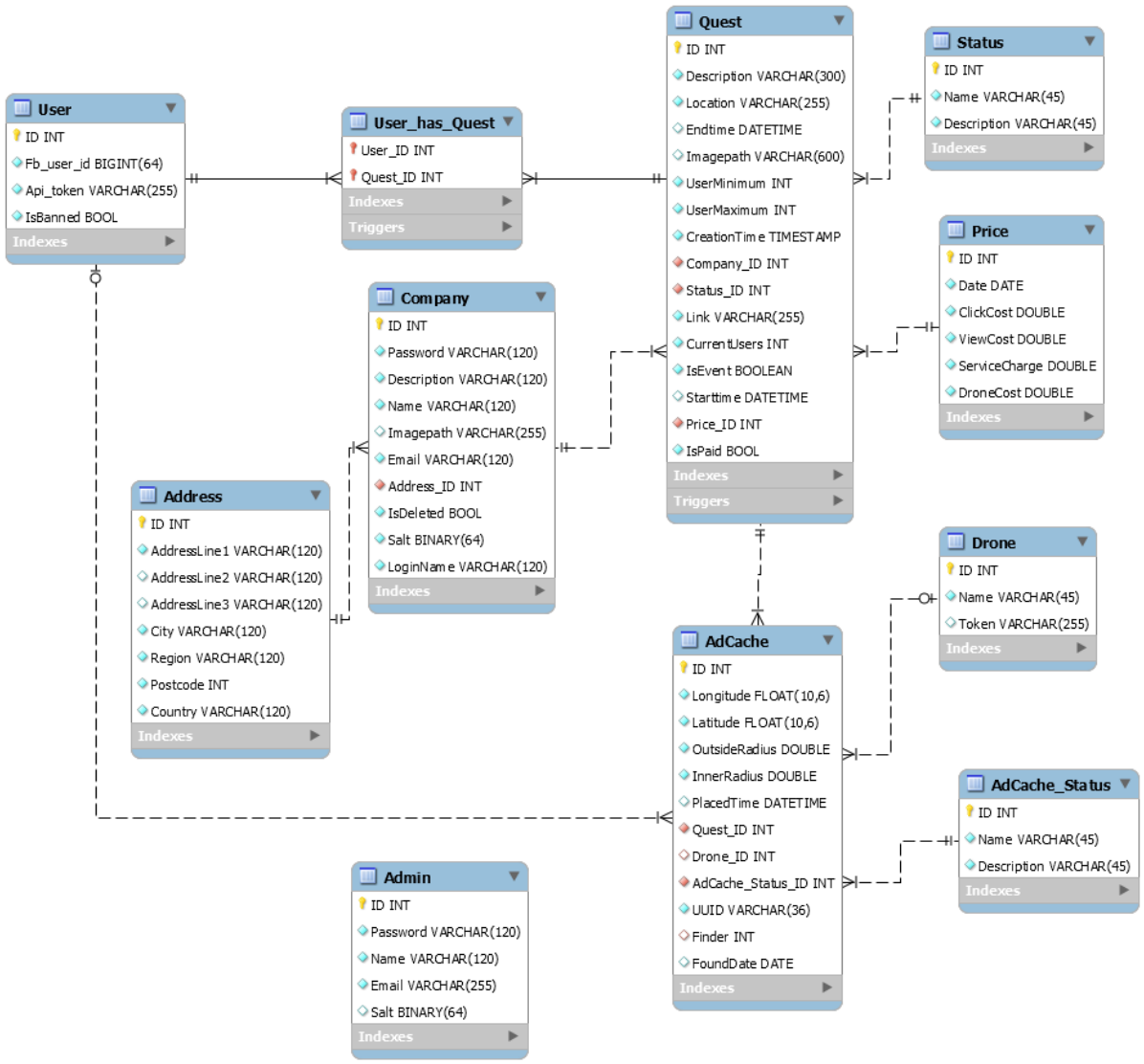


Abbildung 21 - Datenmodell AdQuest

13.4.1. Tabellen

13.4.1.1. User

User	
ID	Eindeutige Identifizierung jedes App-Users
Fb_user_id	Die Facebook-ID des Users
Api_token	Der User registriert sich in der AdQuest-App über Facebook in unserem System und erhält von Facebook einen Token
IsBanned	Legt fest ob ein User blockiert ist

Tabelle 1 - User-Tabelle (Datenbank)

In der „User“-Tabelle werden alle Personen gespeichert, welche sich mittels der AdQuest-App in unserem System registrieren. Durch den „API_token“ können personenspezifische Informationen, wie Name oder Email, aus der Graph-API von Facebook gelesen werden.

13.4.1.2. User_has_Quest

User_has_Quest	
User_ID	ID des Users, welcher an der Quest teilnimmt
Quest_ID	ID der Quest, an welcher teilgenommen wird

Tabelle 2 - User_has_Quest-Tabelle (Datenbank)

In der „User_has_Quest“-Tabelle werden die User_ID und Quest_ID gespeichert, wenn ein User an einer Quest teilnimmt.

13.4.1.3. Quest

Quest	
ID	Eindeutige Identifikation einer Quest
Description	Beschreibung einer Quest
Location	Ungefährer Ort der Quest (z.B. Stadtname)
Endtime	Legt fest, wann eine Quest zu Ende ist

Imagepath	URL zu einem Bild, welches zu dieser Quest angezeigt wird
UserMinimum	Mindestanzahl von teilnehmenden Usern bevor die Quest startet
UserMaximum	Maximalanzahl von teilnehmenden Usern an dieser Quest
CreationTime	Datum, an dem die Quest erstellt wird
Company_ID	Unternehmen, welches die Quest anlegt
Status_ID	Eine Quest durchläuft mehrere Status
Link	Link, welcher im Facebook Post enthalten ist
CurrentUsers	Aktuelle Anzahl der teilnehmenden User
IsEvent	Gibt an, ob es sich um eine Quest oder ein Event handelt
Starttime	Startzeitpunkt
Price_ID	Verweis auf die Price-Tabelle für die Kostenberechnung
IsPaid	Gibt an ob die Quest bezahlt ist

Tabelle 3 - Quest-Tabelle (Datenbank)

In der „Quest“-Tabelle werden alle Quests gespeichert, welche Unternehmen anlegen können.

13.4.1.4. Status

Status	
ID	Identifiziert jeden Status eindeutig
Name	Name des Status
Description	Beschreibung des Status

Tabelle 4 - Status-Tabelle (Datenbank)

In der „Status“-Tabelle werden alle möglichen Status, welche eine Quest durchläuft, gespeichert (erstellt, gestartet, beendet…).

13.4.1.5. Price

Price	
ID	Identifiziert jeden Preis eindeutig
Date	Datum, an dem der Preis eingetragen wurde
ClickCost	Kosten, je erstelltem Facebook-Post
ViewCost	Kosten, je View
ServiceCharge	Bearbeitungsgebühren
DroneCost	Drohnenkosten

Tabelle 5 - Price-Tabelle (Datenbank)

In der „Price“-Tabelle werden die einzelnen Kosten für eine Quest gespeichert. Die Kosten für eine neue Quest werden immer vom aktuellsten Preis, mithilfe von „Date“, zugeordnet.

13.4.1.6. Company

Company	
ID	Identifiziert jedes Unternehmen eindeutig
Password	Passwort
Description	Beschreibung des Unternehmens
Name	Name des Unternehmens
Imagepath	URL zu einem Bild
Email	Email des Unternehmens
Address_ID	Adresse des Unternehmens
IsDeleted	Zeigt an, ob ein Unternehmen als gelöscht gilt
Salt	Wird zum Password-Hashing benötigt
LoginName	Loginname des Unternehmens

Tabelle 6 - Company-Tabelle (Datenbank)

In der „Company“-Tabelle werden die registrierten Unternehmen gespeichert. Da das Passwort nicht im Klartext in der Datenbank stehen sollte, wird dieses gemeinsam mit einem Salt verhasht.

13.4.1.7. Address

Address	
ID	Identifiziert jede Adresse eindeutig
AddressLine1	Adresse des Unternehmens
AddressLine2	Adresse des Unternehmens
AddressLine3	Adresse des Unternehmens
City	Stadt
Region	Region (z.B. Oberösterreich)
Postcode	Postleitzahl
Country	Land

Tabelle 7 - Address-Tabelle (Datenbank)

In der „Address“-Tabelle werden die Adressen für Unternehmen gespeichert.

13.4.1.8. AdCache

AdCache	
ID	Identifiziert jeden AdCache eindeutig
Longitude	Längengrad, an dem der AdCache versteckt ist
Latitude	Breitengrad, an dem der AdCache versteckt ist
OutsideRadius	Betritt ein User diesen Radius startet die Drohne automatisch
InnerRadius	Radius, welcher benötigt werden würde, falls die Drohne ein Manöver fliegen sollte
PlacedTime	Zeitpunkt, an dem der AdCache versteckt wird
Quest_ID	ID der Quest, zu dem der AdCache zugehörig ist
Drone_ID	ID der Drohne welche neben dem AdCache platziert wird
AdCache_Status_ID	Status des AdCaches

UUID	Universally Unique Identifier Zeichen-kette welche den AdCache eindeutig identifiziert (wird für QR-Code benötigt)
Finder	User-ID, welcher den AdCache findet
FoundDate	Datum, an dem der AdCache gefunden wurde

Tabelle 8 - AdCache-Tabelle (Datenbank)

In der „AdCache“-Tabelle wird jeder erstellte AdCache, gemeinsam mit seiner zugehörigen Quest, Drohne (wenn angegeben), seinem aktuellen Status sowie der ID des Users, der den AdCache gefunden hat, gespeichert.

13.4.1.9. Drone

Drone	
ID	Identifiziert eine Drohne eindeutig
Name	Name der Drohne
Token	Google-Firebase-Token, welche die Drohnensteuerungs-App erhält, um eine Push-Notification zu erhalten

Tabelle 9 - Drone-Tabelle (Datenbank)

In der „Drone“-Tabelle werden alle Drohnen gespeichert, die AdQuest zur Verfügung stehen. Den „Token“ erhält die AdQuest-Drohnen-App von Firebase, womit es möglich ist eine Push-Notification an ein bestimmtes Smartphone zu senden und die Drohne starten zu lassen, falls ein User den Radius betritt.

13.4.1.10. AdCache_Status

AdCache_Status	
ID	Identifiziert einen Status eindeutig
Name	Name des Status
Description	Beschreibung des Status

Tabelle 10 - AdCache_Status (Datenbank)

In der „AdCache_Status“-Tabelle werden alle Status gespeichert, die ein AdCache durchläuft (angelegt, platziert, Drohne …).

13.4.1.11. Admin

Admin	
ID	Identifiziert jeden Administrator
Password	Passwort des Administrators
Name	Name des Administrators
Email	Email des Administrators
Salt	Wird zum Passwort-Hashing benötigt.

Tabelle 11 - Admin-Tabelle (Datenbank)

In der „Admin“-Tabelle werden alle Administratoren des AdQuest-Systems gespeichert. Das Feld „Salt“ wird für das Hashing des Passwortes benötigt.

13.5. RESTful Webservice

Wie im Kapitel 13.1 Grundidee angeführt, wird ein RESTful Webservice oder auch REST API genannt, im AdQuest-System verwendet.

13.5.1. Einführung

Representational State Transfer, kurz REST, ist eine Architektur die dem http-Protokoll zugrunde liegt. (vgl. Tilkov, 2017)

13.5.2. Ressourcen

In der REST-Architektur geht es hauptsächlich um Ressourcen, welche User, Produkte, Posts etc. sein können. Eine Ressource muss folgende Anforderungen erfüllen:

- **Adressierbarkeit**
  Jede Ressource muss mittels eines Unique Resource Identifier (URI) eindeutig identifizierbar sein.

  Im AdQuest-System ist beispielweise jede Quest über ihre ID eindeutig identifizierbar und über die URI „http://adquest.htl-perg.ac.at/AdQuest/quest/{id}“ erreichbar.

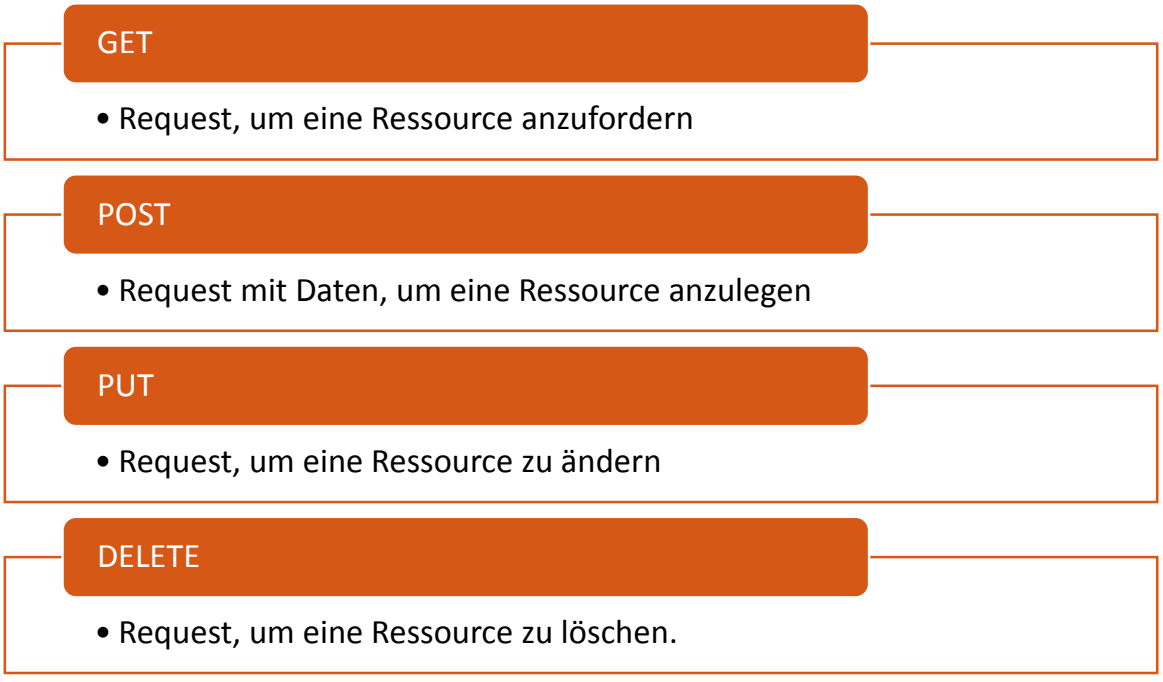
- **Zustandslosigkeit**
  Durch das http-Protokoll ist die Kommunikation zustandslos. Der Server speichert keine Informationen zu vergangenen Anfragen von einem Client, wodurch Sessions entfallen. Der Client muss bei jedem Request alle benötigten Informationen erneut mitsenden, damit der Server diesen bearbeiten kann.

  Im AdQuest-System wird selbst bei der Authentication von App-Usern und Kunden auf die Zustandslosigkeit geachtet.

- **Einheitliche Schnittstelle**
  Durch die Standard-http-Methoden (GET, POST, PUT, DELETE) kann auf die Ressourcen zugegriffen werden.
- **Entkopplung von Ressourcen und Repräsentation**
  Eine Ressource kann in verschiedenen Repräsentationen (z.B. JSON-Format oder XML-Format), je nach Client-Anforderung, ausgeliefert werden. (vgl. Helmich, 2017)

13.5.3. http-Methoden

Das http-Protokoll stellt verschiedene Methoden zur Verfügung. Mithilfe dieser Methoden kann auf die Ressourcen zugegriffen werden, welche sich auf dem Server befinden.



13.5.4. Ablauf eines Requests

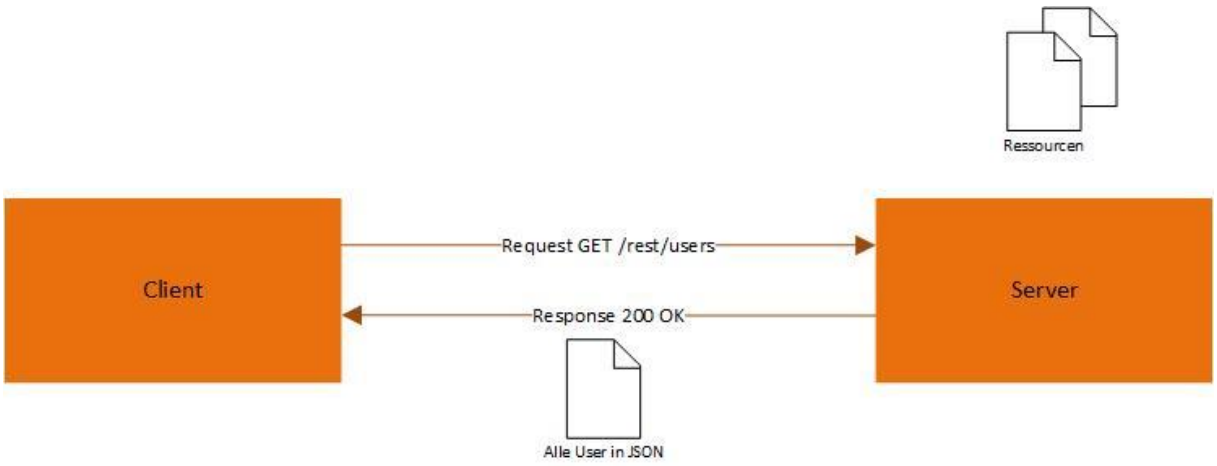


Abbildung 22 - Darstellung Request-Response

Ein Client (im AdQuest-System: Android-App oder Website) fordert alle User, welche im System gespeichert sind, über eine URL „/rest/users/“, an. Der Server liest diese aus der Datenbank und sendet sie als Response (Antwort) in JSON an den Client zurück. Der Statuscode 200 (OK) zeigt dem Client an, dass die Anfrage erfolgreich war.

13.6. JAVA REST API

13.7. Data Access Object

Ein „Data Access Object (DAO)“ ist ein Objekt welches den Zugriff auf eine Datenbank ermöglicht. Für jede Tabelle gibt es eine eigene Java-Klasse und ein eigenes Data Access Object, um Daten aus der Datenbank zu lesen oder zu schreiben. Das DAO besitzt einen EntityManger, welcher Methoden bereitstellt, um Objekte in der Datenbank zu suchen, zu speichern, zu ändern oder zu löschen.

13.8. Persistance.xml

Diese Datei enthält die Verbindung zur Datenbank sowie die Entitätsklassen. Das Mapping zwischen den Java-Objekten und der Datenbank übernimmt das orm.xml. Die JTA-Data-Source gibt die Verbindung zu der Datenbank an, welche im Wildfly angelegt ist.

```
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    <persistence-unit name="AdQuest">
        <jta-data-source>java:/AdQuestDS</jta-data-source>
        <mapping-file>META-INF/orm.xml</mapping-file>
    </persistence-unit>
</persistence>
```

Abbildung 23 - persistance.xml

13.9. ORM.xml

ORM (object-relational mapping) wird benötigt um Objekte der objektorientierten Programmierung in die Tabellen einer Datenbank zu speichern.
Die Konfiguration für das Mapping wird im orm.xml angegeben.

Beispiel-Mapping für Status-Tabelle:

```
<entity class="at.ac.htlperg.entities.Status">
    <table name="Status"></table>
    <named-query name="Status.findAll">
        <query>SELECT s FROM Status s</query>
    </named-query>
    <attributes>
        <id name="id"><column name="ID"/></id>
        <basic name="description"><column name="Description"/></basic>
        <basic name="name"><column name="Name"/></basic>
        <one-to-many name="quests" mapped-by="status"></one-to-many>
    </attributes>
</entity>
```

Abbildung 24 - orm.xml

entity	Jede Entität besitzt einen eigenen entity-Tag
class	Java-Klasse
table	Datenbanktabelle für die Klasse
named-query	Query zur Abfrage der Tabelle
attributes	Attribute der Klasse, welche in der Tabelle gespeichert werden
id	ID der Entität
basic	Spalte in der Tabelle
one-to-many	1:m-Beziehung

Tabelle 12 - ORM-Erklärung

13.10. Entität

Eine Entität ist die Abbildung einer Tabelle in einer Java-Klasse, welche die Tabellenspalten als Attribute besitzt. Für jede Datenbanktabelle in AdQuest gibt es hierfür eine eigene Klasse.

13.11. Service

Ein Service, auch Endpoint genannt, stellt dem Client eine Schnittstelle zur Verfügung, die von diesem über eine URL abgerufen werden kann.

Beispiel QuestService:

Dieser Service ermöglicht den Zugriff auf bestehende Quests, sowie das Erstellen oder Ändern von Quests.

```
@Path("/quest")
public class QuestService {
```

Abbildung 25 - Path-Annotation

Die „@Path(„/quest“)“ Annotation gibt an unter welcher URL dieser Service erreichbar ist. In diesem Fall unter:
http://adquest.htl-perg.ac.at/AdQuest/quest

```
@Inject
private QuestDao questDao;
```

Abbildung 26 - DAO

Die „@Inject“-Annotation ist eine Dependency Injection. Das bedeutet, dass die Abhängigkeit des Objektes erst zur Laufzeit fixiert wird. (Wikipedia, 2017)

```
@Context
private SecurityContext securityContext;
```

Abbildung 27 - Security Context

Durch das SecurityContext-Objekt können Informationen zum Client abgerufen werden. Mithilfe dieser Informationen kann überprüft werden, ob dieser die Berchtigungen besitzt auf diese Methode bzw. Ressource zuzugreifen.

```
@GET
@Produces(MediaType.APPLICATION_JSON)
public Response getAll(@QueryParam("userid") int userid){
```

Abbildung 28 - Java-Methode (GET)

Die „@GET“-Annotation gibt an, dass diese Methode durch einen http-Get Request erreichbar ist.

Die „@Produces“-Annotation gibt das Format wider, in dem die Ressource zurückgesendet wird. In diesem Fall wird die Entität (hier: Quest) in JSON zurückgesendet.

Die „@QueryParam“-Annotation gibt an welche Parameter in der URL übergeben werden können. Diese Parameter werden in der URL am Ende nach einem „?“ angegeben und mittels „&&“ getrennt.
Beispiel:
http://adquest.htl-perg.ac.at/AdQuest/quest?userid=1

```
@POST
@Secured
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response createQuest(Quest quest,@QueryParam("amount")int amount){
```

Abbildung 29 - Java-Methode (POST)

Die „@POST“-Annotation gibt an, dass diese Methode durch einen http-POST Request erreichbar ist.

Die „@Secured“-Annotation gibt an, dass diese Methode geschützt ist und sich der Client mit einem JWT authentifizieren muss. In dieser Methode muss zum Beispiel zusätzlich geprüft werden, ob das Unternehmen, welches in der Quest angegeben wird, wirklich das Unternehmen ist, welches gerade auf diese Methode zugreift. Hier kommt der „SecurityContext“ zum Einsatz.

Die @Consumes-Annotation gibt an in welchem Format die Ressource, welche gespeichert werden soll, erwartet wird (in diesem Fall als JSON). Die anzulegende Quest im JSON-Format wird direkt in ein Quest-Objekt „umgewandelt“.

Für jede Tabelle in der Datenbank stellt die AdQuest-API für den Client ein Service zur Verfügung, mit welchem Daten gespeichert oder bearbeitet werden können.

13.12. Authentication

Da, wie bereits 13.5 RESTful Webservice erwähnt, RESTful-APIs zustandslos sind, muss die Zustandslosigkeit auch beim Login beibehalten werden. AdQuest verwendet für die Authentifizierung von Usern der App und Unternehmen auf der Website JSON Web Tokens. JWTs enthalten alle benötigten Informationen zur Authentifikation und werden bei jedem Request mitgesendet.

13.12.1. Aufbau

JSON Web Token bestehen aus 3 Teile, welche durch Punkten getrennt werden:

- Header
- Payload
- Signature

Der Aufbau sieht folgendermaßen aus:
HEADER.PAYLOAD.SIGNATURE

13.12.1.1. Header

Im Header ist die Art des Tokens und der Signatur bzw. Verschlüsselungsalgorithmus enthalten. Diese Informationen werden in JSON-Form abgelegt und in Base64 kodiert.

