HTL - Perg

Höhere Abteilung für Informatik

Diplomarbeit

EyeOn-Aid

Unterstützungssoftware für hilfsbedürftige Menschen und deren Betreuer

Projektteam:	Dominik Hackl
	Marcel Luckeneder
Projektbetreuer:	Prof. Dipl.-Ing. Dietmar Wokatsch-Ratzberger

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Ort, Datum	Dominik Hackl
Ort, Datum	Marcel Luckeneder

Danksagung

Hiermit bedanken wir uns bei all jenen, die uns während der Erstellung dieser Diplomarbeit unterstützten und Hilfe jeglicher Art boten.

Besonderer Dank gilt unserem Betreuungslehrer Prof. Dipl.-Ing. Dietmar Wokatsch-Ratzberger, von dem nicht nur die ursprüngliche Idee zur Software EyeOn-Aid stammte, sondern der uns auch während des gesamten Projektverlaufs bei technischen oder inhaltlichen Fragen zur Seite stand.

Außerdem bedanken wir uns bei unseren weiteren Lehrkräften, die uns ebenfalls bei Fragen äußerst hilfsbereit unterstützen konnten.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Kurzfassung

Das Ziel der Diplomarbeit ist die Erstellung der medizinischen Unterstützungssoftware EyeOn-Aid.

EyeOn-Aid ist eine Software zur Erleichterung des Alltags hilfsbedürftiger Menschen und deren Betreuer. Die Software bietet Betreuern eine einfache Verwaltung ihrer Patienten und deren gesundheitliche Daten. Betreuer haben jederzeit Kontrolle und Überblick über den Medikamentenkonsum ihrer Patienten und werden in Notfällen benachrichtigt, um schnell reagieren zu können.

Den Patienten bietet die Software per mobiler App für das Smartphone die Möglichkeit, einfach mit ihren Betreuern Kontakt aufnehmen zu können. Des Weiteren erhalten sie entsprechende Erinnerungen an Medikamenteneinnahmen, die jederzeit vom Betreuer angeordnet werden können. Mehr Sicherheit im Alltag bietet die Software durch den integrierten Sturzdetektor, welcher im Falle eines erkannten Sturzes automatisch den Betreuer alarmiert, sowie durch die Notfalltaste, womit Patienten mittels eines einzelnen Klicks den Betreuer alarmieren können.

1.2 Abstract

The goal of the thesis is the development of the medical assistance software EyeOn-Aid.

EyeOn-Aid is a software aiding the every day's life of people in need of help as well as their nurses. The software provides nurses with an easy management of patients and their health information. Nurses have an overview and control over their patients' medication intake while getting notified in case of emergency to ensure fast reaction.

Patients are provided with a mobile app for their smartphone which helps them to stay connected with their nurses easily. Furthermore they get reminded of upcoming medication intakes, prescribed by their nurses. To provide additional safety, the software has an

integrated fall detector which automatically alerts the nurse in case of a detected fall as well as an emergency button which provides a simple and quick way of alerting the nurse.

1.3 Motivation

Ältere, insbesondere an Krankheiten leidende, Menschen sind im Alltag oftmals zahlreichen Belastungen ausgesetzt. Diese Menschen müssen häufig viele verschiedene Medikamente einnehmen, wobei Zeitpunkt, Menge und das Medikament selbst nicht verwechselt werden dürfen. Wenn dazu noch physische oder psychische Schwächen, die sich etwa auf die körperliche Verfassung oder das Denkvermögen auswirken, auftreten, wird es für die Menschen immer schwieriger, selbständig den Alltag zu bewältigen. Hinzu kommt auch noch ein gewisses Sicherheitsrisiko, da es tragischerweise immer wieder vorkommt, dass Menschen in den eigenen vier Wänden stürzen und nicht mehr in der Lage sind, rechtzeitig Hilfe zu rufen.

Betreuer von hilfsbedürftigen Menschen, auf der anderen Seite, haben häufig das Problem, ihre Patienten nicht genügend im Auge behalten zu können. Ihnen fehlt eine einfache und schnelle Übersicht über die Verfassung ihrer Patienten. Oft besteht eine schlechte Kommunikation zwischen Betreuer und Patient, wodurch in Notfällen nicht schnell genug reagiert werden kann.

Kapitel 2

Grundlagen

2.1 Technologien

2.1.1 JavaScript

JavaScript ist eine Skriptsprache zur Programmierung dynamischer Webseiten und wird clientseitig im Browser ausgeführt.

Die Wurzeln von JavaScript reichen weit bis ins Jahr 1995 zurück. Damals präsentierte die Firma Netscape die Sprache LiveScript (früher Mocha), ein neues Feature und Bestandteil ihrer neuen Version des Netscape Navigators, mit welcher Entwickler auf die bisher nur statischen HTML-Seiten zugreifen konnten. Aus Vermarktungsgründung sowie der Beeinflussung von der Sprache Java verwendete man etwas später den Namen JavaScript [21]. Schnell reagierte der Konkurrent Microsoft auf die positive Entwicklung von JavaScript und integrierte die syntaktisch sehr ähnliche Skriptsprache JScript in die nächste Version des Internet Explorers. Eine Zeit lang spielten sich die beiden Giganten Netscape und Microsoft gegenseitig aus und versuchten durch ständige Updates, immer mehr Funktionalitäten ihrer Skriptsprachen zur Verfügung zu stellen. Aus Folge dessen, wurde das browserunabhängige Entwickeln von Webseiten immer schwieriger, da die beiden Skriptsprachen syntaktisch zwar beinahe identisch waren, beide jedoch vom Browser unterschiedlich interpretiert wurden [36]. Die Lösung entstand im Jahr 1996 in einer Zusammenarbeit von Netscape und der Standardisierungsorganisation ECMA, wobei der Standard ECMAScript als „cross-platform scripting technology for creating applications on the Internet and Intranets“ festgelegt wurde [22]. Ein Standard, an welchen sich zukünftige Entwicklungen der Browser hielten und wodurch der Erfolgskurs von JavaScript geebnet wurde.

Obwohl die primäre Rolle von JavaScript clientseitige Programme für Webseiten sind, hat sich die populäre Skriptsprache auch in anderen Bereichen etabliert. Den Einsatz von JavaScript auf der Server-Seite gab es zwar schon seit Einführung von JavaScript durch Netscape, bekannter wurde dieses Thema allerdings erst später mit Frameworks wie Node.js [1].

Neben dem Einsatz bei Webseiten und Web-Applikationen findet JavaScript bzw. andere Implementierungen des ECMAScript-Standards auch andere Einsatzbereiche. Als Beispiel hierfür Adobes ExtendScript, mit welchem Scripts für sämtliche Adobe-Software wie Photoshop erstellt werden können. [2]

2.1.2 Ajax

AJAX stand ursprünglich für „Asynchronous JavaScript And XML“. Da diese Definition allerdings etwas unzutreffend war, wurde der Begriff im Laufe der Zeit eher als Kunstwort anstatt als Akronym angesehen. Bei Ajax handelt es sich um ein Konzept zur asynchronen Kommunikation zwischen der Webseite und dem Server, welches es ermöglicht, Server- bzw. Datenbankoperationen abzuarbeiten, ohne die aktuelle Seite neu laden zu müssen. Die Technologie dahinter ist der sogenannte XMLHttpRequest, ein natives Browser-Objekt, welches zuerst in Microsofts Internet Explorer, dann aber schnell auch in allen anderen gängigen Browsern integriert wurde. Das XMLHttpRequest-Objekt basiert auf der simplen Idee, eine HTTP-Anfrage an den Webserver zu senden, die Rückgabe auszuwerten und das Resultat dann per JavaScript auf der Webseite darzustellen. [36]

Der Grundstein für die Technologie hinter Ajax wurde von Microsoft zu Zeiten des Internet Explorers 5 gelegt. Damals kam von Microsofts Office-Entwicklern die Anforderung einer Funktionalität für die Web-Schnittstelle von Outlook (ehemals „Outlook Web Access“) zur Erkennung von neu eingegangen Emails, ohne dabei permanent die Seite aktualisieren zu müssen. Das Feature wurde als ActiveX-Komponente für den Internet Explorer unter dem Namen XMLHttpRequest implementiert.

Der Begriff Ajax wurde zuerst von Jesse James Garrett im Jahr 2005 geprägt und entstand ein der Anfangszeit von Web-Applikationen, welche damals zwar bereits als innovativ und zukunftsorientiert angesehen wurden, jedoch in Punkto Benutzerfreundlichkeit bei weitem noch nicht mit Desktop-Applikationen konkurrieren konnten. Er beschreibt Ajax mit den Worten: „Several technologies, each flourishing in its own right, coming together in powerful new ways“, sprich einer Kombination aus bereits bekannten Technologien im Zusammenspiel mit den neuen Möglichkeiten von JavaScript. [8]

Google war zu diesem Zeitpunkt bereits ein Verfechter der Technologie und war eines der ersten Unternehmen, welche das Konzept von Ajax in vielen seiner Produkte verwendete. Einige Beispiele waren Google Suggest (die automatische Vervollständigung bei Sucheingaben, ein Feature welches heutzutage nicht mehr wegzudenken ist) oder Google Maps (der Benutzer konnte auf der Karte mit der Maus navigieren oder hineinzoomen, ohne dass die Seite neu geladen werden musste).

2.1.3 JavaScript Object Notation (JSON)

JSON ist ein Datenaustauschformat, welches syntaktisch auf der Sprache JavaScript basiert. Es wird vor allem bei Web-Applikationen in Kombination mit Ajax häufig verwendet

und bietet ein hervorragendes Zusammenspiel mit JavaScript, da sich JSON-Code nahtlos in ein JavaScript-Objekt umwandeln lässt. Ein JSON-Objekt beinhaltet eine beliebige Anzahl an Key-Value-Paaren, d.h. zu jedem Schlüsselattribut (durch eine Zeichenkette dargestellt) existiert genau ein Wert, welcher entweder ein Wert von einem primitiven Datentyp (Nullwert, boolescher Wert, numerischer Wert oder Zeichenkette), ein Array oder ein weiteres Objekt sein kann. Objekte können beliebig verschachtelt werden.

Abhängig vom Einsatzgebiet bietet JSON einige Vorteile gegenüber XML. Ein minimaler, bei größeren Datenmengen aber nicht unwesentlicher Unterschied stellt der Overhead dar, der bei JSON in der Regel sehr gering ausfällt, wodurch der Datenverkehr vermindert werden kann. Weiters ist die Syntax von JSON etwas einfacher aufgebaut, wodurch die Lesbarkeit etwas höher als bei XML erscheint. Der größte Vorteil stellt sich jedoch bei der Verwendung mit JavaScript und Ajax heraus, wobei die Rückgabe einer HTTP-Anfrage im JSON-Format mit der Funktion `eval()`, ein Bestandteil der Sprache JavaScript, direkt in ein JavaScript-Objekt umgewandelt werden kann, während die Verwendung von XML externe Parser erfordert. [20]

2.1.4 Ext JS

Ext JS ist ein JavaScript-Framework zur Erstellung von web-basierten Desktop-Applikationen, welches vom Unternehmen Sencha Inc. entwickelt wird. Seit dem Jahr 2007, in dem die erste Version 1.0 von Ext JS auf den Markt kam, wurde das Framework stetig erweitert, brachte mit der aktuellen Version 5.0, im Jahr 2014 erschienen, wieder zahlreiche innovative Neuerungen und befindet sich derzeit noch immer in aktiver Entwicklung. [19]

Das Framework basiert vollständig auf JavaScript. Bei der Ausführung eines Programms wird der JavaScript-Code vom Browser in eine HTML-Seite mit dazugehörigen Stylesheets umgewandelt. Dadurch dass die Seite nach dem Aufrufen im Normalfall nicht mehr neu geladen werden muss, sondern sämtliche Operationen per JavaScript abgearbeitet werden, macht es die Webseite für den Benutzer kaum von einer Desktop-Applikation unterscheidbar.

Ext JS-Klassensystem

Während JavaScript zwar objektorientiert aber klassenlos ist, basiert Ext JS wiederum auf einem Klassensystem. Die Ext JS API enthält über 300 Klassen für die unterschiedlichsten Anwendungsbereiche. Zur Strukturierung werden Namespaces (dt. Namensräume) verwendet, wobei jede Ebene, also jedes weitere Verzeichnis, mit einem Punkt getrennt wird. Eine beispielhafte Benennung wäre wie folgt: `MyApp.view.people.PeopleView`. Hierbei wäre `MyApp` das Wurzelelement der Applikation, `view` und `people` Verzeichnisse und `PeopleView` die eigentliche Klasse.

Eigene Klassen können mit der Funktion `Ext.define()` angelegt werden, wobei als erstes Argument der Namespace und als zweites Argument ein JSON-Objekt, welches die gesamte Klasse repräsentiert, übergeben werden. Ext JS-Klassen sind grundsätzlich im JSON-

Format aufgebaut und bestehen aus Config-Optionen, Properties, einem Konstruktor und Methoden.

Config-Optionen sind Attribute zum Konfigurieren eines Objekts, sprich Attribute, welche bei der Erzeugung eines Objekts der Klasse zugewiesen werden. Für jedes Config-Attribut werden automatisch demensprechende Getter- und Setter-Methoden angelegt, über welche auf die Attribute zur Laufzeit zugegriffen werden kann.

Properties sind Eigenschaften eines Objekts der Klasse und können zur Laufzeit beliebig verändert werden. Man unterscheidet zwischen öffentlichen und privaten Properties, wobei dies nur eine Angabe für den Entwickler ist und vom JavaScript-Interpreter nicht beachtet wird.

Der Konstruktor ist die erste Methode, die bei der Erzeugung eines Objekts der Klasse aufgerufen wird. Er erhält als Übergabeparameter das Config-Objekt und initialisiert dieses in der Regel auch mit der Methode `initConfig()`.

Methoden sind in einer Ext JS-Klasse nichts anderes als anonyme JavaScript-Funktionen, die einem Namen zugeordnet sind. Der Aufruf erfolgt gewöhnlich über `Objekt.Methode()`.

Bsp 2.1: Der Aufbau einer Ext JS-Klasse

```
1  Ext.define('Car', {
2      config : {
3          make  : undefined,
4          model : undefined
5      },
6      constructor : function (config) {
7          this.initConfig(config);
8      },
9      drive : function () {
10         Ext.Msg.alert('Car','Driving ' + this.getMake() + ' ' +
               this.getModel());
11     }
12 });
13
14 var car = Ext.create('Car', {
15       make  : 'Honda',
16       model : 'Accord'
17 });
18
19 car.drive();
```

Im Bsp 2.1 wird der Aufbau einer einfachen Klasse in Ext JS demonstriert. In Zeile 1 wird die Klasse durch die Funktion `Ext.define()` deklariert und der Name angegeben. In Zeile 2 bis Zeile 5 wird das Config-Objekt mit den zwei Attributen `make` und `model`, beide mit dem Standardwert `undefined`, angegeben. In Zeile 6 bis Zeile 8 ist der Konstruktor der Klasse implementiert, welcher lediglich bei der Instanziierung das Config-Objekt initialisiert. Die einzige Methode der Klasse wird in Zeile 9 bis Zeile 11 implementiert. Sie hat den Namen

`drive`, nimmt keine Übergabeparameter und gibt lediglich die Attribute des Objekts in einer Message-Box aus. Im Vergleich zu gewöhnlichem JavaScript, wo eine Funktion mit Namen deklariert wird (z.B. `function drive()`), wird hierbei eine anonyme Funktion erzeugt und der Variable `drive` zugewiesen. Die Erzeugung eines Objekts der Klasse `Car` findet nun in Zeile 14 bis Zeile 17 statt. Mit der Funktion `Ext.create()` wird ein Objekt erzeugt, worauf dessen Config-Attribute `make` und `model` Werte zugewiesen bekommen. In Zeile 19 wird die Methode `drive()` aufgerufen. [4]

Des Weiteren beherrscht Ext JS das objektorientierte Konzept der Vererbung, welches mit dem Attribut `extend`, wobei der Name der Basisklasse angegeben wird, realisiert werden kann. [14]

Components

Als Components werden in Ext JS sämtliche GUI-Komponenten bezeichnet. Das Framework liefert bereits eine breite Auswahl an vorgefertigten Components mit, welche alle von der Basisklasse `Ext.Component` abgeleitet sind. Daneben besteht auch die Möglichkeit, eigene Components zu erstellen und von bestehenden abzuleiten. Eine spezielle Art von Components sind Container, welche andere Components beinhalten können. So entstehen bei GUI-Komponenten typischerweise tief verschachtelte Strukturen.

Eine komfortable Technik ist die sogenannte Lazy Instantiation (dt. „faules Instanziieren“), was bedeutet, dass Components erst dann instanziiert werden, wenn sie auch benötigt werden. Beispielsweise werden bei einer Tab-Navigation einige Tab-Seiten früher benötigt, während andere gar nicht aufgerufen werden. In Ext JS wird dieses Konzept mittels des Attributs `xtype` umgesetzt. Jeder Component hat einen symbolischen Namen, welcher als xtype angegeben werden kann (z.B.: die Klasse `Ext.panel.Panel` hat den xtype `panel`). [15]

Bsp 2.2: Eine typische View in Ext JS

```
1  Ext.define('MyApp.view.MainView', {
2     extend: 'Ext.panel.Panel',
3     title: 'Main View',
4     items: [{
5        xtype: 'form',
6        items: [{
7           xtype: 'textfield',
8           name: 'vorname',
9           fieldLabel: 'Vorname'
10       }, {
11          xtype: 'textfield',
12          name: 'nachname',
13          fieldLabel: 'Nachname'
14       }]
15    }, {
16       xtype: 'image',
17       src: 'path/img.jpg'
```

```
18     }]
19 });
```

Bsp 2.2 zeigt eine typische View-Klasse, die von der Klasse `Ext.panel.Panel` abgeleitet wird. In Zeile 1 wird der Name der Klasse samt Namespace definiert. Durch das Attribut `extend` wird angegeben, dass die Klasse von `Ext.panel.Panel` abgeleitet werden soll. Zeile 3 legt den Titel fest, ein Attribut, das bereits von der Basisklasse geerbt wurde. Da es sich bei einem Panel um einen Container handelt, können eine beliebige Anzahl an Components untergeordnet werden. Diese werden ab Zeile 4 durch das Attribut `items`, ein Array von Components, angegeben. In Zeile 5 wird unter Verwendung des `xtype`-Attributs ein Form-Panel hinzugefügt. Bei diesem handelt es sich erneut um einen Container, welcher in diesem Fall die einzelnen Formularfelder, abgebildet in Zeile 6 bis Zeile 14, enthält. In Zeile 15 bis Zeile 18 wird dem Haupt-Panel nun ein weiteres Item, nämlich ein HTML-Image mit der angegebenen Quelle (Attribut `src`), hinzugefügt.

MVC und MVVM

Ext JS bietet die Möglichkeit, Applikationen in den Entwurfsmustern Model-View-Controller (MVC) oder Model-View-ViewModel (MVVM) zu erstellen.

MVC ist ein Software-Entwurfsmuster, welches die Benutzeroberfläche einer Applikation in drei Bestandteile gliedert: dem Model, der View und dem Controller. Das Model entspricht der Datenhaltungsklasse und gibt das Format vor, in welchem die Daten vorliegen müssen. Die View repräsentiert die Applikation nach außen und stellt die Daten für den Endbenutzer lesbar dar. Der Controller stellt das Bindeglied zwischen Model und View dar, verarbeitet sämtliche Ereignisse und Benutzerinteraktionen und leitet diese an die jeweils zuständige Stelle im Programm weiter.

Als Alternative zum MVC-Muster bietet Ext JS ab Version 5.0 das MVVM-Muster an. Neben den bereits aus MVC bekannten Komponenten Model und View bedient sich dieses Entwurfsmuster einer Eigenart, dem ViewModel. Wie der Name bereits sagt, verbindet dieses jeweils eine View mit einem dazugehörigen Model und verwaltet Änderungen zwischen den beiden Komponenten. Während diese Aufgabe bei MVC ein spezieller Controller übernehmen würde, wird sie bei MVVM direkt vom ViewModel gehandhabt. Die Implementierung von Konzepten wie Data-Binding wird dadurch sehr stark vereinfacht. Ein Controller ist bei MVVM zwar nicht zwingend notwendig, Ext JS bietet jedoch mit dem ViewController ebenfalls die Möglichkeit, Controller eigens für jede View anzulegen. [19]

Zusätzlich zu den beschriebenen Komponenten enthalten Ext JS-Applikationen meist sogenannte Stores. Ein Store ist ein clientseitiger Speicher von Model-Objekten, d.h. zu jedem Model wird ein dazugehöriger Store existieren, welcher die Objekte der jeweiligen Klasse verwaltet. Der Store hat Methoden zum Lesen, Hinzufügen und Löschen der Datenobjekte und synchronisiert diese mit einer angegebenen Datenquelle (z.B. am Server).

Sencha Cmd

Sencha Cmd ist ein freies Build Tool für die Entwicklung mit Ext JS. Es bietet Funktionen zum Kompilieren und Bereitstellen von Ext JS-Applikationen.

Bei der Inbetriebnahme von JavaScript-Applikationen ist es üblich, sogenannten „minified"-Code zu verwenden, d.h. den Code von sämtlichen unnötigen Zeichen wie Leerzeilen oder Kommentaren zu befreien, um den Datenverkehr bei Aufruf der Webseite zu minimieren. Der Compiler von Sencha Cmd erledigt diese Aufgabe an der kompletten Ext JS-Applikation und optimiert den Code soweit wie möglich. [13]

Mithilfe von Sencha Cmd kann zunächst das Grundgerüst einer neuen Ext JS-Applikation erstellt werden. Dazu wird in der Kommandozeile der Befehl `sencha -sdk /path/to/ext generate app MyApp /path/to/MyApp` ausgeführt. Dies erzeugt im angegebenen Verzeichnis die Standard-Verzeichnisstruktur und sämtliche Dateien, die für die Initialisierung der Applikation benötigt werden. Zusätzlich wird durch das Argument `-sdk` der Speicherort des SDKs angegeben, welcher dann in das Verzeichnis der Applikation kopiert wird.

Mit dem Befehl `sencha app build` wird die Applikation nun erstellt. Dabei werden alle JavaScript-Code-Dateien zu einer einzigen „minified" JavaScript-Datei (`app.js`) und sämtlicher Styling-Code zu einem CSS-Stylesheet zusammengefasst.

Ein weiteres handliches Werkzeug kann mit dem Befehl `sencha app watch` ausgeführt werden. Dadurch wird ein Webserver gestartet, auf dem die Applikation gehostet wird. Zusätzlich wird die Applikation nach jeder Änderung im Code automatisch neu kompiliert.

2.1.5 Sass

Sass (Syntactically Awesome Stylesheets) ist eine Erweiterungssprache für CSS (Cascading Stylesheets), mit welcher das Entwickeln von Stylesheets vereinfacht und besser organisiert wird. Es handelt sich dabei um eine einfache Skriptsprache, die in vollständig validen CSS-Code interpretiert wird. Sass erweitert CSS um diverse Mechanismen, die aus gebräuchlichen Programmiersprachen bekannt sind, darunter Variablen, Funktionen und Vererbung.

Mithilfe des Dollar($)-Operators können Variablen deklariert werden, welche dann beliebigen CSS-Attributen zugewiesen werden können. So kann beispielsweise eine auf einer Webseite häufig verwendete Farbe als Variable definiert werden. Bei einer Änderung muss der Farbwert nur einmal im Code angepasst werden, während sämtliche Attribute darauf referenzieren. Für Sass-Variablen stehen folgende Datentypen zur Verfügung: Zahlen, Zeichenketten, Farbwerte, boolesche Werte, der Nullwert, Listen von Werten sowie Maps (Liste von Key-Value-Paaren).

Mit Sass wird eine Vielzahl an Funktionen mitgeliefert, welche bei der Zuweisung von Variablen oder direkt bei der Zuweisung von CSS-Attributen aufgerufen werden können. Der Standard-Funktionsumfang reicht von der Manipulation von Farbwerten bis hin zur Bearbeitung von Listen oder Maps. Zusätzlich besteht auch die Möglichkeit, eigene Funktionen mithilfe der Skriptsprache Ruby zu implementieren.

