

FBS-Scanner

Android-App zur Erfassung von Schulhardware

Abgabetermin: 18.01.2019

von Christian Wende und Sebastian Schulz

Inhaltsverzeichnis

1. Projektbeschreibung	1
1.1 Projektziel	1
1.2 Projektbegründung	1
1.3 Projektschnittstellen	1
2. Projektplanung	1
2.1 Zeitplanung	1
2.2 Ressourcenplanung	2
2.3 Entwicklungsprozess	2
3. Analysephase	3
3.1 Ist-Analyse	3
3.2 Soll-Zustand	3
3.3 Kostenkalkulation	4
4. Entwurfsphase	4
4.1 Auswahl der Programmiersprache	4
4.2 Ablauflogik	4
4.3 Entwurf der Benutzeroberfläche	5
5. Implementierungsphase	5
5.1 Implementierung der Benutzeroberfläche	5
5.1.1 Allgemeines	5
5.1.2 Hauptmenü	5
5.1.3 Scannen	6
5.1.4 Anzeige Räume	6
5.1.5 Anzeige Raumdetails	6
5.2 Implementierung des Backends	7
5.3 Qualitätskontrolle und Tests	8
5.3.1 Beschreibung der Tests	8
5.3.2 Manuelle Tests	8
6. Abnahme und Einführung	8
6.1 Abnahme	8
6.2 Installation	8

7. Dokumentation	9
7.1 Entwicklerdokumentation	9
7.2 Benutzerdokumentation	9
8. Projektbewertung.....	9
8.1 Soll-/Ist-Vergleich.....	9
8.2 Ausblick	10
8.3 Fazit.....	10
A Anhang.....	11
A.1 Benutzerdokumentation	11
A.2 Diagramme	12
A.2.1 Programmablaufplan.....	12
A.2.2 Use-Case-Diagramm	13
A.2.3 Klassendiagramme	14
A.3 Oberflächenentwürfe	16
A.4 Screenshots.....	18
A.4.1 Projekt-Explorer	18
A.4.2 Code-Ausschnitte	19
A.4.3 Fertige App	24

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. Es kann jede App verwendet werden, wenn diese auf der "Google-Standard-Bibliothek" basiert.

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 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](#) 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](#) 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 Activities 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 linken 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 befindet sich ganz oben eine TextView, welche den Raumnamen zeigt, der im Dialog eingetragen wurde. Darunter ist ein Spinner-Steuerelement. Klickt man auf dieses, so erscheint ein Dropdown-Menü und man kann einen der aufgeführten Gerätetypen auswählen. Der standardmäßig eingestellte Gerätetyp ist PC.

Als nächstes ist ein TextEdit für die Inventarnummer zu sehen. An dieser Stelle hat der Benutzer zwei Möglichkeiten. Er kann entweder das Feld manuell befüllen, oder er benutzt den Button „Scannen“. Dieser öffnet folglich einen Barcodescanner oder fragt, ob einer installiert werden soll, sofern noch keiner vorhanden ist. Ist die Inventarnummer erfolgreich eingetragen, so kann man in einem weiteren Feld eine Bemerkung bzw. Notiz eintragen. Klickt man nun auf den Button „Gerät hinzufügen“, so wird das Gerät hinzugefügt und die EditTexts für Notiz und Inventarnummer werden geleert. Die Anzahl der erfassten Geräte inkrementiert sich. Sie wird unter dem Button angezeigt. Ein Gerät kann nur hinzugefügt werden, wenn mindestens die Inventarnummer befüllt wurde. Ist dies jedoch nicht der Fall, erscheint eine Toast-Meldung. Der untere Button „Raumerfassung abschließen“ beendet die Erfassung und erzeugt eine CSV-Datei. Man ist danach wieder im Hauptmenü. Drückt man die Zurück-Taste, erscheint eine Meldung, die fragt, ob man die Erfassung wirklich abbrechen will. Sie wird aber nur angezeigt, wenn mindestens ein Gerät erfasst wurde.

5.1.4 Anzeige Räume

Hier befindet sich ein ListView, die die bisher erfassten Räume anzeigt. Rechts oben befinden sich 3 Punkte. Drückt man auf diese, so öffnet sich ein Kontextmenü und man kann „alle Räume löschen“ auswählen. Macht man dies, wird in einem Dialog abgefragt, ob man wirklich alle Daten löschen will. Weiterhin kann man auch einzelne Räume löschen. Dies funktioniert, indem man länger auf einen bestimmten Raum drückt. Es öffnet sich dann wieder ein kleines Menü, bei welchem man die Option „löschen“ auswählen kann. Bei kurzem Anklicken eines Raumes öffnet sich die Activity „Raumdetails“. Sollten sich Daten während der Anzeige von Räumen ändern, so können diese bequem über das „Swipe-to-Fresh“-Layout aktualisiert werden.

5.1.5 Anzeige Raumdetails

In den Raumdetails sieht man eine Tabelle, die die erfassten Geräte des zuvor selektierten Raumes anzeigt. Hier kann man ebenfalls über die „Swipe-to-Fresh“-Funktion die Daten aktualisieren. Über den Zurückbutton kommt man in die vorherige Activity.

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

5.2 Implementierung des Backends

Beim Starten der App wird die Hauptmenü-Activity angezeigt. Zu Beginn wird geprüft, ob der FBS-Order im Rootverzeichnis existiert. Ist dies nicht der Fall, wird dieser angelegt. Damit das Erstellen dieses Ordners überhaupt möglich ist, müssen allerdings zuerst Lese- und Schreibberechtigungen eingefordert werden, denen der Benutzer zustimmen muss.

Im Ordner „FBS“ werden die erstellten csv-Dateien abgelegt. Möchte man einen Raum erfassen, so wird der eingegebene Name mithilfe eines Intents zwischengespeichert und kann folglich in der Scannen-Activity wiederverwendet werden.

Drückt man hier auf den Button Scannen, wird eine Instanz der Klasse „IntentIntegrator“ erzeugt. Mithilfe der Methode „initiateScan()“ erfolgt der Scanvorgang und das Feld für die Inventarnummer wird mit dem Scanergebnis befüllt.

Möchte der Benutzer über den Button ein Gerät hinzufügen, wird zunächst überprüft, ob die Pflichtfelder gefüllt sind. Ist dies der Fall, so wird eine Instanz der Klasse „Geraet“ erzeugt und die gefüllten Felder werden dem Konstruktor dieser übergeben. Das Gerät wird anschließend einer Liste hinzugefügt und der Counter zu Anzeige der bisher erfassten Geräte inkrementiert. Daraufhin werden die Felder Inventarnummer und Notiz geleert.

Entscheidet sich der User, die Erfassung abzuschließen, werden erneut Lese- und Schreibberechtigungen angefordert, sofern diese nicht schon vorhanden sind. Anschließend wird eine csv-Datei erstellt und mit den bisher erfassten Daten beschrieben. Dies geschieht, indem über die Geräteliste iteriert wird. Anschließend werden die Elemente der Liste gelöscht und man befindet sich wieder im Hauptmenü.

Möchte sich der Benutzer die bisher erfassten Räume anzeigen lassen, so kann er dies über den Button „Räume anzeigen“ im Hauptmenü machen. Dieser öffnet die Anzeige-Activity, sofern mindestens ein Raum erfasst wurde. Ist dies nicht der Fall, wird eine Meldung angezeigt. Die angezeigten Räume werden hierbei alphanumerisch sortiert. Wählt man in dieser die Funktion „löschen“ für einen Raum aus, wird der ausgewählte Raum im FBS-Verzeichnis gesucht und entfernt. Wenn der Benutzer alle Räume löscht, geschieht dies in ähnlicher Weise mit allen Dateien im Ordner. Bei kurzem Anklicken eines Raumes öffnet sich die Raumdetails-Activity und es wird eine Tabelle erstellt. Die angezeigten Daten werden aus der csv-Datei ausgelesen.

Der Großteil der Operationen bzw. Logik ist in der Klasse „DateiHelper“ implementiert worden. Sie ist unter diversen anderen Codeausschnitten im Anhang [A.4.2](#) zu finden.

5.3 Qualitätskontrolle und Tests

5.3.1 Beschreibung der Tests

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

5.3.2 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 man hat mit Beispieldaten verschiedene Szenarios simuliert, um zu schauen, ob alles reibungslos funktioniert.

