Paul Picher, BSc

Mobile Applikation zur Förderung der Lesekompetenz bei Kindern

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Informatik

eingereicht an der

Technischen Universität Graz

Betreuer
Univ.-Doz. Dipl.-Ing. Dr.techn. Martin Ebner

Institut für Informationssysteme und Computer Medien

Graz, Februar 2015

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Danksagung

An dieser Stelle möchte ich mich bei meinem Betreuer, Univ.-Doz. Dipl.-Ing. Dr.techn. Martin Ebner bedanken, der es mir möglich gemacht hat, diese Masterarbeit zu verfassen. Danke für die Unterstützung während der Erstellung dieser Arbeit.

Besonderer Dank gilt meinen Eltern, die mir mein Studium ermöglicht haben und immer hinter mir gestanden sind. Weiters gilt mein Dank meinen Geschwistern mit Familie, Barbara mit Florian, Maximilian und Emma Sophie, Ursula mit Chris und David.

Außerdem gilt mein Dank allen, die mich bei der Erstellung dieser Masterarbeit unterstützt haben.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung eines Lesetrainers. Ein auf neuen Technologien basierendes System, das es ermöglicht, die Lesekompetenz von Kindern zu erfassen und automatisch auszuwerten. Dass, die Entwicklung eines solchen Lesetrainers wichtig ist zeigt sich tagtäglich in unseren Schulen.

Lesen ist eine der wichtigsten Kompetenzen, die ein Mensch im Laufe seines Lebens erlernt. So wie kein Mensch dem anderen gleicht, so verhält es sich auch mit der Geschwindigkeit des Lernens. Manche eignen sich neue Fähigkeiten oder neues Wissen schneller an als andere. In Österreich lernen Kinder im Alter von 6 Jahren das Lesen in der Volksschule. Durch die unterschiedlichen Lerngeschwindigkeiten ergibt sich jedoch ein Problem. Die Bildungseinrichtung hat durch den Unterricht im Klassenverband nicht immer die Zeit auf jedes Kind individuell einzugehen. Deswegen ist es sehr wichtig, dass eventuelle Defizite früh durch die Lehrerin bzw. den Lehrer erkannt werden und gezielte Maßnahmen gesetzt werden, um diese auszugleichen. Doch wie wird Lesekompetenz erfasst? Verschiedene Lesetests bieten ein diagnostisches Verfahren zur Feststellung von Grundfertigkeiten, die als Teil der Lesefähigkeit gelten.

Der in der vorliegenden Arbeit diskutierte Lesetrainer ermöglicht ein zentrales Verwalten verschiedener Lesetests und deren Ergebnisse mit den dazugehörigen Auswertungen. Dieses webbasierte Administrationstool ermöglicht auch die Verteilung der erstellten Lesetests an die Clientapplikationen. Diese dienen der Durchführung der Lesetests und sind in Form einer Applikation für das iPad von Apple Inc. und einer webbasierten Version für den Webbrowser vorhanden. Durch den fortschreitenden Einsatz neuer Technologien im Bildungswesen ist dies ein logischer Schritt für zeitgemäße Unterrichtsmethoden.

Abstract

The thesis at hand is concerned with the development of a reading-trainer. A system that is based on new technologies and that automatically records and evaluates reading competencies of children. The importance of such a system is important for several reasons.

First, reading is one of the most important skills humans learn in their life. Second, just as no human resembles another, nobody acquires reading skills at the same extent and speed as another one. Some people acquire skills and knowledge faster than others. However, since pupils start to learn reading in school in a class with many different learners and consequently learner types it cannot be reassured that every child gets the attention needed. Therefore, deficits in reading are often not detected early enough and therefore such children are deprived of the required measures to meet and compensate such deficits.

But how can one measure reading skills? These days various reading tests serve as a diagnostic procedure to measure basic skills that are considered as part of reading competency. The reading-trainer discussed in this thesis makes it possible to administrate various reading tests, their results and the according analysis. Further, the administration tool enables the distribution of the developed reading tests to the client application. The client applications serve as a tool to conduct reading tests and are available in form of an application for the Apple iPad as well as in the form of a web based application for the web browser. The implementation of such a reading-trainer is a consequence of the growing importance of emerging technology in education.

Inhaltsverzeichnis

1 Einleitung

Die Tatsache, dass Sie dies hier lesen können, beweist, dass Sie in Ihrem Leben die Fähigkeit zu Lesen erworben haben.
Lernen ist das zentrale Element in der Entwicklung des Menschen, sein ganzes Leben lang werden Fertigkeiten erlernt und eingesetzt. Kinder lernen ab dem Zeitpunkt, an dem sie das Licht der Welt erblicken. Angefangen von motorischen Fähigkeiten, über Fortbewegung bis hin zu Sprache und Schrift bis hin zur Fähigkeit der Lösung komplexer Problemstellungen[1] [2]. Alle diese Fertigkeiten beruhen auf Kompetenzen, die im Laufe eines Lebens erlernt werden. Ein bekanntes Sprichwort bringt dies auf den Punkt und bestätigt sich täglich selbst.

„Man lernt niemals aus!“

In der Beschreibung früher Hochkulturen wird die Existenz von Schriftsprache als notwendig angesehen.[Breidbach, 2014] Daraus folgt, dass die Fähigkeit lesen und schreiben zu können, essentiell für die Teilnahme am gesellschaftlichen Leben ist. In der heutigen Gesellschaft sind diese Fähigkeiten wichtiger denn je zuvor. Im Informationszeitalter geschieht die Kommunikation untereinander zu einem großen Teil in geschriebener Form.

In Österreich erlernt man im Normalfall im Alter zwischen 6 und 10 Jahren das Lesen und Schreiben, zusammen mit gleichaltrigen Kindern, in der Volksschule. So wie jeder Mensch individuell ist, so verhält es sich auch mit der Art des Lernens. Menschen lernen auf unterschiedliche Weise, manche erwerben neue Fähigkeiten schneller als andere, abhängig von der

[1] http://www.stern.de/gesundheit/kinderkrankheiten/12-entwicklung-von-kindern-vom-ersten-schrei-bis-zum-ersten-kuss-632726.html - letzter Aufruf 3.2.2015
[2] http://www.stangl.eu/psychologie/entwicklung/Kognitive-Entwicklung.shtml - letzter Aufruf 3.2.2015

Methode der Wissensvermittlung [Möller, 2013]. Dies führt dazu, dass in der Schule Leistungsunterschiede innerhalb der Klasse auftreten können. Das Ziel der Bildungseinrichtung besteht jedoch darin, dass mit Abschluss einer Schulstufe definierte Grundkompetenzen beherrscht werden sollen. Dies macht eine Feststellung der Leistungen der Schülerinnen und Schüler notwendig, um eventuelle Defizite früh zu erkennen und diesen durch individuelle Förderungen entgegen zu wirken. In dieser Arbeit wird die Lesefähigkeit genauer betrachtet und wie es möglich ist, diese zu bewerten und die gewonnen Erkenntnisse zu interpretieren.

Die Fortschritte in der Informationstechnologie sind allgegenwärtig. Überall die neuesten persönlichen Informationen zu erhalten und zu jedem Zeitpunkt erreichbar zu sein, gehört längst zum Alltag. Es geht soweit, dass es vielen Menschen nicht bewusst ist, dass viele Dinge des täglichen Lebens ohne Informationstechnologie nicht mehr möglich sind. Der Übergang der physischen Welt in die virtuelle ist fließend geworden.

Sehr viele traditionelle Abläufe wurden durch die Informationstechnologie vereinfacht oder gar gänzlich von ihr übernommen. Der Einsatz dieser Technologien in Schulen kann viele Bereiche des Unterrichts vereinfachen bzw. eine schnellere und einfachere Durchführung von bestimmen Aufgaben gewährleisten.

Bezogen auf die Lesekompetenz von Kindern bieten sich neue Technologien an, um damit ein individuelles Leistungsprofil der Schülerinnen und Schüler zu erstellen, welches es den lehrenden Personen möglich macht, dieses zu beurteilen und eventuelle Maßnahmen zu setzen.

Welche Anforderungen muss daher ein Informationssystem erfüllen, damit eine Erfassung eines Leistungsprofils möglich wird?

- Feststellung der Lesefähigkeit durch geeignete Überprüfungsverfahren (Tests).
- Einfache Durchführung der Testungen im Klassenverband.
- Intuitive Bedienung der Testgeräte durch Schülerinnen und Schüler.
- Individuelle Erstellung von Tests durch lehrende Personen, um eine Anpassung an die jeweilige Klasse zu erreichen.
- Zentrales Administrationstool für Bereitstellung und Auswertung.
- Gemeinschaftliche Aufgabensammlung, die ein Zusammenarbeiten von unterschiedlichen Schulen und Schulstufen fördert.
- Komplett automatisiertes Erfassen der Auswertungsgrößen der Tests.
- Automatisierte Auswertung und Aufbereitung der erzielten Leistungen.

Diese Arbeit befasst sich mit der Umsetzung eines Prototypen, der den Namen ***Lesetrainer*** erhalten hat. Dieser soll alle definierten Anforderungen erfüllen. Eine Untersuchung verschiedener Testverfahren führt zur Konkretisierung der zu erfassenden Fertigkeiten und deren Auswertbarkeit.
Basierend auf den fixierten Erfassungsgrößen und der gewünschten Auswertung, konnten geeignete Testverfahren ausgewählt werden. Um dem Anforderungsprofil des Lesetrainers gerecht zu werden, wurde ein Konzept für die Umsetzung definiert und deren Technologien vorgestellt.
Eine Dokumentation über die Realisierung des Prototypen gibt Aufschluss über die Funktionsweise des Systems. Die Evaluierung befasst sich mit einem Usabililtytest und einem Proof of Concept des Lesetrainers und den daraus gewonnenen Erkenntnissen. Eine anschließende Diskussion erlaubt es, auf Grund der Beobachtungen, mögliche Verbesserungen zu erörtern bzw. gibt einen Einblick auf zukünftige Erweiterungen.

2 Stand der Technik

Die Hauptaufgabe bei der Konzeptionierung des Lesetrainers ist das Verifizieren der Grunddefinition der Fähigkeit zu lesen. Grundlegend beruht die Lesekompetenz auf aufbauenden Fertigkeiten, diese sollten vom Lesetrainer überprüft werden. Die Entscheidung, welche erfasst und ausgewertet werden sollen, ist nicht trivial. Eine Übersicht der Definitionen von Lesekompetenz und die Untersuchung verschiedener bereits verfügbarer Lesetests, ermöglicht eine Festlegung auf die essentiellen Fertigkeiten und deren Überprüfbarkeit.

2.1 Lesekompetenz

2.1.1 OECD PISA Studie

Die OECD (Organisation für wirtschaftliche Zusammenarbeit und Entwicklung) führt die bekannte internationale Studie PISA (Programme for International Student Assessment)[1] durch, welche die Schulleistung der teilnehmenden Länder bewertet und vergleichbar macht.

> *„PISA untersucht, inwieweit Schülerinnen und Schüler gegen Ende ihrer Pflichtschulzeit Kenntnisse und Fähigkeiten erworben haben, die es ihnen ermöglichen, an der Wissensgesellschaft teilzuhaben. Schwerpunkt der jüngsten Erhebung ist Mathematik."*[1]

Die PISA Studie bewertet nicht das Beherrschen des Lehrplans, vielmehr überprüft sie, ob es den Schülerinnen und Schülern möglich ist, das in der Schule erworbene Wissen praktisch umzusetzen.[2]

[1] http://www.oecd.org/berlin/themen/pisa-internationaleschulleistungsstudiederoecd.htm letzter Aufruf 25.1.2015
[2] http://www.oecd.org/berlin/themen/pisa-hintergrund.htm - letzer Aufruf 25.1.2015

Hierbei definiert die OECD die Fähigkeit des Lesens wie folgt:

> *„Die Fähigkeit einer Person, geschriebene Texte zu verstehen, zu nutzen und über sie zu reflektieren und sich mit ihnen auseinanderzusetzen, um eigene Ziele zu erreichen, das eigene Wissen und Potenzial weiterzuentwickeln und aktiv am gesellschaftlichen Leben teilzunehmen."*[3]

Diese Definition besagt, dass es einer Person möglich sein muss, geschriebene Informationen zu verstehen, diese zu interpretieren und daraus Schlüsse zu ziehen. Jedoch bietet sie keine genaue Angabe darüber, über welche Fähigkeiten eine Person verfügen muss, um diese Kompetenz zu erwerben.

2.1.2 Österreichsiche Bildungssandards

In Österreich ist das BIFIE (Bundesinstitut für Bildungsforschung, Innovation & Entwicklung des österreichischen Schulwesens) verantwortlich für die Beobachtung des Schulsystems und das sogenannte Bildungsmonitoring.

> *„Das Bildungssystem wird vor allem hinsichtlich seiner Praxis und der erzielten Ergebnisse kontinuierlich beobachtet. Das BIFIE entwickelt und überprüft die neuen nationalen Bildungsstandards in der 4. Schulstufe in Deutsch und Mathematik sowie in der 8. Schulstufe in Deutsch, Mathematik und Englisch und ist für die Durchführung großer internationaler Schülerleistungsstudien (z.B. PISA, PIRLS, TIMSS) verantwortlich."*[4]

Das Ziel der Festlegung und Überprüfung der Bildungsstandards ist es, eine Verbindlichkeit im Schulsystem anzustreben, die sicherstellen soll, dass Schülerinnen und Schüler in ganz Österreich mit Abschluss der 4. bzw. der 8. Schulstufe die gleichen Grundkompetenzen erworben haben.[5]

[3] http://www.oecd.org/berlin/themen/pisa-kompetenzen.htm - letzter Aufruf 25.1.2015
[4] https://www.bifie.at/bifie/kernaufgaben - letzter Aufruf 25.1.2015
[5] https://www.bifie.at/bildungsstandards - letzter Aufruf 25.1.2015

Für die 4. Schulstufe definiert das BIFIE den Kompetenzbereich: Lesen - Umgang mit Texten und Medien wie folgt:

„1. *Die Lesemotivation bzw. das Leseinteresse festigen und vertiefen.*
2. *Über eine altersadäquate Lesefertigkeit und ein entsprechendes Leseverständnis verfügen.*
3. *Den Inhalt von Texten mit Hilfe von Arbeitstechniken und Lesestrategien erschließen.*
4. *Das Textverständnis klären und über den Sinn von Texten sprechen.*
5. *Verschiedene Texte gestaltend oder handelnd umsetzen.*
6. *Formale und sprachliche Gegebenheiten in Texten erkennen.*
7. *Literarische Angebote und Medien aktiv nutzen.*“[6]

Für die 8. Schulstufe werden die Bildungsstandards bezüglich des Kompetenzbereichs Lesen folgend definiert:

> *„Ausgehend von grundlegenden Lesefertigkeiten literarische Texte, Sachtexte, nichtlineare Texte (Tabellen, Diagramme) und Bild-Text-Kombinationen in unterschiedlicher medialer Form inhaltlich und formal erfassen und reflektieren.“*[7]

„1. *Ein allgemeines Verständnis des Textes entwickeln.*
2. *Explizite Informationen ermitteln.*
3. *Eine textbezogene Interpretation entwickeln.*
4. *Den Inhalt des Textes reflektieren.*“[7]

Die zwei Definition der Lesekompetenz des BIFIE und der OECD sind unterschiedlich detailreich. Das BIFIE definiert genauer, welche Fähigkeiten für die Erlangung der einzelnen Kompetenzstufen notwendig sind. Doch auch diese detaillierte Aufschlüsselung lässt es nicht zu, die grundlegenden Fertigkeiten die für das Lesen Voraussetzung sind, zu erkennen.

[6] https://www.bifie.at/system/files/dl/bist_d_vs_kompetenzbereiche_d4_2011-08-19.pdf - letzter Aufruf 25.1.2015
[7] https://www.bifie.at/system/files/dl/bist_d_sek1_kompetenzbereiche_d8_2011-01-02.pdf - letzter Aufruf 25.1.2015

2.2 Standardisierte Leistungstests

Zu den Basisfunktionalitäten des Lesetrainers gehört die Feststellung der Lesefähigkeit der einzelnen Schülerinnen und Schüler. Doch durch welche Methoden ist dies möglich? Standardisierte Tests ermöglichen eine objektive und vergleichbare Erfassung von Fähigkeiten.

> *„Ein Test ist ein wissenschaftliches Routineverfahren zur Untersuchung eines oder mehrerer empirisch abgrenzbarer Persönlichkeitsmerkmale mit dem Ziel einer möglichst quantitativen Aussage über den relativen Grad der individuellen Merkmalsausprägung.“*
> [Lienert & Raatz, 1998, Seite 1]

Ein Test muss gewisse Kriterien erfüllen, um als solcher zu gelten: [Lienert & Raatz, 1998, Seite 1]

- Ein Test muss wissenschaftlich begründet sein.
- Er muss routinemäßig, das bedeutet immer unter den gleichen festgelegten Bedingungen, durchgeführt werden.
- Das Ergebnis muss die Möglichkeit einer *„...relativen Positionsbestimmung des untersuchten Individuums innerhalb einer Gruppe von Individuen oder im Bezug auf ein bestimmtes Kriterium, z.B. ein Lernziel...“*[Lienert & Raatz, 1998, Seite 1] bieten.
- Um der Definition gerecht zu werden, muss der Test *„...bestimmte empirisch - also verhaltens- und erlebnisanalytisch, phänomenologisch und nicht rein begrifflich - abgrenzbare Eigenschaften, Verhaltensdispositionen, Fähigkeiten, Fertigkeiten oder Kenntnisse...“*[Lienert & Raatz, 1998, Seite 1] prüfen.

Diese Definitionen beschreiben die Anforderungen, die eine Untersuchung erfüllen muss, um im wissenschaftlichen Sinn als Test zu gelten. Diese beschreiben nicht, wie eine Vergleichbarkeit zwischen unterschiedlichen Gruppen von Probandinnen und Probanden erreicht werden kann. Eine solche wird über die Standardisierung erreicht.

> *„Standardisierte Tests müssen wissenschaftlich entwickelt, hinsichtlich der wichtigsten Gütekriterien untersucht und unter Standardbedingungen durchführbar und normiert sein. “*
> [Lienert & Raatz, 1998, Seite 14]

Eine Standardisierung kann dieser Definition zufolge nur erreicht werden, wenn der Test bezüglich der wissenschaftlichen Kriterien erstellt wurde, normiert ist und in seinen definierten Bedingungen durchgeführt werden kann.

Eine Normierung der Testergebnisse kann erreicht werden, wenn es ein Bezugssystem gibt, auf welches die erzielten Resultate abgebildet werden. Somit ist es möglich unterschiedliche Testungen, die auf das gleiche Bezugssystem abbilden, untereinander zu vergleichen. Als Beispiel kann der Intelligenztest genannt werden, dieser bildet seine Ergebnisse auf die Intelligenzquotientenskala ab, welche normaltransformiert mit einem Durchschnitt von 100 und einer Standardabweichung von 15 ist. [Lienert & Raatz, 1998]

Bei der Erstellung von Leistungstest ist zu beachten, dass die genannten Kriterien erfüllt werden müssen. Sonst ist eine Vergleichbarkeit verschiedenster Testungen nicht gewährleistbar. Bei den für den Lesetrainer erforderlichen Lesetests liegt die Schwierigkeit in der Normierung. Beim Erlernen des Lesens ist der Zeitpunkt der Testung ausschlaggebend für ein vergleichbares Resultat. Unterschiedliche Schulklassen lassen sich nur gegenüberstellen, wenn die Testung nach der gleichen Unterrichtsdauer durchgeführt wird. Aus diesem Grund gibt es für Lesetests Empfehlungen für den Durchführungszeitpunkt.

2.3 Tabletapplikationen um das Lesen zu lernen (iPad)

Der App Store von *Apple Inc.* bietet eine schier unendliche Auswahl an verschiedenen Applikationen (Apps). Dies macht es notwendig, das bestehende Angebot zu sichten und zu bewerten, ob eine verfügbare Applikation nicht die an den Lesetrainer gestellte Funktionalität bietet. Viele

dieser Programme haben sich der Bildung verschrieben und ermöglichen es, Grundkompetenzen oder auch erweiterte Themenbereiche zu erlernen. Unzählige dieser Programme befassen sich mit dem Thema Lesen, angefangen vom spielerischem Erlernen von Buchstaben bis hin zu einer Applikation „Schneller Lesen[8]“. Diese soll die Lesefähigkeiten von Erwachsenen steigern.

Eine Durchsicht des Angebotes hat ergeben, dass keine der bereits vorhandenen Applikationen die Anforderungen, die an den Lesetrainer gestellt werden, erfüllt. Die meisten dieser Programme beschränken sich auf vorgefertigte Inhalte und bedienen nicht die angestrebte Zielgruppe. Meist sind sie so ausgelegt, dass ein nicht schulpflichtiges Kind auf spielerische Art und Weise das Lesen oder Schreiben erlernt. Als Übung wird meist eine Wort-Bild Zuordnung angeboten. Eine solche Art des Lernens setzt einen gewissen Wortschatz voraus, der nicht von jedem Kind beherrscht werden kann. Als Ausweg dafür bietet z.B. die Applikation „Leseratte“[9] die Möglichkeit, selbst Wörter einzupflegen. Viele dieser Applikationen lassen sich individuell als Ergänzung zur Schule zuhause durch die Eltern einsetzen.

Die bereits erwähnte Applikation „Schneller Lesen“kommt dem Anforderungskatalog des Lesetrainers am nächsten. Sie implementiert viele verschiedene Leseaufgaben und eine statistische Auswertung der erzielten Leistungen. Die Applikation ist auf die Benutzung auf dem eigenen Gerät ausgelegt und ist somit nicht in Schulen einsetzbar, in denen die Schülerin bzw. der Schüler nicht immer das gleiche Tablet benutzt. Dieser Nachteil der Applikation schließt einen sinnvollen Einsatz im Klassenverband aus, da es keine Möglichkeit gibt, die Ergebnisse der Klasse zentral auszuwerten. Ein weiteres Ausschlusskriterium ist das Fehlen der zentralen Erstellung von Leseaufgaben.

[8] https://itunes.apple.com/at/app/schneller-lesen/id406388984?mt=8 - letzter Aufruf 25.1.2015

[9] https://itunes.apple.com/at/app/leseratte-kinder-lernen-lesen/id500950811?mt=8 - letzter Aufruf 25.1.2015

Die anderen getesteten Applikationen „Zebra-Schreibtabelle“[10], „Wörterfresser“[11] und „Conni Lesen“[12] erfüllen die Anforderungen an den Lesetrainer nicht. Somit ist ein Einsatz im Unterricht nicht möglich.

