HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

GEBERIT

Entwicklung einer Cross-Plattform ServiceApp für Sanitäranlagen

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2016

Autor(en): Pascal Marty & Noah Hendrikx
Betreuer: Prof. Dr. Farhad Mehta
Projektpartner: Geberit AG, Jona

Abstract

Die Geberit AG entwickelt hochleistungsfähige, intelligente Systeme und Produkte in der Sanitärtechnik. Darunter auch das neuste Dusch-WC AquaClean Mera, welches über eine Bluetooth-Schnittstelle verfügt. Mit der Geberit AquaClean ServiceApp für Android und iOS, entwickelt Geberit eine Applikation, welche es Servicetechnikern ermöglicht, Fehler auszulesen und ihnen anhand der Fehlermeldungen Vorschläge für Reparaturmassnahmen liefert. Ebenso ist es mit der App möglich Firmware Aktualisierungen des Dusch-WCs vorzunehmen.

Ziel dieser Arbeit war es, die Applikation auf Xamarin zu portieren und herauszufinden, wie viel Code mithilfe von Xamarin Forms zwischen der Android und iOS Version gemeinsam verwendet werden kann und wo es Sinn macht auf nativen Plattform Code zurück zu greifen. Zudem ist auch die Usability der bestehenden Applikation zu evaluieren und gegebenenfalls Verbesserungen zu erarbeiten.

Das Projekt wurde durch Verwendung von agilen Softwareentwicklungsmethoden durchgeführt. Die Benutzeroberfläche wurde durch Mockups unter Berücksichtigung der Material Design Richtlinien gestaltet.

Als Programmiersprache wurde C# eingesetzt. Die Geberit AG pflegt über ein Cloud-Backend, die Konten der Servicetechniker und die Applikationsdaten. Die Applikation kann jeweils beim Start die aktuellsten Daten beziehen, ohne dass eine neue Version installiert werden muss.

Der Funktionsumfang der Applikation besteht aus einem aktualisierbaren Fehlerursachen- und zugehörigem Massnahmen-Katalog. Diese können anhand von Fehlerbildern oder Fehlercodes der verbundenen Sanitäranlage ausgelesen werden. Zudem kann die aktuelle Firmware-Version der einzelnen Komponenten angezeigt und aktualisiert werden.

Durch eine grosse gemeinsame Codebasis, ist die Wartung und Erweiterung der Applikation nun einfacher.

Management Summary

Ausgangslage

Die Geberit AG entwickelt hochleistungsfähige, intelligente Systeme und Produkte in der Sanitärtechnik, darunter auch das neuste Dusch-WC AquaClean Mera. Die Geberit AG produziert und vertreibt diese weltweit. Die Installation und Wartung dieser Anlagen wird jeweils von zertifizierten Servicetechnikern vorgenommen.

Das Aquaclean Mera Dusch-WC verfügt über diverse elektronische Komponenten, welche gewartet und um neue Funktionen erweitert werden können. Um dies zu ermöglichen, ist das Dusch-WC mit einem Bluetooth-Modul ausgestattet, über welches Informationen kabellos ausgetauscht werden können. Das Auslesen allfällig vorhandener Fehler der einzelnen Komponenten sowie das Durchführen von Software Aktualisierungen wird somit ermöglicht.

Damit ein Servicetechniker die Wartung und Reparatur möglichst unkompliziert erledigen kann, hat die Geberit AG bereits eine Applikation entwickelt, welche ihm alle dafür relevanten Informationen zur Verfügung stellt. Somit kann eine Fehlerursache möglichst schnell gefunden und die dazu erforderlichen Massnahmen getroffen werden.

Die bestehende Applikation ist bereits für die meist verbreiteten mobilen Plattformen Android und iOS im Einsatz, jedoch entspricht die dafür verwendete Technologie nicht den Kernkompetenzen des Entwicklungsteams der Geberit AG. Infolgedessen soll die bestehende Applikation in eine für die Geberit AG passenderen Technologie portiert werden. Zudem soll das Design und die Benutzbarkeit überarbeitet werden.

Vorgehen

Die Projektdurchführung wurde in drei Phasen gegliedert. Die erste Phase umfasst die Analyse der Anforderungen, welche die neue Applikation erfüllen muss. Zudem wurde unter Berücksichtigung der bestehenden internen Arbeitsprozesse der Geberit AG untersucht, wie die kontinuierlich aktualisierten Daten in die Applikation eingepflegt werden. Zusätzlich wurden Design

Entwürfe angefertigt, um dem Kunden einen ersten Eindruck zu vermitteln, wie die Applikation aussehen würde und um sich auf ein Design zu einigen. Mithilfe dieser Informationen wurde dann ein erster Architektur-Prototyp entwickelt, welcher alle benötigten Technologien beansprucht.

In der nächsten Phase wurden die Anforderungen in Arbeitspakete unterteilt, welche in einem iterativen Prozess umgesetzt wurden. Dabei wurde stetiger Kontakt mit dem Kunden gepflegt um den Fortschritt der Entwicklung aufzuzeigen, Rückmeldungen einzuholen und somit den Kunden aktiv in den Entwicklungsprozess zu integrieren.

Die letzte Phase beinhaltete den Abschluss des Projekts und die Übergabe an den Kunden.

Technologien

Das gesamte Projekt wurde mit C# umgesetzt. Um möglichst viel gemeinsamen Code für die Android und iOS Applikation zu verwenden, wurde das Cross-Plattform Framework von Xamarin verwendet. Die Applikation ist für alle Android Geräte ab Version 4.3 und iOS Geräte ab Version 7 nutzbar. Für die Bereitstellung der Applikationsdaten wurden die Online-Dienste der Cloud Computing Plattform von Microsoft Azure verwendet.

Die Geberit AG kann die von ihnen laufend aktualisierten Daten im gewohnten Format auf den Applikationsserver hochladen. Dort werden sie automatisch in das nötige Format umgewandelt und für die Applikation bereitgestellt. Zudem können sie die Konten der Servicetechniker verwalten.

Ergebnisse

Die Applikation unterstützt den Servicetechniker besser in seiner Tätigkeit durch die Möglichkeit von proaktiven Meldungen zu vorhandenen Fehlern und Software Aktualisierungen des verbundenen Gerätes. Das überarbeitete Design verwendet bekannte und verbreitete Elemente von mobilen Applikationen. Somit findet sich ein Servicetechniker in der Applikation schneller zurecht. Zudem wurde darauf geachtet die Bedienung möglichst flach zu

strukturieren um die Funktionen auf kurzem Weg zu erreichen.

Für die Bluetooth Funktionalität wurde eine Lösung gefunden, welche die Implementierung im gemeinsamen Code ermöglicht, ohne diese in nativem Plattform Code für Android und iOS programmieren zu müssen. Anhand eines Testgerätes der Geberit AG konnte die Funktionalität getestet werden. Die Umsetzung beschränkt sich jedoch auf das Auslesen von Grunddaten des Testgerätes, da die Bluetooth-Schnittstelle gerade aktualisiert wird und zum Zeitpunkt der Arbeit nicht zur Verfügung stand.

Durch die Implementierung des Backends in C# und das Bereitstellen in der Azure Cloud, ist die Applikation in einer Umgebung, welche den Kernkompetenzen des Geberit AG Entwicklungsteams entspricht. Zudem werden die Applikationsdaten direkt im Backend aufbereitet und entlastet somit die mobile Applikation.

Mithilfe des neuen Admin Interfaces als Webapplikation verfügt die Geberit AG nun über eine zukunftssichere Grundplattform, welche die Administration der Benutzerkonten vereinfacht und vereinheitlicht.

Ausblick

Die Applikation wird an die Geberit AG übergeben und dort als Basis zur Weiterentwicklung verwendet.

Durch einen sehr hohen Anteil an gemeinsamen Code, ist eine optimale Wart- und Erweiterbarkeit gewährleistet. Zudem sind alle Schnittstellen bereits vorhanden, um die Daten eines realen Gerätes zu Verwenden, sobald die Bluetooth-Schnittstelle bereitsteht.

Mit der Verwendung der Cloud Computing Plattform von Microsoft Azure, ist eine hohe Skalierbarkeit und ein weltweiter Einsatz garantiert. Des weiteren ermöglicht die Verwendung des Azure Storage einfache Erweiterungen der Benutzer- und Applikationsdaten für zukünftige Funktionalitäten.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Erklärung der Eigenständigkeit

Hiermit erklären wir,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- das ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Rapperswil, 12.12.16

Pascal Marty

P. Marty

Noah Hendrikx

N. Hendrikx

Aufgabenstellung Studienarbeit
„Entwicklung einer Cross-Plattform ServiceApp für Sanitäranlagen“
HS 2016

1. Auftraggeber und Betreuer

- *Auftraggeber:* Geberit Verwaltungs AG
- *Betreuer:* Prof. Dr. Farhad Mehta

2. Studierende

Diese Arbeit wird als Studienarbeit an der Abteilung Informatik durchgeführt von

- Hr. Pascal Marty
- Hr. Noah Hendrikx

3. Ausgangslage

Geberit entwickelt hochleistungsfähige, intelligente Systeme und Produkte in der Sanitärtechnik. Geberit Systeme für Installationswände, Trinkwasser- und Abwasserinstallationen sind weltweit anerkannt. Mit der Geberit AquaClean ServiceApp für Android und iOS, entwickelt Geberit eine App, welche Servicetechnikern ermöglicht über eine Bluetooth Schnittstelle Fehler aus dem neusten Dusch-WC AquaClean Mera auszulesen und ihnen anhand der Fehlermeldungen Vorschläge für Reparaturmassnahmen liefert. Ebenso ist es mit der App möglich Firmware-Updates des Dusch-WCs vorzunehmen. Die App wurde Hybrid für iOS und Android entwickelt – Bluetooth Schnittstelle jeweils nativ, UI über Cordova plattformübergreifend.

Da die Kompetenz der Softwareentwicklungs-Abteilung in der Geberit klar im .NET C# Bereich liegt, gilt es die App auf Xamarin zu portieren, damit eine mögliche Erweiterung und Pflege der App innerhalb der Abteilung erleichtert werden kann.

4. Beschreibung der Aufgabe

Der Pflichtteil dieser Arbeit umfasst die bereits bestehende Geberit AquaClean ServiceApp nach Xamarin zu portieren und die Businesslogik automatisiert zu testen. Dabei gilt es darauf zu achten, dass möglichst viel SharedCode mittels Xamarin Forms verwendet wird, bzw. es soll untersucht werden was Xamarin Forms bereits hergibt und bis zu welchem UI Grad es Sinn macht Xamarin Forms einzusetzen und ab wo besser nativer Code für die entsprechenden Plattformen verwendet wird.

Der optionale Teil dieser Arbeit umfasst das CodedUI Testing in der Xamarin TestCloud, welche das UI der Apps automatisiert auf den entsprechenden Plattformen testet.