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWS",
  "alg": "HS256"
}
```

Abbildung 30 - JWT-Header

Die Art des Web Tokens ist „Json Web Signature“, da der JWT vom Server signiert wird. Des Weiteren wird als Signaturalgorithmus HMAC SHA256 (HS256) verwendet.

Base64 kodiert ergibt dies:

```
eyJ0eXAiOiJKV1MiLCJhbGciOiJIUzI1NiJ9.
```

Abbildung 31 - Base64 kodierter JWT-Header

13.12.1.2. Payload (Claims)

Der Payload enthält die Daten (Key/Value-Paare), welche der Server für die Authentifizierung benötigt. Diese Daten werden „Claims“ (Key) genannt und geben Auskunft über eine Entität (im Falle von AdQuest über einen App-User oder Unternehmer).

PAYLOAD: DATA


```
{
  "jti": "2",
  "iat": 1490527685,
  "sub": "musterunternehmen",
  "iss": "adquest.htl-perg.ac.at",
  "Role": "Company",
  "exp": 1490531285
}
```


Abbildung 32 - JWT-Payload

Key	Beschreibung
jti	JWT ID Im Falle von AdQuest die ID des Unternehmens oder Users
iat	Issued-at time Ausstellungszeitpunkt des JWTs
sub	Subject Identifiziert den Nutzer (Username)
iss	Issuer Identifiziert den Aussteller des JWT
Role	Rolle Zeigt die Rolle des Clients an (Company, User, Admin)
exp	Expiration Time Ab diesem Zeitpunkt gilt der JWT als abgelaufen bzw. ungültig.

(vgl. Hoogvliet, 2017)

Base64 kodiert ergibt dies:

```
eyJqdGkiOiIyIiwiaWF0IjoxNDkwNTI3Njg1LCJzd
WIiOiJtdXN0ZXJ1bnRlcm5laG1lbiIsImlzcyI6Im
FkcXVlc3QuaHRsLXBlcmcuYWMuYXQiLCJSb2xlIjo
iQ29tcGFueSIsImV4cCI6MTQ5MDUzMTI4NX0.
```

Abbildung 33 - Base64 kodierter Payload

13.12.1.3. Signature

Für die Signatur wird der kodierte Header, der kodierte Payload, ein „Secret“ und der im Header definierte Algorithmus verwendet. Die Signatur verifiziert den Sender und stellt sicher, dass die Nachricht nicht verändert wurde.

Base64 kodiert ergibt dies:

```
f01FwZcI6RM8nx6BhXOxwuRAoZCceGt9McJ6T3fmG
tI
```

Abbildung 34 - Base64 kodierte Signatur

Ein vollständiger JWT im AdQuest System sieht nun folgendermaßen aus:

```
eyJ0eXAiOiJKV1MiLCJhbGciOiJIUzI1NiJ9.eyJq
dGkiOiIyIiwiaWF0IjoxNDkwNTI3Njg1LCJzdWIiO
iJtdXN0ZXJ1bnRlcm5laG1lbiIsImlzcyI6ImFkcX
Vlc3QuaHRsLXBlcmcuYWMuYXQiLCJSb2xlIjoiQ29
tcGFueSIsImV4cCI6MTQ5MDUzMTI4NX0.f01FwZcI
6RM8nx6BhXOxwuRAoZCceGt9McJ6T3fmGtI
```

Abbildung 35 - JWT im AdQuest System

13.12.2. Ablauf

Der User bzw. das Unternehmen meldet sich mit seinen Benutzerdaten an. Sind diese korrekt sendet der Server einen JSON Web Token an den Client zurück, welcher nun diesen JWT bei jedem Request im Authorization-Header mitsenden muss.

```
Authorization: Bearer <token>
```

Abbildung 36 - HTTP-Authorization-Header

Durch diese Zustandslosigkeit speichert der Server nie die Informationen, ob ein User angemeldet ist. Ist eine URL geschützt, validiert der Server den JWT, indem er die Signatur prüft, und den Zugriff erlaubt oder verweigert. Des Weiteren werden die Informationen zum Client aus dem JWT gelesen.
Dadurch, dass der JWT alle benötigten Informationen enthält reduziert sich somit auch der Datenbankzugriff. (JWT, 2017)

Besitzt eine Methode die @Secured-Annotation prüft ein „Filter“ den JWT und speichert die enthaltenen Informationen für den „SecurityContext“. Dieser kann nun überprüfen, ob sich ein Client als falscher Benutzer ausgibt.

13.13. Passwort

Passwörter werden nicht im Klartext in der Datenbank abgelegt, sondern durch das Anwenden einer Hashfunktion auf das Passwort entsteht ein Hashwert, welcher in der Datenbank gespeichert wird. Durch diesen Mechanismus kann ein Angreifer nicht mehr die Passwörter in Klartext einsehen. Durch verschiedene Angriffe ist es aber dennoch möglich aus dem Hashwert das Passwort wieder zu bekommen (Wörterbuchangriffe, Rainbow Tables …). Um diese Methoden zu verhindern bzw. zu erschweren wird ein zusätzlicher Salt verwendet. Dieser zufällig erzeugte Salt wird an das Passwort „angehängt“ und erst dann wird der Hashwert berechnet. Der Salt wird ebenfalls in der Datenbank abgelegt. Um Angriffe noch zu erschweren wird der Algorithmus mehrmals auf das Ergebnis angewandt. (vgl. Helmbold, 2017)
Das AdQuest-System verwendet hierbei PBKDF2 (Password-Based Key Derivation Function 2) mit SHA-1 um diesen Schutz zu gewährleisten.

Beim Login wird gemeinsam mit dem Salt aus der Datenbank und mit dem eingegebenen Passwort wieder ein Hashwert erzeugt und mit dem Hashwert in der Datenbank verglichen. Stimmen die beiden Werte überein ist der User/Unternehmer erfolgreich eingeloggt.

13.14. QR-Code-System

Für jeden AdCache wird beim Erstellen eine zufällige UUID erzeugt, um diesen eindeutig zu identifizieren. Gemeinsam mit einer URL, welche auf eine Methode in der REST-API verweist, wird ein QR-Code generiert, welcher beim AdCache platziert wird. Findet ein App-User den AdCache, scannt er diesen und führt einen Request aus. Auf der Serverseite wird nun die übertragene UUID und die, in der Datenbank gespeicherte UUID verglichen. Stimmen diese überein wird der App-User als Finder des AdCaches gespeichert und das Unternehmen, welches die Quest erstellt hat, per E-Mail, welche die Kontaktdaten des Users beinhaltet (Name, Email), informiert, um sich mit diesem für den Austausch des Geschenks in Verbindung zu setzen. Der AdCache gilt nun als gefunden.

13.15. Paypal

Unternehmer können auf der Website ihre Quests per Paypal bezahlen. Damit das AdQuest-System über diese Zahlung informiert wird, wird beim Bezahlen von Paypal automatisch eine Instant Payment Notificatoin (IPN) an das Webservice von AdQuest gesendet, welche die Informationen über die Transaktion beinhaltet. Das Webservice sendet nun automatisch einen Request an Paypal mit denselben Daten und einem zusätzlich benötigten Parameter ab, um diese zuvor erhaltene Nachricht zu überprüfen. Als Antwort wird nun „VERIFIED“ gesendet, falls die Benachrichtigung echt war bzw. „INVALIED“, falls diese nicht echt war.
Bei „VERIFIED“ fand tatsächlich eine Transaktion statt und die Quest gilt als bezahlt.

13.16. Graph-API

Mittels der Graph API ist es möglich Daten aus der Facebook-Plattform abzurufen. Da sich die App-User über Facebook bei AdQuest registrieren, erhalten sie einen Access Token, welcher in der Datenbank gespeichert wird. Über diesen Token und der Facebook-ID des Users ist es nun möglich die Informationen zu einem Benutzer von Facebook abzurufen. AdQuest erhält durch die vergebenen Berechtigungen in der App Zugriff auf den Vor- und Nachnamen sowie die E-Mail-Adresse.

13.17. Firebase

Um die beim AdCache platzierte Drohne zu starten, falls sich ein App-User in der Nähe befindet, wird eine Push-Notification von der Plattform Firebase an die Drohnen-App gesendet.

In Firebase wurde ein AdQuest-Project angelegt und die Drohnen-App hinzugefügt. Diese App erhält einen Token, mit welchem nur dieses Gerät benachrichtigt werden kann. Befindet sich nun ein App-User bei einem AdCache sendet der AdQuest-Server einen Request, welcher den Token der App enthält, an Firebase. Dadurch kann eine Push-Notification an das Gerät gesendet und die Drohne automatisch gestartet werden.

14. Website

14.1. Grundidee

Die AdQuest-Website stellt die im Geschäftsmodell beschriebene Unternehmenskunden-Seite des Konzeptes dar. Sie ist primär nur für die Interaktion von Werbekunden mit dem AdQuest-System gedacht.
Unternehmer haben auf der Website, die Möglichkeit ihr Unternehmen bzw. ihre Firma zu registrieren und ihre gekauften Produkte zu verwalten. Dazu dient eine Verwaltungstabelle, in der alle gekauften Quests und Events mit ihrem jeweiligen Status aufgelistet sind. Diese Tabelle dient außerdem als „Startseite“ sobald sich der Benutzer eingeloggt hat.
Um diese Tabelle zu befüllen, muss der Benutzer aber zuerst Produkte kaufen. Dies kann er über unsere Kaufübersicht tun.
Zuerst bekommt der Unternehmer eine Auflistung mit unseren verschiedenen Quests und Events die derzeit angeboten werden. Entscheidet sich der Kunde nun für ein Angebot, kann er in einer Formularansicht noch alle weiteren nötigen Daten für seine Quest/Event eingeben.
Bezahlt wird auf zwei verschiedene Arten, abhängig von der Art des ausgewählten Produktes. Zum einen bietet AdQuest die Möglichkeit der Zahlung per Rechnung, zum anderen hat man durch die Anbindung zu PayPal, die Möglichkeit seine Zahlung auch online durchzuführen, allerdings nur bei einigen ausgewählten Produkten.
Will sich der Kunde die QR-Codes, zum Verstecken seiner Produkte ausdrucken, tut er dies über die Cache-Übersicht zu den Quests. Hier werden alle Caches mit dem dazugehörigen Status und der Position angezeigt und man hat die Möglichkeit sich generierte QR-Codes der Caches auszudrucken.

14.2. Technik - Angular 2

siehe Kapitel 10.4.1 Angular2

14.2.1. Komponenten

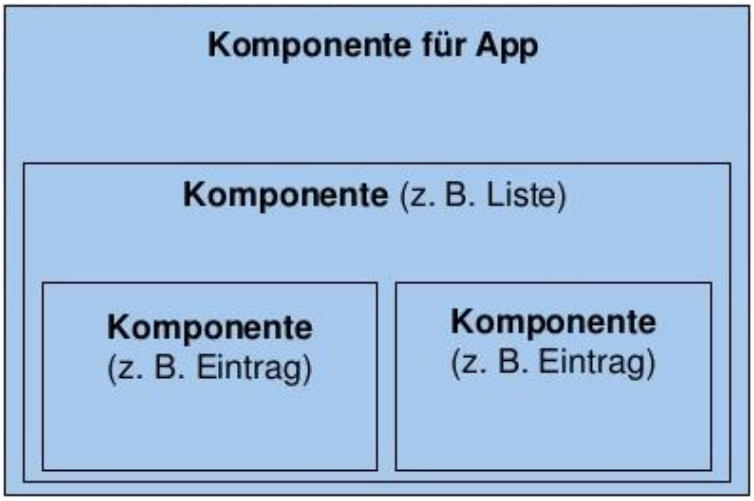


Abbildung 37 – KomponentenBaum (Slideshare, 2017)

Angular baut auf der Zusammensetzung von einzelnen Komponenten auf, die zusammen die Web-App darstellen. Wie in der obigen Abbildung zu sehen ist, sind diese Komponenten ineinander verschachtelt und dienen jeweils dazu, einen gewissen Teil der GUI anzuzeigen. Jeder Component ist dabei für einen anderen Teil der GUI zuständig und arbeitet mit seinen eigenen Daten, d.h. ein Component ist auch nur von seinen eigenen Daten abhängig. Ein Component dient in seiner Funktion aber nicht direkt als View. Vielmehr hat jeder Component eine für ihn zugehörige View, welche aus ganz normalen HTML-Elementen besteht, mit dem Unterschied das direkt auf Daten des Components zugegriffen werden kann. Münzt man dieses Konzept auf die Konvention der MVC (Model – View – Controller)-Architektur um, dient der Component gewissermaßen als Controller und Model und verweist auf die dazugehörige View.

14.2.2. Komponentenbeschreibung

Im folgenden Kapitel erfolgt eine grobe Beschreibung der Verantwortlichkeiten der wichtigsten Components der AdQuest Website.

app.component **/ app.html**	• äußerster Component • zuständig für Navigation-Bar • beinhalten den <router-outlet>-Tag
register.component **/ register.html**	• Registrierungsformular
login.component **/ login.html**	• Login-Formular
profile.component **/ profile.html**	• Benutzereinstellungen
quests.component **/ quests.html**	• Quest-Tabelle
caches.component **/ caches.html**	• Cache-Tabelle zu einer Quest
add.component **/ add.html**	• hält den <router-outlet>-Tag für das nested-routing ins Zahlungssystem
select.component **/ select.html**	• nur View → Informationen zum Zahlungssystem
packages.component **/ packages.html**	• Packageauswahl und Angebotsanzeige zu den Quest-Packages
custom.component **/ custom.html**	• Formular für Custom-Quests bzw. Events
checkout.component **/ checkout.html**	• Formular für die Packages; PayPal Zahlung
contact.component **/ contact.html**	• Senden von Mail an die AdQuest Email-Adresse
event.component **/ event.html**	• Formular für Eventerstellung

14.2.3. Routing

Routing ist ein zentrales Feature von Angular2. Es erlaubt das dynamische Laden von verschiedenen Komponenten der Website in den sogenannten <router-outlet>-Tag der sich meist im App-Component bzw. im App-html befindet. Der <router-outlet>-Tag definiert hier die Stelle in die der Website-Content geladen wird, abhängig davon, welche URL vom Browser aufgerufen wird.
Dafür gibt es spezielle Regeln die im App-Module, bzw. in den Modules der jeweiligen Website-Bereiche angegeben werden müssen. Diese Regeln spezifizieren den jeweiligen Component, bzw. in Folge dessen welches HTML-File beim Aufruf verschiedener URLs geladen wird.
In der nachstehend Abbildung sind diese Regeln von AdQuest dargestellt.

```
RouterModule.forRoot([
    { path: 'profile', component: LoginComponent },
    { path: 'profile/login', component: LoginComponent },
    { path: 'profile/register', component: RegisterComponent },
    { path: 'profile/settings', component: ProfileComponent, canActivate: [ AuthGuard ] },
    { path: 'profile/reader', component: ReaderComponent }
]),
```

Abbildung 38 – RouterModule_Profile

In dieser Abbildung ist die Router-Konfiguration für das *RouterModule* des Profil-Systems abgebildet. Die dabei definierten Pfade werden immer an den sogenannten „base“-Pfad angehängt, sprich im Falle der AdQuest Wesite
an **„adquest.htl-perg.ac.at/“**
Angegeben wird immer der Pfad, welcher aufgerufen wird, mit dem dazugehörigen Component der geladen werden soll.
Das Aufrufen eines solchen Pfades kann dabei auf verschiedene Art erfolgen. Einerseits kann direkt in die Adresszeile des Browsers eine Adresse geschrieben werden. Es ist aber auch möglich das Aufrufen der Adresse auf bestimmte Aktionen bzw. Events der Website zu legen, wie etwa einen Button-Klick.

```
gotoSettings()
{
  this.router.navigateByUrl("/profile/settings");
}
```

Abbildung 39 – Routing-Aufruf

Hier sieht man das *router.navigate*-Statement, welches auch manuell aufgerufen werden kann. Diese Methode etwa wird bei einem Klick auf ein Feld im Side-Nav durchlaufen.

14.2.4. Binding

Grundsätzlich kann zwischen zwei verschiedenen Arten des Bindings unterschieden werden, dem Property-Binding, also dem Binding von Variablen zur Änderung in der View, und dem Event-Binding, also dem Binden von Events in der View an gewisse Methoden im Component.

14.2.4.1. Property-Binding

Beim Property-Binding handelt es sich um ein sogenanntes „One-Way“-Binding. Die Bindung besteht nur in eine Richtung: vom Controller zur View. Ändern sich im Controller Daten, geschieht dies durch das Property-Binding auch in der View.
Im Code ist das Property-Binding durch die sogenannte Injection verwirklicht.

```
<td>{{quest?.description}}</td>
<td>{{quest?.location}}</td>
<td>{{quest?.currentUsers}}/{{quest?.userMaximum}}</td>
```

Abbildung 40 - Injection

Durch die Injection ist es möglich, direkt in der View, Daten des Component anzuzeigen.

14.2.4.2. Event-Binding

Beim Event-Binding erfolgt die Bindung von der View zum Component. Einem bestimmten Event in der View wird also eine Methode zugewiesen. Im folgenden Beispiel wird einem Klick eine Methode zugewiesen.

```
<a (click)="gotoAdd()">
    <i class="material-icons" md-tooltip="Legen Sie einen neuen AdQuest an.">note_add</i>
</a>
```

Abbildung 41 – Event-Binding

14.2.4.3. Two-Way-Binding

Die Bindung in beide Richtungen ist in Angular, bzw. Typescript nicht wirklich realisiert. Vielmehr wird Two-Way-Binding als Kombination von Event- und Property-Binding eingesetzt.

```
<input id="settings-name" type="text" [(ngModel)]="company.name" name="nameC" class="validate" required>
```

Abbildung 42 – Two-Way-Binding

Vor allem bei Textfeldern werden bei AdQuest die sogenannten „*ngModel*“ eingesetzt. Diese binden das Textfeld an den Component und zeigen Veränderungen im Component an, bzw. ändern auch bei Eingabe in der View, Daten des Component.

14.3. Umsetzung

14.3.1. MainPage

Abbildung 44 - MainPage

Im oberen Screenshot wird die MainPage der AdQuest Website gezeigt. Was auf dem Screenshot nicht ersichtlich ist sind die Animationen beim Hereinladen der Seite. Der grüne Graph am unteren Bildschirmrand baut sich von Links nach Rechts auf, während im Hintergrund langsam die Schrift und die Weltkarte eingeblendet werden.

```
@keyframes onloadwidth {
    0%   {width: 0;}
    50% { width: 100%;}
}
@keyframes onloadfade {
     0%   {opacity: 0;}
     33%   {opacity: 0;}
     50% { opacity: 1;}
 }
@keyframes onloadfadecontent {
    0%   {opacity: 0;}
    50%   {opacity: 0;}
    100% { opacity: 1;}
}
```

Abbildung 43 - Animations

Hier sieht man die fade-Funktionen die auf der MainPage verwendet werden. Das Attribut „opacity“ gibt dabei den Grad der Transparenz an.

14.3.2. Side-Nav

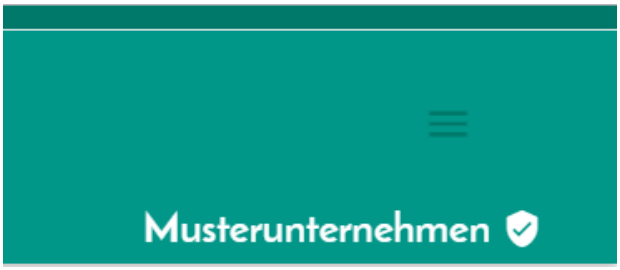


Abbildung 45 – Side-Nav

Zur Navigation des Benutzers durch die Website verwendet AdQuest einen sogenannten Side-Nav. Er ist eine vom Material-Design vorgefertigte GUI-Komponente und kann nach allen Vorstellungen angepasst werden.
Aufgerufen wird der Side-Nav durch einen Klick auf das in der oberen Abbildung gezeigte Symbol über dem Login-Namen.

Da der Side-Nav GUI-technisch betrachtet immer zur Verfügung stehen muss, egal welche inneren Komponenten gerade geladen sind, befindet er sich im äußersten Component der Anwendung, dem App-Component.
Der untere Abschnitt des Side-Navs ist dem Benutzer nicht immer sichtbar. Nur eingeloggte Benutzer haben die Möglichkeit einen Logout zu vollziehen, oder sich ihre Quests anzusehen.

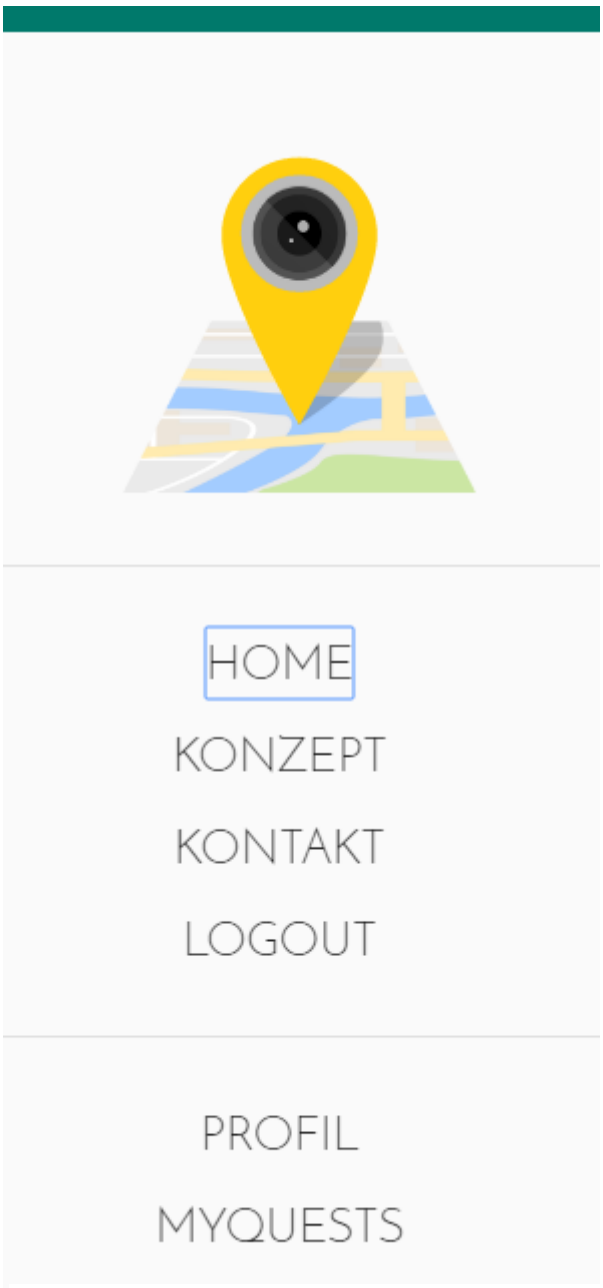


Abbildung 46 - SideNavOut

14.3.3. User-Card

Wie im vorangegangenen Abschnitt bereits beschrieben, stehen bestimmte Aktionen nur eingeloggten Benutzern zur Verfügung. Des Weiteren, wird der Unternehmensname des eingeloggten Benutzers in einer User-Card (siehe Abbildung 45) angezeigt.
Problem bei der Umsetzung dieses Features, war die Komponentenhierarchie. Die User-Card sowie der Side-Nav befinden sich beide im App-Component der Applikation. Das Einloggen und das Verändern der einzelnen GUI-Komponenten erfolgt dabei allerdings innerhalb des App-Component. Es tritt also das Problem auf, Daten in außenliegenden Components zu verändern, wenn ein Event in einem inneren Component auftritt und dabei auch beim Reload einer Seite die richtigen Daten anzuzeigen.
Realisiert wurde dies in AdQuest mit dem „Company-Service“.

```
private companyNameSource = new Subject<string>();
companyLoggedIn$ = this.companyNameSource.asObservable();

logInCompany(company: string) {
    this.companyNameSource.next(company);
}
```

Abbildung 47 - CompanyService

Dieses Service hält den Namen der Company als sogenanntes „*Subject*“, und ein „*Oberservable*“ auf dieses *Subject*. Bei erfolgreichem Login eines Benutzers, wird anschließend die Methode „*logInCompany*“ des Company-Service ausgeführt, welche den Namen des eingeloggten Unternehmens in das *Subject* speichert.
Zusätzlich wird durch die Änderung im *Subject* die *callback*-Methode des Observables aufgerufen, welches den Unternehmensnamen vom *Subject* in den App-Component übergibt.
Der App-Component prüft nun noch einmal, ob tatsächlich ein Benutzer eingeloggt ist. Ist diese Prüfung erfolgreich, kann nun der Name der User-Card auf den vorher übergebenen Unternehmensnamen gesetzt werden und die entsprechenden Sektionen im Side-Nav können freigegeben werden.

14.3.4. Login / Registrierung

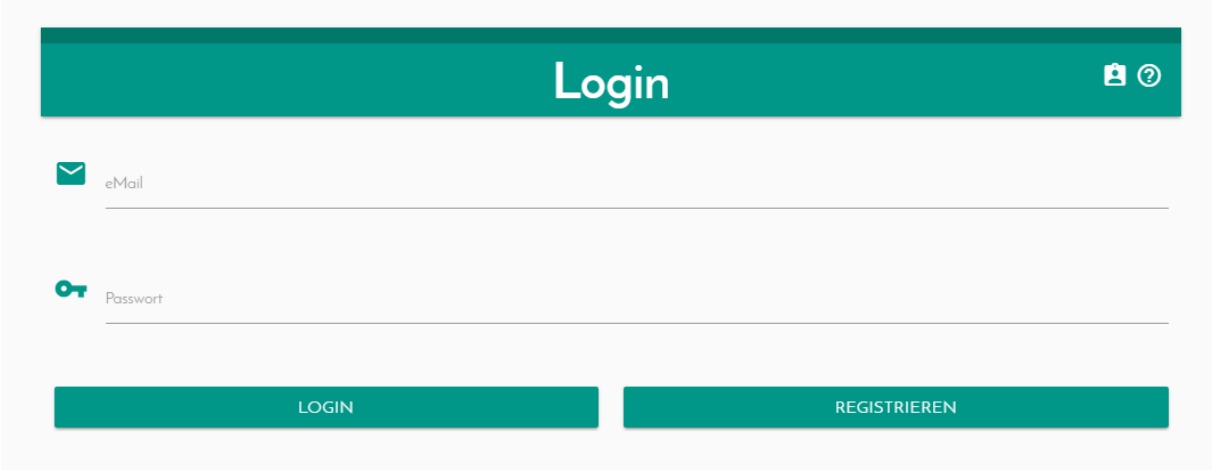


Abbildung 48 – Login-Screen

Im Hintergrund stehen der Login-Component bzw. der Register-Component. Beide nutzen Methoden des „Profile-Service“ welches zuständig für die Breitstellung aller REST-Methoden ist, die mit dem Profil-Systems in Zusammenhang stehen.

