Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Master-Thesis

Konzeption einer Anwendung (App) für die Stadtwerke Karlsruhe GmbH zur Präsentation historischer Anlagen der Trinkwasserversorgung und Umsetzung anhand eines Prototyps

Betreuer:
Hochschule Karlsruhe: Prof. Dr.-Ing. Bernhard Bürg
Stadtwerke Karlsruhe GmbH: Prof. Dr. Matthias Maier
Ingenieurbüro Mach:Idee: Dipl. Ing. Rüdiger Mach

Bearbeitungszeitraum: 1. August 2014 bis 30. Januar 2015

Vorgelegt von: **Christoph Hofmann**
Matrikelnummer: 44130
Studiengang: Geomatik Master
Fakultät: Informationsmanagement und Medien
Hochschule Karlsruhe - Technik und Wirtschaft

Diese Abschlussarbeit wurde im Rahmen des Studiums „Geomatik Master“ an der Hochschule Karlsruhe erstellt.

Diese Arbeit wurde begleitet und unterstützt von:

Stadtwerke Karlsruhe GmbH
Hauptabteilung TT
Abteilung T-TW

Ingenieurbüro Mach:Idee

Referenten/Betreuer:

Hochschule Karlsruhe:	Prof. Dr.-Ing. Bernhard Bürg
Stadtwerke Karlsruhe GmbH:	Prof. Dr. Matthias Maier
	Dr. Bernd Hofmann
Ingenieurbüro Mach:Idee:	Dipl. Ing. Rüdiger Mach

Einige Hardware- und Softwarebezeichnungen sowie einige Angaben zu Produkten oder Firmen in dieser Arbeit, sind als eingetragene Marken geschützt. Da nicht sämtliche Angaben auf Markenschutz überprüft werden, kann ich in dieser Arbeit das ®-Symbol nicht verwenden. Sämtliche Angaben und Informationen stammen aus dem Bearbeitungszeitraum dieser Arbeit. Einige Angaben (vor allem Links) könne sich dadurch ändern oder geändert haben.

Aufgabenblatt für die **Master Thesis**

von

Christoph Hofmann

an der

HOCHSCHULE KARLSRUHE – TECHNIK UND WIRTSCHAFT
Fakultät für Informationsmanagement und Medien – Studiengang Geomatik

in Zusammenarbeit mit

Stadtwerke Karlsruhe GmbH

Ingenieurbüro Mach:Idee Rüdiger Mach

Europäischen Brunnengesellschaft e.V. Sektion Karlsruhe

Thema: Konzeption einer Anwendung (App) für die Stadtwerke Karlsruhe GmbH zur Präsentation historischer Anlagen der Trinkwasserversorgung und Umsetzung anhand eines Prototyps

Designing an application (app) for Stadtwerke Karlsruhe GmbH to present their historical systems of drinking water supply and implementation of the application using a prototype

---

Problemstellung

Die in der Vergangenheit eingesetzten Anlagen der Trinkwasserversorgung der Stadt Karlsruhe zeigen eindrücklich die Entwicklung der Wasserversorgung in Technik und Baukunst. Die Hauptabteilung TT pflegt die heute noch verfügbaren Daten und Informationen und stellt diese dem interessierten Bürger in Form von Veröffentlichungen zur Verfügung.

Ziel der Arbeit

In Ergänzung zu diesen Publikationswegen soll im Rahmen einer Masterarbeit die Möglichkeit und der Einsatz moderner Informationstechniken untersucht werden, mit denen ein mobiles Informationssystem in Form einer App, die auf Smartphones zur Verfügung gestellt werden kann, aufgebaut werden kann. In der Arbeit wird ein Konzept für eine Anwendung (App), auf Grundlage der vorhandenen Daten der Stadtwerke Karlsruhe GmbH und für ausgewählte Trinkwasserlaufbrunnen der Europäischen Brunnengesellschaft (EBG), entwickelt und im Rahmen eines Prototyps umgesetzt. Im wissenschaftlichen Fokus steht die Evaluierung geeigneter Software um das Projekt in einem angemessenen Zeitrahmen und kostengünstig zu realisieren.

Mit dem entwickelten System sollen interessierte Bürger den Weg zu den einzelnen historischen Anlagen der Trinkwasserversorgung in Karlsruhe sowie zu den Trinkwasserlaufbrunnen mit ihrem Smartphone finden können. An den jeweiligen Denkmälern werden Tafeln mit QR-Codes angebracht, die mit dem Smartphone gescannt werden können. Nach dem Erkennen des Codes wird die zum jeweiligen Objekt verfügbare Information angezeigt. Für die Entwicklung des Systems kann auf die umfangreichen Daten der Stadtwerke Karlsruhe und für die Trinkwasserlaufbrunnen auf die Daten der EBG zurückgegriffen werden. Weiterhin ist angedacht, die Informationen aus den historischen Dokumentationen der Karlsruher Wasserwerke zu integrieren. Damit werden dem Betrachter umfangreiche technische, künstlerische und kulturelle Informationen direkt an der historischen Anlage vermittelt.

Arbeitsschritte:

- Erstellung eines Anforderungskatalogs
- Recherche nach vergleichbaren Anwendungen
- Sichtung und Aufbereitung der Grundlagendaten
- Evaluierungen der verschiedenen Komponenten zur Entwicklung einer Anwendung (mobile Betriebssysteme, Art der Anwendung, Software, Einbindung Kartenkomponente, Einbindung QR-Code-Technologie)
- Erstellung eines Konzepts für die geplante Anwendung
- Technische Umsetzung des Konzepts anhand eines Prototyps
- Schriftliche Ausarbeitung

Rahmenbedingungen

Die Arbeit wird an der Fakultät für Informationsmanagement und Medien der Hochschule Karlsruhe-Technik und Wirtschaft in Zusammenarbeit mit der Stadtwerke Karlsruhe GmbH, dem Ingenieurbüro Mach:Idee Rüdiger Mach und der Europäischen Brunnengesellschaft e.V. Sektion Karlsruhe durchgeführt. Der Arbeitsplatz befindet sich im Ingenieurbüro Mach:Idee. Zur Umsetzung der Arbeit wird ein Notebook der G-Serie von emachines verwendet.

Zur Umsetzung des Prototyps und dessen Test kommt das Smartphone Nexus 4 der Google Inc. zum Einsatz.

Nähere Informationen zu den Hardwarekomponenten finden sich im Kapitel Hardware. Die Datengrundlage der Arbeit wird von den Stadtwerken Karlsruhe GmbH und der EBG bereitgestellt.

Alle theoretischen und konzeptionellen Überlegungen sowie die praktischen Arbeiten und deren Ergebnisse sind zu dokumentieren (schriftliche Ausarbeitung der Master Thesis). Darüber hinaus ist eine Präsentation für das abschließende Kolloquium zu erstellen.

Leiter der Master Thesis:	Prof. Dr. Bernhard Bürg
Zweiter Prüfer:	Prof. Dr. Matthias Maier
Externer Betreuer:	Ingenieurbüro Mach:Idee, Rüdiger Mach
Bearbeitungszeit:	6 Monate
Tag der Ausgabe:	01.08.2014
Tag der Abgabe:	30.01.2015
Anschrift des Kandidaten:	Christoph Hofmann Friedrichstraße 106 76287 Rheinstetten 0721/510491 hofmann.christoph@web.de

---

Datum | Leiter der Master Thesis | Zweiter Prüfer | Externer Betreuer

Erklärung

Hiermit versichere ich, die vorliegende Master-Thesis selbstständig verfasst zu haben. Es wurden keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet. Alle Quellen und Textstellen wurden als solche kenntlich gemacht.

Diese Erklärung erstreckt sich auch auf die in der Arbeit enthaltenen Graphiken, Zeichnungen, Kartenskizzen und bildliche Darstellungen.

Christoph Hofmann Rheinstetten, 30.01.2015

Danksagung

Hiermit möchte ich mich bei denjenigen bedanken, die diese Arbeit überhaupt erst möglich gemacht haben.

Ich bedanke mich bei den Stadtwerken Karlsruhe GmbH für die ich diese Arbeit anfertigen durfte. Vor allem bei Herrn Prof. Dr. Matthias Maier und Herrn Dr. Bernd Hofmann die diese Arbeit tatkräftig unterstützt haben.

Ich danke Herrn Prof. Dr.-Ing. Bernhard Bürg für die Unterstützung seitens der Hochschule Karlsruhe.

Ich danke Herrn Faulhaber von der Europäischen Brunnengesellschaft e.V..

Ein weiterer Dank geht an Herrn Dipl. Ing. Rüdiger Mach der diese Arbeit durch sein technisches Wissen begleitet hat. Danken möchte ich Herrn Mach vor allem dafür, dass er diese Arbeit durch sein Fachwissen qualitativ beeinflusst hat und bereit war, sein Wissen mit mir zu teilen. Herr Mach hat sich meinen Fragen und Problemen zu jeder Tageszeit stets offen gestellt und mir bei diesen weitergeholfen.

Danken möchte ich ebenfalls meiner Familie.

Hier gilt ein besonderer Dank meinen Eltern. Ohne ihre Hilfe und Unterstützung während meines kompletten Studiums wäre dies so niemals möglich gewesen und diese Arbeit wäre niemals verfasst worden. Ohne ihre Unterstützung wäre ich heute sicher nicht da, wo ich bin.

Gab es offene Fragen bezüglich der Karlsruher Trinkwasserversorgung oder bezüglich der Inhalte der Applikation war mein Vater jederzeit erreichbar und sofort zur Stelle. Dies führte zu einigen Terminen, welche sich zeitlich doch in die Länge zogen. Hierbei verlor er jedoch nie die Lust mir bei den verschiedenen Aufgaben zur Seite zu stehen.

Ein „kleines“ Dankeschön geht ebenfalls an Michelle. Michelles Aufgabe bestand darin, den Prototypen auf „Kindertauglichkeit“ zu testen. Dies tat sie mit sehr viel Freude, konstruktiver Kritik und wichtigen Anmerkungen.

Abschließend möchte ich mich bei Melody bedanken. Meine Freundin hatte auch während dieser Arbeit wieder einmal viel Geduld. Sie musste teilweise bei einigen Sachen zurückstecken oder auf unsere freie Zeit verzichten. Trotz allem stand Sie mir mit ihrem Verständnis hierfür immer zur Seite und hielt mir den Rücken frei.

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel wird auf die Veranlassung der Masterthesis eingegangen. Die Zielsetzungen werden beschrieben und die Vorgehensweise zur Bearbeitung erläutert.

1.1 Veranlassung

Die in der Vergangenheit eingesetzten Anlagen der Trinkwasserversorgung der Stadt Karlsruhe zeigen eindrücklich die Entwicklung der Wasserversorgung in Technik und Baukunst. Die Hauptabteilung Trinkwassergewinnung (TT) pflegt die heute noch verfügbaren Daten und Informationen und stellt diese dem interessierten Bürger in Form von Veröffentlichungen zur Verfügung.

In Ergänzung zu diesen Publikationswegen sollen dem Bürger nun auch moderne Wege zugänglich gemacht werden, um die Informationen einzusehen. Hierbei ist es wichtig die Informationen zielgerichtet aufzubereiten.

Durch den Einsatz von modernen Techniken soll es den Interessenten möglich sein, schneller und einfacher, an die wichtigen Informationen zu gelangen.

1.2 Zielsetzung

In der Master-Thesis wird ein Konzept für eine Anwendung (App)[1], auf Grundlage der vorhandenen Daten der Stadtwerke Karlsruhe GmbH und für ausgewählte Trinkwasserlaufbrunnen der europäischen Brunnengesellschaft (EBG), entwickelt und im Rahmen eines Prototyps umgesetzt.

Im wissenschaftlichen Fokus steht die Evaluierung geeigneter Software um das Projekt in einem angemessenen Zeitrahmen und kostengünstig zu realisieren.

Mit dem entwickelten System sollen interessierte Bürger beispielsweise den Weg zu den einzelnen historischen Anlagen der Trinkwasserversorgung in Karlsruhe sowie zu den Trinkwasserlaufbrunnen mit ihrem Smartphone finden können.

Durch die Zuhilfenahme der QR-Code-Technologie[2] soll es dem Anwender möglich sein, auf die zuvor aufbereiteten Daten der Stadtwerke Karlsruhe und für die Trinkwasserlaufbrunnen auf die Daten der EBG zuzugreifen.

Das erarbeitete Konzept soll anhand eines Prototypen der Anwendung umgesetzt werden. Die Anwendung soll kompatibel zu anderen Systemen sein. Die einfache Bedienung und die schnelle Informationsübermittlung stehen im Vordergrund.

[1] In der weiteren Arbeit werden die Begriffe Anwendung, Applikation und App verwendet.
[2] „QR“ steht für Quick Response

Die Anwendung soll kostengünstig und zeitnah realisierbar sein. Da die QR-Code-Technologie, sowie Kartendienste eingesetzt werden sollen, ist es sinnvoll, dass die Anwendung auf die bestehenden API[1] des mobilen Betriebssystems zugreifen kann.

1.3 Aufbau der Arbeit

Dieses Kapitel wird eine kurze Übersicht über die schriftliche Ausarbeitung gegeben.

Zu Beginn der schriftlichen Ausarbeitung stehen einleitende Worte welche auf die Veranlassung und die Zielsetzung der Arbeit eingehen. Anschließend werden vergleichbare Anwendungen analysiert, um den „Stand der Technik“ aufzugreifen. Im dritten Teil der schriftlichen Ausarbeitung werden die Grundlagen vorgestellt, welche für die Arbeit von Bedeutung sind. Grundlegende Begriffe und Technologien werden erklärt und technische Grundlagen erläutert. Im darauffolgenden Kapitel wird auf die Möglichkeiten und Varianten der App-Entwicklung eingegangen. Kapitel 5 und 6 gehen auf den technischen Teil der Arbeit ein. Der technische Teil enthält die Konzepterstellung sowie die technische Umsetzung des Prototypen. Am Ende der schriftlichen Ausarbeitung steht die Bewertung der Ergebnisse, welche eine Zusammenfassung und Analyse des Prototypen enthält. Im Kapitel Zusammenfassung und Ausblick wird die gesamte Arbeit noch einmal kurz beschrieben und die maßgebenden Ergebnisse zusammengefasst. Außerdem wird ein Ausblick gegeben. Dieser Ausblick geht auf zukünftige Verbesserungen bzw. Fortschreibung sowie auf zukünftige Technologien ein.

[1] API = Kurzform für application programming interface, beschreibt eine Anwendungsprogrammierschnittstelle. Diese Schnittstelle stellt die Anbindung von Softwaresystemen an andere Programme dar.

2 Vergleichbare Anwendungen

Sämtliche folgende Anwendungen dienen der Informationsübermittlung. Hierbei wird teilweise eine Kartendarstellung verwendet oder eingebunden. Die Kombination zwischen Informationsanwendung und QR-Code-Reader[1] wird in keiner der folgenden Anwendungen verwendet. Die Anwendungen dienen lediglich als Informationsanwendung und können zum Beispiel durch QR-Codes geladen oder aktiviert werden. Folgende Anwendungen wurden ausgewählt, da dem Nutzer nützliche Informationen über die bestimmten Themengebiete übermittelt werden außerdem werden teilweise Kartendarstellungen verwendet. Somit erfüllen diese Anwendungen einige Kriterien, welche sich mit den Anforderungen an die zu entwickelnde Anwendung aus dieser Arbeit decken.

Folgende Anwendungen werden anhand ihres Aufbaus untersucht:

Sindelfinger Brunnennavi Hersteller: TERRACS

Abbildung 1: Logo Brunnennavi

Flughafen Stuttgart Hersteller: Flughafen Stuttgart GmbH

Abbildung 2: Logo Flughafen Stuttgart

Berliner Mauer Hersteller: Bundeszentrale für politische Bildung

Abbildung 3: Logo Berliner Mauer

Sindelfinger Brunnennavi von TERRACS

Die Brunnennavi App der Stadt Sindelfingen ist für Smartphones und Tablets angepasst. Die App enthält grundsätzliche Informationen zur Wasserversorgung der Stadt Sindelfingen. Außerdem findet der Nutzer Bohrprofile und Karten zu allen wichtigen Brunnen und Quellen in Sindelfingen.

Es handelt sich um eine native[2] Anwendung, welche unter anderem kostenlos aus dem Google Play Store[3] heruntergeladen werden kann. Die Anwendung ist mit einer Website gekoppelt, welche für mobile Endgeräte ausgelegt ist.

[1] QR-Code-Reader wird auch als Scanner bezeichnet und beschreibt das Lesegerät/Verfahren um QR Codes zu interpretieren
[2] Native Anwendung = Anwendung welche betriebssystemabhängig funktioniert
[3] Google Play Store = Online Marktplatz der Google Inc. für Anwendungen, Bücher, Filme etc.

Diese kann über Barcodes, welche an bestimmten Brunnen angebracht sind, aufgerufen werden. Nach dem Scannen eines Codes, wird der Nutzer auf mobile Websites weitergeleitet. Die Brunnennavi App enthält 4 Hauptmenüs (Home, Wissen, Brunnen, Karte). Diese befinden sich am unteren Rand der Anwendung (Siehe Abbildung 4 und 5). Das Home-Menü enthält eine Erläuterung zur Anwendung. Außerdem findet man Informationen über den Aufbau, die Bedienung und Kontaktadressen. Unter dem Button „Wissen“ befinden sich allgemeine Informationen über Brunnen, Quellen und die Trinkwassergewinnung.

Abbildung 4: Brunnennavi-Home

Abbildung 5: Brunnennavi-Wissen

Das Menü „Brunnen“ enthält Informationen, Bohrprofile und Standorte aller Brunnen und Quellen in Sindelfingen (Abbildung 6 und 7).

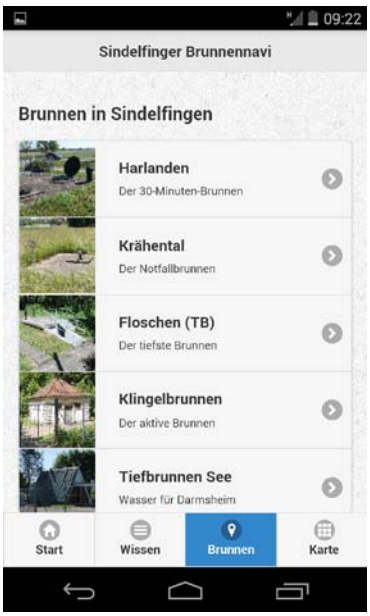


Abbildung 6: Brunnennavi-Brunnen

Abbildung 7: Brunnennavi-Brunnen Info

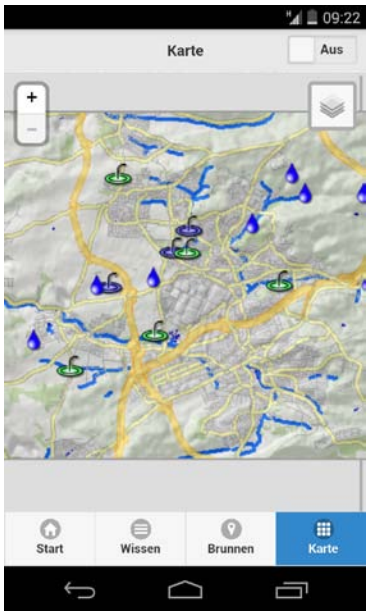


Abbildung 8: Brunnennavi-Karte

Der Hauptmenüpunkt „Karte“ (Abbildung 8) enthält eine Übersichtskarte mit den Standorten aller Brunnen und Quellen. Außerdem bietet dieses Menü eine Navigationsfunktion zu den einzelnen Anlagen.

Die Anwendung ist einfach und strukturiert aufgebaut. Die Bedienung der Anwendung wird über die Buttons gemanaget. Wischfunktionen[1] wurden größtenteils nicht verwirklicht. Die einzige Funktion, welche durch „wischen“ eingesetzt wird, ist das Scrollen der Texte.

Der Nutzer erfährt schnell und übersichtlich sämtliche wichtigen Informationen zu den einzelnen Anlagen. Die Anwendung enthält sehr viele Bilder, welche das Erklärte gut wiederspiegeln. Die Kartendarstellung ist auf ein „Einstiegsfenster“ begrenzt. Dies garantiert schnellere Ladezeiten. Leider ist die Kartenkomponente dadurch auch sehr einfach aufgebaut und macht einen wenig ansprechenden Eindruck. Der Nutzer kann selbst entscheiden, ob er die GPS-Funktion des eigenen Geräts verwenden möchte oder nicht. Außerdem gibt es die Möglichkeit sich das Relief als Hintergrundkarte anzeigen zu lassen. Nach Aktivierung des GPS ist die Anwendung bei den Testläufen immer wieder abgestürzt und hat sich nach der Anzeige einer Fehlermeldung heruntergefahren. Dieses Problem bestätigt wiederum, dass es Handlungsbedarf unter dem Hauptmenü „Karte“ gibt.

Fazit:

Die Anwendung ist einfach und übersichtlich aufgebaut. Sie bietet dem Nutzer viele Informationen über die Wasserversorgung in Sindelfingen. Der Aufbau der Anwendung ist gut strukturiert. Dies macht die Bedienung einfach und intuitiv. Die Performance der Anwendung ist bis auf die Ausnahme des Menüpunkts „Karte“ ebenfalls gut.

[1] Wischfunktionen sind Bestandteil der Gestensteuerung der Smartphones

Flughafen Stuttgart App der Flughafen Stuttgart GmbH

Die Flughafen App des Flughafens Stuttgart ist für Smartphones und Tablets angepasst. Die App enthält sämtliche wichtigen Informationen welche den Flughafen, den Flugbetrieb, Reisende und Dienstleistungen des Flughafens betreffen.

Es handelt sich um eine native Anwendung, welche aus dem Google Play Store geladen werden kann. Außerdem wird die Anwendung im Apple App Store[1] und im Windows Phone Store[2] angeboten.

Die Anwendung ist auf allen drei Plattformen kostenlos erhältlich.

Die App ist über den Startbildschirm in 6 Hauptmenüs unterteilt. Zusätzlich gibt es einen Button, welcher auf das Impressum, Kontakte und die Datenschutzrichtlinien verweist (Abbildung 9). Die Hauptmenüs gliedern sich in Ankunft, Abflug, Shops & Dienstleistungen, Reisende & Besucher, Business to Business und das Unternehmen. Die Auswahl erfolgt durch einzelne Buttons, welche zentral auf dem Startbildschirm angeordnet sind.

Abbildung 9: Flughafen Stuttgart-Home

Unter den Menüpunkten Ankunft und Abflug erhält der Nutzer aktuelle Informationen, welche die abfliegenden und ankommenden Flüge betrifft Abbildung 10 und 11). Diese Menüs sind interessant, da die benötigten Daten in Echtzeit von z.B. Webservern abgerufen werden. Diese Verbindung zwischen den aktuellen Informationen und der „In-App-Darstellung“ der Daten wird selten verwendet. Die meisten anderen Anwendungen verweisen in solchen Situationen auf externe Websites.

[1] Apple App Store = Online Marktplatz des Betriebssystems Apple iOS für Anwendungen Bücher etc.
[2] Windows Phone Store = Online Marktplatz von Windows für Anwendungen, Bücher etc.

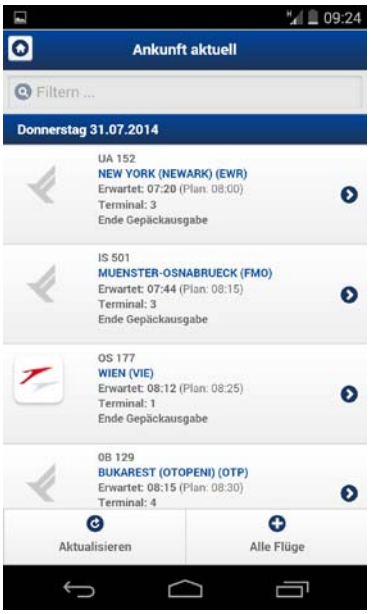


Abbildung 10: Flughafen Stuttgart-Ankunft

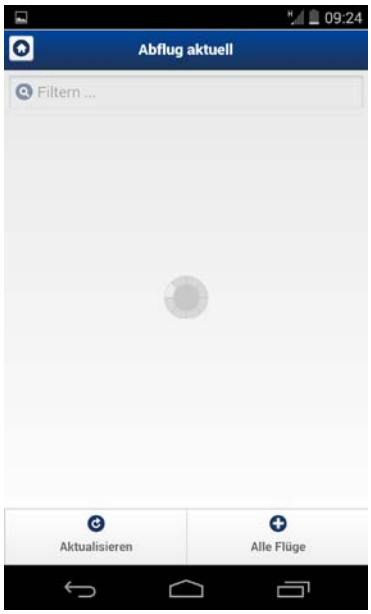


Abbildung 11: Flughafen Stuttgart-Abflug Ladevorgang

Wie in Abbildung 12 und 13 veranschaulicht ist gelangt der Nutzer über den Button „Shops & Dienstleistungen“ zu einer Unterauswahl. Hier kann zwischen Shops, Cafés, Reisebüros oder Dienstleitungen gewählt werden. Außerdem bietet dieses Untermenü eine Suchfunktion. Mit Hilfe dieser Funktion kann der Nutzer sämtliche Bereiche durchsuchen und dadurch schneller zu den gewünschten Informationen gelangen. Durch die Auswahl eines Untermenüs gelangt der Nutzer direkt zu den ausgewählten Informationen und kann sich hierzu weitere, detailliertere Informationen geben lassen.

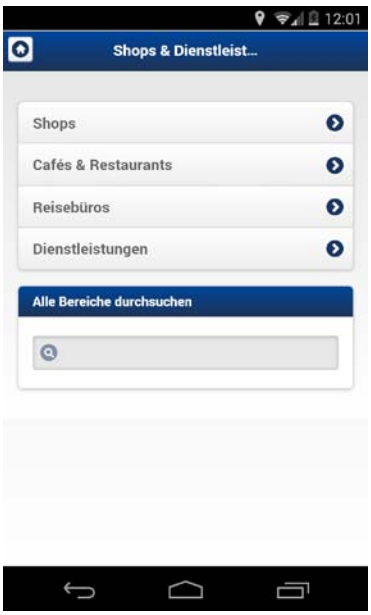


Abbildung 12: Flughafen Stuttgart-Shops & Dienstleistung

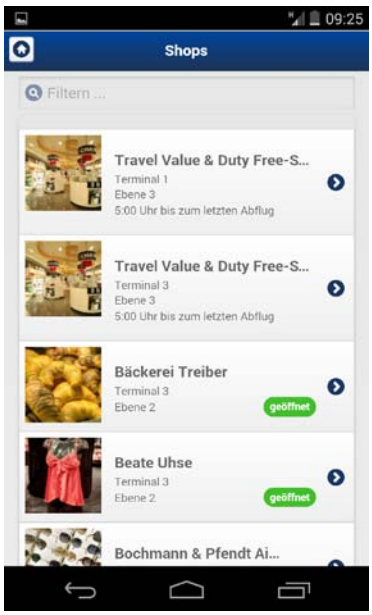


Abbildung 13: Flughafen Stuttgart-Shops

Über das Menü Reisende & Besucher erhält der Nutzer weitere Details wie Fluginformationen, Parken, Service und die Aussichtspunkte für Besucher des Flughafens (Abbildung 14). Die weiteren Informationen werden je nach Auswahl wiederrum in einem Untermenü gegeben. Die Auswahlmöglichkeit „Business to Business“ bietet dieselben Funktionen (Abbildung 15).

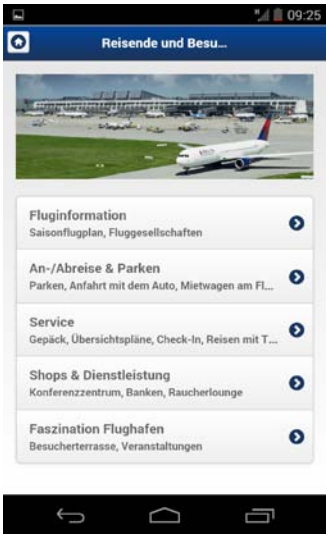


Abbildung 14: Flughafen Stuttgart-Reisende

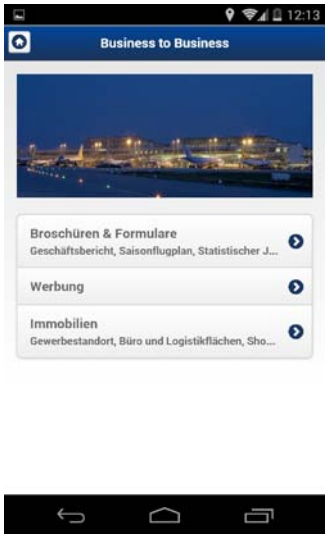


Abbildung 15: Flughafen Stuttgart-Business

Ein weiterer sehr interessanter Menüpunkt ist das Unternehmen. Hier erhält der Nutzer viele wichtige und interessante Informationen über den Flughafen und das Unternehmen. Informationen welche ausgewählt werden können, sind Presse, Zahlen & Daten, Projekte, Jobs & Karriere, Kontakt, Datenschutz und das Impressum. Die verschiedenen Auswahlmöglichkeiten liefern weitere Details. Vor allem das Untermenü Zahlen & Daten ist sehr interessant. Hier werden aufbereitete Informationen kurz und aussagekräftig dargestellt und erläutert. Die Anzeigen werden in Abbildung 16 und 17 gezeigt.

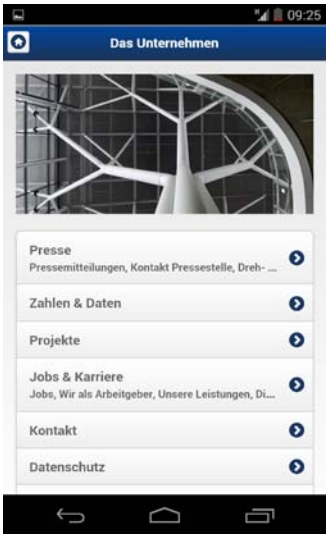


Abbildung 16: Flughafen Stuttgart-Unternehmen

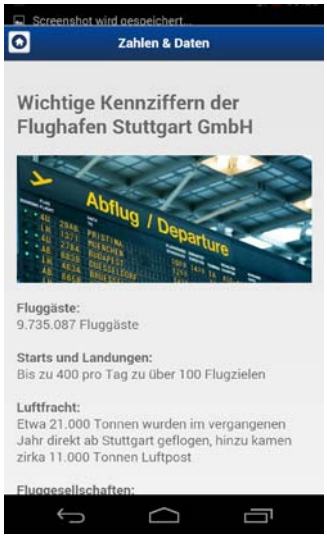


Abbildung 17: Flughafen Stuttgart-Zahlen&Daten

Die Stuttgarter Flughafen App ist sehr strukturiert und übersichtlich aufgebaut. Hauptsächlich wird die Anwendung durch Buttons gesteuert.

Die Daten und Informationen für die Anwendung sind gut aufbereitet. Dem Nutzer werden die Informationen kurz und knapp übermittelt, ohne dass die Anwendung überladen wirkt.

Die App enthält viele nützliche Informationen, welche durch den Einsatz von vereinzelten Bildern unterstützt werden.

Eine Kartendarstellung oder ein QR-Code-Reader werden in der Anwendung nicht verwendet. Die Daten für Ankunft und Abflug werden in Echtzeit geladen und aktualisiert. Hierbei werden die Daten in einer In-App-Darstellung gezeigt. Dieses Verfahren ermöglicht es, nicht durch Links auf externe Websites weiter zu leiten.

Fazit:

Die Anwendung ist einfach und übersichtlich aufgebaut. Sie bietet dem Nutzer viele Informationen über den Flughafen in Stuttgart. Die Bedienung der Anwendung wird über die Buttons gemanagt. Wischfunktionen wurden größtenteils nicht verwirklicht. Die einzige Funktion, welche durch „wischen“ eingesetzt wird, ist das Scrollen der Texte. Die Performance der Anwendung ist trotz der „In-App-Darstellung“ und den verbundenen Ladezeiten sehr gut.

Die Berliner Mauer App der Bundeszentrale für politische Bildung

Die App ist für die Anwendung auf Smartphones und Tablets entwickelt. Sowohl der Google Play Store, wie auch der Apple App Store bieten die Anwendung als Download an. Auf beiden Marktplätzen ist die App kostenlos erhältlich.

Es handelt sich um eine native App. Um den vollen Funktionsumfang der Anwendung nutzen zu können, wird der Nutzer zu Beginn der Anwendung gefragt, ob ein zusätzliches Datenpaket heruntergeladen werden soll (Abbildung 18). Dies hilft dabei, dass die Anwendung auf den Marktplätzen zum Download nicht zu groß wird. Der Download enthält lediglich das reine Grundgerüst der App. Hat der Nutzer die App installiert, kann das Datenpaket und damit die Informationen heruntergeladen werden. Wie in Abbildung 19 gezeigt ist, befinden sich auf dem Startbildschirm der Anwendung 3 Buttons. Diese stellen die drei Hauptmenüpunkte (Mauerverlauf, Mauertouren und Video) dar. Anhand dieser Menüpunkte gelangt der Nutzer zu weiterführenden Untermenüs.

