Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplom Nr. 3717

Entwicklung einer App zur Leseverfolgung auf Mobilgeräten

Victor Riempp

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Albrecht Schmidt
Betreuer/in:	Tilman Dingler
Beginn am:	2. März 2015
Beendet am:	1. September 2015
CR-Nummer:	H.4.m

Kurzfassung

Lesen fördert unsere sprachlichen und kognitiven Fähigkeiten und ist ein wichtiger Bestandteil der modernen Gesellschaft. Dabei hat sich die Art und Weise wie wir lesen durch die allgemeine Verfügbarkeit digitaler Geräte über die letzten 20 Jahre massiv verändert.

Ziel dieser Arbeit war es ein System zu konzipieren, welches den Benutzer dabei unterstützt den Überblick darüber zu bewahren, was er alles in digitaler Form gelesen hat und ihn dabei zu unterstützen es auch im Gedächtnis zu behalten.

Das konzipierte System besteht aus zwei Komponenten: Dem Reading Tracker und dem Reading Archive. Der Reading Tracker ist eine Android App, welche alle vom Benutzer gelesenen Texte an das Reading Archive überträgt. Das Reading Archive bewahrt diese auf und sorgt dafür, dass der Benutzer diese durchsuchen kann. Zusätzlich werden dem Benutzer verschiedene Metriken über sein Leseverhalten angezeigt. Das Reading Archive ist dabei eine in Django realisierte Webseite, die in der Android-App eingebunden wird.

Die Funktionalität und Wirkung dieses Systems wurde anschließend durch eine Benutzerstudie überprüft und deren Ergebnisse diskutiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Motivation

In den letzten 20 Jahren sind Computer in unserer Gesellschaft allgegenwärtig geworden. 87% aller deutschen Haushalte besitzen heutzutage einen Computer und sogar 93% aller Haushalte besitzen mindestens ein Handy [DES14]. Dabei sind 82% aller Handys Smartphones [Bri14].

Durch diese Allgegenwart des Digitalen haben sich auch unsere Gewohnheiten stark verändert. So lesen wir immer mehr auf digitalen Geräten, seien es Computer, E-Book-Reader oder Smartphones. Dabei verwenden wir auch immer mehr Zeit darauf, Texte zu überfliegen und beim Lesen im Text hin und her zu springen. Gleichzeitig lesen wir immer weniger linear und konzentriert den Text von Anfang bis Ende [Liu05].

Allerdings nutzen wir die Möglichkeiten dieser Geräte kaum aus, sondern simulieren häufig lediglich das Lesen eines Buches. So, wie wir es von unseren Büchern gewohnt sind, werden auch die Texte auf den digitalen Geräten in Seiten unterteilt, die wir umblättern müssen.

Es ist inzwischen in den meisten Apps und Programmen möglich, ihren Inhalt zu durchsuchen. Wer allerdings nicht mehr weiß, auf welchen Geräten und in welchen Apps er den gesuchten Text gelesen hat, muss nun jedes Gerät und jede App nach diesem durchsuchen.

Ziel dieser Arbeit war es ein System zu entwickeln, welches alle am Smartphone oder Tablett gelesenen Texte archiviert, es dem Benutzer ermöglicht diese archivierten Texte zu durchsuchen und wieder anzuzeigen. Zusätzlich soll das Gelesene quantifiziert werden und eine Zusammenfassung generiert und angezeigt werden können.

Abschließend wurde mit Hilfe einer Benutzerstudie untersucht, welche Effekte die Verfügbarkeit eines solchen Archivs auf die Benutzer hatte und die Ergebnisse dieser Studie besprochen.

1.2. Aufbau dieser Arbeit

Diese Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen und Verwandte Arbeiten: Zunächst wird auf die verwandten Arbeiten eingegangen und anschließend werden die technischen Grundlagen erläutert.

Kapitel 3 – Systemarchitektur: Am Anfang dieses Kapitels wird ein grober Überblick über das implementierte System gegeben. Danach wird näher auf die verwendeten Frameworks eingegangen und am Schluss wird die Implementierung des Systems beschrieben.

Kapitel 4 – Studie: Hier wird zunächst der Aufbau der Studie beschrieben. Anschließend wird näher auf die Teilnehmer der Studie eingegangen und die Durchführung beschrieben. Das Kapitel schließt mit den Ergebnissen der Studie.

Kapitel 5 – Diskussion: In diesem Kapitel wird auf die Ergebnisse der Studie eingegangen und ihre Bedeutung diskutiert.

Kapitel 6 – Ausblick: Abschließend wird darauf eingegangen,welche Entwicklungsprozesse aufbauend auf dieser Arbeit in diesem Themengebiet initiiert werden können.

2. Grundlagen und Verwandte Arbeiten

2.1. Gedächtnis

Schon im 19. Jahrhundert wurde vermutet, dass das menschliche Gedächtnis ein System aus mehren unterschiedlichen Komponenten ist, die jeweils einen Teilaspekt bearbeiten. So wurde bereits 1890 vermutet, dass erlerntes Wissen anders abgespeichert wird, als erlernte Aktionen und Handlungen [Jam90].

2.1.1. Atkinson-Shiffrin-Modell

Das Atkinson-Shiffrin-Modell wurde 1968 von Richard Atkinson und Richard Shiffrin beschrieben. In ihm werden einzelnen Gedächtniskomponenten definiert und in Beziehung zueinander gestellt [AS68]. Die drei in diesem Modell beschriebenen Komponenten sind:

- **Sensorisches Gedächtnis:** Speichert Sinneseindrücke für einige Sekundenbruchteile ab, bevor sie von neuen Eindrücken verdrängt werden.
- **Kurzzeitgedächtnis:** Kann 7±2 verschiedene Informationsblöcken für mehre Sekunden in unserem Bewusstsein bereit halten.
- **Langzeitgedächtnis:** Dauerhaftes Speichersystem für unsere Erinnerungen.

Wenn die Sinneseindrücke unsere Aufmerksamkeit erregen wandern sie vom Sensorischen Gedächtnis in unser Kurzzeitgedächtnis. Dort können nur etwa 7±2 Elemente gleichzeitig vorgehalten werden. Diese Limitierung des Kurzzeitgedächtnisses wurde schon 1956 durch George A. Miller beschrieben [Mil56].

Informationen können auch aus dem Langzeitgedächtnis abgerufen werden, woraufhin sie ins Kurzzeitgedächtnis gelangen. Dort verbleiben sie nur, wenn sie weiter aktiv Wiederholt werden, entweder durch ausgeführte Aktionen, wiederholten sensorischen Stimulus oder durch aktives Nachdenken. Vom Kurzzeitgedächtnis gelangen Informationen ins Langzeitgedächtnis in dem sie oft und lange im Kurzzeitgedächtnis präsent sind. Im Langzeitgedächtnis werden sie nun dauerhaft abgespeichert und können erneut abgerufen werden. [AS68].

Auch wenn das Atkinson-Shiffrin-Modell ein guter Ansatz ist um die Grundlagen des menschlichen Gedächtnisses zu beschreiben, hat sich schnell gezeigt, dass weitere Unterscheidungen

nötig sind und komplexere Abläufe innerhalb und zwischen den einzelnen Komponenten stattfinden.

2.1.2. Langzeitgedächtnis

Das Langzeitgedächtnis kann man aufteilen in das explizite und das implizite Gedächtnis. Das explizite Gedächtnis umfasst alle Prozesse die in das Bewusstsein des Individuums treten. Dagegen finden alle Prozesse im impliziten Gedächtnis unbewusst statt. [GS85].

Prozedurales Gedächtnis

Das Prozedurale Gedächtnis ist eine Form des impliziten Gedächtnisses und speichert Handlungsabläufe und Fertigkeiten ab, die wir anschließend, nachdem wir die dafür nötigen Bewegungsabläufe bewusst gelernt haben, ohne aktiv darüber nachzudenken, ausführen können wie z.B. Fahrradfahren oder Schwimmen.

Deklaratives Gedächtnis

Das Deklarative Gedächtnis ist eine Form des expliziten Gedächtnisses. Es speichert Tatsachen und Ereignisse ab, die wir bewusst wiedergeben können. Dabei wird zwischen dem episodischen und dem semantischen Gedächtnis unterschieden [Tul72].

Das **semantischen Gedächtnis** enthält unser Weltwissen und von uns selbst unabhängige allgemeine Fakten. Diese Fakten können wir durch Wiederholung lernen, z.B. auswendig Lernen von Vokabeln oder Texten

Das **episodische Gedächtnis** hingegen umfasst Erinnerungen aus dem eigenen Leben in Form von Episoden aus Ereignisketten.

2.1.3. Arbeitsgedächtnismodell

Das Arbeitsgedächtnis (engl. working memory) ist eine Präzisierung des Kurzzeitgedächtnisses, welche 1974 das erste mal von Badeley und Hitch beschrieben wurde [BH74]. In ihm wird das im Atkinson-Shiffrin-Modell definierte Kurzzeitgedächtnis in mehre komplexe Untereinheiten aufgeteilt. Im Jahr 2000 wurde es um eine weitere Komponente, den episodischen Puffer, erweitert [Bad00].

- **Zentrale Exekutive:** Koordiniert die anderen Komponenten und kann die Aufmerksamkeit aufteilen, fokussieren oder umlenken. Sie dient zusätzlich als Schnittstelle zum Langzeitgedächtnis.

- **Phonologische Schleife:** Ist ein begrenzter Kurzzeitspeicher für Lautformen (Phoneme). Auditiv aufgenommene Lautformen werden direkt in diesem Speicher abgelegt, jedoch müssen geschriebene Informationen erst von ihrer Textfrom in eine Lautform übertragen werden, bevor sie abgespeichert werden. Die gespeicherten Lautformen können durch aktives inneres Wiederholen am verblassen gehindert werden.
- **Visuell-räumlicher Notizblock:** Ist ein begrenzter Kurzzeitspeicher für visuelle und räumliche Informationen. Deren Verarbeitung findet jedoch getrennt statt.
- **Episodischer Puffer:** Ist ein begrenzter Kurzzeitspeicher in dem sowohl Lautformen als auch visuelle oder räumliche Informationen in Form von zusammenhängenden Episoden abgespeichert werden können.

Diese Aufteilung ergab sich aus den Ergebnissen verschiedener Gedächtniseffekte, die sich ansonsten nicht hätten erklären lassen [Bad00].

2.2. Lesen

Der Sinn des Lesens ist immer den Inhalt eines Textes verstehen zu können. Dabei stellt das Lesen einen komplexen Prozess dar, der erlernt werden muss. Voraussetzung für das Lesen lernen ist dabei gute Kenntnisse der Sprache und ihrer Grammatik, damit das gelesene auch verstanden werden kann. Darüberhinaus sind Wahrnehmungsprozesse und kognitive Prozesse nötig, um Buchstaben in Wörter umzuwandeln und ihren Sinn zu erfassen.

Obwohl wir das Gefühl haben, dass sich unsere Augen beim Lesen nur langsam in Leserichtung bewegen, führen diese tatsächlich viele kleine Sprünge aus. Dabei folgt die Hauptrichtung dieser, nur 10-20 Millisekunden dauernden Sprünge, der erlernten Leserichtung. Zwischen den Sprüngen fokussiert sich das Auge für 200-250 Millisekunden auf den Text. Dieser Zeitraum wird genutzt, um den Text visuell aufzunehmen. Dabei werden etwa 4 Zeichen links und 15 Zeichen rechts der Fixierung erfasst [EK00].

Das Fokussieren der Wörter findet nicht gleichmäßig statt, sondern folgt gewissen Regeln, die im EZ-Reader Modell wie folgt definiert sind [RPFR98]:

- Seltene Wörter werden länger fixiert
- Vorhersagbare Wörter werden kürzer fixiert.
- Kurze oder sehr häufige Wörter werden zum Teil nicht fixiert.
- Wörter, die auf ein seltenes Wort folgen, werden länger fixiert.

Das Erkennen und Verarbeiten eines Wortes kann dabei auch von einem vorhergehenden Wort positiv oder negativ beeinflusst werden. Diese Beeinflussung wird Priming genannt und ist nicht auf das Lesen limitiert [BEA09].

2.2.1. Auswirkungen des Lesens

Lesen wirkt sich positiv auf den Lesenden aus, denn es hilft dabei das Vokabular zu vergrößern und die kognitiven Fähigkeiten zu verbessern. Der positive Effekt des Lesens wird verstärkt durch frühzeitigen Beginn und Häufigkeit. Allerdings kommt es dabei auch zum einem Mathäus-Effekt (benannt nach einem Bibelzitat aus Matthäus 25.29): Ungeübte Leser tendieren dazu immer weniger zu lesen, während geübte Leser dazu tendieren mehr zu lesen [SC98].

2.2.2. Veränderungen der Lesegewohnheiten

In den letzten 15 Jahren haben sich unsere Lesegewohnheiten massiv verändert. Zunächst durch die allgemeine Verbreitung von Computern und danach von Smartphones und E-Book-Readern. Deshalb lesen wir heutzutage immer mehr an digitalen Geräten und weniger analog auf Papier.

Auch die Art unseres Lesens hat sich verändert: Auf digitalen Geräten verwenden wir mehr Zeit darauf Texte zu überfliegen, nach Stichworten zu suchen und beim lesen im Text hin und her zu springen. Das dauerhafte konzentrierte, lineare Lesen eines Textes hat deutlich nachgelassen und damit auch die durchgehend fokussierte Aufmerksamkeit.

Die analoge Papierform wird allerdings für das dauerhafte und konzentrierte Lesen bevorzugt [Liu05].

Im studentischen Umfeld findet das Lesen auf digitalen Geräten eher als Ergänzung statt. Grund hierfür ist die zumindest derzeit noch mangelhafte technische Unterstützung für wichtige Prozesse, wie das Überfliegen eines Textes (bei E-Books) und des Annotieren eines Textes [TLH+11].

2.3. Wortwolke

Eine Wortwolke (auch Schlagwortwolke, im Englischen Word Cloud oder Tag Cloud) ist eine spezielle Visualisierung von Schlagworten, die flächig angezeigt und unterschiedlich gewichtet sind. Die Gewichtung wird durch verschiedene Textgrößen und/oder Farben visuell dargestellt [Fei09].

Die genaue Herkunft von Wortwolke ist Unbekannt, ein erster Nachweis findet sich auf dem Cover des Buches "Tausend Plateaus. Kapitalismus und Schizophrenie" von Gilles Deleuze, Felix Guattari [DG92].

In Zuge des am Anfang dieses Jahrtausends aufkommenden Web 2.0 wurden Wortwolken sowohl als Visualisierung als auch als Navigationselement populär [Fei09].

2.4. Java

Java ist eine objektorientierte Programmiersprache, welche von Sun Myrosystems entwickelt wurde und heute zu Oracle gehört. Sie ist seit 2006 Open Source. Weiterhin existiert allerdings eine proprietäre Implementation von Oracle.

Programme in Java werden nicht direkt in Maschinencode überführt, sondern zuerst in ein Zwischenformat, welcher Bytecode genannt wird. Dieser Bytecode wird erst zur Laufzeit durch eine Laufzeitumgebung, die Java Virtual Machine (JVM), in Maschinencode umgewandelt und ausgeführt.

Durch diese Zwischenkompilierung können mit Java geschriebene Programme weitgehend Plattform unabhängig realisiert werden, solang für die entsprechenden Plattformen eine JVM Implementierung existiert [Ull04].

2.5. Python

Python ist eine interpretierte Programmiersprache mit dynamischer Typisierung und unterstützt verschiedene Programmierparadigmen, unter anderem funktionale und objektorientierte Programmierung.

Python wurde unter einer eigenen Open Source Lizenz veröffentlicht und wird von der Python Software Foundation beständig weiter entwickelt, welche auch die Referenzimplementierung CPython pflegt. CPython ist in C geschrieben und steht für alle gängigen Betriebssysteme zur Verfügung. Es existieren jedoch viele weitere Implementierungen, welche in anderen Programmiersprachen geschrieben wurden.

Python 3 erschien 2008 und brachte große Änderungen der Sprache mit sich, die um einige „Altlasten“ befreit wurde. Dadurch ist jedoch Python 2 Code nicht mehr in Python 3 lauffähig, was anfangs zum hauptsächlichen Gebrauch von Python 2 führte, da benötigte und beliebte Frameworks nicht in Python 3 verfügbar waren. Inzwischen sind jedoch die meisten Frameworks auch in Python 3 verfügbar [Pyt15].

2.6. JavaScript

Ende 1995 wurde der Browser Netscape in Version 2 veröffentlicht und enthielt eine Programmiersprache, mit der HTML-Seiten beeinflusst werden konnten. Zu Beginn hieß diese Sprache noch LiveScript, wurde jedoch bald darauf schon in JavaScript umbenannt. Die Syntax von JavaScript orientierte sich dabei stark an Java [Wen06].

Im Jahr 1997 wurde JavaScript schließlich als ECMA-262 unter dem Namen ECMAScript standardisiert [ECM97]. Dieser Standard wurde über die Jahre hinweg weiter entwickelt und liegt derzeit in Version 6 vor [ECM15].

JavaScript hat sich über die Jahre stark verbreitet und ist in jedem modernen Browser verfügbar. Zusätzlich wird JavaScript auch bei node.js[1] zum schreiben von Serverprogramme genutzt.

Heutzutage ist JavaScript eine der meistgenutzten Programmiersprachen und wird auch als Lingua franca des Web bezeichnet. Auf GitHub[2] ist JavaScript sogar die beliebteste Programmiersprache [Neu15].

2.7. JSON

Die JavaScript Object Notation (JSON) wurde erstmals von Crockford [Cro06] formal beschrieben. JSON hat sich als leichtgewichtiges Datenaustauschformat etabliert. Sie entstand aus der Serialisierungsnotation von JavaScript Objekten, heutzutage wird sie jedoch auch unabhängig von JavaScript verwendet [Cro08].

```
{
      "vorname": "Victor",
      "nachname": "Riempp",
      "alter": 28,
      "emails": [
            {
                  "art": "privat"
                  "email": "victor.riempp@gmail.com
            },
            {
                  "art": "uni"
                  "email": "riemppvr@studi.informatik.uni-stuttgart.de"
            }
      ]
}
```

Listing 2.1: JSON Beispiel

[1] https://nodejs.org/
[2] https://github.com/

3. Systemarchitektur

Dieses Kapitel spezifiziert thematisch zunächst die Zielvorgaben dieses Projekts und erläutert die Auswahl der Programmiersprachen, Protokolle und Frameworks. Anschließend werden die benutzten Frameworks vorgestellt und zum Abschluss dieses Kapitels wird die Umsetzung genauer beleuchtet.

3.1. Konzept

Um untersuchen zu können wie sich automatisch generierte Zusammenfassungen und die Möglichkeit Inhalte leicht wiederfinden zu können auf das Gedächtnis wie auch das Leseverhalten auswirken, ist die Erstellung eines Lesearchivs nötig, das die entsprechenden Daten bündelt und den Zugriff darauf protokolliert.

Dabei waren folgende Problemstellungen zu beachten:

- Wie soll so ein Archiv aufgebaut sein, was soll gespeichert und berechnet werden?
- Wie kommt man an die Texte, die im Archiv gespeichert werden sollen heran?
- Wie sollen die Daten zwischen dem Archiv und der Quelle ausgetauscht werden?

Das Archiv soll in der Lage sein, alle gelesenen Texte eines Benutzers abzuspeichern und zur Verfügung zu stellen. Anschließend soll es auch möglich sein, die gespeicherten Texte anzuzeigen, zu durchsuchen und automatische Zusammenfassungen zu erstellen.

Um Daten verschiedener Geräte erfassen zu können, welche man später gut für die Studie auswerten kann, war es nötig einen zentralen Server zu erstellen, auf dem die Texte abgespeichert werden. Daher lag die Entscheidung nahe, das Archiv als Webseite zu implementieren die sowohl mit Mobilgeräten als auch mit Laptops und PCs benutzbar ist.