2.4 Lesetests

Im deutschsprachigen Raum gibt es viele standardisierte Verfahren, um die Lesefähigkeit von Schülerinnen und Schülern zu bestimmen. Bei der Suche nach den am besten geeigneten Verfahren für den Lesetrainer, wurden mehrere bereits bekannte Lesetests untersucht. Die Auswertung der verschiedenen Untersuchungsmethoden lässt Rückschlüsse über die Fertigkeiten zu, die für die Lesefähigkeit essentiell sind. Bei der Untersuchung wurde darauf geachtet, ob die Möglichkeit besteht, die Art der Testung in einem Programm umzusetzen. Ein weiteres Kriterium ist die Auswertung und ob diese automatisiert werden kann. Tests, die darauf basieren, dass eine Person die Aufgaben vorträgt oder die Schülerin oder der Schüler die Antwort selbst formulieren muss, sind nicht geeignet für die Umsetzung im Lesetrainer. Folgende Lesetests wurden untersucht:

2.4.1 Würzburger Leise Leseprobe Revision (WLLP-R) [Macari, 2014]

„Die Würzburger Leise Leseprobe ist ein Lesediganoseverfahren mit zeitlich geringem Aufwand, das das Lesetempo bzw. die Dekodiergeschwindigkeit von Kindern erfasst.“[Macari, 2014, Seite 24] Es wird rein die Lesegeschwindigkeit und die Dekodierfähigkeit ermittelt, andere Faktoren werden nicht erfasst. Diese wird durch Wort-Bildzuordnungen festgestellt. Hierzu werden zu einem Wort vier Bilder angezeigt. Das dem Wort entsprechende Bild muss zugeordnet werden. In den Schulstufen 1 - 3 sind 140 solcher Wort-Bild Paare zuordenbar, in der 4.

[10] https://itunes.apple.com/de/app/lesen-und-schreiben-lernen/id751540884?mt=8 - letzter Aufruf 3.2.2015
[11] https://itunes.apple.com/at/app/worterfresser-lesen-und-rechtschreiben/id804613013?mt=8 - letzter Aufruf 3.2.2015
[12] https://itunes.apple.com/at/app/conni-lesen/id524317031?mt=8 - letzter Aufruf 3.2.2015

Schulstufe sind es maximal 180 Paare. Die Probandeninnen und Probanden haben insgesamt 5 Minuten Zeit für die Absolvierung des Tests und müssen so viele Wörter wie möglich zuordnen. Das ermittelte Ergebnis der einzelnen Probanden kann aufgrund der mitgelieferten Normwerte eingestuft werden.[13]

Dieser Test ist theoretisch für den Lesetrainer geeignet, da jede Probandin und jeder Proband für sich arbeitet und keine Person die Aufgaben vortragen muss. Jedoch wurden der Test und die Vergleichsnormen so definiert, dass für jede Schulstufe (außer der vierten in der die maximale Anzahl von Wort-Bild Paaren erhöht ist) der selbe Test durchgeführt wird. Diese einseitige Verwendung ist nicht zielführend für einen erweiterbaren universellen Lesetrainer.

2.4.2 Salzburger Lese-Screening (SLS) [Mayringer & Wimmer, 2002]

Das Salzburger Lesescreening ist ein Gruppentest, dessen Grundgedanke es ist, die basale Lesefähigkeit in einer natürlichen Lesesituation zu erfassen. Hierzu wird die Lesegeschwindigkeit und indirekt die Lesegenauigkeit gemessen. *„Das Screening soll die Unterschiede der basalen Lesefertigkeit durch eine natürliche Leseanforderung, das heißt durch die inhaltliche Beurteilung von Sätzen, erfassen“* [Mayringer & Wimmer, 2002, Seite 5]. Dazu lesen die Schülerinnen und Schüler leise kurze, inhaltlich sehr einfache Sätze durch. Jeder dieser Sätze wird entweder als „Wahr“ oder „Falsch“ gekennzeichnet. Für die Bewertung des Screenings wird die Anzahl der in drei Minuten richtig beantworteten Sätze herangezogen, diese stellen nur die Rohdaten dar. Damit nun diese Werte verglichen werden können, wird der Lesequotient (LQ) verwendet. *„Dieser drückt aus, wie weit die bei einem Kind gemessene Lesefertigkeit vom Durchschnitt der Normierungsstichprobe abweicht.“* [Mayringer & Wimmer, 2002, Seite 8] Um die Vergleichbarkeit mit dem Intelligenzquotienten zu gewährleisten, wurde die selbe Skalierung für den LQ angewendet. Dies bedeutet, dass der LQ auf einer Normalverteilung mit einer Standardabweichung beruht. Somit steht ein LQ von 100 Punkten, für eine genau in der Norm liegende Lesefähigkeit.

[13] http://www.testzentrale.de/programm/wuerzburger-leise-leseprobe-revision.html - letzter Aufruf 25.1.2015

Beispiel für eine Aufgabenstellung:

„Bananen sind blau “ [Mayringer & Wimmer, 2002, Seite 5]

Das Salzburger Lesescreening eignet sich sehr gut, um es in den Lesetrainer zu integrieren. Die Durchführung wird selbständig von den Schülerinnen und Schülern durchgeführt, es bedarf nur einer kurzen Einführung in die Testungsmethode durch die Lehrerin bzw. den Lehrer. Die Auswertung ist durch reine Feststellung der richtig beantworteten Teilaufgaben gegeben. Weiters ist es sehr einfach möglich, selbst Tests in dieser Art zu erstellen. Es ist darauf zu achten, dass der Lesequotient auf selbst erstellte Screenings nicht anwendbar ist, da die Referenzauswertung nicht vorhanden ist.

2.4.3 Hamburger Lesetest für 3. und 4. Klassen (HAMLET3-4) [Macari, 2014]

Hier handelt es sich um einen Lesetest, der aus 2 Teilen besteht, dem „Worterkennungs-Test“ und dem „Leseverständnis-Test“. *„Der ‚Worterkennungs-Test‘ testet nicht nur die Lesefertigkeit und -genauigkeit, sondern bietet auch wesentliche Informationen zur Lesegeschwindigkeit. Es wird ermittelt, wie viele Wörter in einer vorgegebenen Zeitspanne richtig erlesen werden können.“*[Macari, 2014, Seite 26f] Um diese Fähigkeiten festzustellen, wird wie in der Würzburger Leise Leseprobe eine, Wort-Bildzuordnung durchgeführt. Der Unterschied besteht darin, dass es sich nur um 40 Wort-Bild Paare handelt und der Test nach maximal 90 Sekunden beendet wird.

Der Leseverständnis-Test überprüft, ob es der Schülerin bzw. dem Schüler möglich ist, den Inhalt eines selbst erlesenen Textes zu verstehen. Die Schülerin bzw. der Schüler muss mehrere Texte in verschiedenen Längen lesen und im Anschluss dazu Multiple-Choice-Fragen beantworten, deren Schwierigkeit unterschiedlich gestaltet sein kann. Dieser Teil des Testes ist nach maximal 64 Minuten zu Ende. Die Anzahl der bewältigten Aufgaben lassen Rückschlüsse auf die Lesefähigkeit der einzelnen Schülerinnen und Schüler ziehen.

Die Dauer des Tests schließt eine Umsetzung im Lesetrainer aus, weiters ist auch die Zusammenstellung des Tests komplex und würde eine Individualisierung durch die lehrende Person der einzelnen Klassen nur erschweren und in keinem Verhältnis zum erzielten Nutzen stehen.

2.4.4 Frankfurter Leseverständnistest für 5. und 6. Klassen (FLVT 5-6) [Lenhard & Schneider, 2009]

Der Frankfurter Leseverständnistest ist ein einfacher Test, er besteht aus zwei Texten, beide ca. 570 Wörter lang, einem Fiktivtext, der von einer freundschaftlichen Begegnung handelt, und einem Sachtext, der ein Naturereignis als Thema hat. Die Schülerin bzw. der Schüler liest still den Text durch und muss danach 18 Multiple-Choice-Fragen beantworten. Pro Text stehen der Schülerin bzw. dem Schüler 20 Minuten zu Verfügung.[14] Wiederum können anhand der richtig beantworteten Fragen, Schlüsse über die Lesefertigkeit der Schülerin bzw. des Schülers gezogen werden.

Der Ablauf und die Gestaltung des FLVT 5-6 ist sehr gut geeignet, um eine Umsetzung im Lesetrainer in Betracht zu ziehen.

2.4.5 Lesegeschwindigkeits- und Verständnistest (LGVT) [Lenhard & Schneider, 2009]

Hierbei handelt es sich, wie der Name bereits verrät, um ein Testverfahren, das die basale Lesekompetenz und die Fähigkeit das Gelesene zu verstehen, ermittelt. Der Unterschied zum Salzburger Lesescreening besteht darin, dass es sich hier nicht um einzelne, für sich stehende Sätze handelt, sondern um einen kontinuierlichen Text. Die Schülerinnen und Schüler müssen einen Text mit 1727 Wörtern durchlesen.[15] An 23 verschiedenen Stellen

[14] http://www.testzentrale.de/programm/frankfurter-leseverstandnistest-fur-5-und-6-klassen.html - letzter Aufruf 25.1.2015

[15] http://www.testzentrale.de/programm/lesegeschwindigkeits-und-verstandnistest-fur-die-klassen-6-12.html - letzter Aufruf 25.1.2015

im Text muss die Probandin oder der Proband aus drei Möglichkeiten das inhaltlich richtige Wort auswählen und den Satz vervollständigen.[15] Um die Lesegeschwindigkeit zu erfassen, wird der Test nach genau vier Minuten gestoppt. Die Probandin oder der Proband muss das zuletzt gelesene Wort markieren. Die Testbögen haben neben jeder Zeile die Anzahl der bereits gelesenen Wörter angegeben und über diese kann die Lesegeschwindigkeit abgeleitet werden. Die Leseverständniskompetenz wird über die Anzahl der korrekt ausgewählten Wörter erfasst und lässt eine Einschätzung der Fähigkeit des sinnerfassenden Lesens zu.

Das Grundprinzip des Lesegeschwindigkeits- und Verständnistests lässt eine Implementierung im Lesetrainer zu, jedoch kristallisiert sich bei genauer Betrachtung eine Schwierigkeit heraus. Die Erfassung der Anzahl der gelesenen Wörter kann automatisch nur durch ein Mitverfolgung der Augenbewegung, dem sogenannten Eye-Tracking, ermöglicht werden. Dafür ist spezielles Equipment notwendig und es ist deswegen nicht realisierbar und somit nicht im Lesetrainer integrierbar.

2.4.6 Knuspels Leseaufgaben [Adolph, 2009]

„Im Gegensatz zu vielen anderen Lesetest bekommt das Kind bei Knuspel-L keinen Text vorgelegt, zu welchem Fragen gestellt werden, sondern ihm werden Instruktionen gegeben“ [Adolph, 2009, Seite 31] Der Test besteht aus vier Teilaufgaben, welche die verschiedenen Lesefähigkeiten der Schülerinnen und Schüler ermitteln. Die Leseaufgabe ist so konzipiert worden, dass jede Schulstufe, in der sie eingesetzt wird, die gleichen Aufgaben verwendet und sich lediglich die Bearbeitungszeit verkürzt. Je höher die Schulstufe, desto kürzer ist die Zeit. Es wird empfohlen, in der 1. Schulstufe den 4. Teiltest nicht durchzuführen. Die Teilaufgaben setzten sich wie folgt zusammen:

- Hörverstehen: Hierbei werden den Schülerinnen und Schülern Fragen gestellt, die zum Beispiel wie folgt lauten: *„Wie viele verschiedene Buchstaben hat das Wort ‚knuspeln‘“* [Adolph, 2009, Seite 32]

- Der zweite Teil der Leseaufgabe beurteilt die Rekodierfertigkeit der Schülerinnen und Schüler. Es sind 40 Wortpaare (*Wahl - Wal*), die auf Homophonie zu testen sind, angegeben. Paare, die lautgleich sind, werden dementsprechend markiert.

- Der dritte Teil befasst sich mit dem Erkennen von Bedeutungen. Damit dies festgestellt werden kann, muss die Schülerin bzw. der Schüler 40 Pseudoworte auf ihren semantischen Inhalt überprüfen. Sofern ein solches Wort ein lautgleiches Pendant besitzt, muss dieses gekennzeichnet werden. Ein Beispiel für genannte Pseudowörter:

  ROGG[16] - ROCK
  FEDDER[16] - FEDER
  DOOSE[16] - DOSE
  KNAPE[16] - KNABE

- Das letzte Element des Tests untersucht das Verstehen von schriftlich gestellten Fragen. Schülerinnen und Schüler bekommen Aufgaben in schriftlicher Form gestellt. Diese sind inhaltlich ähnlich dem 1. Teil der Leseaufgabe. Beispiel:

  „Mit welchen beiden Buchstaben hört das Wort ‚Knuspel' auf? Schreibe die beiden Buchstaben in den 1. Kreis."[16]

Die Auswertung des Tests kann rein quantitativ erfolgen, bei der Entwicklung des Tests wurden Normwerte für die unterschiedlichen Schulstufen erstellt und diese können als Vergleichswerte herangezogen werden.

Knuspels Leseaufgabe untersucht im Vergleich zu den bereits behandelten Testverfahren die einzelnen Teilbereiche sehr genau, leider ist eine Umsetzung mit Hilfe des Lesetrainers nicht möglich, da der Test darauf beruht, dass die durchführende Person die einzelnen Aufgaben des ersten Teiltests laut vorliest, weiters entspricht die Konzipierung, in allen Schulstufen die gleichen Angabe zu verwenden, nicht den Anforderungen an das System.

[16] http://www.hogrefe.de/uploads/media/Vortrag_marx.pdf - letzter Aufruf 25.1.2015

2.4.7 Ein Leseverständnistest für Erst- bis Sechstklässler (ELFE 1-6) [Lenhard & Schneider, 2009]

Dieser Lesetest geht von der Annahme aus, dass Leseverständnis auf drei Ebenen aufgegliedert werden muss, die Wort-, Satz- und Textebene. Aus dieser Annahme heraus liegt es nahe, die Testung in drei Teilaufgaben aufzutrennen, den Wort-, Satz- und Textverständnistest. Von den untersuchten Lesetests ist dies der einzige, der bereits als Computerversion vorliegt. Diese beinhaltet eine weitere 4. Teilaufgabe, die sich die Vorteile des Computereinsatzes zunutze macht und die Lesegeschwindigkeit testet. Die verschiedenen Teilbereiche sind wie folgt aufgebaut:

- **Wortverständnis**: Die Aufgabe ist es, ein Bild mit dem dafür richtigen Wort zu verknüpfen. Der Schülerin bzw. dem Schüler stehen hierzu vier Wörter zur Auswahl. In der computerbasierten Version dieser Aufgabe ist es möglich, die vier Auswahlmöglichkeiten an randomisierten Positionen anzuzeigen.
- **Satzverständnis:** Der Schülerin bzw. dem Schüler werden nacheinander einzelne Sätze angezeigt, in denen ein Wort fehlt. Es gilt das richtige Wort aus fünf auszuwählen, um den Satz sinnvoll zu vervollständigen.
- **Textverständnis:** Der Schülerin bzw. dem Schüler wird ein kurzer Text und eine Frage mit vier möglichen Antworten angezeigt. Es muss die richtige Antwort gefunden werden. Durch Anpassen der Fragen ist es möglich das Textverständnis genau zu ermitteln. Als Beispiel kann nach einzelnen Informationen oder auch nach Schlussfolgerungen gefragt werden.
- **Lesegeschwindigkeit:** Dieser Test macht sich die Möglichkeit der zeitlich begrenzten Darstellung von Informationen zunutze. Der Schülerin bzw. dem Schüler werden Vornamen kurze Zeit angezeigt Die Aufgabe besteht darin, das Geschlecht des angezeigten Namens festzustellen und zuzuordnen. In den ersten Durchgängen werden die Namen 900ms lang angezeigt, in späteren Durchläufen wird die Zeit, in der die Namen sichtbar sind, den erzielten Ergebnissen angepasst. Bei schlechter Erkennungsrate wird die Anzeigezeit erhöht und im Umkehrschluss verkürzt, wenn gute Ergebnisse erzielt werden.

Die Auswertung dieses Lesetests basiert auf der Anzahl der richtig beantworteten Aufgaben. Dem Test liegen Normtabellen bei, die dabei helfen, die erzielten Leistung der einzelnen Schülerinnen und Schüler zu bewerten. Die manuelle Auswertung ist aufwändiger als bei anderen Testverfahren, weswegen die computerbasierte Version des Tests vorzuziehen ist.

Die angeführten Teilaufgaben eignen sich, um eine Umsetzung im Lesetrainer in Betracht zu ziehen. Neben der Tatsache, dass es bereits eine Implementierung dieses Tests gibt, ist die Individualisierung bei dieser Art der Aufgaben gegeben. Lehrerinnen und Lehrer können die Tests an ihren Unterricht und den bereits erlernten Unterrichtsstoff anpassen.

2.4.8 Auswahl der Lesetests des Lesetrainers

Die genauere Betrachtung der verschiedenen Lesetests zeigt, dass alle bis auf „Knuspels Leseaufgabe“ in der Art der Testung sehr ähnlich sind. Die Lesegeschwindigkeit und Dekodierfähigkeit wird über eine Zuordnung eines Bildes zu einem aus mehreren möglichen Wörtern gemessen. Leseverständnis wird entweder durch die Vervollständigung von einzelnen Sätzen, oder durch Beantwortung von inhaltlichen Fragen zu einem gelesenen Text erfasst. Knuspels Leseaufgabe testet andere Aspekte der Lesefähigkeit, wie das Hörverstehen und die Fähigkeit gleichlautende Wörter zu erkennen. Die Auswertung der gemessenen Grundfertigkeiten lässt den Schluss zu, dass folgende Aspekte unablässig für die Lesekompetenz sind:

- **Dekodierfähigkeit**: Die schnelle Erfassung der Bedeutung eines Wortes.
- **Leseverständnis auf Satzebene**: Verstehen der Semantik eines Satzes.
- **Textverständnis**: Erkennen der inhaltlichen Aussage eines Textes.

Eine Beurteilung der verschiedenen Tests, bezüglich ihrer Implementierbarkeit und der Erfüllung der geforderten Vorgaben ergab, dass eine Anlehnung an die Teilaufgaben des „Leseverständnistest für Erst- bis Sechstklässler“ sinnvoll erscheint.

In einer Diskussion mit Lehrerinnen und Lehrern über das geplante System, wurden die Vorschläge aus der Sicht der lehrenden Person berücksichtigt und eine Einigung über die Lesetests wurde erzielt. Der Lesetrainer implementiert drei verschiedene Testarten, diese werden durch eine zentrale Webapplikation verwaltet und die Leseaufgaben werden durch drei verschiedene Tabletapplikationen realisiert. Folgende Aufgaben werden für den Lesetrainer festgelegt:

Lesegeschwindigkeitstest

Der Lesegeschwindigkeitstest ist ein Kombinationstest, er ermittelt die Dekodierfähigkeit und indirekt die Lesegenauigkeit. Der Test ist dem des Salzburger Lese-Screenings nachempfunden. Jedoch ist die Auswertung nicht an den Lesequotienten des Salzburger Lese-Screenings gebunden. Die Schülerinnen und Schüler müssen in einer beschränkten Zeit möglichst viele Sätze auf ihren Wahrheitsgehalt prüfen. Beispiel:

Ein Gerät, mit dem man Essen warm macht, heißt Kühlschrank.

Leseverständnistest

Hierbei handelt es sich um einen Text, der das Leseverständnis auf Satzebene ermittelt. Er ist an den Satzverständnistest des „Leseverständnistest für Erst- bis Sechstklässler“ angelehnt. Ein kurzer Satz enthält eine Lücke und die Schülerin bzw. der Schüler muss aus vier Antwortmöglichkeiten das inhaltlich korrekte Wort wählen. Wie der Lesegeschwindigkeitstest ist auch diese Aufgabe zeitlich beschränkt. Beispiel:

Die Kinder im Garten ***spielen, schreiben, essen, liegen*** *mit einem Ball.*

Textverständnistest

Dieser Test stellt fest wie es um das Textverständnis der Schülerin bzw. des Schülers beschaffen ist. Auch dieser ist einem Subtest des „Leseverständnistest für Erst- bist Sechstklässler“nachempfunden. Die Schülerinnen und Schüler müssen einen kurzen Text lesen. Als Auswahlmöglichkeit werden ihnen vier verschiedene Sätze angeboten, wobei nur einer inhaltlich zu dem zuvor gelesenen Text passt. Dieser Test verzichtet auf eine zeitliche Beschränkung, Schülerinnen und Schüler haben unbeschränkt Zeit um die Aufgaben zu lösen. Beispiel:

Benjamin spielt mit einem schönen neuen Ball im Garten. Er wirft ihn in die Luft und fängt ihn wieder. Maximilian, David und Emma wollen mitspielen. Doch Benjamin will den Ball für sich alleine haben und lässt die 3 nicht mitspielen.

Emma spielt mit einem neuen Ball alleine in ihrem Zimmer.
Benjamin spielt mit David, Emma und Maximilian.
David, Emma und Maximilian dürfen nicht mit Benjamin spielen.
Die 4 Kinder spielen im Garten fangen.

Auswertungsgrößen des Lesetrainers

Die Auswertung der drei implementierten Leseaufgaben erfolgt bei allen Tests über die Anzahl der richtig beantworteten Teilaufgaben und wird prozentual und explizit ausgegeben. Als weitere Auswertungsgröße wird die Antwortgeschwindigkeit gemessen, diese gibt an, wie viel Zeit zwischen Anzeigen der Fragestellung und der Auswahl der Antwort vergangen ist. Mit diesen Messwerten kann festgestellt werden, ob eine Schülerin bzw. ein Schüler Probleme hatte, eine Frage zu verstehen.

Bis auf den Textverständnistest besteht bei den restlichen Tests ein zeitlicher Druck, es gibt eine maximale Bearbeitungszeit, die eine Schülerin bzw. ein Schüler nicht überschreiten darf. Sobald diese Zeit erreicht ist, wird der Test abgebrochen und die Ergebnisse gespeichert.

Die erfassten Auswertungsgrößen lassen nicht nur eine Bewertung der einzelnen Schülerinnen und Schüler zu, sondern auch ein gemitteltes Ergebnis über die Klasse ist möglich. Dieser Mittelwert wird über zwei Berechnungsarten ermittelt. Einerseits über das arithmetische Mittel und auf der anderen Seite über den Median mit der dazugehörigen Streuung. Der Median bietet den Vorteil, dass statistische Ausreißer das Ergebnis nicht verfälschen.

Durch eine Gegenüberstellung der Anzahl der richtig bzw. falsch gegebenen Antworten pro Frage eines Lesetests, lassen sich Rückschlüsse über die Schwierigkeit der Frage ziehen. Zusammen mit einer Auswertung der Antwortzeiten lassen sich genauere Aussagen über die einzelnen Teilaufgaben treffen.

3 Technologien für die Umsetzung

Die Vorgaben für das System waren es, Lesetests zu erstellen, diese von den Schülerinnen und Schülern durchführen zu lassen und die Ergebnisse auszuwerten.

Es stellte sich die Frage, welche Plattformen sich am besten zur Realisierung dieser Aufgaben eignen. Nach Abwägung vieler verschiedener Ansätze, fiel die Wahl grundsätzlich auf drei Technologien. Die Erstellung, Auswertung und Aufbereitung wird durch eine Webapplikation realisiert. Diese erlaubt es den Lehrerinnen und Lehrern, die administrativen Aufgaben durchzuführen.Bei der Suche nach einer geeigneten Plattform zur Durchführung der Lesetests, stellte sich schnell heraus, dass die beste Wahl eine tabletbasierte Applikation ist. Eine einfache und intuitive Bedienung ist Voraussetzung für ein solches System, dies macht ein Tablet zur perfekten Plattform. Durch die Wahl eines Tablets, verkleinert sich die Anzahl der möglichen Anwenderinnen und Anwender. Nicht alle Bildungseinrichtungen verfügen über Tabletklassen oder Tablets in Klassenstärke. Um eine größere Verbreitung zu erlangen, wurde eine webbasierte Version der Tabletapplikationen als Alternative realisiert. Nach der grundsätzlichen Entscheidung der Plattformen, folgte die Auswahl der speziellen Technologien. Bei der Wahl der Tablets viel die Entscheidung auf das von *Apple Inc.* entwickelte Tablet *iPad* und das mobile Betriebssystem *iOS*. Die Tatsache, dass es bereits Schulen gibt, die Versuchsweise *iPads* im Unterricht einsetzen, bestärkte die getroffene Wahl.

