MedDoser - Visualisierung eines Medikationsplans auf dem RaspberryPi

Projektarbeit
von

Sebastian Büchler

Stefan Kuppelwieser

Fakultät Informatik und Mathematik
Ostbayerische Technische Hochschule Regensburg
(OTH Regensburg)

Betreuer: Prof. Dr. med. Georgios Raptis

Abgabedatum: 15. März 2018

Herr
Sebastian Büchler
Anzengruberstr. 2
93051 Regensburg

Stefan Kuppelwieser
Luitpoldstr. 15b
93047 Regensburg

Studiengang: Medizinische Informatik

1. Uns ist bekannt, dass dieses Exemplar der Projektarbeit als Prüfungsleistung in das Eigentum des Freistaates Bayern übergeht.
2. Wir erklären hiermit, dass wir diese Projektarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtlich und sinngemäße Zitate als solche gekennzeichnet haben.

Regensburg, den 15. März 2018

_____

Sebastian Büchler

_____

Stefan Kuppelwieser

Inhaltsverzeichnis

Abstract
Das Gerät *MedDoser* ist ein mit einem Touchscreen und weiteren Komponenten versehenes *RaspberryPi* und unterstützt Patienten bei der Medikamenteneinnahme. Es bietet akustische und visuelle Erinnerungsfunktionen und stellt eine Historie zur Verfügung, mit welcher der Verlauf der Medikation tages- und zeitpunktgenau nachvollzogen werden kann.

Die Informationen der jeweiligen Medikamente basieren hierbei auf dem vom Arzt verordneten *Medikationsplan* auf Papier, dessen Barcode eingescannt werden kann. Die eingelesenen Daten werden in einer Benutzeroberfläche bereitgestellt, die ausschließlich mit dem Finger bedient wird. Der Benutzer hat zudem die Möglichkeit, die Erinnerungszeitpunkte für jede Tageszeit manuell auf seine Bedürfnisse anzupassen.

1 Einführung

Der demografische Wandel resultiert in einem wachsenden Anteil alter Menschen in unserer Gesellschaft. Einige von ihnen befinden sich an der Schwelle zur Pflegebedürftigkeit und sind auf eine regelmäßige Medikamenteneinnahme angewiesen. Vor allem jene von ihnen, die an den Volkskrankheiten Demenz und Alzheimer leiden, aber auch Patienten, die aus anderen Gründen Probleme bei Medikamenteneinnahme haben, soll diese Prozedur erleichtert werden.

Durch den *MedDoser* kann die Wahrscheinlichkeit, Einnahmen von Medikamenten zu vergessen und den Therapieverlauf dadurch negativ zu beeinflussen, reduziert werden. Der Patient sowie der behandelnde Arzt können zusätzlich retrospektiv den Medikationsverlauf anhand der erfassten Daten nachvollziehen. Durch diese Form der Unterstützung des Patienten in seinem Alltag fungiert der *MedDoser* als Teil von AAL[1].

1.1 Anforderungen

- **Mobilität**: Das Gerät soll auch ohne lokal gebundene Stromzufuhr verwendbar sein
- **Einlesen des Medikationsplans**: Ein neuer beziehungsweise ein aktualisierter Medikationsplan, der auch aus mehreren Seiten bestehen kann, muss sich auf Wunsch einlesen lassen
- **Personalisierung der Erinnerungszeitpunkte**: Der Benutzer kann pro Tageszeit bestimmen, wann er an die Einnahme seiner Medikamente erinnert werden möchte
- **Darstellung der Daten**: *Alle* Medikamente, die Einnahmezeitpunkte und die Metadaten aus dem Medikationsplan werden dargestellt
- **Erinnerungsfunktion (optisch)**: Wenn ein oder mehrere Einnahmezeitpunkte überschritten wurden, wird dies durch wechselnde Farben auf dem Display sowie einer blinkenden LED gekennzeichnet. Hat der Patient die Einnahme der ausstehenden Medikamente quittiert, wird das Gerät automatisch in den Ursprungszustand versetzt
- **Erinnerungsfunktion (akustisch)**: Beim Überschreiten eines Einnahmezeitpunktes ertönt zeitgleich mit der optischen Erinnerungsfunktion ein Ton, so lange bis alle ausstehenden Einnahmen quittiert wurden

[1] Ambient Assisted Living

1.2 Rahmenbedingungen

Die folgenden Rahmenbedingungen gelten für dieses Projekt:

Rechtlich Bezüglich der Benennung des Projekts mit dem Titel *MedDoser* ergaben Recherchen im Internet, dass dieser Name keinem Produkt, einer Firma oder andersartigen Inhabern zugeordnet ist. Lediglich die internationale Gesellschaft AAMD betreibt eine Internetseite namens *MedDos*[2], welche eine phonetische Ähnlichkeit aufweist.

Die Verbreitung der Quelltexte dieses Projekts ist mit offizieller Genehmigung unseres Betreuers gestattet. Das Projekt steht nach Projektende unter der MIT-Lizenz in einem Repository auf *Github*[3] zur Verfügung und kann von Dritten somit beliebig genutzt, verändert oder verbreitet werden.

Zeitlich

- Geplante Soll-Bearbeitungszeit pro Projektteilnehmer: *150 Stunden* (Entspricht fünf ECTS-Leistungspunkten für das Projektmodul). Der jeweilige Zeitaufwand wird pro Teilnehmer regelmäßig erfasst
- Offizieller Start des Projekts: *04. Oktober 2017*
- Offizielles Ende des Projekts: *15. März 2018*
- Pausieren des Projekts während der Prüfungszeit im Januar
- Wöchentliche Treffen mit dem Betreuer

Räumlich Für die Bearbeitung des Projekts steht bis Ende Februar 2018 das eHealth-Labor im Biopark I in Regensburg zur Verfügung. Ab März 2018 finden die Arbeiten im neuen eHealth-Labor im neuen OTH-Gebäude der Fakultät Informatik/Mathematik im Raum K201 statt. Für Lötarbeiten am Gerät und seinen Komponenten muss die jeweilige Laborordnung beachtet werden. Eine Bearbeitung des Projekts außerhalb der offiziellen Räume ist jederzeit möglich.

[2] http://www.meddos.org, Aufruf am 09.03.2018
[3] http://www.github.com, Aufruf am 09.03.2018

2 Grundlagen

Dieses Kapitel stellt Informationen zu der Basis des Projekts - dem bundeseinheitlichen Medikationsplan - zur Verfügung und beschreibt die im Projekt verwendeten Hardwarekomponenten.

2.1 Bundeseinheitlicher Medikationsplan

Der bundeseinheitliche Medikationsplan (BMP) ist ein Plan, der einen Patienten bei seiner Einnahme unterstützen und die Therapiesicherheit gewährleisten soll. Seit dem 01. Oktober 2016 haben Patienten einen Anspruch auf den BMP in Papierform. Damit der BMP erstellt werden kann, müssen die folgenden beiden Anforderungen erfüllt sein [Ges17]:

- Es müssen **mindestens drei Medikamente** vom behandelnden Arzt verordnet worden sein, die gleichzeitig und systematisch einzunehmen oder anzuwenden sind
- Die Medikamente müssen **dauerhaft** eingenommen werden. Eine dauerhafte Einnahme wird mit einem Mindestzeitraum von **28 Tagen** definiert

In der Regel wird der Anspruch gegenüber dem Hausarzt oder auch dem Facharzt geltend gemacht. Ab dem Jahr 2018 kann der Patient auf Wunsch den Medikationsplan in digitaler Version auf seine elektronische Gesundheitskarte (eGK) abspeichern lassen. Die Bundesärztekammer betont, dass sie damit nicht die Papierform des BMP ersetzen will.

Vor dem BMP gab es keine standardisierten und einheitlichen Pläne. Das Problem bestand darin, dass zu viele verschiedene Medikationspläne im Umlauf waren, die verschiedenste Probleme mit sich zogen, wie beispielsweise die Interoperabilität oder Lesbarkeit. Deshalb hat der Gesetzgeber mittels des eHealth-Gesetzes den BMP fest im Sozialgesetz verankert. Der BMP wurde von der Bundesärztekammer, dem deutschen Apothekerverband und der Kassenärztliche Bundesvereinigung gemeinsam spezifiziert. Beispielsweise müssen alle Medikamente im Medikationsplan eingetragen werden. Auch die Medikamente, die von einem andern Arzt verordnet worden sind, sowie rezeptfreie Medikamente. Jedoch bleibt die Verantwortung der verschriebenen Arzneimittel unverändert bei jenem liegen, der das Arzneimittel verschrieben hat.

Der Plan wird mittels einer Software vom Arzt erstellt. Dabei kann der Ersteller unter anderem Freitexte hinzufügen oder auch die Reihenfolge der Medikationen ändern. Des

Weiteren wurden durch die Spezifikation die Softwarehersteller dazu verpflichtet, dass der BMP ab April 2017 ebenso im Praxisinformationssystem gespeichert werden muss, zum einen für die Dokumentation, zum anderen, um den BMP erweitern zu können.

Aufbau Die Struktur eines BMP (siehe Abbildung: 2.1) ist in drei Elemente unterteilt. Zunächst wird im *Header* die wesentliche Information des Arztes und seinem Patienten festgehalten. Zudem gibt es einen *zweidimensionalen Barcode*, mit dem sich der Plan einscannen und gegebenenfalls aktualisieren lässt. Der größte Fokus wird jedoch auf das *Body-Element* gelegt. Dieses enthält die Medikationen, welchen Überschriften zugeordnet werden können. Eine Medikation muss den Wirkstoff, die Dosierung, den Einnahmegrund und Einnahmehinweise beinhalten [Bun17].

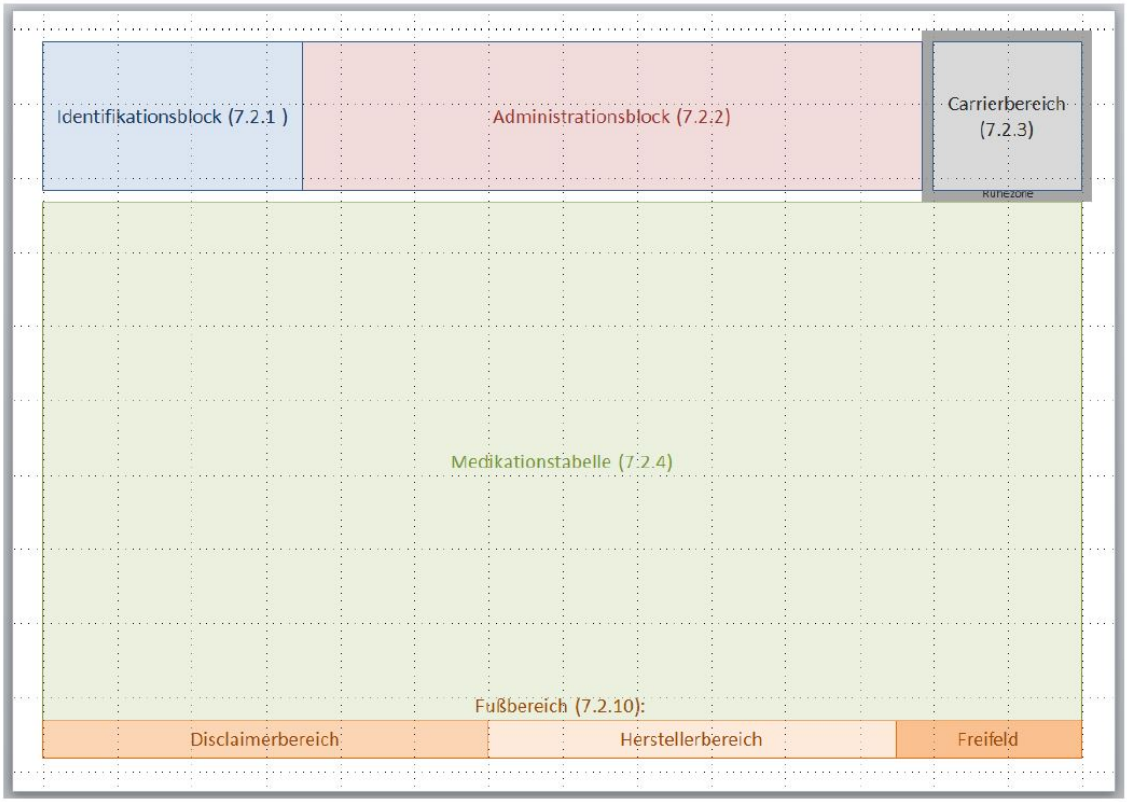

Abbildung 2.1: Die Struktur teilt sich in drei Elemente auf. Im Identifikationsblock und dem Administrationsblock befinden sich die Informationen über den Patient und Ersteller. Der Carrier-Bereich zeigt die Platzierung des Datamatrix-Codes und die Medikationstabelle listet alle Medikamente auf [Bun17].

2.2 Verwendete Hardware

Die in Tabelle 2.1 aufgelisteten Hardwarekomponenten wurden selbstständig ausgewählt und auf Zweckmäßigkeit getestet. Beim Kauf wurde auf angemessene Preise und Qualität Wert gelegt. In Abbildung 2.2 ist das fertig verkabelte Gerät zu sehen.