Ziel dieser Studienarbeit ist es eine Cross-Plattform App zu entwickeln und die Punkte im Pflichtteil zu erfüllen. Das erfüllen des Optionalen teils ist erwähnenswert, aber nicht Pflicht.

Die Nutzungsrechte der auszuliefernden Artefakte dieses Projekts liegen ausschliesslich bei der Geberit Verwaltungs AG. Eine Kopie des Source Codes wird am Ende des Projektes an die HSR übergeben zur HSR-Internen Weiterverwendung.

5. Zur Durchführung

Mit dem Betreuer finden wöchentliche Besprechungen statt. Besprechungen mit dem Auftraggeber sind von den Studierenden nach Bedarf zu initialisieren.

Alle Besprechungen sind von den Studierenden mit einer Traktandenliste vorzubereiten, die Besprechung ist durch die Studierenden zu leiten und die Ergebnisse sind in einem Protokoll festzuhalten, das den Betreuern und dem Auftraggeber per E-Mail zugestellt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

6. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen (siehe https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html). Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Alle Resultate sind vollständig auf CD/DVD in 3 Exemplaren abzugeben. Der Bericht ist ausgedruckt in doppelter Ausführung abzugeben.

7. Termine

Siehe Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html.

8. Arbeitsumfang

Eine erfolgreiche Studienarbeit zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von 30 Stunden budgetiert. Die verwendete Arbeitszeit muss erfasst und dokumentiert werden.

9. Beurteilung

Für die Beurteilung ist der HSR-Betreuer verantwortlich. Die Benotung erfolgt gemäss folgender Tabelle.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte	1/5
3. Inhalt	3/5

Im Übrigen gelten die Bestimmungen der Abteilung Informatik für Studienarbeiten (siehe https://www.hsr.ch/Ablaeufe-und-Regelungen-Studie.7479.0.html).

Rapperswil, den 27.05.2016
Prof. Dr. Farhad Mehta

Inhaltsverzeichnis

1 Analyse

1.1 Ausgangslage

1.1.1 Anwendungsumfeld

Die Geberit AG entwickelt hochleistungsfähige, intelligente Systeme und Produkte in der Sanitärtechnik. Geberit Systeme für Installationswände, Trinkwasser- und Abwasserinstallationen sind weltweit anerkannt. Mit der Aqua-Clean ServiceApp für Android und iOS, entwickelt Geberit eine Applikation, welche es Servicetechnikern ermöglicht über eine Bluetooth-Schnittstelle Fehler aus dem neusten Dusch-WC auszulesen und ihnen anhand der Fehlermeldungen Vorschläge für Reparaturmassnahmen liefert. Ebenso ist es mit der Applikation möglich, Firmware Aktualisierungen des Dusch-WCs vorzunehmen.

1.1.2 Bestehende Applikation

Die Applikation wurde hybrid für iOS und Android mithilfe von Cordova entwickelt. Die Bluetooth-Schnittstelle wurde dort jeweils nativ implementiert und das User Interface in einem gemeinsamen Codeteil. Die Applikationsdaten werden über ein Backend eines externen Entwicklungsteams bereitgestellt.

Der jeweilige native Code war in JavaScript implementiert, was zu sehr viel Low-Level Boilerplate Code führt. Für eine Weiterentwicklung war dies nicht mehr optimal. Zudem war es sehr schwierig ein Testing dafür zu realisieren. Die Code Basis wird ausschliesslich für eine Mobile Applikation und nicht als Browser Web Applikation weiter verwendet, somit war man nicht auf die Cordova Web Basis angewiesen. Zudem entsprach die Implementation mit Cordova nicht den Kernkompetenzen des Entwicklungsteams der Geberit AG.

Ausserdem verfügt die Applikation über einen Vorführmodus, der es ermöglicht, sämtliche Funktionen auszuprobieren, ohne ein entsprechendes Zielgerät zur Verfügung zu haben.

Die Ziele waren, in erster Linie, die Applikation nach Xamarin zu portieren und eine möglichst grosse Codebasis zu haben, welche für Android und iOS

wiederverwendet werden kann, um die Wartbar- und Erweiterbarkeit hoch zu halten. Zudem soll auch in der neuen Applikation ein Vorführmodus vorhanden sein. Als optional definiert, war die Anwendung von Coded-UI Tests (siehe 1.1.3), welche durch das Xamarin Framework ermöglicht werden, um eine Test Basis für die Applikation zu bieten.

Während der Elaboration-Phase wurde festgestellt, dass sich die Bluetooth-Funktionalität zum Zeitpunkt der Arbeit nur in einer Grundversion realisieren lässt. Somit wurde die Aufgabenstellung um die Ablösung des Backends mittels Produkten der Cloud Computing Dienste von Microsoft Azure erweitert. Da die Geberit AG bereits gute Erfahrungen damit gemacht hat und diese auch in C# implementiert werden konnten, entsprach dies den Interessen des Kunden.

1.1.3 CodedUI Testing

Das Prinzip von CodedUI Testing besteht darin, die Bedienung der Applikation zu automatisieren. An beliebiger Stelle kann dann verifiziert werden, ob gewisse Elemente in erwarteter Form vorhanden sind. Somit kann die Funktionalität der UI-Elemente und die Logik der Benutzeroberfläche gleichzeitig getestet werden.

1.2 Domainanalyse

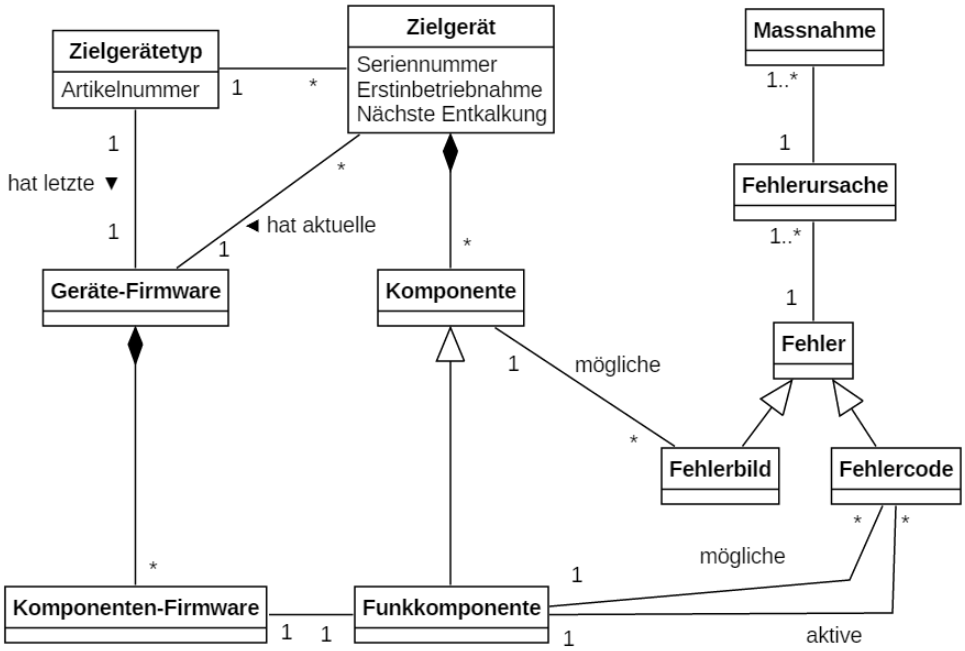

Abbildung 1.1: Domainmodel

1.2.1 Kurzbeschreibung der Konzepte [1]

Zielgerät Ein Zielgerät ist eine Sanitäranlage oder im speziellen Fall der Vorführmodus.

Zielgerätetyp Mehrere gleiche Zielgeräte gehören zu einem Zielgeräte-Typen

Geräte-Firmware Beschreibt die aktuelle Version eines Gerätes

Komponente Ein Zielgerät lässt sich in mehrere Komponenten unterteilen.

[1] Eine detailliertere Ausführung der Konzepte ist im Anhang zu finden. A.1

Funkkomponente Komponente, welche Fehlercodes aussenden könnte.

Komponenten-Firmware Beschreibt die aktuelle Version dieser Komponente.

Fehler Stellvertretend für Fehlerbilder und Fehlercodes einer Komponente. Beispiel: Die Komponente "Orientierungslicht" kann ein Fehlerbild haben "Licht schaltet nicht ein" und ein Fehlercode "0C09 - Helligkeitssensor Fehler" haben.

Fehlerbild Eine Beschreibung des Fehlers, wie er sich zeigen kann.
Beispiel: "Licht schaltet unbeabsichtigt ein"

Fehlercode Eine Beschreibung des Fehlercodes, der vom Gerät gesendet wird. Beispiel: "012E - Steuerung nicht gefunden"

Fehlerursache Beschreibt die möglichen Ursprünge eines Fehlers.
Beispiel: "Steckverbindung fehlerhaft" oder "Kabelbruch"

Massnahme Beschreibt die notwendigen Schritte zur Behandlung einer Fehlerursache.
Beispiel: "Verkabelung prüfen" oder "Komponente ersetzen"

1.3 Use Cases

1.3.1 Use Case Diagram

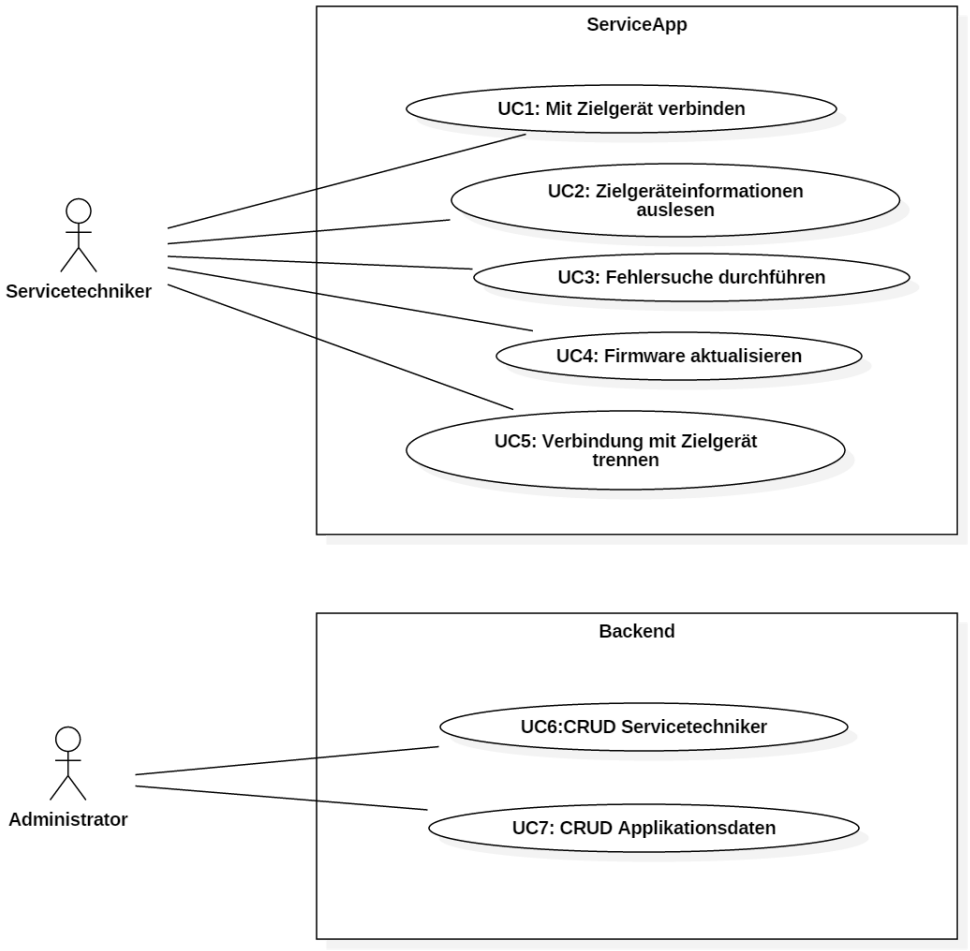

Abbildung 1.2: Use Case Diagram

1.3.2 Aktoren & Stakeholder

- Servicetechniker
- Administrator
- Kunde

1.3.3 Kurzbeschreibung der Use Cases [2]

UC1: Mit Zielgerät verbinden Ein Servicetechniker verbindet seine Applikation mit einem Zielgerät in der Nähe.

