

**FBS-Scanner**

**Android-App zur Erfassung von Schulhardware**

Abgabetermin: 18.01.2019

*von Christian Wende und Sebastian Schulz*

Inhaltsverzeichnis

[**1. Projektbeschreibung** 3](#_Toc535346039)

[**1.1 Projektziel** 3](#_Toc535346040)

[**1.2 Projektbegründung** 3](#_Toc535346041)

[**1.3 Projektschnittstellen** 4](#_Toc535346042)

[**2. Projektplanung** 4](#_Toc535346043)

[**2.1 Zeitplanung** 4](#_Toc535346044)

[**2.2 Ressourcenplanung** 4](#_Toc535346045)

[**2.3 Entwicklungsprozess** 4](#_Toc535346046)

[**3. Analysephase** 5](#_Toc535346047)

[**3.1 Ist-Analyse** 5](#_Toc535346048)

[**3.2 Soll-Zustand** 5](#_Toc535346049)

[**3.3 Kostenkalkulation** 6](#_Toc535346050)

[**4. Entwurfsphase** 7](#_Toc535346051)

[**4.1 Auswahl der Programmiersprache** 7](#_Toc535346052)

[**4.2 Ablauflogik** 7](#_Toc535346053)

[**4.3 Entwurf der Benutzeroberfläche** 7](#_Toc535346054)

[**5. Implementierungsphase** 8](#_Toc535346055)

[**5.1 Implementierung der Benutzeroberfläche** 8](#_Toc535346056)

[**5.1.1 Allgemeines** 8](#_Toc535346057)

[**5.1.2 Hauptmenü** 8](#_Toc535346058)

[**5.1.3 Scannen** 8](#_Toc535346059)

[**5.1.4 Anzeige Räume** 9](#_Toc535346060)

[**5.1.5 Anzeige Raumdetails** 9](#_Toc535346061)

[**5.2 Implementierung des Backends** 9](#_Toc535346062)

[**5.3 Qualitätskontrolle und Tests** 9](#_Toc535346063)

[**5.3.1 Genauer Programmablauf** 10](#_Toc535346064)

[**5.3.2 Beschreibung der Test** 10](#_Toc535346065)

[**5.3.3 Manuelle Tests** 10](#_Toc535346066)

[**6. Abnahme und Einführung** 10](#_Toc535346067)

[**6.1 Abnahme** 10](#_Toc535346068)

[**6.2 Installation** 10](#_Toc535346069)

[**7. Dokumentation** 11](#_Toc535346070)

[**7.1 Entwicklerdokumentation** 11](#_Toc535346071)

[**7.2 Benutzerdokumentation** 11](#_Toc535346072)

[**8. Projektbewertung** 11](#_Toc535346073)

[**8.1 Soll-/Ist-Vergleich** 11](#_Toc535346074)

[**8.2 Ausblick** 12](#_Toc535346075)

[**8.3 Fazit** 12](#_Toc535346076)

[**Glossar** 12](#_Toc535346077)

[**Literaturverzeichnis** 13](#_Toc535346078)

[**A Anhang** 14](#_Toc535346079)

[**A.1 Benutzerdokumentation** 14](#_Toc535346080)

[**A.2 Diagramme** 17](#_Toc535346081)

[**A.2.1 Programmablaufplan** 17](#_Toc535346082)

[**A.2.2 Use-Case-Diagramm** 18](#_Toc535346083)

[**A.2.3 Klassendiagramme** 19](#_Toc535346084)

[**A.3 Oberflächenentwürfe** 21](#_Toc535346085)

[**A.4 Screenshots** 23](#_Toc535346086)

[**A.4.1 Projekt-Explorer** 23](#_Toc535346087)

[**A.4.2 Code-Ausschnitte** 24](#_Toc535346088)

[**A.4.3 Fertige App** 25](#_Toc535346089)

# **1. Projektbeschreibung**

## **1.1 Projektziel**

Ziel des Projektes ist es, eine Android-App zu entwickeln, welche zur Erfassung schulinterner Hardware (z.B. Rechner, Bildschirm usw.) genutzt werden kann. Bei der Erfassung soll es möglich sein, einen Barcode (Inventarnummer) zu scannen und weitere Daten zum jeweiligen Gerät aufzunehmen. Diese Daten sollen anschließend in einer

csv-Datei abgespeichert werden, welche für den Benutzer zugänglich ist.

## **1.2 Projektbegründung**

Die Ferdinand-Braun-Schule benötigt zur besseren Verwaltung der Schulhardware eine App.

Aktuell gibt es nur eine Liste für die Gesamtheit aller Geräte und keine explizite Zuordnung zu den Klassenräumen. Im Rahmen dieses Projektes soll sich dies ändern. Durch die Entwicklung dieser App soll für jeden beliebigen Raum in der Schule eine schnelle und effiziente Erfassung von Geräten ermöglicht werden. Außerdem kann in Fällen von Diebstahl oder Sachbeschädigung eine bessere Einschätzung und Zuordnung erfolgen. Auftraggeber und Ansprechpartner bei diesem Projekt ist Herr Reuter.

## **1.3 Projektschnittstellen**

Die App kann ab der Android-Version 5 (Lollipop, API 21) genutzt werden. Um den Scanvorgang von Barcodes zu ermöglichen, wird von einer externen Bibliothek Gebrauch gemacht. Der Benutzer wird zur Installation einer zusätzlichen App aufgefordert, die die Erfassungs-App bei Bedarf beansprucht. Grundlage ist dabei die “Google-Standard-Bibliothek“.

# **2. Projektplanung**

## **2.1 Zeitplanung**

Für die Umsetzung des Projektes ist eine Gesamtzeit von 35 Stunden vorgesehen. Diese wurden auf diverse Projektphasen verteilt und verschieden stark gewichtet. Die folgende Tabelle zeigt die Zeitplanung des Projektes:

|  |  |
| --- | --- |
| **Projektphase** | **Geplante Zeit** |
| Projektplanung | 4 h |
| Analysephase | 2,5 h |
| Entwurfsphase | 3,5 h |
| Implementierung inkl. Tests | 18 h |
| Abnahme und Einführung | 2 h |
| Dokumentation | 5 h |
| **gesamt:** | **35 h** |

## **2.2 Ressourcenplanung**

Bei der Ressourcenplanung wurde darauf geachtet, dass die Software zur freien Verfügung steht oder die Lizenzrechte bereits vorhanden sind, um das Budget möglichst gering zu halten. Die für die Softwareentwicklung eingesetzte IDE ist Android Studio, da sie über eine Vielzahl von nützlichen Funktionen verfügt und den Entwicklern bereits bekannt ist.

Da es sich hierbei um ein relativ kleines Projekt handelt, entschied man sich, die Planung vollständig den beiden Auszubildenden zu überlassen. Dadurch kommt man zum einen auf geringe Kosten und zum anderen ermöglicht man den Azubis einen enormen Lernfortschritt.

## **2.3 Entwicklungsprozess**

Bevor mit der Realisierung des Projektes begonnen werden konnte, musste ein geeigneter Entwicklungsprozess gewählt werden, welcher die Vorgehensweise definiert, nach der die Umsetzung erfolgen soll. Die beiden Entwickler haben sich unter Abstimmung mit dem Auftraggeber für einen agilen Entwicklungsprozess entschieden. Dieser ermöglicht es, flexibler und transparenter mit dem Kunden zusammenzuarbeiten, da regelmäßig Rücksprache gehalten wird und somit schneller auf kurzfristige Wünsche und Änderungen eingegangen werden kann. Die agile Softwareentwicklung zeichnet sich zudem durch einen möglichst geringen bürokratischen Aufwand aus, wodurch ein schneller Einsatz des entwickelten Systems möglich wird. Aufgrund der Entscheidung für diesen Entwicklungsprozess sind die geplanten Zeiten für Projektplanung, Analysephase und Entwurfsphase im vorigen Abschnitt relativ niedrig angesetzt worden. Die wichtigste Phase ist die der Implementierung, da viele Besonderheiten erst in diesem Zeitraum geklärt werden können und auch das Testen in dieser geschieht. Durch regelmäßige Kommunikation mit dem Auftraggeber und häufiges Feedback können die Entwickler deutlich besser den gegebenen Anforderungen gerecht werden. Der ständige Kontakt hat weiterhin den Vorteil, dass sich der Auftraggeber bereits während des Entwicklungsvorgangs mit dem entstehenden Produkt vertraut machen kann. Dadurch kann Zeit bei der abschließenden Abnahme und Einführung gespart werden, weshalb auch für diese Phase nur wenig Zeit vorgesehen wurde.