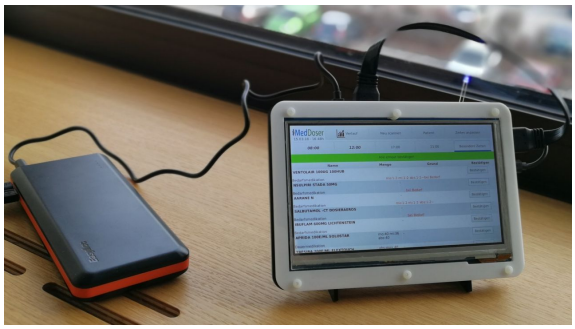

Abbildung 2.2: Der *MedDoser* inklusive aller Komponenten [Quelle: Eigene Darstellung].

Beschreibung	**Hersteller**	**Preis**
RaspberryPi 3 Model B	RaspberryPi[a]	33,99
Netzteil (microUSB-Stecker)	Goobay[b]	6,47
Externe Stromquelle (Powerbank)	EasyAcc[c]	42,99
7-Zoll-Touchscreen (kapazitiv), mit Standfuß	Waveshare[d]	66,99
MicroSD-Karte (16GB)	Intenso[e]	6,90
Plastikgehäuse (schwarz)	RaspberryPi[f]	5,80
Kamera (Flachband)	AZDelivery[g]	11,99
USB-Handscanner	Albasca[h]	145,80
LED (bunt)	BinaryKitchen[i]	0,10
Miniatur-Lautsprecher 84 dB	Conrad[j]	4,09
RTC (Real-Time-Clock)	AZDelivery[k]	6,29

Tabelle 2.1: Die im Projekt verwendete Hardware, der Händler sowie die jeweilige Bezugsquelle und ihr Preis.

[a] https://www.amazon.de/Raspberry-Pi-Model-ARM-Cortex-A53-Bluetooth/dp/B01CD5VC92, Aufruf am 04.03.2018
[b] https://www.amazon.de/Goobay-71889-Schwarz/dp/B01FOTVJJC, Aufruf am 04.03.2018
[c] https://www.amazon.de/dp/B016DA61V2, Aufruf am 04.03.2018
[d] https://www.amazon.de/gp/product/B01HPV7M4I/, Aufruf am 04.03.2018
[e] https://www.amazon.de/Intenso-Micro-Speicherkarte-SD-Adapter-schwarz/dp/B008RDCCR6, Aufruf am 04.03.2018
[f] https://www.amazon.de/dp/B00W7S1BFG, Aufruf am 04.03.2018
[g] https://www.amazon.de/gp/product/B01M6UCEM5, Aufruf am 04.03.2018
[h] http://www.albasca.com, Aufruf am 13.03.2018
[i] https://www.binary-kitchen.de, Aufruf am 15.03.2018
[j] https://www.conrad.de/de/miniatur-lautsprecher-geraeusch-entwicklung-84-db-1-w-130025-1-st-710205.html, Aufruf am 04.03.2018
[k] https://www.amazon.de/gp/product/B01M2B7HQB, Aufruf am 04.03.2018

3 Vorbereitende Maßnahmen

3.1 Einrichten der Entwicklungsumgebung

Dieses Projekt wurde mit der Entwicklungsumgebung *IntelliJ Idea Ultimate 2017 v2.5* in der Sprache Java entwickelt. Studenten der OTH Regensburg können die personalisierte Campuslizenz nutzen, welche ein Jahr ab Registrierung gültig ist. Obwohl das JavaDevelopmentKit (JDK) in der Version 9 mittlerweile auf dem Markt ist, erfolgt die Entwicklung aus Kompatibilitätsgründen mit dem *JDK 1.8.0_65*, das stabil auf dem RaspberryPi läuft beziehungsweise im Repository des Linux Package Managers angeboten wird.

3.2 Konfiguration des RaspberryPi

Für den korrekten Betrieb des Gerätes müssen im Vorfeld diverse Maßnahmen an der Konfiguration des Betriebssystems getroffen werden, die im Folgenden beschrieben werden.

3.2.1 Installation und Einrichtung des Betriebssystems

Beim Aufspielen eines Betriebssystems sind Tools wie *Etcher*[4] oder *PiBakery*[5] hilfreich. Zum Startzeitpunkt des Projekts war das aktuellste Betriebssystem das RaspbianOS in der Version 8 (*Jessie*)[6]. In jedem Fall sollte die microSD-Karte im Vorfeld formatiert werden, um danach die entpackte Image-Datei daraufzuschreiben.
Für das Projekt *MedDoser* ist relevant, dass sich das Gerät für die Dauer der Entwicklung mit einem WLAN-Hotspot verbindet, um einen Zugang über das Netzwerk zu gewährleisten. Das Standardkennwort des Benutzers *pi* soll zu Beginn standardmäßig auf *raspberry* belassen werden. Nach der erfolgreichen Installation des Betriebssystems muss die Datei */boot/config.txt* angepasst werden. Folgende Zeilen sorgen dafür, dass der Touchscreen erkannt und mit der richtigen Auflösung bespielt wird:

[4] http://www.etcher.io, Aufruf am 14.03.2018
[5] http://www.pibakery.org, Aufruf am 10.12.2018
[6] https://www.raspberrypi.org/blog/raspbian-jessie-is-here, Aufruf am 14.01.2018

```
_max_usb_current=1
hdmi_group=2
hdmi_mode=87
hdmi_cvt 1024 600 60 6 0 0 0
hdmi_drive=1_
```

3.2.2 Aktualisieren des Systems

Nach dem ersten Start des Gerätes sollten die Softwarekomponenten auf den aktuellen Stand gebracht werden. Mit folgenden Befehlen wird die Aktualisierung angestoßen:

```
sudo apt-get update
sudo apt-get upgrade
```

3.2.3 Einrichten des SSH-Zugangs

Für die Steuerung des *RaspberryPi* ist keine grafische Oberfläche notwendig. Mit einem SSH-Client wie beispielsweise PuTTY[7] und bekannter IP-Adresse des verbundenen Geräts lässt sich eine Verbindung im Shell-Modus herstellen. Alternativ kann der Desktop-Modus des RaspberryPi auch mit dem Tool *VNC-Viewer*[8] auf den Entwicklungsrechner gestreamt werden. Hierzu ist dessen Installation auf dem Quellgerät sowie auf dem Client-Rechner notwendig.

3.2.4 Installation des JDK

Das Raspberry benötigt zum Ausführen der Java-Anwendung mindestens das JDK in der Version *1.8.0_65*. Ist das noch nicht der Fall, lässt es sich mit den Befehlen

```
sudo apt-get update
sudo apt-get install oracle-java8-jdk
```

auf diese Version installieren beziehungsweise aktualisieren.

3.2.5 Einrichten der Samba-Freigabe

Um einen Ordner auf dem *RaspberryPi* vom Windows Explorer aus zugänglich zu machen, muss ein Samba-Server installiert werden:

[7] https://www.putty.org, Aufruf am 03.03.2018
[8] https://www.realvnc.com/en, Aufruf am 15.12.2017

```
sudo apt-get update
sudo apt-get install samba samba-common smbclient
```

Danach ist der Status des Servers mit folgenden Anweisungen zu überprüfen:

```
sudo service smbd status
sudo service nmbd status
```

Die Samba-Konfiguration ist in der zentralen Konfigurationsdatei */etc/samba/smb.conf* gespeichert. In der Standardeinstellung ist sie äußerst umfangreich und unübersichtlich, weshalb eine Sicherung der Datei und das Anlegen einer neuen Version zu empfehlen ist:

```
sudo mv /etc/samba/smb.conf /etc/samba/smb.conf_alt
sudo nano /etc/samba/smb.conf
```

Der Inhalt der neu erstellten Datei kann jederzeit wieder erweitert werden. Folgenden Inhalt sollte sie dennoch aufweisen:

```
[global]
workgroup = WORKGROUP
security = user
encrypt passwords = yes
```

Mit dem Befehl *textparm* wird die Gültigkeit der neuen Datei überprüft.
Nach jeder durchgeführten Änderung an der Konfigurationsdatei müssen die Samba-Dienste neu gestartet werden, erst dann wird die Konfiguration übernommen:

```
sudo service smbd restart
sudo service nmbd restart
```

Einrichten von Samba-Freigaben für Verzeichnisse Zuerst wird ein neues Grundverzeichnis angelegt, das im Anschluss alle freizugebenden Ordner enthalten soll. Danach werden die gewünschten Verzeichnisse samt Berechtigung darin erstellt:

```
sudo mkdir /home/shares
sudo mkdir /home/shares/MedDoser
sudo chown root:root /home/shares/MedDoser/
sudo chmod 777 /home/shares/MedDoser/
```

Nun müssen die neuen Freigaben der Konfigurationsdatei bekannt gemacht werden. Hierzu wird die Samba-Konfigurationsdatei geöffnet

```
sudo nano /etc/samba/smb.conf
```

und die Freigaben eingetragen, für die zuvor Verzeichnisse angelegt wurden:

```
[Samba_MedDoser]
comment = Samba-MedDoser-Freigabe
path = /home/shares/MedDoser
read only = no
```

Nach der Änderung der Konfigurationsdatei empfiehlt sich ein erneuter Test mit dem Befehl *testparm* sowie ein Neustart der Samba-Dienste.

Einrichten eines Samba-Passworts Für die Authentifizierung ist die Benutzerverwaltung auf dem Server vorgesehen. Allerdings benutzt Samba nur die eingerichteten Benutzernamen und die Datei- und Verzeichnisberechtigungen, nicht die Passwörter. Für jeden Benutzer, der Samba-Freigaben benutzen will, muss ein separates Passwort für Samba eingerichtet werden. Da das *RaspberryPi* bisher lediglich den Standardbenutzer *pi* kennt, wird er auf das Zugreifen auf die Samba-Freigabe berechtigt:

```
sudo smbpasswd -a pi
```

Ausgehend vom Windows-Rechner lässt sich nun ein neues Netzlaufwerk mit dem Befehl

```
\\IP-ADRESSE\Samba_MedDoser
```

einbinden, das auf den freigegebenen Ordner auf dem *RaspberryPi* verweist. Bei obigem Befehl ist darauf zu achten, dass genau der Alias angegeben werden muss, der in der Konfigurationsdatei angegeben wurde.

3.2.6 Modifizieren des Bootvorgangs

Der Startvorgang des *RaspberryPi* generiert üblicherweise viele Konsolenausgaben, die dem Endanwender beispielsweise durch ein Logo, das diese überdeckt, größtenteils vorenthalten werden können.

Aktivieren eines eigenen Splash-Screens Zuerst muss in der Grundkonfiguration des *RaspberryPi* der Splash-Screen aktiviert werden:

```
sudo raspi-config
```

Unter *3 - Boot options* und *B3 - Splash Screen* muss die Option *Yes* gewählt werden.

Umschalten von Desktop- zu Konsolenmodus Damit das *RaspberryPi* auf die Konsole anstatt auf den Desktop bootet, ist in der Raspberry-Konfiguration unter *Boot from CLI* statt *from Desktop* einzustellen.

Personalisieren des Boot-Logos Um das Standard-Boot-Logo mit dem *MedDoser*-Logo zu überschreiben, muss es im *PNG*-Format bereitgestellt werden. Es muss den Namen *splash.png* tragen und im Ordner */home/shares/MedDoser* liegen. Danach muss folgender Befehl ausgeführt werden, um das alte Logo zu überschreiben:

```
sudo cp /home/shares/MedDoser/splash.png /usr/share/plymouth/
    themes/pix/splash.png
```

Reduzieren der Konsolenausgaben Um die Anzahl der Konsolenausgaben zu reduzieren, muss zuerst folgende Datei editiert werden:

```
sudo nano /boot/cmdline.txt
```

Die folgende Datei enthält eine einzige Zeile mit allen Boot-Optionen. Folgende Ersetzungen sind nun vorzunehmen (Hierbei ist darauf zu achten, dass die Datei nur eine Leerzeile am Ende besitzt):

1. *console=tty1* mit *console=tty3* ersetzen
2. *loglevel=3* zur Zeile hinzufügen, um nicht-kritische Log-Meldungen auszublenden
3. *logo.nologo* zur Zeile hinzufügen, um die vier angezeigten Himbeeren beim Bootvorgang auszublenden

Mit *Strg + X* speichern, mit *Y* bestätigen und mittels *sudo reboot* die Änderungen überprüfen.

Aktivieren der JavaFX-Unterstützung Hierzu ist die manuelle Installation eines Patches[9] notwendig. Damit die JavaFX-Anwendung auf dem Touchscreen maximiert und vollständig dargestellt wird, muss sichergestellt sein, dass in der Datei */boot/config.txt* folgende Zeilen vorhanden sind:

```
overscan_left=20
overscan_right=12
overscan_top=10
overscan_bottom=10

disable_overscan=1
```

Automatisches Starten der Anwendung Um eine Anwendung im Shell-Modus zu starten, ist deren Autostart in der Datei */etc/rc.local* einzustellen:

```
sudo nano /etc/rc.local
```

Der Aufruf der Java-Anwendung geschieht mit folgendem Befehl:

```
# Start application MEDDOSER.jar
java -jar /home/shares/MedDoser/medDoser.jar
exit 0
```

Am Ende der Datei muss zwingend die Anweisung *exit 0* stehen. Da dieses Bootskript standardmäßig mit root-Rechten ausgeführt wird, ist eine Angabe des Präfix *sudo* überflüssig.

