HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

FireTabletPlus

Mobiles Service Interface zu Siemens Brandmeldeanlagen

Bachelorarbeit HS 2011

Ramon Müller & Samuel Hüppi

IFS INSTITUTE FOR SOFTWARE

Erklärung der Selbstständigkeit

Hiermit versichern wir, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Rapperswil, den 23.12.2011 Samuel Hüppi Ramon Müller

Aufgabenstellung

Moderne Brandmeldezentralen schützen Personen und Gebäude mit Hilfe modernster Technik. Hochoptimierte Rauchmelder unterscheiden zuverlässig Störgrössen von Bränden und alarmieren die Bewohner und die Feuerwehr oft bevor es überhaupt zu einem nennenswerten Schaden kommt. Siemens ist ein weltweit führender Hersteller solcher Anlagen und bietet in vielen Ländern sowohl die Produkte als auch entsprechende Serviceleistungen an.

Ein wichtiger Teil der Serviceleistungen ist die periodisch Wartung installierter Anlagen um sicherzustellen, dass diese stets zuverlässig funktionieren. Im Rahmen dieser Wartung werden die Rauchmelder auf korrekte Funktion geprüft und die Installation der Anlage kontrolliert.

Unser Ziel ist es immer die besten Produkte und Serviceleistungen anzubieten. Dazu gehört auch eine entsprechende Toolunterstützung für den Servicetechniker. Anhand eines portablen Gerätes mit Touchscreen, Kamera, Audiorekorder und Internetzugang (z.B. iPhone, iPad, Android Handy, etc.) sollen im Rahmen der Studie Benutzeroberflächen und Workflow-Konzepte erarbeitet werden mit denen der Service schneller und zuverlässiger erfolgen kann. Der Servicetechniker soll mit der geplanten Applikation in der Lage sein sowohl direkt mit der Brandmeldeanlage zu interagieren (aktuelle Statusinformationen, Bedienung, etc.) als auch seine Tätigkeit lückenlos zu dokumentieren (Fotos, Sprachkommentare). Nach Abschluss seiner Tätigkeit soll der Report direkt in interaktiver Form verfügbar sein.

Das Ziel ist es zum Abschluss der Studie eine lauffähige Applikation zu haben mit denen die wichtigsten Aspekte in Bezug auf UI und Workflow demonstriert werden können. Die Einbindung in bestehende Infrastruktur (Brandmeldesystem, IT Landschaft) spielt eine untergeordnete Rolle und kann auch simuliert werden.

Voraussetzungen:
Android, ggf. iPhone/iPad Entwicklung
Java, resp. Objective-C
Eclipse resp. XCode
Testautomation, Build-Server, etc., wie in SE Modulen gelernt
Netzprotokolle (HTTP, REST, etc)

Vorrang liegt bei Android (einfachere Umgebung, leichter Code auf Endgerät einspielbar) iPhone/iPad nur auf Wunsch d. Studenten.

Abstract

Ausgangslage: Siemens Brandmeldeanlagen sind komplexe Systeme, die nicht nur bei der Installation einen sehr hohen Arbeitsaufwand verursachen, sondern auch bei der Instandhaltung und jährlichen Revisionen viel Zeit in Anspruch nehmen. Aus diesem Grund versucht Siemens die dazu nötigen Arbeitsabläufe zu generalisieren und durch technische Hilfsmittel zu unterstützen.Zu diesem Zweck soll das in einer Studienarbeit erstellte Android- Programm ≪FireTablet≫ erweitert und verbessert werden.

Vorgehen/Technologien: Die vorhandene Software soll in einem ersten Schritt auf Verbesserdungspotential und Erweiterungsmöglichkeiten geprüft werden. Dabei wird in Zusammenarbeit mit unserem Ansprechpartner bei Siemens, gezielt auf die Wünsche des Unternehmens eingegangen. Die Entwicklung konzentriert sich auf die Optimierung des grafischen Interfaces sowie die Integration der Applikation in den Workflow zur Qualitätssicherung von Brandmeldeanlagen. Um diese Ziele zu erreichen wird weiterhin auf eine Tabletlösung, basierend auf Android, gesetzt. Für die zentrale Datenablage kommt ein Webserver zum Einsatz.

Ergebnis: Mit FireTabletPlus werden Ideen aufgezeigt, die den Arbeitsablauf der regelmässigen Überprüfungen von Brandmeldeanlagen effizienter machen. Die zentral gespeicherten Daten einer Anlage können direkt auf das Tablet heruntergeladen werden, wodurch früher erfasste Kommentare und Fehler einsehbar sind. Während des Tests der Anlage unterstützt FireTabletPlus den Servicetechniker durch eine übersichtliche Darstellung der Anlagenstruktur. Feuermelder können auf dem Gebäudeplan markiert und fokusiert werden. Anschliessend können Testergebnisse wieder zentral gespeichert, sowie Testrapporte automatisch generiert und versendet werden. Mit FireTabletPlus können Brandmeldeanlagen sicherer, effizienter und strukturierter getestet werden.

Management Summary

Ausgangslage

Moderne Brandmeldeanlagen schützen Personen und Gebäude, indem sie zuverlässig und frühzeitig Anzeichen von Bränden erkennen und Bewohner und Feuerwehr alarmieren, bevor es zu nennenswerten Schäden kommt. Damit sie dies garantiert erledigen können, ist eine regelmässige Kontrolle und Wartung nötig. Ein Servicetechniker untersucht eine Anlage und deren Zentralen in vielfältigen Aspekten auf Herz und Nieren. Unter anderem werden an die Zentrale angehängte Brandmelder mittels eines sogenannten Prüfpflückers auf Funktionstauglichkeit getestet. Das Gerät überprüft einerseits, ob der Sensor des Brandmelders einwandfrei funktioniert und sendet andererseits einen Probealarm an die Zentrale, um die Leitung zu testen.

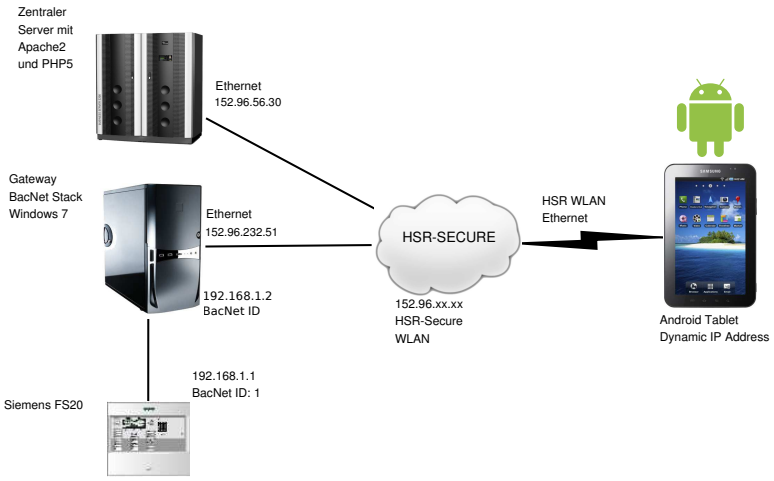
Prüfpflücker zum Testen von Brandmeldern

Um die vielen erfassten Daten konsequent fest zu halten, werden verschiedene Mittel eingesetzt. Zum Beispiel werden Fehler in einer Exceltabelle eingetragen oder erfasste Bilder ausgedruckt. Es fehlt aber ein Mittel das sich konsequent in den Arbeitsablauf der Überprüfung von Brandmeldeanlagen integriert. Erste Probleme treten zum Beispiel auf, wenn ein Anderer Servicetechniker weiter testen will, denn der Neue Techniker weiss nicht was der Vorgänger schon getestet hat. Einige dieser Probleme werden von der Vorgängersoftware FireTablet angegangen oder gelöst. So konnte man zum Beispiel die gesamte Anlagenhierarchie auf dem Tablet betrachten. Das Userinterface war allerdings unübersichtlich und nicht intuitiv. Für die Weiterentwicklung wurden darum folgende Ziele definiert:

- Benutzerfreundliche Bedienung mit übersichtlicher Darstellung.
- Erfasste Daten können direkt elektronisch weiter verwendet werden.
- Anlagepläne können auf dem Tablet betrachtet werden.

Vorgehen

Die Applikation FireTablet wurde zuerst gründlich nach Verbesserungspotential untersucht. Anschliessend konnten in Zusammenarbeit mit Siemens die möglichen Verbesserungen priorisiert und detailliert besprochen werden. Durch die Gespräche mit Siemens hat sich gezeigt, dass FireTabletPlus sich besser in den Workflow der Firma integrieren muss und eine ansprechende Benutzerschnittstelle zu bieten hat. Die bessere Integration wurde durch den Einsatz eines zentralen Servers erreicht der sämtliche Daten für eine Brandmeldeanlage verwaltet.



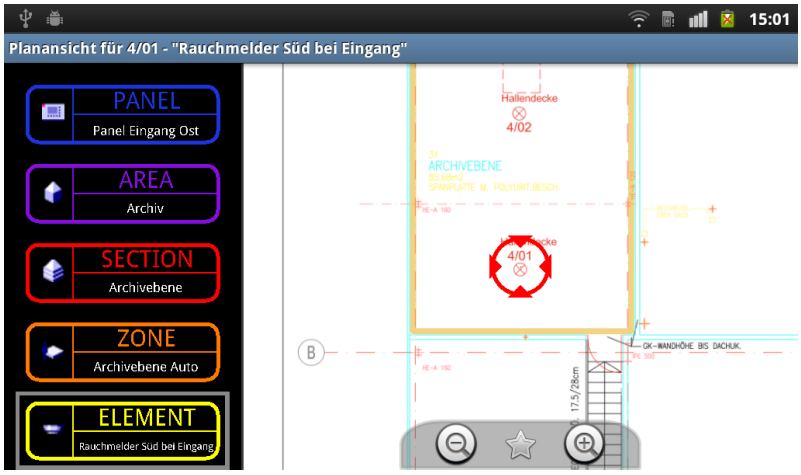
Vorhandene Infrastruktur

In das neue Design des Userinterface wurde viel Zeit investiert um eine möglichst optimale Platzausnutzung in Verbindung mit Übersichtlichkeit zu erreichen. Aus verschieden Prototypen wurde eine Lösung erarbeitet die sowohl die Navigation durch die Brandmeldeanlage-Hierarchie optimal unterstützt, als auch alle interessanten Informationen auf einen Bildschirm vereint.

Brandmeldeanlage in der Übersicht.

Ergebnisse/Ausblick

Die Applikation unterstützt Servicetechniker optimal bei jährlichen Kontrollen von Brandmeldeanlagen. Der Techniker kann bereits in der Firma die komplette Konfiguration einer Brandmeldeanlage auf sein mobiles Gerät laden. Sollte die Anlagestruktur noch nicht erfasst worden sein, kann sie über den Gateway auf dem Tablet ergänzt werden. Fehler können anschliessend in gut strukturierten Schritten zu jedem Gerät erfasst werden und das Tablet unterstützt die Arbeit des Technikers, indem jeder Zeit die Positionen der Geräte, auf einem Plan angezeigt werden können.



Plananzeige mit markiertem Brandmelder

Nach abgeschlossener Inspektion wird ein Inspektionsreport generiert der dem Kunden via Mail direkt zugestellt werden kann. Auch der interne Arbeitsablauf wird weiter unterstützt. Die Konfiguration mit den neu erstellten Fehlern kann wieder an den zentralen Server geschickt werden wo sie gespeichert wird und für den nächsten Benutzer bereit steht.
Mit diesem System ist eine gute Grundlage gelegt um eine Software zu entwickeln die im produktiven Umfeld verwendet werden kann. Allerdings fehlen noch wesentliche Teile wie Kommunikations- und Datensicherheit.

(Quelle oberster Abschnitt: [BA FireTablet])

Inhaltsverzeichnis

1 Einführung

FireTabletPlus ist eine Folgearbeit der [BA FireTablet]. FireTablet wurde im Rahmen einer Semester- und Bachelorarbeit geschrieben. Aus diesem Grund werden in diesem Kapitel zuerst Ausgangslage, Heutiger Arbeitsablauf, Problemstellung und Ziele der Vorgängerarbeit wiederholt da die Problemdomäne gleich geblieben ist. Im Abschnitt Ausgangslage FireTablet wird speziell auf die Ziele der Weiterentwicklung von FireTablet eingegangen. Ganz am Schluss dieses Kapitels wird die Basis für den Umgang mit Android gelegt. In diesem Bereich werden wichtige Begriffe zum Framework erklährt.

1.1 Ausgangslage

Quelle: [BA FireTablet]
Bei Siemens Brandmeldeanlagen handelt es sich um automatische Systeme, welche Brände automatisch erkennen, sowie alle nötigen Aktionen einleiten. Solche Aktionen könnten z.B. alle Lüftungen ausschalten, Brandschutzklappen schliessen oder Lifte deaktivieren. Brandmeldeanlagen sind also komplizierte Anlagen, die über ganze Gebäudekomplexe installiert und zentral verwaltet werden.

Sowohl Inbetriebnahme als auch Wartung solcher Anlagen sind aufwändige Prozesse und erfordern von den Servicetechnikern viel Konzentration und Zeit.
Viele Anforderungen an solche Brandmeldesysteme werden nicht direkt von Siemens erlassen, sondern sind von den jeweiligen Staaten und ihren Normen vorgeschrieben. Die jährliche Überprüfung sämtlicher Richtlinien muss sehr gründlich vorgenommen werden, um erstens sicher zu stellen, dass die Anlage ordnungsgemäss funktioniert, und zweitens zur rechtlichen Absicherung, falls es zu Unfällen kommt. Der Ersteller solcher Anlagen ist somit sehr daran interessiert, eine gute Dokumentation der Wartungs- und Inbetriebnahmearbeiten vorweisen zu können.
Um die Probleme um den bestehenden Arbeitsprozess zu verdeutlichen, wird im folgenden der Ist-Zustand genauer beschrieben.

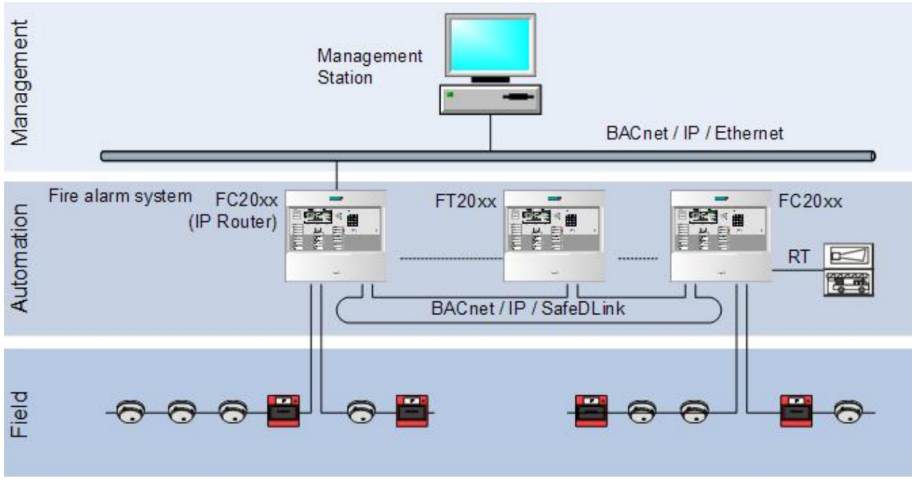


Abbildung 1.1: Hierarchische Struktur von Brandmeldeanalgen. Die Anlagen können über mehrere Gebäude verteilt sein.

1.2 Heutiger Arbeitsablauf

Quelle: [BA FireTablet]
Die heutige Arbeitsweise sieht zwei verschiedene Prozesse vor: Kommissionierung oder Wartung einer Anlage. Die Kommissionierung ist die Inbetriebnahme einer Brandmeldeanlage, und die Wartung ist eine Kontrolle, die jedes Jahr durch einen Siemens Servicetechniker durchgeführt wird. In beiden Fällen stehen dem Servicetechniker folgende Hilfsmittel zur Verfügung:

- **Panel**
  Eine Bedienschnittstelle, welche sich entweder direkt an der Zentrale oder an einer abgesetzten Bedieneinheit befindet. Ein Panel besitzt ein monochromes Display und ist mit diversen Tasten zu bedienen.

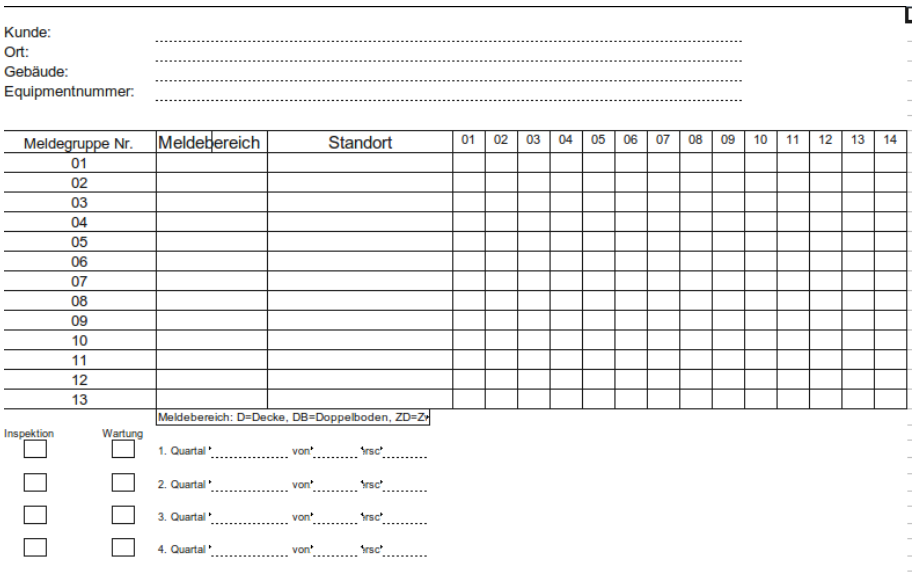
Abbildung 1.2: Bedieneinheit einer FS20 Brandmeldezentrale

- **Prüfplücker**
  Mit diesem Testgerät können Testalarme auf dem Brandmelder ausgelöst werden. Der spezielle Name kommt von der Art der Tätigkeit, wenn der Kontrolleur die Brandmelder, die normalerweise an der Decke hängen, mit einer langen Stange prüft. Bei jedem Prüfvorgang wird der Pflücker an den Melder angedockt und löst auto-

matisch den Test aus. Über drei Leuchtdioden wird dem Techniker angezeigt, ob der Test erfolgreich war oder nicht.

Abbildung 1.3: Prüfplücker um Brandmelder auf korrekte Funktion zu testen.

- **Checklisten**
  Damit der Arbeitsablauf die nötige Struktur hat und Resultate richtig notiert werden, bekommt der Servicetechniker eine ganze Auswahl an Listen, die im Verlauf einer Wartung oder Kommissionierung auszufüllen sind. Mit diesen Listen werden am Schluss die Prüfdokumente erstellt, sowie Rechnungen an den Kunden geschrieben. Werden Mängel festgestellt, muss der Techniker zusätzlich Ersatzmaterial bestellen. Das Bild zeigt eine Liste, die bei Brandmeldertests zum Einsatz kommt. Wie unschwer zu erkennen ist, reicht der Platz für mehr Informationen als "ok" / "fehlerhaft" nicht aus.



Kunde: .....
Ort: .....
Gebäude: .....
Equipmentnummer: .....

Meldegruppe Nr.	Meldebereich	Standort	01	02	03	04	05	06	07	08	09	10	11	12	13	14
01																
02																
03																
04																
05																
06																
07																
08																
09																
10																
11																
12																
13																

Meldebereich: D=Decke, DB=Doppelboden, ZD=Z

Inspektion Wartung

1. Quartal ..... von ..... 'rsc' .....
2. Quartal ..... von ..... 'rsc' .....
3. Quartal ..... von ..... 'rsc' .....
4. Quartal ..... von ..... 'rsc' .....

Abbildung 1.4: Altes listenbasierendes Prüfprotokoll. Quelle: [Pruefprotokoll]

Bei Kommissionierung und Wartung sind folgende Arbeiten zu erledigen (Quelle: [Pruefprotokoll]):

- **Sichtprüfung**
  - Dokumentation und Betriebsbuch
  - Kontakte und Material in der Zentrale
  - Verschmutzung
  - Mindestabstand bei Melder
  - Akku Beschädigung
- **Funktionsprüfung**
  - Meldergruppen
  - Steuerungen der Anlage
  - Meldungen
- **Messung**
  - Akkuleistung
  - Netzversorgung
  - Spannung auf Meldergruppe

Im Unterschied zur Kommissionierung sind bei der Wartung nicht alle Brandmelder mit dem Prüfpflücker zu testen, sondern nur jeweils ein automatischer Melder pro Melderlinie. Handtaster müssen allerdings alle mit einem speziellen Schlüssel betätigt werden, damit nicht jedes mal die Scheibe eingedrückt wird, nur um einen Test durchzuführen.
Bei beiden Testarten ist der Kundentext, der am Panel angezeigt wird, sehr wichtig. Dieser Text zeigt der Feuerwehr, an welchem Ort ein Melder ausgelöst wurde. Darum muss mindestens bei der Komissionierung überprüft werden, ob auch der richtige Kundentext zum richtigen Alarm angezeigt wird. Auf dem Bild sieht man, wie eine Alarmmeldung bei der Zentrale eintrifft. Hierbei wird der Melder, der einen Alarm ausgelöst hat, sowie die gesamte Hierarchie der Anlage, in welcher sich der Melder befindet, angezeigt.

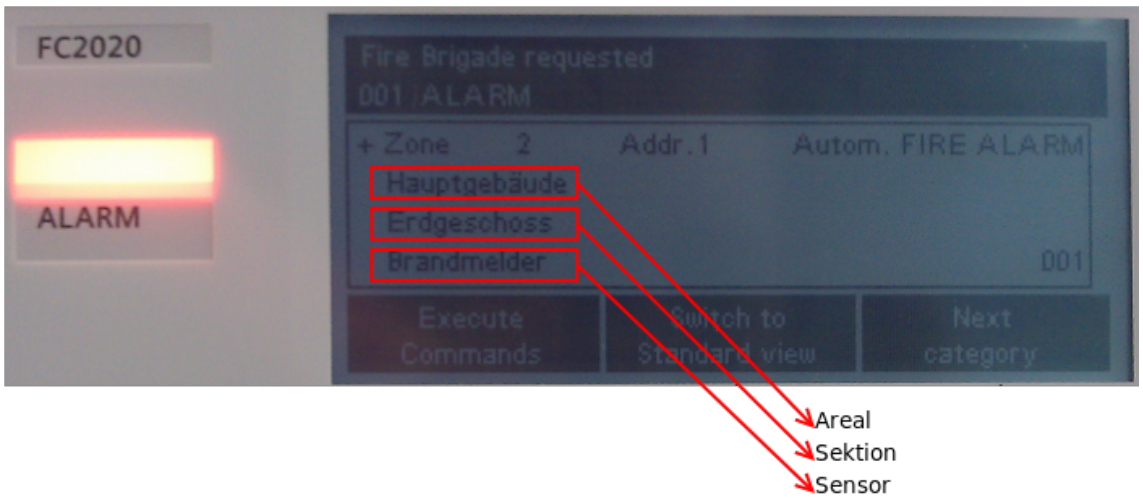


Abbildung 1.5: Anzeige eines Alarms auf dem Panel einer FC2020 Brandmeldeanlage.

1.3 Problemstellung

Quelle: [BA FireTablet]
Die Wartung und Inbetriebnahme von Brandmeldeanlagen ist ein aufwändiger Prozess und muss sicherstellen, dass wirklich alles so funktioniert wie es der Test ergeben hat. Folgende Probleme oder Verbesserungen können durch neue Methoden und Geräte in Angriff genommen werden, damit die Überprüfung von Brandmeldeanlagen auch in Zukunft auf dem aktuellen Stand der Technik abläuft:

1. **Alle Mängel werden in Listen auf Papier eingetragen.** Die Listen müssen nach dem Test von Hand in Exceltabellen übertragen werden. Dieses System ist fehleranfällig, die Listen sind unübersichtlich, falls mehr Details eingetragen werden müssen und es entsteht unnötiger Zeitaufwand beim Übertragen der Resultate.

2. **Das Panel ist nicht sichtbar, wenn ein Test ausgelöst wird.** Es kann nicht sofort überprüft werden, ob Testalarme, die an Brandmeldern oder Handtastern ausgelöst werden, wirklich in der Zentrale mit korrektem Kundentext eintreffen, da man keine Verbindung zur Zentrale hat während man einen Testalarm auslöst. Erst wenn der Servicetechniker das Eventlog auf der Zentrale betrachtet, weiss er, ob die korrekten Sensoren mit der korrekten Standortbeschreibung ausgelöst wurden.

3. **Wenig Platz zur Mangelerfassung auf Papier.** Mängel können auf Excellisten nur sehr einfach erfasst werden oder es ist schwierig, einen Mangel zu erfassen, wenn der Sensor an schlecht zugänglichen Stellen montiert ist.

4. **Keine zeitgemässen Schnittstellen für die Weiterverarbeitung.** Die Weiterverarbeitung von erfassten Tests ist zur Zeit sehr schwierig, da sie nicht elektronisch erfasst werden. Die Person, die einen Auftrag z.B. für eine Reparatur oder Erweiterung einer Anlage kalkuliert, kann sich kaum ein Bild der vorhandenen Lage machen.

5. **Unhandliches Arbeitsgerät Laptop.** Der zur Zeit verfügbare Laptop, den Servicemonteure bei sich haben, ist zu wenig mobil, um ihn immer mit sich zu tragen und alle Testergebnisse direkt einzugeben. Gleichzeitig entsteht ein Diebstahlproblem, wenn der Laptop bei der Zentrale stehen gelassen wird, da sich Kommissionierungen oft noch auf der Baustelle abspielen.

1.4 Ziele

Quelle: [BA FireTablet]
Um Verbesserungen in die Kontrollprozesse von Brandmeldeanlagen zu bringen, möchte Siemens Building Technologies Schweiz einen computergestützten Arbeitsablauf etablieren, der auch einen internationalen Standard bietet. Zum Einsatz soll ein Tabletcomputer kommen, welcher die Mobilität besitzt, ihn stets mit sich zu tragen, aber trotzdem eine komfortable Bedienung mit grossem Bildschirm ermöglicht. Sowohl in normalen als auch in speziellen Situationen (der Servicetechniker steht z.B. auf einer Leiter und hat nur eine Hand frei) ist eine gute Bedienung möglich. Auf dem Tablet soll eine Android Applikation laufen, die den Servicetechniker bei den Wartungsarbeiten unterstützt.
Damit die Brandmeldezentrale mit dem Tablet kommunizieren kann, stellt Siemens ein

Gateway zur Verfügung. Das Gateway und das Tablet sind über WLAN miteinander verbunden und kommunizieren über HTTP. Das Gateway tauscht über Ethernet mit der Brandmeldezentrale Daten aus und kommuniziert über das Gebäudeautomatisierungsprotokoll BACnet. Beim Design der Anwendung soll darauf geachtet werden, die Bedienbarkeit auch in schwierigem Umfeld zu ermöglichen, indem ein intuitieves Interfacekonzept verwendet wird, oder Fehler auch in Form von Voicememos gespeichert werden können.

1. **Zu allen Meldern oder Zonen Fehler erfassen.** Dies soll in Form von Bildern, Voicememo und Text möglich sein. Die Fehlerliste soll in digitaler Form verfügbar sein, damit sie bereit ist für die weitere elektronische Verarbeitung, Archivierung oder Protokollgenerierung. Bei allen Geräten kann nachgeschaut werden, welche Fehler dafür erfasst wurden und es ist ersichtlich, in welcher Zone wie viele Fehler vorhanden sind.

2. **Konzept ermöglicht ständige Verfügbarkeit des Tablets.** Damit der Servicemonteur sein Tablet möglichst ohne Aufwand immer bei sich tragen kann, soll das Userinterface im Querformat erstellt werden, damit das Tablet an den Arm des Servicetechnikers montiert werden kann, um möglichst hohe Bewegungsfreiheit und Verfügbarkeit zu gewährleisten. Eventuell kann auch eine Halterung für das Tablet am Arm des Technikers befestigt werden.

3. **Das Tablet führt den Techniker durch den Test.** Das Tablet unterstützt den Servicetechniker und hilft ihm dabei, die Tests Schritt für Schritt durchzuführen. Dies wäre dann ein Ersatz für die heute noch eingesetzten Checklisten.

1.5 Ausgangslage FireTablet

FireTablet wurde von Daniel Bobst als Bachelorarbeit entwickelt und von Samuel Hüppi als Semesterarbeit. Die aktuelle Arbeit FireTabletPlus wird als Bachelorarbeit von Ramon Müller und Samuel Hüppi weiter entwickelt. Mit der Folgearbeit werden aus der technischen Sicht die gleichen Ziele, wie oben beschriebn, verfolgt. Praktisch liegt der Fokus bei FireTabletPlus mehr auf der Erstellung einer Demonstrationssoftware, die von Siemens dazu benutzt werden kann eine mögliche Richtung der Entwicklung aufzuzeigen. Aus diesem Grund wird das Thema Userinterfacedesign bei FireTabletPlus stärker gewichtet. Weil das Gatway keine neuen Funktionen bietet, steht es nicht mehr im Mittelpunkt. Ansonsten sollen die Funkionen von FireTablet ausgebaut und vervollständigt werden, damit ein fertiges Produkt entsteht. FireTablet wird im Kapitel Analyse der FireTablet Software auf Seite 14 genau analysiert, um das Verbesserungspotential fest zu stellen.

Abbildung 1.6: Haupbildschirm von FireTablet wird neu SiteInfo Screen heissen.

1.6 Vorhandenes technisches Umfeld

Die Software FireTabletPlus wird im hier dokumentierten technischen Umfeld entwickelt. Die IP-Adresse des zentralen Servers ist eine globale Adresse und kann von überall her erreicht werden. Der Gateway steht allerdings im Labor und ist nur aus dem HSR-Secure WLAN ansprechbar. Im Vergleich zu FireTablet steht der Gateway nicht mehr so stark im Zentrum, sondern wird nur noch gebraucht, falls man vom zentralen Server eine leere Anlage lädt.

- Samsung Galaxy Tab P1000, Android 2.3.3, API Level 10
- Apache Webserver dient als zentraler Server
- Brandmeldezentrale FS2020
- REST BACnet Gateway 1.0
  Nicht alle nötigen Funktionen implementiert.

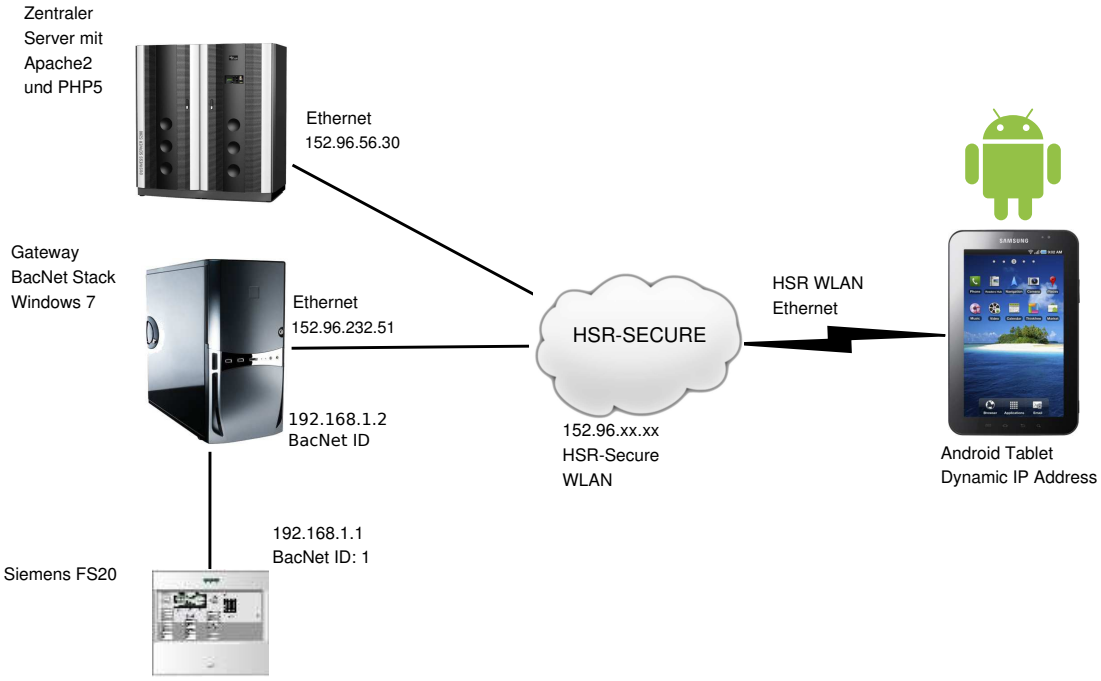


Abbildung 1.7: Grafik der Domänen einer Siemens Brandmeldeanlage

1.7 Aufbau der Dokumentation

Die Dokumentation ist in sechs Bereiche aufgeteilt, deren Sinn und Zweck in diesem Abschnitt erklährt wird, damit der Leser weiss was ihn erwartet.

1. **Einführung**
   Die Einführung ab Seite 1 gibt einen groben Überblick auf die gesammte Thematik der Arbeit und stellt die Probleme für welche eine Lösung gefunden werden soll dar.

2. **Bestehende Software** In Analyse der FireTablet Software ab Seite 14 wird die Software FireTablet genau untersucht. Dabei wird das Userinterface und der Sourcecode analysiert und nach möglichen Verbesserungen untersucht. Mit Hilfe der Erkentnisse aus diesem Kapitel wird das nächste Kapitel aufgebaut.

3. **Neue Anforderungen und Möglichkeiten**
   Das Kapitel Anforderungen und neue Möglichkeiten ab Seite 25 befasst sich mit den Ideen für die Weiterentwicklung. Es werden neue Funktionen definiert sowie ein Prototyp des Userinterfaces erstellt.

4. **Implementation**
   Im Bereich Implementierung von FireTabletPlus ab Seite 43 findet man alle Details zu den verschiedenen Softwarelösungen, die für FireTabletPlus erstellt wurden. Das definitieve Userinterface wird mit allen seinen Funktionen erklährt und dokumentiert.

