▲ Hochschule Harz
Hochschule für angewandte Wissenschaften

Dokumentation

Dennis Naujokat

The Bibliotheca

Android Programmierung in Java

SoSe 2020

Abgabe:
04. Oktober 2020

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Jeder der etwas sammelt kennt das Problem. Manchmal ist es echt schwer den Überblick zu behalten, was man bereits hat und was einem noch fehlt. Gerade auch im Bereich von Büchern, DVDs, Mangas und ähnlichem, wo man meist nicht alle Teile einer Reihe zusammenhängend kauft oder kaufen kann, weil diese zum Beispiel noch nicht erschienen sind.

Dieses Problem soll von dieser App in Angriff genommen werden. Dabei soll die App die Möglichkeit bieten verschiedene Sammlungen in Form von Bibliotheken zu verwalten. Einzelne Stücke, wie zum Beispiel Bücher, (in folgenden als Exemplare bezeichnet) sollen aufgenommen und bei Bedarf auch zu Reihen bzw. Serien zusammengefasst werden können. Zudem sollen die Exemplare auch mit einem Status versehen werden können, um zu zeigen welche man bereits besitzt, welche einem noch fehlen oder welche man evtl. auch schon vorbestellt hat.

Das Programm soll als App für Android entwickelt werden, damit die Nutzer immer und überall die Daten einsehen können. Zum Beispiel, wenn Sie gerade in der Stadt sind und überlegen, ob Sie sich ein neues Buch kaufen sollten. Es wurde sich speziell für Android entschieden, da so die größte Marktabdeckung gewährleistet werden.

2 Grundlagen / Technologien

Als Programmiersprache zur Entwicklung der App wurde sich für Java entschieden. Zum einen ist dies die meistverwendete Programmiersprache für Android-Apps und zum anderen ist Java in der Theorie auch Plattformunabhängig, wodurch Teile des Codes auch bei einer Portierung auf andere Systeme evtl. wiederverwendet werden können. In diesem Kapitel werden zusätzlich noch einige Grundlegende Konzepte erklärt, die zur Entwicklung der App genutzt wurden.

2.1 Navigation Drawer

Der „Navigation Drawer“ ist eine Designvorlage von Android Studio. Sie stellt ein aufklappbares Navigationsmenü an der linken Bildschirmseite zur Verfügung und die eigentlichen Inhaltsseiten können als Fragments einfach über einen Navigationscontroller ausgetauscht werden.

Das Layout besteht aus mehreren XML-Dateien. Die Stammdatei ist die „activity_main.xml“. Sie inkludiert zum einen als Inhaltsseite die „app_bar_main.xml“ und stellt zum anderen die das Navigationselement als „NavigationsView“ zur Verfügung. Das Inhaltslayout besteht wiederum aus der Toolbar und der „conten_main.xml“, welche wiederum dann das eigentliche Fragment enthält. Diesem ist als Navigationsgraph die „mobile_navigation.xml zugewiesen“, in der dann die Verweise auf die anwählbaren Fragmente stehen. Dabei kann diese auch mehr Fragmente enthalten als das Navigationsmenü bereitstellt, diese können dann über den Programmcode geladen werden.

Das Navigationselement besteht aus einem Header, der in der „nav_header_main.xml“ definiert ist und zum anderen aus dem eigentlichen Menü, welches unter „menu“ in der „activity_main_drawer.xml“ definiert ist. In dem Menü können die einzelnen Navigationspunkte angegeben werden. Die IDs müssen dabei mit denen übereinstimmen, die Navigationsgraphen definiert wurden, um die Menüpunkte mit den jeweiligen Fragmenten zu verknüpfen.

Als Activity steht die standartmäßige „MainActivity“ bereit. In deren „onCreate werden die Toolbar und der Navigationscontroller initialisiert. Anpassungen müssen hier nur beim erzeugen des „AppBarConfiguration.Builder“ vorgenommen werden. Hier müssen alle Fragmente des Navigationsgraphen angegeben werden, die als Toplevel bei der Navigation angesehen werden sollen.

2.2 Singleton (Entwurfsmuster)

Ein Singleton ist ein Entwurfsmuster, dass sicherstellt, dass zu jedem Zeitpunkt nur eine einzige Instanz einer Klasse existiert. Üblicher Weise wird dies erreicht indem der Konstruktor als private deklariert wird und der Abruf eines Objektes dieser Klasse nur durch eine statische Methode möglich ist.

Sinnvoll ist dies immer dann, wenn es aus logischer Sicht nie mehrere dieser Objekte geben sollte oder die Existenz mehrerer Objekte zu Fehlern führen kann.

2.3 Fassade (Entwurfsmuster)

Die Aufgabe des Entwurfsmusters Fassade ist es, Schnittstellen zu vereinheitlichen und zu vereinfachen. Sie dient meist dazu die Nutzung tieferer Schichten in einer Schichtenarchitektur zu vereinfachen. Dazu schachtelt sie diese Schichten und verbirgt die Kommunikation zwischen diesen Subsystemen nach Oben. So müssen höhere Schichten nur eine Schnittstelle implementieren und müssen sich nicht um die Koordination der tieferen Schichten kümmern.

3 Konzept

Im Folgenden wird nun das Grundlegende Konzept erläutert, nach welchem die App entwickelt wurde. Dazu wird zuerst das Datenbankschema erklär, um ein Verständnis dafür zu schaffen, welche Daten von dem System verarbeitet werden und wie diese Dargestellt werden. Danach wird die Grundlegende Strukturierung des Programmes erläutert und zum Schluss noch einmal der geplante Aufbau des Nutzerinterfaces aufgezeigt.