Abbildung 18: Berliner Mauer-Datenbank

Abbildung 19: Berliner Mauer-Home

Über den Menüpunkt Mauerverlauf gelangt der Nutzer zu einer Kartendarstellung (Abbildung 20). Die Kartengrundlage bildet Google Maps. Die Karte wird bildschirmfüllend dargestellt. Über einen „Home-Button“ (links oben) gelangt der Nutzer zum Hauptmenü zurück. In der rechten oberen Ecke befindet sich ein weiterer Button. Dieser dient dazu, zwischen der Kartendarstellung und einer Auflistung der Informationen zu wechseln.

Die Hintergrundkarte enthält den damaligen Mauerverlauf. Auf dem Mauerverlauf sind verschiedene Icons verortet. Jedes Icon steht für ein wichtiges Ereignis oder einen wichtigen Ort auf dem damaligen Mauerverlauf. Wählt der Nutzer eines dieser Icons, kann er auf weitere Informationen zugreifen Abbildung 21). Außerdem gibt es die Möglichkeit die Route zu diesem Ort darstellen zu lassen. Da man sich in einer Großstadt vor allem zu Fuß bewegt, zeigt die Routenplanung lediglich die Verbindungen des ÖPNV. Die Informationen werden durch den Einsatz von Bilder ergänzt.

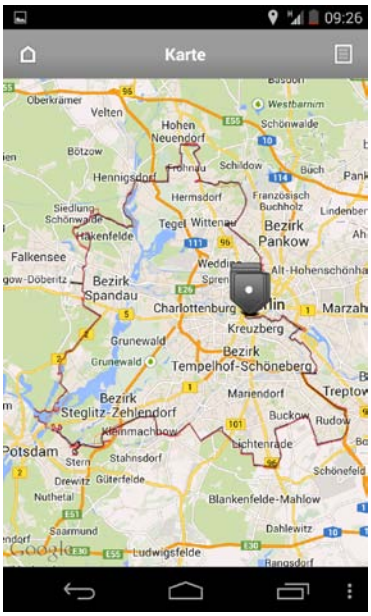


Abbildung 20: Berliner Mauer-Karte

Abbildung 21: Berliner Mauer-Details

Unter dem Menüpunkt Mauertouren (Abbildung 22-24) findet der Nutzer vordefinierte Touren entlang der ehemaligen Berliner Mauer. Hier werden Informationen zur Anzahl der Stationen und der ungefähren Länge der Tour gegeben. Wählt der Nutzer eine Tour, so erhält er auf der folgenden Seite eine kleine Kartendarstellung und weitere Informationen über die einzelnen Stationen. Die Karte zeigt die verorteten Ziele der Tour. Über die Listendarstellung der einzelnen Tourziele können weitere Informationen abgerufen werden. Die Informationen werden wiederum durch den Einsatz von Bildmaterial unterstützt.

Zusätzlich gibt es zu machen Informationen originale Audiomittschnitte aus Interviews oder Reden der damaligen Zeit. Unter dem Menü Mauertouren können zusätzlich eigene Touren erstellt werden. Außerdem hat der Nutzer die Möglichkeit den Entdeckermodus zu aktivieren. Dieser liefert passende Informationen zu den einzelnen Mauerabschnitten, falls der Nutzer keine definierte Tour abläuft und sich frei in Berlin bewegt.

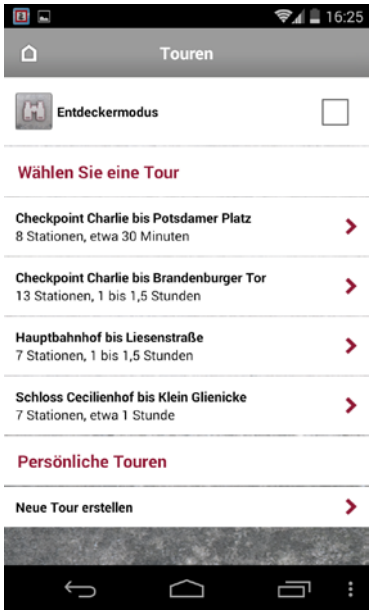


Abbildung 22: Berliner Mauer-Touren

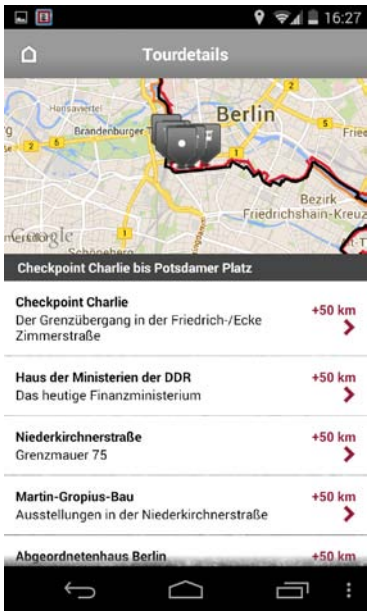


Abbildung 23: Berliner Mauer-Tourendetails

Abbildung 24: Berliner Mauer-Stationen

Der Menüpunkt Video enthält ein Video über die Berliner Mauer. Dieses Video unterstützt die Informationsübermittlung zusätzlich. Das Video fordert eine sehr lange Wartezeit, bevor es betrachtet werden kann. Außerdem wurde die Anwendung nach Anschauen des Videos des Öfteren beendet.

Die Anwendung ist übersichtlich aufgebaut. Dies ermöglicht dem Nutzer eine intuitive Bedienung der Anwendung. Die Bedienung der Anwendung wird über die Buttons gemanagt. Wischfunktionen wurden lediglich dazu eingesetzt, um durch die Informationen in Textform zu scrollen.

Fazit:

Die Anwendung ist einfach und strukturiert aufgebaut. Sie bietet dem Nutzer viele Informationen über den damaligen Mauerverlauf und die Zustände zu dieser Zeit. Die Bedienung ist einfach und intuitiv. Die Performance der Anwendung ist durch das Herunterladen des zusätzlichen Datenpakets gut. Das Mauervideo benötigt jedoch eine lange Zeit zum Laden. Dies könnte einige Nutzer irritieren.

3 Grundlagen und Evaluierungen

In diesem Kapitel wird auf die Grundlagen der Arbeit eingegangen. Hierzu zählen die Datengrundlagen und verschiedene Eckpunkte und Technologien, welche für die Arbeit von Bedeutung sind. Diese Eckpunkte werden genannt und erläutert. Außerdem werden wichtige Entscheidungen begründet. Diese bilden die Grundlage für den späteren Prototypen.

3.1 Datengrundlage

Die Datengrundlage der Arbeit besteht vor allem aus textlichen Erläuterungen und Bildern rund um die Trinkwasserversorgung der Stadt Karlsruhe. Sämtliche Unterlagen wurden von den Stadtwerken Karlsruhe GmbH bereitgestellt.

Ergänzend wurde entsprechende Literatur rund um die Themen „Trinkwasser“ und „Karlsruhe“ verwendet.

Im Folgenden werden die verwendeten Datengrundlagen aufgelistet.

Bücher:

- Förster Katja, Gruber Markus, Maier Matthias; 2011: „Märkte und ihre Brunnen – Häuser und Baugeschichten – Schriftenreihe des Stadtarchivs Karlsruhe Bd. 11“.
  Info-Verlag, Karlsruhe
- Maier, Dietrich; 2004: „Karlsruher Brunnen – Bilder, Modelle, Fotografien“.
  Swiridoff Verlag, Künzelsau
- Stadtwerke Karlsruhe, 2011: „Chronik der Wasserversorgung von Durlach und Karlsruhe“.
  Druck und Verlagsgesellschaft Südwest mbH, Karlsruhe

Historische Bücher:

- Obermüller F., Gerstner E.: „Großherzogliches Hof Wasser Werk Carlsruhe –
  eine Denkschrift mit Atlas“. Druck & Verlag von W. Creuzbauer
- „Das Wasserwerk der Haupt- und Residenzstadt Karlsruhe“ (keine weiteren Angaben)

Veröffentlichungen und Niederschriften:

- Exner Martin: „Hygiene und Öffentliche Gesundheit – in Vergangenheit, Gegenwart und Zukunft“

Poster und Informationstafeln:

- Sämtliche Poster und Informationstafeln der Stadtwerke Karlsruhe GmbH zum Thema Trinkwasserversorgung
- Sämtliche Poster und Informationstafeln des Wasser- und Brunnenmuseums der Stadtwerke Karlsruhe GmbH im geschichtsträchtigen Wasserwerk Durlacher Wald

Festvorträge und Veröffentlichungen:

- Wassergüte im Wandel der Zeit
- Bewertung der Wasserqualität im Wandel der Zeit

Textliche Ausführungen und Pressetexte verschiedener Ausstellungen:

- Wachter Ausstellung, 2004
- Ausstellung „Brunnenmuseum“, 2005
- Ausstellung „Geschichte der Wasserversorgung Durlach“, 2006
- Ausstellung „Abtauchen – Wasser in der Literatur“, 2006
- Ausstellung „Kostbares im Wasserwerk“, 2006
- Ausstellung „R(h)ein“, 2008
- Ausstellung „Wasser im Alltag“, 2008
- Ausstellung „Märkte und ihre Brunnen“, 2011
- Ausstellung „Brunnen an der Talstation der Turmbergbahn“, 2012
- Ausstellung „Wasser alles fließt“, 2013
- Ausstellung „Wasser und Brot“, 2014

Sonstige Datengrundlagen:

- Tonbildschau über die „Historische Trinkwasserversorgung“ von Karlsruhe

3.2 Applikation (App)

Das Kapitel Applikation (App) definiert den Begriff App. Außerdem geht es auf die drei unterschiedlichen Arten von Apps ein (Native App, Web App, Hybrid App).

Mobile Betriebssysteme[1] sind durch Applikationen, sogenannte Apps, erweiterbar. Eine App ist eine Anwendungssoftware für mobile Geräte.

„Der Begriff App stammt ursprünglich aus dem englischen Sprachraum und leitet sich von dem Wort Application ab, was übersetzt so viel wie „Anwendung“ bedeutet. In der englischen Sprache steht die Abkürzung „App“ für „Application Software“ und beschreibt hier alle Art von Anwendungssoftware. Im deutschen Vokabular hingegen bezieht sich der Terminus „App“ in den meisten Fällen bislang auf Software, die lediglich auf mobilen Endgeräten zum Einsatz kommt.[2]

Eine App ist ein Programm,…, das den Benutzer in einem bestimmten Bereich unterstützt. Apps sind Anwendungsprogramme und unterscheiden sich dadurch zum Beispiel von Systemprogrammen, sodass durch die Benutzung ein direkter Wert für den User entsteht.“ ***Gruenderszene*** *[Online]*

Es gibt drei unterschiedliche Arten von Applikationen.

- Native App
- Web App
- Hybrid App

Diese unterscheiden sich vor allem durch ihre Entwicklungsansätze.

[1] Mobile Betriebssysteme sind Betriebssysteme für mobile Endgeräte (Smartphones, Tablets etc.)
[2] Dies ist heute so nicht mehr richtig, da der Begriff ebenfalls in die Desktop-Betriebssysteme integriert wurde

Native App

Das Wort „Nativ“ steht für „echt“. Eine native App wird deshalb auch als echtes/reines Programm bezeichnet. Diese Art der Applikation läuft auf Betriebssystemebene. Aus diesem Grund werden native Apps direkt auf dem Smartphone installiert. Durch die Installation kann die native App, welche aus Binärcode[1] besteht, direkt auf sämtliche API zugreifen, welche das Betriebssystem bereitstellt. Hierdurch besteht die Möglichkeit auf sämtliche Sensoren, die Ortung etc. des Geräts zuzugreifen. Native Apps werden mit Hilfe von Quellcode programmiert.

Die Tabelle 1 zeigt Vor- und Nachteile der nativen App.

Pro	Contra
Freie Gestaltung	Zeitintensive Entwicklung
Zugriff auf Betriebssystem	Hoher Pflegeaufwand
Zugriff auf API	Anpassung

Tabelle 1: Nativet App
***Developergarden** [Online]*

[1] Ein Binärcode verfügt für die Darstellung nur über die Zeichen 0 und 1. Sämtliche Zeichen werden durch eine bestimmte Reihenfolge von 0 und 1 beschrieben.

Web App

Web Apps sind Applikationen die auf Basis einer Website laufen. Da sämtliche neuen Webbrowser der Smartphones Technologien wie HTML5[1], CSS3[2] und JavaScript[3] unterstützen, sind die Web Apps eine gute Alternative. Mit Hilfe der genannten Technologien können nicht nur Websites programmiert werden. Durch diese ist es ebenfalls möglich komplette Applikationen zu verwirklichen, welche dann webbasiert lauffähig sind. Hierbei sind wenig Grenzen gesetzt. Es ist möglich Animationen und interaktive Elemente mit einzubinden. Außerdem kann z.B. auf die Ortungsfunktion zugegriffen werden. Eine Web App kann einfach und wie von Websites gewohnt aktualisiert werden. Durch die einfache Aktualisierungsmöglichkeit sind Web Apps vor allem dann geeignet, wenn es auf hohe Aktualität und leichte Pflege der Anwendung ankommt. Web Apps können meist problemlos auf allen gängigen Betriebssystemen verwendet werden.

Nachteile haben Web Apps dadurch, dass diese in einer nativen App, dem Webbrowser, laufen. Hierdurch ergeben sich einige Einschränkungen. Mit einer Web App ist es z.B. nicht möglich auf sämtliche API des Systems zuzugreifen. Somit können nicht alle Hardwarefunktionen verwendet werden.

In Tabelle 2 sind Vor- und Nachteile der Web App gegenüber gestellt.

Pro	Contra
Webtechnologie	Browserumgebung
Updatefähigkeit	Fehlender Zugriff auf API
Großer Funktionsumfang	Offline-Betrieb meist nicht möglich

Tabelle 2: Web App
***Developergarden** [Online]*

[1] Html 5 = textbasierte Sprache, welche zur Strukturierung von Inhalten wie Texten und Bildern in z.B. Dokumenten oder vor allem auf Websites verwendet wird
[2] CSS = Cascading Style Sheet, Sprache für Stilvorlagen welches vor allem auf Websites im Zusammenhang mit HTML und XML verwendet wird
[3] Java Script ist eine Skriptsprache/Programmiersprache mit der sich komplexe Anwendungen entwickeln lassen.

Hybrid App

Hybrid Apps vereinen die beiden vorgestellten Möglichkeiten einer Applikation. Bei Hybrid Apps gibt es sowohl einen nativen Teil, welcher direkt auf dem Betriebssystem läuft und einen weiteren Teil, welcher auf Webtechnologie basiert. Um die nativen und webbasierten Eigenschaften zu vereinen wird eine „Brücke“ benötigt.

Hybrid Apps haben Vorteile bei kleineren Updates. Diese können meist ohne Probleme durchgeführt werden. Nachteile der Hybrid Apps sind vor allem, dass die Anwendungen nicht offline lauffähig sind. Der Inhalt der Applikation ist leidglich durch das Internet sichtbar.

Die Tabelle 3 zeigt Vor- und Nachteile der Hybrid App.

Pro	Contra
Kombination aus Nativer App und Web App	Offline-Betrieb nicht möglich
Einfache Updates	Schlechte Performance

Tabelle 3: Hybrid App
***Developergarden** [Online]*

In Tabelle 4 werden die Vor- und Nachteile der unterschiedlichen App-Typen zusammenfassend dargestellt.

Native App		Web App		Hybrid App	
Pro	Contra	Pro	Contra	Pro	Contra
Freie Gestaltung	Zeitintensive Entwicklung	Webtechnologie	Browserumgebung	Kombination aus Nativer App und Web App	Offline-Betrieb nicht möglich
Zugriff Betriebssystem	Hoher Pflegeaufwand	Updatefähigkeit	Zugriff auf API	Einfache Updates	Schlechte Performance
Zugriff auf API	Anpassung	Großer Funktionsumfang	Offline-Betrieb meist nicht möglich	-	-

Tabelle 4: Vergleich der drei Varianten

3.3 Mobile Betriebssysteme

Da für den Aufbau des Prototypen eine native Anwendung umgesetzt werden soll und diese nicht plattformunabhängig ist, werden im folgenden Kapitel die gängigsten Betriebssysteme für Smartphones vorgestellt. Es werden Vor- und Nachteile der unterschiedlichen Betriebssysteme genannt und verglichen.

3.3.1 Apple iOS

Das Betriebssystem, welches ausschließlich von Apple Inc. entwickelt und eingesetzt wird, läuft auf verschiedenen hauseigenen Produkten. Diese Geräte sind neben der iPhone Reihe die iPod- sowie die iPad-Reihe. Das Betriebssystem ist einfach zu bedienen. Bei dem System handelt es sich um ein geschlossenes System. Der Programmiercode ist nicht frei verfügbar. Die Softwareupdates werden in der Regel jährlich bereitgestellt und sind ebenfalls für ältere Modellreihen verfügbar. Der Eingriff in die Speicherverwaltung durch das Betriebssystem ist stark eingeschränkt. Es ist nicht möglich weitere Speichermedien einzubinden. Die Synchronisation der Geräte mit einem Computer erfolgt durch die iTunes Software, welche Apple Inc. bereitstellt. Weiter Synchronisationsmöglichkeiten sind durch das Internet gegeben.

Das Betriebssystem ist für seine sehr einfache Handhabung bekannt. Es ist relativ unanfällig gegen Viren und läuft stabil. iOs bietet dem Nutzer viele Multimediaeigenschaften.

Als Nachteile wird die starke Bindung an die iTunes-Software gesehen. Das System ist geschlossen und somit eingeschränkt. Es enthält keine Flash Unterstützung[1]. Die Systemoberfläche ist nur bedingt anpassbar. Der größte Kritikpunkt liegt in der beschränkten Modellauswahl für Geräte aus dem Hause Apple und deren hohen Preise.

[1] Flash = Adobe Flash, Plattform zur Programmierung und Darstellung multimedialer Inhalte

3.3.2 Windows Phone

Dieses Betriebssystem wird von Microsoft Corporation entwickelt. Es wird von verschiedenen Handy-Anbietern verwendet. Diese müssen hierfür an Microsoft Lizenzgebühren entrichten. Das Betriebssystem ist einfach zu bedienen. Der Programmiercode des Systems ist nicht frei verfügbar. Es handelt sich um ein geschlossenes System. Die Updates für das Betriebssystem werden von Microsoft bereitgestellt. Durch die Entrichtung der Lizenzgebühren müssen Handy-Anbieter die Softwareupdates nicht noch einmal überarbeiten. Dies führt zu einer schnellen Auslieferung von Updates. Damit das Betriebssystem auf allen Geräten einwandfrei läuft, werden von Microsoft strenge Hardwarevorgaben an die Entwickler gegeben. Seit der Softwareversion Windows Phone 8 ist es möglich, weitere Speichereinheiten einzubinden. Außerdem ist es seit dieser Version weiterhin möglich, die Synchronisation mit dem Computer ohne Zusatzsoftwareüber das Internet vorzunehmen.

Die Vorteile von Windows Phone 8 liegen in der hohen Performance des Systems. Die Bedienung ist sehr einfach und intuitiv. Um Daten des Endgeräts mit einem Computer zu synchronisieren fungiert das Endgerät als USB-Datenlaufwerk.

Die Nachteile liegen darin, dass es sich um ein geschlossenes System handelt. Dadurch ist das Betriebssystem eingeschränkt. Es bietet keine Flash Unterstützung und keinerlei Offline-Synchronisationsmöglichkeiten[1]. Externe Speichermedien werden erst ab der Softwareversion Windows Phone 8 unterstützt.

[1] Offline Synchronisation = Synchronisation ohne Zugriff aus das des Internet

3.3.3 Android

Android wird von Google Inc. entwickelt. Android kommt bei vielen unterschiedlichen Handy-Anbietern zum Einsatz. Es ist sowohl auf Smartphones, wie auch auf Tablet-PCs einsetzbar. Android ist ein offenes Betriebssystem. Der Programmcode ist im Gegensatz zu den vorherigen Systemen frei verfügbar. Android wird auf verschiedenen Geräten von verschiedenen Herstellern eingesetzt. Neben diesen externen Herstellern entwickelt Google Inc. eigene Smartphones und Tablets der Nexus Baureihe. Softwareupdates werden durch Google entwickelt und bereitgestellt. Das Betriebssystem wird auf Hardwaregrundlage der Nexus-Baureihe aufgebaut und getestet. Diese Geräte erhalten daher die neuen Systemupdates zuerst. Anschließend erhalten andere Hersteller die Updates und müssen diese noch auf ihre eigenen Geräte anpassen. Dies führt teilweise zu langen Wartezeiten bei externen Herstellern. Android bietet generell die Möglichkeit den internen Datenspeicher durch externe Speichermedien zu erweitern. Um Daten der Geräte auf einen Computer zu übertragen, fungiert Android als USB-Datenlaufwerk, welches eine bekannte Ordnerstruktur aufweist. Für die Synchronisation wird keine Zusatzsoftware benötigt. Diese geschieht standardmäßig über das Internet.

Die Vorteile von Android liegen in der Vielseitigkeit. Das System ist sowohl auf einfachen Smartphones, bis hin zu Smartphone-Boliden[1] einsetzbar. Das Betriebssystem bietet eine moderne und funktionale Bedienoberfläche, welche sich individuell anpassen lässt. Durch die Entwicklung der Google Inc. sind sämtliche andere Google-Produkte (Maps, Mail etc.) perfekt in das System eingebunden.

Die Nachteile des Betriebssystems liegen in der Update-Politik[2]. Außerdem ist die Software teilweise anfällig für Schadsoftware[3]. Durch die notwendige Anpassung der Software durch externe Hersteller ist das Erscheinungsbild des Systems nicht auf allen Geräten gleich.

[1] Unter Smartphone Boliden versteht man die Top Modelle verschiedener Hersteller
[2] Update Politik beschreibt die Art und Weise wie Updates vom Hersteller bereitgestellt werden
[3] Schadsoftware = Software schadet dem Betriebssystem

3.3.4 Andere Betriebssysteme

In Kapitel 3.3.1 bis 3.3.3 werden die drei Marktführer in Sachen mobile Betriebssysteme eingeführt. In diesem Kapitel wird auf zwei weitere Systeme eingegangen, die nur einen kleinen Marktanteil ausmachen. Die Zukunft dieser Systeme ist unsicher.

Bada ist eine hauseigene Entwicklung der Samsung Group. Das System läuft ausschließlich auf Geräten dieses Anbieters. Der Programmiercode des Betriebssystems ist nicht frei verfügbar. Es handelt sich um ein geschlossenes System. Softwareaktualisierungen werden zum heutigen Zeitpunkt bereits nicht mehr durchgeführt und bereitgestellt. Dies führt zum langsamen Rückgang der Geräte, welche dieses Betriebssystem unterstützen. Im Endeffekt ist es ein Betriebssystem, das relativ zeitnah nicht mehr auf dem Markt verfügbar sein dürfte.

Blackberry-OS wird vom gleichnamigen Handy-Hersteller BlackBerry Limited entwickelt. Die Geräte des Herstellers sind vor allem auf Unternehmenskunden zugeschnitten. Das System wird lediglich auf hauseigenen Geräten eingesetzt. Es handelt sich um ein geschlossenes System. Der Programmcode ist nicht frei verfügbar. Mit dem neuesten Update haben die Entwickler das Betriebssystem ein wenig offener gestaltet. Seit dieser Version ist es möglich, auch externe Programme zu installieren. Dadurch wurde das System offener und ebenfalls besser durch die Kunden akzeptiert. Die Synchronisation mit einem Computer ist über eine mitgelieferte Software möglich.

Der Marktanteil des Betriebssystems ist sehr gering. Dies liegt vor allem daran, dass sich die Hersteller besonders auf den Markt der Unternehmenskunden fokussieren.

In Tabelle 5 sind die vorgestellten mobilen Betriebssysteme und deren Eigenschaften zusammengefasst.

Betriebssystem	Apple iOS	Windows Phone	Android	Bada	Blackberry-OS
Hersteller	Apple Inc.	Microsoft Corporation	Google Inc.	Samsung	BlackBerry
Programmcode	Nicht frei verfügbar	Nicht frei verfügbar	Frei verfügbar	Nicht frei verfügbar	Nicht frei verfügbar
System	Geschlossen	Geschlossen	Offen	Geschlossen	Geschlossen
Aktualisierung	Jährlich	Unterschiedlich	Jährlich	Eingestellt	Unterschiedlich
PC-Anschluss	Software	USB-Laufwerk	USB-Laufwerk	Software	Software
Synchronisation	iTunes/Internet	Internet	Internet	Software	Software

Tabelle 5: Betriebssysteme

3.4 App Store

Im folgenden Kapitel werden die verschiedenen online Marktplätze beschrieben. Diese stehen teilweise direkt im Zusammenhang mit den Herstellern mobiler Endgeräte oder mobiler Betriebssysteme. Die Stores sind vergleichbar mit digitalen Marktplätzen.

*„**App Store** ist die Bezeichnung für eine Internet-basierte digitale Vertriebsplattform für Anwendungssoftware. Diese stammt entweder vom Betreiber der Plattform selbst, oft aber überwiegend von Drittanbietern wie etwa freien Programmierern. Der Service ermöglicht es Benutzern, Software für Mobilgeräte wie Smartphones und Tabletcomputer oder auch für Personal Computer aus einem Anwendungskatalog herauszusuchen und herunterzuladen. Über den App-Marktplatz wird die Bezahlung der Apps abgerechnet. Dabei erhält der App-Store-Betreiber in der Regel eine Provision. Neben dem Herunterladen hält der App Store die Apps auf dem jeweiligen Gerät auf dem aktuellen Stand. Die Hersteller von Smartphone-Betriebssystemen, wie Apple, Google, Microsoft und Blackberry bieten für ihre Plattform jeweils einen App Store an. Es gibt aber auch Hersteller-unabhängige App Stores, wie z. B. den Amazon Appstore. Auf das Angebot kann üblicherweise über eine eigene App-Store-Software auf einem Mobilgerät oder über eine Website im Internet zugegriffen werden.“ **Wikipedia** [Online]*

Die in Kapitel 3.3 vorgestellten Betriebssysteme haben alle „hauseigenen“[1] Stores für Applikationen. Im Folgenden wird auf die Marktplätze Google Play Store, Apple App Store und Windows Phone Store eingegangen. Diese Systeme haben die größten Marktanteile in Deutschland (siehe Abbildung 25).

Germany	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	69.0	75.4	6.4
BlackBerry	1.1	0.5	-0.6
iOS	21.7	17.3	-4.4
Windows	3.4	5.9	2.5
Other	4.8	0.9	-3.9
GB	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	54.4	54.9	0.5
BlackBerry	6.4	3.2	-3.2
iOS	32.4	29.9	-2.5
Windows	5.9	11.3	5.4
Other	0.9	0.6	-0.3
France	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	61.0	65.9	4.9
BlackBerry	5.1	1.6	-3.5
iOS	23.7	20.3	-3.4
Windows	5.0	11.4	6.4
Other	5.1	0.8	-4.3
Italy	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	54.2	66.2	12.0
BlackBerry	2.6	1.8	-0.8
iOS	23.1	12.8	-10.3
Windows	12.7	17.1	4.4
Other	7.4	2.1	-5.3
Spain	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	85.9	86.2	0.3
BlackBerry	2.4	0.2	-2.2
iOS	7.3	6.7	-0.6
Windows	1.2	5.6	4.4
Other	3.2	1.3	-1.9

USA	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	46.2	50.6	4.4
BlackBerry	0.9	0.4	-0.5
iOS	49.7	43.9	-5.8
Windows	2.4	4.3	1.9
Other	0.8	0.8	0.0
China	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	73.7	78.6	4.9
BlackBerry	0.0	0.1	0.1
iOS	21.2	19.0	-2.2
Windows	0.9	1.1	0.2
Other	4.2	1.3	-2.9
Australia	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	56.0	57.2	1.2
BlackBerry	1.0	0.8	-0.2
iOS	38.5	35.2	-3.3
Windows	3.0	5.2	2.2
Other	1.5	1.7	0.2
LatAm 3 (BR, BX, AR)	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	61.6	83.5	21.9
BlackBerry	10.3	2.8	-7.5
iOS	4.4	4.3	-0.1
Windows	6.8	4.9	-1.8
Other	17.0	4.5	-12.5
EU5	3 m/e Dec 2012	3 m/e Dec 2013	% pt. Change
Android	62.9	68.6	5.7
BlackBerry	3.7	1.5	-2.2
iOS	23.7	18.5	-5.2
Windows	5.6	10.3	4.6
Other	4.0	1.1	-3.0

Abbildung 25: Marktanteil Betriebssysteme

[1] Hauseigene = Vom Hersteller selbst entwickelt und unter deren Verwaltung

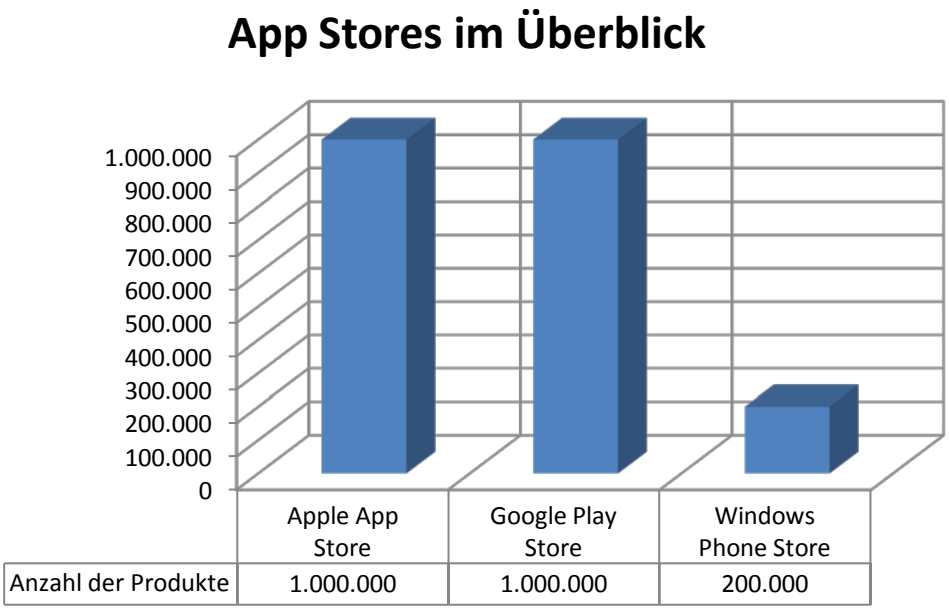
Der Google Play Store ist mit dem Betriebssystem Android verbunden. Auf diesem Marktplatz werden vor allem Apps angeboten. Zusätzliche Angebote bilden Bücher, Filme und Musik. Diese Angebote können aus dem Google Play Store heruntergeladen werden. *„Da sich Android inzwischen als Marktführer durchgesetzt hat, finden sich im Appstore im Grunde alle wichtigen Apps, die man auch von Apples iPhone kennt. Exklusive Apps, die es nur auf einem der beiden Systeme gibt, sind selten geworden.“* ***Süddeutsche*** *[Online]*

Die Benutzung des Play Store ist mit dem Einrichten eines Benutzerkontos verbunden. Der Google Play Store wird von Google Inc. verwaltet. Sämtliche Applikationen, welche auf dem Marktplatz vorhanden sind, unterliegen keinerlei Überprüfung durch Google. Dies bietet zwar eine Hintertür für Schadsoftware, vereinfacht es aber gleichzeitig den Entwicklern von Applikationen, diese schnell und einfach verbreiten zu können.

Der Apple App Store wird von Apple Inc. verwaltet. Der Marktplatz ist eng mit iTunes verbunden. Hier können Apps, Musik, Filme und Bücher heruntergeladen werden. Jegliche Apps werden von Apple-Ingenieruren vor der Aufnahme in den App Store mehrfach auf Kompatibilität und Schadsoftware überprüft. App-Entwickler unterliegen strengen Vorgaben für ihre Apps durch Apple. Die strenge Überprüfung verhindert das Eindringen von Schadsoftware in den App Store. Gleichzeitig beschränkt es aber Entwickler in ihren Möglichkeiten und schafft große Hürden. Das App-Angebot bietet dieselbe Vielseitigkeit wie im Play Store von Google. Beide Stores enthalten quasi dasselbe App-Angebot. ***Süddeutsche*** *[Online]*

Der Windows Phone Store wurde zusammen mit der Windows Phone 8 Software eingeführt. Da diese Einführung einen großen zeitlichen Unterschied zum Google Play Store und dem Apple App Store aufweist, sind im Windows Phone Store weniger Apps enthalten. Das Angebot der wichtigsten Apps und App-Entwickler konnte jedoch durch Microsoft übernommen werden. Viele Entwickler erklärten sich bereit ihre Apps ebenfalls in den Windows Phone Store zu konvertieren und einzustellen. *„Dennoch reicht er quantitativ und qualitativ noch lange nicht an die Auswahl von Apple oder Google heran. Dafür stechen einige vorinstallierte Apps, etwa die Navigationslösung, besonders hervor.“* ***Süddeutsche*** *[Online]*

Die folgende Abbildung gibt einen Überblick über die verschiedenen App Stores.



