



Smartwatches in siot.net

Bachelorthesis 2016

Bachelorthesis im Studiengang Informatik an der Berner Fachhochschule

Studiengang: Informatik
Autoren: Sathesh Paramasamy
Betreuer: Dr. Andreas Danuser
Auftraggeber: AppModule AG
Experten: Armin Blum
Datum: 21. Januar 2016

Kontaktpersonen

Autor

Sathesh Paramasamy
Kirchweg 54
3324 Hindelbank

E-Mail: sathesh.paramasamy@students.bfh.ch

Betreuerender Dozent

Dr. Andreas Danuser
Berner Fachhochschule
Technik und Informatik
Höheweg 80
2502 Biel

E-Mail: andreas.danuser@bfh.ch

Beurteilender Experte

Armin Blum
Burgunderweg 58
2502 Biel

E-Mail: armin.blum@bluewin.ch

Management Summary

An der Berner Fachhochschule konzipiert und entwickelt die Fachgruppe SIOT des Institutes RISIS (Research Institute for Security in the Information Society) mit Industriepartnern (AppModule und NetModule) die siot.net-Plattform, welche Sensoren und Akteure mit Internet-of-Things-Anwendungen verbindet. Diese Plattform soll informatikfremde Personen und Unternehmen das Internet of Things näherbringen. Das Ziel ist Dinge zu vernetzen, welche miteinander kommunizieren sollen. Diese Umgebung befindet sich in der Erst-Implementationsphase und bestrebt für so viele „Things“ wie möglich den Zugang zu ermöglichen.

Einige dieser Dinge, die im Internet der Dinge eine grosse Rolle spielen, sind Smartwatches und Smartphones. Diese integrieren eine grosse Anzahl von Sensoren und Akteuren. Mit der effizienten Anbindung dieser Geräte an die siot.net-Plattform, können vielseitige Messdaten versendet und einkommende Informationen genutzt werden. Es sollen Bedürfnisse, Geräte und Anwendungsfälle für Smartwatches evaluiert und ausgewählt werden und diese mit einer generischen Architektur an die siot.net-Plattform angebunden werden.

Die vorliegende Bachelorthesis beschreibt einen Ansatz, wie Smartwatches und Smartphones ans siot.net angebunden wird. Die in dieser Arbeit spezifizierte Netzwerk-Architektur zeigt, wie die Kommunikation behandelt wird. In der Konzeption liegt der Schwerpunkt eine generische Bibliothek zu designen, welche im Android-Umfeld wiederverwendbar ist. Diese übernimmt die Verantwortung für das korrekte Management der Verbindungen und des Nachrichtentransfers zwischen den mobilen Geräten und der siot.net-Plattform. Anhand dieses Ansatz wird Applikationsentwicklern die Anbindung ans siot.net vereinfacht.

Um die Bibliothek zu verifizieren, wird eine Lösung in Form einer Android App entworfen und umgesetzt. Mit Hilfe der entwickelten Applikation, sollen alle Messwerte der eingebauten, standardisierten Sensoren, von Android Smartphones und Android Wear Smartwatches, publiziert werden. Für einen gewinnbringenden Einsatz der generischen Bibliothek zu ermöglichen, sind noch weitere Aspekte zu identifizieren und nachfolgend zu implementieren. Mit konsequenter Verfolgung der spezifizierten Anforderungen, Konzepte und Nacharbeiten, soll diese Bibliothek in einer Release-Version entstehen. Diese wird Android-Applikationsentwicklern unterstützen, ihre Apps mit wenig Aufwand ans siot.net anzubinden.

Inhaltsverzeichnis

1. Einleitung	1
1.1. Ausgangslage	1
1.2. Problemstellung	1
1.3. Zielsetzung	1
1.4. Abgrenzung	2
1.5. Projektmanagement	2
2. Grundlagen	3
2.1. Internet of Things (IoT)	3
2.2. Smartwatches	3
2.3. MQTT	3
2.4. Node-RED	4
2.5. Android und Android Wear	4
2.6. siot.net	4
2.7. siot.io	5
3. Marktsegmente	6
3.1. Marktsegmente im Internet of Things	6
3.1.1. Mensch	6
3.1.2. Natur	6
3.1.3. Industrie	6
3.1.4. Heimautomation	7
3.1.5. Automobil	7
3.1.6. Städte und Verkehr	8
3.1.7. Detailhandel	9
3.2. Marktsegmente für Smartwatches	9
3.2.1. Mensch	9
3.2.2. Zeit	9
3.2.3. Benachrichtigung	9
3.3. Marktsegmente für Smartwatches im Internet of Things	10
3.3.1. Mensch	10
3.3.2. Benachrichtigung	10
3.3.3. Heimautomation	10
3.3.4. Detailhandel	10
3.3.5. Ortsbezogen	10
4. Bedürfnisanalyse	11
4.1. Smartwatch Applikationen	11
4.1.1. Gesundheit	11
4.1.2. Smart Home	11
4.1.3. Sport	12
4.1.4. Ortsbezogen	12
4.1.5. Authentifikation	13
4.1.6. Finanztechnologie - FinTech	13
4.2. Smartwatch Applikationen für siot.net	13
4.2.1. siot.net Gateway Library	13
4.2.2. siot.net Sensorcenter	14
4.2.3. siot.net Dashboard App	14
4.2.4. Herzfrequenzüberwachung	14
4.2.5. Steuerung von Modellen	14

5. Technische Anforderungen	15
5.1. Allgemeine Applikationen Anforderungen	15
5.1.1. Gesundheit	15
5.1.2. Smart Home	15
5.1.3. Finanztechnologie - FinTech	15
5.2. siot.net Applikationen Anforderungen	16
5.2.1. siot.net Gateway Library	16
5.2.2. siot.net Sensorcenter	16
5.2.3. siot.net Dashboard App	17
5.2.4. Herzfrequenzüberwachung	17
5.2.5. Steuerung von Modellen	17
6. Technologiewahl	19
6.1. Allgemeine Eigenschaften	19
6.2. Kandidaten	20
6.3. Entscheid Smartwatch	21
7. Architektur	23
7.1. siot.net Architektur	23
7.2. Android Wear Paket Architektur	24
7.3. siot.net Android Applikationsarchitektur	24
7.4. Netzwerk-Architektur	25
7.4.1. geplante Architektur	25
7.4.2. effektive Architektur	27
8. Anforderungsdokumente	29
8.1. Requirements - siot.net Gateway Library	29
8.1.1. Allgemeine Beschreibung	29
8.1.2. User Requirements	29
8.1.3. Funktionale Anforderungen und Anwendungsfallbeschreibungen	30
8.1.4. Nicht-funktionale Anforderungen	35
8.2. Requirements - siot.net Sensorcenter	36
8.2.1. Allgemeine Beschreibung	36
8.2.2. User Requirements	36
8.2.3. Funktionale Anforderungen und Anwendungsfallbeschreibungen	37
8.2.4. Nicht-funktionale Anforderungen	40
8.3. Requirements - siot.net Dashboard	41
8.3.1. Allgemeine Beschreibung	41
8.3.2. User Requirements	41
8.3.3. Funktionale Anforderungen und Anwendungsfallbeschreibungen	42
8.3.4. Nicht-funktionale Anforderungen	45
9. Konzepte	46
9.1. Konzept - siot.net Gateway Library	46
9.1.1. Packagediagramm	46
9.1.2. Domäendiagramm	47
9.1.3. Sequenzdiagramme	48
9.2. Konzept - siot.net Sensorcenter	50
9.2.1. Domäendiagramm	50
9.2.2. Sequenzdiagramm	51
9.3. Konzept - Kommunikation Smartphone-zu-Smartwatch	52
9.3.1. Request-Response	52
9.3.2. Fire-and-Forget	52
10. Implementation	53
10.1. Abgrenzung	53
10.1.1. Ziele - siot.net Gateway Library	53
10.1.2. Ziele - siot.net Sensorcenter	53

10.2. Implementation - siot.net Gateway Library	53
10.2.1. Klassendiagramm	53
10.2.2. Logik	55
10.3. Implementation - siot.net Sensorcenter	55
10.3.1. Klassendiagramm	56
10.3.2. Logik	56
10.3.3. MessageAPI Messagetypen	57
10.3.4. Graphical User Interface	57
10.4. Plattformen und Libraries	58
10.4.1. Google Play Services	59
10.4.2. Google Gson	59
10.4.3. Eclipse Paho MQTT	59
10.5. Dokumentation	59
11. Ergebnis	60
11.1. Sensormessungen mit dem siot.net Sensorcenter	60
11.1.1. Voraussetzungen	60
11.1.2. Testresultate	60
11.2. Beurteilung	63
12. Fazit	64
12.1. Schlussfolgerung	64
12.2. Ausblick	65
Glossar	67
Literaturverzeichnis	69
Abbildungsverzeichnis	70
Tabellenverzeichnis	71
A. Entwicklerhandbuch	74
A.1. Download	74
A.2. Dokumentation	74
A.3. Erste Schritte	74
A.4. Entwicklung	75
B. Projektmanagement	76
B.1. Zeitplan	76
B.2. Kanban-Board Verlauf	79

1. Einleitung

1.1. Ausgangslage

Die Fachgruppe SIOT des Instituts RISIS der BFH konzipiert und entwickelt zusammen mit Industriepartnern (AppModule und NetModule) die Plattform siot.net, welche Sensoren und Aktoren weltweit mit IoT-Anwendungen verbindet. Smartwatches, welche eine rasante Marktakzeptanz geniessen, spielen eine grosse Rolle im Bereich IoT, denn sie integrieren eine Anzahl von Sensoren und können am Handgelenk Informationen anzeigen. Betreffend Funktionalität gibt es eine gewisse Spannweite bei den Smartwatches, was deren mögliche Einsatzgebiete schliesslich definiert.

1.2. Problemstellung

Die Projektarbeit 2 erlaubte Android Smartwatch zu analysieren. Diese Erkenntnisse sollen genutzt und mit einer praktischen Umsetzung konkretisiert werden. Dabei sollen folgende Themen genauer betrachtet werden:

- Welche Anwendungsklassen kann man für Smartwatches erkennen?
- Wie werden Smartwatches am weltweiten Internet angebunden?
- Welche GUI-Elemente werden bereitgestellt?
- Welche Sensoren und Aktoren stehen zur Verfügung?

Die Bachelorarbeit beinhaltet eine Markt- und Bedürfnisanalyse, welche die Marktsegmente und die Bedürfnisse aus Sicht IoT für Smartwatch aufzeigen. Für die identifizierten Anwendungen werden Smartwatches evaluiert.

Als weitere Aufgabe wird eine generische System-Architektur definiert, mit welcher Software für Smartwatches für IoT-Anwendungen im siot.net Umfeld realisiert werden kann.

In einem formalen Teil werden die Anforderung bzw. technischen Anforderung, einer bestimmten Smartwatch an eine Anwendung gestellt, untersucht und aufgezeigt. Hierbei sollen auch Genauigkeiten und Zuverlässigkeit betrachtet werden.

Es wird ein Softwaredesign erstellt mit welchem zwei bis drei konkrete Anwendungen implementiert werden könnten. Daraus wird mindestens eine konkrete Anwendung umgesetzt. Zur Realisierung wird eine Dokumentation erstellt welche von Informatik-Ingenieuren gelesen wird.

Schlussendlich werden in diesem Dokument alle Ergebnisse berichtet.

1.3. Zielsetzung

Mit einer Bedürfnisanalyse sollen Anwendungsfälle für Smartwatches erarbeitet werden. Mit den entdeckten Use-Cases werden aktuelle Smartwatches evaluiert und mindestens eine wird genauer betrachtet. Um eine geeignete Plattform für die Softwareentwicklung der gewählten Uhr aufzubauen, wird eine Entwicklungsumgebung eruiert.

Die erstellten Grundlagen helfen eine generische Softwarebibliothek zu erstellen. Mit diesem Stück Software wird ermöglicht, Smartwatches und Smartphones schnell an die siot.net-Plattform anzubinden. Entwickler von Apps erleichtert dies die Arbeit, denn für die Verbindung an das IoT-System kann die Bibliothek verwendet werden. Bei der generischen Anbindung wird in erster Linie das Hauptaugenmerk auf Sensordaten, sowie die Verbindung von Smartwatches gelegt. Zusätzlich wird Programmierern ein Entwicklerhandbuch bereitgestellt. Dieses erläutert die Möglichkeiten (JavaDoc) und beschreibt ein kleines Tutorial.

Einige Funktionen werden mit einer Applikation gezeigt, welche die siot.net Anbindungsbibliothek integriert.

1.4. Abgrenzung

Smartwatches im Allgemeinen gibt es von vielen verschiedenen Anbietern. In dieser Arbeit werden aller Art Smartwatches analysiert. Der Schwerpunkt für das Softwaredesign und die Implementationen liegt auf Smartwatches, die mit dem Betriebssystem Android ausgeliefert werden. Diese Abgrenzung findet statt, um die Entwicklung auf ein offenes System (Open Source) zu beschränken. Weiterhin wird die Stabilität und Sicherheit nicht genauer betrachtet, was im Rahmen dieser Arbeit nicht abzudecken wäre.

1.5. Projektmanagement

Für die Organisation der Arbeit ist ein Zeitplan erstellt und eine Prozesssteuerung verwendet worden. Als Prozesssteuerung diente Kanban. Da dies eine Einzelarbeit war, hielt die schlanken Zeit- und Prozessplanungsmethoden den Overhead in Grenzen. Als Dokumenten- und Quelltextablage, diente die Cloud Plattform github.com, diese verwendet als Versionskontrollsysteem Git. Zu finden sind die Daten auf folgendem Verzeichnis: <https://github.com/paras1/sw-siot>.

1.5.1. Zeitplan

Für die Zeitplanung kam ein tabellarischer Zeitplan zum Einsatz. Der Plan ist im Anhang zu betrachten.

1.5.2. Prozesssteuerung - Kanban

Das Kanban Modell kommt ursprünglich aus Japan und heisst Signalkarte. Der Begriff Signalkarte, weil auf einer Tafel der Fortschritt des Projektes sichtbar ist. Entwickelt wurde das System durch den Toyota Konzern, welches für die Fertigung ihrer Produkte dienen soll. David J. Anderson (www.djaa.com) hat das Modell im Jahre 2007 für die Informationstechnologie adaptiert. Bei dieser Prozesssteuerungsart wird eine Prozesskette definiert und dann Aufgaben, welche Tickets genannt werden, ins Backlog erfasst. Das Kanbanboard besteht aus den Spalten der Prozesskette und dem Ticketstatus. Zur Bearbeitung dieser Bachelorthesis sind folgende Prozessspalten definiert worden: Backlog, Bereit (Ready), In Arbeit (In Progress), Erledigt (Done). Die Backlog Tickets sind dem tabellarischen Zeitplan zu entnehmen. Typischerweise wird bei einem Kanbanprojekt eine maximale Anzahl an Tickets zur gleichzeitigen Bearbeitung definiert. Bei dieser Einzelarbeit, wurde darauf verzichtet, um den Arbeitsfluss nicht zu hindern. Das Kanbanboard wurde nicht physisch geführt, sondern mit einer Webapplikation¹. Diese Applikation bildet den Backlog aus der Versionisierungsablage² in die definierte Prozesstafel ab. Nach beenden eines Tickets werden diese nach fünf Tagen aus dem Prozesssteuerungssystem entfernt. Dies hält die Übersichtlichkeit hoch. In der Versionisierungsablage (GitHub Issues) werden diese, mit dem aktuellen Status, dauerhaft gespeichert.

¹Kanbanboard: <https://waffle.io/paras1/sw-siot>

²GitHub Backlog: <https://github.com/paras1/sw-siot/issues>

2. Grundlagen

2.1. Internet of Things (IoT)

Das Internet of Things ist eine Struktur, welche alle Objekte mit einer eindeutigen Identität gekennzeichnet sind. Dadurch ist die Möglichkeit gegeben, wenn die Dinge verbunden sind, dass Informationen über ein Netzwerk übermittelt werden können. Dies kann ohne Interaktion von Mensch-zu-Mensch oder Mensch-zu-Computer durchgeführt werden. Das IoT hat sich aus dem Zusammenspiel von der drahtlosen Kommunikation, dem Internet und mit den MEMS (Micro-Electromechanical Systems) entwickelt.

Ein „Thing“ im IoT kann zum Beispiel eine Boje mit eingebauten Sensoren, ein selbstfahrendes Fahrzeug, ein Mensch mit einem Herzschrittmacher oder auch ein Haustier mit einem Biochip sein. Jedes dieser Objekte kann nützliche Informationen preisgeben. Man kann durchaus sagen, dass jedes vom Menschen geschaffene Objekt ein Kandidat dazu ist. Die Voraussetzung ist, dass es sich mit einer Netzwerkkadresse beschreiben lässt und es Daten mittels eines Netzwerks übertragen kann. Die Dinge im IoT zeichnen sich durch die Fähigkeit der Maschine-zu-Maschine Kommunikation aus. Deshalb werden diese Objekte oft auch als intelligent oder smart betitelt (vgl. [12]).

Seit Jahren wird Forschung auf diesem Gebiet getrieben, um die Informationslücke zwischen der realen und der virtuellen Welt zu vermindern. Laut dem Forbes Magazin wird es bis Ende Jahrzehnt (2020) bereits 30 Milliarden vernetzte Dinge geben (vgl. [12]). Cisco spricht von 50 Milliarden „Things“ im Jahre 2025 (vgl. [7]).

2.2. Smartwatches

Smartwatches sind kompakte Computersysteme, welche vom Benutzer am Handgelenk getragen werden kann. Diese können viele verschiedene Funktionalitäten mit einem Gerät abdecken. Die Minicomputer sind meist mit einer oder mehreren drahtlos Technologie und verschiedenen Sensoren (Bewegungssensor, Lichtsensor, Herzfrequenzmesser) Aktoren (Bildschirm, Vibrationsmotor) ausgerüstet.

Diese Uhren unterstützen den Träger beim alltäglichen Leben. Sie gehören zur Gruppe der Wearables und damit zu einem essentiellen Bereich des IoT. Die zurzeit grössten Player auf dem Markt sind Apple, mit der Apple Watch, und Google, mit den Android Wear Geräte verschiedener Hersteller.

2.3. MQTT

MQTT wurde im Jahre 1999 von Andy Stanford-Clark (IBM) und Arlen Nipper (damals Eurotech) entwickelt um eine Ölipeline quer durch die Wüste zu überwachen. Das Ziel war ein Protokoll zu erhalten, welches Bandbreiteneffizient ist und wenig Energie konsumiert. Dies musste erreicht werden, weil die eingesetzten Geräte über Satelliten verbunden waren. Zu dieser Zeit war dies sehr teuer.

Das Protokol benutzt die publish/subscribe Architektur, im Gegensatz beansprucht HTTP request/response. Publish/subscribe ist ereignisgesteuert und erlaubt, Nachrichten an den Empfänger zu pushen. Der zentrale Kommunikationspunkt ist der MQTT Vermittler, auch MQTT Broker genannt. Dieser hat die Verantwortung, die Mitteilung zwischen den Sendern und den richtigen Empfängern zu verteilen. Das Routing geschieht anhand von sogenannten Themen (Topics). Topics sind vergleichbar mit Ordnerstrukturen. Sie beginnen mit einem Thema und werden anhand von Slashes unterteilt in Unterthemen. Jede Nachricht beinhaltet ein Topic. Dieser dient für den Broker als Verteilschlüssel. Jeder Empfänger abonniert die gewünschten Topics und der Broker stellt ihnen die Meldungen mit dem passenden Wert zu. Das hat den Vorteil, dass Sender und Empfänger gegenseitig nicht bekannt sein müssen. Diese Architektur erlaubt eine in hohen Massen skalierbare Lösung ohne Abhängigkeiten zwischen Datenproduzent und Konsument.

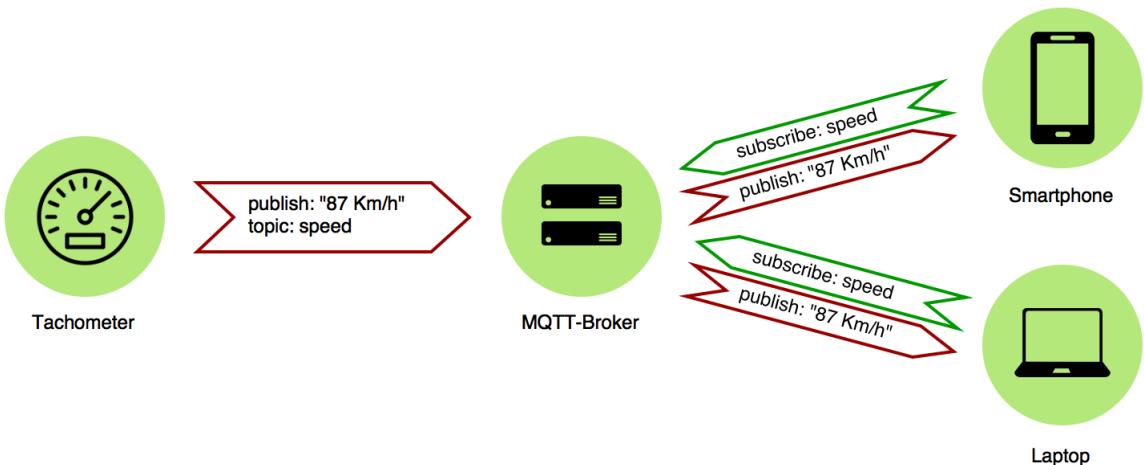


Abbildung 2.1.: Die Publish/Subscribe Architektur von MQTT

Der Unterschied zum HTTP Protokoll ist, dass ein dienstanforderndes Gerät die Informationen nicht holen muss, sondern direkt geliefert bekommt. Voraussetzung dafür ist eine immer offene TCP Verbindung vom Client zum Server. Falls diese Konnexions unterbrochen werden sollte, kann der MQTT Broker Nachrichten zwischenlagern und dann pushen, wenn dieser wieder Verfügbar ist (vgl. [6]).

2.4. Node-RED

Node-RED ist ein Open Source Tool der Firma IBM, um „Things“, APIs und Online Services miteinander zu verbinden. Es ist ein browserbasierender Flusseditor. Es basiert auf Node.js und ist eine effiziente Applikation, um Verbindungen im IoT zu erstellen. Es bietet auch die Möglichkeit Aktionen auszuführen, Scripts zu definieren und auch andere Online Services zu koppeln (vgl. [5]).

2.5. Android und Android Wear

Google's Android ist das marktführende Betriebssystem (vgl. [14]), welches in aller Munde ist. Es betreibt Smartphones und Tablets unterschiedlicher Hersteller und ist ein Open-Source Produkt. Android Wear ist das OS von Google für Wearables, wie Smartwatches. Es basiert auf Android, ist optimiert für kleine Geräte mit weniger Leistung und kurzer Akkulaufzeit.

2.6. siot.net

Die Plattform siot.net ist entstanden, um das Bedürfnis von Nutzern zu stillen, welche Geräte und Dingen ans Internet anbinden wollen, die noch nicht vernetzt sind. Im Interesse der Industriepartner, ist das Ziel diese Plattform zu industrialisieren, um Klein- und Mittlere-Unternehmen (KMU) eine Möglichkeit zu geben ihre Sensoren, Geräte, Maschinen und viele weitere Dinge zu vernetzen. Die Plattform soll ermöglichen, dass auch informatikfremde Personen und Firmen das Internet of Things nutzen können.
 siot.net bietet ein IoT-Center (siehe Abbildung 2.1) mit der IoT-Infrastruktur. Konfiguration und Verwaltung werden im IoT-Center durchgeführt. Die IoT-Infrastruktur beinhaltet einen MQTT Broker und das siot-Interface. Das ist die Schnittstellenspezifikation von siot.net. Nicht nach Schema angelieferte Nachrichten kann das IoT-Center nicht interpretieren (vgl. [13]).

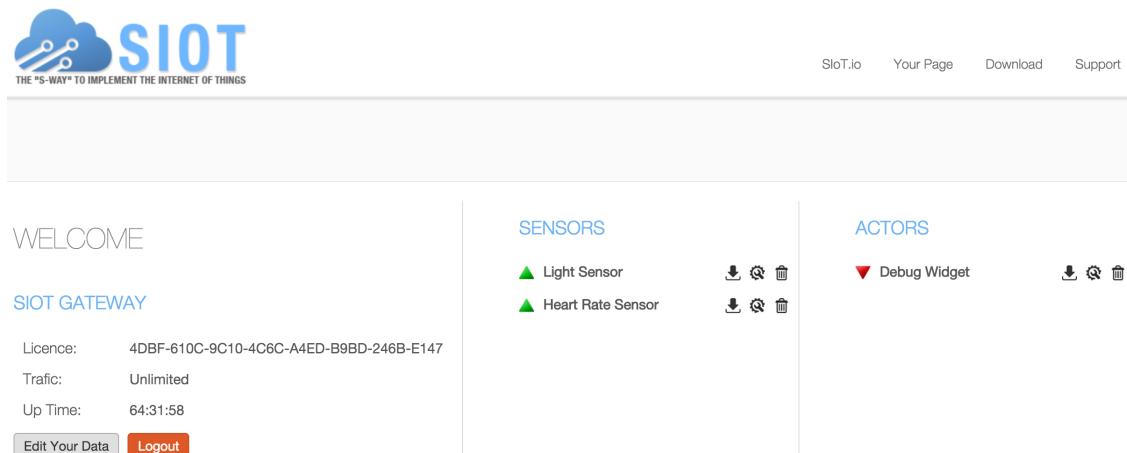


Abbildung 2.2.: siot.net IoT-Center

2.7. siot.io

Für die Verwertung der Daten, welche die siot.net-Plattform sendet und empfängt ist das Projekt siot.io entstanden. siot.io soll die Webschnittstelle zur siot.net-Plattform implementieren. Dabei bietet sie einen Sensorsimulator, eine konfigurierbare Dashboard Applikation mit Widgets, sowie eine Node-RED Instanz für eigene Applikationen.

3. Marktsegmente

Im Kapitel Marktsegmente werden die aktuellen Bereiche von Internet of Things, Smartwatches und Smartwatches im Internet of Things aufgezeigt. Es werden Themen aufgelistet und Bereich davon aufgezeigt. Hierfür ist keine strategische Marktsegmentierung durchgeführt worden, dadurch ist es keine abschliessende Auflistung. Für die Evaluation der Marktsegmente wurde der Bericht, The Internet of Things: Mapping the value beyond the hype von McKinsey Global Institute als Referenz verwendet (vgl. [7]).

3.1. Marktsegmente im Internet of Things

Mensch:	Blutdruck, Puls, Bewegungen, Schlafüberwachung, Körperanalyse (z.B. Gewicht, Fettanteil, Wasseranteil usw.)
Natur:	Erdplattenbewegung, Wasserspiegel Überwachung, Temperatur, Wind, Licht, Luft
Industrie:	Maschinensteuerung, automatisierte Roboter, Lagerüberwachung
Heimautomation:	Nutzung und Überwachung von Haushaltsgeräte, Steuerung, Fernbedienungen
Automobil:	Telemetrie, Geografische Strecke, Fahrverhalten, Nutzungsverhalten, Verkehrsbericht
Städte/Verkehr:	Touristisches Informationen, Dynamische Strassen, Verkehrsregulierung, Navigation, Lageberichte
Detailhandel:	Produktebezeichnung, Kasse, Geldüberweisung, Geldbörse

Der Auflistung zu entnehmen, kommt das Internet der Dinge in sehr vielen verschiedenen Marktsegmenten zum tragen. Es hat noch grosses unausgeschöpftes Potential den Menschen zu unterstützen um seine Aufgaben zu erleichtern.

3.1.1. Mensch

Der Mensch ist ein wichtiges Marktsegment. Hierbei können Sensoren aller Art den Menschen analysieren. Dieser Punkt wird bei der Marktsegmentanalyse von Smartwatches und Smartwatches im Internet of Things genauer betrachtet.

3.1.2. Natur

Viele verschiedene Anwendungsfälle gibt es auch in der Natur. Es können Sensoren eingesetzt werden, um Temperaturen, Luftdruck, Luftfeuchtigkeit oder Windstärke zu messen. Mit Kombinationen von Sensoren, welche miteinander kommunizieren, können Frühwarnsysteme von Naturkatastrophen erschaffen werden. Dieses Segment hängt sehr nahe mit dem Marktsegment des Menschen zusammen.

3.1.3. Industrie

Das Internet der Dinge kommt in der Industrie soweit zum tragen, dass man von Industrie 4.0 spricht. Dies soll die vierte industrielle Revolution zum Ausdruck bringen. Die Fertigungstechnologie soll informatisiert werden. Auch die Logistik soll ihre Automatisierung erleben. Erreicht wird dies, weil Maschinen untereinander kommunizieren können. Möglichst alle Sektoren einer Fabrik sollen vernetzt sein. Das Ziel der Industrie 4.0 ist die intelligente Fabrik.

3.1.4. Heimautomation

Die Heimautomation ist auch besser bekannt als Smart Home. Smart Home dient als Oberbegriff für technische Verfahren und Systeme in Wohnräumen und -häusern, in deren Mittelpunkt eine Erhöhung von Wohn- und Lebensqualität, Sicherheit und effizienter Energienutzung auf Basis vernetzter und fernsteuerbarer Geräte und Installationen sowie automatisierbarer Abläufe steht.

Unter diesen Begriff fällt sowohl die Vernetzung von Haustechnik und Haushaltsgeräten (z.B. Lampen, Jalousien, Heizung, aber auch Herd, Kühlschrank und Waschmaschine), als auch die Vernetzung von Komponenten der Unterhaltungselektronik (wie etwa die zentrale Speicherung und heimweite Nutzung von Video- und Audio-Inhalten).

Von einem Smart Home spricht man insbesondere, wenn sämtliche im Haus verwendeten Lampen, Taster und Geräte untereinander vernetzt sind, Geräte Daten speichern und eine eigene Logik abbilden können. Geräte sind teilweise auch getagged. Dies bedeutet, dass zu den Geräten im Smart Home Informationen z.B. über Hersteller, Produktnamen und Leistung hinterlegt sind. Dabei besitzt das Smart Home eine eigene Programmierschnittstelle, die (auch) via Internet angesprochen und über erweiterbare Apps gesteuert werden kann.

Eng verwandt mit diesen Verfahren und Systemen sind solche des Smart Metering, bei denen der Schwerpunkt auf dem Messen und einer intelligenten Regulierung des Energieverbrauchs liegt (vgl. [17]).

3.1.5. Automobil

IoT kann in vielen Bereichen der Automobilbranche eingesetzt werden. Es können wichtige Daten des Fahrzeugs ausgelesen werden, z.B. die Telemetriedaten. Diese können verwendet werden um das Fahrverhalten vom Lenker festzustellen. Des Weiteren kann, durch Nutzen der Daten, Probleme beim Auto ausgemacht und direkt Fahrer und Mechaniker alarmiert werden. Interessant, für die Autobauer wie auch Autobesitzer, ist die Ortung der Fahrzeuge. Mit den aufgezeichneten geografischen Punkten kann analysiert werden, wie das Automobil verwendet wird und aktuelle verkehrsnahe Verkehrsberichte können genutzt werden. Die Vollendung der Vernetzung von Fahrzeugen ist das selbstfahrende Auto, welches alle nötigen Informationen empfängt, analysiert und verwendet, um das Ziel optimal zu erreichen.

Mercedes-Benz hat ein solches selbstfahrendes Forschungsfahrzeug entwickelt (vgl. [10]). Die Abbildung 3.1 zeigt, wie das Fahrzeug durch die Sensoren einen Fußgänger erkennt und die Laserprojektionstechnik als Aktor verwendet, um dem Überquerenden die Fußgängerstreifen anzudeuten.

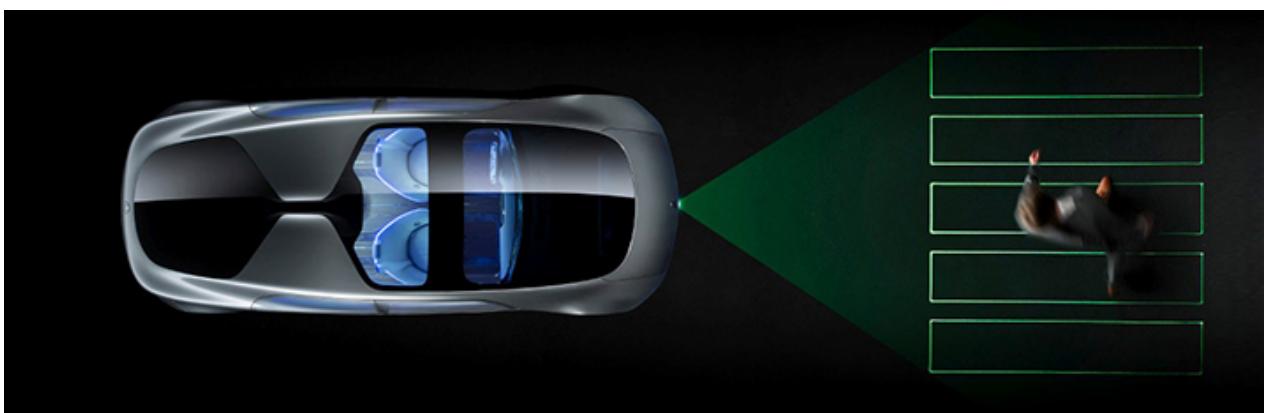


Abbildung 3.1.: Der Mercedes-Benz F 015 erkennt Fußgänger mit seinen Sensoren

Quelle: http://www.mercedes-benz.ch/content/media_library/f_015_luxury_in_motion_gallery_05_715x230_01-2015.jpg,
Stand: 05.11.2015

3.1.6. Städte und Verkehr

In Städten gibt es sehr viele Möglichkeiten. In Verbindung mit dem Verkehr geht dies ins unermessliche. Ein sehr interessantes Thema ist die Touristik. Um ein Beispiel zu nennen: Beacons, welche nötige Information an ein smartes Gerät publizieren, um Daten von der Sehenswürdigkeit abzurufen. Dazu könnte auch gleich Empfehlungen in der Umgebung notifiziert werden. So würde für die meisten Reisenden der Reiseführer wegfallen.

Ein spektakuläres Projekt ist die dynamische Strasse: **Solar Roadways**. Das sind kleine, feste Platten, welche Photovoltaik-Elementen, Elektronik, verschiedenen Sensoren und LEDs integrieren (siehe Abbildung 3.2). Die Platten können wie Pflastersteine verlegt und miteinander verbunden werden. Durch die Sonneneinstrahlung sind sie permanent und umweltschonend mit Strom versorgt. Die LEDs können zentral gesteuert werden, um so die Fahrbahnmarkierungen anzuzeigen und z.B. aus zwei breiten Spuren drei schmale machen, spontane Parkflächen oder Verkehrszeichen. Die Sensoren können feststellen, wenn Tiere oder Menschen darüber laufen und die Fahrer, schon ein paar hundert Meter vorher, über die LEDs warnen. Und die Platten sind beheizbar. Dies erhöht die Verkehrssicherheit und verringert vermutlich Baustellen. Es wurde von Privatpersonen initiiert und gecrowdfunded. Das Vorhaben ist auf Indiegogo (www.indiegogo.com) im Juni 2014 deutlich überfinanziert abgeschlossen worden. In Holland hat man im Jahr 2014 begonnen, Radwege auf diese Weise zu bauen (vgl. [16]).



