Zusammenschrift Mobile App for Healthcare

Browser (Sheraz Azad & Sven Marquardt)

Im folgenden werden alle Browser gelistet die in Betracht gezogen werden für die Web Applikation, welche das Arbeiten mit Bluetooth Geräten ermöglichen..

Browser	
Firefox	Firefox besitzt bereits eine Web Bluetooth API, welche jedoch nicht nach dem W3C Standardkonform entspricht, da diese sich erst in den vergangen Jahren entwickelt hat. Da es noch Diskussionen über die W3C Konformen API gibt, ist es noch fraglich ob Firefox diese überhaupt implementiert. Für das Projekt fällt dieser Browser dadurch raus. (siehe hier)
Servo	Firefox implementiert ihre Sachen zuerst in Servo für Testzwecke, um diese ggf. später in Firefox zu übernehmen. Deshalb fällt dieser Browser ebenfalls raus.
Edge	In Microsoft Edge ist das Arbeiten mit Bluetooth Geräten Low-Priority, weshalb dies auch noch nicht soweit entwickelt wurde, das man dies nutzen kann. Auch dieser Browser ist somit raus.
Webkit (Safari)	Bislang gibt es hierzu noch keine Informationen auf der offiziellen Website, Forenbeiträge oder ähnliches, welches auf eine mögliche Implementation deuten können.
Chrome	Auf der offiziellen Seite von Google gibt es bereits ein Tutorial dazu. (siehe hier) Hierzu gehören all die Browser die auf Chromium aufbauen, wie z.B. Opera, Vivaldi and etc. Der Chrome Browser ist der Browser mit der besten Bluetooth API und wird daher für dieses Projekt verwendet.

Smartphones (Sheraz Azad & Sven Marquardt)

Folgende OS wurden für das Projekt genauer betrachtet.

- IOS
  Apple's built-in web Browser Safari unterstützt Web Bluetooth nicht, daher wäre dieses OS damit raus. Es besteht jedoch die Möglichkeit anhand einer Installation einer anderen App auf Web Bluetooth zuzugreifen. (siehe hier) Daher werden IOS Geräte für das Projekt vorerst ausgeschlossen.
- Microsoft Phone
  Es sind Applikationen vorhanden mit denen man iBeacons und Eddystone Beacon kann jedoch sind diese abhängig von dem Microsoft Edge Browser. Dieser wurde jedoch bereits ausgeschlossen, weshalb die Microsoft Phones sinnvoller weise dann auch ausgeschlossen werden.
- Android
  Android Geräte unterstützen die Web Bluetooth API, jedoch erst ab der Version 6.0. Außerdem wurden hier auch noch nicht alle Features implementiert, welches man in folgender Übersicht eindeutig wird.

Betriebssysteme (Sheraz Azad & Sven Marquardt)

Es wurden die üblichen drei BS Windows, Linux und Mac OS mit dem Browser Chrome (ab Version 53) für dieses Projekt betrachtet.

- Windows
  - Windows 8.1 Verbindung nur zu Geräten die schon erkannt worden sind
  - Windows 10 Verbindung möglich zu neuen Geräten
- Linux
  - Ab Kernel 3.19
  - Benötigt BlueZ ab Version 5.41 (How to get Chrome Web Bluetooth working on Linux)
  -
- Mac
  - Ab OS X Sierra
  - Nur Geräte die auch einen Bluetooth Adapter haben der BLE unterstützt
  - nur mit Chrome Browser möglich

Beacon Standard (Sheraz Azad)

Es gibt drei verschiedene Beacon Standards, welche für das Projekt betrachtet worden und im folgenden erläutert werden.