```
login(loginString, password) {
    let headers = new Headers({ 'Content-Type': 'application/x-www-form-urlencoded'});
    let options = new RequestOptions({ headers: headers });
    return this.http.post("http://adquest.htl-perg.ac.at/AdQuest/companyauthentication/login",
      "email=" + loginString + "&password=" + password,options);

}

register(company) {
    let headers = new Headers({ 'Content-Type': 'application/json' });
    let options = new RequestOptions({ headers: headers });
    return this.http.post("http://adquest.htl-perg.ac.at/AdQuest/companyauthentication/register",
      JSON.stringify(company),options).map((resp) => {
        console.log(resp.status);
        if(resp.status=200)
        {
            this.router.navigateByUrl("/profile/login");
        }
    });
}
```

Abbildung 49 – Methoden_ProfileService

Oben abgebildet sind die wichtigsten Methoden im Profile-Service.
Die Formulardaten werden mittels Binding im Hintergrund im Component gehalten.

Ist das Unternehmen erfolgreich eingeloggt, wird dem Benutzer mittels einer Snackbar die Meldung erteilt, dass sein Login erfolgreich durchgeführt wurde. Im Anschluss werden beim *subscriben* des Login-Requests einige Daten in den *„local-storage“* des Browsers gespeichert. Dies sind zum Beispiel die ID des Unternehmens und die E-Mail Adresse, aber auch ein vom Server generierter Token, welcher im Folgekapitel „Autorisierung“ eine Rolle spielt.

14.3.5. Autorisierung

AdQuest verwendet grundsätzlich die Verschlüsselung „**x-www-form-urlencoded**“. Logged sich ein Benutzer auf der AdQuest Website ein, schickt der Server in der Login-Response einige Daten zur Autorisierung mit.

14.3.5.1. JWT-Token

In der Response findet sich unter anderem ein vom Server generierter JWT-Token. Dieser wird bei Erhalt in den „*local-storage*“ des Browsers gespeichert.

```
this.profileService.login(this.loginString, this.password)
    .subscribe(
        resp => {
            localStorage.setItem('auth_token', resp.headers.get('Jwt'));
```

Abbildung 50 – JWT-Token

Um nun zu gewährleisten, dass ein Benutzer den REST-Dienst des Servers überhaupt nutzen darf, verlangt der Server bei jedem REST-Call das Mitsenden dieses Tokens.

```
let headers = new Headers({ 'Content-Type': 'application/json',
  'Authorization': 'Bearer ' + localStorage.getItem('auth_token') });
let options = new RequestOptions({ headers: headers });
```

Abbildung 51 - Bearer

Der in der obigen Abbildung dargestellte Code-Ausschnitt zeigt das Mitsenden des Tokens im Request-Header. Dazu ist es nötig dem Feld *„Authorization“* den Wert „*Bearer*“ zu zuweisen und anschließend den Token aus dem *local-storage* mit zu übergeben.
Wird kein Token mitgesendet oder ist der Token nicht mehr gültig wird dem Benutzer eine Notification angezeigt, das er nicht mehr eingeloggt ist. Er wird anschließend auf die Login-Seite zurückgeworfen und muss sich erneut anmelden um wieder einen gültigen Token vom Server zu bekommen

14.3.5.2. AuthGuard

Zusätzlich zum JWT-Token ist in AdQuest jede Seite von einem *AuthGuard* geschützt. Dieser *AuthGuard* ist dafür zuständig, nur befugte Benutzer auf die angeforderte Seite weiter zu leiten.

```
RouterModule.forRoot([
{ path: 'quests', component: QuestsComponent, canActivate: [ AuthGuard ] },
{ path: 'quests/caches/:id', component: CachesComponent, canActivate: [ AuthGuard ] }
```

Abbildung 52 - AuthGuard

Wie die obere Abbildung zeigt implementiert der AuthGuard die Methode „*canActivate*“. Diese Methode hat als Rückgabewert die Bestätigung ob der Benutzer, der die Seite anfordert, auch autorisiert ist, sprich eingeloggt ist. Die Seite darf dabei nur aufgerufen werden, wenn die Anfrage vom AuthGuard bestätigt wird.

14.3.6. Quests-Tabelle

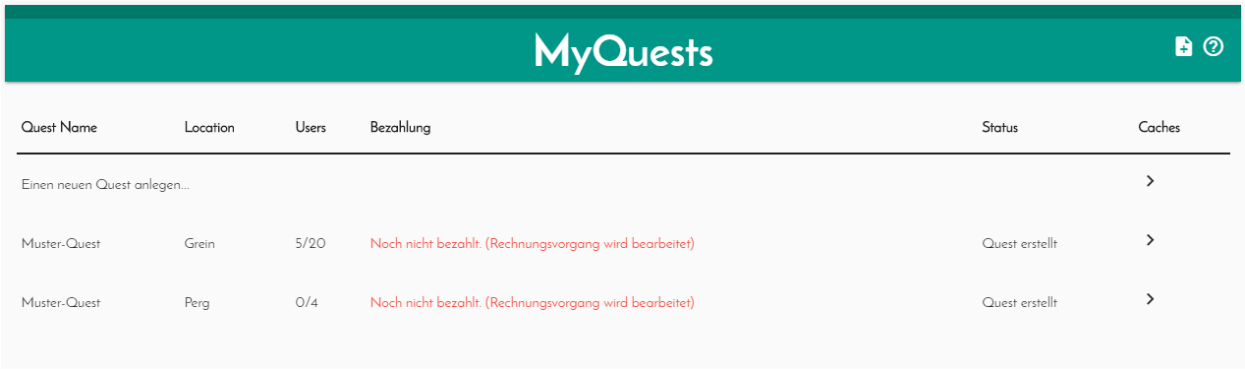


Abbildung 53 – Quest_Tabelle

Im Hintergrund der Quests-Tabelle steht der Quest-Component, welcher zuständig für den Aufbau und das Laden der Quest ist. Der Quest-Component bedient sich dabei der REST-Methoden des Quest-Services.
Um dabei das anfordernde Unternehmen genau zu identifizieren wird beim Request die ID des aktuell eingeloggten Unternehmens mitgesendet.

Die wichtigste Methode des Quest-Service ist dabei das Laden der Quests.

```
getQuests() {
    let headers = new Headers({ 'Content-Type': 'application/json',
      'authorization': 'Bearer ' + localStorage.getItem('auth_token')});
    let options = new RequestOptions({ headers: headers });
    return this.http.get("http://adquest.htl-perg.ac.at/AdQuest/company/"
      +localStorage.getItem('logged_in_id')+"/quests",options);
}
```

Abbildung 54 - GetQuests

Die in der Response enthaltenden Daten, werden anschließend nach erfolgreicher Durchführung des Requests in den Quests-Component gespeichert, wo sie über Property-Binding in der View, also der Quests-Tabelle angezeigt werden.

14.3.7. Cache-Tabelle

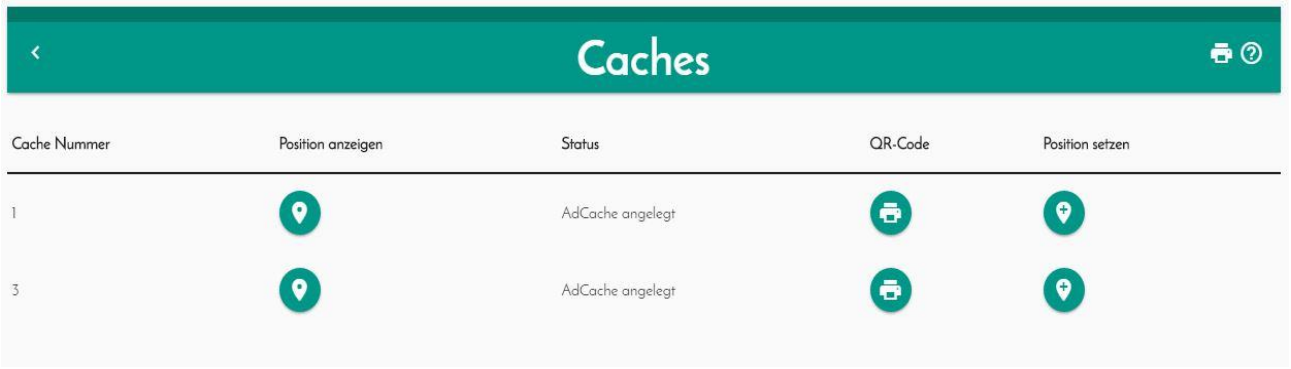


Abbildung 55 - Caches

Der Caches-Component verwaltet im Hintergrund alle Daten für die Cache-Tabelle.
Der Aufruf der Cache-Tabelle erfolgt bei Klick auf eine Quest in der Quests-Tabelle.

```
gotoCaches(id: number)
{
    this.router.navigate(['/quests/caches', id])
}
```

Abbildung 56 -CacheId

Die obige Abbildung zeigt die „*gotoCaches*“-Methode im Quests-Component, welche bei Klick auf einen Quest aufgerufen wird.
Die ID des aufgerufenen Caches wird dabei im Router als Parameter mit übergeben.

```
{ path: 'quests/caches/:id', component: CachesComponent, canActivate: [ AuthGuard ] }
```

Abbildung 58 - CacheIdPath

In der Abbildung wird die Konfiguration des Pfades zur Cache-Ansicht gezeigt. Der Aufruf erwartet den Parameter „*id*“ um den anzuzeigenden Cache genau identifizieren zu können.

```
ngOnInit() {
    this.route.params.subscribe(params => {
        this.id = +params['id'];
        this.fetchCaches();
    });
}
```

Abbildung 57 - RouteParams

Im oben gezeigten Code-Ausschnitt des Caches-Component sieht man die Extraktion des „*id*“-Parameters. Man verwendet dazu eine sogenannte „*ActivatedRoute*“, aus der man Parameter, welche man über den Router mitsendet, mit ihrer entsprechenden Identifikation holen kann. Eine *ActivatedRoute* ist dabei ein Objekt, welches eine Property „*params*“ enthält. *Params* ist hier ein Observable, das wie alle Observable bei Änderung der Daten über *subscribe* ausgelesen werden kann. Im oberen Beispiel wird so die *ID* über den Identifikator ‚id‘ ausgelesen.
Der Caches-Component implementiert zusätzlich noch die Funktion „ngOnInit()“, welche aufgerufen wird, sobald der Component bei Aufruf des Pfades initialisiert wird. Dadurch ist es möglich, noch bevor die View aufgebaut wird, den Request für die Caches abzusetzen.

```
private fetchCaches() {
    this.questsService.getCaches(this.id) .subscribe(
        resp => {
            this.caches = resp.json()
        }
    );
}
```

Abbildung 59 - FetchCaches

14.3.7.1. QR-Codes drucken

Das Generieren des QR-Codes für einen Cache läuft am Server ab. Dieser speichert für jeden Cache eigens einen QR-Code und stellt diesen per REST-Aufruf zur Verfügung.

```
printCache(id: number)
{
    window.open("http://adquest.htl-perg.ac.at/AdQuest/cache/"+id+"/pdf");
}
```

Abbildung 60 - printCache

Der Link wird aufgerufen und das übermittelte PDF wird in einem neuen Tab geöffnet, wo der Benutzer es anschließend drucken kann.

14.3.8. Einstellungen

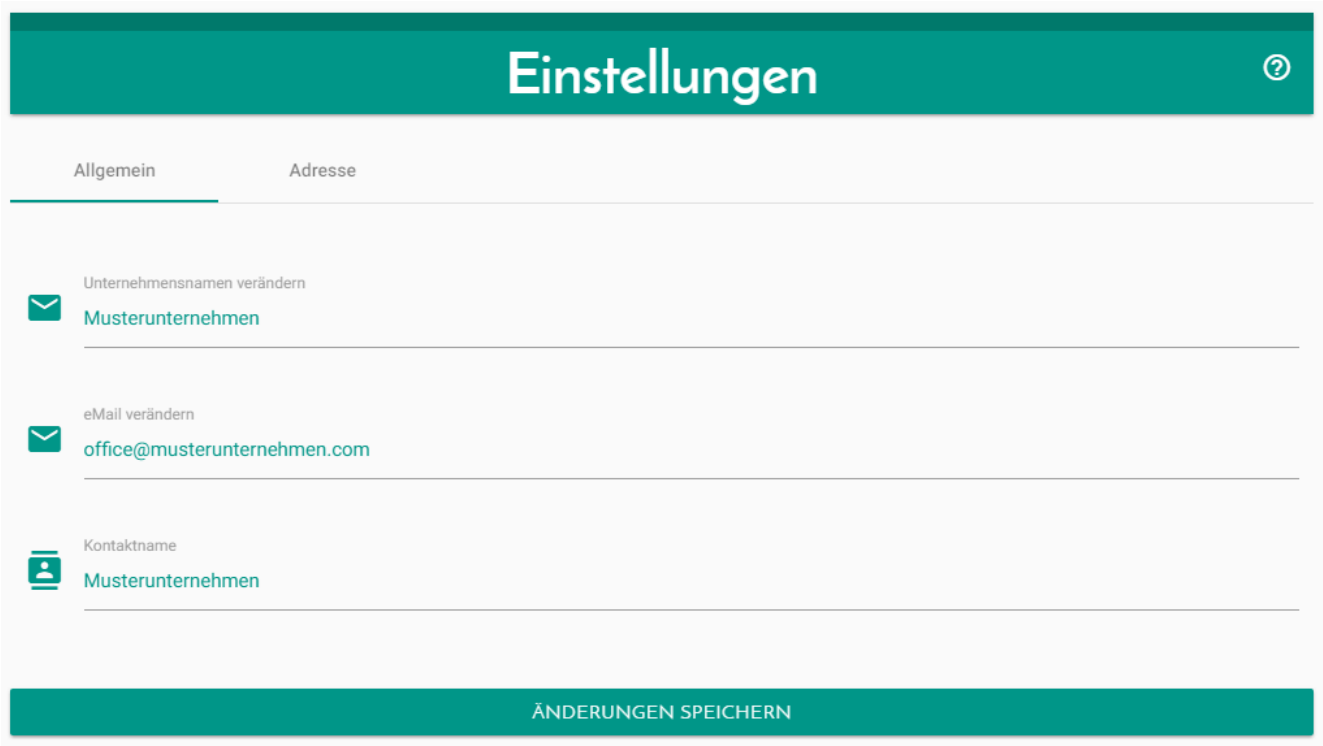


Abbildung 61 - Einstellungen

Auf der Seite „Einstellungen“ hat der Benutzer die Möglichkeit, seine Login-Daten, bzw. Daten zu seinem Unternehmen zu ändern. Im Hintergrund steht der Profile-Component, als View dient die profile.html
Noch bevor der Component geladen wird, wird ein Request auf das aktuelle Unternehmen abgesetzt, um die Daten anschließend im Formular anzeigen zu können. Gespeichert werden die geänderten Daten mittels eines POST-Requests, welcher vom Profile-Service zur Verfügung gestellt wird.

```
updateCompany(company){
    let headers = new Headers({ 'Content-Type': 'application/json',
      'Authorization': 'Bearer ' + localStorage.getItem('auth_token') });
    let options = new RequestOptions({ headers: headers });
    return this.http.put("http://adquest.htl-perg.ac.at/AdQuest/company/"
      +localStorage.getItem('logged_in_id'), JSON.stringify(company),options);

}
```

Abbildung 62 - updateCompany

14.3.9. Lade-Komponente

Ein wesentlicher Bestandteil jeder Applikation ist das Feedback an den Benutzer nach dem Ausführen bestimmter Aktionen. So ist es für die Benutzerfreundlichkeit immer wünschenswert, etwas am Bildschirm geschehen zu lassen, während beispielsweise ein Request Daten vom Server holt.
Zu diesem Zweck implementiert AdQuest einen sogenannten „Spinner“. Dieser überblendet während dem Laden von Daten den angezeigten Component und vermittelt dem Benutzer das Gefühl, dass das System im Hintergrund arbeitet.

```
<div id="content" style="...">
    <div *ngIf="loadingService.requesting">
            <div  style="...">

            </div>
        <md-spinner style="..."></md-spinner>
    </div>
    <router-outlet></router-outlet>
</div>
```

Abbildung 63 - spinner

Wie man dem oben gezeigten Code-Ausschnitt entnehmen kann, wird der „md-spinner“ nur angezeigt, wenn aktuell ein Request ausgeführt wird. Umgesetzt wurde dies durch ein *loadingService*, welches die Variable „*requesting*“ als Boolean hält, und somit Auskunft gibt ob aktuell ein Request performed wird. Verändert wird dieser Wert im „*authenticatedhttp.service*“. In diesem eigens geschriebenen Service, welches die Klasse http erweitert, wird die Methode „*request*“ überschrieben. Diese Erweiterung wirkt somit auf alle von AdQuest abgesetzten Requests.

```
request(url: string | Request, options?: RequestOptionsArgs): Observable<Response> {
  this.loadingService.requesting = true;
  this.requests.push(1);
```

Abbildung 64 – Überschreibung_request

Sobald ein Request gestartet wird, wird der Wert der *requesting*-Variable auf „true“ gesetzt, somit weiß das System das aktuell ein Request läuft. Des Weiteren wird in einer Request-Liste eine Stelle eingefügt. Dies hat den Hintergrund das theoretisch die Möglichkeit besteht, zwei Requests hintereinander abzusetzen. Der Wert der Boolean-Variable soll aber erst dann auf „false“ gesetzt werden, wenn auch wirklich kein Request mehr abläuft. Der Boolean-Wert wird also erst dann auf „false“ gesetzt, wenn die Liste wieder Länge 0 hat, sprich aktuell kein Request abläuft.

14.3.10. Produktauswahl

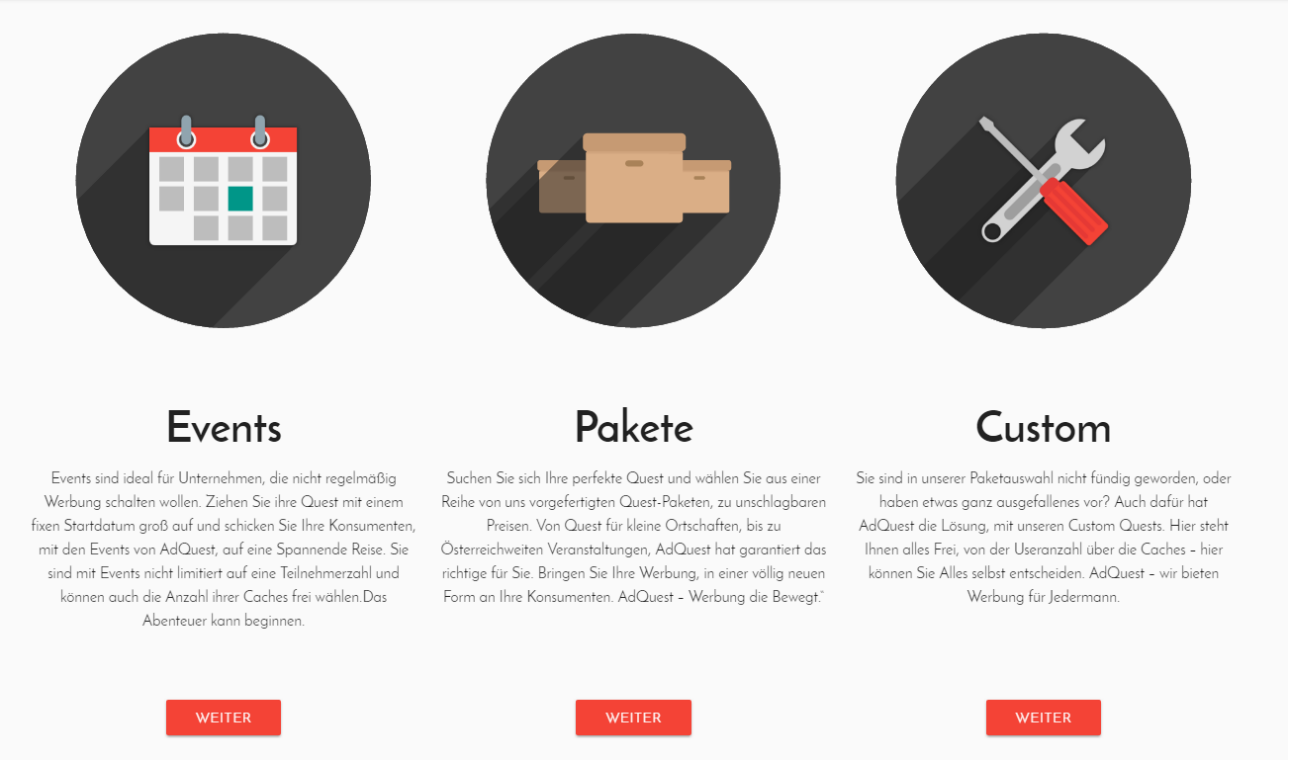


Abbildung 65 - Produktwahl

Unternehmen haben bei AdQuest grundsätzlich die Möglichkeit zwischen drei verschiedenen Produkten zu wählen:

- Events → Quest mit festem Startdatum
- Pakete → vorgefertigte Quests mit fester Cache und Userzahl
- Custom → Userzahl, Caches usw. können vom Kunden selbst bestimmt werden

Die Zahlung erfolgt dabei auf verschiedenen Wegen, abhängig vom gewählten Produkt. Events und Custom-Quests können beispielsweise nur per Rechnung bezahlt werden. Diese wird am Server generiert und dem Kunden zugesendet.
Pakete dagegen können auch direkt über die PayPal-Anbindung bezahlt werden.
Die Grafiken der Paketwahl wurden alle selbst designed.

14.3.11. CustomQuest-Formular / Event-Fomular

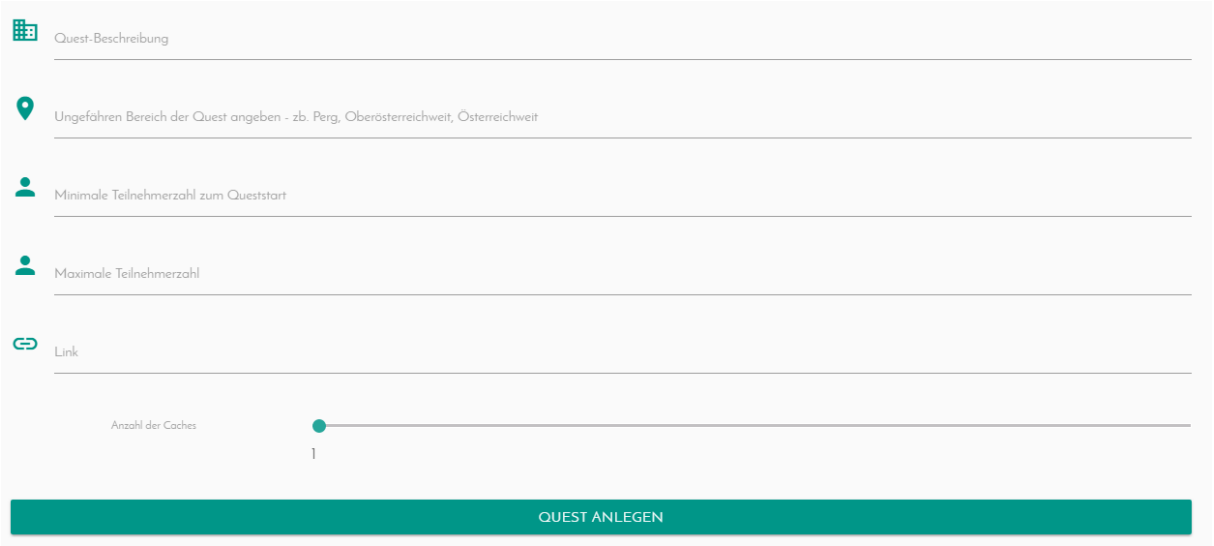


Abbildung 66 - CustomQuestForm

Wählt der Benutzer in der Produktwahl die Option „Custom Quest“ wird er auf das Custom-Quest Formular weitergleitet. Hier gibt er die Daten zu seiner Quest ein und legt sie anschließend an. Als Zahlungsmethode ist fix „per Rechnung“ definiert, diese wird automatisch am Server generiert und an den Benutzer gesendet. Das Event-Formular ist ident mit dem Custom-Fomular. Einziger Unterschied ist das Ersetzten der Teilnehmerzahlen, mit einem DatePicker um das Startdatum auszuwählen.

14.3.12. Paketwahl

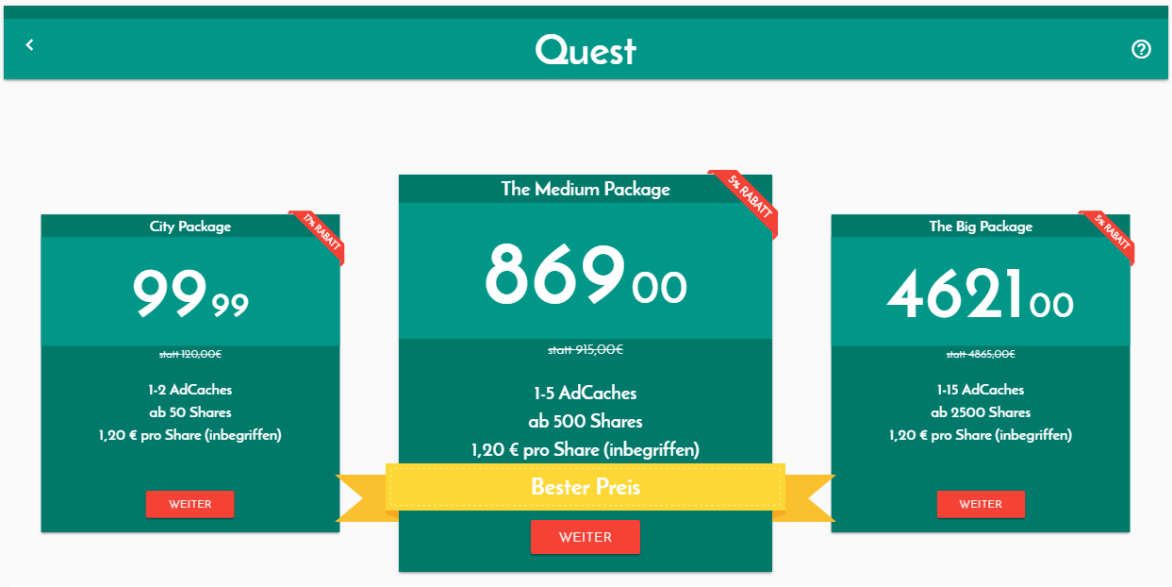


Abbildung 67 - Packagewahl

Hier hat der Benutzer die Wahl zwischen verschiedenen, aktuell bei AdQuest im Angebot stehenden Paketen. Bis auf die Ribbons, sind die angezeigten Kacheln eigens in CSS designed.

14.3.13. PayPal

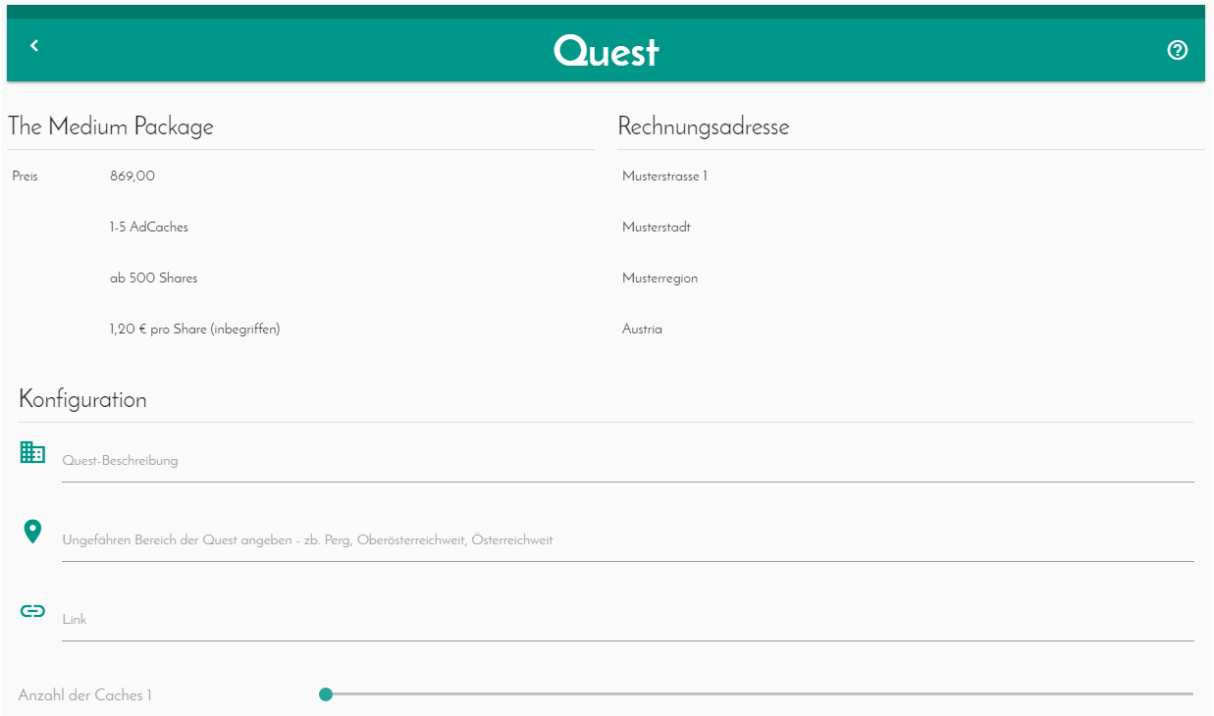


Abbildung 68 - PackageFormular

Wird bei der Paketwahl ein Paket ausgewählt wird der Benutzer auf die oben gezeigt Formularseite weitergeleitet.
Mittels Forms-Validation wird nun überprüft ob der Benutzer das ganze Formular korrekt ausgefüllt hat.

