Bachelorarbeit Informatik

Entwicklung einer Android-App zur mobilen Nutzung einer zentralisierten Comicsammlung

vorgelegt von

André Hahn

zur Erlangung des akademischen Grads eines

Bachelor of Science

im Studiengang

Informatik

an der

Fakultät für Informatik und Ingenieurwissenschaften

der Technischen Hochschule Köln.

Erstgutachterin: Prof. Dr. rer. nat. Birgit Bertelsmeier (Technische Hochschule Köln)

Zweitgutachter: Prof. Dr. rer. nat. Christian Kohls (Technische Hochschule Köln)

28. Oktober 2019

André Hahn
Gutsberg 5
51674 Wiehl
Matrikelnummer: 11117190

Technology
Arts Sciences
TH Köln

Bachelorarbeit

Titel: Entwicklung einer Android-App zur mobilen Nutzung einer zentralisierten Comicsammlung

Gutachter:

Prof. Dr. rer. nat. Birgit Bertelsmeier (TH Köln), Prof. Dr. rer. nat. Christian Kohls (TH Köln)

Zusammenfassung: Neben den großen Anbietern digitaler Comics mit elektronischem Kopierschutz (DRM) haben sich auch Anbieter für DRM-freie Comics etabliert. Als Open Source-Alternative zu den Plattformen Ersterer wurde im Rahmen des Praxisprojekts die Software ComicLib als Web-App zur Verwaltung digitaler Comicsammlungen geschaffen. Diese ermöglicht die gemeinsame Verwaltung DRM-geschützter sowie -freier Comics. Letztere sind außerdem im Webbrowser lesbar. Die im Rahmen dieser Arbeit entstandene Android-App soll eine Open Source-Alternative zu den mobilen Apps der Plattformen der großen Anbieter sein und ComicLib um eine mobile App zur Offline-Nutzung der Comics erweitern. Der erste Prototyp wird im Nachgang des Projekts weiter getestet und verbessert, bevor die erste stabile Version veröffentlicht wird.

Stichwörter: Android, Comics, Kotlin, Open Source, REST

Datum: 28. Oktober 2019

Bachelors Thesis

Title: Development of an Android app for mobile use of a centralized comic book collection

Reviewers:

Prof. Dr. rer. nat. Birgit Bertelsmeier (TH Köln), Prof. Dr. rer. nat. Christian Kohls (TH Köln)

Abstract: Alongside the major vendors of digital comics with digital copy protection (DRM), providers of DRM-free comics have also established themselves. As an open source alternative to the platforms of the former, the software ComicLib was created as a web app for managing digital comic collections as part of the Praxisprojekt. It enables the combined administration of DRM-protected and DRM-free comics. The latter can also be read in the web browser. The Android app developed in the process of this project is intended to be an open source alternative to the mobile apps of the major providers' platforms and to extend ComicLib with a mobile app for offline use of the comics. The first prototype will undergo further testing and improvement in the aftermath of the project before the first stable version is released.

Keywords: Android, Comics, Kotlin, Open Source, REST

Date: 28 October 2019

Inhalt

1 Über das Projekt

Das folgende Kapitel gibt einen Überblick über die Rahmenbedingungen dieses Projekts. Die Aufgabenstellung und Motivation geben einen Einblick in die Ausgangssituation und die Zielsetzung des Projekts sowie die treibende Idee hinter der Durchführung. Zudem werden der Projektname und die Details zur Veröffentlichung der Software unter einer Open Source-Lizenz vorgestellt, da diese für die anschließenden Kapitel dieser Arbeit von wesentlicher Bedeutung sind. Um einen Überblick über die Inhalte dieser Arbeit zu vermitteln, beginnt dieses Kapitel jedoch mit einer Inhaltsübersicht.

1.1 Inhaltsüberblick

Das erste Kapitel widmet sich der Vorstellung des Projekts (*1*). Dort wird die Motivation für das Projekt – die Entwicklung einer freien Alternative zu den digitalen Plattformen der großen Comic-Anbieter – vorgestellt (*1.2*). Im Anschluss daran werden die Aufgabenstellung erörtert und mögliche Lösungsansätze beschrieben (*1.3*). Auf deren Grundlage werden die Ziele für das Projekt entwickelt, die für den dessen Erfolg maßgeblich sind (*1.4*). Im Anschluss daran wird der Projektname – Coboli – und der Weg zu dessen Findung vorgestellt (*1.5*). Zum Abschluss des ersten Kapitels wird die Rolle von Open Source Software im Projekt erklärt (*1.6*).

Im Rahmen der Projektdurchführung kommen verschiedene Hilfsmittel und Technologien zum Einsatz. Diese werden im zweiten Kapitel vorgestellt (*2*). Zu den Hilfsmitteln (*2.1*) zählen, unter anderem, das Versionsverwaltungssystem, die Programmierumgebung und die Software zur Aufgabenplanung. Zu den durch Coboli genutzten Technologien (*2.2*) gehören, neben anderen, die verwendete Programmiersprache Kotlin und die Comic-Verwaltungssoftware ComicLib, auf deren Inhalten die App aufbaut.

Im dritten Kapitel geht es um die Entwicklung der Software im Rahmen der Durchführung des Projekts (*3*). Am Anfang der Entwicklung von Coboli, wie auch am Anfang dieses Kapitels, steht die Einrichtung der Testumgebung (*3.1*). Damit die App getestet werden kann, müssen ein ComicLib-Server und ein Android-Gerät als Testgerät eingerichtet werden.

Die ComicLib API stellt die Basis für die Entwicklung der Software dar, da Coboli alle Inhalte in der App von dieser bezieht. Damit mit der API gearbeitet werden kann, müssen erst Informationen zu Authentifizierung, Ressourcen und Verbindung zur API beschafft werden. Die Eigenschaften der ComicLib-API werden daher detailliert besprochen (*3.2*).

Coboli verwendet einige Software-Bibliotheken zur Bereitstellung seiner Funktionen. Diese dienen unter anderem zum Verbinden mit der API, zur Umwandlung der API-Antworten in

Datenobjekte und zur Speicherung dieser in einer Datenbank (*3.3*). Die auf dieses Teilkapitel folgenden Kapitel nehmen häufig Bezug auf die hier vorgestellten Bibliotheken.

Damit Coboli mit der ComicLib-API kommunizieren kann, muss erst eine Verbindung zum ComicLib-Server aufgebaut werden. Da es beim Verbindungsaufbau neben der Verfügbarkeit des Servers auch auf die Anmeldedaten und eine optionale Verschlüsselung ankommt, ist eine komplexe Fehlerbehandlung für diese notwendig (*3.4*).

Coboli nutzt neben vier Datenbanktabellen auch einige Datenbankviews, um die Datensätze zur Nutzung in der App aufzubereiten. Da die Zusammenhänge zwischen den Tabellen und Views komplex sind, müssen diese genauer betrachtet werden (*3.5*). Zur leichteren Verständlichkeit kommt ein Entity-Relationship-Diagramm zum Einsatz, das die Tabellen und Views visualisiert.

Damit Coboli die Inhalte der ComicLib-API darstellen kann, muss die lokale Datenbank mit der ComicLib-Datenbank synchronisiert werden (*3.6*). Dabei wird die ComicLib-Datenbank in Coboli gespiegelt und der Lesefortschritt zwischen beiden Datenbanken abgeglichen. Außerdem müssen der Bilder- und der Comic-Cache aktualisiert werden.

Die wichtigste Funktion von Coboli besteht in der lokalen Nutzung der Comicdateien (*3.7*). Nachdem ein Comic heruntergeladen wurde, kann er in der App zum Lesen angezeigt werden, falls sein Dateiformat unterstützt wird. Außerdem können Comicdateien mit anderen Apps geteilt werden. Für mehr Komfort beim Lesen der Comics gibt es zudem eine Leseliste, die alle angelesenen Comics zusammenfasst.

Damit Coboli auch unterwegs und ohne eine Internetverbindung genutzt werden kann, gibt es einen Offline-Modus (*3.8*). Dieser schränkt den Funktionsumfang der App ein, falls keine Verbindung zum Server besteht. Da Coboli mit sensiblen Anmeldedaten und personenbezogenen Daten arbeitet, muss auch auf die Sicherheit der Daten und den Datenschutz geachtet werden (*3.9*). Zur Sicherung der Daten kommt daher Verschlüsselung zum Einsatz. Damit Coboli nicht nur im deutschsprachigen Raum genutzt werden kann, muss die App an ein internationales Publikum angepasst werden (*3.10*). Coboli wurde daher konsequent zweisprachig entwickelt.

Am Ende der Durchführung stehen die umgesetzten Sichten und Funktionen der App (*3.11*). Der Benutzer kann die Sammlung betrachten und die enthaltenen Comics lesen. Außerdem können Informationen zur App und den genutzten Bibliotheken direkt in der App eingesehen werden.

Zum Abschluss des Projekts (*4*) wird geprüft, ob die gesteckten Ziele erreicht wurden (*4.1*). Außerdem müssen Fragen zum Vertriebsweg und zur Zukunft des Projekts geklärt werden (*4.2*).

1.2 Motivation

Bildergeschichten, die Urform des modernen Comics, stellen die älteste Form der geschriebenen Erzählung dar. Mit der fortschreitenden Entwicklung des Menschen und seiner Werkzeuge hat auch die Bildergeschichte stets einen Wandel erlebt. Die Höhlenmalereien der Steinzeit stellen das älteste Zeugnis der bildlichen Erzählung dar. Jedoch sind sie nur ein Vorläufer des Mediums Comic, da die Bilder nicht von Schrift begleitet werden. Auch in der Antike dienten Bildsequenzen ohne Text häufig als Erzählmedium. Ein besonders bekanntes Beispiel stellt die *Trajan-Säule* in Rom aus dem Jahr 113 n. Chr. dar, die in 115 Szenen von der Unterwerfung der Daker durch die Römer unter Kaiser Trajan berichtet.[1] Zu den ältesten bekannten Comics zählt der *Codex Nuttall*, eine Bilderhandschrift, deren Entstehung im Zeitraum zwischen dem 13. und 16. Jahrhundert vermutet wird. Erschaffen wurde er von den Mixteken, einer Ureinwohnergruppe Mexikos.[2] Der Codex erzählt eine Geschichte in Form von Bildern und Ideogrammen (Wortzeichen). Ein noch älteres Beispiel stellt der in Frankreich entstandene Teppich von Bayeux aus der zweiten Hälfte des 11. Jahrhunderts dar. Er erzählt in Stickereien von der normannischen Eroberung Englands und enthält neben Bildern auch Inschriften in lateinischer Sprache.[3] Mit dem Aufkommen des Buchdrucks in Europa wurden Bildergeschichten zunehmend auch für die ärmeren Bevölkerungsschichten erschwinglich, so zum Beispiel „The Tortures Of Saint Erasmus“ (etwa 1460).[4] Als Vater des Comicstrips gilt der Schweizer Rodolphe Töpffer, der in der ersten Hälfte des 19. Jahrhunderts der Kombination aus Bild, Text und Panelgrenzen zum Durchbruch verhalf.[5] Im Laufe der zweiten Hälfte des 19. Jahrhunderts fand der Comicstrip durch Zeitungen und Satiremagazine zunehmende Verbreitung. Gegen Anfang des 20. Jahrhunderts entwickelte sich aus den Satiremagazinen dann das Comicheft.[6]

In den letzten Jahren hat sich neben dem Markt für elektronische Bücher ebenfalls ein, in Teilen deckungsgleicher, Markt für digitale Comics etabliert. Die Anbieter für E-Books (z.B. Amazon Kindle[7], Apple Books[8]) handhaben Comics als Genre innerhalb ihrer Kataloge. Dazu gesellen sich weitere Anbieter, die sich auf digitale Comics spezialisiert haben (z.B. Comixology[9]). Die meisten Anbieter versehen ihre Ware mit einem digitalen Kopierschutz,

[1] Vgl. (National Geographic, 2015)

[2] Vgl. (The British Museum, 2008)

[3] Vgl. (Bayeux Museum, 2019)

[4] Vgl. (McCloud, Understanding Comics - The Invisible Art, 1993)

[5] Vgl. (University Press Of Mississippi, 2010)

[6] Vgl. (McCloud, Understanding Comics - The Invisible Art, 1993)

[7] https://www.amazon.de/kindle-dbs/fd/kcp

[8] https://www.apple.com/de/apple-books/

[9] https://www.comixology.com/

dem sogenannten Digital Rights Management (DRM). Auch die großen Verlage setzen beim digitalen Direktvertrieb meist auf DRM. Einige der kleinen und mittleren Verlage, zum Beispiel Valiant Entertainment[10] oder Image Comics[11], verzichten im Direktvertrieb jedoch auf DRM.[12] Zum Teil bieten sie ihre Comics beim Vertrieb über beispielsweise Comixology zusätzlich auch als DRM-freie Sicherungskopie an.[13] DRM-freie Comics werden häufig auch in größeren Paketen zu stark reduzierten Preisen angeboten, so zum Beispiel durch Humble Bundle.[14]

Comics mit DRM sind an die Plattform des jeweiligen Anbieters gebunden. Die Comicdateien können nur mit den Leseprogrammen des Anbieters gelesen werden. Bei manchen Anbietern, beispielsweise Comixology, ist sogar kein Export der Comicdateien zur externen Sicherung möglich. Bei Comics ohne DRM haben sich weitgehend offene Dateiformate durchgesetzt, die mit freier Software genutzt werden können, wodurch keine Bindung an eine Plattform besteht.

Der große Vorteil der Plattformen der großen Anbieter besteht für die Nutzer darin, dass sie ein sortierte und durchstöberbare Comic-Bibliothek erhalten, ohne sich selbst um die Organisation derselben kümmern zu müssen. Die Sammlung ist durchsuchbar, enthält Zusatzinformationen wie Titelbild oder Klappentext und hält automatisch fest, ob und wie weit ein Comicbuch gelesen wurde. Zusätzlich können die Comics häufig direkt im Webbrowser gelesen werden. Diese Vorteile sind bei den Anbietern DRM-freier Comics nicht gegeben. Dort muss sich der Kunde selbst um die Verwaltung seiner Comicsammlung auf seinem lokalen Gerät kümmern.

Eine digitale Comicsammlung besteht auf Grund von Rabattaktionen und Angebotspreisen häufig aus einer Mischung DRM-geschützter und -freier Comics unterschiedlicher Anbieter. Die Software ComicLib wurde im Praxisprojekt entwickelt, um eine gemeinsame Comicverwaltung für solche gemischten Sammlungen zu ermöglichen.[15] Die Motivation hinter der Entwicklung von ComicLib bestand darin, für DRM-freie Comics einen ähnlichen Funktionsumfang zu bieten wie die Plattformen der großen Anbieter, um so eine freie Alternative zu diesen zu schaffen. Dabei sollte es ermöglicht werden, die DRM-freien Comics direkt im Webbrowser zu lesen und alle Comics ungeachtet eines DRM herunterladen zu können. Zur Bereitstellung von Zusatzinformationen sollte ComicLib zudem auf die weltweit größte Online-Comicdatenbank ComicVine zugreifen.

[10] http://valiantentertainment.com/

[11] https://imagecomics.com/

[12] Vgl. (The Hollywood Reporter, 2014)

[13] Vgl. (Comixology Inc., 2019)

[14] https://www.humblebundle.com/

[15] Vgl. (Hahn, Entwicklung einer freien Software zur Verwaltung von digitalen Comics auf Basis von Web-Technologien, 2019)

Mit dem erfolgreichen Abschluss des Praxisprojekts steht ComicLib als freie Alternative zu den Plattformen der großen Comic-Anbieter zur Verfügung.[16] Durch die mobilen Apps haben diese allerdings immer noch einen Vorteil gegenüber ComicLib: Die Apps ermöglichen es, die Comicsammlung auch ohne eine aktive Internetverbindung immer dabei zu haben. Daher stellt ComicLib eine Programmierschnittstelle (API) für andere Anwendungen zur Verfügung, die eine Synchronisierung mit der Datenbank ermöglicht.

Das Ziel dieses Projekts ist es nun, unter Nutzung der ComicLib-API eine Android-App zu entwickeln, die diese Lücke zwischen dem Funktionsumfang der Comic-Plattformen der großen Anbieter und ComicLib schließt. Im Kapitel *Ziele des Projekts* werden die (Teil-)Ziele des Projekts im Detail dargelegt.

1.3 Aufgabenstellung

Mit dem Aufkommen digitaler Comics haben sich viele, zum Teil proprietäre und kopiergeschützte Dateiformate für diese etabliert. Um Comics aller Dateiformate in einer gemeinsamen Sammlung verwalten zu können, wurde im Rahmen des Praxisprojekts die Web-Anwendung ComicLib entwickelt, die sich in ihrem Funktionsumfang an den Webseiten der großen Comic-Anbieter orientiert. Diese Anbieter ermöglichen neben der Nutzung der Comicsammlung im Webbrowser auch das Herunterladen zur mobilen, Internet-unabhängigen Nutzung der Sammlung mittels einer mobilen App. Im Rahmen dieses Projekts soll eine Android-App entstehen, die sich an jenen Apps orientiert und auf der REST-API von ComicLib aufbaut. Aus der Vielzahl der Comicdateiformate und der Synchronisation mit der ComicLib-API ergeben sich besondere Anforderungen an eine solche App.

1.3.1 Anforderungen an die Software

Da die Inhalte der App über die ComicLib-API bereitgestellt werden, muss zunächst eine Verbindung zur API aufgebaut werden. Das setzt voraus, dass der Benutzer die Webadresse der ComicLib-Instanz in der App eingeben und speichern kann. Da die API eine Authentifizierung vorsieht, müssen zudem die Logindaten, bestehend aus Benutzername und Passwort, hinterlegt werden. Nach Authentifizierung mittels der Logindaten muss von der API ein persönlicher API-Schlüssel bezogen werden, der zur Authentifizierung bei allen weiteren API-Anfragen an Stelle von Benutzername und Passwort verwendet wird.[17]

[16] Vgl. (Hahn, ComicLib, 2019)

[17] Vgl. (Hahn, ComicLib API Version 1, 2019)

Da die Logindaten wie auch der persönliche API-Schlüssel bezüglich des Sicherheitsaspekts sehr sensibel sind darf eine Speicherung nur verschlüsselt erfolgen. Zudem sollte sichergestellt sein, dass die Übertragung an die API ebenfalls nur verschlüsselt erfolgt.

Zur lokalen Nutzung der Comicsammlung aus ComicLib muss eine Synchronisierung über die API stattfinden. Diese muss entweder automatisch stattfinden oder durch den Benutzer auslösbar sein.

Damit die Comics auf dem lokalen Gerät genutzt werden können ist es erforderlich, diese herunterladen zu können und heruntergeladene Comics in einer Übersicht darzustellen. Da viele Comicformate proprietär oder wenig verbreitet sind, ist es notwendig, bezüglich der Dateiformate, die in der App den vollen Funktionsumfang inklusive einer Leseansicht erhalten sollen, eine Auswahl zu treffen. Jedes freie Comicformat zu unterstützten ist im begrenzten Zeitrahmen des Projekts nicht möglich und für die proprietären Formate besteht häufig keine Möglichkeit, diese zu nutzen. Auf Grund dieser Unterscheidung zwischen voll und partiell unterstützten Dateiformaten müssen für einige Funktionen bezüglich der voll unterstützten Formate Alternativen für die partiell unterstützten geschaffen werden.

Für alle Comicdateien, unabhängig vom Dateiformat, soll es möglich sein, den Lesezustand (ungelesen, angelesen, zu Ende gelesen) in der Sammlung zu ändern. Für Serien soll es möglich sein, die gesamte Serie als gelesen zu markieren. Die angelesenen Comics sollen in einer Leseliste angezeigt werden. Für Verlage, Serien und Comichefte sollen Zusatzinformationen in Form des Beschreibungstextes aus der ComicVine Datenbank angezeigt werden können.

Da die App auf anderen, quelloffenen Software-Bibliotheken aufbaut, muss eine Liste der verwendeten Bibliotheken sowie deren Lizenzen in die App eingebaut werden, um den Anforderungen der Open Source Lizenzen gerecht zu werden. Zusätzlich kann ein Link zur Webseite des jeweiligen Software-Projekts eingebaut werden. Zudem sollte in der App ein Hinweis auf das Copyright an der App sowie der Lizenztext der verwendeten Lizenz enthalten sein.

1.3.2 Lösungsansätze zur Erfüllung der Anforderungen

Beim Start der App muss direkt als erstes geprüft werden, ob in den App-Einstellungen Logindaten hinterlegt sind. Ist das nicht der Fall muss ein Login-Fenster angezeigt werden, das den Benutzer zum Eintragen der Serveradresse sowie Benutzername und Passwort auffordert. Nach einem Klick auf einen Login-Button muss dann geprüft werden, ob mit den angegebenen Daten eine Verbindung zum Server aufgebaut und ein API-Schlüssel geholt werden kann. Ist das der Fall müssen die Server-Adresse, die Logindaten und der API-Schlüssel in den Einstellungen gespeichert werden, damit beim nächsten Start nicht erneut das Login-Fenster angezeigt wird. Falls die Verbindung scheitert muss eine Fehlerbehandlung stattfinden. Das Login-Fenster wird erst geschlossen, wenn erfolgreich ein API-

Schlüssel geholt wurde oder die App beendet wird. Tritt bei einer Verbindung zur API ein Fehler bei der Authentifizierung auf, zum Beispiel weil in ComicLib das Passwort oder der API-Schlüssel des Benutzers geändert wurde, werden die Login-Daten in den Einstellungen gelöscht und erneut das Login-Fenster angezeigt.

Ohne gültige Logindaten können keine Inhalte von ComicLib bezogen werden. Ein Fehlen der Logindaten würde beim ersten Start der App also zu einer leeren App führen. Da alle Funktionen der App (mit Ausnahme der „Über diese App“-Seite) auf der Sammlung aus ComicLib basieren, wäre die App effektiv funktionslos. Damit die App ihre Inhalte in vollem Umfang bereitstellen kann, ist das Erzwingen eines gültigen Logins also alternativlos. Eine Ausnahme stellt der Offline-Betrieb dar. Sofern Logindaten hinterlegt sind, wird, solange keine Netzwerk-Verbindung zum Server aufgebaut werden kann, angenommen, dass diese korrekt sind.

Zur Sicherung der Speicherung und Übertragung der Logindaten muss Verschlüsselung zum Einsatz kommen. Ein Fehlen der Verschlüsselung würde es Dritten ermöglichen, die Logindaten aus der App oder der Verbindung zum ComicLib-Server auszulesen. Bei der Verschlüsselung der Verbindung kann dem Benutzer gestattet werden, auf eigenes Risiko auf diese zu verzichten. Dies kann beispielsweise bei Verwendung eines gesicherten Netzwerks (VPN, MAC-Adressenfilter, o.ä.) erwünscht sein.

Für die Verschlüsselung zur Speicherung kann Androids Keystore Dienst genutzt werden, der das Erstellen und Speichern von kryptografischen Schlüsseln und Zertifikaten ermöglicht. Alternativ können auch zusätzliche Bibliotheken zur Verschlüsselung genutzt werden, beispielsweise *Conceal* von Facebook.[18] Da der offizielle Android Entwicklerleitfaden für Verschlüsselung Keystore empfiehlt[19] und der Dienst Bestandteil des ohnehin genutzten Android Frameworks ist, ist es nicht notwendig, eine zusätzliche Bibliothek für Verschlüsselung ins Softwareprojekt einzubinden.

Für die Speicherung der Logindaten in den App-Einstellungen ist es sinnvoll, Androids Shared Preferences Dienst zu nutzen, der jeder App eigene, private In-App-Einstellungen bietet. Shared Preferences ist auf die Speicherung von Schlüssel-Wert-Paaren ausgelegt. Als Alternative zu Shared Preferences kommt eine lokale Konfigurationsdatei in Frage. Bei Verwendung einer solchen können allerdings nur Zeichenketten (Strings) gespeichert werden, wodurch sich die Notwendigkeit ergibt, die Umwandlung der Datentypen selbst zu implementieren. Shared Preferences hingegen bietet Unterstützung für alle primitiven Datentypen sowie Strings, wodurch sich ein geringerer Implementationsaufwand sowie eine ge-

[18] Vgl. (Facebook Inc., 2019)

[19] Vgl. (Google, Inc., 2019)

ringere Fehleranfälligkeit ergeben. Daher stellt Shared Preferences die bessere Alternative dar und findet im Projekt Verwendung.[20]

Zur Verschlüsselung der Verbindung zu ComicLib ist es sinnvoll, auf TLS-Verschlüsselung und HTTPS zu setzen. ComicLib unterstützt eine verschlüsselte Verbindung über HTTPS und bringt bereits ein selbstsigniertes TLS-Zertifikat sowie eine Kopie des Let's Encrypt Certbots mit.[21] Eine Alternative zur Verschlüsselung der HTTP-Verbindung mittels TLS (HTTPS) stellt die Verwendung eines VPNs dar. Da VPN jedoch den gesamten Netzwerkverkehr des Geräts verschlüsselt, hat die App keinen Einfluss auf die Nutzung dieser Art der Verbindungsverschlüsselung. Die Nutzung von VPN kann der Benutzer als zusätzliche Sicherheitsmaßnahme in den Android-Systemeinstellungen konfigurieren. Mit HTTPS steht jedoch ein spezialisiertes Standard-Protokoll zur Verschlüsselung von HTTP-Verbindungen zur Verfügung, das daher auch in der App verwendet wird. Näheres zu den Sicherheitsaspekten kann dem Kapitel *Sicherheitsaspekte* entnommen werden.

Die Synchronisierung mit dem ComicLib-Server nimmt mindestens ein paar Sekunden in Anspruch, in denen der Benutzer keine Änderungen an der lokalen Datenbank vornehmen darf. Es ist deshalb sinnvoll, dem Benutzer währenddessen eine Warteseite anzugeigen, um Synchronisationsprobleme zu vermeiden. Um dem Benutzer möglichst wenig Wartezeit zuzumuten sollte die Anzahl der automatischen Synchronisierungen minimal gehalten werden.

Ein geeigneter Lösungsansatz zur automatischen Synchronisierung besteht darin, beim App-Start einmalig eine automatische Synchronisierung vorzunehmen. Diese bringt die lokale Sammlung auf den aktuellen Stand und spiegelt die im Offlinebetrieb vorgenommenen Datenbankänderungen auf den Server. Alternativ könnte eine automatische Synchronisierung auch ausschließlich nach dem ersten erfolgreichen Login am ComicLib-Server vorgenommen werden, um sicherzustellen, dass die App mindestens einmalig mit Inhalten befüllt wird. Danach müsste der Benutzer jede weitere Synchronisierung manuell auslösen. Dieser Lösungsansatz birgt jedoch das Problem, dass sich die Inhalte von lokaler und entfernter Sammlung auf Grund zu seltener Synchronisierung stark voneinander unterscheiden können. Dadurch könnten beispielsweise Downloads fehlschlagen, da ein Comic zwischenzeitlich aus der ComicLib-Sammlung gelöscht wurde. Dieses Problem kann auch bei automatischer Synchronisierung bei jedem App-Start auftreten, jedoch ist dies auf Grund der wesentlich höheren Aktualität der lokalen Sammlung deutlich unwahrscheinlicher. Daher wurde die automatische Synchronisierung bei jedem App-Start bevorzugt.

[20] Vgl. (Google, Inc., 2019)

[21] Vgl. (Hahn, ComicLib, 2019)

Zusätzlich zur automatischen Synchronisierung kann dem Benutzer mittels eines Synchronisieren-Buttons die Möglichkeit geboten werden, manuell eine Synchronisierung anzustoßen, um lokale Datenbankänderungen augenblicklich zu übertragen. Nähere Informationen zum Thema Synchronisierung finden sich im Kapitel *Synchronisierung mit dem Server*.

ComicLib bietet nur für Comicdateien in den Formaten CBR, CBZ und PDF den vollen Funktionsumfang.[22] Diese Dateitypen wurden gewählt, da sie die am weitesten verbreiteten sind. Unterstützung für weitere freie Comicdateiformate kann nachträglich implementiert werden, falls sich eine Nachfrage dafür ergibt. Für die Android-App bietet es sich an, dieselben Dateitypen voll zu unterstützen wie bei ComicLib, um einen Bruch in der Nutzererfahrung zwischen beiden Anwendungen zu vermeiden. In der App eine andere Liste von Dateitypen zu unterstützten als in ComicLib könnte die Benutzer verwirren. Daher ist es sinnvoll, diese Listen identisch zu halten.

Für die voll unterstützten Dateiformate gibt es zusätzlich zum Funktionsumfang für alle Comics einen integrierten Comic-Reader und ein automatisches Lesezeichen für die zuletzt gelesene Seite. Damit die partiell unterstützten Comicdateiformate ebenfalls offline genutzt werden können, ist es sinnvoll, eine Option zum Öffnen der Comicdateien in einer anderen App sowie eine Teilen-Funktion einzubauen. Dadurch lassen sich diese Dateien in einer anderen App oder auf einem anderen Gerät mit Unterstützung für den Dateityp nutzen. Weitere Informationen zu diesem Thema können dem Kapitel *Lokale Nutzung der Comicdateien* entnommen werden.

1.4 Ziele des Projekts

Statt mit ComicLib nur eine Alternative zu den Web-Anwendungen der großen Comic-Anbieter zu bieten, soll eine freie Alternative zu den proprietären und DRM-geschützten Plattformen als Ganzem geschaffen werden. Wie auch ComicLib soll die Android-App freie Software sein und unter einer Open Source Lizenz stehen. Jeder soll die Software nutzen, weiterentwickeln und teilen dürfen (mehr dazu im Kapitel *Open Source*).

Aus den im Kapitel *Anforderungen an die Software* dargelegten Anforderungen sowie den im Kapitel *Lösungsansätze zur Erfüllung der Anforderungen* ermittelten Ansätzen ergeben sich folgende Teilziele des Projekts:

- Bereitstellung der Comicsammlung einer ComicLib-Instanz in einer mobilen Android-App (1)

[22] Vgl. (Hahn, ComicLib, 2019)

- Aufbaue einer Verbindung zum ComicLib-Server unter Nutzung der ComicLib-Logindaten des Benutzers (2) sowie sichere Speicherung von Benutzername, Passwort und API-Schlüssel (3)
- Verschlüsselung der Verbindung zum Server (4)
- Synchronisierung der Sammlung sowie des persönlichen Lese-Fortschritts der Sammlung zwischen Android-App und ComicLib-Server (5)
- Option zum Markieren von einzelnen Comics als (un-/an-)gelesen sowie ganzen Serien als (un-)gelesen direkt in der Sammlung (6)
- Option zum Anzeigen eines Beschreibungstexts für Verlage, Serien und Comichefte (7)
- Downloadoption für alle Comic-Dateien der Sammlung, unabhängig vom Dateiformat (8)
- Übersicht über die heruntergeladenen Comics (9)
- Leseansicht in der App für Comics in den Dateiformaten CBR, CBZ und PDF sowie automatische Lesezeichenerstellung beim Lesen der Comics (10)
- Optionen zum Öffnen und Teilen der heruntergeladenen Comicdateien mit anderen Apps (11)
- Leseliste für alle angelesenen Comics (12)
- „Über diese App“-Ansicht mit Copyright-Hinweis und Lizenztext der App sowie einer Liste der verwendeten Bibliotheken, deren Lizenzen und Links zu den Projektwebseiten (13)

1.5 Projektname

Als Name für die Software wurde ursprünglich „Manhattan“ gewählt. Die beiden größten Comicverlage der USA, Marvel und DC, hatten viele Jahrzehnte lang beide ihren Hauptsitz im New Yorker Stadtteil Manhattan.[23] Außerdem spielen viele Comics in Manhattan oder anderen Stadtteilen New Yorks, beispielsweise Spider-Man, Dare Devil oder die Fantastischen Vier. New York dient ebenfalls als Vorlage für mehrere fiktive Städte in bekannten Comicserien, darunter Gotham City[24] (Batman) und Metropolis[25] (Superman). Insgesamt ist der Name Manhattan also eng mit dem Thema Comics verbunden.

[23] Vgl. (The New York Observer, 2010) und (The San Diego Union-Tribune, 2013)

[24] Vgl. (New York Public Library, 2011)

[25] Vgl. (Publishers Weekly, 2006)

Auf Grund von rechtlichen Bedenken wurde der Name jedoch im Laufe der Projektdurchführung geändert. Eine Recherche mit Hilfe des Internetdienstes *DPMAregister*[26] des Deutschen Patent- und Markenamts (DPMA) ergab, dass es im Bereich Software mehrere Marken mit „Manhattan“ als Bestandteil des Markennamens gibt.[27] „Manhattan“ selbst ist ebenfalls als Marke in Form eines Logos geschützt.[28]

Um möglichen Verletzungen einer eingetragenen Marke aus dem Weg zu gehen, wurde statt „Manhattan“ der Name „Coboli“ gewählt. Coboli steht als Abkürzung für „**Co**mic **Bo**ok **Li**brary“ und ist nach DPMAregister-Recherche nicht als Marke geschützt.[29] Auf Grund der Namensänderungen mussten umfangreiche Änderungen am Softwareprojekt vorgenommen werden, darunter das erneute Erstellen von App-Logos, das Ändern aller Paketpfade und das Ersetzen aller Erwähnungen von „Manhattan“ im Quelltext durch „Coboli“.

1.6 Open Source

Wie bereits ComicLib soll auch Coboli freie Software sein. Jeder soll die Software frei nutzen, verbreiten und weiterentwickeln dürfen. Daher wird Coboli unter einer Open Source Lizenz veröffentlicht, die sicherstellt, dass der Quelltext des Programms jedem frei zur Verfügung steht. Wie bereits bei ComicLib wurde für Coboli die GNU General Public License Version 2 gewählt. Diese garantiert das Recht zur (auch gewerblichen) Nutzung, Verbreitung und Modifikation der Software. Sie enthält jedoch auch Pflichten: Software, die auf Coboli basiert, muss ebenfalls unter der GNU GPL V2 lizensiert sein. Änderungen an der Software müssen kenntlich gemacht werden und Hinweise auf das Copyright des ursprünglichen Entwicklers müssen erhalten bleiben. Außerdem schließt die Lizenz Haftung und Garantie des Entwicklers für die Software aus. Diese Lizenz schützt also gleichzeitig sowohl die Interessen der Nutzer in Form von freier Nutzung, Verbreitung und Modifikation, als auch die Interessen des Entwicklers in Form von Haftungsausschluss und Bewahrung seines Copyrights. Zudem schützt die Lizenz die Software davor, gegen den Wunsch des Entwicklers ihren Status als freie Software zu verlieren, denn für modifizierte Varianten der Software muss der Quelltext bereitgestellt werden, falls diese weiter vertrieben wird.[30]

[26] https://register.dpma.de/DPMAregister/marke/einsteiger

[27] Vgl. **Error! Reference source not found.**

[28] Vgl. (Deutsches Patent- und Markenamt, 2019)

[29] Vgl. Abbildung 2 DPMAregister-Suchergebnis für "Coboli" am 07.10.2019.

