HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

forventis
inventive solutions

PocketDoc- App für die Selbstdiagnose

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2015

Autor(en):	Philipp Christen
	Roman Eichenberger
Betreuer:	Prof. Dr. Eduard Glatz
Projektpartner:	Dr. med. Samuel Huber, Forventis GmbH
	Flavio Trolese

Abstract

PocketDoc ist eine Software, um die Routinearbeit von Ärzten zu automatisieren. Menschen können damit von überall her einen medizinischen Rat einholen, ohne beim Arzt vorbeigehen zu müssen. Dies wird durch die Beantwortung mehrerer Ja-/Nein Fragen über die auftretenden Symptome und anhand eines dafür vorgesehenen Algorithmus ermöglicht. Besonders bei Bagatellerkrankungen können so die Ärzte entlastet und der Kunde zeitnah über seine mögliche Erkrankung informiert werden.

Ziel dieser Arbeit ist es, eine Benutzeroberfläche zu diesem System umzusetzen. Bisher stand eine einfache Weboberfläche zur Verfügung, welche die rudimentäre Funktionalität des Systems aufzeigt.

Mithilfe von Webtechnologien wie AngularJS wurde dieses Ziel erreicht. Ungefähr in der Hälfte des Projekts stellte sich heraus, dass das vorhandene Backendsystem noch nicht praxistauglich ist. Dies erforderte eine Anpassung der Aufgabenstellung: Auf Kosten einiger Frontend-Funktionen wie der Anzeige der bereits durchgeführten Diagnosen wurde ein Simulator direkt im Frontend implementiert. Dieser ermöglicht die Bedienung der kompletten Applikation. Allerdings wird die nächste, am besten auf die bisherigen Antworten passende und am ehesten zu einer Diagnose führende Frage nicht mehr über einen komplexen Algorithmus bestimmt, sondern durch einen einfachen Entscheidungsbaum. Zusätzlich kann mit Hilfe eines Benutzeraccounts eine Nachfolgeuntersuchtung geplant werden, um eine Diagnose zu einem späteren Zeitpunkt zu verifizieren.

Ein besonderer Fokus wurde auf die einfache und intuitive Bedienung gelegt, welche durch die Zusammenarbeit mit einem UX-Experten von atfront.ch sowie Usertests realisiert wurde. Das Layout wurde iterativ durch mehrere Mockups und klickbare Prototypen erarbeitet und verfeinert. Als grafisches Framework diente das Angular Material Project, um ein einheitliches Aussehen zu gewährleisten.

Da die Funktionsweise der Software durch das fehlende Backendsystem stark eingeschränkt ist, wird einer der nächsten Schritte dessen Implementation sein. Dies könnte z.B. im Rahmen einer weiteren Studien- oder Bachelorarbeit stattfinden.

Das umgesetzte System reicht allerdings aus, um die Funktionalität potentiellen Kunden präsentieren zu können. Zukünftig soll das Projekt an Krankenkassen oder ähnliche Unternehmen weiterverkauft werden.

1 Inhalt

Selbstständigkeitserklärung

Hiermit bestätigen wir,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit erwähnt sind,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Regeln korrekt angegeben haben,
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Rapperswil, 28.05.2015

Philipp Christen Roman Eichenberger

Technischer Bericht

2 Management Summary

2.1 Ausgangslage

Viele Personen gehen heutzutage oftmals wegen Bagatellerkrankungen zum Arzt mit Beschwerden wie Kopfschmerzen, Husten oder Müdigkeit. Dies kostet die Ärzte allerdings viel Zeit, die sie in kritischere Fälle investieren sollten. Genau an dieser Stelle soll PocketDoc zum Zug kommen. Das System übernimmt mit Hilfe von Ja/Nein/Weiss nicht Fragen zu diversen Symptomen die Diagnosestellung und unterstützt dabei den Benutzer mit zusätzlichen Informationen. Damit der Benutzer weiss, wie er reagieren soll, wird zusätzlich eine Handlungsempfehlung angezeigt.

Im Rahmen einer früheren Studienarbeit wurde das Backend System bereits umgesetzt.

Dieser in Java geschriebene Service stellt die Kernfunktionalität zur Verfügung. Dazu wurde ein Admin-Tool mit Hilfe von AngularJS (Angular) entwickelt, um die nötigen Daten erfassen zu können.

Im Rahmen der aktuellen Studienarbeit soll nun das passende Frontend dazu erstellt werden. Dieses wird mit Hilfe von modernen Webtechnologien wie AngularJS umgesetzt. Als Designsprache dient die neue Material Design-Spezifikation (GoogleMD) von Google für Android.

2.2 Vorgehen, Technologien

Die Ausgangslage war relativ knapp vorgegeben: Es soll eine Applikation (App) für mobile Geräte entwickelt werden, welche als Frontend für das bereits vorhandene Backend genutzt werden kann. Dafür wurde nun AngularJS in Kombination mit der dafür bereitgestellten Material Design-Library (AngMat) als Technologie gewählt. Die Vorteile liegen auf der Hand: Durch die webbasierte App kann sie auf jedem System mit Webbrowser genutzt werden. Ausserdem wurde bereits das Admin-Tool (Nathan Bourquin) mit Hilfe von AngularJS umgesetzt, was die Anzahl der benötigten Technologien gering hält und so auch den Wartungsaufwand verringert. Als Negativpunkt könnte man die langsamere Ausführungsgeschwindigkeit in Betracht ziehen. Da die App allerdings so gut wie keine komplexen Berechnungen ausführt, hält sich dies allerdings in Grenzen. Die Kommunikation mit dem Backend-System läuft über die vorhandene REST Schnittstelle.

2.3 Ergebnisse

Das Ergebnis der Arbeit ist eine einfach zu bedienende Webapplikation, welche plattformunabhängig funktioniert. Die Benutzer können schnell und einfach eine Diagnose durchführen und so in Erfahrung bringen, ob sie einen Arzt aufsuchen sollen oder nicht. Für Benutzer, die einen grösseren Funktionsumfang nutzen möchten, besteht die Möglichkeit, ein Konto anzulegen. Dies erlaubt den Benutzern nach einer Diagnose ein Follow-Up anzufordern. Dadurch wird zu einem späteren Zeitpunkt eine weitere Diagnose durchgeführt, um diese zu bestätigen oder allenfalls zu korrigieren.

Da sich das Backend während der SA als noch nicht produktionsreif herausgestellt hat, wurde entschlossen, auf Kosten zweier Funktionen ein Mocking-Backend zu erstellen. Dieses arbeitet mit statischen Daten und Abläufen, um eine Demonstration der Applikation zu ermöglichen. Es soll aber zu einem späteren Zeitpunkt einfach möglich sein, ein funktionierendes Backend einzubinden. Die weggefallenen Funktionen, History Einsicht und

die Wiederaufnahme unterbrochener Diagnosedurchläufe, können in einem zukünftigen Projekt implementiert werden.

2.4 Ausblick

Das Ziel der Applikation ist klar: Es soll jeder Person unabhängig von Zeit und Ort möglich sein, sich schnell und einfach eine Diagnose einholen zu können. Beim Arzt fallen, gerade bei weniger kritischen Symptomen, oftmals längere Wartezeiten an. Zusätzlich wird so der Aufwand für Ärzte verringert und es können, falls doch eine Kontaktaufnahme nötig ist, schneller die nötigen Schritte eingeleitet werden.

Um das Produkt fertigzustellen, wird als nächstes das Backendsystem erneuert, um so den Funktionsumfang des Frontends zu vervollständigen. Dies könnte im Rahmen einer weiteren Studien-/Bachelorarbeit geschehen. Schlussendlich sollte es möglich sein, das Produkt an Versicherungen, Krankenkassen o.a. weiterzuverkaufen.

3 Anforderungsspezifikation

3.1 Use Cases

In diesem Kapitel werden die Use Cases sowohl im *brief*- als auch im *fully dressed*- Format beschrieben.

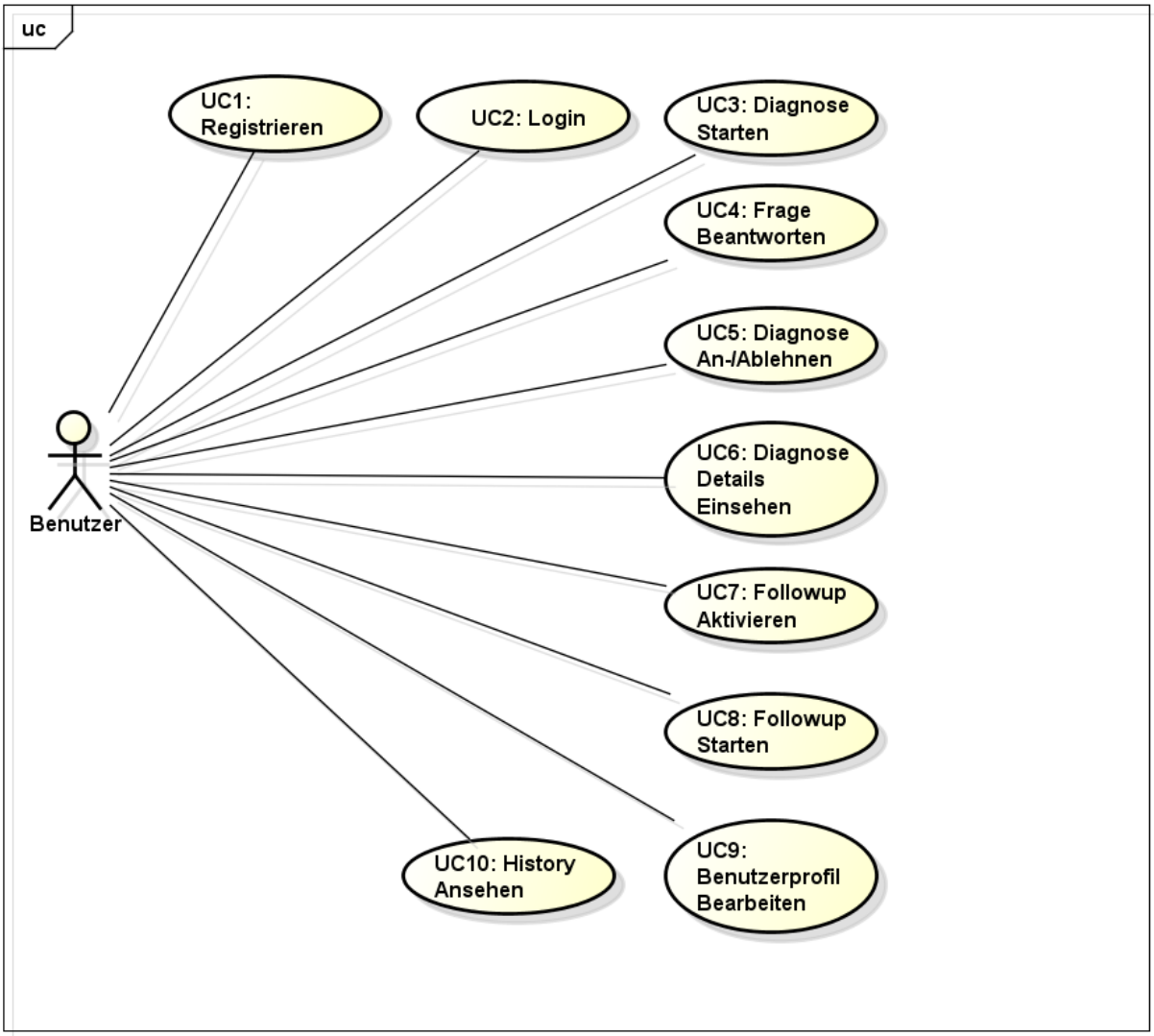


Abbildung 1: Use Case Diagramm

Aktoren und Stakeholders

- Benutzer: Der Benutzer ist der Kunde, welcher mit Hilfe der Applikation eine Diagnose ermitteln möchte.

Beschreibungen “Brief”

ID	Name	Beschreibung
1	Registrieren	Der Benutzer kann ein eigenes Benutzerkonto anlegen, um seine persönlichen Informationen zu speichern und Zugriff auf vergangene Diagnosen zu erhalten sowie um Follow-Ups durchführen zu können.
2	Login	Der Benutzer kann sich mit seinen Benutzerdaten einloggen, um seine persönlichen Informationen einzusehen, zu ändern oder um Follow-Ups respektive offene Diagnosen fortzusetzen.
3	Diagnose starten	Der Benutzer kann den Diagnose-Ablauf starten
4	Frage beantworten	Der Benutzer erhält nacheinander Fragen zu seinen Beschwerden, die er mit Ja, weiss nicht oder Nein beantworten kann.
5	Diagnose An-/Ablehnen	Wenn das System das Gefühl hat, eine Diagnose stellen zu können, erhält der Benutzer eine Benachrichtigung mit der ermittelten Diagnose. Er kann nun entscheiden, ob er die Details dieser anzeigen möchte oder sich nicht sicher ist und weitere Fragen beantworten möchte.
6	Diagnose-Details einsehen	Zur angenommenen Diagnose werden weitere Informationen sowie eine Handlungsempfehlung angezeigt.
7	Follow-Up aktivieren	Am Ende eines Fragedurchlaufs kann der Benutzer das Follow-Up aktivieren, um später einen weiteren Durchlauf durchführen zu können.
8	Follow-Up starten	Der Benutzer Startet das Follow-Up, um die neue Diagnose mit der vorherigen vergleichen zu können.
9	Benutzerprofil bearbeiten	Die Profileinstellungen können vom Benutzer bearbeitet werden. Auch das Löschen des Profils ist möglich.
10	History ansehen	Der Benutzer kann seine bisherigen Diagnosedurchläufe jederzeit wieder ansehen.

Beschreibungen Fully Dressed

Registrieren	
ID	1
Name	Registrieren
Primary Actor	Benutzer
Preconditions	• Der Benutzer wurde noch nicht erfasst
Main Success Scenario	1. Benutzer ruft die Registrationsseite auf. 2. Die Daten zur Registration werden vollständig eingegeben.
Postconditions	Der Benutzer wurde im System erfasst
Alternative Flow	2b) Die eingegebenen Daten sind nicht gültig und die Registrierung wird abgelehnt.
Ergänzungen	Wird die Registration nach einem Diagnosendurchlauf durchgeführt, wird diese Diagnose in der History des Benutzers abgelegt. Zusätzlich wird zu dieser Diagnose ein Follow-Up eingetragen.

Login	
ID	2
Name	Login
Primary Actor	Benutzer
Preconditions	• Der Benutzer befindet sich bereits im System
Main Success Scenario	1. Der Benutzer ruft die Loginseite auf. 2. Er meldet sich mit E-Mail und Password an. 3. Das System erkennt den Benutzer
Postconditions	Der Benutzer wurde vom System erkannt und ist nun eingeloggt.
Alternative Flow	3. (b) Der Benutzer wird nicht erkannt und der Zugriff wird ihm verweigert
Ergänzungen	Wird der Login nach einem Diagnosendurchlauf durchgeführt, wird diese Diagnose in der History des Benutzers abgelegt. Zusätzlich wird zu dieser Diagnose ein Follow-Up eingetragen.

Diagnose starten	
ID	3
Name	Diagnose starten
Primary Actor	Benutzer
Preconditions	• Es sind Fragen im System hinterlegt
Main Success Scenario	1. Der Benutzer betätigt den “Diagnose Starten” Button. 2. Er füllt die persönlichen Daten auf dem Prediagnosis Screen aus 3. Die Daten werden bestätigt und die Diagnose gestartet
Postconditions	Die Diagnose ist gestartet und die erste Frage wird angezeigt.
Alternative Flow	-
Ergänzungen	-

Frage beantworten	
ID	4
Name	Frage Beantworten
Primary Actor	Benutzer
Preconditions	• Die Diagnose wurde gestartet • Es sind Fragen im System hinterlegt
Main Success Scenario	1. Der Benutzer Beantwortet eine Frage mit Ja, Nein oder Weiss nicht 2. Die nächste Frage wird geladen
Postconditions	Die Frage wurde beantwortet und die Nächste anhand der vorherigen Antwort wird geladen.
Alternative Flow	2b) Wenn eine Diagnose gefunden wurde, wird zuerst diese angezeigt, bevor die nächste Frage angezeigt wird. 2c) Wenn keine Frage mehr vorhanden ist, wird der Benutzer darüber informiert und wieder auf die Startseite weitergeleitet. 2d) Der Benutzer kann die Antwort einer bereits beantworteten Frage ändern
Ergänzungen	-

Diagnose An-/Ablehnen	
ID	5
Name	Diagnose An-/Ablehnen
Primary Actor	Benutzer
Preconditions	• Der Benutzer hat genügend Fragen beantwortet, dass eine Diagnose gestellt werden kann.
Main Success Scenario	1. Der Benutzer wertet die Diagnose als Gültig und lässt sich die Details anzeigen.
Postconditions	Die Fragerunde wurde beendet und die Diagnose Details werden geladen
Alternative Flow	1. (b) Der Benutzer lehnt die Diagnose ab und beantwortet weitere Fragen, um die Diagnose zu Verbessern.
Ergänzungen	-

Diagnosedetails einsehen	
ID	6
Name	Diagnosedetails einsehen
Primary Actor	Benutzer
Preconditions	• Die vorgeschlagene Diagnose wurde vom Benutzer angenommen
Main Success Scenario	1. Die Diagnosedetails inkl. der Handlungsempfehlung werden angezeigt.
Postconditions	Die Diagnosesuche wurde beendet.
Alternative Flow	-
Ergänzungen	-

Follow-Up Aktivieren	
ID	7
Name	Follow-Up Aktivieren
Primary Actor	Benutzer
Preconditions	• Der Benutzer befindet sich auf der Seite “Diagnosedetails”
Main Success Scenario	1. Der Benutzer klickt auf den Button “Follow-Up eintragen”.
Postconditions	Follow-Up wird vom System erfasst und gespeichert. System meldet sich per Email beim Benutzer, wenn das Follow-Up gestartet werden soll.
Alternative Flow	1. (b) Der Benutzer ist nicht eingeloggt. 2. (a) Der Benutzer loggt sich ein, damit das Follow-Up eingetragen wird. (b) Der Benutzer registriert sich, damit das Follow-Up eingetragen wird.
Ergänzungen	-

Follow-Up Starten	
ID	8
Name	Follow-Up starten
Primary Actor	Benutzer
Preconditions	• Vom Benutzer wurde zuvor ein Follow-Up eingetragen
Main Success Scenario	1. Auf der Startseite wird vom Benutzer ein durchzuführendes Follow-Up gestartet.
Postconditions	Das Follow-Up wurde gestartet und die erste Frage wird dem Benutzer angezeigt.
Alternative Flow	1b) Der Benutzer löscht das eingetragene Follow-Up 1c) Der Follow-Up ist noch nicht bereit → Der Benutzer kann es trotzdem starten
Ergänzungen	-

Benutzerprofil bearbeiten	
ID	9
Name	Benutzerprofil bearbeiten
Primary Actor	Benutzer
Preconditions	• Der Benutzer hat sich eingeloggt.
Main Success Scenario	1. Die Einstellungen werden aufgerufen 2. Die Daten werden entsprechend geändert. 3. Die Änderungen werden gepspeichert
Postconditions	Die neuen Daten wurden gespeichert
Alternative Flow	2b) Die Daten sind ungültig (z.B. “asdad” als Email-Adresse) → Eine entsprechende Meldung wird angezeigt 2c) Der Benutzer klickt auf “Profil löschen”, die Warnung wird bestätigt und das Profil wird gelöscht
Ergänzungen	-

History ansehen	
ID	10
Name	History ansehen
Primary Actor	Benutzer
Preconditions	• Der Benutzer hat sich eingeloggt • Mindestens ein Diagnosedurchlauf wurde abgeschlossen
Main Success Scenario	1. Der Benutzer öffnet die Einstellungen 2. Der Button “Verlauf” wird angeklickt 3. Ein History-Eintrag wird geöffnet
Postconditions	History-Eintrag wird angezeigt
Alternative Flow	3a) Ein History-Eintrag wird gelöscht
Ergänzungen	-

3.2 Nichtfunktionale Anforderungen

1. **Responsiveness**
   Die Web-App soll so konzipiert werden, dass sie unabhängig vom genutzten Gerät optimal verwendet werden kann. Dies soll anhand von Tests mit Benutzern sichergestellt werden.
2. **Zuverlässigkeit**
   Die Software darf durch Auftreten von Fehlern nicht unbedienbar werden. Es muss immer mindestens die Möglichkeit bestehen, einen fehlerhaften Ablauf abzubrechen und neu zu starten.
3. **Effizienz**
   Auf den grössten Teil der Logik haben wir keinen Zugriff, da dieser sich im bereits implementierten Backendsystem befindet. Allerdings garantieren wir, dass das Frontend nie mehr als 3s für die Verarbeitung der Userinputs oder der erhaltenen Daten in Anspruch nimmt.
4. **Wartbarkeit**
   Die Software soll so aufgebaut sein, dass spätere Anpassungen oder Erweiterungen einfach möglich sind. Insbesondere die Umstellung vom integrierten “Simulator” auf das richtige Backend soll unkompliziert ablaufen.
5. **Übertragbarkeit**
   Die Applikation soll möglichst portabel sein und auf Destop-PCs, Tablets und Smartphones (seit ~2012) laufen. Durch die Wahl, das Frontend mit Webtechnologien umzusetzen, ist es auf jedem modernen Gerät mit Webbrowser nutzbar. Das Hauptaugenmerk wurde auf Chrome 43, Internet Explorer 11 sowie deren Mobile Versionen gelegt.
6. **Benutzbarkeit**
   Die Hauptfunktionen auf jedem Screen sollen ohne Scrolling verfügbar sein. Dazu müssen sich die nötigen Informationen und Aktionen zu jedem Zeitpunkt im Sichtfeld des Benutzers befinden.
7. **Mehrsprachigkeit**
   Die Software soll zu Beginn mindestens Deutsch und Englisch beinhalten, später aber auch mehrere Sprachen ohne grossen Aufwand unterstützen können.

4 Design

4.1 Designentscheide

4.1.1 Technologie

Durch die Vorgaben der Aufgabenstellung war die Auswahl der Technologien nicht bedeutend eingeschränkt, es sollte lediglich in einem Frontend resultieren. Nach Abklärungen mit den Industriepartnern wurde die Anforderung näher spezifiziert. Es soll unter anderem auf mobilen Geräten benutzbar sein. Zur Auswahl standen eine oder mehrere native Apps, welche plattformspezifisch sein würden, eine reine Web-App, welche von jedem modernen Browser unterstützt wird; sowie ein Hybrid wie z.B. PhoneGap (PhoneGap), welches wie eine Web-App entwickelt wird, aber durch ein Tool zu pseudo-nativen Apps portiert wird.

Nach Analysen der Möglichkeiten wurde gemeinsam mit den Projektpartnern die Entscheidung gefällt, das Frontend als Web-App umzusetzen. Gründe dafür waren die grosse Anzahl der zu erreichenden Geräte, welche nicht nur Mobiltelefone sondern auch Tablets und Computer umfassen; die vorherige Erfahrung der Teammitglieder sowie die Möglichkeit, die gleiche Technologie wie beim bereits existierenden Admin-Tool zu verwenden.

