Karlsruher Institut für Technologie

myMD

QUALITÄTSSICHERUNG

Praxis der Softwareentwicklung WS2017/2018

Philipp Pelcz, Philipp Karcher, Jan-Luca Vettel

supervised by
Marc Aurel Kiefer

14. März 2018

Inhaltsverzeichnis

1 Einleitung

"myMD" ist eine mobile Anwendung, die es Patienten erlaubt, ihre Arztbriefe digital über Bluetooth von ihren Ärzten zu erhalten, um dann in der Applikation die darin enthaltenen Informationen, wie die Diagnose oder verschrieben Medikationen, einsehen zu können.

Die App wurde dabei mit Xamarin.Forms in C# und XAML für Android- und iOS-Geräte entwickelt.

Aufbauend auf den Tests, die im Pflichtenheft aufgestellt wurden, haben wir in der Phase der Qualitätssicherung unsere App auf Fehler überprüft. Dazu haben wir viele Unit-KomponentenTests geschrieben, aber auch manche Funktionen, für die Unit-Tests ungeeignet sind, manuell getestet.

In diesem Dokument werden diese Tests aufgeführt und kurz beschrieben. Außerdem werden auch nennenswerte Fehler und ihre Lösung kurz beschrieben.

2 Testvorbereitung

2.1 Zu testende Komponenten

Da sich unser System aus drei Subsystemen (**Model**, **ViewModel**, **View**) zusammensetzt, sollten alle drei Subsysteme getestet werden. Jedoch lässt sich die **View** nicht automatisch testen, da sie nur die Benutzeroberfläche und ihre Logik enthält. Außerdem ist sie sehr stark mit dem **ViewModel** verknüpft. Deshalb haben wir uns entschieden, das **ViewModel** und die **View** zusammen zu testen. Dadurch ergeben sich die UnitTests für das **Model** und die UITests für das **ViewModel** und die **View**.

2.2 Testvorgaben aus dem Pflichtenheft

Im folgenden werden alle Systemtests aus dem Pflichtenheft mit ihren Ergebnissen aufgeführt. Da manche Tests (vorallem Datenübertragungs- und Übersicht-Tests) eine Bluetoothübertragung benötigen konnten diese nicht vollständig automatisch durchgeführt werden. Denn das emulierte Testhandy bietet keine Möglichkeit einer Bluetoothdatenübertragung. Diese Tests wurden manuell getestet. Außerdem bietet Xamarin.Forms keine Möglichkeit UITests (Systemtests) für ein UWP an. Deshalb wurden die alle Tests, die die Desktop-Anwendung betreffen, auch manuell durchgeführt. Da wir uns bei der Implementierung an den, im Pflichtenheft vorgegebenen, Tests entlanggehangelt haben, laufen diese Tests fehlerfrei.

Legende:
A[BT0000] $\widehat{=}$ Test "BT000" wurde automatisch durchgeführt
M[BT0000] $\widehat{=}$ Test "BT000" wurde manuell durchgeführt
E[BT0000] $\widehat{=}$ Die Funktion, die den Test "BT000" benötigt wurde entfernt, nicht implementiert oder modifiziert

2.2.1 Basis Testfälle (BT)

Patientenseitige Datenübertragung

M[BT1010] Desktop Anwendung sendet eine Datei an die myMD App.

E[BT1020] Die Übertragung wird vom Nutzer unterbrochen.

M[BT1030] Desktop Anwendung sendet eine unzulässige Datei (nicht unterstütztes Dateiformat).

M[BT1040] Die Übertragung wird durch äußere Umstände abgebrochen.

Darstellung

M[BT2010] Ein neuer Arztbrief wird in eine leere Übersicht geladen.

M[BT2020] Ein neuer Arztbrief wird in bereits gefüllte Übersicht geladen.

M[BT2030] Ein digitaler Arztbrief wird angetippt/geöffnet.

M[BT2040] Ein digitaler Arztbrief wird geschlossen.

M[BT2050] Ein digitaler Arztbrief wird gelöscht.

Einstellungen

A[BT3010] Das Nutzerprofil wird bearbeitet.

Desktop Anwendung

M[BT4010] Unzulässige Datei wird zum Senden ausgewählt.

M[BT4020] Kompatible Datei wird zum Senden ausgewählt.