[30] Vgl. (Free Software Foundation, Inc., 1991)

2 Hilfsmittel und Technologien

Im Rahmen der Projektdurchführung kommen verschiedene Hilfsmittel zur Unterstützung bei der Umsetzung zum Einsatz. Sie erleichtern die Arbeit mit den eingesetzten Technologien oder stellen diese erst zur Verfügung. Die beiden folgenden Kapitel geben einen Einblick in die verwendeten Hilfsmittel- und Technologien sowie deren Einsatzzwecke im Rahmen des Projekts.

2.1 Hilfsmittel

Die folgenden Teilkapitel geben einen Einblick in die Hilfsmittel, die bei der Umsetzung des Projekts zum Einsatz kommen. Sie dienen der Unterstützung bei Planung, Design und Entwicklung der Software sowie der anschließenden Bereitstellung und Wartung.

2.1.1 Git

Zur Versionierung des Softwareprojekts kommt Git[31] zum Einsatz. Git ist eine Software zur verteilten Versionsverwaltung, die vor allem im Bereich der Softwareentwicklung zur Verwaltung von Quelltext Verwendung findet. Änderungen an der Software werden in Git eingepflegt, um eine nachvollziehbare und rückabwicklungsfähige Historie zu schaffen. Einzelne Historien-Einträge, so genannte Commits, lassen sich nachträglich rückgängig machen oder ändern. Außerdem ist es möglich, auf Basis eines Commits vom Hauptast des Versionsbaums abzuzweigen und im neu entstandenen Nebenast Änderungen vorzunehmen und in Commits zu speichern, ohne den Hauptast zu ändern. Dadurch können Fehlerbeseitigung und Entwicklung neuer Funktionen stattfinden, ohne den Hauptast in einen instabilen Zustand zu versetzen. Ist die Entwicklung in einem Nebenast abgeschlossen, können dessen Änderungen mit Hilfe eines weiteren Commits in den Hauptast überführt und mögliche Kollisionen mit dem Quelltext des Hauptasts beseitigt werden.[32]

Im Gegensatz zu anderen Versionierungssystemen wie CVS[33] und Apache Subversion[34] findet bei Git die Speicherung des Quelltext-Depots eines Softwareprojekts (Repository) nicht auf einem zentralen Server statt, sondern verteilt. Ein Git-Depot lässt sich an viele Orte spiegeln und Änderungen an einer beliebigen, entfernten Kopie des Depots können leicht in das lokale Depot überführt werden.[35] Im Rahmen dieses Projekts wird diese Eigen-

[31] https://git-scm.com/

[32] Vgl. (Git Community, 2019)

[33] Vgl. (Free Software Foundation, Inc., 2015)

[34] Vgl. (Apache Software Foundation, 2018)

[35] Vgl. (Git Community, 2019)

schaft dazu genutzt, neben der Hauptkopie des Depots Sicherungskopien auf anderen Servern zu halten. Daher kann bei der Entwicklung des Projekts kein zentralisiertes Versionierungssystem zum Einsatz kommen und Apache Subversion und CVS scheiden aus der Auswahl aus. Neben Git gibt es mit Mercurial[36] noch eine weitere verbreitete, verteilte Versionierungssoftware. Auf Grund der Erfahrungen mit Git aus vorangegangenen Softwareprojekten wurde für dieses Projekt ebenfalls Git gewählt.

2.1.2 GitHub

Zur Bereitstellung des Quellcodes des Projekts wird GitHub[37] genutzt. Bei GitHub handelt es sich um eine Internetplattform zur Bereitstellung von und Zusammenarbeit an Quellcode. Neben der Bereitstellung und Versionsverwaltung mittels Git stellt GitHub viele weitere Funktionen rund um Softwareprojekte zur Verfügung, darunter Projektwebseiten, -wikis, Bugtracker und Kanban-Boards.[38] Für Coboli kommen in erster Linie die Versionsverwaltung und Bereitstellung mittels Git zum Einsatz. Das Hauptdepot für das Softwareprojekt liegt bei GitHub, als Backup gibt es noch ein Depot auf einem Raspberry Pi Einplatinencomputer, auf dem außerdem die Testinstanz von ComicLib liegt. Nach Veröffentlichung der Software können weitere Funktionen von GitHub zum Einsatz kommen. Der Bugtracker kann zum Sammeln von Fehlern und Problemen mit der Software genutzt werden. Benutzer und Entwickler können dort Probleme melden und diskutieren, sowie Code zur Behebung von Fehlern beitragen. Das Wiki kann dazu genutzt werden, Informationen zur Entwicklung und Nutzung der Software zur Verfügung zu stellen. Mit Hilfe von GitHub Pages kann eine Projektwebseite erstellt werden, die Coboli und seine Funktionen vorstellt.

Neben GitHub existieren noch weitere beliebte Plattformen zur Bereitstellung von quelloffenen Softwareprojekten, beispielsweise GitLab[39], Bitbucket[40] oder SourceForge[41]. GitLab stellt ein nahezu identisches Paket an Funktionen zur Verfügung und basiert ebenfalls auf Git. Bitbucket und SourceForge bieten gegenüber GitHub einen geringeren Funktionsumfang. GitHub wie auch GitLab sind im Bereich der freien und quelloffenen Software sehr beliebt und kommen häufig auch parallel zueinander zum Einsatz. Auf Grund von Vorkenntnissen im Umgang mit GitHub aus vorhergegangenen Projekten und der Tatsache, dass ComicLib bereits über GitHub bereitgestellt wird, kommt es auch hier zum Einsatz.

[36] Vgl. (Mercurial Community, 2019)

[37] https://github.com/

[38] Vgl. (Github, Inc., 2019)

[39] Vgl. (GitLab, Inc., 2019)

[40] Vgl. (Atlassian PLC, 2019)

[41] Vgl. (Slashdot Media, 2019)

2.1.3 Android Studio

Zur Entwicklung von Coboli kommt Android Studio[42] zum Einsatz. Dabei handelt es sich um die offizielle integrierte Entwicklungsumgebung (IDE) für Android. Die IDE unterstützt das Design und die Entwicklung von Android-Apps durch viele Hilfsfunktionen. Das Abhängigkeitsmanagement wird durch eine gute Integration von Gradle[43], des für Android-Apps genutzten Build-Systems, unterstützt. Neue Abhängigkeiten können häufig direkt aus dem Quelltexteditor heraus hinzugefügt werden, ohne das Buildskript manuell bearbeiten zu müssen. Dazu genügt es schon, die gerade benötigte Klasse der hinzuzufügenden Bibliothek im Quelltext zu importieren, damit Android Studio sie nach Bestätigung installiert und das Softwareprojekt aus dem Quelltext neu baut. Beim Schreiben von Quelltext unterstützt die IDE durch Syntaxhervorhebung und Textvervollständigung für Java und Kotlin sowie importierte Bibliotheken und das Android Framework. Neben der Entwicklung der Programmlogik unterstützt Android Studio auch das Design der App durch einen grafischen Editor für Benutzeroberflächen und einen Editor für Übersetzungen.

Als Alternativen zu Android Studio bieten sich Eclipse[44] oder NetBeans[45] an. Für beide IDEs stehen Erweiterungen zur Entwicklung mit Kotlin[46] und Android[47] zur Verfügung. Allerdings fällt die Unterstützung für die Android-App-Entwicklung bei beiden rudimentärer aus als in der offiziellen IDE Android Studio. Außerdem basiert Android Studio auf der IntelliJ IDEA IDE-Plattform der Firma JetBrains, die ebenfalls die für Coboli gewählte Programmiersprache Kotlin entwickeln, wodurch neue Sprachkonstrukte und -funktionen von Kotlin in Android Studio besser unterstützt werden als in den beiden anderen Entwicklungsumgebungen. Aus diesen Gründen und weil bereits Vorkenntnisse im Umgang mit anderen IDEs der IntelliJ-Plattform aus Vorprojekten – unter anderem PhpStorm bei der Entwicklung von ComicLib - bestanden, kommt bei der Entwicklung von Coboli Android Studio zum Einsatz.

2.1.4 Todoist

Um bei der Umsetzung des Projekts nicht den Überblick zu verlieren ist es notwendig, den inhaltlichen und zeitlichen Aufwand zu planen. Bei der Durchführung dieses Projekts wird zur Unterstützung dieser Aufgaben Todoist[48] genutzt. Todoist ist eine Software zur Aufgaben- und Projektplanung, die auf verschachtelten Todolisten basiert. Aufgaben lassen sich

[42] Vgl. (Google, Inc., 2019)

[43] https://gradle.org/

[44] https://www.eclipse.org/

[45] https://netbeans.org/

[46] Vgl. (JetBrains, 2019) und (JetBrains, 2018)

[47] Vgl. (Google, Inc., 2019) und (NBANDROIDTEAM, 2019)

[48] https://todoist.com/

in Projekten organisieren, mit Prioritäten in vier Stufen versehen und bis zu vier Ebenen tief verschachteln.[49] So lässt sich das Projekt beispielsweise in zwei Hauptaufgaben „Durchführung“ und „Dokumentation“ zerlegen, denen jeweils ein Fälligkeitsdatum zugeordnet werden kann. Diesen Hauptaufgaben lassen sich dann weitere Aufgaben unterordnen, im Falle der „Durchführung“ zum Beispiel die zu entwickelnden Funktionen mit deren Subkomponenten als Unteraufgaben. Neben der groben Planung der Teilaufgaben und deren Fälligkeiten sowie Prioritäten lassen sich in diesem System auch kleine Aufgaben gut unterbringen, beispielsweise „Entfernen von Test-Ausgaben“ oder „Größe von Button XY anpassen“. Dadurch können Kommentarblöcke mit Todos im Quellcode weitestgehend vermieden werden, sofern sie sich nicht an andere Entwickler richten. Insgesamt lässt sich mit diesem System mit geringem Aufwand ein großes Maß an Übersicht erreichen.

2.1.5 Postman

Coboli hängt als Client-Software von der REST Schnittstelle (API) des ComicLib Servers ab. Bei der Fehlerbehebung und Weiterentwicklung der ComicLib API kommt Postman[50] als integrierte API-Entwicklungsumgebung zum Einsatz. Die Dokumentation der ComicLib API wird ebenfalls über Postman zur Verfügung gestellt.[51] Postman ermöglicht das Entwerfen, Testen und Dokumentieren von APIs.[52] Bei der Entwicklung von Coboli kommt Postman dann zum Einsatz, wenn ein neu gefundener Fehler in der API behoben oder ein neues Feld implementiert werden muss. Im Laufe des Projekts war dies mehrfach der Fall und es mussten Änderungen in Postman getestet sowie die neuen API-Antworten dokumentiert werden.

2.2 Technologien

2.2.1 Kotlin

Bei der Entwicklung von Coboli kommt Kotlin als Programmiersprache zum Einsatz. Traditionell werden Android-Apps in Java programmiert, jedoch ist Kotlin seit Mai 2019 die von Google priorisierte Sprache bei der Entwicklung von neuen Android-Apps.[53] Dem entsprechend sind die offiziellen Schulungs- und Dokumentationsressourcen für Android sowohl für Java als auch für Kotlin verfügbar, wobei Codebeispiele standardmäßig in Kotlin verfasst sind und die Java-Variante jeweils nur als zusätzliche Alternative mit aufgeführt wird.

[49] Vgl. (Doist Ltd., 2019)

[50] https://www.getpostman.com/

[51] Vgl. (Hahn, ComicLib API Version 1, 2019)

[52] Vgl. (Postman, Inc., 2019)

[53] Vgl. (Google, Inc., 2019)

Kotlin basiert zum Teil auf Java. Kotlin-Code für Android wird beim Bauen der App in Java-Code transkompiliert und als Java-Bytecode in der Java Virtual Machine (JVM) ausgeführt, wenn die App gestartet wird. Auch seitens der Sprachkonstrukte ist Kotlin an Java angelehnt, erweitert dieses aber um einige Funktionen, die häufig von anderen modernen Programmiersprachen übernommen wurden. Eine der häufigsten Fehlerarten in Java ist beispielsweise die Null-Pointer-Exception, welche in Kotlin kaum auftaucht, da Kotlin zwischen *null*-baren und nicht-*null*-baren Typen unterscheidet und den Entwickler so dazu zwingt, die Verhaltensweise für den Fall, dass eine Variable *null* ist, explizit anzugeben. Mapping-, Filter- und Iterator-Funktionen, wie sie beispielsweise aus PHP bekannt sind, sowie Datenklassen mit stark vereinfachter Konstruktorsyntax tragen dazu bei, dass Kotlin-Programme mit wesentlich weniger Zeilen Code auskommen. Insgesamt sind in Kotlin viele moderne Sprachkonstrukte und sinnvolle Neuerungen gegenüber Java eingeflossen, die die Arbeit mit Kotlin gegenüber Java angenehmer und effizienter gestalten.[54]

Da Kotlin der neue, offizielle Standard für die Entwicklung von Android-Apps ist und das Schreiben von Quelltext mit Kotlin wesentlich zügiger und unkomplizierter von Statten geht, wurde für dieses Softwareprojekt Kotlin gegenüber Java der Vorzug gegeben.

2.2.2 Android Framework

Als Android-App macht Coboli intensiven Gebrauch vom Android Java API Framework, das Zugriff auf die Systemdienste von Android gewährt. Dazu gehören das View-System, die Ressourcen-, Benachrichtigungs- und Activity-Manager sowie die Content Provider.[55] Das View-System stellt die Funktionen zur Darstellung von Benutzeroberflächen bereit, während der Activity-Manager für den Übergang zwischen Activities (Hauptsichten der Benutzeroberfläche) zuständig ist. Der Ressourcen-Manager stellt Ressourcen wie Designs, Icons und lokalisierte Zeichenketten zur Verfügung. Der Benachrichtigungsprovider ermöglicht das Erstellen und Verwalten von Benachrichtigungen und kommt zur Darstellung des Downloadfortschritts innerhalb von Coboli zum Einsatz. Mit Hilfe von Content Providern kann Coboli anderen Apps Zugriff auf Dateien im privaten Speicherbereich der App gewähren, wodurch in Coboli heruntergeladene Comics in Apps von Drittanbietern geöffnet werden können. Neben den bereits genannten Komponenten kommen noch der Android Keystore und die Shared Preferences zum Einsatz. Erstere Komponente dient dem Erstellen und Speichern von kryptografischen Schlüsseln und Zertifikaten, die zur Verschlüsselung von Logindaten und Datenübertragungen zum Einsatz kommen. Die Shared Preferences dienen der Speicherung von Einstellungen innerhalb der App und werden in Coboli zur Speicherung der Serveraddresse sowie der Logindaten genutzt.

[54] Vgl. (JetBrains, 2019)

[55] Vgl. (Google, Inc., 2019)

2.2.3 Gradle

Gradle ist das Standard-Buildsystem in Android Studio sowie für Android-Apps. Das Bauen von Android-Apps aus dem Quelltext und den Ressourcen ist eine komplexe Angelegenheit. Meistens müssen viele externe Bibliotheken eingebunden werden. Bei der Nutzung von Kotlin muss außerdem der Kotlin-Code zunächst in Java- und dann in Java-Byte-Code transkompiliert werden. Aus den kompilierten Dateien muss im Anschluss eine Archivdatei im APK-Format gebaut werden, welche noch zusätzlich kryptografisch signiert werden muss. Insgesamt ist zum Bauen von Android-Apps aus dem Quelltext also ein Build-System notwendig, das die Komplexität des Build-Vorgangs auf ein handhabbares Maß reduziert. Diese Aufgabe erfüllt Gradle. Android Studio bringt bereits vorkonfigurierte Gradle-Buildskripte und Ausführungskonfigurationen mit, so dass ein neu erstelltes Projekt mit nur einem Klick gebaut, installiert und auf dem Testgerät ausgeführt werden kann.[56] Trotzdem ist es häufig, gerade bei Verwendung vieler Bibliotheken, noch notwendig, selbst Änderungen am Gradle-Buildskript der App vorzunehmen, da manche Bibliotheken oder Versionen mit einander inkompatibel sind. Jedoch unterstützt Android Studio auch hier häufig mit Hilfetexten und vorkonfigurierten Aktionen zum schnellen Ändern der Buildkonfiguration.

2.2.4 ComicLib

Coboli baut als Client-Software auf dem ComicLib Server auf. Diese Web-App zur Verwaltung einer digitalen Comicsammlung entstand im Rahmen des Praxisprojekts und stellt neben der Weboberfläche zur Verwaltung der Sammlung auch einen Lesemodus zur Verfügung. Die Comics in ComicLib werden unter Zuhilfenahme der ComicVine Comicdatenbank[57] um Metadaten wie beispielsweise die Zuordnung von Heften zu Serien sowie Serien zu Verlagen angereichert und dem Nutzer als Sammlung präsentiert. Die Comicsammlung kann in Form einer Verlagsliste sowie in Form einer Liste aller Serien durchstöbert werden. Dabei kann der Benutzer dank der Cover-Bilder der Serien und Hefte eine informiertere Entscheidung treffen als bei der Verwendung einfacher Ordnerstrukturen ohne Vorschau auf den Inhalt der Comics. ComicLib ist als freie Alternative zu den Katalogen kommerzieller Anbietern digitaler Comics konzipiert. Die Benutzer sollen ihre Comics selbst speichern und verwalten können, ohne auf den Komfort einer durchstöberbaren Sammlung mit Inhaltsvorschauen und Zusatzinformationen verzichten zu müssen. Für Comics in freien Dateiformaten bietet ComicLib zudem einen Lesemodus an, der das Lesen direkt im Webbrowser ermöglicht. Dieser speichert automatisch die zuletzt gelesene Seite für ein späteres Weiterlesen. Angelesene Comics werden außerdem in einer Leseliste festgehalten, um dem Benutzer ein leichtes Wiederfinden noch nicht abgeschlossener Hefte zu ermöglichen. Comics in

[56] Vgl. (Google, Inc., 2019)

[57] Vgl. (CBS Interactive Inc., 2019)

proprietären Dateiformaten können durch ComicLib nicht entpackt werden. Alle Comics können daher zusätzlich auch per Webbrowser aus ComicLib heruntergeladen werden, um sie mit einem lokal installierten Programm öffnen zu können.[58]

Neben der Weboberfläche stellt ComicLib auch eine REST-Schnittstelle (REST-API) zur Verfügung. Über diese API können andere Softwaresysteme Informationen zu den Heften, Serien und Verlagen der Sammlung abrufen, den Lesefortschritt von Serien und Heften auslesen und ändern. Zudem können Comicdateien und Coverbilder sowie Verlagslogos herunterladen werden. Zur Nutzung von ComicLib muss der Nutzer über ein Benutzerkonto in ComicLib verfügen. Im Vorlauf des Projekts wurde ComicLib um einen Mehrbenutzerbetrieb erweitert, so dass nun administrative sowie reguläre Benutzer angelegt werden können. Außerdem kann jeder Nutzer sein Passwort sowie seinen API-Schlüssel ändern. Die Authentifizierung zur Nutzung der API erfolgt unter Verwendung eines API-Schlüssels, der zuvor unter Authentifizierung durch Benutzername und Passwort von der API bezogen werden muss.[59]

[58] Vgl. (Hahn, ComicLib, 2019)

[59] Vgl. (Hahn, ComicLib API Version 1, 2019)

3 Projektdurchführung

Die folgenden Kapitel widmen sich der Durchführung des Softwareprojekts. Sie behandeln, unter anderem, Designentscheidungen, Details zur Architektur der Software, Informationen zu den umgesetzten Funktionen und Dinge aus dem näheren Projektumfeld, wie zum Beispiel die Testumgebung und Sicherheitsaspekte.

3.1 Testumgebung

Zum Testen der Funktionen von Coboli muss eine Testumgebung geschaffen werden, die dem realen Einsatzgebiet der Software ähnelt. Die Testumgebung teilt sich dabei in zwei Teilumgebungen. Zum einen muss eine Test-Instanz des ComicLib-Servers bereitgestellt werden, die die Inhalte für die App zur Verfügung stellt. Zum anderen müssen Testgeräte für unterschiedliche Geräteklassen bereitgestellt werden.

3.1.1 Testinstanzen des ComicLib-Servers

Bei der Entwicklung kommen zwei Testinstanzen des ComicLib-Servers zum Einsatz. Beide werden mit Hilfe von Docker[60] auf einem Raspberry Pi 3B[61] Einplatinencomputer ausgeführt. Als Betriebssystem dient dabei Raspbian 9.8, eine auf Debian basierende Linux-Distribution, die für den Betrieb auf Rechnern der Raspberry Pi-Familie optimiert ist.[62] Die zweite der beiden Testinstanzen kommt erst im späteren Projektverlauf zum Einsatz, nachdem die Kernfunktionen – also die in den Zielen definierten Funktionen der Software - bereits prototypisch implementiert sind. Die erste Testinstanz deckt den Rest, und damit den Großteil, des Projektzeitraums ab.

Die erste Testinstanz verwendet als Comicsammlung die Beispielsammlung[63], die zusammen mit ComicLib veröffentlicht wurde. Diese besteht aus 21 Comicserien mit insgesamt 23 Heften, die dem „Humble Comic Bundle: Doctor Who 2018 By Titan“[64], einem Comic-Paket von Humble Bundle, entnommen wurden. Dabei wurden die echten Comic-Dateien durch Platzhalter-Dateien ersetzt. Diese enthalten jeweils denselben Comic, der nur aus 10 weißen Seiten mit der jeweiligen Seitennummer in schwarzer Schrift besteht. Dieser Platzhalter-Comic ist in der Sammlung in den Dateiformaten CBR, CBZ und PDF enthalten. Durch die im Verhältnis zu den originalen Comicdateien viel geringere Dateigröße der Platzhalter

[60] https://www.docker.com/

[61] Vgl. (Raspberry Pi Foundation, 2019)

[62] Vgl. (Raspberry Pi Foundation, 2019)

[63] Vgl. (Hahn, ComicLib, 2019)

[64] Vgl. (Humble Bundle Inc., 2018)

verringern sich die Download- und Ladezeiten sowie der Speicherbedarf der Dateien enorm. Dadurch können Probleme, die womöglich bei Verwendung großer Comicdateien auftreten würden, zunächst vermieden werden, wodurch sich die Entwicklung der Kernfunktionen einfacher gestaltet.

Die zweite Testinstanz kommt erst zum Einsatz, nachdem die Kernfunktionen bereits implementiert sind. Sie enthält eine größere Comicsammlung mit mehreren Hundert echten Comics und insgesamt mehr als 50 Gigabyte Comicdateien. Da diese Sammlung nur Comics in den Dateiformaten CBR, CBZ und PDF enthält, können beliebige Elemente der Sammlung zum Testen verwendet werden.[65] Diese zweite Sammlung ermöglicht ein Testen der Software unter realitätsnahen Bedingungen. Außerdem ist es so möglich, das Verhalten der Software beim Wechseln der durch die App verwendeten ComicLib-Instanz zu testen.

Während die erste Instanz dazu dient, möglichst einfache Inhalte zum ersten Implementieren der Kernfunktionen bereitzustellen, können mit Hilfe der zweiten Instanz Fehler und Probleme der Software im Umgang mit echten Comicsammlungen entdeckt und behoben werden.

3.1.2 Testgeräte mit Android

Zum Testen der App wird ein Gerät mit Android als Betriebssystem benötigt. Android Studio bietet die Möglichkeit, statt eines echten Geräts eine virtuelle Maschine zu verwenden und mit Hilfe vordefinierter Profile echte Geräte nachzuahmen.[66] Eine virtuelle Maschine mit dem Profil eines Google Pixel 3[67] erhält beispielsweise eine grafische Ausgabe in dessen Seitenverhältnis und Bildschirmauflösung sowie einen Rahmen in Form des Geräts. Dadurch können die Optik sowie die Benutzerinteraktionen auch ohne echtes Gerät gut getestet werden. Allerdings benötigt das Testen mit einem virtuellen Gerät viel Rechenleistung und Speicher, wodurch auch die Leistung der IDE sowie die Geschwindigkeit des Build-Prozesses negativ beeinträchtigt werden. Daher wurde Coboli auf physischen statt virtuellen Geräten getestet.

Zum Testen der App kamen zwei Geräte zum Einsatz. Als Haupttestgerät dient ein Nokia 8.1 mit Android 9. Beim Nokia 8.1 handelt es sich um ein Smartphone mit einem 6,18“ großen Bildschirm im Seitenverhältnis 18,7:9 und einer Bildschirmaussparung (Notch) für Hörermuschel und Frontkamera.[68] Mit dieser Ausstattung liegt es nah am Durchschnitt aktuel-

[65] Vgl. Kapitel Lösungsansätze zur Erfüllung der Anforderungen

[66] Vgl. (Google, Inc., 2019)

[67] https://store.google.com/de/product/pixel_3

[68] Vgl. (HMD Global Oy, 2019)

ler Smartphones und eignet sich daher sehr gut als Testgerät. Die Entwicklung von Coboli findet hauptsächlich an Hand des Nokia 8.1 statt.

Um auch für Tablets eine gute Benutzererfahrung bieten zu können, kommt zusätzlich zum Nokia 8.1 auch ein Google Nexus 7 (2013), ebenfalls mit Android 9, zum Einsatz. Dabei handelt es sich um ein eher kleines Tablet mit einem 7“ großen Bildschirm im Seitenverhältnis 16:10.[69] Die Bildschirmdiagonale des Nexus 7 ist nicht viel größer als die des Nokia 8.1. Durch die unterschiedliche Skalierung werden Inhalte jedoch deutlich anders dargestellt als bei diesem. Außerdem werden Tablets häufig nicht senkrecht, sondern quer genutzt, wodurch sich das Seitenverhältnis umkehrt. Während das Nokia 8.1 als Testgerät für neue Funktionen und die Fehlerbehebung dient, wird das Nexus 7 zum Testen der Benutzeroberflächen auf größeren Bildschirmen sowie im Querformat genutzt, um diese besser an die Geräteklasse Tablet anpassen zu können.

Bei Android 9 handelt es sich nicht um die aktuelle Version des mobilen Betriebssystems. Android 10 steht seit dem 3. September 2019 für die ersten Modelle als Upgrade bereit.[70] Für die beiden Testgeräte steht jedoch noch kein Upgrade auf Android 10 zur Verfügung. Da die absolute Mehrheit der Android-Smartphones eine ältere Android-Version als 9 verwendet,[71] kann es sinnvoll sein, zum Testen ebenfalls eine ältere Version zu nutzen. Jedoch weisen ältere Versionen tendenziell mehr bekannte Sicherheitslücken auf und erhalten seltener Sicherheitsupdates. Außerdem können durch die Nutzung von Android 9 neuere Funktionen des Android-Frameworks genutzt werden, die die Nutzung der App vereinfachen oder die Performance verbessern können. Daher ist es sinnvoll, auf das Downgrade auf eine ältere, jedoch verbreitetere Version zu verzichten. Mit dem Verfügbarwerden von Android 10 für die Testgeräte ist es sinnvoll, diese zu aktualisieren, um die App für neue Geräte optimieren zu können.

3.1.3 Kritische Betrachtung der Lösung

Im Rückblick ist die Kombination zweier Test-Instanzen von Vorteil für die Entwicklung der Software gewesen. Sie ermöglichte das Testen der App sowohl unter idealen als auch unter realen Bedingungen. Die erste Entwicklung der App-Funktionen konnte durch die ideale Testumgebung beschleunigt werden. Dank der zweiten Test-Instanz konnten jedoch Fehler, die durch die idealen Bedingungen unentdeckt geblieben waren, noch im Rahmen des Projekts behoben werden.

[69] Vgl. (ASUSTeK Computer Inc., 2019)

[70] Vgl. (Google, Inc., 2019)

[71] Vgl. (Heise Medien, 2019)

Die Verwendung sowohl eines Smartphones als auch eines Tablets als Testgeräte war ebenfalls ein guter Ansatz. Bei Tests mit dem Nexus 7 stellte sich heraus, dass vor allem das Layout für die Darstellung der Details von Sammlungselementen sowie das Layout der Anmeldeseite zwar gut an das Nokia 8.1 angepasst waren, im Querformat auf dem Nexus 7 jedoch nicht korrekt angezeigt wurden. Dank der frühzeitigen Entdeckung dieser Probleme war eine Fehlerbehebung noch im Rahmen des Projekts möglich.

Insgesamt stellen die bezüglich der Testumgebung getroffenen Entscheidungen eine gute Lösung dar. Da jedoch nur Android 9 zum Testen der App verwendet wurde, wäre es möglich, dass bei Verwendung von Coboli auf älteren Android-Versionen noch Probleme auftreten. Bei der Entwicklung wurde jedoch darauf geachtet, die App bis einschließlich Android-API-Version 23 (Android 6.0) kompatibel zu halten. Daher sollten auf rund 75 Prozent der Android-Geräte keine gravierenden Kompatibilitätsprobleme auftreten.[72]

3.2 ComicLib API

Das folgende Kapitel widmet sich der Verwendung der ComicLib-API in Coboli. Neben den verwendeten API-Ressourcen werden auch Aspekte der Verfügbarkeit und der Authentifizierung an der API sowie die zur Entwicklung von Coboli durchgeführten Änderungen an der Programmierschnittstelle thematisiert.

3.2.1 Allgemeines zur ComicLib-API

Coboli bezieht alle Inhalte, die in der App bereitgestellt werden, über die REST-API einer ComicLib-Instanz. Allgemeine Informationen zu ComicLib können dem Kapitel *ComicLib* entnommen werden. Die API stellt die Comic-Sammlung der ComicLib-Instanz in Form von JSON-Datensätzen zur Verfügung. Die Ressourcen der API sowie Beispielanfragen und -antworten sind mit Hilfe von Postman dokumentiert.[73]

a) Versionierung der API-Ressourcen

Die API verwendet für die Ressourcenpfade ein Versionierungsschema. Dadurch soll verhindert werden, dass die Weiterentwicklung der API die Kompatibilität zu bestehenden Anwendungen, die auf die API zugreifen, beeinträchtigt. Zurzeit befindet sich die API noch in Version 1, welche aktiv weiterentwickelt wird. Daher gibt es noch keine anderen Versionen der API und Version 1 ist noch nicht stabil. Mit der Veröffentlichung von Coboli in Version 1.0 wird die Entwicklung der ComicLib-API in Version 1 ebenfalls abgeschlossen werden, da alle durch Coboli 1.0 benötigten Neuerungen in die API eingeflossen sein werden. Damit

[72] Vgl. (Heise Medien, 2019)

[73] Vgl. (Hahn, ComicLib API Version 1, 2019)

wird Version 1 der API stabil und Version 2 wird zur aktiv gepflegten Version. Neue Ressourcen, Felder und Funktionen dürfen nur in die jeweils neueste Version der API eingebracht werden, damit sich bestehende Anwendungen auf die Unveränderlichkeit der verwendeten, alten API-Version verlassen können und diese nicht in ihrer Funktion beeinträchtigt werden.

Die Versionsnummer der API ist jeweils Teil des Ressourcenpfades jeder API-Ressource. Der Pfad besteht dabei aus dem Pfad zur Wurzel der API auf dem ComicLib-Server, der Versionsnummer sowie dem Pfad zur (Sub-)Ressource. Für die erste der beiden Test-Installationen von ComicLib in diesem Projekt ergibt sich damit beispielsweise der Pfad https://raspberrypi:8082/api/v1/online für die API-Ressource */online*. Die API-Dokumentation enthält für jede Anfrage den Ressourcenpfad mit Platzhaltern für die Server-Adresse sowie, falls benötigt, die ID des Datensatzes. Für diese Platzhalter sind auch Beispielwerte in der Umgebungskonfiguration „Development“ enthalten.[74]

Im Folgenden werden die verwendeten API-Ressourcen, deren verwendete Felder und die daraus für Coboli resultierenden Datentypen besprochen. Da alle Ressourcen Teil der Version 1 der API sind, beginnen alle API-Pfade mit „/api/v1“. Der Einfachheit halber wird allerdings nur der Teil, der die Ressource an sich adressiert, angegeben. Aus „/api/v1/online“ wird im Folgenden also beispielsweise „/online“.

b) Allgemeiner Aufbau von API-Antworten

Alle Antworten der API enthalten im HTTP Message Body (kurz „Body“) ein JSON-Objekt, das ein „Status“ sowie ein „Content“-Objekt enthält. Die einzige Ausnahme stellt die Issue-File-Ressource (*/issues/{id}/*file) dar, welche im Body statt dessen eine Datei enthält. Das JSON-Objekt enthält die folgenden Felder:

Feldname	**Inhalt**	**Datentyp**
Status	HTTP-Status der Antwort.	Objekt
StatusCode	HTTP-Statuscode.	Integer
StatusMessage	Name des Statuscode, bspw. „Unauthorized“.	String
Content	Datensätze der Antwort.	Objekt

Tabelle 1 Allgemeiner Aufbau einer API-Antwort.

Das Status-Objekt mit den Feldern „ResponseCode“ und „ResponseMessage“ enthält den bereits im Header der Antwort enthaltenen HTTP-Status noch einmal. Dadurch soll der Zugriff auf die Status-Informationen vereinfacht werden. Außerdem kann anhand dieses Ob-

[74] (Hahn, ComicLib API Version 1, 2019)

jekts geprüft werden, ob die mit einem HTTP-Statuscode 200 („OK“)[75] erhaltene Antwort tatsächlich von ComicLib stammt, denn nur dann ist der Status auch im Body enthalten. Diesen Umstand macht sich die Online-Ressource zu nutze. Das „Content“-Objekt enthält den oder die Datensätze der Antwort oder *null*, falls kein Datensatz gefunden wurde oder die Ressource keinen Content vorsieht (z.B. bei einem Status-Code, der nicht 200 ist).