Für die Webapplikation zur Administration des Lesetrainers fiel die Wahl auf das *Open-Source-Applikations-Framework Yii*. Die webbasierte Version der Tabletapplikation wurde durch ein weiters Open-Source-Framework mit dem Namen *Ember.js* realisiert. Die Entscheidung für ein weiteres Framework ist

darauf zurückzuführen, dass *Yii* als ein PHP Framework serverseitig ausgeführt wird. Die dadurch entstandenen Schwierigkeiten wurden durch den Einsatz des clientseitigen Javascript Frameworks *Ember.js* gelöst.

3.1 iOS [Apple Inc., 2014]

iOS ist ein, von *Apple Inc.* entwickeltes Betriebssystem, das für den Einsatz in mobilen Geräten entworfen wurde. Im Moment wird es in vier Produktfamilien eingesetzt: *iPhone, iPod Touch, iPad* und *Apple TV*. iOS ist ein proprietäres Betriebssystem und läuft nur auf den Geräten von Apple, die über spezielle Hardwarespezifikationen verfügen.
Für die Entwicklung einer Applikation für iOS ist es entscheidend, dass die Entwicklerin oder der Entwickler versteht, wie das Betriebssystem aufgebaut ist. iOS ist in Schichten (Abbildung 3.1) unterteilt. Darunter ist zu verstehen, dass die Abstraktion der eigentlichen Hardware von Schicht zu Schicht immer höher wird. Durch das nicht direkte Ansprechen der Hardware hat die Entwicklerin oder der Entwickler den Vorteil, dass die Applikation unabhängig von der Hardware erstellt werden kann. Die einzelnen Schichten bieten unterschiedlich tiefen Zugriff auf die Funktionen der Hardware. Diese Funktionen sind wiederum in Pakete, sogenannte Frameworks, zusammengefasst. Für die Appentwicklung bedeutet das, am besten auf der höchst möglichen Schicht, also auf der mit dem höchsten Abstraktionslevel, zu arbeiten und die App dort zu entwickeln, um Hardwareunabhängigkeit zu garantieren. Das mobile Betriebssystem von Apple besteht aus folgenden Schichten:

3.1.1 Betriebssystem [Apple Inc., 2014]

- **Cocoa Touch:** Diese Ebene bietet die höchste Abstraktion der Hardware. Hier sind die grundlegenden Funktionen und Technologien für die Apps definiert. Die wichtigsten Funktionen, wie Multitasking, Touchscreenbedienung, User-Interface-Gestaltung und viele weitere sind hier definiert. Eine der wichtigsten Technologien in dieser Schicht ist das sogenannte Storyboard (Abbildung 3.2). Mit dieser Funktion wird das User Interface erstellt. Die einzelnen Views werden angezeigt und man gibt an, welche Schaltflächen, Eingabefelder und Anzeigen es geben

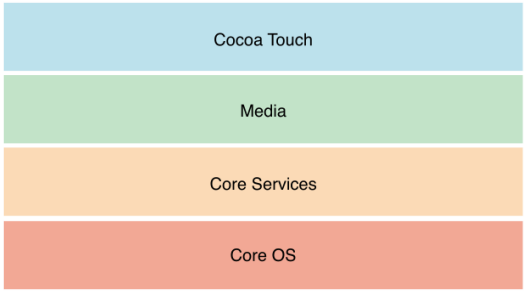

Abbildung 3.1: Schichten von iOS [Apple Inc., 2014]

wird. Weiters können auch animierte Übergänge der Views definiert werden. Der Vorteil ist, dass die Entwicklerin oder der Entwickler das komplette User Interface, ohne eine Zeile Programmcode zu schreiben, designen kann. Dadurch ist es möglich, zuerst das User Interface zu gestalten und erst danach mit der eigentlichen Programmierarbeit zu beginnen. Weitere wichtige High-Level-Funktionalitäten die in der Cocoa Touch Schicht implementiert sind:

- **Multitasking**: Unter Multitasking versteht man die Möglichkeit eines Betriebssystems mehrere Aufgaben parallel auszuführen. Dabei handelt es sich nur selten um echte parallele Ausführung mehrerer Programme, sondern um eine Pseudoparallelität. Dies bedeutet, dass Programme hintereinander ausgeführt werden. Sie werden immer nur für eine bestimme Zeit in den sogenannten Runningstate gesetzt, in dem das Programm Instruktionen ausführen kann. Durch die schnelle Abfolge von verschiedenen Programmen hintereinander scheint es, als würden die Programme parallel ausgeführt werden. Während bei regulären Computern Multitasking keinen Einschränkungen unterliegt und Programme immer parallel ausgeführt werden, ist die Umsetzung von Multitasking in Betriebssystemen für mobile Geräte, wie Smartphones oder Tablets mit Einschränkungen belegt. Das Problem liegt darin, dass parallel ausgeführte Programme die Akkulaufzeit des mobilen Gerätes beeinflussen. Je mehr Programme ausgeführt werden, desto mehr Rechenleistung und Energie wird benötigt und die Akkulaufzeit

sinkt. Aus diesen Gründen wurde Multitasking in iOS auf folgende Weise implementiert: Programme, die gerade nicht aktiv laufen, werden in den Hintergrund gelegt und jede Ausführung des Programms wird gestoppt. Der Status in dem es sich befindet wird gespeichert und sofort wiederhergestellt, sobald das Programm in den Vordergrund gebracht wird. Somit ist eine Ausführung an der Stelle, an der das Programm in den Hintergrund gelegt wurde, gewährleistet. Damit nun Multitasking möglich wird, kann jedes Programm Rechenzeit anfordern, um wichtige Aufgaben auch im Hintergrund auszuführen. Die Idee hinter all diesen Maßnahmen ist es, dass sich eine Applikation explizit für Hintergrundrechenzeit beim System anmelden muss. Dadurch nimmt das Betriebssystem an, dass jedes Programm nicht im Hintergrund laufen muss. Dies verringert den Energiebedarf und verlängert die Akkulaufzeit.

- **UI State Preservation**: State Preservation ist ein System, das bei Multitasking zum Einsatz kommt. Wie bereits beschrieben, muss der Zustand, der sogenannte State der Applikation, gespeichert werden, wenn diese in den Hintergrund gebracht wird. Dieses System sorgt dafür, dass die Applikation, sobald sie in den Vordergrund gebracht wird, um sie zu verwenden, die Ausführung genau an dem Punkt aufnimmt, an dem sie in den Ruhezustand versetzt wurde.

- **Standard System View Controllers**: Für die Entwicklung von Apps stellt das Betriebssystem vorgefertigte ViewController zur Verfügung, damit Funktionen, die oft gebraucht werden, nicht jedes Mal von den Entwicklern neu geschrieben werden müssen. Dazu zählen Funktionen wie Foto und Videoaufnahme, Kalenderfunktionen, Email oder SMS-Funktionalität sowie Kalender und Dateifunktionen.

- **Gesture Recognizers**: Hierbei handelt es sich um die Erkennung der Gesten, die bei Smartphones und Tablets oft zum Einsatz kommen. Dieses System kann diese Gesten erkennen und leitet dies an die zuständigen Controller weiter, die für die Ausführung der gewünschten Funktion verantwortlich sind.

  - **AutoLayout**: Autolayout bietet eine einfache Methode, Elemente der Benutzeroberfläche zu positionieren. Die Position wird nicht in absoluten Werten angegeben, sondern relativ zu benachbarten Objekten oder als Abstand zu den Rändern des Bildschirms. Dies bietet eine sehr flexible Möglichkeit der Anpassung von Applikationen an unterschiedliche Gegebenheiten. Das System des Autolayouts wurde notwendig, da sich die Hardware der von *Apple Inc.* entworfenen Geräte ändert. Durch diese Änderung treten nun verschiedene Displaygrößen und Auflösungen auf, dadurch wird eine statische, auf die tatsächliche Auflösung bezogene Positionierung schwer. Durch die relative Positionierung wird eine korrekte Anzeige auf allen Geräten und Auflösungen gewährleistet.

  Die beschriebenen Funktionen werden in sogenannte Frameworks zusammengefasst. Frameworks sind Gruppen von Funktionen, die nicht bei jeder Applikation zum Einsatz kommen müssen. Damit die Funktionen der optionalen Frameworks genutzt werden können, müssen die entsprechenden Frameworks explizit geladen werden. Diese Maßnahme bewirkt, dass unnötige Bibliotheken nicht eingebunden werden, die nur den Speicherverbrauch der Applikation steigern würden. Das Framework **UIKit Framework** stellt eine Ausnahme dar, da es sich hier um die Sammlung der Standardfunktionalitäten handelt und jede Applikation diese benutzt.

- **Media:** Der Media Layer ist für das Multimediaerlebnis der Benutzerin oder des Benutzers verantwortlich. Diese Schicht wurde entwickelt, um der Programmiererin oder dem Programmierer den Zugang zu den Grafik, Audio, Video und Airplay[1] Technologien zu erleichtern. Bei Grafiktechnologien handelt es sich um Animationsfunktionen und 3D Frameworks. Der Audioteil der Schicht stellt Funktionen für verschiedene Audioformate zur Wiedergabe dieser bereit. Zugriff auf das Mikrofon und den Lautsprecher des Gerätes wird auch durch diesen gewährleistet.

[1] http://www.apple.com/airplay/ - letzter Aufruf 25.1.2015

Abbildung 3.2: Storyboard mit Übergängen

Im Videobereich der Abstraktionsebene werden, wie beim Audioteil, Funktionen zur Wiedergabe von Videos bereitgestellt und der Zugriff auf die Kamera ermöglicht.
Die Airplayfunktionen sind ein Framework, um drahtlos externe Bildschirme oder Lautsprecher anzusprechen. Es lässt sich sehr gut erkennen, dass in dieser Schicht mehr Zugriff auf die Hardware möglich ist, als in der CocoaTouch-Schicht, was dazu führt, dass die Programmiererin oder der Programmierer darauf achten muss, auf welcher Hardware die App gerade ausgeführt wird, da auf den verschiedenen Geräten nicht alle Hardwarefunktionen zur Verfügung stehen.

- **Core Services** Der Core-Service-Teil des Betriebssystems beinhaltet alle wichtigen Systemfunktionen, die jede App benutzt. In den meisten Fällen ist es gar nicht direkt ersichtlich, dass Funktionen aus dieser Schicht verwendet werden, da sie in den abstrakteren Schichten gekapselt werden, beispielsweise ist die Telefonfunktion in dieser Schicht implementiert. Ein paar der wichtigsten Funktionen und Frameworks dieser Schicht:

- **Peer-to-Peer Services**: Diese Funktionen implementieren die Möglichkeit des Betriebssystems, sich mit anderen iOS-Geräten über Bluetooth zu verbinden und Daten auszutauschen. Hauptsächlich werden diese Funktionen in Spielen verwendet, um ein gemeinsames Spielerlebnis zu erschaffen, jedoch gibt es immer mehr Applikationen, die diese Art der Vernetzung nutzen.

- **iCloud Storage**: Die iCloud-Funktionen machen es möglich, Daten aus der App im zentralen Speicher des Nutzers auf den Apple Servern zu hinterlegen, damit sie auch geräteübergreifend genutzt werden können. Dies ermöglicht es zum Beispiel, dass die Benutzerin oder der Benutzer auf einem mobilen Gerät beginnt ein Dokument zu schreiben und auf einem anderen Gerät dieses finalisiert, ohne das Dokument manuell zu synchronisieren.

- **Block Objects**: Bei Block Objekten handelt es sich um eine Konstruktion basierend auf der Programmiersprache C, die in den Objective-C Code eingebunden werden kann. Bei diesem Konstrukt handelt es sich im Grunde um eine anonyme Funktion und deren Daten. In anderen großen Programmiersprachen wird ein ähnliches Konzept angeboten und dort als *Closure* oder *Lambda* bezeichnet. Solche Blöcke eignen sich sehr gut als Callback-Funktionen oder als eigenständige Handler-Funktionen, die nur einmal benötigt werden.

- **Automatic Refernce Counting ARC**: Diese mit der 5. Version des Betriebssystems hinzugefügte Funktion ist eine Erleichterung für Entwicklerinnen oder Entwickler. Der Compiler fügt beim Kompilieren des Programms die entsprechenden Programmzeilen ein, um die Objekte zur richtigen Zeit freizugeben. Dieses System ersetzt das *managed-memory-model* der früheren Versionen, bei dem Entwicklerinnen und Entwickler für die Freigabe der Objekte verantwortlich waren.

- **Data Protection**: Datensicherheit wird wichtiger und immer mehr Applikationen verwenden sensible Benutzerdaten. Damit die Sicherheit dieser gewährleistet ist, gibt es die Data-Protection-

Funktionen in iOS. Diese ermöglichen es, dass die Applikationen bestimmte Dateien als „protected“ deklarieren. Sobald eine Datei als zu schützend markiert ist, wird sie mit den im System vorhandenen Verschlüsselungsverfahren geschützt. Wenn das mobile Gerät nun durch eine Codesperre geschützt ist, kann auf die Datei nicht zugegriffen werden. Durch das Entsperren des Geräts wird ein Entschlüsselungskey für die Applikation erstellt. Mit Hilfe dieses Schlüssel ist ein Zugriff auf die geschützte Datei wieder möglich. Die Implementation von solchen Sicherheitsfeatures, stellt natürlich auch Anforderung an die Applikationsentwicklung. Potentielle Angriffe können sich gegen die Generierung der zu schützenden Daten richten. Sofern diese Erzeugung nicht in gesicherten Bereichen stattfindet, ist es möglich, die unverschlüsselten Daten auszulesen.

- **Grand Central Dispatch GCD**: Der Grand Central Dispatcher ermöglicht es, den Ablauf gewisser Funktionen der Applikation zu steuern. Der GDC bietet somit eine performante und optimierte Alternative zum traditionellen Threading.

- **SQLite**: Mit dieser Funktion kann eine lokale Datenbank verwendet werden. Es ist eine sehr leichtgewichtige Datenbank, um lokal am Gerät Daten zu speichern und diese dauerhaft im Speicher zu behalten.

- **XML Support**: iOS biete eine XML Engine an, die es ermöglicht, Daten aus einem XML Dokument herauszulesen und auch zu verändern. Es wird auch die Möglichkeit angeboten, XML-Datein in HTML Dateien umzuwandeln, um diese auszugeben. Bei XML handelt es sich um ein Dateiformat, in dem Daten zwischen Applikationen ausgetauscht werden können.

Frameworks des Core Service Layers:

- **CFNetwork Framework** ist eine Sammlung von Funktionen die es ermöglicht, Netzwerkverbindungen aufzubauen bzw. Netzwerkfunktionen zu verwenden. Dazu zählen DNS Abfragen, SSL-Verbindungen, das Arbeiten mit FTP, HTTP und HTTPS Servern sowie das Anbieten von Services über das Bonjour Netzwerkprotokoll.

- **Core Data Framework** stellt die Möglichkeit bereit, das Datenmodell des Model-View-Controller-Modells umzusetzen. Hierbei werden Funktionen angeboten, um das Datenmodell einfach und effizient zu erstellen und zu benutzen. Diese Funktionen verringern den Programmieraufwand erheblich und werden von Apple als die zu bevorzugenden Methoden empfohlen, um mit Datenmodellen zu arbeiten.

- **Core Location Framework** Dieses Framework stellt die Funktionen zur Ortsbestimmung zur Verfügung. Sehr viele Applikationen haben die Möglichkeit, ortsabhängige Informationen anzuzeigen oder sind auf die Ortsbestimmung angewiesen, um die korrekte Funktionalität zu gewährleisten. Das Framework stellt mehrere Arten der Lokalisierung bereit. Die genaueste Lokalisierung erfolgt über den GPS/GLONASS Empfänger des Geräts, sofern dieser vorhanden ist. Weiters ist es möglich, die Position über WLAN, Bluetooth-Beacons oder Mobilfunksendeanlagen festzustellen. Je nach Ausstattung des Geräts und Anforderungen an Genauigkeit und Energieverbrauch wählen Entwicklerinnen und Entwickler die Art der Positionsbestimmung.

  - **Core Motion Framework** Hierbei handelt es sich um die gesammelten Funktionen der Bewegungserkennung. Je nach Gerät stehen verschieden Sensoren zur Verfügung. Die Entwicklerinnen und Entwickler können wählen, ob sie die rohen Daten des Beschleunigungssensors oder des Gyroskops erhalten wollen, oder ob sie die vom Betriebssystem interpretierten Daten verarbeiten wollen. Viele Applikationen nutzen diese Daten, um ein erweitertes Userinterface zu ermöglichen bzw. auch um die Lage des Geräts im Raum zu erkennen und diese Information zu verarbeiten.

  - **JavaScript Core** ist eine Sammlung von Funktionen, die es ermöglichen, JavaScript Programmcode auszuführen. Weiters das Erstellen und Verarbeiten von JSON Objekten. JSON Objekte werden genutzt, um Daten auszutauschen, da es ein sehr einfaches Format ist und leicht zu evaluieren ist.

  - **System Configuration Framework** bietet die Möglichkeit festzustellen, über welche Internetverbindung das Gerät verfügt, ob es sich um eine Verbindung über WLAN oder über das Mobilfunknetzwerk handelt.

- **Core OS** Dieser Teil des Betriebssystems stellt den Kern dar. Der Kern kommuniziert direkt mit der Hardware und stellt so, wie alle anderen Schichten, Funktionen zur Kommunikation zur Verfügung. Auch wenn bei der Entwicklung einer Applikation nicht direkt auf die Funktionen dieser Schicht zugegriffen wird, die benutzten Frameworks tun es. Folgende Funktionen sind im Kern realisiert:

  - **Accelerate Framework**: Dieses Framework implementiert Funktionen für lineare Algebra, Bildverarbeitung oder digitales Aufbereiten von Audiodaten. Diese Funktionen sind speziell für die Hardware optimiert, um die bestmögliche Performance zu gewährleisten.

  - **Core Bluetooth Framework**: Diese Sammlung an Funktionen bietet Zugriff auf die Bluetooth Hardware und die damit verbundenen Möglichkeiten, mit externen Geräten zu kommunizieren und Daten auszutauschen.

- **Generic Security Service Framework**: Hier werden häufig verwendete Sicherheitsfunktionen in einem Paket zusammengefasst. Es beinhaltet die Standardimplementation der am häufigst verwendeten Sicherheitsstandards und auch die Möglichkeit Zugangsdaten zu verwalten.

- **Local Authentication Framework**: Es handelt sich um die Implementation einer lokalen Sicherheitspolitik. Manche Applikationen arbeiten mit sensiblen Daten. Um diese zu schützen, kann man dieses Framework nutzen, wobei sich die Benutzerin oder der Benutzer authentifizieren muss, um Zugriff auf bestimmte Daten zu erhalten. Ab iOS 7 ist es möglich, mit der entsprechender Hardware *(ab iPhone 5S)* den gespeicherten Fingerabdruck zu benutzen, um Zugriff zu gewähren.

- **Security Framework**: Zusätzlich zu den in anderen Frameworks enthaltenen Sicherheitsfunktionen wird ein explizites Sicherheitsframework angeboten, um die Sicherheit der Daten einer Applikation zu gewährleisten. Es wird die Funktionalität zur Verfügung gestellt, die es erlaubt Zertifikate, öffentliche und private Schlüssel zu verwalten, sowie die Vertrauenspolitik zu steuern. Weiters wird auch das Speichern von Zertifikaten und Schlüsseln in der sogenannten „Keychain“ möglich. Hierbei handelt es sich um einen sicheren Speicher für sensible Benutzerdaten.

- **System**: Hier ist der eigentliche Kern des Betriebssystems mit den Treibern für die Hardware und allen low-level UNIX Interfaces implementiert. Der Kernel ist für alle low-level-Funktionen des Systems verantwortlich und stellt somit die unterste Schicht des Betriebssystems dar.

- **64-Bit Support**: Ursprünglich basierte iOS auf einer 32-Bit Architektur. Doch mit der Weiterentwicklung der Prozessoren wurde es notwendig, 64-Bit Architekturen zu unterstützen. Ab der Version 7 von iOS ist dies nun der Fall und es ist möglich, eine Applikation für ein 64-Bit oder ein 32-Bit System zu kompilieren. Alle Frameworks des Betriebssystems sind zu beiden Architekturen kompatibel.

3.1.2 Entwicklungsumgebung

Applikationen für iOS werden mit der Entwicklungsumgebung Xcode von *Apple Inc.* erstellt. Diese beinhaltet alle Komponenten, die für die Entwicklung von Apps benötigt werden. Im Gegensatz zu anderen mobilen Betriebssystemen ist es bei Apple notwendig, einen Computer von Apple zu besitzen, um Applikationen für iOS zu entwickeln. Die erwähnte Entwicklungsumgebung Xcode ist nur unter Mac OS X (Betriebssystem für Applecomputer) lauffähig. Die aktuelle Version von iOS ist Version 8 und Apps werden mit Xcode 6 entwickelt[2]. Xcode ist nach einer kostenlosen Registrierung bei Apple als Entwickler frei erhältlich. Dieser kostenlose Entwickleraccount hat Einschränkungen. Es ist mit diesem Account nicht möglich eine Applikation in den App Store zu stellen, dafür muss ein kostenpflichtiger Account erworben werden.

3.1.3 Geräte

iOS wird auf folgenden Geräten verwendet:

- **iPhone**[3]: Das iPhone ist ein Mobiltelefon mit einem großen Touchscreen zur Bedienung des Geräts. Es verfügt über eine Kamera, GPS, GLONASS und einen Kompass zur Positionsbestimmung, einen 3-Achsen-Gyrosensor zur Lagebestimmung des Telefons, einen Beschleunigungssensor, sowie einen Annäherungs- und Umgebungslichtsensor und ein Barometer. In den neueren Versionen ist ein biometrischer Sensor zur Erfassung des Fingerabdrucks der Benutzerin bzw. des Benutzers integriert. Zur Kommunikation verfügt das iPhone über WLAN, Bluetooth, NFC und die Mobilfunktechnologien GSM, UMTS und LTE. Die aktuelle 8. Generation (*iPhone 6 und iPhone 6 Plus*) wurde am 19. September 2014 veröffentlicht.[4]

[2] Stand Jänner 2015
[3] http://www.apple.com/iphone/ - letzter Aufruf 25.1.2015
[4] http://www.apple.com/pr/library/2014/09/09Apple-Announces-iPhone-6-iPhone-6-Plus-The-Biggest-Advancements-in-iPhone-History.html - letzter Aufruf 25.1.2015

- **iPod Touch**[5]: Der iPod Touch ist ein Mediaplayer mit Touchscreenbedienung, der auf dem Design des iPhones beruht. Er verfügt über eine Kamera, WLAN und Bluetooth zur Kommunikation. Ein 3-Achsen-Gyrosskop und ein Beschleunigungssensor sind für die Lagebestimmung verantwortlich, weiters verfügt der iPod Touch über einen Umgebungslichtsensor. Die Mobilfunkstandards sowie die Positionsbestimmung werden vom iPod Touch nicht unterstützt. Die aktuelle Generation des iPod Touch wurde am 21. September 2012 veröffentlicht[6].