Quelle: Apple, Google, Microsoft. Eigene Grafik

Abbildung 26: App Stores im Überblick

3.5 Möglichkeiten zur Einbindung von Kartenkomponenten

In diesem Kapitel werden verschiedene Kartendienste vorgestellt. Die Kartendienste werden analysiert und im Hinblick auf Verfügbarkeit, Zuverlässigkeit und Nutzergewohnheiten bewertet. Außerdem werden Möglichkeiten zur Einbindung der verschiedenen Kartendienste in den Prototypen erläutert.

Um in einer Anwendung raumbezogene Daten darzustellen, ist es sinnvoll einen Kartendienst[1] einzubinden. Durch eine Kartendarstellung werden Nutzer mit den interessanten Orten zusammengebracht. Reine Sachdaten können informativ sein. Diese sind aber meist nicht interessant genug. Durch die Kartenkomponente werden die Sachdaten einem bestimmten Ort zugewiesen. Dies ermöglicht dem Nutzer eine wichtige raumbezogene Zusatzinformation. Außerdem bieten die meisten Kartenkomponenten gleichzeitig eine Navigationsfunktion. Diese Funktion ermöglicht es dem Nutzer einfache und schnell den gesuchten Ort zu finden. Der Zugriff auf die Navigationsfunktion geschieht über die eingebauten Schnittstellen des Endgeräts.

Grundsätzlich kann jeder bestehende Kartendienst in eine Anwendung integriert werden. Hierbei ist darauf zu achten, welche Anforderungen die Anwendung an die Kartenkomponente stellt. Die wichtigsten Anforderungen sind Verfügbarkeit, Zuverlässigkeit, Nutzergewohnheiten und Kosten. Vor der Einbindung einer Kartenkomponente müssen die zur Verfügung stehenden Dienste auf diese Punkte analysiert werden.

Mögliche Kartendienste, welche in Anwendungen eingebunden werden können sind:

- Google Maps — ***Google Maps*** *[Online]*
- Open Street Map — ***Open Street Map*** *[Online]*
- Bing Maps Microsoft — ***Bing*** *[Online]*
- Nokia HERE — ***HERE*** *[Online]*

Google Maps

Google Maps ist der wohl am meisten verbreitete Online-Kartendienst. Nahezu jeder Internetnutzer kennt diese Art der Kartendarstellung. Die Karten von Google Maps weisen eine hohe Qualität auf. Durch die zahlreichen Informationen und Daten der Google Inc., welche ebenfalls in Google Maps enthalten sind, bietet diese Kartenkomponente viele Informationen. Google Maps bietet zahlreiche Hintergrundkarte wie z.B. eine Straßenkarte oder Luftaufnahmen. Außerdem entwickelt sich das System ständig weiter. Das Angebot reicht mittlerweile bis hin zum Google Street View Projekt und 3D-Kartendarstellungen. ***PC Welt*** *[Online]*

[1] Vernetze Services welche dem Nutzer Geodaten zugänglich machen

Open Street Map (OSM)

OSM wird anhand einer Community gepflegt und weiterentwickelt. Der Kartendienst bietet verschiedene Hintergrundkarten. Diese gliedern sich in unterschiedliche Stile. Es gibt einen deutschen Kartenstil[1] sowie z.B. Radfahrkarten. OSM bietet eine große Adressdatenbank sowie zusätzliche Sonderziele wie etwa Landschaften. OSM ist ein völlig freier Kartendienst. Der Dienst kann kostenlos für kommerzielle Projekte verwendet werden. In OSM sind keinerlei Nutzungsbeschränkungen integriert. Es ist erlaubt, eigene Services zu betreiben und diese kommerziell einzusetzen.

***PC Welt** [Online]*

Bing Maps

Bing Maps ist ein Kartendienst, welcher von Microsoft entwickelt und bereitgestellt wird. Bing Maps beinhaltet eine Straßenkarte, Luftansicht und eine Besonderheit, die Kartenansicht in Vogelperspektive. Die Karten sind eher zurückhaltend dargestellt. Bing Maps bietet die nahezu gleiche Adressdatenbank wie Google Maps. Die Kartenqualität ist ebenfalls mit den Google Maps Karten vergleichbar. Herausragend ist die Vogelperspektive der Bing Maps.

***PC Welt** [Online]*

HERE

Der Kartendienst HERE wird von Nokia entwickelt und bereitgestellt. HERE ist einer der neuesten Kartendienste, welche sich auf dem Markt befinden. 2007 übernahm Nokia den Kartenanbieter NAVTEQ und deren Straßenkarten. Der Kartendienst glänzt mit seinen hochwertigen Kartendaten und der einfachen Darstellung der Karteninhalte. HERE bietet, wie die meisten Konkurrenten, Straßenkarten, Sattelitenkarten, Öffentlicher Verkehr Karten und Geländekarten als Kartengrundlage an. Wie Google Maps enthält der Kartendienst eine „Live Verkehrsinformation“ und einen 3D-Modus. Dieser kann aber auf mobilen Endgeräten nicht verwendet werden. Die Adressdatenbank des Kartendienstes ist vergleichbar mit den anderen Konkurrenten.

***PC Welt** [Online]*

[1] Der deutsche Kartenstil ähnelt den amtlichen Karten von Deutschland

Kosten und Nutzungsbeschränkungen

Bei den Kosten und Nutzungsbeschränkungen der einzelnen Kartendienste gibt es große Unterschiede.

Google Maps ist grundsätzlich kostenfrei. Es besteht allerdings eine Nutzungsbeschränkung. Wird diese überschritten, fallen geringe Kosten im einstelligen Bereich an. Um diese Nutzungsbeschränkung zu umgehen, können die kostenpflichtigen Google Enterprise Tools verwendet werden. In Google Enterprise sind mögliche anfallende Kosten bereits enthalten.

„Websites und Anwendungen, die das Google Maps API verwenden, können für jeden Dienst pro Tag kostenlos bis zu 25.000 Kartenladevorgänge generieren. … Zur Berücksichtigung von Websites, bei denen kurze Nutzungsspitzen auftreten, werden die Nutzungsbegrenzungen erst wirksam, wenn eine Website die Begrenzung an mehr als 90 aufeinanderfolgenden Tagen überschritten hat.“

Google Developers *[Online]*

Open Street Map (OSM) ist ein völlig kostenfreier Kartendienst. Es handelt sich um ein Open Source Produkt. Die zugrundeliegenden Daten können sowohl privat wie auch kommerziell genutzt werden. Es gibt keinerlei Einschränkungen. So kann mit Hilfe von OSM z.B. ein Navigationssystem entwickelt und dieses dann kommerziell vertrieben werden.

Bing Maps verfolgt dasselbe Konzept wie Google Maps. Generell ist der Kartendienst kostenfrei. Es gibt aber eine Nutzungsbeschränkung. Diese liegt bei Bing Maps bei 125.000 Transaktionen innerhalb eines Jahres. Dies entspricht einer Nutzung von ca. 343 täglichen Zugriffen. Die Nutzungsbeschränkung ist enger gestaffelt wie bei Google Maps. Wie Google Maps bietet Bing Maps ebenfalls eine Enterprise-Lösung an, welche es vereinfacht die Nutzungsbeschränkung zu regeln.

„Unter folgenden Bedingungen dürfen Entwickler in beliebigen Unternehmen die Bing Maps-Plattform kostenfrei nutzen:

- *die Nutzung erfolgt auf öffentlichen, nicht durch Passwort geschützten Websites,*

  Es gilt eine Begrenzung von 125.000 Transaktionen über 12 Monate“.

BING *[Online]*

HERE Maps darf lediglich für nicht kommerzielle Zwecke eingesetzt werden. In den Nutzungsbestimmungen wird eindeutig eine kommerzielle Nutzung untersagt. Für private Zwecke steht HERE frei und ohne Einschränkungen zur Verfügung. Entwickler, welche HERE Maps kommerziell nutzen möchten, müssen direkt mit Nokia in Verbindung treten um die Nutzungsbestimmungen und mögliche Entgeltansprüche abzuklären.

"You agree to:

- *comply with applicable laws, the Terms and good manners;*
- *use the Service only for your personal, non-commercial purposes.*

You agree:

- *to use the Content only for your personal, non-commercial purposes;*
- *to use the Content in accordance with the restrictions set out in the applicable laws, additional terms, guidelines and policies or on the product pages that apply to that particular piece of the Content;*
- *not to make copies, give, sell, resell, loan, rent, offer, broadcast, send, distribute, transfer, communicate to the public, reproduce, modify, display, perform, commercially exploit or make the Content available unless otherwise authorised in the applicable Terms and to advise Nokia promptly of any such unauthorised use;"*

***Here** [Online]*

Verfügbarkeit, Zuverlässigkeit und Nutzergewohnheiten

Die Verfügbarkeit und die Zuverlässigkeit sind bei allen Kartendiensten gegeben. Sämtliche Systeme laufen zuverlässig und fehlerfrei. Einige Kartendienste sind von Haus aus auf mobilen Endegräten installiert. Google Maps ist der hauseigene Kartendienst von Google auf dem mobilen Betriebssystem Android. Bing Maps ist auf allen Endgeräten von Microsoft und somit dem Betriebssystem Windows Phone vorinstalliert. Der Kartendienst HERE kommt auf sämtlichen Endgeräten von Nokia zum Einsatz. Lediglich der Kartendienst von OSM ist ein völlig freier Kartendienst, welcher in keinem speziellen Betriebssystem zum Einsatz kommt. Bei der Analyse der Nutzergewohnheiten steht Google Maps im Vordergrund. Dies ist nach wie vor der meistverwendetste Kartendienst. Jeder Internetnutzer kennt Google Maps und hat es wahrscheinlich schon einmal verwendet. Die Konkurrenten stehen hier in der Beliebtheit hinten an. Goolge Maps ist der gängigste und bekannteste Kartendienst. Die folgende Tabelle 6 gibt einen zusammenfassenden Überblick über die Onlinekartendienste.

Map Service	**Google Maps**	**Bing Maps**	**OpenStreetMap**	**Nokia Here**
Availability				
Full Extra Functionality	Australia, Canada, China, France, Germany, Israel, Italy, Netherlands, Spain, UK, United States.	United States, Canada (with the exception of the rural northeastern provinces), United Kingdom, Germany, Italy, Australia, New Zealand, Japan(Yahoo Japan only), India	All	More than 180 navigable countries
Officially Supported Web Browsers	IE7+, Firefox 2.0.0.8+, Safari 3+, Mozilla 1.7+, Opera 8.02+, Google Chrome 1+	IE6+, Firefox 2+, Safari 3+	IE7+, Mozilla Firefox 3.5+, Google Chrome 4+, Safari 4+	IE7+, Mozilla Firefox 3.5+, Google Chrome 4+, Safari 4+
Viewing Interface				
Degrees of Motion	Vertical, Horizontal, Depth, Rotation(beta), 360 Panoramic (Street View), 3D Mode (Google Earth JavaScript)	Vertical, Horizontal, Depth, 360 Panoramic (Streetside), 3D Mode(Tilt, Pan, Rotate)	Vertical, Horizontal, Depth	Vertical, Horizontal, Depth (zoom), Tilt (3D), Rotate 360 degrees

Map Types	Map, Satellite, Hybrid, Street, Traffic, 3D	Road, Satellite, Hybrid, Bird's Eye, Traffic, 3D, Street, London Street Map, Ordnance Survey Map, Venue Maps	Standard Map, Transport Map, Cycle Map, MapQuest Open, Humanitarian	Map View, Satellite, Terrain, 3D, Traffic, Public Transportation, Heat Map, Map Creator, Explore Places, Community
3D Mode	Yes	No since November 2010	Yes	Yes
Backend	JSON	.NET	XML	Java, JavaScript
Data				
Age of Map Imagery	Updated Daily		Updated Daily	
Age of Satellite Imagery	1–3 years old	1–3 years old	No	1–3 years old
Map Data Providers	MAPIT, TeleAtlas, DigitalGlobe, MDA Federal	NAVTEQ, Intermap, Pictometry International, NASA	User Contributions	Navteq
Searching				
Entity	Business, Places of Interest, Airport Code	Businesses, Collections, Directories, Postal Codes	All Possibility, no restriction	Business, Places of Interest, Landmarks, Airport Codes
Directions				
Directions	Yes	Yes	Yes	Yes

Public Transport Integration	Yes	Yes	Yes	Yes
Walking Directions	Yes	Yes	Yes	Yes
Bicycle Directions	Yes	No	Yes	No
Collaboration / Embedding				
Application Integration	Google Earth, BMW Assist	Microsoft Office, Microsoft Research WorldWide Telescope, Microsoft Photosynth	Apple Maps, MapQuest, Foursquare, Craigslist, Apple iPhoto, Wikipedia, World Bank	Mercedes-Benz In Car System, Alpine In Car System
API Available	Yes	Yes	Yes	Yes
		Mobile		
Mobile-Specific Website	Yes	Yes	Yes	Yes
Mobile-Specific Application	Yes	Yes	Yes – OSM and for Android	Yes, for Nokia mobile phones and iOS
GPS Integration	Yes	Yes	Yes	Yes
Directions	Yes	Yes	Yes	-
Interactive Maps	Yes	Yes	No	Yes

Types of Map	Map, Satellite, Terrain, Street	Map, Road, Aerial, Street, London Street Map, Ordnance Survey Map, Venue Maps	Map, Terrain, Satellite (exclusive to editing)	Map, Satellite, Terrain, 3D with plugin, 3D without plugin for compatible browsers, Night Mode
Cell-based Location	Yes	No	-	Yes
Wi-Fi Location	Yes	Yes	Yes	Yes

Tabelle 6: Übersicht online Kartendienste-bearbeitete Tabelle

Wikipedia Kartendienste *[Online]*

3.6 Möglichkeiten zur Einbindung der QR-Code-Komponente

In diesem Kapitel wird einen Überblick über die QR-Code Technologie gegeben. Es wird auf QR-Code Generatoren und QR-Code Scanner eingegangen. Verschiedene Anbieter werden verglichen und evaluiert. Außerdem wird auf den Nutzen der Technologie für diese Arbeit eingegangen.

„QR-Codes sind 2D-Codes, die von Handys, Smartphones und Tablets eingescannt und ausgelesen und in denen Webadressen, Telefonnummern, SMS und freier Text untergebracht werden können. Sie verbinden physische und virtuelle Welt und spielen u.a. im Publikationswesen und im Marketing eine Rolle.“

Wirtschaftslexikon *[Online]*

„Der QR-Code (Quick Response) ist ein international anerkannter 2D-Code, der von ISO/IEC unter 18004 standardisiert wurde, primär aber in Fernost eingesetzt wird. Entwickelt hat ihn die japanische Firma Denso. Eingesetzt wird der QR-Code für das schnelle Scannen von Informationen.“

IT Wissen *[Online]*

Die Abkürzung QR steht für den englischen Ausdruck „quick response“. Zu Deutsch handelt es sich um eine „schnelle Antwort“ oder „schnelle Reaktion“. Der QR-Code gehört zu den 2D-Codes. Ein Code besteht aus minimal 21 mal 21 quadratischen Elementen. Die maximale Anzahl an Elementen sind 177 mal 177. Ein QR-Code weist eine maximale Speicherkapazität von 2953 Byte auf. Dies entspricht 7089 Ziffern und 4296 ASCII-Zeichen.

Die QR-Codes bieten unter anderem die Möglichkeit Webadressen, Telefonnummer, SMS und freie Texte zu kodieren. Im Rahmen des Marketings kann ein QR-Code ein bestimmtes Logo enthalten. Dies „personalisiert“ den QR-Code. Durch das Hinzufügen eines Logo von z.B. einem Unternehmen, kann dies gleichzeitig für Werbezwecke eingesetzt werden. Die Einarbeitung eines Logos in den Code ist nicht einfach. Es muss darauf geachtet werden, dass die quadratischen Elemente bestehen bleiben, da der Code andernfalls schlecht lesbar und somit fehleranfällig ist. Das Einfügen eines Logos verlangt somit eine gute Kenntnis des quadratischen Aufbaus des Codes.

„So wie jede Person den QR-Code mithilfe von Handys, Smartphones oder Tablets (und deren Kamera und Reader) einscannen und auslesen kann, kann sie auch ihren eigenen produzieren. Voraussetzung hierfür ist ein Generator, der als Webanwendung und lokal installierbare Anwendung für den Computer, das Handy und das Smartphone verfügbar ist.“ ***Wirtschaftslexikon*** *[Online]*

Viele dieser Generatoren sind über das Internet frei verfügbar. Manche Generatoren ermöglichen es bereits entsprechende Logos mit einzuarbeiten. Die Generatoren bieten vielseitige Möglichkeiten um QR-Codes zu erstellen.

Hierbei muss darauf geachtet werden, für welches Einsatzgebiet der QR-Code verwendet werden soll. Es gibt lizenzrechtliche Einschränkungen, je nach dem, ob der Code kommerziell oder nicht kommerziell verwendet werden soll. Dies ist ein Hauptaspekt, auf welchen vor der Auswahl des benötigten Generators, geachtet werden muss.

Im Folgenden sind einige Generatoren und deren Webadressen aufgelistet.

QR-Code Monkey:	***QR Code Monkey*** *[Online]*
QRCode-Generator:	***QR Code Generator*** *[Online]*
the-qrcode-generator:	***The QR Code Generator*** *[Online]*
qrcode.kaywa:	***QR Code Kaywa*** *[Online]*
Rotacode:	***Rotacode*** *[Online]*

Aufgrund der oben genannten Aspekte wurde der QR-Code Monkey als der am besten geeignetste ausgewählt. . Abbildung 27 zeigt das Logo von QR-Code Monkey. Dieser QR-Code Generator bietet eine kostenlose Erstellung des Codes. Außerdem sind die erstellten QR-Codes frei und kommerziell nutzbar. Der Generator bietet die Möglichkeit den Code individuell durch den Einsatz eines Logos oder farblicher Gestaltung zu verändern. Die erstellten QR-Codes werden in Druckqualität „geliefert“.

Abbildung 27: Logo QR Code Monkey

„Der QR-Code-Generator zum Erstellen individueller QR-Codes mit Logo, Farbe und Druck-AuflösungAlle generierten QR-Codes sind kostenlos und kommerziell nutzbar. Über Weiterempfehlungen oder Verlinkungen würden wir uns natürlich freuen. Danke!“

QR Code Monkey *[Online]*

Der Generator QR-Code Monkey ermöglicht es statische und dynamische QR-Codes zu erstellen. Ein statischer QR-Code kann nach der Erstellung nicht mehr geändert werden. Bei dynamischen QR-Codes kann das verlinkte Objekt im Nachhinein geändert werden. Ist z.B. eine Webadresse mit dem QR-Code verlinkt, besteht bei einem dynamischen Code die Möglichkeit, die URL zu ändern. Ein dynamischer QR-Code muss nicht neu erstellt und gedruckt werden, wenn sich der Weblink ändert. Weiterhin bieten dynamische QR-Codes die Möglichkeit Scan-Statistiken zu analysieren. Mit Hilfe eines dynamischen Codes werden Nutzungsstatistiken nach z.B. Anzahl, Zeitpunkt und Ort der Zugriffe erstellt. Hierfür wird im Gegensatz zu statischen Codes eine Weiterleitungs-URL verwendet. Diese verlinkt wiederum auf den eigentlichen verschlüsselten Inhalt.

Mit Hilfe der dynamischen Codes werden dem Anbieter der Inhalte somit hilfreiche Statistiken bereitgestellt um die von Ihm eingesetzten QR-Codes zu verwalten und z.B. den Ort für Werbemaßnahmen zu optimieren.

Ein Einsatzgebiet, das in dieser Master-Thesis ebenfalls verwendet werden soll, ist das „Mobile Tagging“.

Tagging bedeutet in diesem Zusammenhang, dass durch das Scannen des QR-Codes, physikalische Objekte mit zusätzlichen Informationen verbunden werden. Wird, z.B. ein QR-Code auf einem Werbeplakat gedruckt, ist es dem Anwender möglich, weitere Informationen über das beworbene Produkt zu erhalten. Eine Unterteilung wird in „Commercial Tagging“, „Public Tagging“ und „Private Tagging“, also in kommerziell, nichtkommerziell (öffentlich) und private Anwendungen vorgenommen.

Bei der Nutzung der QR-Code Technologie gibt einige Risiken, welche beachtet werden sollten. Grundsätzlich ist es kaum möglich vor dem eigentlichen Scannen des Codes zu erkennen, welche Art von Anwendung oder Link dahinter steckt. Der Anwender sieht nicht, was der QR-Code enthält. Dies bietet vor allem die Möglichkeit auf kostenpflichtige Webadressen zu verweisen. Des Weiteren können QR-Code Reader und QR-Code Generatoren dazu missbraucht werden personenbezogene Daten des Anwenders zu speichern.

Die QR-Code Technologie ist nicht vor „Vandalismus“ geschützt. Bestehende QR-Code, z.B. auf Plakaten, können durch Dritte verändert werden. Außerdem können über den eigentlichen QR-Codes neue, verfälschte Codes angebracht werden. Somit besteht die Möglichkeit die abgebildeten Codes so zu verändern, dass diese auf eine andere, als die vorgesehene, Quelle verweisen und somit dem Anwender ein Schaden entsteht.

4 Möglichkeiten und Varianten der App-Entwicklung

Das Kapitel Möglichkeiten und Varianten der App-Entwicklung geht auf verschiedene Möglichkeiten ein, welche zur App-Entwicklung verwendet werden können. Die jeweiligen Anforderungen werden erläutert. Außerdem werden die verschiedenen Varianten, welche zur Umsetzung des Prototypen geeignet sind genannt, analysiert und gegenübergestellt. Im Rahmen dieser Arbeit werden lediglich Möglichkeiten betrachtet, welche sich zur Entwicklung für eine native Anwendung auf Basis des mobilen Betriebssystems Android eignen.

Es gibt verschiedene Varianten um eine Anwendung für Android zu entwickeln und zu programmieren. Diese Varianten unterscheiden sich vor allem anhand ihres Entwicklungsaufwands und den Entwicklungskosten.

Im Folgenden werden drei unterschiedliche Varianten zur App-Entwicklung erläutert:

- Online Baukästen,
- Entwicklungsumgebungen zur Erstellung von Anwendungen (MIT App Inventor[1]),
- reine Programmierung.

Online Baukästen

Online Baukästen sind der einfachste Weg der App-Entwicklung. Die Baukästen bieten eine grafische Entwicklungsumgebung. Dem Entwickler werden verschiedene vorgefertigte Layouts angeboten, welche selbstständig angepasst werden. Das Layout und die Funktionen der einzelnen Bauteile werden vorgefertigt bereitgestellt. Der Entwickler muss lediglich den Inhalt der Anwendung anpassen. Die Funktionen und der dahinterstehende Programmcode werden vom Online Baukasten umgesetzt. Die erfolgreichsten Anbieter von Online Baukästen, zum Zeitpunkt der Anfertigung dieser Arbeit, sind AppYourself, iBuildApp und appTITAN. Diese Baukästen bieten nahezu denselben Funktionsumfang und eine ähnliche Kostengestaltung. Da es für die Entwicklung einer Anwendung unabdingbar ist den Programmcode einsehen zu können wählen Entwickler meist die teuersten Varianten der Online Baukästen. Nur in diesen Varianten ist es möglich, den Programmiercode einzusehen.

[1] Der MIT App Inventor ist ein Projekt des Massachusetts Institute of Technology. Dieses bietet eine Entwicklungsumgebung für die Erstellung von Anwendungen auf Basis des mobilen Betriebssystems Android.

Die Preise der oben genannten Online Baukästen variieren zwischen monatlichen und einmaligen Kosten. Für die Baukästen AppYourself und iBuildApp werden monatliche Kosten berechnet. Diese liegen zwischen 39,90 Euro und 45 Euro. ***Appyourself*** *[Online],* ***ibuildapp*** *[Online]*

Für den Online Baukasten appTITAN werden einmalige Kosten von 444 Euro berechnet. Anschließend wird dem Entwickler die Möglichkeit gegeben seine Anwendung über den Baukasten zu hosten[1] und zu veröffentlichen. Hierfür fallen monatliche Kosten von 9,99 Euro an. ***appTITAN*** *[Online]*

Diese Variante der App-Entwicklung ist vor allem für unerfahrene Entwickler geeignet, die schnell zu einem Ergebnis kommen möchten. Durch die teilweise hohen Kosten und die Einschränkungen betreffend des Layouts und des Funktionsumfangs der Anwendungen sind Online Baukästen nicht für die Entwicklung jeder Anwendung geeignet. Der Entwickler muss vor der Erstellung analysieren welche Eigenschaften die Anwendung erfüllen soll und danach entscheiden, ob eine Umsetzung mit Hilfe eines Online-Baukastens in Frage kommt.

Entwicklungsumgebungen zur Erstellung von Anwendungen (MIT App Inventor)

Eine weitere Variante um Anwendungen auf Basis des mobilen Betriebssystems Android zu entwickeln sind Entwicklungsumgebungen. Im Rahmen dieser Arbeit wird auf die Entwicklungsumgebung App Inventor des Massachusetts Institute of Technology (MIT) eingegangen. App Inventor ist ein Projekt, welches von Google Inc. ins Leben gerufen wurde. Nachdem Google das kostenfreie Projekt eingestellt hat, wurde dieses durch das MIT übernommen. Das MIT entwickelt das Entwicklungstool stätig weiter. ***Inventor, M. A****. [Online]*

„Seit Dezember 2010 steht das von Google bereitgestellte Entwicklungstool für jeden kostenfrei zur Verfügung. Mit App Inventor können Sie Ihre eigenen Apps entwickeln, auch wenn Sie nie zuvor auf einem Computer oder gar einem Smartphone programmiert haben. Denn mit App Inventor lassen sich mit spielerischer Leichtigkeit kleine und auch richtig große Apps aus visuellen Bausteinen zusammensetzen, ohne eine einzige Zeile Java-Code. Dabei handelt es sich keinesfalls um eine Spielerei, sondern um ein alternatives und innovatives Entwicklungstool, mit dem Sie schnell und einfach auch komplexe und anspruchsvolle Apps für sich selbst und andere entwickeln können.“

Kloss, J. H. *[Buch]*

[1] hosten ist ein anderes Wort für veröffentlichen

"MIT App Inventor is a blocks-based programming tool that allows everyone, even novices, to start programming and build fully functional apps for Android devices. Newcomers to App Inventor can have their first app up and running in an hour or less, and can program more complex apps in significantly less time than with more traditional, text-based languages. Initially developed by Professor Hal Abelson and a team from Google Education while Hal was on sabbatical at Google, App Inventor runs as a Web service administered by staff at MIT's Center for Mobile Learning - a collaboration of MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) and the MIT Media Lab. MIT App Inventor supports a worldwide community of nearly two million users representing 195 countries worldwide. The tool's more than 85 thousand active weekly users have built more than 4.7 million android apps. An open-source tool that seeks to make both programming and app creation accessible to a wide range of audiences." ***Inventor, M. A****. [Online]*

Das Entwicklungstool zeichnet sich vor allem durch seinen Aufbau aus. Es handelt sich um eine grafische Entwicklungsumgebung mit deren Hilfe die Funktionalitäten und der dazugehörige Programmcode zusammengestellt werden können. Hierbei verwendet App Inventor eine Art Baukastenprinzip. Der Programmcode wird mit Hilfe von einzelnen Puzzleteilen zusammengesetzt. Dies garantiert erfahrenen Entwicklern wie auch Anfängern eine schnelle Einarbeitung und schnelle Erfolge. Das Entwicklungstool funktioniert nach dem „WYSIWYG-Prinzip"[1].

Der MIT App Inventor ist ein kostenfreies Entwicklungstool. Zahlreiche Bücher und eine große Community erleichtern den Einstieg in die App-Entwicklung. Für die Entwicklung einer Anwendung stehen grafische Darstellungen zur Verfügung welche durch die Code-Puzzleteile ihre Funktion erlangen. Der MIT App Inventor bietet vollständigen Zugriff auf die API des mobilen Endgeräts.

[1] WYSIWYG steht für den englischen Ausdruck „What you see is what you get". Dies bedeutet, dass der Entwickler schon während der Entwicklung die momentanen Fortschritte grafisch sehen kann.

Reine Programmierung

Die dritte Variante der App-Entwicklung ist die reine, klassische Programmierung. Diese Variante eignet sich vor allem für erfahrene Programmierer. Die Programmierung setzt gute Programmierkenntnisse voraus. Bei der reinen Programmierung wird die komplette Anwendung mit Hilfe einer Programmiersprache entwickelt. Anwendungen auf Basis des mobilen Betriebssystems Android werden mit Hilfe der Programmiersprache Java umgesetzt. Java ist eine objektorientierte, plattformunabhängige Programmiersprache. Der Aufbau (Syntax) von Java lehnt sich an C++ an. Zur Entwicklung von Applikationen wird außerdem die Java Entwicklungsumgebung Java SDK (Java Software Development Kit) benötigt. Dieses Kit umfasst die Laufzeitumgebung (JRE[1]) und beinhaltet zusätzlich Compiler[2], Debugger[3] und Dokumentationswerkzeuge.

Für die Entwicklung von Android Apps wird zusätzlich die Android-SDK (Android Software Development Kit) benötigt. Dieses Kit beinhaltet einige Anwendungen welche vom Betriebssystem bereitgestellt werden. Diese sind unter anderem Browser, Google Maps, SMS und Mail, Adressverwaltung, Musikplayer, Kamera und Fotogalerie. Die Android SDK stellt die benötigten API Bibliotheken und Entwicklungswerkzeuge zur Verfügung, welche für die Programmierung von Applikationen nützlich sind.

Die reine Programmierung unterliegt keinerlei Beschränkungen welche eingehalten werden müssen. Die Programmierung steht dem Entwickler völlig offen. Diese Variante der Entwicklung eignet sich somit aber vor allem für erfahrene Entwickler, welche im Umgang mit Java sicher sind.

[1] JRE = Java Runtime Environment
[2] Ein Compiler ist ein Programm welches ein Programm das mit einer bestimmten Programmiersprache entwickelt wurde, in eine Form übersetzt, welche von einem Computer ausgeführt werden kann
[3] Ein Debugger ist ein Werkzeug zum Auffinden von Fehlern in einer Anwendung.

5 Erstellung des Konzepts

Das folgende Kapitel beschreibt die Erstellung des Konzepts ein. Der erste Konzeptentwurf wird veranschaulicht und erläutert. Der grafische Entwurf bildet die Grundlage für das Feinlayout. Das Feinlayout zeigt, wie der Prototyp grafisch umgesetzt werden soll.

5.1 Anforderungskatalog

„Ein Anforderungskatalog im Sinne der DIN 69905 ist die "Auflistung von Anforderungen, durch deren Erfüllung ein angestrebtes Projektziel erreicht werden soll." Ein Anforderungskatalog kann beispielsweise Bestandteil des Lastenheftes oder der Abnahmevereinbarung sein. Aber auch durch Qualitätssicherung oder Marktforschung entstehen Anforderungskataloge, die zur Erreichung des Projektzieles zu erfüllen sind.In gewisser Weise kann der Anforderungskatalog im Sinne der DIN als eine detaillierte Darstellung des Projektziels betrachtet werden.“ ***Projektmagazin*** *[Online]*

Im Anforderungskatalog werden technische und funktionale Anforderungen an die Anwendung festgehalten. Durch die Definition verschiedener Kriterien kann der Anforderungskatalog bei verschiedenen Analysen hilfreich sein. Außerdem dient der Anforderungskatalog der späteren Bewertung der Ergebnisse.

In der folgenden Tabelle 7 wird der Anforderungskatalog für die in der vorliegenden Arbeit zu erstellende Anwendung zusammengestellt.