Für alle Ressourcen sind Antworten auf die Anfrage-Methoden GET, POST, PUT und DELETE implementiert.[76] Auf POST- sowie DELETE-Anfragen antwortet die API immer mit dem HTTP-Statuscode 405 („Method Not Allowed“)[77]. Falls die Ressource eine Authentifizierung voraussetzt, antwortet die API mit dem HTTP-Statuscode 401 („Unauthorized“)[78] und verwehrt den Zugang zu den angeforderten Informationen, falls die Authentifizierung ungültig ist. Die GET-Methode ist bei allen Ressourcen erlaubt. Die PUT-Methode hingegen ist nur für die Ressourcen */issues/{id}/readstatus* und */volumes/{id}/readstatus* erlaubt und führt bei allen anderen zu einer Antwort mit HTTP-Statuscode 405 („Method Not Allowed“).[79]

3.2.2 Verwendete Ressourcen der API

Coboli nutzt nicht alle Sub-Ressourcen der API. Zur Synchronisierung genügt es, die Haupt-Ressourcen Publishers (*/publishers*), Volumes (*/volumes*), und Issues (*/issues*) sowie die Sub-Ressource Issue-Readstatus (*/issues/{id}/readstatus*) zu verwenden.[80] Diese wird im Kapitel *Synchronisierung mit dem Server* ausführlich behandelt. Für die Überprüfung des Online-Status nutzt Coboli die Ressource Online (*/online*). Details zur Ermittlung des Online-Status lassen sich dem Kapitel *Verfügbarkeit der API* entnehmen. Bei der Authentifizierung an der API kommen die Ressourcen Tokens (*/tokens*)[81] und Authenticated (*/authenticated*)[82] zum Einsatz. Dieses Thema wird im Kapitel *Authentifizierung an der API* besprochen. Für den Download der Comicdateien wird außerdem die Ressource Issue-File (*/issue/{id}/file*)[83] benötigt. Der Dateidownload wird im Kapitel *Lokale Nutzung der Comicdateien* thematisiert. Die in der API verlinkten Bilder (Verlagslogos und Titelseiten) werden

[75] Vgl. (IETF Trust, 2014)

[76] Vgl. GET (IETF Trust, 2014), POST (IETF Trust, 2014), PUT (IETF Trust, 2014), DELETE (IETF Trust, 2014)

[77] Vgl. (IETF Trust, 2014)

[78] Vgl. (IETF Trust, 2014)

[79] Vgl. (Hahn, ComicLib API Version 1, 2019)

[80] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/publishers, 2019), (Hahn, ComicLib API Version 1 - /api/v1/volumes, 2019), (Hahn, ComicLib API Version 1 - /api/v1/issues, 2019), (Hahn, ComicLib API Version 1 - /api/v1/issues/{id}/readstatus, 2019)

[81] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/tokens, 2019)

[82] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/authenticated, 2019)

[83] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/issues/{id}/file, 2019)

über den Bilder-Cache von ComicLib bezogen, der auch in der Web-Anwendung zum Einsatz kommt.[84] Dieser ist nicht Teil der API.

Die JSON-codierten Inhalte der Antworten der API werden durch Retrofit und GSON in Kotlin-Objekte umgewandelt. Die Datentypen der Felder bleiben dabei erhalten.[85] Da für die Speicherung der Datensätze in der Datenbank Room zum Einsatz kommt, werden die Datentypen für die Spalten in der Datenbank automatisch durch Room von den Kotlin-Datentypen abgeleitet.[86]

a) Die Ressource Online (*/online*)

Die Ressource Online enthält im Content-Teil der Antwort keinen Inhalt. Sie dient nur der Überprüfung, ob die API der ComicLib-Instanz erreichbar ist.[87] Falls Coboli auf eine Anfrage an diese Ressource eine Antwort erhält, die den korrekten HTTP-Statuscode 200 („OK“) im Status-Objekt enthält, ist die ComicLib-API erreichbar. Coboli nutzt diese Ressource zum automatischen Umschalten des Offline-Modus welcher im Kapitel *Offline-Modus* näher erläutert wird.

b) Die Ressource Tokens (*/tokens*)

Die Ressource Tokens enthält im *Content*-Teil nur das Feld „APIKey“ mit dem Datentyp String. Dieses Feld enthält bei erfolgreicher Authentifizierung mittels HTTP Basic Authentication[88] und gültigem Benutzernamen sowie Passwort den API-Schlüssel des jeweiligen Benutzers.[89] Falls die Authentifizierung fehlschlägt ist der *Content*-Teil *null* und der HTTP-Statuscode 401 („Unauthorized“). Coboli nutzt diese Ressource, um den API-Schlüssel des angemeldeten Benutzers zu beziehen. Weitere Informationen zum Thema Authentifizierung hält das Kapitel *Authentifizierung an der API* bereit.

c) Die Ressource Authenticated (*/authenticated*)

Die Ressource Authenticated enthält, wie Online, keinen Inhalt im Content-Teil. Falls die Anfrage an die Ressource mit Bearer Token Authorization[90] und einem gültigen API-Schlüssel versehen war, antwortet diese Ressource mit einem HTTP-Status 200 („OK“). Falls die Authentifizierung fehlschlägt, wird statt dessen ein Code 401 („Unauthorized“) ge-

[84] Vgl. Caching der Bilddateien

[85] Vgl. (Google, Inc., 2019)

[86] Vgl. (Google, Inc., 2019)

[87] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/online, 2019)

[88] Vgl. (The Internet Society, 1999)

[89] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/tokens, 2019)

[90] Vgl. (IETF Trust, 2012)

sendet. Coboli nutzt diese Ressource zur Überprüfung, ob der gespeicherte API-Schlüssel noch gültig ist. Dies wird im Kapitel *Authentifizierung an der API* näher erläutert.

d) Die Ressource Issue-File (*/issues/{id}/file*)

Diese Ressource antwortet, im Gegensatz zu allen anderen, nicht mit einem JSON-Objekt im Body. Der Zugriff auf diese Ressource unter Verwendung einer gültigen Bearer Token Authorization startet stattdessen den Download der Comicdatei der *Issue* mit der im Pfad enthaltenen ID.[91] Weitere Informationen zur Nutzung dieser Ressource können dem Kapitel *Lokale Nutzung der Comicdateien* entnommen werden.

e) Die Ressource Publishers (*/publishers*)

Die Ressource Publishers stellt die Informationen zu den Verlagen der Sammlung zur Verfügung.[92] Sie wird bei der Synchronisierung mit dem ComicLib-Server verwendet, um die Verlage in die lokale Sammlung zu spiegeln. Die Ressource stellt die folgenden Felder im Content-Teil der Antwort zur Verfügung:

Feldname	Inhalt	Datentyp des Felds
ID	ID des Verlags.	String
Link	Pfad zu diesem Datensatz innerhalb der API.	String
Description	Beschreibungstext des Verlags aus der ComicVine-Datenbank.	String
ImageFileURL	Pfad zur Bilddatei mit dem Verlagslogo.	String
Name	Name des Verlags.	String
VolumesURL	Pfad zu den Serien des Verlags innerhalb der API.	String
VolumeCount	Anzahl der Serien des Verlags innerhalb der Comic-Sammlung.	Integer

Tabelle 2 Felder der API-Ressource Publishers.

In Coboli finden nur ein Teil der Felder der Ressource Verwendung. Die Inhalte der Felder „Description“, „Name“ und „VolumeCount“ werden in Activities der App angezeigt.[93] Das Feld „ID“ enthält die eindeutige ID des Verlags, welche auch in der lokalen Datenbank als Primärschlüssel dient.[94] Für den Download des Verlagslogos wird die URL aus dem Feld „ImageFileURL“ benötigt.[95]

[91] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/issues/{id}/file, 2019)

[92] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/publishers, 2019)

[93] Vgl. Kapitel Implementierte Activities

[94] Vgl. Kapitel Datenbankarchitektur

[95] Vgl. Kapitel Caching der Bilddateien

f) Die Ressource Volumes (*/volumes*)

Die Ressource Volumes stellt die Datensätze zu den Serien der Comic-Sammlung zur Verfügung.[96] Sie wird ebenso wie Publishers zur Synchronisierung mit dem Server verwendet. Die Ressource stellt die folgenden Felder im Content-Teil der Antwort zur Verfügung:

Feldname	**Inhalt**	**Datentyp des Felds**
ID	ID des Verlags.	String
Link	Pfad zu diesem Datensatz innerhalb der API.	String
Description	Beschreibungstext der Serie aus der ComicVine-Datenbank.	String
ImageFileURL	Pfad zur Bilddatei mit dem Titelbild des ersten Comichefts der Serie.	String
Name	Name der Serie.	String
StartYear	Jahreszahl des Jahrs, in dem die Serie begann.	Integer
IssuesURL	Pfad zu den Comicheften der Serie innerhalb der API.	String
IssueCount	Anzahl der Comichefte der Serie innerhalb der Comic-Sammlung.	Integer
ReadStatus	Lesestatus der Serie.	Objekt
IsRead	Lesezustand der Serie. Gelesen (true) oder ungelesen (false).	Boolean
Changed	UTC-Zeitstempel der letzten Änderung des Lesezustands.	String
Link	Pfad zu diesem Lesezustand innerhalb der API.	String
Publisher	Verlag, der die Serie herausgibt.	Objekt
PublisherID	ID des Verlags.	String
Link	Pfad zum Verlag innerhalb der API.	String

Tabelle 3 Felder der API-Ressource Volumes.

In Coboli findet nur ein Teil der Felder dieser Ressource Verwendung. Das Feld „ID“ enthält die eindeutige ID des Serien-Datensatzes, die auch in der lokalen Datenbank zur Identifizierung des Datensatzes dient.[97] Die Inhalte der Felder „Description“, „Name“, „IssueCount“ und „IsRead“ werden in Activities der App angezeigt.[98] Der Inhalt des Felds „ImageFileURL“ wird benötigt, um das Titelbild der Serie herunterladen zu können.[99] Das Feld „Changed“ wird zum Abgleich des Lesestatus der Serie zwischen Coboli und ComicLib benötigt.[100] Mit

[96] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/volumes, 2019)

[97] Vgl. Kapitel Datenbankarchitektur

[98] Vgl. Kapitel Implementierte Activities

[99] Vgl. Kapitel Caching der Bilddateien

[100] Vgl. Kapitel Synchronisierung des Lesestatus

dem Inhalt des Felds „PublisherID“ werden die Serien in der lokalen Datenbank ihren Verlagen zugeordnet.[101] Dadurch kann eine Liste aller Serien eines Verlags innerhalb der Sammlung angezeigt werden.[102]

g) Die Ressource Issues (*/issues*)

Die Ressource Issues stellt die Datensätze der Comichefte zur Verfügung.[103] Sie findet wie die Ressourcen Publishers und Volumes bei der Synchronisierung Verwendung. Die Ressource stellt die folgenden Felder im Content-Teil der Antwort zur Verfügung:

Feldname	**Inhalt**	**Datentyp des Felds**
ID	ID des Comichefts.	String
Link	Pfad zu diesem Datensatz innerhalb der API.	String
Description	Beschreibung des Comichefts aus der ComicVine-Datenbank.	String
ImageFileURL	Pfad zur Bilddatei mit dem Titelbild.	String
File	Informationen zur Comicdatei des Hefts.	Objekt
File Name	Dateiname der Comicdatei.	String
FileURL	Pfad zur Comicdatei innerhalb der API.	String
Number	Heftnummer des Hefts. Kann auch bspw. römische Zahlen enthalten.	String
Name	Name des Comichefts.	String
ReadStatus	Lesestatus des Comichefts.	Objekt
IsRead	Lesezustand des Hefts. Gelesen (true) oder ungelesen (false).	Boolean
CurrentPage	Aktuelle Seitenzahl innerhalb der Comicdatei. Automatisches Lesezeichen beim Betrachten des Comics.	Integer
Changed	UTC-Zeitstempel der letzten Änderung am Lesestatus.	String
Link	Pfad zu diesem Lesestatus innerhalb der API.	String
Volume	Serie, innerhalb derer das Heft erschien.	Objekt
VolumeID	ID der Serie.	String
Link	Pfad zur Serie innerhalb der API.	String

Tabelle 4 Felder der API-Ressource Issues.

In Coboli findet nur ein Teil der Felder dieser Ressource Verwendung. Das Feld „ID“ enthält die eindeutige ID des Heft-Datensatzes, die auch in der lokalen Datenbank zur Identifizie-

[101] Vgl. Kapitel Datenbankarchitektur

[102] Vgl. Kapitel Implementierte Activities

[103] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/issues, 2019)

rung des Datensatzes dient.[104] Die Inhalte der Felder „Description“, „Name“ und „IsRead“ werden in den Activities der App angezeigt.[105] Der Inhalt des Felds „Number“ wird zur Sortierung der Hefte in der Liste der Hefte einer Serie verwendet.[106] Das Feld „CurrentPage“ wird zur Speicherung der aktuellen Seite in der Leseansicht von ComicLib und Coboli genutzt.[107] Mit Hilfe des Felds „VolumeID“ erfolgt in der lokalen Datenbank die Zuordnung der Hefte zu den Serien, die das Anzeigen der Liste der Hefte einer Serie ermöglicht.[108]

h) Die Ressource Issue-Readstatus (*/issues/{id}/readstatus*)

Die Ressource Issue-Readstatus stellt Informationen zum Lesezustand der Comichefte zur Verfügung.[109] Coboli nutzt diese nicht, um Datensätze zu lesen, sondern um Datensätze in ComicLib zu aktualisieren. Daher ist diese Ressource die Einzige, auf die Coboli mit einer PUT-Anfrage zugreift. Der Body einer Anfrage an Issue-Readstatus enthält folgende Felder:

Feldname	**Inhalt**	**Datentyp des Felds**
IsRead	Lesezustand des Comichefts. Gelesen (true) oder ungelesen (false).	Boolean
CurrentPage	Aktuelle Seite innerhalb der Comicdatei. Automatisches Lesezeichen beim Betrachten des Comics.	Integer
Changed	UTC-Zeitstempel der letzten Änderung des Lesestatus.	String

Tabelle 5 Felder einer PUT-Anfrage an die API-Ressource Issue-Readstatus

Mit Hilfe der PUT-Anfrage wird der lokale Lesestatus eines Comics an ComicLib gesendet, falls der Zeitstempel im lokalen Datensatz aktueller ist als der im Datensatz von ComicLib.[110] In die Felder „IsRead“, „CurrentPage“ und „Changed“ werden der aktuelle Lesezustand, die aktuelle Seitenzahl und der Zeitstempel der letzten Änderung am Lesestatus eingetragen. Falls die Aktualisierung des Lesestatus im ComicLib-Server erfolgreich war, sendet die API eine Antwort mit den folgenden Feldern:

[104] Vgl. Kapitel Datenbankarchitektur

[105] Vgl. Kapitel Implementierte Activities

[106] Vgl. Kapitel Implementierte Activities

[107] Vgl. Kapitel Implementierte Activities

[108] Vgl. Kapitel Implementierte Activities

[109] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/issues/{id}/readstatus, 2019)

[110] Vgl. Kapitel Synchronisierung des Lesestatus

Feldname	Inhalt	Datentyp des Felds
IssueID	ID des Comichefts.	String
IsRead	Lesezustand des Comichefts. Gelesen (true) oder ungelesen (false).	Boolean
CurrentPage	Aktuelle Seitenzahl innerhalb der Comicdatei. Automatisches Lesezeichen beim Betrachten des Comics.	Integer
Changed	UTC-Zeitstempel der letzten Änderung am Lesestatus.	String

Tabelle 6 Felder der API-Ressource Issue-Readstatus.

Die Felder der Antwort auf eine PUT-Anfrage an Issue-Readstatus entsprechen denen der Antwort auf eine GET-Anfrage an die Ressource.[111] Die Antwort auf die PUT-Anfrage findet in Coboli zurzeit keine Verwendung.

3.2.3 Verfügbarkeit der API

Bevor die App auf die Ressourcen der ComicLib-API zugreifen kann, muss geprüft werden, ob der Zugriff auf diese möglich ist. Falls keine Verbindung zum API-Endpunkt aufgebaut werden kann, soll die App automatisch in den Offline-Modus wechseln. Dadurch werden alle Synchronisierungsversuche, sowohl automatische als auch manuelle, unterbunden. Die Erreichbarkeit der API hängt dabei von mehreren Faktoren ab.

Das Gerät benötigt eine aktive Verbindung zu einem lokalen oder mobilen Netzwerk, um eine Verbindung zum ComicLib-Server aufbauen zu können. Besteht keine Verbindung zu einem W-LAN-, LAN- oder mobilen Netzwerk (z.B. LTE), kann die App sofort in den Offline-Modus wechseln. Zum Überprüfen dieses Faktors genügt eine einfache Abfrage des Netzwerk-Verbindungsstatus des Geräts.[112] Es ist dazu keine zeitaufwändige Netzwerkoperation nötig.

Den zweiten Faktor bildet die Erreichbarkeit der IP-Adresse des Rechners mit dem ComicLib Server. Hierbei kommt es auf zwei Faktoren an: Zum einen muss auf Basis der URL des Servers mittels DNS-Namensauflösung[113] die IP-Adresse des Servers ermittelt werden. Kann dem Domainnamen keine IP-Adresse zugeordnet werden, meldet der zur Auflösung genutzte DNS-Server einen Fehler. Zum anderen muss der Server angeschaltet und mit dem Internet verbunden sein. Ist dies nicht der Fall, erhält die App innerhalb der voreingestellten Timeout-Zeitspanne keine Antwort. Um dies prüfen zu können, kommt es vor allem darauf an, eine geeignete Timeout-Zeitspanne zu definieren, da diese den Zeitaufwand für

[111] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/issues/{id}/readstatus, 2019)

[112] Vgl. (Google, Inc., 2019)

[113] Vgl. (Schnabel, DNS - Domain Name System, 2019)

die Prüfung der Verfügbarkeit festlegt. Coboli verwendet für das Überprüfen der Verfügbarkeit der ComicLib-API eine Zeitspanne von einer Sekunde. Sollte der Server zwar tatsächlich erreichbar sein, jedoch länger als eine Sekunde für die Antwort benötigen, ist die Netzwerkverbindung ohnehin so schlecht oder die Auslastung des Servers so hoch, dass mit diesen nicht gut gearbeitet werden kann und die Nutzung des Offline-Modus sinnvoller ist.

Der dritte und letzte Faktor für die Verfügbarkeit des API-Endpunkts besteht in der Verfügbarkeit der ComicLib-API auf dem Server. Auch hier kommen wieder zwei Faktoren zum Tragen. Zum einen muss auf dem Server eine Instanz von ComicLib ausgeführt werden, die die API bereitstellt. Zum anderen muss die bereitgestellte API die passende API-Version enthalten. Wird keine Instanz von ComicLib auf dem Server ausgeführt, wird die Anfrage an den Port von ComicLib durch den Rechner zurückgewiesen oder eine andere Software, die den Port verwendet, antwortet auf die Anfrage. Stellt ComicLib die benötigte Version der API nicht zur Verfügung, beispielsweise weil die ComicLib-Installation nicht aktualisiert wurde, sendet ComicLib den HTTP-Statuscode 404 („Not Found“)[114] und macht dadurch deutlich, dass die angeforderte Ressource nicht verfügbar ist. Zur Prüfung beider Faktoren muss Coboli überprüfen, ob die erhaltene Antwort mit der erwarteten Antwort bei Verfügbarkeit des API-Endpunkts übereinstimmt.

Um die Verfügbarkeitsprüfung des API-Endpunkts zu vereinfachen wurde im Rahmen dieses Projekts ComicLib um eine */online*-Ressource erweitert. Diese antwortet auf eine GET-Anfrage stehts mit einem HTTP-Statuscode 200 („OK“) im HTTP-Header sowie einer JSON-codierten Nachricht, die ebenfalls die Code-Nummer 200 sowie die Code-Nachricht „OK“ enthält. Sollte der Typ der Anfrage an /online nicht GET, sondern POST, PUT oder DELETE sein, antwortet ComicLib mit einem HTTP-Statuscode 405 („Method Not Allowed“).[115]

Um die Verfügbarkeit des API-Endpunkts zu prüfen, ermittelt Coboli zuerst, ob eine aktive Netzwerkverbindung besteht. Ist dies nicht der Fall, wechselt die App in den Offline-Modus. Besteht eine Netzwerkverbindung, versucht Coboli, von der /online-Ressource der verwendeten API-Version, z.B. */api/v1/online*, eine Antwort zu erhalten. Besitzt die Antwort den erwarteten Inhalt (HTTP-Code 200, JSON-Body, s.o.), so ist der API-Endpunkt erreichbar. Weicht der Inhalt der Antwort von der Erwartung ab oder überschreitet die Anfrage den Timeout von einer Sekunde, ist der Endpunkt nicht verfügbar und Coboli wechselt in den Offline-Modus. Weitere Informationen zum Thema *Offline-Modus* können dem gleichnamigen Kapitel entnommen werden.

[114] Vgl. (IETF Trust, 2014)

[115] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/online, 2019)

3.2.4 Authentifizierung an der API

Coboli nutzt zur Bereitstellung der App-Inhalte die ComicLib API. Da die ComicLib API eine Authentifizierung vorsieht ist es nicht möglich, Coboli ohne ein Benutzerkonto einer ComicLib-Instanz zu nutzen.[116] Daher ist es zwingend erforderlich, dass Coboli Zugriff auf die gültigen Login-Daten des Benutzers hat.

Coboli erzwingt daher einen Login an der ComicLib-API zur Nutzung der App. Beim ersten Start der App wird eine Loginseite angezeigt. Diese schließt sich nur, wenn mit den eingegebenen Anmeldedaten, bestehend aus Benutzername, Passwort und Adresse des ComicLib-Servers, erfolgreich auf die */tokens*-Ressource der ComicLib-Instanz zugegriffen werden konnte.

Die Authentifizierung läuft dabei zweistufig ab. Zunächst sendet Coboli eine GET-Anfrage mit HTTP Basic Authentication[117] auf Basis von Benutzername und Passwort an die /tokens-Ressource. Sind die gesendeten Login-Daten gültig, sendet der ComicLib-Server eine Antwort, die den persönlichen API-Schlüssel des Benutzers enthält.[118] Dieser API-Schlüssel wird für die Authentifizierung an allen weiteren API-Ressourcen mittels Bearer Token Authentication[119] genutzt.[120] Coboli speichert nun die eingegebenen Login-Daten sowie den erhaltenen Schlüssel verschlüsselt in den Shared Preferences.[121]

Die beiden verwendeten Authentifizierungsverfahren besitzen beide eigene Vor- und Nachteile. Daher kommt in ComicLib eine Kombination aus beiden zum Einsatz, um die Vorteile nutzen und die Nachteile entschärfen zu können.

Der große Vorteil bei der Verwendung von HTTP Basic Authentication auf Basis von Benutzername und Passwort besteht darin, dass der Benutzer diese Anmeldedaten selbst festlegen kann. Dadurch sind diese Logindaten leichter zu merken als automatisch generierte. Der Nachteil dieser Methode besteht jedoch in der eher geringen Sicherheit dieses Verfahrens, da Benutzer häufig zu einfache Passwörter wählen[122] oder diese wiederverwenden[123]. Sollte ein Angreifer diese Logindaten aus der Netzwerkkommunikation abgreifen können, könnte er sich damit möglicherweise auch andernorts Zugang verschaffen.

[116] Vgl. (Hahn, ComicLib API Version 1, 2019)

[117] Vgl. (The Internet Society, 1999)

[118] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/tokens, 2019)

[119] Vgl. (IETF Trust, 2012)

[120] Vgl. (Hahn, ComicLib API Version 1, 2019)

[121] Vgl. Kapitel Sicherheitsaspekte

[122] Vgl. (RedaktionsNetzwerk Deutschland GmbH, 2017)

[123] Vgl. (Süddeutsche Zeitung Digitale Medien GmbH, 2016)

Der große Vorteil bei der Nutzung einer Bearer Token Authentication besteht hingegen darin, dass die Wahl der Stärke des Schlüssels nicht dem Benutzer überlassen wird. Es kann ein langes und starkes, automatisch generiertes Zufallspasswort verwendet werden, wodurch die Sicherheit gegenüber typischen, benutzergewählten Anmeldedaten, deutlich steigt. Ein Abgreifen dieses Passworts exponiert zudem nur den Zugang zu ComicLib, nicht jedoch zu weiteren Diensten. Der große Nachteil dieser Methode besteht jedoch darin, dass der Schlüssel für den Benutzer sehr schwer einzuprägen ist.

Durch die Verbindung beider Verfahren lassen sich die Vorteile kombinieren und die Nachteile entschärfen. Durch die Nutzung der benutzerdefinierten Anmeldedaten zur Beschaffung des automatisch generierten, sicheren API-Schlüssels muss sich der Benutzer nur die Anmeldedaten merken, die er ohnehin zum Anmelden an der ComicLib-Weboberfläche nutzt. Für die weitere Kommunikation mit der API wird danach der deutlich sicherere API-Schlüssel genutzt. Dadurch werden die womöglich an anderer Stelle wiederverwendeten Anmeldedaten möglichst selten an den Server übertragen, wodurch ein Abgreifen dieser sensiblen Daten deutlich erschwert wird.

Coboli verwendet nur beim ersten Login die Kombination aus Benutzername und Passwort, um den API-Schlüssel zu holen. Für die Synchronisierung und den Download von Comicdateien kommt der API-Schlüssel zum Einsatz. Die Anmeldedaten des Benutzers werden ansonsten nur verwendet, wenn sich der API-Schlüssel geändert hat. Dadurch sollen einem möglichen Man-in-the-Middle-Angreifer[124] möglichst wenige Gelegenheiten geboten werden, die Anmeldedaten zu stehlen.

Zur Prüfung, ob der API-Schlüssel gültig ist, wurde die API-Ressource */authenticated* geschaffen. Diese antwortet auf eine GET-Anfrage mit einem HTTP-Statuscode 200 („OK“), falls der zur Authentifizierung der Anfrage genutzte API-Schlüssel gültig ist. Falls nicht, sendet die Ressource einen Statuscode 401 („Unauthorized“). Auf POST-, PUT- und DELETE-Anfragen antwortet die Ressource mit einem Statuscode 405 („Method Not Allowed“).[125]

Vor dem Starten einer Synchronisierung oder eines Downloads prüft die App erst an Hand dieser Ressource, ob der API-Schlüssel noch gültig ist. Falls sich der Schlüssel geändert hat, wird versucht, ihn mit Hilfe der */tokens*-Ressource zu aktualisieren. Schlägt die Aktualisierung fehl, wurde neben dem API-Schlüssel auch das Benutzerkennwort geändert. Daher wird in diesem Fall der Benutzer in der App abgemeldet und muss sich erneut anmelden, um die App wieder nutzen zu können.

Die Sicherheitsaspekte der */authenticated*-Ressource werden im Kapitel *Sicherheit der /authenticated API-Ressource* näher erläutert.

[124] Vgl. (Schnabel, Man-in-the-Middle, 2019)

[125] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/authenticated, 2019)

3.2.5 Änderungen an der API

Während der Entwicklung von Coboli mussten an der ComicLib-API einige Änderungen vorgenommen werden. Um schneller prüfen zu können, ob eine Netzwerkverbindung zur API aufgebaut werden kann, wurde die */online*-Ressource zur API hinzugefügt. Diese antwortet auf eine GET-Anfrage mit einer Bestätigung, dass die Ressource erreicht wurde. Dadurch wurde ein definierter Punkt zur Überprüfung der Verbindung geschaffen. Im Gegensatz zu den anderen API-Ressourcen wird für den Zugriff auf */online* keine Authentifizierung benötigt und es entsteht keine Wartezeit durch den Zugriff auf die Datenbank, da die Antwort statisch aus PHP heraus erfolgt.[126]

Um einen definierten Punkt zur Überprüfung des gespeicherten API-Schlüssels zu bieten, wurde die */authenticated*-Ressource geschaffen.[127] Während ein Überprüfen des API-Schlüssels auch an Hand der */tokens*-Ressource stattfinden kann, bietet die neue Ressource dieser gegenüber den Vorteil, dass zur Authentifizierung nicht die Anmeldedaten des Benutzers verwendet werden müssen.[128] Dadurch wird die Häufigkeit, mit der Benutzername und Passwort an den Server gesendet werden müssen, stark reduziert, wodurch es einem potenziellen Man-in-the-Middle[129] erschwert wird, diese Anmeldedaten auszuspähen. Da der API-Schlüssel zur Authentifizierung an allen API-Ressourcen verwendet wird (außer */tokens* und */online*), ist es auch möglich, diese zur Überprüfung der Gültigkeit des Schlüssels zu nutzen. Allerdings müsste dann die (Sub-)Ressource zur Überprüfung willkürlich gewählt werden und die angeforderten Datensätze würden die Ladezeit durch den Zugriff auf die Datenbank und den größeren Inhalt der Antwort deutlich erhöhen. Die Schaffung der */authenticated*-Ressource setzt diesen Problemen einen definierten Punkt für die Prüfung des Schlüssels mit minimaler Ladezeit entgegen. Den Sicherheitsaspekten dieser neu geschaffenen Ressource widmet sich das Kapitel *Sicherheit der /authenticated API-Ressource*.

Damit der Lesestatus zwischen mehreren Geräten synchronisiert werden kann, musste das *ReadStatus*-Objekt in Comicheften und Serien um ein zusätzliches Feld „Changed“ erweitert werden. In diesem wird der UTC-Zeitstempel der letzten Änderung des Lesestatus gespeichert. [130] Nähere Information zum Thema Synchronisierung des Lesestatus können dem Kapitel *Synchronisierung mit dem Server* entnommen werden.

[126] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/online, 2019)

[127] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/authenticated, 2019)

[128] Vgl. Kapitel Authentifizierung an der API

[129] Vgl. (Schnabel, Man-in-the-Middle, 2019)

[130] Vgl. (Hahn, ComicLib API Version 1 - /api/v1/issues, 2019) und (Hahn, ComicLib API Version 1 - /api/v1/volumes, 2019)

Zusätzlich zu den Neuerungen an der API wurden auch Fehlerkorrekturen vorgenommen. Einige der Datentypen aus der API waren in Strings verpackt, wodurch es nötig war, diese vor der Nutzung erst in ihren ursprünglichen Datentyp umzuwandeln. Dieses Problem entstand dadurch, dass die Inhalte dieser Felder 1:1 aus der ComicLib-Datenbank übernommen wurden. Diese Felder wurden so angepasst, dass sie nun den korrekten Datentyp enthalten. Konkret betrifft diese Änderung die Felder *CurrentPage*, *IssueCount*, StartYear, VolumeCount (jeweils Integer) sowie *IsRead* (vormals TinyInt, jetzt Boolean).[131]

3.3 Verwendete Bibliotheken

Coboli greift zur Bereitstellung seiner Funktionen auf einige zusätzliche Bibliotheken zurück. Das folgende Kapitel stellt diese im Zusammenhang mit ihrer Verwendung im Projekt vor. In der App sind die Bibliotheken in einer eigenen Übersichtsseite zusammen mit ihrer jeweiligen Lizenz einsehbar.[132]

Zur Kommunikation mit der ComicLib-API nutzt Coboli Funktionen aus Retrofit[133] und OkHttp[134]. Beide Bibliotheken werden von Square, Inc. unter einer Open Source Lizenz (Apache License Version 2) veröffentlicht.[135] Retrofit ist ein auf Web-APIs spezialisierter HTTP-Client. Mit Hilfe von Retrofit ist es sehr einfach möglich, HTTP-Anfragen an APIs zu stellen und deren Antworten in Java- oder Kotlin-Klassen umzuwandeln. Bei OkHttp handelt es sich um einen allgemeinen HTTP-Client, mit dem Anfragen über das HTTP-Protokoll durchgeführt werden können.

Coboli nutzt OkHttp in erster Linie als Client innerhalb von Retrofit. Für die Kommunikation mit einem ComicLib-Server über HTTPS unter Verwendung eines selbstsignierten TLS-Zertifikats wird ein angepasster OkHttp-Client benötigt, damit die Verbindung nicht mit einer Sicherheits-Fehlermeldung scheitert.[136] Außerdem kommt OkHttp zum Download von Bildern und Comicdateien zum Einsatz.[137]

Retrofit wird in Coboli zur Kommunikation mit allen API-Ressourcen, die mit JSON-Datensätzen antworten, verwendet. Mit Hilfe von Retrofit können APIs und deren Ressourcen in der App als Interface definiert werden. Zur Definition der Eigenschaften einer API-Anfrage, beispielsweise der zu verwendenden HTTP-Methode, des zu sendenden Headers

[131] Vgl. (Hahn, ComicLib API Version 1, 2019)

[132] Vgl. Kapitel Implementierte Activities

[133] Vgl. (Square, Inc., 2019)

[134] Vgl. (Square, Inc., 2019)

[135] Vgl. (Square, Inc., 2019) und (Square, Inc., 2019)

[136] Vgl. Kapitel Verbindung zum Server

[137] Vgl. Kapitel Caching der Bilddateien und Kapitel Lokale Nutzung der Comicdateien

oder des Anfrage-Bodys, kommen Annotationen zum Einsatz.[138] Auf Basis dieses Interfaces generiert Retrofit die zum Zugriff auf die API benötigten Funktionen. Der Entwickler muss also nur die API definieren, die Methoden zur Kommunikation mit dieser aber nicht selbst implementieren. Retrofit nutzt OkHttp als Client zum Zugriff auf die API-Ressourcen. Die erhaltenen HTTP-Nachrichten werden automatisch in Objekte deserialisiert. Dazu werden so genannte Converter genutzt, die ebenfalls Teil der Retrofit-Bibliothek sind. Es gibt unter anderem Converter für JSON, XML und primitive Datentypen.[139] In Coboli kommt der JSON-Converter zum Einsatz.

Für die Umwandlung zwischen JSON- und Kotlin-Datenobjekten wird in Coboli Gson[140] verwendet. Gson ist eine Bibliothek zur (De-)Serialisierung von JSON, die von Google, Inc. entwickelt wird. Ebenso wie Retrofit und OkHttp wird Gson unter den Bedingungen der Apache License in Version 2 veröffentlicht.[141] In der App wird Gson in erster Linie durch Retrofit zur Deserialisierung der API-Antworten genutzt. Die als Ziel der Deserialisierung verwendeten Klassen werden dazu mittels Annotationen so aufbereitet, dass Gson in der Lage ist, die Klassen-Attribute mit den Feldern eines JSON-Objekts abzugleichen.[142] Wie bereits bei Retrofit muss der Entwickler das Verhalten also nur durch Annotationen definieren, aber nicht selbst implementieren. Falls die Namen von Attribut und Feld übereinstimmen, kann sogar auf die Annotation verzichtet werden, da Gson die automatische Konvertierung zwischen verschiedenen, gängigen Namenskonventionen (Camel-Case, Snake-Case u.a.) unterstützt.[143] Neben der Nutzung in Retrofit wird Gson außerdem verwendet, um die Lizenzinformationen der in der App verwendeten Bibliotheken aus einer JSON-Datei in den App-Ressourcen zu laden.

Zur lokalen Speicherung von Datensätzen kommt Room aus Android Jetpack zum Einsatz. Android Jetpack ist eine Sammlung von Hilfsbibliotheken, die von Google für Android entwickelt werden.[144] Bei Room handelt es sich um eine Bibliothek zur persistenten Speicherung von Java- und Kotlin-Objekten in SQLite-Datenbanken.[145] Room bildet eine Abstraktionsschicht über SQLite, die dem Entwickler viele Aufgaben im Umgang mit der Datenbank abnimmt. Tabellen und Views werden in Room durch Annotation von Klassen definiert. Für den Zugriff auf die Datenbank werden so genannte Data Access Objects (DAOs) genutzt.