6. Abnahme und Einführung

6.1 Abnahme

Die Abnahme ist reibungslos erfolgt. Nach der Installation der App auf dem Handy des Kunden konnte dieser sich sofort mit den Funktionen vertraut machen. Es waren keine Mängel zu verzeichnen. Der Kunde war zudem darüber erfreut, dass auch die weniger wichtigen Funktionen implementiert wurden.

6.2 Installation

Die Installation erfolgt über folgenden 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 dann problemlos installiert werden können. Fehlen Berechtigungen, so muss man in die Einstellungen gehen und folgende Option aktivieren:

Sicherheit > Geräteverwaltung > Unbekannte Herkunft zulassen. Danach sollte sich die APK-Datei ohne Probleme installieren lassen.

7. Dokumentation

7.1 Entwicklerdokumentation

Die wesentlichen Dinge des Codes gehen aus den hinzugefügten Kommentaren hervor (siehe Anhang A.4.2). Zudem sind im Anhang A.2 folgende Diagramme enthalten:

- Use-Case-Diagramm
- Ablaufdiagramm
- Klassendiagramm

Mit diesen Diagrammen sollte es relativ einfach sein, die App zu verstehen und die Entwicklung fortzusetzen, sofern es weitere Anforderungen geben sollte.

7.2 Benutzerdokumentation

Da der Kunde eine ausführliche Benutzerdokumentation wünscht, wurde diese erstellt. Sie ist im Anhang A.1 zu finden.

8. Projektbewertung

8.1 Soll-/Ist-Vergleich

Bei rückblickendem Betrachten des Projektes kann festgestellt werden, dass alle 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.

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

8.2 Ausblick

Auch wenn es nicht Teil dieser Anforderung ist, könnte man die App noch um einige Funktionen erweitern. Man könnte z.B. implementieren, dass die Einträge in der Tabelle der Raumdetails-Activity bearbeitet werden können. Zudem könnte man in der Scannen-Activity die Möglichkeit bieten, dass der Benutzer weitere Gerätetypen hinzufügen kann. Dies sind aber alles nur Kleinigkeiten, die nicht wirklich einen Einfluss auf die Kernfunktionen der Anwendung haben.

8.3 Fazit

Bei der Umsetzung des Projekts konnten wir Einiges lernen und es war eine interessante Erfahrung. Zum einem hat man in Erfahrung gebracht, wie man ein schönes und übersichtliches Layout erstellt, zum anderen hat man erste Einblicke in die Android-Welt bekommen und konnte seine Java-Kenntnisse erweitern. Wir haben gelernt, wie man Dateien aus Android erzeugt, wie das Speichersystem funktioniert und welche Berechtigungen man benötigt.

A Anhang

A.1 Benutzerdokumentation

Hauptmenü

Öffnet der Benutzer die App, ist man im Hauptmenü. In diesem befinden sich drei Buttons. Der oberste Button (FBS-Logo) öffnet die FBS-Homepage (URL: <https://www.ferdinand-braun-schule.de/>). Der zweite Button hat die Beschriftung „Raum erfassen“. Betätigt man diesen, öffnet sich eine Dialogbox mit dem Text „Raum erfassen“. Klickt man in das Edittext, öffnet sich die Tastatur und man kann den Raumnamen eingeben. Der letzte Button ist der Button „Räume anzeigen“. Durch diesen wird eine weitere Activity geöffnet.

Erfassung – Raum

Trägt man erfolgreich den Raumnamen ein, so kommt man zur Erfassungsmaske „Erfassung Raum“. Als erstes sieht man den Raum, darunter ist der Gerätetyp zu sehen. Klickt man den Spinner an, so öffnet sich dieser und man kann zwischen den Geräten „PC“, „Monitor“, „Drucker“ oder „Beamer“ auswählen. Standardmäßig ist der „PC“ ausgewählt. Hat man den Gerätetyp bestimmt, so kann man die Inventarnummer erfassen. Hierbei hat man zwei Möglichkeiten. Zum einen kann man das EditText-Feld anklicken und die Nummer händisch eingeben. Die andere Variante beinhaltet, dass man auf den Button „Scannen“ drückt. Hierbei öffnet sich eine Barcodescanner-App, sofern diese vorhanden ist. Sollte sie nicht vorhanden sein, so wird eine Barcodescanner-App vorgeschlagen und man kommt zum Play-Store. Ist die Inventarnummer eingetragen, so kann man nun eine Bemerkung hinzufügen. Diese ist optional. Sind alle Pflichtfelder entsprechend gefüllt worden, drückt man den Button „Gerät hinzufügen“. Es erscheint folglich ein kleines Feld, in dem die Anzahl der bisher erfassten Geräte angezeigt wird. Hat man alle Geräte erfolgreich erfasst, klickt man auf den Button „Raumerfassung abschließen“. Nun ist man wieder im Hauptmenü.

Erfasse Räume

Hier wird eine Übersicht dargestellt, welche Räume bisher erfasst wurden. 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 diese mit „Ja“ werden alle Datensätze gelöscht. Drückt man für längere Zeit auf den Raumnamen, kommt ein kleines Pop-Up mit der Option „löschen“. Wählt man diese aus, wird der selektierte Raum gelöscht. Hierbei erscheint eine Meldung. Klickt man nur kurz auf den Raumnamen so öffnet sich die Maske „Raumdetails“.

Raumdetails

In den Raumdetails ist eine Tabelle zu sehen. Die Überschriften sind „Gerätetyp“, „Inventarnummer“ und „Notiz“. Darunter werden die erfassten Geräte für den jeweils ausgewählten Raum angezeigt. Bei dem Feld „Notiz“ wird nicht die komplette Bemerkung angezeigt.

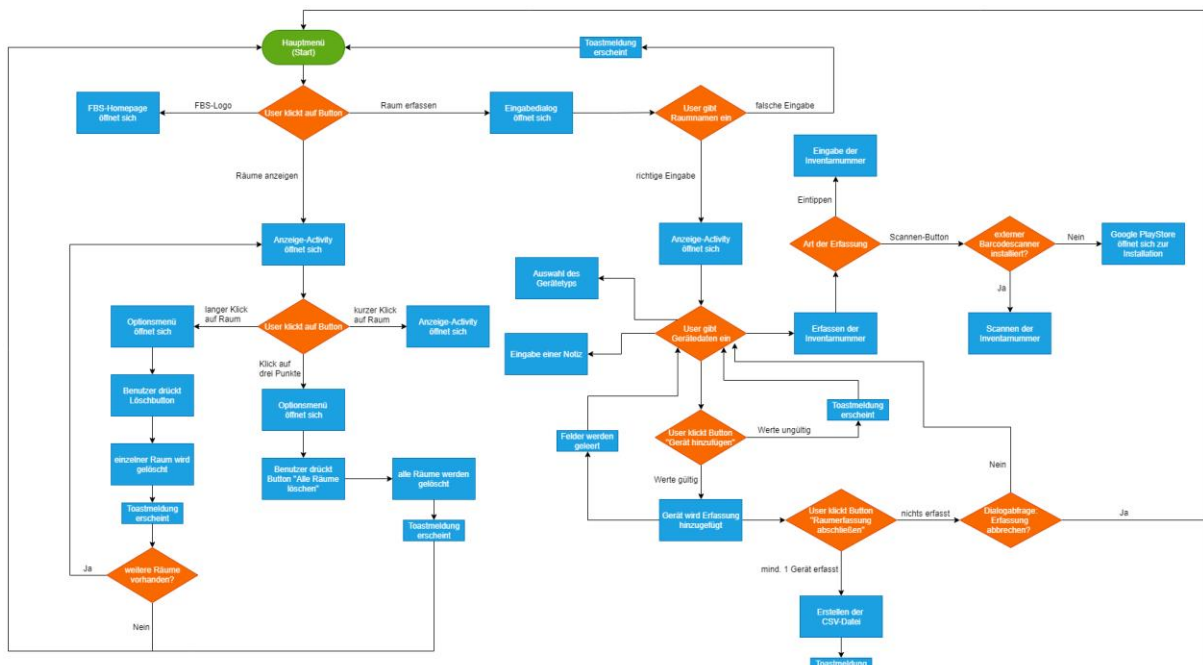
Dateien als CSV-Datei exportieren

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“, in welchem alle CSV-Dateien abgelegt werden. Diese können nun in das gewünschte Verzeichnis auf den PC verschoben werden.