- **iPad/iPad mini**[7]: Das iPad ist ein Tablet Computer, der über einen großen Touchscreen zur Bedienung und über die gleiche Sensorik und Kommunikationsmöglichkeiten wie das iPhone verfügt. Die einzige Einschränkung gegenüber dem iPhone besteht darin, dass es nicht möglich ist, über die Mobilfunktechnologien zu telefonieren. Die aktuelle 6. Generation des iPads (*iPad Air*) und die 3. Generation der kleinere Variante (*iPad mini*) wurden am 16. Oktober 2014 der Öffentlichkeit präsentiert[8].

- **Apple TV**[9]: Apple TV ist eine auf iOS basierende Set-Top-Box. Diese ermöglicht es, Video und Audioinhalte im Internet zu kaufen oder zu entlehnen und diese auf einem TV Gerät anzuzeigen. Apple TV basiert zwar auf iOS, jedoch können auf dem Apple TV keine Apps installiert werden. iOS wird verwendet um die leistungsstarke und energiesparende Hardware des iPhones zu verwenden. Die aktuelle 3. Generation des Apple TV wurde am 7. März 2012 vorgestellt[10].

[5] http://www.apple.com/ipod-touch/ - letzter Aufruf 25.1.2015
[6] http://www.apple.com/pr/library/2012/09/12Apple-Introduces-New-iPod-touch-iPod-nano.html - letzter Aufruf 25.1.2015
[7] http://www.apple.com/ipad/ - lezter Aufruf 25.1.2015
[8] http://www.apple.com/pr/library/2014/10/16Apple-Introduces-iPad-Air-2-The-Thinnest-Most-Powerful-iPad-Ever.html - letzter Aufruf 25.1.2015
[9] https://www.apple.com/appletv/ - letzter Aufruf 25.1.2015
[10] http://www.apple.com/pr/library/2012/03/07Apple-Brings-1080p-High-Definition-to-New-Apple-TV.html - letzter Aufruf 25.1.2015

Objectiv-C

Objective-C ist eine der Programmiersprachen, die Apple für das Betriebssystem iOS und dessen Applikationen verwendet. Sie entstand in den 1980er Jahren aus dem Vorhaben, die prozedurale Sprache C um die objektorientierten Möglichkeiten der Sprache Smalltalk zu erweitern. Deswegen setzt sich die Syntax von Objective-C aus der von C und Smalltalk zusammen. In den 1990er Jahren war Apple auf der Suche nach einem neuen Betriebssystem als Ersatz für das alternde Mac OS. Darum erwarb Apple die Firma NeXT und damit deren Betriebssystem NeXTStep. NeXTStep wurde ursprünglich in Objective-C programmiert und aus diesem Grund wurde Objective-C die Programmiersprache für Mac OS und iOS.
[Smyth, 2012, Seite 15ff]
Da sich die Syntax aus C und Smalltalk zusammensetzt, gibt es Unterschiede zu den anderen populären Programmiersprachen. Ein Beispiel für solch einen Unterschied ist der Methodenaufruf eines Objektes (Listing 3.1). In *Java* oder *C++* wird eine Methode eines Objektes mit *Objektname.Methode* aufgerufen, unter Objective-C mit *[Objektname Methode]*.

```
1 @implementation
2
3 MyClass _class = [[MyClass alloc] init];
4 [_class Method];
```

Listing 3.1: Methoden Aufruf

Eine der wichtigsten Bereiche der Sprache Objective-C, mit der sich jede Entwicklerin und jeder Entwickler beschäftigen muss, ist das Speichermanagement. Objective-C verfügt zwar über einen Garbagecollector, der nicht mehr genutzte Objekte aus dem Speicher entfernt, jedoch ist wegen dem großen Implementierungsaufwands in iOS kein Garbagecollector vorhanden. Deswegen muss die Entwicklerin oder der Entwickler dem Freigeben von Objekten besondere Aufmerksamkeit schenken, damit keine nicht benutzten Objekte im Speicher zurückbleiben oder es zu Abstürze durch bereits entfernte Objekte kommen kann. Die ab iOS Version 5 verfügbare Funktion *Automatic Reference Counting* ist **kein** Garbagecollector, sondern nur eine Erleichterung für die Entwicklerin oder den Entwickler, da zur Compilezeit der Programmcode für die Freigabe der Objekte automatisch eingefügt wird.[Lee, 2012, Seite 573ff]

3.2 Yii-Framework [Winesett, 2010]

Das Yii-Framework ist ein Open-Source-Applikations Framework; Yii ist ein Akronym für „Yes It Is!“ und ist eine Anspielung auf Yii-Anfänger, die auf die Frage, ob etwas wirklich so einfach zu realisieren ist, die Antwort „Yes It Is!“ erhalten. Yii wurde am 1. Jänner 2008 ins Leben gerufen[11] und objektorientiert unter PHP entwickelt. Die Architektur von Yii basiert auf dem Model-View-Controller und dem DRY Prinzip. Das Framework zeichnet sich durch seine hohe Performance, lange Featureliste und eine große aktive Community aus. Durch die große Akzeptanz der Internetcommunity gibt es eine große Auswahl an Erweiterungen und Plattformen, die sich mit dem Framework beschäftigen. Um eine Yii basierte Applikation starten zu können, wird nur ein Webserver benötigt, der PHP 5.1 fähig ist. In folgenden Punkten werden die verwendeten Technologien und Programmierprinzipien erklärt.

- **Model View Controller (MVC)**[12] ist das Prinzip auf dem Yii basiert. Das Model-View-Controller-Prinzip hat das Ziel der Trennung der einzelnen Teile, um die Wiederverwendbarkeit zu erleichtern (Abbildung 3.3). Die Trennung besteht darin, dass im Model Daten gespeichert werden, der View diese anzeigt und der Controller die Kommunikation zwischen Model und View ermöglicht. [Winesett, 2010, Seite 7ff]
  - **Model**: Das Model repräsentiert die Daten, die dargestellt werden sollen. Die Präsentation und Steuerung ist unabhängig von den Daten, somit können Modelle auf unterschiedlichen Systemen eingesetzt werden, ohne diese verändern zu müssen und können in anderen Projekten wiederverwendet werden.
  - **View**: Views stellen die Präsentationsschicht dar, Daten des Models werden in dieser angezeigt, Benutzereingaben entgegengenommen und an den Controller weiter geleitet.
  - **Controller** Der Controller steuert die verschiedenen Views, da es möglich ist, dass ein Controller mehrere Views besitzt. Vom Controller werden Benutzereingaben entgegen genommen und verarbeitet.

[11] http://www.http://www.yiiframework.com/about/ - letzter Aufruf 25.1.2015
[12] http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc - letzter Aufruf 25.1.2015

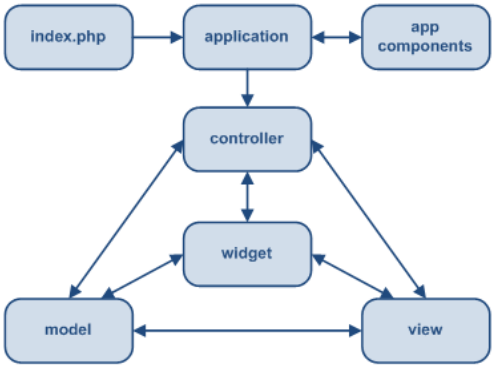

Abbildung 3.3: MVC Struktur von Yii[12]

- **Don't Repeat Yourself (DRY)** ist ein Prinzip, das Anwendung in der Programmierung findet. Es besagt, dass die Entwicklerin oder der Entwickler Wiederholungen des selben Programmcodes vermeiden soll. Wiederholungen machen die Wartung des Codes sehr schwierig, da jedes Vorkommen dieses Codes geändert werden muss, ansonsten kann es passieren, dass das ganze Programm nicht mehr richtig funktioniert. [Hunt & Thomas, 2003, Seite 24ff]

- **PHP** (steht für Hypertext Preprocessor) ist die Programmiersprache in der Yii entwickelt wurde und ist eine serverseitige Skriptsprache, die für das Web entwickelt wurde, um dynamische Webseiten zu generieren. Der Code kann dabei direkt in eine HTML-Seite integriert werden und der Webserver interpretiert den PHP Code mit einem PHP Modul. Dieses generiert die dynamische Webseite. PHP ist eine weit verbreitete Programmiersprache im Bereich der Webentwicklung. Der große Vorteil liegt darin, dass PHP eine sehr gute Datenbankanbindung bietet und die meisten Datenbanksysteme unterstützt. Ursprünglich war PHP nur eine Skriptsprache, wurde jedoch ständig erweitert, sodass auch objektorientierte Strukturen Einzug in PHP hielten[13]. Dadurch ist es in PHP möglich objektorientiert zu programmieren. Durch die große

[13]http://php.net/manual/de/intro-whatcando.php - letzer Aufruf 25.1.2015

Verbreitung gibt es sehr viele Programmbibliotheken, die eine weitere Verbreitung der Programmiersprache nur weiter fördern.
[Lengstorf & Hansen, 2014, Seite 4]

- **Web-Services** Yii macht es sehr einfach, Webservices zu realisieren. Einfache Schlüsselwörter markieren Funktionen und machen diese als Webservice verfügbar. Dazu wird das SOAP Protokoll verwendet. Yii kann ohne Erweiterungen und zusätzlichem Programmieraufwand die SOAP Anfragen beantworten und übermittelt die Daten an den Client zurück.
  Nicht nur SOAP Webservices können realisiert werden, auch auf JSON (JavaScript Object Notation) basierende Webservices können sofort zur Verfügung gestellt werden. Der Vorteil von JSON als Format besteht darin, dass es effizient verarbeitet werden kann. Das durch ein SOAP Webservice zurückgelieferte XML ist meist aufwändiger zu verarbeiten.

- **JSON** steht für JavaScript Object Notation und ist ein sehr einfaches Datenformat, das auf der Syntax von JavaScript basiert. Es werden die Daten mit Array und Objekt Datentypen repräsentiert. Durch diese einfache Formatierung lassen sich die Daten viel effizienter verarbeiten als auf XML basierte Datenformate. [Zakas et al., 2007, Seite 237]

- **SOAP** ist ein standardisiertes Verfahren, um Daten zwischen Applikationen auszutauschen. Strikt gesehen ist SOAP eine spezielle Form von XML, die festlegt, in welchem Format die Daten übertragen werden. Es wird auch festgelegt, wie spezifische Datenformate übermittelt werden müssen, damit sie verwendet werden können. Durch diese Eigenschaften eignet sich SOAP sehr gut, um Webservices zu realisieren. [Snell et al., 2001, Seite 15]

- **JavaScript** JavaScript ist eine Skriptsprache, die in der Webentwicklung zum Einsatz kommt. Hier wird sie verwendet um Webseiten interaktiv zu gestalten. Dies ist durch die clientseitige Ausführung möglich. Es ist möglich, die angezeigte Webseite zu verändern oder auch Teile der Webseite asynchron nachzuladen. Das Yii-Framework verwendet JavaScript für die clientseitige Validierung von Benutzereingaben und das asynchrone Laden von Teilen der einzelnen Views. Unter Yii kommt

die sehr bekannte JavaScript-Bibliothek jQuery zum Einsatz. Diese bietet sehr viele Funktionen zur Manipulation der Webseite und unter anderem baut die gesamt AJAX-Unterstützung von Yii auf jQuery auf. [Zakas, 2012, Seite 1ff]

- **AJAX** Für die dynamischen Aktualisierungen der Webapplikation setzt Yii auf AJAX. AJAX steht für die Wortfolge „Asynchronous JavaScript and XML". Bei diesem System handelt es sich um ein Konzept, das es ermöglicht, asynchrone Datenübertragung in Webbrowsern zu nutzen, um zum Beispiel nur bestimmte Teile einer Webseite und nicht wie üblich die gesamte Seite neu zu laden.[Zakas et al., 2007, Seite 5]

- **MySQL** Als Datenbank kann Yii unterschiedliche Systeme verwenden. In der Umsetzung der Webapplikation Lesetrainers wurde hierbei auf MySQL gesetzt. *„MySQL ist die populärste Open-Source-Datenbank der Welt."*[14] Es handelt sich hierbei um eine Open-Source-Software. Im Fall von MySQL bedeutet das, das die nicht kommerzielle Nutzung kostenlos ist. MySQL wurde im Jahr 1994 entwickelt und seitdem ständig weiterentwickelt. Bevorzugt wird MySQL zusammen mit dem Apache Webserver und der Skriptsprache PHP eingesetzt. Sehr viele erfolgreiche Unternehmen nutzen MySQL, wie zum Beispiel Google[15] oder Facebook[16]. [Tarr, 2011, Seite 239] [King et al., 2002]

3.3 Ember.js[17]

Um die Lesetests auch für Benutzerinnen und Benutzer ohne iPads verfügbar zu machen, wurde eine Webversion der drei Applikationen entwickelt. Dafür wurde eine geeignete Technologie gesucht, die möglichst wenig Änderungen am ursprünglichen System mit sich zieht. Es wurde ein clientseitiges

[14] http://www.mysql.de/products/ - letzer Aufruf 25.1.2015
[15] http://www.informationweek.com/google-releases-improved-mysql-code/d/d-id/1054428? -letzter Aufruf 25.1.2015
[16] https://www.facebook.com/notes/facebook/keeping-up/7899307130 - letzer Aufruf 25.1.2015
[17] http://emberjs.com/guides - letzer Aufruf 25.1.2015

Model-View-Controller Framework gesucht, mit dem es möglich ist, auf das bestehende Webservice zuzugreifen. Eine weitere Forderung war eine möglichst große Plattformunabhängigkeit, um eine große Verbreitung zu erreichen. Die Wahl fiel auf das JavaScript MVC Framework **Ember.js** es handelt sich dabei um ein Single-Page Applikations Framework. Dies bedeutet, das die gesamte Webapplikation nur aus einer einzigen HTML Seite besteht. Die anzuzeigenden Teile werden durch JavaScript Funktionen sichtbar gemacht oder dynamisch von einem Webservice geladen. Ember.js setzt auf JavaScript, die Handlebars Templatesprache und folgende Konzepte zur Erreichung der gewünschten Funktionalität:

- **Templates** sind Teile der Applikation, die in der Handlebars Templatesprache geschrieben werden. Sie beschreiben das Userinterface der Applikation. Jedes Template hat im Hintergrund sein eigenes Datenmodell, welches automatisch die Vorlage aktualisiert falls sich das Modell ändert. Außer HTML-Code kann die Vorlage auch folgende Elemente enthalten:
  - **Expressions** sind Variablen aus dem Datenmodell und können somit eingebunden werden.
  - **Outlets** sind Platzhalter für weitere Templates, die je nach Abhängigkeit der Benutzerin oder des Benutzers, andere Teile der Applikation anzeigen können. Die Anzeige dieser kann kaskadiert erfolgen.
  - **Componentes** sind benutzerdefinierte HTML Elemente, die genutzt werden können, um Wiederholungen von Code zu vermeiden.

- **Router**: Der Router übersetzt die aufgerufene URL in verschachtelte Template Aufrufe. Damit wird der gewünschte Teil der Applikation ausgewählt. Für jedes Template gibt es eine Route, die auf das Template verweist, das angezeigt werden soll.
- **Modelle** sind Objekte, die Daten für die ihnen zugeordneten Templates enthalten. In sehr vielen Applikationen werden die Daten eines Modells über Webservices geladen.

- **Controller** sind für die Programmlogik von Templates verantwortlich, sofern Templates eine solche Logik benutzen. In Falle von Ember.js wäre dies zum Beispiel eine Schaltfläche, die eine bestimmte Funktion implementiert.

- **Views** haben eine Sonderstellung in Ember.js. Die integrierten Templates, durch Handlebars realisiert, sind für sich gesehen ein sehr mächtiges Instrument. Deswegen ist es im Vergleich zu anderen MVC oder JavaScript Frameworks nicht notwendig, einen dedizierten View zu erstellen. Die Templates können die meisten Aufgaben eines Views übernehmen. Jedoch gibt es auch in Ember.js Situationen, in denen die Funktionalität der Templates nicht ausreicht und die Verwendung von dedizierten Views erforderlich macht. Dies ist der Fall, wenn man ein sehr kompliziertes und ausgeklügeltes Eventmanagment benötigt, oder wenn man eine wiederverwendbare Komponente generieren will.

4 Umsetzung des Prototypen

Die Umsetzung des Prototypen beinhaltet die Realisierung des Administrations- und Auswertungssystems, die Applikationen für das iPad sowie die webbasierten Versionen der Tabletapplikationen. Für die Umsetzung wurden die zuvor beschriebenen Technologien verwendet. Das Administrationssystem wurde mit dem Yii Framework, zusammen mit einer MySQL-Datenbank realisiert. Die Durchführung der Lesetests der Schülerinnen und Schüler wurde durch drei iPad Applikationen umgesetzt. Um eine einfache Bedienung zu gewährleisten, wurden die drei unterschiedlichen Lesetests in drei eigenständige Applikationen aufgeteilt. Eine Anforderung an die Applikationen war es, dass die Schülerin oder der Schüler auf einfachstem Weg die Lesetests durchführen kann. Weiters war es auch wichtig, dass die Lehrerin bzw. der Lehrer auf einen Blick erkennen kann, welche Art von Lesetest der Schüler durchführt, deswegen wurden drei farblich unterschiedliche Applikationen entwickelt, die sich in ihrem Kern wesentlich nur in Konfigurationsparametern unterscheiden. Die webbasierten Versionen der iPad Applikationen wurden mit dem Ember.js Framework realisiert, damit auch Schulklassen, die nicht mit iPads ausgestattet sind, die Vorteile dieses Systems nutzen können.

4.1 Administrations- und Auswertungssystem

Entwicklungsumgebung

Die Webapplikation wurde mit dem Yii Framework 1.1.12 realisiert. Der Entwicklungswebserver basiert auf Linux in der Distribution Debian mit Apache 2.2.22 als Webserver und PHP 5. Als Datenbanksystem kommt MySQL in der

Version 5 mit dem Tabellenformat InnoDB zum Einsatz. Als lokale Entwicklungsumgebung wurde Netbeans in der Version 8 verwendet. Die Java basierende Entwicklungsumgebung hat den Vorteil, dass sie plattformübergreifend benutzt werden kann.

4.1.1 Usermanagement

Durch die steigende Zahl an Lernapplikationen an der TU Graz, musste die Webapplikation die Voraussetzung erfüllen, sich in das bestehende Usermanangment einzugliedern. Dies hat den Vorteil, dass eine Registrierung den Zugang zu allen angebotenen Lernapplikationen bietet. Das Usermangament ist durch ein SOAP Webservice realisiert und stellt folgende Funktionen bereit:

- **getName**: Durch die mit der Anfrage mitgelieferte UserID kann das Webservice mit Vorname und Nachname der Benutzerin oder des Benutzers antworten.

- **getSchoolclasses**:Aufgrund der UserID werden die Klassen der Benutzerin oder des Benutzers zurückgesendet. Dies setzt voraus, dass die UserID von einer Lehrerin oder einem Lehrer stammt.

- **getStudentsToClass**: Die Webapplikation übermittelt die ID der Klasse und eine Liste der Schülerinnen und Schüler wird zurückgeliefert.

- **isUserAllowed**: Diese Funktion authentifiziert die Benutzerin bzw. den Benutzer im Usermanagement. Benutzername und Kennwort werden übermittelt und die Benutzerinformationen werden zurückgeliefert.

Um die Sicherheit der Authentifizierung zu gewährleisten, wird das Passwort der Benutzerin bzw. des Benutzers nur nach Anwenden der Hashfunktion SHA-256 übermittelt und zusätzlich noch ein Keyed-Hash Message Authentication Code hinzugefügt, um sicher zu stellen, dass die Nachricht am Weg zum Usermanagement nicht verändert wurde.
Die Webapplikation nutzt eine Wrapperklasse, die das Usermanagement vom System abstrahiert. Dies bietet den Vorteil, dass eine leichte Anpassung an

eine eventuelle Änderungen im Usermanagement möglich ist. Eine grundlegende Änderung der Authentifizierungsmethode zieht nur eine Anpassung in dieser Klasse nach sich. Eine Untersuchung der abgefragten Daten aus dem Usermanagement hat sehr schnell gezeigt, dass die Anzahl der unterschiedlichen Anfragen sehr klein ist. Weiters wurde festgestellt, dass die Daten im Usermanagement keinen ständigen Änderungen unterliegen. Dies führte zum Schluss, dass es möglich ist, die häufig benötigten Informationen zwischenzuspeichern. Für diesen Zweck wurde ein Cache implementiert, der die Arbeitsgeschwindigkeit seht stark erhöht. Zugriffszeiten von mehreren Sekunden konnten auf einige Millisekunden reduziert werden.

4.1.2 Datenbank

Die Datenbank der Webapplikation basiert auf dem MySQL-Datenbanksystem, zusammen mit dem Tabellenformat InnoDB. Die Datenbank beinhaltet neun Tabellen und verwendet *tbl_* als Tabellenprefix. (Abbildung 4.1)

- **Screening**: Diese Tabelle speichert die Grunddaten der Lesetests.
- **Tasks** beinhaltet die einzelnen Aufgaben des Lesetest. Der Identifier *type* gibt an, wie viele Antwortmöglichkeiten bei der Darstellung der Aufgabe angezeigt werden.
- **Deploy** stellt die Freigabe der Lesetests an die einzelnen Benutzerinnen und Benutzer dar, sie beinhaltet unter anderem den Aufgabenidentifier, den Benutzeridentifier und das Startdatum, den Zeitpunkt ab wann ein Lesetest durchgeführt werden kann.
- **Results** speichert die Auswertung des durchgeführten Lesetests jeder Benutzerin bzw. jedes Benutzers.
- **SingleResult** wird für die genau Auswertung der einzelnen Ergebnisse benötigt. Sie beinhaltet die Antwortzeit und die gegebene Antwort.
- **ScreeningHistory**: Diese Tabelle stellt den Verlauf der einzelnen Lesetests der verschiedenen Klassen dar.

- **SchoolGrade**: Hierbei handelt es sich um eine Lookuptabelle für die einzelnen Schulstufen, um eine Einteilung der unterschiedlichen Lesetests zu erreichen.

- **settings**: Hier werden die Einstellungen gespeichert, damit diese zentral geändert werden können.

Auffallend ist, dass keine persönlichen Informationen über die einzelnen Benutzerinnen und Benutzer gespeichert werden. Damit keine Versionsunterschiede entstehen und die Daten immer aktuell bleiben, werden Klasseninformationen und Benutzerinformationen immer neu aus dem Usermanagement geladen.