Anforderungen an die Anwendung	Beschreibung
Entwicklungsaufwand	Der Entwicklungsaufwand muss aus Zeitgründen gering gehalten werden.
Kompatibilität/ Verfügbarkeit	Die Anwendung soll möglichst kompatibel auf mehreren mobilen Betriebssystemen zum Einsatz kommen.
Performance/ Zuverlässigkeit	Die Performance der Anwendung soll möglichst hoch sein. Dies betrifft lange Ladezeiten etc.. Außerdem soll die Anwendung stabil laufen. Abstürze der Anwendung sollen vermieden werden.
Wartung/ Erweiterbarkeit	Der Wartungsaufwand soll niedrig und einfach gehalten werden. Es muss möglich sein neue Inhalte einfach, schnell und ohne Programmierkenntnisse in die Anwendung einfügen zu können.
Datenschutz/ Verschlüsselung	Falls die Anwendung personenbezogene Daten enthält, sollen diese für Dritte unzugänglich sein. Außerdem müssen sämtliche datenschutzrechtlichen Daten verschlüsselt sein.

Informationsübermittlung/ Bedienbarkeit	Die Anwendung muss einfach zu bedienen sein. Jeder Nutzer muss ohne vorherige Kenntnisse in der Lage sein, die Anwendung intuitiv zu bedienen. Für eine anwenderfreundliche App müssen die zu übermittelnden Informationen aufbereitet werden.
Zielgruppe	Die Zielgruppe muss klar definiert sein. Die Zielgruppe der Anwendung sind Smartphoneuser welche sich über die Trinkwasserversorgung informieren möchten. Hiermit ist die Zielgruppe von Schülern bis zu älteren Benutzern gefächert.
Lizenz/Kosten	Die Lizenz der Anwendung soll frei sein. Die Anwendung soll informativen und werbetechnischen Gründen dienen. Hierfür muss diese frei erhältlich für die Nutzer sein.
Kartendienst	Die Anwendung muss einen zuverlässigen und bekannten Kartendienst beinhalten.
QR-Code	Die Anwendung muss einen QR-Code-Reader enthalten.
Anforderungen an die Software:	
Zeitaufwand	Der Entwicklungsaufwand muss aus Zeitgründen gering gehalten werden.
Lernkurve	Die Lernkurve zum Erlenen der Software soll steil ansteigen. Der Umgang mit der Software muss schnell erlernt werden können.
Performance/ Zuverlässigkeit	Die Performance der Software soll möglichst hoch sein. Dies betrifft lange Ladezeiten etc.. Außerdem soll die Software stabil (ohne Abstürze) laufen.
Kosten	Die Kosten für die Anschaffung der Software sollen gering gehalten werden. Hier empfiehlt sich wenn möglich ein Open Source Produkt.
Datenschutz	Die Software soll die eingebundenen Daten nicht an den Softwareentwickler oder Dritte weitergeben.
Erweiterbarkeit	Die Software soll Erweiterungsmöglichkeiten bieten. Dies beinhaltet weitere Updates der Software und mögliche Add-ons.
Gestaltung	Die Anwendung soll frei gestaltet werden können. Die Software soll keinerlei gestalterischen Vorgaben/Grenzen geben.

Zugriff auf API des mobilen Betriebssystems	Die Software soll den Zugriff auf das mobile Betriebssystem unterstützen. Sensoren des Smartphones sollen eingebunden werden können.
Anforderungen an das Betriebssystem:	
Offener Quellcode/ Programmcode	Der Quellcode des Betriebssystems soll aus Programmierungsgründen entweder komplett oder in Teilstücken frei erhältlich sein.
Performance/ Zuverlässigkeit	Die Performance des Betriebssystems soll möglichst hoch sein. Dies betrifft lange Ladezeiten etc.. Außerdem soll das Betriebssystem stabil (ohne Abstürze) laufen.
Kosten	Die Kosten für die Entwicklung von Anwendungen für das spezielle Betriebssystem sollen gering gehalten werden. Dies betrifft vor allem die Kosten für die spätere Verbreitung der Anwendung (App Store) und die Einschränkungen der Anwendungen des Betriebssystemherstellers.
Synchronisation	Die Synchronisation zwischen dem Endgerät und dem Betriebssystem soll einfach z.B. über das Internet geschehen.
Aktualisierung	Das Betriebssystem soll regelmäßig aktualisiert werden. Dies garantiert ebenfalls eine breite Verfügbarkeit des Systems durch die Endnutzer.

Tabelle 7: Anforderungskatalog

5.2 Vergleich der grundlegenden Eigenschaften mit dem Anforderungskatalog

Der Anforderungskatalog ist die Grundlage für die Entscheidungen betreffend des App-Typen, dem mobilen Betriebssystem und der Umsetzungstechnik für den Prototypen. Dieses Kapitel beschreibt anhand des Anforderungskatalogs die grundlegenden Entscheidungen welche getroffen werden müssen bevor der Prototyp technisch umgesetzt wird.

Die folgende Tabelle 8 veranschaulicht, welche App-Typen die Anforderungen an die Anwendung aus dem Anforderungskatalog erfüllen. Hierfür wird für jeden Typ der Anfangsbuchstaben verwendet. Dieser zeigt, ob die Anforderungen erfüllt werden. (Native App = N, Web App = W, Hybrid App= H)

Anforderungen an die Anwendungen	Beschreibung	erfüllt die Anforderungen
Entwicklungsaufwand	Der Entwicklungsaufwand muss aus Zeitgründen gering gehalten werden.	N,W
Kompatibilität/ Verfügbarkeit	Die Anwendung soll möglichst kompatibel auf mehreren mobilen Betriebssystemen zum Einsatz kommen.	W
Performance/ Zuverlässigkeit	Die Performance der Anwendung soll möglichst hoch sein. Dies betrifft lange Ladezeiten etc.. Außerdem soll die Anwendung stabil laufen. Abstürze der Anwendung sollen vermieden werden.	N
Wartung/ Erweiterbarkeit	Der Wartungsaufwand soll niedrig und einfach gehalten werden. Es muss möglich sein neue Inhalte einfach, schnell und ohne Programmierkenntnisse in die Anwendung einfügen zu können.	N, W
Datenschutz/ Verschlüsselung	Falls die Anwendung personenbezogene Daten enthält. Sollen diese für Dritte unzugänglich sein. Außerdem müssen sämtliche datenschutzrechtlichen Daten verschlüsselt sein.	N, W, H

Informationsübermittlung/ Bedienbarkeit	Die Anwendung muss einfach zu bedienen sein. Jeder Nutzer muss ohne vorherige Kenntnisse in der Lage sein, die Anwendung intuitiv zu bedienen. Für eine anwenderfreundliche App müssen die zu übermittelnden Informationen aufbereitet werden.	N, W, H
Zielgruppe	Die Zielgruppe muss klar definiert sein. Die Zielgruppe der Anwendung sind Smartphone Anwender welche sich über die Trinkwasserversorgung informieren möchten. Hiermit ist die Zielgruppe von Schülern bis zu älteren Benutzern gefächert.	N, W, H
Lizenz/Kosten	Die Lizenz der Anwendung soll frei sein. Die Anwendung soll informativen und werbetechnischen Gründen dienen. Hierfür muss diese frei erhältlich für die Nutzer sein.	N, W, H
Kartendienst	Die Anwendung muss einen zuverlässigen und bekannten Kartendienst beinhalten.	N, W
QR-Code	Die Anwendung muss einen QR-Code-Reader enthalten.	N, H

Tabelle 8: Vergleich Anforderungskatalog mit den App-Typen

Fazit

Nach dem Vergleich der unterschiedlichen Typen von Anwendungen wird klar, dass jeder Typ seine entsprechende Vor- und Nachteile aufweist. Aus diesem Grund ist es wichtig vor jedem Projektstart zu analysieren welche Anforderungen an die spätere Anwendung gestellt werden. Im Rahmen dieser Masterthesis bieten sich zwei Möglichkeiten an. Die Anwendung könnte anhand einer nativen App oder einer Web App umgesetzt werden. Der Vorteil einer Web App ist die Plattformunabhängigkeit. Betrachtet man jedoch die Funktionen dieser zwei Typen wird deutlich, dass die Web App nicht auf sämtliche Funktionen (API) eines Smartphones zugreifen kann. Ein Hauptteil der Anwendung welche entwickelt werden soll, ist die Fähigkeit QR-Codes zu scannen und umzusetzen. Für das Einscannen der Codes wird die Kamerakomponente des Smartphones verwendet. Web Apps bieten keinen direkten Zugriff auf die Kamerafunktion der Smartphones.

Aus diesen Gründen wird, die im Rahmen der Masterthesis umzusetzende Anwendung, als native App entwickelt. Dieser Anwendungstyp wird den meisten Anforderungen des Anforderungskataloges gerecht.

Die folgende Tabelle 9 veranschaulicht, welche Anforderungen an das Betriebssystem aus dem Anforderungskatalog erfüllt werden. Hierfür wird für jedes Betriebssystem ein Buchstabenkürzel verwendet. Dieses zeigt, ob die Anforderungen erfüllt werden. (Apple iOS = iOS, Windows Phone = WP, Android = A, Bada = B, Blackberry-OS = Bos)

Anforderungen an das Betriebssystem	Beschreibung	Erfüllt die Anforderungen
Offener Quellcode/ Programmcode	Der Quellcode des Betriebssystems soll aus Programmierungsgründen entweder komplett oder in Teilstücken frei erhältlich sein.	A
Performance/ Zuverlässigkeit	Die Performance des Betriebssystems soll möglichst hoch sein. Dies betrifft lange Ladezeiten etc.. Außerdem soll das Betriebssystem stabil (ohne Abstürze) laufen.	iOS, WP, A
Kosten	Die Kosten für die Entwicklung von Anwendungen für das spezielle Betriebssystem sollen gering gehalten werden. Dies betrifft vor allem die Kosten für die spätere Verbreitung der Anwendung (App Store) und die Einschränkungen der Anwendungen des Betriebssystemherstellers.	WP, A, B, Bos
Synchronisation	Die Synchronisation zwischen dem Endgerät und dem Betriebssystem soll einfach z.B. über das Internet geschehen.	iOS, WP, A
Aktualisierung	Das Betriebssystem soll regelmäßig aktualisiert werden. Dies garantiert ebenfalls eine breite Verfügbarkeit des Systems durch die Endnutzer.	iOS, WP, A

Tabelle 9: Vergleich Anforderungskatalog mit mobilen Betriebssystemen

Fazit

Die native Anwendung soll mit einem möglichst geringen Zeit- und Kostenaufwand umgesetzt werden. Außerdem soll der Großteil der Smartphone Anwender in Deutschland angesprochen werden. Die wichtigste Anforderung an das mobile Betriebssystem ist die Verwendung von Open Source Komponenten[1]. Dies schließt einen offenen Quellcode des mobilen Betriebssystems ein. Nach dem Vergleich der unterschiedlichen mobilen Betriebssysteme fällt auf, dass der Programmcode lediglich bei einem Betriebssystem frei verfügbar ist. Da sich die Betriebssysteme hinsichtlich ihrer Kosten und sonstigen Anforderungen des Anforderungskatalogs nur gering oder gar nicht unterscheiden, wird die zu entwickelnde Anwendung auf Basis des mobilen Betriebssystems Andorid umgesetzt.

Die folgende Tabelle 10 zeigt, welche Varianten der App-Entwicklung die Anforderungen an die Software aus dem Anforderungskatalog erfüllen. Hierfür wird für jede Variante ein Buchstabenkürzel verwendet. Dieses zeigt, ob die Anforderungen erfüllt werden. (Online Baukästen = OB, MIT App Inventor = MIT, Reine Programmierung = RP)

Anforderungen an die Software	Beschreibung	Erfüllt die Anforderungen
Zeitaufwand	Der Entwicklungsaufwand soll aus Zeitgründen gering gehalten werden.	OB, MIT
Lernkurve	Die Lernkurve zum Erlenen der Software soll steil ansteigen. Der Umgang mit der Software muss schnell erlernt werden können.	OB, MIT
Performance/ Zuverlässigkeit	Die Performance der Software soll möglichst hoch sein. Dies betrifft lange Ladezeiten etc. Außerdem soll die Software stabil (ohne Abstürze) laufen.	OB, MIT, RP
Kosten	Die Kosten für die Anschaffung der Software sollen gering gehalten werden. Hier empfiehlt sich wenn möglich ein Open Source Produkt.	MIT, RP
Datenschutz	Die Software soll die eingebundenen Daten nicht an den Softwareentwickler oder Dritte weitergeben.	MIT, RP

[1] Open Source Komponenten = Komponenten welche laut ihren Lizenzbestimmungen für die Öffentlichkeit frei und kostenlos verfügbar sind

Erweiterbarkeit	Die Software soll Erweiterungsmöglichkeiten bieten. Dies beinhaltet weitere Updates der Software und mögliche Add-ons.	OB, MIT, RP
Gestaltung	Die Anwendung soll frei gestaltet werden können. Die Software soll keinerlei gestalterischen Vorgaben/Grenzen geben.	MIT, RP
Zugriff auf API des mobilen Betriebssystems	Die Software soll den Zugriff auf das mobile Betriebssystem unterstützen. Sensoren des Smartphones sollen eingebunden werden können.	OB (bedingt), MIT, RP

Tabelle 10: Vergleich Anforderungskatalog mit den verschiedenen Entwicklungsvarianten

Fazit

Lediglich die Entwicklungsumgebung MIT App Inventor wie auch die reine Programmierung kommen für den umzusetzenden Prototypen in Frage. Dies liegt vor allem daran, dass diese Varianten ohne Kosten verwendet werden können. Die Entwicklung mit Hilfe eines Online Baukastens ist in diesem Rahmen zu kostenintensiv und unübersichtlich, da sich die Kosten schwer eingrenzen lassen.

Um die native Anwendung auf Basis des mobilen Betriebssystems Android, im Rahmen dieser Arbeit, umzusetzen wird, aus den oben genannten Gründen, das Entwicklungstool MIT App Inventor eingesetzt. Dieses Entwicklungstool ermöglicht schnelle Erfolge bei der App-Entwicklung. Obwohl App Inventor ohne Programmierkenntnisse verwendet werden kann ist es sinnvoll Grundlagen in der Programmierung aufzuweisen. Die Software lässt sich dann relativ schnell erlernen und läuft zuverlässig. Da das Entwicklungstool frei verfügbar ist, liegt die Verwendung im Rahmen dieser Arbeit nahe. Anwendungen welche mit MIT App Inventor umgesetzt werden, können jederzeit erweitert und bearbeitet werden. Der Programmcode wird selbst zusammengestellt und ist deshalb offen und einsehbar. Die größten Vorteile dieses Entwicklungstool liegen in der freien Gestaltung der späteren Anwendung und im Zugriff auf sämtliche API, welche das mobile Endgerät bereitstellt. Der Zugriff auf die Kamerakomponente ist, wegen des Scannens der QR-Codes, unabdingbar für diese Arbeit. Ein weiterer wichtiger Punkt ist, dass der MIT App Inventor stätig weiterentwickelt und verbessert wird. Außerdem wird zahlreiche Literatur für die Entwicklung angeboten. Das Tool verfügt daher über eine gute Dokumentation und eine große Community an Entwicklern, welche diese Entwicklungsumgebung für deren Anwendungen verwenden.

5.3 Konzeptentwurf

In diesem Kapitel wird auf den ersten Konzeptentwurf eingegangen. Dieser umfasst den Aufbau und die Funktionen der Anwendung.

Nachfolgend wird dargestellt welche Menüpunkte in der Anwendung enthalten sind. Die vier Hauptbuttons Home, Info, Karte und QR-Code stehen dem Nutzer dauerhaft zur Navigation zur Verfügung. Hierdurch wird gewährleistet, dass der Nutzer durch die Hauptpunkte navigieren kann und somit eine bessere Übersicht und eine schnellere Bedienung erhält. Durch die Hauptbuttons stehen sämtliche Funktionen in einem Zusammenhang. Der Nutzer kann von jeder Information in eine andere wechseln. Die aufgeführten Unterpunkte zeigen die jeweiligen Untermenüs der Hauptfunktionen und deren Inhalt. Abbildung 28 zeigt den strukturellen Aufbau der Anwendung in einem Baumdiagramm.

Die Karlsruher Trinkwasserversorgung

- HOME
  - Zu den Inhalten

    Inhalt: Verlinkung zum Bereich INHALT um einen schneller Zugriff auf die Inhalte und Infos zu gewähren;
  - Infos & Hilfe

    Inhalt: Erklärung des Aufbau und der Bedienung der Applikation und Impressum;
  - Media & Downloads

    Inhalt: Links zu Veröffentlichungen/Plakaten/Flyern etc. und Mediabereich.
- INHALT
  - Daten & Fakten zur Trinkwasserversorgung (Trinkwasserqualität)

    Inhalt: Bild, Text;
  - Chronik der Trinkwasserversorgung

    Inhalt: Bild, Text;
  - Historische Anlagen

    Inhalt: Bild, Text, Kartenbutton;
  - Wasserwerke

    Inhalt: Bild, Text, Ansprechpartner, Kartenbutton;

  - Rohrnetz

    Inhalt: Bild, Text, Ansprechpartner;

  - Hausanschluss

    Inhalt: Bild, Text, Ansprechpartner;

  - Öffentliche Brunnen

    Inhalt: Bild, Text, Kartenbutton.

- KARTE

  Inhalt: Web Viewer mit Kartendarstellung
  (Karte enthält 3 unterschiedliche Signaturen. Jede Signatur enthält den Namen des Objekts und einen Link auf das Datenblatt).

- QR

  Inhalt: Scanbutton und Webviewer zur Anzeige des Datenblatts.

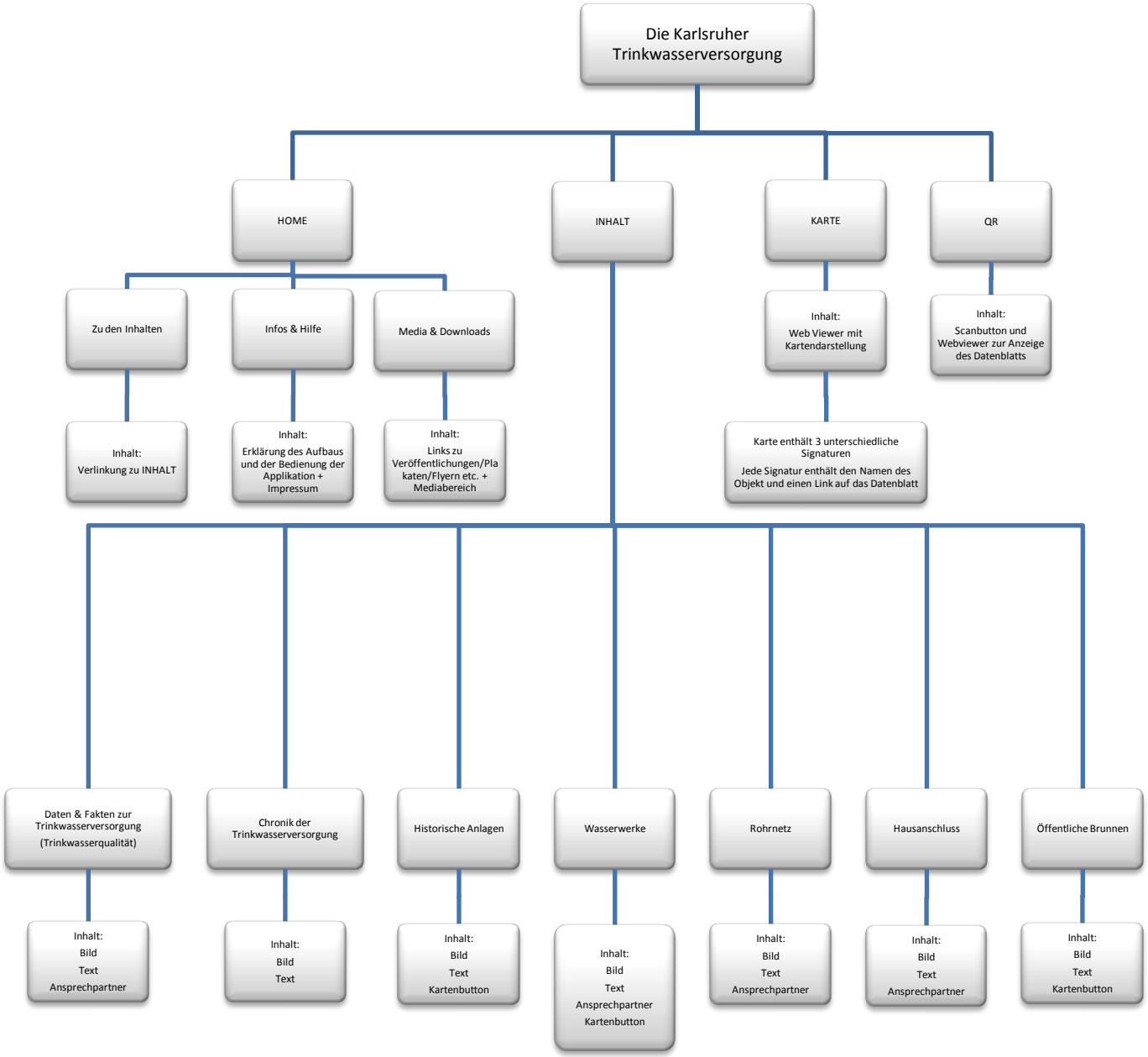


Abbildung 28: Baumdiagramm Anwendung

5.4 Grafischer Entwurf

Im vorliegenden Kapitel wird der erste grafische Entwurf vorgestellt. Dieser bildet die Grundlage für das sich anschliessende Feinlayout.

In den folgenden Skizzen wird der Aufbau der verschiedenen Screens, welche zur Umsetzung der Anwendung benötig werden, dargestellt. Diese unterscheiden sich in ihrem Aufbau und den Komponenten welche auf dem Bildschirm angeordnet sind.

Es sind „starre“ und „freie“ Komponenten vorhanden. Die starren Komponenten sind auf allen Screen enthalten. Hierbei handelt es sich um die Platzierung des Titels, der vier Hauptbuttons und den jeweiligen Freiräumen. Freie Komponente sind diejenigen, die nicht auf allen Screens vorhanden sind. Hierbei handelt es sich um die Menübuttons, das Kartenfeld, Textfelder, Bilder und zusätzliche Funktionsbuttons. Die Abbildungen 29 und 30 zeigen die grafischen Entwürfe anhand erster Skizzen.

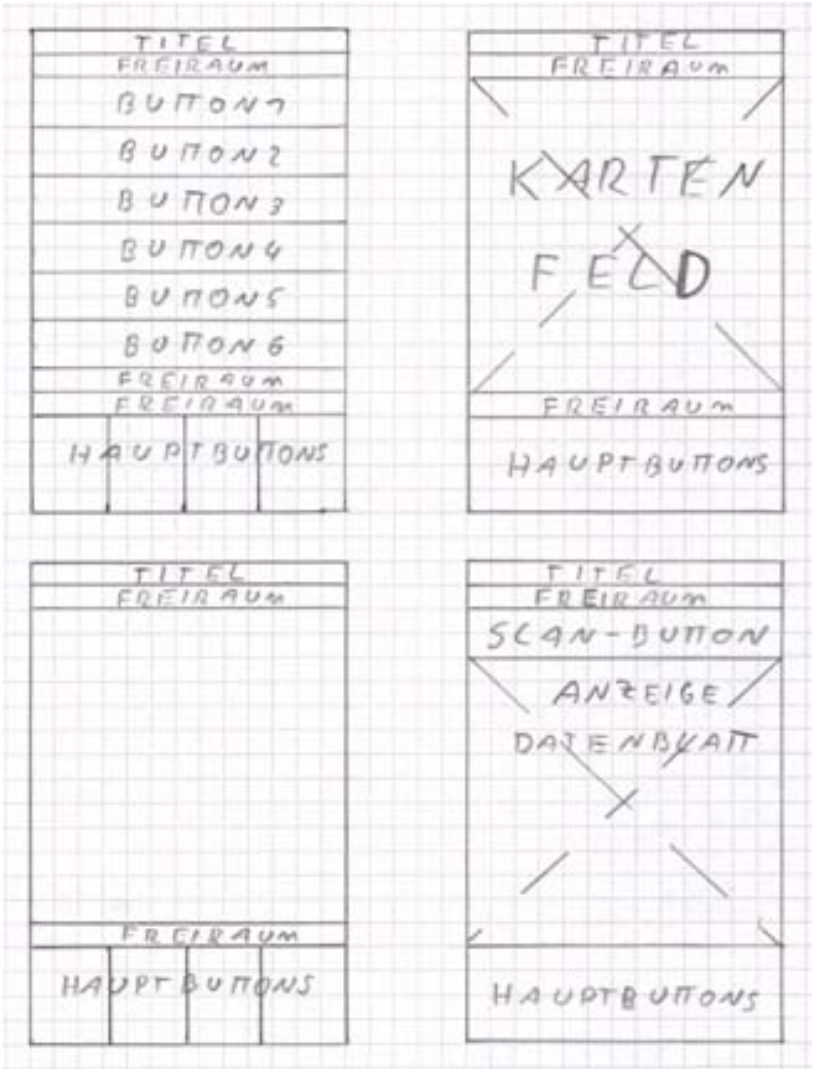


Abbildung 29: Entwurfsskizze 1

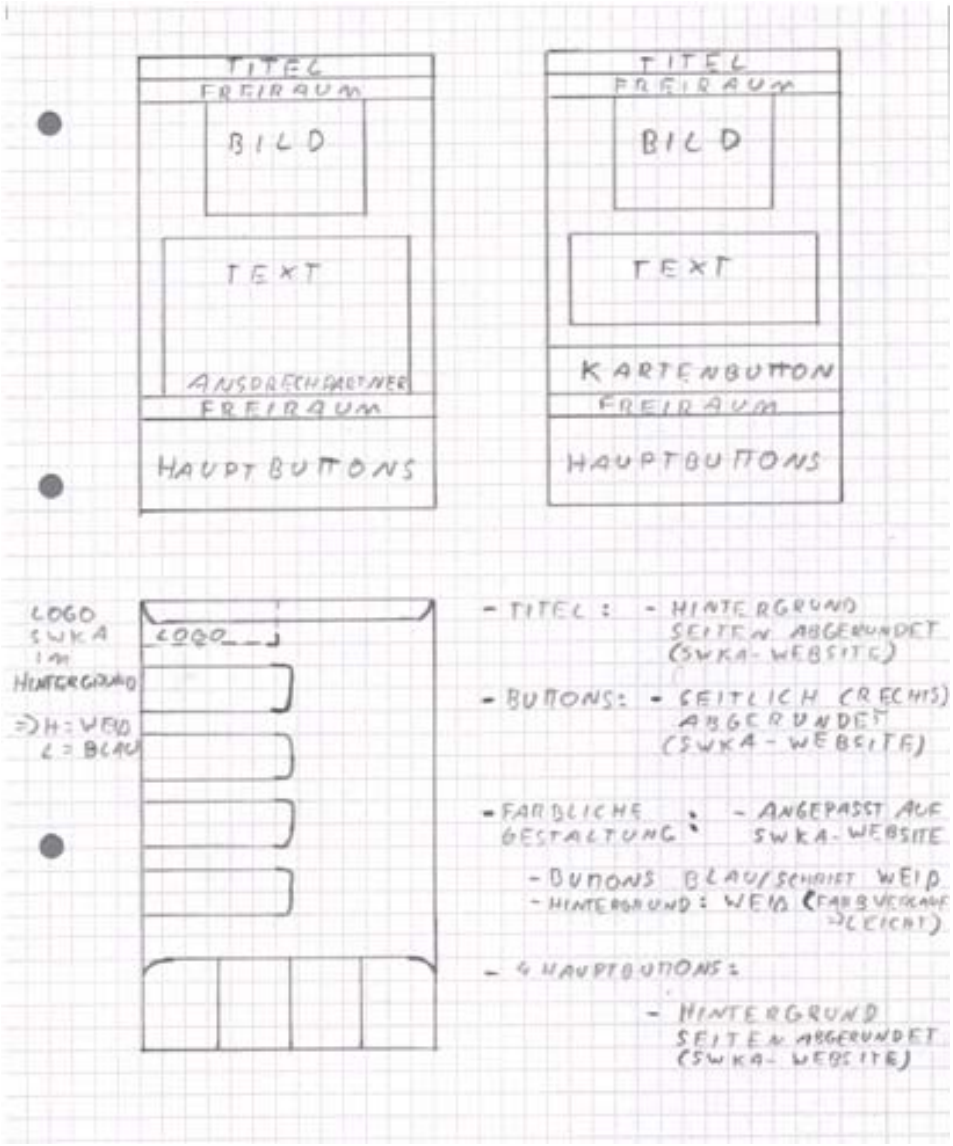


Abbildung 30: Entwurfsskizze 2

Alle Screens enthalten einen Titel, welcher am oberen Rand der Anwendung platziert ist. Dieser zeigt die aktuelle Menüauswahl. Die Seiten der Titelleiste werden unten abgerundet dargestellt. Sämtliche Buttons werden ebenfalls an der rechten Seite abgerundet dargestellt. Die vier Hauptbuttons, welche am unteren Rand der Anwendung platziert sind, werden oben abgerundet dargestellt. Die abgerundeten Elemente orientieren sich am Aufbau der Homepage der Stadtwerke Karlsruhe GmbH.

Aus diesem Grund wird die farbliche Gestaltung auf die Farben Blau und Weiß abgestimmt. Hierbei wird der blaue Farbton des Logos der Stadtwerke Karlsruhe GmbH verwendet. Die Hintergrundgestaltung beschränkt sich auf einen leichten Farbverlauf mit den Grundfarben Weiß und Grau.

5.5 Feinlayout

Auf der Grundlage des grafischen Entwurfs wird das tatsächliche Erscheinungsbild der jeweiligen Screens des Prototypen im Feinlayout erstellt.

Das Feinlayout zeigt die detaillierten grafischen Ausarbeitungen (Abbildung 31 und 32). Es gibt einen Eindruck darüber, wie der Prototyp und die spätere Anwendung tatsächlich aussehen sollen. Das Feinlayout baut auf den Skizzen aus dem grafischen Entwurf auf. Weiterhin wird in den folgenden Grafiken das Icon der Anwendung dargestellt. Dieses Icon wird später auf dem Screen des Smartphones angezeigt um die Anwendung aufzurufen. Das Feinlayout zeigt weiterhin die ausgearbeiteten Elemente und deren Anordnung in der Anwendung. Grafische Elemente wie Buttons, Titel und Hintergrundgestaltung werden detailliert dargestellt.

Abbildung 31: Feinlayout Hauptscreens

Abbildung 32: Feinlayout Unterscreens

6 Erstellung des Prototypen

Dieses Kapitel beschreibt die technische Entstehung des Prototypen. Zu Beginn wird auf die Aufbereitung der Grundlagendaten eingegangen und erläutert, auf welche Art und Weise die Informationen in der Applikation dargestellt werden. Die verwendete Software für die Umsetzung des Prototypen wird beschrieben. Das Hauptaugenmerk liegt auf der Beschreibung der technischen Umsetzung und der Beschreibung des erstellten Prototypen.

6.1 Aufbereitung der Grundlagendaten und Informationsübermittlung in der Applikation

Kapitel 6.1 beschreibt wie die in Kapitel 3.1 genannten Datengrundlagen aufbereitet werden. Es wird weiterhin erläutert welche Daten für die Umsetzung verwendet werden.

Die in Kapitel 3.1 genannten Daten bilden die Grundlage der Arbeit. Bei der Datenaufbereitung wird darauf geachtet, die wichtigsten Informationen aus der Masse der Daten herauszufiltern. Dies gilt vor allem bei der Datenaufbereitung für eine Applikation. In einer Applikation ist es wichtig dem Nutzer eine Vielzahl an Informationen schnell und kompakt zu vermitteln. Aus diesem Grund wurden einzelne Informationen aus den zu Grunde liegenden Daten herausgefiltert und so aufbereitet und zusammengefasst, dass es möglich ist, viele Daten und Informationen kompakt an den Nutzer zu vermitteln. Sämtliche Informationen stammen aus den Grundlagendaten, wie verschiedenen Büchern, Broschüren und Plakaten, welche in Kapitel 3.1 genannt wurden. Ein weiterer Schwerpunkt lag auf der Aufbereitung der verschiedenen Bildinformationen. Hierbei ist es wichtig aussagekräftige Bilder auszuwählen. Die Darstellung der Bilder in der Applikation setzt voraus, dass diese hierfür digital, hinsichtlich deren Auflösung und Größe, angepasst werden.

Der Schwerpunkt der Datenaufbereitung liegt in der Informationssuche und der kompakten Aufbereitung der Inhalte um diese applikationstauglich bereitzustellen. Sämtliche Daten wurden in Zusammenarbeit mit den Stadtwerken Karlsruhe und der EBG ausgewählt und redaktionell überarbeitet. Die Hauptauswahlkriterien richteten sich nach den oben genannten Kriterien der schnellen und kompakten Informationsübermittlung.