```
<div class="row content-fade" *ngIf="validForm">
  <div class="col offset-s1 s10">
    <h5>Zahlungsart</h5>
```

Abbildung 69 - FormsValid

Mittels **ngIf* wird überprüft ob das Fomular korrekt ausgefüllt wurde, nur wenn dies zutrifft wird das Feld „Zahlungsart“ eingeblendet.

14.3.13.1. Forms-Validation

```
this.questForm = new FormGroup({
  desc: new FormControl(),
  loc: new FormControl(),
  min: new FormControl(),
  max: new FormControl(),
  link: new FormControl(),
  caches: new FormControl()
});
```

Abbildung 70 - QuestFormGroup

Das ganze Formular ist in eine *FormGroup* zusammengefasst. Eine *FormGroup* ermöglicht das Abprüfen, der einzelnen Felder in der *Form*.
Im Falle von AdQuest wird, sobald alle Felder ausgefüllt sind, die *Group* auf „valid“ gesetzt und die Zahlungsart kann angezeigt werden.

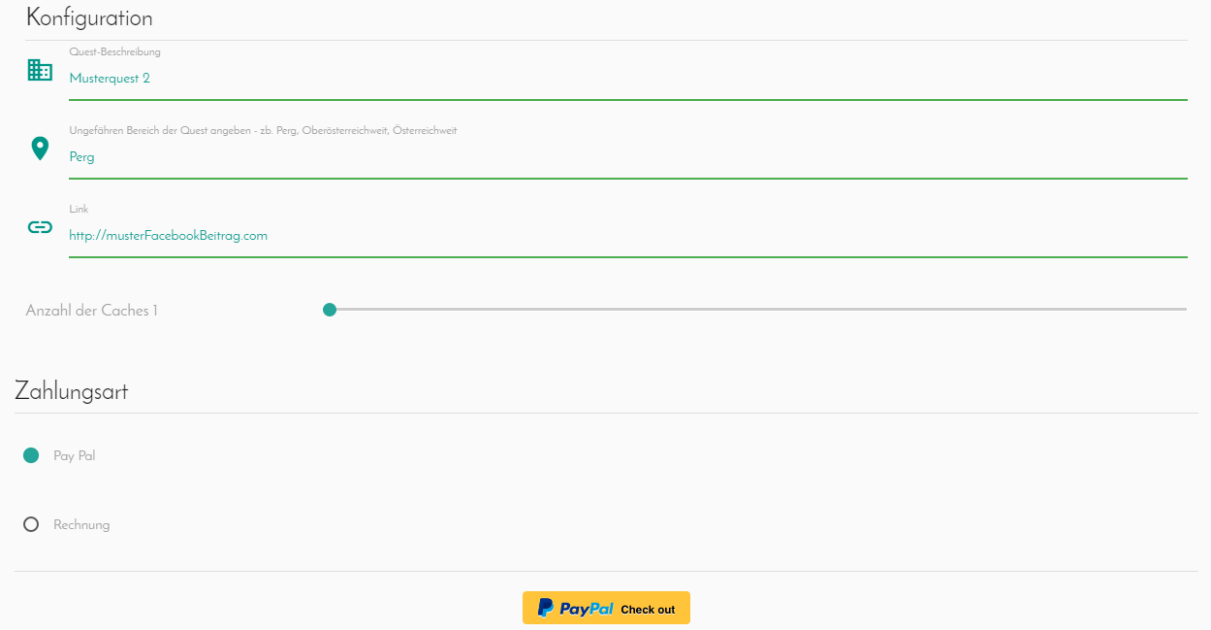


Abbildung 71 – PaymentMethod

Die vorangehende Abbildung zeigt die Auswahl zwischen PayPal und Rechnung.

```
<form #form class="form-inline" name="_xclick" action="https://www.paypal.com/cgi-bin/webscr" method="post">
  <input type="hidden" name="cmd" value="_s-xclick">
  <input type="hidden" name="hosted_button_id" value="{{package.paypalbuttonid}}">
  <input type="hidden" name="custom"value="{{quest.description}}">
  <input (click)="createQuest();form.submit(); " type="image"
         src="https://www.paypalobjects.com/webstatic/en_US/i/btn/png/gold-rect-paypalcheckout-34px.png"
         border="0"
         name="submit" alt="Make payments with PayPal - it's fast, free and secure!"/>
</form>
```

Abbildung 72 - PayPalForm

Nach dem Erstellen eines PayPal-Business Accounts können online sogenannte „PayPal-Buttons“ erstellt werden. Diese Buttons haben immer ein spezielles Produkt bzw. einen speziellen Betrag hinterlegt, welcher beim Erstellen des Buttons angegeben wird. In AdQuest gibt es demnach 3 verschiedenen Buttons, welche wie in Kapitel 14.3.7 mittels ihrer *id* über *ActivatedRouts* identifiziert werden.
Die in der Abbildung gezeigte *form* ist automatisch von PayPal generiert und stellt den Button in der Website dar. Zur Datenüberprüfung werden einige Daten in Form von *hidden-inputs*, also versteckten Textfeldern, an PayPal mitgesendet, wie etwa die Produkt-ID zur Identifikation des Buttons und eine Beschreibung des Produktes. Wird der Button nun geklickt wird ein Post-Request an PayPal gesendet, und die Weiterleitung erfolgt automatisch. Ist die Zahlung abgeschlossen wird der Benutzer wieder auf die Quests-Tabelle umgeleitet.

14.3.14. Kontaktformular

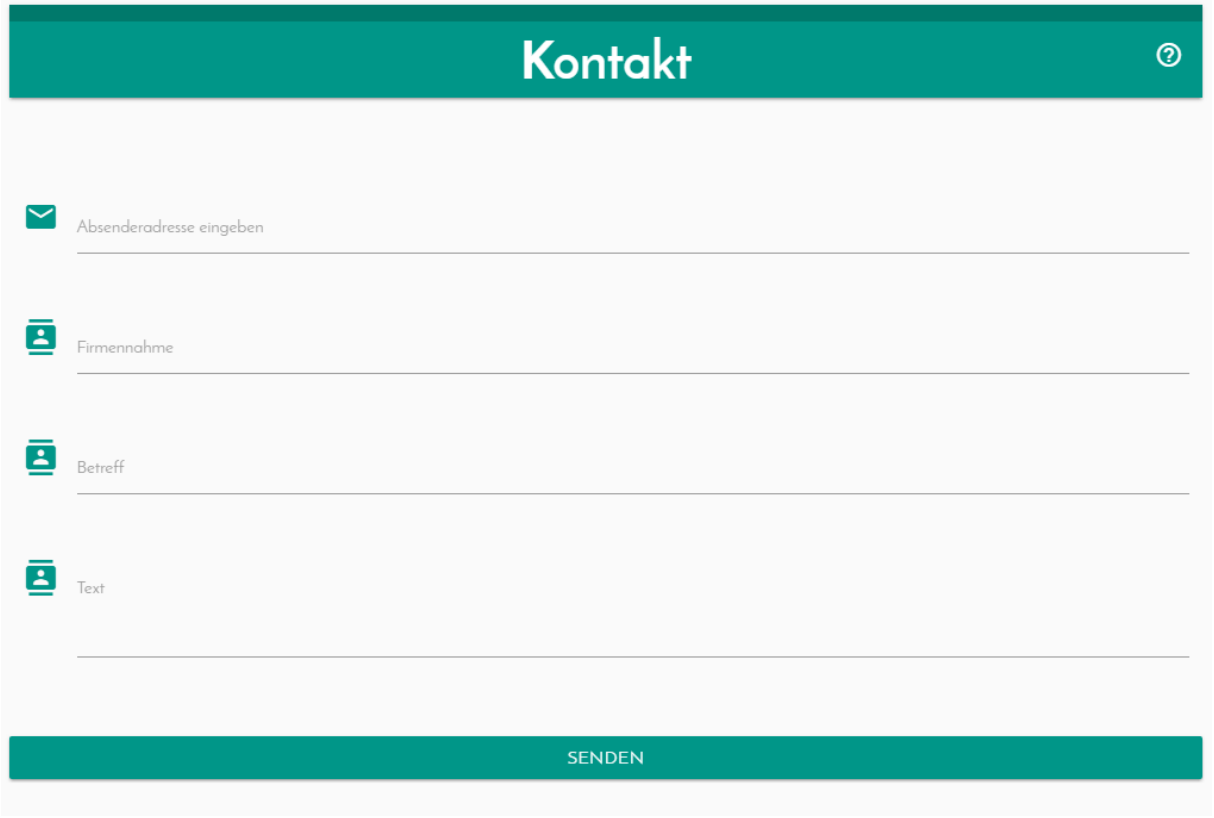


Abbildung 73 - Kontaktformular

AdQuest bietet auch die Möglichkeit der Kommunikation mit dem Entwicklerteam. Über das Kontaktformular von AdQuest können Fragen, Beschwerden, und Anregungen direkt an die AdQuest Email-Adresse gesendet werden, wo sie anschließend in der Administrationsoberfläche gelöst werden können. Ist ein Benutzer bereits eingeloggt sind die Absenderadresse und der Firmenname schon im Vorhinein automatisch ausgefüllt.
Wird der „Senden“-Button gedrückt werden die Formulardaten mittels eines POST-Requests an den Server geschickt, welcher die Mail anschließend auf das AdQuest-Postfach weiterleitet.

14.4. Design

Designtechnisch orientiert sich die AdQuest-Website am Material Design von Google.
Zu diesem Zweck wurden Libraries und vorgefertigte CSS-Klassen verwendet.

14.4.1. MaterializeCSS

MaterializeCSS ist eine open-source CSS-Library welche eine große Menge an vorgefertigten CSS-Klassen, Komponenten und JavaScript-Funktionen bietet um ein ansprechendes Design zu bieten.
Entwickelt wird MaterializeCSS von einem kleinen Entwicklerteam der Carnegie Mellon University. (vgl. Materializecss.com, 2017)

14.4.2. Angular2Material

Angular2 Material ist ebenso wie MaterializeCSS eine auf dem Material Design basierende Design-Library, die in diesem Fall speziell auf die Verwendung mit Angular2 ausgerichtet ist.

15. Android-Applikation

Die Android Applikation AdQuest wurde aufgrund mehrerer Faktoren mit Android Studio entwickelt.
Android Studio hat gegenüber anderen Entwicklungsumgebungen für Android mehrere Vorteile. Einerseits wurde Android Studio von Google, welches auch für das Android Betriebssystem verantwortlich ist, entwickelt und gilt somit als offizielle Entwicklungsumgebung des Android Systems. Andererseits erfreut sich Android Studio allgemein großer Beliebtheit. Daraus resultierend existiert bereits eine große Community und eine umfassende Dokumentation, welche das Entwickeln stark vereinfachen.

15.1. Grundidee

Die Idee der Mobile Android Applikation für AdQuest ist Teil des gesamten Geschäftskonzeptes. Sie soll vor allem den Werbekonsumenten in unserem System nutzen. Ziel ist es, mit dieser Applikation Werbekonsumenten auf unser System aufmerksam zu machen und diese als langfristige User zu gewinnen.

Wichtigste Grundfunktion der Android Applikation, ist das Anzeigen von Quests und Events inklusive etwaiger Zusatzinformationen, wie z.B. aktuelle Teilnehmerzahlen oder Start – sowie Enddatum. Darüber hinaus soll dem User ermöglicht werden Quests oder Events, an denen er teilnehmen möchte, auf Facebook zu teilen, wodurch AdQuest für unsere Business Kunden zum wirksamen Marketingtool wird. Bevor der User auf die Inhalte des Systems zugreifen kann, ist es notwendig sich mittels Facebook-Login anmelden. Diese Art von Authentifizierung, hat einerseits den Vorteil, dass sich der Benutzer der Applikation beim Teilen eines Beitrages bzw. Events nicht erneut auf Facebook einloggen muss, andererseits macht es die Entwicklung eines eigenen Authentifikationssystems obsolet.

Zusätzlich zu den oben genannten Funktionen stellt AdQuest für jeden Benutzer, seine bereits teilgenommenen Suchen in übersichtlicher Form dar. Ist eine Quest oder ein Event gestartet soll jeder User, welcher das Quest oder Event geteilt hat, die ungefähren Koordinaten der versteckten Schätze, in Form einer Google Map Oberfläche mit Markern erhalten. Damit die Suche spannend bleibt, soll der User nur ungefähre Koordinaten in Form eines Umkreises erhalten.
Da nicht die tatsächlichen Produkte unserer Werbekunden in der Natur versteckt werden, sondern symbolisch dafür QR-Codes, beinhaltet unsere Applikation einen QR-Code Reader. Durch Scannen des Codes, wird der Inhalt direkt an unseren Server gesendet und in zwei Schritten überprüft. Im ersten Schritt wird kontrolliert ob der Finder berechtigt ist und danach ob der gesendete Code nicht schon von einem anderen User gescannt wurde. Trifft keiner dieser Fälle zu, ist der Benutzer, welcher den QR-Code gescannt hat der Gewinner der Schatzsuche. Das veranstaltende Unternehmen, setzt sich danach mit dem Gewinner in Verbindung und lässt ihm das versprochene Produkt zukommen.

Damit der Benutzer auch beweisen kann, dass er der Finder dieser Suche ist, soll unserer Applikation dem User eine Übersicht aller bisher gefunden Schätze bieten. Diese beinhaltet neben dem Funddatum auch ein Satellitenbild, wo der Schatz gefunden wurde. Darüber hinaus soll den Benutzern eine Hilfestellung zur Bedienung der Applikationen angeboten werden. Durch ein einfaches Tutorial sollen User schnell in unser System finden und dieses gut bedienen können.

15.2. Technik

Das Programmierkonzept von Android Applikationen, ist Java sehr ähnlich. Android baut grundsätzlich auf diese Programmiersprache auf und weist lediglich einen Unterschied auf. Nämlich jener, dass sogenannte Activities mit dem Model-View-Controller Konzept verwendet werden.

15.2.1. Activity

Eine Activity ist der Teil einer Applikation, in der es dem Benutzer möglich ist selber aktiv zu werden. Die meisten Activities kommunizieren mit dem User, beispielsweise durch Buttons oder Textfelder. Eine Activity bietet also dem Entwickler Platz, seine eigenen Elemente zu platzieren und die Aktionen dahinter zu definieren. Activities sind in Android als Aktivitäten Stapel organisiert. Wird eine neue Activity gestartet, wird sie auf die Spitze dieses Stapels abgelegt und wird zur laufenden Activity – also die Aktivität, welcher der User auf seinem Bildschirm wahrnimmt. Durch das Betätigen des Zurück-Button in Android, wird die oberste Aktivität vom Stapel genommen und die nächste darunter angezeigt.

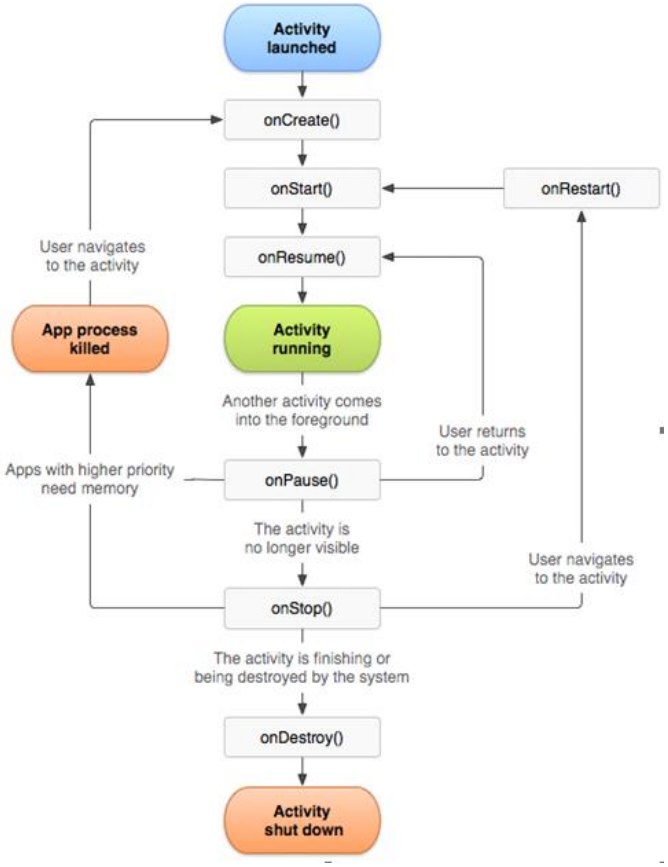


Abbildung 74 – Lebenszyklus_activity (Android.com, 2017)

Android Aktivitäten haben bestimmte Lebenszyklen. Durch das Erreichen eines Zustandes innerhalb des Lebenszyklus, wird intern eine Aktion – sprich eine Methode durch Android aufgerufen. Je nach erreichtem Status im Lebenszyklus sind diese Methoden unterschiedlich.

Um beim automatischen Aufruf dieser Methoden, eigene Aufgaben abarbeiten zu können, müssen diese in der eigenen Activity überschrieben werden.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

Abbildung 75 - onCreate

15.2.2. Aufrufzeitpunkt

onCreate()	Wird aufgerufen, wenn die Activity das erste Mal startet
onRestart()	Wird aufgerufen, wenn die Activity zuvor gestoppt wurde und danach wieder neu gestartet wird
onStart()	Wird aufgerufen, wenn die Activity für den Benutzer sichtbar wird
onResume()	Wird aufgerufen, wenn die Activity bereit ist mit dem User zu kommunizieren. Wird immer von onPause() gefolgt.
onPause()	Wird aufgerufen, wenn man sich auf einer Activity befindet und eine bereits zuvor gestartete Activity aufruft.
onStop()	Wird aufgerufen, wenn die Activity nicht länger sichtbar ist für den Benutzer. Ab diesem Zeitpunkt, hat es den "STOPPED" Status erreicht und diese Methode wird aufgerufen.
OnDestroy()	Wird aufgerufen, wenn die Activity "zerstört" wird. Das ist der letzte Aufruf, den die Activity bekommt und verarbeiten kann. Entweder wurde die finish() Methode aufgerufen oder die Applikation wurde geschlossen.

Tabelle 13 - Methoden-Aufruf-Zeitpunkte

15.3. Umsetzung

15.3.1. Vorkonfiguration

Die AdQuest Applikation ist eine Multi-Activity Anwendung, welche auf allen Android Geräten ab API-Level 21 lauffähig ist. Mit dem API-Level 21 erreichen wir ca. 67% aller Android Nutzer. API-Level 21 entspricht der Android-Version 5.0. Die nachstehende Grafik zeigt die Verteilung der Android-Versionen.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.0%
4.1.x	Jelly Bean	16	3.7%
4.2.x		17	5.4%
4.3		18	1.5%
4.4	KitKat	19	20.8%
5.0	Lollipop	21	9.4%
5.1		22	23.1%
6.0	Marshmallow	23	31.3%
7.0	Nougat	24	2.4%
7.1		25	0.4%

Abbildung 76 – AndroidVersions (Android.com, 2017)

15.3.2. Gründe für Einsatz von API-Level 21

- Unterstützung des Android Material Design
- Unterstützung von Notifications – User werden durch das System benachrichtigt und werden grundsätzlich außerhalb der eigentlichen Applikation angezeigt.

15.3.3. Benötigte Berechtigungen

Im Android System ist es notwendig, Berechtigungen welche die Applikation braucht, vorab zu definieren. Diese müssen im Android-Manifest festgehalten werden. Weiters ist es ab der Android Version 6.0 (API Level 23) erforderlich, bei bestimmten Berechtigungen, wie Kamera oder Standort, den User während des Betriebes der Applikation um Freigabe auf das benötigte Feature zu bitten.
Für AdQuest sind folgende Funktionen notwendig:

- CAMERA
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION
- INTERNET
- MAPS_RECEIVE
- READ_GSERVICES

15.3.4. Verwendete Activities

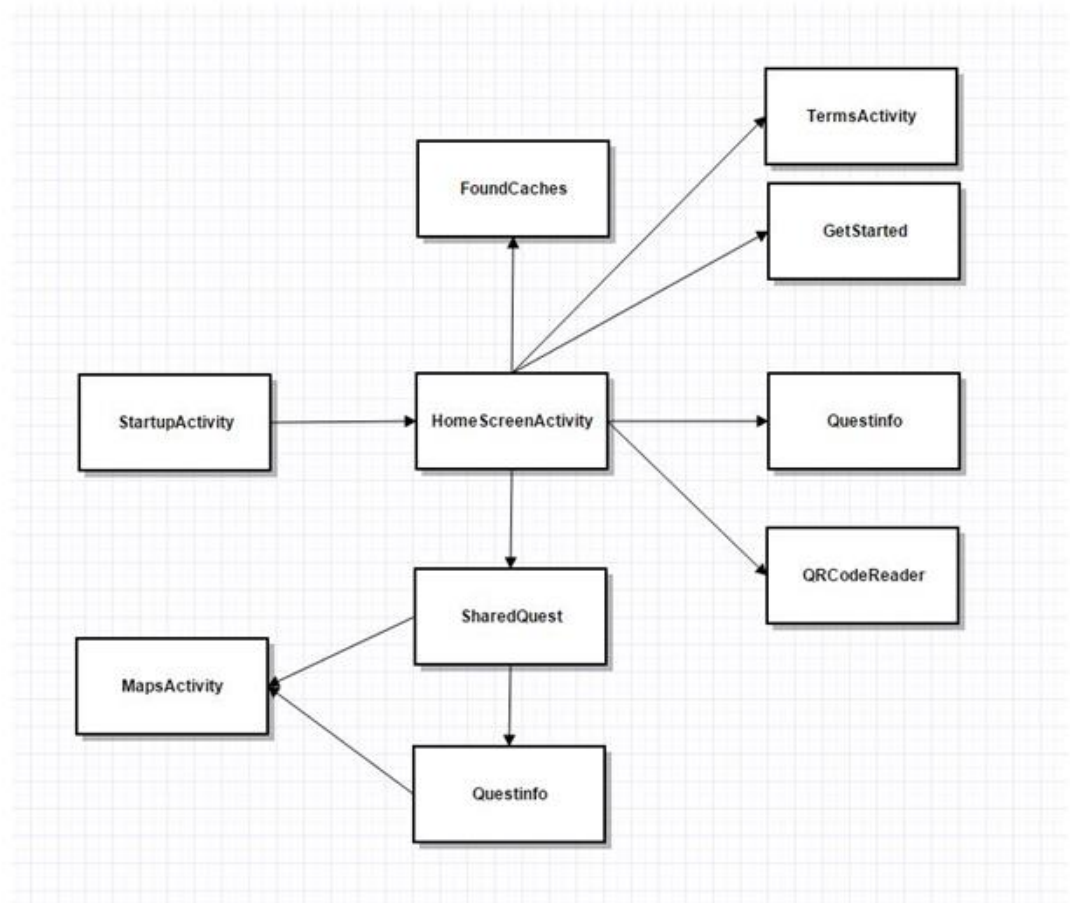


Abbildung 77 – verwendete Activities

Die obenstehende Grafik veranschaulicht den Aufbau der der AdQuest Applikation.

15.3.5. StartupActivity

Die StartupActivity ist der Startpunkt der Applikation. Der User meldet sich hier mittels seiner Facebook-Login-Daten an. Hat der Benutzer die Facebook App auf seinem Android Gerät bereits installiert, reicht für die Anmeldung ein Klick aus und es ist nicht mehr notwendig die Facebook Anmeldedaten erneut einzugeben. Umgesetzt wurde die Anmeldung mittels Facebook, mit Facebook SDK für Android, in der Version 4.6.0.