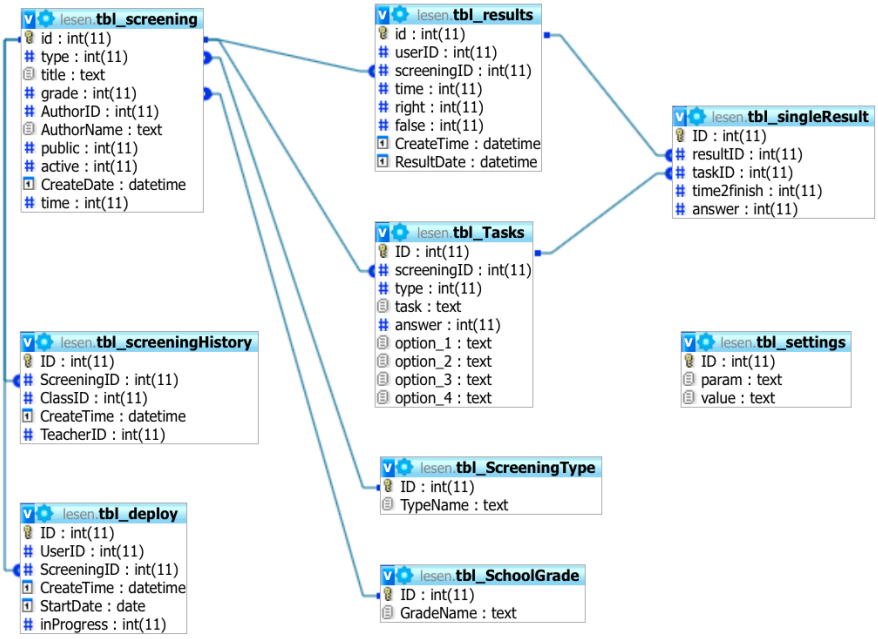

Abbildung 4.1: Datenbank Übersicht

4.1.3 User Interface

Die Webapplikation erfüllt zwei verschiedene Aufgaben, zum Ersten das Administrations- und Auswertungsmodul für Lehrerinnen und Lehrer. Zweitens bietet es für Schüler die Möglichkeit die Lesestests webbasiert durchzuführen. Hierbei wird eine Authentifizierung durchgeführt und die Schülerin bzw. der Schüler wird an die webbasierte Version der Tabletapplikationen weitergeleitet.

Administrations- und Auswertungsmodul

Dieser Teil des Informationssystems beinhaltet die Grundfunktionen für die Erstellung, Verteilung und Auswertung der Lesetests. Der Zugriff ist ausschließlich auf Lehrerinnen und Lehrer beschränkt. Folgende Funktionen werden bereitgestellt:

Hauptmenü

Das Hauptmenü stellt die höchste Navigationsebene des Administrationstools dar.

- **Meine Aufgaben**: Unter dieser Kategorie des Hauptmenüs sind die Funktionen für die Erstellung, Bearbeitung und Verteilung der persönlichen Leseaufgaben zusammengefasst.
  - **Alle Aufgaben**: Hier können alle im System gespeicherten, persönlichen Leseaufgaben angezeigt und an die eigenen Klassen verteilt werden (Abbildung 4.3). Es bestehen auch die Möglichkeiten zur Bearbeitung, Löschung und Duplizierung der Aufgaben.
  - **Lesegeschwindigkeits Aufgaben**: Auflistung aller persönlichen Lesegschwindigkeitsaufgaben.

  - **Leseverständnis Aufgaben**: Auflistung aller persönlichen Leseverständnisaufgaben

  - **Textverständnis Aufgaben**: Auflistung aller persönlichen Textverständnisaufgaben.

  - **Aufgabe erstellen**: Hier können die verschiedenen Leseaufgaben erstellt werden. Die Erstellerin oder der Ersteller kann festlegen, ob die kreierte Aufgabe auch von anderen Benutzerinnen und Benutzern verwendet werden darf. (Abbildung 4.2)

- **Aufgabensammlung** ist ein Pool an öffentlichen Aufgaben. Bei der Erstellung einer Leseaufgabe gibt es die Möglichkeit, der erstellten Aufgabe das Attribut *öffentlich* zu geben. Diese Eigenschaft legt fest, ob die Aufgabe in den Aufgabenpool aufgenommen wird und von jeder Benutzerin und jedem Benutzer für seine Zwecke verwendet und auch verändert werden kann.

- **Aufgabenverlauf** gibt den chronologischen Verlauf der einzelnen Leseaufgaben an, die von der Klasse durchgeführt wurden.
  - **Verlauf** zeigt die Liste der durchgeführten Aufgaben an.

  - **laufende Aufgaben** zeigt die verteilten Aufgaben an, die noch nicht von allen Schülerinnen und Schülern durchgeführt wurden.
- **Schüler** zeigt eine Liste aller Schülerinnen und Schüler, die der Lehrerin bzw. dem Lehrer zugeordnet sind und bietet die Funktion die, Auswertung der Schülerin bzw. des Schülers anzuzeigen.

- **Klassen**: Dieser Teil der Webapplikation zeigt eine Übersicht der einzelnen Klassen und die Auswertung der Lesetestergebnisse der gesamten Klasse. (Abbildung 5.1)

Aufgabenerstellung und Verteilung

Das System implementiert grundsätzlich drei verschiedene Arten von Lesetests, die sich in der Art der Fragestellung beziehungsweise in der Art der Antworten unterscheiden. Während die Schülerinnen und Schüler bei Lesegeschwindigkeits- und Leseverständnisaufgaben durch die Beschränkung der maximalen Aufgabendauer unter einem gewissen zeitlichen Druck stehen, bleibt dieser bei den Textverständnisaufgaben unbeachtet, da hier der zeitliche Faktor eine untergeordnete Rolle spielt.
Es gibt zwei Möglichkeiten Aufgaben zu erstellen:

1. Eine Lehrerin bzw. ein Lehrer kann eine Leseaufgabe komplett neu erstellen. Dazu muss nur die entsprechende Funktion gewählt werden, danach werden die Grundeinstellungen und die einzelnen Aufgaben erstellt. (Abbildung 4.2)

2. Eine weitere Möglichkeit zur Erstellung einer Aufgabe besteht darin, eine bestehende Leseaufgabe zu duplizieren und zu verändern. Entweder wird als Muster eine früher erstellte eigene Aufgabe verwendet oder die Lehrerin bzw. der Lehrer benutzt eine Aufgabe aus dem Aufgabenpool. In beiden Fällen ist die gleiche Vorgangsweise zu berücksichtigen: Zuerst wird die zu verwendende Aufgabe dupliziert und danach an die eigenen Bedürfnisse angepasst.

Die Verteilung der Aufgabe an die gewünschte Klasse wird durch Aufruf der Bereitstellungsfunktion initiiert. Die Lehrerin bzw. der Lehrer wählt die Klasse und ein Datum aus, ab welchem die Aufgabe durchgeführt werden kann. (Abbildung 4.3)

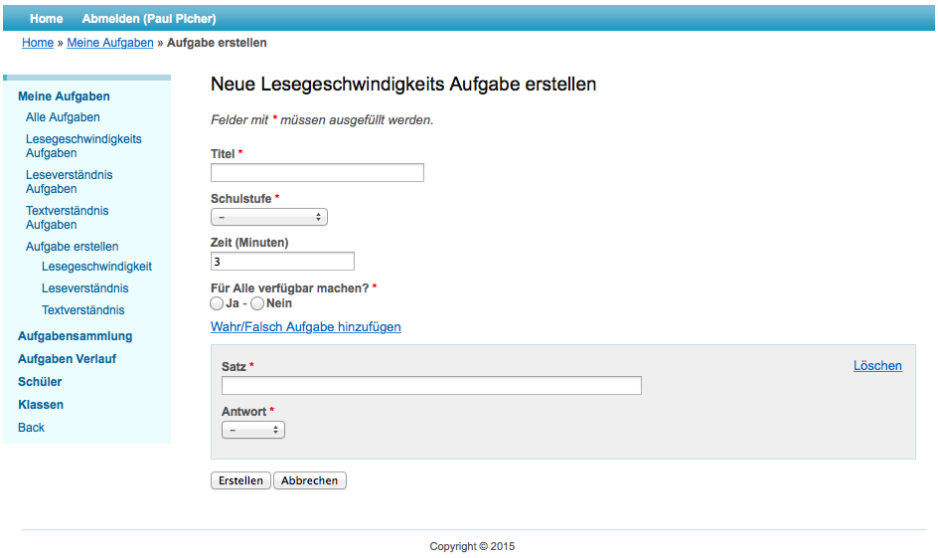

Abbildung 4.2: Lesegeschwindigkeitsaufgabe erstellen

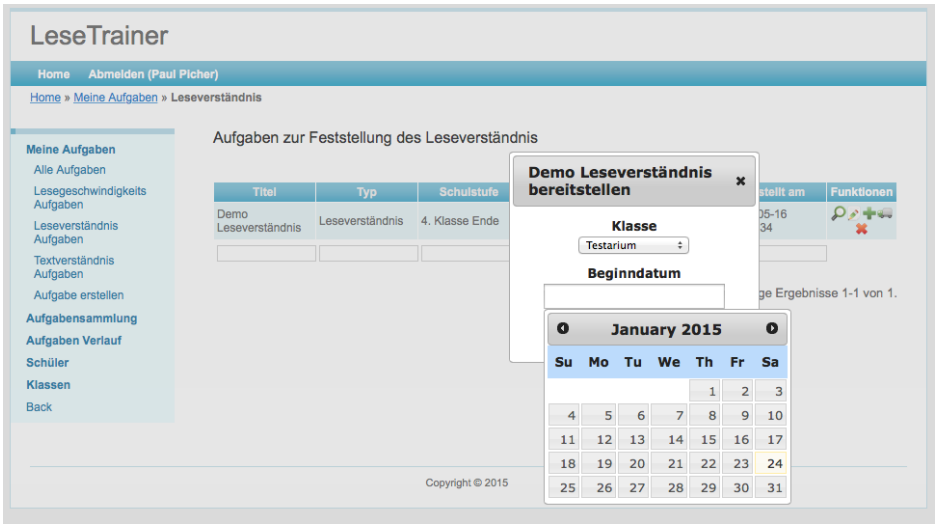

Abbildung 4.3: Leseaufgabe bereitstellen - Liste der Aufgaben im Hintergrund

4.1.4 Auswertung

Die Auswertung der Ergebnisse kann entweder für einzelne Schülerinnen und Schüler oder Klassen abgerufen werden. Die Ergebnisse werden als Prozent der richtigen Antworten angegeben. Das System ermittelt Durchschnittswerte von Resultaten, diese werden auf zwei Varianten berechnet, dem arithmetischen Mittel und dem Median mit Streuung. Der Vorteil des Medians liegt darin, dass statistische Ausreißer das Ergebnis nicht verfälschen. Die Streuung gibt die durchschnittliche Schwankung um den berechneten Mittelwert an. Als Beispiel: Eine Schülerin bzw. ein Schüler mit niedriger Streuung liefert eine sehr konstante Leistung bei den Lesetests ab.

Schüler

Die Auswertung zeigt die Ergebnisse der Schülerin bzw. des Schülers bei den unterschiedlichen Leseaufgaben in Prozent an. Neben den detaillierten Resultaten der Leseaufgaben, steht eine Auflistung der Antwortzeiten pro Teilaufgabe eines Lesetests zur Verfügung. Ein Diagramm zeigt den Verlauf der Ergebnisse an, und als Referenzwert wird das Durchschnittsergebnis der Klasse eingeblendet, um eine bessere Interpretation der erreichten Leistung zu ermöglichen. Damit auf den ersten Blick ersichtlich ist wie gut, die Ergebnisse einer Schülerin bzw. eines Schülers ausgefallen sind, werden die Ergebnisse farblich von Grün (100%) bis Rot (0%) hervorgehoben. (Abbildung 5.3)

Klassen

Die Auswertung der Klassen gestaltet sich ähnlich zu den Ergebnissen der Schülerinnen und Schüler. Die Klassenergebnisse errechnen sich aus den einzelnen Ergebnissen der Schülerinnen und Schüler als Mittelwerte der Resultate. Auch in dieser Ansicht kommt zur erleichterten Ansicht die farbliche Gestaltung in Abhängigkeit der Schülerergebnisse zur Anwendung. (Abbildung 5.1)

Auswertung des Lesetests

Eine Erweiterung der Klassenauswertung ist die Auswertung des Lesetests selbst. Hierzu zählen die durchschnittliche Antwortzeit, sowie die Anzahl der im Mittel richtig bzw. falsch und nicht beantworteten Fragen. Weiters wird angegeben, wie viele Schülerinnen bzw. Schüler zu wenig Zeit hatten, die Aufgabe zu komplettieren. Darüber hinaus wurde eine Analyse der Antwortzeit bzw. der gegebenen Antworten implementiert.

Die Analyse der Antworten stellt die gegebenen Antworten pro Frage gegenüber. Dieses Diagramm beantwortet die Frage *"Wie oft wurde Frage 3 richtig und wie oft wurde sie falsch beantwortet"*. Durch diese Interpretation ist es möglich festzustellen, welche Frage zu Problemen führte. (Abbildung 5.4)

Durch eine andere Interpretation der Daten ist eine Analyse bezüglich der Antwortzeiten möglich. Der Grund für diese Interpretation, ist es einen Anhaltspunkt zu finden, warum eine Frage falsch beantwortet wurde. Um diese zu realisieren, wird die im Mittel benötigte Zeit für die richtige bzw. falsche Beantwortung berechnet. Diese werden einander gegenübergestellt. Die Dauer bis eine Antwort gegeben wird, lässt Rückschlüsse auf die Schwierigkeit einer Fragestellung zu. (Abbildung 5.5)

4.1.5 Webservice

Um die Funktionen der Webapplikation auf dem iPad und auch in den Webversionen nutzen zu können, wurde ein Webservice integriert. Dieses kann über HTTP-Requests angesprochen werden und die Antwort wird in JSON codiert zurückgeliefert. Folgende Funktionen stehen zur Verfügung:

- **UserAuthenticate**: Hierbei handelt es sich um eine Weiterleitung an das Usermanagement, um die Benutzerin oder den Benutzer zu authentifizieren. Zusätzlich zu Username und Passwort wird noch ein Keyed-Hash Message Authentication Code übertragen.

- **saveResult**: Mit dieser Funktion können Ergebnisse in die Datenbank hochgeladen werden. Das Ergebnis wird in JSON codiert und per POST Request übertragen. Zur Sicherstellung, dass die gesendeten Daten nicht verändert wurden, wird auch hier ein Keyed-Hash Message Authentication Code hinzugefügt.

- **getScreeningList**: Aufgrund der übermittelten UserID und der ID der Leseaufgabe werden die durchzuführenden Aufgaben zurückgeliefert.

- **getScreening**: UserInformationen und Aufgabeninformationen werden in JSON codiert und per POST Request übermittelt. Das Webservice liefert die geforderte Leseaufgabe zurück.

- **getQuickScreening**: Hierbei handelt es sich um die Funktion, die für die „Schneller Test“ Funktion der iPad Applikation eine Leseaufgabe liefert. Diese wird am iPad ohne Anmeldung durchgeführt. Damit soll ein Eindruck über die verwendeten Lesetests vermittelt werden.

4.1.6 Ablaufdiagramme

Die folgenden Abbildungen zeigen den Ablauf einzelner Prozesse des Lesetrainers. Die Abbildung 4.6 zeigt die Speicherung eines Ergebnisses durch das Webservice. Das Erstellen eines Lesetests und das Bereitstellen an Klassen wird in den Abbildung 4.4 und 4.5 dargestellt.

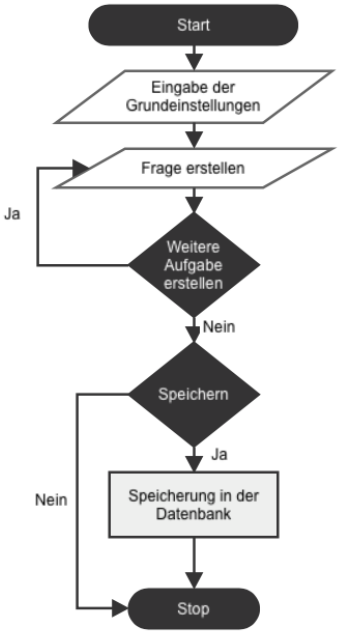

Abbildung 4.4: Leseaufgabe Erstellen

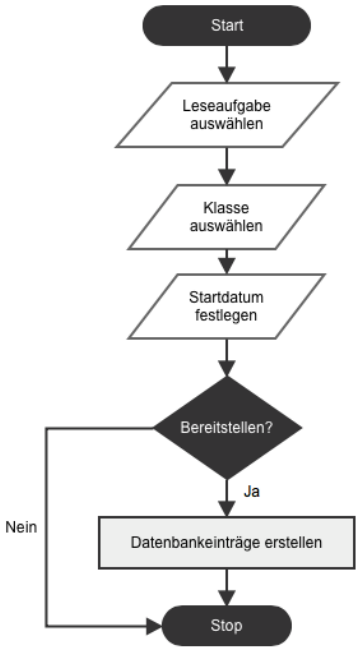

Abbildung 4.5: Leseaufgabe verteilen

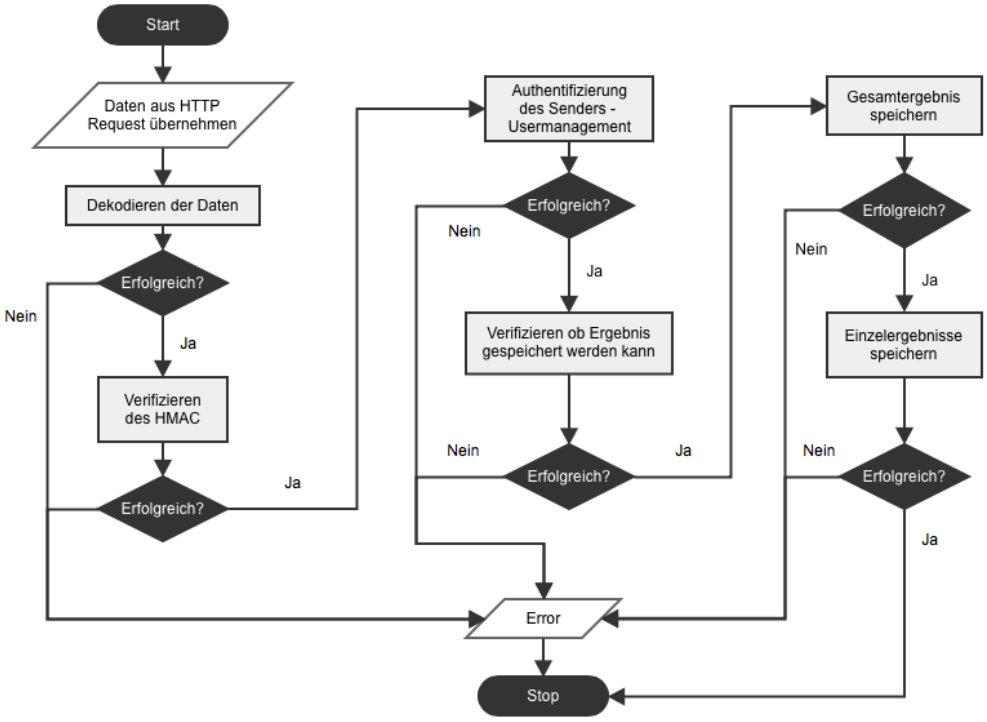

Abbildung 4.6: Ergebnisspeicherung - Webservice

4.1.7 Schwierigkeiten bei der Implementierung

Spezifische Probleme bei der Implementierung gab es im Bereich von AJAX. Die JavaScript-Funktionen werden über die ID des aufrufenden Objekts gebunden. Es kann dadurch zu multiplen AJAX-Requests kommen, da die ID innerhalb des Dokuments eindeutig ist. Bei wiederholter Aktualisierung des Dokuments kann es hier zu mehrfacher Bindung kommen. Dies ergibt zwar keine Probleme bei der Darstellung des Userinterfaces, jedoch kommt es zu erhöhtem Netzwerkverkehr und dadurch zu einer Verlangsamung des UserInterfaces. Im schlechtesten Fall kann es zu doppelten Einträgen in der Datenbank kommen. Dieses Problem kann durch Vergabe von eindeutigen IDs über Zufallszahlen oder Timestamps gelöst werden. Auch dieser

Lösungsansatz hat Schwächen. In speziellen Fällen kann es zum Versagen der Funktionalität führen: wenn die dynamisch generierte ID zu einem Objekt gehört, das durch AJAX aktualisiert wird. Das zu ersetzende Objekt kann deshalb im aktualisierten Dokument nicht gefunden werden und kann nicht angezeigt werden. Durch einen umhüllenden Container mit statischer ID wurde dieses Problem umgangen.
Durch die Designentscheidung, dass keine Userdaten gespeichert werden, kommt es zu erhöhtem Netzwerkverkehr. Bei jeder Ausgabe von Usernamen ist es notwendig, das UserManagement zu kontaktieren, um den Namen der Benutzerin oder des Benutzers anzuzeigen. Diese vielen Verbindungen werden durch einen Cache vermindert. Die gestellten Anfragen an das User-Management werden gespeichert und sind für eine gewisse Zeit gültig. Erst wenn die Gültigkeit überschritten wurde, wird eine neue Anfrage gesendet.

4.2 iPad App

Für die Durchführung der Lesetests wurde ein Client benötigt, der sich durch einfache Handhabung auszeichnet. Dafür wurde eine Applikation für das iPad entwickelt, bei der auf eine intuitive Bedienung und ein schlichtes Design, welches die Ablenkung der Schülerinnen und Schüler minimiert, geachtet wird. Ein weiterer Punkt, der Beachtung benötigt, ist die Abwärtskompatibilität. Es kann nicht davon ausgegangen werden, dass immer die neueste Version des Betriebssystems auf allen iPads installiert ist. Die Lesetrainer Applikationen wurden deshalb für die *iOS* Version 7 und höher entwickelt.

4.2.1 Frameworks [Apple Inc., 2014]

In iOS gibt es Funktionsbibliotheken, die sogenannten Frameworks, die ermöglichen es, dass nicht immer alle Funktionen mit der App kompiliert werden müssen. Diese sind in die Frameworks unterteilt und wie in Kapitel 3 beschrieben zu den einzelnen Schichten des Betriebssystem zuordenbar. Die folgenden Frameworks wurden während der Entwicklung eingebunden:

- **UIKit**: Dieses Framework stellt alle wichtigen Funktionen zur Verfügung, die notwendig sind, um eine App zu entwickeln. Angefangen von dem Applikations Management, über das Anzeigen der verschiedenen Views, bis hin zur Verwendung von Services wie zum Beispiel Twitter und Facebook sind enthalten. Jedoch stellt UIKit nicht nur die fundamentalen App-Funktionen zur Verfügung, sondern ermöglicht auch Zugriff auf die Hardware des Gerätes (Kamera, Geräteinformationen, Annäherungssensor,...) oder auf die Daten der Benutzerin oder des Benutzers (Bildersammlung).

- **Foundation**: Das Foundation Framework beinhaltet hauptsächlich Objective-C Wrapper für das Core Foundation Framework. Dieses wiederum beinhaltet elementare Strukturen und Funktionen, wie Datentypen, Threadverwaltung, URL und Stream Management. Somit stellt dieses Framework die Basisfunktionen für Datenmanagement und Servicefunktionen für iOS zur Verfügung.

- **QuartzCore**: Dies stellt das CoreAnimation Interface zur Verfügung und ermöglicht damit der Programmiererin oder dem Programmierer, die Animationsbibliothek von iOS zu verwenden. Mit dieser ist es auf einem sehr hohen Abstraktionslevel möglich, Animationen zu beschreiben und diese in der Applikation zu verwenden. Oftmals wird dieses Framework auch indirekt durch andere Klassen in die Applikation integriert.

- **CoreGraphics**: Mit dieser Programmbibliothek werden die 2D Grafikfunktionen in die App eingebunden. Es werden Funktionen für Kantenglättung, unterschiedliche Farben, Bilder und Koordinaten-Transformationen zur Verfügung gestellt. Diese Transformationen werden in der App verwendet, um benutzerdefinierte Animationen auf einen View anzuwenden.