Der Technologie Mark für die Beacons wird von drei Standards dominiert. Der erste Beacon Standard ist der **iBeacon** von Apple, welches der verbreitetste Stand ist. Dieser ist ein proprietärer und geschlossener Standard, welcher zu Beginn nur im Zusammenspiel mit Apple-Endgeräten und der entsprechende iBeacon-App funktionierte. Seit der Android Version 4.3 ist es nun auch möglich mit Android Geräten auf iBeacons zuzugreifen.[1] Die iBeacons übermitteln eine UUID (16-stellige Nummer), Major (vierstellige Nummer) und einen Minor Wert (vierstellige Nummer). Es ist nicht möglich weitere Sachen zu übermitteln oder die iBeacon als Miniserver oder als Ortungsgerät zu verwenden. Die Beacons selber können auch nichts orten, nur die Applikationen könnten diese orten falls sie dafür programmiert wurden. Meistens ist die UUID für alle iBeacons die mit derselben Applikation arbeiten dieselbe, sie unterscheiden sich nur anhand der Major und Minor ID's. Die iBeacons bieten zwei Möglichkeiten Geräte zu finden, zu einem durch Monitoring welches sogar funktioniert wenn die Applikation in einem "killed state" ist und zum anderen durch Ranging welches nur für aktive Applikationen funktioniert.[2]

Dann gibt es den quelloffenen Standard **AltBeacon**. Mit diesem Standard können mit jeder Nachricht mehr Informationen versendet werden als mit dem iBeacon. Der AltBeacon enthält alle Funktionen wie der iBeacon ist jedoch nicht unternehmensspezifisch, jedoch ist es noch nicht so weit verbreitet wie der iBeacon. (Stand 24.5.2015) [3]

Zuletzt gibt es noch den **Eddystone (oder URI) Beacon Standard**, welches ein Google Projekt und ebenfalls quelloffen ist. Diese Beacons enthalten URLs in den Payloads, die sie ähnlich zu den QR-Codes macht, welche über BLE nutzbar sind. Die Eddystone Beacons können vier verschiedene Typen übermitteln (alle funktionieren mit IOS und Android):

- Eddystone-**URL**
    - braucht keine eigene Applikation, funktioniert mit einfachen Beacon Scanner/Browser
    - SDK vorhanden anhand dessen man einfach kompatible Applikationen für Inbound-Marketings entwickeln kann
    - übertragen URLs, welches einfache Webseiten sind, die URLs haben nur eine Länge von 16

---

[1] https://www.expertenderit.de/blog/funktionsweise-nutzungsbeispiele-und-risiken-der-beacon-technologie

[2] https://bkon.com/resources/differences-ibeacon-eddystone-beacons/

[3] https://developer.mbed.org/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/

- Eddystone-**UID**
    - es wird eine Applikation benötigt welches die spezifische UUID des Beacons empfängt und für die App zugänglich macht
    - übermittelt ebenfalls eine 16-stellige Zeichenkette wobei die ersten 10 Zeichen für das “Namespace” und die letzten 6 Zeichen für die “Instance” ID reserviert sind. Das Namespace wird für die Objektidentifikation und die Instance ID für ein individuelles Beacon verwendet.
    - zwei Beacons haben niemals dieselbe UUID
- Eddystone-**EID**
    - funktioniert ähnlich zur Eddystone-UUID, nur dass dieser Standard für eine erhöhte Sicherheit konzipiert wurde
    - EID steht für Ephemeral ID und ist ein random Identifier und enthält keinen Namespace oder eine Instance ID
    - durch die ständige Veränderung der EID wird es geschützt vor hijacking, spoofing und führt dazu, dass ein Konsument durch eine fixe ID Beacon nicht verfolgt werden kann
    - Sorgt auch für Zugriffsbegrenzung, das heißt, dass der Entwickler entscheiden kann Zugriff auf den Beacon hat und wer nicht
- Eddystone-**TLM**
    - überbrückt Applikation und Browser Anwendungen und ist konzipiert neben der UUID oder der UID/URL Beacons zu übermitteln
    - aktuell sendet es Informationen über das Beacon selber wie Batteriestatus, die eigene Temperatur, seit wann es in Betrieb ist usw. [4]
    - Es sind Bytes reserviert um weitere Daten übermitteln zu können

