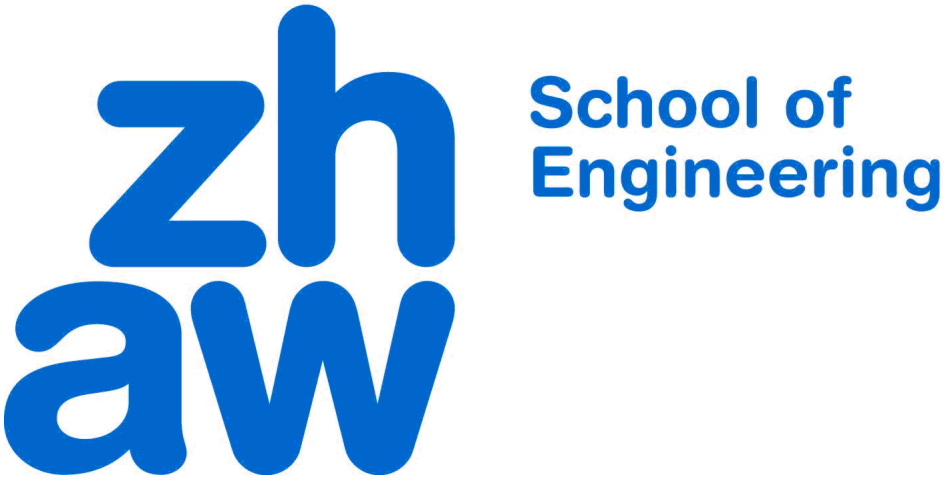

Collector

Semesterarbeit

Autor: Mathias Weigert

Inhaltsverzeichnis

1. Recherche **6**
1.1. My books . . . 6
1.2. Meine Bücher Pro . . . 6
1.3. Codex . . . 7
1.4. Movie Collection . . . 8
1.5. Movielicious . . . 10
1.6. Nintendo Collection . . . 11
1.7. Video Games Manager Collector . . . 11
1.8. Überblick . . . 12

2. Ist-Analyse **13**

3. Anforderungsanalyse **14**
3.1. Allgemeines . . . 14
3.1.1. Dokumentation der Anforderungen . . . 14
3.1.1.1. Natürliche Sprache . . . 14
3.1.2. Systematik der Anforderungen . . . 15
3.1.2.1. Art der Anforderung . . . 15
3.1.2.2. Priorität der Anforderung . . . 15
3.1.2.3. Beispiel: Nummerierung und Bezeichnung der Anforderung . . . 16
3.1.3. Systemanforderungen . . . 16
3.2. Übersicht . . . 16
3.3. Funktionen . . . 16
3.4. Übersicht Funktionen . . . 17
3.4.1. Anlegen eines Items . . . 18
3.4.1.1. Manuelles Anlegen eines Items . . . 18
3.4.1.2. Automatisches Anlegen eines Items . . . 18
3.4.2. Bearbeiten eines Items . . . 19
3.4.3. Löschen eines Items . . . 19
3.4.4. Ausgabe der Items . . . 19
3.4.5. Verwalten verliehener Items . . . 21
3.4.6. Exportieren aller ausgewählter Items . . . 21
3.4.7. Datenbank Administration . . . 21
3.4.8. REST Schnittstelle zur Abfrage von Metadaten diverser Online Datenbanken . . . 23
3.4.9. Kamera als Barcodescanner . . . 23
3.5. Datenbank . . . 23
3.5.1. Übersicht der Tabellen und Felder . . . 24

Inhaltsverzeichnis

3.6. GUI . . . 25
3.6.1. Startseite . . . 25
3.6.2. Barcode Scannen . . . 25
3.6.3. Manuell Anlegen / Item Bearbeiten . . . 25
3.6.4. Detail Ansicht . . . 27
3.6.5. Sicherheitsfrage Item Löschen . . . 27
3.6.6. Item verleihen . . . 28
3.6.7. Filter . . . 28
3.6.7.1. Allgemein . . . 29
3.6.7.2. Buch & Spiel . . . 29
3.6.7.3. Film . . . 29

4. Konzept **31**
4.1. Funktionen . . . 31
4.1.1. Verleih Verwaltung . . . 31
4.1.2. Filtern der Sammlung . . . 31
4.1.3. CSV Export der Items . . . 32
4.2. Datenbank . . . 32
4.2.1. Allgemeines . . . 32
4.2.2. Aufbau der SQLITE Datenbank . . . 32
4.2.3. Übersicht Datenbank . . . 34
4.3. GUI . . . 35
4.3.1. Allgemeines . . . 35
4.3.2. Hauptmenu / Startscreen . . . 35
4.3.3. Neues Item . . . 35
4.3.4. Sammlung . . . 37
4.3.5. Detail Ansicht . . . 37
4.3.6. Leihwesen . . . 37
4.3.7. Verleih Verwaltung . . . 37
4.3.8. Export Database . . . 38
4.3.9. Info . . . 38
4.3.10. Settings . . . 38
4.3.11. Datenbankverwaltung . . . 38
4.3.11.1. Freunde Verwalten . . . 39
4.3.11.2. History . . . 39

5. Umsetzung **40**
5.1. Funktionen . . . 40
5.1.1. Filtern der Sammlung . . . 40
5.1.2. Export der Daten . . . 41

Inhaltsverzeichnis

5.2. Datenbank 43
5.3. GUI 45
5.4. Fazit 48

A. Anhang **49**
A.1. Konventionen 49
A.2. Verwendete Tools & Software 49
A.2.1. Dokumentation & Präsentation 49
A.2.2. Programmierung 49
A.2.3. Code und Versionsverwaltung 49
A.2.4. Datenbank 49
A.2.5. GUI 49
A.2.6. UML 49
A.3. Design Entscheidungen 49

Version

Angabe der aktuellen Version des Dokuments. Alle Änderungen, werden mit Versionsnummer, Datum, Autor und Informationen betreffend der Änderung dokumentiert.

Version	Datum	Autor	Bemerkung
0.01	30.11.15	M. Weigert	Initiales Dokument
0.02	30.11.15	M. Weigert	Aufgabe hinzugefügt
0.03	02.12.15	M. Weigert	Neue LaTex Konfiguration, Literaturverzeichnis, erste Recherche
0.04	04.12.15	M. Weigert	Kapitelstruktur angelegt. Recherche fortgeführt
0.04	03.02.16	M. Weigert	Zeitmanagement und Feinstruktur
0.05	06.02.16	M. Weigert	Neue Datenstruktur und Anforderungen dokumentiert.
0.06	06.03.16	M. Weigert	IST-Analyse angefangen.
0.07	06.03.16	M. Weigert	Angefangen GUI zu dokumentieren.
0.08	07.03.16	M. Weigert	GUI weiter dokumentiert.
0.10	24.03.16	M. Weigert	Anforderungsanalyse neu gegliedert.
0.11	25.03.16	M. Weigert	Übersichten bei der Anforderungsanalayse und dem DB Design.
0.25	10.04.16	M. Weigert	Anforderungsanalyse fertig gestellt.
0.30	16.04.16	M. Weigert	Konzeption angefangen (GUI & Datenbank).
0.50	24.04.16	M. Weigert	Kapitel Konzept abgeschlossen.
0.75	02.05.16	M. Weigert	Umsetzung weitgehend Dokumentiert.
0.90	14.05.16	M. Weigert	Dokument fertiggestellt.
1.00	18.05.16	M. Weigert	Fazit hinzugefügt, Fehlerkorrekturen.

1. Recherche

Die komplette Recherche findet im Google App Store GooglePlay statt. Für die nachfolgende Ist-Analyse werde ich noch auf die einzelne Apps aus dem Apple App Store verweisen, welche ich selbst einsetze und welche mir im Bezug auf Funktionalität und Design von *Collector* die ein oder andere Idee geliefert haben.

1.1. My books

Diese App dient der Inventarisierung und Organisation der persönlichen Bibliothek. Es ist einfach die komplette Bücher Sammlung zu speichern. Bücher können einfach, durch scannen des Barcode hinzugefügt werden.

Features:

- Hinzufügen eines Buches durch scannen des Barcode
- Hinzufügen eines Buches durch eingeben des ISBN Code
- Hinzufügen eines Buches durch eingeben von Titel, Autor, etc...
- Ansicht aller Bücher eines Autors
- Im- und Export aller Daten in ein File

Durch eine Vielzahl von Listen kann der Bücher Katalog organisiert werden.

- Bücher welche ich kaufen möchte
- Bücher welche ich nochmals lesen möchte
- Bücher welche mir gefallen haben
- Bücher welche ich verliehen habe

Es gibt eine Vielzahl von Filtermöglichkeiten, wie Genre, Autor, Beurteilung, Leser und viele weiteren. [1]

1.2. Meine Bücher Pro

Sind Sie ein unersättlicher Book-Reader? Haben Sie Schwierigkeiten, den Überblick über alle Ihre Bücher? Verwalten Sie Ihre Buch-Katalog mit diesem App. Behalten Sie Ihre Büchersammlung zusammen mit persönlichen Bewertungen und Anmerkungen, die Sie mit anderen teilen oder privat halten können. Prüfen Sie, ob Sie bereits ein Buch über das Telefon, während Sie einkaufen sind. [2]

- Verwenden Sie die Bulk-Barcode-Scan-Option, um Ihre Bücher durch schnelles Scannen des Barcodes auf der Titelseite mit der Kamera des Telefons eingeben.
- Denken Sie daran, die Bücher, die Sie kaufen wollen, mit der Wunschliste. • Export Ihr Buch info im CSV-Format als ein Tabellenkalkulationsprogramm auf Ihrem Computer zu öffnen.
- Nehmen Sie ein Bild von dem Buchumschlag mit Ihrem Telefon zu gehen zusammen mit Ihrem Buch Rating und beachten.
- Sichern Sie Ihre Buchkatalog Daten an den Server. Wenn Sie Ihr Telefon verlieren oder ein Upgrade auf ein neues Handy, nur importieren Sie Ihre Sammlung auf das neue Mobiltelefon, so dass Sie nicht haben, um wieder in alles.
- Track Statistiken über Ihre Bücher wie Gesamtzahl der Bücher Tracked, Average Rating, am häufigsten gelesen, Anzahl der Bücher von Genre verfolgt.

1.3. Codex

Codex ist eine Bücherverwaltungs-Anwendung, die Ihnen hilft, die Bücher Ihrer Bibliothek zu verwalten und zu katalogisieren. Alles, was Sie dazu tun müssen, ist, die ISBN des Buchs zu scannen und die Information zum Buch wird aus dem Web heruntergeladen.
Sie können Ihre Bücher ordnen, Ihre verliehenen Bücher verwalten und sogar eine Bücher-Wunschliste anlegen, die Ihnen hilft, die günstigsten Preise für die Bücher, die Sie kaufen wollen, zu ermitteln und sich daran zu erinnern, wo sie sie gefunden haben.
Teilen Sie Bücher mit Ihrem Freund / Ihrer Freundin über eine einfache SMS an Leute in Ihrer Kontaktliste. Codex holt sich alle Informationen, die Ihre Freunde über das Buch haben müssen.
Alle Buchinformationen können über das Internet heruntergeladen werden, einschließlich Buchtitel-Bild. Falls kein Buchtitel-Bild verfügbar ist, können Sie ein Bild aus Ihrem Telefonspeicher als Titelbild auswählen oder ein Foto des Buchs machen.
Codex unterstützt auch mehrere Sprachen, nämlich Englisch, Portugiesisch, Polnisch Französisch und Deutsch.