- **SystemConfiguration**: Dieses Framework beinhaltet hauptsächlich die Funktionen, die genutzt werden, um festzustellen, ob das Gerät auf dem die Applikation läuft, Verbindung mit dem Internet hat. Weiters kann auch herausgefunden werden, ob die Internetverbindung über WLAN oder das Mobilfunknetz hergestellt ist.

4.2.2 ViewController

Die Entwicklung einer iOS-Applikation beginnt mit der Auswahl eines App-Templates. Hierbei kann aus *Master-Detail Application, OpenGL Game, Page-Based Application, Single-View Application, Tabbed Application, Utility Application* und *Empty Application* ausgewählt werden. Für den Lesetrainer wurde eine Single-View Application gewählt und diese entsprechend den Anforderungen erweitert. Durch diese Auswahl wird implizit festgelegt, welcher ViewController verwendet wird. ViewController stellen die Grundklasse dar, die benötigt wird um einen View anzuzeigen - jeder View besitzt einen ViewController. Die unterschiedlichen Arten von ViewControllern beeinflussen das Erscheinungsbild der Applikation. [Conway & Hillegass, 2011, Seite 127ff]

NavigationController

Der generierte ViewController wird um einen NavigationController erweitert, um die einzelnen Views anzuzeigen. Dieser stellt eine Art Stack dar, auf den die einzelnen ViewController gelegt werden und der im Speicher an oberster Stelle liegende wird angezeigt. Eine Titelleiste implementiert einen Back-Button, dieser entfernt den aktuellen ViewController vom Stack. Die Initialisierung des Navigation Controllers erfordert die Angabe eines ViewControllers, der als RootViewController eingesetzt wird und im Stack zum ersten ViewController wird. [Conway & Hillegass, 2011, Seite 203ff]

4.2.3 Webservice

Durch den Einsatz von Webservices wurde eine ganz neue Art von Applikationen möglich. Vor der Benützung solcher Services, waren Applikationen auf die mitgelieferten Inhalte beschränkt, neue Inhalte konnten nur über ein Update nachgeliefert werden. Die Möglichkeit, Inhalte dynamisch zu liefern, ermöglicht es nun immer auf dem neuesten Stand zu sein. Im Fall des Lesetrainers erlaubt es, immer die von der Lehrerin bzw. dem Lehrer erstellten Leseaufgaben zur Verfügung zu haben. Jedoch gibt es unter iOS keine direkte Implementierung von Webservices, es ist nötig alles manuell zu starten und zu verarbeiten. Es gibt unterschiedliche Formen von Webserservices, diese unterscheiden sich in dem Format, in dem Daten geliefert werden. Grundsätzlich zeichnen sich zwei Formate ab, entweder die Daten werden als XML geliefert oder als JSON Objekt. Die Interpretation der Daten muss vom Programmcode verarbeitet werden. Für auf XML basierende Datenübertragungen muss ein XML-Parser verwendet werden, dieser wandelt die Daten in eine Datenstruktur um. Im Falle des Lesetrainers wurde eine JSON basierende Datenübertragung verwendet, diese hat den Vorteil, dass die Umwandlung in eine Datenstruktur sehr effizient möglich ist, und die übermittelten Daten keine komplexen Strukturen aufweisen.

4.2.4 Datenstrukturen

Für die Speicherung der Userdaten und der Ergebnisse der Aufgaben sind spezielle Datenstrukturen entworfen und implementiert worden. Diese speichern nicht nur Daten sondern beinhalten auch Methoden zur Interpretation der Daten oder bieten die Möglichkeit, die Daten in einem gewissen Format auszugeben.

UserInformationen

Die Klasse UserInformation (Listing 4.1) speichert alle Daten der Benutzerin oder des Benutzers, sobald sich dieser bzw. diese authentifiziert hat. Die Klasse wurde als Singleton implementiert, damit diese nur einmal verwendet werden kann und somit immer die aktuellen Userinformationen beinhaltet. Zu jedem Zeitpunkt und an jeder Stelle des Programms können die Daten der Klasse abgerufen werden. Falls sich eine Benutzerin oder ein Benutzer abmeldet, werden die Userinformationen aus dem Speicher des iPads gelöscht.

```
1
2  #import <Foundation/Foundation.h>
3  @interface UserInformation : NSObject
4  {
5  @public
6      NSString    *_UserName;
7      NSString    *_FirstName;
8      NSString    *_LastName;
9      NSString    *_FullName;
10     NSString    *_pwdHash;
11     NSDate      *_LoginTimeStamp;
12     NSDate      *_backgroundTimeStamp;
13     NSUInteger  _UserID;
14     NSUInteger  _Rank;
15 }
16
17 @property (nonatomic, retain) NSString *_UserName;
18 @property (nonatomic, retain) NSString *_FirstName;
19 @property (nonatomic, retain) NSString *_LastName;
20 @property (nonatomic, retain) NSString *_pwdHash;
21 @property (nonatomic, retain) NSString *_FullName;
22 @property (nonatomic, retain) NSDate *_LoginTimeStamp;
23 @property (nonatomic, retain) NSDate *_backgroundTimeStamp;
24 @property (readwrite) NSUInteger  _UserID;
25 @property (readwrite) NSUInteger  _Rank;
26
27 +(UserInformation *)sharedInstance;
28 -(BOOL)isTeacher;
29 -(BOOL)isStudent;
30 -(void)logoutUser;
31
32 @end
```

Listing 4.1: UserInformation.h

ScreeningResult

Das Ergebnis einer Leseaufgabe wird durch die Klasse ScreeningResult (Listing 4.2) repräsentiert. Diese Klasse ist mit dem „NSCoding“ Protokoll implementiert worden, um es zu ermöglichen, die Daten dauerhaft auf dem iPad zu speichern. Das Ergebnis kann unter gewissen Umständen auch nach einem Neustart der Applikation gebraucht werden. Die Daten werden deshalb in den NSUserDefaults gespeichert. Diese Datenstruktur, die in jeder Applikation vorhanden ist, wird dafür verwendet, Einstellungen oder den Status einer Applikation zu speichern. Der leichte Zugriff auf diese

Datenstruktur war der Grund, warum sie zur Speicherung der Ergebnisse gewählt wurde. Eine Speicherung erfolgt nur, wenn keine Internetverbindung vorhanden ist, um das Ergebnis in der Webapplikation zu speichern. Neben dem gesamten Resultat, beinhaltet die Datenstruktur noch ein Dictionary, welches die Detailauswertung pro Teilaufgabe beinhaltet. ScreeningResult speichert zusätzlich noch den Benutzernamen und das Passwort in gehashter Form, damit keine Authentifizierung durch die Benutzerin bzw. den Benutzer nötig ist, um gespeicherte Ergebnisse hochzuladen. Die getJSON Methode der Klasse liefert eine Repräsentation des Ergebnisses in JSON, damit die Daten vom Webservice entgegengenommen werden können.

```
1  #import <Foundation/Foundation.h>
2  #import "Utilities.h"
3  @interface ScreeningResult : NSObject <NSCoding>
4  {
5  @public
6      NSUInteger _userID;
7      NSUInteger _screeningID;
8      NSUInteger _time;
9      NSUInteger _rightAnswer;
10     NSUInteger _falseAnswer;
11     NSString *_createTime;
12     NSString *_resultDate;
13     NSString *_userName;
14     NSString *_pwdHash;
15     NSMutableDictionary *_SingleResults;
16
17 }
18
19 @property (nonatomic, retain) NSString *_createTime;
20 @property (nonatomic, retain) NSString *_resultDate;
21 @property (nonatomic, retain) NSString *_userName;
22 @property (nonatomic, retain) NSString *_pwdHash;
23 @property (nonatomic, retain) NSMutableDictionary *_SingleResults;
24 @property (readwrite) NSUInteger _userID;
25 @property (readwrite) NSUInteger _screeningID;
26 @property (readwrite) NSUInteger _rightAnswer;
27 @property (readwrite) NSUInteger _falseAnswer;
28 @property (readwrite) NSUInteger _time;
29
30 -(NSString*)getJSON;
31 -(void)addSingleTaskResult:(NSInteger)TaskID time2finish:(NSInteger)
       time2finish right_or_false:(NSInteger)answer;
32 @end
```

Listing 4.2: ScreeningResult.h

4.2.5 Applikation

Wie bereits erwähnt, wurden drei verschiedene Applikationen entwickelt, diese unterscheiden sich nur in der Art der Leseaufgabe, die durchgeführt werden kann. Der grundsätzliche Aufbau der Applikationen ist gleich. Die Unterschiede bestehen aus einzelnen Parameter in den Grundeinstellungen und im Design, in dem die einzelnen Teilaufgaben angezeigt werden. Die App besteht aus insgesamt sieben Views. Die Applikation startet mit einem Startmenü (Abbildung 4.7). Folgende Funktionen stehen zur Verfügung:

- **Trainer**: Diese Schaltfläche startet den Lesetrainer und zeigt die Anmeldemaske an. (Abbildung 4.8)
- **Schneller Test**: Dieser Button lädt eine Leseaufgabe aus dem Webservice, startet diese und zeigt das Ergebnis an. Diese Funktion soll zeigen, wie eine Leseaufgabe abläuft, damit mögliche Interessentinnen und Interessenten eine Vorstellung vom Ablauf bekommen. (Abbildungen 4.9 4.10 4.11)
- **Hilfe**: Hier wird ein allgemeiner Hilfetext angezeigt, der die Funktion der Applikation und die dazugehörige Webapplikation kurz erklärt.
- **Information**: Folgt man diesem Button, werden Copyright Informationen zur Applikation angezeigt.

Abbildung 4.7: Hauptmenü - LesenSpeed

Abbildung 4.8: Loginmaske - LesenSpeed

Trainer

Hinter „Trainer“ verbirgt sich die Funktionalität, die mit der Webapplikation kommuniziert, die bereitgestellte Aufgaben vom Server herunterlädt und den Lesetest durchführt. Damit eine Leseaufgabe von der Benutzerin oder dem Benutzer durchgeführt werden kann, muss eine Authentifizierung stattfinden. Hierfür wird der Loginview (Abbildung 4.8) angezeigt. Die Applikation überprüft die eingegebenen Daten über das Webservice und bei erfolgreicher Anmeldung gelangt die Benutzerin oder der Benutzer zum ScreeningStartView. (Abbildung 4.12) Hier werden in einer Liste alle verfügbaren Leseaufgaben angezeigt und die Benutzerin oder der Benutzer kann auswählen, welche Aufgabe gestartet werden soll. Durch die Auswahl wird im Hintergrund die gesamte Leseaufgabe heruntergeladen und der Test vorbereitet. Nachdem die Vorbereitungen abgeschlossen sind, wird die Aufgabe gestartet und der ScreeningView angezeigt (Abbildung 4.9, 4.10, 4.11). Hier werden nacheinander die Teilaufgaben abgearbeitet und das Ergebnis in der Datenstruktur ScreeningResult gespeichert. Der Test ist beendet, wenn entweder alle Aufgaben bearbeitet wurden, oder im Falle der Lesegeschwindigkeits und Leseverständnisaufgabe, die Zeit abgelaufen ist. Nach dem Ende der Aufgabe speichert die Applikation das Ergebnis in der Webapplikation ab. Sollte keine Internetverbindung bestehen, wird das Ergebnis gespeichert und bei der nächsten Ausführung der Applikation wird versucht das Ergebnis hochzuladen.

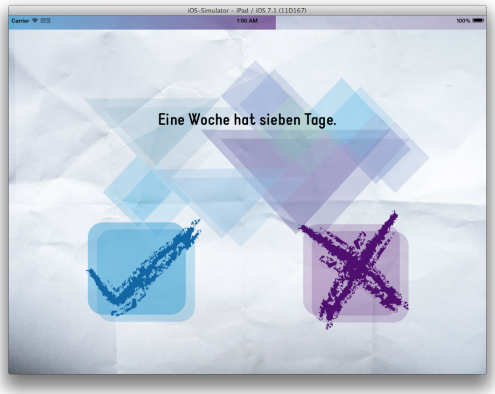

Abbildung 4.9: Aufgabe - LesenSpeed

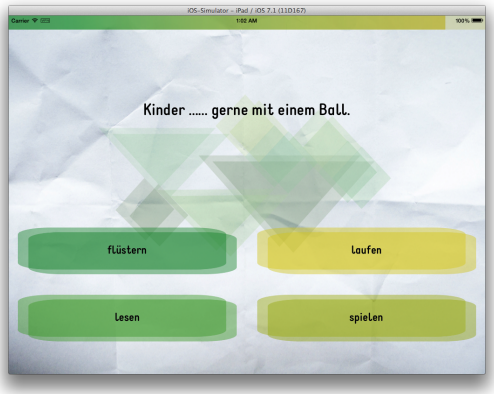

Abbildung 4.10: Aufgabe - LesenSatz

Abbildung 4.11: Aufgabe - LesenText

Abbildung 4.12: Aufgabenauswahl

4.2.6 Ablaufdiagramm

Folgendes Diagramm (Abbildung 4.13) zeigt den schematischen Ablauf der iPad Applikationen des Lesetrainers.

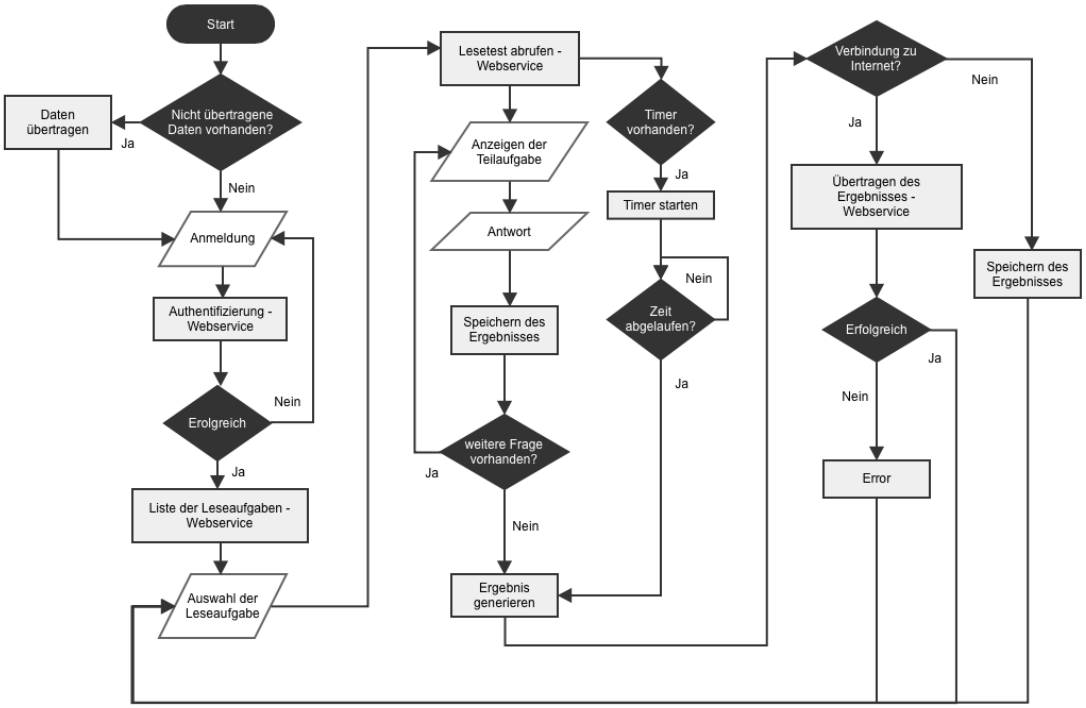

Abbildung 4.13: Ablaufdiagramm LesenSpeed - LesenSatz - LeseText

4.2.7 Implementierungs Details

URLConnection

Um eine HTTP-Verbindung aufzubauen, wird unter iOS die Klasse NSURLConnection verwendet. Dabei wird die Verbindung asynchron aufgebaut und danach werden die Daten empfangen. Bei der Implementierung kam es zu dem Problem, dass mehrere ViewController unter Umständen gleichzeitig eine Verbindung aufbauen können. Deswegen musste sichergestellt werden, dass die empfangenen Daten nicht vermischt werden. Als Lösung wurde die Klasse NSURLConnection um einen Identifier erweitert. Dies bedeutet, dass die Verbindung zusätzlich eine ID speichert damit beim Empfangen der Antwort die Daten richtig zugeordnet werden können.

RequestHandler

Damit nicht jeder ViewController die notwendigen Funktionen für eine Internetverbindung implementieren muss, wurde diese Funktionalität gekapselt und der RequestHandler implementiert. Diese, als Singleton implementierte Helferklasse handelt alle Verbindungen mit dem Webservice ab und liefert die Daten an die aufrufende Funktion zurück. Wenn die Klasse NSURLConnectionWithIdentifier einen asynchronen Request sendet, wird die Abwicklung an den Delegater übergeben und die aufrufende Funktion läuft weiter. Dieser Delegater (Listing 4.3) wird im RequestHandler implementiert. Hierbei werden alle Möglichkeiten abgehandelt: das richtige Zusammenführen der empfangenen Daten, das Weiterleiten der Fehlermeldungen falls Fehler auftreten und das Benachrichtigen der aufrufenden Funktion.
Wenn nun ein Request gesendet werden soll, muss die gewünschte Funktion des RequestHandlers aufgerufen werden. Die aufrufende Funktion registriert sich im Notificationcenter. Nachdem die Anfrage abgehandelt wurde, teilt dies der RequestHandler dem Notifactioncenter mit und übergibt die empfangenen Daten. Eine spezielle Funktion des RequestHandler arbeitet nicht mit asynchronen Verbindungen, sondern sendet synchrone Anfragen. Falls beim Start der Applikation noch nicht hochgeladene Ergebnisse vorhanden sind, werden diese in der Webapplikation gespeichert. Dies muss im Hintergrund sequentiell geschehen und deswegen wurden die Methoden

sendsaveResultRequestAppDelegate (Listing 4.4) und *uploadResultThreadMethod* (Listing 4.4) implementiert. Die erste übernimmt die hochzuladenden Ergebnisse, gespeichert in einem Array, und startet einen Thread mit der Funktion *uploadResultThreadMethod*. Dieser Thread lädt die Daten hoch und meldet eine Statusmeldung an den AppDelegater.

Die Methode *connectionDidFinishLoading* benachrichtigt das NotficationCenter, dass eine Anfrage an das Webservice erfolgreich abgeschlossen wurde und übergibt die empfangenen Daten. Gut zu erkennen ist, dass der ConnectionIdentifier festlegt, welche Benachrichtigung weitergegeben wird. (Listing 4.3 und 4.4)

```
1  -(void) connectionDidFinishLoading:(NSURLConnectionWithIdentifier *)
           connection {
2
3    switch ([connection._identifier intValue])
4    {
5     case WebAPIAuthenticate : {
6       NSMutableDictionary *jsonUserInfo = [NSJSONSerialization
                JSONObjectWithData:[_dataFromConnectionsByID objectForKey:
                connection._identifier] options:kNilOptions error:&error];
7       [self sendNotification:jsonUserInfo noticationName:@"
                AuthenticationNotification"];
8     } break;
9    case WebAPIgetScreening : {NSMutableDictionary *jsonScreening = [
              NSJSONSerialization JSONObjectWithData:[_dataFromConnectionsByID
              objectForKey:connection._identifier] options:kNilOptions error:&
              error];
10              [self sendNotification:jsonScreening noticationName:@"
                      ScreeningNotification"];
11    } break;
12        [...]
13 }
```

Listing 4.3: Auszug aus Request Handler URLConnection Delegater

```
1  -(void)sendsaveResultRequestAppDelegate:(NSArray *)savedResults
2  {
3      [NSThread detachNewThreadSelector:@selector(uploadResultThreadMethod:)
               toTarget:self withObject:savedResults];
4    }
5
6  -(void)uploadResultThreadMethod:(NSArray*) savedResults
7  {    [...]
8    for (int r_Iterator = 0; r_Iterator < [savedResults count]; r_Iterator++)
9    {
10     ScreeningResult *currentResult = [savedResults objectAtIndex:r_Iterator];
11     NSString *urlString = [[NSString alloc] initWithFormat: @"%@/saveResult&
               username=%@&hash_pwd=%@",[[NSUserDefaults standardUserDefaults]
               objectForKey:@"server_preference"],currentResult._userName,
               currentResult._pwdHash];
12
13     NSURL *url = [[NSURL alloc] initWithString:urlString];
14     NSMutableURLRequest *req = [[NSMutableURLRequest alloc] initWithURL:url];
15     NSString *msgLength =[NSString stringWithFormat:@"%d",[[currentResult
               getJSON] length]];
16         [req addValue:@"application/json" forHTTPHeaderField:@"Content-Type"
                   ];
17         [req addValue:msgLength forHTTPHeaderField:@"Content-Length"];
18         [req setHTTPMethod:@"POST"];
19         [req setHTTPBody:[[currentResult getJSON] dataUsingEncoding:
                   NSUTF8StringEncoding]];
20     [...]
21  [[NSNotificationCenter defaultCenter] postNotificationName:@"
           uploadDoneNotification" object:nil userInfo:webAPIresponse];
22  [NSThread exit];
23 }
```

Listing 4.4: Auszug aus Request Handler - Hintergrund Upload

AppDelegater

Im AppDelegater werden Funktionen zur Verfügung gestellt, die aufgerufen werden, wenn die Applikation ihren Status ändert. Ein Applikationsstatus ändert sich, wenn die App gestartet, beendet oder pausiert wird. Im Falle der Lesetrainer Applikationen bekommt der Appdelegater die Aufgabe, beim Starten der Applikation gespeicherte Ergebnisse hochzuladen (Listing 4.6) bzw. falls noch eine Benutzerin oder ein Benutzer angemeldet ist, diese bzw. diesen nach einer gewissen Zeit abzumelden. (Listing 4.5)

```
1  [...]
2  if ([[UserInformation sharedInstance] _LoginTimeStamp] != nil) {
3    NSDate *now = [NSDate date];
4    NSTimeInterval secondsSinceLogin = [now timeIntervalSinceDate:[(
           UserInformation*)[UserInformation sharedInstance] _LoginTimeStamp
           ]];
5   if ([[UserInformation sharedInstance] _backgroundTimeStamp] != nil)
6    {
7     NSTimeInterval secondsSinceBackGround = [now timeIntervalSinceDate:[[
            UserInformation sharedInstance] _backgroundTimeStamp]];
8     if (secondsSinceBackGround >= 60.0) {
9       [[UserInformation sharedInstance] logoutUser];
10      UIStoryboard *mainStoryboard = [UIStoryboard storyboardWithName:@"
              MainStoryboard" bundle: nil];
11      RTNavigationController *navControl = [mainStoryboard
              instantiateViewControllerWithIdentifier:@"NavigationView"];
12       [self.window setRootViewController:navControl];
13      }
14   }
15  }[...]
```

Listing 4.5: Auszug des AppDelegater - Logout User

```
1  [...]
2  NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
3  if ([defaults objectForKey:@"Result"]) {
4      NSData* savedResultsData = [defaults objectForKey:@"Result"];
5      NSArray *savedResults = [NSKeyedUnarchiver unarchiveObjectWithData:
           savedResultsData];
6      if (![savedResults count] == 0) {
7        [NSNotificationCenter defaultCenter] addObserver:self selector:
             @selector(uploadDone:) name:@"uploadDoneNotification" object:
             nil];
8        [[RequestHandler sharedInstance] sendsaveResultRequestAppDelegate:
             savedResults];
9       }
10 } else {
11   NSArray *savedResults = [[NSArray alloc] init];
12   NSData* savedResultsData = [NSKeyedArchiver archivedDataWithRootObject:
         savedResults];
13   [defaults setObject:savedResultsData forKey:@"Result"];
14   [defaults synchronize];
15 } [...]
```

Listing 4.6: Auszug des AppDelegater - Ergebnis Upload

Utilities

Die Helferklasse Utilities beinhaltet Funktionen, die öfters gebraucht werden, aber nicht in ein eigenes Objekt zusammengefasst wurden. Der Lesetrainer implementiert folgende Helferfunktionen:

- **sha256**: Diese Funktion nimmt einen beliebigen String als Input und berechnet den dazugehörigen SHA256 Hashwert. Dies wird verwendet, um das Passwort bei der Benutzerauthentifizierung nicht in Klartext zu übertragen.