Anders als bei iBeacons und AltBeacons welche eine Datenbank brauchen damit man diese sinnvoll nutzen kann, könnte man die URL des Eddystone Standards nutzen, ohne das eine Applikation dahinter ist, da die eine gültige URL definiert. Außerdem funktionieren diese nicht nach dem System “Set up once and leave running forever” wie die iBeacons und AltBeacons, sie müssen gewartet werden und die Informationen müssen in regelmäßigen Zeitabständen aktualisiert werden.

Im Rahmen des Softwaretechnik Projekts wird der iBeacon Standard verwendet, da dieser am verbreitetsten ist.

[4] https://bkon.com/resources/differences-ibeacon-eddystone-beacons/

Frameworks (Sheraz Azad & Sven Marquardt)

Für das Projekt wurden folgende Frameworks für die Implementierung der Applikation in betracht gezogen. Zu diesem Zeitpunkt ist noch nicht festgelegt ob die Applikationen eine Web Applikation oder eine Native Applikation sein soll.

Gluon

Das Gluon Framework arbeitet mit der Programmiersprache Java 8 und unterstützt die folgenden Plattformen:

- Web
- Android
- IOS
- Desktop

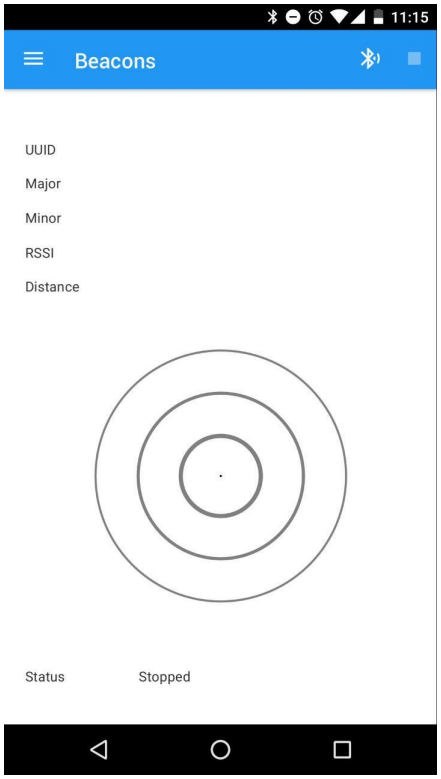
Für diesen Zweck wird Java F mit JavaFXPorts[5] verwendet.
Der Vorteil von Gluon im Rahmen dieses Projekts ist, dass alle Studenten diese Programmiersprachen kennen und keine Einarbeitungszeit benötigt wird. Es ist möglich den Code zu allen gängigen Plattformen zu kompilieren, wobei es zusätzlich noch möglich ist aus dem Code eine Website generieren zu lassen. Gluon wird in das OpenJDK eingegliedert und der Support soll sich in den nächsten Jahren verbessern und es wird so zu einem Offiziellen Projekt der Java Gruppe.

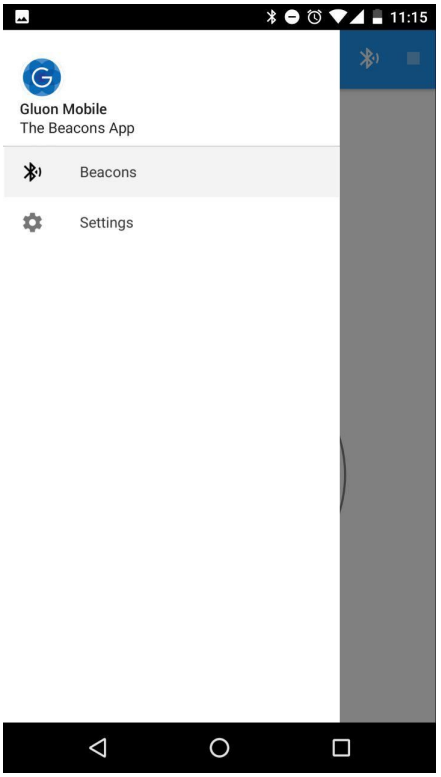
Lizenzen:

- JavaFXPorts selber steht unter der freien Lizenz (kommerzieller und privater Nutzen ist frei)
- Gluon Charm Down BSD
    - für den Zugriff auf die Hardware der Systeme
- Gluon Maps GPL
    - einbinden von Kartendaten
- Gluon VM Opensource
- Gluon Connect BSD
    - einheitliche API zum verbinden zu externen Ressourcen
- Gluon selber steht unter keiner freien Lizenz, es betrifft folgende Zusatzprodukte, welche jedoch im Rahmen dieses Projekts nicht benötigt werden
    - Gluon Cloudlink ähnlich zu Google Firebase
    - Gluon Mobile bessere Unterstützung von JavaFXPorts
    - Gluon Desktop bessere Unterstützung für Desktop Applikationen

[5] http://gluonhq.com/products/mobile/javafxports/

Anhand eines kleinen Prototypen war es möglich sich bereits mit dem iBeacon verknüpfen zu können.[6] Folgende Abbildungen zeigen den Prototypen, der für die Verbindung zum iBeacon implementiert wurde.





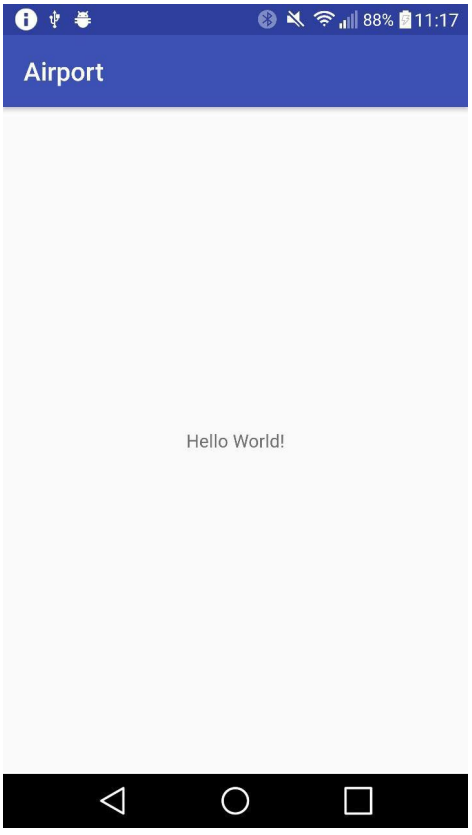
[6] https://github.com/gluonhq/gluon-samples/blob/master/beacons/README.md

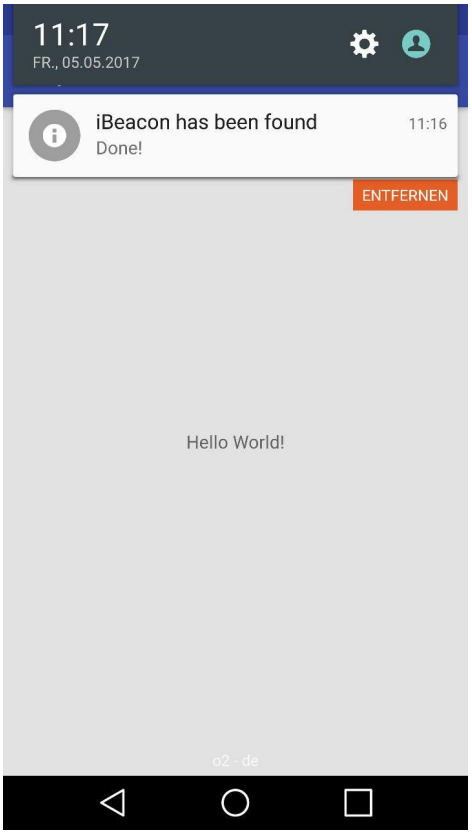
Android + Estimote SDK (Sheraz Azad & Sven Marquardt)