Die Screenshots zur Benutzerdokumentation sind in Anhang [A.4.3](#) zu finden.

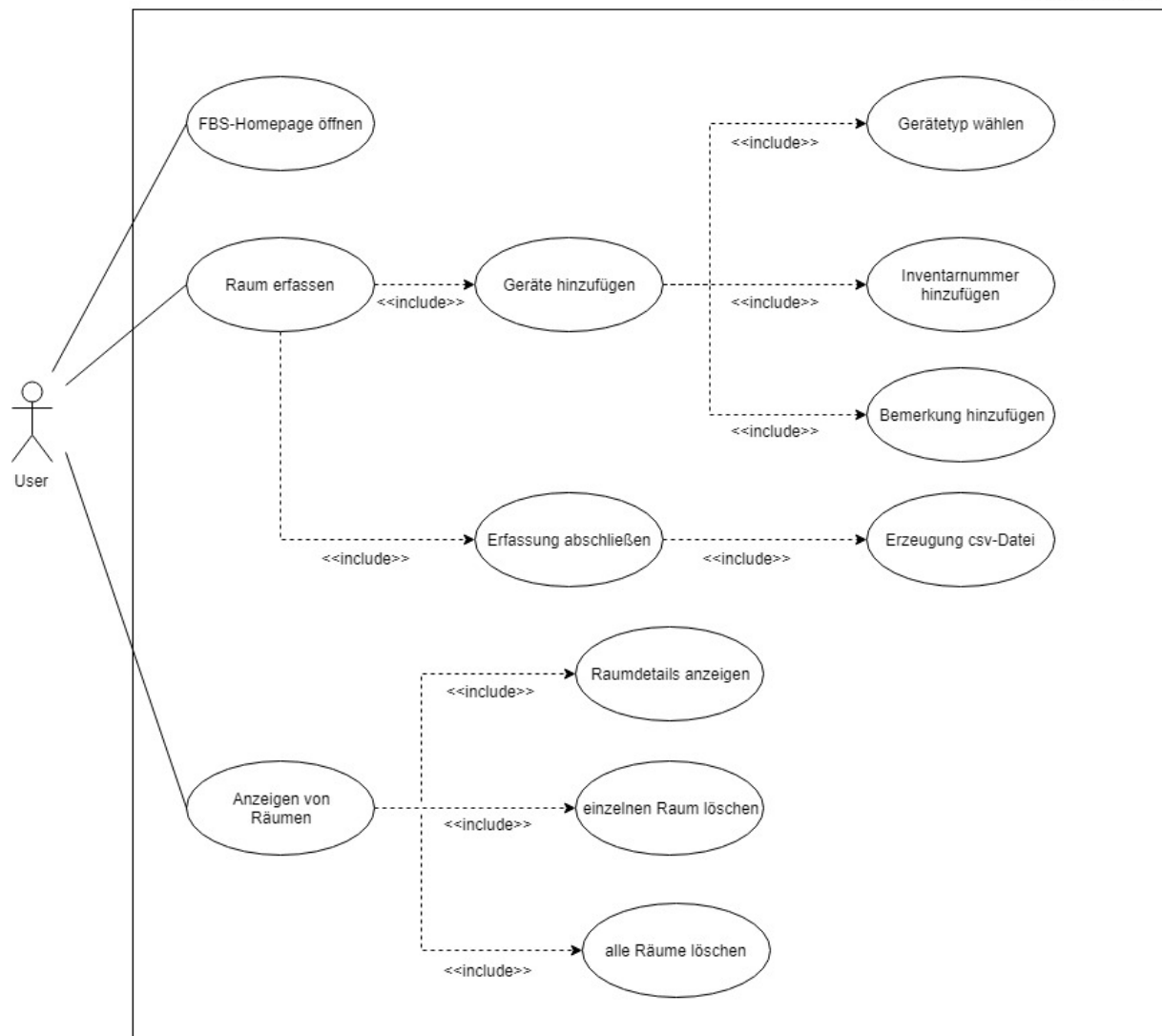
A.2 Diagramme

A.2.1 Programmablaufplan



A.2.2 Use-Case-Diagramm

FBS-Scanner



A.2.3 Klassendiagramme

```
+ Hauptmenue exten... AppCompatActivity...
fields
- btn_raumerfassen : Button
- btn_anzeigen : Button
- ib_homepage : ImageButton
construct...
methods
# onCreate (savedInstanceState... Bundle):void
+ onRequestPermissionsResult... (requestCode:int, permisso... Strin... , grantResu... int[]):void
```

```
+ Scannen exten... AppCompatActivity...
fields
- et_inventarnummer : EditText
- et_notiz : EditText
- btn_scannen : Button
- btn_erfassungse... : Button
- btn_hinzufuegen : Button
- spin_typen : Spinner
- tv_raumname_anz : TextView
- tv_geraeteCounter : TextView
construct...
methods
# onCreate (savedInstanceState... Bundle):void
+ onBackPressed... ():void
+ onActivityResult... (requestCode:int, resultCode:int, intent:Intent):void
+ onRequestPermissionsResult... (requestCode:int, permisso... Strin... , grantResu... int[]):void
+ onOptionsItemSelected... (item:MenuItem):boolean
```

```
+ Anzeige exten... AppCompatActivity...
fields
construct...
methods
# onCreate (savedInstanceState... Bundle):void
+ onCreateOptionsMenu... (menu:Menu):boolean
+ onCreateContextMenu... (menu:ContextMenu... , v:View, menuInfo:ContextMenuInfo):void
+ onContextItemSelected... (item:MenuItem):boolean
- ladeRaume():void
+ onOptionsItemSelected... (item:MenuItem):boolean
```

```
+ Raumdetails exten... AppCompatActivity...
fields
- tv_raumdetails : TableView<String...
construct...
methods
# onCreate (savedInstanceState... Bundle):void
- erstelleTabelle():void
- ladeDaten(raum: String):void
+ onOptionsItemSelected... (item:MenuItem):boolean
```

```
+ Geraet
fields
- raumName : String
- geraeteTyp : String
- inventarnummer : String
- notiz : String
+ geraeteli... : ArrayList<Gera...
construct...
+ Geraet(raumName:String, geraeteTyp:String, inventarnummer:String, not... String)
methods
+ gibDateiFormat():String
+ getRaumName():String
+ getGeraeteTyp():String
+ getInventarnummer():String
+ getNotiz():String
+ setRaumNr(raumName:String):void
+ setGeraeteTyp(geraeteTyp:String):void
+ setInventarnummer(inventarnummer:String):void
+ setNotiz(not... String):void
```

```
+ App exten... Application...
fields
- mCont... : Context
construct...
methods
+ onCreate():void
+ getContext():Context
```

```
+ final DateiHelper
fields
+ activityPlacehol... : Activ...
- externerSpeicherLe... : boolean
- externerSpeicherBeschrei... : boolean
+ final schreibPermis... : String
+ final lesePermiss... : String
+ final STORAGE_REQUEST_CO... : int
- rootVerzeich... : File
- csvVerzeich... : File
construct...
methods
- pruefeSpeicher():void
+ fordereLeseUndSchreibPermissio... (nurVezeich... boolean):void
+ erzeugeUndBeschreibeD... ():void
+ erstelleVerzeic... ():void
+ leseDatei... (raumname:String):ArrayList<Gera...
+ gibRaumListe():ArrayList<Strin...
+ loescheD... (raumname:String, activity:Activ...):void
+ loescheAlleDate... (activity:Activ...):void
+ sortiereLi... (zuSortierendeLi... ArrayList<Strin...):ArrayList<Strin...
```

```

+ IntentIntegrator
  fields
+ final REQUEST_CODE... :int
- final TAG :String
+ final DEFAULT_TITLE :String
+ final DEFAULT_MESSA... :String
+ final DEFAULT_Y... :String
+ final DEFAULT_NO :String
- final BS_PACKA... :String
- final BSPLUS_PACKA... :String
+ final PRODUCT_CODE_TYP... :Collection<Strin...
+ final ONE_D_CODE_TYP... :Collection<Strin...
+ final QR_CODE_TYP... :Collection<Strin...
+ final DATA_MATRIX_TYPES :Collection<Strin...
+ final ALL_CODE_TYP... :Collection<Strin...
+ final TARGET_BARCODE_SCANNER_ON... :List<Strin...
+ final TARGET_ALL_KNO... :List<Strin...
- final FLAG_NEW_D... :int
- final activity :Activ...
- final fragment :Fragment
- title :String
- message :String
- buttonYes :String
- buttonNo :String
- targetApplicati... :List<Strin...
- final moreExtr... :Map<String, Obje...
  construct...
+ IntentIntegrator( activity :Activ... )
+ IntentIntegrator( fragment :Fragment )
  methods
- initializeConfigurati... () :void
+ getTitle () :String
+ setTitle (title :String ) :void
+ setTitleB... (titleID :int) :void
+ getMess... () :String
+ setMess... (messa... String) :void
+ setMessageB... (messageID :int) :void
+ getButton... () :String
+ setButton... (buttonY... String) :void
+ setButtonYesB... (buttonYes... int) :void
+ getButton... () :String
+ setButton... (buttonNo :String) :void
+ setButtonNoB... (buttonNoID :int) :void
+ getTargetApplicati... () :Collection<Strin...
+ final setTargetApplicati... (targetApplicatio... List<Strin... ) :void
+ setSingleTargetApplica... (targetApplicati... String ) :void
+ getMoreExt... () :Map<String, ...
+ final addExtra (key :String , value :Object) :void
+ final initiateSc... () :AlertDialog
+ final initiateSc... (camerald :int) :AlertDialog
+ final initiateSc... (desiredBarcodeForm... Collection<Strin... ) :AlertDialog
+ final initiateSc... (desiredBarcodeForm... Collection<Strin... , camerald :int) :AlertDialog
# startActivityForRe... (intent :Intent , code :int) :void
- findTargetAppPack... (intent :Intent) :String
- contains (availableApps :Iterable<ResolveInfo> , targetApp :String ) :boolean
- showDownloadDialog () :AlertDialog
+ parseActivityRe... (requestCode :int , resultCode :int , intent :Intent) :IntentResult
+ final shareText (te... CharSequence) :AlertDialog
+ final shareText (te... CharSequence , type :CharSequence) :AlertDialog
- list (values :Strin... ) :List<Strin...
- attachMoreExt... (intent :Intent) :void

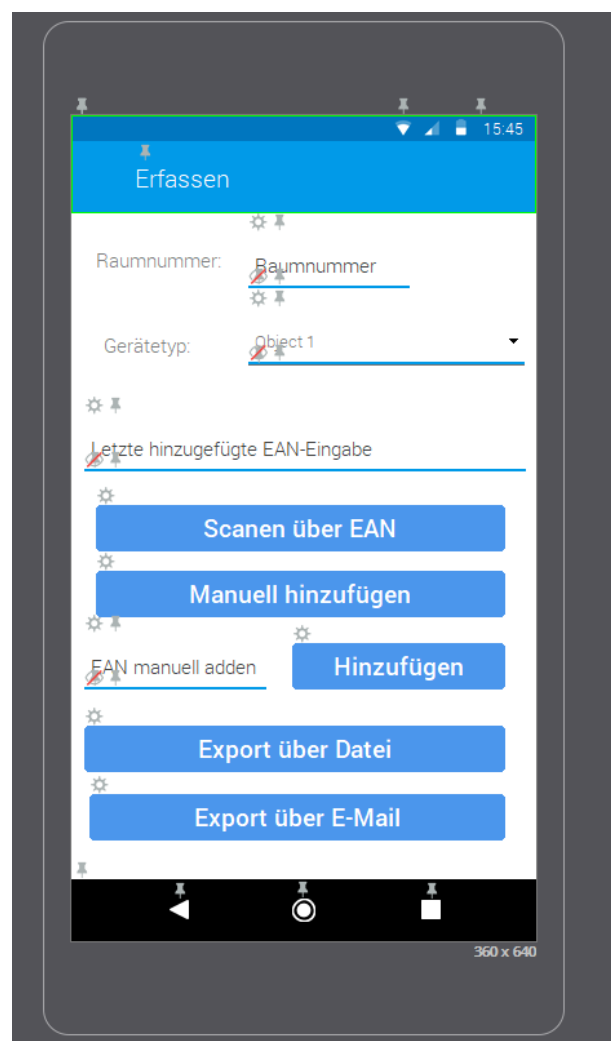
```

