



Hochschule für Angewandte Wissenschaften Hamburg  
*Hamburg University of Applied Sciences*

# **Bachelorarbeit**

Sefai Sari

Eine adaptierbare Fernbedienung für smarte  
Umgebungen

# **Sefai Sari**

## **Eine adaptierbare Fernbedienung für smarte Umgebungen**

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik  
am Department Informatik  
der Fakultät Technik und Informatik  
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Kai von Luck  
Zweitgutachter : Sascha Kluth

Abgegeben am 22.7.2016

**Sefai Sari**

**Thema der Bachelorarbeit**

Eine adaptierbare Fernbedienung für smarte Umgebungen

**Stichworte**

Android™, Adaptierbare Fernsteuerung, Fernbedienung für agentenbasierte Systeme, Smarte Umgebung

**Kurzzusammenfassung**

Im 21. Jahrhundert sind Mobilgeräte wie Mobiltelefone und Tablets nicht mehr aus unserem Alltag wegzudenken. Jeder Mensch führt eins mit sich, entweder um erreichbar zu sein oder um seine Zeit zu vertreiben. Da auch immer mehr intelligente Gegenstände ihren Weg in die heimische Wohnung finden und für ihre Bedienung eine eigene App mitbringen an die der Benutzer sich einarbeiten muss, befasst sich diese Bachelorarbeit mit der Entwicklung einer Software, die das Erstellen von Fernsteuerungen, die auf einem Mobilgerät laufen, um Smart Environments zu steuern, erleichtern soll.

**Sefai Sari**

**Title of the paper**

An adaptable remote control for smart environments

**Keywords**

Android™, Adaptable remote control, Remote control for agent-based systems, Smart Environment

**Abstract**

In the 21 century mobile devices like mobile phones and tablets are an essential part of our everyday live. Every person has one at hand either to be contactable or to pass time. Since more and more intelligent items find their way into the home apartment which have to be operated with their own App, this bachelor thesis will consider the development of software that simplifies creating a remote control to navigate Smart Environments which will work on a mobile device.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>9</b>
1.1	Zielsetzung	10
1.2	Gliederung der Arbeit	11
<b>2</b>	<b>Analyse</b>	<b>12</b>
2.1	Vergleichbare Projekte	12
2.2	Zielsetzung	16
2.3	Szenarien	17
2.4	Anforderungsanalyse	21
2.4.1	Funktionale Anforderungen	21
2.4.2	Nicht-Funktionale Anforderungen	24
2.5	Fazit	25
<b>3</b>	<b>Benutzeroberflächen Gestaltung</b>	<b>26</b>
3.1	Desktop Anwendung	30
3.1.1	Server-Benutzeroberfläche	31
3.1.2	Admin-Benutzeroberfläche	36
3.1.3	Create Remote Controller-Benutzeroberfläche	39
3.2	Mobile Anwendung	56
3.2.1	Login Activity	57
3.2.2	New Server Activity	64
3.2.3	Controllers Activity	65
3.2.4	Account Activity	70

3.2.5	Use Controller Activity .....	71
3.3	Fazit .....	72
<b>4</b>	<b>Design .....</b>	<b>74</b>
4.1	Architektur .....	74
4.1.1	Komponenten.....	78
4.1.2	Aufbau einer Fernsteuerung .....	86
4.1.3	Umrechnung einer Fernsteuerung.....	89
4.1.4	Fazit .....	93
4.2	Kommunikation .....	94
4.2.1	Nachrichten.....	95
4.2.2	Fehlerfall.....	98
4.2.3	Fazit .....	98
4.3	Sequenzdiagramme.....	99
4.3.1	Szenarien .....	99
4.3.2	Fazit .....	106
4.4	Fazit .....	107
<b>5</b>	<b>Evaluation .....</b>	<b>108</b>
5.1	Multi-Function-Controller .....	108
5.1.1	Erfüllung der Anforderungen .....	109
5.1.2	Funktionsweise des Prototyps .....	111
5.1.3	Fazit .....	117
5.2	Fazit .....	118
5.3	Erweiterung des MFC-Systems.....	119
<b>6</b>	<b>Schluss .....</b>	<b>121</b>
6.1	Zusammenfassung .....	121
6.2	Ausblick .....	122

# Abbildungsverzeichnis

Abbildung 3.1: iOS und Android UI-Unterschiede im Überblick (Rixecker (2014)).....	27
Abbildung 3.2: Marktanteile der führenden Betriebssystemversionen weltweit von Januar 2009 bis Januar 2016 (Statista: Betriebssysteme) .....	27
Abbildung 3.3: Prognose zu den Marktanteilen der Betriebssysteme am Absatz vom Smartphones weltweit in den Jahren 2016 und 2020 (Statista: Smartphone Betriebssysteme).....	28
Abbildung 3.4: Server-Benutzeroberfläche nach dem Start .....	31
Abbildung 3.5: Server-Benutzeroberfläche während des Betriebs .....	31
Abbildung 3.6: Log in Fenster der Server-Benutzeroberfläche.....	32
Abbildung 3.7: Menüleiste der Server-Benutzeroberfläche .....	33
Abbildung 3.8: Server Bereich der Server-Benutzeroberfläche.....	34
Abbildung 3.9: Agent-APIs Bereich der Server-Benutzeroberfläche .....	35
Abbildung 3.10: Admin-Benutzeroberfläche .....	36
Abbildung 3.11: Menüleiste der Admin-Benutzeroberfläche .....	38
Abbildung 3.12: New User Fenster der Admin-Benutzeroberfläche .....	38
Abbildung 3.13: Delete User Fenster der Admin-Benutzeroberfläche .....	39
Abbildung 3.14: Create Remote Controller-Benutzeroberfläche nach dem öffnen .....	40
Abbildung 3.15: Create Remote Controller-Benutzeroberfläche während des Gebrauchs ..	41
Abbildung 3.16: New Controller Fenster der Create Remote Controller-Benutzeroberfläche .....	42
Abbildung 3.17: Unsaved Changes Fenster der Create Remote Controller-Benutzeroberfläche.....	43
Abbildung 3.18: Menüleiste der Create Remote Controller-Benutzeroberfläche.....	44
Abbildung 3.19: Funktionsleiste der Create Remote Controller-Benutzeroberfläche.....	45
Abbildung 3.20: Fernsteuerungsstrukturbereich der Create Remote Controller-Benutzeroberfläche.....	46
Abbildung 3.21: Bibliotheksstrukturbereich der Create Remote Controller-Benutzeroberfläche.....	47
Abbildung 3.22: Gerätebereich der Create Remote Controller-Benutzeroberfläche .....	49
Abbildung 3.23: Eigenschaftenbereich der Create Remote Controller-Benutzeroberfläche.	50

Abbildung 3.24: Eigenschaften einer Fernsteuerung und eines Profils im Eigenschaftenbereich.....	51
Abbildung 3.25: Eigenschaften einer Activity im Eigenschaftenbereich. ....	52
Abbildung 3.26: Aufbau der Eigenschaften eines Elementes anhand des Beispiel Elementes-„Button“.....	53
Abbildung 3.27: Eigenschaften der Elemente „Slider“ und „Toggle-Button“ sowie das Wechseln zwischen Activitys.....	55
Abbildung 3.28: Informationsbereich der Create Remote Controller-Benutzeroberfläche ..	56
Abbildung 3.29: LoginActivity .....	57
Abbildung 3.30: Navigationsleiste der Login Activity.....	58
Abbildung 3.31: Filter Funktion der Login Activity.....	59
Abbildung 3.32: Such Funktion der Login Activity .....	59
Abbildung 3.33: Menü Funktion der Login Activity .....	60
Abbildung 3.34: Anmelde Dialogfenster der Login Activity .....	61
Abbildung 3.35 Server Key Dialogfenster der Login Activity.....	62
Abbildung 3.36: Registrations Dialogfenster der Login Activity.....	63
Abbildung 3.37: New Server Activity .....	64
Abbildung 3.38: Controllers Activity .....	65
Abbildung 3.39: Navigationsleiste der Controllers Activity .....	67
Abbildung 3.40: Filter Funktion der Controllers Activity .....	67
Abbildung 3.41: Menü Funktion der Controllers Activity .....	68
Abbildung 3.42: Profiles Dialogfenster der Controllers Activity .....	69
Abbildung 3.43: Account Activity .....	70
Abbildung 3.44: Use Controller Activity.....	71
Abbildung 4.1: Systemüberblick .....	74
Abbildung 4.2: Einordnung der Trennung des verteilten Systems innerhalb der Architekturschichten. (Tanenbaum und Steen (2006), S.41).....	77
Abbildung 4.3: Komponentendiagramm des Gesamtsystems.....	78
Abbildung 4.4: Komponente „GUI-Server“ .....	79
Abbildung 4.5: Komponente „GUI-Admin“ .....	80
Abbildung 4.6: Komponente „GUI-Builder“ .....	80
Abbildung 4.7: Komponente „GUI-Authentication“ .....	80
Abbildung 4.8: Komponente „GUI-Controller“ .....	81
Abbildung 4.9: Komponente „GUI-Account“ .....	81
Abbildung 4.10: Komponente „Helper“ .....	82
Abbildung 4.11: Komponente „Init-Agent“.....	83
Abbildung 4.12: Komponente „Authentication-Agent“ .....	83
Abbildung 4.13: Komponente „Controller-Agent“ .....	84
Abbildung 4.14: Komponente „Device-Agent“ .....	84
Abbildung 4.15: Komponente „Computer-Agent“ .....	85
Abbildung 4.16: Komponente „Control-Agent-Ask“ .....	85
Abbildung 4.17: Erstellte Fernsteuerung über den GUI-Builder (Sicht auf den Gerätebereich).....	90
Abbildung 4.18: Display Maße des Mobilgeräts für die Umrechnung.....	90
Abbildung 4.19: Umgewandelte Fernsteuerung auf dem Mobilgerät.....	91

Abbildung 4.20: Darstellung einer Fernsteuerung in Portrait- und Landscape-Modus auf dem Mobilgerät.....	93
Abbildung 4.21: Blackbox-Ansicht der Middleware (Eichler (2014), S.40) .....	95
Abbildung 4.22: Szenario „Registrieren“ .....	100
Abbildung 4.23: Szenario „Anmelden“ .....	101
Abbildung 4.24: Szenario „Benutzerrechte Ändern“ .....	102
Abbildung 4.25: Szenario „Agent Einbinden“ .....	102
Abbildung 4.26: Szenario „Fernsteuerung Erstellen“ .....	103
Abbildung 4.27: Szenario „Fernsteuerung Verwenden“ .....	105
Abbildung 5.1: „Server“-Oberfläche des MFC-Systems auf dem der Button „Start Authentication UI“ angezeigt wird.....	111
Abbildung 5.2: „Login“-Oberfläche des MFC-Systems auf dem Daten eingetragen sind sowie ein Server gespeichert wurde. ....	112
Abbildung 5.3: „Authentication UI“-Oberfläche des MFC-Systems auf dem ein geblockter Benutzer angezeigt wird. ....	112
Abbildung 5.4: „Controllers“-Oberfläche des MFC-Systems ohne Fernsteuerungen.....	113
Abbildung 5.5: „Server“-Oberfläche des MFC-Systems auf dem der Button „Start Profile UI“ angezeigt wird.....	114
Abbildung 5.6: „Create Remote Controller“-Oberfläche des MFC-Systems auf dem die erstellte Fernsteuerung „Demo“ angezeigt wird. ....	115
Abbildung 5.7: „Mapping UI“-Oberfläche des MFC-Systems auf der eine Aktion zugewiesen wird. ....	116
Abbildung 5.8: „Controllers“-Oberfläche des MFC-Systems mit Fernsteuerungen.....	116
Abbildung 5.9: „Controller“-Oberfläche des MFC-Systems auf der die Fernsteuerung angezeigt wird.....	117
Abbildung 5.10: NES-Controller (Ist das Gamepad einer „Nintendo Entertainment System“ Konsole.).....	118
Abbildung 5.11: Erstellung eines NES-Controllers über den GUI-Builder.....	119
Abbildung 5.12: Anzeigen des erstellten NES-Controllers über das Mobilgerät. ....	119



# 1 Einleitung

Die Präsenz von Computern ist in der heutigen Zeit allgegenwärtig. Sie befinden sich in Chipkarten, Fahrzeugen, Mobilgeräten, Haushaltsgeräten und noch in vielen weiteren Gegenständen. Mit der fortschreitenden technologischen Entwicklung können immer kleinere und leistungsfähigere Computer gebaut werden. Die wiederum als „Smart-Objects“ ihren Weg in die heimischen Wohnungen finden. „Smart-Objects“ sind Gegenstände, die um ein „Embedded System“ erweitert werden, und somit die Fähigkeiten erhalten, Informationen zu ermitteln, diese zu kommunizieren und Entscheidungen zu treffen (vgl. [López u. a. \(2011\)](#)).

„Smart Environment“ beschäftigt sich mit diesem Aspekt. Hierbei werden zwei Hauptziele verfolgt. Als erstes die Verdrängung der Geräte in den Hintergrund, sodass der Benutzer diese gar nicht mehr als solche wahrnimmt und als zweites die Unterstützung des Menschen im alltäglichen Leben. Hierdurch soll Sicherheit sowie der Komfort erhöht werden.

Die Ausprägungen von „Smart Environment“ sind nicht neu, so wurde bereits in den neunziger Jahren unter dem Begriff „Ubiquitous Computing“, der von [Weiser \(1991\)](#) geprägt wurde, dies beschrieben. Im europäischen Raum wird dieses Forschungsgebiet auch unter dem Namen „Ambient Intelligence“ zusammengefasst. Dabei sollen die Systeme aktiv auf die Bedürfnisse des Benutzers eingehen und ihm Aufgaben abnehmen. Hierzu werden alltägliche Gegenstände in sogenannte „Smart-Objects“ verwandelt.

Für die Vernetzung von „Smart-Objects“ zu einem komplexen verteilten System können Multiagentensysteme genutzt werden. Bei diesen wird ein „Smart-Object“ als ein Agent angesehen, der mit anderen Agenten durch den Austausch von Nachrichten kommuniziert und so zusammenarbeitet.

Durch die Miniaturisierung von Computern sind wir im Alltag schon jetzt von unzähligen „Smart-Objects“ umgeben. Von denen wir manche als solche noch wahrnehmen und ande-

re wiederum nicht. So sind nicht nur die Eigenschaften der Geräte entscheidend, sondern auch wie in Zukunft mit diesen interagiert wird. Damit diese in den Hintergrund treten können, müssen neue intuitive Bedienkonzepte und Interaktionsmöglichkeiten entstehen.

## 1.1 Zielsetzung

Diese Arbeit siedelt sich im Bereich „Human–Computer Interaction“ ein und beschäftigt sich somit mit der Schnittstelle zwischen Mensch und Computer. Dabei werden neben Erkenntnissen der Informatik auch solche aus dem Design herangezogen.

Es geht darum eine grafische Oberfläche zu entwerfen, über die der Benutzer Fernsteuerungen für „Smart-Objects“ Designen und Erstellen kann, ohne Programmierkenntnisse vorauszusetzen. Hierbei wird die Perspektive der adaptierenden Fernsteuerung außen vor gelassen um sich voll und ganz dem Aspekt der adaptierbaren Fernsteuerungen zu widmen. Die Bedienung der „Smart-Objects“ soll über ein Android™ Mobilgerät erfolgen. Der Benutzer soll die Möglichkeit haben, bei der Erstellung der Fernsteuerung, diese visuell sowie funktionell frei zu Gestalten. Die funktionelle Gestaltung beschränkt sich hierbei auf die Funktionalitäten die das „Smart-Object“ anbietet, so bietet beispielsweise ein Beleuchtungs-Agent die Funktionen Licht an und Licht aus an, sowie die Intensität zu regulieren. Der Benutzer könnte nun eine Fernsteuerung zusammenstellen der die Funktionalitäten des Beleuchtungs-Agenten nutzt, um so die Lichtquellen zu bedienen.

Um die Umsetzbarkeit zu demonstrieren, soll eine prototypische Anwendung entwickelt werden, über die der Benutzer eine Fernsteuerung erstellen und diese auf einem Android™ Mobilgerät nutzen kann. Zu Testzwecken soll eine Fernsteuerung erstellt werden, mit dessen Hilfe ein Computer gesteuert werden soll. Der Computer ist hierbei stellvertretend für ein „Smart-Object“.

Die Idee Elektronische Geräte in der Wohnung über das Mobilgerät, auf dem eine selbst erstellte Fernsteuerung läuft, zu steuern, entstand Im Rahmen einer Pflichtveranstaltung zusammen mit Holland. Das daraus resultierende Projekt wurde im Rahmen einer Wahlpflichtveranstaltung weitergeführt. Hierbei schloss sich Oster uns an. [Holland \(2015\)](#) war für die Authentifikation auf dem Server zuständig, [Oster \(2015\)](#) hingegen für die Zuweisung und Ausführung der Aktionen. Ich ([Sari \(2015\)](#)) war wiederum für die Erstellung und Nutzung der Fernsteuerungen zuständig. Die daraus entstandenen Erkenntnisse werden in dieser Arbeit fortgesetzt.<sup>1</sup>

---

<sup>1</sup> Ich möchte mich hiermit bei [Holland \(2015\)](#) und [Oster \(2015\)](#) bedanken, die eine großartige Arbeit geleistet haben.

## 1.2 Gliederung der Arbeit

Die Arbeit gliedert sich in sechs Bestandteile, wobei die Einleitung den ersten Teil darstellt. In diesem Kapitel geht es darum sich einen kurzen Überblick über das Themengebiet zu verschaffen, sowie die Zielsetzung der Arbeit darzulegen.

Das Kapitel **2** befasst sich mit der Zielsetzung und bildet somit die Analyse. Dazu werden vergleichbare Projekte vorgestellt und in die Zielsetzung mit eingebunden. Danach werden Szenarien erläutert, die verschiedenste Interaktionsmöglichkeiten mit dem System beinhalten. Daraus werden funktionale als auch nicht-funktionale Anforderungen identifiziert. Das Kapitel wird mit einem Fazit abgeschlossen, welches eine Zusammenfassung des Kapitels darstellt.

Kapitel **3** befasst sich mit der Gestaltung der Benutzeroberflächen des Systems, ausgehend von den Ergebnissen aus Kapitel zwei. So werden hier die Oberflächen der Desktop- und Mobile-Anwendung entwickelt und erläutert.

In Kapitel **4** wird das Design des Systems beschrieben, aufgrund der Ergebnisse aus Kapitel zwei. Hierbei geht man auf die Architektur und die Kommunikation des Systems ein. Des Weiteren erfolgen hier Fazits über alle relevanten getroffenen Entscheidungen.

Die Umsetzung und Evaluation des Designs erfolgt in Kapitel **5**. Hierfür werden die grundlegenden Funktionalitäten aus der Designentscheidung, die relevant für die Realisierung des Systems sind, durchgeführt. Darüber hinaus wird die Funktionsweise des entwickelten Prototyps vorgestellt.

In Kapitel **6** das auch zugleich das letzte Kapitel der Arbeit darstellt, werden alle Kapitel, wie die Ergebnisse der Arbeit, zusammengefasst. Des Weiteren erfolgt ein Ausblick, indem weitere interessante Fragestellungen für mögliche weiterführende Arbeiten dargestellt werden.

## 2 Analyse

Ziel der Analyse ist es, die Zielsetzung dieser Arbeit genauer zu erleuchten und ähnliche Projekte aufzuzeigen. Zudem werden in diesem Kapitel die Anforderungen an die zu erstellende Software formuliert und mögliche Szenarien, die relevant sind, durchgespielt. Zum Ende erfolgt ein Fazit, welches das Kapitel zusammenfasst.

### 2.1 Vergleichbare Projekte

Das Problem gliedert sich in zwei Bereiche auf, einmal die Individualisierung und zum anderen die Kontextualisierung. In der Individualisierung geht es darum, dass eine Anwendung auf die Anforderungen eines Benutzers maßgeschneidert ist. Hierbei gibt es zwei Variationen. Die lernende Variante beschäftigt sich mit der automatischen Anpassung auf die Bedürfnisse eines Benutzers, zum Beispiel ist der Benutzer Linkshänder und hat sein Speichern- sowie Abbrechen-Button gerne links statt rechts stehen. So merkt sich das System dies für diesen Benutzer und übernimmt dies auch für weitere Anwendungen. Die nicht lernende Variante beschäftigt sich mit der manuellen Anpassung der Anwendung. Der Benutzer kann seine Anwendungen frei gestalten, diese Einstellungen werden jedoch für alle Benutzer übernommen. Somit hat jeder Benutzer die gleiche Sicht auf die Anwendung. Kontextualisierung hingegen beschreibt das Verhalten von Anwendungsprogrammen, die Informationen über ihre Umgebung benutzen, um ihr Verhalten darauf abzustimmen.

Das „Distributed Artificial Intelligence Laboratory“<sup>2</sup> kurz DAI-Labor in der TU-Berlin beschäftigt sich neben zahlreichen Themengebieten auch mit dem Thema „Smart Environment“. In den folgenden drei Artikeln aus dem DAI-Labor wird die Erstellung, der Aufbau und die Funktionsweise, der zu entwickelnden Anwendungen, erläutert und gehören in die Kategorie Individualisierung sowie Kontextualisierung.

---

<sup>2</sup> Distributed Artificial Intelligence Laboratory – (DAI-Labor, vgl.)

Im Artikel „A Meta User Interface to Control Multimodal Interaction in Smart Environments“ von [Roscher u. a. \(2009\)](#) geht es um die Entwicklung einer grafischen Oberfläche, die dem Benutzer eine breitere Palette an Konfigurationsmöglichkeiten als herkömmliche Oberflächen bieten soll. Diese besteht aus einer Main-Oberfläche, die die Konfigurationsmöglichkeiten anbietet. Innerhalb dieser Main-Oberfläche wird die Benutzeroberfläche für die „Smart-Objects“ angezeigt. Für die Entwicklung wurde die von ihnen entwickelte Laufzeitumgebung Multi-Access Service Platform kurz MASP verwendet. Diese soll eventuell auftretende Laufzeitprobleme in einer intelligenten Umgebung beheben. Zudem kann die Laufzeitumgebung mehrere Benutzeroberflächen von verschiedenen „Smart-Objects“ verwalten. Diese Benutzeroberflächen können wiederum vom Benutzer angepasst werden oder auf verschiedene Interaktionsressourcen, wie einem Smartphone oder einem Fernseher, verschoben werden. Des Weiteren wurde eine „FollowMe“ Funktion implementiert, die die Location des Benutzers in der Wohnung ermitteln kann und somit dem Benutzer nur die Benutzeroberflächen der „Smart-Objects“ anzeigt, die sich in der Location befinden. Die Laufzeitumgebung unterstützt Multimodale Interaktionsmöglichkeiten, die vom Benutzer gewählt werden können. Des Weiteren wird eine Statusanzeige eingeblendet, auf der die vom „Smart-Object“ unterstützten Multimodalen Bedienungsarten angezeigt werden. Weitere Konfigurationsmöglichkeiten der Benutzeroberfläche werden im Artikel genauer erläutert.

Im Artikel „Adjustable Context Adaptations for User Interfaces at Runtime“ von [Schwartz u. a. \(2010\)](#) beschreibt ein dynamisches Layout Modell, das sich zur Laufzeit durch äußere Einflüsse automatisch anpassen sowie vom Benutzer nach persönlichen Vorlieben angepasst werden kann. Das Layout-Modell besteht aus Containern und Elementen, wobei die Elemente den sichtbaren Teil darstellen. Container hingegen können weitere Container beherbergen und so eine Baumartige Hierarchie bilden, mit Elementen als Blättern. Die automatische Anpassung der Oberfläche erfolgt durch den Constraint-Solver „Cassowary“, der Informationen wie Displaygröße, aktiver Teilbaum und Anweisungen beachtet und aus diesen Informationen die Position und die Größe für jedes Element auf der Benutzeroberfläche setzt. Eine Anweisung beschreibt eine Kontextvariable, um an eine bestimmte Situation eine Aktion zu binden, dies wird im Artikel genauer erläutert.

Im Artikel „Towards Multimodal Interaction in Smart Home Environments: The Home Operating System“ von [Weingarten u. a. \(2010\)](#) geht es um die Entwicklung eines Hausbetriebssystems, dessen grafische Oberfläche so simpel wie möglich gehalten werden soll, durch eine einfache Form sowie einer flachen Informationsstruktur. Die Anwendung teilt sich in vier grundlegende Bereiche ein. Der erste Bereich ist die Lobby, diese zeigt einen Überblick der Wohnung sowie die letzte Systemmitteilung und die zuletzt verwendeten Assistenzsysteme an. Des Weiteren werden hier Temperatur und Energieverbrauch der Wohnung angezeigt. Der zweite Bereich ist das Home Control Center (HCC), dieses zeigt einen Überblick der intelligenten Geräte, die sich im Raum befinden, den man zuvor über die Lobby gewählt hat und deren aktuellen Energieverbrauch. Über den HCC können intelligente Geräte in der Wohnung bedient werden. Der dritte Bereich ist der Assistent-Lauscher, dieser zeigt einen

Überblick der Assistenz-Systeme. Hier können Assistenz-Systeme gestartet und beendet werden. Der vierte Bereich ist das Message Center, hier werden Informationen vom System zu den jeweiligen Assistenz-Systemen angezeigt. Da in dieser Anwendung das MASP dahinter steht, können auch hier multimodale Eingaben getätigt werden.

In den folgenden zwei Artikeln geht es um die Interaktion mit den „Smart-Objects“ und gehört in die Kategorie Kontextualisierung.

Der Artikel „Enhancing Interaction with Smart Objects through Mobile Devices“ von **Bernardos u. a. (2011)** beschreibt ein Interaktionsmodell zwischen Mobilgeräten und „Smart-Objects“ sowie den Prototyp, basierend auf dem „event condition action“ (ECA) Model. Das ECA Model besteht aus mehreren Designprinzipien, diese werden im Artikel genauer erläutert. Um mit einem „Smart-Object“ über das Mobilgerät interagieren zu können, muss einmalig dessen Controller von einem Server heruntergeladen werden. Hierfür wird das „Smart-Object“ mit Bluetooth oder einem NFC-Tag ausgestattet. Nun muss sich der Benutzer in der Nähe des „Smart-Objects“ befinden und kann eine Identifizierungs-URL abrufen, diese liefert den Controller für das „Smart-Objects“. Der Controller wird auf dem Mobilgerät gespeichert, damit dieser nicht erneut heruntergeladen werden muss und jeder Zeit abgerufen werden kann. Auf dem Mobilgerät können weitere Controller von verschiedenen „Smart-Objects“ gespeichert werden, somit ist das Mobilgerät das zentrale Steuergerät, da man über dieses mit den „Smart-Objects“ interagiert. Für die Implementierung des Prototyps wurde die Android Version 2.3 verwendet und mit einem NFC fähigem Google Nexus S Smartphone getestet.

Im Artikel „Point&Control – Interaction in Smart Environments: You Only Click Twice“ von **Budde u. a. (2013)** wird für die Interaktion zwischen Mobilgeräten und „Smart-Objects“ eine Microsoft Kinect verwendet, die an den Server angeschlossen ist. Der Server besitzt Informationen über angeschlossene Geräte, ihre Schnittstelle und ihre Position. Mit Hilfe der Kinect ist der Server in der Lage den Benutzer zu lokalisieren und die Zeigerichtung zu ermitteln. Hierfür wurde ein „Point&Control“ System entwickelt. Der Benutzer zeigt mit seinem Mobilgerät auf ein „Smart-Object“ und drückt auf das Display. Die Kinect erfasst diese Information und leitet sie an den Server weiter, wo berechnet wird auf welches „Smart-Object“ gezeigt wurde. Der Server schickt nun die HTML Schnittstelle, also die Fernsteuerung um mit dem „Smart-Object“ zu interagieren, auf das Mobilgerät. Jede Aktion die über die Fernsteuerung getätigt wird, wird an den Server gesendet, dieser führt dann die Aktion auf die „Smart-Objects“ aus. Damit ein „Smart-Object“ erfasst werden kann, muss dieser nicht explizit im Sichtfeld der Kinect sein, da die Zeigerichtung berechnet wird und der Server wie oben beschrieben Informationen über die Position der „Smart-Objects“ besitzt.

In der Bachelorarbeit „Generischer Smart Home Controller auf Android™ OS“ von Kolbaja (2012) geht es um die Entwicklung einer Fernsteuerung für das Living Place<sup>3</sup>, die auf einem Android Gerät läuft und gehört in die Kategorie Individualisierung. Diese besteht aus einer Main-Oberfläche, die grundlegende Funktionalitäten anbietet. Innerhalb dieser Main-Oberfläche wird die Fernsteuerung für die „Smart-Objects“ angezeigt. Hierfür wurde eine Plugin Host Software entwickelt, die Plugins aufnehmen und die Steuerung neuer und schon vorhandener Komponenten des Living Place übernehmen soll. Die Erstellung neuer Plugins soll hierbei ohne Programmierkenntnisse erfolgen. Hierfür wurde ein System entwickelt, der als Assistent für die Erstellung neuer Plugins dient und auf einem Computer läuft. Die Erstellung eines Plugins läuft in mehreren Schritten ab, diese werden in der Arbeit ausführlich beschrieben. Das daraus resultierende Plugin wird im JSON Format gespeichert. Für die Nutzung der Plugins wurde für Android Mobilgeräte eine App namens „SmartHomeController“ entwickelt, die als Fernsteuerung fungieren soll, um mit den Komponenten im Living Place zu interagieren. Die App verwendet die JSON Dateien und erstellt aus ihnen die sichtbaren Elemente auf der Fernsteuerung, zum Beispiel Buttons. Die Kommunikation mit den Komponenten im Living Place erfolgt über das ActiveMQ. Das ActiveMQ ist ein quelloffener Message Broker. Sobald eine Aktion über die Fernsteuerung getätigt wird, wird eine JSON Nachricht an das ActiveMQ gesendet. Diese wiederum verteilt die Nachricht an alle relevanten Komponenten, die sich für diese Information am Broker registriert haben.

In den Online-Artikeln „Android: Benutzerverwaltung für mehrere User verwenden“ von Joos (2015) und „Android 5.0 Lollipop: Nutzerprofile kurz erklärt“ von Schartel (2014) wird die Erstellung, Einrichtung und die Unterschiede zwischen Profilen und Benutzern erklärt. Seit Android 4.2.2 können auf den Android-Geräten Benutzer angelegt werden. Mit dem Update 4.3 kamen die Profile dazu und mit Android 5.0 das Gast-Konto. Auf einem Android Gerät können sich mehrere Benutzer- sowie Profil-Konten befinden, zwischen denen gewechselt werden kann. Die Konten dienen dazu das Gerät mit anderen Nutzern zu teilen, ohne befürchten zu müssen, dass persönliche Daten eingesehen werden können. Das Gast-Konto speichert alle Informationen und Dateien bis zum Ende einer Session und löscht sie danach, zudem kann sich ein Gast bei Bedarf auf dem Gast-Konto mit seinem Google Account anmelden. Das Profil-Konto hingegen ist ein eingeschränktes Konto. Hier kann der Zugriff auf Apps und Dienste beschränkt werden, so dass der Zugriff auf die Inhalte nicht mehr zugänglich ist. Ein Benutzer-Konto hingegen erlaubt den vollen Zugriff auf alle Apps und Dienste des Android-Systems. Beide Artikel gehören in die Kategorie Individualisierung.

In meiner Arbeit werde ich mich mit der nicht lernenden Variante der Individualisierung beschäftigen und die Kontextualisierung sowie die lernende Variante der Individualisierung außen vor lassen und somit den Individualisierungsaspekt (nicht lernenden Variante) der in diesem Kapitel erläuterten Artikel, in meine Arbeit mit einbringen.

---

<sup>3</sup> Living Place HAW – (von Luck u. a. (2010), vgl.)

## 2.2 Zielsetzung

In den Artikeln von [Roscher u. a. \(2009\)](#) und [Weingarten u. a. \(2010\)](#) ging es um die Entwicklung einer grafischen Oberfläche, die als Fernsteuerung dienen soll und die zur Laufzeit angepasst werden kann. In Meiner Arbeit geht es darum zur Laufzeit Fernsteuerungen zu erzeugen und zu nutzen.

Hierfür sollen zwei Anwendungen implementiert werden. Die erste befindet sich auf dem Computer und stellt den Server dar, dieser dient als Assistent für die Erstellung von Fernsteuerungen. Die zweite Anwendung befindet sich auf dem Mobilgerät und stellt den Client dar. Auf diesem sollen die erstellten Fernsteuerungen genutzt werden. Dies ähnelt der Bachelorarbeit von [Kolbaja \(2012\)](#) mit dem Unterschied, das die Fernsteuerung nicht auf einer Main-Oberfläche laufen soll. Zudem sollen die Fernsteuerungen frei gestaltbar sein, wie das freie Positionieren von Elementen. Zusätzlich soll der Benutzer während der Erstellung einer Fernsteuerung ein visuelles Feedback bekommen, auf dem er die Elemente in Echtzeit positionieren kann. Außerdem soll auch hier ein Konzept erarbeitet werden, über das weitere Agenten<sup>4</sup> zur Laufzeit dem System hinzugefügt werden können, um diese bei der Erstellung von Fernsteuerungen zu verwenden.

Der Aufbau der Fernsteuerung soll genau wie im Artikel von [Schwartz u. a. \(2010\)](#) aus Containern und Elementen bestehen. Die Container stellen hierbei die Activitys dar. Das heißt, eine Fernsteuerung kann aus mehreren Activitys bestehen, zwischen denen gewechselt werden können soll und die jeweils Elemente beinhalten können. Auch hier sind Elemente der sichtbare Teil für den Benutzer. Bei Elementen kann es sich um einen Button, Slider oder einen Toggle-Button handeln.

Die Fernsteuerungen sollen zudem eine besondere Eigenschaft haben. Wie in den Online Artikeln von [Joos \(2015\)](#) und von [Schartel \(2014\)](#), die die Profile- und Benutzer-Konten vom Android Betriebssystem näherbringen, soll es auch hier möglich sein für Fernsteuerungen Profile für verschiedene Benutzer anzulegen. Diese Profile können dann nur von dem Benutzer bearbeitet und genutzt werden. Ein Benutzer soll mehrere Profile einer Fernsteuerung erstellen können, um so zum Beispiel das Profil einer Fernsteuerung zu erstellen, welches er nur an Wochenenden nutzt.

Es wird kein Bluetooth oder NFC-Tag wie im Artikel von [Bernardos u. a. \(2011\)](#) oder eine Microsoft Kinect wie im Artikel von [Budde u. a. \(2013\)](#) genutzt, um die Fernsteuerung eines „Smart-Object“ anzufragen, da eine Fernsteuerung mehrere Agenten nutzen können soll und somit das Steuern von mehreren „Smart-Objects“ ermöglicht.

---

<sup>4</sup> Ein Agent stellt die Funktionalitäten eines „Smart-Objects“ bereit.



## 2.3 Szenarien

In diesem Teil des Kapitels geht es darum, die relevantesten Szenarien für das System aufzuzeigen. In den Szenarien ist der Benutzer der Akteur, der mit dem System interagiert. Das System kann hierbei der Server oder der Client sein. Die Nummerierungen stellen Fehlerfälle sowie optionale Fälle dar und werden am Ende jedes Szenarios erläutert. Die Nummerierung bezieht sich hierbei immer auf den vorangegangenen Satz.

### Registrieren

Das Ziel ist es, dass der Benutzer sich über den Client registriert. Dies möchte er machen, um sich anmelden zu können. Am Ende des Szenarios sollte der Benutzer über das Mobilgerät registriert sein. Um sich registrieren zu können muss eine Verbindung zu einem Server aufgebaut werden.

Der Benutzer fügt einen neuen Server in die Liste der „LoginActivity“ hinzu, indem er auf den „Add Server“-Button drückt. Der Client öffnet die „NewServerActivity“. Über diese Activity trägt der Benutzer die IP-Adresse des Servers sowie eine Bezeichnung für den Server ein und bestätigt die Aktion über den „Save“-Button (a). Der Client schickt dem Server eine Nachricht um zu prüfen ob dieser erreichbar ist. Auf diese Nachricht (1) antwortet der Server. Danach fügt der Client den erreichbaren Server der Liste hinzu und öffnet wieder die „LoginActivity“. Der Benutzer möchte sich nun auf diesem Server registrieren und drückt hierfür auf den hinzugefügten Server sowie danach auf den „New register-Button und gibt einen Benutzernamen sowie ein Passwort an. Der Client verschlüsselt das Passwort und sendet es mit dem angegebenen Benutzernamen an den Server. Der Server prüft ob der angegebene Benutzername vergeben ist und sendet eine Antwort (2). Zum Schluss meldet der Client dem Benutzer, dass die Registrierung erfolgreich war.

Fehlerfall:

1. Der Server antwortet nicht: Dem Benutzer wird mitgeteilt, dass der angegebene Server nicht erreichbar ist.
2. Der Benutzername ist vergeben: Der Server sendet eine Fehlnachricht an den Client, der wiederum zeigt diese dem Benutzer an.

Optional:

- a. Der Benutzer gibt die IP-Adresse an, indem er den QR-Code auf der Serveroberfläche scannt.

## Anmelden

Das Ziel ist es, dass der Benutzer sich über den Client anmeldet. Dies möchte er machen, um Fernsteuerungen abrufen und nutzen zu können. Damit er dies machen kann, muss er den Schritt „Registrieren“ erfolgreich ausgeführt haben. Am Ende des Szenarios sollte der Benutzer über das Mobilgerät angemeldet sein.

Der Benutzer wählt aus der Liste der „LoginActivity“ einen Server aus und gibt seinen Benutzernamen und sein Passwort ein. Der Client verschlüsselt das Passwort und sendet es mit dem angegebenen Benutzernamen an den Server. Der Server prüft, ob ein Benutzer unter dem angegebenen Benutzernamen registriert und das angegebene Passwort korrekt ist und sendet dann eine Antwort (1). Der Client fordert den Benutzer dazu auf, den Server Key einzutragen oder diesen als E-Mail anzufordern. Der Benutzer trägt den Server Key ein und bestätigt, indem er auf den „OK“-Button drückt (a). Der Client sendet den vom Benutzer eingetragenen Key an den Server. Der Server ruft den Key, der sich auf der Serveroberfläche befindet ab, vergleicht die beiden Keys miteinander und sendet danach eine Antwort (2). Der Client zeigt die „ControllersActivity“ an, über die der Benutzer Fernsteuerungen abrufen und nutzen kann.

Fehlerfall:

1. Es existiert kein Benutzer mit dem Benutzernamen: Der Server sendet eine Fehlermeldung an den Client, die besagt, dass der Benutzername ungültig ist.  
Das Passwort ist nicht korrekt: Der Server sendet eine Fehlermeldung an den Client, die besagt, dass das Passwort ungültig ist.
2. Die Keys stimmen nicht überein: Der Server sendet eine Fehlermeldung an den Client, die besagt, dass der Server Key ungültig ist.

Optional:

- a. Der Benutzer fordert den Key per E-Mail an und trägt diesen ein: Der Server prüft, ob der eingetragene Key mit dem in der E-Mail versendeten Key übereinstimmt.

## Benutzerrechte Ändern

Das Ziel ist es, dass der Admin über den Server die Berechtigung eines Benutzers ändert. Dies möchte er machen, um die Berechtigungen des Benutzers zu erweitern oder zu beschränken. Damit er dies machen kann, muss er die Admin-Berechtigungen besitzen sowie über die Serveroberfläche angemeldet sein, zudem muss er den Schritt „Registrieren“ erfolgreich ausgeführt haben. Am Ende des Szenarios sollte der Admin die Berechtigung des Benutzers über die Admin-Oberfläche geändert haben.

Der Admin klickt in der Menüleiste der Serveroberfläche auf den Reiter „Admin“ und danach auf „Admin UI“ um die Admin-Oberfläche zu öffnen. Der Server listet alle Benutzer in der Admin-Oberfläche in einer Liste auf. Der Admin wählt den Benutzer aus der Liste aus, dessen Berechtigung geändert werden soll und wählt aus einer Dropdown-Liste eine neue Berechtigung für den Benutzer. Der Server übernimmt die neue Berechtigung und sendet diese an den Client, auf dem der Benutzer angemeldet ist, damit diese vom Client übernommen wird.

### **Agent Einbinden**

Das Ziel ist es, dass der Benutzer über den Server eine API eines Agenten ins System einbindet. Dies möchte er machen, um bei der Erstellung einer Fernsteuerung die Funktionen des Agenten mit ein zu integrieren. Am Ende des Szenarios sollte der Benutzer eine API eines Agenten über die Serveroberfläche ins System eingebunden haben.

Der Benutzer möchte eine neue API einbinden und öffnet hierfür das Windows Öffnen-Dialogfenster über die Serveroberfläche, indem er auf den „Add“-Button klickt, danach wählt er die hinzuzufügende API aus und drückt auf „Open“. Der Server fügt die ausgewählte API der Liste über den „Add“-Button hinzu.

### **Fernsteuerung Erstellen**

Das Ziel ist es, dass der Benutzer eine Fernsteuerung über den Server erstellt. Dies möchte er machen, um einen Agenten über das Mobilgerät zu bedienen. Damit er dies machen kann, muss er den Schritt „Agent Einbinden“ erfolgreich ausgeführt haben. Am Ende des Szenarios sollte der Benutzer über den GUI-Builder eine Fernsteuerung erstellt haben, welche einen Agenten bedienen kann.

Der Benutzer öffnet den GUI-Builder, indem er auf den „Create Remote Controller“-Button klickt, danach klickt er auf den „New Controller“-Button, der sich in der Menüleiste unter dem Reiter „File“ befindet. Der Server fordert den Benutzer auf einen Namen für die neue Fernsteuerung sowie einen für die erste Activity einzutragen, zudem wird in einer Dropdown-Liste Mobilgerätenamen aufgelistet. Der Benutzer trägt die Bezeichnungen ein und wählt ein Mobilgerät aus der Dropdown-Liste aus, danach bestätigt er indem er auf den „OK“-Button klickt (1) (a). Der Server zeigt im „Gerätebereich“ des GUI-Builders das aus der Dropdown-Liste ausgewählte Mobilgerät als Bild an, zudem wird die erstellte Fernsteuerung, mit der Activity im Fernsteuerungsstrukturbereich, in einer Baumstruktur dargestellt. Der Benutzer erstellt eine neue Aktion und zieht hierfür aus dem „Bibliotheksstrukturbereich“ ein Element auf das Bild des Mobilgerätes im Gerätebereich. Der Server zeigt im „Eigenschaftenbereich“ des GUI-Builders die Eigenschaften des Elementes an. Der Benutzer

weist dem erstellten Element über den Eigenschaftenbereich eine Aktion zu, indem er eine API aus einer Dropdown-Liste wählt (in der Dropdown-Liste sollen nur die APIs aufgelistet werden, die über die Serveroberfläche hinzugefügt wurden), danach wählt er aus einer weiteren Dropdown-Liste eine Funktion, die die ausgewählte API bietet. Der Benutzer klickt auf den „Save“-Button, um die Fernsteuerung zu speichern (b). Der Server speichert die Fernsteuerung und teilt dies dem Benutzer mit (2).