Neben den genannten Techniken bietet Sass noch einige weitere Möglichkeiten, um CSS-Code einfacher und übersichtlicher zu gestalten. [29]

2.1.6 Document Object Model (DOM)

Das DOM ist ein vom W3C (World Wide Web Consortium) spezifizierter Standard und definiert die logische Strukturierung von HTML- bzw. XML-Dokumenten. Ziel ist es, eine plattform- und programmiersprachenunabhängige Schnittstelle zur Navigation und Manipulation von HTML-Inhalten zur Verfügung zu stellen. [34]

Das DOM ist in einer Baumstruktur organisiert, wobei jedes HTML-Element einen Knoten darstellt. Einer der wichtigsten Vorteile von DOM ist die Möglichkeit, aufgrund seiner Baumstruktur einfach Elemente hinzufügen und auch löschen zu können, ohne die Integrität zu gefährden. Eine wesentliche Rolle spielt das DOM im Zusammenhang mit Javascript und Ajax. Die einfache Möglichkeit, auf bestimmte Elemente im DOM zuzugreifen und diese zu modifizieren, ist ein wichtiger Bestandteil bei der Erzeugung von dynamischen Webseiten. Zudem war die Entwicklung des DOM ein wichtiger Schritt in Richtung Browser-Kompatibilität und bildet außerdem die Grundlage für JavaScript-Frameworks wie Ext JS. [36]

2.1.7 Android SDK

Zur Entwicklung von Apps für das Betriebssystem Android wird von Google die freie Android SDK Plattform angeboten. Apps werden in der Programmiersprache Java geschrieben und mithilfe der Werkzeuge des SDK kompiliert. Die Ausgabe erfolgt durch ein APK (Android Package), einem Archiv, in welches der gesamte Inhalt einer App gepackt wird. Das APK kann auf jedem Android Gerät, sofern es von der App unterstützt wird, installiert werden.

Komponenten einer Android-App

Eine Android-App kann aus vier verschiedenen Komponenten bestehen, die beliebig miteinander kombiniert werden können.

- Die Activity repräsentiert die App nach außen und besteht aus GUI-Komponenten und Benutzerinteraktionen. Für jede Ansicht bzw. für jeden Bildschirm in einer App wird grundsätzlich eine eigene Activity angelegt.
- Ein Service ist eine meist komplexere oder längerfristige Aufgabe, die die App im Hintergrund durchführt. Services laufen ohne Benutzeroberfläche und können auch außerhalb der App ausgeführt werden.
- Ein Content Provider verwaltet Datenquellen einer App. Durch einen Content Provider können Apps auf die öffentlichen Daten von anderen Apps zugreifen und diese manipulieren.

- Ein BroadcastReceiver empfängt Nachrichten, die über das gesamte Betriebssystem hinweg gesendet werden. Diese können Benachrichtigungen des Systems, wie etwa das Abschalten des Bildschirms, oder aber Broadcasts einer anderen App sein. Als Antwort auf empfangene Nachrichten kann der BroadcastReceiver beispielsweise eine Benachrichtigung in der Statusleiste des Geräts erstellen, üblicherweise wird die Nachricht aber an eine andere Instanz wie einem Service zur Verarbeitung übergeben.

Intents

Als Intent wird in Android eine asynchrone Nachricht zum Aufruf einer neuen Komponente bezeichnet. Mittels eines Intents können neue Activities, Services oder BroadcastReceiver aktiviert und zur Laufzeit beliebig verkettet werden. Wird eine Activity oder ein Service aufgerufen, wird die jeweilige Klasse angegeben und anschließend mit `startActivity()` bzw. `startService()` gestartet. Um der zu startenden Komponente zusätzliche Daten zu übergeben, können sogenannte Extras hinzugefügt werden. Extras sind einfache Wertepaare, welche bei der Erstellung des Intents definiert werden und beim Empfänger wieder ausgelesen werden können.

Das Android-Manifest

Das Android-Manifest ist eine Datei im XML-Format, welche die Eigenschaften und Komponenten einer App beschreibt. Die App analysiert dieses Dokument beim Start und stellt fest, welche Komponenten existieren. Nur Komponenten, die hier aufgelistet sind, können zur Laufzeit erstellt werden. Sämtliche Komponenten sind dem Element `Application` untergeordnet. In diesem Element werden anwendungsspezifische Eigenschaften wie Name oder Symbol der App definiert. Im Android-Manifest werden außerdem alle von der App benötigten Permissions aufgelistet. Permissions (dt. Berechtigungen) sind spezielle Zugriffsrechte, die aus Sicherheitsgründen für bestimmte Tätigkeiten (z.B. Zugriff auf das Telefonbuch des Benutzers) angefordert werden müssen. Ein weiteres Attribut im Manifest ist die `minSdkVersion`, also die für die App benötigte Mindestversion vom Android-Betriebssystem. Dadurch kann verhindert werden, dass die App auf einem inkompatiblen Gerät installiert wird.

2.1.8 Java Servlets

Java Servlets sind Programme, welche auf einem Web- oder Application-Server (z.B. Apache Tomcat) laufen und fungieren als Mittelschicht zwischen Anfragen, welche von einem Webbrowser oder HTTP-Client kommen, und einer Datenbank oder anderen Programmen auf dem HTTP-Server.

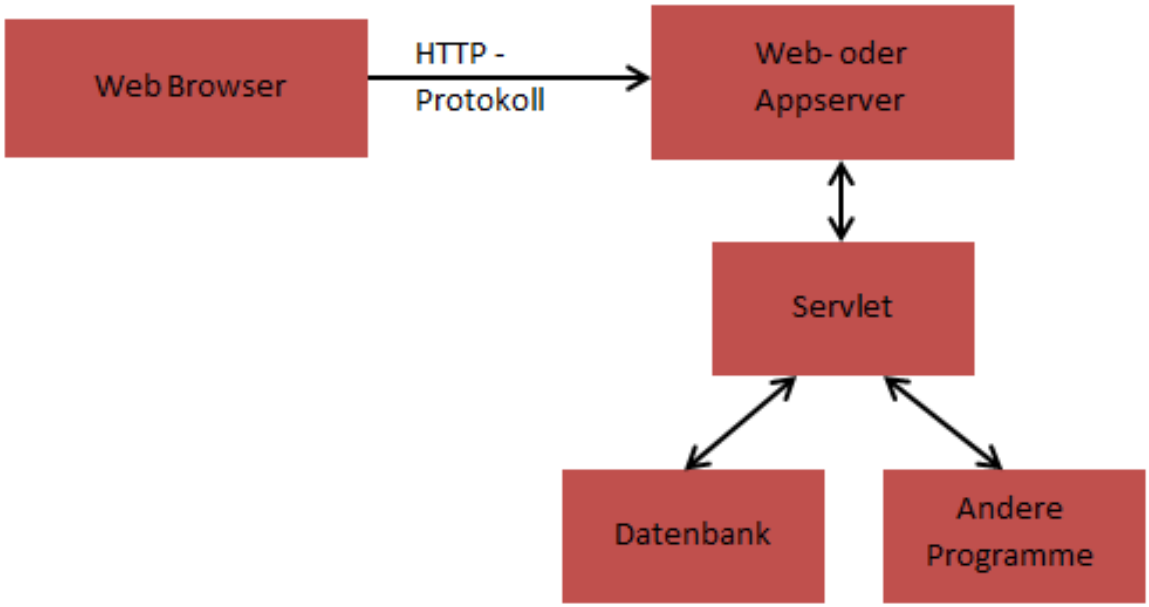


Abbildung 2.1: Kommunikationsdiagramm eines Servlets

Die Aufgabe eines Servlets ist das Sammeln der Eingabeinformationen von einem Benutzer (z.B. durch Formulare, Datenbankeinträge oder andere Quellen), um dynamisch eine Webseite zu erzeugen. Java Servlets dienen oft demselben Zweck wie Programme, welche das Common Gateway Interface (CGI) verwenden. Das CGI ist ein Standard für den Datenaustausch zwischen einem Webserver und dritter Software [44]. Servlets bieten jedoch mehrere Vorteile:

- Signifikant bessere Performance.
- Servlets arbeiten im Adressbereich des Webservers. Es ist somit nicht für jede Client-Anfrage ein eigener Prozess nötig. [33]
- Servlets sind durch den Java Security Manager (ermöglicht das durchlaufen von nicht vertraulichem Bytecode in einer abgeschotteten Umgebung, um somit Schäden zu verhindern) geschützt. [44]
- Die volle Funktionalität der Java Klassenbibliotheken steht dem Servlet zur Verfügung. [33]

Um das Servlet auf einem Webserver oder Appserver registrieren zu können, müssen Metainformationen bereitgestellt werden. Diese werden im Deployment-Descriptor (Konfigurationsdatei in XML-Form), genannt web.xml, hinterlegt. Der Deployment Descriptor wird zusammen mit der kompilierten Servlet-Klasse und, wenn vorhanden, anderen Klassen in einer Archiv-Datei, dem Web-Archiv (Dateiformat welches das Verpacken der vollständigen Web-Applikation nach Java-Servlet-Spezifikationen beschreibt), vereint [48]. Dieses Archiv wird dann dem Servlet-Container übergeben.

Zur Laufzeit greift der Web- oder Appserver dann auf diesen Servlet-Container zu. Dieser erzeugt eine Instanz des Servlets und startet benötigte Methoden [47].

2.1.9 Java Persistence API (JPA)

Die JPA ist ein Framework, bestehend aus einer Kollektion aus Klassen sowie Methoden, welches es ermöglicht, mit einer Datenbank zu interagieren und eine Verbindung zwischen Objekten im Java Programm und den Daten einer relationalen Datenbank zu formen.

Provider

Da die JPA eine Open-Source API ist, gibt es verschiedene Produkte von verschiedenen Herstellern, welche die JPA in ihren Produkten verwenden. Oracle (Toplink), Redhat (Hibernate, JBoss), Eclipse Foundation (Eclipselink), Spring (Spring Data JPA), Apache (OpenJPA) und DataNucleus (DataNucleus AccessPlatform) sind unter anderem Hersteller der in den Klammern angeführten Produkte. [31]

Architektur

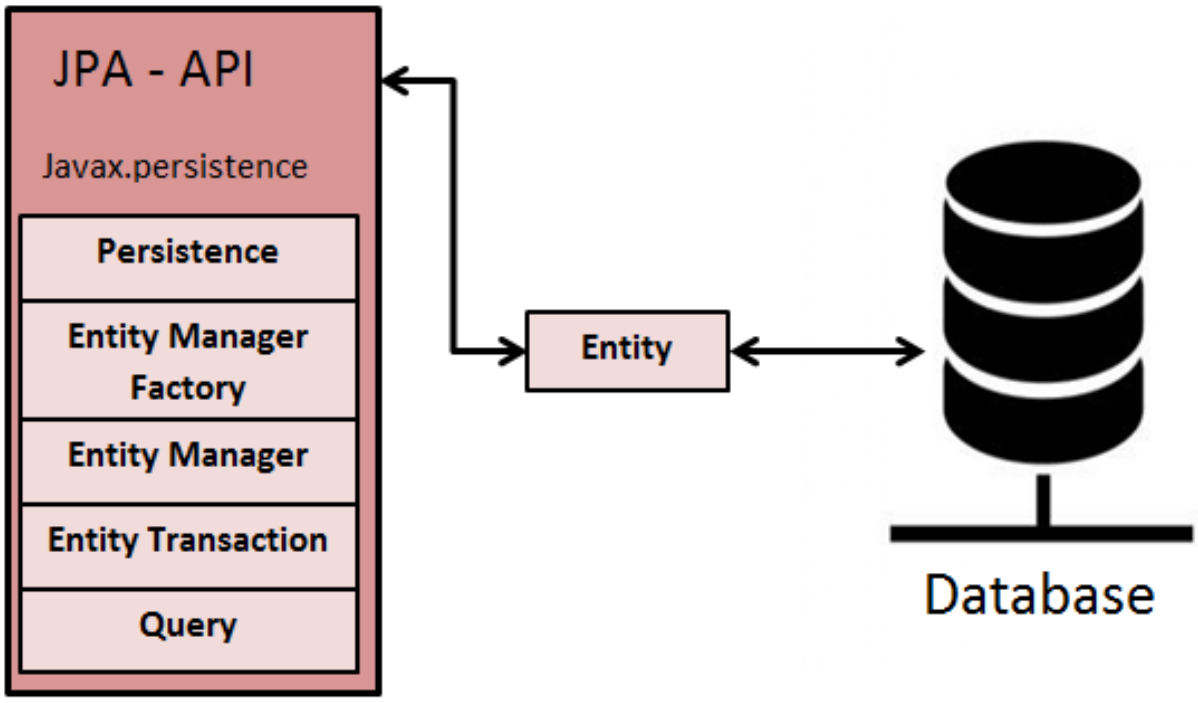


Abbildung 2.2: Java Persistence API Architektur

Abbildung 2.3 gibt einen Überblick über die Architektur der JPA und deren Komponenten.

Komponente	Erklärung
javax.persistence.Persistence	Klasse, welche statische Methoden besitzt, die dazu dienen eine EntityManagerFactory zu erzeugen.
javax.persistence.EntityManagerFactory	Die Fabrik erzeugt und verwaltet mehrere EntityManager.
javax.persistence.EntityManager	Interface, welches die Operationen auf Objekte verwaltet.
javax.persistence.Entity	Entitäten sind die Objekte, die im Programm verwaltet werden. Sie stellen eine Zeile an Daten in der Datenbank dar.
javax.persistence.EntityTransaction	Interface, welches dazu ausgelegt ist, eine Transaktion eines Entity Managers zu verwalten.
javax.persistence.Query	Interface, welches es ermöglicht, eine Datenbankabfrage zu erstellen und durchzuführen. [27]

Die oben angeführten Klassen erleichtern es dem Programmierer, Datenbankoperationen durchzuführen, da die JPA auf der ORM-Technologie basiert.

2.1.10 Object Relational Mapping (ORM)

ORM ordnet die Objekte einer Klasse eines Programmes den Daten einer relationalen Datenbank zu und kann dies auch in umgekehrter Reihenfolge durchführen. Eine relationale Datenbank steht dem Programm sozusagen als objektorientierte Datenbank zur Verfügung. Die Definition für die Zuordnungen wird über Metadatenannotationen für die Sprache Java sowie XML-Deskriptoren realisiert. [11, 32]

Annotationen erlauben das Einbinden von Metadaten in den Quelltext. Somit können Konfigurationen mit einem einfachen `"@"` durchgeführt werden. Ein Beispiel wäre `@Entity` über den Klassennamen zu schreiben, dies bedeutet, dass die Klasse eine Entität oder Tabelle ist. `@Table(name="MEDIKAMENT")` spezifiziert die Tabelle für die annotierte Entität. Die JPA verfügt noch über viele andere Annotationen. [23, 42]

Bsp 2.3: Annotation - Entity

```
1 @Entity
2 public class Medikament implements Serializable {
3         ...
4 }
```

Objekte einer objektorientieren Programmiersprache kapseln Daten und ihr Verhalten hinter einer Schnittstelle und haben eine eindeutige Identität, wobei relationale Daten-

banken ihre Daten in Form von Tabellen ablegen und auf einem mathematischen Konzept der relationalen Algebra basieren. Dieser Widerspruch wurde in den 1990er-Jahren auch als „Object-relational impedance mismatch“ bezeichnet. Um diesem Problem entgegen zu wirken, wurden verschiedene Lösungen, z.B. objektorientierte Datenbanken oder Embedded SQL (ESQL), welches eine Programmiersprache um relationale Konzepte erweitert, entwickelt. ORM löst das Problem ohne der Erweiterung einer Programmiersprache und bietet die Möglichkeit, weiter eine relationale Datenbank verwenden zu können. Ein Nachteil des ORM ist, dass die relationale Datenbank ihre Stärken und Fähigkeiten nicht voll ausnutzen kann. [32]

Architektur

Abbildung 2.3 zeigt, wie Daten in eine relationale Datenbank über ein ORM gespeichert werden.

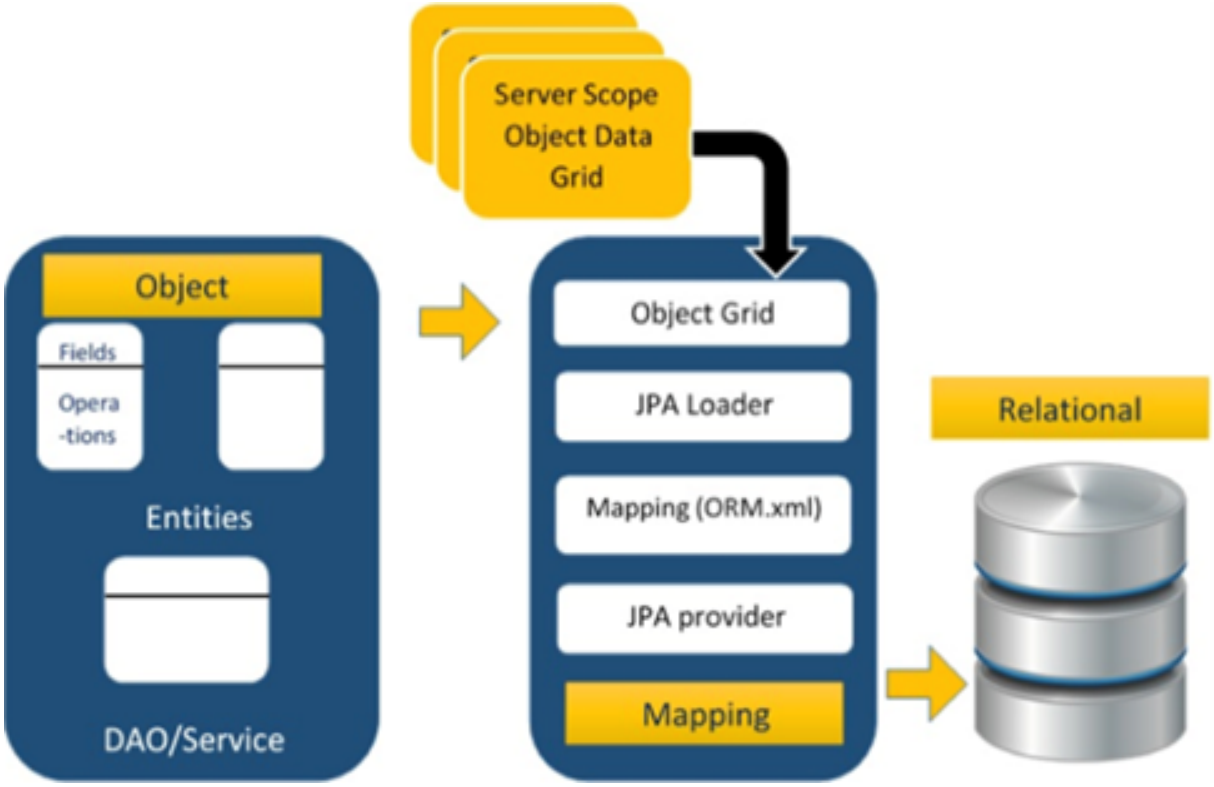


Abbildung 2.3: ORM Architektur

Die erste Phase beinhaltet POJO-Klassen, Service-Interfaces und Klassen. Somit stellt sie die Geschäftslogikschicht dar. Hier werden alle Operationen und Attribute, welche benötigt werden, implementiert.

POJOs (Plain Old Java Objects) sind ganz „normale“ Java Objekte, also alle Objekte, welche kein JavaBean, EntityBean oder dergleichen sind. Sie dienen auch keinem anderen

besonderen Zweck und implementieren weder ein Interface noch ein Java Framework. [35]

Beispiel: Die POJO-Klasse Mitarbeiter enthält die Attribute ID, Name und Abteilung sowie Getter- und Setter-Methoden zu diesen Attributen. Die Mitarbeiter DAO- (Data Access Objects)/Service-Klassen beinhalten Methoden, um Operationen durchführen zu können.

Die zweite Phase wird Zuordnungs- bzw. Persistenzschicht genannt. Diese beinhaltet den JPA-Provider, ein Mapping-File, den JPA-Loader und ein Object Grid.

Komponente	Erklärung
JPA-Provider	Wie oben schon beschrieben, ist dies das JPA Produkt, z.B. EclipseLink.
Mapping File	Dieses beinhaltet die Konfiguration zum Zuordnen der relationalen Daten zu den POJO-Klassen.
JPA-Loader	Der JPA-Loader arbeitet wie ein Cache-Speicher und ist für das Laden und Zwischenspeichern des Object Grid zuständig.
Object Grid	Dient als temporärer Speicher, welcher eine Kopie der Datenbank enthält. Alle Queries, welche sich an die Datenbank richten, werden zuerst auf dem Object Grid ausgeführt. Mit einem Commit werden die Änderungen dann in die eigentliche Datenbank übernommen.

Phase drei stellt die relationale Datenbank dar, welche logisch mit den Geschäftsprozessen des Programms verbunden ist. [32]

2.1.11 Google Cloud Messaging (GCM)

Google Cloud Messaging ist ein Service, mit dem Daten zwischen dem Server und einem Android-Gerät in beide Richtungen ausgetauscht werden können. Man unterscheidet dabei zwischen zwei Arten von Nachrichtenübermittlung, dem Downstream-Messaging (Nachricht vom Server an den Client) und dem Upstream-Messaging (Nachricht vom Client an den Server). GCM ist ein Teil der Google Play Services. [6]

Architektur von GCM

GCM verwaltet die Kommunikation zwischen drei Komponenten, dem App Server, dem Android-Gerät und dazwischen den von Google zur Verfügung gestellten GCM-Verbindungs-Servern.

Der Thrid-Party App Server wird vom Entwickler implementiert und muss mit dem Client kommunizieren können. Eine primäre Aufgabe des App Servers ist es, korrekt formatierte Anfragen an den GCM-Server zu übermitteln und, falls Upstream-Messaging unterstützt wird, hereinkommende Anfragen verarbeiten zu können.

Die GCM-Verbindungs-Server werden von Google zur Verfügung gestellt und sind über das Internet erreichbar. Sie empfangen eine GCM-Nachricht vom App Server, reihen diese zunächst in die Warteschlange ein und speichern sie temporär ab. Danach wird die Nachricht schnellstmöglich an den jeweiligen Client weitergeleitet. Sollte dieser zurzeit nicht erreichbar sein, da er beispielsweise offline ist, wird die Nachricht erst dann übermittelt, wenn die Verbindung wieder aufgebaut werden kann. Google bietet zurzeit zwei Verbindungs-Server mit unterschiedlichen Technologien an. Einen herkömmlichen HTTP-Server, an den Nachrichten mit dem HTTP-Protokoll übermittelt werden sowie den Cloud Connection Server (CCS), ein XMPP-Endpunkt, über welchen bidirektional Nachrichten übermittelt werden können.

Die Client-App läuft auf einem Android-Gerät und muss GCM aktiviert haben. Um GCM-Nachrichten empfangen zu können, muss sich die App einmalig mit dem Google-Service registrieren und erhält dadurch eine sogenannte Registration ID, mit welcher sie eindeutig identifiziert werden kann. Unterstützt der App Server Upstream-Messaging, so kann die Client-App auch eigene Nachrichten über GCM an den Server schicken. [6]

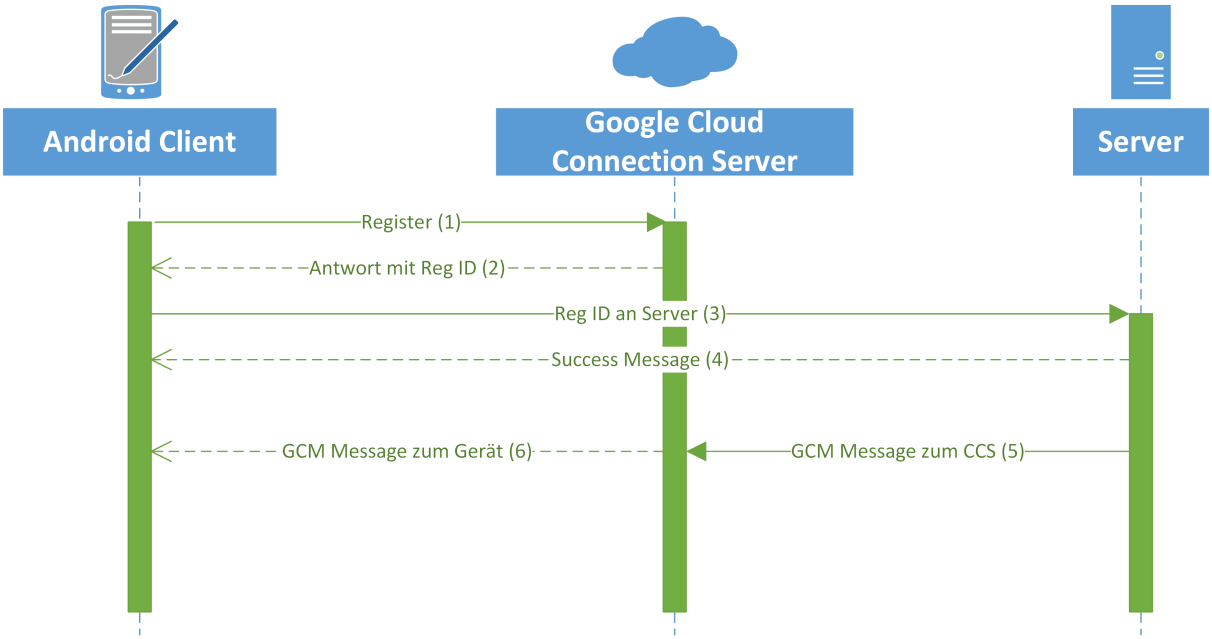


Abbildung 2.4: Ablauf der Kommunikation mit GCM als Sequenzdiagramm

In Abbildung 2.4 ist nun ersichtlich, wie die drei Komponenten bei GCM untereinander kommunizieren. Durch die ersten beiden Schritte wird die Registrierung abgewickelt. Zuerst wird der Google Cloud Messaging API mitgeteilt, dass ein Gerät registriert werden soll (Schritt 1), worauf der Google Server bei erfolgreichem Registrierungsprozess mit der Registration ID antwortet (Schritt 2). Die eben erhaltene Registration ID wird nun im Normalfall zurück zum App Server gesendet und dort persistiert (Schritt 3). Zur Übermittlung wird in der Regel das HTTP-Protokoll verwendet, worauf der Server mit einer Success-Message antworten sollte (Schritt 4). Nun, da die Verbindung aufgebaut ist, kann der App Server Nachrichten zum Google Connection Server schicken (Schritt 5), welcher diese dann zum Gerät mit der jeweiligen Registration ID weiterleitet (Schritt 6).

