HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Naturmuseum
St.Gallen

Museums-App

Interaktiver Museumsrundgang

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2015

Autor(en):	Daniel Keller
	Samuel Müller
Betreuer:	Prof. Dr. Markus Stolze
Projektpartner:	Naturmuseum St. Gallen

1 Abstract

Museums-App
Interaktiver Museumsrundgang

28.05.2015

Mit der iBeacon-Technologie ist es möglich, Geräte in geschlossenen Räumen zu lokalisieren. Ziel dieser Semesterarbeit war es, ein interaktives Mobile-App für das Naturmuseum St. Gallen zu entwickeln, welches sich diese Technologie zunutze macht. Mit Hilfe der App können die Besucher Zusatzinformationen zu Ausstellungsobjekten in ihrer Nähe aufrufen. Zudem beinhaltet das App einen Quiz-Modus, mit welchem das Museum spielerisch erforscht werden kann.

Das Museums-App wurde als Crossplatform-App für Android und iOS entwickelt. Als Technologie wurde auf das Ionic Framework gesetzt welches auf Adobe Cordova basiert. Die gesamte App wurde mit modernen Webtechnologien entwickelt. Als JavaScript Framework wurde AngularJS verwendet.

Alle Anforderungen konnten erfüllt werden. Das Empfangen der iBeacon Signale ist zuverlässig. Jedoch musste festgestellt werden, dass die Distanzangaben von Gerät zu Gerät stark variieren - die iBeacon-Technologie ist insgesamt nur im Meterbereich genau.

Als Abschluss wurde das App an einem Pilottag im Naturmuseum St Gallen den Besuchern zur Verfügung gestellt. 30 Familien und Einzelpersonen haben das App getestet und mehrheitlich positive Rückmeldungen gegeben.

Inhaltsverzeichnis I

Inhaltsverzeichnis II

Abbildungsverzeichnis

2 Aufgabenstellung

Aufgabenstellung Studienarbeit Abteilung I, FS 2015
Samuel Mueller, Daniel Keller

Mobile App Naturkunde Museum St. Gallen

1. Betreuer & Auftraggeber

Betreuer dieser Arbeit ist
Prof. Dr. Markus Stolze, Institut für Software mstolze@hsr.ch

Auftraggeber dieser Arbeit ist das Naturkunde Museum St. Gallen
Dr. Toni Bürgin
Museumstrasse 32
CH-9000 St.Gallen
toni.buergin@naturmuseumsg.ch
Tel: 071 242 06 86

2. Ausgangslage

Im Naturkunde-Museum St. Gallen bestehet die Möglichkeit die Ausstellungen durch die Einbindung einer speziell entwickelten Mobile App für Besucher noch attraktiver zu machen. Im Rahmen des Challenge Projektes an der HSR im Herbstsemester 2014 wurden von verschiedenen Studententeams prototypische Apps entwickelt, welche verschiedene Nutzungsszenarien demonstrierten. Gemeinsam war diesen Apps, dass sie zur Lokalisierung der Nutzer im Museum iBeacons (kleine Bluetooth LE Sender) verwendeten. Das Projekt wurde in Zusammenarbeit mit dem Naturmuseum St. Gallen und der Firma 2ndWest durchgeführt. Das Museum hat sich dazu bereit erklärt, aufbauend auf den entwickelten Prototypen das Projekt in Richtung eines Pilot-Tests weiterzuführen.

3. Ziele der Arbeit

Ziel dieser Studienarbeit ist es, den Prototypen des Challenge Projektes unter Berücksichtigung anderer Optionen so weiterzuentwickeln, dass der Einsatz in der aktuellen Ausstellung des Naturmuseums St. Gallen in einem Pilotversuch mit echten Museumsbesuchern möglich ist. Um den Erfolg des Projektes zu gewährleisten, wird eine enge Zusammenarbeit mit dem Museum, der Museumspädagogin sowie 2ndWest angestrebt. Das Feedback der Versuchspersonen soll dem Museum wichtige Erkenntnisse bezüglich dem Einsatz der iBeacon-Technologie bieten und im Idealfall als Resultat eine App und Server-Backend liefern welche über den Pilotversuch hinaus im Museum weiter genutzt werden können.

4. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollen den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in einem Exemplare abzugeben (Download für Stolze)

Zudem ist eine kurze Projektresultatdokumentation im öffentlichen Wiki von Prof. M. Stolze zu erstellen.

5. Weitere Regeln und Termine

Im Weiteren gelten die allgemeinen Regeln zu Bachelor und Studienarbeiten „Abläufe und Regelungen Studien- und Bachelorarbeiten im Studiengang Informatik" (HSR Intranet)
https://www.hsr.ch/Ablaeufe-und-Regelungen-Studie.7479.0.html)

Der Terminplan ist hier ersichtlich (HSR Intranet)
https://www.hsr.ch/Termine-Bachelor-und-Studiena.5142.0.html

6. Rechte

Die resultierende Software und Dokumentation darf vom Museum St. Gallen, der HSR und den Studierenden genutzt und weiterentwickelt werden. Die HSR darf die Software zu Schulungszwecken und zur Werbung demonstrieren und nutzen. Die Arbeit (ohne geheime Anhänge) wird veröffentlicht. Titel, Abstract und Auftraggeber der Arbeit dürfen von der HSR und Studierenden schon während der Arbeit kommuniziert werden.

7. Beurteilung

Eine erfolgreiche SA zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Entsprechend sollten ca. 240h Arbeit für die Bachelorarbeit

aufgewendet werden. Dies entspricht ungefähr 17h pro Woche (auf 14 Wochen) und damit ca. 2 Tage Arbeit pro Woche pro Student.

Für die Beurteilung ist der HSR-Betreuer verantwortlich.

Die Bewertung der Arbeit erfolgt entsprechend der verteilten Kriterienliste.

Die Aufgabenstellung wurde am 25.2.2015 vorbesprochen.
Die definitive Aufgabenstellung wurde am 11.3 beschlossen.

Rapperswil, 13.3.2015

Prof. Dr. Markus Stolze
Institut für Software
Hochschule für Technik Rapperswil

3 Management Summary

3.1 Ausgangslage

In Zusammenarbeit mit dem Naturmuseum St. Gallen hat die HSR einen interaktiven Museumsrundgang in Form einer Mobilen App entwickelt. Ziel der Arbeit war es, die von Apple entwickelte iBeacon-Technologie zu erforschen. Das Naturmuseum wird ab 2016 in ein neues Gebäude zügeln und eine komplett neue Ausstellung erarbeiten. Mit dieser Arbeit wollte man evaluieren, ob sich der Einsatz der iBeacon-Technologie in der neuen Ausstellung lohnen würde.

3.2 Vorgehen, Technologien

3.2.1 iBeacons

Die iBeacon-Technologie unterstützt die Navigation in geschlossenen Räumen. Hierzu werden kleine Sender eingesetzt, welche Signale an ein mobiles Gerät senden können. Das Gerät kann mit diesen ermitteln, wie weit der Sender entfernt ist, womit sich ein ungefährer Standort bestimmen lässt. Die Signalübertragung basiert auf der Bluetooth Low Energy (BLE) Technologie und ist somit im Vergleich zu anderen Indoor-Positionsbestimmungsverfahren wie z.B. mittels Wi-Fi [1] besonders batterieschonend und genauer. Eine andere Indoor Alternative wäre das Arbeiten mit Infrarot [2], jedoch wird dies von den wenigsten Smartphones unterstützt.

3.2.2 Idee

Es sollte eine App entwickelt werden, welche sich die iBeacon-Technologie im Museum zu Nutze macht. Zwei Hauptkonzepte standen im Vordergrund:

- Abrufen von zusätzlichen Inhalten zu einem Ausstellungsobjekt in Abhängigkeit zur aktuellen Position
- Ein Quiz-Modus, mit welchem das Museum spielerisch erforscht werden kann

3.2.3 Lösungsansatz und eingesetzte Technologien

Die App musste in ca. 480 Arbeitsstunden umgesetzt werden. Ausserdem sollte die App auf den zwei populärsten Mobile-Betriebssystemen betrieben werden können: iOS und Android. Da es in dieser kurzen Zeit kaum möglich gewesen wäre, die App für beide Plattformen umzusetzen, wurde ein Crossplatform-Ansatz gewählt, welcher es erlaubt, den gleichen Code auf verschiedenen Plattformen auszuführen. Zum Einsatz kam ein Open-Source Framework namens Ionic, welches auf dem Crossplatform-Framework Cordova sowie AngularJS basiert. Durch die Wahl dieser Software war gewährleistet, dass die App sowohl auf iOS als auch auf Android installiert werden kann.

3.3 Ergebnisse

Die App wurde entwickelt und an einem Pilottag auf die Probe gestellt: Sie wurde am 17. Mai, dem Tag der Museen, an einem eigenen Stand im Museum präsentiert. Ca. 30 Personen und/oder Familien probierten die App live im Museum aus. Am Schluss des Rundganges wurde ein Fragebogen ausgeteilt, um Feedback einzuholen. Es gab viele positive Reaktionen, die Leute hatten Spass am Quiz Modus und waren interessiert an den zusätzlichen Informationen, welche die App zur Verfügung stellte. Fehlverhalten war in der App nicht auszumachen.

Die App wurde im App-Store für Android aufgeschaltet. Man verzichtete vorläufig auf einen Release im Apple-Store, da die Zeit bis zur Aufschaltung im Schnitt 7 Tage beträgt und dies nicht in das knappe Zeitbudget dieser Arbeit passte. Es wurden für den Pilottag aber einige Leihgeräte organisiert, auf denen die App bereits vorinstalliert war. Es kamen unterschiedliche Marken und Grössen zum Einsatz. Viele Leute besassen ein iOS-Device und mussten somit auf die Leihgeräte zurückgreifen. Downloads auf die eigenen Geräte der Besucher gab es nur selten.

Mit der iBeacon-Technologie wurden insgesamt positive Erfahrungen gesammelt. Das Einrichten und Konfigurieren verlief problemlos. Die Beacons verhielten sich fehlerfrei. Man muss allerdings beachten, dass die Technologie nur im Meterbereich genau ist, da die auf einem Mobile-Device aufgezeichneten Distanzangaben direkt vom Gerät selber abhängig sind: So wurden z.B. zwischen einem iPhone 6 und einem iPad Mini deutliche Unterschiede festgestellt. Auch Android-Geräte zeigten unterschiedliche Daten.

3.4 Ausblick

Die App wird voraussichtlich während der restlichen Dauer der aktuellen Ausstellung im Naturmuseum zum Einsatz kommen. Ob die App auch in die neue Ausstellung integriert werden soll, ist offen.

4 Technischer Bericht

4.1 Einleitung und Übersicht

4.1.1 iBeacon

Der Markenname iBeacon ist ein, im 2013 von Apple Inc. eingeführter, proprietärer Standard für Navigation in geschlossenen Räumen, basierend auf Bluetooth Low Energy (BLE) [3]. Es handelt sich dabei um kleine elektronische Einheiten, Beacon genannt, welche Signale über die Luft aussenden, welche dann von einem mobilen Gerät empfangen werden können. Die Empfangenen Signale enthalten Angaben über die Signalstärke und die Entfernung. Ebenso ist eine ID des jeweiligen Beacons enthalten.

Die Eigenschaften eines Beacons können konfiguriert werden. So kann man z.B. typischerweise Signalstärke, Signalinterval sowie ID genau definieren (Herstellerabhängig).

Beacons werden typischerweise mit Knopfbatterien betrieben. Dank der eingesetzten BLE-Technologie liegt die Lebensdauer bei 1-3 Jahren, je nachdem, wie der Beacon konfiguriert ist.

Es gibt bereits beinahe ein Duzend Hersteller, welche Beacons produzieren [4]. Für diese Arbeit wurden die Beacons zweier Firmen evaluiert: Estimote [5] und Kontakt [6]. Die HSR hatte von beiden Herstellern bereits eine kleine Menge an Beacons im Besitz. Nach einigen Tests stellte sich heraus, dass beide Arten ähnlich genaue Werte liefern. Allerdings sind die Beacons von Kontakt.io etwas günstiger und ermöglichen einen problemlosen Batterie-Austausch, weshalb wir uns für die Beacons dieser Marke entschieden. Es wurden 25 Beacons im Wert von 570.- CHF eingekauft.

Abbildung 4-1 Kontakt Beacon

4.1.2 Ansatz Crossplatform

Diese Arbeit sollte einerseits einen Mehrwert für das Museum bringen, indem man evaluieren würde, ob das Konzept der iBeacons Museumtauglich ist. Andererseits war es aber aus technischer Sicht interessant, wie sich die Technologie auf verschiedenen Geräten und Plattformen verhalten würde. Eine Anforderung lag also darin, eine App für die beiden populärsten Mobile-Betriebssysteme zu entwickeln: iOS und Android.

Beide Plattformen arbeiten mit völlig unterschiedlichen Programmiersprachen und Eco-Systemen. iOS-Applikationen werden in Objective-C oder Swift geschrieben, Android-Apps hingegen in Java. Dies hätte also dazu geführt, dass man zwei separate Code-Bases hätte entwickeln und warten müssen, was nicht im zeitlichen Rahmen dieser Arbeit lag. Stattdessen entschied man sich, einen Crossplatform-Ansatz zu wählen.

Es gibt verschiedene Frameworks, mit welchen man Crossplatform-Apps entwickeln kann, wobei sich die Konzepte folgendermassen unterscheiden:

Im Xamarin-Framework wird die App in der Sprache C# entwickelt und zu Objective-C oder Java transkompiliert, um den Bedürfnissen der jeweiligen Plattform gerecht zu werden [7]. Ähnlich funktioniert RubyMotion: Der Code wird in der Skript-Sprache Ruby entwickelt und transkompiliert. Grafische Oberflächen müssen aber dennoch mit den nativen Entwicklungstools erstellt werden [8].

Ein anderes Konzept ist jenes vom erst kürzlich veröffentlichten NativeScript: hier wird JavaScript Code geschrieben. Dieser wird aber nicht transkompiliert, sondern in der jeweiligen JavaScript-Runtime der Plattform ausgeführt. Die JS-Runtime wird mit zusätzlichen Libraries bestückt, welche den Aufruf von nativen Funktionsaufrufen ermöglichen. Native Aufrufe werden also von einem in JavaScript geschriebenen Zwischenprogramm gesteuert. UIs werden mit einer universellen, von NativeScript definierten Markup-Sprache deklariert [9].

Ein Konzept das ebenfalls JavaScript zur Hilfe nimmt ist Cordova. Die Idee ist, dass man die App so entwickelt, als wäre sie eine normale Web-Applikation und somit im Browser läuft. Es werden Web-Standards wie JavaScript, HTML und CSS eingesetzt. Diese müssen nicht transkompiliert sondern bloss vom jeweiligen Browser interpretiert werden. Das Problem hier ist allerdings, dass eine Web-Applikation, welche im Browser läuft, keinen Zugriff auf native Funktionalität wie z.B. Speicher oder Vibrationsfunktion hat. Um dies zu umgehen, hat Adobe ein Framework namens Cordova entwickelt, welches den Code in Form von JS, HTML und CSS in einer kleinen App ausführt, welche nur aus einer nativen Web-View besteht. Die App ist also so zu sagen ein minimaler Browser, in welchem nur der Code dieser einen Web-App ausgeführt wird [10].

Da die Zeit für dieses Projekt beschränkt war und wir im Web-Bereich am meisten Erfahrung hatten, entschieden wir uns dafür, die App mit Web-Technologien zu realisieren. In Frage kam allerdings nur das Cordova-Framework, da NativeScript zum damaligen Zeitpunkt noch nicht existierte.

4.1.3 Tools & Sprachen

4.1.3.1 Eingesetzte Tools

Die Entwicklung der App geschah mit Hilfe einfacher, aber effizienter Tools. Es kam keine IDE zum Einsatz, wie dies sonst üblich ist bei der Entwicklung von Mobile-Apps.

Es wurde ein Text-Editor eingesetzt für das Editieren von Code. Der lokale Entwicklungsserver sowie Test- und Buildprozesse wurden aus der Kommandozeile heraus gesteuert. Die App wurde im Browser mit den vorhandenen Developer-Tools gedebugged.

Ionic stellt ein CLI-Tool zur Verfügung, um die Entwicklung von Ionic-Apps zu unterstützen. Damit können z.B. Scaffolds generiert oder die App in einem lokalen Server gehostet werden. Das Tool selbst ist auch in JavaScript geschrieben und wird mit NodeJS ausgeführt, auf den Entwicklungsrechnern musste also ein entsprechendes Environment mit NodeJS und dem zugehörigen Package Manager NPM eingerichtet werden.

4.1.3.2 Eingesetzte Sprachen

Es wurde nicht direkt JavaScript, HTML und CSS verwendet. Stattdessen kamen CoffeeScript, Jade und SCSS zum Einsatz. Dies sind alles spezielle Sprachen, die transkompiliert werden. Ziel dieser Sprachen ist es, die verschiedenen Mängel der Zielsprachen auszubügeln. Im Folgenden sind jeweils zwei Beispiele dieser Mängel aufgelistet, zusammen mit den Lösungsansätzen der jeweiligen Ersatzsprachen.

JavaScript

Der == - Operator in CoffeeScript entspricht dem === - Operator in JavaScript:

CoffeeScript	**JavaScript**
`a == b`	`a === b;`

Das var-Keyword entfällt, Variablen können somit nicht ausversehen global deklariert werden:

CoffeeScript	**JavaScript**

`message = "Hello World"`	`var message;`
	
	`message = "Hello World";`

Jade

Jade ist wie CoffeeScript indentationsbasiert, man kann also auf schliessende Tags verzichten:

Jade	**HTML**

`article`
`  p Text`
	`<article>`
	`  <p>Text</p>`
	`</article>`

Klassen-Namen kann man auf elegante Weise mit Punkt-Notation definieren:

Jade	**HTML**
`div.my-class`	`<div class="my-class"></div>`

SCSS

Die Syntax von SCSS sieht zunächst aus wie normales CSS. Man hat aber zusätzliche Möglichkeiten wie Variablen-Deklarationen oder Kontrollstrukturen.

Variablen-Deklaration:

SCSS	**CSS**

`$width: 100px;`
 	
`div {`
`  width: $width;`
`}`
	`div {`
	`  width: 100px;`
	`}`

For-Loop:

SCSS	**CSS**

`@for $i from 1 through 3 {`
 	
`  .item-#{$i} { width: 2em * $i; }`
 	
`}`
	`.item-1 {`
	`  width: 2em;`
	`}`
	`.item-2 {`
	`  width: 4em;`
	`}`
	`.item-3 {`
	`  width: 6em;`
	`}`

4.1.3.3 Asset Pipeline

Diese Sprachen können aber von heutigen Browsern nicht interpretiert werden. Dies bedeutet also, dass nach einer Änderung am Sourcecode dieser zunächst in die Zielsprache übersetzt werden muss: Man editiert z.B. ein CoffeeScript File, und muss dieses dann in JavaScript umwandeln, sodass es vom Browser interpretiert werden kann.

Da es sehr aufwändig wäre, dies jedes Mal manuell zu tun, wurde ein Tool eingesetzt welches auf File-Änderungen hören kann und dann notwendige Aktionen ausführt. Dieses Tool heisst BroccoliJS. Man kann es konfigurieren in dem man Quell-Verzeichnisse angibt, in denen die zu kompilierenden Files liegen und definiert, welche File-Extensions mit welchem Interpreter verarbeitet werden müssen und wo diese dann hingelegt werden sollen. Man spricht bei diesem Vorgang auch von einer "Asset-Pipeline".