Fehlerfall:

1. Der Benutzer trägt die Bezeichnungen nicht ein: Das System kennzeichnet die Textfelder für die Bezeichnungen als Pflichtfelder.
2. Der Server Speichert die Fernsteuerung nicht: Die Fernsteuerung ist fehlerhaft (beinhaltet mehrere Elemente mit der gleichen Bezeichnung usw.) und zeigt dem Benutzer eine entsprechende Fehlermeldung an.

Optional:

- a. Der Benutzer wählt kein Mobilgerät aus der Dropdown Liste: Der Benutzer trägt die Maße in zwei separate Felder für jeweils Höhe und Breite ein. Der Server zeigt im „Gerätebereich“ des GUI-Builders eine weiße Fläche an.
- b. Der Benutzer klickt auf den Button „Save as JSON“ um die Fernsteuerung lokal zu speichern.

### Fernsteuerung Verwenden

Das Ziel ist es, dass der Benutzer eine erstellte Fernsteuerung über den Client verwendet. Dies möchte er machen, um ein „Smart-Object“ zu steuern. Damit er dies machen kann, muss er die Schritte „Anmelden“ und „Fernsteuerung Erstellen“ erfolgreich ausgeführt haben. Am Ende des Szenarios sollte der Benutzer die Fernsteuerung auf dem Mobilgerät verwenden und darüber ein „Smart-Object“ steuern können.

Der Benutzer drückt auf den „Find Controllers“-Button auf der „ControllersActivity“. Der Client ruft alle Fernsteuerungen vom Server ab und stellt diese in einer Liste auf der „ControllersActivity“ dar (1). Der Benutzer drückt auf eine Fernsteuerung in dieser Liste. Der Client öffnet die „UseControllerActivity“, auf der die ausgewählte Fernsteuerung visuell dargestellt wird (a). Der Benutzer drückt auf ein Element, das sich auf der Fernsteuerung befindet, um eine Aktion auszuführen. Der Client sendet die Aktion an das „Smart-Object“, welches die Aktion umsetzt und eine Antwortnachricht an den Client sendet (2). Der Client zeigt dem Benutzer an, dass die Umsetzung der Aktion auf dem „Smart-Object“ erfolgreich war.

Fehlerfall:

1. Der Client hat keine Fernsteuerungen auf dem Server gefunden: Es werden keine Fernsteuerungen aufgelistet.
2. Die Aktion wird vom „Smart-Object“ nicht umgesetzt: Das „Smart-Object“ sendet eine Fehlernachricht an den Client. Der Client zeigt dem Benutzer an, dass die Umsetzung der Aktion auf dem „Smart-Object“ nicht erfolgreich war.

Optional:

- a. Die ausgewählte Fernsteuerung besitzt Profile: Der Client zeigt zu der eigentlichen Fernsteuerung des Benutzers die Profile an.

## 2.4 Anforderungsanalyse

In diesem Teil des Kapitels geht es darum, aus den Szenarien funktionale und nicht funktionale Anforderungen aufzuzählen sowie um weitere Anforderungen zu ergänzen und diese genauer zu erleuchten, damit diese im System umgesetzt werden können.

### 2.4.1 Funktionale Anforderungen

- **Registrieren:**

Der Benutzer soll:

1. Sich registrieren können.
2. Ein Benutzernamen angeben können.
3. Ein Passwort angeben können.
4. Die IP-Adresse des Servers angeben können.
5. Ein QR-Code scannen können, welcher die IP-Adresse beinhaltet.

Das System soll:

6. Den Benutzer erstellen.
7. Prüfen, ob der Benutzername vergeben ist.
8. Das Passwort verschlüsseln.
9. Den QR-Code umwandeln in eine IP-Adresse.
10. Prüfen, ob der Server erreichbar ist.

- **Anmelden:**

Der Benutzer soll:

11. Sich anmelden können.
12. Sein Benutzernamen eingeben können.
13. Sein Passwort eingeben können.
14. Den Server Key per E-Mail anfordern können.

15. Den Server Key eingeben können.

Das System soll:

16. Prüfen, ob es einen Benutzer mit dem Benutzernamen gibt.
17. Prüfen, ob das Passwort für den angegebenen Benutzer korrekt ist.
18. Prüfen, ob der angegebene Key mit dem Key in der E-Mail übereinstimmt.
19. Prüfen, ob der angegebene Key mit dem Server Key übereinstimmt.

- **Benutzerrechte Ändern:**

Der Admin soll:

20. Die Berechtigung von Benutzern bearbeiten können.
21. Alle Benutzer abrufen können.

Das System soll:

22. Den Zugang für Admins auf die Admin UI autorisieren.
23. Die Benutzer in einer Liste darstellen.

- **Agent Einbinden:**

Der Benutzer soll:

24. APIs von Agenten ins System einfügen können.

Das System soll:

25. Die hinzugefügten APIs anzeigen.
26. Die hinzugefügten APIs bei der Erstellung von Fernsteuerungen zur Verfügung stellen.

- **Fernsteuerung Erstellen:**

Der Benutzer soll:

27. Fernsteuerungen erstellen können.
28. Die Fernsteuerung benennen können.
29. Die Profile benennen können.
30. Die Activitys benennen können.
31. Die Elemente benennen können.
32. Ein Mobilgerätenamen wählen können, welcher dargestellt werden soll.
33. Die Größe des Mobilgeräte-Displays separat angeben können.
34. Profile von Fernsteuerung erstellen können.
35. Weitere Activitys hinzufügen können.
36. Elemente hinzufügen können.
37. Die Größe der Elemente verändern können.
38. Die Position der Elemente verändern können.
39. Elementen Funktionen aus APIs zuweisen können.
40. Profile löschen können.
41. Activitys löschen können.
42. Elemente löschen können.

Das System soll:

43. Die Fernsteuerung visuell darstellen.
44. Das Bild des angegebenen Mobilgerätes abbilden.
45. Die Fernsteuerung in einer Baumstruktur auflisten.
46. Die Activity auf die Displaygröße des angegebenen Mobilgerätes anpassen.
47. Die Activity auf die separat angegebene Größe anpassen.

- **Fernsteuerung Speichern:**

Der Benutzer soll:

48. Fernsteuerungen speichern können.

Das System soll:

49. Beim Speichern der Fernsteuerung den Namen des Erstellers mitspeichern.
50. Mehrere Fernsteuerungen mit dem gleichen Namen speichern, solange sie von anderen Benutzern erstellt wurden.
51. Beim Speichern prüfen, ob die Fernsteuerung nicht mehrere Elemente/Activities mit derselben Bezeichnung beinhaltet.

- **Fernsteuerung Abrufen:**

Der Benutzer soll:

52. Fernsteuerungen abrufen können.

Das System soll:

53. Abgerufene Fernsteuerungen visuell anzeigen.
54. Abgerufene Fernsteuerungen in einer Baumstruktur auflisten.

- **Fernsteuerung Bearbeiten:**

Der Benutzer soll:

55. Fernsteuerungen bearbeiten können.
56. Profile die er erstellt hat, bearbeiten können.

Das System soll:

57. Dem Benutzer gestatten, alle Fernsteuerungen zu bearbeiten.
58. Dem Benutzer nur das Bearbeiten seiner Profile gestatten.

- **Fernsteuerung Verwenden:**

Der Benutzer soll:

59. Fernsteuerungen auf dem Mobilgerät verwenden können.
60. Aktionen über das Mobilgerät ausführen können.
61. Zwischen den Profilen wechseln können.
62. Zwischen den Activities wechseln können.

Das System soll:

63. Die Fernsteuerungen visuell auf dem Mobilgerät anzeigen.
64. Die Elemente auf der Fernsteuerung auf die Displaygröße des Mobilgerätes anpassen.

65. Ein visuelles Feedback geben, wenn auf dem Mobilgerät auf ein Element gedrückt und die damit verbundene Aktion ausgeführt wurde.

- **Fernsteuerung Löschen:**

Der Benutzer soll:

66. Fernsteuerungen löschen können.

67. Eigene Profile löschen können.

Das System soll:

68. Beim Löschen prüfen, ob der Benutzer auch der Ersteller des Profils ist.

## 2.4.2 Nicht-Funktionale Anforderungen

- **Mobilität:** Die Ausführung des Client muss als App von einem mobilen, nicht ortsgebundenem Gerät erfolgen, wie zum Beispiel einem Mobiltelefon oder einem Tablet, um so zu jeder Zeit „Smart-Objects“ bedienen zu können.
- **Sicherheit:** Eine Voraussetzung für den Einsatz eines solchen Systems ist die Sicherheit. So dürfen sich nicht autorisierte Nutzer keinen Zutritt in das System verschaffen, um „Smart-Objects“ zu steuern. Zudem muss jederzeit ersichtlich sein, welcher Benutzer welche Aktion getätigt hat, um dies bei einer böswilligen Nutzung nachverfolgen zu können.
- **Laufzeit:** Da der Client auf einem Mobilgerät laufen soll, spielt der Energiebedarf der App eine große Rolle, da bei Mobilgeräten wohl die wichtigste Ressource der Akku darstellt. Daher sollte die App dahingehend optimiert werden, möglichst wenig Energie zu verbrauchen, zum Beispiel beim Nachrichtenaustausch zwischen Server und Client, so dass Nachrichten nicht permanent zwischen Server und Client gesendet werden, sondern nur dann wenn es notwendig ist.
- **Übersichtlichkeit:** Sowohl auf dem Client als auch auf dem Server soll die grafische Oberfläche einem modernen und zeitgemäßen Design entsprechen. Die erstellten Fernsteuerungen müssen sich nahtlos dem Design anpassen und sollen vom Benutzer individuell angepasst werden können. Des Weiteren soll die grafische Oberfläche auf dem Server und dem Client nicht verwirrend oder überfüllt wirken, um eine schnelle und intuitive Bedienung zu ermöglichen.
- **Erweiterbarkeit:** Bei der Erweiterbarkeit handelt es sich um die Erweiterung des Systems in Form von neuen „Smart-Objects“ und somit von neuen Agenten. Die Einbindung von neuen Agenten muss daher ohne Programmierkenntnisse voraussetzen möglich sein.



## **2.5 Fazit**

Ziel der Analyse war es, das Ziel dieser Arbeit näher zu bringen. Zu diesem Zweck wurden acht Projekte, die sich wie diese Arbeit mit dem Themenbereich „Human–Computer Interaction“ beschäftigen, aufgezeigt, um zu zeigen welcher Aufwand in diesem Bereich bereits betrieben wurde. Bei der Erläuterung der Zielsetzung wurden diese acht Projekte mit eingebracht, um Ähnlichkeiten sowie Unterschiede zu dieser Arbeit aufzuzeigen.

Anschließend wurde für das hier zu realisierende System Szenarien mit unterschiedlichen Ausprägungen vorgestellt. Aus diesen wurden wiederum Anforderungen abgeleitet und formuliert, die das System erfüllen soll.

# 3 Benutzeroberflächen Gestaltung

In diesem Kapitel geht es um die Erstellung der Desktop- als auch der Mobilgerät-Oberfläche, über den die Benutzer mit dem System interagieren sollen, als Mockups, unter Berücksichtigung der im vorherigen Kapitel erstellten und spezifizierten Szenarien und Anforderungen an das System.

Es wurde entschieden eine eigene Oberfläche zu entwickeln, statt eine fertige (Open Source) zu nutzen. Eine solche fertige Oberfläche wäre zum Beispiel die „Godot Engine“, die eine Spiele Engine ist. Die Entscheidung eine eigene Oberfläche zu entwickeln wurde auf Grund der Tatsache getroffen, dass fertige Oberflächen für einen bestimmten Zweck konzipiert werden und somit oftmals eine höhere Komplexität bieten als gefordert ist. Dadurch wird eine höhere „mentale Bandbreite“<sup>5</sup> in Anspruch genommen, zudem müsste diese um die gewünschten Funktionalitäten erweitert werden, wenn diese nicht gegeben sind. Um die mentale Bandbreite wiederum zu senken, müssten Funktionen der Oberfläche entfernt werden, die für das Erreichen des Zieles nicht relevant sind.

Die beiden Applikationen sollen zudem nicht als Web-Applikation implementiert werden, da aus früheren Tests der Delay nicht tolerabel war. So hat es eine erhebliche Zeit gedauert, nachdem eine Aktion getätigt und diese tatsächlich umgesetzt wurde. Ein weiterer Grund für die nicht Implementierung der Anwendungen als Web-Applikation ist das Design der Oberflächen, da ansonsten auf Design-Kompromisse eingegangen werden müsste. So fängt dies schon bei Kleinigkeiten, wie der Zeiteingabe im Mobilgeräte-Bereich (in Abbildung 3.1 zu sehen) an. Hier werden für die Zeiteingabe unterschiedliche Elemente genutzt, auch die Anordnung der Buttons ist unterschiedlich.

---

<sup>5</sup> Ist die Zeit die eine Person benötigt um etwas zu verstehen.



Abbildung 3.1: iOS und Android UI-Unterschiede im Überblick (Rixecker (2014))

So wurde entschieden die Desktop-Applikation als Windows-Applikation umzusetzen und beim Design der Anwendung die „Windows Design Guidelines“<sup>6</sup> zu berücksichtigen, da Windows einen großen Marktanteil ausmacht, wie in Abbildung 3.2 zu sehen ist.

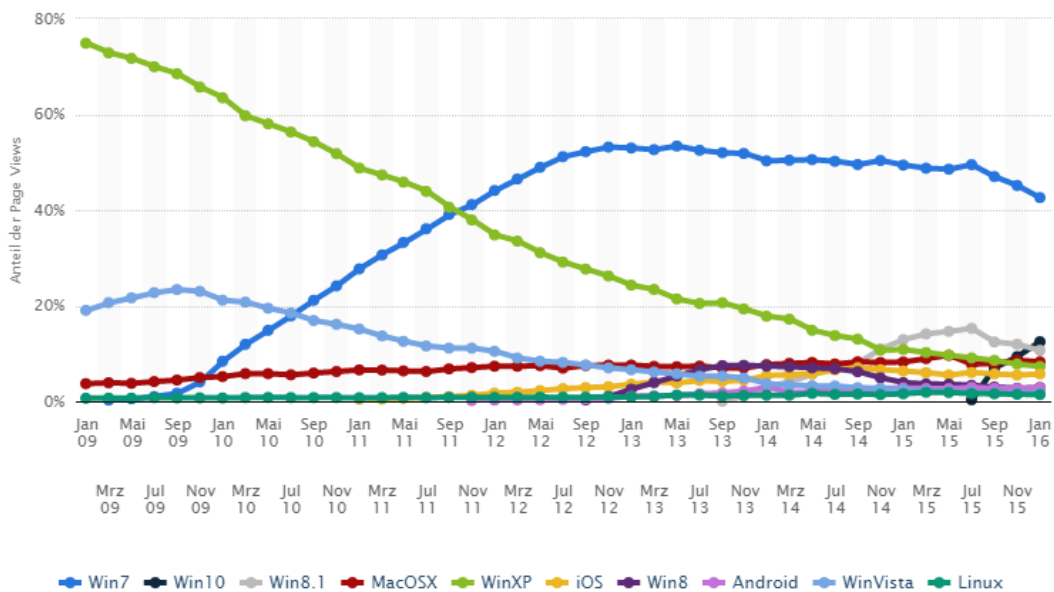


Abbildung 3.2: Marktanteile der führenden Betriebssystemversionen weltweit von Januar 2009 bis Januar 2016 (Statista: Betriebssysteme)

Bei der Mobile-Applikation wiederum, wurde entschieden auf Grund des Marktanteils (Abbildung 3.3) diese als Android-Applikation umzusetzen, da ein Ausblick auf 2020 zeigt, dass der Marktanteil weiter steigen wird. Der Vorteil hierbei ist, dass nahe am System ent-

<sup>6</sup> Windows Design Guidelines – (Microsoft, vgl.)

wickelt werden kann und somit spezifische Funktionen des Systems genutzt werden können. Des Weiteren wurde entschieden, dass „Material Design“<sup>7</sup> von Google zu berücksichtigen, bei der Entwicklung der Oberfläche. Weitere Gründe für die Nutzung von Android war der „Open Source“-Aspekt sowie Apples strenge und kostenpflichtige Entwicklungspolitik und deren Gerätebeschränkung auf Apple Produkte.

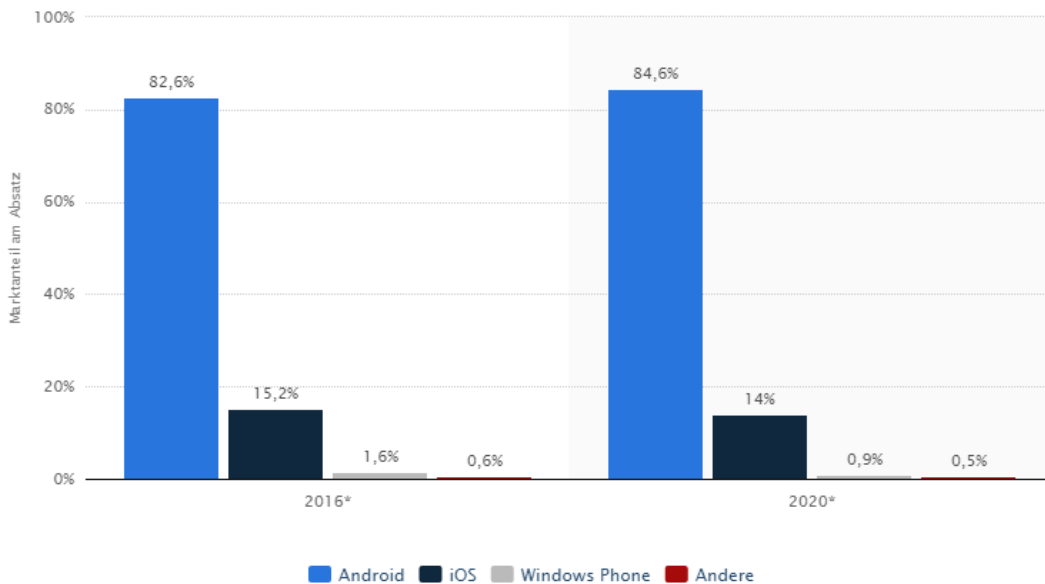


Abbildung 3.3: Prognose zu den Marktanteilen der Betriebssysteme am Absatz vom Smartphones weltweit in den Jahren 2016 und 2020 (Statista: Smartphone Betriebssysteme)

Neben den beiden genannten Design Prinzipien soll der „DIN EN ISO 9241-110“<sup>8</sup>-Standard mit einbezogen werden. Dieser beschreibt Qualitätsrichtlinien zur Sicherstellung der Ergonomie interaktiver Systeme und zeigt die sieben Grundsätze der Dialoggestaltung auf.

Der Grundsatz „Aufgabenangemessenheit“ besagt, dass „ein Dialog dann aufgabenangemessen ist, wenn er den Benutzer unterstützt, seine Arbeitsaufgabe effektiv und effizient zu erledigen“. Hierfür wird empfohlen, die zu tätigende Aufgabe zu identifizieren, wann diese erfolgreich ist und den effizientesten Weg zu finden der zu diesem Ziel führt. Diese Kriterien stellen somit einen Anwendungsfall dar, da auch in diesen auf diese Punkte eingegangen werden muss. Die Anwendungsfälle sind wiederum im Unterkapitel 2.3 zu finden.

Labels die oberhalb jeder grafischen Oberfläche dargestellt werden sollen, sollen dafür sorgen, dass ein Benutzer zu jeder Zeit erkennen kann auf welcher Oberfläche er sich befindet. Ausführlich beschriftete Buttons wiederum sollen dem Anwender verdeutlichen, wo diese

<sup>7</sup> Material Design – (Google, vgl.)

<sup>8</sup> DIN EN ISO 9241-110 - (ERGO-Online, vgl.)

ihn hinführen. Des Weiteren sollen Rückmeldungen über den Erfolg einer durchgeführten Aktion angezeigt werden. Dies soll ein Gefühl der Sicherheit beim Benutzer schaffen. Diese Punkte wurden aus dem Grundsatz „Selbstbeschreibungsfähigkeit“ geschlussfolgert, der besagt, dass „ein Dialog dann selbstbeschreibungsfähig ist, wenn jeder einzelne Dialogschritt durch Rückmeldung des Dialogsystems unmittelbar verständlich ist oder dem Benutzer auf Anfrage erklärt wird“, zudem muss der Anwender immer folgende drei Dinge wissen: „Wo komme ich her?“, „Wo bin ich?“ und „Wo kann ich von hier aus hin?“.

Der Grundsatz „Erwartungskonformität“ der Dialoggestaltung besagt, dass „ein Dialog dann erwartungskonform ist, wenn er konsistent ist und den Merkmalen des Benutzers entspricht, zum Beispiel den Kenntnissen aus dem Arbeitsgebiet, der Ausbildung und der Erfahrung des Benutzers sowie den allgemein anerkannten Konventionen“. Daher wurde bei der Gestaltung des GUI-Builders, über den die Fernsteuerungen erstellt werden sollen, an herkömmlichen GUI-Buildern, wie „JavaFx Scene Builder“ und „Android Studio“ orientiert. Da der hier designte GUI-Builder eine ähnliche Komplexität aufweist, zu den oben genannten GUI-Buildern, ist dieser für Benutzer, die bereits Erfahrung im Umgang mit GUI-Buildern gesammelt haben gedacht. Der Einstieg jedoch für Neulinge soll durch Hilfstexte erleichtert werden.

Bei der Gestaltung der Mobile-Applikation wurde kein Referenzmodell herangezogen. Daher wurde bei der Gestaltung des Designs auf einen logischen Aufbau sowie einen logischen Ablauf geachtet, um dem Benutzer einen schnellen und leichten Einstieg zu ermöglichen. Dies wird im Grundsatz „Lernförderlichkeit“ erwähnt, der besagt, dass „ein Dialog dann lernförderlich ist, wenn er den Benutzer beim Erlernen des Dialogsystems unterstützt und anleitet“.

Die „Fehlertoleranz“ besagt, dass „ein Dialog dann fehlertolerant ist, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand durch den Benutzer erreicht werden kann“. So sollen bei der Erstellung von neuen Fernsteuerungen, Profilen, Layouts und Elementen immer angezeigt werden, ob der vom Benutzer eingetragene Name vergeben ist. Fehlermeldungen sollen zudem in einem separaten Dialogfenster dargestellt werden, den der Benutzer wegeklicken muss, damit dieser nicht ignoriert werden kann.

Für den Grundsatz der „Steuerbarkeit“ sollen alternative Steuermöglichkeiten, wie Kontextmenüs und Tastenkürzel für relevante Funktionen, dem Benutzer angeboten werden, da der Grundsatz besagt, dass „ein Dialog dann steuerbar ist, wenn der Benutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist“. Zudem sollen Undo- sowie Redo-Funktionen angeboten werden, um Aktionen rückgängig machen zu können oder rückgängig gemachte Aktionen wiederherzustellen.

Der Grundsatz der „Individualisierbarkeit“ besagt, dass „ein Dialog dann individualisierbar ist, wenn das Dialogsystem Anpassungen an die Erfordernisse der Arbeitsaufgabe, individuelle Vorlieben des Benutzers und Benutzerfähigkeiten zulässt“. Die Oberfläche auf denen die Fernsteuerungen erstellt werden sollen, soll aus Bereichen bestehen, die umpositioniert und in der Größe veränderbar sein sollen, damit ein Benutzer diese nach seinen Wünschen und Vorlieben anpassen kann. Bei der Mobil-Anwendung wiederum, soll das Design der Oberfläche und somit das Aussehen der Elemente, wie die Farbe, anpassbar sein. Zudem sollen wie bereits erwähnt, Fernsteuerungen frei gestaltbar sein, damit ein Benutzer für jede Situation seiner Meinung nach die idealste Fernsteuerung erstellen kann.

### 3.1 Desktop Anwendung

Die Server-Seite stellt die Desktop Applikation da und besteht aus drei Oberflächen, wovon die eine nur für Admins nutzbar ist. So stellen die anderen beiden Oberflächen den GUI-Builder und den Server dar.

Da festgelegt wurde die „Windows Design Guidelines“ zu übernehmen, wurden die grafischen Oberflächen in diesem Teil des Kapitels nach diesen Design-Prinzipien gestaltet.

Diese besagen, dass Abstand und Positionierung von Elementen von anderen Elementen abhängig sind. So sollen zum Beispiel Labels an der Grundlinie einer Text-Box ausgerichtet werden, diese befindet sich ein paar Pixel über dem tatsächlichen unteren Rand. Da keine einheitlichen Regelungen für Abstände und Ausrichtungen bestehen, wird empfohlen Container wie „TableLayoutPanel“ zu verwenden und die Elemente danach auszurichten.

Auch die Größe der Elemente spielt eine Rolle. So besagen die Design-Prinzipien von Windows, dass ein Button der aus der Toolbox entnommen wurde, schon die ideale Höhe und Breite besitzt, zudem wird davon abgeraten Buttons größer zu ziehen oder andersfarbig darzustellen, um die Aufmerksamkeit des Benutzers drauf zu lenken.

Da eine Anwendung aus mehreren Elementen bestehen kann, wird in den Design-Prinzipien empfohlen diese intuitiv zu gruppieren. So sind zum Beispiel Tabs für die Gruppierung von Elementen sehr gut geeignet, dadurch erspart man sich ein Label, da die Bezeichnung der Gruppe im Tab steht. Die Verschachtelung von Gruppen wiederum wird abgeraten. Eine Ausnahme ergibt sich bei einer Obergruppe die höchstens zwei bis drei Untergruppen beinhaltet.

Den wichtigsten Aspekt bei der Gestaltung einer Benutzeroberfläche stellt die Intuitivität dar. So kann für eine intuitive Bedienung eine Farbcodierung verwendet werden, wie zum Beispiel in Windows der Button zum Schließen eines Dialogfensters.

Des Weiteren besagen die Windows Design-Prinzipien für eine intuitive Bedienung eine simple Sprache zu verwenden, da eine einfache Sprache dazu führt Sachverhalte schneller zu begreifen. Auch die Beschriftung von Text-Boxen und Gruppen sind förderlich, um dem Benutzer mit wenigen Worten etwas zu vermitteln. Auch etwas Vertrautes wiederzusehen, ist förderlich.

Ein wichtiger Faktor des intuitiven Designs ist die mentale Bandbreite, diese soll so gering wie möglich gehalten werden, um dem Benutzer eine bessere Benutzererfahrung zu bieten. Hierfür werden auf der Webseite weitere Tipps vorgeschlagen, die relevant für die Gestaltung einer benutzerfreundlichen Oberfläche sind.

### 3.1.1 Server-Benutzeroberfläche

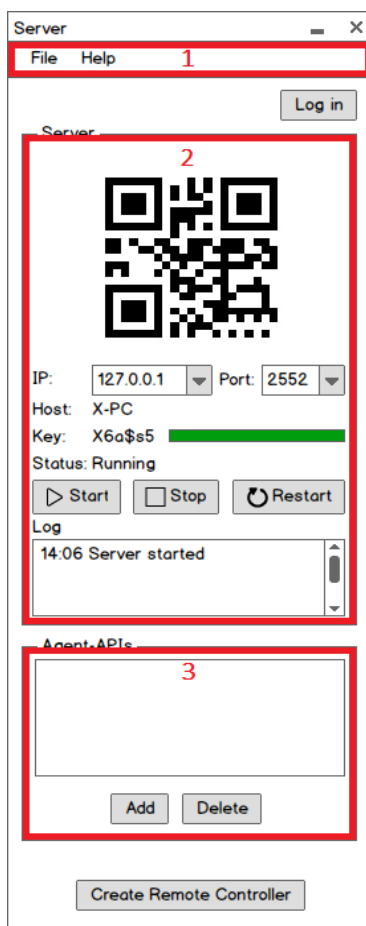


Abbildung 3.4: Server-Benutzeroberfläche nach dem Start

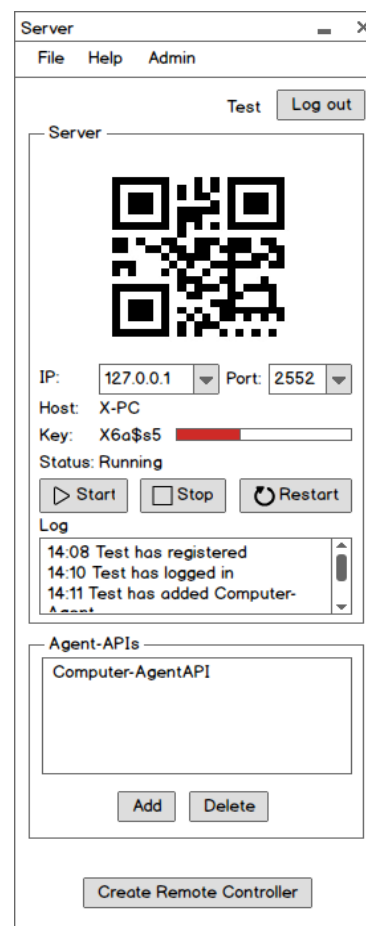


Abbildung 3.5: Server-Benutzeroberfläche während des Betriebs

Über die Serveroberfläche, die in Abbildung 3.4 und Abbildung 3.5 dargestellt wird, soll die Middleware gestartet und beendet werden sowie APIs von Agenten hinzugefügt und entfernt werden können. Diese APIs dienen der Erstellung von Fernsteuerungen, dazu später mehr in Unterkapitel 3.1.3.

Das Serverfenster soll sich in der Größe nicht manipulieren lassen und somit auch nicht maximiert werden können, da ansonsten alle darin befindlichen Elemente in die Länge gezogen werden würden, dies würde einen schlechten Eindruck auf den Benutzer machen. Die Server-Benutzeroberfläche ist die Startseite der Desktop Applikation und somit das erste was der Benutzer auf dem Desktop nach dem Start der Anwendung sieht.

Die Server-Benutzeroberfläche ist in drei Bereiche aufgeteilt. Der erste Bereich ist die „Menüleiste“, der zweite ist der „Server“-Bereich und der dritte der „Agent-APIs“-Bereich. Bei der Gruppierung der beiden Bereiche Server und Agent-APIs wurden „Group“-Elemente genutzt anstelle von Tabs, da dies laut den „Windows Design Guidelines“ es dem Benutzer ermöglicht einen schnelleren Überblick über die Bereiche zu erhalten, da nicht zwischen Tabs gewechselt werden muss. Die drei Bereiche werden weiter unten genauer erläutert und in ihrer funktionsweise beschrieben.

Außerhalb dieser drei Bereiche befinden sich zwei Buttons. Der „Log in“-Button soll dem Öffnen des „Log in“ Fensters in Abbildung 3.6a dienen. Auf dieser Oberfläche sollen sich Benutzer einloggen können. Nach dem erfolgreichen Anmelden wiederum, soll links vom „Log In“-Button der Name des angemeldeten Benutzers stehen. Zudem soll die Bezeichnung des Buttons in „Log out“ umgeändert werden sowie dessen Funktion. So soll sobald auf den Button gedrückt wird, das „Log out“-Fenster, das in b) zu sehen ist, sich öffnen, damit der Benutzer über dieses Dialog Fenster sich abmelden kann. Das in c) wiederum zu sehende Fenster soll sich öffnen, falls beim Anmeldevorgang falsche Daten angegeben wurden. Bevor ein Benutzer sich einloggen kann, muss er sich über sein Mobilgerät beim „Authentication-Agenten“ registrieren. Dies soll so gemacht werden, um zu prüfen, ob der Benutzer im Besitz eines Mobilgerätes ist.

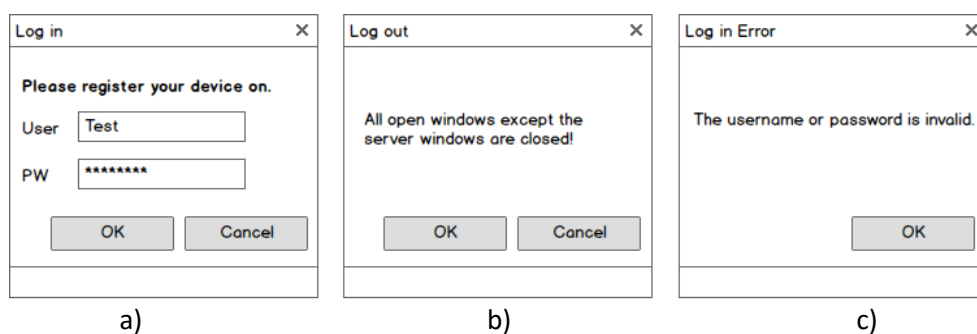


Abbildung 3.6: Log in Fenster der Server-Benutzeroberfläche



Falls das „Log in“ oder das „Log out“ Fenster bereits geöffnet wurde aber sich im Hintergrund befindet, soll beim Klick auf den Button „Log in“ bzw. „Log out“ das Fenster wieder in den Vordergrund geholt werden, anstelle ein weiteres Fenster zu öffnen. Das „Log in Error“ Fenster wiederum soll sich immer im Vordergrund befinden. Damit dieser nicht vom Benutzer ignoriert werden kann, sollen keine Aktionen auf anderen Oberflächen getätigt werden können, die zu der Anwendung gehören. Diese drei Fenster sollen nicht minimiert, maximiert oder in der Größe manipuliert werden können, da ansonsten eine Fläche entstehen würde, auf der sich nichts befindet. Falls die Middleware nicht läuft, soll beim Klicken auf den „Log in“-Button eine Meldung in einem separaten Fenster erscheinen, die besagt das der Server gestartet sein muss, um sich anmelden zu können. Dieses Fenster soll die Eigenschaften und den Aufbau des „Log in Error“ Fensters haben.

Der „Create Remote Controller“-Button soll dazu dienen die gleichnamige Oberfläche zu öffnen. Dieser Button beinhaltet den vollen Titel der Oberfläche, damit erkennbar ist, dass sich die „Create Remote Controller Benutzeroberfläche“ öffnet sobald auf diesen Button geklickt wird. Zudem wurde ein solch ausführlicher Name gewählt, damit der Benutzer ein Bild davon bekommt was geschehen könnte, wenn er auf diesen Button klickt.

## 1. Menüleiste

Die in Abbildung 3.7a zu sehende „Menüleiste“, die den Reiter „Files“ aufgeklappt darstellt, bietet Funktionalitäten der Server-Benutzeroberfläche an und zeigt zudem deren Tastenkürzel für eine alternative und schnelle Bedienung an. In b) ist der Reiter „Help“ aufgeklappt zu sehen, dieser bietet die Funktion „About Server“. Durch einen Klick auf diesen, soll sich das Infenster des Servers öffnen, auf dem die Version des Servers stehen soll. Der „Admin“ Reiter, der in c) aufgeklappt ist, bietet die Funktion „Admin UI“ an. So soll sobald auf diesen geklickt wird, die Admin-Benutzeroberfläche sich öffnen. Dieser Reiter soll nur für Admins sichtbar sein.

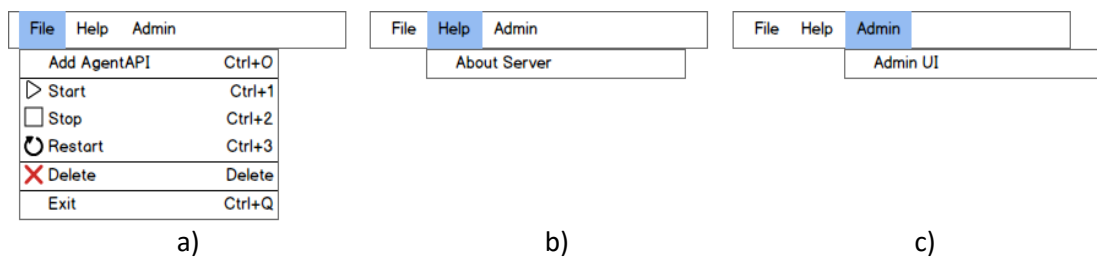


Abbildung 3.7: Menüleiste der Server-Benutzeroberfläche

## 2. Server Bereich

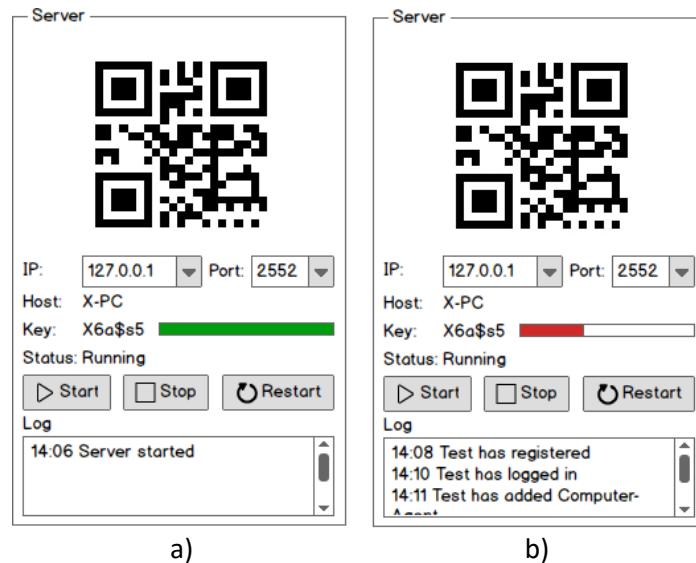


Abbildung 3.8: Server Bereich der Server-Benutzeroberfläche

Der in Abbildung 3.8a zu sehende „Server“-Bereich soll dazu dienen, die Middleware zu starten, damit über Nachrichten mit den Agenten kommuniziert werden kann.

Über die Dropdown Listen sollen die IP-Adresse und der Port gewählt werden können, über den dann die Middleware laufen soll, so kann die lokale oder die Internet-Adresse verwendet werden. Zudem soll es möglich sein, die Dropdown-Listen als Textfeld zu nutzen, um so die IP-Adresse und den Port einer anderen Middleware einzutragen und die Agenten dann über diese Middleware laufen zulassen, wenn auf den „Start“-Button gedrückt wird. Für das Anzeigen der IP-Adresse und des Ports wurde jeweils eine „Dropdown-Liste“ gewählt, da auf einem Blick zu erkennen sein soll, unter welcher Adresse der Server gerade läuft. Bei einer Liste wiederum würden alle Adressen untereinander aufgelistet werden, was es unübersichtlich machen würde.

Der QR-Code soll sich der aus den „Dropdown-Listen“ ausgewählten Adresse anpassen. Dies soll eine einfache und unkomplizierte Variante darstellen, um sich über das Mobilgerät mit der Middleware zu verbinden. Die „Host“-Beschriftung soll angezeigt werden, damit die Benutzer sehen können, auf welchem Computer der Server läuft.

Der „Key“ soll dazu dienen, unautorisierten Personen den Zugriff auf den Server über das Mobilgerät mit einem fremden Account zu verwehren. Dieser soll beim Anmelden über das Mobilgerät abgefragt werden, was zu Folge hat, dass die Person die sich auf dem Server anmelden möchte in der Nähe des Servers aufhalten muss. Alternativ soll der Key per E-Mail angefordert werden können. So ist die Sicherheit immer noch gewährleistet, da der

unautorisierte Benutzer das E-Mail-Passwort kennen muss, um an den Key zu gelangen. Zudem soll der Key alle 60 Sekunden neu generiert werden, dies soll durch den Balken daneben verdeutlicht werden, der seine Farbe von Grün in Rot (b) umändern soll, sobald ein Drittel der Zeit abgelaufen ist. Der Key soll ein sechsstelliger Code sein, der aus Groß- und Klein-Buchstaben (Anzahl: 52) sowie Zahlen (Anzahl: 10) und Sonderzeichen (Anzahl: 32) bestehen kann. Dies macht  $94^6$  Kombinationen möglich<sup>9</sup>. So braucht man mit einem Brute-Force verfahren ca. 6 Minuten um den Key zu knacken. Da der Key aber alle 60 Sekunden generiert werden soll, hat dieser sich in der Zeit 6-mal geändert.

Der „Status“ soll angezeigt werden, damit der Benutzer sehen kann, in welchem Zustand sich der Server momentan befindet. So kann sich der Server in den folgenden Zuständen befinden: „Running“ für laufend, „Stopped“ für nicht laufend und „Restarting“ für neustartend.

Damit der Status geändert werden kann, befinden sich unter der Statusanzeige drei Buttons, jeweils einer für jeden Status. Um den Fokus, wie in den „Windows Design Guidelines“ beschrieben, auf den „Start“-Button zu legen, befindet dieser sich ganz links. Der „Start“-Button hat die Bezeichnung „Start“ und nicht „Run“, damit er mit dem Windows Start-Button assoziiert wird und somit leichter zu verstehen ist. Die Buttons beinhalten Symbole, um im Voraus ihre Funktion auf einem Blick zu erahnen. Alternativ könnten für die Buttons „Start“ und „Stop“ ein „Toggle-Button“ verwendet werden, der die beiden Buttons zusammenfasst. Somit wäre eine Statusanzeige nicht mehr nötig, da über den „Toggle-Button“ ersichtlich wäre, in welchem Zustand sich der Server gerade befindet, dies wurde aber als unnatürlich empfunden und daher verworfen.

Zum Schluss folgt der „Log“-Bereich, dieser soll dazu dienen Server Informationen sowie Aktionen des Benutzers anzuzeigen (b), damit jederzeit nachvollziehbar ist, was auf dem Server geschieht bzw. geschehen ist. Die aktuellste Benachrichtigung soll hierbei immer unten in der Liste dargestellt werden.

### 3. Agent-APIs Bereich

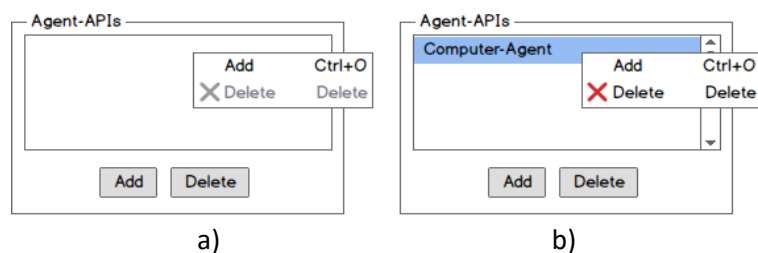


Abbildung 3.9: Agent-APIs Bereich der Server-Benutzeroberfläche

<sup>9</sup> Brute-Force-Angriff - (Password-Depot, vgl.)

Der in Abbildung 3.9a zu sehende „Agent-APIs“-Bereich soll dazu dienen APIs von Agenten in das System einzubinden, um diese bei der Erstellung von Fernsteuerungen zu nutzen. Hierfür befindet sich in diesem Bereich eine Liste, in der die APIs aufgelistet werden sollen. Hierbei wurde für eine Liste entschieden und kein anderes Element, da eine Liste es ermöglicht mehrere Elemente auf einen Blick anzuzeigen.

Die darauf folgenden Buttons „Add“ und „Delete“ sollen dem Hinzufügen neuer Agenten und Löschen vorhandener Agenten dienen. So soll, wenn auf den „Add“-Button geklickt wird, sich das Standard-„Windows-Öffnen-Dialogfenster“ öffnen, über den der Benutzer dann eine API eines Agenten wählen und der Liste hinzufügen können soll. Um eine API zu löschen soll eine API in der Liste ausgewählt werden und dann über den „Delete“-Button aus diesem entfernt werden. Für eine alternative und schnelle Bedienung soll sich, sobald in der Liste ein Rechtsklick ausgeführt wird, ein „Kontextmenü“ öffnen. Es soll sich auch ein „Kontextmenü“ öffnen, wenn auf eine API in der Liste ein Rechtsklick ausgeführt wird, wie in b) zu sehen ist.