3.1 Datenbankschema

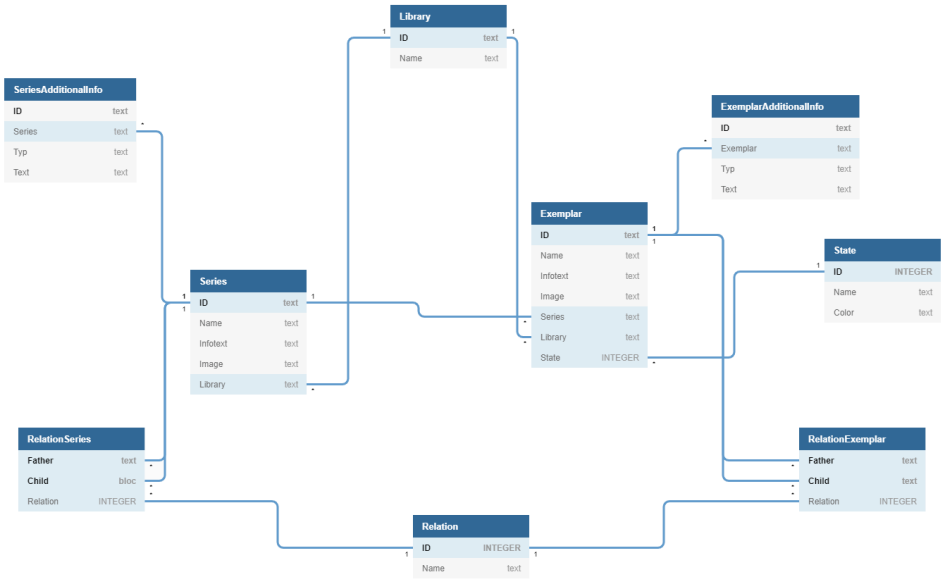


Abbildung 1: Datenbankschema

Da es möglich sein soll die verschiedensten Arten von Sammlungen über die App zu managen, musste auch das Datenschema möglichst allgemein gehalten werden, was sich auch in der Datenbank wiederspiegelt. Um die verschiedenen Sammlungen zu unterscheiden, werden diese verschiedenen Bibliotheken (engl. Library) zugeordnet.

Die einzelnen Objekte einer Sammlung werden als Exemplare abgespeichert und verfügen neben einem Namen noch über einen Info-Text, sowie ein Bild. Alle weiteren Informationen werden alle Zusätzliche Informationen in einer anderen Tabelle gespeichert, mit einem Verweis auf das zugehörige Exemplar. Zusätzlich wird jedem Exemplar noch ein Status zugewiesen, ob dieses Bereits im Besitz ist. Als Werte stehen dabei in der ersten Programmversion „In Besitz“, „Erhältlich“, „Angekündigt“, „Vorbestellt“ und „Nicht Verfügbar“ zur Verfügung. Exemplare können darüber hinaus noch in einer Beziehung zueinanderstehen (aktuell „Nachfolger“ und „Spinoff“).

Außerdem können mehrere Exemplare auch zu einer Serie zusammengefasst werden. Diese verfügt wiederum auch wieder über einen Namen, einen Info-Text und ein Bild. Weitere Informationen können auch hier in einer weiteren Tabelle nach Belieben hinzugefügt werden. Und selbstverständlich können auch verschiedene Serien in Beziehung zu einander stehen.

3.2 Programmstruktur

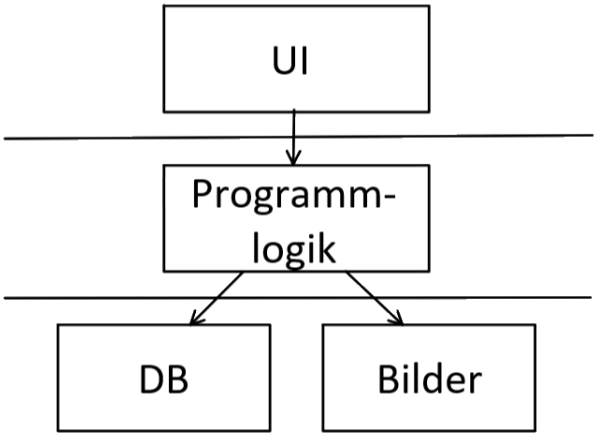


Abbildung 2: Schichtenarchitektur

Als Programmstruktur wurde sich für eine typische Drei-Schichten-Architektur, welche das Programm in Nutzerschnittstelle, Programmlogik und Basissysteme und Schnittstellen aufteilt. Die Basissysteme sind zum einen die Datenbank zur Speicherung von Daten, inklusive der Schnittstelle in Java und die Verwaltung des Bilderspeichers. Es wurde sich dagegen entschieden die Bilder direkt in der Datenbank zu speichern, um mehr Freiraum bei der Wahl des Speicherortes zu erhalten. In der aktuellen Programmversion werden die Bilder zwar immer noch im Appverzeichnis gespeichert, jedoch ist es so einfach möglich, später die Funktion zur Auslagerung der Bilder auf die SD-Karte nachzureichen. Dies kann vor allem bei älteren Smartphones nützlich sein, die noch über sehr wenig internen Speicher verfügen. Da diese Schichten sehr von Betriebssystem abhängig sind, werden die konkreten Implementierungen durch den Einsatz von Interfacen von den oberen Schichten abgekoppelt.

In der zweiten Schicht steht die Programmlogik, welche möglich unabhängig von Faktoren wie dem Betriebssystem sein sollte, um die Wiederverwendbarkeit des Programmcodes für alternative Systeme hoch zu halten.