3.2.7 Montieren und Konfigurieren der RTC (Real-Time-Clock)

Damit das *RaspberryPi* auch nach längerer Zeit im ausgeschalteten Zustand die korrekte Systemzeit anzeigt, muss es beim Booten entweder eine Internetverbindung haben und über das NTP-Protokoll die Zeit synchronisieren, oder aber ein RTC-Modul (Real-Time-Clock) (Siehe 2.2) handhabt diese Aufgabe. Das verwendete Modul (DS3231) besitzt sechs Pins, von denen vier mit Female-Female-Steckkabeln an die GPIO-Pins des *RaspberryPi* angeschlossen werden müssen. Die Belegungen von GND und VCC sind in folgendem Beispiel variabel:

[9] http://gluonhq.com/products/mobile/javafxports/get, Aufruf am 14.03.2018

1. **GND** - Pin 14 (Ground)
2. **VCC** - Pin 1 (+3.3V)
3. **SDA** - Pin 3 (GPIO 2)
4. **SCL** - Pin 5 (GPIO 3)

Setzen der Zeitzone und Ändern der Standorteinstellung Das Umschalten der Zeitzone auf Europa erfolgt in der *RaspberryPi*-Konfiguration unter *Localisation Options*. Dort ist unter *I2 Change Timezone Europe* und dann *Berlin* zu wählen.

Die Standorteinstellung des Geräts lässt sich ebenfalls in der *RaspberryPi*-Konfiguration unter *Change Locale* anpassen. In der Liste muss der Eintrag *de_DE.UTF-8 UTF-8* mit der Leertaste selektiert und alle Einträge, die mit *en_GB* beginnen, abgewählt werden.

Aktivieren des I2C-Bus In den Konfigurationseinstellungen (Befehl: *raspi-config*) unter *5: Interfacing Options* lässt sich der Bus unter *P5 I2C* aktivieren. Nun muss (falls noch nicht vorhanden) das Package *i2c-tools* installiert werden. Zuvor wird empfohlen, alle Packages auf den aktuellen Stand zu bringen:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python-smbus i2c-tools
```

Mit folgendem Befehl lässt sich testen, ob das RaspberryPi das RTC-Modul erkannt und aktiviert hat:

```
sudo i2cdetect -y 1
```

Dies zeigt sich dadurch, dass die ID #68 (Die Adresse des RTC-Moduls) aktiviert ist. Sobald der entsprechende Kerneltreiber aktiv ist, wandelt sich die Zahl in ein *UU*. Den Kerneltreiber aktiviert man beim Systemstart, indem in der Datei */boot/config.txt* die Zeile

```
dtoverlay=i2c-rtc,ds3231
```

einfügt.

Entfernen der Fake-Hardware-Uhr Nachdem der Kerneltreiber für das RTC-Modul aktiviert wurde und eine Verbindung zum RaspberryPi aufgebaut hat, muss das *fake hwclock*-Package vom Gerät entfernt werden. Dieses Package agiert als Platzhalter für ein RTC-Modul, falls keines vorhanden ist. die folgenden Befehle entfernen das Package und dessen Referenz von allen Startup-Skripten [Gus17]

```
sudo apt-get -y remove fake-hwclock
sudo update-rc.d -f fake-hwclock remove
```

Zuletzt müssen folgende drei Zeilen aus der Datei */lib/udev/hwclock-set* entfernt beziehungsweise auskommentiert werden:

```
#if [ -e /run/systemd/system ] ; then
#    exit 0
#fi
```

Synchronisieren der Zeit bei Systemstart Nach dem erfolgreichen Einbinden des RTC-Moduls in das System muss die Zeit einmalig manuell richtig gesetzt werden. Dazu wird das Gerät mit dem Internet verbunden, um über das NetworkTime-Protokoll die korrekte Uhrzeit und das Datum zu beziehen. Mit dem Befehl

```
sudo hwclock -w
```

wird die lokale Zeit auf das RTC-Modul geschrieben und verbleibt dort so lange, bis die Knopfbatterie entfernt wird. Nach diversen Tests baute das RaspberryPi keine dauerhafte Verbindung zum RTC-Modul auf, denn während der Entwicklungsphase scheiterte der Befehl

```
sudo hwclock -D -r
```

zum Abfragen der Zeit aus dem RTC-Modul sporadisch. Damit das automatische Starten der Software (Definiert in der Datei */etc/rc.local*) auch bei einer negativen Antwort auf den Synchronisationsbefehl noch funktioniert und nicht abbricht, hilft folgendes Programmierkonstrukt:

Durch das Einbauen einer While-Schleife vor dem Aufruf der eigentlichen Java-Anwendung wartet das Boot-Skript mit der weiteren Ausführung so lange, bis die lokale Systemzeit korrekt vom RTC-Modul gelesen wurde. Der Befehl

```
hwclock -s
```

sorgt dafür, dass die Zeit des RTC-Moduls mit der lokalen Systemzeit überschrieben[10] und somit synchronisiert wird:

```
# Run this loop on every startup
while ! hwclock -s
do
   echo "Failed to sync RTC to local time."
done

echo "Successfully synced RTC to local time"
```

Abbildung 3.1: Einbau einer While-Schleife in das Bootskript. Wichtig ist das Aufrufen der Schleife, bevor die Java-Anwendung gestartet wird (Quelle: Eigenentwicklung).

3.2.8 Montieren und Konfigurieren des Lautsprechers

Die Tonausgabe zur Erinnerung an ausstehende Medikationen besitzt minimale Anforderungen an die Akustik, weshalb eine simple Membran (Siehe Abschnitt 2.2) ausreichend ist. Das Bauteil muss zuvor mit zwei kurzen Litzen beziehungsweise zwei Male-Female-Steckkabeln mit den GPIO-Pins des *RaspberryPi* verbunden werden. Hierzu wird jeweils ein Ende eines Kabels mit den bipoligen Vorrichtungen des Lautsprechers verlötet. Der Lautsprecher findet anschließend aufgrund seiner geringen Größe im Gehäuse des *RaspberryPi* Platz.

Wahl und Ausgabe des Tons Um den Patient an die Einnahme zu erinnern, genügt ein kurzer Ton, der jedoch regelmäßig bis zur Erledigung aller Aufgaben ertönen soll. Komplexe und differenzenreiche Tonstücke übersteigen die Kapazität des Bauteils, weshalb das Abspielen einer Melodie, bestehend aus einer festgelegten Reihenfolge von Frequenzen, umgesetzt wurde:

```
int track [8] = {0, 0, 340, 0, 640, 0, 340, 0};
```

Die Töne werden jeweils mit einer Verzögerung von 110 Millisekunden abgespielt. Das Abspielen der Frequenzen auf dem *GPIO Pin 3* des *RaspberryPi* steuert eine Funktion, die in C programmiert ist und wird von der Java-Anwendung mit dem Befehl

[10] https://linux.die.net/man/8/hwclock, Aufruf am 22.04.2018

```
Runtime.getRuntime().exec(./tone);
```

aufgerufen, wobei *tone* für den Dateinamen des C-Kompilats steht. Das Ausführungsverzeichnis ist standardmäßig der Speicherort der *.jar*-Datei, von der aus die Java-Anwendung ausgeführt wird. Der verwendete Quelltext stammt aus dem *WiringPi*-Projekt [Gor].

Der Ton wird alle sieben Sekunden, getriggert durch einen Thread, abgespielt, so lange bis keine Einnahmen von Medikamenten mehr ausstehen.

3.2.9 Montieren und Konfigurieren der LED

Das kleinste und preisgünstigste Bauteil ist die Leuchtdiode (Siehe Abschnitt 2.2). Sie blinkt verschiedenfarbig, wobei die Farbübergänge fließend sind. Sie wird mit zwei Female-Female-Steckkabeln mit den GPIO-Pins (Pin 12 (GPIO 18) und Pin 14 (Ground)) des *RaspberryPi* verbunden. Es ist kein separater Vorwiderstand notwendig, weil er bereits im Bauteil integriert ist. Ihre Ansteuerung erfolgt im selben Zeitraum und Häufigkeit wie jene des Lautsprechers.

4 Umsetzung

In diesem Kapitel ist die technische Umsetzung und die Vorgehensweise bei der Implementierung beschrieben. Die Basis für das Funktionieren der Java-Anwendung bildet die Datenspeicherung. Danach wird erläutert, wie Medikationspläne eingescannt und mit dem Parser verarbeitet werden. Abschließend sind der Aufbau und die Elemente der Benutzeroberfläche Thema dieses Abschnitts.

4.1 Datenspeicherung

Der Handelsname, der Arzneimittelcode (PZN), die Wirkstoffbezeichnung sowie die Wirkstoffstärke sind der offiziellen Spezifikation der KBV der Arzneimitteldatenbank der IFA zu entnehmen [Bun17]. Zum Zeitpunkt der Durchführung dieses Projekts stand jedoch keine Lizenz und somit auch kein physisches Schema dieser Datenbank für eine provisorische, aber strukturell äquivalente Datenbank zur Verfügung. Aus diesem Grund beschreibt Abschnitt 4.1.3 den Aufbau einer SQLite-Datenbank als Methode zur Datenhaltung. Die personalisierten Einnahmezeiten, die der Benutzer in der Oberfläche der Anwendung festlegen kann, sowie die täglichen geplanten Einnahmen von Medikamenten werden hingegen in einer separaten *XML-Datei* abgelegt.

4.1.1 XML-Datei

Die Wahl für diese Form der Speicherung begründet sich damit, dass Hierarchien und Strukturen mit XML-Schemata besser abgebildet werden können als über eine relationale Datenbankstruktur. In Abbildung 4.1 ist die Struktur und der Inhalt der Datei dargestellt.

