RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

BACHELOR/MASTER

ARBEIT

Steht noch nicht ganz fest

Autor	Lukas Schmidt
Studiengang	Medizinische Informatik Universtität Heidelberg / Hochschule Heilbronn
Matrikelnummer	182706
Abgabe	2. Dezember 2015
Referent	Prof. Dr.-Ing. Gerrit Meixner
Korreferent	Prof. Dr. Martin Haag

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

Git Werkzeug. 9

SDK A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all free open standards. 14

1. Einleitung

1.1. Motivation

Durch steigenden Anzahl an Medikamenten, die Patienten über den Tag nehmen müssen, kann dies zu einer großen Mentalen Aufgabe für den Patienten werden. Durch Hilfsmittel wie Medikationsplänen oder nach Zeit vorsortierten Medikamente kann die Planung der Einnahme erleichtert werden.

Zur Erleichterung der Patienten soll der Medikationsplan nun per Smartwatch einsehbar gemacht werden und somit dem Patient eine Interaktion mit dem Plan ermöglichen. Daraus bieten sich auch Vorteile für den behandelten Arzt, der Einblick in die Einnahmegewohnheiten seines Patienten bekommt. Initiale Ideen stammen auf dem PITA-Praktikum an der HS-Heilbronn. Hier wird auch in enger Zusammenarbeit mit dieser Arbeit ein Komponente für Ärzte entwickelt, die es ihnen ermöglicht die Medikationen der Patienten zu plegen, zu überwachen und auszuwerten.

1.2. Zielsezung

Die Ziele der Arbeit sind wie folged.

Entwickeln eines interaktives Prototypen, welcher auf der Apple Watch ausgeführt werden kann. Diese Prototyp soll dann mit gegeigten Personen, die zur passenden Zielgruppe gehören, evaluiert werden. Die Ergebnisse der Evaluierung sollen im Rahmen der technischen Möglichkeiten umgesetzt werden. Zum Schluss soll noch die Machbarkeit überprüft werden, die Anwendung an das Backend-System vom PITA-Praktuikum anzuschließen Anbinden des Prototypen an das Backend-System vom PITA-Praktikum

1.3. Aufbau der Arbeit

In Kapitel 2 wird auf das Forschungsumfeld der Arbeit Bezug genommen. Weiter werden technologische Grundlagen beschrieben, wie Apple‘s Swift Programmiersprache beschrieben. Des Weiteren werden Grundlagen in der Evaluierung der Gebrauchstauglichkeit und von Wearables beschrieben Im drittel Kapitel werden Anforderung beschrieben,

die teilweise aus vorherigem PITA-Praktikum stammen. Weiter werden Anforderungen geschildert, die sich aus den Möglichkeiten der Apple Watch ergeben. Abschnitt 4 betrachtet die Kernpunkte der Implementierung. Weiter wird hier der Prototyp beschrieben In Kapitel 5 wird die Planung und Durchführung der Evaluierung beschrieben. Die Ergebnisse der Evaluierung an der Zielgruppe sind hier zu finden. In der Diskussion ,in Kapitel 6, wird aufgezeigt, welche Probleme während der Arbeit entstanden sind. Es wird gezeigt wie der Prototyp im Gesamtbild einzuordnen ist und es wird ein Ausblick gegeben, welche Ziele mit dem System weiter verfolgt werden können

2. Stand der Wissenschaft

2.1. Einleitende Worte

Während das Forschungsfeld der ”Medication Adherence”, also as Einhalten des Medikamentenplans, schon jahrzehntelang erforscht wurde, ist diese Arbeit ist Teil eines noch sehr jungen Forschungsgebietes. Smartwatches existieren noch nicht lange im Consumer Bereich. Im folgenden ist eine Zusammenfassungen der relevanten Forschungen zu finden. Apple‘s Programmiersprache Swift, sowie die Technologie der Wearables wird beschrieben.

2.2. Arbeiten im Forschungsumfeld

Leider gibt es kaum Arbeiten, welche sich mit dem Thema Smartwatch und ”Medication Adherence”beschäftigten. Dies ist auf das noch junge Forschungsfeld zurückzuführen. Sailer Arbeit [8] bietet eine Einstieg für diese Arbeit. Hier wurde auch die Thematik des PITA Praktikum abgeleitet, deren Erkenntnisse hier fortgeführt werden.

Weiter gibt es sehr spannende Forschungen im Bereich der Smartwatchanwednung, die in Fortschreitender Entwicklung auch im Bereich ”Medication Adherence”vorstellbar sind. Mit Ambient Assisted Living, also der technischen Unterstützung älterer oder eingeschränkter Personen im Haushalt, beschäftigt sich die Arbeit ”Non-obstructive Room-level Locating System in Home Environments Using Activity Fingerprints from Smartwatch” [5]. Wenn man nicht nur die Zeit, sondern auch den Ort oder das Zimmer für eine Medikamenteneinnahme definieren kann, könnte der Patienten das Medikament direkt einnehmen und die Gefahr des Vergessens verringert sich. Ebenso gliedert sich die Arbeit von Laput in den Bereich ein, der einen Kontextgewinn zur Folge hat [4]. Die Arbeit beschreibt das Erkennen von Gegenständen an ihrem Elektromagnetischem Feld. Der Kontext Gewinn, zu Erkennen welche Gegenstände der Träger der Uhr berührt, könnte zur Fehler bei Medikamenteneinnahme verhindern.

Orientiert man sich an Arbeiten, deren Ziel die smartphonegestütze ”Medication Adherenceßum Ziel hatten findet man unter anderem die aktuelle nationale Umfrage [3] aus den USA, bei der 1604 Smartphone Nutzer zur ihrer Nutzung von Gesundheitsanwednungen befragt wurden. Mehr dazu wird im Kapitel 3 zur Problemstellung aufgegriffen

Apple Keynote Produkt Vorstellung

2.3. Apple‘s Programmiersprache Swift

Swift wurde im Juni 2014 von Apple vorgestellt und genießt seit dem steigendes Interesse. Im Juni 2015 wurde Version 2.0 veröffentlicht [1]. Mit Version 2.0 wurde ebenfalls der Plan vorgestellt, Swift Open Source zu machen und somit auch anderen Plattformen die Entwicklung mit Swift zu ermöglichen [1].

2.3.1. Objective-C und Swift

In den frühen 80er Jahren entwickelte Brad Cox die Sprache Objective-C [2]. Weiter führt Dalrymple aus, dass die Sprache die Vorteile einer schnellen C-Sprach mit den Vorteilen der objektorientierten Sprache SmallTalk verbinden sollte. Die Firma NextSTEP nutze Objectiv-C und als NEXTStep von Apple aufgekauft wurde, integrierte Apple Objectiv-C und ermöglichte Mac-Entwicklern die Nutzung.

Als Apple nun 2008 seine iOS Plattform öffnete und Entwickler eigene Anwendungen für das System schreiben konnten, bekam Objective-C neue Aufmerksamkeit. Viele Entwickler sahen Objective-C als ein Überbleibsel alter Zeiten und waren der Sprache negativ eingestellt. Apple stand nun unter Zugzwang um seine Plattform, mit der große wirtschaftliche Interessen Verbunden sind, für Entwickler attraktiv zu halten [9]. Da jedoch alle highlevel APIs in Objectiv-C vorhanden sind, ist es technisch nicht Möglich auf eine bekannte Sprache wie Java für iOS und Mac Entwicklung umzusteigen. Man entschied sich für eine Neuentwicklung, mit Hinblick auf neue Programmierparadigmen und sehr guter Kompatibilität zu alten Objectiv-C APIs [9].

2.3.2. Überblick der Neuerung

las

1. Type Inference
2. Closures - Functions as First Class Types
3. Generics
4. Optional Types

2.3.3. Type Inference

Bei Type Inference erkennt der Compiler, welcher Typ in eine Variable instanziert wurde. Es ist nicht nötig für die Vairable eine Typendefinition zu definieren [1]. Hierdurch wird der Quellcode leichter zu lesen.

Listing 2.1: Beispiel zu Type Inference in Swift label

```
//Variable with type Definition
let medication: Medication = Medication()

//Variable with infered type
let medication = Medication()
```

2.3.4. Closures - Functions as First Class Types

Während bei strikt objektorientierten Sprachen nur Objekte und primitive Datentypen existieren, gibt es Sprachen, bei denen Funktionen als Typen existieren. Diese Funktionen können auch als Refernz in eine Variablen gespeichert werden. Folgende Code Beispiele sollen dies veranschaulichen. Wir nutzen hierfür einen Asynchrones Netzwerk-Request [1].

Listing 2.2: Java Beispiel zu First Class Objects

```
public interface NetworkRequestResponseHandler
    public handleNetworkResponse(Error error, Response response);
}

public interface NetworkRequest
    public performNetworkRequest(NetworkRequestResponseHandler
        handler);
}

public class ExampleClass implements NetworkRequestResponseHandler {
    public handleNetworkResponse(Error error, Response response) {
        //Use Response
    }

    public static main(String[] args) {
       NetworkRequestImpl().performNetworkRequest(this)
    }
}
```

In Beispiel 2.2 wird Java als Repräsentant für eine strikt objektorientierte Sprache verwendet. Hier wird ein Interface definiert, welches der Nutzer des Netzwerk-Requests implementieren muss, um den Request zu empfangen. Beim Aufrufen des Request, muss nun der Aufrufer als Referenz übergeben werden, damit bei Abschließen des Request, der Aufrufer benachrichtigt werden kann (handleNetworkResponse).

Listing 2.3: Swift Beispiel zu First Class Objects

```
protocol NetworkRequest {
    func performNetworkRequest(handleNetworkRequest:(Error, Response)
       ->())
}

class ExampleClass {
    func main() {
        //Store a function in a variable
        let handleRequest = {(error, response) in
            //Handle Response
        }
        NetworkRequestImpl.performNetworkRequest(handleRequest)
    }
}
```

Im Codebeispiel 2.3 benötigt es kein Interface für den Nutzer des Netzwerk-Requests. Es ist nun in Swift möglich eine Funktion zu definieren und diese gleichzeitig in einer lokale Variable zu speichern. Nun kann diese Funktion als Referenz zum Netzwerk-Requests übergeben werden. Die Funktion wird aufgerufen, wenn der Netzwerk-Requests beendet ist.

Generics oder Protocol Extension

2.3.5. Optional Types

Optional Types eröffnen neue Möglichkeiten beim Modellieren von Datenmodellen und erstellen von APIs. Es ist so möglich Argumente in einer Methode als Optional zu kennzeichnen und somit dem Nutzer der Methode zu erlauben eine null-Referenz zu übergeben. Ist kein Optional-Type gekennzeichnet, so verbietet der Compiler ein null-Referenz übergabe [1]. Im folgenden Beispiel wird eine Medikation in Java ohne Optional Types und in Swift mit Optional Types modelliert.

Listing 2.4: Java Beispiel mit fehlenden Optional Types

```
public class Medication {
    Date creationDate = new Date();
    Date executionDate;
}

Medication medication = new Medication()
medication.creationDate // null?
medication.executionDate // null?
```

Im Codebeispiel 2.4 ist zu erkennen, dass zur Compilezeit keine Aussage über den Zustand der Instanz-Variablen getroffen werden kann. Der Entwickler muss also aus dem logischen Kontext erkennen, welche Variablen eine null Referenz enthalten könnte. Dies kann zu Laufzeitfehlern führen.

Listing 2.5: Swift Beispiel mit Optional Types

```
class Medication {
    let creationDate = Date()
    var executionDate: Date?
}

let medication = Medication()

medication.creationDate // compiler promises to be not null
medication.executionDate // could be null
```

In 2.5 sind die Instanzvariablen nun mit Optional Types modeliert. Nun kann zur Compilezeit zugesichert werden, welche Variablen eine null Referenz enthalten können und welche Variablen sicher mit einem Wert belegt sind. Dies führt zu weniger Fehler während der Laufzeit.

Auch APIs können so modelliert werden. So darf ein Parameter nicht null sein, wenn er nicht als Optional definiert wurde. Dies macht eine API strikter und führt ebenfalls zu weniger Laufzeitfehlern, da Fehler schon zur Compilezeit erkannt werden.

2.4. Wearables

3. Analyse und Entwurf

In diesem Kapitel Ergebnisse der Analyse und des Entwurfes erläuteret. große Teile der Anforderungsanalyse stammen aus dem PITA Praktikum.

3.1. Entwicklungsmethodik

Da das Projekt nur durch eine Person durchgeführt wurde, kann man nicht von einem definierbaren Methodik sprechen. Es handelte sich um ein iteratives Vorgehen. Funktionen, die bei der Evaluation erarbeitet wurden, sind direkt in eine neue Version der Anforderungen integriert worden. Der Quellcode wurde mit Park Distance Control/Park Pilot (PDC) versioniert verwaltet

3.2. Anforderungsanalyse

Die Anforderungen wurden von eine Gerspräch mit Monika Pobiruchin am Anfang des Pita-Praktikum erhoben. Kernaussage dieser Anforderungen, dass das System von alten Menschen genutzt werden kann, um Medikamente regelmäßiger einzunehmen. Dies beinhaltet eine geringe Auseinandersetzung mit der Technik des Systems. Die Uhr soll möglichst autonom sein und nicht zwingen an ein Smartphone gekoppelt sein.

3.3. Usecases

Von dem Gespräch mit Monika Pobiruchin wurde eine Persona für die Zielgruppe abgeleitet. Diese Persona ist im Anhang zu finden. Mit Hilfe der Persona, wurden die Kern-Usacases abgeleitet. Die folgenden Usecase 1 bis 4 wurden im Praktikum erarbeitet und wurden von dort übernommen.

Usecase 1	**Der Patient wird an ein Medikament erinnert**
Primärer Akteur	Patient

Beschreibung	Ein Erinnerungs Pop-Up erscheint auf dem Display und es wird ein Signal/eine Vibration ausgelost. Das Pop-Up zeigt ein Abbild des Medikaments, dessen Namen und die Uhrzeit, zu der es eingenommen werden soll.
Vorbedingung	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen.
Ablauf	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display. Die Erinnerung beinhaltet Informationen zur Uhrzeit, Menge und Art der Medikation • Der Patient bestätigt, dass er das Medikament genommen hat • Gerät bestätigt visuell dass der Patient das Medikament als genommen markiert hat
Alternativablauf	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient wählt Option zum Verschieben der Medikation aus • Auf einem zusätzlichem Dialog kann er aus einer Auswahl eine Zeitdauer wählen, um das die Medikation verschoben wird
Alternativablauf 2	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient reagiert nicht auf die Erinnerung • Die Erinnerung wird alle x Minuten wiederholt, solange der Patient nicht reagiert. • Das Fehlen einer Reaktion des Patientens innerhalb einer Zeit von x Minuten wird vermerkt
Ergebnis	Der Patient hat die Einnahme des Medikaments bestätigt und dieses auch eingenommen
Alternativergebnis 1	Der Patient hat die Erinnerung an die Medikamenteneinnahme verschoben
Alternativergebnis 2	Der Patient hat die Erinnerung an das Medikament ausgeschaltet

Usecase 2	**Der Patient wird an mehrere Medikamente erinnert**
Primärer Akteur	Patient
Beschreibung	Ein Erinnerungs Pop-Up erscheint auf dem Display und es wird ein Signal/eine Vibration ausgelöst. Das Pop-Up zeigt eine Liste der Medikamente, die eingenommen werden müssen.
Vorbedingung	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen.
Ablauf	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display. Die Erinnerung beinhaltet Informationen zur Uhrzeit, Menge und Art der Medikationen • Der Patient bestätigt, dass er die Medikamente alle genommen hat • Gerät bestätigt visuell, dass der Patient die Medikamente als genommen markiert hat
Alternativablauf	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient wählt Option zum Verschieben der Medikationen aus • Auf einem zusätzlichem Dialog kann er aus einer Auswahl eine Zeitdauer wählen, um das die Medikationen verschoben werden
Alternativablauf 2	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient wählt ein Medikament von der Liste aus • Patient befindet sich nun im Usecase 1

Alternativablauf 3	• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display • Der Patient reagiert nicht auf die Erinnerung • Die Erinnerung wird alle x Minuten wiederholt, solange der Patient nicht reagiert. • Das Fehlen einer Reaktion des Patientens innerhalb einer Zeit von x Minuten wird vermerkt
Ergebnis	Der Patient hat die Einnahme der MEdikamente bestätigt und dieses auch eingenommen
Alternativergebnis 1	Der Patient hat die Erinnerung an die Medikamenteneinnahme verschoben
Alternativergebnis 2	Der Patient hat die Erinnerung an das Medikament ausgeschaltet
Alternativergebnis 3	Der Patient hat bestimmte Medikamente ausgewählt und mit dem weiterführenden Usecase 1 bearbeitet

Usecase 3	**Einzelne Einnahmebestätigung zurücknehmen**
Primärer Akteur	Patient
Vorbedingung	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen. Eine Medikation wurde als genommen makiert
Ablauf	• Das System zeigt das genommene Medikament an • Der Benutzer drückt auf den Button mit der Aufschrift “Rücknahme” • Das System wechselt zur Darstellung eines einzelnen Medikaments, beschrieben im Use-Case 1
Ergebnis	Die Einnahmebestätigung ist zurückgenommen. Die Erinnerung ist erneut zu bestätigen oder zu verschieben.

Usecase 4	**Mehrere Einnahmebestätigungen zurücknehmen**
Primärer Akteur	Patient
Vorbedingung	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen. Mehrere Medikationen, welche zur gleichen Zeit genommen wurden, wurde als genommen makiert
Ablauf	• Das System zeigt die genommenen Medikamente an • Der Benutzer drückt auf den Button mit der Aufschrift “Rücknahme” • Das System wechselt zur Darstellung mehrere Medikamente, beschrieben im UseCase 2
Ergebnis	Die Einnahmebestätigung ist zurückgenommen. Die Erinnerung ist erneut zu bestätigen oder zu verschieben.

Diese 4 Usecases sind auch Grundlage für dies Arbeit. Wie die Usecases umgesetzt sind wird in Kapitel 4 beschrieben. In Kapitel 5 finden sich überarbeitet Usecases, die Verbesserungen enthalten, welche aus der Evaluation mit der Zielgruppe hervorgehen.

3.4. Apple Watch

Eine Andorderung, welche sich aus dem Kontext dieser Arbeit entnehmen lässt, ist die Nutzung der Apple Watch als Zielplatform. Die Apple Watch wurde im September 2014 vorgestellt und staret im April 2015 mit dem Verkauf.

3.4.1. Hardware

Die Apple Watch existiert in zwei Versionen. Einen Uhr mit 38mm (272x340) und eine mit 42mm (312x390) großem Display. Die bietet einen 8GB großen internen Speicher. Mit einer Akkulaufzeit von 18h unter durchschnittlicher Nutzung, hält die Uhr einen Tag durch [6].

3.4.2. Software

Mit erscheinen der Uhr wurde auch das Betriebsystem in Version 1 ausgeliefert und dazu das Extensible Markup Language (XML) Names WatchKit. Dies erlaubte es Entwicklern Anwendungen zu Entwickeln, welche auf dem verbunden iPhone ausgeführt wurden. Diese führte zu schlechte Performance der Anwendungen und zu vollen Abhängigkeit zum iPhone.

Im Juni veröffentlichte Apple die erste Vorabversion von watchOS 2.0, welches später im September 2015 für Endnutzer bereit gestellt wurde. watchOS biete mehr Unabhängigkeit für Anwendungen. Die Andwendugen laufen direkt auf der Uhr.

3.4.3. Schniststellen

Bluetooth 4.0 und Wi-Fi 802.11b/g sind die Netzwerkschnistellen der Apple Watch. Dazu kommt noch ein NFC Chip, der jedoch nicht über eine API nutzbar ist und vorerst nur für Apple Pay, dem Apple eigenen Bezahldienst, vorgesehen ist [7].