Und als oberste Schicht steht die Benutzerschnittstelle. Diese ist natürlich wieder sehr Betriebssystem spezifisch und verfügt daher über die geringste Wiederverwendbarkeit.

3.3 Skizzen

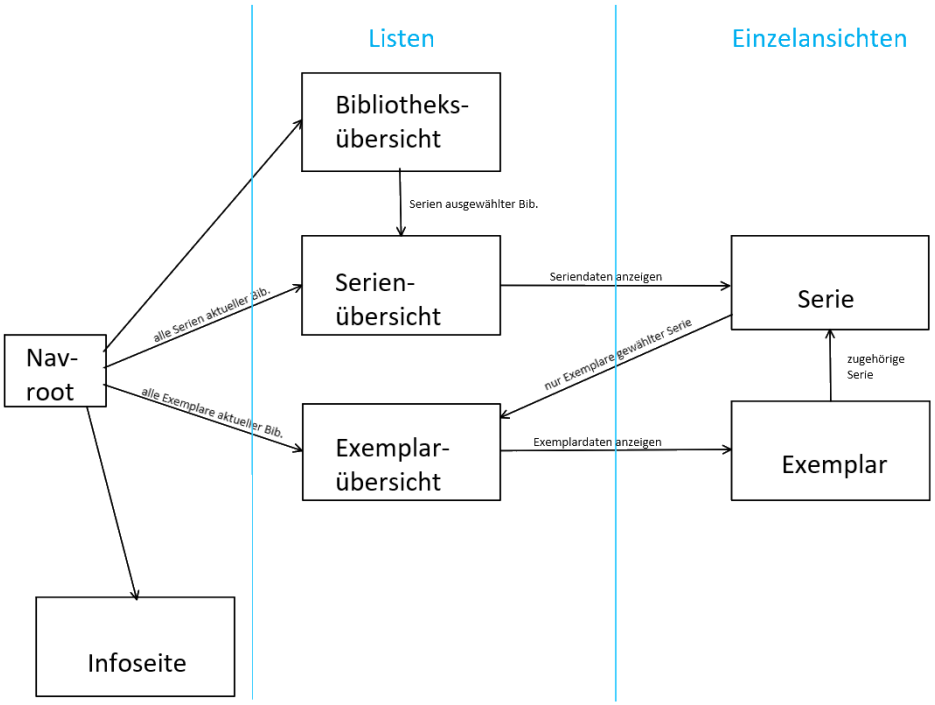
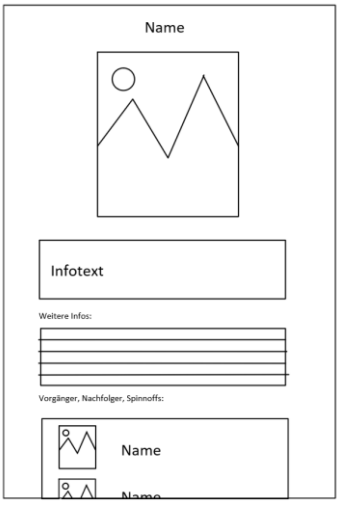


Abbildung 3: Navigationsgraph

Die Navigation durch die App soll möglichst einfach und intuitiv gehalten werden. Durch ein Navigationselement sollen die Übersichten über Bibliotheken, Serien und Exemplare sowie eine Infoseite über die App (inkl. Funktion zum Laden von Demodatensätzen) erreichbar sein. Serien und Exemplare sollen jeweils nur von der aktuell ausgewählten Bibliothek angezeigt werden, welche über die Bibliotheksübersicht gewechselt werden kann. Nach einem Wechsel soll zur Übersicht der enthaltenen Serien gewechselt werden.

Durch die Auswahl einer Serie oder eines Exemplars soll dann ein Fenster mit den genauen Daten angezeigt werden, da in der Übersicht ja, aufgrund des begrenzten Platzes, nur die wichtigsten Informationen angezeigt werden können. Bei Exemplaren soll zudem die Möglichkeit bestehen direkt zur zugewiesenen Serie zu Springen und bei Serien sollte eine Übersicht der ihr zugewiesenen Exemplare aufgerufen werden können.

Die Listenansichten sollen einfach gehalten werden. Sie sollen nur das Bild, den Namen und bei Exemplaren evtl. noch den Status anzeigen, um die Übersichtlichkeit zu waren. Durch einen einfachen Klick sollen dann die jeweiligen Daten angezeigt werden bzw. bei einer Bibliothek der Name bearbeitet werden können. Durch längeres drücken soll sich eine Anfrage zum Löschen des Datensatzes öffnen.



Die Einzelansichten sollen recht Übersichtlich gehalten werden. Sie beginnen mit den wichtigsten Informationen wie Name, Bild und Info-Text. Danach sollen die weiteren Informationen gefolgt von kleinen Auflistungen der Serien und Exemplare, mit denen Beziehungen bestehen.

Bei Exemplaren sollte vor dem Info-Text noch eine Anzeige des Status und vor den Beziehungen noch die Anzeige der zugehörigen Serie liegen.

Das Bearbeiten der Datensätze soll ebenfalls möglichst einfach gehalten werden. Durch längeres Klicken auf eine UI-Element soll sich ein Dialog zum Bearbeiten des Datenwertes öffnen. Beim Namen zum Beispiel ein Eingabedialog und beim Bild die Galerie der Smartphones, um ein neues Bild auszuwählen.

