ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

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

Konzeption und Realisierung einer generischen Crowd-Sensing Anwendung für Android-TV

Masterarbeit an der Universität Ulm

Vorgelegt von:
Michael Schreiber
michael.schreiber@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert
Prof. Dr. Martin Theobald

Betreuer:
Marc Schickler

2015

Fassung 14. September 2015

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Crowd-Sensing-Systeme gibt es in verschiedenen Ausprägungen. Im Kern beabsichtigen alle Systeme die Sammlung von Daten. Die über eine Menschenmenge erhobenen Daten werden dazu verwendet, Aussagen über bisher unerforschte Gebiete der empirischen Sozialforschung zu ermöglichen. Sehr viele der *Crowd-Sensing-Systeme* werden über den Computer oder das Smartphone realisiert. Jedoch besitzen auch Smart-TV's alle Anforderungen um ein solches System zu ermöglichen.

Im Rahmen dieser Arbeit wird die Implementierung eines *Crowd-Sensing-Systems* basierend auf dem Betriebssystem *Android TV* beschrieben. Das für den Fernseher entwickelte Konzept beabsichtigt das komfortable Beantworten von Umfragen über die Fernbedienung. Neben diesem System wird ebenfalls die Implementierung einer vergleichbaren Smartphone-Anwendung beschrieben. Diese dient zur Durchführung einer Vergleichsstudie. Kern der Studie ist die Frage, ob sich der Smart-TV als *Crowd-Sensing-System* eignet.

Danksagung

Zunächst möchte ich Marc Schickler für die Hilfestellung bei der Themenfindung und die umfassende Betreuung während der Masterarbeit danken. Auch die Korrekturleser möchte ich erwähnen. Mit eurer Hilfe konnten sehr viele Rechtschreibfehler und fehlerhafte Satzstellungen korrigiert werden.

An dieser Stelle möchte ich meiner Freundin Marion Hellmann danken, die mir während des gesamten Studiums helfend zur Seite stand. Ich möchte mich ebenso bei meiner ganzen Familie für die seelische Unterstützung in den letzten Jahren bedanken. Mein herzlichen Dank gilt insbesondere meinen Eltern, die mich auch finanziell während der gesamten Studienzeit unterstützten.

Auch meinen Kommilitonen und Freunden möchte ich für die konstruktive Kritik und die anregenden Ideen Danke sagen. Zuletzt möchte ich betonen, dass nur durch die vielen Studienteilnehmer eine objektive Beurteilung der entwickelten Arbeit möglich war.

Inhaltsverzeichnis

1

Einleitung

Entwickelt man eine *Android*-Anwendung, so muss man sehr viele Faktoren beachten, die den Erfolg einer Anwendung beeinflussen können. So kann es beispielsweise sein, dass eine Anwendung in Deutschland sehr beliebt ist, sie in anderen Ländern jedoch keine Verwendung findet. Um Entwickler bei der Ursachenforschung zu unterstützen, bietet *Google* ein eigenes *Crowd-Sensing-System* mit der Bezeichnung „App-Statistik“ an. Durch die autonome[1] Erfassung wichtiger Meta-Daten, wie der Gerätebezeichnung, der Betriebssystemversion oder der gewählten Sprache, können Schlussfolgerungen ermöglicht werden [Goo15d]. So kann der Entwickler in dieser sehen, dass nur die deutschen Nutzer die Anwendung tatsächlich verwenden und zur Erkenntnis kommen, dass dies an der fehlenden Sprachunterstützung liegen könnte. Jedoch eignen sich *Crowd-Sensing-Systeme* nicht nur zur Profitmaximierung von Anwendungen, sondern auch zur Ursachenforschung von Krankheiten [SSP+13]. Mit “*TrackYourTinnitus*“ konnte die Universität Ulm bereits im Jahr 2014 eine Anwendung entwickeln, die es dem Nutzer ermöglicht den Krankheitsverlauf zu dokumentieren. Über die Zuordnung der dokumentierten Tinnituswahrnehmung zur Tageszeit und Aktivität können Auslöser von Schwankungen durch den Patienten entdeckt und umgangen werden. Zudem können die Daten in einem *Crowd-Sensing-System* ebenfalls vom Betreiber zur Gruppierung und Auswertung verwendet werden. Damit lässt sich beispielsweise der häufigste Auslöser für Schwankungen in der Tinnituswahrnehmung bestimmen [Lin14]. Da die Qualität der Schlussfolgerung über ein *Crowd-Sensing-System* maßgeblich von der Datenmenge abhängig ist, muss die Datenerhebung auf möglichst vielen Geräten erfolgen.

[1] Der Begriff Autonomie ist ein Synonym für Selbstständigkeit

Betrachtet man die Verkaufszahlen der Smartphones von 25,6 Millionen Geräten im Jahr 2015, so sieht man, dass fast jeder vierte Einwohner in Deutschland ein neues Smartphone erwarb [Das15b]. Da erscheint es nur logisch, dass die entwickelte "*TrackYourTinnitus*"-Anwendung auf diese Gerätegruppe zielt [SSPR15]. Jedoch zeigen auch die Absatzzahlen von Smart-TV's[2] einen positiven Trend. So konnten die Verkäufe von 2009 bis 2014 auf 4,6 Millionen verdoppelt werden [Das15a]. Setzt man diese Zahlen in Relation zu den Haushalten in Deutschland, so erwarben mehr als 10 Prozent der Haushalte einen internetfähigen Fernseher im letzten Jahr [Sta15].

Neben den steigenden Absätzen lassen sich auch sehr viele Vorteile der mobilen Datenerhebung auf den Smart-TV übertragen. So bieten Smart-TV's ebenfalls alle möglichen Formen der Erfassung. Damit kann der Entwickler selbst entscheiden, ob die Übermittlung der Daten manuell vom Benutzer oder automatisch vom System ausgelöst wird. Ist die manuelle Datenerfassung gewünscht, so kann der Proband auf beiden Geräten mittels Benachrichtigungssystem subtil auf die gewünschte Datenerhebung aufmerksam gemacht werden. Auch zusätzlichen Sensoren[3], die bei der Erfassung eingebunden werden können, sind in beiden Systemen ansprechbar. So stellt sich die Frage, ob Smart-TV's nicht ebenfalls als *Crowd-Sensing-System* geeignet sind.

1.1 Zielsetzung

Die Ziele der Arbeit lassen sich aus der zuletzt beschriebenen Motivation ableiten. So beabsichtigt die Arbeit die Konzeption und Realisierung einer Anwendung für einen internetfähigen Fernseher. Dazu soll mit einer Voruntersuchung die derzeitig angebotene Hardware und Software analysiert und aufgearbeitet werden. Die erfolgversprechendste Plattform unter den Systemen *Apple TV*, *Fire TV* und *Android TV*, soll somit als Grundlage der Arbeit dienen. Die für den Fernseher entwickelte Anwendung beabsichtigt die Realisierung eines *Crowd-Sensing-Systems* mit dem Fokus der Durchführung von

[2] Als Smart-TV's werden Fernseher bezeichnet, die einen Internetanschluss besitzen
[3] Sensoren sind technische Bauteile und werden zur Erfassung von physikalischen oder chemischen Eigenschaften eingesetzt

Umfragen. Dabei sollen möglichst viele der bereits angedeuteten Vorteile einer mobilen Datenerhebung auf dem Smart-TV ausgenutzt werden. Da bisher keinerlei Untersuchungen über die Eignung des Fernsehers zur Durchführung von Umfragen veröffentlicht wurden, beabsichtigt die Arbeit entsprechende Erkenntnisse zu publizieren. Dazu soll eine Vergleichsstudie zwischen einer Umfrage auf dem Fernseher, einer Datenerhebung auf dem Smartphone und einer schriftlichen Befragung durchgeführt werden. Die in der Studie erhobenen Daten sollen letztlich eine Aussage über die Eignung des Fernsehers als *Crowd-Sensing-System* ermöglichen.

1.2 Aufbau der Arbeit

Zu Beginn werden in Kapitel 2 wesentliche Begrifflichkeiten behandelt. Hier ist zunächst eine Definition des Begriffs "*Crowd-Sensing*" zu finden, bevor die Themenfelder der *empirischen Sozialforschung*, der *Crowd-Sensing-Systeme* sowie *Android* behandelt werden. Da *Android* nach der Geräteform zu unterscheiden ist, sind die Grundlagen für den Fernseher in Abschnitt 2.6 und das Basiswissen der Smartwatches[4] in Abschnitt 2.7 untergliedert. In diesem Kapitel ist ebenfalls eine Auflistung der derzeitigen Lösungen von *Crowd-Sensing-Systeme* integriert. Hier werden die zwei bekanntesten Systeme vorgestellt und deren Unterschiede verdeutlicht. Innerhalb der Anforderungsanalyse wird in Kapitel 3 das initiale Konzept, die nötige Voruntersuchung und die letztlich finale Fassung der Projektplanung vorgestellt. Die Beschreibung der im Endprodukt ausgearbeiteten Komponenten erfolgt im anschließenden Abschnitt. Dieses Kapitel untergliedert sich in die zur Umsetzung benötigten Bibliotheken und Anwendungen. Die zur Beurteilung des Systems durchgeführte Vergleichsstudie wird im Abschnitt 5 im Detail erläutert. So sind hier Informationen über den Aufbau der Studie, die Durchführung, als auch die Zusammensetzung der Stichprobe zu finden. Eine Darstellung der Umfrageergebnisse sowie die Auswertung dieser ist ebenfalls enthalten. Die erreichten Arbeitsergebnisse sind in Abschnitt 6 zusammengefasst. Hier ist ebenfalls ein Ausblick auf mögliche Erweiterungen zu finden.

[4] Eine Smartwatch ist eine Uhr die zusätzliche Sensoren besitzt und damit, neben der Uhrzeit, weitere Funktionen, wie zum Beispiel das Anzeigen der Herzfrequenz besitzt

2

Grundlagen

Befasst man sich mit dem Themenfeld des *Crowd-Sensing*, so lassen sich sehr unterschiedliche Auffassungen finden. Die wichtigsten Fakten zum Gebiet sind im Nachfolgenden zusammengefasst.

2.1 Der Begriff “Crowd-Sensing“

In der Wissenschaft werden Systeme zur Datenerfassung über Sensoren in zwei Kategorien untergliedert. Zunächst gibt es die Gruppe des *Personal Sensing*. Dieses Themenfeld beschäftigt sich mit der Erfassung von Daten für ein Individuum. Beispielweise ist eine Anwendung zur Erfassung der persönlichen, sportlichen Aktivitäten dieser Kategorie zuzuschreiben [RKG11] [GG14].

Die zweite Gruppe der Sensor-Systeme wird unter der Bezeichnung *Community Sensing* gehandhabt. Dieser Begriff kann mit der Bezeichnung *Crowd-Sensing* gleichgesetzt werden und umfasst ein Themenfeld, welches weiter zu differenzieren ist. So unterscheidet man in der Wissenschaft zwischen den Bereichen *Participatory sensing* und *Opportunistic sensing* [GG14]. Als *Participatory sensing* werden Systeme bezeichnet, die eine aktive Beteiligung der Probanden benötigt. Beispielsweise wird eine Anwendung zur aktiven Beteiligung an Umfragen als *Participatory sensing* bezeichnet. Daher wird in der folgenden Ausarbeitung der Begriff des *Crowd-Sensing-System* als Synonym für ein System zur Durchführung von Umfragen angesehen. *Opportunistic sensing* umfasst demgegenüber das autonome Erfassen und Übermitteln von Datensätzen. Zum Beispiel

ist die im Hintergrund automatisch ablaufende Positionsübermittlung von *Google Now*[1] als solches zu verstehen. Dieses Teilgebiet der *Crowd-Sensing-Systeme* wird in der Ausarbeitung nicht weiter betrachtet [RKG11] [GG14] [SPSR15]. Um eine Grundlage für die in Kapitel 5 ausgearbeitete Studie zu schaffen, ist eine Zusammenfassung der *empirischen Sozialforschung* im Nachfolgenden zu finden.

2.2 Empirische Sozialforschung

Der Begriff der *empirischen Sozialforschung* wird zunehmend mit dem der *Umfrage* gleichgesetzt. Diese Trivialisierung beruht auf der Unkenntnis über das Themenfeld. Die Bezeichnung ist als Sammelwerk von Techniken und Methoden anzusehen, die zur korrekten Durchführung von wissenschaftlichen Untersuchungen über das menschliche Verhalten und gesellschaftliche Phänomene angewendet werden [RS11].

Abgrenzen lässt sich die wissenschaftliche Forschung von anderen sozialen Aktivitäten anhand vier Kriterien [KG94]. So ist das Ziel der Wissenschaft immer die Inferenz[2]. Folglich zeichnet sich eine Untersuchung dadurch aus, dass Beobachtungen durch den Forscher nicht nur dokumentiert, sondern auch interpretiert werden. Ebenso besticht eine wissenschaftliche Forschung durch die Veröffentlichung der Vorgehensweise. Nur durch die Publikation dieser können Erkenntnisse durch Außenstehende nachvollzogen und eine Abschätzung der Gültigkeit ermöglicht werden. Als drittes Merkmal der Wissenschaft ist die Möglichkeit zur Einschätzung der Unsicherheit von Schlussfolgerungen zu nennen. Da jede Schlussfolgerung prinzipiell fraglich ist, ist eine wissenschaftliche Forschung durch die Einschätzung des Ausmaßes der Unsicherheit zu erkennen. Zuletzt können diese durch die Verwendung von Methoden bzw. Vorgehensweisen identifiziert werden. Da die Gültigkeit der Schlussfolgerung direkt von der verwendeten Vorgehensweise abhängt, ist diese als markantes Merkmal anzusehen [RS11].

[1] Persönlicher Assistent des Android-Betriebssystem der durch die Positionsermittlung kontextabhängige Informationen bietet

[2] Die Inferenz ist mit dem Wort Schlussfolgerung gleichzusetzen

Ziel der *empirischen Sozialforschung* ist die Überprüfung von Theorien zur Erklärung des menschlichen Handelns und der sozialen Strukturen. Das Forschungsfeld dient demzufolge der systematischen Prüfung von Theorien. In den anschließenden Abschnitten findet sich eine Zusammenfassung der wichtigsten Aspekte der *empirischen Sozialforschung* [RS11].

2.2.1 Phasen des Forschungsprozesses

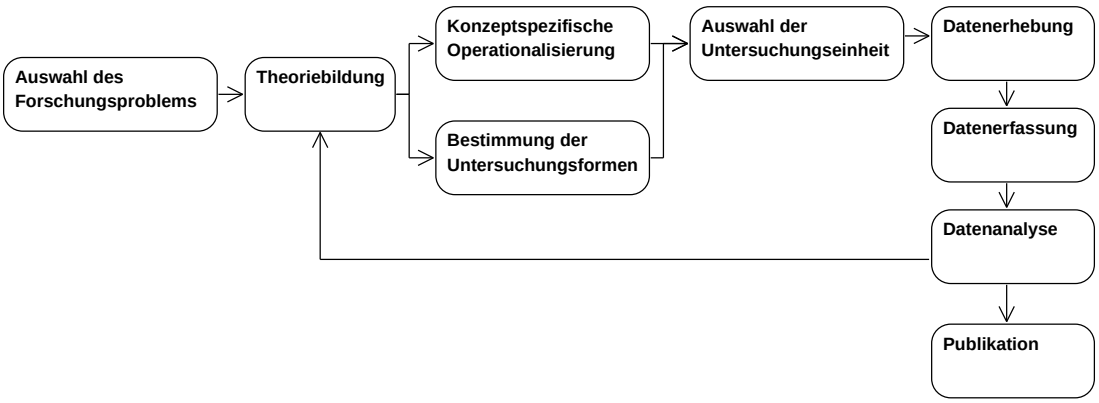

Abbildung 2.1: Phasen der Forschungsprozesse, vgl. Rainer Schnell, Paul B. Hill, Elke Esser, Methoden der empirischen Sozialforschung, Seite 4

Wie der Abbildung 2.1 zu entnehmen ist, lässt sich die Wissenschaft in neun Phasen untergliedern. Gestartet wird die Untersuchung durch die Auswahl des Forschungsproblems. Dabei kann es sich hierbei um ein selbst formuliertes Forschungsproblem oder um ein Problem aus einer Auftragsforschung handeln. Je nach Art der Forschung kann der Spielraum bei der Bestimmung des Untersuchungsgegenstandes stark variieren. Ist das Forschungsproblem definiert, beginnt die Phase der Theoriebildung. Die Theorie des Gegenstandsbereich kann hierbei bereits in der Literatur vorliegen oder muss selbst entwickelt werden. Deshalb beinhaltet die Phase der Theoriebildung vor allem die Recherche der zum Thema existierenden Fachliteratur. Da sehr häufig vage Formulierungen der sozialwissenschaftlichen Theorien vorliegen, müssen Konzepte und Begriffe in der Phase der Konzeptspezifikation und Operationalisierung präzisiert werden. Ebenfalls werden in dieser Phase Voruntersuchungen durchgeführt, um die Gül-

tigkeit und Zuverlässigkeit der Messinstrumente zu testen. Parallel zu dieser Phase wird auch der Prozess zur Bestimmung der Untersuchungsformen durchlaufen. In diesem Abschnitt wird festgelegt, ob die Datenerhebung an einem oder mehreren Messzeitpunkten stattfinden soll. Auch auf die Frage, ob dieselben Personen oder verschiedene Personengruppen zu untersuchen sind, wird hier eine Antwort gefunden. Sind die beiden Prozesse abgeschlossen, muss in der anschließenden Phase entschieden werden, ob die vollständige Menge der Elemente eines Gegenstandsbereichs oder nur eine Teilmenge untersucht werden kann. Beispielsweise stellt sich eine Befragung aller Studenten einer Universität als schwierig dar. Die im anschließenden Prozess der Datenerhebung eingesetzten Techniken sind im Abschnitt 2.2.4 genauer beschrieben. Die durch die Erhebung gewonnenen Daten müssen anschließend in der Phase der Erfassung strukturiert und gespeichert werden, bevor eine anknüpfende Analyse möglich ist. Die Datenanalyse selbst findet primär mit Hilfe von statischen Methoden auf Computern mit speziellen Programmen statt. Da es in dieser Phase häufig zu einer teilweisen Revision der Theorie kommt, ist eine Rückkopplung in Abbildung 2.1 vorgesehen. Sollte es keine weitere Rückkopplung geben, so werden die gewonnenen Erkenntnisse im letzten Schritt veröffentlicht. Hier wird je nach Art der Forschung entweder dem Auftraggeber das Forschungsergebnis zugesendet oder eine Veröffentlichung in diversen Fachzeitschriften angestoßen [RS11].

2.2.2 Wichtige Begriffe der Konzeptspezifikation und Operationalisierung

Um in der Phase der Konzeptspezifikation und Operationalisierung eine Präzisierung der Theorien zu ermöglichen, müssen benötigte Grundbegriffe definiert werden. So sind hier insbesondere die Begriffe *Variablen* und *Indikatoren* hervorzuheben, die im Nachfolgenden beschrieben werden.

Variablen

Eine *Variable* ist ein zusammenfassender Begriff für verschiedene Ausprägungen (Variablenwerte) einer Eigenschaft. Die Anzahl von Variablenwerten hängt dabei nicht von

den Objekten selbst, sondern von den theoretischen Begriffen ab. Sind beispielsweise die Ausprägungen der Ampelfarben gefragt, so ergeben sich augenscheinlich die Variablenwerte “Rot“, “Gelb“ und “Grün“. Wird unter den Farben hingegen die Wellenlänge des Lichtes verstanden, so ergeben sich ganz andere Ausprägungen [RS11].

Variablen lassen sich zudem weiter differenzieren. So kennt die Wissenschaft *dichotome Variablen*, *diskrete Variablen* sowie *stetige Variablen*. Während *dichotome Variablen* nur zwei Werte wie beispielsweise “wahr“ und “falsch“ annehmen können, besitzen *diskrete Variablen* mehr als zwei Ausprägungen. Dennoch besitzen diese im Vergleich zu *stetigen Variablen* einen sehr eingeschränkten Wertebereich [RS11].

Zusätzlich zu dieser Unterteilung wird eine Unterscheidung nach der Wahrnehmbarkeit einer *Variablen* getroffen. Ist eine *Variable* wahrnehmbar, so spricht man von einer *manifesten Variablen*. Ist sie dies nicht, so wird sie als *latente Variable* bezeichnet. Beispielsweise lässt sich die Körpergröße als *manifeste Variable*, das Abstraktionsvermögen hingegen als *latente Variable* bezeichnen [RS11].

Auch eine Differenzierung in die beiden Gruppen *abhängige* und *unabhängige Variable* ist im Forschungsumfeld üblich. Demnach werden *Variablen* als unabhängig aufgefasst, wenn diese für eine Studie festgelegt und vom Probanden nicht beeinflussbar sind. Als *abhängige Variablen* werden hingegen die *Variablen* bezeichnet, die durch den Probanden beeinflusst und in der Studie zu erfassen sind [RS11] [Mac09].

Indikatoren

Der *Indikator* wird in der Wissenschaft auch als *manifeste Variable* bezeichnet und dient zur Verknüpfung eines theoretischen Begriffs mit einem beobachtbaren Sachverhalt. Beispielweise lässt sich die *manifeste Variable* “Anzahl an Schuljahren“ als *Indikator* für den theoretischen Begriff der “kognitiven Kompetenz“ bezeichnen. Dabei besteht das

größte Problem in der Zuordnung eines *Indikators* zu einem theoretischen Begriff [RS11].

Zur Lösung dieses Problems werden in der Wissenschaft drei Ansätze verfolgt. Während die *operationalistische Lösung* und die *typologisch-induktive Lösung* eine untergeordnete Rolle besitzen, bietet die *kausal-analytische Lösung* den fruchtbarsten Ansatz. Bei diesem Ansatz wird die neben der zu testenden Kerntheorie auch eine Hilfstheorie spezifiziert. Über die zusätzliche Formulierung einer Hilfstheorie ist der Wissenschaftler in der Lage, eine Betätigung der Kerntheorie zu erlangen, um einen Beweis der Gültigkeit seiner Verknüpfung zu erbringen [RS11].

2.2.3 Gütekriterien der Messung

Ziel einer Messung ist die Generierung möglichst genauer und fehlerfreier Messwerte. Dieses Ziel kann von kaum einer Messung erreicht werden, weshalb Messfehler bei jeder Messung berücksichtigt werden sollten. Damit Messungen trotz Messfehler sinnvolle Werte enthalten, ist die Anwendung der *klassischen Testtheorie* empfehlenswert [RS11].

Diese Theorie beinhaltet die mehrfache Messung identischer Testvorgänge. Beispielsweise kann die *klassische Testtheorie* zur Bestimmung der durchschnittlichen Sprungweite eines Sportlers herangezogen werden. Der damit ermittelte Messwert kann als vermutlich fehlerfreier Messwert unter der Akzeptanz des Grundmodells der *klassischen Testtheorie* betrachtet werden [RS11].

Zentrum des Modells ist die Annahme, dass ein Messwert immer die Summe eines “wahren Wertes“ und einem Messfehler darstellt. Somit kann der “wahre Wert“ als Mittelwert einer Vielzahl von unabhängigen Messungen desselben Testvorgangs angesehen werden. Diese Annahme basiert auf weiteren Voraussetzungen, die als gegeben angenommen werden. So muss der Mittelwert der Messfehler immer Null sein, da er sonst keinen korrekten Wert darstellen würde. Zudem wird angenommen, dass keine Abhängigkeit zwischen der Größe des Messfehlers und der Größe des Messwertes

besteht. Die dritte Annahme besagt, dass die Messfehler zweier Messreihen niemals untereinander korrelieren[3]. Auch die vierte Annahme beschäftigt sich mit der Korrelation. In ihr wird hingegen die wechselseitige Beziehung zwischen einem Messfehler und dem "wahren Wert" einer anderen Messung ausgeschlossen. Neben dieser Grundlagen sind insbesondere die *Reliabilität* und *Validität* als Gütekriterien entscheidend und werden in den nachfolgenden Abschnitten genauer beschrieben [RS11].

Reliabilität

Die *Reliabilität* kann auch als Zuverlässigkeit einer Messung bezeichnet werden. Es wird erwartet, dass eine wiederholte Messung unveränderter Testvorgänge immer die gleichen Werte liefert. Weiter kann gesagt werden, dass die Höhe der *Reliabilität* maßgeblich von der Stärke des Zusammenhangs zwischen den gemessenen Werten und den tatsächlichen Werten abhängt [RS11].

Verifiziert werden kann die *Reliabilität* einer Messung über die *Test-Retest-Methode* oder einer *Paralleltestmethode*. Während die *Test-Retest-Methode* eine Verifizierung über zwei zeitlich versetzte Tests erlaubt, ermöglicht die *Paralleltestmethode* die Verifizierung über zwei unabhängige Messinstrumente. Die Zuverlässigkeit einer Messung kann von einem Forscher unter- oder überschätzt werden. Weichen die Werte zwischen den Messungen ab, so wird die *Reliabilität* unterschätzt. Geben Probanden hingegen wiederholt falsche Aussagen, so wird diese überschätzt [RS11].

Validität

Die *Validität* ist mit dem Begriff der Gültigkeit gleichzusetzen. Dies umfasst insbesondere das Ausmaß, in dem das Messinstrument den tatsächlich gewünschten Wert misst. Zu beachten ist, dass eine Messung nur eine hohe *Validität* besitzen kann, wenn eine entsprechende *Reliabilität* gewährleistet ist. Die Zuverlässigkeit einer Messung kann

[3] Wenn zwei Objekte korrelieren stehen diese in wechselseitiger Beziehung

hingegen gegeben sein, auch wenn keine Gültigkeit vorhanden ist. Werden beispielsweise bei Messungen gleiche Werte durch die Messung unterschiedlicher Objekte ermittelt, so ergibt sich eine hohe *Reliabilität* jedoch keine *Validität* [RS11].

Die *Validität* einer Messung wird in der Wissenschaft in drei Teilbereiche differenziert. So unterscheidet man zwischen der *Inhaltsvalidität*, der *Kriteriumsvalidität* und *Konstruktvalidität*. Während die ersten beiden Bereiche selten anwendbar sind, wird der *Konstruktvalidität* eine große Bedeutung zugesprochen. Diese Gültigkeit liegt dann vor, wenn mittels empirisch überprüfbaren Aussagen die Zusammenhänge hergeleitet und nachgewiesen werden können. Die Validierung setzt sich dabei aus drei Schritte zusammen. So muss als Erstes die Beziehung zwischen den Konstrukten ermittelt werden. Im zweiten Schritt muss die Relation zwischen den Operationalisierungen der Konstrukte nachgewiesen werden. Zuletzt müssen die empirischen Zusammenhänge dahingehend überprüft werden, ob diese die Hypothese der Validität der Konstrukte tragen [RS11].

Der Begriff *Validität* wird ebenfalls zur Beschreibung des Gültigkeitsbereichs für einen Test verwendet. Spricht man von einer sehr hohen *externen Validität*, so lässt sich das ermittelte Ergebnis auch auf andere Bereiche übertragen. Spricht man hingegen von einer hohen *internen Validität*, so lässt sich festhalten, dass das Ergebnis sehr stark von den Testbedingungen abhängt [RS11].

2.2.4 Datenerhebungstechniken

Die Datenerhebungstechniken lassen sich zunächst in vier Gruppen unterteilen. Wie Abbildung 2.2 zeigt, stellen die *Beobachtung*, die *Inhaltsanalyse*, die *Nicht-reaktiven Messverfahren* und die *Befragung* die Obergruppen dar [RS11] [Kem].

Die *Beobachtung* wird auch als “ursprünglichste“ Datenerhebungstechnik bezeichnet. Dabei besticht die in der Wissenschaft zur Datenerhebung eingesetzte Technik durch ihre Systematik. So lässt sich sagen, dass eine *Beobachtung* stets einen bestimmten

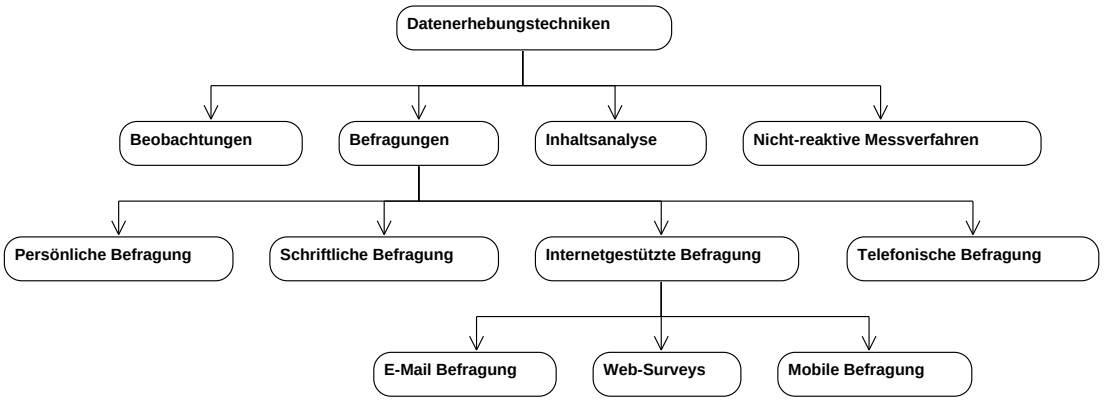

Abbildung 2.2: Datenerhebungstechniken, angelehnt an Rainer Schnell, Paul B. Hill, Elke Esser, Methoden der empirischen Sozialforschung, Seite IX

Forschungszweck verfolgt. Die eingangs erwähnte Systematik ist sowohl bei der Planung, als auch bei der Durchführung, zu erkennen und stellt somit den Ausschluss von Zufällen sicher. Durch Aufzeichnungen und die wiederholte Durchführung kann die Gültigkeit, Zuverlässigkeit und Genauigkeit der Datenerhebung sichergestellt werden. Unter der *Inhaltsanalyse* wird die Recherche von Texten und anderen Medien verstanden. Dabei können sowohl Fotos, Videos als auch Werbeanzeigen analysiert werden. Einsatz findet diese Datenerhebungstechnik insbesondere in der Erforschung der politischen Kommunikation, als auch der Analyse von Massenmedien, unter Berücksichtigung soziologischer Gesichtspunkte. Die Gruppe von *nicht-reaktiven Messverfahren* befasst sich mit dem Begriff Reaktivität. Dabei werden unter dieser Gruppe Verfahren verstanden, die keinerlei Einfluss auf die Reaktion des Probanden besitzen. So gelten insbesondere die *Inhaltsanalyse*, Feldexperimente, die Analyse von Archivdaten als auch bestimmte Beobachtungsverfahren als *nicht-reaktives Messverfahren* [RS11].

Die letzte Obergruppe wird auch als Standardinstrument der Sozialforschung bezeichnet. Dabei wird die *Befragung* weiter in die Untergruppen *persönliche Befragung*, *schriftliche Befragung*, *internetgestützte Befragung* und *telefonische Befragung* unterteilt. Zusätzlich muss die *internetgestützte Befragung* in die verwendete Technik differenziert werden. So ergibt sich eine Unterscheidung in *E-Mail Befragung*, *Web-Survey*[4] und *Mobile Be-*

[4] Ein Web-Survey ist als Umfrage über eine Internetseite aufzufassen

fragung [RS11] [Bec15]. Insbesondere die Bereiche *Web-Survey* und *mobile Befragung* werden sehr häufig in einer Kategorie zusammengefasst. Durch die stetige Weiterentwicklung des Webstandards und der damit verbesserten Darstellung von Internetseiten für mobile Endgeräte, bietet sich die Verwendung von Smartphones für die klassischen *Web-Surveys* zunehmend an. Die stetig gestiegene Nachfrage ist auch der Abbildung 2.3 zu entnehmen. Insbesondere die positiven Eigenschaften von *internetgestützten Befragungen* lassen diese Entwicklung begründen. So kann ein *Web-Survey* sehr schnell und ohne Interviewer durchgeführt werden. Auch die Tatsache, dass die generierten Daten bereits kodiert sind, beschleunigen den Prozess der *Datenerfassung*. Eine Auflistung der derzeit eingesetzten *Crowd-Sensing-Systeme* ist im Abschnitt 2.3 zusammengefasst [RS11] [Bec15].

