



RUPRECHT-KARLS-  
UNIVERSITÄT  
HEIDELBERG



# **BACHELOR**

## **ARBEIT**

**System zur Erinnerung an Medikamenteneinnahme für  
die Apple Watch mit Blick auf Gebrauchstauglichkeit und  
Akzeptanz der Plattform**

---

Autor	Lukas Schmidt
Studiengang	Medizinische Informatik Universität Heidelberg / Hochschule Heilbronn
Matrikelnummer	182706
Abgabe	15. März 2016
Referent	Prof. Dr.-Ing. Gerrit Meixner
Korreferent	Prof. Dr. Martin Haag

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>iii</b>
<b>Tabellenverzeichnis</b>	<b>v</b>
<b>Glossar</b>	<b>vii</b>
<b>1. Einleitung</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Zielsetzung . . . . .	1
1.3. Aufbau der Arbeit . . . . .	1
<b>2. Stand der Wissenschaft</b>	<b>3</b>
2.1. Einleitende Worte . . . . .	3
2.2. Arbeiten im Forschungsumfeld . . . . .	3
2.3. Apple's Programmiersprache Swift . . . . .	4
2.3.1. Objective-C und Swift . . . . .	4
2.3.2. Überblick der Neuerung . . . . .	5
2.3.3. Type Inference . . . . .	5
2.3.4. Closures - Functions as First Class Types . . . . .	5
2.3.5. Generics . . . . .	6
2.3.6. Protocol Extension . . . . .	8
2.3.7. Optional Types . . . . .	9
2.4. Wearables . . . . .	10
<b>3. Analyse und Entwurf</b>	<b>13</b>
3.1. Entwicklungsmethodik . . . . .	13
3.2. Anforderungsanalyse . . . . .	13
3.3. Usecases . . . . .	13
3.4. Apple Watch . . . . .	17
3.4.1. Hardware . . . . .	17
3.4.2. Abhängigkeit zum iPhone . . . . .	18
3.4.3. Software . . . . .	18
3.4.4. Schniststellen . . . . .	19
3.4.5. Sensoren . . . . .	19
3.4.6. Eingabe Interfaces . . . . .	19
3.4.7. Armband . . . . .	20

3.4.8. Prototyping . . . . .	20
<b>4. Umsetzung</b>	<b>21</b>
4.1. Werkzeuge . . . . .	21
4.2. Benutzeroberfläche . . . . .	21
4.3. WatchConnectivity . . . . .	21
4.4. Notification Management . . . . .	24
4.5. Anwendung . . . . .	24
4.5.1. Notification . . . . .	24
4.5.2. Native Anwendung . . . . .	24
<b>5. Evaluation, Zusammenfassung und Ausblick</b>	<b>29</b>
5.1. Evaluation . . . . .	29
5.1.1. Zielgruppe . . . . .	29
5.1.2. Vorgehen . . . . .	29
5.1.3. Auswertung . . . . .	30
5.2. Durchführung der Evaluation . . . . .	30
5.2.1. Beschreibung der Zielgruppe . . . . .	30
5.2.2. Befragung der Patienten . . . . .	30
5.2.3. Auswertung . . . . .	30
5.2.4. Probleme tabellarisch . . . . .	31
5.3. Zusammenfassung . . . . .	31
5.4. Ausblicke . . . . .	33
<b>Literaturverzeichnis</b>	<b>35</b>
<b>A. Appendix</b>	<b>37</b>
A.1. Fragebogen . . . . .	37

# Abbildungsverzeichnis

4.1.	Interface Elemente zu Erstellen von Benutzeroberflächen . . . . .	22
4.2.	Interface Elemente mit Quellcode verknüpfen . . . . .	23
4.3.	Notification für Medikament . . . . .	25
4.4.	Native Anwendung: Interface zu wählen von eines Medikaments (oben links). Medikament als genommen bestätigen (oben rechts). Eigene Zeitdauer auswählen (unten links). Medikament ist in der Übersicht als verschoben markiert (unten rechts) . . . . .	26
4.5.	Native Anwendung: Interface zu verschieben von eines Medikaments (oben links). Zeitdauer für das Verschiebenwählen (oben rechts). Medikament ist als genommen markiert (unten links). Medikament ist in der Übersicht als genommen markiert(unten rechts) . . . . .	27



# Tabellenverzeichnis

5.1. Probleme und mögliche Lösungen des Smartwatch Prototypes . . . . .	32
---	----





# Glossar

**Git** Git ist eine verteilte Versionscontrolle [5]. 13

**PITA** Ein Praktikum im Rahmen des Bachelor Studienganges an der HS-Heilbronn zur Entwicklung von telemedizinischen Informationssystemen. 1, 3, 13

**Remote Server** Ein Computer, der über das Internet Anfragen verarbeitet und beantwortet. 24

**SDK** Eine Schnittstelle zum System, mit dem Drittanbieter Software für die Zielplattform des SDKs entwickeln können. 18



# 1. Einleitung

## 1.1. Motivation

Durch steigende Anzahl an Medikamenten, die Patienten über den Tag nehmen müssen, kann dies zu einer großen mentalen Aufgabe für den Patienten werden. Durch Hilfsmittel wie Medikationspläne oder nach Zeit vorsortierte 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 behandelnden Arzt, der Einblick in die Einnahmegewohnheiten seines Patienten bekommt. Die initiale Idee stammt vom Praktikum Informationssysteme/Telemedizinische Anwendungen an der HS-Heilbronn. An der HS-Heilbronn wird eine Komponente für Ärzte entwickelt, die es ihnen ermöglicht, die Medikationen der Patienten zu pflegen, zu überwachen und auszuwerten.

## 1.2. Zielsetzung

Die Ziele der Arbeit sind wie folgt.

Es wird ein interaktiver Prototyp entwickelt, welcher auf der Apple Watch ausgeführt werden kann. Dieser Prototyp soll dann mit geeigneten Probanden, 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-Praktikum anzuschließen.

## 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 und die Technologie von Wearables. Im dritten Kapitel werden Anforderungen aufgeführt, die teilweise aus dem 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 dargestellt. Die Ergebnisse der Evaluierung an der Zielgruppe sind hier ebenfalls 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" das Einhalten des Medikamentsplans, schon jahrzehntelang erforscht wurde, ist diese Arbeit Teil eines noch sehr jungen Forschungsgebietes. Smartwatches existieren noch nicht lange im Consumer Bereich. Im folgenden ist eine Zusammenfassung 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" beschäftigen. Dies ist auf das noch junge Forschungsfeld zurückzuführen. Sailer's Arbeit [19] bietet einen Einstieg für diese Arbeit. Hier wurde auch die Thematik für das PITA abgeleitet, deren Erkenntnisse hier fortgeführt werden.