Authentisierung

Da der Google Cloud Messaging Service von vielen unterschiedlichen Anwendungen parallel genutzt wird, ist eine sichere Abgrenzung der Applikationen und Authentifizierung aller Parteien essentiell. GCM handhabt dies mit den folgenden Authentisierungswerkzeugen.

- Die Sender ID ist einer Applikation zugeordnet und wird bei der Erstellung eines neuen Projekts in der Google Developer Console generiert. Die Nummer wird bei der Registrierung einer App verwendet, um den dazugehörigen und autorisierten App Server zu finden.
- Der Sender Auth Token ist ein Schlüssel, der dem App Server die Berechtigung zur Verbindung mit den Google Connection Servers gibt. Er wird am App Server gespeichert und im Header von HTTP-POST-Anfragen mitgeschickt.
- Die Registration ID erhält die Client-App als Resultat der GCM-Registrierung. Diese ID wird in der Regel an den App Server übermittelt, damit dieser Nachrichten an den jeweiligen Empfänger schicken kann.
- Durch die Application ID wird sichergestellt, dass die Nachricht an die richtige App am Client-Gerät geleitet wird. Im Falle von Android ist die Application ID durch den Package-Namen im Android Manifest definiert. [6]

2.1.12 Extensible Messaging and Presence Protocol (XMPP)

XMPP ist ein freies Protokoll zur Echtzeit-Kommunikation über ein Netzwerk unter der Verwendung von XML-basierten Daten. Es wurde ursprünglich, zunächst unter dem Open-Source-Projekt Jabber, als Protokoll für Instant Messaging (dt. sofortige Nachrichtenübermittlung) entwickelt, wurde daraufhin schnell beliebt und im Jahr 2004 von der IETF (Internet Engineering Task Force) standardisiert. XMPP ist ein dezentrales Protokoll, d.h. die Architektur ist, ähnlich wie bei Email, so aufgebaut, dass jeder einen eigenen XMPP-Server hosten kann, anstatt einen zentralen Server zu verwenden. Durch die Verwendung von XML-strukturierten-Daten ist das Protokoll außerdem sehr einfach um zusätzliche Funktionalität erweiterbar. Durch diese Flexibilität hat sich XMPP in

den verschiedensten Anwendungsbereichen, von Chat-Programmen über Voice over IP (Internettelefonie) bis hin zu Spielen, etabliert. [7]

2.1.13 GCM Cloud Connection Server

Der GCM Cloud Connection Server (CCS) ist ein XMPP-Endpunkt, über den eine bidirektionale Verbindung (Upstream- und Downstream-Messages) zu den Google Servern aufgebaut wird. Gegenüber den Einsatz eines gewöhnlichen HTTP-Servers bietet der CCS, neben der Möglichkeit zur Datenübermittlung in beide Richtungen, einige Vorteile in der Performance, da mit dem asynchronen XMPP-Protokoll mehr Nachrichten mit vergleichsweise weniger Ressourcenbedarf versendet werden können. [6]

Verbindungsaufbau

Um GCM-Messages senden und empfangen zu können, muss zunächst eine Verbindung mit dem CCS aufgebaut werden. Dies wird in der Regel im eigenen GCM App Server erledigt. Der Server, über den die Kommunikation stattfindet, ist unter der Adresse `gcm.googleapis.com:5235` erreichbar. Nachdem die Verbindung stattgefunden hat, verlangt der Server eine Authentifizierung. Hierbei müssen der Benutzername, welcher sich aus der Sender ID und der Serveradresse zusammensetzt, und als Passwort der Sender Auth Token (API Key) angegeben werden. Nach dem erfolgreichen Login, ist der App Server bereit, GCM-Nachrichten zu versenden und zu empfangen. [6]

Aufbau von Paketen

Zur Übermittlung verwendet der CCS sogenannte XMPP-Stanzas, kleine Pakete im XML-Format, welche die GCM-Nachrichten beinhalten. Die GCM-Nachrichten selbst sind im JSON-Format aufgebaut. XMPP-Downstream-Messages enthalten in jedem Fall die Felder `messageId`, eine eindeutige Zeichenkette zur Identifikation der Nachricht und `to`, die Registration ID des Empfängers. Zusätzlich kann im Feld `payload` ein JSON-Objekt mit bis zu vier Kilobyte Nutzdaten übergeben werden und daneben zahlreiche weitere Parameter angegeben werden.

Die fertige GCM-Nachricht wird in dem XMPP-Paket gekapselt.

Bsp 2.4: Ein XMPP-Stanza mit der GCM Message

```
<message>
  <gcm xmlns="google:mobile:data">
  {
      "to":"REGISTRATION_ID",
      "message_id":"m-1366082849205"
      "data":
      {
          "hello":"world",
      }
```

```
10    }
11    </gcm>
12  </message>
```

Bsp 2.4 zeigt den beispielhaften Aufbau eines solchen XMPP-Pakets. Das Wurzelelement `<message>` enthält das Element `<gcm>`, bei welchem der Namespace vom GCM-Dienst angegeben wird. Dieses enthält die eigentliche GCM-Nachricht im JSON-Format. [6]

2.2 Entwicklungssysteme

2.2.1 Eclipse IDE

Die Eclipse IDE (Integrated Development Environment, dt. integrierte Entwicklungsumgebung) ist ein freies Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Ursprünglich wurde Eclipse als IDE für die Programmiersprache Java entwickelt, inzwischen wird es aufgrund der Erweiterbarkeit für viele andere Entwicklungsaufgaben eingesetzt. Eclipse basiert auf Java-Technik und seit Version 3.0 auf dem OSGi-Framework Equinox. [38]

2.2.2 Debian

Debian ist eine GNU/Linux- und GNU/kFreeBSD-Distribution. Eine Distribution hat zwei Hauptaufgaben, ein freies Betriebssystem auf einem Computer zu installieren und eine Fülle an Software anzubieten, welche alle Benutzeransprüche abdeckt. Debian basiert auf einem Linux Kernel. Ein Kernel ist zentraler Bestandteil eines Betriebssystems, in welchem die Prozess- und Datenorganisation, auf der die Softwarebestandteile aufbauen, festgelegt ist [46].
Der Entwickler Ian Murdock hatte klare Ziele. Qualität, erzielt durch eine Entwicklung mit größter Sorgfalt, um dem Linux Kernel würdig zu sein. Das zweite Ziel war eine nichtkommerzielle Distribution zu schaffen, welche mit erfolgreichen kommerziellen Distributionen mithalten kann. Dieser Aspekt konnte laut Murdock nur durch die Offenlegung der Entwicklung wie bei Linux und GNU Projekten erreicht werden. GNU Projekte sind eine Reihe von freier Software, entwickelt oder gesponsert von der Free Software Foundation (FSF). Debian hat heutzutage schon eine enorme Größe erreicht. Die 13 vorhandenen Architekturen bestehen aus elf Hardware-Architekturen und zwei Kernels (Linux und FreeBSD). Die verfügbare Software zählt 17.300 Source-Packages, welche nahezu jeden Bedarf abdecken. [10]

2.2.3 MySQL

MySQL ist ein relationales Datenbankverwaltungssystem. Es ist als Open-Source Software sowie als kommerzielle Enterprise-Version für verschiedene Betriebssysteme (unter

anderem Unix-Varianten, Mac OS X, Linux, Microsoft Windows, IBM OS/2, IBM i5/OS und seit 2008 Symbian) verfügbar. Das bevorzugte Einsatzgebiet von MySQL ist die Datenspeicherung für Webservices. Große Plattformen wie YouTube, Google, Facebook und Twitter verwenden MySQL zur Zugriffsabwicklung. MySQL und deren offizielle Bibliotheken sind darauf aus, möglichst hohe Performance zu erzielen, weshalb diese auch hauptsächlich in C und C++ implementiert sind.

Struktur

Es wird vorgesehen, dass ein MySQL-Server vorhanden ist, auf dem die Daten gespeichert sind. Mehrere MySQL-Clients können Anfragen an diesen Server schicken. Auf dem Datenbankmanagementsystem (MySQL-Server) können mehrere Datenbanken erstellt werden und in diesem wiederum mehrere Tabellen. MySQL erstellt für jede Datenbank ein Verzeichnis auf der Festplatte, in dem die Dateien über die Struktur und die Daten der einzelnen Tabellen abgelegt sind. Tabellen können jeweils von einem anderen Typ sein. Dieser Typ legt fest, welches Speichersubsystem für die Anfragen verwendet wird.

Speichersubsysteme

Es werden verschiedene Speichersubsysteme, auch Engines genannt, angeboten. Jede Engine ist für ein spezielles Szenario an Anfragen optimiert. Neben offiziellen mitgelieferten Engines gibt es auch welche von anderen Herstellern. Zwei Beispiele für offizielle Engines sind:

- MyISAM:
  Bietet schnellen Zugriff auf Tabellen und Indizes ohne Transaktionssicherung.
- InnoDB:
  Bietet transaktionssichere Lese- und Schreibzugriffe. Es werden somit Begin-, Commit- und Rollback-Funktionen bereitgestellt.

Anfragenverarbeitung

Wenn ein Client eine Anfrage an den Server stellt, ist dieser dafür verantwortlich, jede Anfrage möglichst performant zu bearbeiten. Der Server bearbeitet die Anfrage in vier Stufen:

1. Query-Cache:
   Ergebnisse von Anfragen werden in einem Zwischenspeicher (Query-Cache) abgelegt. Wenn später eine identische Anfrage kommt, ohne dass sich die Daten in der Zwischenzeit verändert haben, kann diese aus dem Cache beantwortet werden, wodurch ein Datenbankzugriff erspart wird. [39]
2. Parsing:
   Es wird überprüft, ob die Syntax der Query korrekt ist. Dazu wird diese in einzelne Komponenten zerlegt und es werden Informationen über Art der Query (SELECT,

INSERT usw.), betroffene Tabellen oder Inhalte einer WHERE-Klausel gesammelt. [39]

3. Optimierung:
   Der Optimizer ist ein Teil eines Datenbankmanagementsystems, welcher versucht, für die Anfrage die optimale Beschreibung zur Abarbeitung zu berechnen. Dieser hat die Aufgabe, die Anzahl der zu lesenden Datensätze so gering wie möglich zu halten. Der Optimizer berücksichtigt beispielsweise, ob es sinnvoll ist, einen Index zur Lokalisierung zu verwenden. [39, 41]
4. Ausführung der Query [39]

2.2.4 MySQL Workbench

MySQL Workbench ist ein visuelles Werkzeug für Datenbankarchitekten, Entwickler und Datenbankadministratoren und bietet Funktionen zur Datenmodellierung und SQL-Entwicklung. Außerdem bietet MySQL Workbench Verwaltungstools für Serverkonfiguration, Benutzerverwaltung, Sicherung und vieles mehr an. Darüber hinaus wird ein besserer Einblick in die Datenbanken zur Verfügung gestellt. Mit der MySQL Workbench können beliebige Arten von Datenbanken visuell entworfen, modelliert, erstellt und verwaltet werden. Alle wichtigen Funktionen wie Forward- und Reverse-Engineering oder Werkzeuge zum Erstellen, Ausführen und Optimieren von SQL-Abfragen sind verfügbar. [26]

2.2.5 phpMyAdmin

PhpMyAdmin ermöglicht es dem Benutzer, seine Datenbank über das Web zu verwalten. Funktionen wie das Verwalten von Datenbanken, Tabellen etc. werden über die grafische Oberfläche ermöglicht, während SQL-Statements auch direkt ausgeführt werden können. [28]

2.2.6 Apache Tomcat

Apache Tomcat ist ein Application Server. Ein Application Server (dt. Anwendungsserver) ist ein Server, der die Fähigkeit hat, Anwendungsprogramme auszuführen. Er stellt spezielle Dienste wie Transaktionen, Authentifizierung oder den Zugriff auf gewisse Dienste wie Webservices oder Datenbanken über definierte Schnittstellen zur Verfügung. [40] Tomcat implementiert die Spezifikation für Java Servlets und Java Server Pages (JSP) und erlaubt es, Applikationen auf Servlet- oder JSP-Basis auszuführen. JSP ist eine Web-Programmiersprache welche auf JHTML (Java HTML) basiert und einfache dynamische Erzeugung von HTML- bzw. XML-Ausgaben eines Webservers ermöglicht [37]. Ein Webserver ist ein Server, welcher die Hauptaufgabe hat, Daten (HTML-Dokumente, Bilder oder dynamisch erzeugte Daten) an Clients (z.B. einem Webbrowser) zu übertragen.

Bestandteile von Apache Tomcat

- Servlet-Container „Catalina“
  Implementiert die Spezifikation für Java Servlets und Java Server Pages. [43]
- Connector „Coyote“
  Unterstützt das HTTP-Protokoll und erlaubt es somit Catalina (also dem Servlet oder dem JSP-Container), sich als einfachen Webserver zu verhalten, welcher lokale Daten als HTTP-Dokumente anbietet. Coyote horcht auf einen TCP-Port für eingehende Verbindungen und leitet die Anfrage an die Tomcat Engine weiter, um diese zu verarbeiten und eine Antwort zurückzuschicken. [43]
- JSP-Engine „Jasper“
  Jasper parst JSP-Dateien und kompiliert diese in Java Code in Form von Servlets. Bei Änderungen der JSP-Datei erkennt Jasper diese Änderung und kompiliert die Datei neu. [43]

2.2.7 Google Developer Console

Die Google Developer Console ist eine Web-Plattform zur Verwaltung von Projekten, welche Google Cloud Ressourcen bzw. Google Developer APIs verwenden. Ein Projekt steht für eine Applikation, an der gearbeitet wird, und enthält sämtliche Einstellungen, Zugangsdaten und Metadaten über diese. Die Arbeitsumgebung und die angelegten Projekte sind jeweils mit einem Google-Konto verknüpft.

Erstellen eines Projekts

Auf der Startseite der Konsole wird mit dem Button „Projekt erstellen“ ein Dialogfenster geöffnet. Hier wird der Name des Projekts sowie eine eindeutige ID, welche ausschließlich innerhalb der Developer Console verwendet wird, angegeben. Per Klick auf „Erstellen“ wird das Projekt erstellt und steht nun in der Übersicht zur Verfügung.

Aktivierung von APIs

Eine der Hauptfunktionen in der Developer Console ist das Aktivieren bzw. Deaktivieren von Google Developer APIs. Google bietet ein breites Sortiment an APIs und Diensten, primär für die Entwicklung mobiler Apps für Android, an. In der Navigationsleiste wird der Punkt „APIs“ ausgewählt, worauf eine Übersicht über das gesamte Angebot angezeigt wird. Nachdem eine konkrete API ausgewählt wurde, kann diese je nach Preis und Verfügbarkeit für das aktuelle Projekt aktiviert werden. Ist die API bereits aktiv, können außerdem Details über die Nutzung und das verfügbare Kontingent abgerufen werden.

API-Schlüssel

Die Google-Dienste müssen eingehende Anfragen mit einem bestimmten Projekt verknüpfen, um den Datenverkehr und die Kontingente überwachen zu können. Dafür erfordert jeder Zugriff einen API-Schlüssel, über den eindeutig auf das Projekt geschlossen werden kann. Diese Schlüssel können in der Konsole automatisch erzeugt werden und sollten geheim gehalten werden. Über den Menüpunkt „Zugangsdaten“ wird eine Übersicht über die Schlüssel gegeben, wo auch ein neuer Schlüssel erstellt werden kann. Im Erstellungsdialog wird zunächst die Plattform ausgewählt, auf der die Applikation läuft. Im Falle eines Servers, wird im nächsten Schritt dessen IP-Adresse angegeben. Daraufhin kann der Schlüssel automatisch generiert werden.

Kapitel 3

Systemdokumentation

3.1 Anwendungsfälle (Use-Cases)

Im Folgenden werden sämtliche Anwendungsfälle (Use-Cases), die durch die Software abgedeckt werden, beschrieben. Das Use-Case-Diagramm in Abbildung 3.1 fasst diese übersichtlich zusammen, während in den Unterkapiteln die Anwendungsfälle aus Sicht des Betreuers sowie aus Sicht des Patienten näher erläutert und durch ein Aktivitätsdiagramm detailliert dargestellt werden.

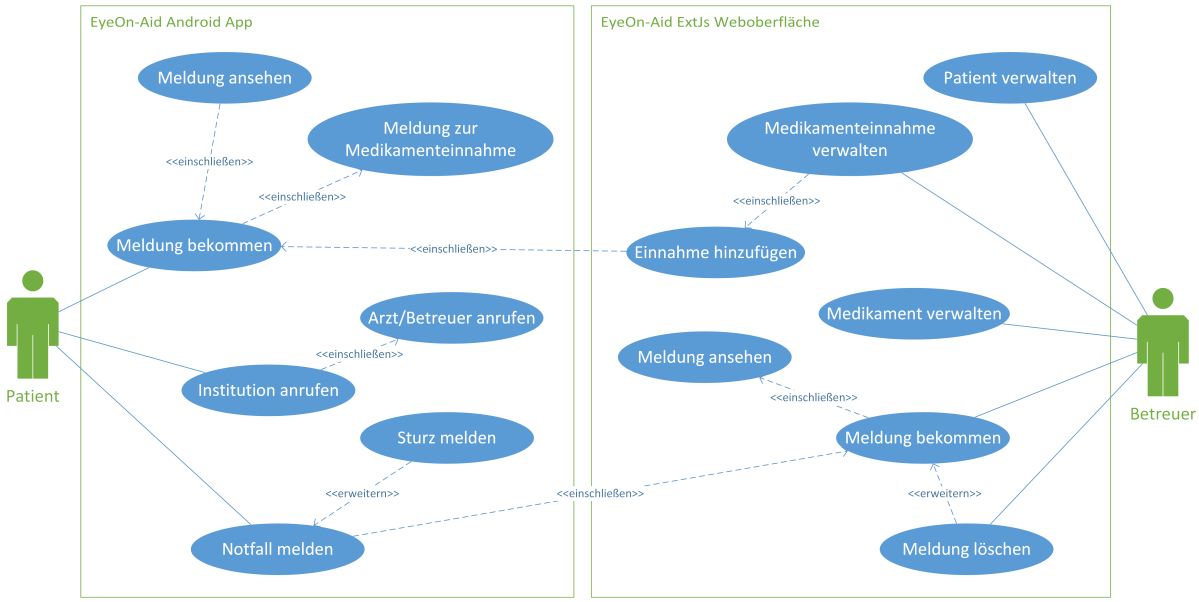


Abbildung 3.1: Use-Case-Diagramm (Anwendungsfalldiagramm)

3.1.1 Use-Case-Beschreibung - Betreuer

Im Folgenden werden die Anwendungsfälle aus Sicht des Betreuers näher erläutert.

Patient verwalten

Ein Patient kann hinzugefügt, bearbeitet und gelöscht werden.

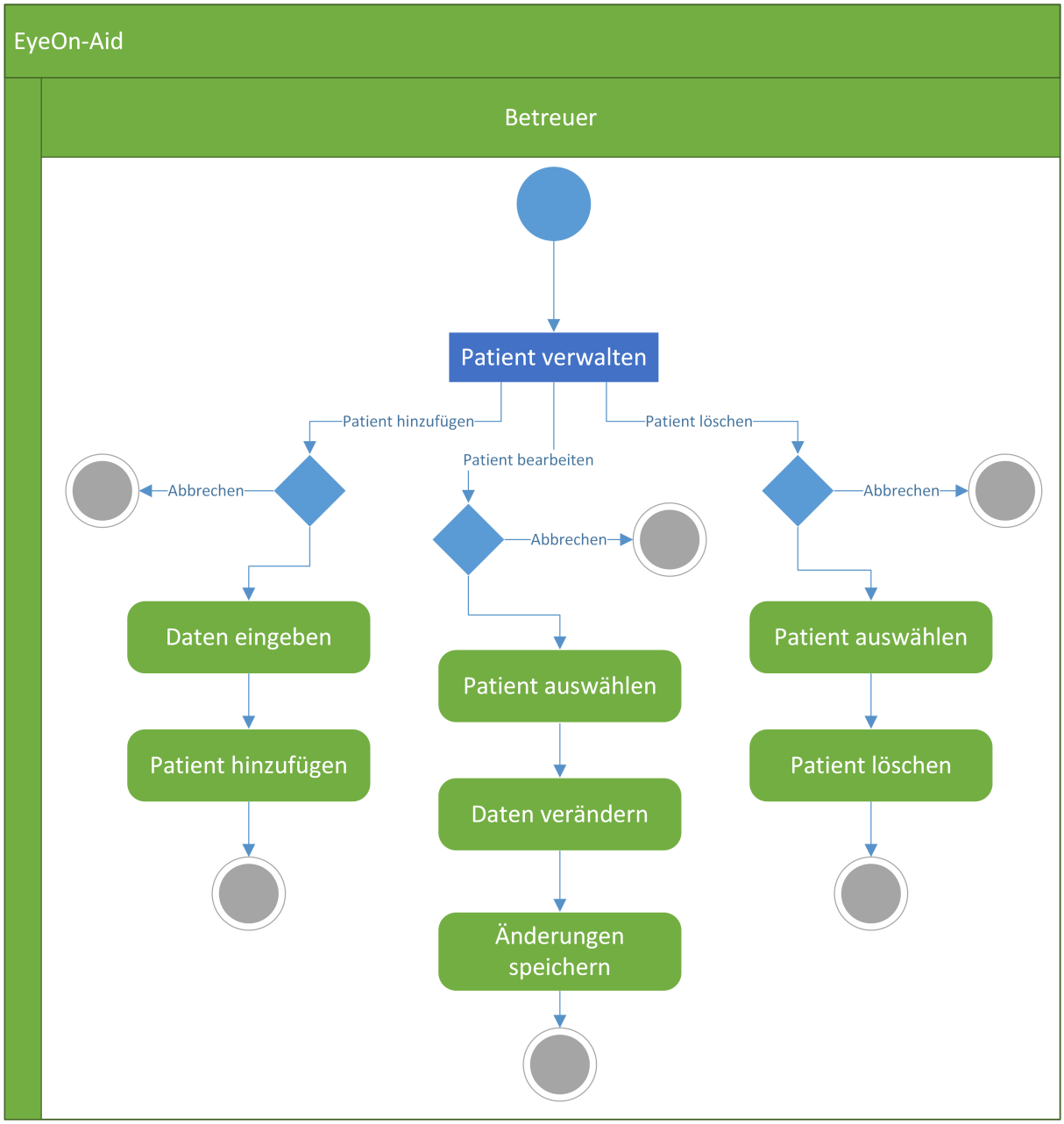


Abbildung 3.2: Aktivitätsdiagramm Patientenverwaltung

Medikamenteneinnahme verwalten

Eine Einnahme kann hinzugefügt, bearbeitet und gelöscht werden. Wenn der Betreuer eine Einnahme hinzufügt, wird beim Patienten zu einem gewissen Zeitpunkt eine Meldung erscheinen, die ihm mitteilt, ein Medikament einzunehmen. Diese Meldung muss innerhalb eines Zeitintervalls bestätigt werden, ansonsten wird der Betreuer benachrichtigt.