# **3. Analysephase**

## **3.1 Ist-Analyse**

Wie bereits in Punkt 1.2 beschrieben, sind die Geräte zwar mit Inventarnummern festgehalten, aber der jeweils zugehörige Raum ist nicht angegeben. Somit muss man die Geräte im Zweifelsfall Raum für Raum durchsuchen. Dies stellt einen sehr großen Zeitaufwand dar.

## **3.2 Soll-Zustand**

Die Ferdinand-Braun-Schule wünscht sich eine App, bei der man die Inventarnummer von Geräten (PC, Monitor, Beamer, Drucker) erfassen kann. Hierzu soll der Benutzer ggf. Bemerkungen tätigen können. Der Barcodescanner muss nicht in der App integriert sein, sondern kann auch extern über eine andere App eingebunden werden. Ist ein Raum vollständig erfasst, so soll eine CSV-Datei erstellt werden. Diese kann dann über USB auf den PC übertragen werden.

Sofern nach Umsetzung der wesentlichen Anforderungen noch Zeit übrig ist, können noch weitere Features wie ein optisch ansprechenderes und übersichtlicheres Design, das Anzeigen sowie das Löschen von Räumen oder eine Verlinkung zur Homepage der FBS implementiert werden.

Folgende Tabelle zeigt alle Anforderungen der Ferdinand-Braun-Schule sortiert nach absteigender Wichtigkeit, wobei die oberen als Grundfunktionen einzuordnen sind und die unteren nicht zwingend erforderlich eingebaut werden müssen:

|  |
| --- |
| - Scannen von Barcodes |
| - pro Raum CSV-Datei erstellen |
| - Zugriff auf Dateien über USB |
| - Anzeige von Räumen |
| - Löschen von Räumen |
| - Verlinkung zur FBS-Homepage |
| - Optimierung des Designs |

Um eine bessere Übersicht über die Anforderungen zu haben, wurde innerhalb der Analysephase ein Use-Case-Diagramm erstellt, welches im Anhang [A.2.2](#_A.2.2_Use-Case-Diagramm) zu finden ist.  
Es bildet in grafischer Form alle Funktionen ab, die aus Endanwendersicht benötigt werden.

## **3.3 Kostenkalkulation**

Die Kosten, die während des Projektes anfallen, sollen in Folgendem kalkuliert werden. Dafür müssen neben den Personalkosten auch die Aufwendungen für Ressourcen wie Hardware, Software, Arbeitsplatz usw. berücksichtigt werden. Da die genauen Personalkosten nicht genannt werden dürfen, wird die Kalkulation anhand von Stundensätzen durchgeführt.  
Für einen Auszubildenden wurde ein Stundensatz von 16,50€ gewählt, für einen normalen Mitarbeiter 28,00€. Für die zusätzlichen Ressourcenaufwendungen wurde ein pauschaler Stundensatz von 10,00€ festgelegt.

Die gesamten Projektkosten lassen sich aus folgender Tabelle entnehmen­:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Vorgang** | **Mitarbeiter** | **Zeit** | **Personal** | **Ressourcen** | **Gesamt** |
| Umsetzungskosten | 2 Auszubildende | 35 h | 577,50 € | 350,00 € | 927,50 € |
| Kundengespräche | 1 Mitarbeiter | 5 h | 140,00 € | 50,00 € | 190,00 € |
| Abnahme | 1 Mitarbeiter | 2 h | 56,00 € | 20,00 € | 76,00 € |
| **Projektkosten gesamt:** 1193,50 € | | | | | |

# **4. Entwurfsphase**

## **4.1 Auswahl der Programmiersprache**

Wir haben uns für die Programmiersprache Java entschieden, da diese im Bereich der Android-Entwicklung führend ist und wir in dieser Sprache bereits einen angemessenen Kenntnisstand haben.

## **4.2 Ablauflogik**

Um sich besser vorstellen zu können, wie der genaue Ablauf des Programms werden soll, hat man vor Beginn der Entwicklung einen Programmablaufplan erstellt, der alle möglichen Benutzereingaben abbildet. Er wurde in Zusammenarbeit mit dem Kunden entwickelt, um Missverständnisse zu vermeiden und den Entwicklungsprozess zu optimieren.   
Der Ablaufplan ist in Anhang [A.2.1](#_A.2.1_Programmablaufplan) zu finden.

## **4.3 Entwurf der Benutzeroberfläche**

Beim Entwurf der Benutzeroberfläche hat man sich zuerst Gedanken gemacht, wie viele Activities innerhalb der App eingesetzt werden sollen. Nach diversen Überlegungen und kleineren Skizzen mit Stift und Papier ist man zum Schluss gekommen, dass vier Stück am sinnvollsten sind.   
Beim Aufrufen der App soll zunächst ein Hauptmenü mit dem FBS-Logo und zwei Buttons für die Erfassung und die Anzeige von Räumen angezeigt werden. Des Weiteren gibt es eine Activity, die der Erfassung eines Raumes dient. In ihr gibt es Felder für Raumname, Gerätetyp, Inventarnummer und eine Notiz. Zudem sind ein Button zum Scannen, Hinzufügen eines Gerätes und Abschließen der Raumerfassung vorhanden.  
Die beiden anderen Activities sind dafür da, alle bereits erfassten Räume anzuzeigen und die Details jedes Raumes anzuschauen. Ein MockUp aller vier Activities ist im Anhang [A.3](#_A.3_Oberflächenentwürfe) zu finden. Es ist Grundlage für die Implementierung des Frontends.

# **5. Implementierungsphase**

## **5.1 Implementierung der Benutzeroberfläche**

### **5.1.1 Allgemeines**

Bei der Implementierung der Benutzeroberfläche wurde darauf geachtet, die verfügbaren Funktionen möglichst transparent zu visualisieren. Das Layout hat über alle Actitvities hinweg eine mintgrüne Hintergrundfarbe. Die verwendeten Buttons hingegen sind in einem Rotton gehalten und dessen Beschriftung ist weiß. Außerdem gibt es auf jeder Seite in der rechten oberen Ecke einen Zurück-Button (Hauptmenü ausgeschlossen). Des Weiteren verfügt die App über eine Titlebar, die in Form einer kurzen Beschreibung anzeigt, in welcher Activity man sich befindet.

### **5.1.2 Hauptmenü**

Das Hauptmenü wird als aller erstes angezeigt, wenn die App gestartet wird. Es besteht aus 2 Buttons und einem Imagebutton, der das FBS-Logo zeigt. Der Imagebutton öffnet im Browser die FBS-Homepage. Der Button mit dem Text „Raum erfassen“ öffnet einen kleinen Dialog, der zur Eingabe eines Raumnamens auffordert. Es handelt sich hierbei um ein Edit-Text-Steuerelement, dessen Inhalt der Benutzer füllt. Ist der Raumname leer, so wird man mit einer Toast-Meldung darauf hingewiesen, dass dieser für die Erfassung erforderlich ist.

Im Hauptmenü gibt es einen weiteren Button mit dem Text „Räume Anzeigen“. Dieser öffnet die Anzeige-Activity. Sind keine erfassten Daten vorhanden, so wird eine Meldung ausgegeben.