### 3.1.2 Admin-Benutzeroberfläche

Die Admin-Benutzeroberfläche, in Abbildung 3.10a zu sehen, soll den Administratoren dazu dienen die Benutzer zu verwalten und deren Berechtigungen zu ändern. Hierfür sollen die Benutzer für eine schnelle Übersicht in einer Liste aufgelistet werden.

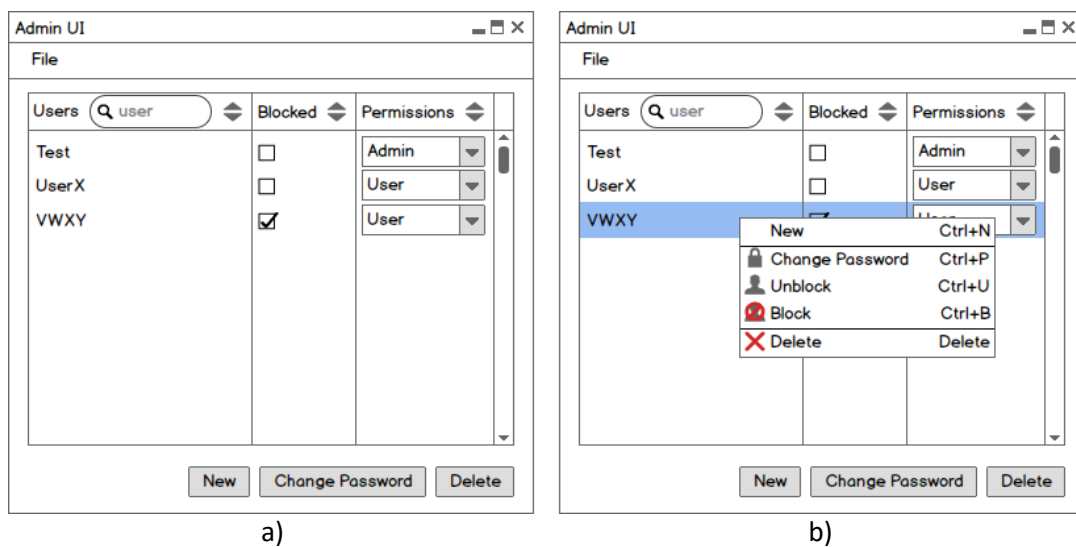





Abbildung 3.10: Admin-Benutzeroberfläche

Die Liste ist in drei Bereiche eingeteilt, die sich in ihrer Breite verändern lassen sollen. Die in der Liste dargestellten Benutzer sollen alphabetisch angeordnet dargestellt werden, damit

diese laut den „Windows Design Guidelines“ schneller gefunden werden können. Sollte auf einen Benutzer in der Liste geklickt werden, soll dieser hervorgehoben werden. Sollte wiederum per Rechtsklick auf einen Benutzer geklickt werden, soll sich, wie in **b)** zu sehen, ein Kontextmenü öffnen.

Der erste Bereich „Users“ zeigt die Namen der Benutzer an. Über die Suchfunktion soll nach bestimmten Benutzern gesucht werden können. Der so gefundene Benutzer soll dann als einziges Element in der Liste dargestellt werden. Über das -Symbol daneben soll es möglich sein, die Benutzer alphabetisch absteigend oder aufsteigend sortiert darzustellen.

Der zweite Bereich „Blocked“ soll den Zustand in dem sich der Benutzer befindet anzeigen. Hierfür sollen Checkboxen verwendet werden. So hat der Admin die Möglichkeit einen Haken für geblockt zu setzen. Gleichzeitig ermöglicht die Verwendung von Checkboxen einen schnelleren Überblick über den Zustand der Benutzer. Über das -Symbol soll es möglich sein die Benutzer so zu sortieren, dass entweder alle geblockten Benutzer oben oder ganz unten angezeigt werden in der Liste.

Der dritte Bereich „Permissions“ soll die Berechtigungen eines Benutzers auflisten. Diese sollen in einer Dropdown-Liste dargestellt werden, damit der Admin sehen kann welcher Benutzer welche Berechtigung hat und zudem wenn nötig die Berechtigung so schnell ändern kann. Über das -Symbol soll es möglich sein die Benutzer nach ihrer Berechtigung sortiert darzustellen. Die Berechtigungen sind einfach gehalten und lauten wie folgt:

- User:
  1. Darf Fernsteuerungen erstellen.
  2. Darf alle Fernsteuerungen bearbeiten und verwenden
  3. Darf nur seine Profile bearbeiten und verwenden.
  4. Darf Server starten und beenden.
  5. Darf APIs hinzufügen und löschen.
  
- Admin:

Darf, zusätzlich zu den Dingen die ein „User“ darf:

  6. Alle Profile bearbeiten und verwenden.
  7. Die Admin UI nutzen.
  8. Benutzer erstellen, löschen und blocken.
  9. Benutzerberechtigungen und Passwörter ändern.

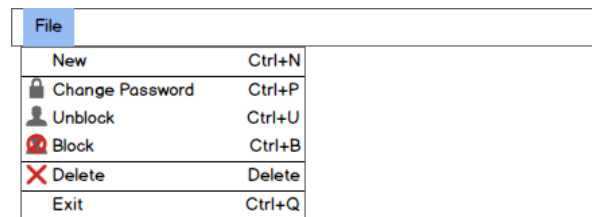


Abbildung 3.11: Menüleiste der Admin-Benutzeroberfläche

Die in Abbildung 3.11 zu sehende „Menüleiste“ soll Funktionalitäten der „Admin UI“-Oberfläche bereitstellen, da diese Funktionalitäten bereits erläutert wurden, wird hier auf eine erneute Erläuterung verzichtet.

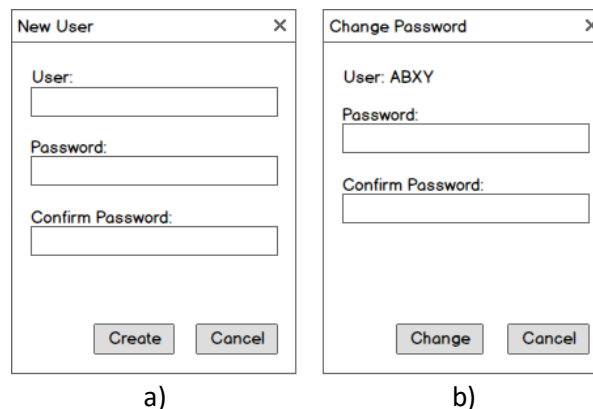


Abbildung 3.12: New User Fenster der Admin-Benutzeroberfläche

Das in Abbildung 3.12a zu sehende „New User“-Fenster soll sich öffnen, sobald auf den „New“-Button gedrückt wird. Dieses soll der Erstellung von neuen Benutzern dienen.

Das in b) zu sehende „Change Password“-Fenster soll dem ändern des Passwortes eines Benutzers dienen und soll sich öffnen, sobald ein Benutzer in der Liste ausgewählt wurde und auf den Button „Change Password“ geklickt hat. Solange kein Benutzer ausgewählt ist, soll der Button ausgegraut dargestellt werden.

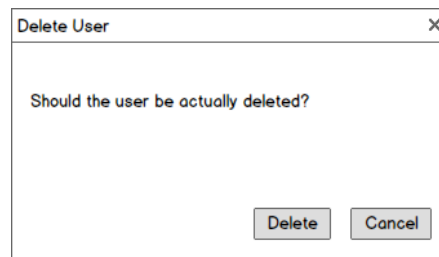


Abbildung 3.13: Delete User Fenster der Admin-Benutzeroberfläche

Sobald auf den „Delete“-Button gedrückt wird, soll sich das in Abbildung 3.13 zu sehende Fenster öffnen. Dieses Fenster soll dazu dienen den Admin zu fragen, ob dieser sicher ist den Benutzer zu löschen, da es sein kann, dass er aus Versehen auf den Button gekommen ist.

### 3.1.3 Create Remote Controller-Benutzeroberfläche

Über die „Create Remote Controller“-Oberfläche, sollen Fernsteuerungen für Mobilgeräte erstellt werden, bei denen die APIs, die über die Server-Benutzeroberfläche hinzugefügt wurden, mit einbezogen werden sollen, um so über die erstellten Fernsteuerungen auf dem Mobilgerät Agenten zu bedienen.

Die „Create Remote Controller“-Oberfläche soll dargestellt werden, wenn auf der Serveroberfläche auf den gleichnamigen Button gedrückt wird. Die Oberfläche soll in der Größe änderbar und somit maximierbar sein. Hierbei sollen sich die Bereiche der Oberfläche der neuen Größe anpassen. In Abbildung 3.14 und Abbildung 3.15 wird die Oberfläche dargestellt.

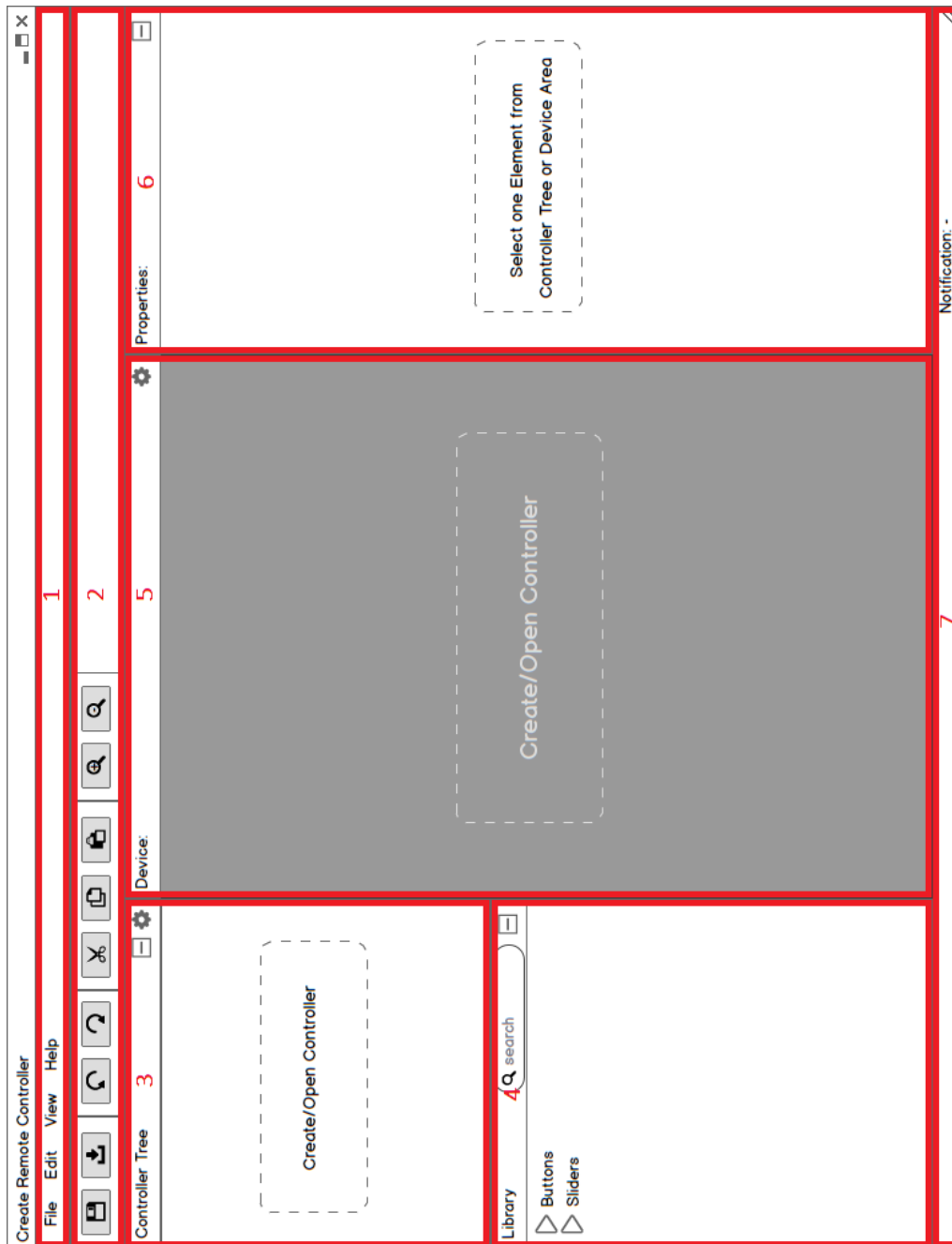


Abbildung 3.14: Create Remote Controller-Benutzeroberfläche nach dem öffnen



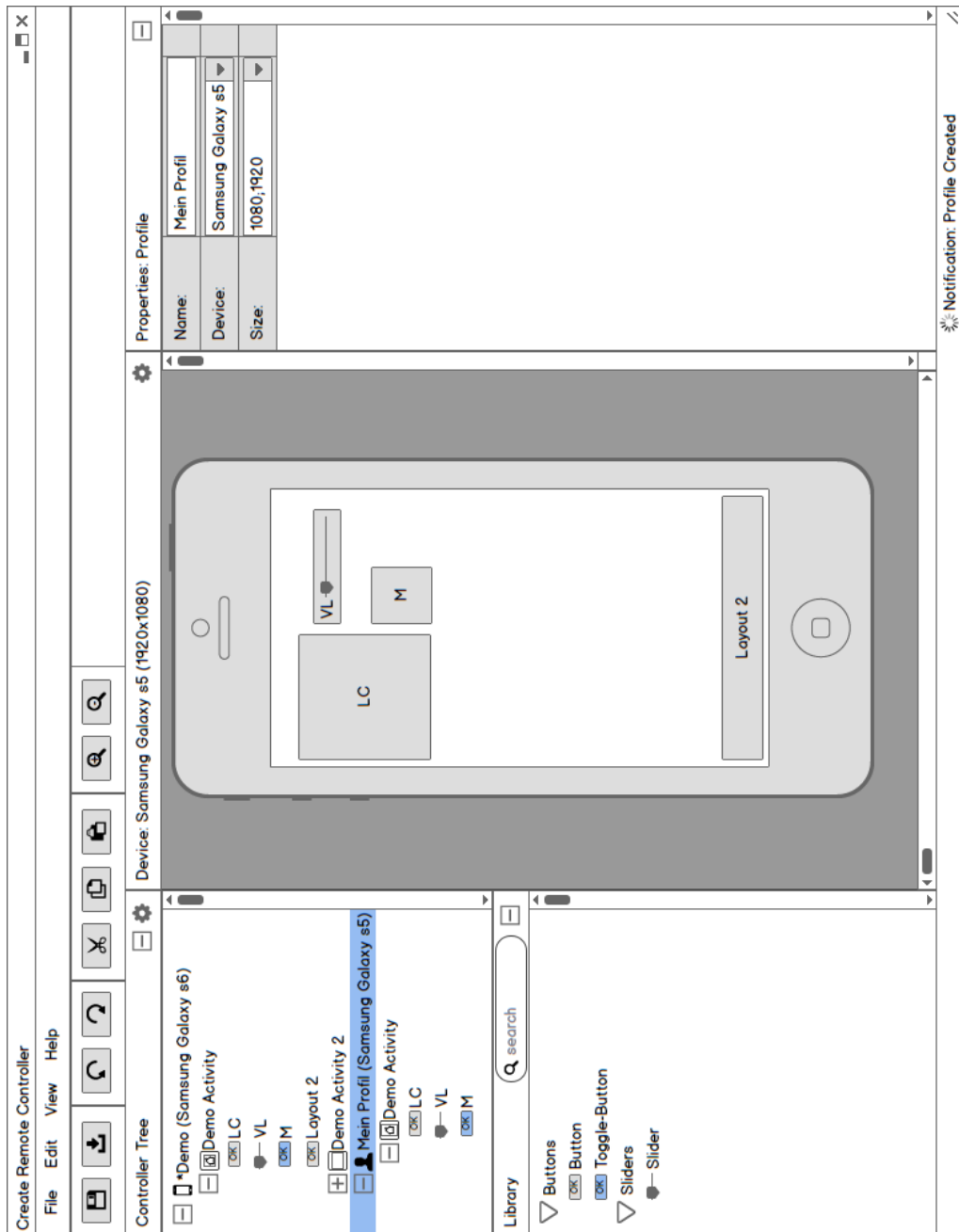


Abbildung 3.15: Create Remote Controller-Benutzeroberfläche während des Gebrauchs

Die Oberfläche ist in sieben Bereiche aufgeteilt, wovon die Bereiche drei bis sechs umpositioniert werden können sowie in der Größe veränderbar sein sollen. So soll der Benutzer die Möglichkeit haben die Oberfläche zu individualisieren und auf seine Bedürfnisse anzupassen. Die getätigten Änderungen sollen in einer Config-Datei vermerkt und beim Öffnen der

Oberfläche ausgelesen werden. Die in den Abbildungen dargestellte Oberfläche bildet das Standard-Layout der Oberfläche. Die sieben Bereiche werden ab Seite 44 genauer erläutert und in ihrer funktionsweise beschrieben.

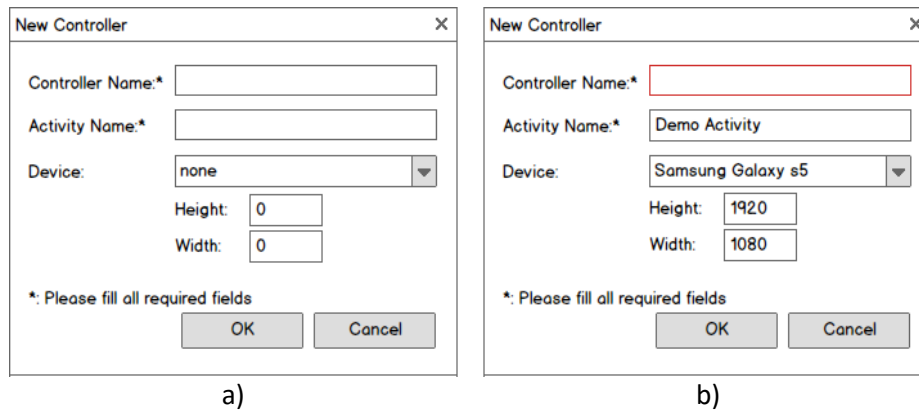


Abbildung 3.16: New Controller Fenster der Create Remote Controller-Benutzeroberfläche

Das Dialogfenster „New Controller“ in Abbildung 3.16a soll der Erstellung neuer Fernsteuerungen dienen. Dieses Fenster soll geöffnet werden, sobald auf den „New Controller“-Button gedrückt wurde.

Auf der Oberfläche befinden sich zwei Textfelder. Jeweils eines für den Namen der Fernsteuerung und den der Activity, diese Felder sind Pflichtfelder. Dies soll durch das \*-Symbol verdeutlicht werden, dass auf die Zeile verweist, die dies besagt. Beim Öffnen der Oberfläche soll in diesen Textfeldern nichts stehen. Beim Klick auf den „OK“-Button soll geprüft werden, ob diese ausgefüllt wurden. Falls nicht, soll wie in b) zusehen das entsprechende Textfeld rot markiert dargestellt werden.

Der Name der eingetragenen Activity soll die Main-Activity darstellen. Das bedeutet, dass diese Activity auf dem Mobilgerät als erste Activity geöffnet dargestellt werden soll, wenn die Fernsteuerung ausgewählt wurde, die sie beherbergt.

Unter den beiden Textfeldern befindet sich eine Dropdown-Liste. Diese soll die Namen der Mobilgeräte auflisten, die sie vom Device-Agenten anfordert, sobald die Oberfläche geöffnet wird. Hierbei sollen nicht nur die Namen, sondern auch die dazugehörigen Maße der Mobilgeräte mitgeliefert und in den beiden unteren Feldern für die Größe angezeigt werden. Falls die Nachricht verloren gegangen sein sollte (Time out), sollen die Namen und Maße noch mal angefordert werden, wenn auf die Dropdown-Liste geklickt wird. Wenn nun auf den „OK“-Button gedrückt wird, soll vom Device-Agenten ein Bild zum Mobilgerätenamen angefordert und an die „Create Remote Controller“-Oberfläche weitergegeben werden, mit den dazugehörigen Maßen. Die Bilder sollen nicht direkt mit den Namen zusam-

men angefordert werden, da die Nachricht zu groß werden würde und es eine gewisse Zeit dauern würde, diese Nachricht zu übertragen.

Damit Fernsteuerungen auch erstellt werden können falls die Middleware nicht laufen sollte, soll alternativ die Größe in den Feldern für die Höhe und Breite manuell eingetragen werden können. So sollen dann die Maße weitergegeben werden, wenn auf den Button „OK“ gedrückt wird.

Über ein separates Dialog Fenster, wie bei der Server-Benutzeroberfläche, sollen dem Benutzer Fehlermeldungen angezeigt werden, die vom Controller-Agenten stammen. Dieses Dialog Fenster soll die Bezeichnung „Controller-Agent Error“ tragen sowie dieselben Eigenschaften aufweisen wie das „Log in Error“ Fenster. Beispiel Fehlermeldungen die hier angezeigt werden können sollen wie folgt aussehen:

- Die Fernsteuerung „Name der Fernsteuerung“ besitzt mehrere Activitys mit der Bezeichnung „Name der Activity“. Bitte ändern sie die Bezeichnung der doppelten Activitys.
- Die Fernsteuerung „Name der Fernsteuerung“ besitzt mehrere Elemente mit der Bezeichnung „Name des Elementes“ in der Activity „Name der Activity“. Bitte ändern sie die Bezeichnung der doppelten Elemente.

Die Fehlermeldungen sollen dem identifizieren eines Fehlers dienen. Weitere Fehlermeldungen sollen daher wie in den obigen Beispielen ausführlich beschrieben dargestellt werden, damit ein Benutzer schnell erkennen kann wo der Fehler liegt, um diesen beheben zu können.

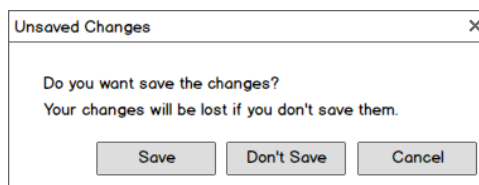


Abbildung 3.17: Unsaved Changes Fenster der Create Remote Controller-Benutzeroberfläche

Das in Abbildung 3.17 zu sehende „Unsaved Changes“ Fenster soll sich öffnen, sobald der Benutzer versucht die „Create Remote Controller“-Benutzeroberfläche zu schließen und ungespeicherte Änderungen vorliegen. Dieses Fenster soll die gleichen Eigenschaften wie das „Log in Error“ Fenster haben.

Durch einen Klick auf den „Save“-Button sollen die Fernsteuerungen gespeichert werden, bei denen Änderungen vorliegen. Diese sollen dann an den Controller-Agenten gesendet und auf diesem gespeichert werden. Falls die Middleware nicht laufen oder kein Benutzer

über die Serveroberfläche angemeldet sein sollte, soll das „Windows-Speichern-Dialogfenster“ sich öffnen, wenn auf „Save“ geklickt wird. Über diesen soll der Benutzer dann die Möglichkeit haben, die Fernsteuerung lokal als JSON-Datei zu speichern. Über den „Don't Save“-Button wiederum sollen die Änderungen nicht übernommen werden und über den „Cancel“-Button soll der Vorgang abgebrochen werden.

## 1. Menüleiste

Die Menüleiste soll Funktionalitäten der „Create Remote Controller“-Oberfläche bereitstellen sowie für eine alternative und schnelle Bedienung Tastenkürzel anzeigen. Die Strukturierung der Elemente in den Reitern wurde der Entwicklungsumgebung „Eclipse“ nachempfunden und dementsprechend so gestaltet. Die „Menüleiste“ ist in Abbildung 3.18 zu sehen.

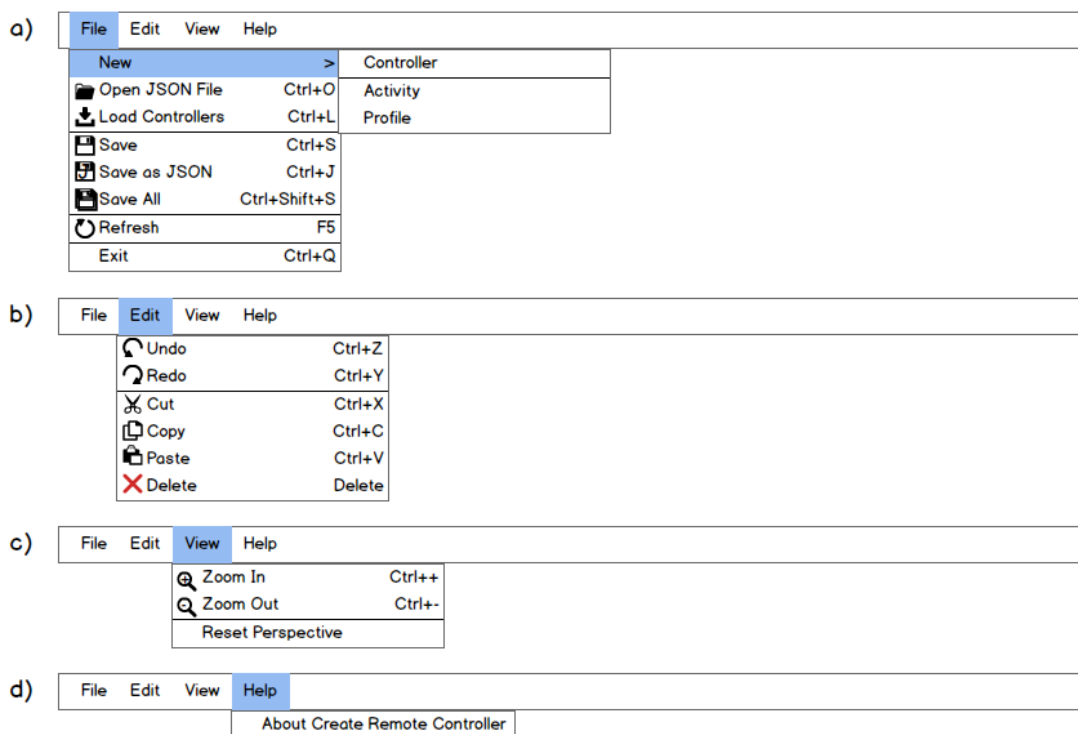


Abbildung 3.18: Menüleiste der Create Remote Controller-Benutzeroberfläche

Der Button „Load Controllers“ in a) soll dazu dienen, die auf dem Controller-Agenten gespeicherten Fernsteuerungen abzurufen und diese im Fernsteuerungsstrukturbereich anzuzeigen. Der „Save“-Button wiederum, um ausgewählte Fernsteuerung im Fernsteuerungsstrukturbereich an den Agenten zu senden. Die Buttons „Save as JSON“ und „Open JSON File“ sollen eine ähnliche Funktion haben. So sollen diese dazu dienen, die erstellten Fernsteuerungen lokal zu speichern oder zu öffnen. Die „Save“-Buttons beziehen sich auf aus-

gewählte Fernsteuerungen aus dem Fernsteuerungsstrukturbereich, so sollen die beiden Buttons ausgegraut werden, wenn keine Fernsteuerung ausgewählt ist. Der „Refresh“-Button soll die ausgewählte Fernsteuerung neu bauen sowie im Gerätebereich neu darstellen, falls im Falle eines Fehlers die Fernsteuerung nicht richtig angezeigt werden sollte.

Der „Undo“-Button in **b)** soll dazu dienen, Aktionen rückgängig zu machen, wie die Erstellung eines Elementes oder die getätigten Änderungen im Eigenschaftenbereich. Der „Redo“-Button wiederum soll „Undo“ Aktionen rückgängig machen. Die darunter befindlichen Buttons beziehen sich auf ausgewählte Elemente (Fernsteuerung, Profil, Activity, Button, Slider, Toggle-Button) aus dem Fernsteuerungsstrukturbereich und Gerätebereich, so sollen die Buttons ausgegraut werden, wenn kein Element ausgewählt ist.

Die in **c)** zu sehenden Buttons „Zoom In“ und „Zoom Out“ sollen dem zoomen im Gerätebereich dienen. Der „Reset Perspective“-Button wiederum soll dazu dienen die „Create Remote Controller“-Oberfläche auf das Standard Layout zurück zu setzen.

Der in **d)** aufgeklappt zu sehende Reiter „Help“ bietet die Funktion „About Create Remote Controller“ an. Durch einen Klick auf diesen soll sich das Infofenster der „Create Remote Controller“-Benutzeroberfläche öffnen, auf dem vorerst nur die Version der „Create Remote Controller“-Oberfläche stehen soll.

## 2. Funktionsleiste



Abbildung 3.19: Funktionsleiste der Create Remote Controller-Benutzeroberfläche

Die Funktionsleiste in Abbildung 3.19 soll grundlegende Funktionen der Oberfläche anbieten. Diese wurden für einen besseren Überblick in kleine Gruppen zusammengefasst. Die Reihenfolge der Buttons ist hierbei nach eigenem Ermessen, nach Relevanz von links nach rechts absteigend geordnet worden.

Auf den Buttons befinden sich Symbole die ihre Funktion widerspiegeln soll. Für den Fall, dass das Symbol auf einem der Buttons nicht eindeutig oder missverständlich sein sollte, soll eine Schnellinfo angeboten werden. Diese soll angezeigt werden, sobald der Benutzer mit der Maus auf den entsprechenden Button geht, wie in der Abbildung über dem „Save“-Button zu sehen ist.

### 3. Fernsteuerungsstrukturbereich

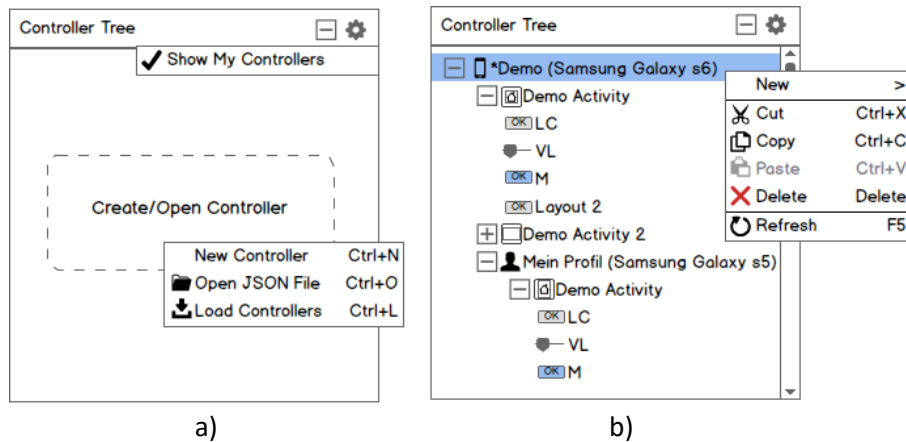


Abbildung 3.20: Fernsteuerungsstrukturbereich der Create Remote Controller-Benutzeroberfläche

Der in Abbildung 3.20 zu sehende Fernsteuerungsstrukturbereich soll dazu dienen, Fernsteuerungen in einer Baumstruktur darzustellen. So soll auf der Oberfläche, wenn keine Fernsteuerung angezeigt wird, ein „Create/Open Controller“, wie in a) zu sehen ist, dargestellt werden, um dem Benutzer so seine Möglichkeiten aufzuzeigen. Sobald eine Fernsteuerung oder ein Profil aus diesem Bereich ausgewählt wird, soll dieser im Fernsteuerungsstrukturbereich visuell dargestellt werden.

Es soll dem Benutzer möglich sein, aus dem Bibliotheksstrukturbereich Elemente in eine Activity im Fernsteuerungsstrukturbereich zu ziehen, mittels „Drag & Drop“. Diese sollen dann auf dem Gerätebereich visuell dargestellt werden. Es soll ein -Symbol angezeigt werden, das verdeutlichen soll, dass die gewollte Aktion nicht möglich ist, wenn der Benutzer versucht ein Element in eine Fernsteuerung oder ein Profil zu ziehen anstelle einer Activity.







Über das -Symbol oben rechts soll der Benutzer die Möglichkeit erhalten, nur seine erstellten Fernsteuerungen in diesem Bereich anzeigen zu lassen. Sobald diese Funktion aktiviert wird, soll dies durch einen Haken symbolisiert werden. Über das -Symbol links daneben wiederum, sollen alle aufgeklappten Fernsteuerungen, Activitys sowie Profile zugeklappt werden. Dies soll dem Benutzer ermöglichen, einen schnelleren Überblick über die Fernsteuerungen zu erhalten.

Der Benutzer soll die Möglichkeit haben, über einen Rechtsklick ein „Kontextmenü“ zu öffnen. So soll ein Kontextmenü mit anderem Inhalt angezeigt werden, wenn wie in b) auf ein Element ein Rechtsklick ausgeführt wird im Vergleich zu a).

Beim Klick auf den Button „New Controller“ oder „New“ -> „Controller“ soll sich das „New Controller Fenster“ öffnen. Beim Klick auf „Activity“ oder „Profile“ über „New“ sollen neue Elemente des ausgewählten Typs erstellt werden, in der ausgewählten Fernsteuerung mit dem Namenszusatz „new“, z.B. „new Activity“.

Ein Profil soll eine Kopie einer Fernsteuerung darstellen, da Fernsteuerungen von jedem Benutzer verwendbar sein sollen, egal wer der Ersteller ist. Ein Profil wiederum soll nur vom Ersteller genutzt werden können. Dieser soll weiterhin angepasst und auf eigene Bedürfnisse abgestimmt werden können, aber nur vom Ersteller. So wurde beispielsweise in **b)** ein Profil erstellt und diesem ein anderes Mobilgerät zugewiesen.

Des Weiteren soll neben den Bezeichnungen spezielle Symbole dargestellt werden, damit der Benutzer sich einen schnelleren Überblick über die Fernsteuerung verschaffen kann. So soll es folgende Informationen geben:

- \*: Es liegen Änderungen an der Fernsteuerung vor.
- : Ist die Main-Activity, diese soll die erste Activity darstellen, die angezeigt werden soll, wenn die Fernsteuerung auf dem Mobilgerät dargestellt wird.
- : Eine gewöhnliche Activity, kann horizontal als auch vertikal genutzt werden auf dem Mobilgerät.
-  (Landscape) /  (Portrait): Landscape für die Activity ist nur horizontal nutzbar bzw. Portrait nur vertikal. Dies soll es auch in der Main-Activity-Variante als  /  geben.

#### 4. Bibliotheksstrukturbereich

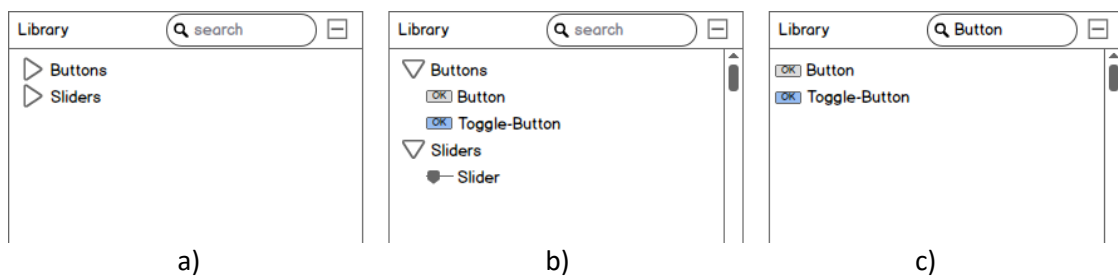


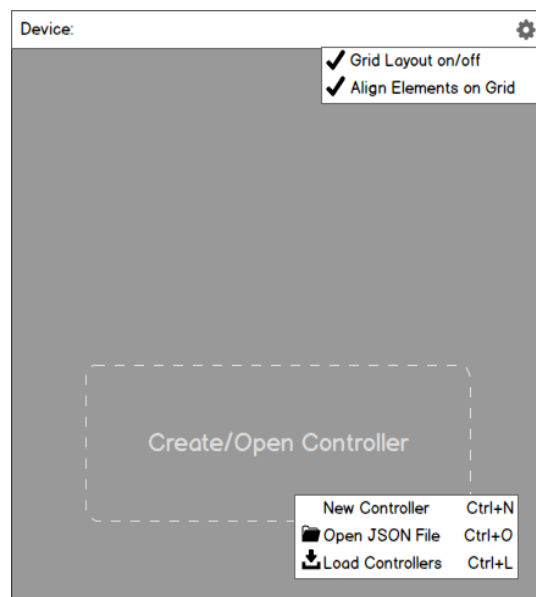
Abbildung 3.21: Bibliotheksstrukturbereich der Create Remote Controller-Benutzeroberfläche

Der Bibliotheksstrukturbereich in Abbildung 3.21 soll dazu dienen die Elemente die sich auf einer Activity befinden, können strukturiert in Gruppen darzustellen und diese per „Drag & Drop“ auf den Fernsteuerungsstrukturbereich oder den Gerätebereich zu ziehen. Die Elemente sollen für einen schnellen wiedererkennungswert Symbole besitzen, die wiederum für ein Element eindeutig sein sollen.

Der Benutzer soll die Möglichkeit haben über das Suchfeld nach bestimmten Elementen zu suchen. Dabei sollen alle Elemente angezeigt werden, die dem eingetragenen Suchbegriff ähneln (c). Die so gefundenen Elemente sollen dann untereinander stehen, statt in ihren Gruppen, damit ersichtlich ist das die Suchfunktion verwendet wird. Damit die Elemente wieder in ihren Gruppen dargestellt werden, soll es notwendig sein den eingetragenen Begriff im Suchfeld zu entfernen. Über das ☒-Symbol rechts neben dem Suchfeld sollen wie im Fernsteuerungsstrukturbereich alle aufgeklappten Gruppen zugeklappt werden.

## 5. Gerätebereich

Der Gerätebereich in Abbildung 3.22 soll dem gestalten der Fernsteuerungen dienen. Auf diesem sollen hierfür, die aus dem Bibliotheksstrukturbereich ausgewählten Elemente, per „Drag & Drop“ platziert und positioniert werden können. Die hinzugefügten Elemente sollen wie beim klassischen Windows Fenster über die Ecken und Seiten größer gezogen werden können, hierbei soll auch dementsprechend der Zeiger als entsprechendes Symbol dargestellt werden. Sollte innerhalb des Bereiches hineingezoomt werden, soll ein Scroll Balken unten und rechts angezeigt werden. Auch hier soll wie im Fernsteuerungsstrukturbereich der Text „Create/Open Controller“ dargestellt werden, wenn keine Fernsteuerung angezeigt wird. Sollte sich wiederum eine leere Fernsteuerung, also eine Fernsteuerung ohne Elemente, auf dem Gerätebereich befinden, soll auf diesem „Drag Library item here“ stehen, aus demselben Grund wie beim „Create/Open Controller“ Text.



a)



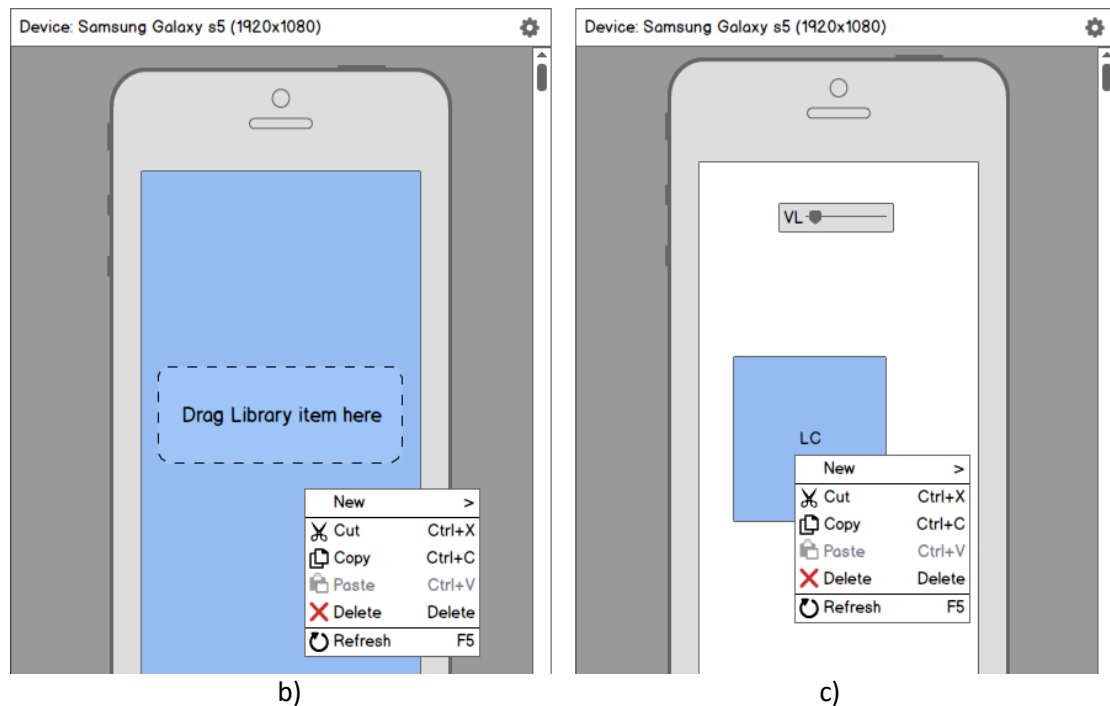


Abbildung 3.22: Gerätebereich der Create Remote Controller-Benutzeroberfläche

Das über das New Controller Fenster ausgewählte Mobilgerät soll hier dargestellt werden. Falls kein Mobilgerät ausgewählt wurde sondern nur die Maße angegeben wurden, soll stattdessen ein Feld in der angegebenen Größe angezeigt werden. In beiden Fällen sollen die Elemente aus dem Bibliotheksstrukturbereich nur innerhalb des angegebenen Bereiches positioniert werden können. In **b)** ist das Feld blau hervorgehoben dargestellt, dieses ist blau dargestellt, da gewollt ist, wenn auf ein Element im Gerätebereich per Links- oder Rechtsklick geklickt wird, dass diese hervorgehoben werden sollen. Das hervorgehobene Feld stellt hierbei zugleich die Activity dar.

Die Leiste oberhalb des dargestellten Mobilgerätes soll dem Anzeigen dessen Namens dienen, dass im Gerätebereich dargestellt wird. Damit der Benutzer falls er das dargestellte Mobilgerät nicht erkennen sollte, es am Namen erkennen kann. So soll wenn kein Mobilgerät angegeben wird neben der Beschriftung „Device“ nichts angezeigt werden.

Über das ⚙-Symbol sollen dem Benutzer zwei Funktionen angeboten werden (**a**). Die erste Funktion soll ein Raster auf dem Feld anzeigen, die zweite wiederum soll dazu dienen Elemente an diesem Raster auszurichten.

In dem Kontextmenü das in **b)** und **c)** zu sehen ist, ist der Button „Paste“ ausgegraut. Dieser soll ausgegraut dargestellt werden, wenn kein Element ausgeschnitten oder kopiert wurde.

## 6. Eigenschaftenbereich

Der Eigenschaftenbereich soll dazu dienen, Eigenschaften der Elemente einer Fernsteuerung anzuzeigen, damit diese bearbeitet werden können. So sollen diese angezeigt werden, sobald ein Element im Fernsteuerungsstrukturbereich oder im Gerätebereich ausgewählt wurde. In diesem Bereich sollen nicht nur Eigenschaften von Elementen dargestellt werden, sondern auch von Fernsteuerungen Profilen und Activitys. In Abbildung 3.23 ist der Eigenschaftenbereich abgebildet.