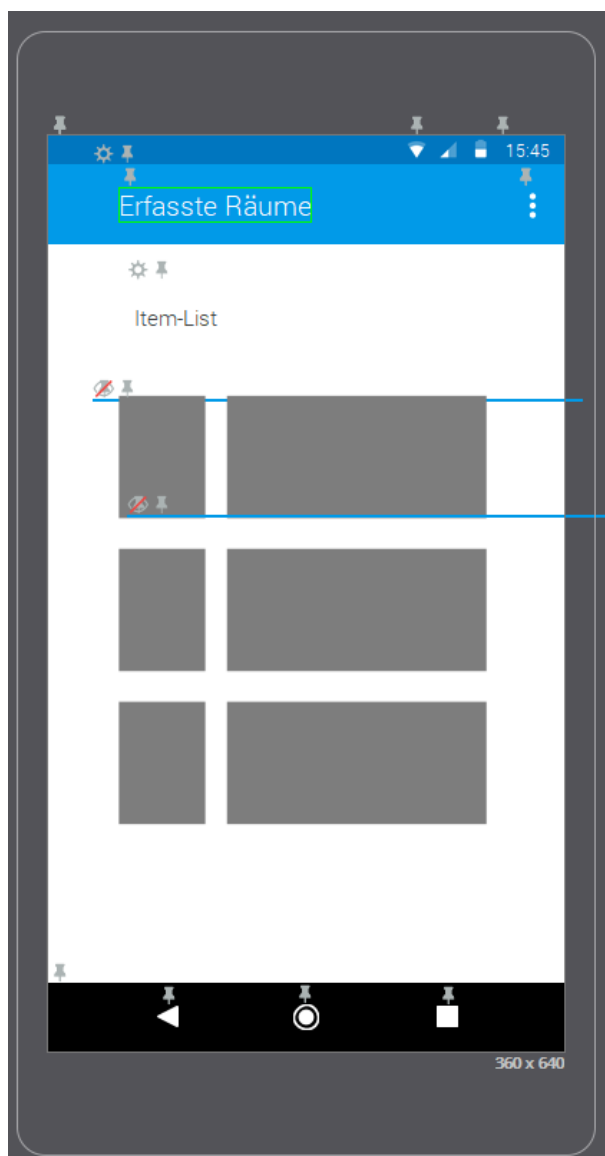
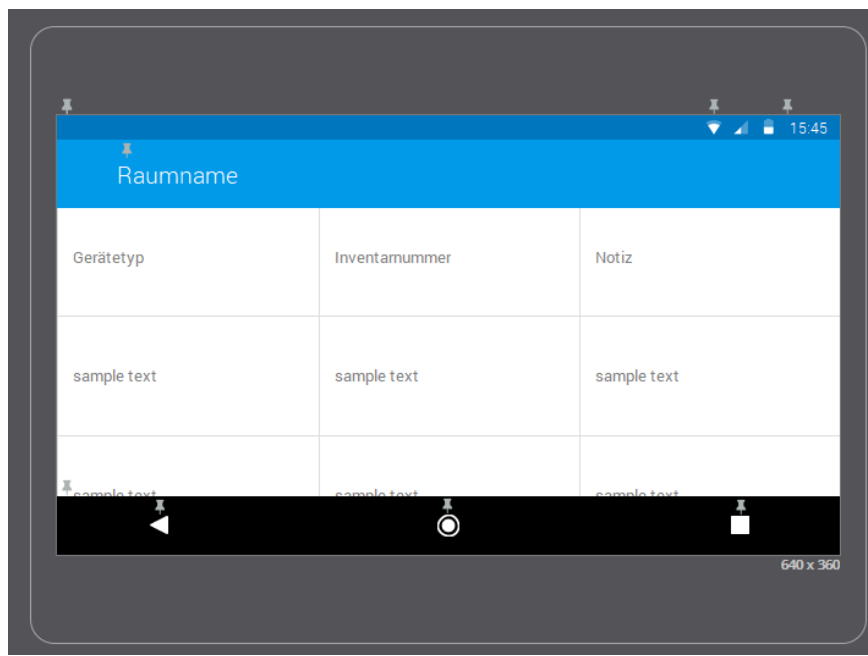
```

+ final IntentResult
  fields
- final conte... :String
- final formatName :String
- final rawBytes :byte[]
- final orientation :Integer
- final errorCorrectionLevel :String
  construct...
+ IntentResult ()
~ IntentResult (conte... String , formatName :String , rawByt... byte[] , orientation :Integer , errorCorrectionLevel :String )
  methods
+ getConte... () :String
+ getFormatName () :String
+ getRawBy... () :byte[]
+ getOrientati... () :Integer
+ getErrorCorrectionLevel () :String
+ toString () :String