Abbildung 4: Skizze Einzelansicht

4 Realisierung

In diesem Abschnitt wird die eigentliche Funktionsweise des Programmes genau erklärt. Dabei werden die Schichten aus Abschnitt 3.2 nacheinander, von unten nach oben behandelt. Dabei wurde jedoch das Datenschema, die Klassen, die die eigentlichen Daten repräsentieren, herausgezogen, da diese in allen Schichten präsent sein müssen.

4.1 Datenschema

Das Datenschema besteht zum einem aus Klassen für die 3 Hauptelemente Bibliotheken (Klasse Library), Serien (Klasse Series) und Exemplaren (Klasse Exemplar). Diese enthalten jeweils Variablen für die Daten, die bereits im Abschnitt 3.1 Datenbankschema erläutert wurden. Zusätzlichen verfügen Serien und Exemplare noch jeweils über Listen, die ihre zusätzlichen Informationen (Klassen AdditionalInfo) und ihre Beziehungen zu anderen Serien bzw. Exemplaren (Klasse Relation) enthalten. Außerdem wurde der Status auch ein eine Klasse gekapselt (Klasse State), da dieser so leichter zu verwalten ist.

Bei der Erstellung dieser Klassen wurde darauf geachtet keine Android spezifischen Datentypen zu verwenden, um die Wiederverwendbarkeit auf anderen Systemen zu gewährleisten. Daher wird zum Beispiel auch die Farbe des Status als String und nicht als Color gespeichert. Für die IDs wurde sich für den Einsatz von UUIDs entschieden. Dadurch wird automatisch sichergestellt das sich keine IDs doppeln und selbst wenn zukünftig eine Synchronisierung von Daten über das Internet geplant wird, stellen die IDs kein Problem dar, da auch bei verschieden Geräten es nahezu ausgeschlossen ist, dass sich die IDs doppeln.

4.2 Datenbank

Zur Handhabung der Datenbank steht den oberen Schichten das Interface „IDatabase“ zur verfügen. Dies ist für die Entkopplung wichtig, da die Implementierung der eigentlichen Datenbankschnittstelle sehr Betriebssystem spezifisch ist.

Im Interface sind Methoden zum Abrufen der Verschiedene Datentypen aus der Datenbank vorgegeben. Diese können entweder einzeln oder als Liste abgerufen werden. Ausgenommen sind davon nur die Beziehungen, die Zusätzlichen Informationen und die Liste aller möglichen Status, diese können nur als Listen abgerufen werden, da der Abruf eines einzelnen Datensatzes für sich keinen Sinn hat. Da der Abruf der Beziehungen und zusätzlichen Informationen für Exemplare und Serien fast identisch ist, wurde hierfür nur eine Methode vorgesehen, diese erhält dann als Enum einen Parameter, der angibt aus welcher Tabelle die Datensätze abgerufen werden sollen.

Die Implementierung erfolgt in der Klasse „DBAndroid“. Als Datenbank wurde sich für eine SQLite-Datenbank entschieden, da diese leicht zu verwalten ist und keine zusätzlichen Dienste benötigt. Dementsprechend erbt der Schnittstelle auch von der Klasse „SQLiteOpenHelper“, diese erleichtert unter Android die Einrichtung einer SQLite-Datenbank, da diese nur mit einem Datenbanknamen und einer Versionsnummer beliefert werden muss und danach die gesamte Verbindungsverwaltung selbst übernimmt. Zusätzlich folgt die Datenbank noch dem Singleton-Entwurfsmuster (siehe 2.2), um Synchronisationsprobleme zwischen Instanzen zu unterbinden. Bevor jedoch die Instanz das erste Mal abgerufen kann, muss sie einmalig mit der Methode „initialize“ initialisiert werden, da für die Erzeugung der Instanz der „Context“ der App notwendig ist. Des Weiteren existiert noch eine zusätzliche Methode zum Löschen der Datenbank, nachdem diese ausgeführt wurde, muss die Klasse erneut initialisiert werden, da die alte Instanz danach verworfen wird.

Durch die Superklasse sind die Methoden „onCreate“ und „onUpdate“ vorgegeben, in welchen jeweils entweder die Tabellen völlig neu geschrieben werden oder ein Update von einer älteren Version erfolgen muss. Die Updatemethode ist in der aktuellen Version leer, da es keine ältere Datenbank Version geben kann. Der Notwendige SQL-Code wurde in eine separate SQL-Datei ausgelagert, um den Programmcode Übersichtlich zu halten.

Des Weiteren sind auch die Methoden des Interfaces implementiert, in welchen jeweils eine oder mehrere Anfragen an die Datenbank durchgeführt werden und die Ergebnisse in das entsprechende Datenformat aus dem Datenschema überführt werden. Zusätzlich befinden sich am Ende der Klasse noch einige private statische Klassen, welche die Strings für die verschiedenen Tabellennamen und deren Spalten enthalten, um so Schreibfehlern vorzubeugen.

Im Testverzeichnis von Android Studio befindet sich für diese Klasse noch zusätzlich die Testklasse „DBAndroidTest“. In ihr werden die wichtigsten Funktionen der Datenbank getestet. Sie diente während der Programmierung als Möglichkeit die Datenbank auch unabhängig von der UI zu testen. Aus diesem Grund wurde bei den Tests auch mehr auf Funktionsfähigkeit und Richtigkeit getestet, als auf Fehlertoleranz.