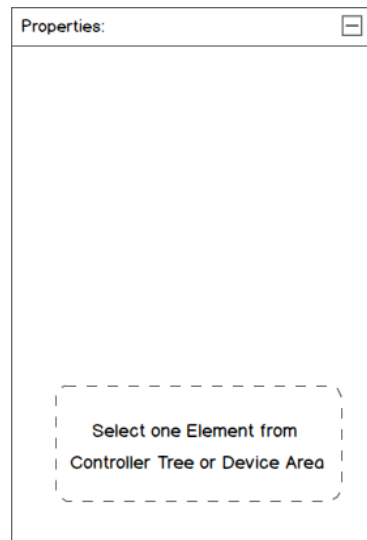



Abbildung 3.23: Eigenschaftenbereich der Create Remote Controller-Benutzeroberfläche.

Wenn kein Element aus den genannten Bereichen ausgewählt wurde, soll ein Infotext dargestellt werden. Falls ein Element wiederum ausgewählt wurde, soll der angezeigte Infotext entfernt und die Eigenschaften des ausgewählten Elementes dargestellt werden. Zudem soll oben neben der Beschriftung „Properties“ die Bezeichnung des ausgewählten Elementes angezeigt werden.

Über das -Symbol sollen die Eigenschaften eines Elementes zugeklappt werden. Die so zuklappbaren Bereiche sind hierarchisch strukturiert und sollen dementsprechend auch zugeklappt werden. Dies wird noch mal genauer für die einzelnen Elemente erläutert.

Die Abbildung 3.24 stellt die Eigenschaften einer Fernsteuerung (a) und eines Profils (b) dar. Die Eigenschaften einer Fernsteuerung und eines Profils sind weitestgehend identisch, daher sind die angezeigten Eigenschaften auch dieselben.

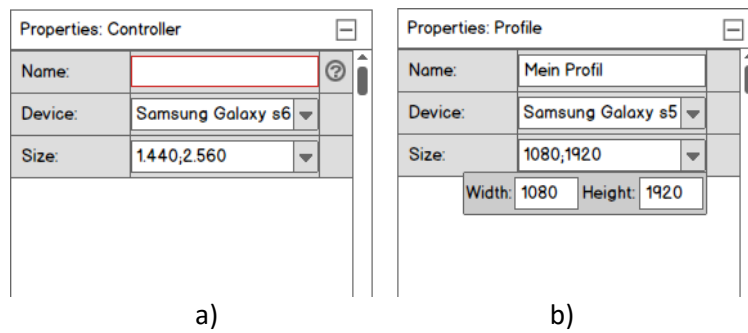


Abbildung 3.24: Eigenschaften einer Fernsteuerung und eines Profils im Eigenschaftenbereich.

So kann über ein Namensfeld eine Bezeichnung eingetragen werden, dieses soll darauf geprüft werden, ob der eingetragene Name verwendet werden kann. Genauer gesagt soll geprüft werden, ob der Name der Fernsteuerung vergeben ist oder ob es in der Fernsteuerung ein Profil mit der eingetragenen Profilbezeichnung gibt. Sollte dies der Fall sein, soll das Namensfeld mit einem roten Rahmen dargestellt werden sowie ein ⓘ-Symbol daneben angezeigt werden. Dieser soll besagen, dass die eingetragene Bezeichnung vergeben ist und beim Speichern die entsprechende Fernsteuerung bzw. das entsprechende Profil überschrieben wird. Im Falle, dass dort keine Bezeichnung eingetragen sein sollte, soll der Info-text besagen, dass nicht gespeichert werden kann, solange keine Bezeichnung eingetragen ist. Diese Benachrichtigung soll somit auch in einem separaten Dialogfenster angezeigt werden, wenn ein Benutzer versucht diese Fernsteuerung zu speichern.

Unterhalb des Textfeldes befindet sich eine Dropdown-Liste, über den es möglich sein soll ein Mobilgerät zu wählen/ändern, das dann im Gerätebereich visuell dargestellt werden soll. Da drunter wiederum befindet sich ein Dropdown-Feld, das die Breite und Höhe des ausgewählten Mobilgerätes darstellen soll. Es soll nicht möglich sein, innerhalb des Dropdown-Felds zu schreiben. So soll, wenn auf das Dropdown-Feld gedrückt wird, sich wie in der Abbildung 3.24b zu sehen, ein extra Feld öffnen, über den der Benutzer dann die Größe manuell eintragen kann. Da es ein solches Dropdown-Feld, wie es dargestellt wird, nicht gibt, muss dieses implementiert werden. Ein solches Dropdown-Feld zu verwenden, hat den Vorteil einer kompakteren Darstellung, so müssen nicht zwei separate Felder angezeigt werden, um die Höhe und Breite anzugeben.

In Abbildung 3.25 werden die Eigenschaften einer Activity dargestellt. Das Namensfeld soll hierbei die gleichen Eigenschaften aufweisen wie bei einer Fernsteuerung oder einem Profil Namensfeld. So soll hier geprüft werden, ob es in der Fernsteuerung oder im Profil (kommt darauf an, wo sich die Activity befindet) eine Activity mit der eingetragenen Bezeichnung bereits befindet.

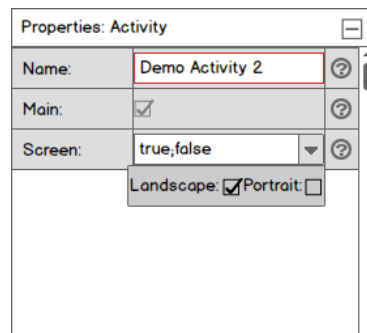


Abbildung 3.25: Eigenschaften einer Activity im Eigenschaftenbereich.

Die Checkbox mit der Beschriftung „Main“ soll dazu dienen, die erste Activity zu wählen, die beim Start der Fernsteuerung auf dem Mobilgerät dargestellt werden soll. Sobald eine Activity zur Main-Activity wird, soll die Checkbox ausgegraut werden, damit der Haken nicht rausgenommen werden kann. Sobald eine andere Activity zur Main-Activity wird soll der Haken bei der alten Main-Activity entfernt werden und die Checkbox nicht mehr ausgegraut sein. Dies soll so wie beschrieben umgesetzt werden, damit es immer zurzeit in einer Fernsteuerung bzw. einem Profil eine Main-Activity gibt und diese als erste Activity, nach dem Öffnen der Fernsteuerung, auf dem Mobilgerät dargestellt werden kann.

Die darunter befindliche Dropdown-Checkbox (muss, wie in der Abbildung 3.25 dargestellt, implementiert werden) soll dazu dienen, einen Bildschirmmodus zu wählen. So befindet sich innerhalb dieser Dropdown-Checkbox zwei Checkboxes, die es ermöglichen sollen, zwischen horizontalem (Landscape) oder vertikalem (Portrait) Betrieb festzulegen. So soll die Activity auf dem Mobilgerät horizontal oder vertikal nutzbar sein, so wie nur ein oder kein Hacken gesetzt werden können. Kein Haken bedeutet, dass die Activity horizontal als auch vertikal genutzt werden können soll. Diese Informationen sollen dem Benutzer über das ?-Symbol mitgeteilt werden, sobald er auf diesen klickt. Falls ein Haken gesetzt wird, soll geprüft werden, ob bei der anderen Checkbox auch der Haken gesetzt wurde und dieser dann entfernt werden.

Die Eigenschaften der Elemente Button, Slider und Toggle-Button sind grundsätzlich die gleichen. Daher werden die Felder, die dargestellt werden, wenn auf ein Element geklickt wird, anhand des Elementes „Button“ exemplarisch erläutert, was in Abbildung 3.26 dargestellt ist.

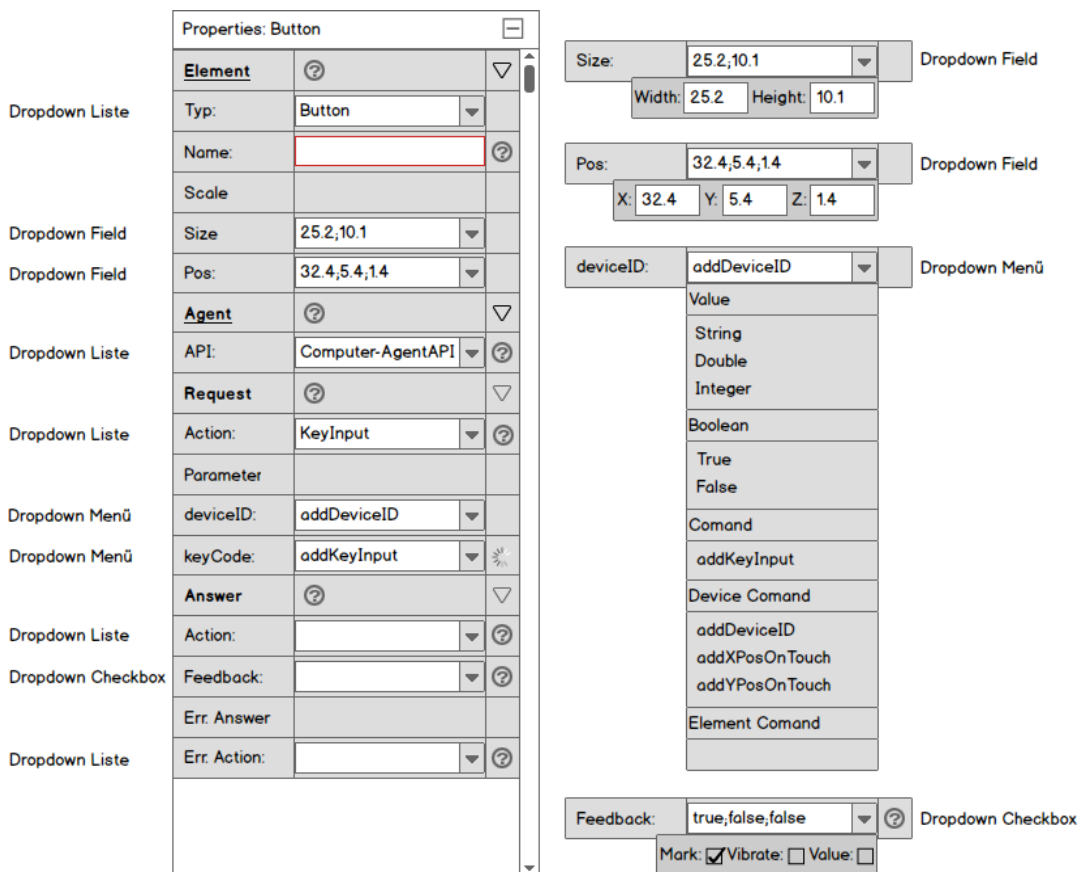




Abbildung 3.26: Aufbau der Eigenschaften eines Elementes anhand des Beispiel Elementes-„Button“.

Über das -Symbol oben rechts sollen alle aufgeklappten Bereiche zugeklappt werden. So sollen alle Bereiche bis zum selben Überschriftentyp zugeklappt werden, so dass wenn alles zugeklappt ist, nur noch die Bereiche „Element“ und „Agent“ zugeklappt zu sehen sind. Das neben den Überschriften befindliche -Symbol soll Informationen zu den in den Bereichen enthaltenen Feldern liefern.

Damit der Benutzer schnell und einfach den Typ eines Elementes ändern kann, ohne dass die getätigten Einstellungen verloren gehen, befindet sich hierfür im Element-Bereich eine Dropdown-Liste, über die jederzeit ein anderes Elementes ausgewählt werden kann. Bei dem Namensfeld soll darauf geprüft werden, ob es in der Activity, in der sich das Element befindet, ein weiteres Element mit derselben Bezeichnung bereits existiert. Über die beiden Dropdown-Felder unterhalb des Namensfeldes soll der Benutzer die Möglichkeit haben, wie in der Abbildung dargestellt, die Größe und Position des Elementes ändern zu können.

Der „Agent“-Bereich soll dem Zuweisen einer Aktion dienen. So gibt es hier einen „Request“- und „Answer“-Bereich. Im „Request“-Bereich soll die Aktion erstellt werden, die an

den Agenten gesendet werden soll. Im „Answer“-Bereich wiederum soll die Antwortnachricht gewählt werden, die man vom Agenten erhalten würde, wenn man die im „Request“-Bereich angegebene Aktion ausführt.

Um eine Aktion zu erstellen, muss in der Dropdown-Liste mit der Bezeichnung „API“, eine API eines Agenten gewählt werden. Hier sollen die APIs aufgelistet werden, die über die Serveroberfläche hinzugefügt wurden. In der darunter befindlichen Dropdown-Liste mit der Beschriftung „Action“ soll der Benutzer dann die Möglichkeit haben, aus der ausgewählten API eine Funktion auszuwählen. Nachdem eine Funktion ausgewählt wurde, sollen dessen Parameter wie in der Abbildung zu sehen ist, untereinander aufgelistet werden, mit einem Dropdown-Menü daneben. Über dieses soll ein Wert oder eine spezielle Funktionen dem Parameter zugewiesen werden können. Bei den Funktionen, die zugewiesen werden, handelt es sich um spezielle Strings. Diese sollen auf dem Mobilgerät ausgelesen werden, damit eine Funktion ausgeführt und dessen Ergebnis, anstelle des Strings, eingetragen werden kann. So gibt es folgende Funktionen, die zugewiesen werden können:

- addKeyInput: Wird nicht auf dem Mobilgerät umgesetzt. Wartet auf eine Tastatur oder Maus Eingabe vom Benutzer der dem Parameter zugewiesen werden soll nachdem er auf „addKeyInput“ gedrückt hat. das Warten wird durch ein \*-Symbol dargestellt.
- addDeviceID: Fügt auf dem Mobilgerät die ID des Mobilgerätes ein.
- addXPosOnTouch/addYPosOnTouch: Fügt auf dem Mobilgerät die x- oder y-Position ein wenn auf den Bildschirm gedrückt wird.

Im „Answer“-Bereich wiederum kann über die Dropdown-Checkbox, mit der Bezeichnung „Feedback“, die Art der Rückmeldung beim Erhalt der Antwortnachricht gewählt werden. So hat der Benutzer die Möglichkeit z wählen, zwischen:

- Mark: Das Element soll markiert werden, beim Erhalt der Antwortnachricht.
- Vibrate: Das Mobilgerät soll vibrieren, beim Erhalt der Antwortnachricht.
- Value: Die Bezeichnung des Elementes soll sich in die Nachricht, die in der Antwortnachricht enthalten ist, ändern, beim Erhalt der Antwortnachricht.

Da schon, wie beschrieben, der Aufbau der Elemente größtenteils identisch ist, werden nun die Unterschiede zwischen den einzelnen Elementen erläutert.

Properties: Slider	
Element	
Typ:	Slider
Name:	VL
Value:	0,0;100,0
Scale	
Size	25,2;10,1
Pos:	32,4;5,4;1,4
Agent	
API:	Computer-AgentAPI
Request	
Action:	SetVolume
Parameter	
deviceID:	addDeviceID
volume:	addSliderValue
Answer	
Action:	
Feedback:	true,true,true
Err. Answer	
Err. Action:	

a)

Properties: Toggle-Button	
Element	
Typ:	Toggle-Button
Name:	M
Scale	
Size	25,2;10,1
Pos:	32,4;5,4;1,4
Key Down	
Agent	
API:	Computer-AgentAPI
Request	
Action:	Mute
Parameter	
deviceID:	addDeviceID
setMute:	True
Answer	
Action:	
Feedback:	
Err. Answer	
Err. Action:	
Key Up	

b)

Properties: Button	
Element	
Typ:	Button
Name:	Layout 2
Scale	
Size	25,2;10,1
Pos:	32,4;5,4;1,4
Agent	
API:	Activitys
Request	
Action:	Demo Layout 2
Answer	
Action:	
Feedback:	
Err. Answer	
Err. Action:	

c)

Abbildung 3.27: Eigenschaften der Elemente „Slider“ und „Toggle-Button“ sowie das Wechseln zwischen Activitys.

In der Abbildung 3.27a ist das Element Slider abgebildet. Dieses besitzt, zusätzlich zu den vorher beschriebenen Eigenschaften, die Eigenschaft „Value“. Dieses ist ein Dropdown-Feld mit zwei Werten. So kann über das „Value“-Dropdown-Feld der minimale und maximale Wert des Slider festgelegt werden. Zudem soll über das Dropdown-Menü im „Element Comand“-Bereich die Funktion „addSliderValue“ zur Verfügung stehen. Diese Funktion soll auf dem Mobilgerät den Wert des Sliders in die Nachricht, die an den entsprechenden Agenten gesendet wird, eintragen.

In b) wiederum ist das Element Toggle-Button dargestellt. Dieser besitzt zwei weitere Bereiche, die beide jeweils den Bereich „Agent“ mit dessen Unterbereichen beinhaltet. So gibt es den Bereich „Key Down“, hier kann eine Aktion festgelegt werden, die getätigt werden soll, wenn der Button rein gedrückt wurde sowie den Bereich „Key Up“, wo wiederum festgelegt werden kann, welche Aktion ausgeführt werden soll, wenn der Button raus gedrückt wurde.


In c) wird das Element Button nochmals dargestellt. Anhand diesem soll gezeigt werden, wie zwischen erstellten Activitys gewechselt werden soll. So soll hierfür in der „API“-Dropdown-Liste „Activitys“ gewählt werden. „Activitys“ ist keine API, sondern soll standardmäßig angeboten werden. Nun soll über die „Action“-Dropdown-Liste eine Activity gewählt werden, auf die gewechselt werden soll, sobald auf dem Mobilgerät auf den Button gedrückt wurde. Zudem ist hier zu sehen, dass der „Parameter“-Bereich nicht dargestellt wird, da die ausgewählte Action keine Parameter besitzt.

## 7. Informationsbereich



Abbildung 3.28: Informationsbereich der Create Remote Controller-Benutzeroberfläche

Im Informationsbereich der in Abbildung 3.28 zu sehen ist, sollen dem Benutzer Erfolgsmeldungen angezeigt werden. Dies soll dazu dienen, dem Benutzer ein Feedback zu geben und ihn so in seinen Aktionen zu bestärken. Die angezeigten Erfolgsmeldungen sollen kurz und knapp gehalten werden, damit diese schnell gelesen werden können.

Des Weiteren soll hier ein -Symbol angezeigt werden, wenn eine Nachricht an einen Agenten gesendet und auf dessen Nachricht (Antwortnachricht) gewartet wird, z.B. beim Speichern einer Fernsteuerung auf dem Controller-Agenten auf dessen Bestätigung, dass die Fernsteuerung auf dem Agenten gespeichert wurde.

## 3.2 Mobile Anwendung

Der Client stellt die Android-Applikation da und besteht aus sechs Activitys. Es wurde überlegt das „Material Design“ von Google zu verwenden für die Gestaltung der Oberfläche.

Das „Material Design“ sieht auf den ersten Blick aus wie herkömmliches Flat-Design und setzt auch wie bei Flat-Design stark auf Icons und Schriftarten als tragende Design Elemente. Der Unterschied zum Flat-Design ist, dass das „Material Design“ auf großflächige, einfarbige Elemente mit kontrastreichen und sehr bunten Farben setzt. Im Gegensatz zum Flat-Design verfügen UI-Elemente nicht nur eine X- und Y-Koordinate, sondern auch eine Z-Koordinate. So können Elemente unabhängig voneinander überlagert, animiert und deren Schatten berechnet werden. (Stückler (2014))

Das „Material Design“ beinhaltet zudem Pixelangaben für Abstände und Größen zwischen und von Elementen sowie Angaben, welche Elemente für welche Aktionen am meisten geeignet sind und wie diese zu animieren sind.



### 3.2.1 Login Activity

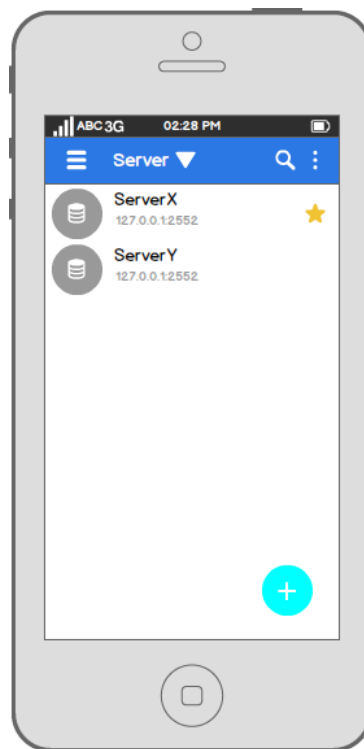


Abbildung 3.29: LoginActivity

Die in Abbildung 3.29 zu sehende „Login Activity“ soll dem Verwalten der Server dienen. Hierfür sollen Server auf dem Mobilgerät gespeichert und in einer Liste dargestellt werden. Des Weiteren soll die Oberfläche dazu dienen, sich mit einem Server zu verbinden und auf diesem sich dann zu registrieren oder anzumelden.

Für die Auflistung der Server soll eine Liste verwendet werden, da laut dem „Material Design“ eine Liste sich eignet, um homogene Datentypen zu repräsentieren. Bei der Liste soll es sich um eine zweizeilige Liste handeln, die zu der Bezeichnung auch die IP-Adresse anzeigen soll, damit anhand dieser der Benutzer den Server identifizieren kann. Die in der Liste enthaltenen Server sollen Alphabetisch nach ihrer Bezeichnung sortiert und angezeigt werden, dies soll laut dem „Material Design“ dem Benutzer das Suchen und Finden erleichtern. Bei den in der Liste favorisierten Servern sollen ★-Icons rechts neben deren Bezeichnung dargestellt werden, damit diese als Favorisierte identifiziert werden können. Links neben der Bezeichnung wiederum soll ein 🖨️-Icon angezeigt werden, damit der Benutzer schnell erkennen kann, dass es sich bei den Elementen in der Liste um Server handelt.

Weiter unten befindet sich die sogenannte „schwimmende Aktionsschaltfläche“, dieser Button soll dem Hinzufügen von neuen Servern in die Liste dienen und soll hierfür, wenn

auf diesen gedrückt wird, die „New Server Activity“ öffnen. Da diese Funktion dem Benutzer leicht zugänglich sein soll, wurde sich hier für eine schwimmende Aktionsschaltfläche entschieden. Damit jedoch dieser Button im Fokus des Benutzers liegt, soll wie im „Material Design“ beschrieben, durch eine Färbung des Buttons ein Akzent auf diesen gelegt werden. Des Weiteren soll innerhalb des Buttons ein Plus-Icon dargestellt werden, das dem Benutzer verdeutlichen soll, dass mit Hilfe dieses Buttons etwas in die Liste hinzugefügt werden kann.

Oberhalb der Liste befindet sich die App-Leiste, diese beinhaltet ein ☰-Icon, welches die Navigationsleiste öffnen soll, den Titel der Activity, ein ▼-Icon um nach bestimmten Servern zu filtern sowie ein 🔍-Icon um nach Servern in der Liste zu suchen und ein ☰-Icon um das Menü zu öffnen. Diese Funktionalitäten werden folgend genauer erläutert.

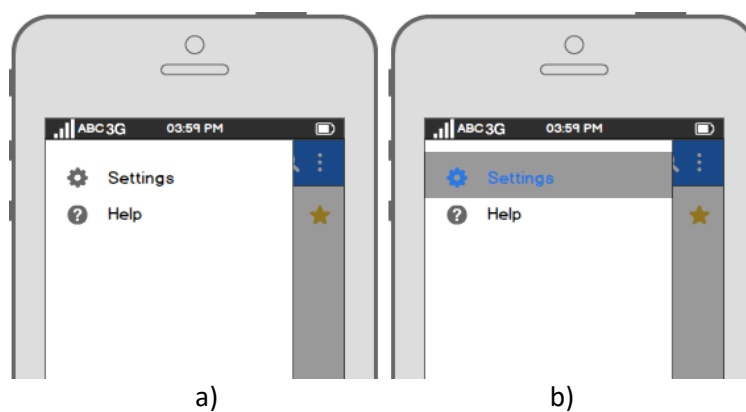


Abbildung 3.30: Navigationsleiste der Login Activity

Die in Abbildung 3.30a zu sehende Navigationsleiste beinhaltet zwei Buttons, eines für die Einstellungen, über die es dem Benutzer möglich sein soll, die Oberfläche der Anwendung zu individualisieren, indem eine andere Farbe ausgewählt werden kann. Da das „Material Design“ Grund- und Akzentfarben umfasst und somit für die farbliche Abstimmung eine große Rolle spielt, um eine harmonische Oberfläche zu schaffen, sollen die Farbmuster des „Material Design“ zur Auswahl stehen. Der zweite Button ist der „Help“-Button, der Informationen zu der App anzeigen soll und wie diese zu bedienen ist. Sobald auf einen Button gedrückt wird, soll dieser hervorgehoben werden (b). Alternativ soll es möglich sein, mit einer Wischgeste vom linken Rand des Displays nach rechts die Navigationsleiste zu öffnen.

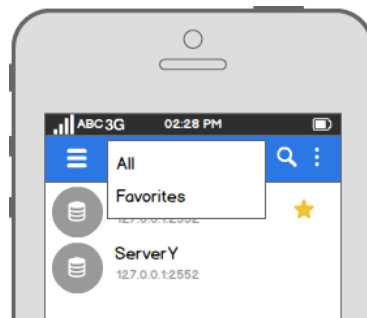




Abbildung 3.31: Filter Funktion der Login Activity

Das neben dem Titel befindliche ▼-Icon soll zum Filtern der Server dienen. Beim Drücken auf diesen soll sich wie in [Abbildung 3.31](#) ein Menü öffnen, über den der Benutzer dann die Möglichkeit haben soll, zwischen zwei Filtern zu wählen. „All“ um alle Server und „Favorites“ um nur die favorisierten Server in der Liste anzuzeigen. Sobald der Benutzer den Filter „Favorites“ auswählt, soll statt dem Titel „Server“ die Bezeichnung „Favorites“ dort stehen. Mit der Auswahl des Filters „All“, was wiederum die Standardansicht darstellt, soll wieder der Titel „Server“ dargestellt werden.



Abbildung 3.32: Such Funktion der Login Activity

Sobald auf das 🔍-Icon gedrückt wird, soll der Titelbereich zu einem Textfeld werden, auf dem grülich „Search“ stehen soll, wie in [Abbildung 3.32a](#) zu sehen ist. Sobald der Benutzer anfängt etwas in das Textfeld einzutragen, soll das Wort „Search“ nicht mehr dargestellt

werden (b). Über das erneute Drücke des -Icons oder des „OK“-Buttons auf der Bildschirm Tastatur, soll die Suche starten. Falls der Benutzer wiederum die Suchfunktion beenden möchte, soll sich hierfür neben dem Textfeld ein -Icon befinden, mit dessen Hilfe der Benutzer zurück zur Standardansicht gelangen kann.

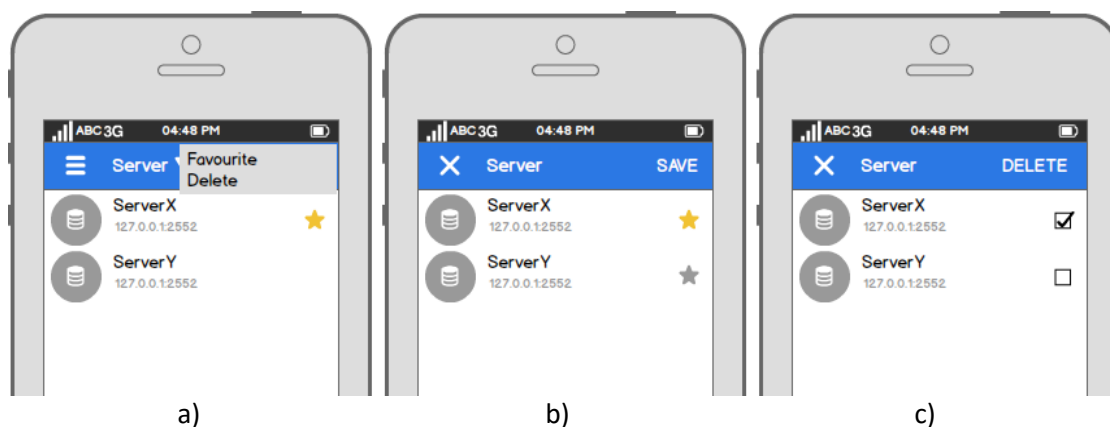





Abbildung 3.33: Menü Funktion der Login Activity

Das in Abbildung 3.33a aufgeklappt zu sehende Menü bietet zwei Funktionen an. Über den „Favourite“-Button wird die Favoriten-Ansicht (b) der Liste geöffnet, über die der Benutzer einzelne Server favorisieren können soll, indem er auf den Server drückt, den er favorisieren möchte und mit „SAVE“ die Aktion bestätigt. Nachdem der Benutzer auf einen Server gedrückt hat, soll aus dem -Icon ein -Icon werden oder andersherum um ein visuelles Feedback zu geben, dass die Aktion erfolgreich war.

Der „Delete“-Button zeigt die Löschen-Ansicht (c) an, über den der Benutzer Server löschen können soll. Alternativ soll es auch möglich sein, diese Ansicht zu öffnen, indem auf einen Server für eine kurze Zeit gedrückt wird. Damit ein Server gelöscht werden kann, muss bei dessen Checkbox der Haken gesetzt werden und auf den „Delete“-Button oben rechts gedrückt werden. Durch die Checkboxes soll es dem Benutzer ermöglicht werden, schnell und einfach mehrere Server gleichzeitig zu löschen.

Sowohl in der Favoriten- als auch in der Löschen-Ansicht soll die schwimmende Aktions-schaltfläche nicht zur Verfügung stehen. Um wiederum beide Ansichten zu beenden und zurück zur Standardansicht zu gelangen, auf der die schwimmende Aktionsschaltfläche wieder angeboten wird, befinden sich auf beiden Ansichten oben links ein -Icon, der beim Drücken auf diesen den Benutzer wieder auf die Standardansicht weiterleiten soll.

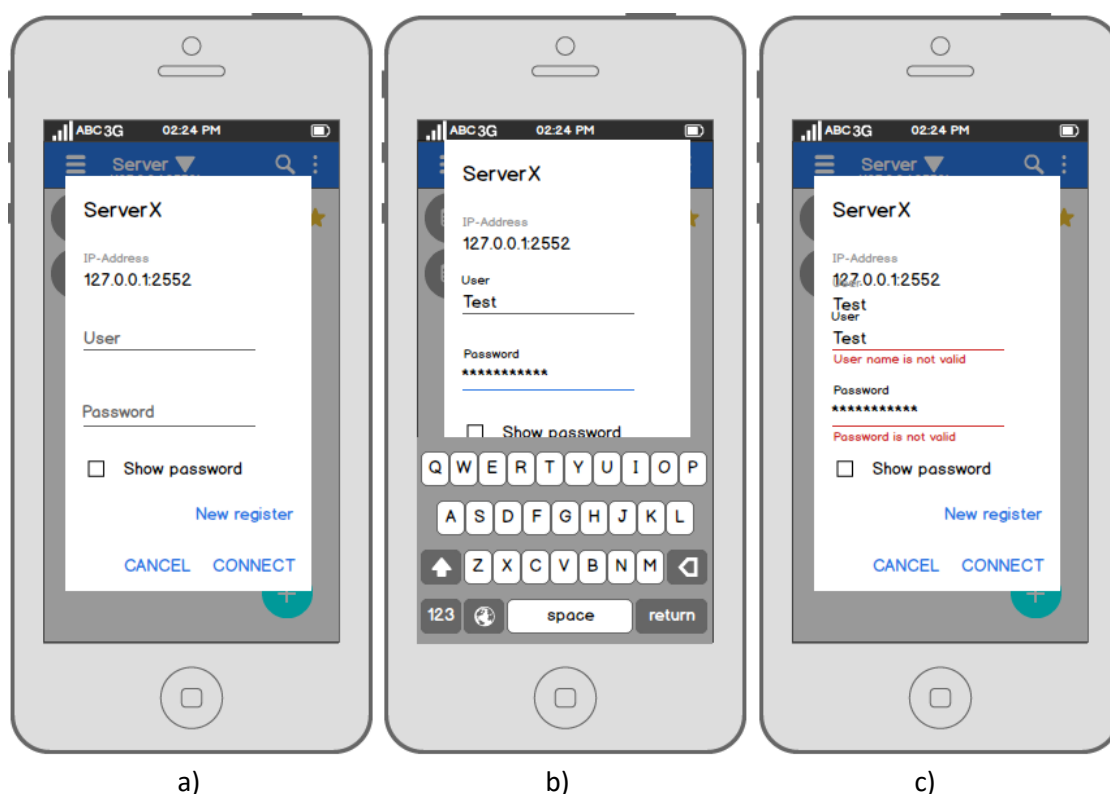


Abbildung 3.34: Anmelde Dialogfenster der Login Activity

Sobald ein Server aus dem im „Login Activity“-Bereich befindlichen Liste gewählt wird, soll sich das in Abbildung 3.34a zu sehende Anmeldedialogfenster öffnen. Der Benutzer hat die Möglichkeit, sich über das Dialogfenster auf dem ausgewählten Server anzumelden oder zu registrieren. Hierbei wurde sich für ein Dialogfenster und nicht für eine separate Activity entschieden, da dies dem Anmeldeverfahren in ein WLAN-Netz auf dem Mobilgerät ähneln soll.

Zur Bestätigung, dass es sich um den ausgewählten Server handelt, steht oben auf dem Dialogfenster dessen Bezeichnung. Da drunter wiederum steht die IP-Adresse des Servers, mit dem man sich verbinden wird. Nach der IP-Adresse folgt das „User“- und das „Password“-Textfeld, hierüber soll der Benutzer seinen Benutzernamen und sein Passwort eingeben können. So soll, wie in b) zu sehen ist, eine Tastatur eingeblendet werden, sobald auf ein Textfeld gedrückt wird. Da die Tastatur Platz wegnimmt und somit nicht der gesamte Inhalt des Dialogfenster dargestellt werden kann, soll es möglich sein innerhalb des Dialogfensters zu scrollen.

Wenn in ein Textfeld etwas eingetragen wird, soll dessen Grundlinie farblich hervorgehoben werden, zudem soll die Bezeichnung des Textfeldes sich nach oben verschieben, damit

jederzeit erkenntlich ist, um was für ein Textfeld es sich handelt und welches gerade ausgewählt ist, dies wird auch so im „Material Design“ erwähnt.

Die „Show password“-Checkbox soll dazu dienen, dass eingetragene Passwort in Klartext darzustellen, so kann der Benutzer dieses auf Korrektheit prüfen. Unterhalb dieser Checkbox befinden sich drei Buttons, mit folgenden Funktionen. Der „New register“-Button soll das Registrierungsdialogfenster öffnen. Der Button „CANCEL“ wiederum soll den Vorgang abbrechen. Der „CONNECT“-Button hingegen soll eine Login-Benachrichtigung an den Server mit den angegebenen Daten senden. Sobald die Anmeldung erfolgreich war, soll sich das in Abbildung 3.35a zu sehende Key-Dialogfenster öffnen. Falls die Anmeldung nicht erfolgreich war, soll, wie in c) zu sehen, eine dementsprechende Fehlermeldung angezeigt werden.

Das „Material Design“ empfiehlt Buttons, die sich auf Dialogfenstern befinden, farblich hervorzuheben, damit diese als solche wahrgenommen werden sowie Buttons die relevant für den Dialog sind, groß geschrieben darzustellen. Des Weiteren wird auch eine bestimmte Anordnung der Buttons empfohlen. So sollen abweisende Aktionen auf der linken und positive auf der rechten Seite dargestellt werden. Deshalb wurden diese Punkte bei der Gestaltung mitberücksichtigt.

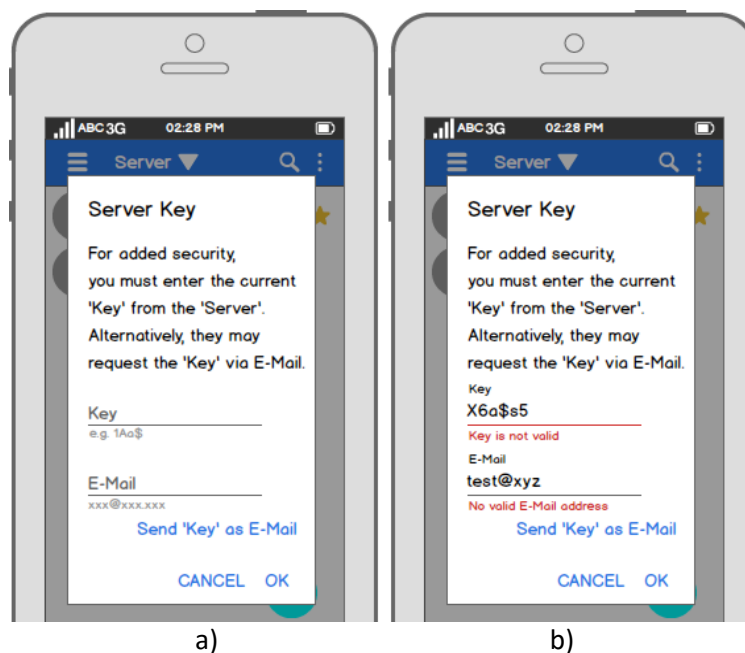


Abbildung 3.35 Server Key Dialogfenster der Login Activity

Das Dialogfenster „Server Key“ in Abbildung 3.35a soll dazu dienen, den Key, der auf der Serveroberfläche dargestellt wird, einzutragen, um so Zugriff auf den Server zu erhalten und so Fernsteuerungen abrufen zu können. Darum befindet sich hierfür im Fenster ein

Textfeld mit der Beschriftung „Key“, um diesen einzutragen. Über den Button „OK“ soll eine „CheckKey“-Benachrichtigung an den Server gesendet werden. Sobald die Überprüfung erfolgreich verlief, soll der Benutzer auf die „Controllers Activity“ weitergeleitet werden. Über den Button „CANCEL“ wiederum soll der angemeldete Benutzer abgemeldet und der Vorgang abgebrochen werden.

Sollte der Benutzer sich nicht in der Nähe des Servers aufhalten, um den Key abzulesen, soll alternativ ein Key, der nur für diesen Benutzer generiert wird, per E-Mail angefordert werden können, indem die E-Mail-Adresse im gleichnamigen Textfeld eingetragen und auf den Button „Send ‘Key‘ as E-Mail“ gedrückt wird. Dieser speziell für den Benutzer generierte Key soll bis zur Eingabe 30 Minuten gültig sein. Sollten jedoch eine oder beide der getätigten Eingaben fehlerhaft oder nicht gültig sein, soll dies, wie in **b)** zusehen dem Benutzer kenntlich gemacht werden.

Das Registrierungsdialogfenster hat im Grunde denselben Aufbau wie das Anmeldedialogfenster, mit dem Unterschied, dass unter dem „Password“-Textfeld eine Beispieleingabe abgebildet ist, damit dem Benutzer verdeutlicht wird, dass er bei der Auswahl seines Passwortes Ziffern, Klein- und Großbuchstaben sowie Sonderzeichen nutzen kann. Zudem gibt es ein zweites Passwort-Feld, auf dem der Benutzer sein Passwort erneut eintragen muss, um sicherzugehen, dass das zuerst angegebene Passwort korrekt ist. In **Abbildung 3.36** wird das Registrierungsdialogfenster dargestellt.



Abbildung 3.36: Registrierungs Dialogfenster der Login Activity

Der „REGISTER“-Button soll hier keine Login-Benachrichtigung sondern eine Register-Benachrichtigung senden. Dementsprechend treten bei einem Fehler andere Fehlermeldungen auf (b). Sollte die Registrierung erfolgreich sein, soll in einem separaten Dialogfenster dies dem Benutzer mitgeteilt werden.

### 3.2.2 New Server Activity

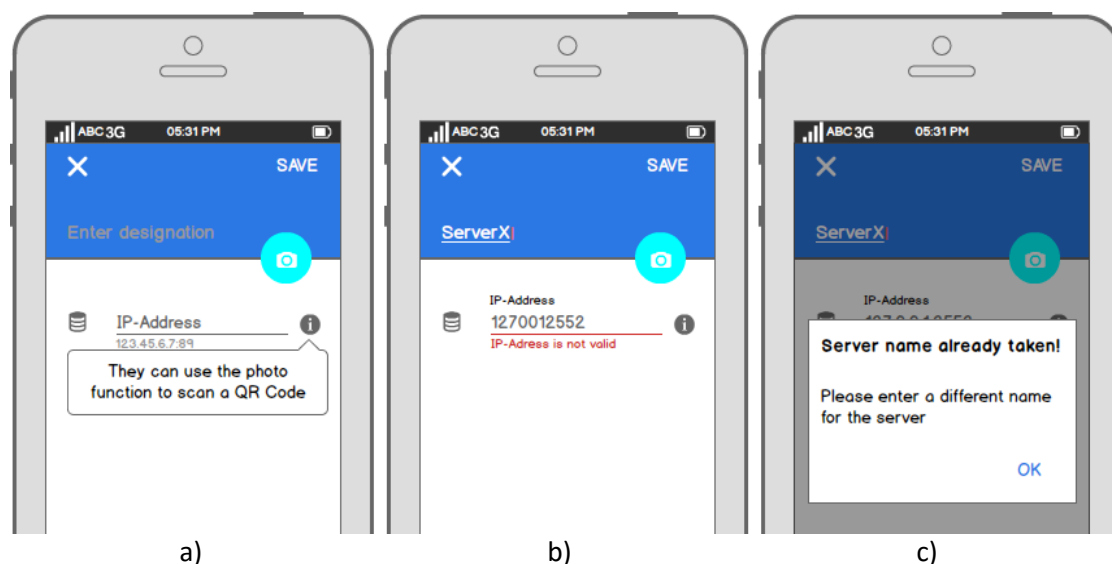


Abbildung 3.37: New Server Activity

Die in Abbildung 3.37 zu sehende „New Server Activity“ soll dem Speichern von Servern auf dem Mobilgerät dienen, damit diese dem Benutzer zur Verfügung stehen, über die auf der „Login Activity“ befindliche Liste und sich Benutzer so schneller mit einem Server verbinden können, ohne dessen Adresse erneut angeben zu müssen.

Die Activity beinhaltet hierfür ein Textfeld, in dem die IP-Adresse sowie der Port des Servers eingetragen werden soll. Links neben dem Textfeld wiederum wird ein 📄-Icon dargestellt, das verdeutlichen soll, dass hier die IP des Servers gemeint ist. Unter dem Textfeld wird eine Beispiel-IP angezeigt, um dem Benutzer zu zeigen, in welcher Form die IP-Adresse angegeben werden muss. Rechts vom Textfeld befindet sich ein ⓘ-Icon, das beim Drücken auf diesen die Information anzeigen soll, dass mit Hilfe der Fotofunktion ein QR-Code gescannt werden kann, der die IP-Adresse und den Port des Servers beinhaltet, damit diese automatisch in das Textfeld eingetragen werden können.

Über dem Textfeld befindet sich die schwimmende Aktionsschaltfläche, die die Scannfunktion anbieten soll. Diese befindet sich nicht wie in der „Login Activity“ unten, da die untere



Fläche keine Liste ist und laut dem „Material Design“ der Button so Elemente verdecken könnte.