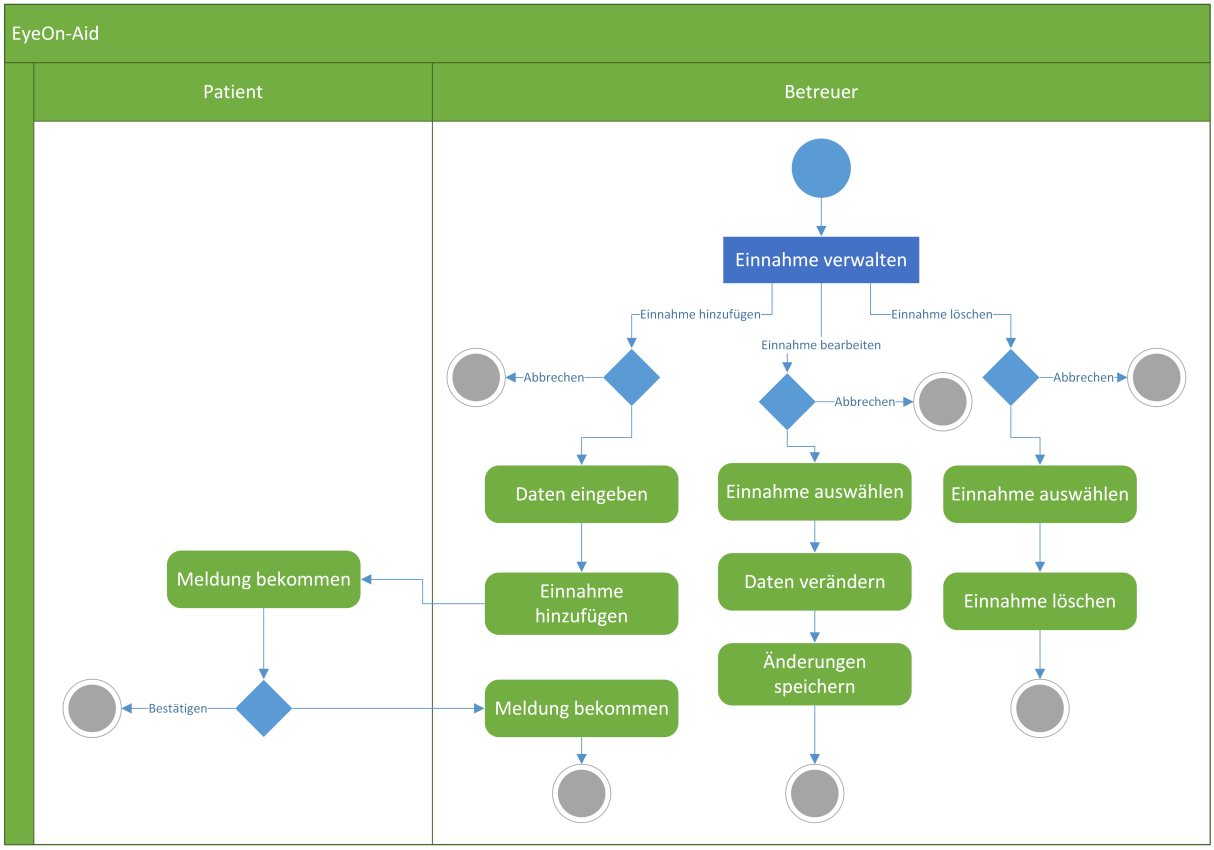


Abbildung 3.3: Aktivitätsdiagramm Einnahmeverwaltung

Medikament verwalten

Ein Medikament kann hinzugefügt, bearbeitet und gelöscht werden. Diese Medikamente stehen bei der Eintragung einer Einnahme zur Verfügung.

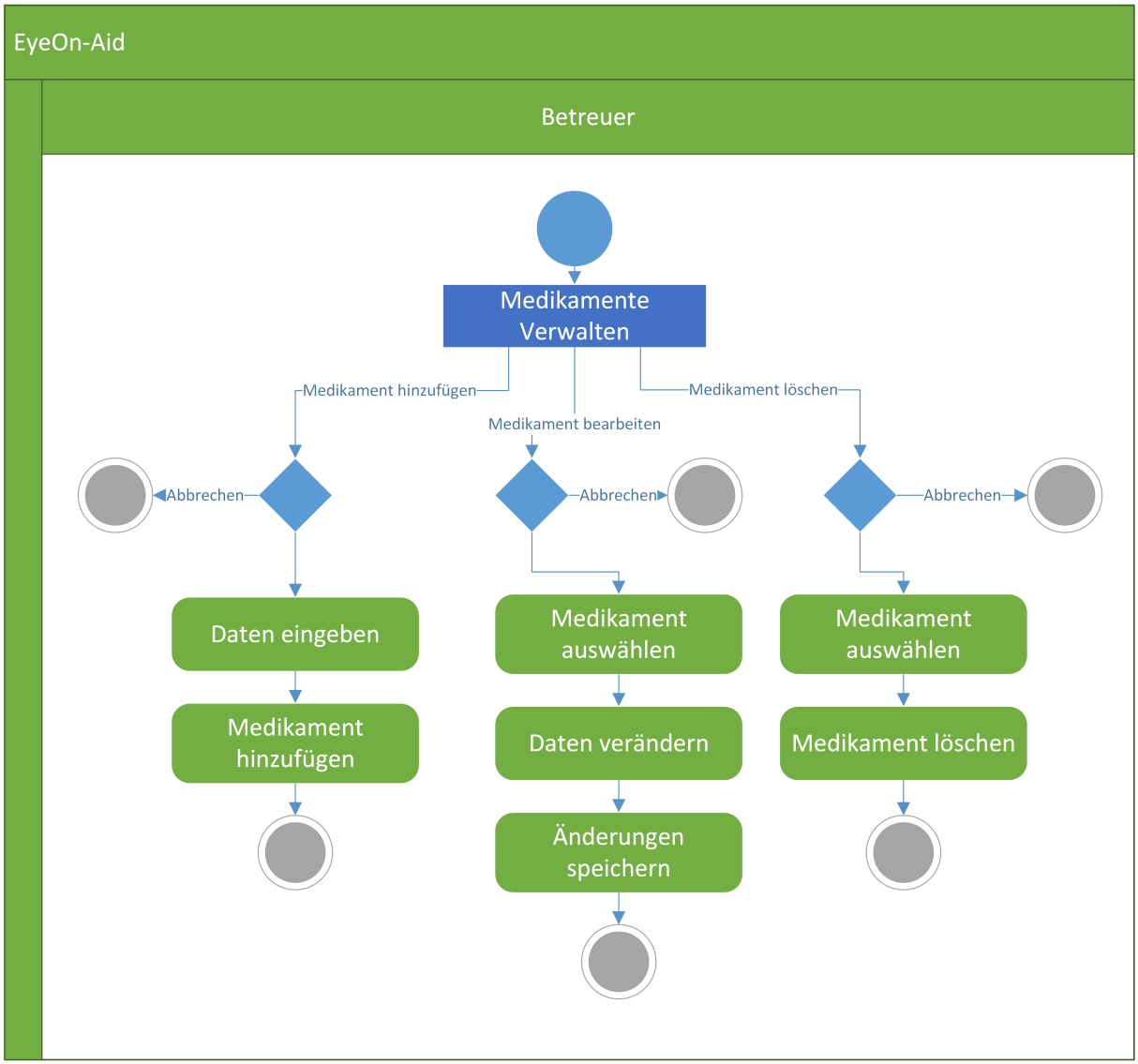


Abbildung 3.4: Aktivitätsdiagramm Medikamentenverwaltung

Meldung empfangen

Ein Betreuer empfängt Meldungen, wenn ein Patient eine Medikamenteneinnahme nicht bestätigt hat, gestürzt ist oder er einen Notfall meldet. Diese Meldungen kann der Betreuer ansehen und auch löschen.

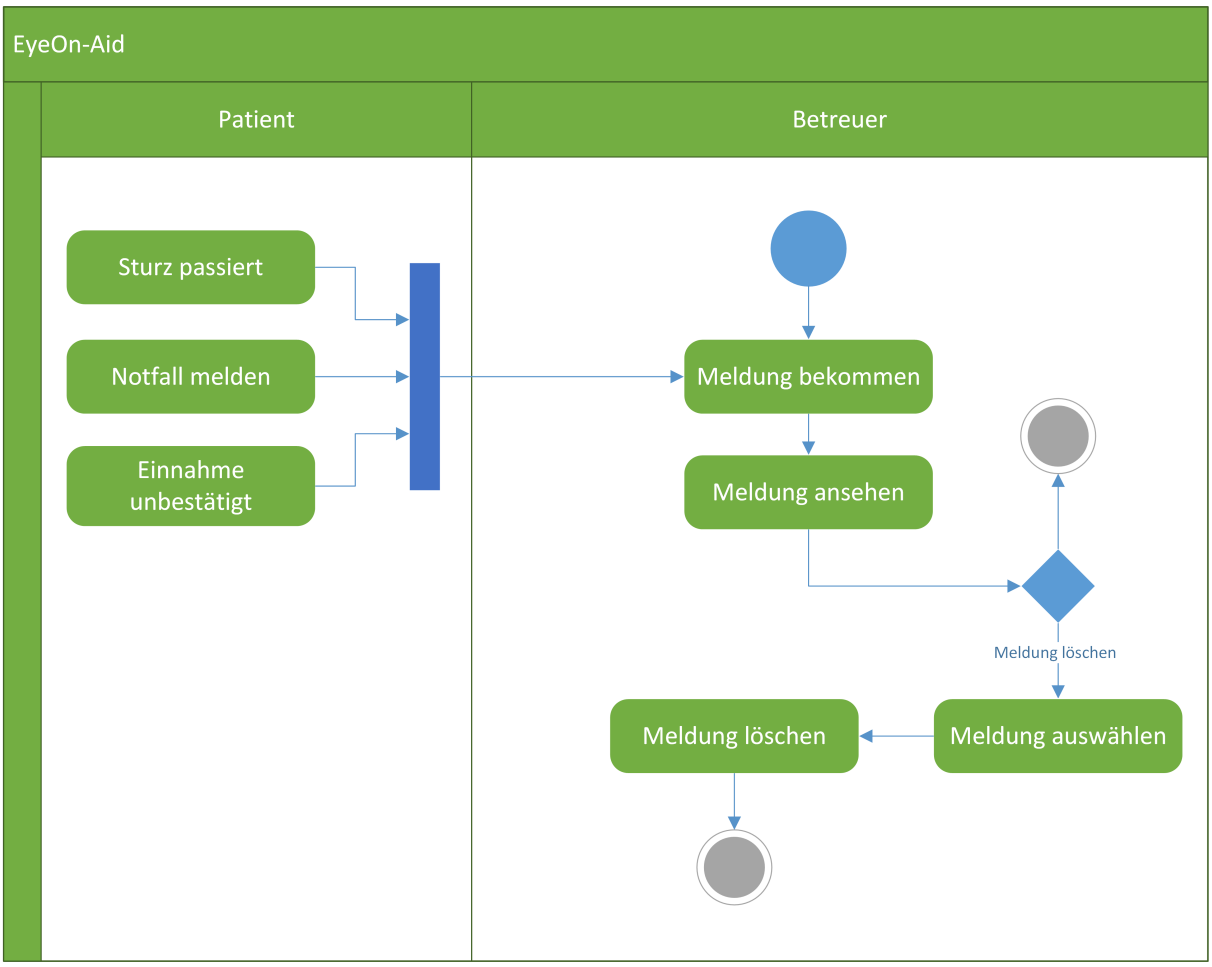


Abbildung 3.5: Aktivitätsdiagramm Meldungen - Betreuer

3.1.2 Use-Case-Beschreibung - Patient

Im Folgenden werden die Anwendungsfälle aus Sicht des Patienten näher erläutert.

Institution anrufen

Der Patient hat die Möglichkeit, über eine einfach gestaltete Oberfläche die wichtigsten Institutionen (Polizei, Rettung und Feuerwehr sowie seinen Betreuer) mit nur einem Klick anzurufen.

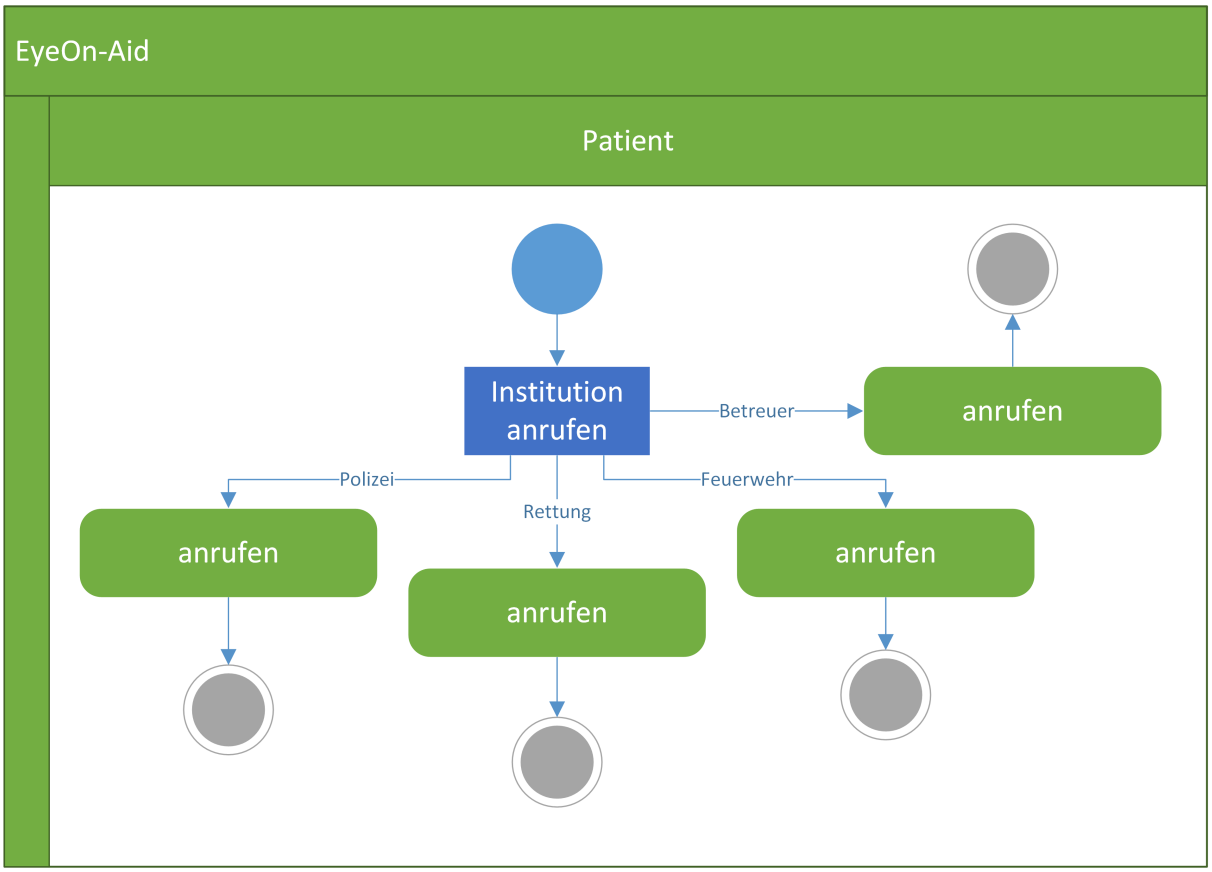


Abbildung 3.6: Aktivitätsdiagramm Institutionen anrufen

Meldung empfangen

Der Patient wird von seinem Betreuer durch eine Meldung über den Zeitpunkt der Medikamenteneinnahme informiert. Das Smartphone des Patienten beginnt dabei zu läuten und zeigt die jeweilige Meldung an. Wenn der Patient die Meldung in einem bestimmten Zeitintervall nicht bestätigt, wird sein Betreuer darüber informiert.

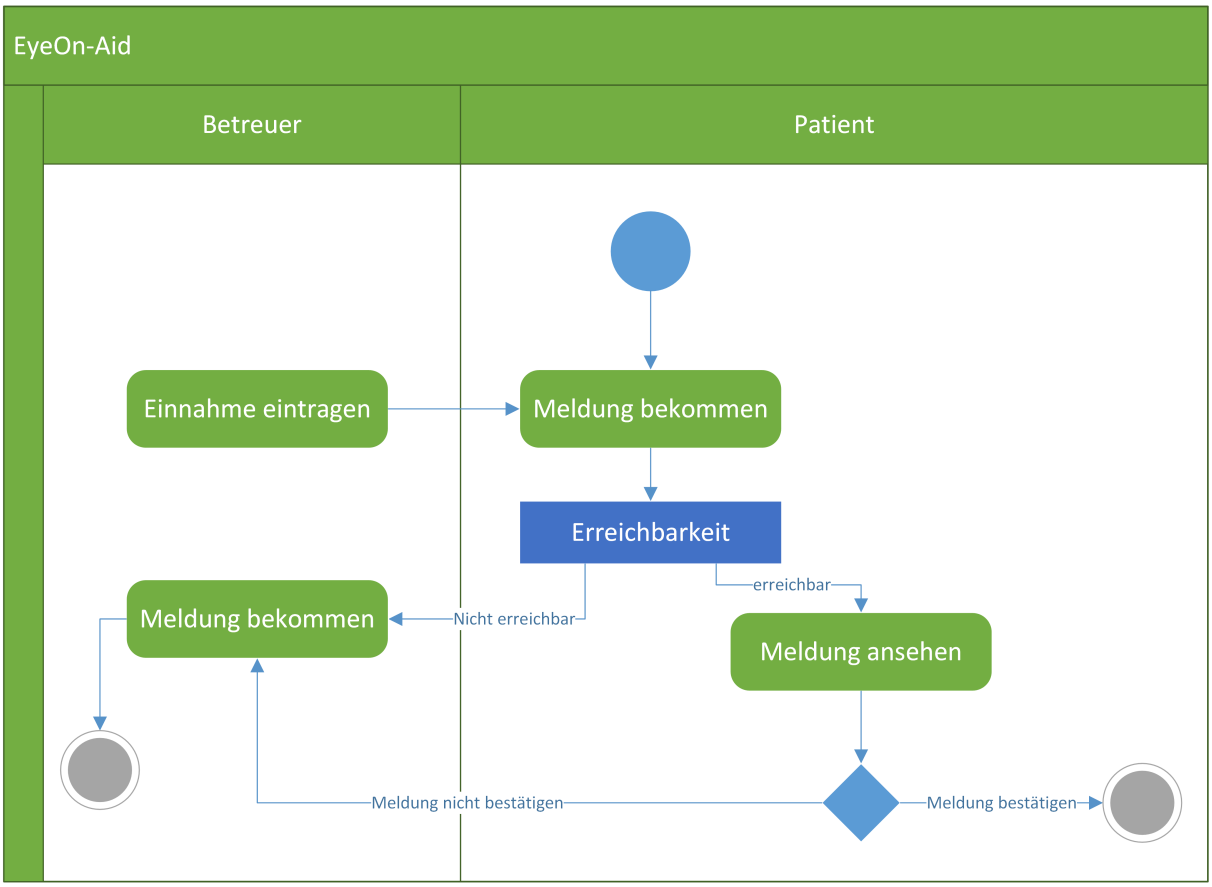


Abbildung 3.7: Aktivitätsdiagramm Meldungen - Patient

Notfall melden

Eine Notfallmeldung kann vom Patienten durch einen Klick auf den Notfall-Button erfolgen und scheint dann bei seinem Betreuer auf. Bei einem Sturz wird automatisch eine Meldung an den Betreuer gesendet.

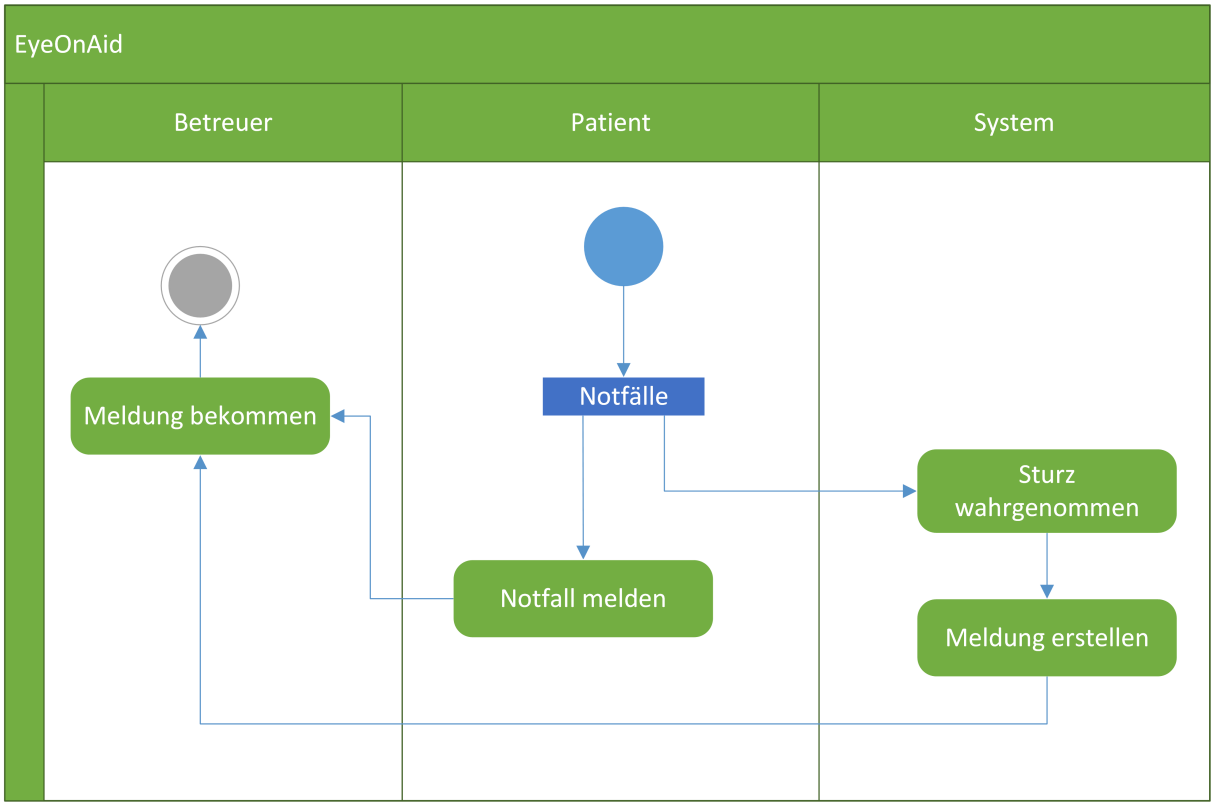


Abbildung 3.8: Aktivitätsdiagramm Notfallmeldung

3.2 Datenmodell

Das Datenmodell wurde in der MySQL Workbench erstellt und besteht aus acht Tabellen sowie zwei assoziative Tabellen. Das Modell zeigt die jeweiligen Tabellen und deren Attribute. Die Tabellen werden in den Unterkapiteln 3.2.1 - 3.2.8 näher erläutert.