```

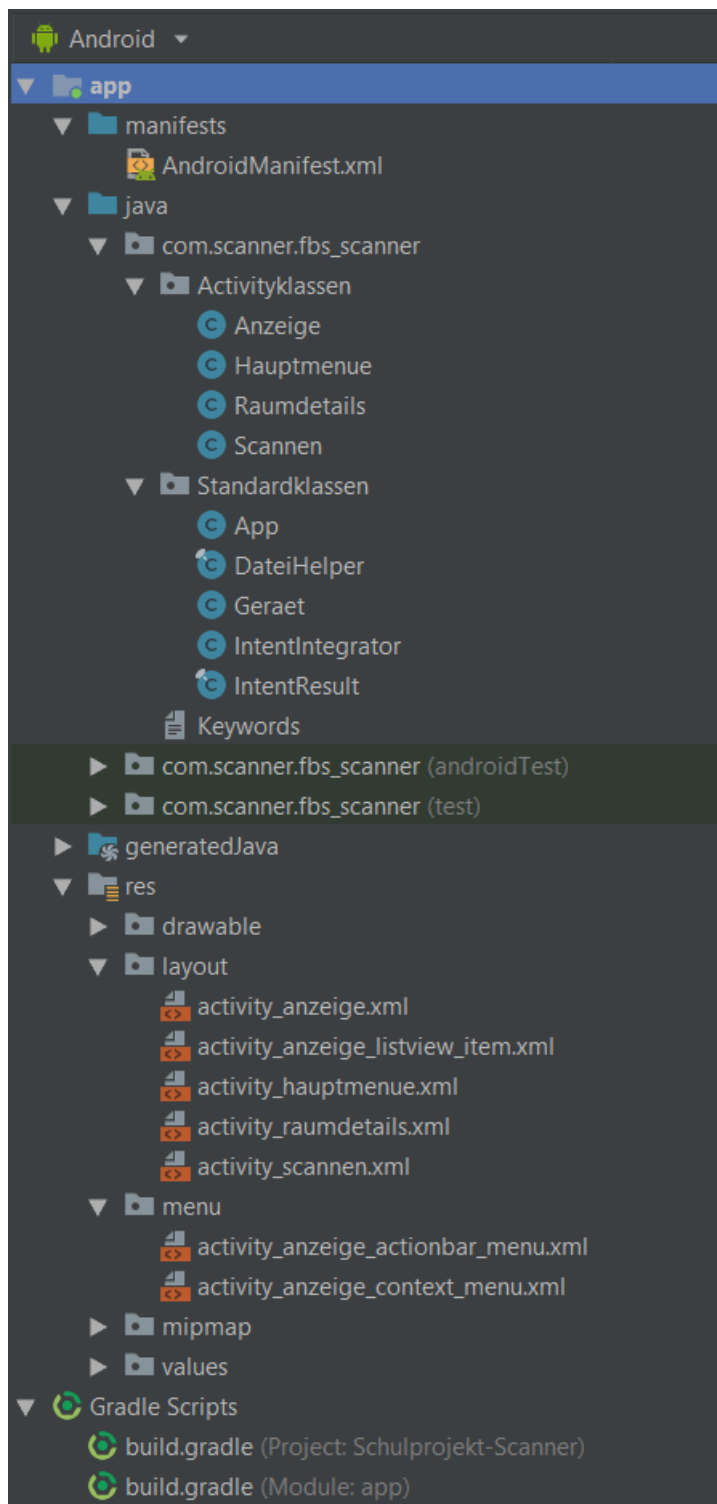
A.3 Oberflächenentwürfe





A.4 Screenshots

A.4.1 Projekt-Explorer



A.4.2 Code-Ausschnitte

Klasse Geraet

```
1 package com.scanner.fbs_scanner.Standardklassen;
2
3 import java.util.ArrayList;
4
5 // Jede Instanz dieser Klasse enthält die Informationen für ein Gerät
6 public class Geraet {
7
8     private String raumName;
9     private String geraeteTyp;
10    private String inventarnummer;
11    private String notiz;
12    public static ArrayList<Geraet> geraeteliste = new ArrayList<>();
13
14    public Geraet(String raumName, String geraeteTyp, String inventarnummer, String notiz)
15    {
16        this.raumName = raumName;
17        this.geraeteTyp = geraeteTyp;
18        this.inventarnummer = inventarnummer;
19        this.notiz = notiz;
20    }
21
22    public String gibDateiFormat(){
23        return this.raumName.trim().replace( target: ";", replacement: ",") + ";" + this.geraeteTyp +
24            ";" + this.inventarnummer.trim().replace( target: ";", replacement: ",") +
25            ";" + this.notiz.trim().replace( target: ";", replacement: ",") + "\n";
26    }
27
28    public String getRaumName() { return raumName; }
31
32    public String getGeraeteTyp() { return geraeteTyp; }
35
36    public String getInventarnummer() { return inventarnummer; }
39
40    public String getNotiz() { return notiz; }
42
```

Klasse App

```
1 package com.scanner.fbs_scanner.Standardklassen;
2
3 import ...
4
5
6
7 // Hilfsklasse, die es ermöglicht, auch innerhalb von Non-Activity-Klassen auf die Stringressourcen
8 // zuzugreifen mit: App.getContext().getResources().getString(R.string.XXXXXX)
9 public class App extends Application {
10
11     private static Context mContext;
12
13     @Override
14     public void onCreate() {
15         super.onCreate();
16         mContext = this;
17     }
18
19     public static Context getContext() { return mContext; }
22
23 }
```

Klasse Scannen (Activity) – Ausschnitt

```
57
58 // Übergabe des Raums
59 String raum = b.getString( key: "KEY_RAUM");
60 tv_raumname_anz.setText(raum);
61
62 // hier erfolgt der Scanvorgang über den Barcodescanner
63 // das Ergebnis wird in der Methode onActivityResult entgegengenommen
64 btn_scannen.setOnClickListener((v) -> {
65     IntentIntegrator integrator = new IntentIntegrator( activity: Scannen.this);
66     integrator.initiateScan();
67 });
68
69
70 // Gerät wird einer Liste hinzugefügt (noch nicht persistent)
71 // zuvor findet eine Validierung der Felder statt
72 btn_hinzufuegen.setOnClickListener((v) -> {
73     if (tv_raumname_anz.getText().toString().length() == 0) {
74         Toast.makeText( context: Scannen.this, "Raum ist leer!", Toast.LENGTH_SHORT).show();
75     } else if (et_inventarnummer.getText().toString().length() == 0) {
76         Toast.makeText( context: Scannen.this, "Inventarnummer ist leer!", Toast.LENGTH_SHORT).show();
77     } else {
78         Geraet geraet = new Geraet(
79             tv_raumname_anz.getText().toString(),
80             spin_typen.getSelectedItem().toString(),
81             et_inventarnummer.getText().toString(),
82             et_notiz.getText().toString().matches( regex: "" ) ? "" : et_notiz.getText().toString());
83
84         Geraet.geraeteliste.add(geraet);
85         tv_geraeteCounter.setText("Erfasste Geräte: {String.valueOf(Geraet.geraeteliste.si...)");
86
87         // Meldung: Gerät hinzugefügt
88         Toast.makeText( context: Scannen.this, "Gerät hinzugefügt.", Toast.LENGTH_SHORT).show();
89
90         //Leeren des Inventarnummer- und Notizfeldes
91         et_inventarnummer.setText("");
92         et_notiz.setText("");
93     }
94 });
95
96
97
98
99
100
```