Zur Erstellung einer solchen Webseite stehen heutzutage eine große Anzahl unterschiedlicher Programmiersprachen und Frameworks zur Auswahl. Enger in Betracht gezogen wurden

Java mit JavaServer Faces[1], JavaScript mit dem derzeit sehr beliebten Framework Node.js[2] und Python mit Django.

Die Wahl viel dabei schließlich auf Python 3 und Django, jeweils in der neusten derzeit stabilen Version. Grund hierfür war, dass Python 3 und Django flexibler eingeschätzt wurden als Java mit JavaServer Faces, gleichzeitig jedoch mehr der benötigten Funktionalität out of the box bereitstellten, als dies bei Node.js der Fall gewesen wäre.

Um die Texte des Benutzers erfassen zu können, müssen diese zunächst an ihren jeweiligen Quellen abgeschöpft werden. Im Rahmen dieser Arbeit beschränkt sich die Erfassung von Texten auf Android, da Android das derzeit am meisten verbreitetste Betriebssystem für Mobilgeräte ist [Kö14].

Ziel war es hierbei eine einfach zu installierende App zu entwickeln, die auf möglichst vielen Geräten lauffähig ist. Die App muss dafür in der Lage sein, dass was der Benutzer auf dem Bildschirm sieht auszulesen und zu verarbeiten. Hierfür gab es mehre Optionen:

1. Ausnutzung der Accessibility-API, welche ursprünglich dafür gedacht war, die Barrierefreiheit für Menschen mit Beeinträchtigungen zu erhöhen.
2. Das Anfertigen von Screenshots mit anschließender Texterkennung (engl. Optical Character Recognition kurz OCR)
3. Ansetzen auf niedrigerer Systemebene in Android, entweder durch das Nutzen eine eigene Android-Variante oder durch das Eindringen in das System durch Ausnutzen von Sicherheitslücken.

Die zweite Option wäre jedoch, durch Einschränkungen im Android-Framework, nur möglich, wenn der Benutzer root-Rechte auf seinem Smartphone gehabt hätte und diese der App zur Verfügung stellen würde [Goo15b].

Somit erfüllte nur Option erste die Fähigkeit die App einfach installieren zu können.

Als Datenaustauschformat zwischen App und Archiv wurde JSON gewählt, da es im Vergleich zu XML deutlich weniger Bandbreite benötigt, was gerade bei Mobilgeräten mit schlechtem Empfang oder beschränktem Volumen vorteilhaft ist. Die Verwendung eigenen oder exotischeren Datenformats hätte dem gegenüber nur noch wenig Bandbreite gespart, jedoch den Implementierungsaufwand beträchtlich gesteigert.

[1] http://javaserverfaces.java.net/
[2] http://nodejs.org/

3.2. Genutzte Frameworks

3.2.1. Android

Android ist ein Betriebssystem für mobile Geräte, welches hauptsächlich von Google entwickelt wurde. Am 5.November 2007 fand die Ankündigung des Android Systems statt [Rea07], im darauf folgenden September erfolgte die Veröffentlichung des Android Software Development Kit 1.0 [Mor08]. Somit war es nun möglich eigene Applikationen (Apps) für Android zu entwickeln. Das erste Android Handy kam etwa einen Monat später auf den Markt.

Inzwischen ist Android 5.1. verfügbar [Bur15] und Android läuft auf fast 85% aller Smartphones [Kö14] und ist damit das führende mobile Betriebssystem.

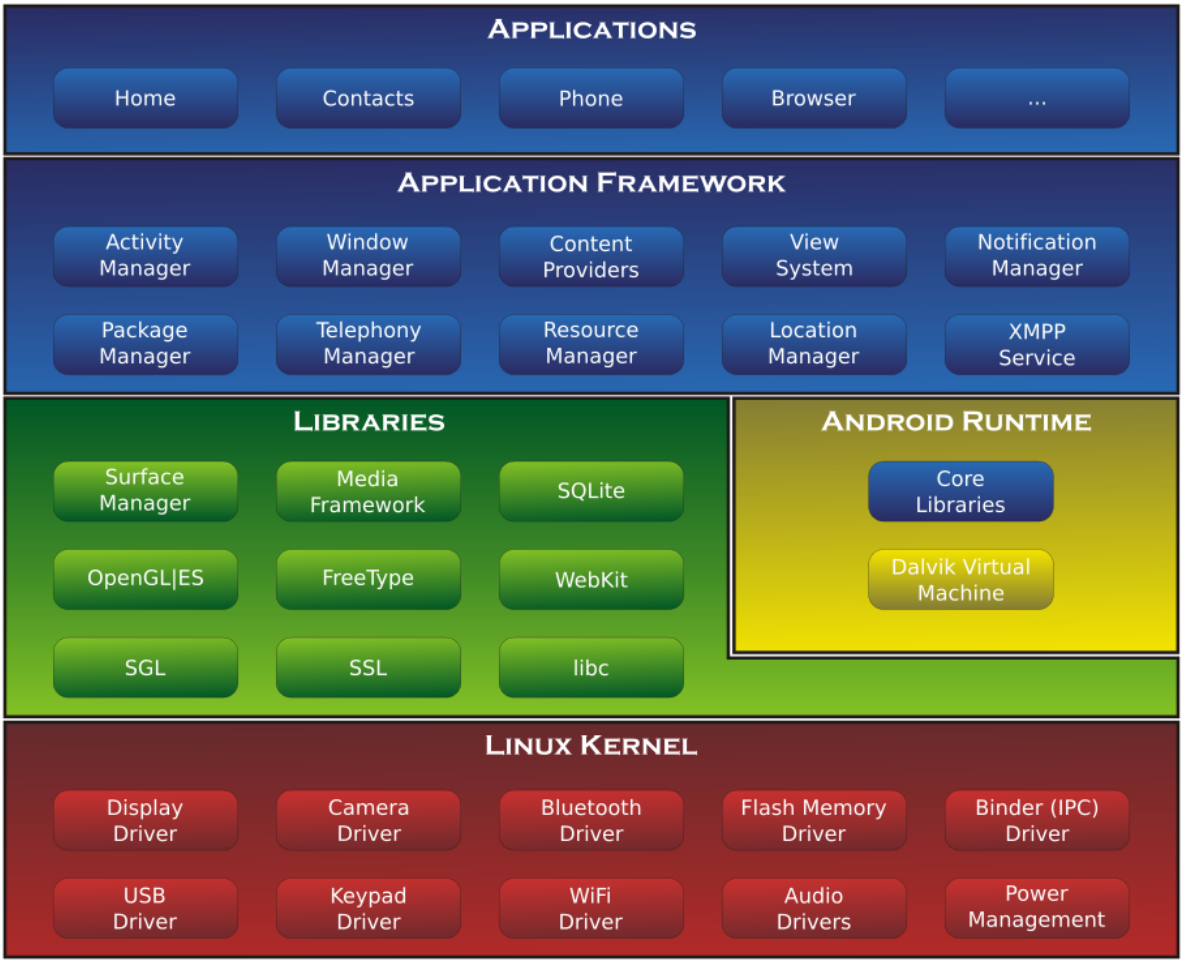

Abbildung 3.1.: Android Architektur [Smi12]

Android benutzt als Basis einen leicht modifizierten Linux Kernel. Darauf wurde eine Hardware-Abstraktionsschicht aufgebaut, welche den leichten Zugriff auf Hardware, wie zum Beispiel die Kamera, ermöglicht. Weiterhin werden mehre Bibliotheken zur Verfügung gestellt, die vom Application Framework und der Android Runtime benötigt werden, meist jedoch auch in den Android Applikationen (kurz Apps) zur Verfügung stehen.

Apps werden in Java programmiert, können jedoch nicht in einer normalen Java Virtual Machine (JVM) ausgeführt werden, da sie auf das Android Application Framework aufbauen, welches nicht für die JVM zur Verfügung steht. Auch eine Portierung von Java Programmen ist kaum möglich, da für Android Apps verschiedene Einschränkungen gelten und nur die von dem Android Application Framework bereitgestellten Funktionen zum Erstellen einer Benutzeroberfläche genutzt werden können [Goo15c].

Apps werden in Bytecode kompiliert und zusammen mit allen benötigten Resourcen (Bildern, Sounds usw.) sowie der zur Ausführung benötigten Metadaten in eine, Android Application Package (APK) genannte, Archivedatei verpackt. Diese APK-Datei wird genutzt um Apps zu verteilen und zu installieren [Goo15a].

Bis einschließlich Android 4.4 wurden Apps in der Dalvik Virtual Machine (Dalvik VM) ausgeführt. Diese ist ähnlich wie die JVM eine Laufzeitumgebung, welche den Bytecode der Apps zur Laufzeit in Maschinencode übersetzt und ausführt.

Seit Android 5.0 wird die Android Runtime (ART) für die Ausführung von Apps genutzt, welche den Bytecode schon bei der Installation in Maschinencode übersetzt. Dadurch können Apps schneller gestartet werden. Sie benötigen jedoch mehr Speicherplatz und die Installation braucht etwas länger [Goo15c].

Jede App läuft bei Android in ihrer eigen Virtuellen Maschine, besitzt auf Linux Ebene ihren eigenen Benutzer und läuft in einem separaten Linux Prozess. Ziel ist hierbei, die starke Abkapselung zwischen den einzelnen Apps, aber auch vom Betriebssystem. Unter gewissen Umständen ist eine Aufweichung dieser Trennung bei Apps des selben Herstellers möglich.

Die Komponenten aus der jede App besteht sind Activities, Services, Content Providers und Broadcast Receivers. Welche Rolle ihnen jeweils zukommt, wird im folgenden noch genauer beleuchtet [Goo15a].

Die folgenden Abschnitte beziehen sich, wenn nicht gesondert erwähnt auf [Goo15b].

Android Manifest

Das Android Manifest ist eine XML Datei (AndroidManifest.xml), in der definiert wird, welche Komponenten die App enthält, wie sie heißt, alle Berechtigungen, die sie benötigt,

welche Android-Version für sie vorausgesetzt wird und vieles mehr. Damit ist sie die zentrale Schnittstelle, welche die einzelnen Komponenten der App miteinander und mit dem Betriebssystem verbindet [Goo15a].

Activity

Eine Activity repräsentiert jeweils eine Bildschirmansicht innerhalb der App. In der Activity wird alles geladen, was zur Anzeige auf dem Bildschirm nötig ist. Des Weiteren wird festgelegt welche Dinge auf dem Bildschirm zu sehen sind und deren Darstellungsform. Dabei greift die Activity meist auf in XML gespeicherte Layouts zurück, die ausschließlich mit Inhalt und Funktionalität versehen werden. Letztlich ist die Activity auch für das Entgegennehmen der Benutzerinteraktion zuständig. Ihre Aufgabe besteht in der Initiierung des vom Benutzer gewünschten Vorgangs.

Technisch gesehen ist eine Activity eine Klasse, die von **android.app.Activity** erbt und im Android Manifest eingetragen wurde. Wenn die Activity gestartet wird durchläuft sie einen bestimmten Lebenszyklus (Siehe Abbildung 3.2).

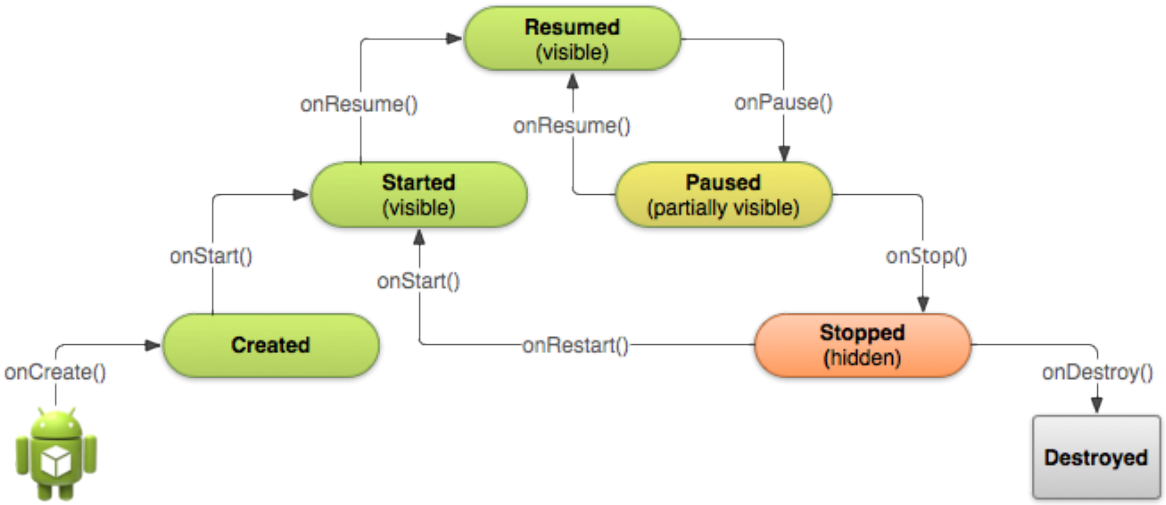

Abbildung 3.2.: Activity Lebenszyklus [Goo15b]

Wenn das Android Framework den Befehl bekommt, eine Activity das erste mal anzuzeigen, so wird zuerst ein Objekt der Klasse instantiiert und anschließend ihre onCreate()-Methode aufgerufen. Sobald diese abgearbeitet ist, wird die onStart()-Methode gestartet. Sobald diese abgearbeitet wurde, ist die Activity für den Benutzer sichtbar, er kann jedoch noch nicht mit ihr interagieren. Nun wird die onResume()-Methode ausgeführt, nach deren Abschluss dem Benutzer die Interaktion mit der Activity möglich ist.

Wenn die Activity den Fokus des Benutzers verlieren soll wird zuerst die onPause()-Methode aufgerufen. In diesem Zustand ist die Activity noch zum Teil sichtbar. Dazu kann es kommen,

wenn ein Dialog angezeigt wird, der aber nicht den kompletten Bildschirm bedeckt. Nach beenden des Dialogs kann die Activity mit onResume() wieder den Fokus des Benutzers bekommen und vollständig sichtbar sein. Alternativ kann es auch vorkommen, dass eine andere Activity gestartet werden soll (auch ohne dass ein Dialog angezeigt wird). Dann wird die onStop()-Methode aufgerufen, die Activity wird ausgeblendet und eine andere Activity wird angezeigt.

In diesem Zustand kann die Activity längere Zeit verharren, während andere Activities (auch von anderen Apps) im Fordergrund sind. Soll die Activity nochmal angezeigt werden, so wird wieder die onStart()-Methode und anschließend die onResume()-Methode aufgerufen.

Wenn der Arbeitsspeicher des Smartphones voll läuft, so kann das Framework die Instanz der Activity aus dem Speicher entfernen. Zunächst wird jedoch noch die onDestroy()-Methode aufgerufen. Selbiges passiert auch, wenn sich eine App selbst vollständig beendet oder von einem Taskmanager beendet wird.

Service

Ein Service läuft im Hintergrund und führt längere Operationen aus, die ansonsten die Oberfläche blockieren würden. Dazu gehören komplizierte Berechnungen ebenso, wie Senden und Empfangen von Daten über das Internet sowie andere Schnittstellen. Somit alle Prozesse, die eine Laufzeit von ein paar Millisekunden überschreiten.

Technisch gesehen ist ein Service eine Klasse, die von **android.app.Service** erbt und im Android Manifest eingetragen wurde. Üblicherweise werden Services von einer Activity gestartet, es gibt allerdings auch Services, die direkt vom Android Framework gestartet werden. Weiterhin stellt die Android API mehrere Klassen bereit, welche von der Service-Klasse erben und häufig benötigte Funktionalitäten zur Verfügung stellen.

Accessibility Service

Ein Accessibility Service ist ein Service, der speziell dafür vorgesehen ist, mehr Barrierefreiheit auf Android Systemen zu erzeugen. Dies macht es möglich die Benutzbarkeit von Android auch für Menschen mit unterschiedlichen Beeinträchtigungen zu gewährleisten. Ein Beispiel hierfür ist Google TalkBack, welches die textuelle Inhalte anderer Apps vorliest und Sprachkommandos entgegen nimmt. Damit die Fähigkeit auf Inhalte fremder Apps zuzugreifen nicht Missbraucht werden kann, müssen Apps, die einen Accessibility Services implementieren im Android Manifest dafür eine Permission anfordern. Diese Permission muss vom Benutzer bei der Installation bewilligt werden. Als zusätzlich Sicherheitsmaßname muss jeder Accessibility Service in den Einstellungen, unter Bedienungshilfen manuell freigeschaltet werden.

Ist der Accessibility Service dort freigeschaltet, so wird er vom Android Framework jedes mal darüber informiert, wenn ein Accessibility Event auftritt. Diese informieren den Accessibility Service darüber, welche Veränderungen gerade auf dem Bildschirm stattfinden oder welche Aktionen vom Benutzer ausgeführt wurden, damit dieser entsprechendes Feedback an den Benutzer geben kann.

Content Providers

Ein Content Provider dient dazu, den Zugriff auf Daten zu bekommen, die von Android zentral verwaltet werden. Hierzu zählen unter anderem das Wörterbuch des Benutzers, seine Kalendereinträge, seine Kontakte sowie der Zugriff auf Dateien. Bei Bedarf , können auch eigene Content Provider im Android Framework registriert werden.

Broadcast Receivers

Der Broadcast Receiver nimmt Ankündigungen des Android Framework entgegen. Er kann je nach Ankündigung eine Benachrichtigung in der Statusbar anzeigen oder einen Service starten. Eine Ankündigung kann in diesem Fall die Verfügbarkeit von Internet oder WLAN sein oder auch die Aktivierung des Displays. Das Senden und Empfangen von eigenen Ankündigungen ist durchaus auch möglich.

3.2.2. Django

Django ist ein in Python geschriebenes Webframework, welches zum Ziel hat den Entwickler dabei zu unterstützen möglichst schnell und unkompliziert eine voll funktionsfähige Webseite zu erstellen. Django wurde ursprünglich von Lawrence Journal-World[3] entwickelt und wurde 2005 als Open Source veröffentlicht. Heute wird Django von der Django Software Foundation weiterentwickelt und gewartet [Dja15b].

Im folgenden bezieht sich der Text, wenn nicht anders bezeichnet, auf die offizielle Dokumentation für Version 1.8 [Dja15a].

Django folgt einem Model-View-Controll (MVC) ähnlichen Entwurfsmuster und wird von den Entwicklern selbst als Model-Template-View (MTV) bezeichnet. Jedoch stellt der von Django bereitgestellte URL Dispatcher einen Controller im sinne des MVC dar.

Als ein Projekt wird bei Django ein Python Paket betrachtet, das alle Einstellungen für eine Django Instanz enthält. Ein Projekt kann dabei aus mehren Apps bestehen. Weiterhin können

[3] http://www.ljworld.com/

auch mehre Projekte gleichzeitig auf einem Server betrieben werden, jedoch müssen sie alle unter verschiedenen URLs zu erreichen sein.

Eine (Django-)App ist hierbei ein Python Paket, dass darüber hinaus bestimmte Anforderungen erfüllt. Eine App muss zumindest eine __init__.py enthalten, um als Python Paket zu gelten. Abgesehen davon muss die App nichts enthalten, wäre jedoch völlig ohne Funktion. Fast immer sind die Dateien admin.py, models.py, tests.py, urls.py und views.py enthalten. Es können jedoch beliebig viele weitere Dateien erstellt werden, die oben genannten unterscheiden sich jedoch insofern von diesen, als dass sie automatisch von Django nach Klassen oder Funktionen durchsucht werden, die mit den entsprechenden Bereichen verknüpft sind. Eine App kann dabei auch Teil mehrerer Projekte sein.

Wenn eine HTTP oder HTTPS Anfrage bei einem Django Projekt ankommt, wird von dem URL Dispatcher geprüft, welche View dafür zuständig ist. Hierfür liest der URL Dispatcher die urls.py des Projekts, in dem alle URLs dieses Projektes aufgeführt werden. Dabei können auch die urls.py der Apps des Projektes und die urls.py von Standartkomponenten eingetragen werden, welche dann auch durchsucht werden. Falls kein passender Eintrag gefunden wird, wird als Antwort eine 404-Fehlermeldung angezeigt. Sollte jedoch ein passender Eintrag gefunden werden, so verweist dieser auf die View, die dafür zuständig ist eine Antwort zu erzeugen.