[138] Vgl. (Square, Inc., 2019)

[139] Vgl. (Square, Inc., 2019)

[140] Vgl. (Google, Inc., 2019)

[141] Vgl. (Google, Inc., 2019)

[142] Vgl. (Google, Inc., 2019)

[143] Vgl. (Google, Inc., 2019)

[144] Vgl. (Google, Inc., 2019)

[145] Vgl. (Google, Inc., 2019)

Dabei handelt es sich um Interfaces, die die Datenbankanfragen in Form von annotierten Funktionen enthalten und von Room in Objekte zum Zugriff auf die Datenbankressourcen umgewandelt werden. Standardanfragen sind in Room sehr leicht umzusetzen, da die Bibliothek über vordefinierte Annotationen und Convenience-Methoden für CRUD-Operationen verfügt, die keine zusätzlichen SQL-Anfragen benötigen.[146] Da die Bibliothek durch die Annotationen in der Lage ist, die Tabellenspalten mit den Klassenattributen in Beziehung zu setzen, können die Convenience-Methoden Datenobjekte annotierter Klassen direkt auf Tabellen übertragen und umgekehrt. Es ist dadurch möglich, Datenobjekte direkt in die Datenbank zu schreiben bzw. aus dieser zu lesen, ohne manuell zwischen Tabellenspalten und Klassenattributen mappen zu müssen. Trotz all dieser Vereinfachungen ist es in Room ebenfalls möglich, sehr komplexe Datenbankanfragen wie beispielsweise Updates über mehrere Tabellen umzusetzen, da Room auch eine Annotation zur Ausführung roher SQL-Anweisungen bereitstellt. Weitere Informationen zum Thema Datenbanken können dem Kapitel Datenbankarchitektur entnommen werden.

Durch die Verwendung von Retrofit, Gson und Room kann die Entwicklung von Android-Apps deutlich beschleunigt werden. Sie stellen dem Entwickler einen enormen Funktionsumfang zur Verfügung, der lediglich die Annotation von Klassen und Funktionen voraussetzt. Die Funktionen zum Zugriff auf die API, zur Deserialisierung von HTTP-Nachrichten und zum Speichern der erhaltenen Datenobjekte selbst zu implementieren bedeutet dem gegenüber einen erheblichen Mehraufwand. Die benötigten Funktionen selbst zu implementieren ist im Vergleich zur Einfachheit der Nutzung von Retrofit, Gson und Room keine gute Alternative.

Zum Entpacken von CBR-Dateien nutzt Coboli JUnRAR. Bei JUnRAR handelt es sich um eine Bibliothek zum Entpacken von RAR-Dateien.[147] Die Software wird leider seit 2012 nicht mehr aktiv weiterentwickelt.[148] Daher können mit JUnRAR nur RAR-Dateien bis einschließlich Version 4 des Dateiformats entpackt werden. Zurzeit gibt es leider keine gute Alternative zur Verwendung von JUnRAR zum Entpacken von RAR-Dateien unter Android. Daher kann Coboli keine CBR-Dateien öffnen, die auf RAR 5 (oder neuer) als Dateiformat basieren. CBR-Dateien mit einer älteren Version des RAR-Dateiformats können jedoch in Coboli zum Lesen geöffnet werden. Weitere Informationen zum Thema finden sich im Kapitel Lokale Nutzung der Comicdateien.

Um dem Benutzer die Orientierung in der App zu erleichtern verwendet Coboli Icons aus Font Awesome. Font Awesome ist eine Bibliothek zur Darstellung von Icons, die von Fonti-

[146] Vgl. (Google, Inc., 2019)

[147] Vgl. (Wagner, JUnRAR, 2012)

[148] Vgl. (Wagner, JUnRAR - Commits, 2012)

cons, Inc. entwickelt wird.[149] In Coboli kommt die Desktop-Variante von Font Awesome zum Einsatz. Bei dieser werden die Icons als Teil mehrerer Schriftarten zur Verfügung gestellt, welche unter den Bedingungen der SIL Open Font License 1.1 veröffentlicht werden.[150] Coboli verwendet ausschließlich die „Font Awesome 5 Free Solid“-Schriftart. Das App-Logo von Coboli basiert zudem auf dem „archive“-Icon von Font Awesome, das unter den Bedingungen der Creative Commons CC BY 4.0 Lizenz veröffentlicht wurde.[151]

Im Rückblick sind die verwendeten Bibliotheken insgesamt von großem Vorteil für die Entwicklung des Projekts gewesen. OkHttp und Retrofit haben die Entwicklung der API-Anbindung sehr einfach und zügig gestaltet. Dank Gson ließ sich auch die Umwandlung der empfangenen Datensätze in Kotlin-Datenobjekte leicht umsetzen. Durch die Verwendung von Room konnte der Aufwand zur Entwicklung der Datenbankarchitektur und -anbindung im Vergleich zur manuellen Umsetzung deutlich reduziert werden. Da Room für die Datenbank eine Versionierung verwendet, können außerdem in Zukunft leicht weitere Tabellen und Spalten der Datenbank hinzugefügt werden.[152] Die Verwendung von Font Awesome hat die Nutzung von zusätzlichen Icons in Coboli sehr vereinfacht. Da Font Awesome als zusätzliche Schriftart eingebunden wurde, lassen sich die Icons außerdem optimal an die Größe von benachbartem Text anpassen.

Dass JUnRAR nur RAR-Dateien bis einschließlich Version 4 des Dateiformats verwenden kann, wurde leider erst während der Umsetzung entdeckt. Zum Zeitpunkt der Entdeckung war die Implementierung der Unterstützung für CBR-Dateien jedoch bereits abgeschlossen. Daher wurde die Verwendung von JUnRAR vorerst beibehalten und die Beispielsammlung in ComicLib so angepasst, dass diese mit Coboli verwendet werden kann. Um in Zukunft alle Versionen des CBR-Formats unterstützen zu können, wäre es sinnvoll zu prüfen, ob in der App statt auf JUnRAR direkt auf UnRAR zurückgegriffen werden kann. Zwar ist UnRAR in C++ implementiert, jedoch unterstützt Android mithilfe des Native Development Kits (NDK) inzwischen auch die Verwendung von C- und C++-Quellcode und Bibliotheken.[153] Die Umsetzbarkeit der Nutzung von UnRAR zu prüfen und diese wenn möglich zu implementieren, stellt eine sinnvolle Verbesserungsmöglichkeit für die Zukunft des Projekts dar.

[149] Vgl. (Fonticons, Inc., 2019)

[150] Vgl. (Fonticons, Inc., 2018)

[151] Vgl. (Fonticons, Inc., 2018)

[152] Vgl. (Google, Inc., 2019)

[153] (Google, Inc., 2019)

3.4 Verbindung zum Server

Coboli stellt die Inhalte in der App nicht selbst zur Verfügung, sondern bezieht diese über die REST-API einer ComicLib-Instanz. Um die Synchronisierung mit dem ComicLib-Server und den Download von Comic-Dateien zu ermöglichen ist es notwendig, eine Verbindung zum Server aufzubauen. Dabei kommt es auf drei Faktoren an. Die ersten beiden Faktoren, Verfügbarkeit der API und Authentifizierung an der API, wurden bereits in eigenen Kapiteln besprochen.

Den dritten Faktor stellt die Verschlüsselung der Verbindung dar, denn die Verbindung zum Server kann optional mit TLS verschlüsselt sein (HTTPS). Falls die Verbindung verschlüsselt ist, kann dazu ein gültiges, von einer offiziellen Zertifizierungsstelle herausgegebenes, oder ein ungültiges, selbstsigniertes Zertifikat zum Einsatz kommen.

ComicLib generiert für jede neue Installation ein selbstsigniertes Zertifikat beim ersten Start des Webservers. Daher ist ohne zusätzlichen Konfigurationsaufwand eine HTTPS-gesicherte Verbindung zum Server möglich. Das Zertifikat erzeugt jedoch beim Aufbau einer Verbindung eine Fehlermeldung, da der eingetragene Hostname ungültig ist. Zur Nutzung eines gültigen Zertifikats einer vertrauenswürdigen Zertifizierungsstelle bietet ComicLib die Option, per Let’s Encrypt Certbot ein Zertifikat zu beziehen und die Konfiguration des Webservers automatisch zur Nutzung des neuen Zertifikats anzupassen.

Neben der Verwendung einer verschlüsselten HTTPS-Verbindung ist auch die Nutzung einer ungesicherten HTTP-Verbindung zu ComicLib möglich. Da diese jedoch nicht verschlüsselt ist, kann nur davon abgeraten werden, diese zu nutzen. Ein potenzieller Angreifer könnte sich bei einer ungesicherten Verbindung leicht zwischen Server und Benutzer schalten und so die Anmeldedaten des Benutzers ausspähen.[154]

Beim Testen der Verbindung in der Login-Ansicht muss auf Grund dieser drei Faktoren eine komplexe Fehlerbehandlung stattfinden. Zum einen müssen Fehlermeldungen angezeigt werden, falls die angegebenen Anmeldedaten ungültig sind oder keine Verbindung zum Server aufgebaut werden kann, da dieser nicht erreichbar ist (z.B., weil die Adresse nicht korrekt ist). Diese Fehler entstehen meist dadurch, dass die vom Benutzer eingegebenen Daten ungültig sind. Daher ist zur Behandlung dieser Fehler eine Fehlermeldung an den Benutzer meist ausreichend, damit dieser den Fehler selbst beheben kann (z.B. durch Korrektur des zuvor falsch eingegebenen Passworts).

Zum anderen können Fehler bezüglich der Verschlüsselung auftreten, die eine Verbindung trotz korrekter Verbindungsdaten unmöglich machen: Zertifikatfehler. Um diese verstehen

[154] Vgl. (Schnabel, Man-in-the-Middle, 2019)

zu können, ist es zunächst notwendig, die grundlegende Funktionsweise von TLS zu verstehen.

TLS verwendet beim Sitzungsaufbau asymmetrische Verschlüsselung, die auf einem öffentlichen und einem privaten Schlüssel basiert. Der Client verschlüsselt dabei seine Anfragen mithilfe des öffentlichen Schlüssels des Servers, den er beim Verbindungsaufbau von diesem erhalten hat. Der Server wiederum entschlüsselt die empfangenen Anfragen mithilfe seines privaten Schlüssels. Damit der Client sicherstellen kann, dass der erhaltene, öffentliche Schlüssel auch tatsächlich dem Server und nicht etwa einem Dritten, der die Verbindung belauscht, gehört, sendet der Server den Schlüssel als Teil eines Zertifikats.[155]

Dieses Zertifikat hat der Server zuvor von einer Zertifizierungsstelle erhalten. Neben dem Hostnamen des Servers und dem öffentlichen Schlüssel enthält das Zertifikat unter anderem auch Informationen zum Antragsteller und der Zertifizierungsstelle. Dabei bürgt die Zertifizierungsstelle dafür, dass der Server tatsächlich den im Zertifikat genannten Hostnamen hat. Der Client kann nun prüfen, ob er der Zertifizierungsstelle vertraut und ob der Hostname des Servers mit dem Hostnamen im Zertifikat übereinstimmt. Ist beides der Fall, kann eine gesicherte Verbindung aufgebaut werden, indem der Client einen symmetrischen Schlüssel für die Sitzung generiert und diesen, mithilfe des öffentlichen Schlüssels verschlüsselt, dem Server mitteilt. Der anschließende Austausch von Nutzdaten wird nun mit dem symmetrischen Schlüssel verschlüsselt.[156]

Stellt der Client allerdings ein Problem mit dem Zertifikat fest, ist die Verbindung nicht vertrauenswürdig. Häufige Probleme im Umgang mit Zertifikaten sind zum Beispiel:

- Der Hostname des Servers stimmt nicht mit dem im Zertifikat überein (z.B. wegen Zugriff via IP-Adresse)
- Der Gültigkeitszeitraum des Zertifikats ist überschritten oder noch nicht erreicht
- Der Client vertraut der Zertifizierungsstelle nicht (z.B. bei selbstsignierten Zertifikaten)

Da ComicLib im Werkszustand ein selbstsigniertes Zertifikat ausliefert, kommt es in Coboli zwangsläufig zu einem Zertifikatfehler beim Verbindungsaufbau, falls das Zertifikat in ComicLib nicht ausgetauscht wurde. Für die maximale Sicherheit der Verbindung wäre es möglich, die Verwendung von selbstsignierten Zertifikaten bei Coboli nicht zu erlauben. Dadurch würde es jedoch nahezu unmöglich, Coboli nur im LAN zu nutzen, da ein ComicLib-Server ohne Internetverbindung kein valides Zertifikat einer vertrauenswürdigen Zertifizierungsstelle beziehen kann. Daher wurde bei der Entwicklung die Entscheidung getroffen, selbstsignierte Zertifikate durch den Benutzer prüfen zu lassen.

[155] Vgl. (Schnabel, SSL - Secure Socket Layer, 2019)

[156] Vgl. (Schnabel, SSL - Secure Socket Layer, 2019)

Falls ein Zertifikatfehler auftritt, zeigt Coboli dem Benutzer einen Dialog an, mit welchem dieser bestätigen kann, dass das Zertifikat trotz Fehler genutzt werden soll. Der Dialog enthält neben einer Beschreibung des Fehlers auch Details zu Antragsteller und Aussteller sowie zum Gültigkeitszeitraum des Zertifikats. Der Benutzer kann nun entscheiden, ob er das Risiko, dem Zertifikat zu vertrauen, eingehen möchte oder nicht.

Entscheidet er sich dafür, wird das Zertifikat im Zertifikatspeicher des Keystore abgelegt und der Hostname des Servers in den Shared Preferences gespeichert. Für jede weitere Verbindung zum Server wird nun durch Coboli das gespeicherte Zertifikat zum Zertifikatspeicher des HTTP-Clients (OkHttp) hinzugefügt und so als vertrauenswürdig markiert. Bei der Überprüfung des Hostnamens durch den HTTP-Client wird außerdem geprüft, ob dieser mit dem gespeicherten übereinstimmt. Durch diese beiden Maßnahmen kann eine verschlüsselte Verbindung zum ComicLib-Server auch mit einem selbstsignierten Zertifikat aufgebaut werden. Entscheidet sich der Benutzer dagegen, das Zertifikat manuell zu akzeptieren, kehrt die App hingegen zur Loginsicht zurück und er kann dort andere Verbindungsdaten eintragen.

Falls das Zertifikat gültig ist, kann die Verbindung sofort aufgebaut werden und die Loginsicht wird geschlossen. Falls die eingegebene Serveradresse kein HTTPS, sondern HTTP verwendet, zeigt Coboli einen Warndialog an, der den Benutzer auffordert, die Verwendung einer unverschlüsselten Verbindung zu bestätigen. Die Entscheidung, die Verwendung von HTTP nicht grundsätzlich zu verbieten, wurde aus ähnlichen Gründen getroffen wie die bezüglich selbstsignierter Zertifikate. In einem privaten (W-)LAN-Netzwerk ist die Verwendung von HTTPS schwierig umzusetzen und die Nutzung von Verschlüsselung möglicherweise nicht erforderlich oder gewünscht. Da der Benutzer auf die Unsicherheit der Verbindung aber zumindest hingewiesen werden soll, wurde der Warndialog implementiert.

Falls sich das Zertifikat des Servers ändert, erzeugt dies in Coboli einen Zertifikatfehler. Als Fehlerbehandlung erzwingt Coboli einen erneuten Login. Bei diesem muss die Verwendung des neuen Zertifikats bestätigt werden, falls das neue Zertifikat selbstsigniert ist. Die lokale Datenbank bleibt von diesem Vorgang unangetastet, es gehen also keine unsynchronisierten Änderungen am Lesestatus verloren.

Im Rückblick erwachsen aus den Entscheidungen bezüglich der Verbindung zum Server neben den Vorteilen auch Probleme. Die Verfügbarkeitsprüfung des API-Endpunkts erzeugt nur eine geringe Verzögerung von maximal einer Sekunde. Durch die automatische Aktualisierung des API-Schlüssels ist die Authentifizierung auch beim Ändern des Schlüssels robust. Durch die Verwendung von HTTPS mit einem von einer vertrauenswürdigen Zertifizierungsstelle herausgegebenen Zertifikat ist ein sicherer Betrieb von Coboli über das Internet möglich. Dank der Möglichkeit, ein Zertifikat mit einem Zertifikatfehler manuell zu akzeptieren, ist auch ein reiner LAN-Betrieb mit Verschlüsselung möglich. Die Option zur Verwendung von unverschlüsseltem HTTP mag für einige Anwendungszwecke sinnvoll sein, stellt

jedoch auch ein großes Sicherheitsrisiko dar. Zwar warnt Coboli explizit vor der Nutzung einer solchen, unverschlüsselten Verbindung, jedoch besteht das Risiko, dass viele ComicLib-Instanzen auf Grund der Verfügbarkeit von unverschlüsseltem HTTP in Coboli und des zusätzlichen Aufwands zur Einrichtung von HTTPS mittels Let's Encrypt in ComicLib nicht per TLS abgesichert werden. Insgesamt führten die getroffenen Entscheidungen jedoch zu einem zufriedenstellenden Ergebnis.

3.5 Datenbankarchitektur

Zur persistenten Speicherung der Daten der Comicsammlung nutzt Coboli eine SQLite-Datenbank. Coboli verwendet dabei Room, eine Bibliothek zur persistenten Speicherung von Datenobjekten, die eine Abstraktionsschicht über SQLite, bestehend aus Data Access Objects und Annotationen, bereitstellt.

3.5.1 Datenbanktabellen

Die Definition von Tabellen wird bei Room durch die Kennzeichnung von Klassen mit der *Entity*-Annotation umgesetzt. Die Festlegung des Tabellennamens und der Fremdschlüssel findet in dieser Annotation statt. Room erstellt für jedes Attribut der so gekennzeichneten Klasse eine Tabellenspalte und leitet den Datentyp der Spalte selbständig aus dem Datentyp des Attributs ab. Falls der Spaltenname vom Namen des Attributs abweichen soll, kann dies mithilfe der *ColumnInfo*-Annotation definiert werden. Soll ein Attribut der Klasse als Primärschlüssel dienen, muss dieses mit der *PrimaryKey*- sowie der *NonNull*-Annotation gekennzeichnet werden. Falls ein Attribut nicht in die Tabelle übernommen werden soll, ist es mit der *Ignore*-Annotation zu versehen.[157] Enthält die Klasse ein Datenobjekt mit mehreren Attributen, das in derselben Tabelle gespeichert werden soll, kann dies mit der *Embedded*-Annotation erreicht werden.[158] Für die Attribute des Datenobjekts gelten dieselben Bedingungen wie für die Attribute der Klasse.[159]

Coboli verwendet die vier Datenbanktabellen „Publishers" (Verlage), „Volumes" (Serien), „Issues" (Comichefte) und „CachedComics" (heruntergeladene Comicdateien). Die ersten drei dieser vier Tabellen dienen der Speicherung von Daten, die aus der ComicLib-API bezogen wurden. Die in diesen Tabellen enthaltenen Spalten und wie diese verwendet werden kann dem Kapitel *Verwendete Ressourcen der API* entnommen werden. Da die Klassen, die diese Tabellen definieren, ihre Daten direkt aus Retrofit beziehen, enthalten diese ne-

[157] Vgl. (Google, Inc., 2019)

[158] Vgl. (Google, Inc., 2019)

[159] Vgl. Abbildung 3 Ausschnitt aus der Definition der Datenbanktabelle "Volumes" in Coboli.

ben den Room- auch Gson-Annotationen, die bei der Deserialisierung der von der API erhaltenen Datensätze Verwendung finden.[160]

Die vierte Tabelle, „CachedComics“, dient der Speicherung der Informationen zu den im Comic-Cache gespeicherten Comicdateien. Für jede dort gespeicherte Datei enthält sie die folgenden Felder:

Spaltenname	**Inhalt**	**Datentyp**
IssueID	ID des Comichefts.	String
FileName	Dateiname der Comicdatei.	String
Readable	Info, ob die Datei zum Lesen entpackt werden kann.	Boolean
Unpacked	Info, ob die Datei bereits entpackt wurde.	Boolean

Tabelle 7 Spalten der Datenbanktabelle "CachedComics".

Durch die Spalten „IssueID“ und „FileName“ erfolgt die Zuordnung der Comicdatei zu ihrem Comicheft. Die Spalte „Readable“ gibt Auskunft darüber, ob Coboli in der Lage ist, die Comicdatei zu entpacken und in der Lesesicht anzuzeigen. Der Inhalt dieses Felds entscheidet darüber, ob für einen heruntergeladenen Comic die Option „Öffnen“ angezeigt wird oder nicht. Die Spalte „Unpacked“ gibt zu guter Letzt Aufschluss darüber, ob die Comicdatei bereits in den Cache entpackt wurde. Dieses Feld sorgt dafür, dass eine Comicdatei nur einmal entpackt wird und die entpackten Seite des Comics beim Löschen der Comicdatei aus dem Cache mitentfernt werden.

Zum besseren Verständnis der Tabellen und Views der Coboli-Datenbank sowie deren Beziehungen zueinander kann das folgende Entity-Relationship-Diagramm herangezogen werden. Wie aus dem Diagramm ersichtlich wird, vereinen die Views Spalten aus mehreren Tabellen in sich, um so auf spezielle Anforderungen optimierte Datenobjekte bereitstellen zu können. So wird zum Beispiel die „Issues“-Tabelle in „CachedIssues“ um Informationen zu den lokale gespeicherten Comicdateien erweitert, um z.B. in der Heftübersicht der App einen Download-Status anzeigen zu können.[161] Den Datenbankviews widmet sich das nächste Kapitel.

[160] Vgl. Abbildung 3 Ausschnitt aus der Definition der Datenbanktabelle "Volumes" in Coboli.

[161] Vgl. Kapitel *Die View „CachedIssues“*

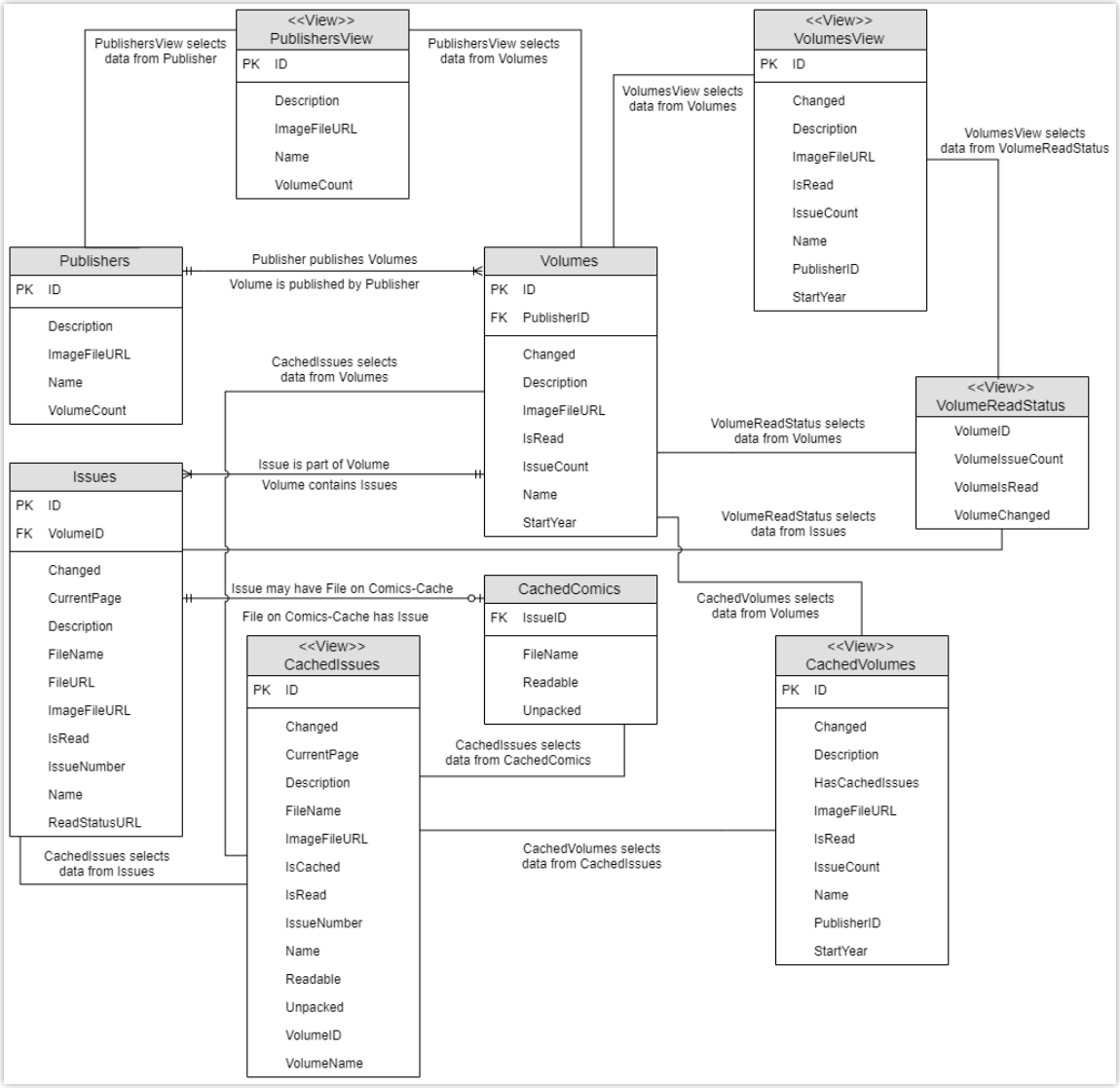

Abbildung 1 Tabellen und Views der Coboli-Datenbank

Da Coboli die Datenobjekte vor allem aus den Views bezieht, wurden diese in das Entity-Relationship-Diagramm mit einbezogen. Da es keine eigene Notationsweise für Views gibt, wurden die Notation für Entitäten um ein Stereotyp erweitert und die Beziehungen zu den Tabellen durch generische Verbindungen visualisiert.

3.5.2 Datenbankviews

Neben den vier Datenbanktabellen nutzt Coboli auch insgesamt fünf Datenbankviews. Diese fassen jeweils Spalten aus mehreren Tabellen zu einer neuen, virtuellen Tabelle zusammen. Coboli bezieht die Daten für viele der Sichten nicht direkt aus den Tabellen, sondern aus den Views. Diese erweitern die Datensätze aus den Tabellen um speziell für eine Sicht benötigte, weitere Felder. Die Felder können *Abbildung 1 Tabellen und Views der Coboli-Datenbank* entnommen werden.

Die Definition einer Datenbankview unterscheidet sich nur in einem Punkt von der einer Datenbanktabelle: An Stelle der *Entity*-Annotation wird die Klasse mit der *DatabaseView*-

Annotation versehen. In diese Annotation werden der SQL-Select-Befehl zur Erstellung der View sowie der Name der View (falls dieser vom Klassennamen abweichen soll) eingetragen.[162]

a) Die View „CachedIssues“

Die View „CachedIssues“ kombiniert Spalten aus den Tabellen „Issues“, „Volumes“ und „CachedComics“ miteinander, um die Datensätze aus „Issues“ um Informationen zu ihrem Caching-Zustand zu erweitern.[163] Die zusätzlichen Felder werden verwendet, um die Darstellung und die Optionen für Hefte in der Sammlung optimieren zu können. Das Feld „IsCached“ ist ausschlaggebend dafür, ob im Popup-Menü eines Hefts die Download- oder die Löschen-Option angezeigt wird. Der Inhalt des Felds „Readable“ entscheidet darüber, ob die Öffnen-Option im Popup-Menü angezeigt wird, welche die Lesesicht in Coboli öffnet. In der Leseliste werden die Hefte anhand des Felds „VolumeName“ sortiert, um die Hefte einer Serie als Gruppe anzuzeigen. Außerdem werden die Datensätze für die Lesesicht aus dieser View bezogen, da sie den Lesestatus und die Informationen zur heruntergeladenen Comicdatei in sich vereint.

b) Die Views „VolumeReadStatus“ und „VolumesView“

Die View „VolumeReadStatus“ wird benötigt, um die Informationen der „Volumes“-Tabelle, die auf den Heften der Serie basieren, zu ermitteln. Die Felder „IssueCount“, „IsRead“ und „Changed“ basieren direkt auf den Heften der Serie und müssen über SQL-Anfragen berechnet werden. Die View „VolumeReadStatus“ berechnet diese Felder und stellt sie als virtuelle Tabelle bereit.[164]

Die View „VolumesView“ kombiniert die Felder der Tabelle „Volumes“ mit denen der View „VolumeReadStatus“ und setzt sie so zusammen, dass ihre Spalten denen der Tabelle „Volumes“ entsprechen. Dadurch können der Lesestatus der Serie und die Heftanzahl auf Basis der lokalen Comicsammlung bestimmt werden.[165]

„Volume“-Datensätze werden in Coboli nur aus „VolumesView“ gelesen. Die Tabelle „Volumes“ kommt nur beim Einfügen, Aktualisieren und Löschen von Datensätzen zur Verwendung. Da sich der Lesestatus in „Volumes“ auf alle Hefte der Serie beziehen, erfolgt ein Update desselben nur über die Tabelle „Issues“.

Die Verwendung von zwei Views bietet den Vorteil, dass die jeweiligen SQL-Select-Anfragen zur Befüllung der Views überschaubar bleiben und die Select-Anfragen im Data

[162] Vgl. (Google, Inc., 2019) und *Abbildung 4 Ausschnitt aus der Definition der Datenbankview "PublisherView"*

[163] Vgl. *Abbildung 1 Tabellen und Views der Coboli-Datenbank*

[164] Vgl. *Abbildung 1 Tabellen und Views der Coboli-Datenbank*

[165] Vgl. *Abbildung 1 Tabellen und Views der Coboli-Datenbank*

Access Object für „Volumes“ nur auf eine View statt auf mehrere Tabellen zugreifen müssen. Dadurch bleiben die Select-Anfragen im Data Access Object sehr simpel und dadurch gut wartbar.

c) Die View „PublishersView“

Die View „PublishersView“ erfüllt einen ähnlichen Zweck wie „VolumeReadStatus“ und „VolumesView“. Das Feld „VolumeCount“ basiert auf den Serien des Verlags in der lokalen Sammlung. Daher übernimmt „PublishersView“ die Spalten aus der „Publishers“ Tabelle und ersetzt nur den Wert von „VolumeCount“ aus „Publishers“ mit dem auf Basis der Serien errechneten. Die Tabelle „Publishers“ wird in Coboli nur zum Schreiben (Einfügen, Aktualisieren, Löschen) genutzt, während „PublishersView“ zum Lesen der Verlagsdatensätze genutzt wird.[166]

d) Die View „CachedVolumes“

Die View „CachedVolumes“ wird benötigt, um die heruntergeladenen Comics serienweise in der Übersicht der heruntergeladenen Hefte anzeigen zu können. Sie erweitert die View „VolumesView“ um eine zusätzliche Spalte „HasCachedIssues“ (Boolean), welche Auskunft darüber gibt, ob mindestens ein Comic der Serie zurzeit im Comic-Cache gespeichert ist.[167] Durch die Verwendung einer View statt einer erweiterten Select-Anfrage im Data Access Object wird die Komplexität der Anfragen im DAO geringgehalten.

3.5.3 Data Access Objects

Der Zugriff auf die Datenbank erfolgt bei Room unter Verwendung so genannter Data Access Objects. Diese DAOs definieren Interfaces zur Kapselung von Datenbankanfragen in einem Objekt. In Coboli existieren vier DAOs, die jeweils einer der vier Tabellen zugeordnet sind.

Room verfügt über Annotationen für Convenience-Methoden, die es ermöglichen, einfache CRUD-Operationen ohne eigenen SQL-Code durchzuführen.[168] Coboli macht von diesen Methoden, wo es möglich ist, gebrauch. Im DAO werden Funktionsköpfe definiert und mit den Room-Annotationen versehen. Dabei ermittelt Room die von der jeweiligen Funktion betroffenen Datenbanktabellen selbständig anhand der Klasse der Eingabe.

[166] Vgl. *Abbildung 1 Tabellen und Views der Coboli-Datenbank*

[167] Vgl. *Abbildung 1 Tabellen und Views der Coboli-Datenbank*

[168] Vgl. (Google, Inc., 2019)

Die *Insert*-Annotation fügt einen oder mehrere Datensätze in die Datenbank ein, während die *Update*-Annotation den übergebenen Datensatz in der Datenbank aktualisiert. Zum Löschen eines Datensatzes aus der Datenbank findet die *Delete*-Annotation Verwendung.[169]

Für komplexe SQL-Anfragen sowie Select-Anfragen steht die *Query*-Annotation zur Verfügung.[170] In Coboli kommt diese Annotation sehr häufig zum Einsatz, da viele Selects getätigt werden müssen. Außerdem können die Datensätze, die mithilfe der Views generiert werden, häufig nicht mithilfe der Convenience-Methoden gespeichert werden, da sie mehrere Tabellen betreffen. Dies betrifft zum Beispiel den Lesestatus in „Volume“-Datensätzen. Eine Änderung an diesem betrifft den Lesestatus jedes Hefts der Serie, weshalb die Speicherung des Lesestatus nicht in „Volumes“, sondern in „Issues“ erfolgen muss.

3.5.4 Kritische Betrachtung der Lösung

Im Rückblick konnte durch die Nutzung von Room der zeitliche Aufwand im Vergleich zur manuellen Implementierung deutlich verringert werden. Die Datenbanktabellen enthalten nur die Felder der API-Antworten, welche auch tatsächlich benötigt werden, wodurch der Speicherbedarf möglichst gering gehalten wird. Die Datenbankviews vereinfachen zum Teil die SQL-Anfragen und ermöglichen die Nutzung spezialisierter Datensätze. Es wäre jedoch möglich, die Views „VolumeReadStatus“ und „VolumesView“ zu einer View zusammenzufassen und somit eine View einzusparen. Insgesamt stellen die bezüglich der Datenbank getroffenen Entscheidungen eine gute Lösung dar.

3.6 Synchronisierung mit dem Server

Da Coboli alle Inhalte in der App von einer ComicLib-Instanz bezieht, muss eine Synchronisierung der App mit der API des ComicLib-Servers stattfinden. Die Synchronisierung besteht aus vier Bestandteilen:

- dem Angleichen der Comicsammlung in Coboli an die des Servers,
- dem Abgleichen des Lesestatus für alle Serien und Hefte,
- dem Herunterladen oder Löschen des Bilds jedes Sammlungselements bei Bedarf und
- dem Löschen heruntergeladener Comicdateien und deren Bestandteilen beim Entfernen eines Comics aus der Sammlung

[169] Vgl. (Google, Inc., 2019) und Abbildung 5 Definition des Data Access Objects "CachedComicsDao"

[170] Vgl. (Google, Inc., 2019)

3.6.1 Spiegeln der ComicLib-Datenbank

Die Comicsammlung in Coboli soll stehts der des ComicLib-Servers entsprechen. Die Entscheidungsgewalt, welche Comics Teil der Sammlung sind, soll ausschließlich beim Verwalter des ComicLib-Servers liegen. Wenn dieser Comics aus der Sammlung entfernt, sollen diese auch aus Coboli gelöscht werden.

Zu Beginn des Synchronisierungsprozesses muss geprüft werden, ob eine Verbindung zum ComicLib-Server aufgebaut werden kann. Ist dies nicht der Fall, kann der Vorgang nicht durchgeführt werden. Besteht eine Verbindung, werden direkt auf einander folgend die Listen der Verlage, Serien und Hefte vom Server heruntergeladen. Waren alle drei Anfragen erfolgreich, kann mit dem Abgleich der Sammlungen begonnen werden.