Klasse Raumdetails (Activity) – Ausschnitt

```
54
55 // erstellt die Tabelle im Hinblick auf Spaltenbreite, Tabellenkopfdaten und SwipeToRefresh
56 // das Setzen der Farben erfolgt über die xml-Datei
57 private void erstelleTabelle(){
58     // Setzen der Spaltenbreite
59     TableColumnPxWidthModel columnModel = new TableColumnPxWidthModel( columnCount: 3, defaultColumnWidthPx: 350);
60     columnModel.setColumnWidth( columnIndex: 0, columnWidthPx: 300);
61     columnModel.setColumnWidth( columnIndex: 1, columnWidthPx: 450);
62     columnModel.setColumnWidth( columnIndex: 2, columnWidthPx: 800);
63
64     tv_raumdetails.setColumnModel( columnModel );
65
66     // Setzen des Tabellenkopfes
67     final String[] spaltennamen = { "Gerätetyp", "Inventarnummer", "Notiz" };
68     tv_raumdetails.setHeaderAdapter(new SimpleTableHeaderAdapter( context: Raumdetails.this, spaltennamen));
69
70     // Aktivieren von SwipeToRefresh-Funktion
71     tv_raumdetails.setSwipeToRefreshEnabled( true );
72 }
73
74 // Laden und Anzeigen der Daten
75 private void ladeDaten(String raum){
76     ArrayList<Geraet> geraeteListe = DateiHelper.leseDateiAus(raum);
77     String [][] tabellendaten = new String[geraeteListe.size()][4];
78
79     for(int i = 0; i < geraeteListe.size(); i++){
80         tabellendaten[i][0] = geraeteListe.get(i).getGeraeteTyp();
81         tabellendaten[i][1] = geraeteListe.get(i).getInventarnummer();
82         tabellendaten[i][2] = geraeteListe.get(i).getNotiz();
83     }
84     tv_raumdetails.setDataAdapter(new SimpleTableDataAdapter( context: Raumdetails.this, tabellendaten));
85 }
86
```

Klasse DateiHelper

```
1 package com.scanner.fbs_scanner.Standardklassen;
2
3 import ...
4
5 // Diese Klasse ermöglicht diverse Dateioperationen
6 public final class DateiHelper{
7
8     // bevor Berechtigungen abgefragt werden, dieser Variable die anfragende Aktivitätsklasse zuweisen
9     // (z.B. DateiHelper.activityPlaceholder = MainActivity.this;)
10    public static Activity activityPlaceholder;
11
12    private static boolean externerSpeicherLesbar, externerSpeicherBeschreibbar;
13    public static final String schreibPermission = Manifest.permission.WRITE_EXTERNAL_STORAGE;
14    public static final String lesePermission = Manifest.permission.READ_EXTERNAL_STORAGE;
15    public static final int STORAGE_REQUEST_CODE = 100;
16    private static File rootVerzeichnis = Environment.getExternalStorageDirectory();
17    private static File osvVerzeichnis = new File(rootVerzeichnis.getAbsolutePath(), child: "/fbs");
18
19    // prüft, ob hardwarseitig Speicher zum Lesen und Schreiben verfügbar ist und setzt entsprechend die Variablen
20    // diese Methode sollte aufgerufen werden, bevor eine Datei erstellt/beschrieben oder ausgelesen wird
21    private static void pruefeSpeicher()
22    {
23        String state = Environment.getExternalStorageState();
24        if (state.equals(Environment.MEDIA_MOUNTED)) {
25            externerSpeicherLesbar = externerSpeicherBeschreibbar = true;
26        } else if (state.equals(Environment.MEDIA_MOUNTED) || state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
27            externerSpeicherLesbar = true;
28            externerSpeicherBeschreibbar = false;
29        } else {
30            externerSpeicherLesbar = externerSpeicherBeschreibbar = false;
31        }
32    }
33}
```

```
59 // prüft, ob Les- und Schreibberechtigungen vorliegen und fordert diese wenn nötig an
60 // liegen die Berechtigungen schon vor, wird bereits in dieser Methode die Datei erstellt
61 public static void fordereLesUndSchreibPermissionAn(boolean nurVerzeichnis) {
62     // ist API-Version kleiner 23, werden Permissions schon bei der Installation zugestimmt
63     if (Build.VERSION.SDK_INT < 23) {
64         return;
65     }
66     else {
67         if (ContextCompat.checkSelfPermission(activityPlaceholder, lesePermission) != PackageManager.PERMISSION_GRANTED &&
68             ContextCompat.checkSelfPermission(activityPlaceholder, schreibPermission) != PackageManager.PERMISSION_GRANTED) {
69
70             if (ActivityCompat.shouldShowRequestPermissionRationale(activityPlaceholder, schreibPermission)) {
71                 new AlertDialog.Builder(activityPlaceholder)
72                     .setTitle("Les- und Schreibberechtigung erforderlich")
73                     .setMessage("Zum Abrufen und Abspeichern der erfassten Daten wird ei...")
74                     .setPositiveButton("OK", (dialog, which) -> {
75                         ActivityCompat.requestPermissions(activityPlaceholder, new String[]{lesePermission, schreibPermission}, STORAGE_REQUEST_CODE);
76                     })
77                     .setNegativeButton("Abbrechen", new DialogInterface.OnClickListener() {
78                         @Override
79                         public void onClick(DialogInterface dialog, int which) {
80                             dialog.dismiss();
81                         }
82                     })
83                     .create().show();
84             }
85             else {
86                 ActivityCompat.requestPermissions(activityPlaceholder, new String[]{lesePermission, schreibPermission}, STORAGE_REQUEST_CODE);
87             }
88         }
89         // sind die Berechtigungen schon vorhanden, springt das Programm nicht in die onRequestPermissionsResult
90         // Methode, deswegen müssen an dieser Stelle die sonst üblichen Operationen durchgeführt werden!
91         else {
92             if (!nurVerzeichnis) {
93                 erzeugeUndBeschreibeDatei();
94                 Toast.makeText(activityPlaceholder, "Datei wurde erstellt.", Toast.LENGTH_LONG).show();
95                 Geraet.geraeteliste.clear();
96                 activityPlaceholder.startActivity(new Intent(activityPlaceholder, MainActivity.class));
97             }
98         }
99     }
```



```

109 // erzeugt eine Datei, dessen Name = jeweiliger Raumname einer Erfassung
110 // beschreibt die Datei mit den hinzugefügten Geräten
111 public static void erzeugeUndBeschreibeDatei() {
112     if (Geraet.gerasetliste.get(0) != null) {
113         String dateiRaumName = Geraet.gerasetliste.get(0).getRaumName().trim().replace( " ", replacement: " " ) + ".csv";
114
115         pruefeSpeicher();
116         if (externerSpeicherBeschreibbar) {
117             if (!csvVerzeichnis.exists()) {
118                 csvVerzeichnis.mkdir();
119
120                 // stelle sicher, dass der Zugriff auf die erstellte Datei möglich ist
121                 // und sich das Filesystem aktualisiert
122                 csvVerzeichnis.setReadable( true );
123                 csvVerzeichnis.setWritable( true );
124                 csvVerzeichnis.setExecutable( true );
125                 MediaScannerConnection.scanFile(activityPlaceholder, new String[] { csvVerzeichnis.toString(), mimeTypees: null, callback: null });
126             }
127             File datei = new File( csvVerzeichnis, dateiRaumName );
128             try {
129                 FileOutputStream fos = new FileOutputStream( datei, append: false );
130                 for (Geraet geraet : Geraet.gerasetliste) {
131                     fos.write( geraet.gibDateiFormat().getBytes() );
132                 }
133                 fos.flush();
134                 fos.close();
135
136                 datei.setReadable( true );
137                 datei.setWritable( true );
138                 datei.setExecutable( true );
139                 MediaScannerConnection.scanFile( activityPlaceholder, new String[] { datei.getPath(), mimeTypees: null, callback: null });
140             } catch (FileNotFoundException e) {
141                 e.printStackTrace();
142             } catch (IOException e) {
143                 e.printStackTrace();
144             }
145         }
146     }
}