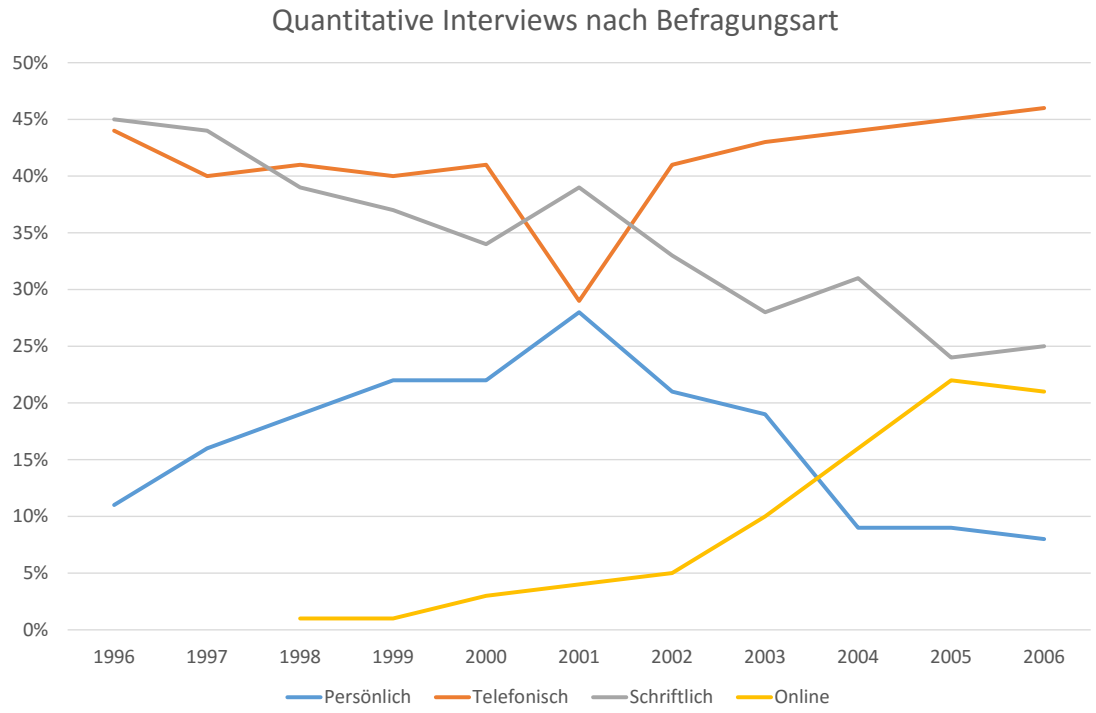

Abbildung 2.3: Quantitative Interviews nach Befragungsart 1996 - 2006 in Deutschland, vgl. Dr. Florian Becker, Entwicklung der Befragungsformen und Trends, [Bec15]

2.3 Crowd-Sensing-Systeme

Wie bereits im Kapitel 2.1 beschrieben, können Systeme zur Befragung von Probanden auch als *Crowd-Sensing-Systeme* bezeichnet werden. Unter den Systemen werden Lösungen auf der Anwendungsebene verstanden, die es dem Anwender ermöglichen, Fragebögen zu erstellen. Im Vergleich zu einfachen Schreibprogrammen bieten diese weitere Funktionen. So können die erstellten Bögen beispielsweise über das System verteilt werden. Auch die Speicherung der erhobenen Daten wird vom *Crowd-Sensing-System* übernommen. Zuletzt bieten die vorhandenen Benutzeroberflächen oftmals eine Darstellung der bereits erfassten Daten an [RKG11] [GG14].

2.3.1 Fragebögen

Wesentliche Bestandteile von Fragebögen sind meist Fragen und Antworten, aber auch andere Medien können ihre Verwendung finden. Dabei differenziert die *empirische Sozialforschung* Fragen in deren Bezug und ihre Form. Eine Zusammenfassung dieser Differenzierung ist im Nachfolgenden beschrieben.

Bezug der Frage

Im Hinblick auf den Bezug lässt sich in der *empirischen Sozialforschung* eine Unterteilung in vier Bezugsgruppen vornehmen. Zunächst kann die Gruppe der *Einstellungsfragen* unterschieden werden. Fragen dieser Gruppe beziehen sich auf den Aspekt der positiven oder negativen Beurteilung von Statements. Erkennbar ist diese Klasse an ihren charakteristischen Antwortmöglichkeiten. So wird dem Probanden hier beispielsweise eine Auswahl zwischen “erwünscht“ bzw. “unerwünscht“ oder “stimme zu“ bzw. “lehne ab“ angeboten. Ebenso definiert ist die Gruppe der *Überzeugungsfragen*. Diese Fragen können ebenfalls an den angebotenen Antwortmöglichkeiten erkannt werden. Wird zu einer Frage die Antwortmöglichkeit “wahr“ bzw. “falsch“ oder “richtig“ bzw. “falsch“ angeboten, so handelt es sich hierbei um eine *Überzeugungsfrage*. Anwendung findet diese Art der Frage jedoch nicht nur zur Wissensabfrage von Sachverhalten, sondern

auch bei Problemstellungen, zu denen keine Antwort bekannt ist. Als dritte und vorletzte Klasse spricht die *empirische Sozialforschung* von *Verhaltensfragen*. Wie der Gruppenname vermuten lässt, beziehen sich diese Fragen auf das Verhalten der Probanden. Im Vergleich zu den bereits beschriebenen *Überzeugungsfragen* bezieht sich diese Klasse nicht auf die Ansicht, sondern auf das Verhalten. Unter den *Einstellungsfragen* wird die Klasse der Fragen verstanden, die auf die persönlichen bzw. demographischen Ausprägungen der Probanden zielen. In der Regel werden Fragen wie beispielsweise “Wie alt sind Sie?“ oder “Was für einer beruflichen Tätigkeit gehen Sie nach?“ zur Klassifizierung der Probanden vorgenommen. Anhand dieser können weitere Schlussfolgerungen über die geäußerten Einstellungen getroffen werden. In Abbildung 2.4 sind beispielhafte Fragen der beschriebenen Bezugsgruppen dargestellt [RS11].

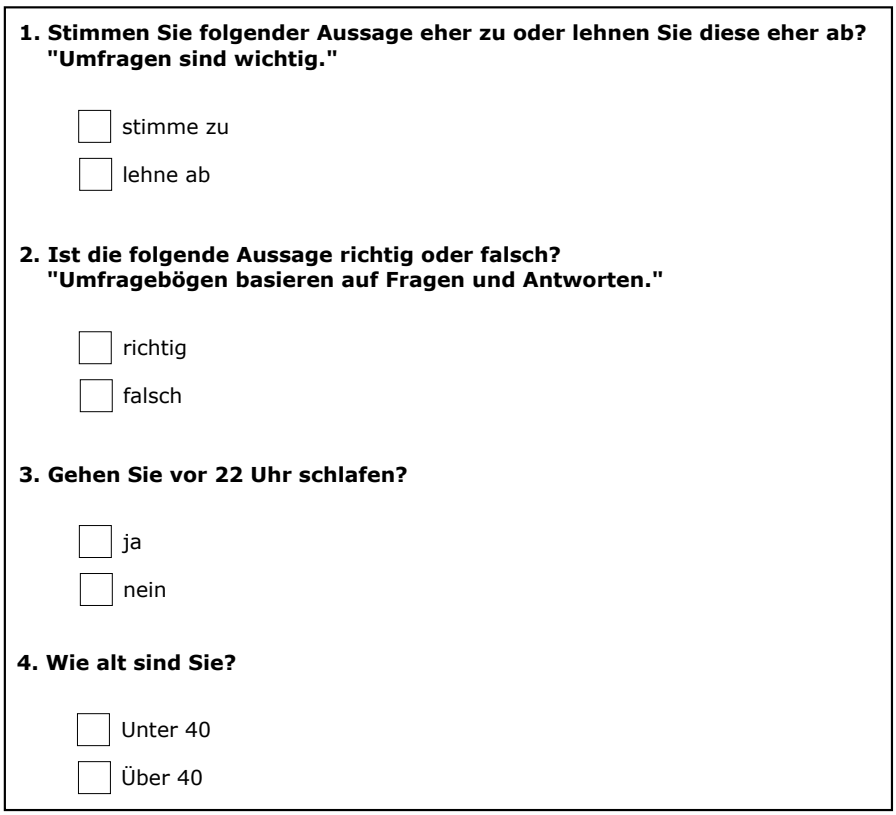

Abbildung 2.4: Einstellungsfrage (1), Überzeugungsfrage (2), Verhaltensfrage (3) und Eigenschaftsfrage (4)

Frageform

Neben dem Bezug differenziert die *empirische Sozialforschung* zwischen den Frageformen: *offene Frage* und *geschlossene Frage*. Unterschieden werden können die Frageformen anhand der Antwortmöglichkeit. Definiert der Umfragebogen für eine Frage vorgefertigte Antwortmöglichkeiten, so handelt es sich hierbei um eine *geschlossene Frage*. Kann der Proband hingegen über ein Textfeld eigene Vorschläge formulieren, so kann diese der Gruppe *offener Fragen* zugeordnet werden. Während *offene Fragen* den Vorteil besitzen, dass diese eine Darstellung der tatsächlichen Vorstellungen des Probanden ermöglichen, besitzen Sie ebenfalls einen entscheidenden Nachteil. So können *offene Fragen* nicht direkt kategorisiert werden. Zudem stellt die anschließende Gruppierung durch den Wissenschaftler eine weitere Fehlerquelle dar. *Geschlossene Fragen* können hingegen die Meinung der Probanden beeinflussen. Hat ein Proband beispielsweise über einen bestimmten Aspekt noch nicht nachgedacht, so verleitet ihn eine entsprechende Antwortmöglichkeit zur Falschaussage. Dennoch werden Fragebögen basierend auf *geschlossenen Fragen* aufgrund der überwiegend positiven Eigenschaften bevorzugt. Abbildung 2.4 zeigt die *geschlossenen Fragen*, eine *offene Frage* ist in Abbildung 2.5 zu sehen [RS11].

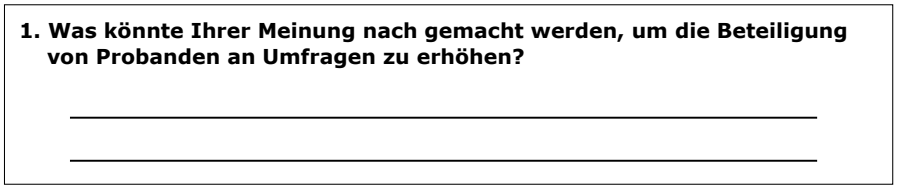

Abbildung 2.5: Offene Frage

2.3.2 Aktuelle Lösungen

Wie bereits in Kapitel 1 der Arbeit erklärt wurde, gibt es derzeit kein veröffentlichtes *Crowd-Sensing-System* für ein Fernsehgerät. Sucht man hingegen ein System für das Smartphone, so lässt sich die eine oder andere Anwendung finden. Eine Vorstellung der bekanntesten Anwendungen ist nachfolgend zu finden.

mQuest Survey market research

Die wohl bekannteste Anwendung zur Durchführung von Umfragen stammt von der cluetec GmbH und besitzt die Bezeichnung *mQuest Survey market research* [clu15]. Wie in Abbildung 2.6 zu sehen ist, bietet diese eine Vielzahl an unterschiedlichen Fragetypen an. So können über das Portal die Fragetypen *Drag&Drop*, *Einfachnennung*, *Heatmap*, *Medieneingabe*, *Mehrfachnennung*, *Nachricht*, *Rangliste*, *Texteingabe*, *Zahleneingabe* genutzt werden. Wie an dem Fragetyp *Medieneingabe* zu sehen ist, kann die Anwendung auch das Mikrofon des Smartphones für die Umfragen einbinden. Damit lassen sich auch sehr schwierige Sachverhalte vom Probanden ausdrücken. Neben dieser nativen Anwendung für Android gibt es alternative Lösungen, die im Folgenden genauer beschrieben werden [clu15].

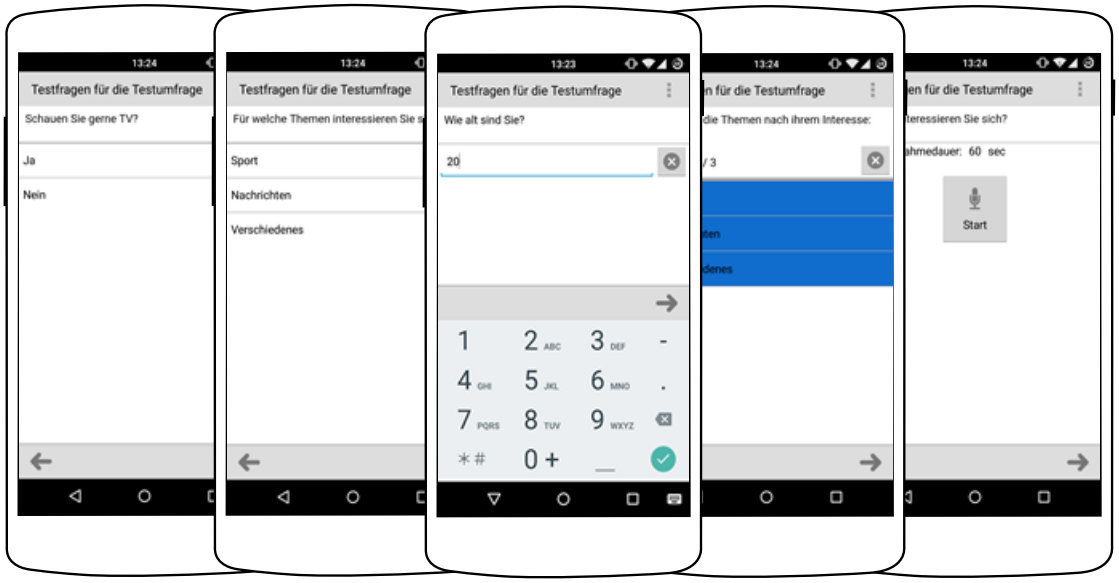

Abbildung 2.6: Screenshots der mQuest Anwendung

Survio

Sehr häufig finden sich Lösungen für Umfragen, wie zum Beispiel die der Firma *Survio*, welche unter ihrer gleichnamigen Webseite ein Portal zur Erstellung von Umfragen anbietet. Diese Umfragen sind nach erfolgreicher Erstellung mittels URL identifizierbar

und müssen selbst über eigene Kommunikationskanäle verteilt werden. *Survio* legt besonderen Fokus auf die dynamische Beschreibung der Umfragen mittels aktuellem Webstandard. Damit lassen sich die generierten Umfragen ebenfalls sehr gut mittels Smartphone durchführen. Zu beachten ist, dass auf eine Vielzahl von Sensoren des Smartphones nicht zugegriffen werden kann, da weder das Portal noch der aktuelle Webstandard dies erlaubt. Eine Gegenüberstellung einer beispielhaften Umfrage ist in Abbildung 2.7 zu sehen [Sur15].

Abbildung 2.7: Gegenüberstellung der Survio-Umfrage auf dem Smartphone und Laptop

2.4 CrowdSensr

Die proprietär entwickelte *Crowd-Sensing-Plattform* mit dem Namen *CrowdSensr* gehört ebenfalls zur Gruppe *Participatory sensing* und benötigt damit die aktive Beteiligung der Probanden. Das in der Arbeit genutzte Teilsystem basiert auf Umfragen, die von Teilnehmern ausgefüllt werden müssen. Die Anwendungslogik selbst basiert auf einem zentralen Hypertext Transfer Protocol (HTTP)-Server[5] und beliebig vielen HTTP-Clients[6].

[5] Ein HTTP-Server besitzt die Anwendungslogik, um die angefragten Webseiten auszuliefern
[6] Ein Browser, wie zum Beispiel der Internet-Explorer, stellt einen HTTP-Client dar

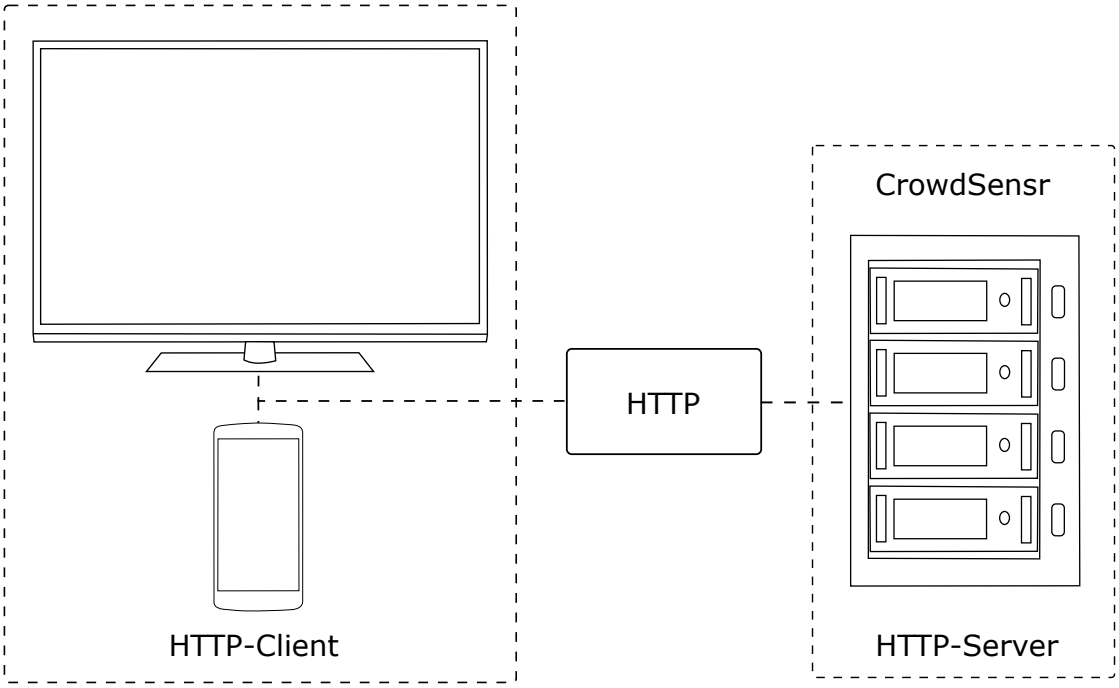

Abbildung 2.8: Architektur CrowdSensr

Dabei können die Clients in unterschiedlicher Form vorliegen, wie in Abbildung 2.8 zu sehen ist. Genauere Informationen über die eingesetzten Standards und die selbst definierten Umfrageelemente werden in den folgenden Abschnitten genauer beschrieben.

2.4.1 JSON

Die JavaScript Object Notation (*JSON*) ist ein schlankes Datenaustauschformat und wird im *CrowdSensr* zur Beschreibung der Umfragen genutzt. Es besticht unter anderem mit seiner Eigenschaft der einfachen Lesbarkeit für den Menschen. Auch für Maschinen ist dieses Format einfach zu analysieren und zu generieren, weswegen es in sehr vielen Projekten eingesetzt wird. Bei *JSON* handelt es sich um ein Textformat, dass die gängigen Konventionen der C-basierten Programmiersprachen[7] berücksichtigt. Dabei baut *JSON* auf die zwei Strukturen *Name/Wert-Paare* und *Geordnete Liste von Werten* auf. Durch diese universelle Datenstruktur wird die Kompatibilität mit sehr vielen Program-

[7] Zu den C-basierten Programmiersprachen zählen insbesondere C, C++ und C#

miersprachen sichergestellt. Während ein *JSON*-Objekt immer aus *Name/Wert-Paaren* besteht, besticht das *JSON*-Array durch eine einfache Aneinanderreihung von Werten. Eine Veranschaulichung dieser elementaren Typen ist im Anhang in Codeausschnitt A.1 zu finden. Dabei kann der Wert neben einer einfachen Zeichenkette (*String*), einer Zahl (*Number*) oder einem Booleschen-Wert[8] wiederum ein *JSON*-Objekt oder ein *JSON*-Array darstellen. Damit stellt das Textformat eine beliebige Verschachtelung sicher und ermöglicht damit die Abbildung komplexer Klassen[9]. Die durch das *JSON*-Format vollständig beschreibbaren Umfragen müssen über einen ebenfalls standardisierten Mechanismus abrufbar sein. Dazu setzt das *Crowd-Sensing-System HTTP* ein, was im Folgenden genauer beschrieben wird [Ecm15].

2.4.2 HTTP

HTTP ist wie der Name vermuten lässt, ein Transportprotokoll. Es ist zustandslos und dient der Übertragung von Daten auf der Anwendungsschicht. Primär findet das *HTTP*-Protokoll seinen Einsatz zur Übertragung von Webseiten im *World Wide Web*. Auch zur Realisierung einer Representational State Transfer (REST)-Webanwendung[10] wird *HTTP* eingesetzt. Dazu werden meist die durch den Standard definierten Methoden `GET`, `POST`, `PUT` und `DELETE` verwendet [Net15] [IBM15].

Möchte man beispielsweise eine Webanwendung entwickeln, bei der die Nutzer ihre Position veröffentlichen können, so würde man das Abfragen der bereits veröffentlichten Positionen über die Methode `GET` abbilden. Möchte er selbst seine Position veröffentlichten, so werden dazu die Methoden `POST` und `PUT` herangezogen. Zu beachten ist hierbei, dass `POST` verwendet wird, sofern es bisher keine veröffentlichte Position des Benutzers gab. Das Löschen seiner veröffentlichten Positionen könnte der Benutzer letztlich über den Aufruf der Methode `DELETE` realisieren. Eine ähnliche Anwendungslogik besitzt auch der in der Universität Ulm entwickelte *CrowdSensr*. Hier werden als Client-Application Programming Interface (API) die Methoden `GET` und `POST` definiert.

[8] Ein Boolescher-Wert kennt die Ausprägungen Wahr oder Falsch
[9] In der objektorientierten Programmierung werden Objekte aus der Welt durch Klassen abgebildet
[10] REST-Webanwendungen werden für Projekte eingesetzt, die eine gute Skalierung benötigen

Während ein `GET` auf den Start-Uniform Resource Locator (URL) eine Liste mit verfügbaren Umfragen für den Anwender liefert, gibt ein `GET` auf die URL einer Umfrage die entsprechende *JSON*-Struktur zurück. Die anschließend ausgefüllte Umfrage kann dann mittels `POST` und der URL der Umfrage an den Server übertragen werden. Damit der Person persönliche Umfragen zugeordnet werden können und auch eine Zugriffskontrolle im System integriert werden kann, muss der Client in seinen Anfragen die *Basic Authentication* von *HTTP* verwenden. Dabei muss er den Benutzernamen und das Passwort mittels *Base64*[11] kodieren und im *Authorization Header*[12] angeben. Eine Beschreibung der definierten *JSON*-Elemente einer Umfrage, welche über *HTTP* ausgetauscht werden, ist im folgenden Abschnitt zu finden [Net15] [IBM15] [Sch15a].

2.4.3 Umfrageelemente

Zur Realisierung des Umfragesystems wird ein Mechanismus benötigt, der es dem Anwender erlaubt, die für ihn zur Verfügung stehenden Umfragen abzurufen. Für diesen Anwendungsfall sieht das System den *Command* vor, welcher anschließend genauer beschrieben wird.

Command

Der *Command* wird als einfaches *JSON*-Objekt abgebildet. Dabei beinhaltet dieser die *Namen/Wert-Paare* `command`, `method` und `info`. Während der Wert des Attributs `method` die *HTTP*-Methode auf die Umfrage definiert, beinhaltet das `info`-Feld eine textuelle Beschreibung der Umfrage für den Client. Die Umfrage, welche letztlich durch den *Command* referenziert wird, ist als relative URL im Attribut `command` abgebildet. Eine Darstellung beispielhafter *Command*-Befehle ist im Anhang in Codeausschnitt A.2 zu finden. Folgt der Anwender der URL des *Command*, so bekommt dieser die entsprechende Umfrage als *JSON*-Struktur zurückgegeben [Sch15a]. Die elementaren Bausteine *Field* und *CompositeField* einer Umfrage werden nachfolgend beschrieben.

[11] Verfahren zur Kodierung von 8-Bit-Binärdaten

[12] HTTP-Nachrichten werden in Header und Body differenziert. Während der Body den Inhalt bzw. die Nutzdaten darstellt, sind im Header Informationen für die Maschine enthalten

Field

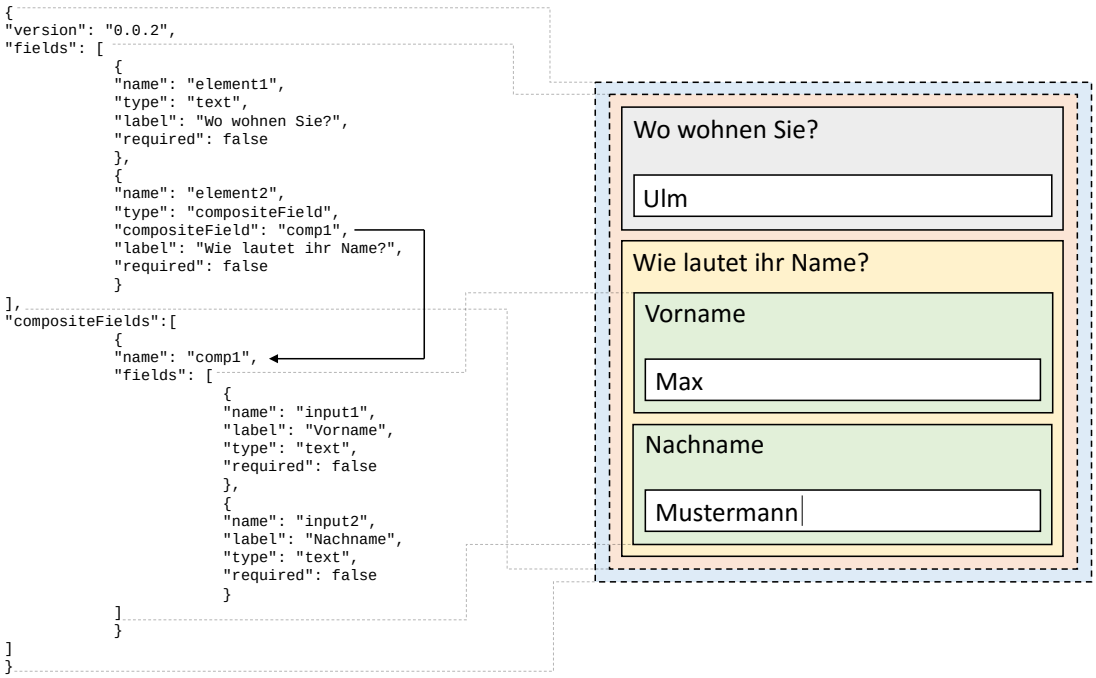

Abbildung 2.9: Gegenüberstellung JSON und visuelle Darstellung

Ein *Field* stellt ein Formularfeld dar und besitzt deshalb zwingend nötige und optionale Attribute. Eine Gegenüberstellung einer JSON-Struktur für ein *Field* und die daraus resultierende visuelle Darstellung ist in Abbildung 2.9 zu sehen. Während die Angabe von `name` und `type` notwendig sind, kann die Angabe von `label`, `pattern` und `required` entfallen. Da eine Referenzierung des Formularfeldes über den angegebenen Namen erfolgt und nur durch die Angabe eines Typs eine korrekte Darstellung möglich ist, ist diese Eigenschaft als elementar kategorisiert. Eine Auflistung der definierten Typen eines Formularfeldes ist in Tabelle 2.1 zu finden. Über die zusätzliche Angabe des Feldes `label` lässt sich eine Beschriftung für das Formularfeld einfügen. Die Attribute `pattern` und `required` können in der Client-Anwendung herangezogen werden, um eine Validierung der Eingaben vor der Übermittlung der Daten an den Server durchzuführen. Eine Darstellung eines komplexen *Fields* ist im Anhang in Codeabschnitt A.3 zu finden. Um eine Gruppierung von Formularfeldern vorzunehmen, kann ein *CompositeField* verwendet werden [Sch15a]. Eine genauere Beschreibung findet sich nachfolgend.

Tabelle 2.1: Auflistung der definierten Typen

Typ	**Zweck**
text	Einzeiliges Eingabefeld für beliebigen Text
textarea	Textbox für Fließtexte
password	Passwortfeld. Eingaben werden als Sterne / Bullets angezeigt und als Klartext zurückgegeben
number	Eingabefeld für Fließkommazahlen. Wertebereich -2 hoch 1024 < number < 2 hoch 1024, Repräsentation als String
email	Eingabefeld für E-Mailadressen
tel	Eingabefeld für Telefonnummern. \r und \n sind nicht erlaubt, sonst keine Validierung
url	Eingabefeld für absolute URLs inkl. Protokoll
date	Eingabemöglichkeit für Datum
time	Eingabemöglichkeit für Uhrzeiten
datetime	Eingabemöglichkeit für einen Zeitpunkt bestehend aus Datum, Zeit und Zeitzonenverschiebung
select	Auswahl einer Option von mehreren, vorgegebenen Möglichkeiten, am besten als Dropdown
multiselect	Auswahl einer oder mehrerer Optionen von mehreren, vorgebebenen Möglichkeiten
checkbox	Binäres Eingabefeld zum Ein-/Ausschalten
radio	Wie select, jedoch kein Dropdown, sondern flache Auflistung der Optionen

CompositeField

Ein *CompositeField* wird im Formular neben den regulären *Fields* als *JSON*-Array übermittelt, wie in Abbildung 2.9 zu sehen ist. Ein einzelnes *CompositeField* besteht dabei immer aus den Attributen `name` und `fields`. Während das Attribut `name` zur Referenzierung der Gruppe herangezogen wird, enthält das Attribut `fields` ein *JSON*-Array mit einer Auflistung der beinhaltenden, einfachen Formularfelder. Eine mehrfache Gruppierung der *CompositeFields* wird vom Umfragesystem nicht unterstützt [Sch15a]. Eine Darstellung eines *CompositeFields* ist im Codeabschnitt A.3 zu finden.

2.5 Android

Mit der Einführung des mobilen Betriebssystem *Android* erlangte *Google* die Marktführerschaft des mobilen Sektors. Seit dem Jahr 2010 verzeichnet das Betriebssystem einen drastischen Anstieg an aktivierten Geräten. So werden zum heutigen Zeitpunkt täglich Millionen neuer Benutzer für das System freigeschaltet. Diesen Erfolg verdankt das mobile Betriebssystem vor allem seiner Offenheit (Open Source) und den mächtigen Tools, die den Entwicklern zur Verfügung gestellt werden. Durch die Vielzahl der entwickelten Anwendungen schafft es der *Play Store*[13] auf eine beachtliche Downloadzahl. So werden monatlich 1,5 Milliarden Downloads im Store der Firma verzeichnet. Mit der aktuellen Version *5.1* werden bereits 4 Gebiete von *Android* unterschieden. So gibt es das Betriebssystem zunächst für die klassischen Smartphones und Tablets. Ebenfalls gibt es einen Fork[14] namens *Android Wear* für die Smartwatches. Auch ein spezieller Ableger für den Fernseher ist mit *Android TV* vorhanden. Zuletzt gibt es mit *Android Auto* ein Betriebssystem, welches die Nutzung von Anwendungen im Fahrzeug ermöglichen soll. Die folgenden Abschnitte geben einen ersten Einstieg in die Entwicklung von klassischen *Android*-Anwendungen. Eine Zusammenfassung der wichtigsten Unterschiede für den Fernseher und die Smartwatch ist in Kapitel 2.6 und 2.7 zu finden [Goo15a].

2.5.1 Activity

Die *Activity* ist die zentrale Klasse bei der Entwicklung von *Android*-Applikationen. Die Anwendungslogik einer Anwendung wird vom Entwickler immer in entsprechende *Activities* untergliedert. Die *Activity* besticht durch eine zentrale Absicht und die dafür benötigte Benutzeroberfläche. Beabsichtigt der Entwickler beispielsweise die Entwicklung einer Chatanwendung mit mehreren Chaträumen, so untergliedert dieser die Anwendung in eine *Activity* zur Darstellung sämtlicher Chaträume und eine *Activity* zur Darstellung des Chatverlaufs eines spezifischen Raums. Damit das *Android*-Betriebssystem die richtige *Activity* beim Start der Anwendung ausführen kann, muss eine zusätzliche Markierung

[13] Zentraler Speicher sämtlicher Anwendungen des mobilen Betriebssystems
[14] Ein Fork bezeichnet in der Softwareentwicklung eine Abzweigung

der entsprechenden *Activity* im *Manifest*[15] der Anwendung erfolgen [Goo15f].

Wird die Anwendung durch das Startmenü des Smartphones aufgerufen, so wird der Lebenszyklus der *Activity* gestartet und ihr durch den Aufruf der vererbten Methode `Activity.onCreate(Bundle bundle)` signalisiert. Diese Methode muss vom Entwickler implementiert werden, um die gewünschte Benutzeroberfläche mit der Klasse zu verbinden. Die Verbindung wird durch den Aufruf der Methode `Activity.setContentView(int id)` hergestellt. Die benötigte `id` erhält der Entwickler über die automatisch generierte Ressourcen-Klasse[16]. Möchte der Entwickler nach der Auswahl eines Chatraums die *Activity* zur Darstellung des Chatverlaufs starten, so sollte er dies über den dafür vorgesehen Startmechanismus machen, um dem vom System zur Verfügung gestellten *Back Stack*[17] nutzen zu können. Dieser Mechanismus sieht vor, dass ein *Intent*, der den Namen der zu startenden *Activity* enthält, über die Methode `Activity.startActivity(Intent intent)` dem Betriebssystem übergeben wird [Goo15f]. Eine genauere Erläuterung des *Intents* ist im folgenden Abschnitt zu finden.