### **5.1.3 Scannen**

In dieser Activity sehen wir links ein TextView mit Raumnamen, daneben steht der Raum, welchen wir in dem Dialog eingetragen haben. Darunter ist Gerätetyp, rechts davon ist ein Spinner, klickt man auf diesen drauf, so öffnet sich dieser und man kann sich zwischen den einzelnen Gerättypen unterscheiden. Standardmäßig ist dieser PC. Dann sieht man Inventarnummer und daneben ein TextEdit mit „Letzter Scan“. Hier kann man entweder das Feld manuell befüllen, oder man lässt dies mithilfe des Buttons „Scannen“ füllen. Dieser öffnet dann einen Barcodescanner, oder fragt ob man einen Installieren will. Ist die Inventarnummer erfolgreich eingetragen, so kann man nun eine Bemerkung im „EditText“ eintragen. Klickt man nun auf den Button „Gerät hinzufügen“ so wird das Gerät hinzugefügt, die Bemerkung und die Inventarnummer werden wieder auf „“ gesetzt und unter dem Button erscheint nun die Anzahl der erfassten Geräte. Ist die Inventarnummer jedoch vorher „“ dann wird ein Toast erscheinen mit folgender Nachricht „“Inventarnummer ist leer!“ Klickt man nun auf den Button „Raumerfassung abschließen“, wird die CSV-Datei erzeugt, und man ist wieder im Hauptmenü.

Drückt man die Zurück-Taste, erscheint eine Meldung, ob man die „Erfassung abbrechen“ will. Hierbei wird unterschieden, ob sich Daten im Zwischenspeicher befinde, oder ob kein Scan stattgefunden hat. (siehe Anhang A.x.1)

### **5.1.4 Anzeige Räume**

Hier befindet sich ein Listview mit den Angezeigten Räumen. Rechts oben befinden sich 3 Punkte, mit diesen kann man alle Dateien löschen. Hierbei wird wieder ein Dialog abgefragt, ob man wirklich alle Daten löschen will. Zudem kann man auch einzelne Räume löschen, dies funktioniert mit einem länger gedrückt halten des Raumes. Dann öffnet sich ein kleines Popup-Menü, bei der man die Option löschen hat. Drück man nur kurz auf einen Raum, so öffnet sich die Activty „Raumdetails“. Sollten sich Daten während der App ändern(ersetzen

der Datei, während Handy am PC offen ist) kann man bequem über das „Swipe-to-Refresh“-Layout die Daten aktualisieren. (siehe Anhang A.x.1)

### **5.1.5 Anzeige Raumdetails**

In den Raumdetails sieht man eine Tabelle, hier kann man wieder über das „Swipe-to-Refresh“-Layout die Daten aktualisieren. Über den Zurückbutton kommt man in das vorherige Layout.

Alle Screenshots zu den genannten Activities sind in Anhang [A.4.3](#_A.4.3_Fertige_App) zu finden.

## **5.2 Implementierung des Backends**

## **5.3 Qualitätskontrolle und Tests**

### **5.3.1 Genauer Programmablauf**

Der genaue Programmablauf findet sich im Anhang wieder, dieser wurde mit einer Android Studio Extension automatisch erzeugt. Wir haben kleine Änderungen vorgenommen und bei unübersichtlichen stellen Kommentare verfasst. *Bild im Anhang A6.1*

### **5.3.2 Beschreibung der Test**

Da wir das System des agilen Protoyping verwendet habe, gab es keine automatischen Tests. Diese wurden alle manuell angesteuert. Bugs wurden schnell gefixt. Es wurde Schrittweiße geschaut ob die jeweilige Komponente funktioniert, dabei wurde schon während der Programmierung Tests gemacht, ob die bis dahin vorgestellten Anforderungen erfüllt sind. Nachdem wir der Meinung waren, dass die Programm fertig ist, wurden ausführlichere Test durchgeführt.

### **5.3.3 Manuelle Tests**

Die ausführlichen Tests liefen soweit erfolgreich ab, kleinere Bugs, die man erst nach der Programmierung sehen konnte, wurden mit wenig Zeitaufwand behoben, dadurch bekommt der Kunde eine bugfreie App. Bei dem Test wurden alle möglichen Varianten durchgetestet und die wurden mit ein paar Beispieldaten gefüllt, um zu schauen, wie es sich in der Praxis schlägt, oder es Verbesserungsvorschläge bekommen könnte

# **6. Abnahme und Einführung**

## **6.1 Abnahme**

## **6.2 Installation**

Die Installation erfolgt über den Link:

<https://github.com/KleeSchulz/Schulprojekt-Scanner/tree/master/APK-Datei>.

Diese Datei muss gedownloadet werden. Nach erfolgreichem Download führt man die APK-Datei aus. Diese sollte sich dann ohne Probleme installieren lassen. Sollte die APK-Datei sich nicht installieren lassen, so muss man in die Einstellungen. Sicherheit > Geräteverwaltung > Unbekannte Herkunft zulassen. Danach sollte sich die APK-Datei ohne Probleme installieren lassen.

# **7. Dokumentation**

## **7.1 Entwicklerdokumentation**

Vieles über die Entwicklung geht über die im Code geschrieben Kommentare hervor. Zudem sind im Anhang unter den Punkten xyz folgende Diagramme enthalten:

* Use-Case-Diagramm
* Ablaufdiagramm
* Klassendiagramm

Mit diesen Diagrammen sollte es relativ einfach und simpel sein, die App zu verstehen und weiter zu entwickeln.

## **7.2 Benutzerdokumentation**

Da der Kunde eine ausführliche Benutzerdokumentation haben will, findet sich diese im Anhang unter dem Punkt A. 1 wieder.

# **8. Projektbewertung**

## **8.1 Soll-/Ist-Vergleich**

|  |  |  |  |
| --- | --- | --- | --- |
| **Projektphase** | **Soll** | **Ist** | **Differenz** |
| Projektplanung | 4 h | 3,5 h | -0,5 h |
| Analysephase | 2,5 h | 2,5 h | 0 h |
| Entwurfsphase | 3,5 h | 3 h | -0,5 h |
| Implementierung inkl. Tests | 18 h | 19 h | +1 h |
| Abnahme und Einführung | 2 h | 2 h | 0 h |
| Dokumentation | 5 h | 5 h | 0 h |
| **gesamt** | **35 h** | **35 h** | **0 h** |

Bei rückblickendem Betrachten des Projektes kann festgestellt werden, dassalle Anforderungen des Kunden erfüllt wurden. Die in Abschnitt 2.1 erstellte Zeitplanung konnte nahezu eingehalten werden.   
In folgender Tabelle ist nun ein Soll-/Ist-Vergleich zu sehen, der die geplanten und tatsächlichen Zeiten der einzelnen Phasen gegenüberstellt und mögliche Differenzen aufzeigt. Es ist zu erkennen, dass nur geringfügig Differenzen zustande gekommen sind, welche sich gegenseitig kompensiert haben. Die Gesamtzeit von 35 Stunden wurde somit nicht über- oder unterschritten.

## **8.2 Ausblick**

Bei der App könnte man noch viele kleine Details hinzufügen, sei es das Bearbeiten in der oder einfach nur das der User selbst seine Gerätetypen hinzufügen kann. Das Kernprogramm als solches bietet jedoch keine Veränderungen mehr.

## **8.3 Fazit**

Wir haben bei der Umsetzung des Projekts einiges gelernt und es war eine interessante Erfahrung. Zum einem hat man viel über ein schönes, jedoch auch übersichtliches Layout gelernt, zum anderen hat man erste Einblicke in die Android-Welt bekommen, samt ihren Tücken. Man hat gelernt, wie man Dateien aus Android erzeugt, das Speichersystem funktioniert und wie und welche Berechtigungen man benötigt.

