ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Konzeption und Realisierung eines Heart Rate Monitoring Framework für Android Wear

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Dennis Drzosga
dennis.drzosga@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Marc Schickler

2015

Fassung 24. Oktober 2015

Satz: PDF-LaTeX 2ε

Kurzfassung

Das im März 2014 vorgestellte Betriebssystem Android Wear, welches speziell für Smartwatches angepasst wurde, bietet Entwicklern neue Möglichkeiten ihre bisherigen mobilen Anwendungen um eine *Android Wear* Anwendung zu erweitern.

Im Rahmen dieser Arbeit wird ein Framework konzipiert und realisiert welches ermöglicht, mit einer Android Wear Smartwatch zu kommunizieren und Zugriff auf die Sensoren der Uhr zu erhalten. Durch die Kommunikation zwischen Smartphone und Smartwatch soll in regelmäßigen Abständen auf den Herzfrequenz-Sensor der Uhr zugegriffen werden und die gemessene Herzfrequenz an das Smartphone übertragen werden. Zusammen mit der erhobenen Herzfrequenz werden weitere Attribute wie Standort, Uhrzeit und aktuelle Aktivität des Nutzers gespeichert, um so eine in der Anwendung möglich Auswertung durchzuführen. Dies soll ermöglichen, dass beispielsweise Ärzte ihre Patienten, die mit einer Android Wear Smartwatch ausgestattet sind, durch eine regelmäßige Abfrage zu untersuchen.

Danksagung

In erster Linie möchte ich mich bei all denen bedanken, die mich während der Erstellung dieser Arbeit unterstützt haben.

Besonders meinem Betreuer, Marc Schickler, der mich bei meiner Arbeit betreut hat und regelmäßig mit Tipps und Verbesserungsvorschlägen weitergebracht hat, bedanke ich mich recht herzlich. Auch möchte ich mich beim Institut für Datenbanken und Informationssysteme bedanken, das mir im Namen von Herrn Prof. Dr. Manfred Reichert, diese Abschlussarbeit ermöglicht haben.

Nicht zuletzt gilt der Dank auch meiner Familie, die mich während der kompletten Arbeit unterstützt und mir das Studium ermöglicht hat.

Inhaltsverzeichnis

1

Einleitung

Mit der Veröffentlichung des Android Wear Betriebssystems im März 2014 hat Google sein bisheriges Betriebssystem erweitert. Ziel war es, das Betriebssystem auf Smartwatches, auch bekannt als Wearables, auszuführen. Bei einer Smartwatch handelt es sich um eine Armbanduhr, die über zusätzliche Computerfunktionalität sowie einen Internetzugriff verfügt. Die Funktionalität der Uhr lässt sich mittels mobiler Anwendungen individuell aufrüsten [BBKS15]. Die Smartwatches bieten eine neue Dimension der Computernutzung im Alltag. Ausgaben von Smartphone oder Tablet können mittlerweile am Handgelenk angezeigt werden. Ebenso sind Eingaben über das Display oder die Spracherkennung möglich. Für viele der Funktionen benötigen die Smartwatches bislang allerdings ein verbundenes Gerät wie ein Smartphone oder ein Tablet [Flo15]. Hersteller wie Samsung, Sony und Motorola haben mit der Veröffentlichung des Android Wear Betriebssystems auch gleich die ersten hauseigenen Geräte vorgestellt. Auf den ersten Blick handelsübliche Uhren, die erst nach genauerer Betrachtung zeigen, dass Zeiger

und Ziffernblätter durch kleine Touch-Displays ersetzt wurden. Eine Smartwatch ersetzt, wie bereits erwähnt, nicht nur die Anzeige von Uhrzeit und Datum, sondern bringt auch eine Reihe neuer Funktionalitäten mit sich. Ein Beispiel hierfür ist die Ausgabe von Benachrichtigungen auf dem Smartphone oder Tablet. Nachrichten werden so direkt auf dem Display der Smartwatch angezeigt. Des Weiteren sind spezielle Sensoren in der Uhr verbaut, die Funktionen wie Schrittzähler und Herzfrequenzmesser ermöglichen. Auf der zuletzt genannten Funktion, dem Herzfrequenz-Sensor, baut diese Arbeit auf. Der Sensor zur Messung der Herzfrequenz, welche durch die Anzahl der Herzschläge pro Minute definiert ist, stellt ein optimales Fundament dar, um die Funktionalität der Uhr individuell aufzurüsten. Dies geschieht mittels eines Frameworks, welches im Rahmen dieser Arbeit entwickelt wird. Ein Framework, das an sich kein eigenständiges Programm ist, dient in diesem Kontext als Schnittstelle und verfügt über gewissen Funktionen die zur Verwendung bereit stehen.

1.1 Problemstellung und Zielsetzung

Die Position der Smartwatch am Handgelenk ist gerade für das Messen der Herzfrequenz optimal, da dort die Arterien oberflächlich liegen und somit eine exakte Messung gewährleistet werden kann [SL11]. Die Herfrequenz bezeichnet die Herzschläge pro Minute und gibt somit Aufschluss über die Geschwindigkeit des Herzschlages. Die Geschwindigkeit des Herzschlages spielt in der Trainingskontrolle eine bedeutende Rolle [Gra01]. Vor allem im Ausdauersport ist ein Herzfrequenzmesser von großer Bedeutung, da der Sportler so seine Leistung optimal überwachen und angemessen ausreizen kann, ohne seine Gesundheit zu gefährden.

Anbieter wie Google, Samsung oder Motorala bieten bereits fertige Anwendungen zur Messung, Speicherung, Auswertung und Darstellung der Herzfrequenz an. Sie setzen aber voraus, dass der Benutzer intime gesundheitliche Werte zur möglichen Verwendung und Weitergabe an Dritte freigibt. Gerade in Zeiten, in denen der Datenschutz eine immer größer werdende Bedeutung gewinnt, reagieren viele Benutzer sensibel auf dieses Thema. Viele Benutzer verzichten, aus Angst vor Missbrauch ihrer wertvollen

persönlichen Werte, auf diese bedeutende Funktion der Herzfrequenzmessung [Bre15a].

Mit dem im Rahmen dieser Arbeit entwickelten Frameworks soll dem künftigen Benutzer die Möglichkeit geboten werden selbst zu entscheiden, wo seine erhobenen Werte und Daten gespeichert werden. Die Speicherung kann lokal auf dem Gerät oder auf einem Webserver stattfinden, ohne dass große Konzerne davon profitieren können. Des Weiteren soll das Framework universell einsetzbar sein, so dass Entwickler mobiler Anwendungen das Framework mit nur wenigen Anpassungen in ihre bereits bestehenden mobilen Anwendungen implementieren können. Im Kontext anderer Anwendungen soll das Framework auch dort Verwendung finden. Beispielsweise im Zusammenhang mit einer Augmented Reality Anwendung, welche es mit dem Framework ermöglichen soll, den Nutzer zu überwachen. So besteht die Möglichkeit das Verhalten des Nutzer aufgrund der Herzfrequenz zu untersuchen [SPSR15].

1.2 Aufbau der Arbeit

Das erste Kapitel dieser Arbeit befasst sich mit den Grundlagen. Um den Sachverhalt genauer verstehen zu können, werden zuerst die Funktionen von Android Wear erläutert, so dass ein Überblick über die zahlreichen Funktionen dargelegt wird. Des Weiteren wird im zweiten Kapitel eine Analyse der bisher vorhandenen mobilen Anwendungen durchgeführt. Im Anschluss folgt im dritten Kapitel die Anforderungsanalyse, welche in funktionale und nicht-funktionale Anforderungen gegliedert ist. Kapitel 4 bietet einen Überblick über die Architektur des Heart Rate Monitoring Frameworks. Dabei wird genauer auf die einzelnen Software- und Hardwarekomponenten eingegangen. Das Kapitel 5 befasst sich mit der Implementierung des Frameworks, welches dort in die einzelnen Komponenten gegliedert und der jeweilige Aufbau erklärt wird. Daraufhin folgt das Kapitel mit dem Anforderungsabgleich, worin die anfangs definierten Anforderungen analysiert und bewertet werden. Letztendlich endet die Arbeit mit einer abschließenden Diskussion zum Thema Datenschutz im Zusammenhang mit Gesundheitsdaten, mit einer Zusammenfassung der Arbeit sowie einem Ausblick auf mögliche Erweiterungen des Frameworks.

2

Grundlagen

In dieser Arbeit wird mit Googles neuem, speziell für Uhren entwickelten Betriebssystem, *Android Wear* gearbeitet. Um den Sachverhalt verstehen zu können, werden gewisse Vorkenntnisse benötigt, welche in diesem Kapitel erläutert werden. Beispielsweise wird der Aufbau des Betriebssystem oder einzelne Funktionen im Detail erklärt. Anschließend folgt eine Marktanalyse, welche die bereits vorhandenen mobilen Anwendungen der Hersteller Google, Samsung und Motorola in einem Vergleich darlegt.

2.1 Android Wear

Android Wear ist eine vereinfachte Version des weltweit bekannten Betriebssystems *Android* von Google. Optimiert für Wearables werden dort nur die wichtigen Informationen angezeigt, wie das Anzeigen seines nächsten Termins mit Ort- und Zeitangabe oder den

Erhalt einer neuen E-Mail, bei welcher nur der Absender sowie der Betreff angezeigt wird, so dass ein guter Überblick gewährleistet ist. Google entwickelt *Android Wear* laufend weiter, da seit kurzem auch Konkurrent Apple seine hauseigene Smartwatch *Apple Watch* auf den Markt gebracht hat. Somit muss Google in Sachen Android Wear ständig innovativ bleiben.

2.1.1 Die Idee

Abbildung 2.1: Android Wear Watchfaces [Goo15a]

Auf der hauseigenen Webseite angepriesen mit dem Slogan “Sechs einzigartige Uhren - Hunderte ansprechende Zifferblatt-Designs“ möchte Google vor allem auf die Möglichkeit aufmerksam machen, dass die Uhren wandelbar sind [Goo15a]. Ein Gerät, hunderte, wenn nicht schon tausende verschiedene Ziffernblatt-Designs sind bereits vorhanden und individuell einsetzbar (siehe Abbildung 2.1) [Goo15c]. Im Großen und Ganzen kann der Nutzer nach Belieben entscheiden, wie seine Uhr aussehen soll. Ebenso sind diese Wearables dazu gedacht, Informationen von mobilen Anwendungen zusammengefasst und ansprechend auf der Uhr darzustellen. Dem Nutzer soll somit ein einfacher und effektiver Überblick über alle nötigen Informationen verschafft werden ohne dabei jedes Mal auf sein Smartphone zurückgreifen zu müssen.

2.1.2 Die Funktionen

Google fokussiert sich mit *Android Wear* auf das Wohlbefinden des Nutzers. Dieses soll gewisse Funktionen anwenden können, ohne dabei das Smartphone zu bedienen. Die Spracheingabe, welche bereits von *Android* bekannt ist, kann auch ganz bequem auf der Smartwatch genutzt werden. Dies ermöglicht Funktionen, wie das Versenden von SMS, Chatnachrichten und E-Mails. Ebenso erhält der Nutzer auf gestellte Fragen innerhalb kürzester Zeit die Antworten kompakt dargestellt.

Durch die in der Smartwatch verbauten Hardware-Komponenten, wie beispielsweise ein Beschleunigungssensor, ergeben sich neue Funktionen, welche es ermöglichen, dass sich das Display der Smartwatch erst nach einer gewohnten Handbewegung einschaltet und auf diese Weise die Akkulaufzeit verlängert werden kann. Funktionen, wie das Navigieren mit Google Maps, bringen ebenso Vorteile mit sich, da die Möglichkeit besteht die Anzeige der Navigationsinformationen auf die Smartwatch zu verlagern. Weitere Funktionen, wie das Anzeigen der täglichen Termine, das Verfolgen von Fitnesszielen, die Erstellung neuer Erinnerungen durch die Spracheingabe sowie die Steuerung vieler weiterer Smartphone-Funktionen werden ermöglicht.