2.5.2 Intent

Die Klasse *Intent* ist als Nachrichtenpaket zu verstehen. Sie besitzt unter anderem eine Methode zur Festlegung der angesprochenen *Activity*. Neben dem Namen des Ziels kann auch eine *Action* angegeben werden. Diese wird mittels *String* dargestellt. Neben einer *Action* bietet das Betriebssystem dem Entwickler an, weitere Daten über den *Intent* zu übermitteln. So beinhaltet die Klasse ein *Bundle*[18] zur Speicherung von einfachen Datentypen. Mittels `Intent.putExtra(String key, String value)` kann der Entwickler relevante Informationen an die andere *Activity* übermitteln. Möchte der Benutzer komplexere Datentypen, wie eigene Klassen über einen *Intent* übertragen, so muss die zu versendende Klasse das *Interface*[19] *Serializable* implemen-

[15] Dokument im XML-Format, dass die Anwendung beschreibt
[16] Eine Klasse die von der Entwicklungsumgebung automatisch erzeugt wird und statische Identifier für Ressourcen besitzt
[17] Anwendungslogik zur Speicherung des Interaktionsverlaufs
[18] Klasse zur Speicherung einfacher Datentypen
[19] Ein Interface gibt in der Softwareentwicklung die Methoden vor, die implementiert werden müssen

tieren und kann dann über die Methode `Intent.putSerializable(String key, Serializable serializable)` angefügt werden [Goo15f].

Intents in Verbindung mit *Actions* werden sehr oft vom Betriebssystem selbst verwendet, um gewisse Events zu signalisieren. So wird beispielsweise nach dem Hochfahren des Betriebssystems ein *Intent* mit der *Action* `android.intent.action.BOOT_COMPLETED` versendet, der es Anwendungen erlaubt, eine gewisse Reaktion, wie beispielsweise das Starten eines Hintergrunddienstes, auszulösen [Goo15f]. Damit eine Anwendung sich für bestimmte *Actions* registrieren kann, werden die Klassen *IntentFilter* und *BroadcastReceiver* benötigt. Diese werden in den folgenden Abschnitten genauer beschrieben.

2.5.3 IntentFilter

Ein *IntentFilter* dient, wie der Name erahnen lässt, zur Filterung der *Intents*, die vom Betriebssystem verteilt werden. Der *IntentFilter* bietet mehrere Möglichkeiten zur Konfiguration an. So können Entwickler bestimmte *Actions*, allgemeine Datentypen oder Datenschemas als Indikator konfigurieren. Das Festlegen eines Datenschemas ermöglicht es beispielsweise, einen eigenen Browser zu entwickeln, der aufgrund der Registrierung für das Schema `http` beim Klicken auf einen Link aufgerufen wird [Goo15f].

2.5.4 BroadcastReceiver

Der *BroadcastReceiver* wird zum Empfangen von *Intents* verwendet und in Verbindung mit einem *IntentFilter* beim System über die Methode `Activity.registerReceiver(BroadcastReceiver receiver, IntentFilter filter)` registriert. *BroadcastReceiver* können jedoch nicht nur dynamisch zur Laufzeit, sondern auch statisch über das *Manifest*, registriert werden. Insbesondere auf das Event `android.intent.action.BOOT_COMPLETED` kann nur über eine statische Verknüpfung reagiert werden [Goo15f].

2.5.5 Fragment

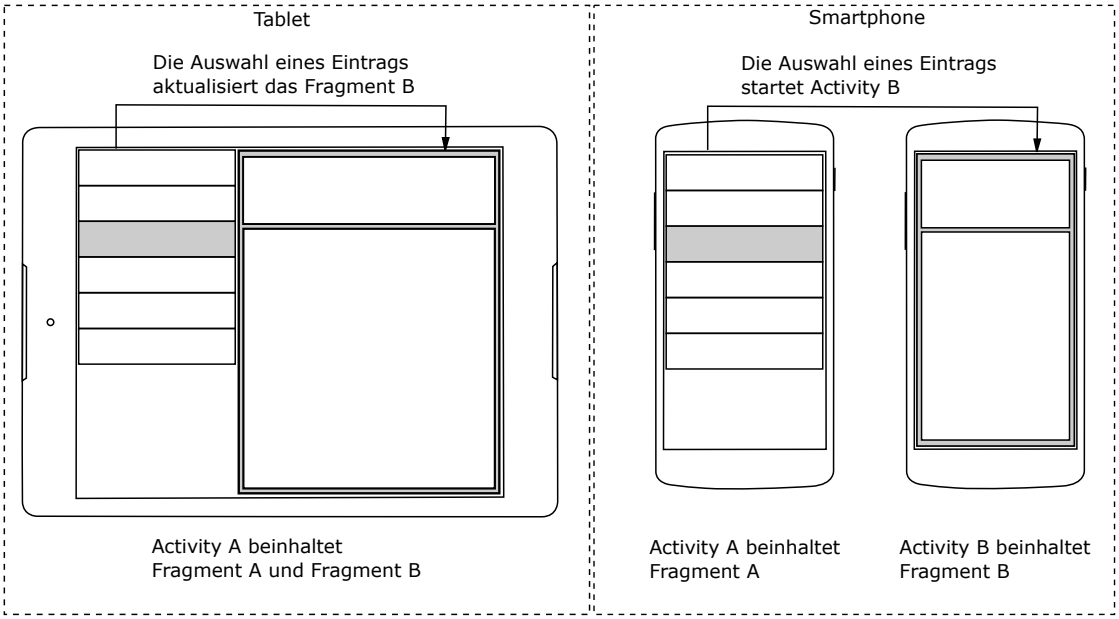

Abbildung 2.10: Gegenüberstellung Tablet und Smartphone, vgl. Google Inc, Fragments, [Goo15g]

Das *Fragment* ist als modularer Baustein einer *Activity* zu sehen und wird insbesondere zur flexiblen Gestaltung von Benutzeroberflächen für Tablets und Smartphones verwendet, wie in Abbildung 2.10 zu sehen ist. Ein *Fragment* kann immer nur innerhalb einer *Activity* existieren und ist somit an deren Lebenszyklus gebunden. Auch bei der dynamischen Veränderung der Benutzeroberfläche mittels Fragments kann der Entwickler auf die Funktion des *Back Stack* zurückgreifen. Analog zur *Activity* wird auch dem *Fragment* der Start über den Aufruf der Methode `Fragment.onCreate(Bundle bundle)` signalisiert. Im Gegensatz zur *Activity* sollte die Benutzeroberfläche des Fragments jedoch erst nach dem Aufruf der Methode `Fragment.onCreateView(LayoutInflater li, ViewGroup vg, Bundle b)` definiert werden. Dies ist auf die Tatsache zurückzuführen, dass ein *Fragment* im Gegensatz zu einer *Activity* sehr wahrscheinlich öfter aus- und eingeblendet wird und deshalb aktualisiert werden sollte. Da nach dem wiederholten Einblenden des *Fragments* lediglich die Methode `Fragment.onCreateView()`

aufgerufen wird, sollte der Entwickler an dieser Stelle die Benutzeroberfläche festlegen [Goo15f].

2.5.6 Service

Ein *Service* stellt einen Hintergrunddienst dar, welcher im Gegensatz zu einem *AsyncTask*[20] für andauernde Prozesse verwendet werden kann. Möchte ein Entwickler beispielsweise einen Nachrichtendienst entwickeln, so sollte dieser einen *Service* starten, damit der Nutzer auch ohne geöffnete Anwendungen Nachrichten empfangen kann. Zum Starten des *Service* gibt es zwei Möglichkeiten. Zunächst kann dieser über einen *Intent* und der Methode `Activity.startService(Intent intent)` gestartet werden. Dieser Mechanismus empfiehlt sich insbesondere, wenn der Hintergrunddienst lediglich einmal gestartet werden muss und anschließend keine weitere Interaktion benötigt. Sollte der Entwickler hingegen eine intensive Interaktion mit dem Hintergrunddienst beabsichtigen, um beispielsweise ausgehende Nachrichten über diesen zu versenden, so empfiehlt es sich, den Hintergrunddienst über eine *ServiceConnection* mittels der Methode `Activity.bindService(Intent i, ServiceConnection sc, int flags)` an die *Activity* zu binden [Goo15f].

Die *ServiceConnection* wird benötigt, um auf die Events des asynchronen Verbindungsaufbaus zum Hintergrunddienst reagieren zu können. Hat das Betriebssystem die Verbindung zwischen den zwei Threads[21] erfolgreich hergestellt, so signalisiert es dies der Anwendung über den Aufruf der Methode `ServiceConnection.onServiceConnected(ComponentName cn, IBinder service)`. Die dabei übergebene Klasse *IBinder*[22] stellt den Hintergrunddienst dar und kann vom Entwickler in die eigentliche Klasse gecastet[23] werden. Daraufhin kann der Entwickler die Instanz wie eine reguläre Klasse zum Aufruf der definierten Methoden verwenden [Goo15f].

[20] Thread mit einem kurzen asynchronen Prozess
[21] Ein Thread ist ein Ausführungsstrang eines Programmes
[22] Ein Interface welches zur Abbildung der Funktionsaufrufe mittels Remote Procedure Call
[23] Unter casten versteht man das Ändern des Betrachtungswinkels einer Klasse

2.6 Android TV

Das unter dem Namen *Android TV* veröffentlichte Betriebssystem basiert auf dem bereits vorgestellten *Android*-OS. Damit ein Einsatz auf dem Fernseher ohne Tastatur und Maus möglich ist, berücksichtigt das System die Bedienung über eine Fernbedienung. Die dafür nötige Anpassung der Benutzeroberfläche und weitere wichtige Aspekte des Systems sind nachfolgend aufgeführt.

2.6.1 App Checkliste

Mit der Einführung von *Android TV* wurde erstmalig eine Checkliste für *Android TV*-Anwendungen veröffentlicht. Diese gliedert sich in die vier Kategorien, die je nach Art der Anwendung, ihren Einsatz finden. So gelten die Kategorien *TV Form Factor Support*[24], *User Interface Design*[25] und *Search and Content Discovery*[26] sowohl für reguläre Anwendungen, als auch für Spiele. Die letzte Kategorie *Games* wurde speziell für Spiele formuliert und muss nur bei diesen beachtet werden [Goo15f].

Über diese Checkliste möchte *Google* sicherstellen, dass eine benutzerfreundliche Bedienung der entwickelten Anwendungen auf dem Fernseher möglich ist. Da beispielsweise der im Abschnitt 2.6.2 vorgestellte *Leanback-Launcher* zur Darstellung der Anwendungen lediglich *Banner*[27] verwendet, muss über die Checkliste sichergestellt sein, dass die Entwickler den Anwendungsnamen in den *Banner* integrieren. Auch die Tatsache, dass auf Fernsehern der *Overscan*[28] berücksichtigt werden muss, ist in der Checkliste aufgeführt. So sieht diese vor, dass sämtliche Oberflächenelemente einen gewissen Mindestabstand zum Rand des Fernsehers aufweisen müssen. Eine vollständige Auflistung der relevanten Punkte ist auf der eingerichteten Entwicklerseite zu finden und sollte vor der Veröffentlichung der Anwendung geprüft werden. Missachtet

[24] Kategorie zur Definition der essentiellen Grundlagen für die Geräteform
[25] Kategorie zur Definition von einheitlichen Standards für Oberflächenelementen
[26] Kategorie zur Definition von einheitlichen Interaktionskonzepten zum Durchsuchen von Inhalten
[27] Ein Banner ist ein Icon mit der Größe von 320x180 Pixel
[28] Als Overscan wird der äußere Bereich des Bildes bezeichnet, welcher je nach Fernseher nicht sichtbar ist

der Entwickler die Richtlinien, so erfolgt die sofortige Entfernung aus dem *Play Store*. Beinhaltet eine Anwendung sowohl die Benutzeroberfläche für ein Smartphone, als auch für einen Fernseher und missachtet die Richtlinien, so wird diese lediglich für den *Play Store* von *Android TV* gesperrt [Goo15f].

2.6.2 Leanback

Abbildung 2.11: Leanback-Launcher

Unter dem Schlüsselwort *Leanback* werden sämtliche, speziell für den Fernseher entwickelten, *Android*-Funktionen zusammengefasst. Der sogenannte *Leanback-Launcher* stellt das Pendant zur Startseite auf einem Android-Smartphone dar und ist speziell auf die Bedienung mittels Remote-Control[29] optimiert. Neben der üblichen Auflistung der installierten Anwendungen und Spiele bietet dieser eine Leiste mit Empfehlungen an, welche im Abschnitt 2.6.3 genauer beschrieben ist. Eine Darstellung des *Leanback-Launchers* ist in Abbildung 2.11 zu sehen. Da sichergestellt werden soll, dass sämtliche, über den *Play-Store*, erhältlichen Inhalte für den Fernseher optimiert sind, muss die

[29] Synonym für den Begriff Remote-Control ist der Begriff der Fernbedienung

jeweilige Anwendung ihre *Activity* im *Manifest* über die Kategorie `android.intent.category.LEANBACK_LAUNCHER` als solche markieren. Mit der Einführung von *Android TV* wurden ebenfalls weitere Klassen innerhalb einer *Support-Library* zur Unterstützung der Entwickler veröffentlicht. Diese sollen ein einheitliches Interaktionskonzept etablieren und die komfortable Bedienung über die Fernbedienung sicherstellen. Eine zentrale Klasse stellt dabei das *BrowseFragment* dar, welches im Anschluss an das Empfehlungssystem genauer beschrieben wird [Goo15f].

2.6.3 Recommendations

Abbildung 2.12: Hervorgehobenes Empfehlungssystem des Leanback-Launchers

Das *Recommendation System* von *Android TV* basiert auf dem bereits in *Android* etablierten *Notification System*[30]. Um Empfehlungen auf dem *Leanback-Launchers* wie in Abbildung 2.12 darstellen zu können, muss die reguläre *Notification* entsprechend markiert werden. Dies kann der Entwickler über die Methode `Notification.setCategory(String category)` des *Notification.Builder* erreichen, indem er den

[30] Das Notification System von Android informiert den Benutzer über Ereignisse, die durch Hintergrunddienste ausgelöst werden

vordefinierten String `Notification.CATEGORY_RECOMMENDATION` verwendet. Analog zur klassischen *Notification* gilt, dass mindestens der Titel und ein kleines Icon gesetzt werden müssen. Vergisst der Entwickler diese Anforderung, wird die *Notification* vom Betriebssystem ignoriert [Goo15f].

2.6.4 BrowseFragment

Abbildung 2.13: Leanback-Beispiel für BrowseFragment, angelehnt an Google Inc., BrowseFragment, [Goo15i]

Das *BrowseFragment* stellt die zentrale Klasse zur Darstellung verschiedener, durchsuchbarer Medien dar. Eingebunden wird das *BrowseFragment* über ein entsprechendes XML-Element im Layout der *Activity*. Auch auf dem Fernseher muss das gewünschte Layout über den im Abschnitt 2.5.1 vorgestellten Mechanismus verbunden werden. Anschließend kann die *Activity* in ihrer Methode `Activity.onCreate(Bundle b)` über den *FragmentManager*[31] das entsprechende *BrowseFragment* mittels `id` suchen. Die in

[31] Eine Klasse des Android-Betriebssystems zur Verwaltung von Fragments

Abbildung 2.13 dargestellten Überschriften und Medien werden durch das Verbinden eines *ArrayObjectAdapter* über die Methode `BrowseFragment.setAdapter(Adapter a)` erzeugt. Weitere Informationen über den *ArrayObjectAdapter* finden sich nachfolgend [Goo15f].

2.6.5 ArrayObjectAdapter

Ein *ArrayObjectAdapter* kann verschiedene Darstellungen repräsentieren. Um eine Oberfläche wie in Abbildung 2.13 zu erreichen, muss dem *ArrayObjectAdapter* ein *ListRowPresenter* in der Parameterliste des Konstruktors[32] übergeben werden. Dadurch wird signalisiert, dass der *ArrayObjectAdapter* am jeweiligen Index nicht nur ein einzelnes Objekt, sondern eine weitere Liste von Objekten enthält. Diese Liste stellt einen weiteren *ArrayObjectAdapter* dar, der in der Parameterliste ein einzelnes Objekt, anstelle einer weiteren Liste, enthält. Die Darstellung der Kategorienbezeichnung wird letztlich durch die Verknüpfung eines *HeaderItems* mit dem entsprechenden *ArrayObjectAdapter* erreicht. Damit eine Darstellung der Vorschaubilder, wie in Abbildung 2.13, für das *BrowseFragment* möglich ist, muss das darzustellende Objekt vom Typ *Presenter* sein [Goo15e] [Goo15f] [Goo15f]. Eine Beschreibung des *Presenters* findet sich im nachfolgenden Abschnitt.

2.6.6 Presenter

Der *Presenter* legt die Darstellung des entsprechenden Objektes fest. Die Verbindung des Layouts erfolgt im *Presenter* in der Methode `onCreateViewHolder(ViewGroup viewGroup)`. Hier kann die entsprechende Extensible Markup Language (XML)-Datei mittels *LayoutInflater*[33] eingeblendet und dem Aufrufer zurückgegeben werden. In der ebenfalls geerbten Methode `Presenter.onBindViewHolder(ViewHolder h, Object item)` bekommt der *Presenter* vor der Darstellung auf der Benutzeroberfläche das darzustellende Model[34] mittels Parameter `item` übergeben. Aus diesem

[32] Ein Konstruktor ist mit einer Methode zu vergleichen und wird beim Erzeugen einer Klasse aufgerufen
[33] Eine Klasse des Android-Betriebssystems zur Einblendung von dynamischen Oberflächenelementen
[34] Als Model werden in der Softwareentwicklung die Datenhalter bezeichnet

Model könnte, wie in Abbildung 2.13, das *Drawable*[35] des Bildes entnommen und der *ImageView*[36] übergeben werden [Goo15f].

2.7 Android Wear

Android Wear basiert ebenso wie *Android TV* auf *Android* selbst. Jedoch sind in *Android Wear* erweiterte Konzepte integriert, um die Bedienbarkeit der Anwendungen auf einem kleinen, tragbaren Gerät sicherzustellen. Zum Beispiel muss beachtet werden, dass eine Installation einer Anwendung auf der Uhr nicht direkt erfolgen kann. *Google* sieht an dieser Stelle einen neuen Mechanismus vor, der im folgenden Abschnitt genauer beschrieben wird.

2.7.1 Packaging

Möchte ein Entwickler eine *Android Wear*-Anwendung veröffentlichen, so muss er die Anwendung in einer regulären *Android*-Anwendung kapseln. Diese Anwendung kann dann vom Benutzer über den *Play Store* auf das Smartphone geladen werden. Die auf dem Smartphone installierte Anwendung (*Android Wear Companion*) erkennt, dass diese Applikation eine *Android Wear*-Anwendung kapselt und übernimmt die automatische Installation auf der Smartwatch. Damit dieser Mechanismus funktioniert, müssen einige Punkte beachtet werden [Goo15f].

Wichtig ist, dass die *Android Wear*- und *Android*-Anwendung identisch Berechtigungen besitzen. Neben den Berechtigungen müssen beide Projekte die gleiche Versionsnummer und das selbe *Package*[37] enthalten. Als dritten und letzten Punkt sollte die Abhängigkeit der Projekte definiert werden. Damit *Android-Studio* beim Erstellen einer

[35] Eine Klasse des Android-Betriebssystems welche ein Bild repräsentiert
[36] Eine Klasse des Android-Betriebssystem welche die Darstellung von Bildern übernimmt
[37] Als Package wird in der Softwareentwicklung mit der Programmiersprache Java ein eindeutiger Namensraum bezeichnet. Oftmals wird eine Kombination aus Domain der Firma und Anwendungsnamen verwendet

signierten APK-Datei[38] die *Android Wear*-Anwendung integrieren kann, muss das entsprechende Projekt als Abhängigkeit in der Smartphone-Anwendung gelistet werden. Neben der nötigen Konfiguration der Projekte sieht *Google* auch die Verwendung neuer Schnittstellen vor. Eine Zusammenfassung der wichtigsten Neuerungen ist anschließend zu finden [Goo15f].

2.7.2 WatchViewStub

Der *WatchViewStub* und der damit einhergehende *WatchViewStub.OnLayoutInflatedListener* bieten Entwicklern für Android Wear Geräten eine Möglichkeit, auf die unterschiedlichen Geräteformen der Uhren zu reagieren. Realisiert wird dieser Mechanismus über das Einbinden des entsprechenden XML-Elements im Layout der *Activity*. In dem XML-Element des *WatchViewStubs* können anschließend die Referenzen auf das runde und rechteckige Layout über die Attribute `rectLayout` bzw. `roundLayout` des entsprechenden XML-Namespaces[39] hinzugefügt werden. In der Methode `onCreate()` der *Activity* muss der in XML definierte *WatchViewStub* über dessen `id` referenziert und mit einem entsprechenden *OnLayoutInflatedListener* verbunden werden. Das erfolgreiche Einblenden des runden oder rechteckigen Layouts wird der Anwendung über die vererbte Interface-Methode `OnLayoutInflatedListener.onLayoutInflated(WatchViewStub wvs)` signalisiert. Über den Parameter lassen sich anschließend sämtliche Oberflächenelemente der Layouts referenzieren [Goo15f].

Zu beachten ist, dass die zu referenzierenden Elemente sowohl im runden als auch im rechteckigen Layout enthalten sein müssen. Damit eigene Nachrichten vom Smartphone an die Uhr gesendet werden können, muss die *Wearable.DataApi* auf dem Smartphone und ein *WearableListenerService* auf der Uhr verwendet werden. Eine Einführung in die angesprochenen Klassen findet sich in den folgenden Abschnitten [Goo15f].

[38] Eine APK-Datei ist vergleichbar mit einer ZIP-Datei und dient zur Installation von Anwendungen

[39] Als Namespace bezeichnet man in der Softwareentwicklung den Gültigkeitsbereich. Er wird zur eindeutigen Identifizierung von Objekten verwendet

2.7.3 Wearable.DataApi

Mit der *Wearable.DataApi* können *DataItems*[40] über den *GoogleApiClient* übermittelt werden. Dazu muss ein *PutDataMapRequest*-Objekt erzeugt und anschließend über dessen Methode `PutDataMapRequest.getDataMap().putString(String s)` mit Inhalt befüllt werden. Zu beachten ist, dass Nachrichten nur übermittelt werden, wenn sich deren Inhalt geändert hat. Möchte man sicherstellen, dass jede Nachricht übermittelt wird, sollte der aktuelle Zeitpunkt als Paketinhalt eingefügt werden. Das befüllte *PutDataMapRequest*-Objekt kann anschließend über die Methode `PutDataMapRequest.asPutDataRequest()` in ein *PutDataRequest*-Objekt umgewandelt werden. Dieses Objekt kann letztlich über die *Wearable.DataApi* durch den Aufruf der Methode `Wearable.DataApi.putDataItem(GoogleApiClient c, PutDataRequest r)` versendet werden. Damit die inaktive Anwendung auf der Uhr ebenfalls das Nachrichtenpaket empfangen kann, muss ein *WearableListenerService* eingesetzt werden. Eine genauere Beschreibung des Hintergrunddienstes ist nachfolgend beschrieben [Goo15f].

2.7.4 WearableListenerService

Der *WearableListenerService* erbt die Eigenschaften eines *Service*. Dadurch kann sichergestellt werden, dass dieser immer im Hintergrund ausgeführt wird. Die spezielle Unterklasse definiert im Vergleich zum *Service* eine zusätzliche Methode `WearableListenerService.onDataChanged(DataEventBuffer deb)`. Über diese Methode können Nachrichtenpakete vom Smartphone empfangen werden. Dazu muss ein *DataMapItem*[41] über die Methode `DataMapItem.fromDataItem(deb.getDataItem())` erzeugt werden. Über dieses *DataMapItem* können die Daten analog zum *PutDataMapRequest*-Objekt über eine Methode ausgelesen werden [Goo15f].

[40] Eine Klasse des Android-Betriebssystem zur Speicherung von Daten im Android Wear Network
[41] Eine Klasse des Android-Betriebssystem, die über ein DataItem erstellt werden kann und eine strukturierte Darstellung der Daten bietet

2.8 UPnP

Die Technologie rund um Universal Plug and Play (*UPnP*) definiert die Architektur von Peer-to-Peer[42] Netzwerken. Besonders zur schnellen und flexiblen Verbindung von Ad-Hoc-Netzwerken[43] eignet sich der *UPnP*-Standard. Der Standard ermöglicht es dem Nutzer beispielsweise andere Netzwerkgeräte aufzufinden, ohne über deren Internet Protocol (IP)-Adresse[44] Bescheid zu wissen. Auch auf der Anwendungsebene ist der *UPnP*-Standard hilfreich. So können Anwendungen zum Beispiel gezielt nach Gerätetypen, wie Drucker, suchen, um dem Benutzer den Ausdruck über Netzwerk zu ermöglichen [UPn15].

UPnP selbst basiert dabei auf bereits standardisierten Protokollen und Formaten. So findet beispielsweise das Vermittlungsprotokoll IP und die Transferprotokolle Transmission Control Protocol (TCP)[45], User Datagram Protocl (UDP)[46] und *HTTP* ihren Einsatz. Damit eine Differenzierung der Gerätetypen möglich ist, werden sämtliche *UPnP*-Geräte über den XML-Standard genauer beschrieben. Der Mechanismus zum Informieren und Auffinden von *UPnP*-Geräten ist im Standard selbst definiert. Das dafür entwickelte Simple Service Discovery Protocol *SSDP* basiert auf Teilen des *HTTP*-Standards. Die zwei elementarsten Nachrichten dieses Protokolls stellen die *Notify* und *M-Search* dar. Während ein *Notify* zur Information über den eigenen Dienst genutzt wird, kann eine aktive Suche nach *UPnP*-Geräten über die *M-Search*-Nachricht ausgelöst werden. In dieser Nachricht können weitere Parameter über das gewünschte Zielgerät eingefügt werden. Da diese Nachrichten über das Transportprotokoll *UDP* versendet werden, sind die Nachrichtenlängen sehr begrenzt. Aus diesem Grund findet die Gerätebeschreibung über ein XML-Dokument statt, welches vom Client über *HTTP* angefordert werden kann. Auf den Ort dieses Dokuments wird mittels *Location-Header* in der *SSDP*-Nachricht

[42] Im Gegensatz zu einer Kommunikation zwischen einem Server und Client sind bei einem Peer-to-Peer Netzwerk beide Parteien gleichgestellt
[43] Ad-Hoc-Netzwerke werden für die aktuelle Situation aufgebaut, besitzen jedoch keine Beständigkeit
[44] Über eine IP-Adresse können Rechner in einem Netzwerk angesprochen werden
[45] Ein Netzwerkprotokoll, welches die Erkennung der fehlerhaften Übertragung erlaubt
[46] Ein Netzwerkprotokoll, welches zur Übertragung ohne Empfangsbestätigung entwickelt wurde

hingewiesen [UPn15].

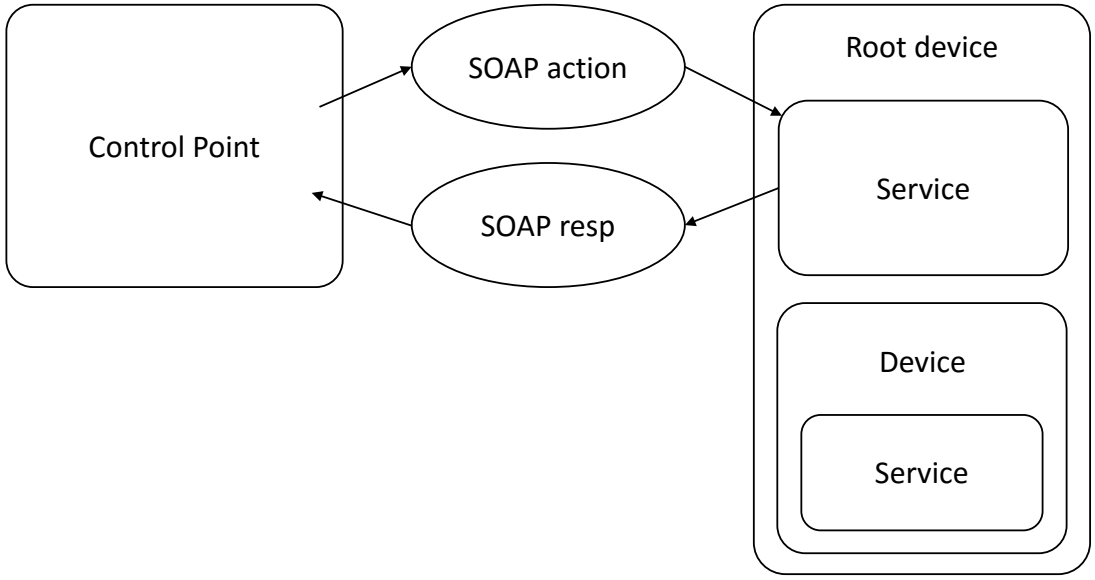

Abbildung 2.14: Kontrollarchitektur, angelehnt an UPnP Forum, UPnP Device Architecture 2.0, Seite 44, [UPn15]

Damit sämtliche Nachrichten an alle Netzwerkgeräte zugestellt werden, sind diese an die Multicast-Adresse[47] 239.255.255.0 und den Standard-Port[48] 1900 adressiert. Eine beispielhafte Darstellung einer *Notify* und *M-Search* ist im Anhang in Codeausschnitt A.4 zu finden. Damit zusätzlich zum Auffinden der Geräte auch eine Kontrolle dieser möglich ist, beinhaltet der *UPnP*-Standard zudem ein Kontrollprotokoll. Dieses Kontrollprotokoll basiert auf dem Simple Object Access Protocol(*SOAP*). Wie in Abbildung 2.14 zu sehen ist, werden dafür sogenannte *SOAP Actions* an einen *Service* des Zielgerätes gesendet. Dieser Dienst wertet die Anfrage aus und erzeugt eine entsprechende *SOAP Response*, welche ebenfalls über das Netzwerk übermittelt wird [UPn15].

[47] Eine Multicast-Adresse wird in Netzwerken eingesetzt, um Nachrichten an sämtliche Teilnehmer zu übermitteln

[48] Ports werden benötigt, damit mehrere Programme auf einem Rechner über das Internet kommunizieren können

2.9 Fitness-Portale

Fitness-Portale stellen einen zentrale Speicherquelle für Fitness-Daten dar. Sie erlauben es dem Benutzer, mehrere Geräte zur Erfassung der sportlichen Aktivitäten zu kombinieren, ohne dass dieser die Daten manuell abgleichen muss. So kann er beispielsweise die Herzfrequenz über eine Smartwatch, die gelaufenen Schritte hingegen über das Smartphone aufzeichnen lassen. Am derzeit bekanntesten sind die Plattformen *Google Fit*, *Apple Health Kit* und *Microsoft HealthVault*, die einen ähnlichen Funktionsumfang anbieten. Eine genauere Vorstellung der angebotenen Funktionen ist im nachfolgenden Abschnitt anhand des Fitness-Portals *Google Fit* zu finden [Goo15h] [Mic15] [App15c].

2.9.1 Google Fit

Das von *Google* veröffentlichte Portal bietet für Entwickler sowohl eine *REST-API*, als auch ein Software Development Kit (SDK)[49], für das eigene Betriebssystem *Android* an. Die Plattform selbst untergliedert sich in drei Komponenten: *Fitness-Stores*, *Sensor-Frameworks* und *Permissions and user control*. Während sich der Funktionsumfang des *Fitness-Stores* auf die Speicherung und Ausgabe der Daten konzentriert, sind die elementaren Datentypen im *Sensor-Framework* definiert. So definiert dieses zunächst alle verfügbaren Sensoren als *Data Source*. Beschrieben sind diese Sensoren über einen eigenen Namen und den erfassten Datentyp. Der Datentyp (*Data Type*) selbst besteht ebenfalls aus einem Namen und einer Liste von Feldern. Beispielsweise lassen sich bei einem Global Positioning System (GPS)-Sensor[50] die Felder *Latitude*, *Longitude* und *Altitude* auffinden. Die Verbindung zwischen Daten und Zeitpunkten sind über ein *Data Point* abgebildet. Möchte ein Entwickler *Data Points* auslesen oder speichern, so kann dies über *Datasets* oder *Sessions* erfolgen. Während ein *Dataset* eine reine Liste von *Data Points* eines bestimmten Typs innerhalb einer Zeitspanne darstellt, bietet die *Session* eine direkte Zuweisung der Daten zu einer sportlichen Aktivität. Um die Zugriffsrechte zum *Fitness-Store* zu regeln, verwendet *Google* das *OAuth*-Verfahren. Dabei wird einer Anwendung ein *Token* durch die erfolgreiche Eingabe des Logins

[49] Ein SDK wird in der Softwareentwicklung als Bibliothek bezeichnet, welche die Programmierung erleichtert
[50] Ein Sensor zur Bestimmung der Ortsdaten

ausgestellt. Dieses *Token* ist als eine Art Vollmacht anzusehen, welche die Anwendung zu bestimmten Handlungen berechtigt. So differenziert *Google* beim *Fitness-Store* die drei Gruppen *Activity*, *Location* und *Body*. Beantragt eine Anwendung lediglich das Leserecht der Gruppe *Body*, so können anschließend nur Informationen dieser Kategorie ausgelesen werden [Goo15h].