Weiter gibt es sehr spannende Forschungen im Bereich der Smartwatchanwendung, 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" [14]. Die von dieser Arbeit abgeleitete Lösung, kann auch für rechtzeitige Medikamenteneinnahme genutzt werden. So könnte ein Medikationsalarm nur ausgelöst werden, wenn der Patient sich auch im richtigen Zimmer befindet. Durch den kürzeren Weg zum Medikament sinkt die Gefahr, auf der Suche nach dem Medikament, die Einnahme wieder zu vergessen. Die Arbeit von Laput gliedert sich ebenfalls in diesen Bereich ein und hat auch einen Kontextgewinn zur Folge [13]. Berührt Träger des Smartwatch Prototyp einen Gegenstand, so erkennt die Uhr das elektromagnetische Feld des Gegenstandes. Durch maschinelles Lernen werden nun die Gegenstände mit ihren elektromagnetischen Feld verknüpft. Nun kann die Uhr erkennen, welchen Gegenstand der Träger berührt. Dieser Kontext Gewinn, welche Gegenstände der Träger der Uhr berührt, könnte Fehler bei Medikamenteneinnahmen verhindern, indem die Uhr die Medikamente erkennt, die der Patient berührt. Die Studie "SSmartwatch in vivo" [21] untersucht das Nutzungsverhalten von Smartwatch Nutzern. Hierzu trägt der Nutzer eine Schulterkamera, die die Interaktion mit der Uhr über 3 Tage filmt. So zeigt die Studie auf, dass neben der Uhrzeit (mit ca. 50%

Nutzungsdauer) die Notification mit 20% an Nutzungsdauer die häufigste Interaktion ist. Anwendungen werden so gut wie nie genutzt. Auch die durchschnittliche Interaktionszeit von ca. 7s ist ein wichtiges Ergebnis.

Orientiert man sich an Arbeiten, deren Ziel die smartphonegestützte Medication Adherence war, findet man unter anderem die aktuelle nationale Umfrage [12] aus den USA, bei der 1604 Smartphone Nutzer zu ihrer Nutzung von Gesundheitsanwendungen befragt wurden. Fast 50% der Befragten gaben an, eine oder mehrere Gesundheitsanwendungen auf ihrem Smartphone installiert zu haben. Eine wichtige Aussage der Umfrage ist, dass der Großteil der Nutzer nicht bereit ist für Gesundheitsanwendungen zu bezahlen.

## **2.3. Apple's Programmiersprache Swift**

Swift wurde im Juni 2014 von Apple vorgestellt und genießt seitdem steigendes Interesse. Im Juni 2015 wurde Version 2.0 veröffentlicht [2]. 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 [2]. Diesen Plan setzte Apple Ende 2015 in die Tat um. Swift ist nun völlig Open Source. Es können Vorschläge für neue Sprachfunktionen gemacht werden. Auch Apple's Entwicklungsteam diskutiert seine Pläne für die Sprache öffentlich [11].

### **2.3.1. Objective-C und Swift**

In den frühen 80er Jahren entwickelte Brad Cox die Sprache Objective-C [3]. Weiter führt Dalrymple aus, dass die Sprache die Vorteile einer schnellen C-Sprache mit den Vorteilen der objektorientierten Sprache SmallTalk verbinden sollte. Die Firma NextSTEP nutzte Objective-C und als NEXTStep von Apple aufgekauft wurde, integrierte Apple Objective-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 gegenüber negativ eingestellt. Apple stand nun unter Zugzwang, um seine Plattform, mit der große wirtschaftliche Interessen verbunden sind, für Entwickler attraktiv zu halten [24]. Da jedoch alle highlevel APIs in Objective-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 Objective-C APIs [24].

### 2.3.2. Überblick der Neuerung

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 instanziiert wurde. Es ist nicht nötig für die Variable eine Typendefinition zu definieren [2]. 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 inferred 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 Referenz in eine Variable gespeichert werden. Folgende Code Beispiele sollen dies veranschaulichen. Wir nutzen hierfür einen Asynchrones Netzwerk-Request [2].

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 Requests 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 lokalen Variablen zu speichern. Nun kann diese Funktion als Referenz zum Netzwerk-Requests übergeben werden. Die Funktion wird aufgerufen, wenn der Netzwerk-Requests beendet ist.

### 2.3.5. Generics

Generics sind schon längere Zeit Teil moderner Programmiersprachen. Sie unterstützen den Entwickler, um Fehler zur Compilezeit zu entdecken. Ein sehr gutes Beispiel hierfür sind Collections (Arrays, Listen, Set, etc). Ein Collection Typ wie Array muss nicht für jeden Typ, den er hält, neu implementiert werden.



Listing 2.4: Swift Beispiel zu Generic Array Collection

```
class Medication: NSObject {
    //...
}
//Could only store type Medication
class CustomMedicationArray {
    //...
    func add(medication: Medication) {
        //...
    }

    func objectAtIndex(index: Int) -> Medication {

    }
}
//Could only store any type which inherits from NSObject
//loses all type information
class CustomArray {
    //...
    func add(medication: NSObject) {
        //...
    }

    func objectAtIndex(index: Int) -> NSObject {
        //...
        //return object
    }
}
//Could store any type
//Does not lose type information
class GenericArray<T> {
    //...
    func add(medication: T) {
        //...
    }

    func objectAtIndex(index: Int) -> T {
        //...
        //return object
    }
}
let genericArray = GenericArray<Medication>()
```

Im Codebeispiel 2.4 werden die Vorteile von Generics aufgezeigt. Der erste Versuch, einen Array zu implementieren, (CustomMedicationArray) zeigt, dass dieser Array zwar eine sehr gute Typen Deklaration enthält. Der Compiler kann den Entwickler also maximal unterstützen, jedoch ist diese Implementierung minimal wiederverwendbar. Der zweite Ansatz (CustomArray) gibt als Typ-Einschränkung den globalen Supertyp an. Dies führt zu einer maximalen Wiederverwendbarkeit, da jede Klasse von diesem globalen Supertyp erbt (funktioniert nur theoretisch, da in Swift kein Zwang besteht, von einem globalen Supertyp zu erben). Die letzte Implementierung (GenericArray) nutzt nun Generics. So wird bei der Definition der Klasse ein generischer Typ "T" eingeführt. Dieser Typ ist eine Art Platzhalter für einen konkreten Typ, der später vom Array gehalten wird. So muss bei der Initialisierung der generischen Klasse die Typ-Information für T mitübergeben werden (siehe letzte Zeile in 2.4).

Das oben beschriebene Beispiel dient nur zur Verdeutlichung des Konzeptes. Swift bietet eine Reihe an Collection Types, darunter auch eine generische Array Implementierung.

### 2.3.6. Protocol Extension

Protocol Extension helfen dem Entwickler Abstraktionen, die über ein Interface (in Swift Protocol) definiert werden, zu implementieren und dadurch Duplizierungen zu minimieren. So kann es sein, dass eine Methode eines Interfaces in jeder Klasse, die das Interface implementiert, gleich umgesetzt ist. Dies führt zu einer Code-Duplizierung.

Listing 2.5: Swift Beispiel zu Generic Array Collection

```
protocol List {
    typealias T
    var count: Int { get }
    func objectAtIndex(index: Int) -> T

    var last: T? { get }
    var first: T? { get }
}

extension List {
    var last: T {
        return objectAtIndex(count - 1)
    }

    var first: T {
        return objectAtIndex(0)
    }
}
```

```

class ArrayList<Element>: List {
    typealias T = Element
    var count: Int = 0

    func objectAtIndex(index: Int) -> T {
        //some Implementation
    }
}

```

Im Codebeispiel 2.5 wird ein List Interface definiert. Die Extension zu diesem Interface ermöglicht die Implementierung von "last" und "first". Diese Implementierung teilen sich nun alle Klassen, die List implementieren.

### 2.3.7. 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 eine null-Referenz übergabe [2]. Im folgenden Beispiel wird eine Medikation in Java ohne Optional Types und in Swift mit Optional Types modelliert.