```
compile 'com.facebook.android:facebook-android-sdk:4.6.0'
```

Abbildung 78 – Einbinden der Facebook SDK

Abbildung 79 – Startup Activity

Die Facebook SDK, bietet neben der Anmeldefunktion noch weitere Funktionen, welche bei der Erstellung von AdQuest benötigt wurden. Wie in der Abbildung zu sehen ist, beinhaltet die Activity neben dem Hintergrund, noch zwei Buttons. Der erste Button löst die Anmeldeaktion aus, damit sich der Benutzer einloggen kann. Das Ergebnis des Logins, wird durch den Callback Manager aufgefangen und ausgewertet.

```
loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
    @Override
    public void onSuccess(LoginResult loginResult) { // User Sucessfully logged in
        startActivityfromHere(loginResult.getAccessToken().getToken());


    }
    @Override
    public void onCancel() { info.setText("Login attempt cancelled."); }

    @Override
    public void onError(FacebookException e) {
        info.setText(e.toString());
    }
});
```

Abbildung 80 - callbackRegister

Beim erfolgreichen Login wird die onSuccess() Methode aufgerufen, welche die neue Activity, also die HomeScreenActivity startet. Dieser wird der Facebook Access Token mitgeben, welcher den Benutzer eindeutig in unserem System identifiziert.
Beim Access Token oder Zugriffsschlüssel handelt es sich um einen verdeckten String, mit dem ein Nutzer identifiziert wird. Dieser wird benötigt um sicheren Zugriff auf die Facebook-Funktionen gewähren zu können.

Beispiel Token: EAAX5KNjW9I4BAMJ3azXko1mZCcnXsnKu0tMYGZA-Pym4P5u9ZAAj6RcZCjvcQag3aacfZCj9e7ezrrUtsO-EmqBs1lT4DP4i4bKa4qCtpp7rRo9KJnog6VK0OBEIRbrBnIsjunQntnCrsBMmAqst-DwfNfe2sOWEdQ8RJ2QpYOR2pIgadXW8Mn3G6NmYEVakoaHxOaarRyZAZB-Wij0fNxoS6nv

Dieser wird in späterer Folge an den Server gesendet, damit sich der Benutzer anmelden kann.

Weiters erfolgt in der Activity eine Weiterleitung an die nächste Aktivität, wenn der Benutzer schon angemeldet ist und diesen Access Token besitzt. Dies erleichtert dem Benutzer die Bedienung unserer Applikation, da er sich nicht jedes Mal wieder einloggen muss, wenn er die App beispielsweise geschlossen hat und danach neu öffnet.

```
AccessToken accessToken = AccessToken.getCurrentAccessToken();

if(accessToken != null) { // if not null he is logged in
    startActivityfromHere(accessToken.getToken());
}
```

Abbildung 81 - Token

Der zweite Button verweist auf unsere Webseite, um eventuelle Geschäftskunden dorthin zu leiten. Wird der Button gedrückt, so öffnet sich automatisch der Browser mit unserer Website.

15.3.6. HomeScreenActivity

Die HomeScreenActivity stellt das Herzstück der AdQuest Applikation dar. Der User wird hier über neue Quests sowie Events informiert. Diese werden übersichtlich in einzelnen Cardviews dargestellt. Zusätzlich werden dem Benutzer noch die aktuellen Teilnehmerzahlen angezeigt.

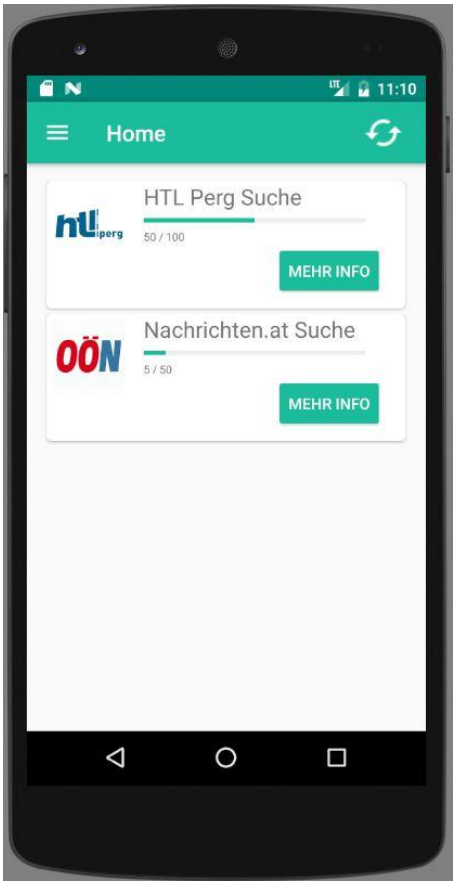


Abbildung 82 - HomeScreen

In der HomeScreenActivity passiert neben der Anzeige der Quests, welchen der User noch nicht beigetreten ist, auch die Login Funktion, die aber für den User unsichtbar ist.
Die Anzeige der Quests, basiert auf Daten, welche von unserem Server abgerufen werden.
Diese Daten werden durch einen JsonObjectRequest im JSON-Format abgerufen.

{"id":1,"description":"AdCache angelegt","name":"Angelegt"}

Abbildung 83 – JSON Zeichenkette

Wie in der Abbildung zu sehen ist, werden die Daten in Zeichen über das Internet vom Server an die App gesendet. Diese Zeichen können durch Umwandlung in echte Java-Objekte oder Basisdatentypen umgewandelt werden.

```
JSONObject obj = (JSONObject) response.get(i);
Quest quest1 = new Quest();
quest1.setId(obj.getInt("id"));
```

Abbildung 84 – JSON-Parse

Die Login Funktion, sendet an den Server den zuvor erlangten Facebook Access Token. In der Response des Aufrufes, erhält der Nutzer seine User-Identifikations-Nummer.

Ist der User noch nicht registriert, sprich der Access Token ist noch nicht beim Server bekannt, so wird automatisch eine andere Methode (doRegister()) aufgerufen. Die User ID, welche der Server bei erfolgreichem Einloggen bzw. Registrieren zurückliefert, wird in den sogenannten Shared Preferences gespeichert, damit jede Aktivität darauf zugreifen kann.

Shared Preferences ermöglichen das Laden und Speichern von Werten. Vorteil dieser Methode ist, dass systemweit darauf zugegriffen werden kann.

```
SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferences.Editor editor = pref.edit();
editor.putString("fbkey", k);
editor.commit();
```

Abbildung 85 - Erstellen der Shared Preferences

Die HomeScreenActivity ist zudem das „Zentrum“ der gesamten Applikation. Unter anderem wird in der HomeScreenActivity die Navigation durch AdQuest abgewickelt. Hierfür wurde auf den Navigation Drawer zurückgegriffen, welcher eine reibungslose und gutaussehende Form ermöglicht, sich durch das System zu bewegen.
Beim Klick auf eine gelistete Activity öffnet sich diese.

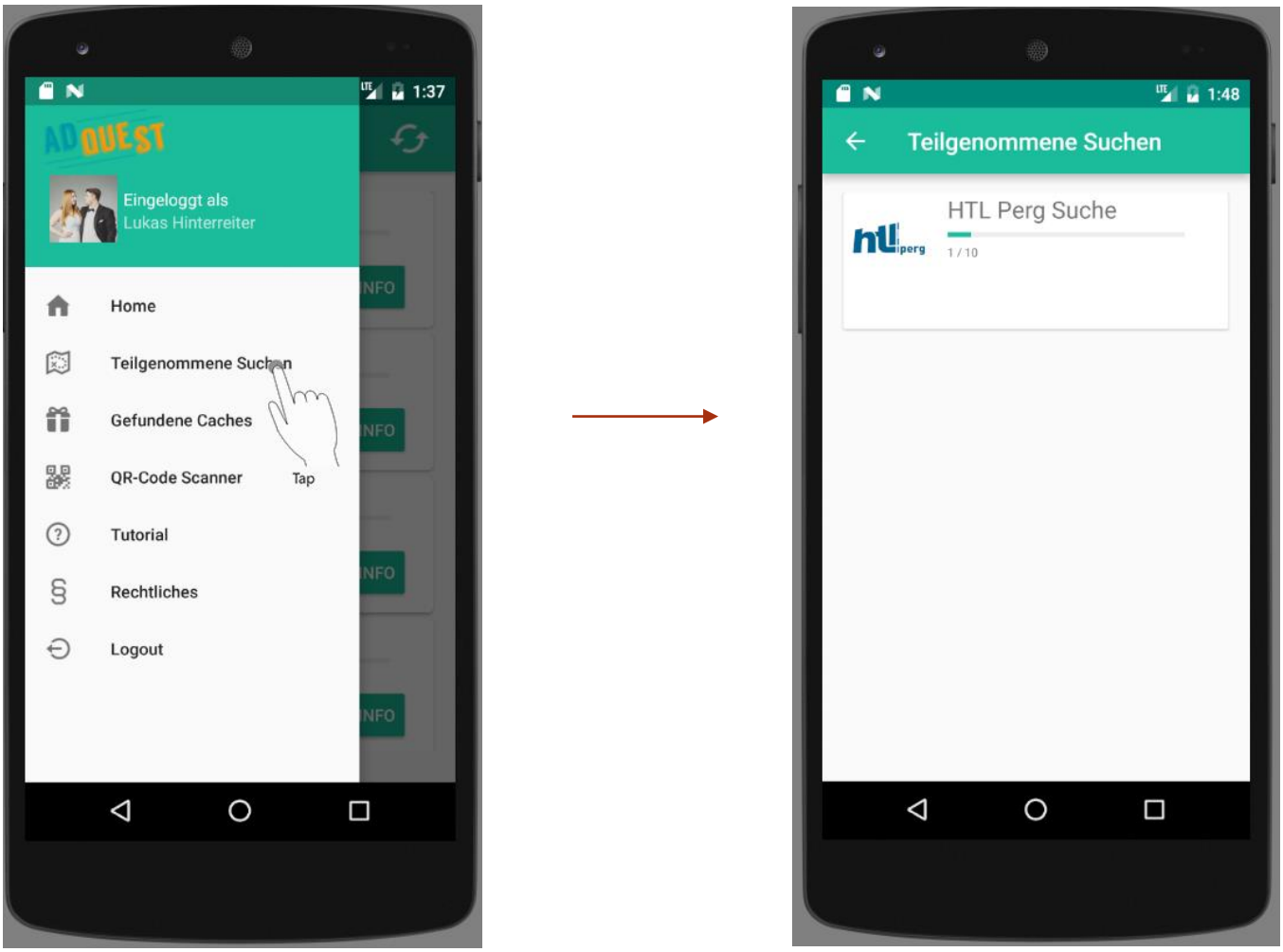


Abbildung 86 - Navigation Drawer Funktion

Durch einen Klick auf den Pfeil in der „Teilgenommene Suchen“ Activity gelangt man wieder zurück in die HomeScreenActivity.

Weiters werden im Navigation Drawer links oben, wie in der Abbildung zu sehen, der Name des Users und das Profilbild von Facebook übernommen und angezeigt. Realisiert wird dies durch die Facebook SDK und die darin enthaltene Klasse „Profile“. Diese Klasse besitzt alle Daten zum Profil des Users, welcher sich zuvor eingeloggt hat.

15.3.7. Questinfo

Die Activity Questinfo stellt eine Informationsseite dar. In ihr werden Informationen zu den Quests oder Events angezeigt, die den Nutzer interessieren könnten. Es wird der ungefähre Austragungsort, die Teilnehmerzahlen und eine Beschreibung des Quests oder des Events angezeigt. Beim Event wird zusätzlich noch das Startdatum angezeigt. Am Ende der Bildschirmseite wird immer der Facebook-Share Button angezeigt. Durch diesen kann der Nutzer bequem aus der Applikation die Beiträge teilen um teilzunehmen.

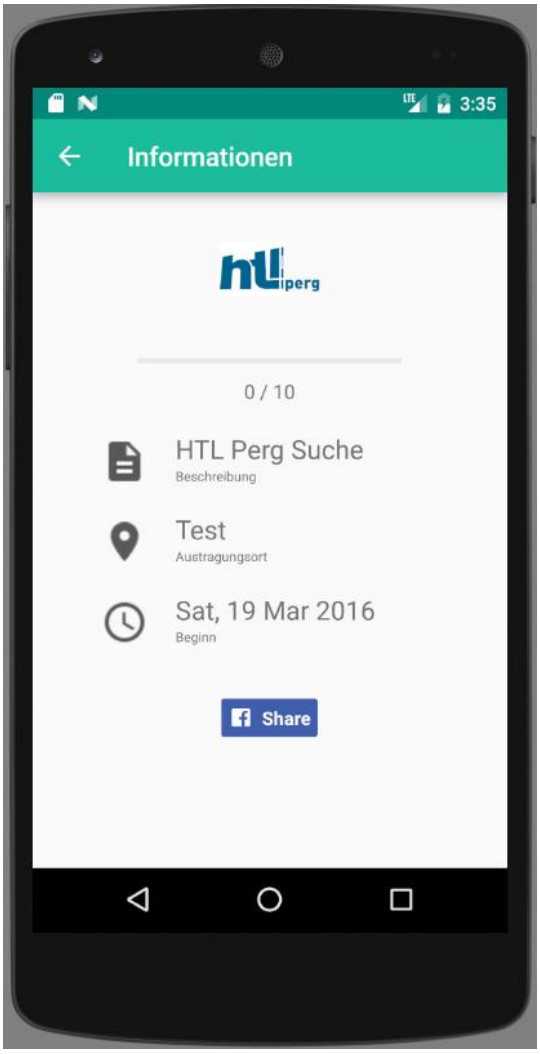


Abbildung 87 - Questinfo

Um prüfen zu können ob der User den Facebook Beitrag wirklich geteilt hat, müssen die Parameter, welche Facebook an die Applikation zurückliefert überprüft werden. Dazu muss unterschieden werden zwischen zwei Situationen.

Entweder hat der User die Facebook Applikation auf seinem Handy installiert oder nicht. Das ist wichtig, da die Parameter die von der Facebook-App zurückgeliefert werden, bei einem erfolgreichen Share unterschiedlich sind.

Daher ist es nötig, beide Fälle zu überprüfen und abzudecken, damit alle Facebook User unsere Applikation nutzen können.

```
try {
    if (data.getExtras().getBundle("com.facebook.platform.protocol.RESULT_ARGS").get("completionGesture").equals("post")) {
        doPutQuest();
        Intent intent = new Intent(this, HomeScreenActivity.class);
        startActivity(intent);
    }
}catch (Exception e){
    try{
        if (!data.getExtras().getBundle("com.facebook.platform.protocol.RESULT_ARGS").get("post_id").equals("")) {
            doPutQuest();
            Intent intent = new Intent(this, HomeScreenActivity.class);
            startActivity(intent);
        }
    }catch (Exception k){
        System.out.println(k.getMessage());
    }
}
```

Abbildung 88 - Share Prüfung

Wie in der Abbildung zu sehen ist, existieren die oben beschriebenen Möglichkeiten. Beim ersten ist im Objekt „RESULT_ARGS“ das Feld „completionGesture“ auf „post“ gesetzt. Dies ist nur der Fall, wenn der User auch die Facebook-App installiert hat. Im zweiten Fall ist im Objekt „RESULAT_ARGS2 eine „post_id“ eingetragen. Treffen beide Fälle nicht zu, so wurde der Post nicht geteilt und der User darf nicht an der Quest oder dem Event teilnehmen.

Hat der User jedoch den Beitrag erfolgreich mit seinen Freunden geteilt, so wird ein PUT auf unseren Server abgesetzt und der Quest oder das Event wird dem User zugeordnet und von nun an nicht mehr am Home Screen angezeigt, sondern in der SharedActivity.

15.3.8. Shared Activity

Die Shared Activity stellt alle Quests oder Events dar, deren Beitrag schon vom User auf Facebook geteilt wurde. Wie in der Homescreen Activity besteht diese aus mehreren Cardviews, welche das Bild und die aktuellen Teilnehmerzahlen anzeigen. Der einzige Unterschied ist, dass bei einem bereits gestartetem Quest oder Event ein Button in der Cardview, sowie in der Questinfo erscheint, welcher zur Google-Map weiterleitet. In dieser Map können dann die ungefähren Orte gesehen werden, wo die QR-Codes versteckt sind.

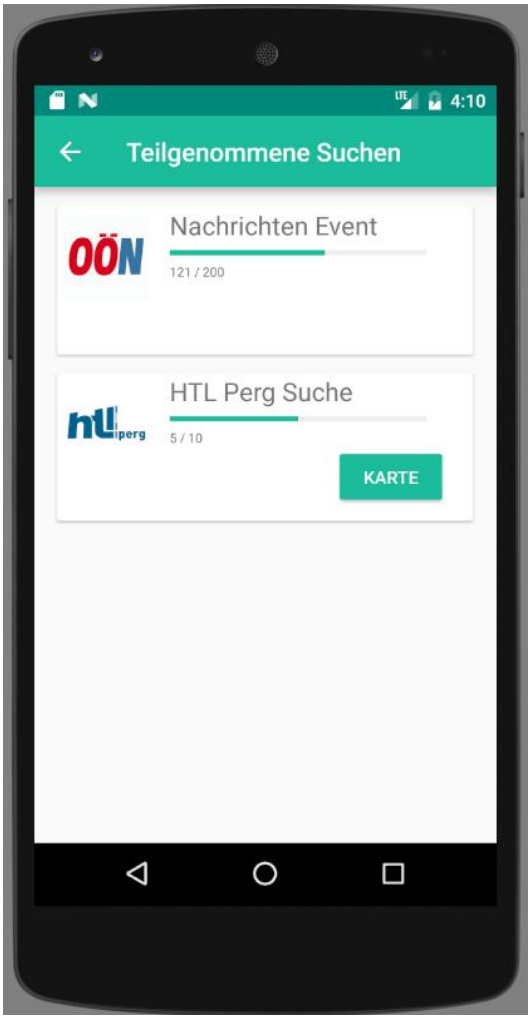


Abbildung 90 - Shared Activity Cardview

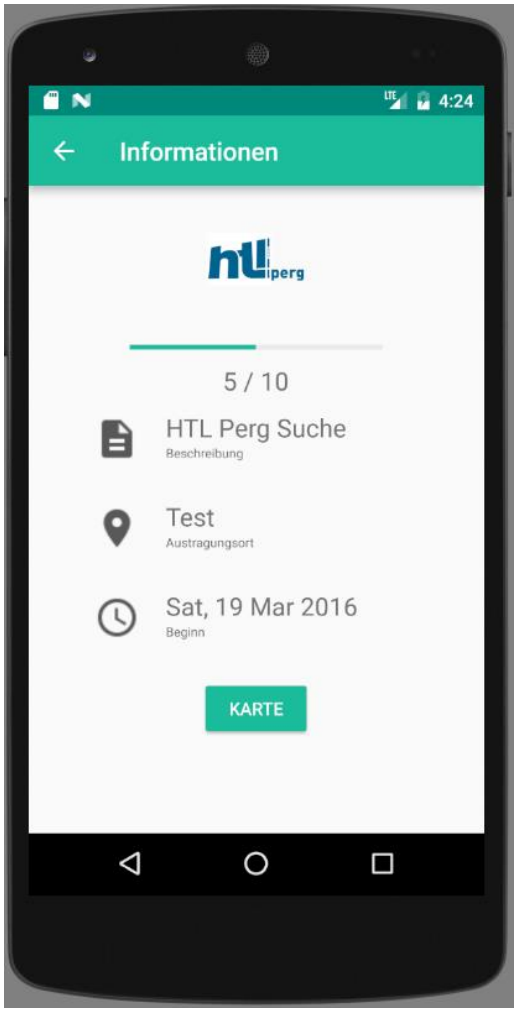


Abbildung 89 - Questinfo ohne Share-Button

Die in der Abbildung zu sehende Questinfo ist ähnlich zu der Questinfo des Home Screens. Der Share-Button wurde durch den „Karte“-Button ersetzt, welcher aber erst angezeigt wird, wenn die Quest oder das Event gestartet ist.

15.3.9. MapsActivity

Die Maps-Activity ist grundsätzlich für die Darstellung der Caches zu einer Quest oder einem Event verantwortlich. Diese werden auf einer Google-Map-Oberfläche, mithilfe eines Umkreises dargestellt. Damit der Cache auch von weitem auf der Karte zu sehen ist, wird im Mittelpunkt ein Marker gesetzt. Dies ist nötig, da die Umkreise beim herauszoomen aus der Karte verschwinden und so bei einer großen Darstellung nicht mehr sichtbar wären.

Abbildung 92 - Karte ohne Marker (Weitansicht)

Abbildung 91 - Karte mit Marker (Weitansicht)

Abbildung 93 - Karte mit Marker (Nahansicht)

Wie in der Abbildung oberhalb zu sehen ist ein blau eingefärbter Umkreis auf der Maps Oberfläche zu sehen. Der Umkreisradius ist variabel einstellbar. Wir haben als Voreinstellung zehn Meter ausgewählt. Der Umkreismittelpunkt wird zufällig berechnet. Dieser wird immer mit maximal zehn Meter Versatz zum echten Standort des QR-Codes gemacht, da dieser möglicherweise außerhalb des auf der Karte zu sehenden Umkreises liegen könnte.

Die Maps Activity sendet zu dem automatisch alle fünf Sekunden, den aktuellen Standort des Users zum Server. Der Server nutzt diese Daten um festzustellen ob sich ein Suchender in der Nähe eines Caches befindet.

Wichtig ist diese Information vor allem dann, wenn eine Drohnenbuchung zu diesem Cache vorliegt. Betritt der User einen vorgegebenen Umkreis um den Cache, so wird die Drohne automatisch gestartet.