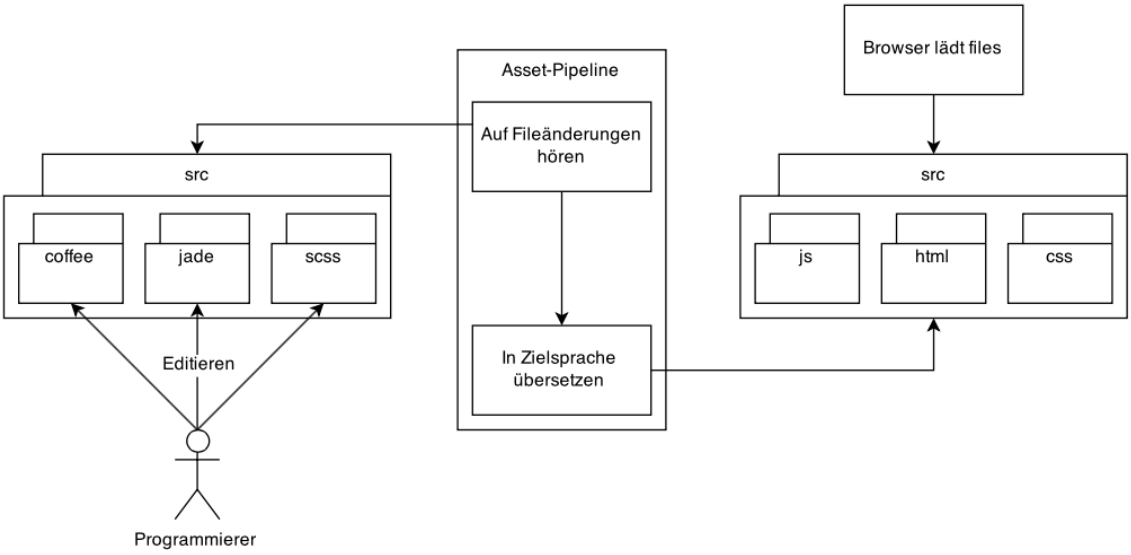


Abbildung 4-2 Illustration Asset Pipeline

4.2 Ergebnis

4.2.1 Devices

Die App wurde auf einer Vielzahl von Devices installiert. Am Pilottag wurden diese dann von den Besuchern ausgeliehen und getestet. Folgende Geräte kamen zum Einsatz:

iOS

- iPad Mini
- iPad 3 Generation
- iPad 4 mit Retina
- iPhone 6

Android

- Samsung Google Nexus S
- Google Nexus 4
- Samsung Galaxy S4 und S6
- Samsung Galaxy Tab (10,5 Zoll Tablett)
- OnePlus One

4.2.1.1 Performance

Die App enthält einige mit CSS umgesetzte grafische Finessen: Es werden Schatten, grosse Hintergrundbilder und CSS-Transitions sowie auch CSS-3D-Transforms eingesetzt. Nicht alle Geräte können diese sauber darstellen. Je neuer der Device ist, desto besser werden die Inhalte dargestellt. Bei Android-Devices konnte die Performance durch den Einsatz eines mitkompilierten Browsers namens Crosswalk leicht verbessert werden.

Man konnte bei älteren bzw. leistungsschwächeren Geräten kleine Mängel in der Darstellung feststellen. z.B. ist die Navigation von einer View auf die nächste typischerweise animiert: die aktuelle View verschwindet nach links und von rechts wird die neue View hereingeschoben. Eine solche Animation wird in Ionic mit einer CSS-Transition gelöst. Da dies nicht nativ, sondern von einem Browser ausgeführt wird, hat man entsprechende Probleme mit der Performance, wenn der Browser nicht dafür optimiert ist, solche Transitions effizient zu rendern.

Die Performance-Probleme beziehen sich aber nicht nur auf das Rendering von CSS. Auch das zugrunde liegende Angular-Framework bringt gewisse Geräte in die Knie: So werden neue Views nicht unmittelbar geladen und man kann beobachten, dass kurz nach dem Laden noch DOM-Manipulationen ausgeführt werden, die man eigentlich nicht sehen dürfte. Dies ist ein bei AngularJS bekanntes Problem: Der Change-Detection-Algorithmus ist nicht besonders effizient und kann bei überladenen Views schnell die CPU-Auslastung in die Höhe treiben.

4.2.1.2 Beacon-Erkennung

Die verschiedenen Devices zeigten Unterschiede bei den empfangenen Beacon-Daten. Die Unterschiede lagen hauptsächlich in der Signal-Stärke und der daraus berechneten Distanz zum Beacon.

Das iPhone6 z.B. zeichnete stets eine sehr hohe Signal-Stärke auf. Die Distanzangaben waren somit recht genau, unterschieden sich aber stark von älteren iOS-Geräten. Die iPads zeigten deutlich schwächere Signal-Stärken und die Distanzangaben hatten höhere Werte.

Bei den Android-Geräten waren die Distanzangaben ebenfalls unterschiedlich. Hier erstaunte uns, dass mit dem PlusOne „one“ ein sehr neues Modell recht schlechte Werte lieferte: die Signalstärke war schwach und die Distanzangaben ungenau.

4.2.1.3 Release in den App-Stores

Die App wurde in den Android-App-Store veröffentlicht.

Auf das Veröffentlichen in den Apple-Store verzichteten wir vorläufig, da das Prozedere dort wesentlich aufwändiger ist: Die App muss z.B. von Apple-Mitarbeitern getestet werden können, was sich schwierig gestaltet, da man zunächst das ganze Beacon-Environment simulieren muss und eine entsprechende Test-Anleitung schreiben muss. Ausserdem dauert das Aufschalten einer neuen Version im Schnitt 7 Tage, was das einliefern von Last-Minute Bugfixes bei unserem knappen Zeitplan erschwert hätte.

4.2.1.4 Lines of Code

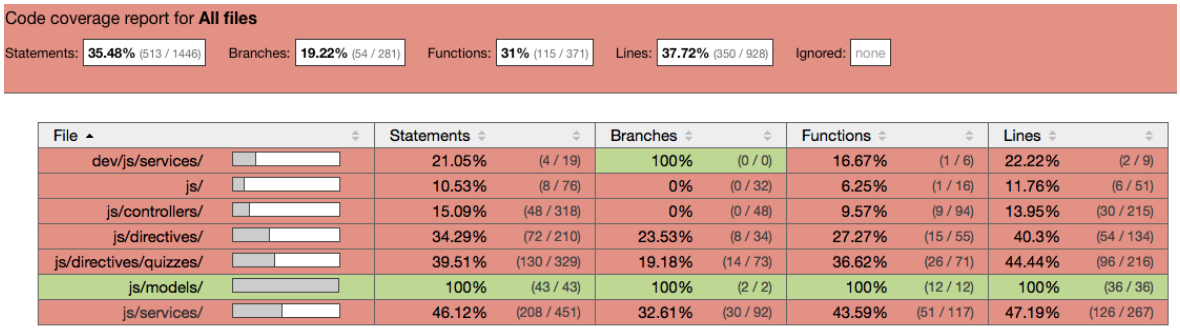
Generiert mit cloc:

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
CoffeeScript                    38            411             16           1648
SCSS                            18            170            138            865
Jade                            25             39              9            278
JSON                             2              0              0              2
-------------------------------------------------------------------------------
SUM:                            83            620            163           2793
-------------------------------------------------------------------------------
```

4.2.1.5 Testabdeckung

Die Testabdeckung liegt bei ca. 30%. Die wichtigsten Klassen sind getestet: Dies sind hauptsächlich die Services, welche das Beacon-Tracking implementieren. Ein grosser Teil des Codes ist UI-Code. Dieser ist schwer zu testen, weshalb wir nicht zuletzt aus zeitlichen Gründen keine Tests dafür geschrieben haben.

Generiert mit istanbul:



File	Statements		Branches		Functions		Lines	
dev/js/services/	21.05%	(4 / 19)	100%	(0 / 0)	16.67%	(1 / 6)	22.22%	(2 / 9)
js/	10.53%	(8 / 76)	0%	(0 / 32)	6.25%	(1 / 16)	11.76%	(6 / 51)
js/controllers/	15.09%	(48 / 318)	0%	(0 / 48)	9.57%	(9 / 94)	13.95%	(30 / 215)
js/directives/	34.29%	(72 / 210)	23.53%	(8 / 34)	27.27%	(15 / 55)	40.3%	(54 / 134)
js/directives/quizzes/	39.51%	(130 / 329)	19.18%	(14 / 73)	36.62%	(26 / 71)	44.44%	(96 / 216)
js/models/	100%	(43 / 43)	100%	(2 / 2)	100%	(12 / 12)	100%	(36 / 36)
js/services/	46.12%	(208 / 451)	32.61%	(30 / 92)	43.59%	(51 / 117)	47.19%	(126 / 267)

Abbildung 4-3 Test Coverage istanbul Export

4.3 Schlussfolgerungen

Als Software-Entwickler hatten wir natürlich stets hohe Anforderungen an das eigens entwickelte Produkt. Die Performance-Probleme waren also insofern störend, weil man wusste, dass diese hauptsächlich auf die eingesetzte Technologie (Web-Technologien) zurückzuführen waren. Hätte man die App mit nativen Tools umgesetzt, hätte man keine Darstellungsprobleme gehabt.

Dies war zwar für uns etwas störend und auch ein kleiner Spassverderber, jedoch stellte sich am Pilottag heraus, dass die Performance-Probleme den Endbenutzern kaum auffielen. Wir erhielten Feedbacks, dass die App sehr professionell wirke und grafisch schön gestaltet sei. Detailliertere Angaben können dem Auswertungsdokument im Anhang entnommen werden.

Somit sind wir insgesamt glücklich mit dem Entscheid, Web-Technologien und das Ionic-Framework eingesetzt zu haben. Zwei Native-Apps hätten wir in dieser kurzen Zeit niemals umsetzen können.

Mit der iBeacon-Technologie sind wir ebenfalls zufrieden. Die ungefähre Standort-Erkennung funktionierte problemlos. Etwas überschätzt hatten wir allerdings die Genauigkeit: Wir konzipierten die App so, dass man die Quiz-Fragen zunächst finden muss, bevor man sie lösen kann. Finden kann man sie, indem man seinen Device so nah als möglich an einen versteckten Beacon hält. Da die Distanzangaben von Device zu Device unterschiedlich sind, funktionierte dies nicht immer ohne Probleme. Wir erhielten einige Feedbacks, dass das Finden von den Quiz-Fragen teilweise mühsam und schwierig war.

Wir empfehlen also, iBeacons für die ungefähre Standorterkennung einzusetzen. Für das Finden von den Quiz würden wir heute eher NFC wählen.

5 Persönliche Berichte

5.1 Daniel Keller

Mit dem Museums-App hatten wir die Chance etwas "Greifbares" zu entwickeln. Es hat grosse Freude bereitet, wie man nach der intensiven Planung- und Entwicklungsphase beobachten konnte wie Bekannte und Fremde das App benutzten. Gross war die Erleichterung als der Pilottag ohne Probleme über die Bühne ging und es mehrheitlich positive Rückmeldungen gegeben hat.

Ich habe die Freiheiten sehr genossen welche uns von Anfang an vom Museum sowie auch von unserer Betreuung durch Herrn Stolze gewährt wurden. So konnten wir unsere eigenen Ideen ausarbeiten, besprechen und danach Umsetzen. Die Zusammenarbeit war stets unkompliziert und zielführend.

Mit Samuel Müller hatte ich einen Projektkollegen auf welchen ich mich verlassen konnte. Da wir aufgrund der verschiedenen Stundenpläne wenige Blöcke hatten, an welchen wir gemeinsam arbeiten konnten, war dies umso wichtiger. So konnten wir flexible Arbeitspakete definierten und diese eigenständig abarbeiteten. Da Samuel mehr Vorwissen bezüglich den eingesetzten Technologien mitbrachte konnte ich von ihm profitieren. Wir haben uns meiner Meinung nach insgesamt gut ergänzt, konnten beide unsere Stärken einbringen und unter anderem deswegen diese erfolgreiche Arbeit so abliefern.

Ebenfalls sehr profitieren konnte ich durch die eingesetzten Technologien. Bei meiner aktuellen Anstellung, welche ich als Teilzeitstudent zwei Tage die Woche verfolge, werden Webtechnologien immer mehr gefragt. So kam es mir sehr entgegen diese Technologien während der Semesterarbeit zu vertiefen. Ich konnte viel dazulernen und dieses Wissen bereits in der Berufswelt anwenden.

5.2 Samuel Müller

Ich freue mich sehr, dass wir die Gelegenheit hatten, eine App zu entwickeln, welche zum Schluss live im Einsatz war.

Für mich war der technische Aspekt am interessantesten. Ich arbeite beruflich viel mit JavaScript und AngularJS und konnte somit viele Synergien erzielen. An dieser Stelle möchte ich mich bei Herrn Stolze bedanken, der sehr offen und unkompliziert war bezüglich der eingesetzten Technologien: Wir konnten die App von Grund auf designen und die eingesetzten Tools selber bestimmen. Dies war für mich sehr wichtig, da ich mich so in Bereichen weiterentwickeln konnte, für die ich mich interessiere.

Auch die Zusammenarbeit mit dem Museum war interessant, unkompliziert und kooperativ. Es war super dass wir einen "echten" Kunden hatten, der an unserer Arbeit interessiert war.

Ich war froh als der Pilottag vorbei war. Es ist anstrengend, auf einen Tag hinzuarbeiten, an dem dann alles funktionieren muss. Die positiven Reaktionen der Leute auf die App waren somit eine grosse Erleichterung.

Das Arbeiten mit Daniel Keller war sehr angenehm. Daniel hat viel zur Projektorganisation beigetragen - eine wichtige Leistung, ohne die der Pilottag wohl nicht so problemlos über die Bühne gegangen wäre. Zudem hat er meinen kleinen Wissensvorsprung über die eingesetzten Technologien schnell aufgeholt und lieferte stets einwandfreie und professionelle Arbeit ab. Ich freue mich darauf, auch weitere Projekte mit ihm zu bewältigen.

6 Glossar

Angular/AngularJS	Ein JavaScript-Framework für die Entwicklung von Rich Client Web-Applikationen (9.4.2.2)
Asset Pipeline	Werkzeug für die automatische Kompilation von Ersatzsprachen (4.1.3.3)
Bug	Ein Fehler in einem Computer-Programm
Crossplatform	Ausführbar auf unterschiedlichen Zielsystemen (4.1.2)
Device	Mobiles Gerät, Handy, Mobiltelefon
HSR	Hochschule für Technik Rapperswil
iBeacon/Beacon	Bluetooth Low Energy Sender. Wird im Kapitel 3.2.1 erklärt
Ionic	Ein JavaScript-Framework für die Entwicklung von Crossplatform-Applikationen. Basiert auf Angular (9.4.2.1)
Kioskmodus	Spezieller Modus in dem die Rechte des Benutzers eingeschränkt sind: So kann der Nutzer z. B. das App nicht beenden.
Skalierung	Die dargestellten Inhalte werden auf die Grösse des Bildschirms angepasst
Testabdeckung	Durch automatisierte Tests ausgeführter Code in Prozent (0)

8 Literaturverzeichnis

[1] D. Schneider, „You Are Here," *Spectrum, IEEE,* pp. 2-6, 28 11 2013.

[2] H. N. H. T. a. M. Y. Hitomi Murakami, „Position Tracking Using Infra-red Signals for Museum Guiding System," *Proceedings of the Second International Conference on Ubiquitous Computing Systems,* pp. 49-61, 2005.

[3] „Wikipedia - IBeacon," [Online]. Available: http://de.wikipedia.org/wiki/IBeacon. [Zugriff am Mai 2015].

[4] „Nodesagency," [Online]. Available: http://www.nodesagency.com/list-9-biggest-beacon-manufacturers/. [Zugriff am Mai 2015].

[5] „Estimote," [Online]. Available: http://estimote.com. [Zugriff am März 2015].

[6] „Kontakt.io," [Online]. Available: http://www.kontakt.io/. [Zugriff am Mai 2015].

[7] „Xamarin," [Online]. Available: http://xamarin.com. [Zugriff am Februar 2015].

[8] „Ruby Motion," [Online]. Available: http://www.rubymotion.com. [Zugriff am Februar 2015].

[9] „NativeScript," [Online]. Available: https://www.nativescript.org. [Zugriff am Mai 2015].

[10] „Cordova," [Online]. Available: https://cordova.apache.org. [Zugriff am September 2014].

[11] „Pivotaltracker," [Online]. Available: http://www.pivotaltracker.com/community/tracker-blog/public-beta-starting-for-redesigned-new-pivotal-tracker. [Zugriff am Mai 2015].

[12] „Pixabay," [Online]. Available: http://pixabay.com/de/portr%C3%A4t-frau-unsch%C3%A4rfe-green-299714/. [Zugriff am Mai 2015].

[13] „Public Domain Images," [Online]. Available: http://www.public-domain-image.com/free-images/people/children-kids/child-young-face-close-up. [Zugriff am Mai 2015].

9 Dokumente des Projekts

9.1 Projektorganisation

Das Entwicklerteam besteht aus den beiden Mitgliedern Daniel Keller und Samuel Müller. Beide übernehmen dieselben Rollen. Der betreuende Dozent ist Prof. Dr. Markus Stolze.

9.1.1 Projektplan

Phase	Meilenstein	Bis	Eingehalten
Evaluation	Brainstorming		✓
	Ideen konkretisiert	01.3.2015	✓
	Wireframes/Templates fertig	08.3.2015	✓
	Planung, Raumplanung, Inhalte, Quiz	18.3.2015	✓
	Kickoff Meeting	19.3.2015	✓
Umsetzung	Start Entwicklung	09.3.2015	✓
	Beacon Empfangstest im Museum	19.3.2015	✓
	Feature Entwicklung abgeschlossen	10.5.2015	✓
	App getestet, Bugs ausgemerzt	10.5.2015	✓
	Testlauf mit Beacons im Museum	10.5.2015	✓
Abschluss	Beacon Installation	11.5.2015	✓
	Pilottag	17.5.2015	✓
	Semester Ende	29.5.2015	✓
	Abgabe Bericht, Kurzfassung, Poster	29.5.2015	✓

9.1.2 Task-Verwaltung

Die Arbeit wurde vor zu und nach Bedarf in kleine Arbeitseinheiten (Tasks) aufgeteilt. Diese wurden mit dem Online-Tool Pivotal Tracker verwaltet. Der Zeitaufwand wurde jeweils geschätzt und mit einem Wert zwischen 1-8 Stunden versehen. Bei mehr als 8 Stunden Aufwand wurde der Task aufgeteilt in Sub-Tasks. Die Tasks wurden nicht fix einem Team-Mitglied zugeteilt. Stattdessen setzten wir auf agile Entwicklung und arbeiteten die Tasks individuell der Wichtigkeit nach ab.

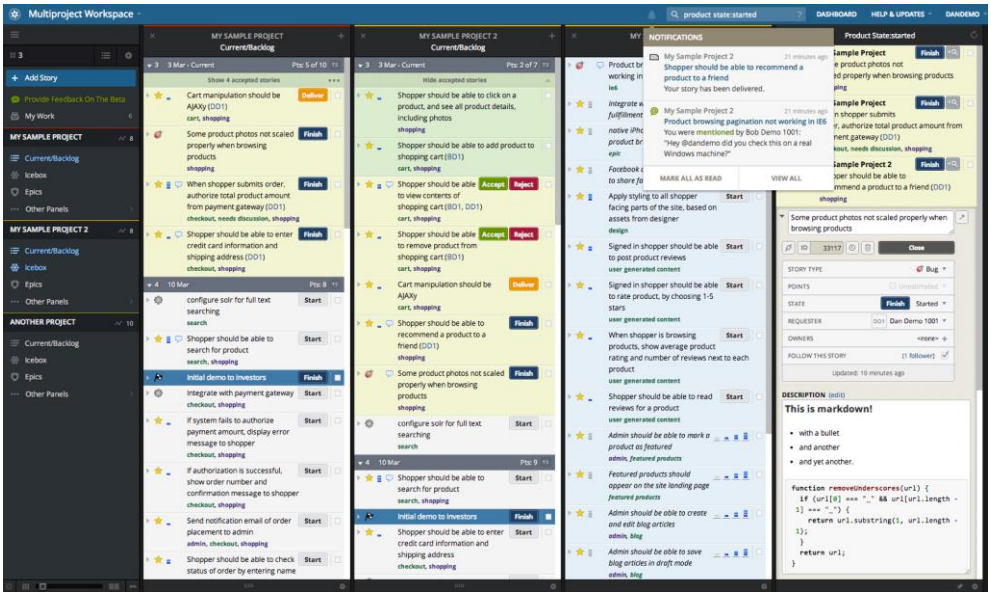


Abbildung 9-1 Screenshot Pivotal Tracker [11]

9.1.3 Source Code Verwaltung

Der Source Code wurde in einem privaten Github Repository verwaltet. Es wurden Pull-Requests verwendet, um Code-Reviews durchzuführen.

9.1.4 Dokumentenverwaltung

Sämtliche Dokumente wurden in einem gemeinsamen Ordner auf Google Drive verwaltet. Somit konnte der Zugriff und das gemeinsame Arbeiten an den Dokumenten sichergestellt werden.

Für die Abgabe wurden die Dokumente in einem Onedrive Word Dokument zusammengeführt da die Formatierungsmöglichkeiten in Google Drive begrenzt sind.

9.2 Zeitkontrolle

Wir hatten keine Teilung der Aufgabenbereiche. Jedes Teammitglied sollte überall mitwirken und es wurde darauf geachtet, dass alle ungefähr gleich viel Zeit in das Projekt investieren. Die Arbeitszeiten wurden in einem Excel erfasst und später ausgewertet.

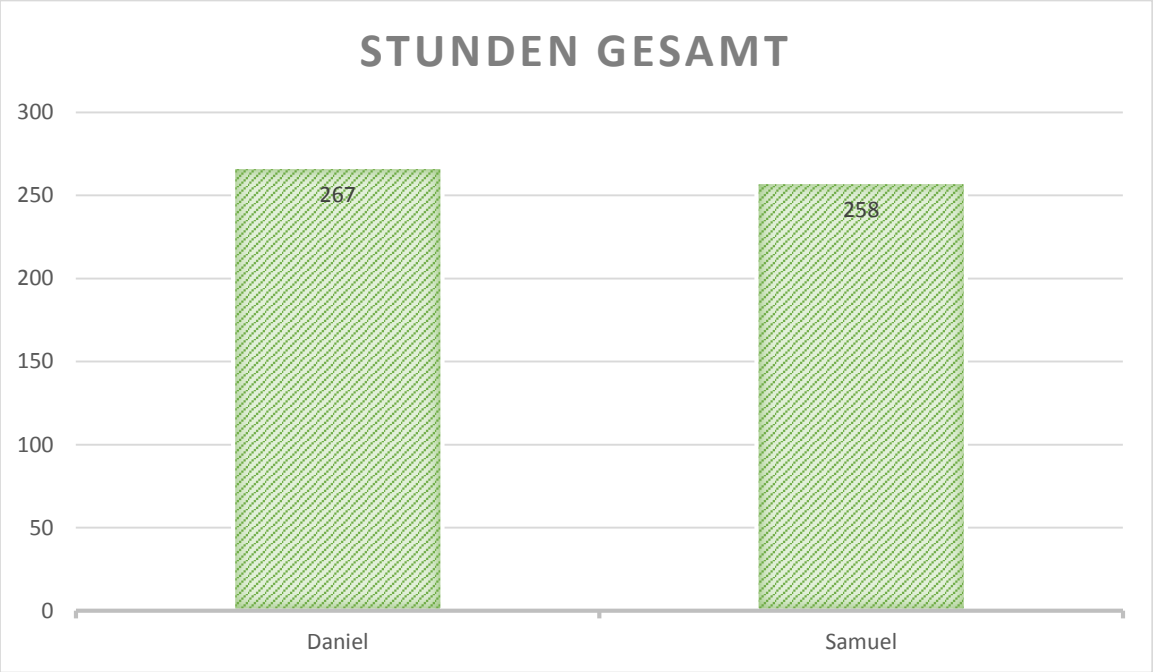
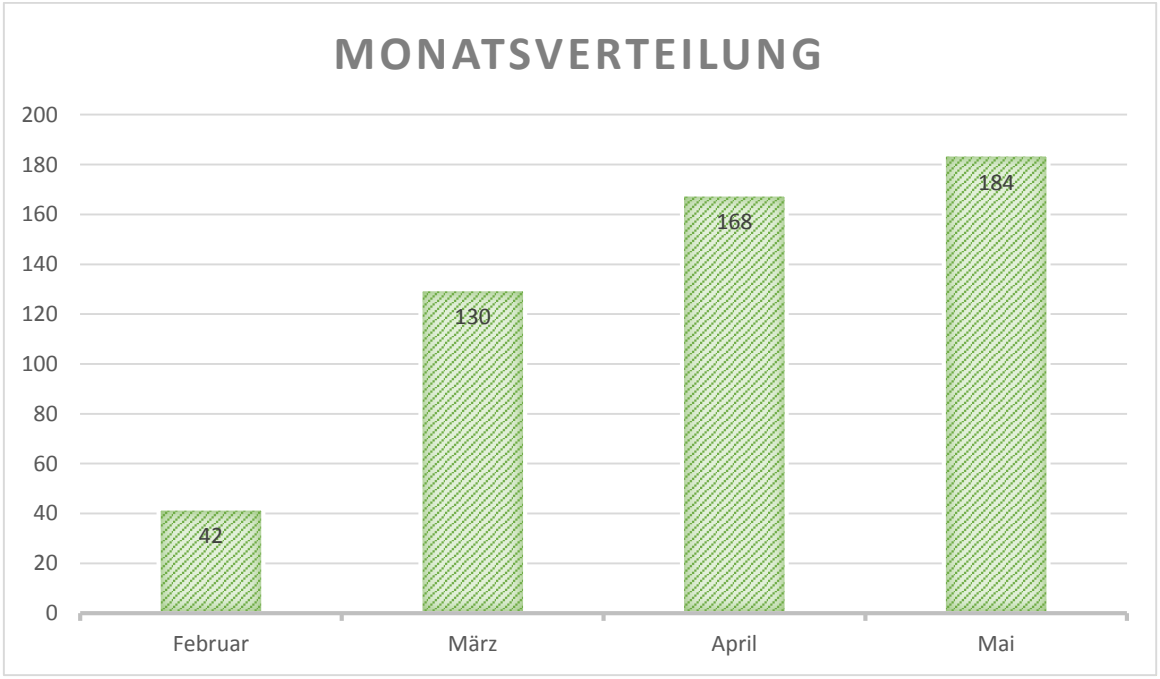


Abbildung 9-2 Zeitkontrolle Stunden gesamt

Am Anfang des Projektes, in der Evaluationsphase, hat es viel Kommunikation und Koordination gebraucht. Da wir da noch nicht entwickelt konnten, haben wir in den folge Monaten etwas mehr gearbeitet. Wir kamen aber gegen Ende des Projektes nicht unter Zeitdruck.



Wie man in der Abbildung 10-2 vernehmen kann, war ungefähr die Hälfte der ca. 500 Stunden Entwicklung. Unter Administration und Diverses fallen sämtliche Projektmanagement bezogene und organisatorische Aufgaben.

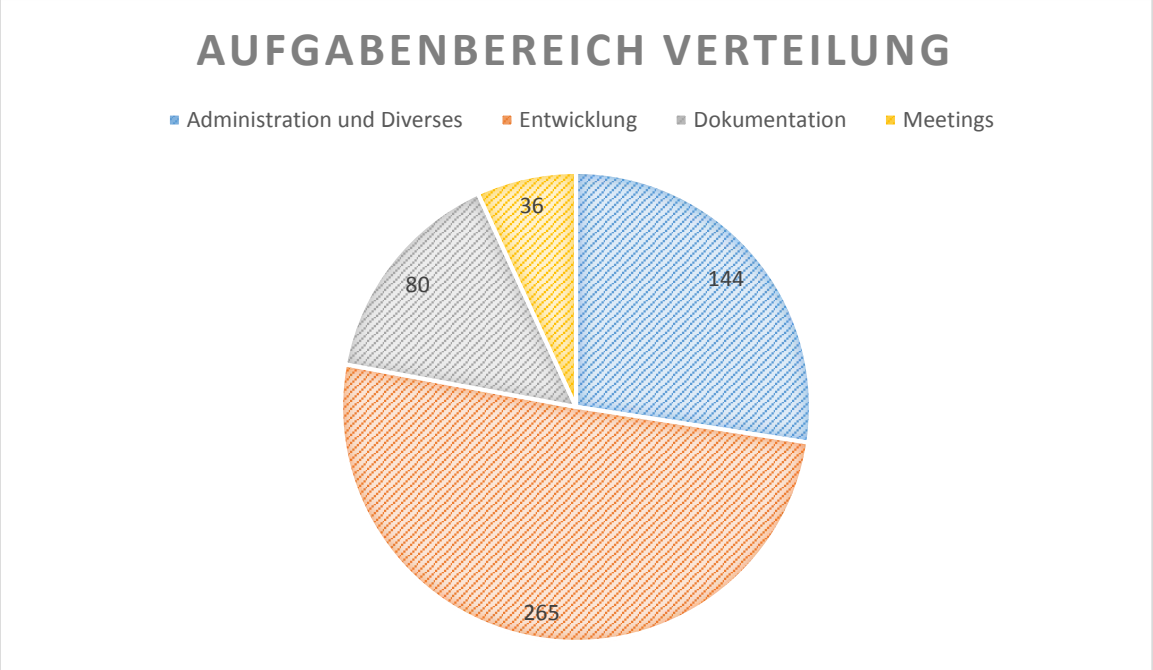


Abbildung 9-3 Zeitkontrolle Aufgabenbereich Verteilung

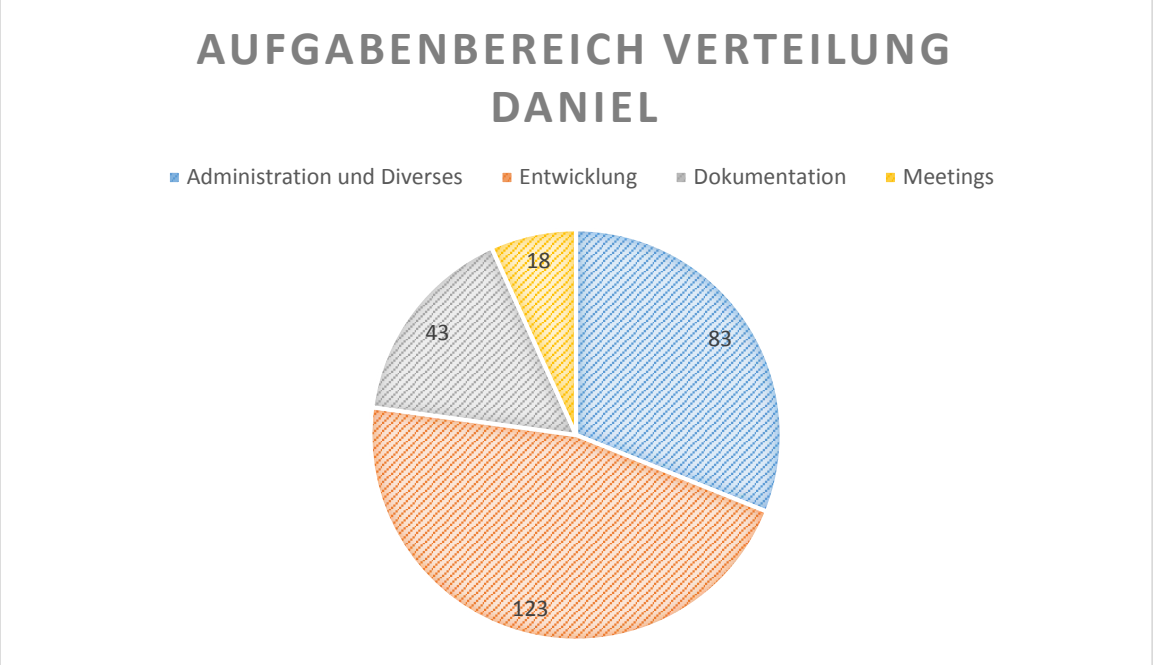


Abbildung 9-4 Zeitkontrolle Aufgabenbereich Verteilung Daniel

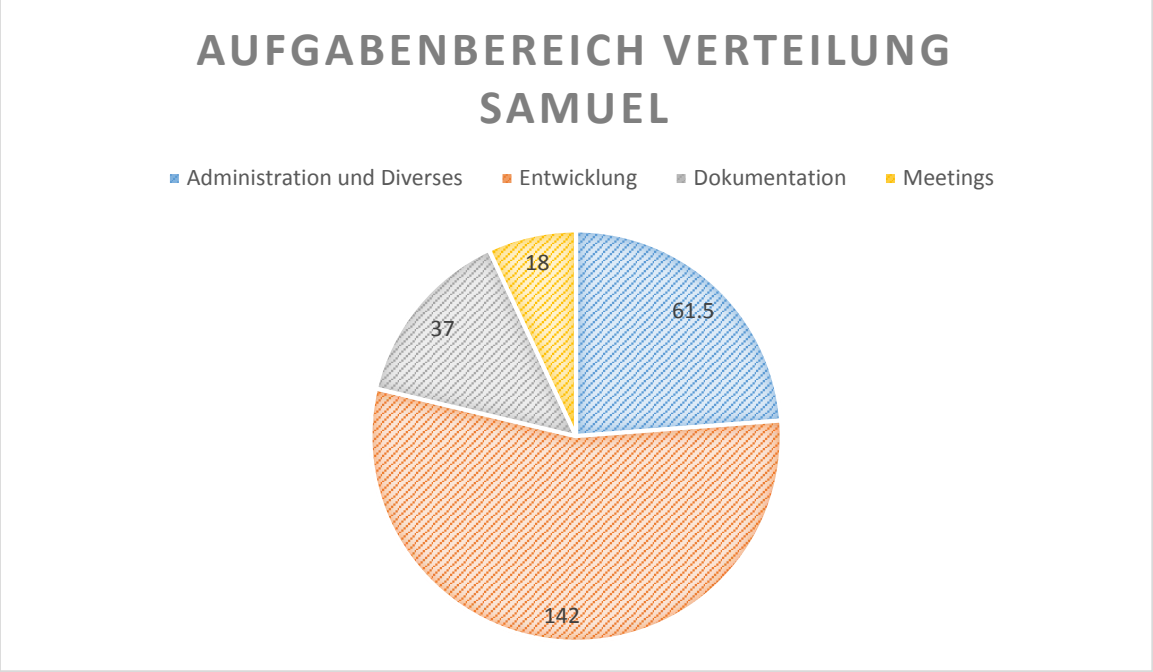


Abbildung 9-5 Zeitkontrolle Aufgabenbereich Verteilung Samuel

9.3 Anforderungsdokument

9.3.1 Einführung

9.3.1.1 Zweck

Dieses Dokument beschreibt die Anforderungsspezifikation und damit die fachliche (nicht technische) Lösung des Museums-Apps.

9.3.1.2 Gültigkeitsbereich

Dieses Dokument ist über die gesamte Projektdauer gültig und wird bei Änderungen der Anforderungen laufend angepasst.

9.3.2 Allgemeine Beschreibung

Dieses Dokument summiert die Anforderungen und korrespondierenden Artefakte für die Semesterarbeit “Museums-App”.

9.3.2.1 Ist / Soll

Das Naturmuseum St. Gallen stellt einen statisch programmierten Rundgang zur Verfügung, welchen man auf einem iPad oder Online durchgehen kann.

Das neue Museums-App soll im Gegensatz zu der bestehenden App vielfältiger und dynamischer sein. Inhalte können von der bestehenden App übernommen werden, werden jedoch optisch neu aufgearbeitet.

9.3.2.2 Benutzer Charakteristik

Es muss davon ausgegangen werden, dass die meisten Benutzer des Apps wenig Smartphone Knowhow besitzen. Sie kennen höchstens die UIs und Dienste von grossen Anbietern wie Facebook, WhatsApp, iPhone etc. Sie sind sich simple und intuitive UIs gewohnt und sind nicht gewillt sich in die Navigation bzw. Bedienung einzuarbeiten.

Ferner ist anzunehmen, dass der durchschnittliche Benutzer eine sehr kleine Frusttoleranz hat. Sollte App nicht performant oder einfach genug sein, wird er sie wohl bald wieder ausschalten.

Es ist also darauf zu achten dem Benutzer eine so einfache und simple Darstellung der Funktionen wie möglich zu bieten. Ebenfalls muss die Performance sehr gut sein, um keine Wartezeiten zu generieren.

Da es sich beim Naturmuseum um eine öffentliche Einrichtung handelt, sollen auch Menschen mit einer Sehschwäche oder Farbblindheit das App bedienen können.

9.3.3 Persona

9.3.3.1 Persona “Maria Meier”

Name	Maria Meier
Alter	43
Funktion	Elternteil
Verpflichtungen	Mutter von 3 Kindern (4, 6, 8).
Smartphone Kenntnisse	Durchschnittliche. Besitzt seit 2 Jahren ein Android Phone. Benötigt es aber hauptsächlich für Telefonate und um ihrem Mann zu schreiben.

Abbildung 9-6 Persona Maria Meier [12]

Frau Meier möchte die Freizeit ihrer Kinder möglichst spannend und sinnvoll gestalten und geht deshalb ab und zu in ein Museum. Sie ist selbst sehr interessiert an den verschiedenen Themen die man sich ansehen kann und liest gerne die Informationen auf den Tafeln durch. Gleichzeitig möchte sie aber auch den Wissensdurst ihrer Kinder fördern und sie für Museumsbesuche begeistern. Dies gestaltet sich manchmal schwierig, da die Kinder hin und wieder die Geduld verlieren und lieber herumrennen oder sich mit anderen Dingen beschäftigen.