5. **Zusammenfassung** Am Schluss findet man das Kapitel Zusammenfassung ab Seite 82. In diesem Teil der Arbeit wird das Ergebnis untersucht betreffend den erreichten Zielen, den vorhandene Probleme sowie den künftige Lösungen.

6. **Projektmanagement** Zwar hat der Abschnitt Projektmanagement ab Seite 85 nicht mehr direkt mit der Software FireTabletPlus zu tun, ist aber trotzdem wichtig für die Bachelorarbeit, um die Vorgehensweise und die Zeitplanung etwas genauer zu beleuchten.

1.8 Zeitaufwand der Arbeit

Die Bachelorarbeit wird in einem Herbstsemester erstellt, was gewisse Einschränkungen mit sich bringt. So stehen nicht wie normal 16 Wochen zur Verfügung, sondern nur 14 Wochen. Wenn man das wöchentliche Arbeitspensum berechnet, sieht das Ganze wie folgt aus:

	ECTS Punkte	Stunden/Punkt	Dauer	Stunden/Woche
BA	12	30h	14 Wochen	ca. 26h

Tabelle 1.1: Übersicht Arbeitsaufwand und Punkte

1.9 Betreuung und Partner

Auf der Seite der HSR wird die Arbeit von Prof. Peter Sommerlad betreut und von Thomas Corbat mit Reviews und Hilfe bei Problemen unterstützt. Die Firma Siemens wird von Herr Patrick Knellwolf representiert. Sowohl mit Herrn Sommerlad als auch mit Herrn Knellwolf werden wöchentliche Meetings oder Netmeetings abgehalten.

1.10 Android als Basisframework

FireTabletPlus wird als Androidapplikation entwickelt. In diesem Abschnitt werden Begriffe und Techniken die über das gesamte Projekt verwendet werden genauer erläutert damit sie für den Leser verständlich sind.

1.10.1 Intent und Activity

(Quelle: [BA FireTablet])
Jede Androidapplikation besteht aus einer beliebigen Menge von Activities.
[Activity Reference] Unter einer Activity versteht man in den meisten Fällen einen Bildschirm auf dem Display. Die Activity ist verantwortlich für die Darstellung der einzelnen Gui-Komponenten, aber auch für die Entgegennahme von Benutzereingaben. Aus diesem Grund hat eine Activity meistens einiges mehr an Verantwortung als reine Gui-Klassen. So muss zum Beispiel jede Activity ihren Lebenszyklus selber verwalten, wozu es aber geeignete Hookmethoden gibt. Im folgenden Bild wird der Lebenszyklus von Activities gezeigt. Am interessantesten sind die Methoden *onCreate()* und *onPause()*. Mit *onCreate()* wird die Activity initialisiert, da diese Methode bei jedem Start aufgerufen wird und eigentlich als Konstruktor gilt. OnPause() wird genau im umgekehrten Fall aufgerufen. Nämlich dann, wenn eine Activity in den Hintergrund tritt, sei es weil sie beendet wurde, oder weil eine neue Activity gestartet wird.

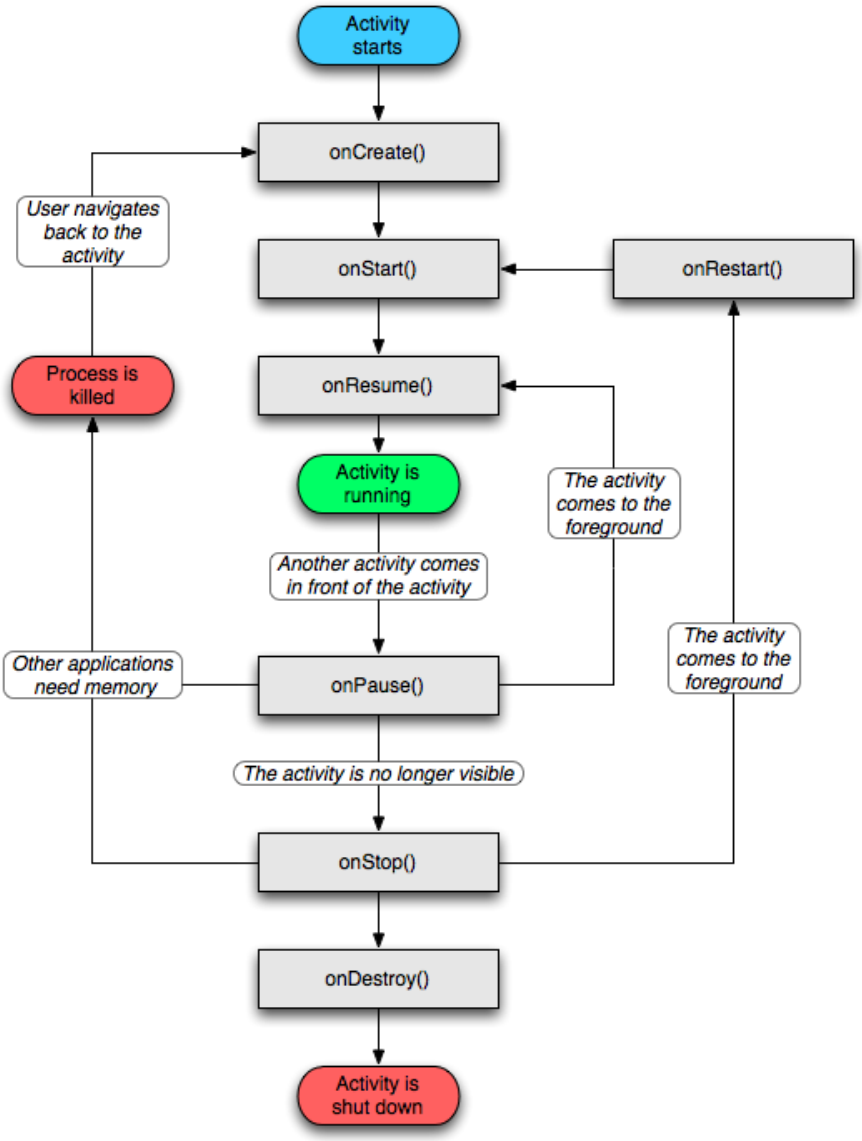


Abbildung 1.8: Lebenszyklus einer Activity. Quelle: [Activity Reference]

Um von einer Activity zur nächsten zu kommen, werden Intents eingesetzt. Dadurch können Activity sehr lose verknüpft werden und das Betriebssystem hat immer die Kontrolle darüber, welche Activities gestartet werden. Mit einem Intent übergibt man eigentlich dem Betriebssystem den Klassennamen der Activity, die gestartet werden soll. Das Betriebssystem startet diese anschliessend. Intents bieten auch die Möglichkeit primitive Datentypen mit zuschicken, um diese zwischen Activities Daten auszutauschen. Intents werden, wie im Listing unten, erstellt und dem Betriebssystem übergeben. [Android 2, S. 137]

```
Intent facilityTree = new Intent(this, LoadFacilityTree.class);
startActivity(facilityTree);
```

Listing 1.1: Intent erstellen und abschicken

1.10.2 Broadcast

Unter Broadcasts versteht man eine Art Postsystem für Nachrichten zwischen verschiedenen *Activities*. Broadcasts sind Nachrichten die abgegeben werden können, ohne einen Empfänger anzugeben. Statt einem Empfänger wird aber ein "Thema"bestimmt. Das Verteilsystem beliefert anschliessend alle Empfänger die Interesse für ein bestimmtes Thema angemeldet haben. Beim Empfänger kann ein *BroadcastReceiever* registriert werden, welchem man wiederum so genannte *IntentFilter* übergibt, mit denen man sein Interesse an einem Thema bekundet. Weiter Informationen zum Thema Broadcast sind der Android Reference unter [Broadcast] zu entnehmen.

1.10.3 AsyncTasks

(Quelle: [BA FireTablet])
AsyncTask ist eine Hilfsklasse des Androidframeworks, die das Zusammenspiel von Activities und Threads vereinfachen. [Painless Threading] Man verwendet Threads um das Userinterface ansprechbar zu halten, während länger dauernde Aktionen in Bearbeitung sind. Dies können Datenbankzugriffe oder Abfragen über das Netzwerk sein. Würden diese Aktionen im gleichen Thread ablaufen wie das Gui, würde dieses blockiert. Darum lagert man grössere Aktionen in eigene Threads aus, die dann parallel zum Userinterface ablaufen.
AsyncTasks kapseln den eben beschriebenen Vorgang, indem sie verschiedene Hilfsmethoden zur Verfügung stellen. Die Wichtigste ist *doInBackground()*. Diese Methode wird überschrieben mit dem Code, der in einem separaten Thread ausgeführt werden soll. Alle anderen Funktionen werden wieder im gleichen Thread der Activity ausgeführt. Die Methoden sind:

- *onCancelled()*
- *onPostExecute(Result result)*
- *onPreExecute()*
- *onProgressUpdate(Progress... values)*

Je nach Methode wird der Code vor, nach oder während der *doInBackground()* Methode ausgeführt.
Beim Erstellen eines *AsyncTask* werden drei Typenparameter deklariert. Diese bezeichnen die akzeptierten Parameter von *doInBackground()*, *onProgressUpdate()* sowie *onPostExecute()*. So kann das Verhalten eines *AsyncTask* völlig flexibel gestaltet werden.
AsyncTasks werden zum Beispiel dafür verwendet, um Fortschrittsanzeigen zu aktualisieren. Dabei läuft ein Prozess ab, der immer an einer gewissen Stelle die Methode *onProgressUpdate()* aufruft, die dann im Gui-Thread die Fortschrittsanzeige aktualisiert.

1.10.4 Android GUI Layout

Das Androidframework bietet die Möglichkeit grafische Interfaces sowohl programmatisch als auch mit XML zu generieren. Die bevorzugte Methode ist klar XML weshalb dieses System hier erklärt wird. Das nächste Bild zeigt ein kleiner Ausschnitt eines Layout XMLs.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    >
    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:orientation="vertical"
        android:id="@+id/document_overview_secound_layout"
         >

        <TableLayout
            android:id="@+id/document_overview_information_fields"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:layout_weight="3" >

            <TableRow
                android:id="@+id/tableRow1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" >
```

Abbildung 1.9: XML Code um das Layout einer Activity zu definieren.

Das XML ist hierarchisch aufgebaut und die verschiedenen Komponenten werden ineinander verschachtelt. So kann zum Beispiel ein *LinearLayout* andere Layouts aufnehmen und anschliessend darstellen. Unterstützt wird man von einem LayoutEditor der hilft das grobe Aussehen zu bestimmen. Will man kleine Details anpassen muss selber Hand angelegt werden und der XML Code bearbeitet werden. Nächstes Bild zeigt ein Ausschnitt des LayoutEditor.

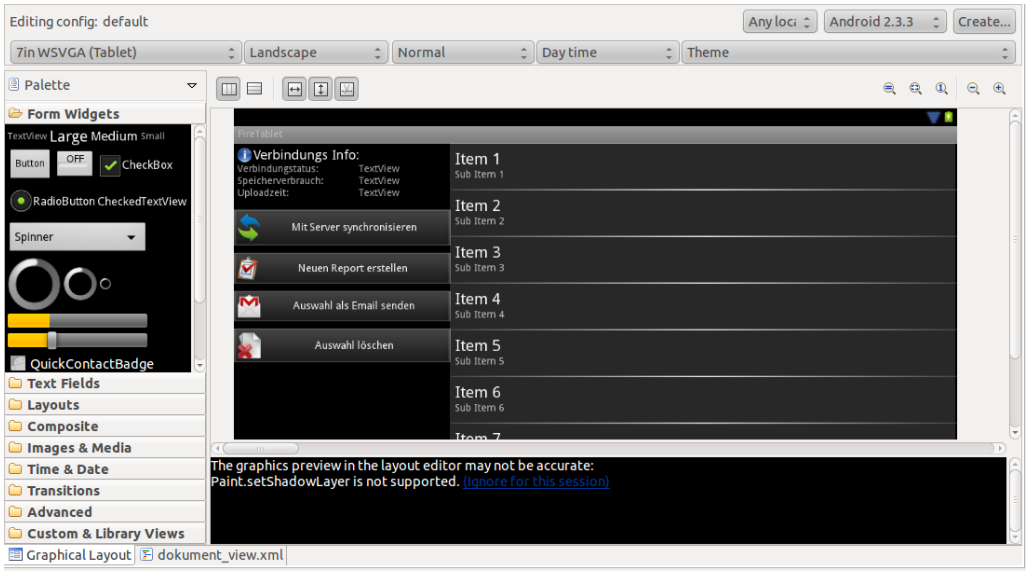


Abbildung 1.10: Android LayoutEditor.

Damit die XML-Layouts in Android verwendet werden können braucht es einen Layoutinflater der XML in GUI-Code umwandelt. Activities können aber direkt XML Layouts aufnehmen und diese selbständig umwandeln. Layout Dateien werden über das R-File von Android benutzt verwendet. Dieser Vorgang ist im folgenden Listing zu sehen.

```
LayoutInflater inflater = getContext().getLayoutInflater();

View aView = inflater.inflate(
   R.layout.facilityinfo_expandablelist_childview_additem_row,
      parent, false);