4.2 Sequenzdiagramme

4.2.1 Login

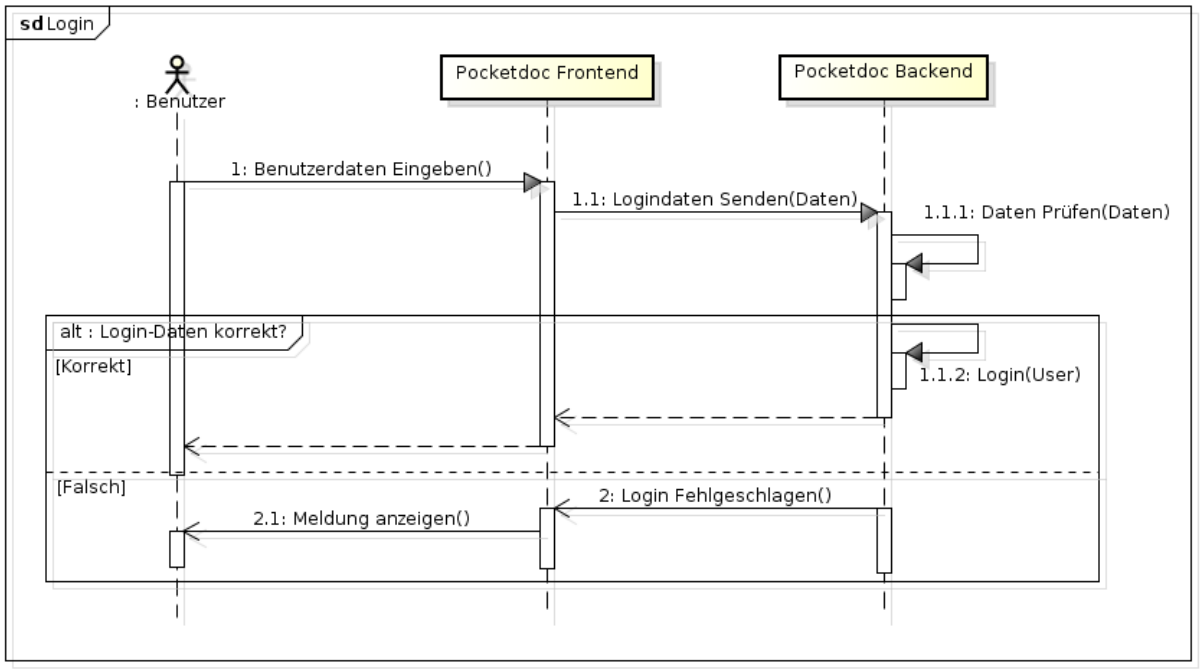


Abbildung 2: Sequenzdiagramm: Login

Das Login läuft ab, indem der Benutzer die Logindaten eingibt und diese an den Server gesendet werden. Wenn die Logindaten gültig sind, wird der Benutzer als eingeloggt markiert, ansonsten wird der Zugriff verweigert. Ein Login ist nicht nötig, um eine Diagnose starten zu können, aber um ein Follow-Up speichern und durchführen sowie auf die eigenen Einstellungen zugreifen zu können.

4.2.2 Registrieren

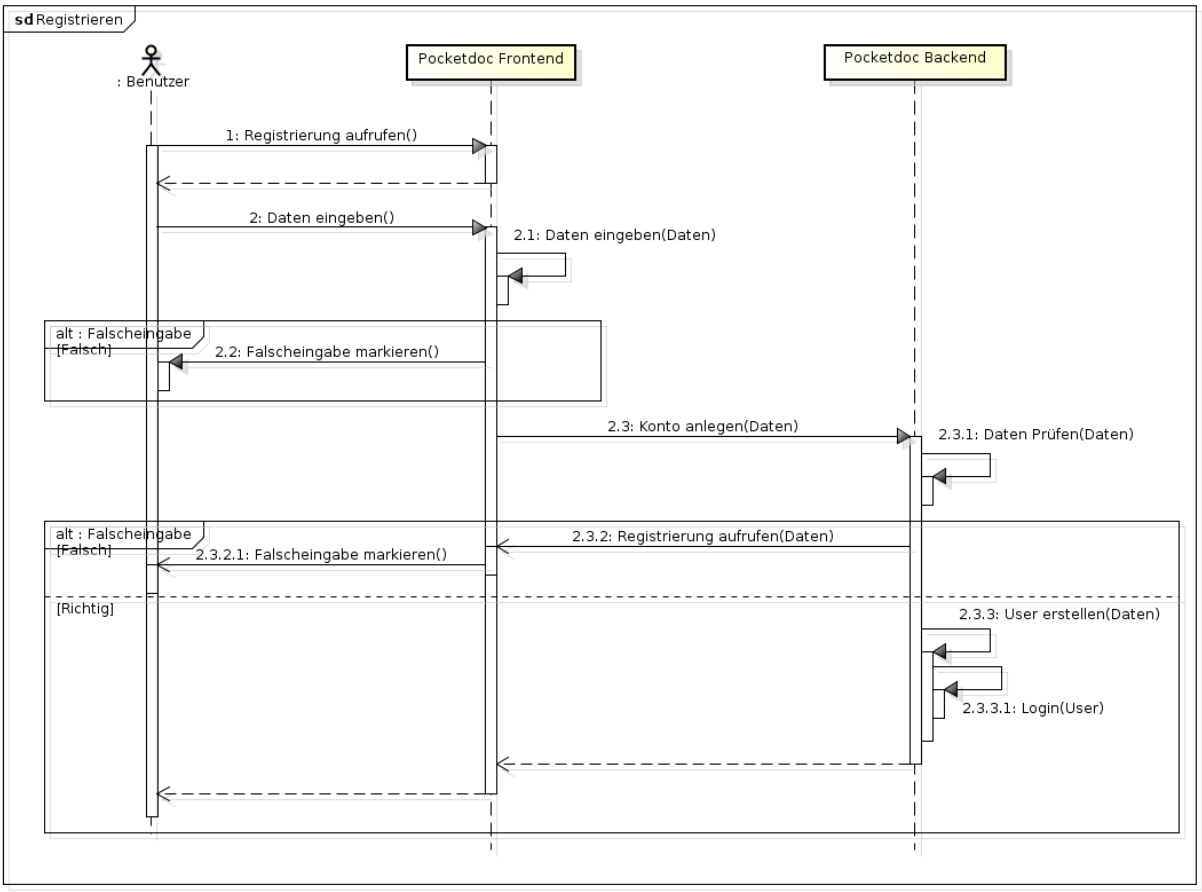


Abbildung 3: Sequenzdiagramm: Registrieren

Der Benutzer kann sich bei der App registrieren, um so persönliche Angaben zu speichern, Follow-Ups durchzuführen oder frühere Diagnosen einzusehen. Dazu muss der Benutzer den Registrationsscreen aufrufen und dort seine Daten (E-Mail, Benutzername, Password, Geschlecht, Sprache, Altersgruppe) angeben. Nachdem er die Eingabe bestätigt hat und diese vom System als gültig erkannt wurden, wird das Konto angelegt und der Benutzer eingeloggt.

4.2.3 Diagnoseablauf

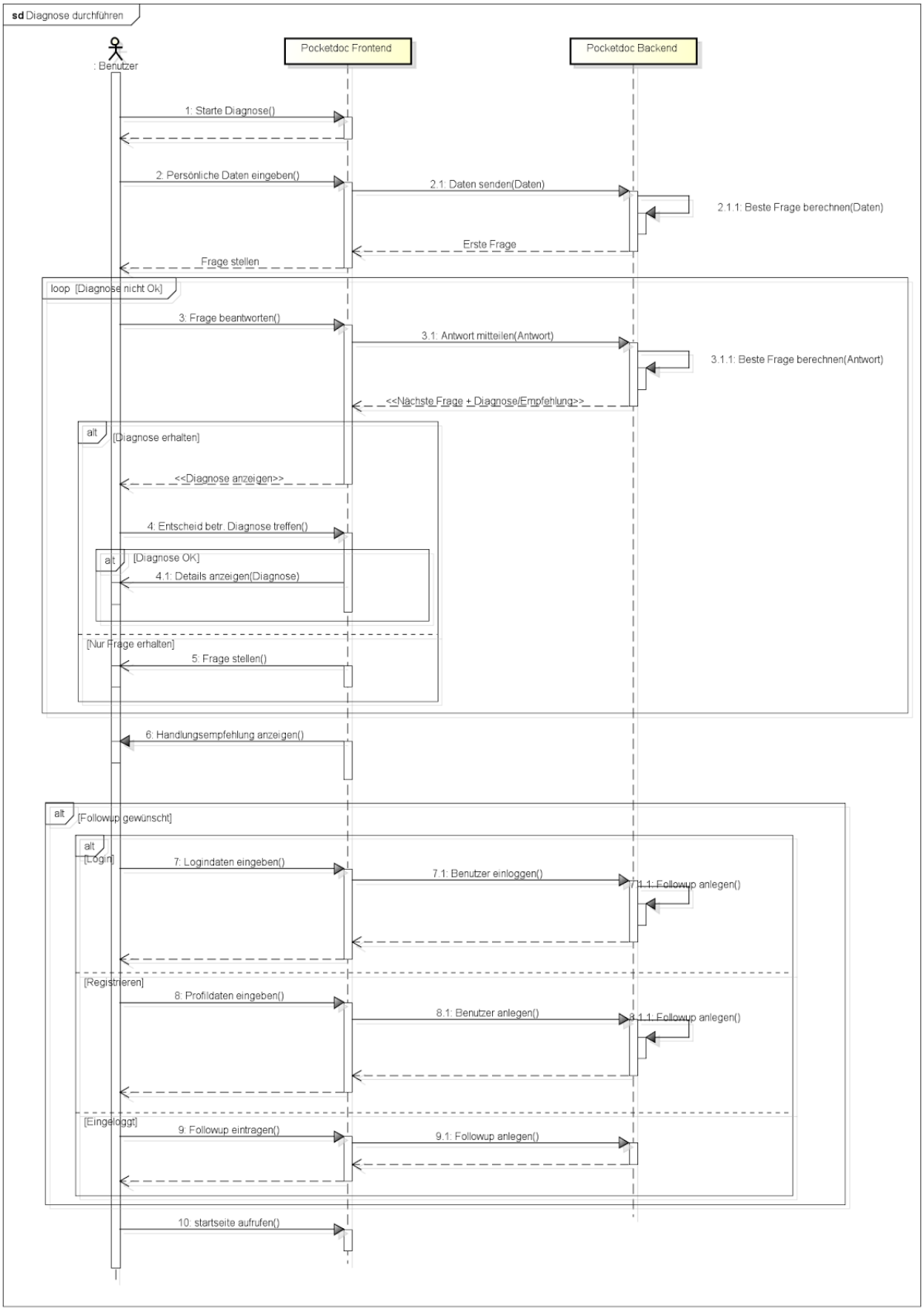


Abbildung 4: Sequenzdiagramm: Diagnoseablauf

Nach dem Start einer neuen Diagnose wird dem Benutzer zuerst eine Eingabemaske angezeigt, welche die Eingabe der persönlichen Daten (Für mich/für Fremdperson,

Geschlecht, Alterskategorie, Sprache) der zu prüfenden Person ermöglicht. Nachdem diese Daten eingegeben und bestätigt wurden, wird die erste Frage angezeigt. Diese sowie die nachfolgenden Fragen werden nach und nach beantwortet, bis das System glaubt, eine Diagnose gefunden zu haben. Lehnt der Benutzer die vorgeschlagene Diagnose ab, werden weitere Fragen zur Verfeinerung des Ergebnisses gestellt. Wird die Diagnose jedoch angenommen, werden auf einem weiteren Screen die detaillierte Diagnose sowie eine Handlungsempfehlung angezeigt.

Hier besteht nun auch die Möglichkeit, einen Follow-Up zu starten. Dadurch wird der Benutzer nach einer festgelegten Zeit per E-Mail erneut aufgefordert, eine Diagnose durchzuführen, um sie mit der aktuellen zu vergleichen. Dieses Feature ist allerdings nur für eingeloggte Nutzer zugänglich. Sollte der Benutzer vergessen haben, sich vor der Diagnose einzuloggen, kann er dies hier noch tun. Ausserdem besteht auch die Möglichkeit, ein neues Konto einzurichten, falls dies noch nicht geschehen ist. Ansonsten ist der Durchlauf für nicht registrierte Benutzer hier zu Ende.

4.2.4 Follow-Up Starten

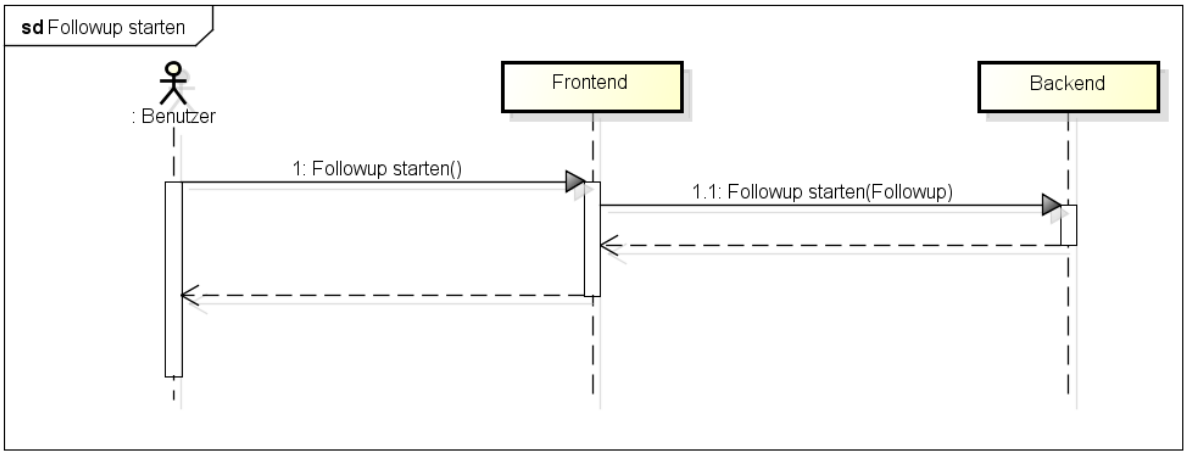


Abbildung 5: Sequenzdiagramm:Follow-Up Starten

Wenn der Benutzer ein Follow-Up angefordert hat, wird dieses auf der Startseite angezeigt. Die Schaltfläche, um das Follow-Up zu starten wird hervorgehoben und mit der Restwartezeit angezeigt. Wenn der Benutzer jedoch das Follow-Up schon früher durchführen möchte, kann er das Follow-Up trotzdem jederzeit starten.

Die Follow-Ups können auch jederzeit aus dem System entfernt werden, falls man sich doch anders entscheidet oder dieses fälschlicherweise eingetragen hat.

4.2.5 Einstellungen ändern

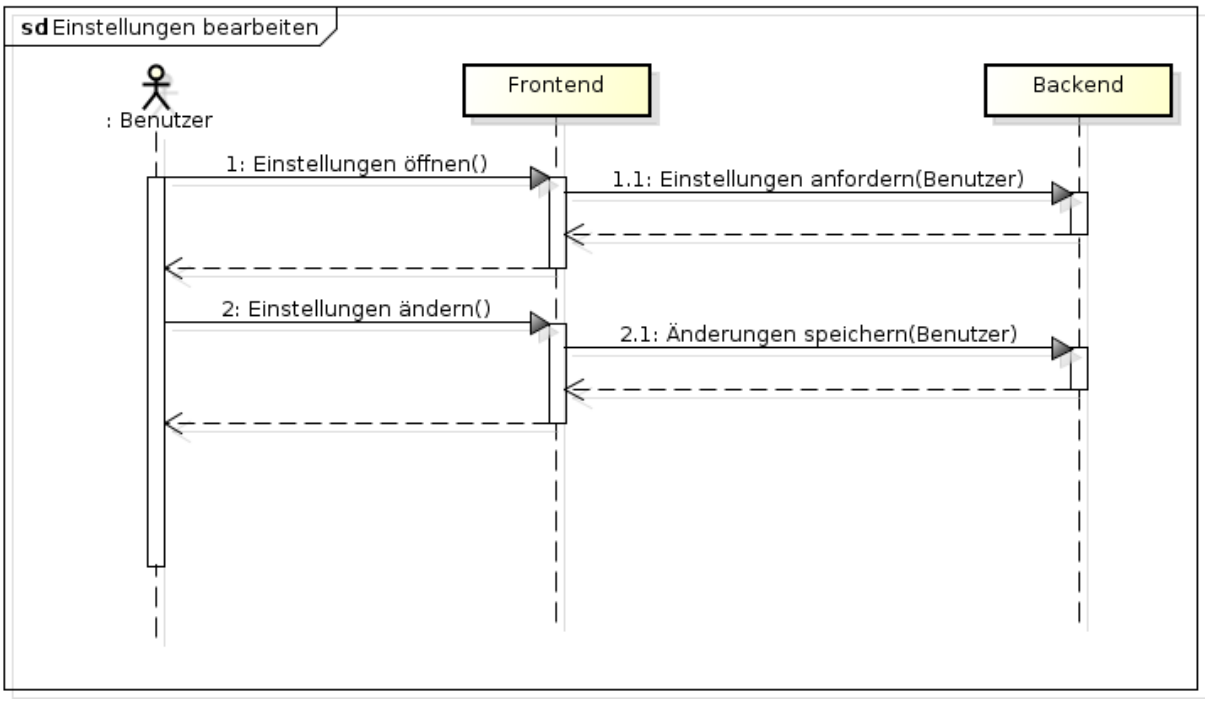


Abbildung 6: Sequenzdiagramm: Einstellungen ändern

Der Benutzer kann seine Einstellungen ansehen und bearbeiten. Dazu muss die Einstellungsseite über das Menu auf der Startseite aufgerufen werden. Hier können die Angaben wie E-Mail-Adresse, Passwort, Geschlecht, Sprache und Alterskategorie geändert werden. Zusätzlich werden Buttons zur Ansicht der History sowie zur Löschung des Accounts bereitgestellt.

4.2.6 History ansehen

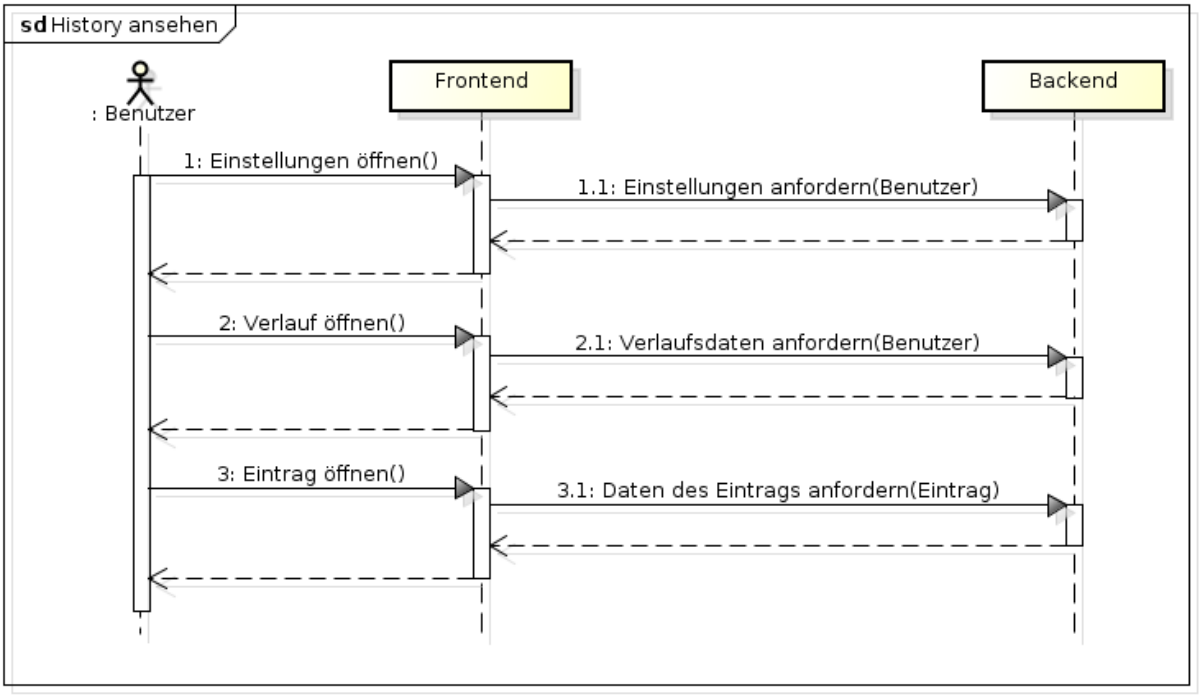


Abbildung 7: Sequenzdiagramm: History ansehen

Es ist dem Benutzer möglich, frühere Diagnosen inkl. der Fragen und den abgegebenen Antworten, einzusehen. Dazu müssen von der Startseite aus die Einstellungen aufgerufen werden. Hier befindet sich nun der Button “Verlauf”, welcher den Nutzer auf die Verlaufsseite weiterleitet. In der Liste ist ersichtlich, welche Diagnosen bisher gestellt wurden und ob diese für sich selbst oder jemand anderen durchgeführt wurden. Durch einen Klick auf einen Eintrag lassen sich Details wie der gesamte Fragenverlauf anzeigen.

Die einzelnen Einträge können auch entfernt werden.

4.2.7 Benutzerkonto löschen

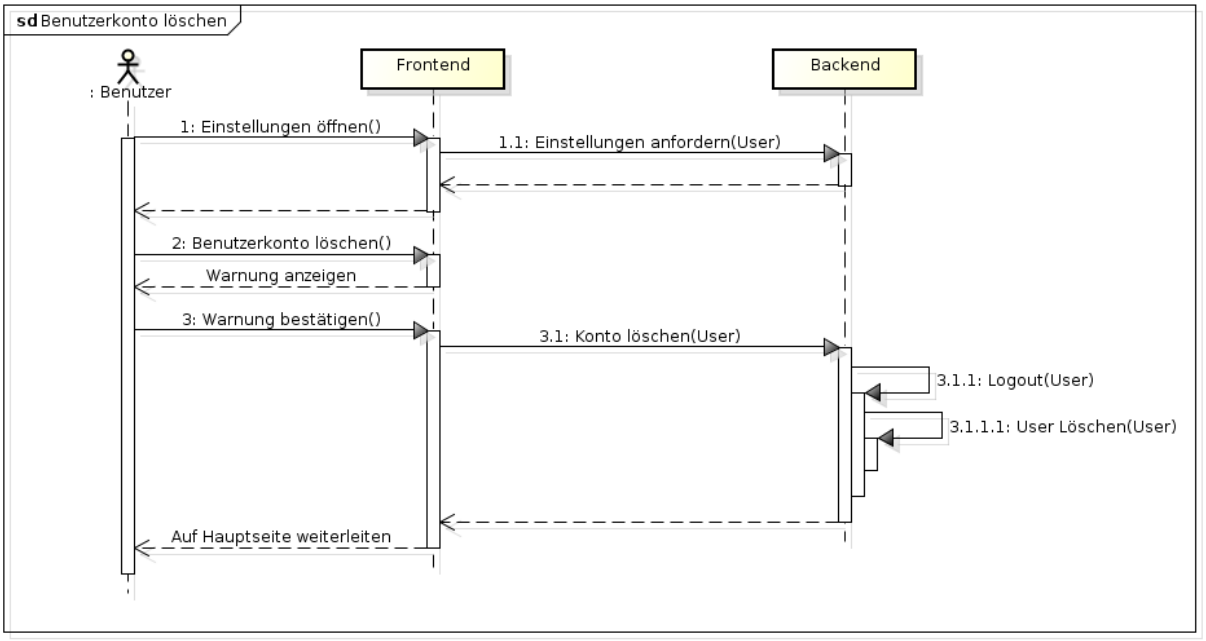


Abbildung 8: Sequenzdiagramm: Benutzerkonto löschen

Sollte der Benutzer sein Profil löschen wollen, müssen die Einstellungen aufgerufen werden, wo sich ein Button zur Löschung des Profils befindet. Wird die Nachfrage bestätigt, so wird das Benutzerkonto restlos entfernt.