Views

Eine View in Django kann entweder als Funktion oder als Klasse implementiert werden. Eine solche Klasse muss dabei von django.views.generic.View erben. Django stellt dabei noch weitere Klassen bereit, die häufig benötigte Funktionsweisen implementieren und leicht angepasst und erweitert werden können. Alle Views einer App befinden sich per Konvention in der jeweiligen views.py Datei.

Die View wird wie oben beschrieben vom URL Dispatcher aufgerufen und nimmt von ihm die Anfrage entgegen. Als Ergebnis kann die View eine Ausgabe in beliebigen Formaten erzeugen, entweder direkt, oder mit Hilfe eines Templates oder sie kann die Anfrage auf eine andere URL weiterleiten.

In der View wird alles Nötige berechnet, um eine Antwort anzuzeigen. Dazu können unter anderem verschiedenste Berechnungen, das Durchführen von Datenbankabfragen oder das Lesen von Dateien gehören.

Templates

Django Templates sind normale HTML-Dateien, welche um Variablen, Filter und Tags erweitert werden können.

Variablen:

```
<p>{{ foo }}</p>
```

Würde beim Übersetzen des Templates den aktuellen Inhalt von foo einfügen, wenn zum Beispiel foo das Wort Blau als String enthalten würde, dann wäre die Ausgabe folgende:

```
<p>Blau</p>
```

Filter:

```
<p>{{ foo|filtername }}</p>
```

Variablen können einen zusätzlichen Filter enthalten, der die Ausgabe der Variable verändert. In Django sind mehre Filter vordefiniert, es können jedoch auch leicht eigene Filter hinzugefügt werden. Filter können hierbei nur beeinflussen, welcher Inhalt in welcher Form ausgegeben wird.

Tags:

```
{% tag1 %}
```

Tags sind wesentlich komplexer. Sie können unter anderem den Inhalt der Ausgabe verändern, den Kontrollfluss manipulieren oder Schleifen realisieren.

Kommentare:

```
{# Ein Kommentar #}
```

Kommentare werden bei der Übersetzung des Templates schlicht entfernt.

Template Vererbung:

Ein Template kann mit hilfe des Tags **extends** auch von anderen Templates erben und dabei in dem Basistemplate mit **block** und **endblock** definierte Blöcke mit neuem Inhalt überschreiben.

```
<!DOCTYPE html>
<html>
      <head>
            <title>{% block title %}Mein Titel{% endblock %}</title>
      </head>
      <body>
            {# Nur ein Kommentar! Ich werde verschwinden! #}
            {% block content %}
            <h1>{{ text_to_show }}</h1>
            {% endblock %}
      </bod>
</html>
```

Listing 3.1: Basistemplate: base.html

```
{% extends "base.html" %}
{% block title %}Neuer Titel{% endblock %}
{% block content %}
      <h1>Jetzt wird ein anderer Text angezeigt!</h1>
{% endblock %}
```

Listing 3.2: Erbendes Template: extended.html

Die Ausgabe wäre in diesem Beispiel:

```
<!DOCTYPE html>
<html>
      <head>
            <title>Neuer Titel</title>
      </head>
      <body>
            <h1>Jetzt wird ein anderer Text angezeigt!</h1>
      </bod>
</html>
```

Listing 3.3: Ausgabe

Middleware

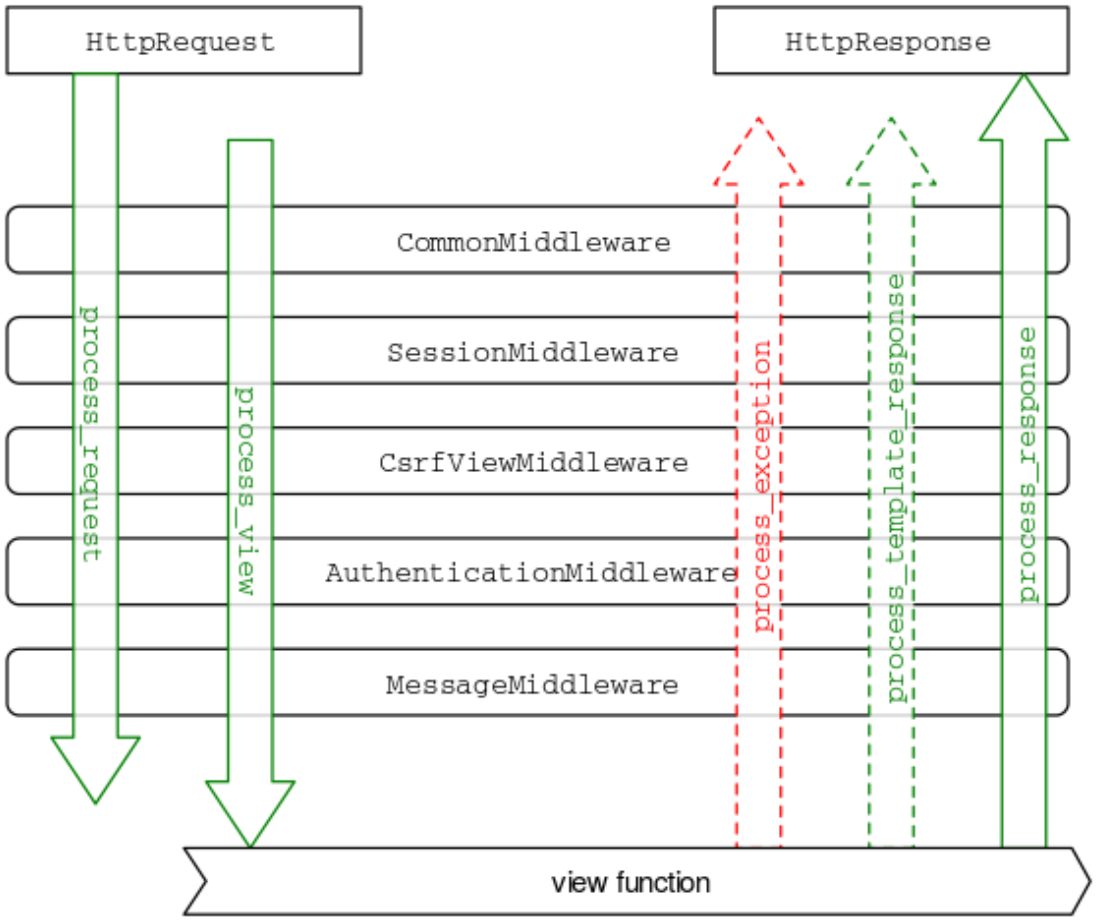

Abbildung 3.3.: Django Middleware Architektur [Dja15a]

Middleware in Django kann eine Anfrage vor oder nach ihrer Abarbeitung durch eine View manipulieren oder umleiten. Django selbst benutzt Middleware zur Authentifizierung der Benutzer und für verschiedene Sicherheitsfeatures. Eine Middleware muss in den Projekteinstellungen angegeben werden und mindestens eine der fünf möglichen Schnittstellen implementieren. Die eingetragenen Middlewares werden dabei vor der View in der Reihenfolge, in der sie in den Einstellungen definiert sind, abgearbeitet. Nach der View werden sie dabei in der umgekehrten Reihenfolge abgearbeitet.

Models

Models in Django sind Klassen, welche von django.db.models.Model erben und per Konvention immer in der Datei models.py enthalten sind. In der Modelklasse können nun alle nötigen Attribute definiert werden. Diese Models bildet Django nun auf die, in den Einstellungen angegebene, Datenbank ab, woraufhin Objekte des jeweiligen Models dort über bereitgestellte Methoden einfach gespeichert und abgefragt werden können. Auch komplexe Datenbankanfragen sind dabei leicht umzusetzen, weiterhin können auch reine SQL-Querys erzeugt werden. Django erleichtert hierbei auch die nachträgliche Veränderung der Datenstruktur, da diese nur im Model geändert werden muss und Django die Datenbank ohne großen Aufwand für den Benutzer anpasst.

3.2.3. D3.js

Das JavaScript-Framework D3.js hat einen strikten Fokus auf die Visualisierung von Daten unter Zuhilfenahme von HTML, CSS und SVG. Im Vergleich zu vielen vergleichbaren Frameworks zeichnet sich D3.js durch die gute DOM-Integration der erzeugten graphischen Elemente und der damit leicht implementierbaren Interaktivität aus [BHO11].

D3-Cloud

D3-Cloud ist eine Implementierung einer Wortwolke mit D3.js. In [Dav14] wird beschrieben, wie der von [Fei09] inspirierte Algorithmus funktioniert. D3-Cloud ist als Open Source veröffentlicht worden und ist bei GitHup zu finden [Dav15].

Um eine Wortwolke zu erstellen, muss D3-Cloud in eine bestehende Webseite integriert und mit einer gewichteten Wortliste versorgt werden.

3.2.4. jQuery

Die JavaScript-Bibliothek jQuery vereinfacht und erweitert häufig benötigte Funktionalität. jQuery ist seit langem das beliebteste JavaScript-Framework (siehe [Bui15]) und bildet die Basis für viele andere Frameworks [Que15].

3.2.5. Bootstrap

Bootstrap ist ein von Twitter entwickeltes CSS- und Javascript-Framework, welches von vielen bekannten Webseiten genutzt wird. Der Fokus von Bootstrap liegt dabei darauf, Webseiten zu gestalten, welche sowohl auf Smartphones, Tabletts und anderen Mobilgeräten, als auch auf Laptops und PCs eine ansprechende Optik und hohe Funktionalität. Bootstrap stellt dem Entwickler eine große Auswahl an UI-Elementen zur Verfügung, welche leicht an jedes gewünschte Farbschema angepasst werden können [Twi15].

3.3. Reading Archive

Um den Zweck des Archives in den Vordergrund zu stellen, wurde es unter dem Namen Reading Archive entwickelt. Als Basis dienten hierfür die jeweils neusten verfügbaren stabilen Versionen von Python (3.4) und Django (1.8).

Es kann jede mit Django kompatible Datenbank verwendet werden. Während der Entwicklung wurde SQLite[4] benutzt, welche bei der Erstellung eines neuen Django-Projektes standardmäßig aktiviert ist. Für die Studie wurde jedoch aufgrund der höheren Effizienz und Geschwindigkeit auf MySQL[5] umgestellt.

Zur Verwaltung der Benutzer wird auf die standardmäßige, in Django integrierte Benutzerverwaltung zurückgegriffen, welche alle benötigten Funktionalitäten aufweist und über dies gut in die automatische Admin-Seite integriert ist.

3.3.1. Datenmodell

Das Datenmodell des Reading Archives wurde, wie in Django üblich, in der models.py definiert und umfasst drei Klassen, welche von django.db.models.Model erben. Diese werden automatisch von Django in eine relationale Repräsentation übertragen und in der Datenbank abgespeichert.

Device

Ein Device-Objekt steht für ein konkretes Gerät, das von einem einzigen Benutzer genutzt wird.

[4]https://www.sqlite.org/
[5]https://www.mysql.de/

App

Ein App-Objekt steht für eine Android App, welche auf dem Gerät eines Benutzers installiert ist. Grund hierfür war es, die Möglichkeit zu schaffen verschiedene App-Versionen, die bei mehreren Benutzern und Geräten vorkommen können, abzubilden.

Entry

Ein Entry-Objekt repräsentiert einen, von einem Benutzer, auf einem Gerät, mit einer App, zu einem bestimmten Zeitpunkt, gelesenen Text.

3.3.2. Oberfläche

Alle Ansichten des Reading Archives basieren auf einem Basistemplate, in dem die generelle Struktur festgelegt und das Navigationsmenü definiert wurde.

Dabei wurde auf das Bootstrap-Framework (siehe 3.2.5) zurück gegriffen, damit das Reading Archive sowohl mit Computern als auch mit Smartphones nutzbar ist, ohne separate Seiten zur Verfügung stellen zu müssen.

In Django wurde Bootstrap durch die (Django-) App *bootstrap3*[6] bereitgestellt und durch einfache Befehle im Basistemplate integriert. Die daraus entstehende Grundstruktur ist in Listing 3.4 veranschaulicht.

[6]https://github.com/dyve/django-bootstrap3

```
<!DOCTYPE html>
{% load staticfiles %}
{% load bootstrap3 %}
<html>
<head>
      ...
      <title>
            {% block bootstrap3_title %}
                  Default Title
            {% endblock %}
      </title>
      {% bootstrap_css %}
      {% bootstrap_javascript jquery=True %}
      ...
</head>
<body>
      {% block bootstrap3_content %}
            ... Navigationsleiste
            ... Inhalt
      {% endblock %}
</body>
</html>
```

Listing 3.4: Basistemplate mit Bootstrap Integration

Das Navigationsmenü wurde mit den Bootstrap-Elementen erstellt und mit dem dunklen Design versehen. Der Schriftzug *The Reading Archive* ist hierbei auch ein Link, welcher die aktuelle Seite, mit den aktuellen Parametern neu lädt. Dies war vor allem für die Mobilansicht wichtig, in der der Rest des Menüs nur bei bedarf angezeigt wird. Es existieren 3 verschiedene Ansichten des Navigationsmenüs: **Standardansicht**, **Benutzeransicht** und die **Adminansicht**

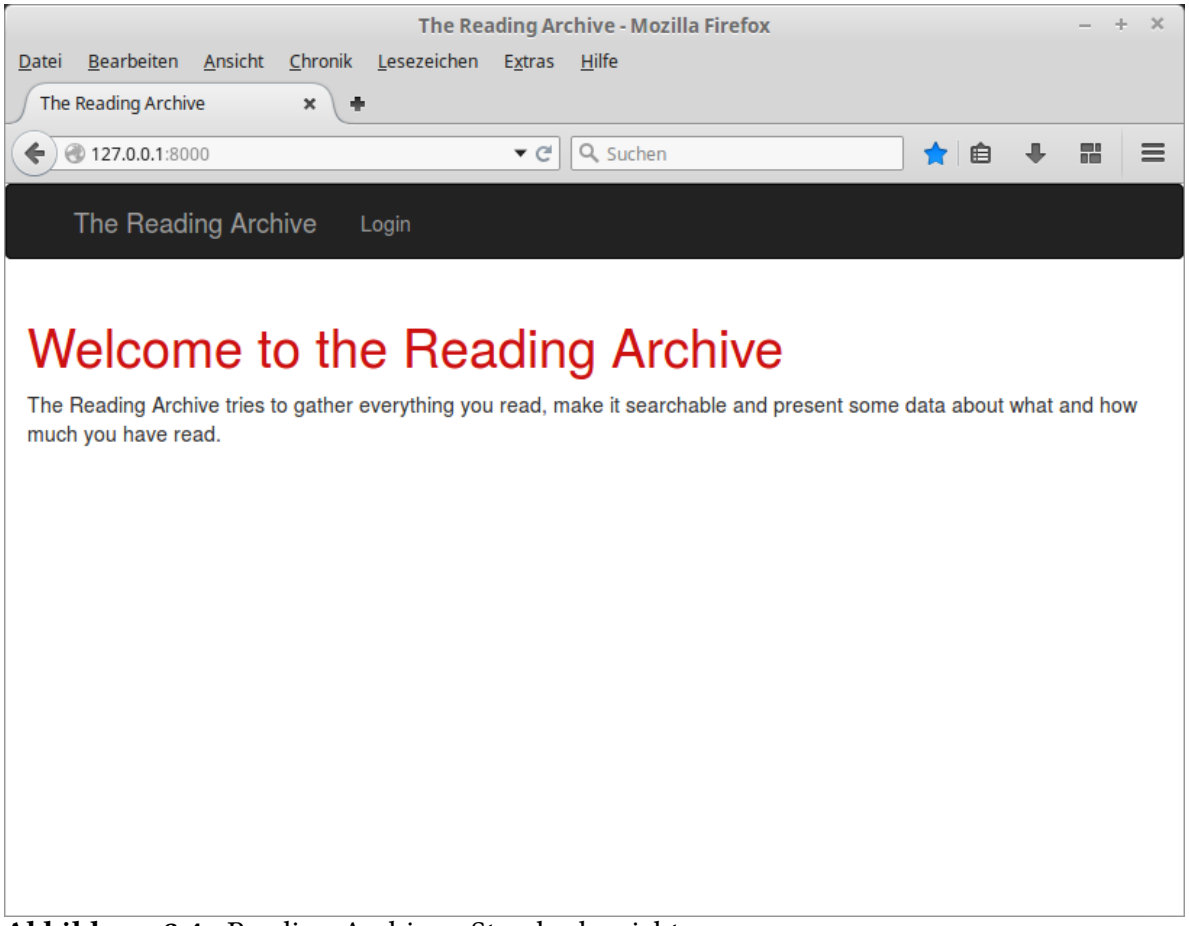

Abbildung 3.4.: Reading Archive - Standardansicht

Die Default-Ansicht für Benutzer, die nicht eingeloggt sind und nur einen kurzen Begrüßungstext anzeigt, sowie den Link zum Login.

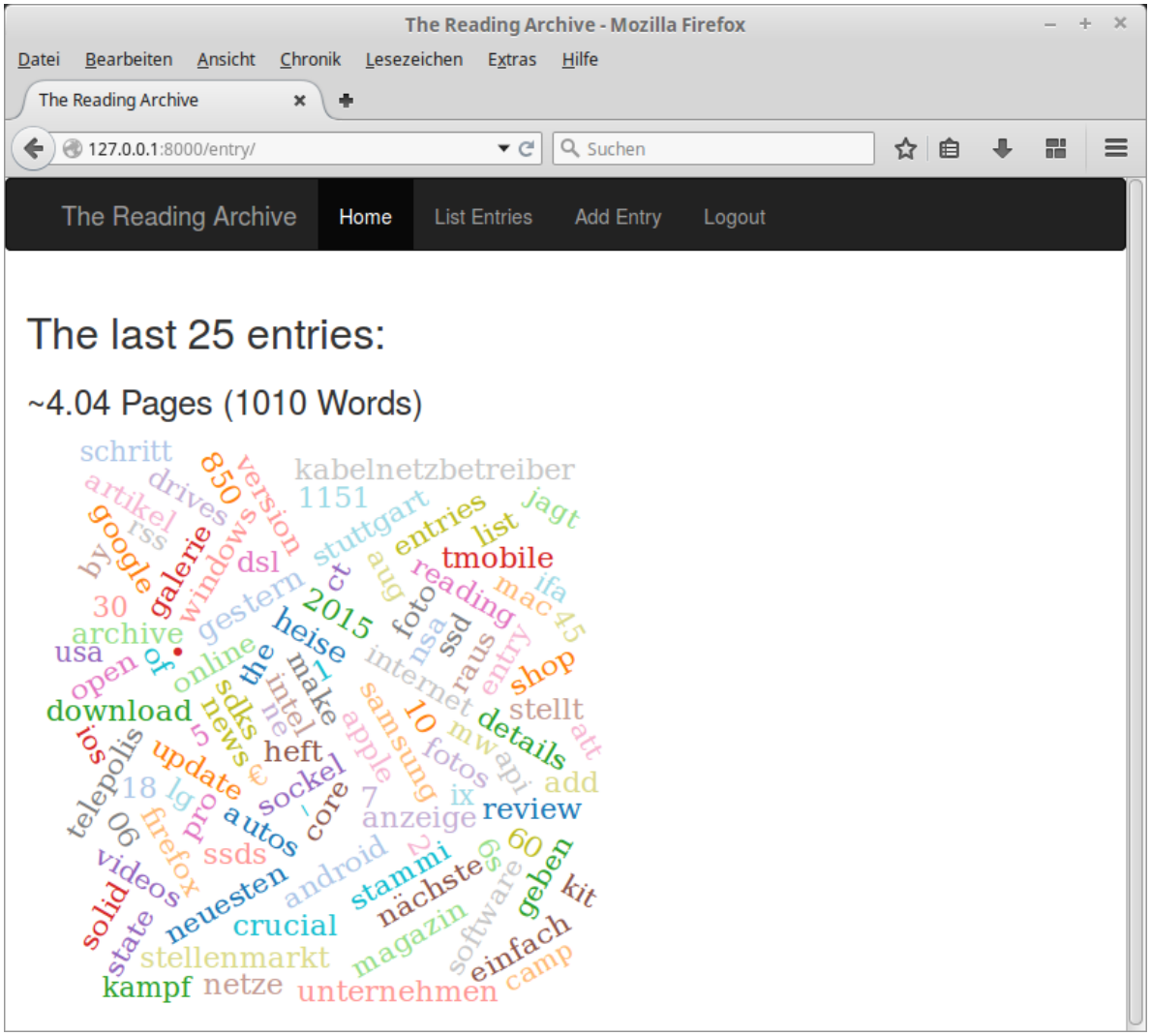

Abbildung 3.5.: Reading Archive - Benutzeransicht

Sobald der Benutzer eingeloggt ist stehen ihm **Home**, **List Entries**, **Add Entry** und **Logout** zur Verfügung.