this.setContentView(aView)
```

Listing 1.2: XML Layout von R beziehen, inflaten und einsetzten.

2 Analyse der FireTablet Software

FireTabletPlus ist eine Weiterentwicklung der im Rahmen einer Bachelor- / Semesterarbeit erstellten Software [BA FireTablet]. Das Ziel der Entwicklung von FireTabletPlus ist eine Verbesserung und Funktionserweiterung von FireTablet weshalb an dieser Stelle der Vorgänger analysiert werden soll. Die Analyse erstreckt sich über die Bereiche Funktionalität und Userinterface aber auch der alte Sourcecode wird unter die Lupe genommen, um ihn gegebenenfalls zu optimieren. Bei der Analyse der alten Version geht es vorallem darum Verbesserungspotential und Schwachstellen zu suchen, welche mit dieser Arbeit verbessert werden können.

2.1 Funktionalität FireTablet

In diesem Abschnitt wird die bestehende Funktionalität von FireTablet dokumentiert damit klar wird, auf welcher Basis die Arbeit startet. Zuerst ist es wichtig nochmal ein Blick auf die technischen Rahmenbedingungen der Vorgängerarbeit zu werfen.

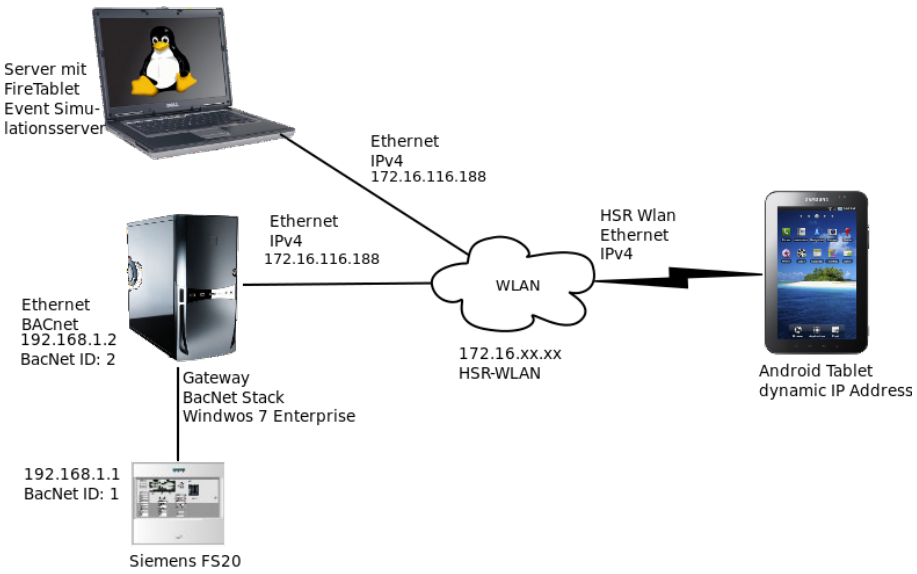


Abbildung 2.1: Infrastruktur von FireTablet

Wie in Bild 2.1 gezeigt wird kann das Tablet über WLAN mit einem Gateway oder mit dem Event Simulationsserver kommunizieren. Die einzelenen Geräte haben folgende Funktionen:

- **Android Tablet**
  Auf diesem Gerät läuft die FireTablet Software. Es ist als ständiger Begleiter eines Servicetechniker gedacht, um Brandmeldeanalgen zu testen.

- **Event-Simulationsserver**
  Damit FireTablet auf eingehende Testalarm-Events der Brandmeldezentrale reagieren kann, müsste das Gateway im Stande sein, diese Events an das Tablet weiter zu leiten. Da dies nicht der Fall ist, werden die Testalarme durch den Simulationsserver verschickt. Weiteres dazu im Abschnitt 2.1.4 Melderlinientest.

- **Gateway**
  Die Aufgabe des Gateway ist es das BacNet-Protokoll der Siemens Brandmeldezentrale in HTTP zu übersetzten damit die Brandmeldezentrale auch über einen Router hinweg erreicht werden kann. Weitere Informationen zur Notwendigkeit des Gateways können der Vorgängerarbeit [BA FireTablet, S.14] entnommen werden.

- **Siemens FS2020**
  Das Herz jeder Brandmeldeanlage ist die Brandmeldezentrale. In diesem Fall ist es eine Sinteso FS2020 Anlage von Siemens. Weiter Informationen zur eingesetzten Zentrale können [FS20 BacNet Spec] entnommen werden. Die Zentrale kennt die gesamte Hierarchie der Anlage und verarbeitet sämtliche Informationen welche gesammelt werden.

Die nun folgenden Funktionalitäten werden von FireTablet zur Verfügung gestellt. Jedes Feature wird im folgenden kurz erklährt und anschliessend wird auf die jeweiligen Probleme und Verbesserungspotentiale eingegangen. Für detailiertere Informationen sei hier nochmal auf die Arbeit [BA FireTablet] verwiesen.

2.1.1 Site von Gateway laden

Die logische Struktur der Brandmeldezentrale kann über das Gateway von der Brandmeldezentrale geladen werden. Dabei werden alle Elemente, die sich in der Anlagenhierarchie befinden, zum Tablet gesendet. Die Elemente sind in verschiedene Kategorien aufgeteilt, wie in Bild 2.2 ersichtlich ist. Es existieren Elemente vom Typ Panel, Area, Section, Zone und Channel.

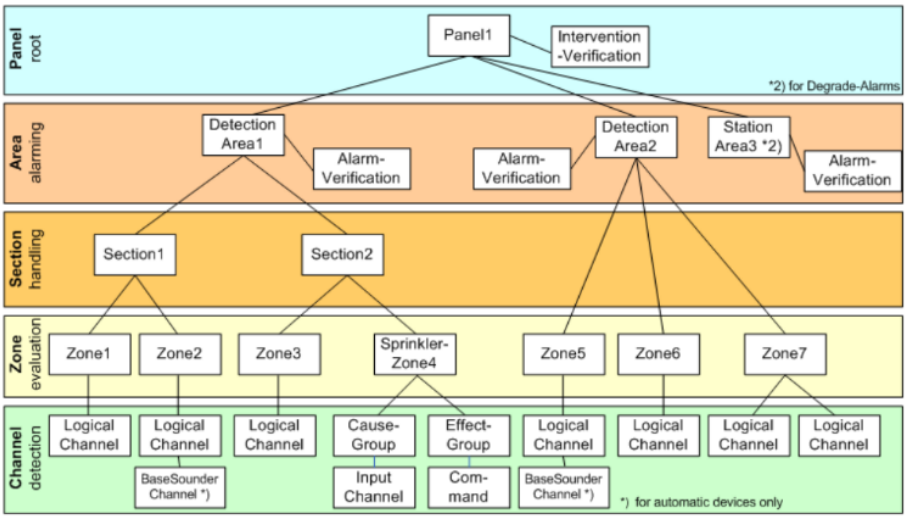


Abbildung 2.2: Die verschiedenen Kategorien der FS2020 Hierarchie. Quelle: [FS20 BacNet Spec]

Um den Ladevorgang anzustossen, muss der Benutzer zuerst die Verbindungsinformationen, bestehend aus Anlagenamen, IP-Adresse, Port und BacNet-Id, eingeben. Danach wird Element um Element via Gateway zum Tablet mit HTTP und XML geschickt. Nachdem die gesammte Struktur geladen wurde, wird die Anlagenhierarchie im *SiteInfo* Bildschirm angezeigt. Bei dieser Funktion wurden folgende Problembereiche festgestellt:

Problembereiche

- **Verbindungsparameter**
  Ein Servicetechniker der sein Tablet mit der Brandmeldezentrale verbinden will, muss zuerst die verschiedenen Parameter, wie oben erwähnt, am Tablet eingeben. Wenn man einmal davon absieht, dass es mühsam ist die IP-Adresse und alle anderen benötigten Parameter am Tablet einzugeben, bleibt immer noch eine hohe Gefahr, sich zu vertippen. Weiter muss der Servicetechniker von jeder Anlage die Verbindungsinformationen wissen, was den Verwaltungsaufwand unnötig kompliziert macht.

- **Zeitdauer**
  Für jedes Element, welches von der Zentrale geladen wird, werden 1-2 Sekunden benötigt. Dies kann bei grossen Anlagen extrem lange gehen, benötigt aber selbst bei der Demoanlage, welche für diese Arbeit zur Verfügung steht, mehr als eine Minute. Hinzu kommt, dass viele Elemente geladen werden, die später nicht gebraucht werden, da erst nach einem erfolgreichen parsen festgestellt werden kann, ob ein Element vom richtigen Typ ist oder nicht.

- **Unzureichende Informationen**
  Zwar kann von der Brandmeldezentrale die gesamte Hierarchie der Anlage bezogen werden, aber zu den einzelnen Elementen fehlen, ausser dem Kundentext, nützliche Informationen. Weitere Informationen über die einzelnen Geräte und Bereiche stellt das Gateway nicht zur Verfügung.

- **Kein Informationsaustausch möglich**
  Nachdem eine Anlage erfolgreich geladen wurde, können Fehler auf sämtlichen Komponenten erfasst werden. Diese werden zwar lokal auf dem Tablet gespeichert, aber es gibt keine Möglichkeit, die erfassten Ergebnisse unkompliziert weiter zu verarbeiten. Hinzu kommt, dass der nächste Servicetechniker mit einem anderen Tablet nicht weiss, welche Fehler bereits erfasst wurden.

2.1.2 Fehler für jede Komponente erfassen

Zu jeder Komponente können Fehler erfasst werden, die wiederum mit Bildern, Voicememos und Notizen versehen werden können. Sämtliche Fehler werden auf dem Tablet gespeichert und können zu einem späteren Zeitpunkt wieder betrachtet, gelöscht oder erweitert werden. Das Userinterface zur Fehlererfassung ist auf folgendem Bild zu sehen.

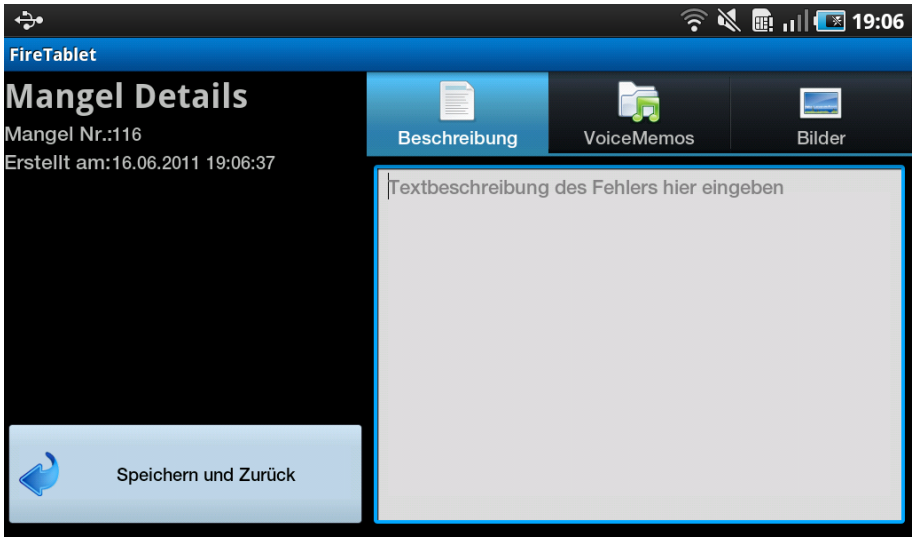


Abbildung 2.3: Bildschirm zur Fehlererfassung

Erfasste Fehler werden anschliessend in der Datenbank gespeichert. Die dazugehörenden Bilder und Voicememos werden im Dateisystem des Tablets abgelegt und es wird nur der Pfad zu den Dateien in der Datenbank gespeichert. Jedem Fehler wird eine eindeutige Fehler-Id zugewiesen, damit die Fehler mit dieser Id in der Datenbank einfach gefunden werden. Obwohl das Erfassen von Fehlern zufriedenstellend realisiert wurde, konnten ein paar Problembereiche lokalisiert werden.

Problembereiche

- **Keine Orientierung**
  Sobald man sich auf dem Fehlerdetailscreen befindet, weiss der Benutzer nicht mehr, für welche Komponente der Fehler erfasst werden soll. Es fehlt also eine Orientierungshilfe, die anzeigt, wo man sich aktuell gerade befindet.

- **Bilder und Memos bleiben auf dem Tablet**
  Da Bilder und Voicememos nur auf dem Dateisystem des Tablets abgelegt werden, weiss man zwar auf dem Tablet, welche Dateien zu welchem Fehler gehören. Es ist aber nicht möglich, die Bilder und Memos einem Fehler zuzuordnen, nachdem man zum Beispiel den ganzen Testbericht auf den PC geladen hat.

- **Keine Bildvorschau**
  Im Bilder Tab des Fehlerdetailscreen wird eine Liste mit Bildern, welche zum aktuellen Fehler erfasst wurden, angezeigt. Dabei sollte auch eine Miniaturansicht jedes Bildes vorhanden sein, damit sich der Benutzer schon ein grobes Bild davon machen kann, was ihn erwartet.

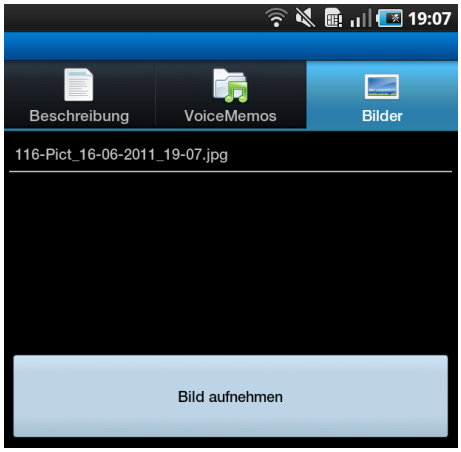


Abbildung 2.4: Liste mit gespeicherten Bildern aber ohne Miniaturansicht der Bilder.

- **Back-Taste löscht automatisch alle Daten**
  Das Bedienungsprinzip des Fehlerdetailscreen funktioniert so, dass der Benutzer auf allen drei Tabs seine Eingaben machen kann. Erst wenn er auf den SSpeichern und zurückButton klickt, wird ein Fehler registiert und auf die Datenbank geschrieben. Betätigt der Benutzer aber die Back-Taste, wird der Fehler nicht gespeichert, der Fehlerdetailscreen wird geschlossen und alle eingegebenen Daten sind verloren. Dieses Verhalten ist nicht intuitiv, denn der Benutzer sollte mindestens gefragt werden, ob er die Daten behalten möchte.

2.1.3 Daten exportieren

Die erfassten Fehler können nach abgeschlossener Anlageprüfung in eine Comma Separated Value (CSV) Datei exportiert werden. Die Datei wird lokal auf dem Tablet gespeichert und könnte für weitere Analysen verwendet werden. Erfasste Bilder und Voicememos werden nicht in die Exportdatei integriert, sondern nur mit dem Dateipfad referenziert. Diese Lösung ist natürlich für den täglichen Einsatz von FireTablet nicht praktikabel. Aus diesem Grund wurden auch hier einige Verbesserungspotentiale festgehalten.

Problembereiche

- **CSV**
  CSV ist nur bei der Verwendung einer Tabellenkalkulationssoftware sinnvoll. Zwar ist das Format vom Speicherverbrauch her optimaler, als z.B. XML. Letzteres ist aber universeller einsetzbar und kann besser von Menschen gelesen werden. Weitere Überlegungen zum Thema XML können dem Abschnitt 3.4.1, Seite 40 entnommen werden.

- **Lokale Speicherung**
  Die lokale Speicherung der Daten bringt Komplikationen mit sich. Es ist nur schwer möglich, die Daten an einen anderen Servicemonteur weiter zu geben oder gar zusammen am selben Objekt zu arbeiten. Zudem können Tablets verlohren gehen, gestohlen werden oder kaputt gehen, was ein Datenverlust zur Folge hätte. Auf diese Art bleibt aber auch die Möglichkeit, Daten mobil nachzuladen verwehrt, denn die CSV Dateien können nur nur mit einem PC nachgeladen werden.

- **Bilder und Voicememos nicht integriert**
  Bild- und Voicememodaten können nicht in ein CSV integriert werden und müssen separat gespeichert werden.

2.1.4 Melderlinientest

Die Idee des Melderlinientest ist es, alle Geräte auf korrekte Kommunikation mit der Brandmeldezentrale zu prüfen. Dabei können mehrere Brandmelder einem Melderlinientest unterzogen werden. Das Tablet wartet anschliessend auf einkommende Testalarmevents und setzt alle Brandmelder, für welche eine Event eingetroffen ist, auf positiv getestet. Brandmelder für die kein Testalarm eingegangen ist, werden als fehlerhaft markiert. Da der Gateway nicht im Stande ist, Testalarmevents von der Zentrale an das Tablet weiter zu leiten, wurde diese Funktion simuliert mit dem Event Simulationsserver. Mehr Informationen dazu findet man weiter oben, auf Seite 14.

Problembereiche

- **Zusätzliches Programm nötig**
  Der Eventserver läuft als ein separates Javaprogramm auf einem Server. Dieses Programm nimmt Ids von den Brandmeldern entgegen, welche man einem Linientest unterziehen möchte und löst anschliessend zufällig einen Testalarm für die einzelnen Brandmelder aus. Das zusätzliche Programm macht den Einsatz von FireTablet unnötig kompliziert.

- **Kein grosser Mehrwert**
  Events von einem Server an das Tablet schicken, ist keine grosse Errungenschaft. Interessanter wäre es, wenn man mit dem Testgerät für Brandmelder, die Testalarme auslösen könnte und diese vom Tablet empfangen werden.

2.1.5 SiteInfo Screen

Die gesamte Anlagenhierarchie wird wie in Bild 2.5 abgebildet dargestellt. Mit den Tabs rechts, kann zwischen den Ansichten: Subelemente, Informationen und Fehler gewechselt werden. Zusätzlich ist von hier die Aktivierung des Testmodus möglich, womit eine funktionierende Kommunikation zwischen Brandmelder und Brandmeldezentrale überprüfen wird.

Abbildung 2.5: Übersicht über die gesammte Anlagenhierarchie. Mit diesem Bildschirm kann durch die gesamte Anlage navigiert werden.

Zu jedem Element in der Analgehierarchie können Fehler erfasst werden und es ist ersichtlich wie viele Fehler bereits erfasst worden sind. Die Suche nach Verbesserungsmöglichkeiten hat hier vor allem Verbesserungen am Desing ergeben.

Problembereiche beim grafischen Interface

- Die Buttons auf der rechten Seite sind verwirrend, da man nicht sofort sieht, auf welches Element sie wirken.
- Mängel stehen nicht im Fokus, da immer direkt Subelemente angezeigt werden.
- Die Führung durch den Baum ist schlecht. Man sieht nicht auf einen Blick, wo man sich in der Hierarchie befindet.
- Die Mangelanzeige ist unklar, da nicht bekannt ist ob, der Mangel auf dem Subelement oder auf dem aktiven Element erfasst wurde.
- Navigation im Baum aufwärts wird mit der Back-Taste realisiert, was verwirrend ist.
- Die Suchfunktion ist über das Optionmenu schlecht zugänglich.
- Icons sind eintönig und es ist nicht auf den ersten Blick sichtbar, was sie bedeuten.
- Der zur Verfügung stehende Platz auf dem Display wird schlecht genutzt, da eigentlich sämtlicher Platz für die Navigation gebraucht wird.
- Es gibt keine Übersicht über die gesamte Hierarchie.
- Popups können vermieden werden.

Allgemeine Problembereiche

- **Neue Activities**
  Bei jedem Aufruf eines Kindelements wird eine neue Activity gestartet. Dies macht die Traversierung durch den Baum sehr langsam.
- **Datenbankabfragen**
  Es werden zu viele Datenbankabfragen gemacht, da immer das ganze Panel persistiert wird, wenn ein Kindelement aufgerufen wird.

2.1.6 Mainmenu Screen

Der MainMenu Bildschirm wird direkt nach dem Programmstart angezeigt. Er gibt die Möglichkeit, entweder eine bereits gespeicherte Anlagenhierarchie zu laden, oder zu einer neuen Anlage über das Gateway zu verbinden.

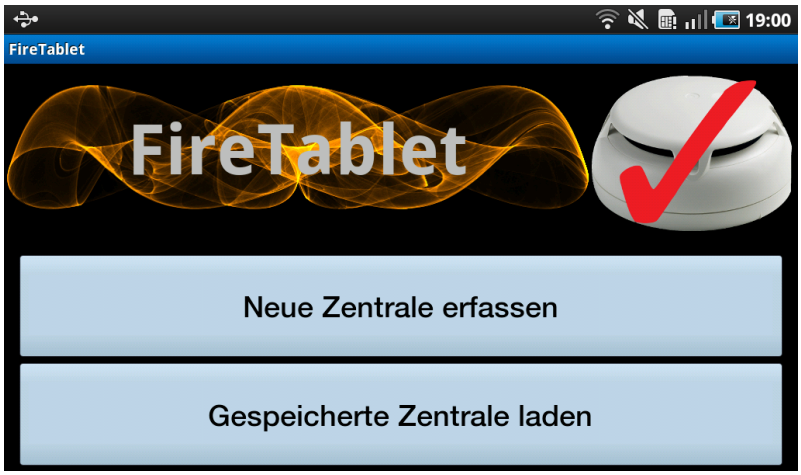


Abbildung 2.6: Hauptmenu direkt nach dem Programmstart

Problembereiche beim grafischen Interface

- Die Buttons sind zu gross.
- Der Button um gespeicherte Zentralen zu laden ist überflüssig. Gespeicherte Zentralen können direkt in einer Liste angezeigt werden.
- Die Tastatur für die Eingabe von IP-Adressen ist schlecht, da ein Punkt nicht ohne Umschaltung zur Verfügung steht.
- Datumstrings können besser leserlich formatiert werden.
- Das Logo ist zu hoch. Der Platz sollte für die Auswahl der gespeicherten Zentralen verwendet werden.

2.2 Der Sourcecode von FireTablet

Das Codereview des FireTablet Sourcecodes hat vor allem sehr viele kleine Anpassungen nach sich gezogen. Die vielen Detailanpassungen werden aber an dieser Stelle nicht spezi-

ell behandelt. Der Fokus liegt auf den grossen, grundlegenden Änderungen an Architektur und Code. Diese werden im folgenden Abschnitt dokumentiert. Weiter Informationen zur Codearchitektur von FireTablet können [BA FireTablet, S.76] entnommen werden. Informationen zu häufig verwendeten Androidklassen sind in [BA FireTablet, S.20] dokumentiert.

2.2.1 Zyklomatische Abhängigkeiten

Einer der Hauptfehler im Code von FireTablet waren die zyklischen Abhängigkeiten unter den Mainpackages. Unter zyklischen Abhängigkeiten versteht man Zugriffe, die sowohl von Package A nach B gehen, als auch von B nach A. Zyklische Abhängigkeiten deuten immer auf zu hohe Kopplung unter den Packages, oder auf eine falsche Einteilung der Packages hin. Die falschen Abhängigkeiten sind im folgenden Bild dargestellt.

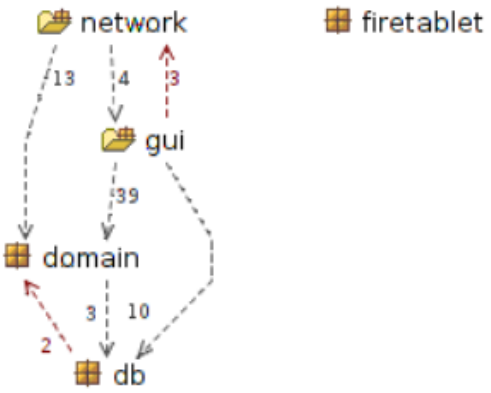


Abbildung 2.7: Abhängigkeiten unter den Mainpackages mit zyklomatischen Abhängigkeiten.

Zwei Grundlegende Probleme konnten bei der Verbesserung identifiziert werden.

Zu hohe Kopplung zwischen gui und network

Im Package *gui* werden zwei *AsyncTasks* gestartet. Diese Klasse ist von Android und ermöglicht einen bequemen Umgang mit Threads, indem sie verschiedene Callbackmethoden zur Verfügung stellt. Die *doInBackground()* Methode wird anschliessend auch wirklich in einem separaten Thread ausgeführt. Alle anderen Methoden laufen im GUI-Thread ab. Dies hat zur Folge, dass diese Funktionen wieder auf die Activities einwirken, was eine sehr hohe Kopplung mit sich bringt. Für diese Probleme gibt es zwei Lösungen.

1. ***AsynTasks als Innerclass***
   Die Asynctasks hätten jeweils in den aufrufenden *Actitivies* als Innerclass implementiert werden können. Dadurch wäre es ohne Aufwand möglich gewesen, aus der AsyncTask-Klasse die *Activity* zu beeinflussen. Der Nachteile ist die Grösse der Activity-Klasse und die Tatsache, dass man den Netzwerkcode in einem Package hat, welches eigentlich nichts mit Netzwerk zu tun hat.

2. **Kommunikation mit *Broadcasts* lösen**
   Android bietet die Möglichkeit Broadcasts zur Kommunikation zu verwenden. *Broadcasts* werden in Abschnitt Broadcast auf Seite 11 vorgestellt. Die Handhabung ist sehr einfach und bietet sich zur Entkopplung an.

Die zyklomatischen Abhängigkeiten zwischen Package *gui* und *network* wurden mit der Broadcast-Lösung entschärft. Broadcasts sind eine sehr praktische Möglichkeit, um Kopplung zu verringern. Sie gestalten die Arbeit mit den betroffenen Klassen enorm einfach. Die resultierende Abhängigkeitsgrafik ist im folgenden Bild ersichtlich.

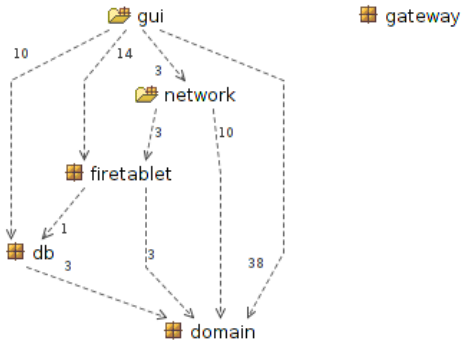


Abbildung 2.8: Bereinigte Abhängigkeiten unter den Mainpackages ohne zyklische Abhängigkeit

2.2.2 Trennung von Fehlern und Anlagebaum auf der Datenbank

In FireTablet enthielten die verschiedenen *Devices* nicht die erfassten Fehler. In einem Device wurde lediglich eine Liste von *FaultIds* gespeichert. Mit dieser Id konnte danach in der Datenbank der Fehler gesucht werden. Erklärungen zu den Klassennamen von FireTablet können in [BA FireTablet, S.77] gefunden werden.
Diese Lösung musste überarbeitet werden, weil die Verbindung über die Datenbank nicht effizient war. Zum Beispiel musste man, um die Fehler eines *Device* anzuzeigen, jedes Mal Datenbankzugriffe ausführen. In einer Scrolllist, wie sie auf Abbildung 2.5 rechts im Bild zu sehen ist, gibt es damit Probleme. Die einzelnen Elemente können nicht genügend schnell geladen werden, sobald sie in den sichtbaren Bereich treten. Die Ursache liegt bei den vielen feingranularen Datenbankzugriffen, welche zu langsam sind. Die Elemente werden darum im Voraus geladen, was man mit Komplexität und dem Verlust an Dynamik büsst. Es macht zudem auch nicht viel Sinn, bei einer objektorientierten Datenbank [Db4o] selber Ids für Fehler zu verteilen. Dieser Vorgang sollte ausschliesslich von der Datenbank selber erledigt werden. Die Lösung mit den Ids für Fehler würde aber Sinn machen, wenn man eine gewaltige Menge von Fehlern hätte. Dies war allerdings ein Überlegungsfehler der Vorgängerarbeit, da man bei Routinekontrollen möglichst wenig Fehler finden sollte.

2.2.3 Entkoppelung der Domainobjekte

Objekte wie *Device*, *Panel* und *Fault* hatten zu viel Kompetenz. Die Objekte konnten sich selbstständig in der Datenbank persistieren. Aus diesem Grund brauchten diese Objekte