Auf der App-Leiste befindet sich ein weiteres Textfeld, um dem zu speichernden Server eine Bezeichnung geben zu können. Damit dieses Textfeld als solches vom Benutzer identifiziert werden kann, steht in dem Feld „Enter designation“, in einer anderen Farbe als der Grundfarbe. Da drüber befinden sich zwei Buttons. Der „X“-Button soll dazu dienen, wieder auf die „Login Activity“ zu gelangen und somit die „New Server Activity“ zu schließen. Der „SAVE“-Button wiederum soll dem speichern des Servers auf dem Mobilgerät dienen und nach erfolgreichem Speichern diesen in der Liste auf der „Login Activity“ darzustellen sowie auf diese zurückkehren.

### 3.2.3 Controllers Activity

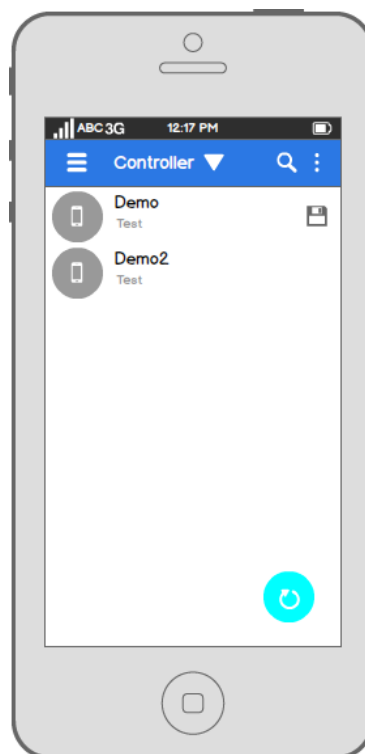





Abbildung 3.38: Controllers Activity

Die in Abbildung 3.38 zu sehende „Controllers Activity“ soll dazu dienen, eine Fernsteuerung aus einer Liste zu wählen und vom Server abzurufen. Diese Liste soll nur die Bezeichnungen der Fernsteuerungen auflisten, die sich auf dem Server oder auf dem Mobilgerät befinden.

Bei den Fernsteuerungen die auf dem Mobilgerät gespeichert wurden, soll ein -Icon angezeigt werden, damit diese als solche zu identifizieren sind. Die Speicherfunktion soll dazu dienen, Fernsteuerungen auf seinem Mobilgerät mit sich zu führen, um so Fernsteuerungen auf unterschiedlichen Servern nutzen zu können, wenn diese die in der Fernsteuerung genutzten Agenten unterstützen.

Da Fernsteuerungen auf dem Mobilgerät gespeichert werden können sollen und es die Möglichkeit geben soll, dass ein anderer Benutzer eine Fernsteuerung mit derselben Bezeichnung erstellen kann, kann es vorkommen, dass Fernsteuerungen mehrmals in der Liste dargestellt werden, jedoch unterscheiden diese sich entweder durch den Autorennamen und/oder durch das -Icon. Durch die Speicherfunktion hat der Benutzer die Möglichkeit eine ältere Version der Fernsteuerung auf dem Mobilgerät zu behalten oder diese zu updaten. Beim Update einer Fernsteuerung soll der Name des Erstellers derselbe sein, um die Fernsteuerung überschreiben zu können.

Durch das -Icon links neben den Fernsteuerungen soll wie auf der „Servers Activity“ dem Benutzer vermittelt werden, dass es sich um Fernsteuerungen in der Liste handelt und nicht mehr um Server.

Weiter unten befindet sich die schwimmende Aktionsschaltfläche, diese soll eine Refresh-Funktion anbieten, um die Liste zu aktualisieren und Fernsteuerungen so vom Server anzufragen, dies könnte auch automatisch geschehen. So könnte nach einer bestimmten Zeit der Server angefragt werden, ob es neue Fernsteuerungen gibt oder der Server könnte den Mobilgeräten mitteilen, dass eine neue Fernsteuerung erstellt wurde, sobald dies gespeichert wird. Es wurde dagegen entschieden, um den Nachrichtenaustausch so gering wie möglich zu halten. So hat der Benutzer die Möglichkeit Fernsteuerungen anzufragen, wenn er es den für nötig hält, zudem wird der Akku des Mobilgerätes dadurch geschont.

Oberhalb der Liste befindet sich wieder die App-Leiste, die dieselben Funktionalitäten anbietet wie auf der „Servers Activity“.

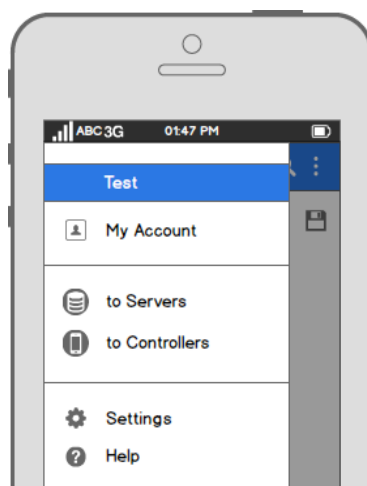


Abbildung 3.39: Navigationsleiste der Controllers Activity

Die Navigationsleiste der „Controllers Activity“ in Abbildung 3.39 bietet zu den vorherigen Buttons „Settings“ und „Help“ auch die Buttons „My Account“, „to Servers“ und „to Controllers“ an und zeigt zudem den Benutzernamen des angemeldeten Benutzers an, so kann gesehen werden, wer gerade angemeldet ist. Der „My Account“-Button soll dazu dienen, die „Account Activity“ anzuzeigen. Der „to Servers“- und „to Controllers“-Button wiederum, soll dazu dienen auf die „Servers Activity“ oder „Controllers Activity“ zu navigieren. Zudem sind die Buttons in Bereiche unterteilt die voneinander getrennt sind, um dem Benutzer eine übersichtlichere und strukturiere Ansicht zu bieten.

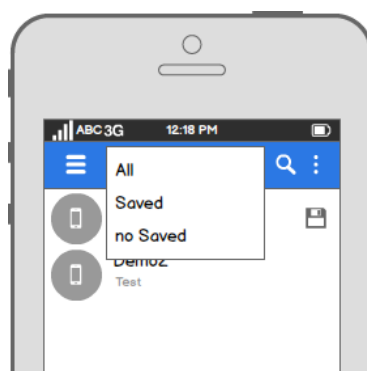


Abbildung 3.40: Filter Funktion der Controllers Activity

Die Filterfunktion in Abbildung 3.40 soll drei Filtermöglichkeiten bieten. Die erste ist „All“, hier sollen alle Fernsteuerungen angezeigt werden. Die zweite ist „Saved“, hier soll dem Benutzer nur die Fernsteuerungen aufgelistet werden, die er auf seinem Mobilgerät gespeichert hat und die letzte ist „not Saved“, diese soll alle nicht gespeicherten Fernsteuerungen auflisten und somit nur die Fernsteuerungen anzeigen, die sich auf dem Server befinden.

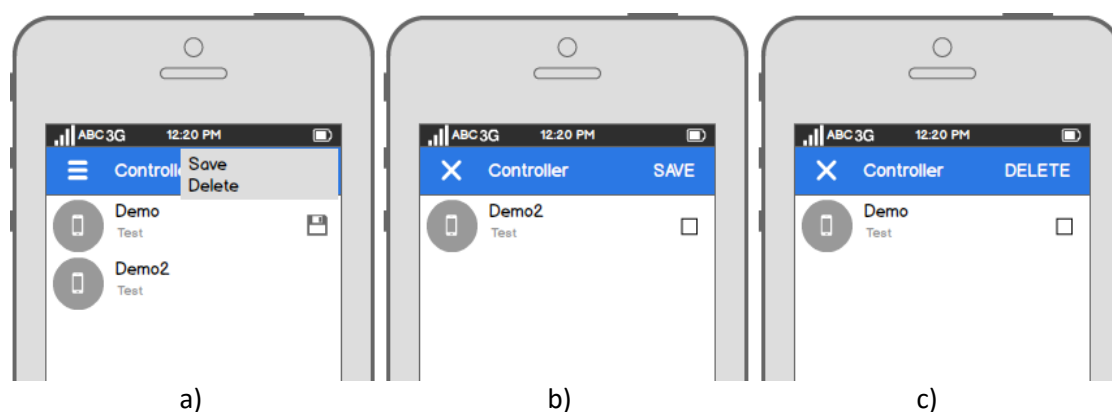



Abbildung 3.41: Menü Funktion der Controllers Activity

Das in Abbildung 3.41a aufgeklappt zu sehende Menü, bietet zwei Funktionen an. Über den „Save“-Button soll die Speicher-Ansicht (b) der Liste geöffnet werden, über die der Benutzer einzelne Fernsteuerungen die vom Server abgerufen wurden, auf dem Mobilgerät speichern können soll. Hierfür soll auf die Checkbox der zu speichernden Fernsteuerung und anschließend auf den Button „SAVE“ gedrückt werden. Der „Delete“-Button wiederum soll die Löschen-Ansicht (c) anzeigen, über die der Benutzer Fernsteuerungen, die auf dem Mobilgerät gespeichert sind, löschen können soll. Auch hier soll zum Löschen der Fernsteuerungen Checkboxen verwendet werden. Beim Drücken auf den Button „SAVE“ oder den Button „DELETE“ sollen die getätigten Änderungen übernommen und die Standardansicht der „Controllers Activity“ angezeigt werden.

Alternativ soll es möglich sein die Speicher- oder Löschen-Ansicht zu öffnen, indem für eine kurze Zeit auf eine Fernsteuerung gedrückt wird. So soll, wenn auf eine Fernsteuerung die bereits gespeichert wurde gedrückt wird, die Löschen-Ansicht sich öffnen. Bei einer nicht-gespeicherten Fernsteuerung wiederum soll sich die Speicher Ansicht öffnen.

Sobald auf eine Fernsteuerung jedoch normal gedrückt wird, soll die Fernsteuerung vom Controller-Agent abgerufen werden. Sollte es sich bei der ausgewählten Fernsteuerung um eine bereits gespeicherte handeln, soll dieser vom Mobilgerät abgerufen werden. Bis zum Laden der Fernsteuerung soll ein entsprechendes -Symbol, dass die Ladezeit darstellen soll, dargestellt werden, wie in Abbildung 3.42a zu sehen.

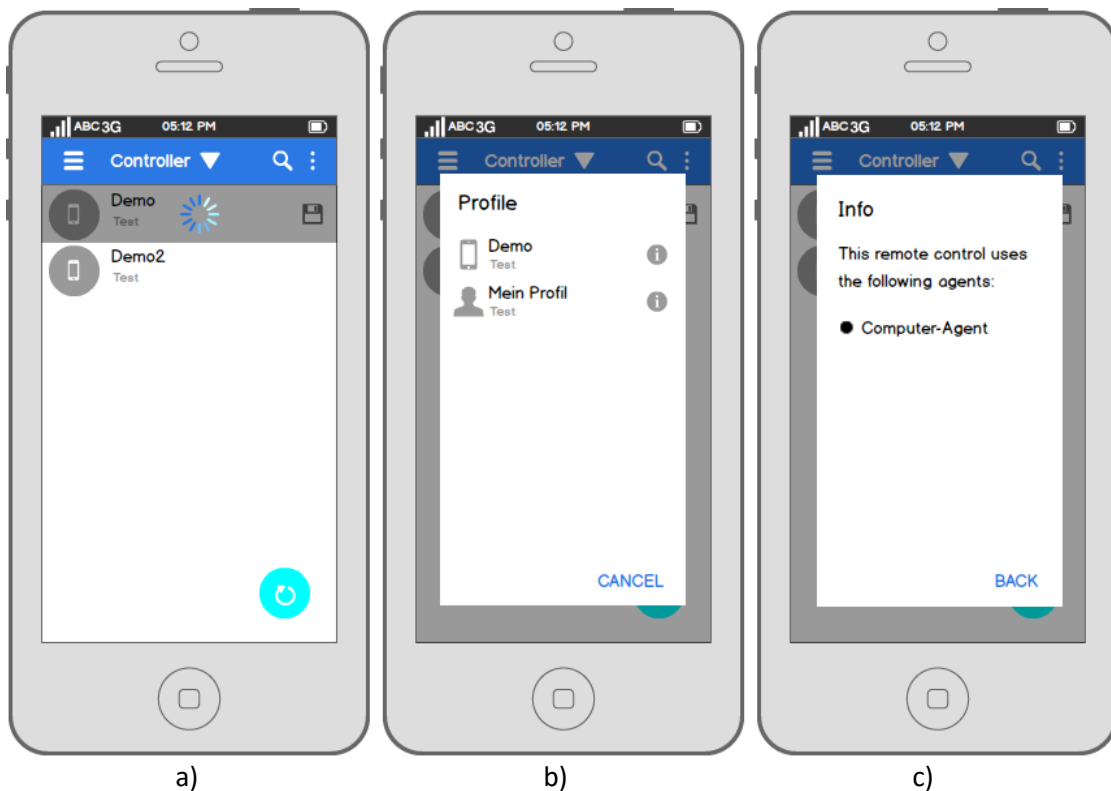


Abbildung 3.42: Profiles Dialogfenster der Controllers Activity

Es soll auch möglich sein diesen Vorgang abbrechen, indem auf eine andere Fernsteuerung oder den „Refresh“-Button gedrückt wird. Der Vorgang soll auch abgebrochen werden, wenn auf eine andere Activity gewechselt wird.

Nachdem die Fernsteuerung geladen wurde, soll diese in einem separaten Dialogfenster in einer Liste angezeigt werden, wie in **b)** zu sehen ist. Hierbei soll das erste Element der Liste die eigentliche Fernsteuerung darstellen, die darauf folgenden Elemente wiederum dessen Profile. Die Profile sollen alphabetisch nach ihrer Bezeichnung geordnet dargestellt werden, zudem sollen hier nur die Profile des angemeldeten Benutzers aufgelistet werden. Sobald auf eine Fernsteuerung oder ein Profil gedrückt wird, soll sich die „Use Controller Activity“ mit dem ausgewählten Profil bzw. der ausgewählten Fernsteuerung öffnen. Sollte die Fernsteuerung keine Profile besitzen, soll sich die „Use Controller Activity“ mit der ausgewählten Fernsteuerung direkt öffnen, anstelle des Profile Dialogfensters. Über den „CANCEL“-Button soll das Dialogfenster geschlossen werden können.

Hierbei wurde sich für ein Dialogfenster entschieden, anstelle einer neuen Activity, da die üblichen Funktionen die sich in der App-Leiste einer Activity befinden würden, wie das Speichern, Löschen oder Favorisieren, nicht benötigt werden, da auf eine Fernsteuerung bereits diese Funktionen angewendet werden können und diese die Profile beherbergen.

Wenn auf das ⓘ-Icon gedrückt wird, soll sich das Info Dialogfenster das in c) zu sehen ist öffnen. Hier sollen dem Benutzer die Agenten aufgelistet werden, die diese Fernsteuerung bedienen können. Über den „BACK“-Button soll der Benutzer wieder auf das „Profiles“-Dialogfenster zurück gelangen.

### 3.2.4 Account Activity

Die in Abbildung 3.43a zu sehende „Account Activity“ soll dem Benutzer dazu dienen, sich vom System abzumelden oder sein Passwort zu ändern. Somit befindet sich auf dieser Activity ein Button, um sich abzumelden, mit der Bezeichnung „LOGOUT“ und ein Button „CHANGE PASSWORD“, um sein Passwort zu ändern. Diese Buttons sollen die Aufmerksamkeit des Benutzers auf sich lenken und sind daher farblich vom Hintergrund hervorgehoben sowie groß geschrieben. Um dem Benutzer eine übersichtlichere und strukturiertere Ansicht zu bieten, sind die beiden Bereiche voneinander visuell getrennt worden.

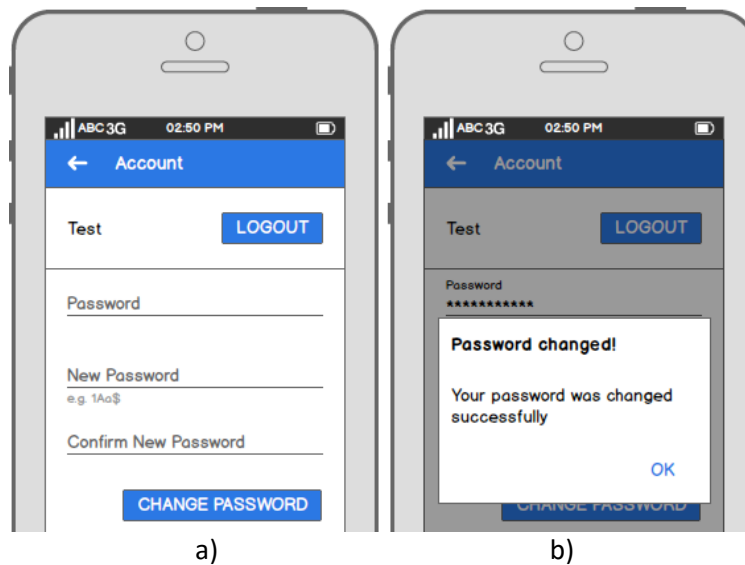


Abbildung 3.43: Account Activity

Sobald auf den „LOGOUT“-Button gedrückt wird, soll der Benutzer abgemeldet werden indem eine Logout-Nachricht an den Server gesendet wird und dieser diese bestätigt, zudem soll der Benutzer nach dem erfolgreichen Abmeldevorgang auf die „Servers Activity“ weitergeleitet werden. Neben dem „LOGOUT“-Button befindet sich der Name des angemeldeten Benutzers, dies soll dem Benutzer verdeutlichen, dass sobald er auf den Button „LOGOUT“ drückt, er sich abmeldet.

Oberhalb des Buttons „CHANGE PASSWORD“ befinden sich drei Textfelder, die der Änderung des Passwortes dienen sollen. Um ein Passwort erfolgreich ändern zu können muss der Benutzer sein aktuelles Passwort eintragen, dies soll der Überprüfung dienen, dass es sich bei dem Benutzer um den angemeldeten Benutzer handelt. Unterhalb des „Password“-Feldes wiederum befinden sich zwei Textfelder, um das neue Passwort einzutragen, hierbei dient das dritte also das „Confirm New Password“-Feld dazu zu prüfen, ob der Benutzer sein Passwort im „New Password“-Feld korrekt eingetragen hat. Sobald der Benutzer nun auf den Button „CHANGE PASSWORD“ drückt, soll geprüft werden ob alle Felder ausgefüllt wurden und dann eine „Change Password“-Nachricht mit den angegebenen Daten und dem Benutzernamen an den Server senden. Falls die Änderung erfolgreich verlief, soll in einem separaten Dialogfenster eine Erfolgsmeldung dargestellt werden (b). Falls jedoch ein Feld fehlerhaft oder nicht ausgefüllt wurde, soll dies durch eine Fehlnachricht die unter dem Textfeld dargestellt werden soll, mitgeteilt werden.

### 3.2.5 Use Controller Activity



Abbildung 3.44: Use Controller Activity

Die in Abbildung 3.44a zu sehende Activity soll dazu dienen, die ausgewählte Fernsteuerung bzw. das Profil darzustellen, damit diese vom Benutzer genutzt werden kann.

Da über eine Fernsteuerung mehrere „Smart-Objects“ gesteuert werden können sollen und diese unterschiedliche Erwartungshaltungen bezüglich der Steuerung aufweisen, wurde entschieden die Steuerungsfunktionalitäten des Mobilgerätes und somit das mentale Modell bezüglich des Trägersystems zu nutzen, damit die Erwartungshaltung während des Nutzens immer die selbige bleibt.

Es wurde entschieden, auf die App-Leiste zu verzichten, damit der Benutzer die gesamte Fläche zur Verfügung stehen hat, auf der seine Fernsteuerung dargestellt werden kann. So hat der Benutzer selbst die Möglichkeit, wenn er den möchte, eine eigene App-Leiste über den GUI-Builder nach seinen Wünschen zu erstellen. Auf die Navigationsleiste jedoch wurde nicht verzichtet, da diese praktische Eigenschaften mit sich bringt. So soll beim Öffnen einer Fernsteuerung ein Hinweis angezeigt werden, dass die Navigationsleiste weiterhin genutzt werden kann (b). Damit aber der Benutzer diese Information nicht jedes Mal zu Gesicht bekommen muss, soll eine Checkbox angeboten werden, die das erneute anzeigen des Hinweises beim Öffnen einer Fernsteuerung verhindern soll.

Die Navigationsleiste (c) soll es zusätzlich zu den schon vorhandenen Funktionen ermöglichen zwischen den Profilen einer Fernsteuerung wechseln zu können. Hierfür sollen die Profile auf der Navigationsleiste über den Einstellungen aufgelistet werden. Es wurde nach eigenem Ermessen entschieden, dass dieser Platz neben dem Platz auf dem sich die Buttons „to Server“ und „to Controllers“ befinden, sich am besten eignet, um diesen mit dem Dauern zu erreichen.

### 3.3 Fazit

Ziel dieses Kapitels war es, eine eigene Benutzeroberfläche zu entwickeln, die alle Anforderungen aus Kapitel 2 sowie das Umsetzen der Szenarien erfüllt. Zu diesem Zweck wurde sich für die Entwicklung einer eigenen Oberfläche entschieden, da diese eine geringere Komplexität aufweist gegenüber einer bereits fertigen (Open Source Oberfläche). So ergab sich, auf Grund des Marktanteiles, bei der Umsetzung der Desktopoberfläche, die „Windows Design Guidelines“ und bei der Umsetzung der Mobile-Oberfläche das „Material Design“ von Google zu berücksichtigen.

Was die Bedienung der Fernsteuerung bzw. der Profile betrifft, so konnte anhand der Tatsache, dass mehrere „Smart-Objects“ über diese gesteuert werden können sollen, gezeigt werden, dass es sinnvoll ist, das mentale Modell bezüglich des Trägersystems zu nutzen, damit die Erwartungshaltung des Benutzers dieselbe ist, bezüglich der Bedienung von „Smart-Objects“. Dahingehend wurde entschieden Buttons und Sliders als grundlegende



Elemente anzubieten, da diese die grundlegenden Elemente eines Mobilgerätes hinsichtlich der Bedingung neben der Touch-Steuerung darstellen, über die der Benutzer dann Aktionen ausführen kann. So ist es beispielsweise möglich, über das Mobilgerät einen Computer zu bedienen. Statt Doppel-Klicks auszuführen, würde man einfach Klicks nutzen. Für die Steuerung des Mauszeigers wiederum kann eine direkte Steuerung über das Display des Mobilgerätes ausgeführt werden. Über einen Agenten wiederum, könnte die direkte Steuerung in eine indirekte umgewandelt werden, damit sich die Steuerung des Zeigers natürlicher anfühlt.

In einem früheren Prototyp wurde ein Sensor-Agent<sup>10</sup> genutzt. Dieser Agent kann Sensorinformationen des Mobilgerätes auslesen, so wurde der Werte des Accelerometers dafür genutzt, um eine Neigung des Mobilgerätes festzustellen und diese auf die Pfeiltasten zu übertragen, dies war jedoch in der Bedingung sehr komplex, da mehrere Faktoren beachtet werden mussten, wie z.B. ab welcher Neigung soll die Pfeiltaste getriggert werden. Auch ob es ein Tablet oder Mobiltelefon war, hat eine große Rolle gespielt, da x- und y-Position vertauscht waren. So wurde vorerst entschieden, die Sensoren für die Bedingung außen vor zu lassen.

---

<sup>10</sup> Sensor-Agent – (Sensor-Agent, vgl.)

# 4 Design

In diesem Kapitel geht es darum, für das System ein Design zu entwickeln, mit zur Hilfe der im vorherigen Kapitel erstellten und spezifizierten Szenarien und Anforderungen an das System.

## 4.1 Architektur

Das hier zu entwickelnde Design besteht aus unterschiedlichen Komponenten. Das System kann aus mehreren mobilen sowie stationären Endgeräten bestehen und einem Nachrichtensystem.

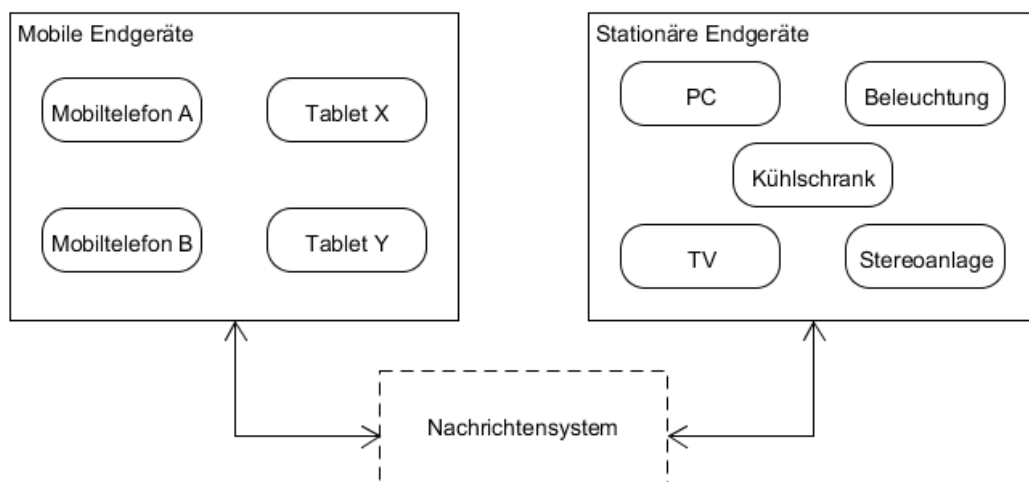


Abbildung 4.1: Systemüberblick

Die Abbildung 4.1 zeigt einen Überblick des Gesamtsystems und die Kommunikationswege untereinander. Hierbei erfolgt der Austausch von Nachrichten zwischen den Komponenten über ein Nachrichtensystem mittels asynchroner Nachrichten. Eine detaillierte Beschreibung der Kommunikation erfolgt in Unterkapitel 4.2.

Um zu erläutern, welche Softwarearchitektur zur Realisierung des hier zu entwickelnden Systems sich am besten eignet, werden Softwarearchitekturen aus dem Bereich verteilte Systeme vorgestellt und erläutert. Des Weiteren wird aufgezeigt, wieso die vorgestellte Softwarearchitektur infrage kommt und welche Nachteile diese mit sich bringen würde:

**Serviceorientierte Architektur:** Im Mittelpunkt dieser Architektur stehen Dienste, die von unterschiedlichen Komponenten innerhalb eines Systems angeboten werden. Im Vergleich zu Komponenten sind Dienste plattformunabhängig und bieten somit eine höhere Interoperabilität, welche die Realisierung verteilter Anwendungen erleichtert. Dienste kapseln Funktionalitäten und Daten und machen diese über wohldefinierte Schnittstellen erreichbar. Für das System ist damit nicht von Bedeutung, welches Gerät in die Umgebung integriert wird, sondern welche Dienste von diesem Gerät zur Verfügung gestellt werden. Zusätzlich können die Dienste auch miteinander kombiniert werden, um dem Anwender komplexere Funktionen zur Verfügung zu stellen. Weitere Aspekte dieser Architektur werden in Schill und Springer (2007) beschrieben. Diese Architektur bietet sich für die Realisierung des Systems an, da diese ein hohes Maß an Flexibilität bietet, durch die lose Kopplung. Dadurch können Komponenten jederzeit hinzugefügt, entfernt oder getauscht werden. Jedoch müssen diese ein hohes Maß an Intelligenz aufweisen, da die Verwaltungsaufgaben hier von keiner zentralen Komponente übernommen werden.<sup>11</sup>

**Die zentralisierte Architektur:** Diese Architektur stellt laut Dunkel u. a. (2008) das grundlegende Modell für verteilte Systeme dar, welche in Server und Client unterteilt wird. Der Client kann auf Wunsch einen Dienst vom Server anfordern. Der Server wiederum beantwortet die Anforderung. Ein solcher Dienst kann im Allgemeinen mit dem Erledigen einer festgelegten Aufgabe gleichgesetzt werden. Diese Architektur bietet sich an, da hier auch ein Client-Server-System realisiert werden soll. Da in diesem System aber auch weitere Komponenten hinzukommen können, in Form von Agenten, bedeutet dies, dass der Server jede Komponente kennen muss. Dazu zählen beispielsweise die Adresse und die angebotenen Dienste eines stationären Endgerätes.<sup>11</sup>

**Peer-to-Peer Architektur:** In dieser Architektur werden die beteiligten nicht mit einer bestimmten Rolle belegt. Jeder Knoten (Peer) in diesem Netzwerk ist ein gleichgestellter Partner. Ein Peer ist in seiner Rolle nicht beschränkt, er kann der Anbieter eines Dienstes sein oder der Kunde, der einen anderen Dienst in Anspruch nehmen möchte (Dunkel u. a. (2008), Dustdar u. a. (2003)). Peers kommunizieren unmittelbar und direkt miteinander.

---

<sup>11</sup> Eine ähnliche Erläuterung der Softwarearchitekturen ist in der Bachelorarbeit von Bornemann (2011) zu finden.

Dies bedeutet, es finden keine Umwege über Server statt. Diese Architektur bietet sich an, da kein „single point of failure“ vorliegt. Falls ein Peer ausfällt, ist das System weiterhin funktionsfähig. Lediglich muss ein neuer Peer gesucht werden, der den gleichen Dienst anbietet. Zudem können Komponenten zur Laufzeit getauscht, hinzugefügt oder entfernt werden. Um diesen Funktionsumfang realisieren zu können, müssen die Komponenten mit zusätzlicher Intelligenz ausgestattet werden. Des Weiteren benötigen die Geräte zusätzliche Schnittstellen, um die Kommunikationsfähigkeit zu gewährleisten. Dadurch wird die Anzahl der infrage kommenden Geräte stark reduziert, da die auf dem Markt verfügbaren Geräte meist nicht die erforderlichen Eigenschaften besitzen.<sup>11</sup>

**Die ereignisgesteuerte Architektur:** Diese Architektur ist eine Softwarearchitektur, bei der das Hauptaugenmerk auf der Erzeugung und Bekanntmachung von Ereignissen liegt. So wird jedes Ereignis einer Komponente, in Form einer Nachricht über eine Middleware versendet. Die Middleware ist dann für die Weiterleitung der Nachricht zuständig. Diese asynchrone Form der Kommunikation ermöglicht eine lose Kopplung der Komponenten (Dunkel u. a. (2008)). Diese Architektur bietet sich für das zu erstellende System an, da Nachrichten interessierten Komponenten bekannt gemacht werden können, ohne detaillierte Informationen über diese zu kennen. Dadurch können dem System weitere Komponenten hinzugefügt werden, ohne merklichen Aufwand. Der Nachteil dieser Architektur stellt die Kommunikation über die Middleware dar. Fällt diese aus, können die Komponenten keine Anfragen mehr an andere Komponenten schicken.<sup>11</sup>

Nun ist zu klären, welche Softwarearchitektur für das hier zu realisierende System am meisten geeignet ist, aufgrund der Vor- und Nachteile der vorgestellten Architekturen.

Aufgrund des Vorteils der einfacheren Einbindung von neuen Komponenten (Agenten) in ein System, sowie den Nachteilen der vorangegangenen Architekturen, wurde bereits im Voraus, für eine ereignisgesteuerte Architektur entschieden. Des Weiteren ermöglicht eine ereignisgesteuerte Architektur eine einfach zu realisierende Kommunikation zwischen den „Smart-Objects“ untereinander. So könnte eine Fernsteuerung erstellt werden, über die indirekt die Heizung bedient wird, wie zum Beispiel eine Fernsteuerung für Fenster. Die Heizung würde ausgeschaltet werden, wenn das Fenster geöffnet wird oder angeschaltet werden, wenn das Fenster geschlossen wird. Des Weiteren lässt sich mit einer ereignisgesteuerten Architektur eine Form des Client-Server-Systems realisieren, dies ist im nachfolgenden Unterkapitel zu sehen. Weitere Informationen zu der gewählten Middleware wiederum folgen in Unterkapitel 4.2.

Um zu erläutern, in welchem Bereich des Client/Server-Modells (Abbildung 4.2), die hier zu realisierende Anwendung sich befinden wird, wird zuerst dieses Modell nach Tanenbaum u. van Steen (2006) zuerst vorgestellt.

Das Client/Server Modell wird grundsätzlich in drei verschiedene Anwendungsschichten unterteilt. Die **Benutzeroberfläche (User interface)** enthält alles, was erforderlich ist, um direkt mit dem Benutzer zu interagieren, wie die Verwaltung der Bildschirmdarstellung. In der **Verarbeitungsebene (Application)** hingegen wird die Kernfunktionalität programmiert. Die persistente Speicherung der Daten erfolgt in der **Datenebene (Database)**. Die Daten werden hierbei unabhängig von der Applikation persistent gespeichert.

Bei einer Unterscheidung von Client- und Server-Seite spricht man von einer two-tier Architektur, hierbei sind folgende Aufteilungen denkbar.

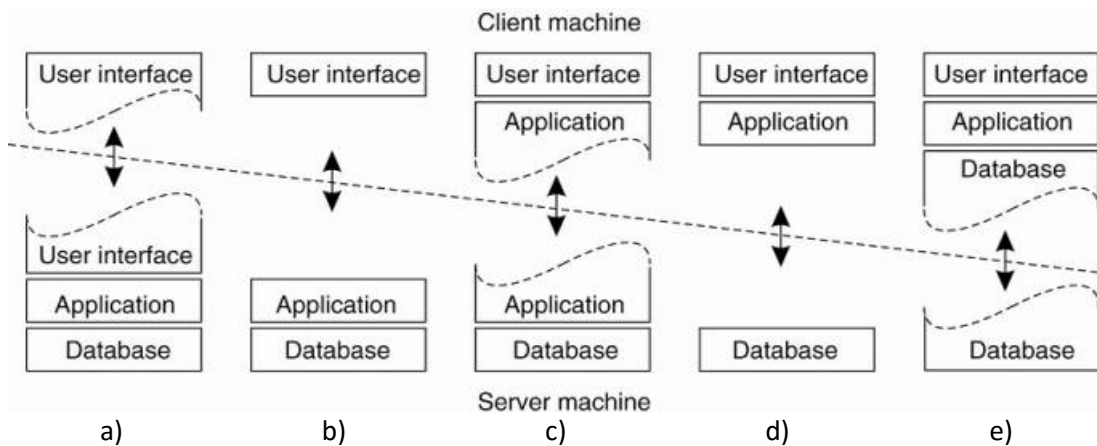


Abbildung 4.2: Einordnung der Trennung des verteilten Systems innerhalb der Architekturschichten. (Tanenbaum und Steen (2006), S.41)

In **a)** ist zu sehen, dass nur ein Teil der grafischen Benutzeroberfläche auf der Client-Seite liegt, dies kann zum Beispiel ein Terminal sein. In **b)** wird das gesamte User Interface vom Client-Seitige Prozess dargestellt, alles andere wird vom Server übertragen. In **c)** wiederum wird ein Teil der Applikation zum Client verlagert, z.B. für einfache Operationen wie das Überprüfen der Korrektheit von Daten, bevor sie zum Server geschickt werden. Der Server wird wie in **d)** zu sehen nur als Persistenzschicht verwendet. Applikationen laufen am Client und führen nur Operationen auf Files oder Datenbanken am Server durch. In **e)** liegen Teile der Daten auch beim Client wie z.B. Caches.

Bei **a)** und **b)** im Client/Server-Modell spricht man von einem Thin Client, bei **d)** und **e)** wiederum von einem Fat Client.

Für das hier zu realisierende System, bietet sich somit ein Fat Client am idealsten an, angesichts der Anforderungen aus der Analyse in Kapitel 2. So soll der Client ein eigenes Interface aufweisen unabhängig vom Server, über den die Fernsteuerungen abgerufen und dargestellt werden sollen. Hierzu kommt noch eine vom Server unabhängige Logik dazu, um diese Fernsteuerungen, wie über den GUI-Builder erstellt, auf dem Client darstellen zu können. Aufgrund der Speichermöglichkeit der Fernsteuerungen sowohl auf dem Client als

auch auf dem Server, handelt es sich um ein Fat Client System, wie in e) zu sehen ist. Dies führt dazu, dass eine geteilte Datenebene vorliegt.

#### 4.1.1 Komponenten

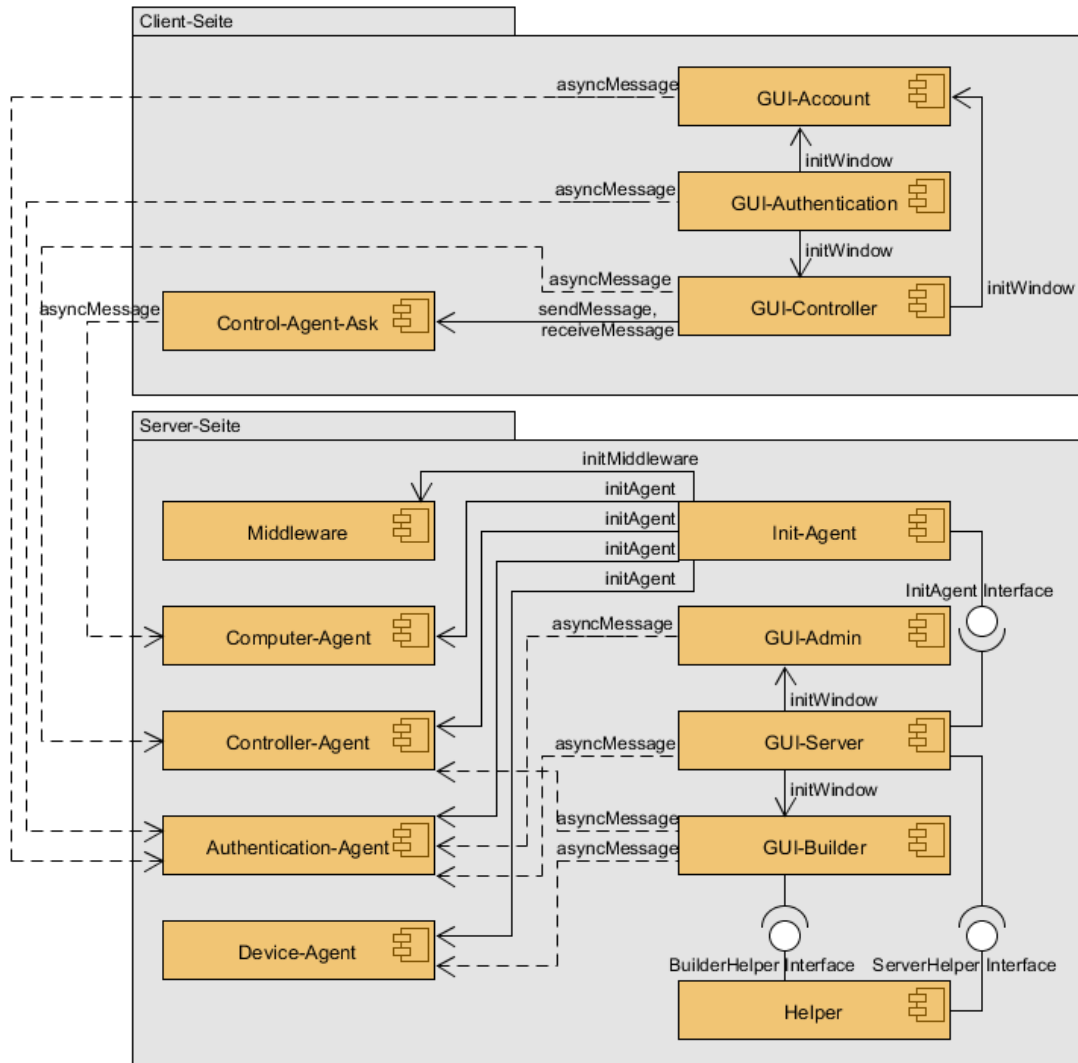


Abbildung 4.3: Komponentendiagramm des Gesamtsystems

Zur Übersicht ist in Abbildung 4.3 das Gesamtsystem als Komponentendiagramm dargestellt. Dies soll verdeutlichen, welche Komponente mit welcher kommuniziert und wo sich diese befindet, zudem soll sie aufzuzeigen, dass das System modular aufgebaut ist, um so Komponenten möglichst einfach auszutauschen. Die gestrichelten Linien sind asynchrone

Nachrichten, die über die Middleware zwischen den Agenten und den Ask-Agenten ausgetauscht werden sollen.

## GUI

Hier soll die aus Kapitel 3 vorgestellte grafische Oberfläche realisiert werden. Hierfür soll das JavaFX-Framework auf der Server- sowie auf der Client-Seite genutzt werden. Durch die Verwendung von JavaFX ist es nach Kostovarov (2013) möglich, die entwickelte Oberfläche in eine Web-Applikation umzuwandeln. Zudem soll auf der Client-Seite Scaloid<sup>12</sup> genutzt werden. Scaloid soll der Implementierung des Clients in Scala anstelle von Java dienen. Die grafischen Oberflächen wurden nach ihrer Zugehörigkeit in Komponenten aufgeteilt, damit diese unabhängig voneinander ausgetauscht oder erweitert werden können. Bei den Ask-Agenten wurde entschieden, diese in die Komponenten mit aufzunehmen, statt eine Komponente zu erstellen die die Ask-Agenten zentral verwaltet, da dies eine bessere Trennung und somit Zugehörigkeit ermöglicht. Die Ask-Agenten dienen der Kommunikation mit den jeweiligen Agenten über die Middleware und werden daher von den Klassen in den Komponenten dazu genutzt, Nachrichten auszutauschen.

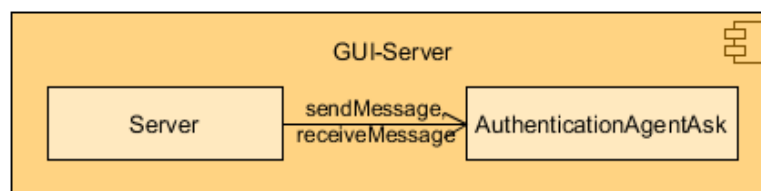


Abbildung 4.4: Komponente „GUI-Server“

Die „GUI-Server“-Komponente besteht aus zwei Klassen und ist in Abbildung 4.4 zu sehen. Die „Server“-Klasse soll die Logik der Serveroberfläche beinhalten. Aufgrund der Verwendung des JavaFX-Frameworks, wird sich die eigentliche Oberfläche in einer .fxml-Datei befinden. FXML<sup>13</sup> ist eine XML-basierte Markup-Sprache, die größtenteils verwendet wird, um den Objekt-Baum von, mit JavaFX erstellten, Benutzeroberflächen zu spezifizieren. Die „Server“-Klasse soll zu gleich auch die Main Klasse auf der Server-Seite darstellen, die beim Start der Anwendung der „Init-Agent“-Komponente Bescheid geben soll, dass diese die Middleware und die Agenten initialisieren kann. Die „AuthenticationAgentAsk“-Klasse wiederum dient dazu, mit der „Authentication-Agent“-Komponente zu kommunizieren.

<sup>12</sup> Scaloid – (Scaloid, vgl.)

<sup>13</sup> FXML – (Javabeginners, vgl.)