UC2: Zielgeräteinformationen auslesen Ein Servicetechniker liest Informationen aus einem verbundenen Zielgerät aus.

UC3: Fehlersuche durchführen Ein Servicetechniker liest Fehlercodes aus dem Zielgerät aus oder analysiert sie anhand von Fehlerbildern.

UC4: Firmware aktualisieren Ein Servicetechniker stellt fest, ob eine neue Firmware für die gerade verbundene Sanitäranlage verfügbar ist. Falls dies der Fall ist, spielt er die aktuellere Version der Firmware auf die Sanitäranlage auf.

UC5: Verbindung mit Zielgerät trennen Ein Servicetechniker trennt die Verbindung zu einer verbundenen Sanitäranlage.

UC6: CRUD Servicemitarbeiter Ein Administrator bearbeitet Daten von Servicetechnikern.

UC7: CRUD Applikationsdaten Ein Administrator verwaltet und bearbeitet Daten, welche für die Applikation wichtig sind (Firmwares, Übersetzungen, etc.). Die Applikation prüft diese regelmässig auf Aktualisierungen und lädt jeweils die neuste Version auf das Mobilgerät.

[2] Eine detailliertere Ausführung der Use Cases ist im Anhang zu finden. A.2

1.4 Functional Requirements

Nr.	Anforderung
FR1	Beim Aufstarten der Applikation wird ein Start Screen mit dem Geberit-Logo in der Mitte angezeigt.
FR2	Die Applikation ist in den folgenden 23 Sprachen verfügbar: Deutsch, Englisch, Französisch, Italienisch, Niederländisch, Dänisch, Norwegisch, Schwedisch, Finnisch, Spanisch, Portugiesisch, Slowenisch, Tschechisch, Polnisch, Slowakisch, Ungarisch, Kroatisch, Serbisch, Russisch, Rumänisch, Türkisch, Chinesisch, Bulgarisch
FR3	Es ist für den Benutzer ersichtlich, dass er sich im Vorführmodus befindet.
FR4	Ein Servicetechniker-Konto darf nur jeweils für ein Gerät verwendet werden. Falls das Konto verwendet wird um ein neues Gerät zu authentisieren, muss das alte Gerät automatisch ausgeloggt werden, sobald es eine Internetverbindung hat.
FR5	Die Applikation verfügt über Hilfe, Impressum, Rechtshinweise und Datenschutzerklärungen.

Tabelle 1.1: Funktionale Anforderungen

1.5 Non Functional Requirements

NFR1: Das Starten der Applikation dauert auf einem Referenzsystem 1.5.1 oder neueren Gerät, mit einer stabilen 3G-Verbindung, nicht länger als 10 Sekunden, solange die Applikationsdaten zusammen nicht mehr als 5 MB gross sind.

NFR2: Die Applikation verfügt über Coded UI-Tests.

1.5.1 Referenzsysteme

Folgende Geräte dienten als Referenzsysteme:

- Samsung Galaxy Nexus i9250 mit Android 4.3
- Samsung Galaxy S5 mit Android 6.0.1

- LG Nexus 5x mit Android 7.1.1
- Apple iPhone 5 mit iOS 7

1.5.2 Randbedingungen

- Minimale Android-Version: 4.3
- Minimale iOS-Version: 7
- Referenzsystem: iPhone 5 & Samsung Galaxy S5

1.5.3 Qualitätsmerkmale

Zuverlässigkeit Die Applikation ist auch ohne bestehende Internetverbindung funktionsfähig mit Ausnahme der Authentifizierung, sowie dem Herunterladen neuer Applikationsdaten.

Verständlichkeit Für den User soll durch Hinweise immer erkenntlich sein, ob eine ausgeführte Aktion erfolgreich war oder nicht. Die Navigation der Applikation soll möglichst simpel gehalten werden.

Erlernbarkeit Wo möglich wurden bereits bekannte Bedienelemente aus den Material Design Guidelines [3] verwendet. Diese geben eine gute Basis und werden weit verbreitet eingesetzt und für den Benutzer wird vertraute Oberfläche geschaffen.

Bedienbarkeit Die Bedienung ist möglichst flach strukturiert. Der Grundgedanke ist, dass alle Funktionen schnell und auf kurzem Weg erreichbar sind. Es ist keine Schulung für die Bedienbarkeit nötig.

1.6 Systemtest Spezifikation [4]

Zur Überprüfung der Applikationsfunktionalität werden vom Entwicklerteam ab Mitte der Construction-Phase wöchentlich Systemtests durchgeführt.

[3] `https://material.io/guidelines/`

[4] Eine detaillierte Ausführung der Systemstests ist im Anhang zu finden A.3

1.6.1 Abdeckung

Mit folgender Tabelle wird gezeigt, dass alle Functional Requirements (FR), Non Functional Requirements (NFR) und Use Cases durch Systemtests abgedeckt sind.

UC	**FR**	**NFR**	**Systemtest**
	FR1	NFR1	ST1: Starten der Applikation
	FR4		ST2: Authentifizieren
UC1, UC2	FR3		ST3: Mit Vorführmodus verbinden
UC3	FR3		ST4: Fehlerbilder im Vorführmodus anzeigen
UC3	FR3		ST5: Fehlercodes im Vorführmodus anzeigen
UC4	FR3		ST6: Firmware im Vorführmodus aktualisieren
UC5	FR3		ST7: Verbindung mit Vorführmodus trennen
UC6			ST8: Benutzer erstellen
UC6			ST9: Benutzer löschen
UC7			ST10: Applikationsdaten bearbeiten
	FR2		ST11: Sprache auswählen
	FR5		ST12: Hilfe, Impressum, Rechtshinweise und Datenschutzerklärungen aufrufen

Tabelle 1.2: Systemtests

1.6.2 Systemtest Durchführungen

Die Systemtests wurden ab Iteration 8 wöchentlich durchgeführt, um den Fortschritt zu kontrollieren.

Systemtest	**W8**	**W9**	**W10**	**W11**	**W12**
ST1	✓	✓	✓	✓	✓
ST2	✗	✗	✓	✓	✓
ST3	✗	✗	✗	✗	✓
ST4	✓	✓	✓	✓	✓
ST5	✓	✓	✓	✓	✓
ST6	✓	✓	✓	✓	✓
ST7	✓	✓	✓	✓	✓
ST8	✓	✓	✓	✓	✓
ST9	✗	✗	✗	✓	✓
ST10	✓	✓	✓	✓	✓
ST11	✗	✓	✓	✓	✓
ST12	✓	✓	✓	✓	✓

Tabelle 1.3: Systemtest Durchführungen

2 Projektmanagement

2.1 Projektplan [5]

Das Projekt beinhaltet sechs Meilensteine, die über folgende vier Phasen verteilt sind:

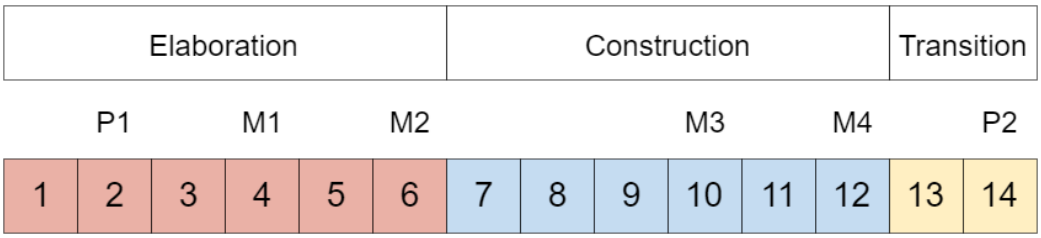

Abbildung 2.1: Projektplan

Inception:

Erstellen des Projektantrags und Meeting mit Industriepartner. (Vor Semesterbeginn erledigt)

Elaboration:

Definieren des Projektablaufs, Use Cases, Domain Models, erste Design Entwürfe, Prototyp, Risikominderung.

Construction:

Entwickeln und Testen der Applikation für die betreffenden Plattformen.

Transition:

Ausführlichere Testfälle, Bereinigung des Source-Codes, Konfiguration und Optimierung der Produktivumgebung und Installation.

[5] Das detailliertere Gantt-Diagramm ist im Anhang zu finden. A.4.1

2.2 Meilensteine

P1: Projektplan 02.10.16 (Woche 2)

Fertigstellung des Projektplans. Der Funktionsumfang und die User-Experience der bisherigen Applikation ist analysiert. Erste Erfahrungen mit Xamarin Forms und dessen Möglichkeiten sind gemacht worden. Alle Use Cases sind erfasst. Die Infrastruktur der lokalen Testumgebung ist aufgesetzt.

M1: GUI Design, UCs, FR, NFR 16.10.16 (Woche 4)

Ein erster Vorschlag des kompletten Applikations Designs ist als Clickable Prototype definiert. Alle Use Cases sind erfasst. Das Domain Model ist komplett erstellt. Die nicht-funktionalen Anforderungen sind erfasst.

M2: End of Elaboration 30.10.16 (Woche 6)

Der Software-Architektur Entwurf steht und alle Schnittstellen zum Backend sind definiert. Ein Prototyp, welcher durch die ganze Architektur durchschlägt, ist erstellt. Die komplette Entwicklungsumgebung ist eingerichtet. Alle Use Cases sind im Format fully dressed erfasst.