- **hmac_message**: Diese Funktion berechnet den Keyed-Hash Message Authentication Code für die Kommunikation mit dem Webservice.

- **isInternetReachable**: Mit dieser Funktion wird überprüft, ob eine Verbindung zum Internet besteht, es wird die Klasse Reachability verwendet.

- **shakeView**: Dies ist eine einfache Funktion, die eine Animation auf den angegeben View ausführt.

4.3 Web Version

Damit nicht nur Benutzerinnen und Benutzer eines iPads den Lesetrainer nutzen können, wurde eine webbasierte Version des Lesetrainers implementiert. Der erste Ansatz versuchte den Lesetrainer mit dem bereits verwendeten Framework Yii zu realisieren, um eine direkte Integration zu gewährleisten. Dieser Ansatz wurde schnell verworfen, da Yii auf PHP basiert und somit eine serverseitige Ausführung darstellt. Der Lesetrainer muss clientseitig ausgeführt werden, da die Auswertung der Leseaufgaben auf Zeitmessungen basiert und diese nur clientseitig ausgeführt werden können. Durch diese Vorgaben wurden die Webversionen mit dem Ember.js Framework realisiert.

4.3.1 Entwicklungsumgebung

Als Entwicklungstool für die webbasierten Versionen der Tabletaktionen kam die Entwicklungsumgebung Eclipse zum Einsatz. Die Verwendung einer weiteren IDE kam durch die Gegebenheiten des Entwicklungsservers zu Stande. Die Anbindung an den Entwicklungsserver war nur mehr durch einen Remotezugang möglich und Eclipse bietet die Möglichkeit, direkt mit solchen Zugängen umzugehen, dies erleichterte die Entwicklung.

4.3.2 Framework

Ember.js ist ein Single-Page Framework, wie bereits erwähnt, werden Teile der Homepage dynamisch eingebunden und wieder entfernt. Dadurch ist die Dateistruktur des Lesetrainers sehr einfach, sie besteht aus jeweils einer HTML, CSS und JS Datei. Die restlichen Dateien beinhalten das Ember.js Framework, die jQuery Javascript Bibliothek und Kryptographie-Bibliotheken.

4.3.3 Webapplikation

Die Webapplikation implementiert alle drei Lesetests. Eine Auswahl der verschiedenen Lesetestapplikationen wird nach der Anmeldung angezeigt (Abbildung 4.14). Da es sich um eine Single-Page Applikation handelt, gibt es, wie der Name des Konzepts bereits erahnen lässt, nur eine HTML-Datei. In dieser werden die JavaScript-Dateien eingebunden, sowie die Templates für die Darstellung der einzelnen URLs angegeben. In den Templates kann man sogenannte Helper, dabei handelt es sich um Funktionen, die häufig verwendete Szenarien einfach realisieren, verwenden. Ein Beispiel ist die Anbindung an den Controller, wenn die Verarbeitung der eingegeben Daten eines Webformulars angestoßen werden soll. Allgemeiner ausgedrückt geschieht dies, wenn HTML-Elementen Actions zugeordnet werden sollen. So ermöglichen die Helper auch den einfacheren Umgang mit dem Datenmodel. Der „each-Helper“ ermöglicht es, alle Elemente des Datenmodels anzuzeigen. Der Lesetrainer verwendet diese Funktion zur Anzeige der möglichen Leseaufgaben, die von der Benutzerin bzw. dem Benutzer durchgeführt werden können.

Eine besondere Bedeutung besitzt das Element „outlet“ in einem Template. Jede Ember.js Applikation muss mindestens ein Vorkommen dieses Keywords beinhalten. Die Templates werden in einer hierarchischen Struktur angezeigt, beim Start der Ember.js-Anwendung wird automatisch das „ApplicationTemplate“ angezeigt (Listing 4.7). Dieses bildet die Grundlage für alle anderen Templates. Wird nun ein weiterführender Link - ein weiteres Template soll angezeigt werden - aufgerufen, muss die Applikation wissen, an welcher Stelle das neue Template eingesetzt werden soll. Diese Stelle wird mit dem Keyword „Outlet“ angegeben. Verfügt ein Template über weitere verschachtelte Ansichten, muss wiederum für jede Unterebene ein Outlet angegeben werden, um eine korrekte Anzeige sicherzustellen.

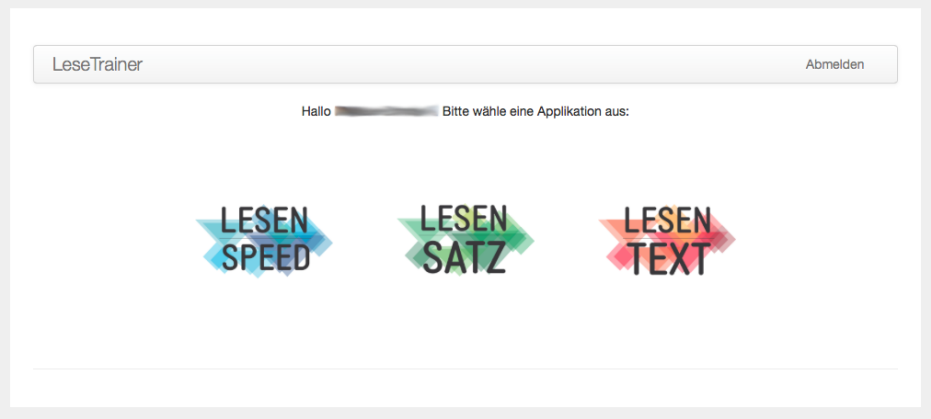

Abbildung 4.14: Auswahl in der Webversion - LesenSpeed - LesenSatz - LeseText

```
1  [...]
2
3  <script type="text/x-handlebars">
4   <div class="container">
5    <div class="navbar">
6     <div class="navbar-inner">
7      <a class="brand" href="#">LeseTrainer</a>
8      <ul class="nav pull-right">
9        <li>{{#linkTo 'logout'}}Abmelden{{/linkTo}}</li>
10     </ul>
11    </div>
12   </div>
13   {{outlet}}
14   <hr/>
15 </div>
16 </script>
17
18 [...]
```

Listing 4.7: Auszug der HTML Datei des Lesetrainer - Outlet - Handlebars Template

4.3.4 Ablaufdiagramm

Die Abbildung 4.15 gibt einen Überblick über die internen Abläufe der webbasierten Version des Lesetrainers. Diese implementiert alle drei verschiedenen Lesetests in einer Webapplikation.

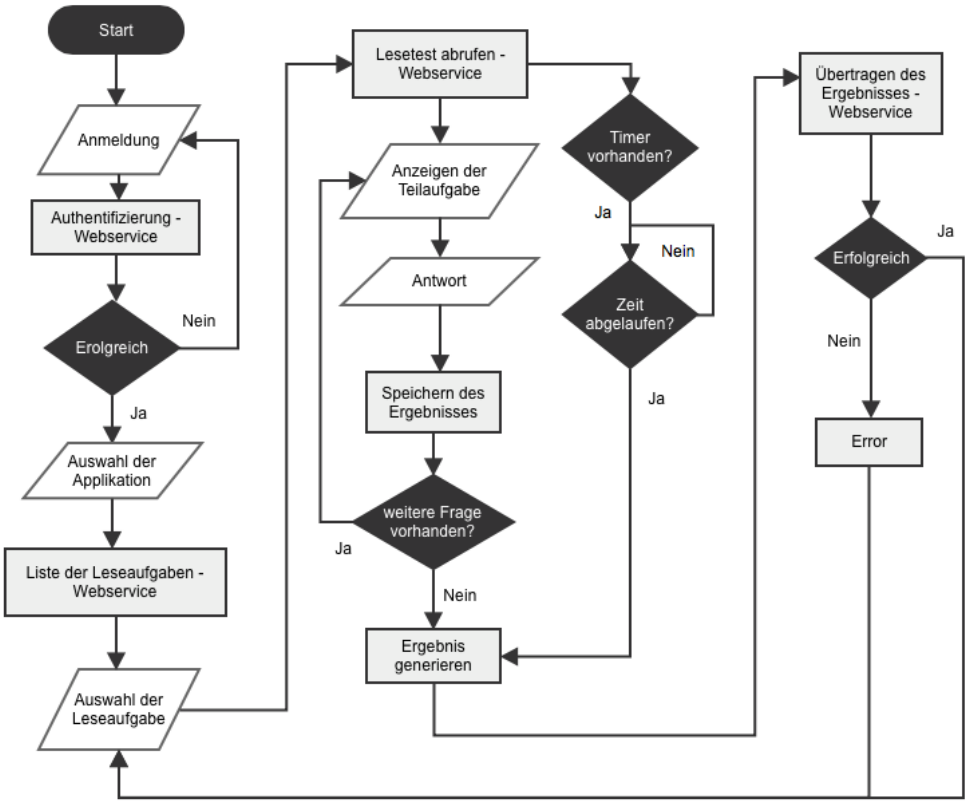

Abbildung 4.15: Ablaufdiagramm webbasierter Lesetrainer

4.3.5 JavaScript Implementierung

Eine Applikation in Ember.js hat vier Bestandteile: **Routemap**, **Routen**,**Controller** und **Views**. Die Routemap gibt an, welche URLs von der Ember.js Applikation verarbeitet werden können und welche Templates aus der HTML geladen werden sollen. Aus diesen Gründen muss diese immer implementiert sein.

Routen

Eine Route erfüllt hauptsächlich die Aufgabe, das Model für den Controller zu laden, um die Daten verarbeiten zu können. Eine weitere Aufgabe besteht darin, Initialisierungs-Funktionen auszuführen. Es ist nicht notwendig, jede Route zu implementieren. Falls es keine explizite Implementierung gibt, wird die Standardroute von Ember verwendet. Bis auf die Abmelderoute sind alle Routen im Lesetrainer von der selbst erstellten Superklasse *AuthenticatedRoute* abgeleitet. Diese hat die Aufgabe, den Zugriff nur nach erfolgreicher Anmeldung zu gewähren. Sie nutzt den speziellen Hookmechanismus von Ember, um eine Umleitung der gewünschten Anforderung durchzuführen. Der „Redirect-Hook“ überprüft bei jedem Aufruf, ob gültige Anmeldedaten gespeichert sind und falls dies nicht der Fall ist, wird die Benutzerin bzw. der Benutzer zur Anmeldeseite umgeleitet.
Wie bereits erwähnt, ist die eigentliche Aufgabe einer Route, das Bereitstellen des Models für die Controller der Applikation. Im Falle des Lesetrainers wird das Model vom Webservice geladen. Es wird das gleiche Webservice verwendet, das bei den iPad Applikationen die Daten liefert. Die Anforderung der Daten über den „Modelhook“ wird bei der Abfrage der möglichen Leseaufgaben durchgeführt. Hierfür wurde die abgeleitete Klasse der *WebAppRoute* erstellt, diese implementiert die Funktion „getTaskList(TypeID)“, welche die geforderte Liste aus dem Webservice abruft. Diese wird im „Modelhook“ aufgerufen und speichert die Daten im Model der Route. Die Notwendigkeit dieser Methode entsteht dadurch, dass es dem Lesetrainer möglich sein muss, über eine URL die geforderte Leseaufgabe eindeutig zu identifizieren. Diese Zuordnung erfolgt über den Identifier der Leseaufgabe selbst und dem Datum, an dem die Lehrerin bzw. der Lehrer die Aufgabe für die Schülerinnen und Schüler der Klasse freigegeben hat. Da die eigentliche

Leseaufgabe über eine URL, die diese Daten enthält, aufgerufen wird, ist es notwendig diese dynamischen Daten in die URL einzufügen. Diese Aufgabe übernimmt der „Serialize-Hook“der Route **TypeID*PlayRoute*. Diese Route implementiert auch den „setupController-Hook “. Dieser ist wie der Name vermuten lässt, dafür zuständig, den Controller zu initialisieren. Im Falle der „PlayRoute“werden die UserInformationen dem Controller übergeben, falls vorhanden werden von vorherigen Leseaufgaben die Daten gelöscht, die Model Daten an den Controller übergeben und Teilaufgaben und Einstellungen der durchzuführenden Leseaufgabe vom Webservice angefordert. Das (Listing 4.8) zeigt einige der beschriebenen Hookfunktionen.

```
1  App.SpeedPlayRoute = App.WebAppRoute.extend({
2
3    serialize: function(model) {
4     return {screeningID: model.screeningID,
5     DeployTime: model.DeployTime};
6    },
7
8    setupController: function (controller, model) {
9     console.log('Setting Up: Speed-Play-Controller');
10    this.setupUserInfo(controller);
11
12    controller.resetTestSettings();
13    controller.set('screeningInfo',model);
14    controller.getScreeningSettings();
15   },
16
17   redirect: function() {
18    this.redirectToLogin('speed.play');
19  },
20 })
```

Listing 4.8: Auszug des JavaScripts des Lesetrainers - SpeedPLayRoute

Controller

Controller sind im Grunde dafür zuständig, die Daten des Models für die Templates aufzubereiten bzw. Funktionen zur Verfügung zu stellen, die im Template benutzt werden. Der Lesetrainer implementiert zwei benutzerdefinierte Controller. Dies sind der *LoginController* und der *PlayController*. Der LoginController stellt die Funktionen zur Verfügung, die

ein Authentifizieren gegen das Webservice erlauben. Hierzu wurde die Login-Funktion erstellt, diese übergibt Benutzername und Passwort an das Webservice und interpretiert die zurückgelieferten Daten. Weiters implementiert der LoginController mehrere Observer-Funktionen. Diese stellen sicher, dass die erhaltenen Informationen aus dem Webservice, die lokal gespeichert werden, immer aktuell sind. Sobald der Controller neue Informationen erhalten hat, werden diese sofort in den LocalStorage des Browsers übernommen. In dem Listing 4.9 ist die Loginfunktion und ein Teil der beschriebenen Observer-Funktionen gut zu sehen.

```
1
2  accepted: localStorage.accepted,
3  acceptedChanged: function()
4   {
5    localStorage.accepted=this.get('accepted');
6   }.observes('accepted'),
7  [...]
8  login: function () {
9   var self = this;
10  var hmackey = Ember.get('App.HMACKey');
11  var hashPWD = CryptoJS.SHA256(this.get('password'));
12  var hmac = CryptoJS.HmacSHA256(this.get('username')+hashPWD,hmackey).
          toString();
13  var json = "{\"hmac\":\""+hmac+"\",\"username\":\""+this.get('username')+"
          \",\"hash_pwd\":\""+hashPWD+"\"}";
14  self.set('errorMessage', null);
15
16  Ember.$.post( Ember.get('App.baseUrl')+'UserAuthenticate',json).then(
          function(response) {
17
18    var jsonObj = Ember.$.parseJSON(response);
19     if (jsonObj.accepted) {
20     self.set('userid',jsonObj.UserID);
21     self.set('firstname',jsonObj.firstname);
22     self.set('lastname',jsonObj.lastname);
23     self.set('fullname',jsonObj.fullname);
24     self.set('accepted',jsonObj.accepted);
25     localStorage.setItem('loginTime',Math.floor(Date.now()/1000));
26     self.set('hashPWD',hashPWD);
27     self.transitionToRoute('index');
28    } else {
29     self.set('errorMessage','Benutzername oder Passwort ist nicht korrekt!');
30    }
31   });
32 },
```

Listing 4.9: Auszug des JavaScripts des Lesetrainers - Logincontroller Observer

Der zweite Controller, der generiert wurde, ist der *PlayController*. Dieser implementiert die gesamte Funktionalität, die notwendig ist, um einen Lesetest des Lesetrainers durchzuführen. Der Controller lädt die gewünschte Leseaufgabe und initialisiert die Datenstrukturen für die Speicherung der Ergebnisse. Eine der Hauptaufgaben bei der Erfassung der Ergebnisse ist, neben der gegebenen Antwort, die Messung der Antwortzeit. Er ist auch dafür zuständig, die Ergebnisse über das Webservice in der Datenbank zu speichern. Es gibt zwei Gründe für das Auslösen einer Speicherung: Entweder wenn alle Teilaufgaben abgearbeitet wurden, oder wenn die vorgegebene Zeit abgelaufen ist. Der letztere Grund kann bei den Textverständnisaufgaben nicht auftreten, da diese keiner zeitlichen Beschränkung unterliegen. Auf dem Listing 4.10 eines kleinen Teils des *PlayControllers* ist die Antwortfunktion zu sehen, welche die gegebene Antwort in der Datenstruktur speichert und entweder das Anzeigen der nächsten Teilaufgabe, oder das Sichern des Ergebnisses in der Datenbank triggert.

```
1  [...]
2    Answer: function(ans){
3      this.hideWebAppshowBusy();
4      this.setLoginTime();
5      this.stopTaskTimer();
6
7      var t2f = App.get('time2finish');
8      console.log('It took you '+(t2f*10)+' ms to answer the question!');
9
10     if (ans == this.get('currentAnswer')) {
11       var right = this.get('rightAnswers');
12       right++;
13       this.set('rightAnswers',right);
14       this.saveSingleResult(this.get('currentTaskID'),1,(t2f*10));
15     } else {
16      this.saveSingleResult(this.get('currentTaskID'),0,(t2f*10)); }
17
18     var iterate = this.get('currentTaskCount');
19     iterate++;
20     if (iterate < this.get('TotalTaskCount')) {
21       this.setNextTask();
22       this.startTaskTimer();
23       this.hideBusyShowWebApp();
24     } else {
25       this.saveResultToWebService();  }
26   },
27 [...]
```

Listing 4.10: Auszug des JavaScripts des Lesetrainers - Playcontroller Antwortfunktion

Views

Wie bereits in der Beschreibung von Ember.js erwähnt, wird die Anzeige von den Handlebars Templates übernommen. Diese mächtige Templatesprache macht den Einsatz von expliziten Views fast irrelevant. Es gibt nur wenige Gründe, die Views notwendig machen. Zu diesen gehören eine sehr ausgefeilte Behandlung von Benutzerevents oder, falls es gewünscht ist, eine wiederverwendbare Komponente zu erstellen.

Im konkreten Fall des Lesetrainers ist der Grund für die Verwendung eines Views ein anderer. Der View wird dazu verwendet, um nach Beantwortung und Speicherung des Teilergebnisses, die Anzeige der neuen Teilaufgabe zu initiieren. Dazu wird eine ObserverFunktion implementiert, diese überwacht die Controllervariable *currentTask*. Sobald eine Änderung in der Controllervaribale auftritt, wird eine jQuery Funktion aufgerufen, die den Inhalt des Views durch die neue Teilaufgabe ersetzt. Die Änderung im View veranlasst Ember.js, das gesamte Template neu zu rendern, dies führt dazu, dass die neuen Antwortmöglichkeiten automatisch angezeigt werden (Listing 4.11).

```
1 [...]
2 {{#view App.SpeedTask}}
3  <div class="speedtask">
4    <p id="speedtasktext">
5      {{currentTask}}
6    </p>
7  </div>
8 {{/view}}
9 [...]
```

Listing 4.11: Auszug der HTML Datei des Lesetrainers - View Definition

4.3.6 Probleme bei der Implementierung

Bei der Implementierung der JavaScript-Version des Lesetrainers kam es nur zu kleinen Schwierigkeiten. Ein kleines Problem stellte das Webservice dar, es wurde für die Verwendung des iPads entwickelt und dort war ein Hauptkriterium, dass eine Leseaufgabe auch zu Ende gebracht werden können muss, wenn die Internetverbindung nach dem Laden der Aufgabe nicht mehr besteht. Bei der JavaScipt-Version wurde aus Gründen der Fragmentierung, auf das Entwickeln eines zweiten Webservices verzichtet. Dies hatte die Folge, dass in der Applikation die gesamten Aufgabedaten verarbeitet wurden und nicht nach jeder beantworteten Teilaufgabe das Ergebnis an das Webservice zurückgeschickt und gespeichert wurde.

5 Evaluierung

Neben den selbst durchgeführten Funktions- und Usabilitytests während der Entwicklung, wurde eine Evaluierung des Systems im Zuge der Diplomarbeit *„Mobile Applikationen in der Grundschule“(Julia Schönhart,2015)* durchgeführt. Zwei Schulklassen einer Volksschule in Wien erklärten sich bereit, den Lesetrainer zu testen. Innerhalb eines Schuljahres wurden Lesetests durchgeführt. Dieser Test hatte zum Ziel, einerseits das System unter realen Bedingungen zu testen, auf der anderen Seite die Auswertung der erfassten Ergebnisse zu evaluieren.

5.1 Funktionstest im Zuge der Realisierung

Während der Umsetzung des Lesetrainers, wurden in jedem Stadium der Entwicklung Funktionstests durchgeführt. Sie beschränkten sich auf reine Tests der implementierten Funktionalität. Während der Implementierung des Administrationsinterfaces kam neben der reinen funktionalen Überprüfung auch ein Usabilitytest zum Einsatz.

5.1.1 Usabilitytest

Ein Usabilitytest in Form eines vereinfachten Thinking-Aloud-Tests wurde durchgeführt, um die einfache Benutzbarkeit des Administrationsinterfaces zu gewährleisten. Die Versuchspersonen führten vordefinierte Aufgaben durch. Hierzu wurde ihnen ein Computer mit bereits geöffnetem und angemeldetem Lesetrainer zu Verfügung gestellt. Der Testleiter las die durchzuführenden Aufgaben laut vor und die Versuchsperson versuchte, diese zu lösen und wurde gebeten, laut mitzusprechen, um den Gedankengang für den Testleiter

verständlich zu machen. Folgende Aufgaben wurden von den Probandinnen und Probanden durchgeführt:

1. Erstellen Sie eine Lesegeschwindigkeitsaufgabe mit drei beliebigen Teilaufgaben.
2. Erstellen Sie eine Leseverständnisaufgabe mit beliebig vielen Teilaufgaben.
3. Erstellen Sie eine Textverständnisaufgabe mit einer Teilaufgabe.
4. Übernehmen Sie einen beliebigen Lesetest aus der Aufgabensammlung und verändern Sie diesen.
5. Stellen Sie die von Ihnen erstellte Lesegeschwindigkeitsaufgabe für die Klasse „Screeningklasse“ ab dem heutigen Tag bereit.
6. Rufen Sie die Durchschnittsergebnisse der Klasse „Screeningklasse“ auf.
7. Rufen Sie die Detailstatistik einer beliebigen durchgeführten Leseaufgabe auf.
8. Finden Sie das durchschnittliche Ergebnis des Schülers „Muster Schueler“ heraus.
9. Rufen Sie die detaillierten Ergebnisse eines beliebigen durchgeführten Tests des Schülers „Muster Schueler“ auf.
10. Lassen sie den Schüler „Muster Schueler“, den von Ihnen erstellen Lesegeschwindigkeitstest wiederholen (Test wurde im Hintergrund durchgeführt).