M[BT4030] Senden wird erfolgreich abgeschlossen.

M[BT4040] Senden wird durch äußere Einflüsse unterbrochen.

M[BT4050] Senden wird durch Nutzer unterbrochen.

E[BT4060] Geräte in der Nähe werden gesucht.

E[BT4070] Nutzergerät wird als Empfänger gewählt.

2.2.2 Erweiterte Testfälle (ET)

Patientenseitige Datenübertragung

E[ET1010]	myMD App sendet eine Datei an die Desktop Anwendung.
E[ET1020]	myMD App sendet mehrere Dateien an die Desktop Anwendung.
M[ET1030]	Desktop Anwendung sendet mehrere Datei an die myMD App.
M[ET1040]	Die Übertragung mehrerer Daten wird durch äußere Umstände abgebrochen.
M[ET1050]	Die Übertragung mehrerer Daten wird durch den Nutzer abgebrochen.
E[ET1060]	Sensible Daten werden von der myMD App an die Desktop Anwendung gesendet.
E[ET1070]	Die Verbindung wird über NFC hergetellt.
E[ET1080]	Ein Profil wird auf ein anderes Gerät übertragen.

Darstellung

A[ET2010]	Medikament wird hinzugefügt.
A[ET2020]	Medikament wird editiert.
A[ET2030]	Medikament wird gelöscht.
E[ET2040]	Laborwerte werden hinzugefügt.
E[ET2050]	Laborwerte werden gelöscht.
E[ET2060]	Bilddatei wird hinzugefügt und dargestellt.
E[ET2070]	Bilddatei wird mit einem Arztbrief verknüpft.
E[ET2080]	Verknüpfung zwischen Arztbrief und Bilddatei wird getrennt.
E[ET2090]	Überprüfung, ob eine Bilddatei originalgetreu ist.
E[ET2100]	Mehrere Arztbriefe gruppieren.
E[ET2110]	Einzelne Arztbriefe aus bestehender Gruppe entfernen.
E[ET2120]	Nach eigenen Gruppen sortieren.
E[ET2130]	Überprüfen, ob Suchfunktion fehlerfrei funktioniert.

Einstellungen

A[ET3010] Neues Nutzerprofil wird angelegt.
E[ET3020] Ein weiteres Nutzerprofil wird angelegt.
E[ET3030] Ein Nutzerprofil wird gelöscht.
E[ET3040] Wechsel zwischen mehreren Nutzerprofilen.
E[ET3050] Einzelnen Arztbrief als sensibel markieren.
E[ET3060] Sensibiltäts-Markierung eines einzelnen Arztbriefes aufheben.
E[ET3070] Gruppe von Arztbriefen als sensibel markieren.
E[ET3080] Sensibiltäts-Markierung einer Gruppe von Arztbriefen aufheben.
E[ET3090] Regelmäßiger Arzttermin wird hinzugefügt.
E[ET3100] Regelmäßiger Arzttermin wird gelöscht.

Desktop Anwendung

M[ET4010] Datei mit falscher Versichertennummer wird gesendet.

3 Automatische Tests

3.1 Unit Tests

3.1.1 DatabaseModel

EntityDatabase

GetDataFromProfileTest	Überprüfung, ob die Daten eines Profils korrekt gespeichert werden.
DLetterGroupDLetter	Überprüfung, ob Arztbriefe korrekt gruppiert werden können.
EqualTest	Überprüfung, ob zwei Arztbriefgruppen gleich sind.
DeleteTest	Überprüfung, ob ein Arztbrief korrekt gelöscht wird.
GetDoctorTest	Überprüfung, der Arzt innerhalb eines Arztbriefes korrekt zurückgegeben wird.
GetProfilTest	Überprüfung, ein Profil aus der Datenbank korrekt zurückgegeben wird.

Database

MedLetterTest	Überprüfung, ob eine Medikation mit einem Arztbrief verknüpft werden kann
GroupTest	Überprüfung, ob Arztbriefe mit verknüpften Medikationen gruppiert werden können.