immer einen *ApplicationContext*, um auf die Infrastruktur von Android Zugriff zu haben. Auch dieser Vorgang macht das Handling dieser Klassen unnötig kompliziert, weswegen diese Abhängigkeit entflechtet wurde. *Devices* werden nun nur noch von den *Activities* persistiert und verloren damit deutlich an Komplexität.

2.2.4 Listadapter als Innerclass

Listen für Android sind eine optimale Art um Informationen auf dem relativ kleinen Bildschirm darzustellen. Dem zufolge werden Listen, als so genannte *ListViews*, in FireTablet häufig eingesetzt. Eine *ListView* besteht immer aus der *ListView* selber und einem Adapter, welcher die angezeigten Daten verwaltet und die einzelnen Elemente der Liste darstellen kann. Die *Adapter* für alle Listen im Programm wurden immer in separaten Klassen implementiert. Weil man aber zwischen dem *Adapter* und der *Activity* eine hohe Abhängigkeit hat, mussten dafür unschöne Lösungen realisiert werden. Darum wurden alle Adapter als *InnerClass* programmiert, wo sie einfach auf Attribute der äusseren Klasse zugreifen können.

3 Anforderungen und neue Möglichkeiten

Zu Beginn der Projektarbeit haben wir uns Gedanken gemacht welche Funktionalität sinnvoll für Siemens wäre und wie realistisch eine Implementation in der vorhandene Zeit ist. In diesem Kapitel wird die Entstehung der Featurelist dokumentiert, sowie was die einzelnen Funktionen im Detail bedeuten. Weiter werden die Anforderungen an die Software festgehalten.

3.1 Featurekatalog

Der Featurekatalog ist das Resultat aus der Analyse des Vorgängerprogramm und Absprachen mit dem Projektbetreuer. Der Katalog wurde erstellt um eine gute Gesprächsgrundlage mit dem Betreuer von Siemens zu haben, damit er sich ungefähr vorstellen kann, was zu erwarten ist.

- **GUI Überarbeiten**
  - SiteInfo-Screen gemäss neuen GUI-Prototypen implementieren.
  - Performancesteigerung durch Benutzung der gleichen Activities für jede Ansicht im SiteInfo-Screen.
  - MainMenu-Screen: Gespeicherte Tests direkt mit einer Liste anzeigen wenn Programm geladen wird.
- **Dokumentgenerierung für Testberichte**
  - Dokumente generieren als XML für maschinelle Weiterverarbeitung der Daten.
  - PDF generieren um diese als Inspectionreport direkt dem Kunden zu schicken.
- **Speichern der Dokumente auf zentralem Server**
  - Dokumente sollen manuell bei guter Internetverbindung zum Server geschickt werden können.
  - Anlage inklusive Fehler und Informationen auf einen zentralen Server synchronisieren.
- **Site Checkliste**
  - Das Programm führt den Kontrolleur durch die Prüfung der Zentrale.
  - Checklisten sollen mit Templates konfiguriert werden können wegen Mehrsprachigkeit, verschiedenen Gesetzen und verschiedenen Anlagen.
- **Gebäudepläne anzeigen**
  - Der Gebäudeplan kann auf dem Tablet angeschaut werden.
  - Bei einem eingehenden Testalarm soll auf den entsprechenden Melder auf dem Plan fokussiert werden können.

- **Gateway Integration ausbauen**
  - Abklären was genau der Status des Gateways ist und je nach Weiterentwicklung neue Funktionen realisieren.
  - Testloggs, Gebäudepläne, Kundeninformationen und Testberichte auf dem Gateway speichern.
  - Gateway kann auf Testalarmereignisse hören und wird von der Zentrale über Alarme informiert.
  - Abruf der Anlagehierarchie via Gateway optimieren.

3.1.1 Anpassungen durch Kundenwunsch

Die Meetings mit dem Projektbetreuer auf der Seite von Siemens haben folgende Anforderungen ergeben.

- **Fokus Userinterface**
  Weil FireTabletPlus vor allem zu Demonstrationszwecken entwickelt wird, liegt der Fokus auf dem Userinterface. Der Kunde möchte der Geschäftsleitung vor allem zeigen welche Möglichkeiten in ihrem Geschäftsbereich bestehen. Zwar sind keine perfekten Designs möglich aber wenigstens von der Bedienung her möchte man eine Referenz haben.

- **Internetverbindungs Status** Weil die Internetverbindung über das GSM/UMTS Netz immer wieder für Probleme verantwortlich ist und der Empfang eines Netzes nicht immer gewährleistet werden kann, wird Wert auf eine genaue Anzeige des Netzwerkstatus gelegt. Zwar zeigt das Androidinterface den Netzwerkstatus immer an, aber es wäre schön wenn an wichtigen Stellen die Anzeige noch etwas prägnanter wäre.

Abbildung 3.1: Das Symbol ganz links zeigt den Netzwerkstatus an. In diesem Fall WLAN.

- **Keine technisch fertige Lösung** Viele Features wie zum Beispiel Gebäudeplan anzeigen oder die Speicherung auf einem zentralen Server benötigen geeignete Schnittstellen. Diese sollen aber nicht im Zentrum stehen, da sie bei einer Weiterentwicklung sowieso neu gemacht werden müssen.

- **Keine Neuigkeiten beim Gateway** Der Status des Gateways wurde mit einem Fragekatalog an den Entwickler des Gateway abgefragt. Leider wurden keine Erweiterungen hinzugefügt. Der Gateway ist zudem nicht spezifisch für Siemens Brandmeldeanlagen, sondern allgemein für BacNet. Die meisten Funktionen werden also nach wie vor nicht unterstützt. Somit fällt der Punkt "Gateway Integration ausbauen"vom Featurekatalog weg.

3.2 Userinterface Design und Evolution

GUI-Design ist ein andauernder Prozess bei welchem Ideen und Vorstellungen mit der Zeit wachsen und entstehen. Im folgenden wird auf einige Gedankengänge und Prozessschritte eingegangen. Anschliessend wird der genaue Ablauf der Arbeitschritte gezeigt.

1. Als Erstes wurde die bestehende Lösung analysiert und Probleme festgehalten.
2. In Gesprächen mit dem Kunde, Arbeitsbetreuer und im Team wurden Verbesserungen diskutiert und festgehalten.
3. Unabhängig von einander erstellt jedes Teammitglied ein persönlicher Prototyp auf Papier.
4. Beide Lösungsvorschläge werden zusammengeführt und die besten Ideen übernommen. Dies führt zu einem zweiten Prototyp.
5. Der zweite Prototyp wurde dem Kunden präsentiert. Die Verbesserungsvorschläge führen direkt zum finalen Prototyp.

3.2.1 Zielsetzung Userinterface Design

Das Redesign soll folgende Ziele verfolgen:

- **Fehler** sollen besser sichtbar sein. Sowohl auf einem einzelnen Element als auch über die ganze Anlagenstruktur. Der Zustand der Anlage soll immer Sichtbar sein.
- **Die Navigation** soll durch eine Übersicht, welche das aktuelle Level der Anlagenstruktur darstellt, effizienter werden.
- **Wiedererkennungseffekt** mit Sinteso erstellen, indem gleiche Icons und die ausklappbare Baumstruktur verwendet wird.
- **Informationen** sollen auf den ersten Blick ersichtlich sein ohne durch verschiedene Tabs zu springen.

3.2.2 Erste Entwürfe

Die Entwürfe wurden, wie weiter oben erwähnt, von beiden Teammitglieder unabhängig erstellt. Sie weisen aber trotzdem eine gewisse Ähnlichkeit auf, da die frühere Version des Programms als Vorlage zugrunde liegt. Anschliessend werden die wichtigsten Entwürfe dokumentiert und die verschiedenen Lösungen gegenüber gestellt.

SiteInfo Entwurf

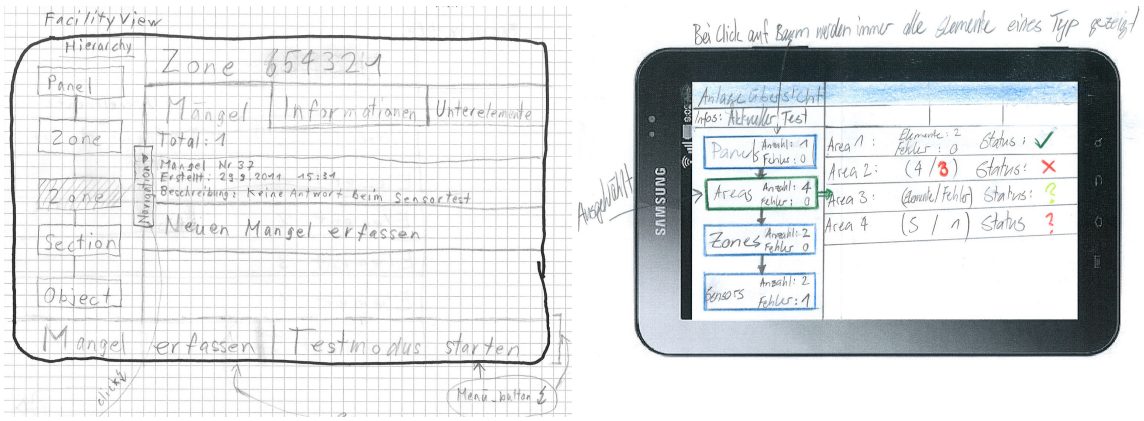


Abbildung 3.2: Entwürfe für FacilityInfo.

Beide Entwürfe zeigen die vorgesehene Navigation in der Linken Seite des Screens, die dazu dient jederzeit die aktuelle Tiefe des Baumes anzuzeigen. Gemeinsam ist auch die Anzeige einer Liste mit Elementen im rechten Bereich. Allerdings gibt es dabei bereits Unterschiede, denn links werden direkt vorhandene Fehler angezeigt wobei rechts Kindelemente aufgelistet werden. Auf beiden Entwürfen sind noch Tabs vorgesehen um die verschiedenen Details zu einem Element anzuzeigen. Dies ist noch nicht mit der Zielsetzung in Abschnitt 3.2.1 vereinbar ist. Der linke Entwurf kennt bereits die Lasche zwischen Navigation und Listelemente um einen Baum der Anlage hervor zu ziehen.

Element auswählen aus der Liste

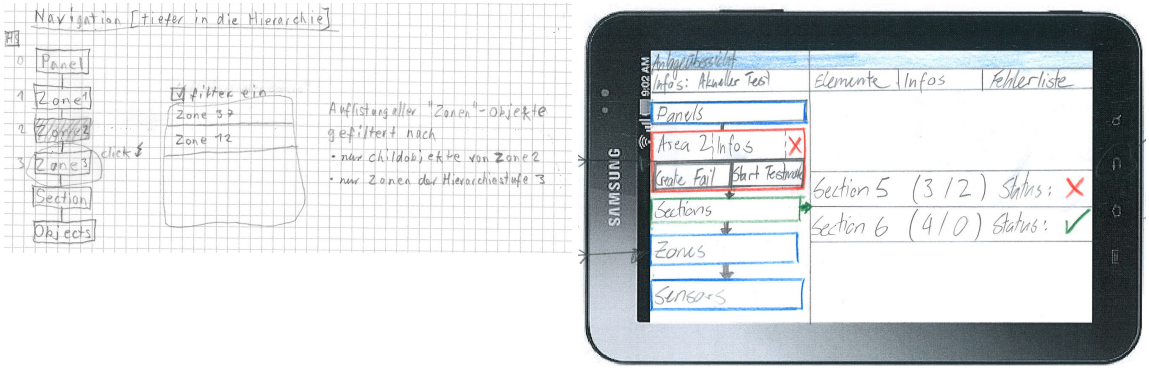


Abbildung 3.3: Die Entwürfe zeigen wie durch die Anlagehierarchie navigiert wird

Im linken Entwurf kann ausschliesslich über die Navigation durch den Baum traversiert werden indem man zuerst in der Navigation auf eine Gruppe von Elementen klickt und anschliessend mit Hilfe eines Popups auswählt welches genaue Element man angezeigt haben möchte. Im rechten Bild wird direkt auf ein Kindelement geklickt, um dort hin zu traversieren. Dabei wird in der Navigation links die Funktionen angezeigt welche man auf dem aktuell ausgewählten Element zur Verfügung hat.

Anlagebaum anzeigen

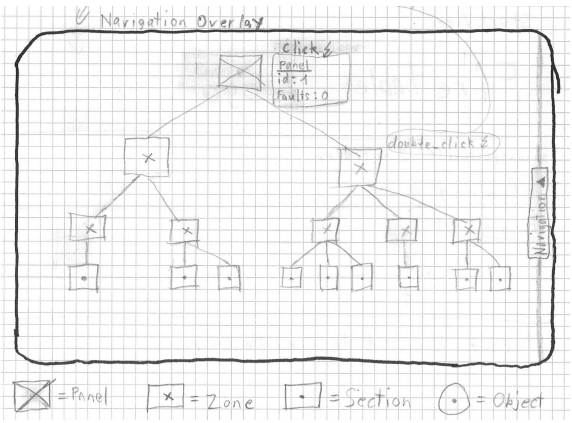


Abbildung 3.4: Anlagebaum kann angezeigt werden.

Für jedes Element kann die entsprechende Position im Anlagebaum angezeigt werden. Im Anlagebaum kann man zoomen und scrollen sowie mit einem Doppelklick wieder zur Listenansicht des ausgewählten Elements wechseln. Die Idee, die Hierarchie in einem Baum darzustellen, wurde zugunsten eines ausklappbaren Baums wieder fallen gelassen, da oben links und rechts zu viel Platz auf dem Display verschwendet würde.

Testübersicht

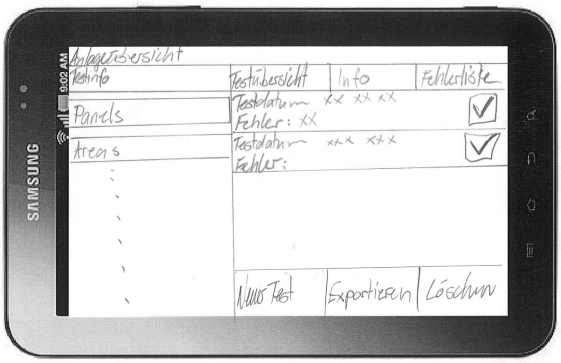


Abbildung 3.5: Testübersicht listet alle erstellten Tests auf

Mit der Testübersicht können bereits erstellte Tests wieder geladen werden. Die Testübersicht wird automatisch angezeigt wenn eine bereits vorhandene Brandmeldezentrale neu geladen wird.

3.2.3 Gui Prototyp

Im Abschnitt Prototyp werden Papierentwürfe des Userinterfaces dokumentiert welche dem Kunden für letzte Änderungsvorschläge vorgelegt wurden. Diese Version setzt sich aus den Erkenntnissen aus Sitzungen und eigenen Gedanken zusammen. Es wird im folgenden auf die wichtigsten Bildschirme und Konzepte eingegangen. Zu beachten ist dabei die Weiterentwicklung der im Kapitel 2.1 ab Seite 14 gemachten Analysen.

MainMenu Screen

MainMenu ist der Bildschirm direkt nach dem Starten der Anwendung. Im unteren Bereich werden bereits erstellte Brandmeldeanlagen in einer Liste angezeigt, mit dem Button *Neue Zentrale erfassen* kann eine Verbindung zu einer Brandmeldezentrale aufgebaut werden, womit das Tablet alle Elemente der Brandmeldeanlage lädt.

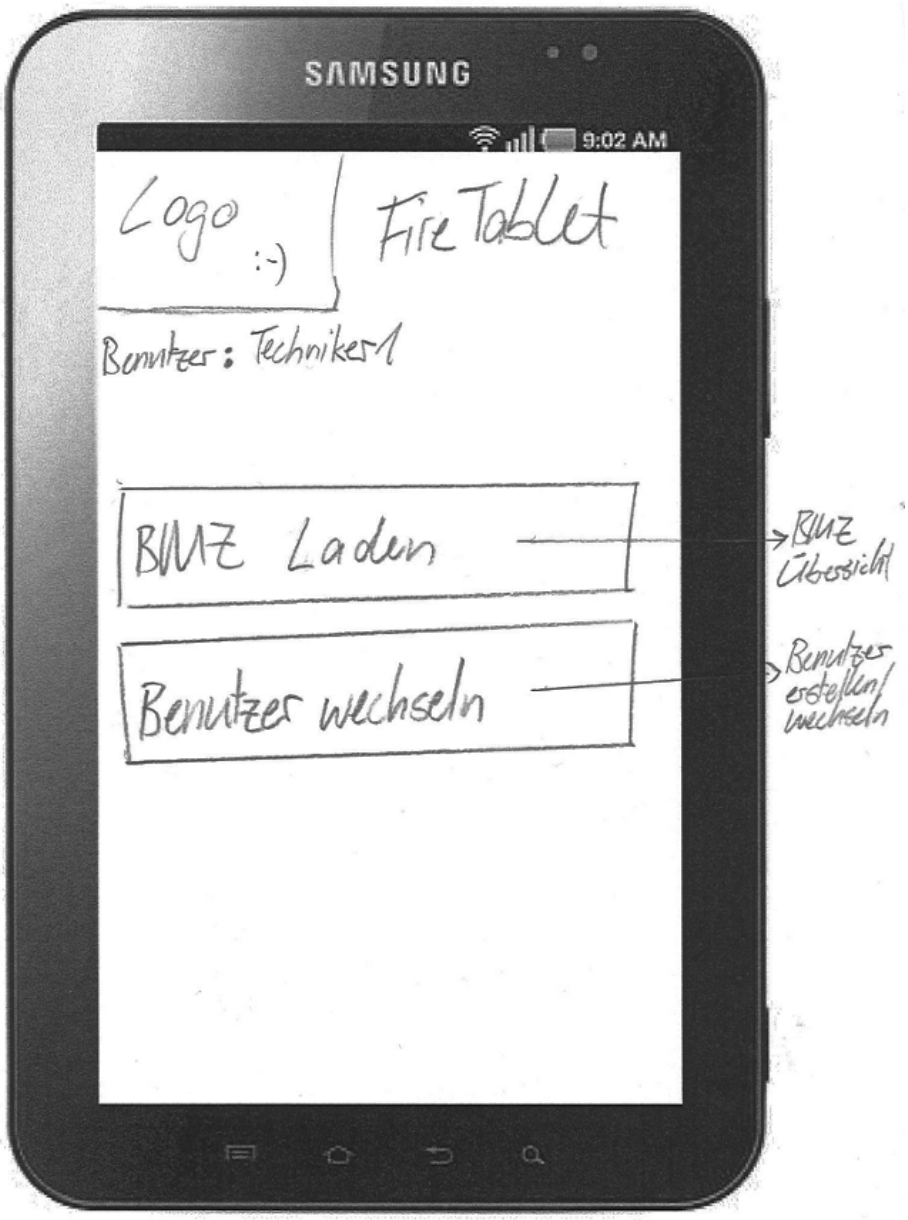


Abbildung 3.6: Startbildschirm MainMenu

SiteInfo Screen

Der SiteInfo Screen ist der Hauptbildschirm der Anwendung. An dieser Stelle kann durch die gesamte Anlage navigiert, sowie Fehler zu jedem Element erfasst werden.

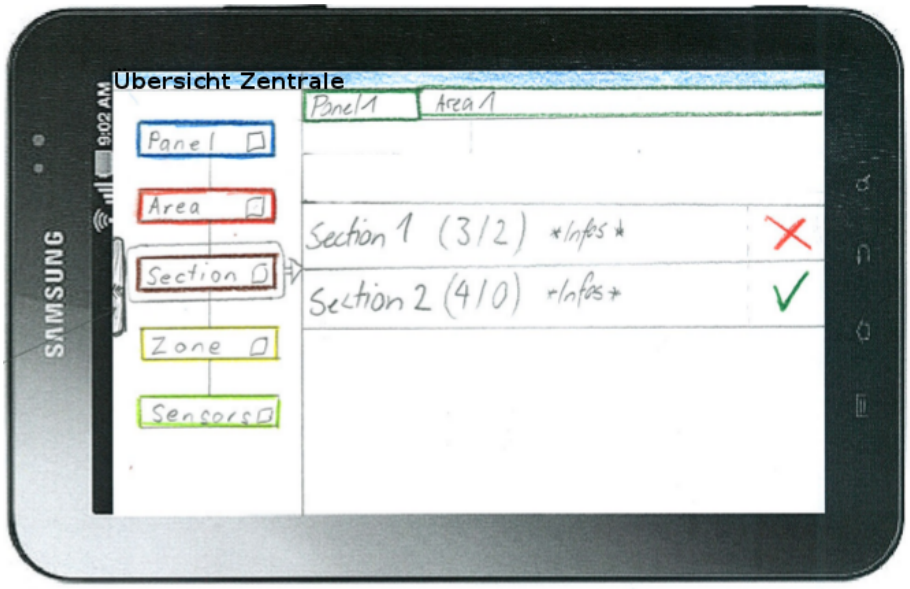
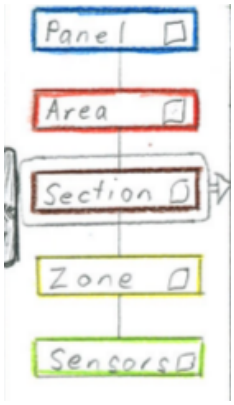


Abbildung 3.7: Übersicht der gesamten Zentralenstruktur

Klick auf eine Kategorie im Anlagebaum



Mit einem Klick auf das entsprechende Level im Anlagebaum werden alle vorhandenen Elemente dieses Levels angezeigt. Zum Beispiel Zeigt ein Klick auf Area im rechten Teil alle Areas die gefunden werden. Gleichzeitig verschiebt sich der Cursor auf das gerade angezeigte Element.Weiter kann an der Lasche links gezogen werden um den ausklappbaren Baum wie in Sinteso, hervor zu ziehen.

Abbildung 3.8: Sprungpunkte auf den Anlagebaum

Klick auf Listenelement

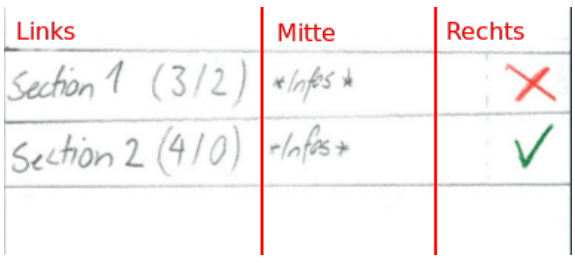


Abbildung 3.9: Listenelemente zeigen die wichtigsten Infos direkt an.

Listenelemente sind in drei Bereiche unterteilt. Je nachdem wo man klickt wird eine andere Aktion ausgeführt.

1. **Links:** Name des Elements sowie in Klammer die Anzahl Kindelemente und Anzahl erfasster Fehler. Beim Klick auf diesen Bereich wird das Element ausgewählt und angezeigt. Siehe: Listelement ausgewählt 3.2.4
2. **Mitte:** In der Mitte eines Listelements befindet sich der Infobereich mit den wichtigsten Informationen zu einem Element. Beim Klick auf den mittleren Bereich wird das Listelement aufgeklappt und es werden sämtliche Informationen dazu angezeigt. Siehe: Aufgeklappte Infos 3.2.6
3. **Rechts:** Ganz Rechts wird dargestellt ob ein Element oder seine Kindelemente registrierte Fehler aufweist. Ist dies der Fall, erscheint ein rotes Kreuz. Wird das Kreuz angeklickt, klappt sich das Listelement auf und es werden alle Fehler aufgelistet. Siehe: Aufgeklappte Fehler 3.2.5 Wenn keine Fehler vorhanden sind, erscheint an dieser Stelle ein grünes Häcklein. Klickt man es an, so wird auf dem entsprechenden Element ein neuer Fehler erfasst und die Fehleransicht öffnet sich. Siehe: Fehler Details 3.2.7

Klick auf den Elementverlauf

Abbildung 3.10: Oberhalb des Listelementbreichs werden die konkreten Elemente angezeigt die man besucht hat.

Über den Listelementen werden in einer separaten Spalte die traversierten Elemente als Pfad angezeigt. Man nennt diese Navigationshilfe Breadcrumbs. Mit Breadcrumbs ist es möglich, sich im Baum wieder zurück nach oben zu bewegen. Im Gegensatz zur Linken Navigation handelt es sich hier um die konkreten Elemente und nicht um die Gruppe von Elementen. Ein weiteres Beispiel für das Verhalten dieses Bereiches ist unter *Listelement ausgewählt* 3.2.4 ersichtlich.

3.2.4 Listelement ausgewählt

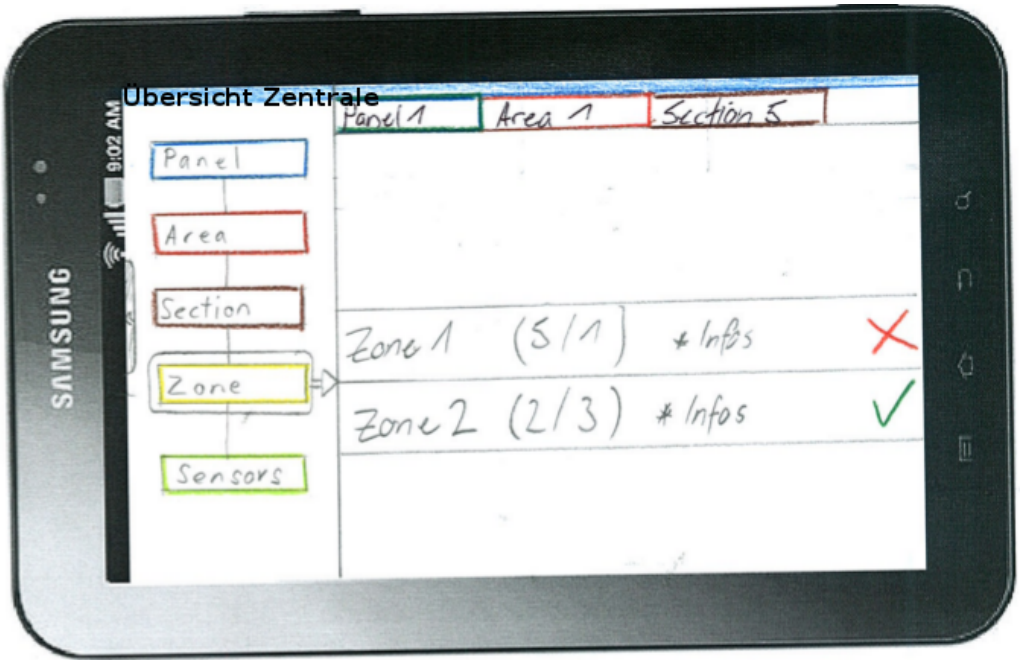


Abbildung 3.11: Ein konkretes Element wurde ausgewählt.

Wie man im Bild 3.11 erkennt haben sich nach der Auswahl eines konkreten Elements folgende Dinge geändert:

- Der Index in der Navigation links ist um ein Level tiefer gerutscht und zeigt nun Elemente vom Typ Zone an.
- Die Liste rechts zeigt nun alle Kindelemente des vorher ausgewählten Elements an.
- Die Pfadanzeige zeigt als zusätzliches Element die vorher besuchte "Section 5" an.

Die Interaktionen auf den Listlementen bleiben gleich wie in Klick auf Listenelement Seite 31 beschrieben.

3.2.5 Aufgeklappte Fehler

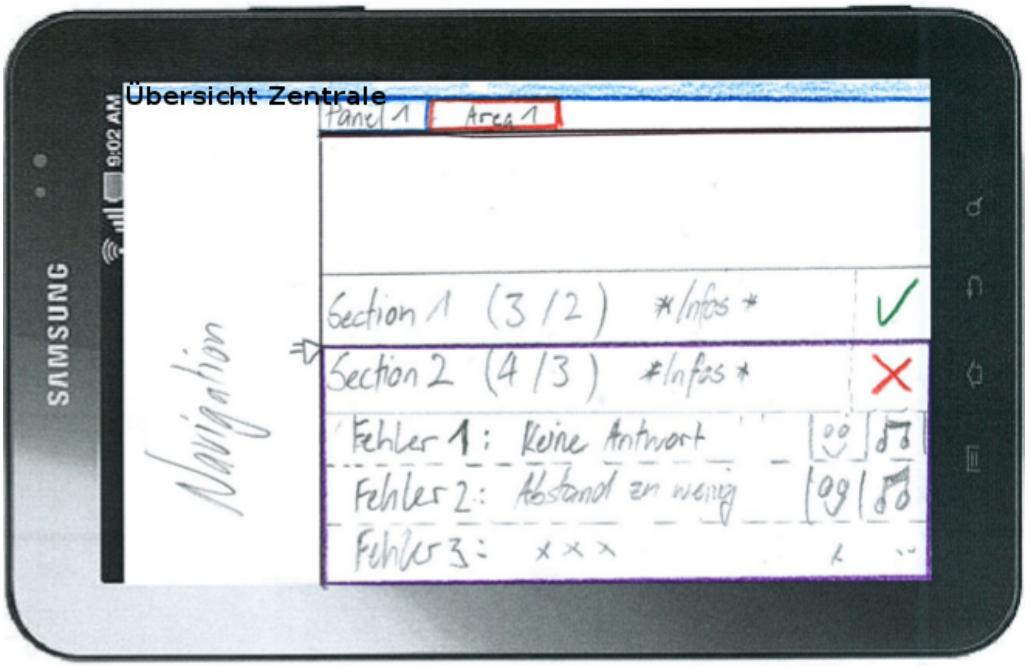


Abbildung 3.12: Das Listelement klappt sich auf und Zeigt alle erfassten Fehler an

Im aufgeklappten Bereich erscheinen alle Fehler mit einer kurzen Beschreibung. Rechts bei den aufgelisteten Fehlern ist über zwei Icons sichtbar ob dem Fehler ein Bild oder eine Voicememo angehängt wurde. Je nachdem sind die Icons ausgegraut oder farbig. Beim Klick auf einen Fehler erscheint die *Fehler Details Ansicht* 3.2.7

Mit der Back-Taste oder durch wiederholtes Anklicken des Listelmentes klappt die Fehleransicht wieder ein.

3.2.6 Aufgeklappte Infos

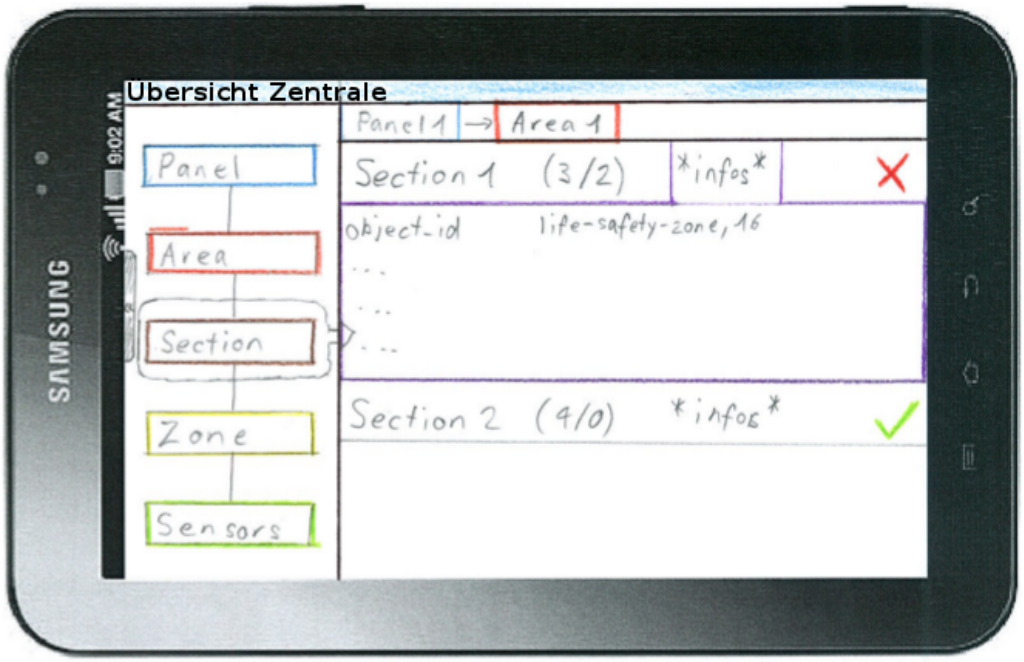


Abbildung 3.13: Das gewählte Listelement klappt sich auf und zeigt weitere Informationen an.

Alle Informationen zu einem Element werden angezeigt. Mit der Back-Taste oder wiederholtes Anklicken des Listelementes klappt die Informationsansicht wieder ein.

Name	Beschreibung
object-identifier	Einmalige Identifikation jeder Life-Safety-Zone.
object-name	Name der Life-Safety-Zone.
description	Kundentext einer Zone, in welcher z.B. beschrieben wird, in welchem Raum sich ein Gerät befindet.
device-type	Zonentyp. In FireTablet werden AreaElem, ZoneElem, SectionElem und SensorElemente verwendet.
notification-class	Meldungsklasse bei welcher sich die Zonen melden. Gleichzeitig kann sich ein Observer bei einer Meldungsklasse registrieren um die Events aus dieser Klasse zu empfangen
notify-type	Die Art des Events der ausgelöst werden kann.
maintenance-required	Dieser Wert wird zu "true"wenn die Life-Safety-Zone gewartet werden sollte.
isa-event-message-texts	Letzte geloggte Aktion die auf der Zone ausgeführt wurde.
profile-name	Profilname der Zone
member-of	Liste von Object Identifiers die zur aktuellen Zone Elternelemente sind.
zone-members	Liste von Object Identifiers die Kindelemente der aktuellen Zone sind.

Tabelle 3.1: Bedeutung der Attribute einer Life-Safety-Zone

3.2.7 Ausgezogene Lasche mit TreeView

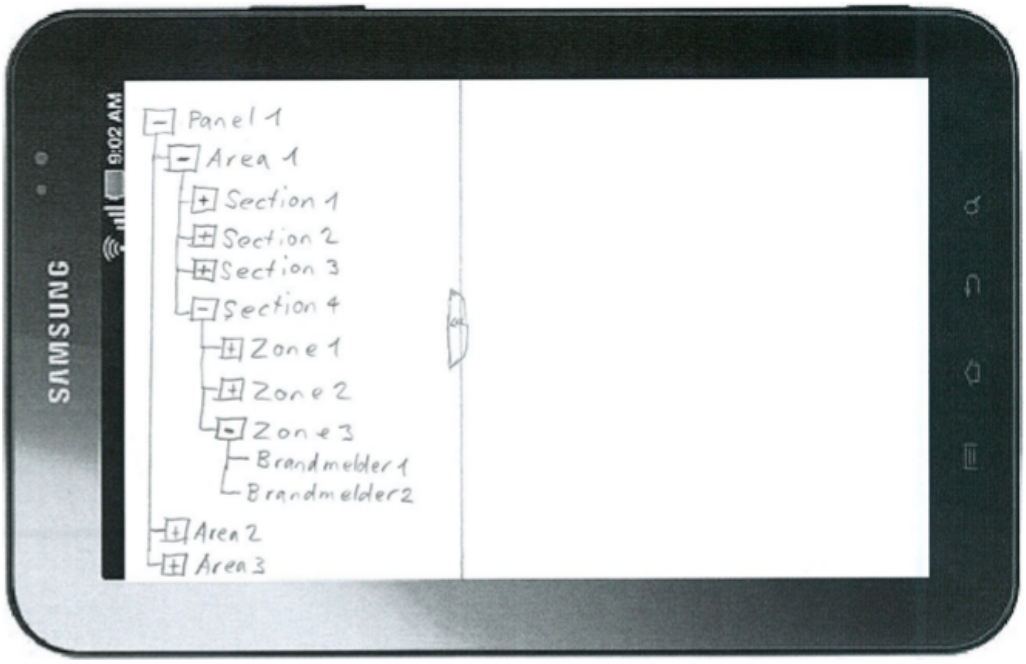


Abbildung 3.14: Aufklappbarer Baum der Brandmeldezentrale

Mit der Lasche links im Bild kann ein ausklappbarer Baum von links nach rechts ins Bild gezogen werden. Dieser Baum soll analog zum Baum in Sinteso funktionieren damit ein Wiedererkennungseffekt entsteht.

Klick auf ein Element

Berührt man ein Pluszeichen eines Element klappt sich das Element auf. Wird das Element selber angeklickt wird das Objekt ausgewählt und wie gewohnt rechts die Liste der Objekte mit ihren ausklappbaren Informationen angezeigt. Das Minuszeichen klappt die Kindelemente wieder ein.

Fehlerdetail Screen

Der Fehlerdetail Screen zeigt alle Attribute eines vorher dokumentierten Fehlers an. Dies sind VoiceMemos, Bilder und eine Beschreibungen als Text. Gleichzeitig wird dieser Bildschirm auch dazu verwendet neue Fehler zu erfassen oder bereits bestehende zu bearbeiten.

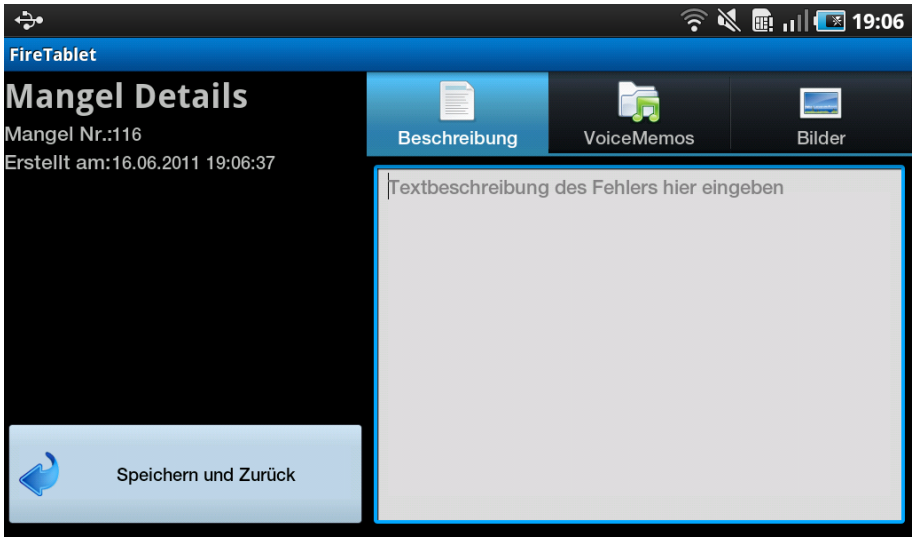


Abbildung 3.15: Detailansicht eines Fehlers. In diesem Fall sind noch keine Informationen zum Fehler gespeichert

OptionMenu

Das OptionMenu ist im Kontext der FacilityInfo aufrufbar indem die Menu Taste betätigt wird.

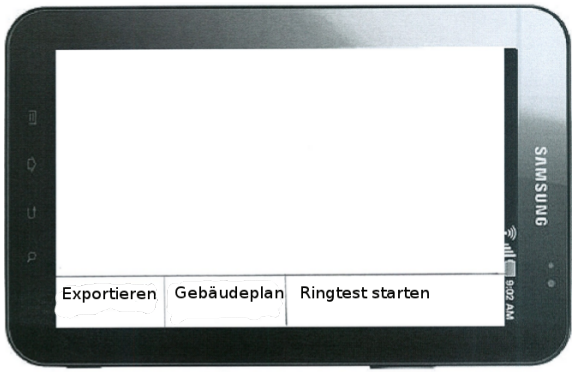


Abbildung 3.16: Menu legt sich bei Betätigung der Menutaste über das vorhandene Bild

Das Menu bietet folgende Möglichkeiten:

- **Exportieren:** Ein Testvorgang wird abgeschlossen und die Prüfdokumente werden generiert. Anschliessend werden die Dokumente einem Zentralen Server geschickt.
- **Gebäudeplan:** Der gesamte Plan des Gebäudes wird durch ein PDF-Viewer angezeigt.
- **Ringtest starten:** Das Tablet wartet auf ankommende Testalarm-Events die von der Brandmeldeanlage geschickt werden.

3.2.8 Gewünschte Anpassungen durch Kunde

Der oben beschriebene Prototyp wurde dem Kunden vorgelegt und diskutiert. Basierend auf den Änderungsvorschlägen des Kunden wurde für den FacilityInfo Bildschirm nochmal ein finaler Prototyp erstellt. Folgende Änderungen wurden besprochen:

- **Die Breadcrumbs** wie in Abschnitt Klick auf den Elementverlauf auf Seite 32 beschrieben, fallen weg. Der Benutzer kann dafür in der Navigation links anhand von Konkreten Objekte mitverfolgen welchen Pfad er gewählt hat. Diese Idee wurde bereits im Abschnitt SiteInfo Entwurf 3.2.2 aufgezeigt allerdings für den Prototyp verworfen.
- **Ein Suchfilter** tritt an die Stelle der Breadcrumbs. Die Liste wird damit immer nach den gerade eingegebenen Zeichen gefiltert.
- **Die Beschriftung** der untersten Hierarchiestufe ist mit "Sensors" unglücklich gewählt. Die richtige Beschriftung lautet "Elemente".

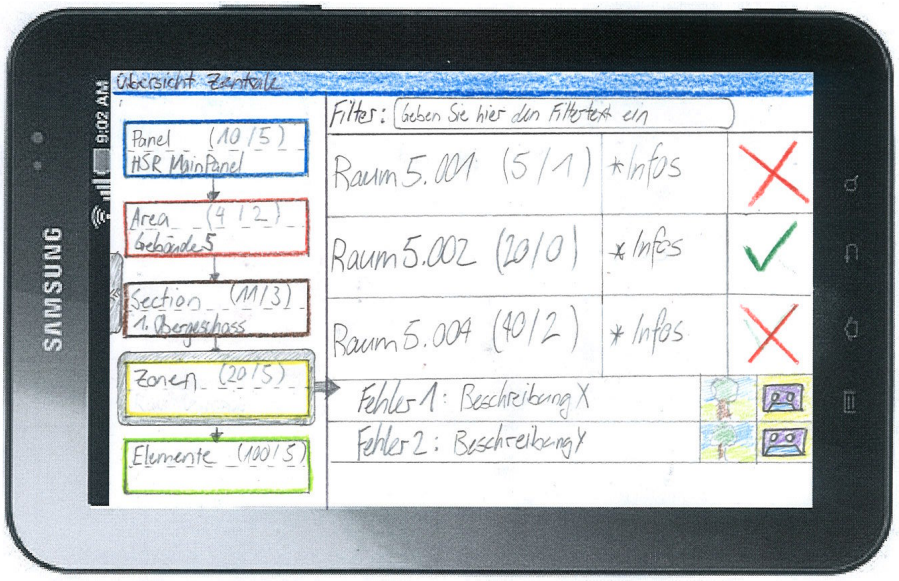


Abbildung 3.17: Finaler Prototyp von FacilityInfo mit den oben aufgelisteten Änderungen.

3.3 Limitierungen und Einschränkungen

Nachdem bis jetzt immer über Möglichkeiten und Features gesprochen wurde machen wir in diesem Abschnitt Limitierungen für die entstehende Software. Diese Limitierungen kommen durch Projekteigenheiten Vereinfachungen zustande.

- **Netzwerksicherheit und Authentisierung**