4.3 Mockups

Mit Hilfe der Mockups soll erreicht werden, bereits vor der Umsetzung Feedback zur geplanten Gestaltung der Benutzeroberfläche zu erhalten. Dadurch soll Zeit gespart werden, da eine Anpassung des Mockups wesentlich schneller umzusetzen ist, als im fertigen Produkt. Andererseits ermöglicht es allerdings auch das Testen mit potentiellen Benutzern in einem frühen Stadium, um so mögliche Probleme so schnell wie möglich beheben zu können. Besonders wichtig war uns auch, Feedback von Nicht-Entwicklern zu erhalten, da man als Entwickler oftmals entscheidende Details übersieht, da sie einem logisch erscheinen.

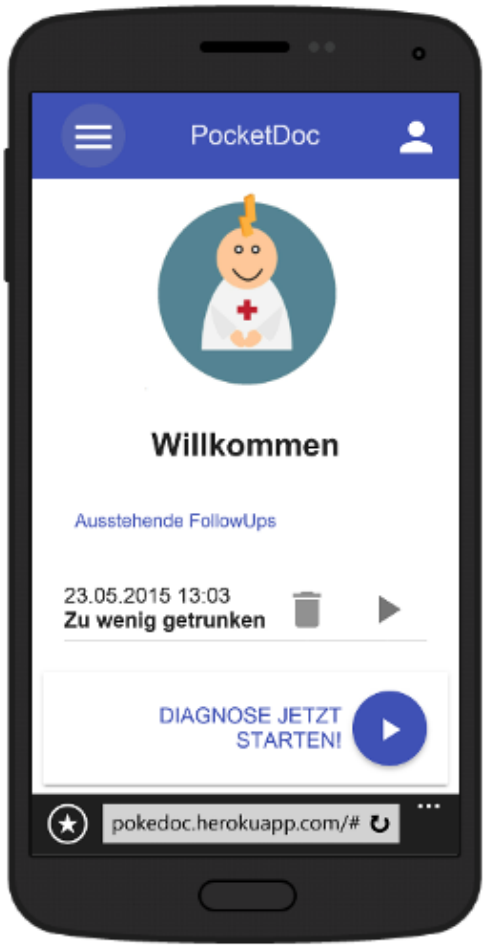
Die Mockups wurden in Balsamiq Mockups (Balsamiq) erstellt und mit MarvelApp (MarvelApp) interaktiv verlinkt und online bereitgestellt.

Nachfolgend werden die Mockups aller wichtigen Screens beschrieben und in Vergleich mit der letztendlichen Umsetzung gestellt.

4.3.1 Startseite (Nicht eingeloggt)

Mockup	Umsetzung
 *Abbildung 9: Mockup: Startseite*	 *Abbildung 10: Implementation: Startseite*
Beschreibung	**Änderungen in der Umsetzung**
Ziel dieses Screens ist es, neuen Usern die Applikation kurz zu erklären. Da das Durchführen einer Diagnose auch ohne Registrierung erfolgen kann, ist der “Diagnose starten”-Button zentral und sollte immer sichtbar sein. Auch die Möglichkeit, sich zu registrieren oder einzuloggen sollte hier immer sichtbar sein. Der grosse, gut sichtbare Titel dient dazu, den Namen der Applikation vorzustellen und dem User klar zu machen, wo er sich befindet. Das Icon/Maskottchen ist simpel, spielerisch und soll den Bezug zum Namen (Doktor <-> Doc/Dok) sowie die Verbindung zum Zweck der Applikation herstellen (Diagnose, Medizinisches Umfeld, aber nicht zu seriös/trocken).	In der Titelzeile der App befindet sich nun der Titel sowie anstelle der Sprachauswahl ein “Hamburger”-Menu. In diesem Menu befinden sich Links auf die About- und AGB-Seite sowie die Sprachauswahl. Zusätzlich befindet sich hier ein Reset-Button, welcher den Local Storage zurücksetzt. Dieser ist allerdings nur in der Version mit Simulator vorhanden.

4.3.2 Startseite (Eingeloggt)

Mockup	Umsetzung
	
Abbildung 11: Mockup: Eingeloggt	*Abbildung 12: Implementation: Eingeloggt*
Beschreibung	**Änderungen in der Umsetzung**
Im Gegensatz zu neuen Usern wird hier davon ausgegangen, dass registrierte User sich bereits über den Zweck der Applikation bewusst sind. Deswegen ist hier die Aufgabe des Screens, einen Überblick über die vorherigen (offenen) Durchgänge sowie ausstehende Follow-Ups zu bieten. Wie fast immer kann man auf diesem Screen die Sprache wechseln und über das Account-Symbol oben rechts auf die persönlichen Einstellungen zugreifen oder diese bearbeiten. Falls der User weder einen offenen Durchgang beenden noch ein Follow-Up durchführen will, kann auch eine neue Diagnose gestartet werden.	Die offenen Durchgänge wurden auf Kosten des Simulators nicht umgesetzt. Bei den Follow-Ups fällt der Link, um weitere anzuzeigen weg und es werden immer alle angezeigt. Die geplanten Follow-Ups werden zwar weiterhin als solche angezeigt, müssen zum Starten allerdings nicht mehr erst aktiviert werden. Anstelle der Sprachauswahl wird hier wie im nicht eingeloggten Zustand das Hamburger-Menu angezeigt mit einer Ausnahme: Die Sprachauswahl ist nicht mehr enthalten. Diese muss nun über die Profileinstellungen vorgenommen werden.

4.3.3 Fragen

Mockup	Umsetzung

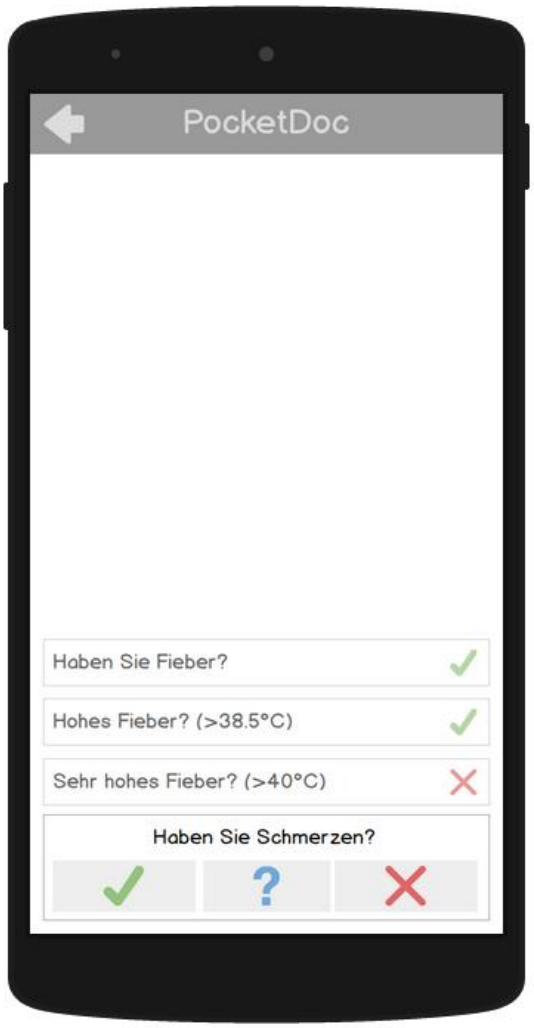
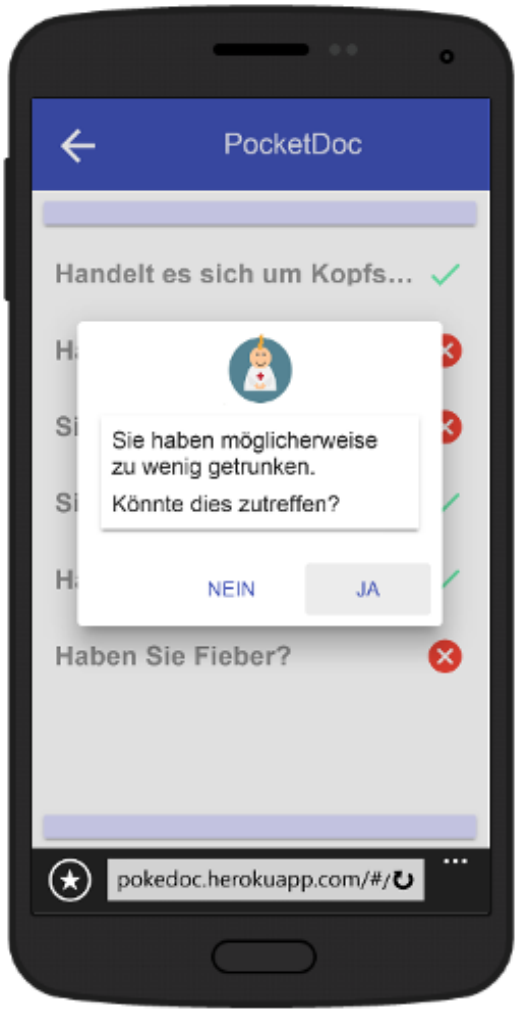



Abbildung 13: Mockup: Fragen	*Abbildung 14: Implementation: Fragen*
Beschreibung	**Änderungen in der Umsetzung**
Die Hauptaufgabe auf diesem Screen ist es, verschieden Fragen zum aktuellen Zustand zu beantworten, so dass nach genügend Informationen eine Diagnose gestellt werden kann. Dies wird in einem sich immer wiederholenden Ablauf durchgeführt: Eine Frage wird angezeigt, der Benutzer wählt die passendste Antwort und klickt sie, die Frage wird als erledigt gekennzeichnet und in die Liste der bereits beantworteten Fragen hinzugefügt, worauf eine neue Frage erscheint, sollte keine Diagnose gefunden worden sein.	Die grösste Änderung gegenüber dem Mockup ist die neue Position der aktuellen Frage. Diese befindet sich nun am oberen Rand des Bildschirms. Die Antwortbuttons befinden sich allerdings weiterhin unten. Dadurch hat sich auch die Reihenfolge der bereits beantworteten Fragen geändert: Die zuletzt beantwortete Frage befindet sich nun ganz oben.

Die Fragen können mit “Ja”, “Nein” oder “Weiss nicht” beantwortet werden, weshalb diese Texte durch Icons und entsprechende Farbgebung ersetzt wurden. Dies garantiert, dass unabhängig von der Sprache und der daraus folgenden Länge der Antworten die Position und Grösse der Buttons gleich bleiben wird. Die bereits beantworteten Fragen werden fortlaufend nach oben geschoben, bis sie hinter dem Header verschwinden. So ist die aktuelle Frage immer zuunterst und die Position der Antworten verändert sich nicht.	

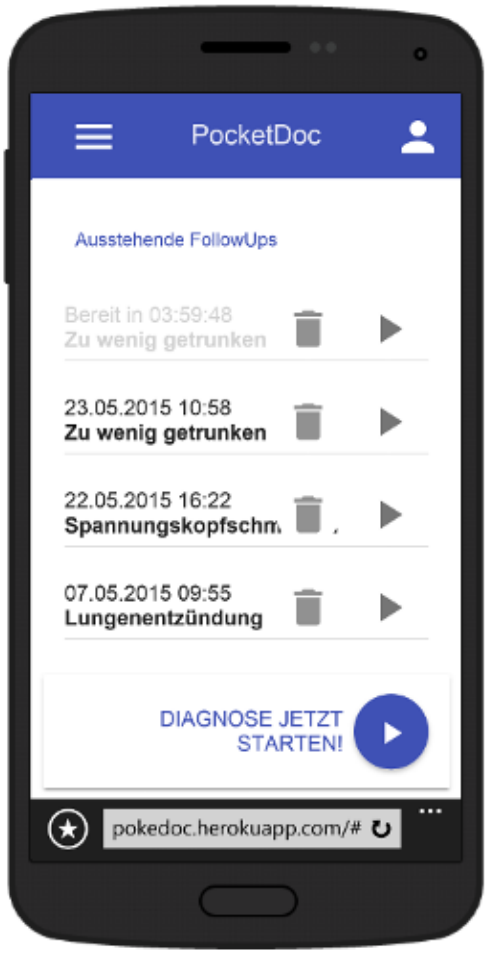
4.3.4 Diagnosenvorschlag

Mockup	Umsetzung
 *Abbildung 15: Mockup: Diagnosenvorschlag*	 *Abbildung 16: Implementation: Diagnosenvorschlag*
Beschreibung	**Änderungen in der Umsetzung**
Sobald die Applikation glaubt, die Krankheit gefunden zu haben, wird die Diagnose dazu angezeigt. Hier hat der Benutzer die Möglichkeit, diese anzunehmen oder abzulehnen. Wird sie angenommen, wird er auf die Detailansicht inkl. Handlungsempfehlung weitergeleitet. Glaub der Nutzer allerdings, dass er eine andere Krankheit hat, kann er die Diagnose ablehnen und weitere Fragen beantworten.	Abgesehen von der Formulierung hat sich nur eine Sache an der Anzeige geändert: Neu wird das PocketDoc-Icon angezeigt, um den Eindruck zu erwecken, dass es zum Benutzer spricht. Dadurch soll die Diagnosestellung persönlicher wirken.

4.3.5 Diagnose einsehen

Mockup	Umsetzung
 *Abbildung 17: Mockup: Diagnose*	 *Abbildung 18: Implementation: Diagnose*
Beschreibung	**Änderungen in der Umsetzung**
Nachdem der Diagnosedurchlauf abgeschlossen wurde, wird die Diagnose angezeigt. Zusätzlich wird eine Handlungsempfehlung ausgegeben, welche dem Nutzer Tipps zur Behandlung der Krankheit gibt oder ob er gegebenenfalls sogar einen Arzt aufsuchen soll. Sollte der Benutzer das Bedürfnis haben, seine Diagnose zu einem späteren Zeitpunkt nochmals zu prüfen, kann er ein Follow-Up registrieren. Ist er bereits eingeloggt, kann dies durch einen einfachen Klick erfolgen. Falls nicht hat er hier die Möglichkeit, sich einzuloggen oder bei Bedarf auch neu Anzumelden. Das Follow-Up wird danach automatisch eingetragen.	Die Diagnose, welche im Mockup noch sehr Prominent angezeigt wurde, wird nun auf dieselbe Art und Weise wie die Handlungsempfehlung dargestellt. Das Login Formular sowie die Buttons wichen einem einfachen Ja und Nein Button, unabhängig davon, ob man eingeloggt ist oder nicht. Bei einem Klick auf ja wird das Follow-Up in eingeloggtem Zustand eingetragen und man wird, wie bei einem Klick auf Nein auf die Startseite weitergeleitet. Falls man nicht eingeloggt ist, erscheint nun ein Popup mit Login-Maske sowie einem Registrationsbutton. Die Funktionalität ist allerdings wie beim geplanten Mockup dieselbe.

4.3.6 Follow-Ups

Mockup	Umsetzung
 *Abbildung 19: Mockup: Follow-Ups*	 *Abbildung 20: Implementation: Follow-Ups*
Beschreibung	**Änderungen in der Umsetzung**
Ein Follow-Up entspricht beim Arzt der Folgeuntersuchung, also einer Kontrolle nach der Diagnose. Diese soll 4 Stunden nach dem Einsehen der Diagnose geschehen und den Benutzer via Email informieren. Wenn ein (oder mehrere) Follow-Ups anstehen, werden diese auf der Startseite nach dem Login angezeigt. Ziel soll sein, dass der Follow-Up erst nach 4 Stunden durchgeführt wird, aber in dringenden Fällen kann er auch sofort gestartet werden. Dazu dient der “Aktivieren”-Link, welcher die Schaltfläche zur Durchführung freischaltet. Da es auch möglich ist, weitere Diagnosen für sich selbst oder Fremdpersonen	Der Link “Weitere Follow-Ups anzeigen” wurde entfernt. Stattdessen werden nun immer alle Follow-Ups angezeigt. Noch geplante Follow-Ups werden nun weiterhin als geplant angezeigt, können allerdings ohne Umwege direkt gestartet werden.

durchzuführen, muss es aber immer noch möglich sein, diese zu starten. Dazu wird bei einem ausstehenden Follow-Up der Text für das Starten einer Diagnose leicht abgeändert, um keine Verwirrung aufkommen zu lassen. Um den Haupt-Screen nicht zu überladen, werden nur die 3 neusten Follow-Ups (und offenen Durchgänge) angezeigt. Weitere Follow-Ups oder Durchgänge können bei Bedarf manuell über einen Link (“Weitere Follow-Ups anzeigen”) nachgeladen werden.	

4.3.7 Profil/Einstellungen

Mockup	Umsetzung

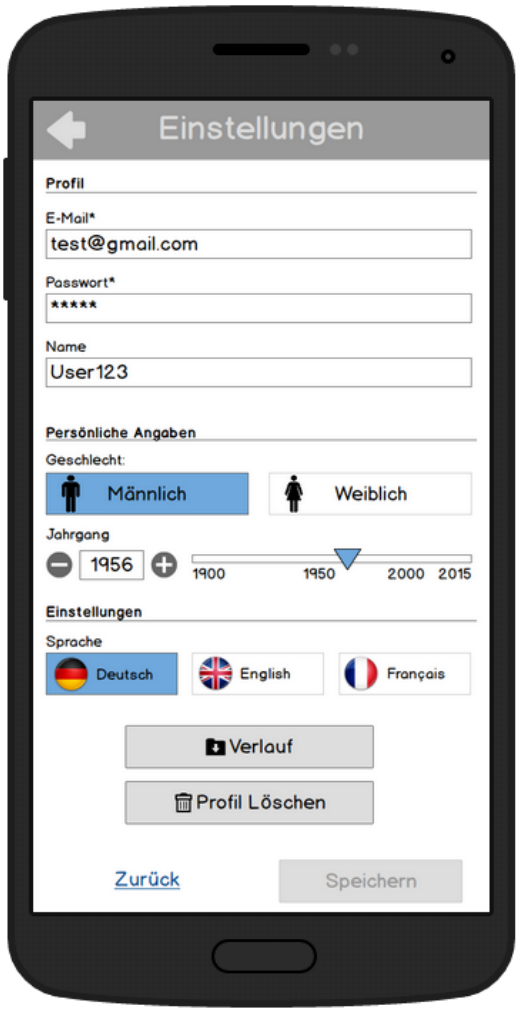
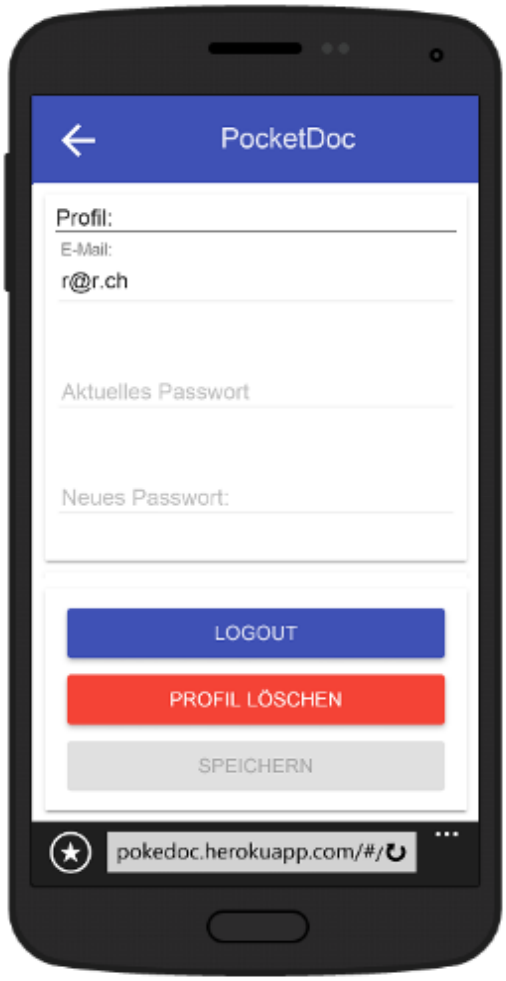

	

Abbildung 21: Mockup: Einstellungen	*Abbildung 22: Implementation: Einstellungen*
Beschreibung	**Änderungen in der Umsetzung**
Die Einstellungen dienen dazu, Präferenzen des Users bezüglich der Applikation sowie einige Daten zu seiner Person festzuhalten. Für den Account zwingend sind eine E-Mail-Adresse sowie ein Passwort, um den Benutzer eindeutig zu identifizieren. Ein Name ist nur für die persönliche Anrede nötig, damit eine Bindung zum Benutzer aufgebaut werden kann. Das Geschlecht und der Jahrgang dienen dazu, die Diagnose zu tätigen. Gemeinsam mit der Sprache werden diese Optionen vor jedem Durchgang angezeigt und müssen vom User bestätigt werden. Falls sie hier	Aus sicherheitstechnischen Gründen wird das Passwort hier nicht mehr angezeigt. Es ist allerdings möglich, ein neues Passwort zu setzen. Dazu muss, wie bei der Änderung der E-Mail Adresse, in einem separaten Feld das aktuelle Passwort eingegeben werden. Anstelle des Jahrgangs wird nun nur noch eine Altersgruppe (0-10, 11-25, …) verlangt. Der Zurückbutton am unteren Rand wurde aufgrund der Redundanz mit dem Standard-Zurückbutton oben links entfernt.

angegeben wurden, werden die entsprechenden Felder vor dem Durchgang jeweils bereits ausgefüllt und der Benutzer muss nur das OK geben. Ansonsten muss diese Information dort jedes Mal angegeben werden, da sie für die Diagnose zwingend benötigt werden. Um nicht zu viele Informationen über den potentiell sicherheitsbewussten User zu sammeln, wird statt dem Geburtsdatum nur ein Altersbereich (<12 Jahre, 12-24 Jahre, …) verlangt, da dies für die Verwendung der Applikation ausreicht. Die Spracheinstellung bestimmt die Default-Sprache für diesen User. Hier kann auch der bisherige Verlauf eingesehen (Siehe Verlauf) oder der eigene Account gelöscht werden. Dieser Screen ist nur für eingeloggte Benutzer sichtbar.	Der Verlauf ist, wie bereits früher im Bericht erwähnt, aufgrund des Simulators entfernt worden.

4.3.8 Verlauf

Mockup

Abbildung 23: Mockup: Verlauf 1

Abbildung 24: Mockup: Verlauf 2

Beschreibung

Um einen Überblick über seine persönliche Krankengeschichte zu erhalten kann über das Profil der Diagnose-Verlauf eingesehen werden. Hier werden abgeschlossene Diagnosen abgespeichert und verwaltet, mit einer Hervorhebung für Diagnosen von Fremdpersonen. Dieser Screen ist nur für eingeloggte Benutzer sichtbar, da für anonyme Benutzer keine Daten gespeichert werden.

Wird ein abgeschlossener Durchlauf ausgewählt, wird dieser geöffnet. Dieser sieht sehr ähnlich wie die normale Diagnose aus, bietet aber keine Möglichkeit zur Interaktion. Zudem werden noch einige Metadaten angezeigt, etwa die Voreinstellungen (für wen wurde diese Diagnose durchgeführt?) sowie angenommene und abgelehnte Diagnosen (mit dem Glühbirnen-Symbol). Diese Fragen können nicht erneut beantwortet werden, sondern dienen nur der Auskunft.

Abgeschlossene Durchgänge können natürlich auch gelöscht werden, etwa wenn sie eine komplett falsche Diagnose lieferten oder für eine Fremdperson bestimmt waren, welche nicht mehr relevant zum Besitzer des Accounts ist.

5 Umsetzung

5.1 Entwicklungsumgebung und Tools

Da sich das alte Backend bereits auf Heroku (Her) befand, bot es sich an, auch das Frontend dort zu hosten. Als Code-Repository wurde git (git) auf GitHub (GitHub) verwendet, da Heroku hier Hooks anbietet und nach jedem Commit den Code neu holt und re-deployt. So konnte der Deploymentprozess vollständig automatisiert und die Zeit vom Commit einer Änderung bis zum Erscheinen auf dem öffentlichen Heroku-Space auf ca. 1-2 Minuten verkürzt werden.

