RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

BACHELOR/MASTER

ARBEIT

Steht noch nicht ganz fest

Author	Lukas Schmidt
Studiengang	Medizinische Informatik Universtität Heidelberg / Hochschule Heilbronn
Matrikelnummer	182706
Abgabe	9. November 2015
Referent	Prof. Dr.-Ing. Gerrit Meixner
Korreferent	Prof. Dr. Mika Musterperson

Inhaltsverzeichnis

Abbildungsverzeichnis iii

Tabellenverzeichnis v

Glossar vii

1. Einleitung 1
- 1.1. Motivation . . . 1
- 1.2. Zielsezung . . . 1
- 1.3. Aufbau der Arbeit . . . 1

2. Stand der Wissenschaft 3
- 2.1. Einleitende Worte . . . 3
- 2.2. Apple‘s Programmiersprache Swift . . . 3
  - 2.2.1. Objective-C und Swift . . . 3
  - 2.2.2. Überblick der Neuerung . . . 3
  - 2.2.3. Type Inference . . . 4
  - 2.2.4. Closures - Functions as First Class Objects . . . 4
  - 2.2.5. Optional Types . . . 5

3. Problemstellung, Ziele und Vorgehensweise 7

4. Umsetzung 9

5. Zusammenfassung, Evaluation und Ausblick 13

Literaturverzeichnis 15

A. Appendix 17

B. Dokumentation 19

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

Parking Spot Measurement Measurement and visualisation during the passing of the parking spot. 2

XML 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. 1

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 Medikationspläne oder nach Zeit vorsortierten Medikamente kann die 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 nachfolgend gelistet.

1. Entwickeln eines interaktives Prototypen, welcher auf der Apple Watch ausgeführt werden kann.
2. Evaluieren des Prototyps mit gegeigten Personen, die zur passenden Zielgruppe gehören.
3. Umsetzung der Ergebnisse der Evaluierung im Rahmen der technischen Möglichkeiten.
4. Anbinden des Prototypen an das Backend-System vom PITA-Praktikum

1.3. Aufbau der Arbeit

- **Kapitel 2 - Grundlagen**

In Kapitel 2 nehmen wir Bezug auf das Forschungsumfeld der Arbeit. Weiter werden technologische Grundlagen beschrieben. Apple‘s Swift Programmiersprache, sowie die Kommunikation der Apple Watch stehen dabei im Mittelpunkt. Desweiteren werden Grundlagen in der Evaluierung der Gebrauchstauglichkeit beschrieben

- **Kapitel 3 - Analyse und Entwurf**

  Hier werden Anforderung beschrieben, die teilweise aus vorherigem PITA-Praktikum stammen. Weiter werden Anforderungen geschildert, die sich aus den Möglichkeiten der Apple Watch ergeben.

- **Kapitel 4 - Implementierung**

  Im vierten Abschnitt betrachten wir Kernpunkte der Implementierung. Weiter wird hier der Prototyp beschrieben

- **Kapitel 5 - Evaluierung**

  In Kapitel 5 wird die Planung und Durchführung der Evaluierung beschrieben. Die Ergebnisse der Evaluierung an der Zielgruppe sind hier zu finden.

- **Kapitel 6 - Diskussion**

  In der Diskussion 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

2.2. 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. 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.

2.2.1. Objective-C und Swift

In den frühen 80er Jahren entwickelte Brad Cox die Sprache Objective-C. Die Sprache sollte die Vorteile einer schnellen C-Sprach mit den Vorteilen der objektorientierten Sprache SmallTalk verbinden. Die Firma NextSTEP nutze Objectiv-C und als NEXTStep von Apple aufgekauft wurde, integrierte Apple Objectiv-C und ermöglichte Mac-Entwicklern die Nutzung [1]. 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 [2]. Da jedoch alle highlevel APIs in Objectiv-C vorhanden sind, war es nicht so einfach auf eine bekannte Sprache 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 [2].

2.2.2. Überblick der Neuerung

Swift bringt viele Neuerungen mit sich. Im folegden werden nun 4 Neuerungen der Sprache erläutert. Diese 4 Neuerungen bieten einen großen Mehrwert für Anwendungsentwickler, es sind jedoch nicht die einzigen Neuerungen. Mehr sind hier [2] zu finden. Es handelt sich um folgenden Sprach Features

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

2.2.3. 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. Hierdurch wird der Quellcode leichter zu lesen.

Codebeispiel 2.1: Beispiel zu Type Inference in Swift label

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

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

2.2.4. Closures - Functions as First Class Objects

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.

Codebeispiel 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).

Codebeispiel 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.

2.2.5. Optional Types

Optional Types eröffnen neue Möglichkeiten beim Modellieren von Datenmodellen und erstellen von APIs. Im folgenden Beispiel wird eine Medikation in Java ohne Optional Types und in Swift mit Optional Types modelliert.

Codebeispiel 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.

Codebeispiel 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 Fehlern

3. Problemstellung, Ziele und Vorgehensweise

4. Umsetzung

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.

5. Zusammenfassung, Evaluation und Ausblick

Literaturverzeichnis

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

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

A. Appendix

B. Dokumentation