Auf der offiziellen Seite[7] der Estimote Beacons gibt es ein kleines Tutorial, welches einem das Verbinden zu den Beacons genau und in einfachen kleinen Schritten erklärt.[8] Dadurch ließ sich das Implementieren eines Prototypen und verbinden mit einem iBeacon einfach realisieren. Für die Implementierung des Prototypen wurde folgendes benötigt:

- Android Studio
- Android SDK
- Estimote SDK
- und der Estiomte Beacon

Es wurde ein Prototyp geschrieben der eine *Notification* sendet wenn ein iBeacon gefunden. Folgende Abbildungen zeigen den Prototypen, der für die Verbindung zum iBeacon implementiert wurde.





[7] http://estimote.com
[8] http://developer.estimote.com/android/tutorial/part-1-setting-up/

Cordova (Kim Huber & Ole Petersen)

Da sich herausgestellt hat, dass die Realisierung der App als reine Webanwendung nicht möglich ist, wurde sich im Rahmen der Suche nach einer Alternative mit dem Cross Compiling Framework Cordova von Apache beschäftigt.

Apache Cordova wurde gewählt, da es gestattet, hybride Apps mit der Verwendung von HTML5, CSS und JavaScript zu entwickeln und diese dann für unterschiedliche Plattformen zu übersetzen. Dadurch können bereits existierende Webanwendungen leicht in eine App integriert werden. Die Präsentation wird mittels eines HTML-Dokuments (WebView) realisiert, während die Möglichkeit existiert, via Plugins auf die verschiedenen APIs eines Gerätes zuzugreifen, um z.B. Zugang zu Kamera, Speicher oder Bluetooth zu erhalten (Abbildung 1). Bei einer reinen Webanwendung wird dies zwar auch ermöglicht, allerdings nur mit großen Einschränkungen. Um auf eine Gerätefunktion zuzugreifen, wird im HTML5-Dokument auf das Cordova-Framework zugegriffen, welches sich bei der Installation auf dem Endgerät die Zugriffsberechtigungen sichert und einen uneingeschränkten Zugriff ermöglicht. Unterstützt werden dabei die gängigen Plattformen Android, iOS und Windows Phone.

Um die Gebrauchstauglichkeit von Cordova für unser Projekt zu testen, wurde anhand von drei Prototypen überprüft, ob ein Endgerät (Samsung Galaxy S4 Mini) mit einem iBeacon interagieren kann, sowie die Möglichkeit existiert, einen zweidimensionalen Code zu scannen.

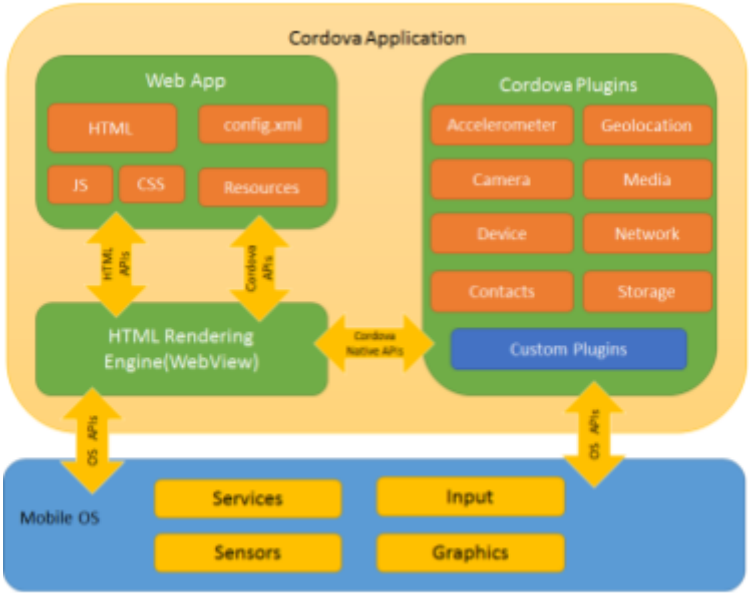


Abbildung 1: Architektur Cordova
(Quelle: https://cordova.apache.org/docs/en/7.x/guide/overview/index.html)