9.3.3.2 Persona “Martin Meier”

Name	Martin Meier
Alter	8
Funktion	Elternteil
Kenntnisse	Kann noch nicht fliessend lesen

Abbildung 9-7 Persona Martin Meier [13]

Martin ist in der 2. Klasse und fasziniert von der Natur. Er geht mit seinen Eltern häufig ins Museum. Er hat viel Fantasie und lässt sich von so einem Museumsbesuch leicht inspirieren. Da er selber noch nicht gut lesen kann, lässt er sich die Dinge von seinen Eltern vorlesen und erklären. Wie jedes Kind möchte er die Sachen am liebsten anfassen und bevorzugt interaktive Ausstellungen, wo man Knöpfe drücken kann oder Töne und Videos abgespielt werden. Er mag es nicht, wenn seine Eltern sich stundenlang bei einem Objekt aufhalten und die Informationen durchlesen. Häufig geht er dann auf eigene Faust weiter und schliesst den Museumsrundgang in 15 Minuten ab. Danach will er spielen.

9.3.4 Use Cases

9.3.4.1 Use Case Diagramm

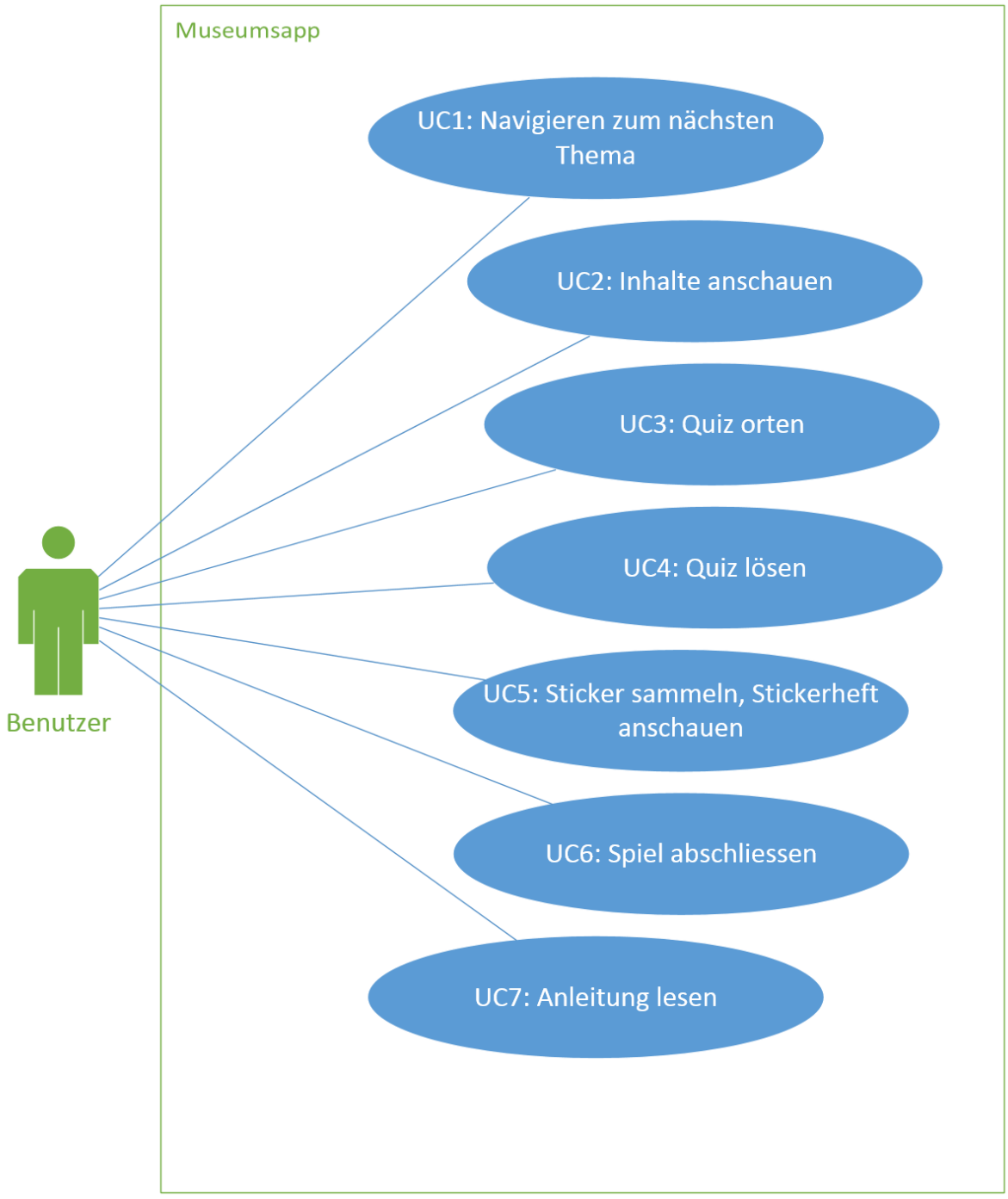


Abbildung 9-8 Usecase Diagramm

9.3.5 Beschreibungen (Brief)

9.3.5.1 UC1: Navigieren zum nächsten Thema

Benutzer können zum jeweiligen nächstgelegenen Thema weiter navigieren. Der Navigationsvorschlag sollte sich ja nach räumlichen Position des Benutzers ändern.

9.3.5.2 UC2: Inhalte anschauen

Benutzer können mit ihrem Smartphone verschiedene Ausstellungsobjekte orten, zu welchen dann automatisch Inhalte präsentiert werden. Ein Inhalt kann ein Text mit Bildern sein oder auch ein Video oder Tonspuren.

9.3.5.3 UC3: Quiz orten

Benutzer können nach versteckten Quiz suchen und diese aufdecken. Um das Quiz aufzudecken, muss der Benutzer anhand eines Hinweises eine Stelle am Ausstellungsobjekt finden, an den er sein Smartphone hinhalten muss.

9.3.5.4 UC4: Quiz lösen

Benutzer können die gefundenen Quiz lösen. Bei einer falschen Antwort erhält er so lange eine neue Chance, bis er die richtige Lösung gefunden hat. Ist das Quiz abgeschlossen, erhält er als Belohnung einen zum Thema passenden virtuellen Sticker, der in ein virtuelles Stickerheft geklebt wird.

9.3.5.5 UC5: Sticker sammeln, Stickerheft anschauen

Hat der Benutzer ein Quiz erfolgreich abgeschlossen, erhält er einen virtuellen Sticker. In der Stickerheft Ansicht erhält der Benutzer eine Übersicht über alle gesammelten Sticker. Noch nicht erhaltene Sticker werden in der Stickerübersicht ausgegraut dargestellt.

9.3.5.6 UC6: Spiel abschliessen

Sobald der Benutzer den letzten Sticker gesammelt hat kommt eine Meldung. Es wird darauf hingewiesen, dass alle Quiz erfolgreich abgeschlossen wurden und an der Kasse eine Belohnung abgeholt werden kann. Wird die Meldung geschlossen, kann der Benutzer wieder zwischen den Themen wechseln und die Inhalte und das Stickerheft anschauen.

9.3.5.7 UC7: Anleitung lesen

Der Benutzer kann jederzeit über das Hauptmenü einen Anleitungstext aufrufen (Ausser wenn er gerade ein Quiz löst. Dann muss die Frage zuerst beantwortet werden).

9.3.5.8 Quiztypen

Wahr/Falsch

Der Benutzer hat eine Reihe von Fragen, welche sich mit Wahr/Falsch bzw. zwei Werten beantworten lassen. Die Frage kann durch ein Bild oder eine Tonspur unterstützt werden. Wurden alle Fragen richtig beantwortet, ist das Quiz abgeschlossen.

Beispiel: Handelt es sich bei dieser Abbildung um einen Schwanz oder einen Froschlurch.

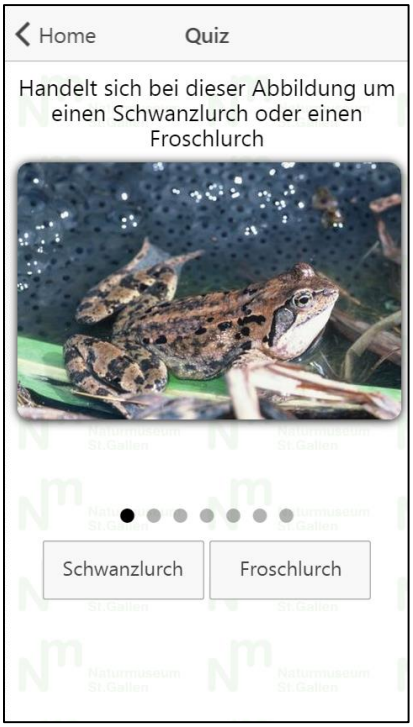


Abbildung 9-9 Screenshot Wahr/Falsch Quiz

Multiplechoice

Der Benutzer hat zu einer Frage mehrere Antwortmöglichkeiten. Die Antworten können in Form von Text oder Bild dargestellt werden. Sobald der Benutzer die korrekte Antwort ausgewählt hat, ist das Quiz abgeschlossen

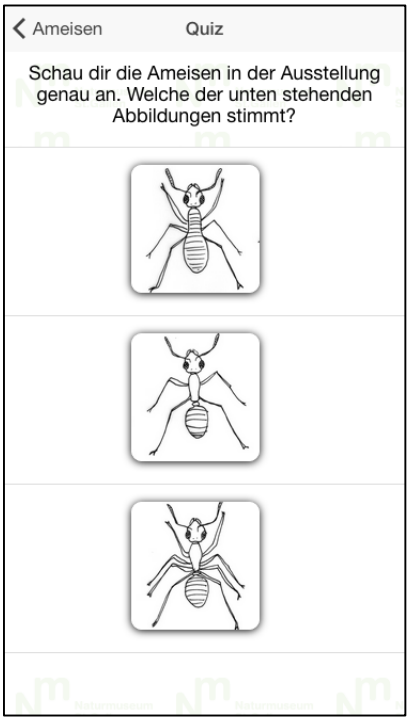


Abbildung 9-10 Screenshot Multiplechoice Quiz

Bilder in korrekte Reihenfolge bringen

Der Benutzer muss eine Reihe von Bildern, welche einen Prozess darstellen, in die richtige Reihenfolge bringen. Ist die Reihenfolge korrekt, ist das Quiz abgeschlossen.

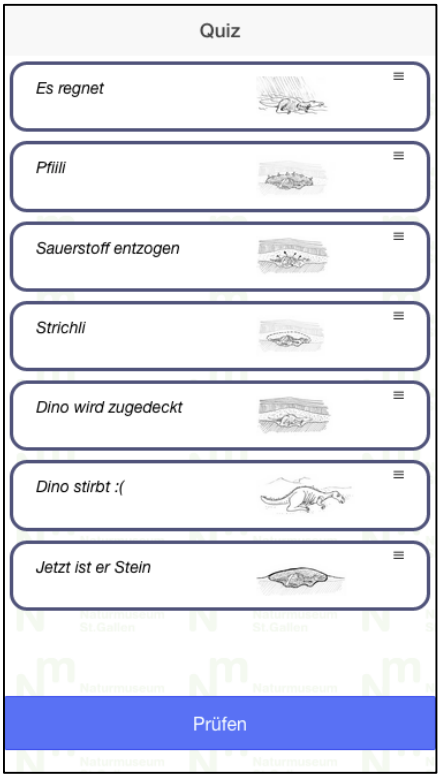


Abbildung 9-11 Screenshot Reihenfolge Quiz

Bildregion bestimmen

Dem Benutzer wird ein Bild dargestellt in welchem er eine Region anklicken muss. Hat er die korrekte Stelle angeklickt ist das Quiz abgeschlossen.

Abbildung 9-12 Screenshot Bildregion Quiz

Bild-Wort Matching

Der Benutzer hat auf der Linken und Rechten Seite Wörter oder Bilder zur Auswahl welche er assoziieren muss. Wurden alle Verbindungen zwischen den Seiten korrekt gesetzt, ist das Quiz abgeschlossen.

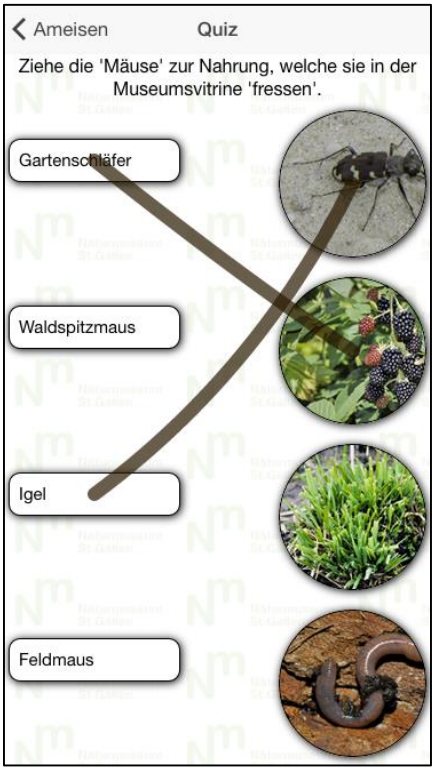


Abbildung 9-13 Screenshot Matching Quiz

9.3.6 Weitere Anforderungen

9.3.6.1 Qualitätsmerkmale (ISO/IEC 9126)

Funktionalität

Richtigkeit	Bei den verschiedenen Quiz-Typen und Inhalten gilt zu beachten, dass diese auch auf einem kleinen Smartphone-Bildschirmen vernünftig dargestellt werden.
Sicherheit	Die Inhalte können nur angeschaut werden, wenn man sich in der Nähe der Beacons befindet.

Zuverlässigkeit

Reife	Die App wird automatisiert und manuell getestet, um mögliche Fehler aufzudecken.

Fehlertoleranz	Die App zeigt keine Fehler an. Stattdessen wird versucht, den Benutzer bei Fehleingaben möglichst intuitiv weiterzuleiten. Testkriterium: Keine Fehlermeldungen.

Benutzbarkeit

Erlernbarkeit	Eine Hilfeseite in der App soll dem Benutzer die Funktionsweise der App erklären können.
Verständlichkeit	Die Software soll leicht Verständlich sein. Um dies zu erreichen sollte es wenige Ansichten geben, wo der Benutzer hin und her Navigieren kann. Insgesamt soll es nicht mehr als 10 unterschiedliche Ansichten geben und zu allen Ansichten sollte man mit maximal vier Klicks navigieren könne. Besonderes Augenmerk erfordert die Navigation von einem Thema zum nächsten: Angenommen, man befindet sich beim Bär, und man geht weiter zum Fisch, soll die App aufgrund der Ortung automatisch erkennen, dass die Fisch-Ausstellung nun am nächsten ist, und zeigt einen entsprechenden Navigationspunkt an, um zu wechseln.
Attraktivität	Sollte das UI auf System-Spezifische Standards optimiert werden, werden diese Plattform abhängig dargestellt.

Effizienz

Zeitverhalten	Die App soll für den Benutzer keine wahrnehmbare Verzögerung haben. Wird auf dem Bildschirm ein Ereignis ausgelöst, sollte eine Antwort innerhalb von 300ms folgen.
Verbrauchs-verhalten	Die App soll verhältnismässig batterieschonend sein. Messen: Wird der Rundgang mit 100% Akku gestartet und dauert 1.5 Stunden mit geöffnetem App sollte der Akku danach bei mindestens 50% liegen (Andere Apps, Datenverkehr und GPS werden ausgeschaltet)

Wartbarkeit

Konformität:	Es werden bewährte, weit verbreitete Tools (Sprachen, Libraries, Frameworks) eingesetzt. Diese bringen meist direkt einen passenden Styleguide mit sich.
Analysierbarkeit	Die Prinzipien der hohen Kohäsion und geringen Koppelung sollen eingehalten werden.
	Statische Code-Analyse (Linting) soll dazu dienen, bestimmte Strukturvorschriften einzuhalten (z.B. Spacing zwischen Operatoren, etc.). Dies wird ermöglicht durch existierende Open-Source-Programme.
Prüfbarkeit	Der Code soll durch automatisierte Unit-Tests gut abgedeckt sein.
Skalierbarkeit	Die Dargestellten Inhalte werden jeweils beim Start der App von einem Json-Web-Service heruntergeladen. Dieser Service wird in der Cloud deployed. Da nur Daten gelesen und keine geschrieben werden, kann der Service leicht skaliert werden.
Lokalisierung	Das Museum App wird nur in der Sprache Deutsch verfügbar sein.

9.3.7 Benutzer Test

Die App wird je 3 iOS und 3 Android Benutzern zum Testing gegeben. Diese sollten sich in Geschlecht, Alter und Tätigkeit unterscheiden. Für das Test-Szenario werden 2 Area und 2 Quiz Beacons aufgestellt. Danach müssen die Tester einen Fragebogen ausfüllen. Der Fokus wird auf folgende Fragen gelegt:

- Verhält sich das App Responsive oder nicht
- Ist die Navigation verständlich
- Allgemeines Feedback

9.3.8 Wireframes

Das App hat vier verschiedene Views welche nachfolgend erklärt werden. Die Navigation wird unter Sitemap erklärt.

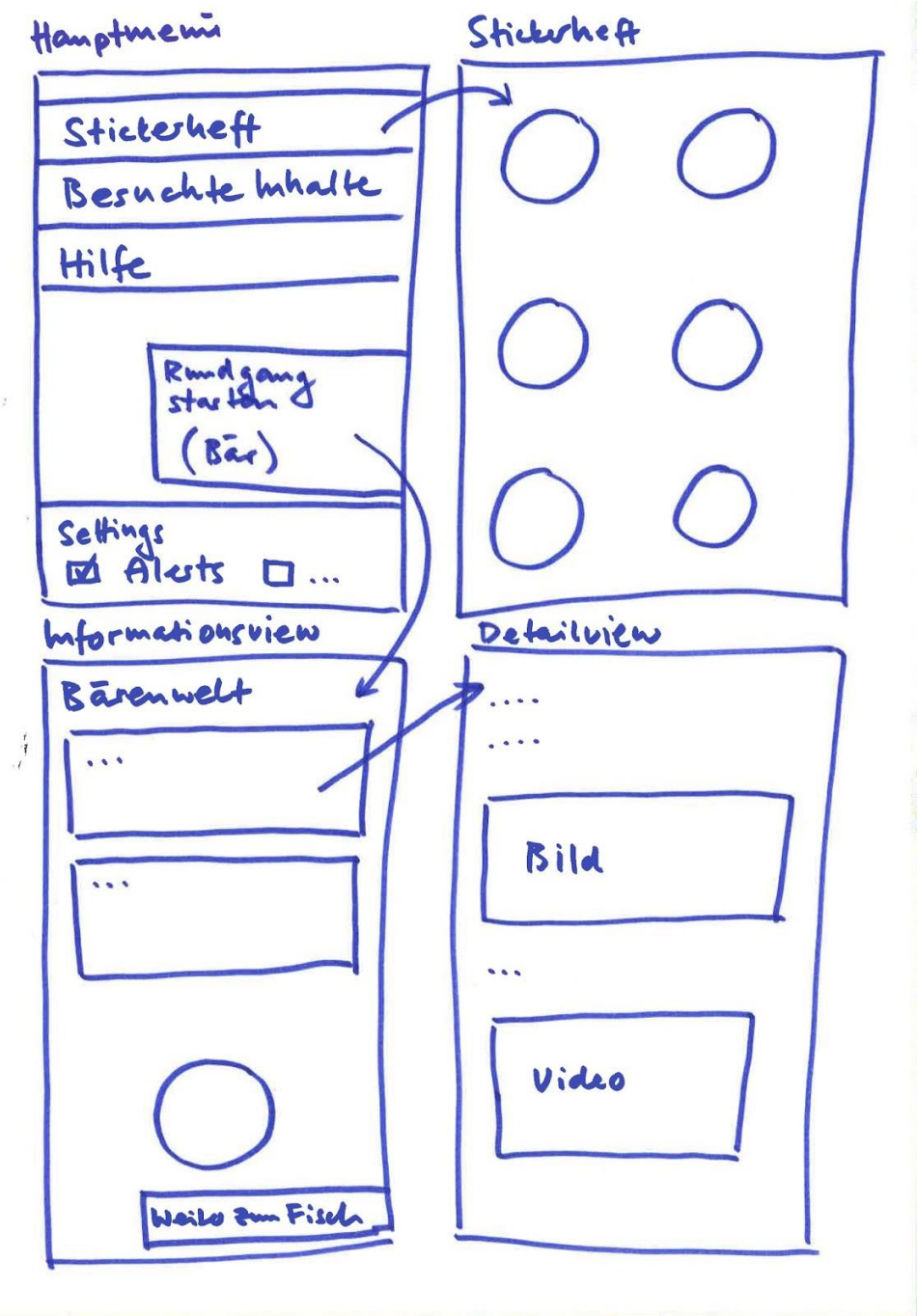


Abbildung 9-14 Wireframe Übersicht

9.3.8.1 Hauptmenu

Das Hauptmenu ist der Einstiegspunkt der App. Man hat die Möglichkeit zum Stickerheft zu navigieren, einen Hilfe Text zu lesen oder den Rundgang zu starten.

Zudem kann man hier allfällige Einstellungen vornehmen, wie zum Beispiel das Ein- und Ausschalten von Benachrichtigung.

Mit “Rundgang Starten” wird die am nächsten gelegene Themenansicht geladen. Wird festgestellt, dass sich der Benutzer nicht im Museum befindet, kommt eine Meldung, dass man sich dorthin begeben soll.

9.3.8.2 Themenansicht

Die Themenansicht bezieht sich immer auf einen Teil der Ausstellung und ist optisch entsprechend gestaltet. Es werden die anklickbaren Links zu weiteren Inhalten (Texte, Bilder, Videos, Ton) aufgelistet sowie der Sticker angezeigt. Klickt man auf den Sticker und befindet sich noch nicht in der Nähe, wird ein Hinweis angezeigt, wo man suchen soll. Hat man den Sticker im Museum gefunden und Klickt ihn an, öffnet sich das Quiz. Unten wird der Navigationslink zur nächsten Themenansicht angezeigt.

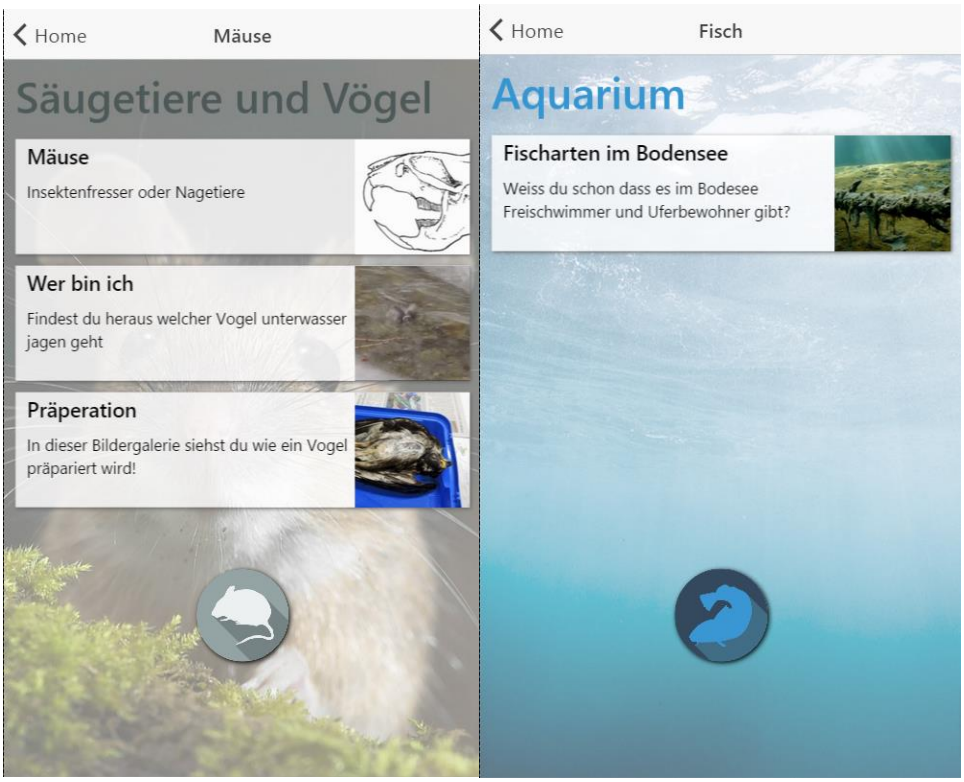


Abbildung 9-15 Screenshot Themenansicht

9.3.8.3 Detailansicht

In der Detailansicht wird der zuvor ausgewählte Inhalt dargestellt. Dies ist typischerweise ein Text mit Bildern oder Videos dazwischen.

Abbildung 9-16 Screenshot Detailansicht

9.3.8.4 Stickerheft

In dieser View sieht man alle gesammelten Sticker. Nicht gesammelte Sticker werden ausgegraut dargestellt.

Abbildung 9-17 Screenshot Stickerheft

9.3.9 Sitemap

Nachfolgend werden die einzelnen Ansichten der App dargestellt. Die Pfeile stellen Navigationsmöglichkeiten dar.

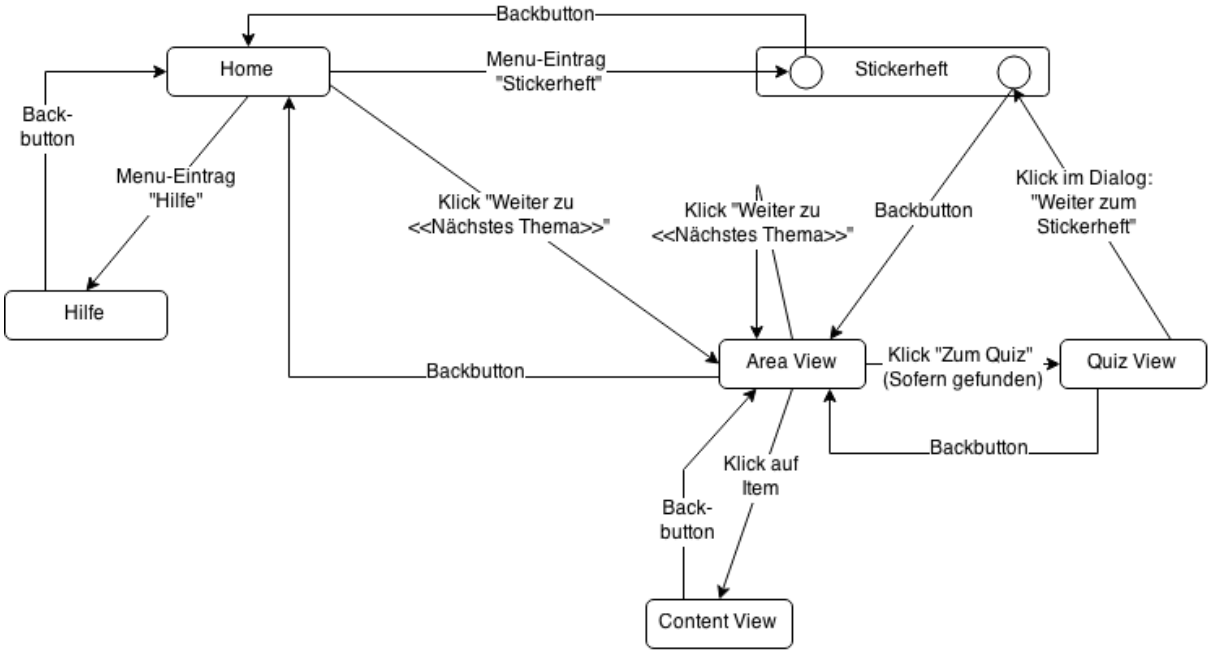


Abbildung 9-18 Sitemap Übersicht

9.4 Architekturdokument

9.4.1 Systemübersicht

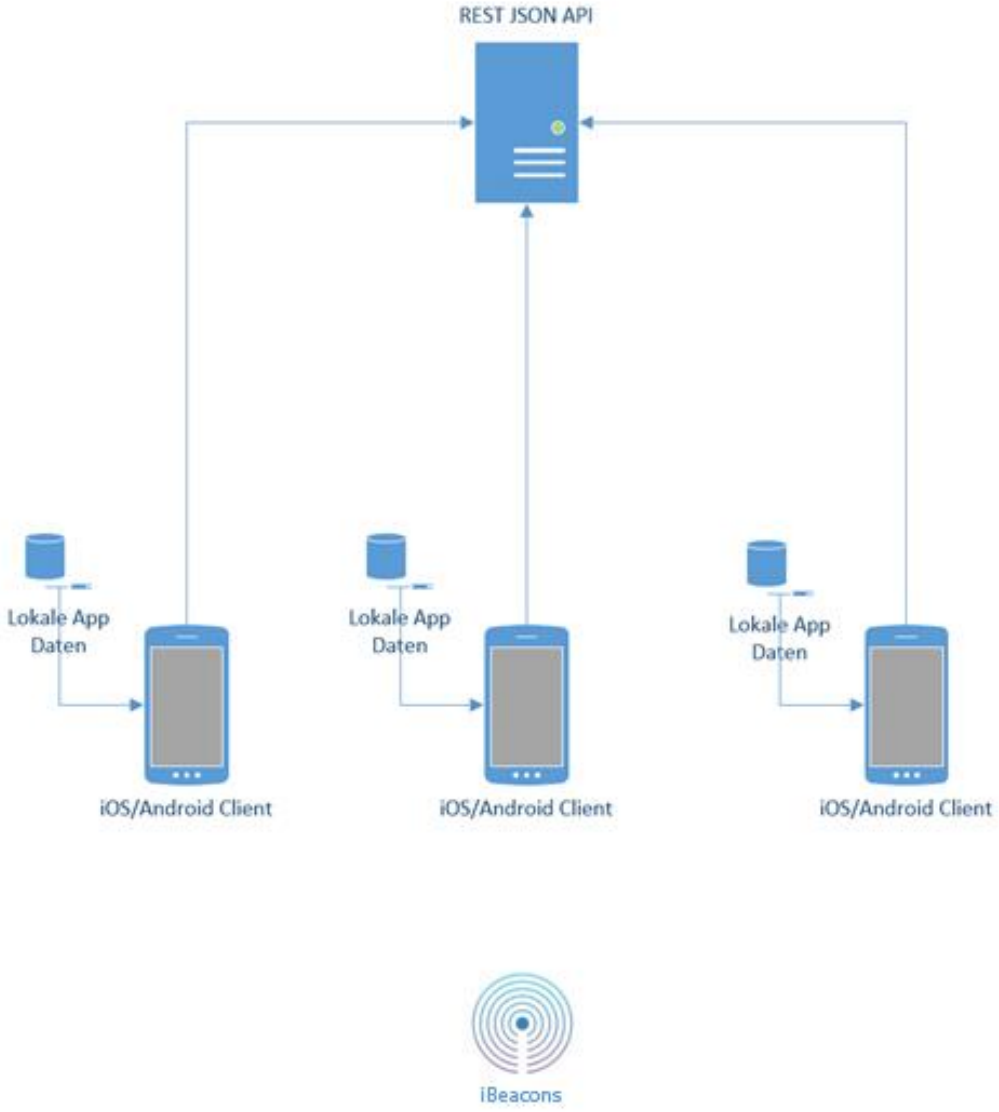


Abbildung 9-19 Systemübersicht Diagramm

Das App wird auf einem iOS oder Android Smartphone installiert. Mit der Installation werden alle Darstellungsrelevanten Daten auf dem Smartphone angelegt. Dazu gehören sämtliche Bilder, JavaScript, CSS und HTML-Seiten. Die Inhalte der App werden über das REST JSON API beim Starten jedes Mal neu bezogen und auf dem Gerät zwischengespeichert. Zu den Inhalten gehören die Texte, Quiz und Beacon-Informationen.

9.4.2 Technologien-Stack

9.4.2.1 Ionic

Ionic verbindet die Konzepte von Cordova und Angular und unterstützt somit die Entwicklung von Crossplatform-Applikationen mittels Web-Technologien. Es stellt nützliche Angular-Services und Directives zur Verfügung, sowie vorkonfigurierte Stylesheets und Widgets.

Alternativen

Name	**Nicht eingesetzt weil...**
Cordova	Ionic baut darauf auf und bietet Erweiterungen
React Native (von Facebook)	Sehr neu, wenig dokumentiert
trigger.io	Nicht kostenlos

9.4.2.2 AngularJS 1

Angular ist das wohl bekannteste SPA-Framework. Es bietet gute Konzepte um eine saubere Architektur und Testbarkeit einer Applikation zu gewährleisten. Es ermöglicht Two-Way-Bindings zwischen View und Model.

Gründe für Einsatz: wird von Ionic vorausgesetzt.

9.4.2.3 CoffeeScript

CoffeeScript ist eine Sprache die zu JavaScript transpiliert wird. Ziel ist es, die Tücken von JavaScript zu bewältigen und gleichzeitig eine knappe und elegante Syntax für maximale Produktivität anzubieten.

Alternativen

Name	**Nicht eingesetzt weil...**
TypeScript	Ist ein Microsoft-Produkt (Opensource) und wurde zu diesem Zeitpunkt vor allem im .NET-Bereich eingesetzt. Visual Studio war daher die einzige Vernünftige IDE mit der man TypeScript entwickeln konnte. Ein Team-Member arbeitet jedoch auf iOS und wollte keinen OS-Switch. TS wird aber heute in Angular 2 eingesetzt und das Ecosystem hat sich massiv entwickelt seit Beginn 2015 (Unterstützung in alternativen Editoren). Eine Migration zu

	TypeScript wäre also heute durchaus denkbar.
Plain JavaScript	Viele Stolpersteine, aufwändige Syntax

9.4.2.4 Jade

Jade ist eine spezielle HTML-Syntax. Ähnlich wie bei CoffeeScript ist die Syntax knapp und elegant gehalten.

Alternativen

Name	**Nicht eingesetzt weil...**
Plain HTML	Aufwändigere Syntax

9.4.2.5 SCSS

SCSS (bzw. SASS) dient dazu, CSS-Code besser zu organisieren. Es können z.B. Variablen verwendet werden sowie Kontrollstrukturen wie For-Loops.

Alternativen

Name	**Nicht eingesetzt weil...**
Plain CSS	Organisation Umfangreicher Stylesheets ist sehr schwierig, es gibt keine Möglichkeiten für Code Re-Use, keine Scripting-Möglichkeiten, etc.
LESS	SCSS wird standardmässig von ionic eingesetzt

9.4.2.6 BroccoliJS

BroccoliJS ist eine Asset-Pipeline, mit welcher die oben genannten Sprachen (CoffeeScript, Jade, SCSS) in die Zielsprachen kompiliert werden können. Die Entwicklung von BroccoliJS wird vom EmberJS-Team vorangetrieben.

Alternativen

Name	**Nicht eingesetzt weil...**
Gulp	Langsamer, nicht flexibel genug
Grunt	Noch langsamer, kein Caching

9.4.2.7 KarmaJS

KarmaJS ist ein Testrunner, der Unit Tests automatisch und manuell ausführen kann. KarmaJS wird vom AngularJS-Team gewartet und verwendet.

Alternativen wurden keine in Betracht gezogen, da KarmaJS von Anfang an hervorragend funktionierte.

9.4.2.8 Ruby on Rails

Ruby on Rails ist ein Full-Stack MVC-Web-Framework. Für dieses Projekt wurde es für die Implementation des Rest-APIs verwendet.

Gründe für Einsatz: Einziges Web-Framework, mit dem das Projekt-Team genügend Erfahrung hatte, um produktiv zu arbeiten. Kann ausserdem sehr leicht und gratis auf Heroku deployed werden.

9.4.2.9 Github

Github ist ein Host für Git-Repositories. Es stellt kollaborative Funktionen wie Issue-Verwaltung oder Pull-Requests zur Verfügung.

Alternativen

Name	Nicht eingesetzt weil...
Bitbucket	Projekt-Team hat keine Erfahrung damit und Github ist absolut ausreichend

9.4.2.10 Travis-CI

Travis ist eine Server-Infrastruktur, auf welcher automatische Builds durchgeführt werden können.

Gründe für Einsatz: Natlose Integration mit Github und KarmaJS. Alternativen wurden nicht in Betracht gezogen, weil Erfahrung mit Travis im Projekt-Team bereits vorhanden war.

9.4.3 Code Guidelines

9.4.3.1 CoffeeScript

Der eingesetzte Styleguide für CoffeeScript-Code basiert auf den heutzutage in vielen Open-Source Projekten anzutreffenden Regeln:

- Einrückung mit zwei Leerschlägen
- Keine Semikolons an Zeilenenden
- Keine Leerschläge an Zeilenenden
- Eine einzige Leerzeile am Ende eines Files
- Ein Space vor und nach dem Pfeil-Symbol
- Ein Space vor und nach arithmetischen Operatoren

Dies ist nur ein Auszug. Die Regeln werden mit dem Tool CoffeeLint forciert. Die Konfiguration ist im File “coffeelint.json” zu finden.

Die Komplexitätsanalyse wurde ebenfalls mit CoffeeLint gemacht. Es gibt momentan 11 Warnungen:

```
www_source/js/app.coffee

 #9-88: The cyclomatic complexity is too damn high. 19.

www_source/js/controllers/AreaCtrl.coffee

 #1-95: The cyclomatic complexity is too damn high. 12.

www_source/js/directives/AreaSlider.coffee

 #1-68: The cyclomatic complexity is too damn high. 11.

 #7-68: The cyclomatic complexity is too damn high. 11.

www_source/js/directives/quizzes/QuizAssign.coffee

 #61-133: The cyclomatic complexity is too damn high. 16.

 #66-133: The cyclomatic complexity is too damn high. 16.

www_source/js/directives/quizzes/TrueFalseQuiz.coffee

 #1-133: The cyclomatic complexity is too damn high. 13.

 #6-84: The cyclomatic complexity is too damn high. 13.

www_source/js/services/BeaconListener.coffee

 #1-72: The cyclomatic complexity is too damn high. 15.

www_source/js/services/BeaconManager.coffee

 #3-67: The cyclomatic complexity is too damn high. 11.

www_source/js/services/DataStore.coffee

 #1-72: The cyclomatic complexity is too damn high. 17.
```

9.4.3.2 Jade

- Einrückung mit zwei Leerschlägen
- Keine Leerschläge an Zeilenenden
- Eine einzige Leerzeile am Ende eines Files
- Klassen werden mit der Dot-Notation angegeben

9.4.3.3 SCSS

- Einrückung mit zwei Leerschlägen
- Keine Leerschläge an Zeilenenden
- Eine einzige Leerzeile am Ende eines Files
- Klassen werden mit “pap-” prefixed (für “paper-chase”)

9.4.4 Folderstruktur