4.3 Bilderverwaltung

Das zweite Subsystem auf der untersten Schicht dient zur Verwaltung der Bilder. Auch hier wurde wieder ein Interface zur Entkopplung implementiert („IImageManager“). Dieses ist jedoch recht einfach gehalten, es enthält nur je eine Methode zum Abrufen, Speichern und Löschen eines Bildes. Dabei wird beim Speichern ein boolescher zurückgegeben, ob das Speichern erfolgreich war. Zusätzlich existiert noch eine Methode zum Herunterladen der Bilder der Beispieldatensätze, da dies asynchron Erfolg, muss beim Aufruf dieser Methode eine Callback-Funktion übergeben werde, welche als Wert erhält, ob der Download erfolgreich war oder nicht.

Die Implementierung erfolgt in der Klasse „ImageManagerAndroid“ und ist auch recht einfach, da es sich um simple IO-Aktionen handelt. Beim Erzeugen eines Objektes dieser Klasse wird im Konstruktor der Dateipfad zum Speichern übergeben. Beim Speichern werden alte Dateien einfach Überschrieben, da sich bei der Konzeption dafür entschieden wurde, die UUIDs der zugehörigen Objekte als Dateinamen zu nutzen und so beim Ändern von Bildern die alten automatisch überschrieben werden.

4.4 Programmlogik

In der mittleren Schicht liegt die Programmlogik (Klasse ProgramLogic). Sie verfolgt das Entwurfsmuster der Fassade (siehe 2.3). Es wurde hier auf ein Interface verzichtet, da die Programmlogik selbst bereits so Konzipiert ist, dass sie kein Betriebssystem spezifischen Code enthält. Sie folgt des Weiteren auch dem Entwurfsmuster des Singletons, da es zu jedem Zeitpunkt nur eine Steuerlogik geben sollte. Außerdem ist sie dadurch Problemfrei aus jeder Klasse der Benutzeroberfläche abrufbar, ohne Verweise zwischen den Klassen austauschen zu müssen.

Auch sie muss vor dem ersten Abruf der Instanz initialisiert werden, dabei erhält sie je ein Objekt der beiden Interface‘ der unteren Schicht, mit denen Sie dann Arbeitet. In der aktuellen Programmversion leitet sie die meisten Funktionen der Datenbank zum Abrufen und Speichern von Daten einfach weiter. Die Methoden, die in der Datenbank noch für Serien und Exemplare gleich sind, spaltet sie in zwei separate Methoden auf und parametrisiert diese entsprechend, um so die Handhabung in der Benutzeroberfläche zu vereinfachen.

Beim Laden und Speichern von Bildern, wird die Verwaltung der Dateinamen und das Speichern und Abrufen dieser in den Serien und Exemplaren von der Programmlogik übernommen, so das sich in den höheren Schichten nicht mehr damit befasst werden muss.

Bei den Löschfunktionen wird auch sichergestellt, dass nicht nur die Datensätze aus der Datenbank gelöscht werden, sondern auch die Bilder im Speicher. Zusätzlich wird noch die Verwaltung der aktuell aktiven Bibliothek durch diese Ebene übernommen, um auch hier wieder die Handhabung in der Benutzeroberfläche zu vereinfachen.

4.5 Benutzerinterface

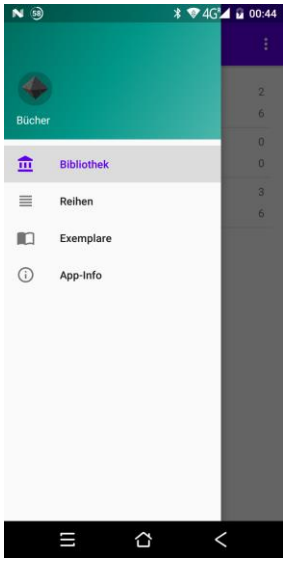


Abbildung 5: Navigationsleiste

Wie bereits im Abschnitt Skizzen erwähnt ist die graphische Oberfläche primär in zwei Teile geteilt: die Listenansichten und die Einzelansichten. Entsprechend werden sie auch hier vorgestellt. Nachdem zu Beginn kurz die allgemeine Funktionsweise der Oberfläche erläutert wird, werden dann zuerst die Listen- gefolgt von den Einzelansichten genauer erklärt.

Als Grundlage für das Layout des „Navigation Drawers“ (siehe 2.1). Dabei wurde der Navigationsgraph, wie in Abschnitt 3.3 aufgezeigt, konfiguriert. Als Rootelemente des Graphen dienen dabei die Listenansichten, sowie die Infoseite der App. Diese werden auch in der „MainActivity“ als Rootelemente konfiguriert.

In der „MainActivity“ werden des Weiteren die Objekte der Klassen der untersten Schicht zu Programmstart erzeugt und an die Programmlogik übergeben. Außerdem sind hier die Methoden zum Steuern innerhalb der Navigationsgraphen ohne das Navigationsmenü abgelegt. So können hier die Einzelansichten oder die gefilterte Listenansicht für Exemplare einer spezifischen Serie geöffnet werden.

Dem verwendeten Navigationslayout entsprechend sind die Inhaltsseiten als Fragments angelegt, welche dann über den Navigationskontroller geladen werden.

4.5.1 Übersichten

Bei den Listenansichten handelt es sich um einfach Übersichtlisten, in denen alle Elemente die zur aktuellen Bibliothek gehören in Kurzform aufgelistet werden.