Da die Verlage den Serien und Heften hierarchisch übergeordnet sind – keine Serie ohne Verlag und kein Heft ohne Serie - beginnt der Abgleich bei diesen. Verlage, die noch nicht in der lokalen Datenbank sind, werden dieser hinzugefügt. Dadurch ist sichergestellt, dass für neue Serien bereits der entsprechende Verlag in der Datenbank gespeichert ist. Die Verlage, die bereits in der lokalen Datenbank gespeichert sind, werden mit den Datensätzen aus ComicLib aktualisiert, falls diese vorliegen. Falls für einen Verlag der lokalen Datenbank kein Datensatz aus ComicLib vorliegt, wird dieser aus der lokalen Datenbank gelöscht. Zuvor müssen jedoch die Serien des Verlags und deren Hefte gelöscht werden, da diese in der Datenbank vom Verlag abhängen und so das Löschen behindern. Durch das Verketten des Löschens reduziert sich die Anzahl der abzugleichenden Datensätze beim Abgleich der Serien und Hefte bereits zu diesem Zeitpunkt.

Im Anschluss an den Abgleich der Verlage werden die Serien abgeglichen. Das Vorgehen ist dabei weitestgehend identisch. Fehlt eine Serie noch in der lokalen Datenbank, wird sie hinzugefügt. Dadurch ist sichergestellt, dass für neue Hefte bereits die jeweilige Serie vorhanden ist. Ist die Serie in beiden Datenbanken enthalten, wird sie mit den Daten aus ComicLib aktualisiert. Falls die Serie aus ComicLib entfernt wurde, wird sie aus der lokalen Datenbank gelöscht. Bevor die Serie gelöscht werden kann, müssen jedoch erst deren Hefte aus der Datenbank gelöscht werden, da diese von der Serie abhängen und das Löschen sonst blockieren.

Nachdem der Abgleich der Serien abgeschlossen ist, werden die Hefte abgeglichen. Neue Hefte werden der lokalen Datenbank hinzugefügt, vorhandene mit den Daten aus ComicLib aktualisiert und in ComicLib gelöschte aus der lokalen Datenbank entfernt.

3.6.2 Synchronisierung des Lesestatus

Beim Abgleich der Serien und Hefte muss der Lesestatus mit dem Server synchronisiert werden. Damit festgestellt werden kann, welcher der beiden Datensätze jeweils der aktuellere ist, wurde der Lesestatus um ein Feld „Changed“ erweitert. Dieses enthält den UTC-Zeitstempel der letzten Änderung des Lesestatus.[171] Durch einen Vergleich der beiden Zeitstempel kann so leicht festgestellt werden, welcher der aktuellere ist. Der jeweils ältere Lesestatus wird nun mit dem aktuelleren überschrieben. Bei den Zeitstempeln ist es von elementarer Bedeutung, dass beide dieselbe Zeitzone verwenden, da der Zeitstempel keine Informationen zur zugrundeliegenden Zeitzone enthält. UTC wurde deshalb als Zeitzone für das Feld festgelegt, da es sich um die koordinierte Weltzeit handelt und diese somit bereits den Standard für ortsunabhängige Zeitangaben bildet.

3.6.3 Caching der Bilddateien

Coboli nutzt zur Visualisierung der Elemente der Sammlung Bilddateien, die in einem Album angezeigt werden.[172] Dabei ist jedem Element eine Bilddatei zugeordnet. Im Falle der Verlage handelt es sich dabei um das Firmenlogo des Verlags. Bei Heften wird hingegen die Titelseite des Hefts und bei Serien die Titelseite des ersten Hefts der Serie verwendet.

Beim Hinzufügen eines neuen Elements zur lokalen Comicsammlung muss daher auch das jeweilige Bild dem Bildercache hinzugefügt werden. Daher wird im Anschluss an das Einfügen des neuen Datensatzes in die Datenbank sofort der Download des Bilds in einem Hintergrundthread gestartet. Dazu wird das Feld „ImageFileURL“ des jeweiligen Datensatzes verwendet, das den Link zur jeweiligen Bilddatei auf dem ComicLib-Server enthält.

Wird ein Element aus der lokalen Comicsammlung gelöscht, so wird auch dessen Bilddatei nicht mehr benötigt. Daher wird unmittelbar vor dem Löschen des jeweiligen Datensatzes aus der Datenbank erst das Löschen der Bilddatei in einem Hintergrundthread angestoßen. Dadurch wird der Speicherbedarf des Bildercaches möglichst geringgehalten.

Die Bilddateien werden im Bildercache unter dem jeweiligen Dateinamen aus „ImageFileURL“ abgelegt. Dieser steht ganz am Ende der URL und kann leicht aus dieser ausgelesen werden. Zum Auffinden der lokalen Bilddatei muss daher nur der Dateiname aus der URL abgeleitet und dem Pfad zum Bildercache angehangen werden. Dadurch kann auf die Speicherung des Dateipfads der Bilddatei in der Datenbank verzichtet werden.

[171] Vgl. Kapitel *Änderungen an der API*

[172] Vgl. Kapitel *Implementierte Activities*

3.6.4 Caching der Comicdateien

Neben dem Bildercache muss im Zuge der Synchronisierung auch der Comiccache Beachtung finden. Auf Grund des unvorhersehbaren Umfangs der ComicLib-Sammlung und der hohen Dateigröße der Comicdateien ist ein automatischer Download aller Comicdateien nicht sinnvoll. Statt dessen kann in Coboli der Download für jeden Comic einzeln gestartet werden. Auch das Löschen der Comicdateien wird comicweise durchgeführt.

Falls ein Heft bei der Synchronisierung aus der lokalen Datenbank gelöscht wird, werden auch dessen Dateien im Comiccache gelöscht. Dies betrifft sowohl die Comicdatei als auch die aus dieser möglicherweise entpackten Einzeldateien. Da die Kontrolle über die Inhalte der Comicsammlung allein beim Verwalter des ComicLib-Servers liegen soll, muss sich die Löschung von Comicheften in ComicLib auch auf die Comicsammlung in Coboli auswirken. Die Dateien des Hefts aus dem Cache zu löschen sorgt dafür, dass in Coboli keine Überreste des Hefts zurückbleiben und der Speicherbedarf geringgehalten wird.

Die Comicdateien werden im Comiccache unter ihrem Dateinamen aus ComicLib, welcher im Header des Dateidownloads enthalten ist, abgelegt. Dadurch soll sichergestellt werden, dass bei Nutzung der „Öffnen mit“- oder „Teilen“-Option die Datei anhand ihres Dateinamens gut identifizierbar bleibt. Dies setzt allerdings voraus, dass der Verwalter von ComicLib die Comicdateien so benennt, dass neben der durch ComicLib ohnehin benötigten Heftnummer auch die Serie aus dem Dateinamen hervorgeht. Damit es nicht zu Kollisionen zwischen Dateinamen von Comics im Comiccache kommt, falls der Verwalter die Comicdateien schlecht benennt (z.B. nur mit der Heftnummer oder Heftnummer plus festes Präfix), wird dem Dateinamen aus ComicLib noch die „IssueID“ als einzigartiges Präfix vorangestellt.

3.6.5 Kritische Betrachtung der Lösung

Im Rückblick besteht bezüglich der Synchronisation mit dem Server noch Verbesserungspotenzial. Um zu vermeiden, dass der Benutzer während des Synchronisierungsvorgangs die Funktionen der App nutzt und damit bereits synchronisierte Datensätze ändert, zeigt Coboli während des Vorgangs nur eine Wartemeldung an und sperrt somit den Zugriff auf die Funktionen der App. Es wäre jedoch möglich, von dieser Sperrung abzusehen, um dem Benutzer die Wartezeit zu ersparen. Dazu müssten die Änderungen durch den Benutzer während eines Synchronisierungsvorgangs zunächst protokolliert werden, um diese erst im Anschluss an diesen in die Datenbank zu schreiben. Im Anschluss müssten die neuen Änderungen dann nachträglich mit dem Server synchronisiert werden. Außerdem müsste dafür gesorgt werden, dass die Sichten in der App dynamisch auf das Hinzufügen und Löschen von Datensätzen in der Datenbank reagieren. Die Abschaffung der Wartemeldung beim Synchronisationsvorgang ist eine gute Möglichkeit, die App in der Zukunft noch zu verbessern. Abgesehen von der Wartemeldung funktioniert die Synchronisierung mit dem Server

gut. Der Synchronisierungsvorgang wird beim App-Start automatisch durchgeführt und kann auch später manuell gestartet werden. Die Bilder werden im Hintergrund heruntergeladen, um die Wartezeiten zu verringern. Bilder und Comics werden automatisch gelöscht, um den Speicherverbrauch minimal zu halten. Insgesamt führen die Entscheidungen bezüglich der Synchronisierung mit dem Server zu einer zufriedenstellenden Lösung.

3.7 Lokale Nutzung der Comicdateien

Der wohl wesentlichste Teil der Ziele dieses Projekts besteht darin, für Comicdateien aus ComicLib Möglichkeiten zur lokalen Nutzung in Coboli zu schaffen. Die folgenden Kapitel widmen sich der Umsetzung der Teilziele, die die lokale Nutzung der Comicdateien betreffen.[173]

3.7.1 Download von Comicdateien

Coboli bietet für alle Comichefte der Sammlung die Option, die jeweilige Comicdatei zur lokalen Nutzung herunterzuladen. Über einen Eintrag im Popup-Menü des Comichefts steht daher das Herunterladen der Comicdatei zur Verfügung. Zurzeit kann der Download nur bei geöffneter App durchgeführt werden und bricht ab, sobald diese verlassen wird. Die Fortführung des Downloads auch bei Verlassen der App stellt eine sinnvolle Möglichkeit zur weiteren Verbesserung der App nach dem Ende dieses Projekts dar.

Vor dem Start des Downloads prüft Coboli zunächst, ob eine Verbindung zum ComicLib-Server hergestellt werden kann. Falls dies gelingt, wird der Downloadvorgang gestartet und die erhaltene Datei im Comiccache gespeichert. Andernfalls zeigt die App eine Fehlermeldung an. Beim Download werden die erhaltenen Daten sofort in die Zieldatei im Comiccache geschrieben, um einen Abbruch des Downloads wegen eines zu hohen Verbrauchs an Arbeitsspeicher zu vermeiden. Während der ersten Tests kam es diesbezüglich zu Problemen, denn die Comicdateien wurden zu diesem Zeitpunkt noch in den Arbeitsspeicher heruntergeladen und erst nach Abschluss des Downloads in den Comiccache geschrieben. Auf Grund dieser Erfahrungen wurde das Downloadverhalten daher umgestellt.

Coboli zeigt, während ein Download läuft, dessen Fortschritt in einer Benachrichtigung an. Beim Erzeugen der Benachrichtigung wird nur das Coboli-Logo in der Statusleiste von Android angezeigt und es werden weder ein Benachrichtigungston noch eine Vibration ausgelöst (stumme Benachrichtigung). Die Benachrichtigung kann im Mitteilungszentrum angesehen werden und enthält einen Ladebalken, der den Fortschritt des Downloads anzeigt.

[173] Vgl. Kapitel Ziele des Projekts

Nach Abschluss des Downloads wird an Stelle des Ladebalkens ein Benachrichtigungstext angezeigt, der über den Abschluss informiert.[174]

In den Heftübersichten der Serien und in der Leseliste werden noch nicht heruntergeladene Hefte durch ein Cloud-Symbol in der linken, oberen Ecke des Hefteintrags gekennzeichnet.[175] Falls die Comicdatei eines Hefts bereits im Cache liegt, wird im Popup-Menü die Herunterladen-Option durch die Löschen-Option zum Entfernen der Datei aus dem Cache ersetzt. Die Herunterladen-Option und das Cloud-Symbol werden durch den *Offline-Modus* beeinflusst - siehe dazu im gleichnamigen Kapitel.

Damit die heruntergeladenen Comics leicht identifiziert werden können, werden diese zusätzlich in einer eigenen Übersicht angezeigt. Dadurch ist es leichter möglich, die lokalen Comicdateien zu lesen oder zu löschen.[176]

3.7.2 Lesen von Comicdateien

Coboli bietet für Comics in den unterstützten Dateiformaten CBR, CBZ und PDF die Möglichkeit an, diese direkt in der App zu lesen. Dazu wird bei Heften mit diesen Dateiformaten im Popup-Menü die Option „Öffnen“ eingeblendet, welche die Lesesicht für das Heft öffnet.[177]

Beim ersten Öffnen des Comics werden die Einzeldateien aus der heruntergeladenen Comicdatei in einen mit der „ID“ des Hefts benannten Ordner im Cache entpackt. Dabei kommt für jeden der Dateitypen eine andere Methode zum Entpacken zum Einsatz. CBR-Dateien basieren auf dem RAR-Archivdateiformat und können mithilfe von JUnRAR entpackt werden. Da die Comicseiten in der Datei als Bilddateien komprimiert vorliegen, dauert das Entpacken nur wenige Sekunden. Die CBZ-Dateien basieren auf dem Zip-Archivdateiformat und können mit Java-Bordmitteln entpackt werden – es ist dazu also keine weitere Bibliothek nötig. Da auch hier die Seiten als komprimierte Bilddateien vorliegen, dauert das Entpacken von CBZ-Dateien ebenfalls nur wenige Sekunden. Für PDF-Dateien wird ebenfalls keine zusätzliche Bibliothek benötigt, da das Android-Framework bereits Möglichkeiten zur Verarbeitung von PDF-Dateien enthält. Wie bereits unter ComicLib stellt auch in Coboli das Entpacken von PDF-Dateien jedoch ein Problem dar. Die Comicseiten liegen in der PDF-Datei als Dokumentseiten vor, welche jeweils in eine Bilddatei gerendert werden müssen. Dies dauert wesentlich länger als das Entpacken der anderen beiden Formate. Im Produktiveinsatz ist mit mehr als einer Minute zum Öffnen einer rund 100-seitigen PDF-Comicdatei

[174] Vgl. Abbildung 18 Benachrichtigung über einen laufenden Download und Abbildung 19 Benachrichtigung über einen abgeschlossenen Download

[175] Vgl. Abbildung 13 *Issues-Fragment* zur Übersicht über die Hefte einer Serie mit Popup-Menü an einer Karte

[176] Vgl. Abbildung 20 Liste der Serien mit heruntergeladenen Heften auf Basis des *Volumes-Fragments*

[177] Vgl. Kapitel Reader-Activity

zu rechnen. Coboli zeigt während des Entpackens der Comicdatei eine Wartemeldung an, welche auch auf den hohen Zeitbedarf für das Entpacken von PDF-Dateien hinweist.[178] Bei jedem weiteren Öffnen des Comics in der Lesesicht werden die Bilder direkt aus dem Cache geladen, wodurch die Wartezeit entfällt.

Nachdem die Datei entpackt wurde, wird die Lesesicht mit den Seiten befüllt. Dabei werden jeweils nur die Bilder der aktuellen sowie der direkt vorhergehenden und folgenden Seite geladen, um den Arbeitsspeicherverbrauch und die Wartezeiten zu minimieren. Die Lesesicht ermöglicht es, durch horizontales Wischen durch die Seiten des Comics zu blättern. Mit dem Anzeigen einer neuen Seite wird deren Seitenzahl als „CurrentPage“ im Lesestatus hinterlegt, wodurch ein automatisches Lesezeichen erzeugt wird. Beim Öffnen eines Comics in der Lesesicht wird automatisch zur in „CurrentPage“ gespeicherten Seitennummer gesprungen, damit der Benutzer dort weiterlesen kann, wo er zuletzt aufgehört hatte. Beim Erreichen der letzten Seite wird der Comic außerdem automatisch als gelesen markiert, wodurch er aus der Leseliste verschwindet.

Um die Navigation im Comicheft zu erleichtern, bietet Coboli über ein Popup-Menü in der Lesesicht Optionen zum Springen innerhalb des Heft an.[179] Neben einer Option zum Zurückspringen zur Titelseite wird hier die Möglichkeit geboten, eine Seitenübersicht einblenden zu lassen. In dieser werden alle Seiten des Comics in Miniatur angezeigt und durch ein Tippen auf eine der Seiten kann direkt zu dieser gesprungen werden.[180] Dies ist vor allem dann sehr praktisch, wenn man eine bestimmte Stelle noch einmal nachlesen möchte.

Damit der Benutzer den Überblick über seine angelesenen Comics behält, stellt Coboli eine Leseliste zur Verfügung.[181] In dieser werden alle Comichefte angezeigt, die als ungelesen markiert sind und deren aktuelle Seitenzahl nicht „0“ (also die Seitenzahl des Coverbilds) ist. Angelesene Hefte, die nicht im Comiccache gespeichert sind, werden wie in der Heftübersicht der Serien mit einem Cloud-Symbol markiert.

Damit auch Comics mit Comicdateien in nicht unterstützten Dateiformaten von der Leseliste Gebrauch machen können, wurden dem Popup-Menü der Hefte Optionen zum Ändern des Lesestatus hinzugefügt. Mit diesen lassen sich die Comics als ungelesen, angelesen sowie gelesen markieren. Für Serien gibt es zudem in deren Popup-Menü die Option, deren Hefte kollektiv als gelesen oder ungelesen zu markieren. Bei beiden Menüs wird stets die Markieren-Option des aktuellen Status ausgeblendet.

[178] Vgl. Abbildung 23 Wartemeldung in der *Reader-Activity* beim Entpacken einer Comicsdatei zum Anzeigen

[179] Vgl. Abbildung 24 Popup-Menü als Overlay der *Reader-Activity*

[180] Vgl. Abbildung 25 Seitenübersicht in der *Reader-Activity*

[181] Vgl. Abbildung 26 Leseliste auf Basis des *Issues-Fragment*

3.7.3 Nutzung mit anderen Apps

Da Coboli nur wenige Dateiformate unterstützt, wurden für die Comicdateien in anderen Dateiformaten Ausweichmöglichkeiten geschaffen. Diese können jedoch auch mit Comicdateien in den unterstützten Formaten genutzt werden. Damit alle Comics in der Leseliste angezeigt werden können, wurde als Ergänzung zum automatischen Lesezeichen des Lesemodus das manuelle Markieren eines Comics als ungelesen, angelesen sowie gelesen ermöglicht. Dadurch können Comics, die in einer anderen App geöffnet werden müssen, trotzdem als angelesen markiert werden und so auf der Leseliste erscheinen.

Damit alle Comicdateien auch in anderen Apps, die deren Dateiformat unterstützen, geöffnet werden können, wurden die „Öffnen mit“- und die „Teilen“-Funktion umgesetzt und als Optionen dem Popup-Menü der Comichefte hinzugefügt.

Mithilfe der „Öffnen mit“-Funktion können Comicdateien direkt in einer App mit Unterstützung für das Dateiformat der Comicdatei geöffnet werden. Dies setzt allerdings voraus, dass die App nicht nur das Dateiformat unterstützt, sondern auch das Öffnen der Datei durch den Intent „ACTION_VIEW“.[182] Beim Testen dieser Funktion mit einer Amazon Kindle Datei (AZW-Format) war das Öffnen mit der installierten Amazon Kindle App beispielsweise nicht möglich, obwohl diese das Dateiformat selbst nutzt. Comicdateien in den unterstützten Dateiformaten ließen sich mithilfe dieser Funktion jedoch problemlos in anderen Apps öffnen.

Die „Teilen“-Funktion ermöglicht es, die Comicdatei mithilfe einer anderen App zu versenden.[183] Das Ziel kann dabei ein anderes Gerät, eine andere Person oder der lokale Speicher des Geräts sein. Manche Apps, zum Beispiel Amazon Kindle, bieten die Möglichkeit, Dateien durch das Ablegen in einem bestimmten Verzeichnis in die App zu laden. Das „Teilen“ der Comicdatei mit dem lokalen Speicher ermöglicht es daher, diese Option zu nutzen. Leider war die Kindle-App auch auf diesem Wege nicht in der Lage, die AZW-Datei zu öffnen. Die so abgelegte Comicdatei wurde in Kindle zwar angezeigt, allerdings ließ sie sich nicht öffnen, da Kindle beim Laden des Inhalts stehen blieb und nur eine Vorschau auf das Titelbild des Comics zeigte. Es gibt daher leider zurzeit keine Möglichkeit, Kindle-Dateien mit Coboli zu nutzen. Stattdessen muss der Download der Comics direkt in Kindle geschehen. Der Lesestatus von Kindle-Dateien lässt sich aber trotzdem in Coboli verwalten. Tests der Teilen-Funktion mit Comics in den unterstützten Formaten liefen hingegen problemlos.

[182] Vgl. (Google, Inc., 2019)

[183] Vgl. Abbildung 22 Teilen-Funktion für eine Comicdatei

3.7.4 Kritische Betrachtung der Lösung

Im Rückblick besteht bei der lokalen Nutzung der Comicdateien noch weiteres Verbesserungspotenzial. Downloads können zurzeit nur durchgeführt werden, solange die App auf dem Gerät im Vordergrund ist. Auf Grund der mitunter langen Wartezeiten kann es jedoch sein, dass der Benutzer zwischenzeitlich die App verlässt. Zurzeit führt dies jedoch zum Abbruch des Downloads. Außerdem wird bei mehreren Downloads für jeden Download eine eigene Benachrichtigung erstellt und diese bieten keine Option zum Abbruch des jeweiligen Downloads. Daher wäre es sinnvoll, den Downloadvorgang in der Zukunft noch einmal zu überarbeiten. Eine bessere Lösung könnte zum Beispiel darin bestehen, neue Downloads in eine Warteschlange einzureihen und diese so im Hintergrund abzuarbeiten, dass das Verlassen der App nicht zum Abbruch der Downloads führt. Für die noch nicht abgeschlossenen Downloads könnte eine gemeinsame Benachrichtigung über den Gesamtfortschritt erstellt werden, welche auch ein Pausieren oder einen Abbruch der Downloads ermöglicht. Eine Übersicht über die eingereihten Downloads in der App könnte außerdem das Priorisieren und Abbrechen von noch nicht abgeschlossenen Downloads ermöglichen. Eine weitere Verbesserungsmöglichkeit könnte darin bestehen, weitere Dateiformate in Coboli zu unterstützen. Bezüglich der Teilen- und der „Öffnen mit“-Funktionen besteht kaum noch Raum zur Verbesserung, da die Möglichkeit zur Nutzung dieser Funktionen in erster Linie von anderen Apps abhängt. Insgesamt stellen die bezüglich der lokalen Nutzung der Comicdateien getroffenen Entscheidungen jedoch eine zufriedenstellende Lösung dar.

3.8 Offline-Modus

Im Offline-Modus wird der Button für das manuelle Starten der Synchronisierung ausgegraut und deaktiviert. Dadurch wird dem Benutzer signalisiert, dass sich die App im Offline-Modus befindet und er keine Synchronisierung starten kann. Außerdem wird im Offline-Modus die Download-Option für Comics ausgeblendet und das Wolken-Symbol, das darauf hinweist, dass ein Comic nicht auf dem lokalen Gerät gespeichert ist, durch einen diagonalen Balken negiert. Dadurch wird dem Benutzer signalisiert, dass zurzeit kein Download möglich ist.

Da abgesehen von Synchronisierung und Download alle Funktionen der App von der Erreichbarkeit des ComicLib-Servers unabhängig sind, hat der Offline-Modus keinen Einfluss auf diese. Die Comicsammlung kann auch offline durchstöbert werden und auch das Lesen bereits heruntergeladener Comics ist möglich (sofern das Dateiformat unterstützt wird). Auch die „Öffnen mit“- und „Teilen“-Funktionen bleiben vom Online-Status unberührt

Die (De-)Aktivierung des Offline-Modus erfolgt automatisch. Nach dem Start der App prüft Coboli im 10-Sekunden-Intervall, ob der ComicLib API-Endpunkt erreichbar ist.[184] Ist das nicht der Fall wechselt die App in den Offline-Modus.

Das Zeitintervall der Verfügbarkeitsprüfung wurde auf 10 Sekunden festgelegt, um die Last, die diese auf dem Server erzeugt, gering zu halten. Eine Prüfung im Sekundenintervall könnte auf einem ComicLib-Server mit vielen Benutzern und wenig Leistung, z.B. einem Raspberry Pi, zu Problemen führen, insbesondere wenn gleichzeitig Downloads oder der Lesemodus in ComicLib bedient werden müssen. Ein Intervall von 10 Sekunden hält die Last gering, garantiert jedoch gleichzeitig eine ausreichend geringe Wartezeit bei der Wiederherstellung der Verbindung nach einem Abriss derselben. Das Intervall wurde mit 10 Sekunden allerdings bewusst großzügig dimensioniert. Der Wert kann in der Zukunft noch reduziert werden, wenn genügend Wissen über die die Nutzungsweise von ComicLib und Coboli durch die Benutzercommunity gesammelt wurde.

Im Rückblick wurde durch den Offline-Modus das Nutzungserlebnis beim Fehlen einer Verbindung zum Server deutlich verbessert, denn dem Benutzer wird deutlich vermittelt, welche Funktionen nicht zur Verfügung stehen. Das Intervall von 10 Sekunden ist jedoch recht lang und sollte in der Zukunft evaluiert und gegebenenfalls reduziert werden, um Wartezeiten beim Benutzer zu vermeiden. Insgesamt führten die Entscheidungen bezüglich des Offline-Modus zu einer guten Lösung.

3.9 Sicherheitsaspekte

Da der Benutzer seine Anmeldedaten, bestehend aus Benutzername und Passwort, Coboli anvertraut, muss die App diese vor unbefugtem Zugriff schützen. Dazu ist es notwendig, die Anmeldedaten nur verschlüsselt zu speichern und an den ComicLib-Server zu übertragen. Neben der Verschlüsselung der sensiblen Anmeldedaten sind weitere Sicherheitsaspekte bei der Umsetzung des Projekts zu beachten. Bei der Benutzung von ComicLib fallen möglicherweise personenbezogene Daten an. Daher muss geklärt werden, in wie fern bei der Nutzung von Coboli in Kombination mit einer ComicLib-Instanz auf Aspekte des Datenschutzes geachtet werden muss. Diesen und weiteren Sicherheitsaspekten von Coboli widmen sich die folgenden Kapitel.

3.9.1 Verschlüsselung der Übertragung

Zum Schutz der Anmeldedaten vor Ausspähung durch einen Dritten beim Zugriff auf die ComicLib-API muss die Verbindung verschlüsselt werden. Coboli ermöglicht die Verschlüs-

[184] Vgl. Kapitel Verfügbarkeit der API

selung der Verbindung zum ComicLib-Server mittels HTTPS und TLS. Die dazu verwendeten Zertifikate können von einer öffentlichen Zertifizierungsstelle oder selbst signiert werden. Zudem kann die Verbindung auf Wunsch auch ohne Verschlüsselung erfolgen.[185]

Die Entscheidung, ob und wie Verschlüsselung verwendet werden kann, liegt beim Verwalter des ComicLib-Servers. Er entscheidet darüber, ob ComicLib über HTTP oder HTTPS erreichbar ist und welches Zertifikat für HTTPS genutzt wird. ComicLib ermöglicht im Werkszustand die Nutzung von HTTP sowie HTTPS. Für letzteres findet ein selbstsigniertes Zertifikat Verwendung.

Der Verwalter des Servers kann jedoch den HTTP- oder HTTPS-Port abschalten und so nur eine der beiden Verbindungsmethoden ermöglichen. Die Abschaltung des HTTP-Ports zwingt die Benutzer, HTTPS zu nutzen und somit die Verbindung zu verschlüsseln, wodurch die Sicherheit erhöht wird. Die Abschaltung des HTTPS-Ports hingegen senkt die Sicherheit massiv, da keine Verschlüsselung genutzt werden kann und es dadurch einem Angreifer enorm erleichtert wird, die Anmeldedaten sowie die möglicherweise personenbezogenen Nutzdaten abzugreifen.

Außerdem kann der Verwalter das vorinstallierte Zertifikat gegen ein anderes austauschen. Tauscht er das Zertifikat gegen eines einer vertrauenswürdigen, öffentlichen Zertifizierungsstelle ein, erhöht dies die Sicherheit der Verbindung weiter, da so eine Verifizierung der Echtheit des ComicLib-bereitstellenden Servers möglich wird. Bei Nutzung eines selbstsignierten Zertifikats hingegen ist eine automatische Verifizierung nicht möglich, wodurch ein potenzielles Sicherheitsrisiko entsteht. Ein Angreifer könnte in der Lage sein, mit einem gefälschten Server-Zertifikat eine Man-in-the-Middle-Attacke[186] durchzuführen, indem er seinen eigenen, öffentlichen Schlüssel als den des Servers ausgibt. Coboli fordert daher vor der Verwendung eines selbstsignierten Zertifikats eine Bestätigung durch den Benutzer an und speichert danach den Hostnamen und das Zertifikat. Mit diesen kann bei jedem neuen Verbindungsaufbau geprüft werden, ob sich das Zertifikat geändert hat. Ist dies der Fall, muss der Benutzer die Verwendung des neuen Zertifikats prüfen und bestätigen. Dadurch wird die Vertrauenswürdigkeit bei Verwendung eines selbstsignierten Zertifikats gewährleistet.[187]

Falls sowohl HTTP als auch HTTPS zur Verfügung stehen, obliegt es dem Coboli-Benutzer, zu entscheiden, welche Verbindungsart er wählt. Damit er nicht aus Unwissenheit die unsichere HTTP-Verbindungsart wählt, weist Coboli bei Verwendung einer Nicht-HTTPS-Adresse in der Anmeldesicht explizit darauf hin, dass die Verbindung nicht verschlüsselt

[185] Vgl. Kapitel *Verbindung zum Server*

[186] Vgl. (Schnabel, Man-in-the-Middle, 2019)

[187] Vgl. Kapitel *Verbindung zum Server*

sein wird und fordert vor der Nutzung einer solchen Adresse eine Bestätigung an. Da es in Ausnahmefällen erwünscht sein kann, auf HTTPS zu verzichten – beispielsweise zur Analyse der Anfragen an ComicLib - ist diese Möglichkeit in Coboli enthalten. Jedoch kann grundsätzlich nur davon abgeraten werden, diese zu nutzen. Falls der Benutzer die Sicherheit seiner Anmeldedaten riskieren möchte, steht es ihm frei, dies zu tun. Coboli rät jedoch ausdrücklich durch Nachrichten und Dialoge davon ab.

3.9.2 Verschlüsselte Speicherung der Anmeldedaten

Die Anmeldedaten werden mithilfe von Shared Preferences gespeichert und geladen. Shared Preferences erzeugt zum Speichern eine Datei im privaten Datenverzeichnis von Coboli. Coboli nutzt den Dateiberechtigungsmodus „MODE_PRIVATE“[188], damit die Datei nur von Coboli und sonst von keiner anderen App gelesen werden kann. Da das Systemkonto Root jedoch grundsätzlich alles auf dem Gerät lesen kann, hat dieses auch Zugriff auf die Datei mit den Anmeldedaten. Zur Beschaffung der Anmeldedaten müsste sich ein Angreifer also nur Root-Rechte auf dem Gerät verschaffen. Damit auch in diesem Falle die Anmeldedaten noch sicher sind, ist es notwendig, diese gut zu verschlüsseln.

Zur Verschlüsselung der Anmeldedaten wird die *Cipher*-Klasse des Android-Frameworks genutzt. Diese ermöglicht das Ver- und Entschlüsseln von Datenströmen. Als Verschlüsselungsalgorithmus kommt dabei „AES/GCM/NoPadding“ zum Einsatz, da diese Kombination im offiziellen Leitfaden zu Kryptographie empfohlen wird.[189] Zur Erzeugung und Speicherung des kryptographischen Schlüssels wird der Android Keystore verwendet, welcher die Erzeugung zufälliger Schlüssel und deren sichere Speicherung garantiert.[190]

Der verwendete Verschlüsselungsalgorithmus benötigt einen Zufallszahlengenerator zum Ver- und Entschlüsseln. Dieser wird genutzt, um sicherzustellen, dass zwei identische (Teil-)Klartexte nicht zum selben Ciphertext führen, da sonst anhand dessen ohne Kenntnis des Schlüssels Rückschlüsse auf den Inhalt des Klartexts möglich wären. Dem Algorithmus muss daher ein zufälliger Initialisierungsvektor (IV) für den Zufallszahlengenerator übergeben werden. Da dieser Vektor auch zur Entschlüsselung des Ciphertexts benötigt wird und selbst nicht kryptographisch sensibel ist, wird er vor der Speicherung dem Ciphertext vorangestellt. Dadurch ist sichergestellt, dass für jeden Verschlüsselungsvorgang ein anderer Initialisierungsvektor Verwendung findet und für jeden Ciphertext der korrekte Vektor zur Entschlüsselung vorhanden ist.[191]

[188] Vgl. (Google, Inc., 2019)

[189] Vgl. (Google, Inc., 2019)

[190] Vgl. (Google, Inc., 2019)

[191] Vgl. (Google, Inc., 2019)

3.9.3 Datenschutz

Mit der Datenschutz-Grundverordnung der EU, die zum 25. Mai 2018 in Kraft getreten ist, wurde das Recht auf Datenschutz weiter gestärkt.[192] Da Coboli Daten erhebt, die mit einem ComicLib-Server geteilt werden, müssen grundlegende Überlegungen bezüglich dieses Themas angestellt werden. Welche Daten werden ausgetauscht? Welche sind personenbezogen? Wer hat auf die erhobenen Daten Zugriff? Im Folgenden sollen diese Fragen geklärt werden.

Da Coboli zum Zugriff auf die Comicsammlung aus ComicLib Anmeldedaten verwenden muss, besteht die Möglichkeit, den Datenfluss zwischen Coboli und ComicLib auf den Benutzer zu beziehen. Da für die Anmeldedaten ein Benutzername verwendet wird, erfolgt der Zugriff auf ComicLib unter einem Pseudonym. Der Zusammenhang zwischen Pseudonym und Person ist dabei mindestens demjenigen ComicLib-Administrator, der das Benutzerkonto angelegt hat, bekannt. Sofern dieser jedoch keinen administrativen Zugriff auf den Server, auf dem ComicLib betrieben wird, hat, kann dieser die Aktionen des Benutzers nicht einsehen. ComicLib stellt selbst keine Möglichkeit zur Einsichtnahme in die Aktivitäten anderer Benutzer bereit. Jeder Benutzer hat Zugriff auf die komplette Comicsammlung und seine eigenen Lesestati. Lediglich der Server-Administrator kann sich Zugang zu den Aktivitäten der Benutzer verschaffen, indem er die Log-Dateien des Webservers nach Anfragen bestimmter IP-Adressen durchsucht oder die Lesestati eines Benutzers manuell aus der Datenbank ausliest.