Listing 2.6: 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 2.6 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 2.7: 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 2.7 sind die Instanzvariablen nun mit Optional Types modelliert. 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

Übersetzt man es ins deutsche, bedeutet es "Tragbares oder Anziehbares im Sinne von einem Kleidungsstück tragen. Wenn man nun den Begriff Wearables mit Kleidung assoziiert, liegt man nicht falsch. Anstatt Computer auf dem Schreibtisch oder in der Hosentasche zu haben, trägt man sie am Körper wie Kleidungs- oder Schmuckstücke [4]. Die Grenzen zwischen Kleidung und Computer verschmelzen und es ist manchmal nicht klar, was man schon als Wearable bezeichnen kann oder auch nicht.

Wearables sind meist mit Sensoren ausgestattet, die Daten aus der Körperregion sammeln, an der sie getragen werden [23]. Diese Daten zeigen immer nur einen Teilausschnitt. Durch Tragen von mehreren Geräten, am Körper verteilt, können mehr Daten gesammelt werden. So ist es möglich einen noch genaueren Überblick über den Zustand des Körpers zu erhalten [23].

Erst mit der Miniaturisierung der Computertechnik und Sensoren, war es möglich Computer mit integriertem Akku in Größe einer Streichholzschachtel herzustellen. Während Swatch 1995/1996 eine Uhr vorstellte, die als Skipass funktionierte stellte Apple 18 Jahre später die Apple Watch vor, die über einen Mikroprozessor, WLAN und eine Vielzahl an Sensoren verfügt (mehr zur Apple Watch in 3.4)

Doch nicht nur Uhren zählen zu den Wearables. Google präsentierte mit der Google Glass eine Datenbrille, die über eine Kamera, ein Heads "Prismatic head-mount" Display, Sprachsteuerung, sowie Internetverbindung und weitere Sensoren verfügt [15]. Auch ein Ring, der mit Hilfe eines Sensoren den Puls misst existiert als Prototyp [23].

Während im Smartphone-Markt noch einen Focus auf Leistung und Funktion der Geräte legte, darf man beim Wearables-Markt des Faktor der Ästhetik nicht vergessen. Hier wird ein Bereich betreten, der starke Einflüsse von Mode aufzeigt. Anwender einer Gerätes

achten also nicht mehr nur auch die Funktion, sondern auch auf Form, Farbe und Lifestyle, den das Produkt verkörpert. Apple bietet unter dem Namen "Watch Edition" eine Apple Watch aus echtem Gold an, deren Preis über 10.000 Euro beträgt und sich an den Luxusmarkt richtet. Auch TAGHeuer, eine Firma die sich auf luxuriöses modische Uhren spezialisiert hat, betritt nun auch den Markt der Wearables [22].



## 3. Analyse und Entwurf

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

### 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 Git versioniert verwaltet.

### 3.2. Anforderungsanalyse

Die Anforderungen wurden von eine Gerspräch mit Monika Pobiruchin am Anfang vom PITA 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-Usecases abgeleitet. Die folgenden Usecase 1 bis 4 wurden im PITA erarbeitet und wurden von dort übernommen.

Usecase 1	Der Patient wird an ein Medikament erinnert
<i>Primärer Akteur</i>	Patient

<i>Beschreibung</i>	Ein Erinnerungs Pop-Up erscheint auf dem Display und es wird ein Signal/eine Vibration ausgelöst. Das Pop-Up zeigt ein Abbild des Medikaments, dessen Namen und die Uhrzeit, zu der es eingenommen werden soll.
<i>Vorbedingung</i>	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen.
<i>Ablauf</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display. Die Erinnerung beinhaltet Informationen zur Uhrzeit, Menge und Art der Medikation</li> <li>• Der Patient bestätigt, dass er das Medikament genommen hat</li> <li>• Gerät bestätigt visuell dass der Patient das Medikament als genommen markiert hat</li> </ul>
<i>Alternativablauf</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient wählt Option zum Verschieben der Medikation aus</li> <li>• Auf einem zusätzlichen Dialog kann er aus einer Auswahl eine Zeitdauer wählen, um das die Medikation verschoben wird</li> </ul>
<i>Alternativablauf 2</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient reagiert nicht auf die Erinnerung</li> <li>• Die Erinnerung wird alle x Minuten wiederholt, solange der Patient nicht reagiert.</li> <li>• Das Fehlen einer Reaktion des Patienten innerhalb einer Zeit von x Minuten wird vermerkt</li> </ul>
<i>Ergebnis</i>	Der Patient hat die Einnahme des Medikaments bestätigt und dieses auch eingenommen
<i>Alternativergebnis 1</i>	Der Patient hat die Erinnerung an die Medikamenteneinnahme verschoben
<i>Alternativergebnis 2</i>	Der Patient hat die Erinnerung an das Medikament ausgeschaltet



<b>Usecase 2</b>	<b>Der Patient wird an mehrere Medikamente erinnert</b>
<i>Primärer Akteur</i>	Patient
<i>Beschreibung</i>	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.
<i>Vorbedingung</i>	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen.
<i>Ablauf</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display. Die Erinnerung beinhaltet Informationen zur Uhrzeit, Menge und Art der Medikationen</li> <li>• Der Patient bestätigt, dass er die Medikamente alle genommen hat</li> <li>• Gerät bestätigt visuell, dass der Patient die Medikamente als genommen markiert hat</li> </ul>
<i>Alternativablauf</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient wählt Option zum Verschieben der Medikationen aus</li> <li>• Auf einem zusätzlichem Dialog kann er aus einer Auswahl eine Zeitdauer wählen, um das die Medikationen verschoben werden</li> </ul>
<i>Alternativablauf 2</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient wählt ein Medikament von der Liste aus</li> <li>• Patient befindet sich nun im Usecase 1</li> </ul>

<i>Alternativablauf 3</i>	<ul style="list-style-type: none"> <li>• Das Pop-Up mit der Erinnerung erscheint auf dem Smartwatch-Display</li> <li>• Der Patient reagiert nicht auf die Erinnerung</li> <li>• Die Erinnerung wird alle x Minuten wiederholt, solange der Patient nicht reagiert.</li> <li>• Das Fehlen einer Reaktion des Patienten innerhalb einer Zeit von x Minuten wird vermerkt</li> </ul>
<i>Ergebnis</i>	Der Patient hat die Einnahme der MEDikamente bestätigt und dieses auch eingenommen
<i>Alternativergebnis 1</i>	Der Patient hat die Erinnerung an die Medikamenteneinnahme verschoben
<i>Alternativergebnis 2</i>	Der Patient hat die Erinnerung an das Medikament ausgeschaltet
<i>Alternativergebnis 3</i>	Der Patient hat bestimmte Medikamente ausgewählt und mit dem weiterführenden Usecase 1 bearbeitet

<b>Usecase 3</b>	<b>Einzelne Einnahmebestätigung zurücknehmen</b>
<i>Primärer Akteur</i>	Patient
<i>Vorbedingung</i>	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen. Eine Medikation wurde als genommen markiert
<i>Ablauf</i>	<ul style="list-style-type: none"> <li>• Das System zeigt das genommene Medikament an</li> <li>• Der Benutzer drückt auf den Button mit der Aufschrift "Rücknahme"</li> <li>• Das System wechselt zur Darstellung eines einzelnen Medikaments, beschrieben im Usecase 1</li> </ul>
<i>Ergebnis</i>	Die Einnahmebestätigung ist zurückgenommen. Die Erinnerung ist erneut zu bestätigen oder zu verschieben.