2.1.3 Aufbau einer Android Wear Anwendung

Um eine *Android Wear* Anwendung zu implementieren, empfiehlt Google die Nutzung von *Android Studio*. Wird dort ein neues Projekt erstellt, so ist dies auch gleichzeitig der Workspace, der bereits aus Eclipse bekannt ist. Jedes dieser Projekte besitzt sogenannte Module, welche unabhängig voneinander erstellt werden können. Die einzelnen Module in *Android Studio* sind gleichgestellt mit den aus Eclipse bekannten Projekten, welche sich dort in einem Workspace befinden. In der Norm besteht ein neu angelegtes *Android Studio* Projekt aus einem *mobile*-Modul. Über das Menü kann ein bestehendes Projekt durch hinzufügen eines *Android Wear*-Moduls erweitert werden. Das neu erstellte Modul wird als eigenständige Anwendung implementiert. Um diese aber installieren zu können, wird eine Smartphone-Anwendung benötigt, welche auch als *Companion-App* bezeichnet wird. Die Smartphone-Anwendung installiert so auch die *Android Wear* Anwendung,

da eine direkte Installation durch den Nutzer noch nicht vorgesehen ist, was ebenfalls für die Deinstallation einer Anwendung gilt. Damit alles reibungslos funktionieren kann, müssen beide Module die selbe Package-ID besitzen (siehe Abb. 2.2). Zudem muss das auf Gradle basierende Build-Skript der Smartphone-Anwendung die Abhängigkeit des *Android Wear*-Moduls enthalten. Um den in der Arbeit benötigten Sensor der Smartwatch nutzen zu können, müssen Smartphone und Smartwatch miteinander kommunizieren, da das Smartphone keinen direkten Zugriff auf die Uhr hat. Neben der aktiven Bluetooth-Verbindung erfolgt die Kommunikation anhand der von Google veröffentlichten *Message API*. Durch die Übermittlung bestimmter Nachrichten kann so die Smartwatch angesprochen und dazu animiert werden, die Herzfrequenz zu messen und anschließend den erhobeben Wert an das Smartphone zu übermitteln.

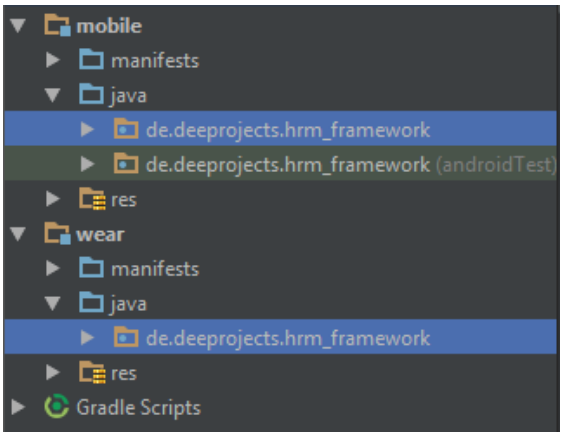

Abbildung 2.2: Aufbau einer Android Wear Anwendung

2.1.4 Implementierung

Jeder Android-Entwickler hat die Möglichkeit, die bereits entwickelten mobilen Anwendungen um ein *Android Wear* Modul zu erweitern, so dass beispielsweise bisher nur auf dem Smartphone angezeigte Benachrichtigungen mit gewissen Anpassungen auch auf der Smartwatch erscheinen. Durch die Smartwatch wird das Kommunizieren via Messengern wie *WhatsApp* oder *Google Hangouts* von einen neuen Blickwinkel betrachtet, da durch die Spracheingabe die gewünschte Nachricht problemlos eingesprochen werden

kann. Die Abhängigkeit vom Smartphone muss allerdings aktuell noch bestehen bleiben, denn mobile Anwendungen, die bereits auch speziell für Smartwatches entwickelt worden sind, lassen sich, aufgrund der fehlenden Internetverbindung, bisher nur über den auf dem Smartphone installierten *Play Store* heruntergeladen und anschließend auf die Uhr übertragen werden. Die größte Hürde der Entwickler besteht derzeit noch darin die Anwendungen so zu implementieren, dass die bisher noch relativ kurzen Akkulaufzeiten nicht noch intensiver beansprucht werden.

2.1.5 Fitness

Mit den *Android Wear* Uhren hat sich definitiv auch der Fitness-Trend enorm gesteigert [KL15]. Die Vielzahl der Funktionen, die die Uhren mit sich bringen, sind das ideale Fundament für die Nutzung solcher Fitness-Anwendungen. Die Herausgeber dieser Anwendungen haben gleich einen doppelten Nutzen durch ihre mobilen Anwendungen, da diese erstens sehr gern genutzt werden und zugleich jegliche Gesundheitsdaten der Nutzer einsehen und verwerten können. Diese Tatsache der Datennutzung spielt für das Thema Datenschutz weiterhin eine wichtige Rolle, da die wenigsten Nutzer darüber informiert sind was genau mit ihren Daten passiert. In Kapitel 7.1 wird dieses Thema noch explizit diskutiert werden.

2.1.6 Ausblick

Für die Weiterentwicklung des Betriebssystems *Android Wear* wäre es erstrebenswert, dass neue vor allem aber nützliche Funktionen konzipiert werden, wie beispielsweise Uhren, die unabhängig vom Smartphone betrieben werden können oder etwa mit einer Anruffunktion durch einen SIM-Karten Slot versehen werden. Die mögliche direkte WLAN-Anbindung für die geeigneten Smartwatches wurde bereits vor wenigen Wochen vorgestellt. Je weiter der Fortschritt in der Entwicklung der Smartwatches ist, desto höher werden die Erwartungen bezüglich neuer Funktionen.

2.2 Marktanalyse von mobilen Fitness-Anwendungen

Da, wie schon erwähnt, mehrere Anbieter ihre eigenen Fitness-Anwendungen auf den Markt gebracht haben, vergleicht dieser Teil der Arbeit nun ausgewählte, bereits veröffentlichte Anwendungen. Diese werden genau beleuchtet und deren einzelne Funktionen aufgeführt. Zu guter Letzt werden die erarbeiteten Informationen gegenübergestellt und ausgewertet. Bei der Analyse der Anwendungen werden neben deren Funktionalität auch die technischen Aspekte der Anwendungen, wie die Verwendung von Sensoren, begutachtet.

2.2.1 Google Fit

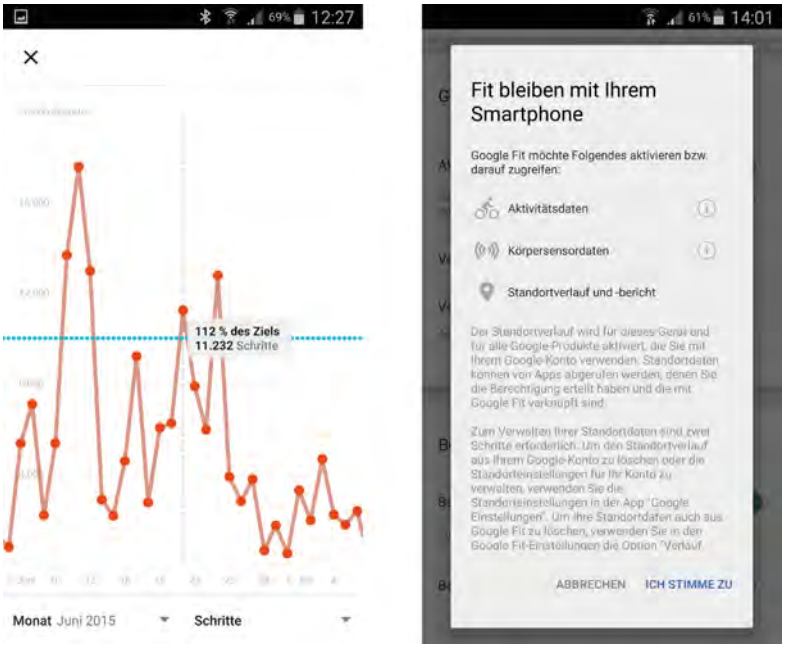

Abbildung 2.3: Google Fit: Ausgabe der erfassten Daten und Zugriffserlaubnis

Google Fit ist eine mobile Anwendung zur Messung und Erfassung der sportlichen Aktivitäten. Durch eine eigene Programmierschnittstelle ermöglicht Google anderen Entwicklern Google Fit als Basis für ihre Fitness-Anwendungen zu verwenden, so dass die erreichten Fitnessziele alle gemeinsam an einem zentralen Ort verwaltet werden können.

Die hauseigene Realisierung bietet einen sehr großen Funktionsumfang. Angefangen bei den Aktivitäten, welche kategorisiert, favorisiert und bearbeitet werden können, so dass beispielsweise Laufen oder Rad fahren als Aktivität definiert werden kann. Features wie Schritte zählen, Angabe des Kalorienverbrauchs, der täglichen körperlichen Aktivität sowie die Körpergewichtskontrolle, können erfasst und auf Wunsch in anschaulichen Diagrammen dargestellt werden (siehe Abbildung 2.3). Die Einstellungsmöglichkeiten gliedern sich in zahlreiche Teilbereiche. Es können Tagesziele in Form von verschiedenen Einheiten definiert werden, welche sich in die Angabe der zurückgelegten Strecke, der ausgeführten Schrittzahl, der Dauer der speziellen Aktivität und den Kalorienverbrauch unterteilen. Unter der Rubrik “Allgemeine Informationen“ können Körpergewicht, Körpergröße und das Geschlecht angeben werden. Die Maß- und Gewichtseinheiten können länderspezifisch angepasst werden. Ein Beispiel hierfür wäre die Auswahl von Kilogramm, Pfund oder Stone bei der Gewichtsangabe. Das Thema Datenschutz spielt auch hier eine wichtige Rolle, da mit der Zustimmung der Aktivitätserkennung Google die Erlaubnis erteilt wird, auf jegliche Sensoren des Gerätes zurückgreifen zu können und so die dort erhobenen Werte zu speichern (siehe Abbildung 2.3). Durch die Zustimmung erhält Google ebenfalls Zugriff auf den vollständigen Standortverlauf. Im Endeffekt bedeutet dies, dass Google jederzeit Zugriff auf die intimen Gesundheitswerte des Nutzers erhält und dazu noch seinen genauen Standort kennt. Allerdings besteht auch die Option den kompletten Aktivitätenverlauf zu löschen, wobei sich da wiederum die Frage stellt, wo genau die Daten gelöscht werden, ob etwa nur auf dem lokalen Gerät oder auch auf dem Google-Server direkt.

2.2.2 Samsung S Health

Samsung hat mit *S Health* eine eher schlichte Anwendungen auf dem Markt gebracht. Die Oberfläche wirkt in erster Linie sehr überschaubar, was natürlich keinen Einfluss auf den relativ großen Funktionsumfang hat. Neben den Funktionen, welche auch von Google angeboten werden, besteht die Möglichkeit Angaben zum Blutzuckerspiegel, welcher nicht direkt mit dem Smartphone gemessen werden kann, zu machen.

Eine Alternative zur Messung des Blutzuckerspiegels wäre die Nutzung von externen Sensoren, sogenannter Vital Sensors, die es ermöglichen die erhobenen Daten auf das Smartphone zu übertragen [SSP+13]. Ebenfalls ergibt sich die Gelegenheit die UV-Einstrahlung sowie den Stressfaktor zu messen. Das Protokollieren des täglichen Wasser- sowie Koffeinkonsums ist im Übrigen auch möglich (siehe Abbildung 2.4). Sportanfängern wird angeboten, gewisse Programme zu absolvieren. Diese sollen beispielsweise die Vorbereitung auf einen 5-km Lauf unterstützen. Hierbei handelt es sich um eine bestimmte Anzahl von Trainingseinheiten zur Vorbereitung für den bereits erwähnten Lauf. Selbstverständlich lassen sich mit dieser mobilen Anwendung auch die gewünschten Trainingsziele definieren, was für das angestrebte Training einen enormen Motivationsschub bedeuten kann. Interessant in diesem Zusammenhang sind auch die Schlafgewohnheiten des Nutzers, diese können durch eine spezielle Option aufgezeichnet werden und stehen somit auch zu einer Auswertung zur Verfügung (siehe Abbildung 2.4).