```
1
2  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
3  <MedDoser>
4    <CreationDate>03/12/2018 00:11</CreationDate>
5    <LastOpenedDate>03/13/2018 00:53</LastOpenedDate>
6
7    <IngestionTime>
8      <morning time="08:00"/>
9      <midday time="12:00"/>
10     <evening time="17:00"/>
11     <night time="21:00"/>
12   </IngestionTime>
13
14   <MedicationIngestion>
15     <ingestion medStatementReference="urn:uuid:09e8d5b1-0e6f-4f0d-8871-5d7eeecbe532" scheduledDate=
           "03/12/2018" scheduledTime="" scheduledTimeCode="PCV" status="2" time=""/>
16     </MedicationIngestion>
17 </MedDoser>
```

Abbildung 4.1: Exemplarische Struktur der XML-Datei mit einer vorhandenen Medikation. Sie wurde noch nicht bestätigt, was der Status *2* kennzeichnet. Nach Quittierung durch den Benutzer wird der Status entweder auf *0* (Nicht eingenommen) oder *1* (Eingenommen) wechseln. Das Attribut *scheduledDate* kennzeichnet den geplanten Tag der Einnahme. Im Kopf der Datei befinden sich zudem die vom Benutzer festgelegten Einnahmezeiten.

4.1.2 Performanceoptimierung

Bei *jeder* Benutzerinteraktion müssen Informationen aus den genannten Datenstrukturen bezogen werden. Da bei jedem Zugriff auf die XML-Datei deren Inhalte zeitintensiv interpretiert werden müssen, werden nach einer Änderung der Daten die Informationen im RAM aktuallisiert. Zugriffe auf die XML-Datei erfolgen nur noch bei Schreibvorgängen. Dies betrifft alle Informationen zu den Medikamenten sowie zu jenen, die geplant sind oder bereits vom Benutzer bestätigt oder als nicht eingenommen deklariert wurden.

4.1.3 SQLite-Datenbank

Aufgrund des bereits genannten Fehlens der externen Arzneimitteldatenbank der IFA wurden stattdessen exemplarisch 70 Medikamente in einer SQLite-Datenbank angelegt, welche die Java-Anwendung über eine Schnittstelle mit Daten versorgt. Die geringe Komplexität und der niedrige Speicherbedarf aufgrund des nicht existierenden Overheads im Gegensatz zu einer vollständigen SQL-Datenbank mit DBMS[11] im Hintergrund spricht für die Wahl einer SQLite-Datei als Datenspeicher. In Abbildung 4.2 ist das Schema definiert, welches die Tabellen und ihre Felder beschreibt. Die Tabellen stehen untereinander in keiner Beziehung.

Abbildung 4.2: Beschreibung des Schemas der SQLite-Datenbank mit zwei Tabellen und ihren Spaltennamen [Quelle: Eigene Darstellung].

4.2 Einscannen des Medikationsplans

Das Einlesen des sogenannten *DataMatrix*-Codes auf dem Papier-Medikationsplan ist die Voraussetzung für den weiteren Ablauf des Programmflusses.

DataMatrix-Code Ein DataMatrix-Code ist ein zweidimensionaler Strichcode, der eine quadratische oder rechteckige Form besitzt. Er besteht aus mindestens *10 x 10* und maximal *144 x 144 Modulen*, die eine maximale Datenkapazität von 2335 alphanumerischen Zeichen aufweisen. Wenn die Anzahl an Modulen pro Seite die Zahl 24 übersteigt, so wird der Code in sog. *Blöcke* aufgeteilt. Der Rand um den gesamten Code herum sollte breiter als ein Modul sein, um eine richtige Dekodierung zu gewährleisten. Zum Zwecke der Wiederherstellung von korrupten Codes wird der *Reed-Solomon-Code* angewandt [Key]. Dieser Code der Fehlererkennung- und Behandlung ist im internationalen Standard *ISO/IEC*

[11] Datenbankmanagementsystem

16022 definiert, welcher auch Vorgabe der KBV ist. Ein DataMatrix-Code bietet aufgrund seines Aufbaus eine höhere Kapazität als andere Arten von Strichcodes. Nichtsdestotrotz ist der Platz begrenzt, weshalb der Medikationsplan im **Ultrakurzformat** (UKF) kodiert wird.

Scannen Zum Erkennen des Codes auf einem auf weißem DIN-A4-Papier ausgedruckten Medikationsplans stehen aus technischer Sicht drei Komponenten (Siehe Abschnitt 2.2 zur Verfügung:

- **Standard-Kamera**: Sie wird über den eigens dafür vorgesehenen Port auf der Platine des RaspberryPi verbunden. Der Autofokus dieser Kamera liefert jedoch lediglich bei einer Entfernung von circa *0.5 m* des Gerätes zum Papier scharfe und brauchbare Bilder. Im realen Anwendungsfall beträgt dieser Abstand jedoch normalerweise nur einige Zentimeter

- **USB-Kamera**: Sie wird über einen der vier USB-Ports des RaspberryPi angebunden. Hier ist laut mehreren Tests der Autofokus besser, trotzdem muss das Gerät sehr ruhig gehalten werden. Dies gewährleistet jedoch noch keine stetig scharfen Bilder

- **USB-Handscanner**: Der externe Handscanner wird ebenfalls über einen USB-Port verbunden und beherrscht Plug-and-Play, d.h. es sind keine zusätzlichen Installationen notwendig. Sobald während der Bedienung des Scanners ein valider Code erkannt wird, wird die eingelesene Zeichenkette an das fokussierte Eingabefeld gestreamt

Bezüglich der ersten beiden Varianten muss die Oberfläche der Software zum einen eine *Live-Vorschau des Kamerabildes* als Orientierung für den Benutzer aufweisen, zum anderen muss sie das fehlerfreie und zügige *Dekodieren* des DataMatrix-Codes gewährleisten. Für das Lösen der ersten Aufgabe bietet sich die quelloffene Bibliothek *openCV*[12] an, die zu diesem Zweck entsprechend implementiert werden müsste. Das Dekodieren des Codes wurde mit folgenden Bibliotheken getestet:

- **ZXing**[13]: Funktioniert resultierend aus eigenen Tests nur für optimale Bilder ab einer Auflösung von *250x250px*

- **libdmtx**[14]: Funktioniert für Bilder mit beliebiger Dimension, jedoch nicht mit Bildern, die mit der Kamera aufgenommen wurden

- **OnBarcode**[15]: Funktioniert ebenso wenig wie ZXing und ist zudem kommerziell.

[12] http://opencv-java-tutorials.readthedocs.io/en/latest, Aufruf am 13.12.2017
[13] http://zxing.org, Aufruf am 13.12.2017
[14] http://libdmtx.sourceforge.net, Aufruf am 13.12.2017
[15] http://www.onbarcode.com/products/java_barcode_reader/barcodes/data_matrix.html, Aufruf am 13.12.2017

Generell muss sichergestellt sein, dass die Helligkeit beziehungsweise die Ausleuchtung der Umgebung sowie der Winkel zum Papier stimmt, um brauchbare Bilder zu erhalten. Sind diese Faktoren nicht erfüllt, führt das zu kontrastarmen und unscharfen Bildern, auf denen der DataMatrix-Code nicht erkannt wird.

Als praktikabelste und zuverlässigste Lösung hat sich aufgrund der begrenzten Bearbeitungsdauer des Projekts der oben beschriebene *Handscanner* durchgesetzt. Er muss lediglich zum Einlesen eines Medikationsplan mit dem Gerät verbunden werden. Zum alltäglichen Betrieb des *MedDoser* ist er nicht notwendig.

Konfiguration des Handscanners Der Handscanner des Herstellers *Albasca*[16] lässt sich mittels vorgefertigter Barcodes, die einmalig eingescannt werden müssen, konfigurieren. Diese Codes sowie weitere Bedienungshinweise befinden sich in einer PDF-Datei[17] auf der beigelegten Mini-CD.

Für die Entwicklung unter Windows ist das Tastaturlayout bereits auf Europa voreingestellt und muss nicht verändert werden. Für den Betrieb mit dem RaspberryPi muss diese Einstellung jedoch auf das US-amerikanische Layout geändert werden. Die benötigten 2D-Barcodes für das Umschalten zwischen den Konfigurationen finden sich in Tabelle 4.1 und müssen lediglich einmalig eingescannt werden, um wirksam zu werden.

Deutschland	United States
	
17 – Germany	**1 - U.S.**

Tabelle 4.1: Mit dem Handscanner einlesbare Konfigurationsbarcodes für europäische und US-amerikanische Tastaturlayouts

Zusätzlich muss der Handscanner auf das Einlesen der speziellen Form der Barcodes vorbereitet werden. Die Konfiguration hierfür befindet sich in dem in Abbildung 4.3 gezeigten Barcode:

Ablauf des Scanvorgangs Falls der Benutzer den *MedDoser* erstmalig benutzt, wird er aufgefordert, seinen Medikationsplan einzuscannen. Da sich auf einer Seite eines Medikationsplans maximal 15 Medikamente befinden dürfen, können sie aus mehreren Seiten bestehen. In Abbildung 4.4 ist der Ablauf des Einscannens schematisch dargestellt.

[16] http://www.albasca.com, Aufruf am 11.03.2018
[17] MK-6200 User Guide V 1 4.pdf

Abbildung 4.3: Konfigurationsbarcode für das Aktivieren von DataMatrix-Codes

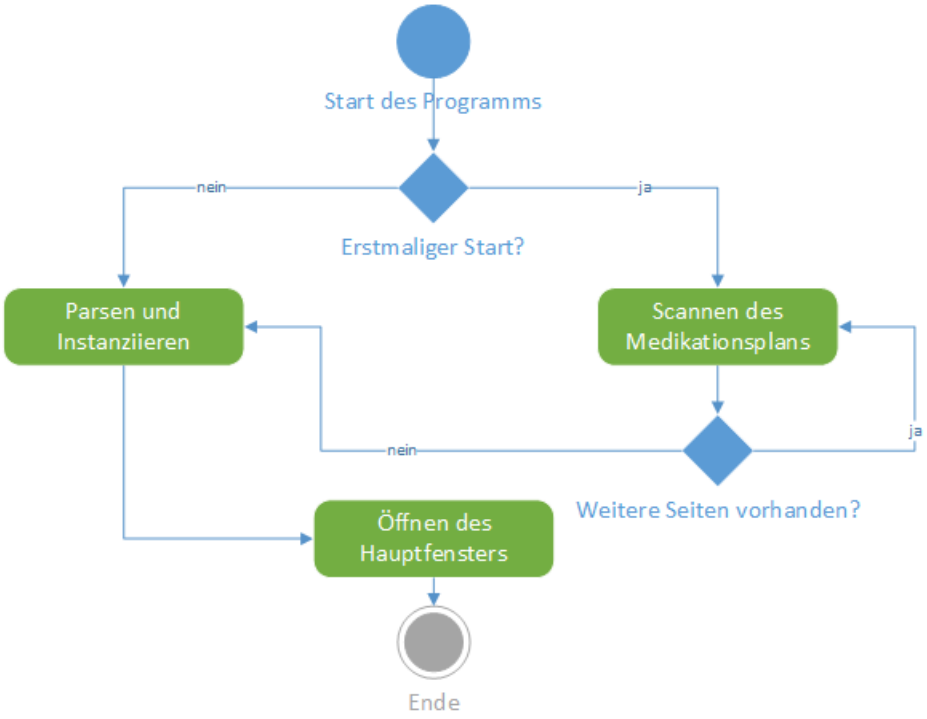

Abbildung 4.4: Beim Scannen des Medikationsplans wird überprüft, ob dies das erste Mal geschieht. Ist das der Fall, öffnet sich das Hauptfenster erst nach dem erfolgreichen Scannen aller Seiten des Medikationsplans. Andernfalls wird der Scanvorgang übersprungen und die vorhandenen Daten werden in die Benutzeroberfläche geladen [Quelle: Eigene Darstellung].

Der Scanvorgang für die aktuelle Seite endet, wenn folgende Zeichenfolge in der Zeichenkette erkannt wird:

```
char[] delimiter = new char []{'<','/', 'M', 'P', '>'};
```

Alle durch den Handscanner eingegebenen Zeichen inklusive dieser fünf erkannten Zeichen werden an den UKF-Parser (Siehe Abschnitt 4.3) weitergereicht, der den Erfolg des Parsens zurückgibt. War der Inhalt der Zeichenkette fehlerhaft, beispielsweise durch einen falschen Winkel des Scanners zum Barcode oder aufgrund falscher Kodierung durch den Aussteller des Plans, wird dies dem Benutzer mitgeteilt. Eine andere Fehlerursache kann die falsche

Konfiguration (Siehe Abschnitt 4.2) sein. In diesem Fall wird der Eingabestrom auf die Existenz von folgender Zeichenfolge hin geprüft:

```
char[] wrongDelimiter = new char[]{'&', 'S', '&', 'M', 'P'};
```

Die gezeigte falsche Zeichenfolge tritt auf, weil die Spracheinstellung des Handscanners die gelesenen Zeichen fehlerhaft kodiert, weil der Handscanner einem mit dem Gerät inkompatiblen Zeichensatz verwendet. Bei einem erfolgreichen Scanvorgang werden die bisherigen gespeicherten Daten auf dem Gerät (nach Bestätigung) gelöscht und durch die neuen Daten ersetzt.

4.3 Parsen

Im folgenden Kapitel wird das Ultrakurzformat und der Aufbau eines HL7-FHIR-Dokuments erklärt. Danach wird auf die Implementierung eingegangen, speziell auf die Behandlung von Sonderfällen.

4.3.1 Ultrakurzformat zu HL7-FHIR

Der Schritt der Konvertierung des Medikationsplans vom Ultrakurzformat hin zum HL7-FHIR-Format wird mit einem Parser durchgeführt. Der Parser wurde bereits in einem früheren Projekt entwickelt. Um Objektorientierung zu implementieren und ihn stabiler zu entwerfen, wurde dieser jedoch im Rahmen dieses Projekts neu entwickelt. Als Eingabe wird eine Zeichenkette erwartet, deren Inhalt dem UKF-Format entspricht. Das Ergebnis der Konvertierung ist eine XML-Datei, deren Inhalte dem FHIR-Standard[18] entsprechen.

Aufbau eines UKF-Strings Nachdem der UKF-String vom Handscanner eingelesen wurde, muss er zunächst interpretiert werden. Die Zeichenkette kann verschiedene Pflicht- und weitere Elemente enthalten. In der Abbildung 4.5 wird gezeigt, welche Elemente ein UKF-String enthalten kann. Er beinhaltet anfangs genau einmal die *Metainformationen*, die mit dem Element **MP** beschrieben sind. Die folgenden Bestandteile *Patient* mit dem Element **P** und *Ersteller* mit dem Element **A** kommen gleichermaßen einmalig vor. Die *Organisation* mit dem Element **C** und die *Observations* mit dem Element **O** können maximal einmal vorkommen. Danach werden die Medikationsabschnitte als *Sections*, die mit dem Element **S** beschrieben sind, aufgelistet. Dabei können 0 bis 23 Abschnitte vorkommen. Jeder Medikationsabschnitt, der mit dem Element **M** gekennzeichnet ist, hat mindestens ein Medikament und kann bis zu 45 Medikamente beinhalten, wobei jedem Medikament 0 bis 3 Wirkstoffe mit dem Element **W** zugeordnet werden können. Jeder Medikationsabschnitt kann optional eine gebundene *Freitextzeile* mit dem Element **X** beinhalten, die einen Bezug zum Medikationsabschnitt hat. Zuletzt besteht die Möglichkeit, am Ende aller Medikationsabschnitte eine weitere Section zu erstellen, die genau ein Freitext-Element

[18] https://www.hl7.org/fhir, Aufruf am 13.12.2017

enthält. Dieses dient dazu, einen Freitext für die gesamten Medikationen beziehungsweise den ganzen Medikationsplan zu hinterlegen.

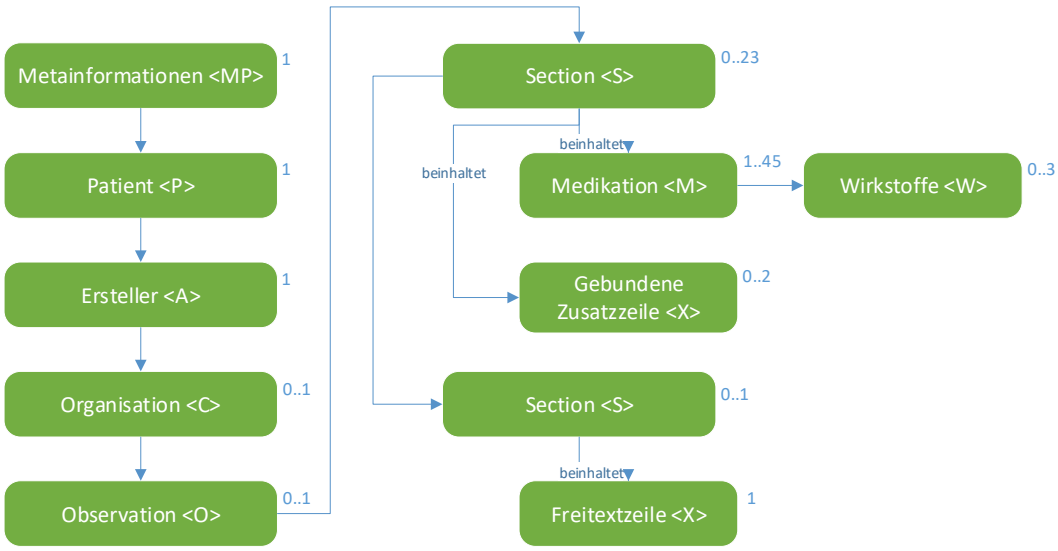

Abbildung 4.5: Der UKF-String beinhaltet Informationen zu den Metadaten, zum Patient und zum Ersteller, die genau einmal vorkommen müssen. Die Organisation und die Observation können einmal vorkommen. Der Medikationsabschnitt, der die Medikationen, Wirkstoffe und eine gebundene Freitextzeile beinhaltet, kann mehrmals vorkommen. Zum Schluss kann ein Freitext für alle Medikationen hinterlegt werden [Quelle: Eigene Darstellung].

Attribute der Elemente Ein UKF-Elemente kann Attribute aufweisen, die jedoch optional vorhanden sein können. Der Implementierungsleitfaden der Organisation *IHE Germany*[19] beinhaltet eine Übersicht über alle Elemente und ihre Attribute.

[19] wiki.hl7.de/index.php?title=IG:Ultrakurzformat_Patientenbezogener_Medikationsplan, Aufruf am 15.03.2018

4.3.2 Praktisches Beispiel eines UKF-Strings

Im Quellcode-Beispiel in Abbildung 4.6 wird ein UKF-String gezeigt, der den beschriebenen Aufbau widerspiegelt.

Jeder UKF-String enthält ein **MP**-Element, das die Metainformationen eines Medikationsplans beschreibt. Es ist zu erkennen, dass die Pflichtfelder (Datenfeld: MP.U und MP.l) ausgefüllt worden sind und aktuell die erste (Datenfeld: MP.a) von zwei (Datenfeld: MP.z) Seiten eingelesen worden ist.

Als Sprachkennung (Datenfeld: MP.l) wurde die deutsche Sprache gekennzeichnet. Danach folgt die Patientin (Datenfeld: MP.P.s) Erika-Annemarie Demon-Lier (Datenfeld: MP.P.g und MP.P.f), die eine eGK-Nummer (Datenfeld: MP.P.egk) und ein Geburtsdatum (Datenfeld: MP.P.b) aufweisen kann.

Im Anschluss daran werden die Informationen über den Ersteller, der in der Regel der behandelnde Arzt ist, im **A**-Element aufgeführt. Der Name (Datenfeld: MP.A.n) muss enthalten sein. Im aufgeführten UKF-String sind zudem die Straße (Datenfeld: MP.A.s), der Ort (Datenfeld: MP.A.c), die Telefonnummer (Datenfeld: MP.A.p), die E-Mail Adresse (Datenfeld: MP.A.e) und das Ausstelldatum (Datenfeld: MP.A.t), an dem der BMP zuletzt aktualisiert worden ist, charakterisiert.

In der Observation werden im **O**-Element die klinischen Parameter, die Allergien beziehungsweise Intoleranzen sowie die Gesundheitsbelange beschrieben. Danach folgen die Medikamente.

Die Medikamente werden mit dem Element **E** beschrieben, welche zunächst in die Medikationsabschnitte (**S**-Element) eingeteilt sind. Die Medikamente können eine Pharmazentralnummer (Datenfeld: MP.S.M.p) aufweisen, womit der Handelsname ableitbar ist. Des Weiteren werden die Dosierungen (Datenfeld: MP.S.M.m, MP.S.M.d, MP.S.M.d und MP.S.M.u) hinterlegt, die zur jeweiligen Tageszeit einzunehmen sind.

Zuletzt weist das Beispiel einen Freitext (MP.S.X) auf, der unter der Medikationstabelle eines BMP platziert wurde. Wenn der Freitext ohne Bezug zu einem Medikationseintrag vorhanden ist, wird dieser in der letzten Section mit dem Element **X** definiert.

```
<MP U="8576C1A02F9340A1BA73704ABEF8B70F" v="021" a="1" z="2" l="De-de">
  <P g="Erika-Annemarie" f="Demon-Lier" egk="N994842106" b="1984-10-19" s="W"/>
  <A n="Apotheke Winfried Wagenmüller" s="Hauptstraße 55" z="12348"
     c="Berlin - Treptow" p="030-12345678" e="apotheke@website.de" t="2015-12-15"/>
    <C s="http://kbv.de/BSNR" v="218099900"/>
    <O w="125" h="188" c="1.3" a="Katzenhaare" ai="Laktose" b="1" p="1" />
    <S>
      <M p="766699" m="1" d="1" v="1" h=" " du="5" i="kompletter Sprühstoß" r="Heuschnupfen"/>
      <M f="SUS" t="alle 8 Stunden 1" du="&lt;" i="max. 3 Tage (1 Messlöffel=5ml)"
      <M p="4274616" m="1" du="1" i="nach der Mahlzeit" r="erhöhte Blutfette"/>
         r="Harnwegsinfekt">
        <W w="Hydrochlorothiazid" s="25mg"/>
        <W w="Olmesartan Medoxomil" s="40mg"/>
      </M>
      <X>Hautsalbe (Polidocanol-600-Zinkoxidschüttelmixtur 5% 2x täglich auf Handflächen
         nach dem Händewaschen</X>
    </S>
    <S c="418">
      <M f="TAB" t="1-0-1-0-1-0" du="1" i="bei Bedarf">
        <W w="Diphenhydramin-HCl" s="50mg"/>
      </M>
    </S>
    <S t="Sehr wichtige Angaben">
      <X>Bitte messen Sie Ihren Blutdruck täglich! Nächster Impftermin: 24.02.2016. Bei Rissen
         in der Hornhaut bitte Desinfektion auftragen.</X>
    </S>
</MP>
```

Abbildung 4.6: Nach dem Einlesen des Datamatrix-Codes liefert der Handscanner eine Zeichenkette, die den BMP im Ultrakurzformat enthält. Zur Vereinfachung wurde der String übersichtlich formatiert. Standardmäßig wird der UKF-String in einer Zeile eingelesen.

4.3.3 Besonderheiten beim Parsen

- Beim Erstellen der FHIR-Elemente wird mit der SQLite-Datenbank interagiert, indem die Titel der Sections sowie die Daten zu einer PZN-Nummer verwendet werden

- Beim Parsen von mehrseitigen BMPs wird darauf geachtet, dass die Medikationen, deren Section auf der nächsten Seite fortgesetzt wird, der richtigen Section zugeordnet werden

- Falls eine Section mehrere gebundene Freitextzeilen aufweist, werden sie der Reihenfolge entsprechend zur einer Freitextzeile zusammengeführt

- Hat die letzte Section lediglich ein X-Element und keine anderen Bestandteile, wird dieser Abschnitt nicht in die Medikation mit aufgenommen

- Eine bestimmte Anzahl von Sonderzeichen (<, >, ', &, ..) werden gesondert umgewandelt, weil diese in der XML-Spezifikation[20] für die Beschreibung der XML-Elemente belegt sind

4.3.4 Interpretation des HL7-FHIR-Dokuments

Nachdem ein valides HL7-FHIR-Dokument erstellt wurde, wird mit dem FHIR-Parser das Dokument interpretiert, um damit die Informationen auf der Benutzeroberfläche des *MedDoser* zu visualisieren.

Aufbau eines HL7-FHIR-Dokuments Der BMP in UKF-Form, der in ein FHIR-Dokument konvertiert wird, ist eine Sammlung von FHIR-Profilen, die verwendet werden, um den gesamten Medikationsplan darstellen zu können. Das Dokument wird immer in der selben Reihenfolge erstellt. In der Abbildung 4.7 wird die Reihenfolge des Aufbaus dargestellt. Am Anfang werden immer die Metadaten, der Patient und der Ersteller genau einmal erstellt. Falls ein Dokumentenverwalter existiert, der als Organisation aus dem UKF-String bekannt ist, wird er mit ausgedruckt, ansonsten wird er weggelassen. Danach können die klinischen Parameter, die Allergien und Unverträglichkeiten sowie die Gesundheitsbelange folgen. Jetzt folgen die aktuellen Medikationen und am Ende des FHIR-Dokuments kann eine wichtige Angabe hinterlegt werden, die keinen Bezug auf einen Medikationseintrag besitzt.

[20] https://www.w3.org/TR/xml/, Aufruf am 22.04.2018

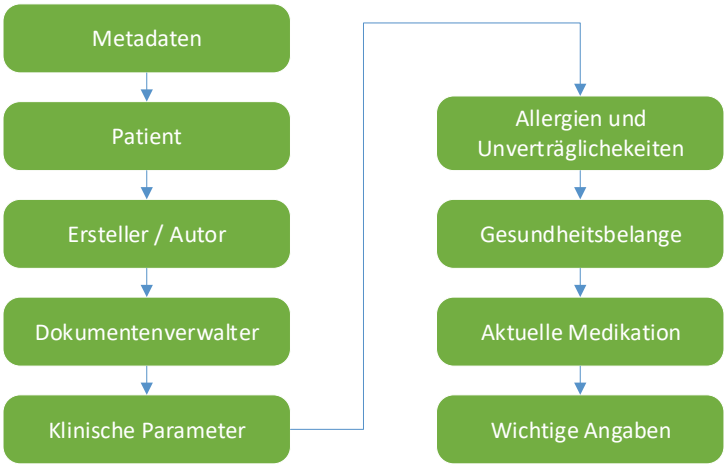

Abbildung 4.7: Der Aufbau eines HL7-FHIR-Dokuments folgt einer bestimmten chronologischen Reihenfolge, die immer gleich ist [Quelle: Eigene Darstellung].

Technischer Aufbau des HL7-FHIR-Dokuments Ein sogenanntes **Bundle** wird als **Container** bezeichnet. Ein FHIR-Bundle beinhaltet eine *Composition*, die im FHIR-Dokument mit dem Element *Entry* eingeleitet wird. Die Composition beschreibt die Referenz zum Patienten, den Ersteller, die Organisation, die klinischen Parameter, die Allergien und die Intoleranzen, die Gesundheitsbelange und den Medikationsplan. Die Referenzen aus den Basisinformationen referenzieren dann auf die eigentlichen Ressourcen im Bundle, die als *Entries* erstellt werden. In der Grafik 4.8 wird das Zusammenspiel der Composition und den Ressourcen gezeigt.

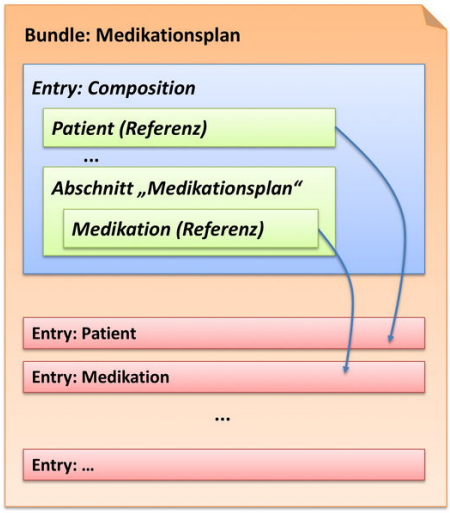

Abbildung 4.8: Der HL7-FHIR Container wird als Bundle bezeichnet, der im Entry-Element die Basisinformationen der Observation und der Medikationstabelle besitzt, die durch Referenzen bei anderen Ressourcen beschrieben werden [HL717].

Im Bundle-Beispiel 4.9 wird gezeigt, wie diese Auflistung in der Praxis vorkommen kann.

Das Bundle enthält den Typ, der in der Zeile sechs sowie die zwei Entries zu den Compositionen, die zwischen den Zeilen neun und vierzehn sowie zwischen den Zeilen 15 und 21 beschrieben worden sind. Ab der Zeile 22 ist es möglich, weitere Ressourcen mit Entries aufzulisten.