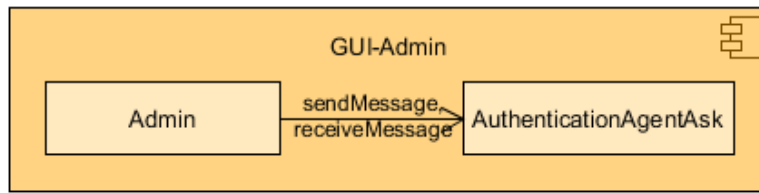


Abbildung 4.5: Komponente „GUI-Admin“

Die „GUI-Admin“-Komponente die in Abbildung 4.5 dargestellt wird, besteht wie die „GUI-Server“-Komponente aus zwei Klassen, wovon die eine auch eine „AuthenticationAgentAsk“-Klasse ist. Die „Admin“-Klasse wiederum beinhaltet die Logik der Admin-Oberfläche.

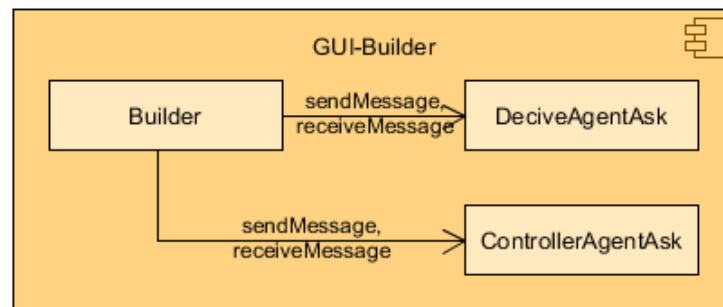


Abbildung 4.6: Komponente „GUI-Builder“

Die „GUI-Builder“-Komponente in Abbildung 4.6 beinhaltet neben den beiden Ask-Agenten die Klasse „Builder“, die die Logik der „Create Remote Controller“-Oberflächen (GUI-Builder) beinhaltet, auf der der Benutzer seine Fernsteuerung erstellen können soll. Da der GUI-Builder über die Serveroberfläche aufgerufen werden soll, soll die „Builder“-Klasse von der „Server“-Klasse initialisiert werden.

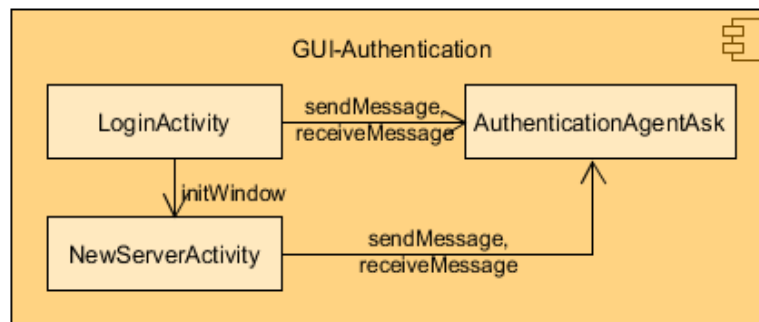


Abbildung 4.7: Komponente „GUI-Authentication“

In der Komponente „GUI-Authentication“, die in Abbildung 4.7 dargestellt ist, soll die Logik der grafischen Oberflächen „Login Activity“ und „New Server Activity“ realisiert werden. Da Server auf dem Mobilgerät gespeichert werden können sollen, soll das Play-JSON-



Framework<sup>14</sup> verwendet werden, damit die Server in ein JSON-Format umgewandelt und als JSON-Datei auf dem Mobilgerät gespeichert werden können. Die Klasse „LoginActivity“ stellt die Main-Klasse der Client-Seite dar. Diese initialisiert die Klassen „NewServerActivity“ und „ControllersActivity“, die sich in der Komponente „GUI-Controller“ befinden, welche wiederum in Abbildung 4.8 dargestellt ist.

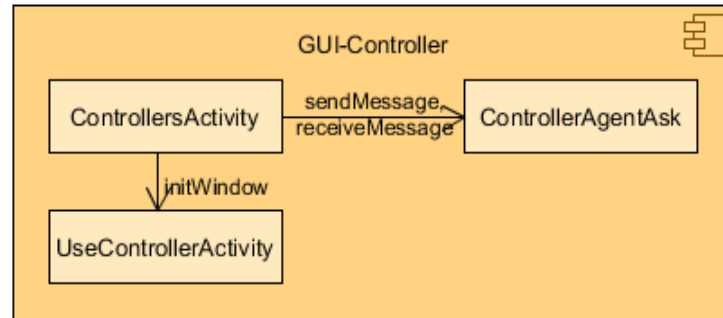


Abbildung 4.8: Komponente „GUI-Controller“

In der „GUI-Controller“-Komponente soll die Logik der grafischen Oberflächen „Controller Activity“ und „Use Controller Activity“ realisiert werden. Da hier Fernsteuerungen auf dem Mobilgerät gespeichert werden sollen, und diese hierfür in ein JSON-Format umgewandelt werden sollen, soll in der „Controller Activity“-Klasse das Play-JSON-Framework verwendet werden. Die „ControllersActivity“ soll die „UseControllerActivity“ initialisieren wenn eine Fernsteuerung ausgewählt wurde. Die Aufgabe der „UseControllerActivity“ ist es dann, die ausgewählte Fernsteuerung bzw. das Profil visuell darzustellen und somit die Elemente umzuwandeln, damit diese auf dem Mobilgerät dargestellt und genutzt werden können.

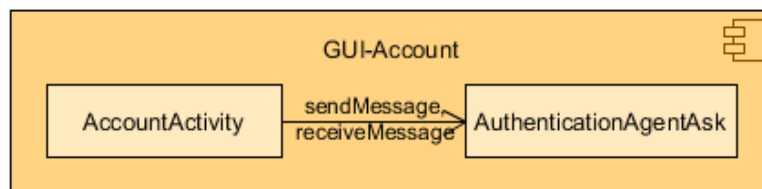


Abbildung 4.9: Komponente „GUI-Account“

Die in Abbildung 4.9 dargestellte „GUI-Account“-Komponente soll die Logik der Oberfläche „Account Activity“ beinhalten. Da diese sich, sobald ein Benutzer Angemeldet hat auf dem Mobilgerät, zu jeder Zeit über jede Oberfläche öffnen lassen soll, soll diese Oberfläche auch über die Komponenten „GUI-Authentication“ und „GUI-Controller“ initialisiert werden können.

<sup>14</sup> Play-JSON-Framework – (Playframework, vgl.)

## Helper

Diese Komponente beinhaltet die Klassen „AgentAPIHelper“ und „ElementHelper“, die beide das Play-JSON-Framework verwenden. Die Komponente wiederum soll die beiden Interfaces „BuilderHelper“ und „ServerHelper“ zur Verfügung stellen. Die Klasse „AgentAPIHelper“ soll dazu dienen APIs der Agenten zu verwalten. So soll sie über das „ServerHelper Interface“ die Funktionen anbieten, APIs in ein JSON Format umzuwandeln und als solche zu speichern, damit diese bei einem Neustart der Anwendung ausgelesen werden können. Die „AgentAPIHelper“-Klasse wiederum soll dazu verwendet werden, die Funktionen aus den JSON-Dateien auszulesen, damit diese dem Benutzer über den GUI-Builder dargestellt werden können. Diese Funktion soll über das „BuilderHelper Interface“ zur Verfügung stehen, so wie die nachfolgenden Funktionen. Die Aufgabe des „ElementHelper“ wiederum soll es sein, die Elemente auf der Fernsteuerung umzuwandeln, damit diese an den „Controller-Agent“ gesendet und auf diesem gespeichert werden können. Dies muss getan werden, da bei der Erstellung der Fernsteuerungen auf dem GUI-Builder JavaFX Elemente verwendet werden sollen, um diese visuell darzustellen. So muss auch die „ElementHelper“-Klasse eine Funktion anbieten, um die Elemente auf der abgerufenen Fernsteuerung vom „Controller-Agenten“ in JavaFX Elemente umzuwandeln, damit diese auf dem GUI-Builder angezeigt werden können. Die Abbildung 4.10 stellt die Innensicht der Komponente als UML-Diagramm dar.

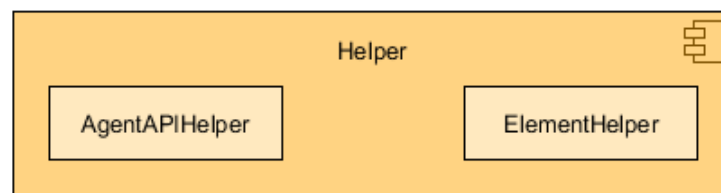


Abbildung 4.10: Komponente „Helper“

## Init-Agent

Diese Komponente soll dazu dienen, die Middleware und die Agenten zu starten, damit die Kommunikation über die Agenten stattfinden kann. Hierfür beinhaltet die Komponente die Klasse „InitAgent“, die dies machen soll. Die Abbildung 4.11 stellt die Innenansicht der Komponente als UML-Diagramm dar. Des Weiteren soll diese Komponente den QR-Code generieren und liefern, der benötigt wird, um sich mit dem Server zu verbinden, sobald die Middleware läuft. Für das Generieren des QR-Codes soll der „qrcodegenerator“<sup>15</sup> verwendet werden. Des Weiteren soll diese Komponente Funktionen über das „InitAgent Interface“ anbieten, mit der die Middleware gestartet, gestoppt oder neu gestartet werden kann.

---

<sup>15</sup> qrcodegenerator – (Qrcodegenerator, vgl.)

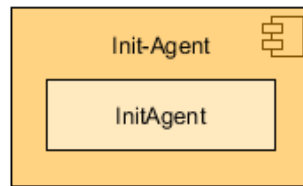


Abbildung 4.11: Komponente „Init-Agent“

## Agenten

Die Komponenten der Agenten sind nach dem gleichen Schema aufgebaut und bestehen daher aus einer Agent- und einer Logik-Klasse. Die Agents-Klassen sollen zur Kommunikation mit den Ask-Agenten dienen. Die Logik-Klassen wiederum beinhalten die Logik des Agenten. So sollen bei Anfragen von Ask-Agenten diese vom jeweiligen Agenten entgegen genommen und in der Logik-Klasse umgesetzt werden. Nach der Umsetzung soll dann über den Agenten an den Ask-Agent eine Nachricht mit dem Resultat gesendet werden. Im Falle eines Fehlers wiederum soll eine Fehlermeldung gesendet werden.

Die Agenten sollen in Scala implementiert werden, da Nachrichten durch Case-Klassen repräsentiert werden können. Um nicht zwischen den Sprachen wechseln zu müssen, wurde zudem entschieden die gesamte Anwendung in Scala zu implementieren.

Für die Implementierung soll eine asynchrone Programmierung genutzt werden, so ist zu keinem Zeitpunkt eine Blockierung des Programmablaufs notwendig. Hierfür sollen asynchrone Aufrufe mittels Futures realisiert werden. Futures sind anonyme asynchrone Funktionen, die als Teil der Scala-Bibliothek implementiert sind. Anstatt auf eine Berechnung zu warten, um daraus einen Rückgabewert zu erstellen, soll ein „Future“ generiert werden. Bei jedem „Future“ können beliebig viele Callbacks registriert werden, die ausgeführt werden, sobald ein Ergebnis vorliegt. Damit lässt sich das Warten auf Ressourcen oder Ereignisse vollständig vermeiden. Um auszuschließen, dass Futures nicht terminieren, sollen Timeouts angegeben werden. Wird die angegebene Zeit überschritten, ist das Ergebnis des Futures ein Fehler. Dies trifft auch zu, wenn während der Ausführung eine Exception geworfen wird. Deswegen soll der Fehlerfall ebenfalls durch Callbacks abgedeckt werden. (vgl. [Eichler \(2014\)](#), S.56)

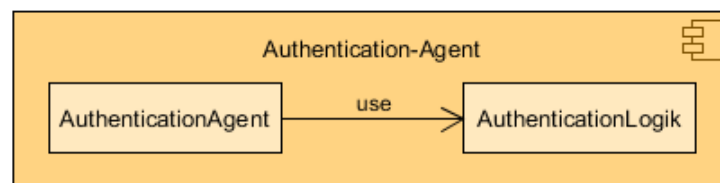


Abbildung 4.12: Komponente „Authentication-Agent“

Die „Authentication-Agent“-Komponente in Abbildung 4.12 soll anfragen, wie das Registrieren oder Einloggen eines Benutzers bearbeiten, zudem soll diese die registrierten Benutzer verwalten. Hierfür sollen diese in einer Datenbank gespeichert werden. Bei der Auswahl der Datenbank wurde sich für „H2 Database Engine“<sup>16</sup> und das Framework „Sorm“<sup>17</sup> entschieden, da die Einarbeitungszeit durch die ausführlichen Beispiele auf deren Seiten minimal ist und die Features gut zu den Anforderungen passen.

Des Weiteren soll der Agent den sechsstelligen Key generieren, der für die Anmeldung und Registrierung über das Mobilgerät benötigt wird.

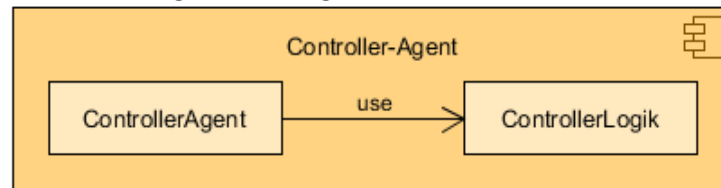


Abbildung 4.13: Komponente „Controller-Agent“

Die „Controller-Agent“-Komponente in Abbildung 4.13 soll anfragen, wie das Speichern, Löschen, Suchen und Abrufen von Fernsteuerungen bearbeiten, zudem soll für die Verwaltung der Fernsteuerungen das „Sorm-Framework“ und die „H2 Database Engine“ verwendet werden.

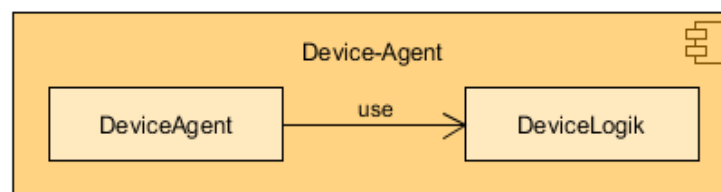


Abbildung 4.14: Komponente „Device-Agent“

Die „Device-Agent“-Komponente in Abbildung 4.14 soll anfragen, wie das Abrufen einer Liste mit den Namen der aktuellen Mobilgeräte und das Abrufen spezifischer Informationen zu einem Mobilgerät, wie dessen Bild und Maße, dienen. Hierfür sollen sich die Bilder im selben Verzeichnis wie der Agent befinden und im Ordner „Devices“ gespeichert werden. Zudem soll sich in dem Ordner eine JSON-Datei zu jedem Bild befinden, die auf das Bild verweist und die Maße des darauf abgebildeten Mobilgerätes beinhaltet. Dies soll so gemacht werden, damit der Benutzer später über eine Plattform weitere Devices herunterladen und diese in den Ordner hinzufügen und somit nutzen kann.

<sup>16</sup> H2 Database Engine – (H2Database, vgl.)

<sup>17</sup> Sorm – (Sorm-Framework, vgl.)

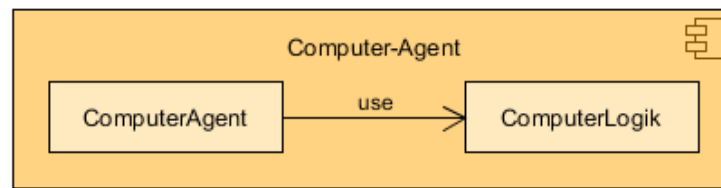


Abbildung 4.15: Komponente „Computer-Agent“

Die „Computer-Agent“-Komponente in Abbildung 4.15 ist für die Umsetzung der Aktionen auf der Server-Seite zuständig, die über die Fernsteuerung auf dem Client getätigt werden, sobald ein Button auf dem Profil gedrückt wurde.

### Control-Agent-Ask

Diese Komponente beinhaltet die Klasse „ControlAgentAsk“ die zur Kategorie Ask-Agent gehört aber sich in einer eigenen Komponente befindet. Dieser soll dazu dienen, mit den Agenten zu kommunizieren, die außerhalb der Anwendung liegen und soll hierfür die Nachrichten generisch zur Laufzeit erstellen und verschicken. Um dies zu machen soll die in den Elementen enthaltene „Action“ in eine Nachricht umgewandelt und an den in der „Action“ angegebenen Agenten gesendet werden. Sollte dazu in der „Action“ eine Antwort-Aktion, also eine Antwortnachricht angegeben sein, soll ein Prozess gestartet werden, der auf diese Antwortnachricht wartet. Die Abbildung 4.16 stellt die Innenansicht der Komponente als Komponentendiagramm dar.

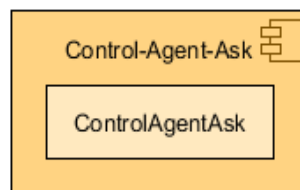


Abbildung 4.16: Komponente „Control-Agent-Ask“

### Programmablauf beim Start der Anwendung

Wenn man den Programmablauf in Worte fassen soll, dann muss das wie folgt aussehen:

Nach dem Start des Servers (GUI-Server) soll die „InitAgent“-Klasse die Middleware und die Agenten initialisieren. Nachdem die Agenten initialisiert wurden, soll vom Authentication-Agent der Key angefordert werden. Der Key soll alle 60 Sekunden vom Agenten angefordert werden. Der wiederum bei jeder Anforderung einen neuen generieren soll. Der Server soll danach die „AgentAPIHelper“-Klasse nutzen um nach APIs zu suchen, die bereits als JSON-Format gespeichert wurden und diese auf der Serveroberfläche anzeigen.

Die JSON-Dateien sollen in einem Ordern gespeichert werden, die sich im selben Verzeichnis wie die Anwendung befindet und den Namen „Agent APIs“ trägt. Standardmäßig soll sich die API des „ComputerAgent“ in diesem als JSON-Datei befinden. Dieser soll sich dort befinden, da gewollt ist, dass der „ComputerAgent“ dem Benutzer von Beginn an zur Verfügung stehen soll (so kann der Benutzer nach dem Start des Servers über den GUI-Builder eine Fernsteuerung erstellen mit dem er seinen Computer steuern kann). Daher befindet sich dieser Agent, der von einer Fernsteuerung aus Aktionen ausführen kann, auf der Server-Seite und nicht außerhalb des Servers, wie es im Regulären der Fall wäre.

Nachdem der Server gestartet und die Middleware initialisiert wurde, können sich die Clients mit dem Server verbinden. Hierfür muss nun der Client gestartet werden. Nachdem der Client gestartet wurde, soll dieser auf dem Gerät nach bereits gespeicherten Servern suchen und diese auf der „LoginActivity“, die gleichzeitig die Startseite darstellt, anzeigen. Über die Startseite kann sich der Benutzer nun mit einem der gefunden Server verbinden (wenn der ausgewählte Server den läuft) oder einen Server hinzufügen, um sich mit diesem dann zu verbinden.

#### 4.1.2 Aufbau einer Fernsteuerung

Bei einer Fernsteuerung soll es sich auch um ein Profil handeln, somit gibt es im System nur Profile. Eine Fernsteuerung sowie ein Profil sind im Grunde das gleiche. Der Unterschied liegt darin, dass ein Profil nur von seinem Ersteller bearbeitet und genutzt werden können soll. Ein weiterer Unterschied ist, dass eine Fernsteuerung mehrere Profile beinhalten kann. So sieht eine Fernsteuerung folgendermaßen aus.

- `Controller(name: String, profiles: List[Profile], author: String, deviceName: String, activityHeight: Double, activityWidth: Double, activitys: List[Activity])`

Ein Profil wiederum hat einen fast identischen Aufbau, mit dem Unterschied, dass dieses keine weiteren Profile beinhaltet.

- `Profile(name: String, author: String, deviceName: String, activityHeight: Double, activityWidth: Double, activitys: List[Activity])`

Daher wurde, wie am Anfang beschrieben, entschieden, nur Profile zu nutzen. Der Aufbau einer vollständigen Fernsteuerung soll somit wie folgt aussehen.

- `Controller(name: String, author: String, profiles: List[Profile])`

Um weiterhin zwischen Fernsteuerung und Profil unterscheiden zu können, wurde in der Fernsteuerung der Parameter „name“ belassen. Anhand dessen soll die Fernsteuerung aus „profiles“ ermittelt werden, da diese dieselbe Bezeichnung auch im Profil tragen soll. Der Parameter „author“ wiederum soll der Eindeutigkeit der Fernsteuerung dienen, da unterschiedliche Benutzer Fernsteuerungen mit derselben Bezeichnung erstellen können.

Dass die Parameter „name“ und „author“ weiterhin in der Fernsteuerung bestehen sollen, hat einen weiteren Grund. Alternativ könnte ein Parameter vom Typ Boolean in „Profile“ genutzt werden, um die Fernsteuerung zu ermitteln, wie z.B. „isController“. Die Parameter sollen dazu dienen, den Namen und den Autor der Fernsteuerung schneller zu liefern, ohne über die Profile iterieren zu müssen, da diese Angaben zur Auswahl einer Fernsteuerung benötigt werden, wie in Unterkapitel 3.2.3 Controllers Activity Abbildung 3.38 gezeigt wurde.

- `Profile(name: String, author: String, deviceName: String, activityHeight: Double, activityWidth: Double, activities: List[Activity])`

Damit wie beschrieben ein Profil nur vom Ersteller genutzt werden kann, befindet sich hierfür im Profil der Parameter „author“. Der Parameter „deviceName“ dient wiederum dazu, das Mobilgerät zu ermitteln. Dieses wird bei der Erstellung eingetragen und wird benötigt, sobald die Fernsteuerung abgerufen und auf dem GUI-Builder dargestellt werden soll. Die Parameter „activityHeight“ und „activityWidth“ sind relevant für die Umrechnung der Elemente auf die Displaygröße des Mobilgeräts. Der Parameter „activities“ beinhaltet eine Liste der Activities.

- `Activity(name: String, mode: String, isMain: Boolean, elements: List[Element])`

Eine Activity besteht aus dem Namen der Activity, wie oben zu sehen ist, sowie dem Parameter „mode“. Dieser besagt, ob die Activity im Landscape- oder Portrait-Modus nutzbar sein soll. So sollen, wenn kein Modus eingetragen ist (Leerer String), beide Modis auf dem Mobilgerät nutzbar sein. Der „isMain“ Parameter soll der Ermittlung der Main-Activity dienen. Die Main-Activity ist die erste Activity, die dargestellt werden soll, sobald auf dem Mobilgerät eine Fernsteuerung ausgewählt wurde. Des Weiteren beinhaltet eine Activity eine Liste der Elemente, die sich auf ihr befinden.

- `Element(name: String, type: String, bounds: Bounds, action: Action)`

Der Parameter „name“ in Element soll dem Beschrifteten des Elementes dienen und soll somit auf dem visuellen Element dargestellt werden. Der Parameter „Type“ soll den Typ des Elementes angeben, so kann dieser ein „Button“, „Slider“ oder „Toggle-Button“ sein. Die Positions- sowie Größeninformationen sollen in dem Parameter „bounds“ angegeben wer-

den. Diese beinhaltet die x-, y- und z-Position sowie die Höhe und Breite des Elementes, was unten dargestellt ist.

- `Bounds(xPos: Double, yPos: Double, zPos: Double, height: Double, width: Double)`

Der in den Elementen enthaltene Parameter „action“ bildet das Kernstück einer Fernsteuerung, da diese die Aktion beinhalten soll, mit dessen Hilfe „Smart-Objects“ bedient werden können.

- `Action(agentInfo: String, requestAction: RequestAction, answerAction: AnswerAction)`

Der in der Aktion enthaltene Parameter „agentInfo“ beinhaltet den Namen des Agenten, an den die Aktion gesendet werden soll. Die Parameter „requestAction“ und „answerAction“ wiederum beinhalten die eigentliche Aktion („message“). So soll die „requestAction“ die Nachricht beinhalten, die an den angegebenen Agenten gesendet werden soll. Die „answerAction“ wiederum soll die Antwortnachricht beinhalten, welche vom Agenten als Reaktion auf die „requestAction“ gesendet wird.

- `RequestAction(groupRequest: String, requestMessage: Message)`

- `AnswerAction(groupAnswer: String, answerMessage: Message, errorMessage: Message)`

Beide Aktionen beinhalten einen Gruppenparameter. Der Parameter „groupRequest“ gibt die Gruppe an, auf den der Agent hört, an diese soll die Nachricht („requestMessage“) gesendet werden. Der Parameter „groupAnswer“ wiederum gibt die Gruppe an, an den die Antwortnachricht („answerMessage“) vom Agenten gesendet wird, damit diese empfangen werden kann. Der Parameter „requestMessage“ stellt die Aktion dar, die vom Agenten umgesetzt werden soll. Der Parameter „answerMessage“ wiederum stellt die erwartete Nachricht dar, die vom Agenten gesendet wird. Der Parameter „errorMessage“ stellt die erwartete Fehlernachricht dar, die vom Agenten im Falle eines Fehlers gesendet wird.

Eine Nachricht beinhaltet den Namen der Nachricht sowie eine Liste mit Parametern. Die Parameter wiederum bestehen aus dem Namen und dem Typ des Parameters sowie dem Wert, der über den GUI-Builder zugewiesen wurde.

- `Message(name: String, parameters: List[Parameter])`

- `Parameter(name: String, type: String, value: String)`



Eine Nachricht bildet somit eine Case-Klasse nach. Diese soll an den Agenten gesendet werden oder den Aufbau der zu empfangenden Nachricht beschreiben.

### 4.1.3 Umrechnung einer Fernsteuerung

Die Umrechnung bezieht sich hierbei auf die Elemente, die sich auf einer Fernsteuerung bzw. auf einem Profil befinden, damit diese, wie über den GUI-Builder erstellt, auf dem Mobilgerät dargestellt werden.

Eine Umrechnung der Elemente soll auf dem Mobilgerät erfolgen, da eine vorab Umrechnung keinen Sinn macht. So müssten bei einer vorab Umrechnung alle Profile, die in der Fernsteuerung enthaltenen sind, mit umgerechnet werden, obwohl der Benutzer nur ein bestimmtes Profil nutzen möchte. Des Weiteren müsste die Displaygröße des Mobilgeräts abgefragt werden. Was bedeutet, dass eine Nachricht mehr gesendet werden müsste, die zu Gunsten der Laufzeit vermieden werden kann.

Für die Umrechnung der Elemente wurden vier Formeln aufgestellt, die wie folgt lauten:

1. X Position des Elementes auf dem Mobilgerät = X Position des Elementes auf der Activity \* Delta
2. Y Position des Elementes auf dem Mobilgerät = Y Position des Elementes auf der Activity \* Delta
3. Breite des Elementes auf dem Mobilgerät = Breite des Elementes auf der Activity \* Delta
4. Höhe des Elementes auf dem Mobilgerät = Höhe des Elementes auf der Activity \* Delta

Der Delta-Wert spielt hierbei eine besondere Rolle, da dieser den Faktor vorgibt, um wie viel die Elemente vergrößert bzw. verkleinert werden müssen, damit diese der Größe (Höhe, Breite, X-Pos, Y-Pos, Schriftgröße) entsprechend auf dem Mobilgerät richtig dargestellt werden können. Der Wert Delta muss je nach Bildmodus (Portrait/Landscape) der Fernsteuerung errechnet werden. Sollte in der Fernsteuerung kein Bildmodus festgelegt sein, so soll der Bildmodus des Mobilgeräts zur Ermittlung des Delta-Wertes genutzt werden. Somit ergeben sich je nach Modus folgende Berechnungen.

Bildmodus Portrait (Vertikal):

- $\Delta = \text{Mobilgerät Höhe} / \text{Activity Höhe}$

Bildmodus Landscape (Horizontal):

- $\Delta = \text{Mobilgerät Breite} / \text{Activity Breite}$

Folgend ist ein Beispielszenario dargestellt, das die Anpassung der Elemente auf die Displaygröße des Mobilgerätes erläutern soll:

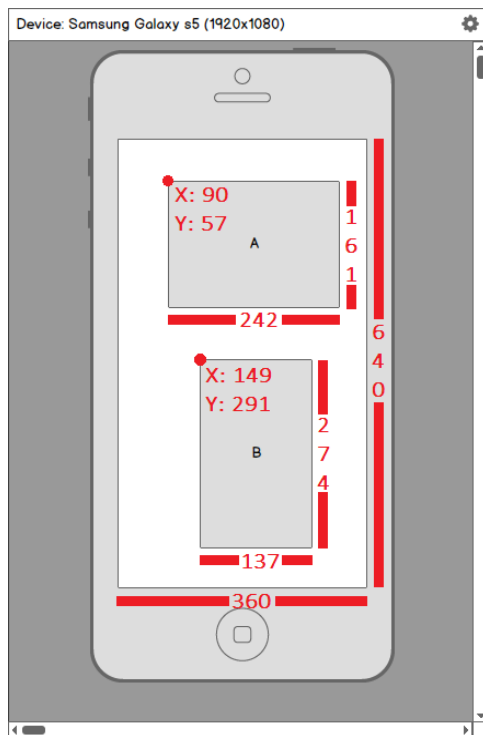


Abbildung 4.17: Erstellte Fernsteuerung über den GUI-Builder (Sicht auf den Gerätebereich).

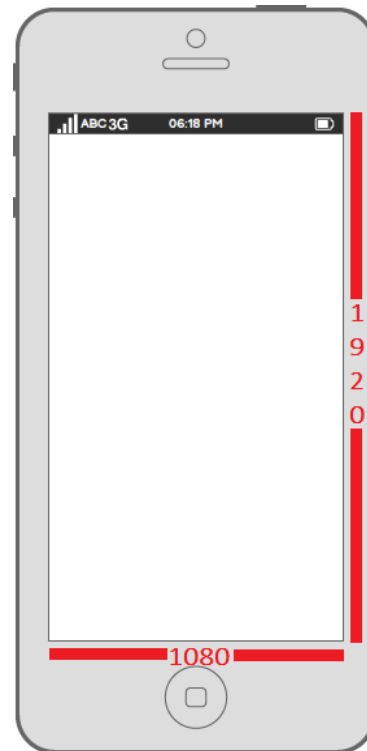


Abbildung 4.18: Display Maße des Mobilgeräts für die Umrechnung.

Über den GUI-Builder wurde eine Fernsteuerung, wie in Abbildung 4.17 auf dem Gerätebereich zu sehen ist, erstellt und soll nun auf dem Mobilgerät in Abbildung 4.18 genutzt werden. Um die Umrechnung zu verdeutlichen wurden Beispielzahlen eingesetzt, die auch so in der Realität vorkommen können.

So muss laut der Formel folgendes Ergebnis rauskommen:  
Bildmodus Portrait: Delta =  $1920 / 640 = 3$

	Element: A	Element: B
Formel: 1.	X-Pos = $90 * 3 = 270$	X-Pos = $149 * 3 = 447$
Formel: 2.	Y-Pos = $57 * 3 = 171$	Y-Pos = $291 * 3 = 873$
Formel: 3.	Höhe = $242 * 3 = 726$	Höhe = $137 * 3 = 411$
Formel: 4.	Breite = $161 * 3 = 483$	Breite = $274 * 3 = 822$

In Abbildung 4.19 ist das Resultat visuell dargestellt:

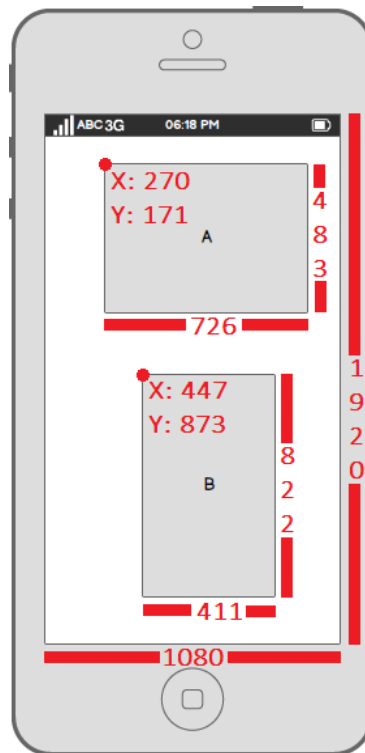


Abbildung 4.19: Umgewandelte Fernsteuerung auf dem Mobilgerät.

Da die Berechnung korrekt und somit Fernsteuerungen erfolgreich umgewandelt werden können, ist daran zu erkennen, dass die angezeigten Elemente auf dem Mobilgerät (Abbildung 4.19) im Verhältnis 3:1 zu den auf dem GUI-Builder dargestellten Elementen entsprechen und demnach um den Faktor drei vergrößert wurden.

Durch die Auswahl der Delta Berechnung für den entsprechenden Bildmodus der Fernsteuerung wird zugleich ein weiteres Problem gelöst, das wie folgt aussehen würde. Sollte die in Abbildung 4.17 erstellte Fernsteuerung (360 x 640 | 9:16) auf einem Mobilgerät (Portrait) genutzt werden, dessen Auflösung 1440 x 1920 und damit einem Bildformat von 3:4 entspricht, würden die Elemente, ohne die Auswahl einer bestimmten Delta-Berechnung gestreckt, gestaucht oder falsch positioniert dargestellt werden. Durch die Errechnung des Delta-Wertes für einen bestimmten Bildmodus, wird dies jedoch vermieden, da dieser den Faktor für die größtmögliche Auflösung der Fernsteuerung vorgibt. So würde dieser weiterhin 3 lauten, da in dem Beispiel die größtmögliche Auflösung 1080 x 1920 beträgt.

Beweis:

Gegeben:	Gesucht:
Mobilgerät Höhe: 1920	Delta: ?
Mobilgerät Breite: 1440	Neue Fernsteuerung Höhe: ?
Mobilgerät Bildmodus: Portrait	Neue Fernsteuerung Breite: ?
Fernsteuerung Höhe: 640	
Fernsteuerung Breite: 360	

Berechnung:

Delta = Mobilgerät Höhe / Fernsteuerung Höhe

$$\text{Delta} = 1920 / 640 = \underline{3}$$

Neue Fernsteuerung Höhe = Fernsteuerung Höhe \* Delta

$$\text{Neue Fernsteuerung Höhe} = 640 * 3 = \underline{1920}$$

Neue Fernsteuerung Breite = Fernsteuerung Breite \* Delta

$$\text{Neue Fernsteuerung Breite} = 360 * 3 = \underline{1080}$$

Da nun jedoch die Breite der Fernsteuerung (Breite: 1080) mit der des Mobilgeräts (Breite: 1440) nicht mehr übereinstimmt, soll die Fernsteuerung auf dem Mobilgerät 180 Pixel ((Breite Mobilgerät - Neue Breite der Fernsteuerung) / 2) vom linken Rand des Mobilgeräts entfernt dargestellt werden, so dass sich links und rechts von der Fernsteuerung ein schwarzer Balken mit jeweils einer Breite von 180 und einer Höhe von 1920 Pixeln befindet. Dadurch soll die Fernsteuerung auf dem Mobilgerät mittig dargestellt werden.

Jedoch besteht weiterhin ein Problem, das dazu führt, dass die Fernsteuerung und somit die darin enthaltenen Elemente auf dem Mobilgerät falsch dargestellt werden. So wäre es beispielsweise bei einer Auflösung von 1440 x 1920, dass im Landscape Modus genutzt wird, ein Problem, dass die Landscape Berechnung für Delta genutzt werden würde. Dies würde somit für eine Fernsteuerung mit einer Auflösung von 360 x 640 den Delta-Wert 4 und eine neue Auflösung für die Fernsteuerung von 1440 x 2560 errechnen. Diese neu errechnete Auflösung wäre zu groß für das Mobilgerät und könnte nicht ordentlich nutzbar dargestellt werden.

Design-Guides wie das „Material design“ von Google bieten unterschiedliche Lösungen für dieses Problem an. So können die Elemente neu angeordnet dargestellt oder eine Scroll-Funktion angeboten werden. Eine Neuordnung der Elemente macht hier keinen Sinn, da die Fernsteuerung nicht mehr der über den GUI-Builder erstellten Fernsteuerung entsprechen würde und es sich somit um eine andere Fernsteuerung handeln würde, in die der Benutzer sich erst einarbeiten müsste. Jedoch macht auch eine Scroll-Funktion keinen Sinn,

da es gewollt ist, dass der Benutzer die Fernsteuerung als Ganzes auf dem Mobilgerät überblicken können soll, um so jede darauf befindliche Aktion schnell zu nutzen. Durch eine Scroll-Funktion müsste der Benutzer erst nach unten oder zur Seite scrollen, um die dort befindlichen Elemente und die damit verknüpfte Aktion auszuführen.

Da beide Varianten in diesem Fall keinen Sinn machen und es keine wirklich brauchbare Lösung für dieses Problem gibt, wurde für die Scroll-Funktion entschieden, da der Nachteil einer Neuordnung der Elemente und die damit verknüpfte Eingewöhnung als schwerwiegender empfunden wurde, so wie die Tatsache, dass es sich aus der Perspektive des Benutzers nicht mehr um die Fernsteuerung handelt, die er erstellt hat. In Abbildung 4.20 ist ein Beispiel dargestellt, wie dies mit einer Scroll-Funktion aussehen könnte. Hierbei soll die Fernsteuerung von oben beginnend dargestellt werden. Im Falle von Portrait wiederum soll diese von links beginnend dargestellt werden.

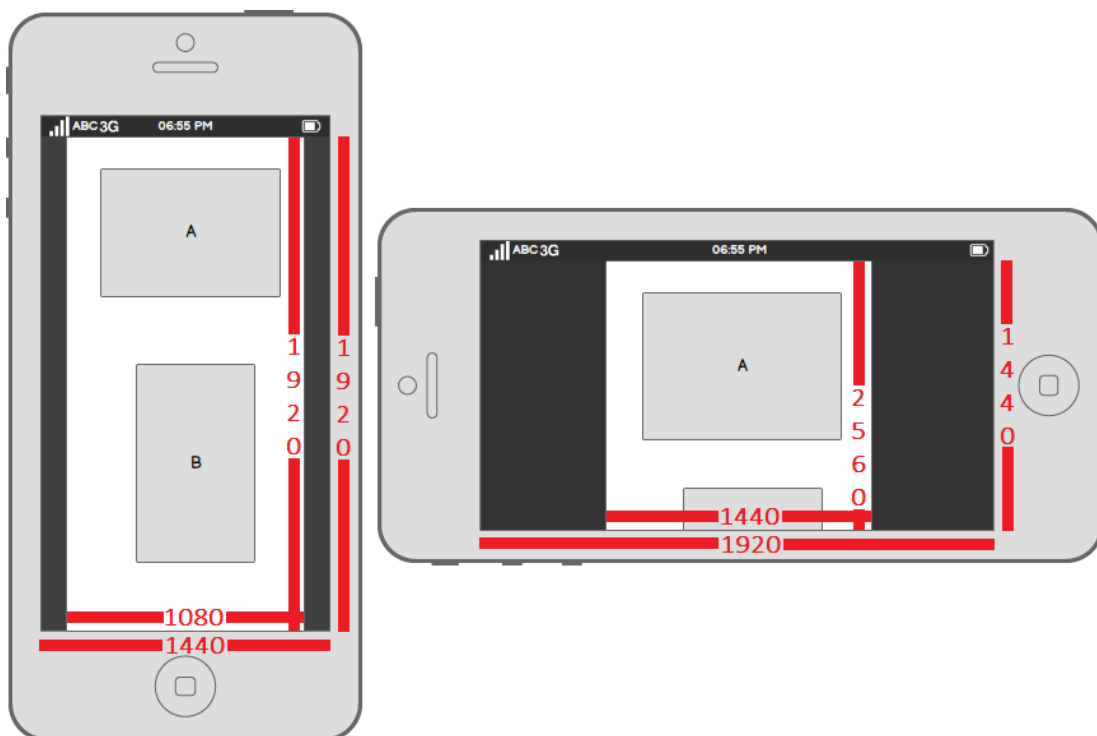


Abbildung 4.20: Darstellung einer Fernsteuerung in Portrait- und Landscape-Modus auf dem Mobilgerät.

#### 4.1.4 Fazit

Das Ziel war es, eine geeignete Architektur zu finden, damit das System realisiert werden kann. So wurden hierfür mehrere Softwarearchitekturen aus dem Bereich verteilte Systeme vorgestellt, die für das System in Frage kämen. Hierbei ergab sich, dass eine ereignisgesteu-

erte Architektur für das System am besten geeignet ist, da diese das Einbinden von neuen Agenten in das System deutlich einfacher gestaltet.

Des Weiteren stellte sich heraus, Ausgehen der Anforderungen aus Kapitel 2, dass für die Realisierung ein Fat Client am Idealsten ist, da der Client neben einer eigenen Oberfläche sowie einer eigenen Logik auch über eine eigene Persistenz verfügen soll, um Fernsteuerungen nicht nur auf dem Server, sondern auch auf dem Mobilgerät zu Speichern.

Im nächsten Schritt wurde eine mögliche auf der Architektur basierende Komponentenarchitektur des zu realisierenden Systems vorgestellt und erläutert. So wurden hier wichtige Entscheidungen über die Frameworks, die genutzt werden sollen, getroffen. Hier ergab sich das Nutzen vom „JavaFX-Framework“ als sinnvoll, da so falls nötig die Oberfläche aus Kapitel 3 als Web-Applikation genutzt werden kann.

Für den Aufbau einer Fernsteuerung wurde entschieden, dass diese nur aus Profilen bestehen soll, um Redundanzen zu vermeiden und so eine Fernsteuerung kompakter zu gestalten. Somit ist die eigentliche Fernsteuerung neben den Profilen auch ein Profil. Durch die vorgestellte Berechnung wiederum ist es möglich, auf unterschiedlichen Mobilgeräten unterschiedliche Fernsteuerungen darzustellen und zu nutzen, obwohl diese über den GUI-Builder für eine andere Auflösung erstellt wurden.

Durch den Aspekt der Individualisierbarkeit ist eine Fernsteuerung tragfähig als persönliche Fernsteuerung. Jedoch ist diese durch diesen Aspekt nicht wirklich austauschbar, da eine Fernsteuerung nach persönlichen Vorlieben sowie persönlicher Erfahrungen gestaltet wird. So kann dies dazu führen, dass die persönliche Note die mit eingebracht wird, von anderen Benutzern nicht geteilt wird.

## 4.2 Kommunikation

Für die Kommunikation soll die von Eichler (2014) entwickelte Middleware (Abbildung 4.21), die auf der Theorie von Multiagentensystemen basiert, verwendet werden. Laut Wooldridge (2002) gibt es keine allgemein anerkannte Definition eines Agenten. Doch ist man sich einig, dass ein Agent selbstständig (autonom) sein muss, darüber hinaus wiederum gibt es wenig Einigung. In der Middleware übernehmen Agenten die Rolle von „Smart-Objects“. So können „Smart-Objects“ als Agenten betrachtet werden, die es zu bedienen gilt.