4.5.1.1 ListFragment

Da die Listenform für alle Übersichten gleich ist, wurde diese in der abstrakten Klasse „ListFragment“ zusammengefasst. Hier wird auch das Layout aus der „fragment_simplelistview.xml“ geladen, welches entsprechend nur eine einfache Listview enthält. Das Verknüpften der Listview mit dem entsprechenden Adapter muss von jeder abgeleiteten Klasse selbst in der Methode „reloadList“ übernommen werden. Weiterhin müssen abgeleitete Klassen je eine Methode implementieren, die bei einem kurzen und einem langen Klick auf ein Listenelement ausgelöst wird, dabei bekommen sie das entsprechend verknüpfte Objekt aus dem Adapter als Parameter übergeben.

4.5.1.2 Libraries

Die erste Übersicht ist die der Bibliotheken (Klasse Libraries). Das Layout der Zeilen wird hier durch ein Objekt der Klasse „LibraryListAdapter“ geliert, welcher es aus der „row_library.xml“ lädt und mit Werten befüllt. Dabei werden neben dem Namen der Bibliothek auch die Anzahl der enthaltenen Serien und Exemplare angezeigt.

Bei einem kurzen Klick auf eine Bibliothek wird diese als aktive in der Programmlogik ausgewählt und zur Übersicht der in ihr enthaltenen Serien gewechselt. Bei einem langen Klick wird eine Anfrage gezeigt, ob diese Bibliothek gelöscht werden soll.

Es wird zudem noch ein Optionsmenü in der Toolbar angezeigt um eine neue Bibliothek zu erstellen. Wird diese Ausgewählt wird einen Dialog zum eingeben des Namens der neuen Bibliothek geöffnet und nach erfolgreicher Eingabe die Bibliothek erstellt. Das Layout für das Optionsmenü ist in der „overwiew.xml“ definiert.

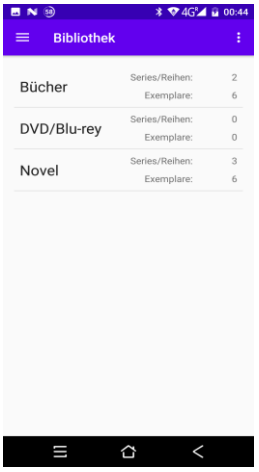


Abbildung 6: Bibliotheksübersicht

4.5.1.3 SeriesOverview

Abbildung 7: Serienübersicht

Die Serienübersicht (Klasse SeriesOverview) ist ähnlich simpel aufgebaut. Als Adapter für die Listview wird hier der „SeriesListAdapter“ genutzt, welcher das Layout aus der „row_series.xml“ lädt und das Bild der Serie in Miniaturansicht zeigt, sowie deren Namen und den Beginn des Infotextes.

Bei einem langen Klick auf eine Serie wird hier ebenfalls eine Anfrage zum Löschen gestellt und bei einem kurzen Klick wird die Einzelansicht der entsprechenden Serie geöffnet. Beim Löschen wird zusätzlich noch abgefragt, ob die in der Serie enthaltenen Exemplare ebenfalls mitgelöscht werden sollen.

Als Optionsmenü dient auch hier die „overwiew.xml“, welche bei Auswahl des Menüpunktes „Neu“ den Namen der neuen Serie abfragt. Nach erfolgreicher Eingabe wird hier allerdings die Einzelansicht der neuen Serie geöffnet, in der dann alle anderen Daten eingegeben werden können.

4.5.1.4 ExemplarOverview

Die Exemplarübersicht (Klasse ExemplarOverview) funktioniert grundlegend genauso wie die Serienübersicht, nur dienen hier als Adapter der „ExemplarListAdapter“ und als Layout die „row_exemplar.xml“. Besonders ist hier jedoch, dass zusätzlich noch Parameter bei der Navigation mit übergeben werden können. Diese dienen für den Fall das die gezeigten Exemplare nach einer spezifischen Serie gefiltert werden sollen.

Der Parameter „dynamicTitle“ enthält hierbei den Namen der Serie und wird durch den Navigationskontroller ausgewertet um diesen in der Toolbar anzuzeigen. Dazu wurde im Navigationsgraphen einsprechend der Titel mit einem Verweis auf diesen Parameter und mit einem Default-Wert für den Fall einer ungefilterten Auflistung angelegt.

Der Parameter „id“ enthält wiederum die ID der Serie, welche zur Filterung notwendig ist und bei der bestücken des Adapters mit Datensätzen zum Einsatz kommt.

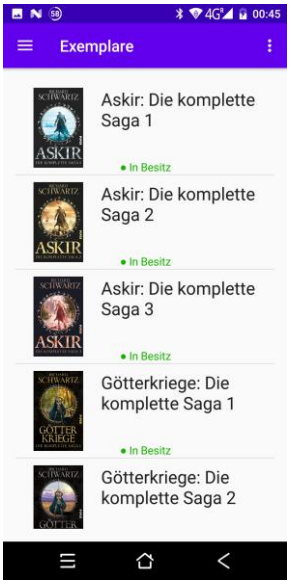


Abbildung 8: Exemplarübersicht

4.5.2 Einzelansichten

Die zweite große Gruppe an Fragmenten in der App sind die Einzelansichten. Sie dienen zur Anzeige und Bearbeitung der verschiedenen Datentypen. Um die Benutzeroberfläche möglichst einfach und intuitiv zu halten, erfolgt die Bearbeitung durch längeres „drücken“ auf einen Datenwert. Dadurch wird ein Dialog geöffnet, indem der entsprechende Wert bearbeitet werden kann. Da Bibliotheken jedoch nur aus ihrem Namen und ihrer ID bestehen, existieren nur Einzelansichten für Serien und Exemplare.