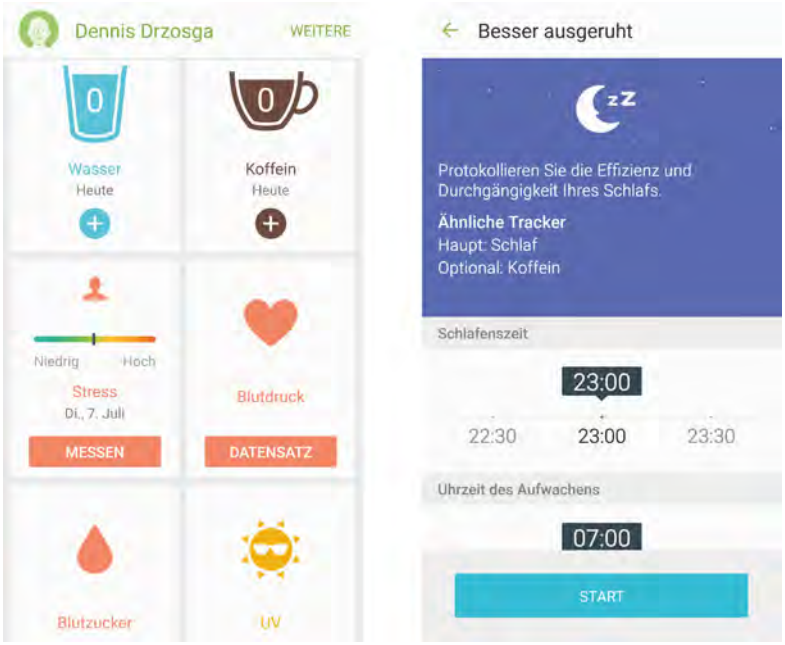

Abbildung 2.4: Samsung S Health: Übersicht ausgewählter Funktionen und Protokollierung der Schlafdauer

2.2.3 Motorola Connect/ Motorola Body

Die von Motorola vorgestellte Lösung *Motorola Connect* besitzt den geringsten Funktionsumfang. Die Funktionalität liegt, im Gegensatz zu den anderen Anwendungen, direkt auf der Uhr, so dass dort, wie bereits auch in den vorgestellten Fitness-Anwendungen, die Herzfrequenz gemessen, die Schritte gezählt, die Herzaktivität sowie der aktuelle Kalorienverbrauch angezeigt werden können. Die erhobenen Daten werden übersichtlich, aber explizit auf der Smartwatch angezeigt (siehe Abbildung 2.5). Eine weitere Möglichkeit zur Einsicht der Daten besteht nur mittels einer wöchentlich verschickten E-Mail mit den darin ausgewerteten Daten.

Abbildung 2.5: Moto Body: Kalorienmessung auf der Smartwatch

2.2.4 Vergleich

Zusammengefasst bietet Samsung wohl den größten Umfang an Funktionen an. Mit einem Marktanteil von 24,5% im Jahr 2014 [Sta15b] ist Samsung zwar unangefochten Marktführer, aber durch die enorme Dominanz des Android Betriebssystems [Sta15a] besteht auch für Google die Chance, durch die hauseigene Fitness-App, den Markt für sich zu entscheiden. Eine Betrachtung der Downloads im Google Play Store zeigt, dass Samsung mit 50 Millionen Downloads klar die Nummer 1 ist. Überraschenderweise kann Motorola, mit einer fast nicht zu vergleichenden mobilen Anwendung, Google, was die Anzahl der Downloads, betrifft durchaus die Stirn bieten (siehe Abbildung 2.6).

	Google Fit	S Health	Motorola Connect
Play Store Downloads	5 Millionen	50 Millionen	5 Millionen
Play Store Bewertung	4,0	3,3	4,2
Grundfunktionen	ja	ja	ja
Auswertungen	ja	ja	ja
technische Aspekte	hauptsächlich durch verfügbare API	Heart Rate Sensor,Standortverlauf	Heart Rate Sensor, Standortverlauf
weitere Features	ja	ja	nein

Tabelle 2.1: Vergleich der vorgestellten Anwendungen

Immerhin belaufen sich bei beiden Anbietern die Downloads auf 5 Millionen. Motorola kann höchstwahrscheinlich durch seine eigene Smartwatch punkten, da Google selbst, neben dem Android Wear Betriebssystem, noch nicht mit einer Smartwatch auf dem Markt vertreten ist. Wie Tabelle 2.1 zeigt, konzentriert sich Motorola auch nur auf die Grundfunktionen wie die Pulsmessung, den Kalorienvebrauch, die Aktivitätenmessung und den Schrittzähler. Vergleicht man die mobilen Anwendung auf deren Verwendung einzelnen Sensoren, so bietet Google Fit, dank der bereits erwähnten Programmierschnittstelle, eine optimale Ausgangsposition für die Verwendung der Sensoren jeglicher Art. Ebenfalls bietet *Google Fit* auch eine angepasste Variante der mobilen Anwendung für *Android Wear* an. Samsung hingegen beschränkt sich mit der *S Health* Anwendung auf die am Smartphone angebrachten Sensoren, so kann die Herzfrequenz bei neueren Samsung Modellen, mit einem direkt an der Kamera angebrachten Sensor gemessen werden. Motorola bietet durch die Implementierung einer unabhängigen *Android Wear* Anwendung nur Messungen an, welche durch die *Moto 360*, also Herzfrequenzmessung und Schrittzähler, ermöglicht werden.

Durch den von Google aufgezeichneten Standortverlauf wird allen Anwendungen mit Hilfe der zurückgelegten Strecke ermöglicht, die Berechnung weiterer Werte, wie Kalorienverbrauch, durchzuführen.

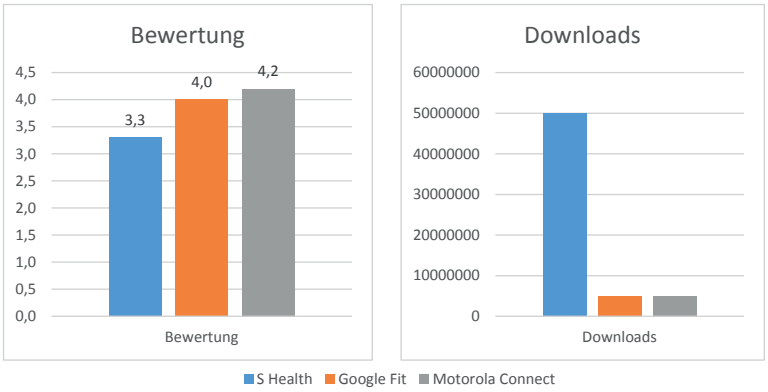

Abbildung 2.6: Darstellung der Downloads und Bewertungen

3

Anforderungen

Softwareanforderungen, welche unterteilt sind in funktionale und nicht-funktionale Anforderungen, sind die Basis aller Softwaredesignentscheidungen. Die funktionalen Anforderungen definieren dabei den Leistungsumfang bzw. die Leistungsbeschreibung der zu entwickelnden Software. Dazu zählen Operationen und Methoden, die das System ausführen und verarbeiten können muss. Die funktionalen Anforderungen fließen vollständig in die Implementierung der Anwendung ein. Sie umfassen alle vom Anwender erwarteten Funktionen der zu entwickelnden Software.

Anforderungen, welche keine speziellen Funktionen des System darstellen, definieren die nicht-funktionalen Anforderungen. Softwareeigenschaften wie Zuverlässigkeit, Reaktionszeit, Akkuverbrauch und Speicherbedarf sind einige Beispiele für nicht-funktionalen Anforderungen. Da sich viele der nicht-funktionalen Anforderungen viel mehr auf das zusammenhängende Framework als auf einzelne Eigenschaften beziehen, werden sie

oft als die entscheidendere Größe im Bezug auf Softwarearchitektur gewertet. Alle nicht-funktionalen Anforderungen, welche für den geplanten Anwendungszweck als notwendig vorausgesetzt wurde, sichern so die Stabilität des ganzen Systems [Mor04].

3.1 Funktionale Anforderungen

Die funktionalen Anforderungen (kurz FA.) des Frameworks beschreiben, wie bereits erwähnt, die Operationen sowie Methoden, die es zu erfüllen hat. Es werden genau diese Methoden als Anforderung definiert, welche das fertige Framework nach der Fertigstellung repräsentieren. Der Benutzer kann bei der späteren Verwendung dieses Frameworks davon ausgehen, dass die Funktionen ihre Aufgaben reibungslos erfüllen.

#1 FA: Herzfrequenzmessung
Das Framework soll in regelmäßigen Abständen, welche durch Zeitintervalle definiert sind, die Herzfrequenz messen. und diese je nach Vorgabe des Benutzers lokal auf dem Smartphone oder auf einem Webserver speichern.

#2 FA: Auswertungen
Die Auswertungen sollen, wenn möglich immer abrufbar sein und leserlich in Diagrammen oder Listen dargestellt werden. Es sollte nach Angabe des Alters des Nutzers ersichtlich sein, ob sich die gemessene Herzfrequenz im Normbereich bewegt oder ob etwa Unregelmäßigkeiten auftreten.

#3 FA: Sicherungen
Es soll dem Nutzer ebenso ermöglicht werden, bereits gemessene Werte separat zu exportieren, so dass auch nach einem möglichen Geräte- bzw. Serverabsturz die bereits erhobenen Daten weiterhin zur Verfügung stehen.

#4 FA: Einstellungen
Das Framework soll über eine Auswahl an Einstellungsmöglichkeiten verfügen, welche man als Nutzer selbst festlegen kann. Beispiele hierfür wären, der Speicherort der

Messungen sowie die Zeitintervalle zwischen den einzelnen Messungen.

#5 FA: Lokalisierung
Die Lokalisierung des Nutzers soll, sofern das GPS-Modul auf dem Smartphone aktiviert ist, gewährleistet sein und bei jeder Messung den aktuellen Standort in Form des geographischen Längen- sowie Breitengrades ausgeben.

#6 FA: Speicherung der Daten in einer Datenbank
Dem Framework soll es ermöglicht werden die erhobenen Datensätze in einer lokalen Datenbank abzulegen.

#7 FA: Speicherung der Daten auf einem Webserver
Ebenfalls soll, wie bei der Speicherung auf der Datenbank, als Alternative ein Webservice zur Verfügung stehen, welcher die erhobenen Daten auf Wunsch dort ablegt.

#8 FA: Sicherstellung der regelmäßigen Überprüfung
Durch diverse Abfragen in der Implementierung soll die regelmäßige, eine in bestimmten Zeitintervallen festgelegte, Abfrage der Herzfrequenz ermöglicht werden, so dass eine qualitative Auswertung vorgenommen werden kann.

3.2 Nicht-funktionale Anforderungen

Unter den nicht-funktionalen Anforderungen (kurz NFA.) des Frameworks werden die Softwareeigenschaften beschrieben wie Stromverbrauch, Performance und Verfügbarkeit. Nicht-funktionale Anforderungen werden benötigt, da diese die Brauchbarkeit eines gesamten Softwaresystems sichern.

#1 NFA: Stromverbrauch
Das Framework soll so ausgelegt sein, dass es einen möglichst geringen Stromverbrauch aufweist. Dies bedeutet, dass vorhandene Ressourcen nur genutzt werden sollen, wenn diese auch wirklich benötigt werden. Zwischen den einzelnen Messungen sollen

die Ressourcen für andere Anwendungen zur Verfügung stehen.

#2 NFA: Problembehandlung
Sollten Fehler auftreten sind diese so zu behandeln, dass der Nutzer davon ausgehen kann, dass bereits erhobene Daten nach einem aufgetretenen Fehler nicht verloren gehen. Des Weiteren sollen Unregelmäßigkeiten so gehandhabt werden, dass der Nutzer benachrichtigt wird, wenn eine Messung nicht erfolgreich war und er die Möglichkeit hat die Messung zu wiederholen.