Mit Codex können Sie:

- Bücher einfach zu Ihrer Liste hinzufügen, indem Sie den ISBN-Barcode des Buchs scannen. Die Daten zum Buch werden aus dem Internet heruntergeladen.
- Ein Buch einfach hinzufügen, indem Sie die ISBN eintippen. Wenn kein Barcode verfügbar ist können Sie die ISBN immer manuell eingeben.
- Ein Buch hinzufügen, indem Sie die Buchinformationen eingeben. Falls das Buch keinen Barcode hat oder es nicht möglich ist, Buchinformationen im Web aufzufinden, können sie manuell eingegeben werden.

- Ihre verliehenen Bücher verwalten. Jedesmal wenn Sie ein Buch verleihen, hilft Ihnen Codex, sich daran zu erinnern, indem es Ihnen ermöglicht, das Buch als verliehen zu markieren und dazu die Person, der Sie es ausgeliehen haben, aus Ihren Kontakten einzutragen.
- Eine Wunschliste mit den Büchern, die Sie kaufen wollen, erstellen. Ein Buch, das Sie nicht besitzen, hinzufügen und das Geschäft, in dem Sie es gefunden haben. Jedes Buch kann mehreren Geschäften zugeordnet werden, so daß Sie Preise vergleichen und den günstigsten Preis suchen können.
- Sich an die Lage des Geschäfts erinnern, wo Sie die Bücher in der Wunschliste gefunden haben. Falls Sie die Adresse nicht wissen, kann Codex für Sie die GPS-Lokalisation vornehmen.
- Ihre Bücher nach Autor, Verlag oder Kategorie filtern.
- Leicht Bücher Ihrer Wunschliste oder von Ihnen verliehene Bücher finden.
- Ein Bild aus dem Bildspeicher Ihres Telefons auswählen, um es als Buchtitel-Bild zu verwenden. Wenn Sie kein Bild Ihres Buchtitels haven, fertigen Sie einfach ein Foto davon an.
- Wollen Sie dieses Buch mit einem Freund teilen? Empfehlen Sie das Buch einfach. Mit Codex können Sie ein Buch über eine einfache SMS-Mitteilung empfehlen. Wenn Ihr Freund / Ihre Freundin auf seinem Telefon Codex ebenfalls hat, wird Codex alle Buchinformationen zusammenstellen, die Ihr Freund / Ihre Freundin benötigt.
- Unterstützung in mehreren Sprachen, nämlich Englisch, Portugiesisch, Französisch und Deutsch erhalten.
- Bücher in den Formaten CSV und XML importieren und exportieren, so daß sie zwischen Geräten synchronisiert werden können.
- Büchersammlungen durchsuchen, um leicht das richtige Buch zu finden.
- Wenn Sie ein Buch verleihen, können Sie eine Rückgabefrist eingeben und Codex wird Sie dann erinnern, wenn die Rückgabefrist erreicht ist. Codex wird eine Verleih-Chronik für jedes Buch unterhalten, so daß Sie wissen, wem Sie es geliehen hatten.

Codex ist kostenlos, Sie können es solange Sie wollen ohne jegliche Einschränkungen nutzen. Wenn Sie irgendwelche Anregungen haben, kontaktieren Sie bitte das Entwickler-Team über codex.android@gmail.com. [3]

1.4. Movie Collection

Mithilfe der Hauptanwendung 'Movie Collection' ist es möglich die eigene Filmsammlung zu katalogisieren und zu verwalten. Nach dem Hinzufügen neuer Titel werden detaillierte Informationen geladen

und in der lokalen Datenbank gespeichert. [4]

Folgende Features sind in der Hauptanwendung bereits enthalten:

- Detaillierte Informationen zu Filmen wie Inhalt, Darsteller, Poster, Laufzeit, Regisseur ...
- Detaillierte Informationen zu Serien mit Staffel und Episoden informationen. (Es muss eine separate Liste für Serien in der Listenverwaltung angelegt werden. Anschließend können zu der neuen Liste Serien hinzugefügt werden)
- Verknüpfung von Film-Trailern. Ermöglicht das Schauen der Trailer über YouTube-App
- Hinzufügen von Filmen/Serien durch die Suche nach Titel / Schlagwort.
- Hinzufügen von Filmen/Serien durch das Scannen des Barcodes des Filmes (Schneller IN-APP Barcode-Scanner. Keine separate Applikation nötig).
- Hinzufügen eigener/privater Filme inkl. eigener Cover, Beschreibung etc.
- Vollwertige Listenverwaltung (es können so viele Listen erstellt werden wie man benötigt. (z.B. separate Listen für Filme, Serien, Musik DVDs, Wunschliste usw.)
- Verschieben/Kopieren von Filmen zwischen den Listen.
- Markieren von Filmen/Serien-Staffeln als verliehen.
- Listen- und Kachelansicht.
- Freie Anpassung der Cover/Postergröße in Listen- und Kachelansicht.
- Ansprechendes Look and Feel.
- Holo-Light und Holo-Dark Theme.
- Unterschiedliche Sortierungs/Filter-Optionen.
- Filme mit Zeitstempel als „Gesehen“ markieren.
- Suchen in der eigenen Datenbank.
- Backup-Manager zum Sichern und Wiederherstellen der internen Datenbank (Export/Import in verschiedenen Formaten möglich wie Binär/XML/CSV).
- Sichern der Backups auf dem externen Speicher/SD-Karte. Hochladen der Backups in die Cloud (Dropbox/Box/Google-Drive) oder Versenden an Freunde per Mail etc.
- Anzeige der Poster/Backdrops in Vollbildmodus.

- Anzeige der Filmdaten in der IMDB Applikation oder der Webpräsenz.
- Keine Registrierung notwendig.
- Kompatibel zu Android 3.0 „Honeycomb“.

1.5. Movielicious

Hast Du eine grosse Blu-ray oder DVD Sammlung? Movielicious ist eine einfache aber mächtige App welche dir hilft deine Filmsammlung zu organisieren. Mit Movielicious kannst du schnell Filme zu deiner Sammlung hinzufügen, finden und bearbeiten. Neue Filme können durch durchsuchen von verschiedenen Quellen, durch benutzen von Freitextsuche, oder durch scannen des Barcodes auf der DVD oder Blu-ray Hülle, hinzugefügt werden. Du kannst die Filme in deiner Sammlung oder Wunschliste organisieren. [5]

- Honeycomb Tablet Unterstützung
- IMDb Integration
- Wunschliste
- Export Sammlung oder Wunschliste via E-Mail, etc.
- Hinzufügen von Filmen via scannen des Barcode
- Hinzufügen von Filmen via Onlinesuche in verschiedenen Quellen
- Hinzufügen von Filmen via manuelle Eingabe
- Regal- oder Listenansicht der Filmbibliothek
- Suche in der Sammlung nach Film, Titel, Jahr oder Schauspieler
- Filter deine Sammlung nach Genre, Regisseur, Format, verliehen oder gesehen.
- Teile deine Lieblingsfilme via Facebook, Twitter oder Google+
- Bewerte Filme
- Stelle fest, wann du diesen Film das letzte mal gesehen hast
- Behalte den Überblick, welche Filme du an wen verliehen hast
- Funktioniert mit allen modernen Formaten (DVD, Blu-ray, HD-DVD) wie auch mit älteren Formaten (VHS, 35mm, 16mm), zusätzlich können noch eigene Formate hinzugefügt werden
- Verlinkung mit der IMDb Android App

- Sichere oder Stelle deine Filmbibliothek wieder her
- Import der Filmbibliothek aus der iOS Version von Movielicious
- Import aus einer Vielzahl von Quellen, inklusvie Textfile oder CSV

1.6. Nintendo Collection

We lovers of a time when games were true to their cause have been poorly served when it comes to a dynamic and modern way of cataloguing our trusty old games.

This stops now.

This application allows you to keep your games collection by your side at all times.

Initially this only caters for the NES to gauge the response from you. If you like the premise, your feedback will be taken on board to improve the application and to include more systems.

Also currently the box arts are hit and miss and so a more robust alogirthm to retrieve them is being constructed. [6]

1.7. Video Games Manager Collector

PS4, Xbox One, Wii U, PS3, 360, Nintendo 3DS, PlayStation Network, PS2, Vita, DS, und mehr Videospiele verwalten mit dieser Datenbank. Diese App bietet zu ihrer Sammlung Katalog, Verwaltung und Bibliothek.

Verwendet UPC Barcode-Scanning, Datenbanksuche und manuelle Barcode-Eingabe, sowie eine Vielzahl von unglaublichen Funktionen wie Android-, iOS- und Website-Synchronisierung, Wunschlisten, Filtern, Sortieren, Bibliotheksansichten, Bulk Hinzufügen und vieles mehr. Die App zeigt Informationen, Titelbild, und persönliche Daten an. Ideal für das Sammeln. [7]

Funktioniert für Videospiele von diesen Plattformen:

Playstation 4	Xbox One	Nintendo Wii U
Playstation 3	Xbox 360	Nintendo Wii
Playstation 2	Xbox Live	Nintendo 3DS
Playstation Vita	Xbox	Nintendo DS
Playstation Network		*und viele mehr!*

1.8. Überblick

Name	Bücher	Filme	Spiele	Scanner	Export	Verleih	Preis	Rating
My books	Ja	Nein	Nein	Ja	Ja	Ja	Gratis	3.7
Meine Bücher Pro	Ja	Nein	Nein	Ja	CSV	Nein	CHF 3.65	4.0
Codex	Ja	Nein	Nein	Ja	CSV, XML	Ja	Gratis	4.2
Movie Collection	Nein	Ja	Nein	Ja	Binär, CSV, XML	Ja	CHF 2.15	4.5
Movielicious	Nein	Ja	Nein	Ja	Ja	Ja	CHF 2.50	3.2
Nintendo Collection	Nein	Nein	Ja (NES)	Nein	Nein	Nein	CHF 1.47	5.0
Video Games Manager Collector	Nein	Nein	Ja	Ja	Nein	Nein	Gratis	3.6

2. Ist-Analyse

Die in Kapitel 1 beschrieben und in Sektion 1.8 zusammen gefassten Apps zeigen deutlich, dass es momentan weder in GooglePlay noch im Apple Appstore, eine App gibt, welche die im nächsten Kapitel definierten Anforderungen Gesamthaft erfüllen kann.

Apps, welche über Internet Datenbanken, Meta Informationen zu Büchern, Filmen oder Spielen abgleichen gibt es einige. Allerdings konnte diese Apps nie mehr als einen Bereich abdecken. Apps für die Archivierung von Spielen, sind sogar nur in der Lage Spiele eines Herstellers abzubilden. Meist sogar nur eine einzige Konsolen Generation des Herstellers.

Damit ich in Zukunft nicht mehr zwischen vielen Apps wechseln muss um meine Bücher, Filme und Spiele zu verwalten, habe ich mich entschlossen eine App zu programmieren, welche die wichtigsten Fähigkeiten der im vorigen Kapitel aufgeführten Apps vereint.

3. Anforderungsanalyse

Die Anforderungsanalyse ist in die folgenden 4 Teile gegliedert.