```

```

149 // erstellt den FBS-Ordner
150 public static void erstelleVerzeichnis() {
151     pruefeSpeicher();
152     if (externerSpeicherBeschreibbar) {
153         if (!csvVerzeichnis.exists()) {
154             csvVerzeichnis.mkdir();
155         }
156     }
157 }
158
159 // gleicht den übergebenen Raumnamen (ohne .csv) mit den im Verzeichnis FBS befindlichen csv-Dateien
160 // ab, bei Übereinstimmung wird die jeweilige Datei ausgelesen und in Form eines Geraete-Arrays zurückgegeben
161 public static ArrayList<Geraet> leseDateiAus(String raumname)
162 {
163     ArrayList<Geraet> listeErfassungen = new ArrayList<>();
164
165     for (File f : csvVerzeichnis.listFiles()) {
166         if (f.getName().substring(0, f.getName().length() - 4).equals(raumname)) {
167             File file = new File(csvVerzeichnis, f.getName());
168             try {
169                 BufferedReader br = new BufferedReader(new FileReader(file));
170                 String zeile;
171
172                 while ((zeile = br.readLine()) != null) {
173                     String[] temp = zeile.split( " " );
174                     Geraet erfassung = new Geraet( temp[0], temp[1], temp[2], temp[3] );
175                     listeErfassungen.add( erfassung );
176                 }
177                 br.close();
178             } catch (FileNotFoundException e) {
179                 e.printStackTrace();
180             } catch (IOException e) {
181                 e.printStackTrace();
182             }
183         }
184     }
185
186     return listeErfassungen;
}

```

```

188
189 // gibt alle im Verzeichnis FBS befindlichen csv-Dateien ohne Suffix in Form eines String-Arrays zurück
190 public static ArrayList<String> gibRaumListe() {
191     ArrayList<String> raumliste = new ArrayList<>();
192
193     for(File f : csvVerzeichnis.listFiles()) {
194         if(f.isFile() && f.getName().endsWith( ".csv" )) {
195             raumliste.add(f.getName().substring(0,f.getName().length() -4));
196         }
197     }
198     return sortiereListe(raumliste);
199 }
200
201 // löscht die Datei, die als Parameter in Form des reinen Raumnamens übergeben wird
202 public static void löscheDatei(String raumname, Activity activity){
203     boolean gelöscht = false;
204     for(File f : csvVerzeichnis.listFiles()){
205         if(f.getName().substring(0,f.getName().length() -4).equals(raumname)){
206             gelöscht = f.delete();
207         }
208         if(gelöscht){
209             String message = "Raum" + " " + raumname + " " + "wurde gelöscht.";
210             Toast.makeText( activity, message, Toast.LENGTH_LONG ).show();
211         }
212     }
213 }

```

```

214
215 // löscht alle Raumdateien im Verzeichnis FBS
216 public static void löscheAlleDateien(final Activity activity) {
217     new AlertDialog.Builder( activity )
218         .setTitle( "Alle Räume löschen?" )
219         .setMessage( "Sollen wirklich alle Räume unwiderruflich gelöscht werden?" )
220         .setPositiveButton( "Ja", (dialog, which) -> {
221             boolean gelöscht = false;
222             for (File f : csvVerzeichnis.listFiles()) {
223                 if (f.isFile() && f.getName().endsWith( ".csv" )) {
224                     gelöscht = f.delete();
225                 }
226             }
227             if (gelöscht) {
228                 String message = "Alle Räume gelöscht.";
229                 Toast.makeText( activity, message, Toast.LENGTH_LONG ).show();
230                 activity.startActivity(new Intent(activity, Hauptmenue.class) );
231             }
232         })
233         .setNegativeButton( "Abbrechen", (dialog, which) -> {
234             dialog.dismiss();
235         })
236         .create() .show();
237 }

```

```

244
245 // Methode, die es ermöglicht, eine alphanumerische Sortierung des übergebenen Arrays vorzunehmen
246 private static ArrayList<String> sortiereListe(ArrayList<String> zuSortierendeListe){
247
248     final Pattern p = Pattern.compile("^\\d+$");
249
250     Comparator<String> c = (object1, object2) -> {
251         Matcher m = p.matcher(object1);
252         Integer number1 = null;
253         if (!m.find()) {
254             return object1.compareTo(object2);
255         }
256         else {
257             Integer number2 = null;
258             number1 = Integer.parseInt(m.group());
259             m = p.matcher(object2);
260             if (!m.find()) {
261                 return object1.compareTo(object2);
262             }
263             else {
264                 number2 = Integer.parseInt(m.group());
265                 int comparison = number1.compareTo(number2);
266                 if (comparison != 0) {
267                     return comparison;
268                 }
269                 else {
270                     return object1.compareTo(object2);
271                 }
272             }
273         }
274     };
275
276     Collections.sort(zuSortierendeListe, c);
277     return zuSortierendeListe;
278 }

```

A.4.3 Fertige App

Hauptmenü



Hauptmenü mit Raumname eingeben

The screenshot shows the main menu of the 'FBS-Scanner' application. At the top, there is a dark red header bar with the text 'FBS-Scanner' in white. Below this, the background is a dark grey-green color. In the center, there is a large, semi-transparent white dialog box titled 'Raum erfassen' (Record Room). The dialog box contains the text 'Bitte Raumnamen eingeben:' (Please enter room name:) followed by a white input field with a yellow underline. Below the input field, there are two yellow buttons: 'ABBRECHEN' (Cancel) and 'RAUM ERFASSEN' (Record Room). In the background, behind the dialog box, there is a large, semi-transparent logo for 'Ferdinand Braun' (fB) and a dark red button labeled 'RAUM ERFASSEN'. At the bottom of the screen, there is another dark red button labeled 'RÄUME ANZEIGEN' (Show Rooms). The top status bar of the phone shows various icons including a speech bubble, Facebook, music, YouTube, email, Bluetooth, a square icon, a clock, 'VoLTE 4G', a signal strength indicator, a battery icon at 84%, and the time 11:14.

FBS-Scanner

Raum erfassen


Bitte Raumnamen eingeben:

ABBRECHEN RAUM ERFASSEN

RAUM ERFASSEN

RÄUME ANZEIGEN

Raumerfassung



← Erfassung - Raum: test

Raumname: test

Gerätetyp: PC ▼

Inventarnummer: Scannen oder Eingeben ...

SCANNEN

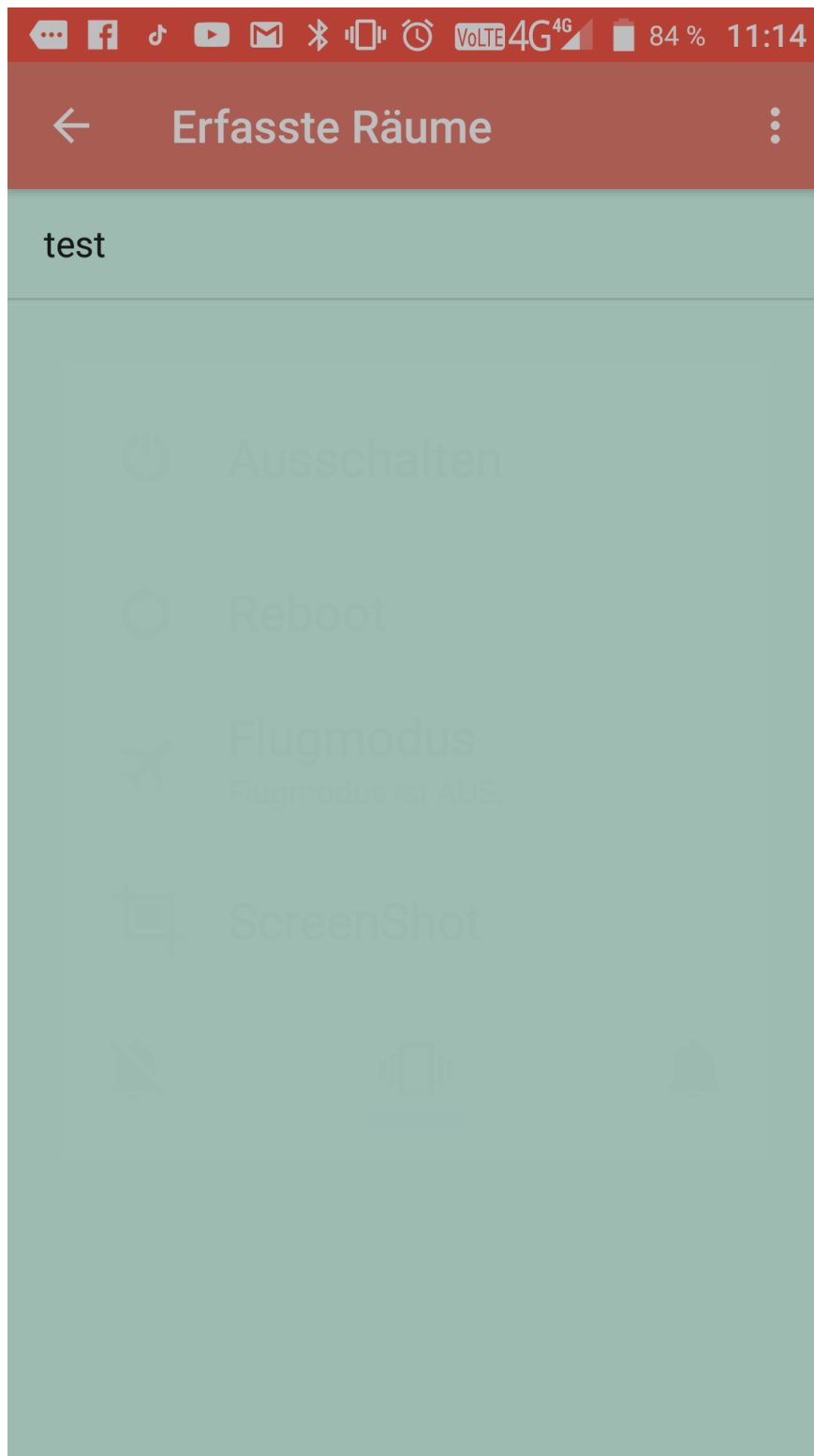
Notiz: Bemerkung ... (optional)