#3 NFA: Performance
Unnötige Verzögerungen führen oft zu Unzufriedenheit, deshalb sollen die Funktionen des Frameworks möglichst flüssig und klar strukturiert ablaufen. Die Implementierung der einzelnen Funktionen hat effizient programmiert zu sein.

#4 NFA: Verfügbarkeit
Das Framework soll auf möglichst vielen Versionen des Android Betriebssystem problemlos ausführbar sein. Da aber Android Wear mindestens die Android Version 4.3 Jelly Bean (API Level 18) benötigt, werden lediglich rund 55% der Android-Geräte unterstützt.

#5 NFA: Benutzerfreundlichkeit
Ebenso soll das Framework intuitiv bedienbar sein. Es sollen keine langen Einarbeitungszeiten nötig sein, so dass nach ein paar benötigten Einstellungen das Framework sofort verwendet werden kann. Auch das Menü soll verständlich und klar strukturiert sein.

4

Architektur

Die in Abbildung 4.2 gezeigte Softwarearchitektur des Frameworks soll die einzelnen Komponenten beschreiben sowie deren Zusammenspiel innerhalb des Frameworks darstellen. Die Architektur stellt den gesamten Entwurf eines Softwaresystems dar und ist durch Treffen einiger grundlegenden Entscheidungen der verwendeten Komponenten der Ausgangspunkt für die darauffolgende Implementierung.

Im Rahmen dieser Arbeit soll ein Framework entwickelt werden, welches die Möglichkeit bietet über den Herzfrequenz-Sensor einer Android Wear Smartwatch die Herzfrequenz in regelmäßigen Abständen auszulesen. Um dieses Framework anzuwenden, wird eine aktive Verbindung zu einem Android Smartphone mit dem API Level 18 (Android 4.3 Jelly Bean) benötigt. Von Google wird dies vorausgesetzt, um eine erfolgreiche Verbindung mit einer Smartwatch aufzubauen. Die steuernde Einheit des Frameworks ist das Smartphone, weil die Smartwatch nur über einen sehr leistungsschwachen Akku verfügt und

somit dort ressourcenschonend gearbeitet werden muss. Nachdem die Herzfrequenz gemessen wurde, besteht die Möglichkeit diese auf einer lokalen Datenbank auf dem Smartphone oder auf einen externen Webserver abzuspeichern. Die gemessenen Werte sind dann wiederum auf dem Smartphone abrufbar und zur Auswertung einsehbar.

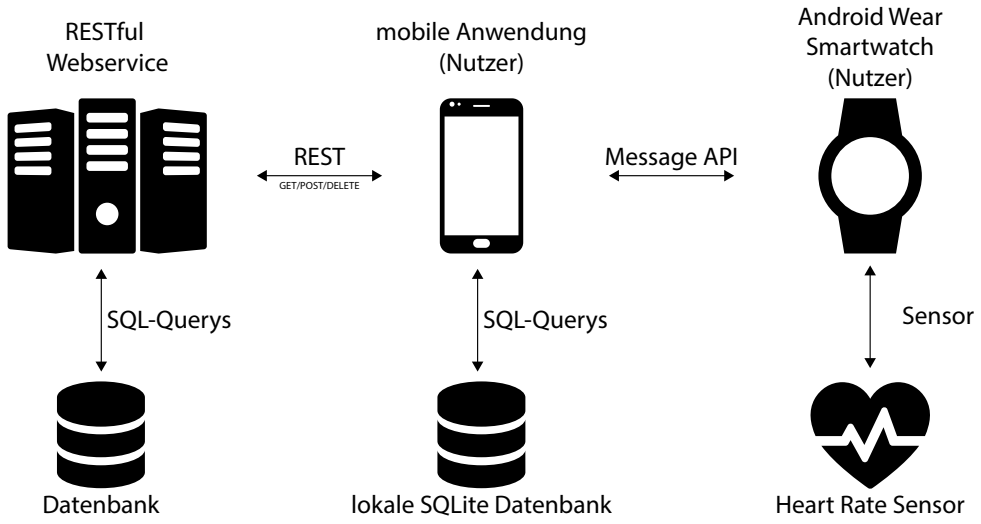

Abbildung 4.1: Architektur des Frameworks

4.1 Smartphone

Das Smartphone spielt in dieser Arbeit die zentrale Rolle, da es alle möglichen Abläufe des Frameworks steuert und mögliche Optionen verwaltet. Der Benutzer selbst kann bestimmen, ob die erhobenen Daten lokal auf dem Gerät oder auf einem Webserver abgespeichert werden sollen. Die bereits gemessenen Datensätze stehen dem Nutzer ebenfalls zur Verfügung und können anschaulich in Diagrammen eingesehen und exportiert werden. Dafür wird die Open-Source Library MPAndroidChart verwendet [Jah15].

Nach Angabe des persönlichen Alters und Gesundheitszustands, wird angezeigt in welchem gesundheitlich angemessenen Rahmen sich der Ruhepuls befinden sollte [Gmb15]. Zu guter Letzt verfügt die mobile Anwendung auch noch über eine Backup-Option, welche die erhobenen Daten bzw. Einstellungen als JSON-Datei exportieren kann.

4.2 Smartwatch

Die Aufgabe der Smartwatch bezieht sich in erster Linie darauf, nach Aufforderung durch das Smartphone, die Herzfrequenz in regelmäßigen Abständen zu messen und diese wiederum an das Smartphone zu übertragen. Dies geschieht durch die von Google speziell dafür entwickelte Message API, welche zusammen mit dem Betriebssystem *Android Wear* veröffentlicht wurde. Da, wie schon erwähnt, der Akku einer Smartwatch durch die geringe Kapazität nicht so leistungsfähig ist, wurden die wesentlichen Funktionen auf dem Smartphone implementiert. Während einer Testphase, bei dem Funktionsaufrufe direkt auf der Uhr getestet wurden, erwies sich der Akku als zu schwach, so dass dies für Endnutzer im Endeffekt nicht tragbar ist.

4.3 Datenbank

Bei den Datenbanken wird lokal auf die von Android vorgegebene SQLite Datenbank zurückgegriffen [SQL15]. Dort können mittels der CRUD (**C**reate **R**ead **U**pdate **D**elete)-Funktionen die einzelnen Datensätze erstellt, editiert, aktualisiert sowie gelöscht werden. Auf der Seite des Webservice ist ebenfalls eine Datenbank zum Verwalten der Datensätze in Betrieb, allerdings wird dort auf die Schema-freie und dokumentenorientierte Open-Source Datenbank MongoDB gesetzt [Mon15].

4.4 Webservice

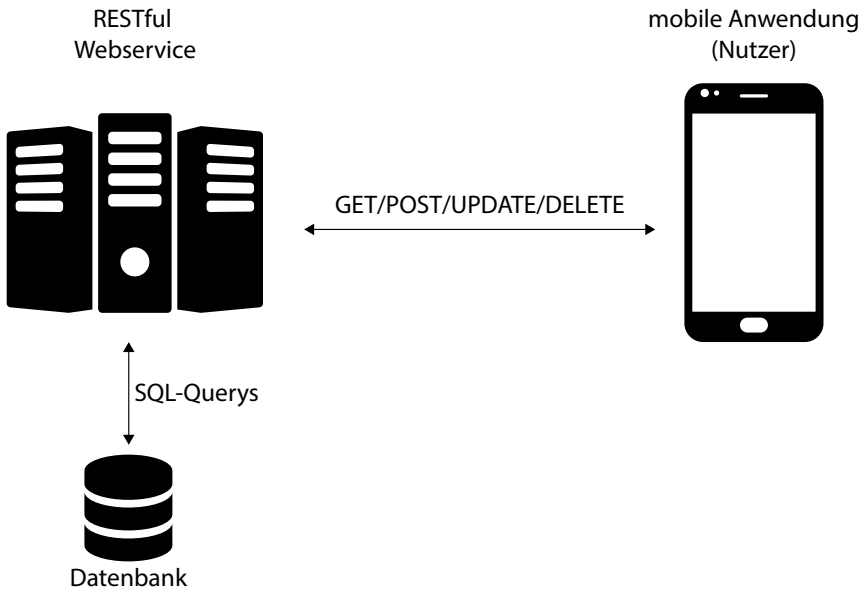

Abbildung 4.2: Webservice

Der RESTful Webservice bietet neben dem Smartphone ebenso eine Möglichkeit die gemessenen Datensätze zu verwalten. Mittels des sogenannten REST (**Re**presentational **S**tate **T**ransfer) verfügt man dort über ähnlich funktionierenden Methoden, um die Daten zu speichern, abzurufen oder zu löschen. Der größte Vorteil der webbasierten Lösung zur Speicherung der Daten liegt vor allem darin, dass der Benutzer bei einer gewünschten Überwachung die Daten, auch ohne weiteren Aufwand, direkt an eine Vertrauensperson, beispielsweise einen Arzt, weiterleiten kann. Dieser hat durch eine webbasierte Oberfläche Einsicht in die Daten und kann somit Unregelmäßigkeiten feststellen und demzufolge handeln. Realisiert wurde der gesamte Service mit Node.JS [Nod15]. Die verwendete Datenbank welche in diesem Fall verwendet wird nennt sich MongoDB [Mon15].

5

Implementierung

Dieses Kapitel beschreibt die gesamte Implementierung des Frameworks. Da das Framework aus mehreren Soft- und Hardwarekomponenten besteht, werden diese explizit betrachtet und wichtige Bestandteile erläutert sowie die Abläufe grafisch dargestellt. Das Erstellen von auswertbaren Datensätzen soll durch die Implementierung ebenfalls gesichert werden. Ein erhobener Datensatz, welcher mehrere Eigenschaften besitzt, besteht aus einer eindeutigen ID, einer gemessenen Herzfrequenz, der zur Messung aktuellen Uhrzeit und Standort, sowie einer Beschreibung der aktuellen Aktivität des Nutzers und einer freiwählbaren User-ID.

5.1 Kommunikation zwischen Smartphone und Smartwatch

Damit beide Geräte miteinander kommunizieren können, muss mittels der Message API zu erst eine Verbindung aufgebaut werden. Wie dieser Aufbau der Verbindung und die Kommunikation zustande kommt, wird in diesem Abschnitt der Implementierung aufgezeigt und erklärt.

5.1.1 Nachricht an die Smartwatch senden

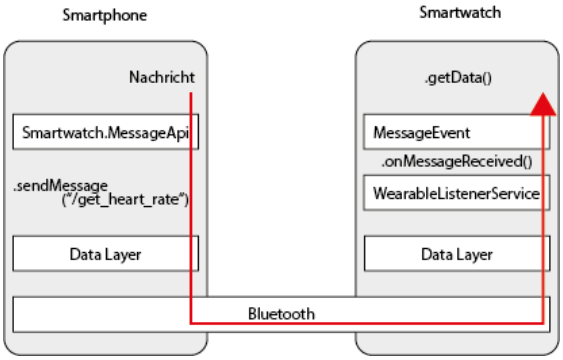

Abbildung 5.1: Nachricht an die Smartwatch senden

Die Kommunikation zwischen den beiden mobilen Geräten erfolgt über die Message API (siehe Abbildung 5.2). Um Nachrichten austauschen zu können muss eine aktive Bluetooth-Verbindung zwischen Smartphone und Smartwatch bestehen. Dies geschieht über den *GoogleApiClient*. Die implementierte Funktion *onConnected(Bundle bundle)* prüft, ob die Verbindung besteht und gibt bestenfalls ID des verbundenen Gerätes zurück. Diese wird benötigt um die Nachricht an den richtigen Empfänger, die Smartwatch, zu übermitteln. Durch die *sendMessage()*-Methode der *Wearable.MessageAPI* wird die Nachricht übermittelt. Auf der Seite der Smartwatch ist ein *WearableListenerService* implementiert welcher über die *onMessageReceived*-Methode die zuvor übermittelte Nachricht empfängt und diese verarbeitet (siehe Listing 5.1).

```
1 public class ListenerService extends WearableListenerService {
2     @Override
3     public void onMessageReceived(MessageEvent messageEvent) {
4         if (messageEvent.getPath().equals("/get_heart_rate")) {
5             // Aufruf der WearMainActivity zur Messung
6             // der Herzfrequenz
7         }
8     }
9 }
```

Listing 5.1: Listener Service

5.1.2 Nachricht an das Smartphone senden

Die gleichen Voraussetzungen zur erfolgreichen Kommunikation zwischen dem Smartphone und der Smartwatch gelten natürlich auch für die Übermittlung einer Nachricht der Smartwatch an das Smartphone. Es wird neben einer aktiven Bluetooth-Verbindung auch ein *ListenerService* benötigt, welcher in Abschnitt 5.2.3 genauer erläutert wird. Der auf beiden Geräten implementierte WearableListenerService reagiert standardmäßig nicht auf die versendeten Nachrichten, sodass zuerst festgelegt werden muss wie die Nachrichten zu verarbeiten sind (siehe Listing 5.1).

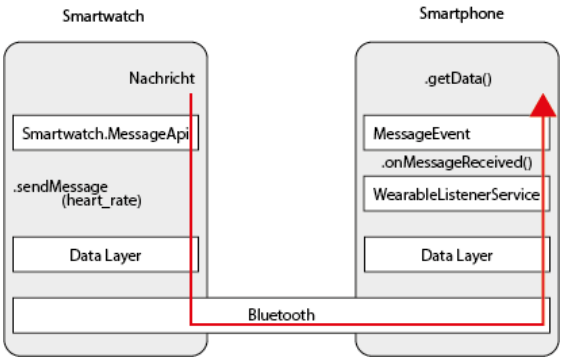

Abbildung 5.2: Nachricht an das Smartphone senden

5.2 Smartphone

Das Smartphone spielt, wie bereits erwähnt, in dieser Arbeit die zentrale Rolle, da die Smartwatch den häufigen Funktionsaufrufen, aufgrund des leistungsschwachen Akkus, nicht standhalten kann. Die möglichen Einstellungen sowie die Speicherung der erhobenen Werte werden ebenfalls auf dem Smartphone ausgeführt.

5.2.1 Einstellungen

Um die mobile Anwendung durch Einstellungen zu personalisieren, wurde eine sogenannte *PreferenceActivity* erstellt. Diese ermöglicht es durch Angabe vorgegebener Auswahlmöglichkeiten das Framework für den persönlichen Gebrauch anzupassen. Durch Angabe eines freigewählten Nutzernamens, kann man so für eine anonyme Speicherung sorgen. Weiterhin kann durch die Angabe des persönlichen Alters, welches in bestimmte Zielgruppen und Zustand des eigenen Wohlbefindens gegliedert ist, der Ruhepuls festgelegt werden. Zudem kann das Intervall angegeben werden, welches die Pausen zwischen den einzelnen Messungen definiert. Zusätzlich kann durch eine Auswahl der Speicherort der erhobenen Daten angegeben werden. Hierbei handelt es sich um die Auswahlmöglichkeiten zwischen einer lokalen Sicherung auf dem Gerät oder einer Sicherung auf einem Webservice. Die genannten Optionen werden vom System in den *SharedPreferences*, welche vom Betriebssystem zur Verfügung gestellt werden, abgespeichert. Durch einen Aufruf können diese dann ausgelesen und zur Konfiguration des Systems verwendet werden.

5.2.2 HrmObject

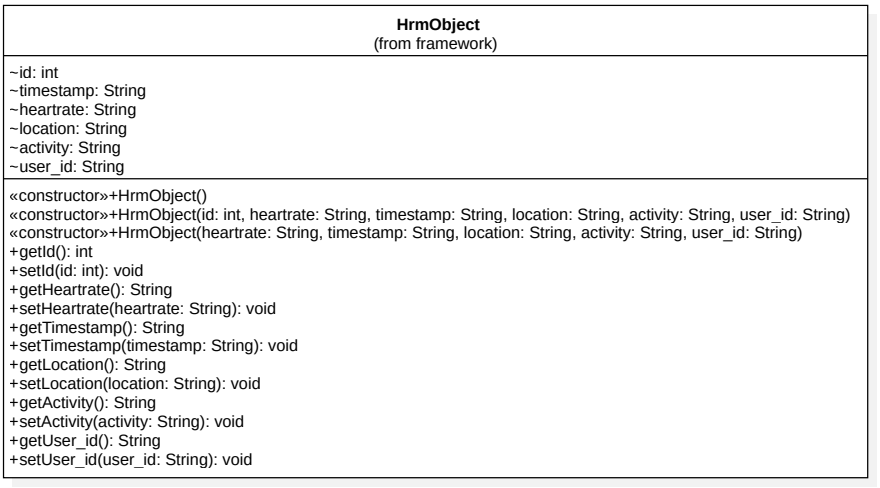

Abbildung 5.3: HrmObject

Das *HrmObject* (siehe Abbildung 5.3) dient in dieser Arbeit als Objekt zur Speicherung der erhobenen Datensätze. Wie aus der bereits aufgeführten Abbildung 5.3 hervorgeht, besitzt das Objekt folgenden Attribute:

id: Wird von der jeweiligen Datenbank gesetzt.
heartrate: Die durch die Smartwatch gemessene Herzfrequenz.
time: Speichert die zum Zeitpunkt der Messung aktuelle Uhrzeit.
location: Speichert die zum Zeitpunkt der Messung aktuelle Position in Form von Breiten- und Längengrad ab.
activity: Gibt die Art der Aktivität des Nutzers zum Zeitpunkt der Messung aus.
user_id: Kann zur eindeutigen Identifikation der Messwerte in den Einstellungen selbstständig angegeben werden.

Das HrmObject (siehe Abbildung 5.3) dient zur vereinfachten Verarbeitung der erhobenen Daten. Durch die klare Struktur des Objekts ist es so möglich alle Daten kompakt in einem Objekt abzulegen und als Datensatz weiterzuverarbeiten.

5.2.3 MobileListenerService

Der *MobileListenerService* empfängt, wie bereits in Kapitel 5.1.2 beschrieben, die gemessene Herzfrequenz und verarbeitet diesen weiter (siehe Abbildung 5.4). Nach Aufruf der Smartwatch wird die aktuelle Uhrzeit ausgelesen und im Format “dd-MM-yyyy hh:mm:ss“ abgespeichert. Ist dies erfolgt, so wird geprüft, ob das GPS-Modul zur Lokalisierung der aktuellen Position aktiviert ist. Sofern dieses aktiviert ist, wird durch die Installation dieser mobilen Anwendung die Erlaubnis erteilt, die aktuelle Position auszulesen. Anderenfalls wird dieser Schritt übersprungen und mit der Bestimmung der Aktivität durch die *ActivityRecognitionApi* fortgeführt. In diesem Schritt wird festgestellt wo sich der Nutzer der Anwendung gerade befindet und sendet bei einer Auswertung der erhobenen Daten eine mögliche Begründung. Die *ActivityRecognitionApi*, welche durch die *DetectedActivity* die aktuelle Aktivität ausgibt, unterscheidet zwischen möglichen Werten wie, im Fahrzeug, auf einem Fahrrad, zu Fuß unterwegs, Stillstand, Rennen, Gehen, sowie eine undefinierte Aktivität. Zu guter Letzt wird die in den Einstellungen angegebene User-ID ausgelesen und mit den erfassten Werten ein neues HrmObject erzeugt. Dieses wird nach Abfrage des gewünschten Speicherplatzes in der lokalen Datenbank oder auf dem Webserver gesichert.

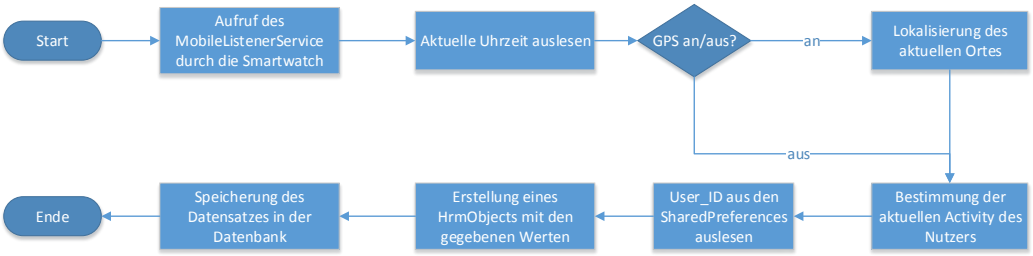

Abbildung 5.4: MobileListenerService

5.2.4 AlarmManager

Um eine regelmäßige Abfrage der Herzfrequenz zu gewährleisten, wird hier auf den *AlarmManager* zurückgegriffen. Dieser ermöglicht es nach einer einmaligen Initialisierung anhand der aktuellen Uhrzeit und dem auszuführenden Service (siehe Listing 5.2), die Abfrage in den gewünschten Zeitintervallen zu wiederholen.

Um eine eindeutige sowie erfolgreiche Auswertung der erhobenen Daten durchführen zu können, wird durch eine Abfrage geprüft, ob der *AlarmManager* bereits initialisiert wurde. Dies verhindert, dass eine erneute Initialisierung statt findet. Würde dieses Szenario eintreten, so wäre zwar das erwünschte Zeitintervall unverändert, doch würde sich bei jedem Start der Anwendung der Timer des *AlarmManagers* zurücksetzen. Demzufolge wäre in diesem Fall eine regelmäßige Abfrage nicht garantiert.

```
1 AlarmManager alarm = (AlarmManager)
2         getSystemService(Context.ALARM_SERVICE);
3 Calendar cal = Calendar.getInstance();
4 Intent intent = new Intent
5         (this, StartHeartRateMeasurement.class);
6 PendingIntent pIntent = PendingIntent
7         .getService(this, 0, intent, 0);
8 alarm.setRepeating(AlarmManager.RTC,
9         cal.getTimeInMillis(),intervall,pIntent);
```

Listing 5.2: Initialisierung des AlarmManagers

5.2.5 Datenbank

Um die erhobenen Daten lokal zu speichern, wird auf eine SQLite-Datenbank zurückgegriffen. Die lokale Sicherung der Daten hat den Vorteil, dass keine aktive Internetverbindung zur Sicherung benötigt wird, da sich die Datenbank auf dem Smartphone befindet. Zudem besteht aufgrund der Offline-Verfügbarkeit der Datensätze jederzeit

die Möglichkeit diese auswerten zu können. Der Aufbau der Datenbank besteht aus einer Tabelle *Records*, wo die einzelnen Datensätze abgespeichert werden (siehe Abbildung 5.5). Die im Datenbankmodell angegebenen Attribute findet man auch in der Klasse *HrmObject* (siehe Kapitel 5.2.2) wieder. Dort wird bei der Erstellung eines jeden Datensatzes ein Objekt erstellt, welches die erhobenen Werte enthält. In der Klasse *MyDatabaseHandler* befinden sich alle nötigen Methoden zur Erstellung und Verwaltung der Datenbank. Die Methode *onCreate(SQLiteDatabase db)* erstellt die Datenbank nach dem Aufruf der *MyDatabaseHandler*-Klasse. Die Methoden *addHrmObject(HrmObject hrmObject)*, *List<HrmObject> getAllHrmObjects()* und *deleteHrmObject(HrmObject hrmObject)* dienen zur Verarbeitung der erhobenen Daten.

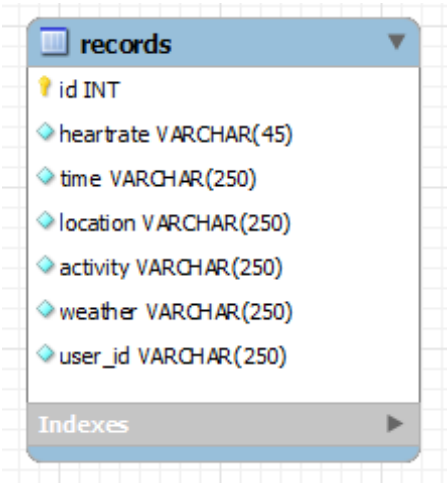

Abbildung 5.5: Datenbankmodell

5.3 Smartwatch

Die Smartwatch dient in diesem Framework zur Messung der Herzfrequenz. Durch den verbauten Sensor, der sich auf der Unterseite der Uhr befindet, wird sofern eine aktive Bluetooth-Verbindung zum Smartphone besteht, jederzeit eine Messung der Herzfrequenz (siehe Abbildung 5.6) ermöglicht. Als Entwicklungsgerät wurde in dieser Arbeit die Smartwatch Moto 360 von Motorola verwendet.

Abbildung 5.6: Herzfrequenz-Sensor

5.3.1 MainWearActivity

```
1 mSensorManager = ((SensorManager)
2                   getSystemService(SENSOR_SERVICE));
```

Listing 5.3: Sensor Manager

In der *MainWearActivity* findet nach Aufruf durch den ListenerService (siehe Kapitel 5.2) die Herzfrequenzmessung statt. Um Zugriff auf den Sensor zu erhalten, muss auf den *SensorManager* zurückgegriffen werden. Nach der Initialisierung des *SensorManagers* besteht die Möglichkeit auf alle vorhandenen Sensoren des Gerätes zuzugreifen. Die Moto 360 Smartwatch, welche in dieser Arbeit verwendet wird, bietet neben dem Heart Rate Sensor auch einen Schrittzähler, sowie einen Lichtsensor zur Regulierung der Displayhelligkeit an. Um Zugriff auf die Sensoren zu erhalten, muss die Berechtigung *BODY_SENSORS* im *AndroidManifest* angegeben werden. Das AndroidManifest ist ein wichtiger Bestandteil jeder mobilen Android Anwendung, dort werden alle in der App geforderten Berechtigungen sowie Name, Icon und weitere notwendige Eigenschaften einer mobilen Anwendung deklariert. Durch diese Angaben erhält der Benutzer noch vor der Installation einen Überblick über die geforderten Rechte der Anwendung, so dass entschieden werden kann, ob dies überhaupt gewünscht ist. Nachdem die Berechtigung vermerkt wurde, kann nun mittels des *SensorManagers* auf den Heart Rate Sensor

zugegriffen werden. Hier gilt allerdings zu beachten, dass verschiedene Typen des Sensors vorhanden sind. Beachten sollte man hier, dass sich die IDs der Sensoren je nach Hersteller ändern. Beispielsweise besitzt der Sensor der *Moto 360* von Motorola die ID *65538*, die Samsung Gear Live hingegen die ID *65562*. Die IDs aber ändern sich nicht untereinander, so dass beispielsweise jede *Moto 360* die gleiche ID besitzt. Um die Herzfrequenz messen zu können, muss auf das Interface *SensorEventListener* (siehe Listing 5.4) zurückgegriffen werden, welches die Methoden *onSensorChanged(SensorEvent sensorEvent)* und *onAccuracyChanged(Sensor sensor, int i)* mitbringt. Die erste der beiden Methoden dient zur Ausgabe der gemessenen Herzfrequenz, sobald diese gemessen ist oder der Wert der Herzfrequenz sich verändert hat. Diese Methode dient auch in dieser Arbeit zur Messung, so dass im Anschluss der erhobene Wert ausgelesen und via Message API (siehe 5.1) an das Smartphone via *sendMessageToSmartphone(String message)* an das Smartphone zur weiteren Verarbeitung übergeben werden kann.

```
1  public class MainWearActivity extends Activity
2          implements SensorEventListener {
3
4          @Override
5      public void onSensorChanged(final SensorEvent sensorEvent) {
6          ...
7          // Herzfrequenz aus SensorEvent auslesen
8          int heartRate = Math.round(sensorEvent.values[0]);
9          ...
10     }
11
12     @Override
13     public void onAccuracyChanged(Sensor sensor, int i) {
14         ...
15     }
16 }
```

Listing 5.4: SensorEventListener

5.4 Webservice

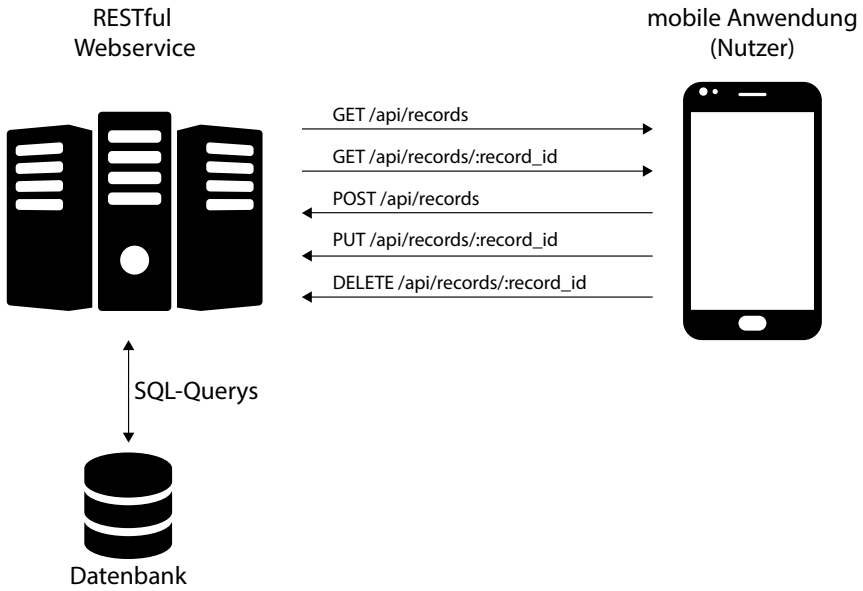

Abbildung 5.7: REST-API zur Kommunikation zwischen Smartphone und Webservice

Der Webservice steht dem, als alternative Speichervariante zu der bereits bekannten lokalen Speicherung, zur Verfügung. Sofern der Nutzer wünscht die erhobenen Daten im Webservice zu speichern, so kann dies in den Einstellungen vorgenommen werden.

Die Kommunikation zwischen Smartphone und Webservice erfolgt über eine REST-API, welche die Daten im JSON-Format zur Verfügung stellt. Wie aus Abbildung 5.7 entnommen werden kann, stellt die Schnittstelle, die bereits in Kapitel 4 erwähnten CRUD-Funktionen, zur Verfügung. HTTP-Request in Form von GET-Aufrufen geben bestehende Datensätze zurück, ohne diese zu verändern. PUT-, POST-, und DELETE-Aufrufe hingegen ermöglichen durch Aufruf implementierter Methoden, das Bearbeiten, Erstellen sowie Löschen von Daten.

Die Datenbank *MongoDB*, welche auf dem Webservice verwendet wird, besteht nicht, wie die aus den SQLite-, oder MySQL-Datenbank bekannten Tabellen sondern, aus sogenannten Collections. Dort werden einzelne Einträge als JSON-Objekte abgelegt und liegen so schon im gewünschten Format zur Abfrage vor.

6

Anforderungsabgleich

In diesem Kapitel der Arbeit werden die zu Beginn definierten (siehe Kapitel 3) funktionalen und nicht-funktionalen Anforderungen auf ihre tatsächliche Umsetzung überprüft. Dies wird mittels einer Tabelle abgeglichen, so dass ersichtlich ist, inwieweit diese erfüllt wurden. ’1’ bedeutet hierbei ’sehr gut’ und ’6’ ’ungenügend’ - entsprechend dem üblichen Schulnotensystem. Um die Bewertungen besser nachvollziehen zu können, werden die einzelnen Punkte nochmals genauer erläutert.

6.1 Abgleich der funktionalen Anforderungen

Anforderung	Bewertung	Beschreibung
Herzfrequenzmessung	1	Die Messung der Herzfrequenz auf der Smartwatch ist durch den Heart Rate Sensor sichergestellt und kann somit jederzeit zur Verfügung gestellt werden.
Auswertungen	2	Auswertungen werden durch die Third Party Library *MPAndroidChart* dargestellt und bieten die Option diese Diagramme auch als Bild zu exportieren. Natürlich kann die Auswertung um einige Optionen erweitert werden, so dass die Auswertung durch mehrere Variablen beeinflusst werden kann.
Sicherungen	1	Die mögliche Sicherung der erhobenen Daten wird ermöglicht und zuverlässig als JSON-File exportiert.
Einstellungen	1	Dem Nutzer wird ermöglicht, Einstellungen vorzunehmen und diese zu speichern.
Lokalisierung	1	Über das aktivierte GPS-Modul kann die aktuelle Position des Nutzer zum Zeitpunkt der Messung bestimmt werden.
Speicherung der Daten in einer Datenbank	1	Die Sicherung der Daten lokal auf dem Gerät sind gewährleistet und können verarbeitet werden.
Speicherung der Daten auf einem Webserver	1	Als Alternative zur lokalen Sicherung der erhobenen Daten dient der Webservice welcher alle Funktionen besitzt um die erhobenen Daten zu verarbeiten.
Sicherstellung der regelmäßigen Überprüfung	1	Die durch den *AlarmManager* gesetzt Zeitintervalle werden regelmäßig abgefragt und somit eine qualitative Auswertung ermöglicht.

6.2 Abgleich der nichtfunktionalen Anforderungen

Anforderung	Bewertung	Beschreibung
Stromverbrauch	1	Da die Akkuleistung der Smartwatch sehr gering ist, wurden alle Funktionen, die nicht direkt auf der Smartwatch ausgeführt werden müssen ausgelagert und somit die Laufzeit des Akkus geschont.
Problembehandlung	2	Mögliche auftretende Fehler werden ordnungsgemäß abgefangen und behandelt. Durch die große Auswahl an Anwendungen kann es durchaus vorkommen das die Anwendung aus unerklärlichen Gründen beendet wird. Durch die eine regelmäßige Überprüfung wird versucht, die Funktionalität der entwickelten Anwendung sicherzustellen.
Performance	1	Die Anwendung läuft absolut flüssig, die Prüfung der Herzfrequenz läuft ohne unerwartete Verzögerung ab.
Verfügbarkeit	1	Da Android für die Nutzung einer Android Wear basierten Smartwatch Android 4.3 erwartet, wurde die Anwendung auf diesem API Level entwickelt und erfüllt somit das Kriterium.
Benutzerfreundlichkeit	2	Es wurde viel Wert draufgelegt, eine intuitive Benutzeroberfläche zu gestalten, so dass der Nutzer durch eine sehr kurze Eingewöhnungszeit die Anwendung eigenständig bedienen kann.

7

Zusammenfassung und Ausblick

Das Resultat dieser Arbeit, die Konzeption und Realisierung eines Heart Rate Monitoring Framework für *Android Wear* wird in diesen Kapitel zusammenfassend formuliert. Ebenso wird über das Thema Datenschutz, welches im Rahmen dieser Arbeit eine bedeutende Rolle spielt diskutiert. Des Weiteren wird ein Ausblick gegeben, wie das Framework in Zukunft erweitert werden kann und wie eine Smartwatch durch neue Sensoren und Technologien die Gesundheit des Nutzers sicherstellen kann.

7.1 Diskussion zum Thema Datenschutz

Werden Daten, wie beispielsweise der aktuelle Standort oder die eigene Herzfrequenz, in einer Anwendung gespeichert, so ist das Thema Datenschutz unumgänglich. Viele der Nutzer sind in den meisten Fällen ahnungslos, wie mobile Anwendungen mit ihren

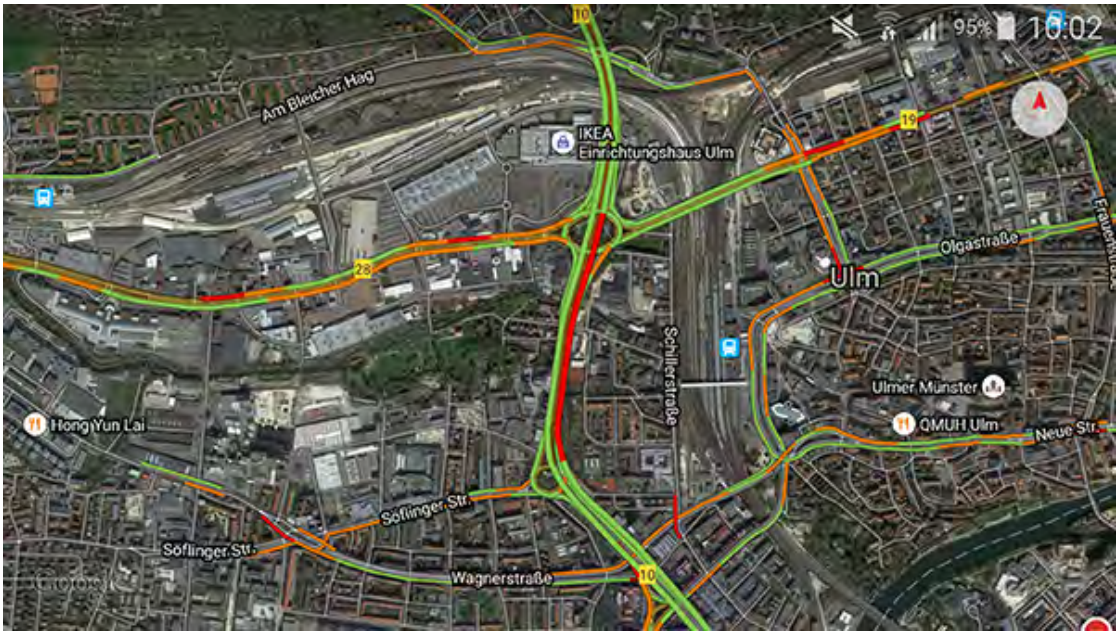

Abbildung 7.1: Google Maps Anwendung mit Anzeige der aktuellen Verkehrssituation

Daten umgehen. Je mehr Berechtigungen eine Anwendung verlangt, desto misstrauischer werden die Nutzer, was schlimmstenfalls zur Verweigerung der Anwendung führt. Besonders sorgen Berechtigungen wie Zugriff auf Nachrichten-, Kontakte, Mikrofon sowie Zugriff auf diverse Systemfunktionalitäten für erhöhtes Misstrauen [SSPR15]. In den meisten Fällen jedoch benötigt eine Anwendung gewisse Informationen, um dem Nutzer eine speziell für ihn angepasste Information darstellen zu können. Ein Beispiel hierfür wäre Google Maps, welches laut Angaben von Google, anonymisierte Daten von Android-Nutzer auswertet und so teilweise minutengenaue Verkehrsinformationen auf einer interaktiven Karte darstellen kann (siehe Abbildung 7.1) [Goo15b]. Dies wird erst möglich, wenn der Nutzer seinen aktuellen Standort zur Verfügung stellt und sich dadurch lokalisieren lässt.

Im Rahmen dieser Arbeit werden die erhobenen Daten nur zur Einsicht des Nutzers gespeichert. Da der Nutzer selbst entscheiden kann, wo die erhobenen Daten gespeichert werden sollen, ist ihm bei einer lokalen Sicherung gewährleistet, dass die erhobenen Daten nicht an Dritte weitergegeben werden. Als Entwickler sollte man dem Nutzer die

Sicherheit geben, dass die Daten vertraulich behandelt werden und nicht unverschlüsselt versendet werden.

7.2 Zusammenfassung

Das Ziel dieser Arbeit, die Konzeption und Realisierung eines Heart Rate Monitoring Framework für Android Wear, wurde mit der Implementierung des gesamten Frameworks erreicht. Durch die Verwendung von *Android Wear* wurden zu Beginn dieser Arbeit die Funktionalitäten des Betriebssystems erläutert. Ebenso wurde eine Analyse ausgewählter Fitness-Anwendungen durchgeführt, in welcher diese auf Funktionalität und technischen Aspekten untersucht wurden.

Im Kapitel 3 wurden die funktionalen sowie nicht-funktionalen Anforderungen, welche es zu erfüllen galt, definiert. Diese Anforderungen wurden benötigt, um schon bei der Entwicklung des Frameworks gewisse Eigenschaften einzuhalten sowie angeforderte Funktionen bereitzustellen. Anschließend folgte die Architektur des Frameworks, wo die einzelnen Komponenten beschrieben und anschaulich dargestellt wurden. Zudem wurde dort der Zusammenhang der einzelnen Komponenten erläutert und die verwendete Software erwähnt.

Die Implementierung des Frameworks fand in Kapitel 5 statt. Dort wurde neben der Kommunikation von Smartphone und Smartwatch durch die Message API auch explizit auf die einzelnen Komponenten eingegangen und deren implementierte Funktionalität erläutert sowie anschaulich dargestellt. Im Kontext der Implementierung bestand die Schwierigkeit darin die regelmäßige Anfrage an die Smartwatch gewährleisten zu können. Zudem traten unerwarteter Weise Probleme mit anderen Anwendungen auf, welche ohne weitere Folgen behoben werden konnten und keinen Einfluss auf die Funktionalität des Framework nehmen. Ebenfalls konnte die Einarbeitung in die Message API nicht ganz ohne Hürden durchgeführt werden, da Fehler aufgetreten sind, wo ursprünglich keine Fehler waren und diese erst durch tagelangen Debugging beseitigt werden konnten. Die verwendete REST-API, welche mit Node.JS und der Datenbank MongoDB

realisiert wurde, konnte erst nach ausgiebiger Recherche realisiert werden und nahm außerplanmäßig viel Zeit in Anspruch.

Das Kapitel *Anforderungsabgleich* bewertet das implementierte und funktionierende Framework mit den in Kapitel 3 festgelegten Anforderungen. Auch hier fällt das Ergebnis sehr positiv aus, da größtenteils alle Anforderungen mit äußerster Zufriedenheit erfüllt wurden. Durch die in der Anwendungen erhobenen Daten, welche im Kontext zu dem Nutzer stehen, führte kein Weg an einer Diskussion zum Thema Datenschutz vorbei. Das Ergebnis fiel allerdings relativ klar aus. Demzufolge möchten die Nutzer so wenig Daten wie nur nötig preisgeben, aber einen vollen Funktionsumfang einer Anwendung genießen. Dies ist, wie im aufgezeigten Beispiel der Anwendung *Google Maps*, nicht möglich. Dadurch wurde der Entschluss gefasst, dass Nutzer eben gewissen Daten preisgeben müssen, auch wenn diese sehr persönlich sind. Nur so können Entwickler Anwendungen entwickeln die dem Nutzer helfen erhobene Daten auszuwerten.

7.3 Ausblick

Im Ausblick auf die Zukunft könnte das im Rahmen dieser Arbeit entwickelte Framework zusätzlich um gewisse Funktionen erweitert werden. Durch die Anbringung neuer Sensoren, wie beispielsweise einem UV-Belichtungssensor, bietet es sich an den UV-Index zu messen um die Nutzer zu benachrichtigen wenn diese ihre Haut schützen sollten. Zudem könnte man sich durch die Anbringung eines Temperatursensors vorstellen, die Körpertemperatur zu beobachten und so, bei einer normwidrigen Änderung, eine Benachrichtigung zu erhalten.

Durch eine Smartwatch, welche mit einem eigenständigen LTE-Modul ausgestattet ist, könnte das realisierte Framework ohne Abhängigkeit eines Smartphones genutzt werden. Dies würde bedeuten, dass der Webservice direkt auf der Uhr ausgeführt werden kann und somit die Daten auf einen beliebigen Server übertragen werden können. Ein bereits gestartetes Crowdfunding Projekt namens *Neptune Suite* bietet eine solche Smartwatch an [Ind15].

Mit Hilfe neuer Akkutechnologien könnte es durchaus vorstellbar sein, Smartwatches mit einer viel längeren Akkulaufzeit auf den Markt zu bringen. Dies würde die intelligenten Uhren für den Endverbraucher noch attraktiver und so die gesundheitsvorbeugenden mobilen Anwendungen populärer machen.

Krankenkassen in den USA zahlen ihren versicherten Kunden einen Bonus, wenn diese erlauben ihre Fitnessdaten aufzeichnen zu lassen [Bre15b]. Durch die Aufzeichnung der Daten animieren die Krankenkassen mit Bonuszahlungen die Nutzer zu mehr sportlicher Aktivität. Dies bietet natürlich Vorteile für beide Parteien. Zum Einen halten sich die Nutzer durch mehr Sport körperlich fit und leben gesünder. Zum Anderen sparen die Krankenkassen eine Menge an Kosten ein, in dem sie die Ausgaben für Medikamente und Therapien für ihre Versicherten deutlich senken können.

Literaturverzeichnis

[BBKS15] BENGEL, Günther ; BAUN, Christian ; KUNZE, Marcel ; STUCKY Karl-Uwe: In: *Masterkurs Parallele und Verteilte Systeme: Grundlagen und Programmierung von Multicore-Prozessoren, Multiprozessoren, Cluster, Grid und Cloud*. Springer-Verlag, 2015

[Bre15a] BREUSTEDT, Hannes: *Smartwatch: Wenn Krankenkassen Fitnessdaten sammeln - DIE WELT*. http://www.welt.de/gesundheit/article137929788/Krankenversicherer-sind-begierig-auf-Fitnessdaten.html, 2015. – (Abruf: 27.07.2015)

[Bre15b] BREUSTEDT, Hannes: *Smartwatch: Wenn Krankenkassen Fitnessdaten sammeln - DIE WELT*. http://www.welt.de/gesundheit/article137929788/Krankenversicherer-sind-begierig-auf-Fitnessdaten.html, 2015. – ((Abruf: 31.07.2015)

[Flo15] FLORIN, Alexander: *User Interface Design - Einstieg und Praxisratgeber*. Books on Demand, 2015

[Gmb15] GMBH, SERVIER D.: *Pulsmessung - Puls richtig messen | Initiative Pulsgesund*. http://www.pulsgesund.de/pulsmessung/, 2015. – (Abruf: 24.07.2015)

[Goo15a] GOOGLE: *Android Wear*. https://www.android.com/wear/, 2015. – (Abruf: 07.07.2015)

[Goo15b] GOOGLE: *Official Google Blog: The bright side of sitting in traffic: Crowdsourcing road congestion data*. http://googleblog.blogspot.de/2009/08/bright-side-of-sitting-in-traffic.html, 2015. – (Abruf: 31.07.2015)

[Goo15c] GOOGLE: *Zeigen Sie Ihren Stil – Android-Apps auf Google Play*. https://play.google.com/store/apps/collection/promotion_3001507_wear_watch_faces_all, 2015. – (Abruf: 07.07.2015)

[Gra01] GRAF, Christine: *Lehrbuch Sportmedizin: Basiswissen, präventive, therapeutische und besondere Aspekte*. Deutscher Ärzte-Verlag, 2001

[Ind15] INDIEGOGO: *Neptune Suite - One Hub, Infinite Possibilities | Indiegogo*. https://www.indiegogo.com/projects/neptune-suite-one-hub-infinite-possibilities#/story, 2015. – (Abruf: 26.07.2015)

[Jah15] JAHODA, Philipp: *MPAndroidChart*. https://github.com/PhilJay/MPAndroidChart, 2015. – (Abruf: 06.07.2015)

[KL15] KRÖSMANN, Christoph ; LUTTER, Timm: *Fit in den Frühling mit digitaler Unterstützung*. https://www.bitkom.org/Presse/Presseinformation/Pressemitteilung_4169.html, 2015. – (Abruf: 26.07.2015)

[Mon15] MONGODB: *MongoDB*. https://www.mongodb.org/, 2015. – (Abruf: 07.07.2015)

[Mor04] MORO, Martin: *Modellbasierte Qualitätsbewertung von Softwaresystemen - Bewertung von Softwarearchitekturen in Bezug auf ihren Erfüllungsgrad der Qualitätsanforderungen*. Books on Demand, 2004

[Nod15] NODE.JS: *Node.js*. https://nodejs.org/, 2015. – (Abruf: 07.07.2015)

[SL11] SCHWEGLER, Johann S. ; LUCIUS, Runhild: *Der Mensch - Anatomie und Physiologie, 5. Auflage*. Thieme-Verlag, 2011

[SPSR15] SCHICKLER, Marc ; PRYSS, Rüdiger ; SCHOBEL, Johannes ; REICHERT, Manfred: An Engine Enabling Location-based Mobile Augmented Reality Applications. Version: 2015. http://dbis.eprints.uni-ulm.de/1137/. In: *Web Information Systems and Technologies - 10th International Conference, WEBIST 2014, Barcelona, Spain, April 3-5, 2014, Revised Selected Papers*. Springer, 2015 (LNBIP)

[SQL15] SQLITE: *SQLite*. https://www.sqlite.org/, 2015. – (Abruf: 06.07.2015)

[SSP+13] SCHOBEL, Johannes ; SCHICKLER, Marc ; PRYSS, Rüdiger ; NIENHAUS, Hans ; REICHERT, Manfred: Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned. In: *9th Int'l Conference on Web Information Systems and Technologies (WEBIST 2013), Special Session on Business Apps*, 2013, 509–518

[SSPR15] SCHOBEL, Johannes ; SCHICKLER, Marc ; PRYSS, Rüdiger ; REICHERT, Manfred: Process-Driven Data Collection with Smart Mobile Devices. Version: 2015. http://dbis.eprints.uni-ulm.de/1136/. In: *Web Information Systems and Technologies - 10th International Conference, WEBIST 2014, Barcelona, Spain, Revised Selected Papers*. Springer, 2015 (LNBIP)

[Sta15a] STATISTA: *Marktanteile Smartphone-Betriebssysteme 2015 und 2019*. http://de.statista.com/statistik/daten/studie/182363/umfrage/prognostizierte-\marktanteile-bei-smartphone-betriebssystemen/, 2015. – (Abruf: 08.07.2015)

[Sta15b] STATISTA: *Smartphone - Marktanteile der Hersteller am Absatz weltweit bis 2014*. http://de.statista.com/statistik/daten/studie/173051/umfrage/weltweite\-martkanteile-der-fuehrenden-smartphone-hersteller-seit-\2007/, 2015. – (Abruf: 08.07.2015)

A

Storyboard

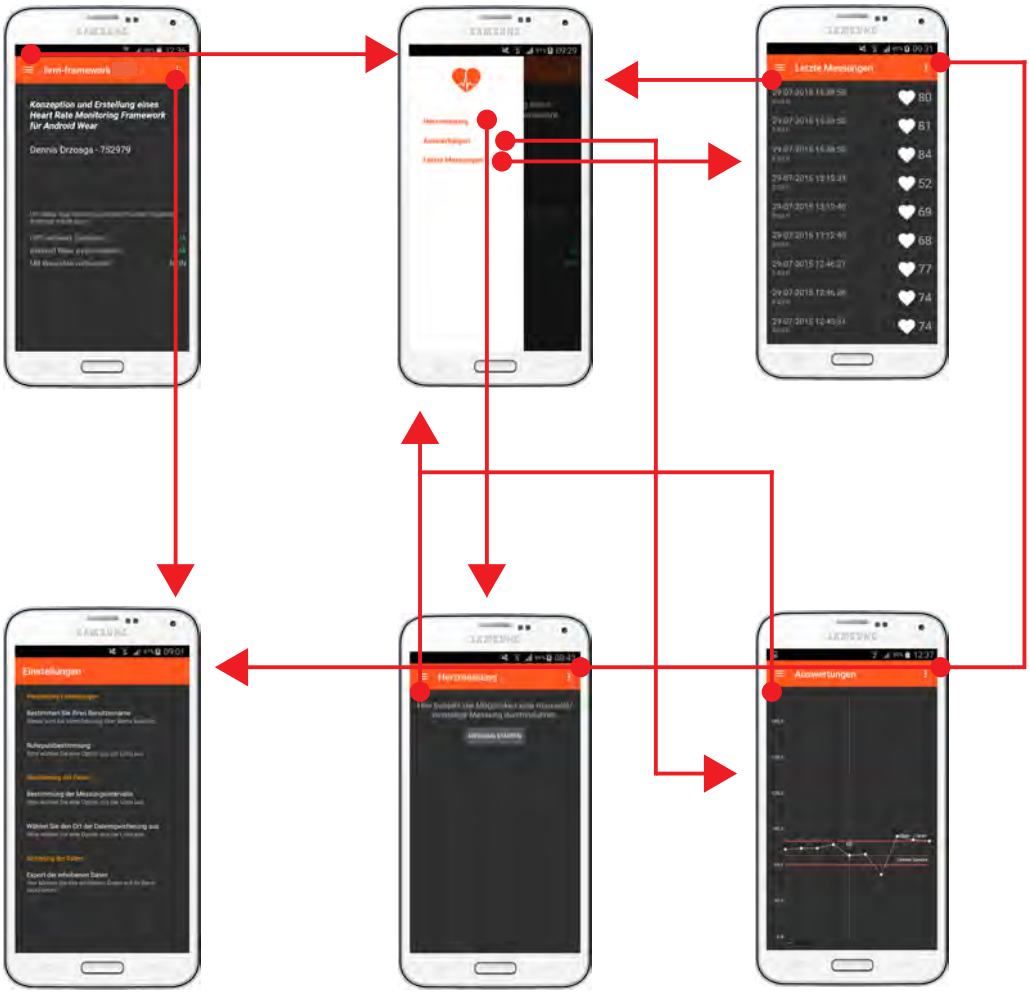

Abbildung A.1: Storyboard der mobilen Anwendung

B

Klassendiagramm

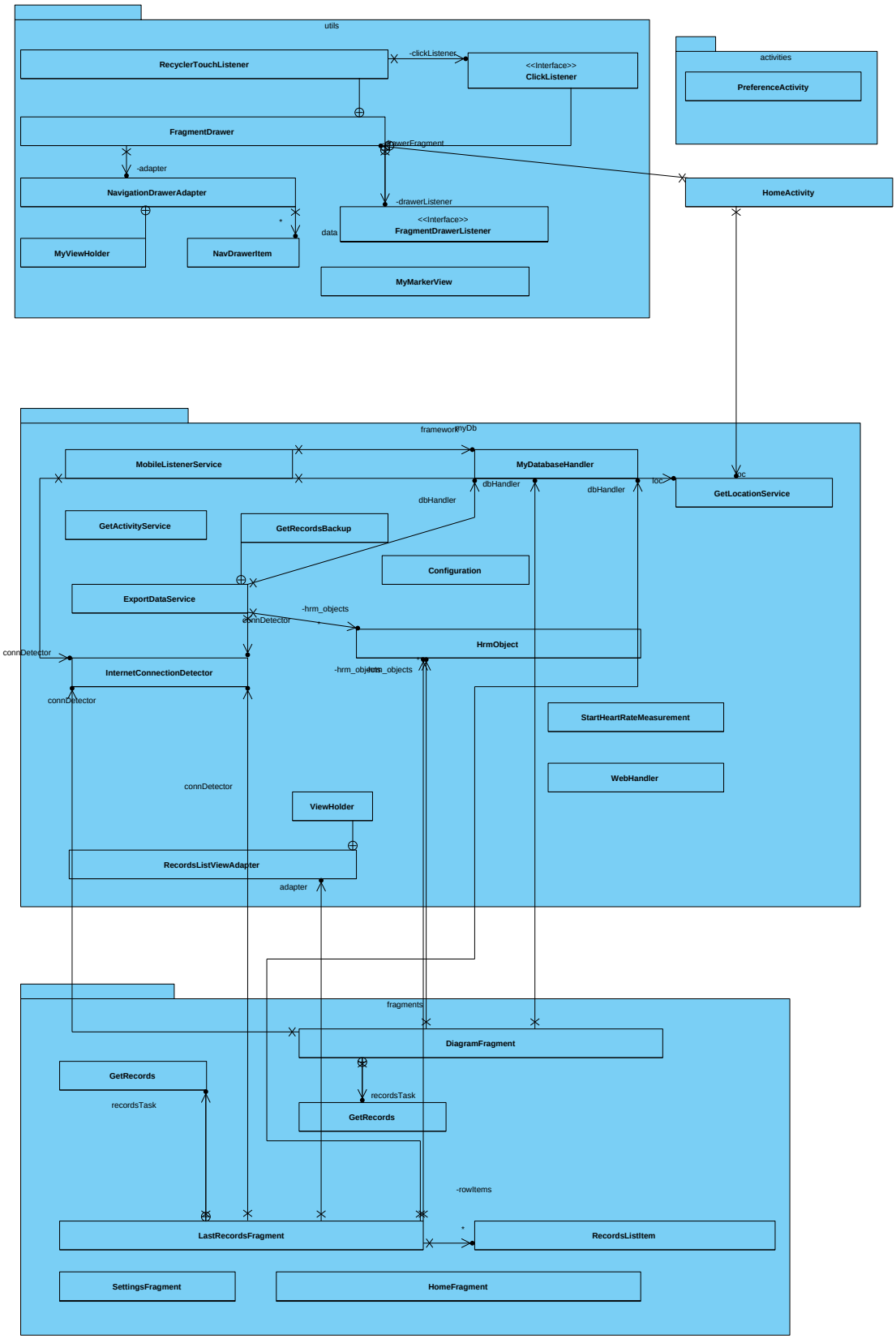

Abbildung B.1: Klassendiagramm der mobilen Anwendung

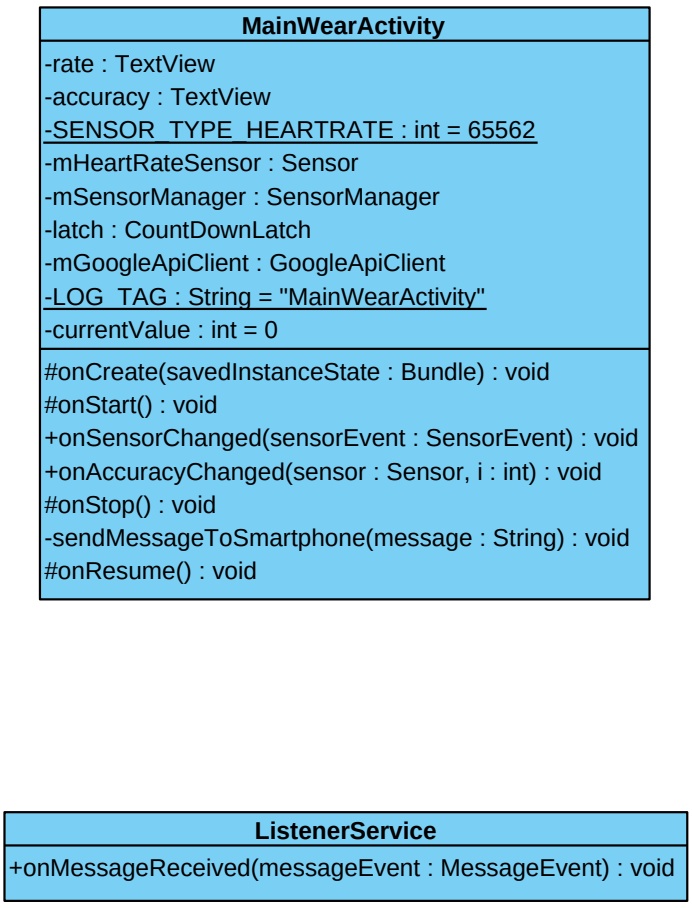

Abbildung B.2: Klassendiagramm der *Android Wear* Anwendung

C

Quelltext

Auf der beiliegenden CD befindet sich der vollständige Quellcode des entwickelten Frameworks. Neben der mobilen Anwendung für Android (inkl. Android Wear Modul), welche mit Android Studio 1.3 Release Candidate 1 entwickelt wurde, liegt auch der Quellcode des Webservice bei.

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Dennis Drzosga Matrikelnummer: 752979

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Dennis Drzosga