3.1.2 DataModel

MedicationTest

SameEqualTest	Überprüfung, ob die Überprüfung der Gleichheit zweier Medikamente korrekt ist.
CopiedEqualTest	Überprüfung, ob die Überprüfung der Gleichheit einer dublizierten Medikation korrekt ist.
NullEqualtest	Überprüfung, ob der Vergleich einer Medikation mit *null* korrekt ist.
NotEqualTest	Überprüfung, ob die Überprüfung zweier ungleicher Medikamente auf Gleichheit korrekt ist.
LetterTest	Überprüfung, ob eine Medikation korrekt mit einem Arztbrief verknüpft wird.
DeleteTest	Überprüfung, ob eine Medikation, die mit einem Arztbrief verknüpft ist, korrekt gelöscht wird.

DoctorTest

SameEqualTest	Überprüfung, ob die Überprüfung der Gleichheit zweier Ärzte korrekt ist.
CopiedEqualTest	Überprüfung, ob die Überprüfung der Gleichheit von dublizierten Ärzte korrekt ist.
NullEqualtest	Überprüfung, ob der Vergleich eines Arztes mit *null* korrekt ist.
NotEqualTest	Überprüfung, ob die Überprüfung zweier ungleicher Ärzte auf Gleichheit korrekt ist.
IDoctorToDoctor	Überprüfung, ob ein *IDoctor*-Objekt korrekt zu einem *Doctor*-Objekt konvertiert wird.

DoctorTest

SameEqualTest	Überprüfung, ob die Überprüfung der Gleichheit zweier Ärzte korrekt ist.
CopiedEqualTest	Überprüfung, ob die Überprüfung der Gleichheit von dublizierten Ärzten korrekt ist.
NullEqualtest	Überprüfung, ob der Vergleich eines Arztes mit *null* korrekt ist.
NotEqualTest	Überprüfung, ob die Überprüfung zweier ungleicher Ärzte auf Gleichheit korrekt ist.
IDoctorToDoctor	Überprüfung, ob ein *IDoctor*-Objekt korrekt zu einem *Doctor*-Objekt konvertiert wird.

DoctorsLetterTest

SameEqualTest	Überprüfung, ob die Überprüfung der Gleichheit zweier Arztbriefe korrekt ist.
CopiedEqualTest	Überprüfung, ob die Überprüfung der Gleichheit von dublizierten Arztbriefen korrekt ist.
NullEqualtest	Überprüfung, ob der Vergleich eines Arztbriefes mit *null* korrekt ist.
NotEqualTest	Überprüfung, ob die Überprüfung zweier ungleicher Arztbriefe auf Gleichheit korrekt ist.
DeleteTest	Überprüfung, ob Arztbrief korrekt gelöscht wird.
MedicationTest	Überprüfung, ob Arztbrief verknüpfte Medikationen korrekt behandelt.
GroupTest	Überprüfung, ob sich Arztbriefe korrekt gruppieren lassen.

DoctorsLetterGroupTest

DateTest	Überprüfung, ob eine Gruppe von Arztbriefen, richtige Daten behält.
AddTest	Überprüfung, ob eine Gruppe von Arztbriefen korrekt zu der Datenbank hinzugefügt wird.
CompareLetterTest	Überprüfung, ob die Überprüfung zweier gleicher Arztbriefgruppen korrekt funktioniert.
FirstLastDateTest	Überprüfung, ob die das früheste und das späteste Datum einer Arztbriefgruppe korrekt erkannt wird.
DeleteTest	Überprüfung, ob eine Arztbriefgruppe korrekt gelöscht wird.

3.1.3 FileHelper

WriteFromBytesTest	Überprüfung, ob eine Datei korrekt in die Datenbank geschrieben wird.

3.1.4 ModelFacadeTest

ProfileTest	Überprüfung, ob ein Profil korrekt aktiviert wird.
MedicationTest	Überprüfung, ob alle Medikationen von der Fassade zurückgegeben werden.
GroupTest	Überprüfung, ob alle Gruppen von Arztbriefen zurückgegeben werden.
UpdateTest	Überprüfung, ob die Fassade die Gruppen von Arztbriefen korrekt aktualisiert.
DeleteTest	Überprüfung, ob die Fassade eine Medikation korrekt löscht.

3.1.5 Parser

ParserFacadeTest

ParseLetterToFile	Überprüfung, ob eine Datei korrekt in das ursprüngliche Dateiformat geschrieben wird.
ParseFileToDB	Überprüfung, ob eine Datei korrekt in die Datenbank geparsed wird.
ParseInvalidFile	Überprüfung, ob der Parser eine fehlerhafte Datei korrekt behandelt.