```
<Bundle xmlns="http://hl7.org/fhir">
    <meta>
        <versionId value="201"/>
        <profile value="http://fhir.hl7.de/medikationsplan/bundle"/>
    </meta>
    <type value="document"/>
    <entry>
        <fullUrl value="http://mein.medikationsplan.de/composition/1433e0a7-...-5280508da565"/>
        <resource>
            <Composition>
                ...(mit Referenzen zu Patient, Medikationsplan-Einträgen usw.)
            </Composition>
        </resource>
    </entry>
    <entry>
        <fullUrl value="urn:uuid:e3e65616-8fd4-427urnd-b560-d847c2ca0a3a"/>
        <resource>
            ... (Entry: in der Composition referenzierte Resourcen wie Patient,
                 Medikationsplan-Einträgen usw.)
        </resource>
    </entry>
    (weitere Entry-Einträge)
</Bundle>
```

Abbildung 4.9: Die FHIR-Bundles referenzieren auf die eigentlichen Compositions, die als Ressource beschrieben wurden.

4.4 Nebenläufigkeit (Threads)

Damit der Betrieb des *MedDoser* dauerhaft auch über einen längeren Zeitraum funktioniert, überprüfen mehrere **Threads** periodisch den Zustand des Systems sowie der Datenbestände und führen festgelegte Aktionen aus. Im Folgenden sind die drei implementierten Threads im Detail beschrieben:

4.4.1 Erstellen von Einträgen in der XML-Datei

Beim erstmaligen Einscannen eines Medikationsplans mit dem *MedDoser* wird für jede Einnahme eines Medikaments des aktuellen Tages ein Eintrag in der XML-Datei (Siehe Abschnitt 4.1.1) generiert.

Falls das Gerät über mehrere Tage hinweg ausgeschaltet war, werden bei Programmstart für jeden Tag, der seit dem letzten Tag, an dem das Gerät eingeschaltet war, vergangen ist, Einträge in der XML-Datei erstellt.

Zusätzlich überprüft ein Thread alle 60 Sekunden, ob ein neuer Tag begonnen hat. Ist dies der Fall, werden automatisch die Einträge für den neuen Tag in der XML-Datei generiert und die Ansicht im Hauptfenster aktualisiert sich.

4.4.2 Uhrzeit

Das aktuelle Datum sowie die Uhrzeit wird unterhalb des *MedDoser*-Logos in einem Label angezeigt. Weil der Zustand eines Labels statisch ist, muss die Uhrzeit regelmäßig aktualisiert werden. Hierzu legt der Thread alle fünf Sekunden die neue Uhrzeit als Text des Labels fest.

4.4.3 Erinnerung an die Einnahme

Der Thread überprüft alle acht Sekunden, ob Medikationen noch nicht quittiert wurden. Hierbei werden Medikationen vom aktuellen Tag sowie der Vergangenheit in Betracht gezogen, die eine bestimmte Einnahmezeit definiert haben. Medikamente, die zu besonderen Zeiten eingenommen werden müssen, sind hiervon nicht eingeschlossen. Sobald mindestens eine Einnahme noch nicht vom Anwender quittiert wurde, beginnt er mit der Ausführung folgender Aktionen:

- **Farbliches Hervorheben der Tabs**:
  Alle Tabs für die Einnahmezeiten des heutigen Tages, welche unbestätigte Medikationen enthalten, werden hervorgehoben
- **Aktivieren und Deaktivieren von LED und Lautsprecher**:
  Die GPIO-Pins des Lautsprechers und der LED werden aktiviert und sie beginnen, den Erinnerungston abzuspielen beziehungsweise zu blinken. Der Thread bemerkt,

wenn der Benutzer alle Einnahmen erledigt hat und deaktiviert die beiden Komponenten in diesem Fall wieder

4.5 Benutzeroberfläche

Die grafische Schnittstelle zur Anwendung hat aufgrund der Art der Stakeholder einen hohen Stellenwert, weil von mangelnder Erfahrung und Fähigkeiten im Umgang mit digitalen Endgeräten ausgegangen werden muss. Die Oberfläche wurde möglichst aufgeräumt und schlicht gehalten, um eine einfache Bedienung ohne Einarbeitungszeit zu ermöglichen. Neue Fenster werden immer modular und maximiert geöffnet, zudem wurde auf Metaelemente wie Fenster- oder Taskleisten verzichtet. Der standardmäßig auf der Konsole angezeigte Startvorgang des RaspberryPi wird durch das Erscheinen des *MedDoser*-Logos verborgen (Siehe Abschnitt 3.2).

Interaktionsmöglichkeiten des Benutzers Der Benutzer kann auf folgende Arten mit dem Gerät interagieren:

- Festlegen der Uhrzeit für die Erinnerung an die ausstehende Einnahme (Pro Einnahmezeit: Morgens, mittags, abends und zur Nacht)
- Bestätigen oder Ablehnen der Einnahme *eines einzelnen* Medikaments oder *aller* Medikamente eines Einnahmezeitpunkts
- Festlegen der Einnahmezeit eines oder aller Medikamente, falls sie von der Standardzeit abgewichen ist
- Einscannen eines (mehrseitigen) neuen Medikationsplans

Die akustische und visuelle Erinnerungsfunktion wird automatisch ohne Zutun des Anwenders aktiviert oder deaktiviert, je nachdem ob noch Medikamenteneinnahmen ausstehen oder nicht.

4.5.1 Die Bibliothek JavaFX

Die Benutzeroberfläche wurde auf Basis von **JavaFX** erstellt. Die JavaFX-APIs sind ab der Version 7 vollständig in die Laufzeitumgebung (JRE) und das JDK von Java integriert. Dies ermöglicht das Erstellen von plattformunabhängigen Benutzeranwendungen und das Nutzen von allen standardmäßigen Java-Bibliotheken[21] in der UI.

Das Gestalten der Oberfläche kann auf klassische Art über die Entwicklungsumgebung oder durch den *JavaFX SceneBuilder*[22] erfolgen. Die Beschreibung des Layouts ist in der

[21] https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm, Aufruf am 03.03.2018

[22] http://www.oracle.com/technetwork/java/javase/downloads/sb2download-2177776.html, Aufruf am 03.03.2018

Sprache XML definiert und in Dateien mit der Endung *.fxml* gespeichert. Für jede Stage (Siehe Abschnitt 4.5.1) existiert eine eigene XML-Datei, welche sich im Projektordner *main/resources* befindet. Das Aussehen und teilweise das Verhalten der UI-Elemente der Java-Anwendung wird mit einer *.css*-Datei gesteuert, die im selben Ordner abgelegt ist und jeweils im Ankerelement einer .fxml-Datei referenziert wird. Die Klasse, welche die Stage initialisiert und mit Logik versieht, wird wie in Abbildung 4.10 ebenfalls im Ankerelement angegeben:

```
<GridPane fx:id="gridPaneRoot" stylesheets="@meddoser.css"
    fx:controller="guifx.TimesController">
```

Abbildung 4.10: Die Definition des Ankerelements der *.fxml*-Datei referenziert die css-Datei für die Formatierung sowie die Controller-Klasse für die Logik. Mit dem Attribut 'fx:id' wird das UI-Element im Quelltext identifiziert.

Basismethoden von JavaFX Jedes Fenster einer Anwendung basiert auf einer sogenannten *Stage*, die mehrere *Scenes* enthalten kann. Eine Scene wiederum enthält einen *sceneGraph*, der die für die Darstellung nötigen UI-Elemente, wie beispielsweise Listen, Buttons und Tabellen, enthält [Ora]. Eine Stage wird durch folgende Methoden gesteuert:

- Launch() zum Starten des Lebenszyklus der Anwendung
- Init() zur Initialisierung von globalen Parametern
- Start() lädt das Layout aus der *.fxml*-Datei, instanziiert Controller-Klassen und erstellt die primäre Stage
- Stop() – Optionale Methode zum Beenden laufender Verbindungen, z.B. zur Datenbank

Die statische Einstiegsroutine *main* der Anwendung muss den Aufruf *launch(args)* enthalten, mit dem die JavaFX-Instanz aufgebaut wird. Eine kompakte Übersicht in deutscher Sprache über die Architektur von JavaFX bietet dieser Beitrag[23].

JavaFX im Vergleich zu Swing Zu Projektbeginn wurden die ersten Entwürfe der Benutzeroberfläche mit dem Java-Framework *Swing* gestaltet. Schnell stellte sich jedoch heraus, dass das Aussehen der Elemente nicht zeitgemäß und sich ihre Verknüpfung mit den Daten im Hintergrund als umständlich gestaltete. Diese Probleme wurden durch JavaFX behoben, zudem vereinfacht und beschleunigt das bereits erwähnte Programm *JavaFX SceneBuilder 2.0* von Oracle das Erstellen von Stages und Scenes erheblich.

[23] https://www.heise.de/developer/artikel/Visualisierung-in-Java-mit-JavaFX-1902233.html?seite=all, Aufruf am 03.03.2018

4.5.2 Aufbau der UI-Elemente

Die Benutzeroberfläche setzt sich aus mehreren Stages und Scenes zusammen. Ihre Strukturen, ihr Aussehen und ihre Aufgaben werden im Folgenden erklärt.

Hauptfenster

Das Hauptfenster ist äquivalent zur primären *Stage* und Ausgangspunkt für alle im Folgenden beschriebenen Funktionen. Ihre Hauptbestandteile im oberen Bildschirmbereich sind das *MedDoser*-Logo samt aktueller Uhrzeit und Datum sowie vier Buttons als Zugangsmöglichkeit zu den weiteren Funktionen. Den größten Teil des Touchscreens füllt eine *TabPane*, welche die noch ausstehenden Medikationen für den aktuellen Tag, unterteilt in fünf verschiedene Kategorien, in einzelnen Tabs anzeigt. Diese Tabs sind, wie in Abbildung 4.11 ersichtlich, mit den vom Benutzer festgelegten Einnahmezeiten beschriftet.

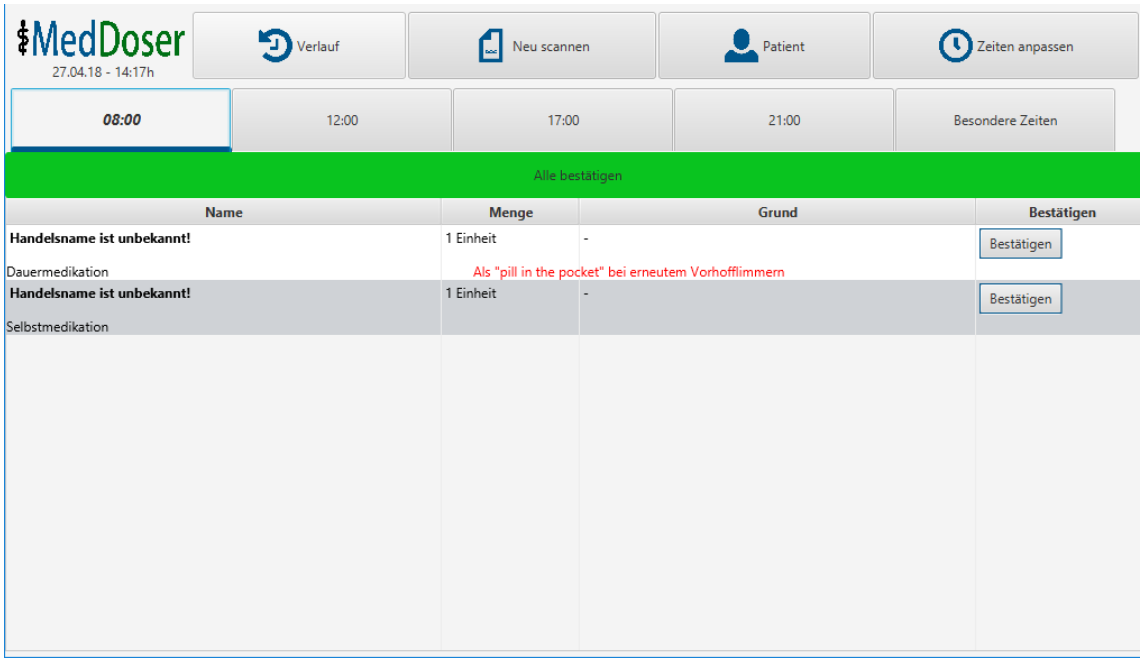

Abbildung 4.11: Das Hauptfenster ist für den Anwender der Einstiegspunkt in die Anwendung. Hier erhält er einen Überblick über die ausstehenden Medikationen des aktuellen Tags und hat Zugriff auf die weiteren Funktionen des *MedDoser* [Quelle: Eigene Darstellung].

Die ersten vier Tabs enthalten die Medikationen, die pro Tageszeit im Medikationsplan vom Arzt definiert wurden. Der letzte Tab enthält Medikationen, die zu unregelmäßigen Zeiten eingenommen werden sollen und daher keiner Tageszeit zugeordnet werden können. Sie verbleiben in dieser Ansicht auch nach Quittierung vom Anwender, im Gegensatz zu den Medikationen der festgelegten Einnahmezeitpunkte.

Bei der Implementierung des Hauptfensters wurden unter anderem folgende Funktionen berücksichtigt:

- Das **automatische Selektieren** des der aktuellen Tageszeit entsprechenden Tabs bei Programmstart

- Die der *CorporateIdentity* (Siehe Abschnitt 4.5.3) entsprechende **optische Hervorhebung von Tabs**, die Medikationen enthalten, die noch nicht durch den Anwender quittiert wurden

- **Deaktivieren von Buttons** zur Bestätigung von Einnahmen, falls deren Einnahmezeitpunkte noch nicht eingetreten sind

- Die **Anzeige der genauen Uhrzeit** der Quittierung, falls die zugehörige Einnahme bereits quittiert wurde

- Trennung von Texten, welche die Zellenbreite in der Tabelle überschreiten, durch einen **Zeilenumbruch**

- **Einblendbarer Freitext** einer Section des Medikationsplans, falls dieser existiert

- Die Anzeige der **Überschrift einer Section** zu jedem Datensatz, beispielsweise *Selbstmedikation*

- Die Anzeige der **Notiz einer Medikation** über die volle Breite des Displays in roter Schrift

Bestätigen oder Ablehnen einer Einnahme

Wenn der Anwender die Einnahme eines Medikaments bestätigt oder ablehnt, verschwindet dieser Datensatz aus der Tabelle des aktuellen Tabs. Mehrere Medikationen lassen sich auch **gemeinsam** bestätigen oder ablehnen. In jedem Fall enthält das sich öffnende Fenster nochmals eine Liste der ausgewählten Medikationen sowie einen *Switcher*, mit dem die tatsächliche Einnahmezeit vom Benutzer angepasst werden kann. Die voreingestellten Werte entsprechen hierbei den definierten Erinnerungszeiten. Buttons zum Bestätigen oder Ablehnen der Einnahme sowie ein Button zum Abbrechen der Aktion führen zurück ins Hauptfenster und aktualisieren die Ansicht, wie in Abbildung 4.12 zu sehen.

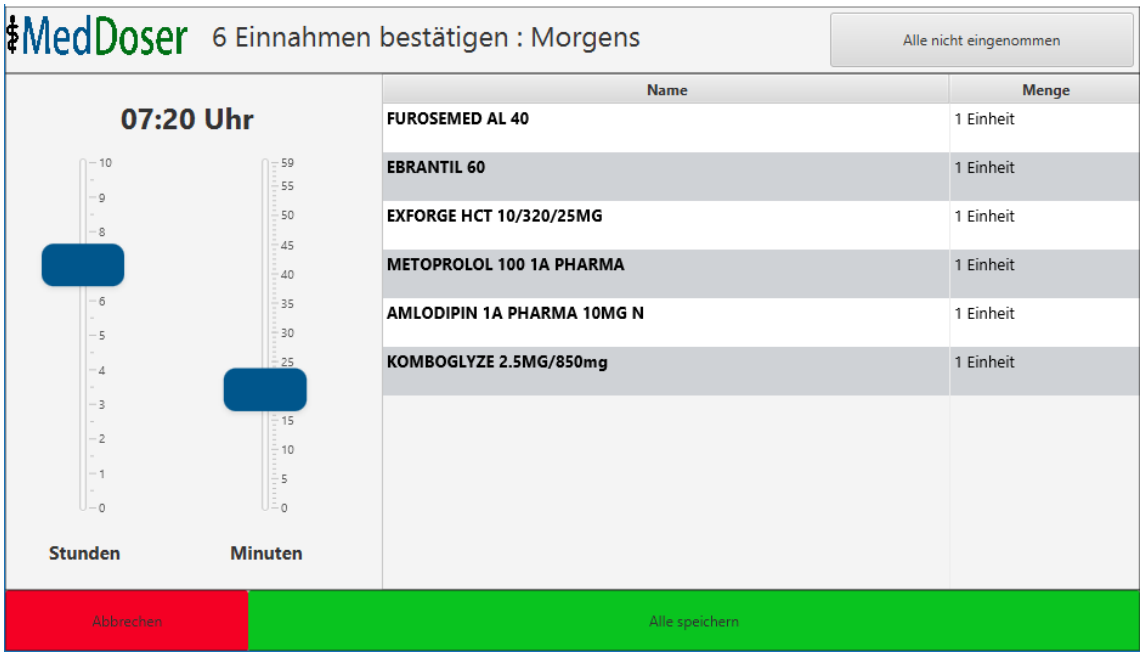

Abbildung 4.12: Einnahmen können einzeln oder gemeinsam quittiert werden: Der Anwender hat die Wahl zwischen dem Bestätigen der Einnahme und der Angabe, dass er sie nicht getätigt hat. Die genaue Uhrzeit lässt zudem sich für die gewählten Medikationen variabel einstellen [Quelle: Eigene Darstellung].

Verlauf

Da der Medikationsplan für eine Anwendung der Medikation von bis zu 28 Tagen ausgelegt ist, muss die Übersicht des Einnahmeverlaufs mindestens diese Anzahl von Tagen unterstützen. Um zudem der Bedienung mit dem Finger gerecht zu werden, befindet sich am oberen Bildschirmrand ein *Switcher*, der jeweils bis zu sieben Tage einer Kalenderwoche zur Auswahl anzeigt (Siehe Abbildung 4.13). Durch Betätigen von einem der beiden Buttons an seinem linken und rechten Rand hat der Anwender die Möglichkeit, zwischen den Wochentagen zu wechseln. Befindet sich die aktuelle Auswahl auf dem am Weitesten links oder rechts möglichen Wochentag, wechselt der Switcher die Ansicht zur entsprechend vorangehenden oder folgenden Kalenderwoche. Bei nur einem verfügbaren Wochentag (Beispielsweise nach dem initialen Start des *MedDoser*) sind beide Buttons deaktiviert, da keine Möglichkeit des Wochentagwechsels besteht.

Die aktuelle Kalenderwoche wird links neben dem Switcher angezeigt. Wochentage, für die noch unbestätigte Medikationen ausstehend sind, werden bis zur Erledigung aller Einnahmen in roter Schrift markiert.

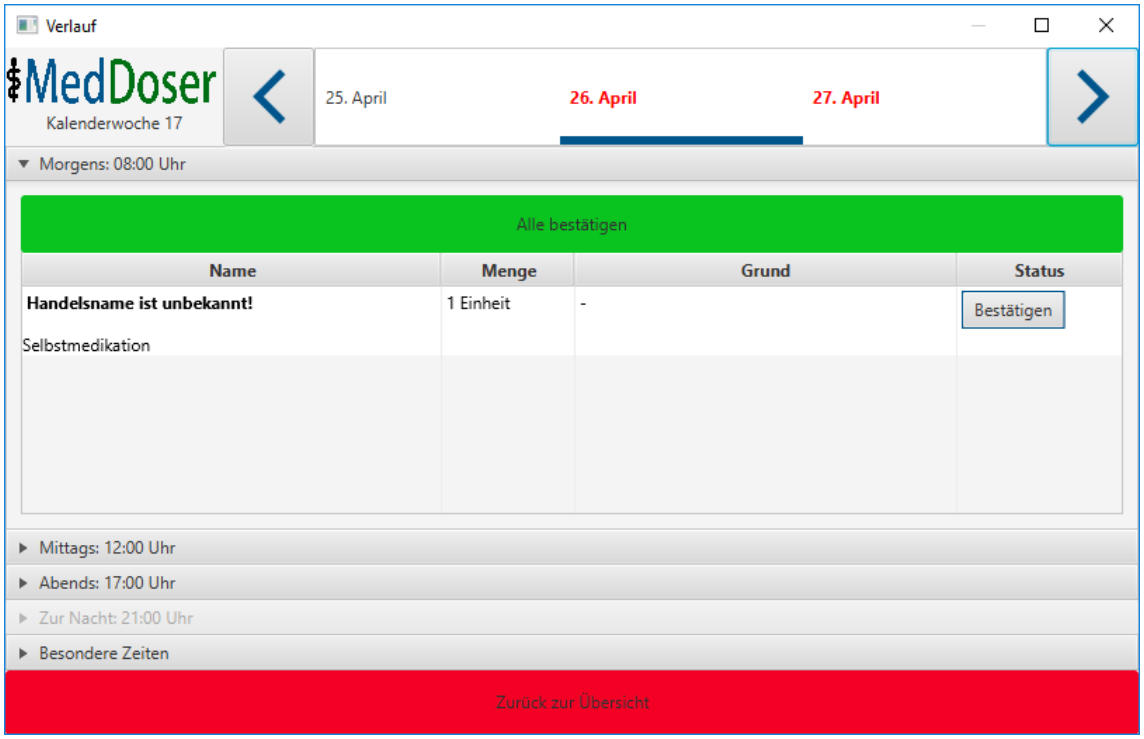

Abbildung 4.13: Im Verlauf lassen sich sämtliche Einnahmen aller Tage, gruppiert nach Einnahmezeitpunkt, seit dem Einscannen des Medikationsplans nachvollziehen. Durch den Switcher lässt sich zwischen den Tagen wechseln und Einnahmen im Nachhinein bestätigen [Quelle: Eigene Darstellung].

Im Verlauf lassen sich Einnahmen der vergangenen Tage im Nachhinein bestätigen. Auch hier hat der Anwender noch die Möglichkeit, die Uhrzeit der Einnahme zu variieren, jedoch ist er dabei auf das entsprechende Datum beschränkt. Der aktuell vom Benutzer selektierte Wochentag zeigt die Medikationen und deren Stati in einem *Accordion*-Element, dessen *TitledPanes* sich auf- und zuklappen lassen. Jede *TitledPane* ist mit der vom Benutzer festgelegten Erinnerungszeit beschriftet und deaktiviert, falls noch keine Medikamente für diese Uhrzeit bestätigt wurden und angezeigt werden können. Auch in dieser Ansicht ist die quittierte Einnahmezeit pro Medikament ersichtlich. Bei einer Medikation ohne festgelegte Einnahmezeit wird eine Liste aller Zeiten angezeigt, zu denen das Medikament bestätigt wurde.

Einscannen eines Medikationsplans

Der Prozess für das Einlesen von ein- oder mehrseitigen neuen oder aktualisierten Medikationsplänen ist bereits ausführlich in Abschnitt 4.2 beschrieben. Das zugehörige Fenster in Abbildung 4.14 enthält neben dem obligatorischen Logo und der Überschrift einen großen Button mit wechselndem Textinhalt, mit welchem das Scannen gestartet wird. Sobald der Button betätigt wird, aktiviert sich ein verborgenes Textfeld, in dem die resultierende Zeichenkette vom Handscanner entgegengenommen wird. Je nach Erfolg des Parsens erhält der Anwender eine Rückmeldung, ob die zuvor eingescannte Seite valide war, welche Seiten

des Plans noch fehlen oder ob der Handscanner die richtige Konfiguration aufweist. Nach erfolgreichem Scanvorgang öffnet sich automatisch das Hauptfenster.

Abbildung 4.14: Die Benutzerhinweise beim Einscannen des Medikationsplans variieren je nach Erfolg des Scanvorgangs. Bei mehrseitigen Plänen wird die aktuelle sowie die Gesamtseitenzahl angegeben. Nach Abschluss des Scanvorgangs öffnet sich das Hauptfenster mit den neuen Inhalten automatisch [Quelle: Eigene Darstellung].

Informationen zum Plan, Patienten und Arzt

Die Metainformationen, die neben den Medikationen im UKF-Format kodiert sind, werden in einer separaten Stage *ohne Interaktionsmöglichkeit* präsentiert, da sie statisch und nicht modifizierbar sind. Das zweispaltige Layout gibt alle verfügbaren Informationen aus dem Medikationsplan wieder, die den Patienten sowie den Arzt beziehungsweise die Praxis identifizieren (Siehe Abschnitt 4.3.1). Das Erstellungsdatum des Plans befindet sich in der rechten oberen Ecke, wie in Abbildung 4.15 ersichtlich.

Abbildung 4.15: Diese Stage präsentiert Informationen zum Patient und zum Arzt bwz. der Praxis sowie das Ausstelldatum und das Datum des letzten Scans [Quelle: Eigene Darstellung].

Anpassen von Einnahmezeiten

Die vier beschriebenen Einnahmezeiten wurden mit Standardzeiten für die Erinnerung versehen und mit folgenden Standardwerten versehen:

1. PCM (Morgens): 08:00 Uhr
2. PCD (Mittags): 12:00 Uhr
3. PCV (Abends): 18:00 Uhr
4. HS (Zur Nacht): 21:00 Uhr

Der Benutzer kann diese Zeiten stunden- und minutengenau mit Hilfe von vertikalen *Slidern* variieren, wie in Abbildung 4.16 zu sehen.

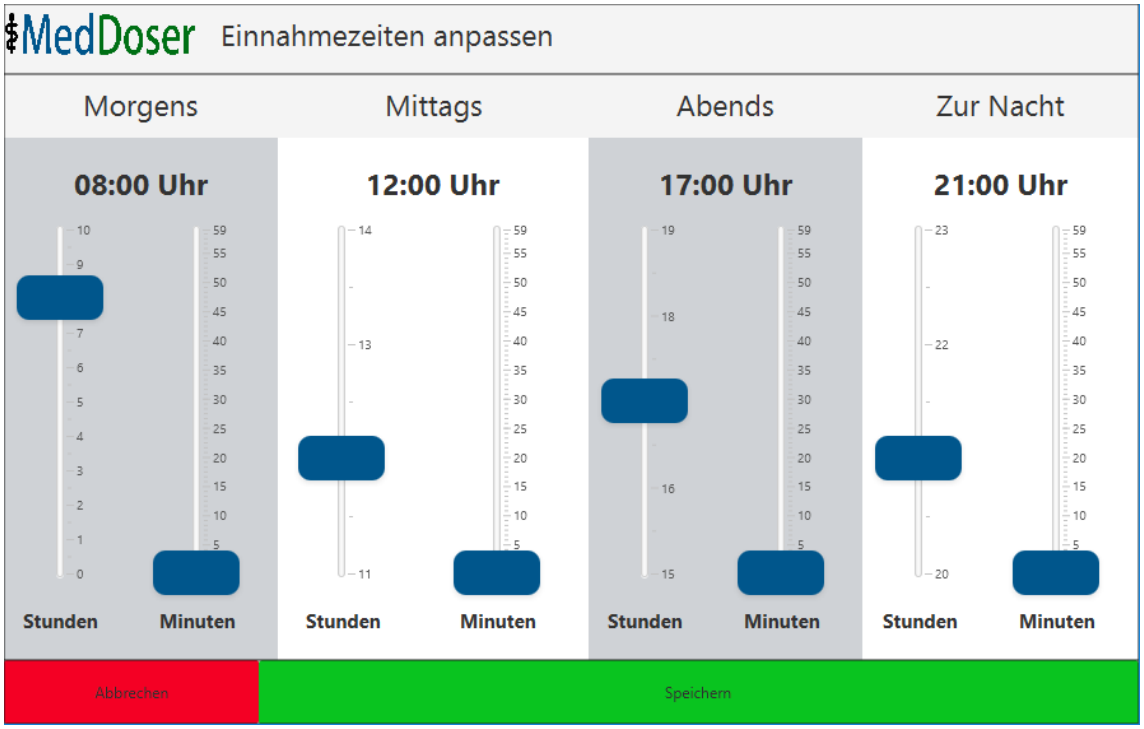

Abbildung 4.16: Die Erinnerungszeiten lassen sich vom Benutzer pro Tageszeit personalisieren. Die Beschriftungen der Zeiten in den anderen Stages passt sich den neuen Daten an, auch die akustische und visuelle Erinnerung richtet sich nach den hier definierten Werten [Quelle: Eigene Darstellung].

Nach dem Speichern der vorgenommenen Änderungen schließt sich das Fenster und die Erinnerungszeiten in den Tabs des Hauptfensters werden automatisch aktualisiert. Die Threads, welche für die akustische und visuelle Erinnerung zuständig sind, erfahren von diesen Änderungen über die Aktualisierung des Datenbestands in der XML-Datei.

4.5.3 Corporate Identity

Die verwendeten Farben in allen Projektdokumenten sowie in der Benutzeroberfläche orientieren sich an den Hauptfarben des *MedDoser*-Logos in Abbildung 4.17:

Abbildung 4.17: Das Projektlogo, das von zwei Farben dominiert wird [Quelle: Eigene Darstellung].

- Grün (#007015)
- Blau (#00568c)

5 Tests und Qualitätsmaßnahmen

5.1 Modultests mit JUnit

Für die wichtigsten Klassen und Methoden werden Modultests implementiert. Beim Kompilieren der Anwendung zu einer *.jar*-Datei werden sie automatisch ausgeführt und informieren den Entwickler darüber, ob seine letzten Änderungen am Code Auswirkungen auf andere Bereiche des Projekts hatten. Mit Hilfe eines IntelliJ-eigenen Features lässt sich die Zeilenabdeckung der Testfälle ermitteln.

Weitere Maßnahmen zur Erhöhung der Softwarequalität sind folgende:

5.2 Versionsverwaltung

Die Verwaltung aller Artefakte des Projekts erfolgt mit der Versionsverwaltung *git*[24]. Regelmäßige Commits auf dem Hauptzweig gewährleisten das Arbeiten ohne Wartezeiten und Versionsversatz. Das Repository wird während der Projektdauer auf Servern des RCBE betrieben und nach Projektende gemäß der gewählten Lizenz auf eine öffentliche Plattform migriert.

5.3 Dokumentation

Zur Dokumentation der Quelltexte wird *JavaDoc* verwendet. Die generierte HTML-Dokumentation wird nach Projektende auf der hochschulinternen Projekthomepage zur Verfügung gestellt.

5.4 Bugtracking

Das Bugtracking erfolgt mit dem Tool *mantis* und ist unter *mantis.rcbe.de* gehostet. An dieser Stelle werden auch offene Punkte und Notizen gespeichert, die jeweils einem Bearbeiter zugeteilt werden können.

5.5 Logging

Ein möglichst exakter Fehlerhinweis in einer Logdatei im Stammverzeichnis der Java-Anwendung führt zur Nachvollziehbarkeit von Fehlern bei beispielsweise fehlerhaften Ein-

[24] https://git-scm.com, Aufruf am 15.03.2018

gaben des Benutzers und oder bei Systemereignissen. Auch unkritische Ereignisse zur Laufzeit werden teilweise protokolliert, jedoch mit einem anderen *Loglevel* versehen. Beim initialen Programmstart wird die Logdatei automatisch angelegt. Sie bleibt nach Beenden der Software bestehen, wird bei erneutem Programmstart fortgeführt und besitzt eine Maximalgröße von 5MiB, maximal drei Dateien werden erstellt. Wird dieses Limit überschritten, werden die ältesten Einträge der ältesten Logdatei entfernt.
In der Implementierung findet die Bibliothek *java.util.logging.logger* Anwendung. In der Initialisierungsmethode *Init* wird der Logger gestartet und beendet. Beim Aufruf des Loggers wird die aktuelle Systemzeit sowie die entsprechende Meldung protokolliert. Ein beispielhafter Aufruf des Loggers zum Protokollieren einer Meldung ist in Abbildung 5.1 beschrieben.