1. Allgemeines
   Generelle Definitionen zum Aufbau der Anforderungsanalyse. Allgemeine System Anforderungen.
2. Funktionen
   Anforderungen an die Funktionen der App.
3. Datenbank
   Anforderungen an die Datenbank der App.
4. User Interface
   Anforderungen an das Design des User Interface.

3.1. Allgemeines

3.1.1. Dokumentation der Anforderungen

Die Anforderungen werden um Mehrdeutigkeit und offene Definitionen zu vermeiden nach folgenden Modellen der Anforderungsanalyse

- Natürliche Sprache
- Aktivitätsdiagramm
- Use Case

dokumentiert.
Bei komplexeren Anforderungen, welche durch die verwendeten Modelle nicht eindeutig dokumentiert werden können, werden zur Spezifizierung noch weitere UML Modelle eingesetzt.

Diese weiteren Modelle wird aus den Notationen von UML 2.3 so ausgewählt, dass die Anforderung Eindeutig beschrieben werden kann.

3.1.1.1. Natürliche Sprache

Um die Mehrdeutigkeit und den möglichen Interpretationsspielraum der natürlichen Sprache zu minimieren, muss die Dokumentation der Anforderung in natürlicher Sprache folgenden Regeln genügen.

- Anforderungen immer anhand der Prozesse erklären.
- Jeder Prozess einer Anforderung muss eindeutig beschrieben sein.

- Substantive, welche zur Anforderungs- oder Prozess Beschreibung eingesetzt werden, müssen über den Bezug für den Leser eindeutig sein.
- Bei Mengen und Häufigkeiten nur folgende Quantoren benutzen.
  - immer / nie
  - jeder / kein
  - alle / irgendein(er) / nichts
- Bedingungen immer vollständig dokumentieren.
- Nur den sprachlichen Aktiv verwenden.

3.1.2. Systematik der Anforderungen

Jede Anforderung erhält eine eindeutige sechsstellige Anforderungsnummer. Diese leitet sich aus folgenden drei Bestandteilen ab.

1. Art der Anforderung (Funktionelle, Datenbank, GUI, Sonstige)
2. Priorität (1 bis 4)
3. Laufende eindeutige Nummer

3.1.2.1. Art der Anforderung

Zur Gliederung wird eine zweistellige Nummer verwendet.

Nr.	Art der Anforderung
10	Anforderung an die Funktion der App
20	Anforderung an die Datenbank
30	Anforderung an das GUI
40	Sonstige Anforderungen

3.1.2.2. Priorität der Anforderung

Die Priorität der Anforderung wird in Ziffern 1 (höchste) bis 4 (niedrigste) definiert.

1. Grundlage (bzw. absolutes Muss)
2. Muss für Grundlegende Funktionen der App
3. Steigerung der Usability für den Nutzer
4. Wäre eine tolle zusätzliche Funktion.

3.1.2.3. Beispiel: Nummerierung und Bezeichnung der Anforderung

Die Anforderung „*Datenbank Design*“ hat folgende Anforderungsnummer (ID):

Art	**Prio**	**lfd. Nummer**
2-stellig	*1-stellig*	*3-stellig*
20	1	001

Die komplette Bezeichnung der Anforderung würde wie folgt aussehen.

[201001] Datenbank: Design

3.1.3. Systemanforderungen

Die App wurde für ein Samsung Galaxy S5 mit Android 5.0 entwickelt. Aufgrund der verwendeten Android Layouts, sollte die App allerdings ohne Probleme auf allen Android Smartphones und Tablets mit mindestens Android 5.0 funktionieren.

3.2. Übersicht

ID	Prio	Anforderung
101001	1	Item automatisch und manuell anlegen
101002	1	Item Information bearbeiten
101003	1	Item löschen
102001	2	Filter für die Sammlung
102002	2	Verleihverwaltung
102003	2	Items exportieren (CSV)
102004	2	Datenbank Administration implementieren
103001	3	REST Schnittstelle für Metadaten Abfrage diverser Online Datenbanken
104001	4	Kamera als Barcodescanner
201001	1	Datenbank Design
201002	1	Datenbank in App erstellen
301001	1	GUI: Item bearbeiten
302001	2	GUI: Items exportieren
302002	2	GUI: Liste verliehener Items
304001	4	GUI: Kamera als Barcodescanner

3.3. Funktionen

Die App Collector soll den folgenden Beschriebenen Funktionsumfang besitzen. Dieser wurde anhand der Recherche, der Ist-Analyse und dem Bedarf von M. Weigert ermittelt. Probleme bei der Umsetzung der Funktionen und deshalb nötige Anpassungen werden im Kapitel 5 ab Seite 40 dokumentiert.

3.4. Übersicht Funktionen

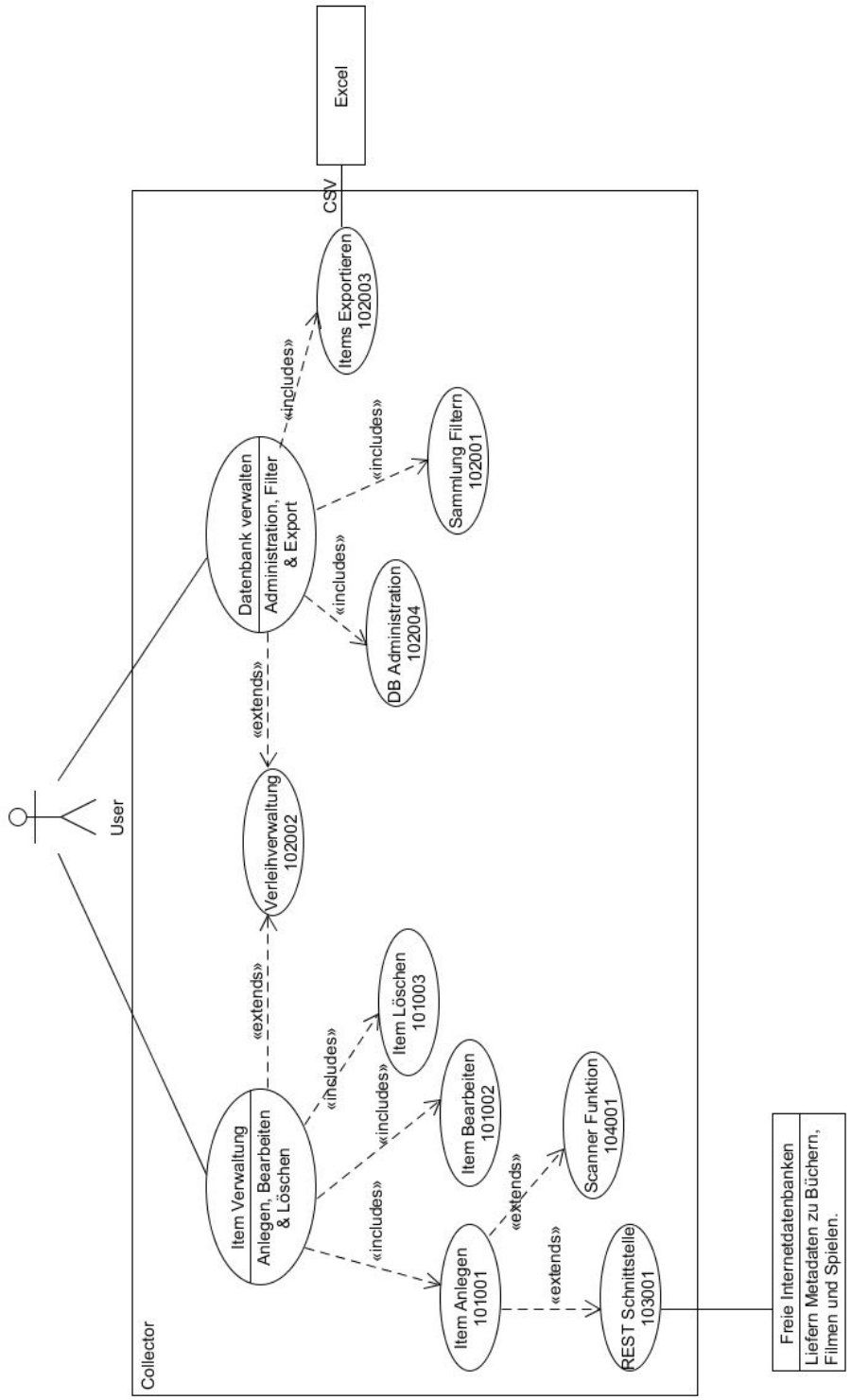

Abbildung 3.1.: Überblick Funktionen

3.4.1. Anlegen eines Items

[101001] Funktion: Item Anlegen

Das Anlegen eines neuen Items soll auf zweierlei Arten Möglich sein.

1. Manuell
2. Automatisch

Nach der Auswahl `Neues Item` wird standardmässig der Screen für das automatische Anlegen eines Items, mit aktivierter Kamera, geöffnet. Auf diesem Screen gibt es einen Button für das manuelle Anlegen eines Items.

3.4.1.1. Manuelles Anlegen eines Items

Sobald der User `manuell` ausgewählt hat öffnet sich der Screen für Bearbeiten und Manuelles anlegen eines Items. Da es sich hierbei um die neu Anlage eines Items handelt ist der Screen noch komplett ohne angezeigte Daten.

Nun kann der User alle Parameter, welche ein Item besitzt manuell eingeben und speichern. Um welche Daten es sich hierbei handelt ist in diesem Kapitel im Absatz 3.5.1 ab Seite 24 dokumentiert.

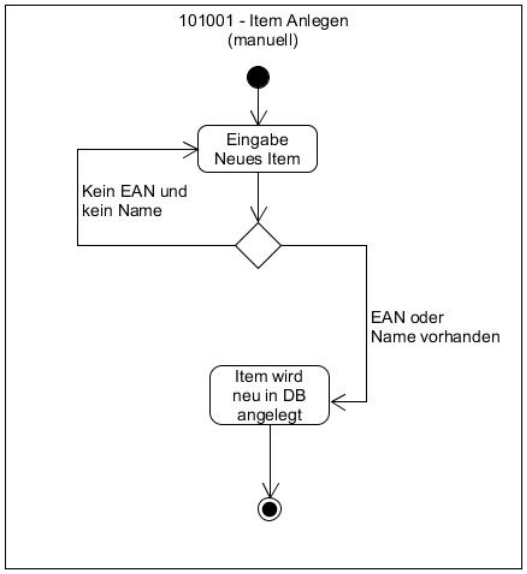

Abbildung 3.2.: Aktivitätsdiagramm: Manuell Item anlegen

3.4.1.2. Automatisches Anlegen eines Items

Das automatische Anlegen eines Items ist in der App als Standard, für das Anlegen eines Items definiert. Sobald der User `Neues Item` ausgewählt hat öffnet sich die Kamera. Der User muss nun nur noch den

3. Anforderungsanalyse

Barcode scannen (photographieren) und eine automatische Suche, im Internet, nach Metadaten zu diesem Barcode wird gestartet. Die gefundenen Metadaten werden dem User angezeigt und er kann diese nun bestätigen oder noch anpassen.