3

Anforderungsanalyse

Um eine solide Grundlage für die zu implementierende Anwendung zu schaffen, ist eine Anforderungsanalyse nötig. Im Nachfolgenden werden die Anwendungsfälle, das Grundkonzept, die Hard- und Softwareanalyse und das erweiterte Konzept der durchgeführten Anforderungsanalyse vorgestellt.

3.1 Anwendungsfälle

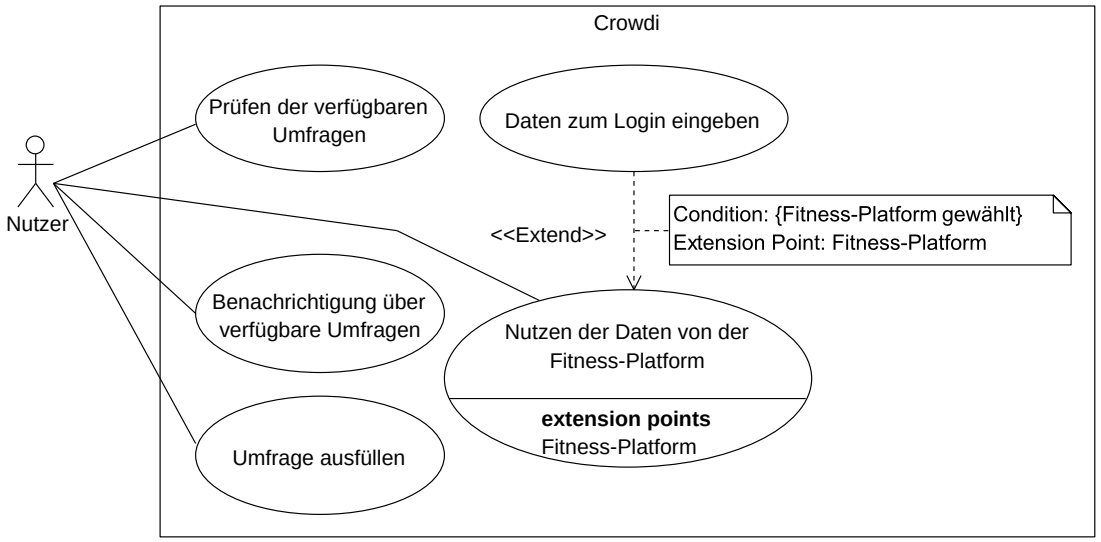

Abbildung 3.1: Anwendungsfälle

Bei der Planung eines Softwareprojektes sollte immer ein *Use-Case-Diagramm* erstellt werden. Durch die Formulierung der wesentlichen Funktionen eines Systems kann der

Auftraggeber prüfen, ob alle für ihn wichtigen Elemente berücksichtigt sind [Sch15b]. Auch für den Entwickler bietet das *Use-Case-Diagramm* einen entscheidenden Vorteil. So können durch die abgebildeten *Use-Cases* benötigte Softwarekomponenten bereits früh erkannt und geplant werden. Abbildung 3.1 zeigt das für diese Arbeit erstellte *Use-Case-Diagramm*. Die darin berücksichtigten *Use-Cases* sind selbst definiert und werden im Nachfolgenden genauer erläutert.

Zunächst erwartet ein Benutzer von einer Umfrage-Anwendung, dass er mit dieser an Umfragen komfortabel teilnehmen kann. Dazu möchte er über die Anwendung sehen, welche Umfragen für ihn zur Verfügung stehen. Da er die Anwendung jedoch nicht täglich öffnen möchte, um die Verfügbarkeit einer neuen Umfrage zu überprüfen, erwartet er, dass die Applikation ihn über neue Umfragen informiert. Zur Erfassung seiner Fitness-Aktivitäten verwendet der Benutzer eines der vielen Fitness-Portale und möchte deshalb die erfassten Daten für entsprechende Fragen einer Studie nutzen können. Deshalb rechnet er damit, dass die Anwendung einen entsprechenden Mechanismus zum Verbinden seines Fitness-Portals zur Verfügung stellt.

3.2 Grundkonzept

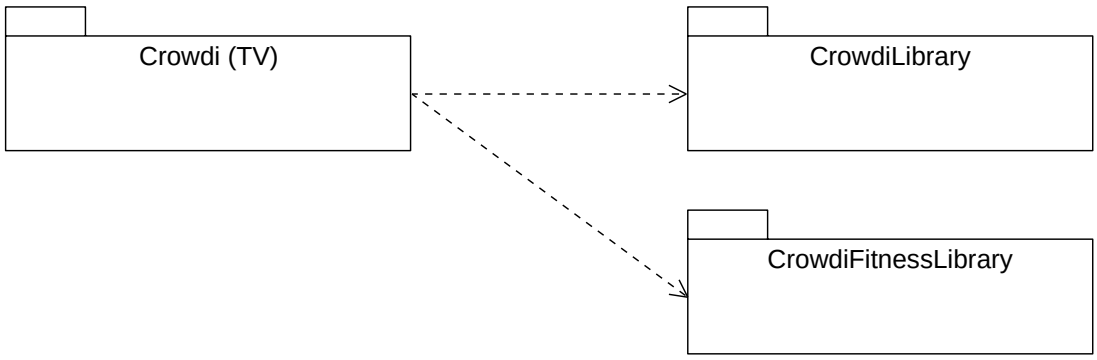

Abbildung 3.2: Grundkonzept der Anwendung

Um die im letzten Abschnitt geschilderten Kundenwünsche für den Fernseher erfüllen zu können, wird eine Anwendung für das Betriebssystem eines Fernsehers benötigt.

Damit die allgemeingültige Anwendungslogik zur Anbindung an den *CrowdSensr* auch auf weiteren Endgeräten verwendet werden kann, sollte diese in einer Bibliothek abstrahiert werden. Auch die vom Kunden geforderte Integration seines Fitness-Portals sollte generalisiert sein. So ergeben sich für die gewünschte Anwendung drei Softwarekomponenten, welche in Abbildung 3.2 zu sehen sind. Bedingt durch die Entwicklung einer Anwendung für das Betriebssystem eines Fernsehers, ist eine Hard- und Softwareanalyse der zur Verfügung stehenden Systeme nötig. Eine Zusammenfassung der Analyse ist im folgenden Kapitel zu finden.

3.3 Hard- und Softwareanalyse

Abbildung 3.3: Gegenüberstellung der Produktgruppen Apple TV (links), Amazon Fire TV (mitte) und Android TV (rechts), vgl. [App15b], [Gam15], [AOL15b], [Goo15b], [AOL15a], [The15], [NVI15]

Betrachtet man den aktuellen Markt an verfügbaren Betriebssystemen für den Fernseher, welche die Entwicklung von Anwendungen erlauben, so lassen sich insgesamt drei Systeme finden. In Abbildung 3.3 sind diese gegenübergestellt. Pionier der Betriebssysteme für den Fernseher stellt dabei *Apple TV* dar. *Apple TV* wird als kleine Box ausgeliefert und kann mittels HDMI-Kabel an aktuelle Fernseher angeschlossen werden. Während bereits sehr viele Firmen wie *Netflix* oder *WATCHEVER* Anwendungen für *Apple TV* besitzen, gibt es zum Zeitpunkt der Arbeit noch immer kein öffentliches *SDK* [App15a].

Alternativen zu *Apple TV* gibt es auf Basis des Android-Betriebssystems. Zunächst bietet *Amazon* mit dem entwickelten *Fire OS* ein Betriebssystem als Erweiterung von *Android 4.4* an. Dabei stehen insbesondere die hauseigenen Dienste wie *Amazon Instant Video*, *Amazon Music* und *Amazon Cloud Drive* im Fokus des Systems. Auch *Fire TV* wird analog zu *Apple TV* als Box oder Stick an den Fernseher angeschlossen. *Amazon* bietet im Vergleich zu *Apple* jedoch auch Entwicklern die Möglichkeit an, Anwendungen für den *Fire TV* zu entwickeln. Hierzu wurden eigene API's für die Benachrichtigung auf dem Fernseher und die Anbindung des Game-Controllers veröffentlicht. Dennoch besitzt das entwickelte System auch einige Einschränkungen für Entwickler von Anwendungen. Beispielsweise verweigert *Amazon* den Entwicklern den Zugriff auf das Mikrofon, was die Eignung des Betriebssystems für diese Arbeit ausschließt [Ama15a] [Ama15b].

Das jüngste System auf dem Markt stellt *Android TV* dar. Auch *Android TV* kann als Erweiterung an vorhandene Fernseher mittels Box angeschlossen werden. Neben dem Android-TV-Referenzgerät namens *Nexus Player* bieten auch Firmen wie *Razer* und *Nvidia* eigene Spielekonsolen mit dem Betriebssystem an. Im Vergleich zu den bereits erwähnten Alternativsystemen ist *Android TV* nicht nur als Erweiterung für den Fernseher vorgesehen. So bieten die Hersteller *Philips*, *Sharp* und *Sony* auch Fernseher mit dem Betriebssystem an. Da *Android TV* keine klassischen Benachrichtigungen unterstützt, wird ein neues Konzept benötigt, welches im nachfolgenden Abschnitt genauer beschrieben ist [Goo15c].

3.4 Erweitertes Konzept

Um die bereits im Kapitel 3.2 geschilderten Wünsche des Probanden unter *Android TV* vollständig erfüllen zu können, ist eine Erweiterung des Grundkonzepts nötig. Das in Abbildung 3.4 dargestellte Konzept sieht neben den bereits vorgestellten Softwarekomponenten weitere Bibliotheken und Anwendungen vor. Zentrale Idee der Erweiterung ist die Benachrichtigung des Benutzers über neue Umfragen mittels Smartwatch. Da zur Kommunikation mit der Smartwatch eine Anwendung auf dem Smartphone nötig ist, sieht das

Konzept neben der *Android Wear* Anwendung auch eine klassische *Android* Anwendung vor. Damit Komponenten zur Kommunikation über *UPnP* und der *Wearable.DataAPI* ebenfalls auf sämtlichen Anwendungen zur Verfügung stehen, berücksichtigt die Planung auch hier eine Abstraktion mittels Bibliotheken. Eine ausführlichere Beschreibung der ausgearbeiteten Softwarekomponenten findet sich im Abschnitt 4.

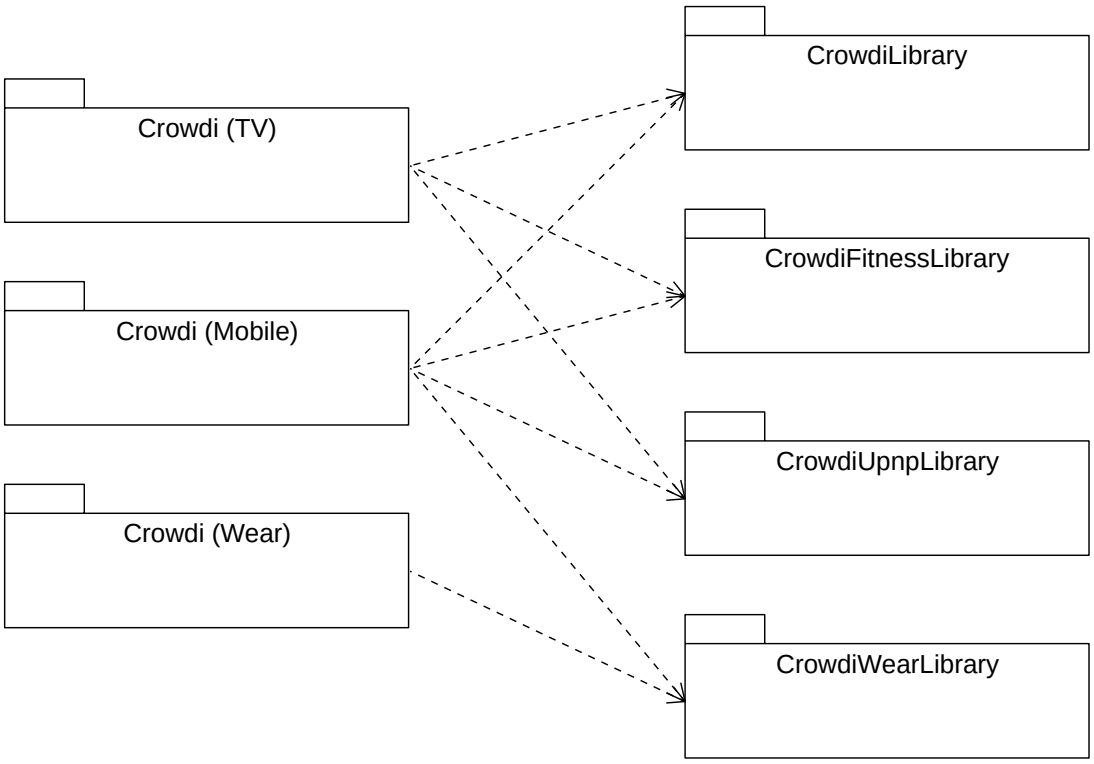

Abbildung 3.4: Erweitertes Konzept der Anwendung

4

Implementierung

Die im Nachfolgenden beschriebenen Softwarekomponenten gliedern sich in Bibliotheken und Anwendungen. Während in 4.1 die generischen Softwarekomponenten beschrieben werden, befinden sich im anschließenden Abschnitt die Endanwendungen.

4.1 Bibliotheken

Die in den nachfolgenden Kapiteln beschriebenen *Android*-Bibliotheken sind in ihre Anwendungsgebiete untergliedert. Zu Beginn findet sich die Beschreibung der elementarsten Bibliothek.

4.1.1 CrowdiLibrary

Die *Android*-Bibliothek *CrowdiLibrary* stellt eine Abstraktion der Anwendungslogik rund um das Themenfeld *CrowdSensr* dar. Zentrale Aufgabe der Bibliothek ist die Anbindung von *Android*-Applikationen an das proprietäre System. Die dazu nötigen Pakete und Klassen sind in Abbildung 4.1 dargestellt. Zentrale Schnittstelle der *Android*-Bibliothek stellt die Klasse *Library* dar.

Der Singleton[1], welcher über die Funktion `Library.getInstance()` erzeugt werden kann, bietet der *Android*-Anwendung unter anderem die Funktionen zum Prüfen verfügbarer Umfragen. Bei der Entwicklung der Schnittstelle wurde darauf geachtet,

[1] Als Singleton wird in der Softwareentwicklung ein Pattern bezeichnet, welches sicherstellt, dass es von einer Klasse nur eine Instanz geben kann

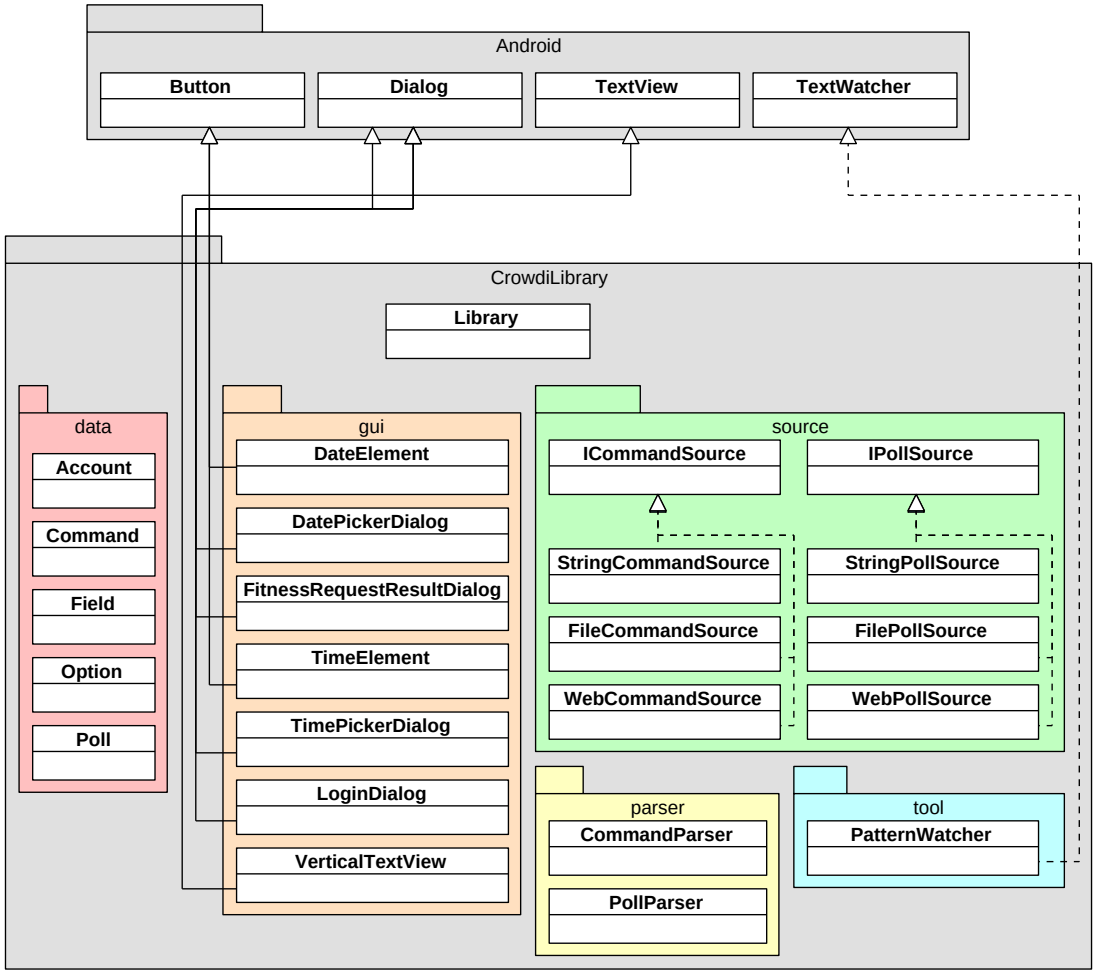

Abbildung 4.1: Klassendiagramm der CrowdiLibrary

eine möglichst lose Anbindung an die *CrowdSensr*-Plattform zu bieten. Deshalb wurde die benötigte Quelle als Parameter in die Methodensignatur integriert. Um mehrere Arten von Quellen zu unterstützen, beinhaltet die Bibliothek das Interface *ICommandSource*. Damit ist die Bibliothek in der Lage, neben einer Webquelle (*WebCommandSource*) auch lokale Dateien (*FileCommandSource*) und reine Zeichenketten (*StringCommandSource*) zu akzeptieren. Wird beim Aufruf der Schnittstellenfunktion `Library.loadCommands(String source, String className)` eine URL übergeben, so verwendet die Bibliothek die *WebCommandSource*, um die Verfügbarkeit von Umfragen zu prüfen. Stellt die Quelle hingegen einen lokalen Pfad dar, kann auf

die Instanz *FileCommandSource* zurückgegriffen werden. Sollte die Quelle weder eine URL noch einen Pfad darstellen, geht die Bibliothek von einer reinen Zeichenkette aus. Die alternativen Quellen können von Entwicklern insbesondere zum Testen von neuen Funktionen verwendet werden. Möchte dieser beispielsweise die Darstellung einer neuen Umfrageliste überprüfen, so kann dies über einen lokale String-Variable erreicht werden. Regulär müsste die *JSON*-Struktur, welche in Abschnitt 2.4 beschrieben ist, als Ressource auf dem Webserver geladen und mittels URL übergeben werden. Eine Veranschaulichung des Prüfprozesses ist in Form eines Sequenzdiagramms in Abbildung 4.2 dargestellt.

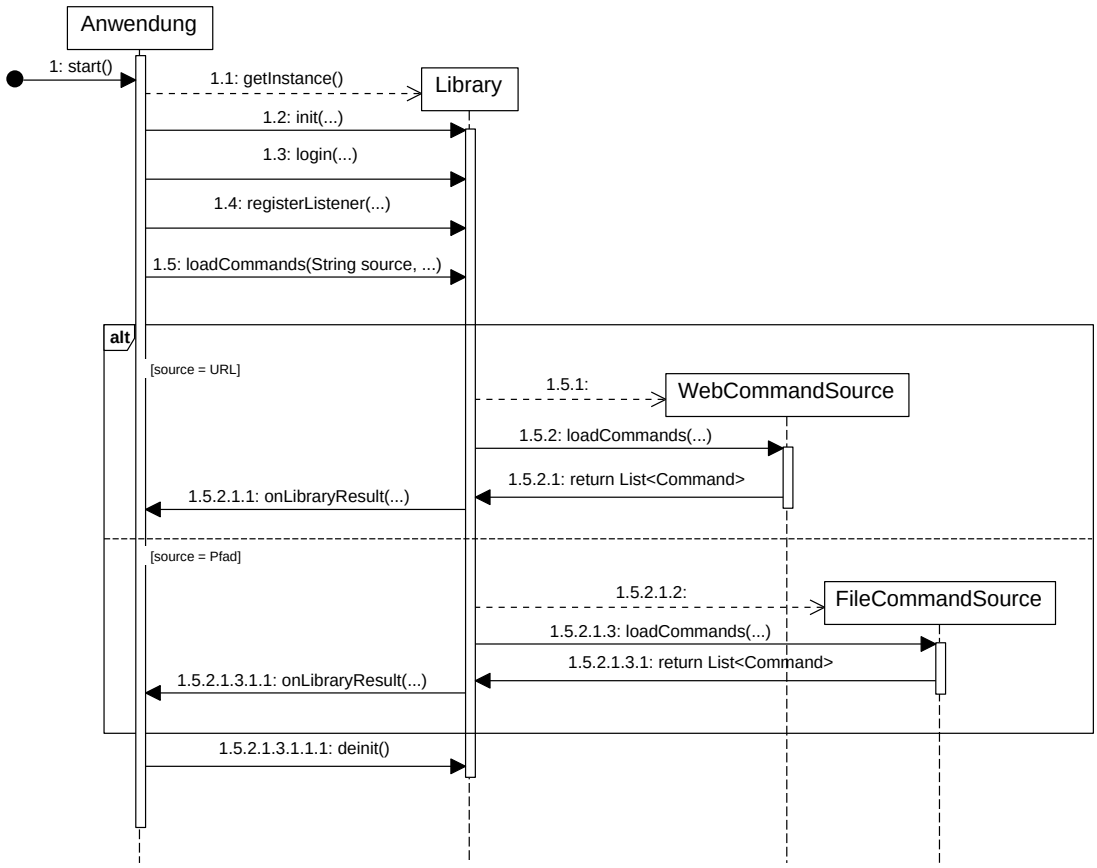

Abbildung 4.2: Sequenzdiagramm zum Abrufen der verfügbaren Umfragen

Da Netzwerkoperationen unter *Android* nicht im *Thread* der Anwendung ausgeführt werden dürfen, wird der Prüfprozess der Bibliothek asynchron ausgeführt. Damit die Anwendung über das Ergebnis des Prüfprozesses informiert werden kann, wird ein entsprechender *Callback-Mechanismus*[2] benötigt. Dazu muss die Anwendung das Interface *ILibraryResultListener* implementieren und sich vor dem Aufruf der Methode `Library.loadCommands(String source, String className)` über die Funktion `Library.registerListener(ILibraryResultListener listener)` für Updates bei der Bibliothek registrieren. Ein Update zur Abfrage wird der Anwendung fortan über den Aufruf der Interface-Funktion `ILibraryResultListener.onLibraryResult(ExecutionStatus exStatus, List<Command> commands, String className)` mitgeteilt. Dabei kann die Anwendung anhand des *ExecutionStatus* prüfen, ob die Abfrage erfolgreich war. Die zur Verfügung stehenden Umfragen werden der Anwendung anhand einer Liste von *Commands* dargestellt. Dabei kapselt die Klasse *Command* die über *JSON* übertragenen Informationen: *Command*, *Method* und *Info*. Da sich mehrere Klassen an der Bibliothek registrieren können und immer alle Instanzen informiert werden, wird der Klassenname als Identifier für den Verantwortlichen übergeben.

Neben dem Prüfen der verfügbaren Umfragen bietet die Bibliothek ebenfalls die Möglichkeit, entsprechende Umfragen abzurufen. Die dafür entwickelte Schnittstelle folgt dabei derselben Anwendungslogik. Dafür wurde ebenfalls ein entsprechendes *Interface* (*IPollSource*) erstellt. Um einen identischen Funktionsumfang bieten zu können, stellt die Bibliothek auch für die Umfrage die Implementierungen *WebPollSource*, *FilePollSource* und *StringPollSource* zur Verfügung. Das Abrufen und Informieren erfolgt auch hier über die bereits vorgestellten Klassen *Library* und *ILibraryResultListener*. Dabei unterscheidet sich die *Callback-Methode* lediglich in der Methodensignatur. So wird ein Update hier über die Methode `ILibraryResultListener.onLibraryResult(ExecutionStatus status, Poll poll, String className)` verteilt. Die erfolgreich erzeugte Umfrage wird der Anwendung über die Klasse *Poll* übergeben, welche eine Liste von Umfragefeldern (*Field*) enthält. Ein *Field* stellt das wesentliche Element

[2] Als Callback-Mechanismus wird in der Softwareentwicklung das Prinzip verstanden, wenn der Aufgerufene den Aufrufer zurückruft

einer Umfrage dar und kann im Falle eines *CompositeFields* eine weitere Liste mit Feldern beinhalten.

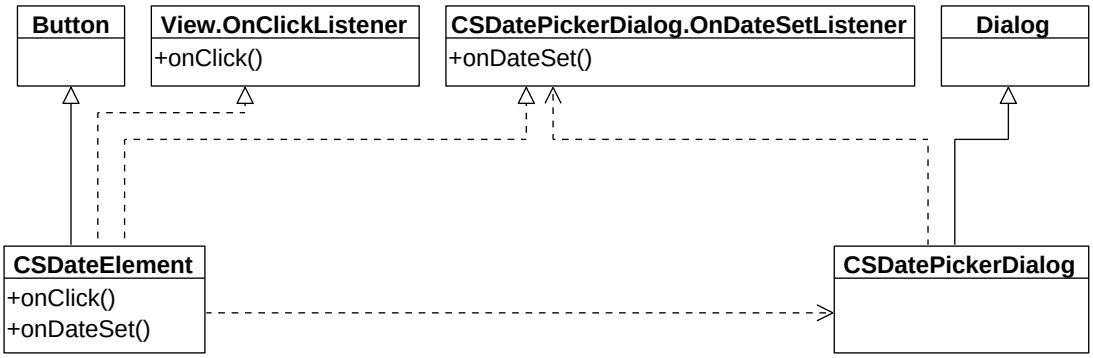

Abbildung 4.3: Klassendiagramm zum CSDateElement

Zusätzlich zur bisher beschriebenen Anwendungslogik beinhaltet die *CrowdiLibrary* ebenso eigene Oberflächenelemente, um die fehlende Unterstützung der Fernbedienung durch die Elemente *DatePicker* und *TimePicker* auszubessern. Die im Rahmen der Arbeit entwickelten Elemente *CSDateElement* und *CSTimeElement* stellen eine Erweiterung einfacher *Buttons*[3] mit einer entsprechenden Beschriftung dar. Das jeweilige Element implementiert bereits das *Interface View.OnClickListener* und ist damit in der Lage, auf das Drücken des Elements zu reagieren. Wählt der Benutzer das Element durch einen Klick aus, so wird ein entsprechender Dialog dargestellt (*CSDatePickerDialog* oder *CSTimePickerDialog*). Wie in Abbildung 4.3 zu sehen ist, stellen diese Elemente Erweiterungen des regulären *Dialogs* dar. Das für die *Dialoge* selbst entwickelte Layout beruht auf dem Prinzip der ehemaligen *Picker* und ist in Abbildung 4.4 konzeptionell dargestellt.

So gibt es auf dem *CSDatePickerDialog* insgesamt 10 *Buttons*. Während drei *Buttons* zur Darstellung der Eigenschaften: Tag, Monat und Jahr dienen, gibt es für die jeweilige Eigenschaft je einen *Button* zum Erhöhen und Verringern des aktuellen Wertes. Um eine möglichst effiziente Bedienung mit der Fernbedienung zu realisieren, wurden die

[3] Oberflächenelement, das geklickt werden kann

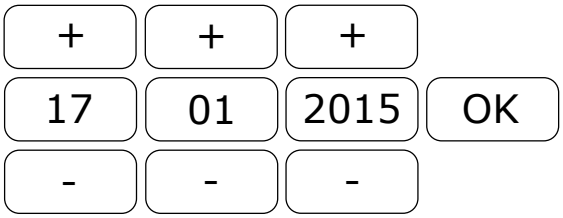

Abbildung 4.4: Konzept des CSDatePickerDialogs

Button zum Erhöhen und Verringern der Werte für die Fernbedienung deaktiviert. Eine Manipulation der entsprechenden Werte kann auf dem Fernseher, durch das Auswerten der *Key-Events*, mittels der Tasten “Auf“ und “Ab“ der Fernbedienung vorgenommen werden. Eine beispielhafte Darstellung der ablaufenden Prozesse ist im Sequenzdiagramm 4.5 dargestellt.

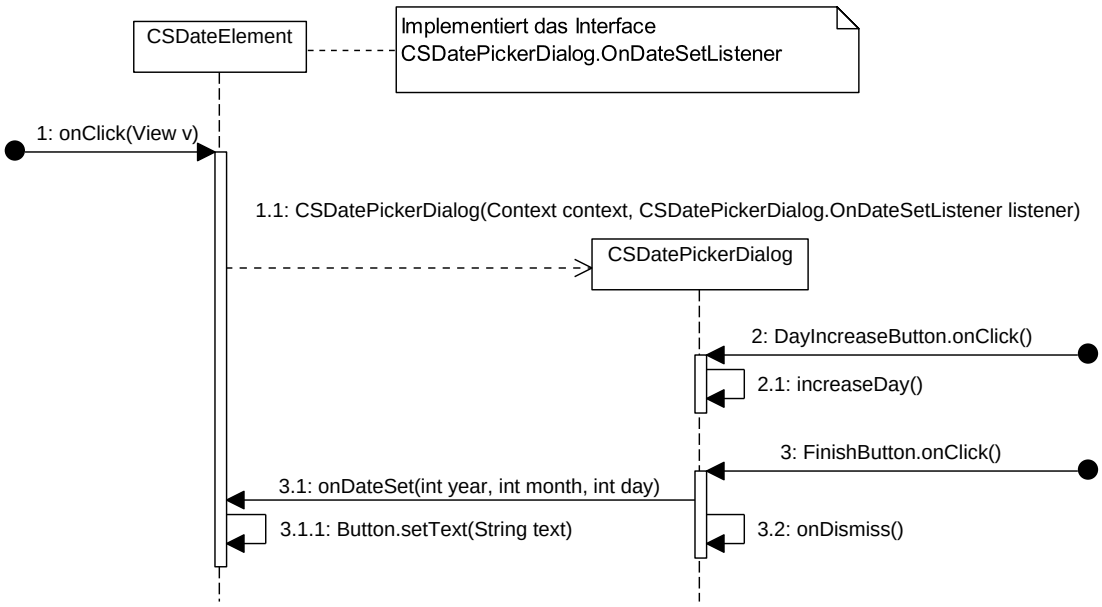

Abbildung 4.5: Sequenzdiagramm zum CSDateElement

Neben den Oberflächenelementen, die zur Ausbesserung von Elementen mit fehlender Unterstützung der Fernbedienung vorgesehen sind, enthält die Bibliothek ebenfalls ein Oberflächenelement, das als Ergänzung der von *Andorid* angebotenen Elemente anzusehen ist. Mit dem *CSFitnessRequestResultDialog* bietet die *CrowdiLibrary* einen

Dialog an, der Daten in Diagrammform darstellen kann. Eine konzeptionelle Darstellung des *CSFitnessRequestResultDialogs* ist in Abbildung 4.6 zu sehen. Zur Erstellung eines solchen Diagramms müssen die entsprechenden Parameter im Konstruktor des *CSFitnessRequestResultDialogs* übergeben werden.