<b>Usecase 4</b>	<b>Mehrere Einnahmebestätigungen zurücknehmen</b>
<i>Primärer Akteur</i>	Patient
<i>Vorbedingung</i>	Es wurde ein Medikationsplan aus der DB auf die Uhr geladen. Mehrere Medikationen, welche zur gleichen Zeit genommen wurden, wurde als genommen markiert
<i>Ablauf</i>	<ul style="list-style-type: none"> <li>• Das System zeigt die genommenen Medikamente an</li> <li>• Der Benutzer drückt auf den Button mit der Aufschrift "Rücknahme"</li> <li>• Das System wechselt zur Darstellung mehrere Medikamente, beschrieben im UseCase 2</li> </ul>
<i>Ergebnis</i>	Die Einnahmebestätigung ist zurückgenommen. Die Erinnerung ist erneut zu bestätigen oder zu verschieben.

Diese vier Usecases sind auch Grundlage für diese Arbeit. Wie die Usecases umgesetzt sind wird in Kapitel 4 beschrieben.

## 3.4. Apple Watch

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

### 3.4.1. Hardware

Die Apple Watch existiert in zwei Versionen. Eine 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 [17].

### 3.4.2. Abhängigkeit zum iPhone

Die Apple Watch wurde als Erweiterung zum iPhone entwickelt. Und so sind auch viele integrale Funktionen nur vom iPhone aus steuerbar. Die Uhr kann ohne ein iPhone nicht durch den Setup-Prozess geleitet werden. Auch native Anwendungen (siehe 3.4.3) können nur über das iPhone installiert werden. Sind diese Schritte getan, also die Uhr betriebsbereit und Anwendungen installiert, kann die Uhr teilweise auch autonom agieren. So kann sie auch ohne iPhone, über WLAN, mit dem Internet kommunizieren.

### 3.4.3. Software

Mit dem Erscheinen der Uhr wurde auch das Betriebssystem in Version 1 ausgeliefert und dazu das Software Development Kit namens WatchKit. Dies erlaubte es Entwicklern Anwendungen zu entwickeln, welche auf dem verbundenen iPhone ausgeführt wurden. Diese führte zu schlechter Performance der Anwendungen und zu voller Abhängigkeit zum iPhone.

Im Juni 2015 veröffentlichte Apple die erste Vorabversion von watchOS 2.0, welches später im September 2015 für Endnutzer bereit gestellt wurde. watchOS bietet mehr Unabhängigkeit für Anwendungen, da diese direkt auf der Uhr ausgeführt werden. Für Anwendungsentwickler gibt es vier Arten Informationen auf der Uhr darzustellen. Es handelt sich um native Anwendungen (Apps), Glances, Complications, und Actionable Notifications [9].

Native Anwendungen sind fest installiert auf der Uhr. Sie können unabhängig auf der Uhr gestartet werden. In einer native Anwendung lassen sich komplexere Anwendungen realisieren, da der Nutzer durch viele Möglichkeiten hat der Eingaben und Interaktionen (3.4.6 zu tätigen). Installiert werden die Apps vom iPhone aus. Eine Watch-App benötigt immer eine iPhone App, die jeweils auf dem iPhone installiert ist.

Complications sind kleine Interface Elemente, die sich auf dem Zifferblatt der Uhr platzieren lassen. So kann mit einem Blick auf die Uhrzeit auch Informationen aus der App abgelesen werden.

Ein Glance ist ein Interface, auf dem die wichtigsten Informationen einer App übersichtlich dargestellt werden. Der Nutzer soll mit einem Blick die Informationen erkennen. Es ist keine Interaktion mit einem Glance möglich. Tippt der Nutzer auf einen Glance öffnet sich die zugehörige App. Glances sind optional zu einer App zu entwickeln.

Notifications oder auch auf deutsch Benachrichtigungen informieren den Benutzer über Ereignisse. Diese Ereignisse können entweder zeitlich geplant sein, vom Betreten einer Ortskoordinate ausgelöst werden oder von einem Server auf das Gerät gepusht (Server sendet ein Ereignis, wie z.B. eine Nachricht) werden. Notification können Aktionen beinhalten. So kann eine Bestätigung einer Notification direkt geschehen, ohne dafür die dazugehörige Anwendung zu starten. Dies nennt man eine Actionable Notification [9].

Es gibt zwei verschiedene Arten von Notification, die beide die gleiche Funktion haben, jedoch mit unterschiedlichen Informationsgehalt für die Uhr angereichert werden. Zum einen die Standard Notification. Diese Notification wird vom iPhone gespiegelt, bietet also nicht mehr Informationen gegenüber der iPhone Notification. Es ist jedoch möglich, mit einer nativen Anwendung auch eine optimale Darstellung der Notification zu entwickeln. Diese Darstellung kann detaillierte Informationen beinhalten wie Bilder, Karten oder genauer Beschreibung der Information [10].

#### **3.4.4. Schniststellen**

Bluetooth 4.0 und Wi-Fi 802.11b/g sind die Netzwerkschnittstellen 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 Bezahl Dienst, vorgesehen ist [18].

#### **3.4.5. Sensoren**

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

#### **3.4.6. Eingabe Interfaces**

Neben eines normalen Touchscreen führte Apple in der Uhr auch eine Eingabeart Name 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 Interaktionsdimension 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.

### **3.4.7. Armband**

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

Das in der günstigsten Version mitgelieferte 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 bieten eine einfache 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.

### **3.4.8. Prototyping**

Zum Erstellen des frühen Prototyps wurden Papier-Prototypen erstellt. Durch die Nutzung dieser Werkzeuge konnte schnell und iterativ gearbeitet werden. Da diese Arbeit nur eine Person gearbeitet hat, bietet sich diese Methode an, da sie schnell zu erlernen ist, bzw. keine Vorkenntnisse voraussetzt und schnell Ergebnisse erzielt.

## 4. Umsetzung

Die in Kapitel 3 aufgeführten Analyse Ergebnisse wurden für die erste Iteration des Prototyps umgesetzt. Es handelt sich hierbei um ein funktionsfähigen Prototypen, der nativ auf der Apple Watch ausführbar ist. Im folgenden werden Schritte der Umsetzung genauer beschrieben. Hierbei wird genauer auf die Benutzeroberflächenerstellung, sowie auch die Verbindung zwischen Uhr und iPhone eingegangen.

### 4.1. Werkzeuge

Für die Entwicklung wurde Apple Entwicklungsumgebung Xcode in Version 7.2.1 mit den SDKs iOS 9.2 und watchOS 2.1. Zum Verwalten des des Source-Codes wurde Git [5] verwendet. Als Git-Remoteserver wurde Github [6] verwendet. Die graphische Repräsentationen der Medikamente keine statischen Bilder. Jede Art von Medikament hat eine Vorlage, die dynamisch mit jeglicher Farbe eingefärbt werden kann. Zum Erstellen dieser Vorlagen wurde PaintCode in Version 2.4 [16] genutzt.

### 4.2. Benutzeroberfläche

Xcode bietet für visuelle Erstellung von Benutzeroberflächen ein eigenen integrierten Editor namens InterfaceBuilder bereit. Hiermit können graphische Elemente per DragAnd-Drop zu einer Benutzeroberfläche zusammengestellt werden 4.1. Ebenfalls per DragAnd-Drop werden diese Inerface-Elemente mit dem Quellcode verbunden 4.2.