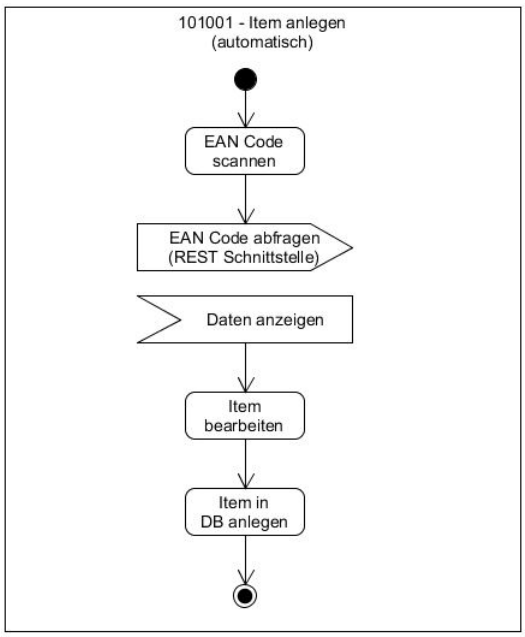

Abbildung 3.3.: Aktivitätsdiagramm: Automatisch Item anlegen

3.4.2. Bearbeiten eines Items

[101002] Funktion: Item Bearbeiten

Ein vom User ausgewähltes Item, kann durch die Funktion `Bearbeiten` manuell bearbeitet werden. Es gibt für den User keine Einschränkungen, jedes Datenelement kann angepasst werden.

3.4.3. Löschen eines Items

[101003] Funktion: Item löschen

Ein vom User ausgewähltes Item, wird komplett (mit all seinen Datenelementen) und unwiderruflich aus der Datenbank gelöscht. Zum Schutz vor unfreiwilligem Löschen, wird vor dem durchführen der Löschung der User nochmals aufgefordert den Löschvorgang zu bestätigen.

3.4.4. Ausgabe der Items

[102001] Funktion: Items anzeigen

3. Anforderungsanalyse

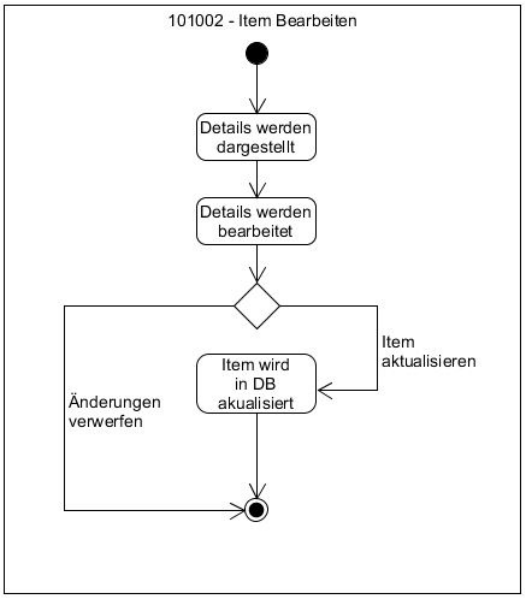

Abbildung 3.4.: Aktivitätsdiagramm: Item bearbeiten

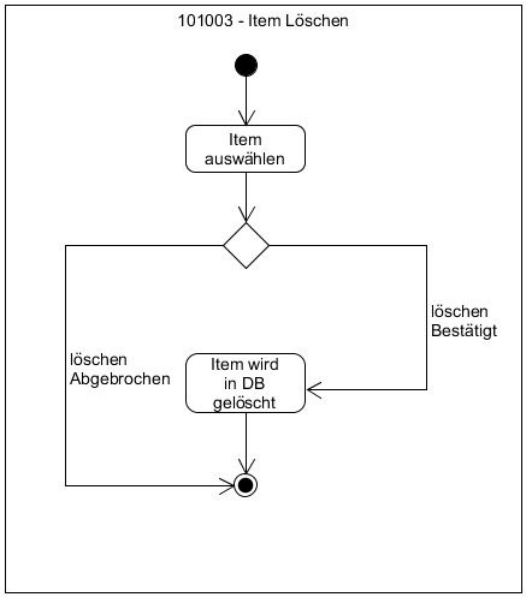

Abbildung 3.5.: Aktivitätsdiagramm: Item löschen

3. Anforderungsanalyse

Der User kann sich entweder die komplette Sammlung anzeigen lassen oder durch Eingabe eines oder mehreren Filterkriterien die angezeigt Auswahl einschränken. Jedes Kriterium, welches von einem Item gespeichert wird kann als Filter verwendet werden. Werden vom User mehrere Kriterien angegeben, werden diese mittels einer *UND Verknüpfung* im Filter verwendet.

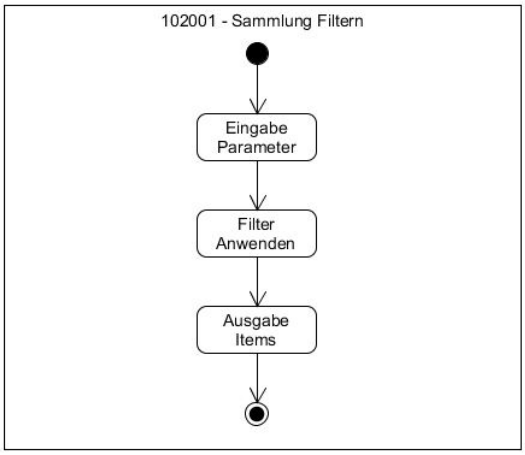

Abbildung 3.6.: Aktivitätsdiagramm: Sammlung anzeigen

3.4.5. Verwalten verliehener Items

[102002] Funktion: Items anzeigen

Der User kann ein Item als verliehen markieren. Wird ein Item so markiert, wird der User aufgefordert die leihende Person und das Datum einzugeben. Bei der Rückgabe des Items wird der Status von verliehen auf nicht verliehen geändert und das Item erscheint nun nicht mehr auf der Liste der verliehenen Gegenständen.
Eine History zeigt alle jemals verliehenen Gegenstände an.

3.4.6. Exportieren aller ausgewählter Items

[102003] Funktion: Items anzeigen

Eine vollständige oder gefilterte Übersicht der Items einer Sammlung kann als CSV File exportiert werden. Das Export File wird auf der Speicherkarte des Smartphones gespeichert. Zusätzlich wird ein Mailprogramm geöffnet mittels welchem man das Export File versenden kann.

3.4.7. Datenbank Administration

[102004] Funktion: Datenbank Administration

Die Datenbank Administration unterstützt folgende drei Aktionen für alle vorhanden Tables.

3. Anforderungsanalyse

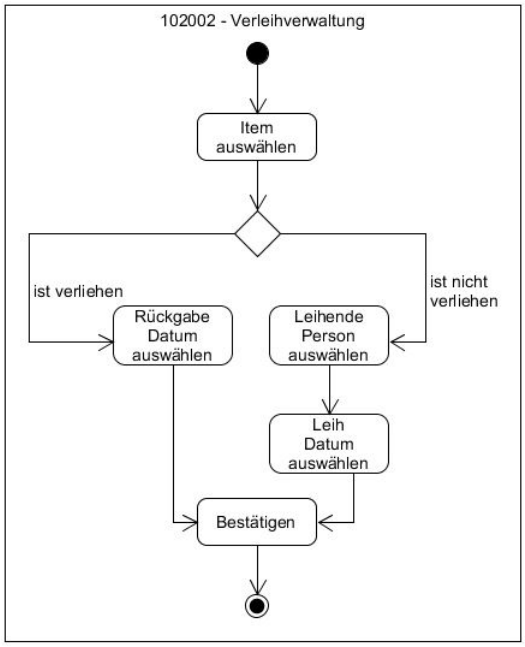

Abbildung 3.7.: Aktivitätsdiagramm: Verwalten verliehener Items

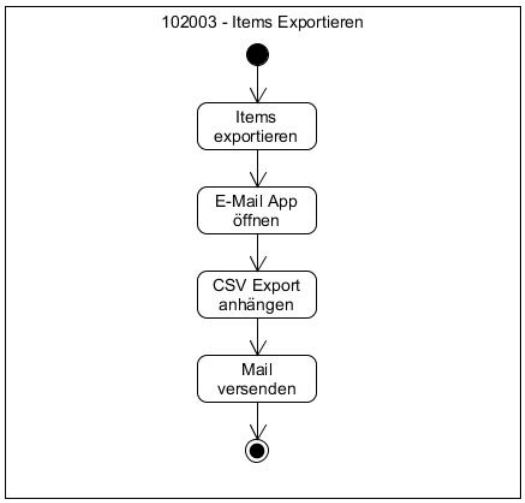

Abbildung 3.8.: Aktivitätsdiagramm: Export der Sammlung

1. Hinzufügen eines Eintrages
2. Entfernen eines Eintrages
3. Löschen der ausgewählten Tabelle

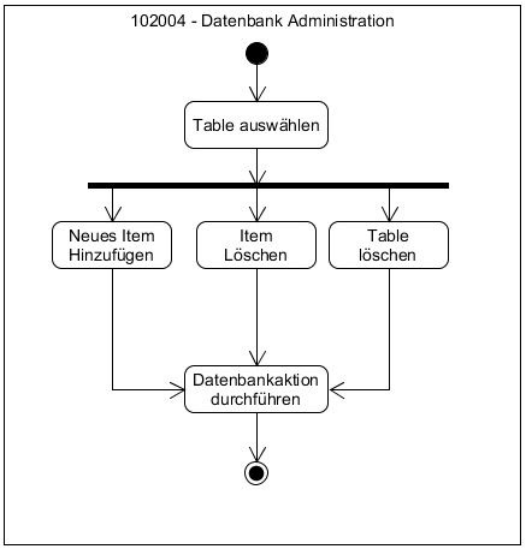

Abbildung 3.9.: Aktivitätsdiagramm: Datenbank Administration

3.4.8. REST Schnittstelle zur Abfrage von Metadaten diverser Online Datenbanken

[103001] Funktion: REST Schnittstelle zur Abfrage von Metadaten

Über eine REST Schnittstelle soll die App verschiedene Online Datenbanken abfragen. Anhand des gesendeten EAN Codes sollen eventuell vorhandene Metadaten zu den Items erfragt werden, so damit der User nicht alle Daten von Hand eingeben muss.

3.4.9. Kamera als Barcodescanner

[104001] Funktion: Kamera als Barcodescanner

Als weitere Eingabehilfe für den User soll die Kamera des Smartphones als EAN Scanner genutzt werden. Sollte diese den Barcode nicht erkennen können, kann der Barcode vom User manuell eingegeben werden.

3.5. Datenbank

[201001] Datenbank: Grunddesign

3. Anforderungsanalyse

Die Anforderungen der App an die Datenbank sind einfach und leicht überschaubar. Eine vollständige Übersicht der Tabellen und Felder ist im Abschnitt 4.2.3 zu finden. Um eine einfach zu erweiternde und leicht zu wartende Datenbank zu schaffen, werden für immer wiederkehrende Daten einzelne Tabellen verwendet.

3.5.1. Übersicht der Tabellen und Felder

Die SQLITE Datenbank soll folgende Daten zu den einzelnen Items speichern.

Allgemein	Bücher	Filme	Spiele
Barcode	Verlag	Studio	Entwickler
Titel	Autor	Speichermedium	System
Medientyp		Regisseur	FSK
Genre		FSK	
Sprache			
Erscheinungsjahr			
Bewertung			
Verleihstatus			

Tabelle 3.1.: Datenbankfelder

Die Verleih Verwaltung soll folgende Felder beinhalten.