Ein weiterer wichtiger Punkt der Datenaufbereitung liegt in der späteren Pflege und Aktualisierung der Daten, welche in der Applikation bereitgestellt werden. Um die Daten immer aktuell halten zu können ist es wichtig, dass diese auch ohne zusätzliche Kenntnisse bezogen auf die App-Entwicklung aktualisiert werden können. Die Daten müssen ohne Programmierkenntnisse und großen Aufwand aktualisierbar sein. Aus diesem Grund werden die Daten und Informationen im Rahmen dieser Arbeit in einer HTML[1]-Umgebung aufbereitet.

[1] HTML steht für Hypertext Markup Language, eine strukturierende Auszeichnungssprache, welche die Grundlage für das World Wide Web bildet und von Webbrowsern dargestellt wird.

Diese HTML-Seiten werden dann mit Hilfe eines Web-Viewers in der Applikation angezeigt. Die Umsetzung dieses Konzepts garantiert eine einfache Pflege und Aktualisierung der Daten.

Außerdem können die Aktualisierungen bei der gewählten Vorgehensweise ohne Programmierkenntnisse im App-Bereich mit Hilfe eines Editors oder Web-Editors durchgeführt werden.

Das Folgende Beispiel in Abbildung 33 zeigt den Aufbau einer HTML-Seite anhand des Beispiels „Wasserturm Bergwald“:

Wasserturm Bergwald

Der Wasserturm wurde zur Versorgung der Bergwaldsiedlung im Jahr 1967 erstellt. Er enthält zwei je 200 Kubikmeter fassende Ringkammern im oberen Teil des Turmes, sowie zwei je 20 Kubikmeter fassende Behälter im Fuß des Turmes. Der obere Wasserspiegel liegt bei NN+ 292 Meter, beziehungsweise 33,30 Meter über Gelände.

Abbildung 33: Beispiel Wasserturm

Die HTML-Umgebung spiegelt sich im Programmiercode der einzelnen HTML-Seite wieder, wobei diese immer gleich aufgebaut sind. Sie bestehen aus einem „Rahmen“, dem Stylesheet und dem Informationstext. Im Folgenden wird der Programmiercode der HTML-Umgebung ebenfalls am Beispiel „Wasserturm Bergwald“ dargestellt.

```
<body>
  <div id="main" align="left">
                <b>Wasserturm Bergwald</b><br /><br />
                 <center><img src="./grafics/turm.png" width="265" alt="header" /><br /><br
                /></center>
                Der Wasserturm wurde zur Versorgung der Bergwaldsiedlung im Jahr 1967 erstellt. Er
                enthält zwei je 200 Kubikmeter fassende Ringkammern im oberen Teil
                des Turmes.
                Der obere Wasserspiegel liegt bei NN+ 292 Meter, beziehungsweise 33,30 Meter
                über Gelände.<br /><br />
</body>
</html>
```

Der blaue Text stellt die eigentlichen Informationen dar, welche mit Hilfe des Web-Viewers in der Applikation angezeigt werden.

Sämtliche HTML-Seiten beziehen sich auf dasselbe Stylesheet. Dieses Stylesheet legt die grundlegenden gestalterischen Regeln für das HTML-Gerüst fest und dient damit der Gestaltung und Anordnung der HTML-Seiten.

Nachstehend wird der Programmcode des Stylesheets vorgestellt.

```
@charset "utf-8";
/* CSS Document */
body {
        text-align:left;
}
#main{ font-family:"Arial", "Arial", sans-serif;
        width:265px;
        height: 100%;
        float: left;
        margin: 0px 0px 0px 0px;
        padding: 10px 10px 8px 20px;
}
```

In diesem Stylesheet wird das Erscheinungsbild für die HTML-Seiten definiert. Sie legen fest, dass der Informationstext immer linksbündig angeordnet werden soll. Der Font wird auf „Arial“ gesetzt. Weiterhin wird die Breite der HTML-Seite bestimmt. Dies ist eine der wichtigsten Funktionen im Rahmen der HTML-Umgebung, da einer Applikation meist nicht so viel Platz wie auf einem Desktop-Bildschirm zur Verfügung steht. Die Breite der HTML-Seite wird auf 265 Pixel festgelegt. Die Angabe in Pixel wird gewählt, da die Displaygröße der Smartphone ebenfalls in Pixel definiert wird. Die angesprochenen Rahmenbedingungen werden im Programmcode blau dargestellt.

Durch die Definition eines Stylesheets ist es möglich, das Aussehen der Screens im Hinblick auf die Positionierung zu definieren und somit unabhängig von textlichen Änderungen zu machen.

6.2 Technische Umsetzung der Kartenkomponente

In diesem Kapitel wird auf die technische Umsetzung der Kartenkomponente eingegangen. Die Auswahl der einzusetzenden Komponente wurde in Kaptiel 3.5 beschrieben. Für die technische Umsetzung der Kartenkomponente im Rahmen des Prototypen wird der Kartendienst Google Maps verwendet.

Um Google Maps im Rahmen des Prototypen in eine Website einzubinden gibt es zwei Möglichkeiten. Diese sind die Einbindung über eine Google Maps API und die Einbindung im Rahmen einer URL welche in einer bestehenden Website eingebettet wird. Dieses Verfahren kann jedoch lediglich für nicht kommerzielle Zwecke verwendet werden. Für die Entwicklung des Prototypen würden grundsätzlich beide Möglichkeiten in Frage kommen. Wird jedoch berücksichtigt, dass der Prototyp zu einer finalen Applikation ausgebaut werden soll und somit auch zur kommerziellen Nutzung eingesetzt wird, kann Google Maps lediglich über die Google Maps API in eine Website eingebettet werden. Aus diesem Grund basiert die Einbindung des Kartendienstes bereits im Rahmen dieser Arbeit auf der Google Maps API. Außerdem ist es nur bei der Entwicklung mit Hilfe der Google Maps API möglich, eigene Kartenanwendungen zu erstellen, da hierfür auf die Funktionen der API zugegriffen werden muss. Da die Infokarten verschiedene Informationen enthalten sollen, ist es wichtig diese selbst erstellen zu können. Hierbei unterstützt die Google Maps API bei der Bereitstellung des Kartenmaterials und deren Funktionen wie z.B. die Verortung und Darstellung einzelner Objekte. Außerdem stellt die API weitere Funktionen bereit, welche für die Infokarte im Rahmen dieses Projekts benötigt werden.

Die Infokarte nutzt als Kartengrundlage die Google Maps Karten. Außerdem werden drei unterschiedliche Themengebiete auf der Karte dargestellt. Dabei handelt es sich um unterschiedliche Objekte derselben Kategorie. Die drei Kategorien sind „Historische Anlagen“, „Brunnen“ und „Wasserwerke“. Jede dieser Kategorien enthält standortbezogene Punktobjekte welche durch einen Marker dargestellt werden. Abbildung 34 zeigt die unterschiedlichen Marker.

Abbildung 34: Marker-Historische Anlagen, Brunnen, Wasserwerke

Zusätzlich zu diesen, befindet sich ein weiterer Marker auf der Karte (Abbildung 35). Dieser zeigt die aktuelle Position des Nutzers für eine bessere Orientierung an.

Abbildung 35: Marker-Aktuelle Position

Die mit den Markern gekennzeichneten Punktobjekte stellen jeweils eine historische Anlage, einen Brunnen oder ein Wasserwerk dar. Wird ein solches Objekt von einem Nutzer ausgewählt, öffnet sich ein Info-Fenster, welches Basisinformationen über das Objekt bereitstellt. Zusätzlich zu diesen Informationen können weitere Informationen abgerufen werden. Diese werden durch einen Klick auf „Weiter Infos“ abgerufen. Durch Klicken auf diese Schaltfläche gelangt der Nutzer auf eine HTML-Seite, die dann die umfassenden Informationen enthält.

Die für die technische Umsetzung erforderliche HTML-Seite enthält lediglich die Karte, welche für die Applikation entwickelt wird. Mit Hilfe der Google Maps API und dem Datenaustauschformat JSON[1] wird der Kartendienst Google Maps in diese HTML-Seite implementiert. Das erforderliche JSON Script stammt von der Website www.prototypejs.org und wird unter der MIT-style license[2] verwendet. Das Grundgerüst der HTML-Seite gleicht jeder anderen Website. Durch die Google Maps API und das JSON Script werden die Kartengrundlage und sämtliche Funktionen der API in die Website eingebunden. Dies erfolgt durch spezielle Skripte, welche als Quelle der Website zugewiesen werden. ***PrototypeJS*** *[Online]*

Durch den folgenden Ausdruck werden die Google Maps API und das JSON-Script in die Website eingebunden.

```
<script src="https://maps.googleapis.com/maps/api/js?sensor=false"></script>
<script src="prototype.js"></script>
```

Anschließend wird eine neue Karte erstellt. Die Skriptsprache der Google Maps API ist JavaScript[3]. Um die Karte zu erstellen wird durch den Ausdruck „<script type="text/javascript">“ ein neuer JavaScript Bereich innerhalb der HTML-Seite eingeleitet. Die Karte erhält einige Grundeinstellungen wie z.B. die Zoomstufe, welche beim ersten Laden der Karte dargestellt wird sowie die Kartenmitte. Diese wird mit Hilfe von Koordinaten (UTM-System) bereitgestellt. Zusätzlich wird der Kartentyp ermittelt, welcher bereitgestellt werden soll. Die durch Google bekannte Standardansicht ist die Straßenkarte (Roadmap).

```
function initialize() {
        var mapOptions = {
            zoom: 15,
            center: new google.maps.LatLng(49.014383, 8.404396),
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };
        map = new google.maps.Map($('map'), mapOptions);
```

[1] JSON (Java Script Object Notation) ist ein Datenformat mit dessen Hilfe Daten zwischen Anwendungen ausgetauscht werden (Datenaustauschformat).
[2] License: www.prototypejs.org/license.html
[3] JavaScript ist eine Skriptsprache. Diese wurde für dynamische HTML-Seiten entwickelt und sollte die Möglichkeiten von HTML und CSS verstärken und ausbauen. Heute kommt JavaScript jedoch auch in anderen Bereichen wie zum Beispiel bei Servern zum Einsatz.

Die Punktobjekte und die dazugehörigen Informationen werden auf der Karte anhand von Markern und dazugehörigen Info-Fenstern dargestellt. Für jedes einzelne Objekt wird hierfür ein Marker und ein Info-Fenster erstellt. Um einen Marker zu erstellen wird folgendes Script benötigt.

```
var marker1 = new google.maps.Marker({
                                        position: new google.maps.LatLng(48.990567, 8.387967),
                                        map: map,
                                        icon: './b.png',
                                        title: 'Albbrunnen'
                                        });
```

Der Marker enthält die Position welche durch die passenden Koordinaten bereitgestellt wird. Er wird der Karte (map) zugewiesen und erhält ein Icon. Das Icon „b“ steht für Brunnen und wird in der Karte, wie in Abbildung 34 gezeigt. Außerdem wird dem Marker ein Titel zugewiesen welcher diesen beschreibt.

Das Skript für das dazugehörige Infowindow lautet:

```
var infowindow1 = new google.maps.InfoWindow({content: 'Albbrunnen <br /><br /><a
target=\"_blank\" href=\"http://www.brunnengesellschaft-
karlsruhe.de/erg_brunnen.php?stadtteilnummer=13&laufendenummer=01">Weitere
Infos</a></font>'});

                                        google.maps.event.addListener(marker1, 'click', function() {
                                        infowindow1.open(map, marker1);
                                        });
```

Das Info-Fenster enthält einen Titel. Dieser steht in Verbindung mit dem dazugehörigen Marker. Außerdem wird dem Info-Fensterw unter „content“ der Inhalt zugewiesen. Dieser besteht im Rahmen der Infokarte aus einer Web-URL[1]. Zusätzlich wird die Beschriftung für den Link „Weitere Infos“ erstellt. Bei Klick auf „Weitere Infos“ öffnet sich die zugewiesene URL. Mit Hilfe eines Eventlisteners[2] wird das Info-Fenster durch eine Funktion (click) mit dem zugehörigen Marker in Verbindung gebracht und eine Klick-Funktion ausgeführt welche bei Klick auf den Marker das Info-Fenster öffnet.

Sämtliche Marker und Info-Fenster sind auf dieselbe Art und Weise umgesetzt und nach dem oben aufgeführten „Grundgerüst“ aufgebaut. Die Marker unterscheiden sich durch ihre Koordinaten, den Titel und das Icon. Marker des Themengebiets „Historische Anlagen“ werden durch einen roten Marker mit einem „H“ dargestellt. Das Themengebiet „Brunnen“ wird durch einen blauen Marker mit einem „B“ und Marker des Themengebiets „Wasserwerke“ durch die Farbe Gelb und einem „W“ veranschaulicht (vergleiche Abbildung 34).

[1] URL steht für Uniform Resource Locator und beschreibt den Link einer Webseite.
[2] Ein Eventlistener ist ein Ereignisempfänger. Dieser empfängt ein Ereignis und setzt dieses um.

Die Info-Fenster unterscheiden sich durch den im „content“ enthaltenen Titel und dem Inhalt (Weblink). Jedes Info-Fenster ist einem Marker zugewiesen und enthält somit denselben Titel wie der dazugehörige Marker und den passenden Weblink zu den Informationen.

Eine weitere Funktion der Karte ermittelt die aktuelle Position des Nutzers und zeigt diese an. Diese Funktion hilft dem Nutzer sich auf der Karte zu orientieren und dient dazu weitere interessante Objekte in der direkten Umgebung zu entdecken. Die aktuelle Position wird durch einen weiteren Marker dargestellt. Dieser Marker ist grün und enthält ein „A“ (siehe Abbildung 35). Um die aktuelle Position darstellen zu können wird ein weiteres Skript innerhalb der HTML-Seite benötigt. Dieses Skript beinhaltet eine Funktion mit verschiedenen Variablen. Die aktuellen Koordinaten werden abgerufen und durch den Marker verortet.

Die Variable „latlng“ beschreibt die Koordinaten mit Hilfe der Breiten- und Längengrade. Der Marker erhält seine Position jeweils durch die „latlng“ Variable. Außerdem wird der Marker durch seinen Titel und das Icon ergänzt.

Um die aktuelle Position zu erhalten, wird die „getCurrentPosition“ Funktion des Objekts „geolocation“ verwendet. Dieses wird durch die Google Maps API bereitgestellt. Um die Position zu erhalten gibt es zwei mögliche Rückmeldungen. Konnte die aktuelle Position ermittelt werden, werden im Rahmen des „position“ Objekts die aktuellen Koordinaten zurückgegeben. Falls es nicht möglich war die Position zu bestimmen besteht die Rückmeldung aus einer Fehlermeldung. Das folgende Scritp zeigt den Programmiercode für die Standortbestimmung.

```
<script>
    function initialize(coords) {
      var latlng = new google.maps.LatLng(coords.latitude, coords.longitude);


      var marker = new google.maps.Marker({
        position: latlng,
        map: map,
                icon: './a.png',
        title: "Aktuelle Position"
      });
    }

    navigator.geolocation.getCurrentPosition(function(position){
      initialize(position.coords);
    }, function(){
       document.getElementById('pos').innerHTML = 'Die aktuelle Position konnte nicht ermittelt
       werden.';
    });
  </script>
```

Zur Positionsbestimmung werden mehrere „Kanäle“ verwendet. Im Rahmen der Infokarte ist es möglich die Position über den im Smartphone verbauten GPS-Sender, einem WLAN-Netz und dem Handynetz zu bestimmen.

Hierbei ist die Positionsbestimmung am genausten wenn alle 3 Funktionen parallel zur Verfügung stehen. Wird die Position lediglich über ein WLAN-Netz bestimmt liefert dies das ungenaueste Ergebnis.

6.3 Verwendete Software zur Umsetzung des Prototypen

Im folgenden Kapitel wird auf die verwendete Software zur Umsetzung des Prototypen eingegangen. Zur technischen Umsetzung der Applikation wird die Entwicklungsumgebung „MIT App Inventor“ verwendet, welche in Kapitel 4 eingeführt wurde. Bei App Inventor handelt es sich um eine webbasierte Entwicklungsumgebung. Diese wird über den Webbrowser aufgerufen. App Inventor ist keine eigenständige Programmiersprache. App Inventor versteht sich selbst als visuelle Entwicklungssprache. Bei der Entwicklung von Applikationen mit Hilfe von App Inventor kann der Quellcode nicht durch einen Texteditor geöffnet und bearbeitet werden, da dieser interaktiv über die Entwicklungsumgebung generiert wird.

App Inventor besteht aus zwei grundlegenden Arbeitsbereichen, die als DESIGNER und BLOCKS EDITOR bezeichnet werden. Diese zwei Bereiche arbeiten zusammen und werden für die Entwicklung einer Applikation verwendet.

Im DESIGNER wird die Oberfläche der Anwendung gestaltet. Für die Gestaltung der Oberfläche stehen verschiedene vorbereitete Komponenten zur Verfügung. Aus diesen Komponenten wird die funktionale Oberfläche zusammengesetzt. Der DESIGNER ist in fünf Funktionsbereiche aufgeteilt. Diese sind PALETTE, VIEWER, COMPONENTS, MEDIA und PROPERTIES. Diese Funktionsbereiche stellen verschiedene grafische und funktionale Komponenten zur Gestaltung und zum „designen“ der Applikation zur Verfügung. Die folgende Abbildung 36 zeigt den DESIGNER-Bereich der Entwicklungsumgebung und dessen oben genannte Funktionsbereiche.

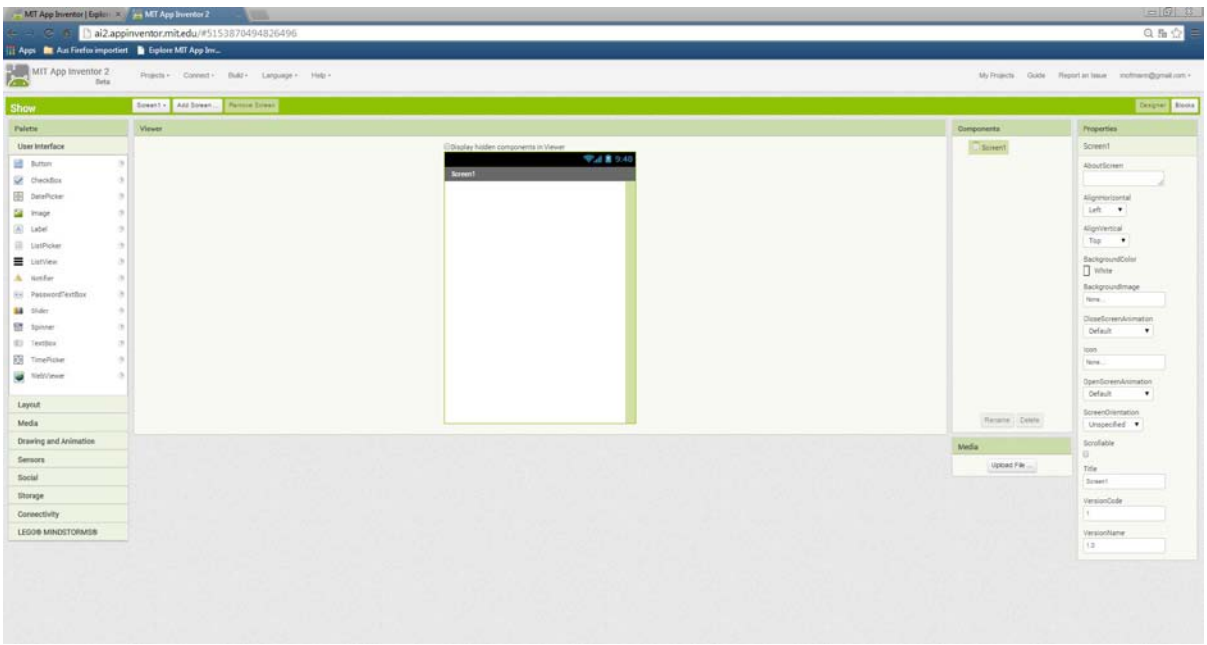


Abbildung 36: Designer Funktionsbereich des App Inventors

Die verschiedenen Funktionsbereiche haben unterschiedliche Aufgaben. Der Funktionsbereich PALETTE, welcher am linken Rand der Entwicklungsumgebung angeordnet ist, listet sämtliche grafische und funktionale Komponenten auf. Diese Komponenten werden zur Gestaltung und Entwicklung der Applikation verwendet. Sie sind wiederum in verschiedene Gruppen gegliedert. Diese Gruppen sind User Interface, Layout, Media, Drawing and Animation, Sensors, Social, Storage, Connectivity und LEGO MINDSTORMS. Durch Auswahl der Gruppe werden die verschiedenen Komponenten angezeigt. Neben den einzelnen Komponenten wird jeweils ein Fragezeichen eingeblendet. Dieses gibt dem Nutzer eine ausführliche Erklärung der gewählten Komponente.

Der zweite Funktionsbereich VIEWER zeigt eine Art Display eines Smartphones. Das Display enthält die typische Darstellung eines Smartphone-Displays mit den Icons für Wlan, Empfang, Akkuladezustand und die Uhrzeit. Diese Icons dienen lediglich zur Darstellung und weisen keinerlei Funktionen auf. Unter diesen Icons wird der eigentlich Bereich für die grafische Entwicklung der Applikation bereitgestellt und angezeigt. In diesem Bereich werden die einzelnen Komponenten aus der PALETTE eingefügt. Dies geschieht durch Hineinziehen der Komponenten aus der PALETTE in den VIEWER-Bereich. Wurde eine Komponente in diesen Bereich eingefügt wird diese auf dem improvisierten Smartphone-Display dargestellt. Der VIEWER gibt somit einen ersten Eindruck von der Darstellung der Komponenten auf dem späteren Display des Endgeräts. Hierbei muss darauf hingewiesen werden, dass die Darstellung unter Umständen von der späteren Darstellung abweicht. Der VIEWER liefert daher nur einen ersten Eindruck der Komponenten und deren Anordnung auf dem Display des Endgeräts.

Der dritte Funktionsbereich, COMPONENTS, zeigt sämtliche Komponenten, welche in den Bereich VIEWER eingefügt werden. Diese werden in einer hierarchischen Struktur (Baumstruktur) angezeigt. Hierdurch lassen sich Gruppen und Untergruppen mit gleichen Eigenschaften vereinen. Somit zeigt die Struktur ebenfalls Abhängigkeiten und Vererbung der Komponenten untereinander an. Abbildung 37 zeigt die Entwicklungsumgebung mit einer Komponente im VIEWER und somit derselben Komponente im Bereich COMPONENTS.

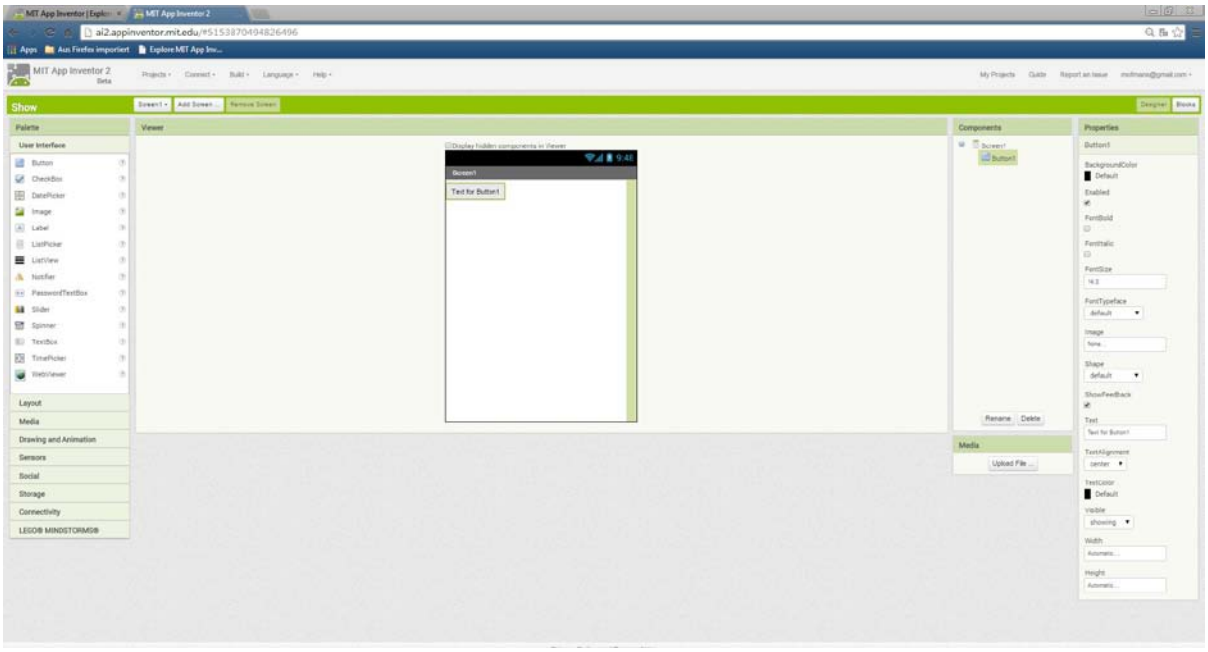


Abbildung 37: Komponente im Viewer und unter Components

Im Funktionsbereich MEDIA werden sämtliche Medien dargestellt, welche in der späteren Applikation enthalten sind. Diese Medien beschränken sich auf Audio-, Video- und Bilddateien. Über einen „Upload File Button“ lassen sich die Medien in die Entwicklungsumgebung hochladen. Die Medien werden hierfür vom lokalen Computer in die Entwicklungsumgebung geladen. In Abbildung 37 wird der Bereich MEDIA im rechten, unteren Bildrand dargestellt. Der Bereich enthält momentan keinerlei Medien. Die Verwendung der hochgeladenen Medien funktioniert lediglich mit den entsprechenden Medienkomponenten welche in der PALETTE in der Gruppe „Media“ aufgeführt sind.

Der letze Funktionsbereich, PROPERTIES, zeigt die Eigenschaften der Komponenten. Hierbei wird jeweils die aktuell im VIEWER oder in COMPONENTS ausgewählte Komponente berücksichtigt. Im Bereich PROPERTIES wird je nach ausgewählter Komponente deren Eigenschaften und Einstellungsmöglichkeiten angezeigt. Je nach Komponente können verschiedene Einstellungen vorgenommen werden. Einstellungsmöglichkeiten sind unter anderem die Hintergrundfarbe, der Font, eine zugehörige Bilddatei, Texte. Werden Änderungen in den Komponenteneigenschaften vorgenommen sind diese sofort im Bereich VIEWER sichtbar. Abbildung 38 zeigt den Bereich PROPERTIES und die Eigenschaften und Einstellungsmöglichkeiten für die Beispielkomponente „Button 1“.

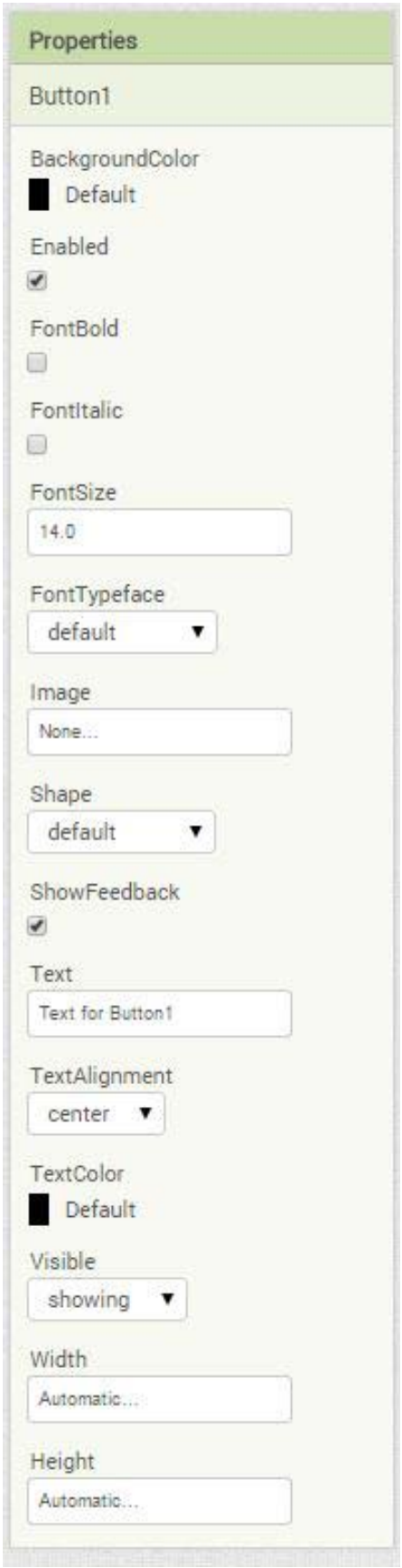


Abbildung 38: Properties der Beispielkomponente "Button 1"

Der zweite Arbeitsbereich in MIT App Inventor ist der BLOCKS EDITOR. Dieser bildet den zweiten zentralen Arbeitsbereich. Der BLOCKS EDITOR wird verwendet um die Applikation mit „Leben“ zu versehen. Mit Hilfe des BLOCKS EDITOR werden die Aufgaben und Funktionalitäten umgesetzt welche der Applikation ihre Gesamtfunktionalität verleiht. Der BLOCKS EDITOR wird durch einen Button am oberen rechten Rand der Entwicklungsumgebung aufgerufen. Durch Klicken auf diesen Button wird die Ansicht des Webbrowsers auf den BLOCKS EDITOR geändert. Die folgende Abbildung 39 zeigt die Arbeitsoberfläche des BLOCKS EDITOR.

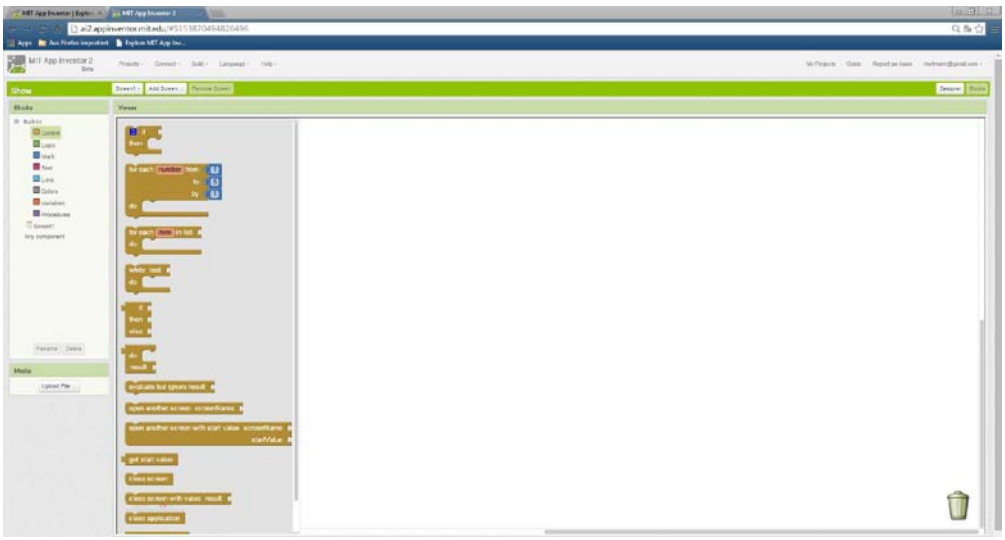


Abbildung 39: Hauptansicht Blocks Editor

Die Arbeitsoberfläche des BLOCKS EDITOR bezieht sich stets auf den Screen, welcher im DESIGNER ausgewählt ist. Außerdem ist es, wie im DESIGNER, möglich den aktuellen Screen über den Auswahlbutton zu ändern.

Der BLOCKS EDITOR ist ebenfalls in verschiedene Funktionsbereiche gegliedert. Diese sind BLOCKS, MEDIA und VIEWER.

Der Funktionsbereich BLOCKS stellt im BLOCKS EDITOR keine Komponenten dar. Dieser Bereich beinhaltet eine Liste von funktionalen Blöcken. Diese Blöcke sind mit dem Befehlssatz einer klassischen Programmiersprache gleichzustellen. Die Blöcke machen somit den Syntax[1] der visuellen Entwicklungssprache von App Inventor aus. Einzelne Befehle (Puzzleteile) werden zu Blöcken zusammengefügt. Diese Blöcke bilden dann die eigentliche Funktionalität für die Applikation.

Mit Hilfe der Blöcke ist es mögliche den einzelnen Komponenten des DESIGNER eine Funktion zuzuweisen. Die einzelnen Blöcke werden zusammengefügt und bilden somit schlussendlich den gesamten Funktionsaufbau der späteren Applikation.

[1] Unter Syntax versteht man die Regeln (ein System aus Regeln) einer Programmiersprache. Der Syntax regelt die erlaubten Konstruktionen und Ausdrücke welche im Rahmen der Programmiersprache gebildet werden dürfen.