3.4.4. Sensoren

Die Apple Watch besitzt einen Beschleunigungssensor und Gyroskop welche genaue Bewegungsdaten liefern. Ebenso ist ein optischer Herzschagsensor verbaut, der an der Unterseite der Uhr auf der Haut anliegt. Ein Mikrofon, welches für Spracheingabe genutzt werden kann ist auch vorhanden.

3.4.5. Eingabe Interfaces

Neben eines normalen Touchescreen führte Apple in der Uhr auch eine Eingabeart Names Force Touch ein. Diese Technologie erlaubt es der Uhr zu erkennen, wie fest der Nutzer auf das Display drückt. Dies ermöglicht eine Neue Art der Eingabe. Besonders bei einer kleinen Eingabefläche, wie die der Apple Watch, ist eine neue Interaktionsdimesion interessant, da es eine differenziertere Interaktion erlaubt. Leider ist eine mögliche Interaktion mit Force Touch optisch nicht zu erkennen, was eine klare Nutzerführung schwer macht.

Von Analogen Uhren ist die Krone, also das Rad an der Seite einer Uhr, an dem die Uhr eingestellt oder aufgezogen werden kann bekannt. An der Apple Watch ist die Krone auch zu finden. Apple nennt sie ”Digital Crown”, also digitale Krone. Sie befindet sich ebenfalls an der Seite der Uhr. Die Krone ist drehbar und lässt sich ebenfalls als Druckknopf nutzen. Sie dient zum scrollen von Inhalten, sowie präzisen Auswählen von Elementen aus einer Auswahlliste. Durch nutzen der Krone wird der Bildschirm nicht durch einen Finger verdeckt, was bei einem kleinen Display von Vorteil ist.

3.4.6. Armband

Apple biete 6 verschiede Armbänder für die Apple Watch an (Stand Nov. 2015). Zusätzlich ist es möglich, Armbänder von Drittherstellern zu erwerben.

Das in der günstigsten Version mitgelieferter Version (Sportarmband) verfügt über einen sehr komplizierten Verschlussmechanismus und ist deswegen weniger für Menschen geeignet, die über schwache sensomotorische Fähigkeiten verfügen. Es gibt auch Armbänder, die einen magnetischen Verschluss bieten, diese biete eine einfachen Handhabung, sind jedoch 3 mal so teuer, wie das Sportarmband.

An der Verbindung zwischen Armband und Uhr ist eine nicht weiter spezifizierter Wartungsport verbaut. Dieser Anschluss könnte in Zukunft Armbänder ermöglichen, welche Informationen aus dem Armband an die Uhr weiterleiten.

4. Umsetzung

5. Zusammenfassung, Evaluation und Ausblick

Literaturverzeichnis

[1] *The Swift Programming Language*. Apple, Inc., 2014.

[2] DALRYMPLE, M. Learn objective-c on the mac. 1–20.

[3] KREBS P, D. D. Health app use among us mobile phone owners: A national survey. *MIR mHealth uHealth* (2015).

[4] LAPUT, G., YANG, C., XIAO, R., SAMPLE, A., AND HARRISON, C. Em-sense: Touch recognition of uninstrumented, electrical and electromechanical objects. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (New York, NY, USA, 2015), UIST '15, ACM, pp. 157–166.

[5] LEE, S., KIM, Y., AHN, D., HA, R., LEE, K., AND CHA, H. Non-obstructive room-level locating system in home environments using activity fingerprints from smartwatch. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (New York, NY, USA, 2015), UbiComp '15, ACM, pp. 939–950.

[6] RICHES, G. Apple watch for developers.

[7] RITCHIE, R. Apple watch specs, 2015.

[8] SAILER, F., POBIRUCHIN, M., WIESNER, M., AND MEIXNER, G. An approach to improve medication adherence by smart watches, 2015.

[9] WELLS, G. The future of ios development: Evaluating the swift programming languag. Master's thesis, Claremont McKenna College, 2015.

A. Appendix

B. Dokumentation