Die Home-Ansicht beinhaltet dabei Statistiken über das Leseverhalten des Benutzers. Diese umfassen die Anzahl der in den letzten 24 Stunden gelesenen Wörtern, die Umrechnung dieser Wörter auf DIN A4 Seiten und eine Wortwolke aus den Wörtern, die in dieser Wortmenge am häufigsten vorkommen, näheres dazu in Abschnitt 3.3.4.

Die ungefähre Anzahl der benötigten DIN A4 Seiten berechnet sich aus der Normseite[7] und der durchschnittlichen Länge des deutschen Wortes[8]. Die Normseite ist mit 30 Zeilen auf 60 Anschlägen definiert und entspricht somit 1800 Zeichen. Das durchschnittliche deutsche Wort ist ungefähr 6 Buchstaben lang, 7 mit jeweils folgendem Leer- oder Satzzeichen. Daraus ergeben sich etwa 250 Wörter pro Seite.

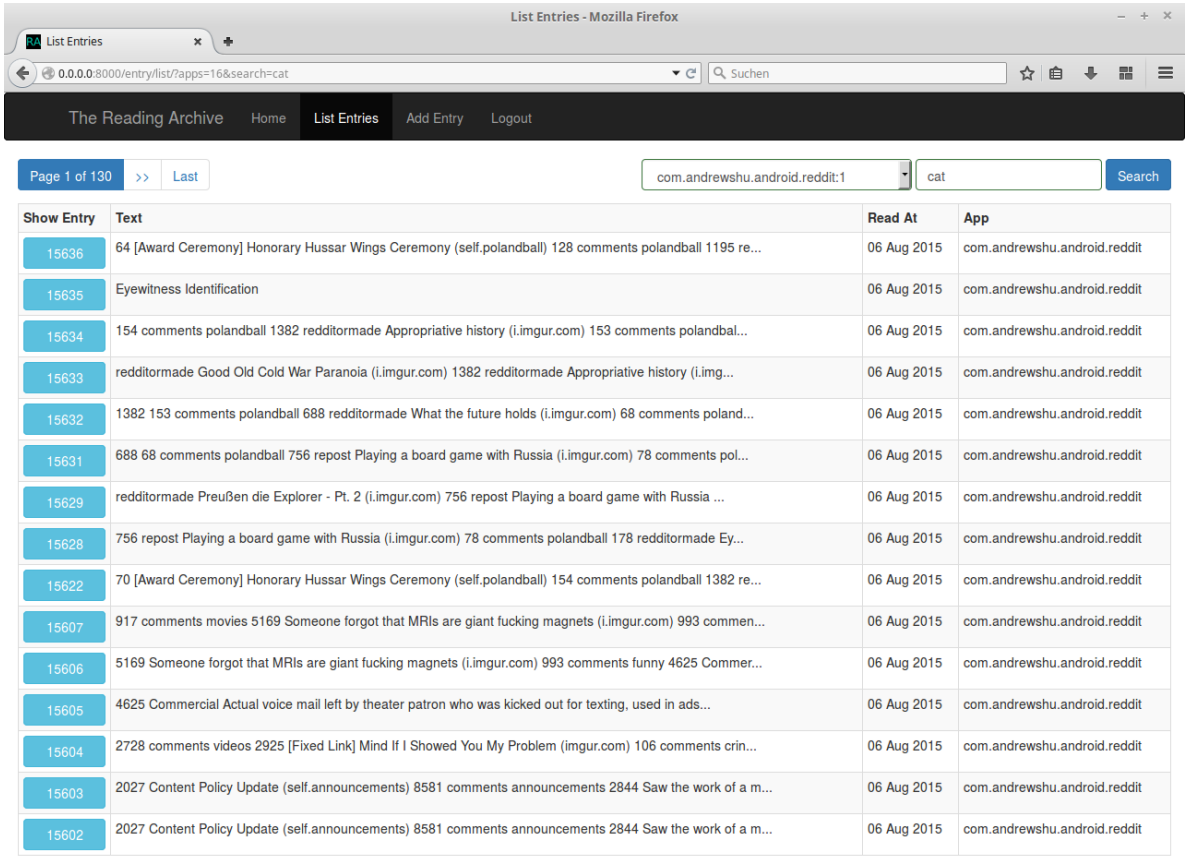

Abbildung 3.6.: Reading Archive - List Entries

List Entries liefert dem Benutzer eine Liste aller von ihm erstellten Entries, also aller gesammelten Texte. Diese Liste lässt sich nach Begriffen durchsuchen und auf (Android-) Apps einschränken. Die neusten Einträge werden dabei zuerst angezeigt.

[7] http://www.literaturuebersetzer.de/pages/wissenswertes-archiv/normseite.htm
[8] http://www.duden.de/sprachwissen/sprachratgeber/durchschnittliche-laenge-eines-deutschen-wortes

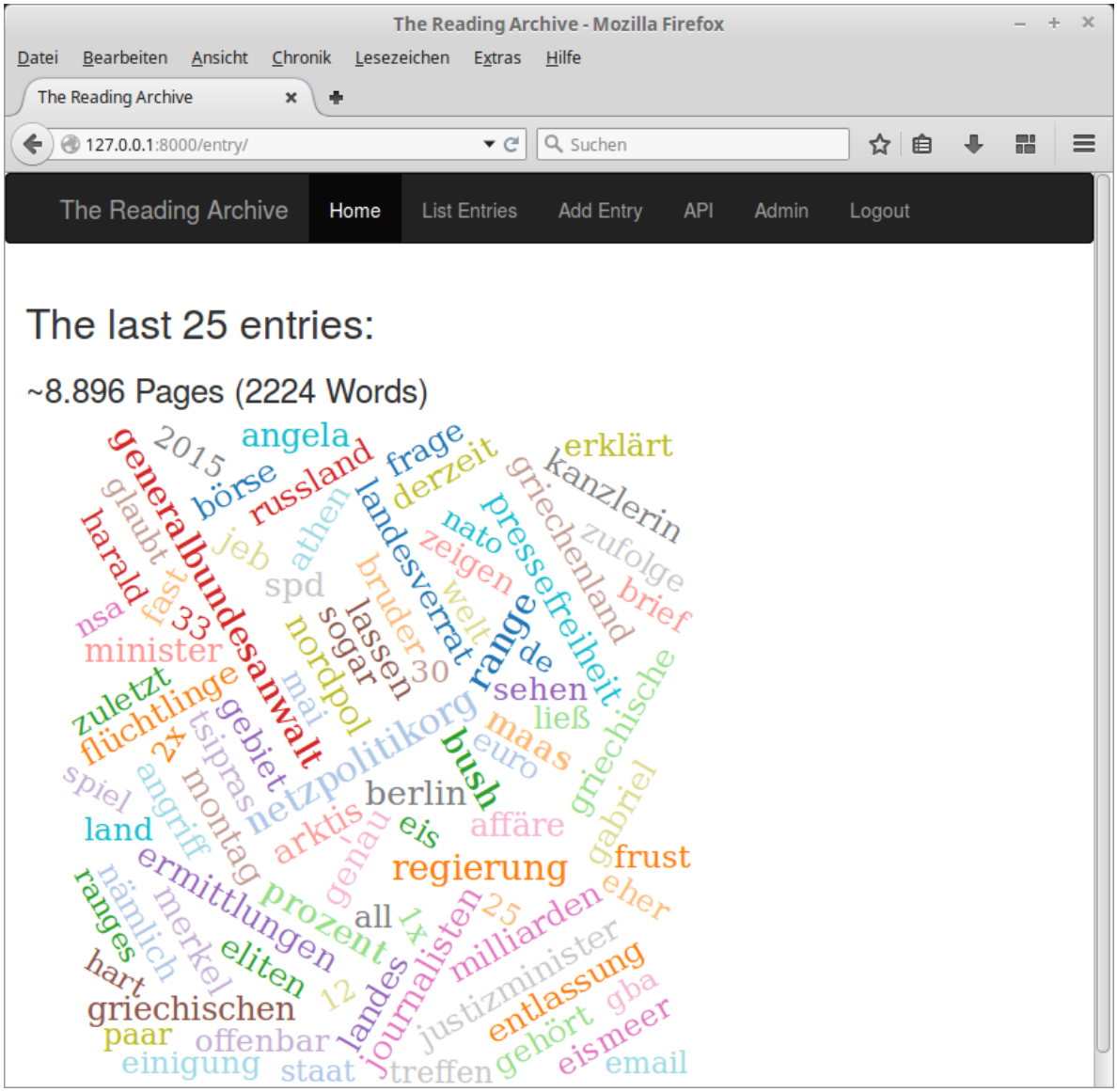

Abbildung 3.7.: Reading Archive - Adminansicht

Für eingeloggte Admin-Benutzer wurde die Navigationsleiste um zwei Elemente erweitert: **API** und **Admin**. Über erstere gelangt man in eine HTML-Ansicht der REST-API und über letzteres in die von Django bereitgestellte Admin-Seite.

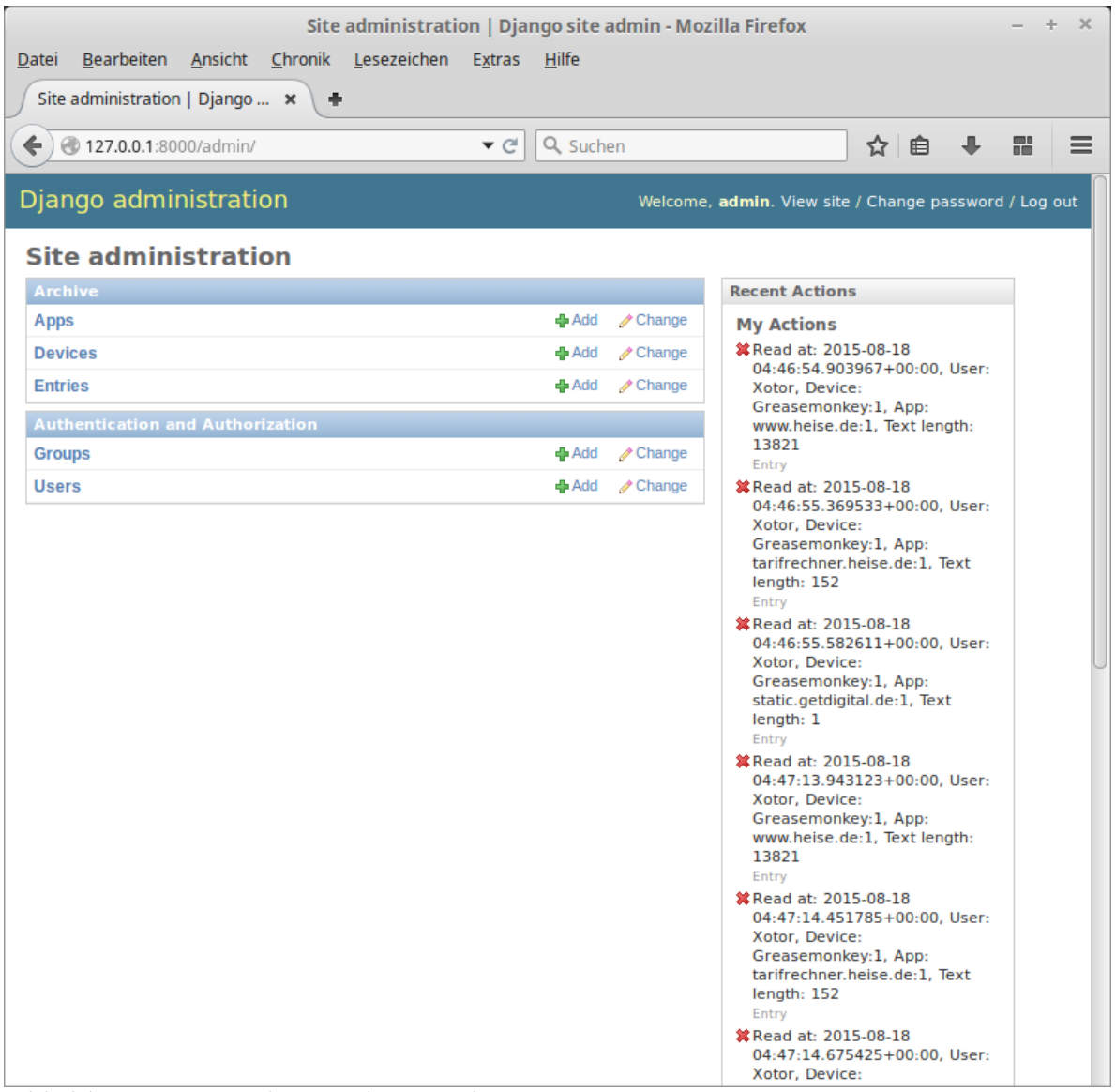

Abbildung 3.8.: Reading Archive - Admin-Menü

Die Admin-Seite wurde um die drei erstellten Modell-Klassen erweitert und bietet die Möglichkeit diese hinzuzufügen, anzuzeigen, zu ändern oder zu löschen. Selbiges steht natürlich für Benutzer und Benutzergruppen zur Verfügung, welche von Django standardmäßig bereit gestellt werden.

Die Gestaltung der Admin-Seite wurde nicht angepasst, da der Benutzer sie niemals sehen wird. Aus dem selben Grund wurden auch die Ansichten für Apps, Devices und Entries auf dem nötigen Minimum belassen. Sie sind jedoch trotzdem voll funktionsfähig.

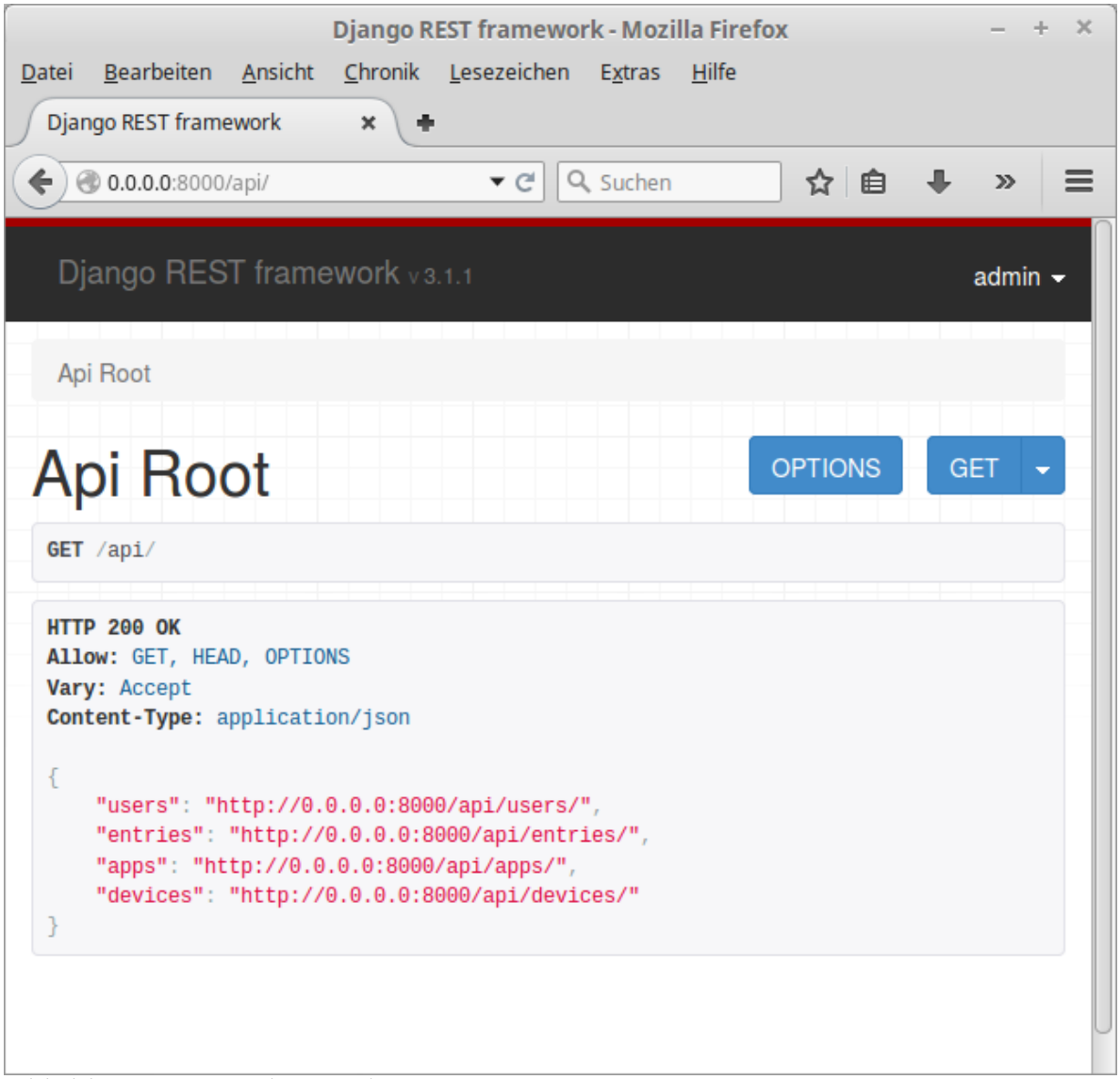

Abbildung 3.9.: Reading Archive - REST-API

Die REST-API-Ansicht wird von der Django-App *Rest_Framework*[9] für alle Modelle bereitgestellt, welche serialisierbar sind und über das Implementieren einer ViewSet-Klasse dem REST-Framework zur Verfügung gestellt werden.

Die REST-API ist auch direkt, ohne HTML-Ansicht für alle Benutzer zugänglich. Dabei kann jeder Benutzer über diese API die Objekte sehen und verändern, die er auch über die normale Webansicht sehen könnte.

[9]http://www.django-rest-framework.org/

Durch diese REST-API ist das Reading Archive sehr leicht mit anderen Programmen koppelbar und kann gut in andere Projekte integriert werden.

3.3.3. Logging

Um besser untersuchen zu können, wie das Reading Archive genutzt wird, wurde eine eigene Django-Middleware implementiert, welche alle eingehenden Anfragen abspeichert und nach Benutzern sortiert ablegt.

Dabei wurde darauf geachtet, die eventuell gesendeten Passwörter nicht abzuspeichern, um diese nicht zu kompromittieren. Somit ist es im Nachhinein möglich nachzuvollziehen, wie oft auf eine View zugegriffen wurde oder von wem.

3.3.4. Erstellung einer Wortwolke

Um die Wortwolke darstellen zu können muss zunächst eine Liste aller Wörter mit ihren Häufigkeiten erstellt werden. Allerdings befinden sich in jedem Text inhaltslose Worte, die zwar sehr häufig auftreten, aber nichts über den Inhalt des Textes aussagen. Diese werden Stoppwörter genannt und umfassen bei deutschen Texten unter anderem bestimmte sowie unbestimmte Artikel, Konjugationen und Präpositionen. Um diese Stoppwörter nun aus dem Text heraus filtern zu können, benutzt man in der Regel so genannte Stoppwortlisten. Verschiedene dieser, meist mehre hundert Worte umfassenden, Listen kann man problemlos im Internet finden.

Die Stoppwortlisten der Universität Neuenburg [10] waren eine der Umfangreichsten. Gleichzeitig bieten sie auch sehr viele Sprachen an. Derzeit sind nur die englischen und deutschen Stoppwortlisten integriert, weitere lassen sich jedoch bei Bedarf mit sehr geringem Aufwand hinzufügen.