# **Glossar**

# **Literaturverzeichnis**

# **A Anhang**

## **A.1 Benutzerdokumentation**

**Hauptmenü:**

Öffnet der Benutzer die App, ist man im Hauptmenü, findet man drei Buttons. Der oberste Button (FBS-Logo) öffnet die FBS-Homepage(URL: <https://www.ferdinand-braun-Schule.de/blog/category/news/>). Der zweite Button öffnet mit der Schrift „Raum erfassen“. Betätigt man diesen, öffnet sich eine Dialogbox mit dem Text „Raum erfassen“. Klickt man in den Edittext, öffnet sich die Tastatur und man kann den Raumnamen eingeben. Der letzte Button ist der Button „Räume anzeigen“ Hierbei wird eine weitere Activty geöffnet.

Screenshot: siehe A..4.3.1/A4.3.2

**Erfassung – Raum:**

Trägt man erfolgreich den Raumnamen ein, so kommt man zur Erfassungsmaske „Erfassung Raum“. Als erstes sieht man den Raum, darunter ist der Gerätetype zu sehen. Klickt man den Spinner, an, so öffnet sich dieser und man kann zwischen den Geräten „PC“,“Monitor“,“Drucker“,“Beamer“ auswählen. Standardmäßig ist der „PC“ ausgewählt. Hat man den Gerätetyp bestimmt, so kann man die Inventarnummer erfassen. Hierbei hat man zwei Möglichkeiten. Möglichkeit Nummer eins , man klickt in das Edittext „Letzter Scan“ und trägt die Inventarnummer ein. Die andere Variante ist, man klickt auf das Feld Scannen. Hierbei öffnet sich eine Barcodescanner-App, sofern diese vorhanden ist. Sollte sie nicht vorhanden sein, so schlägt er eine Barcodescanner-App vor und man kommt zum Play-Store. Ist die Inventarnummer eingetragen, so kann man nun eine Bemerkung hinzufügen, diese muss nicht befüllt sein. Ist das Gerät erfolgreich erfasst wurden, drückt man den Button „Gerät Hinzufügen“ . Dabei ist nun eine „TextView“ sichtbar, die anzeigt, wie viele Geräte man erfasst hat. Hat man alle Geräte erfolgreich erfasst, klickt man auf den Button „Raumerfassung abschließen“. Nun ist man wieder im Hauptmenü. Screenshot: siehe A..4.3.3

**Erfasse Räume:**

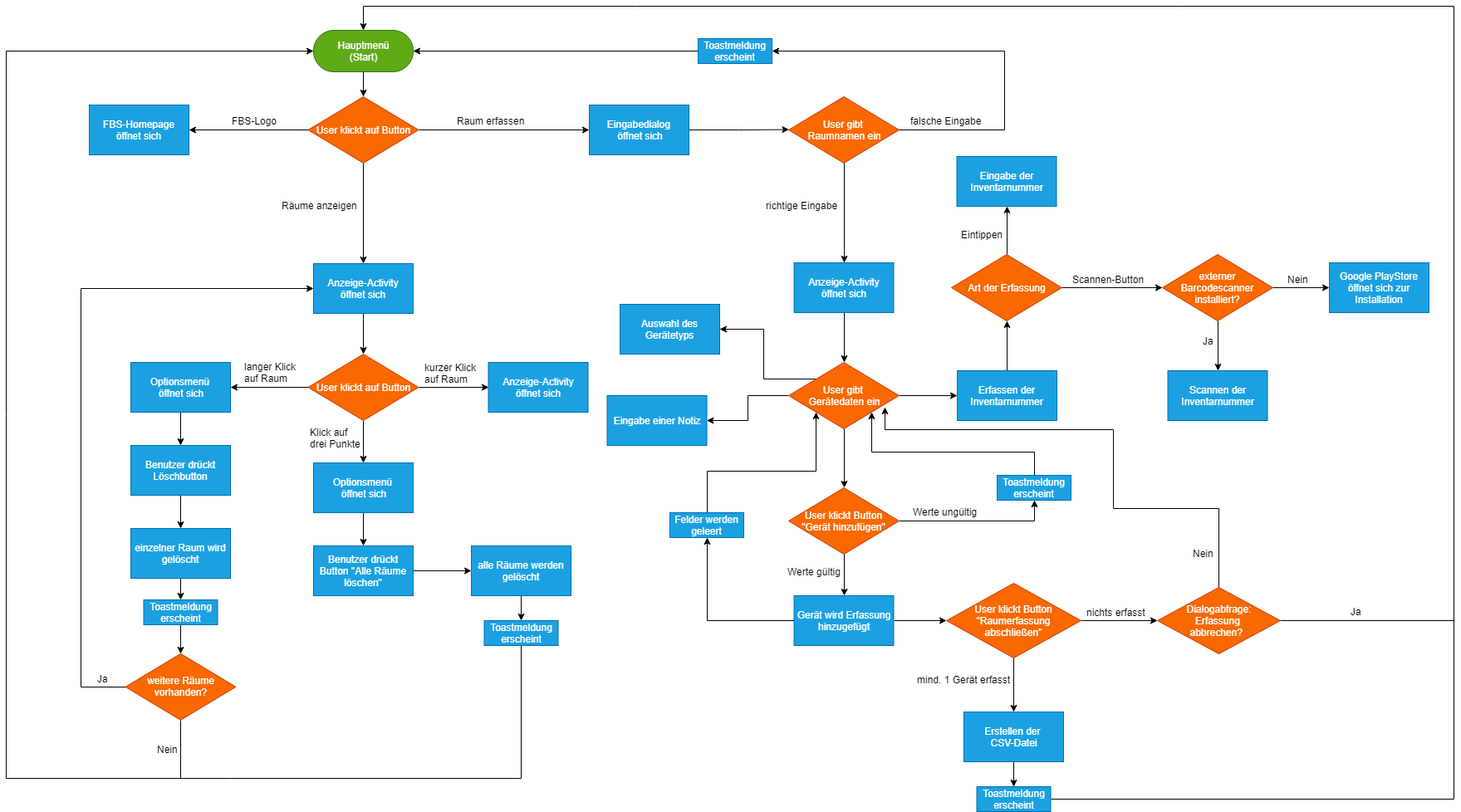
Hier wird eine Übersicht dargestellt, welche räume man erfasst hat. Die Räume werden sortiert nach Nummer und Alphabet angezeigt. Über die drei Punkte in der rechten oberen Ecke kann man alle erfassten Räume löschen. Hierbei kommt eine Meldung, ob man wirklich alle Räume löschen will. Bestätigt man das mit „Ja“ werden alle Räume gelöscht. Hält man länger gedrückt auf den Raumnamen, kommt ein kleines Pop-Up mit der Funktion löschen, klickt man hier auf „Löschen“ wird der Raum gelöscht. Hierbei erscheint keine Meldung. Klickt man nur kurz auf den Raumnamen so öffnet sich die Maske „Zeige Raumdetails“. Screenshot: siehe A4.3.4/A4.3.5

**Raumdetails:**In den Raumdetails findet sich eine Tabelle wieder. Die Überschriften sind „Gerätetyp“, „Inventarnummer“ und „Notiz“. Darunter werden die erfassten Geräte angezeigt. Bei dem Feld „Notiz“ wird nicht die komplette Bemerkung angezeigt. Die Raumdetails sollen nur einen kurzen Überblick geben über erfasste Geräte. Screenshot: siehe A..4.3.6