Die Auswahl an Interface-Elemente ist sehr begrenzt und bietet kaum Möglichkeiten eigene Interface-Elemente zu erstellen. Trotz dieser Begrenztheit lassen sich Anwendungen komplexere Anwendungen bauen.

### 4.3. WatchConnectivity

Wichtig für die Kommunikation zwischen Uhr und iPhone ist das WatchConnectivity Framework. Hierbei ist im Listing xx zu sehen, wie genau eine Verbindung aufgebaut

Abbildung 4.1.: Interface Elemente zu Erstellen von Benutzeroberflächen

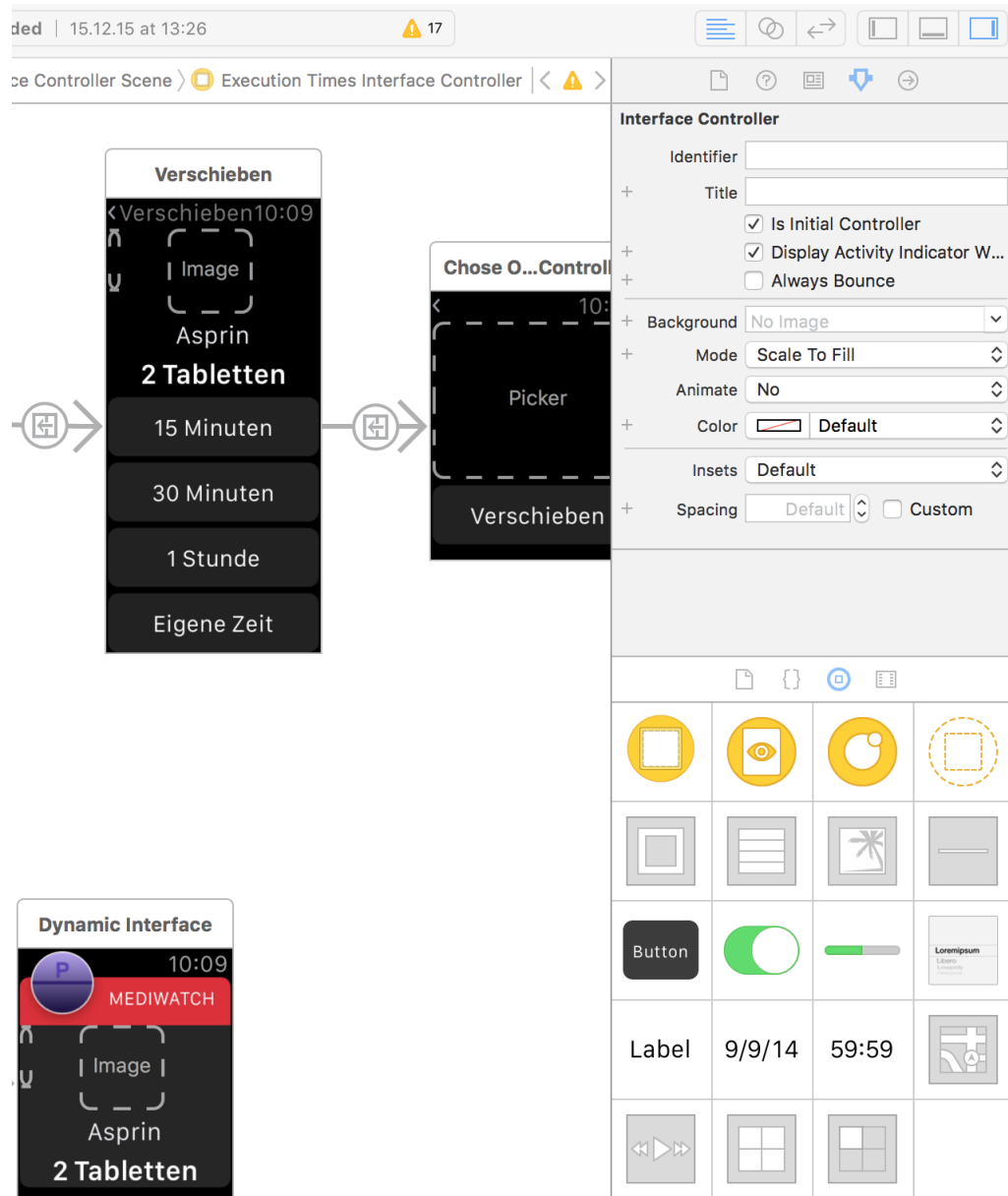
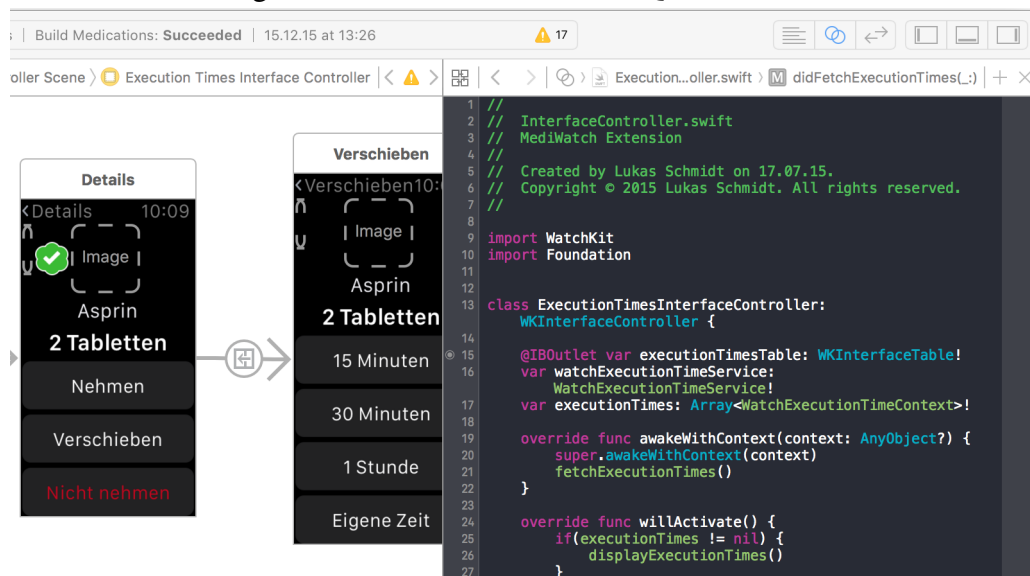




Abbildung 4.2.: Interface Elemente mit Quellcode verknüpfen



werden kann. Wichtig ist, dass diese Verbindung zum richtigen Zeitpunkt im Application-Lifecycle aufgebaut wird, da es sonst zu Datenverlusten kommen kann.

Mit der `sendMessage:`-Methode können Nachrichten und Datenpakete gesendet werden, welche sich durch eine ID identifizieren lassen. Auch ist möglich eine direkte Antwort auf eine solche Nachricht über einen Replay-Handler zurück zu senden. So kann eine Art Request/Response Datentransfer realisiert werden. Zu beachten ist jedoch, dass die Kommunikation relativ langsam ist. Sie sollte nur verwendet werden um kleine Informationen zu senden.

Werden erst beim Watch-App Start benötigte Daten vom der iPhone angefordert, kann dies zu einer großen Verzögerung kommen. Daher sollten Informationen, die auf der Uhr angezeigt werden, nicht erst zum Zeitpunkt der Darstellung angefordert werden. Gibt es eine Datenänderung auf dem iPhone, die relevante Daten für die Uhr enthält, so sollten diese mit der Methode `transferUserInfo:` an die Uhr gesendet werden. Mit dem Aufrufen dieser Methode wird die übergebene Information nicht direkt zu Uhr geschickt. Das System sendet die Daten zu einem optimalen Zeitpunkt (starke Verbindung zur Uhr, mögliche WLAN Verbindung) an die Uhr. Wenn nun die Watch-App startet, so sind die Daten bereit und können direkt dargestellt werden.