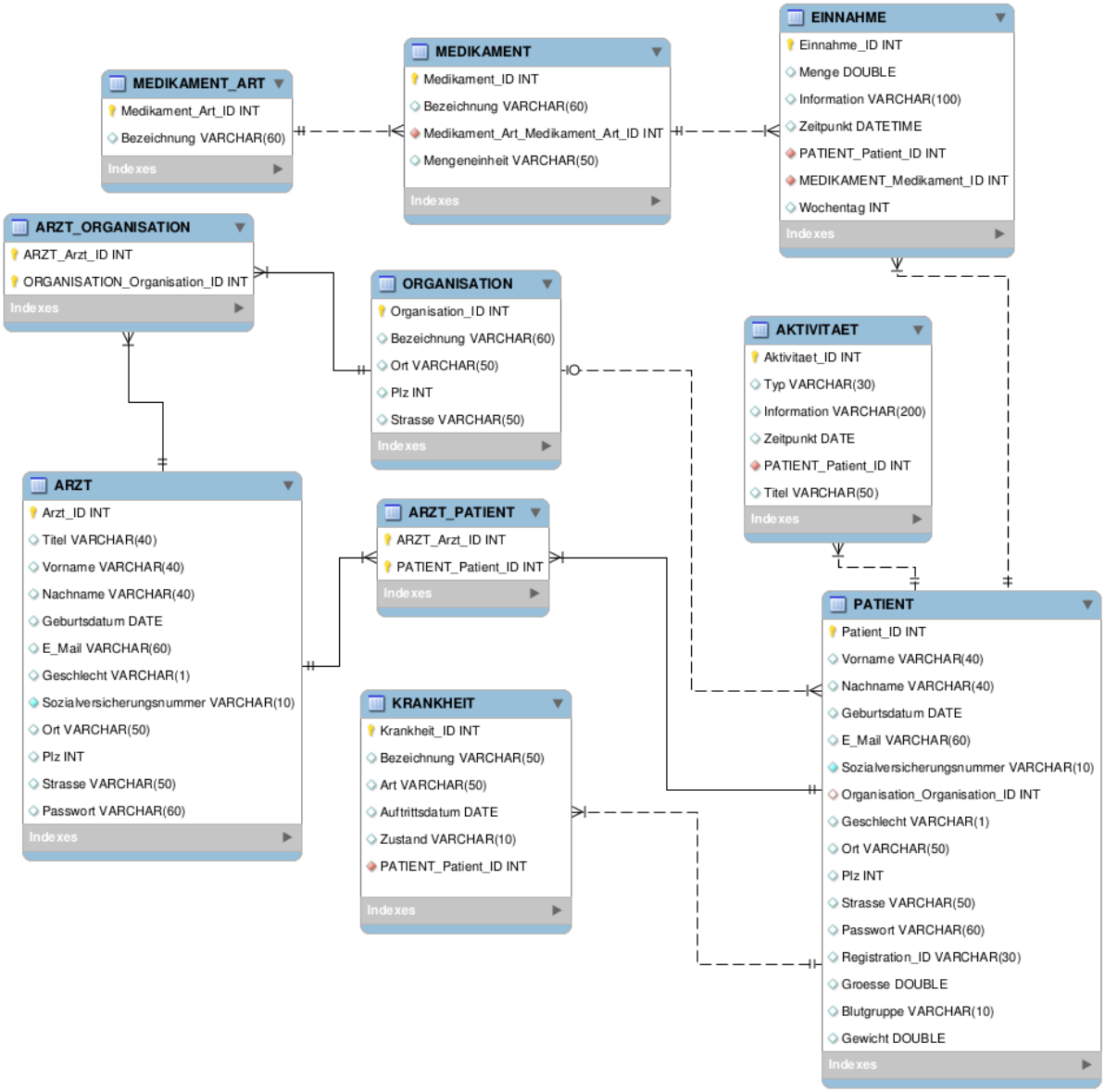


Abbildung 3.9: Datenbankmodell

3.2.1 Arzt

Die Tabelle Arzt speichert Stammdaten wie Name, Adresse, Sozialversicherungsnummer etc. des Betreuers. Die E-Mail-Adresse und das Passwort werden für den Login verwendet. Viele Ärzte bzw. Betreuer haben viele Patienten. Diese n:m-Beziehung wird durch

eine assoziative Tabelle ARZT_PATIENT aufgelöst. Außerdem können viele Ärzte bzw. Betreuer zu vielen Krankenhäusern bzw. Organisationen gehören. Diese n:m-Beziehung wird durch eine assoziative Tabelle ARZT_ORGANISATION aufgelöst.

3.2.2 Patient

Die Tabelle Patient speichert die Stammdaten des Patienten, welche Name, Adresse, Sozialversicherungsnummer etc. beinhalten. Auch körperliche Daten wie Blutgruppe, Gewicht und Größe werden gespeichert. Das Attribut `Registration_ID` (verwendet für den Google Cloud Messaging Service) wird bei erstmaliger Registrierung gesetzt und danach im Normalfall nicht mehr verändert. Ein Patient kann auch mehrere Aktivitäten und Einnahmen zu verschiedenen Zeitpunkten haben. Viele Patienten haben viele Ärzte bzw. Betreuer. Diese n:m-Beziehung wird durch eine assoziative Tabelle ARZT_PATIENT aufgelöst.

3.2.3 Medikament

Medikamente haben eine Bezeichnung, also der Name des Medikaments, eine Art (z.B. Tabletten, Tropfen, Ampullen etc.) und die Einheit in welcher die Einnahmemenge angegeben wird.

3.2.4 MedikamentArt

Die MedikamentArt beschreibt die Art des Medikamentes. Mehrere Medikamente haben eine MedikamentArt.

3.2.5 Einnahme

Die Tabelle Einnahme enthält alle Informationen die für eine Medikamenteneinnahme benötigt werden. Sie besitzt Attribute wie Menge, Information zur Einnahme, Zeitpunkt etc. Die Einnahme hat eine ID des Patienten, um genau zugeordnet werden zu können, da ein Patient mehrere Einnahmen haben kann. Außerdem ist auch eine Medikament_ID vorhanden, um einsehen zu können, welches Medikament eingenommen werden soll.

3.2.6 Organisation

Eine Organisation stellt eine Firma dar. Mehrere Ärzte bzw. Betreuer können in mehreren Organisationen arbeiten. Diese n:m-Bezeichung wird mit der Tabelle ARZT_ORGANISATION aufgelöst.

3.2.7 Aktivitaet

Die Tabelle Aktivität speichert alle Meldungen und Notrufe der Patienten. Sobald ein Medikament eingenommen werden soll oder der Patient einen Notruf auslöst, wird eine Meldung erzeugt und in dieser Tabelle gespeichert.

3.2.8 Krankheit

Eine Krankheit gibt Auskunft über den Gesundheitszustand des Patienten. Die Tabelle speichert die Bezeichnung und Art der Krankheit sowie dessen Auftrittsdatum. Das Attribut `zustand` gibt den aktuellen Zustand der Krankheit an. Wenn die Krankheit schon geheilt wurde, ist dieser Wert 0, wenn sie noch aktuell ist, ist der Wert 1. Ein Patient kann mehrere Krankheiten haben, darum gibt es auch eine n:1-Beziehung zwischen Krankheit und Patient.

Kapitel 4

Produktdokumentation

4.1 Web-Applikation für Betreuer

4.1.1 Registrierung

Bevor die Applikation verwendet werden kann, muss zunächst ein neuer Benutzer angelegt werden. Auf der Startseite wird das Registrierungsformular ausgefüllt, wobei Name, Email-Adresse und Passwort angegeben werden müssen. Nach dem die Daten abgesendet wurden, ist die Registrierung abgeschlossen und der Benutzer wird erzeugt.

4.1.2 Anmeldung

Beim Aufruf der Applikation muss sich der Benutzer zunächst über ein einfaches Anmeldeformular authentifizieren. Sobald die Daten abgesendet und am Server validiert wurden, wird der Benutzer automatisch zur eigentlichen Applikation weitergeleitet.

4.1.3 Patientenverwaltung

Im Bereich der Patientenverwaltung wird ein Überblick über alle Patienten des angemeldeten Betreuers gegeben. Hier besteht die Möglichkeit, neue Patienten hinzuzufügen und bestehende zu bearbeiten bzw. zu löschen. Mithilfe des Buttons „Neuer Patient“ wird ein Dialogfenster zur Eingabe der Stammdaten des Patienten geöffnet. Hier werden auch bereits die Zugangsdaten (Email und Passwort), mit denen sich der Patient später anmeldet, eingegeben. Per Doppelklick oder Klick auf den Button „Bearbeiten“ in der Zeile eines bestehenden Patienten öffnet sich erneut das Dialogfenster mit der Detailansicht des Patienten. Hier können sämtliche Daten bearbeitet werden. Per Klick auf den Button „Speichern“ werden die Änderungen in der Datenbank übernommen. Per Klick auf den Button „Verwerfen“ wird der zuletzt gespeicherte Zustand wiederhergestellt. Ein Klick auf den „Entfernen“-Button in der Zeile eines bestehenden Patienten entfernt ihn aus der Liste, insofern die Sicherheitsabfrage bestätigt wurde.

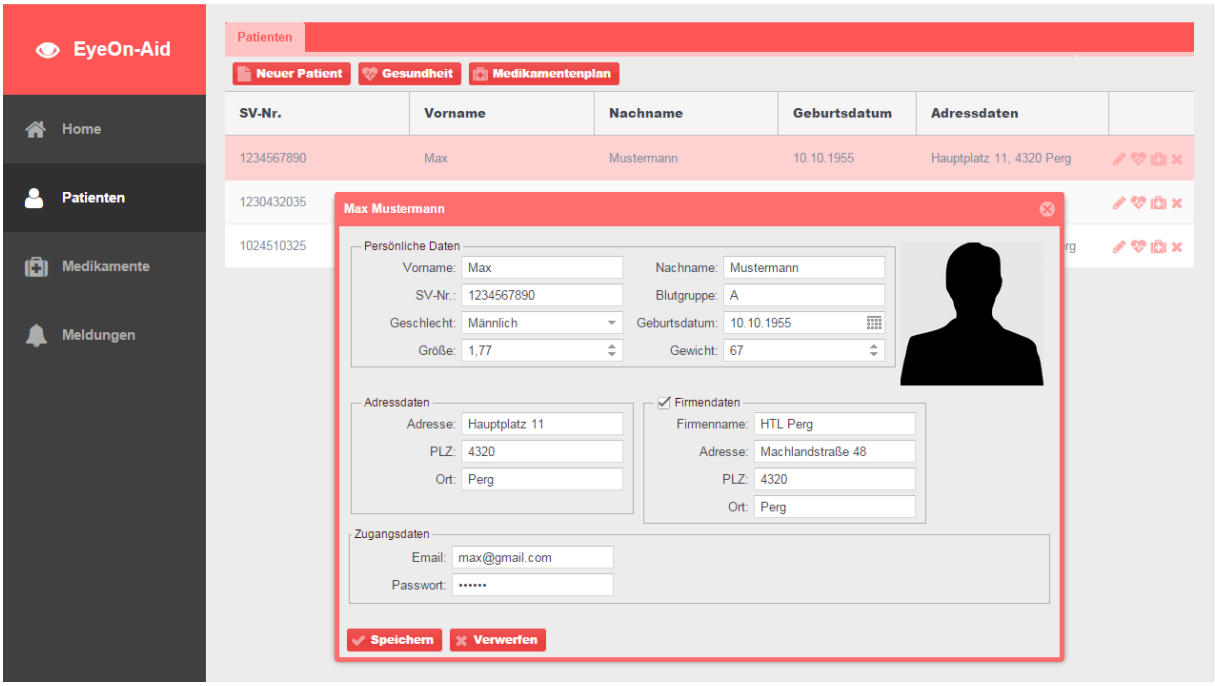


Abbildung 4.1: Die Patientenverwaltung mit Detailfenster für einen Patienten

Mit dem Button „Gesundheit“ und dem entsprechenden Symbol in der Zeile eines Patienten wird die Gesundheitsübersicht des Patienten in einem neuen Tab aufgerufen. In der oberen Tab-Leiste kann jederzeit zwischen geöffneten Ansichten gewechselt werden. In der Gesundheitsübersicht befindet sich neben der Ansicht einiger medizinischer Daten eine Auflistung der eingetragenen Krankheiten des Patienten. Mit dem Button „Neue Krankheit“ öffnet sich das Dialogfenster zum Hinzufügen neuer Krankheiten, wobei Bezeichnung, Art, Auftrittsdatum und aktueller Zustand der Krankheit festgelegt werden können. Ebenso können bestehende Krankheiten bearbeitet oder entfernt werden.

Zurück in der Patientenübersicht kann mit dem Button „Medikamentenplan“ oder dem gleichwertigen Symbol in der Schnellzugriffsspalte der Medikamentenplan des ausgewählten Patienten angezeigt werden. In dieser Ansicht werden sämtliche angeordnete Einnahmen für die einzelnen Tage der Woche aufgelistet. Per Klick auf „Neue Einnahme“ können neue Medikamenteneinnahmen für den Patienten festgelegt werden. Im Dialogfenster können der gewünschte Wochentag, die Uhrzeit, das einzunehmende Medikament und die Menge davon sowie optionale Textinformationen festgelegt werden. Nach dem Bestätigen wird der neue Eintrag je nach Wochentag im jeweiligen Bereich hinzugefügt. Das Bearbeiten und Löschen der Einträge funktioniert ebenfalls wie gewohnt.

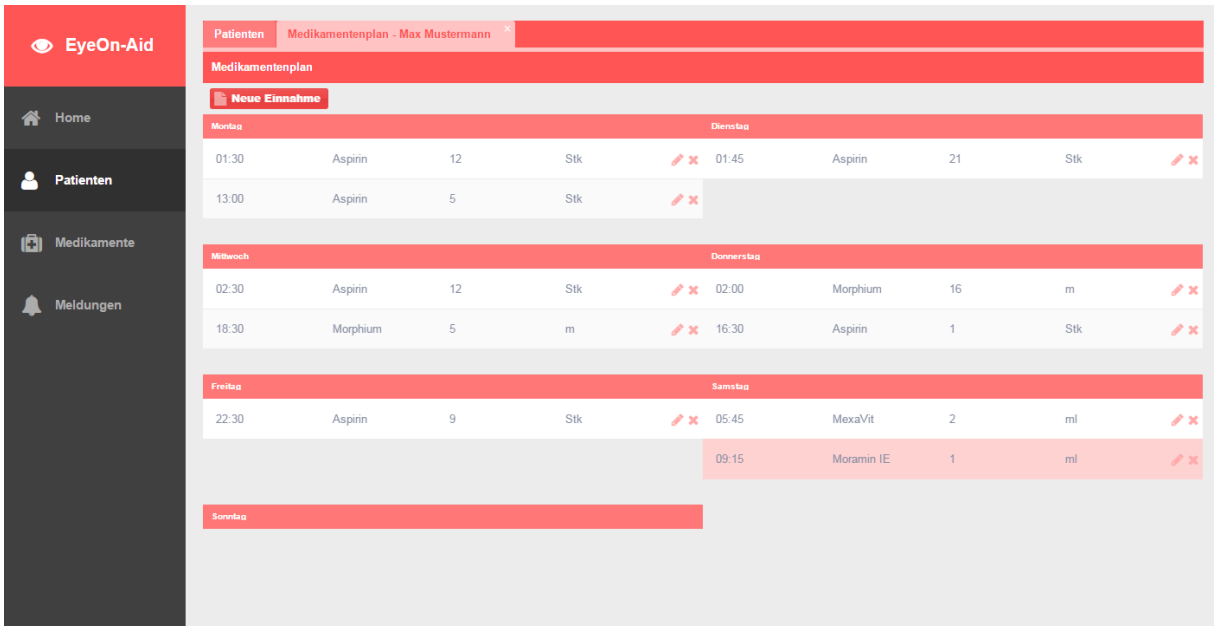


Abbildung 4.2: Der Medikamentenplan eines Patienten

4.1.4 Medikamente

In der Sektion „Medikamente“ werden alle Medikamente verwaltet, welche für die Einnahme der Patienten bestimmt sind. Wie gewohnt kann ein neues Medikament hinzugefügt werden. Im Dialogfenster sind die Bezeichnung, die Art sowie die verwendete Maßeinheit anzugeben. Wie üblich funktioniert auch wieder das Bearbeiten bzw. Löschen von Einträgen.

4.1.5 Meldungen

Im Bereich „Meldungen“ werden alle ausgelösten Ereignisse, die einen Patienten des angemeldeten Betreuers betreffen, aufgelistet. Hierbei werden drei Arten von Meldungen unterschieden: Nachrichten, Termine und Alarmierungen. In der Auswahlliste der Werkzeugleiste besteht die Möglichkeit, nach diesen Arten zu filtern. Die Aktualisierung der Meldungen erfolgt automatisch in einem Intervall von 15 Sekunden, kann aber auch manuell mit dem Button „Aktualisieren“ durchgeführt werden. Nach der Aktualisierung wird die Anzahl an neu eingetroffenen Meldungen in der Navigationsleiste gekennzeichnet und neue Einträge werden farblich hervorgehoben. Der Eintrag einer Meldung kann mittels dem „Entfernen“-Button von der Liste entfernt werden.

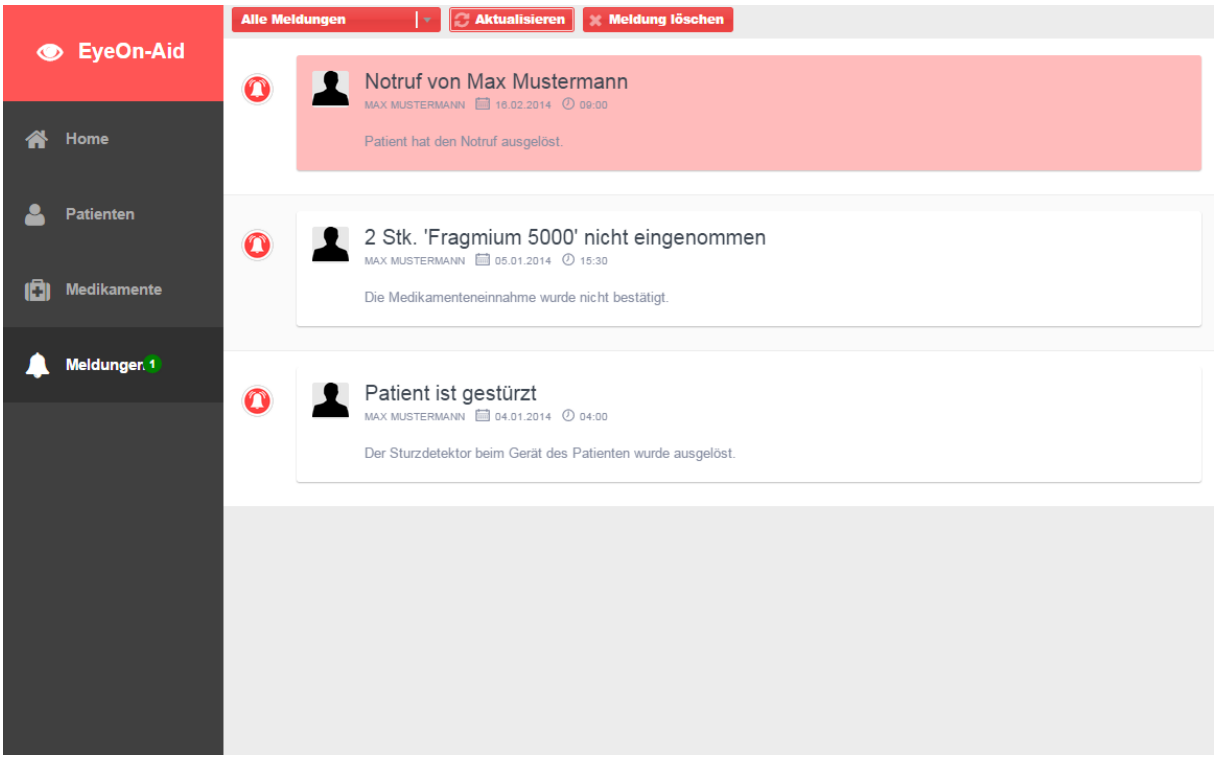


Abbildung 4.3: Die Meldungsübersicht mit einer neu eingegangenen Meldung

4.2 Android-App für Patienten

4.2.1 Anmeldung

Beim erstmaligen Start der App werden zunächst die Login-Daten, sprich E-Mail und Passwort, abgefragt. Hier sind jene Daten einzugeben, die bei der Erstellung eines neuen Patienten in der Betreuer-Applikation angegeben wurden. Ist die Authentifizierung erfolgreich, wird zur Hauptansicht der App weitergeleitet, ansonsten können die Daten neu eingegeben werden. Nach einem erfolgreichen Login werden die Anmeldedaten am Gerät gespeichert und beim nächsten Start der App automatisch abgerufen. Dadurch müssen die Daten nicht bei jedem Start neu eingegeben werden. Erst nach einer Abmeldung ist ein neuer Login erforderlich.

4.2.2 Erinnerung zur Medikamenteneinnahme

Eine der Hauptfunktionalitäten der App ist die Erinnerungsfunktion, welche auf vom Betreuer definierte Einnahmen reagiert. Zum Zeitpunkt einer für den angemeldeten Patienten bestimmten Einnahme wird am Gerät ein Alarm ausgelöst. Neben einem akustischen Signal werden Informationen über die Einnahme dargestellt, darunter das einzunehmende Medikament und die Menge sowie optionale Textinformationen, die vom Betreuer eingegeben wurden. Der Alarm läuft über einen festgelegten Zeitraum (standardmäßig 60 Se-

kunden) und sollte innerhalb diesem vom Patient per Klick auf den Button „Bestätigen“ bestätigt werden. Wird der Alarm nicht bestätigt, ist davon auszugehen, dass der Patient die Erinnerung nicht wahrgenommen hat, worauf eine Benachrichtigung zurück zum Betreuer gesendet wird.

Die Erinnerungsfunktion ist nicht nur während dem laufenden Betrieb der App verfügbar, sondern wird auch im Standby-Modus des Geräts ausgeführt und wird sogar nachgetragen, falls das Gerät zum Zeitpunkt der Einnahme kurzfristig keine Verbindung aufbauen kann.

4.2.3 Erleichtertes Telefonieren

Durch eine Kontaktliste können Anrufe schnell und einfach gewählt werden. Die Liste enthält neben den Notrufnummern der Feuerwehr, der Polizei und der Rettung die Telefonnummern sämtliche Betreuer des angemeldeten Patienten. Jeder Eintrag ist zur schnelleren Übersicht mit einem Bild versehen. Ein Klick auf einen Eintrag der Liste reicht, um einen Anruf an die gewählte Nummer zu tätigen.

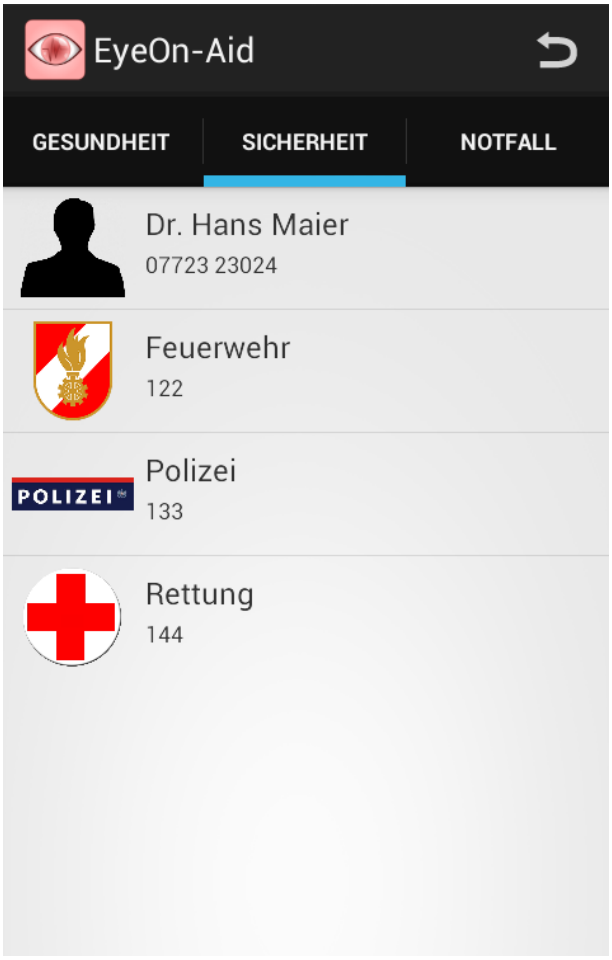


Abbildung 4.4: Schnellzugriff auf wichtige Kontakte in der App

4.2.4 Sturzerkennung

Durch die Funktion der Sturzerkennung soll im Falle eines ernsthaften Sturzes des Patienten automatisch der Betreuer alarmiert werden. Die Sturzerkennung läuft im Hintergrund und überprüft konstant den Zustand des Smartphones. Im Falle eines erkannten Sturzes erhält der Patient zur Vermeidung einer falschen Alarmierung noch eine letzte Chance zur Ablehnung des Alarms, indem er auf den Button „Falscher Alarm“ klickt. Wenn dieser nicht betätigt wird, wird davon ausgegangen, dass es sich um einen ernsthaften Vorfall handelt, worauf der Betreuer benachrichtigt wird.

4.2.5 Notfallmeldung

Die Notfallmeldung bietet dem Patienten eine schnelle und einfache Möglichkeit, bei einem Notfall den Betreuer zu alarmieren. Durch einen einzigen Klick auf die Notfalltaste wird wie gewohnt der Betreuer benachrichtigt.