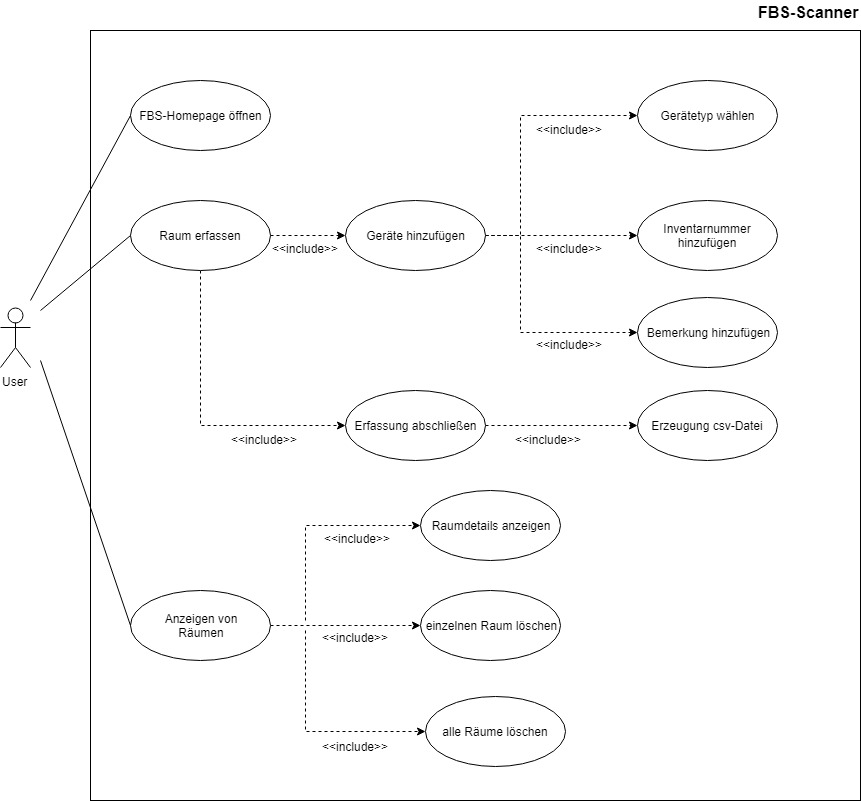
**Dateien als CSV-Datei exportieren:**Als erstes Verbindet man das Handy mit dem PC, danach muss man bei dem Handy „Dateien Übertragen auswählen. Nun ist man im Wurzelverzeichnis. Dort findet man einen Ordner Namens: „FBS“. Dort sind alle CSV-Dateien. Diese können nun in das gewünschte Verzeichnis auf den PC verschoben werden. Screenshot: siehe A..4.3.7