4.5.2.1 InfoAndRelation

Serien und Exemplare verfügen beide über eine Liste von zusätzlichen Informationen, sowie Listen für die verschiedenen Beziehungen. Da diese in beiden Fällen identisch aufgebaut sind, wurde für die Anzeige und Verwaltung dieser ein eigenständiges Fragment (Klasse InfoAndRelation) geschrieben, welches von beiden Einzelansichten genutzt wird.

An dieser Stelle wurde sich für ein vollständiges Fragment anstelle einer abstrakten Oberklasse entschieden, da hier auch die gesamte Auswertung und Verwaltung der Daten identisch ist. Auf den Typ (Serie oder Exemplar) muss nur beim Abrufen der Daten auf der Programmlogik geachtet werden. Daher muss der Typ auch nur einmalig im Konstruktor gemeinsam mit der ID der Serie bzw. des Exemplars übergeben werden. Es kann an dieser Stelle auch ausnahmsweise der Konstruktor anstelle eines Bundels für die Übergabe der Parameter genutzt werden, da dieses Fragment nur von den umschließenden Fragmenten selbst und nicht von einem Navigator erzeugt wird.

Das Layout wird aus der „fragment_info_and_relation.xml“ geladen. Diese enthält für die zusätzlichen Informationen und die Beziehungen jeweils zugehörige Listen. Hier kommen allerdings keine Listview zum Einsatz, da an dieser Stelle gewünscht ist, dass alle Items der Listen gleichzeitig angezeigt werden und eine Listview diese evtl. hinter einer Scroll-Funktion versteckt. Unter Android ist für diesen Anwendungsfall das Linearlayout vorgesehen. Dieses verfügt jedoch über keine Unterstützung für Adapter zur Verwaltung der Daten, daher wurde die eigene Klasse „StackView“ geschrieben, welche das standartmäßige Linearlayout um eine Adapterverbindung und „onItemClickListener“ erweitert.

Als Adapter dienen der „InfoListAdapter“, der das Layout aus der „row_info.xml“ nutzt und der „RelationListAdapter“, welcher das Layout der „row_relation.xml“ nutzt. Die Anzeige ist dabei recht simple gehalten. Die Informationen werden in Tabellenform mit Typ und Text angezeigt und die Beziehungen jeweils mit Bild und Name der Serie oder des Exemplars, mit dem die Beziehung besteht. Der Dialog zum Bearbeiten von Informationen besteht aus zwei Eingabefeldern, eines für den Typ und eines für den Text (Layout: „dialog_info.xml“).

Da die Bearbeitung einer Beziehung etwas komplexer ist, weil neben den Namen der beiden Beziehungspartner auch noch Typ und Richtung der Beziehung konfiguriert werden müssen, wurde sich hier gegen einen Dialog entschieden. Anstelle dessen wird eine neue Activity zur Bearbeitung geöffnet, die „RelationEditActivity“ (Eine Activity und kein Fragment, da die Navigationselemente bei der Bearbeitung nicht benötigt werden.). Sie zeigt für beide Beziehungspartner Eingabefelder, welche über eine Funktion zur automatischen Vervollständigung zur leichteren Eingabe verfügt, sowie eine Auswahlbox für den Beziehungstyp (Layout: „activity_relation_edit.xml“). Dabei ist das Eingabefeld für die aktuell gewählte Serie oder Exemplar immer schreibgeschützt, da ja nur der Beziehungspartner für die Eingabe relevant ist. Nach Bestätigung der Eingaben durch den Nutzer wird die bearbeitete Beziehung gespeichert.

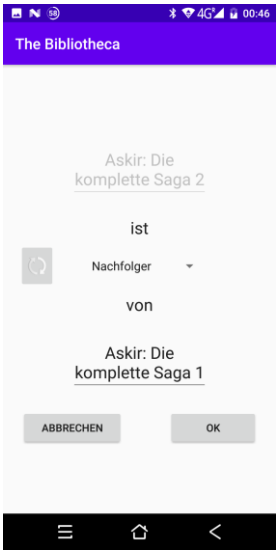


Abbildung 9: RelationEditActivity

Um neue Informationen und Beziehungen hinzuzufügen verfügt das „InfoAndRelation“-Fragment noch über Einträge im Optionsmenü der Toolbar (Layout: „info_and_relation.xml“). Zur Eingabe der neuen Werte werden dieselben Anzeigen wie für die Bearbeitung genutzt.

Um den Nutzern ein angenehmeres Navigieren durch die App zu ermöglichen besteht bei den Beziehungen noch die Möglichkeit, durch einen einfachen Klick auf diese zur Einzelansicht des Beziehungspartners zu wechseln.

4.5.2.2 SeriesView

Abbildung 10: Einzelansicht Serie

Ebenso wie die Übersicht für Exemplare verfügt auch die Einzelansicht für Series (Klasse SeriesView) über Parameter, die beim Aufruf übergeben werden müssen. Der Parameter „dynamicTitle“ enthält wieder den Namen der Serie, der in der Toolbar angezeigt wird. Der Parameter „series“ enthält aber dieses Mal die gesamte Serie und nicht nur die ID, dies ist darin begründet, da beim Wechsel aus der Serienübersicht die gesamte Serie bereits vorliegt und diese dann hier nicht nochmals aus der Datenbank abgerufen werden muss.

Das Layout ist in der „fragment_series_view.xml“ defniert und enthält neben Felder für den Seriennamen, das Bild und den Info-Text auch ein Framelayout, in welches dann nach Erstellung der View das „InfoAndRelation“-Fragment geladen wird. Da die gesamte Anzeige so evtl. länger sein kann als das Display befindet sich das gesamte Layout innerhalb einer ScrollView.