HL7ParserTest

ParseFromDocument	Überprüfung, ob eine .hl7 Datei korrekt in die Datenbank geschrieben wird.
ParseSample	Überprüfung, ob eine Datei korrekt geparsed wird.
ParseHL7File	Überprüfung, ob eine .hl7 Datei korrekt in die Datenbank geschrieben wird.
ParseLetterToFile	Überprüfung, ob eine .hl7 Datei korrekt in das ursprüngliche Dateiformat geschrieben wird.
UnsupportedFile	Überprüfung, ob der Parser eine nichtunterstützte Datei korrekt behandelt.
InvalidFile	Überprüfung, ob der Parser eine fehlerhafte Datei korrekt behandelt.

FileToDatabaseParser

ParseFileTest	Überprüfung, ob eine Datei korrekt in die Datenbank geschrieben wird.

3.1.6 TransmissionModel

ListToArrayTest	Überprüfung, ob eine Liste an Byte-Arrays korrekt zu einer Datei zusammengesetzt wird.
EmptyListToArrayTest	Überprüfung, ob eine leere Liste an Byte-Arrays korrekt zu einer Datei zusammengesetzt wird.
GetNumberOfFilesTest	Überprüfung, ob die Anzahl der zu empfangenen Dateien korrekt ausgelesen wird.
GetReadCycles	Überprüfung, ob die Anzahl der Lesezyklen zu einer Datei korrekt ausgelesen wird.

3.2 UITests

Die folgenden Tests werden mit Hilfe der UITests von Xamarin.Forms realisiert. Da diese auf einem emuliertem Handy, auf dem die ausführbare Datei der App läuft, getestet werden, können von diesen Tests keine Codeüberdeckung o.Ä. ermittelt werden.

3.2.1 OverviewUITest

AppLaunches	Überprüfung, ob die App korrekt startet.
EveryElement	Überprüfung, ob jedes Element der View vorhanden ist.

3.2.2 MedicationUITest

CreateNewMed	Überprüfung, ob die View/ViewModel das erstellen einer neuen Medikation korrekt durchführt.
DeleteMed	Überprüfung, ob die View/ViewModel eine Medikation korrekt löscht.
EditMed	Überprüfung, ob die View/ViewModel eine Medikation korrekt editiert.

3.2.3 SendDataUITests

AllButtonsThere	Überprüfung, ob alle Elemente des SendenTabs existieren.

3.2.4 ProfilUITests

CreateNewProfil	Überprüfung, ob ein neues Profil korrekt erstellt wird.
EditProfil	Überprüfung, ob ein Profil korrekt editiert wird.

4 Manuelle Tests

4.1 Testszenarien

Anhand der im Pflichtenheft erörterten Testszenarien haben wir die Testszenarien, dessen Funktionen wir auch umgesetzt haben,erfolgreich durchgeführt.

4.1.1 Typischer Arztbesuch

Der Nutzer geht wegen einer Beschwerde zum Arzt und lässt sich untersuchen. Sobald die Untersuchung fertig ist, verlangt der Nutzer, dass der Arzt ihm die eben aufgenommenen Daten für die myMD App zur Verfügung stellt. Der Arzt öffnet dann die Desktop Anwendung und schickt dem Patienten den Arztbrief auf die myMD App. Der Nutzer schaut sich dann kurz den neuen Eintrag in der Übersicht an, schließt die App und geht nach Hause.

[BT4060] Geräte in der Nähe werden gesucht.
[BT4070] Das Mobilgerät des Nutzers wird als Empfänger ausgewählt.
[BT4030] Senden wird erfolgreich abgeschlossen.
[BT2020] Der neue Arztbrief wird in bereits gefüllte Übersicht geladen.
[BT2030] Der neue Arztbrief wird geöffnet.
[BT2040] Der neue Arztbrief wird geschlossen.

4.1.2 Krankenhistorie wird überprüft

Der Nutzer ist daheim und schaut sich in der myMD App seine Übersicht an. Er klickt auf einige Arztbriefe und liest sich durch, was darin steht. Dann bemerkt er, dass er die verschriebenen Medikamente zu seinem gebrochenen Fuß in den Tab *Medikamente* hinzufügen sollte. Versehentlich gibt er zuerst eine falsche Menge an, korrigiert diese aber und schließt daraufhin die App.