## **A.2 Diagramme**

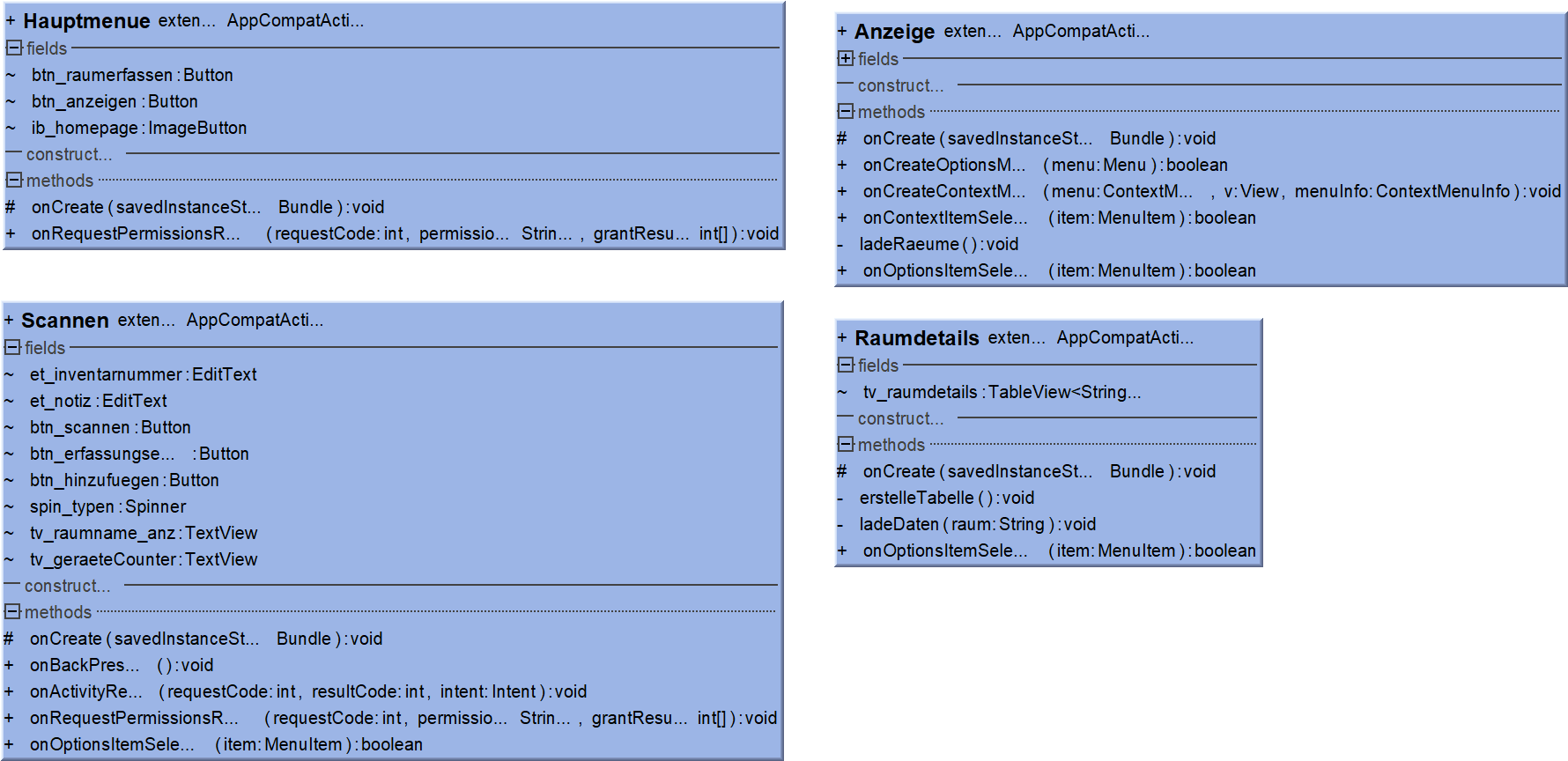
### **A.2.1 Programmablaufplan**

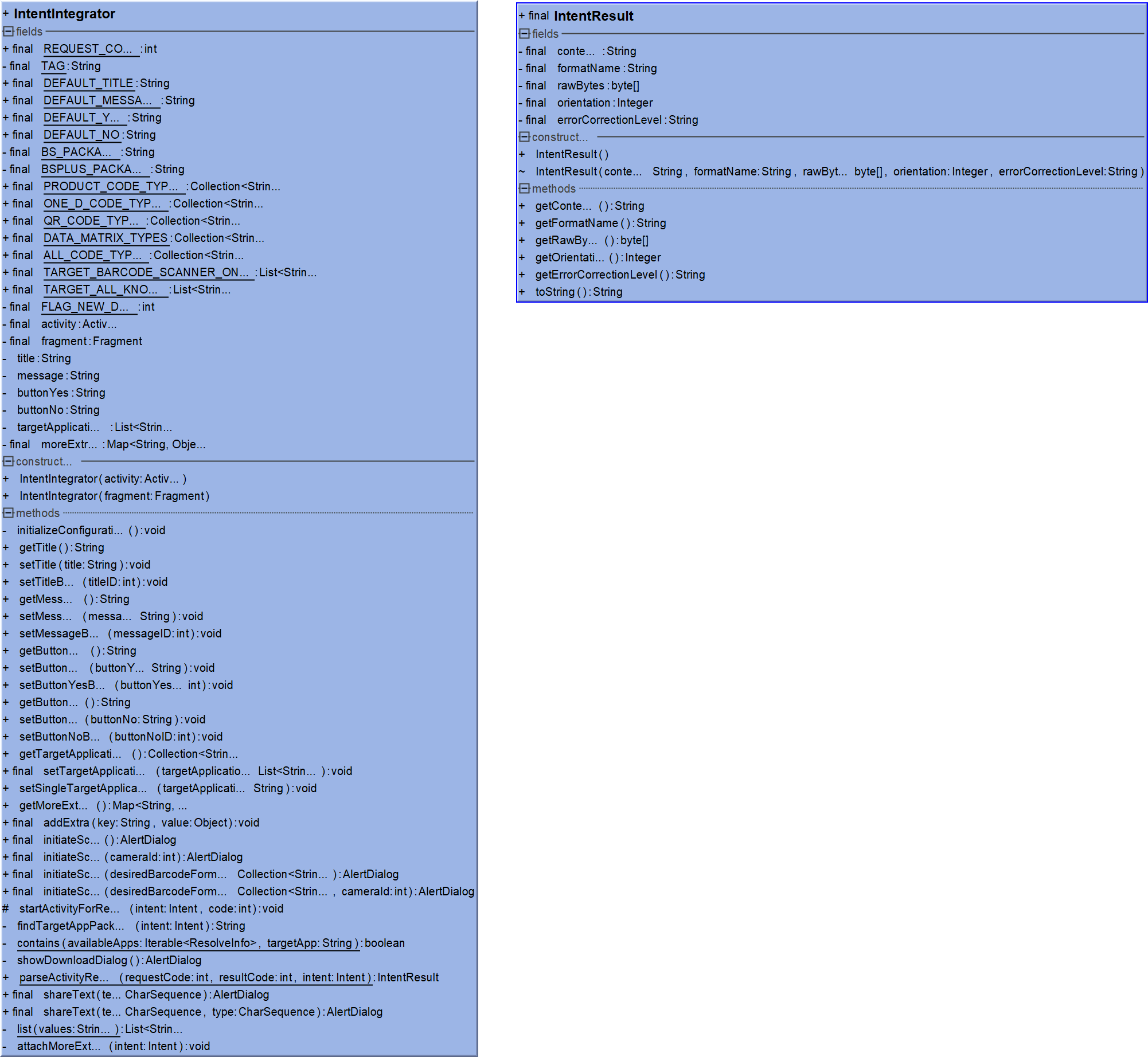


### **A.2.2 Use-Case-Diagramm**

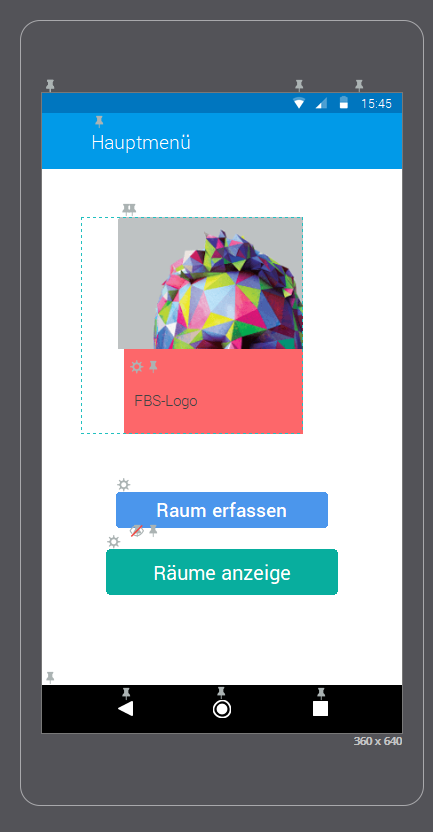


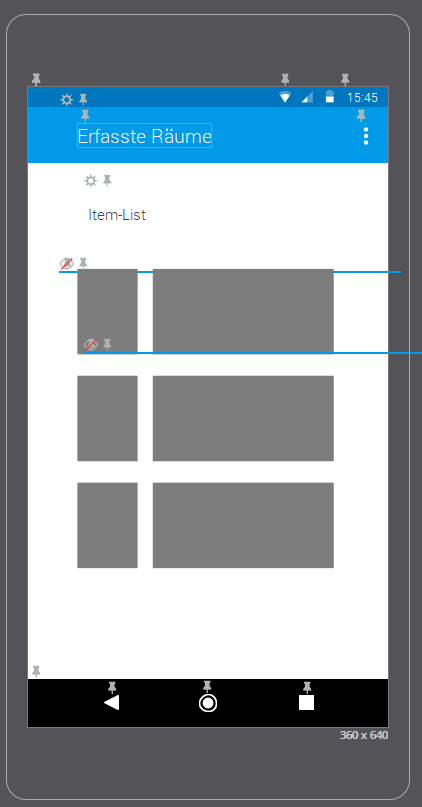
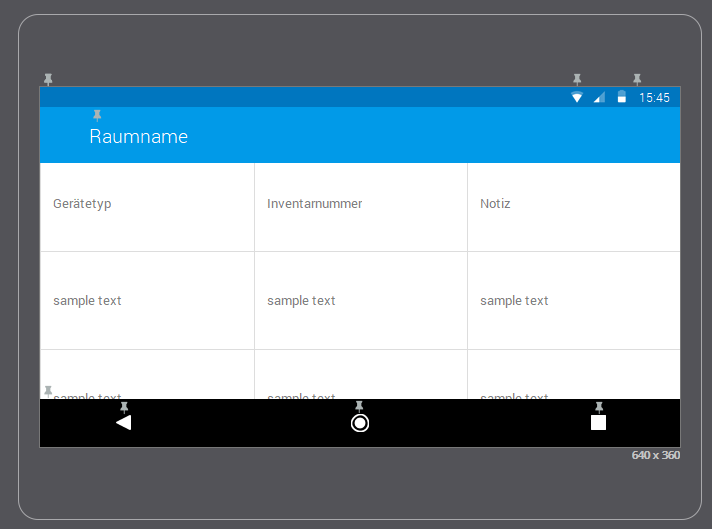
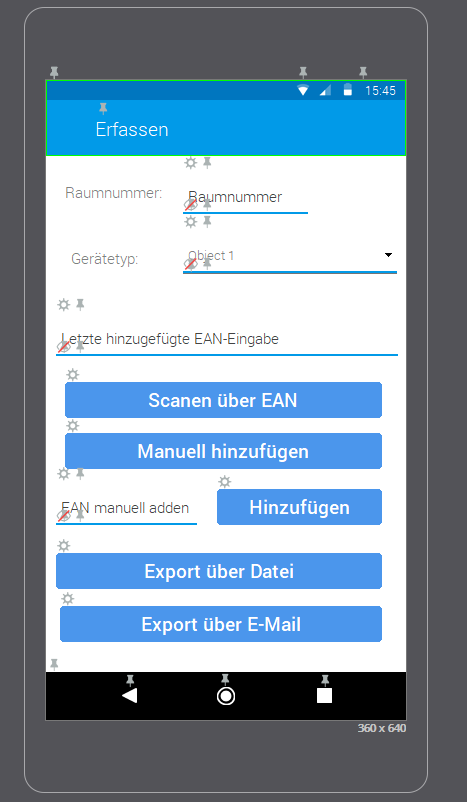
### **A.2.3 Klassendiagramme**