```
String url = "http://adquest.htl-perg.ac.at/AdQuest/cache/" + this.id +
"/gps?userid="+userid+"&&longitude="+lastLocation.getLongitude()+"&&lati-
tude="+lastLocation.getLatitude();
```

Abbildung 94 - Senden der Location an den Server

Das Senden des Standortes funktioniert automatisch über das Aufrufen dieser URL. Die Parameter sind dabei die Quest- oder Event ID (this.id), die User ID (userid) und die Längen- und Breitengrade.

15.3.10. QR-Code Reader

Der QR-Code-Reader hat in der AdQuest Applikation eine essentielle Funktion. Durch ihn werden die in der Natur gefundenen QR-Codes, gescannt und die dahinter versteckte Nummer an den Server gesendet. Der Reader wurde als Library (ZXing) eingebunden. Sie verwendet die Kamera des Smartphones und scannt den vorgehaltenen QR-Code und wandelt die enthaltenen Informationen in Textform um.

Bei einem erfolgreichen Scan, wird eine Meldung an den User ausgegeben, welchen ihn als Finder deklariert. Dieser Cache wird, ab diesem Zeitpunkt, keinem Nutzer mehr angezeigt und der QR-Code verliert seine Gültigkeit. Der Finder, kann seine gefundenen Caches in der Found Caches Activity ansehen.

Abbildung 95 - QR-Code Reader

15.3.11. Found Caches Activiy

Die Found Caches Activity ist für die Anzeige der gefundenen Caches. Der Benutzer sieht neben dem Funddatum, eine Beschreibung zum gefundenen Schatz sowie einen Kartenausschnitt, wo er diesen gefunden hat.

Der Karten ausschnitt wird mittels einer statischen Google-Map angezeigt. Google bietet hierfür eine eigene API. Diese liefert beim Aufruf einer URL ein Bild des betroffenen Kartenausschnittes mit. Unter Angabe von Koordinaten, die darauf befindlichen Marker und den Zoom auf die Karten, liefert Google den Ausschnitt an unsere Applikation zurück.

Dieses Bild wird automatisch für jeden gefundenen Cache generiert und in die Pager-Seite geladen.

Ein Pager ist eine einfache Möglichkeit, mehrere Bildschirmseiten in einer Activity anzuzeigen. Durch das Wischen nach rechts oder links, wird auf die nächste bzw. vorherige Bildschirm Seite gescrollt.

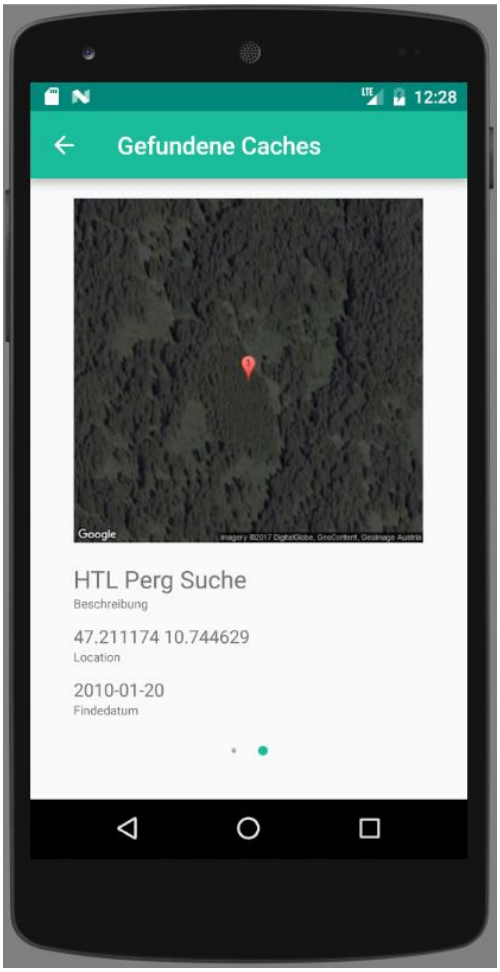


Abbildung 96 - Pager mit gefundenen Caches

Am unteren Ende der Activity wird noch zusätzlich angezeigt, auf welcher Seite man sich derzeit befindet. Der grüne Punkt gibt dabei die aktuelle Seite an, die grauen Punkte jene Seiten, welche man sich noch ansehen kann.

15.3.12. GetStarted und TermsActivity

Beide Activities sind zur Kommunikation mit dem Benutzer vorgesehen. Die Terms Activity beinhaltet eine ListView mit allen rechtlichen Bestimmungen, welche für die Benutzung von AdQuest einhalten werden müssen. Sollten sich User nicht daran halten, können ihnen dafür Gewinne aberkannt werden. Weiters könnten sie ausgeschlossen werden, und deren Facebook-Benutzer in unserem System gesperrt werden.

15.3.12.1. Rechtliche Bestimmungen

- Mit der Nutzung der AdQuest App und dem darin enthaltenen Goodie Programm erklärt sich der Nutzer mit diesen Teilnahmebedingungen einverstanden.
- Es gilt österreichisches Recht und der Rechtsweg ist ausgeschlossen. Gewinne können nicht in Bargeld abgelöst werden.
- Bei Verstoß gegen die Nutzungs- und Teilnahmebedingungen behält sich AdQuest das Recht vor, Personen von der Teilnahme auszuschließen bzw. Goodies nicht auszugeben bzw. nachträglich abzuerkennen und zurückzufordern.
- Ausgeschlossen werden auch Personen, die sich unerlaubter Hilfsmittel bedienen oder sich anderweitig durch Manipulation Vorteile verschaffen. Gegebenenfalls können in diesen Fällen Gewinne auch nachträglich aberkannt und zurückgefordert werden.
- Um an einer Quest teilzunehmen muss der dazugehörige Beitrag geteilt und bis auf weiteres auf der Facebook-Seite des App Benutzers ersichtlich sein. Sollte der Beitrag zwischenzeitlich gelöscht werden, erfolgt keine Aushändigung der Goodies.
- AdQuest hat nichts mit der Aushändigungen der Gewinne zu tun. Goodiepartner sind für die Aushändigung deren Gewinne verantwortlich.
- Ansprüche wegen möglicher Sach- und/ oder Rechtsmängel an den von den Goodiepartnern gestifteten Gewinnen sind ausschließlich diesen gegenüber geltend zu machen.
- AdQuest haftet nicht für die Insolvenz eines Goodie Partners sowie die sich hieraus für die Durchführung des Goodie Programmes ergebenden Folgen.
- Diese Teilnahmebedingungen können jederzeit von AdQuest ohne gesonderte Benachrichtigung geändert werden.
- AdQuest behält sich vor, einzelne Quest oder das gesamte Programm, zu jedem Zeitpunkt ohne Vorankündigung und ohne Angabe von Gründen abzubrechen oder zu beenden.
- Sollten einzelne dieser Bestimmungen ungültig sein oder werden, bleibt die Gültigkeit der übrigen Teilnahmebedingungen hiervon unberührt.

Abbildung 97 - Rechtliche Bestimmungen

Die GetStarted Activity ist eine Anleitung, welche es dem Benutzer ermöglicht, den Umgang mit der AdQuest Applikation zu lernen. Wie bereits in der Found Caches Activity, wird auch hier ein Pager verwendet, welcher es dem Benutzer erlaubt, sich durch mehrere Bildschirmseiten zu bewegen. Jede Bildschirmseite besteht aus einer selbsterklärenden Grafik und einem passenden Text dazu.

Abbildung 98 - Tutorial

Wie auch schon in der Found Caches Activity, wird auch hier durch die Punkte angezeigt, auf welcher Seite sich der Benutzer befindet. Mit einem Wischen nach rechts oder nach links bewegt sich der Benutzer durch die Bildschirmseiten.

16. Administrations- und Supportoberfläche

16.1. Grundidee

Die Administrations- und Supportoberfläche ist ein Webprogramm, welches es den AdQuest-Mitarbeitern ermöglicht, wichtige Geschäftsfunktionen abzuwickeln. Sie dient außerdem zur Unterstützung der Geschäftskunden bei Problemen verschiedenster Art. Die wichtigsten Administrationsfunktionen sind: das Bannen (Ausschließen von App-Benutzern bei einem Regelbruch), das Verwalten von Benutzern, Unternehmen, Quests, Admins und das Löschen von Unternehmen auf Anforderung, sowie die automatisierte Rechnungserstellung. Zu den Supportfunktionen gehören, die Unterstützung eines Kunden bei einem Registrierungsvorgang und das Bearbeiten von Quests. Jeder registrierte Admin hat vollen Zugriff auf alle Daten, deshalb sind hier Sicherheitsfunktionen besonders sorgfältig umgesetzt. Passwörter werden lediglich als Hashwert hinterlegt und eine Login-Autorisierung wird nach jeder Session beendet.

16.2. Technik – Java Server Faces

Java Server Faces (JSF) ist ein Framework zur Entwicklung von Webanwendungen. Es baut auf dem MVC (Model-View-Controller) – Konzept auf. Es werden XHTML-Pages entwickelt welche von einer Java Klasse (Servlet) dynamisch befüllt werden. Dies geschieht bei jeder Anfrage eines Clients an den Server, welcher das Java Projekt ausführt.
JSF unterstützt JPA (Java Persistence API) arbeitet mittels Objekt Relationales-Mapping extrem Datenbank nahe und ist somit perfekt geeignet für eine Administrationsoberfläche.

16.2.1. PrimeFaces

PrimeFaces ist eine JSF-Bibliothek, welche JSF um viele Funktionen und Komponenten erweitert. Unter anderem stellt es ein AJAX-Framework zur Verfügung, welches es ermöglicht die XHTML-Seiten dynamisch (ohne sie erneut zu laden) upzudaten.

16.3. Funktions- und Komponentenübersicht

16.3.1. Komponenten Überblick

Die Administrationsoberfläche besteht aus 5 verschiedenen Oberflächen (XHTML-Seiten):

- Login-Seite
- Admin - Verwaltungsseite
- Quest - Verwaltungsseite
- Unternehmens - Verwaltungsseite
- App-Benutzer – Verwaltungsseite

Die zu verwaltenden Daten sind auf jeder Seite in Form einer Datentabelle dargestellt.

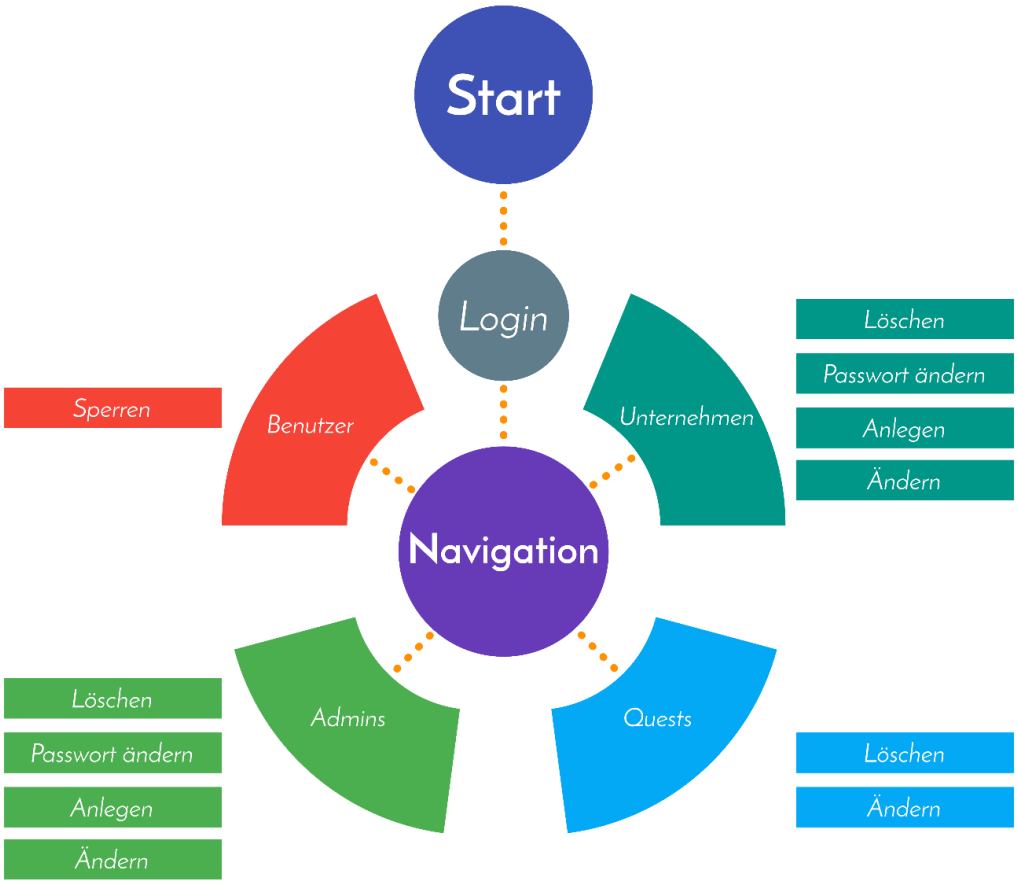


Abbildung 99 - Komponenten- und Funktionen der Administrationsoberfläche

16.3.2. Komponenten

Alle Komponenten sind Teil der Verwaltungsseiten und werden genutzt um die gewünschten Funktionen zu implementieren.

16.3.2.1. Navigation-Panel

Das Navigation-Panel ermöglicht es zwischen den Verwaltungsseiten zu wechseln und enthält das AdQuest-Logo. Das Panel ist auf der linken Seite zu finden (siehe Abbildung 101).

16.3.2.2. Datentabelle

Die Datentabelle ist der Hauptbestand der Verwaltungsoberfläche. Sie enthält die Datenanzeige (Spalten sind sortierbar), eine Suchleiste und verschiede Buttons um die Funktionen aufzurufen (siehe Abbildung 101). Die technische Grundlage für die Tabelle ist die PrimeFaces-Tabellen-Komponente.

16.3.2.3. Bearbeitungsdialog

Beim Aufrufen von Funktionen die eine Dateneingabe benötigen wird ein Dialog, mit den entsprechenden Feldern aufgerufen.

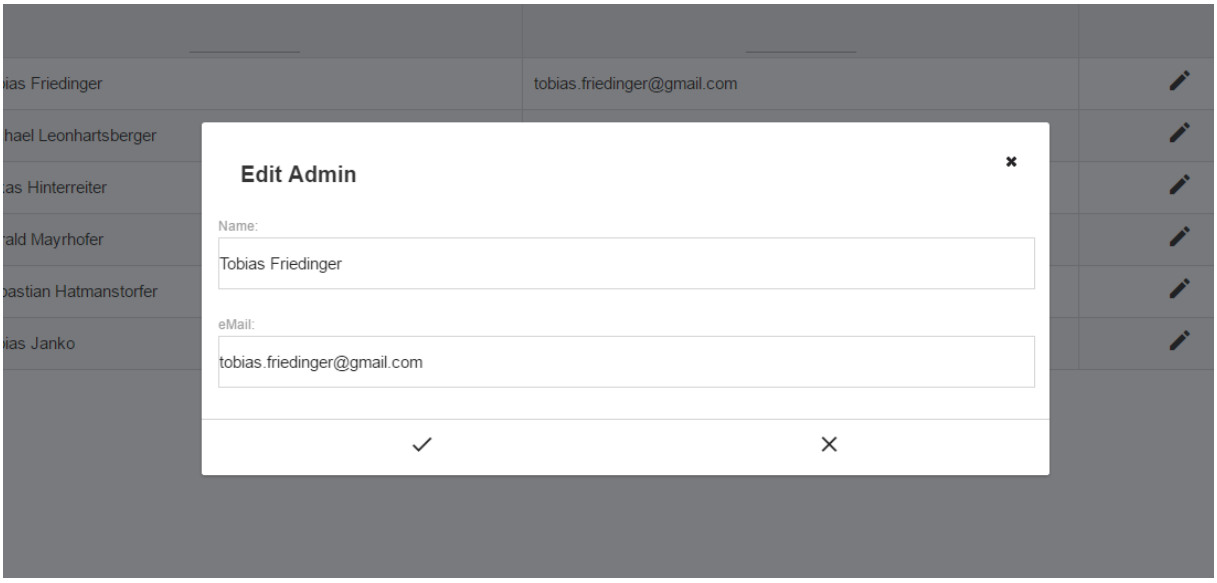


Abbildung 100 - Bearbeitungsdialog

16.3.2.4. Benachrichtigungs-Panel

Um den Benutzer eine Rückmeldung zu geben ob durchgeführte Operationen, erfolgreich abgelaufen sind, gibt es unten auf der Seite eine Anzeige.

Abbildung 101 - Benachrichtigungs - Panel

16.3.3. Funktionen

Jede dieser Seiten enthält verschiedene Verwaltungsfunktionen (Edit-Spalte in der Datentabelle), welche alle wesentlichen Punkte, die während des operativen Betriebs von AdQuest anfallen könnten, abdecken. Vereinheitlicht sind die Sortierung und Suchfunktionen, welche auf jeder Verwaltungsseite vorhanden sind und deshalb in der nachfolgenden Grafik ausgespart wurden.

16.3.3.1. Umsetzung der Funktionen

Die Umsetzung der Funktionalität erfolgt in Kombination von den XHTML-Seiten mit den Servlets (Java-Klassen).

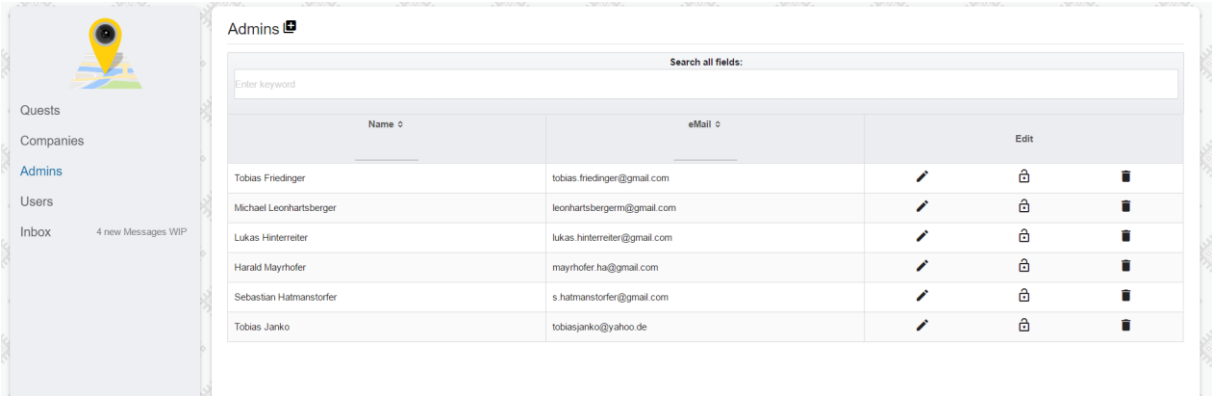


Abbildung 102 - Ausschnitt aus der Admin-Verwaltungsseite

Über die Seite werden Funktionen der Servlets aufgerufen, welche die Daten, direkt in der Datenbank verändern. Die Daten der Datenbank werden mittel Objekt-Relationalem-Mapping in den Arbeitsspeicher geladen, verändert und nach Beendigung aller Operationen wieder in der Datenbank gespeichert. Die vier Servlets enthalten die Methoden:

- add()
- remove()
- changePW()
- edit()

Es enthalten nicht alle Servlets alle Funktionen, sondern nur die für sie relevanten.

Beim Aufrufen einer Funktion öffnet sich ein Bearbeitungsdialog. Im unteren Bereich werden Benachrichtigungen angezeigt. Diese melden ob der Bearbeitungsvorgang erfolgreich war oder nicht.

Zusätzlich gibt es noch die Login-Funktion (auf der Login-Seite), welche es den Admins ermöglicht sich einzuloggen. Generell ist es erst möglich, die anderen Seiten aufzurufen, nachdem man sich eingeloggt hat. Ansonsten wird man auf die Login-Seite weitergeleitet.

16.4. Design

Das Design- und Layout wurde selbständig in CSS entwickelt. Als Basis-CSS-Bibliothek wurde Materialize-CSS verwendet. Bis am Ende der Entwicklung ist jedoch so gut nichts davon übriggeblieben. Die CSS-Klassen wurden überschrieben, um in das generelle Design zu passen. Die PrimeFaces Komponenten haben eigene CSS-Klassen und werden mit einem Standard-Theme ausgeliefert. Doch auch hier wurde der Großteil der CSS-Klassen überschrieben.

16.4.1. Transitions

Wenn eine andere Seite aufgerufen wird, wird eine Animation abgespielt, um Übergänge zu verschönern. Dies wurde mit einer CSS-Animation und JS realisiert.

17. Drohnensteuerung

17.1. Grundidee

Die automatisierte AdQuest-Drohnensteuerung wurde in Form einer Android Application implementiert. Sie stellt ein zusätzliches Angebot für die Unternehmenskunden dar, indem sie bei größeren Events gebucht werden kann. Bei diesen Events wird die Suchaktion von unserer Drohne aufgezeichnet, und dadurch erhält der Unternehmenskunde Rohmaterial für ein Werbevideo.
Drohnenbuchungen sind nur bei Events möglich, da aus Sicherheitsgründen immer ein AdQuest-Mitarbeiter vor Ort sein muss. Dieser Mitarbeiter erstellt aufgrund der jeweiligen Lokalitäten einen individuellen Flugplan für die Drohne, welcher nach einer Benachrichtigung durch den Server automatisch abgeflogen wird. Diese Benachrichtigung erfolgt, wenn ein Konsument den Suchradius des Cache betritt. Dieser Cache muss einer aktivierten Quest zugeordnet werden können.
Um die Sicherheit der Konsumenten, welche an der Suche teilnehmen, gewährleisten zu können, ist es dem anwesenden AdQuest-Mitarbeiter dauerhaft möglich, die Steuerung der Drohne manuell zu übernehmen. So kann dieser Mitarbeiter schnellstmöglich auf Gefahrensituationen, technische Störungen oder sonstige Probleme reagieren.

17.2. Technische Daten der Drohne

- Wi-Fi: 802.11a/b/g/n/ac
- Signalstärke: bis zu 250 Meter
- Höchstgeschwindigkeit: 13 m/s
- Sensor: CMOS 14Mpx
- Video Auflösung: 1920 x 1080p (30fps)
- Foto Auflösung: 4096 x 3072 px
- Interner Speicher: 8 GB Flash
- Maße: 28 x 32 x 3.6 cm
- Masse: 400 g
- Kompatibilität: iOS, Android Smartphones und Tablets, Windows Phone

(vgl. Parrot, 2017)

17.2.1. Einschränkungen

17.2.1.1. Wifi Anbindung

Für eine Steuerung der Drohne ist eine Verbindung mit dem drohneneigenen Wifi zwingend notwendig. Dadurch gibt es bei der Ausführung eines Flugplanes keine Möglichkeit einer Kommunikation mit dem Server. Dieses Problem wird im AdQuest-System mithilfe einer Firebase-Notification und einer automatisierten Funktion, welche eine Verbindung zur Drohne herstellt, umgangen.

17.2.1.2. Wetter

Durch das geringe Eigengewicht der Bebop Drohne ist diese sehr windanfällig. Zusätzlich kann die Drohne nicht bei Schneefall, Hagel oder Regen eingesetzt werden. Dadurch ergibt sich nur eine begrenzte Einsetzbarkeit. Die vom Wind verursachten Störungen müssen entweder in den Flugplan grob eingeplant werden, oder die Steuerung muss bei Gefahrensituationen manuell übernommen werden.

17.2.1.3. Technische Schwachstellen

Die Bebop Drohne ist aufgrund einiger technischer Schwachstellen für den Echtbetrieb ungeeignet. So schränkt das Wifi-Signal, welches zwingend für die Steuerung der Drohne benötigt wird, den Operationsbereich erheblich ein. Zusätzlich beträgt die Akkulaufzeit je nach Flugmanöver nur 10-15 Minuten. Sie ist daher bei einer längeren Suchaktion nicht ausreichend. Durch das geringe Gewicht der Bebop Drohne ist diese sehr windanfällig. Es ergeben sich daher die im Punkt „17.2.1.2 Wetter“ dargestellten Einschränkungen.

17.2.2. Rechtliche Situation

Die verwendete Bebop Drohne wird als Flugmodell klassifiziert und ist laut der Bestimmung von Austro Control bewilligungsfrei. Die Verwendung bei einem Event bedarf jedoch einer Genehmigung der Austro Control und ist nur in Einzelfällen möglich. Für die unterschiedlichen Einsatzgebiete, abhängig von der Besiedelungsdichte, ergeben sich je nach Gewicht der Drohne unterschiedliche Genehmigungsvorschriften.

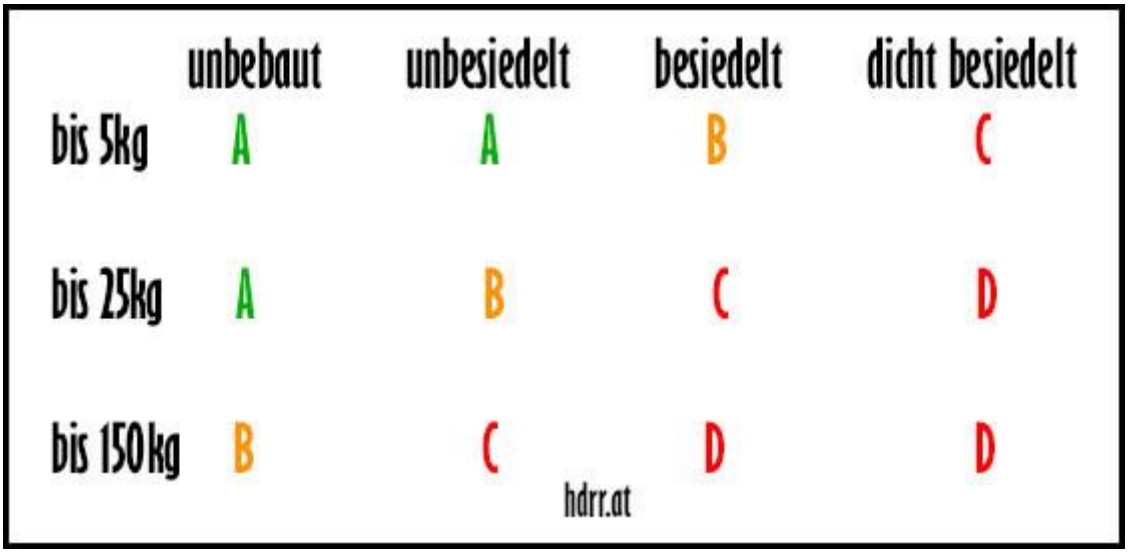


Abbildung 103 – Genehmigungskategorien für Drohnen

In den Kategorien A und B kann ohne Pilotenschein mit der Drohne operiert werden, für die Kategorien C und D wird ein Pilotenschein benötigt.
(vgl. Haderer, 2017)
Bei der Verwendung der AdQuest-Drohnensteuerung kann ohne weitere Berechtigung in unbebauten bis besiedelten Gebieten geflogen werden. Für Einsätze bei Events werden jedoch von der Austro Control zusätzliche Genehmigungen benötigt.

17.3. Ablauf der Drohnensteuerung

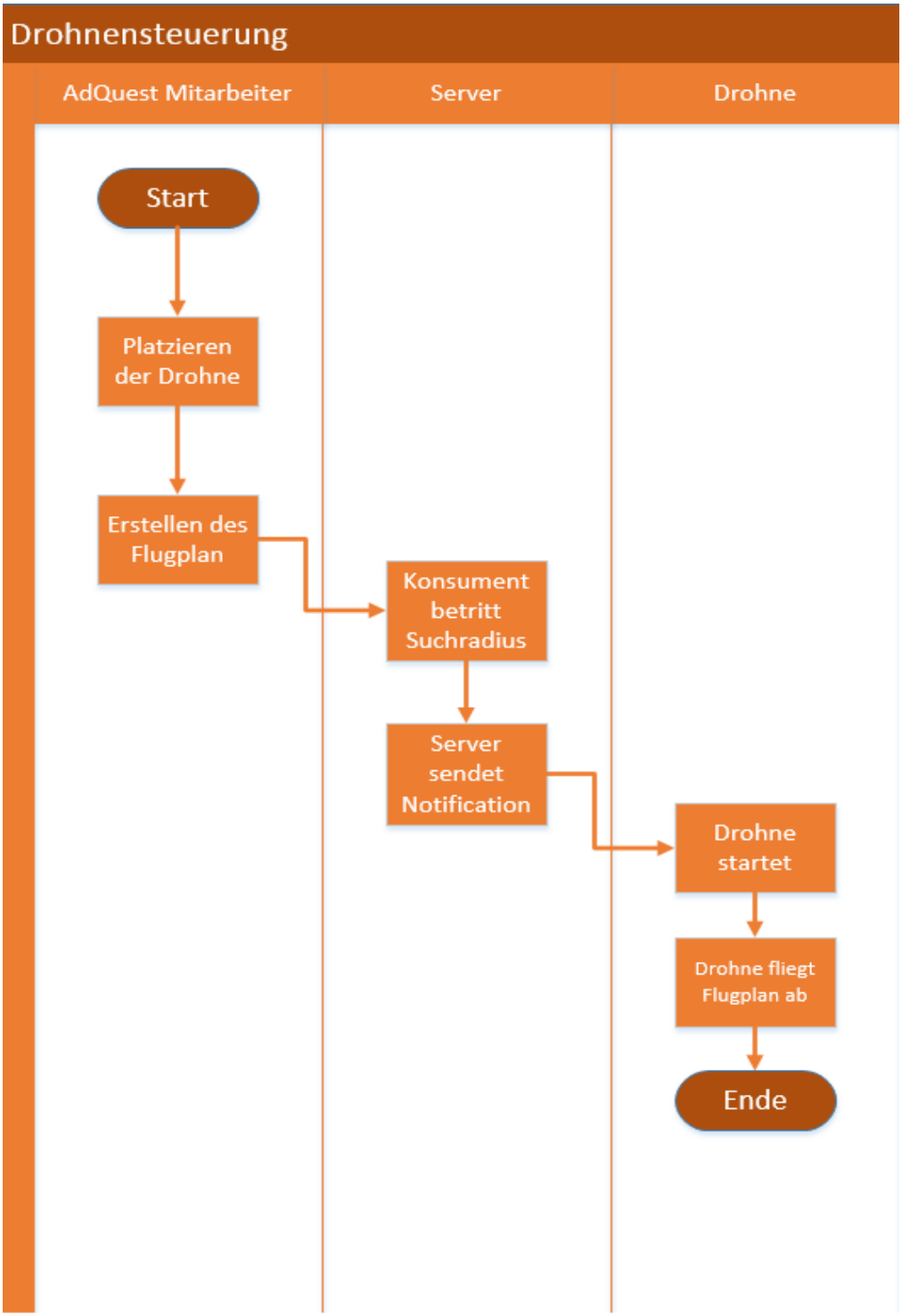


Abbildung 104 – Ablauf Drohnensteuerung

Ein AdQuest-Mitarbeiter muss bei einer Drohnenbuchung durch den Auftraggeber die Drohne an den vorhergesehenen Startpunkt platzieren, und den Flugplan je nach Örtlichkeit und Hindernissen etc. erstellen. Während der ganzen Ausführung des Flugplanes muss ein AdQuest-Mitarbeiter anwesend sein, um in Notfällen über eine manuelle Steuerung eingreifen zu können.

17.4. Verwendete Techniken

17.4.1. ArDroneSDK3

Als Basis für die automatische Steuerung wird das von Bebop zur Verfügung gestellte ARDroneSDK-Sample verwendet. Die Struktur des davon verwendeten Teiles wird in der nachfolgenden Abbildung dargestellt.

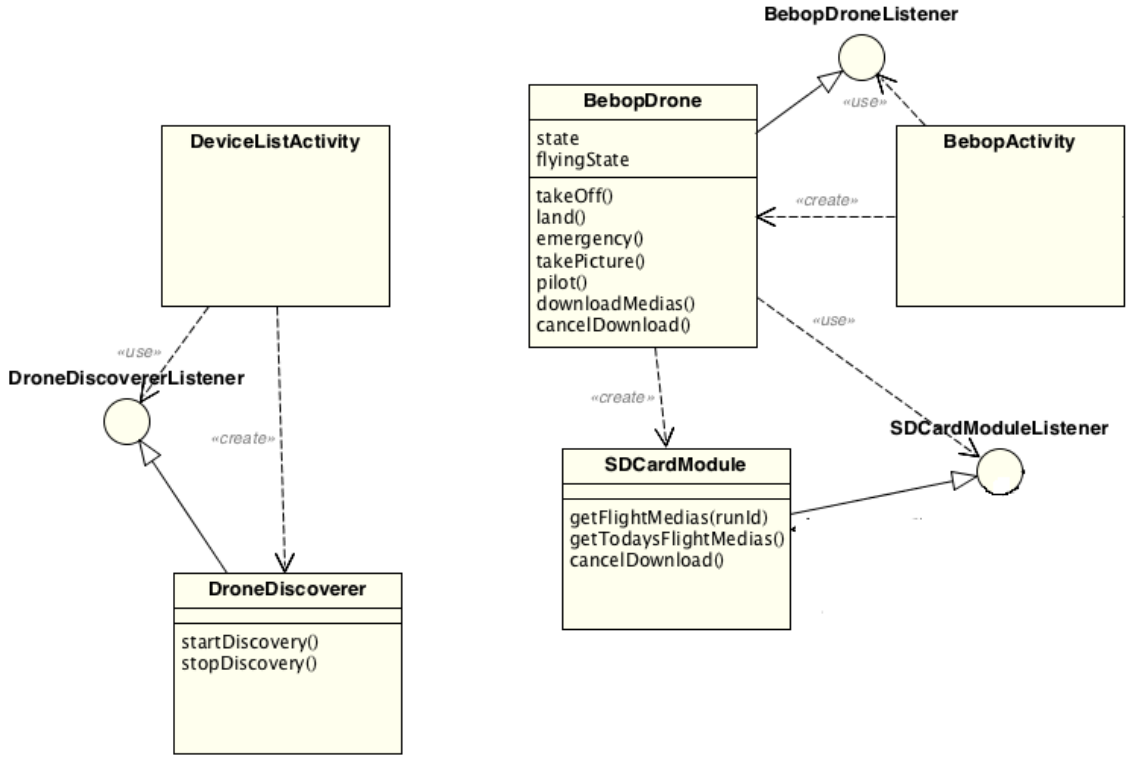


Abbildung 105 - Bebop-SDKSample Modell (Bebop, 2017)

Dieses Modell stellt grundlegende Steuerungsmöglichkeiten der Drohne sowie einige weitere Funktionalitäten wie zum Beispiel die Flugmanöver „land“ und „takeOff“ zur Verfügung. Dieses Modell wurde nun im Rahmen der Diplomarbeit nicht nur erweitert, sondern auch angepasst, um den Anforderungen einer vollautomatischen Steuerung entsprechen zu können.
Dieses von Bebop bereitgestellte Beispielprojekt baut auf der ARSDK Bibliothek auf.