Coboli tauscht mit ComicLib zwei Kategorien von Daten aus. Zum einen wird bei der Synchronisierung die Comicsammlung aus ComicLib auf das lokale Gerät gespiegelt. Dieser Vorgang gibt nur Aufschluss darüber, dass Coboli genutzt wird und wann dies geschieht. Die hierbei ausgetauschten Daten sind für alle Benutzer gleich, daher sind diese von eher geringem Interesse. Die zweite Kategorie sind Daten, aus denen auf die Vorlieben eines Benutzers geschlossen werden kann. Hierzu gehören die Lesestati des Benutzers sowie die Liste der durch den Benutzer heruntergeladenen Comics. Anhand dieser Informationen wäre es beispielsweise möglich, personalisierte Werbung für den Benutzer zu erstellen.

Neben diesen Daten speichert ComicLib natürlich auch die Anmeldedaten. Das Passwort wird durch ComicLib allerdings gehasht gespeichert. Dabei wird es durch eine Einweg-Funktion in eine neue Zeichenkette umgewandelt, aus der das Passwort nicht mehr rekonstruiert werden kann. Für einen Angreifer sind die Anmeldedaten in der Datenbank damit unbrauchbar. Da der Server-Administrator jedoch vollen Zugriff auf die ComicLib-Installation hat, wäre es diesem möglich, die Anmeldedaten im Klartext zu erlangen, indem er Änderungen am Quelltext von ComicLib vornimmt, um diese beim Anmelden abzufangen.

[192] Vgl. (dejure.org Rechtsinformationssysteme GmbH, 2018)

Da ComicLib und Coboli vor allem zur Nutzung als persönliche, selbst gehostete Comicsammlung gedacht sind, spielen datenschutzrechtliche Bedenken eine eher untergeordnete Rolle. Bei dieser Art der Nutzung ist der Coboli-Benutzer auch der Server-Administrator, wodurch keine personenbezogenen Daten an weitere Personen gelangen. Bei der Bereitstellung von ComicLib für andere Benutzer ist es jedoch sehr zu empfehlen, datenschutzrechtliche Beratung einzuholen und die entsprechenden Erklärungen und Verträge anfertigen zu lassen.

3.9.4 Sicherheit der /authenticated API-Ressource

Im Zuge des Projekts wurde die */authenticated*-Ressource der ComicLib-API hinzugefügt.[193] Diese ermöglicht es, API-Schlüssel einfach auf ihre Gültigkeit hin zu überprüfen. Aus der Perspektive der IT-Sicherheit wirkt es nicht wie eine gute Idee, eine API-Ressource zu schaffen, die einem potenziellen Angreifer auf Nachfrage zu jedem vorgezeigten API-Schlüssel verrät, ob dieser gültig ist. Ein solches Verhalten lädt geradezu dazu ein, zu versuchen, ein Passwort mit roher Gewalt (Brute-Force-Angriff[194]) zu knacken. Schließlich muss man nur solange zufällige Zeichenketten vorzeigen, bis man durch Zufall eine richtige erwischt. Mit genügend Rechenleistung ist der enorme Zeitaufwand für Brute Force-Verfahren kompensierbar und in Zeiten minutenweise mietbarer Computerleistung nur noch eine Frage der Kosten.[195]

ComicLib ist jedoch gegen Brute Force-Attacken auf API-Schlüssel gut gewappnet. Die verwendeten, zufällig generierten Schlüssel bestehen aus 128-stelligen Hexadezimalzahlen. Diese müssen als Zeichenketten gespeichert werde, da SQL über keinen Ganzzahl-Datentypen verfügt, der eine derart große Zahl aufnehmen kann. Hinzu kommt, dass die Anzahl der möglichen Anfragen pro Sekunde beim Zugriff über die API sehr stark begrenzt ist. Die Anzahl möglicher Anfragen pro Sekunde hängt von der Leistung des Rechners, auf dem ComicLib läuft, ab und kann durch den Angreifer auch nicht erhöht werden. Im Falle des Raspberry Pi der Testinstallation dieses Projekts ergaben Tests unter Zuhilfenahme von Apache JMeter, dass maximal 10 Anfragen pro Sekunde möglich sind.[196] Bei einem API-Schlüssel von 128 Stellen mit 16 möglichen Zeichen pro Stelle ergeben sich 16^128 unterschiedliche, mögliche Schlüssel. Bei 10 Anfragen pro Sekunde alle möglichen Schlüssel durchzuprobieren würde $\frac{16^{128}}{10\frac{1}{s} * 60\frac{s}{min} * 60\frac{min}{h} * 24\frac{h}{d} * 365\frac{d}{a}} \approx 4{,}25 * 10^{145}$

[193] Vgl. Kapitel Änderungen an der API

[194] Vgl. (Vogel IT-Medien GmbH, 2018)

[195] Vgl. (Heise Medien, 2011)

[196] Vgl. Tabelle 11 Maximale Anzahl paralleler Zugriffe auf /authenticated auf einem Raspberry Pi 3 Model B.

Jahre dauern. Daher ist es vollkommen unrealistisch, mittels der */authenticated*-Ressource einen Schlüssel erraten zu können. Die eventuell vorhandene Verschlüsselung der Verbindung zu knacken und dort den Schlüssel oder womöglich direkt Benutzername und Passwort mittels einer Man-in-the-Middle-Attacke[197] auslesen zu können, stellt die weitaus realistischere Bedrohung dar.

3.9.5 Kritische Betrachtung der Lösung

Im Rückblick besteht bezüglich der Sicherheit von Coboli noch weiteres Verbesserungspotenzial. Die umgesetzten Sicherheitsmaßnahmen ermöglichen eine hohe Sicherheit bei der Benutzung von Coboli. Durch die Verwendung eines vertrauenswürdigen Zertifikats für die HTTPS-Verbindung kommt eine starke Verschlüsselung bei der Verbindung zu ComicLib zum Einsatz. Durch den Einsatz von Verschlüsselung beim Speichern der Anmeldedaten sind auch diese vor unerlaubtem Zugriff geschützt. Außerdem sind die durch ComicLib erzeugten API-Schlüssel sehr komplex und daher schwer zu knacken. Auf Grund der Möglichkeit, die Verbindung zum ComicLib-Server mit unverschlüsseltem HTTP zu betreiben, lässt sich eine der Sicherheitsmaßnahmen jedoch komplett aushebeln. Es könnte daher sinnvoll sein, diese Option in der Zukunft aus Coboli zu entfernen. Insgesamt stellen die in Bezug auf die Sicherheit von Coboli getroffenen Entscheidungen jedoch eine gute Lösung dar.

3.10 Lokalisierung

Die folgenden Kapitel widmen sich den Anpassungen in Coboli für die Nutzung in unterschiedlichen Regionen und mit unterschiedlichen Sprachen. Dabei liegt das Hauptaugenmerk auf der Übersetzung von Coboli in andere Sprachen. Es werden jedoch auch weitere Möglichkeiten zu regionalen Anpassungen erörtert.

3.10.1 Sprachen in Coboli

Coboli wird, ebenso wie ComicLib, auf Englisch entwickelt. Klassen-, Variablen- und alle sonstigen Namen sind auf Englisch verfasst. Auch die Kommentare im Quelltext sowie die Commit-Nachrichten in Git sind in englischer Sprache. Das soll es Entwicklern überall auf der Welt ermöglichen, die Software selbst verstehen und ändern zu können.

Da viele Menschen kein Englisch sprechen ist es sinnvoll, die App so zu gestalten, dass ihre Inhalte leicht in andere Sprachen übersetzt werden können. Da viele Inhalte in Coboli, wie auch in ComicLib, aus der ComicVine-Datenbank stammen, ist es nicht möglich, diese

[197] Vgl. (Schnabel, Man-in-the-Middle, 2019)

Inhalte zu übersetzen. ComicVine enthält in seiner Datenbank auch viele nicht-englischsprachige Comics und Verlage, jedoch sind die Beschreibungstexte immer auf Englisch. Die Texte, die die App selbst bereitstellt, wie beispielsweise Button-Beschriftungen, können jedoch übersetzt werden.

Android nutzt zur Umsetzung von Mehrsprachigkeit ein String-Ressourcensystem. Dieses prüft, welche Lokalisierungseinstellungen auf dem Gerät vorliegen und lädt alle benötigten Zeichenketten aus der entsprechenden, übersetzten Variante der Zeichenketten-Datei, sofern sie existiert. Falls keine Übersetzung der Zeichenketten für die Systemsprache des Geräts existiert, werden statt dessen die Zeichenketten aus der Standardübersetzung geladen.[198]

Bei der Entwicklung einer mehrsprachigen App muss der Entwickler darauf achten, die Zeichenketten, die dem Benutzer angezeigt werden sollen, nicht direkt im Quelltext festzulegen. Statt dessen muss in der Datei mit den Zeichenketten (Strings-Datei) der Standardübersetzung eine neue String-ID vergeben werden, der dann der anzuzeigende Text zugewiesen wird. Im Quelltext wird an Stelle des Strings nur die String-ID angegeben. Anhand dieser lädt Android dann die Zeichenkette der passenden Übersetzung.[199]

Die Standardsprache der App legt der Entwickler fest, indem er sie in der Standard-Strings-Datei verwendet. Für Coboli wurde als Standardsprache Englisch verwendet, damit alle Texte in der App automatisch auf Englisch erscheinen, falls keine zur Systemsprache passende Übersetzung vorhanden ist.

Für jede weitere Sprache muss der Entwickler eine eigene Strings-Datei erstellen und dort für jede String-ID eine Übersetzung festlegen. Manche Zeichenketten können nicht übersetzt werden. Diese müssen in der Standard-Übersetzung als unübersetzbar gekennzeichnet werden.[200]

Neben der Standard-Sprache Englisch verfügt Coboli über eine vollständige, deutschsprachige Übersetzung. Übersetzungen in weitere Sprachen können nach der Veröffentlichung von Coboli durch Dritte beigetragen werden. Für eine hohe Qualität der Übersetzungen ist es allerdings unabdingbar, dass die Übersetzer über gute Sprachkenntnisse in einer der vorhandenen Sprachen der App sowie der Sprache der neuen Übersetzung verfügen. Da meine Kenntnisse in meiner zweiten Fremdsprache, Französisch, eher mäßig sind, habe ich diesbezüglich von einer Übersetzung abgesehen.

[198] Vgl. (Google, Inc., 2019)

[199] Vgl. (Google, Inc., 2019)

[200] Vgl. (Google, Inc., 2019)

Coboli verwendet zum Anzeigen einiger Icons die Font Awesome Bibliothek. Die Symbole werden dabei als zusätzliche Schriftart mit eigenen Unicode-Zeichen eingebunden. Daher stehen die verwendeten Unicode-Zeichen aus Font Awesome ebenfalls in den String-Dateien. Sie sind allerdings als unübersetzbar markiert und können daher beim Übersetzen in eine neue Sprache ignoriert werden.

Android Studio unterstützt den Entwickler dabei, die App gut lokalisierbar zu gestalten. Beispielsweise zeigt es eine Warnung an, wenn in einer Layout-Datei statt einer String-ID eine Zeichenkette eingegeben wird. Das soll verhindern, dass Teile der App nicht übersetzt werden können.

Außerdem enthält Android Studio einen Editor für Übersetzungen, der alle Übersetzungen der App in einer gemeinsamen, tabellarischen Ansicht vereint. In dieser können Zeichenketten in allen Übersetzungen geändert werden, ohne jede String-Datei einzeln öffnen zu müssen. Zudem werden in der Tabelle Zeichenketten mit fehlender Übersetzung hervorgehoben. Gerade für Nicht-Programmierer erleichtert der Übersetzungseditor das Arbeit mit den Übersetzungen enorm, da er kein Vorwissen über XML-Dateien oder die Verzeichnisstruktur des Projekts voraussetzt.[201]

3.10.2 Weitere Möglichkeiten der Lokalisierung

Coboli beschränkt sich bezüglich der Lokalisierung darauf, die Texte der App zweisprachig zur Verfügung zu stellen. Für die Inhalte, die über die ComicLib-API bezogen werden, kann jedoch keine Übersetzung bereitgestellt werden.

Neben der Übersetzung in zusätzliche Sprachen sind weitere Anpassungen an regionale Gegebenheiten denkbar. Zum Beispiel wäre es möglich, für Regionen mit Sprachen, die nicht von links nach rechts geschrieben und gelesen werden, alternative Benutzeroberflächen anzubieten, die besser an die jeweilige Schreibrichtung angepasst sind. Für diese Regionen wäre es möglicherweise auch sinnvoll, die Blätter-Richtung der Leseansicht anzupassen. Japanische Comics (Mangas) werden beispielsweise – aus deutscher Sicht – von hinten nach vorne gelesen. Android unterstützt neben lokalisierten Texten auch lokalisierte Layouts mit Hilfe eines Ressourcensystems.[202]

3.10.3 Kritische Betrachtung der Lösung

Im Rückblick besteht bezüglich der Lokalisierung von Coboli noch weiteres Verbesserungspotenzial. Durch die konsequente Nutzung der Lokalisierungsfunktionen von Android konnte Coboli gut an ein deutschsprachiges sowie internationales Publikum angepasst werden. Die

[201] Vgl. (Google, Inc., 2019)

[202] Vgl. (Google, Inc., 2019)

App verfügt über eine vollständige deutsch sowie englische Übersetzung. Weitere Übersetzungen können bei guter Kenntnis einer der beiden sowie einer dritten Sprache leicht erstellt werden. Zurzeit ist es noch nicht nötig, weitere Layouts für Regionen mit anderer Schreibrichtung bereitzustellen. Allerdings erfreuen sich auch im deutschsprachigen Raum japanische Comics (Mangas) großer Beliebtheit. Da diese jedoch von rechts nach links und von hinten nach vorne gelesen werden, funktioniert die Lesesicht bei diesen nicht korrekt. Beispielsweise kann das Heft nicht automatisch bei Erreichen der letzten Seite als gelesen markiert werden, da die letzte Seite in der Lesesicht eines Mangas das Cover statt der tatsächlich letzten Seite darstellt. Daher wäre es sinnvoll, in der Zukunft eine Option zum Umkehren der Leserichtung in der Lesesicht anzubieten. Insgesamt fällt die Lösung jedoch zufriedenstellend aus.

3.11 Implementierte Activities

Android verwendet zur Darstellung der Benutzeroberflächen sogenannte Activities.[203] Diese dienen als Hauptsichten in der App und können weitere Teilsichten – Fragments[204] - enthalten. In Coboli wurden fünf Activities implementiert, die zum Teil auf Fragmente zurückgreifen, um Inhalte darzustellen. Die folgenden Kapitel stellen die Activities und ihre Funktionen vor. Screenshots der hier erläuterten Sichten finden sich im Anhang.

3.11.1 Main-Activity

Die Main-Activity dient nur als Programm-Einstiegspunkt. Sie erbt die Funktionalität der Toolbar-Activity, welche den größten Teil der App-Funktionalität bereitstellt, und erweitert diese lediglich um den automatischen Start der Sync-Activity beim App-Start. Dadurch wird, sofern Anmeldedaten hinterlegt sind und eine Verbindung zum ComicLib-Server möglich ist, beim App-Start eine Synchronisierung mit dem Server durchgeführt, um die App-Inhalte auf den neuesten Stand zu bringen (vgl. *Abbildung 8*).

3.11.2 Login-Activity

Die Login-Activity wird beim ersten Start der App angezeigt und dient der Eingabe der Verbindungsdaten für den Zugriff auf den ComicLib-Server (vgl. *Abbildung 6*). Die Eingabefelder für Benutzername und Passwort verfügen dabei über Unterstützung für Androids Autofill-Dienst.[205] Dieser steht ab Android-Version 8 zur Verfügung und ermöglicht das automatische Eintragen von Anmeldedaten in Anmeldefenster. Beim Start von Coboli auf einem

[203] Vgl. (Google, Inc., 2019)

[204] Vgl. (Google, Inc., 2019)

[205] Vgl. (Google, Inc., 2019)

Gerät mit Android 8 oder neuer können daher zuvor gespeicherte Anmeldedaten zum automatischen Ausfüllen der entsprechenden Felder ausgewählt werden. Beim ersten erfolgreichen Login in Coboli mit einer neuen Kombination aus Benutzername und Passwort bietet die App an, diese mithilfe von Autofill für zukünftige Anmeldungen zu speichern. Wie sicher die Anmeldedaten gespeichert werden, hängt dabei davon ab, welche App im Hintergrund den Anmeldedatenspeicher für Autofill bereitstellt, da diese austauschbar ist. Standardmäßig wird Google Autofill genutzt.

Nach einem Tippen auf „Login“ versucht Coboli, eine Verbindung zu einer ComicLib-Instanz unter Verwendung der eingegebenen Anmeldedaten herzustellen. Gelingt dies, schließt sich die Activity und die Main-Activity löst die Synchronisierung mit dem Server aus, indem sie die Sync-Activity startet. Andernfalls wird eine Fehlermeldung angezeigt und die Activity bleibt bestehen. Falls ein Zertifikatfehler auftritt, wird ein Dialog angezeigt, der dem Benutzer das Überprüfen der Zertifikatdetails und das manuelle Akzeptieren des Zertifikats ermöglicht (vgl. *Abbildung 7*).

3.11.3 Sync-Activity

Die Sync-Activity startet die Synchronisierung mit dem ComicLib-Server. Sie zeigt eine Wartemeldung an, solange die Synchronisierung noch nicht abgeschlossen wurde (vgl. *Abbildung 8*). Falls keine Verbindung zum Server aufgebaut werden konnte oder die Synchronisierung abgeschlossen wurde, wird die Activity geschlossen und statt dessen die Toolbar-Activity (als Teil der Main-Activity) angezeigt. Das Thema *Synchronisierung mit dem Server* wird im gleichnamigen Kapitel behandelt.

3.11.4 Toolbar-Activity

Die Toolbar-Activity stellt den größten Teil der Funktionalität der App bereit. Sie verwendet Fragments, um die Bestandteile der Sammlung sowie die „Über diese App“-Seite darzustellen. Die Activity enthält eine Action-Bar[206], die eine Suchfunktion für Serien (vgl. *Abbildung 15*), einen Button zum Starten der Sync-Activity sowie ein ausklappbares Navigationsmenü bereitstellt.

Das Navigationsmenü klappt als Schublade („Navigation Drawer“)[207] von der linken Seite des Bildschirms herein und ermöglicht die Navigation zu anderen Bestandteilen der Sammlung. Durch das Navigationsmenü kann zu den Fragmenten für die Listen der Verlage, der Serien sowie der heruntergeladenen und angelesenen Hefte gelangt werden. Außerdem

[206] Vgl. (Google, Inc., 2019)

[207] Vgl. (Google, Inc., 2019)

ermöglicht es den Zugriff auf die „Über diese App“-Seite und das Abmelden aus Coboli (vgl. *Abbildung 12*).

Die Bestandteile der Sammlung werden durch das Publishers-, Volumes- und Issues-Fragment dargestellt. Diese Fragmente dienen der Darstellung jeweils eines Elementtyps der Sammlung. Das Publishers-Fragment stellt beispielsweise die Liste der Verlage dar (vgl. *Abbildung 16*). Die Fragmente werden an unterschiedlichen Stellen wiederverwendet. Dabei wird die im Fragment dargestellte Liste dem jeweiligen Kontext angepasst, die Darstellung und das Verhalten bleiben jedoch gleich. Beispielsweise zeigt das Volumes-Fragment im Bereich „Serien“ des Navigationsmenüs alle Serien der Sammlung an (vgl. *Abbildung 9*). Tippt man jedoch im Bereich „Verlage“ des Navigationsmenüs auf eine Verlagskarte, wird das Volumes-Fragment zur Darstellung der Liste der Serien des Verlags genutzt. Die Fragmente teilen sich ein gemeinsames Layout zur Darstellung der Bestandteile der Sammlung. Für dieses und einige der anderen Layouts existieren Variationen zur Optimierung der Darstellung bei Verwendung der App im Querformat oder auf Tablets (vgl. *Abbildung 29*).

Die Sammlung gliedert sich in zwei Bereiche. Der Bereich „Serien“ enthält das Volumes-Fragment als Ausgangspunkt. Die Serien der Sammlung werden als Karten in einer Albumansicht dargestellt (vgl. *Abbildung 9*). Mithilfe eines Buttons kann für jede Serie ein Popup-Menü ausgeklappt werden (vgl. *Abbildung 10*). Das Menü bietet jeweils die Optionen, die Hefte der Serie als (un-)gelesen zu markieren oder die Details zur Serie inkl. eines Beschreibungstexts als Overlay einzublenden (vgl. *Abbildung 11*). Nach dem Tippen auf eine der Karten wird die Liste der Hefte der jeweiligen Serie mithilfe des Issues-Fragments dargestellt (vgl. *Abbildung 13*).

Der Bereich „Verlage“ enthält das Publishers-Fragment als Ausgangspunkt. Dieses zeigt die Verlage der Sammlung als Karten in einem Album an (vgl. *Abbildung 16*). Das Popup-Menü der Verlage enthält nur die Option, die Details zum Verlag als Overlay anzuzeigen (vgl. *Abbildung 17*). Ein Tippen auf die Karte eines Verlags öffnet das Volumes-Fragment zur Darstellung der Liste der Serien des Verlags. Dieses verhält sich so wie im Bereich „Serien“.

Das Issues-Fragment dient der Darstellung der Hefte einer Serie als Karten in einem Album (vgl. *Abbildung 13*). Das Popup-Menü bietet die Optionen, die Details zum Heft als Overlay anzuzeigen (vgl. *Abbildung 14*), den Lesestatus zu ändern oder die Comicdatei des Hefts herunterzuladen. Beim Download des Hefts wird eine Benachrichtigung angezeigt, die über den Fortschritt des Vorgangs informiert (vgl. *Abbildung 18* und *Abbildung 19*). Die heruntergeladenen Hefte werden serienweise mithilfe des Volumes-Fragments im Bereich „Heruntergeladen“ angezeigt (vgl. *Abbildung 20*). Falls die Comicdatei eines Hefts bereits im Comiccache liegt, wird die Herunterladen-Option im Popupmenü durch eine Löschen-Option ersetzt. Außerdem werden die Optionen zur lokalen Verwendung der Comicdatei eingeblendet (vgl. *Abbildung 21*). Falls das Dateiformat der Comicdatei durch Coboli unterstützt

wird, wird die Öffnen-Option angezeigt, die die Comicdatei in der Reader-Activity öffnet. Unabhängig vom Dateiformat werden außerdem die Teilen- und die „Öffnen mit“-Funktion angezeigt. Mit Hilfe der Teilen-Funktion kann die Datei durch eine andere App geteilt oder im Gerätespeicher außerhalb der App gespeichert werden (vgl. *Abbildung 22*).[208] Die „Öffnen mit“-Option zeigt eine Liste der Apps an, die den Dateityp der Datei öffnen können. Falls nur eine kompatible App vorhanden ist, wird diese direkt geöffnet. Falls keine passende App installiert ist, wird eine Fehlermeldung angezeigt, die über das Fehlen einer passenden App informiert.

Im Bereich „Leseliste“ werden die angelesenen Hefte der Sammlung angezeigt (vgl. *Abbildung 26*). Hierzu wird das Issues-Fragment genutzt. Die Leseliste zeigt sowohl heruntergeladene als auch nicht heruntergeladene Hefte an. Die nicht heruntergeladenen Hefte werden durch ein Cloud-Symbol gekennzeichnet. Die Hefte in der Leseliste werden anhand ihres Seriennamens sortiert und gruppiert.

Der „Über“-Bereich dient der Vorstellung der App und der verwendeten Bibliotheken (vgl. *Abbildung 27*). Das dazu verwendete About-Fragment zeigt im oberen Bereich die Copyright-Informationen zu Coboli sowie jeweils einen Button zum Anzeigen des Lizenztexts sowie zum Öffnen der Coboli-Projektwebseite im Browser. Darunter wird die Liste der verwendeten Bibliotheken angezeigt. Auch hier können der Lizenztext und die Projektwebseite angezeigt werden. Für die Overlays mit dem jeweiligen Lizenztext kommt das License-Fragment zum Einsatz (vgl. *Abbildung 28*).

3.11.5 Reader-Activity

Die Reader-Activity stellt die Lesesicht für Coboli zur Verfügung. Beim Öffnen einer Comicdatei mit dem Lesemodus wird zunächst eine Wartemeldung angezeigt, während die Comicdatei entpackt wird (vgl. *Abbildung 23*). Ist die Comicdatei entpackt, wird die im Lesestatus des Hefts enthaltene, aktuelle Seitennummer geladen. Dadurch kann der Comic auf der zuletzt gelesenen Seite fortgesetzt werden. Von dort aus kann durch horizontales Wischen über den Bildschirm durch den Comic geblättert werden. Beim Umblättern auf eine neue Seite wird deren Seitennummer automatisch im Lesestatus gespeichert (automatisches Lesezeichen). Ein Tippen auf die aktuelle Seite dunkelt diese ab und blendet einen Button zum Anzeigen des Popup-Menüs als Overlay ein (vgl. *Abbildung 24*). Dieses enthält Optionen zum Zurückspringen zum Cover sowie zum Anzeigen einer Seitenübersicht (vgl. *Abbildung 25*). Die Seiten in der Seitenübersicht werden durch Coboli verkleinert geladen, um die Performance beim Scrollen durch die Übersicht zu verbessern. In der Übersicht wird die aktuelle Seite blau markiert. Beim Tippen auf eine der Seiten springt die Lesesicht zu

[208] Vgl. (Google, Inc., 2019)

dieser und die Seitennummer im Lesestatus wird aktualisiert. Dadurch kann der Benutzer eine zuvor gelesene Stelle schnell wiederfinden, wenn er diese noch einmal nachlesen möchte.

3.11.6 Kritische Betrachtung der Lösung

Im Rückblick konnten alle in den Projektzielen definierten Sichten umgesetzt werden. Da es sich nur um einen ersten Prototypen handelt, besteht bei allen Sichten noch Verbesserungspotenzial, das in Zusammenarbeit mit den Benutzern ausgeschöpft werden kann. Besonders großes Verbesserungspotenzial besteht in der Anpassung der Lesesicht an alternative Leserichtungen. Außerdem findet bei Tablets häufig das 4:3-Seitenverhältnis Verwendung. Für dieses konnte Coboli noch nicht angepasst werden, da kein entsprechendes Testgerät zur Verfügung stand. Hier besteht also auch noch Verbesserungspotenzial. Insgesamt stellen die umgesetzten Sichten jedoch ein zufriedenstellendes Ergebnis dar.

4 Zusammenfassung

Die Durchführung des Softwareprojekts ist abgeschlossen. Insgesamt konnte das Projekt wie geplant realisiert werden.[209] Die folgenden Kapitel widmen sich der Zielerreichung der Projektziele und der Zukunft der Software Coboli.

4.1 Zielerreichung des Projekts

Zu Beginn des Projekts wurden auf Basis der Anforderungen an die Software und möglichen Lösungsansätzen die Teilziele des Softwareprojekts bestimmt.[210] Um zu ermitteln, wie erfolgreich das Projekt durchgeführt wurde, muss der Funktionsumfang der entstandenen Software mit den dreizehn Projektteilzielen verglichen werden.

Coboli stellt die Comic-Sammlung eines ComicLib-Servers in einer mobilen Android-App bereit (1). Dazu synchronisiert die App die Elemente der Comic-Sammlung und den Lesefortschritt des Benutzers mit dem Server (2 & 5). Dabei kann Coboli sowohl eine unverschlüsselte als auch eine verschlüsselte Verbindung zum Server aufbauen (4). Zur Authentifizierung gegenüber ComicLib nutzt Coboli den Benutzernamen und das Passwort des Benutzers sowie dessen API-Schlüssel, welche die App verschlüsselt speichert (3). Coboli ermöglicht den Download aller Comicdateien der Sammlung zur mobilen und netzwerkunabhängigen Nutzung (8). Dabei kann die App Comics in den Dateiformaten CBR, CBZ und PDF entpacken und in einer Lesesicht anzeigen, die beim Durchblättern eines Comics automatisch ein Lesezeichen für die aktuelle Seite setzt (10). Damit der Benutzer den Überblick über seine heruntergeladenen und angelesenen Comics behält, führt Coboli diese jeweils in einer Liste zusammen (9 & 12). Da nicht alle Comicdateien mit der Lesesicht genutzt werden können, ermöglicht die App das Ändern des Lesestatus von Heften und Serien mithilfe von Popup-Menüs (6). Um dem Benutzer zusätzliche Informationen zu den Verlagen, Serien und Heften der Comicsammlung geben zu können, kann durch Popup-Menüs und Overlays ein Beschreibungstext angezeigt werden (7). Damit die Comicdateien auch mit anderen Apps genutzt werden können, stellt Coboli Optionen zum Teilen und Öffnen der Dateien mit weiteren Apps zur Verfügung (11). Durch die Bereitstellung des „Über“-Bereichs im Navigationsmenü kann sich der Benutzer über Coboli und die in der App verwendete Bibliotheken informieren (13).

[209] Vgl. Kapitel Zielerreichung des Projekts

[210] Vgl. Kapitel Ziele des Projekts

Insgesamt wurden alle dreizehn Teilziele des Projekts umgesetzt. Das Projekt war also erfolgreich. Coboli stellt in Kombination mit ComicLib eine freie Alternative zu den Plattformen der großen Anbieter digitaler Comics zur Verfügung.

4.2 Veröffentlichung und Zukunft von Coboli

Coboli wird als Open Source Software auf GitHub veröffentlicht.[211] Ab Version 1.0 wird mit jedem Versionsupdate die jeweils aktuelle Version der App als installierbare Android-App dort zur Verfügung gestellt. Die Veröffentlichung der App im Google Play Store ist hingegen unwahrscheinlich. Google erhebt für neue Entwicklerkonten im Play Store eine einmalige Registrierungsgebühr in Höhe von 25$.[212] Da Coboli jedoch ohne Gewinnabsicht entwickelt wird, kann sich dieser Betrag nicht amortisieren. Daher sollte statt dessen nach alternativen Vertriebswegen gesucht. Da Coboli freie Software ist, bietet sich der Vertrieb über F-Droid an. F-Droid ist ein App-Store für freie und Open Source Software, der durch Spenden finanziert wird und daher keine Gebühren für das Einstellen von Apps erhebt.[213] Da F-Droid in der Freie-Software-Gemeinschaft viel genutzt wird, bietet es für Coboli ein geradezu ideales Publikum: Menschen, die sich für Freie Software und Alternativen zu den Plattformen großer Internetkonzerne interessieren. Daher wird Coboli ab Version 1.0 wahrscheinlich auch über F-Droid vertrieben.

Die Weiterentwicklung von Coboli endet nicht mit dem Abschluss dieses Projekts. Die im Zuge des Projekts entwickelte App ist lediglich ein erster Prototyp, der im Nachgang des Projekts noch weitere Tests durchlaufen muss, bevor die erste stabile Version veröffentlicht werden kann. In der Zukunft kann Coboli um weitere Funktionen erweitert werden. Beispielsweise gibt es neben den drei zurzeit unterstützten Comic-Dateiformaten CBR, CBZ und PDF noch weitere, freie Dateiformate, die häufig verwendet werden. Es könnte daher sinnvoll sein, Unterstützung für diese Dateiformate in Coboli und ComicLib zu implementieren, falls sich eine Nachfrage dafür ergibt. Eine weitere Möglichkeit zur Verbesserung der App besteht darin, die Downloadfunktion so umzugestalten, dass Downloads beim Verlassen der App nicht abbrechen. Dies sind nur zwei Beispiele für zukünftige Verbesserungsmöglichkeiten.

Da Coboli und ComicLib eine gemeinsame Plattform bilden, wäre es sinnvoll, für beide auch einen gemeinsamen Namen zu verwenden. Es böte sich an, ab der ersten stabilen Version (1.0) beide Softwareprojekte unter dem Namen Coboli weiterzuentwickeln. Aus ComicLib

[211] Vgl. https://github.com/ahahn94/Coboli

[212] Vgl. https://play.google.com/apps/publish/signup/, am Ende des Registrierungsprozesses

[213] Vgl. https://f-droid.org/de/

würde dann Coboli Server. Die Entscheidung, ob beide Projekte denselben Namen verwenden sollen, wird jedoch erst in der Zukunft getroffen werden.

Coboli und ComicLib schaffen eine freie Alternative zu den Comic-Plattformen der großen Anbieter. Jedoch ist diese auf die Benutzung im Internet-Browser oder auf Android beschränkt. Neben Geräten mit Android sind jedoch auch Apples iPhones und iPads stark am Markt vertreten. Da diese mit iOS ein anderes Betriebssystem verwenden, kann Coboli auf diesen nicht genutzt werden. Stattdessen kann dort nur ComicLib im Internetbrowser genutzt werden. Die Vorteile von Coboli können dort nicht genutzt werden. Daher stellt die Entwicklung eines Coboli-Klons für iOS den nächsten logischen Schritt in der Entwicklung der Plattform dar.

4.3 Rückblick

In den vorangegangenen Kapiteln wurden detaillierte Einblicke in die Umsetzung des Softwareprojekts „Coboli“ gegeben. Zu Beginn wurde das Projekt vorgestellt (*1*). Dabei wurden unter anderem die Motivation und Aufgabenstellung vorgestellt, die das Projekt treiben. Das darauffolgende Kapitel (*2*) widmete sich den verwendeten Hilfsmitteln und Technologien, die bei der Entwicklung von Coboli zum Einsatz kamen. Im Anschluss daran wurden detaillierte Einblicke in die Durchführung des Projekts und die Softwarearchitektur gegeben (*3*). Dabei wurden die umgesetzten Funktionen, Activities und Sicherheitsmaßnahmen vorgestellt. Außerdem wurde ein Einblick in die verwendeten Bibliotheken und Schnittstellen gegeben. Zum Abschluss wurde ein Resümee über das Projekt gezogen (*4*). Anhand der Analyse der Zielerreichung wurde der Erfolg des Projekts ermittelt. Außerdem wurden ein Einblick in die Wahl eines Vertriebsweg der Software und ein Ausblick auf die Zukunft des Projekts gegeben.