****

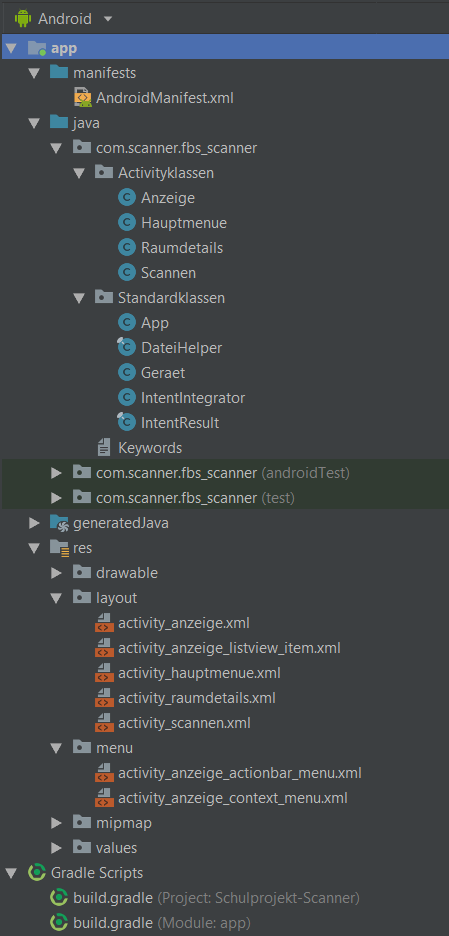
## **A.3 Oberflächenentwürfe**

****



## **A.4 Screenshots**

### **A.4.1 Projekt-Explorer**

****

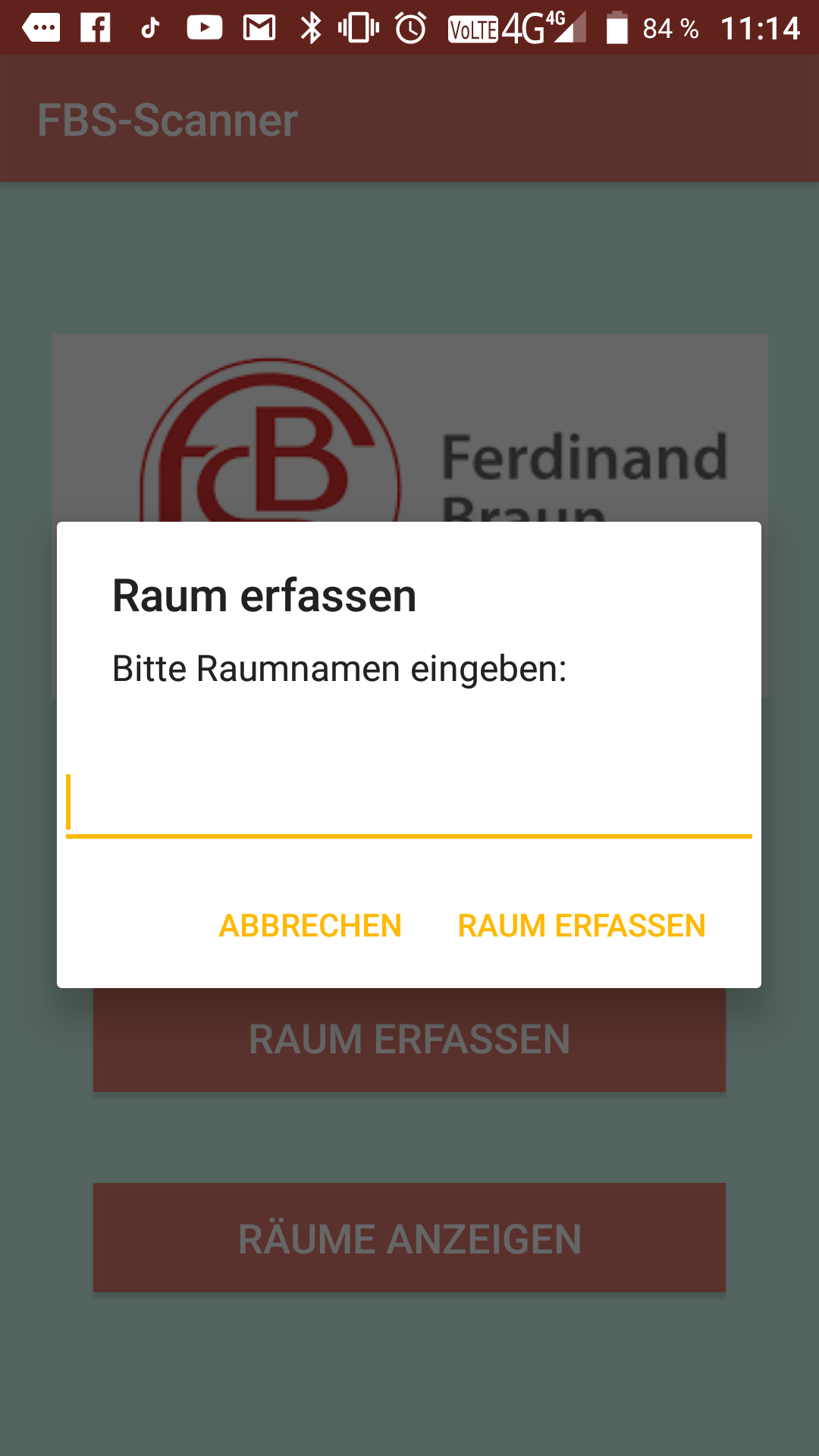
### **A.4.2 Code-Ausschnitte**

### **A.4.3 Fertige App**

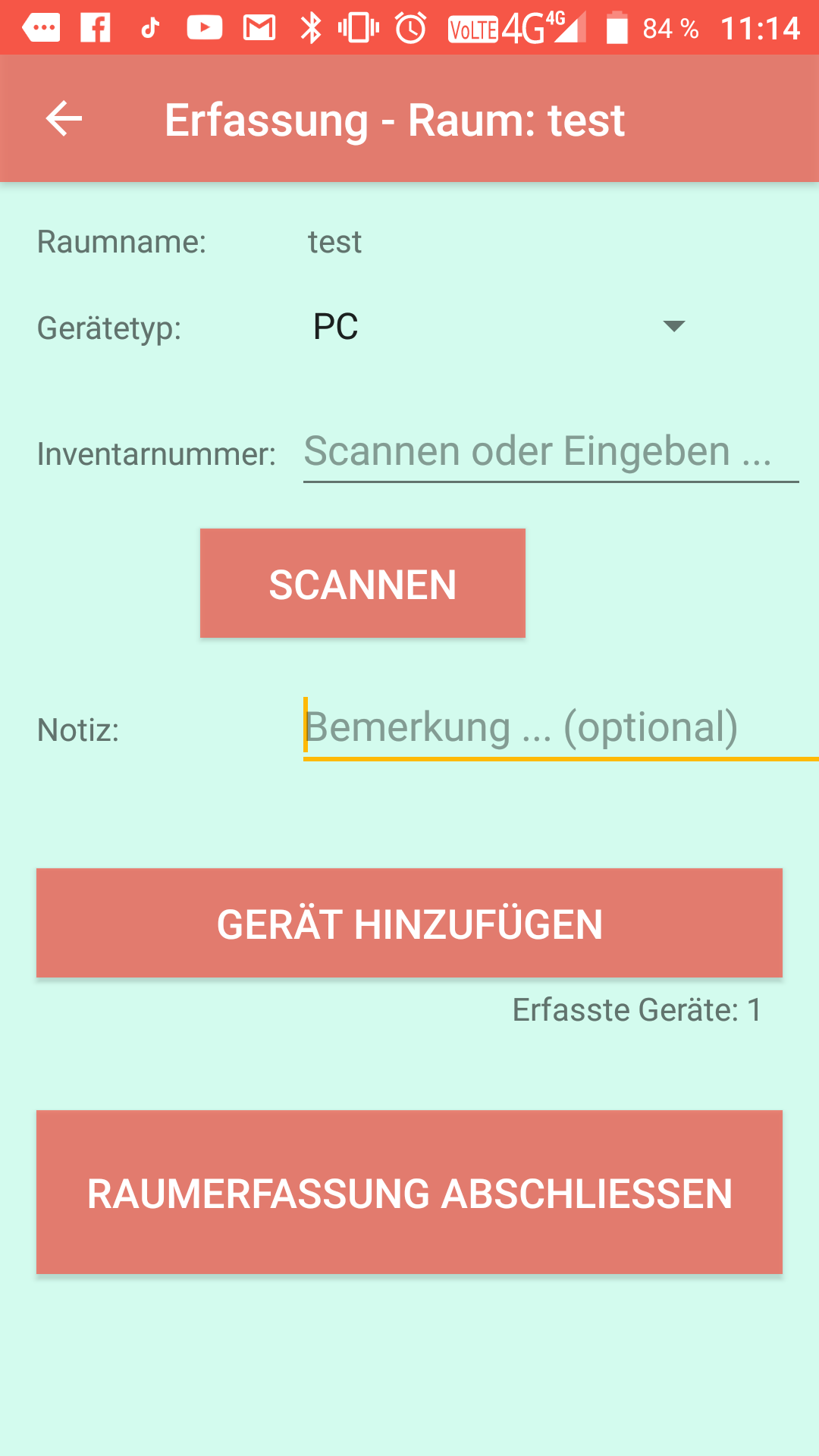
