

# Softwareprojekt: Mobile Healthcare

## Entwicklerhandbuch Mobile App

Sheraz Azad, Daniel Dzimitrowicz, Jonas Grams,

Malte Grebe & Niklas Klatt

Stand: 09. Juli 2017

### Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>2</b>
1.1 Beschreibung der Funktionen des Programms	3
<b>2 Installation und Bedienungsanleitung</b>	<b>4</b>
2.1 Voraussetzungen	4
2.1.1 Windows	4
2.1.2 Mac OS	4
2.2 Installation unter Windows/Mac	4
2.3 Bedienungsanleitung	5
<b>3 Architektur</b>	<b>6</b>
3.1 UML (Model)	6
<b>4 Namespace Struktur</b>	<b>7</b>
<b>5 Klassen</b>	<b>9</b>
5.1 /Healthcare	9
5.1.1 Model	9
AppCore	9
Global	9
5.1.1.1 Database	9
5.1.1.2 Entity	11
5.1.1.3 Enumeration	13
5.1.1.4 REST	13
5.1.1.5 Serializer	14
5.1.1.6 Service	14
5.1.1.7 Synchronization	14
5.1.2 View/ViewModel	15
5.1.2.1 MVVM	16

5.1.2.2 Data Bindings	16
5.1.2.3 Klassenbeschreibung View	17
5.1.2.3 Klassenbeschreibung ViewModel	21
5.2 /Healthcare.Droid	23
5.2.1 Helper	23
5.2.2 Service	23
5.2.3 Database	24
5.3 /Healthcare.iOs	24
5.3.1 Service	24
5.3.2 Database	24
/Healthcare	25
/Healthcare.Droid	26
/Healthcare/iOS	27
<b>7 Software Funktionen</b>	<b>27</b>
7.1 Bluetooth (Beacon) Suche	27
7.2 QR Code Scanner	29

## 1 Einleitung

Die vorliegende Software, für eine mobile Applikation, ist das Ergebnis des Softwareprojekts ‘Mobile Healthcare’ an der Fachhochschule Lübeck im Sommersemester 2017. Sie wurde durch die auf dem Deckblatt angegebenen Studenten in Gruppenarbeit entwickelt und dokumentiert. Das gesamte Projekt unterliegt der Apache Lizenz 2.0, welche auf der offiziellen Website genauer erläutert ist. <http://www.apache.org/licenses/LICENSE-2.0>

[Sheraz Azad]

### 1.1 Beschreibung der Funktionen des Programms

Es wurde eine grafische Benutzeroberfläche in Xamarin C# entwickelt, anhand dessen mit Cyber-physischen-Objekten kommuniziert werden kann. Im Rahmen des Projektes wurden Bluetooth Low Energy (BLE) iBeacons verwendet, die in regelmäßigen Abständen ihre Identifier übermitteln. iBeacons sind ein Apple-Standard welche eine UUID, einen Major und einen Minor als Identifier besitzen. Anhand der Identifier ist es möglich die Cyber-physischen-Objekte klar voneinander zu trennen und präzise Informationen zu übermitteln. Diese Informationen werden verwendet um Aufträge zu den dazugehörigen Objekten zu übermitteln und dem User auf die Aufträge in der Nähe aufmerksam zu machen und diese in der App anzuzeigen. Um diese und weitere Funktionalitäten bereitzustellen wurde eine interne Datenbank implementiert welche, sofern eine Verbindung zum Server besteht, mit diesem verglichen und aktualisiert wird. Durch die interne Datenhaltung ist es darüber hinaus möglich auch andere Aufträge aus dem Arbeitsbereich des Nutzers zur Verfügung zu stellen und ihm damit eine effizientere Verwaltung seiner Tätigkeit zu ermöglichen.

[Sheraz Azad, Jonas Grams]

## 2 Installation und Bedienungsanleitung

### 2.1 Voraussetzungen

#### 2.1.1 Windows

Zum entwickeln der Applikation wurde die IDE Visual Studio 2017 verwendet, welches während der Installation die zusätzliche Installation von Xamarin anbietet.

#### 2.1.2 Mac OS

Die mobile Applikation setzt die installierte Visual Studio IDE oder Xamarin Studio IDE und die aktuellste Version von XCode voraus. Die Software wurde erfolgreich mit *Visual Studio for Mac Community Version: 7.01 (build 24)*, mit *Xamarin Studio Community Version: 6.3 (build 863)* und *XCode Version 8.3.3 (8E3004b)* getestet. Möglicherweise könnte diese auch unter älteren Versionen ausgeführt werden, dies kann jedoch nicht garantiert werden.

[Sheraz Azad]

### 2.2 Installation unter Windows/Mac

Für die Installation der jeweiligen IDE, unter den angegebenen die dmg Datei herunterladen und den Anweisungen auf dem Bildschirm folgen.

- Visual Studio: <https://www.visualstudio.com/de/>
- Xamarin Studio: [https://developer.xamarin.com/guides/ios/getting\\_started/installation/mac/manual\\_installation/](https://developer.xamarin.com/guides/ios/getting_started/installation/mac/manual_installation/)
- XCode: <https://itunes.apple.com/de/app/xcode/id497799835?mt=12>

Nach der erfolgreichen Installation der gewünschten IDE, muss die HealthcareApp.zip in einem gewünschten Ordner entpackt werden. Nun kann die *HealthcareApp.sln* in die IDE importiert werden.

Das Getting Started [https://developer.xamarin.com/guides/cross-platform/getting\\_started/](https://developer.xamarin.com/guides/cross-platform/getting_started/) ist sowohl für Xamarin als auch für Visual Studio geeignet. Dasselbe gilt auch für die ausführliche API Dokumentation <https://developer.xamarin.com/api/>. Es müssen keine

besonderen Einstellungen in der IDE vorgenommen werden um das Projekt starten zu können. Die Software lässt sich sowohl im Emulator als auch an einem physischen Gerät ausführen.

### 2.3 Bedienungsanleitung

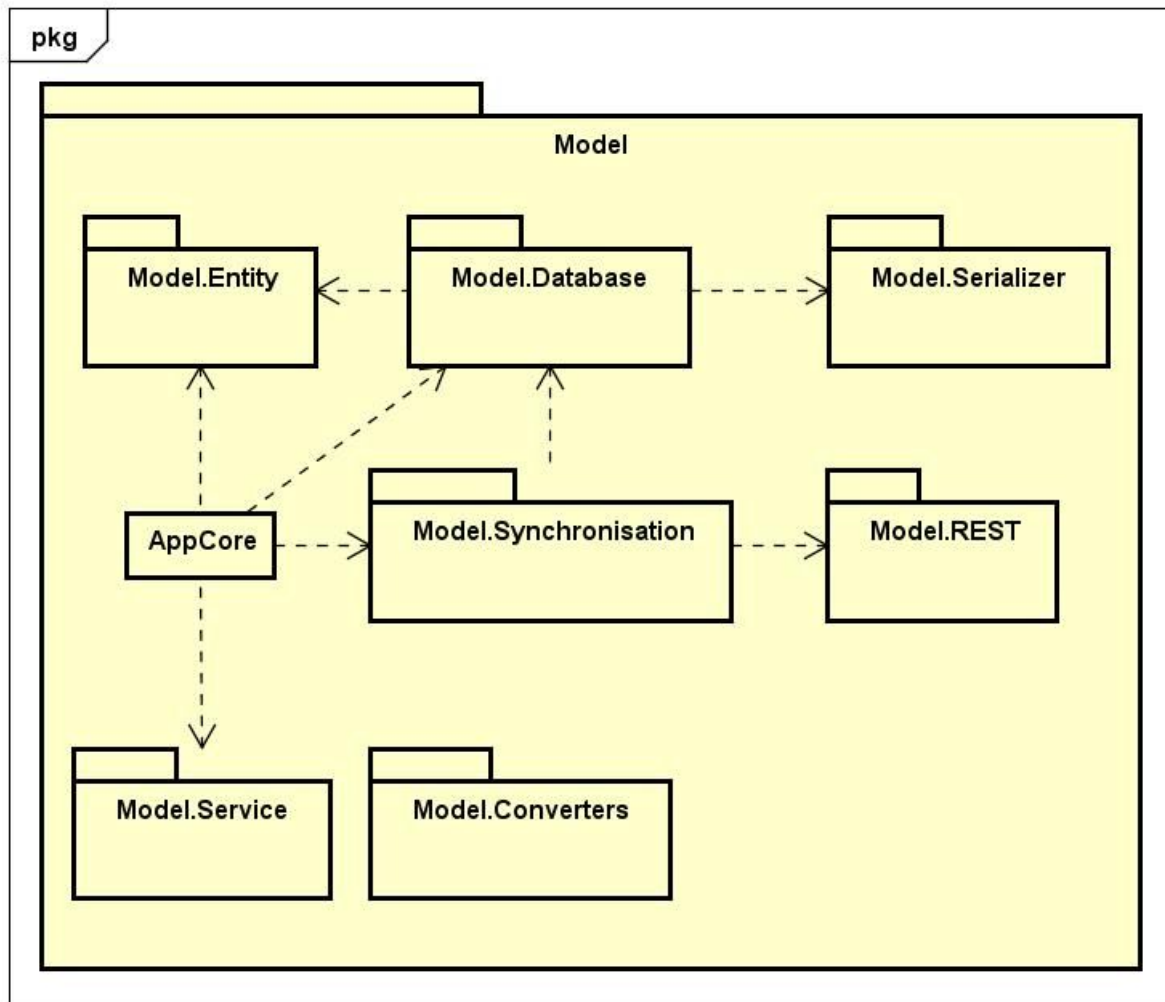
Um die Funktionstüchtigkeit der Applikation auf den eigenen Bedarf anzupassen, sind in der Datei */HealthcareApp/Model/Global.cs* einige Variablen angelegt, welches diese Anpassungen ermöglichen:

- *restAd* enthält die URL, an welche die Rest-Anfragen gesendet werden
- *serverIsInLocalNetwork* beeinflusst die Einstellung im *WifiConnectionManager* und entscheidet darüber, ob beim Verbindungsaufbau Adressen außerhalb des lokalen Netzes zugelassen werden
- *server, port* und *timeout* haben Einfluss auf die Verbindungsprüfung durch *WifiConnectionManager*
- *beaconInRange* gibt die Reichweite in Metern an, in der nach iBeacons Ausschau gehalten werden soll
- *synchIntervallInMin* gibt den Zeitintervall in Minuten an, in dem während einer bestehenden Serververbindung erneut synchronisiert werden soll
- *roleNameEmployee* & *roleNameTech* geben die Bezeichnungen der Rollen an, die eine Berechtigung zur Nutzung der Applikation haben. Sollten weitere Rollen angelegt werden, müssen diese ebenfalls in *AppCore* ergänzt werden

[Sheraz Azad, Jonas Grams]

## 3 Architektur

### 3.1 UML (Model)



[Niklas Klatt]

Das Model der App besteht aus verschiedenen Teilen, welche im AppCore zusammenlaufen und von dort aus gesteuert werden. Der Datenbank-Namespace ist zum Speichern und Aufbewahren der verschiedenen Aufträge, der REST-Namespace für die Kommunikation mit dem Server über REST-Anfragen, Synchronization-Namespace für das überprüfen einer Verbindung zum Server sowie auf Aktualisierungen der Aufträge abfragen und der Serializer-Namespace für den BinarySerializer, welcher zum Serialisieren und Deserialisieren der Objekte für die Datenbank genutzt wird. Im Service-Namespace befindet sich der Bluetooth-Service.

[Daniel Dzimitrowicz]

## 4 Namespace Struktur

Das Projekt hat eine PCL-Struktur ([https://developer.xamarin.com/guides/cross-platform/application\\_fundamentals/pcl/](https://developer.xamarin.com/guides/cross-platform/application_fundamentals/pcl/)), weshalb der Quellcode des Projektes in drei verschiedenen Namespaces liegt. Im folgenden Link wird genau erklärt was der Vorteil von der PCL-Struktur gegenüber der Shared Struktur ist <http://hotcrossbits.com/2015/05/03/xamarin-forms-pcl-vs-shared-project/>.

Das Package */HealthcareApp* ist das Shared Package und enthält Cross Plattform Code. In diesem Package gibt es folgende Unterverzeichnisse:

- Converters Enthält Konverter zum umwandeln von Daten
  - DateConverter Wandelt die von C# verwendete DateTime in ISO 8601 und umgekehrt
  - SelectedItemEventArgsToSelectedItemConverter Konvertiert ein ausgelöstes Event beim bedienen des Touchpads in das entsprechende Item
- Model Enthält die Logik des Projektes
  - AppCore Stellt die Verbindung der einzelnen Module dar
  - Global Enthält Konstanten für den Betrieb der Applikation
  - Database Enthält Tabellen für die Datenbank
  - Entity Enthält Klassen für die Datenbank und für den REST Service
  - Enumeration Enthält die Enumerationen des Projekts
  - REST Enthält die Logik für den REST Service
  - Serializer Enthält Klassen für das Serialisieren
  - Service Enthält die Logik für das Beacon Service
  - Synchronization Enthält die Logik für die Synchronisierung mit der DB
- View Enthält die Spezifikationen für die Benutzeroberfläche
- ViewModel Enthält die Logik für die Benutzeroberfläche

Das Package */HealthcareApp.Droid* enthält Android spezifischen Code. In diesem Package gibt es folgende Unterverzeichnisse:

- Helper Enthält die Logik für die Notifications
- Service Enthält die Plattform spezifische Implementation des Beacon Service
- Datenbank Enthält die Plattform spezifische Implementation für die Datenbank

Das Package */HealthcareApp.iOS* enthält Android spezifischen Code. In diesem Package gibt es folgende Unterverzeichnisse:

- Service Enthält die Plattform spezifische Implementation des Beacon Service
- Datenbank Enthält die Plattform spezifische Implementation für die Datenbank

[Sheraz Azad]



## 5 Klassen

### 5.1 /Healthcare

#### 5.1.1 Model

##### AppCore

Stellt das Bindeglied zwischen den einzelnen Modulen der Applikation dar. Dies wurde gerade im Bezug auf die View häufig mit Hilfe des “[Xamarin.Forms MessagingCenter](#)” umgesetzt. Zusätzlich ist AppCore auch für die korrekte Filterung und Verarbeitung der Aufträge verantwortlich. Ebenfalls hält AppCore sämtliche Informationen für die einzelnen Module bereit und sorgt für die Kommunikation mit der internen Datenbank und löst unter den richtigen Voraussetzungen eine Synchronisation mit dem Server aus.

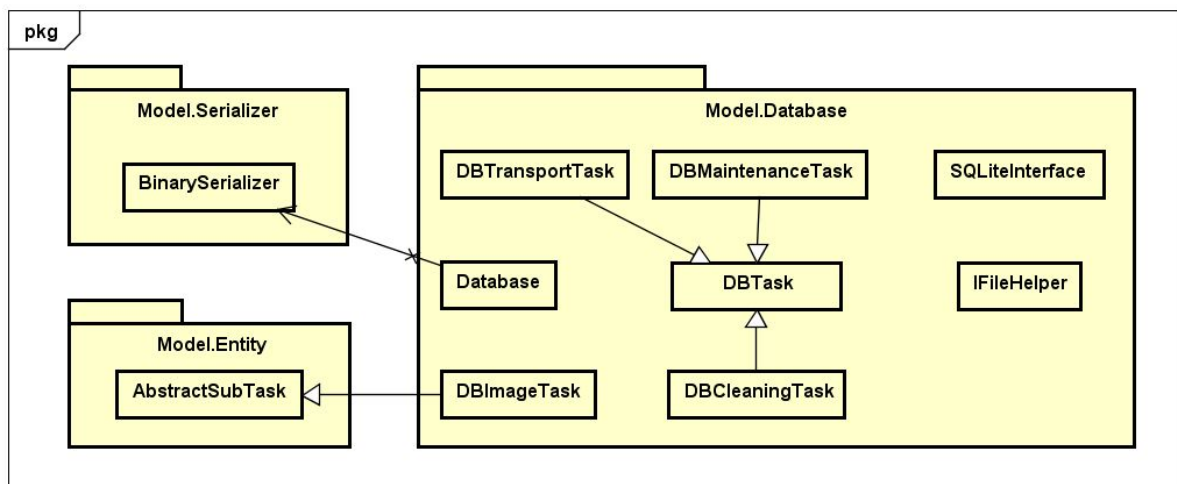
[Jonas Grams]

##### Global

Enthält eine Reihe von konstanten Parametern, wie zum Beispiel der Serveradresse, die in verschiedenen Modulen der Applikation Verwendung finden.

[Jonas Grams]

#### 5.1.1.1 Database



Der Database Namespace beinhaltet alle Klassen, welche die Datenbank benötigt, sowie die Interfaces, welche als Plattform-Spezifischer Code implementiert werden müssen. Die

Datenbank nutzt den BinarySerializer, um die ihr übergebenen Objekte beim Speichern zu Serialisieren und beim Lesen zu Deserialisieren. [Daniel Dzimitrowicz]

### Database

In der Datenbank.cs-Datei liegt die komplette Logik der Datenbank, sowie die Methoden welche sie nach außen hin zur Verfügung stellt. Die Datenbank speichert alle Objekte in speziellen DB-Surrogaten der echten Objekte ab. Der Inhalt dieser DB-Objekte besteht meist aus einer ID, welche den Primärschlüssel ausmacht, einer UUID sowie dem Byte-Array in welchem das Ursprungs Objekt kodiert ist.

[Daniel Dzimitrowicz]

### DBCleaningTask

Ein Surrogat für den normalen CleaningTask, um ihn in die Datenbank speichern zu können.

Ist von DBTask abgeleitet.

[Daniel Dzimitrowicz]

### DBMaintenanceTask

Ein Surrogat für den normalen MaintenanceTask, um ihn in die Datenbank speichern zu können. Ist von DBTask abgeleitet.

[Daniel Dzimitrowicz]

### DBTransportTask

Ein Surrogat für den normalen TransportTask, um ihn in die Datenbank speichern zu können.

Ist von DBTask abgeleitet.

[Daniel Dzimitrowicz]

### DBTask

Ein Surrogat für den abstrakten Task, um ihn in die Datenbank speichern zu können.

[Daniel Dzimitrowicz]

### DBImageTask

Ein surrogat für den ImageTask, um ihn in die Datenbank speichern zu können. Ist von AbstractSubTask abgeleitet.

[Daniel Dzimitrowicz]

### IFileHelper

IFileHelper ist ein Interface für das erstellen des Pfades zum Speicherort der Datenbank, welches anhand vom Dependency Services auf jeder Plattform implementiert wird.

[Daniel Dzimitrowicz]

### SQLiteInterface

Das SQLiteInterface wird für den Dependency Service genutzt.

[Daniel Dzimitrowicz]



### PictureOfObject

Einhält die Binärdaten eines Bild von dem Beacon selber. Muss extra gespeichert werden, da der REST Aufruf separat passiert.

### Beacon

Repräsentiert den Zustand des Beacons mit UUID, Major und Minor.

### Location

Der Ort, da dem sich der Beacon befindet.

### User/Role

Der Benutzer des System und seine Rolle innerhalb der Anwendung. Über die Rolle werden Filter gesetzt und Berechtigungen geprüft.

### CleaningTask/TransportTask/MaintenanceTask

Spezielle Aufträge, die Auftragspezifische Informationen enthalten.

### AbstractSubTask

Subtask sind die möglichen Rückmeldungen, die ein Nutzer geben kann. Es gibt verschiedene Ausprägungen der SubTasks.

### SubTaskSlider

Der Nutzer kann mit hilfe eines Sliders eine Beurteilung vornehmen.

### SubTaskCheckBox

Der Nutzer kann mit hilfe einer Checkbox Rückmeldung über den Erfolg eines Auftrags geben.

### SubTaskImage

Der Nutzer kann in Form eines Fotos Rückmeldung geben.

[Niklas Klatt]

### 5.1.1.3 Enumeration

#### Major

Die Enumeration Major ist für die iBeacons um anhand der Major Identifier unterscheiden zu können, ob das Cyber-physische-Objekt für einen Techniker oder einen Arbeiter sichtbar ist.

[Sheraz Azad]

#### Minor

Die Enumeration Minor ist für die iBeacons um anhand der Minor Identifier unterscheiden zu können, ob das Cyber-physische-Objekt eine Brandschutztür oder ein Bett ist.

[Sheraz Azad]

#### TaskStatus

Die Enumeration TaskStatus ist um herauszufinden ob der Status eines Auftrag undefiniert, offen, angenommen oder abgeschlossen ist.

[Sheraz Azad]

### 5.1.1.4 REST

#### RestMng

Die Rest-Anfragen die benötigt werden, um eine sinnvolle Kommunikation zwischen App und Server zu gewährleisten. GET-Requests zu den einzelnen Tasks werden hier mittels POST-Requests umgesetzt, da es mit Xamarin nicht möglich ist mit einem GET-Request einen Body zu übergeben.

[Jonas Grams]

### 5.1.1.5 Serializer

#### BinarySerializer

Der BinarySerializer wird genutzt um alle Objekte welche in die lokale Datenbank gespeichert werden sollen, in ein Byte-Array umzuwandeln und in der jeweiligen surrogat

DB-Klasse zu speichern. Ebenfalls wird er dazu genutzt aus diesen Byte-Arrays die Ursprungs Objekte wiederherzustellen.

[Daniel Dzimitrowicz]

#### 5.1.1.6 Service

##### IBeaconService

IBeaconService ist ein Interface für den Beacon Service, welches anhand von Dependency Services in jeder Plattform implementiert wird.

[Sheraz Azad]

##### EventListUpdate

Enthält den EventHandler um die Liste zu aktualisieren. Dieser EventHandler wird in den Plattform spezifischen Codes aufgerufen.

[Sheraz Azad]

##### BLEManager

Der BLEManager initialisiert den Beacon Service und ruft den Plattform spezifischen Code auf. Außerdem schickt es die neuen Aufträge an das Model, welches dann die View aktualisiert.

[Sheraz Azad]

#### 5.1.1.7 Synchronization

##### SynchronizationManager

Enthält Methoden für einen effizienten Austausch von Server und App. Dies beinhaltet das Filtern von Aufträgen und das Verwalten von Zugriffszeiten.

[Jonas Grams]

##### WifiConnectionManager

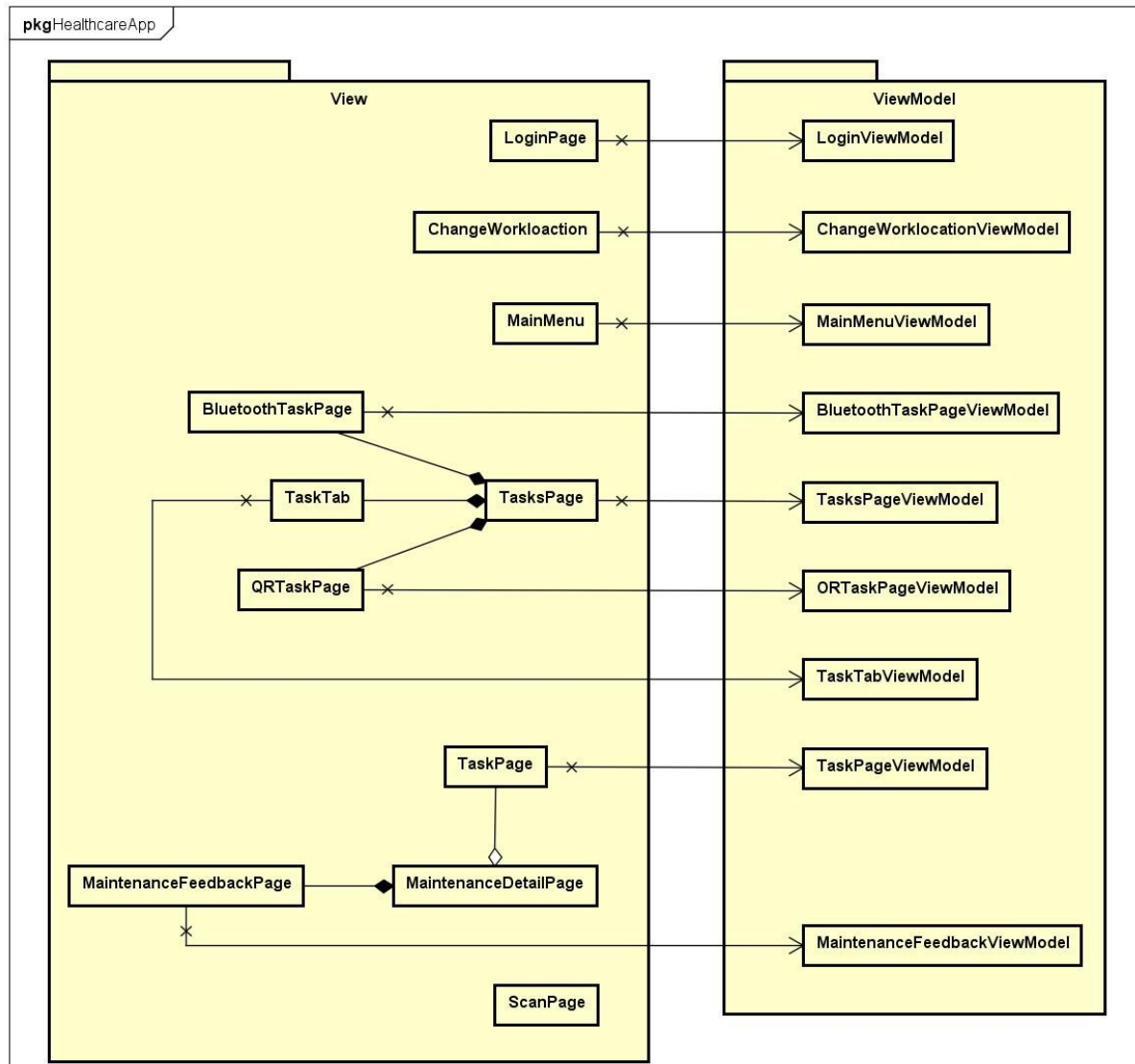
Ist verantwortlich für die Prüfung auf eine bestehende Verbindung zum Server und gibt somit Auskunft über den aktuellen Status dieser Verbindung. Zur Implementierung dieser Verbindungskontrolle wurde das Plugin [Xam.Plugin.Connectivity](#) verwendet. Weitere

Informationen zu diesem Plugin finden sich unter:

<https://github.com/jamesmontemagno/ConnectivityPlugin>

[Jonas Grams]

### 5.1.2 View/ViewModel



[Niklas Klatt]

#### 5.1.2.1 MVVM

Die gesamte Software ist nach dem MVVM Pattern entwickelt. MVVM stellt eine Abwandlung des MVC dar. Der Unterschied findet sich im sogenannten "ViewModel".



Das ViewModel enthält die Logik der View und bereitet Daten aus dem Modell speziell für die View auf.

Außerdem werden in Xamarin View und ViewModel mit “Data Bindings” verbunden. Ändert sich etwas in der View, wird das entsprechende Binding im ViewModel geändert und umgekehrt. Die genaue Funktion des MVVM in Xamarin und der Data Bindings kann unter [Xamarin Data Bindings to MVVM](#) nachgelesen werden.

Wie Data Bindings in diesem Projekt umgesetzt sind, wird im nächsten Abschnitt näher beschrieben.

[Niklas Klatt]

#### 5.1.2.2 Data Bindings

Data Bindings werden genutzt um die Eigenschaften von zwei Objekten zu verbinden, sodass Änderungen an einem Objekt, auch das andere Objekt entsprechend ändern. Man kann Data Bindings im Code definieren, oder auch über die XAML-Dateien.

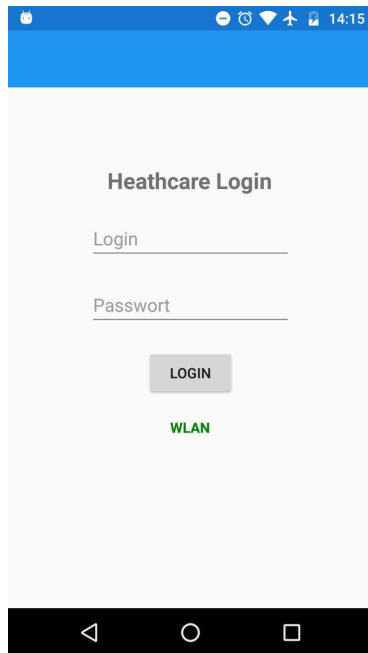
Die beiden Objekte werden als Source und als Target bezeichnet. Um ein Binding im Code durchzuführen, muss man folgende Schritte ausführen. Im Target Objekt muss der BindingContext das Source Objekt zugewiesen werden und die Methode SetBinding muss im Target Objekt aufgerufen werden, um eine Eigenschaft vom Target Objekt mit einer Eigenschaft vom Source Objekt zu verbinden. Genauer findet man hier: <https://developer.xamarin.com/api/type/Xamarin.Forms.BindableObject/>

Data Bindings werden genutzt um die View eines Programms mit dem darunterliegenden Model zu verbinden, unter Anwendung des MVVM - Patterns.

[Malte Grebe]

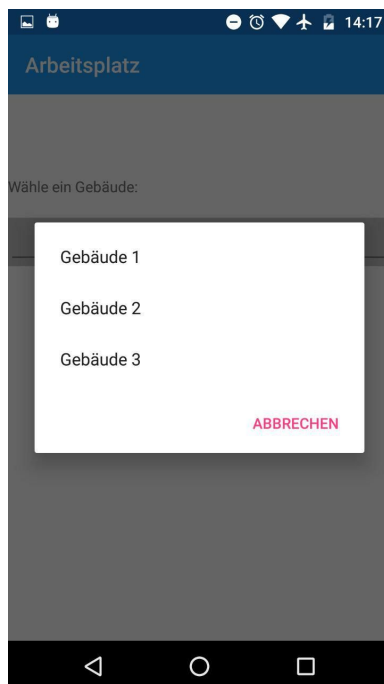
### 5.1.2.3 Klassenbeschreibung View

#### LoginPage



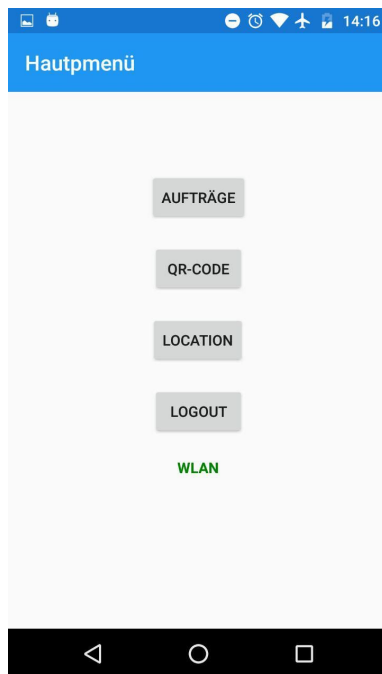
Stellt den Anmeldebildschirm dar.

#### ChangeWorklocation



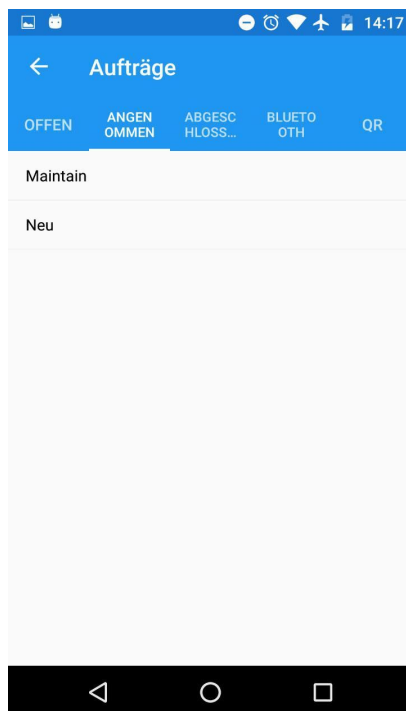
Stellt den Bildschirm zum ändern des Arbeitsplatzes dar.

## MainMenu



Stellt das Hauptmenü dar von dem man die verschiedenen Funktionen der Applikation ansteuern kann.

## TasksPage



Zeigt die verschieden gefilterten Auftragsseiten an.

## TaskTab

Zeigt die Tabs mit Aufträgen und unterschiedlichen Status an

## BluetoothTaskPage

Zeigt den Tab der *TaskTab*-Page an, wo die Bluetooth-Tasks aufgelistet werden.

## QRTaskPage

Zeigt den Tab der *TaskTab*-Page an, wo die QR-Tasks aufgelistet werden.

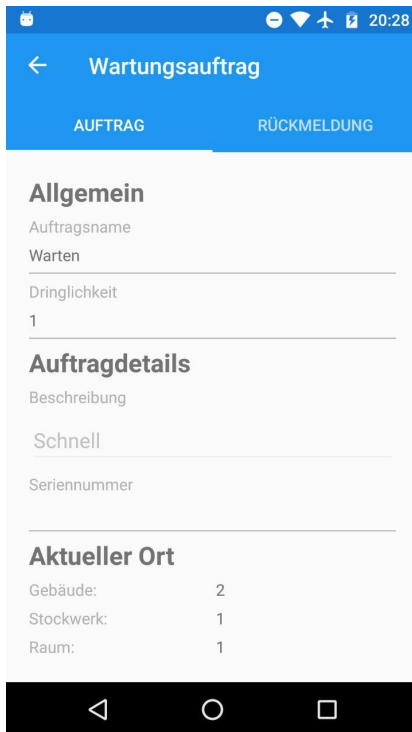
## TaskPage



The screenshot shows a mobile application interface for task management. At the top is a blue header bar with a back arrow and the title 'Auftrag'. Below this is a form with three sections: 'Allgemein' (General), 'Auftragdetails' (Task details), and 'Aktueller Ort' (Current location). The 'Allgemein' section contains fields for 'Auftragsname' (CleanMe) and 'Dringlichkeit' (1). The 'Auftragdetails' section contains a 'Beschreibung' field with the text 'eine wichtige Aufgabe'. The 'Aktueller Ort' section contains fields for 'Gebäude' (1), 'Stockwerk' (2), and 'Raum' (1). At the bottom of the form are two buttons: 'ANNEHMEN' and 'ABSCHLIESSEN'. The entire form is set against a light gray background with white text and lines.

Zeigt die Details eines Auftrags an

## MaintenanceDetailPage



Zeigt eine spezielle Ansicht für einen Wartungsauftrag an. Diese View ist eine *TabPage* und enthält im ersten Tab die normale *TaskPage* und im zweiten Tab die *MaintenanceFeedbackpage*.

## MaintenanceFeedbackPage



Eine View, die dem Nutzer ermöglicht eine Rückmeldung über den abgeschlossenen Auftrag zu geben.

### ScanPage



Öffnet die Kamera um einen QR-Code zu scannen.

[Malte Grebe, Niklas Klatt]

### 5.1.2.3 Klassenbeschreibung ViewModel

#### LoginViewModel

Füllt die View mit Daten über Databindings. Kommuniziert mit AppCore, um den Login zu synchronisieren.

[Malte Grebe]

#### ChangeWorklocationViewModel

Füllt die View mit Daten über Databindings. Kommuniziert mit AppCore, um die aktuellste Liste mit Gebäuden zu erhalten. Dient zum setzen der Gebäudenummer im Appcore und zum aktualisieren der lokalen Datenbank.

[Malte Grebe]

### MainMenuViewModel

Dient zur Navigation für die verschiedenen Funktionen der Applikation. Um sich alle Aufträge anzeigen zu lassen. Einen QR-Code zu scannen oder den Arbeitsplatz zu wechseln.

[Malte Grebe]

### TasksPageViewModel

Enthält die fünf Childpages für die TabbedPage: drei TaskTabPage, die BluetoothTaskPage und die QRTaskPage.

[Malte Grebe]

### BluetoothTaskPageViewModel

Füllt die View mit Daten über Databindings. Kommuniziert mit AppCore, um die aktuellste Liste mit Aufträgen zu erhalten und aktualisiert die View. Setzt bei Auswahl eines Auftrags diesen im AppCore, um die DetailAnsicht zu erzeugen.

[Malte Grebe]

### ORTaskPageViewModel

Füllt die View mit Daten über Databindings. Kommuniziert mit AppCore, um die aktuellste Liste mit Aufträgen zu erhalten und aktualisiert die View. Setzt bei Auswahl eines Auftrags diesen im AppCore, um die DetailAnsicht zu erzeugen.

[Malte Grebe]

### TaskTabViewModel

Unterscheidet die Aufträge aus der lokalen Datenbank, je nach Status “Offen”, “Angenommen” und “Abgeschlossen” und füllt die View mit Daten über Databindings. Kommuniziert mit AppCore, um die aktuellste Liste mit Aufträgen zu erhalten und aktualisiert die View. Setzt bei Auswahl eines Auftrags diesen im AppCore, um die DetailAnsicht zu erzeugen.

[Malte Grebe]

### TaskPageViewModel

Das ViewModel für die *TaskPage*. Enthält alle Data Bindings für die View. Außerdem ist hier die Logik implementiert, welche die Sichtbarkeiten der UI Elemente entsprechend den Aufträgen anpasst.

### MaintenanceFeedbackViewModel

Das ViewModel für die *MaintenanceFeedbackPage*. Enthält außerdem die Logik, womit auf Basis der *Subtasks* UI Elemente in eine ListView generiert werden.

## **5.2 /Healthcare.Droid**

### **5.2.1 Helper**

#### Notifier

Enthält die Plattform spezifische Implementierung für die Notifications wenn ein Auftrag in der Nähe ist.

[Sheraz Azad]

### **5.2.2 Service**

#### MonitorNotifier

Enthält die Implementierung für die Notifications für das Monitoring.

[Sheraz Azad]

#### RangingNotifier

Enthält die Implementierung für die Notifications für das Ranging.

[Sheraz Azad]

#### BeaconService

Enthält die Plattform spezifische Implementierung für das Beacon Service. Diese Klasse implementiert das Interface IBeaconService und enthält weitere Methoden um die Beacon Suche zu verfeinern. Außerdem können hier verschiedene Einstellungen vorgenommen werden, wie z.B. welche Beacon Standards gefunden werden sollen und in welchem Zeitabstand aktiv und inaktiv nach Beacons gesucht werden soll. Für die Beacon Suche wurde die AltBeacon Library verwendet.

[Sheraz Azad]



### 5.2.3 Database

#### SQLiteAndroid

Dies ist die Android-Spezifische Implementation. Hierüber kann der Pfad zum Speicherort der App abgerufen werden um zu überprüfen ob es bereits eine vorhandene SQLite-Datenbank auf dem Handy gibt.

[Daniel Dzimitrowicz]

## 5.3 /Healthcare.iOs

### 5.3.1 Service

#### BeaconService

Enthält die Plattform spezifische Implementierung für das Beacon Service. Diese Klasse implementiert das Interface IBeaconService und enthält weitere Methoden um die Beacon Suche zu verfeinern. Außerdem können hier verschiedene Einstellungen vorgenommen werden, wie z.B. welche Beacon Standards gefunden werden sollen und in welchem Zeitabstand aktiv und inaktiv nach Beacons gesucht werden soll. Für die Beacon Suche wurde die Estimote SDK Library verwendet.

[Sheraz Azad]

### 5.3.2 Database

#### SQLiteiOS

Dies ist die iOS-Spezifische Implementation. Hierüber kann der Pfad zum Speicherort der App abgerufen werden um zu überprüfen ob es bereits eine vorhandene SQLite-Datenbank auf dem Handy gibt.

[Daniel Dzimitrowicz]

## 6 Plug-Ins

Im folgenden werden alle Plug-Ins die für das Projekt genutzt worden sind aufgelistet, wobei die wichtigen Pakete markiert sind.

### /Healthcare

Plug-In	Autor	Version	Beschreibung
Behaviors.Forms	David Britch	1.1.0	Um die Verhaltensweisen und Aktionen in Xamarin Forms zu beeinflussen. Verringert C# Code und welches kompakt in XAML abgehandelt wird.
Microsoft.NETCore.Platforms	Microsoft	1.1.0	Diese Plug-Ins werden benötigt um Pakete über den Nuget Paket Manager oder die Paket Konsole installieren zu können.
Microsoft.NETCore.Targets	Microsoft	1.1.0	
NETStandardLibrary	Microsoft	1.6.1	
<b>Newtonsoft.Json</b>	James Newton King	10.0.3	Ein JSON Framework für .Net Projekte.
<b>sqlite-net-pcl</b>	Frank A Kruger	1.3.3	SQLite Library für .Net Projekte.
<b>SQLitePCLRaw.bundle_green</b>	Eric Sink	1.1.6	
<b>SQLitePCLRaw.core</b>	Eric Sink	1.1.6	
System.Collections	Microsoft	4.3.0	Stellt generische Klassen bereit, welche im Concurrent Paket außerdem noch Threadsicher sind.
System.Collections.Concurrent	Microsoft	4.3.0	
System.Diagnostics...	Microsoft	4.3.0	Stellt Klassen zum Diagnostizieren des Projektes bereit.
System.IO...	Microsoft	4.3.0	Stellt Klassen für File Handling in Xamarin Forms bereit.
System.Linq...	Microsoft	4.3.0	Stellt Klassen und Interfaces bereit, welche Language-Integrated Queries unterstützen.
System.Runtime...	Microsoft	4.3.0	Enthält Standard Klassen und Interfaces.

[Sheraz Azad]

Plug-In	Autor	Version	Beschreibung
System.Threading...	Microsoft	4.3.0	Stellt Klassen und Interfaces für Threads bereit.
System.Xml...	Microsoft	4.3.0	Stellt Klassen und Interfaces für die XML Notationen bereit.
<b>Xam.Plugin.Connectivity</b>	James Montemagno	2.3.0	Paket zum Arbeiten mit dem WLAN.
<b>Xam.Plugin.Media</b>	James Montemagno	2.6.2	Ein Framework um Cross Platform Bilder aufnehmen zu können.
<b>Xam.Plugins.Vibrate</b>	James Montemagno	2.0.0	Ein Framework um Cross Platform die Vibrationsfunktionen ansprechen zu können.
<b>Xamarin.Forms</b>	Xamarin Inc.	2.3.4.247	Paket um Native UI's für Android und iOS zu implementieren.
<b>ZXing.Net.Mobile...</b>	Redth	2.2.9	OpenSource Paket um Barcodes zu scannen und verarbeiten.

[Sheraz Azad]

### /Healthcare.Droid

Alle aufgelisteten Plug-Ins müssen installiert werden um die Software fehlerfrei zu starten.

Plug-In	Autor	Version	Beschreibung
AndroidAltBeacon Library	AltBeacon.org	2.7.0	Android Paket um mit BLE Beacons zu interagieren.
Newtonsoft.Json	James Newton King	10.0.3	Ein JSON Framework für .Net Projekte.
FastAndroidCamera	James Athey	2.0.0	Framework um die Android Kamera anzusprechen.
Xamarin.Android.Support.v4	Xamarin Inc	25.3.1	v4 Android Support Library C# bindings für Xamarin
Xamarin.Android.Support.v7...	Xamarin Inc	25.3.1	v7 Android Support Library C# bindings für Xamarin
ZXing.Net.Mobile...	Redth	2.2.9	OpenSource Paket um Barcodes zu scannen und verarbeiten.

[Sheraz Azad]

## /Healthcare/iOS

Alle aufgelisteten Plug-Ins müssen installiert werden um die Software fehlerfrei zu starten.

Plug-In	Autor	Version	Beschreibung
Newtonsoft.Json	James Newton King	10.0.3	Ein JSON Framework für .Net Projekte.
Xamarin.Estiomote. iOS	Xamarin	4.7.3.2	Estimote SDK für iOS um aktive nach Bluetooth Beacons zu scannen.
ZXing.Net.Mobile	Redth	2.2.9	OpenSource Paket um Barcodes zu scannen und verarbeiten.

[Sheraz Azad]

## 7 Software Funktionen

### 7.1 Bluetooth (Beacon) Suche

Da die Bluetooth Funktion nicht anhand von Unit Tests getestet werden kann, wird im folgenden ein Szenario einer Bluetooth mit Zeitangaben des System angezeigt. Die Systemzeit wurde anhand von *System.DateTime.Now* auf der Anwendungsausgabe ausgegeben. Im folgenden Szenario meldet sich ein User mit seinen Daten an und wählt eine Location, in diesem Moment wird die Bluetooth Suche nach Beacons gestartet, welches man am folgenden Screenshot der Anwendungsaufgabe sehen kann.

```

Anwendungsausgabe

+++++
Starting the Beacon Service
+++++

[art] JNI RegisterNativeMethods: attempt to register 0 native methods for md58c37b54e6aaad0d483cedbb094f5aca7.BeaconService
[BeaconParser] Parsing beacon layout: m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24
[BeaconManager] Cannot contact service to set scan periods
[Choreographer] Skipped 65 frames! The application may be doing too much work on its main thread.
[BluetoothCrashResolver] Can't read macs from BluetoothCrashResolverState.txt
[SplitWindow] update focus...
[BluetoothAdapter] STATE_ON
[BluetoothLeScanner] onClientRegistered() - status=0 clientIf=5
[BluetoothAdapter] STATE_ON
Thread started: #13
[BluetoothAdapter] STATE_ON
[BluetoothLeScanner] onClientRegistered() - status=0 clientIf=5

+++++
[Mono] Assembly Ref addrf HealthcareApp.Droid[0x7f7501e300] -> System[0x7f5515b600]: 17
Ranging for Beacons at 01.07.2017 15:41:08
+++++

```

Danach wird im Plattform spezifischen Code die Suche nach den Beacons gestartet, welches anhand der “Ranging for Beacons ...” signalisiert wird. Befindet sich kein Beacon in der Nähe wird der Prozess der Suche erneut ausgeführt, bis ein Beacon in der Nähe ist.

```

Anwendungsausgabe

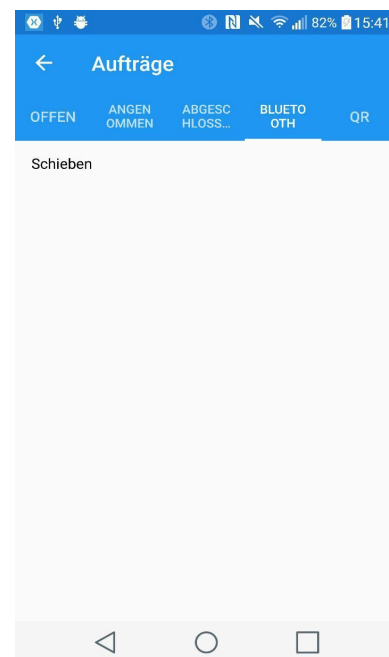
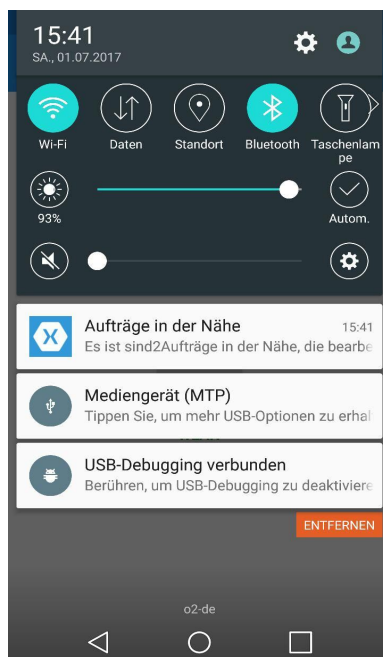
[BluetoothAdapter] STATE_ON
+++++
Ranging for Beacons at 01.07.2017 15:41:20
+++++

Thread finished: #15

+++++
Converting the list of Identifiert to a list of String Id's
+++++

+++++
Send list to the View
+++++
  
```

In diesem Fall wird die UUID des Beacon in ein String umgewandelt, um diese weiterverarbeiten zu können. Diese Liste wird dann im weiteren Verlauf des Programms an die View geschickt. Falls ein passender Auftrag zur ID in der Nähe ist wird dieser in der View dargestellt, das Handy vibriert und erhält eine Notification um den User auf diesen Auftrag hinzuweisen. Meldet sich der User aus der App ab, wird die Bluetooth Suche ebenfalls gestoppt.



```

+++++
Updating Bluetooth View at 01.07.2017 15:41:20
+++++

Thread finished: <Thread Pool> #2
Thread started: <Thread Pool> #16
[ViewRootImpl] ViewRoot's Touch Event : ACTION_DOWN
[ViewRootImpl] ViewRoot's Touch Event : ACTION_UP
[ViewRootImpl] ViewRoot's Touch Event : ACTION_DOWN

+++++
User logged out at 01.07.2017 15:41:29
Stopping Bluetooth
+++++
  
```

[Sheraz Azad]

## 7.2 QR Code Scanner

Als Alternative zur Bluetooth Suche, da die Beacons nicht an jedes medizinische Gerät angebracht werden dürfen, wurde ein QR Code Scanner implementiert. Beim lesen eines QR Codes wird die UUID, der Major und der Minor übermittelt, anhand dessen die dazugehörigen Aufträge aus der Datenbank geholt werden und dem User angezeigt werden. Existiert kein Auftrag zum gescannten Objekt gibt es eine Fehlermeldung, außerdem gibt es noch eine Fehlermeldung wenn der User nicht autorisiert ist das Objekt zu scannen und Aufträge zu dem Objekt zu sehen. Ist der User dazu berechtigt und das gescannte Objekt hat mindestens einen Auftrag, wird eine Ansicht mit den zugehörigen Aufträgen geöffnet. Die Aufträge können dann wie gewohnt in der Detailansicht geöffnet werden.

[Sheraz Azad]