  Da die Software FireTabletPlus vor allem für Demonstrationen geschrieben wurde, wird auf Netzwerksicherheit verzichtet. Somit ist weder die Kommunikation mit dem Server verschlüsselt noch kontrolliert der Server wer gerade auf die Daten zugreift. Es sind keine Daten auf dem Server die einen Wert haben sondern nur erfunden Daten für eine Demonstration.

- Der Server kontrolliert nicht wann jemand die Daten benutzt noch werden die Daten gemerged. Sobald ein neues Tablet seine Daten zum Server sendet werden alle Daten innerhalb der gleichen Site überschrieben.
- Als Gerät wurde das vorhandene Galaxy Tap P1000 verwendet. Auf allen anderen Tablets kann nicht garantiert werden, dass die Software einwandfrei dargestellt wird. Die Chancen für eine einwandfreie Funktion auf anderen Tablets stehen zwar gut können aber nicht sicher gestellt werden. Weiter wird nur der Bildschirm nur im Querformatmodus unterstützt, so wie in den Prototypen ersichtlich.
- Als Programmsprache wird Deutsch verwendet. Eine Lokalisierung für- Englisch ist nicht Teil dieser Arbeit.

3.4 Definitive Funktionalität

Anschliessend geht dieser Abschnitt auf die definitive Funktionalität von FireTabletPlus ein und erläutert einzelne Funktionen detaillierter. Hier werden aber nur die Konzepte beschrieben. Details zur Implementierung findet man im Kapitel Implementierung von FireTabletPlus auf Seite 43. Nicht alle Features die im Abschnitt Featurekatalog auf Seite 25 festgehalten wurden, werden realisiert. Nicht weiter verfolgt wurden:

- **Site Checkliste**
  Die Checklisten um eine Anlage zu Prüfen wurden fallen gelassen da dies im wesentlichen nur eine zusätzliche Schnittstelle gewesen wäre um XML irgendwie darzustellen. Zudem mussten Prioritäten gesetzt werden um der verfügbaren Zeit gerecht zu werden.
- **Gateway Integration ausbauen**
  Wie weiter oben im Abschnitt Anpassungen durch Kundenwunsch auf Seite 26 beschrieben fand beim Gateway keine Weiterentwicklung statt weshalb nicht mehr Funktionen für das Gateway erstellt werden können.

3.4.1 Zentrale Speicherung

Diese Idee für diese Funktion wurde im Verlauf des Projekt am stärksten weiter getrieben. Anfangs nur ein nur simples zentrales Speichern von Brandmeldeanlagedaten die Idee. Man hätte also alle erfassten Bilder, Voicememos und Fehler auf dem Server gespeichert. Weil aber bald festgestellt wurde, dass man zum Beispiel für die Generierung der Inspektionreports viele weitere Informationen genötigt kam die Idee diese Informationen auch über den Server zur Verfügung zu stellen. Hinzu kamen die Gateway-Verbindungsinformationen, welche man sonst immer über das GUI eingeben müsste. Somit entsteht ein neuer Arbeitsablauf:

1. Der Servicetechniker bekommt einen neuen Auftrag in dem steht um welche Brandmeldeanlage (Site) es sich handelt und was zu tun ist.
2. Noch in der Firma verbindet der Techniker via WLAN mit dem zentralen Server und such sich dort die entsprechende Anlage.

3. Sämtliche Artefakte einer Brandmeldeanlage werden über WLAN auf das Tablet geladen.

4. Der Servicetechniker fährt vor Ort und stellt die Verbindung des Tablets mit der Brandmeldeanlage her. Da das Tablet schon alle Verbindungsinformationen geladen hat ist dies völlig unkompliziert.

5. Das Tablet lädt anschliessend die gesamte Struktur der Brandmeldeanlage über den Gateway.

6. Der Servicetechniker führt seinen Auftrag aus vor Ort aus.

7. Nachdem die Qualitätskontrolle abgeschlossen ist generiert der Techniker den Inspektionsreport und mailt ihn dem Brandmeldeanlagenbesitzer zu.

8. Um alle Infos auf dem zentralen Server zu aktualisieren synchronisiert der Servicetechniker zum Abschluss wieder das Tablet mit dem Server. Je nach Datenverbindung kann er dies gleich vor Ort machen oder erst wieder in der Firma mit dem schnellen WLAN.

Der ganze Prozess ist nochmal in einem Bild dargestellt. Die Akteure sind der Zentrale Server, das Tablet und die Brandmeldezentrale. Der Server steht bei Siemens, die Brandmeldezentrale beim Brandmeldezentralenbesitzer und das Tablet ist mobil.

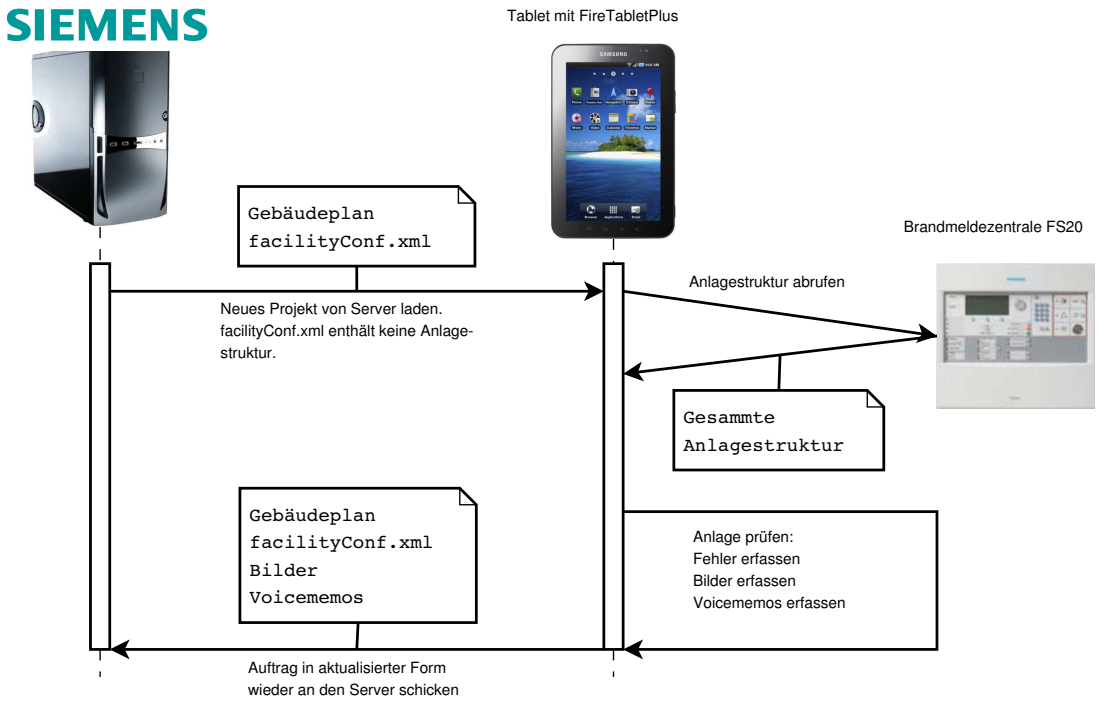


Abbildung 3.18: Prozessablauf der zentralen Datenspeicherung auf einem Server.

Datenexport in XML

In der Vorgängerversion von FireTabletPlus wurden die erfassten Fehler in CSV exportiert. Neu wird XML verwendet was an dieser Stelle begründet werden soll. Dabei geht es immer

um die in Bild 3.18 erwähnte Datei *facilityConf.xml*. In dieser Datei sind alle Informationen über eine Brandmeldeanlage gespeichert.

- **ToDo** ⇒ Der aktuelle Auftrag für den Servicetechniker.
- **CustomerAffairs** ⇒ Sämtliche Informationen über den Kunden der Brandmeldeanalge.
- **FacilityDetails** ⇒ Wichtige Technische Informationen für FireTablet.
- **Structure** ⇒ Gesamte hierarchische Struktur ein Brandmeldeanlage mit allen erfassten Fehlern.

Der Hauptnachteil von XML ist die zu hohe Redundanz die durch die immer wieder gleichen Tags entsteht. Vor allem in mobilen Applikationen sind solche unnötige Zusatzinformationen gut zu bedenken da Datenverbindungen sehr schmalbandig werden können. Trotzdem waren die Vorteile Grund genug trotzdem XML zu verwenden. Die Vorteile von XML gegenüber CSV sind:

- XML ist besser von Menschen lesbar.
- XML kann mit Schemas überprüft werden.
- Mit XML können Abhängigkeiten zu mehreren Objekten erzeugt werden.
  (Device besitzt mehrere Fehler)
- XML wird sehr gut von Java unterstützt mit Parsern und Serializer.
- Siemens verwendet in Sinteso auch XML um die SiteReports zu erstellen.

3.4.2 Prüfungsbericht Generierung

Die Prüfungsbericht Generierung wird auf zweierlei Arten vollzogen. Erstens auf eine für Maschinen leserliche Art als XML wie oben erwähnt, und zweitens in für Menschen leserlicher Art als PDF.
Der Inspectionreport ist ein Bericht an den Besitzer der Brandmeldeanlage. Nebst vielen Informationen zum Kunden selber enthält er den Ausgeführten Auftrag, eine Auflistung der gefundenen Mängel sowie eine Übersicht der Arbeitszeit welche verrechnet wird. Ein Beispiel eines [InspectionReport] befindet sich bei den referenzierten Dokumenten. Die aufgeführten Arbeitszeiten werden nicht erfasst sondern nur zu demonstrationszwecken frei erfunden.

3.4.3 Gebäudeplan anzeigen

Der Servicetechniker hat die Möglichkeit einen auf dem Tablet gespeicherten Plan zu betrachten. Im Plan sind die Positionen von Brandmeldern, Handtastern und Hörner eingezeichnet. Ein kleiner Ausschnitt eines Gebäudeplan kann im Bild 3.19 betrachtet werden.

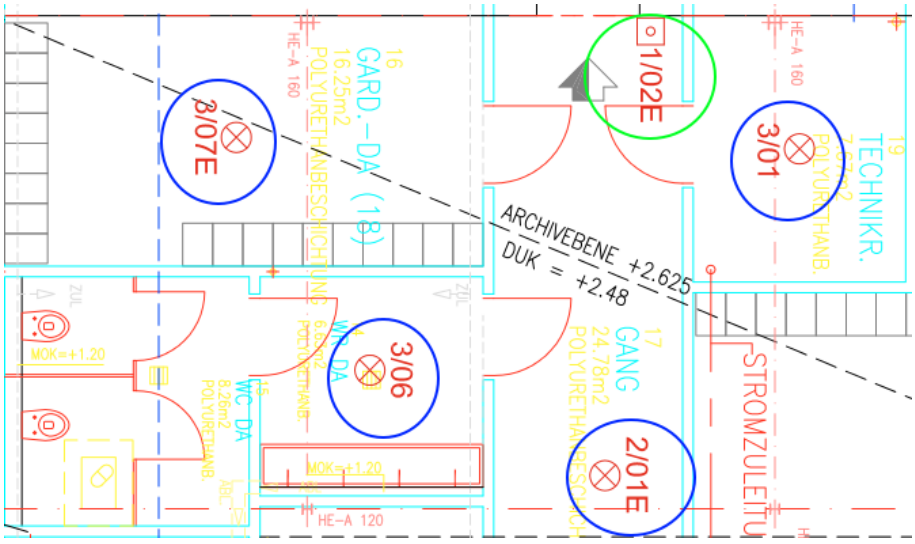


Abbildung 3.19: Ausschnitt aus einem Gebäudeplan. Brandmelder sind rot, Handtaster mit grünen Kreisen markiert.

Vom SiteInfo Screen aus kann der Benutzer die Sensoren auf dem Plan direkt fokussieren. Allerdings muss jedes Element zuerst auf dem Plan markiert werden da auf dem Plan keine Informationen über den Ort der Geräte vorhanden sind.

3.5 Abschluss der Analyse

Hiermit ist die Analyse abgeschlossen und es folgt als nächstes die Dokumentation der Implementation. Was hier säuberlich getrennt wurde entstand in der Realität vermischt. Es wurde also schon lange implementiert bevor die letzte Anforderung erfasst war. Dies liegt vor allem an der agilen Projektart bei welcher funktionsfähiger Code eine Art Leitfaden bildet. [The Agile Samurai]

4 Implementierung von FireTabletPlus

In diesem Kapitel wird auf die Implementation der Applikation eingegangen. Es werden als erstes die verschiedenen Funktionalitäten im Zusammenhang mit dem UserInterface aufgezeigt. Anschliessend wird auf die Datenhaltung eingegangen, einerseits die lokale Speicherung in der Datenbank, andererseits die zentrale Speicherung auf einem Webserver. Es wird dann die Pakagestruktur, verschiedene Klassendiagramme sowie das Domainmodell anhand von Diagrammen aufgezeigt. Zum Schluss dieses Kapitels wird auf die speziellen und kniffligeren Probleme und ihre Lösungen eingegangen.

4.1 Erläuterung der Funktionalitäten anhand des GUI

In diesem Abschnitt werden die verschiedenen Funktionalitäten von FireTabletPlus direkt anhand des Userinterfaces erläutert.

4.1.1 Navigationsprinzip

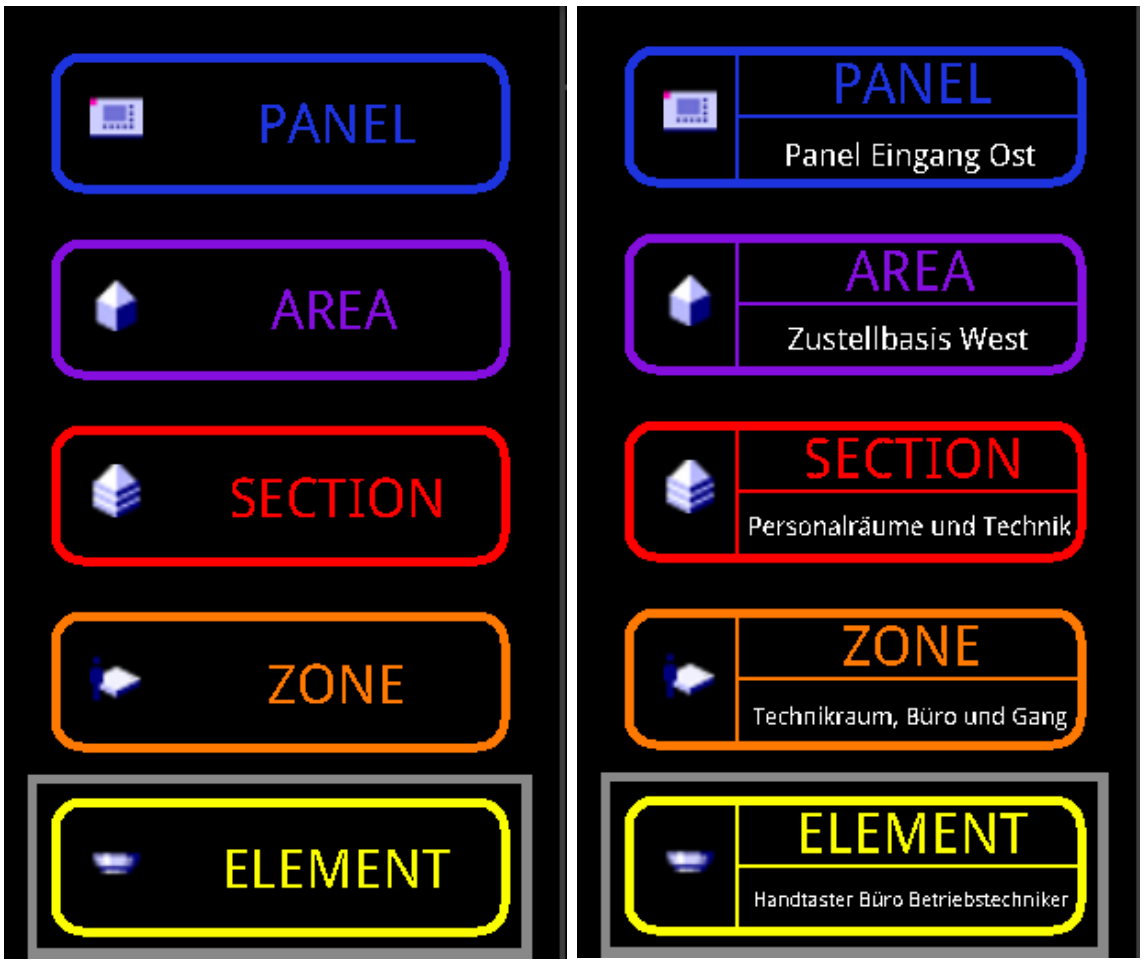


Abbildung 4.1: Veranschaulichung des Navigationsprinzip anhand der Navigationsbuttons der SiteInfo

Die Navigation der ungerichteten Baumstruktur der verschiedenen Devices folgt einer klaren Strategie. Die Navigation über die Hierarchiebuttons (siehe Bild oben) zeigt zu Beginn stets alle Devices einer Hierarchiestufe an. Dies ermöglicht zum Beispiel das Anzeigen von allen Areas einer Site. Durch einen Klick auf ein Element in der Liste wird dieses explizit Ausgewählt. Es werden dann zuerst die Oberhierarchien mit den richtigen Objekten gefüllt, sodass man nun den Pfad über die Hierarchien erkennen kann um zum ausgewählten Element zu gelangen. Ab nun wird der gewählte Pfad als Filter in die Anzeige von Elementen einfliessen. Wenn ich also zum Beispiel eine spezifische Area auswähle wird das entsprechende Objekt auf Hierarchiestufe Area und dessen Oberelement auf Stufe Panel in den Pfad aufgenommen. Wenn ich nun über den Hierarchiebutton auf Hierarchiestufe Zone navigiere werden alle Zonen Objekte angezeigt die dem bisher ausgewähltem Pfad entsprechen, also nur noch jene Zonen die im Baum Unterelemente der ausgewählten Area sind. Um die Selektion wieder aufzuheben kann auf eine Hierarchiestufe gewechselt werden die bereits eine Selektion aufweist, diese selektion wird dann gelöscht. Um den kompletten Selektionspfad zu löschen wird auf die Hierarchiestufe Panel gewechselt.

4.1.2 MainMenu

Abbildung 4.2: Funktionalitäten auf dem MainMenu Screen

1 **Site aus der Datenbank laden**
Die in der Datenbank gespeicherten Sites werden direkt in der ListView auf dem MainMenu angezeigt. Durch einen klick auf eine der Sites wird diese ausgewählt und weiter zum SiteInfo Screen gesprungen.

2 **Neue Site von Server laden**
Durch einen klick auf diesen Button wird der Server kontaktiert und die darauf liegenden Anlagen angezeigt. Siehe Abschnitt MainMenu Screen - Site laden auf Seite 46

OPTIONS **Menu-Button auf dem Gerät geklickt**
Durch klicken des Menu-Buttons auf dem Gerät öffnet sich folgendes Option-Menu:

Demo Site	Demo Connection
Server Adresse setzen	Dokumentenübersicht öffnen

- **Demo Site:** Startet SiteInfo Screen mit einer Demo Site.
- **Demo Connection:** Versucht über eine vorkonfigurierte Verbindung den BacNet-Gateway zu erreichen und darüber eine neue Site zu laden.
- **Server Adresse setzen:** Ändert die konfigurierte IP-Adresse des zentralen Servers.

– **Dokumenteübersicht öffnen:** Startet DocumentOverview Screen und zeigt alle generierten Dokumente an.

4.1.3 MainMenu Screen - Site laden

Abbildung 4.3: MainMenu - Site laden

1 **Anzeige des Verbindungsstatus**
Hier wird der momentane Verbindungsstatus sowie das zu transferierende Datenvolumen angezeigt. Es wird dann die erwartete Download Zeit berechnet und anhand der berechneten Zeit ein Ratschlag abgegeben. Diesen Ratschlag erkennt man anhand der Farbe beim Verbindungsstatus. Grün bedeutet dass es kein Problem ist diese Datenmenge über die momentane Verbindung herunterzuladen. Orange bedeutet dass es einige Zeit in Anspruch nehmen könnte, grundsätzlich aber immer noch sinnvoll ist die Daten mit der aktuellen Verbindung zu laden. Wenn der Ratschlag Rot zeigt ist es nicht empfohlen die Daten mit der aktuellen Verbindung herunterzuladen. Es ist dann sinnvoll die Daten über ein naheliegendes Wireless LAN oder im Geschäft selber herunterzuladen.

2 **Eine spezifische Site vom Server laden**
Durch einen Klick auf ein spezifisches Element in der Liste wird der Download-Vorgang der entsprechenden Site begonnen. Siehe MainMenu Screen - Site laden (Fortschrittsanzeige auf Seite 47

BACK **Back-Button auf dem Gerät geklickt**
Mit dem Klick auf den Back-Button des Geräts wird der Dialog geschlossen und der User wird in den MainMenu zurückgeführt.

4.1.4 MainMenu Screen - Site laden (Fortschrittsanzeige

Abbildung 4.4: MainMenu - Site laden (Fortschritt)

1 **Download abbrechen**
Während dem Download bleibt dem Benutzer nur noch übrig den Download abzubrechen oder zu warten bis der Download komplett ist. Für das Abbrechen des Downloads siehe MainMenu Screen - Site laden auf Seite 46, beim Beenden des Downloads wird der Benutzer auf den SiteInfo Screen auf Seite 48 weitergeleitet.

BACK **Back-Button auf dem Gerät geklickt**
Mit dem Klick auf den Back-Button des Geräts wird der Filetransfer abgebrochen und der User wird in den MainMenu Screen - Site laden zurückgeführt.

4.1.5 SiteInfo Screen

Abbildung 4.5: Funktionalitäten auf dem SiteInfo Screen

1 **NavigationsPanel**
Dieser Teil des Bildschirms ist für die Navigation über die Hierarchie reserviert.

2 **Navigation über Hierarchie Buttons**
Durch einen Klick auf einen der Hierarchie Buttons kann auf die entsprechende Hierarchie gesprungen werden. Es werden nun Devices dieser Hierarchie angezeigt, gefiltert nach der Auswahl der darüber liegenden Hierarchien. Siehe **??** für weitere Informationen zum Navigationsprinzip.

3 **Deviceanzeige**
Dieser Teil des Bildschirms zeigt die entsprechenden Devices gemäss der aus-ge-wählten Hierarchie und der selektierten Devices an. Siehe **??** für weitere Informationen zum Navigationsprinzip.

4 **Infos anzeigen**
Durch einen Klick auf diesen Button eines Elements werden die Infos zu diesem konkreten Element angezeigt. Siehe SiteInfo Screen in der Infoansicht auf Seite 52

5 **Kommentare anzeigen / erfassen**
Durch einen Klick auf diesen Button eines Elements werden die Kommentare zu diesem konkreten Element angezeigt. Siehe SiteInfo Screen in der Kommentaransicht auf Seite 51.

6 **Fehler anzeigen / dokumentieren**
Durch einen Klick auf diesen Button eines Elements werden die Fehler zu diesem konkreten Element angezeigt. Siehe SiteInfo Screen in der Fehleransicht auf Seite 50.

7 **Zu Element navigieren**
Durch den Klick auf ein Element aus der Liste (Also nicht auf ein Info- Fehler- oder Kommentar- Button) wird dieses ausgewählt. Es wird die Hierarchie eine Stufe tiefer gesetzt und die Auswahl auf dieses Element gesetzt (inklusive der übergeordneten Hierarchiestufen) sodass nun nur die Kinder dieses ausgewählten Elements angezeigt werden. Siehe Navigationsprinzip auf Seite 44 für weitere Informationen betreffend dem Navigationsprinzip. Falls sich das ausgewählte Element auf der Hierarchiestufe ELEMENT befindet, wird auf den SiteInfo Screen in der Planansicht (Seite 52) gewechselt.

8 **Filtertext eingeben**
In diesem Textfeld kann ein Filtertext eingegeben werden. Es ist ein Freitext der anschliessend auf den momentan angezeigten Devices gesucht wird. Es bleiben nur jene Devices in der Liste die den Text enthalten. Der Hintergrund des Filters wird dabei gekennzeichnet um zu symbolisieren dass der Filter aktiv ist.
Zur Zeit werden folgende Attribute auf den Devices durchsucht:

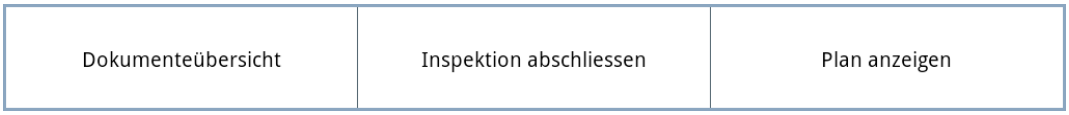
- Description / Customertext
- Object-identifier
- Object-name
- Device-type

9 **Filter löschen**
Mit diesem Button kann der Filter wieder gelöscht werden. Der Text der im Filter Textfeld steht wird dabei gelöscht und die Markierung des Hintergrunds aufgehoben.

10 **Anzeige der Fehler auf Unterelementen**
Hier wird die Anzahl Fehler in den Unterelementen angezeigt. So kann man relativ schnell durch die einzelnen Hierarchiestufen der Site navigieren um einen dokumentierten Fehler zu finden.

11 **Wischgeste nach rechts**
Durch eine Wischgeste nach Rechts kann direkt aus dem SiteInfo Screen herausgesprungen werden um wieder auf den MainMenu (Seite 45) zurückzukehren.

OPTION **Option-Button auf dem Gerät geklickt**
Durch einen Klick auf den Menu-Button auf dem Gerät wird folgendes OptionMenu angezeigt:



- **Dokumenteübersicht:** Wechselt in den DocumentOverview Screen.
- **Inspektion abschliessen:** Wechselt in den DocumentOverview Screen und generiert einen neuen InspectionReport.
- **Plan anzeigen:** Startet die Ansicht des kompletten PDF Plans in einem externen PDF-Viewer.

BACK **Back-Button auf dem Gerät geklickt**
Wenn auf dem SiteInfo Screen der Back-Button auf dem Gerät geklickt wird werden alle vorherigen Aktionen in einzelnen Schritten rückgängig gemacht. So kann eine falsche Navigation oder die falsche Auswahl von Devices schnell und Schritt für Schritt rückgängig gemacht werden. Um schnell ganz zurück auf das Hauptmenü zu kommen wird die Wischgeste empfohlen.

4.1.6 SiteInfo Screen in der Fehleransicht

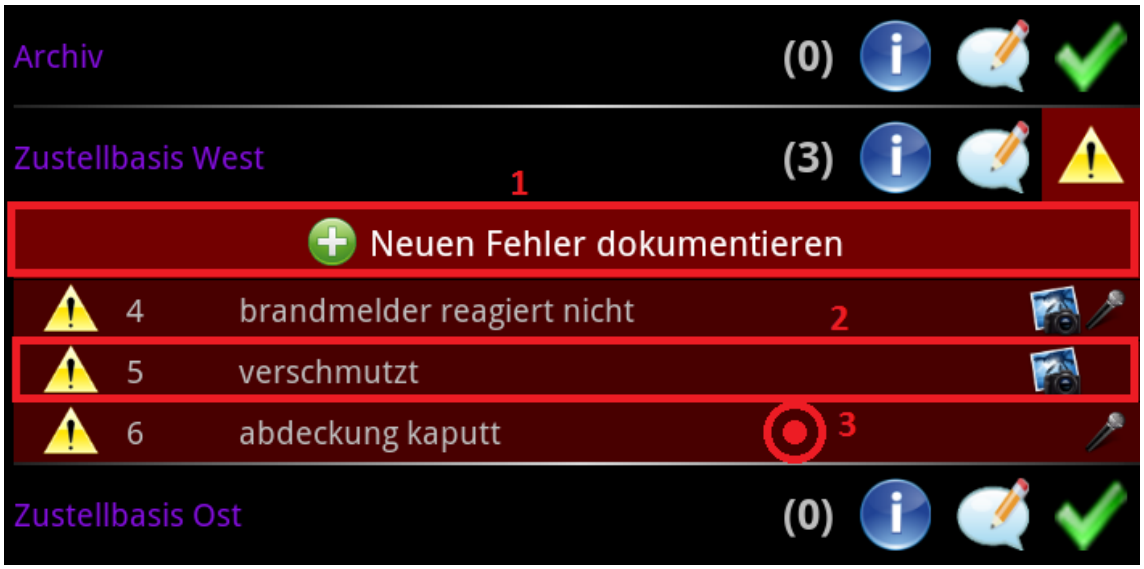


Abbildung 4.6: Funktionalitäten auf dem SiteInfo Screen in der Fehleransicht

1 **Neuen Fehler dokumentieren**
Durch Klick auf "Neuen Fehler dokumentieren" wird ein neuer Fehler auf dem Device erfasst und in die Fehleransicht gesprungen. Siehe Faultdetail Screen auf Seite 53.

2 **Spezifischen Fehler ändern**
Durch den Klick auf einen spezifischen Fehler in der Liste kann dieser geändert werden. Es wird dann abermals in den Faultdetail Screen (Seite 53) gesprungen und die Daten aus dem Fehler geladen und zur Änderung freigegeben.

3 **Fehler löschen**
Durch einen Long-Klick auf einem Fehler in der Liste wird ein Dialog geöffnet in dem der Benutzer abermals das Löschen des Fehlers bestätigen muss. Bei Bestätigen des Löschens wird der Fehler aus der Liste gelöscht, andernfalls wird der Dialog beendet und nichts gelöscht.

4.1.7 SiteInfo Screen in der Kommentaransicht

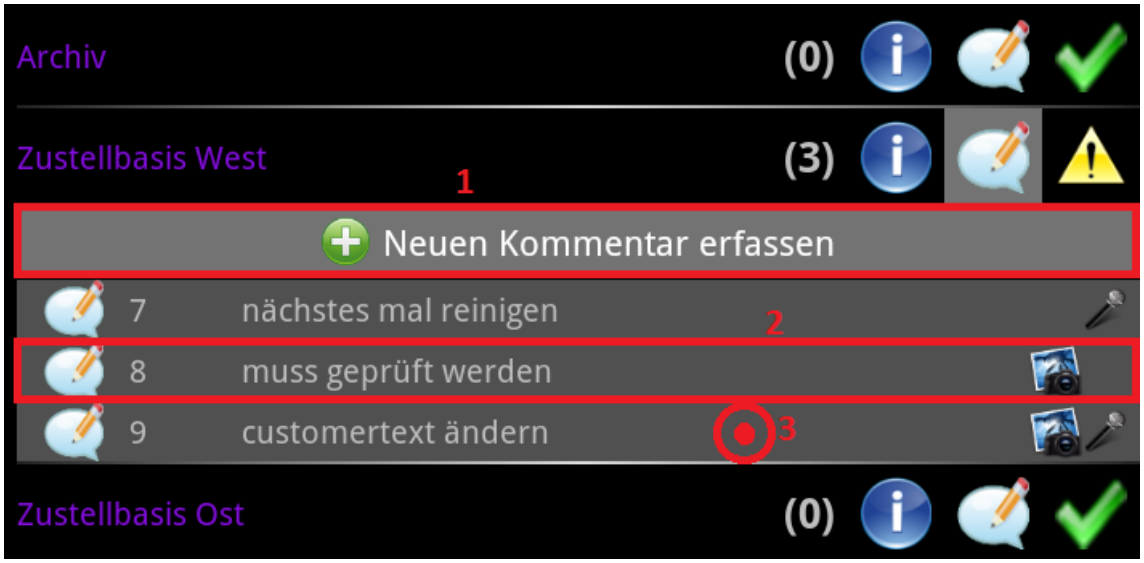


Abbildung 4.7: Funktionalitäten auf dem SiteInfo Screen in der Kommentaransicht

1 **Neuen Kommentar erfassen**
Durch Klick auf "Neuen Kommentar erfassen"wird ein neuer Kommentar auf dem Device erfasst und in den Faultdetail Screen (Seite 53) gesprungen.

2 **Spezifischen Kommentar ändern**
Durch den Klick auf einen spezifischen Kommentar in der Liste kann dieser geändert werden. Es wird dann abermals in den Faultdetail Screen (Seite 53) gesprungen, die Daten aus dem Kommentar geladen und zur Änderung freigegeben.

3 **Kommentar löschen**
Durch einen Long-Klick auf einem Kommentar in der Liste wird ein Dialog geöffnet in dem der Benutzer abermals das Löschen des Kommentars bestätigen muss. Bei Bestätigen des Löschens wird der Kommentar aus der Liste gelöscht, andernfalls wird der Dialog beendet und nichts gelöscht.

4.1.8 SiteInfo Screen in der Infoansicht

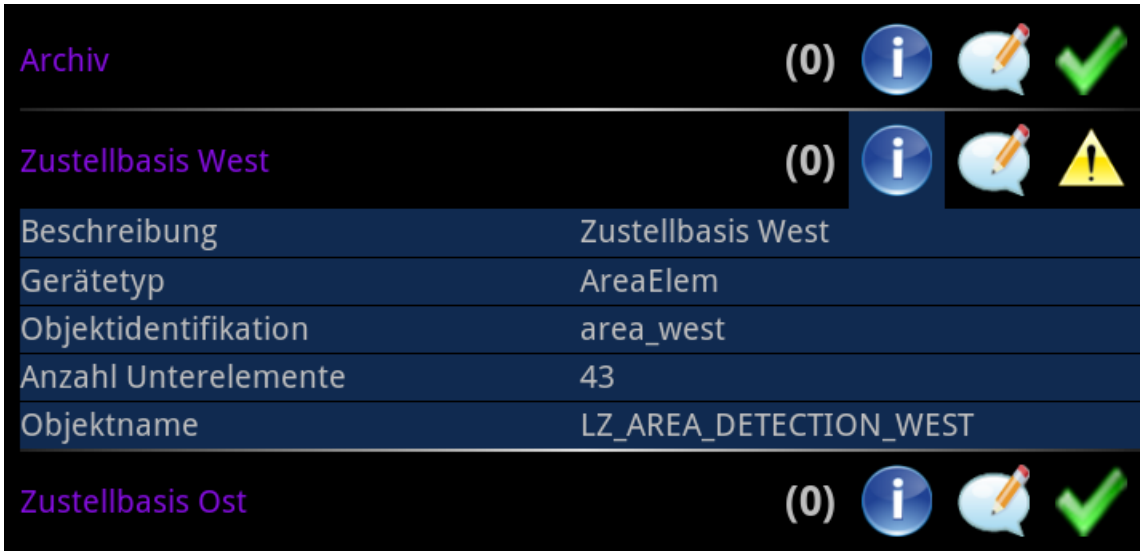