**Hauptmenü**

****

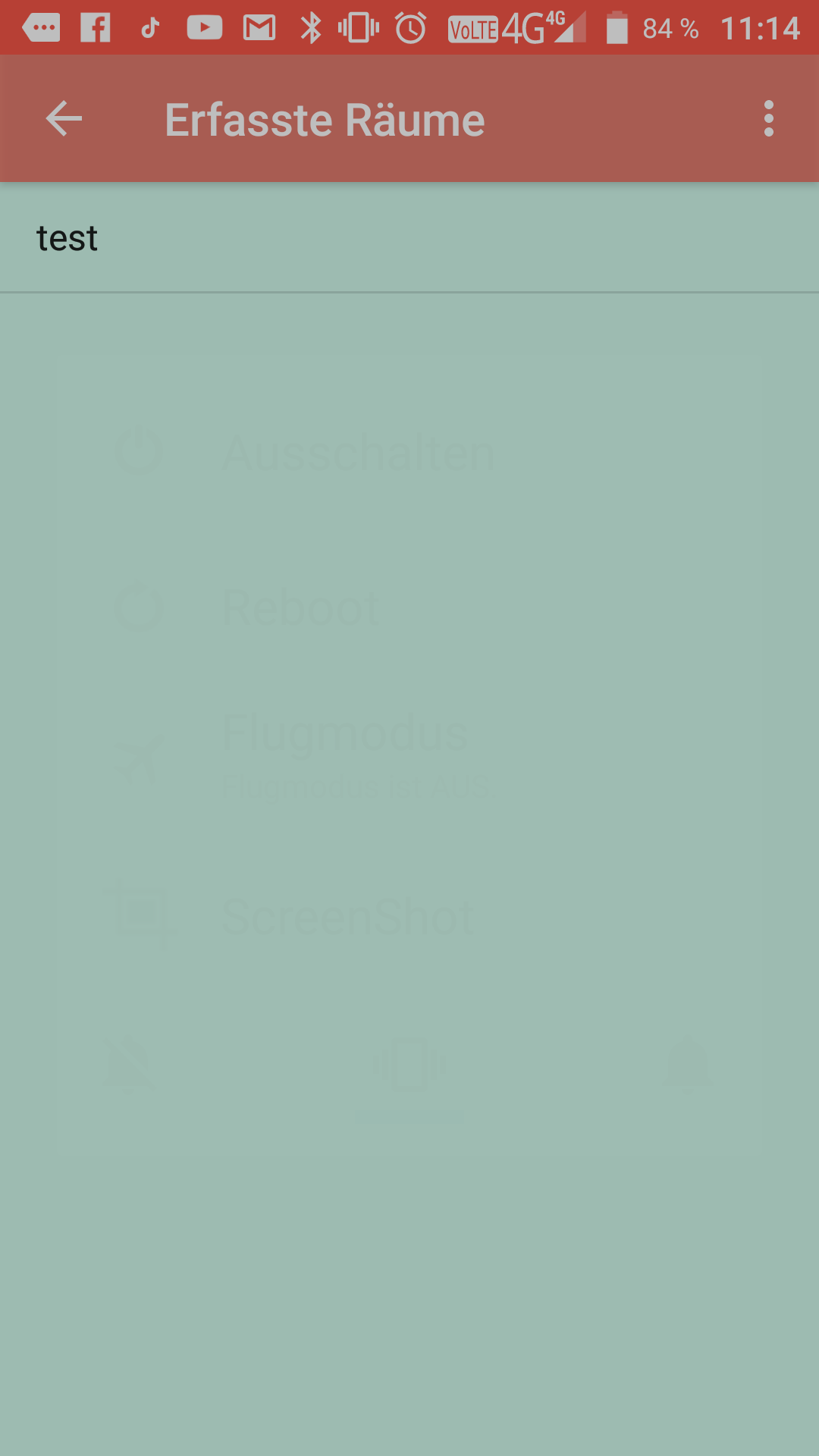
**Hauptmenü mit Raumname eingeben**

****

**Raumerfassung**

****

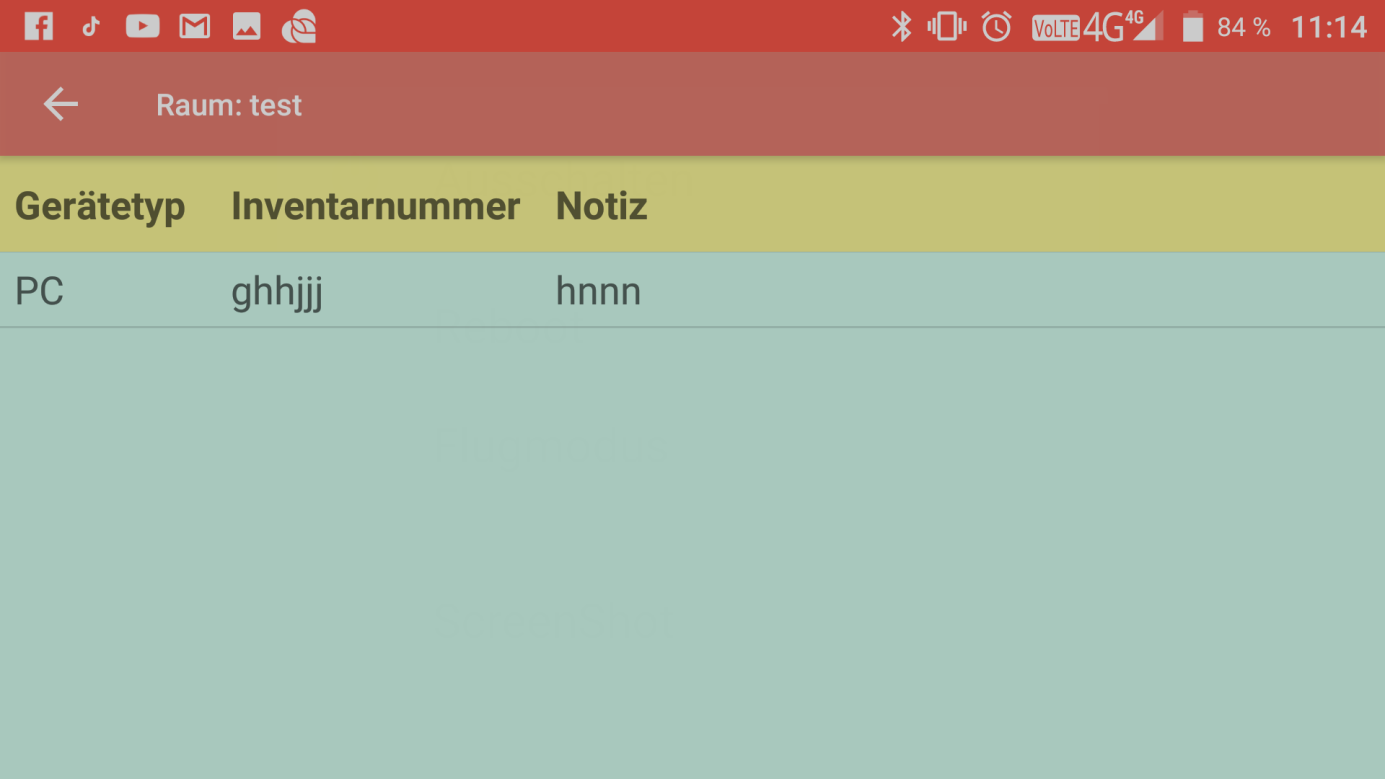
**Erfasste Räume**

****

**Erfasste Räume alle Löschen**

****

**Raumdetails**

****