```
+ root
 - resources
 - tests
 + www_source
   - data
   - dev
   - img
   - js
   - lib
   - styles
   - templates
```

9.4.4.1 resources

Enthält Icons und Splash-Screens für die verschiedenen Ziel-Plattformen.

9.4.4.2 tests

Unit-Tests.

9.4.4.3 www_source

Enthält alle Files, die von der Asset-Pipeline verarbeitet und in den temporären www-Folder kompiliert werden.

data

Medien wie audio oder json Files.

dev

Files, die für den Development-Build verwendet werden.

img

Alle Bild-Dateien: Hintergründe, Content-Bilder- etc.

js

Alle CoffeeScript-Files.

lib

Alle 3rd-Party-Libraries, inklusive Ionic.

styles

Alle SCSS-Files.

templates

Alle Jade-Files.

9.4.5 Layers

9.4.5.1 UI

Die UI-Layer-Architektur ist durch den Einsatz von Angular klar vorgegeben. Angular implementiert ein MVVM-Modell. Eine View wird mit HTML deklariert und mit dem Scope eines selbst definierten Controllers verknüpft. Dem Scope werden Models zugewiesen, deren Inhalte mit einem Two-Way-Binding-Mechanismus in die View gebunden werden können.

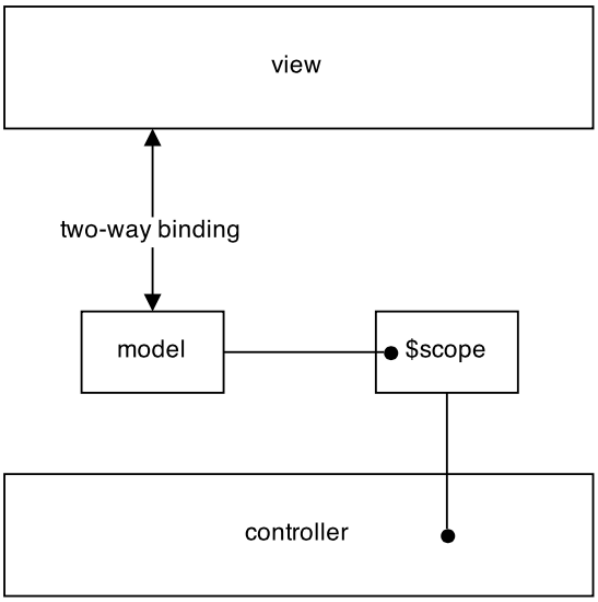


Abbildung 9-20 AngularJS MVVM-Modell

9.4.5.2 Backend-Kommunikation

Die darzustellenden Daten werden von einem Rest-API zur Verfügung gestellt. Ein Service ist dafür zuständig, die jeweiligen HTTP-Requests zu verschicken. Dieser Service wird in die Controllers injected, welche ihn brauchen.

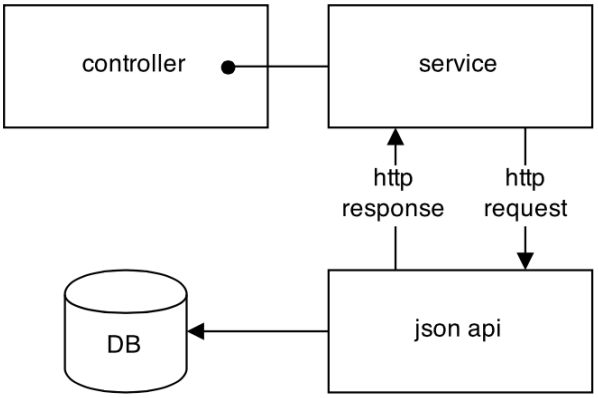


Abbildung 9-21 Backend-Kommunikation Modell

9.4.6 AngularJS Service Übersicht

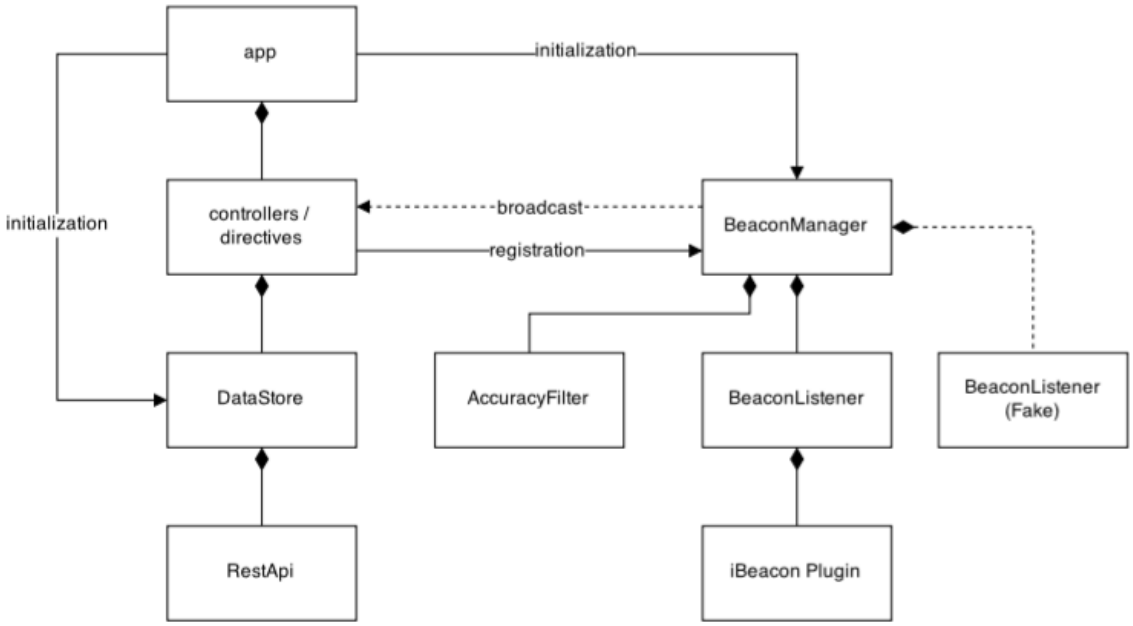


Abbildung 9-22 AngularJS Service Übersicht

9.4.6.1 iBeacon Plugin

Da eine Crossplatform-Technologie eingesetzt wird (Cordova), wird ein Plugin benötigt, welches die nativen APIs von iOS und Android für das Beacon-Tracking ausführen kann:

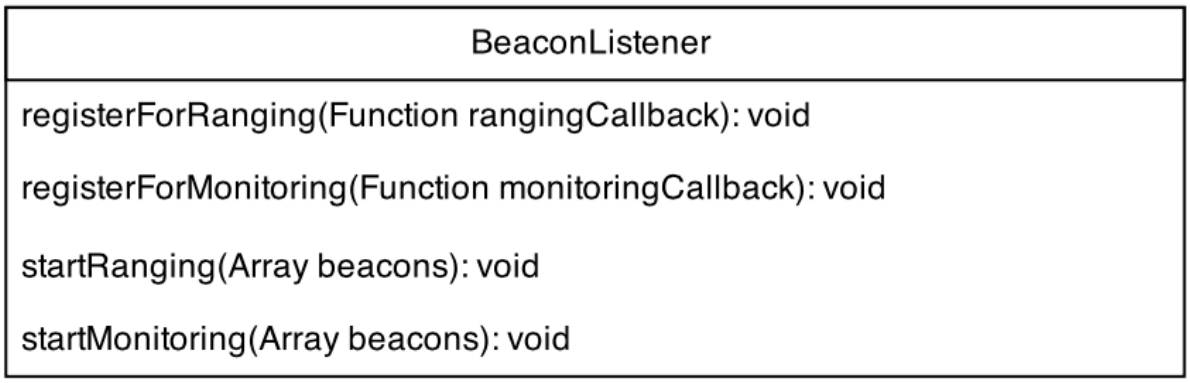
https://github.com/petermetz/cordova-plugin-ibeacon

Dieses stellt eine primitive Schnittstelle der für die Beacon-Technologie spezifischen Ranging- und Monitoring-Mechanismen zur Verfügung.

9.4.6.2 BeaconListener

Der BeaconListener dient dazu, das API des iBeacon-Plugins in knapper Form als Angular-Service zur Verfügung zu stellen. Er kann eine Liste von Instanzen des Typs “Beacon” entgegen nehmen, welche getrackt werden sollen, sowie eine beliebige Anzahl Callbacks, um über die Ranging- und Monitoring-Resultate des iBeacon Plugins benachrichtigt zu werden.

Der BeaconListener dient ausserdem zur Abstraktion einer realen und virtuellen Beacon-Umgebung: Da die App vorwiegend im Browser entwickelt und getestet wird, können häufig keine echten Beacons eingesetzt werden - die Beacon-Daten müssen also simuliert werden. Hierzu existiert eine alternative Implementation des BeaconListeners, welche eine virtuelle Beacon-Umgebung simulieren kann und nicht mit dem iBeacon Plugin arbeitet.



9.4.6.3 BeaconManager

Der BeaconListener wurde unabhängig zur Business-Logik der Museums-App entwickelt. Der Service könnte also in einer beliebigen Angular-Applikation eingesetzt werden. Der BeaconManager hingegen dient dazu, ein für die Anforderungen der App geeignetes API zu exponieren, und verwendet intern den BeaconListener.

Die Museums-App kennt zwei Arten von Beacons: Area-Beacons und Quest-Beacons. Diese unterscheiden sich in ihrer Konfiguration: Da Quest-Beacons sehr genau sein müssen, um das Orten der Quiz zu ermöglichen, senden sie ein stärkeres Signal in einer höheren Frequenz. Area-Beacons hingegen dienen nur zur ungefähren Standortbestimmung und sind daher schwächer und somit batterieschonender eingestellt.

In Anbetracht dieser Anforderung hat der BeaconManager die folgenden Aufgaben:

1. Registrieren von Beacons

Er nimmt eine beliebige Anzahl Beacons zusammen mit einem Key und einem Flag entgegen und registriert die Beacons beim BeaconListener. Wenn er vom BeaconListener Beacon-Daten erhält, filtert er diese nach Key, und verschickt einen Angular-Broadcast wenn der Flag “true” ist. Der Broadcast enthält den Key sowie eine nach Accuracy sortierte Liste von “BeaconData”-Instanzen. Mit dieser Liste kann dann festgestellt werden, welcher Beacon am nächsten ist, in dem man das erste Element herausliest.

Der Broadcast-Mechanismus wird zum Tracken der Area-Beacons verwendet.

2. Registrieren eines einzelnen Beacons

Der BeaconManager stellt eine Methode zur Verfügung, mit welcher ein einzelner Beacon getrackt werden kann. Man übergibt eine ID welche einen Beacon eindeutig identifiziert, zusammen mit einem Callback, welches der Beacon Manager aufruft, sobald er vom BeaconListener Daten zum gewünschten Beacon erhält.

Bei den Quest-Beacons macht es keinen Sinn, jeweils die Daten aller Beacons zu erhalten, da man nur einen Quest aufs Mal suchen kann. Der Registrationsmechanismus eines einzelnen Beacons wird also zum Tracken von Quest-Beacons verwendet.

3. Beacon-Daten glätten

Die Beacon-Daten welche der BeaconListener erhält sind nicht immer genau. Die Distanzangaben können manchmal im Bereich von mehreren Metern schwanken. Dies würde zu einem nervösen wechseln zwischen verschiedenen Areas in der App führen. Um dies zu verhindern, verwendet der Beacon-Manager einen Algorithmus, mit welchem die Daten geglättet werden können. Der Algorithmus ist in einem eigenen Service implementiert (Siehe AccuracyFilter)

BeaconManager
registerSingleBeaconRanging(String id, Function callback): void
deregisterSingleBeaconRanging(): void
resetInterval(): void
registerBeacons(String key, Array beacons, Boolean sendBroadcast): void
start(): void

9.4.6.4 AccuracyFilter

Die Beacon-Daten werden in unregelmässigen Zeitabständen vom BeaconListener empfangen. Die Distanzangaben zu den Beacons können Ausreisser enthalten. Die Aufgabe des AccuracyFilters ist es, mit den Rohdaten des CordovaPlugin gefüllt zu werden. Sein Algorithmus löscht zu alte Messungen anhand eines Timestamps und berechnet aufgrund von den aktuellsten Messungen den akkuratesten Wert. Dafür werden die Messungen sortiert und der Median genommen. Wie viele Messungen beachtet werden sollen und wie alt die älteste Messung sein darf, lässt sich konfigurieren.

AccuracyFilter
configure(Number historyLength, Number milliseconds): void
get(Number beaconId): BeaconData
set(BeaconData data, Date timestamp): void

Die Services BeaconManager, BeaconListener und AccuracyFilter bilden zusammen den komplexesten Teil der App. Hier ein Überblick, wie die Komponenten zusammenspielen:

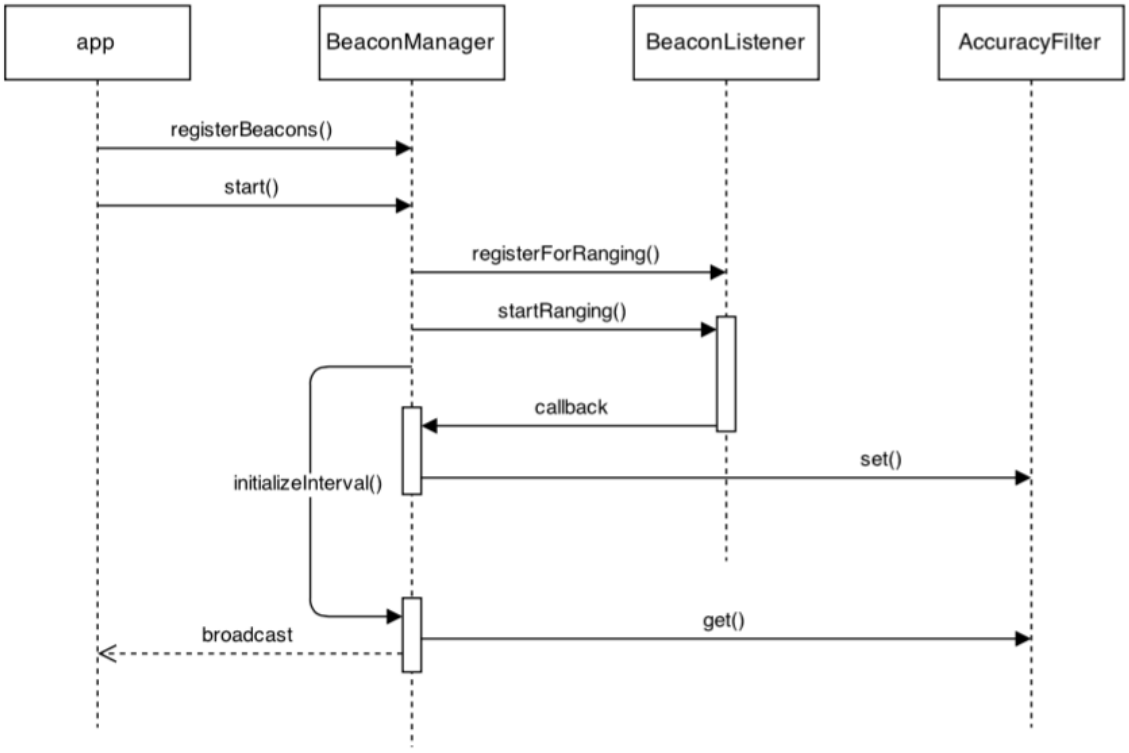


Abbildung 9-23 Sequenz-Diagramm: Zusammenspiel der Services

9.4.6.5 RestApi

RestApi ist ein Angular-Service welcher als Schnittstelle zum Backend dient. Er schickt HTTP-Requests an die Adressen "/areas.json" sowie "/beacons.json" und gibt jeweils ein Promise-Objekt zurück.

RestApi
getBeacons(): Promise getAreas(): Promise

9.4.6.6 DataStore

Der DataStore holt sich Inhalte vom RestApi und speichert diese ab. Dies ist der einzige Service, welcher auf das API zugreift. Der DataStore wird beim Starten der App initialisiert.

DataStore
awaitLoadCompletion(): Promise initialize(): void getBeacons(): Array getAreas(): Array getAreaKeys(): Array getBeaconById(id): Beacon getBeaconByMinor(minor): Beacon getAreaByBeacon(beacon): Area getAreaByQuestBeacon(beacon): Area getAreaByBeaconData(beaconData): Area getAreaByKey(key): Area getContent(areaKey, contentId): Content

9.4.6.7 AppData

Exponiert ein API für das Speichern von App-Spezifischen Daten. Dies beinhaltet hauptsächlich Methoden zum Speichern von gelösten Quiz sowie das Laden ebendieser.

AppData
saveQuestCompletion(areaKey): void isQuestCompleted(areaKey): Boolean getCompletedQuests(): Array reset(): void allQuetsCompleted(): Promise

9.4.6.8 LocalStorage

Abstrahiert Zugriffe auf das localStorage Objekt, welches bei Cordova-Projekten als persistenter Speicher genutzt werden kann.

LocalStorage
set(String key, Object value): void get(String key, Object defaultValue): Object setObject(String key, Object object): void getObject(String key): Object

9.4.7 Beacon Simulator

Um während der Entwicklung den Museumsrundgang zu simulieren, wurde ein kleines Tool namens Beacon Simulator entwickelt. Dieses Tool ermöglicht es, virtuell durch das Museum zu gehen: Man hat ein interaktives GUI wo man ein kleines blaues Feld umher schieben kann, welches den Benutzer im Museum darstellt. Beacons sind entsprechend markiert.

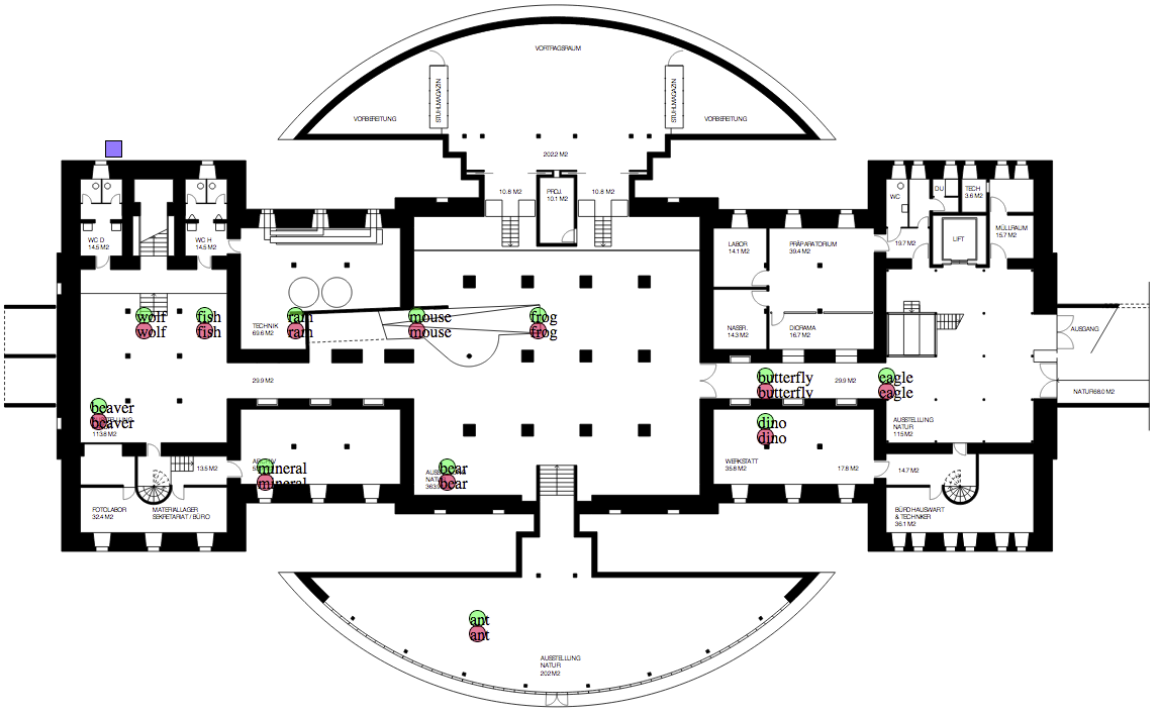


Abbildung 9-24Beacon Simulator

9.4.8 iBeacon: Begriffe

9.4.8.1 Beacon

Ein Beacon hat 3 verschiedene IDs

- UUID: Ist 32-stellig und Herstellerabhängig. Kann nicht selber vergeben werden.
- Major: Ein Integer welchen man selber setzen kann. Er ist zur Bildung von Beacon-Gruppen gedacht.
- Minor: Ebenfalls ein Integer welchen man selber setzen kann um die Beacons in der Gruppe zu identifizieren.

9.4.8.2 Ranging

Man kann eine Gruppe von Beacons rangen. Dies kann man auf Basis der UUID, der Major-ID oder Minor-IDs machen. Beim Ranging bekommt man pro Beacon die Distanzangabe.

9.4.8.3 Monitoring

Beim Monitoring wird festgestellt, wenn man den Radius eines Beacons verlässt oder Betritt. Das Monitoring wir in der Museums-App nicht verwendet.

9.4.9 Code-Review

Der Code wurde von Michael Gfeller reviewed. Wir erhielten folgende Inputs:

Input	Folgeschluss
Architektur ist gut (man kann bei einer Angular-Applikation sowieso nicht viel falsch machen)	-
Entwickler-Anleitung sollte erweitert werden	Wird gemacht, sobald entschieden ist, ob das Projekt weitergeführt wird
Es ist kein Error-Handling vorhanden	Wird in einer allfälligen Weiterentwicklung berücksichtigt
Service Dictionary enthält hart codierte Labels	Wurde gefixt
Code enthält auskommentierte Abschnitte	Wurden entfernt
Code enthält TODOs	Werden resolved bei einer allfälligen Weiterentwicklung
Test-Abdeckung ist OK	-
Backend-Deployment beschreiben	Wird bei einer allfälligen Weiterentwicklung berücksichtigt

9.4.10 Installationsguide

Es muss zunächst Node und npm installiert sein. Danach müssen folgende Befehle in der Kommandozeile mit Administratorrechten ausgeführt werden:

```
> git clone https://github.com/samu/paper-chase.git
> cd paper-chase
> npm install
> npm install -g cordova ionic grunt-cli broccoli-cli karma-cli bower
> bower install
```

Dies installiert alle nötigen npm-packages.

Nun kann die App in ein Zielverzeichnis kompiliert werden:

```
> broccoli build outputFolder
```

Wenn man dies nicht ständig wiederholen möchte kann man auch diesen Befehl ausführen (File-Watch):

```
> grunt broccoli:test:watch
```

Zum Schluss muss der Ionic-Server gestartet werden, damit man die App im Browser testen kann:

```
> ionic serve
```

Die App ist dann unter folgendem Link verfügbar:

http://localhost:8100/#/app/home

9.5 Accessability

9.5.1 Accessability bei Farbblindheit

Die Area-Views sind farbenfroh gestaltet: Jede der zwölf Views hat ein zum Thema passendes Hintergrundbild, die Titel sind passend eingefärbt, und jeder Sticker hat ein spezielles Design. Als Test haben wir einige Views mit einem Tool namens Vischeck geprüft (rechts das Original, links die Farbblindheits-Simulation):

Abbildung 9-25 Screenshots Accessability Farbblindheit 1

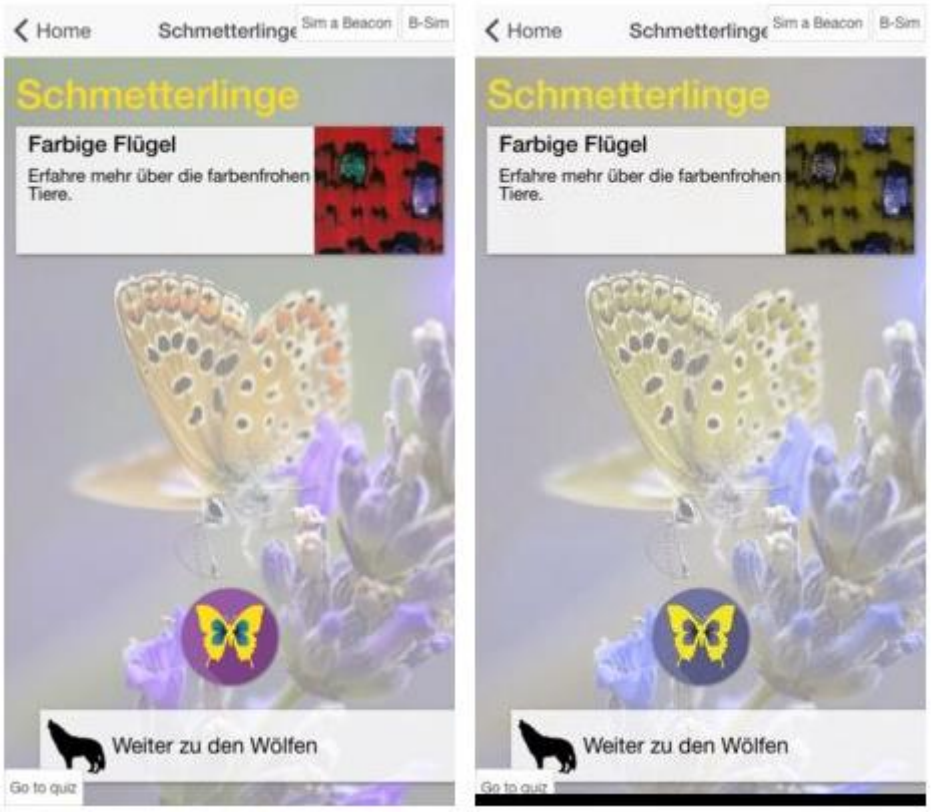


Abbildung 9-26 Screenshots Accessability Farbblindheit 2

Abbildung 9-27 Screenshots Accessability Farbblindheit 3

Abbildung 9-28 Screenshots Accessability Farbblindheit 4

9.5.2 Accessability bei Sehschwäche

Die App skaliert automatisch je nach Screen-Grösse. Auf einem grossen Device wird die Schrift grösser dargestellt. Im Folgenden ein Vergleich zwischen einem iPhone 6 und einem iPad 4 (Man beachte den "Zurück"-Button als Grössenvergleich):

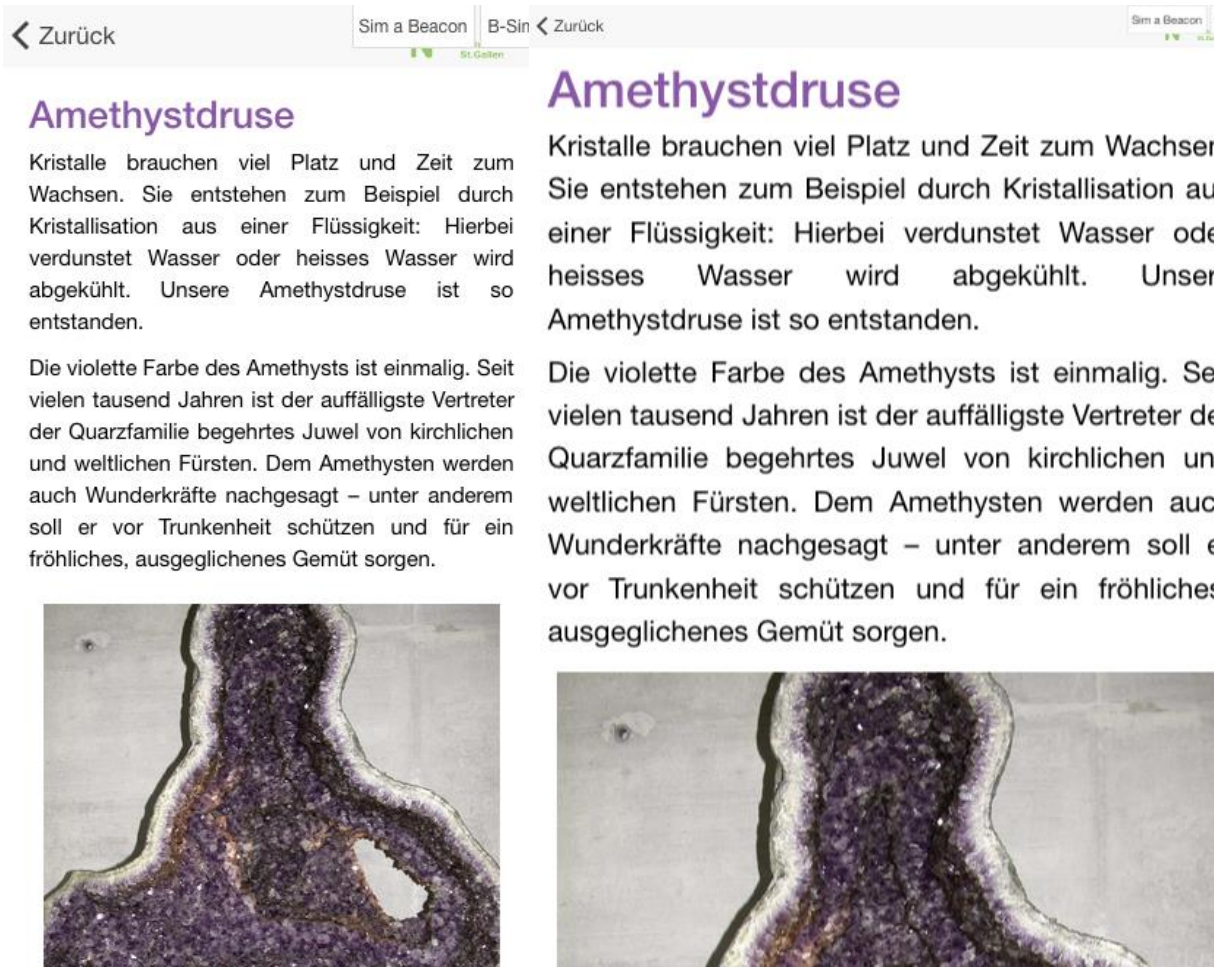


Abbildung 9-29 Screenshots Accessability Sehschwäche

9.5.3 Fazit

Die Tests vermitteln den Eindruck, dass genügend Kontraste und Linien vorhanden sind, um auch Farbblinden die Navigation in der App zu ermöglichen. Die Skalierung funktioniert gut, der Inhalt wird auf grossen Devices grösser dargestellt. Am Pilottag haben wir keine negativen Kritiken bezüglich der Skalierung erhalten.

9.6 Benutzertest

Die App wurde iOS- und Android-Benutzern zum Testing gegeben. Diese unterschieden sich in Geschlecht, Alter und Tätigkeit. Für das Test-Szenario wurden 2 Area und 2 Quiz Beacons aufgestellt. Danach wurden die Tester befragt. Der Fokus wurde auf folgende Fragen gelegt:

- Verhält sich das App Responsive oder nicht
- Ist die Navigation verständlich
- Allgemeines Feedback

Die Tests wurden anfangs Mai durchgeführt als die App zu ca. 80% fertig war.

9.6.1 Testpersonen

9.6.1.1 Testpersonen Android

Testperson 1

Name	Jonas H.
Alter	28
Beruf	Softwareentwickler
Android Kenntnisse	Sehr gut, langjähriger Android Benutzer. Besitzt ein Android Smartphone seit mehr als 4 Jahren,

Testperson 2

Name	Barbara K.
Alter	53
Beruf	Sachbearbeiterin
Android Kenntnisse	Nicht sehr gut, mangelndes technisches Interesse. Besitzt Android seit 2 Jahren.

Testperson 3

Name	Sandra K.
Alter	25
Beruf	Primarschullehrerin
Android Kenntnisse	Durchschnittliche Android Kenntnisse. Besitzt Android seit 3 Jahren.

9.6.1.2 Testpersonen iOS

Testperson 1

Name	Frank F.
Alter	41
Beruf	Softwareentwickler
iOS Kenntnisse	Sehr gut, langjähriger iOS Benutzer

Testperson 2

Name	Martin S.
Alter	23
Beruf	Informatikstudent
iOS Kenntnisse	iOS Kenntnisse: Sehr gut, langjähriger iOS Benutzer

9.6.2 Feedback

9.6.2.1 Responsive

- Die Testpersonen haben bemängelt, dass vor allem der Seitenwechsel ein wenig ruckelt
- Laut Testpersonen lässt sich das App im allgemeinen aber gut bedienen

9.6.2.2 Navigation

- Quiz Button wurde nicht als anklickbar wahrgenommen
- Die Testbenutzer haben intuitiv auf die Inhalts-Verlinkungen geklickt für Zusatzinformationen
- Die Rücknavigation mit dem Zurück-Button wurde von allen Android Testpersonen genutzt. Zum Teil wurde auch der In-App Navigationslink verwendet
- Bei den Zusatzinformationen wurde auf das Bild geklickt und eine Reaktion erwartet
- Im Stickerheft wurde auf die Stickers geklickt und ein Feedback erwartet
- iOS: Dass man mit einem Swipe von Links nach rechts nicht zur vorherigen View kann, wurde bemängelt.

9.6.2.3 Allgemeines Feedback

- Die Testpersonen haben ein positives Feedback gegeben. Jedoch war es für die meisten schwer es sich in echt vorzustellen da die Beacons nicht im Museum aufgestellt wurden.

9.7 Resultierende Massnahmen

- Für den Seitenwechsel wird ein Overlay Layer eingeblendet, damit der Seitenwechsel flüssiger wirkt
- Es wird einen Hilfetext geben welcher die Navigationsmöglichkeiten erklärt
- Der Button wurde mit einer Animation auffälliger gestaltet
- Wenn man in der Inhaltsansicht auf ein Bild klickt, wird es vergrössert angezeigt
- Im Stickerheft kann man auf die Stickers klicken und erhält ein Feedback
- Die Swipe-For-Back Navigation kann in iOS aus technischen Gründen nicht angeboten werde

9.9 Risikomanagement

9.9.1 Risiken

R1 Technologie ungenügend

Die ausgewählten Technologien und Frameworks können die Anforderungen nicht erfüllen.

Vorbeugung: Schreiben eines Prototypen (Bereits im Challenge-Projekt geschehen)

Verhalten beim Eintreten: Workarounds anwenden

R2 Arbeitsumgebung defekt

Arbeitsumgebung wird unbrauchbar, z.B. kaputter Laptop oder fehlerhafte Software.

Vorbeugung: Diverse Workstations vorbereiten

Verhalten beim Eintreten: Weiterarbeiten auf anderen Workstations oder Neuinstallation

R3 Performance Probleme

Das App läuft nicht auf allen Geräten flüssig oder mit zu hohem Akkuverbrauch.

Vorbeugung: Altes Android Phone organisieren und von Anfang an darauf testen (Samsung Google Nexus)

Verhalten beim Eintreten: Die Software optimieren

R4 Veröffentlichung im Appstore macht Probleme

Veröffentlichung der App auf den Appstores der verschiedenen Plattformen macht Probleme.

Vorbeugung: Möglichst früher Release der App in den Stores

Verhalten beim Eintreten: Einen Workaround finden um App auf Geräten zu installieren ohne Veröffentlichung in App-Store

R5 Inkompatibilität mit verschiedenen Geräten

Die App läuft nicht wie vorgesehen auf älteren Devices der Benutzer.

Vorbeugung: Möglichst umfangreiche Tests auf möglichst vielen Devices im Vorfeld.

Verhalten beim Eintreten: Evtl. zurückgreifen auf iPads im Museum

R6 Schlechte Testabdeckung

Automatisierte Unit- und E2E-Tests sind nicht umfangreich genug um ein schnelles Vorankommen zu ermöglichen.

Vorbeugung: Konsequentes Arbeiten nach TDD bei der Core Entwicklung.

Verhalten beim Eintreten: Bei genügend Zeit Tests nachliefern.

R7 Schlechte Architektur

Code wird schlecht aufgebaut was Wartbarkeit und Erweiterbarkeit erschwert.

Vorbeugung: Gängige Entwurfsmuster verwenden.

Verhalten beim Eintreten: Bei genügend Zeit neu entwerfen.

R8 Mangelndes Interesse des Museums

Das Naturmuseum hat wenig Interesse am App oder kann keine Zeit dafür aufwenden. Somit müssten grösste Teile des Inhalts selber ausarbeitet werden.

Vorbeugung: Museum die Möglichkeit geben, zu helfen und mitbestimmen wo sie wollen. Jedoch auch selbständig sein wo es möglich ist.

Verhalten beim Eintreten: Meeting einberufen mit den Beteiligten.

R9 Content

Der Content wird vom Museum nicht zur Verfügung gestellt.

Vorbeugung: Früh genug mit dem Museum kommunizieren, was wir benötigen.

Verhalten beim Eintreten: Inhalte selber zusammenstellen.

R10 Beacon-Installation

Beacon-Installation erweist sich als schwierig.

Vorbeugung: Möglichst umfangreiche Tests auf möglichst vielen Devices im Vorfeld.

Verhalten beim Eintreten: Evtl. Zurückgreifen auf iPads des Museums.

R11.1 Pilottag: App funktioniert nicht

Unerwartete Fehler oder Abstürze am Tag des Probelaufes.

Vorbeugung: Vor dem Pilot auf verschiedenen Devices im Museum testen.

Verhalten beim Eintreten: Leihgeräte zur Verfügung stellen für die Besucher.

R11.2 Pilottag: App nicht lauffähig

Die App ist noch nicht so weit entwickelt, damit sie von den Besuchern genutzt werden kann.

Vorbeugung: Eine Woche vor dem Pilot keine neuen Features mehr entwickeln sondern die App nur noch verbessern.

Verhalten beim Eintreten: Den Pilottag verschieben.

R11.3 Pilottag: App zu komplex

Die Besucher können die App nicht bedienen da sie zu kompliziert ist.

Vorbeugung: Prototyp auch von nicht involvierten Personen abnehmen lassen.

Verhalten beim Eintreten: Hilfestellung vor Ort.

R11.4 Pilottag: Besucher haben keine Interesse

Die Besucher am Pilottag interessieren sich nicht für das App.

Vorbeugung: Im Vorfeld Werbung machen und vor allem vor Ort auf die App aufmerksam machen. Einen Anreiz schaffen wie zum Beispiel Belohnung für Kinder.

Verhalten beim Eintreten: Zweiter Pilottag.

9.9.2 Risikoeinschätzung und Verlauf

Risiko	Maximaler Schaden [h]	EW (Projektbeginn)	EW 25. März	EW 8. April	EW 22. April	EW 6. Mai	EW 15. Mai	EW 18. Mai (Nach Pilottag)	Gewichteter Schaden [h] Projektbeginn	Gewichteter Schaden Projektende [h]
R1	50	10%	10%	5%	5%	0%	0%	0%	5	0
R2	3	50%	40%	40%	40%	30%	10%	0%	1,5	0
R3	20	20%	20%	60%	60%	30%	30%	30%	4	**6**
R4	10	20%	60%	60%	50%	50%	50%	50%	2	**5**
R5	20	40%	50%	50%	50%	40%	30%	30%	8	**6**
R6	25	20%	20%	20%	40%	40%	0%	0%	5	0
R7	40	20%	20%	20%	10%	0%	0%	0%	8	0
R8	30	20%	10%	10%	10%	5%	0%	0%	6	0
R9	15	20%	10%	10%	5%	5%	5%	0%	3	0
R10	3	30%	20%	20%	15%	15%	0%	0%	0,9	0
R11.1	0	20%						0%	0	0
R11.2	0	10%						0%	0	0
R11.3	0	20%						0%	0	0
R11.4	0	40%						0%	0	0

EW= Eintrittswahrscheinlichkeit

9.9.3 Analyse

R3 Performance Probleme

Wiederholt musste mich Performance Problemen auf älteren Geräten gekämpft werden. Diverses Tuning und das Verwenden einen hineinkompilierten Browser hat zu Mehraufwand geführt.

R4 Veröffentlichung im Appstore macht Probleme

Die Publikation im Apple App-Store dauert ca. 3 Wochen. Aufgrund von zeitlichen Problemen war somit ein rechtzeitiges Veröffentlichen nicht möglich. Der Android Playstore benötigt ca. 4 Stunden und war deshalb problemlos machbar.

R5 Inkompatibilität mit verschiedenen Geräten

Da für die Beacon Kommunikation Bluetooth Low Energy (BLE) vorausgesetzt wird, funktioniert das App nur auf neueren Geräten ab ca. dem Jahr 2011. Die iPads der 2. Generation, von welchen das Museum 7 besitzt, unterstützen kein BLE weshalb ein Mehraufwand betrieben werden musste für Abklärungen und um neuere Geräte zu organisieren.