Kapitel 5

Implementierung

5.1 Architektur

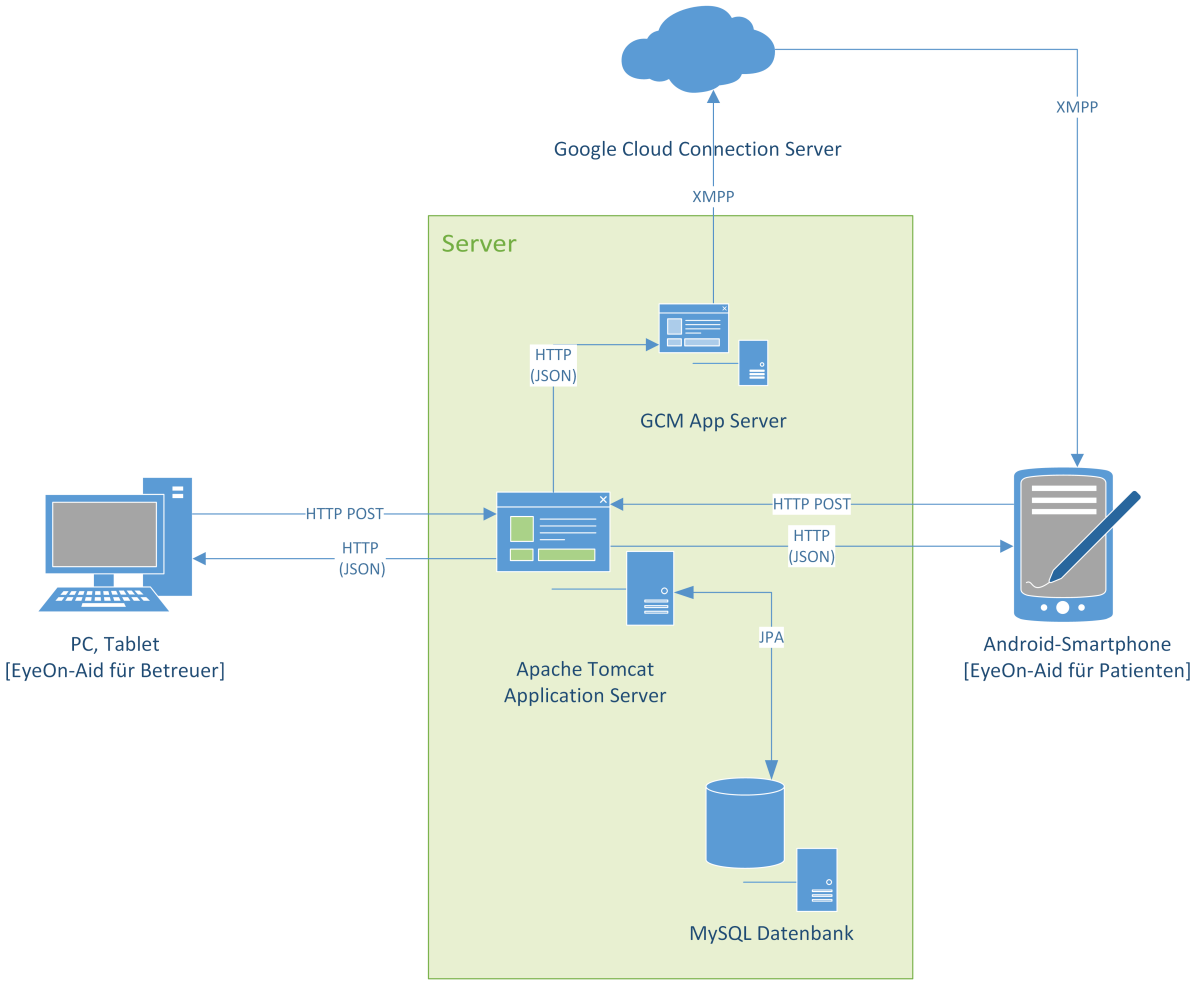


Abbildung 5.1: Die Software-Architektur von EyeOn-Aid

Die Software-Architektur, wie in Abbildung 5.1 dargestellt, kann in folgende Teile gegliedert werden.

- Web-Applikation für Betreuer
- Android-App für Patienten
- Java Servlet als Schnittstelle zwischen den Clients und dem Server
- MySQL-Datenbank zur Persistierung der Daten
- GCM App Server zum Versenden von GCM-Nachrichten
- Google Cloud Connection Server zum Empfangen von GCM-Nachrichten und deren Weiterleitung an die Android-App

5.2 Web-Applikation in Ext JS

Die Web-Applikation für Betreuer wurde in JavaScript unter Verwendung des Ext JS-Frameworks implementiert. Sie läuft plattformunabhängig und wird von jedem gängigen Browser unterstützt. Die Applikation kommuniziert per Ajax-Anfragen mit dem Server, wodurch die Seite, einmal aufgerufen, nicht mehr neu geladen werden muss.

5.2.1 Umsetzung von MVVM

Durch die Unterstützung von Namespaces und Klassenhierarchien ermöglicht es Ext JS, komplexe und umfangreichere Applikationen strukturiert und übersichtlich zu halten. In der Abbildung 5.2 ist das App-Verzeichnis dargestellt, woran bereits das MVVM-Muster ersichtlich ist.

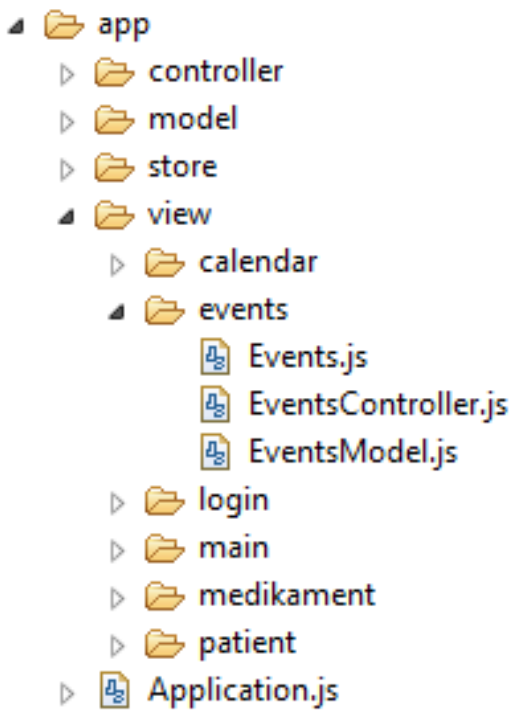


Abbildung 5.2: Die Verzeichnisstruktur der Applikation

View

Jede View in Ext JS ist eine Klasse, die von der Basisklasse `Ext.Component` erbt. Wie in Abbildung 5.2 ersichtlich, gehört zu jeder View (`Events.js`) ein ViewController (`EventsController.js`) sowie ein ViewModel (`EventsModel.js`), welche sich im gleichnamigen Verzeichnis befinden.

Bsp 5.1: Zwei Buttons mit Referenz auf den ViewController

```
1  tbar: [{
2   xtype: 'button',
3   text: 'Neuer Patient',
4   glyph: 'xf15b@FontAwesome',
5   handler: 'onAddPatient'
6  }, {
7     xtype: 'button',
8     text: 'Patient entfernen',
9     glyph: 'xf00c@FontAwesome',
10    handler: 'onDeletePatient'
11 }]
```

Im Bsp 5.1 werden zwei Buttons erstellt, welche ihre Funktionalität jeweils an den Controller auslagern. Dazu wird der Name der jeweiligen Funktion im ViewController im Attribut `handler` angegeben.

ViewController

Im ViewController werden sämtliche Funktionen, die von der View benötigt werden, implementiert. Dabei ist auf die Signatur der Funktion zu achten, da sowohl der Name der Funktion mit dem in der View angegebenen Wert übereinstimmen, als auch die übergebenen Parameter des Event-Handlers berücksichtigt werden müssen. Die jeweiligen Funktionen werden aufgelistet und können beliebigen JavaScript-Code enthalten.

Bsp 5.2: Eine Funktion des ViewControllers

```
1  onAddPatient: function() {
2     var viewModel = this.getViewModel();
3     var store = viewModel.getStore('patienten');
4     var patient = viewModel.get('thePatient');
5
6     store.add(patient);
7
8     this.sendAjaxRequest(patient, 'addPatient');
9
10    this.getView().close();
11 }
```

Bsp 5.2 zeigt den Aufbau einer Funktion des ViewControllers zum Hinzufügen eines neuen Patienten. Zuerst wird dem Wert onAddPatient die Funktion zugewiesen. In Zeile 2 bis Zeile 4 werden der Store, in dem alle Patienten gespeichert werden sowie das Patient-Objekt, welches hinzugefügt werden soll, aus dem ViewModel abgefragt. In Zeile 6 wird das Patient-Objekt dem Store hinzugefügt. Nach dem die Operation im client-seitigen Store stattgefunden hat, wird sie in Zeile 8 persistiert, indem per AJAX-Anfrage die serverseitige Funktion aufgerufen wird. Zuletzt wird das Dialogfenster in Zeile 10 geschlossen.

ViewModel

Das ViewModel ist wie auch der ViewController einer View zugeordnet. Das enthaltene `data`-Objekt speichert beliebige Daten, welche von der View verwendet werden. Daneben können Stores definiert werden, die nur für die jeweilige View zugänglich sind. Mit dem Objekt `stores` wird eine Liste von Stores verwaltet, wobei entweder private Stores, die nur von der jeweiligen View zugänglich sind, definiert werden oder aber auf globale Stores referenziert werden kann.

Bsp 5.3: Zwei Arten zum Definieren von Stores im ViewModel

```
1  medikamente: {
2     type: 'medikamente',
3     autoLoad: true
4  },
5  einnahmen: {
6     model: 'EOA.model.Einnahme',
7     autoLoad: true,
```

```
8
9     proxy: {
10          type: 'ajax',
11          url: servletUrl,
12          reader: {
13               type: 'json'
14          },
15          extraParams: {
16             action: 'getEinnahmen',
17             sv_nr: '{thePatient.sv_nr}'
18          }
19    }
20 }
```

Bsp 5.3 zeigt zwei Stores im ViewModel. Der Store `medikamente` in Zeile 1 bis Zeile 4 referenziert auf den globalen Medikamente-Store, welcher mit dem Attribut `type` angegeben wird. Das Attribut `autoLoad` wird auf `true` gesetzt, was bewirkt, dass der Store nach dem Erstellen automatisch die Daten lädt. Der Store `einnahmen` ab Zeile 5 wird hingegen selbst erzeugt. Durch das Attribut `model` wird die gewünschte Datenstruktur des Stores festgelegt, während das Objekt `proxy` in Zeile 9 bis Zeile 19 die Datenquelle und Verbindung zum Server beschreibt. Im Objekt `extraParams` werden zusätzliche Parameter für die HTTP-Anfrage angegeben. Der Parameter `action` beschreibt die Funktion die serverseitig aufgerufen wird, um die Daten zu erhalten und über den Parameter `sv_nr` wird der gewünschte Patient identifiziert. [12]

Data Binding

Data Binding ist eine Technologie, mit der Daten des ViewModels an bestimmte Komponenten gebunden werden können. Sobald eine Änderung eintrifft, werden die beteiligten Komponenten benachrichtigt und rufen automatisch die jeweiligen Getter- bzw. Setter-Methoden auf, ohne dass ein manuelles Event Handling notwendig ist. Alle Ext JS-Components haben das Attribut `bind`, über welches bestimmten Attributen der Klasse ein Wert durch Data Binding zugewiesen werden kann. [16]

Bsp 5.4: Ein Grid dessen Store mittels Data Binding zugewiesen wird

```
1  reference: 'patientsGrid',
2  xtype: 'grid',
3  title: 'Patienten',
4  bind: {
5     store: '{patienten}'
6  },
7  columns: [
8        { text: 'SV-Nr.', dataIndex: 'sv_nr'},
9        { text: 'Vorname', dataIndex: 'vorname'},
10       { text: 'Nachname', dataIndex: 'nachname'},
```

```
11      { xtype: 'datecolumn', text: 'Geburtsdatum', dataIndex: 'geb_datum',
            format: 'd.m.Y'},
12  ],
```

Im Bsp 5.4 wird ein Grid erstellt, welches die Daten des Stores des ViewModels mittels Data Binding erhält. In Zeile 4 bis Zeile 6 wird im `bind`-Objekt des Grids dem Attribut `store` der Patienten-Store des ViewModels zugewiesen. Durch die geschwungenen Klammern wird gekennzeichnet, dass es sich um ein Datenobjekt aus dem ViewModel handelt, welches demnach an den Store gebunden wird. In Zeile 7 bis Zeile 12 werden die Spalten des Grids definiert. Das Attribut `dataIndex` ist dabei der Name des jeweiligen Feldes der Datenquelle.

Neben einfachem Binding besteht auch die Möglichkeit des Two-Way-Bindings. Two-Way-Binding (dt. zweifache Bindung) beschreibt Data Binding in beide Richtungen, was bedeutet dass die Daten in Echtzeit zwischen View und Model synchronisiert werden. Jede Änderung, die in der View durchgeführt wird, wird automatisch in das ViewModel zurückgeschrieben. Daraufhin werden auch alle anderen Komponenten, die auf diese Daten gebunden sind, aktualisiert.

Ein weiteres Konzept des ViewModels sind sogenannte Formulas (dt. Formeln). Mithilfe von Formulas können kleinere Berechnungen und Manipulationen zwischen Daten im ViewModel gekapselt werden, um die View frei von Anwendungslogik zu halten. Im Bsp 5.5 wird eine Formula erstellt, mit der das aktuell ausgewählte Medikament einer Auswahlliste ermittelt wird. In Zeile 4 wird das Objekt an die Auswahl der ComboBox gebunden. Das Attribut `deep` in Zeile 5 bewirkt, dass beim Data Binding nicht nur eine Benachrichtigung stattfindet, wenn das Objekt selbst verändert wurde, sondern auch dann, wenn sich Attribute des Objekts ändern. In der Getter-Methode in Zeile 7 wird nun das aktuelle Medikament zurückgegeben. In dieser Methode könnte noch weitere Logik implementiert werden, in diesem Fall wird das Objekt aber ohne Bearbeitung zurückgegeben.

Bsp 5.5: Formula mit Binding

```
1  formulas: {
2     currentMedikament: {
3        bind: {
4           bindTo: '{medikamentComboBox.selection}',
5           deep: true
6        },
7        get: function(medikament) {
8           return medikament;
9        }
10    }
11 }
```

Model

Im Verzeichnis `model` befinden sich sämtliche Models, welche in der Applikation verwendet werden. Diese stellen eine Abbildung der Tabellen aus der Datenbank dar.

Store

Im Verzeichnis `store` befinden sich globale Stores, welche von der gesamten Applikation verwendet werden können.

Controller

Im Verzeichnis `controller` befindet sich ein globaler Controller (`Root.js`). Dieser ist beim Start der Applikation für die Weiterleitung zum Login-Bereich und danach zur Hauptansicht zuständig.

5.2.2 Styling von Ext JS-Components

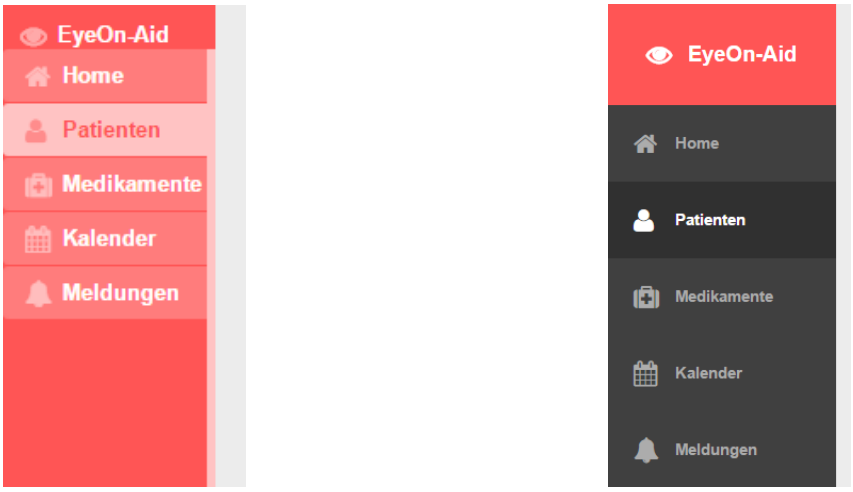
Durch die strikte Trennung von Funktionalität und Aussehen bei Ext JS-Components ist es möglich, für dieselben Components verschiedene Styles zu verwenden. Mithilfe sogenannter Mixins kann das Aussehen eines Components über zahlreiche CSS-Variablen angepasst werden.

Benutzerdefinierte Styling-Codes für Components kommen in der Applikationsstruktur in das Verzeichnis `sass/src`. Um die Styles mit der jeweiligen View, in der sie verwendet werden, zu verknüpfen, wird eine Datei mit demselben Namen in derselben Verzeichnisstruktur wie die der View und der Dateiendung `.scss` angelegt.

Bsp 5.6: Benutzerdefiniertes Styling für die Navigationsleiste

```
1 @include extjs-tab-panel-ui(
2    $ui: 'navigation',
3    $ui-tab-background-color: transparent,
4    $ui-tab-color: #acacac,
5    $ui-tab-padding: 24px,
6 );
```

Bsp 5.6 zeigt einen Ausschnitt des Mixins für die Navigationsleiste der Applikation. Die Navigationsleiste verwendet das Tab-Panel als Ausgangskomponente, welches unter anderem das CSS-Mixin `extjs-tab-panel-ui` verwendet. Dieses Mixin wird in Zeile 1 mit dem Sass-Befehl `@include` eingebunden, wodurch alle dort definierten CSS-Attribute übernommen werden. In Zeile 2 wird durch die Variable `$ui` der Name des User Interfaces bzw. des Styles definiert. Dieser Name wird in der Konfiguration des Components, welcher den Style verwenden soll, angegeben. In den folgenden Zeilen werden nun die gewünschten Variablen durch eigene Werte überschrieben.



(a) Tab-Panel mit Standard-Style (b) Tab-Panel mit eigenem Style

Abbildung 5.3: Vergleich: Navigationsleiste mit und ohne CSS Mixins

5.3 Android-App

Die mobile App wurde für das Betriebssystem Android unter Verwendung des Android-SDK für Java entwickelt.

5.3.1 Kommunikation mit Google Cloud Messaging

Google Cloud Messaging wird für die Echtzeit-Übertragung zwischen Server und Smartphone bei der Erinnerungsfunktion eingesetzt.

GCM Registrierung

Nachdem der Login erfolgreich durchgeführt worden ist, wird zunächst überprüft, ob dem angemeldeten Patienten bereits eine Registration ID zugeteilt wurde. Wenn dies nicht der Fall ist, also wenn das Feld Registration ID leer ist, muss zuerst der GCM-Registrierungsvorgang abgeschlossen werden. Bevor dies erledigt wird, wird sichergestellt, dass das Android-Gerät die Google Play Services installiert hat, welche zwingend für GCM benötigt werden. Sollte dies nicht der Fall sein, wird dies dem Benutzer per Fehlermeldung mitgeteilt und die App wird beendet. Andernfalls wird nun die Registrierung asynchron mithilfe der Android-Klasse `AsyncTask` durchgeführt. [6]

Bsp 5.7: Der GCM Registrierungsprozess

```
1 protected String doInBackground(Void... params) {
2    String msg = "";
3    try {
4       if (gcm == null) {
```

```
5           gcm = GoogleCloudMessaging.getInstance(context);
6       }
7       String regId = gcm.register(GCM_SENDER_ID);
8       patient.setRegId(regId);
9
10      msg = "Gerät wurde registriert, Registration ID=" + regId;
11
12      // Registration ID in Patient-Tabelle aktualisieren
13      HttpHelper http = new HttpHelper();
14
15      List<NameValuePair> httpParams = new ArrayList<>();
16      httpParams.add(new BasicNameValuePair("sv_nr", patient.getSvNr()));
17      httpParams.add(new BasicNameValuePair("regId", patient.getRegId()));
18      http.request("setRegId", httpParams);
19
20  } catch (IOException ex) {
21      msg = "Fehler bei Registrierung: " + ex.getMessage();
22      Toast.makeText(getApplicationContext(), msg,
            Toast.LENGTH_LONG).show();
23      finish();
24  }
25  return msg;
26 }
```

Im Bsp 5.7 ist die Methode aufgeführt, welche den Registrierungsprozess durchführt. In Zeile 4 bis Zeile 6 wird eine Singleton-Instanz der Klasse `GoogleCloudMessaging` erstellt, wobei der Context der Applikation als Parameter übergeben wird. Diese Klasse wird von der Google Play Services API zur Verfügung gestellt und bietet die Methoden zum Registrieren sowie zum Senden von GCM-Upstream-Messages an. In Zeile 7 findet nun der eigentliche Registrierungsprozess statt. Die Methode `register()` nimmt als Übergabeparameter die Sender ID der Applikation und liefert als Rückgabewert die erhaltene Registration ID, welche in Zeile 8 dem Patient-Objekt hinzugefügt wird. In Zeile 13 bis Zeile 18 wird die Registration ID dann per HTTP-Anfrage an den Server übermittelt und dort persistiert. Als HTTP-Parameter werden die Sozialversicherungsnummer zur Identifikation des Patienten sowie die eben erhaltene Registration ID übergeben. Die HTTP-Anfrage wird mithilfe des Apache-HTTP-Clients durchgeführt und wird in der Klasse `HttpHelper` gekapselt. Sollte die Registrierung fehlschlagen, da keine Verbindung zum Google Cloud Connection Server aufgebaut werden kann, wirft die Methode `register()` eine Exception, welche im Catch-Block in Zeile 20 bis Zeile 24 abgefangen wird. In diesem Fall wird eine Fehlermeldung ausgegeben und die App beendet.

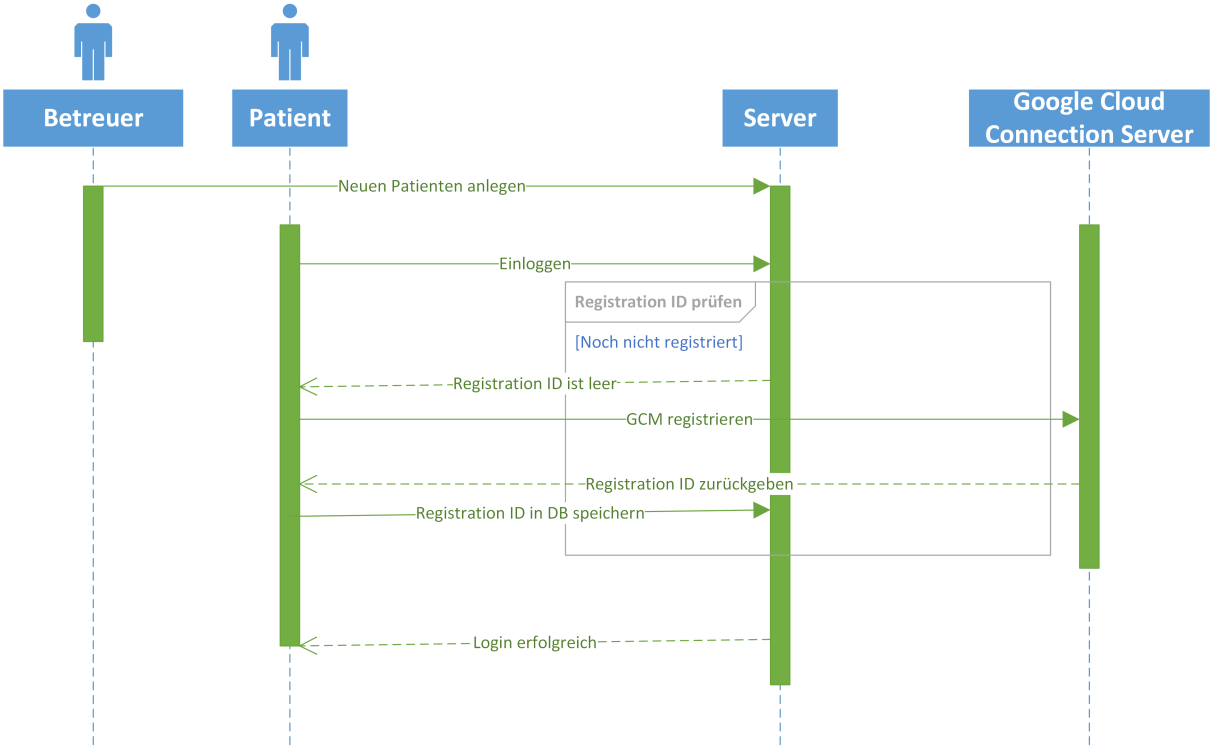


Abbildung 5.4: Login und GCM Registrierung des Patienten als Sequenzdiagramm

In Abbildung 5.4 wird nochmals der gesamte Anmeldevorgang eines Patienten dargestellt. Nachdem ein neuer Patient vom Betreuer hinzugefügt wurde, meldet sich dieser mit den angegebenen Login-Daten an. Danach wird das oben beschriebene Prozedere durchgeführt.