Auswertung des Usabilitytests

Nach der ersten Durchführung des Usabililtytest wurde festgestellt, dass ein großes Problem die Navigation innerhalb des Systems darstellt. Die Testpersonen konnten die geforderten Aufgaben nicht durchführen bzw. war es ihnen nicht möglich, die gewünschten Funktionen schnell aufzufinden.

Eine Änderung des Navigationsschemas wurde implementiert und der Usabililtytest wurde wiederholt. Dieser ergab keine schwerwiegenden Fehler in der Benutzbarkeit. Die Testpersonen konnten alle die geforderten

Tasks absolvieren. Die gesamte Durchführungsdauer betrug im Mittel ca. 15 Minuten. Die Bearbeitungsdauer der einzelnen Teilaufgaben betrug im Durchschnitt nicht mehr als vier Minuten.

Die Einschätzung der Probandinnen und Probanden über die Benutzbarkeit des Systems im Vergleich zur ersten Version fiel positiv aus. Trotz der signifikanten Verbesserungen waren folgende Kritikpunkte aus der Auswertung der Tests ersichtlich.

- Die Detailauswertung der einzelnen Leseaufgaben ist nicht sofort auffindbar.
- Bei der Erstellung der unterschiedlichen Testaufgaben ist das Hinzufügen der einzelnen Teilaufgaben nicht sofort schlüssig.
- Die Erstellung der Testaufgaben wäre einfacher, wenn eine genaue Beschreibung der einzelnen Tests innerhalb der Erstellmaske vorhanden wäre.
- Teilweise zu verschachtelte Navigation.
- Verwendete Symbole manchmal nicht sofort richtig zuzuordnen.

Obwohl es noch Kritikpunkte an dem System bezüglich der Benutzbarkeit gab, wurden keine weiteren Änderungen an jener durchgeführt. Die vorgeschlagenen Verbesserungen waren nicht schwerwiegend genug, um eine komplette Umstrukturierung des Prototypen zu rechtfertigen. Weiters zeigte der Test, dass eine länger andauernde Testphase des Prototypen in der vorliegenden Version möglich ist.

5.2 Evaluierung anhand von realen Testdaten

Wie bereits erwähnt wurde der Lesetrainer von zwei Schulklassen einer Wiener Volksschule getestet. Dieser Test hatte zum Ziel, eine Kompletteinschätzung des Systems abzugeben. Nicht nur über das Administrationsinterface, sondern auch über die Tabletapplikationen. Der entscheidende Unterschied dieser

Evaluierung bestand darin, dass die Durchführung von der angestrebten Zielgruppe durchgeführt wurde und nicht von zufällig ausgewählten Testpersonen.

Insgesamt wurde an zwei Tagen in zwei Klassen je ein Lesegeschwindigkeits-, Leseverständnis- und Textverständnistest durchgeführt. Der erste Testtag zeigte keine Fehler in der Funktionalität. Die Authentifizierung der einzelnen Schülerinnen und Schüler, sowie das Abrufen der Lesetests, lief ohne auf den Lesetrainer zurückzuführende Probleme ab.

Die Testung der Tabletapplikation durch die Schülerinnen und Schüler förderte jedoch ein Usabilityproblem zu Tage. Die Schülerinnen und Schüler hatten teilweise Probleme sich am Lesetrainer mit ihren Zugangsdaten (Benutzername und Passwort) anzumelden. Die Schwierigkeit bestand darin, dass Schülerinnen und Schüler in der Volksschule nicht daran gewöhnt sind, ihr Passwort einzugeben, ohne es währenddessen zu sehen. Die Ersetzung der einzelnen Buchstaben durch einen Platzhalter war für einige neu.

Damit dieses Problem nicht wieder auftreten kann, wurde die Tabletapplikationen geändert. Es wurde ein Button implementiert, der es ermöglicht das Passwort in Klartext anzuzeigen, damit es den Schülerinnen und Schülern, bzw. den Lehrerinnen und Lehrern möglich ist, das Passwort zu überprüfen. Die Änderungen an den Tabletapplikationen konnten am zweiten Testtag bereits angewendet werden.

Es traten keine weiteren technischen Probleme bei der Durchführung der Lesetests auf.

5.2.1 Evaluierung der Auswertungen

Neben dem Funktions- und Usabillitytest war der hauptsächliche Grund für die Durchführung der Evaluierung in der Schule, eine Einschätzung der Aussagekraft der erfassten Werte und die daraus gezogenen Interpretationen zu erhalten.

Klassenauswertung

Der Lesetrainer gibt die erreichte durchschnittliche Leistung der Klasse an. Dies wird auf zwei Arten angegeben: einerseits als arithmetisches Mittel und andererseits als Median der erreichten mittleren prozentualen Leistung pro Lesetest. Zusätzlich wird die mittlere absolute Abweichung bezüglich des Medians berechnet, diese gibt die Streuung des ermittelten Ergebnisses an. Diese ist ein Indikator, wie unterschiedlich die einzelnen Resultate ausfallen. Es ist durchaus möglich, dass Median und Durchschnittswert stark voneinander abweichen, der Grund liegt in der Berechnung der beiden. Während der Median stark abweichende Ergebnisse nicht berücksichtigt, beeinflussen solche statistischen Ausreißer den Mittelwert sehr stark.

Abbildung 5.1: Klassenansicht - Durchschnittliches Ergebnis

Schülerauswertung

Als weitere Auswertung werden auch die Ergebnisse der Schülerinnen und Schüler aufbereitet. Die erreichten Leistungen werden der Klassenleistung gegenübergestellt. Dies ermöglicht eine Einschätzung der abgelieferten Leistung. Ohne Vergleichswerte ist keine fundierte Aussage über die erbrachte Leistung möglich, auch hier werden Mittelwert und Median als Referenzen für eine Gesamtleistungseinschätzung angegeben. Neben der visuellen Aufbereitung werden im unteren Bereich der Schülerstatistik, die Resultate in tabellarischer Form dargestellt.
Die Abbildung 5.2 zeigt eine Auswertung einer Schülerin bzw. eines Schülers, zu beachten ist hier, dass durch Angabe der Streuung die Konstanz der erbrachten Leistungen ersichtlich wird. Interessant an der Auswertung der Ergebnisse aus Abbildung 5.3 ist die signifikante Steigerung der erbrachten Leistung, die von der Schülerin bzw. dem Schüler erzielt wurde. Das Resultat des letzten Lesetests ist besser als das durchschnittliche Ergebnis der restlichen Klasse.

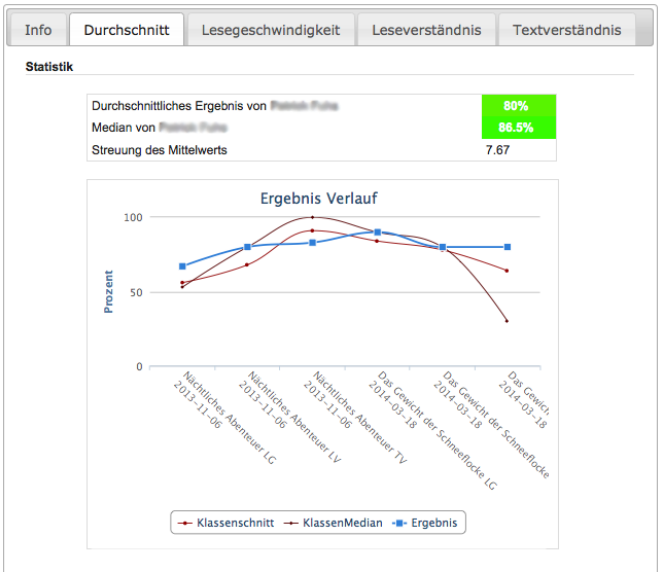

Abbildung 5.2: Schüleransicht - konstante Leistung

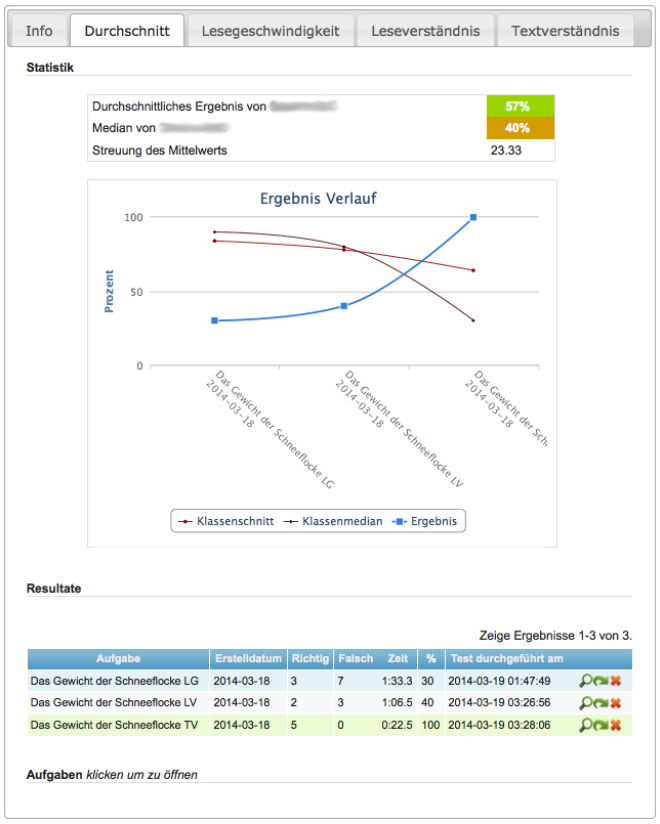

Abbildung 5.3: Schüleransicht - Verbesserung der Ergebnisse

Lesetestauswertung

Die vorgestellten Auswertungen geben Auskunft über die Leistungen der gesamten Klasse oder über die Resultate von einzelnen Schülerinnen bzw. Schülern. Doch daraus lassen sich keine Informationen über den Lesetest ableiten. Eine Gegenüberstellung der Anzahl der richtig, falsch und nicht beantworteten Fragen (Abbildung 5.4) zusammen mit einer Analyse der Antwortzeiten (Abbildung 5.5) gibt Aufschluss über die Schwierigkeit einzelner Teilaufgaben.

Von besonderem Interesse sind hier die Fragen 2 und 4, sie wurden von ähnlich vielen Schülerinnen und Schülern richtig und falsch beantwortet. Diese Information an sich zeigt, dass Schülerinnen und Schüler Schwierigkeiten mit den Fragen hatten. Die zusätzliche Information der Antwortzeiten lässt weitere Interpretationen zu. Die Antwortzeit für eine falsch gegebene Antwort liegt bei der 2. Frage bei ca. 20 Sekunden. Dies zeigt, dass Schülerinnen und Schüler lang überlegten, welche der Antworten die richtige ist. Das lässt darauf schließen, dass ein Verständnisproblem bezüglich dieser Frage vorliegt. Die 4. Aufgabe hingegen hat ähnliche Antwortzeiten bei richtiger und falscher Beantwortung, dies zeigt, dass die Schülerinnen und Schüler keine unterschiedlichen Zeiten zur Entscheidungsfindung benötigten. Der Grund für die Falschbeantwortung liegt demnach nicht an der Frage selbst. Eine falsch interpretierter Inhalt ist die wahrscheinlichste Begründung für dieses Ergebnis.
Die Evaluierung des Lesetrainers mit Hilfe des Usabiliytests und der Testung unter Realbedingungen im Rahmen von Schulklassen zeigt, dass der Prototyp seine Funktionen erfüllt.

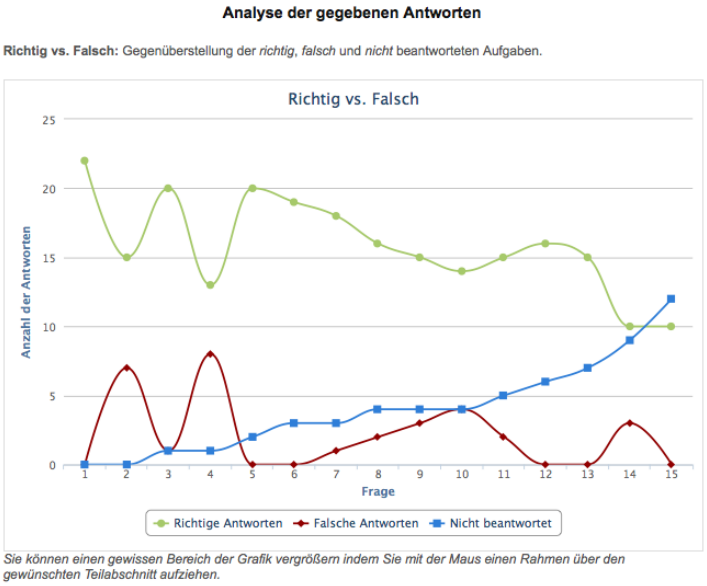

Abbildung 5.4: Leseaufgabe - Richtige, falsche und nicht gegebene Antworten

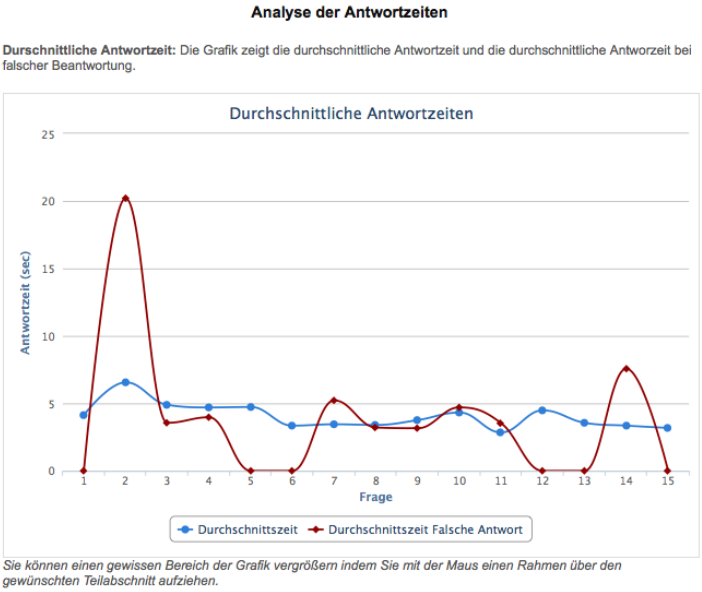

Abbildung 5.5: Leseaufgabe - Antwortzeitenanalyse

6 Diskussion

Der implementierte Prototyp des Lesetrainers ist ein erster Versuch ein System zu erstellen, das mit Hilfe neuer Technologien die Lesekompetenz von Kindern verbessert. Während der Umsetzung des Prototyps traten einige Probleme auf, die einer weiteren Diskussion bedürfen, doch auch einige positive Aspekte konnten beobachtet werden.

6.1 Hervorzuhebendes

Die unkomplizierte und schnelle Fertigstellung der Tabletapplikationen, sowie der ohne Probleme durchgeführte Funktionstest in einer Schulklasse zeigten, dass die entworfene Applikation keiner großen Veränderungen bedarf. Weiters kann die Auswertung und Erfassung der Testgrößen als gelungen bezeichnet werden. Die erfassten Daten lassen eine Einschätzung über die Lesefähigkeit der Schülerin bzw. des Schülers, sowie eine Evaluierung der Lesetests selbst zu. Mögliche Defizite, die eine Schülerin bzw. ein Schüler eventuell aufweisen, können schnell erkannt werden und mit gezielter Förderung kann ihnen entgegengewirkt werden. Probleme in der Fragestellung der Leseaufgaben können über die Auswertung der Antwortzeiten und dem Antwortverhalten erkannt werden.

6.2 Mögliche strukturelle Verbesserungen

Manche Designentscheidungen brachten oftmals Schwierigkeiten mit sich, doch waren sie nicht schwerwiegend genug, um eine komplette Umstrukturierung zu rechtfertigen.

Ein zu erwähnendes Beispiel ist die Datenbank, deren Design durch das Usermanagement begründet ist. Die Auswertung der Ergebnisse wird durch diese Entscheidung etwas komplizierter. Während der Generierung der Ergebnisse, Statistiken und Diagrammen kommt es, bei großer Anzahl von Schülerinnen und Schülern in einer Klasse, bzw. bei großer Anzahl von durchgeführten Leseaufgaben, zu längeren Wartezeiten, bis die Ergebnisse vollständig berechnet und angezeigt werden. Die Ursache für dieses Verhalten liegt in der Datenbank. Diese speichert keine Klasseninformationen zu den Ergebnissen einzelner Schülerinnen und Schüler. Dadurch wird es notwendig, dass für das Abrufen der Ergebnisse immer eine Suche nach den UserIDs der Klassenschüler innerhalb der gesamten Ergebnisliste mit sich zieht. Weiters ist es notwendig, das Usermanagement zu kontaktieren, um die UserIDs aller Klassenschüler zu erhalten. Durch eine verbesserte Struktur der Datenbank könnte die Berechnungszeit verringert werden und somit ein schnelleres und flüssigeres Arbeiten ermöglichen.

Ein weiteres Beispiel ist das Webservice, welches die Daten für die Tabletapplikationen und die webbasierten Versionen bereitstellt. Wie bereits erwähnt, ist die Tabletapplikation so ausgelegt worden, dass während der Durchführung des Lesetests keine Verbindung mit dem Internet bestehen muss. Auch nach Abschluss des Tests wurden bei nicht bestehender Verbindung die Ergebnisse gespeichert und bei der nächsten Aktivierung der Applikation an das Webservice übertragen. Dasselbe Webservice bedient auch die webbasierten Applikationen, dies führt dazu, dass der gesamte Test an den Client übertragen und verarbeitet wird. Ein eigenständiges Webservice für diese Versionen würde die clientseitige Verarbeitung verbessern.

6.3 Mögliche Verbesserungen

Eine Verbesserung des Lesetrainers ist ohne Zweifel möglich. Die Frage die sich dabei stellt, ist, in welchem Bereich eine Änderung sinnvoll ist. In Betracht gezogen werden muss eine Verbesserung der Kommunikation zwischen den Applikationen, wie bereits erwähnt eine Optimierung der Kommunikation selbst, weiters sollten Sicherheitsaspekte beachtet werden. Eine Verbesserung bezüglich der Datensicherheit und der Authentifizierung ist abzuwägen.

Ein im Detail durchgeführter Usabililitytest würde Erkenntnisse bezüglich möglicher Veränderungen am Administrationstool liefern. Eine Verbesserung der Navigation lässt sich immer erzielen, ohne genau Testung ist dies jedoch nicht möglich. Dies kann auch erweiterte Suchoptionen beinhalten oder auch ein Hinzufügen von weiteren Beschreibungsfeldern für Lesetests.

In späteren Versionen können andere Varianten der Lesetests implementiert werden, welche weitere Grundfertigkeiten des Lesens erfassen. Darüber hinaus wäre es möglich, andere Messparameter zu erfassen. Eventuell besteht die Option, über integrierte Kameras die Augenbewegungen der Leserin bzw. des Lesers zu erfassen und somit weitere Auswertungen der Lesefähigkeit zu realisieren.

Eine Erweiterung um eine soziale Komponente könnte möglich sein. Hierbei wäre ein eventuelles Diskussionsforum bezüglich der Lesetests oder allgemeineren Themen realisierbar. Ein Bewertungssystem oder Bestenlisten der öffentlichen Leseaufgaben, könnte die Testerstellerinnen und Testersteller motivieren, den besten Lesetest zu erstellen. Eine Community könnte neuen Benuterzinnen und Benutzern den Einstieg erleichtern.

6.4 Ausblick

Der nächste Schritt um eine fundierte Auskunft über die Lesekompetenz von Kindern zu erfassen, wäre eine genaue Untersuchung der einzelnen Lesetests. Weiters könnte eine Normierung der Tests in Betracht gezogen werden, um eine Standardisierung der Lesetests zu erreichen. Diese würde es erlauben, eine Vergleichbarkeit zwischen unterschiedlichen Tests zu gewährleisten.

Bezüglich des Lesetrainers selbst muss, eine Ausweitung der unterstützten Plattformen in Erwägung gezogen werden. Trotz der Möglichkeit der webbasierten Applikationen, könnte eine Ausweitung auf andere Tablet Betriebssysteme wie *Android* oder *Windows* einer weiteren Verbreitung des Lesetrainers förderlich sein.

Listings

Abbildungsverzeichnis

Literaturverzeichnis

[Adolph, 2009] Adolph, S. (2009). *Knuspels Leseaufgaben: Untersuchung zur Nützlichkeit eines Gruppenlesetests anhand der Lehrer einer bilingualen Schule*. Diplom.de.

[Apple Inc., 2014] Apple Inc. (2014). About the iOS Technologies. https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf. letzter Aufruf 25.1.2015.

[Breidbach, 2014] Breidbach, O. (2014). *Geschichte der Naturwissenschaften: I: Die Antike*. Geschichte der Naturwissenschaften. Springer Berlin Heidelberg.

[Conway & Hillegass, 2011] Conway, J. & Hillegass, A. (2011). *iOS Programming: The Big Nerd Ranch Guide (2nd Edition) (Big Nerd Ranch Guides)*. Big Nerd Ranch Guides.

[Hunt & Thomas, 2003] Hunt, A. & Thomas, D. (2003). *Der Pragmatische Programmierer.* Hanser Fachbuch.

[King et al., 2002] King, T., Reese, G., Yarger, R., & Williams, H. (2002). *Managing & Using MySQL: Open Source SQL Databases for Managing Information & Web Sites*. O'Reilly Media.

[Lee, 2012] Lee, W.-M. (2012). *Beginning iOS 5 Application Development*. Wrox.

[Lengstorf & Hansen, 2014] Lengstorf, J. & Hansen, T. (2014). *PHP for Absolute Beginners*. Expert's voice in Web development. Apress.

[Lenhard & Schneider, 2009] Lenhard, W. & Schneider, W. (2009). *Diagnostik und Förderung des Leseverständnisses*. Hogrefe Verlag.

[Lienert & Raatz, 1998] Lienert, G. A. & Raatz, U. (1998). *Testaufbau und Testanalyse*. Beltz, PsychologieVerlagsUnion.

[Macari, 2014] Macari, K. (2014). *Lesekompetenz und Förderarbeit: Die Bedeutung von Lesediagnose und Leseförderung bei Fünftklässlern*. Diplomica Verlag.

[Mayringer & Wimmer, 2002] Mayringer, H. & Wimmer, H. (2002). Salzburger Lese-Screening. http://www.eduhi.at/dl/Salzburger_Lesescreening_Handbuch.pdf. letzter Aufruf 25.1.2015.

[Möller, 2013] Möller, S. (2013). *Erfolgreiche Teamleitung in der Pflege*. SpringerLink : Bücher. Springer.

[Smyth, 2012] Smyth, N. (2012). *Objective-C 2.0 Essentials - Second Edition*. CreateSpace Independent Publishing Platform.

[Snell et al., 2001] Snell, J., Tidwell, D., & Kulchenko, P. (2001). *Programming Web Services With SOAP*. O'Reilly Media.

[Tarr, 2011] Tarr, A. (2011). *PHP and MySQL 24-Hour Trainer*. Wrox programmer to programmer. Wiley.

[Winesett, 2010] Winesett, J. (2010). *Agile Web Application Development with Yii 1.1 and PHP5*. Packt Publishing.

[Zakas, 2012] Zakas, N. C. (2012). *Professional JavaScript for Web Developers*. Wrox.

[Zakas et al., 2007] Zakas, N. C., McPeak, J., & Fawcett, J. (2007). *Professional Ajax, 2nd Edition (Programmer to Programmer)*. Wrox.