M3: Mid of Construction 27.11.16 (Woche 10)

Die Software-Architektur ist komplett definiert und grösstenteils implementiert. Die Verwaltung und Pflege des Backends funktioniert.

M4: End of Construction 11.12.16 (Woche 12)

Die Applikation ist fertig implementiert. Die Funktionalität ist komplett umgesetzt.

P2: Abgabe Arbeit 23.12.16 (Woche 14)

Abgabe aller Dokumente.

2.3 Risikomanagement [6]

R1: Framework Unkenntnis

Beschreibung	Zu wenig Erfahrung mit den verwendeten Frameworks
Massnahme	Einstudieren mit Tutorials und Testprojekte erstellen
Verhalten bei Eintritt	Kontaktaufnahme mit einer Fachperson

Tabelle 2.1: R1: Framework Unkenntnis

R2: Kommunikationsverlust

Beschreibung	Kommunikation zwischen Kunde und Entwicklerteam erweist sich als schwierig
Massnahme	Vorausschauende Planung von Meetings und fortlaufende Klärung von Problemen
Verhalten bei Eintritt	Iterationsplanung anpassen und Kontakt zum Kunden aufnehmen, Probleme besprechen

Tabelle 2.2: R2: Kommunikationsverlust

R3: Kommunikation zu Umsystemen zu komplex

Beschreibung	Ablösung und Umstellung der bisherigen Datenablage wird zu aufwendig
Massnahme	Gründliche Anforderungsanalyse der Umsysteme und Abklärung der vorhandenen Möglichkeiten
Verhalten bei Eintritt	Bestehende Lösung weiterverwenden

Tabelle 2.3: R3: Kommunikation zu Umsystemen zu komplex

[6] Eine Einschätzung der Risiken ist im Anhang zu finden. A.5

R4: Datenverlust oder Manipulierung

Beschreibung	Die sensiblen Daten werden gestohlen, manipuliert oder gehen verloren
Massnahme	Sicherheitsschranken so weit möglich und sinnvoll implementieren, Best Practices in diesem Bereich verfolgen, Nutzen eines Versionsverwaltungssystems
Verhalten bei Eintritt	Sicherheitsmassnahmen anpassen

Tabelle 2.4: R4: Datenverlust oder Manipulierung

R5: Komplexität wird unterschätzt

Beschreibung	Ein oder mehrere Teammitglieder verlieren sich in einem Feature oder können ein ihnen zugeteiltes Arbeitspaket nicht in der geforderten Zeit finalisieren
Massnahme	Wöchentliche Meetings mit Feedback, gemeinsame Aufteilung der Arbeitspakete, ständiges Kontrollieren des Backlogs und des Burndown-Charts, die Arbeitspakete mit Zeitreserven planen
Verhalten bei Eintritt	Neuaufteilung der noch anstehenden Arbeit, Überarbeitung der Arbeitspakete, eventuell Featureredukt ion

Tabelle 2.5: R5: Komplexität wird unterschätzt

R6: Suboptimale Verwendung von Komponenten in der Toolchain

Beschreibung	Die Toolchain greift nicht wie geplant ineinander oder eine Komponente erweist sich als nicht ideal
Massnahme	Zurückgreifen auf vorhandene Erfahrung, ausführliches Testen zu Beginn
Verhalten bei Eintritt	Toolchain neu definieren, einzelne Werkzeuge austauschen oder neu konfigurieren

Tabelle 2.6: R6: Suboptimale Verwendung von Komponenten in der Toolchain

R7: Ausfall Entwicklermaschine

Beschreibung	Der Laptop eines Teammitglieds geht verloren oder fällt langfristig aus und er kann nicht weiterarbeiten
Massnahme	Geht nur begrenzt, Umgang mit Laptops wie gewohnt nach gesundem Menschenverstand, nutzen eines Versionsverwaltungssystems, Dokumentation zur Installation der Umgebung, regelmässige Commits und Backups
Verhalten bei Eintritt	Ersatz besorgen, dann gemäss Dokumentation die Umgebung wieder installieren und die aktuelle Version des Projektes aufspielen

Tabelle 2.7: R7: Ausfall Entwicklermaschine

R8: Bluetooth-Schnittstelle

Beschreibung	Die Implementation der Bluetooth-Schnittstelle wird unterschätzt und erweist sich als schwierig.
Massnahme	Frühes Nachforschen über Bluetooth in der Elaboration, um eine eigene Übersicht und Einschätzung des Risikos zu haben. Gespräche mit dem Kunden führen, über Erfahrungen in der Entwicklung von Bluetooth-Schnittstellen.
Verhalten bei Eintritt	Reduzierung der Bluetooth-Funktionalität.

Tabelle 2.8: R8: Bluetooth-Schnittstelle

3 Konzeption und Design

3.1 Backend

Das Backend muss die Benutzerkonten Authentifizierung durchführen und Zugang zu den Applikationsdaten ermöglichen. Der Mobile App Service von Microsoft Azure ermöglicht dies, neben anderen Sprachen, mit C# zu implementieren. Zudem wird eine Library für diverse Plattformen wie Xamarin bereitgestellt, welche die Kommunikation zum Backend erleichtert.

Die Applikationsdaten werden in einem Azure Storage persistiert, welcher mit dem Mobile App Service verbunden ist

3.2 Applikation

Die Technologie wurde durch die Aufgabenstellung mit Xamarin als Cross-Plattform Framework bereits definiert.

3.2.1 Funktionsweise Xamarin

Grundsätzlich ermöglicht es das Xamarin Framework, Android und iOS Applikationen in C# zu schreiben mithilfe von Xamarin.Android und Xamarin.iOS. Zusätzlich erlaubt Xamarin.Forms, ausgewählte Grundelemente in einem eigenen Projekt zu implementieren, die beim Kompilieren für beide Plattformen verwendet werden. Diese Elemente sind begrenzt, da nur solche in Frage kommen, welche in beiden Plattformen abgebildet werden können.

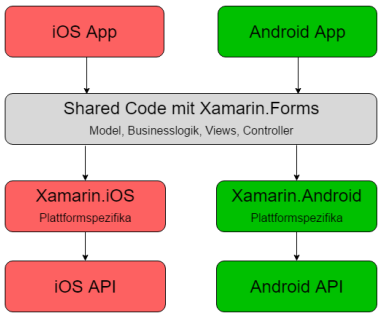

Abbildung 3.1: Xamarin.Forms Struktur

3.2.2 Shared Code Projekt

Gemäss Aufgabenstellung soll soviel Applikationscode wie möglich in diesem Teil implementiert werden, da dies die Wartung und Erweiterung vereinfacht. Sämtliche Model, Views und Controller werden hier definiert, sowie die Abhandlung der Businesslogik.

3.2.3 Android und iOS Projekt

Plattformspezifische Funktionen welche gar nicht oder nur unzureichend im Shared Code realisiert werden können, müssen auf den jeweiligen Plattformen nativ implementiert werden. Dies sollte nach Möglichkeit vermieden werden, da ein Feature jeweils an zwei Orten angepasst werden muss und unter Umständen die User-Experience nicht gleich ist, oder pro Plattform anders gelöst werden muss.

3.2.4 Bluetooth

Der aufwändigste Teil der bisherigen Applikation war der Bluetooth Teil, da der Unterschied in dieser API zwischen Android und iOS recht gross ist. Auch von Xamarin selber gibt es hierfür keine Lösung und die allgemeine Empfehlung ist, die jeweils native API in den entsprechenden Plattformen zu verwenden.

Jedoch konnte mit dem Bluetooth LE plugin for Xamarin [7] ein Open Source Projekt gefunden werden, das eine Implementierung grosser Anteile der Bluetooth-Basisfunktionalität im Shared Code ermöglicht. Die nativen Elemente werden dabei von diesem Plugin bereits zur Verfügung gestellt.

Somit war eine Implementierung der Grundfunktionalität mit dem Testgerät der Geberit AG möglich, ohne nativen Plattform Code schreiben zu müssen.

[7] `https://github.com/xabre/xamarin-bluetooth-le`

3.2.5 Fazit

Während der Elaboration-Phase konnte durch erstellen diverser Prototypen festgestellt werden, dass alle Bedienelemente, welche durch die Mockups 4.2 in der Design Analyse definiert wurden, mithilfe von Xamarin Forms Elementen realisiert werden können.

3.3 Admin Interface

Microsoft Azure stellt für die Verwaltung seiner Azure Storages ein eigenes Tool [8] zur Verfügung. Nach Rücksprache mit dem Kunden wurde klar, dass die Verwaltung, speziell die der Benutzerkonten, vereinfacht werden sollte.

[8] Azure Storage Explorer Link

4 UI-/UX-Konzept

4.1 Navigationskonzept

Die Abbildung 4.1 zeigt eine Übersicht der wichtigsten Navigationsmöglichkeiten innerhalb der Applikation. Die Menüpunkte sind über ein Navigationsmenü auf der linken Seite (siehe 4.7 und 4.8) zu finden.

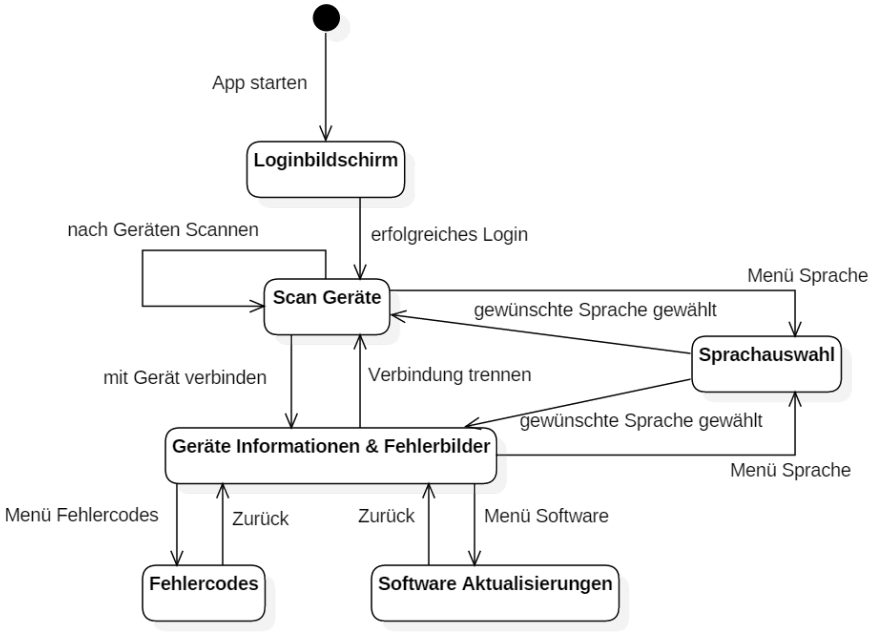

Abbildung 4.1: Navigations-Diagramm

4.2 Mockups

4.2.1 Geräte Scan

Die Scan Seite listet alle Geräte in der Umgebung auf, mit denen sich die Applikation verbinden kann.