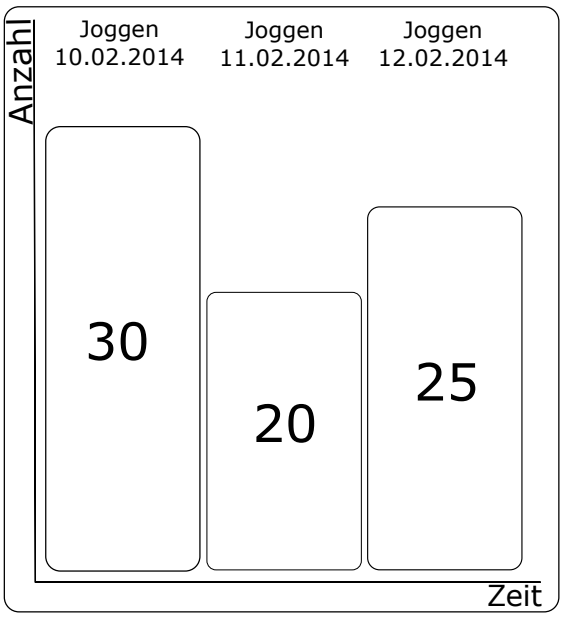

Abbildung 4.6: Konzept des CSFitnessRequestResultDialogs

Neben den Beschriftungen der horizontalen und vertikalen Achse erwartet das Diagramm zudem zwei Listen bestehend aus *Strings*. Während die erste Liste mit Zeichenketten zur Beschriftung der Diagrammelemente am oberen Rand dient, werden in der zweiten die Werte der Elemente erwartet. Bei der Erzeugung des *Dialogs* wertet das *CSFitness-RequestResultDialog* die übergebenen Werte aus. Handelt es sich bei den übergebenen Werten um Zahlen, die als Zeichenketten kodiert sind, so werden die Elemente anhand ihres Werts skaliert. Dabei bildet das Element mit dem höchsten Wert auch das Maximum des Diagramms. Damit zur Darstellung von Fitness-Daten keine Abhängigkeit zur *CrowdiFitnessLibrary* entsteht, wurde an dieser Stelle bewusst der Einsatz von einfachen Zeichenketten gewählt.

4.1.2 CrowdiFitnessLibrary

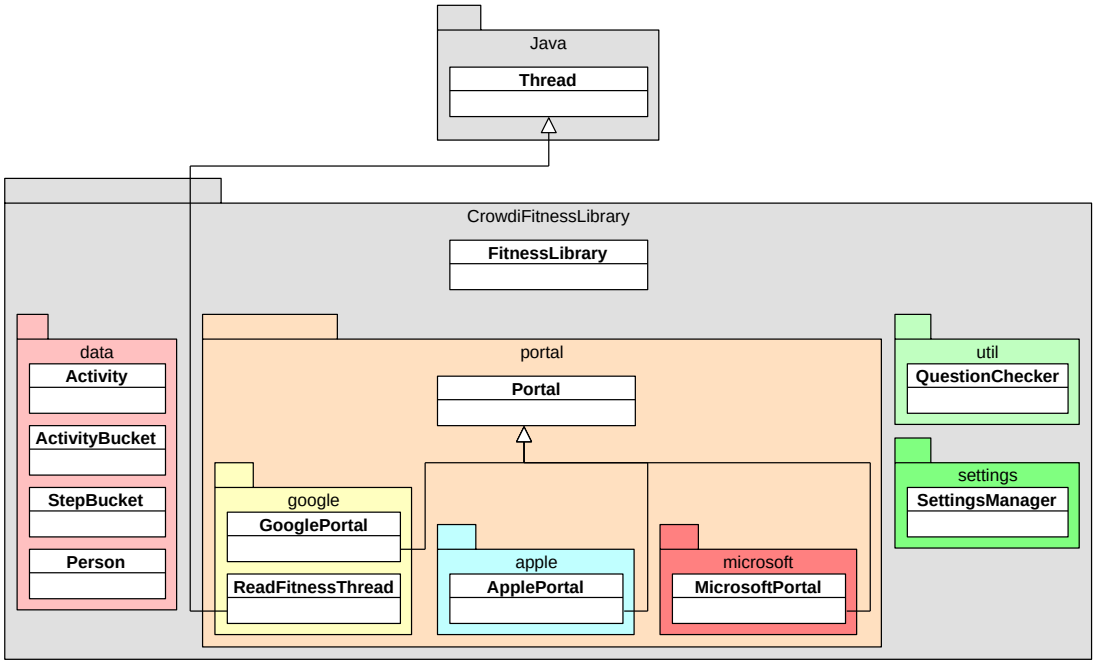

Abbildung 4.7: Klassendiagramm der CrowdiFitnessLibrary

Auch die *CrowdiFitnessLibrary* ist als *Android*-Bibliothek umgesetzt. Das Themenfeld der Bibliothek umfasst hier insbesondere die Anbindung von *Android*-Anwendungen an Fitness-Portale. Im Vergleich zu den durch die Hersteller angebotenen Bibliotheken bietet die *CrowdiFitnessLibrary* eine einheitliche Schnittstelle sämtlicher Portale an. Dazu werden die angebotenen Bibliotheken eingebunden und mittels Adapter auf die eigene Schnittstelle übersetzt.

Analog zur *CrowdiLibrary* besitzt auch die *CrowdiFitnessLibrary* einen *Singleton* (*FitnessLibrary*), der zur Bearbeitung der Anfragen herangezogen werden kann. Nach der Erzeugung der *Singleton*-Instanz über die statische Methode `FitnessLibrary.getInstance()` muss ebenfalls eine Initialisierung der Bibliothek mit dem *Context*[4] der *Android*-Anwendung erfolgen. Dies wird einheitlich verlangt, da die angebundenen Bibliotheken oftmals einen entsprechenden *Context* zur Erzeugung von Oberflächenelemente,

[4]Eine Klasse des Android-Betriebssystem die zur Einblendung von Oberflächenelemente benötigt wird

wie *Dialogen*, benötigen.

Um eine einheitliche Schnittstelle für alle Fitness-Portale anbieten zu können, beinhaltet die Bibliothek die abstrakte Klasse *Portal*. Diese Klasse definiert die vom System bekannten Ausprägungen mittels *PortalType*. Soll ein weiteres Portal durch die Bibliothek unterstützt werden, so muss eine entsprechende Erweiterung des *Enums*[5] erfolgen. Im Rahmen der Arbeit sind an dieser Stelle die Ausprägungen *Google*, *Apple* und *Microsoft* für die Portale *Google Fit*, *Apple Health Kit* und *Microsoft HealthVault* definiert. Neben den Ausprägungen legt die Klasse *Portal* durch die definierten, abstrakten Methoden auch die zu unterstützende Schnittstelle fest.

So definiert die Klasse derzeit drei Methoden zum Abfragen von Fitnessdaten. Zunächst definiert die Methode `getPerson()` die verfügbare *API* zum Abfragen von Personeninformationen. Damit die erhaltenen Informationen auch einheitlich zurückgegeben werden können, definiert die Bibliothek die Klasse *Person* zur Darstellung der Personeninformationen. In dieser Klasse werden die elementaren Attribute: *Name*, *Größe* und *Gewicht* der Person abgebildet. Neben der Abfrage von Personeninformationen unterstützt die Bibliothek auch die Abfrage der sportlichen Aktivitäten. Dazu erwartet die Bibliothek in der dafür entwickelten Schnittstelle eine Reihe von Parameter, die im Codeabschnitt 4.1 zu sehen sind.

```
1 FitnessLibrary.getActivities(Portal.PortalType type,
2                               long startTime,
3                               long endTime,
4                               TimeUnit rangeUnit,
5                               int duration,
6                               TimeUnit durationUnit,
7                               int requestId);
```

Listing 4.1: API zur Abfrage der sportlichen Aktivitäten

[5] Enums stellen in der Softwareentwicklung einfache Datentypen dar

Damit eine Eingrenzung der Daten möglich ist, muss die Anwendung über die Parameter `startTime` und `endTime` einen Zeitbereich definieren, dessen Einheit über `rangeUnit` festgelegt wird. Damit eine Gruppierung der Fitnessdaten möglich ist, erwartet die Schnittstelle mit den Parametern `duration` und `durationUnit` genauere Zeitangaben über die darzustellenden Gruppen. Möchte man beispielsweise wissen, welche sportlichen Aktivitäten man pro Tag in der letzten Woche absolviert hat, so kann als `duration` der Zahlenwert `1` und die Zeiteinheit `TimeUnit.DAYS` übergeben werden. Die durch das jeweilige Portal empfangenen Daten müssen anschließend über die Klassen *Activity* und *ActivityBucket* einheitlich übersetzt und zurückgegeben werden.

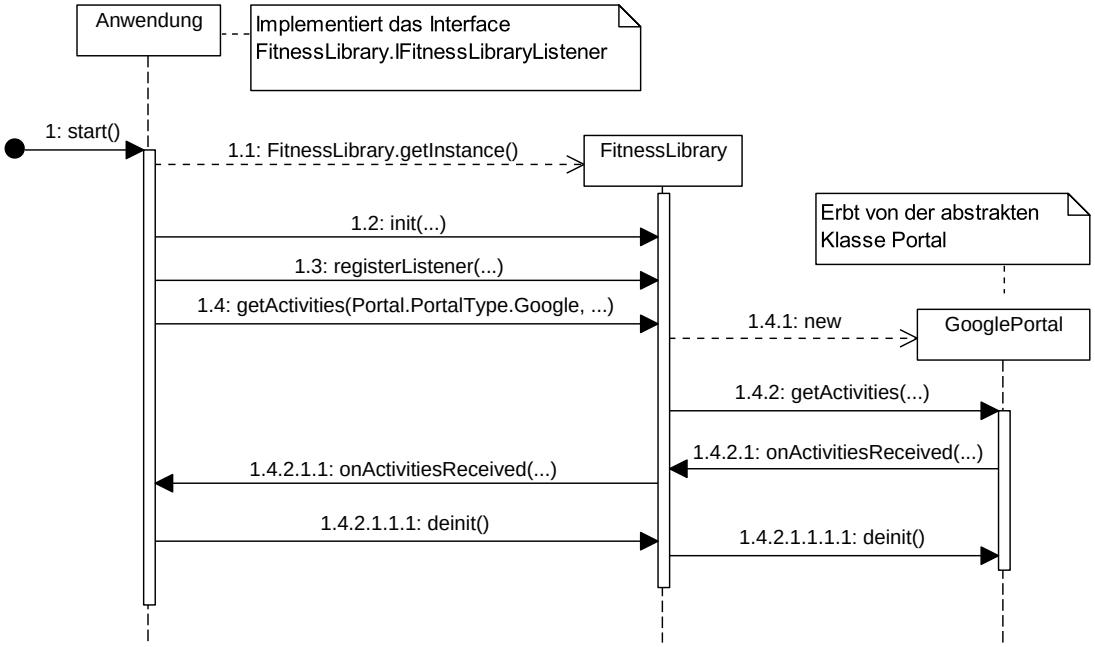

Abbildung 4.8: Sequenzdiagramm zur Abfrage verfügbarer Aktivitäten

Eine vergleichbare Schnittstelle bietet die Bibliothek zur Abfrage der zurückgelegten Schritte während einer Zeitspanne an. Da die Fitnessdaten der Portale nicht lokal auf den Geräten gespeichert sind, müssen die Netzwerkanfragen analog zur *CrowdiLibrary* über einen *Callback-Mechanismus* realisiert werden. Auch in dieser Bibliothek ist dazu ein *Interface IPortalListener* definiert, welches über die in der Bibliothek angebotene

Funktion `Portal.registerListener(IPortalListener listener)` registriert werden muss. Eine beispielhafte Darstellung der ablaufenden Prozesse ist im Sequenzdiagramm 4.8 dargestellt. Im Rahmen der Arbeit wurde für die beschriebene Schnittstelle ein Adapter als Referenzimplementierung ausgearbeitet. Dazu findet die von *Google* veröffentlichte *Android*-Bibliothek für das Portal *Google-Fit* Anwendung [Goo15h].

4.1.3 CrowdiUpnpLibrary

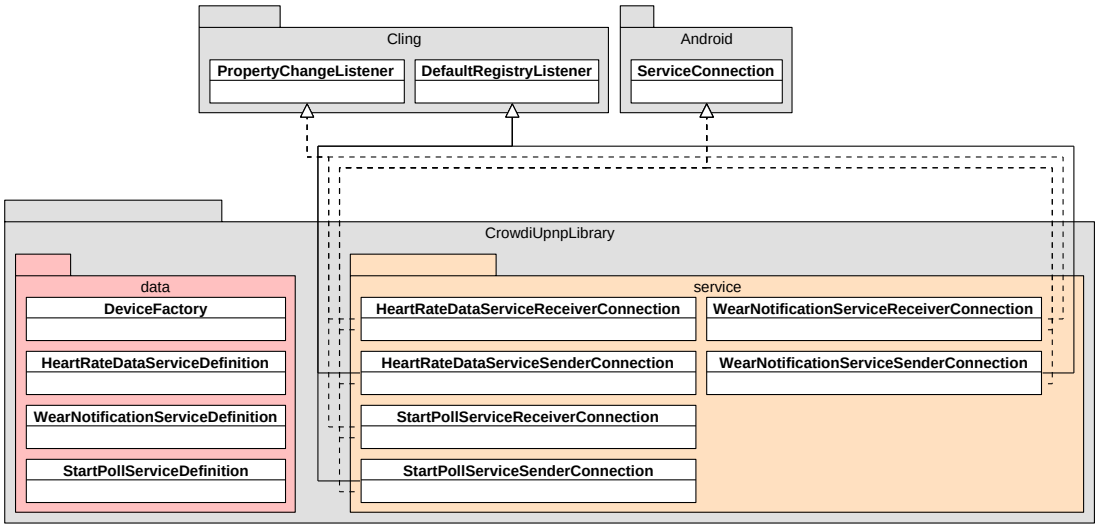

Abbildung 4.9: Klassendiagramm der CrowdiUpnpLibrary

Bei der *Android*-Bibliothek *CrowdiUpnpLibrary* handelt es sich im Wesentlichen um eine Implementierung der *UPnP*-Bibliothek von der *4th Line GmbH* [4th15]. Die unter dem Namen *Cling* veröffentlichte Bibliothek bietet alle nötigen Schnittstellen zum Suchen und Finden von *UPnP*-Geräten innerhalb eines Netzwerkes. Neben dieser elementaren Funktionen bietet *Cling* weitere Möglichkeiten an. So können eigene *UPnP*-Dienste definiert und anschließend veröffentlicht werden. Dazu müssen reguläre Java-Klassen mit Annotationen[6] der Bibliothek versehen werden. Anhand dieser Annotationen kann *Cling* eine entsprechende *XML*-Struktur für den zugrunde liegenden *SOAP*-Zugriff erstellen. Durch die Annotationen *UpnpServiceId* und *UpnpServiceType* kann der Name

[6] Annotationen sind Erweiterungen die am @-Zeichen erkannt und an Methoden und Variablen angebracht werden können

und Typ des angebotenen *UPnP*-Dienstes festgelegt werden. Variablen und Methoden des Service sind über *UpnpStateVariable* und *UpnpAction* zu definieren. Um die Ein- und Rückgabewerte der Methoden zu definieren, müssen die Annotationen *UpnpInputArgument* und *UpnpOutputArgument* an der entsprechenden *UpnpAction* verwendet werden. [4th15]

Über den beschriebenen Mechanismus definiert die *CrowdiUpnpLibrary* insgesamt drei *UPnP*-Dienste. Um Benachrichtigungen auf ein *Android*-Gerät übertragen zu können, ist ein entsprechender Dienst über die Klasse *WearNotificationServiceDefinition* festgelegt. Dieser Dienst besitzt zwei Methoden, welche über *SOAP* angesprochen werden können. Zunächst kann über die Methode `WearNotificationServiceDefinition.startNotification(String title, String content, String url)` eine Benachrichtigung über eine Umfrage gesendet werden. Dazu kann der Absender über die Parameter `title`, `content` und `url` die wichtigsten Informationen an das *UPnP*-Gerät übermitteln. Neben dieser Methode bietet der Dienst auch die Möglichkeit an, eine spezifische Benachrichtigung zur Messung der Herzfrequenz zu übermitteln. Die dafür festgelegte Methode `getHeartRate()` erwartet keine weiteren Parameter.

Damit ein *UPnP*-Gerät die Anfrage nach der Herzfrequenz nicht sofort beantworten muss, bietet die Bibliothek einen weiteren Dienst. Dieser sollte zum Empfangen der Herzfrequenz vom Absender der Benachrichtigung implementiert werden. Der durch die Klasse *HeartRateDataServiceDefinition* festgelegte *UPnP*-Dienst beinhaltet lediglich eine Methode zum Übertragen der Herzfrequenz. Dabei wird der Messwert als *String* an den Service übermittelt.

Der dritte *UPnP*-Dienst ist ebenfalls als Rückkanal zum bereits vorgestellten Benachrichtigungsdienst zu sehen. Mit dem über die Klasse *StartPollServiceDefinition* festgelegten Service kann der Empfänger einer Umfrage-Benachrichtigung die entsprechende Umfrage beim Absender starten. Dazu muss die ausgewählte URL zur Umfrage als

Eingabeparameter der *UpnpAction* `startPoll(String)` übergeben werden.

Damit die definierten Dienste nutzbar sind, müssen diese über den von *Cling* angebotenen *Android-Service* hinzugefügt und publiziert werden. Zu diesem Zweck definiert die *CrowdiUpnpLibrary* für jeden der drei Dienste eine entsprechende *ServiceConnection*. Damit eindeutig differenziert werden kann, auf welchem Gerät der Dienst angeboten wird und wo er nutzbar ist, sind für jeden *UPnP*-Dienst je zwei *ServiceConnections* definiert.

Anbieter eines Dienstes können über den Einsatz einer entsprechenden *Receiver-Connection* das automatische Hinzufügen und Publizieren bewirken. Die Verwendung einer *SenderConnection* bewirkt hingegen, dass die Anwendung über neue *UPnP*-Dienste dieses Typus informiert wird und diesen über die öffentlichen Methoden der *ServiceConnection* nutzen kann. Abbildung 4.10 zeigt eine beispielhafte Nutzung der angebotenen *UPnP*-Dienste über die vorgestellten *ServiceConnections*.

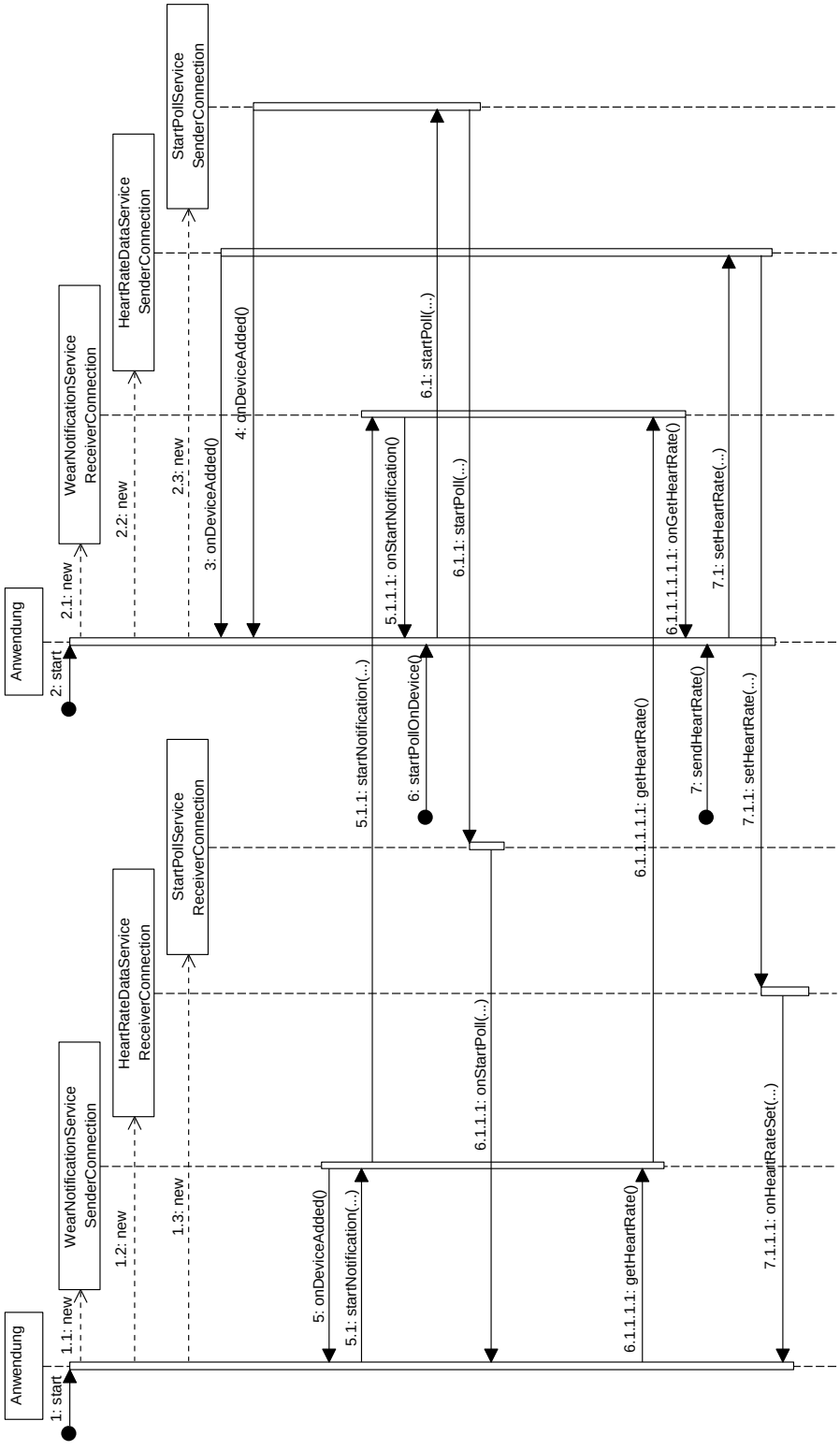

Abbildung 4.10: Benachrichtigung über neue Umfragen mittels UPnP, Abrufen der Herzfrequenz über UPnP

4.1.4 CrowdiWearLibrary

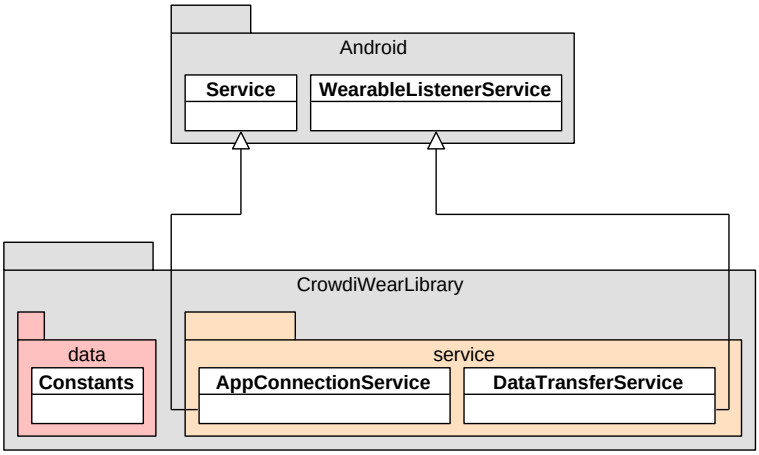

Abbildung 4.11: Klassendiagramm der CrowdiWearLibrary

Die in der *CrowdiWearLibrary* zusammengefasste Anwendungslogik konzentriert sich auf den Datenaustausch zwischen einer *Android*-Anwendung und einer *Android-Wear*-Anwendung. Die für den Datenaustausch zugrundeliegende *Wearable.DataApi* ist bereits im Kapitel 2.7.3 eingehend beschrieben. Zur einfacheren Anbindung der *API* beinhaltet die Bibliothek die Klasse *Constants*, welche auf der Sender- und Empfänger-Seite verwendet werden kann. Damit kann sichergestellt werden, dass sowohl Absender als auch Empfänger die identischen Werte innerhalb des Nachrichtenpakets verwenden.

Um eine vereinfachte Verwendung der *Wearable.DataApi* zu ermöglichen, besitzt die Bibliothek die Klasse *AppConnectionService*. Diese stellt eine Erweiterung des klassischen *Service* von *Android* dar und kann beliebig gestartet und gestoppt werden. Um den durch die Bibliothek von *Android-Wear* zur Verfügung gestellten *WearableListenerService* zum Datenaustausch nutzen zu können, besitzt die *CrowdiWearLibrary* die Klasse *DataTransferService*. Diese stellt eine Erweiterung des besagten Dienstes dar und leitet die empfangenen Nachrichten über den *Broadcast-Mechanismus* des Systems weiter. Die bereits vorgestellte Klasse *AppConnectionService* besitzt folglich einen *BroadcastReceiver*, um über die weitergeleiteten Nachrichten informiert zu werden.

Damit eine Anwendung die Funktion des Dienstes nutzen kann, muss diese einen eigenen *Service* durch die Erweiterung des *AppConnectionService* erstellen. Die hierdurch vererbten Methoden `AppConnectionService.onDataReceived(String data)` und `AppConnectionService.onStartOnTvReceived(String url)` werden zur Laufzeit beim Empfang einer entsprechenden Nachricht durch den *BroadcastReceiver* aufgerufen. Nachrichten, die von der Anwendung auf dem Smartphone an die Uhr gesendet werden sollen, können ebenfalls über diesen Hintergrunddienst versendet werden. Dazu muss lediglich die entsprechende Implementierung des *AppConnectionServices* gebunden und die vererbte Methode `AppConnectionServices.showNotification(String title, String url)` aufgerufen werden. Eine beispielhafte Verwendung der genannten Komponenten ist im Sequenzdiagramm 4.12 dargestellt.

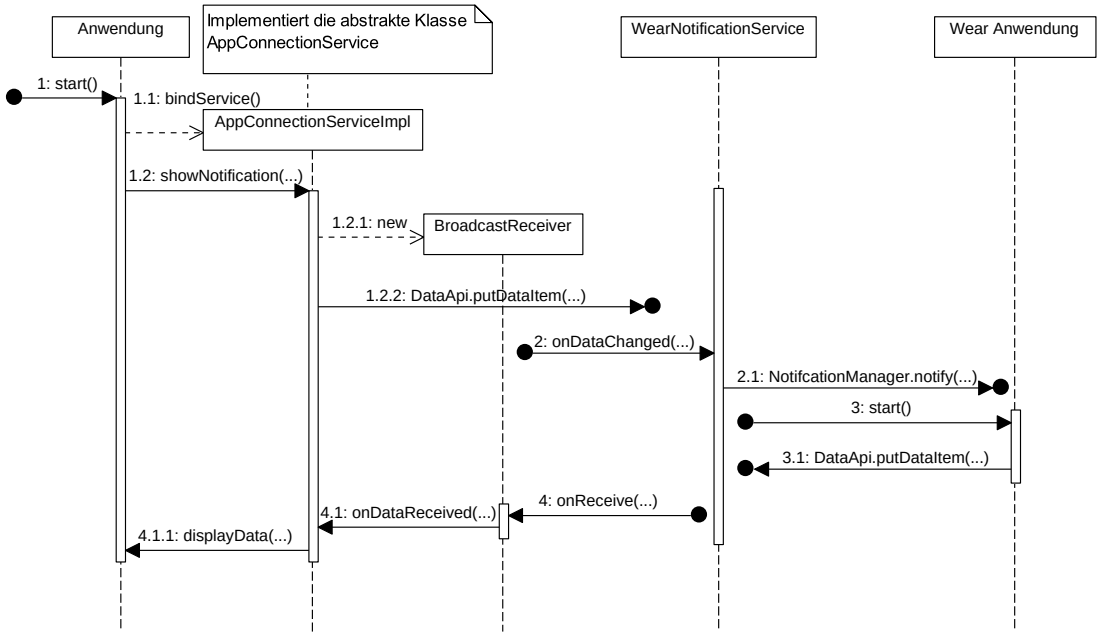

Abbildung 4.12: Verwendung des AppConnectionService

4.2 Anwendungen

Die in den nachfolgenden Kapiteln beschriebenen *Android*-Anwendungen sind ebenfalls nach den Gerätetypen untergliedert. Zu Beginn findet sich die Beschreibung der Smart-TV-Anwendung.

4.2.1 Crowdi (TV)

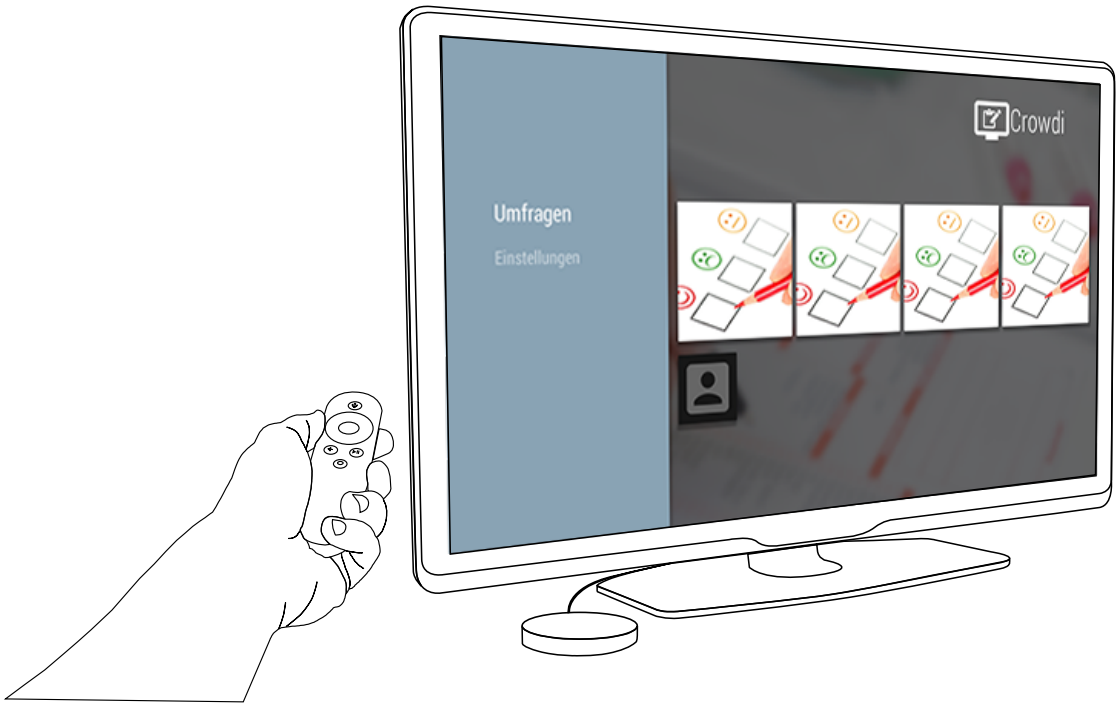

Abbildung 4.13: Startscreen der Anwendung Crowdi (TV)

Crowdi (TV) stellt eine Anwendung auf Basis von *Android 5.1* dar, die mit Hilfe der *Support-Library* von *Google* für den Fernseher optimiert ist. In Abbildung 4.13 ist der Startbildschirm der Anwendung zu sehen. Die mit Hilfe des *BrowseFragments* umgesetzte Benutzeroberfläche sieht zwei unterschiedliche Reihen von Elementen vor. Während die obere Reihe zur Darstellung der verfügbaren Umfragen angedacht ist, stellt die untere Reihe eine Auflistung der angebotenen Einstellungen dar.

Umgesetzt ist diese Benutzeroberfläche mit dem im Abschnitt 2.6.5 vorgestellten *ArrayObjectAdapter*. Zur Darstellung der unterschiedlichen Elemente wurden zwei *Presenter* implementiert. Während die Klasse *CommandPresenter* eine Möglichkeit zur Darstellung von verfügbaren Umfragen bietet, können Einstellungen mit Hilfe der Klasse *ActionPresenter* angezeigt werden. Damit eine Anzeige der verfügbaren Umfragen möglich ist, implementiert die *Activity* das in der *CrowdiLibrary* definierte Interface *ILibraryResultListener*.

Nach dem erfolgreichen Start der *Activity* wird ein entsprechender *HTTP-GET* auf den Server mit Hilfe der *CrowdiLibrary*-API ausgeführt. Nachdem die Bibliothek die verfügbare Umfrage-Liste empfangen hat, signalisiert sie dies der *Activity* über die *Interface*-Funktion. Daraufhin kann durch die *Activity* ein Update des *ArrayObjectAdapters* erfolgen. Möchte der Proband eine dargestellte Umfrage auswählen, so muss dieser zunächst über die “Select“-Taste der Fernbedienung die Reihe “Umfragen“ selektieren. Daraufhin verbirgt sich die im Bild links zu sehende Menüleiste und ermöglicht so die Darstellung relevanter Titel- und Inhaltsinformationen der Umfragen. Wählt der Benutzer nun eine Umfrage aus der Liste aus, so wird die zur Darstellung der Umfrage benötigte URL aus dem *Command* entnommen und mittels *Intent* an die anschließende *SinglePollActivity* weitergereicht.

Diese entnimmt die im *Intent* übertragene URL und ruft die Umfrage mittels *CrowdiLibrary* ab. Analog zur *MainActivity* wird auch die *SinglePollActivity* über die *Callback-Methode* des implementierten *ILibraryResultListener*-Interfaces informiert. Ist die Abfrage erfolgreich, so werden die in der Klasse *Poll* gekapselten *Field*-Elemente ausgelesen und mit Hilfe von standardmäßigen Oberflächenelementen dargestellt. Für die von *Android TV* nicht unterstützten Typen werden eigene Oberflächenelemente aus der *CrowdiLibrary* verwendet. Tabelle 4.1 zeigt die Abbildung der durch *CrowdSensr* definierten Elemente.