[BT2030] Ein Arztbrief wird geöffnet.
[BT2040] Ein Arztbrief wird geschlossen.
[ET2010] Medikament wird hinzugefügt.
[ET2020] Medikament wird editiert.

4.2 Desktop-Anwendung

Das automatische Testen der Desktop-Anwendung hat sich als sehr schwierig herausgestellt, da die Unterstützung von Unit-Tests einer UWP-Anwendung nur sehr dürftig ausfällt. Die oben beschriebenen UITests, die für die App benutzt wurden, werden für eine UWP-Anwendung sogar überhaupt nicht angeboten. Um dennoch eine möglichst fehlerfreie Funktionsweise der Anwendung gewährleisten zu können, wurde diese ausgiebig manuell getestet. Da zugleich der Funktionsumfang der Desktop-Anwendung nur gering ausfällt, war dies eine legitime Vorgehensweise.

5 Verbleibende Bugs

Beschreibung	Priorität
Der Server auf der Desktop-Anwendung wird nicht gestoppt.	Mittel
Durch den physischen Zurück-Button lassen sich leere Medikamente erstellen.	Mittel
Server ist manchmal nicht entdeckbar, falls er über eine längere Zeit (>15 Minuten) läuft.	Hoch
Durch schnelles Klicken lassen sich manche Pages mehrmals öffnen.	Niedrig
Die App stürzt ab, wenn man mit einem Gerät ohne Bluetooth (z.B. Emulator) nach Geräten sucht.	Hoch
Die App schließt sich unerwartet, wenn man während dem Geräte suchen das Bluetooth deaktiviert.Leider konnten wir diesen Bug noch nicht reproduzieren.	Mittel

6 Interessante Bugs und ihre Lösung

Beschreibung: Beim ersten Start der App ist die App abgestürzt. Dies lag daran, dass die App ein aktives Profil benötigt um eine Entität (hier: ein neues leeres Profil) der Datenbank hinzuzufügen. Da zum ersten Start jedoch noch kein Profil existierte stürzte die App ab.
Lösung: Das neue Profil wird zuerst erstellt aktiviert bevor es in die Datenbank eingefügt wird.

Beschreibung: Das Löschen einer Datei war nicht möglich, da einem die Zugriffsrechte auf die Datei verweigert wurden.
Lösung: Es wurde nichtmehr der komplette Dateipfad, sondern der lokale Teilpfad gespeichert.

Beschreibung: Die App ist beim Öffnen des *SendDataTabs* abgestürzt, da wir im Konstruktor auf den *BluetoothAdapter* zugegriffen haben. Jedoch war zu diesem Zeitpunkt der Kontext noch nicht verfügbar.
Lösung: Der Adapter wird nichtmehr im Konstruktor eines *ViewModels* aufgerufen, sondern immer nur durch einen *ButtonClick*

Beschreibung: Die App lässt sich nichtmehr kompilieren, da der Linker sich über zu große Dateien beschwert. Die zu große Datei war das *Everest-Framework*, welches wir zum Parsen der empfangenen Datei nutzten. Da dieses Framework viele generische Datentypen nutzte und auch anbot, wuchs die Dateigröße beim compilieren enorm.
Lösung: Das *Everest-Framework* wurde aus dem Projekt geschmissen und wir schrieben unseren eigenen Parser.

Beschreibung: Es ließ sich auf dem Handy kein *GATT-Server* zur Datenübertragung erstellen. Dies lag daran, dass das Handy den *Bluetooth-State* als "UNKNOWN" erkannte, obwohl dieser "POWERED-ON" sein sollte. Dies war ein Bug aus dem Bluetooth-Framework, welches wir für die Datenübertragung nutzten.
Lösung: Der *GATT-Server* wird nun auf der Desktop-Anwendung gestartet und die App

ließt die benötigten Dateien aus der Dektop-Anwendung.