Lokal wurden der Node Package Manager (NPM15) mit Bower (Bower) auf einem Node-Server (NodeJS) verwendet, um benötigte Pakete automatisch herunterzuladen und Abhängigkeiten aufzulösen. Die benötigten Pakete sind teilweise durch die gewählte Technologie und die geforderten Funktionalitäten gegeben (z.B. Angular, Angular-Material, Angular-Translate (angTransl), …), andere bieten einen Mehrwert für die Entwicklung (Underscore.JS (UnderscoreJS), express.js (expressJS)) oder beim Testing (Karma,JS (KarmaJS), Protractor (Protractor), Angular-Mocks, …).

5.1.1 Workflow

Entwickelt wurde lokal mit einem lokalen express.JS-Server, unabhängig von der IDE. Im Team wurden hauptsächlich die Texteditoren/IDEs Notepad++ und Sublime Text 3 verwendet, es spielt grundsätzlich aber keine Rolle. Mit git wurden die letzten Änderungen von GitHub lokal kopiert und eigene, neue Änderungen wieder auf das Repository gepusht. Heroku registriert Änderungen am Source-Tree automatisch und holt sich die neuesten Files, bildet das Paket und veröffentlicht sie.

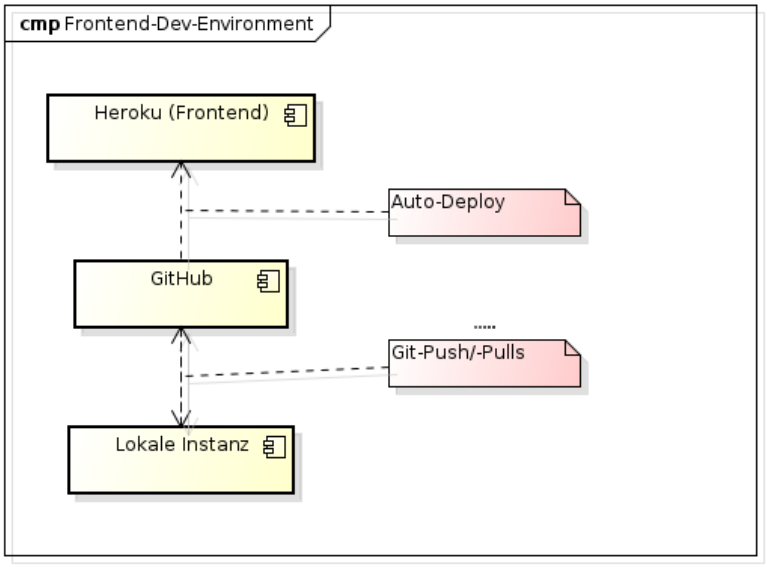


Abbildung 25: Workflow Codeverwaltung

5.1.2 Testing

Es wurden von Beginn an Unit-Tests eingeplant und das Framework dazu bereitgestellt (primär Karma.JS, welches Unit-Tests in beliebigen Browsern startet).

Für Integrationstests wurde in Zusammenarbeit mit den Projektpartnern ein Demo-Skript erstellt, welches auch an Präsentationen für potentielle Geschäftspartner durchgespielt werden sollen. Daher konnte nach jeder Iteration getestet werden, ob dieses Skript, welche bereits die Mehrheit der Features abdeckt, auch ohne Probleme durchgespielt werden kann. Das gleiche Skript wurde auch verwendet, um mit Laien Usertests durchzuführen.

5.2 Simuliertes Backend

Da das Backend nicht die benötigten Anforderungen erfüllt, wird das Frontend nur mit Testdaten laufen. Dazu wird das Backend im Frontend simuliert. Dies wird so bewerkstelligt, dass später das Austauschen der Simulation und dem normalen Backend problemlos möglich ist. Um dies zu erreichen, werden Schnittstellen definiert, welche sowohl vom simulierten Backend als auch später von den REST-Aufrufen genutzt werden.

5.2.1 Testdaten

Die Testdaten entsprechen vom Umfang her den Daten, welche vom Frontend benötigt werden plus denjenigen, die für die Simulation des Backends nötig sind. Da das simulierte Backend allerdings nur eine Baumstruktur für die Fragen verwendet anstelle eines komplexen Berechnungsalgorithmus, sind weniger Daten als beim originalen Backend nötig.

<table>
<tr><th colspan="2">Fragen</th></tr>
<tr><td colspan="2">

```
{
   "id": <Number>,
   "type": <String>,
   "description": [
      {
         "lang": <Number>,
         "text": <String>"
      }
   ],
   "answers": [
      {
         "id": <Number>,
         "desc":
         [
            {
               "lang": <Number>,
               "text": <String>
            }
         ],
         "next_questions": [<Number>],
         "diagnosis": <Number>,
         "action_suggestion": <Number>
      }
   ]
```

}
</td></tr>
<tr><td>type</td><td>Typ der Frage (yesnomaybe für Ja/Nein/Weiss Nicht oder ein Sondertyp. Es wird momentan aber nur Ja/Nein/Weiss Nicht verwendet.)</td></tr>
<tr><td>description</td><td>Frage-Beschreibung in den verschiedenen Sprachen</td></tr>
<tr><td>lang</td><td>ID der Sprache</td></tr>
<tr><td>text</td><td>Frage in der jeweiligen Sprache</td></tr>
<tr><td>next_questions</td><td>IDs von Fragen, die gestellt werden können, wenn diese Antwort ausgewählt wurde. Beinhaltet in den Testdaten nur eine weitere Frage.</td></tr>
</table>

diagnosis	ID einer Diagnose, die durch das Geben einer bestimmten Antwort erreicht wird.
action_suggestion	ID einer Handlungsempfehlung, die durch das Geben einer bestimmten Antwort erreicht wird.

Diagnosen

```
{
   "id": <Number>,
   "short_desc":[
      {
         "lang": <Number>,
         "text": <String>
      }
   ]
   "description":[
      {
         "lang": <Number>,
         "text": <String>
      }
   ]
}
```

short_desc	Kurzdiagnose in den jeweiligen Sprachen.
description	Beschreibung der Diagnose in den jeweiligen Sprachen.
lang	ID der Sprache
text	Diagnose in der jeweiligen Sprache

Handlungsempfehlungen

```
{
   "id": <Number>,
   "description":
      {
         "lang": <Number>,
         "text": <String>
      }
   ]
}
```

description	Beschreibung der Handlungsempfehlung in der jeweiligen Sprache
lang	ID der Sprache
text	Handlungsempfehlung in der jeweiligen Sprache

Sprachen	

{
"id": <Number>,
"locale": <String>
}	

locale	Kurzschreibweise der Sprache, welche z.B. für das Laden von Icons verwendet wird. (en, de, fr, …)

Benutzer	

{
"id": <Number>,
"name": <String>,
"email": <String>,
"password": <String>,
"gender": <Number>,
"age_category": <Number>,
"lang": <Number>
}	

name	Name des Benutzers
email	E-Mail Adresse des Benutzers
password	Password
gender	ID des Geschlechts (0 = Männlich, 1 = Weiblich)
age_category	ID der Alterskategorie (0-10, 11-25, ...)
lang	ID der Sprache

History	

<pre>{
"id": <Number>,
"user_id": <Number>,
"timestamp": <String>,
"patient":{
"self": <Boolean>,
"name": <String, optional>,
"age_category": <Number, optional>,
"gender": <Number, optional>,
"lang": <Number, optional>
},
"diagnosis":{
"id": <Number>,
"description":[
{
"lang": <Number>,
"text": <String>
}
]
},
"content":[
{
"type": <String>,
"id": <Number>,
"answer": <Number, optional>,
"accepted": <Boolean, optional>
}
]
}</pre>	

user_id	ID des Benutzers, dem der History-Eintrag gehört
timestamp	Zeitstempel, wann die Diagnose durchgeführt wurde.
patient	Infos über die betroffene Person. Wenn es die gleiche Person war, welcher der History-Eintrag gehört, kann “self” auf true gesetzt werden. Andernfalls müssen die Einträge zur Fremdperson (Name, Geschlecht, Alterskategorie und Sprache) abgelegt sein.
content	Array mit den gestellten Fragen, den gefundenen Diagnosen und Handlungsempfehlungen.
content[i].type	question, diagnosis oder actionsuggestion. Dient zum Nachbilden der damaligen Diagnose, so dass ersichtlich wird, welche Frage wann erschien, welche Antworten gegeben und welche Diagnosen gefunden wurden.
content[i].id	ID der entsprechenden Frage, Diagnose oder Handlungsempfehlung.
content[i].answer	Optional. ID der Antwort, welche auf eine Frage gegeben wurde.

content[i].accepted	Optional. Beschreibt, ob die gefundene Diagnose angenommen (true) oder abgelehnt (false) wurde.

5.3 Schnittstellen

Das Frontend kommuniziert mit dem Backend über festgelegte Schnittstellen. Diese werden sowohl vom normalen als auch vom simulierten Backend implementiert, was die Nutzung des Frontends unabhängig vom vorhandenen Backend macht. Für jeden Teilbereich (Userhandling, Diagnosedurchlauf, Followu, etc.) wird ein eigener Service mit den nötigen Funktionen implementiert. Die Funktionen entsprechen alle demselbem Schema, welches folgendermassen aussieht:

Service		
Beschreibung des Services		
Funktion(data, success(data), errror(error))		
Funktionsbeschreibung		
Daten, die die Funktion für die korrekte Ausführung benötigt.	*Parameter-Daten, welche an die success() Funktion nach erfolgreicher Ausführung übergeben werden.*	*Parameter-Daten, welche nach einem Fehler an die error() Funktion übergeben werden.*

RunService

Der Run-Service stellt Funktionen für einen Diagnosedurchlauf bereit.

UserService

Dieser Service stellt alle Funktionen, die für die Benutzerverwaltung nötig sind, zur Verfügung.

FollowupService

Dieser Service dient als Schnittstelle für alle Funktionen, die mit dem Follow-Up zusammenhängen.

HistoryService

Dieser Service dient als Schnittstelle für alle Funktionen, die mit der History eines Benutzers interagieren.

DiagnosisService

Beinhaltet Funktionen, welche mit Diagnosen in Verbindung stehen.

MetaDataService

Einstellungen, welche im Backend verwaltet werden können, werden über Funktionen in diesem Service abgerufen. Primär handelt es sich um die Sprachen und Alterskatergorien.

5.3.1 Implementierung / Architektur

Die Architektur von PocketDoc ist folgendermassen aufgebaut:

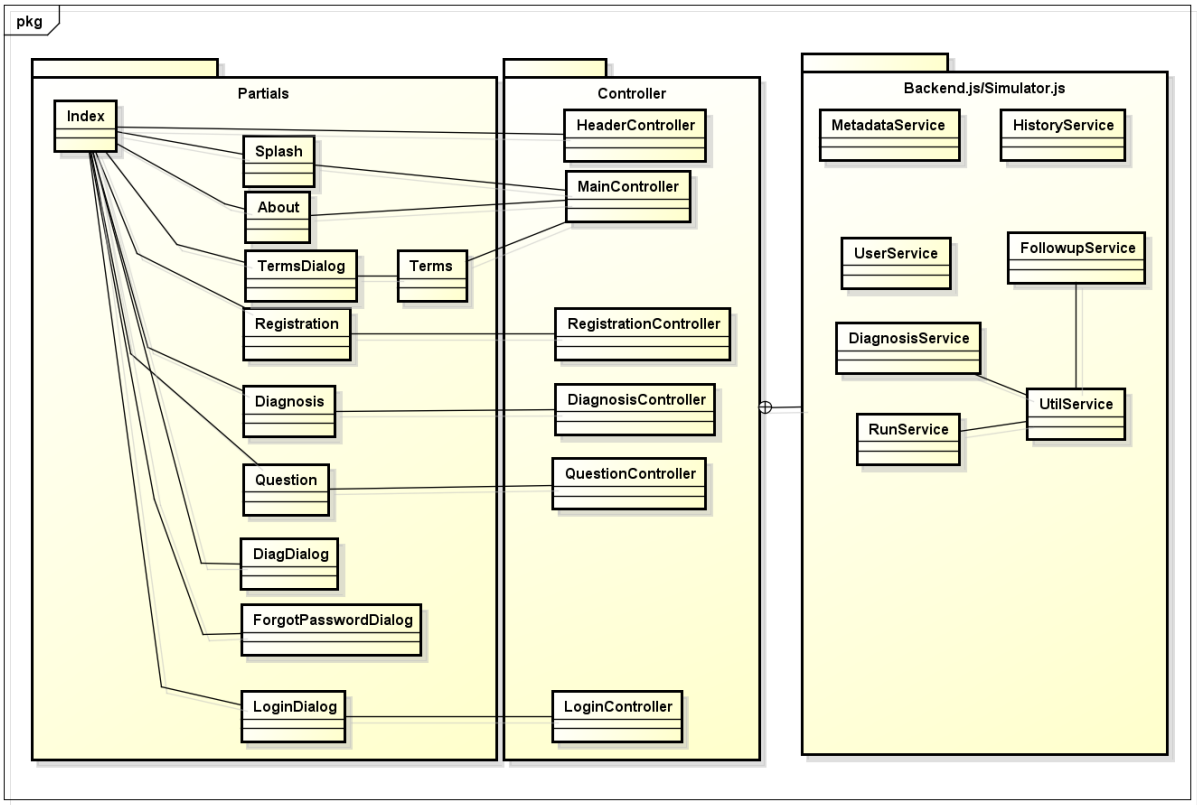


Abbildung 26: Projektstruktur

Nachfolgend werden die einzelnen Komponenten beschrieben.

Seite/View	
Beschreibung und Funktionsumfang	
Controller	**Partials**
Genutzte Controller	*Genutzte Partials*
Services	**Factories**
Genutzte Services	*Genutzte Factories*
URL zu diesem Teil der Webapp.	

Index	
Dieses Partial übernimmt eine Sonderrolle. Es wird beim Aufruf der Website automatisch geladen und dient der Website als Rahmen. Es stellt Standardelemente wie die Headerzeile und die Sidebar zur Verfügung. Diese enthalten Funktionen wie Login, Links zu anderen Teilen der Seite sowie ein Zurückbutton. Alle anderen Partials werden bei ihrem Aufruf jeweils in dieses Partial geladen.	
Controller	**Partials**
• HeaderController	• Alle
Services	**Factories**
• UserService	
n/A	

Splash	
Diese View enthält den Inhalt der Startseite. Standardmässig werden hier der Einleitungstext sowie ein Button zum Start einer Diagnose angezeigt. Sollte ein Benutzer eingeloggt sein, werden anstelle des Einleitungstexts die offenen Diagnosen und Follow-Ups angezeigt.	
Controller	**Partials**
• MainController	• splash.html
Services	**Factories**
• UserService • FollowupService	
/	

Registration	
Über dieses Partial wird eine Eingabemaske für die Registrierung angezeigt. Hier kann der Benutzer E-Mail, Passwort, Name und weitere Daten eingeben und ein Benutzerprofil anlegen. Sollte bereits ein Benutzer eingeloggt sein, wird dieses Template auch zur Anzeige und Bearbeitung der persönlichen Daten verwendet. Zusätzlich bestehen dann die Möglichkeiten, von hier aus die History aufzurufen oder das eigene Konto zu löschen. Wichtige Informationen wie E-Mail Adresse und Passwort können nur geändert werden, wenn das alte Passwort angegeben wird.	
Controller	**Partials**
• RegistrationController	• registration.html
Services	**Factories**
• UserService	
• /registration (nicht eingeloggt) • /profile (eingeloggt)	

Run	
Dieses Partial bietet das Template für einen Diagnoseablauf. Hier werden die Fragen und Antwortmöglichkeiten angezeigt. Jede beantwortete Frage wird in einer Liste angezeigt. Über diese Liste lassen sich Antworten nachträglich auch ändern.	
Controller	**Partials**
• QuestionController	• diagDialog.html • questions.html
Services	**Factories**
• DiagnosisService • RunService	
/run	

Diagnosis	
Hier werden nach einem Diagnoseablauf die Diagnose sowie Handlungsempfehlung angezeigt. Zusätzlich hat der Benutzer die Möglichkeit, ein Follow-Up zu registrieren, sofern er eingeloggt ist.	
Controller	**Partials**
• DiagnosisController	• diagnosis.html
Services	**Factories**
• UserService • FollowupSerivce	
/diagnosis	

About	
Enthält weitere Informationen über die Applikation.	
Controller	**Partials**
• MainController	• about.html
Services	**Factories**
/about	

Terms	
Enthält die AGBs der Seite.	
Controller	**Partials**
• MainController	• terms.html
Services	**Factories**
•	
/terms	

5.4 Mehrsprachigkeit

Im Frontend werden für die Umsetzung der Mehrsprachigkeit das AngularJS-Modul angular-translate (angTransl) benutzt. Für jede Sprache ist ein eigenes JSON-File vorhanden, in welchem die einzelnen Übersetzungen mit einer ID abgelegt sind. Dies ermöglicht es dem Besitzer der Applikation, zu einem späteren Zeitpunkt einzelne Ausdrücke oder ganze Sätze ohne grossen Aufwand austauschen zu können. Durch das einfache Datenformat kann auch ein technisch nicht versierter Übersetzer diese Aufgabe übernehmen. Die IDs sind auf Englisch verfasst und mit Präfixen versehen, welche den Zweck und Verwendungsort jeder Übersetzung beschreiben. Der Präfix “main_” steht zum Beispiel für die Hauptseite, die ID “main_followUp_delete_title” wird also auf der Startseite benötigt, hier etwa als Titel des Dialogs beim Löschen eines Follow-Ups.

Die einzelnen Übersetzungen werden dynamisch nachgeladen, um die Verbindungen beim ersten Besuch der Seite zu minimieren. Sollten einmal mehr als die zwei aktuellen Sprachen (Deutsch und Englisch) vorhanden sein, kann der Inhalt einer einzelnen Datei mit allen Sprachinhalten schnell gross werden, obwohl ein User typischerweise nicht mehr als 2 oder 3 Sprachen benutzt.

5.5 Demo-Modus

Ein Ziel dieses Projektes war es, dass die Applikation potentiellen Käufern vorgestellt und demonstriert werden kann. Dazu wurde von den Projektpartnern ein Demo-Skript gewünscht, welches bei einer solchen Präsentation durchgeführt werden kann und die Funktionen der Applikation im Kontext eines fiktiven Benutzers zeigt.

Es wurden zwei unabhängige Abläufe erstellt: Alfred, welcher die App bereits kannte und registriert ist, hat Kopfschmerzen. Bruce hat lediglich von der Applikation gehört und probiert sie zum ersten Mal aus und legt gleich ein Follow-Up an um die Diagnose, Lungenentzündung, zu verifizieren. Beide Abläufe können Schritt für Schritt durchgeführt werden und beschreiben auch die Absichten hinter den Entscheidungen, welche die fiktiven Benutzer fällen sowie die einzelnen Aktionen im Detail, welche der Moderator dieser Präsentation ausführen muss.

Innerhalb der Applikation wurde ein Reset-Button eingebaut, damit vor/während einer Präsentation sämtliche Lokalen Daten bezüglich der Applikation gelöscht werden, so dass in der Zwischenzeit angelegte Benutzer und sonstige neuere Daten nicht die Präsentation stören können:

Abbildung 27: Demo-Modus: Daten Zurücksetzen

Die Skripte befinden sich im Anhang.

6 Abschluss

6.1 Verworfene Funktionen

Im Verlauf des Projekts, besonders während der Mockup-Phase, wurden mehrere Funktionen beschrieben und geplant, welche allerdings während der Implementation wieder verworfen wurden. In diesem Kapitel wird kurz auf diese Funktionen eingegangen und beschrieben, weshalb sie nicht umgesetzt wurden.

6.1.1 Auf Kosten des Simulators verworfen

Aufgrund des erhöhten Aufwands durch die Implementierung des Backend-Simulators fielen nach Absprache mit den Projektpartnern folgende Punkte aus dem Projekt:

History

Ursprünglich war geplant, dass Benutzer ihre bereits abgeschlossenen Diagnosedurchläufe jederzeit wieder einsehen können. Dieses Feature wurde gestrichen, da es für die Demonstration der Applikation nicht erforderlich ist und keine weiteren Features darauf aufbauen.

Abbildung 28: Verworfen: History 1

Abbildung 29: Verworfen: History 2

Vom eigenen Profil aus konnte die History eingesehen werden, welche alle beantworteten Fragen und gefundenen Diagnosen und Handlungsempfehlungen beinhaltet sowie die Daten

zur betroffenen Person aufzeigt, da es sich nicht zwingend um den Besitzer des Accounts handeln muss, sondern auch eine Fremdperson sein kann.

Fortsetzung einer begonnenen Diagnose

Ein Benutzer befindet sich mitten in einer Diagnose, wobei er durch einen wichtigen Zwischenfall unterbrochen wird. Nun sollte es dem Benutzer möglich sein, die App zu beenden und beim nächsten Start direkt an jener Stelle wieder zu starten.

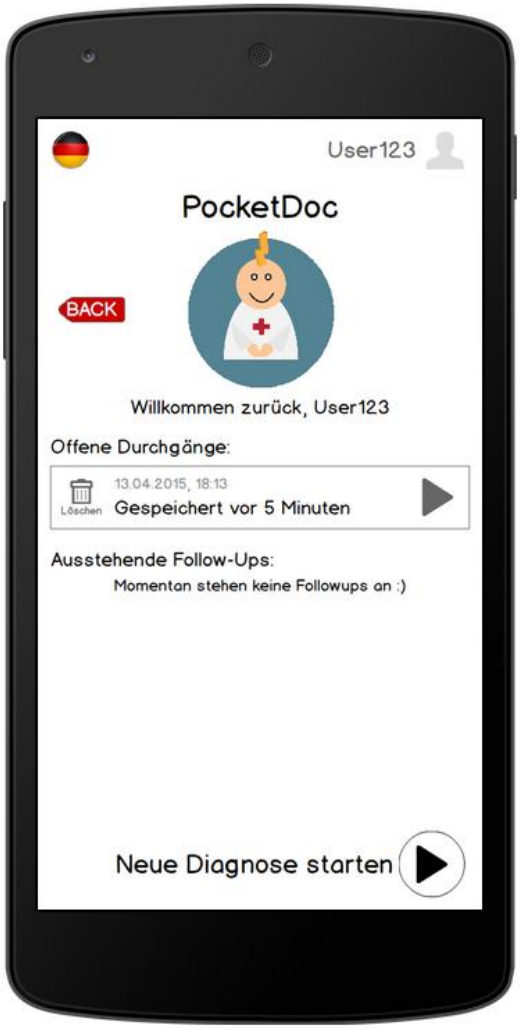


Abbildung 30: Verworfen: Diagnose Fortsetzen

Das Feature war so geplant, dass es über den offenen Follow-Ups auf der Startseite aufgeführt sein sollte und ebenfalls gelöscht oder weiter ausgeführt werden.

6.1.2 Aus anderen Gründen verworfen

Ein paar weitere Features wurden im Laufe des Projekts verworfen.

Mehrere Fragetypen

Während der Mockup-Phase wurden neben Ja/Nein/Weiss Nicht-Fragen auch noch andere Fragetypen definiert. Darunter waren Multiple Choice Auswahllisten, Slider sowie auch Dropdownlisten. Schlussendlich haben wir uns allerdings auf die anfangs geplanten Ja/Nein/Weiss Nicht Fragen beschränkt, da sich die anderen Typen als wenig Praxistauglich herausstellten. Die einzige Ausnahme bildet der Pre-Diagnosis Screen, in welchem der Nutzer Name, Alter, Geschlecht sowie Sprache definiert.