R6 Schlechte Testabdeckung

Zu Beginn bei der Kernachritektur wurde die Korrektheit mit Unit-Tests gesichert. Gegen Ende des Projektes, als es mehr darum ging die Ansichten zu verbessern, wurde aus Effizienzgründen auf Unit-Testing verzichtet. Durch diesen Entscheid wurde aber keine schlechte Architektur gefördert weshalb kein Schaden verbüsst werden musste.

9.10 Benutzeranleitung

Die Benutzeranleitung wird im App unter „Hilfe“ angezeigt.

Du hast dich also dafür entschieden, einen interaktiven Rundgang mit der Museums-App zu unternehmen. Super! Du wirst es nicht bereuen.

Lies zuerst die Anleitung durch, damit du dich nicht verirrst.

Diese App erkennt automatisch, wo du dich im Museum befindest, und kann dir passende Inhalte anzeigen. Wenn du dich also z.B. in die Nähe des Bibers begibst, wird dir die App im unteren Bereich einen Navigationsvorschlag machen:

Wenn du den Navigationsvorschlag anwählst, kommst du in die Detailansicht des zugehörigen Tieres. Hier kannst du Inhalte lesen und ein Quiz lösen.

Um ein Quiz zu lösen musst du es zuerst freischalten. Du kannst ein Quiz freischalten in dem du auf das Stickerbild drückst und dein Mobiltelefon ganz nah an den gleich aussehenden Sticker hältst, der am Ausstellungsobjekt angebracht ist. Die App wird das Quiz freischalten, sobald du nah genug bist.

Sobald du das Quiz gelöst hast, wird dir der zugehörige Sticker in dein Stickerheft geklebt.

Es gibt insgesamt 12 Sticker. Sammle alle ein, um ein Geschenk an der Kasse abzuholen!

9.11 Auswertung Pilottag

Der Pilottag wurde am Internationalen Museumtag am 17. Mai 2015 im Naturmuseum St. Gallen durchgeführt. Aufgrund des Rahmenprogramms, dem Gratiseintritt und der durchzogenen Witterung konnte das Naturmuseum viele Besucher verzeichnen. Es wurde ein Stand im Eingangsbereich aufgebaut und für das App Werbung gemacht.

9.11.1 Besucher Statistik

Besucher insgesamt	745
App Teilnehmer	
Insgesamt (Einzelpersonen oder Familien)	30
Durchschnittliche Anzahl Kinder pro Familie	1.9
Durchschnittsalter der Kinder	8.5 Jahre
Durchschnittsalter Einzelpersonen	30 Jahre
Jüngste Einzelperson	11 Jahre
Älteste Einzelperson	60 Jahre

9.11.2 Fragebogen Auswertung

Um Feedback zu sammeln wurde allen Testern ein Fragebogen ausgehändigt. Nachfolgend die Auswertung. Der Fragebogen befindet sich im Anhang.

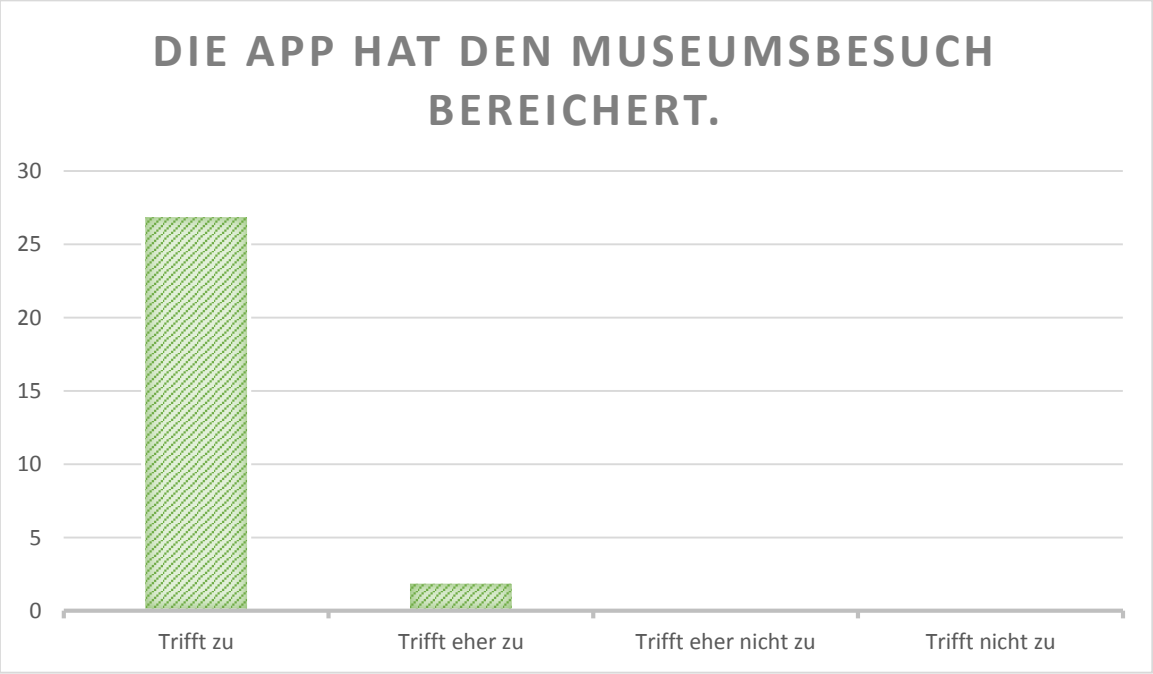


Abbildung 9-30 Pilottag Auswertung 1

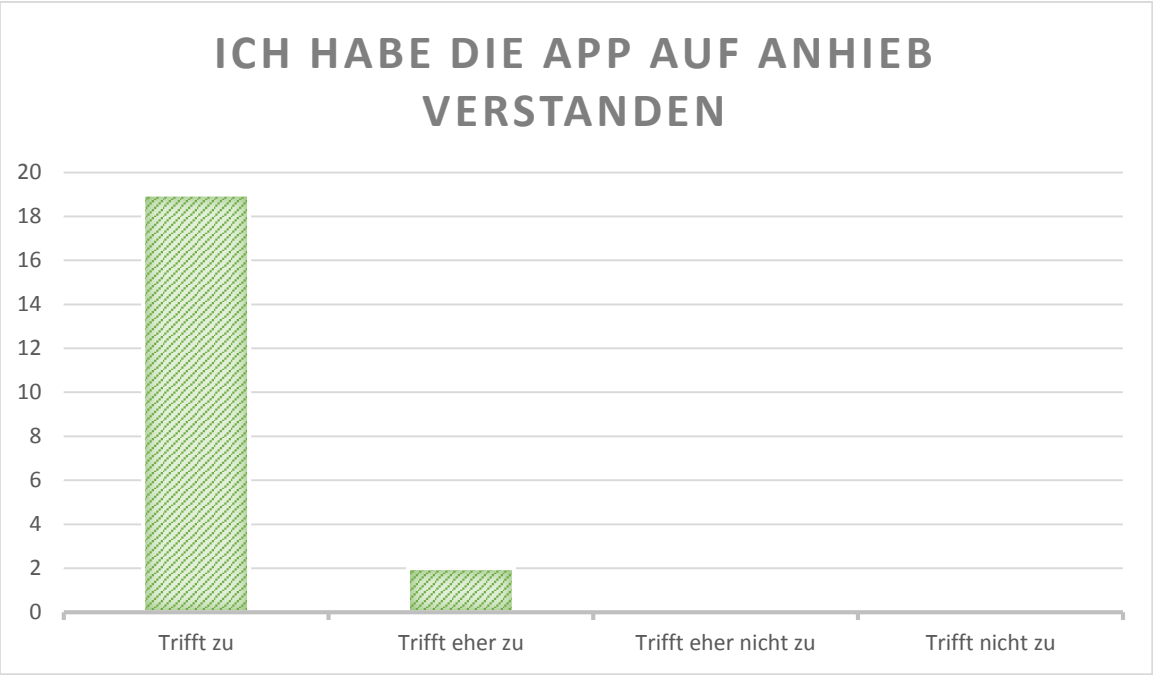


Abbildung 9-31 Pilottag Auswertung 2

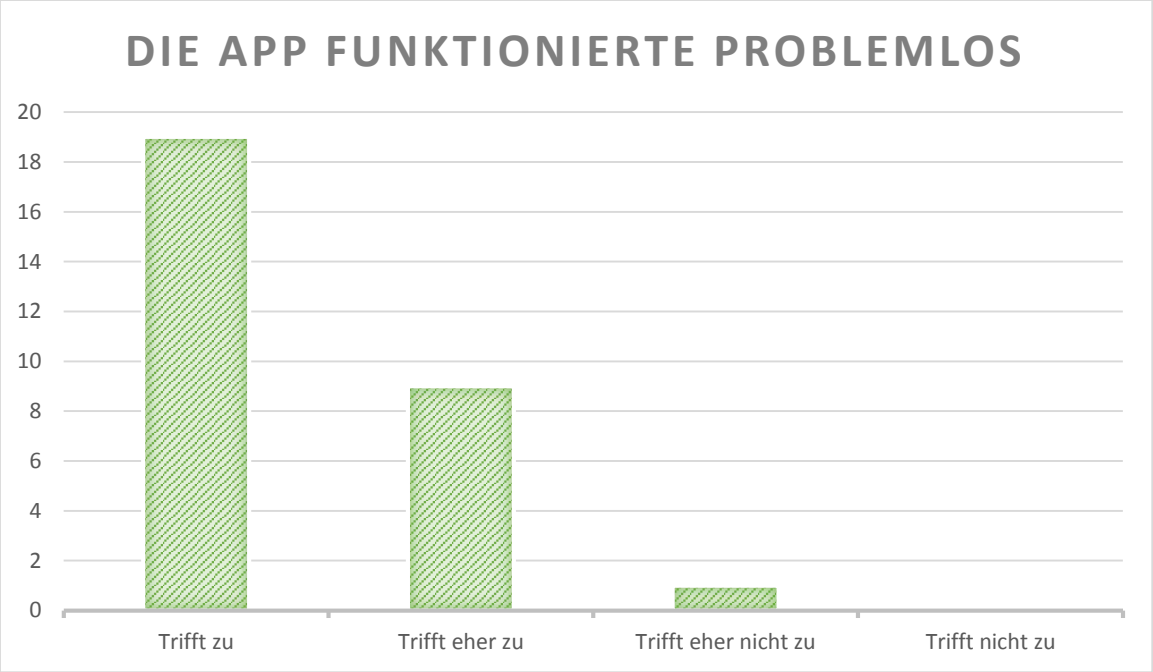


Abbildung 9-32 Pilottag Auswertung 3

Kommentare

Markierung nicht sofort erkannt

Es dauert etwas lange bis Quiz geladen wird (Beacon Kontakt)
3-mal geschrieben

Zuerst haben wir "Kontaktpunkt" auf Tier gesucht

War nicht immer klar wo Infos zu finden sind (Museum oder App)

Schwierigkeit war teilweise, den Sendern zu finden!

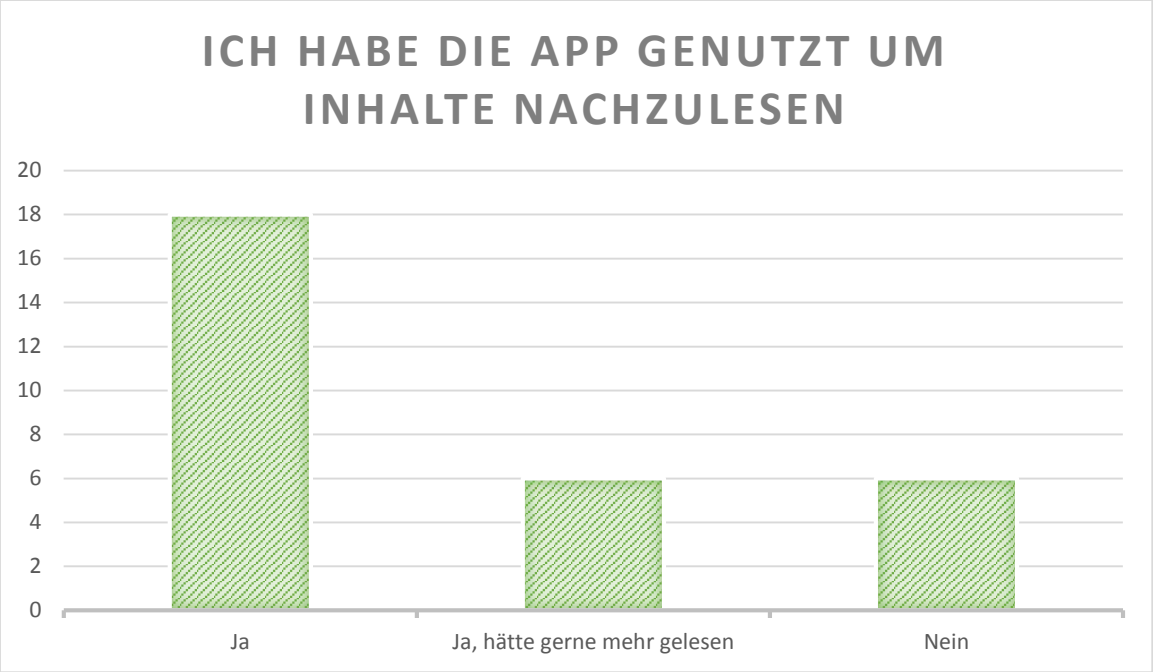


Abbildung 9-33 Pilottag Auswertung 4

Kommentare

Damit wir einige Fragen beantworten konnten
Antwort: Ja

Die Kinder wollten sofort das nächste Quiz lösen
Antwort: Ja

War sehr auf Sticker konzentriert
Antwort: Ja

Da die Ausstellung bereits dokumentiert ist, gibt es eine grosse Menge an Infos, was etwas ermüdet. Für Kinder ist es aber sicher sehr spannend.
Antwort: Ja

Mein Sohn (9) hatte es sehr eilig
Antwort: Nein

Hatte genügend Infos
Antwort: Nein

Mit Kindern (4 und 6) liest man nicht viel nach
Antwort: Nein

Zu wenig für die restlichen Dinge möglich
Antwort: Nein

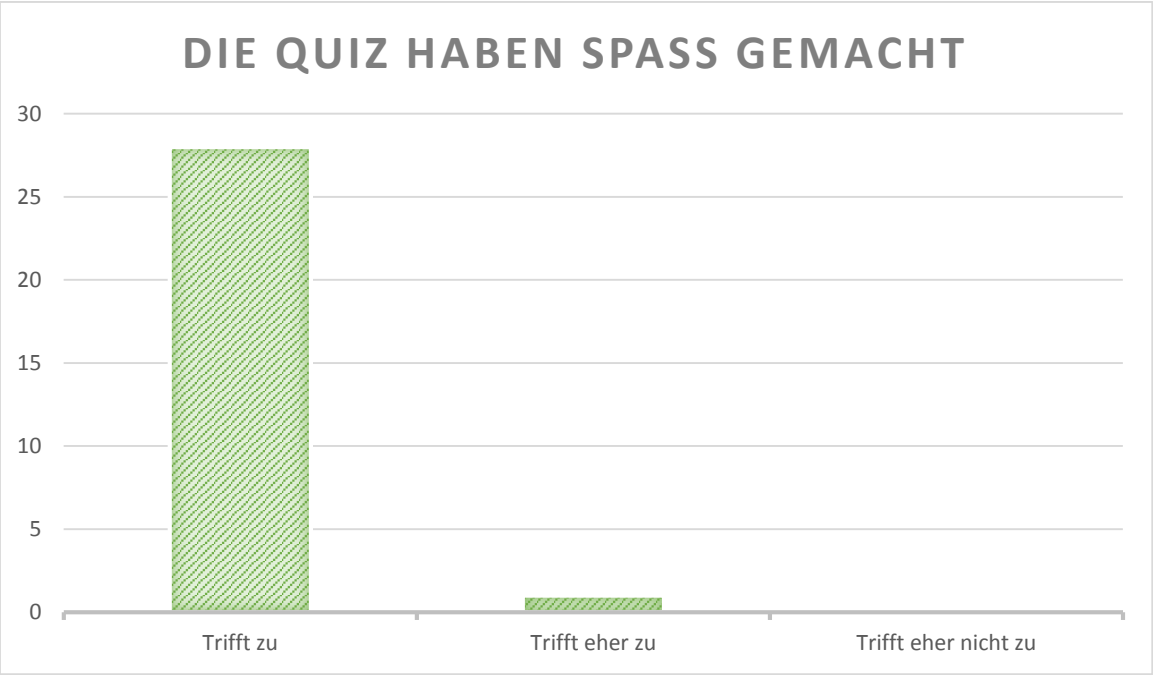


Abbildung 9-34 Pilottag Auswertung 5

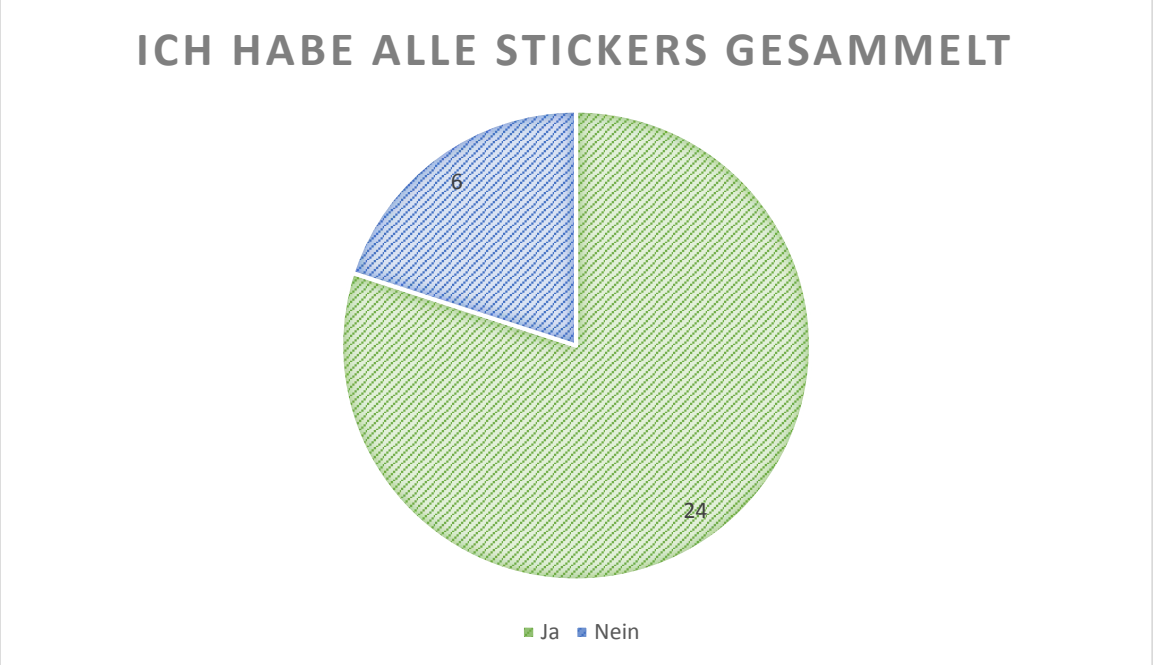


Abbildung 9-35 Pilottag Auswertung 6

Kommentare

Zu wenig Zeit
Antwort: Nein, 3-mal geschrieben

Es ist viel Information
Antwort: Nein

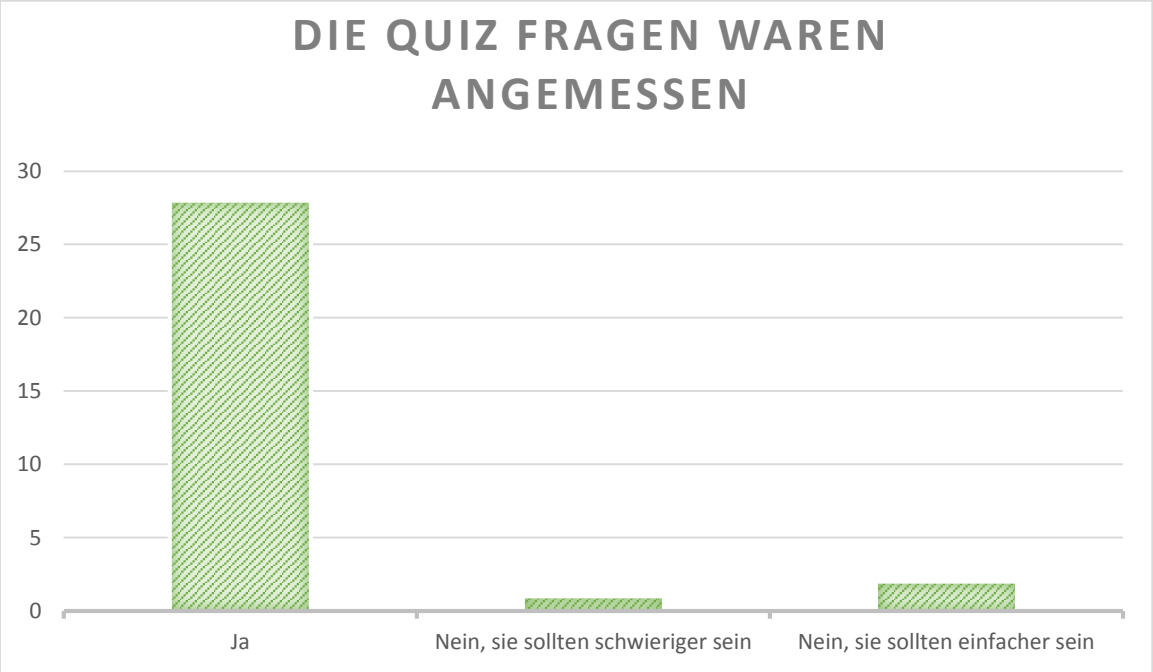


Abbildung 9-36 Pilottag Auswertung 7

Abbildung 9-37 Pilottag Auswertung 8

Weitere Bemerkung

Hat sehr viel Spass gemacht

Merci

Supercool

Genial!

Die Kinder haben schnell gemerkt, dass man einfach eine Antwort nach der anderen drücken kann, bis sie stimmt -> Eventuell max. mögliche Fehlerquote definieren

- Danke –

Funktion mit Kopfhörer, Fragen könnten vorgelesen werden

Es wäre cool, wenn der Weg durch Museum angezeigt wäre

Für uns war es eine lustige Abwechslung zu unserem Alltag, in welchem Computer, iPhone, iPad etc. nicht im Zentrum stehen. Im Allgemeinen finde ich jedoch, dass das App von der Ausstellung und Beobachten ablenkt, Für die Kids steht plötzlich das Medium im Zentrum

Die App ist ansprechend und interessant

Cool gemacht

Ihr habt eine ausgezeichnete App geschaffen. Sie ist klar aufgebaut, informativ und kreativ. Sie lädt zum Ausprobieren und Forschen ein. Für Kinder/Jugendliche ist sie evtl. fast zu umfangreich (-> Geduld aufbringen)
Super Idee und die Qualität sowie Inhalte sind hervorragend

Wenn die Sticker auf Augenhöhe platziert wären, müsste man sich weniger bücken. Ist eine Supersache, es hat viel Spass gemacht. Die Sache mit dem Quiz finde ich sehr originell! Allerdings führt es dazu, dass man noch länger steht und dies hat mich ermüdet. Sitzgelegenheiten wären hilfreich :)

9.11.3 Fazit

Der Pilottag ging ohne Technische Probleme über die Bühne und die App konnte sich der Dauerbelastung stellen. Die 6 Leih-Tablets waren während den 7 Stunden meistens in Gebrauch. Da es von Jung und Alt hauptsächlich angenehme Rückmeldungen gab, kann eine positive Bilanz gezogen werden.

9.11.3.1 Verständnis

Es waren viele verschiedene Gerätetypen zum Verleih Angeboten. Bei ein paar der Geräte war der iBeacon Empfang eher schlecht. Dies führte dazu, dass bei der Quiz Aktivierung viel Geduld aufgebracht werden musste. Bei manchen Benutzern hat dies für Verwirrung gesorgt und dazu geführt, dass sie zu Beginn Mühe hatten die Quiz Aktivierung zu begreifen. Im Allgemeinen sind die Personen aber gut mit der Bedienung zurechtgekommen. Eine kurze mündliche Erklärung der Funktionalität hat gereicht. Es gab den ganzen Tag keine Rückfragen bezüglich der Bedienung.

Für den Fall, dass Besucher auf die Home Taste kommen, sollte man bei Leihgeräten die App möglich im Kioskmodus laufen lassen. So kann sichergestellt werden, dass die App nicht ausversehen geschlossen wird. Bei Android ist dies nur mit einem zusätzlichen App möglich.

9.11.3.2 Mehrwert

Viele Personen haben angegeben die App genutzt zu haben um Inhalte nachzulesen. Eventuell müsste man die Inhalte kategorisieren damit klar wird, welche Inhalte für das lösen des Quiz sinnvoll sind und welche reine Zusatzinformationen enthalten.

Auch mehrmals als Rückmeldung kam der Wunsch nach einer Schwierigkeitsstufe. Somit könnte man sicherstellen, dass alle gleich gefordert werden bei den Quiz.

Wir konnten beobachten dass viele Besucher, auch ältere Leute, ehrgeizig versuchten alle Sticker zu sammeln und interessiert die Inhalte studierten. Durch die App wurde Anreiz geschaffen sich mit den Inhalt auf moderne Art und Weise auseinanderzusetzen. Würde man auf das konstruktive Feedback eingehen, wäre es möglich ein massentaugliches App einzuführen welches als Ergänzung dem Museum einen Mehrwert bringen könnte oder zumindest für Abwechslung sorgt.

9.11.4 Impressionen

Abbildung 9-38 Pilottag Impression 1

Abbildung 9-39 Pilottag Impression 2

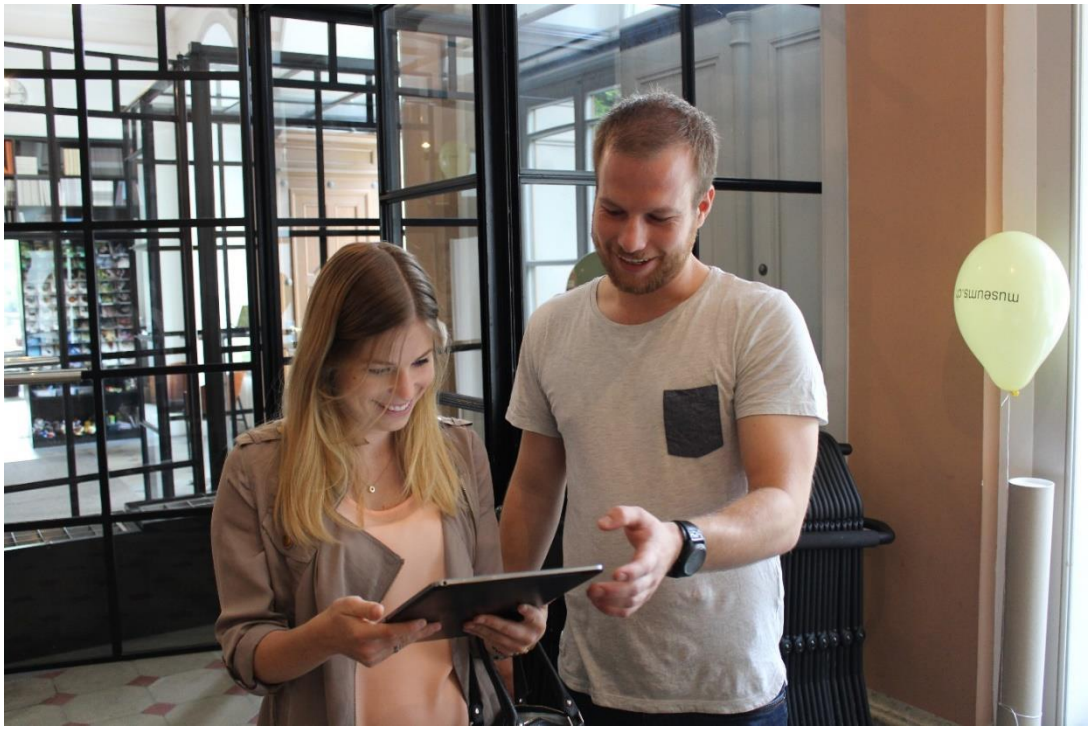


Abbildung 9-40 Pilottag Impression 3

Abbildung 9-41 Pilottag Impression 4

Abbildung 9-42 Pilottag Impression 5

9.12 Testlog

Die App wurde mit den Testläufen vor und während dem Pilottag ausgiebig getestet. Im Folgenden ist eine Auswertung der Non-Functional Requirements, spezifiziert im Anforderungsdokument:

NFR	Erreicht	Bemerkung
Richtigkeit	Ja	Siehe z.B. Assign- oder Order-Quiz
Sicherheit	Ja	App zeigt nur Inhalte an, wenn man in der Nähe von Beacons ist
Reife	Ja	Am Pilottag wurden keine Fehler entdeckt
Fehlertoleranz	Ja	Es gab keine Fehlermeldungen
Benutzbarkeit	Ja	Es existiert eine Hilfeseite
Verständlichkeit	Ja	Gemäss dem eingeholten Review haben sich die Leute mit der App gut zurecht gefunden
Attraktivität	Ja	Siehe z.B. Back Button
Zeitverhalten	Bedingt	Auf einem älteren iPad wurden Lags festgestellt
Verbrauchsverhalten	Ja	Der Akku-Verbrauch wurde jeweils am Ende des Grundganges kurz überprüft. Die Entladung betrug nie mehr als 50%.
Konformität	Ja	Es wurde z.B. Linting eingesetzt
Analysierbarkeit	Ja	Siehe Protokoll vom Codereview
Prüfbarkeit	Ja	Siehe Protokoll vom Codereview
Skalierbarkeit	Ja	Wurde in der Cloud deployed
Lokalisierung	Ja	App ist in Deutsch verfügbar

9.12.1 Anhang: Fragebogen

Museums-App-Fragebogen

Ich bin

❒ Erwachsener mit Kind (___ Kind(er) im Alter von _____ Jahren)
❒ Allein (Alter: ___)

Die App hat den Museumsbesuch bereichert.

❒ Trifft zu ❒ Trifft eher zu ❒ Trifft eher nicht zu ❒ Trifft nicht zu

Ich habe die App auf Anhieb verstanden.

❒ Trifft zu ❒ Trifft eher zu ❒ Trifft eher nicht zu ❒ Trifft nicht zu

Die App funktionierte problemlos.

❒ Trifft zu ❒ Trifft eher zu ❒ Trifft eher nicht zu ❒ Trifft nicht zu

Bemerkungen:

_____

Ich habe die App genutzt um Inhalte nachzulesen.

❒ Ja ❒ Ja, hätte auch mehr gelesen! ❒ Nein

Wenn nein, weshalb?

_____
_____

Die Quiz haben Spass gemacht.

❒ Trifft zu ❒ Trifft eher zu ❒ Trifft eher nicht zu ❒ Trifft nicht zu

Ich habe alle Stickers gesammelt.

❒ Ja ❒ Nein

Wenn nein, weshalb?

_____

Die Quiz Fragen waren angemessen.

❒ Ja
❒ Nein, sie sollten schwieriger sein
❒ Nein, sie sollten einfacher sein

Jedes Museum sollte eine solche App zur Verfügung stellen.

❒ Unbedingt!
❒ Wäre cool, muss aber nicht sein.
❒ Nein.

Weitere Bemerkungen

_____
_____
_____