Stoppwortlisten sind für jede Sprache sehr unterschiedlich und enthalten gleichzeitig die Wörter die am häufigsten vorkommen. Daher kann man sie auch zur Spracherkennung einsetzen, indem man zählt, welche Stoppwortliste am meisten Wörter aus dem Text enthält.

Die EntryHomeView, die dafür zuständig ist die Wortwolke zu erstellen, wird nun zuerst alle Entries, die in den letzten 24 Stunden erstellt wurden abfragen. Sofern dies weniger als 25 Entries sind, werden die letzten 25 Entries angezeigt. Dies soll verhindern, dass einem Benutzer nach mehrtägiger Pause ein leerer Bildschirm angezeigt wird.

Der Text jeder Entry wird nun nacheinander Verarbeitet. Zuerst wird der Text von Sonderzeichen befreit, danach werden alle Großbuchstaben in Kleinbuchstaben umgewandelt. Als

[10] http://members.unine.ch/jacques.savoy/clef/index.html

nächstes wird mit Hilfe der Stoppwortliste die Sprache des Textes erkannt, anschließend werden alle Stoppwörter entfernt, die Häufigkeit aller verbliebenen Wörter gezählt und mit den Häufigkeiten der anderen Entries aufaddiert.

Die gewichtete Wortliste, die dadurch erstellt wurde, wird über das Template an D3-Cloud (siehe Abschnitt 3.2.3) übergeben. Dieses erstellt nun nach dem von Feinberg in [Fei09] beschriebenen Algorithmus eine Wortwolke.

Die Textgröße in der Wortwolke wird dabei logarithmisch an der Worthäufigkeit skaliert. Die Größe der Wortwolke wird automatisch an die Displaygröße angepasst, so dass die Wortwolke vollständig auf dem Display zu sehen ist.

Die gewichteten Wortlisten werden bei jedem Aufruf durch das Reading Archive neu berechnet, während die Darstellung der Wortwolke im Browser des Benutzers berechnet wird.

3.4. Reading Tracker

Die Android-App zum Abgreifen der Texte, wurde Reading Tracker genannt. Es wird mindestens Android 4.4 vorausgesetzt, da ansonsten einige benötigte API-Schnittstellen nicht vorhanden sind. Des Weiteren fordert der Reading Tracker bei der Installation die folgenden Berechtigungen an:

- Zugriff auf das Internet
- Netzwerkstatus abfragen
- Einbinden einer Bedienungshilfe

Die Reading Tracker App kann in drei Bereiche aufgeteilt werden: Den Tracker Service, den Send Service und die Benutzeroberfläche, welche aus 9 verschiedenen Activities besteht.

3.4.1. Tracker Service

Der Tracker Service ist ein Accessibility Service (Siehe Abschnitt 3.2.1), dessen Aufgabe es ist die gelesenen Texte zu extrahieren. Er muss dafür allerdings manuell vom Benutzer über die Einstellungen aktiviert werden. Dieser manuelle Zwischenschritt ist nötig, um eine böswillige Ausspähung der Benutzer zu verhindern.

Der Tracker Service wird nicht von der App, sondern direkt vom Android-System gestartet. Dies geschieht jedes mal, wenn ein Accessibility Event auftritt, für das der Tracker Service aufgerufen werden will. Diese Accessibility Events werden in einer Konfigurationsdatei eingetragen, welche im Android Manifest mit dem Tracker Service verbunden wird und noch weitere wichtige Einstellungen für den Tracker Service enthält.

Wenn der Tracker Service vom System aufgerufen wird, so geschieht das über die onAccessibilityEvent-Methode, welcher das auslösende Accessibility Event als Parameter übergeben wird. Dieses Event hat zugriff auf ein so genanntes AccessibilityNodeInfo-Objekt, dieses repräsentiert jeweils ein Element der Benutzeroberfläche und erlaubt den Zugriff sowohl auf sein Elternelement als auch seine Kinderelemente.

Die Benutzeroberfläche in Android ist als Baumstruktur aufgebaut, deren äußerstes Element immer ein Layout-Element ist, welches bestimmt wie die enthaltenen Kind-Elemente angeordnet sind. Dies können weitere Layout-Elemente, Buttons, Texte oder Bilder sein.

Bei den für den Tracker Service wichtigen Events TYPE_WINDOW_CONTENT_CHANGED und TYPE_View_SCROLLED wird hier einfach das oberste Benutzeroberflächenelement zurückgeliefert. Davon ausgehend lässt sich jetzt die komplette Baumstruktur der Benutzeroberfläche, durchlaufen. Allerdings hat man nicht direkt zugriff auf die Elemente der Benutzeroberfläche, sondern nur auf die AccessibilityNodeInfo-Objekte, welche die jeweiligen Elemente repräsentieren. Über diese Nodes kann man jedoch den Text, welcher direkt in ihnen angezeigt wird, abfragen.

Um also an jeden angezeigten Text zu gelangen geht der Tracker Service von dem übergebenen Accessibility Event aus durch den kompletten Baum der AccessibilityNodeInfo-Objekte und fügt die dabei gefundenen Texte aneinander. Aus diesem Text und dem Namen der App, in welcher der Text gefunden wurde, wird ein Entry-Objekt erstellt.

Dies funktioniert leider nur bei vom Android-Framework vorgegebenen Elementen der Benutzeroberfläche. Denn diese haben jeweils die Schnittstelle implementiert, welche dafür nötig ist, um aus einem Element der Benutzeroberfläche ein entsprechendes AccessibilityNodeInfo-Objekt zu erstellen. Allerdings ist es in Android auch möglich, eigene Elemente für die Benutzeroberfläche zu implementieren, indem man von den entsprechenden Basisklassen ableitet. Wenn dabei die entsprechenden Schnittstellen für die Accessibilty implementiert werden, funktionieren auch diese Elemente.

Beim Testen des Tracker Service wurde jedoch klar, dass viele Entwickler diese Schnittstelle nicht implementiert haben, was dazu führt, dass ein leeres AccessibilityNodeInfo-Objekt erstellt wird, das weder den textuellen Inhalt, noch eventuell vorhanden Kindknoten enthält. Dies führt dazu, das nicht immer der komplette Text, welcher auf dem Display zu sehen ist, ausgelesen werden kann.

Die erstellte Entry wird nun mit allen Entries, welche schon in der Datenbank gespeichert sind, verglichen. Dies soll verhindern, dass zwei Entries mit dem selben Text erstellt werden. Sollte es ein neuer Text sein, so wird die Entry im Tracker Service zwischengespeichert und der Zeitpunkt der Erstellung dieser Entry festgehalten.

Beim nächsten Accessibility Event, dass einen anderen, nicht leeren Text enthält, wird nun geprüft, ob der Zeitpunkt der zwischengespeicherten Entry eine vorgegebene Mindestzeit überschreitet. Wenn dies der Fall ist, so gilt der Text als gelesen und die Entry wird in

der internen SQLite-Datenbank gespeichert, ansonsten wird sie verworfen. Nach einigen Testläufen wurde diese Mindestzeit auf 1500 Millisekunden festgelegt, da auch sehr kurze Texte am Smartphone gelesen werden und auch diese registriert werden sollten.

Wurde eine Entry in der SQLite-Datenbank gespeichert, so wird vom Tracker Service der Send Service gestartet, welcher die weitere Verarbeitung der Entry übernimmt.

3.4.2. Send Service

Der Send Service ist im Gegensatz zum Tracker Service ein normaler Service (Siehe Abschnitt 3.2.1) und dient dazu, alle noch nicht versendeten Entrys in der SQLite-Datenbank an das Reading Archive zu schicken. Dies wurde nicht im Tracker Service realisiert, da dieser sonst zu lange blockiert sein könnte.

Wenn der Send Service gestartet wurde, prüft er zunächst, ob eine Netzwerkverbindung vorhanden ist und iteriert anschließend über alle Entries aus der Datenbank. Dabei verschickt er jede Entry, die noch nicht versendet wurden, an das Reading Archive.

3.4.3. Benutzeroberfläche

Der Reading Tracker besteht aus 9 verschiedenen Activities (siehe Abschnitt 3.2.1). Zusammen bilden sie die Benutzeroberfläche dieser App.

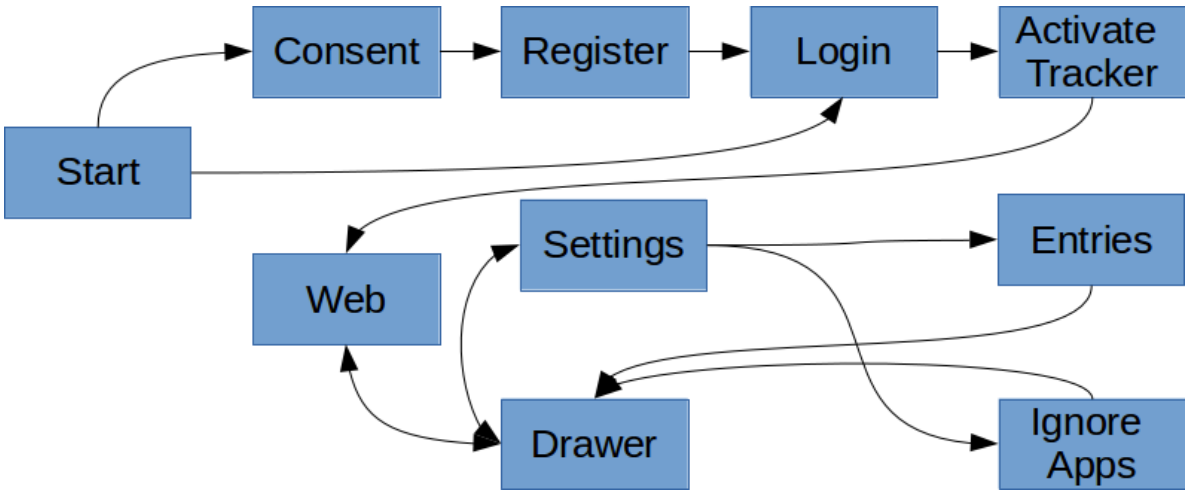

Abbildung 3.10.: Activities

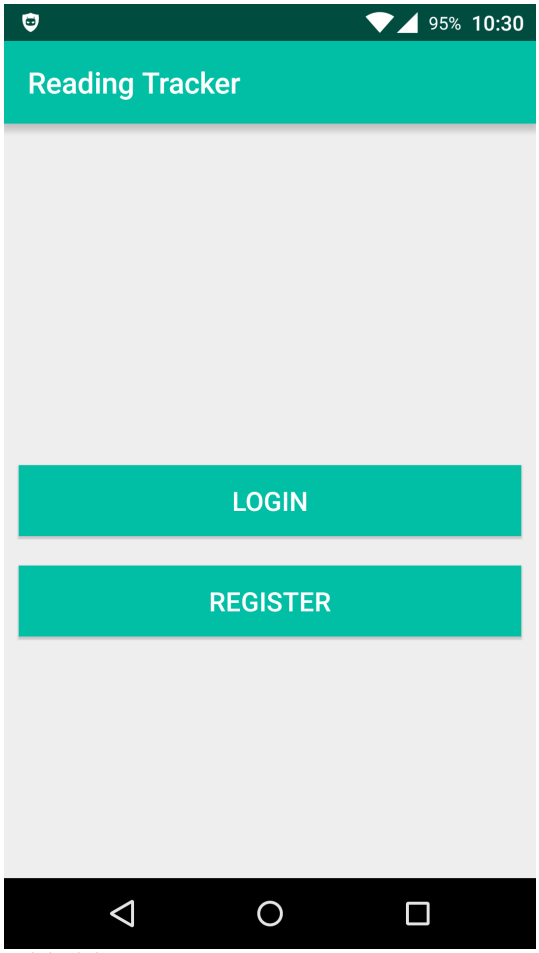

Abbildung 3.11.: Start Activity

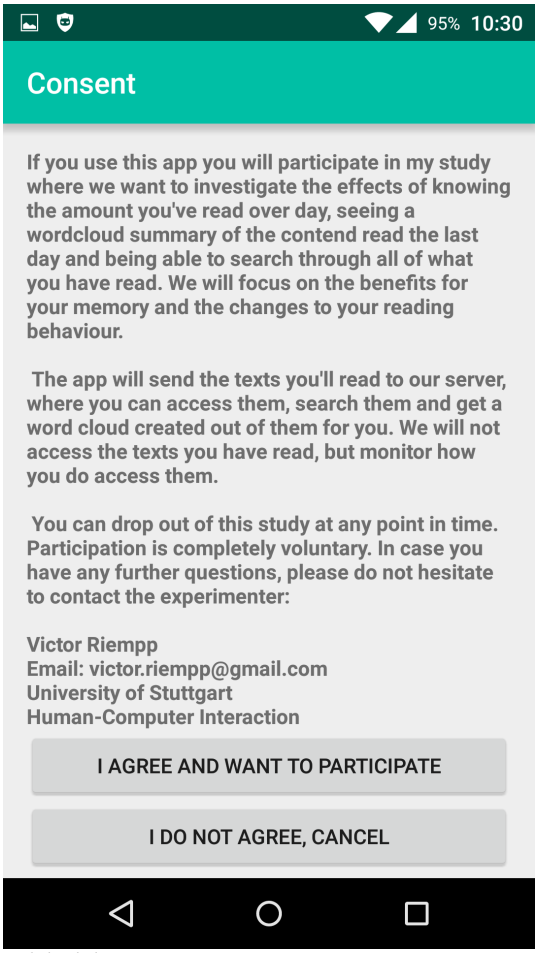

Abbildung 3.12.: Consent Activity

Zunächst wird die Start Activity gestartet. Von hier kann man sich entweder mit einem existierenden Benutzerkonto einloggen oder ein neues Benutzerkonto registrieren.

Bevor man sich registrieren kann, muss man jedoch erst der Teilnahme an der Studie und der Speicherung der Texte im Reading Archive zustimmen.

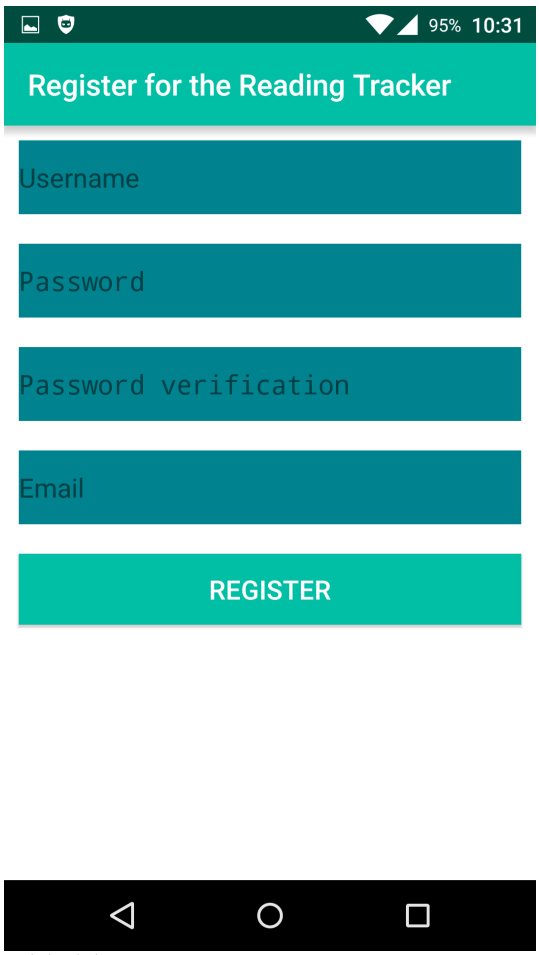

Abbildung 3.13.: Register Activity

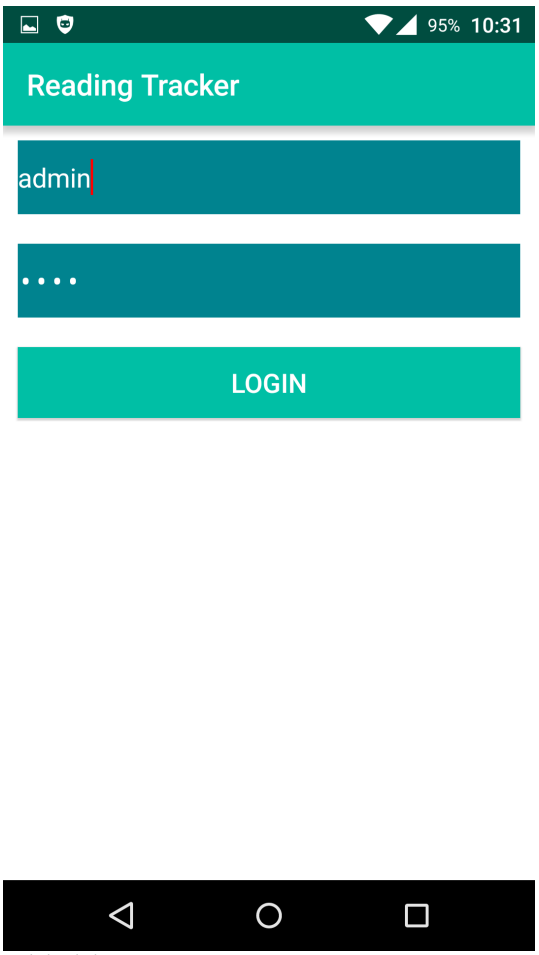

Abbildung 3.14.: Login Activity

Um den Reading Tracker benutzen zu können muss ein Benutzerkonto im Reading Archive erstellt werden. Hierfür muss ein unbenutzter Benutzername und eine unbenutzte E-Mail-Adresse angegeben werden. Das Passwort muss zweimal eingegeben werden, um eine versehentliche Fehleingabe auszuschließen.

Um sich mit einem bestehenden Benutzerkonto einzuloggen genügt es den Benutzername und das Passwort angegeben. Der Reading Tracker speichert die letzten erfolgreich benutzten Anmeldedaten. Diese werden beim Ausloggen gelöscht, stehen jedoch beim Neustart der Benutzeroberfläche noch zur Verfügung.

Sofern Anmeldedaten gespeichert sind wird der Benutzer automatisch eingeloggt.

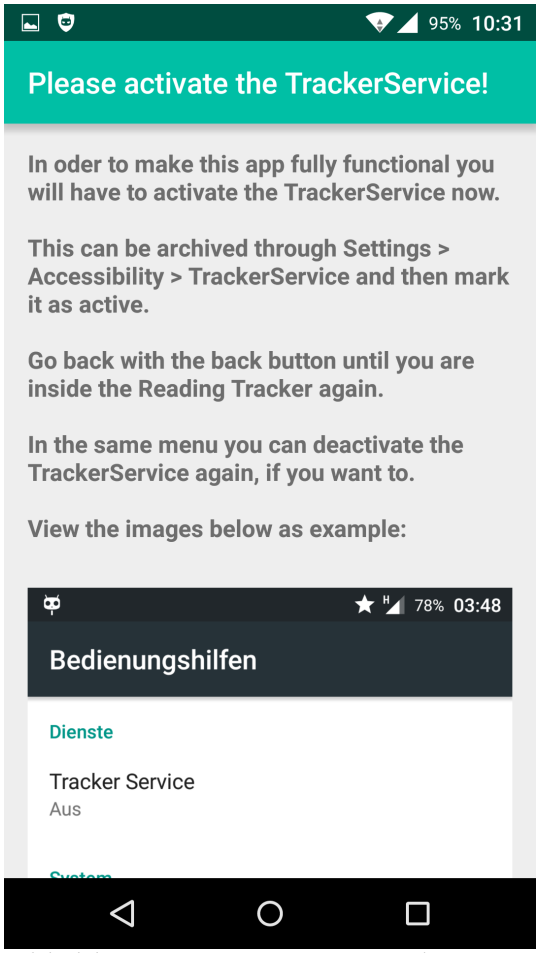

Abbildung 3.15.: Activate Tracker Activity 1

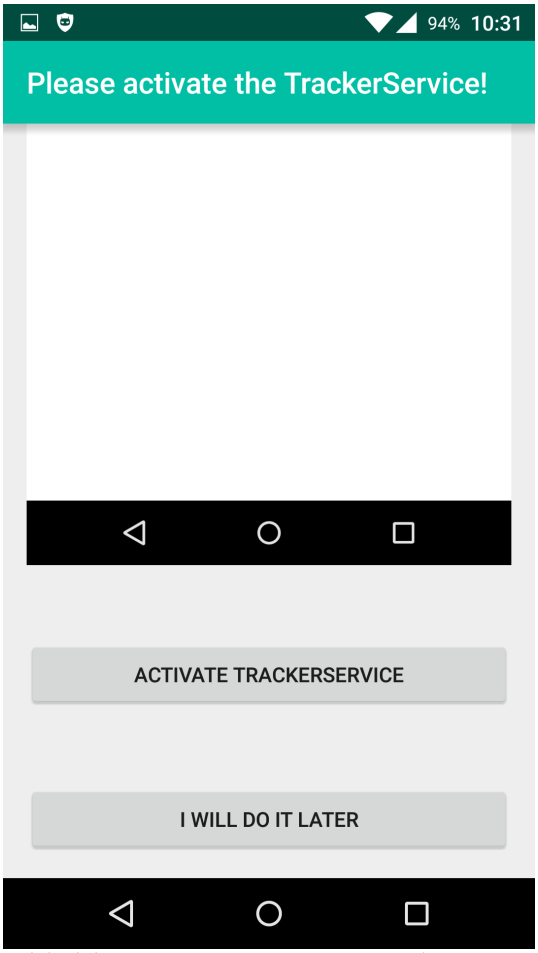

Abbildung 3.16.: Activate Tracker Activity 2

Nach einem erfolgten Login wird überprüft, ob der Tracker Service (Siehe Abschnitt 3.4.1) in den Einstellungen aktiviert wurde.