Quellenverzeichnis

Apache Software Foundation. (2018). *Apache Subversion Features*. Abgerufen am 24. Oktober 2019 von https://subversion.apache.org/features.html

ASUSTeK Computer Inc. (2019). *Nexus 7 (2013)*. Abgerufen am 24. Oktober 2019 von https://www.asus.com/Tablets/Nexus_7_2013/

Atlassian PLC. (2019). *The code collaboration platform for modern software teams*. Abgerufen am 24. Oktober 2019 von https://bitbucket.org/product/features

Bayeux Museum. (2019). *THE BAYEUX TAPESTRY OR THE STORY OF A CONQUEST*. Abgerufen am 24. Oktober 2019 von https://www.bayeuxmuseum.com/en/the-bayeux-tapestry/discover-the-bayeux-tapestry/what-is-the-bayeux-tapestry-about/

CBS Interactive Inc. (2019). *ComicVine API*. Abgerufen am 24. Oktober 2019 von https://comicvine.gamespot.com/api/

Comixology Inc. (2019). *DRM-Free Backups Now Available*. Abgerufen am 24. Oktober 2019 von https://www.comixology.com/drm-free-backup

dejure.org Rechtsinformationssysteme GmbH. (2018). *Datenschutz-Grundverordnung*. Abgerufen am 24. Oktober 2019 von https://dejure.org/gesetze/DSGVO

Deutsches Patent- und Markenamt. (2019). *Nummer der Marke: 001457084*. Abgerufen am 24. Oktober 2019 von https://register.dpma.de/DPMAregister/marke/registerHABM?AKZ=001457084&CURSOR=2

Doist Ltd. (2019). *Ab jetzt alles unter Kontrolle*. Abgerufen am 24. Oktober 2019 von https://todoist.com/tour

Facebook Inc. (2019). *Conceal*. Abgerufen am 24. Oktober 2019 von http://facebook.github.io/conceal/

Fonticons, Inc. (2018). *Font Awesome Free License*. Abgerufen am 24. Oktober 2019 von https://github.com/FortAwesome/Font-Awesome/blob/master/LICENSE.txt

Fonticons, Inc. (2019). *Font Awesome*. Abgerufen am 24. Oktober 2019 von https://fontawesome.com/

Free Software Foundation, Inc. (1991). *GNU General Public License*. Abgerufen am 24. Oktober 2019 von https://www.gnu.de/documents/gpl-2.0.de.html

Free Software Foundation, Inc. (2015). *CVS - Concurrent Versions System*. Abgerufen am 24. Oktober 2019 von http://www.nongnu.org/cvs/

Git Community. (2019). *About Git*. Abgerufen am 24. Oktober 2019 von https://git-scm.com/about

Git Community. (2019). *Distributed*. Abgerufen am 24. Oktober 2019 von https://git-scm.com/about/distributed

Github, Inc. (2019). *How developers work*. Abgerufen am 24. Oktober 2019 von https://github.com/features

GitLab, Inc. (2019). *Product*. Abgerufen am 24. Oktober 2019 von https://about.gitlab.com/product/

Google, Inc. (2019). *Accessing data using Room DAOs - Define methods for convenience*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/data-storage/room/accessing-data#convenience

Google, Inc. (2019). *Accessing data using Room DAOs - Query for information*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/data-storage/room/accessing-data#query

Google, Inc. (2019). *Add the app bar*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/appbar

Google, Inc. (2019). *Android Development Tools for Eclipse*. Abgerufen am 24. Oktober 2019 von https://marketplace.eclipse.org/content/android-development-tools-eclipse

Google, Inc. (2019). *Android Jetpack*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/jetpack

Google, Inc. (2019). *Android keystore system*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/articles/keystore.html

Google, Inc. (2019). *Android Studio*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/studio

Google, Inc. (2019). *Autofill framework*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/guide/topics/text/autofill

Google, Inc. (2019). *Configure your build*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/studio/build

Google, Inc. (2019). *Context - MODE_PRIVATE*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/reference/android/content/Context.html#MODE_PRIVATE

Google, Inc. (2019). *Create and manage virtual devices*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/studio/run/managing-avds

Google, Inc. (2019). *Create views into a database*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/data-storage/room/creating-views

Google, Inc. (2019). *Cryptography*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/guide/topics/security/cryptography

Google, Inc. (2019). *Define relationships between objects - Create nested objects*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/data-storage/room/relationships#nested-objects

Google, Inc. (2019). *Defining data using Room entities*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/data-storage/room/defining-data

Google, Inc. (2019). *Determine and monitor the connectivity status*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/monitoring-device-state/connectivity-monitoring

Google, Inc. (2019). *Fragment*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/reference/android/app/Fragment.html

Google, Inc. (2019). *Getting Started with the NDK*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/ndk/guides

Google, Inc. (2019). *Google I/O 2019: Empowering developers to build the best experiences on Android + Play*. Abgerufen am 24. Oktober 2019 von https://android-developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html

Google, Inc. (2019). *Gson*. Abgerufen am 24. Oktober 2019 von https://github.com/google/gson

Google, Inc. (2019). *Gson - License*. Abgerufen am 24. Oktober 2019 von https://github.com/google/gson#license

Google, Inc. (2019). *Gson User Guide - JSON Field Naming Support*. Abgerufen am 24. Oktober 2019 von https://github.com/google/gson/blob/master/UserGuide.md#json-field-naming-support

Google, Inc. (2019). *Gson User Guide - Primitives Examples*. Abgerufen am 24. Oktober 2019 von https://github.com/google/gson/blob/master/UserGuide.md#primitives-examples

Google, Inc. (2019). *Introduction to Activities*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/guide/components/activities/intro-activities

Google, Inc. (2019). *Java API Framework*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/guide/platform#api-framework

Google, Inc. (2019). *Localize the UI with Translations Editor*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/studio/write/translations-editor

Google, Inc. (2019). *Localize your app - Design a flexible layout*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/guide/topics/resources/localization#design-a-flexible-layout

Google, Inc. (2019). *Localize your app - How to create alternative resources*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/guide/topics/resources/localization#creating-alternatives

Google, Inc. (2019). *Localize your app - Overview: Resource switching in Android*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/guide/topics/resources/localization#resource-switching

Google, Inc. (2019). *Localize your app - Why default resources are important*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/guide/topics/resources/localization#defaults-r-important

Google, Inc. (2019). *Migrating Room databases*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/data-storage/room/migrating-db-versions

Google, Inc. (2019). *Referencing complex data using Room*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/data-storage/room/referencing-data

Google, Inc. (2019). *Room Persistence Library*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/topic/libraries/architecture/room

Google, Inc. (2019). *Save key-value data*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/data-storage/shared-preferences

Google, Inc. (2019). *Sending simple data to other apps*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/sharing/send

Google, Inc. (2019). *Sending the User to Another App*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/training/basics/intents/sending

Google, Inc. (2019). *Update UI components with NavigationUI - Add a navigation drawer*. Abgerufen am 24. Oktober 2019 von https://developer.android.com/guide/navigation/navigation-ui#add_a_navigation_drawer

Google, Inc. (2019). *Welcoming Android 10*. Abgerufen am 24. Oktober 2019 von https://android-developers.googleblog.com/2019/09/welcoming-android-10.html

Hahn, A. (2019). *ComicLib*. Abgerufen am 24. Oktober 2019 von https://github.com/ahahn94/ComicLib

Hahn, A. (2019). *ComicLib*. Abgerufen am 24. Oktober 2019 von https://github.com/ahahn94/ComicLib#testing-data

Hahn, A. (2019). *ComicLib API Version 1*. Abgerufen am 24. Oktober 2019 von https://documenter.getpostman.com/view/5715403/S1a35U3H?version=latest

Hahn, A. (2019). *ComicLib API Version 1 - /api/v1/authenticated*. Abgerufen am 24. Oktober 2019 von https://documenter.getpostman.com/view/5715403/S1a35U3H?version=latest#2afced3e-aff7-4773-a1e6-05f55a3db8ac

Hahn, A. (2019). *ComicLib API Version 1 - /api/v1/issues*. Abgerufen am 24. Oktober 2019 von https://documenter.getpostman.com/view/5715403/S1a35U3H?version=latest#da098b2a-3ba3-439a-82b1-6009b131f940

Hahn, A. (2019). *ComicLib API Version 1 - /api/v1/issues/{id}/file*. Abgerufen am 24. Oktober 2019 von

https://documenter.getpostman.com/view/5715403/S1a35U3H?version=latest#1e6ab9df-406e-4395-96ad-0135b0eb04f8

Hahn, A. (2019). *ComicLib API Version 1 - /api/v1/issues/{id}/readstatus*. Abgerufen am 24. Oktober 2019 von https://documenter.getpostman.com/view/5715403/S1a35U3H?version=latest#4076a072-2693-483d-b3ab-1361b6162709

Hahn, A. (2019). *ComicLib API Version 1 - /api/v1/online*. Abgerufen am 24. Oktober 2019 von https://documenter.getpostman.com/view/5715403/S1a35U3H?version=latest#4674e73c-9365-45d2-b548-00008539b80c

Hahn, A. (2019). *ComicLib API Version 1 - /api/v1/publishers*. Abgerufen am 24. Oktober 2019 von https://documenter.getpostman.com/view/5715403/S1a35U3H?version=latest#ec4c6c25-c018-47a0-b7c5-3b4aa298a056

Hahn, A. (2019). *ComicLib API Version 1 - /api/v1/tokens*. Abgerufen am 24. Oktober 2019 von https://documenter.getpostman.com/view/5715403/S1a35U3H?version=latest#a342b1a5-f8c5-466d-9809-490e5ea2e0f6

Hahn, A. (2019). *ComicLib API Version 1 - /api/v1/volumes*. Abgerufen am 24. Oktober 2019 von https://documenter.getpostman.com/view/5715403/S1a35U3H?version=latest#043433dc-9cb5-4b54-baa3-0b26c3685010

Hahn, A. (2019). *Entwicklung einer freien Software zur Verwaltung von digitalen Comics auf Basis von Web-Technologien.* Technische Hochschule Köln.

Heise Medien. (2011). *WPA-Schlüssel in der Cloud knacken*. Abgerufen am 24. Oktober 2019 von https://www.heise.de/security/meldung/WPA-Schluessel-in-der-Cloud-knacken-1168061.html

Heise Medien. (2019). *Android-Verteilung: Oreo und Pie legen zu*. Abgerufen am 24. Oktober 2019 von https://www.heise.de/newsticker/meldung/Android-Verteilung-Oreo-und-Pie-legen-zu-4417181.html

HMD Global Oy. (2019). *Nokia 8.1. Erwarte mehr.* Abgerufen am 24. Oktober 2019 von https://www.nokia.com/phones/de_de/nokia-8

Humble Bundle Inc. (2018). *HUMBLE COMICS BUNDLE: DOCTOR WHO 2018 BY TITAN*. Abgerufen am 24. Oktober 2019 von https://web.archive.org/web/20181010175907/https://www.humblebundle.com/books/doctor-who-comics-2018

IETF Trust. (2012). *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc6750

IETF Trust. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Authentication - 401 Unauthorized*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc7235#section-3.1

IETF Trust. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content - 200 OK*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc7231#section-6.3.1

IETF Trust. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content - 404 Not Found*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc7231#section-6.5.4

IETF Trust. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content - 405 Method Not Allowed*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc7231#section-6.5.5

IETF Trust. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content - DELETE*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc7231#section-4.3.5

IETF Trust. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content - GET*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc7231#section-4.3.1

IETF Trust. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content - POST*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc7231#section-4.3.3

IETF Trust. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content - PUT*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc7231#section-4.3.4

JetBrains. (2018). *Kotlin plugin for NetBeans IDE*. Abgerufen am 24. Oktober 2019 von https://github.com/JetBrains/kotlin-netbeans

JetBrains. (2019). *Comparison to Java Programming Language*. Abgerufen am 24. Oktober 2019 von https://kotlinlang.org/docs/reference/comparison-to-java.html

JetBrains. (2019). *Getting Started with Eclipse IDE*. Abgerufen am 24. Oktober 2019 von https://kotlinlang.org/docs/tutorials/getting-started-eclipse.html

McCloud, S. (1993). Understanding Comics - The Invisible Art. Kitchen Sink Press.

McCloud, S. (1993). Understanding Comics - The Invisible Art. Kitchen Sink Press.

Mercurial Community. (2019). *Mercurial source control management*. Abgerufen am 24. Oktober 2019 von https://www.mercurial-scm.org/about

National Geographic. (2015). *Römisches Reich: Die Säule von Trajan*. Abgerufen am 24. Oktober 2019 von https://www.nationalgeographic.de/geschichte-und-kultur/roemisches-reich-die-saeule-von-trajan

NBANDROIDTEAM. (2019). *NBANDROID-V2*. Abgerufen am 24. Oktober 2019 von https://github.com/NBANDROIDTEAM/NBANDROID-V2

New York Public Library. (2011). *So, Why Do We Call It Gotham, Anyway?* Abgerufen am 24. Oktober 2019 von https://www.nypl.org/blog/2011/01/25/so-why-do-we-call-it-gotham-anyway

Postman, Inc. (2019). *The Collaboration Platform for API Development*. Abgerufen am 24. Oktober 2019 von https://www.getpostman.com/

Publishers Weekly. (2006). *New York Is Comics Country*. Abgerufen am 24. Oktober 2019 von https://web.archive.org/web/20070817120136/http://www.publishersweekly.com/article/CA6302532.html

Raspberry Pi Foundation. (2019). *Raspberry Pi 3 Model B*. Abgerufen am 24. Oktober 2019 von https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Raspberry Pi Foundation. (2019). *Raspbian*. Abgerufen am 24. Oktober 2019 von https://www.raspberrypi.org/downloads/raspbian/

RedaktionsNetzwerk Deutschland GmbH. (2017). *Das sind die meistgenutzten Passwörter 2017*. Abgerufen am 24. Oktober 2019 von https://www.haz.de/Nachrichten/Digital/Das-sind-die-meistgenutzten-Passwoerter-2017

Schnabel, P. (2019). *DNS - Domain Name System*. Abgerufen am 24. Oktober 2019 von https://www.elektronik-kompendium.de/sites/net/0901141.htm

Schnabel, P. (2019). *Man-in-the-Middle*. Abgerufen am 24. Oktober 2019 von https://www.elektronik-kompendium.de/sites/net/1710251.htm

Schnabel, P. (2019). *SSL - Secure Socket Layer*. Abgerufen am 24. Oktober 2019 von https://www.elektronik-kompendium.de/sites/net/0902281.htm

Slashdot Media. (2019). *Discover the New SourceForge*. Abgerufen am 24. Oktober 2019 von https://sourceforge.net/create/

Square, Inc. (2019). *OkHttp*. Abgerufen am 24. Oktober 2019 von https://square.github.io/okhttp/

Square, Inc. (2019). *OkHttp - License*. Abgerufen am 24. Oktober 2019 von https://square.github.io/okhttp/#license

Square, Inc. (2019). *Retrofit*. Abgerufen am 24. Oktober 2019 von https://square.github.io/retrofit/

Square, Inc. (2019). *Retrofit - Introduction*. Abgerufen am 24. Oktober 2019 von https://square.github.io/retrofit/#introduction

Square, Inc. (2019). *Retrofit - License*. Abgerufen am 24. Oktober 2019 von https://square.github.io/retrofit/#license

Square, Inc. (2019). *Retrofit - Retrofit Configuration*. Abgerufen am 24. Oktober 2019 von https://square.github.io/retrofit/#restadapter-configuration

Süddeutsche Zeitung Digitale Medien GmbH. (2016). *So sorglos gehen Nutzer mit Passwörtern um*. Abgerufen am 24. Oktober 2019 von

https://www.sueddeutsche.de/digital/it-sicherheit-forscher-zeigen-wie-sorglos-nutzer-mit-passwoertern-umgehen-1.3280425

The British Museum. (2008). *Codex Zouche-Nuttall*. Abgerufen am 24. Oktober 2019 von https://web.archive.org/web/20080927082504/http://www.britishmuseum.org/explore/highlights/highlight_objects/aoa/c/codex_zouche-nuttall.aspx

The Hollywood Reporter. (2014). *Valiant Entertainment Goes DRM-Free With New Digital Partnership*. Abgerufen am 24. Oktober 2019 von https://www.hollywoodreporter.com/heat-vision/valiant-entertainment-goes-drm-free-726044

The Internet Society. (1999). *HTTP Authentication: Basic and Digest Access Authentication - Basic Authentication Scheme*. Abgerufen am 24. Oktober 2019 von https://tools.ietf.org/html/rfc2617#section-2

The New York Observer. (2010). *Look! Up in the Air! Marvel Has a 60K-Foot Sublease!* Abgerufen am 24. Oktober 2019 von https://web.archive.org/web/20151009030057/https://observer.com/2010/09/look-up-in-the-air-marvel-has-a-60kfoot-sublease/

The San Diego Union-Tribune. (2013). *Warner's DC comic-book unit leaving Gotham*. Abgerufen am 24. Oktober 2019 von https://web.archive.org/web/20190815202822/https://www.sandiegouniontribune.com/sdut-warners-dc-comic-book-unit-leaving-gotham-2013oct29-story.html

University Press Of Mississippi. (2010). *A critical study of the Swiss artist who created the comic strip*. Abgerufen am 24. Oktober 2019 von https://web.archive.org/web/20101206155345/https://www.upress.state.ms.us/books/869

Vogel IT-Medien GmbH. (2018). *Was ist ein Brute-Force-Angriff?* Abgerufen am 24. Oktober 2019 von https://www.security-insider.de/was-ist-ein-brute-force-angriff-a-677192/

Wagner, E. (2012). *JUnRAR*. Abgerufen am 24. Oktober 2019 von https://github.com/edmund-wagner/junrar

Wagner, E. (2012). *JUnRAR - Commits*. Abgerufen am 24. Oktober 2019 von https://github.com/edmund-wagner/junrar/commits/master

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Abkürzung	**Bedeutung**
API	„Application Programming Interface“. Eine Programmierschnittstelle.
AZW	Ein Dateiformat, das von Amazon Kindle für E-Books genutzt wird.
CBR	„Comic Book RAR“. Ein Dateiformat zur Speicherung digitaler Comichefte, das auf dem RAR-Archivformat basiert.
CBZ	„Comic Book ZIP“. Ein Dateiformat zur Speicherung digitaler Comichefte, das auf dem RAR-Archivformat basiert.
CRUD	„Create Read Update Delete“. Die vier grundlegenden Operationen auf persistenten Speichern: Erstellen, Lesen, Ändern, Löschen.
CVS	„Concurrent Versions System“. Eine Versionsverwaltungssoftware.
DAO	„Data Access Object“. Siehe dort.

DC	„Detective Comics“. Ein US-amerikanischer Comicverlag.
DNS	„Domain Name System“. Ein Dienst zur Auflösung von Domainnamen zu IP-Adressen.
DRM	„Digital Rights Management“. Siehe dort.
ERD	„Entity Relationship Diagram“. Siehe dort.
GNU	„GNU’s Not Unix“. Ein Unix-ähnliches Betriebssystem, dessen Entwicklung auch zum Entwurf der GNU General Public License führte.
GPL	„General Public License“. Siehe dort.
HTTP	„Hypertext Transfer Protocol“. Ein Protokoll zur Übertragung von Dokumenten und Dateien über das Internet. Wird vor allem zum Laden von Webseiten geladen und wird zunehmend durch HTTPS ersetzt, weil dieses Verschlüsselung verwendet.
HTTPS	„Hypertext Transfer Protocol Secure“. Das um TLS-Verschlüsselung erweiterte HTTP-Protokoll.
ID	„Identifikator“. Ein eindeutig identifizierendes Merkmal eines Objekts.
IDE	„Integrated Development Environment“. Siehe Integrierte Entwicklungsumgebung.
IV	„Initialization Vector“. Siehe dort.
JSON	„JavaScript Object Notation“. Ein ursprünglich aus JavaScript stammendes Datenformat zur Darstellung strukturierter Daten als Text.
JVM	„Java Virtual Machine“. Siehe dort.
LAN	„Local Area Network“. Ein lokales Computernetz. Bezeichnet häufig synonym den Ethernet-Standard.
LTE	„Long Term Evolution“. Ein Mobilfunkstandard.
NDK	„Native Development Kit“. Siehe dort.
PDF	„Portable Document Format“. Ein Dateiformat für digitale Dokumente.
PHP	„PHP: Hyptertext Preprocessor“. Eine Programmiersprache zur Entwicklung von Web-Anwendungen.
RAR	„Roshal Archive“. Ein Archivdateiformat.
REST	„Representational State Transfer“. Ein Paradigma zur Gestaltung von Programmierschnittstellen.
SQL	„Structured Query Language“. Eine standardisierte Datenbanksprache.
TLS	„Transport Layer Security“. Ein Protokoll zur Verschlüsselung von Datenübertragungen
URL	„Uniform Resource Locator“. Eine Zeichenfolge, die eine Ressource an Hand des verwendeten Protokolls und des Pfads zur Ressource lokalisiert.
UTC	„Universal Time Coordinated“. Koordinierte Weltzeit.
VPN	„Virtual Private Network“. Eine verschlüsselte Netzwerkverbindung zur sicheren Kommunikation über ein unsicheres Medium.
W-LAN	„Wireless Local Area Network“. Ein drahtloses, lokales Computernetzwerk. Bezeichnet meistens die Funknetzwerke nach den Standards der IEEE-802.11-Familie.
XML	„Extensible Markup Language“. Eine Sprache zur Darstellung strukturierter Daten als Text (Auszeichnungssprache).

ZIP	Vom englischen Wort „Zipper“. Ein Archivdateiformat.

Tabelle 8 Abkürzungsverzeichnis

Glossar

Begriff	**Bedeutung**
Abhängigkeitsmanagement	Management der durch eine Software benötigten Bibliotheken.
Action-Bar	Die Menüleiste am oberen Ende einer Activity in einer Android-App. Sie enthält häufig die Suchfunktion und den Button zum Einblenden des Navigationsmenüs.
Activity	Eine Hauptsicht innerhalb einer Android-App. Sie kann andere Sichten (Fragments) enthalten.
AES/GCM/NoPadding	Ein Verschlüsselungsalgorithmus im Android-Framework. Daten werden mit AES unter Verwendung der Parameter GCM und NoPadding verschlüsselt.
Albumansicht	Eine Ansicht, die Karten in Zeilen und Spalten organisiert. Dadurch sieht die Ansicht wie ein Bilderalbum aus, in dem Fotos eingeklebt sind.
Amazon Kindle	Die E-Book-Plattform von Amazon.
Android Framework	Die Sammlung der Schnittstellen zum Zugriff auf die Android-Systemdienste.
Android Java API Framework	Siehe Android Framework.
Android Jetpack	Eine Sammlung von Hilfsbibliotheken zur Entwicklung von Android-Apps.
Android Studio	Die offizielle integrierte Entwicklungsumgebung für Android.
Anfrage-Body	Siehe HTTP Message Body. Hier auf die Anfrage-Nachricht bezogen.
Anmeldedatenspeicher	Im Zusammenhang mit Autofill der Speicher, indem die Anmeldedaten zum automatischen Ausfüllen von Anmeldemasken hinterlegt ist.
Annotation	Eine Anmerkung zu einer Klasse oder einem Attribut, die dem annotierten Objekt Metadaten hinzufügt.
Antragsteller	Die Organisation, die bei einer Zertifizierungsstelle den Antrag auf ein Zertifikat stellt.
Apache JMeter	Eine Software zur Durchführung von Lasttests an Web-Services.
Apache License Version 2	Eine Lizenz für freie Software der Apache Software Foundation.
Apache Subversion	Eine Versionsverwaltungssoftware.
API-Endpunkt	Der Zugriffspunkt auf die API, also die Adresse der API-Wurzel auf dem API-Server.
API-Ressource	Eine Informationsressource innerhalb einer Programmier-

	schnittstelle.
API-Schlüssel	Eine kryptografische Zeichenkette zur Authentifizierung an einer Programmierschnittstelle.
APK-Format	Das Dateiformat von Android-Apps.
Apple Books	Die E-Book-Plattform von Apple.
Asus Nexus 7 (2013)	2013er Modell eines 7“-Android-Tablets der Marke Asus. Nicht identisch mit dem 2012er Modell, daher die Jahreszahl im Namen.
Asymmetrische Verschlüsselung	Eine Verschlüsselungsmethode, die je einen Schlüssel zur Verschlüsselung (öffentlicher Schlüssel) und zur Entschlüsselung (privater Schlüssel) nutzt.
Ausführungskonfiguration	Eine Sammlung von Einstellungen zum Bauen und Ausführen von Software.
Authentifizierung	Ein Identitätsnachweis zur Erlangung von Berechtigungen in einem Softwaresystem.
Autofill	Ein Dienst, der ab Android 8 zur Verfügung steht und das automatische Eintragen von Anmeldedaten in Anmeldemasken ermöglicht.
Batman	Eine Comicfigur von DC Comics.
Bearer Token Authorization	Ein Verfahren zur Authentifizierung mittels eines einzelnen Schlüssels (Bearer Token) als Identitätsnachweis.
Bilder-Cache	Ein Zwischenspeicher für Bilder.
Bitbucket	Eine Internetplattform zum gemeinsamen Bearbeiten und Bereitstellen von Quellcode.
Boolean	Ein Datentyp für Wahrheitswerte. Seine möglichen Werte sind true (wahr) und false (falsch).
Brute-Force	Eine Brute-Force-Methode („rohe Gewalt-Methode“) versucht, ein Problem durch das Ausprobieren aller Möglichkeiten zu lösen. Es wird nur so lange ausprobiert, bis die Lösung gefunden ist.
Bugtracker	Eine Software zum kollaborativen Sammeln und Beheben von Fehlern in Softwareprojekten.
Buildskript	Eine Datei, die Anweisungen zum Bauen einer Software aus deren Quelltext enthält.
Build-System	Ein Softwaresystem zum Bauen von Software aus deren Quelltext.
Bytecode	Ein Zwischencode zwischen Quelltext und ausführbarem Binärcode. Er ist für Menschen nur noch schwer lesbar und kann mit Hilfe eines Interpreters (z.B. JVM) als Programm ausgeführt werden.
Camel-Case	Eine Schreibweise für zusammengesetzte Wörter (vor allem in Quelltext), bei der der Anfang jedes neuen Worts großgeschrieben wird. Den dadurch entstehenden „Höckern“ verdankt diese Schreibweise ihren Namen.
Certbot	Ein Programm, das automatisch Zertifikate von Let’s Encrypt anfordern und verlängern kann.
Cipher-Klasse	Eine Klasse im Android-Framework, die Funktionen und Algorithmen zur Ver- und Entschlüsselung von Daten zur Ver-

	fügung stellt.
Ciphertext	Wird auch „Geheimtext“ genannt. Das Ergebnis der Verschlüsselung eines Klartexts.
Cloud-Symbol	Ein Symbol einer Wolke, das in Coboli darauf hinweist, dass ein Comic nur in der „Cloud“ (also auf einem entfernten Rechner) vorliegt.
Codex Nuttall	Eine Bildhandschrift auf Hirschhaut, die die Geschichte eines Herrschers der Mixteken in Form von Bildern und Ideogrammen erzählt.
Comic	Ein Erzählmedium, das eine Bilderfolge aus voneinander abgegrenzten Bildern mit Text kombiniert.
Comic-Cache	Der Zwischenspeicher für die Comicdateien.
ComicLib	Eine Web-Anwendung zur Verwaltung digitaler Comicsammlungen.
Comic-Strip	Eine Folge von Panels eines Comics.
ComicVine	Die weltweit größte, allgemein zugängliche Comicdatenbank.
Comixology	Ein Anbieter für digitale Comics. Wurde zwischenzeitlich von Amazon aufgekauft, existiert jedoch weiterhin.
Commit	Ein Eintrag in einer Versionshistorie.
Conceal	Eine Java-Bibliothek zur Verschlüsselung von Daten, die von Facebook entwickelt wird.
Content Provider	Eine Klasse von Diensten, die in Android dazu dienen, anderen Apps Zugriff auf Daten im geschützten Speicherbereich einer App zu gewähren.
Convenience-Methode	Eine Funktion, die dem Entwickler eine lästige Aufgabe einfacher gestaltet, zum Beispiel indem sie die Komplexität der Aufgabe reduziert. Beispiel: eine Funktion, die gegen Eingabe eines Dateinamens den Inhalt der Datei einliest und dabei den lästigen Teil (Arbeiten mit Pointern und Buffern) wegabstrahiert.
Converter	Ein modularer Bestandteil von Retrofit, der der Deserialisierung von HTTP Message Bodies dient.
Creative Commons CC BY 4.0 Lizenz	Eine Lizenz zur Veröffentlichung von Dokumenten zur gemeinfreien Nutzung durch die Öffentlichkeit.
Dare Devil	Eine Comicfigur von Marvel.
Data Access Object	Die Methoden zum Zugriff auf eine Datenquelle werden in einem Objekt gekapselt. Die hinter dem DAO stehende Datenquelle ist dabei austauschbar.
Debian	Ein auf Linux basierendes Betriebssystem.
DELETE	Eine Methode des HTTP-Protokolls, die eine Ressource löscht.
Depot	Der Speicherort eines Softwareprojekts innerhalb eines Versionsverwaltungssystems.
Deserialisierung	Die Rückumwandlung zuvor serialisierter Daten in ihr ursprüngliches Format.
Die Fantastischen Vier	Eine Gruppe von Comicfiguren von Marvel.

Digital Rights Management	Maßnahmen zum elektronischen Kopierschutz von digitalen Gütern.
DNS-Namensauflösung	Das Auflösen eines Domainnamens zu einer IP-Adresse durch DNS.
Docker	Eine Software zur Virtualisierung von Software-Anwendungen.
Domainname	Ein Name, der einen Teilbereich (Domain) innerhalb des Domain Name Systems (DNS) kennzeichnet.
E-Book	Ein elektronisches/digitales Buch.
Eclipse	Eine integrierte Entwicklungsumgebung, die für viele Programmiersprachen genutzt werden kann.
Einwegfunktion	Eine Funktion, die für einen Eingabewert stets denselben Ergebniswert liefert, welcher jedoch keine Rekonstruktion des Eingabewerts ermöglicht.
Entity Relationship Diagramm	Ein Diagramm, dass Entitäten und deren Beziehungen zueinander visualisiert.
Filter-Funktion	Eine Funktion, die nur diejenigen Elemente einer Liste liefert, die die ihr übergebene Bedingung erfüllen.
Font Awesome	Eine Bibliothek, die Icons als eine Schriftart mit eigenen Unicode-Zeichen bereitstellt.
Fragment	Eine Subsicht innerhalb einer Android-App. Sie kann in einer Activity oder als Dialog angezeigt werden.
Freie Software	Software, die nicht proprietär lizensiert ist, sondern von jedem frei genutzt, geteilt und weiterentwickelt werden darf.
General Public License V2	Eine häufig verwendete Lizenz für freie Software.
GET	Eine Methode des HTTP-Protokolls, die den Inhalt einer Ressource anfordert.
Git	Ein Software zur verteilten Versionsverwaltung.
GitHub	Eine Internetplattform zum gemeinsamen Bearbeiten und Bereitstellen von Quellcode.
GitHub Pages	Eine Funktion von GitHub, die das Erstellen von Projektwebseiten ermöglicht.
GitLab	Eine Internetplattform zum gemeinsamen Bearbeiten und Bereitstellen von Quellcode.
Google Pixel 3	Ein Android-Smartphone der Marke Google.
Gotham City	Eine fiktive Stadt in der Comicwelt von DC Comics. Außerdem ein alter Spitzname der Stadt New York.
Gradle	Ein Buildsystem, das unter anderem zum Bauen von Android-Apps zum Einsatz kommt.
Gson	Eine Bibliothek zur Serialisierung und Deserialisierung von JSON-Datensätzen.
Hashing	Das Umwandeln einer Eingabe in einen Wert eines definierten Wertebereichs mithilfe einer Einwegfunktion.
Header	Der Metadatenteil eines Datenpakets.
Hostname	In einem TLS-Zertifikat der Name des Servers innerhalb des

	Domain Name Systems (DNS).
HTTP Basic Authentication	Ein Verfahren zur Authentifizierung mittels Benutzername und Passwort, das im Standard zum HTTP-Protokoll definiert ist.
HTTP Message Body	Der Nutzdatenteil einer HTTP-Nachricht.
HTTP-Client	Ein Clientprogramm zur Nutzung des HTTP-Protokolls.
HTTP-Status	Der HTTP-Statuscode.
HTTP-Statuscode	Der im Header einer HTTP-Nachricht enthaltene Statuscode, der dem Client Auskunft über den Erfolg der Anfrage erteilt.
Humble Bundle	Ein Anbieter für digitale Güter, der vor allem durch den Verkauf von Paketangeboten nach dem Zahle-so-viel-du-willst-Prinzip bekannt ist.
Icon	Ein Piktogramm.
Ideogramm	Ein Bildzeichen, das für ein ganzes Wort oder einen Begriff steht, zum Beispiel „%“ oder „€“.
Image Comics	Ein US-amerikanischer Comicverlag.
In-App-Einstellungen	Die Einstellungen innerhalb einer App, die über den Neustart hinaus gespeichert werden.
Initialization Vector	Der Initialisierungsvektor für einen Zufallszahlengenerator.
Integer	Ein Datentyp für Ganzzahlen.
Integrierte Entwicklungsumgebung	Eine Sammlung von Software-Werkzeugen zur Entwicklung von Softwareprojekten.
IntelliJ IDEA	Eine IDE-Plattform der Firma JetBrains, die als Basis für spezialisierte IDEs für unterschiedliche Programmiersprachen dient.
Interface	Eine Schnittstellenspezifikation. Ein Interface gibt Klassen, die es implementieren, vor, welche Attribute und Funktionen mindestens enthalten sein müssen.
IP-Adresse	Eine Adresse innerhalb eines Internet-Protocol-Netzwerks.
Issue	Englisch für „Heft“.
Iterator-Funktion	Eine Funktion, die auf jedes Element einer Liste angewendet wird.
Java	Eine Programmiersprache für allgemeine Anwendungen.
Java Virtual Machine	Die virtuelle Maschine, die benötigt wird, um Java-Programme auszuführen.
JUnRAR	Eine Java-Implementation von UnRAR, einer Software zum Entpacken von RAR-Archivdateien.
Kanban	Eine Methode zur Planung von Arbeit in der Softwareentwicklung, bei der eine obere Grenze für die Anzahl paralleler Arbeiten gesetzt wird.
Karte	Ein Element in einer Albumansicht. Es sieht ähnlich wie eine Karteikarte oder Spielkarte aus, daher der Name.
Keystore	Ein Android-Systemdienst, der dem Erzeugen und sicheren Speichern von kryptografischen Schlüsseln und Zertifikaten dient.