Empfangen von GCM-Downstream-Messages

Nachdem das Android-Gerät die GCM-Registrierung durchgeführt hat, ist es bereit, GCM-Nachrichten zu senden bzw. zu empfangen. GCM verwendet zum Übermitteln der Nachrichten einen sogenannten Broadcast Receiver. Dieser empfängt die Nachrichten und leitet sie an einen Service zur weiteren Verarbeitung weiter. Die Klasse `GcmBroadcastReceiver` ist von der Android-Klasse `WakefulBroadcastReceiver` abgeleitet. `WakefulBroadcastReceiver` ist eine spezielle Art des Broadcast Receivers, welche während der Verarbeitung der GCM-Nachricht durch den Service einen sogenannten Partial Wake Lock erstellt, wodurch sichergestellt wird, dass das Gerät nicht in den Standby-Modus übergeht. [6]

Bsp 5.8: Definition von GcmBroadcastReceiver im AndroidManifest

```
1 <receiver
2       android:name=".GcmBroadcastReceiver"
3       android:permission="com.google.android.c2dm.permission.SEND" >
4       <intent-filter>
5             <action android:name="com.google.android.c2dm.intent.RECEIVE" />
```

```
6             <category android:name="com.halu.eyeonaidapp" />
7       </intent-filter>
8   </receiver>
```

Durch den Eintrag im Android Manifest wird festgelegt, dass GCM-Nachrichten durch den `GcmBroadcastReceiver` empfangen werden. Bsp 5.8 zeigt das Receiver-Element und seine Attribute. In Zeile 2 wird der Name der Klasse des verantwortlichen Receivers angegeben. Durch die Permission `SEND` in Zeile 3 wird sichergestellt, dass der Receiver ausschließlich GCM-Nachrichten empfängt. In Zeile 4 bis Zeile 8 wird durch den Intent-Filter festgelegt, welche Typen von Broadcasts der Receiver empfängt. Der Receiver soll ausschließlich Intents vom Typ `RECEIVE` empfangen. Im Element `category` wird der Package-Name der App angegeben.

Die Klasse `GcmBroadcastReceiver` überschreibt lediglich die Methode `onReceive()`, welche den eintreffenden Broadcast verarbeitet bzw. an einen angegebenen Service zur weiteren Verarbeitung übergibt.

Die Klasse `GcmIntentService` ist nun für die eigentliche Verarbeitung der eingegangen GCM-Nachricht verantwortlich. Sie hat wieder nur die eine überschriebene Methode `onHandleIntent()`, welche im Bsp 5.9 aufgeführt ist.

Bsp 5.9: Die Verarbeitung der eingegangen GCM-Messages

```
1  protected void onHandleIntent(Intent intent) {
2       Bundle extras = intent.getExtras();
3       GoogleCloudMessaging gcm = GoogleCloudMessaging.getInstance(this);
4
5       String messageType = gcm.getMessageType(intent);
6
7       if (!extras.isEmpty()) {
8            // Normale GCM-Message
9            if
                 (messageType.equals(GoogleCloudMessaging.MESSAGE_TYPE_MESSAGE))
                 {
10                 String einnahmeText = extras.getString("einnahme_text");
11                 String einnahmeDetails =
                       extras.getString("einnahme_details");
12                 String svNr = extras.getString("sv_nr");
13
14                 Intent alarmIntent = new Intent(this,
                       AlarmActivity.class);
15                 alarmIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
16                 alarmIntent.putExtra("com.halu.eyeonaidapp.EinnahmeText",
                       text);
17                 alarmIntent.putExtra("com.halu.eyeonaidapp.EinnahmeDetails",
                       details);
18                 alarmIntent.putExtra("com.halu.eyeonaidapp.EinnahmeSvNr",
                       svNr);
```

```
19                  startActivity(alarmIntent);
20              }
21          }
22          // Wake Lock vom WakefulBroadcastReceiver aufheben
23          GcmBroadcastReceiver.completeWakefulIntent(intent);
24      }
```

In Zeile 2 werden die Extras des Intents, das sind durch den Intent übergebene Werte, abgerufen. In Zeile 3 wird, wie oben bereits erwähnt, eine Singleton-Instanz der Klasse `GoogleCloudMessaging` erstellt. Mit der Methode `getMessageType()` dieser Klasse wird in Zeile 5 der Typ der GCM-Nachricht anhand des übergebenen Intents ermittelt. Zurzeit sind die drei Typen `MESSAGE`, `SEND_ERROR` und `DELETED` in Verwendung. `MESSAGE` beschreibt eine herkömmliche Nachricht, `SEND_ERROR` bedeutet, dass bei der Übertragung ein Fehler aufgetreten ist und `DELETED` bedeutet, dass der Server die Nachricht aufgrund zu hohen Speicherbedarfs gelöscht hat. Nachdem in Zeile 7 sichergestellt wird, dass der Intent Daten übergeben hat, wird in Zeile 9 nun festgestellt, ob es sich um eine normale GCM-Nachricht handelt. Ist dies der Fall, können die Schritte zur Auslösung des Alarms eingeleitet werden. In Zeile 11 bis Zeile 13 werden die Felder, die bei der Anzeige des Alarms benötigt werden und als Extras übergeben wurden, abgerufen. In Zeile 15 bis Zeile 20 wird nun ein Intent zum Start der `AlarmActivity` erstellt. Durch das Flag `FLAG_ACTIVITY_NEW_TASK` in Zeile 16 wird ermöglicht, dass eine neue Activity außerhalb einer Activity (also im IntentService) gestartet werden kann. Die zuvor abgerufenen Parameter werden als Extras übergeben und in Zeile 20 findet der Start der Activity statt. Ist die Verarbeitung abgeschlossen, wird der Wake Lock des Receivers wieder aufgehoben, sodass das Gerät wieder in den Standby-Modus gehen darf.

5.3.2 Sturzdetektor

Das Herzstück der Sturzerkennungsfunktion ist der Sturzdetektor, welcher ermittelt, ob ein Alarm ausgelöst werden muss. Zum Erkennen eines möglichen Sturzes wird der Accelerometer (=Beschleunigungssensor), der in den meisten Android-Smartphones integriert ist, verwendet. Der Accelerometer misst Geschwindigkeitsveränderungen auf drei Achsen (X, Y und Z). Durch die Länge des Vektors der drei Achsen ($\sqrt{x^2 + y^2 + z^2}$) wird nun die tatsächliche Beschleunigung errechnet.

Theorie

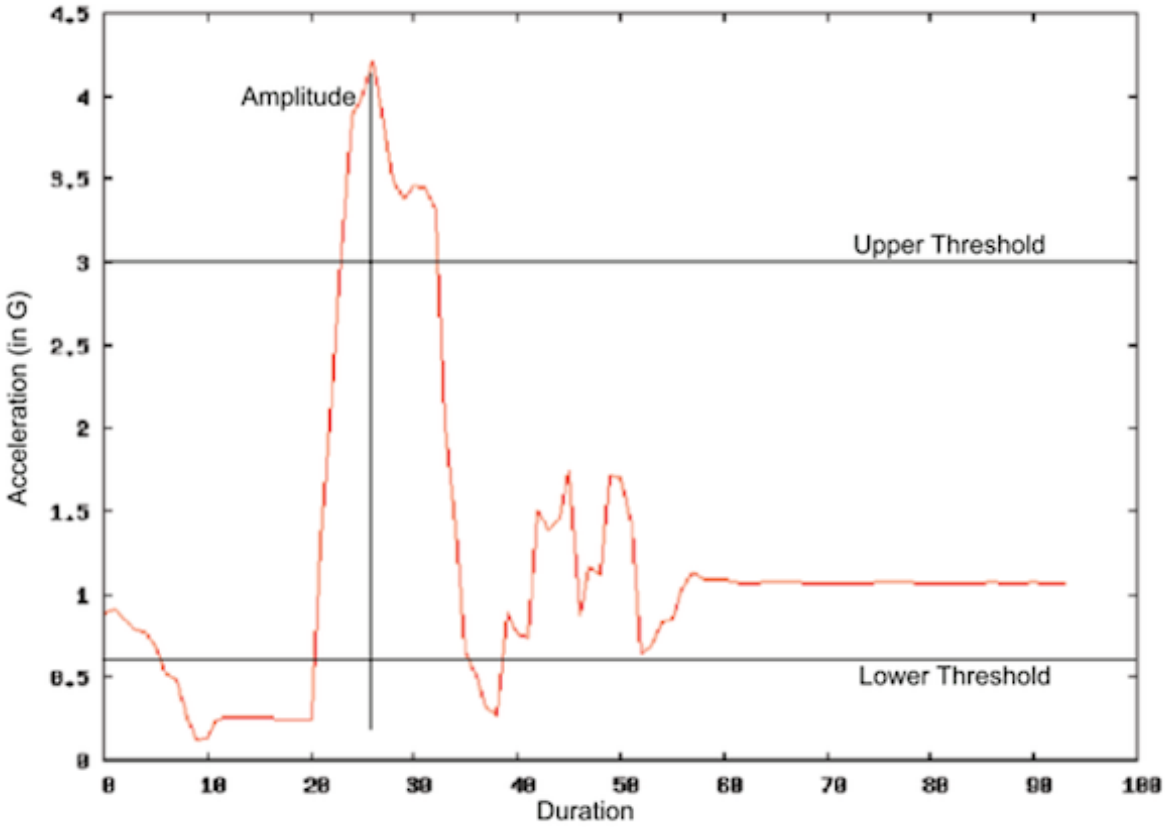


Abbildung 5.5: Ablauf eines typischen Sturzes

In Abbildung 5.5 wird ein typischer Sturz als Graph dargestellt, wobei die X-Achse den Zeitverlauf darstellt während die Y-Achse die jeweilige Beschleunigung in G (*Erdbeschleunigung* = $9.81 m/s^2$) beschreibt. Zu Beginn eines Sturzes befindet sich das Smartphone üblicherweise für kurze Zeit im freien Fall, wobei die Beschleunigung weit unter 1G, oft sogar gegen 0, sinkt. Während dieser Phase findet der eigentliche Fall statt. Auf den Fall folgt der Aufprall auf den Boden. Beim Aufprall schlägt der Graph bis zum Höchstwert aus, was je nach Fallhöhe eine Kraft von annähernd 3G (dreifache Erdbeschleunigung) ausmacht. Wird diese obere Grenze überschritten, wird zunächst von einem Sturz ausgegangen. Ein weiterer Parameter, der danach noch berücksichtigt wird, ist die Beschleunigung und Position des Smartphones nach dem Aufprall. Es wird davon ausgegangen, dass die Person nach einem ernsthaften Sturz am Boden liegen bleibt, was im Graph durch die Stabilisierung bei etwa 1G abgebildet wird. Alle diese Ereignisse spielen sich in der Regel in einem relativ kurzen Zeitraum ab. [30]

Realisierung

Die Sturzerkennung läuft auf dem Android-Gerät als Background-Service, wodurch sie ständig im Hintergrund läuft, für den Benutzer aber nur im Falle eines erkannten Sturzes bemerkbar wird. Die Erkennung erfolgt nach dem oben beschriebenen Verfahren, wobei die Vorgehensweise in Bsp 5.10 näher erläutert wird.

Bsp 5.10: Algorithmus der Sturzerkennung

```
1  float acceleration = (float) Math.sqrt(x*x + y*y + z*z);
2
3  // Freier Fall (alle Achsen gegen 0)
4  if (acceleration < FREEFALL_THRESHOLD) {
5     freefall = true;
6  }
7
8  // Impact erkannt (Beschleunigung hoch genug)
9  if (acceleration >= IMPACT_THRESHOLD && freefall) {
10    impact = true;
11    impactTimer = IMPACT_TIME;
12 }
13
14 if (impact) {
15    // Nach Aufprall, kurze Zeit warten, bis geprüft wird, ob Gerät
          waagrecht ist
16    if (impactTimer > 0) {
17       impactTimer -= diffTime;
18    }
19    else {
20       if (layOnGroundTimer < LAY_ON_GROUND_TIME) {
21          // Prüfen, ob Gerät nach Aufprall in waagrechter Lage ist
22          if (Math.abs(y) < LAY_ON_GROUND_THRESHOLD) {
23             layOnGroundTimer += diffTime;
24          }
25          else {
26             // Gerät wurde wieder aufgehoben, Falscher Alarm
27             reset();
28          }
29       }
30       else {
31          // Werte zurücksetzen
32          reset();
33
34          Intent i = new Intent(getApplicationContext(),
               ConfirmFallActivity.class);
35          i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
36          i.putExtra("sv_nr", patientSvNr);
37          startActivity(i);
38       }
```

```
39      }
40  }
```

Bsp 5.10 zeigt den Code, der zur Ermittlung eines Sturzes zehn Mal pro Sekunde ausgeführt wird. In Zeile 1 wird zunächst die Beschleunigung des Geräts errechnet. Das erste Kriterium bei der Ermittlung ist der freie Fall. Dieses wird in Zeile 4 abgefragt, indem geprüft wird, ob die Beschleunigung die untere Grenze unterschreitet. Ist dieses Kriterium erfüllt, wird in Zeile 5 die boolesche Variable `freefall` auf wahr gesetzt.

Das nächste Kriterium, der Aufprall, wird in Zeile 9 überprüft. Dabei wird, unter Voraussetzung, dass bereits das erste Kriterium erfüllt wurde, festgestellt, ob die Beschleunigung die obere Grenze überschreitet. Bei einem positiven Ergebnis wird die boolesche Variable `impact` auf wahr gesetzt. Zusätzlich wird in Zeile 11 ein Timer initialisiert, mit welchem in Zeile 16 bis Zeile 18 die Zeit zwischen Aufprall und waagrechter Position des Geräts überbrückt wird.

Nach Ablauf des Timers wird eine weitere Zeitmessung gestartet, welche nun die Zeit bis zum Auslösen des Alarms überbrückt und dabei stets überprüft, ob sich das Gerät in einer annähernd waagrechten Position befindet, wobei der absolute Wert der Y-Achse herangezogen wird (Zeile 20 bis Zeile 29).

Ist nun auch der zweite Timer abgelaufen, wird vorläufig von einem Sturz ausgegangen. Bevor der Betreuer alarmiert wird, erhält allerdings der Patient zur Sicherstellung noch die Möglichkeit, die Meldung als falschen Alarm zu deklarieren. Die dafür zuständige Activity wird in Zeile 34 bis Zeile 39 gestartet.

5.4 GCM App Server

Der App Server vermittelt zwischen dem Google Cloud Connection Server und dem Android-Endgerät. Zur Implementierung wurde die Smack Library, eine XMPP-Klassen-Bibliothek für Java, entwickelt von der Firma Jive Software, verwendet.

5.4.1 Medikamenteneinnahmen versenden

Damit das Android-Endgerät des Patienten eine Benachrichtigung zur Einnahme eines Medikaments erhält, wird eine GCM-Downstream-Message, also eine Nachricht vom Server zum Client, gesendet. Dazu wird beim Start des App Servers ein `TimerTask` erstellt, welcher in einem Intervall von 15 Minuten abfragt, ob zum jeweiligen Zeitpunkt Einnahmen fällig sind. Ist dies der Fall, wird für jede Einnahme eine GCM-Nachricht an das jeweilige Gerät, welches mit der Registration ID identifiziert werden kann, versendet.

Bsp 5.11: Das Versenden der Erinnerung zur Medikamenteneinnahme

```
1  public void handleEinnahmen() {
2     SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");
3     String currentTime = sdf.format(new Date());
```

```
4
5     HttpHelper http = new HttpHelper();
6     List<NameValuePair> loginParams = new ArrayList<>();
7     loginParams.add(new BasicNameValuePair("time", "00:45"));
8
9     String responseString = http.request("getEinnahmenByTime", loginParams);
10
11    JSONArray einnahmen = (JSONArray) JSONValue.parse(responseString);
12    for (Object o : einnahmen) {
13       JSONObject einnahme = (JSONObject) o;
14       JSONObject medikament = (JSONObject) einnahme.get("medikament");
15
16       Map<String, String> payload = new HashMap<String, String>();
17       payload.put("einnahme_text", einnahme.get("menge") + " " +
             medikament.get("einheit") + " " + medikament.get("name"));
18       payload.put("einnahme_details", (String)
             einnahme.get("information"));
19       payload.put("sv_nr", (String) einnahme.get("patientId"));
20       payload.put("action", "com.halu.eyeonaidapp.EINNAHME");
21
22       String to = (String) einnahme.get("regId");
23       if (to != null) {
24            Map<String, Object> message = new HashMap<String, Object>();
25            message.put("to", to);
26            message.put("message_id", messageId);
27            message.put("data", payload);
28            String jsonMessage = JSONValue.toJSONString(message);
29
30            Packet request = new GcmPacketExtension(jsonMessage).toPacket();
31            connection.sendPacket(request);
32       }
33    }
34 }
```

Im Bsp 5.11 ist die Methode `handleEinnahmen()` aufgeführt, welche regelmäßig aufgerufen wird. In Zeile 2 bis Zeile 3 wird die aktuelle Zeit ermittelt und in Stunden und Minuten formatiert. In Zeile 5 bis Zeile 9 werden per HTTP-Anfrage alle Einnahmen, die zur angegebenen Zeit am aktuellen Wochentag fällig sind, abgefragt. Hierzu wird wieder die Kapselungsklasse `HttpHelper` verwendet. Der Rückgabewert liegt als JSON-Array vor und wird in Zeile 11 geparst. Ab Zeile 12 wird das gesamte Array durchlaufen, wobei ein Eintrag im Array jeweils ein JSON-Objekt darstellt. Dieses und das untergeordnete JSON-Objekt `medikament` werden in Zeile 13 und Zeile 14 abgerufen. In Zeile 16 bis Zeile 20 wird der Payload der Message, also die Nutzdaten, festgelegt. In diesem Fall werden die Details zur Einnahme übergeben, welche dann beim Alarm am Endgerät angezeigt werden. Mit dem Attribut `action` kann am Client unterschieden werden, um welche Art von Nachricht es sich handelt. In Zeile 22 wird die Registration ID des Empfängers abgerufen. Wenn

diese vorhanden ist, wird nun in Zeile 24 bis 28 die GCM-Nachricht im JSON-Format zusammengestellt. Das fertige Paket zur Übermittlung wird in Zeile 30 erstellt, indem ein Objekt der Hilfsklasse `GcmPacketExtension` erstellt wird und dessen Methode `toPacket` aufgerufen wird. Diese Methode packt die eigentliche Nachricht, die im JSON-Format vorliegt, nochmals in ein XMPP-Stanza im XML-Format. In Zeile 31 findet letztendlich die Übermittlung mit dem bestehenden Connection-Objekt an den Google Cloud Connection Server statt.

5.5 EoA_Caretaker

Serverseitig läuft ein Java-Programm (EoA_Caretaker), welches sich um Anfragen von einem Client sowie um Antworten für diesen kümmert. Die Verwaltung von Anfragen und Antworten übernimmt ein Servlet (EoA_Servlet), welches Funktionen einer Datenverwaltungsklasse (EoA_DataHandler) und einer Klasse, welche dafür verantwortlich ist, Daten in einen JSON-String zu packen (EoA_JsonHandler), verwendet.

5.5.1 Begriffserklärungen

HTTP-GET-Anfrage

Die Parameter werden in der URL mitgegeben und mit einem `"?"` eingeleitet sowie durch ein `"&"` getrennt. Die Daten, welche übertragen werden, stehen somit in der URL. [45]

HTTP-POST-Anfrage

Daten stehen im HTTP-Nachrichtenrumpf des Pakets und sind somit nicht in der URL sichtbar. Dadurch ist es möglich, große Daten wie Bilder zu übertragen. [45]

HttpServletRequest

Erweitert das Interface ServletRequest und stellt Anfrageinformationen für ein HTTP-Servlet zur Verfügung. [25]

HttpServletResponse

Erweitert das Interface ServletResponse um HTTP-spezifische Funktionalitäten, um das Senden von Antworten (Responses) bereitzustellen. Das Objekt hat Zugriff auf HTTP-Headers und Cookies. [3]

PrintWriter

Der PrintWriter schreibt Objekte an einen Text-Output-Stream. Die Klasse implementiert alle print-Methoden eines PrintStreams. Der HttpServletResponse des Servlets stellt eine Methode bereit, welche einen PrintWriter zurückgibt. [24]

```
PrintWriter pw=response.getWriter();
```

5.5.2 EoA_JsonHandler

Der EoA_JsonHandler hat die Aufgabe, die Datensätze in das JSON-Format zu bringen, um dann an die Client-Applikation gesendet zu werden. Der EoA_JsonHandler hat ein Objekt der Klasse EoA_DataHandler, um Methoden, welche eine Liste von gewünschten Datensätzen zurückgeben, verwenden zu können. Als schemenhaftes Beispiel wird die Methode `MedikamentToJson()` des EoA_JsonHandler verwendet. Diese Methode wandelt eine Liste von Medikamenten in einen JSON-String um und gibt diesen zurück. Die verwendete Klassenbibliothek JSON-Simple stellt Funktionen zur Verfügung, um aus Objekten einen JSON-String zu erstellen.

```
JSONObject mainObj;
```

Das JSONObject ist eine Kollektion aus Key-Value-Paaren, welches die Methode `get()` für das Zugreifen auf die Werte und die Methode `put()` für das Hinzufügen oder Ersetzen von Werten bietet. [18]

```
JSONArray mainArr;
```

Das JSONArray ist eine Sequenz von Werten, welche Objekte wie ein weiteres JSONArray, ein JSONObject, ein Number-Objekt oder ein String sein können. [17]

Bsp 5.12: Medikamentenliste in JSONString konvertieren

```
1  ArrayList<Medikament>medikamentList=dataHandler.getMedikamentList();
2  for(Medikament m : medikamentList){
3     mainObj = new JSONObject();
4     mainObj.put("id", m.getMedikament_ID());
5     mainObj.put("name", m.getBezeichnung());
6     mainObj.put("art", m.getMedikamentArt().getBezeichnung());
7     mainObj.put("einheit", m.getMengeneinheit());
8     mainArr.add(mainObj);
9     mainObj = null;
10 }
11 return mainArr.toJSONString();
```

Die Methode `getMedikamentList()` des EoA_DataHandler gibt eine Liste aller Medikamente zurück. Die zurückgegebene Liste wird durchgelaufen und es wird ein JSONObject

erstellt. Mit der Methode `put()` der Klasse JSONObject werden Key-Value-Paare gesetzt, der Schlüssel ist ein String und der Wert das jeweilige Attribut des Medikaments. Nach einem Durchlauf wird das JSONObject in das Array eingefügt und wird danach auf null gesetzt, ansonsten würden beim nächsten Durchlauf einfach weitere Parameter an dasselbe JSONObject angefügt werden. Wenn die Liste vollständig durchgelaufen wurde, wird das JSONArray als Zeichenkette zurückgegeben. Dieser Vorgang wird durch die Methode `toJSONString()` ermöglicht.

5.5.3 EoA_Servlet

Das Servlet verwaltet die Verwaltung von Anfragen und Antworten. Der Client gibt beim Ausführen einer Aktion in der Web-Applikation einen Parameter `"action"` mit, dieser beschreibt die Aktion, welche durchgeführt werden soll (z.B. `"addMedikament"`, um ein Medikament einzufügen, `"updateMedikament"`, um ein Medikament zu bearbeiten, `"deleteMedikament"`, um ein Medikament zu löschen und `"getMedikamente"`, um alle vorhandenen Medikamente zu bekommen). Das Servlet hat zwei Methoden, `doPost()` und `doGet()`. In diesen wird die Funktionalität, welche bei der jeweiligen Anfrage durchgeführt werden soll implementiert.

Bsp 5.13: Standard Servlet-Methoden für Get und Post

```
protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
      ...
}

protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
      doPost(request,response);
}
```

Wenn nicht zwischen GET- und POST-Anfrage unterschieden werden soll, muss eine Methode die andere aufrufen. So reagiert das Servlet auf beide Anfragearten. Um einen Parameterwert in einen String zu speichern, gibt es die Methode `getParamteter("Parametername")`, welche von HttpServletRequest bereitgestellt wird.

```
request.getParameter("name");
```

Diese Parameter existieren für alle benötigten Entitätsklassen. Um herauszufinden, um welchen Parameter es sich handelt, wird ein switch-Statement implementiert, welches alle möglichen Parameter abfragt.

Bsp 5.14: Switch-Anweisung für Parameter

```
switch(parameter){
case "getMedikamente":
      ...
```

```
4       break;
5  case "add...":
6       ...
7  }
```

Parameter „get“

Wenn der Parameter mit "get" beginnt, werden die benötigten Daten im JSON-Format über einen PrintWriter an die Web-Applikation übertragen. Die Klasse EoA_JsonHandler stellt dazu die Methoden bereit.

Bsp 5.15: getMedikamente()-Methode

```
1  case "getMedikamente":
2     pw.write(jsonhandler.MedikamentToJson());
3     break;
```

Wenn der Client alle Patienten zu einem Betreuer anfordert, wird noch ein zusätzlicher Parameter benötigt.

Bsp 5.16: Zusätzlicher Parameter `asv_nr`

```
1  case "getPatienten":
2    svNr=request.getParameter("asv_nr");
3     pw.write(jsonhandler.PatientenToJson(svNr));
4     break;
```

Die Sozialversicherungsnummer des Betreuers wird mit `getParameter()` in einen String gespeichert, da die Methode `PatientenToJson()` diese benötigt.

Parameter „add“

Bei einem "add" Parameter wird ein Datensatz über die Datenverwaltungsklasse EoA_DataHandler eingefügt.

Bsp 5.17: addMedikament()-Methode

```
1  case "addMedikament":
2     Medikament medikament =
          eoahandler.insertIntoMedikament(request.getParameter("name"),
          request.getParameter("art"), request.getParameter("einheit"));
3     break;
```

Die benötigten Parameter werden durch `request.getParameter()` einer insert-Methode übergeben, welche sich um Validierung der Daten und Einfügen in die Datenbank kümmert.

Parameter „delete“ und „update“

Der `"delete"`- als auch der `"update"`-Parameter lösen eine Methode der Datenverwaltungsklasse EoA_DataHandler aus, welche sich um die nächsten Schritte kümmert, z.B. die Prüfung, ob ein Medikament existiert, um dieses dann anschließend löschen zu können. Benötigte Parameterwerte werden wieder durch die Methode `request.getParameter()` ermittelt.

5.5.4 EoA_DataHandler

Bei der Instanziierung eines Objekts der Klasse EoA_DataHandler wird ein Objekt der Klasse `EntityManagerFactory` und ein Objekt der Klasse `EntityManager` erzeugt, um Operationen auf Entitäten durchführen zu können.

Bsp 5.18: Erzeugung EntityManagerFactory und EntityManager

```
1 EntityManagerFactory factory =
      Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
2 EntityManager manager = manager = factory.createEntityManager();
```

Der `PERISTENCE_UNIT_NAME` ist jener Name, welcher in der `persistence.xml` festgelegt ist. Der EoA_DataHandler bietet Funktionen, um Operationen (SELECT, INSERT, UPDATE und DELETE) auf die Datenbankobjekte durchführen zu können. Man kann in der JPA dank ORM-Technologie mit Objekten arbeiten und auf die gebrauchten Variablen oder auf andere Objekte über Getter-Methoden zugreifen, welche durch die JavaBeans-Spezifikation vorgeschrieben sind. Es ist jedoch auch möglich, Queries in der Sprache JPQL abzusetzen. Um Queries zu erstellen, ist das Interface `javax.persistence.Query` nötig.

Als Beispiel wird die Medikament- und MedikamentArt-Entität verwendet.

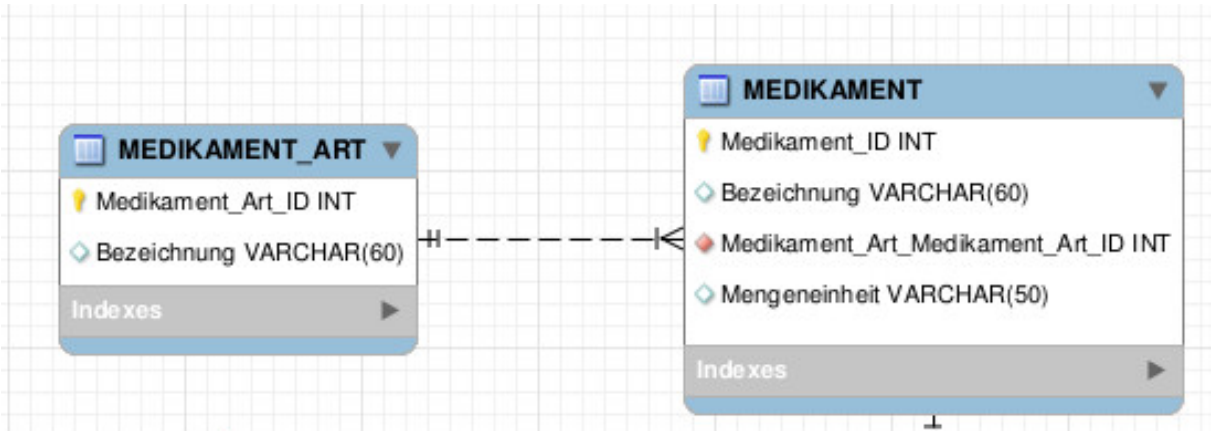


Abbildung 5.6: ERD Ausschnitt der Medikament-Entität

SELECT

Um etwas aus der Datenbank zu lesen, wird ein SELECT-Statement verwendet. Ein einfaches SELECT-Statement mit einer Query hat folgende Form:

```
SELECT alias FROM tablename alias
```

In der Praxis würde dies so aussehen:

Bsp 5.19: createQuery()-Methode

```
1 Query medikamentQ = manager.createQuery("SELECT m FROM Medikament m");
```

Der EntityManager bietet eine Funktion, eine Query abzusetzen, welche ein Query-Objekt zurückgibt. Dieses Objekt liefert jetzt alle Entitäten, welche im Persistence-Kontext gefunden werden.

Bsp 5.20: getResultList()-Methode

```
1 List<Medikament> medikamentList = medikamentQ.getResultList();
```

`Query.getResultList();`gibt die gefundenen Entitäten in Listenform zurück. Diese kann dann wie in Java üblich mit einer for-Schleife durchgelaufen werden, um danach die gewünschten Operationen durchführen zu können.

Bsp 5.21: for-Schleife

```
1 for(Medikament medikament : medikamentList){
2       ...
3 }
```

Bei SELECT-Statements mit einer WHERE-Klausel kann ein Parameter mitgegeben werden, somit muss der Wert, mit dem verglichen werden soll, nicht direkt in die Query geschrieben werden.

Bsp 5.22: SELECT-Statement mit WHERE-Klausel

```
1 Query medikamentQ = manager.createQuery("SELECT m FROM Medikament m WHERE
     UPPER(m.bezeichnung) = :bezeichnung");
2 medikamentQ.setParameter("bezeichnung", medikamentBezeichnung);
```

Mit `Query.setParamater("parametername", parameter)` wird ein String übergeben, mit dem in der Query verglichen werden soll, der `":"` in der Query kennzeichnet den übergebenen Parameter. `Alias.variable` ist das Attribut der Entität. `UPPER` ist eine Funktion um den Wert in Großbuchstaben zu vergleichen. Die JPA bietet durch ORM die Möglichkeit, mit Objekten zu arbeiten und erspart das Schreiben einer Query. Dies wird mit dem Ersetzen der Query mit WHERE-Klausel durch einen Funktionsaufruf ermöglicht. Es ist beispielsweise sehr einfach nach einem Objekt über die ID zu suchen, dies spart Schreibarbeit.

Bsp 5.23: find()-Methode

```
1 Medikament medikament=manager.find(Medikament.class, medikament_Id);
```

Durch die vom EntityManager bereitgestellte Methode `find()` kann ein Objekt durch Übergeben der spezifischen Klasse der Entität und der benötigten ID gesucht werden. Bei einer Fremdschlüsselbeziehung ist es bei einer Query nötig, einen JOIN durchzuführen.

Bsp 5.24: JOIN JPQL

```
1 Query patientQ = manager.createQuery("SELECT m FROM MedikamentArt m JOIN
      m.medikaments ms where ms.medikamentId = :id");
2 patientQ.setParameter("id", medId);
```

Hier wird die MedikamentArt eines Medikaments, welches mit der übergebenen ID übereinstimmt, zurückgegeben. Dies vereinfacht die JPA enorm. Durch einfache Getter-Methoden ist es möglich, das Fremdschlüssel-Objekt zu bekommen.

Bsp 5.25: MedikamentArt Getter-Methode

```
1 MedikamentArt art=medikament.getMedikamentArt();
```

INSERT

Um einen Datensatz einzufügen, müssen ausschließlich die Funktionen auf Objekte verwenden werden, es wird keine Query von JPQL bereitgestellt, um diese Operation ausführen zu können. Zum Einfügen des Datensatzes müssen die Setter-Methoden der jeweiligen Klasse verwendet werden, mit diesen werden die Attribute des Objekts gesetzt. Der EntityManager ist dann noch dafür zuständig, das Objekt zu persistieren und die Operation abzuschließen.

Bsp 5.26: INSERT-Vorgang

```
1  Medikament medikament = new Medikament();
2  MedikamentArt medikamentArt = new MedikamentArt();
3
4  manager.getTransaction().begin();
5  medikamentArt.setBezeichnung(medikamentArtBezeichnung);
6  manager.persist(medikamentArt);
7  manager.getTransaction().commit();
8
9  manager.getTransaction().begin();
10 medikament.setBezeichnung(medikamentBezeichnung);
11 medikament.setMedikamentArt(medikamentArt);
12 medikament.setMengeneinheit(mengeneinheit);
13 manager.persist(medikament);
14 manager.getTransaction().commit();
```

Zuerst müssen konkrete Objekte der Entitäten erzeugt werden. Dessen Attribute werden dann mit den Setter-Methoden gesetzt. Die Methode `EntityManager.persist()` setzt das jeweilige Entitätsobjekt auf `managed` und persistiert, dieser Vorgang speichert die Veränderungen im Persistence-Kontext und bei einem Commit werden diese Änderungen übernommen.

`EntityManager.getTransaction()` liefert eine EntityTransaction zurück, dies ist ein Interface, welches Transaktionen des EntityManagers verwaltet. Mit `EntityTransaction.begin()` wird die Transaktion gestartet. Die Methode `EntityTransaction.commit()` schreibt die Veränderungen in die Datenbank.

UPDATE mit einer Query

JPQL bietet die Möglichkeit, ein Update durchzuführen.

```
UPDATE tablename alias SET value = parameter where value =/>/< parameter2
```

Praktisch sieht dies dann so aus:

Bsp 5.27: UPDATE-Vorgang

```
1 Query medikamentQ = manager.createQuery("UPDATE Medikament m SET
      m.bezeichnung = :bezeichnung where m.medikamentId = 1");
2 medikamentQ.setParameter("bezeichnung", medikamentBezeichnung);
3 int result=manager.executeUpdate();
```

Diese Query aktualisiert die Bezeichnung aller Instanzen mit der ID = 1. Da eine ID eindeutig sein sollte, liefert die Variable `result` den Wert 1 zurück. Der EntityManager bietet die Methode `executeUpdate()`, welche die Anzahl der veränderten Datensätze zurückliefert. Dies kann nützlich sein, wenn eine gewisse Anzahl an Veränderung stattfinden soll.

UPDATE mit JPA

Ein Update ist mit einem Insert zu vergleichen. Ein Objekt wird über eine SELECT-Query oder über `EntityManager.find()` ausgewählt. Um Attribute, welche verändert werden sollen, auszuwählen, werden die Setter-Methoden verwendet. Der nächste Schritt ist die Persistierung des veränderten Objekts und das Durchführen eines Commits.

DELETE

Delete-Queries bieten eine Alternative zur Lösung mit der `remove()`-Methode des EntityManagers.

Ein einfaches Delete-Statement sieht so aus:

```
DELETE FROM tablename alias
```

Dieses Statement löscht alle Instanzen der Entitätsklasse. Die WHERE-Klausel ermöglicht es, spezifische Objekte zu löschen.

```
DELETE FROM tablename alias WHERE value =/>/< :parameter
```

Angewendet auf die Medikamentenklasse sieht dies nun so aus:

Bsp 5.28: DELETE-Vorgang

```
1 Query medikamentQ = manager.createQuery("DELETE FROM Medikament m WHERE
      m.bezeichnung = :bezeichnung");
2 medikamentQ.setParameter("bezeichnung", medikamentBezeichnung);
```

Dieses Statement löscht alle Instanzen, welche den übergebenen String als Bezeichnung haben. Die Lösung mittels `remove()`-Methode des EntityMangers entpuppt sich als einfacher, da das Objekt, welches gelöscht werden soll, nur übergeben werden muss. Die Select-Query oder die `manager.find()`-Methode liefert das benötigte Objekt, welches schließlich der `EntityManager.remove()`-Methode übergeben wird.

Bsp 5.29: remove()-Methode

```
1 manager.getTransaction().begin();
2 manager.remove(medikament);
3 manager.getTransaction().commit();
```

`Manager.remove()` entfernt die übergebene Instanz aus dem Persistence-Kontext, mit `.commit()` wird das Entfernen auf der Datenbank durchgeführt.

Kapitel 6

Zusammenfassung

6.1 Erbrachte Leistungen

Im Folgenden werden sämtliche Leistungen, die im Rahmen der Arbeit erreicht wurden, zusammenfassend aufgelistet.

6.1.1 Funktionen für Betreuer

- Verwaltung von Patienten
  - Die Stammdaten des Patienten können verwaltet werden
  - Eine Liste von Krankheiten des Patienten kann geführt werden
  - Der Medikamentenplan kann verwaltet werden, wobei Einnahmen für den Patienten angeordnet werden können
- Verwaltung von Medikamenten
  - Eine Liste von Medikamenten für die Einnahme der Patienten kann verwaltet werden
- Übersicht über Meldungen und Ereignisse
  - Eine Liste von eingegangenen Meldungen kann betrachtet werden
  - Neu eingegangene Meldungen werden entsprechend gekennzeichnet

6.1.2 Funktionen für Patienten

- Erinnerung an Medikamenteneinnahmen
  - Zum Zeitpunkt einer vom Betreuer angeordneten Einnahme wird am Smartphone ein Alarm ausgelöst

  - Wird dieser nicht bestätigt, so wird dies dem Betreuer gemeldet
- Erleichtertes Telefonieren
  - Eine Liste von wichtigen Kontakten wird angezeigt
  - Mit nur einem Klick kann ein Anruf getätigt werden
- Sturzerkennung
  - Im Falle eines erkannten Sturzes wird automatisch der Betreuer alarmiert
- Notrufmeldung
  - Bei einem Notfall kann mit einem einzigen Klick der Betreuer alarmiert werden

6.2 Verbesserungen

Während alle zu Beginn definierten Anforderungen erfüllt wurden, gibt es durchaus einige Verbesserungsmöglichkeiten, bzw. Arbeiten, die auf der bestehenden Software aufsetzen können.

6.2.1 Verwendung von Smartwatches

Die Verwendung einer Smartwatch anstatt des Smartphones könnte einen erheblichen Vorteil in Punkto Benutzerfreundlichkeit bringen.

Ältere Menschen sind häufig noch nicht so sehr mit dem Umgang mit Smartphones vertraut. Für sie kann es schnell zu einer Belastung werden, das Gerät immer und überall bei sich zu tragen. Vor allem bei Menschen, die an leichter Demenz leiden, kann es schnell passieren, dass sie ihr Smartphone verlegen und nicht wieder finden. Die Software verliert dadurch gewissermaßer ihren Nutzen.

Mithilfe der Smartwatch, die bequem am Handgelenk, gewohnt von einer Armbanduhr, getragen wird, können diese Probleme größtenteils eliminiert werden. Grundsätzlich muss das Gerät nicht abgenommen werden, wodurch sich die Patienten nicht mehr darum kümmern müssen, ob bzw. wie sie es bei sich tragen.

Eine große Verbesserung könnte des Weiteren bei der Sturzerkennung erreicht werden. Dadurch, dass das Gerät direkt am Körper getragen wird, entsprechen die Werte des Beschleunigungssensors eher der tatsächlichen Bewegung der Person. Somit können genauere Ergebnisse erzielt werden. Falsche Alarmierungen, z.B. durch versehentliches Fallenlassen des Geräts, können bei der Smartwatch ebenfalls ausgeschlossen werden.

Technisch wäre die Umstellung auf eine Smartwatch kein großer Aufwand, da auf den meisten Smartwatches das Betriebssystem Android Wear, welches von Android abgeleitet ist, läuft. Bis auf einige kleine Anpassungen könnte der Quellcode direkt übernommen werden.

6.2.2 Aufzeichnung der Position von Patienten

Eine weitere Maßnahme zur Überwachung der Patienten wäre die Aufzeichnung der aktuellen Position eines Patienten über die GPS-Koordinaten seines Smartphones. Durch eine regelmäßige Übertragung dieser Werte könnte der Betreuer eine Übersicht über die Aktivitäten eines Patienten erhalten und so bei unregelmäßigem Verhalten handeln.

Eine solche vollständige Überwachung wäre allerdings rechtlich und moralisch gesehen etwas fragwürdig, da die Freiheit des Patienten gewissermaßen eingeschränkt werden würde. Eine praktikablere Variante wäre es, die Position nur im Falle eines Sturzes bzw. bei Betätigen der Notfalltaste zu übermitteln.

6.2.3 Anbindung an ein Bluetooth-EKG-Gerät

Eine Möglichkeit, die gesundheitliche Verfassung eines Patienten überwachen zu können, ist die Auswertung der Daten eines EKG (Elektrokardiogramm). Das EKG misst Herzstromaktivitäten und stellt diese in Form von Kurven dar. Durch die Analyse dieser Daten kann beispielsweise auf Herzinfarkt oder auch Überdosierung bestimmter Medikamente geschlossen werden. [5]

Diverse Hersteller bieten kompakte EKG-Geräte für Endverbraucher an, welche die Daten über Bluetooth übertragen können. Diese Daten könnten vom Smartphone des Patienten empfangen und ausgewertet sowie an den Betreuer übermittelt werden. Der Betreuer erhält so eine zuverlässige Übersicht und könnte bei kritischen Werten alarmiert werden. [9]

6.3 Erfahrungen

Durch die Diplomarbeit EyeOn-Aid konnten wir zahlreiche Erfahrungen im Umgang mit den verwendeten Technologien sammeln. Besonders die mehrteilige Architektur der Software, bestehend aus der Web-Applikation und der mobilen App, und dessen Zusammenspiel und Kommunikation über den Server stellte sich als komplexe Aufgabe heraus. Dazu kam noch die Anforderung, diverse Nachrichten in Echtzeit zwischen dem Server und der mobilen App austauschen zu können, wobei sich der Service Google Cloud Messaging zwar als ideale Lösung aber gleichzeitig als komplettes Neuland für uns herausstellte. Alle diese Schwierigkeiten konnten wir letztendlich hervorragend lösen.

Nicht nur die technische Seite, sondern auch die Verfassung dieser Diplomschrift brachte uns unschätzbare Vorteile für unsere zukünftige Karriere. Rückblickend betrachtet ist die Diplomarbeit mit viel Arbeitsaufwand verbunden, welcher sich jedoch als äußerst lohnend erwies und uns mit Sicherheit in der Zukunft, ob im Studium oder im Berufsleben, weiterhelfen wird.

Abbildungsverzeichnis

Codebeispiele

Literaturverzeichnis

[1] Server-side javascript: Back with a vengeance. http://readwrite.com/2009/12/17/server-side_javascript_back_with_a_vengeance, Dezember 2009.

[2] Adobe. Adobe photoshop cc 2014 javascript scripting reference. http://wwwimages.adobe.com/content/dam/Adobe/en/devnet/photoshop/pdfs/photoshop_scriptref_js.pdf, 2014.

[3] Apache. Interface httpservletresponse. https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html.

[4] Stan Bershadskiy. A dive into the sencha class config system. http://moduscreate.com/a-dive-into-the-sencha-class-config-system/, Oktober 2014.

[5] Dr. Peter Borlinghaus and Univ. Prof. Dr. Heinz Drexel. Elektrokardiographie (elektrokardiogramm, ekg). http://www.netdoktor.at/untersuchung/ekg-8296, Februar 2006.

[6] Google Developers. Google cloud messaging for android. https://developer.android.com/google/gcm/index.html.

[7] XMPP Standards Foundation. Xmpp technologies overview. http://xmpp.org/about-xmpp/technology-overview/.

[8] Jesse James Garrett. Ajax: A new approach to web applications. http://www.adaptivepath.com/ideas/essays/archives/000385.php, Februar 2005.

[9] Medset Medizintechnik GmbH. Ekg-sensor mit bluetooth...flashlight sensor bt 12. http://www.medset.com/ekg-verstaerker-bluetooth1.html.

[10] Raphaël Hertzog and Roland Mas. What is debian? http://debian-handbook.info/browse/stable/the-debian-project.html\#sect.what-is-debian.

[11] IBM. Jpa - loader. http://www-01.ibm.com/support/knowledgecenter/SSTVLU_7.1.0/com.ibm.websphere.extremescale.over.doc/cxsljpaload.html.

[12] Sencha Inc. Ext js api documentation. http://docs.sencha.com/extjs/5.0.1/\#!/api, November 2014.

[13] Sencha Inc. Sencha compiler reference. http://docs.sencha.com/cmd/5.x/advanced_cmd/cmd_compiler_reference.html, Dezember 2014.

[14] Sencha Inc. Class system. http://docs.sencha.com/extjs/5.1/core_concepts/classes.html, März 2015.

[15] Sencha Inc. Components. http://docs.sencha.com/extjs/5.1/core_concepts/components.html, März 2015.

[16] Sencha Inc. View models and data binding. http://docs.sencha.com/extjs/5.0/application_architecture/view_models_data_binding.html, März 2015.

[17] json.org. Class jsonarray. http://www.json.org/javadoc/org/json/JSONArray.html.

[18] json.org. Class jsonobject. http://www.json.org/javadoc/org/json/JSONObject.html.

[19] Arthur Kay. Ext js 5: Mvc, mvvm and more. http://www.sencha.com/blog/ext-js-5-mvc-mvvm-and-more, Mai 2014.

[20] Sean Kelly. Speeding up ajax with json. http://www.developer.com/lang/jscript/article.php/3596836/Speeding-Up-AJAX-with-JSON.htm, April 2006.

[21] Paul Krill. Javascript creator ponders past, future. http://www.infoworld.com/article/2653798/application-development/javascript-creator-ponders-past--future.html, Juni 2008.

[22] Netscape. Industry leaders to advance standardization of netscape's javascript at standards body meeting. http://cgi.netscape.com/newsref/pr/newsrelease289.html, November 1996.

[23] objectdb. Annotation table. http://www.objectdb.com/api/java/jpa/Table.

[24] Oracle. Class printwriter. http://docs.oracle.com/javase/7/docs/api/java/io/PrintWriter.html.

[25] Oracle. Interface httpservletrequest. http://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpServletRequest.html.

[26] Oracle. Mysql workbench. https://www.mysql.de/products/workbench/.

[27] Oracle. Package javax.persistence. http://docs.oracle.com/javaee/6/api/javax/persistence/package-summary.html.

[28] phpMyAdmin contributors. Bringing mysql to the web. http://www.phpmyadmin.net/home_page/index.php.

[29] Sass-Team. Sass-reference. http://sass-lang.com/documentation/file.SASS_REFERENCE.html, Februar 2015.

[30] Frank Sposaro and Gary Tyson. ifall: an android application for fall monitoring and response. Master's thesis, Florida State University, 2009.

[31] Tutorialspoint. Jpa - introduction. http://www.tutorialspoint.com/jpa/jpa_introduction.htm.

[32] Tutorialspoint. Jpa - orm components. http://www.tutorialspoint.com/jpa/jpa_orm_components.htm.

[33] Tutorialspoint. Servlets - overview. http://www.tutorialspoint.com/servlets/servlets_overview.htm.

[34] World Wide Web Consortium (W3C). Document object model (dom) level 1 specification. http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/, Oktober 1998.

[35] Webopedia. Pojo. http://www.webopedia.com/TERM/P/POJO.html.

[36] Christian Wenz. *JavaScript und AJAX*. Rheinwerk Computing, 2006.

[37] Wiki. Javaserver pages. http://de.wikipedia.org/wiki/JavaServer_Pages, Oktober 2014.

[38] Wikipedia. Eclipse ide. http://de.wikipedia.org/wiki/Eclipse_(IDE).

[39] Wikipedia. Mysql. http://de.wikipedia.org/wiki/MySQL.

[40] Wikipedia. Anwendungsserver. http://de.wikipedia.org/wiki/Anwendungsserver, März 2014.

[41] Wikipedia. Anfrageoptimierer. http://de.wikipedia.org/wiki/Anfrageoptimierer, März 2015.

[42] Wikipedia. Annotation(java). http://de.wikipedia.org/wiki/Annotation_(Java), März 2015.

[43] Wikipedia. Apache tomcat. http://en.wikipedia.org/wiki/Apache_Tomcat, April 2015.

[44] Wikipedia. Common gateway interface. http://de.wikipedia.org/wiki/Common_Gateway_Interface, März 2015.

[45] Wikipedia. Hypertext transfer protocol. http://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol, April 2015.

[46] Wikipedia. Kernel (operating system). http://en.wikipedia.org/wiki/Kernel_(operating_system), April 2015.

[47] Wikipedia. Servlet. http://de.wikipedia.org/wiki/Servlet, März 2015.

[48] Wikipedia. Web-archivierung. http://de.wikipedia.org/wiki/Web-Archivierung, April 2015.