Zudem kann sich der Benutzer mit einem Vorführgerät verbinden, welches immer zur Verfügung steht. Damit kann die Applikation getestet werden, ohne ein Gerät zur Verfügung zu haben.

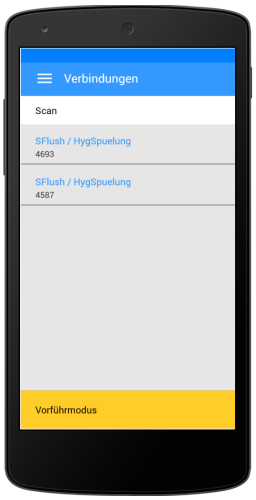

Abbildung 4.2: Geräte Scan

4.2.2 Geräte Informationen

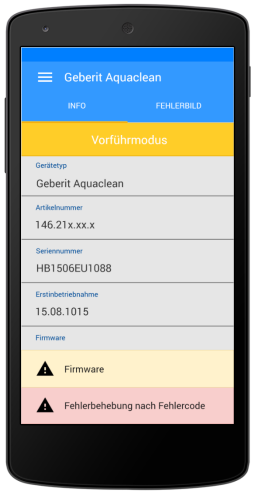

Abbildung 4.3: Geräte Informationen

Auf der Geräte Informationen Seite sind alle wichtigen Angaben zum aktuell verbundenen Gerät aufgelistet. Bei Verbindung mit dem Vorführgerät, wird dies am oberen Rand gekennzeichnet.

Falls Fehlercodes oder Firmware Aktualisierungen vorhanden sind, werden die entsprechenden Buttons am unteren Rand eingeblendet. Dies ermöglicht es dem Benutzer auf einen Blick zu erkennen, ob ein Handlungsbedarf vorliegt.

4.2.3 Fehlerbilder

Die Fehlerbilder Seite ist ein Nachschlagewerk für Fehlerursachen und die dazugehörigen Massnahmen. Diese sind nach Komponenten der Sanitäranlagen gruppiert.

Diese Seite verfügt auch über eine Kennzeichnung am oberen Rand, falls das Mobilgerät mit dem Vorführgerät verbunden ist.

Abbildung 4.4: Fehlerbilder

4.2.4 Fehlercodes

Abbildung 4.5: Fehlercodes

Die Fehlercode Seite listet alle Fehlerursachen und die dazugehörigen Massnahmen zu Fehlercodes auf, die das verbundene Gerät sendet. Zusätzlich werden zu den Fehlerursachen die genauen Fehlercodes angezeigt.

Auch diese Seite verfügt über die Vorführgerät Kennzeichnung am oberen Rand.

4.2.5 Firmware

Innerhalb der Firmware Seite werden alle Komponenten aufgelistet, für welche eine Firmware Aktualisierung ansteht. Durch ein Button am unteren Rand, kann dieser Vorgang gestartet werden. Während der Aktualisierung zeigt ein Fortschrittsbalken den aktuellen Zustand an.

Durch ein Text-Element zwischen der Komponenten-Auflistung und dem Update-Button, wird der aktuelle Status beschrieben, ob gerade Aktualisierungen durchgeführt werden, diese fehlgeschlagen sind oder erfolgreich abgeschlossen wurden.

Abbildung 4.6: Firmware

4.2.6 Menü nicht verbunden

Abbildung 4.7: Menü nicht verbunden

Das Menü der Applikation besteht aus drei Teilen. Ein Indikator, welcher anzeigt, ob ein Gerät gerade verbunden ist, ein dynamischer Menü-Teil, welcher sich ändert, wenn ein Gerät verbunden ist, und ein statischer Menü-Teil, welcher Punkte umfasst, die immer verfügbar sind.

Solange kein Gerät verbunden ist, zeigt der Indikator dies entsprechend an und der Menü-Punkt Scan ist verfügbar, um sich mit Geräten in der Nähe zu verbinden. Der statische Menü-Teil umfasst Dinge wie die Sprachauswahl oder Datenschutzbestimmungen.

4.2.7 Menü verbunden

Wenn die Applikation mit einem Gerät verbunden ist, wird dies am oberen Rand des Menüs durch einen Indikator angezeigt.

Der dynamische Menü-Teil umfasst im verbundenen Zustand die Punkte: Geberit Aquaclean (was die aktuelle Geräte-Information aufruft), Fehlercodes, Software und Trennen.

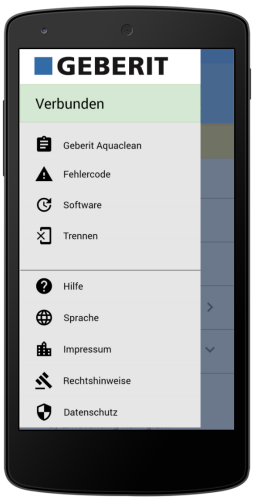

Abbildung 4.8: Menü verbunden

4.2.8 Sprachauswahl

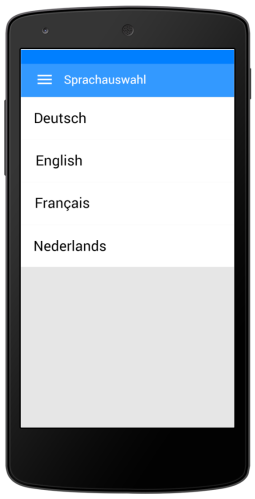

Abbildung 4.9: Sprachauswahl

Über die Seite Sprache kann die Applikation in alle verfügbaren Sprachen umgestellt werden.

Die Sprachen werden untereinander aufgelistet.

5 Ergebnisse

5.1 Deployment

Die Abbildung 5.1 zeigt die Komponenten der Applikation und wie diese untereinander kommunizieren.

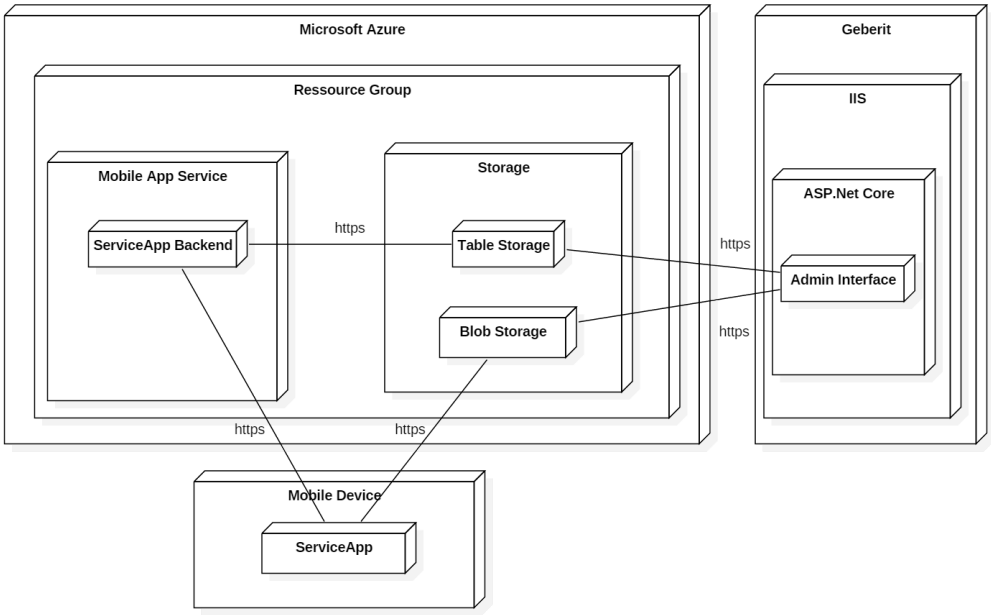

Abbildung 5.1: Deployment Diagram

5.2 Azure Backend

Das Azure Backend besteht aus den Komponenten Mobile App Service und Storage (siehe 5.1).

5.2.1 Mobile App Service

Der Mobile App Service Teil beinhaltet alle Schnittstellen, welche von der Applikation aufgerufen werden. Diese sind als Rest-Schnittstellen implementiert und umfassen in unserem Fall die Authentisierungslogik der Servicetechniker

Konten, das Parsing und Aufbereiten der von der Geberit AG gelieferten Applikationsdaten, sowie das Bereitstellen dieser.

5.2.2 Storage

Der Azure Storage umfasst Blobcontainer, in welchen die Applikationsdaten gespeichert werden und NoSQL Tabellen, in denen die Konten der Servicetechniker verwaltet werden. Der Mobile App Service und der Azure Storage können untereinander direkt kommunizieren. Die Applikation selber kann jedoch nur über eine Indirektion auf den Storage zugreifen.

Damit im Client-Applikations Code keine Zugriffsdaten zum Storage gespeichert werden müssen, wird von Microsoft empfohlen, sogenannte Shared Access Signatures (SAS) zu verwenden. Diese werden vom Mobile App Service nur an authentifizierte User ausgestellt und erlauben es dem Client dann, für einen bestimmten Zeitraum auf definierte Applikationsdaten aus den Blobcontainern zuzugreifen. Somit haben nur authentifizierte User Zugriff auf neue Applikationsdaten. Zusätzlich wird der Mobile App Service entlastet und beschränkt sich auf die Authentisierung und das Ausstellen der SAS Tokens.

5.3 Admin Interface

Für die Grundfunktionalität bietet Azure einen Storage Explorer an, welcher es ermöglicht, mit den Blobs und den Tabellen zu interagieren. Jedoch ist der Umgang mit diesem Tool nicht sehr praktikabel, da Daten schnell korrumpiert werden können, wenn das Format nicht genau den Systemanforderungen entspricht.

Das Admin Interface besteht aus einer MVC Webapplikation auf Basis von ASP.NET Core. Diese ermöglicht es, die Servicetechniker Konten übersichtlicher und einheitlicher zu verwalten. Das Projekt kann im Intranet des Kunden auf einem internen Server gehostet werden, was den Schutz zusätzlich erhöht, da dieser nicht öffentlich zugänglich ist. Zudem wird die Erweiterbarkeit und Interoperabilität durch die Verwendung von Rest-Schnittstellen gewährleistet.