GERÄT HINZUFÜGEN

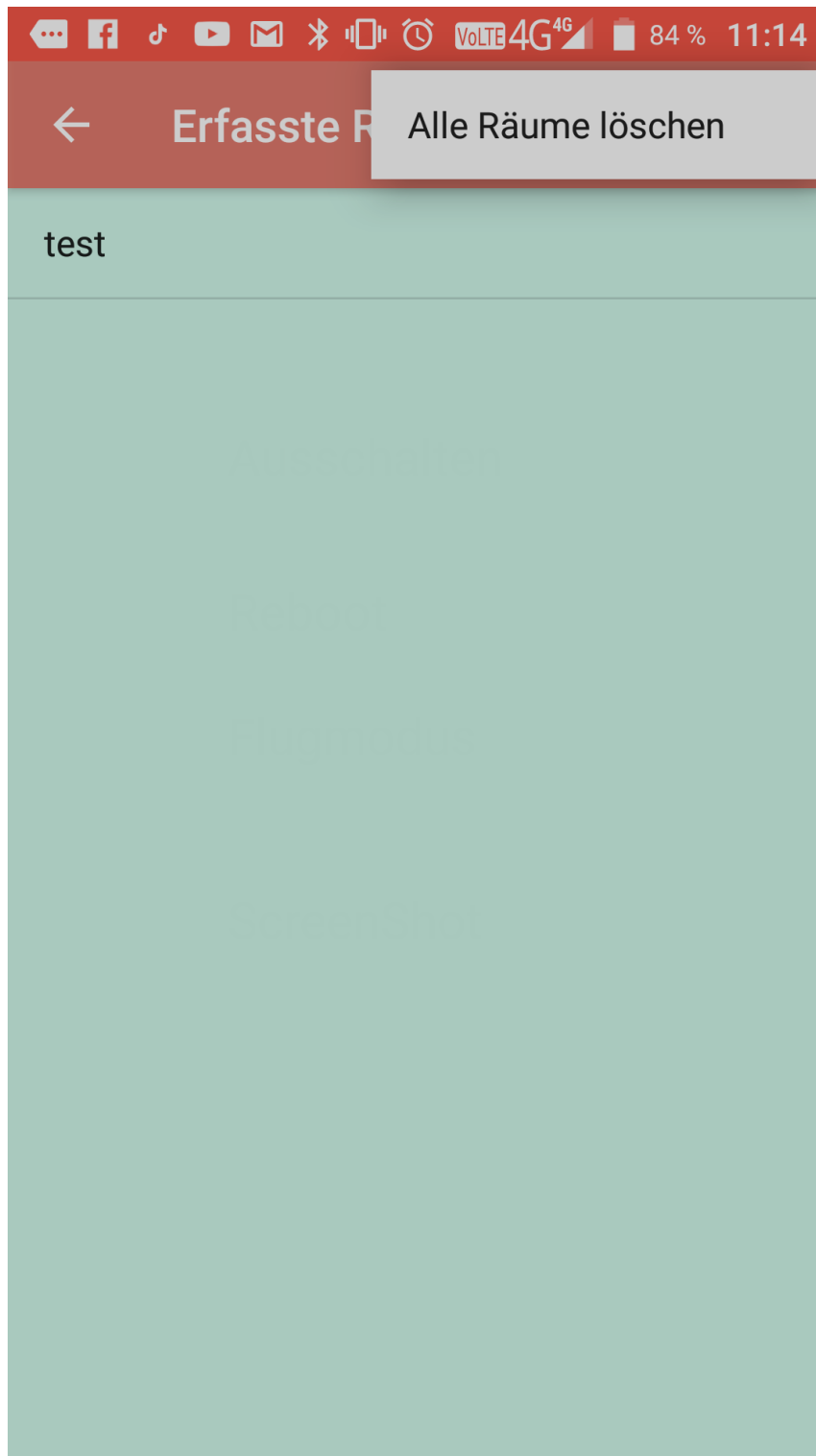
Erfasste Geräte: 1

RAUMERFASSUNG ABSCHLIESSEN

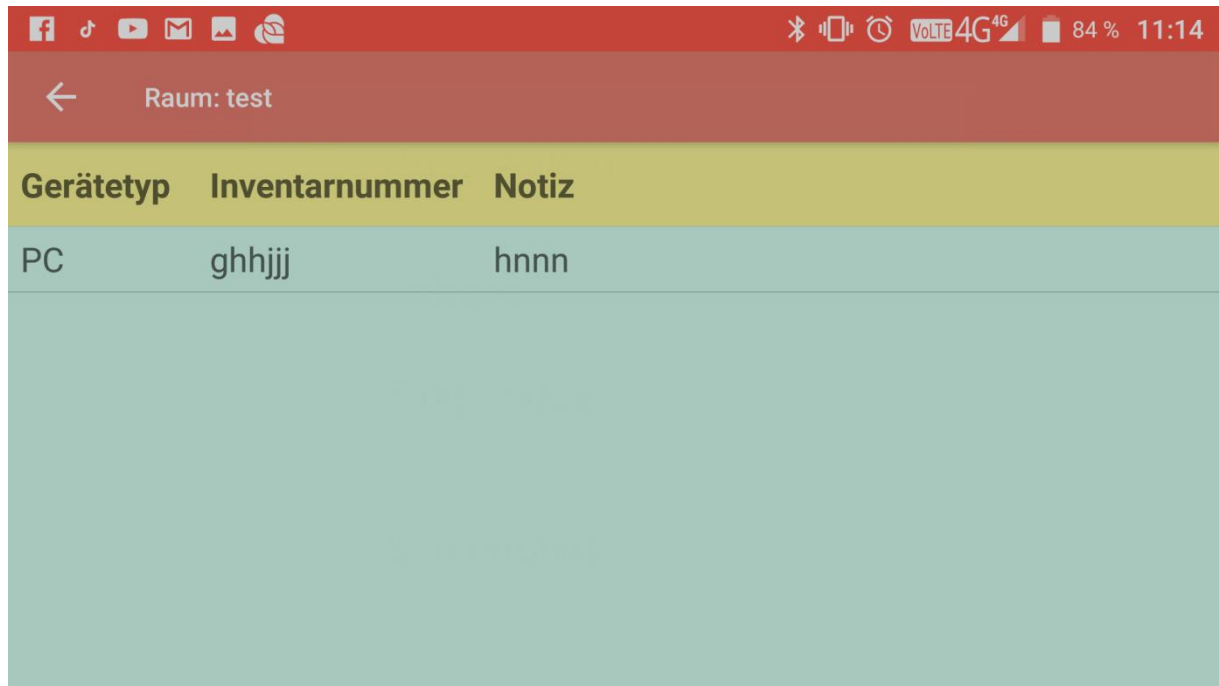
Erfasste Räume



Erfasste Räume alle Löschen



Raumdetails



The image shows a mobile application interface for 'Raumdetails'. At the top is a red status bar with various icons and the time '11:14'. Below it is a red header bar with a back arrow and the text 'Raum: test'. The main content area has a light green background and contains a table with three columns: 'Gerätetyp', 'Inventarnummer', and 'Notiz'. The table has one data row with the values 'PC', 'ghhj', and 'hnnn'.

Gerätetyp	Inventarnummer	Notiz
PC	ghhj	hnnn