Tabelle 4.1: Abbildung der Umfrageelemente

Typ	**Android**
text	CSEditTextGroup
textarea	CSEditTextGroup + InputType.TYPE_TEXT_FLAG_MULTI_LINE
password	EditText + InputType.TYPE_TEXT_VARIATION_PASSWORD
number	CSEditTextGroup + InputType.TYPE_CLASS_NUMBER
email	EditText + InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS
tel	EditText + InputType.TYPE_CLASS_PHONE
url	EditText + InputType.TYPE_TEXT_VARIATION_URI
date	CSDateElement
time	CSTimeElement
datetime	CSDateElement + CSTimeElement
select	RadioGroup
multiselect	LinearLayout + CheckBox
checkbox	CheckBox
radio	RadioGroup

Wie der Tabelle zu entnehmen ist, differenziert die Anwendung bei der textuellen Eingabe in die zwei Grundtypen *EditText* und *CSEditTextGroup*. Dabei wird der von *Android* selbst definierte Typ *EditText* bei Fragen angewendet, die keine weitere Analyse erfordern. Wird beispielsweise in einem Umfrageelement der Typ *email* verwendet, so kann die Frage nicht mit der Unterstützung der Fitness-Portale beantwortet werden. Handelt es sich hingegen um den regulären Typ *text* so besteht die Möglichkeit, dass die Frage mit Hilfe einer Fitness-Plattform zu lösen ist. Dazu wird die entsprechende Frage an den *QuestionChecker* der *CrowdiFitnessLibrary* weitergereicht. Dieser nimmt anhand vordefinierter Textbausteine eine Kategorisierung der Frage vor. Stellt er fest, dass es sich anbei um eine Frage handelt, die mit Hilfe eines verbundenen Fitness-Portals beantwortet werden kann, so wird in der *CSEditTextGroup* neben dem eigentlichen *EditText* je ein *Button* des verbundenen Fitness-Portals eingeblendet. Um eine Entscheidung im *QuestionChecker* zu ermöglichen, sind in der Bibliothek bestimmte Schlüsselwörter definiert, die als Indiz bestimmter Fragetypen herangezogen werden.

Da sich die textuelle Eingabe über die Fernbedienung als sehr umständlich erweist, muss eine verbesserte Eingabemethode angeboten werden. Dabei kann der von *Android TV* bereits etablierte Mechanismus zur Suche von Inhalten über das Mikrofon adaptiert werden. Dazu beinhaltet das Projekt *Crowdi (TV)* die Klasse *CSActivity*, welche eine Erweiterung der Klasse *Activity* darstellt. Um auf Eingaben über das Mikrofon reagieren zu können, wird in der vererbten Methode `onCreate(Bundle b)` zunächst eine Instanz der Klasse *SpeechRecognizer* erzeugt. Über diese Instanz registriert sich die Klasse anschließend mittels `SpeechRecognizer.setRecognitionListener(RecognitionListener rl)` für die Spracheingabe und muss infolge dessen das *Interface RecognitionListener* implementieren. In der ebenfalls vererbten Methode `Activity.onKeyUp(int keyCode, KeyEvent event)` kann auf den Tastendruck der Fernbedienung reagiert werden, indem die Sprachaufzeichnung über den bereits erwähnten *SpeechRecognizer* gestartet wird. Über die *Interface*-Methode `RecognitionListener.onResult(Bundle result)` kann die *Activity* die gesprochenen Wörter auslesen und verarbeiten. Damit diese im fokussierten *EditText* dargestellt werden können, muss der *CSActivity* mitgeteilt werden, welches Oberflächenelement derzeit fokussiert ist. Dies geschieht über die selbst definierte, abstrakte Methode `CSActivity.getFocusedView()` die von der entsprechenden *Activity* implementiert werden muss.

Um nicht zu viele Oberflächenelemente auf der *SinglePollActivity* darzustellen, wird auf dem Fernseher nur ein Umfrageelement mittels *Fragments* dargestellt. Zusätzlich bietet die Oberfläche eine Fortschrittsanzeige, die dem Benutzer den aktuellen Zustand der Umfrage näher bringt. In Abbildung 4.14 ist die Darstellung eines Umfrageelements auf dem Fernseher zu sehen. Wie diesem Bild zu entnehmen ist, bietet die Oberfläche zwei übergeordnete Schaltflächen, die zur Navigation innerhalb der Umfrage dienen. Ist der Proband beim letzten Umfrageelement angekommen, kann er die Umfrage über die erneute Auswahl der “Weiter“-Taste fertigstellen. Kann die ausgefüllte Umfrage über die Schnittstelle der *CrowdiLibrary* erfolgreich hochgeladen werden, so wird anschließend ein *Dialog* über die Klasse *FinishedPollDialog* dargestellt. Dieser bietet dem Benutzer weitere Umfragen an. Die Darstellung der Umfragen wird hier mittels einfacher *Buttons* in einer vertikalen Liste erreicht.

Abbildung 4.14: Umfrage mit Hilfe der Anwendung Crowdi (TV)

Damit die Anwendung die Probanden auf neue verfügbare Umfragen aufmerksam machen kann, implementiert diese das von *Android TV* angebotene *Empfehlungssystem*. Um eine Benachrichtigung über einen autonom gestarteten Hintergrunddienst zu ermöglichen, besitzt die Anwendung einen *BroadcastReceiver*, der über das *Android-Manifest* auf das Signal `Intent.ACTION_BOOT_COMPLETED` registriert ist. Der als *BootCompletedReceiver* implementierte *BroadcastReceiver* beantragt nach dem Erhalt des Signals die regelmäßige Benachrichtigung des Hintergrunddienstes über den *AlarmManagers* des Betriebssystems. Dazu gibt dieser mittels *Intent* einen Verweis auf den Dienst an. Läuft die Zeitspanne des gewählten Alarms ab, wird der referenzierte Empfänger über den Aufruf der Funktion `Service.onHandleIntent(Intent intent)` informiert. Die Klasse *RecommendationService* kann daraufhin analog zur *MainActivity* die API der *CrowdiLibrary* verwenden, um eine Liste der verfügbaren Umfragen zu erhalten. Die erhaltenen Umfragen können anschließend über *Notifications* im *Empfehlungssystem* dargestellt werden. Abbildung 4.15 zeigt die Benachrichtigung.

Abbildung 4.15: Benachrichtigungen der Anwendung Crowdi (TV)

Um den Benutzer nicht nur durch den *Leanback-Launcher* über Empfehlungen auf die verfügbaren Umfragen aufmerksam zu machen, implementiert die Anwendung das in Abschnitt 3.4 vorgestellte Konzept. Dazu werden die Benachrichtigungen innerhalb des Hintergrunddienstes über die *CrowdiUpnpLibrary* an gefundene Smartwatches übermittelt. Ermöglicht wird dies über die Klasse *WearNotificationServiceSenderConnection*, welche eine entsprechende Methode zur Verfügung stellt.

Damit die informierte Gegenstelle auch die Möglichkeit hat, Umfragen über die erhaltene Benachrichtigung am Fernseher zu starten, ist ein weiterer *Android Service* in der Anwendung *Crowdi (TV)* vorgesehen. Dieser wird ebenfalls über die Klasse *BootCompletedReceiver* gestartet und wartet anschließend über die *StartPollServiceReceiverConnection* auf entsprechende Befehle. Empfängt der Hintergrunddienst über die *ServiceConnection* ein Signal, so wird die *SinglePollActivity* über einen *Intent* gestartet. Dabei gilt es ein wichtiges Detail zu beachten. Da die Anwendung über einen Hintergrunddienst gestartet werden soll, muss der *Intent* zusätzlich mit dem *Flag* `Intent.FLAG_ACTIVITY_NEW_TASK` markiert werden.

Über den beschriebenen Mechanismus kann eine Benachrichtigung im klassischen Sinne auch auf *Android TV* umgesetzt werden. Sieht der Benutzer gerade einen Film über *Android TV* und empfängt währenddessen eine Benachrichtigung auf seiner Smartwatch, so kann er die Umfrage auf dem Fernseher starten, ohne die derzeitige Anwendung aktiv beenden zu müssen. Wurde die Umfrage vom Probanden ausgefüllt, kann er die Anwendung *Crowdi (TV)* wiederum beenden und direkt mit dem Film fortfahren.

Abbildung 4.16: Verwaltung der Accounts über die Anwendung Crowdi (TV)

Damit der Proband seine Login-Daten für die Crowd-Sensing-Plattform (*CrowdSensr*) hinterlegen kann, wird eine weitere Benutzeroberfläche benötigt. Das als *ManageAccountsActivity* umgesetzte Layout ist in Abbildung 4.16 dargestellt. Gestartet werden kann die *Activity* über den entsprechenden Eintrag in der *MainActivity*. Auch diese Oberfläche wurde auf Basis des *BrowseFragments* realisiert und sieht drei Reihen von Elementen vor. Während die erste Reihe zur Darstellung des Accounts für den *CrowdSensr* dient, zeigen die zweite und dritte Reihe verbundene bzw. verfügbare Fitness-Portale an.

Realisiert wird die Oberfläche über den *ArrayObjectAdapter* und zwei verschiedene *Presenter*. So enthält das Projekt den *LibraryAccountPresenter* zur Darstellung des Accounts für die *CrowdSensr*-Plattform. Die im Projekt angebotenen Fitness-Portale werden hingegen über den *FitnessAccountPresenter* dargestellt. Die Zuordnung der verbundenen bzw. verfügbaren Portale erfolgt dabei über den angebotenen *Settings-Manager* der *CrowdiFitnessLibrary*. Wird ein derzeit nicht verbundenes Fitness-Portal über die Fernbedienung selektiert, so wird der Login-Vorgang über das angebotene *SDK* gestartet. Kann der Login erfolgreich durchgeführt werden, so speichert der *SettingsManager* das verbundene Portal über die *SharedPreferences*[7] im System ab und informiert die Benutzeroberfläche über eine entsprechende *Callback-Methode*. Erhält die *ManageAccountActivity* dieses Signal, stößt sie ein Update des *ArrayObjectAdapters* an. Daraufhin wird das nun verbundene Fitness-Portal in der Kategorie “Verbundene Portale“ dargestellt. Wählt dieser hingegen ein bereits verbundenes Portal aus, so wird mittels *AlertDialog*[8] darauf aufmerksam gemacht, dass der Proband die Trennung des Portals beauftragt hat. Bestätigt dieser die Aktion, wird eine *Trennung* über das *SDK* veranlasst, was ein Update der Benutzeroberfläche zur Folge hat. Um ein ähnliches Interaktionskonzept für den *CrowdSensr* zu ermöglichen, bietet die *CrowdiLibrary* einen eigenen *LoginDialog* an. Dieser informiert ebenfalls mittels *Callback-Methoden* über Updates zum Anmeldevorgang. Eine Übersicht der zur Umsetzung nötigen Klassen ist in Abbildung 4.17 zu finden.

[7] Eine Klasse des Android-Betriebssystems, die zur Speicherung von Einstellungen in einer Datenbank dient

[8] Eine Klasse des Android-Betriebssystem, welche bereits eine visuelle Darstellung besitzt

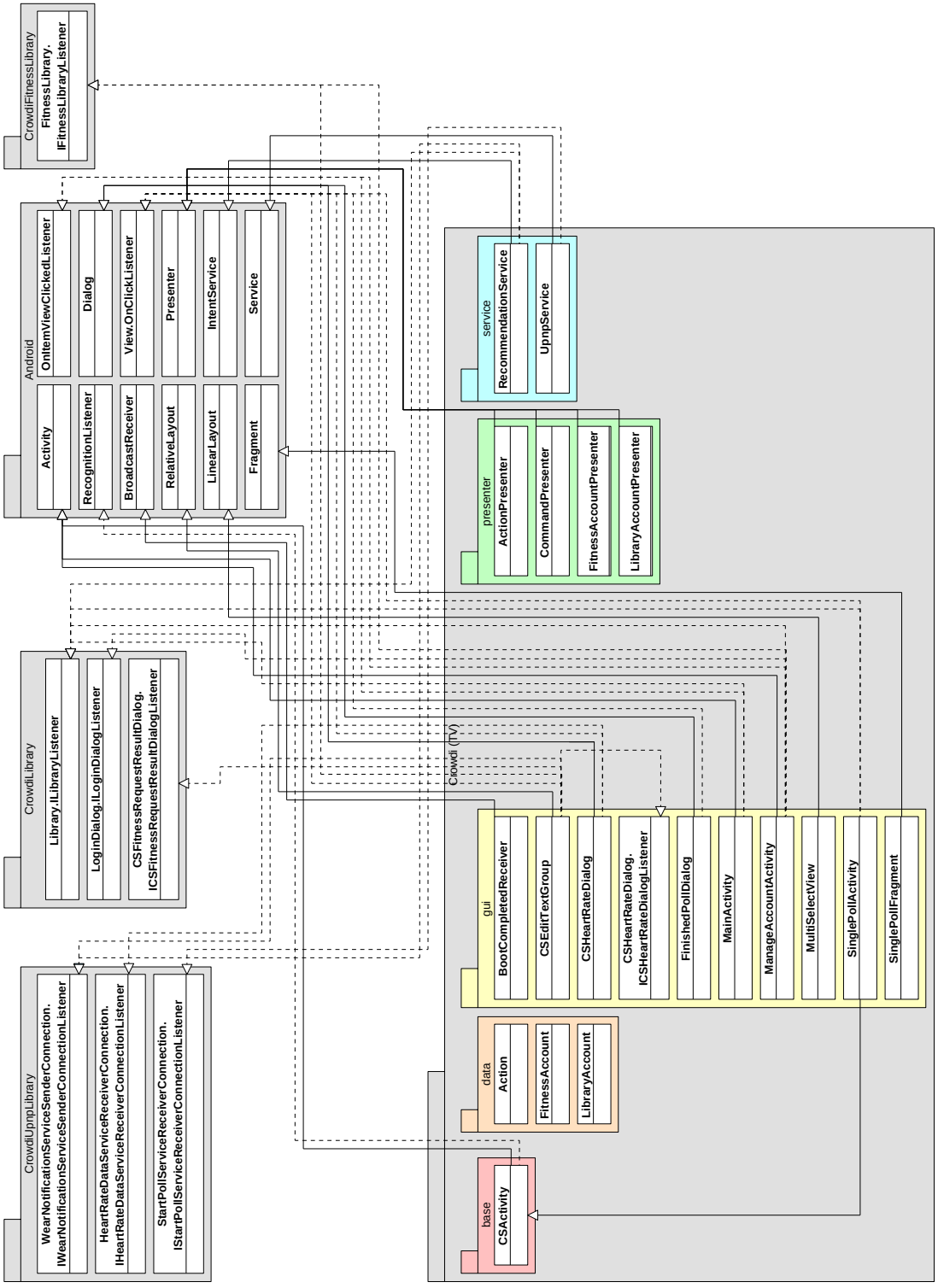

Abbildung 4.17: Klassendiagramm zum Projekt Crowdi (TV)

4.2.2 Crowdi (Mobile)

Die Anwendung *Crowdi (Mobile)* dient primär zur Kapselung der Anwendung *Crowdi (Wear)*. Dies ist nötig, um den im Abschnitt 2.7.1 beschriebenen Installationsmechanismus zu ermöglichen. Zusätzlich zu dieser Funktion dient die Anwendung als Realisierung der in 4.9 vorgestellten *UPnP* Funktionalität. Um über einen einzigen *Android Service* die Datenübertragung zur Smartwatch und die *UPnP* Funktionalität zu ermöglichen, implementiert die Anwendung den *AppConnectionService* aus der *CrowdiWearLibrary* und hält in diesem zugleich die nötigen Verbindungen über die *ServiceConnections* aus der *UPnP* Bibliothek. Im Detail besitzt der Dienst drei Verbindungen. Die *WearNotificationServiceReceiverConnection*, eine *StartPollServiceSenderConnection* sowie eine *HeartRateDataServiceSenderConnection*. Die über die *WearNotificationServiceReceiverConnection* empfangenen Benachrichtigungen werden mittels *Wearable.DataApi* an die Smartwatch weitergeleitet, die dort entsprechend darzustellen sind. Reaktionen auf die übermittelten Benachrichtigungen werden ebenfalls über die *Wearable.DataApi* empfangen und mittels *SenderConnection* an den Fernseher übertragen. Während ausgewählte Umfragen über die *StartPollServiceSenderConnection* laufen, wird die gemessene Herzfrequenz mittels *HeartRateDataServiceSenderConnection* verarbeitet.

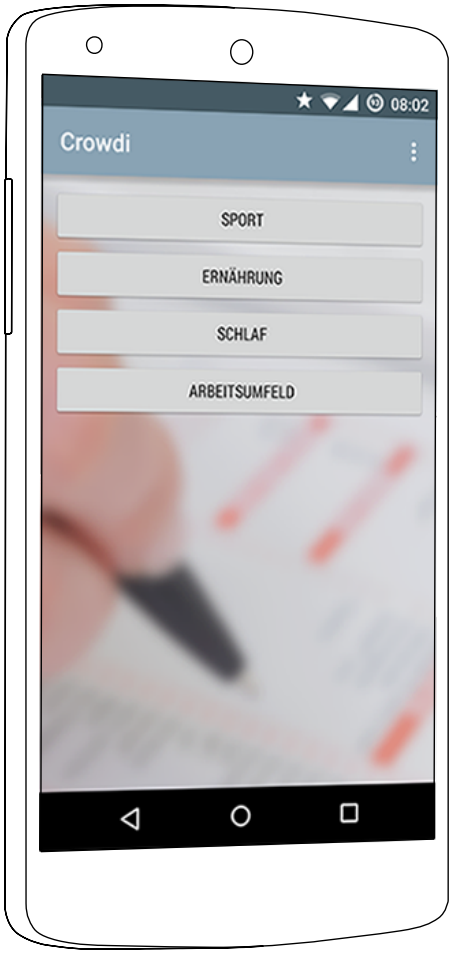

Abbildung 4.18: Startscreen der Anwendung Crowdi (Mobile)

Abbildung 4.19: Umfrage auf der Anwendung Crowdi (Mobile)

Neben dieser Hintergrundlogik besitzt die Anwendung auch eine klassische Benutzeroberfläche, die in Abbildung 4.18 zu sehen ist. Das dargestellte User-Interface dient primär zur Durchführung der in der Arbeit vorgesehenen Studie. Über die parallele Umsetzung kann ein Vergleich zwischen der Oberfläche der Smartphone-Anwendung und Fernseher-Anwendung ermöglicht werden. Wie in der Darstellung zu sehen ist, besitzt die Benutzeroberfläche eine sehr einfache Struktur. Diese wird über ein *LinearLayout* umgesetzt, welches die verfügbaren Umfragen mittels *Buttons* vertikal anordnet. Um die Kompatibilität des User-Interfaces für sehr viele Umfragen zu gewährleisten, wird das beschriebene *LinearLayout* zudem von einer *ScrollView*[9] umschlossen. Klickt der Benutzer auf einen der *Buttons*, so wird dieser auf eine zweite *Activity* geführt. Die Klasse *SinglePollActivity* entnimmt die über den *Intent* übermittelte URL und ruft die Umfrage über die API der *CrowdiLibrary* ab.

Das erfolgreiche Parsen der Umfrage bekommt die *Activity* über die im Abschnitt 4.1 vorgestellte *Callback-Methode* mitgeteilt und kann anschließend die benötigten Oberflächenelemente erstellen und darstellen. Eine beispielhafte Darstellung einer Umfrage auf dem Smartphone ist in Abbildung 4.19 zu sehen. Wie dem Bild zu entnehmen ist, werden analog zur Startseite sämtliche Umfrageelemente in einer vertikalen Liste dargestellt. Die dabei zugrunde liegende Struktur ist ebenfalls über ein *LinearLayout* mit vertikaler Anordnung umgesetzt. Die in der Abbildung dargestellten Umfrageelemente sind als

[9] Eine Klasse des Android-Betriebssystems welche die Einblendung verdeckter Elemente über eine Geste ermöglicht

eigenes *RelativeLayout* umgesetzt, welches mittels *LayoutInflater* eingeblendet wird. Dieses *RelativeLayout* besteht aus dem Titel, der mittels *TextView* am oberen Rand des Elements dargestellt wird. Der beinhaltende Inhalt wird über ein weiteres *RelativeLayout* abstrahiert.

Die Inhalte selbst werden über die reguläre Oberflächenelemente und den Oberflächenelementen aus der *CrowdiLibrary* auf die Typen der *CrowdSensr*-Plattform abgebildet. Dabei findet die selbe Abbildung statt, welche bereits im Abschnitt 4.2.1 beschrieben wurde. Die Bestätigung der erfolgreichen Eingabe der Umfragedaten kann über den *Button* in der *ActionBar*[10] erfolgen. Quittiert der Benutzer die Eingabe, so werden die Daten über die API der *CrowdiLibrary* an den Server übermittelt. Ist der Upload erfolgreich, so empfängt der Client über die *Callback-Methode* der Bibliothek eine Empfehlungsliste für weitere Umfragen. Diese Liste wird mittels *Dialog* über die Klasse *FinishedPollDialog* dargestellt. Dabei adaptiert der *Dialog* dieselbe Anwendungs- und Darstellungslogik wie die bereits beschriebene *MainActivity*. Wählt der Benutzer daraufhin eine weitere Umfrage aus, so wird eine neue Instanz der *SinglePollActivity* mit der entsprechenden URL gestartet. Möchte der Anwender an keiner weiteren Umfrage teilnehmen, so kann er den dargestellten *Dialog* über die “Zurück“-Taste schließen. Die aktuelle *SinglePollActivity* wird bei der Betätigung ebenfalls durch den *FinishedPollDialog* beendet, was zur Folge hat, dass sich der Benutzer wiederum auf der Startseite befindet. Abbildung 4.20 veranschaulicht das beschriebene Verhalten anhand eines Zustandsdiagrammes.

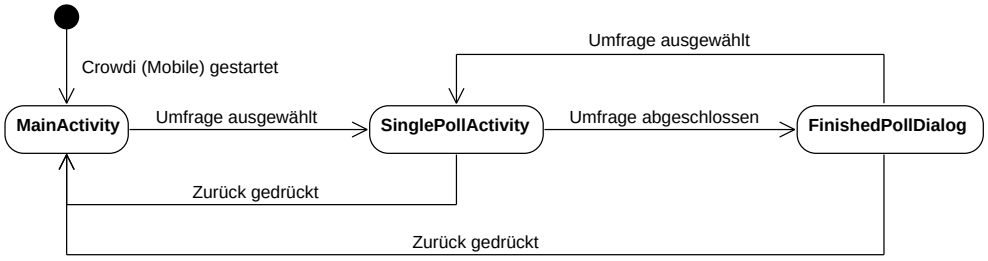

Abbildung 4.20: Zustandsdiagramm der Anwendung Crowdi (Mobile)

[10] Eine Klasse des Android-Betriebssystems, welche zur Darstellung einer Leiste im oberen Bereich der Anwendung dient. In dieser können Menüs umgesetzt werden

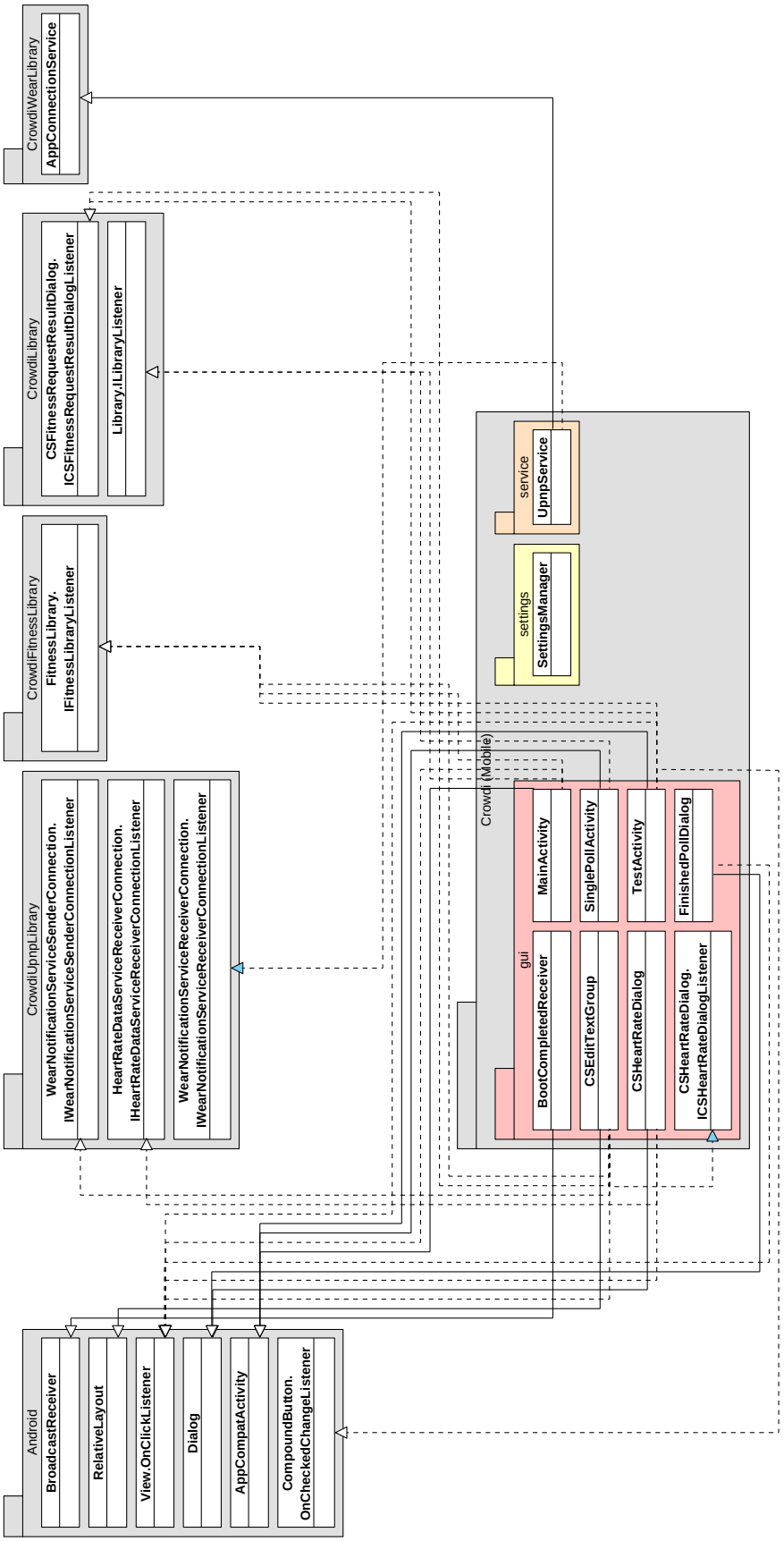

Abbildung 4.21: Klassendiagramm zum Projekt Crowdi (Mobile)

4.2.3 Crowdi (Wear)

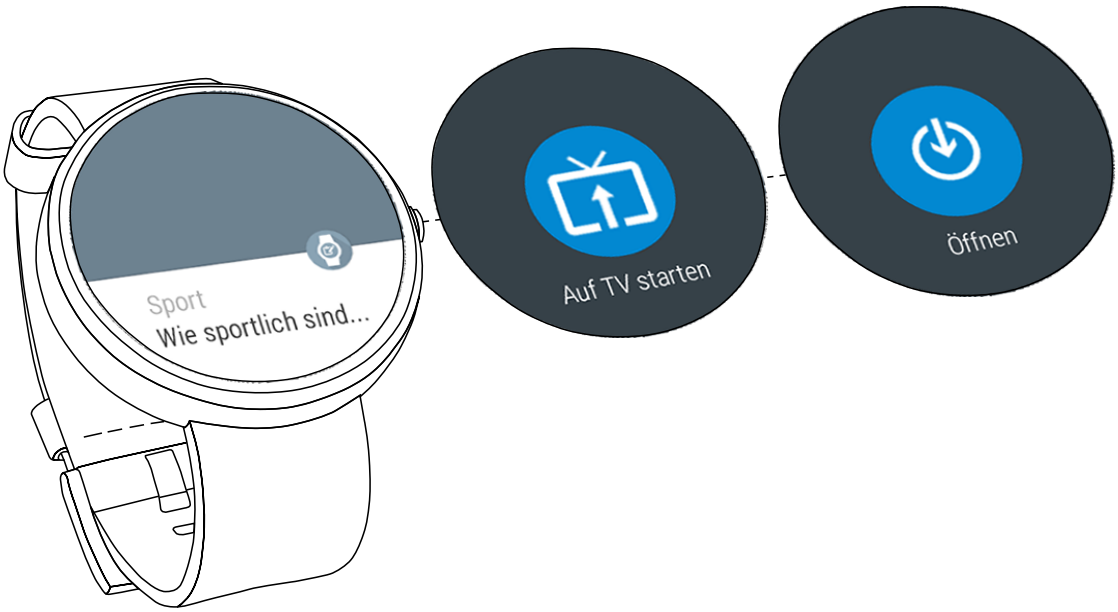

Abbildung 4.22: Benachrichtigung der Anwendung Crowdi (Wear)

Bei der *Crowdi (Wear)* handelt es sich um eine Anwendung, die im Wesentlichen zwei *Use-Cases* besitzt. Zunächst stellt die Anwendung einen Kommunikationskanal für die eigentliche *Android*-Anwendung *Crowdi (TV)* dar. Durch die aktive Verbindung zur Anwendung *Crowdi (Mobile)* auf dem Smartphone können die über die *CrowdiUpnpLibrary* empfangenen Benachrichtigungen auf der Uhr dargestellt werden. Abbildung 4.22 zeigt eine beispielhafte Benachrichtigung auf einer Smartwatch mit dem Betriebssystem *Android-Wear*. Je nach Art der Benachrichtigung wird auf der Smartwatch nur die standardmäßige *Action* “Öffnen“ oder eine weitere *Action* dargestellt.

Möchte die Anwendung *Crowdi (TV)* den Benutzer auf eine Umfrage aufmerksam machen, so findet man neben dem Titel und Inhalt der *Notification* auch die *Action* “Auf TV starten“. Über diese wird umgekehrt zur beschriebenen Benachrichtigung eine Nachricht über die *Wearable.DataApi* und anschließend über die *CrowdiUpnpLibrary* an den Fernseher übertragen. Abbildung 4.23 zeigt den beschriebenen Nachrichtenverlauf.

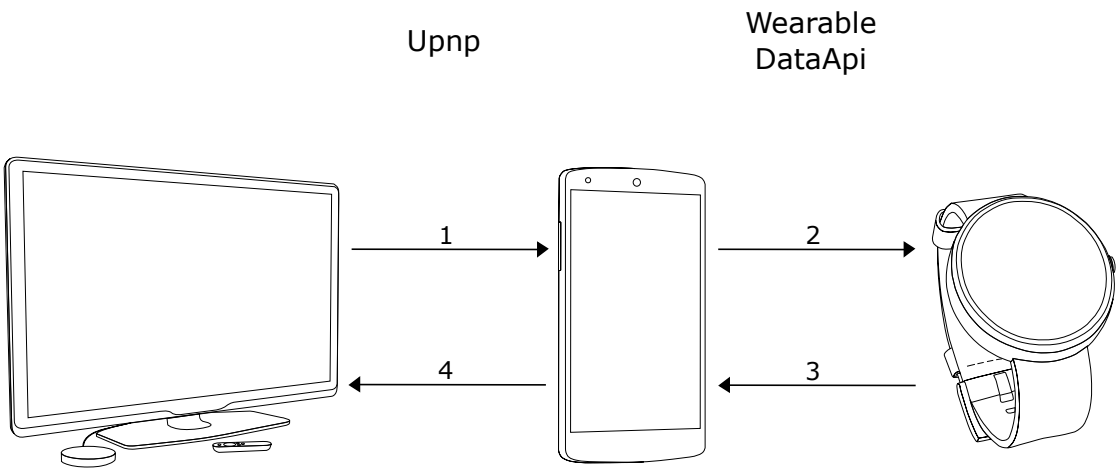

Abbildung 4.23: Nachrichtenverlauf

Beabsichtigt die Anwendung *Crowdi (TV)* das Starten der Herzfrequenzmessung, so wird eine Benachrichtigung mit vorgefertigtem Titel und Inhalt erzeugt. Diese Benachrichtigung enthält keine weitere *Action* und dient primär zum erleichterten Starten der Anwendung. Ein einfacher Klick startet die Anwendung *Crowdi (Wear)* auf der Smartwatch. Die Anwendung selbst besitzt lediglich eine *Activity*. Wird die Anwendung gestartet, so wird abhängig von der Geräteform ein rundes oder rechteckiges Layout eingeblendet. Beide Layouts besitzt eine *TextView*, welche zentral angeordnet ist. Wird das entsprechende Layout erzeugt, so initialisiert die Anwendung den benötigten *SensorManager* über den Aufruf der Funktion `Activity.getSystemService(SENSOR_SERVICE)`. Anschließend gelangt die Anwendung über den *SensorManager* an den Sensor für die Herzfrequenz. Registriert sich die Anwendung für Updates eines Sensors, so wird die Methode `SensorEventListener.onSensorChanged(SensorEvent event)` des zu implementierenden *SensorEventListener* durch das Betriebssystem aufgerufen. In dieser Funktion entnimmt die Anwendung den derzeitigen Wert der Herzfrequenz und aktualisiert den Inhalt der *TextView*. Um gleichbleibende Werte der Herzfrequenz besser unterscheiden zu können, wird zudem die Hintergrundfarbe der Anwendung geändert.