Wenn dies nicht der Fall ist, so wird der Benutzer gebeten diesen zu aktivieren. Dazu wird auch gleich eine Beschreibung in Wort und Bild angezeigt, wie er den Tracker Service aktivieren kann.

Dem Benutzer steht natürlich auch frei, das Aktivieren des Tracker Service zu umgehen.

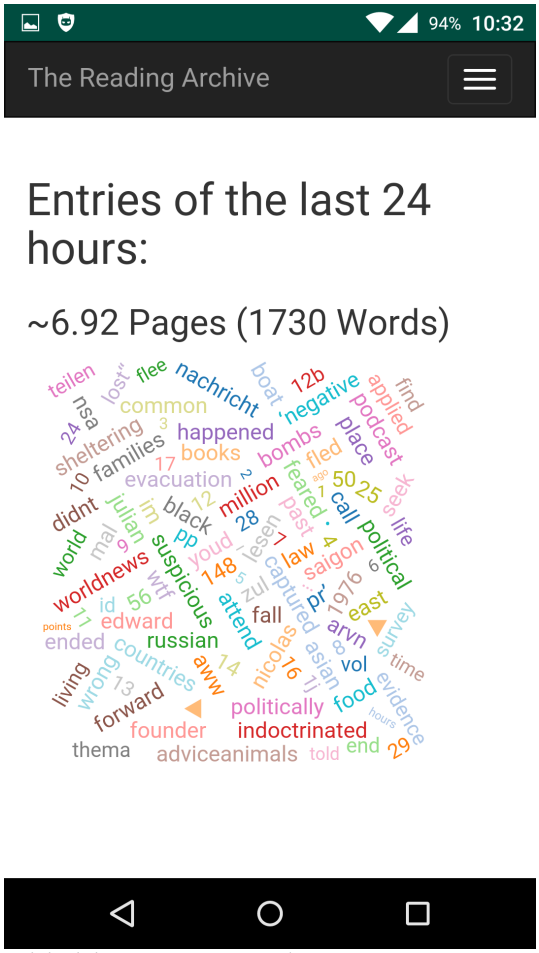

Abbildung 3.17.: Web Activity

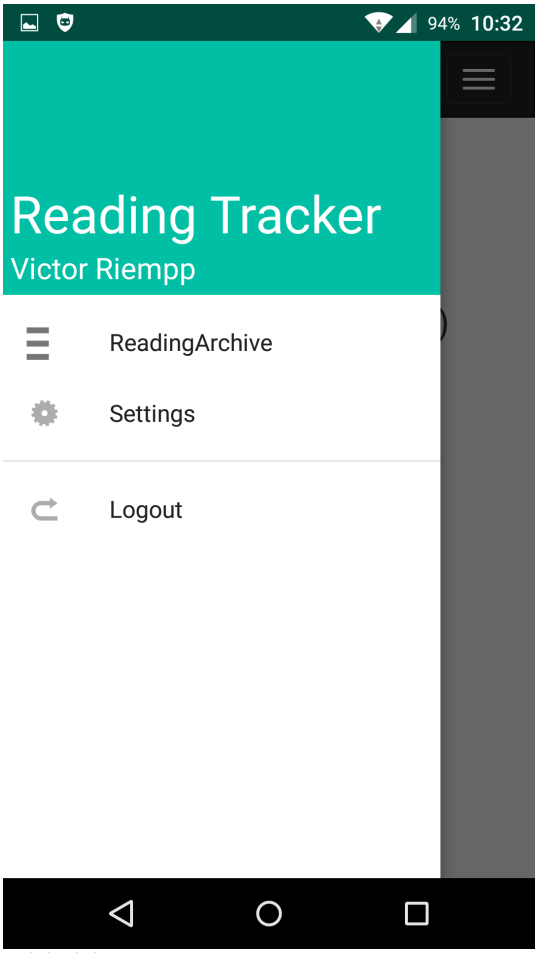

Abbildung 3.18.: Navigation Drawer

Nach erfolgtem Login wird dem Benutzer die Web Activity angezeigt. In dieser wird das Reading Archive geladen und der Benutzer automatisch eingeloggt. So präsentiert sich ihm direkt die Standartansicht mit der Quantifizierung der gelesenen Texte und der Wortwolke.

Streicht der Benutzer mit dem Finger vom linken Bildschirmrand aus in die Mitte, so schiebt sich der Navigation Drawer über die derzeitige Activity. Über diesen kann der Benutzer zwischen dem Reading Archive und den Einstellungen für den Reading Tracker wechseln oder sich ausloggen.

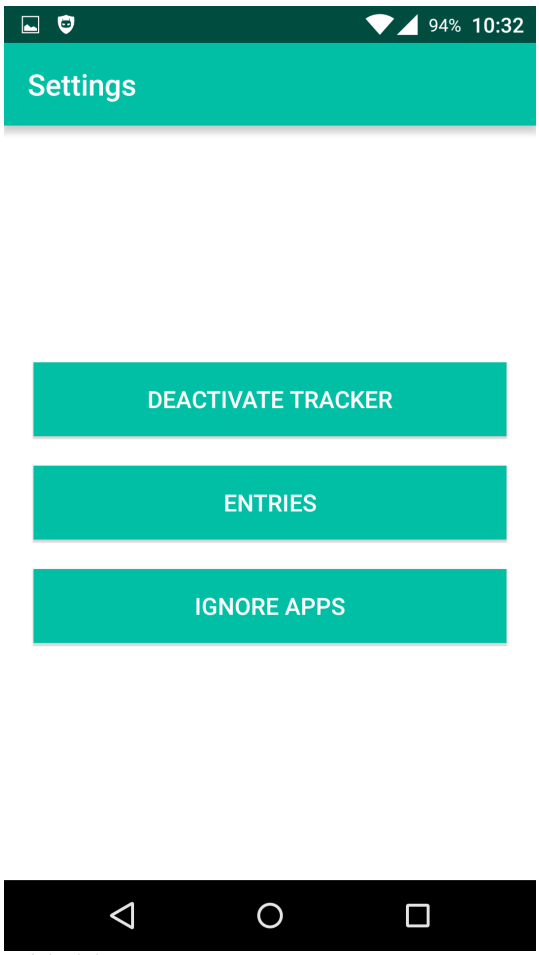

Abbildung 3.19.: Settings Activity

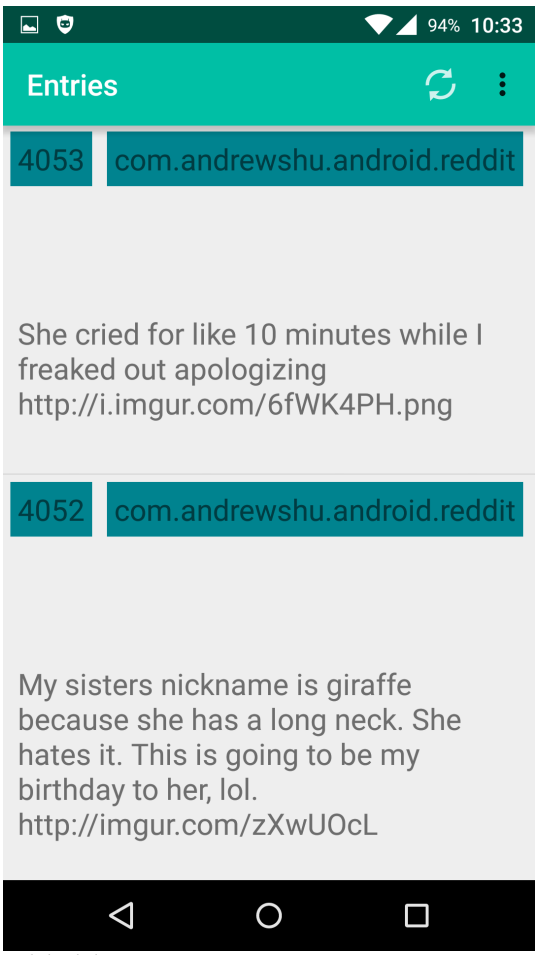

Abbildung 3.20.: Entries Activity

Über den Activate oder Deactivate Tracker Button kommt der Benutzer in die Systemeinstellungen, wo er den Tracker Service aktivieren oder deaktivieren kann. Zusätzlich gibt die Beschriftung des Buttons Auskunft über den aktuellen Zustand des Tracker Service.

Über Entries gelang der Benutzer in eine Ansicht aller, in der SQLite-Datenbank gespeicherten, Entries. Hierbei werden die neusten Entries zuerst angezeigt. Über das Menü in der oberen rechten Ecke kann der Benutzer Endries aus der SQLite-Datenbank löschen.

Über Ignore Apps kann der Benutzer den Tracker Service daran hindern, die Texte einer App zu erfassen.

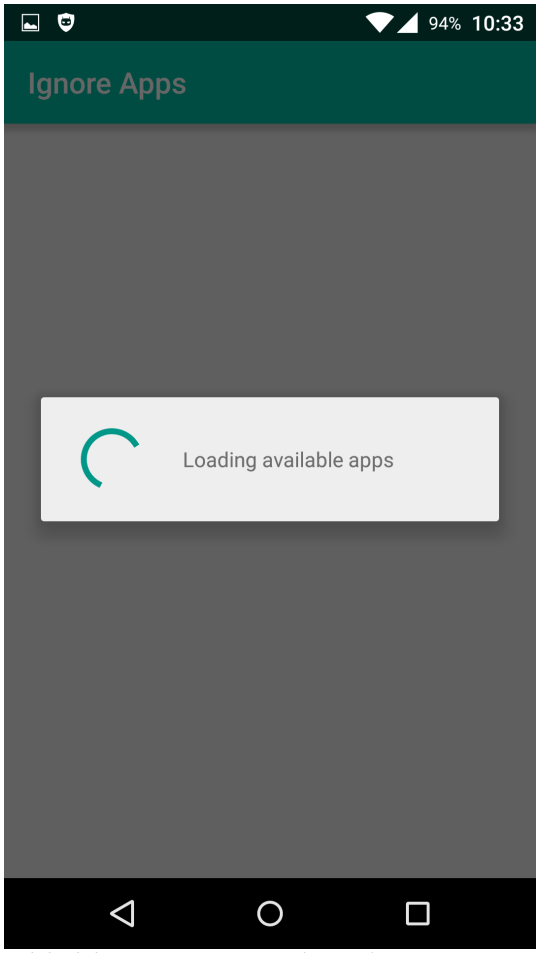

Abbildung 3.21.: Loading the Apps

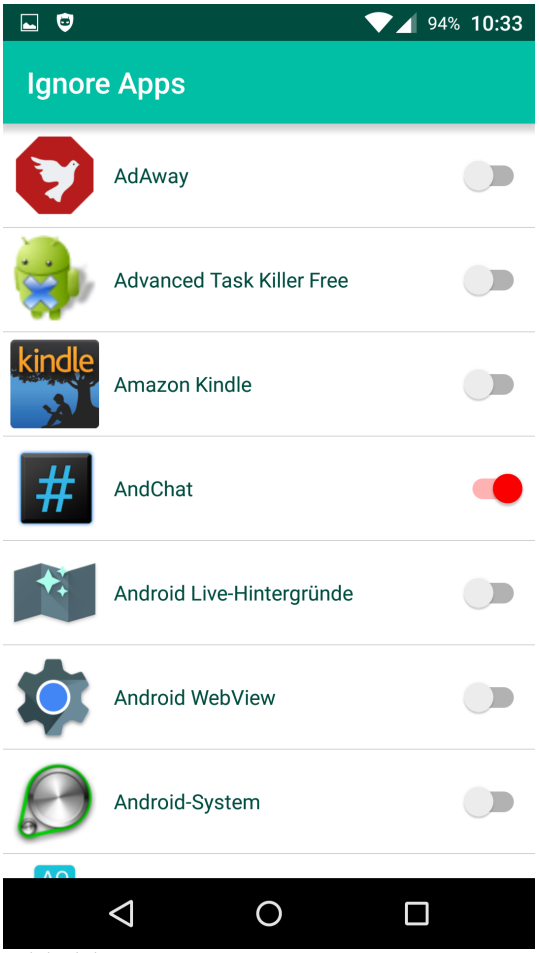

Abbildung 3.22.: Ignore Apps Activity

Beim erstmaligen Start der Ignore Apps Activity müssen alle installierten Apps und deren Icons abgefragt werden. Dieser Vorgang kann bei vielen installierten Apps oder bei langsamen Geräten einige Zeit in Anspruch nehmen. Sobald alle Apps geladen wurden, werden sie alphabetisch sortiert in einer Liste dargestellt.

Durch ein einfaches umlegen des Schalters kann nun eine Activity auf Ignore gesetzt werden. Dies geschieht sofort beim Betätigen des Schalters und verhindert das der Tracker Service Texte dieser Activity erfassen kann.

3.5. Reading Tracker Web

Die Erweiterungen Greasemonkey für Firefox und Tampermonkey für Chrome erlauben das Ausführen von beliebigem Javascript-Code im Browser. Dabei wird der Code in der Umgebung der aktuellen Webseite ausgeführt.

Der Reading Tracker Web ist ein solches Greasemonkey-Script und sendet den sichtbaren Text aller Webseiten an das Reading Archive.

Der Reading Tracker Web konnte nicht für die Studie eingesetzt werden, da momentan der Benutzername und das Passwort im Klartext in die Scriptdatei geschrieben werden müssen. Auch gibt es bisher noch keine Möglichkeit Webseiten von der Übertragung auszuschließen. Es handelt sich hierbei vor allem um einen Nachweis der Machbarkeit.

4. Studie

In diesem Kapitel wird der Aufbau der Studie und ihre Ergebnisse beschrieben. Ziel dieser Studie war es, Antworten auf die folgenden Fragen zu finden:

- Ist es für den Benutzern hilfreich, ein durchsuchbares Archiv seiner gelesenen Texten zu besitzen?
- Welchen Effekt hat die Quantifizierung der im Archiv gespeicherten Texte auf den Benutzer?
- Hilft die Präsentation einer Wortwolke der gelesenen Texte dem Benutzer, sich an deren Inhalt zu erinnern?

4.1. Teilnehmer

Die Versuchspersonen haben sich freiwillig zur Teilnahme bereit erklärt und wurden zufällig ausgewählt.

An der Studie haben 17 Versuchspersonen teilgenommen. Davon waren 11 in der Universität anwesend, die restlichen 6 haben online an der Studie partizipiert. Der Teilnehmerkreis bestand aus 12 Männern und 5 Frauen.

Das Alter der teilnehmenden Personen erstreckte sich von 18 bis 28 Jahren. Das durchschnittliche Alter betrug 23,8 Jahre bei einer Standardabweichung von 3,2.

Unter den Versuchspersonen befanden sich 9 Studentinnen/Studenten, 4 Absolventen einer Hochschulausbildung und 4 Personen mit dem Bildungsabschluss der mittleren Reife.

Aus dem Umfeld der Informatik kamen 10 Teilnehmer, die restlichen stammten aus sehr unterschiedlichen Berufsfeldern.

Ein Teilnehmer hat die Studie nicht beendet und ist daher in den weiteren Ergebnissen nicht berücksichtigt worden.

4.2. Durchführung

Zunächst mussten die Teilnehmer die Einverständniserklärung unterschreiben und anschließend den Vorgesprächsfragebogen ausfüllen, in dem allgemeine Daten und die Grundeinstellung abgefragt wurden.

Anschließend bekamen sie den Link zur Teilnahme am Beta-Test des Reading Trackers, woraufhin sie die App über den Play Store von Google installieren konnten. Nach dem Abschluss der Installation mussten sich die Teilnehmer im Reading Tracker registrieren und den Tracker Service in den Einstellungen aktivieren.

Um die richtige Einrichtung des Reading Trackers zu testen, wurden die Versuchspersonen nun dazu aufgefordert einen Text zu lesen und anschließend das Reading Archive zu öffnen um die Quantifizierungen und die Wortwolke zu betrachten.

Weiterhin sollten die Probanden ihr Handy die nächsten 5 Tage normal nutzen und mindestens einmal täglich die Zusammenfassung ihrer Aktivitäten betrachten.

Zum Abschluss sollten die Teilnehmer noch einen Fragebogen ausfüllen, mit dem sie die Nutzung des Reading Trackers und die Auswirkung desselben auf sich einschätzen sollten. Danach wurde der Reading Tracker wieder von ihren Geräten entfernt.

Die Einverständniserklärung und die beiden verwendeten Fragebögen finden sich im Anhang.

4.3. Methode

Die Studie kann als Feldstudie klassifiziert werden, da die Versuchspersonen den Reading Tracker in ihrem gewohnten Umfeld, außerhalb eines Labors, in ihrem Alltag einsetzten.

Die statistische Auswertung der Fragebögen erfolgte in einer Excel-Tabelle. Dabei wurden die Antworten soweit möglich operationalisiert und jeweils das Minimum, das Maximum, der Mittelwert und die Standardabweichung berechnet.

VPn bedeutet in diesem Kontext Versuchspersonennummer, Min steht für Minimum, Max für Maximum, MW für Mittelwert und SD für Standartabweichung.