Zum Übertragen von größeren Daten steht die Methode `transferFile:` zur Verfügung. Die Methode verhält sich äquivalent zu `transferUserInfo:`, wird jedoch in der Realisierung nicht verwendet.

## 4.4. Notification Management

Notifications für die Medikationen werden mit der Notification API registriert werden [10]. Auch kann das Verhalten der Notifications angepasst werden. So werden vordefinierte Aktionen wie „Einnehmen“ oder „Verschieben“ in die Notifications konfiguriert.

Bei der Notification handelt sich um eine `UILocalNotification` [10]. Die Art von Notification benötigt keinen Remote Server, sondern wird vom verbundenen iPhone verwaltet. Zum jetzigen Zeitpunkt ist es nicht möglich `UILocalNotification` von der Uhr zu verwalten. Dies würde die Uhr im Falle der Mediwatch-Anwendung noch unabhängiger machen. Sind die Notifications einmal auf dem iPhone registriert, werden sie zum Ausführungszeitpunkt auf der Uhr angezeigt. Dazu ist keine Verbindung zum iPhone mehr nötig.

## 4.5. Anwendung

Die Anwendung besteht aus 2 Komponenten auf der Apple Watch. Eine native Notification und eine auf der Uhr installierte und ausgeführte Anwendung (siehe 3.4.3).

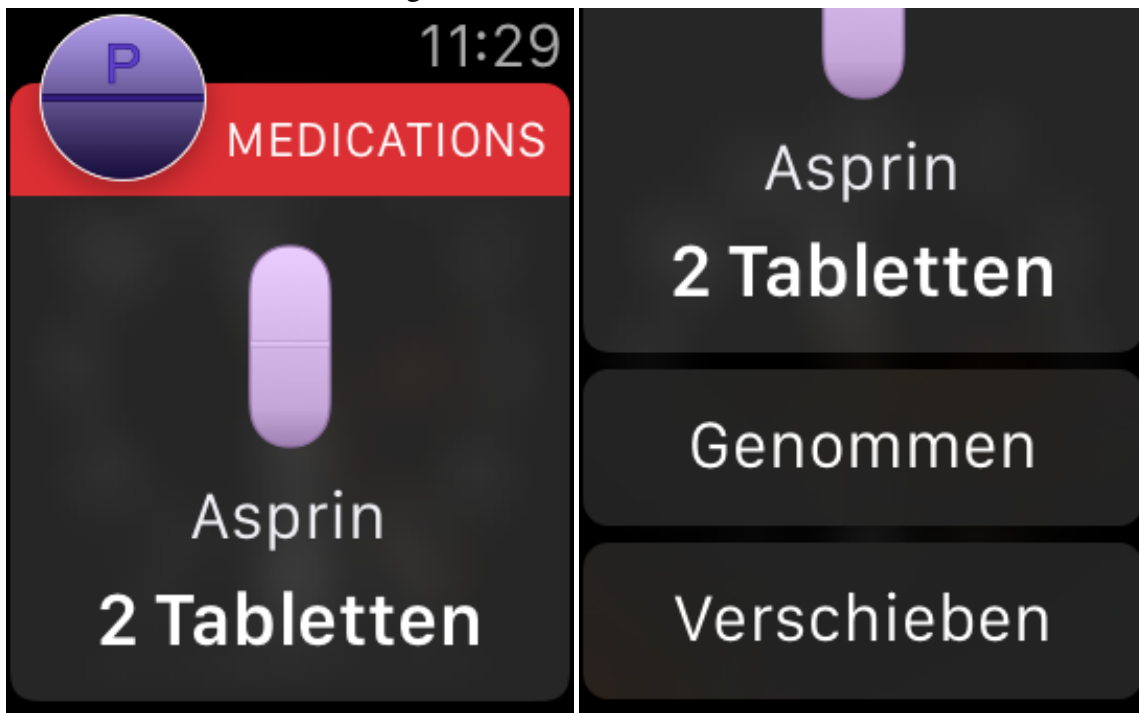
### 4.5.1. Notification

Die Notification bringt die Uhr zum Zeitpunkt der geplanten Medikamenteneinnahme zum vibrieren und lässt optional einen Ton erklingen. Dies führt dazu, dass der Anwender auf die Uhr Blick und die bevorstehende Medikamenteneinnahme bemerkt. Die Notification wurde darauf hin optimiert, dass eine große graphische Repräsentation des Medikaments angezeigt wird. Dem Nutzer wird so die Wiedererkennung des Medikaments erleichtert (siehe 4.3). Neben dem Medikamentennamen wird auch die Dosierung des Medikaments dargestellt. Es wird dem Nutzer ermöglicht mit nur einer Aktion das Medikament zu Bestätigen oder es zu Verschieben. Die Notification ist darauf ausgelegt eine sehr schnelle Interaktion zu ermöglichen, damit der Nutzer nicht länger als 10 Sekunden mit der Uhr interagieren muss (siehe 4.3).

### 4.5.2. Native Anwendung

Neben der Notification gibt es noch eine native App. Diese läuft auf der Uhr und stellt eine Liste der Medikamente des aktuellen Tages dar. Die App muss aktiv vom Nutzer gestartet werden. In 4.4 wird ein Medikament aus der List ausgewählt. Nach dem man ein Medikament ausgewählt hat, bekommt man in einer Detailansicht des Medikaments zu sehen. Dort kann man nun das Medikament als genommen oder nicht genommen markieren. Ein visuelles Feedback zur Einnahme zeigt dem Nutzer seine Aktionen an. Von der Detailansicht ist es auch möglich ein Medikament zur späteren Einnahme zu markieren.

Abbildung 4.3.: Notification für Medikament



Hier hat der Anwender die Möglichkeit aus vordefinierten Zeiten auszuwählen oder eine eigene Zeit zu wählen (siehe 4.5).

Abbildung 4.4.: Native Anwendung: Interface zu wählen von eines Medikaments (oben links). Medikament als genommen bestätigen (oben rechts). Eigene Zeitdauer auswählen (unten links). Medikament ist in der Übersicht als verschoben markiert (unten rechts)



Abbildung 4.5.: Native Anwendung: Interface zu verschieben von eines Medikaments (oben links). Zeitdauer für das Verschiebenwählen (oben rechts). Medikament ist als genommen markiert (unten links). Medikament ist in der Übersicht als genommen markiert(unten rechts)





## **5. Evaluation, Zusammenfassung und Ausblick**

### **5.1. Evaluation**

Die Evaluierung des Prototypen soll eine quantifizierte Ergebnis liefern, anhand dessen Schwachstellen aufgezeigt werden sollen. Diese Schwachstellen sollen mit der nächsten Iteration der Software ausgebessert werden.

#### **5.1.1. Zielgruppe**

Die Zielgruppe sollten Menschen sein, deren Alltag von Medikamenteneinnahmen geprägt ist. Es sollte sich um Menschen mit wenig oder keinen Vorkenntnissen mit Touchscreen-basierten Geräten handeln. Patienten fortgeschrittenen Alters, die sich stationär im Krankenhaus aufhalten eignen sich gut, Medikamente nehmen und viel freie Zeit während des Aufenthaltes im Krankenhaus für eine Befragung haben.