Um das Beenden der Anwendung durch den Bildschirm-Timeout der Smartwatch zu verhindern, implementiert diese den in *Android 5.1* vorgestellten *Ambient-Mode*[11]. Dazu wird zunächst die derzeitige Hintergrundfarbe deaktiviert. Die beim Update des Sensors erhaltenen Daten werden zunächst zwischengespeichert und erst nach dem Aufruf der Funktion `onUpdateAmbient()` an die *TextView* weitergegeben. Abbildung 4.24 zeigt die Gegenüberstellung der beiden Modi.

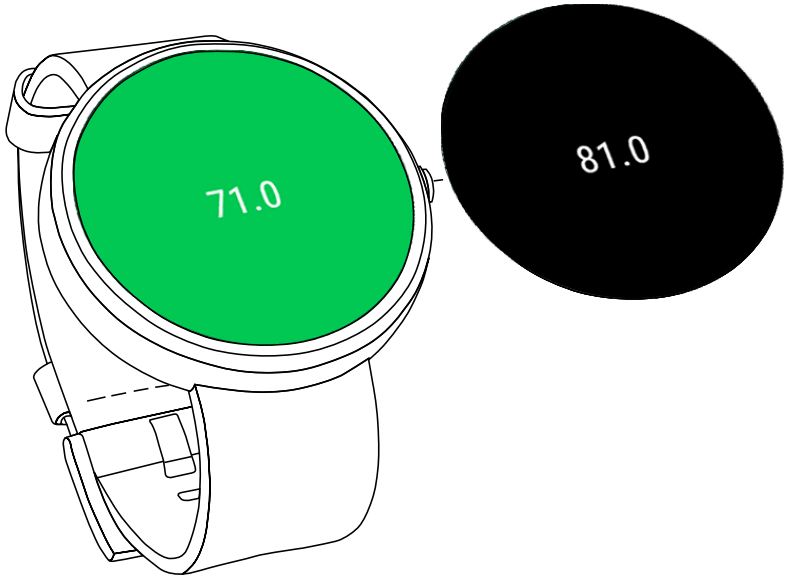

Abbildung 4.24: Die Anwendung Crowdi (Wear)

[11] Ein Energiesparmodus des Betriebssystems Android Wear

5

Studie

Da bisher keinerlei Informationen über Umfragesysteme auf dem Fernseher zur Verfügung stehen, wurde eine Studie mit Hilfe des entwickelten Umfragesystems durchgeführt. Die dabei inbegriffenen Hypothesen sind in Tabelle 5.1 aufgelistet. Daran anknüpfend ist eine Einführung in die Studie zu finden.

Tabelle 5.1: Hypothesen der Studie

Nr	Hypothese
1	Durch optimierte Benutzeroberflächen kann ein *Crowd-Sensing-System* für den Fernseher entwickelt werden, das mit einem System auf dem Smartphone vergleichbar ist.
2	Durch die Integration der Sprachbedienung können offene Fragen komfortabel am Fernseher ausgefüllt werden.
3	Die virtuelle Tastatur auf dem Smart-TV stellt keine akzeptable Eingabeform zur Beantwortung offener Fragen dar.
4	Über das Benachrichtigungssystem können mehr Probanden zur Umfrage animiert werden.
5	Ein Smart-TV bietet analog zum Smartphone viele Möglichkeiten zur Einbindung vielfältiger Sensoren, die auch von Probanden genutzt werden.
6	Die Anbindung von Fitness-Portalen ist zur gezielten Hilfestellung im Sinne des Probanden und würde durch ihn genutzt werden.

5.1 Aufbau

Der im Nachfolgenden beschriebene Aufbau untergliedert sich in die Abschnitte *Struktur*, *Fragebogen* und *Vergleichsfragebogen*. Während im ersten Abschnitt eine Zusammenfassung der wesentlichen Informationen wie die Teilnehmerzahl zu finden ist, sind die Inhalte der Fragebögen im Anschluss zu finden.

5.1.1 Struktur

Nach der Begrüßung der Probanden wird diesen im einleitenden Gespräch das Ziel der Studie näher gebracht. Dabei wird ihnen erklärt, dass mit Hilfe der Studie geklärt werden soll, ob *Crowd-Sensing-Systeme* auf dem Fernseher sinnvoll erscheinen und ob sich die Bedienbarkeit zu anderen Geräten gravierend unterscheidet. Anschließend wird dem Probanden im Gespräch der Testfragebogen und die Bedienung von *Android TV* erläutert. Um bessere Vergleichsdaten erzielen zu können, orientiert sich die Studie am *Within subject design* [Mac09]. Dabei füllt jeder Proband den Fragebogen auf dem Fernseher, dem Smartphone und dem Papier aus. Um Lerneffekte auszuschließen, wird das *Counterbalancing*-Verfahren angewendet. Dazu werden die 18 Probanden über ein Los-System in sechs Gruppen, je drei Personen, untergliedert. Das gezogene Los gibt die Reihenfolge der Umfragetechniken vor. Ein weiteres Los gibt Aufschluss darüber, ob der Proband am Fernseher mit der Sprachbedienung oder der virtuellen Tastatur beginnen muss. Um eine Aussage über die benötigte Zeit treffen zu können, wird diese mit Hilfe einer manuellen Stoppuhr gemessen und protokolliert. Nach dem Durchlauf aller Umfragetechniken wird ein Vergleichsfragebogen in Papierform ausgefüllt. Ist dieser abgeschlossen, werden dem Benutzer die speziellen Funktionen der Anwendung *Crowdi (TV)* vorgeführt. Dabei wird diesen zunächst gezeigt, wie eine Umfrage während des Betrachtens eines Films gestartet werden kann. Anschließend wird die Messung und Übertragung der Herzfrequenz mittels Smartwatch vorgeführt. Zuletzt wird gezeigt wie die Daten aus dem Fitness-Portal verwendet werden können. Im Anschluss daran wird die Meinung des Probanden zu den gezeigten Funktionen in einem Gespräch

eingeholt und protokolliert. Abgeschlossen wird die Studie über den Austausch der Nutzererfahrungen.

5.1.2 Fragebogen

Der Testfragebogen besteht aus insgesamt vier Fragekategorien. So werden im ersten Abschnitt des Fragebogens *Eigenschaftsfragen* über die Person gestellt. Mit insgesamt drei Fragen wird eine Gruppierung der Probanden beabsichtigt. Dabei wird nach dem Alter (*geschlossene Frage*), dem Geschlecht (*geschlossene Frage*) und der Eigenschaft "technikbegeistert" (*geschlossene Frage*) gefragt. Der zweite Abschnitt des Fragebogens enthält *Einstellungsfragen*. Mit der Frage nach der Relevanz von Umfragen und deren Belohnung soll die Einstellung der Probanden gegenüber Umfragen abgebildet werden. Im daran anschließenden Abschnitt des Fragebogens soll die derzeitige Akzeptanz von *Crowd-Sensing-Systemen* und Fitness-Portalen mittels *Verhaltensfragen* geprüft werden. Über die Angabe bezüglich der regelmäßigen Teilnahme an Umfragen kann eine Validierung der zuvor getätigten Aussage getroffen werden. Die im Abschnitt vier untergebrachten, *offenen Fragen* dienen zur Auswertung der textuellen Eingabe auf dem *Crowd-Sensing-System*. Um einen zeitlichen Vergleich der Frage zu ermöglichen, wird der auszufüllende Text über die Frage vorgegeben. Der in der Studie eingesetzte Fragebogen ist in Anhang B zu finden.

5.1.3 Vergleichsfragebogen

Im Vergleichsfragebogen werden sechs gezielte Fragen über die drei untersuchten Umfrageformen gestellt. So werden hier zunächst drei *Einstellungsfragen* bezüglich der Eignung der Umfrageform für regelmäßige Umfragen überprüft. Daran anknüpfend werden über zwei *Einstellungsfragen* die Meinungen über die reguläre textuelle Eingabe sowie die Eingabe mittels Mikrofon eingeholt. Abgerundet wird dieser Fragebogen indem die Umfrageformen anhand der eigenen Beliebtheit sortiert werden. Der zur Untersuchung verwendete Fragebogen ist in Anhang B zu finden.

5.2 Durchführung

Um bei der Durchführung ein möglichst natürliches Umfrageszenario zu bieten, wird die Umfrage am Fernseher sitzend auf dem Sofa durchgeführt. Der zum Einsatz kommende Fernseher und *Nexus Player* stellen unabhängige Variablen dar, die vom Probanden nicht beeinflusst werden können. Da bei schriftlichen Umfragen ein fester Untergrund benötigt wird, findet diese Umfrageform sitzend am Tisch statt. Der auf DIN A4 ausgedruckte Fragebogen kann mit einem der angebotenen Stifte ausgefüllt werden. Bei der Befragung über das Smartphone findet diese, je nach gezogener Reihenfolge, sitzend am Tisch oder auf dem Sofa statt. Dabei stellt das Smartphone ein *Nexus 5* dar, welches ebenfalls als unabhängige Variable aufzugreifen ist.

5.3 Zusammensetzung der Stichprobe

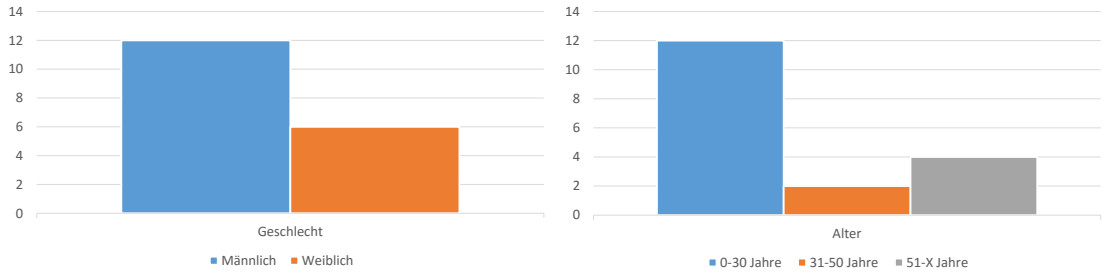

Abbildung 5.1: Zusammensetzung der Stichprobe, Geschlecht (links), Alter (rechts)

Beteiligt an der Studie waren insgesamt zwölf männliche und sechs weibliche Probanden. Ebenso hat die Erfassung alle drei Altersgruppen abgedeckt, wie in Abbildung 5.1 zu sehen ist. Während die Gruppe der Teilnehmer zwischen 0 und 30 Jahren mit insgesamt zwölf Probanden am stärksten vertreten war, nahmen ebenfalls zwei Freiwillige zwischen 31 und 50 Jahren und vier Kandidaten über 51 Jahren teil. Erstaunlicherweise konnte in der Studie kein Proband gefunden werden, der nicht im Besitz eines Smartphones ist. Einen Smart-TV besitzen zehn Teilnehmer wobei diese Zahl in 70% männlich und 30% weiblich zu differenzieren ist. Als “technikbegeistert“ haben sich insgesamt 11

Probanden bezeichnet. Lediglich ein Mann hat analog zu den sechs Frauen angegeben, nicht “technikbegeistert“ zu sein, wie in Abbildung 5.2 zu sehen ist.

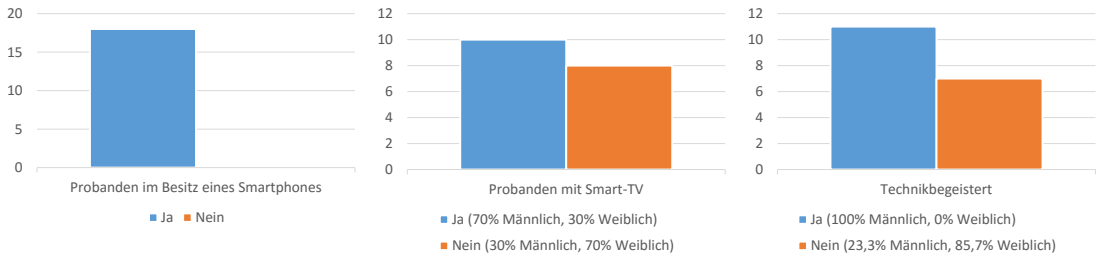

Abbildung 5.2: Zusammensetzung der Stichprobe, Probanden im Besitz eines Smartphones (links), Probanden mit Smart-TV (mitte), technikbegeistert (rechts)

Bei der Überzeugungsfrage bezüglich der Wichtigkeit von Umfragen zur Erforschung von Krankheiten haben 14 von 18 Probanden angegeben, die Relevanz dieser zu erkennen. Lediglich zwei Teilnehmer haben die Aussage verneint und zwei weitere Kandidaten haben sich enthalten. Anzumerken ist, dass sich lediglich männliche Personen gegen die Relevanz ausgesprochen haben. Die Enthaltungen sind hingegen lediglich von weiblichen Probanden in Anspruch genommen worden. Bezüglich der Relevanz einer Belohnung haben sich insgesamt zwölf Kandidaten für den Anreiz und sechs dagegen geäußert. Dabei haben 66,6% männliche und 33,3% weibliche Teilnehmer die Aussage verneint. Dies ist auch Abbildung 5.3 zu entnehmen. Somit ist festzuhalten, dass die Mehrzahl der Beteiligten zwar die Relevanz erkennt, dennoch einen zusätzlichen Anreiz für die Beteiligung als nötig ansehen.

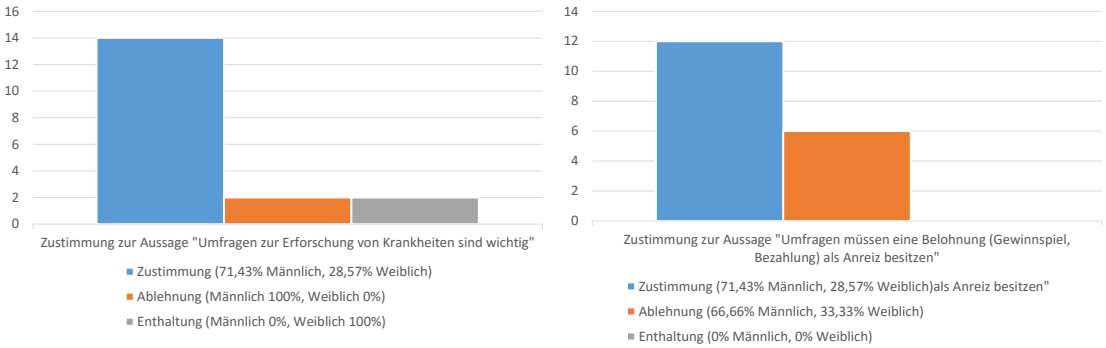

Abbildung 5.3: Zustimmung zur Wichtigkeit von Umfragen (links), Relevanz einer Belohnung (rechts)

Bei der Verhaltensfrage bezüglich der derzeitigen Beteiligung an regelmäßigen Umfragen haben lediglich drei der 18 Probanden angegeben, mindestens einmal im Monat an einer Datenerhebung beteiligt zu sein. Dabei haben diese Aussage zwei männliche und ein weiblicher Teilnehmer getroffen, wie in Abbildung 5.4 zu sehen ist. Bei der Nutzung des Smartphones zur Erfassung der sportlichen Aktivitäten stellt sich ein ähnliches Bild dar. Lediglich vier Kandidaten haben die Aussage bejaht, ein solches Crowd-Sensing-System aktiv zu nutzen. Die Aussage ist in gleichem Maße von männlichen wie weiblichen Probanden getroffen worden.

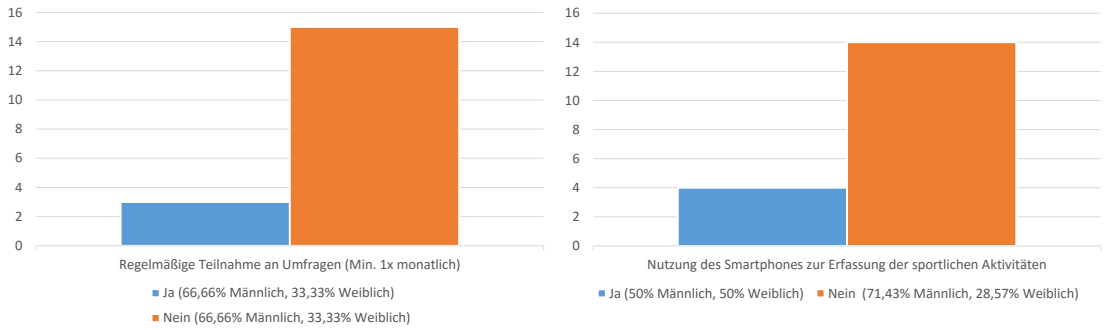

Abbildung 5.4: Teilnahme an Umfragen (links), Nutzung des Smartphones als Fitness-Tracker (rechts)

5.4 Umfrageergebnisse

Neben der Vergleichbarkeit der Systeme soll ebenfalls die Akzeptanz der neuartigen Konzepte überprüft werden. Während im nächsten Abschnitt die ermittelten Vergleichsdaten vorgestellt werden, findet sich anschließend die Nutzermeinung bezüglich den Erweiterungen.

5.4.1 Vergleich der Systeme

Nach der Durchführung der Studie sind die Mittelwerte der benötigten Durchführungszeiten ausgewertet worden. Dazu ist mit einem Smartphone die Ausfülldauer zwischen der ersten und neunten geschlossenen Frage ermittelt worden. Die benötigte Zeit jedes

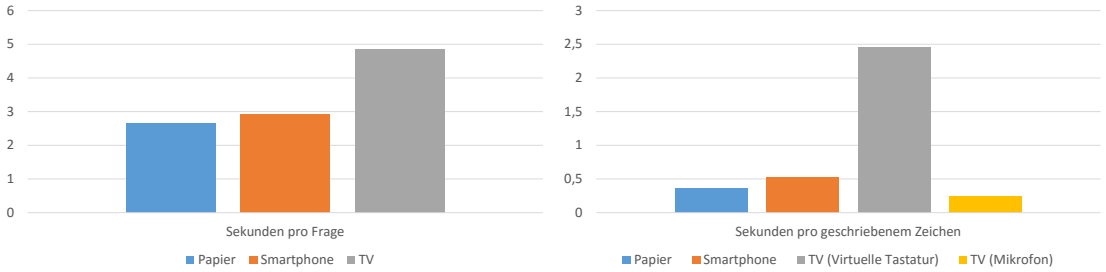

Abbildung 5.5: Gemessene Eingabezeiten, Sekunden pro Frage (links), Sekunden pro Zeichen (rechts)

Probanden ist durch die Anzahl der geschlossenen Frage dividiert worden. Wie der Abbildung 5.5 zu entnehmen ist, haben die Probanden zur Beantwortung der Fragen am wenigsten Zeit bei der Papier-Umfrage benötigt. So haben diese eine geschlossene Frage im Schnitt innerhalb von 2,65 Sekunden ausgefüllt. Auch das Smartphone hat mit den ermittelten 2,93 Sekunden einen vergleichbaren Wert erreicht. Lediglich die Umfrage auf dem Fernseher hat wesentlich mehr Zeit benötigt. So haben die Probanden hier eine geschlossene Frage innerhalb von 4,84 Sekunden ausgefüllt.

Bei der Ermittlung der benötigten Zeitdauer pro geschriebenem Zeichen zeigt sich ein noch markanterer Unterschied zwischen den Systemen. Zur Ermittlung der Zeitwerte sind drei offene Fragen in die Umfrage integriert worden. Die Antwort auf die Frage ist fest vorgegeben und durch den Probanden abgeschrieben bzw. nachgesprochen worden. Die ermittelten Zeitwerte jeder Frage sind zunächst durch die Anzahl der Zeichen im Satz dividiert worden. Anschließend ist der Mittelwert der drei Werte für jeden Teilnehmer gebildet worden. Bei der Bildung der Mittelwerte über sämtliche Teilnehmer hat sich erneut gezeigt, dass die Beantwortung von offenen Fragen sehr schnell auf der klassischen Papier-Umfrage erfolgt ist. So haben die Probanden in der Studie 0,37 Sekunden pro Zeichen benötigt. Es fällt wiederum auf, dass auch das Smartphone, trotz deaktivierter Rechtschreibkorrektur, eine ähnlich gute Eingabemöglichkeit darstellt. Hier brauchen die Teilnehmer durchschnittlich 0,52 Sekunden pro Zeichen. Anzumerken ist, dass die geringste Zeitdauer von 0,21 Sekunden am Smartphone sogar unterhalb des Spitzenwerts von 0,29 Sekunden bei der Papierumfrage liegt. Hier zeigt sich, dass die

ständige Verwendung des Smartphones zu einer Konditionierung der Eingabe auf dem Smartphone führt. Bei der benötigten Zeitdauer am Fernseher ist deutlich zu erkennen, dass die reguläre Eingabe über die virtuelle Tastatur deutlich höher ist. Hier haben die Probanden im Schnitt 2,46 Sekunden pro Zeichen benötigt. Im Extremfall hat ein Teilnehmer für das Abschreiben des Satzes "*Ich nehme an einer Studie ueber Umfragesysteme im Rahmen einer Masterarbeit teil*" über vier Minuten beansprucht. Die Spracheingabe des Satzes stellt in diesem Vergleichstest bezüglich der Eingabegeschwindigkeit, die beste Eingabemöglichkeit dar. So haben die Teilnehmer nur 0,24 Sekunden pro Zeichen benötigt. In diesem Wert einbezogen sind ebenfalls die 17 zunächst falsch erfassten und durch eine erneute Eingabe verbesserten Spracherkennungen. Dabei hat sich gezeigt, dass die Spracheingabe insbesondere bei Probanden mit stark ausgeprägtem Dialekt zu falschen Interpretationen neigt.

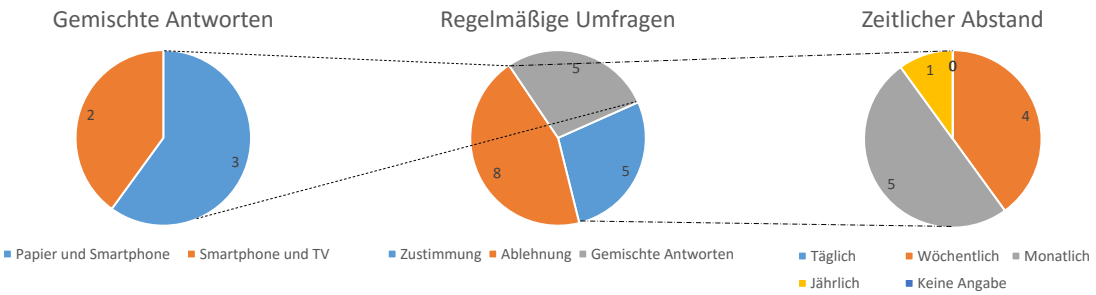

Abbildung 5.6: Zusammensetzung der gemischten Antworten (links), Akzeptanz des Umfragesystems (mitte), akzeptierte zeitliche Abstände (rechts)

Bei der Datenerhebung über den Vergleichsfragebogen hat sich ein gespaltenes Bild gezeigt. Während fünf Probanden angeben, sich auf sämtlichen Umfrageformen eine regelmäßige Untersuchung vorstellen zu können, lehnen acht Probanden jegliche Form ab. Die restlichen fünf Probanden lassen sich in zwei Gruppen untergliedern. So gibt es insgesamt zwei männliche Probanden, die lediglich zur digitalen Untersuchung bereit sind, wie in Abbildung 5.6 zu sehen ist. Eine Umfrage in Papierform oder auf dem Smartphone können sich hingegen zwei Männer und eine Frau vorstellen. Dabei haben die zehn Probanden sehr unterschiedliche Zeitabstände zwischen den Umfragen angegeben. Während sich keiner einer täglichen Befragung unterziehen mag, können

sich immerhin vier Probanden eine wöchentliche Untersuchung vorstellen. Die Mehrheit hat sich hingegen lediglich für eine monatliche Datenerhebung ausgesprochen. Ein Kandidat hat sogar angegeben, nur einmal im Jahr für eine Umfrage zur Verfügung zu stehen.

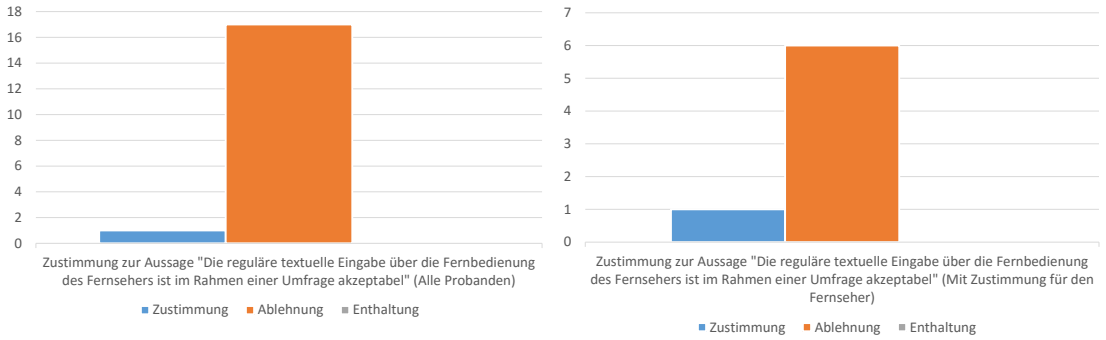

Abbildung 5.7: Akzeptanz der textuellen Eingabe über die virtuelle Tastatur

Da bereits vor der Studie die textuelle Eingabe als mögliche Schwachstelle des Fernsehers vermutet wurde, ist anhand der Vergleichsstudie ebenfalls die Akzeptanz der textuellen Eingabe auf dem Fernseher ermittelt worden. Dabei bestätigt sich die zuvor formulierte Hypothese, dass die Eingabemöglichkeit über die virtuelle Tastatur keine akzeptable Lösung darstellt. So haben insgesamt 17 Teilnehmer angegeben, diese Eingabemöglichkeit als nicht akzeptabel für die Beantwortung von offenen Fragen anzusehen. Lediglich ein Proband spricht sich für das reguläre Texteingabesystem aus. Unter den Probanden, die sich eine regelmäßige Umfrage auf dem Fernsteher tatsächlich vorstellen können, ergibt sich eine ähnliche Verteilung. So lehnen hier sechs der sieben Probanden die Eingabeform ab. Dies ist auch in Abbildung 5.7 zu erkennen.

Bei der Frage, ob die Sprachbedienung eine bessere Eingabemöglichkeit als die virtuelle Tastatur darstellt, zeigt sich ein eindeutiges Bild. Hier stimmen 17 Teilnehmer für die Sprachbedienung und lediglich ein Proband dagegen. Unter den sieben Kandidaten, die sich eine regelmäßige Umfrage auf dem Fernseher vorstellen können, ist die Zustimmung noch höher. So geben hier 100% der Befragten an, dass die Sprachbedienung die bessere Eingabeform darstellt. Diese Erkenntnis wird in Abbildung 5.8 sehr deutlich.

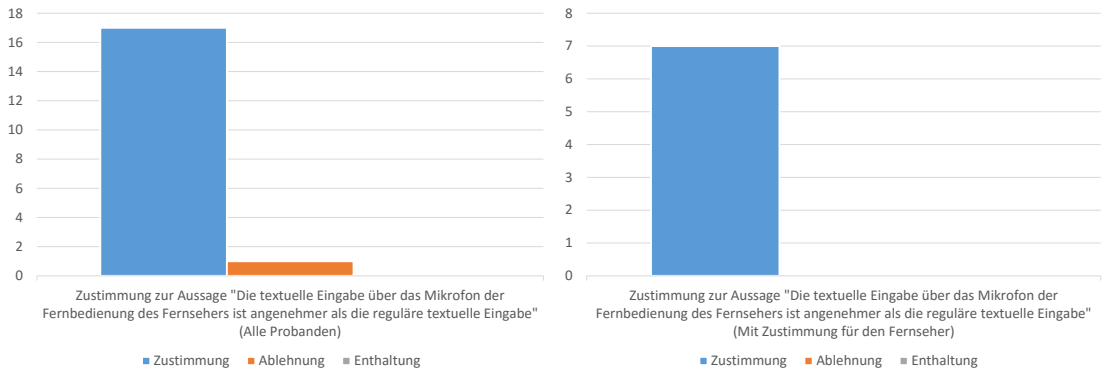

Abbildung 5.8: Bevorzugung der textuellen Eingabe über die Spracherkennung

Um eine eindeutigere Aussage über die Präferenzen bezüglich der Umfrageform treffen zu können, ist der Proband in der letzten Frage aufgefordert worden, das bevorzugte *Crowd-Sensing-System* zu benennen.

Bei der Auswertung der Ergebnisse, die in Abbildung 5.9 zu sehen sind, zeigt sich, dass die Umfrage auf dem Smartphone am beliebtesten ist. So geben insgesamt neun der 18 Probanden an, eine Umfrage auf dem Smartphone zu bevorzugen. Lediglich vier Teilnehmer sprechen sich für das Papierformat und zwei für den Fernseher aus. Während 75% der Teilnehmer für das Papierformat über 51 Jahre sind, spricht sich auch ein weiblicher Proband unter 31 Jahre für die klassische Form aus. Für den Fernseher hat sowohl ein weiblicher und ein männlicher Proband aus der Gruppe der jungen Teilnehmer gestimmt. Unter den sieben Probanden, die sich eine regelmäßige Umfrage auf dem Fernseher vorstellen können, zeigt sich ebenfalls, dass die Präferenz beim Smartphone liegt. Hier geben fünf Teilnehmer an, eine Umfrage auf dem Smartphone zu bevorzugen. Lediglich ein Proband spricht sich für den Fernseher, sowie ein weiterer für die Papierform, aus.

5.4.2 Akzeptanz der Zusatzfunktionen

Zur Abfrage der Akzeptanz ist dem Teilnehmer die entsprechende Funktion vorgeführt und der entsprechende Kontext erläutert worden. Beispielsweise ist dem Probanden

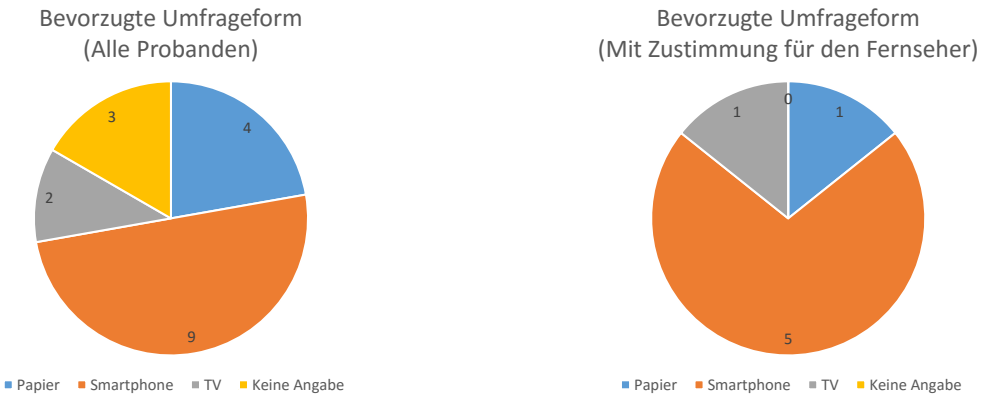

Abbildung 5.9: Bevorzugte Umfrageform

erklärt worden, dass er die Startfunktion dazu nutzen könne, das aktuelle Fernsehprogramm für die Teilnahme an der Umfrage zu unterbrechen. Bei der Datenerhebung haben 13 der 18 Probanden angegeben, dass die entwickelte Funktion einen Mehrwert darstellt und genutzt werden würde. Unter den sieben Teilnehmern, die sich einer regelmäßigen Befragung auf dem Fernseher unterziehen würden, ist die Akzeptanz der Funktion noch höher. So haben hier 100% der Befragten angegeben, das Feature als nützlich zu empfinden. Eine visuelle Darstellung der ermittelten Ergebnisse ist in 5.10 zu finden.