5.4 Applikation

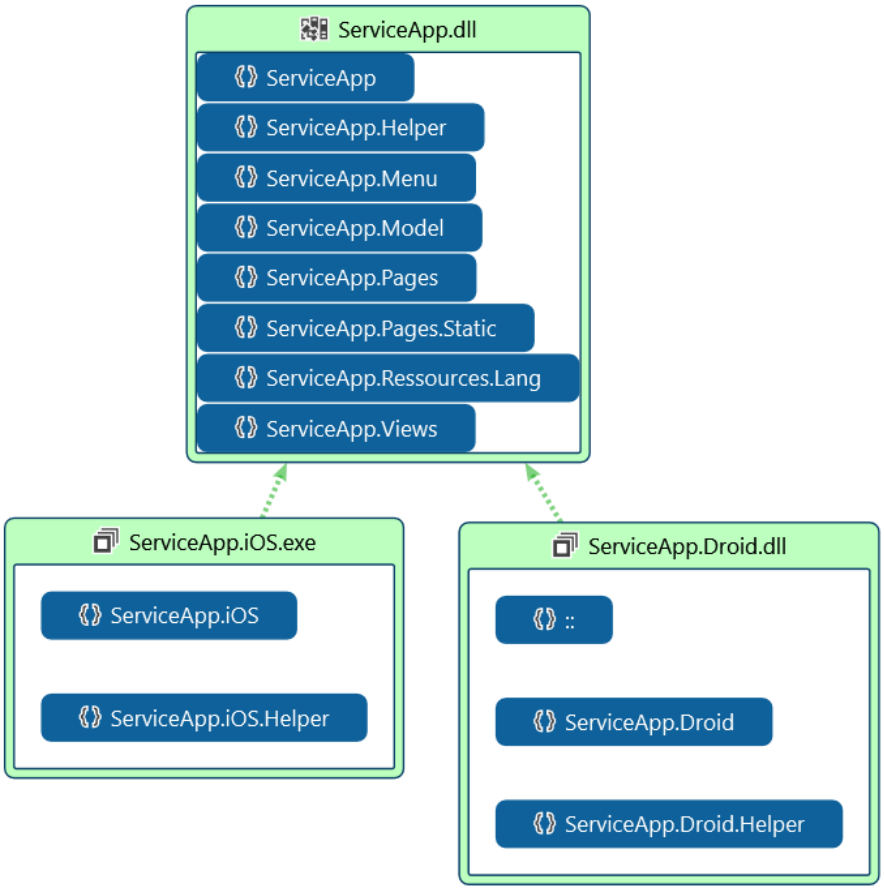

Abbildung 5.2: Projektstruktur

5.4.1 ServiceApp

Im Shared Project befinden sich sämtliche Controller, Businesslogik und UI-Komponenten, welche vom Android- und dem iOS-Projekt genutzt werden. Spezielle UI-Elemente wurden ebenfalls mit Elementen aus Xamarin.Forms individuell zusammengestellt.

5.4.2 ServiceApp.Droid und ServiceApp.iOS

Ein kleiner Teil der Funktionalität konnte nicht im Shared Project realisiert werden und musste auf den nativen Plattformen implementiert werden. Zu diesen gehört zum einen die Internationalisierung, welche beim ersten Aufstarten der Applikation die lokale Sprache des Mobilgerätes ausliest und anhand dieser Information die Applikation auf eine geeignete Sprache umstellt. Da das Auslesen und Setzen der Gerätesprache nur nativ möglich ist, wurde diese Funktion jeweils in dem Android- und iOS-Projekt implementiert. Zum anderen ist der Zugriff auf das Filesystem des Mobilgeräts ebenfalls nur nativ möglich.

5.4.3 Codemetriken

Um die Codequalität einschätzen zu können, wurde mittels der "ndepend"[9] Software eine Analyse des Sourcecodes gemacht. Nachfolgend ist die Statistik für das Projektende aufgeführt.

Lines of Code	App	1135
Lines of Code	Backend	307
Total Lines of Code		**1442**

Tabelle 5.1: Umfang

[9] `http://www.ndepend.com/`

Anzahl Klassen	App	60
Anzahl Klassen	Backend	19
(Avg) Methoden pro Klasse	App	7.77
(Avg) Methoden pro Klasse	Backend	7.71
(Avg) Lines of Code pro Methode	App	2.33
(Avg) Lines of Code pro Methode	Backend	2.26
(Max) Lines of Code Methode	App	29
(Max) Lines of Code Methode	Backend	27

Tabelle 5.2: Klassen Aufteilung

5.5 Zielerreichung

Die in dieser Arbeit entwickelte Applikation entspricht dem aktuellen Stand der Technik und erfüllt alle vom Kunden erwünschten Anforderungen (siehe 1.4).

Zusammengefasst wurden folgende Punkte erreicht:

- Die Applikation ist kompatibel für alle Geräte mit Android-Version 4.3 und iOS-Version 7 oder höher.
- Ein modernes User Interface, welches sich am Material Design orientiert
- Eine vereinfachte und flach strukturierte Navigation
- Ein Funktionsumfang, welcher bis auf die Bluetooth-Funktionalität dem Vorgänger entspricht
- Erste CodedUI-Tests, die als Basis für die Entwicklung weiterer automatisierter Tests dient
- Ein neues Admin Interface für das Erstellen und Bearbeiten von Benutzerkonten
- Eine komplette Applikationsumgebung implementiert mit Technologien, die den Kernkompetenzen des Kunden entsprechen

Somit kann dem Kunden eine gute Basis zur Weiterentwicklung der Applikation übergeben werden.

5.6 Schlussfolgerung

5.6.1 Projektverlauf

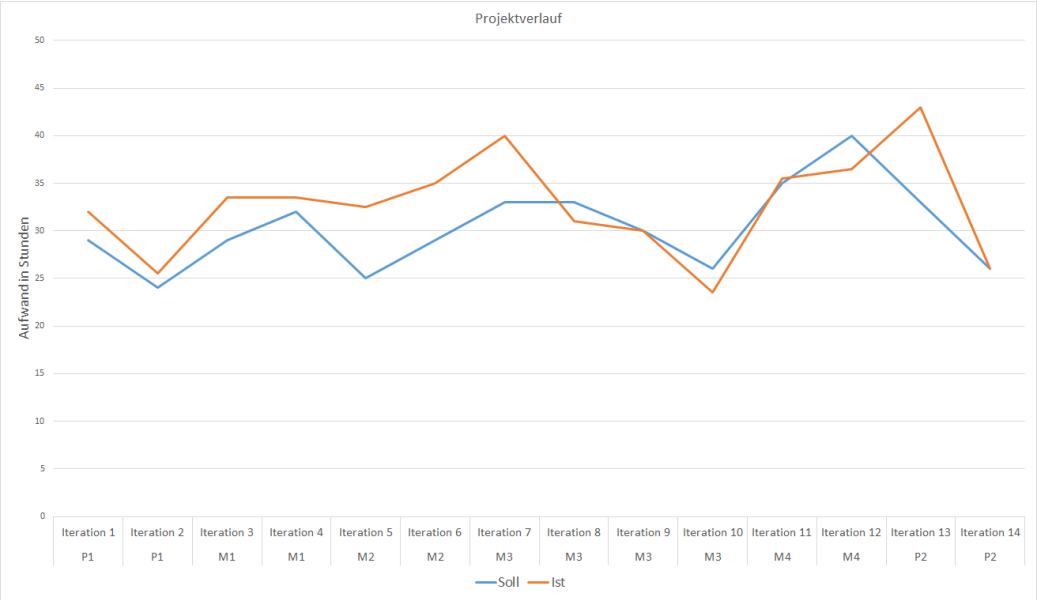

Abbildung 5.3: Projektverlauf

Die Anzahl aufgewendeter Stunden entsprach dem erwarteten Rahmen einer Studienarbeit. Die Arbeit war gleichmässig unter den Beteiligten aufgeteilt. Während der Elaboration-Phase wurde der Aufwand ein wenig unterschätzt. Jedoch ist die Eintrittswahrscheinlichkeit von Risiken während der Elaboration-Phase am höchsten, somit ist dies nicht sehr aussergewöhnlich.

Während der Construction-Phase waren die Aufwandschätzungen präziser. Dies ist ist darauf zurückzuführen, dass die meisten Risiken in der Elaboration-Phase gemindert werden konnten.

Gegen Ende verschob sich der Arbeitsaufwand aufgrund von Abschlussarbeiten und organisatorischen Gründen von der letzten auf die vorletzte Iteration.

5.6.2 Risikoanalyse

Das Risiko R5 (siehe 2.5) hat sich während der Elaboration-Phase manifestiert. Im Speziellen wurde die Komplexität der Applikationsdaten Aufbereitung in ein passendes Format, sowie die Implementierung der eigenen Authentifizierungs-Logik unterschätzt.

Das Risiko R8 (siehe 2.8) konnte zum Anfang der Construction-Phase und der verbundenen Neubewertung der Risiken, durch die beschlossene Einschränkung der Bluetooth Funktionalität, reduziert werden. Da für die Construction-Phase der Bluetooth Teil noch offen war, stellte dies zu dem Zeitpunkt das höchste Risiko dar.

5.6.3 Fazit zum Cross-Plattform Ansatz

Mit dieser Arbeit konnte ganz klar ein Mehrwert durch den Cross-Plattform Ansatz erreicht werden. Der Grossteil der Applikation konnte im Shared Code Projekt implementiert werden.

Jedoch muss im Vorfeld gut analysiert werden, was für Anforderungen an die Applikation bestehen. Xamarin Forms bietet eine begrenzte Anzahl an User Interface Elementen an. Sobald individuelle Ansprüche an die Benutzeroberfläche vorhanden sind, kommt man um eine native Implementierung kaum herum. Dann besteht die Schwierigkeit darin, dass sich die Elemente auf allen Plattformen gleich verhalten.

Sobald auf Funktionen des mobilen Gerätes zugegriffen wird, wie das File System oder die Spracheinstellungen, muss in Betracht gezogen werden, dass hier ein Mehraufwand entstehen kann, da für jede zu unterstützende Plattform nativer Code geschrieben werden muss.

5.6.4 Fazit zu CodedUI-Testing

Der klare Vorteil von CodedUI-Testing besteht darin, dass Applikationen auf verschieden Geräten, sogar plattformübergreifend, mit den selben Bedingungen getestet werden können. Zusätzlich wird auch getestet, ob sich der Applikations-Kontext korrekt verhält, da sich dessen Zustand automatisch während der Tests mitverändert. Zudem kann ein Problem genau lokalisiert werden, falls dieses nur bei einer Plattform, Betriebssystemversion oder eines Geräteherstellers auftritt.

Falls man sich für CodedUI-Tests entscheidet, muss man einen relativ hohen Implementierungs Aufwand einplanen. Es hat sich gezeigt, dass das Testen, aller möglichen Zustände einer einzelnen, kleinen Applikations Seite auf ihre Korrektheit, bereits sehr aufwändig ist. Zudem sind die automatisierten Testdurchläufe auf vielen Geräten, je nach Anbieter mit hohen Kosten verbunden.

6 Ausblick

6.1 Bluetooth

Eine einfache Kommunikation mit Bluetooth-Geräten ist bereits implementiert und dient als Basis, um die API einzubinden, sobald diese fertig gestellt wurde. Ausserdem ist eine Integration zukünftiger Gerätetypen möglich.