Kotlin	Eine Programmiersprache für allgemeine Anwendungen, die unter anderem auf Java basiert.
Layout	Die Anordnung von Elementen auf einer Seite.
Let’s Encrypt	Eine freie und automatisierte Zertifizierungsstelle für X509-Zertifikate zur Nutzung mit TLS.
Linux-Distribution	Ein auf dem Linux Betriebssystemkern basierendes Betriebssystem.
Lokalisierung	Die Anpassung einer Software an die Gegebenheiten einer Region oder Sprache.
MAC-Adressenfilter	Ein Filter, der nur explizit zugelassene Netzwerk-Hardwareadressen in ein Netzwerk hineinlässt.
Manga	Mangas sind japanische Comics. Sie werden von hinten nach vorne und von rechts nach links gelesen.
Man-in-the-Middle-Attacke	Eine Attacke auf einen Kommunikationskanal, bei der sich der Angreifer zwischen die Kommunikationspartner schummelt und die Kommunikation belauscht.
Mappen	Das Abbilden von zwei Datenmodellen aufeinander.
Mapping-Funktion	Eine Funktion, die aus einer Liste eines Typs eine Liste anderen Typs macht, indem sie auf jedes Listenelement eine Konvertierungsfunktion anwendet.
Marvel	Ein US-amerikanischer Comicverlag.
Mercurial	Ein Software zur verteilten Versionsverwaltung.
Metropolis	Eine fiktive Stadt in der Comicwelt von DC Comics.
Mitteilungszentrum	Das Mitteilungszentrum sammelt in Android die Benachrichtigungen aller Apps.
Mixteken	Eine Ureinwohnergruppe Mexikos.
Native Development Kit	Ein Framework zur Verwendung von C und C++ Code in Android.
Navigation-Drawer	„Navigations-Schublade“. Ein Navigationsmenü, das wie eine Schublade vom linken Rand des Bildschirms hereingezogen werden kann.
NetBeans	Eine integrierte Entwicklungsumgebung, die für viele Programmiersprachen genutzt werden kann.
Nokia 8.1	Ein Android-Smartphone, das von HMD Global unter der Marke Nokia vertrieben wird.
null	In vielen Programmiersprachen (z.B. Java, Kotlin) die Referenz auf eine nicht existierende Variable.
Null-Pointer-Exception	Ein Fehler in Java, der auftritt, wenn versucht wird, eine Variable, die null ist, so zu behandeln, als hätte sie einen Wert.
Öffentlicher Schlüssel	Der Schlüssel, der bei asymmetrischer Verschlüsselung zur Verschlüsselung dient.
Offline	Die Nicht-Verfügbarkeit einer Internetverbindung.
OkHttp	Eine Bibliothek, die einen HTTP-Client für Java und Kotlin zur Verfügung stellt.
Online-Status	Die Information darüber, ob Coboli eine Verbindung zum

	ComicLib-Server aufbauen kann oder nicht.
Open Source	Ähnlich wie Freie Software geht es bei Open Source darum, den Quelltext von Software zur Verfügung zu stellen. Bei Open Source geht es jedoch mehr um den technischen als den philosophischen Aspekt freier Software.
Overlay	Ein Element wird vor den restlichen Bildschirminhalten eingeblendet und verdeckt diese zum Teil.
Panel	Ein Einzelbild in einem Comic(-Strip)
Panelgrenze	Die Grenze eines Einzelbilds (Panel) in einem Comic(-Strip).
Performance	Ausführungsgeschwindigkeit von Software.
PhpStorm	Eine auf der IntelliJ IDEA Plattform basierende IDE zur Entwicklung von Webanwendungen mit PHP.
Popup-Menü	Ein Menü, dass erst nach einer Aktion (Z.B. Klick auf einen Button) angezeigt wird.
Port	Der Teil einer Netzwerkadresse, der Aufschluss über das Programm gibt, an das sich die Verbindung richtet.
POST	Eine HTTP-Methode, die dem Erstellen von Ressourcen dient.
Postman	Eine Entwicklungsumgebung für Programmierschnittstellen.
Primitiver Datentyp	Ein Datentyp, der nur einen Wert eines fest definierten Wertbereichs aufnehmen kann. Bspw. sind Integer, Boolean, Double und Char primitive Datentypen, Listen jedoch nicht.
Privater Schlüssel	Der Schlüssel, der bei asymmetrischer Verschlüsselung zur Entschlüsselung dient.
Proprietär	Technologien, die nicht frei verfügbar sind, sind proprietär. Im Bereich der Software bilden proprietär und frei bzw. Open Source ein Gegensatzpaar.
Publisher	Englisch für „Verlag“.
PUT	Eine HTTP-Methode, die dem Aktualisieren des Inhalts einer Ressource dient.
Raspberry Pi	Ein Einplatinenrechner in Kreditkartengröße mit geringem Stromverbrauch, der häufig für Bastelprojekte genutzt wird.
Raspberry Pi 3 Model B	Ein Einplatinenrechner der Raspberry Pi Familie.
Raspbian	Ein auf Debian basierendes Betriebssystem, das für den Betrieb auf Rechnern der Raspberry Pi Familie optimiert wurde.
Rendern	Das Erstellen einer Grafik aus Rohdaten. In Coboli werden Dokumentenseiten aus PDF-Dokumenten zu je einem Bild pro Seite gerendert.
Repository	Ein Quelltext-Depot.
Ressourcenpfad	Der Pfad zu einer Ressource einer API.
Ressourcensystem	In Android werden viele Inhalte, beispielsweise Layouts oder Zeichenketten, als Ressourcendateien bereitgestellt, für die mehrere Varianten existieren können. Das Ressourcensystem lädt die jeweils benötigte Variante einer Ressourcendatei.
Retrofit	Eine Bibliothek, die einen auf Web-APIs spezialisierten

	HTTP-Client für Java und Kotlin bereitstellt.
Room	Eine Bibliothek zur persistenten Speicherung von Datenobjekten in Android. Fungiert als Abstraktionsschicht über SQLite.
Selbstsigniertes Zertifikat	Ein TLS-Zertifikat, das nicht von einer öffentlichen Zertifizierungsstelle herausgegeben wurde.
Select-Anfrage/Befehl	Eine SQL-Anfrage, die dem Lesen aus einer Datenbank dient.
Serialisierung	Das Umwandeln eines Datenobjekts in eine serielle Repräsentation seiner selbst.
Shared Preferences	Ein Android-Systemdienst, der die Speicherung von Einstellungen innerhalb einer App ermöglicht.
SIL Open Font License 1.1	Eine Lizenz für offene Schriftarten.
Snake-Case	Eine Schreibweise für zusammengesetzte Wörter, bei der zwischen den einzelnen Wörtern ein Unterstrich steht.
SourceForge	Eine Internetplattform zum gemeinsamen Bearbeiten und Bereitstellen von Quellcode.
Spider-Man	Eine Comicfigur von Marvel.
SQL-Anfrage/Befehl	Eine Anfrage an eine Datenbank, die in einem Dialekt der Sprache SQL verfasst ist.
SQLite	Ein auf SQL basierendes Datenbanksystem zur Speicherung einer Datenbank in einer einzigen Datei.
Square, Inc.	Ein US-amerikanisches Unternehmen, das OkHttp und Retrofit entwickelt.
String	Eine Zeichenkette.
String-ID	Jeder Zeichenkette wird im Ressourcensystem ein identifizierender Name zugeordnet. Übersetzungen derselben Zeichenkette verwenden die gleiche ID.
Strings-Datei	Eine Ressourcendatei, die Zeichenketten enthält.
Superman	Eine Comicfigur von DC Comics.
Symmetrischer Schlüssel	Ein Schlüssel, der bei symmetrischer Verschlüsselung sowohl zur Ver- als auch zur Entschlüsselung dient.
Systemsprache	Die im System für Systemtexte (bspw. Dialoge, Benachrichtigungen, Buttons) eingestellte Sprache.
The Tortures Of Saint Erasmus	Eine Bildgeschichte, die in mehreren Bildern die Folterungen des Heiligen Erasmus von Antiochia während der Christenverfolgung zeigt.
Timeout	Die Zeit, die auf ein Ereignis gewartet werden soll, bevor eine Fehlerbehandlung ausgelöst wird.
TinyInt	Ein Datentyp in SQL zum Speichern der Werte „0“ und „1“. Kann für Wahrheitswerte genutzt werden (siehe Boolean).
Todo	Eine noch zu erledigende Aufgabe, beispielsweise auf einer Todoliste oder in einem Kommentarblock.
Todoist	Eine Software zur Aufgabenplanung mittels Todolisten.
Todoliste	Eine Liste von Todos, also eine Liste ausstehender Aufga-

	ben.
Token	Englisch für „Beweis“. Ein Identitätsnachweis in Form eines einzelnen Merkmals.
Trajan-Säule	Eine zu Ehren des römischen Kaisers Trajan errichtete Ehrensäule aus dem Jahr 113 n. Chr., die in einem umlaufenden Relief die Unterwerfung der Daker durch die Römer darstellt.
transkompilieren	Quelltext von einer Programmiersprache in eine andere oder einen Bytecode übersetzen.
Unicode	Ein Zeichencode für Schriftzeichen.
Update (Datenbank)	Das Aktualisieren eines existierenden Datensatzes in einer Datenbank.
Valiant Entertainment	Ein US-amerikanischer Comicverlag.
Versionsbaum	Die Versionshistorie in einer Versionsverwaltungssoftware ist nicht linear, sondern ein Baum aus Änderungen.
Versionsverwaltung	Eine Software zur Verfolgung von Änderungen an einer Software.
View	Eine Benutzeroberfläche.
View (Datenbank)	Eine virtuelle Tabelle in einer Datenbank, die Felder aus anderen Tabellen in sich vereint.
Virtuelle Maschine	Eine Softwareumgebung zum parallelen Betrieb mehrerer Betriebssysteme auf einem Computer.
Volume	Englisch für „Band“. Hier als Bezeichnung für eine Comicserie.
Web-API	Eine Programmierschnittstelle, die über HTTP oder HTTPS bereitgestellt wird.
Wiki	Eine Wissensdatenbank, die durch ihre Benutzer gepflegt wird.
Wurzel der API	Das Ausgangsverzeichnis der API, dem alle Ressourcen untergeordnet sind.
XML-Datei	Eine Datei, die Inhalte in der Extensible Markup Language enthält.
Zeitstempel	Eine Zeichenkette, die einen Zeitpunkt identifiziert.
Zertifikat	Ein (TLS)-Zertifikat bescheinigt einem Server, dass er tatsächlich über die im Zertifikat angegebenen Eigenschaften (Hostname, öffentlicher Schlüssel) verfügt.
Zertifizierungsstelle	Eine Organisation, die (TLS)-Zertifikate herausgibt. Sie stellt damit das Gegenstück zum Antragsteller dar.

Tabelle 9 Glossar

Anhang

Tabellen

Suchanfrage:	**wdv="Software" and (marke="Manhattan")**		
Datenbe-stand	**Aktenzeichen/Registernummer**	**Markendarstellung**	**Aktenzu-stand**
DE	3020150066184	MANHATTAN DE-SIGN	Marke nicht eingetragen
DE	3020150066249	MANHATTAN AGENTUR	Marke nicht eingetragen
EM	1457084		Marke einge-tragen
EM	2150829	MANHATTAN TOY	Eintragung abgelaufen
EM	17913196		Marke einge-tragen
EM	17913199	MANHATTAN ACTI-VE	Marke einge-tragen

Tabelle 10 DPMAregister-Suchergebnis für „Manhattan“ und Ware „Software“ am 07.10.2019.
Registernummer 1457084 ist die Marke „Manhattan“ in Form eines Logos (daher nicht abgebildet). Leider kann auf Suchergebnisse von DPMAregister nicht verlinkt werden, da es keine Permanentlinks gibt.

Anzahl der Threads	**Anzahl der Anfragen**	**Durchsatz (Anfrage/Sekunde)**
7	1001	9,6
8	1000	10,1
9	999	9,9

Tabelle 11 Maximale Anzahl paralleler Zugriffe auf /authenticated auf einem Raspberry Pi 3 Model B.
Die Daten wurde mittels Apache JMeter ermittelt. Die Durchführung von 1000 Anfragen durch 8 parallele Threads mit je 125 Anfragen pro Thread erzielte den maximalen Durchsatz von 10,1 Anfragen/Sekunde. Für die Anfragen wurde jeweils ein ungültiger API-Schlüssel verwendet.

Abbildungen

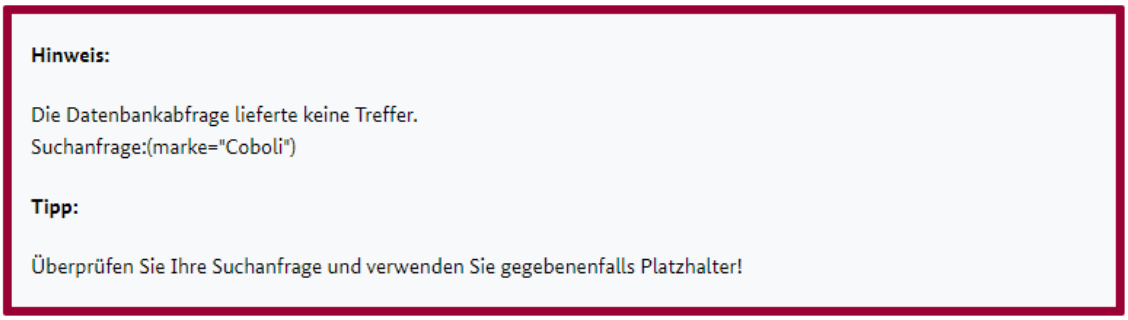

Abbildung 2 DPMAregister-Suchergebnis für "Coboli" am 07.10.2019.
Leider ist in DPMAregister kein Permanentlink auf oder Export von leeren Suchergebnissen möglich.

```
/**
 * Entity data class for volume datasets.
 */
@Entity(
    tableName = "Volumes",
    foreignKeys = [ForeignKey(
        entity = PublisherEntity::class,
        parentColumns = ["ID"],
        childColumns = ["PublisherID"]
    )],
    indices = [Index(value = ["PublisherID"], unique = false)]
)
data class VolumeEntity(

    @SerializedName( value: "ID")
    @ColumnInfo(name = "ID")
    @PrimaryKey
    @NonNull
    var id: String = "",

    @SerializedName( value: "Description")
    @ColumnInfo(name = "Description")
    var description: String = "",

    @SerializedName( value: "ImageFileURL")
    @ColumnInfo(name = "ImageFileURL")
    var imageFileURL: String = "",

    @SerializedName( value: "ReadStatus")
    @Embedded
    var readStatus: ReadStatus? = null,
```

Abbildung 3 Ausschnitt aus der Definition der Datenbanktabelle "Volumes" in Coboli.

```
@DatabaseView(
    value = "SELECT P.ID, P.Description, P.ImageFileURL, P.Name, COUNT (DISTINCT V.ID) " +
            "AS VolumeCount FROM Publishers P LEFT OUTER JOIN Volumes V ON P.ID = V.PublisherID " +
            "GROUP BY P.ID",
    viewName = "PublishersView"
)
data class PublishersView(

    @ColumnInfo(name = "ID")
    @PrimaryKey
    @NonNull
    var id: String = "",

    @ColumnInfo(name = "Description")
    var description: String = "",

    @ColumnInfo(name = "ImageFileURL")
    var imageFileURL: String = "",

    @ColumnInfo(name = "Name")
    var name: String = "",

    @ColumnInfo(name = "VolumeCount")
    var volumeCount: Int = 0

)
```

Abbildung 4 Ausschnitt aus der Definition der Datenbankview "PublisherView"

```
/**
 * Data access object for the CachedComics database table.
 */
@Dao
interface CachedComicsDao {
    @Insert(onConflict = OnConflictStrategy.ABORT)
    fun insert(cachedComicEntity: CachedComicEntity)

    @Update
    fun update(vararg cachedComicEntity: CachedComicEntity)

    @Query( value: "SELECT * FROM CachedComics")
    fun getAll(): Array<CachedComicEntity>

    @Query( value: "SELECT * FROM CachedComics WHERE IssueID = :issueID")
    fun get(issueID: String): CachedComicEntity?

    @Delete
    fun delete(vararg cachedComicEntity: CachedComicEntity)
}
```

Abbildung 5 Definition des Data Access Objects "CachedComicsDao"

App-Screenshots

Die in den Abbildungen gezeigten Verlags-Logos und Comicheft-Titelseiten sind geistiges Eigentum von IDW Publishing und Titan Comics.

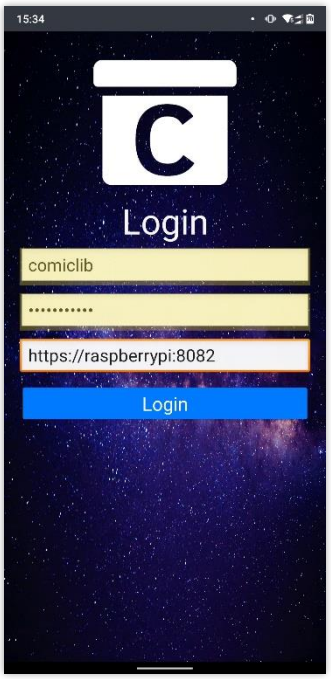

Abbildung 6 *Login-Activity* mit eingetragenen Verbindungsdaten

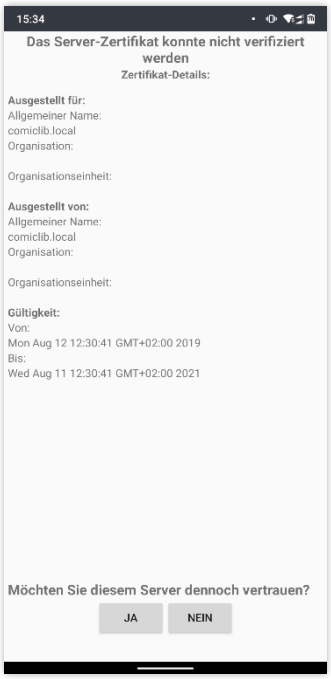

Abbildung 7 Dialog zur Bestätigung der Nutzung eines nicht vertrauenswürdigen Zertifikats

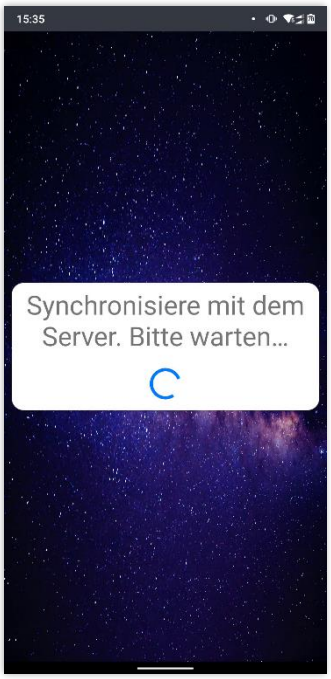

Abbildung 8 *Sync-Activity* mit Wartemeldung

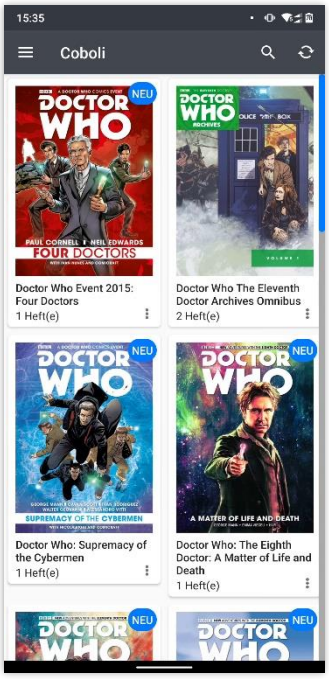

Abbildung 9 *Volumes-Fragment* als Übersicht über alle Serien der Sammlung

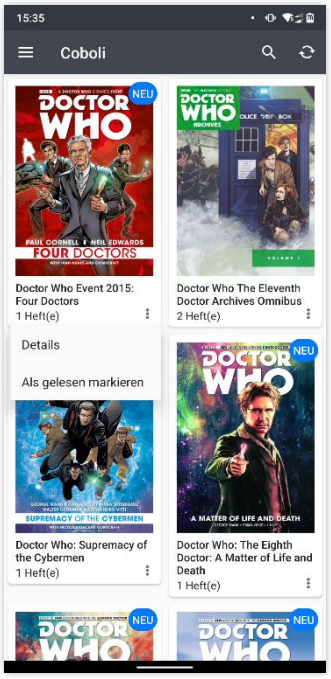

Abbildung 10 Popup-Menü an einer Karte im *Volumes-Fragment*

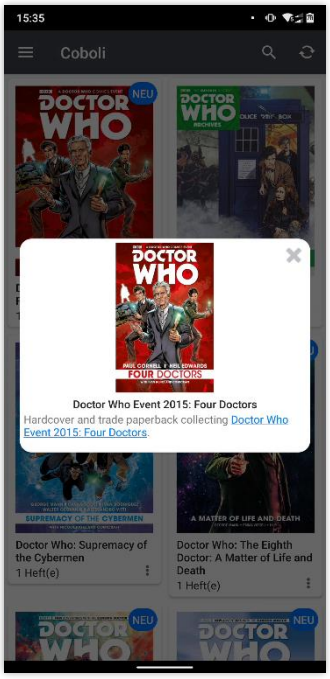

Abbildung 11 Details zu einer Serie als Overlay im *Volumes-Fragment*

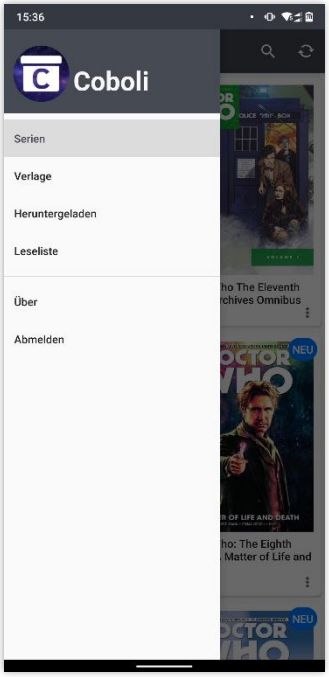

Abbildung 12 Navigationsmenü zum Wechseln zwischen den Fragmenten.

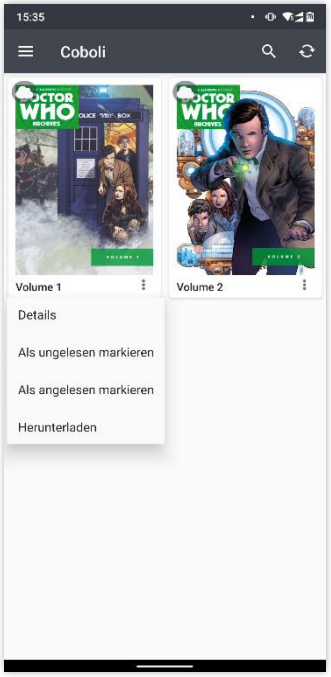

Abbildung 13 *Issues-Fragment* zur Übersicht über die Hefte einer Serie mit Popup-Menü an einer Karte

Abbildung 14 Details zu einem Heft als Overlay im *Issues-Fragment*

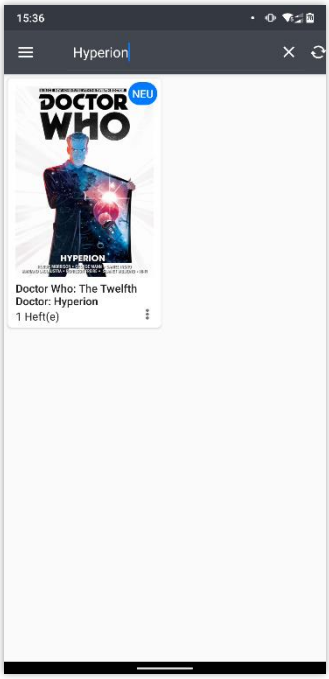

Abbildung 15 Serien als Ergebnisse einer Suche nach "Hyperion"

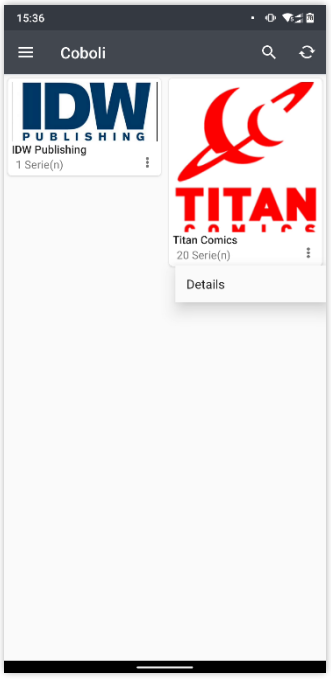

Abbildung 16 *Publishers-Fragment* zur Übersicht über die Verlage mit Popup-Menü an einer Karte

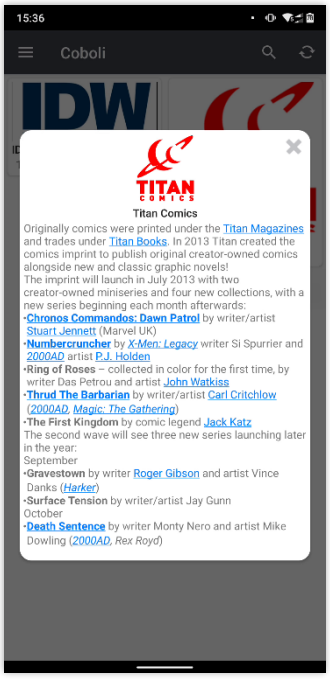

Abbildung 17 Details zu einem Verlag als Overlay im *Publishers-Fragment*

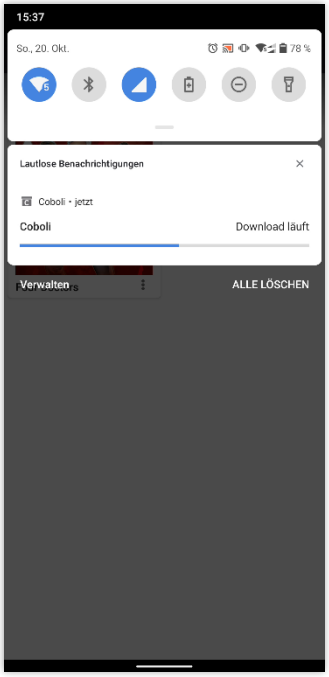

Abbildung 18 Benachrichtigung über einen laufenden Download

Abbildung 19 Benachrichtigung über einen abgeschlossenen Download

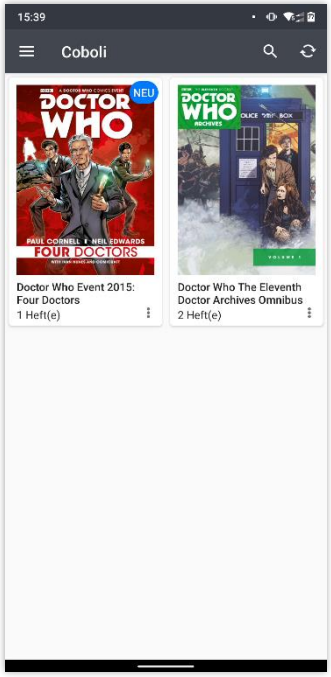

Abbildung 20 Liste der Serien mit heruntergeladenen Heften auf Basis des *Volumes-Fragments*

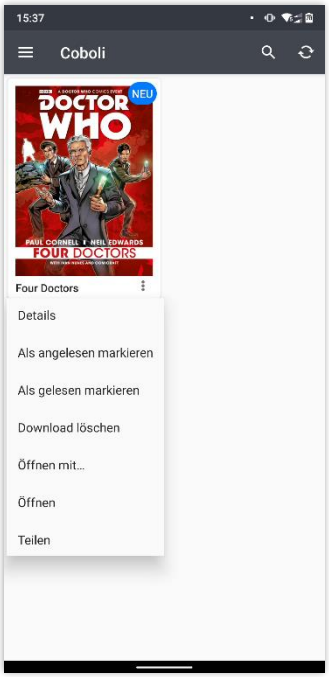

Abbildung 21 Popup-Menü an einem heruntergeladenen Heft im *Volumes-Fragment*

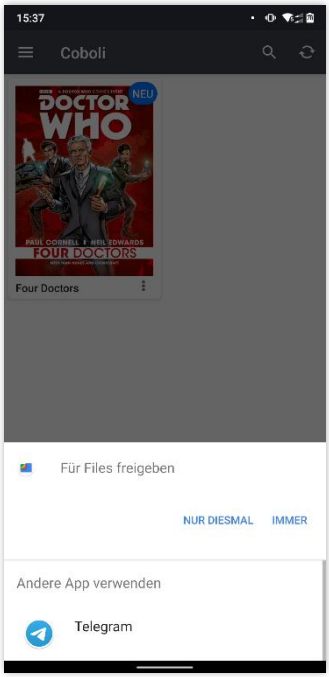

Abbildung 22 Teilen-Funktion für eine Comicdatei

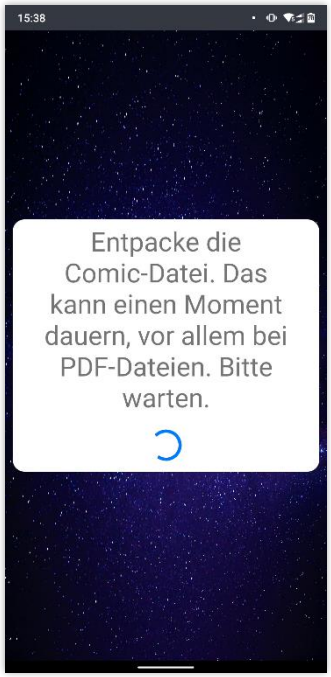

Abbildung 23 Wartemeldung in der *Reader-Activity* beim Entpacken einer Comicsdatei zum Anzeigen

Abbildung 24 Popup-Menü als Overlay der *Reader-Activity*

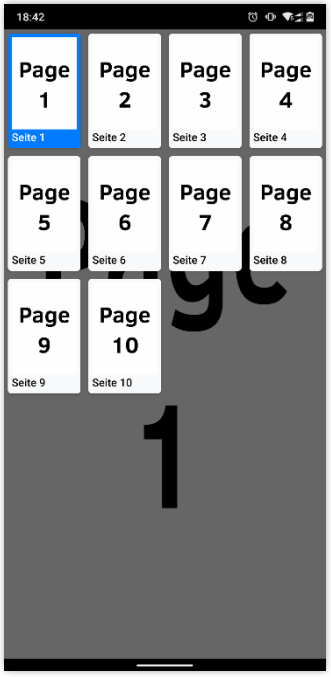

Abbildung 25 Seitenübersicht in der *Reader-Activity*

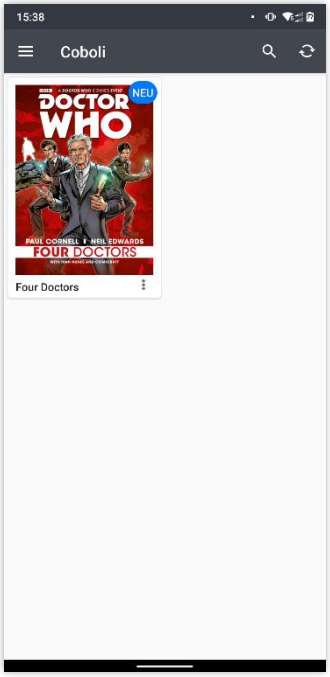

Abbildung 26 Leseliste auf Basis des *Issues-Fragment*

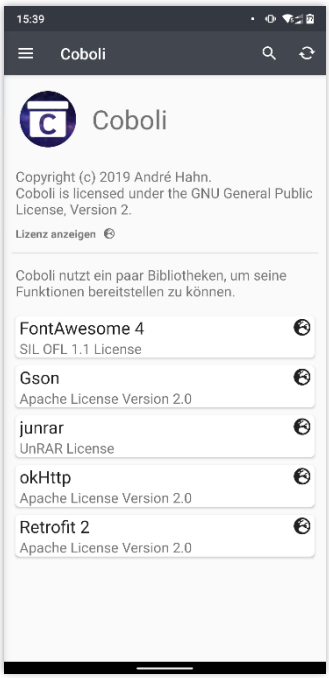

Abbildung 27 *About-Fragment* mit den Informationen zur App und den verwendeten Bibliotheken

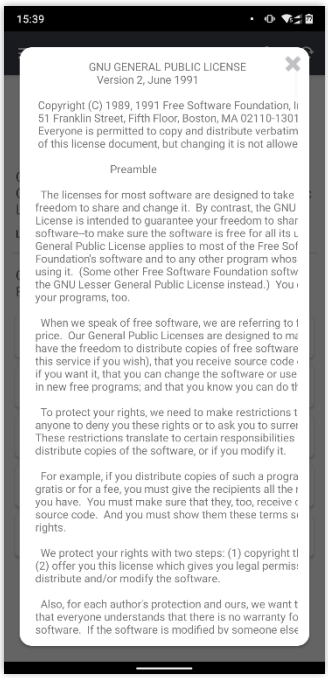

Abbildung 28 Lizenztext als Overlay im *About-Fragment*

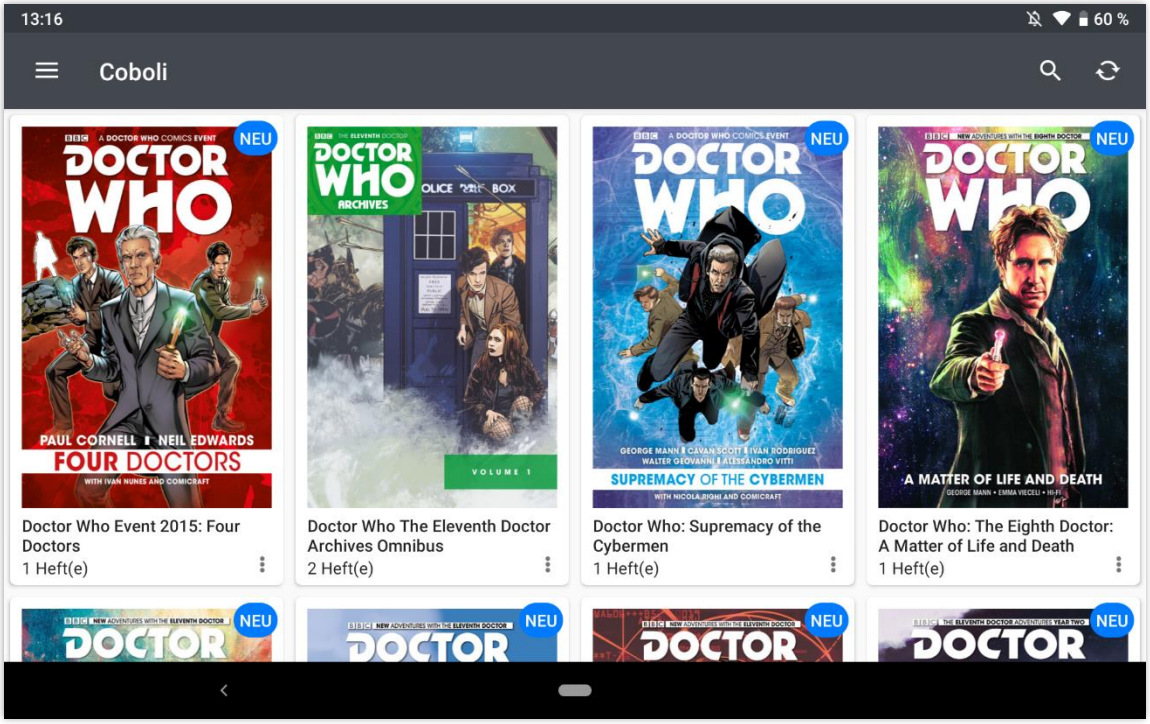

Abbildung 29 Variation eines Layouts zur Nutzung im Querformat auf dem Nexus 7

Eidesstattliche Erklärung

Ich versichere an Eides Statt, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, den 28. Oktober 2019

Unterschrift

(André Hahn)

TH Köln
Gustav-Heinemann-Ufer 54
50968 Köln
www.th-koeln.de

TH Köln
Campus Gummersbach
Steinmüllerallee 1
51643 Gummersbach

Technology
Arts Sciences
TH Köln