Auch hier sind Name, Bild und Info-Text wieder durch längeres „drücken“ von diesen Bearbeitbar. Beim Namen öffnet sich ein Dialogfeld zur Bearbeitung des bisherigen Namens und beim Bild wird die standartmäßige Bilderauswahl des Betriebssystems genutzt, um den Nutzer ein neues Bild auswählen zu lassen. Bei diesem handelt es sich um eine standartmäßige Activity, daher muss die Methode „onActivityResult“ implementiert werden, um das Ergebnis auszuwerten.

Da der Info-Text auch ziemlich lang sein kann, wäre es eher ungünstig diesen in einem Dialogfenster zu bearbeiten. Daher befindet dieser sich in einem EditText, welches erst nach längeren „drücken“ bearbeitbar wird und dann auch Buttons zur Bestätigung und zum Abbrechen der Bearbeitung anzeigt.

Im Optionsmenü steht die Option zum Anzeigen der in der Serie enthaltenen Exemplare zur Verfügung (Layout: „show_exemplars.xml“). Bei Auswahl dieses Menüpunktes wird die Exemplarübersicht mit der aktuellen Serie als Filter aufgerufen.

Beim Speichern von Werten besteht die Besonderheit, das vorher nochmals die alten Werte aus der Datenbank ausgelesen werden bevor diese überschrieben werden. Dies hat den Grund darin, das die zusätzlichen Infos und Beziehungen separat in dem „InfoAndRelation“-Fragment genhandhabt werden und sich daher in dem aktuell vorhandenen Serienobjekt veraltete Daten befinden könnten.

4.5.2.3 ExemplarView

Die Einzelansicht für Exemplare (Klasse ExemplarView) enthält dieselben Felder für Name, Bild und Info-Text wie auch die für Serien und diese funktionieren auch identisch. Zusätzlich enthält das hier verwendete Layout („fragment_exemplar_view.xml“) aber auch noch ein Feld für den Status. Bei längeren „drücken“ auf diesem wird ein Auswahlmenü geöffnet, indem der neue Status ausgewählt werden kann.

Außerdem steht hier auch ein Feld zur Verfügung, indem die zugehörige Serie angezeigt wird. Bei der Bearbeitung dessen wird ein Eingabedialog geöffnet, der über dieselbe Funktion zur Autovervollständigung wie auch in der Bearbeitung der Beziehungen verfügt. Da dieses Feld jedoch nur angezeigt wird, wenn auch eine Serie zugeordnet ist, ist der Eingabedialog auch über das Optionsmenü (Layout: „change_associated_series.xml“) aufrufbar.

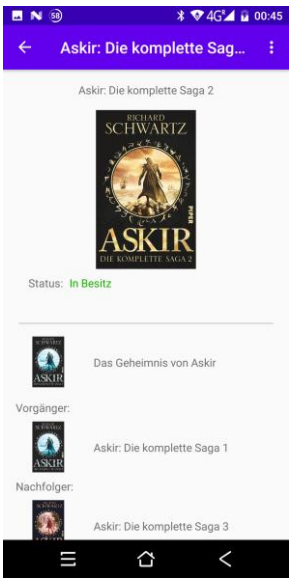


Abbildung 11: Einzelansicht Exemplar

4.5.3 Infoseite

Abbildung 12: App-Info

Das letzte Fragments ist das „InfoFragment“ (Layout: „fragment_info.xml“). Es enthält neben einer kurzen Beschreibung der App auch die Buttons mit den Testfunktionen zum Löschen der Datenbank und zum Einspielen der Demodatensätze.

5 Zusammenfassung und Erweiterungen

Insgesamt setzt die App das gewünschte Ziel erfolgreich um. Sie bietet eine Möglichkeit verschiedenste Sammlungen wie zum Beispiel DVDs und Bücher zu verwalten und immer den Überblick über den aktuellen Bestand der eigenen Sammlung zu behalten.

5.1 Mögliche Erweiterungen

Als optimal kann die App aber noch nicht angesehen werden. So wäre zum Beispiel noch eine Filterfunktion der Exemplare nach bestimmten Status praktisch. Und auch eine Synchronisation mit einem Server, um Daten zwischen verschiedenen Geräten abzugleichen, wäre in Erwägung zu ziehen.

Auch Programmiertechnisch könnte es noch ein paar kleine Verbesserungen geben. So ist zum Beispiel das Layout des „Navigation Drawer“ auf viele Dateien aufgesplittert, obwohl diese meist nur ein oder zwei Elemente enthalten. Diese ließen sich eventuell zusammenfassen, um die Übersichtlichkeit zu erhöhen. Außerdem übergibt die „StackView“ als „parent“ bei den „onItemClickListenern“ null, da diese eine Instanz der Klasse AdapterView erwarten, jedoch nicht sowohl von AdapterView als auch vom Linearlayout geerbt werden kann. Hier sollte eventuell noch nachgeforscht werden, ob es nicht irgendwie möglich ist die Vererbung über Interfaces oder ähnliches zu realisieren, um hier auch das echte „parent“ übergeben zu können.

Des Weiteren sind die Einzelansichten für Exemplare und Serien zu großen Teilen (was Name, Bild und Info-Text betrifft) identisch. Hier könnte überlegt werden ob man den Code hier eventuell zusammenfassen kann, um die Redundanz zu verringern.

Aber dies wären Verbesserungsvorschläge, die in einer zweiten Version der App angegangen werden können. Für den Moment ist jedoch diese Version auch vollständig zufriedenstellend und erfüllt ihren Zweck.

6 Indexverzeichnis