```
1 try{
2     // SQL statement
3 }
4 catch (SQLException e) {
5     Logging.logger.log(Level.SEVERE, "Error while executing
          the SQL statement, e);
6 }
```

Abbildung 5.1: Exemplarischer Aufruf des Loggers mit der Meldung, die in der Logdatei protokolliert wird.

6 Ausblick

Im Hinblick auf eine Weiterentwicklung des *MedDoser* sind folgende Features denkbar:

- **Implementierung von Datenschutzmechanismen**:
  Die Abfrage eines Kennworts oder einer PIN beim Hochfahren des Geräts, um persönliche Daten vor Fremdzugriff zu schützen
- **Implementierung von Sicherheitsmechanismen**:
  Die Verschlüsselung des Speichers der microSD-Karte beziehungsweise der Einbau physischer Barrieren, um ein Entfernen des Speichers zu verhindern
- **Gewährleisten von Barrierefreiheit**:
  Das Vorlesen von Medikationen für blinde Patienten oder die Unterstützung einer Sprachsteuerung für motorisch eingeschränkte Anwender
- **Vermeiden von Verwechslungen**:
  Das Ermöglichen des Aufnehmens von Bildern der Verpackung oder vom Behälter des entsprechenden Medikaments, um Verwechslungen zu vermeiden
- **Erfassen von Zusatzinformationen**:
  Das Erfassen des Grunds, falls eine Medikation durch den Patienten als *nicht eingenommen* gekennzeichnet wird
- **Deaktivierung bei Nichtbenutzung**:
  Das Ausschalten des Bildschirms bei Abwesenheit des Patienten in Verbindung mit einem Bewegungssensor
- **Entwurf eines Gehäuses**:
  Um einen optisch und haptisch besseren Gesamteindruck des Geräts zu erhalten, ist der Bau eines Gehäuses mit einem 3D-Drucker denkbar
- **Integrieren eines Scanners**:
  Das Ersetzen des bisher extern anzuschließenden Scanners durch einen integrierten Scanner oder eine Kamera. Die Evaluierung des Scannens mit einer Kamera ist in Abschnitt 4.2 beschrieben
- **Optimieren der GUI**:
  Das Anzeigen von Ladevorgängen oder ähnlichem kann dazu beitragen, die Akzeptanz der Benutzeroberfläche beim Anwender weiter zu steigern

7 Fazit

An dieser Stelle werden die verwendeten Bibliotheken und Packages beschrieben sowie Statistiken zu den Quelltexten aufgezeigt. Anschließend folgt ein Fazit aus Sicht der Projektteilnehmer.

7.0.1 Packages

Die in Tabelle 7.1 beschriebenen Packages gliedern das Java-Projekt:

Name	**Beschreibung des Inhalts**
enums	Enumerationen
fhirparser	Interpretierung des HL7-FHIR-Dokuments
guifx	Benutzeroberfläche
init	Initialisierung globaler Variablen und Instanzen
logging	Steuerung des Loggers
medicationplaninterpreter	Speichern aller Informationen in Objekten
persistenceSQL	Datenspeicherung: Erstellen und Abfragen der SQLite-Datenbank
persistenceXML	Datenspeicherung: Erstellen und Modifizieren der XML-Datei
threads	Logik der Threads
ukfparser	Interpretierung des UKF-Strings und Generierung des HL7-FHIR-Dokuments
util	Globale und statische Hilfsmethoden

Tabelle 7.1: Alle Packages des Projektes und ihre Beschreibung, um Klassen logisch zu strukturieren.

7.0.2 Bibliotheken

Folgende Bibliotheken kommen unter anderem bei *MedDoser* zum Einsatz:

- log4J: Funktionalität des Loggers
- Hapi FHIR: Erstellen des HL7-FHIR-Dokuments
- Pi4J: Ansteuerung der GPIO-Pins
- wiringPi: Ausgabe eines Tons

- sqlite: Verbindung zur SQLite-Datenbank
- junit: Modultests

7.0.3 Statistiken

Die quantitative Erhebung von Kennzahlen lieferte das in Tabelle 7.2 ersichtliche Resultat:

Merkmal	**Quantität**
Codezeilen gesamt	18.400
Anteil Logik	54 %
Anteil Kommentare und Leerzeilen	30 %
Klassen	82
Commits gesamt	900
Zeitaufwand pro Teilnehmer	300 Stunden

Tabelle 7.2: Die Statistiken des Projekts bezüglich Quellcode, Versionsverwaltung und Arbeitsaufwand.

7.0.4 Persönliche Einschätzung und Danksagung

Das Integrieren von Soft- und Hardwarekomponenten und die gleichzeitige Orientierung an einer gut zu bedienenden Benutzeroberfläche machte das Projekt zu einem spannenden, aber auch umfangreichen und arbeitsaufwändigen Unterfangen. Im Laufe des Projekts benutzten wir viele neue Frameworks und Bibliotheken und probierten uns an der Steuerung der Hardware. Manche Gedanken und Ideen wurden nach einiger Zeit wieder verworfen, weshalb der Lerneffekt in den letzten Monaten sehr hoch war und uns viele neue Erkenntnisse beschert hat. Der *MedDoser* erfüllt alle im Vorfeld definierten Anforderungen und ist zu dem Produkt geworden, das uns vor Projektbeginn vorschwebte.

Wir bedanken uns herzlich bei unserem Betreuer, Herrn Dr. med Georgios Raptis, für die jederzeit professionelle und zuverlässige Betreuung während der gesamten Projektdauer. Regelmäßige Besprechungen zum Status des Projektfortschritts sowie die zeitliche und organisatorische Koordination bildeten eine wichtige Komponente dieser Arbeit, in der wir unsere Idee zu *MedDoser* im Umfeld der OTH Regensburg verwirklichen konnten.

Literatur

[Bun17] K. Bundesvereinigung. Medikationsplan. URL: http://www.kbv.de/html/medikationsplan.php (besucht am 22. 12. 2017).

[Ges17] B. für Gesundheit. Fragen und Antworten zur elektronischen Gesundheitskarte und zum E-Health-Gesetz. URL: https://www.bundesgesundheitsministerium.de/service/begriffe-von-a-z/e/e-health-gesetz/faq-e-health-gesetz/ (besucht am 14. 03. 2018).

[Gor] Gordon. WiringPi. URL: https://projects.drogon.net/raspberry-pi/wiringpi/ (besucht am 14. 03. 2018).

[Gus17] Gus. Raspberry Pi RTC: Adding a Real Time Clock. URL: https://pimylifeup.com/raspberry-pi-rtc (besucht am 14. 03. 2018).

[HL717] HL7. Ultrakurzformat Patientenbezogener Medikationsplan. URL: http://wiki.hl7.de/index.php?title=IG:Ultrakurzformat_Patientenbezogener_Medikationsplan#Patientenbezogener_Medikationsplan (besucht am 15. 03. 2018).

[Key] Keyence. What is a DataMatrix code? URL: https://www.keyence.com/ss/products/auto_id/barcode_lecture/basic_2d/datamatrix/ (besucht am 12. 12. 2017).

[Ora] Oracle. Java Platform, Standard Edition (Java SE) 8. URL: https://docs.oracle.com/javase/8/javase-clienttechnologies.htm (besucht am 14. 03. 2018).

Tabellenverzeichnis

Abbildungsverzeichnis