- ID des Freundes
- ID des Items
- Start des Leihvorgangs
- Rückgabe des Items

Um die Datenbank möglichst schlank und übersichtlich zu halten werden folgende Tabellen und Felder angelegt.

- Freunde
  ID (Key), Vorname, Nachname
- Autoren
  ID (Key), Autor
- Regisseure
  ID (Key), Regisseur
- Genres
  ID (Key), Genre

- Sprachen
  ID (Key), Sprache
- Verlage
  ID (Key), Verlag
- Studios
  ID (Key), Studio
- Systeme
  ID (Key), System

Die beiden grösseren Tabellen Items und die der Verleih Verwaltung sollen um Speicherplatz zu sparen nur die ID's der einzelnen Daten enthalten.

3.6. GUI

Allgemein wird von der GUI erwartet, dass die App intuitiv bedienbar ist und fehlerfrei auf möglichst vielen Hardware Typen dargestellt werden kann.

Des weiteren ist die GUI Design so angelegt, dass die App jederzeit übersichtlich ein Maximum an Informationen darstellen kann.

3.6.1. Startseite

Die Startseite der App soll dem User ermöglichen auf alle wichtigen Funktionen direkt zuzugreifen. Deshalb besteht diese nur aus folgenden vier Buttons `Neues Item`, `Sammlung`, `Leihwesen` und `Info`.

3.6.2. Barcode Scannen

Dieser Screen startet die eingebaute Kamera und wird automatisch angezeigt, wenn man ein neues Item anlegen will. Sollte das Item keinen Barcode besitzen, kann man von hier aus die Manuelle Eingabe erreichen. Eine zurück zur Startseite 3.6.1 ist ebenfalls möglich.

3.6.3. Manuell Anlegen / Item Bearbeiten

Dieses Screenlayout wird sowohl für das manuelle Anlegen eines Items, als auch für das Bearbeiten eines vorhandenen Items genutzt.

Der einzige Unterschied besteht zwischen den beiden Aktionen darin, das dass beim manuellen Anlegen keine Datenelemente angezeigt werden. Während beim Bearbeiten eines Items, alle bisher gespeicherten Datenelemente angezeigt werden.

Abbildung 3.10.: Startseite

Abbildung 3.11.: Barcode Scannen

3. Anforderungsanalyse

Abbildung 3.12.: Manuell Anlegen / Item Bearbeiten

3.6.4. Detail Ansicht

Hier werden dem User alle Datenelemente eines Items angezeigt. Ausserdem kann in dieser Ansicht das Item bearbeitet, verliehen oder gelöscht werden.

Abbildung 3.13.: Detail Ansicht

3.6.5. Sicherheitsfrage Item Löschen

Wenn der User ein Item aus der Datenbank (Sammlung) löschen will. Erscheint immer bevor der Löschvorgang gestartet wird folgende Sicherheitsfrage.

Abbildung 3.14.: Sicherheitsfrage

3.6.6. Item verleihen

Wird ein Item als verliehen markiert, erscheint eine erweiterbare Liste mit Personen. Aus dieser Liste wählt der User nun die Person aus, welche das Item geliehen hat. Ist die Person noch nicht in der Liste, kann der User diese in die Liste eintragen.

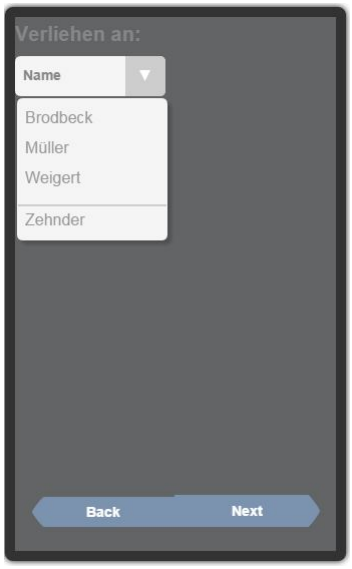

Abbildung 3.15.: Item verleihen

3.6.7. Filter

Die Filteraktivität besteht aus einem Screen mit den drei Reitern `Allgemein`, `Buch & Spiel` und `Film`.

3.6.7.1. Allgemein

Folgende Datenelemente können auf dem Reiter `Allgemein` 3.16 der Filter Activity gefiltert werden.

Barcode, Titel, Typ, Genre, Sprache, Erscheinungsjahr und Verliehen.

Abbildung 3.16.: Filter - Allgemein

3.6.7.2. Buch & Spiel

Folgende Datenelemente können auf dem Reiter `Buch & Spiel` 3.17 der Filter Activity gefiltert werden.

Für Bücher: Verlag, Autor und Auflage Für Spiele: System und FSK

3.6.7.3. Film

Folgende Datenelemente können auf dem Reiter `Film` 3.18 der Filter Activity gefiltert werden.

Studio, Regisseur, DVD, BlueRay und FSK

3. Anforderungsanalyse

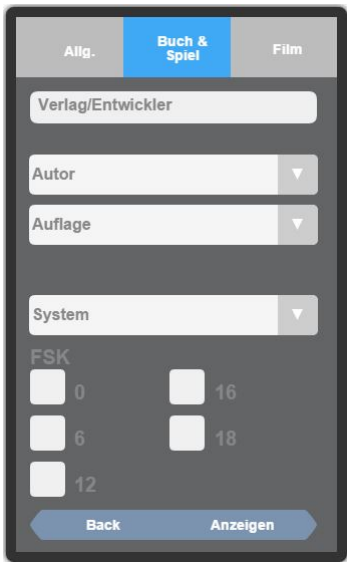

Abbildung 3.17.: Filter - Buch & Spiel

Abbildung 3.18.: Filter - Film

4. Konzept

4.1. Funktionen

Da nach dem Design Review es allen Anschein hat, dass die beiden Funktionen

1. REST Schnittstelle zur Abfrage von Internet Datenbanken
2. Kamera als Barcodescanner

nicht mehr rechtzeitig implementiert werden können. Werde ich an dieser Stelle erst einmal auf die Verleih Verwaltung, das Filtern der Sammlung und den CSV Export der Items eingehen. Sollte am Ende doch noch die fehlenden Funktionen implementiert werden können, werden diese noch nachgetragen.

4.1.1. Verleih Verwaltung

Damit eine schnellen Verwaltung verliehener Gegenstände der Sammlung möglich ist, sollte der User in der Lage sein jedes Item seiner Sammlung schnell als Verliehen markieren. Dazu bedarf es keinen speziellen Librarys. Bei den Abschnitten zur Datenbank und zur GUI kann man erkennen, dass für die Verwaltung eine extra Tabelle in der Datenbank angelegt werden muss und ein Layout, welches dem User die Möglichkeit gibt zu definieren was, wann von wem geliehen oder zurück gegeben wurde.

4.1.2. Filtern der Sammlung

Der Filter soll sich über alle Felder der Tabelle Items erstrecken. Damit es dem User möglichst einfach gemacht wird schnell die gewünschten Items seiner Sammlung zu filtern, wird bei der Titel Eingabe vor und nach dem Text automatisch eine Wildcard gesetzt und im dem Filter entsprechenden SELECT Statement ein LIKE anstelle von „=“ verwendet.

Alle vom User im Filter definierten Parameter werden mit einer UND-Verknüpfung an die Datenbank weiter gegeben.

Um den Aufwand an die Programmierung der App möglichst gering zu halten, wird das Layout für die Anzeige der Sammlung auch für die Ausgabe der vom User gefilterten Sammlung und die Ausgabe der von der Verleih Verwaltung gefilterten Übersicht der verliehenen Items genutzt. Der User kann die Parameter für den Filter im selben Layout wie zur manuellen Eingabe oder Bearbeitung der Items nutzen.

4.1.3. CSV Export der Items

Für den CSV Export der Daten wird eine externe Library (Opencsv) verwendet. Da nicht jedes Android Gerät über eine externe Speicherkarte oder USB Zugriff auf alle App Daten verfügt, wird sich beim

4. Konzept

Export der Items (mit oder ohne Filter) sich das interne E-Mail Programm melden, mittels welchem die CSV Datei schnell und einfach an eine Mail Adresse gesendet werden kann.

4.2. Datenbank

4.2.1. Allgemeines

Um bei grösseren Datenmengen Speicherplatz zu sparen soll bei der grössten Tabelle (Items) weitgehend Integer als Feldtyp verwendet werden. In diesen ID Feldern soll die ID der Informationen aus den entsprechenden Untertabellen gespeichert werden, damit diese später in der App für den User angezeigt werden können und dieser nicht nur eine nicht aussagekräftige Zahl angezeigt bekommt.

Da SQLITE den Datentyp Boolean nicht kennt soll auch hier auf Integer zurück gegriffen werden. Alle solch verwendeten Felder werden nur mit 1 (true) und 0 (false) gefüllt.

Da auch der Datentyp Date in SQLITE nicht existiert soll hier auf String zurück gegriffen werden. Das Jahr wird hierbei als 4-stelliger String gespeichert.

4.2.2. Aufbau der SQLITE Datenbank

Die Datenbank der App hat folgende Tabellen und Felder.

	Typ	Key	Information/Verwendung
Items			Table in welchem alle Informationen eines Items gespeichert werden.
ITEM_ID	Integer	Ja	Id des Items.
EAN	Integer		Barcode des Items.
TITLE	String		Name des Items.
RATING	Integer		Von 0 bis 5 Sterne das persönliche Rating des Users.
MEDIA_TYPE	String		Art des Items: Book, Movie oder Game.
GENRE_ID	Integer		ID des Genres kann für alle Arten von Items verwendet werden.
LANGUAGE_ID	Integer		ID der Sprache des Items.
LAUNCH	String		Erscheinungsjahr des Items.
RENTAL	Integer		Boolean (1 oder 0) welcher den Leihstatus angibt.
PUBLISHER_ID	Integer		ID des Verlegers (nur Bücher).
AUTHOR_ID	Integer		ID des Autors (nur Bücher).
SYSTEM_ID	Integer		ID des Systems (nur Spiele)
DVD	Integer		Boolean welcher angibt ob es sich um eine DVD handelt.
BLURAY	Integer		Boolean welcher angibt ob es sich um eine BluRay handelt.
STUDIO_ID	Integer		Id des Studio (Film und Spiel).
DIRECTOR_ID	Integer		Id des Regisseur (nur Film).

PARENTAL	Integer		Altersfreigabe des Items (Film und Spiel).
REMARKS	String		Bemerkungen des Users.
History			Tabelle mit der Verleih Historie.
HISTORY_ID	Integer	Ja	ID des Verleih Vorgangs.
ITEM_ID	Integer		ID des verliehenen Items.
FRIEND_ID	Integer		ID des Freundes welcher das Item geliehen hat.
START	String		Datum an welchen das Item verliehen wurde.
BACK	String		Datum an welchem das Item wieder zurück gegeben wurde.
Friends			Tabelle in welchem alle Freunde hinterlegt sind.
FRIEND_ID	Integer	Ja	ID des Freundes.
FIRST_NAME	String		Vorname des Freundes.
LAST_NAME	String		Nachname des Freundes.
Authors			Tabelle mit allen Autoren.
AUTHOR_ID	Integer	Ja	ID des Autors.
AUTHOR	String		Name des Autors.
Directors			Tabelle mit allen Regisseuren.
DIRECTOR_ID	Integer	Ja	ID des Regisseur.
DIRECTOR	String		Name des Regisseur.
Genres			Tabelle mit allen Genres.
GENRE_ID	Integer	Ja	ID des Genres.
GENRE	String		Genre
Languages			Tabelle welche alle Sprachen enthält.
LANGUAGE_ID	Integer	Ja	ID der Sprache
LANGUAGE	String		Sprache
Publishers			Tabelle welche alle Verlage enthält.
PUBLISHER_ID	Integer	Ja	ID des Verlags.
PUBLISHER	String		Name des Verlags.
Studios			Welche alle Studios enthält.
STUDIO_ID	Integer	Ja	ID des Studios.
STUDIO	String		Name des Studios.
Systems			Tabelle welche alle Systeme enthält.
SYSTEM_ID	Integer	Ja	ID des Systems.
SYSTEM	String		Name des Systems.