Variable	Text der Frage
var 1	Ich lese hauptsächlich auf digitalen Medien
var 2	Ich kann mir gut merken, was ich gelesen habe
var 3	Mir fällt es leicht Texte auch im Nachhinein wiederzufinden
var 4	Ich habe ein guten Überblick darüber, was ich alles gelesen habe
var 5	Ich lese (digital) hauptsächlich mittels
var 6	Die Benutzeroberfläche der App ist einfach zu bedienen
var 7	Die Benutzeroberfläche der App ist optisch ansprechend gestaltet
var 8	Ich finde es hilfreich zu wissen, wieviel ich lese
var 9	Ich finde es hilfreich zu wissen, was ich lese
var 10	Ich finde es hilfreich zu wissen, wieviel ich in welcher Sprache lese
var 11	Ich finde es hilfreich zu wissen, wann ich wieviel lese
var 12	Ich lese bewusster, seit ich die App benutze
var 13	Ich lese mehr, seit ich die App benutze
var 14	Die App hilft mir dabei, mich daran zu erinnern, was ich gelesen habe
var 15	Die App hilft mir dabei, mir bewusst zu machen, wie viel ich lese
var 16	Die Suchfunktion ist sehr nützlich, um Texte wieder zu finden, von denen mir nur noch Teile bekannt waren
var 17	Die Suchfunktion ist sehr nützlich, um ein Überblick dafür zu bekommen, was ich alles zu einem Thema gelesen habe
var 18	Ich habe die App benutzt um Fakten nachzuschlagen, die ich gelesen hatte
var 19	Ich hatte den Eindruck das Archive umfasste alles was ich gelesen habe
var 20	Ich habe kein Problem damit einen Onlineservice zur Verwaltung meiner Texte zu verwenden
var 21	Ich hatte Bedenken gewisse Apps zu benutzen während der Tracker aktiv war
var 22	Ich habe das Speichern von Texten für sehr sensible Apps blockiert
var 23	Ich habe die Löschfunktion von einzelen Einträgen verwendet
var 24	Das hat mir besonders gut gefallen
var 25	Das hat mir nicht so gut gefallen
var 26	Ich bin auf folgende Probleme gestoßen
var 27	Folgende Funktionalität hat mir noch gefehl
var 28	Sonstige Kommentare
var 29	Anzahl der Entries je Versuchsteilnehmer
var 30	Anzahl der archivierten Wörter je Versuchsteilnehmer

Tabelle 4.1.: Variablen Auflistung

VPn	var 1	var 2	var 3	var 4	var 6	var 7	var 8	var 9	var 10	var 11	var 12
2	3	2	2	4	1	1	3	4	3	2	5
3	1	2	2	2	1	2	2	3	2	4	2
4	1	2	4	2	3	2	1	2	1	1	2
5	2	3	2	3	1	1	3	3	3	3	2
6	4	1	3	2	1	1	2	3	5	2	3
7	1	2	1	2	1	2	4	4	5	4	5
8	1	2	2	2	2	3	1	3	2	3	3
9	1	1	3	2	2	4	3	3	2	5	3
10	3	4	5	4	2	2	4	2	4	4	4
11	3	3	1	2	1	2	3	3	3	3	3
12	3	5	4	3	1	1	2	1	1	1	3
13	1	2	3	3	2	1	3	2	2	4	4
14	1	1	3	2	2	1	2	1	3	2	3
15	1	2	1	1	2	2	3	4	3	4	5
16	1	1	2	3	3	4	5	5	5	5	5
17	2	2	3	2	1	1	3	2	4	4	2
Min	1	1	1	1	1	1	1	1	1	1	2
Min	4	5	5	4	3	4	5	5	5	5	5
MW	1,81	2,19	2,56	2,43	1,63	1,89	2,75	2,81	3	3,18	3,38
SD	1,01	1,07	1,12	0,79	0,70	0,99	1,03	1,07	1,27	1,24	1,11

Tabelle 4.2.: Auswertung der operationalisierbaren Fragen 1 bis 4 und 6 bis 12

Die Variablen 1 bis einschließlich 4 und 6 bis einschließlich 23 sind kodiert in

- 1=trifft zu
- 2=trifft eher zu
- 3=neutral
- 4=trifft eher nicht zu
- 5=trifft nicht zu

Für Variable 5 wird die absolute Häufigkeit der einzelnen Nennungen angegeben.

Die Variablen 24 bis einschließlich 28 sind nicht operationalisierbar, sondern zählen nur Aussagen der Versuchspersonen auf.

Die Variablen 29 und 30 geben absolute Häufigkeiten an. Über die MySQL-Datenbank wurden die Werte je Teilnehmer ermittelt und daraus die unten stehenden Diagramme erstellt.

VPn	var 13	var 14	var 15	var 16	var 17	var 18	var 19	var 20	var 21	var 22	var 23
2	5	1	1	1	2	5	1	2	5	4	5
3	2	2	2	3	1	3	3	4	2	1	5
4	1	2	2	1	1	3	2	3	4	5	5
5	5	3	2	3	3	4	2	4	3	1	3
6	4	3	3	1	3	4	2	1	1	1	5
7	5	1	1	2	4	5	4	2	1	1	5
8	4	4	3	3	2	4	3	2	1	1	5
9	1	1	5	3	2	2	2	2	3	5	5
10	4	2	4	2	2	2	5	4	1	1	5
11	3	3	3	3	3	3	5	1	1	5	5
12	1	1	2	1	1	3	3	1	5	5	5
13	4	3	3	3	2	4	2	2	4	5	5
14	3	2	3	3	2	3	2	4	3	4	2
15	5	2	4	1	2	5	2	3	5	5	5
16	5	1	2	3	3	5	5	5	1	1	5
17	4	2	3	2	2	5	3	2	1	5	5
Min	1	1	1	1	1	2	1	1	1	1	3
Min	5	4	5	3	4	5	5	5	5	5	5
MW	3,43	2,06	2,69	2,19	2,19	3,75	2,88	2,63	2,56	3.13	4,69
SD	1,46	0,90	1,04	0,88	0.81	1,03	1,22	1,22	1,58	1.90	0,85

Tabelle 4.3.: Auswertung der operationalisierbaren Fragen 13 bis 23

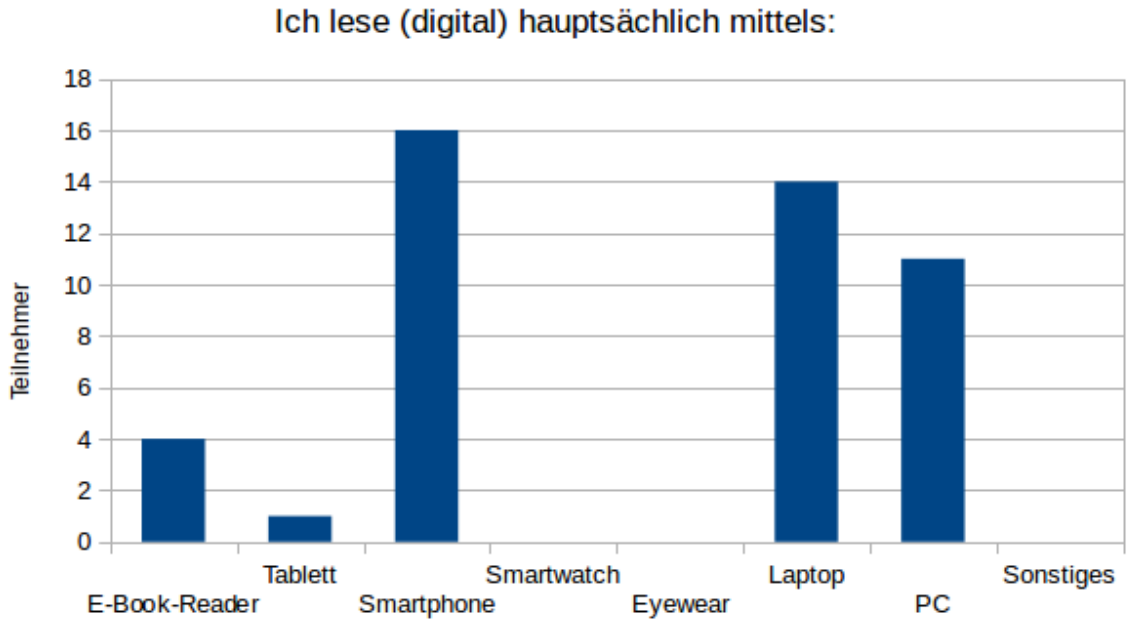

Abbildung 4.1.: Auflistung der Geräte

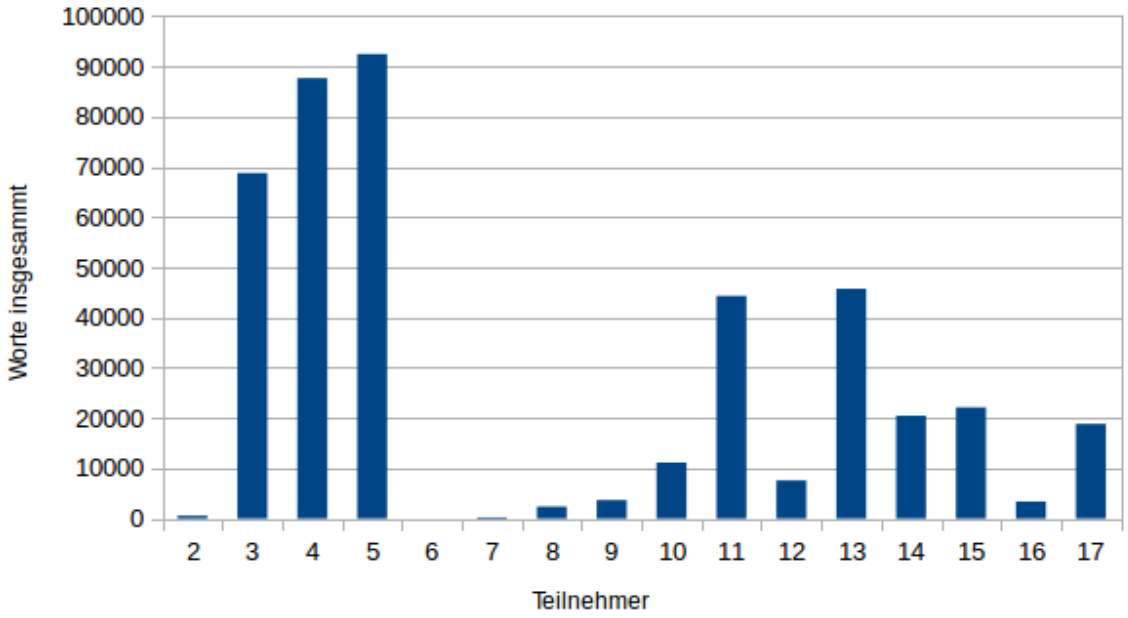

Abbildung 4.2.: Worte pro Teilnehmer

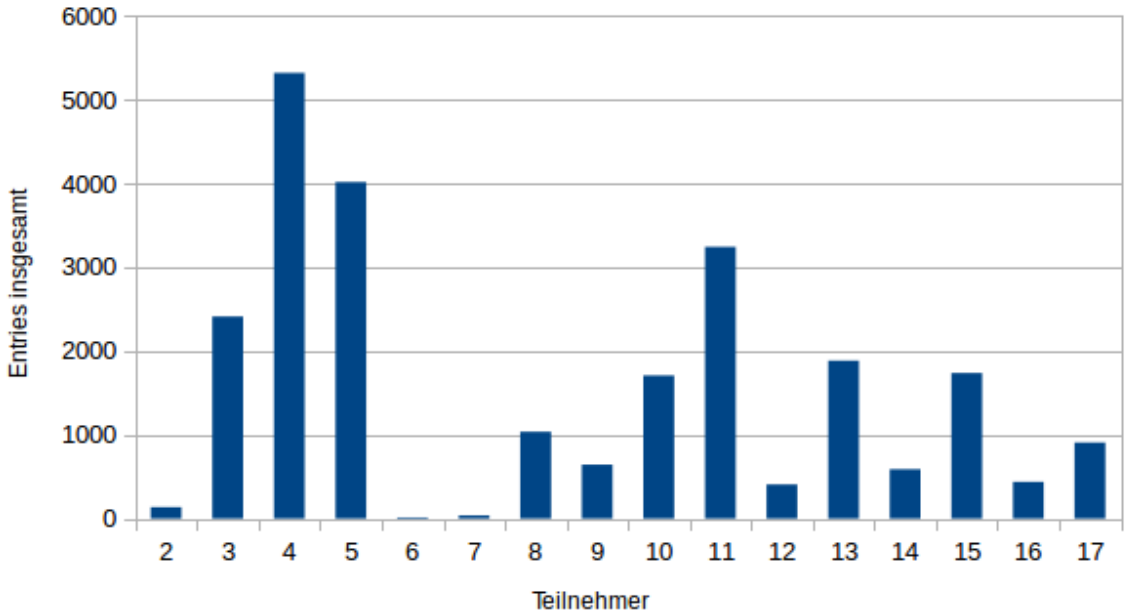

Abbildung 4.3.: Entries pro Teilnehmer

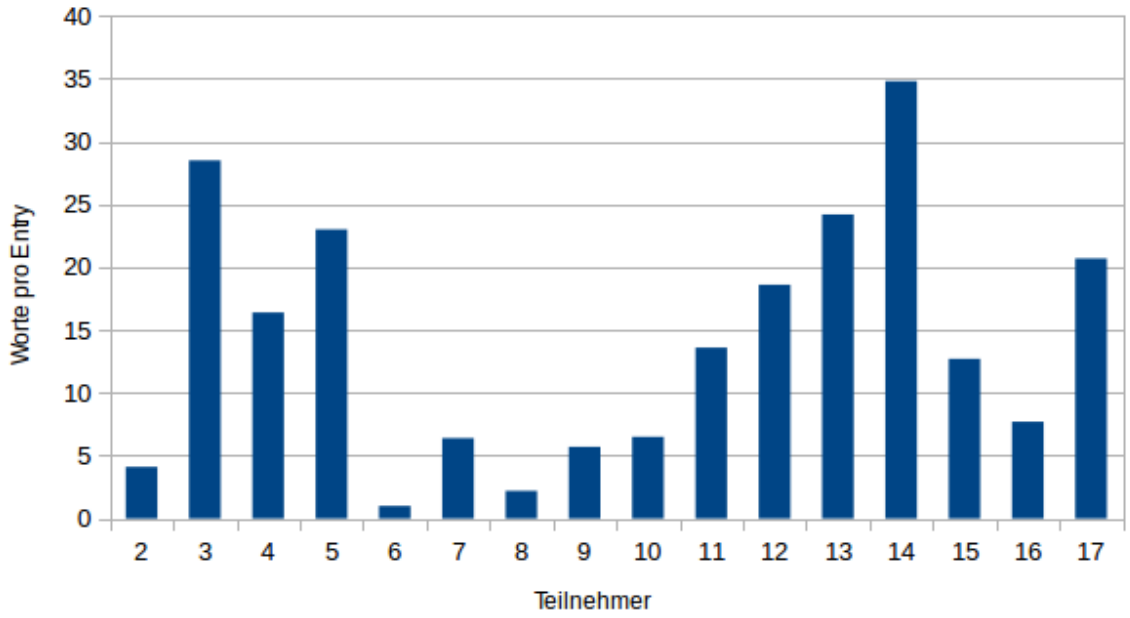

Abbildung 4.4.: Worte pro Entry

App	Worte
com.whatsapp	176796
com.facebook.katana	91119
com.netbiscuits.kicker	17159
com.amazon.mShop.android	16058
com.htc.launcher	13409
com.viber.voip	12846
com.google.android.gm	8918
com.google.android.youtube	7888
com.facebook.orca	6256
com.ebay.mobile	4004
Summe der Worte der gelisteten Apps	354453
Summe der Worte von allen Apps	429120

Tabelle 4.4.: Worte pro App

Ebenfalls aus der Datenbank extrahiert wurde die Anzahl der erfassten Wörter pro App. in der Tabelle 4.4 wurden die 10 Apps mit den meisten erfassten Wörtern abgebildet.

4.4. Ergebnisse

In diesem Kapitel werden die Ergebnisse der Studie und ihre Bedeutung besprochen.

Wie man in Abbildung 4.1 entnehmen kann benutzen die Versuchspersonen hauptsächlich ihr Smartphone, Laptop oder PC zum lesen. E-Book-Reader sowie Tabletts sind relativ gering verbreitet und Smartwatches oder Eyewear kommt gar nicht zum Einsatz. Da mehr als die Hälfte der Versuchspersonen noch im Studium sind kann man annehmen, dass dedizierte Geräte zum Lesen finanziell noch nicht tragbar waren. Dies Vermutung wird zusätzlich dadurch gestützt, dass die Versuchspersonen, welche ein abgeschlossenes Hochschulstudium aufweisen eher mehr Geräte benutzen.

Wie man aus den Tabellen 4.2 und 4.3 entnehmen kann sind die meisten Versuchspersonen der Meinung vor allem auf digitalen Medien zu lesen. Sie nehmen in der Regel an, dass sie sich Texte relativ gut merken können und nicht sehr überzeugt Texte auch im Nachhinein wiederfinden könnten. Sie nehmen an, dass sie einen einigermaßen guten Überblick darüber haben, was sie so gelesen haben.

Die App wurde als intuitiv bedienbar bewertet und kam auch optisch gut bei den Probanden an. Der Quantifizierung der gelesenen Texte durch das Reading Archive gegenüber waren die Versuchsteilnehmer relativ zurückhaltend und schätzen den Sinn dieser Informationen eher gering ein. Auch glauben sie nicht, dass das benutzen der App Auswirkungen auf ihr Leseverhalten hat.

Im Gegensatz dazu wurde der Einfluss der App auf das Erinnerungsvermögen positiv eingeschätzt. Zusätzlich waren sie auch der Meinung, dass die App das Bewusstsein dafür, wie viel sie lesen, erhöht. Ebenfalls sehr positiv wurde der Einfluss der Suchfunktion eingeschätzt.

Die wenigsten Benutzer gaben jedoch an, die Suchfunktion aktiv benutzt zu haben um etwas nachzuschlagen. Als Grund hierfür kann man annehmen, dass dies schlicht daran liegt, das viele die Suchfunktion nicht gefunden haben. Dies wurde oft als Rückmeldung angegeben.

Der Umfang der erfassten Texte im Archiv wurde sehr unterschiedlich bewertet und wurde im Mittel nur leicht positiv bewertet.

Der Frageblock zum Datenschutz, welcher in die Variablen 20 bis 24 kodiert wurde, offenbart eine tiefe Spaltung der Versuchspersonengruppe. Manche Versuchspersonen haben absolut keine Bedenken einen Onlineservice zum Speichern ihrer Texte zu benutzen, andere hingegen hatten massive Bedenken. Obwohl der Mittelwert relativ in der Mitte des Spektrums liegt, kann man anhand der Standardabweichung klar erkennen, das vor allem die Extremen ausgeprägt waren. Selbiges gilt auch für die Bedenken gewisse Apps zu nutzen, während der Reading Tracker aktiv war. Auch bei der Frage nach der Verwendung der Ignorierfunktion ist dies zu beobachten. Hier war die Standardabweichung mit 1.9 am größten.

Die Löschfunktion für einzelne Beiträge wurde in der Studie anscheinend nur von einer einzigen Person wirklich verwendet.

Im Freitext wurde die Gestaltung der App, sowie die Quantifizierung der gelesenen Texte und die Wortwolke sehr positiv hervorgehoben. Kritisiert wurde, vor allem dass viele Apps nicht korrekt ausgelesen wurden und die App gelegentlich abstürzte. Die Abstürze scheinen jedoch nur manche Android-Geräte zu betreffen.

Einige Versuchsteilnehmer hatten die Suchfunktion nicht gefunden, bei denen, die sie Gefunden hatten kam sie jedoch sehr gut an.

Am meisten wünschten sich die Versuchsteilnehmer eine bessere Unterstützung für Browser und ein interaktives Tutorial für die App.

Anhand der Abbildung 4.2 und Abbildung 4.3 ist gut zu erkennen, das einige wenige Versuchspersonen für die meisten erzeugten Texte verantwortlich waren. Der Tabelle 4.4 kann man entnehmen, dass die 10 Apps mit den meisten extrahierten Texten zusammen für etwa 82% aller Texte verantwortlich sind. Dabei entfallen etwa 42% allein auf WhatsApp.

5. Diskussion

In diesem Kapitel werden die Ergebnisse der Studie, wie in Abschnitt 4.4 vorgestellt auf ihre Validität untersucht und ihre Bedeutung für den Reading Tracker und das Reading Archive erläutert.

Durch die Studie kann belegt werden, dass das Konzept des Reading Archive in Verbindung mit dem Reading Tracker funktioniert. Der implementierte Prototyp hat trotz einigen, in der Studien offenbarten, Schwachstellen gezeigt, dass es möglich ist auf diesem Weg Texte auszulesen und diese den Benutzern aufbereitet zur Verfügung zu stellen.

Die Benutzeroberfläche der App sowie die Darstellung der Wortwolke kamen gut bei den Versuchsteilnehmern an und stellen eine gute Basis für weitere Studien dar.