#### **5.1.2. Vorgehen**

Im ersten Schritt wird der Zielgruppe die Apple Watch gezeigt und die dahinterliegende Technologie erklärt, um Neugier bei den Patienten zu wecken. Nun können die Patienten die Uhr anlegen. Im zweiten Schritt wird eine Notification, die eine Vibration auslöst gestartet. Die Patienten müssen die Vibration spüren und dann auf die Uhr schauen. Dort sollen sie den Bildschirminhalt wiedergeben. Haben sie den Bildschirminhalt wiedergegeben, so müssen sie die Benachrichtigung bestätigen, also auf den Button mit der Aufschrift "Genommen" tippen.

In Schritt Drei müssen die Patienten durch drücken auf die digital Crown den App Bildschirm aufrufen um dort die Mediwatch App zu öffnen. Wenn die App geöffnet ist, soll ein Medikament ausgewählt werden und dieses um eine definierte Zeit verschoben werden. Ist das Medikament verschoben, so ist die Aktion beendet.

### **5.1.3. Auswertung**

Für die Evaluation werden 2 Fragebögen genutzt. Einmal AttrakDiff [7], welcher die subjektiv Wahrnehmung der Bedienbarkeit und Aussehen des Prototypen erfragt. Es handelt sich um einen standardisierten Fragebogen und eine standardisierten Auswertungsmethode. Da dieser Fragebogen keine Antworten über Funktionen des Prototyp gibt, ist es nötig einen zweiten Fragebogen zu erstellen. Dieser Fragebogen erfragt die Situation, also den Kontext in dem sich der Anwender befindet und die daraus Folgenden Ansprüche an den Prototypen. So sollen fehlende Funktionen oder Fehler in der Analyse aufdecken werden. Die Fragen sind im Anhang zu finden. Die Erfassung der Fragen wird mit Limesurvey [1] in Version 2.05 durchgeführt. Da sich Limesurvey selbst hosten lässt, bleiben die erfragten Daten auf einem sicheren Hochschulserver und werden nicht bei Drittanbietern gespeichert.

## **5.2. Durchführung der Evaluation**

### **5.2.1. Beschreibung der Zielgruppe**

Es haben insgesamt 11 Patienten an der Befragung teilgenommen. Davon waren 7 weiblich und 4 männlich. Alle waren im Alter zwischen 70 und 85.

### **5.2.2. Befragung der Patienten**

Die Befragung in der Zielgruppe verlief sehr schwierig und nicht wie geplant. Der erste Schritt der Befragung klappte bei fast allen Probanden sehr gut. So erkannten alle die Notification mit der einhergehenden Vibration wurde von allen Patienten als verständlich geschildert. Das darauf folgende Bestätigen einer Notification verlief zu großen Teilen schlecht. Die Patienten konnten die Buttons zum Bestätigen nicht treffen, da ihnen die Feinmotorik fehlte. Öffnen einer App ist auf Grund der fehlenden Feinmotorik ebenfalls nicht möglich. Die App-Icons sind zu klein und werden nicht getroffen. Das Öffnen der Anwendung wurde bei allen Patienten nicht erreicht.

### **5.2.3. Auswertung**

Der Fragebogen AttrakDiff ist in dieser Zielgruppe nicht praktikabel. Eine sehr genauer Unterscheidung zwischen den Begrifflichkeit, die der AttrakDiff abfragt ist für die Zielgruppe nicht möglich. Dies fällt auf, wenn man mit den Probanden spricht. Die Probanden schweifen oft ab und es ist nicht möglich ihre Aussagen zu erfassen. Direkte Fragen, die den Alltag der Probanden betreffen werden zuverlässig beantwortet. Schwierigkeiten, die



bei der Medikamenten Einnahme werden nicht zugegeben. Die Befragung wurden deswegen auf eine Thinking Aloud Methode [20] umgestellt. Leider sind die Ergebnisse so nicht mehr quantifizierter, jedoch sind aus den Aussagen der Patienten und den Beobachtungen gute Schlussfolgerungen möglich. Auch der zweite Fragebogen zu den Funktionen und deren Nutzungskontext wurde nicht genutzt. Hier wurde versucht Während er Gesprächs mehr Einblick in das Leben der Probanden zu erhalten. Probanden erzählen gerne über ihr Leben und suchen eher das persönliche Gespräch. Aus diesen Gesprächen geht hervor, dass die Patienten mit großer Übereinstimmung keinen Internetanschluss besitzen. Dies schließt entfernte Aktualisierungen des Medikationsplans und Einnahme Bestätigungen für den Arzt aus. Bedenken äußern die weiblichen Patientinnen über die Größe und das Aussehen der Uhr. Eine Uhr muss dem Geschmack der Patientin gefallen. Hierbei spielt die Größe der Uhr, wie auch das Aussehen eine Rolle. Bei Beschreibung von anderen modischen Farben der Uhr oder der Armbänder, zeigen sie sich interessiert. Ein Großteil der Patienten sieht jedoch ein, dass sie auf Grund Ihrer Sehschwäche die große Uhr (42mm) brauchen. Die große Uhr wird in Gegenzug als zu Groß beschrieben. Die meisten Patienten haben keine Probleme die Schrift zu lesen. Meist nehmen Sie Ihre Brille zur Hilfe, die im Krankenhaus natürlich immer griffbereit ist. Nur bei einem kleinen Teil kommt es vor, dass sie den Text auf dem Display auch mit Brille nicht lesen können. Das größte Problem ist jedoch die schon genannte fehlende Feinmotorik. So können die Probanden die Uhr nicht wirklich aktiv bedienen, sondern reagieren nur auf die Vibration am Handgelenk. Dies führt zu einer Hilflosigkeit gegenüber der Technik der Uhr.

#### **5.2.4. Probleme tabellarisch**

### **5.3. Zusammenfassung**

Ziel der Arbeit war es, die Akzeptanz eines Systems zur Erinnerung an Medikamente zu testen. Dieses Ziel wurde mit einer Patientenbefragung verfolgt. Patienten stehen dem Konzept der Uhr am Handgelenk, ob sie jetzt digital oder Analog ist positiv gegenüber. So ist die Uhr als Medium für solche Zwecke geeignet. Die Apple Watch, mit ihrer touch-screenbasierten Navigation, stellt für die getestete Zielgruppe eine große Herausforderung da. Fehlende Feinmotorische Fähigkeiten erschwert den Patienten die Interaktion mit der Uhr. Auch sind Vibration und Geräuschwiedergabe zu schwach und können so leicht überhört werden. Das ästhetische Aussehen der Uhr ist nicht zu vernachlässigen, da die Uhr oft als Art Schmuck getragen wird. So trägt die Farbe, Form und Größe zur Akzeptanz der Uhr bei. Findet der Patient die Uhr unschön/unmodisch, so wird er sie nicht tragen und schlimmstenfalls seine Medikamente vergessen.