Beschreibung: Wir wollten gefundene Geräte nur einmal in der Liste darstellen und haben dafür das Attribut ListRepeatedBroadcast auf false gesetzt. Dadurch wurde ein Gerät aber für diese ServerSession nur noch einmal angezeigt. Dies war ein Problem, falls man längere Sessions geführt hat oder erneut nach verfügbaren Geräten gesucht hat
Lösung: Wir haben das oben genannte Attribut auf true gesetzt und unsere eigene Logik für das mehrmalige Aufzählen von dem gleichen Gerät implementiert.

7 Anhang

7.1 Nutzerstudie

Diese Nutzerstudie soll uns Informationen über die Nutzerfreundlichkeit geben. Dabei wurde jedem Tester gesagt welche Aktion er ausführen soll und wir dokumentierten den Erfolg über diese Aktionen. *Hinweis: Nicht alle Tests konnten von allen Testern durchgeführt werden, da nicht alle Tester Zugriff auf die Desktop-Anwendung hatten.*
Legende:
X $\widehat{=}$ Aktion wurde erfolgreich durchgeführt
F $\widehat{=}$ Aktion konnte nicht durchgeführt werden

Alter: 22
Beruf: Azubi, Kauffrau für Büromanagement
Betriebssystem des Testgerätes: Android

Auszuführende Aktion	**Ergebnis**	**Kommentar**
App starten	X	
Namen im Profil festlegen	X	Es wurde mehrfach auf das Label des Namens getippt, statt auf den Bearbeiten-Button
Versicherungsnummer im Profil festlegen	X	
Neue Medikation hinzufügen	X	
Medikationdosis ändern	X	
Medikationstartdatum ändern	X	
Einnahmehäufigkeit ändern	X	
Editieren abbrechen	X	
Medikation löschen	X	
Geräte suchen	X	

Alter: 50
Beruf: Koch
Betriebssystem des Testgerätes: Android

Auszuführende Aktion	**Ergebnis**	**Kommentar**
App starten	X	
Namen im Profil festlegen	F	Weitere Verwirrung mit dem Bearbeiten-Button
Versicherungsnummer im Profil festlegen	X	
Neue Medikation hinzufügen	X	
Medikationdosis ändern	X	
Medikationstartdatum ändern	X	
Einnahmehäufigkeit ändern	X	
Editieren abbrechen	X	
Medikation löschen	F	Das Tippen und Halten eines Medikamenteneintrages wurde nicht verstanden
Geräte suchen	X	

Alter: 23
Beruf: Student, Wissentschaftskommunikation
Betriebssystem des Testgerätes: iOS

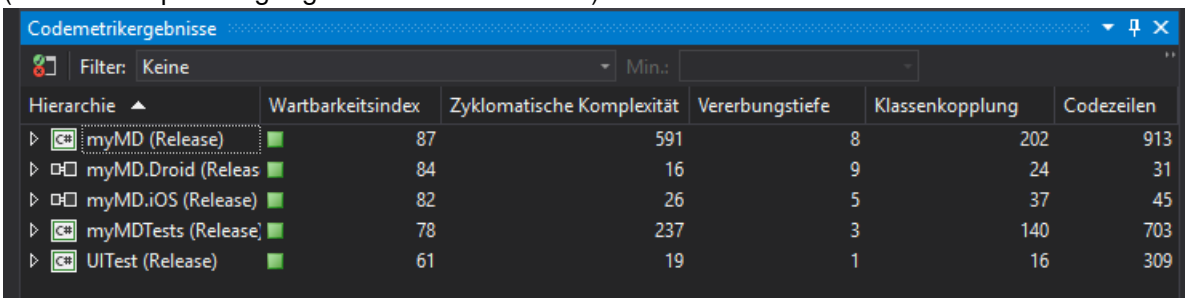
Auszuführende Aktion	**Ergebnis**	**Kommentar**
App starten	X	
Namen im Profil festlegen	F	Weitere Verwirrung mit dem Bearbeiten-Button
Versicherungsnummer im Profil festlegen	X	
Neue Medikation hinzufügen	X	
Medikationdosis ändern	X	
Medikationstartdatum ändern	X	
Einnahmehäufigkeit ändern	X	Wünscht sich eine Berechnung der Gesamt-Tagesdosis
Editieren abbrechen	X	
Medikation löschen	F	Swipe to delete wurde nicht verstanden
Geräte suchen	X	
Computer des Arztes verbinden	X	
Dateien empfangen	X	
Empfangenen Arztbrief öffnen	X	
Arztbrief schließen	X	
Arztbrief löschen	X	
Desktop-Anwendung starten	X	
Server starten	X	
Datei auswählen	X	
Dateiauswahl bestätigen	F	Fertig-Button wurde nicht gefunden