Genaue Altersangabe

Anfangs war vorgesehen, den Jahrgang des Nutzers für die Diagnose zu verwenden. Da ein Nutzer bei so präzisen, persönlichen Angaben eher dazu geneigt ist, etwas Falsches anzugeben, wurde dies über vordefinierte Altersgruppen gelöst.

6.2 Code-Statistiken

Das PocketDoc-Frontend besteht als Angular-Project aus Javascript-Code, HTML und CSS. Am Ende umfasste die Applikation aus 3862 Zeilen reinem Code oder Markup/Style:

Sprache	Dateien	Leer	Comment	Code
Javascript	15	400	496	3052
HTML	13	126	28	587
CSS	1	45	1	223
Total:	29	571	525	**3862**

(erstellt mit (CLOC))

Dies erscheint auf den ersten Blick etwas wenig. Es sollte aber berücksichtigt werden, dass zu Beginn des Projekts auch einige Änderungen am Backend durchgeführt wurden und während mehr als 2 Sprints kein Code geschrieben, sondern Mockups und Wireframes erstellt wurden.

Insgesamt wurden laut GitHub-Statistik aber 15592 Zeilen in 193 commits erstellt, wovon aber 10113 wieder entfernt wurden. Die Differenz zum obigen Total befindet sich in JSON-Files, z.B. als Übersetzung in Deutsch und Englisch.

6.3 Persönliche Berichte

6.3.1 Philipp Christen

Das Projekt war nicht meine erste Wahl, was sich im Nachhinein als Falscheinschätzung entpuppte, ich bin insgesamt froh, dass wir diese Semesterarbeit umsetzen konnten.

Roman als Projektpartner war extrem unkompliziert, er hat sehr gut und selbstständig gearbeitet und ist zuverlässig. Wir haben uns vor diesem Projekt nicht gekannt, was aber nie zu Problemen führte. Etwas mehr direkte Zusammenarbeit wäre ab und zu wünschenswert gewesen, was aber durch die Stundenpläne und Teilzeitjobs von uns beiden nicht wirklich klappte; wir sahen uns nur zwei Mal pro Woche und dann auch mehrheitlich in Vorlesungen. Aber über Email, JIRA und Skype hat die Kommunikation gut funktioniert.

Die Zusammenarbeit mit Samuel und Flavio war ebenfalls ziemlich unkompliziert. Trotz vielen Terminen nahmen sie sich sehr oft Zeit für uns und nahmen auch an Skype-Meetings teil, wenn sie gerade im Zug eine Busse erhielten. Schade war eigentlich nur, dass der Grossteil der Meetings mit ihnen über Skype geführt wurde. Die Treffen in Zürich waren meiner Meinung nach meistens angenehmer und führten zu besseren Resultaten. Allerdings ist es bei Teilzeitstudenten auch nicht einfach, bereits nur für 2 Personen ein Meeting zu finden.

Die Projektpartner waren dem Projekt gegenüber mehrheitlich positiv eingestellt. Dass das Backend nicht ihren Erwartungen und Hoffnungen entsprach ist schade, war für uns aber eine Chance, den Fokus mehr auf das Frontend zu legen und dort eine gute Arbeit zu leisten. Leider verging eine relativ lange Zeit, bis diese Entscheidung fiel. Wir waren uns bewusst, dass das Backend nicht so schnell und stabil war, wie wir es uns erhofften, dachten aber dass wir das reparieren könnten. Gemeinsam mit den Projektpartnern entschieden wir uns dann aber für ein reines Frontend-Projekt. Mir kam das sehr gelegen, da ich mich in Zukunft sehr gerne mehr in Richtung UI/UX bewegen möchte. So waren auch die Sprints mit

dem Erstellen und Verbessern der Mockups eher mein Gebiet als Schnittstellen zu definieren oder Berichte zu schreiben. Auch die Umsetzung der Mockups mit dem AngularJS-Framework und Material Design war spannend, ich habe hier viel gelernt was ich zufälligerweise in den letzten 2 Wochen bereits bei einem Projekt an meinem Teilzeitjob einbringen konnte.

Die Inputs von Daniel (atfront.ch) waren ebenfalls interessant, auch wenn das leider etwas spät im Projekt durchgeführt wurde. Die Inputs basierten noch auf den Mockups, während wir schon seit zwei Wochen an der Umsetzung davon arbeiteten. So waren ein Teil der Bemerkungen nicht mehr relevant und Daniel hätte sich dabei auch den Prototypen anschauen können, wo es sicherlich auch noch einige Verbesserungsmöglichkeiten gegeben hätte. Aber alleine schon für den Kontakt zu atfront bin ich froh, eventuell ergibt sich daraus einmal eine Zusammenarbeit für ein weiteres Projekt.

Professor Glatz als Betreuer hielt sich meistens im Hintergrund, setzte sich aber bei den Schwierigkeiten mit dem Backend sehr für uns ein und ermöglichte es uns, das Projekt auch ohne Backend fertigzustellen.

Mit dem Resultat der Semesterarbeit bin ich zufrieden. Wenn wir von Anfang an auf das Backend verzichtet hätten, wäre es sicherlich möglich gewesen, die noch fehlenden Funktionalitäten auch noch einzubauen und eine weitere Testing- und Anpassungs-Runde durchzuführen, um Mängel bei der UX zu beseitigen. Aber wenn man bedenkt dass wir eigentlich “nur” zwei Sprints lang am AngularJS-Frontend-System arbeiteten und wir beide vorher Angular nicht kannten, bin ich doch positiv überrascht, wie gut es schlussendlich funktioniert.

6.3.2 Roman Eichenberger

Die Arbeit verlief für mich mehrheitlich positiv. Da es seit längerem meine Erste komplett geplante und umgesetzte Arbeit war auch entsprechend spannend.

Die Arbeit mit Philipp war, entgegen meinen bisherigen Erfahrungen in Gruppenarbeiten an der Schule, super. Obwohl wir aufgrund unserer Stundenpläne und Jobs oftmals nicht gemeinsam am Projekt arbeiten konnten, konnte man sich auf ihn verlassen. Das ermöglichte auch eine reibungslose Zusammenarbeit über Skype und Jira.
Da wir die Bachelor Arbeit an unterschiedlichen Semestern durchführen, ist eine weitere Zusammenarbeit leider nicht möglich.

Die Zusammenarbeit mit Samuel und Flavio stellte sich als sehr unkompliziert heraus. Neben den zweiwöchigen Sprint Meetings hatten wir jederzeit die Möglichkeit, Fragen zu stellen und falls nötig, ein weiteres Meeting zu planen. Diese wurden meist über Skype gehalten, zwei Mal trafen wir uns auch persönlich. Auch wenn teilweise Unklarheiten entstanden, konnten wir die meist beseitigen.

Die Arbeit selbst verlief besonders am Anfang etwas chaotisch. Gerade als wir unsere Entwicklungsumgebung aufgesetzt hatten, stellte sich heraus, dass das Backend System fehlerhaft und unperformant war. Nach dem ersten Schock wurde allerdings beschlossen, die Arbeit wie geplant fortzuführen mit dem Zusatz, dafür einen Simulator zu implementieren. Abgesehen vom erhöhten Aufwand erlaubte uns dies, einen grösseren Einfluss auf die geplanten Funktionen einzunehmen.
Ausserdem stellte sich die Zusammenarbeit mit atfront, ein auf UX spezialisiertes Unternehmen, als sehr wertvoll heraus. Obwohl wir einige der Kritikpunkte an unserem Mockup zu diesem Zeitpunkt bereits in der Implementation korrekt umgesetzt hatten, wurden wir dadurch immerhin in unserer Entscheidung bestätigt.

Mit dem erreichten Stand bin ich zufrieden. Sehr vieles war für mich, von den genutzten Tools bis zum Framework, neu. Wir verloren relativ viel Zeit mit der Korrektur des vorhandenen Backends. Da solche Probleme allerdings auch im Geschäftsleben auftreten können, war‘s auf jeden Fall eine lehrreiche Erfahrung. Da ich voraussichtlich im nächsten Semester dieses Projekt weiterführe, freue ich mich auf eine weitere Zusammenarbeit mit den Projektpartnern und bin gespannt auf die neuen Herausforderungen.

7 Glossar

Begriff	Beschreibung
Backend	Der Teil einer verteilten Anwendung, welcher auf der Server-Seite agiert und typischerweise Daten für ein Frontend bereitstellt.
Commit (git)	Ein Commit bei git "friert" den lokalen Code zu einem gewissen Zeitpunkt ein und speichert ihn, so dass dieser Snapshot später auf das Repository hochgeladen werden kann.
Follow-Up	Englisch für Nachfolgeuntersuchung, ein erneuter Besuch beim Arzt um die Diagnose zu festigen oder die Wirksamkeit der Behandlungsmethode zu prüfen. Hier dient ein Follow-Up dazu, die Diagnose 4 Stunden nach der ersten Diagnose zu kontrollieren und gegebenenfalls anzupassen.
Frontend	Der Teil einer verteilten Anwendung, welcher auf der Seite der Benutzer existiert und typischerweise mit einem Backend kommuniziert.
History	Umfasst die bisher durchgeführten Diagnose-Durchläufe
Mockup	Ein Mockup ist ein "Wegwerf-Prototyp", welcher dem fertigen Produkt ähnlich sieht, aber entweder keinen oder nur einen kleinen Funktionsumfang besitzt.
PP	Projektpartner, namentlich Flavio Trolese und Dr. med. Samuel Huber (Forventis GmbH)
Repository (git)	Ein Repository ist eine Code-Basis, typischerweise online auf einem Server. Von hier kann der Code lokal kopiert werden und die verschiedenen Änderungen von Entwicklern werden hier zusammengeführt.
Run	Ein Run bezeichnet intern einen Diagnosedurchlauf.
SCRUM	SCRUM ist ein iteratives und inkrementelles Framework für das Produktmanagement.
SW	Semesterwoche

8 References

Angular Material [Online]. - 28. 5 2015. - https://material.angularjs.org.

AngularJS [Online]. - 28. 5 2015. - https://angularjs.org/. - 1.3.15.

angular-translate [Online]. - 28. 5 2015. - https://angular-translate.github.io/.

Balsamiq Mockups [Online]. - 29. 5 2015. - https://balsamiq.com/products/mockups/.

Bower [Online]. - 28. 5 2015. - http://bower.io/.

CLOC [Online]. - 28. 5 2015. - http://cloc.sourceforge.net/.

expressJS [Online]. - 28. 5 2015. - http://expressjs.com/.

git [Online]. - 28. 5 2015. - https://git-scm.com/.

GitHub [Online]. - 28. 5 2015. - https://github.com/.

Google Material Design [Online]. - Google. - 28. 5 2015. - http://www.google.com/design/spec/material-design.

Heroku [Online]. - https://www.heroku.com/.

Karma-Runner [Online]. - 28. 5 2015. - https://karma-runner.github.io.

MarvelApp [Online]. - 29. 5 2015. - https://marvelapp.com/.

Node.JS [Online]. - 28. 5 2015. - https://nodejs.org/.

NPM [Online]. - 28. 5 2015. - https://www.npmjs.com/.

PhoneGap [Online]. - 28. 5 2015. - http://phonegap.com/.

PocketDoc Admin-Tool [Online] / Verf. Nathan Bourquin Oliver Frischknecht. - 28. 5 2015. - http://pocketdoc.herokuapp.com/.

Protractor [Online]. - 28. 5 2015. - https://angular.github.io/protractor/#/.

Underscore.JS [Online]. - 28. 5 2015. - http://underscorejs.org/.

9 Abbildungsverzeichnis

Anhang

10 Schnittstellen

10.1 RunService

<table>
<tr><th colspan="3">RunService</th></tr>
<tr><td colspan="3">Dieser Service stellt Funktionen für einen Diagnosedurchlauf bereit.</td></tr>
<tr><th colspan="3">startRun()</th></tr>
<tr><td colspan="3">Startet einen neuen Diagnose-Durchlauf</td></tr>
<tr><td>{
  id : <number>,
  patient : {
  self : <boolean>,
  name : <string>,
  gender : <number>,
  age_category : <number>,
  lang : <number>
  }
}</td><td>{
  id : <number>
  description : <string>
  answers :[
  id : <number>,
  desc : <string>,
  style : <string>,
  diagnosis : {
  short_desc : <string>
  description: <string>
  },
  action_suggestion : {
  description: <string>
  }
  ]
}</td><td>{
  errorMessage : <string>
}</td></tr>
<tr><th colspan="3">answerQuestion()</th></tr>
<tr><td colspan="3">Beantwortet eine Frage, um den Durchgang fortzusetzen.</td></tr>
<tr><td>{
  answerId : integer
}</td><td>{
  id : <number>
  description : <string>
  answers :[
  id : <number>,
  desc : <string>,
  style : <string>,
  diagnosis : {
  short_desc : <string>
  description: <string>
  },
  action_suggestion : {
  description: <string>
  }
  ]
}</td><td>{
  errorMessage : <string>
}</td></tr>
<tr><th colspan="3">changeAnswer()</th></tr>
<tr><td colspan="3">Ändert eine bereits gegebene Antwort einer Frage. Die nachfolgenden Fragen werden zurückgesetzt und die nächste Frage wird gestellt.</td></tr>
<tr><td>{
  questionId : <number>
}</td><td>{
  id : <number>
  description : <string>
  answers :[
  id : <number>,
  desc : <string>,
  style : <string>,
  diagnosis : {
  short_desc : <string>
  description: <string>
  },
  action_suggestion : {
  description: <string></td><td>{
  errorMessage : <string>
}</td></tr>
</table>

	}] }	
acceptDiagnosis()		
Akzeptiert vorgeschlagene Diagnose, um den Durchgang abzuschliessen.		

{
}
	{
	}
		{
		errorMessage : <string>
		}

getQuestionData()		
Holt die nächste Anwort.		

{
id : <number>
}
	{
	id : <number>
	description : <string>
	answers :[
	id : <number>,
	desc : <string>,
	style : <string>,
	diagnosis : {
	short_desc : <string>
	description: <string>
	},
	action_suggestion : {
	description: <string>
	}
	]
	}
		{
		errorMessage : <string>
		}

10.2 UserService

UserService		
Dieser Service stellt alle Funktionen, die für die Benutzerverwaltung nötig sind, zur Verfügung.		
createUser()		
Erstellt einen neuen User		

{
name : <string>,
email : <string>,
password : <string>,
gender : <number>,
age_category : <number>,
lang : <number>
}
	{
	id : <number>
	name : <string>,
	email : <string>,
	password : <string>,
	gender : <number>,
	age_category : <number>,
	lang : <number>
	}
		{
		errorMessage : <string>
		}

updateUser()		
Ändert die Einstellungen eines Users. Sollte das neue Passwort gesetzt werden, muss das alte Passwort zwingend auch gesetzt werden.		

{
name : <string>,
email : <string>,
newPassword : <string>,
oldPassword : <string>,
gender : <number>,
age_category : <number>,
lang : <number>
}
	{
	name : <string>,
	lang : <number>
	}
		{
		errorMessage : <string>
		}

deleteUser()		
Löscht den aktuellen Benutzer.		

{
}
	{
	name : <string>
	}
		{
		errorMessage : <string>
		}

loginUser()		
Loggt einen Benutzer ein.		

{
Session : <string>,
email : <string>,
password : <string>
}
	{
	id : <number>
	name : <string>,
	email : <string>,
	gender : <number>,
	age_category : <number>,
	lang : <number>
	}
		{
		errorMessage : <string>
		}

logoutUser()		
Loggt den aktuellen Benutzer aus.		

{
}
	{
	name : <string>
	}
		{
		errorMessage : <string>
		}

getUser()		
Gibt den aktuellen User mit allen Einstellungen zurück.		

{
}
	{
	name : <string>,
	email : <string>,
	gender : <number>,
	age_category : <number>,
	lang : <number>
	}
		{
		errorMessage : <string>
		}

10.3 FollowupService

FollowupService		
Dieser Service dient als Schnittstelle für alle Funktionen, die mit dem Follow-Up zusammenhängen.		
registerFollowup()		
Erstellt ein neues Follow-Up.		

{
userID : <number>,
oldDiagnosis : <number>,
oldActionSuggestion : <number>
}
	{
	}
		{
		errorMessage : <string>
		}

startFollowup()		
Startet ein Follow-Up und löscht es aus der Liste der ausstehenden Follow-Ups.		

{
followupId : <number>
}
	{
	}
		{
		errorMessage : <string>
		}

deleteFollowup()		
Löscht ein Follow-Up.		

{
followupId : <number>
}
	{
	followUpId: <number>
	}
		{
		errorMessage : <string>
		}

getFollowupsForUser()		
Gibt eine Liste aller Follow-Ups des aktuellen Benutzers zurück.		

{
}
	{
	followups : [
	id : <number>,
	timestamp : <datetime>,
	diagnosis : <string>,
	planned : <datetime>,
	forPerson : <string>
	]
	}
		{
		errorMessage : <string>
		}

getFollowup()		
Gibt ein bestimmtes Follow-Up zurück.		

{
followUpId : <number>
}
	{
	id : <number>,
	timestamp : <datetime>,
	diagnosis : <string>,
	planned : <datetime>,
	forPerson : <string>
	}
		{
		errorMessage : <string>
		}

10.4 HistoryService

HistoryService		
Dieser Service dient als Schnittstelle für alle Funktionen, die mit der History eines Benutzers interagieren.		
getUserHistory()		
Gibt eine Liste mit allen History-Einträgen eines Benutzers zurück.		

{
owner: <number>
}
	{
	[
	id : <number>,
	timestamp : <datetime>,
	patient: {
	self: <boolean>,
	name: <string>,
	gender: <number>,
	age_category: <number>,
	lang: <number>
	},
	content: [
	type: <string>,
	id: <number>,
	answer: <number>,
	accepted: <boolean>
	]
	]
	}
		{
		errorMessage: <string>
		}

getHistoryEntry()		

Liefert die Details eines einzelnen History-Eintrags eines Benutzers.

{
historyId: <number>
}
	{
	id : <number>,
	timestamp : <datetime>,
	patient: {
	self: <boolean>,
	name: <string>,
	gender: <number>,
	age_category: <number>,
	lang: <number>
	},
	content: [
	type: <string>,
	id: <number>,
	answer: <number>,
	accepted: <boolean>
	]
	}
		{
		errorMessage: <string>
		}

deleteHistoryEntry()

Löscht einen bestimmten History-Eintrag.

{
id: <number>
}
	{
	id: <number>
	}
		{
		errorMessage: <string>
		}

createHistoryEntry()

Erstellt einen neuen History-Eintrag für den aktuellen User

{
owner: <number>,
timestamp : <datetime>,
patient: {
self: <boolean>,
name: <string>,
gender: <number>,
age_category: <number>,
lang: <number>
},
content: [
type: <string>,
id: <number>,
answer: <number>,
accepted: <boolean>
]
}
	{
	id: <number>
	}
		{
		errorMessage: <string>
		}

10.5 DiagnosisService

DiagnosisService		
Beinhaltet Funktionen zu Diagnosen und Handlungsempfehlungen.		
getDiagByID()		
Liefert die (übersetzten) Daten zu einer Diagnose.		

`{`
`  id: <number>`
`}`
	`{`
	`  id: <number>,`
	`  short_desc: <string>`
	`  description: <string>`
	`}`
		`{`
		`  errorMessage : <string>`
		`}`

getActionByID()		
Liefert die (übersetzten) Daten zu einer Handlungsempfehlung		

`{`
`  id: <number>`
`}`
	`{`
	`  id: <number>,`
	`  description: <string>`
	`}`
		`{`
		`  errorMessage : <string>`
		`}`

10.6 MetaDataService

MetaDataService		
Holt Daten vom Backend, welche dort verwaltet werden und für die App generell benutzt werden. Dieser Service sollte nur beim Initialisieren der App benutzt werden.		
getLanguages()		
Holt eine Liste mit allen Sprachen und den jeweiligen Locales.		

`{`
`}`
	`{`
	`  id: <number>,`
	`  locale: <string>`
	`  nativeName: <string>`
	`}`
		`{`
		`  errorMessage : <string>`
		`}`

getAgeRanges()		
Liefert die Alterskategorien vom Backend.		

`{`
`}`
	`{`
	`  id: <number>`
	`  start: <number>,`
	`  end: <number>`
	`}`
		`{`
		`  errorMessage : <string>`
		`}`

11 Evaluationen

11.1 Technologien Frontend

Web-App	*Helpful*	*Harmful*
Internal Origin	Strengths: • viel Erfahrung	Weaknesses:
External Origin	Opportunities: • gleiche Technologie wie Admin-Tool • REST-API einfach anzusprechen • Einfach anzubieten	Threats: • Sehr viele Geräte-/Browser-Kombinationen • verschiedenste Frameworks • Potentiell Langsamer als Nativ • Kurzlebiges Feld

Native App	*Helpful*	*Harmful*
Internal Origin	Strengths: • Android: Java-Kentnisse	Weaknesses: • wenig Erfahrung • iOS: Objective C
External Origin	Opportunities: • Schnelle Reaktionszeiten • Android: gleiche Sprache wie Backend (Java) • Etablierte Prozesse für Entwicklung/Deploy • pro OS optimierter Styleguide	Threats: • Verschiedene Systeme und Versionen • neue/wechselnde OS-Versionen + Styleguides • Vertrieb über Marktplätze

Hybrid (Phonegap)	*Helpful*	*Harmful*
Internal Origin	Strengths: • Kentnisse in Web-Entwicklung	Weaknesses: • wenig Erfahrung
External Origin	Opportunities: • Entwicklung in “HTML”	Threats: • Potentielle Probleme bei Deploy (Inkompatibilität) • Immer noch langsamer als Nativ

Gemeinsam mit den Projektpartnern wurde entschieden, das Projekt als Web-App umzusetzen, da hier die Erfahrungen bereits zu einem Teil vorhanden sind und das Admin-Tool ebenfalls als Web-App existiert.

11.2 Mockup-Tools

Zur Auswahl standen verschiedene Tools. Die Auswahlkriterien umfassen:

- Kosten
- Verfügbarkeit (Online-Tool? Offline? Beides?)
- Erstellen von Layouts
- Verlinken von Screens
- Synchronisierung
- Präsentationsmodus

Tool	Kosten	Verfügbarkeit	Layouting	Linking	Synch	Präsentation
Balsamiq	Lizenz available	Offline	Ja	Ja	Ja	Offline/PDF
Mockup.io	-	Online	Nein	Ja + Anim.	Eigene Server	Ja
Marvelapp	-	Online	Nein	Ja	Dropbox	Ja
Concept-board	- 30d!	Online	Nein	Ja	Eigene Server	Ja

Ausgewählt wurde Balsamiq zur Erstellung der Grafiken und Marvelapp zum Verbinden und Präsentieren. Balsamiq war im Team schon bekannt und Marvelapp hat mit Dropbox-Synchronisation und dem einfach zu bedienenden Editor überzeugt.

12 Anpassungen im Backend

12.1 Erweiterung: Mehrfachabhängigkeit einer Antwort

Das Backend wurde so konzipiert, dass maximal eine Frage pro Antwort abhängen kann. Das hatte zur Folge, dass keine Baumstruktur artige Abhängigkeiten möglich waren, was die benötigte Funktionalität einschränkte. Diese Funktionalität wurde im Rahmen des Projekts implementiert

12.1.1 Implementation

Folgende Änderungen wurden dazu vorgenommen:

- Der Question Calculator wurde um einen Fragen-Stack erweitert. Wenn eine neue Frage beantwortet wurde, werden alle von dieser Antwort abhängigen Fragen auf einen Stack gespeichert. Wenn nun eine Antwort gegeben wird, welche keine abhängige Frage mehr enthält, wird die nächste vom Stack geladen, bis der Stack geleert wurde. Anschliessend wird über den Calculator die nächste passende Frage ermittelt.
- Diverse Anpassungen in den OR-Mapper-Klassen, damit pro Antwort mehrere Fragen als Abhängigkeit eingetragen werden können.
- Anpassung in der questions.html, damit die Option zur Abhängigkeit bei einer bereits vorhandenen nicht mehr ausgeblendet wird.