Da Smartwatches noch keine große Verbreitung haben, sind auch die Werkzeuge zur Umsetzung von Anwendungen begrenzt. Interaktive Prototypen sind auf der Uhr nicht die Verwendung von Code möglich. So vermehrt sich der Aufwand für schnelles iteratives Vorgehen. Das WatchOS SDK von Apple bietet eine relativ übersichtliche Schnittstel-

Tabelle 5.1.: Probleme und mögliche Lösungen des Smartwatch Prototypes

Problem	Auswirkungen	mögliche Lösung
<b>fehlender Internetanschluss</b>	keine Aktualisierungen des Medikamentenplans und keine Benachrichtigungen für den Arzt	Uhr oder Smartphone mit Mobilfunkverbindung ausstatten
<b>fehlende Feinmotorik der Probanden</b>	Buttons werden nicht getroffen oder falsche Aktionen werden ausgelöst	große haptische Knöpfe in die Hardware integrieren, die gut drückbar sind
<b>Vibration zu leicht</b>	Vibration wird nicht gespürt und Medikament wird vergessen	Stärkere Vibration, Lauten Ton dazu abspielen, starken optisches Feedback (grelles Blinken)
<b>Nur mit Brille lesbar</b>	Benachrichtigung wird erkannt, bis jedoch die Brille im Haushalt gefunden, ist die Medikation schon vergessen oder die Uhr hat die Benachrichtigung schon zurückgestellt	Größeres Schrift, die Auch ohne Brille lesbar ist. Audio-wiedergabe der Medikation
<b>Armband schwer anlegbar</b>	Uhr wird morgens nicht gerne Angezogen und bleibt so liegen und Patient bekommt keine Benachrichtigungen	Verzicht auf Standart Sport Band und dafür Armbänder die leicht anzuziehen sind finden. Viele Patienten tragen dehnbare Bänder ohne Verschluss
<b>Uhr ist unästhetisch</b>	Patient trägt die Uhr nicht und wird so nicht an die Medikamente erinnert	Keine Standart Uhren kaufen, sondern den Patienten die Farbe und Form der Uhr und die Art des Armbandes aussuchen lassen

le. Die Schnittstellen ermöglichen es komplexere Anwendungen zu erstellen. Diese sind jedoch an manchen Stellen noch limitiert. Da es sich um die erste Geräte-Generation handelt, muss abgewartet werden, mit welchen Leistungsverbesserungen weiter Generation nachgerüstete werden, um die Uhr zu einem wirklich nützlichen Werkzeug zu machen. Das Potential zu einem guten Nutzen zeigt die erste Generation auf, jedoch ist sie an manchen Stellen (Geschwindigkeit, Lange Wartezeiten) noch nicht ausgereift.

Apples neue Programmiersprache Swift, die zur Implementierung genutzt wurde, bietet Konzepte moderner Programmiersprachen. Diese Konzepte unterstützen den Entwickler meist schon zur Compilezeit. Dies führt zu einer frühen Fehlererkennung. Durch diese Spracheigenschaften, wie Optionals (2.3.7), die Sicherheit in der Modellierung geben oder Closures und First Class Functions (2.3.4) die Konzepte Funktionales Programmierung bereitstellen, eignet sich Swift sehr gut als Lehrsprache. Dadurch das Swift unter einer Open Source Lizenz veröffentlicht wurde, kann man nun für mehr Zielplattformen entwickeln.

## **5.4. Ausblicke**

Mit den gewonnenen Ergebnissen kann dieses Projekt neu ausgerichtet werden. So sollte die Zielgruppe über mehr Erfahrung mit digitalen Geräten verfügen. So wäre eine Unterstützung von Kindern und Jugendlichen im Alter ab 12 Jahren denkbar. Diese könnten bei ihrer Therapie unterstützt werden und durch die moderne Technik, mit der sie aufgewachsen sind, motiviert werden, Medikamente zeitgemäß einzunehmen. Nachdem die Defizite mit der Lesbarkeit durch die Evaluation entdeckt wurden, ergab die Recherche, dass die watchOS Plattform über Funktionen der Accessibility verfügt [8]. Damit ist es möglich Nutzungs-Erleichterungen für Menschen mit körperlichen Einschränkungen zu schaffen. So kann mit DynamicType, also eine dynamische Schriftgröße in der App, die Lesbarkeit verbessert werden. Der Nutzer kann so seine eigene Schriftgröße wählen. Auch die Navigation zum Öffnen von Anwendungen wird mit Accessibility Einstellungen verbessert.



# Literaturverzeichnis

- [1] Limesurvey.
- [2] *The Swift Programming Language*. Apple, Inc., 2014.
- [3] DALRYMPLE, M. Learn objective-c on the mac. 1–20.
- [4] DVORAK, J. L. Moving wearables into the mainstream.
- [5] GIT. Git.
- [6] GITHUB, I. Github.
- [7] GMBH, U. I. D. Attrakdiff fragebogen.
- [8] INC., A. Accessibility - apple watch.
- [9] INC., A. Developing for apple watch.
- [10] INC., A. Notification essentials.
- [11] INC., A. Swift open source.
- [12] KREBS P, D. D. Health app use among us mobile phone owners: A national survey. *MIR mHealth uHealth* (2015).
- [13] 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.
- [14] 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.
- [15] MUENSTERER, O. J., LACHER, M., ZOELLER, C., BRONSTEIN, M., AND KÜBLER, J. Google glass in pediatric surgery: An exploratory study. *International Journal of Surgery* 12, 4 (2014), 281 – 289.
- [16] PIXELCUT. Paintcode.
- [17] RICHES, G. Apple watch for developers.
- [18] RITCHIE, R. Apple watch specs, 2015.
- [19] SAILER, F., POBIRUCHIN, M., WIESNER, M., AND MEIXNER, G. An approach to improve medication adherence by smart watches, 2015.
- [20] SOMMERVILLE, I. *Software engineering*. Pearson, Boston, Munich [u.a.], 2016.
- [21] STEFANIA PIZZA, BARRY BROWN, D. M. A. L., Ed. *Smartwatch in vivo* (2016).
- [22] TAGHEUER. Tagheuer connected.
- [23] TENG, X.-F., ZHANG, Y.-T., POON, C., AND BONATO, P. Wearable medical systems for p-health. *Biomed-*

*cal Engineering, IEEE Reviews in I*  
(2008), 62–74.

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

# A. Appendix

## A.1. Fragebogen

1. Tragen Sie eine Uhr im Alltag
  - a) Ja
  - b) Nein
2. Wären Sie bereit eine Digitale Computer Uhr zu tragen
  - a) Ja
  - b) Nein
3. Finden Sie die Idee gut, von der Uhr an ihre Einnahme erinnert zu werden.
  - a) Ja
  - b) Nein
4. Finden Sie die Idee gut, von der Uhr an ihre genauen Medikamente erinnert zu werden?
  - a) Ja
  - b) Nein
5. Wie viele Medikamente nehmen Sie täglich
  - a) ein Medikament
  - b) mehr als drei
  - c) mehr als fünf
  - d) mehr als acht
6. Haben Sie Probleme, Ihre Medikamente rechtzeitig zu nehmen?
  - a) Ja
  - b) Nein
7. Wissen Sie welche Medikamente die nehmen?
  - a) Ja
  - b) Nein
8. Können Sie die Medikamente an Form und Farbe unterscheiden
9. Haben Sie einen Internetzugang zu Hause?
  - a) Ja
  - b) Nein
10. Benutzen Sie einen Computer?
  - a) Ja
  - b) Nein
11. Benutzen Sie ein mobiles Telefon?
  - a) Ja
  - b) Nein

12. Würden Sie die Erinnerung an die Medikamente mit der Uhr nutzen?

a) Ja

b) Nein

13. Welche Funktionen fehlen, die Sie als wichtig ansehen?

14. Gibt es Funktionen die Sie für unwichtig halten?