Alter: 50
Beruf: Hausfrau
Betriebssystem des Testgerätes: iOS

Auszuführende Aktion	**Ergebnis**	**Kommentar**
App starten	X	
Namen im Profil festlegen	F	Weitere Verwirrung mit dem Bearbeiten-Button
Versicherungsnummer im Profil festlegen	X	
Neue Medikation hinzufügen	X	
Medikationdosis ändern	X	
Medikationstartdatum ändern	X	
Einnahmehäufigkeit ändern	X	
Editieren abbrechen	X	
Medikation löschen	X	
Geräte suchen	F	Bluetooth wurde nicht angeschaltet
Computer des Arztes verbinden	X	
Dateien empfangen	X	
Empfangenen Arztbrief öffnen	X	
Arztbrief schließen	X	
Arztbrief löschen	X	
Desktop-Anwendung starten	X	
Server starten	X	
Datei auswählen	X	
Dateiauswahl bestätigen	F	Fertig-Button wurde nicht gefunden

7.2 Statistiken

Hier werden noch die CodeMetriken unseres Codes angegeben. Leider ist aber die Lines of Code Angabe fehlerhaft. Hier müsste eigentlich ein Wert von ca. 8000 zutreffender sein (Das Tool Supercharger gibt diesen Wert zurück).



Hierarchie	Wartbarkeitsindex	Zyklomatische Komplexität	Vererbungstiefe	Klassenkopplung	Codezeilen
myMD (Release)	87	591	8	202	913
myMD.Droid (Releas	84	16	9	24	31
myMD.iOS (Release)	82	26	5	37	45
myMDTests (Release)	78	237	3	140	703
UITest (Release)	61	19	1	16	309

Abbildung 7.1: Berechnete Codemetriken von VisualStudio

Glossar

App Als Mobile App (auf Deutsch meist in der Kurzform die App, eine Abkürzung für den Fachbegriff Applikation) wird eine Anwendungssoftware für Mobilgeräte beziehungsweise mobile Betriebssysteme bezeichnet. 4, 5, 13

Arztbrief Der Arztbrief, oft synonym als Epikrise, Entlassungsbrief, Patientenbrief oder Befundbericht bezeichnet, ist ein Transferdokument für die Kommunikation zwischen Ärzten. Der Arztbrief gibt einen zusammenfassenden Überblick über den Status des Patienten bei der Entlassung, einen Rückblick über den Krankheitsverlauf, die veranlasste Therapie, eine Interpretation des Geschehens zum Krankheitsverlauf im speziellen Fall. 5, 6, 13

Desktop Anwendung Als Desktop Anwendungen (auch Anwendungsprogramm, kurz Anwendung oder Applikation; englisch application software, kurz App) werden Computerprogramme bezeichnet, die genutzt werden, um eine nützliche oder gewünschte nicht systemtechnische Funktionalität zu bearbeiten oder zu unterstützen. Sie dienen der „Lösung von Benutzerproblemen“. 4, 5, 7, 13

Medikament Arzneimittel oder gleichbedeutend Medikamente (lateinisch medicamentum „Heilmittel“) sind Stoffe oder Stoffzusammensetzungen, die „zur Heilung oder zur Verhütung menschlicher oder tierischer Krankheiten bestimmt sind“ oder sich dazu eignen, physiologische Funktionen zu beeinflussen oder eine medizinische Diagnose zu ermöglichen. 6, 13

Nutzer Ein Benutzer (auch Endbenutzer, Bediener oder kurz Nutzer genannt sowie englisch User) ist eine Person, die ein Hilfs- oder Arbeitsmittel zur Erzielung eines Nutzens verwendet, beispielsweise für eine Zeitersparnis oder Kostensenkung. 4, 5, 13

Tab Eine Registerkarte, auch Reiter oder Tab genannt, ist ein Steuerelement einer grafischen Benutzeroberfläche, das einem Registerblatt aus Aktenschränken nachempfunden wurde . 13

Abbildungsverzeichnis