Zusammenspiel von Komponenten im Designer und Blöcken im Blocks Editor

*Der AI Designer dient vor allem dazu, die Benutzerschnittstelle einer App zu gestalten. Hierfür stellt der Designer verschiedene Komponenten zur Verfügung: interaktive Buttons, Schreibfelder, Sensoren und vieles mehr für die Benutzereingaben (Input) sowie Text- und Bildanzeigen, Audio- und Videoplayer und vieles mehr für die Ausgaben an den Benutzer (Output). Mit dem AI Blocks Editor ist es dann möglich, die bis dahin isolierten Input- und Output- Komponenten systematisch miteinander zu verbinden. So werden Texteingaben, Auswahlen, Geokoordinaten und vieles mehr aus den Input-Komponenten entgegengenommen, innerhalb der App und mitunter durch Zugriff auf externe Datenquellen im Web weiterverarbeitet und die Ergebnisse schließlich an die Output-Komponenten zur Ausgabe weitergereicht. Durch das systematische Zusammenwirken der aufeinander abgestimmten Komponenten und Blöcke entsteht die interaktive App." **Kloss, J. H.** [Buch]*

Der Funktionsbereich BLOCKS ist unterteilt in drei weitere Bereiche. Diese sind „Built-in", „Screen" und „Any Components". Abbildung 40 zeigt die drei Bereiche des Funktionsbereichs BLOCKS.

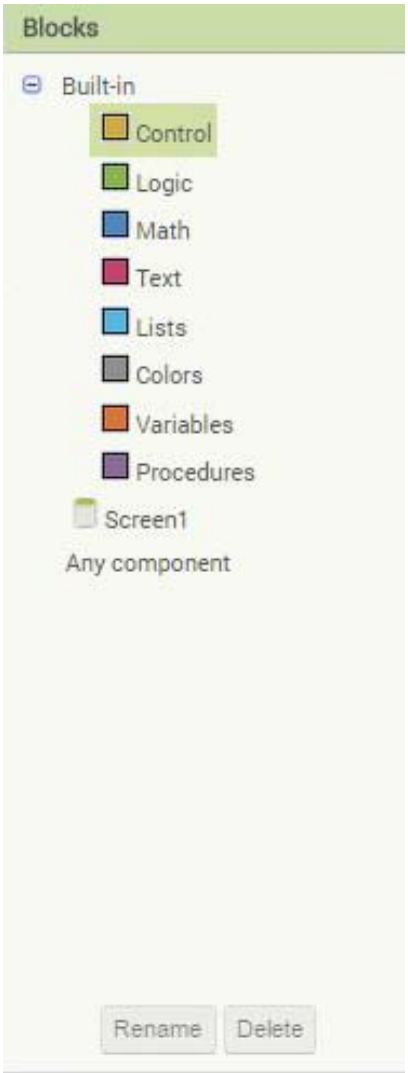


Abbildung 40: Bereiche des Funktionsbereichs Blocks

Unter dem Bereich „Built-in“ werden 8 ständig vorhandene Blockgruppen bereitgestellt (Abbildung 41). Diese sind Control, Logic, Math, Text, Lists, Colors, Variables und Procedures. Mit einem Klick auf eine Gruppe öffnet sich diese und die verschiedenen Blöcke werden angezeigt. Diese Blöcke werden in Form der oben angesprochenen Puzzleteilen dargestellt.

Im Bereich „Screen“ werden sämtliche Blockgruppen und Blöcke bereitgestellt, die für die aktuelle Applikation eine Rolle spielen. Hierbei werden die möglichen Blöcke für jede Komponente bereitgestellt, die zuvor im DESIGNER in den VIEWER integriert wurden und somit einen direkten Zusammenhang mit der Applikation bilden. Dieser Bereich stellt somit die komponentenspezifischen Blöcke bereit. Die Blockgruppen tragen in diesem Fall den Namen der zugehörigen Komponente. Abbildung 41 zeigt die Beispielkomponente Button 1 im BLOCKS Bereich.

Abbildung 41: Beispielkomponente Button 1

Im Bereich „Any Components“ werden ebenfalls Blöcke bereitgestellt, die sich auf vorher hinzugefügte Komponenten beziehen. Hier werden im Gegensatz zum Screen Bereich jedoch nur die grundlegenden Blöcke der Komponenten angezeigt. Diese Blöcke beziehen sich nicht konkret auf eine bestimmte Komponente, sondern lediglich auf die grundsätzlichen Komponenten welche hinzugefügt wurden.

Im Funktionsbereich MEDIA werden sämtliche Medien angezeigt, welche zuvor im DESIGNER hochgeladen wurden.

Der Funktionsbereich VIEWER des BLOCKS EDITOR veranschaulicht die zusammengesetzten Blöcke und stellt somit den eigentlichen „Editor“ im klassischen Sinne dar. Im VIEWER werden die verschiedenen Befehle (Puzzleteile) aus dem Bereich BLOCKS zu Blöcken zusammengesetzt und damit die Programmlogik entwickelt. Die Blöcke bilden, wie vorher erwähnt, die Funktionalitäten der einzelnen Komponenten des DESIGNER. Abbildung 42 zeigt beispielhaft die Zusammensetzung einzelner Puzzleteile zu einem Block, der der Komponente Button 1 die Funktion für die Änderung der Hintergrundfarbe in Gelb nach dem Klicken auf den Button zuweist.

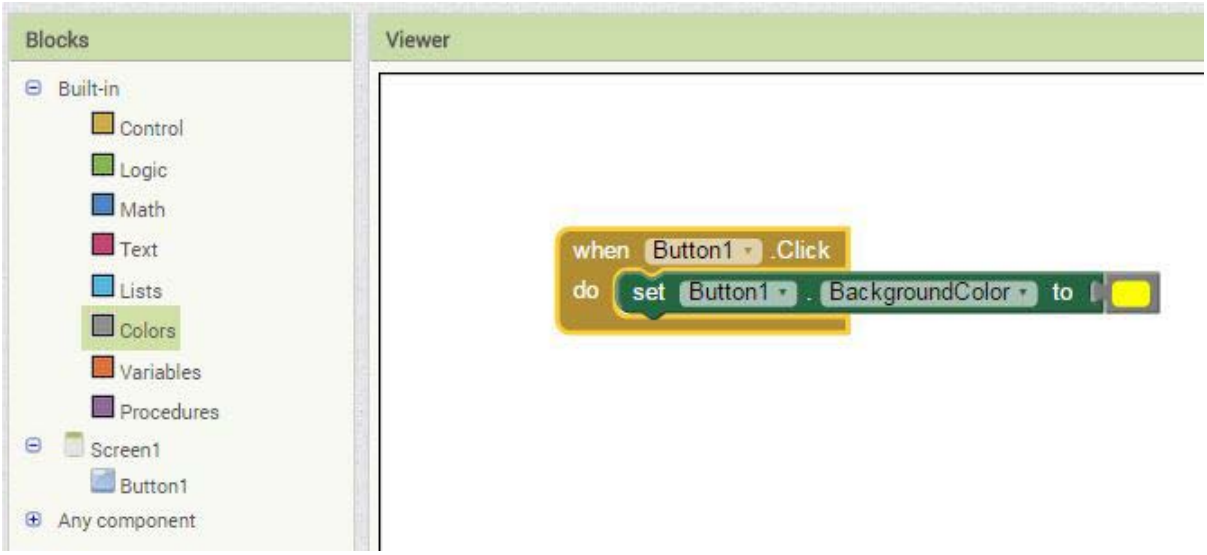


Abbildung 42: Beispielblock Button 1

Ein weiterer wichtiger Aspekt der Entwicklungsumgebung ist die Integration eines Smartphones. App Inventor bietet hier verschiedene Möglichkeiten an. Zum einen kann der Entwicklungszustand und vor allem der Fortschritt in einem Emulator angezeigt werden. Dieser Emulator wird von App Inventor zur Verfügung gestellt.

Eine weitaus interessantere Lösung bietet die „AI Companion App“, welche ebenfalls von App Inventor zur Verfügung gestellt wird. Mit Hilfe dieser Applikation ist es möglich ein Smartphone entweder über eine USB-Verbindung oder über eine WLAN-Verbindung mit der webbasierten Entwicklungsumgebung zu verbinden. Hierzu muss auf dem zu verbindenden Smartphone die „AI Companion App“ installiert sein. In der Entwicklungsumgebung wird über den Button „Connect“ ausgewählt, über welche Schnittstelle die Verbindung zum Smartphone hergestellt werden soll. Wird hier die AI Companion Variante ausgewählt, stellt die Entwicklungsumgebung einen QR-Code bereit. Dieser wird durch die Companion App auf dem Smartphone gescannt und die Verbindung wird aufgebaut. Die Abbildungen 43 und 44 zeigen die beschriebene Vorgehensweise.

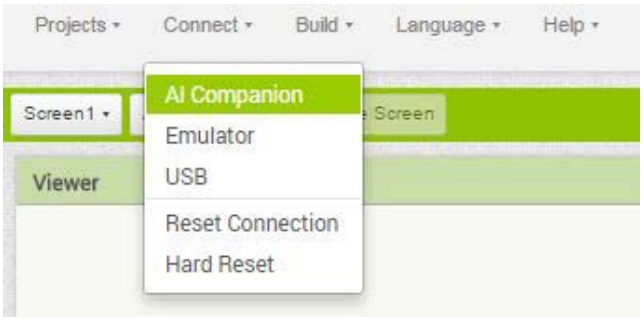


Abbildung 43: Verbindung durch AI Companion

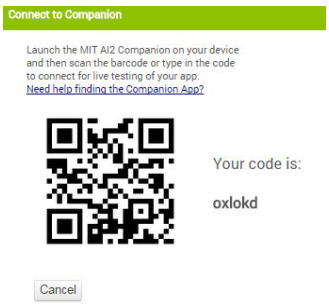


Abbildung 44: QR Code zur Verbindung

Ist die Verbindung zum Smartphone hergestellt wird die Anzeige in Echtzeit aktualisiert. Das Smartphone zeigt stets die aktuelle Darstellung des VIEWER aus dem Designer an. Wird im VIEWER eine neue Komponente eingefügt oder eine bestehende verändert, wird die Aktualisierung sofort auf dem verbundenen Smartphone angezeigt. Die Companion App beschränkt sich momentan auf die aktuelle Darstellung und die Funktionen welche der Applikation durch den BLOCKS EDITOR hinzugefügt wurden. Die App ermöglicht jedoch nicht den Zugriff auf Funktionen welche vom Smartphone bereitgestellt werden. Es besteht kein Zugriff auf Sensoren oder andere Anwendungen des Smartphones. Somit eignet sich die Companion App vor allem zur Überprüfung der Anordnung und der Darstellung der Applikation und deren aktuellen Änderungen. Die komplette Funktionsweise der Anwendung kann jedoch nicht getestet werden.

Um den kompletten Funktionsumfang der Applikation zu testen ist es nötig, die Applikation lokal auf einem geeigneten Smartphone zu installieren. Um eine Applikation auf einem Endgerät zu installieren wird eine installationsfähige Datei benötigt. Im Rahmen dieses Projekts und unter dem Betriebssystem Android werden installationsfähige Dateien im Format .apk[1] (Android Package) bereitgestellt. Um ein Projekt im apk-Format zu installieren stellt App Inventor wiederum zwei Möglichkeiten zur Verfügung.

[1] apk steht für Android application package und bildet das Dateiformat welches zur Installation von Anwendungen auf Smartphones mit dem Betriebssystem Android verwendet wird.

Im Rahmen der ersten Möglichkeit stellt die Entwicklungsumgebung einen QR Code zur Verfügung. Mit Hilfe eines geeigneten Scanners auf dem Smartphone wird der Code gescannt und die apk-Datei direkt auf das Smartphone heruntergeladen (Abbildung 45). Nach Beendigung des Downloads kann die Applikation direkt auf dem Smartphone installiert werden.

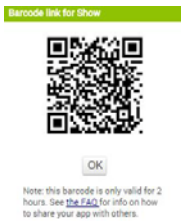


Abbildung 45: QR Code zum Laden der apk-Datei

Die zweite Möglichkeit besteht darin, die eigentliche apk-Datei auf den Computer herunterzuladen (Abbildung 46). Wurde die Datei heruntergeladen, kann diese über eine USB-Verbindung auf ein geeignetes Smartphone übertragen undproblemlos auf dem Smartphone installiert werden.

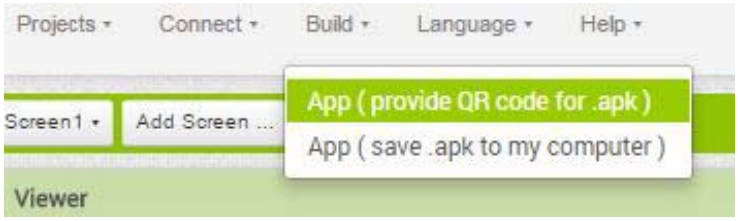


Abbildung 46: Download der apk-Datei auf den Computer

Grundsätzlich kann die komplette Funktionsweise der Applikation nach erfolgreicher Installation auf einem geeigneten Endgerät getestet werden. Durch die lokale Installation ist es möglich auf sämtliche vorhandenen Funktionen und Sensoren des Smartphone zuzugreifen und somit den kompletten Funktionsumfang der Applikation zu testen.

Um die Applikation abschließend im Google Play Store zu veröffentlichen sind noch einige Schritte nötig. Wurde die Applikation mit Hilfe von App Inventor entwickelt und als .apk-Datei auf einem Computer gespeichert kann diese nicht direkt in den Google Play Store hinzugefügt werden. Um die Applikation App Store tauglich zu machen sind eine weiter Konvertierung und einige Einstellungen nötig. Für die Konvertierung wird eine weitere Software verwendet. Diese hilft dem Entwickler dabei die Applikation für den Google Play Store zu bearbeiten.

Die verwendete Software ist frei erhältlich und trägt den Namen „Marketizer“. Um eine Applikation im Google Play Store veröffentlichen zu können muss diese einige Metadaten enthalten. Dies sind unter anderem Daten über den Entwickler. Diese werden über einen „Cert“ definiert. Um einen neuen „Cert“ zu erstellen werden die Angaben in „Marketizer“ vorgenommen. Die folgende Abbildung 47 zeigt die Oberfläche von „Marktizer“ für die Erstellung eines neuen „Cert“.

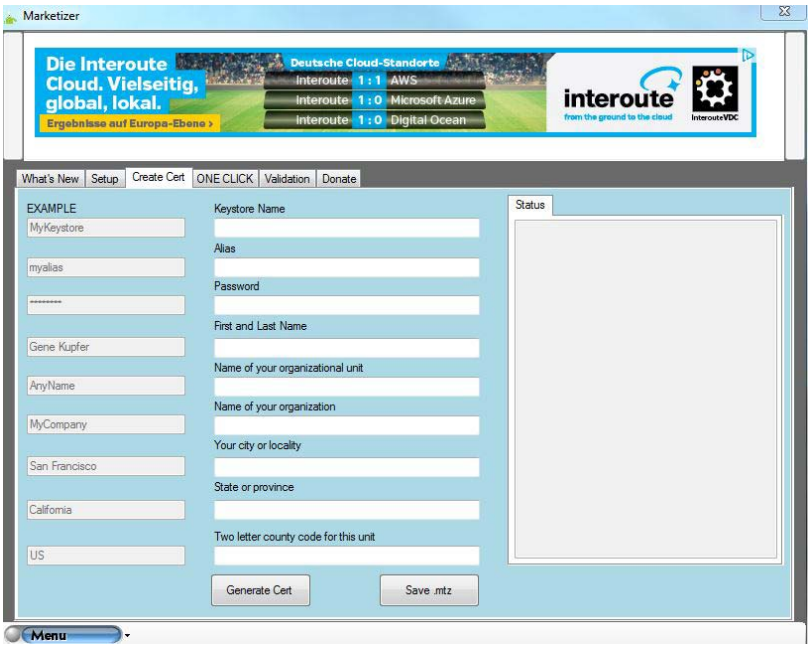


Abbildung 47: Marketizer-Create Cert

Im nächsten Schritt wird die zuvor aus App Inventor gespeicherte apk-Datei in „Marketizer“ ausgewählt und die Eingaben für den „Version Code“, den „Version Name“, die Bezeichnung der apk-Datei und die Bezeichnung der eigentlichen Applikation vorzuenhmen. Unter „Advanced Settings“ können dann noch weitere Einstellungen durchgeführt werden. Diese betreffen vor allem die Anpassung der Applikation auf unterschiedliche Displaygrößen. Der Entwickler kann unter anderem festlegen auf welchen Displaygrößen die Anwendung lauffähig sein soll und ob diese im Fall eines Größenunterschieds angepasst werden soll. In Abbildung 48 ist die Vorgehensweise zur Auswahl der apk-Datei dargestellt.

Abbildung 48: Marketizer-Select apk

Sind sämtliche Eingaben und die benötigten Einstellungen vorgenommen, kann die Applikation unter „ONE CLICK“ konvertiert werden. „Marketizer“ nimmt hierfür die originale apk-Datei und erstellt eine neue apk-Datei, welche den zuvor festgelegten Namen erhält. In der neuen apk-Datei sind sämtliche Änderungen und Zusatzinformationen enthalten.

Nachdem eine neue apk-Datei durch „Marketizer“ erstellt und gespeichert wurde kann diese Datei für die Veröffentlichung der Applikation im Google Play Store verwendet werden.

6.4 Technische Umsetzung des Prototypen

Im vorangegangenen Kapitel wurde auf die Entwicklungsumgebung eingegangen. In diesem Kapitel wird die technische Umsetzung des Prototypen beschrieben. Um die technische Umsetzung strukturiert darzustellen wird zuerst auf die Umsetzung im DESIGNER Bereich der Entwicklungsumgebung eingegangen. Anschließend wird die Entwicklungsarbeit im BLOCKS EDITOR beschrieben.

Die Arbeiten im DESIGNER orientieren sich stark am Feinlayout, welches in Kapitel 5.5 beschrieben wurde. Bei der Umsetzung ist darauf zu achten, dass jeder einzelne Screen separat umgesetzt werden muss. Hierfür unterstützt App Inventor rein theoretisch „unendlich“ viele Screens. Werden jedoch mehr als 10 einzelne Screens angelegt weist die Entwicklungsumgebung darauf hin, dass mehr als zehn einzelne Screens Probleme erzeugen können. Dies liegt an der Speicherverwaltung der Endgeräte. Wird eine Applikation entwickelt welche aus einer Vielzahl aus einzelnen Screens besteht wird der Speicher auf dem Endgerät schnell überlastet und die Applikation hierdurch beendet. Eine Applikation mit vielen Screens stößt deshalb schnell an ihre Grenzen und ist nur bedingt lauffähig. Aus diesem Grund ist bei der Entwicklung darauf zu achten, dass so viele Inhalte wie möglich auf einem Screen gebündelt werden. Screens welche nahezu dieselben Ereignisse beinhalten und sich lediglich die Informationen ändern, müssen deshalb auf einem Screen umgesetzt werden. Die verschiedenen Komponenten werden hierfür auf dem Screen platziert und lediglich ein- beziehungsweis ausgeblendet wenn diese benötigt werden. Somit kann ein einzelner Screen eine Vielzahl an Komponenten enthalten, welche je nach Informationsübermittlung dargestellt oder ausgeblendet werden. Durch dieses Entwicklungsverfahren ist es möglich die Anzahl unterschiedlicher Screens gering zu halten und die Performance der Applikation somit zu steigern.

Aus diesem Grund werden für den Prototypen lediglich fünf einzelne Screens verwendet. Diese Screens werden erzeugt und je nach ihrem Inhalt benannt, die in der folgenden Abbildung 49 dargestellt sind.

Abbildung 49: Screens

Die Screens sind Screen1, Screen2, Screen3, der Downloadbereich und der Info Screen. Jeder Screen enthält unterschiedliche Komponenten und Inhalte. Die unterschiedlichen Komponenten welche auf den Screens angeordnet sind werden in den folgenden Tabellen 11 bis 15 zusammengestellt. Hierbei werden die einzelnen Komponenten genannt. Die Spalte „Unterkomponente“ listet die genauen Komponenten und deren Bezeichnungen auf, welche im jeweiligen Screen vorhanden sind. In der Spalte Beschreibung wird erläutert, welche Funktionen und Eigenschaften die einzelnen Komponenten aufweisen. Hierfür wird der aktuelle Hilfetext der Entwicklungsumgebung MIT App Inventor verwendet.

Screen1

Komponenten	Unterkomponente	Beschreibung
HorizontalArrangement	HorizontalArrangement 1, 2, 3, 5, 6, 7, 8, 9, 10, 11	A formatting element in which to place components that should be displayed from left to right. If you wish to have components displayed one over another, useVerticalArrangement instead.
Image	Logo_SWKA,	Component for displaying images. The picture to display, and other aspects of the Image"s appearance, can be specified in the Designer or in the Blocks Editor.
Label	Starttext, Info_Hilfe, Home	A Label displays a piece of text, which is specified through the Text property. Other properties, all of which can be set in the Designer or Blocks Editor, control the appearance and placement of the text.
Button	Test_Button, Infos_Hilfe_Button, Button_Download, Home_Button, Info_Button, Karte_Button, QR_Button	Button with the ability to detect clicks. Many aspects of its appearance can be changed, as well as whether it is clickable (Enabled), can be changed in the Designer or in the Blocks Editor.
Web Viewer	Web Viewer 1, 2	Component for viewing Web pages. The Home URL can be specified in the Designer or in the Blocks Editor. The view can be set to follow links when they are tapped, and users can fill in Web forms.

Tabelle 11: Screen 1

Screen2

Komponenten	Unterkomponente	Beschreibung
HorizontalArrangement	HorizontalArrangement 1, 2, 4, 5, 6	A formatting element in which to place components that should be displayed from left to right. If you wish to have components displayed one over another, useVerticalArrangement instead.
Label	Label_QR	A Label displays a piece of text, which is specified through the Text property. Other properties, all of which can be set in the Designer or Blocks Editor, control the appearance and placement of the text.
Button	ButtonScan, Button_Home, Button_Inhalt, Button_Karte, Button_QR	Button with the ability to detect clicks. Many aspects of its appearance can be changed, as well as whether it is clickable (Enabled), can be changed in the Designer or in the Blocks Editor.
Web Viewer	Web Viewer 1	Component for viewing Web pages. The Home URL can be specified in the Designer or in the Blocks Editor. The view can be set to follow links when they are tapped, and users can fill in Web forms. Warning: This is not a full browser. For example, pressing the phone´s hardware Back key will exit the app, rather than move back in the browser history….
BarcodeScanner	BarcodeScanner 1	Component for using the Barcode Scanner to read a barcode

Tabelle 12: Screen 2

Screen3

Komponenten	Unterkomponente	Beschreibung
HorizontalArrangement	HorizontalArrangement 1, 3, 4, 5	A formatting element in which to place components that should be displayed from left to right. If you wish to have components displayed one over another, useVerticalArrangement instead.
Label	LabelKarte	A Label displays a piece of text, which is specified through the Text property. Other properties, all of which can be set in the Designer or Blocks Editor, control the appearance and placement of the text.
Button	Button_H, Button_I, Button_K, Button_Q	Button with the ability to detect clicks. Many aspects of its appearance can be changed, as well as whether it is clickable (Enabled), can be changed in the Designer or in the Blocks Editor.
Web Viewer	Web Viewer 1	Component for viewing Web pages. The Home URL can be specified in the Designer or in the Blocks Editor. The view can be set to follow links when they are tapped, and users can fill in Web forms. Warning: This is not a full browser. For example, pressing the phone´s hardware Back key will exit the app, rather than move back in the browser history….
LocationSensor	LocationSensor1	Non-visible component providing location information, including longitude, latitude, altitude (if supported by the device), and address. This can also perform "geocoding", converting a given address (not necessarily the current one) to a latitude (with the LatitudeFromAddress method) and a longitude (with the LongitudeFromAddress method).

Tabelle 13: Screen 3

Screen Downloadbereich

Komponenten	Unterkomponente	Beschreibung
HorizontalArrangement	HorizontalArrangement 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12,	A formatting element in which to place components that should be displayed from left to right. If you wish to have components displayed one over another, useVerticalArrangement instead.
Label	Label_Media_Download, LabelTextOben, LabelTextMitte, LabelTextUnten	A Label displays a piece of text, which is specified through the Text property. Other properties, all of which can be set in the Designer or Blocks Editor, control the appearance and placement of the text.
Button	Button_SWKA, Button_EBG, Button_Media, ButtonHome, ButtonInhalt, ButtonKarte, ButtonQR	Button with the ability to detect clicks. Many aspects of its appearance can be changed, as well as whether it is clickable (Enabled), can be changed in the Designer or in the Blocks Editor.
ActivityStarter	ActivityStarter 1, 2, 3	A component that can launch an activity using theStartActivity method. Activities that can be launched include: • starting other App Inventor for Android apps • opening a browser to a specified web page • opening the map application to a specified location…….

Tabelle 14: Screen Downloadbereich

Info_Screen

Komponenten	Unterkomponente	Beschreibung
HorizontalArrangement	HorizontalArrangement 1, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, HorizontalArrangement 1_Daten, HorizontalArrangement_Info_1, HorizontalArrangement3_Info, HorizontalArrangement4_Info, HorizontalArrangement5_Info, HorizontalArrangement6_Info, HorizontalArrangement7_Info, HorizontalArrangement8_Info, HorizontalArrangement9_Info, Horizontal Arrangement_Karte_Ansprechpartner, HorizontalArrangement_Kartenbutton	A formatting element in which to place components that should be displayed from left to right. If you wish to have components displayed one over another, useVerticalArrangement instead.
Label	Hausannschluss_Label, Rohrnetz_Label, DatenundFakten, Chronik_Label, Wasserwerke_Label, historische_label, brunnen_label, Infos_zur_TWV	A Label displays a piece of text, which is specified through the Text property. Other properties, all of which can be set in the Designer or Blocks Editor, control the appearance and placement of the text.
Button	Daten, Chronik, Historische, Wasserwerke, Rohrnetz, Hausanschluss, Brunnen, Ansprechpartner_btn, Kartenbutton, Home, Inhalt, Karte, QR	Button with the ability to detect clicks. Many aspects of its appearance can be changed, as well as whether it is clickable (Enabled), can be changed in the Designer or in the Blocks Editor.
Web Viewer	WebViewerWasserwerke, WebViewer_Chronik, WebViewer_Rohrnetz, WebViewer_Historische, WebViewer_Hausanschluss, WebViewer_Brunnen, WebViewer_Daten	Component for viewing Web pages. The Home URL can be specified in the Designer or in the Blocks Editor. The view can be set to follow links when they are tapped, and users can fill in Web forms. Warning: This is not a full browser. For example, pressing the phone"s hardware Back key will exit the app, rather than move back in the browser history….
PhoneCall	PhoneCall_SWKA	A non-visible component that makes a phone call to the number specified in the PhoneNumber property, which can be set either in the Designer or Blocks Editor.

Tabelle 15: Info Screen

Sämtliche Komponenten aus den Tabellen, welche in der Spalte “Unterkomponente“ gelistet sind werden aus den verschiedenen Kategorien der PALETTE aus dem DESIGNER in den VIEWER per Drag&Drop hineingezogen und somit auf dem Screen platziert. Im nächsten Schritt werden die Komponenten innerhalb des VIEWER strukturiert und angeordnet. Zur Anordnung dienen die in den Tabellen aufgeführten „HorizontalArrangements“. Diese Komponente dient einerseits als Platzhalter zwischen den Komponenten und andererseits als Platzhalter für bestimmte Komponenten.

Hierfür können Komponenten wie „Label“, „Buttons“ und viele mehr innerhalb der „HorizontalArrangement“ Komponente angeordnet werden. Die Eigenschaften der jeweiligen Komponente werden innerhalb des PROPERTIES Bereichs im DESIGNER angezeigt.

Im nächsten Schritt werden die einzelnen Komponenten bearbeitet. Jede Komponente hat unterschiedliche Funktionen und kann entsprechend angepasst werden. Hierfür wird eine Komponente ausgewählt. Durch die Auswahl werden die Eigenschaften der Komponente im PPROPERTIES Bereich angezeigt. Sämtliche Eigenschaften der Komponente können nun bearbeitet und angepasst werden.

Da es für jede Komponente unterschiedliche Einstellungsmöglichkeiten gibt wird in der folgenden Tabelle 16 beispielhaft auf die für den Prototypen wichtigen Veränderungen und Einstellungen der einzelnen Komponenten eingegangen.

Komponenten	Wichtigste Änderungen/Einstellungen
Screen (Eigenschaften gelten für jeden Screen)	Name des Screens AlignHorizontal: Left AlignVertical: Top BackgroundColor: None BackgroundImage: Eigener Displayhintergrund CloseScreenAnimation: Fade Icon: Eigenes Icon ScreenOrientation: Unspecified Scrollable: False Title: Die Karlsruher Trinkwasserversorgung VersionCode: 1 VersionName: 1.0

Label (Eigenschaften gelten für jedes Label)	BackgroundColor: None FontBold: True FontItalic: False FontSize: 14.0 Text: Wird auf jeweiliges Label angepasst TextAlignment: Left TextColor: White Visble: Hidden or Showing Width: Automatic Height: Automatic
HorizontalArrangement	AlignHorizontal: Left AlignVertical: Top Visible: Hidden or Showing Width: Die Breite wird jeweils auf die Anforderungen angepasst. Die Breite wird in der Einheit „Pixel“ bestimmt Height: Die Höhe wird auf die jeweiligen Anforderungen angepasst. Die Höhe wird in der Einheit „Pixel“ bestimmt
Image	Picture: Hier wird das Bild ausgewählt Visible: Hidden or Showing Width: Die Breite wird auf die jeweiligen Anforderungen angepasst. Height: Die Höhe wird auf die jeweiligen Anforderungen angepasst.
WebViewer (Eigenschaften gelten für jeden Webviewer)	FollowLinks: True HomeURL: Hier wird die URL der jeweiligen zu erreichenden Website übermittelt PromtforPermission: True Uses Location: False Uses Location: True, für den Karten Webviewer Visible: Hidden or True

	Width: 320 Pixel Height: Wird jeweils angepasst
Button	BackgroundColor: Default (Black) Enable: True FontBold: False FontIlatic: False FontSize: 14.0 FontTyperface: Default Image: Jeweiliges Bild des Buttons Shape: Default (Shape wird durch das Image bestimmt) ShowFeedback: True Text: Jeweiliger Text des Button (Kann auch durch Image bestimmt werden) TextAlignment: Left TextColor: Default (Kann auch durch Image bestimmt werden) Visible: Hidden or Showing Width: Wird auf die jeweilige Größe des Buttons angepasst. Height: Wird auf die jeweilige Höhe des Buttons angepasst
BarcodeScanner	Der BarcodeScanner enthält keine Einstellungsmöglichkeiten
LocationSensor	DistanceInterval: 0 Enable: True TimeInterval: 60000
ActivityStarter (Eigenschaften gelten für jeden ActivityStarter)	Action: android.intent.action.VIEW (Dieser Ausdruck beschreibt, dass die Applikation durch den ActivityStarter auf den Webbrowser des Endgeräts zugreifen kann)
PhoneCall	PhoneNumber: Jeweilige Telefonnummer, welche angerufen werden soll

Tabelle 16: Änderungen/Einstellungen Komponenten

Nachdem sämtliche Komponenten platziert und deren Eigenschaften bearbeitet sind, kann mit den Funktionen begonnen werden. Die einzelnen Komponenten im DESIGNER Bereich lediglich platziert und angepasst. Die Komponenten sind sichtbar und könnten als Applikation installiert werden. Die Komponenten weisen jedoch noch keinerlei Funktionalität auf. Wie in Kapitel 6.3 beschrieben wird die Funktionalität der Komponenten und somit der Applikation im BLOCKS EDITOR Bereich umgesetzt.

Aus diesem Grund wird im Folgenden lediglich auf den Programmtierteil des BLOCKS EDITOR eingegangen. Dieser besteht aus verschiedenen Blöcken welche in Form von Puzzleteilen dargestellt werden. Die einzelnen Blöcke enthalten Funktionen und Befehle aus denen der Programmcode zusammengesetzt wird.

Der folgende Abschnitt zeigt den Programmcode der unterschiedlichen Screens. Außerdem wird nach den entsprechenden Abbildungen auf die Befehle und Funktionen eingegangen.

In den Abbildungen 50 und 51 sind Funktionsweisen der Buttons des Screens 1 und der Info Hilfe Funktion dargestellt.

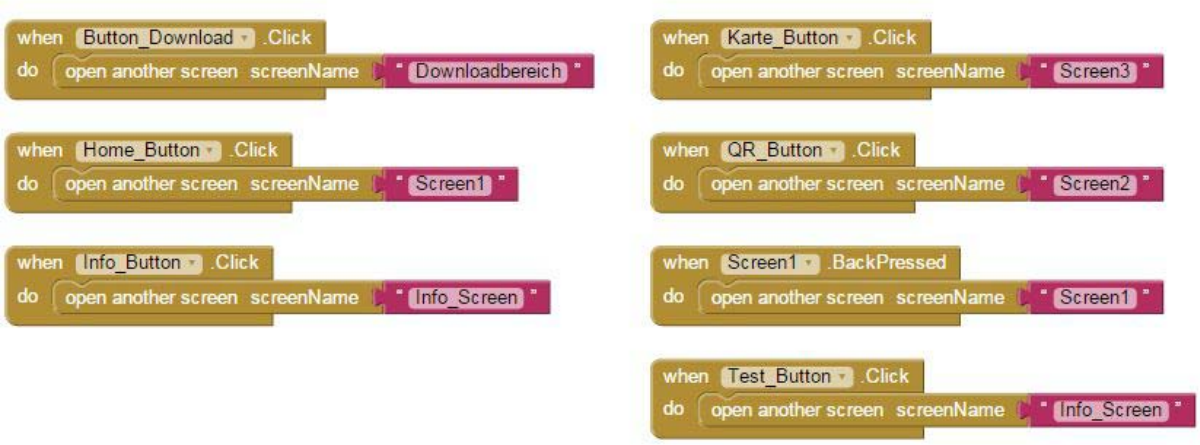


Abbildung 50: Screen 1-Buttons

Abbildung 51: Screen 1-Blocks

In Abbildung 50 ist die Funktionsweise der Buttons abgebildet. Der Befehl *„When Button.Click do“* weist dem Button die Funktion zu, welche bei einem Klick auf den Button ausgeführt werden soll. In den Fällen der Buttons auf Screen 1 haben sämtliche Buttons dieselbe Funktionsweise. Wenn der Button geklickt wird, wird ein anderer Screen geöffnet. Dies zeigt der Befehl *„open another screen“*. Mit Hilfe des Ausdrucks *„screenName“* wird der Screen aufgerufen welcher geöffnet werden soll. Dieser wird durch den Screennamen beschrieben. Der untere Bereich (Abbildung 51) beschreibt die Funktionen des *„Infos_Hilfe_Button“*. Wenn dieser Button durch einen Klick ausgewählt wird folgen eine Reihe von Aktionen. Durch den Ausdruck *„set … .Visible to True/false“* wird eine Komponente eingeblendet oder ausgeblendet. Hier ist darauf zu achten welche Ausgangsstellung die einzelne Komponente hat. Es muss beachtet werden, ob die einzelnen Komponenten bereits dargestellt werden oder verborgen sind. Außerdem wird hier bereits aufgezeigt dass es während der Entwicklung sehr wichtig ist jede einzelne Komponente sofort entsprechend und aussagekräftig zu benennen. Dies hilft bei der weiteren Entwicklung den Überblick zu behalten. Der Ausdruck *„set Info_Button.Image to Inhalt_Standard.png“* weist darauf hin, dass sich das Aussehen des Buttons bei Klick auf diesen ändert. Hierdurch wird dem Button ein neues Hintergrundbild zugewiesen.

In den folgenden Abbildungen wird lediglich auf noch nicht bekannte Programmteile und Ausdrücke eingegangen. Sämtliche Ausdrücke werden erwähnt, jedoch nur die neuen Ausdrücke und damit die neuen Funktionen erläutert.

In Abbildung 52 sind die Funktionen der Buttons des Screens 2 dargestellt.

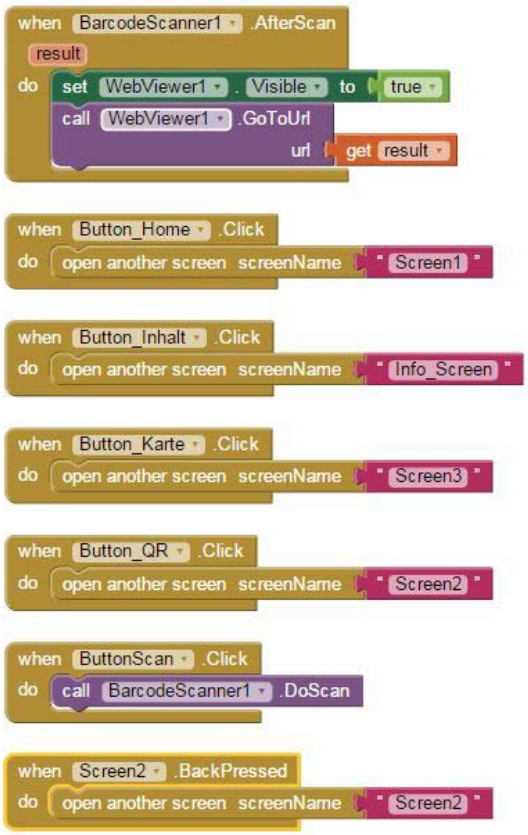


Abbildung 52: Screen 2 Blocks

Die vier Standardbuttons (Home, Inhalt, Karte und QR) weisen die schon bekannten Funktionen auf. Der Ausdruck *„when Screen2.BackPressed do open another screen screenName Screen2“* greift auf eine interne Komponente des Smartphones zu. Durch den „BackPressed“ Befehl wird der „Zurückbutton“ des Smartphone aktiviert. Wird der Zurückbutton geklickt wird ein anderer Screen geöffnet. Durch den Befehl *„when ButtonScan. Click do call BarcodeScanner1. DoScan“* wird die Barcode Scanner Komponente aufgerufen, wenn der Button geklickt wird. Außerdem wird der Barcode Scanner angewiesen einen Scan durchzuführen. Um den Barcode zu Scannen wird eine externe Anwendung auf dem Smartphone benötigt. Diese wird durch den Klick aufgerufen und der Barcode wird gescannt. Der Codeteil *„when BarcodeScanner1 AfterScan do (result) set Webviewer1.Visible to true and call WebViewer1.GoToUrl /url (geet result)“* beschreibt die Funktion des Barcodescanners nach dem Scannen des Codes. Wurde der Barcode gescannt wird unter der Variablen „result“ das Ergebnis gespeichert. Die Webviewer Komponente wird sichtbar und dieser wird eine URL zugewiesen. Diese URL wird aus der „result“-Variablen ausgelesen. Mit Hilfe der URL wird die dazugehörige Website geöffnet und somit die Informationen bereitgestellt.

In Abbildung 53 sind die Funktionen der Buttons des Screens 3 dargestellt.

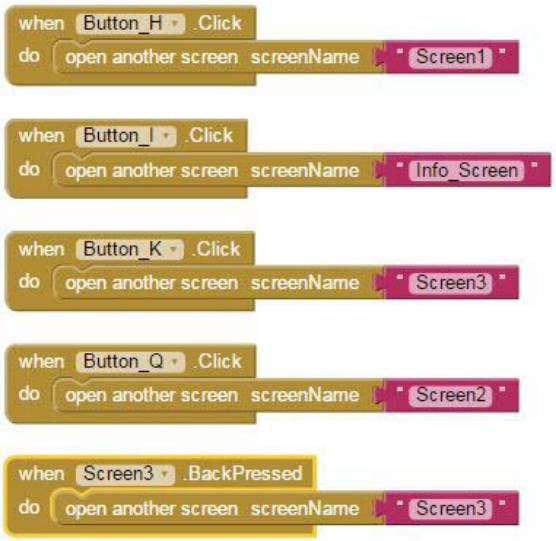


Abbildung 53: Screen 3 Hauptbuttons

Im BLOCKS EDITOR Bereich von Screen 3 werden lediglich schon bekannte Ausdrücke und Befehle verwendet. Die vier Hauptbuttons sowie der Zurückbutton des Smartphones erhalten ihre Funktionen.

In Abbildung 54 sind die Funktionen des „Screen Downloadbereichs“ dargestellt.

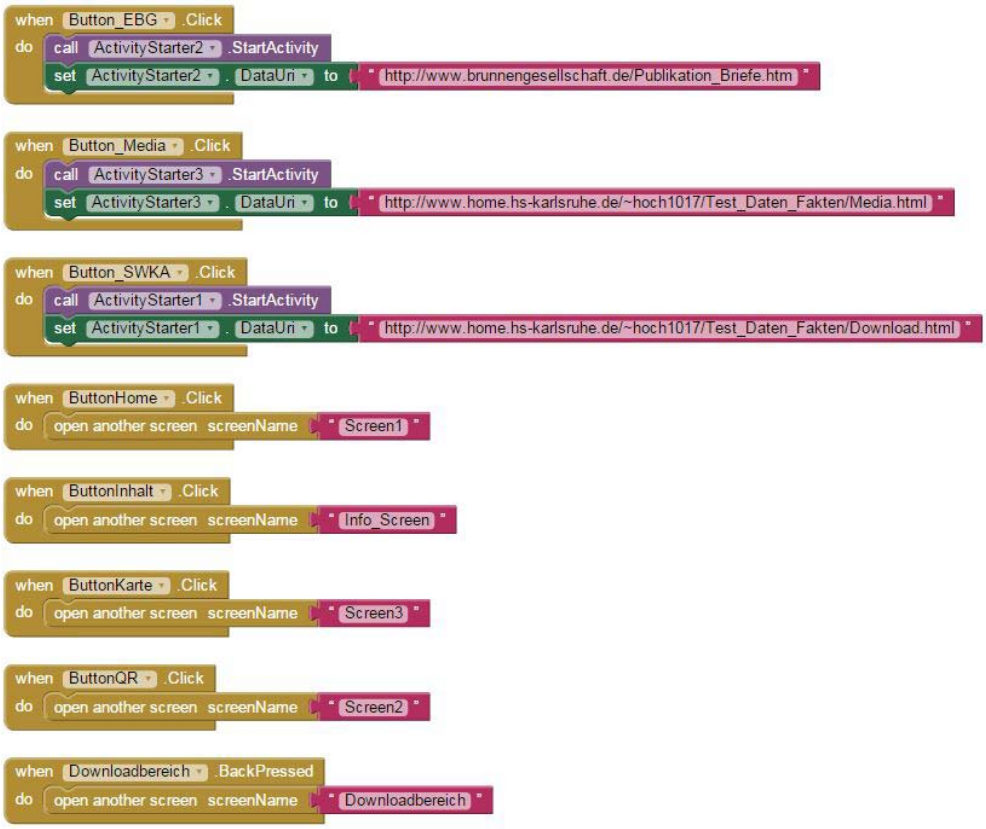


Abbildung 54: Screen Downloadbereich Blocks

Der Screen Downloadbereich enthält bereits bekannte, sowie neue Befehle. Die vier Hauptbuttons werden durch ihre bekannten Funktionen und Befehle beschrieben. Außerdem enthält dieser Screen ebenfalls die „BackPressed“ Funktion welche auf den Zurückbutton des Smartphones zugreift und diesem seine Funktion zuweist.

Die neuen Ausdrücke beschreiben ebenfalls die Funktionsweise einiger Buttons. Diese sind im oberen Bereich der Abbildung dargestellt. Die Buttons (Button_EBG, Button_Media und Button_SWKA) weisen dieselbe Funktionen auf. Bei einem Klick auf den jeweiligen Button wird ein „Activity Starter“ aufgerufen. Dieser setzt eine zugewiesene „DataUri“. Eine „DataUri“ repräsentiert hier jeweils eine URL. Durch den Befehl *„call ActivityStarter.StartActivity“* wird der „Activity Starter“ aufgerufen und eine Aktion ausgeführt. Diese Aktion wird durch den zweiten Teil des Blocks, dem Ausdruck *„set ActivityStarter.DataUri to (URL)“* beschrieben. Durch diese Befehlszeile wird dem „ActivityStarter“ eine URL zugewiesen und der Befehl erteilt diese zu öffnen. Damit der „Activity Starter“ weiß welche Aktion dieser ausführen soll ist es wichtig im DESIGNER Bereich unter der „Activity Starter“ Komponente im Eigenschaftsfeld „Action“ den Ausdruck *„android.intent.action.VIEW“* hinzuzufügen. Durch diesen Ausdruck wird die vorinstallierte Webbrowser Applikation des Smartphones aufgerufen. Der „Activtiy Starter“ öffnet den Webbrowser des Smartphones und zeigt dort die durch den Block zugewiesene URL an.

In Abbildung 55 sind die Funktionen des „Info Screen Buttons“ dargestellt.

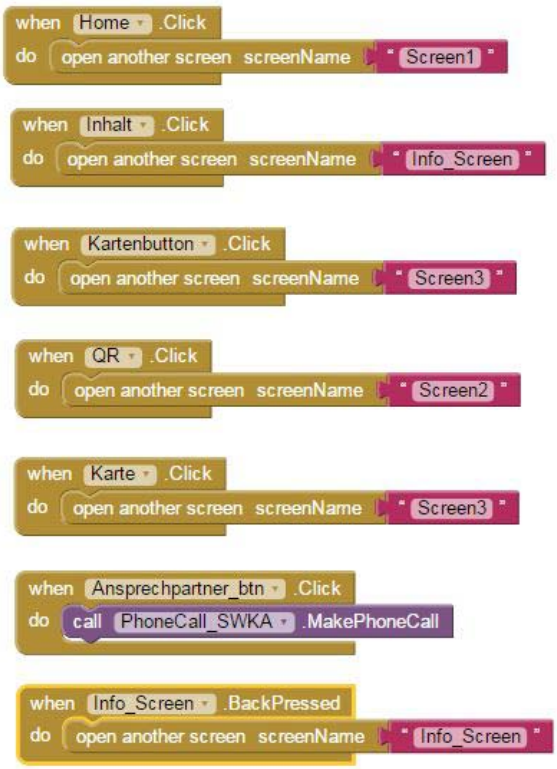


Abbildung 55: Info Screen Buttons

Die ersten Blöcke des Info Screen zeigen die bekannten Ausdrücke und Befehle der vier Hauptbuttons und des Zurückbuttons des Smartphones. Außerdem wird ein neuer Block eingeführt. Dieser beschreibt die Funktion des Ansprechpartner_btn. Wird dieser Buttons geklickt wird die Anruffunktion des Smartphones aufgerufen. Dies geschieht durch den Ausdruck *„call PhoneCall_SWKA.MakePhoneCall“*. Um diese Funktion einwandfrei nutzen zu können wird der Komponente im DESIGNER Bereich unter dem Eigenschaftsfeld „PhoneNumber“ die Telefonnummer zugewiesen welche angerufen werden soll.

Die folgenden Abbildungen 56 bis 62 zeigen die Funktionen der Buttons Brunnen, Chronik, Hausanschlüsse, Daten, Historische Anlagen, Rohrnetz und Wasserwerkewelche auf dem Info Screen angeordnet sind. Da es sinnvoll ist mit wenigen Screens in einer Applikation zu arbeiten werden auf dem Info Screen viele Komponenten angeordnet. Die meisten Komponenten werden „versteckt“ dargestellt und erst bei Klick auf den entsprechenden Button aufgerufen. So besteht der Info Screen aus den vier Hauptbuttons sowie sieben weiteren Buttons. Diese Ansicht wird bei jedem Aufruf des Info Screens gezeigt. Wird nun eine Funktion durch die Auswahl eines Buttons aufgerufen ändert sich die Ansicht. Die vorher sichtbaren Komponenten werden versteckt dargestellt und andere Komponenten werden sichtbar.

Dies wird durch den schon bekannten Ausdruck *„set … .Visible to true/false“* realisiert. Nach jedem Klick auf einen Button öffnet sich eine zugeordnete Webviewer Komponente. Hinter dieser Komponente verbirgt sich jeweils eine separate URL, welche durch den Webviewer angezeigt wird. Die jeweilige URL ist statisch und wird der Webviewer Komponente im Designer Bereich unter dem Eigenschaftsfeld „HomeUrl“ zugewiesen.

Durch dieses Verfahren wird die Anzahl der Screens reduziert und die Applikation benötigt nicht zu viel Speicher auf dem Endgerät.

Die vier Hauptbuttons werden bei jedem Klick auf einen Button angezeigt. Dies hat den Sinn dass die gewohnte Navigation durch die Applikation für den Nutzer erhalten bleibt. Um die Proportionen und die Einteilung bezogen auf die Displaygröße beizubehalten ändern sich vor allem die „HorizontalArrangement“ Komponenten. Außerdem werden die verschiedenen Buttons welche zuvor angezeigt wurden ausgeblendet und das Label ändert sich je nach Auswahl des entsprechenden Buttons und des Themengebiets welcher dieser aufruft.

Abbildung 56: Info Screen Blocks-Brunnen

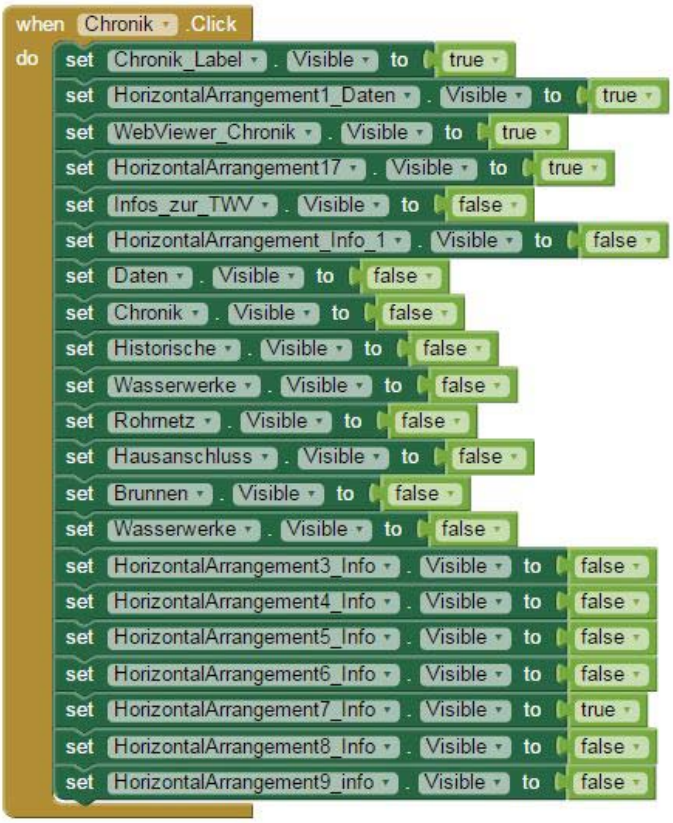


Abbildung 57: Info Screen Blocks-Chronik

Abbildung 58: Info Screen Blocks-Hausanschluss

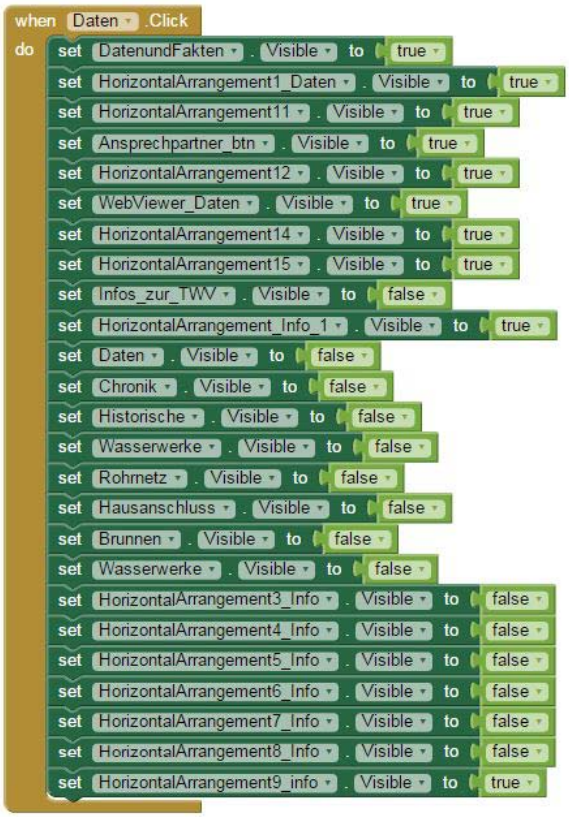


Abbildung 59: Info Screen Blocks-Daten

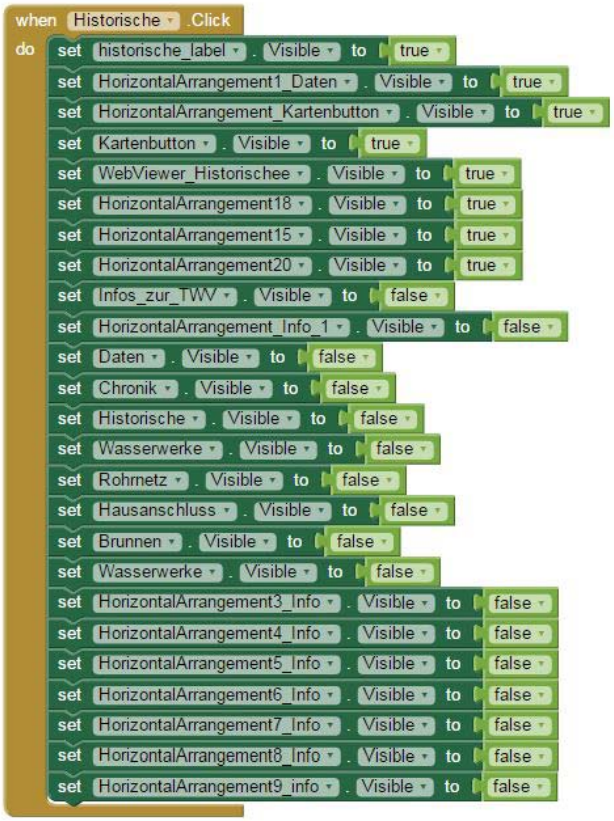


Abbildung 60: Info Screen Blocks-Historische Anlagen

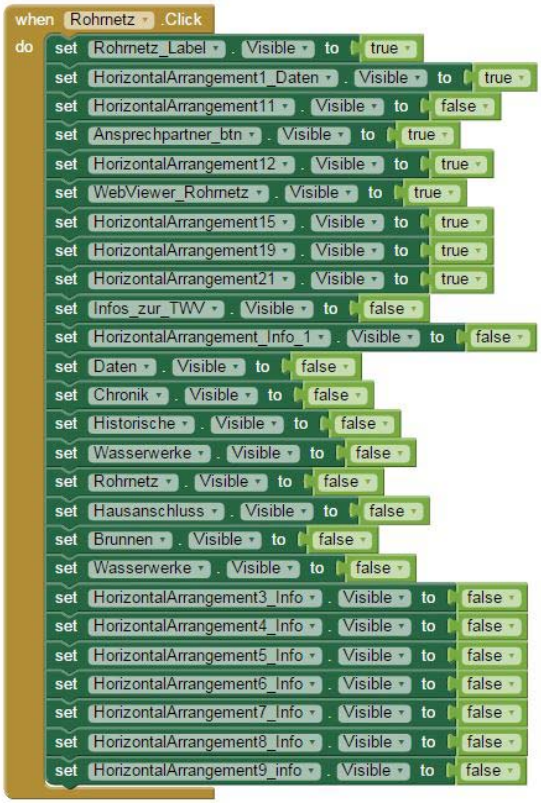


Abbildung 61: Info Screen Blocks-Rohrnetz

Abbildung 62: Info Screen Blocks-Wasserwerke

Bei der Entwicklung mit dem BLOCKS EDITOR muss darauf geachtet werden, dass sich keine Codeblöcke überschneiden oder doppelt angelegt werden. Dies beeinflusst die Funktionsweise der Applikation erheblich. Ein weiterer wichtiger Punkt ist, dass einige Komponenten zusätzlich durch ihre Eigenschaftsfelder im DESIGNER Bereich beeinflusst werden. Werden diese Einstellungen nicht vorgenommen, kann die Applikation nicht auf den vollen Funktionsumfang zurückgreifen.

Um doppelte oder fehlerhafte Codeblöcke zu vermeiden wird im BLOCKS EDITOR Bereich ein Hinweisfeld eingeblendet. Dieses Hinweisfeld zeigt mögliche Konflikte und erhebliche Probleme, die durch fehlerhafte Blöcke entstehen können. Die Abbildung 63 zeigt das Hinweisfeld. Beendet man die Entwicklung im BLOCKS EDITOR sollte dieses Hinweisfeld keine Fehler oder Konflikte enthalten.

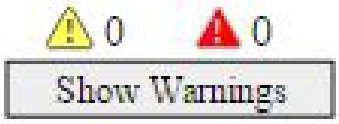


Abbildung 63: Show Warnings

6.5 Beschreibung des erstellten Prototypen

Im folgenden Kapitel wird der erstellte Prototyp kurz und allgemein beschrieben. Der Prototyp wird hinsichtlich seiner Umsetzung bewertet. Außerdem wird auf mögliche Probleme bei der technischen Realisierung des Prototypen eingegangen.

Das Grundgerüst der Anwendung wurde mit Hilfe der Entwicklungsumgebung MIT App Inevntor erstellt. Diese beinhaltet das komplette Layout sowie die Grundfunktionen wie die einzelnen Screens, Buttons und Funktionen. Die gesamte Logik und Funktionsweise wird mit Hilfe der Entwicklungsumgebung realisiert.

Um den Pflege- und Aktualisierungsaufwand so gering wie möglich zu halten werden sämtliche Informationen, welche die Anwendung bietet, mit Hilfe von HTML-Seiten bereitgestellt. Dies geschieht in der Anwendung mit Hilfe von sogenannten „Webviewern“. Diese bieten die Möglichkeit HTML-Seiten, welche im Web oder auf Servern liegen, darzustellen. Hierdurch werden Aktualisierungen der Informationen lediglich auf den HTML-Seiten vorgenommen und durch die Webviewer in der Anwendung bereitgestellt.

Die folgende Abbildung zeiget verschiedene Screenshots der Displays der Anwendung. Diese sind nach den vier Hauptbuttons und den dazugehörigen Untergruppen geordnet. In Abbildung 64 ist der Home Screen (links) zusammen mit den bei Aufruf der Buttons „Zu den Inhalten“ … und „Infos & Hilfe“ angezeigten Bildschirmen dargestellt. Außerdem wird der Screen„Media & Downloads“ angezeigt.

Home

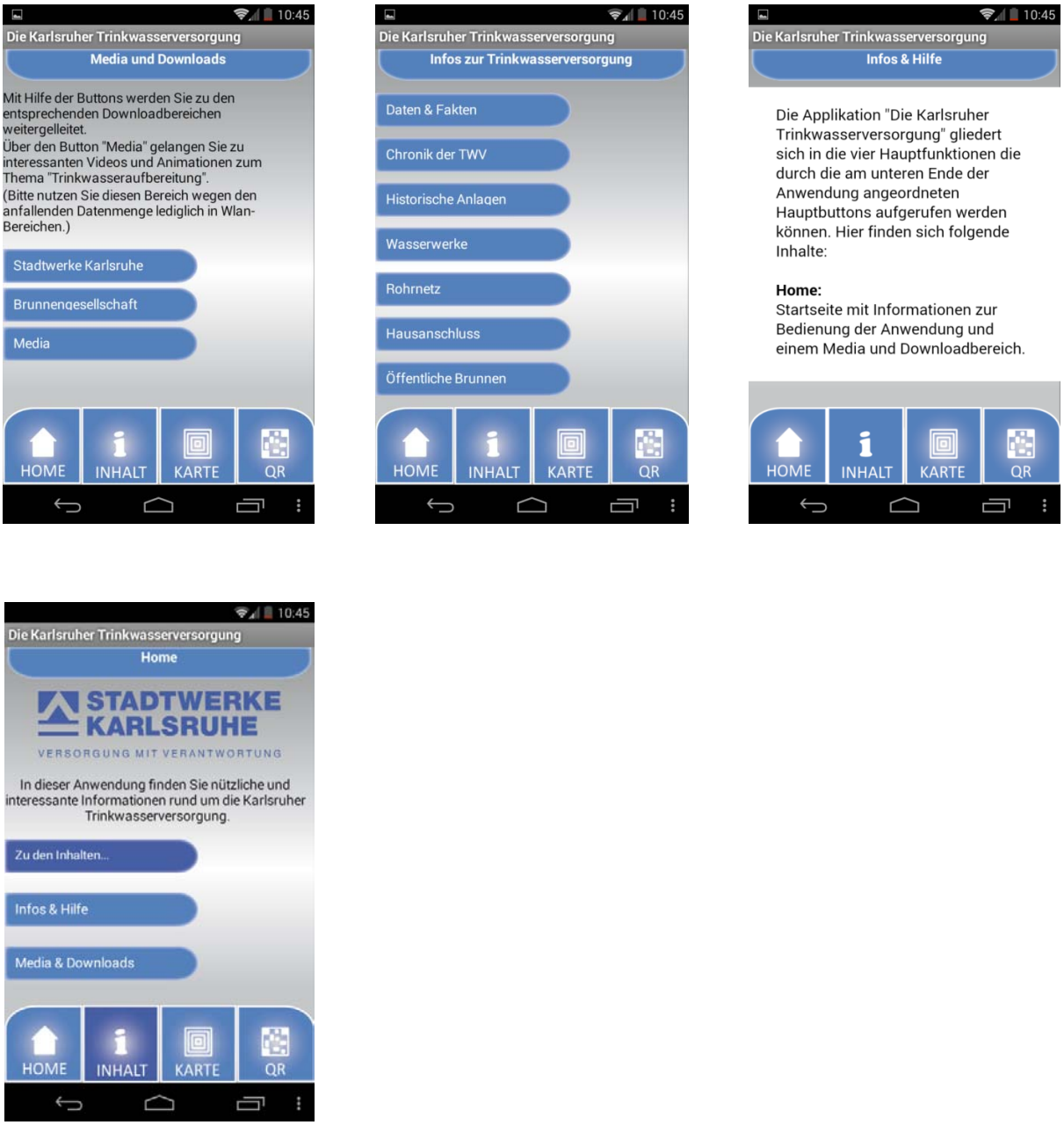


Abbildung 64: Screenshots Home

In der Abbildung 65 sind die Screens des Buttons „INHALT“ im Hauptmenü (unten am Bildschirm) und der beim Aufruf der Inhalte (horizontale Buttons) dargestellt.

Inhalt

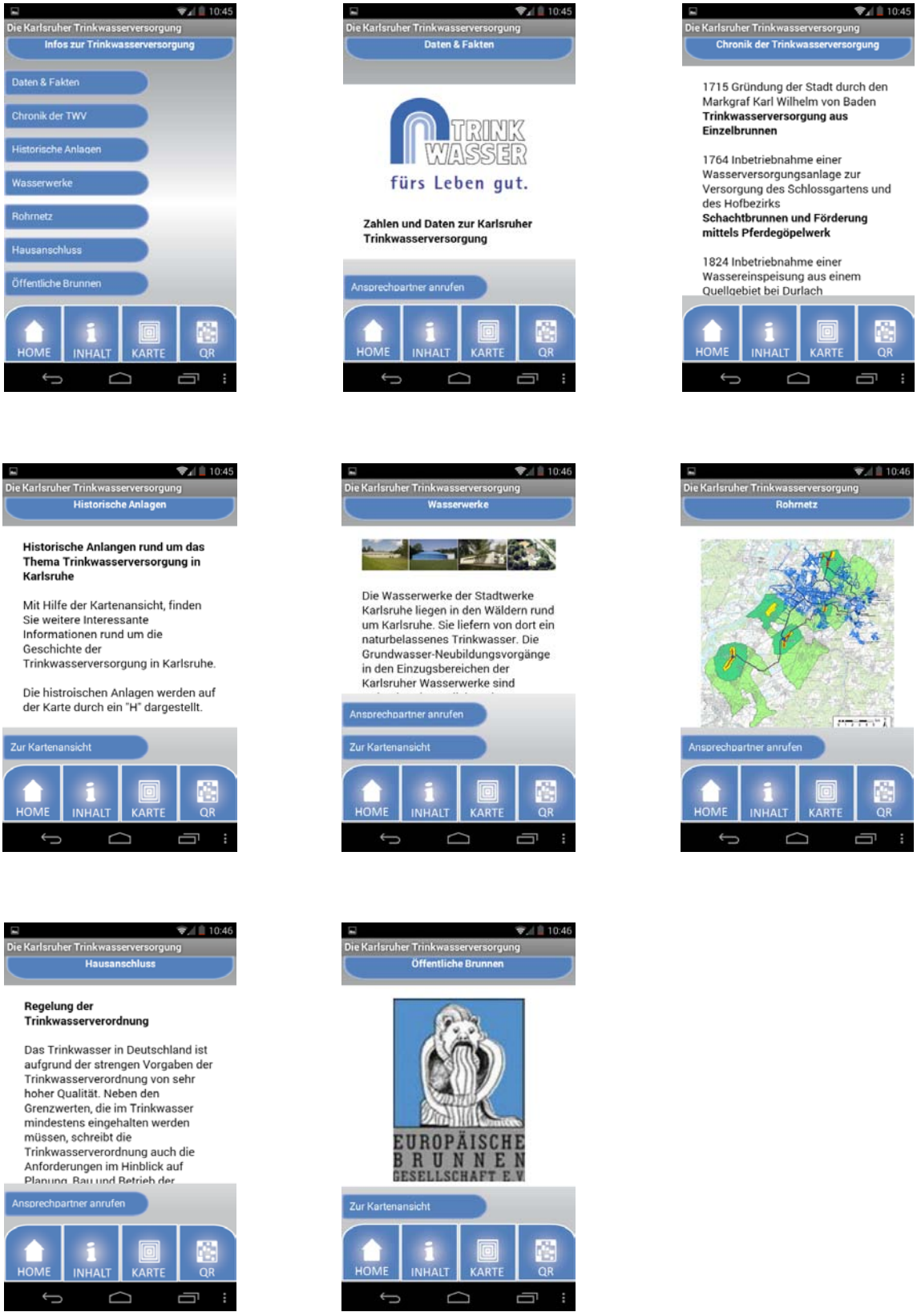


Abbildung 65: Screenshots Inhalt

In Abbildungen 66 sind die Screens des Buttons „KARTE“ im Hauptmenü (unten am Bildschirm) und der nach Aufruf angezeigten Kartendarstellungen dargestellt.

Karte

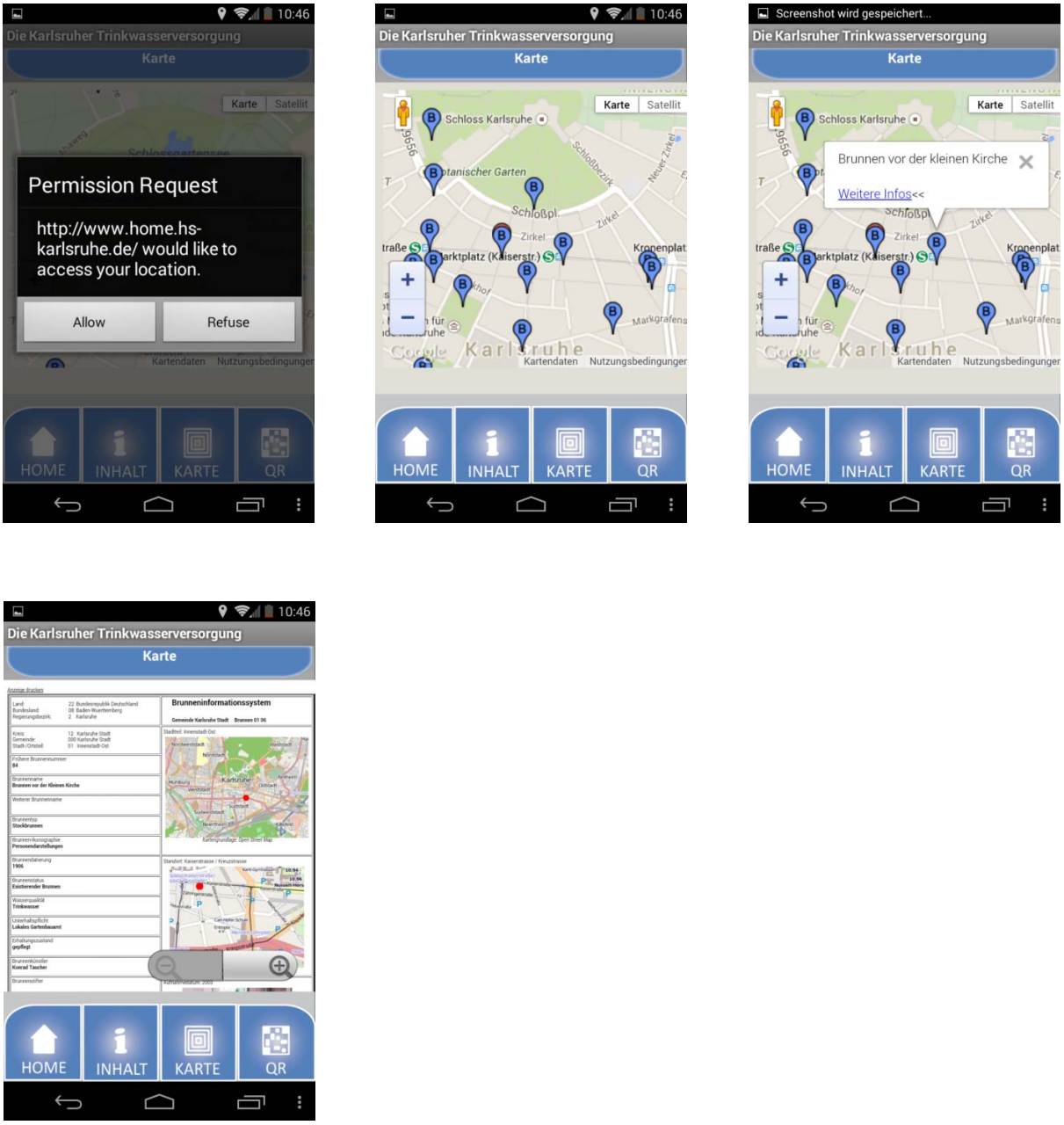


Abbildung 66:: Screenshots Karte

In Abbildungen 67 ist der Bildschirm zum Aufruf des QR-Codescanners dargestellt, der nach Auswahl des Buttons „QR“ im Hauptmenü (unten am Bildschirm) angezeigt wird.

QR

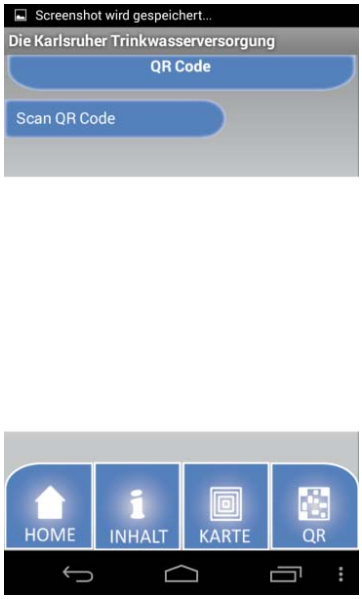


Abbildung 67: Screenshot QR

Der Prototyp konnte im Rahmen dieser Masterthesis voll funktionsfähig umgesetzt werden. Sämtliche technische Funktionen wurden implementiert und die von den Stadtwerken Karlsruhe GmbH bereitgestellt Informationen eingearbeitet .
Während der technischen Umsetzung haben sich Probleme ergeben, welche größtenteils gelöst werden konnten.

Nachfolgend werden einige Probleme aufgelistet:

- Anzahl der Screens
- Handling der Screens untereinander
- Fehlende Funktionen welche nur durch den Zusammenschluss einzelner Funktionen technisch umgesetzt werden können
- Unterteilung des Display
- Anordnung der Komponenten auf dem Display
- Displayanordnungen welche sich auf verschiedene Displaygrößen anpassen

Auch ist zu beachten ist, dass sich die Entwicklungsumgebung MIT App Inventor in einer ständigen Weiterentwicklungsphase befindet. Die meisten Funktionen und Komponenten welche zur Gestaltung eine Applikation benötigt werden sind bereits in einem gewissen Stadium in der Entwicklungsumgebung vorhanden. Dies betrifft vor allem die Größenänderung und Anpassung der Komponenten an verschiedene Displaygrößen. Der Funktionsumfang der Entwicklungsumgebung wird ständig fortgeschrieben und weitere Funktionen werden vom Softwareentwicklerteam in Zukunft realisiert. Aus diesem Grunde müssen bei einigen technischen Umsetzungen Kompromisse eingegangen werden. Hierzu zählt vor allem die Anordnung der einzelnen Komponenten auf dem Viewer Bereich im „Designer“. Zum jetzigen Zeitpunkt ist es sehr kompliziert sämtliche Komponenten untereinander anzuordnen. Ein weiterer Punkt, welcher zu Problemen führt, ist die Anpassung der Applikation auf verschiedenen Displaygrößen. Während der Entwicklung einer Applikation wird ein bestimmtes Endgerät hierfür verwendet. Die komplette Applikation wird auf dieses Endgerät hinsichtlich der Größe der Komponenten und deren Anordnung angepasst. Wird die Applikation für andere Endgeräte bereitgestellt kann es zu unterschiedlichen Darstellungen kommen. Aus diesem Grund ist es wichtig die Applikation vor der Veröffentlichung nochmals mit Hilfe der in Kapitel 6.3 beschriebenen Software „Marketizer“ zu bearbeiten. Das Entwicklerteam widmet sich diesem Problem im Rahmen der Entwicklungsarbeit von App Inventor bereits heute und eine Lösung wird voraussichtlich in naher Zukunft bestehen. Ein weiteres Problem welches während der Entwicklungsarbeit auftrat besteht darin, dass die Webviewer Komponente nicht alle heute üblichen Techniken umsetzten kann. So ist es nicht möglich in der Webviewer Komponente eine HTML-Seite mit Silbentrennung anzuzeigen. Die Silbentrennung wird von der Komponente nicht unterstützt. Aus diesem Grund werden HTML-Seiten mit Silbentrennung sehr „verzerrt“ dargestellt.

Verschiedene Probleme werden im Rahmen der Weiterentwicklung der Entwicklungsumgebung MIT App Inventor schon heute angegangen. Eine Garantie, dass diese beim nächsten Update gelöst werden besteht wegen der Weiterentwicklung in einer Open Source Community jedoch nicht.

7 Bewertung der Ergebnisse und Test der Anwendung

Das folgende Kapitel bewertet die Endergebnisse bei der Umsetzung der Anforderungen in den Prototypen. Dazu wird ein Vergleich durchgeführt. Weiterhin werden Verbesserungsvorschläge unterbreitet.

Um nachzuvollziehen welche Anforderungen an den Prototypen gestellt wurden und wie diese technisch umgesetzt werden konnten, wird der in Kapitel 5.1 erarbeitete Anforderungskatalog mit dem Ergebnis der Umsetzung in Tabelle 17 verglichen. Anforderungen, die erfüllt werden konnten werden durch ein grün unterlegtes Feld dargestellt. Anforderungen, die nur teilweise oder gar nicht umgesetzt werden konnten werden durch ein gelb, beziehungsweise rot unterlegtes Feld markiert.

Anforderungen an die Anwendung	Beschreibung
Entwicklungsaufwand	Der Entwicklungsaufwand muss aus Zeitgründen gering gehalten werden
Kompatibilität/ Verfügbarkeit	Die Anwendung soll möglichst kompatibel auf mehreren mobilen Betriebssystemen zum Einsatz kommen
Performance/ Zuverlässigkeit	Die Performance der Anwendung soll möglichst hoch sein. Dies betrifft lange Ladezeiten etc. Außerdem soll die Anwendung stabil laufen. Abstürze der Anwendung sollen vermieden werden
Wartung/ Erweiterbarkeit	Der Wartungsaufwand soll niedrig und einfach gehalten werden. Es muss möglich sein neue Inhalte einfach, schnell und ohne Programmierkenntnisse in die Anwendung einfügen zu können
Datenschutz/ Verschlüsselung	Falls die Anwendung personenbezogene Daten enthält. Sollen diese für Dritte unzugänglich sein. Außerdem müssen sämtliche datenschutzrechtlichen Daten verschlüsselt sein
Informationsübermittlung/ Bedienbarkeit	Die Anwendung muss einfach zu bedienen sein. Jeder Nutzer muss ohne vorherige Kenntnisse in der Lage sein, die Anwendung intuitiv zu bedienen. Für eine anwenderfreundliche App müssen die zu übermittelnden Informationen aufbereitet werden
Zielgruppe	Die Zielgruppe muss klar definiert sein. Die Zielgruppe der Anwendung sind Smartphoneuser welche sich über die Trinkwasserversorgung informieren möchten. Hiermit ist die Zielgruppe von Schülern bis zu älteren Benutzern gefächert.
Lizenz/Kosten	Die Lizenz der Anwendung soll frei sein. Die Anwendung soll informativen und werbetechnischen Gründen dienen. Hierfür muss diese frei erhältlich für die Nutzer sein

Kartendienst	Die Anwendung muss einen zuverlässigen und bekannten Kartendienst beinhalten.
QR-Code	Die Anwendung muss einen QR-Code-Reader enthalten
Anforderungen an die Software:	
Zeitaufwand	Der Entwicklungsaufwand soll aus Zeitgründen gering gehalten werden
Lernkurve	Die Lernkurve zum Erlenen der Software soll steil ansteigen. Der Umgang mit der Software muss schnell erlernt werden können
Performance/ Zuverlässigkeit	Die Performance der Software soll möglichst hoch sein. Dies betrifft lange Ladezeiten etc. Außerdem soll die Software stabil (ohne Abstürze) laufen
Kosten	Die Kosten für die Anschaffung der Software sollen gering gehalten werden. Hier empfiehlt sich wenn möglich ein Open Source Produkt
Datenschutz	Die Software soll die eingebundenen Daten nicht an den Softwareentwickler oder Dritte weitergeben
Erweiterbarkeit	Die Software soll Erweiterungsmöglichkeiten bieten. Dies beinhaltet weitere Updates der Software und mögliche Add-ons
Gestaltung	Die Anwendung soll frei gestaltet werden können. Die Software soll keinerlei gestalterischen Vorgaben/Grenzen geben
Zugriff auf API des mobilen Betriebssystems	Die Software soll den Zugriff auf das mobile Betriebssystem unterstützen. Sensoren des Smartphones sollen eingebunden werden können
Anforderungen an das Betriebssystem:	
Offener Quellcode/ Programmcode	Der Quellcode des Betriebssystems soll aus Programmierungsgründen entweder komplett oder in Teilstücken frei erhältlich sein
Performance/ Zuverlässigkeit	Die Performance des Betriebssystems soll möglichst hoch sein. Dies betrifft lange Ladezeiten etc. Außerdem soll das Betriebssystem stabil (ohne Abstürze) laufen

Kosten	Die Kosten für die Entwicklung von Anwendungen für das spezielle Betriebssystem sollen gering gehalten werden. Dies betrifft vor allem die Kosten für die spätere Verbreitung der Anwendung (App Store) und die Einschränkungen der Anwendungen des Betriebssystemherstellers
Synchronisation	Die Synchronisation zwischen dem Endgerät und dem Betriebssystem soll einfach z.B. über das Internet geschehen
Aktualisierung	Das Betriebssystem soll regelmäßig aktualisiert werden. Dies garantiert ebenfalls eine breite Verfügbarkeit des Systems durch die Endnutzer

Tabelle 17: Vergleich Anforderungen und technische Umsetzungen

Es wird deutlich, dass die meisten Anforderungen erfüllt werden konnten. Die Kompatibilität der Applikation konnte nicht erfüllt werden. Dies liegt vor allem daran, dass die Anwendung schon durch die Wahl der Entwicklungsumgebung MIT App Inventor eingeschränkt wurde. Durch diese Entwicklungsumgebung werden Applikationen entwickelt, die für das mobile Betriebssystem Android vorgesehen sind. Die Anforderung an den Datenschutz/Verschlüsselung konnte nur teilweise umgesetzt werden. Daher wurden in der Anwendung keine nutzerbezogene Daten integriert. Somit kann diese Anforderung vernachlässigt werden. An die Entwicklungsumgebung wird die Anforderung gestellt, dass der Gestaltung der Applikation keinerlei Grenzen gesetzt werden. Diese Anforderung wird ebenfalls nur teilweise umgesetzt. Die Entwicklungsumgebung MIT App Inventor setzt Grenzen bei der Gestaltung. Wie zuvor beschrieben sind die Gestaltungsmöglichkeiten der Entwicklungsumgebung begrenzt, aber ausreichend.

Diese werden vor allem durch die begrenzte Möglichkeit der Anordnung der Komponenten deutlich. Die Grenzen betreffen jedoch lediglich die Anordnung. Gestalterische Grenzen bezüglich des Aussehens der Applikation gibt es nicht.

Der Prototyp wurde auf mehreren Endgeräten durch unbedarfte Testperson getestet. Die Applikation wurde auf einem Nexus Smartphones entwickelt und auf dieses angepasst. Um den Prototypen zu testen wurde dieser ebenfalls auf einem Smartphone der HTC Desire Baureihe, einem Samsung Galaxy S2 und einem Samsung Galaxy S5 verschiedener Testpersonen installiert. Um die Anwendung ebenfalls auf Tablets zu testen wurde diese auf einem Samsung Galaxy Tab 2 7.0, einem Samsung Galaxy Tab 3 10.0 und einem Amazon Kindle Fire HD einiger Testpersonen installiert.

Die folgende Abbildungen 70 zeigt zum Vergleich die Darstellung des Home Screen der Applikation auf einem Smartphone der Google Nexus Baureihe (links) und dem Smartphone der HTC Desire Baureihe rechts.

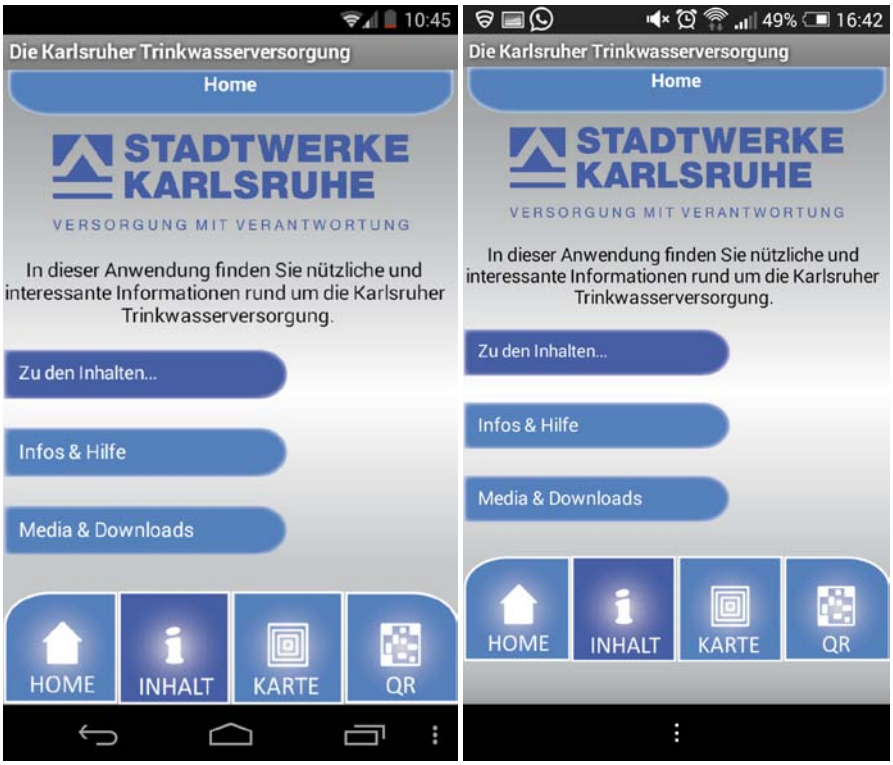


Abbildung 68: Vergleich Nexus-HTC

Im direkten Vergleich wird deutlich, dass die rechte Abbildung einen Abstand zwischen den Hauptbuttons und dem unteren Displayrand aufweist. Diese Eigenschaft ist bei sämtlichen getesteten Geräten festzustellen. Dieser Abstand entsteht durch die unterschiedliche Anordnung der Hauptbuttons bei unterschiedlichen Smartphonetypen. Heute werden Smartphones angeboten, deren Hauptbuttons entweder im Randbereich des Displays des Smartphones oder am unteren Displayrand angeordnet sind. Sind diese Buttons am unteren Displayrand angeordnet würde die Applikation die Benutzung des Smartphone behindern, da sie die eigentlichen Hauptbuttons des Smartphones überlagern würde. Aus diesem Grund wird der Abstand dargestellt. Dies hilft dabei die Applikation für verschiedene Endgeräte bereitstellen zu können und dabei sicherzustellen dass die Funktionsweise in keiner Weise eingeschränkt wird.

Auf sämtlichen Endgeräten konnte die Anwendung problemlos installiert und erfolgreich getestet werden. Es gab keinerlei Einschränkungen betreffend der stabilen Nutzung und Performance. Die Reaktionen der Testpersonen waren durchweg positiv. Die Meinung der Testpersonen bezogen auf die Informationsübermittlung des Prototypen war ebenfalls positiv. Laut den Testpersonen kann die Anwendung intuitiv bedient werden und läuft stabil.

Soweit die Testpersonen es beurteilen konnten, ähnelt die Anwendung dem Aussehen der Homepage der Stadtwerke Karlsruhe GmbH. Einigen Testpersonen ist der Freiraum zwischen dem unteren Displayrand und den Hauptbuttons aufgefallen, welcher oben genannt und erläutert wurde. Dies empfanden die Testpersonen jedoch nicht als störend.

Der erstellte Prototyp bildet eine solide Grundlage um mit relativ geringem Aufwand eine lauffähige Anwendung zu erstellen. Sämtliche Funktionen für eine lauffähige Anwendung wurden bereits in den Prototypen integriert und erfolgreich umgesetzt. Um die Anwendung zu finalisieren müssen weitere Informationen eingepflegt und die schon bestehenden überarbeitet werden. In den Prototypen wurden lediglich die wichtigsten Brunnen aus Karlsruhe integriert. Diese Auswahl der EBG müsste nochmals erweitert werden.

8 Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde ein modernes und zeitgerechtes Konzept zur Erstellung einer Applikation für die Stadtwerke Karlsruhe GmbH im Bereich Trinkwasserversorgung und für ausgewählte Trinkwasserlaufbrunnen der europäischen Brunnengesellschaft (EBG) entwickelt und in einen lauffähigen Prototypen umgesetzt. Dabei wurde besonderes Augenmerk auf die Evaluierung geeigneter Software gelegt, um das Projekt in dem beschränkten Bearbeitungszeitraum einer Masterthesis zur realisieren. Neben dem konzeptionellen Design und der softwaretechnischen Realisierung wurden umfangreiche Sachdaten integriert.

Im Einzelnen wurde auf der Grundlage der vorhandenen Daten der Stadtwerke Karlsruhe GmbH, bzw. der EBG ein Konzept entwickelt um interessierten Bürgern Informationen über die Trinkwasserversorgung von Karlsruhe bereitzustellen. Diese wurde so aufbereitet, dass die Informationen zeitgerecht im Rahmen einer Applikation bereitgestellt werden konnten. Neben der reinen Informationsbereitstellung, beispielsweise über technische oder historische Sachverhalte zu den Anlagen der Trinkwasserversorgung, wurde besonderer Wert auf die Herstellung des räumlichen Bezugs in der Stadt gelegt. Dazu wurde eine Kartenkomponente integriert, in der die interessanten Anlagen durch Marker verortet sind und bei deren Auswahl eine direkte Verlinkung mit den Informationen hergestellt werden kann. Somit können dem Benutzer Informationen über historische Anlagen sowie Brunnen und Wasserwerke aus einer Karte heraus bereitgestellt werden. Außerdem wurde im Rahmen des Prototypen ein QR Code Scanner in die Applikation integriert. Es ist vorgesehen an den Anlagen entsprechende QR-Codes anzubringen, so dass beim Scannen dieser Codes die Information über das mobile Endgerät direkt beim Betrachten der Anlage vor Ort zur Verfügung stehen.

Der Prototyp konnte im möglichen Zeitfenster einer Masterthesis fertiggestellt werden. Darüber hinaus wurden sämtliche wichtige Informationen für die Applikation aufbereitet und bereitgestellt. Auf Grundlage des erstellten Prototypen kann eine lauffähige Anwendung mit überschaubarem zusätzlichen Aufwand realisiert werden.

Als zusätzliche Weiterentwicklung wäre ein Quizspiel denkbar, welches in die Anwendung integriert werden könnte. Dieses Quizspiel könnte als Art „Geocaching[1]“ umgesetzt werden. An ausgewählten Brunnen könnten Zusatzinformationen angebracht werden welche dem Nutzer die Lösung eines Quiz ermöglichen.

[1] Geocaching beschreibt eine Art der elektronischen Schatzsuche. An verschiedenen Orten, die durch ihre geografischen Koordinaten festgelegt sind, werden Hinweise oder kleinere „Schätze“ versteckt welche mit Hilfe von GPS-Geräten gesucht werden.

In Zukunft werden Endgeräte weiterentwickelt und neue Technologien entstehen.

Zum heutigen Zeitpunkt richtet sich der Fokus auf die neue Technologie „Google Glass“. Das Projekt ist momentan jedoch, vor allem aus datenschutzrechtlichen Gründen, sehr umstritten und noch nicht komplett ausgereift. Google Glass befindet sich im Moment in einer „offenen Betaphase“. Das Produkt ist in England und den USA für ungefähr 1500 Euro erhältlich. In Deutschland ist die „Google Brille“ aus den angesprochenen datenschutzrechtlichen Gründen noch nicht verfügbar. Die Funktionen beschränken sich auf die Telefonfunktion, Navigation, verschiedene internetgestützte Suchanfragen, sowie die Aufnahme von Fotos und Videos. Die Vision und Zukunft des Google Glass Projekts liegt jedoch in Anwendungen, welche durch „augmented reality“[1]-Komponenten geprägt werden. Zukünftige Anwendungen könnten dem Nutzer solcher Technologien Zusatzinformationen in Echtzeit bereitstellen. Zukünftige Anwendungen könnten somit auf eine neue Ebene der Informationsübermittlung gehoben werden. Man stelle sich nur vor mit einer Google Glass oder einer ähnlichen Technologie durch Karlsruhe zu laufen und sämtliche Informationen, welche im Moment durch die Applikation bereitgestellt werden, direkt vor den Augen zu haben. So könnten zum Beispiel dreidimensionale Modelle der Brunnen erstellt und in die Anwendung integriert werden. Hier ist es denkbar dem Nutzer zusätzlich die technische Funktionsweise der einzelnen Brunnen oder interessante Zusatzinformationen direkt zu vermitteln.

[1] Unter augmented reality (erweiterte Realität) wird die computergestützte Erweiterung der Realität verstanden.

Abbildungsverzeichnis

Die Abbildungen 1 bis 24 wurden aus Screenshots der genannten Apps erstellt.

Abbildung 25: [Online]. - 12.01.2015 - http://www.techstage.de/news/Marktanteile-iOS-schrumpft-Windows-Phone-legt-zu-2098360.html

Abbildung 27: [Online]. - 12.01.2015 - http://www.qrcode-monkey.de/

Die Abbildungen 26 und 28 bis 35 wurden selbst erstellt.

Die Abbildungen 36 bis 46 und 49 bis 63 wurden aus Screenshots der Entwicklungsumgebung MIT App Inventor erstellt.

Die Abbildungen 47 und 48 wurden aus Screenshots der Software Marketizer erstellt.

Die Abbildungen 64 bis 68 wurden aus Screenshots des Prototypen erstellt.

Tabellenverzeichnis

Tabelle 6: [Online]. - 12.01.2015 - http://en.wikipedia.org/wiki/Comparison_of_web_map_services Die Tabelle wurde anschließend bearbeitet.

Tabelle 1 bis 5 und 7 bis 17 wurden selbst erstellt.

Quellenverzeichnis

Literatur

Dietrich Maier. [Buch]; 2004: „Karlsruher Brunnen Bilder – Modelle – Fotografien“. Karlsruhe: Swiridfoff Verlag

Jan-Dirk Rausch (Text), Heinz Gockel (Fotos). [Buch]; 2006: „Durlachs historische Bauten“. Karlsruhe: Bürker Verlag

Katja Förster, Markus Gruber, Matthias Maier. [Buch]; 2011: „Märkte und ihre Brunnen“. Karlsruhe: Info Verlag

Kloss, J. H. [Buch]; 2011: „Android-Apps Programmierung für Einsteiger“. München: Markt+Technik Verlag

Mirko Felber, Matthias Maier, Anke Mührenberg. [Buch]; 2006: „Wasser-Geschichte der Wasserversorgung in Durlach“. Karlsruhe: Info Verlag

Prof. Dr. Dietrich Maier, Hans Eberhardt, Prof. Dr. Matthias Maier, Dr. Bernd Hofmann, Ulrike Erdrich. [Buch]; 2011: „Chronik der Wasserversorgung von Durlach und Karlsruhe". Karlsruhe: Druck und Verlagsgesellschaft Südwest mbH

Online

appTITAN [Online]. - 14.01.2015 - http://shop.apptitan.de/app-lizenzen/1/apptitan-app

Appyourself [Online]. – 14.01.2015 - http://appyourself.net/de/preise

Bing [Online]. - 14. August 2014 - http://www.bing.com/maps

BING [Online]. - 1. September – (PDF) „Ihr Leitfaden für Bing Maps for Enterprise“ - https://partner.microsoft.com/download/germany/40167821

Developergarden [Online]. - 8. August 2014 - https://www.developergarden.com/de/blog/artikel/article/native-vs-web-app-vs-hybrid-was-ist-die-perfekte-entwicklerstrategie/

Google Developers [Online]. - 16. August 2014 - https://developers.google.com/maps/faq?hl=de#usagelimits

Google Maps [Online]. - 12. August 2014 - https://www.google.de/maps

Gruenderszene [Online]. - 7. August 2014 - http://www.gruenderszene.de/lexikon/begriffe/app

Here [Online]. - 1. Septmeber 2014 - http://here.com/terms/service-terms/?lang=de-DE

HERE [Online]. - 14. August 2014 - http://here.com

ibuildapp [Online]. - 14.01.2015- http://ibuildapp.com/pricing.php

Inventor, M. A. [Online]. - 25. August 2014 - http://appinventor.mit.edu

IT Wissen [Online]. - 5. August 2014 - http://www.itwissen.info/definition/lexikon/quick-response-QR-QR-Code.html

Open Street Map [Online]. - 10. August 2014 - http://openstreetmap.de/karte.html

PC Welt [Online]. - 20. August 2014 - http://www.pcwelt.de/ratgeber/OSM__Google_Maps__Nokia_Here-Die_besten_Kartendienste_fuer_den_Smartphone-Browser__mit_Galerie_-7115972.html

Projektmagazin [Online]. - 6. August 2014 - https://www.projektmagazin.de/glossarterm/anforderungskatalog

PrototypeJS [Online]. - 17. Januar 2015 - http://prototypejs.org

QR Code Generator [Online]. – 3. August 2014 - http://www.qrcode-generator.de/

QR Code Kaywa [Online]. - 3. August 2014 - http://qrcode.kaywa.com/

QR Code Monkey [Online]. - 3. August 2014 - http://www.qrcode-monkey.de/

Rotacode [Online]. - 3. August 2014 - http://rotacode.com/en/index

Süddeutsche [Online]. - 7. August 2014 - http://www.sueddeutsche.de/digital/apps-fuer-das-smartphone-welches-betriebssystem-bietet-die-meisten-apps-1.1650208

The QR Code Generator [Online]. - 3. August 2014 - https://www.the-qrcode-generator.com/

Wikipedia [Online]. - 11. August 2014 - http://de.wikipedia.org/wiki/App_Store

Wikipedia Kartendienste [Online]. - 3. September 2014 - http://en.wikipedia.org/wiki/Comparison_of_web_map_services

Wirtschaftslexikon [Online]. - 5. August 2014 – http://wirtschaftslexikon.gabler.de/Definition/qr-code.html

Hardware

Für diese Arbeit wurde ein Notebook der G-Series von emachines verwendet.

Betriebssystem:	Windows 7 Professional
Systemtyp:	64 Bit-Betriebssystem
Prozessor:	Pentium(R) Dual-Core CPU T4400 @ 2,20 Ghz
Arbeitsspeicher:	4 GB
Grafikkarte:	NVIDIA

Für diese Arbeit wurde ein Smartphone der Nexus Serie der Google Inc. verwendet.

Bezeichnung:	Nexus 4
Hersteller:	LG Electronics
Kategorie:	Smartphone
Betriebssystem:	Androis OS
Systemversion:	4.4.4 KitKat
Display:	4,7 Zoll (11,94 cm) / 1280 x 768 Pixel
Chipsatz:	Snapdragon S4 Pro
Hauptprozessor:	Quad-Core ARM Cortex A9 / 4 x 1,5 Ghz
Arbeitsspeicher:	2 GB
Sensoren:	Annäherungssensor / Digitaler Kompass / Gyroskop / Lagesensor / Lichtsensor

Anhang

Der Programmcode in dieser Arbeit wird in kursivem Font und in der Farbe grau dargestellt.

Die verschiedenen Begriffe der Arbeitsbereiche und Funktionsbereiche der Entwicklungsumgebung MIT App Inventor werden in dieser Arbeit durch Versalschrift dargestellt.

CD:

- Prototyp
- HTML-Seiten
- schriftliche Ausarbeitung der Thesis