13 Projektabwicklung

13.1 Projektplan

13.1.1 Projektorganisation

Nach Absprache mit den Projektpartnern wurden folgende Rollen im SCRUM-Kontext übernommen: Philipp übernimmt die Rolle des Assistant Product Owners, Roman die des Scrum-Masters.

Flavio ist der Product Owner, Samuel der Kunde. Als externer Betreuer berät Prof. Dr. Eduard Glatz die Studenten.

13.1.2 Arbeitspakete

Die Arbeitspakete werden jeweils vor dem nächsten Sprint in JIRA erfasst und während dem Sprintmeeting mit dem Product Owner diskutiert.

13.1.3 Zeitplan

Aufgeteilt wurde das Projekt gemäss SCRUM in Sprints. Jeder Sprint startet mit einem Sprintplanungsmeeting und endet mit einem Review, wo das Ergebnis präsentiert wird, welches (theoretisch) auslieferbar ist.

SW	Sprint	Milestone / Deliverable	Empfänger	Meetings
1	Setup (0)			Betreuer
2				Betreuer, PP
3		Zeitplan	Betreuer	Betreuer, PP
4	1			Betreuer
5		1. Prototyp	Projektpartner	PP
6	2		Betreuer, PP	Betreuer, PP
7	3			Betreuer
8		Mockup-Prototyp	Projektpartner	Projektpartner
9	4			Betreuer
10		2. Prototyp	Projektpartner	Projektpartner
11	5			Betreuer
12		3. Prototyp	Projektpartner	Projektpartner
13	6	Vorabdokumentation	Betreuer	Betreuer
14		Finale Version	Projektpartner	Betreuer, PP
15	Reserve (7)	Dokumentation, Poster	Betreuer	Betreuer

SW = Semesterwoche

Da im Projekt PocketDoc nach SCRUM gearbeitet wird, gestaltet sich die exakte Planung schwierig. Jeder Sprint ist in sich geschlossen und als Ziel soll ein theoretisch auslieferbares Produkt erstellt werden. Was genau jedoch in jeden Sprint kommt wird vor jedem Sprint neu entschieden aufgrund der bisher geleisteten Arbeit, neuen Ideen und sich eventuell ändernden Anforderungen. Das Ziel ist grob umrissen, aber nicht genau festgelegt. So kann es sein, dass gewisse Funktionen früher, später oder gar nicht im Projekt umgesetzt werden, zum Beispiel wenn man merkt dass etwas gar nicht funktioniert oder eine Idee während dem Projekt eine andere Idee überflüssig macht. Bei herkömmlichen Vorgehensweisen ist es nicht immer möglich, agil auf solche Veränderungen zu reagieren.

Trotzdem wird versucht, die Entwicklung nach bestem Wissen und Gewissen vorherzusagen und einzuschätzen:

13.1.4 Sprints

Setup-Sprint

Im ersten Sprint wird noch keine Arbeit am Produkt verrichtet. Hier geht es lediglich um den Aufbau der Umgebung, um das Einarbeiten in die Domäne und das Kennenlernen der Tools, Personen und des Projektes. Gleichzeitig werden durch die Projektpartner und das Team einige Entscheidungen gefällt, welche substantiell für das Projekt sind, etwa welche Technologie verwendet wird.

Sprint 1

Im ersten Sprint wird ein erster Prototyp erstellt. Ziel ist es, die grundlegenden Funktionen des Frontends rudimentär zu implementieren und kleinere Fehler im Backend zu reparieren, so dass die nachfolgenden Sprints auf einer soliden Grundlage aufbauen können.

Sprint 2

Dieser Sprint wurde ursprünglich für Korrekturen im Backend sowie Implementation von automatisierten Tests im Frontend gedacht. Er wurde allerdings mit Einverständnis des Kunden abgebrochen, um die zukünftigen Sprints neu zu planen und das gesamte neue Vorgehen zu erarbeiten. Grund dafür war das Backend, welches nicht korrekt funktioniert und zu träge auf auf Anfragen reagiert.

Sprint 3

Das Hauptziel dieses Sprints ist das Erstellen eines umfangreichen, klickbaren Prototypen. Dies wird mit Hilfe von Mockup-Tools realisiert, um vor der tatsächlichen Implementation ein Gefühl für die Bedienung zu erhalten und Designfehler möglichst früh zu erkennen.

Sprint 4

In diesem Sprint werden die Mockups von externen Partnern, welche sich auf UX-Design spezialisiert haben, geprüft und bewertet. Dazu wird einige Zeit für die Kommunikation mit der Fachperson aufgewendet.

Gleichzeitig wird ein Konzept für das Ersatz-Mini-Backend erstellt und umgesetzt. Rudimentäre Screens dienen als Platzhalter für den späteren Ablauf, bis alles vom Kunden und der UX-Fachperson gutgeheissen wurde.

Sprint 5

Mit der bestehenden Datenstruktur und dem Feedback der UX-Fachperson wird das Frontend umgesetzt. Es werden die Hauptfunktionalität umgesetzt, um das System auch

ohne funktionierendes Backend testen und nutzen zu können. Zusätzlich werden automatisierte Tests geschrieben, um die Korrektheit der Funktionen zu gewährleisten

Sprint 6

Im letzten Sprint wird die Arbeit an der Software abgeschlossen. Es werden letzte Fehler behoben und je nach Wunsch das ein oder andere Feature implementiert.

Im Bericht werden fehlende Kapitel noch für die Vor-Abgabe eingefügt und dieser schliesslich abgegeben.

Reservewoche

Die letzte Woche dient dazu, die Übergabe des Produkts zu organisieren und den Bericht abzuschliessen. Falls bei den Testings im vorherigen Sprint noch grössere Missstände auftauchen, kann diese Woche alternativ auch für letzte Anpassungen benutzt werden.

13.1.5 Meilensteine

Zeitplan, Woche 3

- Alle benötigten Systeme sind aufgesetzt (Codeverwaltung, Backend lokal installiert, Projektplanungs-Tool u.a.)
- Grober Zeitplan für den Verlauf der Arbeit erstellt
- In diversen Meetings die nötigen Bedingungen für das Projekt besprochen (Verwendete Technologien, Funktionsumfang, Projektablauf u.a.)

1. Prototyp, Woche 5

- Grundlegende Funktionen in der Applikation sind umgesetzt, sprich Fragen werden vom Backend geholt und aufgrund der Antworten kann anschliessend eine Diagnose gestellt werden.
- Applikation kann für Demo-Zwecke verwendet werden
- Bug der Abhängigkeiten zwischen den Fragen im Backend ist behoben
- Wireframes und Mockups für Grundfunktionalität erstellt

Mockup-Prototyp, Woche 8

- Mockups für alle Screens erstellt und als klickbarer Prototyp verfügbargestellt.
- Sequenzdiagramme erstellt
- Use Case-Diagramm erstellt und im “Brief”-Format definiert
- Management Summary verfasst
- Nichtfunktionale Anforderungen definiert

2. Prototyp, Woche 10

- Die Applikation wird um die in der zuvor gemeinsam erarbeiteten UX-Analyse ermittelten Verbesserungen ergänzt und überarbeitet.
- Ersatzbackend für die Testläufe steht.
- Rudimentäre GUI, um mit dem Ersatzbackend zu kommunizieren und die Funktionalität zu testen.
- Umsetzung (Aufbau der Applikation) rudimentär dokumentiert (Domänenmodell).

3. Prototyp, Woche 12

- Anpassungen der GUI anhand weiterer Feedbacks bezüglich UX
- Software umgesetzt mit der besprochenen Funktionalität
- Ergänzung des Berichts anhand der neuen Erkenntnisse
- Automatisierte Tests für Software erstellt
- gewünschte API-Schnittstelle dokumentiert
- Kommunikationsdiagramme erstellt

Vorabdokumentation, Woche 13

- Auf das Angebot des Betreuers hin wird eine Vorabversion des Berichts einige Zeit vor dem finalen Abgabetermin eingereicht. So können Missstände noch korrigiert sowie kleinere unschöne oder fehlende Details ergänzt oder weggelassen werden.
- Komplettierung des Berichts für die Korrektur

Finale Version, Woche 14

- Die gewünschten Features sind eingebaut und getestet.
- Der Bericht wird aufgrund des Feedbacks aus Woche 13 ergänzt und korrigiert
- Abschluss-Präsentation wird vorbereitet und gehalten

Dokumentation, Woche 15

- Der abschliessende Projektbericht wird aufgrund der Vorlagen erstellt und abgegeben.
- Plakat für die Arbeit wird erstellt.

13.1.6 Meetings

Sämtliche Meetings werden vorbereitet und protokolliert. Die Protokolle sind auf dem gemeinsamen Google Drive abgelegt, welches für die Projektpartner einsehbar ist. Dem Betreuer werden sie per Email zugestellt.

Wochensitzung

Alle zwei Wochen wird ein kurzes (ca. 30min) Treffen mit dem Betreuer stattfinden, wo Probleme besprochen und Fragen vom Team gestellt werden können. Normalerweise findet dieses am Montag nach dem Sprintmeeting statt. Auf Wunsch können auch weitere Meetings einberufen werden.

Sprintplanungsmeeting

Dieses Meeting findet idealerweise zwischen zwei Sprints statt. Es soll der vergangene Sprint reviewt und ausgewertet sowie das Ergebnis des Sprints vorgestellt werden. Gleichzeitig wird vom Product Owner und dem Entwicklerteam beschlossen, welche Tasks im nächsten Sprint angegangen werden. Als Resultat soll der Inhalt des nächsten Sprints festgelegt worden sein.

Dailies

Laut SCRUM-Guide soll sich das Entwickler-Team täglich in einem Standup-Meeting treffen und kurz ihre vergangenen, aktuellen und zukünftigen Tasks schildern sowie Probleme, welche aufgetreten sind. Da beide Teammitglieder Teilzeitstudenten sind und sich die Stundenpläne nur selten überschneiden, gibt es nur zwei fixe Termine, wo ein Daily stattfinden kann. An den anderen Tagen wird das Daily nach gemeinsamer Absprache

gehalten, da nicht an allen Tagen am Projekt gearbeitet wurde. Mindestens 3-mal pro Woche werden sich die Entwickler aber austauschen.

13.1.7 Arbeitsaufteilung

Für die Semesterarbeit werden 8 ECTS-Punkte vergeben, wobei man pro Punkt von 30 Stunden Aufwand ausgeht. Für zwei Teammitglieder resultiert das in 16 Punkten oder 480 Stunden, welche auf 15 Wochen aufgeteilt wurden, was dem gesamten Soll-Aufwand entspricht und in jeder Woche 16 Stunden und insgesamt 240 Stunden pro Person bedeutet. Es wird nach SCRUM/Agile gearbeitet und das Projekt in mehrere Sprints eingeteilt. In jedem Sprint befinden sich geschätzte und auf kleinere Häppchen heruntergebrochene Tasks, welche vor und während dem Sprint auf die Entwickler aufgeteilt werden. Daher lässt sich im Voraus die Arbeit nicht genau verteilen. Ziel ist aber, dass beide ca. gleich viel Aufwand investieren.

13.2 Projektrückblick

13.2.1 Sprints

Sprint 0 (Setup)

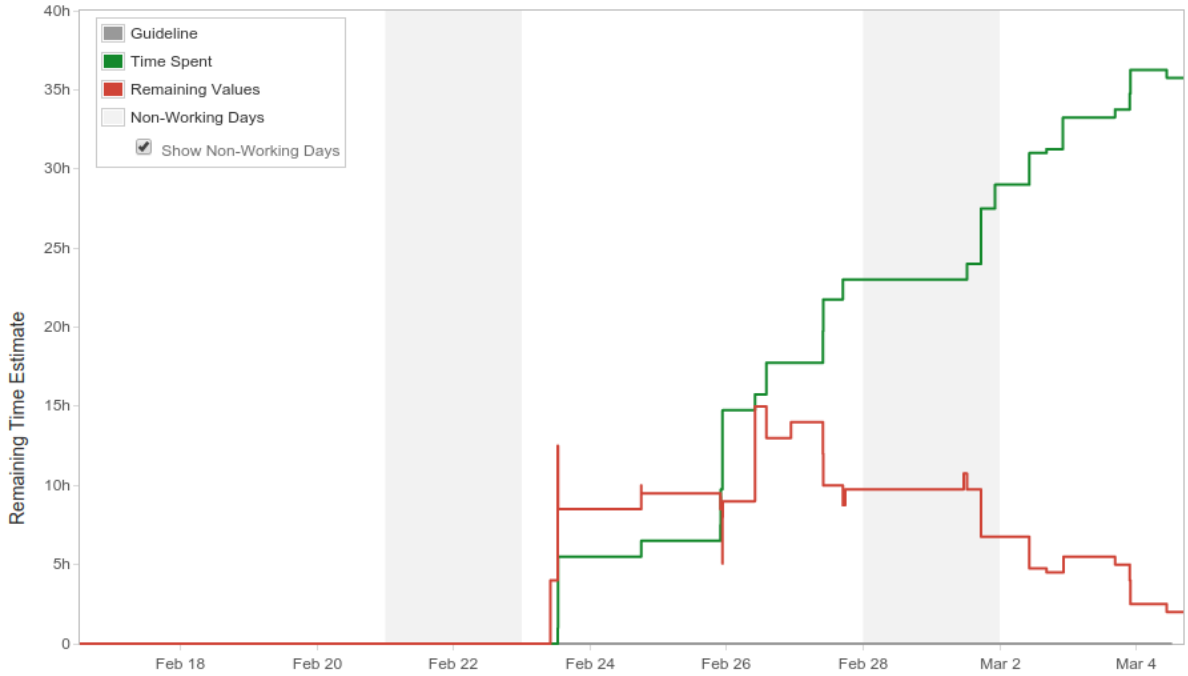


Abbildung 31: Projektrückblick: Sprint 0

Der Vorbereitungssprint diente als Setup für das ganze Projekt. Hier wurden der Bericht der vorherigen SA analysiert, Vorbereitungen für den Arbeitsprozess getroffen, Entscheidungen betreffend dem groben Ablauf/Technologie gefällt und Kentnisse in den verwendeten Technologien gesammelt.

Sprint 1

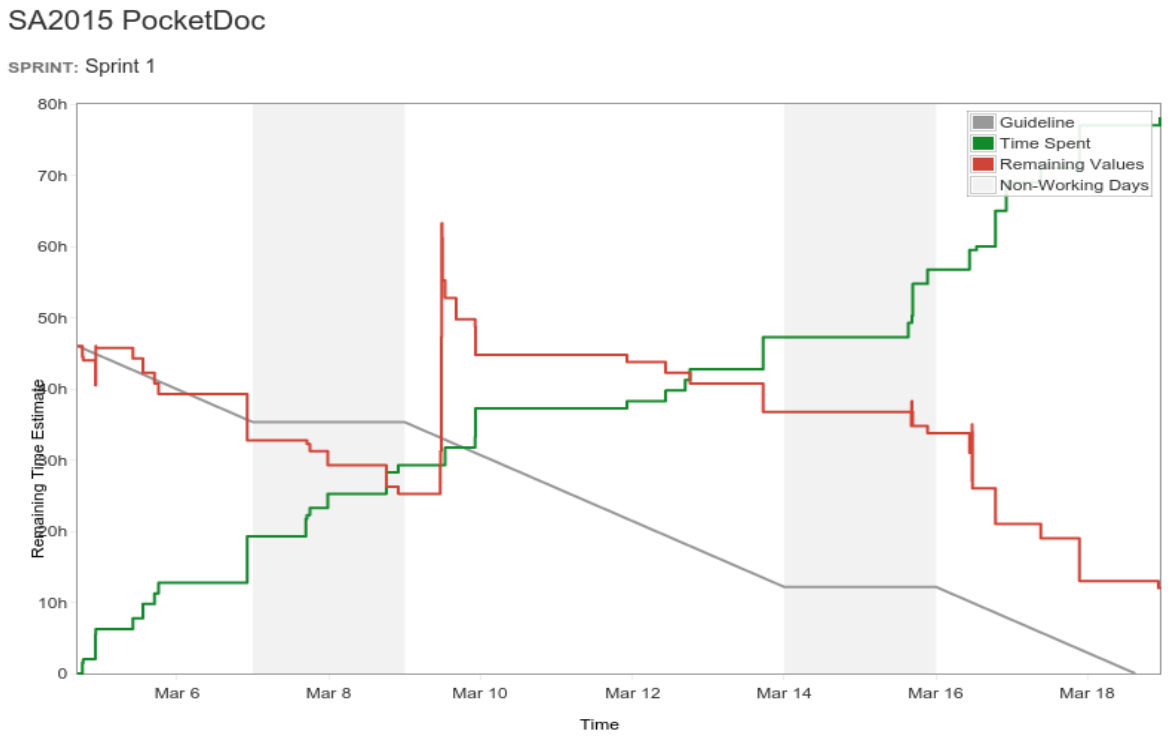


Abbildung 32: Projektrückblick: Sprint 1

Im 1. “richtigen” Sprint wurde vergessen, bei zwei Tasks die geschätzte Zeit einzutragen, was nachträglich geändert wurde. Dies erklärt den Spike am 9. März und die falsche Ursprungszeit.

Zudem wurde fast keine Zeit für Administratives wie den Abschlussbericht eingerechnet, sondern auf effektive Arbeit am Projekt aufgewendet. Kurz vor Ende des Sprints stellte sich heraus, dass das Backend nicht stabil genug ist und bereits bei nur einem gleichzeitigen User in der Mehrheit der Fälle abstürzt und kein Resultat liefert. Die grösste Schwierigkeit beim ersten Sprint war neben dem instabilen Backend die fehlende Möglichkeit, CORS auf dem Server zu aktivieren und Cross-Domain-Zugriffe zu erlauben. Am Tag vor dem Sprintende wurde entschieden, das Frontend auf den gleichen Server wie das Backend zu legen. Dies hatte die Konsequenz, dass bei jedem Commit der gesamte Build-Prozess ausgelöst wird, was die Entwicklung erschwert.

Sprint 2

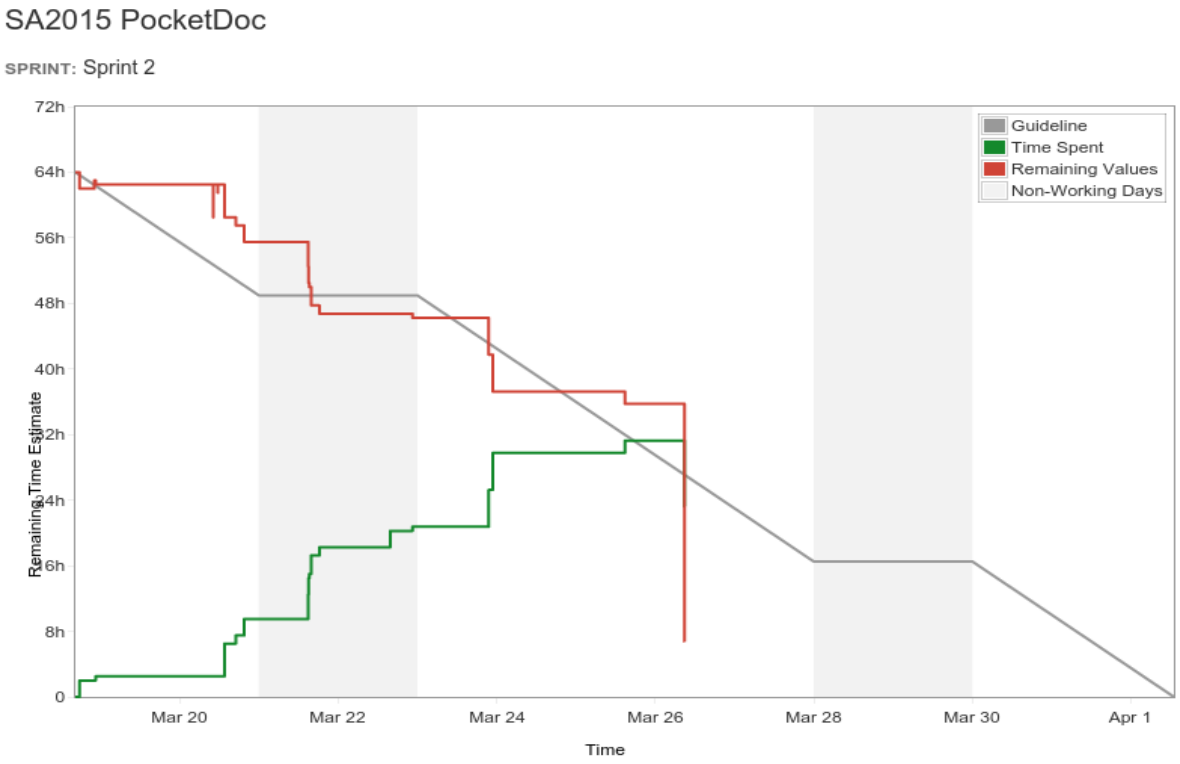


Abbildung 33: Projektrückblick: Sprint 2

Zu Beginn vom zweiten Sprint wurde entschieden, einige Zeit in die Analyse des Backends zu investieren. Ziel war, eine Entscheidung zu treffen betreffend dem weiteren Vorgehen.

Zu Beginn der zweiten Woche wurde gemeinsam entschieden, den Sprint abzubrechen. Grund dafür war der Zustand des Backends, welches so nicht brauchbar war.

Sprint 3

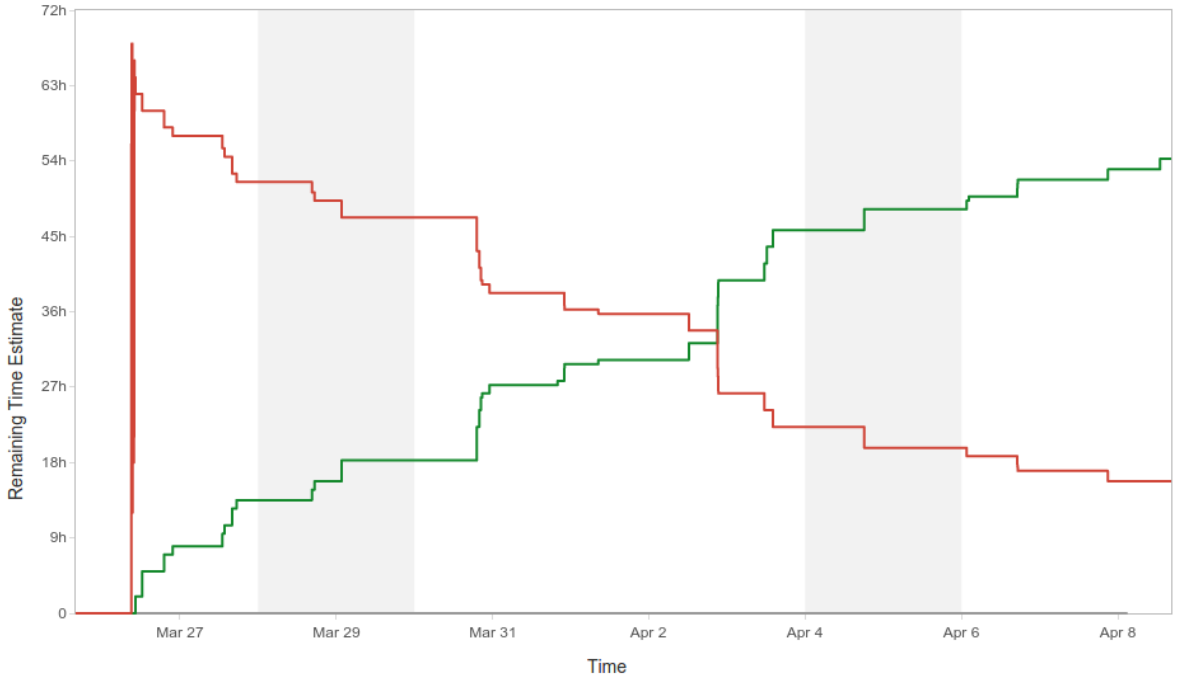


Abbildung 34: Projektrückblick: Sprint 3

Der dritte Sprint umfasste wieder zwei Wochen. Das Hauptaugenmerk in diesem Sprint wurde auf die Design Mockups gelegt. Mit Hilfe von Balsamiq und marvelapp.com konnte so ein klickbarer Prototyp der PocketDoc-App erstellt werden.