6.2 Clouddienste

Dank der Verwendung der Cloud Computing Plattform von Microsoft Azure, steht einer weltweiten Verteilung nichts im Weg. Zusätzlich könnte die Applikation um ein Notification Hub erweitert werden, welches eine einfache Benachrichtigung der Nutzer ermöglicht. Diese Features bietet Azure mit den verwendeten Produkten bereits an.

6.3 Verwaltung der Nutzerdaten

Das bereitgestellte Admin Interface bietet eine Grundfunktionalität zur Verwaltung der Nutzerdaten. Dieses lässt sich gut um weitere Features, wie das Verwalten der Applikationsdaten erweitern. Zusätzlich eignen sich die verwendeten schemalosen NoSQL-Datenbanken sehr gut, um später um zusätzliche Informationen erweitert zu werden, ohne dass jedes mal eine Datenbank Migration vorgenommen werden muss.

6.4 Testing

Die vorhandenen CodedUI-Tests dienen als Grundlage, die Applikation vollständig automatisiert zu testen. Diese können auch für die Xamarin Test-Cloud verwendet werden, um die Tests gleichzeitig auf einer Vielzahl von ausgewählten Mobilgeräten zu testen.

Abbildungsverzeichnis

Tabellenverzeichnis

A Anhang

A.1 Konzept Detail

A.1.1 Zielgerät

Beschreibung	Ein Zielgerät ist eine Sanitäranlage oder im speziellen Fall der Vorführmodus
Attribute	**Seriennummer:** Identifikation eines einzelnen Zielgerätes **Erstinbetriebnahme:** Datum wird bei der ersten Inbetriebnahme gesetzt **Nächste Entkalkung:** Datum der nächsten geplanten Entkalkung
Beziehungen	**Zielgerätetyp:** Zielgeräte sind in verschiedene Zielgerätetypen unterteilt **Komponente:** Definiert aus welchen Komponenten ein Zielgerät besteht

A.1.2 Zielgerätetyp

Beschreibung	Mehrere gleiche Zielgeräte gehören zu einem Zielgerätetypen.
Attribute	Artikelnummer: Identifikation eines Gerätetypes
Beziehungen	-

A.1.3 Geräte-Firmware

Beschreibung	Beschreibt die aktuelle Version eines Gerätes
Attribute	-
Beziehungen	**Zielgerät:** Die aktuelle Version der Firmware, die auf diesem Zielgerät gerade läuft. **Zielgerätetyp:** Die letzte (neueste) Firmware, die für diesen Zielgerätetyp verfügbar geworden ist. **Komponenten-Firmware:** Die Geräte-Firmware setzt sich aus allen Komponenten-Firmwares zusammen.

A.2 Use Cases Detail

A.2.1 UC1: Mit Zielgerät verbinden

Description	Ein Servicetechniker verbindet seine Applikation mit einem Zielgerät in der Nähe.
Primary Actor	Servicetechniker
Trigger	Ein Servicetechniker möchte sich mit einem Zielgerät verbinden.
Stakeholder and Interests	Servicetechniker: Will ein Zielgerät analysieren.
Preconditions	Es ist ein kompatibles Zielgerät in der Nähe.
Postconditions	Die Applikation ist mit dem ausgewählten Zielgerät verbunden.
Main Success Scenario	1. Benutzer scannt nach Zielgeräten in der Nähe. 2. Applikation zeigt verfügbare Zielgeräte in der Nähe an. 3. Benutzer wählt eine Sanitäranlage aus der Liste der in der Nähe befindlichen Geräte aus. 4. Die Applikation verbindet sich mit der Sanitäranlage. 5. Übersicht über die verbundene Sanitäranlage wird angezeigt.
Extensions	2a. Applikation zeigt keine verfügbaren Zielgeräte an. 1. Benutzer wählt den Vorführmodus aus.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.1: UC1: Mit Zielgerät verbinden

A.2.2 UC2: Zielgeräteinformationen auslesen

Description	Ein Servicetechniker liest Informationen aus einem verbundenen Zielgerät aus.
Primary Actor	Servicetechniker
Trigger	• Servicetechniker möchte prüfen, ob neue Firmware Aktualisierungen für das Zielgerät verfügbar sind. • Servicetechniker möchte prüfen, ob das Zielgerät einen Fehlercode sendet.
Stakeholder and Interests	Servicetechniker: Ein Servicetechniker möchte über den aktuellen Zustand eines Zielgeräts informiert sein.
Preconditions	Die Applikation hat sich mit einem Zielgerät verbunden oder befindet sich im Vorführmodus.
Main Success Scenario	1. Der Servicetechniker wählt eine Informations Kategorie aus. 2. Die Applikation zeigt die entsprechenden Informationen an.
Extensions	2a. Die Verbindung zum Zielgerät ist nicht mehr möglich. 1. Die Applikation weist auf Verbindungsprobleme hin. 2. Die Applikation zeigt eine Übersicht der verfügbaren Geräte an.

Tabelle A.2: UC2: Zielgeräteinformationen auslesen

A.2.3 UC3: Fehlersuche durchführen

Description	Ein Servicetechniker liest Fehlercodes aus dem Zielgerät aus oder analysiert sie anhand von Fehlerbildern.
Primary Actor	Servicetechniker
Trigger	• Das Zielgerät sendet einen Fehlercode. • Das Zielgerät zeigt ein nicht erwünschtes Verhalten, welches jedoch nicht auf eine durch einen Fehlercode angezeigte Fehlerursache zurückzuführen ist.
Stakeholder and Interests	Servicetechniker: Will alle Lösungsvorschläge zu einem bestimmten Fehler erhalten.
Preconditions	Die Applikation hat sich mit einem Zielgerät verbunden oder befindet sich im Vorführmodus.
Main Success Scenario	1. Benutzer wählt die Fehlersuche nach Fehlercode aus. 2. Applikation zeigt eine Liste der von dem Zielgerät gesendeten Fehlercodes an. 3. Benutzer wählt einen Fehlercode aus. 4. Applikation zeigt Lösungsvorschläge zum Fehlercodes.
Extensions	2a. Die Applikation zeigt keine Fehlercodes an. 1. Benutzer wählt die Fehlersuche nach Fehlerbild aus. 2. Applikation zeigt eine Liste mit bekannten Fehlerbildern an. 3. Benutzer durchsucht die Liste nach geeignetem Fehlerbild und wählt diesen aus. 4. Applikation zeigt Lösungsvorschläge zum Fehlerbild.
Frequency of Occurrence	Mehrmals pro Tag

Tabelle A.3: UC3: Fehlersuche durchführen

A.2.4 UC4: Firmware aktualisieren

Description	Ein Servicetechniker stellt fest, ob eine neue Firmware für das gerade verbundene Zielgerät verfügbar ist. Falls dies der Fall ist, spielt er die aktuellere Version der Firmware auf das Zielgerät auf.
Primary Actor	Servicetechniker
Trigger	Neue Firmware ist verfügbar.
Stakeholder and Interests	Servicetechniker: Möchte das Zielgerät auf dem aktuellen Stand haben.
Preconditions	• Die Applikation befindet sich im Vorführmodus oder hat sich mit einem Zielgerät verbunden. • Neue Firmware ist verfügbar.
Postconditions	• Es ist keine neue Firmware verfügbar. • Zielgerät ist auf aktuellem Stand.
Main Success Scenario	1. Benutzer verbindet sich mit Zielgerät. 2. Applikation zeigt neue verfügbare Firmwares an. 3. Benutzer startet die Aktualisierung. 4. Applikation führt Firmware Aktualisierung durch.

Tabelle A.4: UC4: Firmware aktualisieren

A.2.5 UC5: Verbindung mit Zielgerät trennen

Description	Ein Servicetechniker trennt die Verbindung zum Zielgerät.
Primary Actor	Servicetechniker
Trigger	• Der Servicetechniker möchte sich mit einem anderen Zielgerät verbinden. • Der Servicetechniker hat die Arbeiten am Zielgerät abgeschlossen.
Stakeholder and Interests	Servicetechniker: Möchte die Verbindung zum Zielgerät aufheben.
Preconditions	Die Applikation ist mit einem Zielgerät verbunden.
Postconditions	Die Applikation ist mit keinem Zielgerät verbunden.
Main Success Scenario	1. Benutzer wählt den Befehl die Verbindung zu trennen. 2. Die Applikation trennt die aktuelle Verbindung zum Zielgerät. 3. Die Applikation zeigt an, welche Zielgeräte sich in der Nähe. befinden

Tabelle A.5: UC5: Verbindung mit Zielgerät trennen

A.2.6 UC6: CRUD Servicemitarbeiter

Description	Ein Administrator bearbeitet die Daten von Servicetechnikern.
Primary Actor	Administrator
Trigger	• Ein Servicetechniker möchte sich bei der Applikation anmelden und er ist noch nicht erfasst. • Die Informationen eines Servicetechnikers ändern sich und müssen angepasst werden.
Stakeholder and Interests	Servicetechniker: Möchte die Applikation verwenden. Administrator: Möchte wissen, wer Zugriff auf die Applikation hat. Kunde: Möchte bestehende Servicetechniker einfach in das Backend übernehmen und bearbeiten können.
Main Success Scenario	1. Ein Administrator verbindet sich mit dem Backend. 2. Er aktualisiert die Daten eines Servicemitarbeiters.

Tabelle A.6: UC6: CRUD Servicemitarbeiter

A.2.7 UC7: CRUD Applikationsdaten

Description	Ein Administrator möchte die Applikationsdaten aktualisieren.
Primary Actor	Administrator
Trigger	Die Applikationsdaten wurden überarbeitet.
Stakeholder and Interests	Administrator: Möchte neue Applikationsdaten zur Verfügung stellen. Servicetechniker: Möchte immer die aktuellsten Applikationsdaten zur Verfügung haben. Kunde: Möchte die bestehenden Applikationsdaten möglichst einfach übernehmen und verwalten können.
Main Success Scenario	1. Ein Administrator verbindet sich mit dem Backend. 2. Er aktualisiert die Applikationsdaten.

Tabelle A.7: UC7: CRUD Applikationsdaten

A.3 Systemtest Detail

A.3.1 ST1: Starten der Applikation

Testziel	Der Applikationsstart wird wie definiert ausgeführt.
Vorbedingungen	Keine
Durchführung	1. Starten der Applikation
Erwartetes Ergebnis	Nach dem Start wird der Splashscreen mit dem Geberit-Logo in der Mitte des Bildschirms angezeigt.

Tabelle A.8: ST1: Starten der Applikation

A.3.2 ST2: Authentifizieren

Testziel	Der Benutzer kann sich mit seinen Benutzerinformationen bei der Applikation nur auf einem Gerät authentifizieren.
Vorbedingungen	• Der Benutzer verfügt über ein Benutzerkonto, welches zuvor von einem Administrator erstellt wurde. • Gerät A: Der Benutzer ist erfolgreich authentifiziert und die Applikation ist nicht gestartet. • Gerät B: Kein Benutzer ist authentifiziert und die Applikation ist gestartet. • Beide Geräte haben eine aktive Internetverbindung.
Durchführung	1. Der Benutzer meldet sich beim Gerät B mit den gleichen Benutzerinformationen an, wie zuvor bei Gerät A. 2. Der Benutzer startet die Applikation auf Gerät A.
Erwartetes Ergebnis	Der Benutzer ist auf Gerät B erfolgreich authentifiziert. Auf Gerät A ist kein Benutzer authentifiziert.