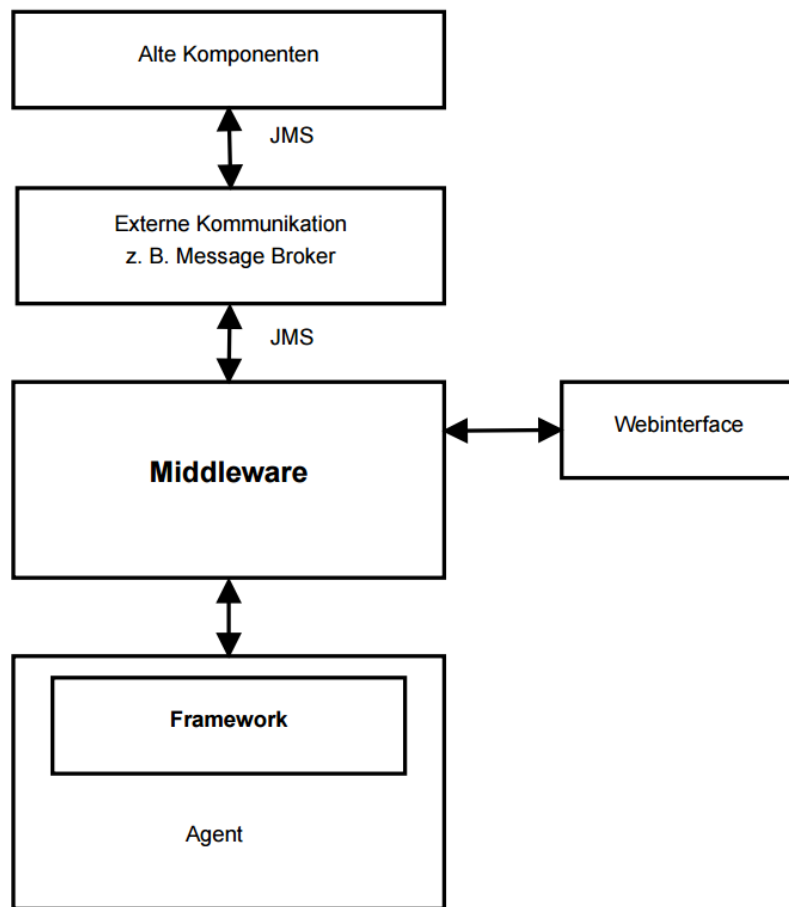


Abbildung 4.21: Blackbox-Ansicht der Middleware (Eichler (2014), S.40)

Die entwickelte Middleware bietet die Möglichkeit der Gruppenkommunikation, dadurch können Agenten Gruppen abonnieren und Nachrichten (Strings oder Objekte) über diese Gruppen veröffentlichen und empfangen. Die Übermittlung von Nachricht über das Netz erfolgt als JSON-Format. Für die Serialisierung und Deserialisierung der Nachrichten in ein JSON-Format müssen beide Agenten die Schnittstellenbibliothek importiert haben. Dadurch ermöglicht die Middleware eine heterogene Kommunikation zwischen unterschiedlichen Technologien.

#### 4.2.1 Nachrichten

In diesem Teil des Kapitels werden die relevantesten Nachrichten des Controller-Agenten exemplarisch genauer erläutert, da diese weitestgehend in den Sequenzdiagrammen im nachfolgenden Kapitel verwendet werden.

Hierfür wurde entschieden diese in Scala umzusetzen, da der Implementierungsaufwand von Schnittstellen auf diese Weise sehr gering ausfällt. So können Nachrichten durch Case-Klassen definiert werden (vgl. Eichler (2014), S.42). Dafür werden die Angaben des Objekt-namens und der Parameter mit Typinformation angegeben.

Die Buchstaben „R“ und „A“ vor den Nachrichten stehen für „Request“ und „Answer“ und dienen der Verdeutlichung der versendeten oder als Antwort empfangenen Nachrichten. Der „id“-Parameter in den Nachrichten dient der eindeutigen Zielbestimmung, an den die Antwortnachricht („Answer“-Nachricht) gesendet werden soll. So soll sich dieser Parameter in jeder Nachricht im System befinden, um falls nötig Antwortnachrichten an ein bestimmtes Ziel senden zu können. Sollte wiederum keine ID angegeben sein (Leerer String), hat dies zur Folge, dass die Antwortnachricht an alle gesendet werden soll, die auf die Gruppe hören. Im Falle des Controller-Agenten kann die ID, die ID des Mobilgerätes (Name des angemeldeten Benutzers) oder des Servers sein. Die Nachrichten des Controller-Agenten sehen wie folgt aus:

Die Nachricht „SaveController“ soll dem Speichern von Fernsteuerungen dienen. Hierfür soll die Fernsteuerung übergeben werden. Die Antwortnachricht beinhaltet den Namen der gespeicherten Fernsteuerung, um darzustellen welche Fernsteuerung gespeichert wurde.

```
R. SaveController(id: String, controller: Controller)
```

```
A. SavedController(name: String)
```

Damit mehrere Fernsteuerungen auf einmal gespeichert werden können, ohne die „SaveController“-Nachricht einzeln für jede Fernsteuerung zu senden, wurde entschieden, dies über eine einzelne Nachricht zu tätigen. So soll die Nachricht „SaveAllControllers“ dies machen. Hierfür soll der Nachricht alle zu speichernden Fernsteuerungen in einer Liste überreicht werden. Die Antwortnachricht beinhaltet dementsprechend eine Liste der Namen der gespeicherten Fernsteuerungen.

```
R. SaveAllControllers(id: String, controllers: List[Controller])
```

```
A. SavedAllControllers(names: List[String])
```

Mit der Nachricht „DeleteController“ sollen Fernsteuerungen bzw. Profile gelöscht werden. So soll über die Nachricht angegeben werden können, welches Profil in der Fernsteuerung gelöscht werden soll. Eine Fernsteuerung soll erst dann gelöscht werden können, wenn diese keine Profile mehr beinhaltet. Zudem soll über die Nachricht der Name des Autors, also des Erstellers des Profils, mit angegeben werden, um so ein Profil eindeutig identifizieren zu können, da unterschiedliche Benutzer ein Profil der selben Fernsteuerung erstellt



haben könnten, die wiederum denselben Namen trägt. Die Antwortnachricht beinhaltet wie in der Nachricht „SavedControllers“ den Namen der Fernsteuerung.

```
R. DeleteController(id: String, controllerName: String, profileName: String, author: String)
```

```
A. DeletedController(name: String)
```

Die Nachricht „GetControllers“ soll dem Abrufen der Fernsteuerungsnamen dienen. So soll über diese Nachricht die Namen der Fernsteuerungen sowie deren Autoren abgerufen werden.

```
R. GetControllers(id: String)
```

```
A. Controllers(controllerNames: List[ControllerName])
```

Der Typ „ControllerName“ hat hierbei folgenden Aufbau und beinhaltet die eben erwähnten Informationen, die relevant für die „GetControllers“-Nachricht sind.

```
• ControllerName(name: String, author: String)
```

Über die „GetController“ Nachricht wiederum soll dann die eigentliche Fernsteuerung abgerufen werden. Hierfür soll der Name der Fernsteuerung sowie des Erstellers angegeben werden. Die Antwortnachricht stellt die Fernsteuerung dar (vgl. 4.1.2 Aufbau einer Fernsteuerung). Diese Fernsteuerung soll nur die Profile beinhalten die vom angegebenen Autor erstellt wurden.

```
R. GetController(id: String, name: String, author: String)
```

```
A. Controller(name: String, author: String, profiles: List[Profile])
```

Die „ControllerException“-Nachricht soll dazu dienen Fehlermeldungen, die vom Controller-Agent kommen, zu übertragen. Diese Nachricht ist eine Antwortnachricht und soll im Fehlerfall gesendet werden, anstelle der richtigen Antwortnachricht.

```
A. ControllerException(reason: String)
```

### 4.2.2 Fehlerfall

Für den Fall, dass der Client bei einer Aktion, die eine Antwort vom Server benötigt aber diese nicht bekommt, weil der Server abgestürzt oder aus anderen Gründen nicht mehr erreichbar ist, soll ein Timer auf der Client-Seite gestartet werden vor dem Abschicken einer Nachricht, welche wenn der Timer abläuft eine Fehlermeldung anzeigt, die besagt, dass der Server nicht reagiert.

Für den Fall wiederum, dass der Client abgestürzt, soll alle 10 Sekunden ein Ping vom Client an den Server gesendet werden (es muss geprüft werden, ob der angegebene Intervall zu kurz oder zu lang ist sowie der Akkuverbrauch beobachtet werden). Der Server wiederum soll einen Timer für jeden angemeldeten Benutzer verwalten und diesen zurücksetzen, sobald die Ping-Nachricht empfangen wird. Der Timer soll auf 60 Sekunden gesetzt werden und runterzählen. Da Nachrichten verloren oder verzögert ankommen können, sollten nach eigenem Ermessen 60 Sekunden ausreichen. So können sechs Nachrichten innerhalb einer Minute vom Client an den Server gesendet werden. Dass alle sechs Nachrichten wiederum nicht verloren gehen können, kann nicht garantiert werden. Nach Ablauf der 60 Sekunden soll der Server den Benutzer automatisch abmelden. Auf eine Antwortnachricht vom Server soll verzichtet werden, um den Nachrichten Austausch gering zu halten.

Sollte der Benutzer versuchen, nach dem Absturz des Clients, sich wieder mit dem Server zu verbinden aber der Timer noch nicht abgelaufen ist und somit der Benutzer für das System weiterhin angemeldet ist, soll eine Nachricht in einem dafür eigenen Dialogfenster angezeigt werden, die besagt, dass der Benutzer bereits angemeldet ist und er es in einer bestimmten Zeit erneut versuchen soll. Um diese bestimmte Zeit jedoch zu ermitteln, soll bei der Login-Nachricht eine Antwort (Fehler-Nachricht) vom Server gesendet werden, die die Zeit des Timers beinhaltet.

### 4.2.3 Fazit

Ziel des Kommunikationskapitels war es die gewählte Middleware näher zu bringen, die für das zu realisierende System gewählt wurde. Diese wurde bereits im Rahmen eines Projektes genutzt, zudem bietet sie alle gewünschten Funktionalitäten. Da bereits Erfahrung im Umgang mit der Middleware vorhanden ist sowie die Tatsache, dass diese alle gewünschten Funktionalitäten besitzt, war eine Gegenüberstellung und somit der Vergleich zu anderen Middlewares nicht nötig.

Bei den Nachrichten, die über die Middleware gesendet werden sollen, wurde wiederum entschieden diese als Case-Klassen umzusetzen, da der Implementierungsaufwand in Scala geringer ist.

## 4.3 Sequenzdiagramme

In diesem Kapitel geht es darum, die Szenarien aus dem Unterkapitel 2.3, die für die Umsetzung des Systems essenziell sind, als Sequenzdiagramme genauer darzustellen und zu erläutern. Die Szenarien werden in Verbindung mit den erwähnten Komponenten und Nachrichten beschrieben.

### 4.3.1 Szenarien

#### Registrieren

Ein Benutzer möchte einen neuen Server auf dem Mobilgerät hinzufügen, über den er sich dann registrieren möchte.

Der Name des angemeldeten Benutzers kann hier nicht als ID angegeben werden, da dieser gerade im Begriff ist sich zu registrieren. Es wäre jedoch möglich, den Benutzernamen, der während der Registrierung angegeben wird, als ID zu nutzen, dies hätte jedoch den Nachteil, dass zu diesem Zeitpunkt noch nicht feststeht, ob der Name vergeben ist. Dies hätte zur Folge, dass die Antwortnachrichten während der Registrierung von zwei Personen empfangen werden würden. Damit dies nicht geschieht, wurde überlegt die IMEI-Nummer des Mobilgerätes zu verwenden. Die IMEI steht für „International Mobile Equipment Identity“ und ist eine 15-stellige Seriennummer, die weltweit ein Mobilgerät eindeutig identifiziert.

Die IMEI-Nummer soll jedoch nicht permanent genutzt werden, hierfür gibt es drei Gründe. Der erste ist, die Nummer lässt sich mit entsprechender Software ändern, so könnte ein böswilliger Nutzer sich als jemand anderes ausgeben.

Der zweite Grund ist, dass durch das Nutzen des Benutzernamens Update-Nachrichten sich einfacher realisieren lassen. Update-Nachrichten sind Antwortnachrichten mit dem Unterschied, dass diese gesendet werden, obwohl vom Empfänger keine Anfragenachricht (Request) gesendet wurde. Eine solche Update-Nachricht wäre z.B. das Ändern einer Berechtigung.

Der ausschlaggebende Grund für die Entscheidung, dass der Benutzername anstelle der IMEI-Nummer genutzt werden soll, ist, dass ein Benutzer ein anderes Mobilgerät nutzen könnte, als das, das er bei der Registrierung genutzt hat, so würde sich auch seine IMEI Nummer ändern, sein Benutzername aber nicht. So müsste die IMEI Nummer beim Anmelden immer mit angegeben werden. Daher ist der Aufwand zu hoch, um diese Nummer zu pflegen und aktuell zu halten. In Abbildung 4.22 ist das Szenario „Registrieren“ als Sequenzdiagramm zu sehen.

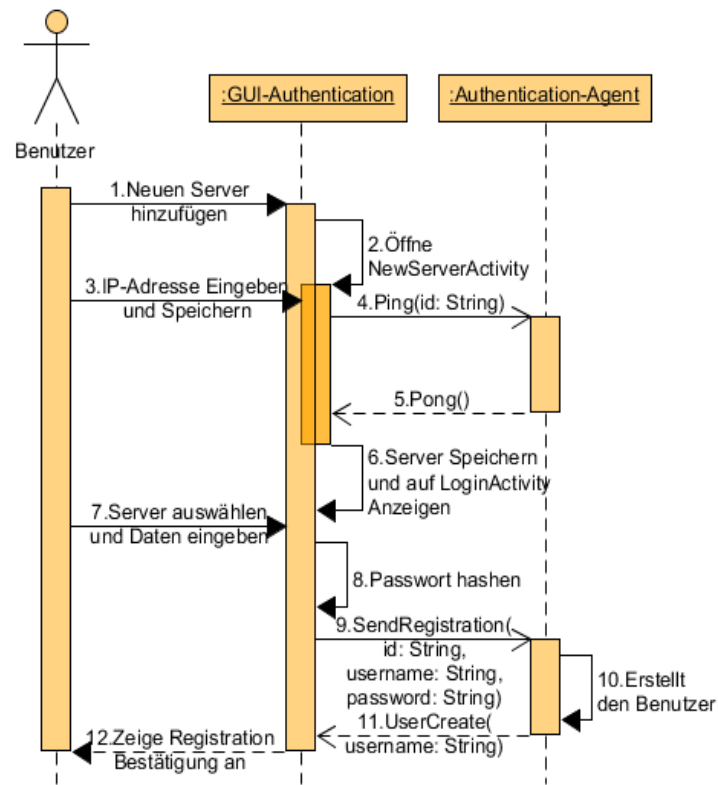


Abbildung 4.22: Szenario „Registrieren“

## Anmelden

Das Sequenzdiagramm in [Abbildung 4.23](#) stellt das Szenario „Anmelden“ dar. Ein Benutzer möchte sich auf einem Server über das Mobilgerät anmelden.

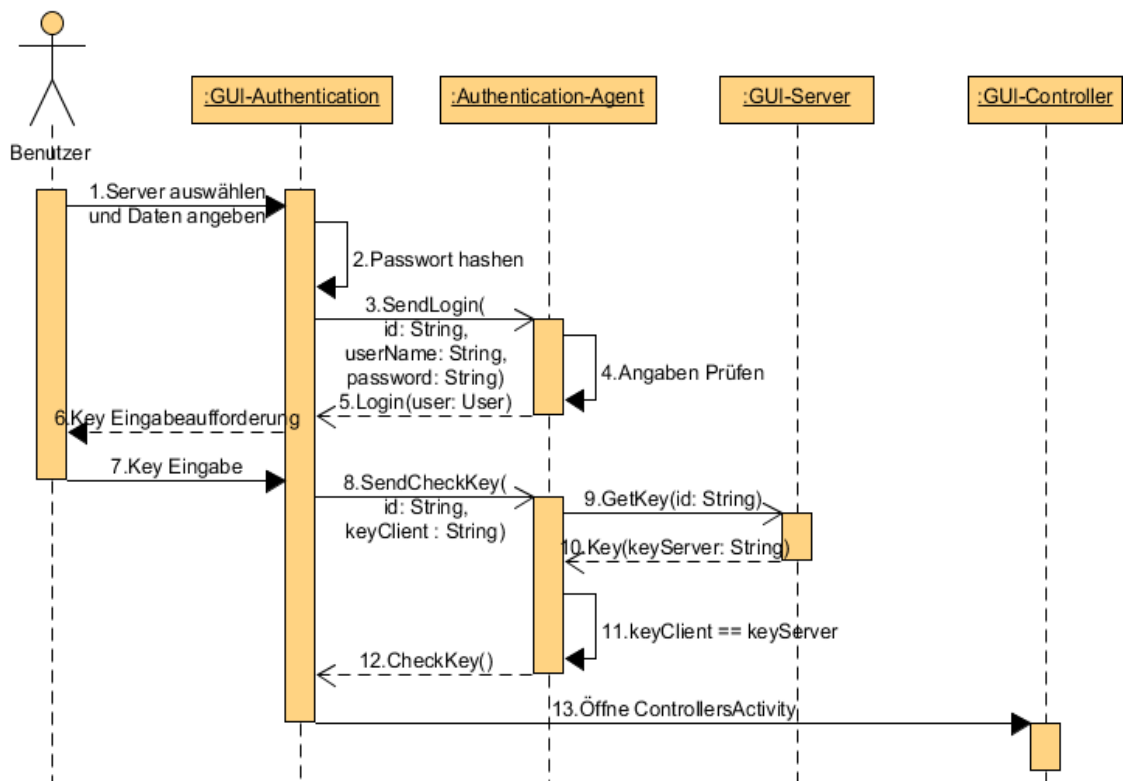


Abbildung 4.23: Szenario „Anmelden“

Der Typ „User“ soll den Namen des Benutzers, dessen Zustand, ob er geblockt ist oder nicht sowie dessen aktuelle Berechtigung beinhalten.

- `User(name: String, blocked: Boolean, permission: String)`

Die Anmeldung über die Serveroberfläche erfolgt bis Schritt 5 identisch. Die Key-Abfrage, die mit Schritt 6 beginnt, wurde weggelassen, da diese keinen Sinn macht und ansonsten erneut von der Serveroberfläche abgetippt werden müsste.

### Benutzerrechte Ändern

Abbildung 4.24 stellt das Szenario „Benutzerrechte Ändern“ als Sequenzdiagramm dar. Ein Admin möchte die Berechtigung eines Benutzers über die Admin Oberfläche ändern.

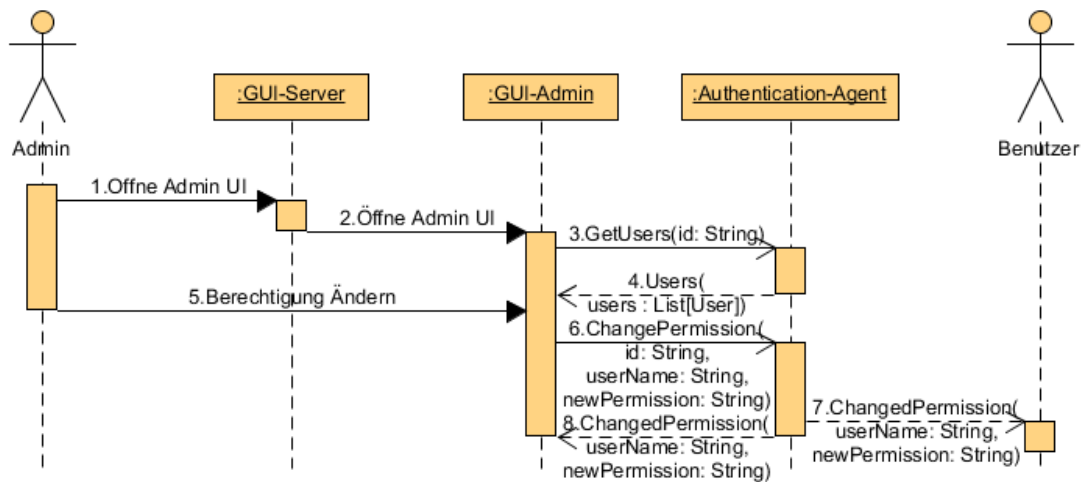


Abbildung 4.24: Szenario „Benutzerrechte Ändern“

Der Schritt 7 stellt eine Update-Nachricht dar, die an das Mobilgerät gesendet werden soll, auf dem der Benutzer angemeldet ist, damit die Änderung auch auf dem Mobilgerät übernommen werden kann. Sollte der Benutzer nicht angemeldet sein, um diese Nachricht zu empfangen, so stellt dies kein Problem dar, da beim Anmeldevorgang die Berechtigung mitgesendet wird.

### Agent Einbinden

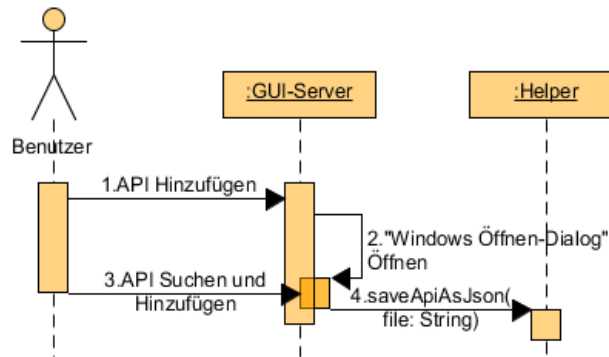


Abbildung 4.25: Szenario „Agent Einbinden“

In Abbildung 4.25 wird das Szenario „Agent Einbinden“ als Sequenzdiagramm dargestellt. Ein Benutzer möchte einen neuen Agenten nutzen und fügt hierfür dessen API über die Serveroberfläche in das System ein.

Fernsteuerung Erstellen

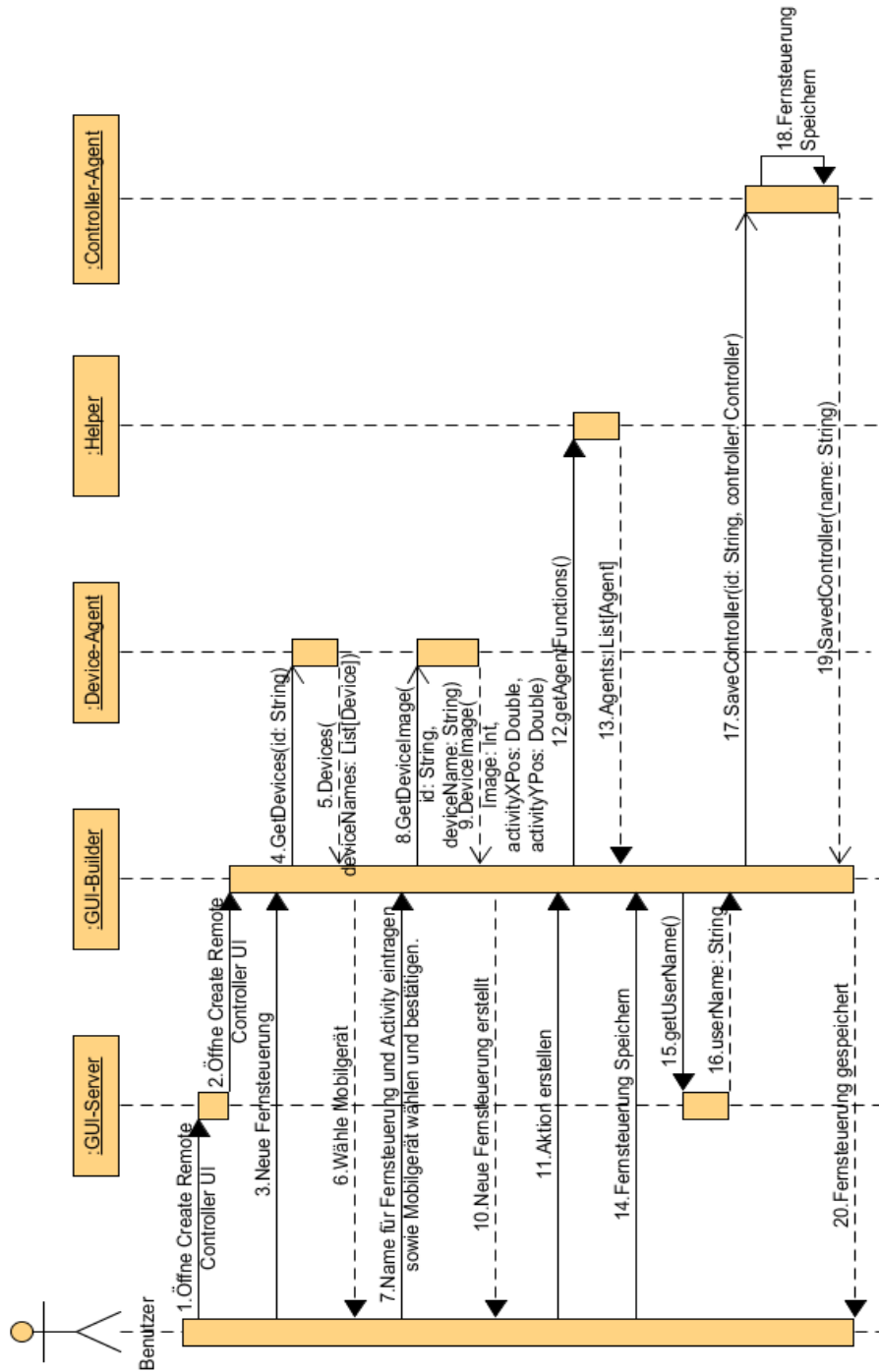


Abbildung 4.26: Szenario „Fernsteuerung Erstellen“

Auf dem Sequenzdiagramm das in Abbildung 4.26 dargestellt wird, wird das Szenario „Fernsteuerung Erstellen“ abgebildet. Ein Benutzer möchte sich über den GUI-Builder eine Fernsteuerung erstellen.

Der Typ „Device“ soll den Namen des Mobilgerätes sowie dessen Display Maße beinhalten.

```
• Device(name: String, width: Int, height: Int)
```

„DeviceImage“ beinhaltet neben dem Bild Informationen, wo die Activity dargestellt werden soll. So soll die x- und y-Position den Startpunkt angeben. Ein Endpunkt wird nicht benötigt, da vom Startpunkt aus die Höhe und Breite des Displays aus dem „Device“-Typ aufgerechnet werden soll.

Die beiden Schritte 4 und 8 sollen voneinander getrennt durchgeführt werden. Eine Nachricht zu versenden, die alle Informationen in einer Liste liefert, würde unnötige Informationen beinhalten, da der Benutzer nur ein Mobilgerät wählen kann aber zu allen Mobilgeräten die Bilder hat. Ein Problem hierbei wäre die Übertragung der Nachricht, da diese durch die Größe eine längere Zeit benötigen würde durch die Größe.

Der Typ „Agent“ soll den Namen der API sowie eine Liste aller Methoden aus der API beinhalten. „Method“ wiederum soll den Namen der Methode und eine Liste der Parameter beinhalten. „Param“ soll den Namen sowie den Typ des Parameters beinhalten. Diese Informationen in den Typen sind relevant, da diese dazu genutzt werden sollen, Elementen Aktionen zuweisen zu können.

```
• Agent(apiName: String, methods: List[Method])
```

```
• Method(name: String, params: List[Param])
```

```
• Param(name: String, type: String)
```

Der Benutzername soll separat abgefragt werden, wie in Schritt 15 zu sehen ist. Dies soll so gemacht werden, da es sein kann, dass während der Erstellung der Fernsteuerung kein Benutzer angemeldet ist. Wenn wiederum beim Öffnen des GUI-Builders der Benutzername mitgeliefert werden würde, ohne diesen separat nochmal abzufragen, könnte die erstellte Fernsteuerung nicht auf dem Server gespeichert werden, da nicht der Autornamen angegeben werden kann.



## Fernsteuerung Verwenden

Das Szenario „Fernsteuerung Verwenden“ wird in Abbildung 4.27 dargestellt. Ein Benutzer möchte über das Mobilgerät eine Fernsteuerung abrufen und diese nutzen.

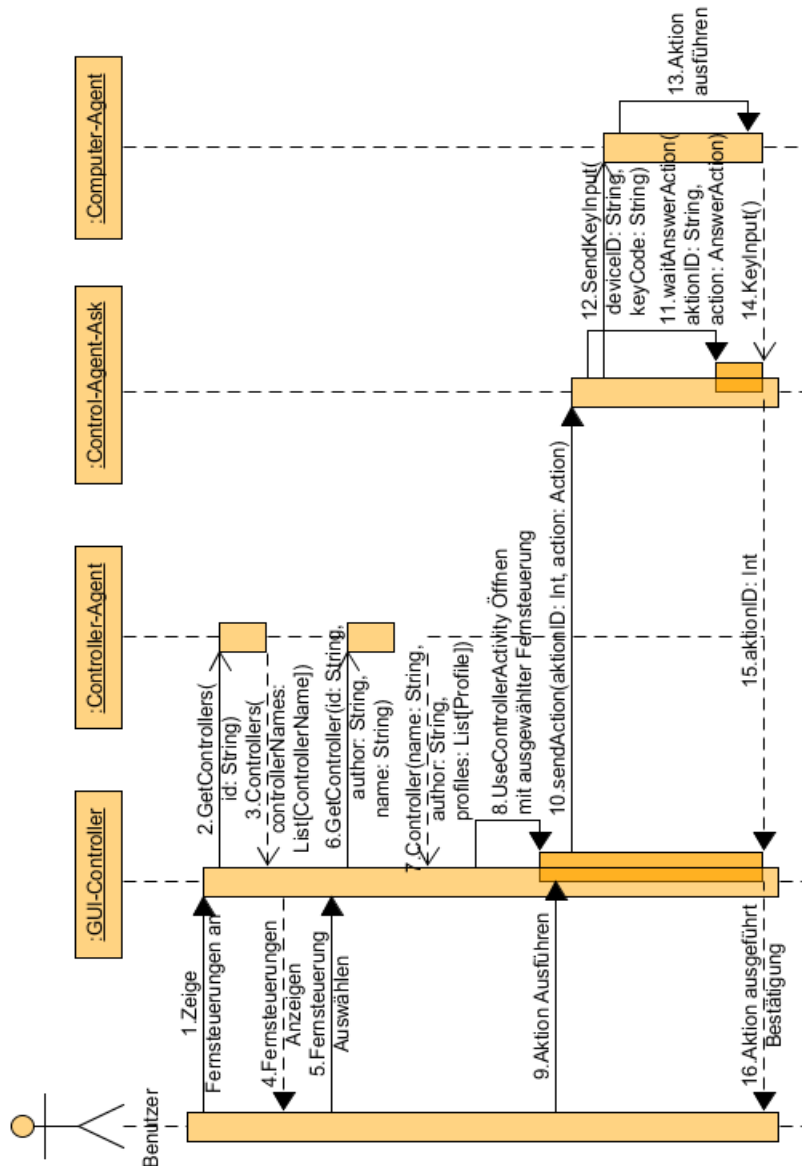


Abbildung 4.27: Szenario „Fernsteuerung Verwenden“

Das Abrufen der Fernsteuerungen soll aus Performancegründen in zwei Schritten erfolgen. In 2. ist zu sehen, dass die Fernsteuerungen abgerufen werden. Als Antwort wiederum kommt eine Liste mit dem Typen „ControllerName“ zurück, diese beinhaltet die Namen der

Fernsteuerungen sowie die dazugehörigen Namen der Autoren, die sich auf dem Controller-Agent befinden. Im zweiten Schritt (6.) wird die eigentliche Fernsteuerung anhand des Namens und des Autors abgerufen. Die Entscheidung, dies wie beschrieben umzusetzen wurde auf Grund der Tatsache getroffen, dass in frühen Tests eine Nachricht für das Abrufen der Fernsteuerungen genutzt wurde. Dies hatte zur Folge, dass sobald sich eine gewisse Menge an Fernsteuerungen auf dem Agenten befanden, die Nachricht die diese Fernsteuerungen beinhaltet dementsprechend länger braucht, um übertragen zu werden.

Das Abrufen der Fernsteuerungen soll auch wie beschrieben über den GUI-Builder in zwei Schritten ablaufen. So sollen im Fernsteuerungsstrukturbereich nur die Namen der Fernsteuerungen aufgelistet werden und sobald auf einen Namen gedrückt wird, soll die Fernsteuerung abgerufen werden.

### 4.3.2 Fazit

Wie in den Sequenzdiagrammen zu sehen ist, wurde auf einen geringen Nachrichtenaustausch zwischen Client und Agenten geachtet, da dies relevant für den Akkuverbrauch des Mobilgerätes ist. So sollen abgerufene Fernsteuerungen alle Profile beinhalten, da gewollt ist, dass der Benutzer zwischen den Profilen wechseln können soll, wie in Unterkapitel 3.2.5 Abbildung 3.44c dargestellt wurde. Alternativ könnte es so implementiert werden, dass die Profile einzeln abgerufen werden, was dann wiederum zum Nachteil hätte, das der Akkuverbrauch steigen würde, da für den Wechsel eines Profils immer eine Nachricht gesendet sowie die Antwortnachricht empfangen werden müsste.

In einem früheren Projekt wurde ein Sensor-Agent<sup>18</sup> entwickelt und genutzt. Dieser hat alle Sensorinformationen vom Client abgerufen und an den Server weitergeleitet. Dies hatte zur Folge, dass der Akku des Mobilgeräts schnell verbraucht war. So wurde entschieden, dass die Akkulaufzeit für das hier zu realisierende System eine große Rolle spielt.

Die Akkulaufzeit ist auch ein Grund, wieso der GUI-Builder nicht als Agent realisiert werden soll, da schon die kleinste Änderung an einem Element, wie z.B. die Position oder die Größe, zur Folge hätte, dass eine Nachricht an den GUI-Builder-Agenten gesendet werden würde. Der Vorteil wiederum eines solchen Agenten wäre, dass er es ermöglicht, über das Mobilgerät Fernsteuerungen zu erstellen.

---

<sup>18</sup> Sensor-Agent – (Sensor-Agent, vgl.)

## 4.4 Fazit

Ziel des Design-Kapitels war es, aus den im Analyse-Kapitel definierten Szenarien und Anforderungen ein Design zu entwickeln, damit das System realisiert werden kann. Zu diesem Zweck wurde eine mögliche Architektur entwickelt sowie die Kommunikation und Sequenzdiagramme, die auf dieser Architektur basieren, vorgestellt. Dabei ergab sich, dass sich eine ereignisgesteuerte Architektur am besten für die Realisierung des Systems eignet. Denn diese ermöglicht das einfache Einbinden von neuen Agenten in das System ohne merklichen Aufwand.

Was die Wahl des Kommunikationsmittels und somit der Middleware betrifft. Es wurde hierfür die Middleware von Eichler (2014) entschieden, da bereits Erfahrung im Umgang mit dieser besteht. Andererseits musste festgestellt werden, dass Nachrichten über die Middleware an Gruppen gesendet werden und nicht an ein direktes Ziel. Zwar kann anhand dieses Befundes darauf geschlossen werden, dass nur Gruppen angesprochen werden können, jedoch ist es möglich, Nachrichten an ein bestimmtes Ziel zu senden, unter folgenden Bedingungen:

- Eine Kennung (Benutzername) muss der Gruppenbezeichnung angehängt werden.
- Die Kennung muss eindeutig sein.
- Die Kennung muss sich in jeder Anfragenachricht (Request) befinden.
- Die Antwortnachricht (Answer) muss an die Gruppenbezeichnung + Kennung gesendet werden.
- Der Agent der die Anfragenachricht gesendet hat muss auf die Gruppe Gruppenbezeichnung + Kennung hören, um die Antwortnachricht zu erhalten.

Abschließend kann gesagt werden, dass das hier realisierte Design alle gestellten Anforderungen aus Kapitel 2 an das zu realisierende System erfüllt.

# 5 Evaluation

Um die Realisierbarkeit des vorgestellten Systems zu beweisen, kann ich auf eine frühere Implementierung zurückgreifen, die im Rahmen eines Projektes entwickelt wurde<sup>19</sup>. Diese erfüllt die wesentlichen Anforderungen aus Unterkapitel 2.4 sowie die Vorgaben aus Kapitel 4. Die prototypische Implementierung ist im GitLab unter der Bezeichnung „MFC“<sup>20</sup> (Multi-Function-Controller) zu finden und wurde von zwei Kommilitonen sowie mir entwickelt.

## 5.1 Multi-Function-Controller

Es wurden bei der Entwicklung der prototypischen Anwendung die Frameworks benutzt, die im Design-Teil genannt worden sind. Somit wurde für die Entwicklung der Serversoberfläche und der Oberfläche des GUI-Builders JavaFX und für die App scaloid verwendet. Für die Speicherung der Fernsteuerungen wurde Sorm und H2 auf der Server-Seite und Play-JSON<sup>21</sup> auf der Client-Seite genutzt. Für den Nachrichtenaustausch zwischen den Agenten und den Ask-Agenten wurde das Aktor-Framework genutzt.

Abschließend wurde ein Prototyp einer Fernsteuerung über den GUI-Builder erstellt und auf dem Mobilgerät abgerufen, der die Machbarkeit der Idee demonstrieren soll.

---

<sup>19</sup> Ich möchte mich hiermit bei [Holland \(2015\)](#) und [Oster \(2015\)](#) für ihre Vorarbeit bedanken.

<sup>20</sup> MFC – ([MFC](#), vgl.)

<sup>21</sup> Play-JSON wird zum Umwandeln der Fernsteuerungen in ein JSON-Format und wieder zurück verwendet.

### 5.1.1 Erfüllung der Anforderungen

In diesem Abschnitt erfolgt die Evaluierung des entwickelten MFC-Systems. Die Bewertung erfolgt nach den festgelegten Kriterien aus dem Analyse-Kapitel. Dabei wird auf die einzelnen Anforderungen, die an das zu entwickelnde System gestellt wurden, eingegangen.

#### Funktionale Anforderungen

- **Registrieren:** Diese Anforderung wurde erfüllt. Dem Benutzer ist es möglich, sich zu registrieren.
- **Anmelden:** Diese Anforderung wurde erfüllt. Dem Benutzer ist es möglich, sich anzumelden.
- **Benutzerrechte Ändern:** Diese Anforderung wurde erfüllt. Einem Admin ist es möglich, die Berechtigungen eines Benutzers zu ändern.
- **Agent Einbinden:** Diese Anforderung wurde nicht erfüllt aber stellt auch kein Risiko dar, da durch die Verwendung des Computer-Agenten gezeigt wurde, dass Agenten genutzt werden können bei der Erstellung von Fernsteuerungen.
- **Fernsteuerung Erstellen:** Diese Anforderung wurde zum Teil umgesetzt. Dem Benutzer ist es möglich, Fernsteuerungen zu erstellen. Ihm ist es jedoch nicht möglich, Profile zu erstellen. Da Fernsteuerungen und Profile im Grunde dasselbe sind mit dem Unterschied, dass ein Profil nur vom Ersteller genutzt werden kann, stellt dies kein Risiko in der Implementierung da und wurde daher weggelassen.
- **Fernsteuerung Speichern:** Diese Anforderung wurde erfüllt. Dem Benutzer ist es möglich, Fernsteuerungen auf dem Server als auch auf dem Mobilgerät zu speichern.
- **Fernsteuerung Abrufen:** Diese Anforderung wurde erfüllt. Dem Benutzer ist es möglich, Fernsteuerungen vom Server abzurufen.
- **Fernsteuerung Bearbeiten:** Diese Anforderung wurde erfüllt. Dem Benutzer ist es möglich, erstellte Fernsteuerungen über den GUI-Builder anzupassen oder zu verändern.
- **Fernsteuerung Verwenden:** Diese Anforderung wurde erfüllt. Dem Benutzer ist es möglich, Fernsteuerungen auf dem Mobilgerät zu nutzen.

- **Fernsteuerung Löschen:** Diese Anforderung wurde erfüllt. Dem Benutzer ist es möglich, Fernsteuerungen zu löschen, sowohl die auf dem Server als auch die auf dem Mobilgerät.

### Nicht-Funktionale Anforderungen

- **Mobilität:** Die Mobilität des Systems ist durch die getroffenen Designentscheidungen gewährleistet. Durch die Implementierung des Clients als App für das Mobilgerät wurde der Aspekt der Mobilität berücksichtigt.
- **Sicherheit:** Hinsichtlich der Sicherheit des Systems wurde darauf geachtet, dass nicht autorisierte Nutzer sich keinen Zutritt in das System verschaffen können. Hierfür wird jeder neu registrierte Nutzer vorerst geblockt. Dieser muss durch den Admin entblockt werden, um sich auf dem System anmelden zu können. Der Nachteil hierbei ist, sollte das Mobilgerät abhandenkommen, kann sich ein unautorisierter Nutzer, der das Passwort sowie den Benutzernamen kennt, Zugang ins System verschaffen. Im neuen System soll daher das beschriebene Key-Verfahren zu Authentifizierung genutzt werden.
- **Laufzeit:** Eine lange Laufzeit des Systems ist durch die getroffenen Designentscheidungen gewährleistet, da darauf geachtet wurde, den Nachrichtenaustausch zwischen Server und Client auf das nötigste zu beschränken, um den Akku des Mobilgerätes zu schonen. So werden keine Nachrichten automatisch ausgetauscht, außer der Ping-Nachricht, die vom Client an den Server gesendet wird, um mitzuteilen, dass dieser noch da ist. Um den Nachrichtenaustausch minimal zu halten, wurde hierbei auf eine Pong-Nachricht also die Antwortnachricht verzichtet.
- **Übersichtlichkeit:** Aus Zeitgründen, die die Implementierung der grafischen Oberfläche in Anspruch genommen hätte, wurden die Windows- und Android-Styleguides nicht berücksichtigt. Die Anpassung des Prototyps auf die Styleguides stellt keine Risiken dar. Der dazu nötige Aufwand jedoch würde den zeitlichen Rahmen sprengen.
- **Erweiterbarkeit:** Die Erweiterbarkeit des Systems ist nur zum Teil gewährleistet. So ist es möglich, neue APIs von Agenten ins System einzubinden aber nicht ohne Programmierkenntnisse vorauszusetzen, da diese nicht wie beschrieben in ein JSON-Format umgewandelt werden, um die darin enthaltenen Funktionen auszulesen.