1. Prototyp: iBeacon-Scanner ohne iBeacon-Plugin

   Bei diesem Prototypen handelt es sich um eine Beispielapp von Intel zum scannen nach Bluetooth Low Energy-Geräten. Die App ist in der Lage die Geräte-IDs von Bluetoothgeräten auszulesen. Ein Nachteil hierbei ist jedoch, dass nicht nur iBeacons, sondern alle Bluetoothgeräte in der Umgebung gefunden werden. Dies ist jedoch ausreichend um zu belegen, dass hybriden Apps, die mit Apache Cordova erstellt wurden, in der Lage sind mit iBeacons zu interagieren. Die App wurde mit AngularJS und dem Intel-Framework für mobile Apps implementiert. Mithilfe von Cordova wurde eine APK-Datei generiert und auf dem Endgerät installiert.

   Ergebnis: Die App war wie erwartet in der Lage die IDs der Bluetoothgeräte in der Umgebung anzuzeigen.

2. Prototyp: iBeacon-Scanner mit iBeacon-Plugin

   Aufgrund dessen, dass sämtliche Bluetoothgeräte gefunden wurden, wurde ein Prototyp mithilfe eines iBeacon-Plugins für Cordova erstellt, welcher für das Suchen nach iBeacons ausgelegt ist. Bei dem verwendeten Plugin handelt es sich um das Open Source-Projekt von Peter Somogyvari (siehe hier), das es erlaubt nach spezifischen UUIDs, sowie Minor und Major des iBeacons zu scannen. Es wurde dazu das vom Ersteller des Plugins geschriebene Script “Start monitoring a single iBeacon” angepasst und mithilfe von Cordova erneut eine APK-Datei generiert und auf dem Endgerät installiert.

   Ergebnis: Die App war in der Lage das getestete iBeacon gezielt anzusprechen, allerdings gab es teilweise Probleme, da bei einigen Scanvorgängen das iBeacon nicht entdeckt wurde.

3. Prototyp: iBeacon-Scanner mit iBeacon-Plugin

   Auch bei dieser App handelt es sich um ein Beispiel von Intel, welche die integrierte Kamera des Endgeräts verwendet, um einen Barcode zu scannen und seine Informationen anzuzeigen oder zu verwenden.

   Ergebnis: Die von Cordova erstellte hybride App war in der Lage, ohne Probleme Barcodes zu scannen und die Informationen anzuzeigen.

Zusammenfassung

Die Möglichkeit Webanwendungen in eine App einzubinden ist ein klarer Vorteil gegenüber den Alternativen. Allerdings kann bei dem verwendeten Plugin vorläufig keine 100%ige Sicherheit gewährleistet werden, dass ein iBeacon gefunden wird. Da Cordova aber alle erforderlichen Bedingungen erfüllen konnte und bereits eine Teillösung existiert, lässt sich sagen, dass Cordova für dieses Projekt ein geeignetes Framework darstellt.

Prototypen die noch erstellt werden (Sheraz Azad & Sven Marquardt)

Es werden im weiteren Verlauf des Projektes, vor der Entscheidung welches Framework für die mobile Applikation verwendet wird, werden noch Prototypen für Flutter (Sven Marquardt) und Xamarin (Sheraz Azad) erstellt.

IDE für das Projekt (Sheraz Azad & Sven Marquardt)

Nach der Erstellung diverser Prototypen entschied sich die Gruppe für die IDE Xamarin um eine Cross-Plattform Applikation zu entwickeln.