Abbildung 4.8: Funktionalitäten auf dem SiteInfo Screen in der Infoansicht

4.1.9 SiteInfo Screen in der Planansicht

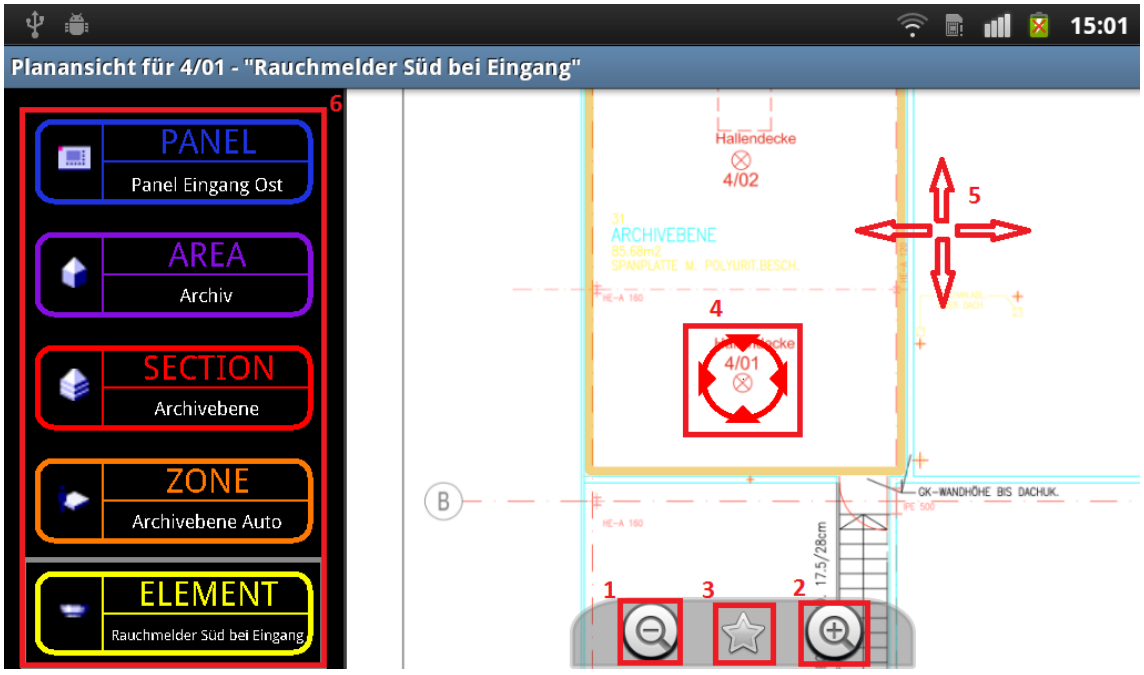


Abbildung 4.9: Funktionalitäten auf dem SiteInfo Screen

1 **Zoom - kleiner**
Durch einen Klick auf diesen Button wird die Ansicht skaliert sodass der Plan nun kleiner erscheint. Jeder Klick verringert die Skalierungsstufe um 0.5 und es kann maximal bis auf eine Skalierungsstufe von 0.5 weggezoomt werden.

2 **Zoom - grösser**
Durch einen Klick auf diesen Button wird die Ansicht skaliert sodass der Plan nun grösser erscheint. Jeder Klick erhöht die Skalierungsstufe um 0.5 und es kann bis auf eine Skalierungsstufe von 4 herangezoomt werden.

3 **Markierungsmodus einschalten**
Durch einen Klick auf den Markierungsbutton wird der Markiermodus eingeschalten. Dadurch kann die Markierung eines Devices neu gesetzt werden. Sobald der Markiermodus eingeschalten ist kann durch einen einfachen Klick auf den Plan die neue Position des Devices gesetzt werden. Die Position wird dann direkt neu gesetzt und angezeigt, der Markiermodus wird anschliessend wieder beendet.

4 **Markierung für den Device**
Diese Markierung zeigt die aktuell konfigurierte Position des ausgewählten Devices. Sie passt sich dem aktuellen Skalierungsgrad an und lässt dsich über den Markierungsmodus neu setzen.

5 **Wischgesten zum verschieben des Plan**
Der Plan lässt sich durch Wischgesten auf dem Plan verschieben. Es wurde darauf geachtet dass die gewohnte Bedienung, wie man sie zum Beispiel von Google Maps her kennt, übernommen wurde

6 **Verlassen der Planansicht über Hierarchie**
Durch einen Klick auf einen der Hierarchie Buttons wird erneut wie gewohnt die Liste angezeigt und aus der Planansicht in die normale Ansicht zurückgewechselt.

4.1.10 Faultdetail Screen

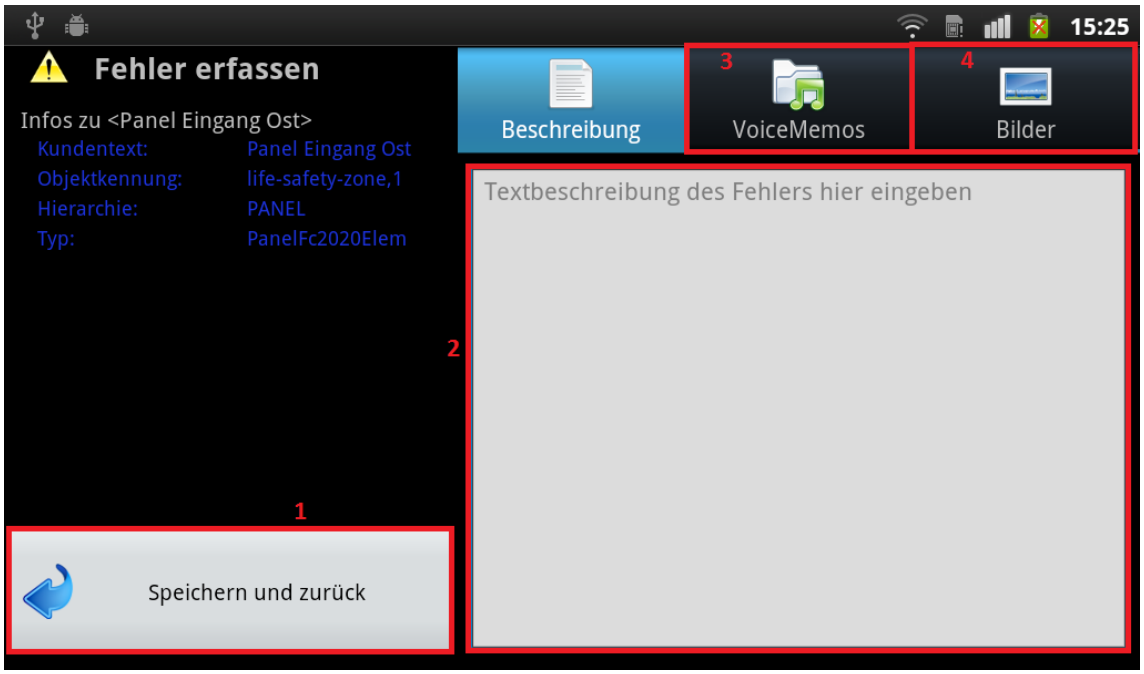


Abbildung 4.10: Funktionalitäten auf dem Faultdetails Screen

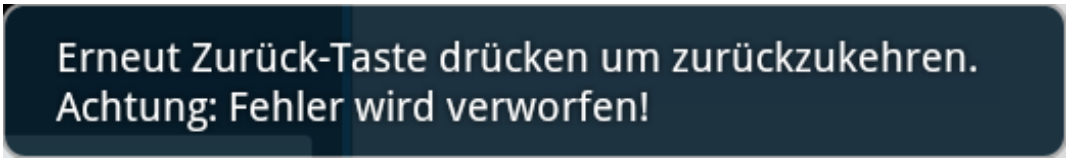
1 **Fehler / Kommentar Speichern und zurückkehren**
Durch den Klick auf SSpeichern und zurück"wird der bearbeitete Fehler oder Kommentar gespeichert und der Benutzer wird zurück auf den SiteInfo Screen (Seite 48) geleitet.

2 **Textuelle Beschreibung anpassen**
Hier kann durch die Aktivierung der Textbox die textuelle Beschreibung angepasst werden. Die Änderungen werden erst übernommen wenn der Button SSpeichern und zurück"geklickt wird.

3 **Zum VoiceMemo Reiter wechseln**
Durch einen Klick auf diesen Reiter wird auf den Faultdetails Screen VoiceMemo und Picture Ansicht (Seite 55) gewechselt.

4 **Zum Picture Reiter wechseln**
Durch einen Klick auf diesen Reiter wird auf den Faultdetails Screen VoiceMemo und Picture Ansicht (Seite 55) gewechselt.

BACK **Back-Button auf dem Gerät geklickt**
Mit dem Klick auf den Back-Button des Geräts wird zuerst eine Meldung auf dem Bildschirm angezeigt. Der Benutzer hat nun drei Sekunden Zeit erneut auf den Back-Button zu klicken um das Ändern des Fehlers abzubrechen, die gemachten Änderung am Fehler zu verwerfen und in den SiteInfo Screen zurückzukehren. Sobald die drei Sekunden abgelaufen sind wird erneut erst die Meldung angezeigt, bevor die Änderungen am Fehler verworfen werden.



4.1.11 Faultdetails Screen VoiceMemo und Picture Ansicht

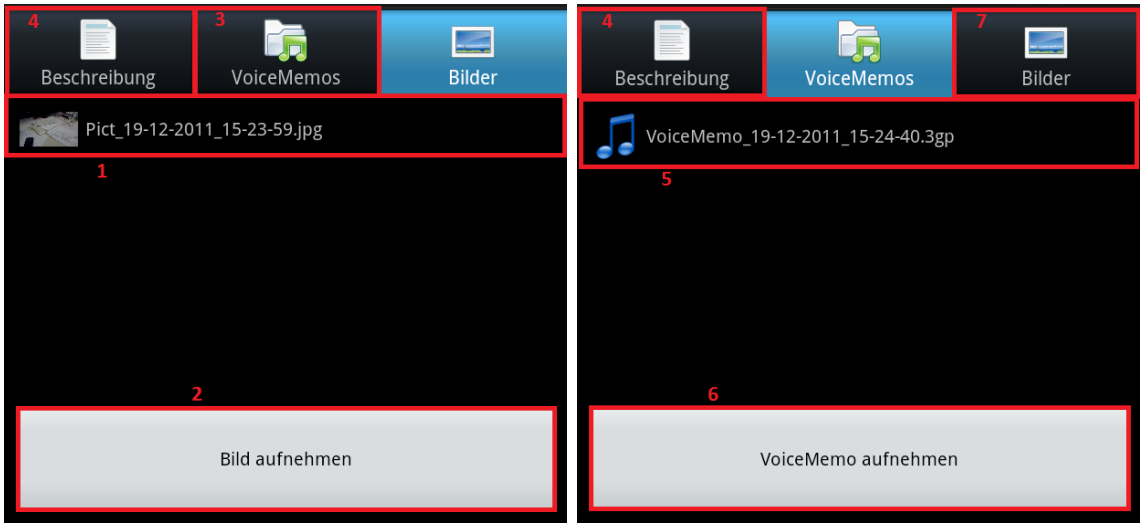


Abbildung 4.11: Die Funktionalität auf den beiden Reitern Picture und VoiceMemo.

1 **Bild ansehen**
Durch den Klick auf ein spezifisches Element in der Bilderliste kann ein Bild angezeigt werden. Dafür wird erneut eine Anfrage an das Android System gestellt um ein geeignetes Programm zu finden.

2 **Bild aufnehmen**
Durch den Klick auf "Bild aufnehmen"wird abermals das Android System nach einer geeigneten Kamera Applikation angefragt. Diese übernimmt dann das Aufnehmen des Bildes und liefert es anschliessend an unsere Applikation zurück. Dort wird es dann in der Liste angezeigt.

3 **Zum Reiter VoiceMemos wechseln**
Durch den Klick auf den Reiter VoiceMemos wird auf die VoiceMemo Liste gewechselt. Dort werden alle VoiceMemos aufgelistet die zu diesem Fehler gehören.

4 **Zum Reiter Beschreibung wechseln**
Durch den Klick auf den Reiter Beschreibung wird zurück auf die textuelle Beschreibung des Fehlers gewechselt.

5 **VoiceMemo auswählen**
Durch den Klick auf ein Element in der VoiceMemo Liste wird das entsprechende VoiceMemo abgespielt. Dies geschieht über die Lautsprecher des Geräts. Während dem Abspielen kann die Medienlautstärke angepasst werden.

6 **VoiceMemo aufnehmen**
Durch den Klick auf diese Schaltfläche kann ein neues VoiceMemo aufgenommen werden. Beim ersten Klick auf die Schaltfläche wird das Aufnahmegerät gestartet und die Aufnahme begonnen. Beim erneuten klicken der Schaltfläche wird die Aufnahme gestoppt und in der Liste angezeigt.

7 **Zum Reiter Bilder wechseln**
Durch den Klick auf den Reiter Bilder wird die Bilderliste angezeigt. Dort werden alle Bilder angezeigt die zum aktuellen Fehler oder Kommentar gehören.

4.1.12 DocumentOverview Screen

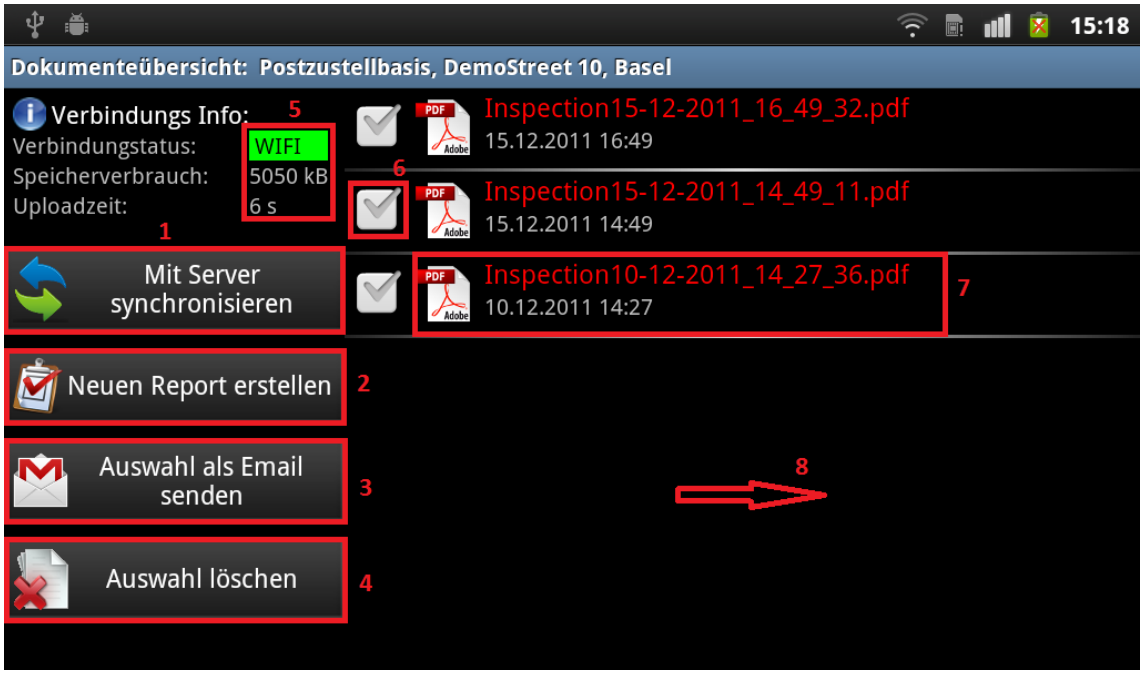


Abbildung 4.12: Funktionalitäten auf dem DocumentOverview Screen

1 **Die ausgewählte Site mit dem Server synchronisieren**
Durch den Klick auf "Mit Server synchronisieren"wird die Site auf der man aktuell arbeitet auf den zentralen Server hochgeladen. Es wird dann ein Dialog geöffnet auf dem der Benutzer den Fortschritt des Uploads sieht und den Prozess abbrechen kann.

2 **Einen neuen Testreport erstellen**
Hier kann ein neuer Report generiert werden. Es werden dann alle erforderlichen Daten aus der Site gelesen und mit diesen Daten das PDF generiert.

3 **Dokumente als E-Mail versenden**
Hier können ausgewählte Dokumente als E-Mail versendet werden. Es wird dabei eine Anfrage an das Android System (ein sogenannter Intent) nach einem geeigneten E-Mail Programm gestellt. Der Benutzer hat dann die Möglichkeit eines der vorgeschlagenen Programme zu verwenden oder ein eigenes E-Mail Programm aus dem Android Market herunterzuladen.

4 **Ausgewählte Dokumente löschen**
Hier können ausgewählte Dokumente aus der Liste und vom Dateisystem entfernt werden. Diese Aktion kann nicht Rückgängig gemacht werden.

5 **Verbindungsstatus**
In diesem Teil des Screens sieht man den aktuellen Verbindungsstatus sowie die Grösse der Hochzuladenden Dateien. Aufgrund dieser Werte färbt sich der (im Bild grüne) Balken entweder rot gelb oder grün, je nach Art der Verbindung und dem Datenvolumen.

6 **Auswählen eines Dokumentes**
Durch das Klicken der Checkbox kann ein Dokument für weitere Operationen markiert werden. Es kann anschliessend zum Beispiel per Email versendet oder gelöscht werden.

7 **Ein Dokument zur Anzeige öffnen**
Durch einen Klick auf das Element selbst kann das Dokument in einem externen PDF Viewer angezeigt werden. Es wird dabei erneut eine Anfrage an das Android System gestellt um ein geeignetes Programm zu finden.

8 **Wischgeste nach Rechts zur Rückkehr auf den SiteInfo Screen**
Durch eine Wischgeste nach Rechts kann direkt zurück in den SiteInfo Screen (Seite 48) gesprungen werden

4.2 Datenhaltung

In diesem Kapitel wird die Datenhaltung erläutert. Es wird dabei die lokale Datenhaltung in einer Datenbank von der serverseitigen Datenhaltung unterschieden.

4.2.1 Server

Die serverseitige Datenhaltung basiert auf einem einfachen Webserver mit PHP Laufzeitumgebung. Es wird im Folgenden auf diese drei Komponenten näher eingegangen: Webserver, PHP Scripts und Dateisystem.

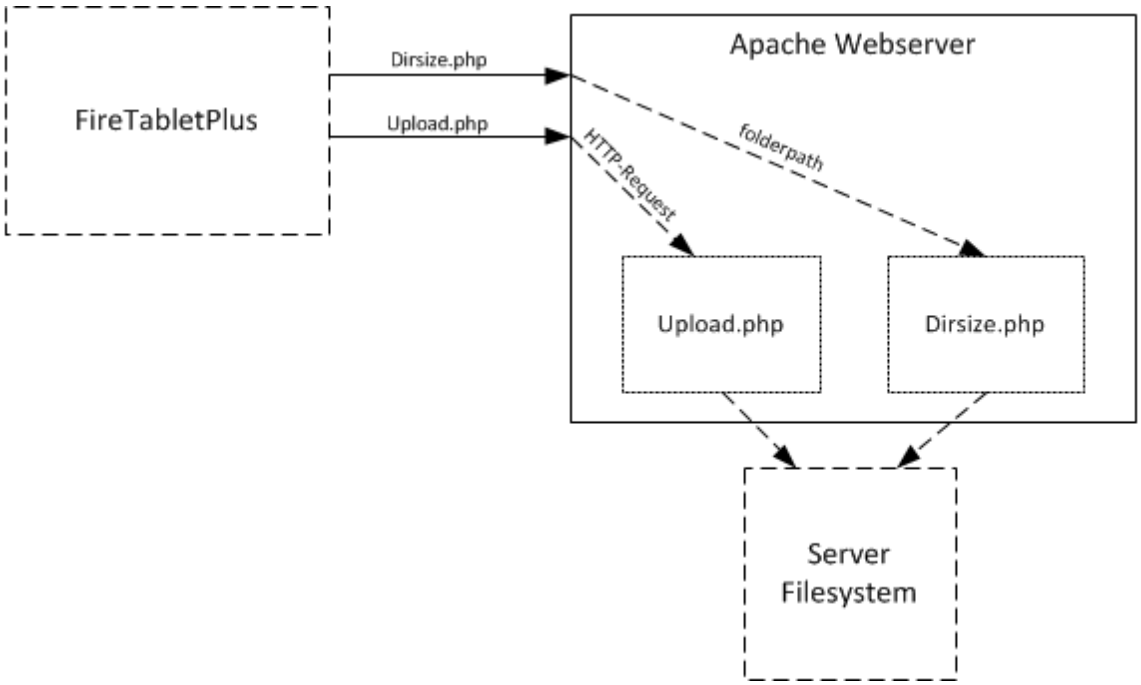


Abbildung 4.13: Übersicht Serverzugriffe

Webserver

Der Webserver besteht aus einem Apache XAMPP Webserver mit PHP Laufzeitumgebung. Er wurde auf unserem ursprünglichen GIT Server installiert, wodurch er durch eine statische IP-Adresse auch aus dem Internet zur Verfügung steht.

Upload Script (upload.php)

Das Upload Script wurde in der Scriptsprache PHP geschrieben. Es wird auf der Clientseite, also in unserer FireTabletPlus Applikation, ein HTTP Request zusammengesetzt und das gewünschte Upload-Directory als HTTP-Header Eintrag mitgeschickt. Das File wird dann wie gewohnt im Body des HTTP Requests übertragen. Unser Script nimmt diesen Request entgegen, liest den Header Eintrag heraus und beginnt den Upload des Files. Das hochgeladene File wird anschliessend in das entsprechende Verzeichnis verschoben.

Verzeichnisgrösse Script (dirsize.php)

Das Berechnen der Grösse eines Verzeichnisses wird aus Performanzgründen Serverseitig durchgeführt. Dafür ist ein PHP Script dirsize.php verantwortlich. Das Script erwartet den GET-Parameter "folder". Der übergebene Verzeichnispfad spezifiziert das Verzeichnis für das die Grösse auf dem Filesystem berechnet werden soll. Als Pfad wird direkt der Site-Ordner erwartet. Der interne Pfad zum Directory in dem alle Sites gespeichert werden ist dabei im Script selber hinterlegt.

Dateisystem

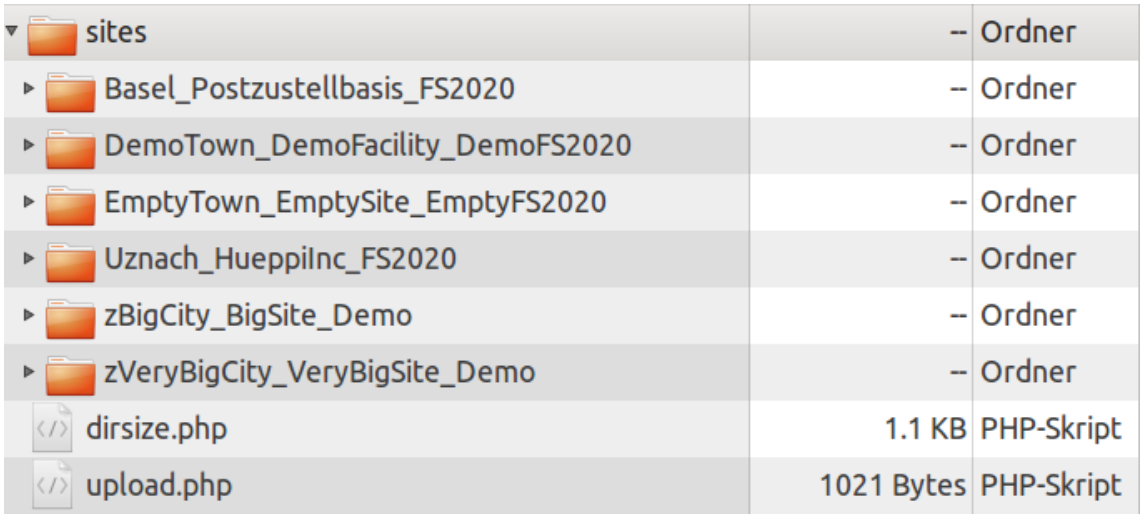


Abbildung 4.14: Ordnerstruktur auf dem Server

Die Ordnerstruktur auf dem Server zeigt die Scripte "dirsize.phpünd üpload.php"mit dem Datenordner ßites/äuf der obersten Ebene an. Im Ordner sites/ werden alle abgelegten Sites gespeichert, auf der Abbildung oben sieht man unsere DemoSites die in dem Ordner abgelegt sind. Grundsätzlich könnte der Datenordner auch ausserhalb des über den webserver ansprechbaren Verzeichnisses liegen, allerdings machen wir uns mithilfe des Apache Directorylisting das Leben etwas leichter.

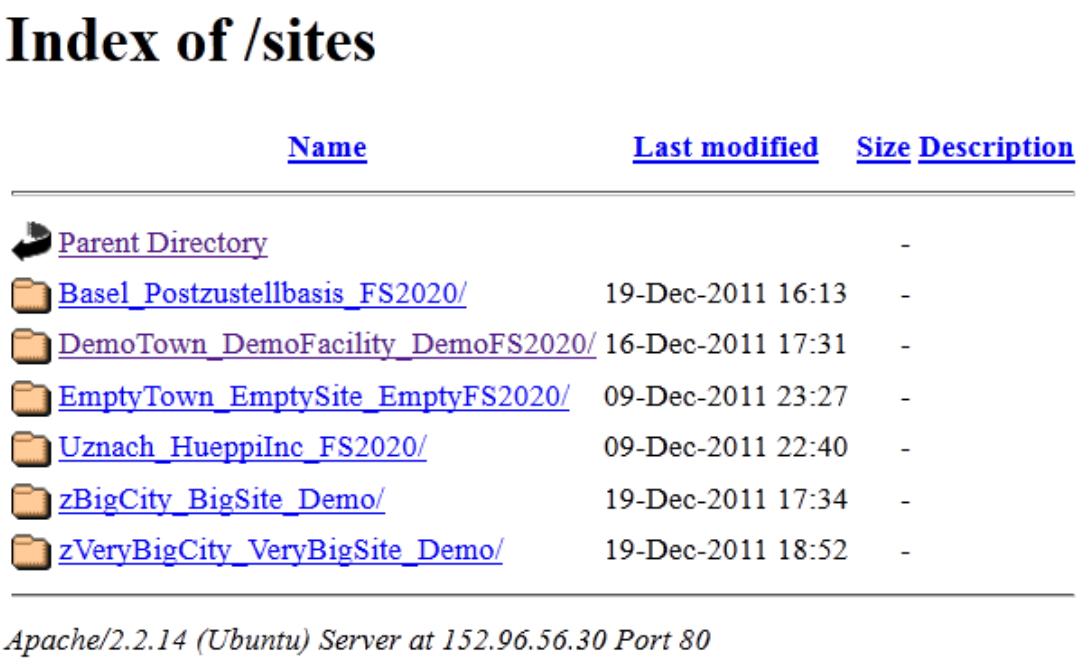


Abbildung 4.15: Apache Directorylisting auf dem Server

Durch den Umstand, dass unsere Sites im Webordner liegen, können wir nun mithilfe des Directorylistings direkt auf die Daten der Sites zugreifen. Ein mögliches Problem welches hier auftaucht ist offensichtlich: Die Daten sind für jeden frei Verfügbar. Falls in Zukunft an dieser Lösung festgehalten wird muss auf jeden Fall ein Securitymechanismus eingeführt werden. Siehe dazu auch Limitierungen und Einschränkungen auf Seite 38.

4.2.2 Lokal

DB4O Datenbank

Für unsere lokale Datenhaltung verwenden wird die DB4O Datenbank. Die Wahl unserer Datenbank begründet sich auf der einfachen Persistierung von Objektstrukturen. Da wir in unserer Applikation ohnehin einen kompletten Objektgraphen kreieren und mit diesem Arbeiten ist es relativ simpel diesen direkt in die Datenbank zu speichern. Die Möglichkeit ein Panel zu persistieren und dann dynamisch alle Objekt, die von diesem Panel aus referenziert werden mit zu speichern bringt einen enormen Vorteil, kann aber auch zu etwas merkwürdigen Nebenwirkungen führen wenn plötzlich Objekte persistiert werden ohne dass man diese explizit persistiert. Die grundsätzliche Strategie der Persistenz in unserer Applikation sieht folgendermassen aus: Wir persistieren beim herunterladen der Site das Panel einmalig in der Datenbank. Durch die Konfiguration von CascadeOnUpdate und einer updateDepth bis auf 10 stufen tief, wird das Komplette Panel, alle Devices und alle Fehler einmalig in die Datenbank geschrieben. Ab nun werden allerdings nur noch sehr spezifische updates von Objekten durchgeführt. Beim Erfassen eines neuen Fehlers wird der Fehler (nachdem er komplett erfasst aus der FaultDetailsActivity zurückkommt) dem Device angehängt und anschliessend das Device auf der Datenbank updated. Dadurch wird der neue Fehler automatisch mit persistiert. Wenn wir jedoch einen Fehler nur ändern werden lediglich die Attribute des der Datenbank bereits bekannten Fehlers geändert und der Fehler persistiert. Durch diese Strategie achten wir sehr streng darauf dass keine unnötigen Datenbankzugriffe geschehen und jeweils nur die Objekte persistiert werden, die uach geändert wurden.

Android saveInstanceState

In Activities die keinen Zugriff auf die Datenbank haben wird die Erhaltung der Daten beim wechseln der Activity oder anderen Unterbrüchen wie sie in Android auftreten können über die Android eigene onSaveInstanceState-Methode durchgesetzt. Diese Methode wird vom Android System aufgerufen sobald eine Activity zur Seite gelegt wird. Man hat dort die Möglichkeit Daten in einem Bundle zu speichern. Dieses Bundle wird dann beim hervorholen der Activity wieder übergeben und man kann die Daten wieder entsprechend herausholen. Dies wird vorallem in der Activity Faultdetails so umgesetzt.

4.3 Software Design

In diesem Kapitel wird auf das Design der Software eingegangen. In einem ersten Teil wird das Domainmodell aufgezeigt und erläutert. Aschliessend wird die Struktur und die Abhängigkeiten unter den Packages aufgezeigt und erklärt. Zum Schluss werden alle Packages als separate Klassendiagramme dargestellt und dazu die wichtigsten Klassen hervorgehoben und vorgestellt.

4.3.1 Domainmodell

In diesem Abschnitt wird das Domain Modell aufgezeigt und erläutert.

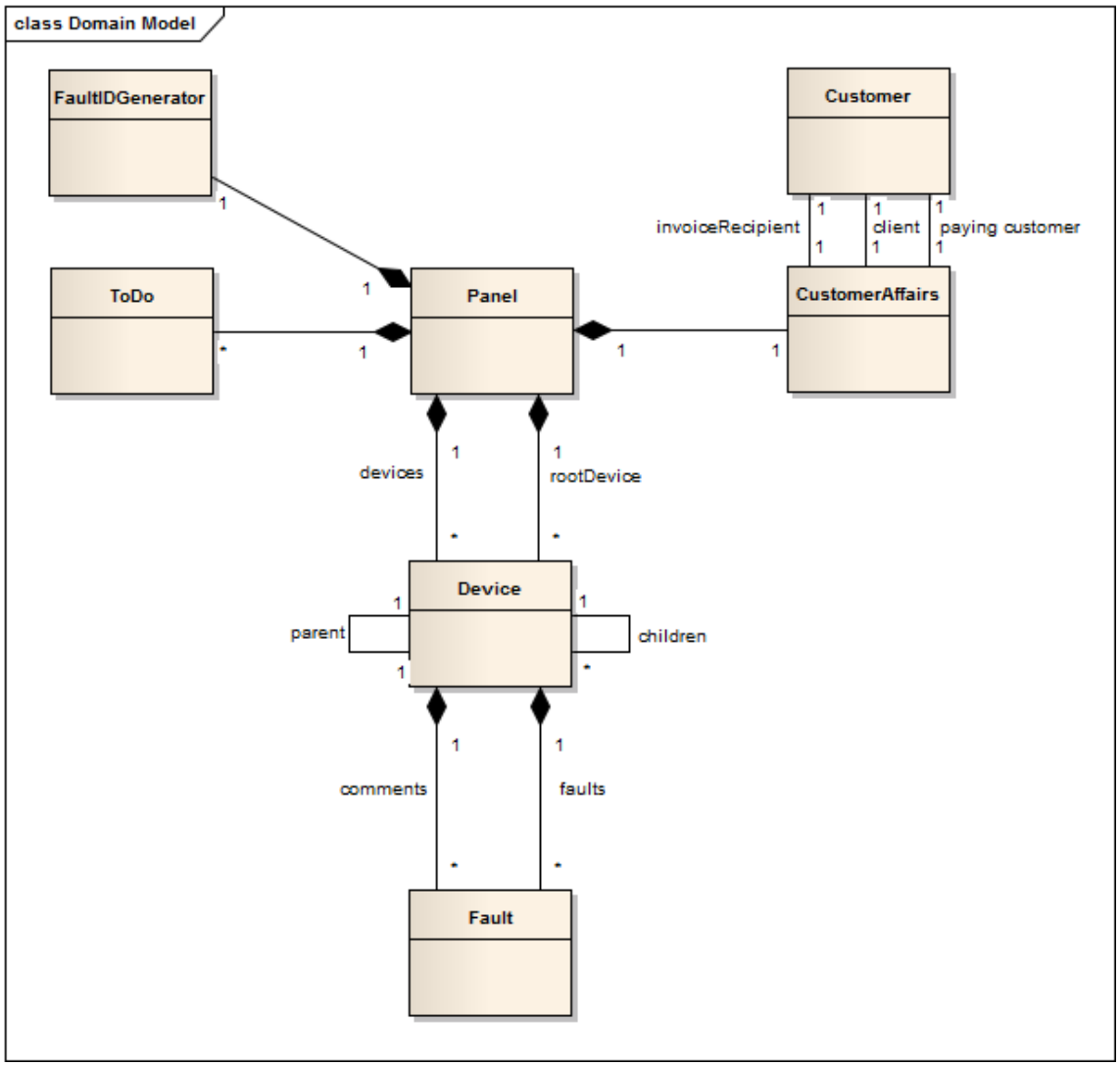


Abbildung 4.16: Struktur der PackagesDas Domainmodel von FireTabletPlus

Das Domainmodell von FireTabletPlus zeigt sehr schön unser zentralstes Objekt, das Panel. Das Panel repräsentiert eine Site mit all ihren Informationen und Objekten. Der wichtigste Teil befindet sich unter dem Panel: Die Baumstruktur der verschiedenen Devices einer Site sowie die erfassten Fehler und Kommentare die wiederum an den einzelnen Devices angehängt sind. Kommentare und Fehler unterscheiden sich in ihrer Art nicht voneinander, weshalb sie mit einem einzigen Objekt abgebildet werden können. Das Panel enthält ausserdem eine Liste mit ToDo, in der zum Beispiel Aufgaben für den Techniker erfasst werden könnten. Es existiert für jedes Panel ein eigener FaultIDGenerator, da die Fault ID nur im Umfeld eines Panels unique sein soll. Dies soll dazu dienen die IDs klein zu halten damit der Benutzer sich eine ID auch merken kann. Die Fault ID die hier generiert wird hat keinen direkten Bezug zur Datenbank. Auf der anderen Seite werden die administrativen Informationen zum Panel in separaten Objekten CustomerAffairs

(für Informationen zur Site) und Customer (für Informationen zu Rechnungsadressen / Personen).

4.3.2 Package Übersicht

In diesem Abschnitt wird zuerst die allgemeine Package Struktur angeschaut und anschliessend auf die Analyse der zirkulären Abhängigkeiten eingegangen.

Package Struktur

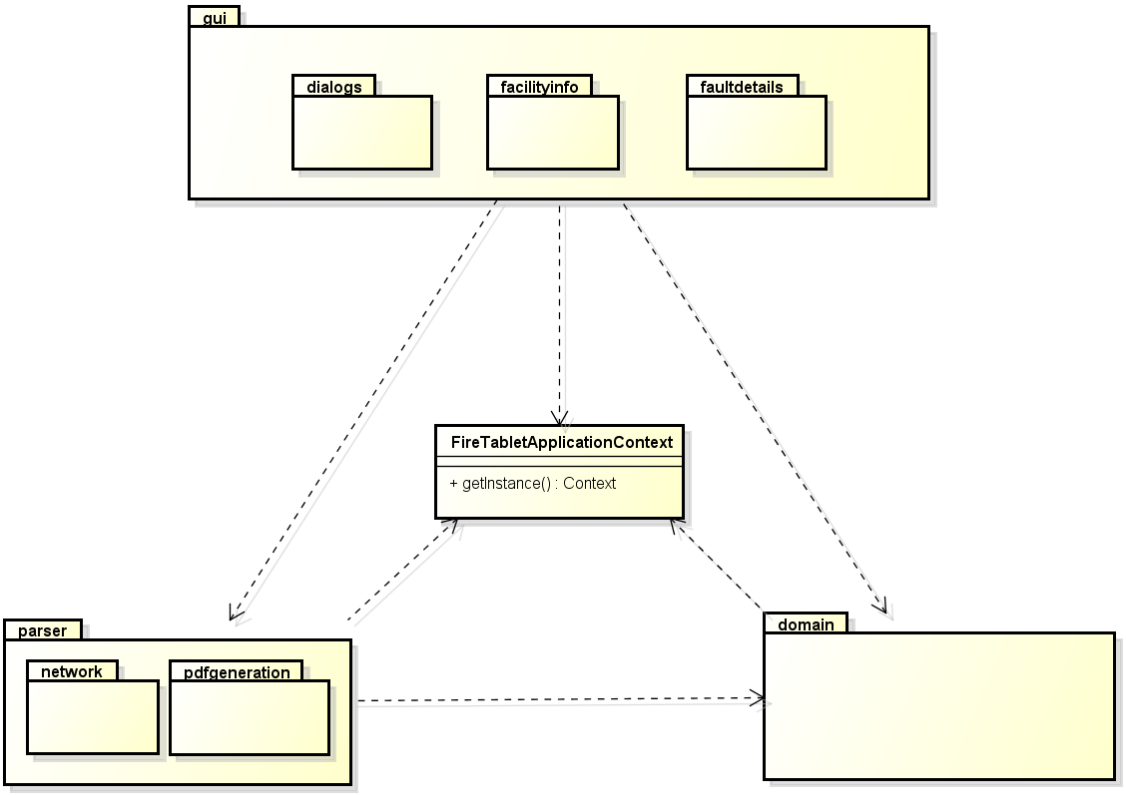


Abbildung 4.17: Struktur der Packages

Die Packagestruktur in unserer Applikation stellt das gui an oberste Stelle. Dies ist teilweise durch Android gegeben, da Activities in Android sowohl Controller als auch View-Elemente sind und wir diese im gui package unterbringen. Dadurch greift in unserem Fall das gui sowohl auf die Business-Objekte im domain, als auch auf Funktionalitäten aus dem parser package zu. Die Abhängigkeiten aus dem parser package zum domain ergeben sich durch Zugriffe auf Business-Objekte zum Beispiel bei der PDF-Generierung oder dem Up- und Download der Sites. Der FireTabletApplicationContext ist unser Singleton der vom Android Framework instanziert wird um eine Schnittstelle zum Android System zu erhalten oder Objekte über einen Activity-Verlauf zu halten ohne sie dabei serialisieren zu müssen. Wie man auf der Abbildung oben sieht wird der Singleton von allen Packages benutzt.

Zirkuläre Abhängigkeiten

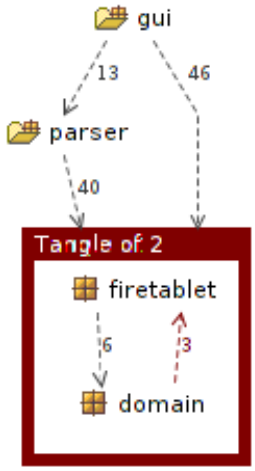
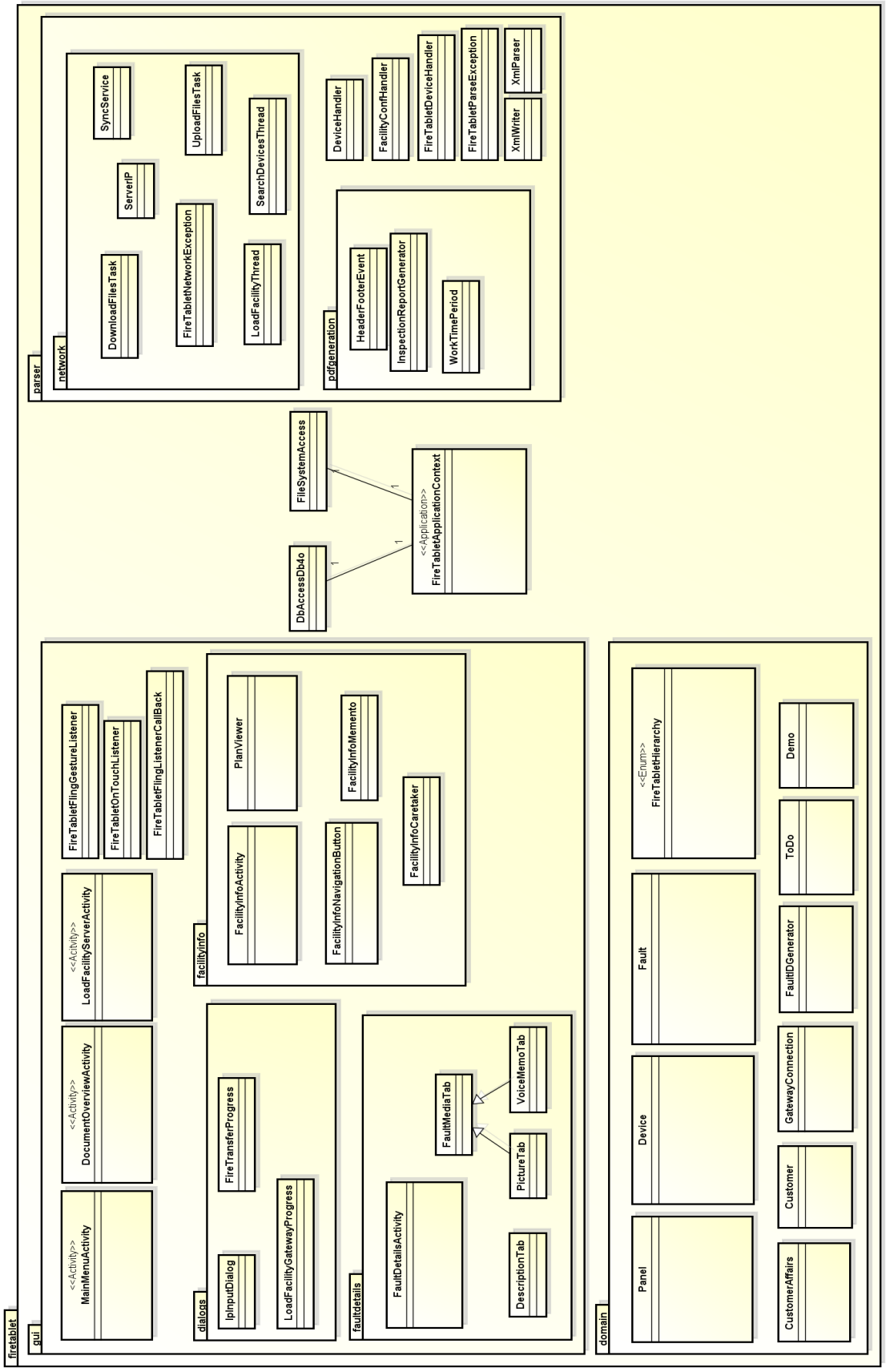


Abbildung 4.18: Packagestruktur

Bei der Analyse unseres Sourcecodes mit Structure101 haben wir noch diverse Zirkuläre Abhängigkeiten gefunden. Viele haben wir durch Techniken wie Broadcast oder Umstrukturierungen im Code lösen können, jedoch blieb bis zum Schluss eine Abhängigkeit vom firetablet package zum darunterliegenden domain package. Diese Abhängigkeit beruht auf unserem Singleton FireTabletApplicationContext, der im domain package einmal für den Zugriff auf die Android Ressourcen benutzt wird und das zweite mal für den Zugriff auf die Datenbank. Die erste Abhängigkeit ist nicht zu beseitigen, da die entsprechende Klasse ein Java Enumeration ist die nicht explizit instanziert wird. Die einzige Möglichkeit wäre gewesen die Farbkonfiguration hartcodiert im Code anzubringen, was absolut gegen die ganze Android Ressourcenverwaltung spricht. Die zweite Abhängigkeit entsteht durch einen Datenbankzugriff im FaultIDGenerator, der sich selbst verwalten soll und nicht der Entwickler der diese Klasse benutzt sich um die Persistierung kümmern muss. Da für einen Datenbankzugriff zuerst der Context angefordert werden muss und anschliessend die Datenbankklasse DbAccessDb4o, entstehen hier 2 Abhängigkeiten in das firetablet package. Demnach sind es insgesamt 3 Abhängigkeiten, die allerdings alle nur Zugriffe auf unseren Singleton darstellen. Nach reiflicher Überlegung und diversen Diskussionen mit Thomas Corbat und Prof. Sommerlad haben wir uns dann dazu Entschlossen diese Warnungen von Structure101 zu akzeptieren, aber diese Abhängigkeiten nicht weiter aufzulösen, da diese Abhängigkeit zum FireTabletApplicationContext aus unserer Sicht zu keinen Nebenwirkungen haben kann.

4.3.3 Komplette Klassenübersicht


firetablet
gui
<<Activity>>
MainMenuActivity
<<Activity>>
DocumentOverviewActivity
<<Activity>>
LoadFacilityServerActivity
FireTabletFlingGestureListener
FireTabletOnTouchListener
FireTabletFlingListenerCallBack
dialogs
IpInputDialog
FireTransferProgress
LoadFacilityGatewayProgress
faultdetails
FaultDetailsActivity
FaultMediaTab
VoiceMemoTab
PictureTab
DescriptionTab
facilityInfo
FacilityInfoActivity
PlanViewer
FacilityInfoNavigationButton
FacilityInfoMemento
FacilityInfoCaretaker
parser
network
SyncService
ServerIP
DownloadFilesTask
FireTabletNetworkException
UploadFilesTask
SearchDevicesThread
LoadFacilityThread
pdfgeneration
HeaderFooterEvent
InspectionReportGenerator
WorkTimePeriod
DeviceHandler
FacilityConfHandler
FireTabletDeviceHandler
FireTabletParseException
XmlParser
XmlWriter
FileSystemAccess
DbAccessDb4o
<<Application>>
FireTabletApplicationContext
1
1
1
1
domain
<<Enum>>
FireTabletHierarchy
Fault
Device
Panel
Customer
CustomerAffairs
GatewayConnection
FaultIDGenerator
ToDo
Demo

4.3.4 Package firetablet

In diesem Bereich wird das Package *firetablet* dokumentiert. In diesem Package befindet sich vor allem der *ApplicationContext* mit seinen beiden Hilfsklassen. Der Context wird über das gesamte Programm hinweg verwendet und stellt ein Singleton dar.

Abbildung 4.19: Package firetablet