4.2.3. Übersicht Datenbank

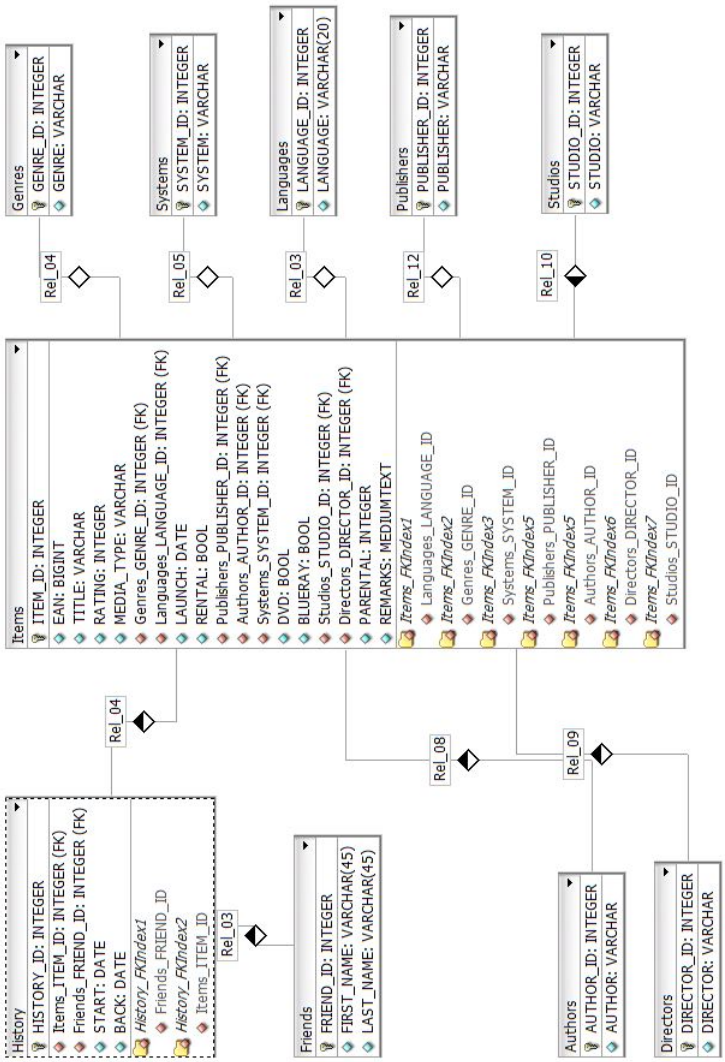

Abbildung 4.1.: Überblick Datenbank

4.3. GUI

4.3.1. Allgemeines

Damit die App auf möglichst vielen Android Geräten, mit verschiedener Bildschirmgrösse, mit einer gleichbleibend hohen Useability funktioniert. Wird beim Layout der App auf folgende Punkte wert gelegt.

Die App soll eine einfache und Flache Hierarchie bieten, in welcher der User sich schnell und vor und zurück bewegen kann.
Zusätzlich sollen alle GUI Elemente einen relativen Bezug zueinander und zur Bildschirmgrösse haben. So das diese immer an der selben Position den maximalen zur Verfügung stehenden Platz ausnutzen.
Um die App schlank und leicht wartbar zu halten, wird wo es geht das Layout für verschiedene Zwecke verwendet. Schaltflächen die bei solch einer Mehrfachverwendung, im speziellen Fall keine Funktion haben, werden automatisch ausgeblendet.
Da jedes Android Gerät einen Hardware Zurück Knopf besitzt, ist die GUI der App darauf angelegt das der User diesen Benutzt und stellt selbst keinen Zurück Knopf zur Verfügung. An stellen wo der User eine Handlung abschliesst wird die App automatisch auf das vorherige Layout zurück springen.

4.3.2. Hauptmenu / Startscreen

Damit der User einfach und schnell Zugriff auf alle wichtigen Funktionen der App erhält befinden sich nur fünf Knöpfe auf dem Start- bzw. Hauptlayout. Vier sind davon am oberen Bildschirmrand angeordnet und einer am unteren Bildschirmrand.

Von oben nach unten handelt es sich um Knöpfe mit den folgenden Funktionen.

- Neues Item
- Sammlung
- Leihwesen
- Export Database
- Info

4.3.3. Neues Item

In der App Version in welcher alle Anforderungen umgesetzt sind, wird sich hier die Kamera des Gerätes öffnen und als Barcode Scanner den EAN Code des Items einscannen. Anschliessend wird dieser EAN Code genutzt um über die REST Schnittstelle mit diversen Online Datenbanken eventuell vorhandene Metadaten zu erhalten. Diese werden dann im Layout für die manuelle Eingabe oder das manuelle

4. Konzept

Bearbeiten eines Items angezeigt und können vom User bearbeitet oder in der Datenbank gespeichert werden.

Dieses Layout für die manuelle Eingabe (bzw. für das Bearbeiten eines Items) ist wie folgt von oben nach unten aufgebaut.

- EAN Code: Text Eingabezeile, welche nur Nummern anbietet und akzeptiert.
- Titel: Text Eingabezeile welche eine alphanumerische Eingabe akzeptiert.
- Rating Sterne: 5 Sterne, welche das persönliche Rating des Users wiedergeben. Es können nur ganze Sterne ausgewählt werden.
- Typ Auswahl: Hier kann der User auswählen ob es sich um ein Buch, Film oder Spiel handelt. Es ist immer nur ein Typ möglich.
- Genre DropDown: Liste mit allen Genres aus der entsprechenden Untertabelle.
- Sprache DropDown: Liste mit allen Sprachen aus der entsprechenden Untertabelle.
- Jahr DropDown: Liste mit den Jahren 1899 bis 2020
- Verliehen CheckBox: Hier kann ein Item bereits als Verliehen markiert werden.
- Verlag DropDown: Liste mit den Verlagen aus der entsprechenden Untertabelle.
- Autor DropDown: Liste mit den Autoren aus der entsprechenden Untertabelle.
- System DropDown: Liste mit den Systemen aus der entsprechenden Untertabelle.
- DVD und BluRay CheckBox: Hier kann das Item als DVD und/oder BluRay ausgewählt werden.
- Studio DropDown: Liste mit den Film- und Entwickler Studios aus der entsprechenden Untertabelle.
- Regisseur DropDown: Liste mit den Regisseuren aus der entsprechenden Untertabelle.
- Altersfreigabe CheckBox: Zeile mit CheckBoxen der möglichen Altersfreigabe.
- Hinzufügen Knopf: Erscheint nur bei einem neuen Item und speichert die oben gegebenen Informationen in der Datenbank.
- Update Knopf: Erscheint nur wenn ein bereits bestehendes Item bearbeitet wurde und speichert die Änderungen in der Datenbank.
- Export Database Knopf: Exportiert die Items der Datenbank und nutzt die oben eingegeben Informationen als Filter für den Export.

- Search Knopf: Führt mit den oben eingegeben Informationen eine Suche über alle Items der Sammlung durch und gibt das Ergebnis der Suche als Liste aus.

Um die Vielzahl der Elemente übersichtlich Darzustellen, wird hier der ScrollView (vertikal) verwendet.

4.3.4. Sammlung

Diese Layout besteht aus einer einfachen vertikalen Liste, welche die Komplette Sammlung anzeigt. Sobald der User ein Item auswählt öffnet sich das Layout mit den Detailinformationen.

Dieses Layout wird auch genutzt um das Ergebnis einer Datenbanksuche darzustellen.

4.3.5. Detail Ansicht

Damit der User alle Informationen auf einem Bildschirm sieht, wird hier der Android ScrollView (vertikal) verwendet. Es handelt sich hier um ein reines Ansicht Layout, welches nur folgende drei Aktionen des Users zulässt.

1. Verliehen: Ein Schalter, welcher die Verleih Verwaltung anzeigt.
2. Bearbeiten Knopf: Öffnet das Layout 4.3.3 mit den Informationen des Items um diese zu bearbeiten.
3. Remove Knopf: Öffnet ein Dialog Fenster, welches die Bestätigung für das Löschen des Items aus der Datenbank anfordert.

Ansonsten werden Analog der Item Anlage oder Bearbeitung alle verfügbaren Informationen angezeigt, welche hier aber nicht vom User geändert werden können.

4.3.6. Leihwesen

Dieses Layout besteht aus einer einfachen vertikalen Liste, welche alle noch nicht abgeschlossenen Verleih Vorgänge anzeigt. Wählt der User einen Eintrag der Liste aus, öffnet sich am unteren Bildschirmrand ein kleines Dialogfenster, welches dem User anzeigt, wer das Item seit wann ausgeliehen hat.

4.3.7. Verleih Verwaltung

Hierbei handelt es sich um ein Layout, dass auf einer Bildschirmseite von oben nach unten dem User dem User folgende Auswahlmöglichkeiten beziehungsweise Informationen bietet.

- Freund DropDown: Liste welche alle Freunde der Untertabelle darstellt.
- Textfeld mit Item Name und der Information ob das Item gerade verliehen zurück gegeben wird.

- Kalender Auswahlfeld, welches das aktuelle Datum anzeigt.
- Ok Knopf

Dabei ist zu beachten, das je nachdem, ob das Item verliehen wird oder wieder zurück gegeben das Layout entweder die bestehenden Daten des aktuellen Leihvorgangs anzeigt oder einfach nur das Item vorgibt und der User noch den Freund auswählen muss, welchem er das Item verleiht.

Die Funktion des Ok Knopfes variiert, je nachdem ob das Item verliehen wird oder zurück gegeben wird, schreibt er das Datum in die entsprechende Spalte der History Tabelle.

4.3.8. Export Database

Die App öffnet eine Auswahl an verfügbaren Mailprogrammen auf dem Gerät. Sobald der User eines ausgewählt hat wird eine Standard Mail mit einem CSV File als Attachment erstellt. Der User muss nun nur noch den Adressaten eingeben und kann die komplette Items Tabelle als CSV Export File versenden.

4.3.9. Info

Dieses Layout zeigt am oberen Bildschirmrand die installierte Version der App und den Programmierer an. Am unteren Bildschirmrand, ist ein Knopf welcher den User die Einstellungen (Datenbank Administration) anzeigen lässt.

4.3.10. Settings

Um bei möglichst guter Übersicht dem User eine Vielzahl von Möglichkeiten zu bieten, wird in diesem Layout die ScrollView (vertikal) angewendet. Das Layout ist grob in zwei Bereiche unterteilt. In der oberen Hälfte kann der User die Daten der einzelnen Untertabellen anzeigen lassen und einzelne Informationen zu den Tabellen hinzufügen oder löschen.

Wählt man die Verwaltung einer dieser Untertabellen so bekommt der User das Datenbankverwaltung Layout (s.h. 4.3.11 Datenbankverwaltung) angezeigt. Zwei Ausnahmen bestehen, welche die Untertabellen `Friends` und `History` betreffen.
In der unteren Hälfte können alle Untertabellen gelöscht, beziehungsweise in den Urzustand versetzt werden. Wählt der User einen dieser Knöpfe öffnet sich aus Sicherheitsgründen ein Dialogfenster, in welchem der User den Vorgang nochmals bestätigen muss.

4.3.11. Datenbankverwaltung

Der Grossteil des Bildschirms wird von einer Liste eingenommen, welche die aktuell in der Untertabelle gespeicherten Daten anzeigt. Am unteren Bildschirmrand gibt es von oben nach unten den Knopf

`Remove`, eine Eingabezeile und einen `Add` Knopf.

Will der User einen Eintrag der Untertabelle löschen, wählt er diesen in der Liste aus und drückt den `Remove` Knopf. Das Layout wird über ein Dialogfenster den User auffordern das löschen des Eintrages zu bestätigen.

Um einen neuen Eintrag in die Untertabelle vorzunehmen, kann der User die entsprechende Information in die Eingabezeile eingeben und den Knopf `Add` drücken. Eine Abfrage prüft nach ob der User überhaupt etwas eingegeben hat und speichert diese Eingabe in die entsprechende Untertabelle.

4.3.11.1. Freunde Verwalten

An sich ist das Layout, welches die Eintrage der Friends Tabelle verwaltet, identisch mit den Layouts der anderen Untertabellen Verwaltungen. Da der User aber seine Freunde mit Vor- und Nachnamen eingeben kann existieren anstelle der einen Eingabezeile in diesem Layout zwei Eingabezeilen. Alle anderen Funktionen sind analog zur allgemeinen Datenbank Verwaltung.

4.3.11.2. History

Da die Tabelle der Verleih Verwaltung grösser ist als die der anderen Untertabellen und ausserdem die Verleih Verwaltung über das Layout der Item Details und des Leihwesens erfolgt. Zeigt der Knopf für die Verwaltung der Tabelle nur eine Liste mit allen abgeschlossenen und noch laufenden Leihaktionen an.
Wählt man einen Eintrag aus, öffnet sich ein kleines Dialogfenster mit Detailinformationen.

5. Umsetzung

Um die komplette Umsetzung im Detail zu zeigen, fehlt der Platz und die Zeit. Zusätzlich würde dies, das vorliegende Dokument extrem Unübersichtlich machen. Deshalb wird im folgenden die Funktionsweise und Umsetzung an einigen typischen Beispielen erklärt. Alle im Laufe der Semesterarbeit erstellten Dateien inklusive des kompletten Quellcode kann man auf Github[1] finden.

Wer sich den Quellcode genauer ansieht, wird entdecken, das von Anfang an ein Boolean (debugMode) übergeben wird. Diesen Boolean habe ich genutzt um an vielen Stellen einen Eintrag ins Logfile zu generieren, welcher es mir ermöglicht hat ohne aus der laufenden App auszusteigen den Verlauf der Variablen und Objekte jederzeit nachzuvollziehen.

5.1. Funktionen

An dieser Stelle hätte ich gerne mehr gezeigt. Dieser Abschnitt sollte zwei für die Useability der App entscheidende Funktionen beinhalten.

- Die Kamera als Barcodescanner.
- Eine REST Schnittstelle zur Metadaten Abfrage im Internet.

Wie es sich bei unserem Design Review abgezeichnet hatte, konnte ich diese beiden in der Anforderung und im Konzept beschrieben Funktionalitäten leider nicht mehr realisieren.

5.1.1. Filtern der Sammlung

Technisch habe ich zwei bereits eingesetzte Klassen verwendet um eine einfache und leicht zu erweiternde Filterfunktion in der App zu realisieren. Ich habe die Klasse `DatabaseOperations` um eine Methode erweitert, welche mir auf ein übergebenes Objekt der Klasse Item ein SELECT Statement liefert zum finden der Parameter des Items.

Listing 5.1: Methode: getSelectStatement

```
1  // Method which returns a string with an SQL select statement from given
   // item data
3      public String getSelectStatement(Item item) {

5          String whereClause = "SELECT " + DatabaseInfo.ITEMS_ID_COL +
                   " FROM " + DatabaseInfo.ITEMS_TABLE + " WHERE ";
7
           if (item.getEAN() > 0) whereClause += DatabaseInfo.ITEMS_EAN_COL +
9                  "=" + Long.toString(item.getEAN()) + " AND ";
```

[1] https://github.com/MWeigert/Collector

```
                .
11              .
                .
13              .
                .
15          if (item.isDvd()) whereClause += DatabaseInfo.ITEMS_DVD_COL +
                "=1 AND ";
17          if (item.isBluRay()) whereClause += DatabaseInfo.ITEMS_BLURAY_COL +
                "=1 AND ";
19          if (!item.getStudio().equals("STUDIO")) whereClause +=
                DatabaseInfo.ITEMS_STUDIO_ID_COL + "=" + Integer
21              .toString(item.getStudio_id()) + " AND ";
            if (!item.getDirector().equals("DIRECTOR")) whereClause +=
23              DatabaseInfo.ITEMS_DIRECTOR_ID_COL + "=" + Integer
                .toString(item.getDirector_id()) + " AND ";
25          if (item.getFsk() != 42)
             whereClause += DatabaseInfo.ITEMS_PARENTAL_ID_COL + "=" +
27              Integer.toString(item.getFsk()) + " " + "AND ";
            whereClause = whereClause.substring(0, whereClause.length() - 4) +
29              ";";
            return whereClause;
31      }
```

Dazu habe ich wie im Listing 5.1 ersichtlich ist, die Klasse Item als Container für die vom User eingegeben Parameter genutzt und aus diesen wird ein SELECT Statement zusammengesetzt. Pro Parameter wird ein Substring mit einer AND Verknüpfung angehängt. Zum Schluss entferne ich noch das letzte AND und schliesse das SELECT Statement mit „;“.

Der String des zurück gegeben SELECT Statement kann nun einfach von einem Intent zum anderen weiter gegeben werden. Im Anzeige Layout der Sammlung können nun entweder alle oder nur die Items der Sammlung in der Liste dargestellt werden, welche dem SELECT Statement entsprechen.

5.1.2. Export der Daten

Um die Daten der SQLite Datenbank zu exportieren habe ich die frei Verfügbare Library Opencsv (Version 3.7) eingebunden. Der Export erfolgt auf zwei verschiedene Arten

- die vollständige Tabelle Items
- die gefilterte Tabelle Items

und wird von der Methode `exportDatabaseCSV` aus Listing 5.2 ausgeführt.

Listing 5.2: Methode: exportDatabaseCSV

```
// Export method of the database
public void exportDatabaseCSV(DatabaseOperations dop, String
        whereClause) throws IOException {

        SQLiteDatabase db = dop.getReadableDatabase();
        File exportDir = new File(Environment
                .getExternalStorageDirectory(), "");
        if (!exportDir.exists()) {
                exportDir.mkdirs();
        }

        File file = new File(exportDir, "collector.csv");
        try {
                file.createNewFile();
                Log.d("DOP", "File: " + file);
                CSVWriter csvWrite = new CSVWriter(new FileWriter(file));
                Cursor curCSV = null;
                if (whereClause != null) {
                        curCSV = db.rawQuery(whereClause, null);
                } else {
                        curCSV = db.rawQuery("SELECT * FROM " +
                                DatabaseInfo.ITEMS_TABLE, null);
                }
                csvWrite.writeNext(curCSV.getColumnNames());
                while (curCSV.moveToNext()) {
                //Which column you want to export
                String arrStr[] = {curCSV.getString(0), curCSV.
                        getString(1), curCSV.getString(2), curCSV
                        .getString(3), curCSV.getString(4), curCSV
                        .getString(5), curCSV.getString(6), curCSV
                        .getString(7), curCSV.getString(8),     curCSV
                        .getString(9), curCSV.getString(10), curCSV
                        .getString(11), curCSV.getString(12), curCSV
                        .getString(13), curCSV.getString(14), curCSV
                        .getString(15), curCSV.getString(16), curCSV
                        .getString(17)};
                        csvWrite.writeNext(arrStr);
                }
```

```
39                  csvWrite.close();
                    curCSV.close();
41              } catch (Exception sqlEx) {
                Log.e("MainActivity", sqlEx.getMessage(), sqlEx);
43              }
}
```

Dieser Methode kann neben dem Datenbank Object ein String mit einem SELECT Statement übergeben werden und je nachdem ob ein String übergeben wird oder nicht schreibt die Methode die gefilterte oder ungefilterte Sammlung an Items in den externen Gerätespeicher. Nach erfolgreichem Export startet die Klasse welche den Export aufgerufen hat eine Intent, welcher das lokale E-Mail Programm ausführt und diesem die gespeicherte CSV Datei als Attachment übergibt.

Aufgerufen wird die Methode entweder über das Hauptmenu (Export aller Items) oder über die Filteransicht (Export der Items, welche den Filterkriterien genügen).

5.2. Datenbank

Das Handling der Daten in der App ist wie folgt strukturiert. Wie im Abschnitt 5.3 dargestellt folgt die App dem MVC Pattern. Damit die Kommunikation und Verwaltung der Daten im Modell möglichst Übersichtlich und leicht zu Warten, beziehungsweise zu erweitern ist wurden folgende drei Java Klassen programmiert.

- DatabaseInfo
- DatabaseBase
- DatabaseOperations

Diese Teilen sich die Aufgaben in der Verwaltung der Daten auf.

Die Klasse DatabaseInfo enthält alle Informationen zur Datenbank, den Tabellen und Feldern. Die Klasse besteht einzig und allein aus Variablen Deklarationen, welche die Namen der Tabellen und Felder enthalten. Zusätzlich gibt es für jede Tabelle eine Variable `CREATE_XYZ`[2] welche das komplette CREATE TABLE Kommando der SQLite Datenbank enthält.

Die Klasse DatabaseBase ist ebenfalls eine Klasse welche nur Variablen deklariert. Hierbei handelt es sich um String Arrays, welche für die Untertabellen einen Basis Datensatz zur Verfügung stellen. In der App erfüllt diese Klasse zwei Funktionen.

[2] XYZ ist ein Platzhalter für den Tabellen Namen

1. Sie stellt dem User bereits eine Grundlage von Daten der Untertabellen zur Verfügung, so dass dieser nicht alles selbst erfassen muss.

2. Sie hat das Testing der Funktionen im Bereich Datenbank der App erleichtert, da bei einer kompletten Neuinstallation so schnell Daten verfügbar waren.

Die Dritte Verwendete Klasse DatabaseOperations ist der Controller, welcher zwischen dem Controller der App und dem Modell sitzt. Hier werden sämtliche SQL Befehle auf die Datenbank angewendet. Der Befehlsumfang geht vom erstellen der Datenbank und den Tabellen über hinzufügen und entfernen von Daten, bis hin zu SELECT Abfragen, welche die benötigten Informationen aus der Datenbank an den Controller und somit an die View zurück liefern.

Listing 5.3: Hinzufügen von Daten zur Tabelle

```
// Method to add a new entry in authors table
    public void addAuthor(DatabaseOperations dop, String author) {

        SQLiteDatabase db;
        db = dop.getWritableDatabase();
        ContentValues values = new ContentValues();

        values.put(DatabaseInfo.AUTHORS_AUTHOR_COL, author);
        db.insert(DatabaseInfo.AUTHORS_TABLE, null, values);

        if (debugMode) {
            Log.d("DOP", "Table authors --> added " + author + ".");
        }
    }
```

Im Listing 5.3 ist zu sehen wie die Klasse DatabaseOperations ein Datenbank Objekt (Zeile 5) erstellt und die Übergebenen Daten in die entsprechende Tabelle schreibt (Zeile 9).

Listing 5.4: Hinzufügen von Daten zur Tabelle

```
// Method to return all entries in authors table. Returns a Cursor with
// all authors.
    public Cursor getAuthors(DatabaseOperations dop) {

        if (debugMode) {
            Log.d("DOP", "Starting to receive all entries from authors" +
                                "table.");
        }
```

```
10          SQLiteDatabase db;
            db = dop.getReadableDatabase();
12          String[] authors = {DatabaseInfo.AUTHORS_AUTHOR_COL};

14          return db.query(DatabaseInfo.AUTHORS_TABLE, authors, null, null,
                        null, null, null);
16      }
```

Eine häufig genutzte Methode liefert alle Einträge einer Tabelle, Listing 5.4 zeigt dies anhand der Untertabelle Authors. In den Zeilen 14 & 15 wird ein `SELECT * FROM authors` ausgeführt und die komplette Tabelle in einen Cursor geschrieben und an den Controller zurück gegeben, damit dieser die Daten darstellen oder weiter verarbeiten kann.

5.3. GUI

Die GUI, der App wird gemäss dem MVC Pattern (Abb. 5.1) umgesetzt. Der Aufbau von Android unterstützt den Entwickler sich an dieses Pattern zu halten.

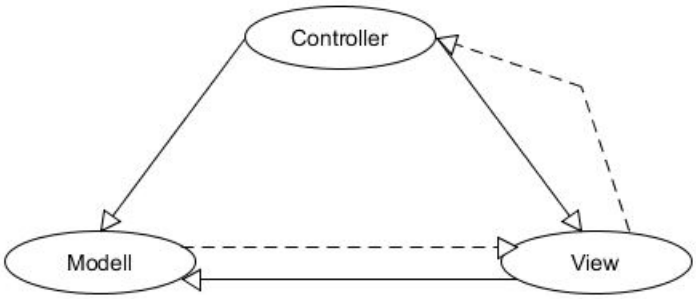

Abbildung 5.1.: Modell, View & Controller Pattern

Die Struktur, welche einem von Android vorgegeben wird sieht bei der App wie folgt aus:

- Controller - Java Files welche die Funktion beinhalten und die Kommunikation zwischen View und Modell steuern.
- View - XML Files welche ausschliesslich das Aussehen der App definieren.
- Modell - SQLite Datenbank, welche die Daten für die App beinhaltet.

Wie das Pattern umgesetzt wurde anhand einzelner Ausschnitte aus dem Code der App. Auf einen Ausschnitt aus der SQLite Datenbank (dem Modell), wird an dieser Stelle verzichtet. Die Verwendete Datenbank Struktur zeigt die Abbildung 4.2.3. Sehr schön sieht man das Zusammenspiel zwischen Controller und View an der Zeile 3 & 4 im Ausschnitt aus dem Button Listener 5.5 und der Zeile 2 im

5. Umsetzung

Ausschnitt des XML Files 5.6.

Hier sieht man im Controller eine Switch Verzweigung, welche anhand der übergebenen Button Id (btn_mng_author) eine Aktion ausführt. Welche ein neues Layout startet in welchem die Daten aus dem Modell 5.7 angezeigt.

Listing 5.5: Controller: Button Listener

```
//Button listener for the database activity
    public void buttonOnClick(View v) {
        switch (v.getId()) {
            case R.id.btn_mng_author:
                //Button author pressed by user.
                if (debugMode) {
                    Log.d("USERACTION", "Entering author administration.");
                }
                final Intent authorIntent = new Intent(this,
                                            TableActivity.class);
                authorIntent.putExtra("debugMode", debugMode);
                authorIntent.putExtra("tableName", "AUTHOR");
                startActivity(authorIntent);
                break;
```

Listing 5.6: View: Button im XML Layout

```
<Button
                android:id="@+id/btn_mng_author"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:onClick="buttonOnClick"
                android:text="@string/btn_mng_author"/>
```

Listing 5.7: Controller: Daten aus Modell in View

```
// Fill authors from database to list item
        String author;
        final ArrayList<String> authors = new ArrayList<String>();
        Cursor authorCrs = db.getAuthors(db);
        anz = authorCrs.getCount();

        if (debugMode) {
                Log.d("TABAC", "DATABASE: " + anz.toString() +
```

```
                " authors in table.");
        }
        if (anz > 0) {
                authorCrs.moveToFirst();
                do {
                        int index = authorCrs.getColumnIndex(
                        DatabaseInfo.AUTHORS_AUTHOR_COL);
                        if (debugMode) {
                                Log.d("TABAC", "Index (AUTHORS_AUTHOR): "
                                + index);
                        }
                        author = authorCrs.getString(index);
                        if (debugMode) {
                                Log.d("TABAC", "DATABASE: Get " + author);
                        }
                        authors.add(author);
                        if (debugMode) {
                                Log.d("TABAC", "Size of authors = "
                                + authors.size());
                        }
                } while (authorCrs.moveToNext());

                if (debugMode) {
                        Log.d("TABAC", "Putting data in Adapter");
                }
                final ArrayAdapter adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1, authors);

                LIST.setAdapter(adapter);
        }
```

Im Kapitel Anforderung im Abschnitt 3.6 sind noch einige Design Anforderungen dokumentiert, bei welchen ich von einer App mit Bildschirmtabs ausgegangen bin. Dies wurde in einer ersten lauffähigen Alpha Version der App so realisiert. Durch meine dabei gewonnenen Erfahrungen, was die Programmierung und das User Handling der App angeht. Unterstützt von den Ergebnissen einer kleinen Internet Recherche, habe ich alle Design Entscheidungen zu Gunsten einer vertikal scrollenden ScrollView abgeändert. Die Vorteile bei der nun eingesetzten Layout Technik sind.

- Modularer und Übersichtlicher Aufbau des Layout.
- Skaliert automatisch auf jeden Bildschirm.

- Leicht zu erweitern.
- Geringerer Aufwand beim Controller der View und damit weniger Fehler anfällig.

5.4. Fazit

Die Semesterarbeit hat mir viel Freude bereitet und ich kann sagen, dass die Erfahrungen, welche ich auf dem Weg der Fertigstellung gesammelt habe mich wachsen haben lassen. Ganz klar muss ich eingestehen, dass meine mangelnde Erfahrung in der Software Entwicklung, vor allem im Bereich Android Applikationen, es mir deutlich schwerer gemacht hat, als ich am Anfang dachte und geplant hatte.

Viele in der Planungsphase scheinbare einfache User Stories haben mich letztendlich in der Implementierung ein vielfaches der dafür vorgesehenen Zeit gekostet. Auf der anderen Seite, hat die Arbeit an der App geholfen einige mir bisher nur theoretisch bekannte Arbeitsweisen praktisch nahe zu bringen. Nun am Ende der Entwicklung, bemerke ich eine deutliche Zunahme, was mein Wissen betreffend der Software Entwicklung für Android Geräte angeht.

Würde ich mit meinem heutigen Wissen die gleiche Aufgabe nochmals angehen, dann würde ich einiges anders machen und wäre diesmal sicher, dass ich in der mir zur Verfügung stehenden Zeit alle Anforderungen fristgerecht umsetzen könnte.

Vor allem im Bereich der Objekt Orientierten Programmierung haben sich mir, leider erst recht spät im Projekt, einige grundlegende Mechanismen erschlossen, welche mir in der Theorie bekannt waren, ich aber bisher nicht verinnerlicht hatte. Dies kann man sicher am Code sehen, der am Anfang noch sehr Prozedural ist und erst gegen Ende Objekt Orientierte Ansätze zeigt.

Durch diese anfänglichen Schwierigkeiten, habe ich mir viel unnötige Arbeit gemacht und unzählige überflüssige Zeilen Code generiert.

Trotz all den Schwierigkeiten und den Abstrichen an der Umsetzung der Anforderungen, muss ich gestehen bin ich darauf stolz das ich am Ende, auch dank der einmaligen Fristverlängerung, eine stabile lauffähige App erstellt habe, die zwar nicht alle gewünschten Features beinhaltet aber so hoffe ich die minimal Anforderung dieser Arbeit erfüllt.

A. Anhang

A.1. Konventionen

Für die Beschreibung von Eingaben oder Beschriftungen in der App wird folgendes Format benutzt.
`Beispiel einer Eingabe durch den User oder Beschriftung in der App.`

A.2. Verwendete Tools & Software

A.2.1. Dokumentation & Präsentation

Alle Dokumente und Präsentationen wurden in LaTeX geschrieben. Dazu wurde TeXstudio in der Version 2.10.4 genutzt.

A.2.2. Programmierung

Die App wurde im Android Studio 1.5.1 von Google erstellt.
Zusätzlich wurde für den CSV Export die Library Opencsv in der Version 3.7 genutzt.

A.2.3. Code und Versionsverwaltung

Für die Versionsverwaltung aller Dokumente und des Programmcode wurde auf GitHub ein Repository eingerichtet.

A.2.4. Datenbank

Für die Erstellung des Datenbank Design wurde der DB Designer 4 von fabFORCE.net verwendet.

A.2.5. GUI

Für die Funktion und das Design der GUI wurde das Webtool Fluid genutzt.

A.2.6. UML

Sämtliche UML wurden mit dem freien Tool Umlet erstellt.

A.3. Design Entscheidungen

ID	Datum	Autor	Bemerkung
001	12.03.16	M. Weigert	Hauptsprache für App wird Englisch
002	12.03.16	M. Weigert	Es wird auf jeden Zurück/Back Button verzichtet.
003	25.03.16	M. Weigert	Extra Tabelle Parental um internationale Jugendfreigaben zu managen.
004	09.04.16	M. Weigert	Verzicht auf die Tabelle Parental um die App nicht unötig zu komplizieren.

Literaturverzeichnis

[1] Sylvain Hébuterne. My books. https://play.google.com/store/apps/details?id=com.shebuterne.maBibliotheque. 1.1

[2] Metosphere. Meine bücher pro. https://play.google.com/store/apps/details?id=com.metosphere.book. 1.2

[3] Emílio Simoes. Codex. https://play.google.com/store/apps/details?id=com.iris.codex. 1.3

[4] olbuappdev. Movie collection. https://play.google.com/store/apps/details?id=de.olbu.android.moviecollection. 1.4

[5] Netwalk. Movielicious. https://play.google.com/store/apps/details?id=be.netwalk.movies. 1.5

[6] Hentie Studios. Nintendo collection. https://play.google.com/store/apps/details?id=com.hentie.neslistdonate. 1.6

[7] Sort It! Apps. Video games manager collector. https://play.google.com/store/apps/details?id=com.hentie.neslistdonate. 1.7