Dieser Sprint verlief im Allgemeinen problemlos ab und konnte sogar mit weniger Zeitaufwand als geschätzt beendet werden.

Sprint 4

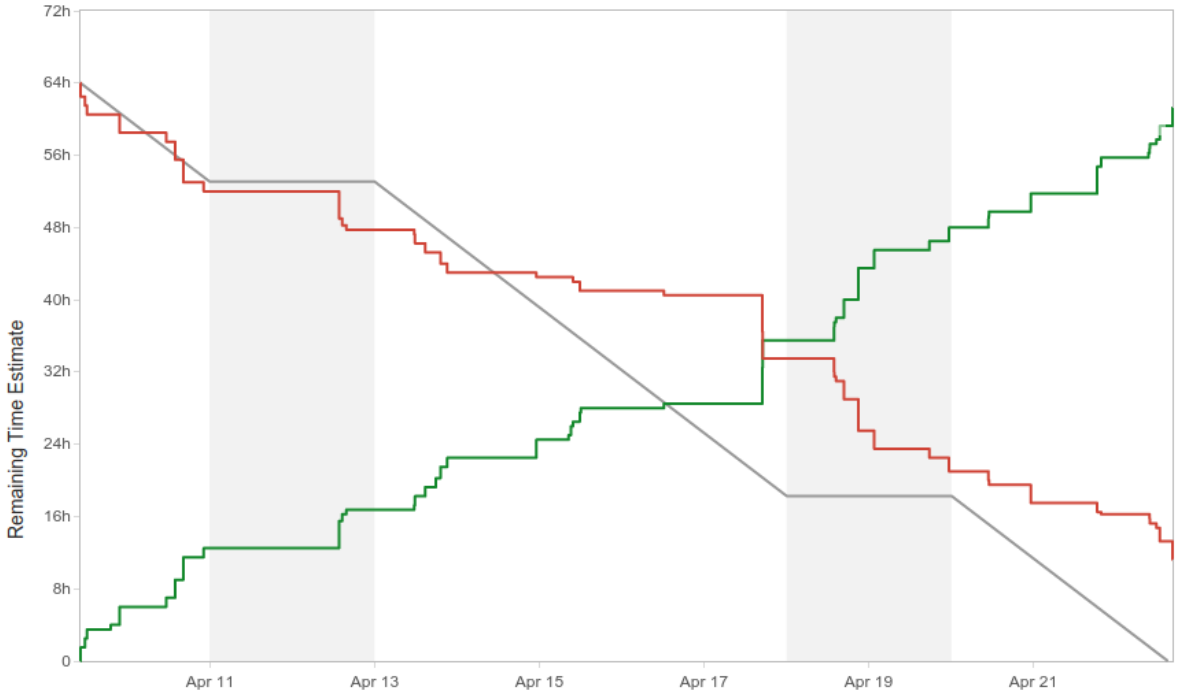


Abbildung 35: Projektrückblick: Sprint 4

Der vierte Sprint umfasste hauptsächlich das Umschreiben des Frontends, so dass statt dem alten Backend ein im Frontend integrierter “Simulator” angesprochen wird. Zier war es, dass das Holen und Beantworten der Fragen und Diagnosen/Handlungsempfehlung funktioniert, so wie es beim vorherigen Prototyp mit Backend-Anbindung der Fall war.

Zudem wurden die Mockups nochmals verfeinert; es wurden (auf Verlangen der PP hin) Vorschläge bezüglich weiteren Fragetypen (Checkboxen, Radio-Buttons), der Position der Fragen und Antworten sowie Follow-Ups ohne Dialoganzeigen erstellt. Kleinere Abweichungen zwischen einzelnen Mockups wurden ebenfalls entfernt. Mit den fertigen Mockups wurden anschliessend zwei Usertests durchgeführt mit der ersten der beiden Demo-Skripte. Die Ergebnisse daraus waren zufriedenstellend, die Bemerkungen und Einwände wurden den PP präsentiert.

Im Bericht wurden die Schnittstellen zum noch zu erstellenden neuen Backend (Nicht Bestandteil der SA) definiert und erläutert, sowie die wichtigsten Mockups im Detail beschrieben.

Probleme traten bei der Bedienung mit mobilen Geräten auf, so wurde z.B. oftmals ein Klick-Event doppelt registriert. Dies wurde allerdings im Sprint behoben.

Beim Sprintmeeting am Ende des Sprints kam es zu technischen Problemen und es musste mit jedem PP getrennt kommuniziert werden, was aber schlussendlich funktionierte.

Sprint 5

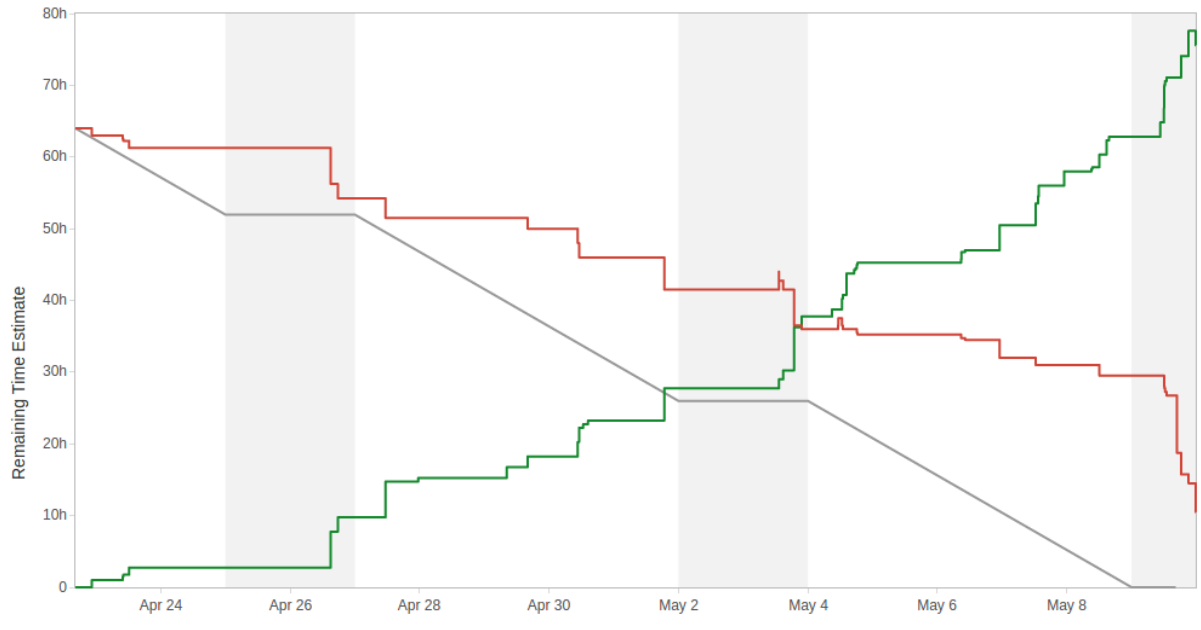


Abbildung 36: Projektrückblick: Sprint 5

Sprint 5 enthielt den Grossteil der Umsetzungstasks. Es wurden die Login-/Logout- und Registrierungsfunktion implementiert, die Fragen mehrfach umgestellt und anschliessend eingebaut, die Follow-Up-Funktion erstellt und einiges am Design geschraubt, so dass die einzelnen Views eher als Teil einer einzigen Applikation anstatt als eigenständige Seiten aufgefasst werden.

Diese Umsetzungstasks wurden eher optimistisch geplant und es wurde schlussendlich mehr Zeit damit verbracht als mit dem Fertigstellen des Berichts für die Vorabgabe. Da ein Missverständnis bezüglich den verschiedenen Fragetypen auftrat (nur Ja-Nein- vs. Ja-Nein-UND Auswahl-Fragen), musste hier und bei der endgültigen Reihenfolge der fix hinterlegten

Fragen inklusive den Demo-Skripten ebenfalls mehr Zeit als geplant eingesetzt werden. Gemäss Rücksprache mit den PP konnte dafür aber die History-Funktionalität aussen vorgelassen werden, da diese in der geplanten Demo für potentielle Kunden/Partner sowieso nicht vorgesehen ist und für deren Umsetzung ein Tag eingeplant war.

Das Sprintmeeting wurde in Zürich am Wochenende abgehalten und verlief zufriedenstellend, die PP waren mit dem Resultat glücklich.

Sprint 6

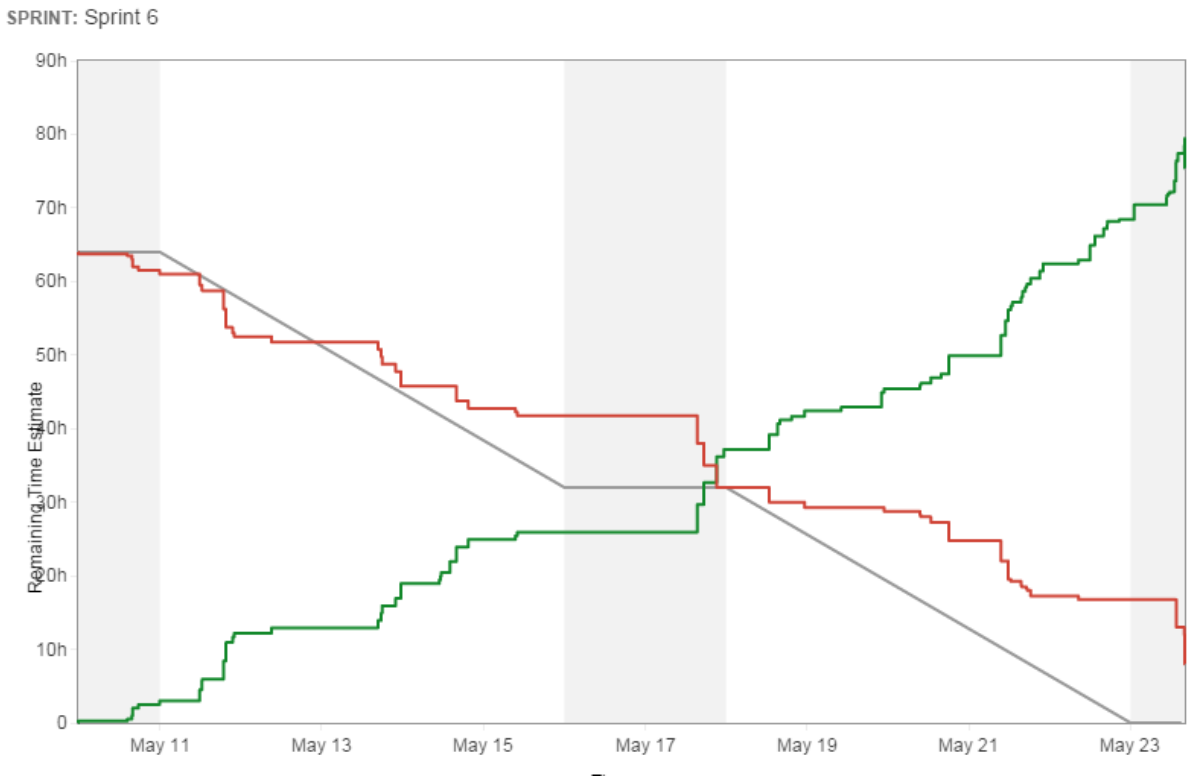


Abbildung 37: Projektrückblick: Sprint 6

Im letzten Sprint vor der Reservewoche handelte sich alles um den Abschluss des Projekts. Begonnen haben wir den Sprint mit einem Meeting mit atfront, um auch noch ein Feedback von UX Experten zu erhalten. Die Korrekturvorschläge bezogen sich hauptsächlich auf die Mockups. Da wir allerdings viele der angesprochenen Punkte bereits in der Applikation angepasst hatten, hielten sich die Anpassungen in Grenzen. Allerdings wurden unsere Design Entscheide von den Mockups zur endgültigen Lösung nochmals bestätigt.

Neben diesen Ergänzungen waren noch diverse Fehler sowie ein paar Wünsche der Projektpartner auf der Auftragsliste vermerkt. Der Aufwand für alles stellte sich als höher aus als ursprünglich geplant, weshalb die verbuchte Zeit die Geplante überschritt. Die Software konnte allerdings wie geplant fertiggestellt werden.

Nebenbei erhielten wir die Korrektur zum vorläufigen Bericht zurück. Durch den höheren Aufwand in der Korrektur der Applikation konnte für den Bericht allerdings nicht mehr so viel Zeit aufgewendet werden, wie geplant war. Dadurch verschiebt sich die Fertigstellung hauptsächlich auf die letzte Woche.

Reservewoche

Die Woche zwischen dem letzten Sprint und der Abgabe wurde für die Fertigstellung des Plakats und des Berichts verwendet. An der Software selbst wurden keine grossen

Änderungen mehr vorgenommen, abgesehen von wenigen CSS-Anpassungen oder noch nicht ganz perfekten Übersetzungen in Englisch.

13.2.2 Meetings

Die Betreuer-Meetings wurden zu Beginn wöchentlich eingeplant. Nach einigen Iterationen wurde klar, dass es während dem Sprint nicht viel zu besprechen gab. Zusammen mit dem Betreuer wurde entschieden, die gemeinsamen Meetings nur noch jede zweite Woche abzuhalten, jedes Mal am Montag nach dem Sprintmeeting.

Zusätzlich zu den eingeplanten Sprintmeetings wurden einige ausserordentliche Treffen einberufen. So trafen sich die Entwickler beispielsweise mit dem Product Owner im Sprint 2, um die Entscheidung betreffend dem weiteren Vorgehen bezüglich dem Backend-Problem zu besprechen. Diese Art von Treffen wurde nicht im Zeitplan, aber im Sprintplanungsmeeting eingeplant.

13.3 Vergleich SOLL-/IST-Aufwand

Erreicht wurden total 482.08 Stunden verwendet, welche pro Woche folgendermassen auf das Team aufgeteilt wurden:

IST	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**Total**
Philipp	8.00	15.75	20.25	16.25	27.50	14.50	20.00	6.25	13.00	14.50	15.50	27.08	21.50	25.50	8.25	**253.83**
Roman	3.00	14.00	20.50	12.75	25.50	14.25	13.50	12.25	13.75	11.00	11.50	23.08	15.17	23.00	15.00	**228.25**

Philipp hat etwas mehr Zeit investiert, wovon aber ein Teil daher kommt, dass Roman einmal für zwei Tage abwesend war. Philipp hat ebenfalls beim Implementieren vom Follow-Up-Feature im Prototyp 8h länger investiert als geplant war.

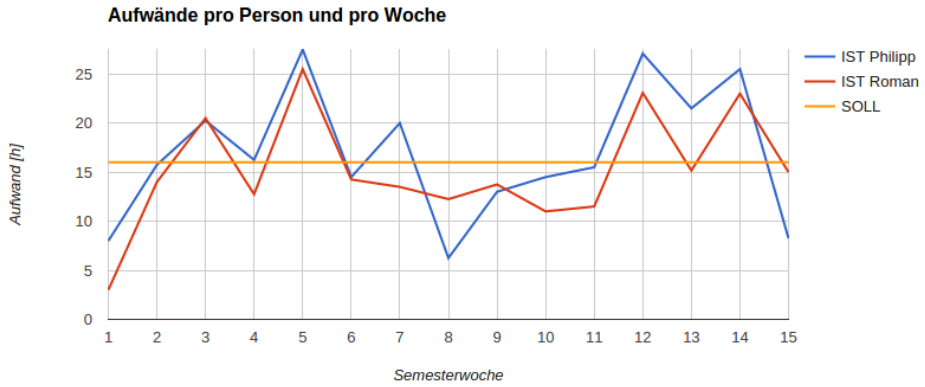


Abbildung 38: Aufwände pro Person pro Woche

Erkennbar ist die zweiwöchentliche Schwankung der Aufwände. Dies ist ein Resultat der zweiwöchigen Sprints: in fast jeder zweiten Woche wurde nebst dem üblichen Aufwand noch ein Sprintmeeting abgehalten, welches zusätzlich Zeit beanspruchte. Dazu kommt, dass der Druck kurz vor Sprintende höher ist und entsprechend mehr Zeit investiert wird als zu Beginn eines Sprints.

Die “Senke” in der Mitte des Projektes (ca. Semesterwochen 7 bis 10) umfasst die Sprints, in welchen es primär um die Erstellung der Mockups ging, welche jeweils schneller als geplant fertiggestellt werden konnten. Die beiden Spitzen in Woche 5 und in den Wochen 12 bis 14 umfasste intensive Programmierphasen; in der ersten wurde noch versucht, das existierende Backend zu verbessern, was viel Zeit beanspruchte. In letzteren ging es darum, die Mockups und die Funktionen umzusetzen, was ebenfalls in vermehrtem Aufwand resultierte. Vor allem im Sprint 6 (Wochen 13 und 14) wurden einige Änderungen in letzter Minute gefordert.

14 Protokolle

14.1 User-Tests

Es wurden mehrere User-Tests durchgeführt, wobei die meisten jeweils lediglich ein Feature oder einen bestimmten Screen abdeckten. Nach dem Fertigstellen des ersten Klickbaren Prototypen wurden allerdings drei umfassendere User-Tests auf deren Basis vorgenommen, womit der ganze Ablauf (ohne Follow-Up) getestet werden konnte.

Ausgangslage

“Du bist Alfred (Männlich, 58 Jahre alt). Du arbeitest im Servicebereich. Durch deinen Arbeitsort bist du ziemlich bewandert mit technischen Dingen und benutzt fast jeden Tag einen Computer. Du bist normalgewichtig und einigermassen fit für dein Alter.

Seit einigen Tagen leidest du an Kopf- und Bauchschmerzen und bist inzwischen etwas besorgt darüber.

Von einer Arbeitskollegin hast du von "PocketDoc" gehört und denkst, dass dies genau das Richtige für dich ist.”

Aufgabe

“Stelle mit der App PocketDoc sicher, dass es sich nur um eine Bagatellkrankheit handelt.”

14.1.1 Jan

Testperson: Jan, 22, Informatik-Student HSR
Testgerät: iPhone 5, Safari

Ablauf:

- Startet App
- Sieht den "Diagnose jetzt starten"-Button bevor er den Text durchliesst, fragt ob das der richtige Button sei
- Sieht "Register/Login" und fragt, ob er sich einloggen muss. Es ist ihm nicht klar ob man die App auch ohne Account benutzen kann. Vermutet aber ja, da der Button unten existiert. ("Könnte aber auch auf der nächsten Seite kommen, dass man sich einloggen oder registrieren soll...")
- Startet Diagnose
- Gibt die Persönlichen Daten ein
- Beantwortet erste Frage mit Nein (Fieber?)
- Beantwortet die zweite Frage mit Ja (Schmerzen?)
- Beantwortet die dritte Frage mit Weiss nicht (Grosse Schmerzen?)
- Beantwortet die vierte Frage mit Ja (Kopfschmerzen?)
- Zögert bei der Diagnose, akzeptiert sie aber (Erkältung)
- "Die App meint, ich soll zuhause bleiben”, befindet die Aufgabe als erledigt

Bemerkungen

- Zu viel Text beim Beginn
- Es ist nicht 100% klar ob man sich registrieren muss oder nicht

14.1.2 Sarah

Testperson: Sarah, 26, Psychologiestudentin UZH
Testgerät: Samsung Galaxy 5, Chrome

Ablauf:

- Startet App
- Liest Text durch
- Startet Diagnose
- Gibt Daten ein
- Antwortet “Nein” (Fieber?)
- Antwortet “Ja” (Schmerzen?)
- Grosse Schmerzen? → Ist sich nicht sicher wie gross sie sind. Klickt auf “?”
- Nächste Frage erscheint. Fragt nach, ob “?” nicht “Hilfe” sei
- Klickt auf Ja (Kopfschmerzen?)
- Akzeptiert Diagnose (Erkältung)
- Klickt auf “Registrieren” (Grund: Möchte das Follow-Up durchführen, da Alfred besorgt ist)
- Gibt Daten ein (“Warum muss ich die Angaben nochmal machen?”)
- Klickt “Registrieren”
- Aktiviert Follow-Up

Bemerkungen

- “Logo ist herzig”, aber zu wenig umgesetzt (Stil nicht durchgezogen)
- “?” (für “weiss nicht”) wurde als Frage/Hilfe-Option angesehen
- Registrierungsseite zu voll, nicht klar warum Angaben benötigt werden
- Wenn registriert nach anonymen Durchgang → Daten sollten behalten werden (persönliche Angaben)

14.1.3 Christoph

Testperson: Christoph, 28, Informatik-Student HSR
Testgerät: Laptop, Chromium

Ablauf:

- Startet App
- Überfliegt den Einleitungstext.
- Startet Diagnose über “Diagnose jetzt starten”
- Gibt die persönlichen Daten beim ersten Screen ein
- Beantwortet die erste Frage mit Nein (Fieber?)
- Beantwortet die zweite Frage mit Ja (Schmerzen?)
- Beantwortet die dritte Frage mit Nein (Grosse Schmerzen?)
- Beantwortet die vierte Frage mit Ja (Kopfschmerzen?)
- Akzeptiert Diagnose (Erkältung)

Bemerkungen

- Spielt nach Beenden der eigentlichen Aufgabe den Ablauf noch einmal durch und möchte die Fragen anders beantworten. Durch die Limitierung des Mediums (Klickbarer Prototyp mit vorgegebenem Ablauf) ist dies aber nicht möglich.

14.2 Folgerungen

- Es wurde eine Kurzanleitung eingebaut, welche solange erscheint bis die erste Antwort gegeben wurde. So ist klarer, dass es sich bei “?” um “Weiss Nicht” handelt.
- Zudem wurde nach Bestätigung durch den UX-Experten von atfront.ch ein Teil der Einleitung auf der Startseite auf eine “About”-Seite ausgelagert. Dies hat den Nebeneffekt, dass auch eingeloggte User noch auf diesen Text Zugriff haben.
- Die Daten, welche ein anonymer User eingab, werden bei der (optionalen) Registrierung von der Diagnoseseite aus behalten und bereits in das Registrierungsformular abgefüllt.
- Die Texte, welche den Ablauf erklären, sind sehr wichtig und sollten genauer formuliert werden. Hier würde es sich lohnen, einen Experten zu engagieren, welcher sämtliche Texte einheitlich formuliert und den richtigen Ton und Länge trifft.