```
ARSDK.loadSDKLibs();
```

Abbildung 106 – Laden der ARSDK-Bibliothek

Durch den in der obigen Abbildung dargestellten Befehl werden alle benötigten Bibliotheken geladen, die für die Drohnensteuerung benötigt werden.

Zusätzlich müssen die benötigten Bibliotheken auch, wie in der nachfolgenden Abbildung dargestellt, im „build.gradle“-File hinzugefügt werden.

```
compile project(':arsdk')
compile 'com.android.support:appcompat-v7:24.2.0'
compile 'org.apache.commons:commons-collections4:4.1'
compile 'com.android.support:multidex:1.0.0'
compile 'com.google.firebase:firebase-messaging:10.0.1'
testCompile 'junit:junit:4.12'
```

Abbildung 107 – Build.gradle Bibliotheken einbinden

17.5. Activities

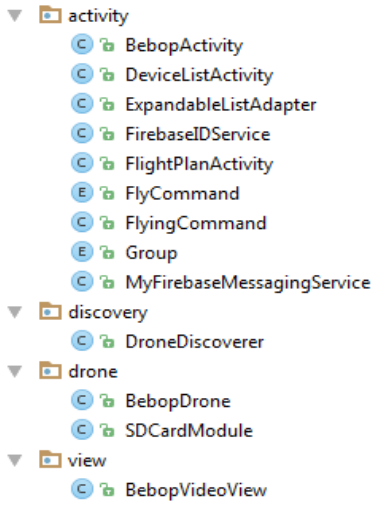


Abbildung 108 – Aufbau Aktivitäten

Der Aufbau und Lebenszyklus einer Activity wurde bereits in Punkt 16.2.1 erläutert, dieser wird auch in diesem Fall übernommen und verwendet.

In der Grafik links ist der grundlegende Aufbau der Activities der Drohnensteuerung dargestellt.

17.5.1. DeviceListActivity

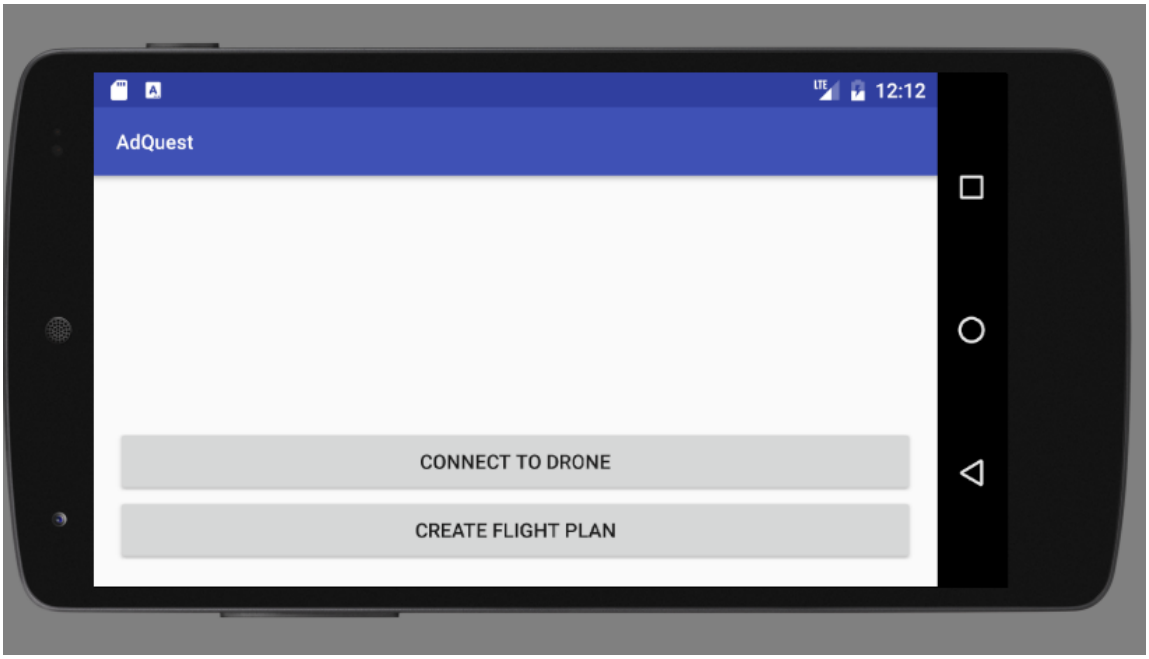


Abbildung 109 – DeviceListActivity

In der DeviceListActivity werden in einer ListView alle verfügbaren Drohnen im aktuellen Netz angezeigt und zu diesen kann man sich per Knopfdruck auf den Listeneintrag verbinden. Wird so, wie in der oben gezeigten Abbildung, keine verfügbare Drohne gefunden, so können mit dem „Connect to Drone“ – Button die verfügbaren Wifi-Verbindungen nach einem Drohnen-Wifi durchsucht und automatisch zu diesem verbunden werden. Um einen Flugplan zu erstellen, muss auf den „Create Flight Plan“- Button gedrückt werden und dadurch wechselt die Ansicht auf die GUI der FlightPlanActivity.

```
connectWifiButton.setOnClickListener((v) -> { connectToWifi(); });
```

Abbildung 110 – Wifi – OnClickListener

In der oberen Abbildung wird der Funktionsaufruf bei einem Button-Click dargestellt.

17.5.1.1. Verbinden zum Drohnen-Wifi

Nach dem Drücken des „Connect to Drone”-Buttons wird ein verfügbares Netzwerk einer Bebop Drohne mithilfe eines Wifi Managers gesucht. Falls ein Bebop-Netzwerk vorhanden ist, wird automatisch eine Verbindung zu diesem hergestellt.

```
WifiConfiguration conf = new WifiConfiguration();
conf.SSID = "\"" + networkSSID + "\"";
conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);

int netId = wifi.addNetwork(conf);

wifi.disconnect();
wifi.enableNetwork(netId, true);
wifi.reconnect();
```

Abbildung 111 – Verbinden zum Drohnen-Wifi

In der *connectToWifi* Methode werden mittels eines Android WiFi Managers die verfügbaren Wifis gescannt und auf eine Bebop-Kennung durchsucht. Beim Auffinden eines Bebop Drohnen Wifis wird die bestehende Wifi Verbindung getrennt und mit dem neuen Netzwerk verbunden. Dafür müssen in Android die Berechtigungen im AndroidManifest.xml festgelegt und von einem späteren User akzeptiert werden.

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
```

Abbildung 112 – Android.manifest User-Permissions

17.5.1.2. Verbinden zur Drohne

Die Verbindung mit der Bebop Drohne erfolgt automatisch nach dem erstmaligen Erkennen der Drohne. Dieser Vorgang erfolgt durch den Aufruf der Methode *connectToDrone* im SDK-Sample bereitgestellten DroneDiscoverer-Listener.

```
public void onDronesListUpdated(List<ARDiscoveryDeviceService> dronesList) {
    mDronesList.clear();
    mDronesList.addAll(dronesList);
    mAdapter.notifyDataSetChanged();
    if(mAdapter.getCount()>=1){
        connectToDrone();
    }
}
```

Abbildung 113 – Aufruf der connectToDrone Methode

In der *connectToDrone* Methode wird zur BebopActivity gewechselt und in dieser eine Verbindung zur Drohne hergestellt.

17.5.1.3. Kommunikation mit dem Server

Die Kommunikation mit dem Server erfolgt über eine Firebase Notification. Mit dem Server kann aufgrund der technischen Einschränkungen der Drohne nur vor dem Start des Flugplanes kommuniziert werden, da, wie schon im Punkt „18.2.1 technische Einschränkungen Drohne“ erwähnt wurde, die Steuerung der Drohne nur mit aktiver Wifi Verbindung möglich ist.

```
MyFirebaseMessagingService service=new MyFirebaseMessagingService(){
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        connectToWifi();
        super.onMessageReceived(remoteMessage);
    }
};
```

Abbildung 114 – Firebase onMessageRecieved

Beim Eintritt des ersten Teilnehmers in den Suchradius wird die Notification vom Server versendet. Daraufhin wird die oben schon beschriebene Methode *connectToWifi* aufgerufen.
Hierzu müssen im Android.manifest die benötigten Services hinzugefügt werden.

```
<service android:name=".activity.MyFirebaseMessagingService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>
<service
    android:name=".activity.FirebaseIDService">
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>
```

Abbildung 115 – Android.manifest Firebase Services

17.5.2. BebopActivity

Abbildung 116 – BebopActivity

Die BebopActivity stellt eine manuelle Steuerung bereit, die bereits im SDK-Sample beinhaltet ist. Sie besteht aus den Steuerelementen und einer BebopVideoView, welche den Flugvorgang automatisch bei Flugstart aufzeichnet. Dieses Videomaterial wird auf dem internen Speicherchip gespeichert. Zusätzlich wird das Bild der Drohne in Echtzeit übertragen, um die korrekte Steuerung der eingesetzten Drohne zu ermöglichen.

17.5.2.1. Flugplan ausführen

Der Flugplan wird bei der Veränderung des Status der Drohne gestartet. Das Take-Off-Flugmanöver wird im Status „landed“ durchgeführt. Da dieses Flugmanöver auch das erste Element des Flugplanes ist, startet die for-Schleife bei Veränderung des Status auf „flying“ oder „hovering“ mit dem zweiten Element des Flugplanes.

```
case ARCOMMANDS_ARDRONE3_PILOTINGSTATE_FLYINGSTATECHANGED_STATE_FLYING:
case ARCOMMANDS_ARDRONE3_PILOTINGSTATE_FLYINGSTATECHANGED_STATE_HOVERING:

    ArrayList<FlyingCommand> commandList = FlightPlanActivity.flyingCommandList;

    if(firstTimeFlying) {

        for (int i = 1; i<commandList.size();i++){
            commandList.get(i).executeFlyCommand(mBebopDrone,commandList.get(i));
        }
        firstTimeFlying = false;
    }
```

Abbildung 117 – Ausführen des Flugplanes

Damit der Flugplan nur ein einziges Mal ausgeführt wird, muss eine boolesche Variable gesetzt werden, welche weitere Aufrufe verhindert. Nach dem Ausführen des Flugplanes wird nach einer kurzen Verzögerung das „land“-Flugmanöver ausgeführt und die Drohne landet.

```
ScheduledExecutorService scheduledExecutorService = Executors.newScheduledThreadPool(1);
ScheduledFuture scheduledFuture = scheduledExecutorService.schedule((Callable) () -> {
        FlyingCommand landCommand = new FlyingCommand(FlyCommand.LAND);
        landCommand.executeFlyCommand(mBebopDrone,landCommand);
        return null;
},3000, TimeUnit.MILLISECONDS);
```

Abbildung 118 – Landen der Drohne nach Ausführen des Flugplanes

17.5.3. FlightPlanActivity

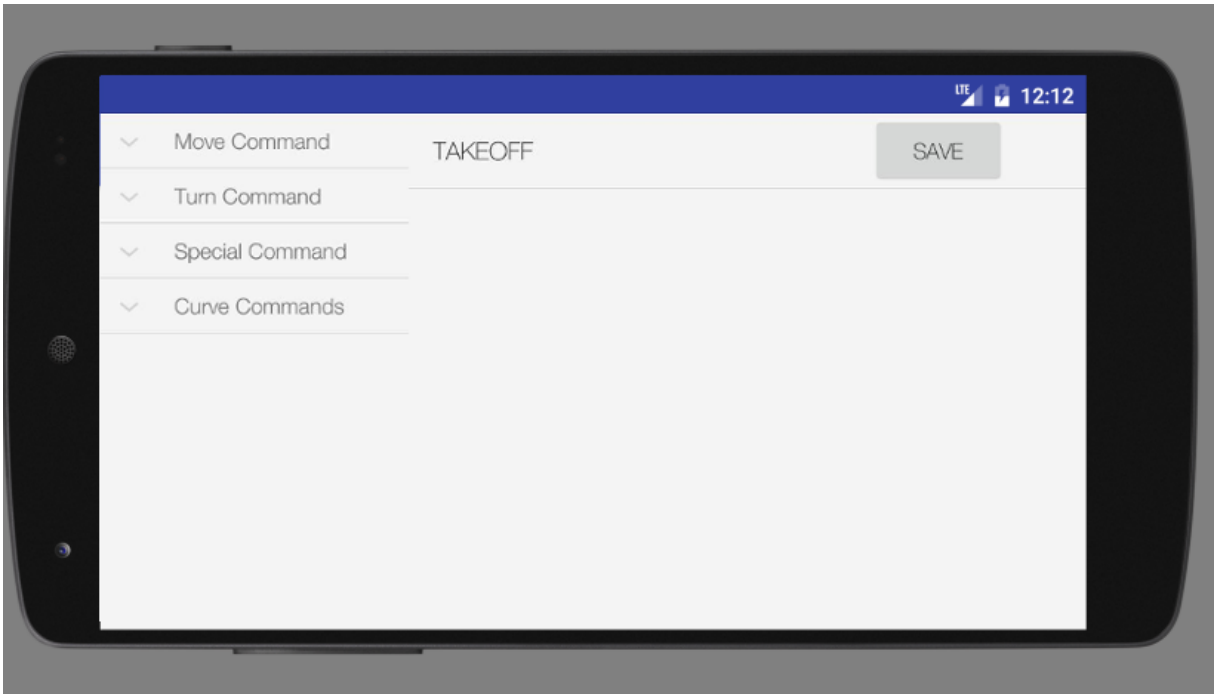


Abbildung 119 – FlightPlanActivity

In der FlightPlanActivity erstellt der AdQuest Mitarbeiter vor Ort einen Flugplan. Dazu wählt er die verschiedenen Flugmanöverkommandos aus und reiht diese in einer Liste aneinander. Der erstellte Flugplan kann mitthilfe des Save-Buttons gespeichert werden. Am linken Rand der View werden alle implementierten Flugmanöver in einer ExpandableListView abgebildet.

```
prepareListData();
expListAdapter = new ExpandableListAdapter(this, listDataHeader, listDataChild);
```

Abbildung 120 – Erstellen der Flugmanöverliste

Die ExpandableListView verwendet einen modifizierten Adapter und der Aufbau des Baumes erfolgt in der Methode *prepareListData*. Das Design der jeweiligen Elemente wird in den .xml Ressourcendateien festgelegt.
Standardmäßig enthält die Liste das „takeOff“-Flugmanöver, welches in jedem Flugplan als erstes Kommando ausgeführt werden muss. Am Ende eines jeden Flugplanes wird der Liste automatisch ein „land“-Flugkommando hinzugefügt.