Tabelle A.9: ST2: Authentifizieren

A.3.3 ST3: Mit Vorführmodus verbinden

Testziel	Die Applikation kann sich mit dem Vorführmodus verbinden, ohne dass sich ein Zielgerät in der Nähe befindet.
Vorbedingungen	• Die Applikation ist gestartet und mit einem gültigen Benutzer authentifiziert. • Die Applikation ist mit keinem Gerät verbunden.
Durchführung	1. Der Benutzer ruft die "Verfügbare Verbindungen"-Seite auf. 2. Der Benutzer betätigt den "Scan"-Button. 3. Der Benutzer klickt auf den Vorführmodus.
Erwartetes Ergebnis	Der Benutzer sieht die Geräteinformationen des Vorführmodus. Zusätzlich ist ersichtlich, dass Firmware Aktualisierungen verfügbar sind, aktive Fehlercodes gesendet werden und der Vorführmodus aktiv ist.

Tabelle A.10: ST3: Mit Vorführmodus verbinden

A.3.4 ST4: Fehlerbilder im Vorführmodus anzeigen

Testziel	Der Benutzer kann alle Fehlerbilder des Vorführmodus durchsuchen.
Vorbedingungen	• ST3 erfolgreich durchgeführt. (siehe A.3.3)
Durchführung	1. Der Benutzer ruft die “Geräte-Informationen”-Seite auf. 2. Der Benutzer wechselt auf die “Fehlerbilder”-Seite.
Erwartetes Ergebnis	Der Benutzer sieht alle Fehlerbilder. Zusätzlich sind für jedes Fehlerbild, Ursachen und Massnahmen ersichtlich. Es ist ersichtlich, dass der Vorführmodus aktiv ist.

Tabelle A.11: ST4: Fehlerbilder im Vorführmodus anzeigen

A.3.5 ST5: Fehlercodes im Vorführmodus anzeigen

Testziel	Der Benutzer kann alle Fehlercodes des Vorführmodus durchsuchen.
Vorbedingungen	• ST3 erfolgreich durchgeführt. (siehe A.3.3)
Durchführung	1. Der Benutzer ruft die “Geräte-Informationen”-Seite auf. 2. Der Benutzer wechselt auf die “Fehlercodes”-Seite
Erwartetes Ergebnis	Der Benutzer sieht alle möglichen Fehlercodes. Zusätzlich sind für jeden Fehlercode, Ursachen und Massnahmen ersichtlich. Es ist ersichtlich, dass der Vorführmodus aktiv ist.

Tabelle A.12: ST5: Fehlercodes im Vorführmodus anzeigen

A.3.6 ST6: Firmware im Vorführmodus aktualisieren

Testziel	Der Benutzer kann die mögliche Firmware Aktualisierung des Vorführmodus durchführen.
Vorbedingungen	• ST3 erfolgreich durchgeführt. (siehe A.3.3) • Es ist ersichtlich, dass Firmware Aktualisierungen verfügbar sind.
Durchführung	1. Der Benutzer ruft die “Geräte-Informationen”-Seite auf. 2. Der Benutzer wechselt auf die “Software”-Seite. 3. Der Benutzer startet die Firmware Aktualisierung.
Erwartetes Ergebnis	Der Benutzer sieht, dass Firmware Aktualisierungen vorhanden sind. Zusätzlich wird der Fortschritt der einzelnen Funkkomponenten angezeigt. Es ist ersichtlich, dass der Vorführmodus aktiv ist.

Tabelle A.13: ST6: Firmware im Vorführmodus aktualisieren

A.3.7 ST7: Verbindung mit Vorführmodus trennen

Testziel	Der Benutzer kann Verbindung mit dem Vorführmodus trennen.
Vorbedingungen	• ST3 erfolgreich durchgeführt. (siehe A.3.3)
Durchführung	1. Der Benutzer wählt “Trennen”.
Erwartetes Ergebnis	Der Benutzer sieht, dass er mit keinem Zielgerät verbunden ist.

Tabelle A.14: ST7: Verbindung mit Vorführmodus trennen

A.3.8 ST8: Benutzer erfassen

Testziel	Ein Administrator kann einen neuen Benutzer erfassen.
Vorbedingungen	• Der Administrator ist an der Benutzerverwaltung authentifiziert.
Durchführung	1. Der Administrator erfasst den Benutzer in der Benutzerverwaltung. 2. Der Administrator weist dem erstellten Benutzer die entsprechenden Berechtigungen zu.
Erwartetes Ergebnis	Der Benutzer kann sich mit den Benutzerinformationen in der Applikation authentifizieren.

Tabelle A.15: ST8: Benutzer erfassen

A.3.9 ST9: Benutzer löschen

Testziel	Ein Administrator kann einen Benutzer löschen.
Vorbedingungen	• Der Administrator ist an der Benutzerverwaltung authentifiziert. • Der Benutzer ist in der Benutzerverwaltung erfasst.
Durchführung	1. Der Administrator löscht den Benutzer in der Benutzerverwaltung.
Erwartetes Ergebnis	Applikationen, welche mit dem gelöschten Benutzer authentifiziert sind, werden ausgeloggt. Der gelöschte Benutzer kann sich nicht mehr authentifizieren.

Tabelle A.16: ST9: Benutzer löschen

A.3.10 ST10: Applikationsdaten bearbeiten

Testziel	Ein Administrator kann Applikationsdaten bearbeiten.
Vorbedingungen	• Der Administrator ist an der Applikationsdatenverwaltung authentifiziert.
Durchführung	1. Der Administrator bearbeitet eine Applikationsdatei.
Erwartetes Ergebnis	Die Veränderungen der Applikationsdatei sind in der Applikation ersichtlich.

Tabelle A.17: ST10: Applikationsdaten bearbeiten

A.3.11 ST11: Sprache auswählen

Testziel	Ein Benutzer kann die Sprache der Applikation ändern.
Vorbedingungen	Keine
Durchführung	1. Der Benutzer ruft die “Sprache“-Seite auf. 2. Der Benutzer wählt die Sprache “Englisch“ aus.
Erwartetes Ergebnis	Die Applikation ist nun in “Englisch“ übersetzt.

Tabelle A.18: ST11: Sprache auswählen

A.3.12 ST12: Hilfe, Impressum, Rechtshinweise und Datenschutzerklärungen aufrufen

Testziel	Ein Benutzer kann Hilfe, Impressum, Rechtshinweise und Datenschutzerklärungen aufrufen.
Vorbedingungen	Keine
Durchführung	1. Der Benutzer ruft die entsprechende Seite auf.
Erwartetes Ergebnis	Die Applikation zeigt die entsprechende Seite an.

Tabelle A.19: ST12: Hilfe, Impressum, Rechtshinweise und Datenschutzerklärungen aufrufen

A.4 Projektmanagement Detail

A.4.1 Gantt-Diagramm

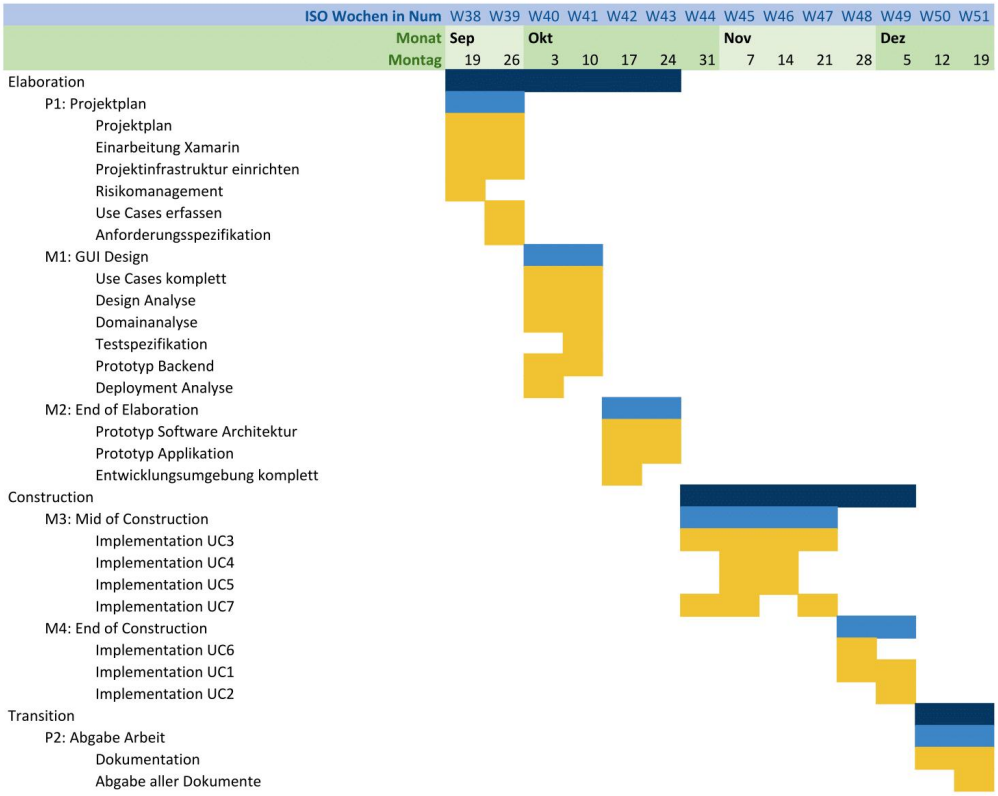

Abbildung A.1: Gantt-Diagramm

A.5 Risikomanagement Detail

A.5.1 Risikoanalyse der Elaboration-Phase

Risiko	**max.Schaden [h]**	**Eintrittswahrscheinlichkeit [%]**	**Gewichteter Schaden [h]**
R1	20	30	6
R2	10	10	1
R3	20	20	4
R4	5	5	0.25
R5	40	60	24
R6	10	10	1
R7	20	5	1
R8	40	40	16
Total	**165**		**53.25**

Tabelle A.20: Risikoanalyse Elaboration

A.5.2 Risikoanalyse nach der Elaboration-Phase

Risiko	**max.Schaden [h]**	**Eintrittswahrscheinlichkeit [%]**	**Gewichteter Schaden [h]**
R1	20	15	3
R2	10	10	1
R3	10	10	1
R4	5	5	0.25
R5	20	20	4
R6	10	0	0
R7	20	5	1
R8	40	20	8
Total	**135**		**18.25**

Tabelle A.21: Risikoanalyse Construction