FireTabletApplicationContext

Die FireTabletApplicationContext Klasse erbt von der Android System Klasse *Application* und stellt somit unseren ApplicationContext innerhalb unserer Applikation dar. Diese Klasse wird vom Android System als Singleton instanziert und ist somit dem System bekannt. Über diese Klasse lassen sich nun verschiedene Framework Funktionen aufrufen, wie zum Beispiel der Zugriff auf das Dateisystem des mobilen Geräts. Sie enthält 2 weitere Hilfsklassen genannt *FileSystemAccess* und *DbAccesDb4o* um eine Schnittstelle auf das Dateisystem respektive unsere Datenbank anzubieten. Der Kontext hält ausserdem die aktuell ausgewählte Site auf der gearbeitet wird (*Panel*).

DbAccessDb4o

Die DbAccessDb4o Klasse ist eine Hilfsklasse die den Zugriff auf die Datenbank gewährleistet. Die hält sich den *ObjectContainer* über den die Kommunikation mit der Datenbank erfolgt. Die Methoden in dieser Klasse wurden so generisch wie möglich implementiert und bieten store, get, getAll und delete für alle Objekttypen an. Einzig die get- / setServerIP Methoden wurden speziell behandelt, da diese gewährleisten müssen dass jeweils nur 1 Objekt vom Type *ServerIP* in der Datenbank abgelegt/zurückgegeben wird.

FileSystemAccess

Die FileSystemAccess Klasse stellt die Schnittstelle zum Dateisystem des mobilen Geräts dar. Sie enthält diverse Methoden für Pfadgenerierungen. Sie kümmert sich darum dass Dateien stets in der korrekten Datenstruktur und im richtigen Site Ordner abgelegt werden.

4.3.5 Package gui

Hier halten sich alle Klassen auf die mit der grafischen Darstellung zu tun haben. Im Normalfall sind die Klassen alle vom Typ *Activity* aber es befinden sich auch die Hilfsklassen der *Activities* in diesem Package.

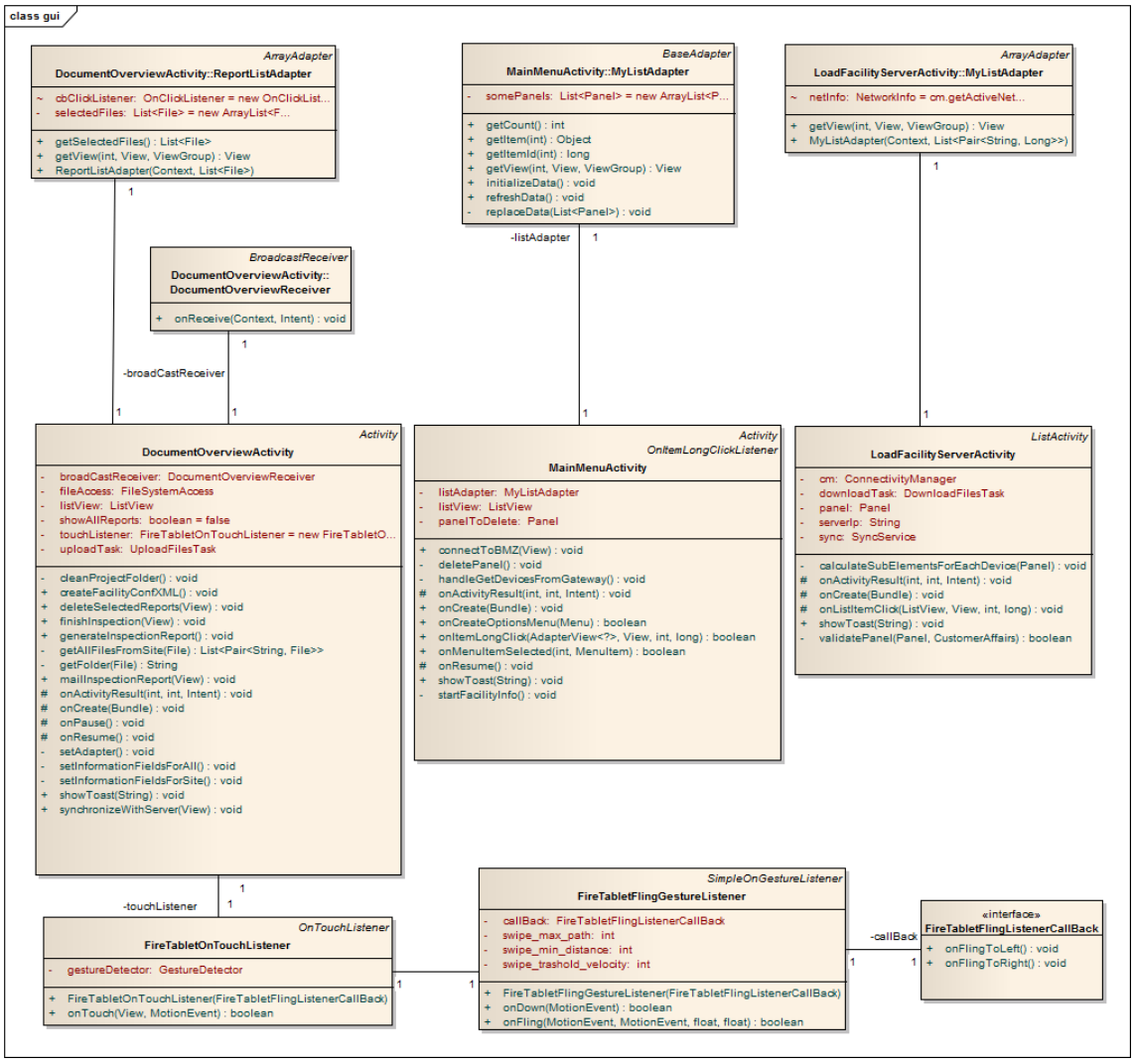


Abbildung 4.20: Package gui

MainMenuActivity

Die MainMenuActivity ist abgeleitet von einer Android Activity, was bedeutet dass diese Klasse die Darstellungs- und Kontrolleraufgaben eines Screens übernimmt. In diesem Fall ist dies der MainMenu (Seite 45). Es wurde eine inner class *MyListAdapter* implementiert der sich um die Anzeige der Liste von Sites kümmert.

DocumentOverviewActivity

Die DocumentOverviewActivity ist abgeleitet von einer Android Activity. Sie übernimmt die Darstellungs- und Kontrollaufgaben des DocumentOverview Screen (Seite 56). Es wurde auch hier eine inner class *ReportListAdapter* implementiert der die Liste von Reports verwaltet.

LoadFacilityServerActivity

Die LoadFacilityServerActivity ist abgeleitet von einer Android ListActivity, die einige Hilfestellungen für Activities mit Listen bietet. Sie übernimmt die Darstellungs- und Kontrollaufgaben des MainMenu Screen - Site laden (Seite 46). Es wurde auch hier eine inner class MyListAdapter erstellt der sich um die Verwaltung der Liste von Sites kümmert.

FireTabletOnTouchListener

Die FireTabletOnTouchListener ist eine Implementation des OnTouchListener Interfaces. Dafür wird bei der Instanzierung ein CallBackObjekt genannt *FiteTabletFlingListenerCallBack* übergeben. Es wird dann ein neuer *GestureDetector* mit einer neuen Instanz des *FireTabletFlingGestureListener* instanziert. Es wurde die onTouch Methode überschrieben, die nun die Behandlung des *TouchEvents* an den GestureDetector (und somit unseren *FireTabletFlingGestureListener*) delegiert.

FireTabletFlingGestureListener

Die FireTabletFlingGestureListener Klasse wird von der Android Klasse *SimpleOnGestureListener* abgeleitet. Es wurden die Methoden onDown sowie onFling überschrieben. OnDown gibt statisch false zurück, um anderen Komponenten ihre InputEvents (zum Beispiel ClickEvents auf der Tabelle) nicht zu stehlen. Die OnFling Methode überprüft anhand von der zurückgelegten Distanz und Geschwindigkeit des Fingers auf dem Bildschirm ob es sich dabei um einen Fling gehandelt hat und auf welche Seite dieser gemacht wurde. Sobald ein Fling erkannt wurde wird auf dem CallBack Objekt die entsprechende Methode aufgerufen die von der entsprechenden Activity dann implementiert wurde und die gewünschte Funktionalität auf dem Screen auslöst.

DocumentOverviewReceiver

Die inner class DocumentOverviewReceiver wird von der Android Klasse BroadcastReceiver abgeleitet. Es wurde die Methode onReceive überschrieben um Broadcast Meldung zu empfangen und darauf reagieren zu können. Siehe Broadcast auf Seite 11 für weitere Informationen betreffend dem Broadcast System.
Das Package der Dialoge trennt spezielle *Activities* von den restlichen GUI Klassen. Dialoge füllen nicht ganze Displays aus sondern sondern tauchen nur kurz als neue Fenster auf.

4.3.6 Package gui.dialog

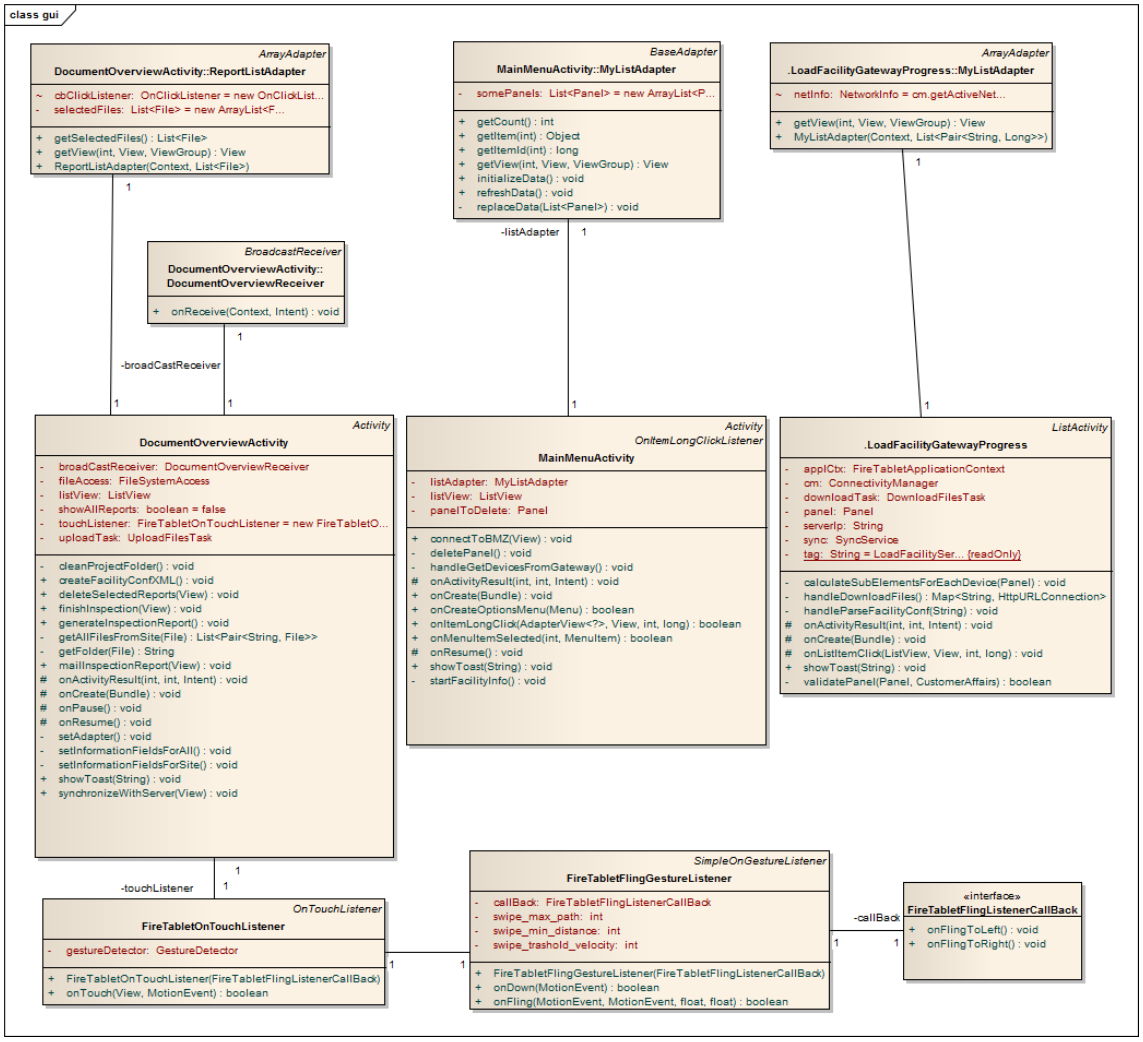


Abbildung 4.21: Package gui.dialog

LoadFacilityGatewayProgress

Die Klasse LoadFacilityTree wurde von der Android Klasse *Activity* abgeleitet. Sie ist dafür zuständig den Dialog beim herunterladen anzuzeigen und mit aktuellen Informationen zu versorgen. Es wird daher auch hier ein Broadcast verwendet um Aktualisierungen des Downloads zu erhalten.

IpInputDialog

Die Klasse IpInputDialog ist für die Anzeige eines Dialogs verantwortlich in dem die IP für den zentralen Site-Ablageserver gespeichert wird. Beim Klick auf ÖK"wird die IP in die Datenbank geschrieben.

FileTransferProgress

Die Klasse FileTransferProgress zeigt in einem Dialog den Fortschritt beim Hoch- sowie Herunterladen von Dateien vom Server an. Ausserdem prüft er mit Hilfe eines **??** Regex die Gültigkeit der eingegebenen IP.

4.3.7 Package gui.facilityinfo

Der Screen SiteInfo ist sehr komplex und benötigt viele Klassen. Aus diesem Grund werden alle beteiligten Klassen in diesem Package abgetrennt.

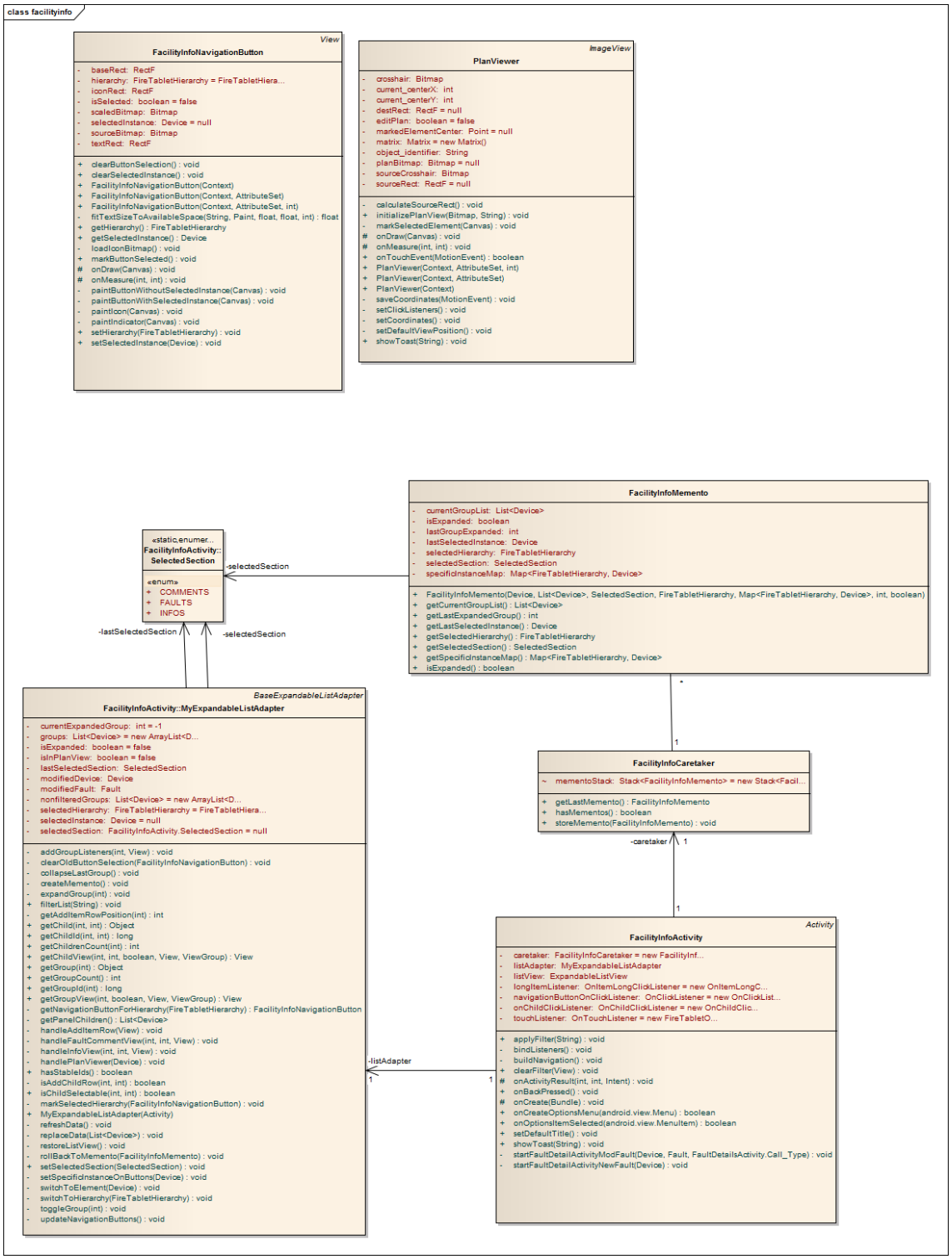


Abbildung 4.22: Package gui.facilityinfo

FacilityInfoNavigationButton

Die Klasse FacilityInfoNavigationButton leitet von der Android Klasse *View* ab und repräsentiert somit eine Eigenimplementation eines UI-Elements. Die speziellen Navigation Buttons werden im SiteInfo Screen auf der linken Seite angezeigt. Sie enthalten jeweils immer eine *FireTabletHierarchy* und ein der Hierarchie entsprechendes Icon. Ausserdem eine Instanz von *Device*, vorausgesetzt es wurde auf dieser Hierarchiestufe ein spezifisches Element ausgewählt. Es wurde die Methode onDraw überschrieben, wodurch sich die Komponente nun selber anhand den oben erwähnten Attributen auf dem User Interface zeichnet. Der FacilityInfoNavigationButton wird durch das Android System gemäss den Vorgaben im entsprechenden Layout XML instanziert und angezeigt.

PlanViewer

Die Klasse PlanViewer leitet von der Android Klasse *ImageView* ab, einer spezifischen *View* Klasse für die Anzeige von Bildern. Der PlanViewer ist ebenfalls eine Eigenimplementation eines UI-Elements das vom Android System gemäss dem Layout XML instanziert und angezeigt wird. Speziell zu erwähnen ist, dass der PlanViewer von Anfang an im FacilityInfo Screen instanziert und vorhanden ist, jedoch die Visibility der View auf GONE gesetzt wird. Dies ermöglicht die Anzeige des PlanViewers durch einfaches setzen der Visibility auf VISIBLE. Dadurch existiert stets nur ein Objekt des PlanViewers das einfach wiederverwendet werden kann. Es wird dabei jeweils eine initialize Methode aufgerufen in der das Bild des Plans übergeben wird sowie den Object Identifier des aktuell ausgewählten Devices. Anhand dieser Daten sucht der PlanViewer in einem Property File nach der Position des Devices auf dem Plan. Wenn er die Position findet markiert und zentriert der PlanViewer das Device auf der Planansicht. Für das scrollen des Plans innerhalb des PlanViewers wurde ein eigener TouchListener implementiert der die Bewegungen des Fingers erfasst und den Plan entsprechend bewegt. Die Bewegungen sowie die verschiedenen Zoomstufen werden über Matrixtransformationen auf das Bild angewendet. Die Zoomstufe wird über die zwei Buttons geregelt und auch darauf begrenzt. Zoomen über die Multitouchfunktionalität

4.3.8 Package gui.faultdetails

Die Auftrennung in mehrere Klassen beim Fehlerdetails Screen entsteht durch die Taps welche in dieser Activity verwendet werden. Jeder Tab ist eine eigene *Activity*.

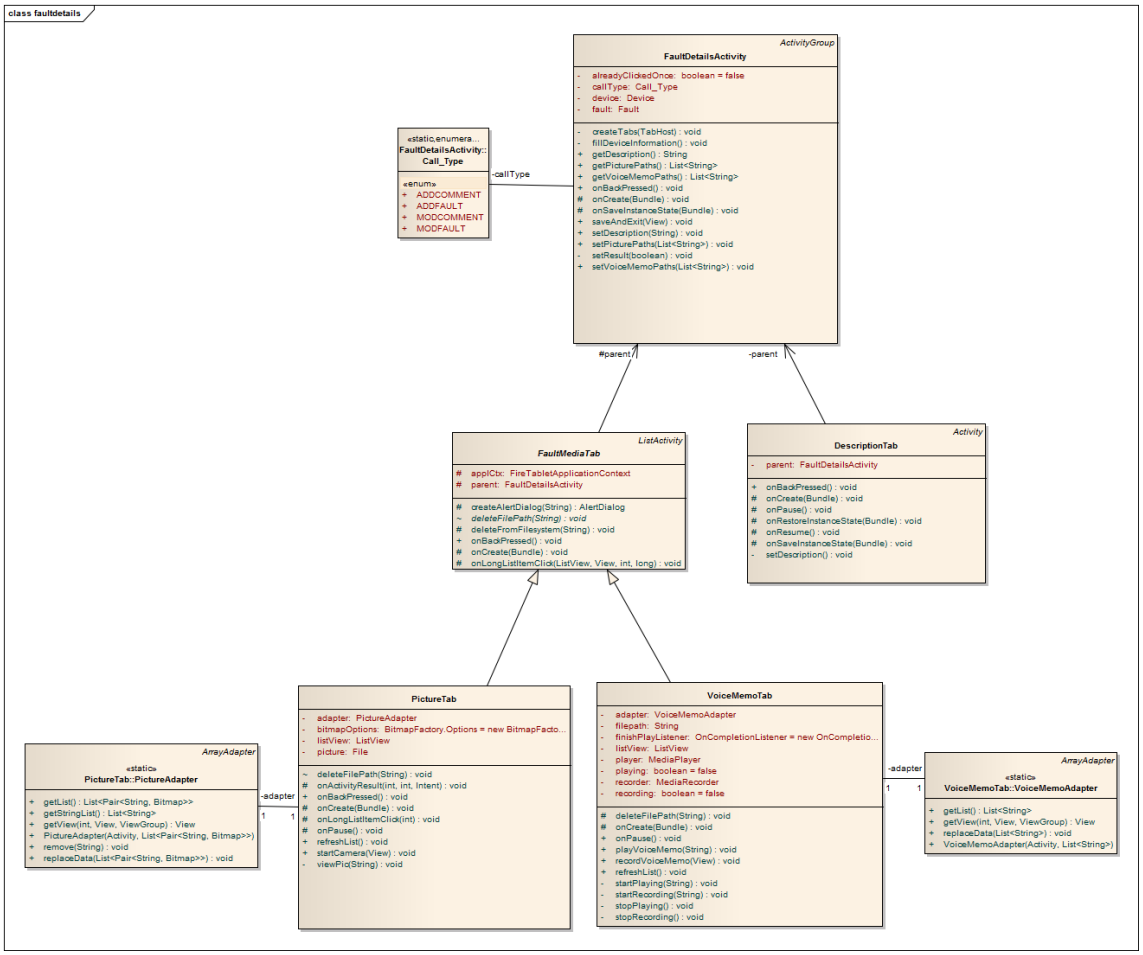


Abbildung 4.23: Package gui.faultdetails

FaultDetailsActivity

Die FaultDetailsActivity ist abgeleitet von der Android Klasse *ActivityGroup* und ist somit verantwortlich für die Darstellung und User Interaktionen des Faultdetail Screen. Sie leitet von der Klasse *ActivityGroup* ab, da sie mehrere UnterActivities enthält, die sich um die verschiedenen Reiteransichten kümmern. In dieser Klasse wird der Screen aufgebaut, der *TabHost* und die verschiedenen Reiteransichten initialisiert. Jede Reiteransicht hat wiederum ihre eigene *Activity* die sich um die Ansicht kümmern. Der FaultDetailsActivity wird im aufrufenden Intent ausserdem das Device sowie der zu ändernde Fehler übergeben, den sie sich dann speichert und am Ende wieder an die ursprünglich aufrufende Activity mit einem Statuscode ÖKöder "Cancelledßurückgibt.

PictureTab / VoiceMemoTab

Die beiden Klassen sind jeweils *ListActivities* für die beiden Reiteransichten für Bild und Tonaufnahmen. Sie leiten beide von der gemeinsamen Oberklasse *FaultMediaTab* ab, in der einige gemeinsam genutzte Methoden hinterlegt sind. Sie enthalten beide jeweils eine Liste für ihre Aufnahmen und verwalten diese selber über einen Adapter der auch hier als inner class Implementiert wurde. Über das parent objekt, das im *FaultMediaTab* bereitge-

stellt wird kann auf die übergeordnete *Activity* zugegriffen werden um ihre Daten in einer zentralen *Activity* zu speichern.

DescriptionTab

Die *DescriptionTab* Klasse erbt ebenfalls von einer *Activity* und ist für die Reiteransicht des Beschreibungstext verantwortlich. Sie greift ebenfalls über das Eltern-Objekt auf die übergeordnete *FaultDetailsActivity* Klasse zu, um ihre Daten zu speichern.

4.3.9 Package parser

Hier werden die von FireTabletPlus benötigten Textdateien generiert oder gelesen. Weiter beinhaltet dieses Package die dazu benötigten Hilfsmittel wie Netzwerkkommunikation oder PDF-Generierung.

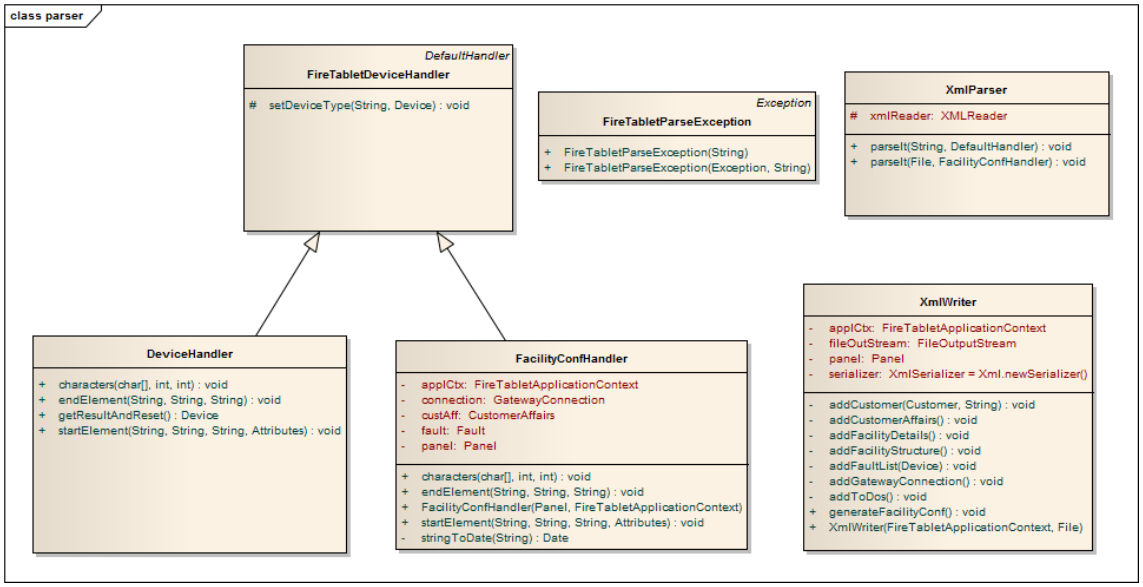


Abbildung 4.24: Package parser

DeviceHandler

Der *DeviceHandler* ist dafür verantwortlich ein Device vom XML in ein Device Objekt zu parsen.

FacilityConfHandler

Der *FacilityConfHandler* parst das facilityConf.xml und füllt schlussendlich die XML Daten in eine Panel Objekt ab.

XMLWriter

Der XMLWriter schreibt das facilityConf.xml aus der internen Objektstruktur.

4.3.10 Package parser.pdfgeneration

In diesem Packet ist nur die InspectionReportGenerator Klasse wichtig. die beiden Klassen sind Hilfsklassen. *HeaderFooterEvent* generiert die Kopf und Fusszeile des PDFs und *WorkTimePeriod* ist dazu da, die Zeitberechnung zu simulieren.

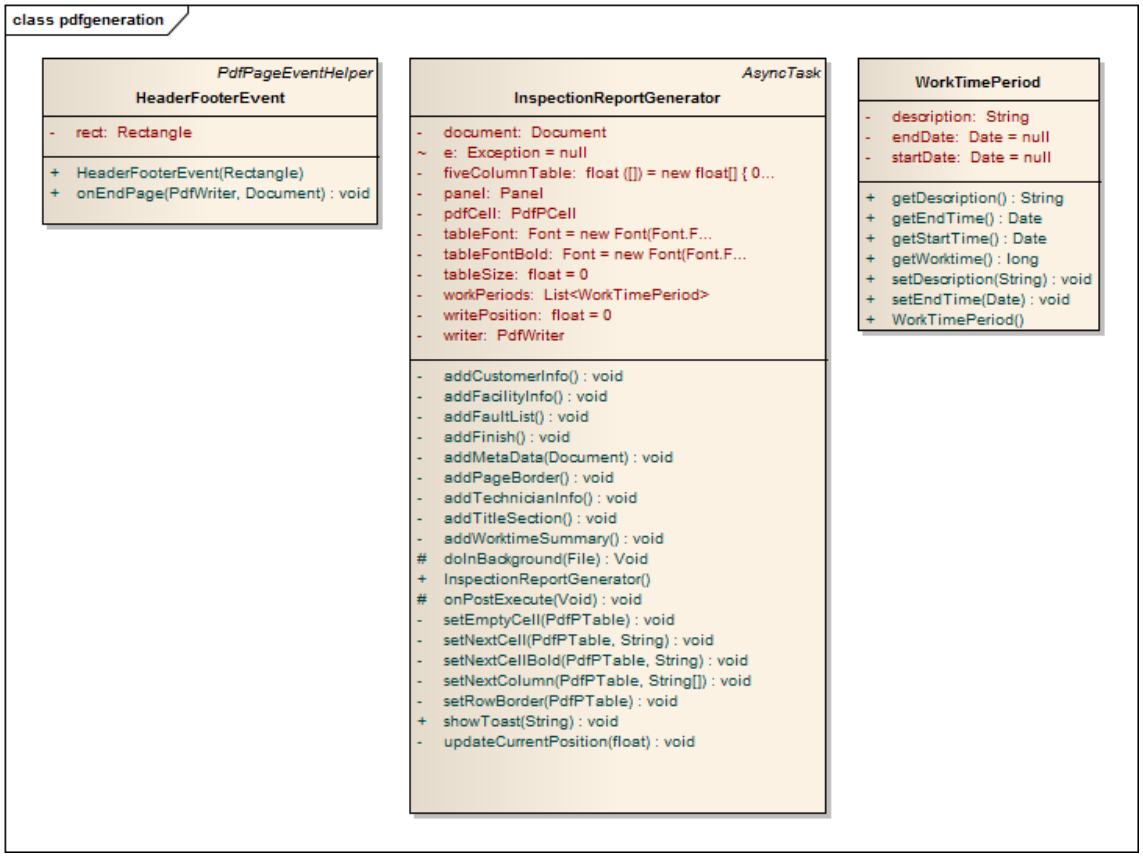


Abbildung 4.25: Package parser.pdfgeneration

InspectionReportGenerator

Die Klasse *InspectionReportGenerator* erbt von der Android Klasse *AsyncTask*. Da das generieren eines grossen PDFs unter Umständen einen Moment dauern kann, haben wir uns Entschieden die Generierung asynchron abzuarbeiten, um dem Benutzer keine langen Antwortzeiten vom UI zumuten müssen. Die Methoden dieser Klasse sind ziemlich selbsterklärend und generieren jeweils verschiedene Absätze des PDF Dokuments.
Das Package *network* beinhaltet sämtliche Klassen die mit dem Netzwerk in Verbindung stehen. Die meisten Klassen sind *AsyncTasks* welche die Netzwerkkommunikation in separate Threads auslagern. *SyncService* ist eine allgemeine Hilfsklasse für Netzwerkabfragen.

4.3.11 Package parser.network

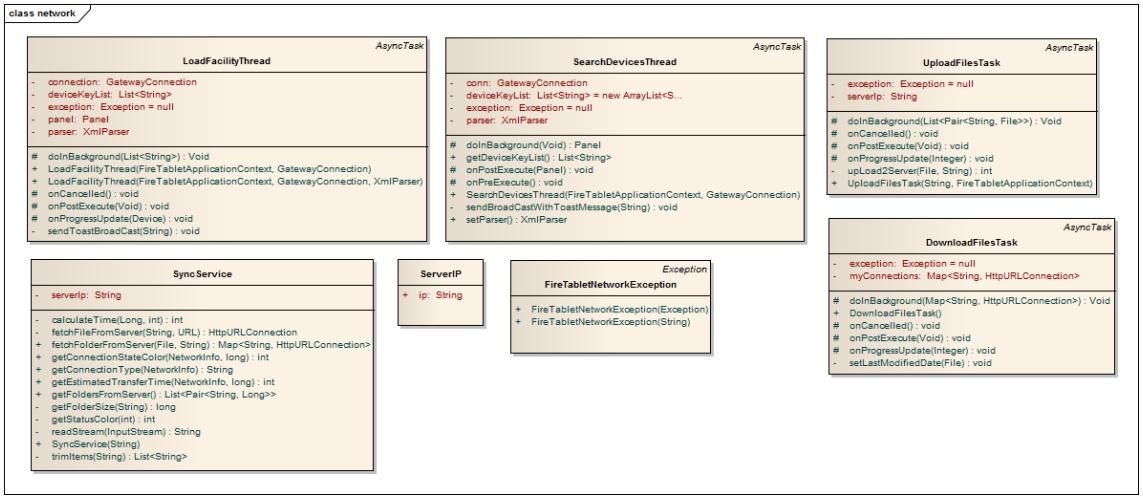


Abbildung 4.26: Package parser.network

DownloadFilesTask

Die DownloadFilesTask Klasse ist verantwortlich für das Herunterladen aller Dateien die sich in einem Site-Ordner auf dem Server befinden. Sie lagert den Download der Dateien in einen separaten Thread aus und schickt Broadcasts mit den Informationen über den Fortschritt des Downloads an den FileTransferProgress. Dem *AsyncTask* kann eine Map mit String-Keys und *HttpURLConnections*-Values übergeben werden. Die Connections müssen bereits geöffnet sein damit berechnet werden kann wie viel Daten zu erwarten sind. Mit den Keys weiss die Klasse unter welchem Pfad die empfangenen Dateien zu speichern sind.

UploadFilesTask

Der UploadFilesTask ist das Gegenstück zu DownloadFilesTask. Der Unterschied besteht darin, dass man diesem *AsyncTask* eine *List* mit *Pairs* bestehend aus *String* und *File* übergibt. Der String gibt den Pfad des Speicherorts auf dem Server an während das *File* die zu uploadende Datei ist. Auch diese Klasse verschickt Broadcasts um FileTransferProgress auf dem aktuellen Stand zu halten.

SearchDevicesThread und LoadFacilityThread

Diese beiden *AsyncTasks* übernehmen das Laden der Anlagestruktur von der Brandmeldezentrale. Mit dem SearchDevicesThread wird eine Anfrage an den Gateway geschickt die alle vorhandenen Geräte liefert. Diese Information verwendet der LoadFacilityThread um jedes Device vom Gateway zu laden. Ganze Ablauf ist im folgenden Sequenzdiagramm verdeutlicht.

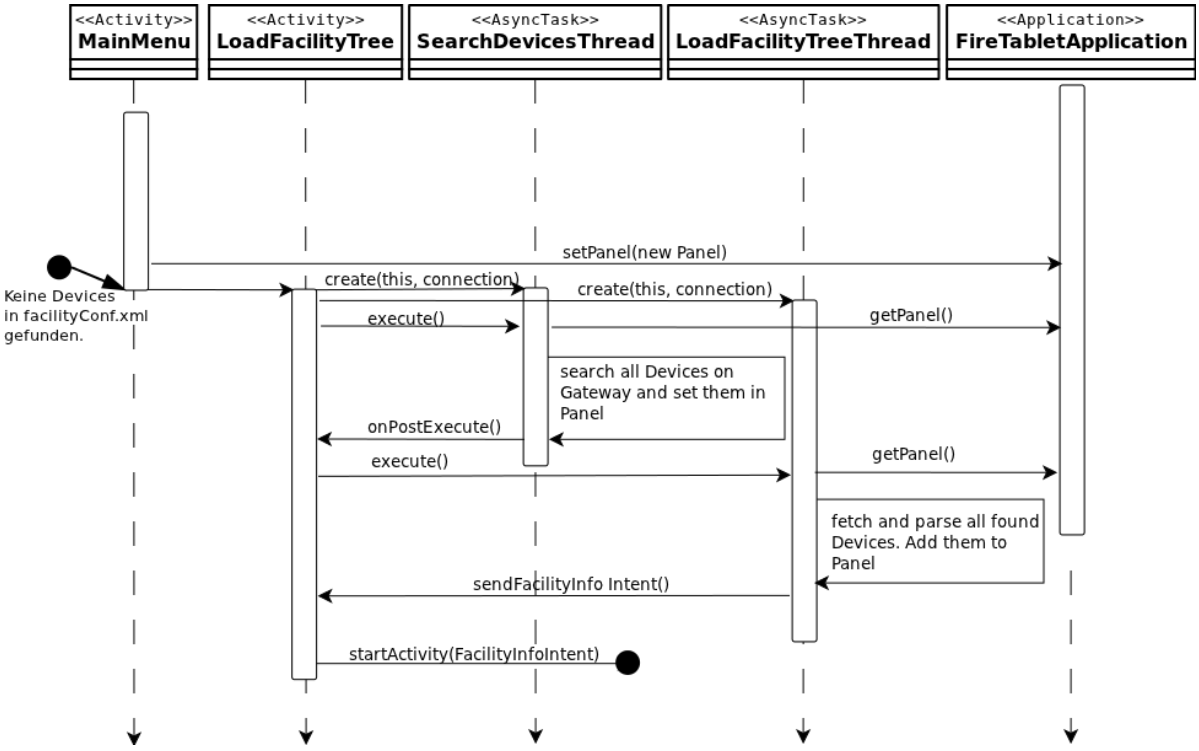


Abbildung 4.27: Sequenzdiagramm um alle Devices einer Brandmeldezentrale zu laden.

SyncService

In dieser Klasse werden verschiedene Aufgaben zum Thema Netzwerk untergebracht. Die Hilfsklasse *SyncSercie* wird von verschiedenen *Activities* verwendet um mit dem Netzwerk zu kommunizieren. Die Klasse ist statuslos, wird also von jeder Activity bei Gebrauch neu initialisiert.

4.3.12 Package domain

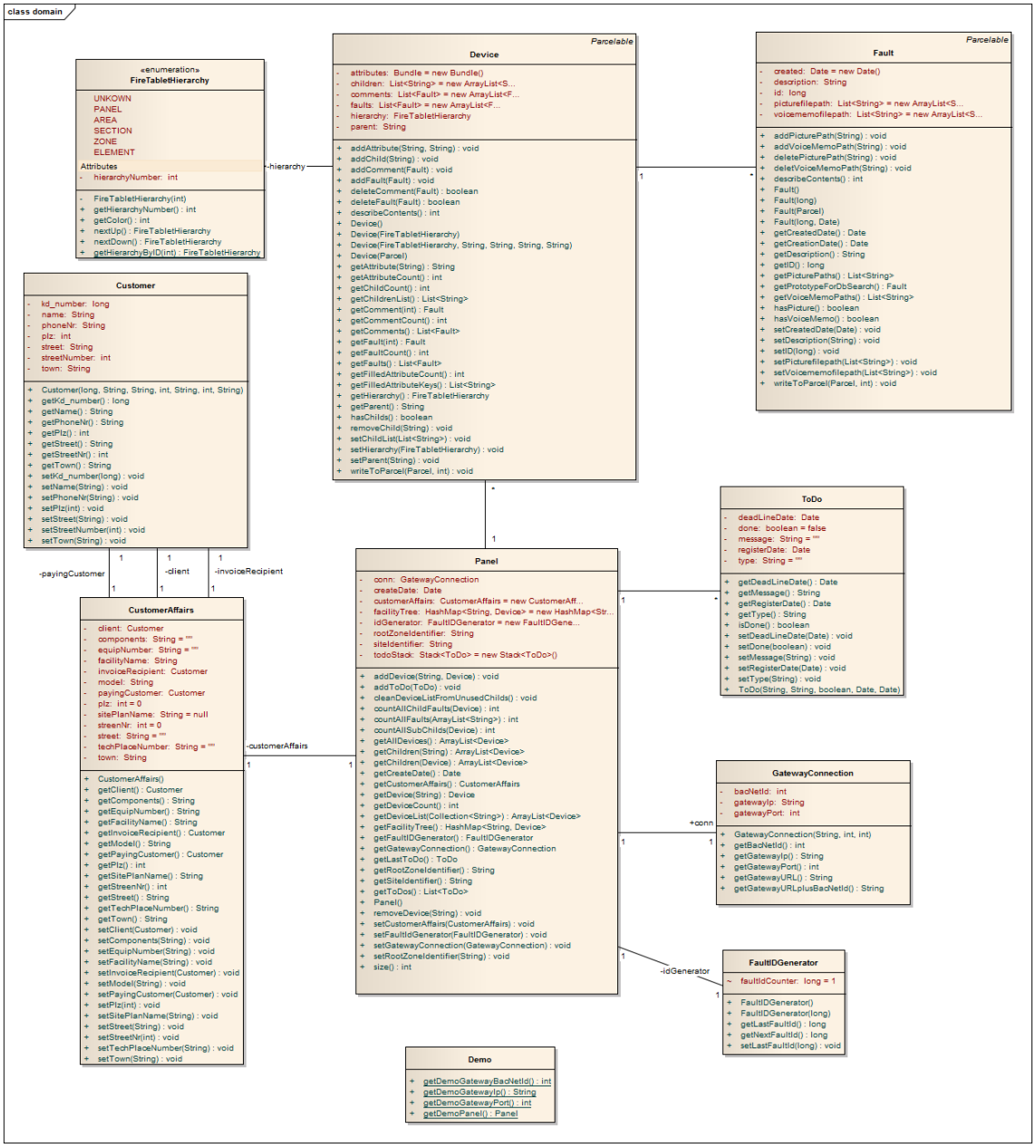


Abbildung 4.28: Package domain

FireTabletHierarchy

Die Klasse *FireTabletHierarchy* ist eine *Java Enumeration*. Sie enthält alle möglichen Hierarchiestufen in unserer Applikation. Ausserdem definiert sie für jede Hierarchiestufe eine Farbe, damit überall in der Applikation eine Hierarchie stets in der selben Farbe angezeigt wird. Hinter den Hierarchien stehen ausserdem Zahlen, um eine Reihenfolge in den Hierarchien zu definieren und von einer Hierarchie auf die nächst tiefere / höhere zu wechseln.

CustomerAffairs

Die Klasse *CustomerAffairs* ist eine einfache Datenhaltungsklasse die administrative Informationen zur Site und alle Personen die mit dieser Site involviert sind speichert.

Customer

Die *Customer* Klasse enthält Daten zu einzelnen Personen oder Unternehmen, die mit der Site involviert sind.

FaultIDGenerator

Der *FaultIDGenerator* generiert neue IDs für Faults. Er ist mit dem Panel verknüpft, da die ID der Faults für jedes neue Panel wieder bei 1 anfangen soll. Der Sinn hinter dieser ID ist es dem Benutzer eine einfache Nummer zu liefern mit der er einen Fehler im Rahmen einer Site erkennen kann. Der FaultIDGenerator bietet eine Methode getNextFaultId mit der einfach eine neue ID angefordert werden kann, der FaultIDGenerator kümmert sich dann selbst darum sich zu selbst persistieren um sicher zu gehen dass keine ID mehrfach vergeben wird.

GatewayConnection

In der Klasse *GatewayConnection* wird die Verbindung zum Gateway mit allen nötigen Informationen bereitgestellt.

Demo

Die *Demo* Klasse enthält mehrere statische Methoden um einfach und schnell eine Site für Demonstrationszwecke zu erzeugen oder die Verbindung zum Gateway herzustellen.

4.4 Knacknüsse

Während der Entwicklungsphase tritt man an verschiedene Probleme aller Art heran. Die Aufwändigsten sind in diesem Bereich dokumentiert.

4.4.1 Datenbank

Für unsere lokale Datenhaltung wurde DB4O Datenbank verwendet, mit der wir uns in einige Probleme gestürzt haben. Das erste Problem auf das wir mit Db4o gestossen sind und das einige merkwürdige Nebeneffekte mit sich gebracht hat war das Problem mit der ActivationDepth. Da Db4o einen Standardwert von 5 für die ActivationDepth annimmt, wurden beim Laden der Site nicht alle Objekte Aktiviert und zurückgegeben. Das kuriose dabei ist, dass alle Objekte bis auf die letzte Stufe (Faults) aktiviert wurden. Dies hatte zur Folge das scheinbar keine Faults persistiert wurden, obwohl sie in Wirklichkeit richtig persistiert, aber nicht korrekt aus der Datenbank ausgelesen wurden. Durch das Konfigurieren auf der Datenbank dass die Klasse Panel eine ActivationDepth von 10 haben soll wurde dieses Problem relativ einfach gelöst. Das zweite Problem hatte mit der Serialisierung von Objekten zwischen zwei Activities bei Android zu tun. Durch die Serialisierung

(oder Parcelable wie es in Android heisst), geht natürlich die Objektreferenz im ObjectContainer verloren. Dieses Problem war uns von Anfang an bewusst und wurde auch entsprechend behandelt. Wir haben uns Entschlossen das Objekt in der neuen Activity erneut aus der Datenbank zu lesen um die Objektreferenz dem ObjectContainer bekannt zu machen und darauf dann Datenbankoperationen durchführen zu können. Was wir aber zu dieser Zeit nicht bedacht hatten war die Möglichkeit dass die gleiche Site mehrfach auf das mobile Gerät heruntergeladen werden kann. Dadurch ergeben sich gewisse Probleme, da ein Fehler nicht mehr eindeutig in der Datenbank vorhanden ist. Selbst wenn wir eine ID Einführen wird die durch den Export / Import der Site ebenfalls potentiell kopiert und es entstehen abermals duplikate in der Datenbank. Diese Duplikate haben zur Folge dass der Fault zwar verändert und auch wieder in die Datenbank geschrieben wird, allerdings ist es reiner Zufall welchen Fault man gerade ändert wenn zwei gleiche Sites auf dem Gerät sind. So modifiziert man zum Teil einen Fault auf einer anderen Site und sieht demnach die Änderungen in der aktuellen Site nicht. Wir haben für dieses Problem drei mögliche Lösungen ausgearbeitet:

Verhindern von mehreren gleichen Sites auf dem Gerät Wir könnten das herunterladen von gleichen Sites verhindern, wodurch keine identischen Faults mehr in der Datenbank auftauchen sollten. Dadurch entsteht aber eine zusätzliche Limitation die wir eigentlich nicht eingehen wollten.

Globale Unique IDs für Fault Generieren Wir könnten Unique IDs einführen um die Faults eindeutig identifizieren zu können. Dafür müsste aber gewährleistet werden dass die ID auch über alle Geräte hinweg eindeutig bleibt. Das bedeutet es müsste eine neue Serverkomponente zum Einsatz kommen die gewisse ID Ranges an die einzelnen Geräte verteilt. Dies bedeutet einen riesigen Overhead für uns sowie das Problem dass die Geräte nicht zu jeder Zeit über eine Internetanbindung verfügen. Abgesehen davon wird die ID sehr kryptisch und unleserlich für den Benutzer, was wir unbedingt vermeiden wollten.