15 Aufgabenstellung

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Aufgabenstellung zur Studienarbeit FS 2015

„PocketDok - App für die Selbstdiagnose“

Gruppe: Christen, Philipp / Eichenberger, Roman

Vision

Wir haben die Vision, dass Menschen sich jederzeit, überall und unabhängig von ihrem Versicherungsmodell einen ersten medizinischen Rat holen können. Dabei fasziniert uns die Herausforderung, eine grundlegende ärztliche Tätigkeit zu automatisieren, sowie das sich damit eröffnende gesellschaftliche und ökonomische Potential.
PocketDok ist ein Web- und App-basiertes medizinisches Expertensystem mit dessen Hilfe sich Bagatellerkrankungen erkennen und von schwereren Erkrankungen unterscheiden lassen. Damit können User zuverlässig entscheiden, ob sie sich selber behandeln können oder einen Arzt aufsuchen sollten. Im Falle einer Bagatellkrankheit gibt das System Therapieempfehlungen ab und erlaubt dem User die Evaluation seines Heilungsverlaufs.
Unsere einmalige Chance besteht darin, als Erste ein echtes Empowerment des Nutzers in einem der wichtigsten Lebensbereiche, der Gesundheit, zu bewirken und gleichzeitig die Kosten für den Patienten und das Gesundheitswesen zu reduzieren. Unser System leistet einen entscheidenden Beitrag zum Patienten-Empowerment, indem es die Selbstbestimmung erkrankter Menschen unterstützt. Als weiterer Effekt werden teure medizinische Institutionen von Bagatellfällen entlastet, was deren Verfügbarkeit erhöht und Gesundheitskosten spart.

Stand

Wir haben einen Algorithmus entwickelt, der bisher rund fünfzig Bagatellerkrankungen anhand von Ja-/Nein Symptomabfragen erkennt. Ferner wurde ein einfacher, auf Webtechnologien basierender *Proof of Concept (Prototyp)* mit beschränktem Funktionsumfang realisiert, welcher bei ersten Tests mit medizinischen Fachexperten und Medizinlaien auf positive Resonanz gestossen ist. Der Proof of Concept steht unter http://quiet-samurai-3688.herokuapp.com/ zur Verfügung.

Aufgabe

Basierend auf dem bestehenden Algorithmus und einem im Rahmen einer Studienarbeit erstellten Backend [1] soll im Rahmen einer Folgearbeit das Frontend der Applikation neu konzipiert, designed und umgesetzt werden. Der umzusetzende Funktionsumfang kann - falls gewünscht - bei Projektstart abschliessend definiert werden. Für die Umsetzung sind wir (die Industriepartner) bezüglich einzusetzenden Technologien offen. Das Frontend kann beispielsweise auch für iOs, Android oder Windows Phone umgesetzt werden. Wir suchen Studierende mit Talent für

Softwareentwicklung und Usability. Als Industriepartner verfügen wir über langjährige und ausgewiesene Erfahrung in den Bereichen Human Factors, Software Engineering, Project Management nach Scrum und Medizin.

Allgemeine Vorgaben

Die Arbeit ist gemäss den allgemeinen Vorgaben [2] durchzuführen. Dies beinhaltet auch Vorgaben zur Berichtsgestaltung.

Termine

16. Feb. 2015	Arbeitsbeginn
29. Mai 2015	Abgabe des Berichts an den Betreuer (bis 17:00)

Weitere Termine: siehe Terminangaben auf dem HSR-Web (intern).

Betreuung

Betreuer: Prof. Dr. Eduard Glatz, Email: eglatz@hsr.ch

Während der Durchführung der Arbeit findet nach Möglichkeit regelmässig jede Woche eine Besprechung mit dem Betreuer statt. Dazu werden entsprechende Termine bei Arbeitsbeginn festgelegt. Die Studierenden verfassen vor jder Besprechun eine Traktandenliste, die spätestens am Vortag dem Betreuer per Email zu senden ist. Jede Besprechung wird von den Studierenden in einem Protokoll dokumentiert.

Praxispartner

Forventis GmbH, Kontaktperson: Dr. med. Samuel Huber (Email: huber@forventis.ch)

Panter AG, Kontaktperson: Hr. Flavio Trolese (Email: tro@panter.ch)

Referenzen

[1] Bourquin, N.; Frischknecht, O.; „PocketDok - App für die Selbstdiagnose“; Studienarbeit FS 2014, HSR - Hochschule für Technik Rapperswil.

[2] Glatz, E., „Vorgaben zur Arbeitsdurchführung“, Ausgabe des 9. Februar 2015.

Rapperswil, den 9. Feb. 2015

Eduard Glatz

16 Anleitungen

16.1 Heroku

Durch das automatische Deployment nach Heroku nach Pushes auf das verwendete Github-Repository ist der Aufwand dafür praktisch sehr klein.

Um das automatische Deployment (eigentlich ein Hook von Heroku auf Github) zu aktivieren, muss das entsprechende Github-Repository bei den Einstellungen der Heroku-App verlinkt werden:

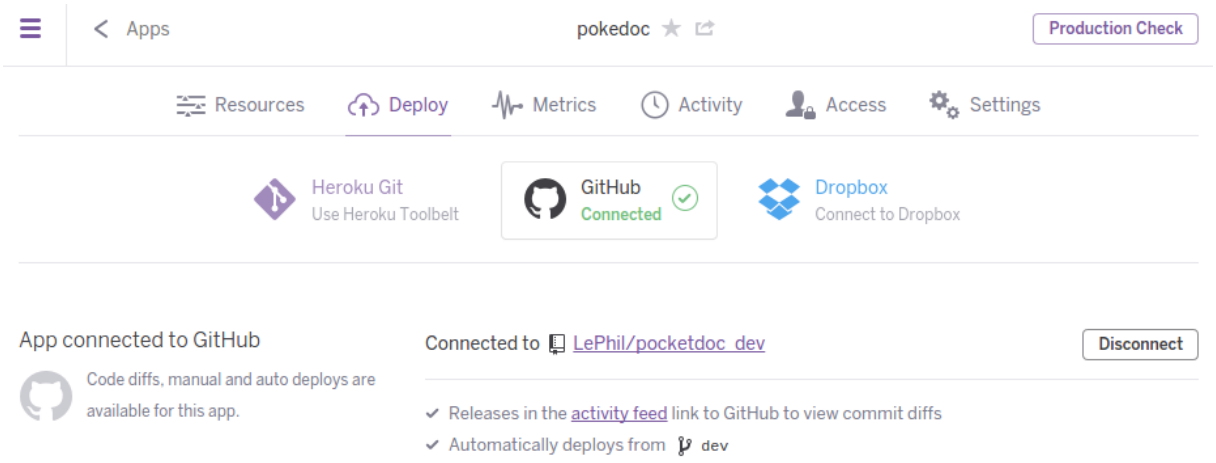


Abbildung 39: Anleitung Heroku

Damit die App auch auf Heroku läuft ist eine *procfile*-Datei[1] nötig, welche Heroku mitteilt, um was für eine Art Applikation es sich handelt und welche Prozesse ausgeführt werden müssen. Bei PocketDoc ist dies eine Web-App, es muss der NodeJS-Prozess gestartet und auf main.js zugegriffen werden, welches sich um alles weitere kümmert:

```
web: node main.js
```

16.2 Lokales Setup

Für die Entwicklung wurde ein ExpressJS-Server verwendet, welche die Daten lokal bereitstellt. Mit NodeJS und Bower standen bewährte Tools bereit, welche schnell und einfach installiert werden können und den Deployment-Prozess vereinfachen.

16.2.1 Voraussetzungen

- git
- node.js (nodejs & npm)

16.2.2 Linux/Mac

Getestet auf

- Ubuntu 14.10
- elementaryOS 0.3
- Mac OSX Yosemite 10.10.3

Kurzfassung

```
git clone https://github.com/LePhil/pocketdoc_dev.git myFolder
cd myFolder/
```

[1] https://devcenter.heroku.com/articles/procfile

```
npm start
xdg-open http://localhost:5000
```

Diese Befehle erstellen eine lokale Kopie des Github-Repositories im Ordner *myFolder* und starten dann einen NPM-Task, welcher die benötigten Pakete installiert und einen lokalen expressJS-Server startet. Der letzte Befehl öffnet den Default-Webbroser und navigiert zur lokalen Web-App. Diese 4 Befehle setzen voraus, dass die benötigten Programme (git und NodeJS) bereits installiert und aufgesetzt wurden.

Ausführlich

1. Paketlisten aktualisieren:

   ```
   sudo apt-get update
   ```

2. git, NodeJS und den Node Package Manager (NPM) installieren

   ```
   sudo apt-get install git nodejs npm
   ```

3. Symlink für legacy naming (node statt nodejs) erstellen (nicht immer notwendig)

   ```
   sudo ln -s /usr/bin/nodejs /usr/bin/node
   ```

4. Ordner erstellen und betreten, in welchem das Repository angelegt werden soll

   ```
   mkdir git
   ```

   ```
   cd git
   ```

5. Repository von git clonen und in den Ordner *myFolder* verschieben

   ```
   git clone https://github.com/LePhil/pocketdoc_dev.git myFolder
   ```

6. Repository betreten (lokal)

   ```
   cd myFolder/
   ```

7. Lokalen Server starten (dieser Task wird auch die depencies installieren)

   ```
   npm start
   ```

8. Webbrowser öffnen, https://localhost:5000 besuchen, Fertig!

16.2.3 Windows

Unter Windows ist das Vorgehen dasselbe: Github-Repository klonen, mit NPM den Task starten, localhost:5000 besuchen.

16.3 Unit-Tests

Unit-Tests werden durch Karma durchgeführt. Durch das Aufrufen in der Konsole von

```
npm test
```

aus dem Root-Directory, wo das Repository gecloned wurde, wird Karma gestartet. Es ist so konfiguriert, dass es eine Config-Datei (test/karma.conf.js) benutzt, worin weitere Details beschrieben werden. Die verschiedenen Konfigurationsmöglichkeiten sind auf der Webseite von Karma beschrieben[2].

[2] https://karma-runner.github.io/0.12/config/configuration-file.html

Karme öffnet ein Browserfenster, in welchem der Frontend-Code ausgeführt wird. Die Ergebnisse werden in der Konsole angezeigt.

17 Story-Skripte

17.1 Story 1: “Alfred”

Profil:

- männlich
- 62 Jahre alt
- arbeitet im Servicebereich, generell etwas stressig
- besitzt einen Universitätsabschluss
- spricht fliessend Deutsch und Englisch
- ist kein “digital native”, ist aber grundsätzlich vertraut mit technischen Geräten
- besitzt bereits einen Account bei PocketDoc und hat es schon ein paar Mal benutzt
- ist normalgewichtig und ziemlich fit für sein Alter

Alfred leidet seit einigen Tagen an leichten Kopf- und Nackenschmerzen. Er ist inzwischen etwas besorgt und fragt sich, ob er einen Arzt aufsuchen sollte.

Ablauf:

1. Alfred öffnet die Seite pocketdoc.ch auf seinem Smartphone und landet auf der Startseite
    a. Da er erst kürzlich eine Diagnose durchgeführt hat, ist er bereits eingeloggt
2. Er klickt auf “Diagnose jetzt starten!”, da er ja eine Diagnose durchführen lassen will
3. Intro-Screen zeigt die persönlichen Infos an:
    a. Diese Diagnose ist… Für mich
    b. Name: Alfred
    c. Geschlecht: Männlich
    d. Altersbereich: 51 bis 70 Jahre
    e. Sprache: Deutsch
4. Diese Angaben sind korrekt, er startet die Diagnose mit diesen Einstellungen “**Diagnose starten**”
5. Es taucht die erste Frage auf:
   “Hat das Problem innerhalb der letzten 24 Stunden angefangen?”
   Alfred antwortet mit “Nein”, da er die Beschwerden schon länger hat.
6. Die zweite Frage erscheint:
   “Hat das Problem innerhalb der letzten 2 Wochen angefangen?”
   Alfred antwortet mit “Ja”.
7. Die dritte Frage, um den Zeitpunkt genauer zu bestimmen:
   “**Hat das Problem innerhalb der letzten Woche angefangen?**”
   Alfert antwortet “**Ja**”, da dies ziemlich genau zutrifft.
8. Die vierte Frage taucht auf, diesmal eine Symptom Frage:
   “Haben Sie Fieber?”
   Alfred antwortet mit “Nein”.
9. Fünfte Frage:
   “Haben Sie Schmerzen?”
   Alfred antwortet wahrheitsgemäss mit “Ja”.
10. Die sechste Frage erscheint:
    “Sind die Schmerzen leicht?”
    Alfred wählt nach kurzem Überlegen “Ja” aus.
11. Siebte Frage:
    “**Sind die Schmerzen sehr leicht?**”
    Alfred wählt “**Nein**”, da die Schmerzen doch gut spürbar sind.

12. Achte Frage:
    “Haben Sie Sodbrennen?”
    Alfred hat kein Sodbrennen und antwortet deshalb mit “Nein”.
13. Neunte Frage:
    “Handelt es sich um Kopfschmerzen?”
    Dies trifft zu und er beantwortet die Frage mit “Ja”.
14. Es erscheint ein Dialog mit einem Vorschlag:
    *Sie haben möglicherweise **zu wenig getrunken.** Könnte dies zutreffen?*
    *Alfred ist mit der Antwort nicht zufrieden, weil er regelmässig Wasser getrunken hat. Er klickt auf den Button “***Nein**” und die Befragung geht weiter, der Dialog verschwindet.
15. Zehnte Frage:
    “Sind die Schmerzen stetig und ununterbrochen da?”
    Leider ist dies der Fall und Alfred klickt auf “**Ja**”.
16. Elfte Frage:
    “Gibt es etwas, was die Schmerzen viel schlimmer werden lässt?”
    “Zum Glück nicht!”, denkt Alfred und klickt auf “Nein”.
17. Die zwölfte Frage erscheint:
    “Leiden Sie unter Nackenschmerzen?”
    “Ja”, antwortet Alfred.
18. Wieder erscheint ein Dialog mit einer Diagnose:
    *Wie es aussieht könnten Sie an **Spannungskopfschmerzen** leiden. Diese Kopfschmerzen werden durch muskuläre Verspannungen verursacht. Könnte dies zutreffen?*
    *Diese Diagnose und Erklärung leuchten Alfred ein und er klickt auf den Button “***Ja**”
19. Der Dialog wird geschlossen, die beantworteten Fragen verschwinden und es erscheinen einige Details zur Diagnose:
    Diagnose
    Sie könnten an Spannungskopfschmerzen leiden. Diese Kopfschmerzen werden durch muskuläre Verspannungen verursacht.

    Handlungsempfehlung
    ***Sie können sich selber behandeln**.*
    Nehmen Sie ein Schmerzmittel, wie Aspirin oder Dafalgan.
    Falls dies nicht hilft oder sich die Symptome verstärken empfehlen wir Ihnen, trotzdem einen Arzt aufzusuchen.
    Follow-Up
    Soll sich die App in 4h wieder nach Ihrem Zustand erkundigen?
    *Zur Auswahl stehen zwei Optionen: “***Ja”** und “**Nein”**. Alfred ist mit diesem Resultat zufrieden und klickt auf “Nein”.
20. Alfred wird auf die Startseite weitergeleitet, macht sich eine Tasse Tee und nimmt ein Aspirin.

17.2 Story 2 - “Bruce”

Profil:

- männlich
- 29 Jahre alt
- von Beruf Erbe
- sportlich, täglich mehrstündiges Fitnesstraining
- noch kein registrierter Benutzer bei PocketDoc

Bruce hat seit wenigen Tagen starke Schmerzen beim Atmen, Husten und Fieber. Durch seinen Butler hörte er zum ersten Mal von der App “PocketDoc” und entscheidet sich dazu, sie auf dem Weg nach Hause in einem Taxi auszuprobieren.

Ablauf:

1. Bruce startet die App
2. Er liest sich kurz den Einführungstext durch und will eine Diagnose starten, ohne sich anzumelden. Dazu klickt er auf “**Diagnose jetzt starten**”.
3. Der Intro-Screen zeigt ein Formular an:
    a. Geschlecht
    b. Altersbereich
    c. Sprache: Deutsch
4. Bruce passt das Geschlecht auf **Männlich** und den Altersbereich auf **26-50 Jahre** an, und bestätigt dies mit “**Diagnose starten**”.
5. Die erste Frage wird angezeigt:
   “Hat das Problem innerhalb der letzten 24 Stunden angefangen?”
   Bruce wählt “Nein” aus, da er sich sicher ist, erst vor 3 Tagen die Beschwerden gehabt zu haben.
6. Nächste Frage:
   “**Hat das Problem innerhalb der letzten 2 Wochen angefangen?**”
   Bruce stimmt dem zu (“**Ja**”).
7. Die App möchte es noch etwas genauer wissen:
   “**Hat das Problem innerhalb der letzten Woche angefangen?**”
   Auch hier stimmt er wieder mit “**Ja**” zu.
8. Die Frage verschwindet und es wird die nächste angezeigt:
   “**Haben Sie Fieber?**”
   Bruce antwortet “**Ja**”.
9. Nächste Frage:
   “**Hohes Fieber?**”
   Die Antwort ist “**Ja**”, da Bruce sein Fieber bereits gemessen hat und er auf 39.2°C kam.
10. Die nächste Frage,
    “**Haben Sie Schmerzen?**”,
    beantwortet er wieder mit “**Ja**”.
11. Nächste Frage:
    “**Sind die Schmerzen leicht?**”
    Aus den möglichen Antworten wählt Bruce “**Nein**” aus.
12. Nächste Frage:
    “**Sind die Schmerzen stark?**”
    Bruce antwortet mit “**Ja**”.
13. Wieder geht die App ins Detail:
    “**Sind die Schmerzen sehr stark?**”

Bruce antwortet “**Nein**”, da sie zwar stark sind, aber er schon schlimmere Schmerzen hatte.

14. Die nachfolgende Frage erscheint:
    “**Sind die Schmerzen stetig und ununterbrochen da?**”
    “**Ja**”, antwortet Bruce.
15. Fast augenblicklich erscheint die nächste Frage:
    “**Gibt es etwas, was die Schmerzen viel schlimmer werden lässt?**”
    Dies beantwortet er mit “**Nein**”.
16. Die Frage danach benötigt ein wenig länger:
    “**Haben Sie Husten?**”
    Bruce antwortet hier mit “**Ja**”.
17. Nächste Frage:
    “**Haben Sie Brustschmerzen?**”
    “**Ja**”, antwortet er.
18. Bevor die nächste Frage erscheint, wird der Bildschirm abgedunkelt und ein Dialog erscheint:
    *Sie könnten eine **Lungenentzündung** haben. Könnte dies zutreffen?*
    ***Bruce klickt auf* Ja**, da dies für ihn - leider - plausibel klingt.
19. Die Diagnoseseite erscheint:
    Diagnosis
    *Sie könnten eine **Lungenentzündung** haben.*

    Handlungsempfehlung
    Suchen Sie sofort einen Arzt auf.

    Follow-Up
    Soll Sich die App in 4h wieder nach Ihrem Zustand erkundigen?
20. Bruce ist durch die Handlungsempfehlung und Diagnose verunsichert, kann aber momentan nicht zum Arzt, da er gerade an seinem Anwesen ankommt und er dort Gäste erwartet. Er will sich aber eine Erinnerung an ein Follow-Up schicken lassen und klickt auf **Ja**.
21. Ein Dialog erscheint, wo er sich einloggen oder registrieren kann. Da er noch keinen Account bei PocketDoc besitzt, auf “**Registrieren**” statt auf “Login”.
22. Die Registrierungsseite erscheint. Es wird seine Emailadresse, ein Passwort und sein Name verlangt, dazu sein Geschlecht, Altersbereich und seine bevorzugte Sprache. Die letzten drei sind bereits auf seine vorherigen Einstellungen voreingestellt. Bruce füllt die anderen Felder mit seinen Daten:
    Email: “bruce@wayne-enterprise.com“
    Passwort: ein super-sicheres Passwort aus seiner Passwort-App
    Name: mrwayne
    Danach klickt er die Checkbox “Ich stimme den AGBs zu” an und bestätigt anschliessend seine Eingaben mit “**Registrieren**”. Auf die Nachfrage, ob seine Email-Adresse korrekt ist, antwortet er mit **Ja**.
23. Bruce landet auf der Startseite.
    Unter “*Ausstehende Follow-Ups*” ist ein Eintrag zu sehen, worauf eine Zeitanzeige von 4 Stunden im Sekundentakt nach unten zählt. Bruce hätte die Möglichkeit, den Follow-Up sofort zu starten, aber mittlerweile ist er vor seiner Villa angekommen. Er legt sein Smartphone zur Seite. bezahlt den Taxifahrer und verlässt das Taxi.
24. Kurz nachdem seine Gäste sein Anwesen verlassen haben und sich Bruce ins Bett legte, vibriert sein Telefon. Er hat eine Mail von PocketDoc erhalten.
    Bruce öffnet die Mail:

Hallo Bruce!
Du hast vor kurzem eine Diagnose bei PocketDoc durchgeführt und hast ein Follow-Up eingetragen. Mittlerweile sind über 4 Stunden vergangen und es wäre bereit für dich :)
Klicke auf diesen Link (http://pocketdoc.ch/followup/3wec32rfea39f22c) um dein Follow-Up zu starten. Du kannst dies natürlich auch später durchführen.
Liebe Grüsse,
PocketDoc

25. Fast hätte er Follow-Up vergessen! Er klickt den Link an und findet sich auf der Startseite der Applikation wieder. Der Eintrag unter "Ausstehende Follow-Ups" hat den Countdown beendet und ist nun bereit.
26. Bruce klickt auf den Play-Button neben dem Eintrag, da er das Follow-Up jetzt starten möchte.
27. Die erste Frage erscheint:
    "**Haben Sie Fieber?**"
    Bruce antwortet "**Ja**", da sich dies nicht geändert hat. Er hat zwar die Temperatur seit der letzten Diagnose nicht mehr gemessen, aber es fühlt sich definitiv noch so an, wenn nicht sogar noch schlimmer.
28. Die zweite Frage taucht auf:
    "**Hohes Fieber?**"
    Die Antwort ist auch hier wieder "**Ja**"
29. Die nächste Frage,
    "**Haben Sie Schmerzen?**",
    beantwortet er wieder mit "**Ja**".
30. Daraufhin erscheint Frage Nummer vier:
    "**Sind die Schmerzen leicht?**"
    Aus den möglichen Antworten wählt Bruce "**Nein**" aus.
31. Die App fragt weiter:
    "**Sind die Schmerzen stark?**"
    Dies ist immer noch der Fall, also wählt Bruce "**Ja**".
32. Danach geht die App ins Detail:
    "**Sind die Schmerzen sehr stark?**"
    Da sich seine Situation während dem Treffen mit seinen Gästen nicht verbessert, sondern verschlimmert hat, wählt er hier "**Ja**".
33. Die nächste Frage erscheint:
    "**Sind die Schmerzen stetig und ununterbrochen da?**"
    "**Ja**", antwortet Bruce.
34. Eine weitere Frage taucht auf, diesmal etwas anders als im ersten Durchgang:
    "**Haben Sie Auswurf?**"
    Bruce denkt zurück an die letzten Stunden und erinnert sich, dass er doch einige Mal beim Husten etwas Auswurf hatte, also antwortet er hier mit "**Ja**".
35. Die nächste Frage erscheint:
    "**Haben Sie viel Auswurf?**"
    Bruce findet, dass er doch etwas mehr als nur ein wenig Auswurf hatte und antwortet auch hier mit "**Ja**".
36. Der Bildschirm wird wieder abgedunkelt und es erscheint ein Dialog:
    *Sie könnten eine **Lungenentzündung** haben. Könnte dies zutreffen?*
    ***Bruce hat das bereits befürchtet und klickt auf* Ja**.
37. Die Diagnoseseite erscheint:

    Follow-Up abgeschlossen

*Sie könnten eine **Lungenentzündung** haben.*

Handlungsempfehlung
Suchen Sie sofort einen Arzt auf.

Follow-Up
Soll sich die App in 4h wieder nach Ihrem Zustand erkundigen?

Da Bruce ziemlich sicher ist, dass die Diagnose korrekt ist, wählt er “**Nein**” und landet wieder auf der Startseite.

38. Er ruft seinen Butler, bittet ihn doch schnell seinen Leibarzt zu informieren und fährt noch 5 Minuten später los zu ihm.