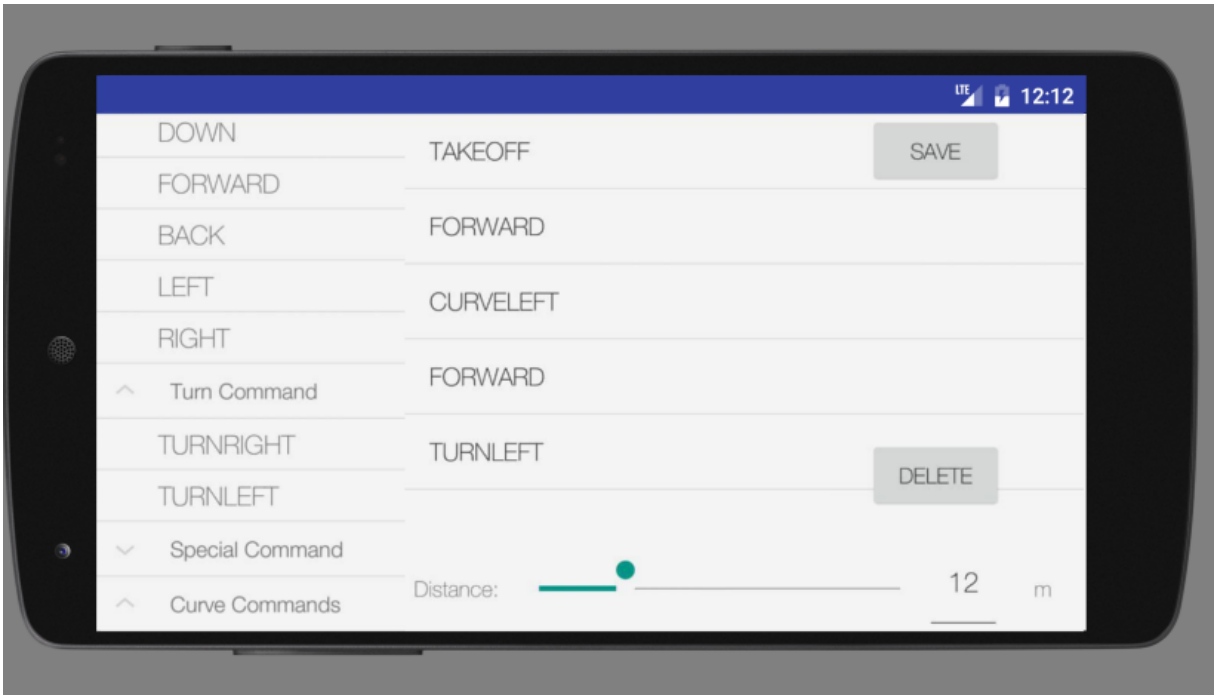


Abbildung 121 - Flugplan

Werden dem Flugplan Elemente hinzugefügt, dann werden diese auf der linken Hälfte in einer Standard-ListView angezeigt. Bei Klick auf eines dieser Elemente wird am unteren rechten Rand eine Detailansicht der Eigenschaften dieses ausgwählten Kommandos angezeigt. Wie in der oberen Abbildung dargestellt, kann hier beispielsweise die geflogene Distanz in Metern entweder mittels einer Android Seekbar oder einer Zahleneingabe angepasst werden.

17.5.3.1. Aufbau eines Flugmanöver

Ein ausführbares Flugmanöver besteht aus einem in einem Enum definierten Flugmanöver und dessen Distanz bzw. Rotationswert.

```
private FlyCommand flyCommand;
private int distance;
```

Abbildung 122 – Aufbau Flugmanöver

Ein Flugmanöver kann mit der Methode *executeFlyCommand* ausgeführt werden.

17.5.3.2. Aufbau eines Flugkommando

Alle verfügbaren Flugkommandos sind in einem Enum definiert, in dem jeweils eine Beschreibung und eine Gruppe mitgespeichert wird.

```
UP("UP",Group.MoveCommand)
```

Abbildung 123 – Aufbau Flugkommando

17.5.3.3. Aufbau einer Gruppe

Jedes Flugkommando wird einer Gruppe zugeteilt. Diese Gruppe beinhaltet eine Beschreibung, die auch für die Anzeige in dem GUI verwendet wird.

```
MoveCommand("Move Command")
```

Abbildung 124 – Beispiel Gruppierung

17.5.3.4. Distanzberechnung

Ausgangsbasis für die Distanz- und Rotationsberechnung sind Messergebnisse. Aufgrund dieser Ergebnisse wurden, mithilfe exponentieller Regression, Funktionen erstellt. Die Distanzberechnung ist nach den Gruppierungen gegliedert und einzelne Flugmanöver bedürfen meist einer eigenen Berechnung.

```
return (int) (1000 * (0.96 * Math.sqrt(2.08*flyingCommandDistance +0.48) - 1.202))
```

Abbildung 125 – Beispiel Distanzberechnung – Forward-Flugmanöver

Das Ergebnis einer solchen Berechnung ist die Zeit, angegeben in Millisekunden, welche die Drohne für die Ausführung des Flugmanövers für die gewünschte Distanz benötigt. Die Messergebnisse wurden für diesen Zweck in einer witterungsunabhängigen und störfaktorfreienTestumgebung ermittelt.

17.5.3.5. Rotationsberechnung

Die Rotationsberechnung erfolgt, wie auch die Distanzberechnung, aufgrund von Messwerten. Auf Basis dieser Messwerte wurden mittels linearer Regression die benötigten Funktionen ermittelt.

```
return (int) (1000 * (flyingCommandDistance+3.57)/48.15);
```

Abbildung 126 – Beispiel Rotationsberechnung

Das Ergebnis der Rotationsberechnung ist die Zeit in Millisekunden, welche für die Rotation der Drohne bis zu dem gewünschten Winkel benötigt wird.

17.5.3.6. ScheduledExecutorService

Das ScheduledExecutorService wird dazu benötigt, um die Flugmanöver nach der berechneten Zeit wieder zu stoppen. Dies wird mittels der *schedule*-Methode und einem neuen Callable mit der Methode *call* umgesetzt.

```
drone.setPitch((byte) -50);
drone.setFlag((byte) 1);
ScheduledFuture scheduledFuture = scheduledExecutorService.schedule(new Callable() {
    public Object call() throws Exception {
        drone.setPitch((byte) 0);
        drone.setFlag((byte) 0);
        return null;
    }
},duration, TimeUnit.MILLISECONDS);
```

Abbildung 127 – Beispiel ScheduledExecutorService

In der obigen Abbildung ist ein Beispiel der Verwendung des ScheduledExecutorService in der Ausführung des Flugmanövers „Back“ dargestellt.

17.5.3.7. GUI-Aufbau

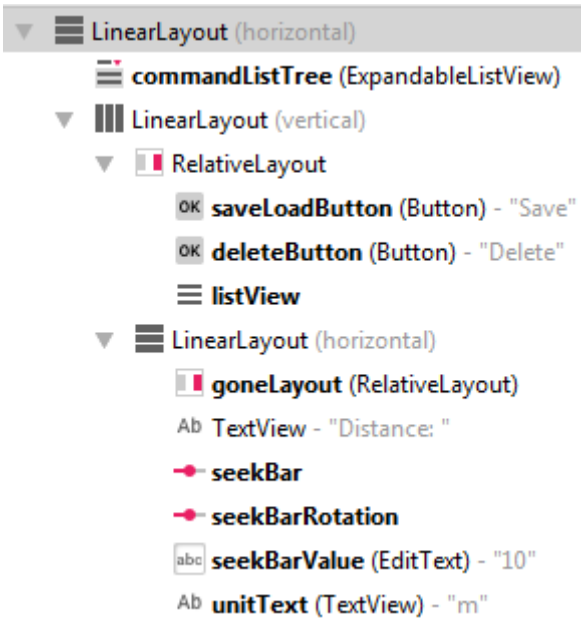


Abbildung 128 – Komponenten Flugplan-GUI

Die Trennung der Flugmanöverliste zur Flugplanliste erfolgt mittels eines horizontal ausgerichteten LinearLayouts. Im rechten Teil dieses LinearLayouts wird eine weitere Unterteilung unter Verwendung eines vertikalen LinearLayouts erzeugt. Im oberen Abschnitt dieser Teilansicht werden der Save- und Delete-Button, sowie eine ListView in einem RelativeLayout angeordnet. Im unteren Abschnitt werden die Details der jeweiligen Auswahl angezeigt. Um die Detailansicht bei einer Auswahl ohne Verzögerungen und ohne hohen Aufwand zu realisieren, wird ein leeres Layout verwendet. Dieses Layout wird entweder ein- oder ausgeblendet, je nach Auswahl. Benötigt eine Auswahl eine Detailansicht, so wird das Layout auf „GONE“ gesetzt. Wird keine Detailansicht benötigt, so wird das Layout auf „Visible“ gesetzt und blockiert jeglichen User-Input in diesem Abschnitt. Dieses Prinzip wird auch auf den Delete-Button angewendet, welcher auch je nach

Bedarf aus- oder eingeblendet wird. Zusätzlich werden zwei verschiedene Seek-Bars benötigt, da während der Laufzeit der Wertebereich einer solchen nicht verändert werden kann.

```
deleteButton.setVisibility(View.GONE);
goneLayout.setVisibility(View.VISIBLE);
```

Abbildung 129 – Beispiel Ein- und Ausblenden von Komponenten

In der oben dargestellten Abbildung wird ein Beispiel angeführt, in dem der Delete-Button, sowie die Detailansicht ausgeblendet werden. Dies erfolgt durch das Setzen des Delete-Button auf „Gone“ und das Setzen des Layouts auf Visible, welches dadurch die Detailansicht blockiert und somit ausblendet.

17.5.3.8. Speicherung des Flugplanes

Dem Benutzer, in diesem Fall dem AdQuest-Mitarbeiter, wird die Möglichkeit zur Verfügung gestellt, den Flugplan temporär zu speichern. Geschieht dies nicht, dann wird bei jedem neuen Aufruf der FlightPlanActivity der Flugplan zurückgesetzt.

```
if(savedFlyingCommandList.size()==0){
    ArrayList<FlyingCommand> flyingCommands = new ArrayList<>();
    flyingCommands.add(new FlyingCommand(FlyCommand.TAKEOFF));
    flyingCommandList = flyingCommands;
}
else{
    flyingCommandList = savedFlyingCommandList;
}
```

Abbildung 130 – Zurücksetzen des Flugplanes

In der oben dargestellten Abbildung erfolgt eine Überprüfung, ob der erstellte Flugplan gespeichert wurde. Wurde kein Flugplan gespeichert, so wird dieser auf den Ausgangszustand zurückgesetzt, andernfalls wird der gespeicherte Flugplan übernommen. Bei jedem neuen Aufruf der FlightPlanActivity erfolgt diese Überprüfung in der o*nResume* Methode.

18. Verteilung der Aufgabengebiete

Sebastian Hatmanstorfer

- Kurzbeschreibung
- Abstract
- Beteiligte
- Einleitung
- Allgemeines
- Technische Grundlagen
- Vorgehensmodell
- Website

Tobias Friedinger:

- Kurzbeschreibung
- Abstract
- Geschäftsmodell
- Administrationsoberfläche

Lukas Hinterreiter:

- Kurzbeschreibung
- Vorgehensmodell
- Android-Applikation

Michael Leonhartsberger:

- Kurzbeschreibung
- Allgemeines
- Technische Grundlagen

Harald Mayrhofer:

- Kurzbeschreibung
- Vorgehensmodell
- Drohnensteuerung

19. Literaturverzeichnis

19.1. Quellen

Affinity. (2017). Von https://affinity.serif.com/de/designer/ abgerufen

Allfacebook.de. (2017). Von http://allfacebook.de/zahlen_fakten/what-people-share abgerufen

Android. (2017). Von https://developer.android.com/images/activity_lifecycle.png abgerufen

Android. (2017). Von https://developer.android.com/guide/topics/ui/layout/linear.html abgerufen

Android.com. (2017). Von https://developer.android.com/images/activity_lifecycle.png abgerufen

Android.com. (2017). Von https://developer.android.com/about/dashboards/index.html abgerufen

Atlassian. (2017). Von https://www.sourcetreeapp.com/ abgerufen

Bebop, P. (2017). Von http://developer.parrot.com/docs/bebop/ abgerufen

Futurebiz. (März 2017). Von http://www.futurebiz.de/artikel/deutschland-liegt-beim-teilen-von-inhalten-in-sozialen-netzwerken-weit-hinten/ abgerufen

Google. (13. März 2017). Von https://developer.android.com/studio/features.html abgerufen

Google. (2017). Von https://firebase.google.com/features/ abgerufen

Gründerservice. (15. 03 2017). Von https://www.gruenderservice.at/site/gruenderservice/planung/Gruendungskosten.html abgerufen

Habert, S. (2017). Von https://www.sebastien-habert.com/ abgerufen

Haderer, R. (2017). Von https://hdrr.at/drohnengesetz-in-oesterreich/ abgerufen

Helmbold, C. (29. März 2017). Von http://helmbold.de/artikel/passwoerter-sicher-speichern/ abgerufen

Helmich, M. (29. März 2017). Von https://www.mittwald.de/blog/webentwicklung-webdesign/webentwicklung/restful-webservices-1-was-ist-das-uberhaupt abgerufen

Hoogvliet, O. (29. März 2017). Von https://blog.codecentric.de/2016/11/json-web-token-jwt-im-detail/ abgerufen

infoq.com. (15. März 2017). Von https://www.infoq.com/articles/Angular2-TypeScript-High-Level-Overview abgerufen

itwissen.info. (15. März 2017). Von http://www.itwissen.info/Java-Java.html abgerufen

itwissen.info. (2017). Von http://www.itwissen.info/JPA-Java-persistence-API.html abgerufen

jaxenter.de. (26. März 2017). Von https://jaxenter.de/resteasy-2-mit-cdi-support-2-8129 abgerufen

Jetbrains. (2017). Von https://confluence.jetbrains.com/display/WI/WebStorm+EAP abgerufen

Jetbrains. (15. März 2017). Von https://www.jetbrains.com/webstorm/features/ abgerufen

Jetbrains. (15. März 2017). Von https://www.jetbrains.com/idea/features/ abgerufen

JWT. (29. März 2017). Von https://jwt.io/introduction/ abgerufen

kreft.at. (2017). Von http://www.kreft.at/ abgerufen

logos.wikia.com. (2017). Von http://logos.wikia.com/wiki/File:Java-logo.jpg abgerufen

Lohn-Info.de. (22. März 2017). Von http://www.lohn-info.de/gehaltsverzicht.html abgerufen

Materializecss.com. (29. März 2017). Von http://materializecss.com/about.html abgerufen

mysql.de. (15. März 2017). Von https://www.mysql.de/products/workbench/ abgerufen

Parrot. (2017). Von https://www.parrot.com/us/drones/parrot-bebop-drone#technicals abgerufen

Pinterest.com. (2017). Von https://www.pinterest.com/pin/439171401143602985/ abgerufen

Postman. (2017). Von https://www.getpostman.com/ abgerufen

quintetsolutions.com. (2017). Von http://quintetsolutions.com/services/mysql-workbench-consultation/ abgerufen

Seeklogo. (2017). Von http://seeklogo.com/free-vector-logos/eclipse abgerufen

Slideshare. (22. März 2017). Von https://www.slideshare.net/ManfredSteyer/angular-2-slides abgerufen

Tilkov, S. (29. März 2017). Von https://jaxenter.de/rest-der-bessere-web-service-8988 abgerufen

tutorialspoint.com. (15. März 2017). Von https://www.tutorialspoint.com/jsf/jsf_overview.htm abgerufen

Twitter. (2017). Von https://twitter.com/intellijidea abgerufen

Wikipedia. (15. März 2017). Von https://de.wikipedia.org/wiki/Bitbucket abgerufen

Wikipedia. (15. März 2017). Von https://de.wikipedia.org/wiki/Eclipse_(IDE) abgerufen

Wikipedia. (15. März 2017). Von https://en.wikipedia.org/wiki/Firebase abgerufen

Wikipedia. (25. März 2017). Von https://de.wikipedia.org/wiki/Material_Design abgerufen

Wikipedia. (15. März 2017). Von https://de.wikipedia.org/wiki/TypeScript abgerufen

Wikipedia. (29. März 2017). Von https://de.wikipedia.org/wiki/Node.js abgerufen

Wikipedia. (26. März 2017). Von https://de.wikipedia.org/wiki/MySQL abgerufen

Wikipedia. (29. März 2017). Von https://de.wikipedia.org/wiki/Dependency_Injection abgerufen

Wikipedia. (1. April 2017). Von https://de.wikipedia.org/wiki/Geocaching abgerufen

Wikipedia. (1. April 2017). Von https://de.wikipedia.org/wiki/Framework abgerufen

Wikipedia. (1. April 2017). Von https://de.wikipedia.org/wiki/Client-Server-Modell abgerufen

Wikipedia. (1. April 2017). Von https://de.wikipedia.org/wiki/Git abgerufen

Wikipedia. (1. April 2017). Von https://de.wikipedia.org/wiki/Objekt_(Programmierung) abgerufen

Wikipedia. (1. April 2017). Von https://de.wikipedia.org/wiki/Tag_(Informatik) abgerufen

Wikipedia. (1. April 2017). Von https://de.wikipedia.org/wiki/Web_Storage#localStorage abgerufen

Wikipedia. (1. April 2017). Von https://de.wikipedia.org/wiki/Programmbibliothek abgerufen

Wikipedia. (2017). Von https://de.wikipedia.org/wiki/Aufz%C3%A4hlungstyp abgerufen

Wikipedia. (2017). Von https://de.wikipedia.org/wiki/Aufz%C3%A4hlungstyp abgerufen

Wikipedia. (2017). Von https://de.wikipedia.org/wiki/Software_Development_Kit abgerufen

Wikipedia. (2017). Von https://de.wikipedia.org/wiki/Parameter_(Informatik) abgerufen

Wikipedia. (2. April 2017). Von https://de.wikipedia.org/wiki/Server abgerufen

Wikipedia. (2. April 2017). Von https://de.wikipedia.org/wiki/W%C3%B6rterbuchangriff abgerufen

Wikipedia. (2. April 2017). Von https://de.wikipedia.org/wiki/Rainbow_Table abgerufen

Wikipedia. (2. April 2017). Von Zeichenfolge abgerufen

Wikipedia. (2. April 2017). Von https://de.wikipedia.org/wiki/Uniform_Resource_Identifier abgerufen

Wikipedia. (2017). *Wikipedia*. Von https://de.wikipedia.org/wiki/Software_Development_Kit abgerufen

Wikipedia. (2017). *Wikipedia*. Von https://de.wikipedia.org/wiki/Software_Development_Kit abgerufen

Wkipedia. (2. April 2017). Von https://de.wikipedia.org/wiki/Client abgerufen

wko.at. (2017). Von https://www.wko.at/service/zahlen-daten-fakten/KMU-definition.html abgerufen

19.2. Abbildungsverzeichnis

19.3. Tabellenverzeichnis

20. Glossar

Activity

ist eine Bildschirmseite in einer Applikation

Authentifizierung

anmelden mit Username und Passwort beim Server. Der Server ermittelt, ob eine Person tatsächlich der ist, wer er vorgibt zu sein

Android

Betriebssystem von Google vor allem für mobile Geräte

API-Level

Versionsnummer, welche Programmierschnittstelle von einem Geräte unterstützt wird

Backend

Teil eines Systems, welche sich mit der Datenverarbeitung beschäftigt. Meist wird die Anwendung, welche sich auf dem Server befindet, als Backend bezeichnet.

Bibliothek (Softwareentwicklung)

Sammlung von Unterprogrammen und Lösungswegen, für zusammengehörende Problemstellungen. Sie enthalten Hilfsmodule die von Programmen aufgerufen werden können. (vgl. Wikipedia, 2017)

Binding

Bezeichnet das Aneinanderbinden von Werten oder Methoden bzw. Events.

Client

Software, welche mit einem Server kommuniziert. (vgl. Wkipedia, 2017)

code-completition / code-inspection

Bezeichnen Hilfen von Entwicklungsumgebungen, die dem Programmierer das Entwickeln erleichtern. Completition beschreibt das erkennen und automatische Vervollständigen des Codes, Inspection die dynamische Fehlerfindung.

Cardviews

sind abgetrennte Bereiche innerhalb einer Activity, welche sich wie eine Liste anordnen lassen

Datenbank

System zur Datenverwaltung

Enum

Ein Enum ist ein Aufzählungstyp. Er wird als Datentyp für Variablen mit endlichen Wertemengen verwendet. (vgl. Wikipedia, 2017)

Emulator

Dient zum Simulieren von Hardware wie zum Beispiel Handys.

Framework

Ein Framework ist mehr oder weniger ein Programmiergerüst, welches dem Programmierer Entwurfsmuster, Strukturen oder ähnliches zur Verfügung stellt. (Wikipedia, 2017)

Flugmanöver

Ausführbares Manöver welches ein Flugkommando sowie die Distanz/Rotation dieses beinhaltet.

Flugkommando

Beschreibt einen Ablauf der Drohne um eine gewünschte Veränderung dieser zu erreichen. (Bsp. Veränderung der Position oder Ausrichtung)

Flugplan

List von Flugmanöver welche der Reihe nach ausgeführt werden.

GeoCaching

GPS-Schnitzeljagt/Schatzsuche. Die Verstecke der Schätze werden mittels geographischer Koordinaten im Internet veröffentlicht. (vgl. Wikipedia, 2017)

Git-Repository

Git ist die Bezeichnung einer freuen Software zur Versionsverwaltung. Ein Repository ist ein Bereich in Git, in dem Daten für Projekte abgelegt werden können. (vgl. Wikipedia, 2017)

GUI

Graphical User Interface → Benutzeroberfläche

HTTP (Hypertext Transfer Protocol)

Protokoll zur Übertragung von Daten über ein Netzwerk.

Hashfunktion

Bildet eine beliebig lange Zeichenfolge auf eine Zeichenfolge mit fester Länge ab. (vgl. Wikipedia, 2017)

Hashwert

Ergebnis einer Hashfunktion

Hosten

Bereitstellen eines bestimmten Dienstes (im Internet).

JSON

JavaScript Object Notation ist eine kompakte Form um Daten über das Internet auszustauschen

JSON-Object Request

das Anfordern von Daten im JSON Format bei einem Server

Klassen / Objekte

Eine Klasse ist in der Softwareentwicklung ein abstrakter „Bauplan“ eines Realen Objektes. Ein Objekt wiederum ist ein Exemplar dieser Klasse, mit dem gearbeitet werden kann. (vgl. Wikipedia, 2017)

LinearLayout

Ansichtsgruppe in Android welche alle Kind-Elemente in einer einzigen Richtung entweder horizontal oder vertikal aneinanderreiht. (vgl. Android, 2017)

Lineare/exponentielle Regressions

Statistische Analyseverfahren um eine Funktion aus mehreren Werten zu ermitteln. Dabei wird die Funktion mit dem geringsten Fehler als Lösung verwendet.

ListView

ermöglicht die Anzeige von mehreren Elementen in Listenform

LocalStorage

Speicher Wertepaare im lokalen Speicher des Browsers. (vgl. Wikipedia, 2017)

Marker

Ist eine Art Pin-Nadel, welche auf einer Karte angebracht wird, um auch von weitem das Ziel zu sehen

Navigation Drawer

seitliche Leiste zur Navigation innerhalb einer Applikation

Observable (Angular2)

„Überwacher“. Wird beispielsweise bei Serveranfragen genutzt. Hat eine Methode, die automatisch aufgerufen wird, sobald das Observable, bei der Methode bei der es verwendet wird, eine Werteveränderung vernimmt.

Pager

Fasst in einer Activity mehrere Bildschirmseiten zusammen und ermöglicht das Bewegen durch diese Seiten

Parameter

sind in der Informatik Variablen, über die ein Computerprogramm oder Unterprogramm, für einen Aufruf gültig, auf bestimmte Werte „eingestellt“ werden kann. Diese Einstellungen werden bei der Verarbeitung berücksichtigt und beeinflussen damit meist auch die Ergebnisse des Programms. (vgl. Wikipedia, 2017)

QR-Code

Art Barcode, welcher es erlaubt Informationen zu speichern

Rainbow Table

Datenstruktur für eine schnelle Suche nach den Ursprungszeichen eines Hashwertes. (vgl. Wikipedia, 2017)

Request

Datenanfrage eines Clients an den Server (vgl. Wikipedia, 2017)

Response

Antwort des Servers auf eine Datenabfrage.

Salt

Zufällige Wert, der an das Passwort vor dem Hashverfahren angehängt wird.

ScheduledExecutorService

Stellt die Möglichkeit bereit Kommandos und Methoden verzögert aufzurufen. Wird genutzt um die Drohne nach Beendigung eines Flugmanövers wieder auf den Ausganszustand zurückzusetzten.

SDK

Ein Software Development Kit (SDK) ist eine Sammlung von Programmierwerkzeugen und Programmbibliotheken, die zur Entwicklung von Software dient. (Wikipedia, 2017)

SeekBar

GUI-Komponente von Android welche einen Schieberegler darstellt.

Server

Software oder Computer, welcher Dienste, Daten oder Ressourcen für Clients bereitstellt. (vgl. Wikipedia, 2017)

Tag (Webentwicklung)

Tags sind sogenannte Auszeichnungsmarkierungen, die den Beginn und das Ende verschiedener Elemente markieren. (vgl. Wikipedia, 2017)

Token

Wird vom Server ausgestellt und enthält Daten um einen Client zu identifizieren.

UUID (Universally Unique Identifier)

Eine 128-Bit Nummer, welche Daten eindeutig identifiziert.

URL (Uniform Resource Locator)

Adresse einer Website

URI (Uniform Resource Identifier)

Identifiziert eine Ressource (Website, Dateien…) (vgl. Wikipedia, 2017)

Wörterbuchangriff

Methode, um ein unbekanntes Passwort mithilfe einer Liste von Passwörtern festzustellen. (vgl. Wikipedia, 2017)