Abbildung 3.2.: Das Solar Roadway verändert die Strasse für die aktuelle Verkehrssituation
Quelle: <http://www.solarroadways.com/images/intro/Downtown%20Sandpoint%202%20-%20small.jpg>, Stand: 05.11.2015

3.1.7. Detailhandel

Im Verkauf hat die Revolution schon teilweise begonnen. Die grossen Unternehmen in der Schweiz beginnen alle ihre Filialen mit WLAN auszustatten. Momentan bieten diese den WiFi Zugang zur freien Verfügung den Kunden an. Somit steht der Kommunikationskanal für das IoT im Laden bereit und die Kunden sind zur gegebenen Zeit bereits verbunden damit. Weiter werden heutzutage Selbstbezahlkassen eingesetzt. Momentan werden zwei Modelle verfolgt. Eine Einkaufsart ist: Der Kunde wählt seine Produkte und scannt diese selber ein, bezahlt mit Karte oder Bar und verlässt das Geschäft. Bei der IoT näheren Methode, registriert sich der Konsument beim Eingang an einem Terminal und rüstet sich mit einem mobilen Strichcodeleser der Filiale aus oder nimmt sein Smartphone als Scanner. Die Person liest alle Produkte mit dem Scanner ein und bezahlt, beim verlassen des Ladens, am Bezahlterminal. Beim abmelden des Scanners wird der Kunde ermittelt und das Total eingefordert.

Ein weiterer Schritt ist hier, alle Produkte mit einer RFID zu taggen. Somit könnte der Kunde nur seine Kreditkarte registrieren, die Ware in den Einkaufskorb legen und die Filiale verlassen. Beim verlassen wird durch die Information auf dem RFID Tag gemerkt, welche Ware mitgenommen wurde und die Kreditkarte wird automatisch belastet.

3.2. Marktsegmente für Smartwatches

Mensch:	Blutdruck, Puls, Bewegungen, Schlafüberwachung, Lebensüberwachung, Sportbeobachtungen, Sporttracking
Zeit:	Individuelle Zeitansichten, Zeitfunktionen
Benachrichtigung:	Informationen am Handgelenk, Kommunizieren

Momentan werden Smartwatches hauptsächlich zu Notifikationszwecken und Fitnesstracking des Menschen genutzt. Noch wird das Potenzial nicht ausgenutzt Smartwatches in vielen anderen Segmenten einzusetzen. Um dies zu ermöglichen müssen die Bedürfnisse zuerst erkannt oder geschaffen werden.

3.2.1. Mensch

Heute werden Smartwatches verwendet, um den Menschen bei Aktivitäten überwachen zu können. Diese Wearables verfügen viele eingebaute Sensoren, die die Bewegungen des Trägers analysieren und dem interessierten die Daten zur Verfügung stellen. Zu den Sensoren gehören z.B. ein Bewegungssensor, Schrittzähler, Herzfrequenzmesser und viele mehr. Viele Hersteller von Smartwatches rüsten Ihre Produkte mit Sensoren und mit Auswertungsapplikationen (z.B. Google Fit, Apple Health oder Motorola Moto Body) aus. Mit diesen Apps kann der User seine Daten während dem Training auf der Uhr verfolgen oder später auf dem Smartphone auswerten. Dies macht zusätzliche Sport-/Pulsuhren überflüssig.

3.2.2. Zeit

Die Hauptaufgabe einer Uhr ist es, die Uhrzeit genau anzuzeigen. Die Smartwatches haben nicht nur die Möglichkeit die aktuelle Uhrzeit anzuzeigen, sondern auch als Weltuhr, Stoppuhr und Countdown-Rechner zu fungieren. Dabei hat der Träger der Computeruhr die Wahl, wie das Ziffernblatt aussehen soll. Wie individuell sie gestaltet werden wird vom Benutzer oder Entwickler bestimmt. Für Individualisten ist sie sehr geeignet.

3.2.3. Benachrichtigung

Eine Smartwatch wird neben der Uhrzeitfunktion auch als Notifikationsbildschirm verwendet. Alle relevanten Benachrichtigungen an ein Smartphone können auch von der Smartwatch angezeigt werden. Dabei dient die Uhr meist als verlängerter Arm des Mobilgerätes. Es können Nachrichten empfangen, Telefonate geführt, Erinnerungen ausgelöst, der Wecker gestellt werden oder anzeigen was auf dem Smartphone ausgeführt wird, z.B. aktuell abgespieltes Musikstück.

3.3. Marktsegmente für Smartwatches im Internet of Things

Mensch:	Blutdruck, Puls, Bewegungen, Schlafüberwachung, Gesundheitsbenachrichtigung
Benachrichtigung:	Alarne, Informationen
Heimautomation:	Fernbedienung, Statusanzeigen, Alarming
Detailhandel:	Geldbörse, Produktebezeichnung, Einkaufsliste
Ortsbezogen:	Navigation, Ortsspezifische Informationen, Ortung, Personen in der Nähe

3.3.1. Mensch

Um die Gesundheit eines Menschen zu überwachen, eignet sich eine Smartwatch sehr gut. Sie ist immer am Handgelenk und kann bei Unregelmässigkeiten Alarm schlagen. Durch die Vernetzung werden die Daten auch an anderen Geräten zur Verfügung gestellt. Eine wichtige Benachrichtigung kann von einer anderen Smartwatch oder einem Smartphone verwendet werden. Ein sich vorstellbares Szenario: Ein Paar, beide tragen eine smarte Uhr, bei einer potenziellen Gefahr, z.B. schwacher/kein Puls, Sturz, ungewöhnliche Bewegungsabläufe, wird der Andere alarmiert.

3.3.2. Benachrichtigung

Um Informationen darzustellen, eignet sich eine Smartwatch nur beschränkt so gut, wie ein Smartphone oder andere grössere Anzeigegeräte. Sie ist jedoch prädestiniert einzelne kleine Datenmengen anzuzeigen. Die im Internet der Dinge übermittelten und verwerteten Daten, können für den Menschen, auf einem praktisch sichtbaren Display am Handgelenk, angezeigt werden. Dies gewährt einen schnellen Zugriff zu den Benachrichtigungen.

3.3.3. Heimautomation

Im Smart Home Bereich kann die Kombination Smartwatch und Internet of Things ihre stärken ausspielen. Durch den Zusammenschluss aller Haushaltgeräte, wie Fernseher, Lampen oder Herdplatte, sind alle Daten zentral erreichbar und verwaltbar. Nun bestehen viele Möglichkeiten die Computeruhr ins System einzubinden. Einige geeignete Anwendungsfälle sind: Das Licht ein- und auszuschalten, Alarmierung von nicht ausgeschalteten Haushaltsgeräten oder das Fernbedienen von Geräten. All dies soll unabhängig vom Ort des Uhrträgers möglich sein.

3.3.4. Detailhandel

Im Detailhandel sind besonders Finanztechnologie Applikationen schon stark vertreten, wie Apple Pay, Google Wallet oder auch die schweizerische Lösung TWINT. Diese Anwendungen erlauben Geldüberweisungen mit Smartphones oder Smartwatches mittels drahtloser Verbindung über ein Terminal, welches die Bezahlung anfordert. Des weiteren könnten Smartwatches in Selbstbedienungsgeschäften als erweiterter Informationsschild von Produkten dienen. Durch die Anbindung ans Internet hat man die Möglichkeit, jedes weitere Detail, welches nicht auf der Ware beschrieben ist, auf dem Bildschirm am Arm anzuschauen. Ein sehr grosser Vorteil ist, dass verschiedene Medien genutzt werden können, z.B. detailliert Nährwertangaben, Anleitungsfilme aller Art, Explosionszeichnungen von Modellen und viele weitere.

3.3.5. Ortsbezogen

Applikationen für die Ortung oder zur Anzeige standortabhängiger Daten werden bei vielen Anwendungen benutzt. Dies auf die Smartwatch zu erweitern ist ein logischer Schritt. Ein ortssrelevantes Thema kann direkt am Handgelenk angeschaut werden ohne das Smartphone heranzuziehen.

4. Bedürfnisanalyse

Im Kapitel der Bedürfnisanalyse werden Anwendungsfälle ermittelt, welche mit Smartwatches abgedeckt werden können. Zusätzlich wird eine grobe Bedarfsanalyse für Applikationen durchgeführt, welche in Verbindung zur siot.net-Plattform stehen.

4.1. Smartwatch Applikationen

4.1.1. Gesundheit

Im Gesundheitssektor gibt es einige Anwendungsfälle, welche mit einer smarten Uhr abgedeckt werden können.

Eine Smartwatch bietet die Möglichkeit, sich überwachen zu lassen. Da die Computeruhr mit vielen Sensoren, wie z.B. Bewegungssensor oder Herzfrequenzmesser ausgerüstet ist, hat sie die Möglichkeit, den Träger sehr genau zu analysieren.

Ein Sturz des Benutzers kann durch die Auswertung der Sensormessdaten erkannt werden. Dies kann genutzt werden um einen Benachrichtigungen an Personen oder Systeme auszulösen, welche diese Informationen empfangen dürfen. Um nicht jede Ungereimtheit zu melden, sollte ein Zwischenschritt vorhanden sein. Den Träger mit taktilen und optischen Signalen aufmerksam machen, dass ein Gesundheitsbericht gesendet wird. Falls keine Reaktion erfolgt, kann ein Alarm an Vertrauenspersonen oder bei schwerwiegenden Bedrohungen gleich ein Notruf ausgelöst werden. Dieser Notruf kann mit wichtigen Daten angereichert werden, wie z.B. aktuelle Pulsdaten, die GPS Koordinaten und Gesundheitskarte des Patienten. Der Sturz ist nur ein Beispiel, dieser Anwendungsfall kann für weitere Gesundheitsgefährdende Lebenslagen verwendet werden.

Ein weiterer nützlicher Use-Case ist, Alarne von Patienten im Spital. Hier kann das Pflegepersonal mit Smartwatches, Patientenalarme erhalten. Die Alarne sollten möglichst nur Pfleger/innen erhalten, welche sich in der Nähe des Patienten aufhalten. Wenn mehrere Pfleger/innen benachrichtigt werden, kann eine Pflegeperson den Alarm bestätigen und die Verantwortung für den Patienten übernehmen. Dies verkürzt die Reaktionszeit und vermeidet Doppelspurigkeiten.

4.1.2. Smart Home

Für die Fernbedienung von Geräten im Haus oder Wohnung eignet sich die Smartwatch gut. Mit eingebautem Touchscreen und Vibrationsmotor, haben die kleinen Handgelenkrechner die Möglichkeit, Informationen visuell wie auch taktil an die Person zu bringen.

Geräte im Haushalt können überwacht werden. Dies hilft Gefahren abzuwenden. Wenn eine Herdplatte noch läuft, kann ein Alarm ausgelöst werden und es kann gleich mit der Uhr die Platte ausgeschalten werden.

Die Funktion, das Licht vom Handgelenk aus zu bedienen, bietet für jeden einen Mehrwert. Es ist bequem das Licht im Zimmer mit dem Touchscreen ein- und auszuschalten. Weiterhin können auch virtuelle Dimmer eingesetzt und Zeitsteuerung bedient werden. Auch eine automatische Beleuchtung, durch erkennen der Helligkeit im Raum, ist eine Funktion mit hohem Potenzial.

Eine weitere Anwendungsmöglichkeit besteht bei der Waschmaschine. Die Restzeit des Waschgangs kann auf dem Bildschirm angezeigt werden. Wenn der Waschgang beendet ist, wird der Träger mit einer Nachricht notifiziert.

Des Weiteren ist das Fernbedienen von allen Multimediasgeräten übers Handgelenk sehr praktisch. Es genügt eine Uhr und braucht somit keine verschiedenen proprietären Steuerungen. Dies wird heute bereits mit Smartphone Apps praktiziert. Durch die Sprachsteuerung können Personen mit einer Leseschwäche die Uhr verwenden ohne die Lesebrille aufzusuchen.

4.1.3. Sport

Heute werden Smartwatches hauptsächlich als Fitnesstracker verwendet (vgl. [9]. Das Praktische an den Uhren unter den Wearables ist, dass diese nicht nur für zum Sport treiben genutzt werden kann. Hier erhält der Endkunde ein Gerät für den Alltag und die Freizeit.

Im Sportbereich können mit den vorhandenen Sensoren viele verschiedene Werte ermittelt und analysiert werden. Mit den nötigen Voreinstellungen, wie Körpermasse, Schrittlänge, Alter und Geschlecht, ist es möglich, Bewegungsdaten genau aufzuzeichnen. Mit den Daten können für den Anwender interessante Informationen berechnet werden. Für Hobbysportler meist relevante Berechnungen sind Zeit, Schritte, Geschwindigkeit und Kalorienverbrauch. Erfahrene Sportler verbessern ihre Fähigkeiten, durch betrachten von Auswertungen der Körperbelastungen, z.B. Beschleunigung, Stärke, Drehmoment und weitere.

4.1.4. Ortsbezogen

Applikationen, welche umgebungsorientiert arbeiten, sind geeignete Kandidaten für Smartwatches. Durch die permanente Anzeige am Handgelenk, können schnell ändernde Daten dauerhaft im Auge behalten werden.

Ein Punkt ist Geofencing. Mit Geofencing werden automatische Aktionen eingeleitet. Diese geschehen, sobald bestimmte geografische Grenzen überschritten werden. Über eine Ortung des Gerätes kann ermittelt werden, ob sich dieses in einer Geofencing Zone befindet. Mit Smartwatches kann dies effektiv genutzt werden, da durch den immer sichtbaren Bildschirm, die ortsspezifischen Daten ohne Zeitverlust verfügbar gemacht werden.

Eine Geofencing Anwendung könnte unter anderem Sehenswürdigkeiten präsentieren. Eine App, welche für jede Sehenswürdigkeit eine Geofencing Zone errichtet oder kennt und jeweils die Auskunft des Objektes darstellt.

Ein Anwendungsfall dazu ist auch, das Smartphone zu überwachen. Die Uhr kann den Träger informieren, wenn sich die Geofencingzonen, welche sich beide errichten durch ihre Sende- und Empfangsleistung, nicht mehr überschneiden. Wobei hier die geografischen Daten irrelevant sind.

Die gleiche Methode bietet sich an, Personen mit Smartwatches in der Nähe zu scannen. Diese Funktion kann bei Partnervermittlungsapplikationen effektiv eingesetzt werden. Durch den vorhandenen Touchscreen können potentielle Datingpartner angezeigt und kontaktiert werden. In der schnelllebigen Welt sind sich schnell erschließende Kontakte sehr willkommen.

Die Indoornavigation ist ein Bedürfnis, welches mit Smartwatches nicht gelöst, jedoch eine Erweiterung sein kann. In Zusammenarbeit mit Beacons/Eddystones und/oder Access Points können die Standorte von Smartwatchträgern, im inneren von Räumen, ermittelt werden. Beacons und Eddystones sind Sender/Empfänger, die per Bluetooth Low Energy (BLE, Bluetooth 4.0 oder Bluetooth Smart) Informationen aussenden.

Grossfirmen könnten einen grossen Nutzen aus dieser Technologiekombination schöpfen. Die Mitarbeiter dürften ihren Standort preisgeben, damit diese auf einer digitalen Gebäudekarte angezeigt werden können. Es würde die Effizienz erhöhen. Es können Geofencing Zonen definiert werden, um die Zeiterfassung zu automatisieren. Beim eintreten, des geografisch definierten Bereichs, welcher zu Arbeitszone gehört, wird Arbeitszeit erfasst. Verlässt der Mitarbeiter diesen Teil des Gebäudes, wird die Arbeitszeiterfassung gestoppt.

Zusätzlich kann das Problem mit den Shared-Desk Arbeitsplätzen gelöst werden. Bei diesem Arbeitsplatzmodell haben die Mitarbeiter keinen festen Arbeitsplatz. Es muss jeden Tag ein neuer aufgesucht werden. Um keine unnötige Zeit beim Aufsuchen zu verlieren, kann sich der Angestellte, mit einer Smartwatch ausgestattet, anzeigen lassen ob in einem Bereich ein freier Platz ist. Und beim erreichen des Schreibtisches kann dieser sich mit der Watch anmelden und den Platz als besetzt markieren.

4.1.5. Authentifikation

Im Authentifikationssektor gibt es einige Bedürfnisse, welche mit Smartwatches gedeckt werden können.

Mit der Smartwatch Türen entriegeln, ist ein geeigneter Anwendungsfall. Heute benutzen die meisten Automobilhersteller ein Keyless System für ihre Fahrzeuge. Mit diesen Systemen hat der Fahrer ein Sender/Empfänger als Schlüssel, mit einem einzigartigen Zertifikat, welches sich mit jenem im Fahrzeug korreliert. Türen werden geöffnet und Motoren werden gestartet. Diese Funktionen könnte auch die Smartwatch übernehmen. Damit hätte der Fahrzeugbesitzer ein Utensil weniger mit sich zu tragen.

Die intelligente Uhr hat das Potenzial, Personalausweise zu ersetzen. Wie bereits im vorherigen Abschnitt - Ortsbezogen - erwähnt, kann es zur Zeiterfassung genutzt werden. Das heisst, Angestellte können sich damit auch ausweisen.

4.1.6. Finanztechnologie - FinTech

Die Möglichkeit zu haben, mit der Uhr Zahlungen zu autorisieren, ist ein Bedürfnis, welches erschaffen werden kann. Denn es scheint praktisch zu sein, einkaufen zu gehen ohne das Portemonnaie dabei zu haben. Es sind Lösungen vorhanden, welche mit Smartphones funktionieren (Abbildung 4.1 Apple Pay/Google Wallet/TWINT). Mit der Lancierung der Apple Watch, erreichte die erste Smartwatch mit einer Zahlfunktion (sie unterstützt Apple Pay) den Markt.



Abbildung 4.1.: Bekannte Zahlungslösungen für Smartphones: Apple Pay, Google Wallet und TWINT powered by PostFinance

<http://www.apple.com/apple-pay/> <https://wallet.google.com/> http://www.twint.ch/ueber_uns/medien/, 04.12.2015

4.2. Smartwatch Applikationen für siot.net

Die siot.net-Plattform bietet sich bestens als Kommunikationsschnittstelle für die Applikationen an, welche im vorherigen Abschnitt ermittelt wurden. Die meisten Bedürfnisse verlangen irgendeine Art von Kommunikation. Ob es nur übermitteln der Sensordaten ist oder die Abfrage eines Sicherheitstokens ist. Die siot.net-Plattform erlaubt sämtliche Informationen über ihre Schnittstelle auszutauschen. Smartphones und Smartwatches sind ideale „Things“ für IoT-Anwendungen. Mit der Anbindung an die siot.net-Plattform kann jede beliebige IoT-Applikation von den Daten dieser Geräte Nutzen erzielen.

4.2.1. siot.net Gateway Library

Um Verknüpfungen individueller Applikationen von Smartwatches oder Smartphones mit der Plattform zu ermöglichen, sollte es eine generische Bibliothek geben. Diese sollte eine einfache Schnittstelle implementieren, welche Applikation an die siot.net-Plattform anbindet. Dies erleichtert ein Smartphone oder eine Smartwatch als ein „Thing“ in das IoT zu integrieren. Die Daten welche gesendet werden sollen, hängt vom Nutzen der Applikation ab. Dies hat jeder Entwickler selber zu definieren. Somit soll das gesamte Potenzial eines Android Gerätes augenutzt werden können. Zusätzlich vereinfacht die automatische Erkennung aller Sensoren und Vorbereitung ihrer Konfigurationsdaten die Entwicklung von Apps, welche Messwerte ans siot.net kommunizieren wollen.

4.2.2. siot.net Sensorcenter

Jedes Android Gerät kann mit den eingebauten Sensoren hervorragend als Sensorstation dienen. Um die Vielzahl von Sensoren in Eigenregie zum siot.net zu manifestieren und Sensordaten preiszugeben, sollte es eine App geben, mit einer leicht verständlichen, grafischen Benutzeroberfläche.

4.2.3. siot.net Dashboard App

Um eine Auswertung und Darstellung von den Daten zu erhalten, gibt es bereits eine Dashboard Webapplikation (siot.io). Um diese Funktionen einem Smartphone, in einer kompakten Form, zur Verfügung zu stellen, eignet sich eine App. Diese App sollte von Vorteil durch den Benutzer konfigurierbar sein.

4.2.4. Herzfrequenzüberwachung

Personen, welche ihren Puls im Griff haben wollen, leichte gesundheitliche Probleme haben oder bei Unregelmäßigkeiten der Herzfrequenz jemanden alarmieren wollen, würden eine Herzfrequenzüberwachung begrüßen. Diese Anwendungsfälle können mit einer Smartwatch, mit Pulsmesser, und der siot.net-Plattform abgedeckt werden.

4.2.5. Steuerung von Modellen

Modellbau spielt eine grosse Rolle im Bereich des Internet of Things. Es gibt viele Forschungsprojekte, welche mit Hilfe von Drohnen durchgeführt werden. Darum ist die Kombination von Modellbau und Smartphones keine Weltneuheit. Es wird schon rege verwendet, wie bei der Parrot bebop Drohne (siehe Abbildung 4.2). Das Potenzial, mit einer Steuerung (Smartphone oder Smartwatch). Mehrere Modelle simultan zu steuern, scheint jedoch noch nicht im grossen Stile abgerufen zu werden. Mit der Einbindung von Modellen ins Internet der Dinge, zusätzlich gekoppelt mit einer smarten Steuerungseinheit, kann dieses Bedürfnis abgedeckt werden. Die siot.net Umgebung bietet dazu eine günstige Ausgangslage.



Abbildung 4.2.: Parrot bietet Drohnen an, welche mit dem Smartphone gesteuert werden können: Parrot bebop
https://s.gravis.de/p/z1/parrot-bebop-drone-kamera-drohne-fuer-smartphones-tablets-gps-blau_z1.jpg, 04.12.2015

5. Technische Anforderungen

In diesem Kapitel werden die Technischen Voraussetzungen, für die, in der Bedürfnisanalyse ermittelten Anwendungen, definiert. Dabei wird unterschieden zwischen Allgemeine Applikationen, bei welcher nur eine kleine Auswahl beachtet wird, und zwischen siot.net Applikationen.

5.1. Allgemeine Applikationen Anforderungen

Die Technischen Anforderungen für die Bedürfnisse Gesundheit, Smart Home und Finanztechnologie werden in diesem Abschnitt definiert.

5.1.1. Gesundheit

Bei den Gesundheitsapplikationen ist es wichtig, dass die Smartwatch über einen Herzfrequenzmesser verfügt und die Kombination aus Gyroskop, Rotationssensor und Bewegungssensor eingebaut ist. Diese Elemente sind erforderlich, um Pulsraten eines Menschen zu messen, sowie Analysen von Bewegungen zu erstellen.

Benötigte Sensoren:

- Gyroskop
- Bewegungssensor
- Rotationssensor
- Herzfrequenzmesser

5.1.2. Smart Home

Für die Steuerung von Haushaltsgeräten, Heizung oder Licht, wird vor allem eine Steuerungsanzeige benötigt. Diese ermöglicht die Apparate ein- und auszuschalten, die Beleuchtung zu dimmen und Benachrichtigungen (z.B. von offenem Fenster, beendetem Waschgang uvm.) anzuzeigen. Für eine automatische Helligkeitsregulierung sollte ein Lichtmesser eingebaut sein. Um Alarne ausgeben zu können, ist ein Lautsprecher oder Vibrationsmotor hilfreich.

Bedienelement:

- Touchscreen

Aktoren:

- Lautsprecher
- Vibrationsmotor

Benötigter Sensor:

- Lichtsensor

5.1.3. Finanztechnologie - FinTech

Um Geldtransaktionen durchführen zu können, wird eine drahtlose Verbindungseinheit vorausgesetzt. Bereits erwähnten Applikationen, wie Apple Pay und Google Wallet verwenden den NFC (Near Field Communication) Chip und TWINT verwendet BLE. Die Kommunikationsschnittstelle ist notwendig um Zahlungsanforderungen zu erhalten und diese auch zu autorisieren. Zum auswählen von Zahlungsoptionen und bestätigen von Überweisungen ist ein Bedienelement notwendig. Wenn mehrere Kreditkarten hinterlegt sind, muss der Anwender die Möglichkeit haben, eine bestimmte Kreditkarte für die Überweisung auszuwählen.

Auswahl von benötigten Kommunikationsschnittstellen:

- NFC Chip
- Bluetooth
- GSM/UMTS/LTE
- WLAN (weniger geeignet)

Mögliche Bedienelemente:

- Touchscreen (beste Voraussetzung)
- Mikrofon zur Sprachsteuerung
- Physische Taste

5.2. siot.net Applikationen Anforderungen

Technische Voraussetzung für die siot.net Applikationen werden für alle, in der Bedürfnisanalyse ermittelten Klassen definiert.

5.2.1. siot.net Gateway Library

Die Bibliothek, welche smarte Geräte an die siot.net-Plattform anbinden soll, wird in erster Linie nur für das Android Betriebssystem entwickelt. Für die Benutzung von Apps, welche ans siot.net angeschlossen werden, müssen diese mindestens die Android Tools der SDK Version 21 beherrschen (ab Android 5.0 Lollipop). Um die Daten an den MQTT Broker übermitteln zu können, braucht es ein Netzwerkmodul, dass eine Verbindung ins Internet erlaubt. Die Bibliothek sollte alle verfügbaren und bekannten Sensoren selber erkennen. Die Verwendung, ob Sensoren aktiviert werden oder nicht, wird durch den Appentwickler oder der Software selber definiert.

Auswahl von benötigten Kommunikationsschnittstellen:

- NFC Chip
- Bluetooth
- GSM/UMTS/LTE
- WLAN

Betriebssystem:

- ab Android 5.0
- ab SDK Tools Version 21 (Android Tools)

5.2.2. siot.net Sensorcenter

Mit der Sensorcenter Applikation von siot.net, soll jeder beliebige Benutzer die Sensormesswerte seines Android Gerätes ans siot.net senden können. Um dies zu ermöglichen muss eine Anmeldung ans siot.net, manifestieren von Sensoren, sowie senden von Messungen möglich sein. Diese App kann vom Bestehen der siot.net Gateway Library profitieren, welche integriert werden soll. Technisch benötigt diese Applikation, zusätzlich zu den Voraussetzungen der siot.net Gateway Library, ein Touchscreen.

Auswahl von benötigten Kommunikationsschnittstellen:

- NFC Chip
- Bluetooth
- GSM/UMTS/LTE
- WLAN

Bevorzugtes Bedienelement:

- Touchscreen

Betriebssystem:

- ab Android 5.0
- ab SDK Tools Version 21 (Android Tools)

5.2.3. siot.net Dashboard App

Um Sensordaten von der siot.net-Plattform darzustellen, eignet sich eine Dashboard App. Anforderungen für diese Applikation sind in dieser Arbeit nur für Android Geräte spezifiziert. Für die Darstellung einer derartigen digitalen Instrumententafel eignet sich ein Display, bevorzugt ein Touchscreen. Ein Berührbildschirm erlaubt es die gewünschten Anzeigen bequem einzublenden. Eine weitere Voraussetzung ist die Vernetzung. Das Gerät muss einen Zugang zum Internet herstellen können. Nur durch eine erfolgreiche Anmeldung an den siot.net MQTT Broker, können die Informationen empfangen werden.

Auswahl von benötigten Kommunikationsschnittstellen:

- NFC Chip
- Bluetooth
- GSM/UMTS/LTE
- WLAN

Bevorzugtes Bedienelement:

- Touchscreen

Betriebssystem:

- ab Android 5.0
- ab SDK Tools Version 21 (Android Tools)

5.2.4. Herzfrequenzüberwachung

Die Herzfrequenzüberwachung ist eine schlanke Variante des Dashboards. Um eine Anzeige, wie bei einem Elektrokardiogramm (EKG) darzustellen, braucht es ein Display. Und um Messdaten zu erhalten braucht es einen Puls- messer. Für einen Alarm auszulösen, beim Überschreitung von definierten Werten, braucht es einen Lautsprecher und/oder einen Vibrationsmotor, damit akustische oder taktile Signale ausgesendet werden können.

Auswahl von benötigten Kommunikationsschnittstellen:

- Bluetooth
- GSM/UMTS/LTE
- WLAN

Bevorzugtes Anzeigen / Aktoren:

- Touchscreen
- Lautsprecher
- Vibrationsmotor

Sensoren:

- Herzfrequenzmesser

Betriebssystem:

- ab Android 5.0
- ab SDK Tools Version 21 (Android Tools)

5.2.5. Steuerung von Modellen

Um die Kontrolle über ein Modellfahrzeug oder Modellflugzeug zu erhalten, benötigt es einen Hardwarekontroller, welcher sich mit dem siot.net MQTT Broker verbinden kann. Die Steuerungseinheit muss auf die MQTT Topic subscriben und die empfangenen Bewegungsdaten interpretieren und an die erforderlichen Antriebsmotoren und Servos weitergeben werden. Als Kontroller könnte ein Raspberry Pi Zero oder Ähnliches zum Tragen kommen. Auf den Hardwarekontroller wird nicht weiter eingegangen. Als Sender und Ermittler der Bewegungs- und Beschleunigungsdaten kommt ein Android Device (z.B. mit siot.net Sensorcenter) zum Einsatz. Damit muss die Spezifikation für die Steuerung nicht mehr weiter betrachtet werden, da diese mit dem Sensorcenter abgedeckt wird.

Auswahl von benötigten Kommunikationsschnittstellen:

- Bluetooth

- GSM/UMTS/LTE
- WLAN

Bevorzugtes Anzeigen / Aktoren:

- Touchscreen
- Lautsprecher

Sensoren:

- Bewegungssensor
- Beschleunigungssensor
- Gyroskop

Betriebssystem:

- ab Android 5.0
- ab SDK Tools Version 21 (Android Tools)

6. Technologiewahl

Im Kapitel der Technologiewahl werden einige aktuell erhältliche Smartwatch Modelle mit den theoretischen Daten verglichen, allgemeine Eigenschaften aufgeführt und die Entwicklerfreundlichkeit bewertet. Aufgrund der Bewertung von aufgestellten Kriterien werden für die Arbeit ein bis zwei Uhren ausgewählt.

6.1. Allgemeine Eigenschaften

Heute (Januar 2016) sind weitgehend alle erhältlichen tragbaren Computer am Handgelenk vom Smartphone abhängig (vgl. [15]). Deshalb bieten sie nur einen kleinen Mehrwert. Sie können zwar viel mehr als nur die Zeit anzeigen, jedoch sehr beschränkt und zum grössten Teil wird das Mobiltelefon benötigt. Nachrichten, Termine und einkommende Telefonate können sie zwar anzeigen, aber ein Gespräch kann nur mit den wenigsten Geräten geführt werden, wie z.B. der Apple Watch. Mit den meisten Smartwatches können Nachrichten gelesen, jedoch nicht beantwortet werden, da diese über keine virtuelle Tastatur verfügen. Falls doch, geschieht dies hauptsächlich über die Spracheingabe. Mit dem integrierten Touchscreenkeyboard, stellt die Samsung Gear S und S2, die einzige Ausnahme dar.

Smartwatches wollen die Gesundheit fördern. Etliche davon messen den Puls, zählen die Schritte und erfassen zurückgelegte Wegstrecke. Weitere Apps erlauben den Schlaf zu überwachen, bei langer Inaktivität zur Bewegung aufzufordern und den Kalorienbedarf und Verbrauch zu berechnen. Durch die genauen Daten haben Amateur-sportler eine praktische Hilfe am Handgelenk, um ihren Körper fit zu halten. Für ambitionierte Sportler ist der Funktionsumfang noch zu gering.

Manche Uhren der Hersteller Apple, LG, Samsung und Alcatel messen den Puls nahe zu Elektrokardiogramm-genau. Apple arbeitet bereits an einem Elektrokardiogramm (EKG) Armband, welches Daten per Ultraschall übermitteln soll. Das Band ermittelt die Herzströme über zwei angebrachte Elektroden. Das übermitteln der Messwerte als Ultraschall, soll den Stromverbrauch geringhalten. Dadurch wird der Apple Smartwatch der Einsatz als mobiles EKG ermöglicht.¹.

Ein grosse Schwachstelle von Smartwatches sind ihre leistungsschwachen Akkumulatoren. Bei geringer Nutzung erreichen die Uhren eine durchschnittliche Laufzeit von zirka 24 Stunden.

Applikationen, welche einen grossen Mehrwert bringen, sind selten. Zum heutigen Zeitpunkt stellt sich heraus, dass die Minicomputeruhren überwiegend als Erweiterungsbildschirm des Smartphones dienen. Wenige Geräte werkeln autonom, wie zum Beispiel die Samsung. Jedoch bleiben die meisten ein verlängerter Arm des dazugehörigen Mobiltelefons. Dies liegt daran, weil die smarten Uhren noch Heute in den Kinderschuhen stecken.

¹vgl. <http://www.smartwatch.de/news/neues-apple-watch-armband-mit-ultraschall-ekg-geplant>, 20.11.2015

6.2. Kandidaten

Smartwatch	Apple Watch [15]	LG G Watch	LG Watch Urbane [15]
Betriebssystem	WatchOS	Android Wear	Android Wear
Funktionen	gut	befriedigend	befriedigend
- Nachrichten	empfangen möglich, senden per Sprachnachricht, Emojis senden	empfangen möglich, senden per Sprachnachricht	empfangen möglich, senden per Sprachnachricht
- Telefonieren	führen von Gesprächen	nur Notifikation	nur Notifikation
- Display	42mm 312x390px 38mm 272x340px	1.65 Zoll 280x280px	1.3 Zoll 320x320px
∅ Akkulaufzeit	19h	30h	20h
Datensendungsverhalten	verschlüsselt	verschlüsselt	verschlüsselt
Smartphone-Betriebssystem	ab iOS 8.2	ab Android 4.3 ab iOS 8.2	ab Android 4.3 ab iOS 8.2
Pulsmesser	optischer Pulsmesser	kein Pulsmesser	optischer Pulsmesser
Gesamteindruck	bestes Display viele Funktionen befriedigend - gut	Always-On Display befriedigender Akku befriedigend	rundes Display edle Verarbeitung befriedigend

Tabelle 6.1.: Technische Daten von Smartwatches - Teil 1

			
Smartwatch	Motorola Moto 360 [15]	Samsung Gear S2 [8]	Alcatel Onetouch Watch [15]
Betriebssystem	Android Wear	Tizen	proprietäres Alcatel OS
Funktionen	befriedigend	gut	befriedigend
- Nachrichten	empfangen möglich, senden per Sprachnachricht	empfangen möglich, senden möglich	empfängt nur Kurznachrichten
- Telefonieren	nur Notifikation	nur Notifikation bei eSIM Variante möglich	nur Notifikation
- Display	1.57 Zoll 360x290px	1.2 Zoll 360x360px	1.3 Zoll 320x320px
∅ Akkulaufzeit	10h	38h	20h
Datensendungsverhalten	verschlüsselt	verschlüsselt	unverschlüsselt
Smartphone-Betriebssystem	ab iOS 8.2	ab Android 4.4	ab Android 4.3 ab iOS 7
Pulsmesser	optischer Pulsmesser	optischer Pulsmesser	optischer Pulsmesser
Gesamteindruck	bestes Display meiste Funktionen befriedigend	Intuitive Lünette gute Akkuleistung befriedigend - gut	schwaches Display mangelhafte Datenübertragung mangelhaft

Tabelle 6.2.: Technische Daten von Smartwatches - Teil 2

6.3. Entscheid Smartwatch

Wenn nur der Funktionsumfang betrachtet wird, wäre eine Apple Watch oder die Samsung Gear S2 die bevorzugte Wahl gewesen. Die Samsung ist erst seit Oktober 2015 erhältlich, folglich fiel diese, Aufgrund des späten Erscheinungsdatum und dessen Samsung-proprietären Betriebssystem Tizen, aus dem Katalog.

Die Apple Watch wurde nicht gewählt, weil die Apple-Plattform kein offenes System ist. Ein weiterer Entscheidungspunkt ist, dass der Markt de facto aus einem Produkt besteht. Es sei denn, man unterscheidet die Modellvarianten Watch und Watch Sport sowie die Größen, welche sich nur optisch voneinander abheben.

Ausgewählt wurden zwei Produkte, welche nicht sehr aufgefallen sind. Obwohl sie nicht besonders spektakulär sind, leisten sie genug, um den meisten Bedürfnissen, aus der vorhergehenden Analysen, zu genügen: Die **LG G Watch** und die **Motorola Moto 360**.

Als die erste auf dem Markt erhältliche Android Wear Smartwatch, wurde die LG G Watch ausgewählt. Der eingebaute Snapdragon 400 Prozessor rechnet mit 1,2 GHz, hat 512MB RAM und hat einen rechteckigen Bildschirm. Ein Pluspunkt der LG Datenuhr: Sie kann über USB gedebuggt werden. Dies erhöht die Effizienz bei der Entwicklung von Apps.

Um ein Android Wear Vergleichsobjekt zu erhalten, kam die neuere Motorola Moto 360 zusätzlich in die Entscheidung. Diese arbeitet mit einem Texas Instrument OMAP 3 Prozessor mit 1 GHz, hat ebenfalls 512MB RAM und hat einen runden Touchscreen. Mehrwert der Moto 360 zur G Watch ist, ein eingebautes WLAN-Modul, Lichtsensor, Pulssensor und eine physische Taste. Bedauerlicherweise kann sie nur über Bluetooth gedebuggt werden, was viel Zeit beansprucht, um entwickelte Apps zu übertragen.

Ein wichtiger Unterschied beider Geräte ist der Bildschirmmodus. Die G Watch erlaubt den Always-On Betrieb, was einer App erlaubt das Display während der Laufzeit immer eingeschaltet zu lassen. Die Moto 360 hingegen hat

den Ambient Mode, welcher den Bildschirm abdunkelt, die App jedoch am laufen hält. Hierbei ist zu erwähnen, dass Android Wear Apps sich schliessen, wenn der Bildschirm in den Ruhezustand geht.

Durch die Entscheidung für Android Wear liegt der Schwerpunkt der Entwicklung auf Java sowie das Layouting auf XML. Die Entwicklungsumgebung konnte auch auf Android Studio eingeschränkt werden, da im Sommer 2015 bekannt gegeben wurde, dass die Entwicklung des Eclipse Plugin eingestellt wird (vgl. [4]). Eine alternative Entwicklungsumgebung, welche die Effizienz eines Android Studios hat oder bedeutende Vorteile hat, ist keine aufgefallen.

7. Architektur

7.1. siot.net Architektur

Die Gesamtarchitektur von siot.net mit dem Zusammenspiel mit Android und Android Wear ist auf der Abbildung 7.1 zu betrachten.

Im Zentrum steht das siot.net IoT-Center, welches zwingend mit dem siot-Interface (vgl. [13]) angesprochen werden muss.

Darunter sind die Sensoren und Aktoren. Diese sind hierarchisch aufgezeigt. Das heisst ein Sensor oder Aktor, von einem Android Wear Gerät, muss die darüber liegenden Stationen durchqueren, um sich am siot.net zu manifestieren. Aus dieser Architektur ist ersichtlich, dass Android Wear Komponenten nicht direkt über ein siot.net Gateway kommunizieren. Dieser Aspekt wird im Abschnitt Netzwerk-Architektur genauer betrachtet.

Am oberen Teil sind die Applikationen eingezeichnet. Diese ziehen nutzten von den Sensoren und Aktoren, welche im siot.net manifestiert sind und Daten senden oder empfangen.

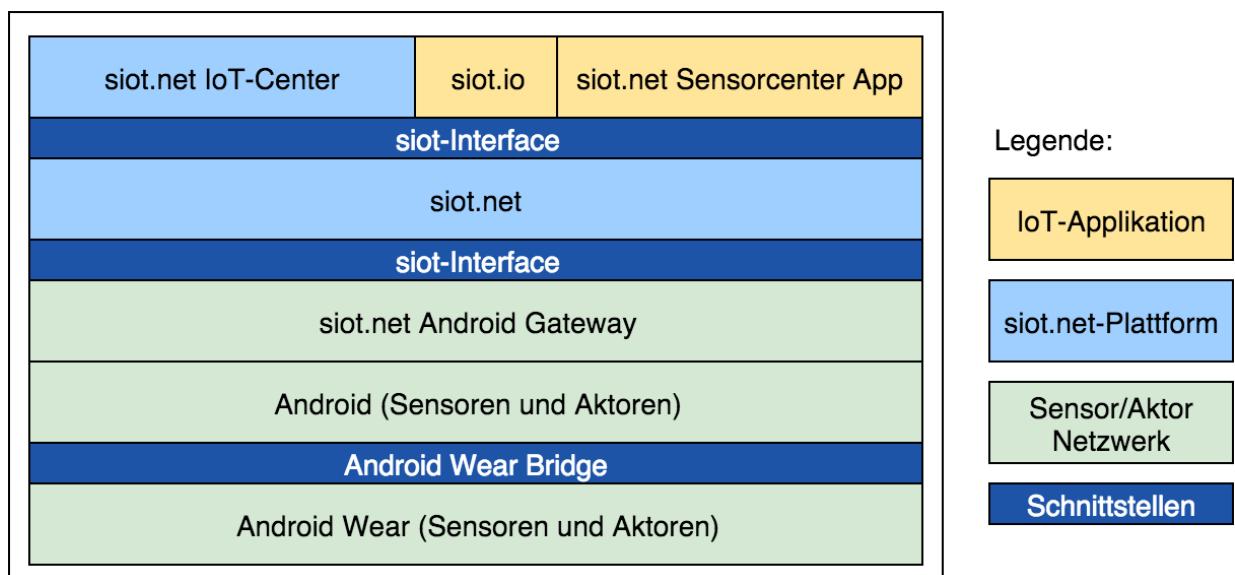


Abbildung 7.1.: Die Gesamtarchitektur von siot.net im Zusammenspiel mit der Android Welt

7.2. Android Wear Paket Architektur

Android Wear hat eine strikte Paket Architektur. Abbildung 7.2 veranschaulicht die Übersicht dieser Architektur. Eine Android Wear App muss immer zwingend eine dazugehörige Android App beinhalten. Wenn beide Apps gemeinsam verwendete Klassen haben, müssen diese in ein solches Paket ausgelagert werden. Die in der Android Mobile Paket entwickelten Klassen sollten nicht in einer Android Wear Applikation instanziert werden. Üblicherweise wird bei der Installation der .apk auf ein Android Smartphone, parallel die Smartwatch App auf die Uhr mit installiert.

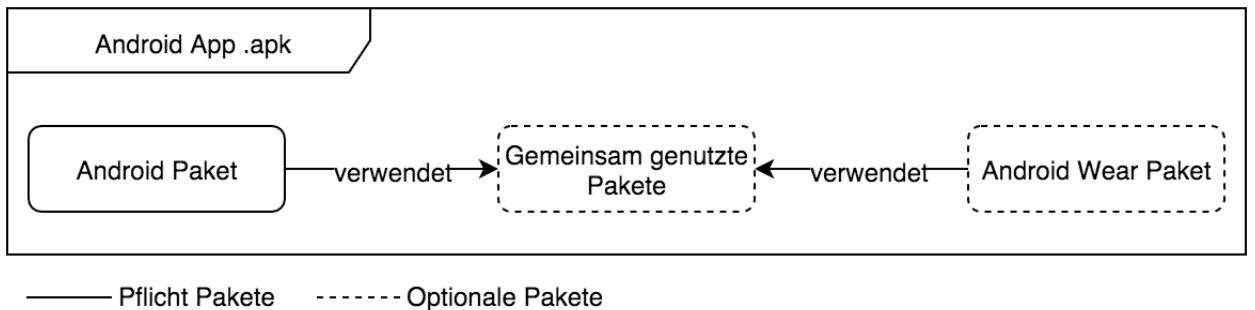


Abbildung 7.2.: Übersicht über die einzuhaltende Paketstruktur

7.3. siot.net Android Applikationsarchitektur

Die Abbildung 7.3 zeigt die Softwarearchitektur, welche für Android und Android Wear Applikationen im siot.net Umfeld spezifiziert wurde. Mobile und Wearable Apps für siot.net können so am effizientesten konzipiert werden.

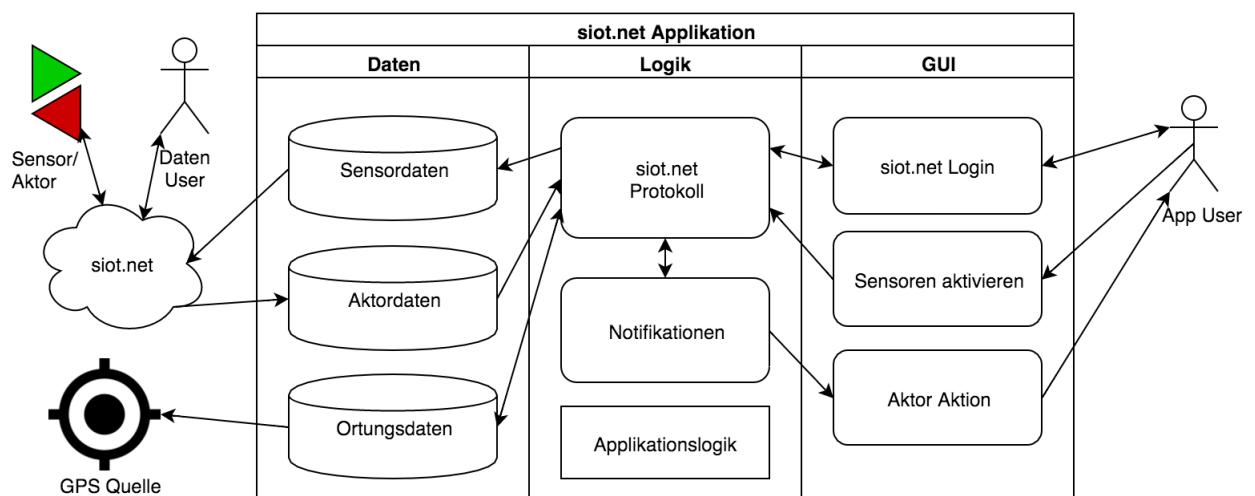


Abbildung 7.3.: Applikationsarchitektur für siot.net Anwendungen

7.4. Netzwerk-Architektur

7.4.1. geplante Architektur

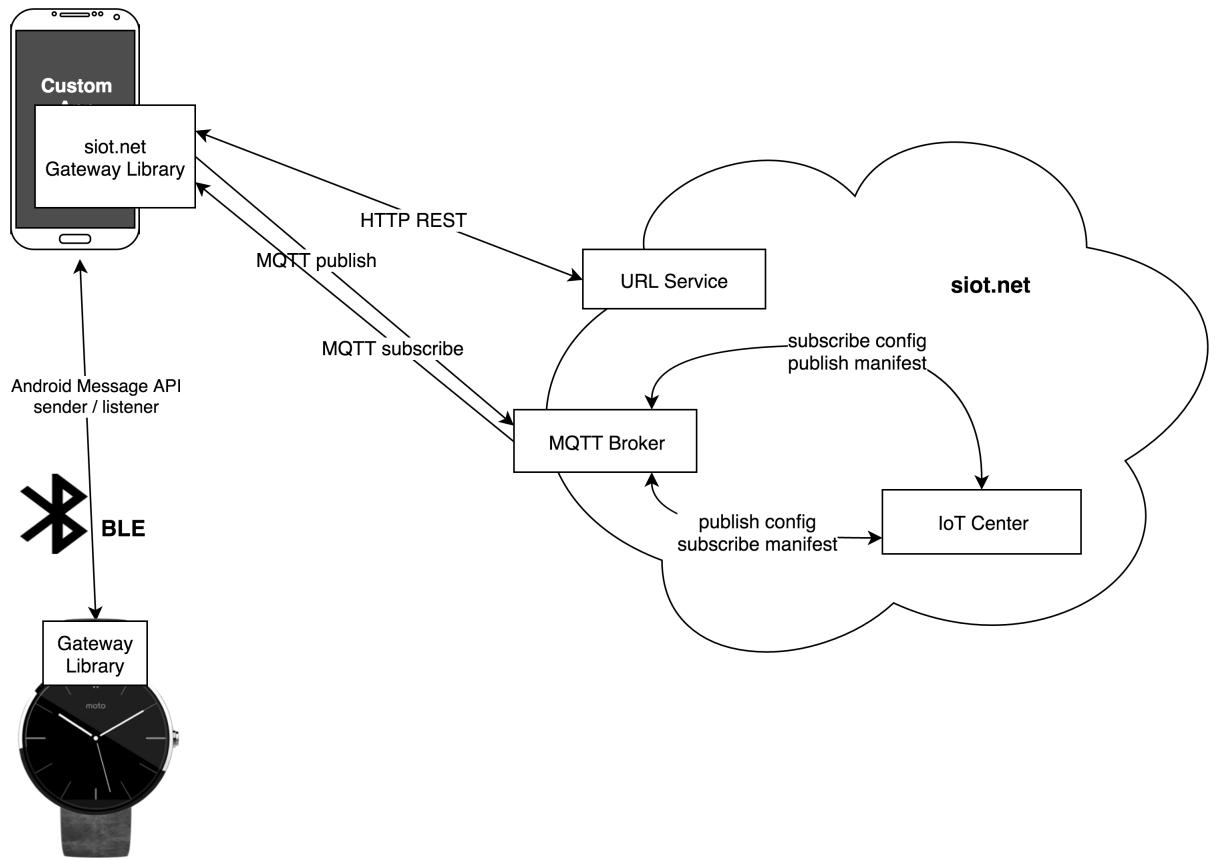


Abbildung 7.4.: Geplante Kommunikation zwischen Smartwatch und Smartphone, sowie Smartphone und siot.net

Die Abbildung 7.4 visualisiert, wie die Kommunikation zwischen der Smartwatch, dem Smartphone und der siot.net-Plattform stattfinden soll. Im Falle, dass die Smartwatch eine Bluetooth Low Energy (BLE) Verbindung mit dem Smartphone aufweist, wird das Smartphone via REST und MQTT mit der siot.net-Plattform Daten austauschen. Dabei dient das Mobiltelefon als das Gateway von der Smartwatch zur IoT-Cloud. Die Android Wear Uhr versendet die Daten, per Android MessageAPI via BLE zum Android Smartphone. Dies leitet die Pakete mit Hilfe des MQTT Protokoll weiter ans siot.net. Diese Variante der Datenübermittlung ist sehr viel stromsparender, als jene die in Abbildung 7.5 aufgezeigt wird.

Das Bridging muss in der Smartphone Begleit-App (Android Paket) der Smartwatch Applikation (Android Wear Paket) implementiert sein.

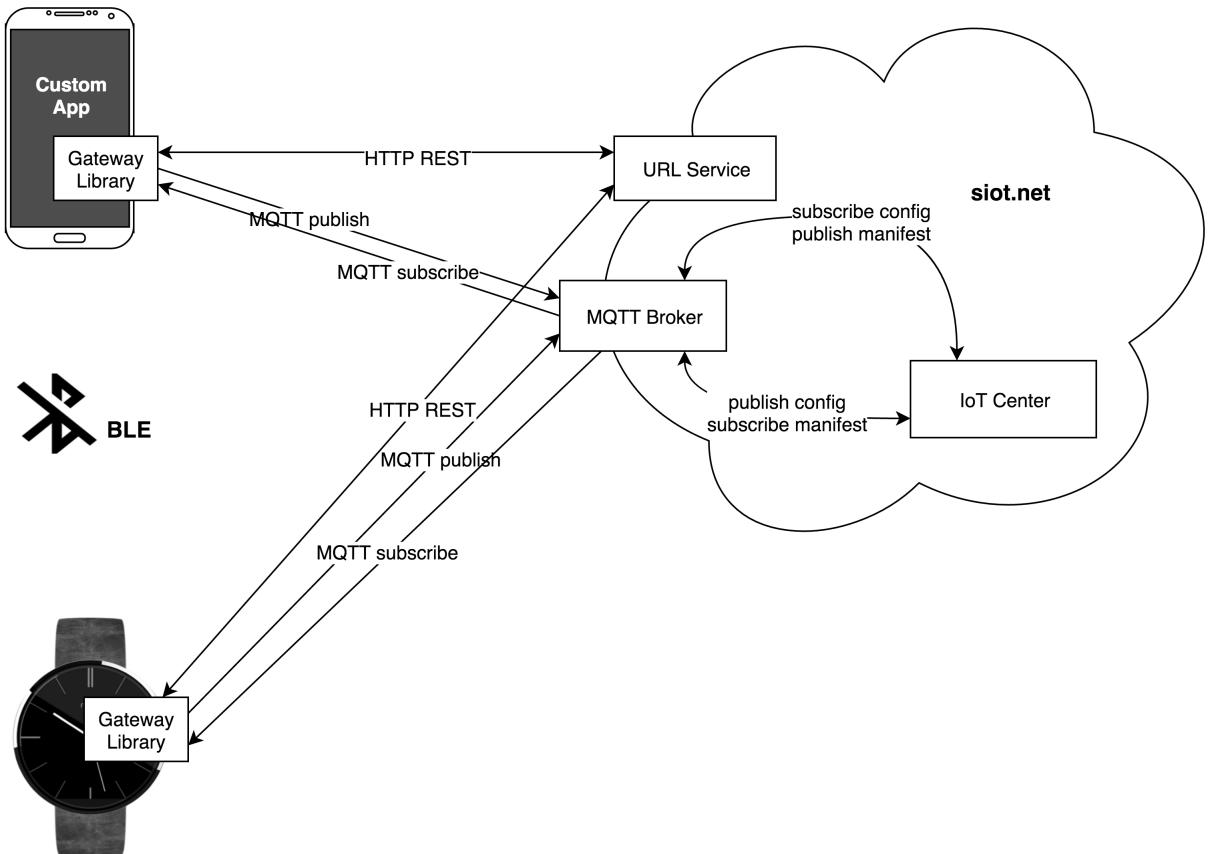


Abbildung 7.5.: Geplante Kommunikation zwischen Smartwatch und siot.net, sowie Smartphone und siot.net

Bei Abbruch der Verbindung von Smartwatch zu Smartphone, soll die Computeruhr die Kommunikation zum siot.net selber übernehmen. Dazu muss zuerst über REST die URL des MQTT Broker ermittelt werden, um folglich eine Kopplung durchzuführen. Diese Architektur ist auf Abbildung 7.5 zu betrachten. Weil die Android Smartwatches keine direkte Internetverbindung zulassen, kann diese gewünschte Architektur noch nicht umgesetzt werden. Wenn diese Beschränkung durch das Betriebssystem aufgelöst wird, wäre die Architektur bereits spezifiziert.

7.4.2. effektive Architektur

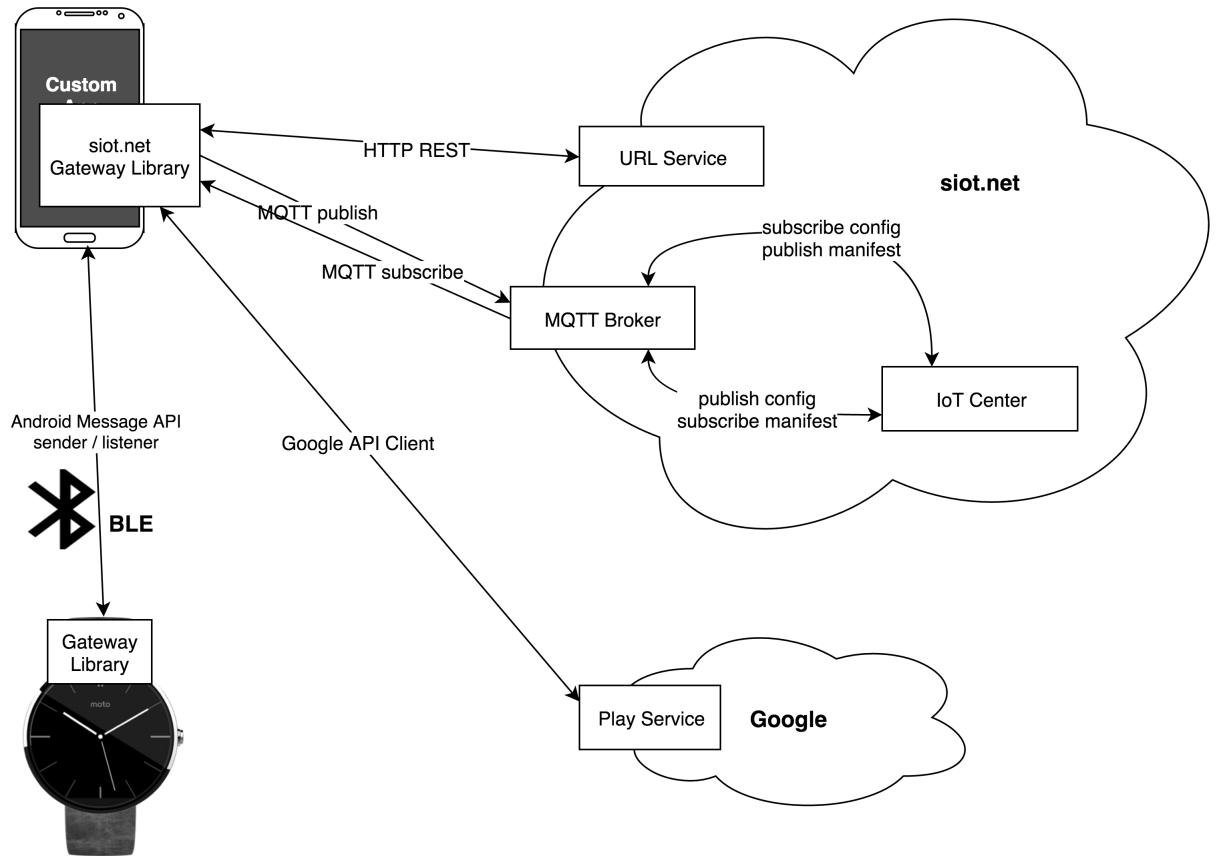


Abbildung 7.6.: Effektive Kommunikation zwischen Smartwatch und siot.net, sowie Smartphone und siot.net

Diese Architektur (siehe Abbildung 7.6) unterscheidet sich geringfügig zu jener auf Abbildung 7.4. Hier kommt nun die Google Cloud dazu. Diese Anbindung ist notwendig, da Android für die Kommunikation über die Android MessageAPI eine eindeutige Identität verlangt. Es muss vorhergehend eine Nodeld über den Google Play Service eingefordert werden. Dank dieser Nodeld dürfen die Geräte (Smartphone und Smartwatch benötigen jeweils eine eindeutige Identifikation) mittels MessageAPI Nachrichten transferieren. Der Datenaustausch geschieht dann direkt via BLE. Diese Netzwerk-Architektur kommt zum tragen, wenn die Kopplung von Smartwatch und Smartphone über Bluetooth Smart steht.

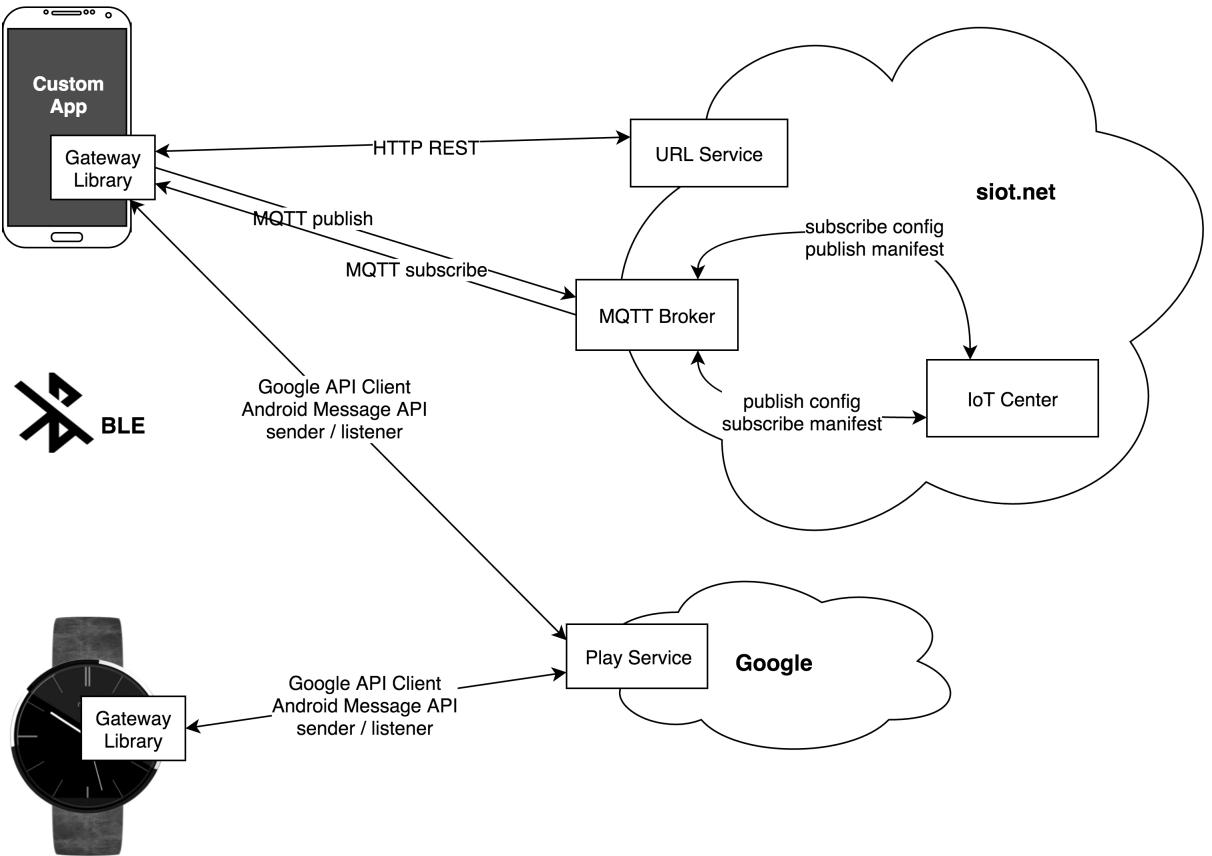


Abbildung 7.7.: Effektive Kommunikation zwischen Smartwatch und siot.net, sowie Smartphone und siot.net

Abbildung 7.7 zeigt die funktionierende Architektur, wenn keine intakte BLE Verbindung vorhanden ist. Diese differenziert sich wesentlich von der geplanten Netzwerk-Definition. Auch hier spielt der Faktor Android eine grosse Rolle. Es verbietet Android Wear Geräten den direkten Kontakt zum Internet. So ist hier eine Verbindung über den Google Play Service zum Smartphone nötig. Bei dieser Anbindungsart werden die Daten via Google Cloud an das Android Handy gesendet. Dieses muss dann in jedem Fall als siot.net Gateway fungieren.

Der grosse Nachteil dieser Variante ist, dass die Smartwatch nicht als autonomes Gerät fungieren kann. Wenn das Smartphone nicht erreichbar ist, z.B. Akku leer oder keine Netzwerkverbindung, ist die Computeruhr für siot.net nutzlos.

8. Anforderungsdokumente

Dieser Teil des Dokuments beinhaltet drei Anforderungsdokumente von der vorhergehend ermittelten Anwendungen. Requirements von der siot.net Gateway Library, siot.net Sensorcenter und siot.net Dashboard App sind aufgeführt.

8.1. Requirements - siot.net Gateway Library

In diesem Requirementsdokument werden die Anforderung an die siot.net Gateway Library (**sGWLib**) spezifiziert.

8.1.1. Allgemeine Beschreibung

Die Anforderung an die siot.net Gateway Library ist, die Entwicklung von Android Apps zu erleichtern, die ans siot.net verbunden werden sollen. Programmieren von Applikationen muss immer einfacher und schneller werden, da Projekte immer mehr an höherem Zeitdruck leiden. Diese Bibliothek soll helfen, Integrationen von Apps an das siot.net ohne grosses Vorwissen, schnell durchzuführen. Die Library übernimmt die Verwaltung von den Sensoren der Geräte, sowie die Kommunikation von Mobile Device und Smartwatch zu siot.net.

8.1.2. User Requirements

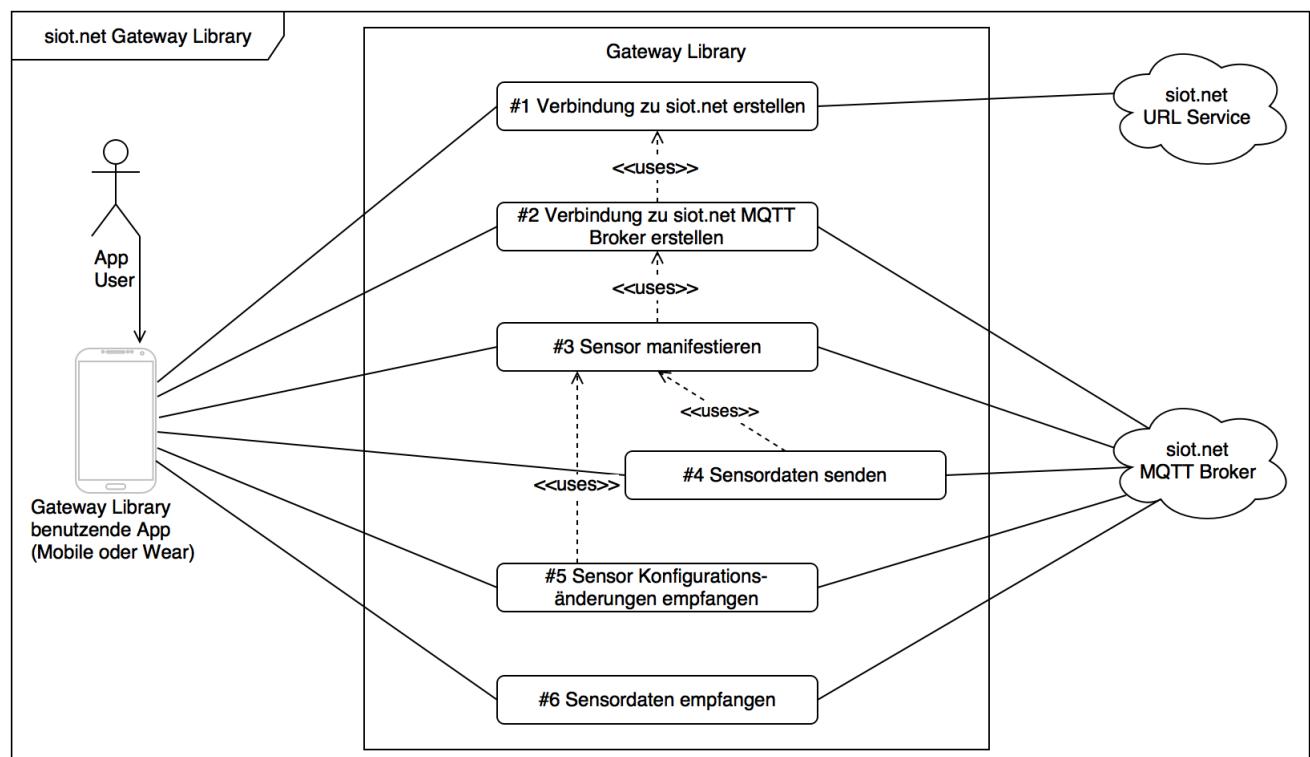


Abbildung 8.1.: Das allgemeine Use Case Diagramm für die siot.net Gateway Library

8.1.3. Funktionale Anforderungen und Anwendungsfallbeschreibungen

8.1.3.1. Anwendungsfall #1

Nr. und Name:	#1 Verbindung zu siot.net erstellen.
Szenario:	Android Gerät wird mit dem siot.net verbunden.
Kurzbeschreibung:	Die Verbindung des Android Devices zum siot.net wird hergestellt. Dies geschieht über den URL Service, um die nötigen URLs zu erhalten (IoT-Center und MQTT Broker).
Beteiligt Akteure:	Android App (User), siot.net Gateway Library, siot.net URL Service, siot.net MQTT Broker.
Auslöser und Vorbedingung:	User gibt über seine App den Befehl, die Applikation zu seiner IoT-Cloud zu verbinden. Wenn siot.net als Ziel gewählt wird, muss die siot.net Gateway Library eingebunden sein. Für eine erfolgreiche Verbindung muss eine gültige siot.net Lizenz vorhanden sein.
Ergebnisse und Nachbedingung:	Der URL Service von siot.net kann die Lizenz richtig verifizieren und liefert die benötigten URLs.

Tabelle 8.1.: siot.net Gateway Library: Übersicht Anwendungsfall #1

Ablauf:

1-1	App User	Startet die App.
1-2	App User	Gibt siot.net Lizenz ein und klickt „Verbinden zu siot.net“.
1-3	App	Fordert bei der siot.net Gateway Library (sGWLib) die Verbindung an.
1-4	sGWLib	Verfiziert sich beim siot.net URL Service mit dem Lizenzschlüssel.
1-5	sGWLib	Erhält URL vom siot.net MQTT Broker und verbindet sich (Anwendungsfall #2).
1-6	sGWLib	App über Verbindungsstatus benachrichtigen.
1-7	App	Benachrichtigung verarbeiten, z.B. Meldung an App User.

Tabelle 8.2.: siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #1

Ausnahmen und Varianten:

1-4.1	sGWLib	Ausnahme: siot.net URL Service nicht erreichbar, App benachrichtigen.
1-4.2	sGWLib	Ausnahme: Lizenzschlüssel nicht valide, App benachrichtigen.
1-5.1	sGWLib	Ausnahme: MQTT Broker nicht erreichbar, App benachrichtigen.

Tabelle 8.3.: siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #1

8.1.3.2. Anwendungsfall #2

Nr. und Name:	#2 Verbindung zu siot.net MQTT Broker erstellen.
Szenario:	Verbindung zum siot.net MQTT Broker herstellen.
Kurzbeschreibung:	Durch die vom URL Service erhaltene MQTT URL wird das Android Gerät wird mit dem Broker gekoppelt.
Beteiligt Akteure:	siot.net Gateway Library, siot.net URL Service, siot.net MQTT Broker.
Auslöser und Vorbedingung:	Durch das erhalten einer gültigen MQTT Broker URL wird der Verbindungsversuch gestartet.
Ergebnisse und Nachbedingung:	Es kann erfolgreich eine MQTT Client Verbindung hergestellt werden.

Tabelle 8.4.: siot.net Gateway Library: Übersicht Anwendungsfall #2

Ablauf:

2-1	sGWLib	Empfang der URLs vom siot.net URL Service.
2-2	sGWLib	MQTT URL wird verwendet, um eine Verbindung zum Broker aufzubauen.
2-3	sGWLib	Verbindungsstatus wird überprüft und stellt die Verbindungsinstanz der App zur Verfügung.

Tabelle 8.5.: siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #2

Ausnahmen und Varianten:

2-2.1	sGWLib	Ausnahme: Authentifizierung fehlgeschlagen - Lizenzschlüssel ungültig, App benachrichtigen.
2-2.2	sGWLib	Ausnahme: Keine MQTT Broker URL erhalten, Information an App senden.
2-2.3	sGWLib	Ausnahme: MQTT Broker nicht erreichbar, App benachrichtigen.
2-2.4	sGWLib	Variante: Mehrere MQTT Broker URLs erhalten, absteigend der Reihe nacheinander Verbindungsversuche starten. Bei erstem Erfolg diesen Client verwenden und der App zur Verfügung stellen.

Tabelle 8.6.: siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #2

8.1.3.3. Anwendungsfall #3

Nr. und Name:	#3 Sensor manifestieren.
Szenario:	Manifestieren vom gewünschten Sensor beim siot.net
Kurzbeschreibung:	Jeder Sensor, welcher an die siot.net-Plattform angebunden wird, muss sich zuerst manifestieren. Dies geschieht durch eine MQTT Meldung des Datentyps Manifest (definiert durch siot.net), an den siot.net Broker.
Beteiligt Akteure:	siot.net Gateway Library, siot.net MQTT Broker.
Auslöser und Vorbedingung:	Wenn die App zum ersten Mal eine App ans siot.net anmelden will und dieser nicht dem IoT-Center nicht bekannt ist.
Ergebnisse und Nachbedingung:	Der Sensor kann erfolgreich am IoT-Center von siot.net manifestiert und danach verwaltet werden.

Tabelle 8.7.: siot.net Gateway Library: Übersicht Anwendungsfall #3

Ablauf:

3-1	App	App meldet einen Sensor zur Anmeldung am siot.net dem siot.net Gateway Library.
3-2	sGWLib	Sensor wird instanziert, eindeutige GUID wird generiert und Manifest wird erstellt.
3-3	sGWLib	Manifest wird an siot.net MQTT Broker gesendet.
3-4	sGWLib	Gateway Library abonniert die Konfigurationstopic des manifestierten Sensors beim MQTT Broker.
3-5	sGWLib	Sensor ist nun für die App bereitgestellt.

Tabelle 8.8.: siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #3

Ausnahmen und Varianten:

3-3.1	sGWLib	Ausnahme: MQTT Verbindung nicht verfügbar, App informieren.
3-2.1	sGWLib	Variante: Gemeldete Sensor verwaltet mehrere physische Werte. Pro Angabe wird ein Sensorsmanifest angelegt.
3-2.2	sGWLib	Variante: Sensor ist dem siot.net IoT-Center bereits bekannt. Es wird kein neuer Sensor angelegt, sondern als bereits verfügbaren registriert. Keine Aktion notwendig.

Tabelle 8.9.: siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #3

8.1.3.4. Anwendungsfall #4

Nr. und Name:	#4 Sensordaten senden.
Szenario:	Sensorwerte an den siot.net MQTT Broker senden.
Kurzbeschreibung:	Sensordaten werden vom Datentyp Sensordaten (spezifiziert von siot.net) an den Broker per MQTT gesendet.
Beteiligt Akteure:	Android System, siot.net Gateway Library, siot.net MQTT Broker.
Auslöser und Vorbedingung:	Sensor meldet, dass neue Daten verfügbar sind.
Ergebnisse und Nachbedingung:	Werte können mit Erfolg an den MQTT Broker publiziert werden.

Tabelle 8.10.: siot.net Gateway Library: Übersicht Anwendungsfall #4

Ablauf:

4-1	Android	Sensor meldet an, dass neue Werte ermittelt wurden.
4-2	sGWLib	Sensordaten Datentyp wird angefertigt.
4-3	sGWLib	Sensordaten Meldung wird an siot.net MQTT Broker gesendet.

Tabelle 8.11.: siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #4

Ausnahmen und Varianten:

4-3.1	sGWLib	Ausnahme: MQTT Verbindung nicht verfügbar, App informieren.
--------------	--------	---

Tabelle 8.12.: siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #4

8.1.3.5. Anwendungsfall #5

Nr. und Name:	#5 Sensor Konfigurationsänderung empfangen.
Szenario:	Konfigurationsänderungen im siot.net IoT-Center empfangen und verarbeiten.
Kurzbeschreibung:	Sensoren werden im siot.net IoT-Center verwaltet und konfiguriert. Bei der Änderung einer Einstellung wird der Sensor, per MQTT Message, notifiziert. Die Gateway Library empfängt die Nachricht und stellt den Sensor, den Werten entsprechend, ein.
Beteiligt Akteure:	siot.net Gateway Library, siot.net MQTT Broker.
Auslöser und Vorbedingung:	Konfigurationsnachricht von siot.net MQTT Broker erhalten.
Ergebnisse und Nachbedingung:	Sensor kann korrekt umkonfiguriert werden.

Tabelle 8.13.: siot.net Gateway Library: Übersicht Anwendungsfall #5

Ablauf:

5-1	sGWLib	Erhält Message mit neuen Sensoreinstellungen.
5-2	sGWLib	Konfiguration wird geparsed und einem Sensor zugeordnet.
5-3	sGWLib	Sensor wird umkonfiguriert.

Tabelle 8.14.: siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #5

Ausnahmen und Varianten:

5-3.1	sGWLib	Variante: Konfiguration ist einem physischen Werte eines Sensors zugeordnet, welcher mehrere verwaltet. Gesamter Sensor neu einstellen. Weiteres Verhalten muss in einem neuen Projekt genauer definiert werden (Sensorkombinationen sind im siot.net vorgesehen, jedoch noch nicht spezifiziert und freigegeben).
--------------	--------	--

Tabelle 8.15.: siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #5

8.1.3.6. Anwendungsfall #6

Nr. und Name:	#6 Sensordaten empfangen.
Szenario:	Sensordaten von anderen Sensoren im siot.net empfangen und bereitstellen.
Kurzbeschreibung:	Die siot.net Gateway Library kann auch Sensordaten empfangen, welche beim siot.net IoT-Center angemeldet sind. Sensordaten können beim MQTT Broker subskribiert werden. Beim Empfang einer Informationsnachricht, wird diese interpretiert und der App zur Verwendung gestellt.
Beteiligt Akteure:	siot.net Gateway Library, siot.net MQTT Broker.
Auslöser und Vorbedingung:	Sensordaten Message von siot.net MQTT Broker erhalten.
Ergebnisse und Nachbedingung:	Informationen wurden interpretiert und verfügbar gemacht.

Tabelle 8.16.: siot.net Gateway Library: Übersicht Anwendungsfall #6

Ablauf:

6-1	sGWLib	Erhält Message mit Sensordaten.
6-2	sGWLib	Infomationen interpretieren.
6-3	sGWLib	Datenzugriff der App erlauben.

Tabelle 8.17.: siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #6

Ausnahmen und Varianten:

6-2.1	sGWLib	Ausnahme: Empfangene Nachricht kann nicht als Sensordaten interpretiert werden. Information wird verworfen.
--------------	--------	---

Tabelle 8.18.: siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #6

8.1.4. Nicht-funktionale Anforderungen

Produktanforderungen:

Benutzbarkeitsanforderungen	Die Bibliothek muss eine Java Library sein, um diese möglichst einfach in ein Android Projekt integrieren zu können.
Effizienzanforderungen	Ein übersichtliche und schlanke Dokumentation (z.B. JavaDoc oder Entwicklerhandbuch) soll dem Entwickler helfen, die Bibliothek in App zu integrieren.
Zuverlässigkeitssanforderungen	Die Daten, welche transferiert werden, verwenden das fire-and-forget Servicedesignmuster. Bei diesem Muster werden die gesendeten Nachrichten nicht mehr verfolgt. Wenn eine Message nicht ankommt, wird dies von keinem System bemerkt. Da Daten nur durch die vom Benutzer gewünschten Sensoren versendet werden, vereinfacht dieses Pattern die Kommunikation zwischen Smartdevice und siot.net. Der Sicherheitsaspekt und die Übertragungsverifizierung wird in einem späteren Release verfolgt. Technisch sollte die Applikation, die vorhandene Hardware nicht überfordern.
Portierbarkeitsanforderungen	Die Informationen werden in einem Format übertragen und empfangen, wie sie von siot.net spezifiziert sind.
Performanceanforderung	An die Performance sind keine spezifischen Anforderungen gestellt. Daten sollten in nützlicher Frist gesendet werden (z.B. <5 Sekunden Verzögerung pro Nachricht)

Tabelle 8.19.: siot.net Gateway Library: Nicht-funktionale Produktanforderungen

8.2. Requirements - siot.net Sensorcenter

In diesem Requirementsdokument werden die Anforderung an die Android App siot.net Sensorcenter Android (**sSC-A**) und siot.net Sensorcenter Android Wear (**sSC-AW**) spezifiziert.

8.2.1. Allgemeine Beschreibung

Das siot.net Sensorcenter für Android Geräte, ist die erste Applikation, welche die siot.net Gateway Library (sGWLib) integrieren soll. Ein Android Device eignet sich sehr gut als eigenes Sensorcenter, weil die mobilen Telefone und Uhren eine grosse Anzahl von Sensoren eingebaut haben. Diese App bietet die Möglichkeit, alle vom User gewünschten Sensordaten, dem siot.net preiszugeben. Der Benutzer kann die Sensoren selber aktivieren und deaktivieren. Die gemessenen Daten werden via MQTT, mit Hilfe der siot.net Gateway Library, übermittelt.

8.2.2. User Requirements

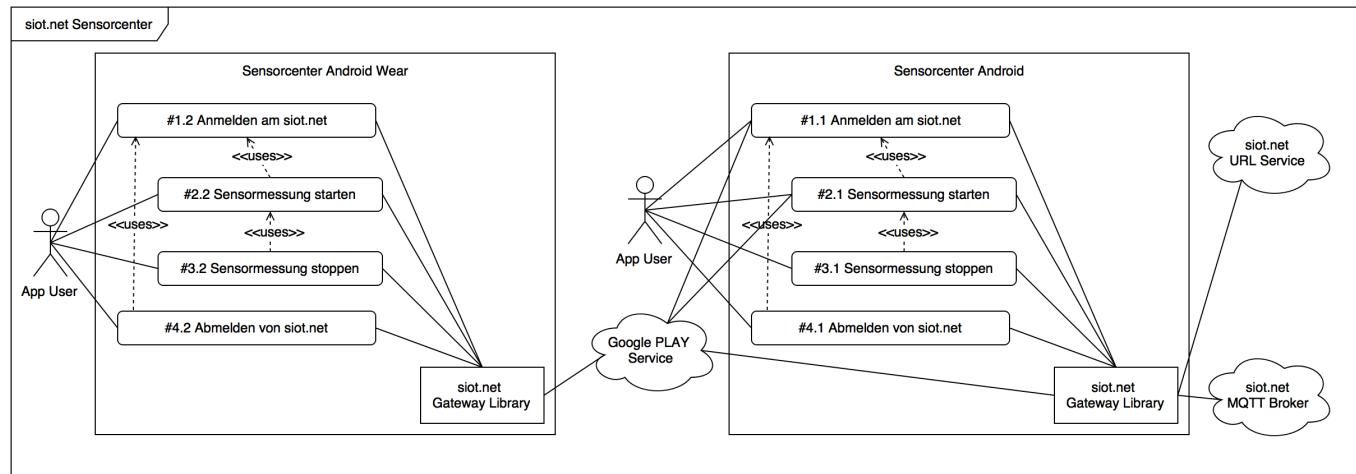


Abbildung 8.2.: Das allgemeine Use Case Diagramm für das siot.net Sensorcenter

8.2.3. Funktionale Anforderungen und Anwendungsfallbeschreibungen

8.2.3.1. Anwendungsfall #1.1 und #1.2

Nr. und Name:	#1.1 und #1.2 Anmelden am siot.net.
Szenario:	Android Mobil oder Wear Gerät wird am siot.net angemeldet.
Kurzbeschreibung:	Die Verbindung zum siot.net wird hergestellt. Bei beiden Use Cases geschieht dies über die siot.net Gateway Library. Jedoch wird beim Anwendungsfall 1.2 die Verbindung über das gekoppelte Android Mobile Device hergestellt.
Beteiligt Akteure:	App User, siot.net Gateway Library, siot.net URL Service, siot.net MQTT Broker, Google Play Service (bei #1.2).
Auslöser und Vorbedingung:	User gibt siot.net Lizenzschlüssel ein und aktiviert den „Verbinden“ Button.
Ergebnisse und Nachbedingung:	Mit den eingegebenen Daten kann die siot.net Gateway Library erfolgreich eine Verbindung aufbauen.

Tabelle 8.20.: siot.net Sensorcenter: Übersicht Anwendungsfall #1.1 und #1.2

Ablauf #1.1:

1.1-1	App User	Sensorcenter App auf dem Smartphone starten.
1.1-2	App User	Lizenzschlüssel von siot.net eingeben und „Verbinden“ klicken.
1.1-3	sSC-A	Fordert bei der siot.net Gateway Library (sGWLlib) die Verbindung an.
1.1-4	sSC-A	Erhält die Antwort des Verbindungsversuches.
1.1-5	sSC-A	Zeigt die Informationen zur Verbindung an. Alle verfügbaren Sensoren werden mit Ein-/Ausschalter eingeblendet.

Tabelle 8.21.: siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #1.1

Ablauf #1.2:

1.2-1	App User	Sensorcenter App auf der Smartwatch starten.
1.2-2	sSC-AW	App auf dem Smartphone starten lassen.
1.2-3	sSC-A	siot.net Sensorcenter Android App auf dem Android Mobil Gerät starten.
1.2-4	App User	„Verbinden mit siot.net“ auf der Uhr klicken.
1.2-5	sSC-AW	Verbindung zu siot.net wird von sSC-A verlangt.
1.2-6	sSC-A	sSC-A fordert, unter Verwendung des bereits eingegebenen Lizenzschlüssels, die Kopplung mit siot.net von der Gateway Library.
1.2-7	sSC-A	Verbindet Smartphone zu siot.net und gibt die Verbindungsdaten an sSC-AW weiter.
1.1-5	sSC-AW	Visualisiert Aktiverungs- und Deaktivierungsschalter aller verfügbaren Sensoren.

Tabelle 8.22.: siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #1.2

Ausnahmen und Varianten #1.1 und 1.2:

1.2-2.1	sSC-AW	Ausnahme: Smartphone nicht erreichbar, Sensorcenter kann auf der Smartwatch nicht verwendet werden.
1.2-2.2	sSC-AW	Ausnahme: Google Play Service nicht verfügbar, Sensorcenter kann auf der Smartwatch nicht verwendet werden.
1.1-3.1 1.2-6.1	sSC-A	Ausnahme: siot.net Gateway Library antwortet, dass siot.net URL Service nicht erreichbar ist. Verbindung kann nicht aufgebaut werden.
1.1-3.2 1.2-6.2	sSC-A	Ausnahme: siot.net Gateway Library antwortet, dass siot.net Lizenz nicht gültig ist. User benachrichtigen.
1.1-3.3 1.2-6.3	sSC-A	Ausnahme: siot.net Gateway Library antwortet, dass siot.net MQTT Broker nicht verfügbar ist. Verbindung kann nicht aufgebaut werden.

Tabelle 8.23.: siot.net Sensorcenter: Ausnahmen und Varianten von Anwendungsfall #1.1 und #1.2

8.2.3.2. Anwendungsfall #2.1 und #2.2

Nr. und Name:	#2.1 und #2.2 Sensormessungen starten.
Szenario:	Sensor wird auf Android oder Android Wear Gerät gestartet. Messungen werden an siot.net übermittelt.
Kurzbeschreibung:	Mit den eingeblendeten Ein- und Ausschaltern kann jeder verfügbare Sensor einzeln aktiviert werden. Die gemessenen Daten werden bei #2.1 mittels siot.net Gateway Library via MQTT an den siot.net Broker weitergeleitet. Beim Anwendungsfall #2.2 werden diese über den Google Play Service ans Smartphone geleitet, welches dann die Informationen an den MQTT Broker sendet.
Beteiligt Akteure:	App User, siot.net Gateway Library.
Auslöser und Vorbedingung:	Ein oder mehrere Sensoren werden mit der App eingeschaltet.
Ergebnisse und Nachbedingung:	Sensor kann erfolgreich aktiviert werden.

Tabelle 8.24.: siot.net Sensorcenter: Übersicht Anwendungsfall #2.1 und #2.2

Ablauf #2.1 und #2.2:

2-1	App User	Einen gewählten Sensor via Ein-/Ausschalter aktivieren.
2-2	sSC-A/sSC-AW	Aktiviert den Sensor in der siot.net Gateway Library (Verantwortung für das Manifestieren und die Verwaltung der Kommunikation wird an die Bibliothek übergeben).
2-3	sSC-A/sSC-AW	Sensor im GUI als „eingeschaltet“ darstellen.

Tabelle 8.25.: siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #2.1 und #2.2

Ausnahmen und Varianten #2.1 und #2.2:

-	-	Keine Ausnahmen und Varianten
---	---	-------------------------------

Tabelle 8.26.: siot.net Sensorcenter: Ausnahmen und Varianten von Anwendungsfall #2.1 und #2.2

8.2.3.3. Anwendungsfall #3.1 und #3.2

Nr. und Name:	#3.1 und #3.2 Sensormessungen stoppen.
Szenario:	Sensor wird auf Android oder Android Wear Gerät deaktiviert.
Kurzbeschreibung:	Jeder aktivierte Sensor kann, durch die auf dem GUI sichtbaren Schalter, ausgeschaltet werden.
Beteiligt Akteure:	App User, siot.net Gateway Library.
Auslöser und Vorbedingung:	Ein oder mehrere Sensoren werden mit der App ausgeschaltet.
Ergebnisse und Nachbedingung:	Sensor kann erfolgreich deaktiviert werden.

Tabelle 8.27.: siot.net Sensorcenter: Übersicht Anwendungsfall #3.1 und #3.2

Ablauf #3.1 und #3.2:

3-1	App User	Einen gewählten Sensor mittels Schalter deaktivieren.
3-2	sSC-A/sSC-AW	Aktiviert den Sensor in der siot.net Gateway Library (Verantwortung für das Manifestieren und die Verwaltung der Kommunikation wird an die Bibliothek übergeben).
3-3	sSC-A/sSC-AW	Sensor im GUI als „ausgeschaltet“ darstellen.

Tabelle 8.28.: siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #3.1 und #3.2

Ausnahmen und Varianten #3.1 und #3.2:

-	-	Keine Ausnahmen und Varianten
---	---	-------------------------------

Tabelle 8.29.: siot.net Sensorcenter: Ausnahmen und Varianten von Anwendungsfall #3.1 und #3.2

8.2.3.4. Anwendungsfall #4.1 und #4.2

Nr. und Name:	#4.1 und #4.2 Abmelden von siot.net.
Szenario:	Alle Sensoren stoppen und die Verbindung zu siot.net trennen.
Kurzbeschreibung:	Durch die App gestarteten Sensoren werden deaktiviert.
Beteiligt Akteure:	App User, siot.net Gateway Library.
Auslöser und Vorbedingung:	Ein oder mehrere Sensoren werden mit der App ausgeschaltet.
Ergebnisse und Nachbedingung:	Sensor kann erfolgreich deaktiviert werden.

Tabelle 8.30.: siot.net Sensorcenter: Übersicht Anwendungsfall #4.1 und #4.2

Ablauf #4.1:

4.1-1	App User	„Abmelden“ Button geklickt.
4.1-2	sSC-A	Deaktiviert alle in der App aktivierten Sensoren. Meldung an sSC-AW, dass abgemeldet wird.
4.1-3	sSC-AW	Schaltet alle Sensoren aus, welche der User aktiviert hat. Die Verbindung wird getrennt.
4.1-4	sSC-AW	Anmelde Ansicht wird angezeigt.
4.1-5	sSC-A	Gibt der siot.net Gateway Library den Befehl, die Verbindung zum MQTT Broker zu schließen.
4.1-5	sSC-A	Display zeigt das Login GUI.

Tabelle 8.31.: siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #4.1

Ablauf #4.2:

4.2-1	App User	„Abmelden“ Button geklickt.
4.2-2	sSC-AW	Schaltet alle Sensoren aus, welche der User aktiviert hat. Die Verbindung wird getrennt.
4.2-3	sSC-AW	Anmelde Ansicht wird angezeigt.

Tabelle 8.32.: siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #4.2

Ausnahmen und Varianten #4.1 und #4.2:

4.1-2.1	sSC-A	Kein Gerät verbunden welches sSC-AW ausführt. Keine Aktion notwendig.
----------------	-------	---

Tabelle 8.33.: siot.net Sensorcenter: Ausnahmen und Varianten von Anwendungsfall #4.1 und #4.2

8.2.4. Nicht-funktionale Anforderungen**Produktanforderungen:**

Benutzbarkeitsanforderungen	Die Applikation muss mit einem Android Smartphone oder einer Android Smartwatch bedienbar sein. Die Smartwatch muss die Funktion Always-On Display oder Ambient Modus beherrschen.
Effizienzanforderungen	Die Bedienung muss einfach gehalten werden, erleichtert die Verwendung für aller Art Benutzer.
Zuverlässigkeitssanforderungen	Displayanzeigen müssen verzögerungsarm sein. Speicherung von mindestens einem Lizenzschlüssel muss gewährleistet werden. Verbindung zum Internet muss vorhanden sein.
Portierbarkeitsanforderungen	Es werden nur Daten gespeichert, welche die für diese Applikation notwendig sind. Es gibt keine Portierbarkeitsanforderungen.
Performanceanforderung	Die Anzeige muss immer in einer für Menschen brauchbaren Geschwindigkeit reagieren. Es braucht eine Internetverbindung, um Sensordaten zu senden. Da die Daten Pakete sind sehr klein sind (<1KB), ist keine grosse Bandbreite notwendig.

Tabelle 8.34.: siot.net Sensorcenter: Nicht-funktionale Produktanforderungen

8.3. Requirements - siot.net Dashboard

In diesem Requirementsdokument werden die Anforderung an die Android App siot.net Dashboard (**sDB**) spezifiziert.

8.3.1. Allgemeine Beschreibung

Das siot.net Dashboard für Geräte mit dem Android Betriebssystem, ist eine App, welche die siot.net Gateway Library (**sGWLib**) integrieren soll. Auf einem grossen Touchscreen können ermittelte Daten übersichtlich dargestellt werden. Es soll mindestens die Anzeige von Graphen, Landkarten und Freitext Wertanzeige möglich sein.

8.3.2. User Requirements

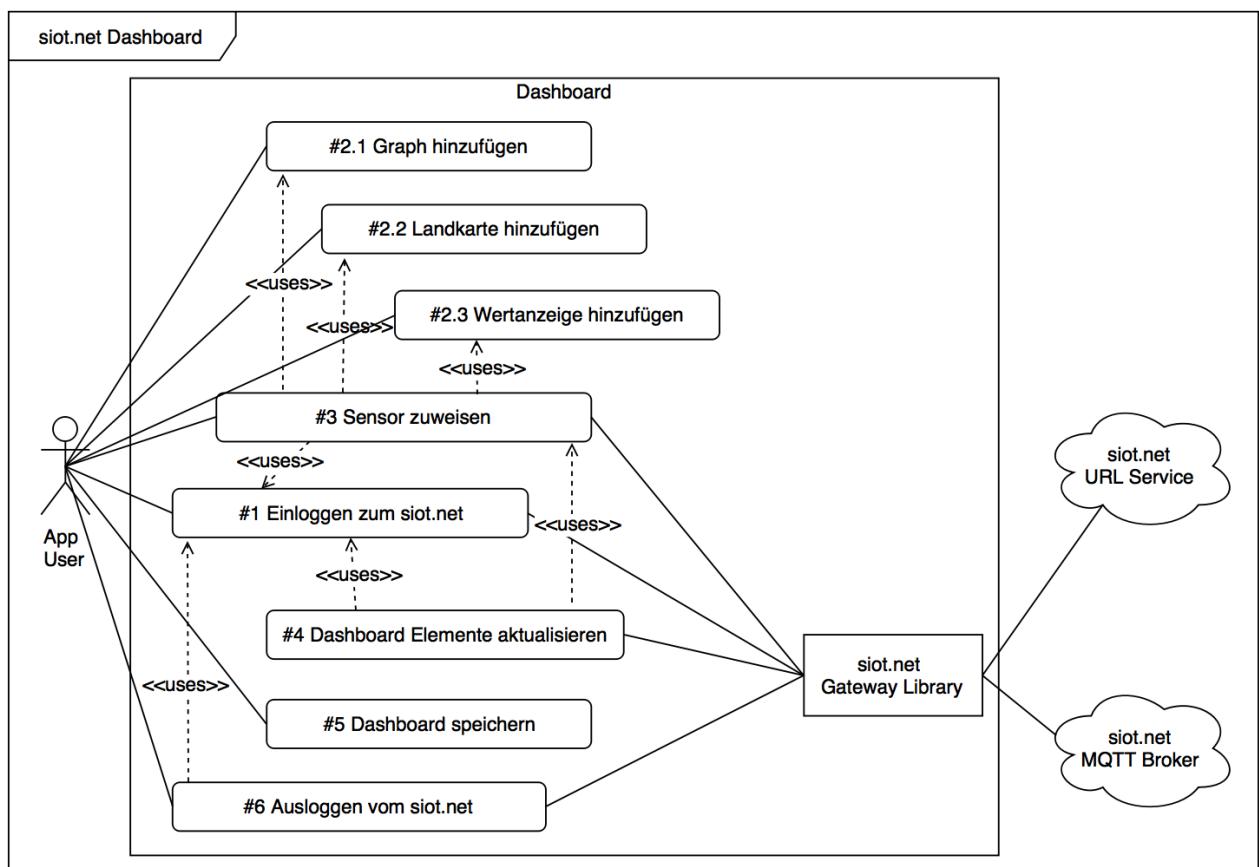


Abbildung 8.3.: Das allgemeine Use Case Diagramm für das siot.net Dashboard

8.3.3. Funktionale Anforderungen und Anwendungsfallbeschreibungen

8.3.3.1. Anwendungsfall #1

Dieser Anwendungsfall #1 ist Deckungsgleich mit dessen im Kapitel 7.2.3, dem Anwendungsfall #1.1. Das Loginverfahren wird in beiden Applikationen identisch behandelt. Das Dashboard soll nur auf grösseren Android Geräten zur Verfügung stehen. Auf einem kleinen Display, der Wearable Devices, wäre die Darstellung der Informationen nicht übersichtlich.

8.3.3.2. Anwendungsfall #2

Nr. und Name:	#2.1 Graph, #2.2 Landkarte und #2.3 Wertanzeige hinzufügen.
Szenario:	Dem Dashboard ein Anzeigeelement hinzufügen.
Kurzbeschreibung:	Der Applikation sollen während dem Betrieb, neue Objekte hinzugefügt werden, um Informationen anzuseigen.
Beteiligt Akteure:	App User.
Auslöser und Vorbedingung:	User wählt ein neues Element aus.
Ergebnisse und Nachbedingung:	Das ausgesuchte Anzeigobjekt ist auf dem App sichtbar.

Tabelle 8.35.: siot.net Dashboard: Übersicht Anwendungsfall #2

Ablauf #2:

2-1	App User	Wählt einen Graph, Landkarte oder Freitext Wertanzeige aus.
2-2	sDB	Generiert das neue GUI Element und eine leere Auswahlliste.
2-3	sDB	Zeigt beide Objekte an.

Tabelle 8.36.: siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #2

Ausnahmen und Varianten #2:

2-2.1	sDB	Variante: Dem siot.net Dashboard sind bereits die Sensoren vom IoT-Center bekannt, Auswahliste kann bereits befüllt werden.
--------------	-----	---

Tabelle 8.37.: siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #2

8.3.3.3. Anwendungsfall #3

Nr. und Name:	#3 Sensor zuweisen.
Szenario:	Einer sichtbaren Informationsanzeige einen dazugehörigen Sensor zuweisen.
Kurzbeschreibung:	Die im Anwendungsfall #2 generierte Auswahliste soll mit passenden Sensoren gefüllt werden. Die Daten eines ausgewählten Sensors sollen auf der Anzeige dargestellt werden.
Beteiligt Akteure:	App User, siot.net Gateway Library, siot.net MQTT Broker.
Auslöser und Vorbedingung:	Neues Anzeigeelement wurde erzeugt. User will ein Sensor dazu referenzieren.
Ergebnisse und Nachbedingung:	Daten des ausgewählten Sensors werden auf dem Graph, der Landkarte oder der Wertanzeige dargestellt.

Tabelle 8.38.: siot.net Dashboard: Übersicht Anwendungsfall #3

Ablauf #3:

3-1	App User	Wählt eine Auswahlliste eines Anzeigeobjekt.
3-2	sDB	Verlangt von der siot.net Gateway Library alle vorhanden Sensoren mit passendem Typ.
3-3	sDB	Zeigt die verfügbaren Sensoren an.
3-4	App User	Wählt einen Sensor an.
3-5	sDB	Abonniert die Informationen bei der siot.net Gateway Library.
3-6	sDB	Anzeige wird mit den Daten dargestellt und synchronisiert.

Tabelle 8.39.: siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #3

Ausnahmen und Varianten #3:

3-2.1	sDB	Ausnahme: Keine Sensoren vorhanden. Keine Aktion möglich. Auswahliste bleibt leer.
3-6.1	sDB	Ausnahme: Keine Sensordaten vorhanden. Anzeige wird erst angereichert, wenn Daten ein treffen.

Tabelle 8.40.: siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #3

8.3.3.4. Anwendungsfall #4

Nr. und Name:	#4 Dashboard Elemente aktualisieren.
Szenario:	Neu verfügbare Daten den Anzeigeobjekten zur Verfügung stellen.
Kurzbeschreibung:	Die im Anwendungsfall #3 abonnierte Daten müssen verarbeitet werden. Ein klassisches Model-View-Controller Pattern kommt hier zum Einsatz.
Beteiligt Akteure:	siot.net Gateway Library, siot.net MQTT Broker.
Auslöser und Vorbedingung:	siot.net Gateway Library empfängt neue MQTT Message und notifiziert das Dashboard.
Ergebnisse und Nachbedingung:	Graph, der Landkarte oder der Wertanzeige wird aktualisiert.

Tabelle 8.41.: siot.net Dashboard: Übersicht Anwendungsfall #4

Ablauf #4:

4-1	sDB	Erhält von der siot.net Gateway Library die Benachrichtigung, dass neue Daten vorhanden sind.
4-2	sDB	Analysiert die Daten.
4-3	sDB	Aktualisiert die zu den Informationen gehörigen Anzeigen.

Tabelle 8.42.: siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #4

Ausnahmen und Varianten #4:

3-2.1	sDB	Ausnahme: Keine Daten, welche relevant sind. Keine Aktion nötig.
--------------	-----	--

Tabelle 8.43.: siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #4

8.3.3.5. Anwendungsfall #5

Nr. und Name:	#5 Dashboard speichern.
Szenario:	Das erstellte Dashboard speichern.
Kurzbeschreibung:	Speichern des Dashboards, um es ein anderes Mal wieder zu verwenden oder auch anderen Usern im gleichen IoT-Center bereit zu stellen.
Beteiligt Akteure:	App User.
Auslöser und Vorbedingung:	User drückt „Dashboard speichern“ Taste.
Ergebnisse und Nachbedingung:	Dashboard kann in einer Datei persistiert werden.

Tabelle 8.44.: siot.net Dashboard: Übersicht Anwendungsfall #5

Ablauf #5:

5-1	App User	Auslösen der Aktion, durch betätigen des „Dashboard speichern“ Knopfes.
5-2	sDB	Struktur und Zuordnungen werden in eine Datei auf dem Gerät niedergeschrieben.
5-3	sDB	User erhält Bestätigung.

Tabelle 8.45.: siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #5

Ausnahmen und Varianten #5:

5-2.1	sDB	Ausnahme: Datei kann nicht geschrieben werden. User benachrichtigen.
--------------	-----	--

Tabelle 8.46.: siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #5

8.3.3.6. Anwendungsfall #6

Nr. und Name:	#6 Ausloggen vom siot.net.
Szenario:	Löst die Verbindung zu der IoT-Cloud auf.
Kurzbeschreibung:	Dieser Schritt erlaubt eine saubere Abmeldung vom Netzwerk. Hier wird der Impuls der siot.net Gateway Library gegeben, um alle Verbindungen Sauber zu trennen.
Beteiligt Akteure:	App User, siot.net Gateway Library.
Auslöser und Vorbedingung:	„Abmelden“ Schaltfläche wird aktiviert.
Ergebnisse und Nachbedingung:	Dashboard erfolgreich vom siot.net abgemeldet.

Tabelle 8.47.: siot.net Dashboard: Übersicht Anwendungsfall #6

Ablauf #5:

5-1	App User	Abmelden wird ausgeführt.
5-2	sDB	Meldet dem siot.net Gateway zum trennen der Verbindungen.
5-3	sDB	Login Bildschirm wird angezeigt.

Tabelle 8.48.: siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #6

Ausnahmen und Varianten #6:

-	-	Keine Ausnahmen und Varianten
---	---	-------------------------------

Tabelle 8.49.: siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #6

8.3.4. Nicht-funktionale Anforderungen

Produktanforderungen:

Benutzbarkeitsanforderungen	Die Applikation muss mit einem Android Smartphone funktionstüchtig sein.
Effizienzanforderungen	Die Anwendung muss für alle Smartphone User, mit Android 5.0 (Lollipop) und höher, bedienbar sein.
Zuverlässigkeitssanforderungen	Graphen, Landkarten und Werteanzeigen sollten möglichst flüssig angezeigt und aktualisiert werden.
Portierbarkeitsanforderungen	Speichern eins Dashboards soll möglichst in einer generischen Struktur geschehen. In einem weiteren Release, wird in Betracht gezogen, dass Dashboards portierbar sind. Das heisst, sie können auch im siot.net abgelegt werden.
Performanceanforderung	Die Anzeige muss flüssig reagieren, um die Graphen brauchbar zu visualisieren. Es braucht eine Internetverbindung, um Sensorinformationen zu empfangen. Da die Datenpakete sehr klein sind (<1KB), ist keine sehr grosse Bandbreite notwendig.

Tabelle 8.50.: siot.net Dashboard: Nicht-funktionale Produktanforderungen

9. Konzepte

Dieses Kapitel dokumentiert das Softwaredesign der siot.net Gateway Library und des siot.net Sensorcenters.

9.1. Konzept - siot.net Gateway Library

9.1.1. Packagediagramm

Das auf Abbildung 9.1 visualisierte Paketdiagramm, zeigt die Einteilung der Klassen und deren Abhängigkeiten.

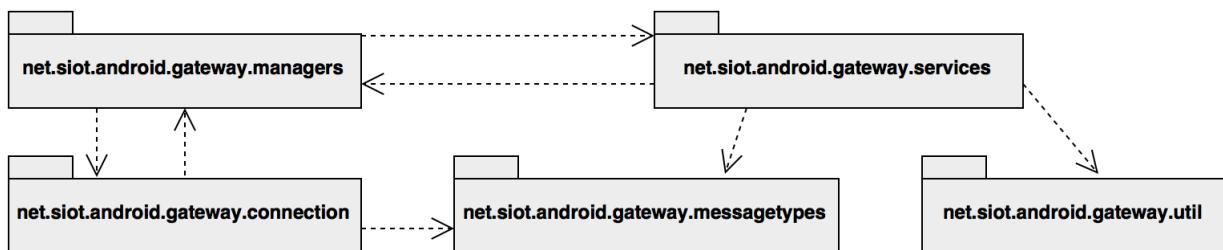


Abbildung 9.1.: Java-Package Aufteilung der siot.net Gateway Library

Die Auflistung der einzelnen Packages mit den enthaltenen Klassen (Verwendung von Entwurfsmuster sind in Klammern).

net.siot.android.gateway.managers

- SiotNetGatewayManager (Abstrakte Fabrikklasse)
- SiotNetGatewayManagerMobile (Fabrikklasse)
- SiotNetGatewayManagerWear (Fabrikklasse)
- ReceivedMessageManager(Beobachtbare-Klasse)

net.siot.android.gateway.connection

- MQTTClient (Singleton Klasse)
- RestClient

net.siot.android.gateway.messagetypes

- SensorActorManifest
- SensorConfig
- ActorConfig
- SiotUrl
- Urls
- WearableData

net.siot.android.gateway.services

- SensorService (Abstrakte Fabrikklasse)
- SensorServiceMobile (Fabrikklasse)
- SensorServiceWear (Fabrikklasse)
- LocationService

net.siot.android.gateway.util

- GUIDUtil
- TopicUtil
- SensorTypeKeys

Tabelle 9.1.: siot.net Gateway Library: Auflistung der Packageinhalte

9.1.2. Domänendiagramm

Auf der folgenden Abbildung 9.2, ist das gesamte konzeptionelle Domänendiagramm der siot.net Gateway Library aufgezeichnet.

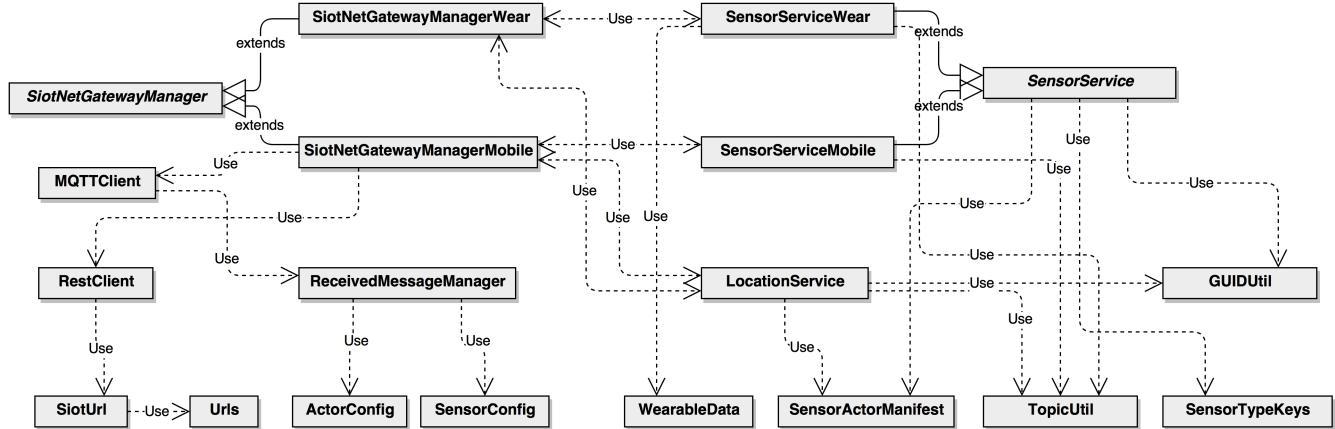


Abbildung 9.2.: Die Gesamtübersicht aller Klassen der siot.net Gateway Library

Apps, welche dieses Paket integrieren, instanzieren in erster Linie die Managerklassen. Die *SiotNetGatewayManagers* sind in Anlehnung des Designpatterns der Fabrikmethode (vgl. [3] Chapter 3. Creational Patterns) konzipiert. Falls noch weitere Android Gerätearten hinzukommen, kann eine dazu passende Fabrik dazu programmiert werden (z.B. *SiotNetGatewayManagerCar*). Die *ReceivedMessageManager* Klasse ist für den Gebrauch in einem Beobachter-Entwurfsmuster (vgl. [3] Chapter 5. Behavioral Patterns) vorbereitet. Sie ist Observable und sollte von einem Observer Objekt instanziert werden. Von diesem Objekt darf nur eine Instanz erstellt werden. Deswegen kommt hier das Singleton-Entwurfsmuster (vgl. [3] Chapter 3. Creational Patterns) zur Anwendung. Die Serviceklassen beinhalten die Verwaltung und die Kommunikation von den Sensoren und den Ortungsdaten. Die Sensorservices sind ebenfalls mit der Fabrikmethode konzipiert.

Das Connection-Paket ist für die Verbindungsobjekte zuständig. Der *MQTTClient* wird gleichwohl, wie der *ReceivedMessageManager*, mit einer Singleton-Klasse angesteuert. *RestClient* implementiert die statische Methode zum konsultieren des URL-Services.

In Utils sind Helferklassen enthalten und in messagetypes werden die Datentypen der Kommunikation definiert.

9.1.3. Sequenzdiagramme

Die grundlegenden Abläufe sind mit Sequenzdiagrammen spezifiziert.

9.1.3.1. App starten

Die Vorgänge, welche beim starten einer Smartphone- und Smartwatch-App durchlaufen werden, sind in der Abbildung 9.3 und 9.5 illustriert.

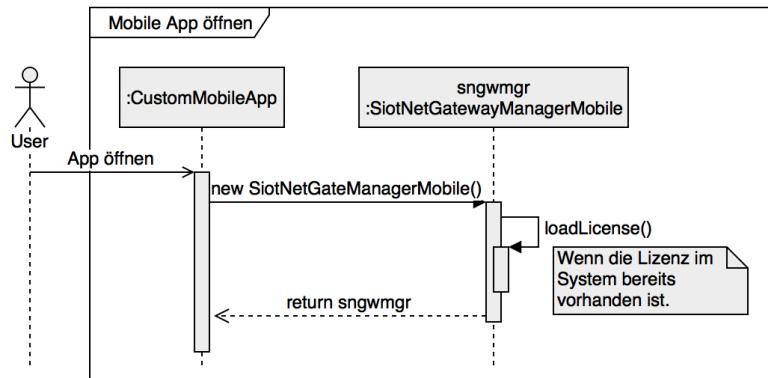


Abbildung 9.3.: Sequenzdiagramm: Starten von Smartphone App

Das normale Starten einer Smartphone Applikation, welche die siot.net Gateway Library integriert, wird auf der Abbildung 9.3 geschildert.

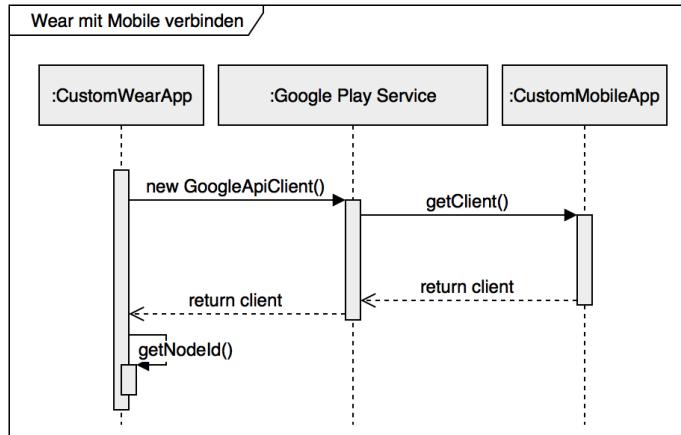


Abbildung 9.4.: Sequenzdiagramm: Verbindungsauflaufbau von Smartwatch zu Smartphone

Wenn die dazugehörige Smartwatch Anwendung aufgerufen wird, welche im vorgehenden Diagramm erwähnt wurde, verbindet es sich zuerst via Google Play Service zum gekoppelten Android Smartphone. Dazu die erklärende Abbildung 9.4. Das danach folgende Sequenzdiagramm auf Abbildung 9.5 hat eine Abhängigkeit darauf.

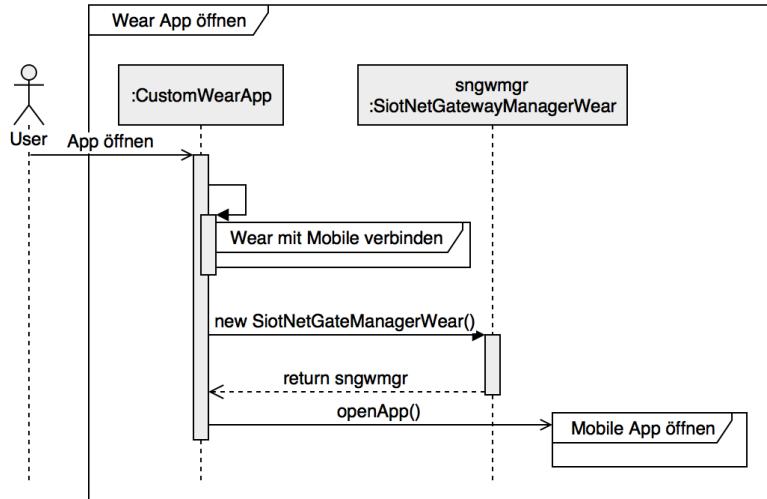


Abbildung 9.5.: Sequenzdiagramm: Starten von Smartwatch App

Der Minicomputer am Handgelenk kann die siot.net Schnittstelle nur dann zur Verfügung stellen, wenn die Verbindung zum Smartphone aktiv ist. Zum erzeugen des Managerobjektes muss der vorher erhaltene GoogleApiClient mitgegeben werden.

9.1.3.2. Login zu siot.net

Die Kommunikationsabfolgen für das Login zum siot.net sind auf Abbildung 9.6 und 9.7 visualisiert.

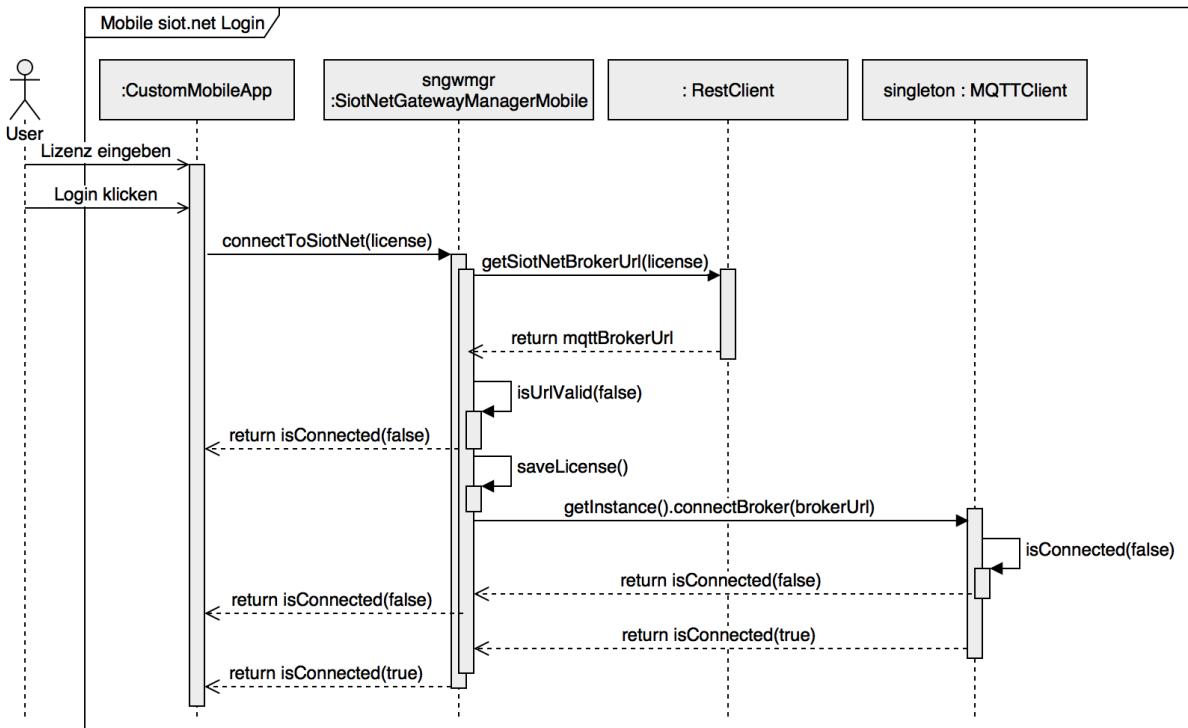


Abbildung 9.6.: Sequenzdiagramm: Einloggen von einer Smartphone App

Abbildung 9.6 erläutert das Loginverfahren über ein Android Mobile Gerät. Für das Login benötigt es immer einen gültigen siot.net Lizenzcode. Nach einer erfolgreichen Autorisierung beim siot.net URL-Service, wird der Schlüssel im System in einer Konfigurationsdatei abgelegt. Dieser wird bei jedem Start geladen (siehe Abbildung 9.3). Aus

dieser positiven Bestätigung kann die Adresse des MQTT Brokers gelesen werden. Der weitere Verlauf ist auf der Abbildung 9.6 ersichtlich.

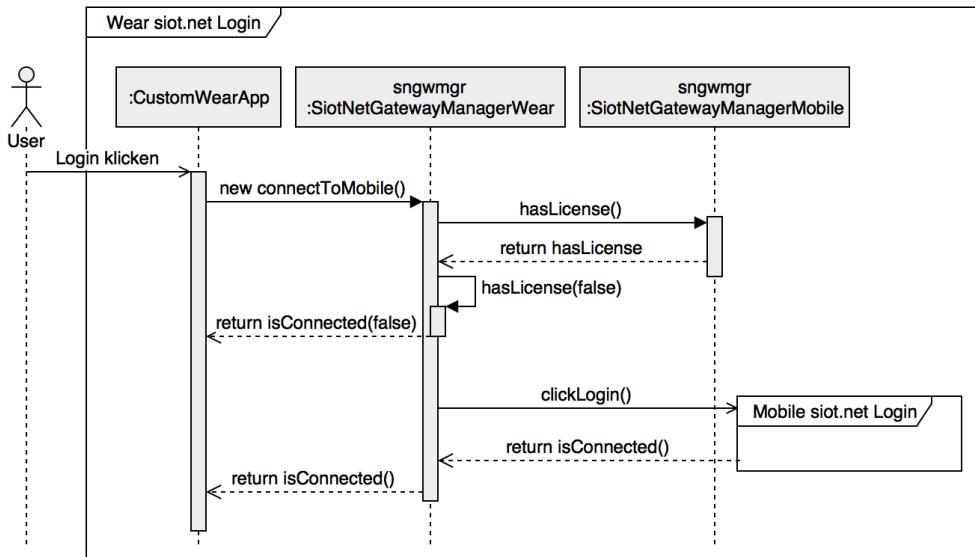


Abbildung 9.7.: Sequenzdiagramm: Einloggen von einer Smartwatch

Die Autorisierung via Smartwatch geschieht nur indirekt. Dieser verlangt, dass bereits ein valider Lizenzcode auf dem Smartphone existiert. Dies ist notwendig, da die Android Wear Geräte keine virtuelle Tastatur besitzen. Eine Eingabe wär nur per Sprachbefehl möglich, was die siot.net Gateway Library nicht unterstützt. Ist dieser vorhanden, fordert die Datenuhr das Smartphone dazu auf sich mit dem siot.net MQTT Broker zu verbinden. Die Abbildung 9.7 stellt das genaue Sequenzdiagramm zu diesem Vorgang dar.

9.2. Konzept - siot.net Sensorcenter

9.2.1. Domänendiagramm

Die Abbildung 9.8, ist das Domänendiagramm vom siot.net Sensorcenter.



Abbildung 9.8.: Die Klassenübersicht vom siot.net Sensorcenter im Domänendiagramm

Die Klassenstruktur der Endanwender-App kann sehr schlank gehalten werden, da die meisten Aufgaben die siot.net Gateway Library bereits erledigt. Es ist notwendig, dass zwei fast identische Applikationsstrukturen in einem Projekt bestehen. Die Wearable App differenziert sich erheblich zur Smartphone App. Des weiteren ist von Android vorgegeben, dass für die Smartphone Activity und die Watch Activity ein separates Package erstellt wird (vgl. [2]). Die Kernelemente jeweiliger Applikationen sind die Activityklassen. Diese sind zuständig für die grafische Benutzeroberfläche und instanzieren die siot.net Gateway Library. Das nötige Managerobjekt wird mit Hilfe der passenden Manager-Fabrik erzeugt. Der `ListenerService` und der `MessageReceiver` sind zuständig für die Kommunikation zwischen Smartphone und Smartwatch.

9.2.2. Sequenzdiagramm

Die folgenden Abbildungen 9.9 und 9.10 legen fest, wie die Sensorsteuerung, in Kombination mit der Gateway Bibliothek, von stattent geht.

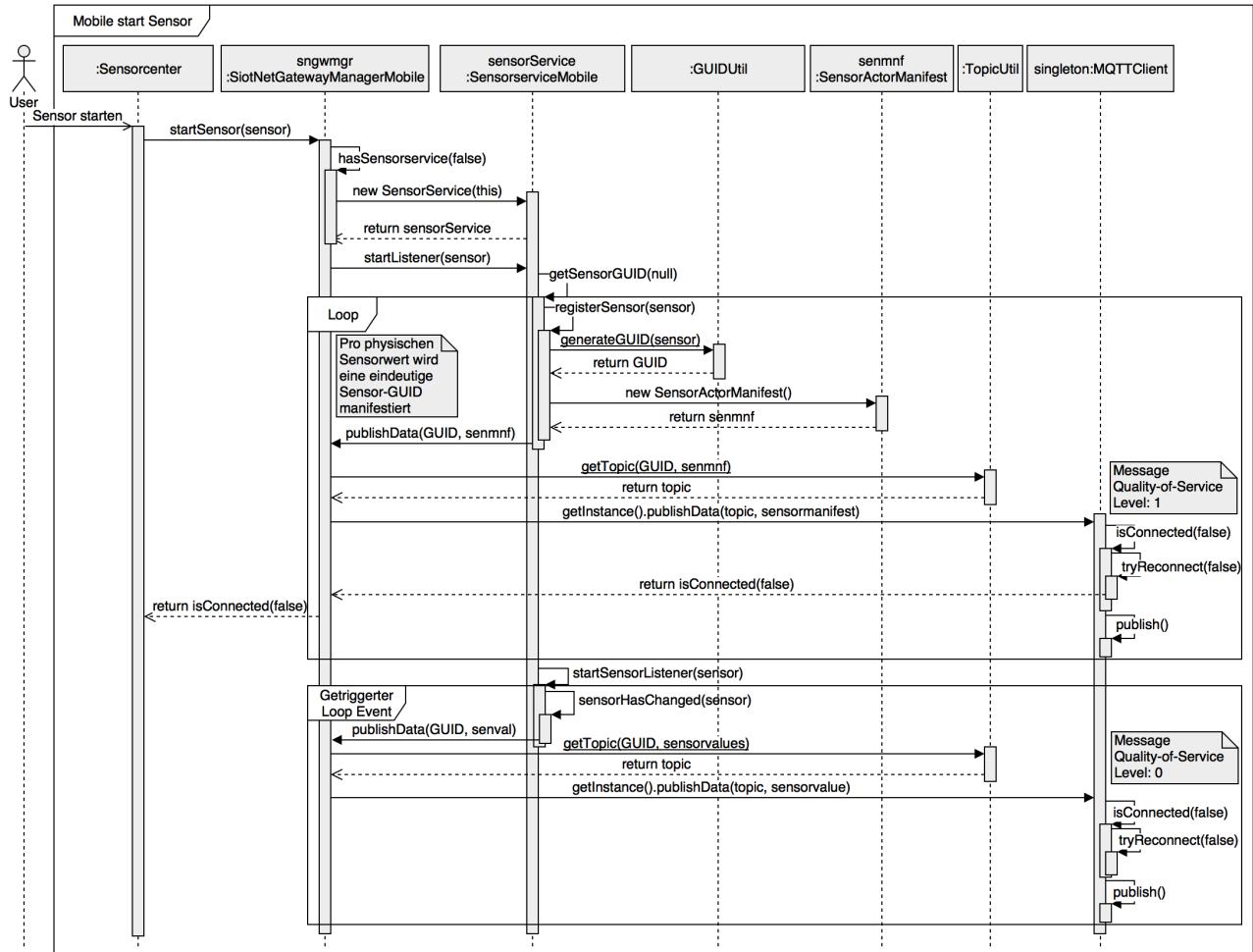


Abbildung 9.9.: Der Ablauf beim starten eines Sensors im Sensorcenter

Die erste Spezialität in Abbildung 9.9 ist beim **Loop** Rahmen. Dieser Turnus kommt dann zum Zug, wenn der Sensor zum ersten Mal gestartet wird. Ein Sensor muss sich unter Umständen mehrfach manifestieren, aufgrund mehreren physischen Werten. Beispielsweise hat ein Beschleunigungsmesser in einem dreidimensionalen System die Werte der X-Achse, Y-Achse und Z-Achse, das heisst pro Achse wird ein Manifest des Sensors erzeugt. Nach der erfolgreichen Anmeldung des Sensors, wird der, vom Android System bereitgestellten Zuhörer des Sensors, aktiviert. Dieser gibt bei jeder Veränderung des Sensors einen Messwert zurück, dass dann eine MQTT Nachricht an den siot.net Broker auslöst. Solange der Messwertgeber nicht ausgeschaltet ist, wird nach jeder Auslösung des Sensorlisteners die Sequenz im Rahmen **Getrigerter Loop Event** durchlaufen.

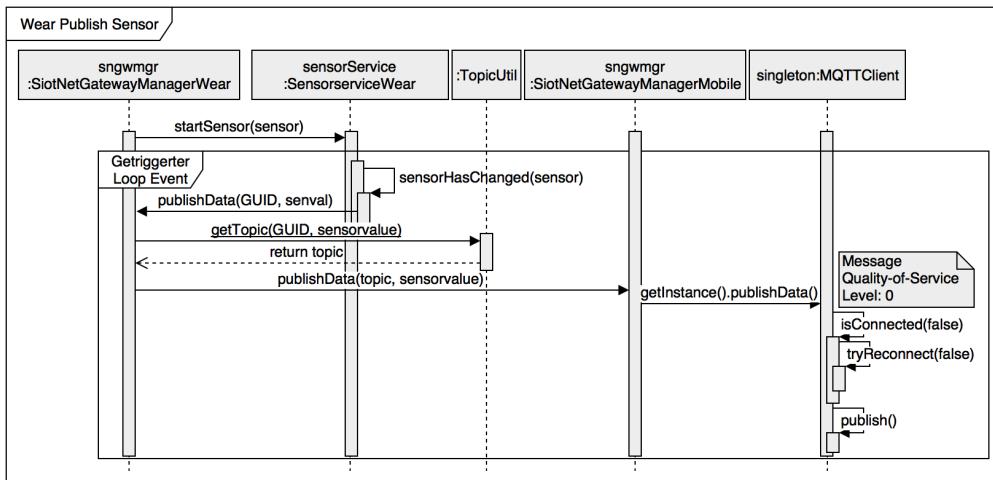


Abbildung 9.10.: Kommunikationsverhalten beim senden von Daten bei Smartwatch

Das Auslösen von Nachrichten bei der Smartwatch benötigt noch einen weiteren Schritt. Durch die Beschränkung, dass keine Werte direkt von der Uhr ins Internet durchgestellt werden können, müssen die Datenpaket zuerst dem Smartphone zugestellt werden. Um die Informationen durchzuschleusen, wird dieses als Datenbrücke verwendet (wie auf Abbildung 7.5 und 7.6 geschildert).

9.3. Konzept - Kommunikation Smartphone-zu-Smartwatch

Für die Kommunikation von Smartphone zu Smartwatch und umgekehrt wurde ein Kommunikationskonzept zugewiesen. Die Kommunikation soll ausschliesslich über die asynchrone Schnittstelle der Android MessageAPI verlaufen.

9.3.1. Request-Response

Anweisungen und Informationen, welche zwischen den zwei Geräten ausgetauscht werden, müssen im Request-Response Verfahren ablaufen. Dies ist notwendig, weil Aktionen des einen Gerätes, Einfluss auf das Andere haben kann. Kommunikation und Funktionalität auf der Smartwatch sind stark gebunden mit dem Status des Smartphones. Darum müssen die Messages gegenseitig quittiert werden. Die Message-Struktur und -Inhalt ist durch den individuellen Entwickler selber zu bestimmen. Es ist eine synchrone Kommunikation über eine asynchrone Schnittstelle. Dies wird auch öfters als sync-over-async betitelt. Um eine rein synchrone Schnittstelle zu erhalten kann auch die ClientAPI aus den Google Play Services genutzt werden. Je nach Umfang der synchronen Daten ist dies sinnvoll. Der Vorteil des sync-over-async ist, dass die Daten nicht auf dem System persistent vorhanden sein müssen. Die aus den Nachrichten erfolgenden Aktionen sind immer ereignisorientiert.

9.3.2. Fire-and-Forget

Bei den Sensormesswerten der Smartwatch ist es anders geregt. Diese werden im Fire-and-Forget-Verfahren verschickt. Das bedeutet, die Daten werden ausgesendet aber werden nicht verfolgt oder kontrolliert, ob diese korrekt übermittelt wurden. Dies wird angewendet, weil es systemunkritische Informationen sind. Dadurch kann Energie und Bandbreite gespart werden. Ausnahme: Wenn bei der MQTT Kommunikation, der Quality-of-Service > 0 wird (1 oder 2), sollten die Messwerte von der Smartwatch im Nachrichtenaustauschmuster Request-Response übergeben werden. Sonst kann keine Verifikation durchgeführt werden ob die Nachricht angekommen ist.

10. Implementation

Dieses Kapitel beleuchtet einige Gesichtspunkte der Umsetzung. Die Logik, Abgrenzungen, Einhaltung des Konzepts, verwendete Plattformen und Libraries gehören dazu.

10.1. Abgrenzung

Zu Beginn der Realisierung mussten die Ziele abgegrenzt werden. Der Umfang vom definierten Anforderungskatalog und Konzept ist sehr gross. Für die Umsetzung wurden einige Kernelemente ausgesucht.

10.1.1. Ziele - siot.net Gateway Library

- Die siot.net Gateway Library als Prototypen realisiert werden.
- Mindestens die Verbindung zum siot.net aufnehmen können.
- Sensoren nach dem siot.net Messagedesign (vgl. [13] manifestieren.
- Daten an die IoT-Cloud im richtigen Format senden.
- In einer Applikation integriert werden.

10.1.2. Ziele - siot.net Sensorcenter

- Als Demoapplikation für die Gateway Library umsetzen.
- Für Smartphone und Smartwatch lauffähig.

10.2. Implementation - siot.net Gateway Library

Die Umsetzung erfolgte grösstenteils nach dem konzeptionierten Domänenmodell (siehe Abbildung 9.2). Nicht alle Funktionen wurden realisiert. LocationService und der ReceivedMessageManager sind nicht erfolgreich funktional umgesetzt worden. Der LocationService wurde ausgelassen, da dieser sich sehr dem SensorService ähnelt. Beim ReceivedMessageManager wurde bei der Umsetzung festgestellt, dass das Konzept nicht konsistent den Nutzen abdeckt. Dieser Manager muss überarbeitet und womöglich neu konzipiert werden.

10.2.1. Klassendiagramm

Auf der folgenden Seite ist (auf Abbildung 10.1) das Klassendiagramm der siot.net Gateway Library zu sehen. Die realisierten Klassen, sowie die entworfenen, jedoch nicht funktional umgesetzten Objekte (weisser Hintergrund), sind auf dieser Grafik visualisiert.

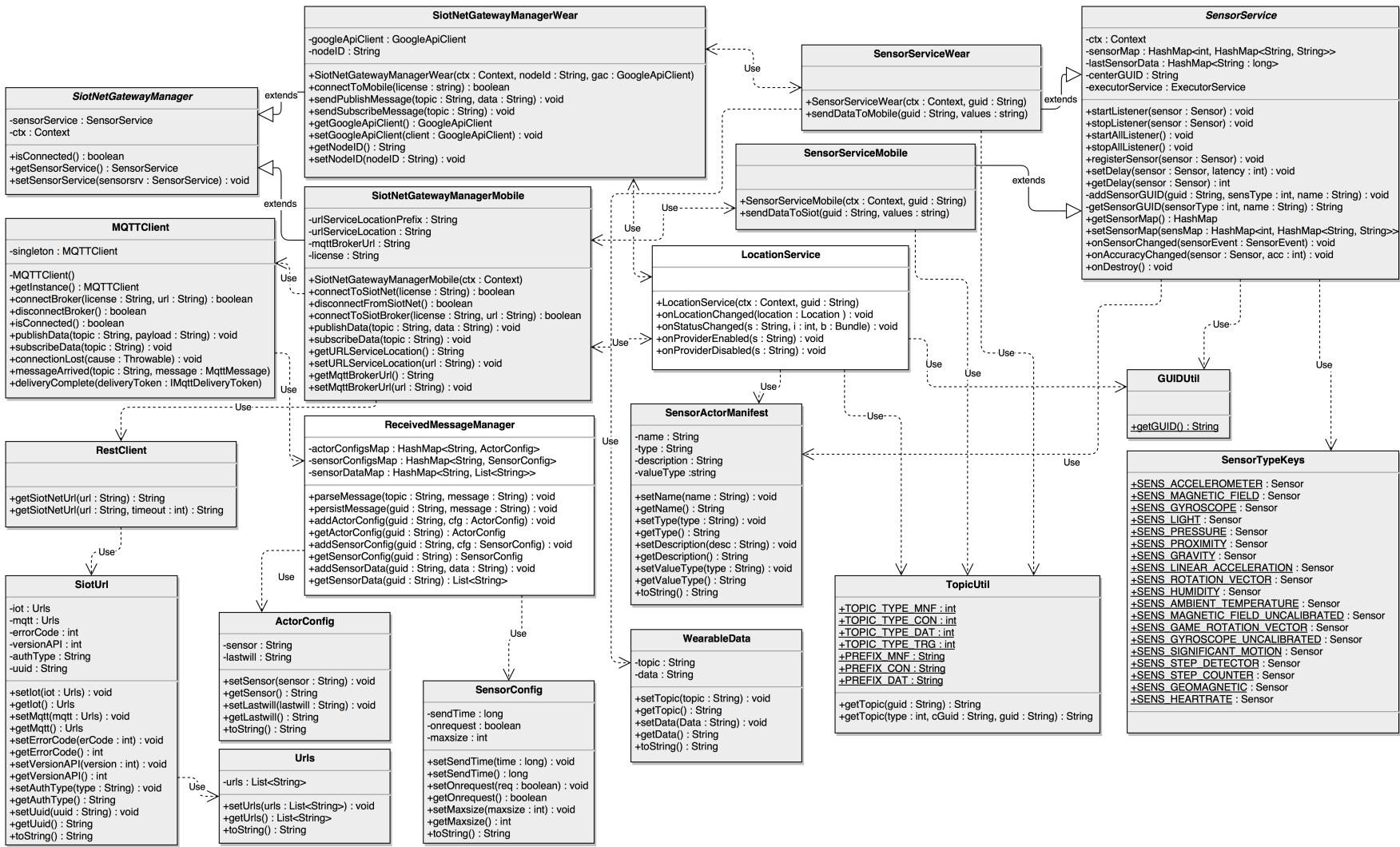


Abbildung 10.1.: Die Gesamtübersicht aller Klassen der siot.net Gateway Library

10.2.2. Logik

Einige interessante Logikaspekte sind in den *SiotNetGatewayManager* Klassen enthalten. Mit Hilfe des Mobile Managers benötigt es für den Verbindungsauflauf zum MQTT Broker nur die siot.net Lizenz. Beim ausführen der *connectToSiotNet(Lizenz)* Methode, wird per REST-API ein Authentifikationstoken angefordert. Um ein solches zu erhalten, wird ein HTTP GET Request abgesetzt, welches eine Bestätigung oder Zurückweisung im JSON-Format retourniert. Diese ist in der *RestClient* Klasse implementiert. Die Antwort wird auf den Messagetypr *SiotUrl* geparst. Die Broker URL wird aus der positiven Bestätigung gelesen.

Mit der aktuell erhaltenen Broker URL kann die Kopplung zum MQTT Broker starten. Für diese Angelegenheit dient die *MQTTClient* Klasse. Der *MQTTClient* Objekttyp ist eine, nach dem Prinzip des Singleton-Entwurfsmuster (vgl. [3] Chapter 3. Creational Patterns) implementierte, Klasse. Von der Library soll immer nur ein Objekt der Klasse *MQTTClient* bestehen, da dieser die Verbindung zum Broker verwaltet, also die Meldungen sendet und auch empfängt. Das Empfangen von Nachrichten durch mehrere Clients würde redundante Daten schaffen, wenn mehrere Subscriber auf dieselbe Topic hören. Desweitern würde die Netzwerkverbindung und Stromversorgung unnötig belastet werden. Der Singleton eignet sich ideal, diese Probleme zu beseitigen.

Um ein Objekt aus der *SiotNetGatewayManagerWear* Fabrikklasse (Entwurfsmuster Fabrikmethode, vgl. [3] Chapter 3. Creational Patterns) zu erzeugen, ist zwingend eine gültige Verbindung zum Smartphone nötig. Diese wird in Form von einer *GoogleApiClient* Instanz und der Nodeld des Smartphone verlangt. Beim instanziieren muss die Überprüfung der gültigen Kopplung von Mobile und Wearable bestanden werden. Im Konzept wurde eine Überprüfung nicht in Erwägung gezogen. Die vorhandene Verbindung ist essentiell für die Funktionsweise der Smartwatch-App.

Nach erfolgreichem kreieren der Manager Elemente, können die Sensoren aktiviert werden. Die *SensorenService* Klassen sind auch nach der Fabrikmethode konzipiert und implementiert. Sensoren müssen bei der ersten Aktivierung beim siot.net manifestiert werden. Eine Besonderheit dabei ist, dass jeder physische Sensorwert einzeln angemeldet wird. Dies muss in der Applikation verwaltet werden. Für dies wurde eine zweidimensionale Hashmap entworfen und verwendet. Die erste Dimension enthält den Schlüssel des SensorTypes, welcher eindeutig vom System vergeben ist und mit der zweiten Hashmap verknüpft ist. Die zweite beinhaltet den Namen des Messwertes (z.B. X-Achse) als Schlüssel und die siot.net GUID des physischen Sensorswertes. Beim stoppen und starten eines Sensors kann die GUID anhand vorhandener Informationen (SensorType und Messwertnamen) und der Zuweisung in der Hashmap evaluiert werden. Die GUID wird, für das Manifestieren des Sensors, durch die *GUIDUtil* Klasse generiert. Jeder Senorservice ist auch für das Versenden der Messwerte zuständig. Der MobileService sendet diese direkt per MQTT an den Broker. Der WearService sendet eine MessageAPI Nachricht, im Format des Message-typr *WearableData*, an das Smartphone. Beim Konzept wurde definiert, dass über die Manager versendet wird. Dies wurde umkonzeptioniert, so dass die *SensorService* den Manager nicht kennen müssen. Es wurde eine nicht notwendige Abhängigkeit aufgelöst.

Die *ReceivedMessageManager* Klasse wurde während der Entwicklung gestoppt, da erkannt wurde, dass das Konzept für die Bedürfnisse nicht genügend tief konstruiert war. Bei einer Umsetzung, wie in Abschnitt 9.1 beschrieben, wäre dies zu Dateninkonsistenzen und Redundanzen gekommen. Erste Ansätze wurden betrachtet. Eine Überlegung daraus war ein Konstrukt aus der Kombination vom Singleton und dem Observer-Pattern zu formen. Der Stopp dieses Konstruktes führt dazu, dass die Bibliothek die Aktoren noch nicht mit Daten beliefern kann.

10.3. Implementation - siot.net Sensorcenter

Vom Domänendiagramm (siehe Abbildung 9.8) wurde leicht abgewichen. Die *MessageReceiver* Klasse wurde für Android und Android Wear als lokale, innere Klasse der Activity Klasse realisiert. Dies hat den Vorteil, die Objekte der äusseren Klasse zu kennen und selbst nicht nach Aussen sichtbar ist. Das Kernelement dieser Applikation ist das Kommunikationsverhalten von Smartwatch und Smartphone. Weitere Informationen dazu im Abschnitt Logik.

10.3.1. Klassendiagramm

Abbildung 10.2 stellt klar, wie kompakt die Entwicklung einer externen Applikation sein kann. Sie zeigt alle Klassen der Android und Android Wear App.

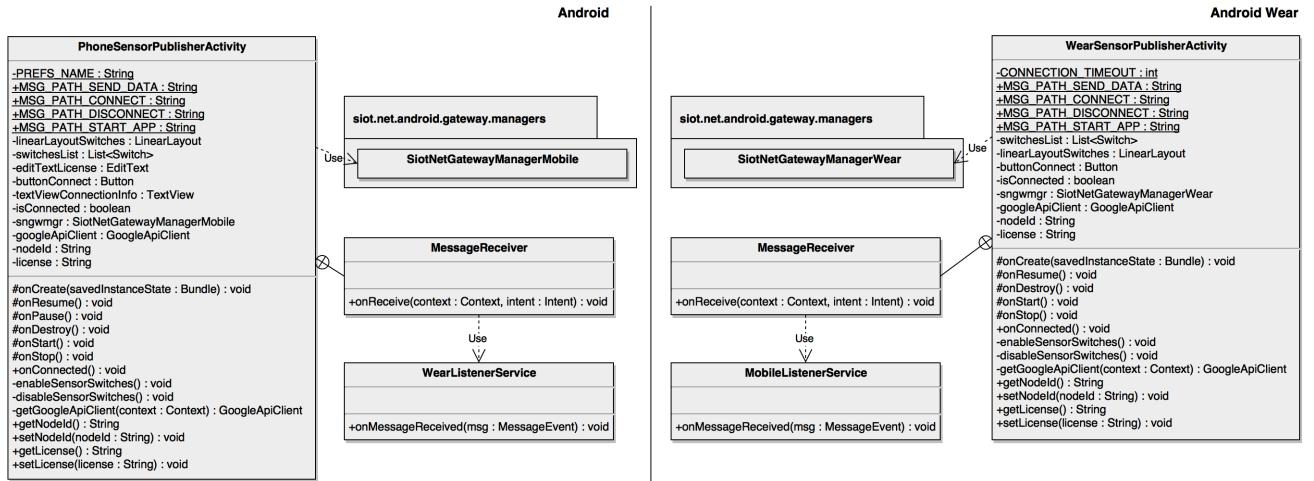


Abbildung 10.2.: Klassendiagramme für das Android und Android Wear Sensorcenter

10.3.2. Logik

Diese Applikationen verwenden die siot.net Gateway Library und deren Logik. Erweiterte logische Elemente gibt es für die Handhabung der Verbindung von Smartwatch und Smartphone und das Verhalten der grafischen Oberfläche.

Die Kommunikation wurde, wie im Konzept definiert, umgesetzt. Wenn eine Smartwatch die Applikation startet, wird per *GoogleApiClient* ein Kommunikationstunnel zum Smartphone hergestellt. Dabei wird gleich die erste Nachricht via MessageAPI versendet, mit der Aufforderung, das Sensorcenter auf dem Smartphone zu öffnen. Die Klasse *WearListenerService* empfängt alle Messages, welche durch diesen eröffneten Kanal gesendet werden. Auch wenn die App noch nicht läuft, kann die Nachricht verarbeitet werden. Das ist möglich, weil der Listener als Service im Android System registriert wird. Zu erreichen ist dies durch einen Eintrag im AndroidManifest.

Nachdem die Applikation läuft, kann die Verbindung zu siot.net aufgebaut werden. Für die Wear App ist es notwendig, dass die Mobile App auch läuft. Der Aufbau der Kommunikationsleitung erfolgt, wie auf Abbildung 9.6 (Smartphone) und Abbildung 9.7 (Smartwatch) illustriert. Die Android Wear Anwendung sendet hier eine CONNECT MessageAPI Nachricht ans Android Gerät. Wenn die Leitung zu siot.net noch nicht steht, wird diese aufgebaut und eine positive Antwort an die Uhr verschickt. Nach erfolgtem Verbindungsaufbau, werden die Ein- und Ausschalter der Sensoren freigegeben. Es werden nur jene angezeigt, welche für das Gerät relevant sind.

Beim Abmelden verhält es sich ein wenig differenziert. Die Smartwatch meldet sich nur beim Smartphone als Sensorcenter ab. Es geschehen keine weiteren Aktionen. Durch die Mobile App wird die Verbindung zu siot.net instand gehalten. Dies wird so gehandhabt, da das Sensorcenter des Smartphones in der Zwischenzeit auch zu Verwendung kommen könnte. Das Smartphone löst beim Abmelden eine DISCONNECT Nachricht, an die verbundene Smartwatch, aus. Es signalisiert, dass keine Verbindung mehr zum siot.net MQTT Broker besteht. Das Wearable reagiert umgehend und stoppt die Messungen und zeigt an, dass es nicht mehr verbunden ist.

Messwerte der Uhr können nicht direkt ins Internet verbreitet werden, weshalb das Smartphone diese per MessageAPI als WearableData Messagetypr erhält. Diese können nun mit Hilfe der öffentlichen Methode *publishData(topic, data)* der SiotNetGatewayManagerMobile Instanz, an den MQTT Broker publiziert werden.

Aufgrund nicht lange ausdauernden Akkus, sind Android Wear Apps für den Betrieb im Hintergrund nicht geeignet. Normalerweise wird die laufende Smartwatch Applikationen beendet, wenn das Display in den Standby-Modus geht. Um bei Verdunklung des Bildschirmes die Arbeit des Sensorcenters nicht zu unterbrechen, muss das Gerät eines der folgenden Funktionen unterstützen: Always-On Screen oder den Ambient Mode. Ersteres lässt die Uhr nie in

den Ruhezustand kehren. Das Zweite lässt die Applikation laufen und verfinstert den Touchscreen, um den Akku zu schonen. Um beide Methoden zu aktivieren, muss dies in der *WearSensorPublisherActivity* implementiert werden.

10.3.3. MessageAPI Messagetypen

START_APP	Nachricht von Smartwatch zu Smartphone, dass App gestartet werden soll. Wird bei Erfolg quittiert.
CONNECT	Nachricht von Smartwatch zu Smartphone, dass die Verbindung zu siot.net aufgebaut werden soll. Wird bei Erfolg mit ok oder bei Misserfolg mit einer DISCONNECT Nachricht quittiert.
DISCONNECT	Nachricht von Smartwatch zu Smartphone, dass die Verbindung nicht mehr benötigt wird. Smartphone quittiert mit ok. Nachricht von Smartphone zu Smartwatch, dass die Verbindung zu siot.net geschlossen wird. Smartwatch quittiert mit ok und schliesst die Verbindung.
SEND_DATA	Nachricht von Smartwatch zu Smartphone. Daten werden im <i>WearableData</i> Messagetyper der siot.net Gateway Library gesendet. Nachricht wird nicht quittiert.

Tabelle 10.1.: siot.net Sensorcenter: MessageAPI Messagetypen

10.3.4. Graphical User Interface

Das GUI beider Softwareartefakte ist sehr simpel und geradlinig gehalten, da es hauptsächlich zu Demonstrationszwecke dienen soll.

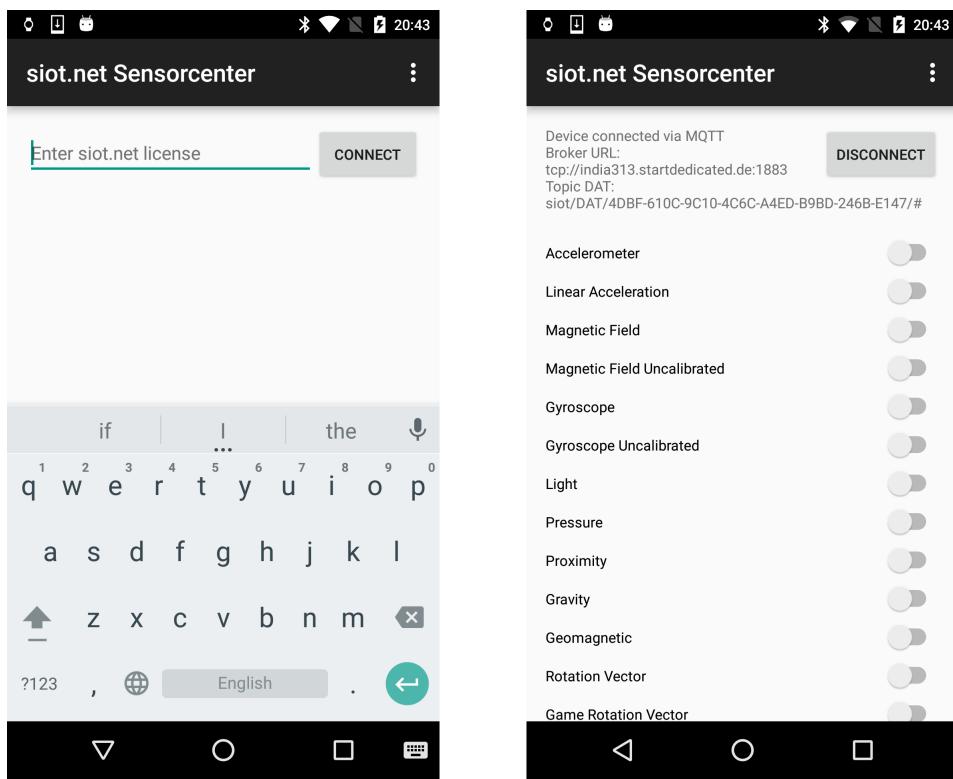


Abbildung 10.3.: Die möglichen Ansichten des Sensorcenter eines Smartphones

Die Android Mobile App ist eine schlanke, zweisegmentale Ansicht. Oben das Feld mit der Texteingabe oder dem Informationsbereich und dem Knopf. Die Abbildung 10.3 zeigt, dass das untere Teilstück die Schalter für die Sensoren darstellt, wenn der Benutzer eingeloggt ist.

Zur Verwendung gekommene GUI Elemente sind: Textfeld, Informationstextfeld, Knopf und mehrere Ein- und Ausschalter. Für Benachrichtigungen werden Toast am unteren Teil des Bildschirms eingeblendet.

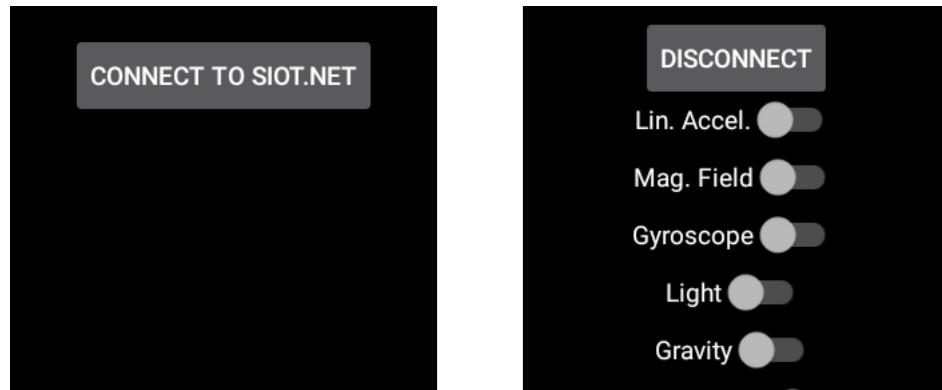


Abbildung 10.4.: GUI Elemente des Smartwatch Sensorcenters

Das GUI der Smartwatch Anwendung ist noch minimalistischer, dies visualisiert die Abbildung 10.4. Der Touchscreen der Uhr ist eher klein, infolge dessen werden nur die wichtigsten Informationen angezeigt. Aus diesem Grund wird nur ein Knopf und die Ein- und Ausschalter für die einzelnen Sensoren eingeblendet. Die grafische Benutzeroberfläche ist so implementiert, dass es im Always-On Display Modus, sowie Ambient Mode funktioniert. Ein Aspekt, der bei der Realisierung des GUIs beachtet werden musste, waren die unterschiedlichen Formen, sowie den Spezialitäten wie ein unregelmäßiges Display. Die zwei Test-Smartwatch waren nicht vom gleichen Typ. Die LG G Watch hat ein rechteckiges Display, die Motorola Moto 360 ein Rundes mit einem abgeflachten Ende.

10.4. Plattformen und Libraries

Für die Entwicklung der Android, Android Wear Bibliothek und App, wurde ausschließlich Android Studio verwendet. Die Abbildung 10.5 zeigt, dass die Entwicklungsumgebung einige Abhängigkeiten hat. Für die eingesetzten Sprachen Java und XML wurde Android Studio (A) genutzt, welche auf der IntelliJ IDEA Plattform basiert. Es bietet Gradle als flexibles Build-Management-Automatisierungs-Tool. Dies nutzt eine Groovy(B) basierende domänenpezifische Sprache (DSL) zur Beschreibung der zu erstellenden Artefakte. Die benutzten Bibliotheken werden in dieser Beschreibungsdatei erfasst. Libraries können in einem Lokalen Verzeichnis(D) oder in einer globalen Ablage, z.B. jCentral (vgl. [1])(C), referenziert sein.

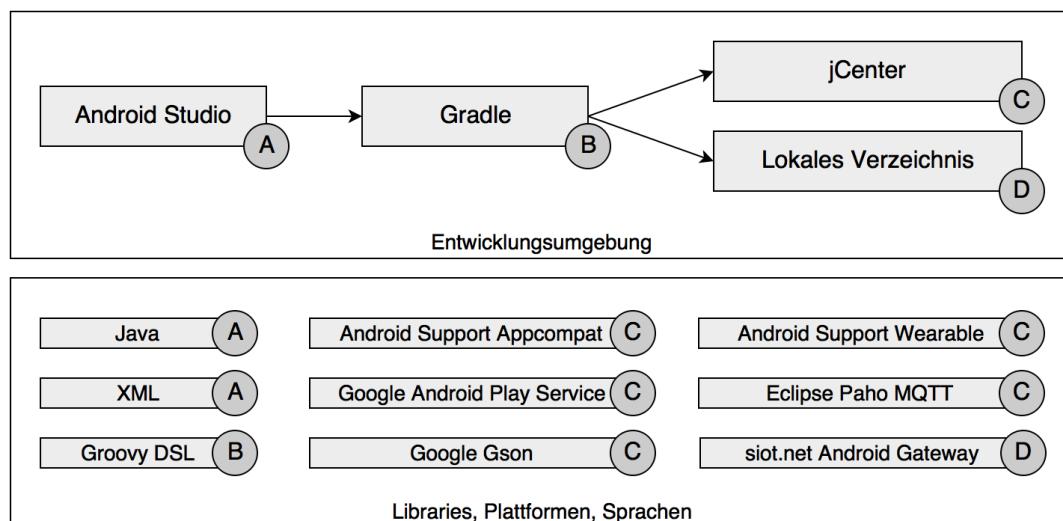


Abbildung 10.5.: Genutzte technische Elemente und deren Abhängigkeiten

10.4.1. Google Play Services

Die Google Play Services sind ein proprietärer Hintergrund Service und eine Applikationsschnittstelle (API) für Android Geräte. Die verwendeten MessageAPI und GoogleClientApi stammen aus dieser Sammlung.

10.4.2. Google Gson

Diese Bibliothek wird verwendet um die Messagetypen von einem POJO in ein JSON zu konvertieren.

10.4.3. Eclipse Paho MQTT

Das Paho Projekt (vgl. [11]) von Eclipse, liefert die Client Implementation des MQTT Nachrichtenprotokoll. Für die Implementation des Nachrichtenaustausches zu siot.net, kommt diese zum Einsatz.

10.5. Dokumentation

Der gesamte Quelltext ist dokumentiert und als JavaDoc auf dem digitalen Anhangsmedium abgelegt. Zusätzlich ist ein Entwicklerhandbuch geschrieben worden, welches im Anhang zu finden ist.

11. Ergebnis

11.1. Sensormessungen mit dem siot.net Sensorcenter

Um das Ergebnis beider entwickelten Programmarteakte zu testen, wurden einige Messungen mit dem siot.net Sensorcenter getätigt, welcher die siot.net Gateway Library integriert. Als Messinstrument wurde der Herzfrequenzmesser der Motorola Moto 360 Smartwatch verwendet. Für die Herzfrequenzmessungen sind drei verschiedene Testpersonen gewählt worden. Für erste Tests wurden die Daten, auf der Abbildung 11.1 visualisierte Node-RED Instanz, manuell ausgewertet. Sobald das siot.io Dashboard verfügbar war, wurden alle Test nochmals Durchgeführt, um die Messwerte dort zu analysiert.

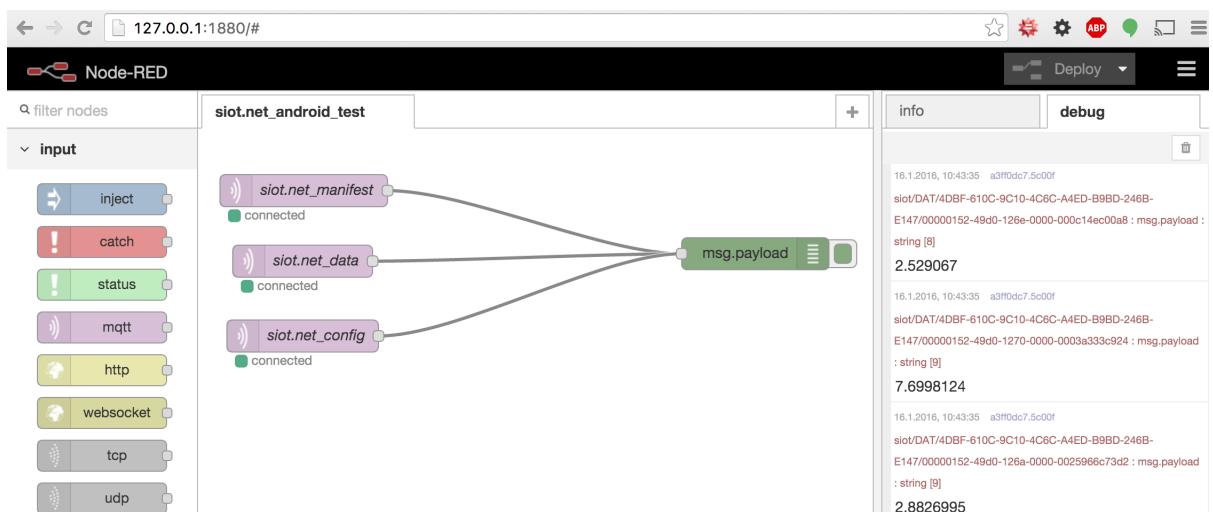


Abbildung 11.1.: Verwendete Node-RED Fluss zum Testdaten empfangen

11.1.1. Voraussetzungen

Es wurden drei Testpersonen gewählt, zwei männliche und eine weibliche. Eine männliche und die weibliche Person sind im frühen Erwachsenenalter (18-35), die zweite männliche Person ist im mittleren Alter (35-65), diese Person weist eine Arrhythmie auf. Die Messungen wurden jeweils sitzend ohne körperliche Anstrengung durchgeführt.

11.1.2. Testresultate

Bei der ersten und zweiten Testperson sind die Messwerte kontinuierlich gemessen und übermittelt worden. Dies ist auf der Node-RED Instanz und auf dem siot.io Dashboard Graphen ersichtlich. Genau so ist zu erkennen, dass bei der dritten Person kein Puls gemessen werden konnte.

11.1.2.1. Testperson 1

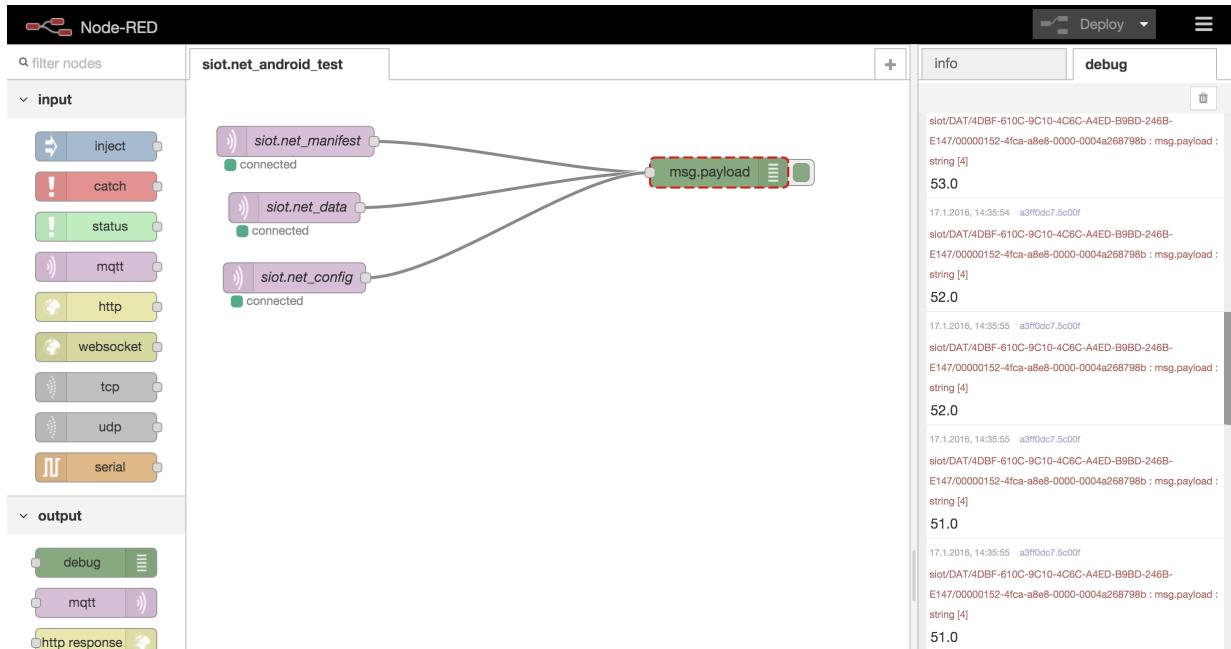


Abbildung 11.2.: Messwerte der ersten Testperson, ersichtlich in Node-RED



Abbildung 11.3.: Der Herzfrequenzgraph auf siot.io der Testperson 1

11.1.2.2. Testperson 2

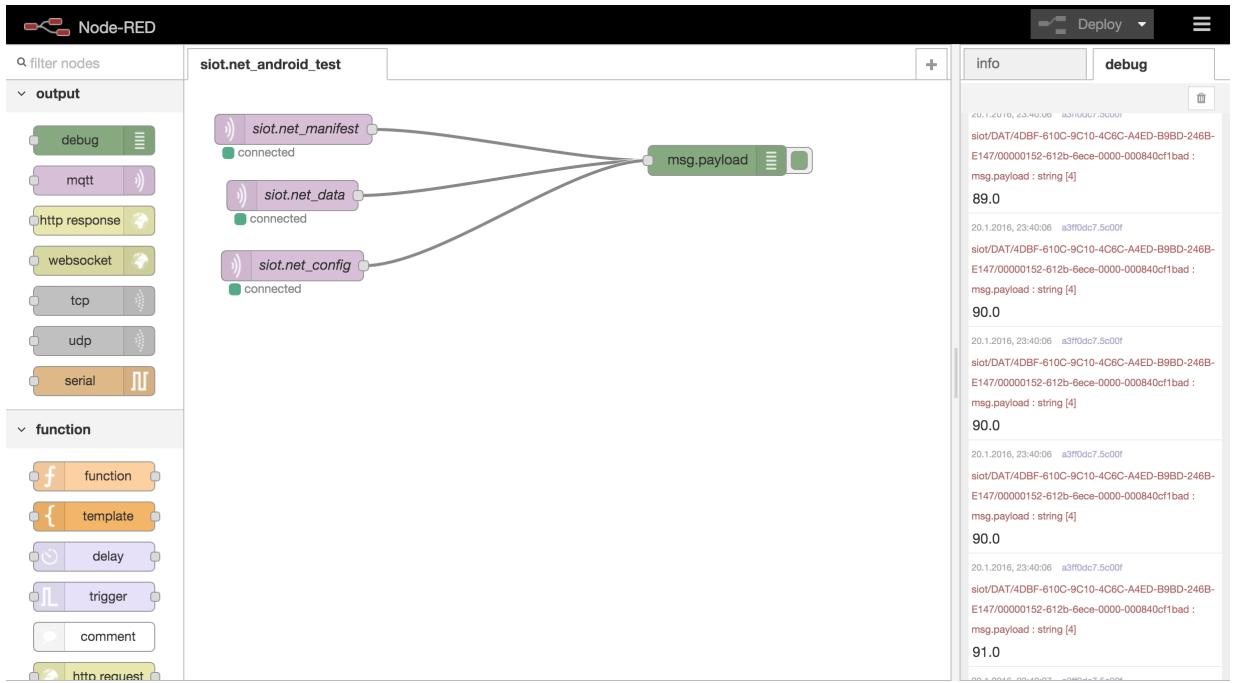


Abbildung 11.4.: Messwerte der zweiten Testperson, ersichtlich in Node-RED



Abbildung 11.5.: Der Herzfrequenzgraph auf siot.io der Testperson 2

11.1.2.3. Testperson 3

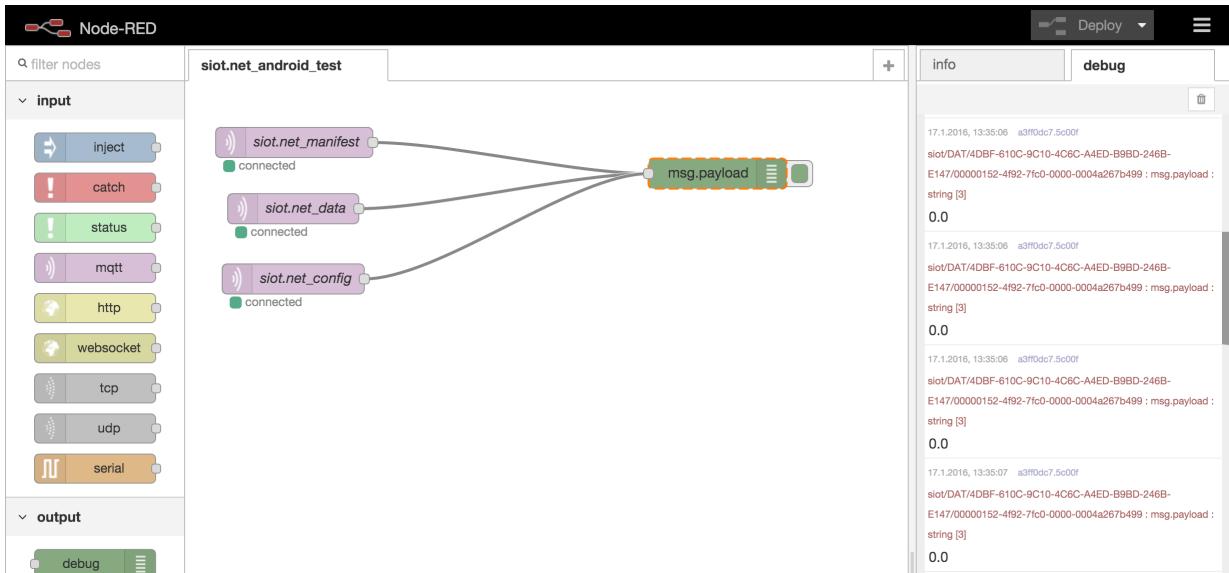


Abbildung 11.6.: Messwerte der dritten Testperson, ersichtlich in Node-RED



Abbildung 11.7.: Der Herzfrequenzgraph auf siot.io der Testperson 3

11.2. Beurteilung

Die Tests wurden mehrmals durchgeführt, außer bei der dritten Testperson. Das Ergebnis bestätigte, dass die Werte immer im ähnlichen Bereich befanden. Jedoch ist sie nicht immer zuverlässig. Es konnte beobachtet werden, dass wenn der Arm stark bewegt wurde, der Herzfrequenzmesser den Puls nicht mehr korrekt berechnen konnte. In diesen Fällen versendet die Smartwatch eine lange Periode den zuletzt gemessenen Wert. Nach der Rekalibrierung, was meist (in über 60% der Fälle) automatisch geschieht, wurden wieder korrekte Messwerte empfangen.

Die korrekt gemessenen Werte, weisen eine kleine Abweichung zu einem Brustgurt auf. Der Differenzwert betrug durchschnittlich -4.5 Herzschläge. Dies wurde in einem Selbsttest, mit einem Brustgurt der Marke Kettler und einem Sportgerät desselben Herstellers, verifiziert. Bei Personen (z.B Testperson 3), welche an einer Herzrhythmusstörung leiden, konnte gar keinen Puls ermittelt werden. Aus diesem Grund wurde dieser Test beim dritten Kandidaten abgebrochen.

Der Herzfrequenzmesser berechnet die Puls Rate aus einer Summe von Messungen in einem bestimmten Intervall. Bei der Unregelmäßigkeit des Pulses konnte die Smartwatch keine zuverlässigen Werte messen, was zu einem Ausfall führte. Damit kann gesagt werden, dass der Pulsmesser bei Personen mit rhythmischer Herzfunktion zufriedenstellende Messwerte liefert. Jedoch bei Personen mit Arrhythmie kann dieser keine Messungen durchführen.

12. Fazit

Zum Abschluss der Bachelorthesis erfolgt eine Schlussfolgerung, welche die Arbeitsweise und die Erarbeitung der Ziele zusammenfassend erläutert. Des Weiteren gibt es einen kurzen Ausblick, in welche Richtung das Projekt weiterverlaufen wird.

12.1. Schlussfolgerung

Das Ziel dieser Arbeit war Smartwatches im Allgemeinen zu betrachten, vorhandene und mögliche Marktsegmente aufzuzeigen und eine grobe Bedürfnisanalyse durchzuführen. Zusätzlich war das spezifizieren und implementieren einer generischen Architektur, für die Anbindung von Smartphones und Smartwatches an die siot.net-Plattform, zu erreichen.

Unter Einfluss des McKinsey Report (siehe [7]) vom Juni 2015 wurden Gruppen ausgesucht. Diese dienten für die Definition der Marktsegmente und die Erarbeitung der Bedürfnisanalyse. Dabei wurde ersichtlich, dass eine Zielgruppe bereits sehr prominent ist. Smartwatches decken vor allem Bedürfnisse ab, welche mit der Überwachung des Menschen in Verbindung sind. Zusätzlich wurde klar, die Computeruhren stecken nach wie vor in den Kinderschuhen. Stand Januar 2016 gibt es noch keine Uhr oder Applikation, welche ein allgemeines Bedürfnis deckt. Trotz dieser Erkenntnis, wurde auch viel Potenzial gefunden, wie z.B. simultan Steuerung oder Smart Home Anwendungen.

Es wurden Smartwatches evaluiert, welche für einen allgemeinen Einsatz geeignet waren. Es führte zu der Technologieauswahl von Android. Nun wurde die Smartwatches und Smartphones, mit dem Betriebssystem von Google, ein Konzept erarbeitet, sie an die siot.net-Plattform anzubinden.

Um diese zu erreichen, musste die siot.net-Umgebung studiert und verstanden werden. Zusätzlich kam die Schwierigkeit dazu, dass die gesamte siot.net-Plattform, zu Beginn der Thesis, erst ein Konzept war. Die siot.net-Plattform befindet sich weiterhin im Aufbau. Infrastruktur für Testumgebungen mussten zuerst selber geschaffen werden.

Aus dem Anforderungsdokument und dem Konzept ist zu erkennen, dass eine generische Bibliothek konzipiert wurde. Diese wird Android und Android Wear Entwicklern die Möglichkeit geben, ihre Applikationen mit schnellen und einfachen Mitteln an die siot.net-Plattform zu koppeln. Um ein solches Konstrukt zu erstellen, war das definieren von Architekturen unerlässlich. Ein Kernpunkt der Architektur ist das Kommunikationsverhalten. Dort wird abgehendelt wie die Smartwatch ins Internet kommunizieren kann.

Mit Hilfe der spezifizierten Architektur ist die siot.net Gateway Library für Android System in einer Prototyp-Version realisiert worden. Diese implementiert das siot-Interface(siehe [13]). Zusätzlich kann die Verwaltung der Sensoren vollständig der Bibliothek übergeben werden. Beim Implementieren ist bedauerlicherweise auch ein Konzeptionsproblem entdeckt worden, was dazu führte, dass Aktoren noch nicht mit Daten angereichert werden können.

Um diese generische Lösung zu verifizieren wurde eine Android und Android Wear App entwickelt: Das siot.net Sensorcenter. Diese erlaubt, ausgewählten Sensoren, Messdaten an die IoT-Plattform zu senden. Bei der Realisierung der Applikation wurde ein Kommunikationskonzept, für den Datentransfer zwischen Smartphone und Smartwatch, entwickelt. Um die Applikation zu testen, wurde ein Sensortyp, der verwendeten Smartwatch, genauer angeschaut. Dazu wurden Messungen durchgeführt, welche auf der siot.io analysiert und ausgewertet wurden.

Das persönliche Fazit. Von der Bachelorthesis wurde erhofft, dass die Erfahrung von Konzeptionierungen von Architekturen im IoT-Bereich erweitert und die allgemeine Kenntnisse im Android Umfeld ausgebaut werden können.

Die Entwicklung mit der Android und Android Wear Umgebung hat gezeigt, wie diese miteinander funktionieren. Zusätzlich wurde die Erkenntnis erlangt, dass es für Android Wear Smartwatches keine direkte Möglichkeit gibt, um mit dem Internet zu kommunizieren. Dies führte zur Einbuße, dass Smartwatches mit der siot.net Gateway Library auch mit Hilfe des Smartphones Daten austauschen müssen. So konnte die gesamte Kommunikationsarchitektur, für die Android Anbindung, selber spezifiziert werden.

Diese Erkenntnis zeigt leider auf, dass der Hersteller Google eine Abhängigkeit zu ihnen geschafft hat. Dies hat für sie den Vorteil, dass sie unter Umständen die Daten, welche von der Smartwatch gesendet werden, auch kennen. Dies ist nicht sehr zufriedenstellend. In einer möglichen anderen Arbeit, könnte ein Aspekt erarbeitet werden, welche ermöglicht diese Barriere zu umgehen.

Bei der Umsetzung konnte erweitertes Know-how gewonnen werden, wie eine Erst-Implementation eines industriellen Projektes verläuft. Durch die Mitarbeit beim siot.net Projekt, in Kooperation mit den Entwicklern von AppModule, konnten interessante neue Coaching-Kenntnisse erlangt werden, da es ein internationales Projekt ist.

Durch diese Erfahrungswerte, konnten die persönlich gesetzten Ziele, zur vollsten Zufriedenheit erreicht werden.

12.2. Ausblick

Im Rahmen dieser Bachelorthesis wurde ein Prototyp der siot.net Gateway Library erstellt und die siot.net Sensorcenter App realisiert. Beide Teile der Umsetzung sind für die Android-Plattform getätigten worden.

Um die Bibliothek in eine Release-Version zu heben, müssen noch einige Punkte nachgebessert werden. Die genaue Konzeptionierung des Aktorhandlings, welches in der Implementationsphase gestoppt wurde, will neu aufgegelistet sein. Des weiteren muss der Anwendungsfall vom Remote-Konfiguration von Android Sensoren aufgegriffen werden. Sicherheits- und Stabilitätsaspekte waren in dieser Arbeit nicht vorgesehen. Für die Kommunikation zu siot.net ist es unweigerlich, dass Lasttests durchgeführt werden. Wichtig ist auch ein Sicherheitskonzept zu erarbeiten. Ein Release dieser Bibliothek würde die siot.net-Plattform für Android Entwickler interessant machen, da sie dadurch eine Menge Zeit sparen können.

In der siot.net-Plattform steckt sehr viel Potenzial. Je mehr Geräte an diese IoT-Umgebung angebunden werden können, desto populärer wird es für die Anwender. Aus diesem Grund wird auch das Projekt der Android Bibliothek weitergeführt.

Selbständigkeitserklärung

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelorthesis selbstständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Name, Vorname
.....

Datum
.....

Unterschrift
.....

Glossar

API Application Programming Interface oder Anwendungsprogrammierschnittstelle.

BLE Bluetooth Low Energy.

EKG ElektroKardioGramm.

GHz Giga-Hertz.

GPS Global Positioning System.

GSM Global System for Mobile Communications, früher Groupe Spécial Mobile.

GUI Graphical User Interface.

GUID Globally Unique IDentifier.

HTTP HyperText Transfer Protocol.

IoT Internet of Things oder Internet der Dinge.

JSON JavaScript Object Notation.

KB KiloByte.

LED Light-Emitting Diode oder Licht-Emittierende Diode.

LTE Long Term Evolution.

M2M Maschine-zu-Maschine.

MB MegaByte.

MEMS Micro-Electromechanical Systems.

MQTT Message Queue Telemetry Transport.

NFC Near Field Communication.

RAM Random-Access Memory.

REST Representational State Transfer.

RFID Radio-Frequency IDentification.

sDB siot.net DashBoard.

SDK Software Development Kit.

sGWLib siot.net GateWay Library.

SC-A siot.net SensorCenter Android.

SC-AW siot.net SensorCenter Android Wear.

UMTS Universal Mobile Telecommunications System.

URL Uniform Resource Locator.

WiFi Wireless Fidelity oder WLAN.

WLAN Wireless Local Area Network.

XML eXtensible Markup Language.

Literaturverzeichnis

- [1] J. Bintray, "jCenter," [Stand 23.12.2015]. [Online]. Available: <https://bintray.com/bintray/jcenter>
- [2] A. Developers, "Managing Projects from Android Studio," [Stand 18.12.2015]. [Online]. Available: <http://developer.android.com/sdk/installing/create-project.html>
- [3] R. H. J. V. E. Gamma, R. Johnson, *Design Patterns*. Addison-Wesley Professional, 1994.
- [4] J. Eason, "An update on Eclipse Android Developer Tools," [Stand 12.12.2015]. [Online]. Available: <http://android-developers.blogspot.ch/2015/06/an-update-on-eclipse-android-developer.html>
- [5] N. Heidloff, "What is Node-RED," [Online; Stand 20.11.2015]. [Online]. Available: <http://heidloff.net/article/21.01.2015081841NHEAL8.htm>
- [6] HiveMQ, "How to get started with MQTT," [Stand 20.11.2015]. [Online]. Available: <http://www.hivemq.com/blog/how-to-get-started-with-mqtt>
- [7] M. G. Institut, "The Internet of Things: Mapping the value beyond the hype," 2015. [Online]. Available: http://www.mckinsey.com/~media/McKinsey/dotcom/Insights/Business%20Technology/Unlocking%20the%20potential%20of%20the%20Internet%20of%20Things/Unlocking_the_potential_of_the_Internet_of_Things_Full_report.ashx
- [8] S. O. Kwok, "Samsung Gear S2 im Test," [Stand 12.12.2015]. [Online]. Available: <https://www.androidpit.de/samsung-gear-s2-test>
- [9] E. Ludewig, "Studienband smartwatch umfrage," 2015, [Veröffentlicht am Juni 2015].
- [10] Mercedes-Benz, "Mercedes-Benz F 015," [Stand 05.11.2015]. [Online]. Available: <https://www.mercedes-benz.com/de/mercedes-benz/innovation/forschungsfahrzeug-f-015-luxury-in-motion/>
- [11] E. Paho, "MQTT client," [Stand 23.12.2015]. [Online]. Available: <http://www.eclipse.org/paho/>
- [12] M. Rouse, "Internet der Dinge - Definition," [Stand 23.10.2015]. [Online]. Available: <http://www.searchnetworking.de/definition/internet-der-Dinge-Internet-of-Things-IoT>
- [13] siot.net, "CookBook siot.net release 1," 2015.
- [14] statista, "Prognose zu den Marktanteilen der Betriebssysteme am Absatz vom Smartphones weltweit in den Jahren 2015 und 2019," [Stand 10.01.2016]. [Online]. Available: <http://de.statista.com/statistik/daten/studie/182363/umfrage/prognostizierte-marktanteile-bei-smartphone-betriebssystemen/>
- [15] S. Warentest, "Smartwatches," 2015, [Ausgabe 10/15].
- [16] C. Weiss, "Die Sache mit dem Horizont," [Stand 05.11.2015]. [Online]. Available: <https://www.holisticon.de/2014/11/die-sache-mit-dem-horizont-iot-blogserie-episode-5/>
- [17] Wikipedia, "Smarthome," [Stand 05.11.2015]. [Online]. Available: https://de.wikipedia.org/wiki/Smart_Home

Abbildungsverzeichnis

2.1. MQTT Architektur	4
2.2. siot.net IoT-Center	5
3.1. Fussgängererkennung des Mercedes-Benz F 015	7
3.2. Solar Roadway, dynamische Strasse	8
4.1. Mobile Zahlungslösungen: Apple Pay, Google Wallet und TWINT powered by PostFinance	13
4.2. Smartphone gesteuerte Drohne: Parrot bebop	14
7.1. Gesamtarchitektur zu siot.net	23
7.2. Übersicht Android Wear Paket Architektur	24
7.3. Übersicht siot.net Applikationsarchitektur	24
7.4. Geplante Netzwerk-Architektur mit URL/ohne WLAN	25
7.5. Geplante Netzwerk-Architektur ohne BLE/mit WLAN	26
7.6. Effektive Netzwerk-Architektur mit BLE/ohne WLAN	27
7.7. Effektive Netzwerk-Architektur ohne BLE/mit WLAN	28
8.1. Use Case siot.net Gateway Library	29
8.2. Use Case siot.net Sensorcenter	36
8.3. Use Case siot.net Sensorcenter	41
9.1. siot.net Gateway Library Packagediagramm	46
9.2. siot.net Gateway Library Domäendiagramm	47
9.3. siot.net Gateway Library Sequenzdiagramm App starten 1	48
9.4. siot.net Gateway Library Sequenzdiagramm App starten 2	48
9.5. siot.net Gateway Library Sequenzdiagramm App starten 3	49
9.6. siot.net Gateway Library Sequenzdiagramm Login 1	49
9.7. siot.net Gateway Library Sequenzdiagramm Login 2	50
9.8. siot.net Sensorcenter Domäendiagramm	50
9.9. siot.net Sensorcenter Sequenzdiagramm Sensor starten 1	51
9.10. siot.net Sensorcenter Sequenzdiagramm Sensor starten 2	52
10.1. siot.net Gateway Library Klassendiagramm	54
10.2. siot.net Sensorcenter Klassendiagramm	56
10.3. siot.net Sensorcenter Smartphone Screens	57
10.4. siot.net Sensorcenter Smartwatch Screens	58
10.5. Plattforme, Libraries	58
11.1. Node-RED Testflow	60
11.2. Node-RED Messwerte Person 1	61
11.3. siot.io Dashboard Graph Person 1	61
11.4. Node-RED Messwerte Person 2	62
11.5. siot.io Dashboard Graph Person 2	62
11.6. Node-RED Messwerte Person 3	63
11.7. siot.io Dashboard Graph Person 3	63
B.1. Zeitplan Teil 1	76
B.2. Zeitplan Teil 2	77
B.3. Zeitplan Teil 3	78
B.4. Kanban-Board 24.09.2015	79

B.5. Kanban-Board 09.10.2015	80
B.6. Kanban-Board 08.11.2015	81
B.7. Kanban-Board 04.12.2015	82
B.8. Kanban-Board 10.12.2015	83
B.9. Kanban-Board 16.12.2015	84
B.10. Kanban-Board 23.12.2015	85
B.11. Kanban-Board 02.01.2016	86
B.12. Kanban-Board 05.01.2016	87
B.13. Kanban-Board 10.01.2016	88
B.14. Kanban-Board 14.01.2016	89
B.15. Kanban-Board 17.01.2016	90
B.16. Kanban-Board 21.01.2016	91

Tabellenverzeichnis

6.1. Technische Daten von Smartwatches - Teil 1	20
6.2. Technische Daten von Smartwatches - Teil 2	21
8.1. siot.net Gateway Library: Übersicht Anwendungsfall #1	30
8.2. siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #1	30
8.3. siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #1	30
8.4. siot.net Gateway Library: Übersicht Anwendungsfall #2	31
8.5. siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #2	31
8.6. siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #2	31
8.7. siot.net Gateway Library: Übersicht Anwendungsfall #3	32
8.8. siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #3	32
8.9. siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #3	32
8.10. siot.net Gateway Library: Übersicht Anwendungsfall #4	33
8.11. siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #4	33
8.12. siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #4	33
8.13. siot.net Gateway Library: Übersicht Anwendungsfall #5	34
8.14. siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #5	34
8.15. siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #5	34
8.16. siot.net Gateway Library: Übersicht Anwendungsfall #6	35
8.17. siot.net Gateway Library: Chronologischer Ablauf von Anwendungsfall #6	35
8.18. siot.net Gateway Library: Ausnahmen und Varianten von Anwendungsfall #6	35
8.19. siot.net Gateway Library: Nicht-funktionale Produktanforderungen	35
8.20. siot.net Sensorcenter: Übersicht Anwendungsfall #1.1 und #1.2	37
8.21. siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #1.1	37
8.22. siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #1.2	37
8.23. siot.net Sensorcenter: Ausnahmen und Varianten von Anwendungsfall #1.1 und #1.2	38
8.24. siot.net Sensorcenter: Übersicht Anwendungsfall #2.1 und #2.2	38
8.25. siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #2.1 und #2.2	38
8.26. siot.net Sensorcenter: Ausnahmen und Varianten von Anwendungsfall #2.1 und #2.2	38
8.27. siot.net Sensorcenter: Übersicht Anwendungsfall #3.1 und #3.2	39
8.28. siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #3.1 und #3.2	39
8.29. siot.net Sensorcenter: Ausnahmen und Varianten von Anwendungsfall #3.1 und #3.2	39
8.30. siot.net Sensorcenter: Übersicht Anwendungsfall #4.1 und #4.2	39
8.31. siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #4.1	39
8.32. siot.net Sensorcenter: Chronologischer Ablauf von Anwendungsfall #4.2	40
8.33. siot.net Sensorcenter: Ausnahmen und Varianten von Anwendungsfall #4.1 und #4.2	40
8.34. siot.net Sensorcenter: Nicht-funktionale Produktanforderungen	40
8.35. siot.net Dashboard: Übersicht Anwendungsfall #2	42
8.36. siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #2	42
8.37. siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #2	42
8.38. siot.net Dashboard: Übersicht Anwendungsfall #3	42
8.39. siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #3	43
8.40. siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #3	43
8.41. siot.net Dashboard: Übersicht Anwendungsfall #4	43
8.42. siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #4	43
8.43. siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #4	43
8.44. siot.net Dashboard: Übersicht Anwendungsfall #5	44
8.45. siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #5	44
8.46. siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #5	44

8.47. siot.net Dashboard: Übersicht Anwendungsfall #6	44
8.48. siot.net Dashboard: Chronologischer Ablauf von Anwendungsfall #6	44
8.49. siot.net Dashboard: Ausnahmen und Varianten von Anwendungsfall #6	44
8.50. siot.net Dashboard: Nicht-funktionale Produktanforderungen	45
9.1. siot.net Gateway Library: Auflistung der Packageinhalte	46
10.1. siot.net Sensorcenter: MessageAPI Messagetypen	57

A. Entwicklerhandbuch

Diese Entwicklerhandbuch zeigt die ersten Schritte wie die siot.net Gateway Library in die App einzubinden sind.

A.1. Download

Die Android und Android Wear Library im Prototyp-Release, kann unter folgender URL geladen werden:

<https://github.com/paras1/sw-siot/blob/master/dev/SiotNetGateway/library/>

Der offizielle Release wird in der jCentral Ablage verfügbar sein. Somit kann sie in der *build.gradle* Datei, des jeweiligen Modules (Mobile und/oder Wear) eingefügt werden.

A.2. Dokumentation

Die Referenzdokumentation (JavaDoc) kann unter folgender URL heruntergeladen werden:

https://github.com/paras1/sw-siot/tree/master/doc/02_javadoc/01_SiotNetGateway

A.3. Erste Schritte

Die siot.net Gateway Library funktioniert ab der SDK Version 21. Das heisst, sie ist ab Android 5.0 Lollipop kompatibel.

Für die Integration der siot.net Gateway Library muss die siotnetgateway.vX_X.aar Datei im *build.gradle* referenziert werden. Zusätzlich benötigt, diese noch weitere Bibliotheken. Beim Beispiel ist die Library im **libs** Verzeichnis des Projektordner abgelegt. Die zusätzlich, in den *dependencies*, angegebenen Bibliotheken werden zwingend benutzt für das Paket.

Ein Beispiel:

```
// build.gradle (project)
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:x.x.x'
        classpath 'com.google.gms:google-services:x.x.x'
    }
}
allprojects {
    repositories {
        jcenter()
        maven {
            url 'https://repo.eclipse.org/content/repositories/paho-releases/'
        }
        flatDir {
            dirs 'libs'
        }
    }
}
```

```
//build.gradle (mobile)
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    wearApp project(':wear')

    compile 'com.android.support:appcompat-v7:21.x.x'
    compile 'com.google.android.gms:play-services:7.3.0'
    compile 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.0.2'
    compile 'com.google.code.gson:gson:2.4'
    compile(name:'siotnetgateway_vX_X', ext:'aar')
}

//build.gradle (wear)
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.google.android.support:wearable:x.x.x'
    compile 'com.android.support:appcompat-v7:21.x.x'
    compile 'com.google.android.gms:play-services:7.3.0'
    compile 'com.google.android.gms:play-services-wearable:7.3.0'
    compile 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.0.2'
    compile 'com.google.code.gson:gson:2.4'
    compile(name:'siotnetgateway_vX_X', ext:'aar')
}
```

A.4. Entwicklung

Um die Elemente in der Bibliothek zu nutzen, muss ein *SiotNetGatewayManagerMobile(Context)* für Android Geräte oder ein *SiotNetGatewayManagerWear(Context, GoogleClientApi, String)* für Wearables instanziert werden. Der Parameter *Context*, ist der Kontext der Activity, welcher mitgegeben werden muss. *GoogleClientApi* und *String* (Nodeld des GoogleClients) müssen nur bei der Wear Instanz mitgegeben werden. Diese sind notwendig für die Kommunikation zwischen Mobile und Wear Geräte.

```
// MobileActivity.java
...
SiotNetGatewayManagerMobile mobile = new SiotNetGatewayManagerWear(this);
...

// WearActivity.java
...
SiotNetGatewayManagerWear wear =
    new SiotNetGatewayManagerWear(this, googleApiClient, nodeld);
...
```

Der Verbindungsauflauf gelingt mit der persönlichen siot.net Lizenz. Für die Smartwatch ist dies nicht nötig, da diese eine offene Verbindung von Mobile Gerät voraussetzt.

```
// MobileActivity.java
...
mobile.connectToSiotNet(license);
...
```

Mit diesen Schritten kann, die Bibliothek erfolgreich in die App eingebunden werden.
Für weitere Informationen, verwenden Sie bitte die Referenzdokumentation.

B. Projektmanagement

Für die Organisation der Aufgaben dieser Bachelorthesis, wurde ein Zeitplan aufgestellt. Um die Prozesse präzise zu steuern wurden ein Kanban-Board verwendet. Einige Task haben mehrere Kanban-Tickets erhalten, um ein iteratives Vorgehen zu ermöglichen.

B.1. Zeitplan

	Issue	Subtask 1	Subtask 2	Subtask 3	Subtask 4		geplantes Startdatum	geplantes Enddatum
1. Allgemein	Aufgaben Planung						12.09.2015	20.09.2015
	Kanban Board aufsetzen						13.09.2015	20.09.2015
	Dokument aufsetzen für Markt und Bedürfnisanalyse	TeX Dokument					20.09.2015	25.09.2015
	Dokument aufsetzen für System-Architektur	TeX Dokument					20.09.2015	25.09.2015
	Dokument aufsetzen für Technische Dokumentation	TeX Dokument					20.09.2015	25.09.2015
	Ungeplante Aufgaben							
2. Markt und Bedürfnisanalyse	Analyse Marktsegment für Smartwatches in IoT	Recherche	Dokumentation				24.09.2015	25.09.2015
	Analyse Anwendungen für Smartwatches in IoT	Recherche	Dokumentation				25.09.2015	02.10.2015
	Technische Anforderungen für analysierte Anwendungen	Recherche	Dokumentation				30.09.2015	03.10.2015
	Smartwatch Produkte evaluieren für die analysierten Anwendungen	Recherche	Wenn möglich Hands-On	Dokumentation			03.10.2015	06.10.2015
	Analyse - Dokument Review mit Dr. A. Danuser	Review	Nacharbeiten				08.10.2015	09.10.2015

Abbildung B.1.: Zeitplan der Bachelorthesis Teil 1

3.1 System-Architektur	Netzwerk-Architekturen evaluieren	Recherche	Hands-On	Dokumentation			08.10.2015	09.10.2015
	System-Architekturen evaluieren	Recherche	Hands-On	Dokumentation			08.10.2015	09.10.2015
	Software-Architekturen evaluieren	Recherche	Hands-On	Dokumentation			08.10.2015	09.10.2015
	Integration von Softwarestack für siot.net	Implementation	Dokumentation fachlich	Dokumentation technisch			09.10.2015	12.10.2015
	System Architektur - Dokument Review mit Dr. A. Danuser	Review	Nacharbeiten				22.10.2015	23.10.2015
3.2 Anforderungen	Klassen für Anwendungen	Klassen definieren	Klassen auswählen				12.09.2015	12.10.2015
	Rahmenbedingungen für Anwendungen	Rahmenbedingungen definieren pro Klasse					10.10.2015	20.10.2015
	Anforderungen für Anwendungen	Anforderungen Klasse I	Anforderungen Klasse II	Anforderungen Klasse III	Anforderungen Klasse IV		10.10.2015	20.10.2015
	Technische Anforderung an eine Smartwatch für Anwendungen	Anforderungen für Smartwatches definieren	Smartwatches auflisten/ eingrenzen	Smartwatches auswählen			20.10.2015	22.10.2015
	Auswahl von Klassen für Design und Implementation	Klasse I auswählen	Klasse II auswählen	Klasse III auswählen (optional)			20.10.2015	22.10.2015
	Anforderungen - Dokument Review mit Dr. A. Danuser	Review	Nacharbeiten				22.10.2015	23.10.2015
3.3 Design	Software Design Anwendungsfall I	Use-Case	Klassendiagramm	Zusammenfassung			22.10.2015	27.10.2015
	Software Design Anwendungsfall II	Use-Case	Klassendiagramm	Zusammenfassung			27.10.2015	02.11.2015
	Software Design Anwendungsfall III	Use-Case	Klassendiagramm	Zusammenfassung			02.11.2015	06.11.2015
	Auswahl mind. 1 Anwendungsfall für Implementation	Anwendungsfall I auswählen	Anwendungsfall II auswählen	Anwendungsfall III auswählen			06.11.2015	06.11.2015
	Design - Dokument Review mit Dr. A. Danuser	Review	Nacharbeiten				06.11.2015	13.11.2015

Abbildung B.2.: Zeitplan der Bachelorthesis Teil 2

4.1 Implementation	Integration der Software	Iteration I	Iteration II	Iteration III			06.11.2015	07.01.2016
	Testen der Software	Iteration I	Iteration II	Iteration III			06.11.2015	07.01.2016
4.2 Dokumentation	Code Dokumentation						06.11.2015	07.01.2016
	Entwicklerhandbuch						01.01.2016	07.01.2016
	Tech. Dok. - Dokument Review mit Dr. A. Danuser	Review	Nacharbeiten				07.01.2016	08.01.2016
5. Präsentationen	Erstellen der Präsentation für Final Tag						08.01.2016	21.01.2016
	Demo für Final Tag	Plakat für Ausstellung	Demonstration einer App testen				08.01.2016	21.01.2016
	Präsentation für Verteidigung						20.01.2016	25.01.2016
9. Schlussarbeiten	Dokumente abschliessen	Teil I	Teil II	Teil III			08.01.2016	21.01.2016

Abbildung B.3.: Zeitplan der Bachelorthesis Teil 3

B.2. Kanban-Board Verlauf

In diesem Abschnitt ist der Verlauf der Kanban-Tickets ersichtlich.

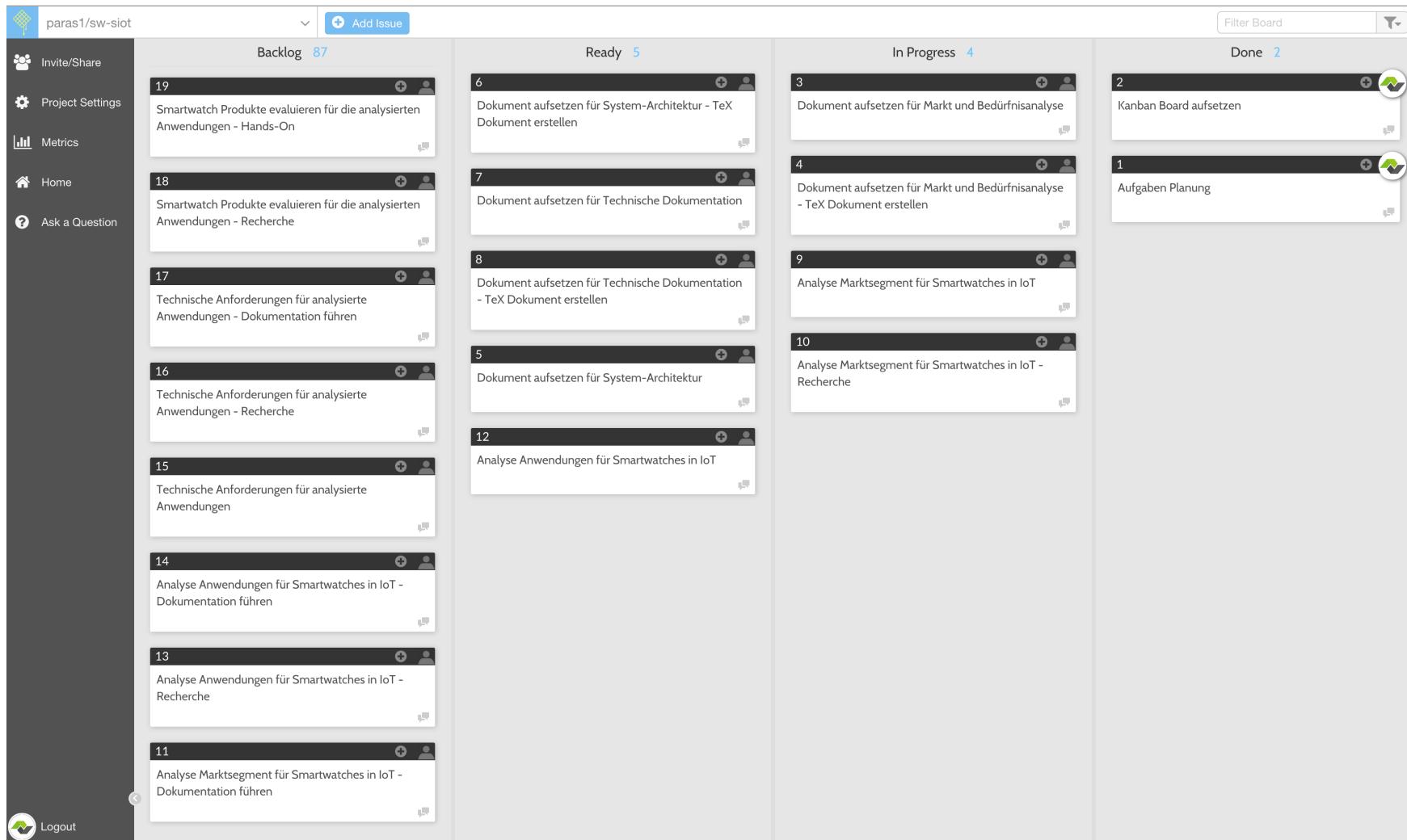


Abbildung B.4.: Kanban-Board Fortschritt am 24.09.2015

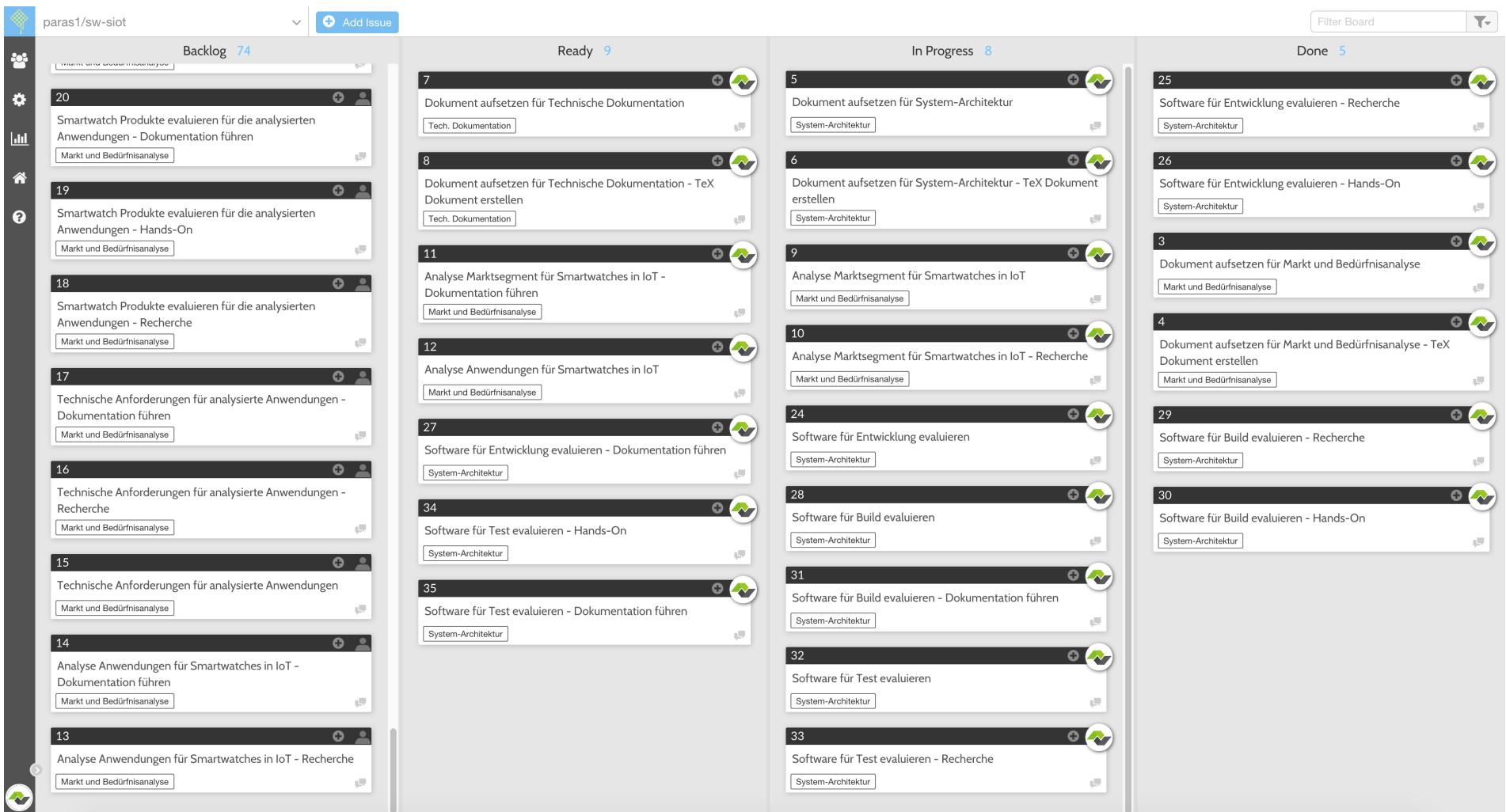


Abbildung B.5.: Kanban-Board Fortschritt am 09.10.2015

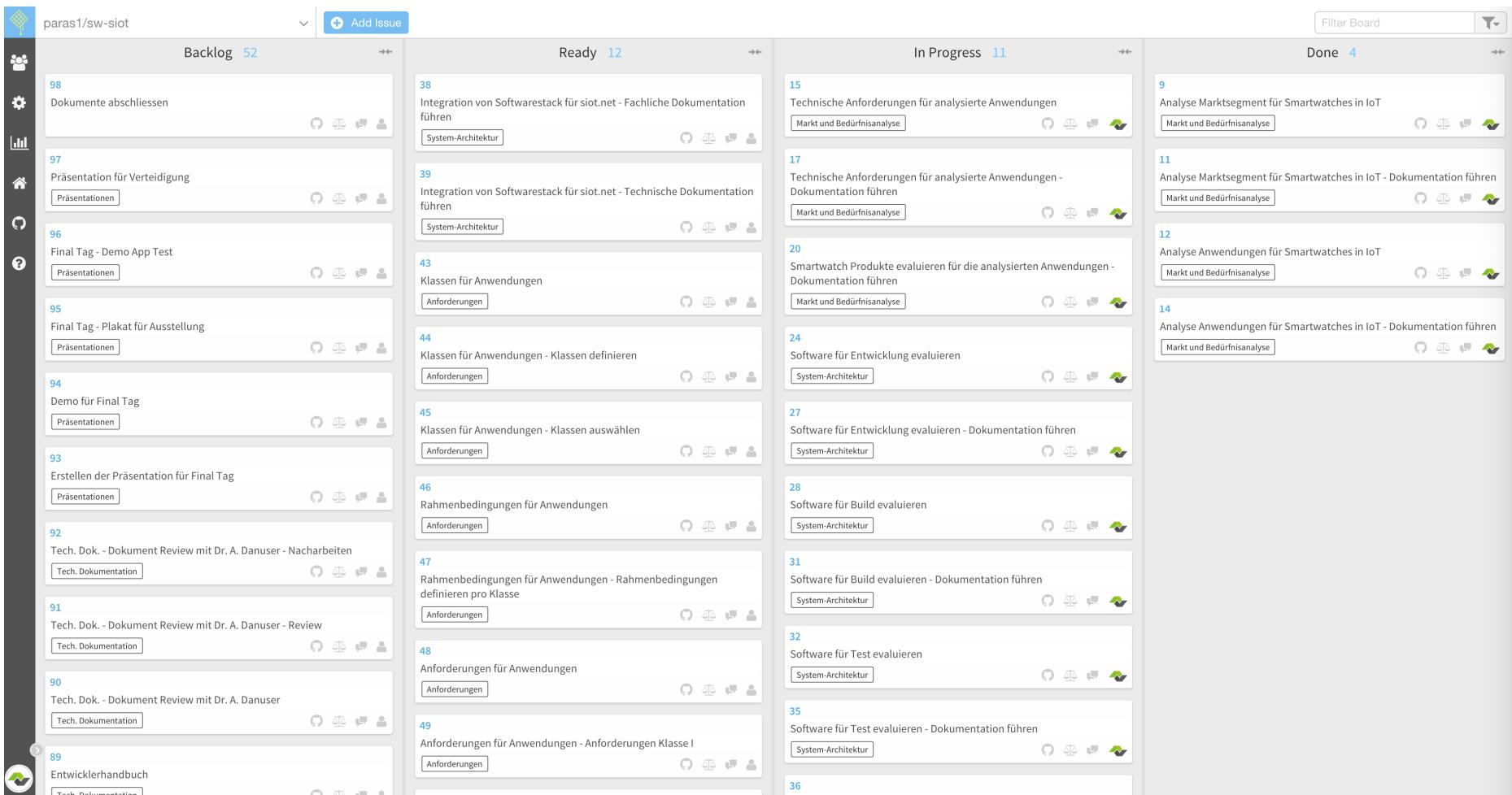


Abbildung B.6.: Kanban-Board Fortschritt am 08.11.2015

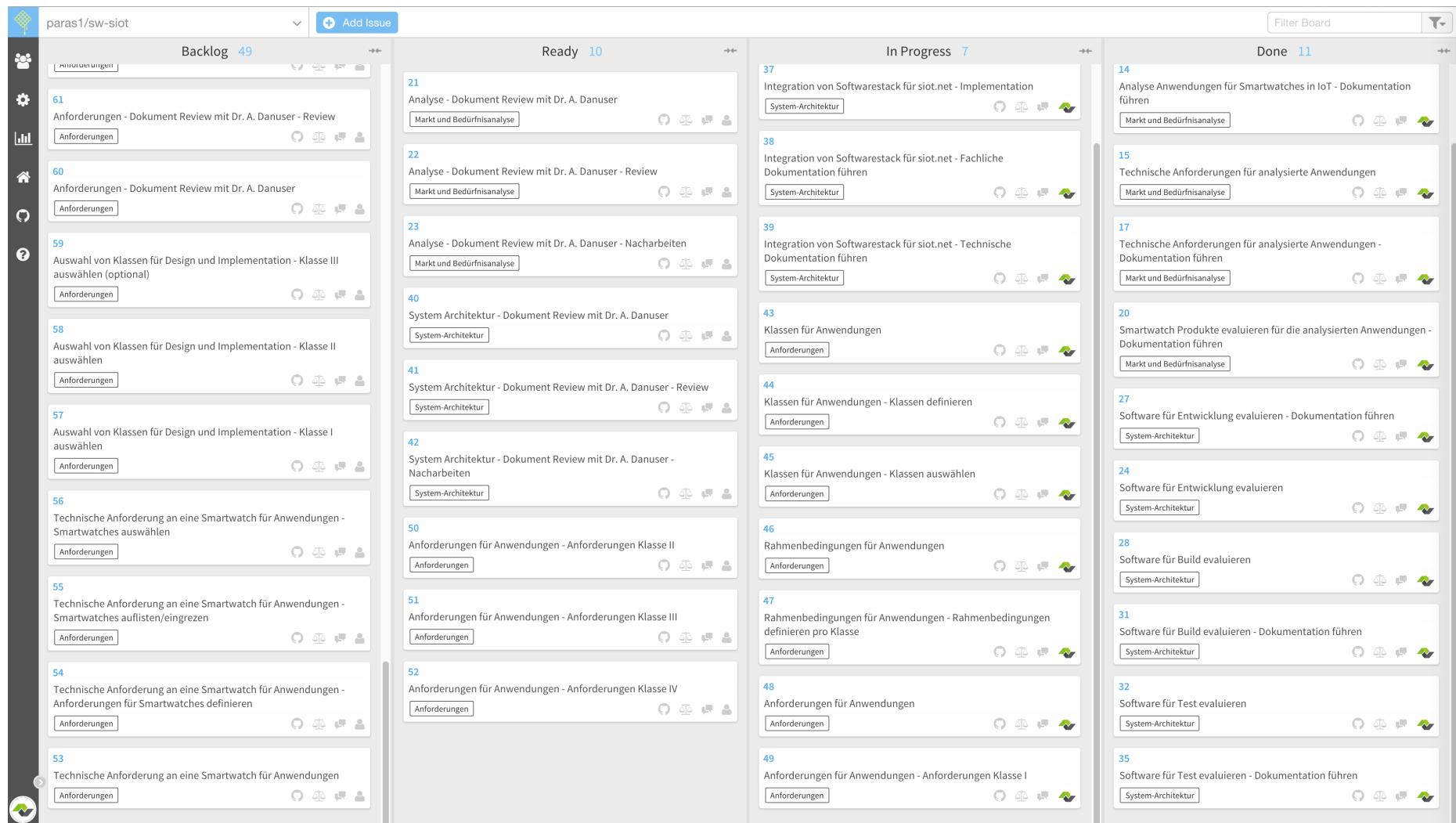


Abbildung B.7.: Kanban-Board Fortschritt am 04.12.2015

Smartwatches in siot.net

Column	Task Number	Description	Status
Backlog	70	Software Design Anwendungsfall III - Use-Case	Design
Backlog	69	Software Design Anwendungsfall II - Zusammenfassung	Design
Backlog	68	Software Design Anwendungsfall II - Klassendiagramm	Design
Backlog	67	Software Design Anwendungsfall II - Use-Case	Design
Backlog	66	Software Design Anwendungsfall I - Zusammenfassung	Design
Backlog	65	Software Design Anwendungsfall I - Klassendiagramm	Design
Backlog	64	Software Design Anwendungsfall I - Use-Case	Design
Backlog	63	Software Design Anwendungsfall	Design
Backlog	62	Anforderungen - Dokument Review mit Dr. A. Danuser - Nacharbeiten	Anforderungen
Backlog	61	Anforderungen - Dokument Review mit Dr. A. Danuser - Review	Anforderungen
Backlog	60	Anforderungen - Dokument Review mit Dr. A. Danuser	Anforderungen
Ready	21	Analyse - Dokument Review mit Dr. A. Danuser	Markt und Bedürfnisanalyse
Ready	22	Analyse - Dokument Review mit Dr. A. Danuser - Review	Markt und Bedürfnisanalyse
Ready	23	Analyse - Dokument Review mit Dr. A. Danuser - Nacharbeiten	Markt und Bedürfnisanalyse
Ready	40	System Architektur - Dokument Review mit Dr. A. Danuser	System-Architektur
Ready	41	System Architektur - Dokument Review mit Dr. A. Danuser - Review	System-Architektur
Ready	42	System Architektur - Dokument Review mit Dr. A. Danuser - Nacharbeiten	System-Architektur
Ready	57	Auswahl von Klassen für Design und Implementation - Klasse I auswählen	Anforderungen
Ready	58	Auswahl von Klassen für Design und Implementation - Klasse II auswählen	Anforderungen
Ready	59	Auswahl von Klassen für Design und Implementation - Klasse III auswählen (optional)	Anforderungen
In Progress	36	Integration von Softwarestack für siot.net	System-Architektur
In Progress	38	Integration von Softwarestack für siot.net - Fachliche Dokumentation führen	System-Architektur
In Progress	39	Integration von Softwarestack für siot.net - Technische Dokumentation führen	System-Architektur
In Progress	53	Technische Anforderung an eine Smartwatch für Anwendungen	Anforderungen
In Progress	54	Technische Anforderung an eine Smartwatch für Anwendungen - Anforderungen für Smartwatches definieren	Anforderungen
In Progress	55	Technische Anforderung an eine Smartwatch für Anwendungen - Smartwatches auflisten/eingrenzen	Anforderungen
In Progress	56	Technische Anforderung an eine Smartwatch für Anwendungen - Smartwatches auswählen	Anforderungen
Done	37	Integration von Softwarestack für siot.net - Implementation	System-Architektur
Done	43	Klassen für Anwendungen	Anforderungen
Done	44	Klassen für Anwendungen - Klassen definieren	Anforderungen
Done	45	Klassen für Anwendungen - Klassen auswählen	Anforderungen
Done	46	Rahmenbedingungen für Anwendungen	Anforderungen
Done	47	Rahmenbedingungen für Anwendungen - Rahmenbedingungen definieren pro Klasse	Anforderungen
Done	48	Anforderungen für Anwendungen	Anforderungen
Done	49	Anforderungen für Anwendungen - Anforderungen Klasse I	Anforderungen
Done	50	Anforderungen für Anwendungen - Anforderungen Klasse II	Anforderungen
Done	51	Anforderungen für Anwendungen - Anforderungen Klasse III	Anforderungen
Done	52	Anforderungen für Anwendungen - Anforderungen Klasse IV	Anforderungen

Abbildung B.8.: Kanban-Board Fortschritt am 10.12.2015

Smartwatches in siot.net

paras1/sw-siot

Add Issue

Filter Board

The Kanban board displays the project's progress across four columns: Backlog, Ready, In Progress, and Done.

- Backlog:** Contains 22 items, including:
 - 89 Entwicklerhandbuch [Tech. Dokumentation]
 - 88 Code Dokumentation [Tech. Dokumentation]
 - 87 Testen der Software - Iteration III [Implementation]
 - 86 Testen der Software - Iteration II [Implementation]
 - 85 Testen der Software - Iteration I [Implementation]
 - 84 Testen der Software [Implementation]
 - 83 Integration der Software - Iteration III [Implementation]
 - 82 Integration der Software - Iteration II [Implementation]
 - 79 Design - Dokument Review mit Dr. A. Danuser - Nacharbeiten [Design]
 - 78 Design - Dokument Review mit Dr. A. Danuser - Review [Design]
 - 77 Design - Dokument Review mit Dr. A. Danuser [Design]
- Ready:** Contains 12 items, including:
 - 22 Analyse - Dokument Review mit Dr. A. Danuser - Review [Markt und Bedürfnisanalyse]
 - 23 Analyse - Dokument Review mit Dr. A. Danuser - Nacharbeiten [Markt und Bedürfnisanalyse]
 - 40 System Architektur - Dokument Review mit Dr. A. Danuser [System-Architektur]
 - 41 System Architektur - Dokument Review mit Dr. A. Danuser - Review [System-Architektur]
 - 42 System Architektur - Dokument Review mit Dr. A. Danuser - Nacharbeiten [System-Architektur]
 - 62 Anforderungen - Dokument Review mit Dr. A. Danuser - Nacharbeiten [Anforderungen]
 - 61 Anforderungen - Dokument Review mit Dr. A. Danuser - Review [Anforderungen]
 - 60 Anforderungen - Dokument Review mit Dr. A. Danuser [Anforderungen]
 - 70 Software Design Anwendungsfall III - Use-Case [Design]
 - 71 Software Design Anwendungsfall III - Klassendiagramm [Design]
 - 72 Software Design Anwendungsfall III - Zusammenfassung [Design]
- In Progress:** Contains 8 items, including:
 - 36 Integration von Softwarestack für siot.net [System-Architektur]
 - 38 Integration von Softwarestack für siot.net - Fachliche Dokumentation führen [System-Architektur]
 - 39 Integration von Softwarestack für siot.net - Technische Dokumentation führen [System-Architektur]
 - 63 Software Design Anwendungsfall [Design]
 - 66 Software Design Anwendungsfall I - Zusammenfassung [Design]
 - 69 Software Design Anwendungsfall II - Zusammenfassung [Design]
 - 73 Auswahl mind. 1 Anwendungsfall für Implementation [Design]
 - 76 Auswahl mind. 1 Anwendungsfall für Implementation - Anwendungsfall III auswählen (optional) [Design]
- Done:** Contains 24 items, including:
 - 56 Technische Anforderung an eine Smartwatch für Anwendungen - Smartwatches auswählen [Anforderungen]
 - 57 Auswahl von Klassen für Design und Implementation - Klasse I auswählen [Anforderungen]
 - 58 Auswahl von Klassen für Design und Implementation - Klasse II auswählen [Anforderungen]
 - 59 Auswahl von Klassen für Design und Implementation - Klasse III auswählen (optional) [Anforderungen]
 - 64 Software Design Anwendungsfall I - Use-Case [Design]
 - 65 Software Design Anwendungsfall I - Klassendiagramm [Design]
 - 67 Software Design Anwendungsfall II - Use-Case [Design]
 - 68 Software Design Anwendungsfall II - Klassendiagramm [Design]
 - 74 Auswahl mind. 1 Anwendungsfall für Implementation - Anwendungsfall I auswählen [Design]
 - 75 Auswahl mind. 1 Anwendungsfall für Implementation - Anwendungsfall II auswählen (optional) [Design]

Abbildung B.9.: Kanban-Board Fortschritt am 16.12.2015

Smartwatches in siot.net

Column	Issue Number	Description	Status
Backlog	97	Präsentation für Verteidigung	Präsentationen
Backlog	96	Final Tag - Demo App Test	Präsentationen
Backlog	94	Demo für Final Tag	Präsentationen
Backlog	93	Erstellen der Präsentation für Final Tag	Präsentationen
Backlog	92	Tech. Dok. - Dokument Review mit Dr. A. Danuser - Nacharbeiten	Tech. Dokumentation
Backlog	91	Tech. Dok. - Dokument Review mit Dr. A. Danuser - Review	Tech. Dokumentation
Backlog	90	Tech. Dok. - Dokument Review mit Dr. A. Danuser	Tech. Dokumentation
Backlog	89	Entwicklerhandbuch	Tech. Dokumentation
Backlog	87	Testen der Software - Iteration III	Implementation
Backlog	83	Integration der Software - Iteration III	Implementation
Ready	23	Analyse - Dokument Review mit Dr. A. Danuser - Nacharbeiten	Markt und Bedürfnisanalyse
Ready	40	System Architektur - Dokument Review mit Dr. A. Danuser	System-Architektur
Ready	41	System Architektur - Dokument Review mit Dr. A. Danuser - Review	System-Architektur
Ready	42	System Architektur - Dokument Review mit Dr. A. Danuser - Nacharbeiten	System-Architektur
Ready	70	Software Design Anwendungsfall III - Use-Case	Design
Ready	71	Software Design Anwendungsfall III - Klassendiagramm	Design
Ready	72	Software Design Anwendungsfall III - Zusammenfassung	Design
Ready	77	Design - Dokument Review mit Dr. A. Danuser	Design
Ready	78	Design - Dokument Review mit Dr. A. Danuser - Review	Design
Ready	79	Design - Dokument Review mit Dr. A. Danuser - Nacharbeiten	Design
In Progress	36	Integration von Softwarestack für siot.net	System-Architektur
In Progress	38	Integration von Softwarestack für siot.net - Fachliche Dokumentation führen	System-Architektur
In Progress	39	Integration von Softwarestack für siot.net - Technische Dokumentation führen	System-Architektur
In Progress	73	Auswahl mind. 1 Anwendungsfall für Implementation	Design
In Progress	76	Auswahl mind. 1 Anwendungsfall für Implementation - Anwendungsfall III auswählen (optional)	Design
In Progress	80	Integration der Software	Implementation
In Progress	84	Testen der Software	Implementation
In Progress	86	Testen der Software - Iteration II	Implementation
In Progress	88	Code Dokumentation	Tech. Dokumentation
In Progress	95	Final Tag - Plakat für Ausstellung	Präsentationen
Done	60	Anforderungen - Dokument Review mit Dr. A. Danuser	Anforderungen
Done	61	Anforderungen - Dokument Review mit Dr. A. Danuser - Review	Anforderungen
Done	62	Anforderungen - Dokument Review mit Dr. A. Danuser - Nacharbeiten	Anforderungen
Done	63	Software Design Anwendungsfall	Design
Done	66	Software Design Anwendungsfall I - Zusammenfassung	Design
Done	69	Software Design Anwendungsfall II - Zusammenfassung	Design
Done	81	Integration der Software - Iteration I	Implementation
Done	85	Testen der Software - Iteration I	Implementation
Done	82	Integration der Software - Iteration II	Implementation

Abbildung B.10.: Kanban-Board Fortschritt am 23.12.2015

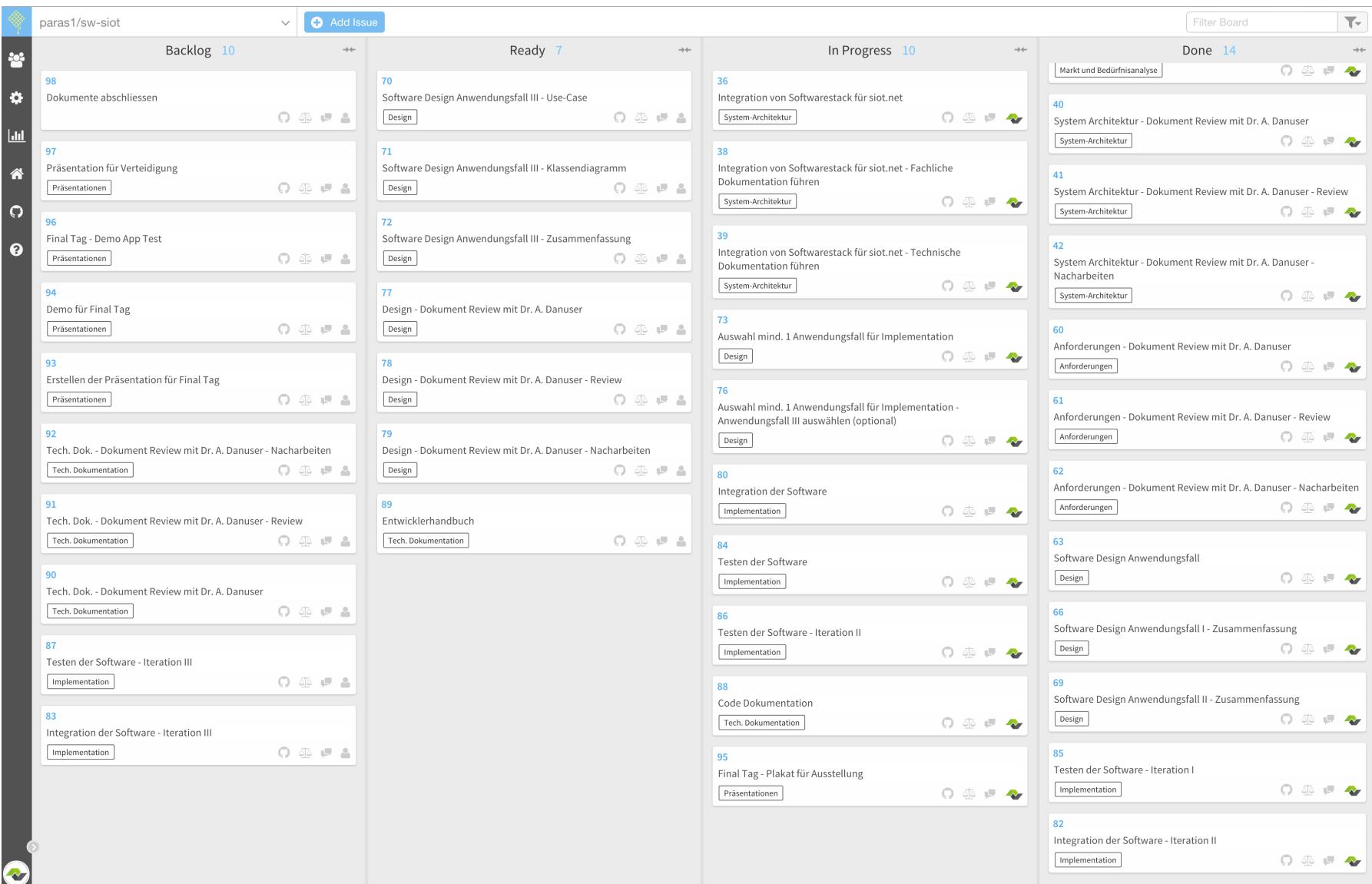


Abbildung B.11.: Kanban-Board Fortschritt am 02.01.2016

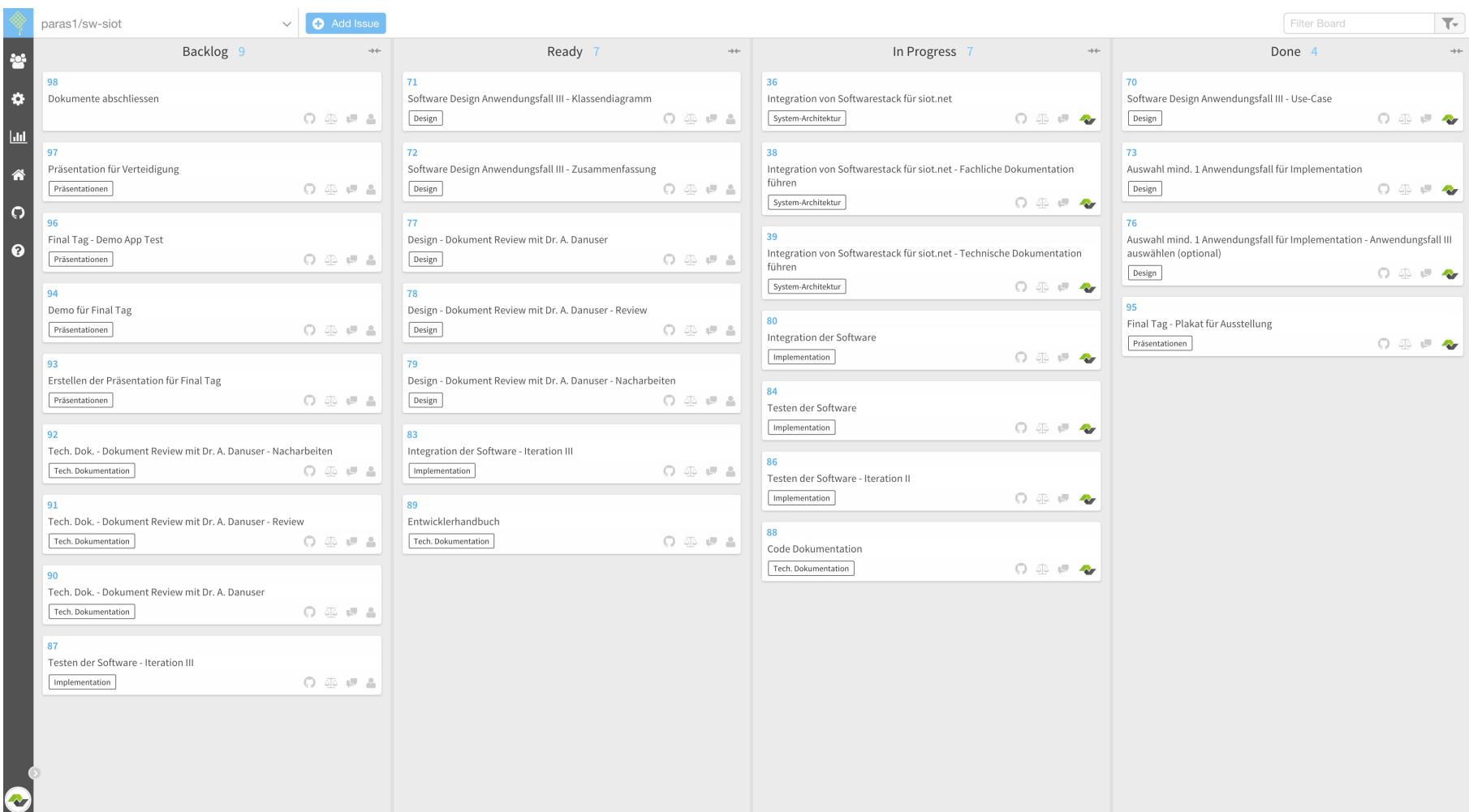


Abbildung B.12.: Kanban-Board Fortschritt am 05.01.2016

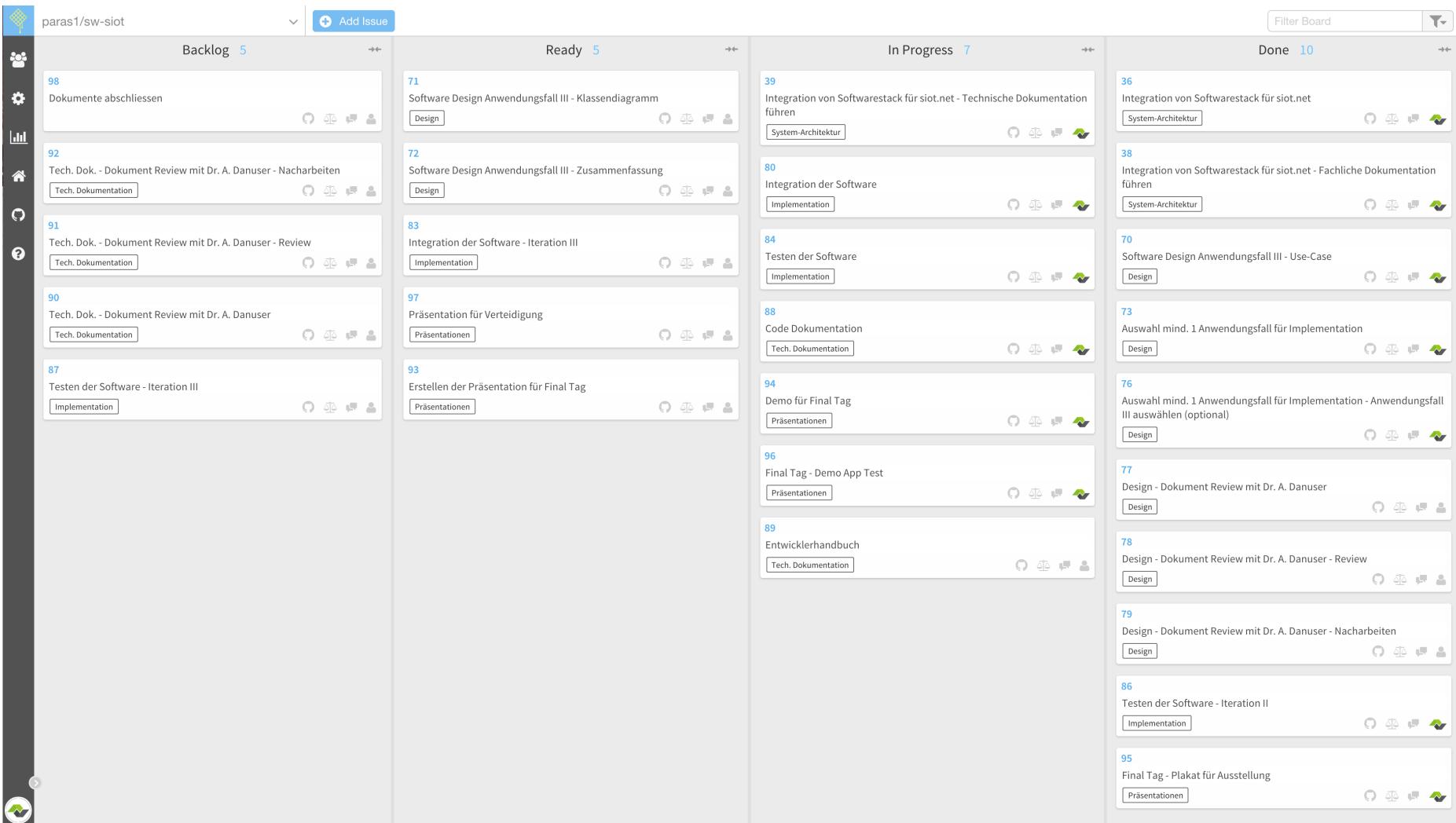


Abbildung B.13.: Kanban-Board Fortschritt am 10.01.2016

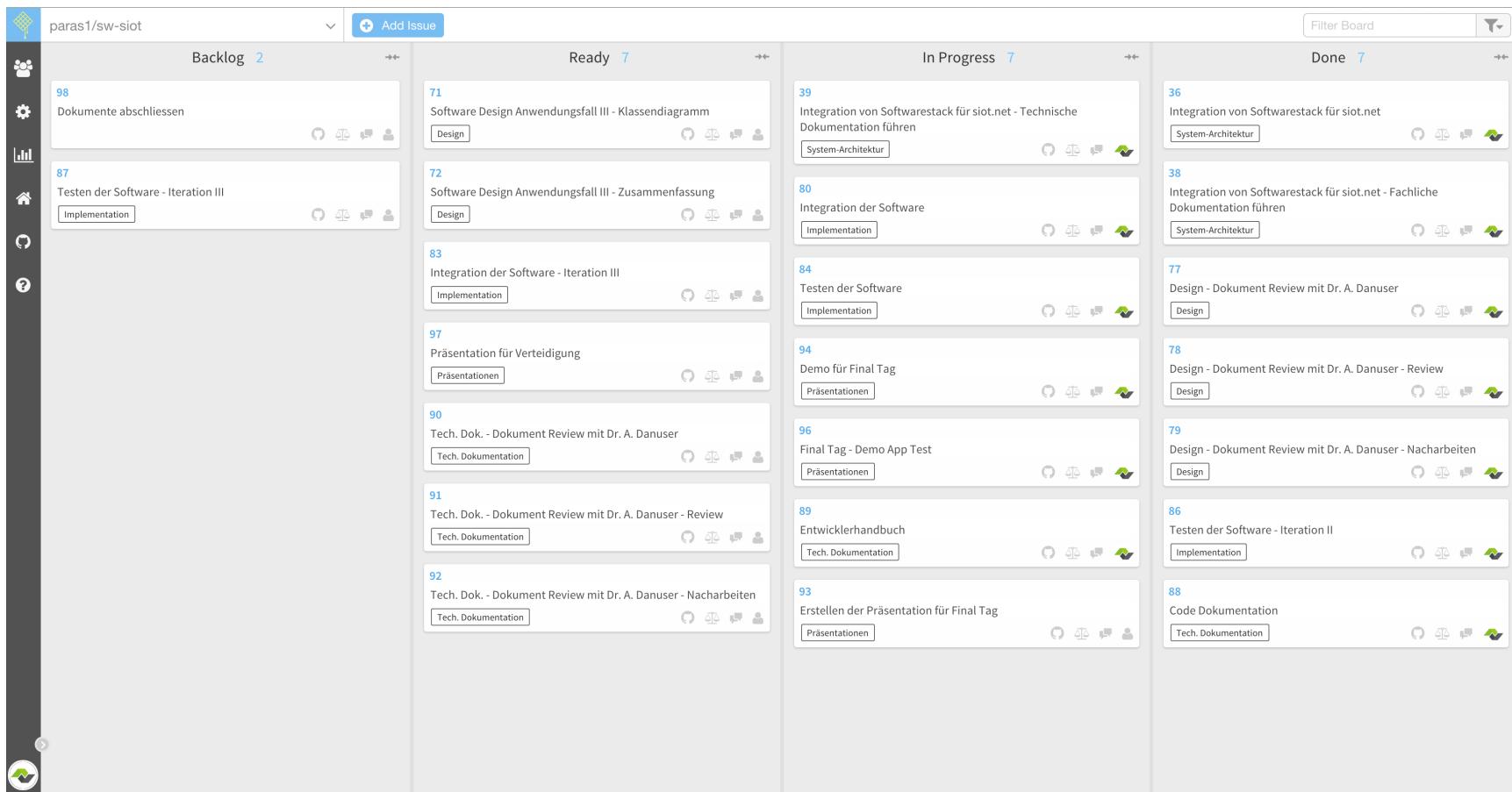


Abbildung B.14.: Kanban-Board Fortschritt am 14.01.2016

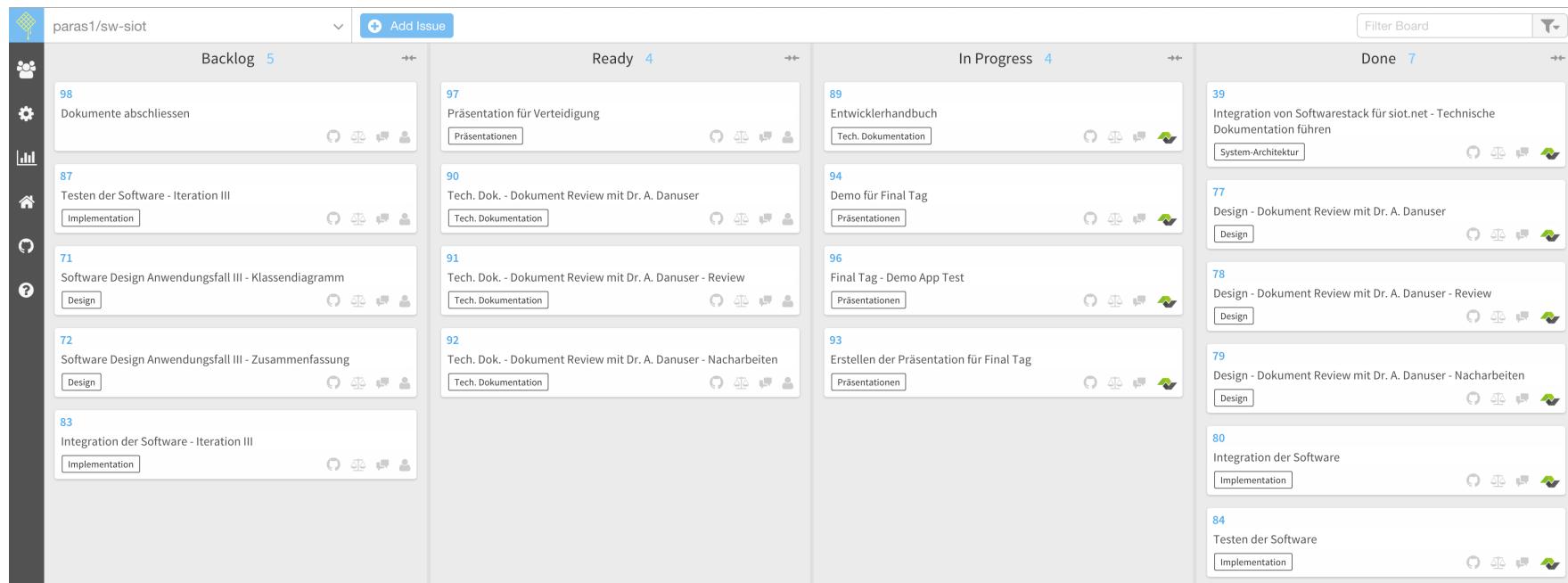


Abbildung B.15.: Kanban-Board Fortschritt am 17.01.2016

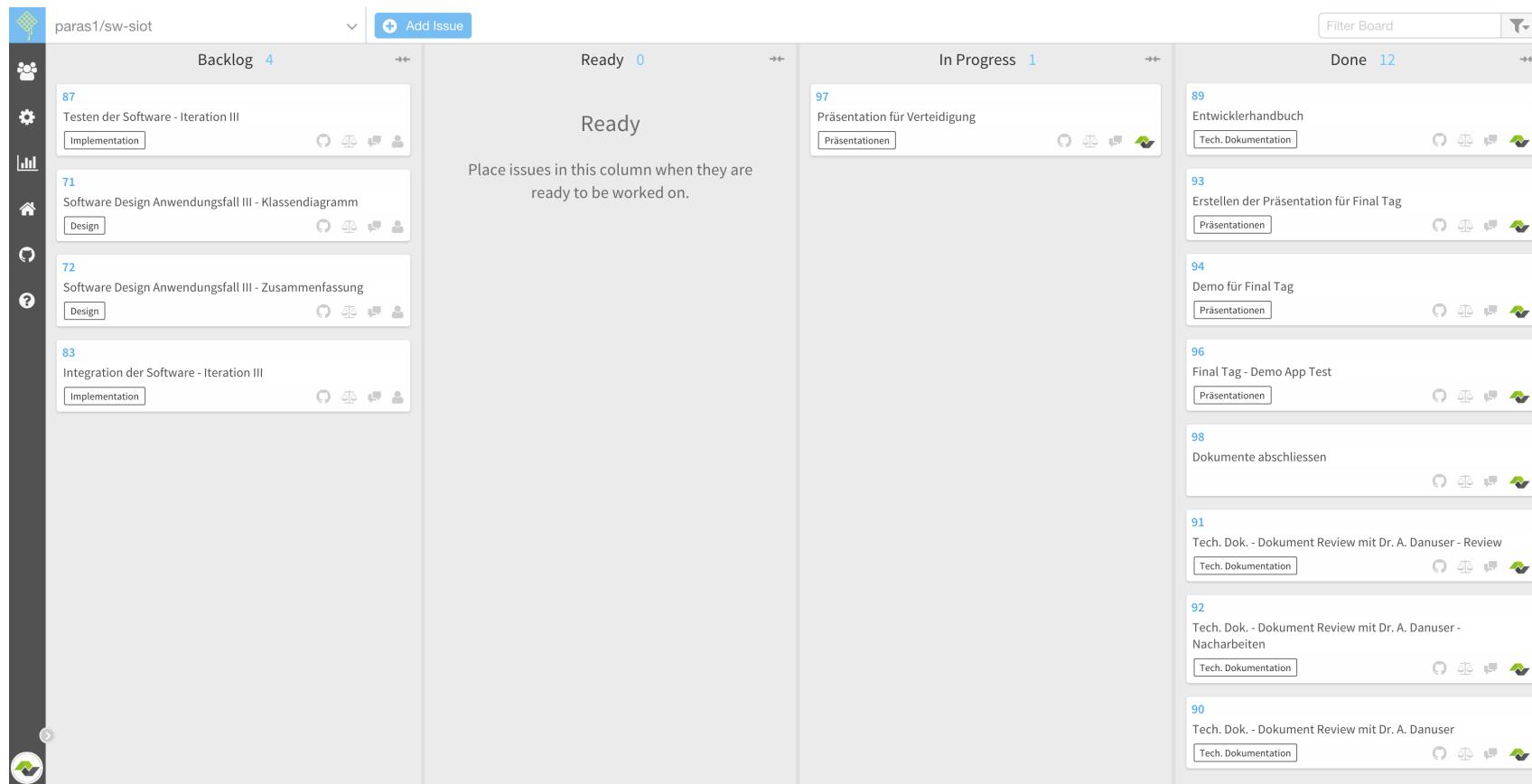


Abbildung B.16.: Kanban-Board Fortschritt am 21.01.2016