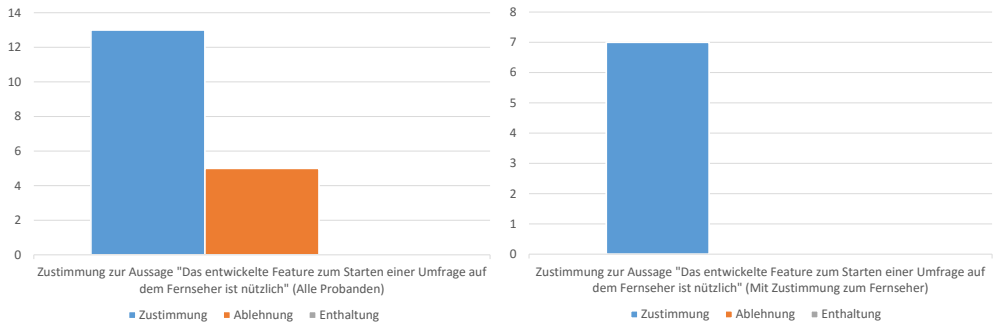

Abbildung 5.10: Akzeptanz des Benachrichtigungssystem

Die demonstrierte Funktion zur Einbindung des Herzfrequenzmessers der Smartwatch wird ebenfalls sehr nützlich eingestuft. Anzumerken sei, dass den Probanden hier erläu-

tert worden ist, dass die demonstrierte Funktion ein Beispiel darstellt und sie sich weitere Sensoren vorstellen können. Ebenfalls ist den Benutzern erklärt worden, dass diese die fehlerhafte Messung der Smartwatch nicht in die Beurteilung einbeziehen dürfen. Bei der Studie haben ebenfalls 13 von 18 Probanden angegeben, dass diese Funktionen einen deutlichen Mehrwert darstellt. In der Gruppe der Teilnehmer, die sich eine regelmäßige Untersuchung auf dem Fernseher vorstellen können, geben fünf von sieben Probanden an, dass die entwickelte Funktion nützlich ist. Lediglich ein Proband spricht sich gegen die Funktion aus, wie in Abbildung 5.11 zu sehen ist. Auch eine Enthaltung ist in dieser Gruppe vertreten.

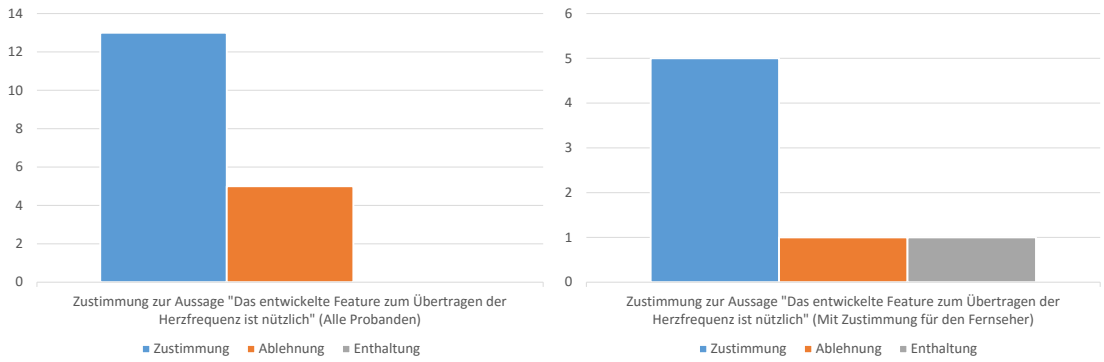

Abbildung 5.11: Akzeptanz der Anbindung des Herzfrequenzmessers

Auch die zuletzt überprüfte Funktion zur Anbindung von Fitness-Portalen erreicht eine vergleichbare Akzeptanz. Hier ist den Probanden gezeigt worden, wie sie mit Hilfe der verbundenen Fitness-Platform eine relativ abstrakte Frage wie "*Wie viele Schritte laufen Sie durchschnittlich am Tag?*" beantworten können. Auch hier ist den Teilnehmer erläutert worden, dass dies nur ein Beispiel darstellt und weitere Fragestellungen denkbar seien. Nach der Demonstration haben zwölf der Probanden angegeben, dass dies eine nützliche Funktion darstellt. Lediglich drei Probanden sprechen sich gegen die Anbindung von Fitness-Portalen aus und weitere drei enthalten sich der Aussage. Dies ist auch in der Abbildung 5.12 zu sehen. Analog zur Aussage beim Herzfrequenzmesser geben auch in dieser Frage fünf der sieben Probanden an, dass die Funktion einen

deutlichen Mehrwert darstellt. Auch hier hat ein Teilnehmer die Anbindung verneint. Die Schlussfolgerung zu den vorgestellten Umfrageergebnisse wird im nachfolgenden Abschnitt gezogen.

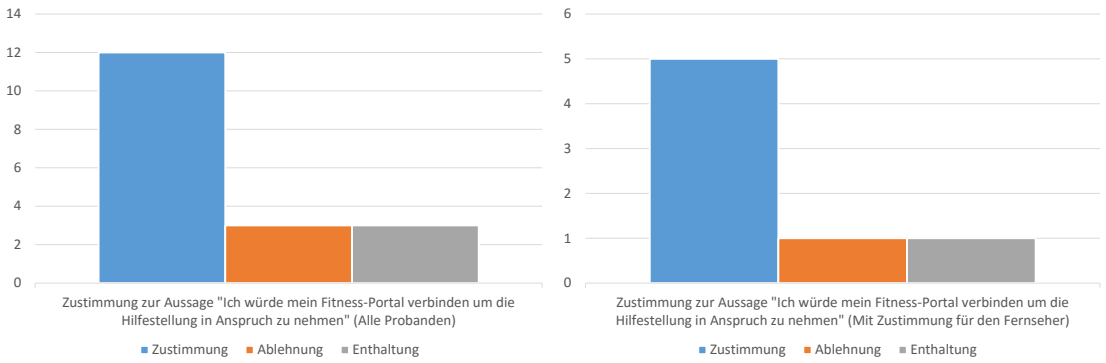

Abbildung 5.12: Akzeptanz der Anbindung des Fitness-Portals

5.5 Auswertung

Betrachtet man die in 5.4 vorgestellten Studienergebnisse, so kann die erste Hypothese nicht bestätigt werden. Mit dem hier realisierten *Crowd-Sensing-System* für den Fernseher kann trotz optimierter Benutzeroberfläche kein System entwickelt werden, dass mit einem System auf dem Smartphone standhalten kann. Während die ermittelten Eingabezeiten für geschlossene Fragen auf dem Smartphone niedriger sind, kann die Sprachbedienung am Fernseher bei offenen Fragen punkten. Die rein objektive Erfassung der Eingabezeiten würde somit für ein vergleichbares System sprechen. Jedoch zeigt die subjektive Präferenz der Probanden deutlich, dass das System auf dem Smartphone als benutzerfreundlicher empfunden wird und lässt somit folgern, dass zum aktuellen Zeitpunkt kein vergleichbares System entwickelt werden kann.

Es wird jedoch durch die eingebundene Sprachbedienung gezeigt, dass auch offene Fragen auf dem Fernseher möglich sind. Durch die erfassten Daten ist zu erkennen, dass die Sprachbedienung kürzere Eingabezeiten als die textuelle Eingabe auf dem

Smartphone besitzt. Damit wird auch bestätigt, dass der aktuelle Entwicklungsstand der Sprachbedienung bereits sehr stabil ist und nur zu sehr wenigen, fehlerhaften Datensätzen führt. Folglich lässt sich die zweite Hypothese, dass die Sprachbedienung eine komfortable Beantwortung von offenen Fragen ermöglicht, als belegt bezeichnen.

Auch die dritte Hypothese wird durch die Studie bestätigt. So zeigen die erfassten Datensätze, dass die textuelle Eingabe über die virtuelle Tastatur auf dem Fernseher wesentlich mehr Zeit in Anspruch nimmt. Obwohl in der Studie nur relativ kurze Sätze bis maximal 80 Zeichen verwendet werden, benötigen die Probanden bereits mehrere Minuten zur Eingabe. Auch die im Vergleichsfragebogen erfassten Daten bestätigen die Hypothese. Hiermit wird gezeigt, dass fast alle Probanden die textuelle Eingabemöglichkeit als nicht akzeptabel beurteilen.

Durch die Umfrageergebnisse bezüglich der Akzeptanz des Startmechanismus über die Smartwatch kann gezeigt werden, dass diese Funktion von der Mehrzahl der Probanden als nützlich eingestuft wird. Basierend auf der Aussage der Probanden wird vermutet, dass hierdurch ebenfalls die Rücklaufquote der Umfragen erhöht werden kann. Jedoch ist diese Vermutung nur über eine Langzeitstudie zu verifizieren. Hier ist über die parallele Verteilung von zwei Systemversionen mit und ohne Benachrichtigungssystem die Rücklaufquote zu erfassen und gegenüber zu stellen. Damit kann die Hypothese, dass sich das Benachrichtigungssystem positiv auf die Rücklaufquote auswirkt, nicht gänzlich bewiesen werden.

Die fünfte Hypothese ist wiederum vollständig bewiesen worden. Durch die Anbindung des Herzfrequenzsensors der Smartwatch in das *Crowd-Sensing-System* wird gezeigt, dass ein System auf dem Smart-TV analog zum Smartphone durch zusätzliche Sensoren über einfache Mittel erweitert werden kann. Durch die in der Studie formulierte Frage bezüglich der Nützlichkeit dieser Einbindung von Sensoren, wird ebenfalls belegt, dass auch Anwender die Einbindung von weiteren Sensoren zur Erfassung der Vital-Funktion als nützlich einstufen.

Auch die zuletzt formulierte Hypothese über die Akzeptanz von Fitness-Portalen zur Hilfestellung bei abstrakten Fragen kann durch die Studie bewiesen werden. Hier zeigt sich, dass durch die richtige Integration der Fitness-Portale eine entsprechende Akzeptanz erreicht werden kann.

Wird ein Fitness-Portal verbunden und überträgt ohne das Zutun der Probanden die Daten in das Internet, so wird dies häufig als negativ empfunden. Durch die gezielte Hilfestellung über die verbundenen Fitness-Portale und die manuelle Auswahl des entsprechenden Datensatzes besitzt der Proband die nötige Kontrolle über seine persönlichen Messwerte. Dadurch ist bewiesen, dass die zusätzliche Anbindung von Fitness-Portalen durch die Probanden als nützlich eingestuft wird und entsprechenden Einsatz findet.

6

Zusammenfassung und Ausblick

Resümierend ist eindeutig zu erkennen, dass sämtliche Ziele der Arbeit erreicht worden sind. Mit der durchgeführten Hard- und Softwareanalyse ist *Android TV* auf einem *Nexus Player* als erfolgversprechendste Plattform für die beabsichtigte Anwendung ausgewertet worden. Hier hat sich gezeigt, dass insbesondere das fehlende Angebot eines *SDK* gegen *Apple TV* und die nicht freigegebene Sprachbedienung gegen den *Fire TV* sprechen.

Wie die Arbeit zeigt, ist es über die Konzeption möglich, ein *Crowd-Sensing-System* für *Android TV* zu entwickeln. Durch das selbst entwickelte Benachrichtigungssystem über eine *Android Wear*-Smartwatch kann die größte Schwachstelle von *Android TV* ausgebessert werden. Hier zeigt sich, dass das erweiterte Konzept der Anforderungsanalyse den Ansprüchen gänzlich folge leisten kann. Ebenfalls kann durch die Anbindung der Smartwatch die Modularität des Betriebssystems belegt werden. Durch die beispielhafte Anbindung des Herzfrequenzsensors kann gezeigt werden, dass das Betriebssystem auf dem Smart-TV ebenfalls von zusätzlichen Sensoren als Eingabemittel profitiert. Auch die Anbindung von Fitness-Portalen in der *Crowdi*-Anwendung spricht für die Flexibilität des Systems. Somit ist eindeutig bewiesen, dass die Vorteile des mobilen Betriebssystems ebenfalls auf das OS des Smart-TV's übertragbar sind.

Durch die zusätzlich entwickelte Anwendung auf dem Smartphone ist eine solide Grundlage für die durchgeführte Studie geschaffen worden. Nur dadurch ist gewährleistet, dass weder die Darstellung noch das Interaktionskonzept der Umfrageelemente unkontrollierte Einflüsse auf die Ergebnisse besitzen. Dennoch sind optimierte Benutzeroberflächen

auf beiden Systemen realisiert worden, um keine künstliche Manipulation der Eingabezeiten zu erzeugen. Wie die Studie zeigt, konnte ein System entwickelt werden, das insbesondere durch die Integration der Sprachbedienung eine bessere Eingabezeit für offene Frage besitzt als ein Smartphone. Jedoch macht die Studie ebenfalls deutlich, dass die Toucheingabe auf dem Smartphone bei der Navigation durch die geschlossenen Fragen einen deutlichen Vorteil gegenüber dem Smart-TV's besitzt. Somit ist ein System geschaffen worden, das durch die rein objektive Analyse der Eingabezeiten mit einem *Crowd-Sensing-System* auf dem Smartphone vergleichbar ist. Jedoch wird auch gezeigt, dass die subjektive Empfindung der Probanden gegen den Einsatz des Smart-TV's als *Crowd-Sensing-System* spricht. Hier sprechen sich lediglich sieben der 18 Probanden für eine regelmäßige Befragung über den Fernseher aus und zeigen über die Kontrollfrage, dass sie das System auf dem Smartphone dennoch präferieren. Abschließend kann gesagt werden, dass sich ein Crowd-Sensing-System derzeit auf dem Fernseher wegen der fehlenden Akzeptanz nicht eignet. Dennoch ist die zukünftige Eignung des System nicht gänzlich auszuschließen, da sich die Akzeptanz durch die steigende Verbreitung des Smart-TV's ändern kann.

Des weiteren müsste die Akzeptanz über eine Langzeitstudie verifiziert werden. In diesem Kontext wäre die Entwicklung von zwei Systemversionen denkbar, um Erkenntnisse über die Rücklaufquote mit und ohne Benachrichtigungssystem zu gewinnen. Wie in einem Abschlussgespräch der Studie ebenfalls angemerkt wurde, wäre die Erkennung von Werbespots interessant, um die Rücklaufquote durch die gezielte Benachrichtigung während der Sendepausen zu steigern.

A

Quelltexte

```
// Object
{
        "id": 1,
        "name": "value"
}

// Array of object
[
        {
                "id": 1,
                "name": "value"
        },
        {
                "id": 2,
                "name": "value"
        }
]
```

Listing A.1: JSON-Objekt und JSON-Array

```
{
    "next": [
        {
            "command": "poll1",
            "method": "GET",
```

```
            "info": "Poll 1"
        },
        {
            "command": "poll2",
            "method": "GET",
            "info": "Poll 2"
        }
    ]
}
```

Listing A.2: Command

```
{
    "form":
    {
        "version": "0.0.2",
        "fields": [
            {
                "name": "deliveryAddress",
                "type": "compositeField",
                "compositeField": "address",
                "label": "Lieferadresse"
            },
            {
                "name": "invoiceAddress",
                "type": "compositeField",
                "compositeField": "address",
                "label": "Rechnungsadresse"
            },
            {
                "name": "banking",
                "type": "compositeField",
```

```
            "compositeField": "banking",
            "label": "Bezahldaten"
        }
    ],
    "compositeFields": [
        {
            "name": "address",
            "fields": [
                {
                    "name": "givenName",
                    "label": "Vorname",
                    "type": "text",
                    "maxlength": 100,
                    "required": true
                },
                {
                    "name": "surname",
                    "label": "Nachname",
                    "type": "text",
                    "maxlength": 100,
                    "required": true
                },
                {
                    "name": "street",
                    "label": "Strasse",
                    "type": "text",
                    "maxlength": 100,
                    "required": true
                },
                {
                    "name": "streetNumber",
```

```
52                          "label": "Hausnummer",
53                          "type": "text",
54                          "pattern": "[0-9]+(\\\\[0-9]*[a-z]?)?",
55                          "required": true
56                      },
57                      {
58                          "name": "tel",
59                          "label": "Tel-Nr",
60                          "type": "tel"
61                      }
62                  ]
63              },
64              {
65                  "name": "banking",
66                  "fields": [
67                      {
68                          "name": "ownerName",
69                          "label": "Kontoinhaber",
70                          "type": "text",
71                          "maxlength": 100,
72                          "required": true
73                      },
74                      {
75                          "name": "iban",
76                          "label": "IBAN",
77                          "type": "text",
78                          "maxlength": 100,
79                          "required": true,
80                          "pattern": "[a-zA-Z]{2}[0-9]{2}..."
81                      },
82                      {
```

```
                        "name": "bic",
                        "label": "BIC/SWIFT",
                        "type": "text",
                        "maxlength": 100,
                        "required": true,
                        "pattern": "([a-zA-Z]{4}[a-zA-Z]..."
                    },
                    {
                        "name": "bank",
                        "label": "Bankname",
                        "type": "text",
                        "pattern": "[0-9]+(\\\\[0-9]*[a-z]?)?",
                        "required": false
                    }
                ]
            }
        ]
    }
}
```

Listing A.3: Field und CompositeField

```
// M-Search
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
ST: ssdp:all
MX: 3

// Notify
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
```

```
11 CACHE-CONTROL: max-age = 1800
12 LOCATION: http://192.168.1.100/description.xml
13 NT: upnp:rootdevice
14 NTS: ssdp:alive
15 SERVER: Unix/3.4.0.256021 UPnP/1.1 MyDevice.Ssdp/1.0
16 USN: uuid:d1::upnp:rootdevice
17 BOOTID.UPNP.ORG: 23
18 CONFIGID.UPNP.ORG: 4
```

Listing A.4: M-Search und Notify

B

Umfrage

1. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab.
   „Das entwickelte Feature zum Starten einer Umfrage auf dem Fernseher ist nützlich.“
- o Stimme zu
- o Lehne ab
- o Keine Angabe

2. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab.
   „Das entwickelte Feature zum Übertragen der Herzfrequenz ist nützlich.“
- o Stimme zu
- o Lehne ab
- o Keine Angabe

3. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab.
   „Ich würde mein Fitness-Portal verbinden um die Hilfestellung in Anspruch zu nehmen.“
- o Stimme zu
- o Lehne ab
- o Keine Angabe

Abbildung B.1: Zusatzfragen

1. Wie alt sind Sie?
o 0-30
o 31-50
o 51-X
o Keine Angabe

2. Sind Sie männlich oder weiblich?
o Männlich
o Weiblich
o Keine Angabe

3. Sind Sie im Besitz eines Smartphones?
o Ja
o Nein
o Keine Angabe

4. Sind Sie im Besitz eines Smart-TV (Internetfähiger Fernseher) ?
o Ja
o Nein
o Keine Angabe

5. Würden Sie sich als „technikbegeistert“ beschreiben?
o Ja
o Nein
o Keine Angabe

6. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab?
„Umfragen zur Erforschung von Krankheiten sind wichtig.“
o Stimme zu
o Lehne ab
o Keine Angabe

7. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab?
„Umfragen müssen eine Belohnung (Gewinnspiel, Bezahlung) als Anreiz besitzen.“
o Stimme zu
o Lehne ab
o Keine Angabe

8. Nehmen Sie regelmäßig (Mindestens 1x pro Monat) an Umfragen teil?
o Ja
o Nein
o Keine Angabe

9. Nutzen Sie ihr Smartphone zur Erfassung von sportlichen Aktivitäten?
o Ja
o Nein
o Keine Angabe

Abbildung B.2: Papierumfrage, Seite 1

10. Schreiben Sie folgenden Satz ab:
    „Ich nehme an einer Studie teil“

    _____

11. Schreiben Sie folgenden Satz ab:
    „Ich nehme an einer Studie ueber Umfragesysteme teil“

    _____

12. Schreiben Sie folgenden Satz ab:
    „Ich nehme an einer Studie ueber Umfragesysteme im Rahmen einer Masterarbeit teil“

    _____

Abbildung B.3: Papierumfrage, Seite 2

1. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab?
   „Regelmäßig Umfragen in Papierform auszufüllen ist zumutbar.“
- o Stimme zu
- o Lehne ab
- o Keine Angabe

2. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab?
   „Regelmäßig Umfragen auf dem Handy auszufüllen ist zumutbar.“
- o Stimme zu
- o Lehne ab
- o Keine Angabe

3. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab?
   „Regelmäßig Umfragen auf dem Fernseher auszufüllen ist zumutbar.“
- o Stimme zu
- o Lehne ab
- o Keine Angabe

4. Welchen zeitlichen Abstand zwischen Umfrage würden Sie als Maximum für Forschungszwecke einstufen?
- o Täglich
- o Wöchentlich
- o Monatlich
- o Jährlich
- o Keine Angabe

5. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab?
   „Die reguläre textuelle Eingabe über die Fernbedienung des Fernsehers ist im Rahmen einer Umfrage akzeptabel.“
- o Stimme zu
- o Lehne ab
- o Keine Angabe

6. Stimmen Sie folgender Aussage eher zu oder lehnen Sie diese eher ab?
   „Die textuelle Eingabe über das Mikrofon der Fernbedienung des Fernsehers ist angenehmer als die reguläre textuelle Eingabe.“
- o Stimme zu
- o Lehne ab
- o Keine Angabe

7. Welche Umfrageform bevorzugen Sie?
- o Fernseher
- o Smartphone
- o Papier
- o Keine Angabe

Abbildung B.4: Vergleichsfragebogen

TV (Virtuelle Tastatur)

35,45	48,38	90,76	156,37	1,78229984	3,93888889	
80,3	216,62	257,83	461,37	6,01442729	8,92222222	
37,21	86,06	123,38	185,46	2,53537745	4,13444444	
69,15	210,18	183,39	287,35	4,73125245	7,68333333	
40,82	68,43	97,15	193,9	2,20321732	4,53555556	
27,75	54,56	69,34	156	1,70942484	3,08333333	
34,66	53,59	84,61	131,04	1,69445098	3,85111111	
41,83	52,35	84,74	140,28	1,72002288	4,64777778	
21,42	64,19	78,32	117,8	1,71595098	2,38	
36,53	60,98	93,25	141,83	1,87799101	4,05888889	
37,66	51,25	83,68	129,77	1,65708088	4,18444444	
46,14	54,42	83,88	139,29	1,73327696	5,12666667	
40,45	75,71	99,15	128,66	2,02534477	4,49444444	Sek. p. F.
64,18	149,09	213,5	292,71	4,27160539	7,13111111	4,84765432
46,11	104,1	168,11	217,1	3,16000817	5,12333333	
54,14	74,09	98,68	149,56	2,09135621	6,01555556	
36,41	43,55	89,05	128,91	1,60304003	4,04555556	Sek. p. Z.
35,11	48,23	97	150	1,79487582	3,90111111	2,46227796

Abbildung B.5: Gemessene Zeiten (Smart-TV), Seite 1

TV (Mikrofon)

10,72	7,77	13,88	0,22772876		
6,74	9,52	10,99	0,18290278		
3,79	5,89	7,51	0,11189951		
5,44	9,19	8,47	0,15580147		
5,07	6,24	9,82	0,13803431		
13,41	112,58	9,43	0,92410866		
6,72	9,56	12,56	0,18948366		
4,1	6,06	7,64	0,11699673		
4,39	5,07	7,44	0,11291503		
4,17	7,31	7,83	0,12673611		
11,38	5,65	7,33	0,19391422		
4,76	5,47	10,76	0,13347386		
16,28	9,51	13,71	0,30017075		
8,09	15,73	23,54	0,29078268		
3,11	4,78	34,11	0,20792239		
24,27	24,87	48,39	0,63384069		
3,45	16,98	14,99	0,21177206	Sek. p. Z.	
4,44	13,12	14,17	0,19412663	0,24736724	

Abbildung B.6: Gemessene Zeiten (Smart-TV), Seite 2

Handy				Sek. p. Z.	Sek. p. F.	
27,94	18,94	22,63	29,47	0,48114461	3,10444444	
41,7	21,66	48,3	51,46	0,77076961	4,63333333	
18,71	14,36	26,32	33,52	0,47124837	2,07888889	
38,74	31,13	51,46	69,22	0,97064542	4,30444444	
31,33	11	20,22	40,91	0,42483742	3,48111111	
15,01	13,96	34,8	30,18	0,50831209	1,66777778	
42,82	18,58	27,16	40,65	0,55333578	4,75777778	
27,02	14,44	19,64	25,1	0,39339379	3,00222222	
16,82	5,48	12,26	16,81	0,21106127	1,86888889	
20,28	19,99	21,17	34,8	0,50547712	2,25333333	
42,46	10,04	24,01	34,59	0,41260866	4,71777778	
20,64	15,11	27,1	30,78	0,47326307	2,29333333	
18,23	6,72	11,22	16,75	0,21779167	2,02555556	Sek. p. F.
29,12	43,17	57,23	107,11	1,30001062	3,23555556	2,92802469
30,17	14,75	30,11	37,23	0,51581127	3,35222222	
18,64	9,9	17,4	28,94	0,34430882	2,07111111	
15,94	15,8	15,27	30,57	0,40273448	1,77111111	Sek. P. Z.
18,77	19,2	27,1	30	0,51545752	2,08555556	0,52623398

Abbildung B.7: Gemessene Zeiten (Smartphone)

Papier						
20,08	10,49	16,1	27,71	0,33724265	2,23111111	
37,45	14,66	23,4	38,68	0,47699673	4,16111111	
20,35	10,99	19,01	29,76	0,37035948	2,26111111	
24,96	10,83	16,88	26,55	0,34128513	2,77333333	
37,25	12,3	18,61	28,98	0,37905065	4,13888889	
21,9	13,04	20,73	30,96	0,40937908	2,43333333	
22,76	11,26	17,34	29,62	0,36186111	2,52888889	
26,8	10,24	19,99	29,13	0,36580637	2,97777778	
25,34	10,54	16,9	27,58	0,34248529	2,81555556	
22,43	10,21	20,13	34,04	0,38684641	2,49222222	
18,83	9,83	16,71	24,8	0,32177124	2,09222222	
21,4	11,84	23,49	35	0,4309183	2,37777778	
9,97	9,09	14,86	23,99	0,29808252	1,10777778	Sek. p. F.
28,71	9,51	20,17	30,51	0,36462173	3,19	2,65141975
37,27	9,09	18,77	30,72	0,35167974	4,14111111	
19,92	11,96	20,37	27,57	0,38090114	2,21333333	
15,11	10,23	18,3	28,94	0,35385784	1,67888889	Sek. p. Z.
19	10,87	15,2	19,98	0,30337418	2,11111111	0,3653622

Abbildung B.8: Gemessene Zeiten (Papierbogen)

Literaturverzeichnis

[4th15] 4TH LINE GMBH: *Cling - Java/Android UPnP library and tools*. http://4thline.org/projects/cling/. Version: 29.07.2015

[Ama15a] AMAZON.COM INC: *Amazon Fire TV*. https://developer.amazon.com/public/solutions/devices/fire-tv. Version: 05.08.2015

[Ama15b] AMAZON.COM INC: *Amazon Fire TV SDK Frequently Asked Questions*. https://developer.amazon.com/public/solutions/devices/fire-tv/docs/amazon-fire-tv-sdk-frequently-asked-questions. Version: 05.08.2015

[AOL15a] AOL DEUTSCHLAND MEDIEN GMBH: *Google Nexus Player*. http://de.engadget.com/2014/10/15/google-nexus-player-noch-eine-streaming-box-noch-ein-game-cont/. Version: 12.08.2015

[AOL15b] AOL INC: *Amazon goes after Rokku and Chromecast with 39 Fire TV Stick*. http://www.engadget.com/2014/10/27/amazon-fire-tv-stick/. Version: 12.08.2015

[App15a] APPLE INC: *Apple TV*. https://www.apple.com/de/appletv/. Version: 05.08.2015

[App15b] APPLE INC: *AppleCare Protection Plan für AppleTV*. http://www.apple.com/de/support/products/appletv.html. Version: 12.08.2015

[App15c] APPLE INC: *HealthKit*. https://developer.apple.com/healthkit/. Version: 12.08.2015

[Bec15] BECKER, Prof. Dr. F.: *Wirtschaftspsychologische Gesellschaft*. http://www.wpgs.de/content/blogcategory/79/336/. Version: 29.07.2015

[clu15] CLUETEC GMBH: *mQuest*. https://www.mquest.de/mymquest/produkt-demo/. Version: 29.07.2015

[Das15a] DAS STATISTIK PORTAL: *Absatz von internetfähigen Fernsehgeräten (Smart-TV) in Deutschland von 2009 bis 2014 (in Millionen Stück)*.

http://de.statista.com/statistik/daten/studie/183083/umfrage/anzahl-der-verkauften-internetfaehigen-fernseher-in-deutschland/. Version: 12.08.2015

[Das15b] DAS STATISTIK PORTAL: *Absatz von Smartphones in Deutschland in den Jahren 2008 bis 2015 (in Millionen Stück)*. http://de.statista.com/statistik/daten/studie/77637/umfrage/absatzmenge-fuer-smartphones-in-deutschland-seit-2008/. Version: 12.08.2015

[Ecm15] ECMA INTERNATIONAL: *The JSON Data Interchange Format*. 29.07.2015

[Gam15] GAMELOFT: *Amazon Fire TV*. http://blog.gameloft.com/index.php/2014/04/02/gameloft-amazon-fire-tv/. Version: 12.08.2015

[GG14] GIN GUO, Xingshe Z. Zhiwen Yu Y. Zhiwen Yu (Hrsg.): *From Participatory Sensing to Mobile Crowd Sensing*. Budapest : IEEE, 2014

[Goo15a] GOOGLE INC: *Android*. https://www.android.com/. Version: 12.08.2015

[Goo15b] GOOGLE INC: *Android TV*. https://www.android.com/tv/. Version: 12.08.2015

[Goo15c] GOOGLE INC: *Android TV*. https://developer.android.com/tv/index.html. Version: 29.07.2015

[Goo15d] GOOGLE INC: *App-Statistiken und -Berichte ansehen*. https://support.google.com/googleplay/android-developer/answer/139628?hl=de. Version: 29.07.2015

[Goo15e] GOOGLE INC: *Creating a Catalog Browser*. https://developer.android.com/training/tv/playback/browse.html. Version: 29.07.2015

[Goo15f] GOOGLE INC: *Developer*. http://developer.android.com. Version: 29.07.2015

[Goo15g] GOOGLE INC: *Fragments*. http://developer.android.com/guide/components/fragments.html. Version: 29.07.2015

[Goo15h] GOOGLE INC: *Google Fit*. https://developers.google.com/fit/. Version: 29.07.2015

[Goo15i] GOOGLE INC: *Providing a Card View*. https://developer.android.com/training/tv/playback/card.html. Version: 29.07.2015

[IBM15] IBM: *RESTful Web services: The basics*. http://www.ibm.com/developerworks/webservices/library/ws-restful/. Version: 30.07.2015

[Kem] KEMMERER, Tobias: *Umfragesysteme auf Basis mobiler Endgeräte - Untersuchung der Mehrwerte gegenüber papierbasierten und Online Fragebögen auf stationären Geräten*. – Diplomarbeit an der Goethe Universität, Frankfurt am Main

[KG94] KING G., S. V. R.O. Keohane K. R.O. Keohane: *Designing Social Inquiry, Scientific Inference in Qualitative Research*. University Press Group Ltd, 1994. – ISBN 0691034710

[Lin14] LINDINGER, Michael: *Konzeption und Implementierung einer mobilen Anwendung zur Untersütztung von Tinnitus-Patienten*. 2014. – Masterarbeit an der Uni Ulm

[Mac09] MACKENZIE, I. S.: *Empirical Research Methods for Human-Computer Interaction*. 07.04.2009

[Mic15] MICROSOFT CORPORATION: *Microsoft HealthVault*. https://www.healthvault.com/us/de. Version: 12.08.2015

[Net15] NETWORK WORKING GROUP: *Hypertext Transfer Protocol – HTTP/1.1*. 30.07.2015

[NVI15] NVIDIA CORPORATION: *Nvidia Shield TV*. http://shield.nvidia.com/android-tv. Version: 12.08.2015

[RKG11] RAGHU K. GANTI, Hui L. Fan Ye Y. Fan Ye (Hrsg.): *Mobile Crowdsensing: Current State and Future Challenges*. IEEE, 2011

[RS11] RAINER SCHNELL, Elke E. Paul B. Hill H. Paul B. Hill: *Methoden der empirischen Sozialforschung*. Oldenburg Wissenschaftsverlag GmbH, 2011. – ISBN 3486591061

[Sch15a] SCHINDLER, Arnim: *Konzeption und Entwicklung einer modularen, ereignisgesteuerten Server-Client-Architektur zur Crowd-basierten Datenerfassung mit mobilen Endgeräten*. September 2015. – Masterarbeit an der Uni Ulm

[Sch15b] SCHÄLING, Boris: *Das Use-Case Diagramm*. `http://www.highscore.de/uml/usecasediagramm.html`. Version: 29.07.2015

[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)

[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)

[Sta15] STATISTISCHES BUNDESAMT: *Haushalte und Familien*. `https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/Bevoelkerung/HaushalteFamilien/HaushalteFamilien.html`. Version: 12.08.2015

[Sur15] SURVIO S.R.O: *Survio*. http://www.survio.com/de. Version: 29.07.2015

[The15] THE VERGE: *Razer Forge TV*. https://cdn2.vox-cdn.com/thumbor/-tQ46SapoEwGc8WJHtj35qLHqNo%3D/cdn0.vox-cdn.com/uploads/chorus_asset/file/2902768/razer-forge-tv-4.0.jpg. Version: 12.08.2015

[UPn15] UPNP FORUM: *UPnP Device Architecture 2.0*. 29.07.2015

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Michael Schreiber Matrikelnummer: 850938

Erklärung

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

Ulm, den .....

Michael Schreiber