### 5.1.2 Funktionsweise des Prototyps

Das Ziel hier ist es über den Prototyp eine Fernsteuerung zu erstellen und diese zu nutzen. Hierzu werden die notwendigen Schritte erläutert, die getätigt werden müssen, um dieses Ziel zu erreichen<sup>22</sup>:

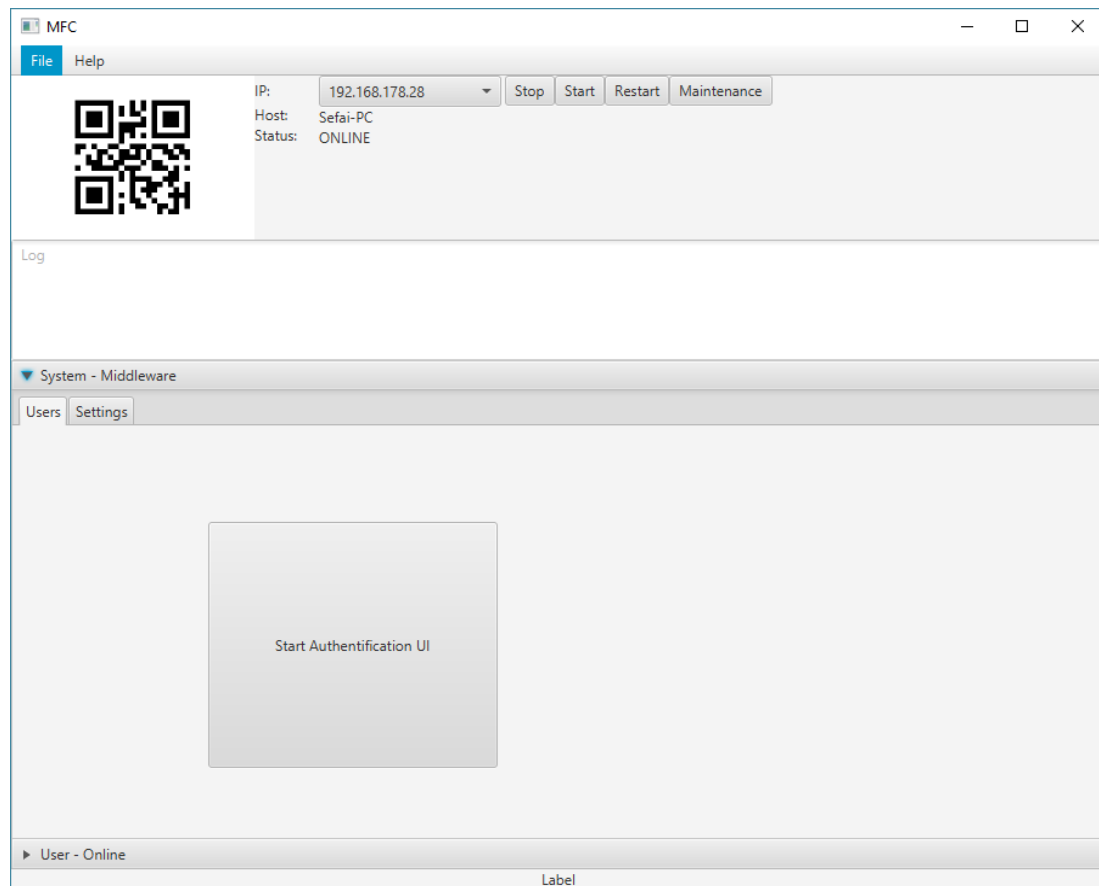


Abbildung 5.1: „Server“-Oberfläche des MFC-Systems auf dem der Button „Start Authentication UI“ angezeigt wird.

Nachdem der Server gestartet und die in Abbildung 5.1 zu sehende Oberfläche dem Benutzer angezeigt wird, kann dieser sich über die App, die in Abbildung 5.2 angezeigt wird, auf dem Server registrieren.

---

<sup>22</sup> Die in dem Prototy verwendete Bezeichnung „Profil“ entspricht nicht der in der Arbeit erwähnten Profile. Diese entsprechen eher einer Fernsteuerung.

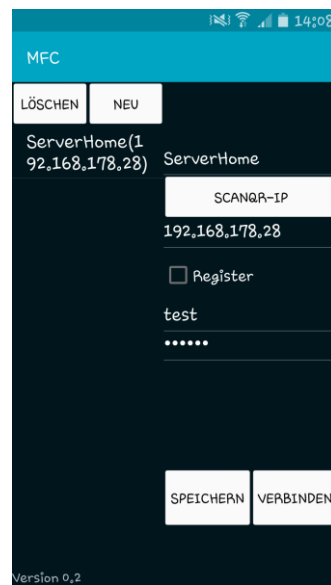


Abbildung 5.2: „Login“-Oberfläche des MFC-Systems auf dem Daten eingetragen sind sowie ein Server gespeichert wurde.

Damit der Benutzer sich registrieren kann, müssen die Felder ausgefüllt sowie der Haken bei „Register“ gesetzt sein. Als nächstes soll auf den Button „VERBINDEN“ gedrückt werden, der eine Registrierungsnachricht an den Authentication-Agenten sendet. Nun wird auf der Serveroberfläche auf den „Start Authentication UI“-Button gedrückt, der die in Abbildung 5.3 zu sehende Oberfläche öffnet. Über diese kann nun der neue Benutzer durch den Admin entblockt werden.

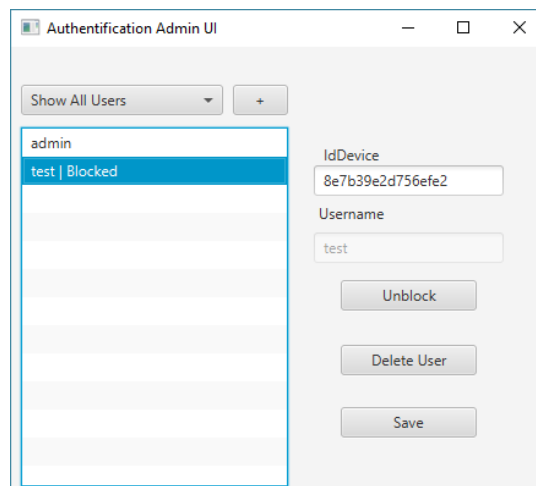


Abbildung 5.3: „Authentication UI“-Oberfläche des MFC-Systems auf dem ein geblockter Benutzer angezeigt wird.



Um den Benutzer zu entblocken wird dieser aus der Liste links gewählt und auf den Button „Unblock“ geklickt, was zur Folge hat, dass neben dem Namen des Benutzers nicht mehr „Blocked“ steht. Nun da der Benutzer nicht mehr geblockt ist, kann er sich auf dem System anmelden. Hierfür wird der Haken bei „Register“ auf der App entfernt und wieder auf „VERBINDEN“ gedrückt, dieses mal wird eine Login-Nachricht an den Authentication-Agenten gesendet. Dieser prüft, ob der Benutzer registriert und nicht geblockt ist und sendet dann eine Bestätigung zurück, was zur Folge hat, dass sich die in Abbildung 5.4 dargestellte Oberfläche auf dem Mobilgerät öffnet. Über diese können nun Fernsteuerungen gewählt werden. Da im Moment keine aufgelistet werden, muss eine erstellt werden.



Abbildung 5.4: „Controllers“-Oberfläche des MFC-Systems ohne Fernsteuerungen.

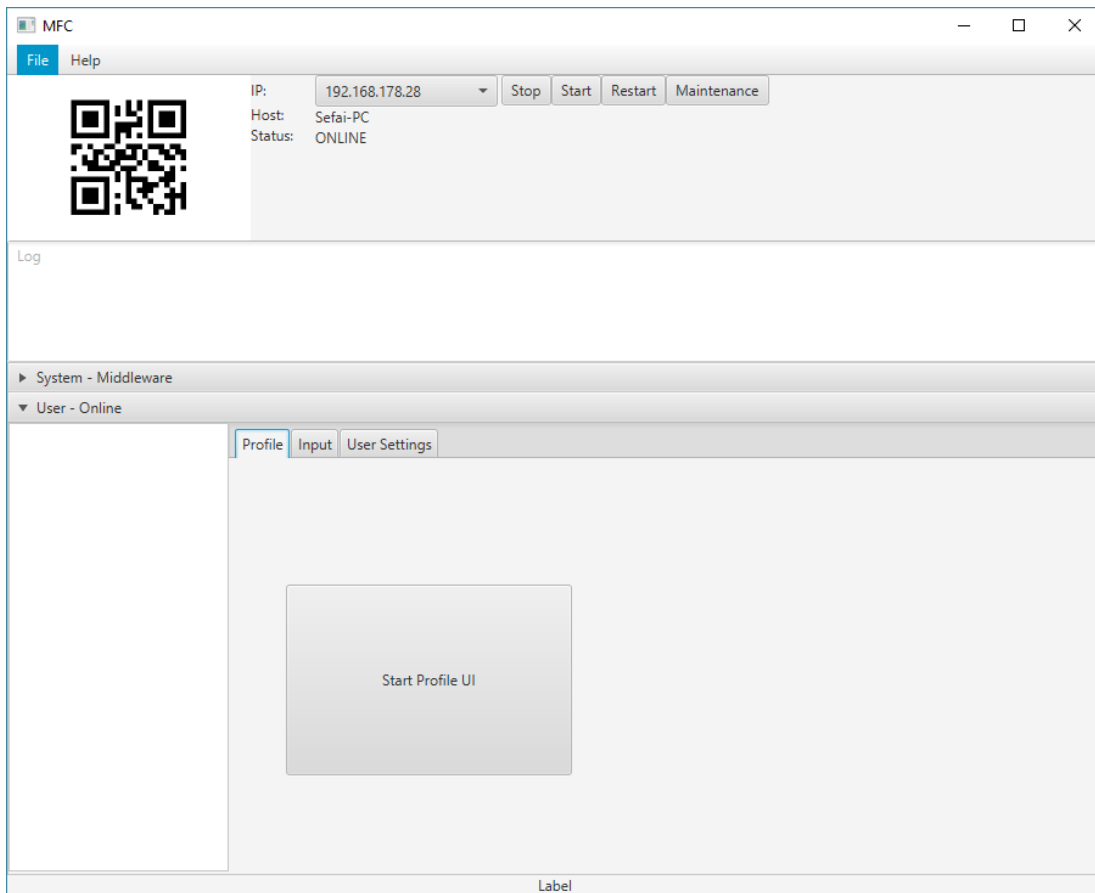


Abbildung 5.5: „Server“-Oberfläche des MFC-Systems auf dem der Button „Start Profile UI“ angezeigt wird.

Hierfür wird auf der Serveroberfläche in Abbildung 5.5 auf den Button „Start Profile UI“ geklickt. Dieser öffnet das in Abbildung 5.6 zu sehende Fenster, über den nun eine Fernsteuerung erstellt werden kann.

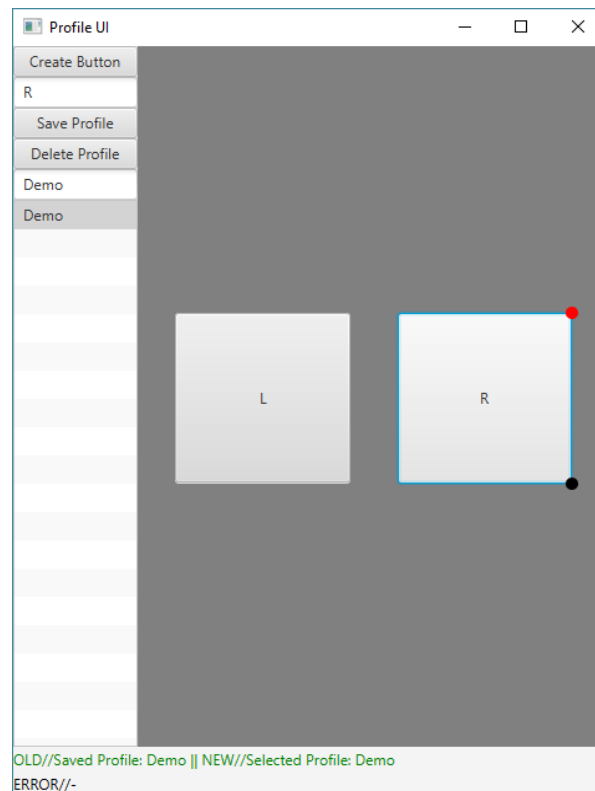


Abbildung 5.6: „Create Remote Controller“-Oberfläche des MFC-Systems auf dem die erstellte Fernsteuerung „Demo“ angezeigt wird.

Mit einem Klick auf den Button „Create Button“ wird auf der rechten grauen Hälfte ein Button erstellt, mit der Bezeichnung, die im darunter befindlichen Textfeld eingetragen wurde, in diesem Fall mit der Bezeichnung „L“. Erstellte Buttons lassen sich über den roten Kreis löschen und über den schwarzen in ihrer Größe ändern, sobald auf diesen geklickt wurde. Um einem Button eine Aktion zuzuweisen, wird auf diesen ein Doppelklick ausgeführt, was zur Folge hat, dass sich das Fenster, das in Abbildung 5.7 zu sehen ist, öffnet.

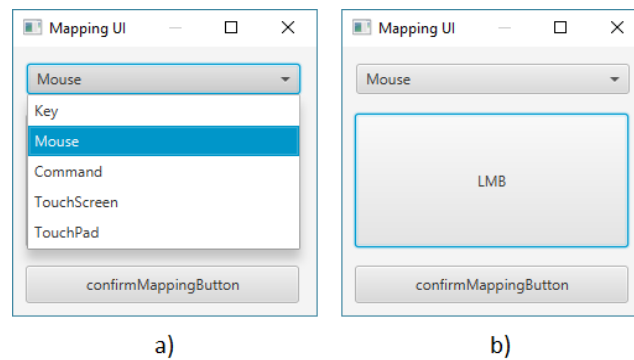


Abbildung 5.7: „Mapping UI“-Oberfläche des MFC-Systems auf der eine Aktion zugewiesen wird.

Über die Dropdown-Liste in a) wird eine Aktion ausgewählt. In dem Beispiel wurde Mouse gewählt, über den „Action Mappen“-Button in b) die Aktion „Left Mouse Button“ (LMB) zugewiesen und über den „confirmMappingButton“ bestätigt. Damit die erstellte Fernsteuerung nun genutzt werden kann, wird diese über den GUI-Builder gespeichert. Hierfür wird unter dem Button „Save Profile“ eine Bezeichnung eingetragen, die die Fernsteuerung erhalten soll und auf den „Save Profile“-Button gedrückt. Dieser sendet nun eine „SaveController“-Nachricht an den Controller-Agenten, auf dem die Fernsteuerung gespeichert wird. Sobald die Fernsteuerung erfolgreich gespeichert wurde, wird eine Bestätigungsnachricht gesendet. Beim Erhalt der Bestätigung wird die Fernsteuerung in der Liste in Abbildung 5.6 angezeigt. Nun kann die Fernsteuerung über die App abgerufen werden.

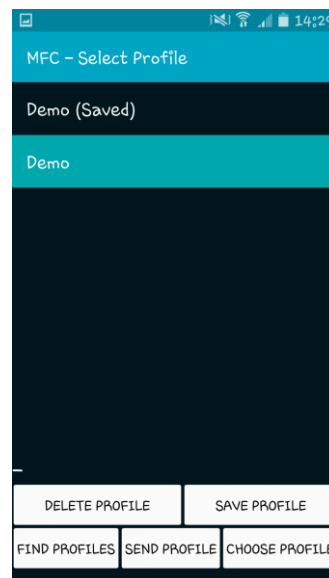


Abbildung 5.8: „Controllers“-Oberfläche des MFC-Systems mit Fernsteuerungen.

Hierfür wird auf den „FIND PROFILES“ Button in Abbildung 5.8 gedrückt. Dieser sendet eine „GetConrollers“-Nachricht an den Controller-Agenten, der wiederum sendet eine Liste mit

den Namen der gespeicherten Fernsteuerungen zurück. Diese werden auf der Oberfläche in der Liste aufgelistet. Fernsteuerungen, die auf dem Mobilgerät gespeichert werden, erhalten den Zusatz „Saved“. Als nächstes wird die Demo-Fernsteuerung aus der Liste gewählt und auf den Button „CHOOSE PROFILE“ gedrückt. Dieser sendet eine „GetController“-Nachricht mit dem Namen der ausgewählten Fernsteuerung an den Controller-Agenten, der wiederum sendet die vollständige Fernsteuerung zum Namen zurück. Daraufhin öffnet sich die in [Abbildung 5.9](#) zu sehende Oberfläche, die das Profil visuell darstellt. Sobald nun auf dieser Oberfläche auf einen Button gedrückt wird, wird die damit verbundene Aktion an den Computer-Agenten gesendet, wo die Aktion wiederum umgesetzt wird. Zudem bekommt der Benutzer ein visuelles Feedback, ob der Button gedrückt wurde, indem der gedrückte Button blau aufleuchtet.

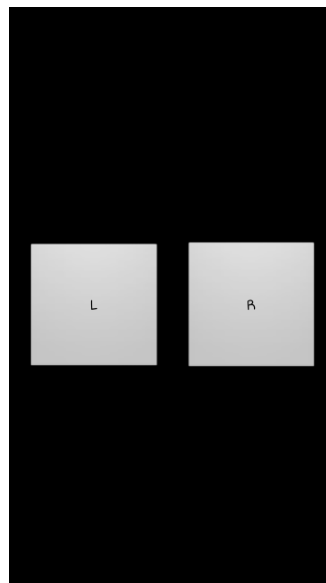


Abbildung 5.9: „Controller“-Oberfläche des MFC-Systems auf der die Fernsteuerung angezeigt wird.

### 5.1.3 Fazit

Mit dieser prototypischen Implementierung sind weitestgehend alle relevanten funktionalen und nicht funktionalen Anforderungen aus dem Unterkapitel [2.4](#) abgedeckt. So können über den GUI-Builder Fernsteuerungen erstellt und bearbeitet und über das Mobilgerät abgerufen und genutzt werden, diese Funktionalitäten wurden wie im Design-Kapitel beschrieben umgesetzt.

Weitere Elemente wie Slider und Toggle-Button wurden nicht implementiert, da gezeigt wurde, dass durch die Implementierung des Buttons als Element weitere Elemente kein Risiko darstellen und daher aus Zeitgründen weggelassen wurden.

## 5.2 Fazit

Das Ziel dieser Arbeit war die Entwicklung einer adaptierbaren Fernsteuerung, die es ermöglicht smarte Objekte zu bedienen. So wurden hierfür ein Android-Mobilgerät, ein Windows-Rechner, die Middleware von [Eichler \(2014\)](#) und der Computer-Agent von [Oster \(2015\)](#) verwendet, um dieses Ziel zu erreichen. Auch die Erkenntnisse aus dem im Projekt entwickelten Prototyp sind in diese Arbeit mit eingeflossen.

Mit der entwickelten Lösung lassen sich einfache Fernsteuerungen erstellen und nutzen, ohne Programmierkenntnisse vorauszusetzen. So ist es möglich, nicht nur Fernsteuerungen zur Bedienung von „Smart-Objects“ zu erstellen, sondern auch individualisierte Fernsteuerungen zur Steuerung von Spielen. So kann ein NES-Controller (Abbildung 5.10) als Vorlage dienen.



Abbildung 5.10: NES-Controller (Ist das Gamepad einer „Nintendo Entertainment System“ Konsole.)

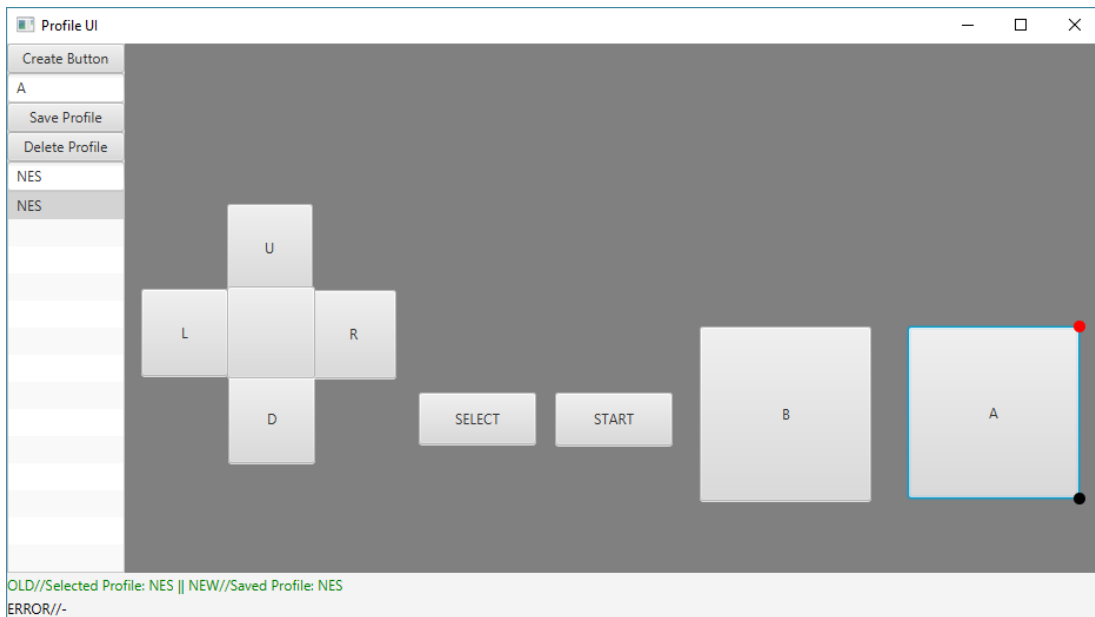


Abbildung 5.11: Erstellung eines NES-Controllers über den GUI-Builder.

Über den GUI-Bilder (Abbildung 5.11) kann dieser dann umgesetzt und über das Mobilgerät (Abbildung 5.12) wiederum genutzt werden.

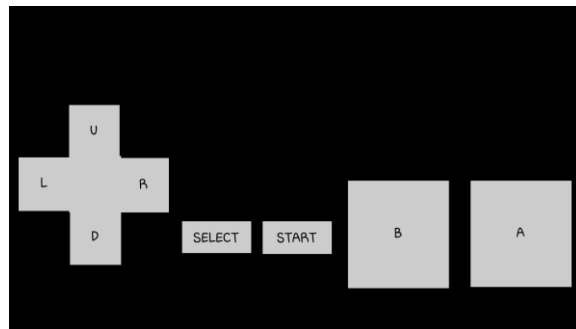


Abbildung 5.12: Anzeigen des erstellten NES-Controllers über das Mobilgerät.

### 5.3 Erweiterung des MFC-Systems

Durch die Evaluation des MFC-Systems wurde entschieden, dass folgende Erweiterungen für dieses System noch erdenklich wären.

Das System könnte dahingehend erweitert werden, dass die Positionsangabe des Mobilgeräts (GPS) dafür genutzt wird, die in der Nähe befindlichen „Smart-Objects“ zu lokalisieren. So könnten Fernsteuerungen der in der Nähe befindlichen „Smart-Objects“ angeboten

werden. Alternativ könnte eine Fernsteuerung direkt abgerufen und angezeigt werden, um so alle in der Nähe befindliche „Smart-Objects“ zu bedienen.

Auch im Bereich des persönlichen Nutzens könnte das System erweitert werden. So könnte ein Profil zur Authentifizierung auf einem Computer genutzt werden, indem der Fingerabdrucksensor eines Mobilgeräts genutzt wird. Hierfür müsste die Einbindung eines Sensor-Agenten erfolgen.

Das System könnte um eine Sharing-Funktion erweitert werden. So könnten Benutzern Fernsteuerungen von anderen Benutzern angeboten werden, die dieselben Interessen bei der Gestaltung der Fernsteuerung aufweisen. Des Weiteren könnten Benutzern vorgefertigte Muster zur Erstellung der Fernsteuerungen zur Auswahl stehen.

Im Bereich Akkuverbrauch wurden bereits Maßnahmen zur Energieeinsparung getroffen, wie beispielsweise, dass der GUI-Builder nicht als Agent realisiert werden soll, da schon für die kleinste Änderung an einer Fernsteuerung eine Nachricht gesendet werden müsste. Um weiter Energie zu sparen müsste der Fokus auf eine Akku sparende Programmierung gelegt werden. So wäre ein Ansatz weiße Flächen zu vermeiden, da ein Display aus drei LED's, den Grundfarben (Rot, Grün und Blau), besteht. Um einen Pixel weiß darzustellen müssten alle drei LED's leuchten.

Auch im Bereich Sicherheit müsste einiges getan werden, da im Moment Nachrichten unverschlüsselt ausgetauscht werden und dies das Abhören von Nachrichten und Replay-Angriffe ermöglicht. So wäre hier der Einsatz eines Protokolls aus dem Bereich IT-Sicherheit erdenklich, wie beispielsweise das Kerberos-Protokoll zur Authentifizierung.



# 6 Schluss

In diesem Kapitel erfolgt eine Zusammenfassung dieser Arbeit sowie ein Ausblick auf weitere interessante Fragestellungen.

## 6.1 Zusammenfassung

In dieser Arbeit wurde ein System entwickelt zur Erstellung von adaptierbaren Fernsteuerungen für smarte Umgebungen. Agenten stellen hierbei die „Smart-Objects“ dar, die es zu steuern gilt. Dahingehend wurde eine Möglichkeit geschaffen, weitere Agenten von „Smart-Objects“ während der Erstellung von Fernsteuerungen anzugeben. Die Erstellung einer Fernsteuerung erfolgt hierbei über eine grafische Oberfläche auf einem Computer. Die Nutzung der Fernsteuerungen hingegen erfolgt über ein Mobilgerät.

In der Analyse (s. Kapitel 2) wurde das Ziel, dass für diese Arbeit gesetzt wurde, näher gebracht. So wurden hierfür ähnliche Projekte aufgezeigt, die sich wie diese Arbeit mit dem Themenbereich „Human–Computer Interaction“ beschäftigen. Diese wurden bei der Erläuterung der Zielsetzung mit eingebracht, um Ähnlichkeiten sowie Unterschiede zu dieser Arbeit aufzuzeigen. Anschließend wurde, anhand der Zielsetzung für das hier zu realisierende System, Szenarien mit unterschiedlichen Ausprägungen vorgestellt. Dazu gehören Szenarien wie das Erstellen und Nutzen einer Fernsteuerung sowie das Einbinden von neuen Agenten für die Erstellung. Aus diesen wurden wiederum funktionale und nicht-funktionale Anforderungen abgeleitet und formuliert, die das System erfüllen muss. So spielen die nicht-funktionalen Anforderungen, wie die Sicherheit und Laufzeit, bei der Entwicklung des Systems eine große Rolle.

Ausgehend von den Szenarien und Anforderungen wurden grafische Oberflächen entworfen und designt (s. Kapitel 3). Hierfür wurde entschieden die Windows Design Guidelines bei der Desktop-Anwendung und das Material Design von Google bei der Mobile-

Anwendung zu berücksichtigen. So wurden drei Oberflächen auf der Desktop-Seite entwickelt. Eine für die Middleware, damit diese gestartet und beendet werden kann, und eine um Fernsteuerungen zu erstellen sowie eine für den Administrator, um die Benutzer zu verwalten. Auf der Mobile-Seite wiederum entstanden mehrere Oberflächen bzw. Activities. So war hier die Activity zum Nutzen von Fernsteuerungen nicht als einzige relevant. Neben dieser Activity waren die Activities zum Verbinden mit der Middleware sowie die zum Abrufen der Fernsteuerungen mitrelevant.

Weiterhin ausgehend von den Szenarien und Anforderungen, wurde ein mögliches Design (s. Kapitel 4), das auf der ereignisgesteuerten Architektur basiert, entwickelt. So wurden hier die Komponenten des möglichen Designs für das zu realisierende System vorgestellt sowie die einzelnen Komponenten in ihrer Funktionsweise und ihrem Zuständigkeitsbereich erläutert. Anschließend wurde der Aufbau einer Fernsteuerung beschrieben sowie eine mögliche Form der Umwandlung von Fernsteuerungen auf dem Mobilgerät exemplarisch aufgezeigt, damit diese auf dem Mobilgerät, so wie sie erstellt wurden, dargestellt und somit genutzt werden können. Des Weiteren wurde im Design die Kommunikation erläutert. Hierbei wurden exemplarisch die Nachrichten des Controller-Agenten beschrieben sowie eine zu nutzende Middleware vorgestellt. Zum Abschluss des Kapitels wurden die Szenarien aus Kapitel zwei als Sequenzdiagramme mit der Einbindung der vorgestellten Komponenten und Nachrichten erläutert.

Die Evaluation (s. Kapitel 5) erfolgt über einen Prototyp. Hierbei wird aus Zeitgründen auf eine frühere Implementierung zurückgegriffen. Diese erfüllt alle relevanten Anforderungen aus Kapitel 2, die an das zu realisierende System gestellt wurden. Auch die verwendete Architektur entspricht der aus Kapitel vier erwähnten Architektur. Dies beweist daher die Machbarkeit eines solchen Systems. So wurde in einem simplen Szenario eine Fernsteuerung zur Bedienung eines Computers erstellt und anschließend auf dem Mobilgerät genutzt.

## 6.2 Ausblick

Zukünftig wäre der Einsatz eines solchen Tools im „Living Place Hamburg“ (von Luck u. a. (2010)) erdenklich. So könnten Studenten mit Hilfe dieses Tools Fernsteuerungen für ihre Agenten erstellen und diese im Living Place bereitstellen. Hierbei stellt sich die Frage, ob das Anbieten von allen Fernsteuerungen oder das der „Smart-Objekts“, die sich im Raum befinden in dem der Benutzer sich aufhält, sinnvoller ist.

Da bei der Implementierung einer App der Akkuverbrauch des Mobilgeräts im Vordergrund liegt, stellt sich die Frage, ob eine bestimmte Programmiersprache sich besser dafür eignet energiesparender zu programmieren.

Zu untersuchen wäre auch, ob eine alternative Erstellungsmöglichkeit für Fernsteuerungen geeigneter wäre. Beispielsweise eine Fernsteuerung zu erstellen, indem die Elemente gezeichnet, anstelle per Drag & Drop positioniert werden.

Aufgrund der Entscheidung nur den Individualisierungsaspekt bei der Gestaltung einer Fernsteuerung zu berücksichtigen und den Aspekt der Kontextualisierung<sup>23</sup> außen vor zu lassen, würden sich neue Anforderungen, bei der Einbeziehung der Kontextualisierung, ergeben. Außerdem müssen vorhandene Anforderungen erweitert oder angepasst werden. Hierbei wäre eine mögliche Ergänzung das Einbinden von Benutzerprofilen (keine Fernsteuerungsprofile). Benutzerprofile würden Informationen über den Benutzer darstellen, wie Handicaps, die wiederum Profilen zugewiesen werden können. Dies könnte im Bereich „Ambiente Assisted Living“ kurz AAL interessant werden, da sich dabei das angesteuerte „Smart-Object“ auf die Bedürfnisse des Nutzers einstellen müsste, auch gegebenenfalls müsste sich die Fernsteuerung den Bedürfnissen anpassen.

Die Idee der Fernsteuerung, die alles bedienen kann (Everything as a Service (XaaS)), kann jedoch noch weiter gesponnen werden, da bereits Computer allgegenwärtig sind, welches als „Ubiquitous Computing“ von Weiser (1991) schon in den neunziger Jahren beschrieben wurde. So könnten Fernsteuerungen nicht nur die heimische Wohnung bedienen, sondern auch alles außerhalb dieser vier Wände. Hierfür müssten jedoch öffentliche Einrichtungen ihre Agenten und somit deren APIs zur Verfügung stellen. Ein Beispiel hierfür wären Einkaufszentren. Der Kunde könnte so seinen Einkauf tätigen, ohne den Laden zu betreten. Aus Sicht der Läden wiederum könnten diese den Kunden Angebote oder Werbung, die sie interessieren könnte, schicken. Auch im Fußgänger- oder Straßenbereich wäre der Einsatz erdenklich. So zum Beispiel bei Verkehrsampeln, hier könnte eine Fernsteuerung dazu genutzt werden, der Ampel Bescheid zu geben, dass man über die Straße gehen oder weiterfahren möchte.

---

<sup>23</sup> Kontextualisierung beschreibt das Verhalten von Anwendungsprogrammen, die Informationen über ihre Umgebung benutzen, um ihr Verhalten darauf abzustimmen.

# Literaturverzeichnis

[Bernardos u. a. 2011] Bernardos, Ana M. ; Casar, José R. ; Cano, Jesús ; Bergesio, Luca: Enhancing interaction with smart objects through mobile devices. 2011. – URL <http://dl.acm.org/citation.cfm?id=2069170&CFID=572295281&CFTOKEN=64356439>. – ISBN: 978-1-4503-0901-1

[Bornemann 2011] Bornemann, Sven Boris: Android-basierte Smart Home Interaktion am Beispiel einer Gegensprechanlage. 2011. S. 37-42. – URL <http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/bornemann.pdf>.

[Budde u. a. 2013] Budde, Matthias ; Berning, Matthias ; Baumgärtner, Christopher ; Kinn, Florian ; Kopf, Timo ; Ochs, Sven ; Reiche, Frederik ; Riedel, Till ; Beigl, Michael: Point & control -- interaction in smart environments: you only click twice. 2013. – URL <http://dl.acm.org/citation.cfm?id=2494184&CFID=572295281&CFTOKEN=64356439>. – ISBN: 978-1-4503-2215-7

[DAI-Labor] DAI-Labor: DAI-Labor. – URL <http://www.dai-labor.de/>. – Zugriffsdatum: 11.12.2015

[Dunkel u. a. 2008] Dunkel, Jürgen ; Enerhart, Andreas ; Fischer, Stefan ; Kleiner, Carsten ; Koschel, Arne: Systemarchitekturen für Verteilte Anwendungen. Hanser Fachbuchverlag, 2008. – URL <http://amazon.com/o/ASIN/3446413219/>. – ISBN 9783446413214

[Dustdar u. a. 2003] Dustdar, Schahram ; Gall, Harald ; Hauswirth, Manfred: Software-Architekturen für Verteilte Systeme: Prinzipien, Bausteine und Standardarchitekturen für moderne Software (Xpert.press) (German Edition). 1. Springer, 7 2003. – URL <http://amazon.com/o/ASIN/3540430881/>. – ISBN 9783540430889

[Eichler 2014] Eichler, Tobias: Agentenbasierte Middleware zur Entwicklerunterstützung in einem Smart-Home-Labor. 2014. – URL <https://kataloge.uni-hamburg.de/DB=2/SET=4/TTL=1/CMD?ACT=SRCHA&IKT=1016&SRT=YOP&TRM=Agentenbasierte+Middleware+zur+Entwicklerunterst%FCtzung+in+einem+Smart-Home-Labor>. – PPN: 799921572

[ERGO-Online] ERGO-Online: DIN EN ISO 9241-110. – URL [http://www.ergo-online.de/site.aspx?url=html/software/grundlagen\\_der\\_software\\_ergon/grundsaeetze\\_der\\_dialoggestalt.htm](http://www.ergo-online.de/site.aspx?url=html/software/grundlagen_der_software_ergon/grundsaeetze_der_dialoggestalt.htm). – Zugriffsdatum: 18.07.2016

[Google] Google: Material design. – URL <https://www.google.com/design/spec/material-design/introduction.html#>. – Zugriffsdatum: 10.11.2015

[H2Database] H2Database: H2 Database Engine. – URL <http://www.h2database.com/html/main.html>. – Zugriffsdatum: 03.12.2015

[Holland 2015] Holland, Alexander: Multi-Functional-Controller mit Hilfe von Android, Scala und Akka, HAW Hamburg, WP-Smart-Environment. 2015. – URL [http://devsupport.informatik.haw-hamburg.de/projekte/wp15/wp-content/uploads/2015/08/HAW\\_WP\\_SWE\\_SS15\\_MFC\\_Alexander\\_Holland\\_v2.pdf](http://devsupport.informatik.haw-hamburg.de/projekte/wp15/wp-content/uploads/2015/08/HAW_WP_SWE_SS15_MFC_Alexander_Holland_v2.pdf).

[Javabeginners] Javabeginners: Was ist und wozu dient FXML?. – URL <http://javabeginners.de/Frameworks/JavaFX/FXML.php>. – Zugriffsdatum: 21.02.2016

[Joos 2015] Joos, Thomas: Android: Benutzerverwaltung für mehrere User verwenden. 2015. – URL <http://workshop.tecchannel.de/a/benutzerverwaltung-fuer-mehrere-user-verwenden,3277617>. – Zugriffsdatum: 11.12.2015

[Kolbaja 2012] Kolbaja, Viktor: Generischer Smart Home Controller auf Android™ OS. 2012. – URL <https://kataloge.uni-hamburg.de/DB=2/SET=2/TTL=1/CMD?ACT=SRCHA&IKT=1016&SRT=YOP&TRM=Generischer+Smart+Home+Controller+auf+Android%99+OS>. – PPN: 729293688

[Kostovarov 2013] Kostovarov, Dmitry: Deploying JavaFX Applications. 2013. – URL [http://docs.oracle.com/javafx/2/deployment/deployment\\_toolkit.htm](http://docs.oracle.com/javafx/2/deployment/deployment_toolkit.htm). – Release 2.2.40 E20472-11

[López u. a. 2011] López, Tomás S. ; Ranasinghe, Damith C. ; Patkai, Bela ; Mcfarlane, Duncan: Taxonomy, Technology and Applications of Smart Objects. In: Information Systems

Frontiers 13 (2011), April, Nr. 2, S. 281–300. – URL <http://link.springer.com/article/10.1007%2Fs10796-009-9218-4>. – ISSN 1387-3326

[MFC] Holland, Alexander ; Oster, Alexander ; Sari, Sefai: MFC. 2015. – URL <http://devsupportgit.informatik.haw-hamburg.de/Alexander.Holland/mfc/tree/master>. – Zugriffsdatum: 17.10.2015

[Microsoft] Microsoft: Windows Design Guidelines. – URL [https://msdn.microsoft.com/en-us/library/windows/desktop/ff728831\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ff728831(v=vs.85).aspx). – Zugriffsdatum: 02.01.2016

[Oster 2015] Oster, Alexander: Multi-Functional-Controller mit Hilfe von Android, Scala und Akka, HAW Hamburg, WP-Smart-Environment. 2015. – URL [http://devsupport.informatik.haw-hamburg.de/projekte/wp15/wp-content/uploads/2015/08/Alexander\\_-\\_Oster\\_14.8.2015\\_HAW\\_WP\\_SWE\\_Ausarbeitung\\_v2.pdf](http://devsupport.informatik.haw-hamburg.de/projekte/wp15/wp-content/uploads/2015/08/Alexander_-_Oster_14.8.2015_HAW_WP_SWE_Ausarbeitung_v2.pdf).

[Password-Depot] Password-Depot: Brute-Force-Angriff. – URL [http://www.password-depot.de/know-how/brute\\_force\\_angriffe.htm](http://www.password-depot.de/know-how/brute_force_angriffe.htm). – Zugriffsdatum: 09.01.2016

[Playframework] Playframework: Documentation. – URL <https://www.playframework.com/documentation/2.2.x/ScalaJson>. – Zugriffsdatum: 24.03.2016

[Qrcodegenerator] Holland, Alexander: Qrcodegenerator. 2014. – URL <http://devsupportgit.informatik.haw-hamburg.de/po1415/qrcodegenerator/tree/master>. – Zugriffsdatum: 16.09.2015

[Rixecker 2014] Rixecker, Kim: Cheat-Sheets für App-Entwickler: Alle wichtigen Maße und Angaben für iOS, Android und Windows Phone. 2014. – URL <http://t3n.de/news/cheat-sheet-fuer-app-entwickler-556817/>. – Zugriffsdatum: 15.03.2016

[Roscher u. a. 2009] Roscher, Dirk ; Blumendorf, Marco ; Albayrak, Sahin: A meta user interface to control multimodal interaction in smart environments. 2009. – URL <http://dl.acm.org/citation.cfm?id=1502725&CFID=572295281&CFTOKEN=64356439>. – ISBN: 978-1-60558-168-2

[Sari 2015] Sari, Sefai: Multi-Functional-Controller mit Hilfe von Android, Scala und Akka, HAW Hamburg, WP-Smart-Environment. 2015. – URL [http://devsupport.informatik.haw-hamburg.de/projekte/wp15/wp-content/uploads/2015/08/HAW\\_WP\\_SWE\\_SS15\\_MFC\\_Sefai\\_Sari\\_v2.pdf](http://devsupport.informatik.haw-hamburg.de/projekte/wp15/wp-content/uploads/2015/08/HAW_WP_SWE_SS15_MFC_Sefai_Sari_v2.pdf).

[Scaloid] Lee, Sung-Ho: Scaloid. – URL <https://github.com/pocorall/scaloid>. – Zugriffsdatum: 17.10.2015

[Schartel 2014] Schartel, Christian: Android 5.0 Lollipop: Nutzerprofile kurz erklärt. 2014. – URL <http://www.cnet.de/88139515/android-5-0-lollipop-nutzerprofile-kurz-erklaert/>. – Zugriffsdatum: 11.12.2015

[Schill und Springer 2007] Schill, Alexander ; Springer, Thomas: Verteilte Systeme: Grundlagen und Basistechnologien (eXamen.press) (German Edition). 1. Springer, 3 2007. – URL <http://amazon.com/o/ASIN/3540205683/>. – ISBN 9783540205685

[Schwartzte u. a. 2010] Schwartzte, Veit ; Blumendorf, Marco ; Albayrak, Sahin: Adjustable context adaptations for user interfaces at runtime. 2010. – URL <http://dl.acm.org/citation.cfm?id=1843051&CFID=572295281&CFTOKEN=64356439>. – ISBN: 978-1-4503-0076-6

[Sensor-Agent] Holland, Alexander ; Sari, Sefai: Sensor-Agent. 2014. – URL <http://devsupportgit.informatik.haw-hamburg.de/po1415/sensor-agent>. – Zugriffsdatum: 06.08.2015

[Sorm-Framework] Sorm-Framework: Sorm. – URL <http://sorm-framework.org/>. – Zugriffsdatum: 03.12.2015

[Statista: Betriebssysteme] Statista: Marktanteile der führenden Betriebssystemversionen weltweit von Januar 2009 bis Mai 2016. – URL <http://de.statista.com/statistik/daten/studie/157902/umfrage/marktanteil-der-genutzten-betriebssysteme-weltweit-seit-2009/>. – Zugriffsdatum: 15.03.2016

[Statista: Smartphone Betriebssysteme] Statista: Prognose zu den Marktanteilen der Betriebssysteme am Absatz vom Smartphones weltweit in den Jahren 2016 und 2020. – URL <http://de.statista.com/statistik/daten/studie/182363/umfrage/prognostizierte-marktanteile-bei-smartphone-betriebssystemen/>. – Zugriffsdatum: 15.03.2016

[Stückler 2014] Stückler, Moritz: Material Design: Google enthüllt neue Designsprache für Android, Chrome und das Web. 2014. – URL <http://t3n.de/news/material-design-google-enthueellt-553560/>. – Zugriffsdatum: 16.03.2016

[Tanenbaum u. van Steen 2006] Tanenbaum, Andrew S. ; Steen, Maarten van: Distributed Systems: Principles and Paradigms. 2nd Edition. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 2006. S. 41-43. <https://www.pearsonhighered.com/program/Tanenbaum-Distributed-Systems-Principles-and-Paradigms-2nd-Edition/PGM104647.html>. – ISBN 0132392275

[von Luck u. a. 2010] Luck, Prof. Dr. K. von ; Klemke, Prof. Dr. G. ; Gregor, Sebastian ; Rahimi, Mohammad A. ; Vogt, Matthias: Living Place Hamburg - A place for concepts of IT based modern living / Hamburg University of Applied Sciences. 2010. URL – [http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg\\_en.pdf](http://livingplace.informatik.haw-hamburg.de/content/LivingPlaceHamburg_en.pdf). – Forschungsbericht

[Weingarten u. a. 2010] Weingarten, Florian ; Blumendorf, Marco ; Albayrak, Sahin: Towards multimodal interaction in smart home environments: the home operating system. 2010. – URL <http://dl.acm.org/citation.cfm?id=1858255&CFID=572295281&CFTOKEN=64356439>. – ISBN: 978-1-4503-0103-9

[Weiser 1991] Weiser, Mark: The Computer for the 21st Century. 1991. – URL <http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html>. – Zugriffsdatum: 01.12.2015

[Wooldridge 2002] Wooldridge, Michael: Intelligent Agents: The Key Concepts. 2002. S. 5. – URL <http://dl.acm.org/citation.cfm?id=645699.665762>. – ISBN:3-540-43377-5



# Versicherung über Selbstständigkeit

*Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.*

*Hamburg, den* \_\_\_\_\_