Keine Datenbankzugriffe in der FaultDetailsActivity Dies ist die Lösung für die wir uns Entschieden haben. Die Lösung sieht so aus dass wir den zu ändernden Fault an die FaultDetailsActivity serialisiert mitschicken, diesen dort verändern und wieder zurück an die FacilityInfoActivity zurückschicken. Da wir in der FacilityInfoActivity stets alle Objektreferenzen im ObjectContainer referenziert haben, können wir nun sehr einfach das Objekt updaten. Dies ist ausserdem der von Android empfohlene Ansatz, dass Daten für die Manipulation eines Objektes per Intent mitgeschickt werden und nach der Manupulation wieder mit dem Intent zurückkommen.

4.4.2 Sardine Library

Die Datenablage auf dem Server sollte anfangs mit WebDav gelöst werden. WebDav ist eine unkompliziertes Protokoll um Daten über HTTP zu versenden. Das Protokoll erfüllt sehr ähnliche Funktion wie FTP aber benutzt den Port 80 als Kommunikationskanal. Weitere Informationen zu Sardine können hier entnommen werden: [Sardine Library].

Durch die einfache Handhabung dieser Library war sie wie geschaffen für FireTablet, denn für es war wichtig möglichst wenig mit der Netzwerkschicht in Kontakt zu kommen, da dies nicht Teil der Arbeit ist. Zudem wäre es sehr praktisch gewesen eine Library verwenden zu können die sich um alles kümmert und stabil läuft.

In einem Versuchsprojekt auf Java funktionierte Sardine ohne Probleme. Beim Versuch dies unter Android zu versuchen kam jedoch die Entäuschung: Sardine benötigt Java Core Libraries wie zum Beispiel die JAXB Library um XML zu parsen. Diese Libraries sind nicht

im Android Framework enthalten und werden beim Versuch sie trotzdem auszuführen mit einer Compilermeldung quittiert dessen letzter Teil hier abgebildet ist.

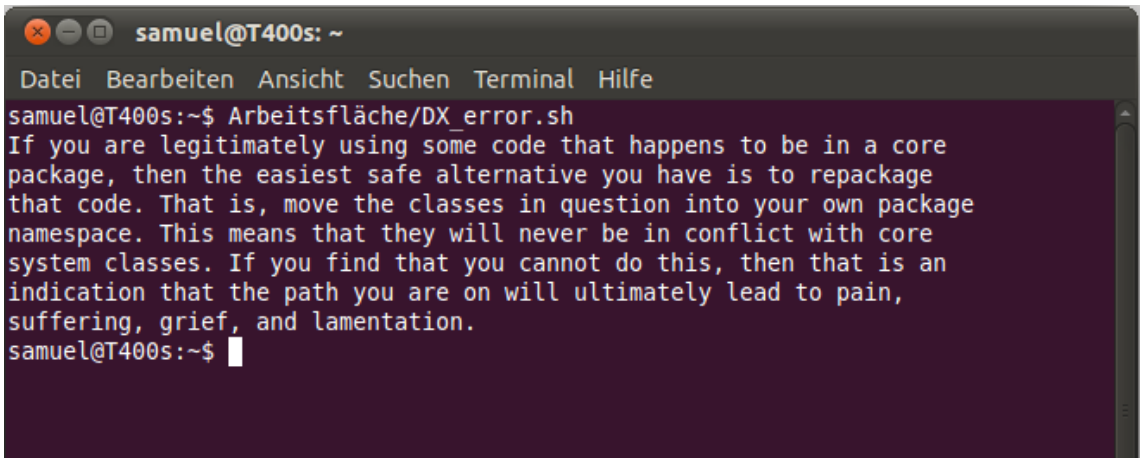


Abbildung 4.29: Fehlermeldung des Compilers beim Versuch Core Libraries in Android auszuführen.

Wie die Meldung besagt können Core Libraries in Androidapplikationen verwendet werden wenn sie aus dem Core Package in ein eigenes Package repacked werden. Dieser Vorgang wird realisiert indem das Androidprojekt nicht mehr durch Eclipse gebaut wird sondern man ANT verwendet. ANT ist ein Buildscript welches häufig für Java eingesetzt wird. ANT teil den ganzen Buildvorgang in verschiedene Targets auf welche neu gebuilded werden falls eine Abhängigkeit neu gebuildet wurde. Dadurch entsteht eine Abhängigkeitskette die stehts ein funktionierender Build garantiert ohne immer alles neu zu generieren. Der Repackager namens JarJar wird im post-compile Target aufgerufen. Er kopiert alle .class Files der Core Library in ein eigenes Package und ändern innerhalb der Classfiles die Referenzen auf das neue Package um.
Dieser Vorgang wird unter [Repackage] genauer beschrieben und wurde während dem Projekt auf alle erdenklichen Arten und Weisen durchgeführt. Allerdings hätte man sich den letzten Satz der im Bild oben gezeigter Fehlermeldung schon von Anfang an zu Herzen nehmen können. Weil dieser Satz extrem schön in Worte fasst, was man beim oben beschrieben Vorgang erlebt, ist er hier nochmals abgedruckt:

> If you find that you cannot do this, then that is an indication that the path you are on will ultimately lead to pain, suffering, grief, and lamentation.

5 Zusammenfassung

An dieser Stelle wird das Ergebnis der Arbeit nochmal mit kritischem Blick betrachtet. Welche Anforderungen erfüllt das Programm? Welche Dinge hätte man besser lösen können und wo hat die Software noch Potential für Weiterentwicklung?

5.1 Erreichte Ziele

Die meisten der gesetzten Ziele konnten erreicht werden. Dies hängt auch damit zusammen, dass die Ziele sehr offen formuliert waren und viel Spielraum bei der konkreten Realisierung liessen. Folgende wichtige Punkte konnten realisiert werden:

- Mit dem SiteInfo Screen wurde eine übersichtliche Darstellung aller relevanten Teile einer Brandmeldeanlage erstellt. Er hilft dem Benutzer sich zurecht zu finden und verwendet viele optische Hilfsmittel um dies zu erreichen.
- Fehler können multimedial zu allen Vorhandenen Geräten einer Brandmeldezentrale erfassen kann.
- Zu jeder Prüfung können Prüfungsberichte erstellt werden. Sowohl für die technische Weiterverarbeitung als auch direkt für den Kunden.
- Der Gebäudeplan kann angezeigt werden. Zudem können auf dem Plan Markierungen erstellt werden die danach automatisch, beim Auswählen des entsprechenden Elements, fokussiert werden.
- Mit der Synchronisation mit dem zentralen Server wurde eine Möglichkeit geschaffen Testdaten sowohl mit dem Firmenstandort als auch mit anderen Tablets auszutauschen.

Auf zwei Funktionen muss leider bei FireTabletPlus verzichtet werden da diese Features nicht realisiert wurden.

- Zentralen Checklisten
- Bessere Gatewayintegration

5.2 Vorhandene Probleme

Natürlich konnte nicht in allen Bereichen eine optimale Lösung gefunden werden. Dies lag vor allem an zwei Umständen. Erstens standen nur 14 Wochen für das gesamte Projekt zur Verfügung und es mussten irgendwo Schlussstriche gezogen werden. Zweitens Sind auf der Seite von Siemens die zum Teil benötigten Schnittstellen nicht bekannt oder vorhanden. Drittens waren einige Bereiche nicht Ziel der Arbeit aber gehören trotzdem zu einer guten Software dazu.

- **Schnittstelle zum Gateway**
  Die Interaktion mit der Brandmeldezentrale konnte nicht verbessert werden. Dies lag daran, dass das Gateway nicht mehr Funktionen bietet. Alternativ könnte man die selbe Schnittstelle benutzen die Sinteso verwendet, diese Informationen standen aber nicht zur Verfügung

- **Datensicherheit**
  Wie im Abschnitt Limitierungen und Einschränkungen auf Seite 38 bereits erwähnt fehlt jegliche Sicherheit sowohl für die Verbindung als auch für die Daten. In einem produktiven Umfeld müsste jedes Tablet mit Zertifikaten überprüft werden.

- **Datenablage auf dem Server**
  Der Server kontrolliert zur Zeit nicht ob eine Datei am gleichen Speicherort schon vorhanden ist oder nicht. Falls eine Datei mit gleichem Pfad und Namen ankommt wird die bestehende Datei einfach überschrieben. Es ist zudem nicht möglich Dateien wieder zu löschen. In einer produktiven Lösung müssten Dateien gemerged werden falls sie gleich heissen und nicht binär sind. Oder alle Daten werden in eine Datenbank abgefüllt.

- **Kompression der Daten**
  Wenn eine komplette Brandmeldeanlage zum Server geschickt wird, wird jede Datei einzeln geschickt. Dies ist vor allem bei schlechter Verbindungsqualität sehr ungünstig. Alle Daten müssten komprimiert über das Netz gesendet werden. In FireTabletPlus wurde auf diese Funktion verzichtet weil man die Logik im Server so klein wie möglich halten wollte.

- **Positionsinformationen im Plan**
  Die Koordinaten der Brandmelder auf dem Planbitmap sind in einer Property-Datei gespeichert. Es wurde absichtlich darauf verzichtet die Koordinaten in den *Devices* und dem XML zu speichern, da diese Information zum Plan gehört. Ein Brandmelder weiss also absichtlich nicht wo er sich im Plan befindet. In Zukunft muss ein CAD-Plan verwendet werden der sämtliche Informationen enthält.

5.3 Künftige Lösungen

In Zukunft gibt es viel Arbeit zu tun falls man eine ähnliche Software wie FireTablet-Plus in einem produktiven Umfeld einsetzen will. Einige Vorschläge werden anschliessend aufgeführt.

- **Ausgereifter Server**
  Sehr viel Arbeit kann in die Entwicklung eines für Siemens geeigneten Servers gesteckt werden. Der Server sollte einige Funktionen haben:
  - Schnittstellen zu allen benötigten Bereichen wie Buchhaltung, Lohnabrechnung, Kundendatenbank usw...
  - Userverwaltung damit jedes Tablet unterschieden werden kann.
  - Zugriffmanagement um gleichzeitig mehrere Zugriffe auf eine Site zu erlauben.
  - Security

  - ...

- **Gebäudeplan mit mehr Informationen**
  Der Gebäudeplan kann mit viel mehr Informationen angereichert werden z.B. wo sich alle Elektrischen Leitungen befinden, wo alle Zonen und Sections sind.

- **Plan nach Kompass ausrichten**
  Der Gebäudeplan dreht sich in der aktuellen Version nicht. Dies wäre aber möglich da die meisten Android-Tablets einen internen Kompass besitzen. Somit müsste der Servicetechniker nicht immer überlegen ob er jetzt den Plan richtig hält oder nicht.

- **Prüfpflücker mit Tablet verbinden**
  Der Prüfpflücker sollte sich z.B. mit Bluetooth mit dem Tablet verbinden können damit man direkt weiss welche Brandmelder erfolgreich getestet worden sind. Damit könnten auch Melder konfiguriert werden und z.B. die Kundentexte auf den Meldern geändert werden. Es setzt aber voraus, dass der Prüfpflücker eine Schnitstelle wie Bluetooth oder NFC (Near Field Communication) hat.

- **Arbeitszeit auf dem Tablet rapportieren**
  Techniker können ihre Arbeitsstunden auf dem Tablet rapportieren und so mithelfen den Kunden Rechnungen zu schreiben oder ihre eigenen Arbeitsstunden festzuhalten.

6 Projektmanagement

In diesem letzten Teil der Dokumentation wird das Projektmanagement dokumentiert. Hier befindet sich die Stundenübersicht der aufgewendeten Zeit, der Projektplan und eine Auflistung der verwendeten Software. Am Schluss dieses Kapitels befinden sich die beiden Erfahrungsberichte.

6.1 Projektplan

Auf der nächsten Seite ist der Projektplan abgebildet. Die wichtigen Features sind von 1 bis 5 nummeriert und werden im Kapitel Implementierung von FireTabletPlus auf Seite 43 dokumentiert. Alle anderen Arbeitspakete werden im Folgenden erklährt.

- **Infrastruktur einrichten**
  Dieses Arbeitpaket umfasst alle Arbeiten die mit der Projektwiedererstellung zu tun haben damit anschliessend gearbeitet werden kann. Darunter befindet sich zum Beispiel die IDEs konfigurieren und installieren, Eventsimulationsserver testen, Git Repo einrichten usw...
- **Aufgabenstellung erstellen**
  Dieser Arbeitschritt beinhaltet die Erstllung des Featurekatalogs. Die Ergebniss sind unter Featurekatalog auf Seite 25 fest gehalten.
- **Zirkuläre Abhängigkeiten**
  Das komplete Refactoring wie im Abschnitt Der Sourcecode von FireTablet auf Seite 21 beschrieben ist, fällt unter diese Kategorie.
- **Meetings**
  Die Meetings fanden wärend des gesammten Projekts statt.
- **Papierprototypen erstellen**
  Ganz am Anfang des Projekt wurden Prototypen für das Userinterface auf Papier erstellt. Dokumentiert ist diese Arbeit bei Gui Prototyp auf Seite 29
- **Dokumentation**
  Die Dokumentation sollte verteilt über das gesammte Projekt erstellt werden. Der grosse Teil entstand aber in den letzten beiden Wochen des Projekts. Aus diesem Grund ist das Arbeitspaket zum jetzigen Zeitpunkt auch noch nicht ganz abgeschlossen. Schon fürher dokumentiert wurde das Gui-Design sowie komplizierte Probleme wärend dem Projekt.
- **Bugfix & Aufräumen**
  Mit diesem Arbeitspaket wird dem Programm der letzte feinschliff gegeben. Es werden kleine Anpassungen gemacht, Fehler behoben und der Code von Thomas Corbat gereviewt.

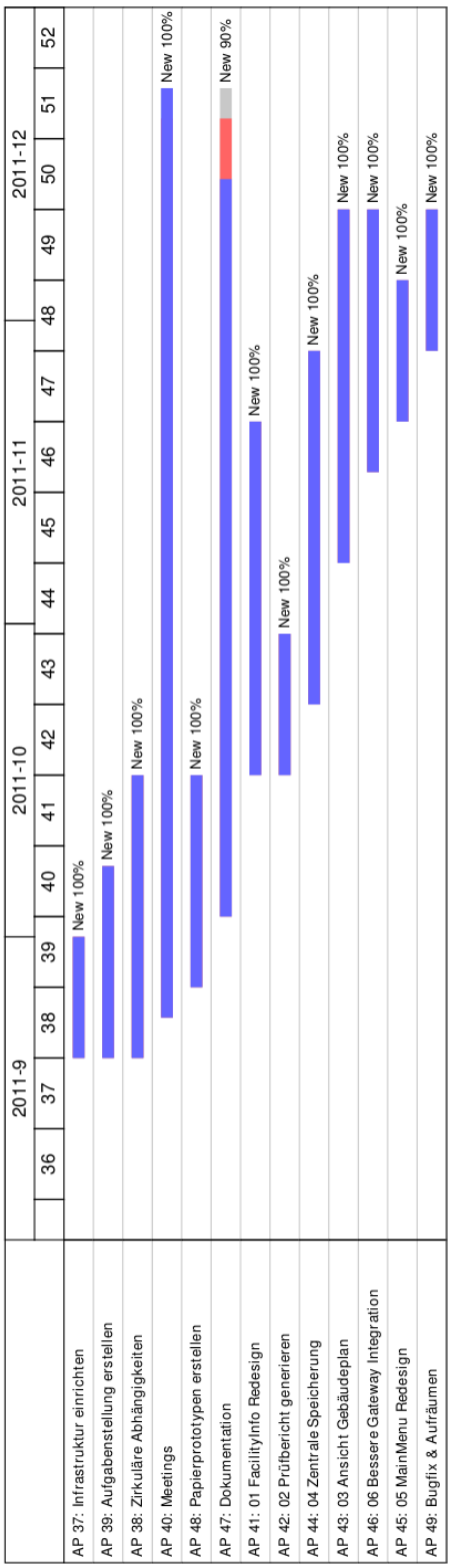


Abbildung 6.1: Projektplan als Ganttdiagramm von Redmine gennerierert.

6.2 Stundenübersicht

Die Stunden wurden zu den Arbeitspacketen rapportiert und sind in der unten abgebildeten Grafik nach Aufwand pro Woche und Person dargestellt. Die braune Lienie zeigt den Sollwert von 26h pro Woche und Person in Prozent an. Die grüne Linie stellt dar um wieviele Prozent der Soll unter oder überschritten wurde. Das Arbeitspensum in der zweitletzten Woche bricht stark ein da diese Woche direkt nach dem Code-Freeze kam und die Motivation erst wieder gefunden werden musste.

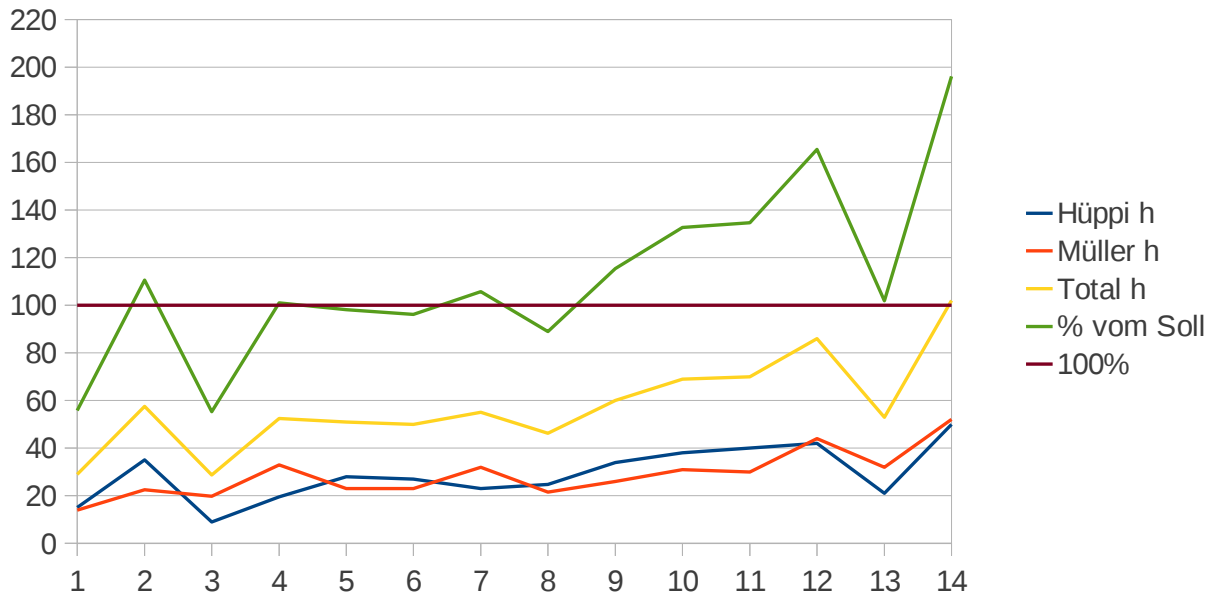


Abbildung 6.2: Projektplan als Ganttdiagramm von Redmine genneriert.

6.3 Verwendete Software

Für die Entwicklung von FireTabletPlus wurden die folgenden Programme eingesetzt:

Software	Einsatzbereich
Eclipse Indigo 3.7	glside
Android Development Toolkit 16.0.1	Eclipse Plugin für Android Entwicklung
GIT	Version Control System
Hudson 1.396	Continuous Integration Server
Texmaker 2.0	LaTeX Editor für die Dokumentation
Dia 0.97.1	Grafikwerkzeug zur Erstellung von Diagrammen
Gimp 2.7.3	Bildbearbeitung
Viso 2010	Erstellung von Diagrammen
Astah Community	Erstellung Klassendiagramme
Google CodePro 7.1	Codemetrics
Structure101 for Java 3.5	Codemetrics
Findbugs 2.0	Codeanalyse und Fehlersuche
Inkscape 0.48	Vektorbasierte Bildbearbeitung
Ubuntu 11.04 und Windows 7	Betriebssystem Arbeitsrechner
Redmine	Projektmanagement Server

Tabelle 6.1: Bei der Entwicklung von FireTablet verwendete Software

6.4 Erfahrungsberichte

6.4.1 Erfahrungsbericht Samuel Hüppi

Am Ende der Bachelorarbeit bin ich auf der einen Seite froh die Arbeit hinter mir zu haben, auf der anderen Seite wäre man ab Mitte der 14 Wochen auch endlich richtig in Schwung gekommen. In dieser Phase hatte ich viele Ideen was man alles noch implementieren könnte. Es ist darum auch schade wenn sich das Projekt langsam dem Ende zuneigt. Man sollte das Ende aber immer vor Augen halten da die Zeit am Schluss sowieso schon knapp ist.
Während dem Projekt konnte ich sehr viel lernen. Vor allem was das Programmieren anbelangt, habe ich viele neue Lösungswege gesehen die ich vorher zwar kannte aber mir nicht geläufig waren. Eine grosse Unterstützung in diesem Zusammenhang war für mich mein Projektpartner Ramon Müller, der im Gegensatz zu mir schon einige Jahre mit Java gearbeitet hat. Allgemein empfand ich die Zusammenarbeit als sehr angenehm mit produktiven Diskussionen und Anregungen. Ich konnte zudem feststellen, dass es ohne Probleme möglich ist auf zwei verschiedenen Betriebsystemen zusammen zu arbeiten. Ramon benutze Windows, ich Ubuntu und es gab nahezu nie Probleme. Ein Vorteil war dabei aber sicher die Verwendung von Latex als Dokumentationswerkzeug da es auf beiden Platformen reibunglos läuft. Sehr positiv überrascht war ich von unserem Projektbetreuer von Siemens, welcher sich nahezu jede Woche Zeit nahm um ein Netmeeting oder ein Livemeeting zu halten. Er war zudem sehr interessiert an unserer Arbeit und hatte obwohl Softwareentwicklung nicht sein Fachgebiet war ein breites Wissen im Bereich Informatik. Android interessiert mich auch nach dieser Arbeit sehr und es ist je tiefer man in das Sys-

tem hinein sieht je interessanter damit zu arbeiten. Ich hoffe nun auf einen erfolgreichen Abschluss des Studiums und bin gespannt welche Projekte es in Zukunft zu erledigen gibt.

6.4.2 Erfahrungsbericht Ramon Müller

Mir persönlich hat die Bachelorarbeit sehr viel Spass bereitet. Wir hatten ein gutes Thema und auch einen praktischen Bezug, sowie gutes Feedback von Seiten Siemens. Er war sehr spannend in Zusammenarbeit mit Siemens neue Features und Lösungen auszuarbeiten, die auch der Anforderung in der praktischen Nutzung genügen.
Die zwei Wochen weniger Zeit für die Bachelorarbeiten wären so gegen den Schluss der Arbeit doch sehr wünschenswert gewesen, da noch viele tolle Ideen in unseren Köpfen herumschwirrten, leider aber die Zeit einfach zu knapp wurde diese umzusetzen. Der zusätzliche Druck der Bachelorarbeit unter dem Semester war spürbar und ich bin froh dass ich nicht mehr viele Module besuchen musste, ansonsten wäre das ganz schön knapp geworden mit der investierten Zeit. Die Zusammenarbeit mit meinem Projektpartner Samuel Hüppi war sehr angenehm. Wir hatten viele interessante Diskussionen und wir haben uns sehr gut ergänzt. Samuel Hüppi hatte natürlich anfangs etwas mehr Einsicht in das Projekt als ich, da er die Vorgängerarbeit als Studienarbeit abgeschlossen hat. Er hatte auch mehr Erfahrung mit dem Android Framework, wo er mich sehr gut unterstützen konnte. Auch die Zusammenarbeit mit Herrn Knellwolf, unserem Partner bei Siemens, war meiner Meinung nach Erfolg geprägt. Herr Knellwolf hat sich sehr für unsere Arbeit interessiert und wir standen das Ganze Projekt über in engem Kontakt mit wöchentlichen NetMeeting Sitzungen, was ich durch die wichtige Position die Herr Knellwolf bei Siemens wahrnimmt als nicht selbstverständlich erachte.
Das Arbeiten mit Android hat sehr viel Spass gemacht und ich habe wieder sehr viel dazugelernt. Das arbeiten mit Latex hat erstaunlich gut geklappt, obwohl ich bisher noch gar nicht damit gearbeitet habe. Ich bin im Allgemeinen sehr zufrieden mit unserer Arbeit und hoffe dass unsere Arbeit vielleicht als Inspiration für ein zukünftiges Siemens Produkt dienen kann.

Literaturverzeichnis

[BA FireTablet] Bachelorarbeit von Daniel Bobst in Zusammenarbeit mit Samuel Hüppi
Daniel Bobst & Samuel Hüppi
2011
Das PDF befindet im Ordner "referenzdokumenteäuf der CD.

[Db4o] Datenbank für Objekte
Versant
http://www.db4o.com/
Abgerufen am: 25.5.2011

[FS20 BacNet Spec] FS20 Sinteso Fire Detection System MP3.0
BACnet Interface Description Specification
Siemens Building Technologies / Fire Safety & Security Products
2010
Das PDF befindet im Ordner "referenzdokumenteäuf der CD.

[InspectionReport] Von FireTablet gennerierter Report für eine Inspektion.
Das PDF befindet im Ordner "referenzdokumenteäuf der CD.

[Android 2] Grundlagen und Programmierung
Arno Becker, Marcus Pant
dpunkt.verlag
http://www.dpunkt.de/buecher/3319.html
2010

[The Agile Samurai] How Agile Masters Deliver Great Software
Jonathan Rasmusson
The Pragmatic Programmers
The Pragmatic Bookshelf
2010

[Pruefprotokoll] PruefprotokollFuerBMA1.xls
Prüfprotokoll
Siemens Building Technologies
25.05.2010

[Sardine Library] Sardine ist eine sehr einfach zu verwendende WebDav Library.
http://code.google.com/p/sardine/
22.12.2011

[Repackage] Anleitung zur Verwendung von Core Libraries des javax Packages unter Android.

http://code.google.com/p/dalvik/wiki/JavaxPackages
22.12.2011

[Broadcast] Android Reference zum Thema Broadcast
http://developer.android.com/reference/android/content/BroadcastReceiver.html
22.12.2011

[Activity Reference] Activity auf der Android Reference
Google
http://developer.android.com/reference/android/app/Activity.html
Abgerufen am: 30.5.2011

[Painless Threading] Verwendung von Threads zusammen mit Activities
Google
http://developer.android.com/resources/articles/painless-threading.html
Abgerufen am: 30.5.2011