Die Tatsache, dass einige Apps besser durch den Reading Tracker ausgelesen werden können, als andere, kann darauf zurück geführt werden, dass nicht alle Apps die nötigen Schnittstellen der Accessibility-API von Android implementiert haben. Die Apps, die dabei eine Vorreiterrolle spielen, wie zum Beispiel WhatsApp oder TextSecure, liefern sehr gute Ergebnisse.

Es ist davon auszugehen, dass mit der Zeit mehr Apps diese Schnittstellen implementieren werden und so ist zu hoffen, dass in Zukunft eine besseren Abdeckung aller gelesener Texte erzielt werden kann.

Interessant war zum einen die relativ gute Bewertung der Suchfunktion durch die Versuchsteilnehmer, obwohl einige diese gar nicht gefunden und benutzt hatten, wie man aus den Freitexten schließen kann. Es ist hierbei zu hoffen, das die Versuchspersonen, welche die Suchfunktion benutzt haben diese durchwegs gut bewertet haben und selbst die, die sich nicht gefunden hatten, sie als neutral bewertet haben. Es ist allerdings zu befürchten, dass die Versuchspersonen die Frage nach der Suchfunktion durch ihre Erwartungshaltung positiver bewertet haben.

Die meisten Versuchsteilnehmer gaben an, dass die App ihnen dabei hilft, sich besser zu erinnern und sich bewusst zu machen, wie viel sie lesen. Dabei wird die Wortwolke, im Vergleich zur Anzeige der gelesenen Wörter und DIN A4 Seiten, von den Versuchsteilnehmern eindeutig bevorzugt. Diese Aussagen sind jedoch sehr subjektive und nicht durch operationalisierbare Daten gestützt. Die Wortwolke könnte von den Versuchspersonen besser beurteilt worden sein, weil sie auch schon mit einer geringen Anzahl an Worten funktioniert.

Durch relativ geringe Anzahl von Teilnehmern an der Studie und die Studiendauer von nur 5 Tagen sorgen zudem dafür, dass die gewonnen Ergebnisse nur eine geringe Aussagefähigkeit besitzen.

Ein Problem dieser Studie waren auch die Datenschutzbedenken potentieller Studienteilnehmer, welche auch zum Teil für die relativ geringe Anzahl an Studienteilnehmern verantwortlich sind. Die Rückmeldungen auf die Einladung zur Studie waren ernüchternd gering. Die meisten Studienteilnehmer konnten nur durch ein persönliches Gespräch zur Teilnahme an der Studie überzeugt werden. Die dabei am meisten geäußerten Bedenken bezogen sich darauf, ob die übertragenen Texte von jemand gelesen oder ausgewertet würden.

Hier stellt sich die Frage, ob das Konzept mit der serverseitigen Verarbeitung der Benutzerdaten in der Praxis akzeptiert werden würde.

6. Ausblick

Um belastbare Zahlen für die Fragestellung der Wirkung des Reading Archives auf seine Benutzer zu bekommen bieten sich verschiedene Möglichkeiten an, wobei auf eine größere Anzahl an Versuchspersonen und eine längere Studiendauer geachtet werden sollte.

6.1. Laborstudie

Man könnte unter Rückgriff auf das Reading Archive und den Reading Tracker eine Laborstudie vorbereitet werden.

Diese Laborstudie würde eine zusätzlich eine App benötigen, welche gut vom Reading Tracker ausgelesen werden kann und den Studienleitern die möglichkeit bietet der Versuchspersonen regelmäßig Texte anzeigt zu lassen, welche ihnen gelesen werden müssten. Einen Tag später müssten die Versuchspersonen dann Verständnisfragen zum dem gelesenen Text beantworten.

Dies sollteüber mehre Wochen hinweg durchgeführt werden. In der ersten Hälfte dieser Laborstudie würde dabei auf die Verwendung des Reading Trackers und des Reading Archives verzichtet werden. In der zweiten Hälfte der Laborstudie würden diese schließlich aktiviert werden und von den Versuchspersonen täglich genutzt werden. Durch die Auswertung der Verständnisfragen könnte man nun belastbare Zahlen für die Wirkungen des Reading Archives auf die Versuchspersonen bekommen.

6.2. Erweiterte Felstudie

Alternativ oder zusätzlich dazu könnte eine weitere Feldstudie mit dem Reading Tracker und dem Reading Archive durch geführt werden.

Hierfür sollte der Reading Tracker so erweitert werden, dass er mit root-Rechten ausgestattet, dazu in der Lage wäre Screenshots zu erstellen, während andere Apps im Vordergrund sind.

Aus diese Screenshots könnten mit Hilfe eines Texterkennungsalgorithmus die vom Benutzer gelesenen Texte extrahiert werden. Dadurch wäre es möglich, die Beschränkungen, denen der Reading Tracker derzeit noch unterliegt, zu umgehen.

Das Problem an dieser Studie wäre allerdings genügend Versuchspersonen zu finden, die zum einen an dieser Studie teilnehmen würden und zum anderen auch ein gerootetes Android-Smartphone besitzen. Alternativ könnten auch alle Versuchspersonen für die Studie mit entsprechenden Geräten versorgt werden.

6.3. Praxis Erwägungen

Um das Reading Archive außerhalb von wissenschaftlichen Studien für die Benutzer attraktiv zu machen, müsste überlegt werden, wie man die Datenschutzbedenken der Benutzer verringern könnte.

Eine Möglichkeit hierfür wäre die Analyse der Texte lokal auf dem Smartphone auszuführen, so dass die Benutzer nicht befürchten müssen, dass jemand anderes Zugriff auf ihre Daten hat. Optional könnte man zusätzlich die verschlüsselten Daten über einen Server zwischen verschiedenen Geräten synchronisieren.

A. Anhang

Auf den folgenden Seiten sind die Einverständniserklärung und die beiden Fragebögen zu finden, welche bei der Studie verwendet wurden.

Studie: Reading Tracker

Universität Stuttgart

Teilnehmernummer: _____

Einverständniserklärung

Vielen dank dafür, dass sie an dieser Studie teilnehmen.

In dieser Studie geht es darum zu Untersuchen welche Effekt es auf den Menschen hat wenn man:

- weiß wie viel man den Tag über so gelesen hat
- eine Wortwolken der am Tag gelesenen Texte sehen kann
- alle gelesenen Texte durchsuchen kann

Wir werden uns dabei auf die Auswirkungen auf das Gedächtnis und die Lesegewohnheiten beschränken.

Die Android-App wird alle Texte, die sie lesen zu unserem Server übertragen, wo nur sie diese abrufen und durchsuchen können. Auf Basis dieser Texte wird eine Wortwolke erstellt und ihnen in der App angezeigt. Wir werden ihre Texte nicht lesen, aber überwachen, wie sie darauf zugreifen und wie sie dieses System nutzen.

Sie können jederzeit aus der Studie aussteigen. Die Teilnahme ist vollständig Freiwillig. Wenn sie weitere Fragen haben, zögern sie nicht mich zu Fragen:

Victor Riempp
Email: victor.riempp@gmail.com
Universität Stuttgart
Mensch-Computer-Interaktion

Studie: Reading Tracker

Universität Stuttgart

Teilnehmernummer: _____

Einverständniserklärung

- ❑ Ich habe den Text auf dem ersten Blatt gelesen und verstanden.
- ❑ Ich hab den Sinn dieser Studie verstanden und erkläre mich bereit Teilzunehmen.
- ❑ Ich habe verstanden, dass ich meine Teilnahme jederzeit abbrechen kann.
- ❑ Ich stimme zu und erlaube es, dass die auf meinem Smartphone angezeigten Texte aufgezeichnet werden.
- ❑ Ich bin mir bewusst, dass ich der einzige bin, der diese Daten einsehen kann und dass sie automatisch 6 Wochen nach dem Ende der Studie gelöscht werden

Unterschrift

Teilnehmer _____ Datum _____

Durchführender _____ Datum _____

First Questionaire

1. General

Gender:

male	female	won't say
☐	☐	☐

Age: _____

Highest educational archievement: _____

Profession / Background: _____

2. Wie zutreffend sind für Sie die folgenden Aussagen?

	trifft zu	trifft eher zu	neutral	trifft eher nicht zu	trifft nicht zu
Ich lese hauptsächlich auf digitalen Medien	☐	☐	☐	☐	☐
Ich kann mir gut merken, was ich gelesen habe	☐	☐	☐	☐	☐
Mir fällt es leicht Texte auch im Nachhinein wiederzufinden	☐	☐	☐	☐	☐
Ich habe ein guten Überblick darüber, was ich alles gelesen habe	☐	☐	☐	☐	☐

3. Ich lese (digital) hauptsächlich mittels:

E-Book Reader	☐
Tablet	☐
Smartphone	☐
Smartwatch	☐
Eyewear	☐
Laptop	☐
PC	☐
Sonstiges	☐

Abschlussfragebogen

1. Aussehen und Bedienung

	trifft zu	trifft eher zu	neutral	trifft eher nicht zu	trifft nicht zu
Die Benutzeroberfläche der App ist einfach zu bedienen	☐	☐	☐	☐	☐
Die Benutzeroberfläche der App ist optisch ansprechend gestaltet	☐	☐	☐	☐	☐

2. Allgemeine Einstellung

	trifft zu	trifft eher zu	neutral	trifft eher nicht zu	trifft nicht zu
Ich finde es hilfreich zu wissen, wieviel ich lese.	☐	☐	☐	☐	☐
Ich finde es hilfreich zu wissen, was ich lese.	☐	☐	☐	☐	☐
Ich finde es hilfreich zu wissen, wieviel ich in welcher Sprache lese.	☐	☐	☐	☐	☐
Ich finde es hilfreich zu wissen, wann ich wieviel lese.	☐	☐	☐	☐	☐

3. Auswirkungen auf das Leseverhalten

	trifft zu	trifft eher zu	neutral	trifft eher nicht zu	trifft nicht zu
Ich lese bewusster, seit ich die App benutze.	☐	☐	☐	☐	☐
Ich lese mehr, seit ich die App benutze.	☐	☐	☐	☐	☐

4. Auswirkungen auf das Erinnerungsvermögen

	trifft zu	trifft eher zu	neutral	trifft eher nicht zu	trifft nicht zu
Die App hilft mir dabei, mich daran zu erinnern, was ich gelesen habe.	☐	☐	☐	☐	☐
Die App hilft mir dabei, mir bewusst zu machen, wieviel ich lese.	☐	☐	☐	☐	☐
Die Suchfunktion ist sehr nützlich, um Texte wieder zufinden, von denen mir nur noch Teile bekannt waren.	☐	☐	☐	☐	☐
Die Suchfunktion ist sehr nützlich, um ein Überblick dafür zu bekommen, was ich alles zu einem Thema gelesen habe.	☐	☐	☐	☐	☐
Ich habe die App benutzt um Fakten nachzuschlagen, die ich gelesen hatte.	☐	☐	☐	☐	☐
Ich hatte den Eindruck das Archive umfasste alles was ich gelesen habe	☐	☐	☐	☐	☐

5. Datenschutz

	trifft zu	trifft eher zu	neutral	trifft eher nicht zu	trifft nicht zu
Ich habe kein Problem damit einen Onlineservice zur Verwaltung meiner Texte zu verwenden	☐	☐	☐	☐	☐
Ich hatte Bedenken gewisse Apps zu benutzen während der Tracker aktiv war	☐	☐	☐	☐	☐
Ich habe das Speichern von Texten für sehr sensible Apps blockiert	☐	☐	☐	☐	☐
Ich habe die Löschfunktion von einzelen Einträgen verwendet	☐	☐	☐	☐	☐

6. Das hat mir besonders gut gefallen:

7. Das hat mir nicht so gut gefallen:

8. Ich bin auf folgende Probleme gestoßen:

9. Folgende Funktionalität hat mir noch gefehlt:

10. Sonstige Kommentare:

Literaturverzeichnis

[AS68] Atkinson, Richard C. ; Shiffrin, Richard M.: Human memory: A proposed system and its control processes. In: *The psychology of learning and motivation* 2 (1968), S. 89–195 (Zitiert auf Seite 11)

[Bad00] Baddeley, Alan: The episodic buffer: a new component of working memory? In: *Trends in cognitive sciences* 4 (2000), Nr. 11, S. 417–423 (Zitiert auf den Seiten 12 und 13)

[BEA09] Baddeley, AD ; Eysenck, MW ; Anderson, M: *Memory*. Psychology Press, 2009 (Zitiert auf Seite 13)

[BH74] Baddeley, Alan D. ; Hitch, Graham J.: Working memory. In: *The psychology of learning and motivation* 8 (1974), S. 47–89 (Zitiert auf Seite 12)

[BHO11] Bostock, Michael ; Heer, Jeffrey ; Ogievetsky, Vadim: *D3.js*. http://d3js.org/. Version: 2011 (Zitiert auf Seite 28)

[Bri14] Briegleb, Volker: *Smartphones prägen den deutschen Mobilfunk-Markt*. http://www.heise.de/newsticker/meldung/Smartphones-praegen-den-deutschen-Mobilfunk-Markt-2117455.html. Version: 2014 (Zitiert auf Seite 9)

[Bui15] BuildWith: *jQuery Usage Statistics*. http://trends.builtwith.com/javascript/jQuery. Version: 2015 (Zitiert auf Seite 28)

[Bur15] Burke, Dave: *Android 5.1: Unwrapping a new Lollipop update*. http://officialandroid.blogspot.de/2015/03/android-51-unwrapping-new-lollipop.html. Version: 3 2015 (Zitiert auf Seite 19)

[Cro06] Crockford, Douglas: The application/json media type for javascript object notation (json). (2006). https://tools.ietf.org/html/rfc4627 (Zitiert auf Seite 16)

[Cro08] Crockford, Douglas: *JavaScript: The Good Parts: The Good Parts*. Ö'Reilly Media, Inc.", 2008 (Zitiert auf Seite 16)

[Dav14] Davies, Jason: *How the Word Cloud Generator Works*. https://www.jasondavies.com/wordcloud/about/. Version: 2014 (Zitiert auf Seite 28)

[Dav15] Davies, Jason: *D3 Word Cloud*. https://github.com/jasondavies/d3-cloud. Version: 2015 (Zitiert auf Seite 28)

[DES14] DESTATIS: *Ausstattung privater Haushalte mit Informations- und Kommunikationstechnik - Deutschland*. https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/EinkommenKonsumLebensbedingungen/AusstattungGebrauchsguetern/Tabellen/Infotechnik_D.html. Version: 2014 (Zitiert auf Seite 9)

[DG92] Deleuze, Gilles ; Guattari, Fèlix: *Tausend Plateaus: Kapitalismus und Schizophrenie / Kapitalismus und Schizophrenie*. 1992 (Zitiert auf Seite 14)

[Dja15a] Django Software Foundation: *Django Documentation (Version 1.8)*. https://docs.djangoproject.com/en/1.8/. Version: 2015 (Zitiert auf den Seiten 7, 23 und 27)

[Dja15b] Django Software Foundation: *Django (Version 1.8)*. https://djangoproject.com/. Version: 2015 (Zitiert auf Seite 23)

[ECM97] ECMA: 262: Ecmascript language specification. In: *ECMA (European Association for Standardizing Information and Communication Systems), pub-ECMA: adr*, 1 (1997). http://www.ecma-international.org/publications/standards/Ecma-262-arch.htm (Zitiert auf Seite 16)

[ECM15] ECMA: 262: Ecmascript language specification. In: *ECMA (European Association for Standardizing Information and Communication Systems), pub-ECMA: adr*, 6 (2015). http://www.ecma-international.org/publications/standards/Ecma-262.htm (Zitiert auf Seite 16)

[EK00] Eysenck, Michael W. ; Keane, Mark T.: *Cognitive psychology: A student's handbook*. Taylor & Francis, 2000 (Zitiert auf Seite 13)

[Fei09] Feinberg, Jonathan: *Wordle-beautiful word clouds*. 2009 (Zitiert auf den Seiten 14, 28 und 39)

[Goo15a] Google: *Android Developers API Guide*. http://developer.android.com/guide/index.html. Version: 2015 (Zitiert auf den Seiten 20 und 21)

[Goo15b] Google: *Android Developers Training*. http://developer.android.com/training/index.html. Version: 2015 (Zitiert auf den Seiten 7, 18, 20 und 21)

[Goo15c] Google: *Android Interfaces and Architecture*. https://source.android.com/devices/. Version: 2015 (Zitiert auf Seite 20)

[GS85] Graf, Peter ; Schacter, Daniel L.: Implicit and explicit memory for new associations in normal and amnesic subjects. In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 11 (1985), Nr. 3, S. 501 (Zitiert auf Seite 12)

[Jam90] James, William: *1950. The principles of psychology.* 1890 (Zitiert auf Seite 11)

[Kö14] Költzsch, Tobias: *Android läuft auf fast 85 Prozent aller Smartphones.* http://www.golem.de/news/mobile-betriebssysteme-android-laeuft-auf-fast-85-prozent-aller-smartphones-1408-1 html. Version: 2014 (Zitiert auf den Seiten 18 und 19)

[Liu05] Liu, Ziming: Reading behavior in the digital environment: Changes in reading behavior over the past ten years. In: *Journal of documentation* 61 (2005), Nr. 6, S. 700–712 (Zitiert auf den Seiten 9 und 14)

[Mil56] Miller, George A.: The magical number seven, plus or minus two: some limits on our capacity for processing information. In: *Psychological review* 63 (1956), Nr. 2, S. 81 (Zitiert auf Seite 11)

[Mor08] Morrill, Dan: *Announcing the Android 1.0 SDK, release 1.* http://android-developers.blogspot.de/2008/09/announcing-android-10-sdk-release-1.html. Version: 9 2008 (Zitiert auf Seite 19)

[Neu15] Neumann, Alexander: *JavaScript und Java sind die populärsten Programmiersprachen auf GitHub.* http://www.heise.de/newsticker/meldung/JavaScript-und-Java-sind-die-populaersten-Programmiersprachen-auf-GitHub-2787516.html. Version: 2015 (Zitiert auf Seite 16)

[Pyt15] Python Software Foundation: *Python (Version 3.4.3).* http://www.python.org/. Version: 2015 (Zitiert auf Seite 15)

[Que15] Query Team: *jQuery.* http://jquery.com/. Version: 2015 (Zitiert auf Seite 28)

[Rea07] Reardon, Marguerite: *Google unveils cell phone software and alliance.* http://www.cnet.com/news/google-unveils-cell-phone-software-and-alliance/. Version: 2007 (Zitiert auf Seite 19)

[RPFR98] Reichle, Erik D. ; Pollatsek, Alexander ; Fisher, Donald L. ; Rayner, Keith: Toward a model of eye movement control in reading. In: *Psychological review* 105 (1998), Nr. 1, S. 125 (Zitiert auf Seite 13)

[SC98] Stanovich, KE ; Cunningham, AE: What reading does for the mind. In: *American Education Journal* (1998) (Zitiert auf Seite 14)

[Smi12] Smieh: *Anatomy Physiology of an Android.* https://upload.wikimedia.org/wikipedia/commons/a/af/Android-System-Architecture.svg. Version: 2012. – CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0) (Zitiert auf den Seiten 7 und 19)

[TLH+11] Thayer, Alexander ; Lee, Charlotte P. ; Hwang, Linda H. ; Sales, Heidi ; Sen, Pausali ; Dalal, Ninad: The imposition and superimposition of digital reading technology: the academic potential of e-readers. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* ACM, 2011, S. 2917–2926 (Zitiert auf Seite 14)

[Tul72] Tulving, Endel: Episodic and semantic memory 1. In: *Organization of Memory. London: Academic* 381 (1972), Nr. e402, S. 4 (Zitiert auf Seite 12)

[Twi15] Twitter: *Bootstrap*. http://getbootstrap.com/. Version: 2015 (Zitiert auf Seite 29)

[Ull04] Ullenboom, Christian: *Java ist auch eine Insel*. Galileo Press, 2004 (Zitiert auf Seite 15)

[Wen06] Wenz, Christian: *JavaScript und AJAX*. 7. Rheinwerk Computing, 2006 http://openbook.rheinwerk-verlag.de/javascript_ajax/ (Zitiert auf Seite 15)

Alle URLs wurden zuletzt am 1. 09. 2015 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift