ulm university universität
uulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Konzeption und Realisierung einer web-basierten Anwendung zur systematischen Bewertung medizinischer und psychologischer Anwendungen

Abschlussarbeit an der Universität Ulm

Vorgelegt von:
Philipp Dörzenbach
philipp.doerzenbach@uni-ulm.de
849794

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Michael Stach
Robin Kraft

2019

Fassung 25. Juni 2019

© 2019 Philipp Dörzenbach

Satz: PDF-LaTeX 2ε

Kurzfassung

Mobile Gesundheitsapps haben das Potential, Patienten bei der Behandlung ihrer Leiden zu unterstützen. Leider ist die Güte solcher Apps für ihre Benutzer im Voraus kaum einzuschätzen. In den App-Stores werden diese zum Großteil anhand subjektiver Benutzer-Reviews bewertet, die nicht auf ihre Vertrauenswürdigkeit überprüft werden können. Aus diesem Grund wurde die objektive Bewertungsskala Mobile Application Ratings Scale (MARS) entwickelt, die eine reliable Qualitätsmessung der Gesundheitsapps ermöglicht.

Um den Bewertungsprozess zu optimieren, wurde im Rahmen dieser Arbeit ein System entwickelt, das die systematische Bewertung medizinischer und psychologischer Anwendungen mithilfe der MARS realisiert. Dabei wird der gesamte Reviewvorgang vom System abgedeckt. Dies beinhaltet das Suchen in den App-Stores und Übernehmen von relevanten Apps, das Verteilen der Reviews an Experten und Bewerten der Apps durch die Reviewer. Weiterhin bereitet das System den Import der Reviews in das Mobile Health App Database (MHAD)-Projekt vor, das seinen Benutzern ein übersichtliches Angebot geprüfter Gesundheitsapps bietet.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Im Folgenden wird die gegebene Problemstellung erläutert, die als Motivation für die vorliegende Arbeit dient. Des Weiteren wird die Zielsetzung der Arbeit definiert, sowie deren Struktur aufgezeigt.

1.1 Motivation

Mit der Einführung des Smartphones und dessen Etablierung innerhalb der vergangenen Jahre stieg auch die Nutzung von mobile health (m-Health) Anwendungen rasant an. Dabei können ihre Anwender auf eine Vielzahl von potenziellen Hilfsangeboten zurückgreifen: Bereits 2014 wurden über 100.000 Apps in den beiden relevantesten App-Stores iTunes und Google Play angeboten [2]. Diese Menge stellt die Nutzer allerdings vor die Herausforderung, die für ihre Beschwerden passende Anwendung zu finden. Als Auswahlkriterium werden dabei oftmals die subjektiven Benutzerbewertungen in den entsprechenden App-Stores angewandt. Diese sind als alleiniges Merkmal jedoch ungenügend, da sie nur wenig über die tatsächliche Qualität der App aussagen [5].
Um das Problem zu lösen, wurde die Mobile Health App Database (MHAD), eine Informationsquelle für Benutzer, Behandelnde und Leistungsträger, entwickelt. Die MHAD listet die Bewertungen zweier speziell geschulter Experten zu den m-Health Anwendungen auf. Besucher können gezielt nach Altersgruppe, Störungsbild, Behandlungsziel und weiteren Variablen filtern, um so die optimale Anwendung für ihr Leiden zu finden [28].
Für die objektive Bewertung der Experten wird die Mobile Application Ratings Scale (MARS) beziehungsweise deren deutsche Übersetzung, die MARS-G, verwendet. Die MARS ist eine 2015 von einem interdisziplinären Team der Technischen Universität Queensland entwickelte Skala zur Bestimmung der fachlichen Güte von Apps

mithilfe von Expertenbewertungen [21]. Die manuelle Bewertung gestaltet sich hierbei jedoch sehr zeitaufwendig. An dieses Effizienzproblem soll in der vorliegenden Arbeit angeknüpft werden.

1.2 Problemstellung und Zielsetzung

Der Bewertungsprozess wird aktuell mit Stift und Papier umgesetzt. Um die Mittelwerte der Skalen-Abschnitte (siehe Kapitel 2.2) zu berechnen, die für die Anbindung an die MHAD benötigt werden, müssen die Daten zusätzlich von Hand in ein weiteres System eingetragen werden. Die benötigten Applikationen werden den Experten ebenfalls gesondert zugeordnet. Eine IT-Unterstützung bei der App-Akquisition, der Verteilung der Anwendungen auf die Experten und des Ratingprozesses durch die Reviewer, wäre wünschenswert. Außerdem sollten die Bewertungen vom System für den Import in die MHAD vorbereitet werden. Ein solches System bündelt sämtliche Arbeitsschritte des Bewertungsprozesses und optimiert das Bereitstellen des Services für die Endnutzer.
Um ein solches System umzusetzen, müssen eine Reihe von Voraussetzungen geschaffen werden. Ziel dieser Arbeit ist es, ein System zu entwickeln, das den gesamten Ratingprozess optimiert. Dazu gehören das Durchsuchen der relevanten App Stores, das Verteilen der Apps auf Experten, eine Benutzerverwaltung mit Gruppen und Expertisen, die Einbindung der MARS und mehr. Eine ausführliche Auflistung der Systemanforderungen ist in Kapitel 3 zu finden.

1.3 Struktur der Arbeit

Diese Arbeit ist wie folgt aufgebaut: In Kapitel 2 werden zunächst die Ratingskala MARS und das Projekt MHAD erläutert. Darauf folgend sind in Kapitel 3 die Anforderungen an das zu entwickelnde System definiert, auf deren Basis die Systemarchitektur in Kapitel 4 erörtert wird. Kapitel 5 zeigt anhand ausgewählter Beispiele Problemfälle und deren Lösungen während der Implementierung auf. In Kapitel 6 erfolgt der Anforderungsabgleich, bevor die Arbeit in Kapitel 7 zusammengefasst und ein Ausblick auf die aufbauenden Arbeiten am System gegeben wird.

2 Grundlagen

Dieses Kapitel erläutert mit der MARS und MHAD die dem System zugrunde liegende Bewertungsskala und die Datenbank, die die Systemergebnisse weiter verarbeitet. Im folgenden werden die Motivation und der Aufbau der MARS erörtert, sowie das Angebot der MHAD für die Endnutzer der Apps skizziert.

2.1 Mobile Application Ratings Scale (MARS)

m-Health Apps haben das Potential, das Leben ihrer Anwender und deren Gesundheitsmanagement erheblich zu erleichtern. Als Gegenstand des täglichen Lebens sind Mobiltelefone die optimale Plattform für diese Unterstützung. Sie sind ohne großen Aufwand zu benutzen und ermöglichen den Benutzern Zugang zu Therapieangeboten unabhängig von aktuellem Standort, sozialer oder finanzieller Umstände. Leider existieren innerhalb der App-Stores keine objektiven Rating-Systeme, was vor allem mit Blick auf die Patientensicherheit ein erhebliches Problem darstellt. Jeder Benutzer des entsprechenden Stores kann eine Bewertung abgeben, sogar ohne Installation der App, was die Überprüfung auf Vertrauenswürdigkeit der Quelle nahezu unmöglich macht [14]. Eine Auswahl anhand der Popularität der App ist zusätzlich nicht zu empfehlen, da Benutzerbewertungen oftmals nicht mit den Expertenratings korrelieren [13]. Diese Umstände motivierten im Jahre 2015 ein interdisziplinäres Team der Technischen Universität Queensland die MARS zu entwickeln. Sie bietet eine objektive, reliable und valide Qualitätsmessung von Gesundheitsapps durch Experten. Die deutsche Version (MARS-G) wurde von Teams der Universitäten Ulm, Göttingen und Marburg übersetzt um die Anwendung der Skala auf deutschsprachige Apps zu erweitern. Im System, das im Rahmen dieser Abschlussarbeit entstand, ist die MARS-G implementiert.

Aufgebaut ist die Skala in drei Bereiche. Zuerst wird die App klassifiziert, indem beispielsweise technische Informationen der App, Sicherheit und Datenschutzaspekte, oder die geeignete Altersgruppe erfasst werden. Danach wird die App anhand 23 Fragen mit einer fünfstufigen Skala bewertet. Innerhalb dieses App Qualitäts-Rankings existieren Subskalen für Engagement, Funktionalität, Ästhetik, Information und Subjektive Qualität. Abschließend wird mit einer app-spezifischen Subskala die fachliche Meinung des Experten zum Potential zur Bewusstseins- und Verhaltensänderung gemessen. Der exakte Aufbau der Skala inklusive aller Fragen kann in Anhang B betrachtet werden.

2.2 Mobile Health App Database (MHAD)

Selbst wenn Gesundheitsapps von Experten auf Basis einer reliablen Skala bewertet werden, führt dies nicht sofort zu einem informierten Entscheidungsprozess der potentiellen Benutzer. Es ist wahrscheinlich, dass diese Bewertungen in der Masse an subjektiven Ratings von Laien kaum wahrzunehmen sind. Da die App-Stores selbst keine Filterung der Reviews durchführen und so die Benutzer kaum die Reliabilität der Ratings beurteilen können, bedarf es einer weiteren Plattform, die ausschließlich reliable Bewertungen bündelt. Diesen Service bietet die MHAD ihren Benutzern. Sie erlaubt es, die Apps verschiedener Themenbereiche zu durchsuchen um objektive und verlässliche Bewertungen zur Appgüte zu erhalten. Die Bewertung der einzelnen Apps basieren dabei auf den MARS-Ratings der beiden Experten. Zusätzlich sind alle Gutachten erst nach der Prüfung durch einen unabhängigen Editor veröffentlicht worden. Die Apps erhalten schließlich eine Gesamtbewertung auf Basis der Expertenbewertungen, die eine unkomplizierte Entscheidungsfindung ermöglicht. Bei Bedarf können für jede App die Mittelwerte der Kategorien Engagement, Funktionalität, Ästhetik und Information und Subjektive Qualität des MARS-Ratings eingesehen werden. Eine solche App-Übersicht ist in Abbildung 2.1 dargestellt.

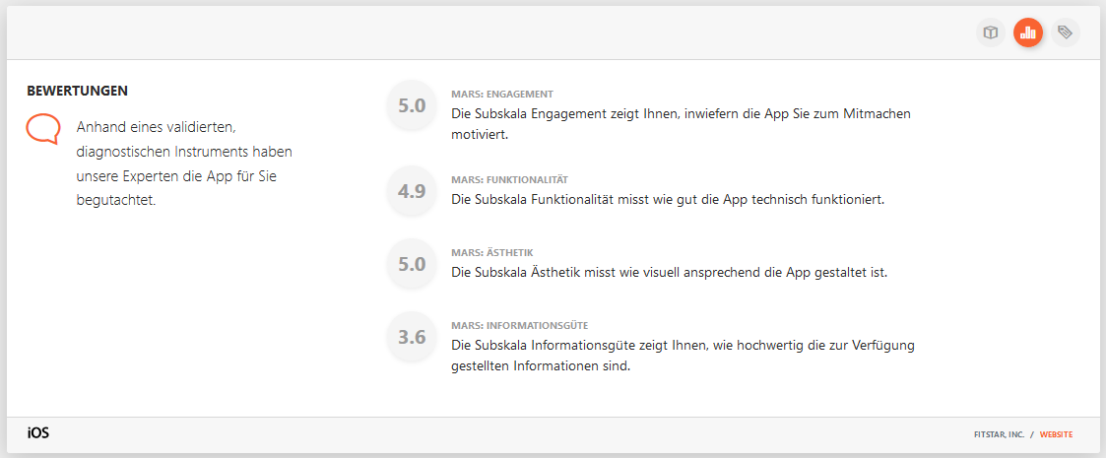


Abbildung 2.1: Übersicht einer Gesundheitsapp auf MHAD

3 Anforderungsanalyse

Das folgende Kapitel umfasst die Analyse und Definition der Anforderungen an das in dieser Arbeit entwickelten Anwendung. Dabei werden zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden. Um die Übersicht über die Menge der Anforderungen zu verbessern, werden diese in Module aufgeteilt dargestellt. Zu jeder Anforderung gehört ein Kürzel und eine Bezeichnung, anhand derer auf die Anforderungen in anderen Abschnitten verwiesen werden können. Außerdem erhält jede Anforderung eine Priorität gemäß der MoSCoW-Priorisierung [31].

Kürzel	**Bezeichnung**	**Priorität**	**Seite**
Account:			
F-01	Autorisierung	MUSS	10
F-02	Registrierung	MUSS	10
F-03	Authentifizierung	MUSS	11
F-04	Benutzerverwaltung	MUSS	11

Kürzel	Bezeichnung	Priorität	Seite
Benutzer:			
F-05	Administrator hinzufügen	MUSS	11
F-06	Administrator bearbeiten	MUSS	11
F-07	Administrator löschen	MUSS	11
F-08	Reviewer-Übersicht anzeigen lassen	SOLL	11
F-09	Reviewer hinzufügen	MUSS	12
F-10	Reviewer bearbeiten	MUSS	12
F-11	Reviewer löschen	MUSS	12
Gruppen und Kategorie:			
F-12	Gruppe hinzufügen	SOLL	12
F-13	Gruppe bearbeiten	SOLL	12
F-14	Gruppe löschen	SOLL	12
F-15	Kategorie hinzufügen	SOLL	12
F-16	Kategorie bearbeiten	SOLL	13
F-17	Kategorie löschen	SOLL	13
Suche:			
F-18	App-Stores durchsuchen	MUSS	13
F-19	Apps in Datenbank speichern	MUSS	13
F-20	App-Suche zwischenspeichern	SOLL	13
F-21	App-Suche wiederaufnehmen	SOLL	13
F-22	App-Suche exportieren	MUSS	14
App:			
F-23	App-Übersicht anzeigen lassen	MUSS	14
F-24	App bearbeiten	MUSS	14
F-25	App löschen	MUSS	14
F-26	Reviewer zuweisen (automatisch und manuell)	MUSS	14

Kürzel	Bezeichnung	Priorität	Seite
Review:			
F-27	Review-Konflikte anzeigen lassen	MUSS	15
F-28	Review-Übersicht anzeigen lassen	MUSS	15
F-29	App reviewen	MUSS	15
F-30	App ablehnen	MUSS	15
F-31	Review zwischenspeichern	SOLL	15
F-32	Review wiederaufnehmen	SOLL	15
System:			
NF-01	Zuverlässigkeit	MUSS	16
NF-02	Robustheit	MUSS	16
NF-03	Sicherheit/Datenschutz	MUSS	16
NF-04	Effizienz	SOLL	16
NF-05	Wartbarkeit	KANN	16
NF-06	Erweiterbarkeit	SOLL	16

3.1 Funktionale Anforderungen

Die funktionalen Anforderungen definieren die Funktionalitäten, die die Anwendung leisten können soll. Diese Anforderungen beeinflussen zum Großteil die Entscheidungen, die die System- und Datenarchitektur betreffen. Um ihre Menge übersichtlich zu halten, werden diese auf den nächsten Seiten in Module aufgeteilt und mit Anwendungsfalldiagrammen beschrieben. Akteure des Systems, die sich auch in den Anwendungsfällen wiederfinden, werden wie folgt definiert:

Gast: Gäste sind Benutzer, die aktuell nicht im System angemeldet sind. Sie haben lediglich Zugriff auf den Bereich zur Registrierung im System und das Zurücksetzen ihres Passworts, sollten sie bereits im System registriert sein.

Benutzer: Jeder angemeldete Akteur erhält die abstrakte Rolle Benutzer. Sie haben Zugriff auf die Verwaltung ihres Profils und ihrer Reviews. Alle Benutzer erhalten zwingend entweder die Rolle des Reviewers oder des Administrators.

Reviewer: Reviewer sind Benutzer des Systems, die keine Administratorenrechte besitzen. Sie können den kompletten Review-Prozess ausführen, sobald ihnen Reviews zugewiesen wurden.

Administrator: Administratoren können das System und dessen Benutzer verwalten. Beispielsweise erhalten sie Zugriff auf die Appsuche, das Erstellen und Verwalten von Benutzergruppen und weisen Reviewern Apps zu. Sie können ebenfalls die Reviews anderer Benutzer einsehen und exportieren, jedoch nicht bearbeiten.

3.1.1 Account

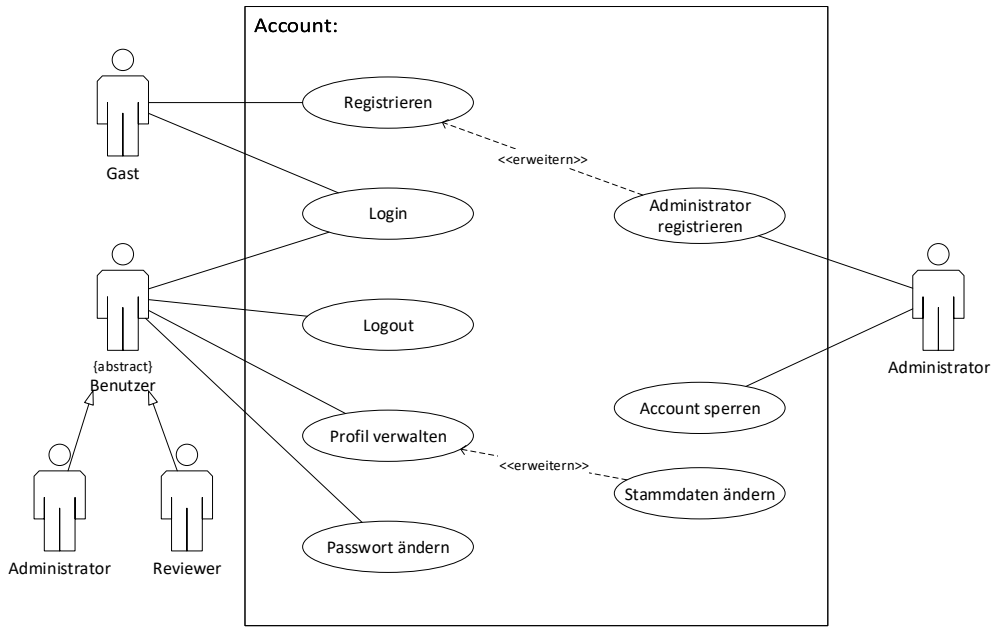


Abbildung 3.1: Anwendungsfalldiagramm für das Account-Modul

F-01 (Autorisierung): Das System begrenzt den Zugriff auf verschiedene Funktionalitäten anhand einer Zuordnung zu den Benutzergruppen Administrator und Reviewer.

F-02 (Registrierung): Gäste können sich beim System als Benutzer registrieren.

F-03 (Authentifizierung): Registrierte Benutzer können sich mit ihren Daten beim System einloggen und erhalten so Zugriff auf die für sie freigeschalteten Funktionalitäten.

F-04 (Benutzerverwaltung): Angemeldete Benutzer können in der Profilverwaltung ihre Daten ändern.

3.1.2 Benutzer

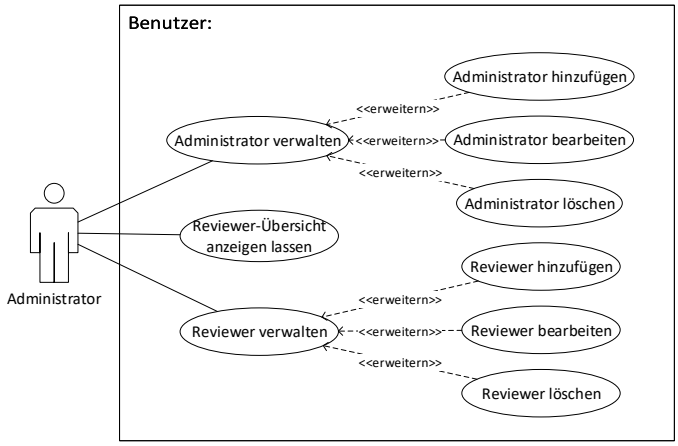


Abbildung 3.2: Anwendungsfalldiagramm für das Benutzer-Modul

F-05 (Administrator hinzufügen): Administratoren können weitere Benutzer mit Administratorrechten anlegen.

F-06 (Administrator bearbeiten): Administratoren können bereits bestehende Administratorenkonten bearbeiten, beispielsweise durch das Ändern der Email-Adresse.

F-07 (Administrator löschen): Administratoren können Benutzer mit Administratorrechten aus dem System entfernen.

F-08 (Reviewer-Übersicht anzeigen lassen): Administratoren können sich eine Übersicht von allen sich in der Datenbank befindenden Reviewern und deren Informationen anzeigen lassen.

F-09 (Reviewer hinzufügen): Administratoren können neue Reviewer der Datenbank hinzufügen.

F-10 (Reviewer bearbeiten): Administratoren können die Eigenschaften von Reviewern, die in der Datenbank vorhanden sind, ändern. Dazu gehört beispielsweise die App-Kategorie, für die ein Reviewer zuständig ist.

F-11 (Reviewer löschen): Administratoren können Reviewer aus dem System entfernen.

3.1.3 Gruppen und Kategorien

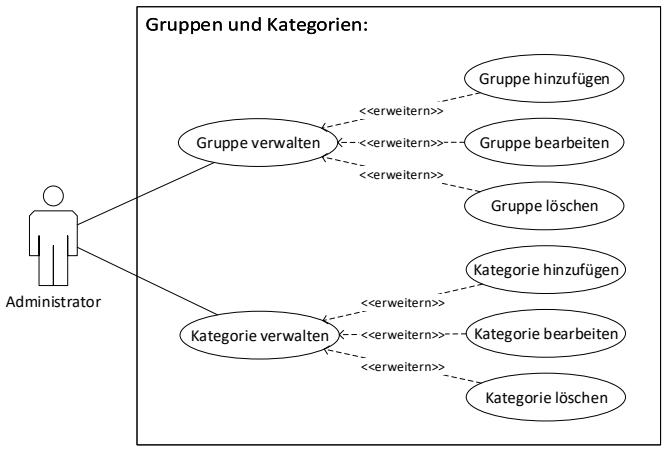


Abbildung 3.3: Anwendungsfalldiagramm für das Gruppen-Modul

F-12 (Gruppe hinzufügen): Administrator können neue Gruppen anlegen.

F-13 (Gruppe bearbeiten): Administratoren können existierende Gruppen ändern. Im Besonderen können aktuelle Gruppenmitglieder entfernt, oder neue hinzugefügt werden.

F-14 (Gruppe löschen): Administratoren können Gruppen aus dem System entfernen.

F-15 (Kategorie hinzufügen): Administrator können neue Kategorien anlegen.

F-16 (Kategorie bearbeiten): Administratoren können existierende Kategorien ändern.

F-17 (Kategorie löschen): Administratoren können Kategorien aus dem System entfernen.

3.1.4 Suche

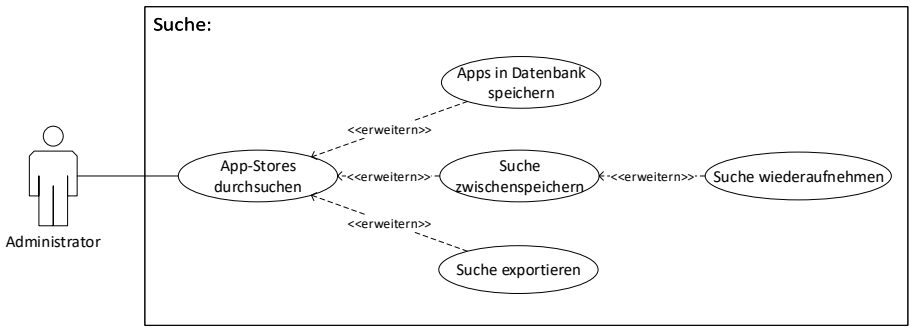


Abbildung 3.4: Anwendungsfalldiagramm für das Suche-Modul

F-18 (App-Stores durchsuchen): Administratoren können sowohl Apples App Store, als auch Googles Play Store nach geeigneten Apps durchsuchen. Die Suche kann mit Hilfe eines Suchbegriffs, der Anzahl der zu findenden Apps, dem zu durchsuchenden Store und der Sprache der App angepasst werden.

F-19 (Apps in Datenbank speichern): Die Ergebnisse der App-Suche können je nach Eignung in die Datenbank übernommen werden.

F-20 (App-Suche zwischenspeichern): Die App-Suche kann während der Auswahl der App-Übernahme jederzeit zwischengespeichert werden, um diese an einem späteren Zeitpunkt wiederaufzunehmen.

F-21 (App-Suche wiederaufnehmen): Eine zuvor zwischengespeicherte Suche kann jederzeit wiederaufgenommen und abgeschlossen, oder abermals zwischengespeichert werden.

F-22 (App-Suche exportieren): Eine durchgeführte App-Suche kann mit ihren Informationen (Zeitpunkt der Suche, Suchergebnisse, Übernahme/Nicht-Übernahme einzelner Apps etc.) als CSV-Datei exportiert werden.

3.1.5 App

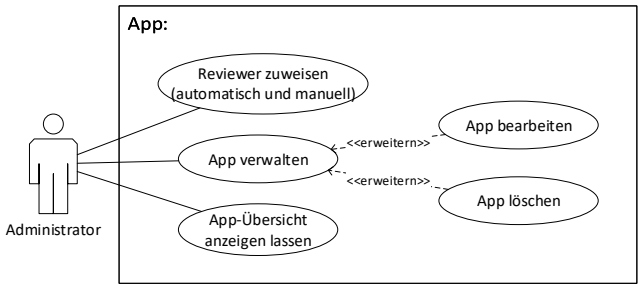


Abbildung 3.5: Anwendungsfalldiagramm für das App-Modul

F-23 (App-Übersicht anzeigen lassen): Eine aktuelle Übersicht der Apps soll angezeigt werden können. Dabei sollen Informationen über die App selbst, aber auch zugewiesene Reviewer oder Bewertungen eingesehen werden können.

F-24 (App bearbeiten): Administratoren können die Informationen von Apps, die sich in der Datenbank befinden, ändern.

F-25 (App löschen): Administratoren können Apps aus der Datenbank entfernen.

F-26 (Reviewer zuweisen (automatisch und manuell)): Administratoren können den Apps in der Datenbank Reviewer zuweisen. Dies kann entweder automatisiert (anhand einer Kategorie oder Gruppenzugehörigkeit) oder manuell geschehen.

3.1.6 Review

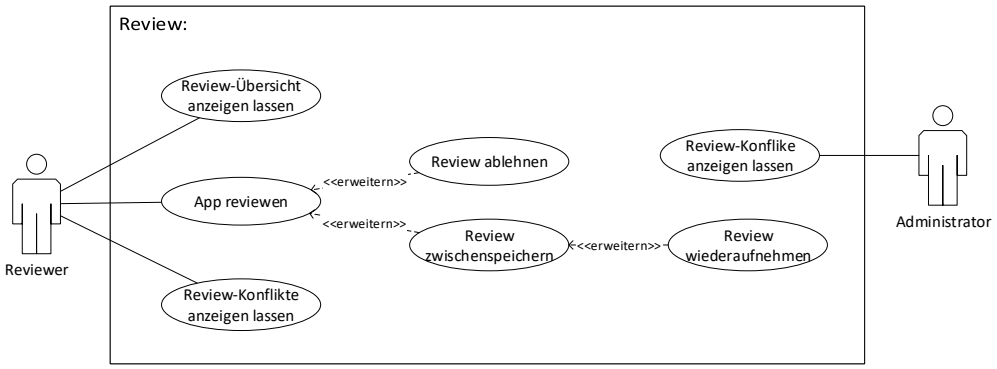


Abbildung 3.6: Anwendungsfalldiagramm für das Review-Modul

F-27 (Review-Konflikte anzeigen lassen): Sollten die Bewertungen der beiden Reviewer für einen oder mehrere Aspekte einer App stark auseinanderliegen - beispielsweise bei der Beurteilung der Relevanz für die Datenbank - soll dies den Administratoren in einer Übersicht angezeigt werden, von der aus sie entsprechende Maßnahmen der Konfliktbeseitigung ergreifen können.

F-28 (Review-Übersicht anzeigen lassen): Die aktuell ausstehenden Reviews werden für den zugeordneten Reviewer in einer Übersicht angezeigt.

F-29 (App reviewen): Zugeteilte Apps können von ihren Reviewern anhand der MARS (siehe Kapitel 2.1) bewertet werden.

F-30 (Review ablehnen): Reviewer können das bewerten einer App ablehnen, sollten sie der Meinung sein, dass sie nicht für die Datenbank relevant ist.

F-31 (Review zwischenspeichern): Die Bewertung einer App kann jederzeit zwischengespeichert werden, um diese an einem späteren Zeitpunkt wiederaufzunehmen.

F-32 (Review wiederaufnehmen): Eine zwischengespeicherte Bewertung kann jederzeit wiederaufgenommen und abgeschlossen, oder abermals zwischengespeichert werden.

3.2 Nicht-funktionale Anforderungen

Im Folgenden sind die nicht-funktionalen Anforderungen aufgelistet. Sie definieren die Qualität, in welcher die geforderten Funktionalitäten zu erbringen sind.

NF-01 (Zuverlässigkeit): Das System sollte zuverlässig, also fehlerfrei für seine Benutzer zu erreichen sein.

NF-02 (Robustheit): Das System soll fehlerhafte der Benutzereingaben abfangen und verarbeiten können. Dies bedeutet insbesondere, dass es für diese Fälle vorher definierte Reaktionen des Systems gibt.

NF-03 (Sicherheit/Datenschutz): Benutzerdaten sollen sicher im System abgelegt werden können und nur von den berechtigten Akteuren einzusehen oder zu verändern sein.

NF-04 (Effizienz): Das System soll effizient und performant arbeiten. Dies betrifft vor allem das Aufrufen und Senden von Daten durch den Benutzer, da so eine flüssige Arbeitsweise gewährleistet werden kann.

NF-05 (Wartbarkeit): Das System sollte ausreichend dokumentiert sein, um eine Erweiterung zu einem späteren Zeitpunkt nicht unangemessen zu erschweren.

NF-06 (Erweiterbarkeit): Eine unkomplizierte Erweiterung der Systemfunktionalitäten sollte gewährleistet sein. Insbesondere sollen weitere Fragebögen zur Grundlage der App-Reviews unkompliziert eingebaut werden können.

4 Architektur

Aufbauend auf den Systemanforderungen aus Kapitel 3 wird in diesem Kapitel die Struktur der Systemarchitektur erörtert. Kapitel 4.1 skizziert dabei einen Überblick auf das Gesamtsystems und dessen Komponenten. Anschließend wird in Kapitel 4.2 der Ablauf eines typischen Anwendungsfalls veranschaulicht, bevor Kapitel 4.3 die Datenstruktur aufzeigt.

4.1 Gesamtsystem

Das Gesamtsystem setzt sich aus vier zusammenhängenden und miteinander kommunizierenden Komponenten zusammen: Der Web-Anwendung (siehe Abbildung 4.1①, bzw. Kapitel 4.1.1), der System-Serveranwendung (siehe Abbildung 4.1②, bzw. Kapitel 4.1.2), der Scraper-Serveranwendung (siehe Abbildung 4.1③, bzw. Kapitel 4.1.3), sowie der relationalen Datenbank (siehe Abbildung 4.1④, bzw. Kapitel 4.1.4). Diese Komponenten übernehmen verschiedene Aufgaben innerhalb der klassischen Schichtenarchitektur. Die Interaktion der Benutzer mit dem System erfolgt über die Präsentationsschicht, die die notwendigen Benutzerschnittstellen implementiert. In der Logikschicht wird das Scrapen der App-Stores und die Interaktion der Datenmodelle bereitgestellt. Die Datenzugriffsschicht bildet diese Datenmodelle auf die Tabellen in der Datenbank ab. Ein object-relational mapping (ORM)-Framework schreibt diese persistent in die relationale Datenbank, die sich in der Datenhaltungsschicht befindet. Abbildung 4.1 veranschaulicht die Interaktion der Softwareschichten und deren Komponenten.

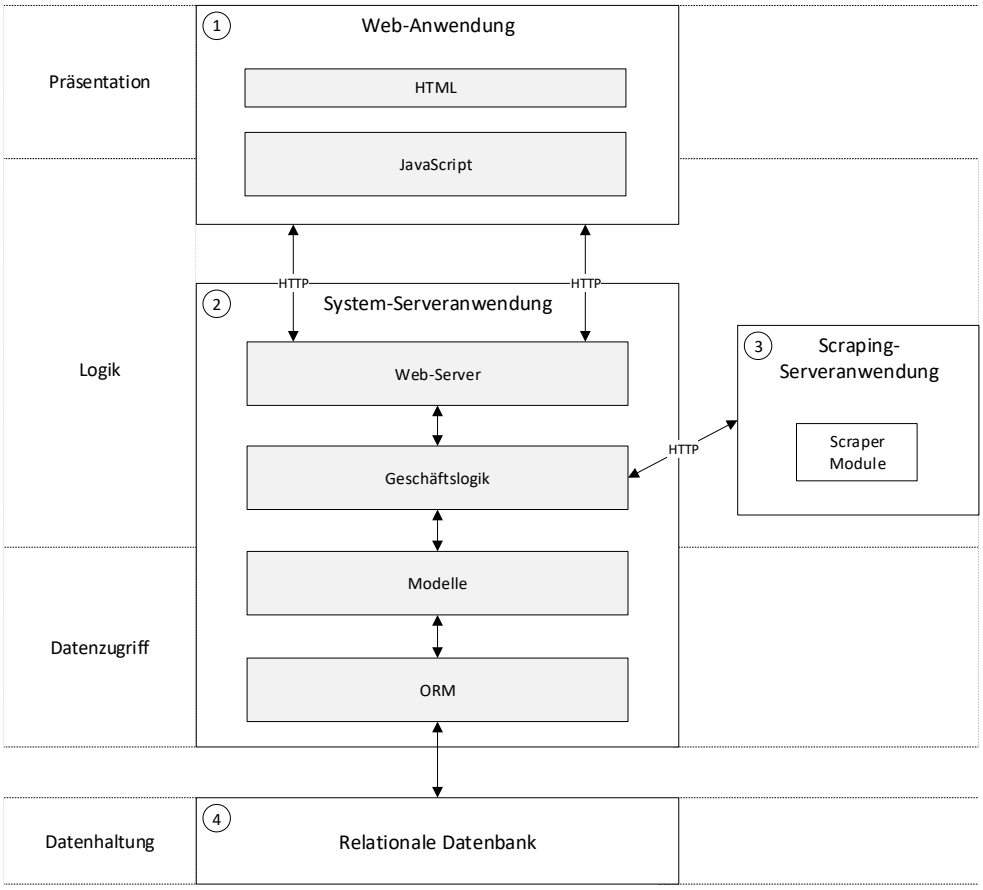


Abbildung 4.1: Architektur des Gesamtsystems

4.1.1 Web-Anwendung

Mit Hilfe der Web-Anwendung haben sowohl Administratoren als auch Reviewer Zugriff auf das System. Administratoren verwalten so Benutzer, suchen nach Apps oder verteilen Reviews auf die Reviewer. Das Bewerten der Apps durch die Reviewer wird ebenfalls in der Anwendung, allerdings in einem eigenen Bereich, durchgeführt. Die Anwendung wurde mit HTML [29] und JavaScript [16] entwickelt und ist zum größten Teil zur Informationsanzeige für die Benutzer angelegt. Die dafür benötigten Informationen erhält diese vom Web-Server. Um die hoch priorisierte nicht-funktionale Anforderung **Erweiterbarkeit** (Kapitel 3.2) zu erfüllen, wird die Bereitstellung und Bearbeitung der wissenschaftlichen Fragebögen über die

SurveyJS-Bibliothek [4] realisiert. Dies ermöglicht eine schnelle Erstellung der Fragebögen über das SurveyJS Creator Tool [3] und unkomplizierte Einbindung in die Anwendung. Da diese so auch Teile der Geschäftslogik beinhaltet, erstreckt sich die Web-Anwendung bis in die Logikschicht.

4.1.2 System-Serveranwendung

Die System-Serveranwendung stellt den Großteil des Systems dar und wurde in PHP [25] entwickelt. Über einen Web-Server wird mit der Web-Anwendung kommuniziert. Der Web-Server erhält die Datenmodelle zur Bereitstellung der Informationen über die Geschäftslogik. Eingehende Anfragen werden nach einer Autorisierungsprüfung vom Web-Server weitergeleitet und innerhalb der Geschäftslogik auf Korrektheit und Vollständigkeit überprüft. Ist diese Prüfung erfolgreich, werden die Datenänderungen an die Datenzugriffsschicht weitergegeben, wo sie vom vom ORM-Framework persistent in die relationale Datenbank geschrieben werden.

4.1.3 Scraping-Serveranwendung

Das Scrapen der App-Stores wird durch eine separate Serveranwendung realisiert. Die hoch priorisierten nicht-funktionalen Anforderungen **Robustheit** und **Erweiterbarkeit** (Kapitel 3.2) erfordern diese Trennung der beiden Anwendungen. Das System muss auf die unvermeidbaren Änderungen der Hersteller in der Struktur ihrer App-Stores reagieren können. Durch die Auslagerung der Scraper-Module kann schnell auf diese Änderungen reagiert und die Funktionalität wiederhergestellt werden. Die dafür zuständigen Node.js [17]-Module stammen aus den GitHub [7] Projekten *google-play-scraper* [10] und *app-store-scraper* [8]. Auf dem Server läuft eine RESTful API [9] desselben Erstellers, die leicht modifiziert wurde um beide App-Store Scraper über die selbe API zugänglich zu machen. Diese API durchsucht auf Anfrage der System-Serveranwendung die entsprechenden App-Stores und gibt die Ergebnisse dieser Suche zurück.

4.1.4 Relationale Datenbank

In der Datenhaltungsschicht ist eine relationale Datenbank für die persistente Speicherung der Systemdaten verantwortlich. Die Daten werden mithilfe eines ORM-Frameworks für die Anwendung zugänglich gemacht. Die Datenbank ist lose an die System-Serveranwendung gekoppelt, was zu einem verbesserten Ressourcenverbrauch führt und die nicht-funktionale Anforderung **Effizienz** (Kapitel 3.2) erfüllt.

4.2 Ablauf

An dieser Stelle soll das Systemverhalten während eines kompletten Interaktionszyklus verdeutlicht werden. Ein Zyklus besteht aus dem Durchsuchen der App-Stores durch einen Administrator, dem Abspeichern relevanter Apps in der System-Datenbank, dem Verteilen der Apps auf zwei Experten, dem Bewerten der App durch diese Experten sowie das Exportieren des Reviews durch einen Administrator.

Abbildung 4.2 zeigt den Startbildschirm (Dashboard) eines einsatzbereiten Systems.

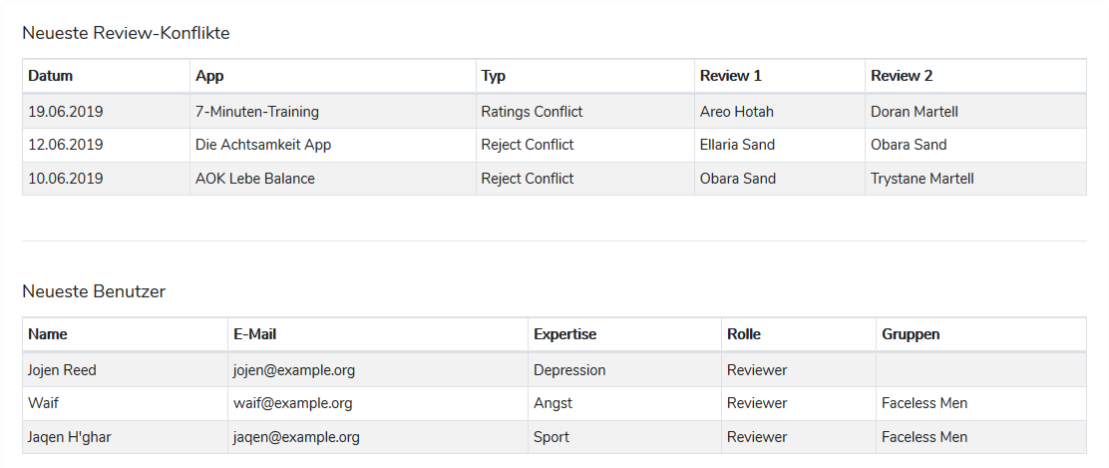


Abbildung 4.2: Das Administrator-Dashboard

Das Administrator-Dashboard zeigt eine Übersicht der zuletzt zum System hinzugefügten Benutzer. Außerdem werden entdeckte Review-Konflikte (siehe Kapitel 3.1.6) aufgelistet. Existieren im System genügend Experten, können Gruppen und Kategorien (bzw. Expertisen) [1] erstellt und die Benutzer auf diese verteilt werden.
Um mit den Experten das Reviewverfahren durchführen zu können, müssen zuerst Apps im jeweiligen App-Store gefunden und in die System-Datenbank übernommen werden. Die dafür benötigten Schritte werden im folgenden skizziert. Eine neue Suche beginnt damit, dass der Administrator die Suchdaten (Suchbegriff, Anzahl der Apps, App-Store(s), Land, Sprache) in die Suchmaske eingibt und die Suche startet. Werden von den Scrapern valide Ergebnisse zurückgegeben, werden diese dem Benutzer im nächsten Fenster präsentiert. In Abbildung 4.3 ist ein solches Suchergebnis abgebildet. Neben dem Titel und Entwickler der einzelnen

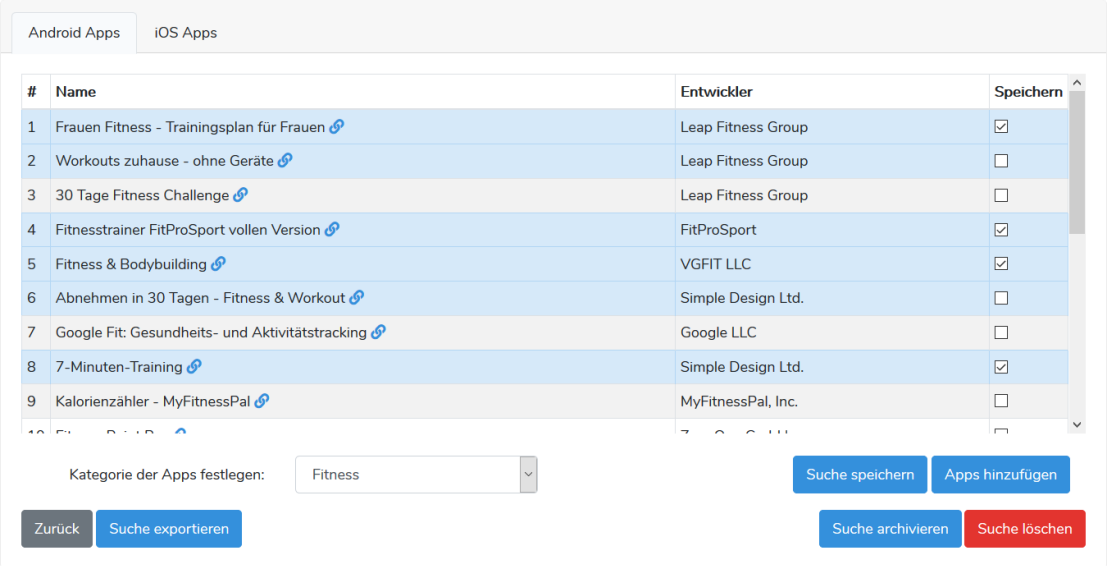


Abbildung 4.3: Mögliches Suchergebnis für den Begriff *Fitness*

Apps bekommt der Benutzer durch eine farbliche Markierung angezeigt, ob diese App ein Duplikat einer bereits in der Datenbank existierenden App ist, oder nicht. Sollte bereits einer App mit derselben App-Id und Version gespeichert sein, so wird das Suchergebnis blau hinterlegt. Außerdem kann der Benutzer sich direkt über

[1] Apps und Benutzer können den Kategorien im System zugeordnet werden. Bei den Experten stellt dies ihre Expertise dar.

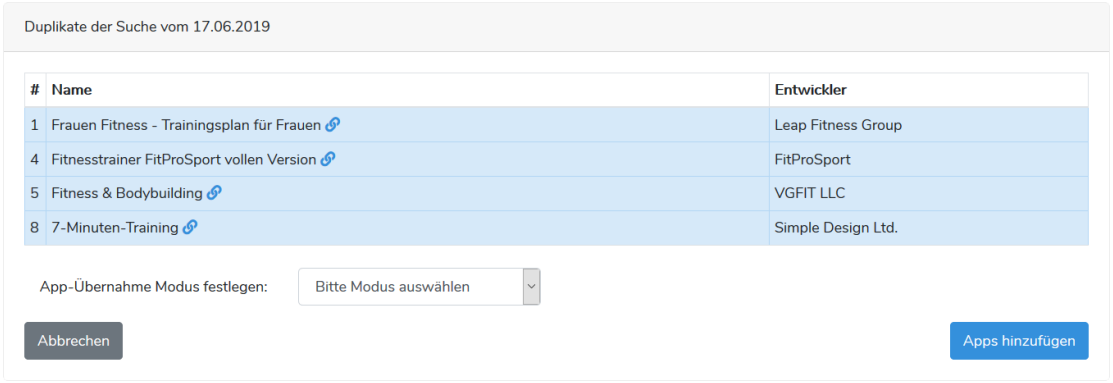


Abbildung 4.4: Duplikatsbehandlung einer Suche

einen Klick auf das Suchergebnis Kurzinformationen über die App einblenden lassen, oder die Store-Seite über einen Klick auf das Link-Symbol besuchen. In dieser Liste der Suchergebnisse markiert der Administrator nun jede App, die ins System übernommen werden soll. Er kann diese Auswahl auch jederzeit mit einem Klick auf *Suche speichern* beenden und an einem anderen Zeitpunkt mit dem aktuellen Stand fortfahren. Ist die Auswahl abgeschlossen, kann optional eine Kategorie für alle zu übernehmenden Apps ausgewählt werden. Mit einem Klick auf *Apps hinzufügen* werden alle ausgewählten Apps in die Datenbank geschrieben. Sollten sich unter den markierten Apps Duplikate befinden, muss auf einer nächsten Seite entschieden werden, wie die betreffenden Apps in die Datenbank übernommen werden. App und Suchergebnis können entweder verknüpft, oder die App dupliziert werden. Die erste Möglichkeit bedeutet, dass die App lediglich einen Verweis auf die neue Suche erhält und keine neue App gespeichert wird. Eine solche Behandlung dieses Sonderfalles wird in Abbildung 4.4 gezeigt.

Befinden sich die Apps in der System-Datenbank, können sie den Experten zugewiesen werden. Dabei kann der Administrator aus drei Verfahren wählen:

Manuell: Für jede App werden vom Administrator händisch ein bis zwei Experten aus einer Liste ausgewählt und der App zugewiesen.

Automatisch anhand der Benutzerexpertise: Dabei wählt der Administrator zuerst die Menge an Experten aus und selektiert in einem zweiten Schritt die zu verteilenden Apps. Ein Algorithmus verteilt anschließend automatisch die Apps auf

die Benutzer. Dabei werden deren Expertisen und die Anzahl an bisher erhaltenen Reviews berücksichtigt.

Automatisch anhand der Gruppenangehörigkeit: Hierbei wählt der Benutzer zuerst die zu verteilenden Apps aus der Liste aus, wählt dann die Gruppe, auf die die Apps verteilt werden und zuletzt die Menge an Apps, die jedes Gruppenmitglied zugeteilt bekommen soll. Klickt er nun auf *Apps verteilen* werden die Apps zufällig den Gruppenmitgliedern zugewiesen. Sollte der Benutzer zu wenig Apps ausgewählt haben, bricht das System den Vorgang ab und weist den Benutzer mit einer Fehlermeldung darauf hin. Sind zu viele Apps ausgewählt, verteilt das System so lange, bis alle Mitglieder die vorgesehene Anzahl an Reviews zu erledigen haben und verwirft die restlichen Apps. Abbildung 4.5 zeigt die Verteilung von Apps auf eine Gruppe.

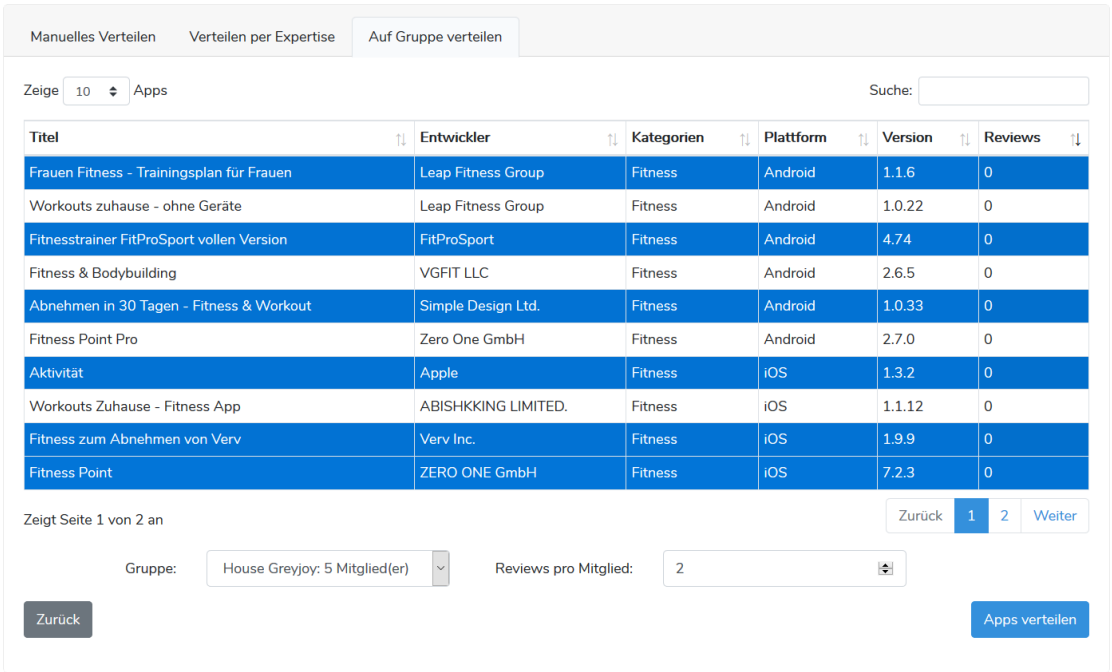


Abbildung 4.5: Verteilen von Apps auf eine Gruppe

Mit dem Verteilen der Apps werden automatisch leere Reviews für jede App und jeden Benutzer erstellt. Die betreffenden Experten werden zusätzlich mit einer E-Mail auf den Erhalt der App hingewiesen.

Ist ein Reviewer für das Bewerten einer App zugewiesen worden, sieht er das neue Review direkt auf der Startseite seines Systemabschnittes, dem Reviewer-Dashboard. Dabei wird zwischen komplett neu erhaltenen Reviews und bereits bearbeiteten und zwischengespeicherten Reviews unterschieden (siehe Abbildung 4.6).

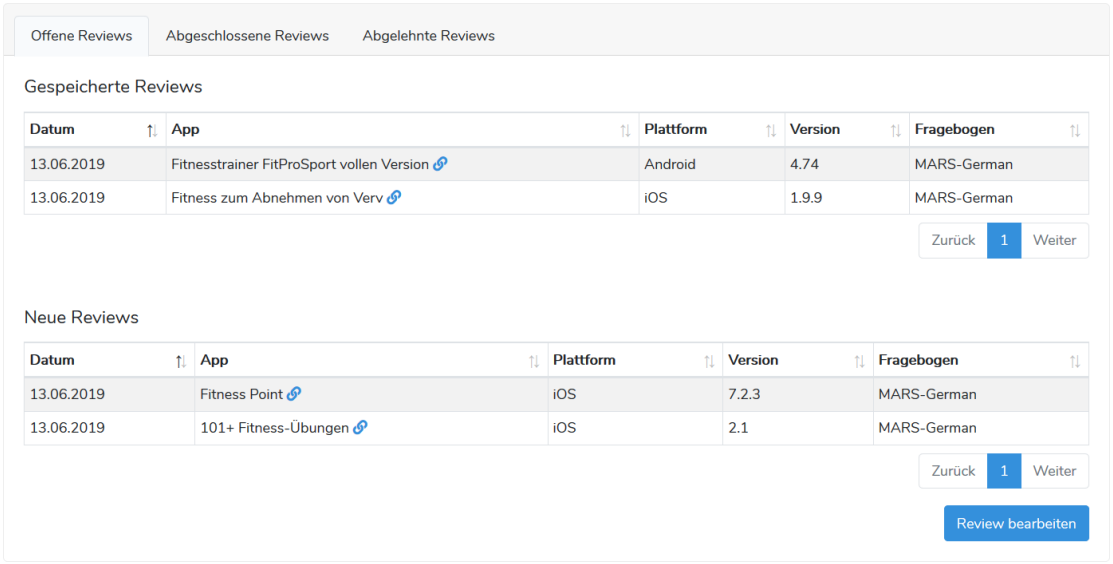


Abbildung 4.6: Das Reviewer-Dashboard

Als Bewertungsskala ist im System die MARS-G implementiert (siehe Kapitel 5.2.3). Während dieses Bewertungsprozesses können die Experten jederzeit das Review zwischenspeichern und zu einem anderen Zeitpunkt fortsetzen. Sollten sie der Meinung sein, dass diese App keine Bewertung bedarf, beispielsweise wegen fehlender Relevanz oder medizinischer Mängel, können sie das Rating unter Angabe einer Begründung ablehnen.

Wird ein Review von einem Experten abgeschlossen, errechnet das System die Mittelwerte der einzelnen Sektionen, die für die Übersicht der App-Güte im Rahmen des MHAD-Projekts (Kapitel 2.2) benötigt werden. Dieser Prozess wird in Kapitel 5.2.3 genauer erläutert. Das abgeschlossene Review kann im csv-Dateiformat exportiert und mit Tabellenkalkulationsprogrammen wie Microsoft Excel [15] eingebunden und verarbeitet werden. Dieser Export kann für jeden Fragebogen individualisiert werden. Abbildung 4.7 zeigt den Export eines abgeschlossenen Reviews.

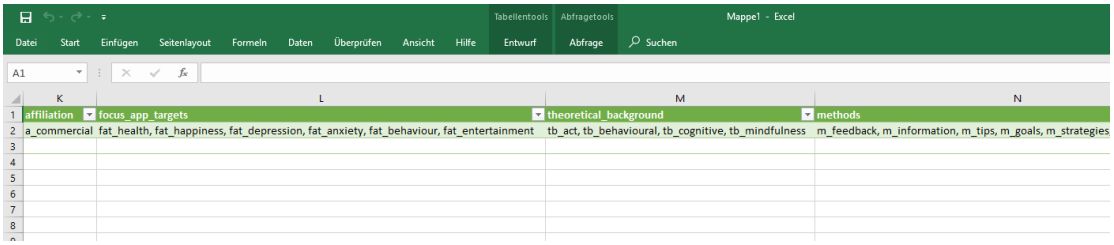


Abbildung 4.7: Exportiertes Review, eingebunden in Microsoft Excel

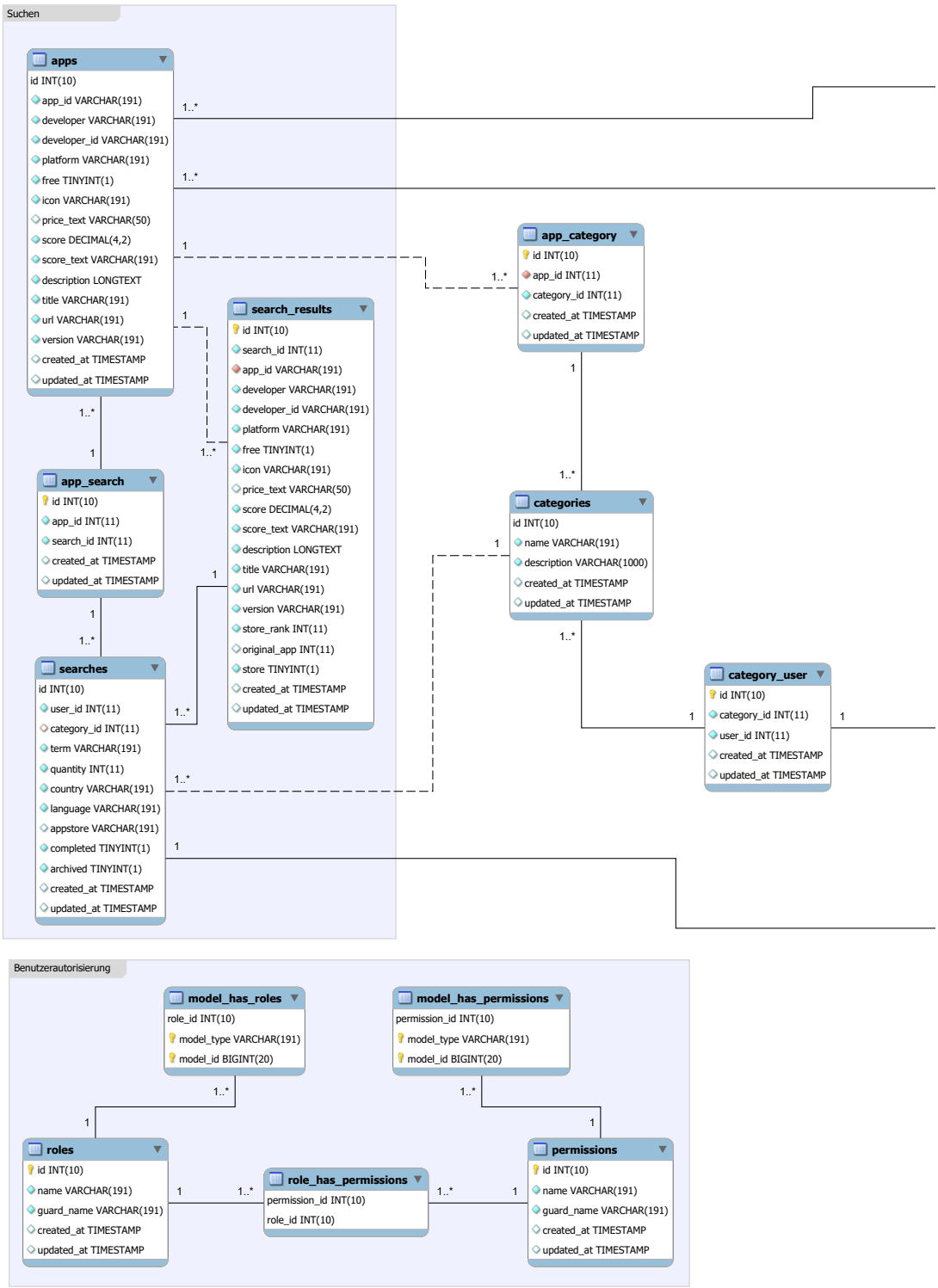
4.3 Datenstruktur

Abbildung 4.8 zeigt das Datenbankschema der relationalen Datenbank des System. Die Tabellen *model_has_roles*, *roles*, *role_has_permissions*, *model_has_permissions* und *permissions* sind Teil der Rechteverwaltung innerhalb des Systems und stammen aus dem Laravel-Package *laravel-permission* [19]. Die Rechteverwaltung basiert auf der Definition von Benutzerrollen und deren Berechtigungen. Da für die einfache Trennung zwischen Administratoren und Reviewern im System keine feingranularen Rechteabgrenzungen benötigt wurden, sind nur Rollen und deren dazugehörige Benutzer-Modelle in den Tabellen gespeichert. Feinere Rechteunterscheidungen können allerdings durch die *permissions* Tabellen gemäß der nichtfunktionalen Anforderung **Erweiterbarkeit** (Kapitel 3.2) nachträglich implementiert werden.
Datenbankmigrationen werden in der Tabelle *migrations* festgehalten. Die Tabelle *jobs*, respektive *failed_jobs* sind Teil der Implementierung von Mail-Queues innerhalb des Systems. Um eine reibungslose Interaktion mit der Anwendung zu ermöglichen und unerwünschte Wartezeiten während des E-Mail-Versands zu verhindern, wurde auf ein Warteschlangensystem zurückgegriffen. Das System legt bei Bedarf Jobs der Art „Mailversand“ in die *jobs* Tabelle und arbeitet diese nach einer kurzen Wartezeit ab. Fehlgeschlagene Jobs sind in der *failed_jobs*-Tabelle gespeichert.
Systembenutzer sollen wie in Kapitel 3 definiert auf Gruppen aufgeteilt werden und Expertisen besitzen. Die Tabellen *groups* und *categories* repräsentieren diese Gruppen und Expertisen. Die zweite Tabelle trägt nicht den Namen *expertises*, da die App-Kategorien den Benutzer-Expertisen entsprechen und darum ein einheitlicher Name gewählt wurde. Die Tabelle *group_user* hält die many-to-many-Beziehungen zwischen den Gruppen- und Benutzer-Modellen und ist der Konven-

tion des Laravel-Frameworks [2] entsprechend benannt. Ebenso verhält es sich mit den Tabellen *category_user*, *app_category* und *app_search*, die die many-to-many-Beziehung ihrer jeweiligen Modelle beinhalten.
Die *users*-Tabelle enthält alle Stammdaten der im System registrierten Benutzer wie beispielsweise ihre E-Mail-Adresse, das Passwort, ihre Spracheinstellung oder Benutzerbild. Tokens für eine mögliche Zurücksetzung des Passworts durch die Benutzer sind in der Tabelle *password_resets* gespeichert.
Die für das System integralen Suchen der App-Stores werden in *searches* repräsentiert. Diese Tabelle trägt Informationen wie die Suchdetails (Begriff, Menge, Store, Land, Sprache, Erfüllungsgrad), welcher Kategorie die gefundenen Apps zugewiesen wurden und wer die Suche durchführte. Die Suchergebnisse, also die Rückgaben der Scraper, werden in einer separaten *search_results*-Tabelle gespeichert. Diese ist nötig, da nicht jedes Suchergebnis in die *apps*-Tabelle übertragen wird, aber ebenso nicht verworfen werden darf, da sie für Analysezwecke benötigt und exportiert werden. Die Duplikatsüberprüfung der Suchergebnisse macht einen Verweis auf die *apps*-Tabelle nötig, die die tatsächlich ins System übernommenen Apps speichert. Die *apps*-Tabelle beinhaltet alle Informationen der gespeicherten Apps. Insbesondere werden die Versionsnummer und die einmalige App-Id für die Duplikatserkennung benötigt. Da Apps in mehreren Suchen gefunden werden können, hält die *app_search*-Tabelle die many-to-many-Beziehungen von Suchen und Apps.
Sind Apps ins System übertragen worden, sollen dazu Reviews erstellt werden. Diese werden durch die *reviews*-Tabelle repräsentiert. Ein Review gehört immer zu genau einem Benutzer und einer App und hat einen Fragebogen als Grundlage. Neben Zusatzinformationen wie der aktuellen Seite oder dem Fortschritt des Bearbeitens hält die Tabelle auch den Inhalt der ausgefüllten Antworten des Reviews im *content*-Feld. Obwohl im System zum Zeitpunkt dieser Arbeit nur ein Fragebogen existiert, ist das Hinzufügen weiterer Fragebögen gemäß der nichtfunktionalen Anforderung **Erweiterbarkeit** (Kapitel 3.2) möglich. Diese werden in der *surveys*-Tabelle gespeichert. Das *template*-Feld hält dabei den Fragebogen im JSON-Dateiformat, der durch den Inhalt der *reviews*-Tabelle gefüllt wird. Ebenso sollen Mittelwerte der MARS-Abschnitte berechnet (siehe Kapitel 2.2) und das Ex-

[2] Tabellen, die many-to-many-Beziehungen beinhalten, sollten in Laravels ORM *Eloquent* [22] nach den beiden Modellen - verbunden mit einem Unterstrich - benannt werden. Hierbei wird der alphabetisch zuerst auftretende Modellname links geschrieben.

portieren personalisiert werden. Die dafür benötigten JSON-Zeichenketten sind in den Feldern *average_sections* beziehungsweise *export_keys* gespeichert. Durch die Beziehung zwischen Review und Fragebogen kann es einfach Reviews auf Grundlage einer anderen Skala geben. Außerdem ist die Konflikterkennung nach Kapitel 3 Teil der Systemanforderungen. Wird ein Konflikt erkannt, wird er in der *conflicts*-Tabelle gespeichert, die Verweise zur entsprechenden App und den beiden im Konflikt stehenden Reviews enthält.


Suchen
apps
id INT(10)
app_id VARCHAR(191)
developer VARCHAR(191)
developer_id VARCHAR(191)
platform VARCHAR(191)
free TINYINT(1)
icon VARCHAR(191)
price_text VARCHAR(50)
score DECIMAL(4,2)
score_text VARCHAR(191)
description LONGTEXT
title VARCHAR(191)
url VARCHAR(191)
version VARCHAR(191)
created_at TIMESTAMP
updated_at TIMESTAMP
app_category
id INT(10)
app_id INT(11)
category_id INT(11)
created_at TIMESTAMP
updated_at TIMESTAMP
search_results
id INT(10)
search_id INT(11)
app_id VARCHAR(191)
developer VARCHAR(191)
developer_id VARCHAR(191)
platform VARCHAR(191)
free TINYINT(1)
icon VARCHAR(191)
price_text VARCHAR(50)
score DECIMAL(4,2)
score_text VARCHAR(191)
description LONGTEXT
title VARCHAR(191)
url VARCHAR(191)
version VARCHAR(191)
store_rank INT(11)
original_app INT(11)
store TINYINT(1)
created_at TIMESTAMP
updated_at TIMESTAMP
app_search
id INT(10)
app_id INT(11)
search_id INT(11)
created_at TIMESTAMP
updated_at TIMESTAMP
categories
id INT(10)
name VARCHAR(191)
description VARCHAR(1000)
created_at TIMESTAMP
updated_at TIMESTAMP
searches
id INT(10)
user_id INT(11)
category_id INT(11)
term VARCHAR(191)
quantity INT(11)
country VARCHAR(191)
language VARCHAR(191)
appstore VARCHAR(191)
completed TINYINT(1)
archived TINYINT(1)
created_at TIMESTAMP
updated_at TIMESTAMP
category_user
id INT(10)
category_id INT(11)
user_id INT(11)
created_at TIMESTAMP
updated_at TIMESTAMP
Benutzerautorisierung
model_has_roles
role_id INT(10)
model_type VARCHAR(191)
model_id BIGINT(20)
model_has_permissions
permission_id INT(10)
model_type VARCHAR(191)
model_id BIGINT(20)
roles
id INT(10)
name VARCHAR(191)
guard_name VARCHAR(191)
created_at TIMESTAMP
updated_at TIMESTAMP
role_has_permissions
permission_id INT(10)
role_id INT(10)
permissions
id INT(10)
name VARCHAR(191)
guard_name VARCHAR(191)
created_at TIMESTAMP
updated_at TIMESTAMP

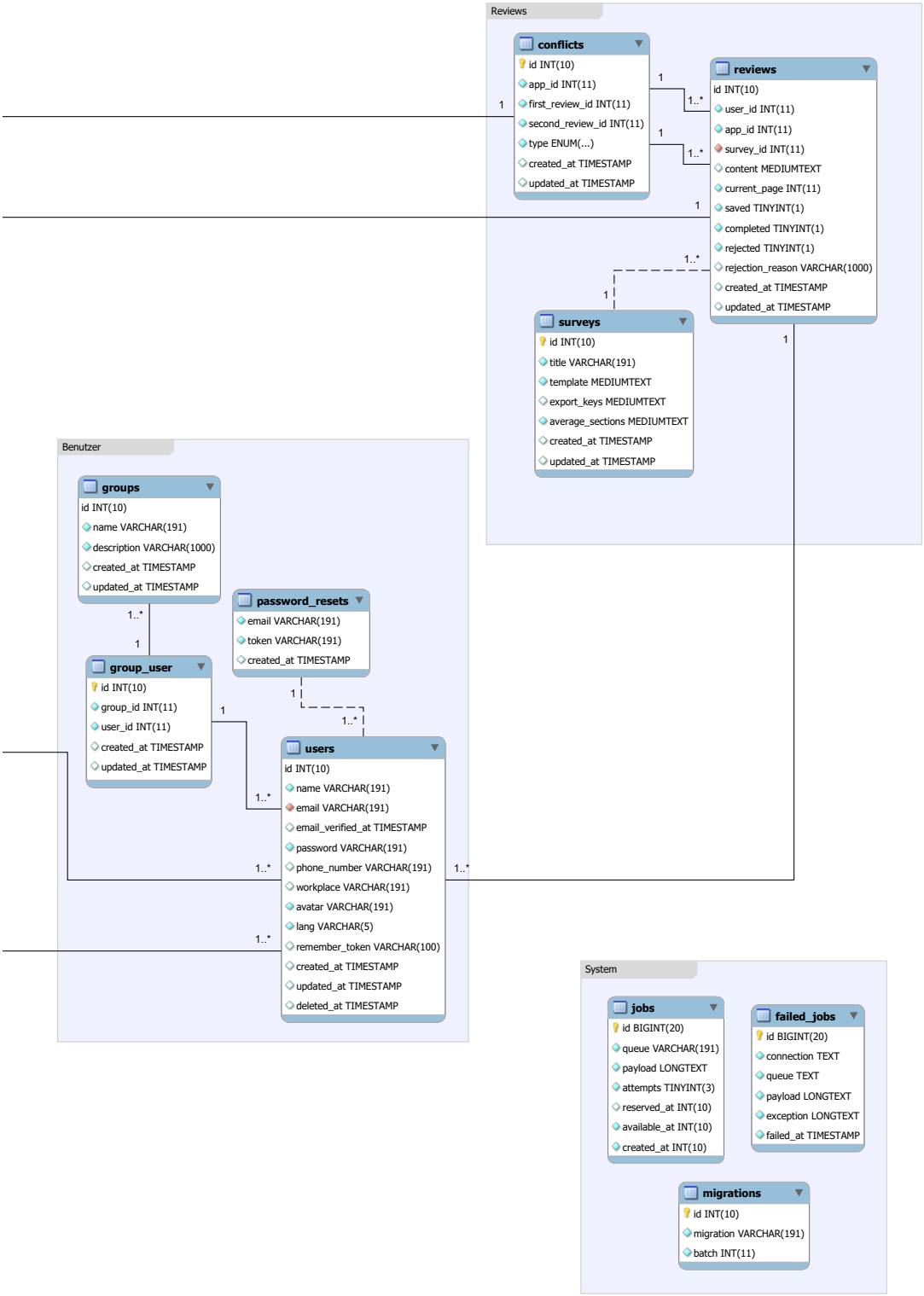


Abbildung 4.8: Das Schema der Systemdatenbank

5 Ausgewählte Aspekte der Implementierung

In diesem Kapitel werden einzelne Implementierungsaspekte vorgestellt. Kapitel 5.1 listet die verwendeten Technologien auf, bevor in Kapitel 5.2 drei ausgewählte Anwendungsfälle und deren Umsetzung im System erläutert werden.

5.1 Technologien

Es folgt eine Auflistung von Technologien die während des Implementierungsprozesses benutzt wurden.

PHP und Laravel:
PHP [25] ist eine etablierte Programmiersprache, die sich mit ihren zahlreichen Funktionsbibliotheken ausgezeichnet zur Entwicklung von Webanwendungen eignet. Unterstützend wurde das PHP-Framework *Laravel* [23] für die Implementierung des Systems verwendet. Dies war für die Entwicklung zwar nicht vorgeschrieben, allerdings empfohlen, da bereits das MHAD-Projekt (siehe Kapitel 2.2) mit den selben Technologien entwickelt wurde. Laravel bietet zahlreiche Features, die ein schnelles Entwickeln unterstützen. Beispielsweise werden direkt Authentifizierungs-, Autorisierungs- und Lokalisierungseigenschaften geboten. Diese müssen nur ins System integriert und nicht selbst entwickelt werden. Außerdem besitzt Laravel mit den Laracast-Screencasts [12] eine umfangreiche und kostenlose Möglichkeit, die Grundlagen des Frameworks zu lernen.

Atom:
Das System wurde mithilfe des Atom [6] Texteditor entwickelt. Dank der Möglichkeit,

eine Vielzahl optionaler Packages zu installieren, kann der Editor auf die persönlichen Wünsche und Bedürfnisse angepasst werden. Die unkomplizierte Möglichkeit, Atom mit GitHub [7] zu verbinden und so die Versionskontrolle zu vereinfachen, war außerdem ein Argument für die Benutzung von Atom.

XAMPP:
Die Apache-Distribution XAMPP [1] wurde als PHP-Entwicklungsumgebung gewählt. XAMPP stellt nach der Installation sowohl PHP, als auch MariaDB zur Verfügung und erweist sich damit als hervorragende Alternative.

MariaDB:
Bei der Wahl der relationalen Datenbank wurde auf das relationale Datenbankmanagementsystem MariaDB [24] zurückgegriffen. Die vorhandenen Vorkenntnisse in relationalen Datenbanken und einfache Anbindung an Laravel beeinflussten die Wahl zugunsten von MariaDB.

Bootstrap:
Das Frontend-CSS-Framework Bootstrap [27] wird mit Laravel ausgeliefert und wurde als Styling-Framework bei der Implementierung des zu entwickelnden Systems eingesetzt.

DataTables:
Die Benutzung von DataTables [20] als Plug-in für die jQuery [26] Javascript Bibliothek wurde gewählt, da so unkompliziert die zahlreichen HTML-Tabellen im System mit nützlichen Funktionen erweitert werden konnten. Um den Komfort der Bedienung des Systems zu erhöhen, wurden mithilfe von DataTables Sortier-, Selektions- und Suchfunktionen eingefügt. Das Bootstrap Styling-Framework wird zusätzlich von DataTables unterstützt.

SurveyJS: Die Fragebögen wurden mit SurveyJS [4] modelliert. Da die Anforderungen einen flexiblen und unkomplizierten Austausch der Fragebögen voraussetzten, wurde das Einbinden über SurveyJS einer Implementierung in reinem HTML vorgezogen. Auf der SurveyJS Homepage können mithilfe des Creator Tools komfortabel Fragebögen erstellt werden. Diese werden im JSON-Dateiformat in der Datenbank gespeichert und bei Bedarf aus dieser auf die Seite geladen. So lässt sich durch das einfache Austauschen der JSON-Zeichekette das flexible Ändern oder Wechseln des Fragebogens realisieren.

Google Play und Appstore Scraper:
Das Laden der Apps erfolgt durch das Durchsuchen der jeweiligen App-Stores mithilfe eines Scrapers. Diese Scraper sind durch offene Node.js Module [10, 8], aus den GitHub-Projekten *google-play-scraper* [10] und *app-store-scraper* [8], umgesetzt. Um diese Module in der PHP-Anwendung zu benutzen, wurde außerdem auf eine API [9] desselben Erstellers zugegriffen. Diese API stand jedoch nur für den Google Play-Store zur Verfügung, weshalb es einer Modifizierung bedurfte. Nach dieser Anpassung sind nun sowohl die Ergebnisse der Suche im Google Play-Store, als auch in Apples App-Store über die API zugänglich.

Zeit Now:
Zeit Now [35] ermöglicht ein sofortiges Hosten von Webseiten und Services und benötigt minimale Konfiguration. Now stellte sich als hervorragende Möglichkeit heraus, den Code der oben beschriebenen API einzusetzen. Nach dem Veröffentlichen der API mithilfe von Zeit Now erhält man eine einzigartige URL, über die sie vom System benutzt werden konnte. Die Plattform unterstützt sogar das Verknüpfen des Entwicklungs-Repositorys und aktualisiert eventuelle Veränderungen eigenständig. Diese Eigenschaften machten es zu einer idealen Alternative im Entwicklungsprozess.

Mailtrap:
Um die E-Mail-Funktionalitäten des Systems während der Entwicklungsphase zu testen, wurde auf den fake-SMTP Server Mailtrap [18] zurückgegriffen. Mailtrap bietet die Möglichkeit, realitätsnah den E-Mail-Versand zu testen und dabei keine echten E-Mail-Adressen oder SMTP-Server zu benötigen. Laravel bietet außerdem standardmäßig eine Schnittstelle zum Mailtrap-Service, was die Wahl für den Dienst erleichterte.

5.2 Auszüge aus der Implementierung

An dieser Stelle werden anhand ausgewählter Beispiele gezeigt, wie Teilaspekte innerhalb der Implementierung umgesetzt wurden. Als Beispiele wurden drei integrale Anwendungsfälle des Systems gewählt. Kapitel 5.2.1 verdeutlicht die Implementierung der App-Suche und Kapitel 5.2.2 die Übernahme in die System-Datenbank. Kapitel 5.2.3 veranschaulicht die Umsetzung des Anwendungsfalls App reviewen.

5.2.1 Apps suchen

Das Durchsuchen der App-Stores und Speichern der relevanten Apps in die System-Datenbank ist einer der ersten durchgeführten Schritte im Systemzyklus. Wie in Kapitel 4 beschrieben, sind die beiden für das Durchsuchen der App-Stores verantwortlichen Module vom restlichen System getrennt. In einem früheren Status der Implementierung liefen diese Module noch innerhalb des Webbrowsers der Benutzer. Stößt der Web-Server eine neue Suche an, werden die Details zum Client gesendet und die Suchanfrage dort ausgeführt. Nachdem die Ergebnisse empfangen wurden, werden diese zum Server zurück geschickt. Dieses Vorgehen war nicht zufriedenstellend, da Cross-Origin-Requests ein Sicherheitsrisiko darstellen und von den Webbrowsers verhindert werden. Um dieses Problem zu umgehen und den Datenzugriff zu vereinfachen, wurde der direkte Datenaustausch über eine externe API implementiert. Listing 5.1 zeigt das Erstellen einer Suchanfrage.

```
1  $base_url = config('app.scraping_url');
2  $fields = array('q' => $request->searchTerm,
3     'num' => $request->searchQuantity,
4     'lang' => $request->searchLanguage,
5     'country' => $request->searchCountry,
6     'fullDetail' => true
7  );
8  $search_url = 'api/apps/';
9  $url = $base_url . $search_url . "?" . http_build_query($fields, '', "&")
       ;
10 $data = file_get_contents($url);
11 $googleResults = json_decode($data);
```

Listing 5.1: Erstellen einer Suche und Aufnahme der Ergebnisse

Die Adresse des Scraper-Servers kann in der System-Konfiguration angegeben werden und wird in Listing 5.1, Zeile 1 abgerufen. In Listing 5.1, Zeile 2-7 wird ein Teil der Suchanfrage aufgebaut, die sich aus den Benutzereingaben im Suchfeld zusammensetzt. Die Scraper für Googles Play-Store und Apples App-Store sind beide über die selbe API, allerdings über unterschiedliche Adressen anzusprechen. In diesem Beispiel sollen Android Apps gefunden werden, dementsprechend zeigt Listing 5.1, Zeile 8 die Adresse des Play Store Scraper. In Listing 5.1, Zeile 9 wird die komplette Adresse der Suchanfrage zusammengesetzt, in Listing 5.1, Zeile 10 benutzt, die Suchanfrage erstellt und das Ergebnis im JSON-Format aufgenommen. Ein Ausschnitt der Rückgabe einer solchen Suchanfrage ist in Abbildung 5.1 zu sehen.

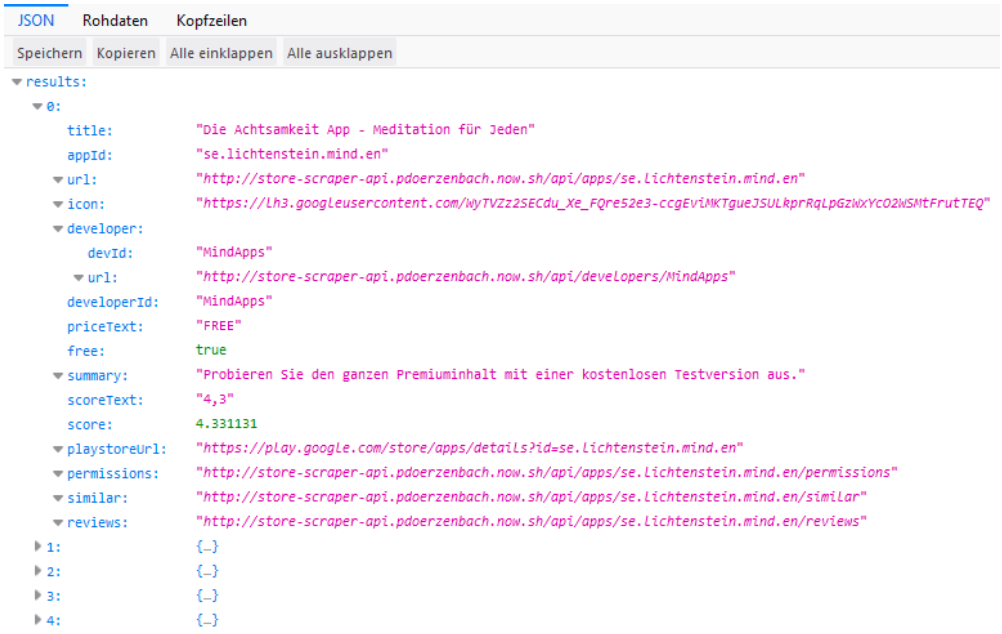


Abbildung 5.1: Rückgabe der Suchergebnisse im JSON-Format

Falls ein Eintrag der Ergebnisrückgabe ein Duplikat einer App ist, die bereits in der System-Datenbank existiert, soll der Benutzer diese Information in der Übersicht der Suchergebnisse angezeigt bekommen. Daher wird die Ergebnisliste durchlaufen und jeder Eintrag auf ein mögliches Duplikat überprüft. Listing 5.2 zeigt diese Überprüfung.

```
1 if (App::where([
2     ['app_id', '=', $resultData->appId],
3     ['version', '=', $resultData->version]
4 ])->exists()) {
5     $originalApp = App::where('app_id', '=', $resultData->appId)->first();
6     $original = $originalApp->id;
7 } else {
8     $original = null;
9 }
```

Listing 5.2: Überprüfen der Suchergebnisse auf Duplikate

Der Codeausschnitt verdeutlicht, dass die Werte *App-Id* und *Version* der Scraper-Rückgaben zur Duplikatserkennung überprüft werden (Listing 5.2, Zeile 2 und 3). Sind diese Werte identisch mit den Werten einer App in der Datenbank, liegt ein Duplikat vor. Ist ein Suchergebnis ein Duplikat, wird die bereits in der Datenbank stehende App mit dem Suchergebnis verknüpft (Listing 5.2, Zeile 6). Andererseits erhält es keinen Verweis (Listing 5.2, Zeile 8). Dieser Schritt ist wichtig für die eventuelle Übernahme dieses Suchergebnisses als App in die Datenbank. Möchte der Benutzer ein Duplikat übernehmen, muss vorher der Übernahme Modus ausgewählt werden.
Nach der Duplikatsprüfung kann ein Suchergebnis als solches in die Datenbank geschrieben werden. Listing 5.3 veranschaulicht, wie ein Suchergebnis aus den Werten der Scraper-Rückgabe erstellt (Listing 5.3, Zeile 1-7) und in der Datenbank abgespeichert wird (Listing 5.3, Zeile 8).

```
1 $searchResult = SearchResult::create([
2     'search_id' => $search->id,
3     'app_id' => $resultData->appId,
4     [...]
5     'version' => $resultData->version,
6     'original_app' => $original
7 ]);
8 $searchResult->save();
```

Listing 5.3: Persistieren eines Suchergebnisses in die Datenbank

5.2.2 Apps in die Datenbank speichern

Listing 5.4 veranschaulicht das Vorgehen zum Speichern der Apps in die Datenbank.

```
1  $duplicates = collect();
2  $originals = collect();
3
4  foreach ($request->addedSearchResults as $resultId) {
5      $searchResult = SearchResult::find($resultId);
6      if (is_null($searchResult->original_app)) {
7          $originals->push($searchResult);
8      } else {
9          $duplicates->push($searchResult);
10     }
11 }
12
13 if ($duplicates->isEmpty()) {
14     foreach ($originals as $original) {
15         $app = App::create([
16             'app_id' => $original->app_id,
17             'developer' => $original->developer,
18             [...]
19             'url' => $original->url,
20             'version' => $original->version,
21         ]);
22         $app->save();
23     }
24 } else {
25     $request->session()->put('search', $search);
26     $request->session()->put('duplicates', $duplicates);
27     $request->session()->put('originals', $originals);
28
29     return redirect('searches/' . $search->id . '/duplicates');
30 }
```

Listing 5.4: Behandlung von Originalen und Duplikaten bei der App-Speicherung

Möchte der Benutzer Suchergebnisse als App in die Datenbank übernehmen, muss zunächst für jedes Suchergebnis überprüft werden, ob es sich um ein Duplikat handelt. Ist dies nicht der Fall, kann die App direkt aus den Daten des Suchergebnisses

erstellt und in die Datenbank geschrieben werden. Bei Duplikaten muss ein Zwischenschritt eingefügt werden, indem der Benutzer entscheidet, anhand welches Übernahmemodus die App gespeichert wird. Zu diesem Zweck werden zunächst alle zu übernehmenden Suchergebnisse je nach Vorhandensein eines Verweises auf eine Originalapp auf separate *Collections* [1] verteilt (Listing 5.4, Zeile 1 und 2). Sollte die *duplicates*-Collection nach diesem Schritt leer sein, können mit der *originals*-Collection neue Apps erstellt und abgespeichert werden (Listing 5.4, Zeile 13-23). Andererseits werden die benötigten Daten in die Session geschrieben um im nächsten Request verfügbar zu sein und die Duplikatsbehandlungs-View geladen (Listing 5.4, Zeile 24-30).

Im nächsten Schritt kann der Benutzer zwischen den beiden Übernahme-Modi *Verknüpfen* und *Duplizieren* wählen. Sollen die Suchergebnisse Duplikate in die App-Datenbank schreiben, wird der Duplizierungs-Modus ausgewählt. Wählt er jedoch *Verknüpfen*, wird keine neue App erstellt, sondern nur die Suche mit der bereits vorhandenen App verknüpft. Dies ist in Listing 5.5 zu sehen.

```
1 if ($request->storeMode == 'merge') {
2     foreach ($duplicates as $duplicate) {
3         $duplicate->update([
4             'store' => true
5         ]);
6         $app = App::where('app_id', '=', $duplicate->app_id)->first();
7         $app->searches()->sync($search, false);
8     }
9 }
```

Listing 5.5: Verknüfen von Suche und App

[1] Laravels Collection-Klasse ist ein Datentyp, der eine Menge von Daten aufnehmen kann und mit zahlreichen vorab implementierten Hilfsfunktionen ausgestattet ist.

5.2.3 Apps reviewen

Haben die Experten Reviews erhalten, können diese bearbeitet werden. Die MARS-G (siehe Kapitel 2.1) ist im System mithilfe der SurveyJS-Bibliothek implementiert. Diese erlaubt es, den Fragebogen bequem mit dem Creator-Tool [3] zu erstellen. Der so entstandene Fragebogen wird im JSON-Dateiformat in die Datenbank gespeichert und bei Bedarf aus dieser geladen. Hierfür müssen die Daten aus der Datenbank an die JavaScript-Komponente der Webseite weitergegeben werden. Um diesen Prozess zu vereinfachen wurde ein Package benutzt, das PHP Variablen in JavaScript zugänglich macht [11]. Listing 5.6 zeigt diesen Vorgang.

```
1 JavaScript::put([
2     'json' => $review->survey->template,
3     'answers' => $review->content,
4     'currentPage' => $review->current_page,
5     'sections' => $review->survey->average_sections,
6     'url' => $url,
7     'token' => $token,
8 ]);
```

Listing 5.6: Weitergabe der PHP-Variablen an JavaScript

Neben dem Fragebogen-Template (Listing 5.6, Zeile 2) werden weitere Variablen an JavaScript übergeben. Die *answers* und *currentPage*-Variablen (Listing 5.6, Zeile 3-4) beinhalten mögliche Benutzerantworten und die zuletzt angezeigte Fragebogenseite, die bei einer Wiederaufnahme des Reviews benötigt wird. Die *sections*-Variable (Listing 5.6, Zeile 5) wird für die spätere Mittelwertberechnung benutzt, *url* und *token* (Listing 5.6, Zeile 6-7) für das Abschicken des Fragebogens nach Vollendung oder Speicherung. Sind diese Variablen in JavaScript zugänglich kann der Fragebogen geladen werden.
Dazu wird zuerst das benötigte Skript in Listing 5.7, Zeile 1 in die Seite geladen. In den weiteren Zeilen folgen das Laden des Fragebogen-Templates (Listing 5.7, Zeile 3), der Expertenantworten (Listing 5.7, Zeile 4), der aktuellen Seite (Listing 5.7, Zeile 5) und der Mittelwert-Sektionen (Listing 5.7, Zeile 6). Listing 5.7, Zeile 8-12 zeigt das Systemverhalten, wenn der Experte sein Review beendet abschließt.

```
<script src="https://surveyjs.azureedge.net/1.0.75/survey.jquery.min.js"
    ></script>

window.survey = new Survey.Model(reviews.json);
survey.data = JSON.parse(reviews.answers);
survey.currentPageNo = reviews.currentPage;
var sections = JSON.parse(reviews.sections);

survey.onComplete.add(function () {
    completed = true;
    calculateAverages();
    makeRequest();
});
```

Listing 5.7: Bereitstellung des Fragebogens in der HTML-Seite

Zuerst werden die Mittelwerte der Expertenantworten der einzelnen Sektionen berechnet Dies ist in Listing 5.8 veranschaulicht.

```
function calculateAverages () {
    sections.forEach(function(section) {
        var questions = survey.getPanelByName(section).questions;
        var sum = 0;
        var zeroes = 0;
        questions.forEach(function(question) {
            if(question.value != 0){
                sum += parseInt(question.value, 10);
            } else {
                zeroes += 1;
            }
        });
        var avg = sum/(questions.length - zeroes);
        averages[section] = avg;
    });
};
```

Listing 5.8: Berechnen der Mittelwerte der MARS-G-Sektionen

Dabei werden Fragen, die mit 0 bewertet wurden, nicht beachtet (Listing 5.8, Zeile 7-11).
Anschließend wird der Fragebogen zurück an den Server gesendet. Listing 5.9 zeigt die Anfrage.

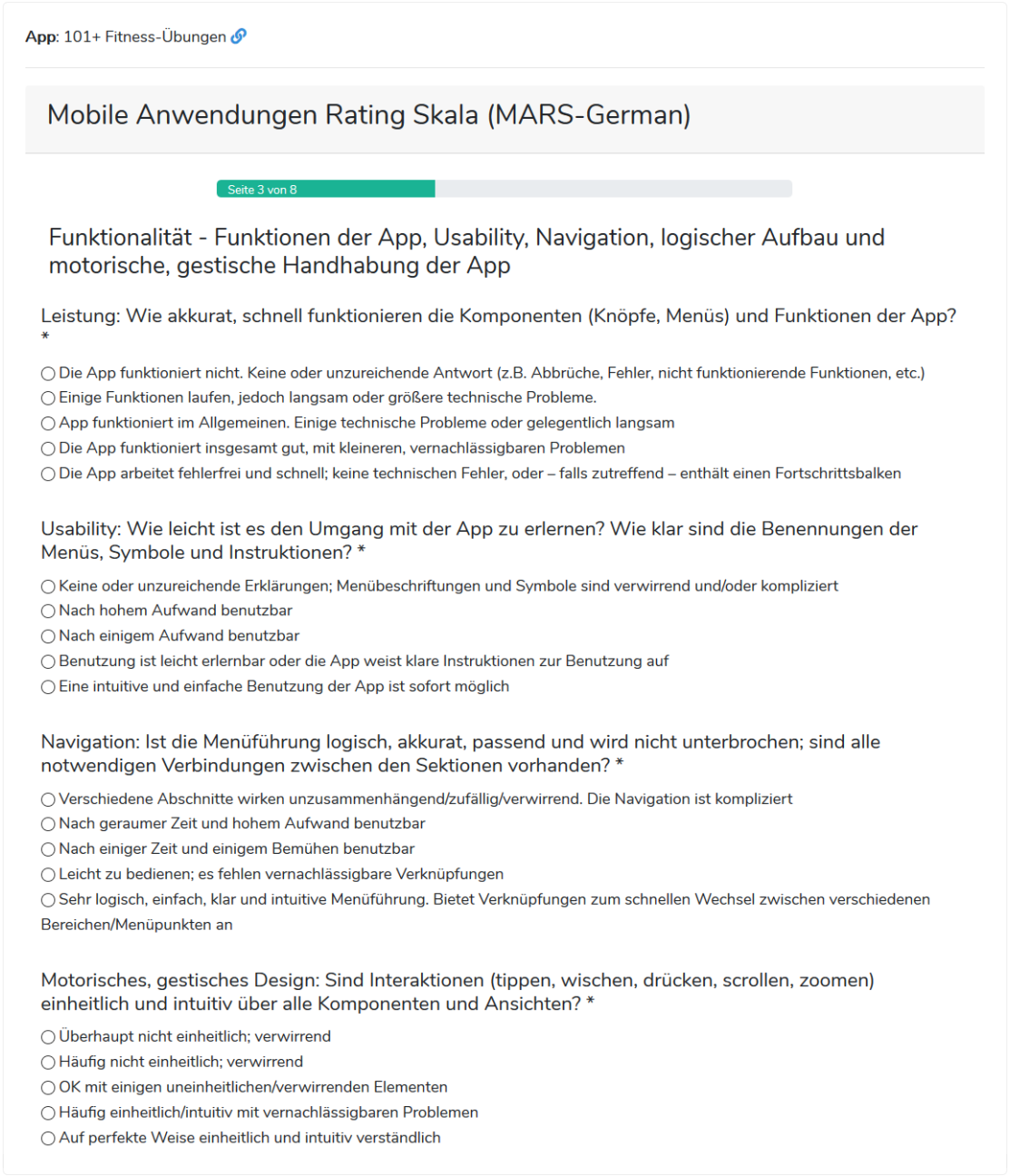
```
1  function makeRequest() {
2     var xhr = new XMLHttpRequest();
3     xhr.onreadystatechange = function () {
4        if(xhr.readyState === 4 && xhr.status === 200) {
5           //redirect
6           window.location.assign(xhr.responseText);
7        }
8     };
9     xhr.open('POST', reviews.url, true);
10    xhr.setRequestHeader('X-CSRF-TOKEN', reviews.token);
11    var data = JSON.stringify(survey.data).slice(0, -1) + ',"currentPage":
      ' + currentPage + ',"completed":' + completed + '}';
12    if (completed) {
13       data = data.slice(0, -1) + ',' + JSON.stringify(averages).slice(1,
      -1) + '}';
14    }
15    xhr.send(data);
16 };
```

Listing 5.9: Berechnen der Mittelwerte der MARS-G-Sektionen

Um den Fragebogen schlussendlich auf der Webseite anzeigen zu lassen, muss eine Zeile in die HTML-Seite eingefügt werden:

```
1  <div id="surveyContainer"></div>
```

Das Ergebnis dieses Fragebogen-Imports zeigt Abbildung 5.2.



App: 101+ Fitness-Übungen

Mobile Anwendungen Rating Skala (MARS-German)

Seite 3 von 8

Funktionalität - Funktionen der App, Usability, Navigation, logischer Aufbau und motorische, gestische Handhabung der App

Leistung: Wie akkurat, schnell funktionieren die Komponenten (Knöpfe, Menüs) und Funktionen der App? *

- ◯ Die App funktioniert nicht. Keine oder unzureichende Antwort (z.B. Abbrüche, Fehler, nicht funktionierende Funktionen, etc.)
- ◯ Einige Funktionen laufen, jedoch langsam oder größere technische Probleme.
- ◯ App funktioniert im Allgemeinen. Einige technische Probleme oder gelegentlich langsam
- ◯ Die App funktioniert insgesamt gut, mit kleineren, vernachlässigbaren Problemen
- ◯ Die App arbeitet fehlerfrei und schnell; keine technischen Fehler, oder – falls zutreffend – enthält einen Fortschrittsbalken

Usability: Wie leicht ist es den Umgang mit der App zu erlernen? Wie klar sind die Benennungen der Menüs, Symbole und Instruktionen? *

- ◯ Keine oder unzureichende Erklärungen; Menübeschriftungen und Symbole sind verwirrend und/oder kompliziert
- ◯ Nach hohem Aufwand benutzbar
- ◯ Nach einigem Aufwand benutzbar
- ◯ Benutzung ist leicht erlernbar oder die App weist klare Instruktionen zur Benutzung auf
- ◯ Eine intuitive und einfache Benutzung der App ist sofort möglich

Navigation: Ist die Menüführung logisch, akkurat, passend und wird nicht unterbrochen; sind alle notwendigen Verbindungen zwischen den Sektionen vorhanden? *

- ◯ Verschiedene Abschnitte wirken unzusammenhängend/zufällig/verwirrend. Die Navigation ist kompliziert
- ◯ Nach geraumer Zeit und hohem Aufwand benutzbar
- ◯ Nach einiger Zeit und einigem Bemühen benutzbar
- ◯ Leicht zu bedienen; es fehlen vernachlässigbare Verknüpfungen
- ◯ Sehr logisch, einfach, klar und intuitive Menüführung. Bietet Verknüpfungen zum schnellen Wechsel zwischen verschiedenen Bereichen/Menüpunkten an

Motorisches, gestisches Design: Sind Interaktionen (tippen, wischen, drücken, scrollen, zoomen) einheitlich und intuitiv über alle Komponenten und Ansichten? *

- ◯ Überhaupt nicht einheitlich; verwirrend
- ◯ Häufig nicht einheitlich; verwirrend
- ◯ OK mit einigen uneinheitlichen/verwirrenden Elementen
- ◯ Häufig einheitlich/intuitiv mit vernachlässigbaren Problemen
- ◯ Auf perfekte Weise einheitlich und intuitiv verständlich

Abbildung 5.2: Raten einer App mit der MARS

6 Anforderungsabgleich

Es folgt der Abgleich der in Abschnitt 3.1 und 3.2 definierten Funktionalitäten mit der tatsächlich erreichten Funktionalität des entwickelten Systems. Dabei wird zwischen sechs Erfüllungsgraden unterschieden:

(5) Die Funktionalität wurde vollständig erfüllt.
(4) Die Funktionalität wurde zufriedenstellend erfüllt.
(3) Die Funktionalität wurde ausreichend erfüllt.
(2) Die Funktionalität wurde teilweise erfüllt.
(1) Die Funktionalität wurde vorbereitet.
(0) Die Funktionalität wurde nicht erfüllt.

6.1 Funktionale Anforderungen

Diese Tabelle betrachtet die Erfüllung der in Abschnitt 3.1 beschriebenen funktionalen Anforderungen.

Kürzel	Bezeichnung	Priorität	Erfüllungsgrad
Account:			
F-01	Autorisierung	MUSS	5
F-02	Registrierung	MUSS	5
F-03	Authentifizierung	MUSS	5
F-04	Benutzerverwaltung	MUSS	5

Kürzel	Bezeichnung	Priorität	Erfüllungsgrad
Benutzer:			
F-05	Administrator hinzufügen	MUSS	5
F-06	Administrator bearbeiten	MUSS	5
F-07	Administrator löschen	MUSS	5
F-08	Reviewer-Übersicht anzeigen lassen	SOLL	5
F-09	Reviewer hinzufügen	MUSS	5
F-10	Reviewer bearbeiten	MUSS	5
F-11	Reviewer löschen	MUSS	5
Gruppen und Kategorie:			
F-12	Gruppe hinzufügen	SOLL	5
F-13	Gruppe bearbeiten	SOLL	5
F-14	Gruppe löschen	SOLL	5
F-15	Kategorie hinzufügen	SOLL	5
F-16	Kategorie bearbeiten	SOLL	5
F-17	Kategorie löschen	SOLL	5
Suche:			
F-18	App-Stores durchsuchen	MUSS	4
F-19	Apps in Datenbank speichern	MUSS	5
F-20	App-Suche zwischenspeichern	SOLL	5
F-21	App-Suche wiederaufnehmen	SOLL	5
F-22	App-Suche exportieren	MUSS	5
F-23	App-Übersicht anzeigen lassen	MUSS	5
F-24	App bearbeiten	MUSS	5
F-25	App löschen	MUSS	5
F-26	Reviewer zuweisen (automatisch und manuell)	MUSS	4

Kürzel	Bezeichnung	Priorität	Erfüllungsgrad
Review:			
F-27	Review-Konflikte anzeigen lassen	MUSS	5
F-28	Review-Übersicht anzeigen lassen	MUSS	5
F-29	App reviewen	MUSS	5
F-30	App ablehnen	MUSS	5
F-31	Review zwischenspeichern	SOLL	5
F-32	Review wiederaufnehmen	SOLL	5

6.2 Nicht-funktionale Anforderungen

Analog zu den funktionalen Anforderungen beschreibt die folgende Tabelle den Erfüllungsgrad der in Abschnitt 3.2 nicht-funktionalen Anforderungen.

Kürzel	Bezeichnung	Priorität	Erfüllungsgrad
System:			
NF-01	Zuverlässigkeit	MUSS	5
NF-02	Robustheit	MUSS	4
NF-03	Sicherheit/Datenschutz	MUSS	4
NF-04	Effizienz	SOLL	5
NF-05	Wartbarkeit	KANN	4
NF-06	Erweiterbarkeit	SOLL	5

7 Zusammenfassung und Ausblick

Dieses Kapitel fasst die Zielsetzung und den tatsächlich geleisteten Beitrag der Arbeit zusammen. Zuerst wird in Kapitel 7.1 das entwickelte System und die dadurch entstandenen Verbesserungen betrachtet, bevor in Kapitel 7.2 ein Ausblick auf darauf aufbauende Erweiterungen der Anwendung gegeben wird.

7.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein System entwickelt, das das systematische Bewerten von medizinischen und psychologischen Anwendungen ermöglicht. Dazu wurde die wissenschaftliche Skala MARS vorgestellt, mit der die Bewertungen durchgeführt werden. Außerdem wurde das MHAD-Projekt erläutert, in das die Ergebnisse des im System stattfindenden Reviewprozesses fließen. Es wurden Systemanforderungen analysiert und darauf aufbauend eine Systemarchitektur definiert. Diese Architektur wurde so gewählt, dass auf unvermeidliche, temporäre Systemausfälle schnell reagiert werden kann. Anhand dieser Architektur wurden die Systemfunktionalitäten implementiert.
Es entstand eine Anwendung, die alle bisher getrennt durchgeführten Arbeitsschritte des Reviewprozesses vereint und so die Effizienz enorm erhöht. Das System verfügt über eine ausgeprägte Benutzerverwaltung, in der sich neue Experten selbst registrieren können, oder von Administratoren hinzugefügt werden. Diese Experten lassen sich in Gruppen gliedern und mit Expertisen versehen. Weiterhin bietet die Anwendung die Möglichkeit, die beiden App-Stores automatisch nach Gesundheitsapps durchsuchen zu lassen und die gefundenen Apps in die Systemdatenbank zu übertragen. Diese Apps können manuell oder automatisiert an die Experten verteilt werden, die wiederum ein komplettes App-Review innerhalb des Systems durchführen können. Fertiggestellte Reviews werden automatisch auf Bewertungskonflikte

mit anderen Reviews zu dieser App überprüft, die Mittelwerte der Sektionen berechnet und so für den Import in die MHAD vorbereitet.

7.2 Ausblick

Diese Arbeit erfüllt alle Voraussetzungen zur systematischen Bewertung deutschsprachiger Gesundheitsapps. Sie entstand mit dem Fokus auf der Bewertungsskala MARS-G. Während der Implementierung wurde allerdings bereits bedacht, dass die Bewertung auf der Basis anderer Skalen möglich sein sollte und deshalb der unkomplizierte Import weiterer Fragebögen durch die Systemarchitektur vorbereitet. Aktuell müssen Veränderungen am Fragebogen noch direkt auf der Datenbankebene durchgeführt werden. Es ist allerdings denkbar, die Anwendung so zu erweitern, dass das Erstellen und Bearbeiten von Fragebögen über Benutzungsschnittstellen innerhalb des Systems umgesetzt werden kann. Insbesondere wird die englische Version der MARS implementiert werden müssen, da sich die Bewertung nicht nur auf deutschsprachige Apps beschränken soll. Aus diesem Grund wurde die Systemoberfläche bereits ins englische übersetzt.

Weiterhin ist es denkbar, die Analyse von Reviews und Suchen anhand zusätzlicher Funktionen zu erweitern. Deren Export beschränkt sich zur Zeit auf die wesentlichen Informationen wie Such- oder Appdetails, die Übernahme in die Datenbank oder Review-Mittelwerte. Während der Entwicklung wurden allerdings bereits weitere Funktionen von den zukünftigen Endbenutzern angefragt, deren Umfang den Rahmen dieser Arbeit überschritten hätte. Die Implementierung dieser Wünsche ist ein weiterer Anwendungsfall einer sinnvollen Systemerweiterung.

Auch beim Löschen der Daten innerhalb des Systems gibt es die Möglichkeit, die Anwendung zu ergänzen. Aktuell werden beispielsweise Kategorien und Gruppen endgültig aus dem System gelöscht, wenn es durch einen Administrator angestoßen wird. Dieser Vorgang lässt sich nicht rückgängig machen. Registrierte Benutzer können zwar nur deaktiviert werden, verlieren dabei aber ihre Gruppen- oder Kategoriemitgliedschaften. Außerdem müssen erstellte Reviews auf andere Experten umgeschrieben werden, sollte man den Ersteller deaktivieren. Das Abspeichern des aktuellen Zustands einer Ressource zur späteren Wiederherstellung wurde

zwar schon während der Implementierungsphase angedacht, allerdings wegen des damit verbundenen Aufwands auf einen späteren Zeitpunkt außerhalb des Rahmens dieser Arbeit verschoben.

Alle hier erwähnten Erweiterungen sind denkbar und erweitern das System sinnvoll. In Kooperation mit den zukünftigen Endnutzern sollte die geplante Weiterentwicklung evaluiert werden, um die Realisierung der verschiedenen Zusatzkomponenten angemessen zu priorisieren und so den Nutzen des Systems zu maximieren.

Literatur

[1] Apache Friends. *XAMPP, Version 7.3.0*. https://www.apachefriends.org/de/index.html (Abgerufen: 12.04.2019). 2018.

[2] Stephen Armstrong. „Which App Should I Use?“ In: *BMJ (Clinical research ed.)* 351 (2015). DOI: 10.1136/bmj.h4597.

[3] Devsoft Baltic OÜ. *SurveyJS Creator Tool*. https://surveyjs.io/create-survey/ (Abgerufen: 13.06.2019). 2019.

[4] Devsoft Baltic OÜ. *SurveyJS, Version 1.0.75*. https://surveyjs.io/ (Abgerufen: 12.04.2019). 2019.

[5] Andrea Girardello und Florian Michahelles. „AppAware: Which Mobile Applications Are Hot?“ In: *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '10*. Hrsg. von Marco de Sá, Luís Carriço und Nuno Correia. New York, USA: ACM Press, 2010, S. 431–434. ISBN: 9781605588353. DOI: 10.1145/1851600.1851698.

[6] GitHub, Inc. *Atom Text Editor, Version 1.36.0*. https://atom.io/ (Abgerufen: 12.04.2019). 2019.

[7] GitHub, Inc. *GitHub*. https://github.com/ (Abgerufen: 24.06.2019). 2019.

[8] GitHub, Inc. *facundoolano/app-store-scraper: scrape data from the itunes app store, Version 0.16.0*. https://github.com/facundoolano/app-store-scraper (Abgerufen: 17.06.2019). 2019.

[9] GitHub, Inc. *facundoolano/google-play-api: Turns google-play-scraper into a RESTful API*. https://github.com/facundoolano/google-play-api (Abgerufen: 17.06.2019). 2019.

[10] GitHub, Inc. *facundoolano/google-play-scraper: Node.js scraper to get data from Google Play, Version 6.2.6*. https://github.com/facundoolano/google-play-scraper (Abgerufen: 17.06.2019). 2019.

[11] GitHub, Inc. *laracasts/PHP-Vars-To-Js-Transformer: Transform PHP data to JavaScript*. https://github.com/laracasts/PHP-Vars-To-Js-Transformer (Abgerufen: 18.06.2019). 2019.

[12] Jeffrey Wey. *Laracasts*. https://laracasts.com/ (Abgerufen: 12.04.2019). 2019.

[13] Bart de Langhe, Philip M. Fernbach und Donald R. Lichtenstein. „Navigating by the Stars: Investigating the Actual and Perceived Validity of Online User Ratings“. In: *Journal of Consumer Research* 42.6 (2016), S. 817–833. ISSN: 0093-5301. DOI: 10.1093/jcr/ucv047.

[14] MARS-Rating Universität Ulm. *MARS - Mobile Anwendungen Rating Skala*. https://www.youtube.com/watch?v=5vwMiCWC0Sc (Abgerufen: 27.05.2019). YouTube. 2017.

[15] Microsoft Corporation. *Microsoft Excel, Version 2019*. https://products.office.com/de-de/excel (Abgerufen: 14.06.2019). 2019.

[16] Netscape Communications Corporation, Mozilla Foundation and Ecma International. *JavaScript, Version 1.8.5*. https://developer.mozilla.org/de/docs/Web/JavaScript (Abgerufen: 17.06.2019). 2010.

[17] Node.js Foundation. *Node.js, Version 10.16.0*. https://nodejs.org/ (Abgerufen: 13.06.2019). 2019.

[18] Railsware. *Mailtrap*. https://mailtrap.io/ (Abgerufen: 12.04.2019). 2019.

[19] Spatie. *laravel-permission, Version 2.37.0*. https://github.com/spatie/laravel-permission (Abgerufen: 16.06.2019). 2019.

[20] SpryMedia Ltd. *DataTables, Version 1.10.18*. https://cdn.datatables.net/ (Abgerufen: 12.04.2019). 2018.

[21] Stoyan R. Stoyanov u. a. „Mobile App Rating Scale: A New Tool for Assessing the Quality of Health Mobile Apps“. In: *JMIR mHealth and uHealth* 3.1 (2015), e27. ISSN: 2291-5222. DOI: 10.2196/mhealth.3422.

[22] Taylor Otwell. *Laravel Documentation: Eloquent*. https://laravel.com/docs/5.7/eloquent (Abgerufen: 22.06.2019). 2018.

[23] Taylor Otwell. *Laravel, Version 5.7*. https://laravel.com/ (Abgerufen: 12.04.2019). 2018.

[24] The MariaDB Foundation. *MariaDB, Version 10.1.37*. https://mariadb.org/ (Abgerufen: 12.04.2019). 2014.

[25] The PHP Group. *PHP Language Reference, Version 7.3.0*. https://www.php.net/ (Abgerufen: 12.04.2019). 2018.

[26] The jQuery Team. *jQuery, Version 3.4.1*. https://jquery.com/ (Abgerufen: 24.06.2019). 2019.

[27] Twitter. *Bootstrap, Version 4.0.0*. https://getbootstrap.com/ (Abgerufen: 12.04.2019). 2018.

[28] Universität Ulm. *Mobile Health App Database*. http://mhad.science/ (Abgerufen: 27.05.2019). 2019.

[29] Web Hypertext Application Technology Working Group. *Hypertext Markup Language, Version 5.2*. https://www.w3.org/TR/2017/REC-html52-20171214/ (Abgerufen: 17.06.2019). 2017.

[30] Wikipedia-Autoren. *Cross-Origin-Request*. https://de.wikipedia.org/w/index.php?title=Cross-Origin-Request&oldid=189229657 (Abgerufen: 22.06. 2019). 2019.

[31] Wikipedia-Autoren. *MoSCoW-Priorisierung*. https://de.wikipedia.org/w/index.php?title=MoSCoW-Priorisierung&oldid=187126821 (Abgerufen: 19.04. 2019). 2019.

[32] Wikipedia-Autoren. *Objektrelationale Abbildung*. https://de.wikipedia.org/w/index.php?title=Objektrelationale_Abbildung&oldid=181202941 (Abgerufen: 22.06. 2019). 2018.

[33] Wikipedia-Autoren. *Programmierschnittstelle*. https://de.wikipedia.org/w/index.php?title=Programmierschnittstelle&oldid=185336207 (Abgerufen: 22.06. 2019). 2019.

[34] Wikipedia contributors. *Web scraping*. https://en.wikipedia.org/w/index.php?title=Web_scraping&oldid=898762637 (Abgerufen: 22.06. 2019). 2019.

[35] Zeit. *Zeit Now, Version 2.0*. https://zeit.co/now (Abgerufen: 12.04.2019). 2018.

A Glossar

API	Ein Softwaresystem stellt anderen Programmen über eine Programmierschnittstelle (englisch application programming interface, API) eine Möglichkeit zur Anbindung an das System zur Verfügung [33].
App-Id	Die App-Id ist der eindeutige Identifikator der App innerhalb ihres App-Stores.
Cross-Origin-Request	Greift eine Webseite oder Webanwendungen auf Ressourcen eines anderen Servers oder einer unterschiedlichen Domain zu, spricht man von einem Cross-Origin-Request. Aus Sicherheitsgründen wird diese Art von Zugriff von Webbrowsern blockiert [30].
Framework	Ein Framework ist ein Programmiergerüst, das dem Programmierer den Entwicklungsrahmen für seine Anwendung definiert indem es beispielsweise APIs spezifiziert und Programmierwerkzeuge bereitstellt.
Job	Ein Job ist eine Systemaufgabe, die vom System auf die Warteschlange gelegt wurde und zu einem späteren Zeitpunkt bearbeitet wird. Ist die Aufgabe bearbeitet worden, wird sie aus der Warteschlange entfernt.
JSON	JavaScript Object Notation (JSON)-Dateien sind für den Menschen einfach lesbare Daten in Textform. Sie werden zum Datenaustausch zwischen Anwendungen verwendet.

Mail-Queue	Warteschlangen (englisch queues), werden im System verwendet, um dem Benutzer vermeidbare Wartezeiten während Systeminteraktionen zu ersparen. Dabei werden Systemaktionen mit erhöhtem Zeitaufwand, beispielsweise das Verschicken einer oder mehrerer E-Mails, nicht sofort ausgeführt, sondern auf eine Warteschlange gelegt. Diese Warteschlange wird vom System zu einem späteren Zeitpunkt abgearbeitet. Dieses Vorgehen hat den Effekt, dass der Benutzer vom erhöhten Zeitaufwand der Systemaktion nichts mitbekommt und effizienter, weil schneller, mit dem System arbeiten kann.
many-to-many	In einer relationen Datenbank definieren many-to-many Beziehungen eine Verbindung zwischen zwei Entitäten, bei denen jede Instanz der ersten Entität mit beliebig vielen Instanzen der zweiten Entität in Verbindung steht und umgekehrt.
ORM	Anwendungsprogramme, die in einer objektorientierten Programmiersprache wie PHP geschrieben sind, brauchen eine Objektrelationale Abbildung (englisch object-relational mapping, ORM) um ihre Objekte in eine relationale Datenbank abzulegen. Diese Technik vereinfacht die Programmierung enorm [32].
Web-Scraping	Web-Scraping bezeichnet das Extrahieren von Daten auf Webseiten. Obwohl auch menschliche Benutzer manuelles Scraping durchführen können, wird mit dem Begriff meist ein automatisierter Prozess mithilfe eines Computerprogramms bezeichnet. Dabei werden die Daten typischerweise von der ursprünglichen Webseite in eine lokale Datenbank kopiert [34].

B MARS-G

Mobile Anwendungen Rating Skala (MARS-German)

Einordung der App

Diese Sektion dient dazu, beschreibende und technische Informationen über die App zu sammeln. Bitte lesen Sie dazu die Beschreibung der App auf iTunes, Google Play, um diese Informationen zu erhalten.

App Name:

Bewertung dieser Version:

Bewertung aller Versionen:

Entwickler:

Anzahl der Bewertungen dieser Version:

Anzahl der Bewertungen aller Versionen:

Version:

Letztes update:

Kosten für die Basisversion:

Kosten für die erweiterte Version:

Plattform:

- □ iOS
- □ Android
- □ Windows Mobile
- □ Andere:_____

Kurze Beschreibung:

Angliederung

□ Unbekannt □ Kommerziell □ Regierung □ Gemeinnützige Organisation (NGO) □ Universität

Fokus: Ziel der App (Mehrfachauswahl)

1. Verbesserung von Befindlichkeit, Wohlbefinden
2. Reduktion depressiver Symptome
3. Reduktion von Angst
4. Reduktion von Stress
5. Verbesserung der Emotionsregulation
6. Unterstützung bei Verhaltensänderungen
7. Reduktion von Suchtverhalten
8. Unterstützung bei der Erreichung individueller Ziele
9. Unterhaltung
10. Verbesserung des Sozialverhaltens
11. Förderung körperlicher Gesundheit
12. Andere:_____

Theoretischer Hintergrund bei Interventionen (Mehrfachauswahl)

13. Verhaltenstherapie
14. Kognitive Verhaltenstherapie (Verhaltenstherapie der zweiten Welle)
15. Verhaltenstherapie der dritten Welle (z. B. Mindfulness-based Psychotherapie, Akzeptanz und Commitment, Schematherapie, CBASP, DBT)
16. Systemische Therapie
17. Psychodynamische Psychotherapie
18. Humanistische Therapie
19. Integrative Therapie (z. B. Interpersonelle Therapie)

Andere:_____

Methoden (Mehrfachauswahl)

1. EMDR-Methode (zur Behandlung der Posttraumatischen Belastungsstörung),
2. Hypnotherapie (zur Raucherentwöhnung und zur Mitbehandlung bei somatischen Erkrankungen)
3. Datenerhebung, Messungen
4. Monitoring, Tracking
5. Feedback
6. Information, Edukation
7. Tipps, Ratschläge
8. Verfolgen eigener Ziele
9. Strategien, Fertigkeiten, Training (z.B.: Problemlösen, Stressmanagement, etc.)
10. Entspannungsübungen
11. Atemübungen
12. Körperübungen (z.B.: Yoga, Pilates, PMR)
13. Achtsamkeit/ Dankbarkeit
14. Akzeptanz
15. Ressourcenorientierung
16. Exposition
17. Gamification
18. Serious Games
19. Alternativmedizinische Interventionselemente (Homöopathie, Akupressur, etc.)
20. Erinnerungen, Reminder, Verstärker, etc.
21. Maßgeschneiderte Interventionen (tailored interventions), Echtzeit-Feedback
22. Schulmedizin
23. Andere:_____

Altersgruppe (Mehrfachauswahl)

24. Kleinkinder (unter 6)
25. Kinder (6-12)
26. Jugendliche (13-17)
27. Junge Erwachsene (18-25)
28. Erwachsene (26-64)
29. Ältere Erwachsene (ab 65)
30. Unspezifiziert

Technische Aspekte der App

31. Austausch mit anderen ist möglich (Facebook, Whatsapp, Telegram etc.)
32. Verfügt über eine App-Gemeinschaft
33. Benötigt Internetzugang
34. Andere:_____

Kategorien im App-Store

35. Lifestyle
36. Medizin
37. Gesundheit & Fitness
38. Andere: _____

Einbettung in therapeutische Angebote

47. Keine
48. Kommunikation mit Behandlern ist über die App oder ein Webinterface möglich
49. Zuteilung von Modulen durch den Behandler ist möglich
50. Teilen von Inhalten (z.B.: Fragebogenergebnisse, Nutzerstatistiken,

Einsatzbereich

39. Prävention

40. Behandlung

41. Rehabilitation

42. Nachsorge

43. Andere

Behandlungsart

□ Stand-Alone

□ Blended Care (BC)

□ Nicht anwendbar

Unterstützung

□ Guided synchron

□ Guided asynchron

□ Technically guided

□ Unguided

Zertifizierung

44. Keine

45. Entspricht dem Medizinproduktegesetz

46. Andere: _____

Bearbeitungsfortschritt, etc.) mit dem Behandler ist möglich

51. Andere: _____

Sicherheit und Datenschutz

O Ja O Nein Erlaubt Passwortnutzung

O Ja O Nein Erfordert einen Login

O Ja O Nein Verfügt über eine Datenschutzerklärung

O Ja O Nein Erfordert das aktive Bestätigen einer Einverständniserklärung

O Ja O Nein Hinweise zur Finanzierung/ Interessenskonflikten

O Ja O Nein Kontakt/Ansprechperson/ Impressum

O Ja O Nein Sicherheit der Datenübertragung

O Ja O Nein Ort der Datenspeicherung

O Ja O Nein Werden Daten mit Dritten geteilt

O Ja O Nein Notfallfunktionen sind vorhanden

O Ja O Nein Sicherheitsstrategien bei Handyverlust

O Ja O Nein Andere:_____

App Qualitätsbewertung

Das MARS Rating erfasst die Qualität von Apps auf 4 Dimensionen. Alle Fragen werden von 1 „Mangelhaft“ bis 5 „Exzellent“ bewertet. Bitte markieren Sie die Nummer die sie am zutreffendsten empfinden. Bitte benutzen Sie die Beschreibungen welche in jeder Kategorie zur Verfügung gestellt werden.

SECTION A

Engagement - Spaß, Interesse, individuell Anpassbarkeit, Interaktivität (z.B. sendet Alarme, Nachrichten, Erinnerungen, Feedback, Austausch mit anderen ist möglich), Zielgruppenspezifität

I. Unterhaltung: Ist die App unterhaltsam und lustig zu benutzen? Werden Strategien zur Steigerung des Engagements durch Unterhaltung (z.B.: Gamification) geboten?

1. Langweilig, gar nicht unterhaltsam
2. Überwiegend langweilig
3. OK, unterhaltsam genug um sich für eine kurze Zeit mit der App zu beschäftigen (< 5 Minuten)
4. Mittelmäßig lustig und unterhaltsam, würde einen User für einige Zeit unterhalten (5-10 Minuten)
5. Sehr unterhaltsam und lustig, regt zu mehrmaligen Benutzen an

II. Interesse: Ist die App interessant zu benutzen? Werden Strategien angewandt um das Engagement durch eine interessante Präsentation der Inhalte zu erhöhen?

1. Überhaupt nicht interessant
2. Überwiegend nicht interessant
3. OK, weder interessant noch uninteressant; Nutzer würden sich damit eine kurze Zeit beschäftigen (< 5 Minuten)
4. Mittelmäßig interessant; Nutzer würden sich damit einige Zeit beschäftigen (insgesamt 5-10 Minuten)
5. Überaus interessant, so dass man zur wiederholten Nutzung der App angeregt wird

III. Individuelle Anpassbarkeit: Können alle nötigen Einstellungen den eigenen Wünschen entsprechend angepasst werden und werden diese Anpassungen abgespeichert (z.B. Klänge, Inhalt und Benachrichtigungen)?

1 Individuelle Anpassungen sind nicht möglich oder es ist nötig, diese bei jeder Nutzung neu den eigenen Wünschen anzupassen
2 Einige wenige individuelle Anpassungen sind möglich, aber durch die nur wenigen Möglichkeiten ist die Funktionalität der App beeinträchtigt
3 Grundlegende Einstellungen können individuell angepasst werden, so dass die Funktionalität der App gewährleistet ist
4 Vielfältige Optionen zur individuellen Anpassbarkeit stehen zur Verfügung
5 Eine maßgeschneiderte Anpassung an die Charakteristika und Wünsche der Nutzer ist

möglich und alle individuellen Anpassungen werden für zukünftige Nutzungen übernommen

IV. Interaktivität: Werden Nutzereingaben, Feedback und Aufforderungen (Erinnerungen, Optionen zum Austausch mit anderen, Benachrichtigungen, etc.) gegeben? Bemerkung: Diese Funktionen müssen individuell anpassbar und nicht überfordernd sein.

1. Keine interaktiven Funktionen und/oder keine Eingaben der Nutzer sind möglich
2. Unzureichende Interaktivität, oder Feedback, oder Optionen für Eingaben der User, limitierte Funktionalität
3. Grundlegende interaktive Funktionen, angemessene Funktionalität
4. Vielfältige interaktive Funktionen, Feedback und Optionen für Eingaben der Nutzer sind vorhanden
5. Interaktive Funktionen, Feedback und Optionen für Eingaben der Nutzer sind auf einem so hohen Niveau, dass man das Gefühl hat, die App stellt sich auf die Person des Nutzers ein

V. Zielgruppe: Ist der Inhalt der App (visuelle Aufbereitung, Sprache, Design) passend um die Zielgruppe anzusprechen?

1. Sehr unpassend/unklar/verwirrend
2. Eher unpassend/unklar/verwirrend
3. OK, aber nicht Zielgruppen orientiert. Kann unpassend/unklar/verwirrend sein
4. Zielgruppenspezifisch, mit kleinen Mängeln
5. Sehr zielgruppenspezifisch, keine Probleme

A: Engagement Mittelwert=_____

SECTION B

Funktionalität - Funktionen der App, Usability, Navigation, logischer Aufbau und motorische, gestische Handhabung der App

VI. Leistung: Wie akkurat, schnell funktionieren die Komponenten (Knöpfe, Menüs) und Funktionen der App?

1. Die App funktioniert nicht. Keine oder unzureichende Antwort (z.B. Abbrüche, Fehler, nicht funktionierende Funktionen, etc.)
2. Einige Funktionen laufen, jedoch langsam oder größere technische Probleme.
3. App funktioniert im Allgemeinen. Einige technische Probleme oder gelegentlich langsam
4. Die App funktioniert insgesamt gut, mit kleineren, vernachlässigbaren Problemen
5. Die App arbeitet fehlerfrei und schnell; keine technischen Fehler, oder – falls zutreffend – enthält einen Fortschrittsbalken

VII. Usability: Wie leicht ist es den Umgang mit der App zu erlernen? Wie klar sind die Benennungen der Menüs, Symbole und Instruktionen?

1. Keine oder unzureichende Erklärungen; Menübeschriftungen und Symbole sind verwirrend und/oder kompliziert
2. Nach hohem Aufwand benutzbar
3. Nach einigem Aufwand benutzbar
4. Benutzung ist leicht erlernbar oder die App weist klare Instruktionen zur Benutzung auf
5. Eine intuitive und einfache Benutzung der App ist sofort möglich

VIII. Navigation: Ist die Menüführung logisch, akkurat, passend und wird nicht unterbrochen; sind alle notwendigen Verbindungen zwischen den Sektionen vorhanden?

1. Verschiedene Abschnitte wirken unzusammenhängend/zufällig/verwirrend. Die Navigation ist kompliziert
2. Nach geraumer Zeit und hohem Aufwand benutzbar
3. Nach einiger Zeit und einigem Bemühen benutzbar
4. Leicht zu bedienen; es fehlen vernachlässigbare Verknüpfungen
5. Sehr logisch, einfach, klar und intuitive Menüführung. Bietet Verknüpfungen zum schnellen Wechsel zwischen verschiedenen Bereichen/Menüpunkten an

IX. Motorisches, gestisches Design: Sind Interaktionen (tippen, wischen, drücken, scrollen, zoomen) einheitlich und intuitiv über alle Komponenten und Ansichten?

1. Überhaupt nicht einheitlich; verwirrend
2. Häufig nicht einheitlich; verwirrend
3. OK mit einigen uneinheitlichen/verwirrenden Elementen

4. Häufig einheitlich/intuitiv mit vernachlässigbaren Problemen
5. Auf perfekte Weise einheitlich und intuitiv verständlich

B: Funktionalität Mittelwert=_____

SECTION C

Ästhetik - Graphisches Design, visueller Anreiz, farbliche Gestaltung und stilistische Einheit

X. Layout: Sind das Arrangement und die Größe der Knöpfe/Symbole/Menüführung/ Inhalt passend oder entsprechend zoombar?

1. Mangelhaftes Design – unordentlich/ungeordnet; einige Optionen können nicht ausgewählt, gefunden, gesehen oder gelesen werden; Bildschirm wird nicht optimal genutzt
2. Schwaches Design - nicht einheitlich, unklar; einige Optionen sind schwierig auszuwählen, zu finden, zu sehen und zu lesen.
3. Mittelmäßige Qualität des Designs - einige Probleme Optionen auszuwählen, zu finden, zu sehen und zu lesen.
4. Gutes Design - überwiegend klar; Optionen können ausgewählt, gefunden, gesehen und gelesen werden.
5. Professionelles Design - einfach, klar, übersichtlich, logisch organisiert; Bildschirm wird optimal genutzt; jede einzelne Komponente des Designs erfüllt seinen Zweck

XI. Grafik: Wie hoch ist die Qualität und Auflösung der Grafiken von Knöpfen, Symbolen, Menüführung und Inhalten?

1. Grafiken wirken amateurhaft, schlechtes visuelles Design - völlig unproportional, stilistisch nicht einheitlich
2. Niedrige Qualität (z.B. Auflösung) der Grafiken. Niedrige Qualität des visuellen Designs - unproportional und stilistisch nicht einheitlich
3. Mittelmäßige Qualität der Grafiken und des visuellen Designs - überwiegend einheitlich im Design
4. Hohe Qualität der Grafiken und gute Auflösung des visuellen Designs - überwiegend proportional und einheitlich im Stil
5. Sehr hohe Qualität der Grafiken und sehr gute Auflösung des visuellen Designs - durchgehend proportional und einheitlich im Stil

XII. Visueller Anreiz: Wie gut sieht die App aus?

1. Kein visueller Anreiz - unangenehm zu betrachten, schwaches Design, beißende oder nicht zusammen passende Farben
2. Wenig visueller Anreiz - schlechtes Design, schlechter Farbgebrauch, visuell langweilig
3. Etwas visueller Anreiz - durchschnittlich weder angenehm noch unangenehm
4. Hoher visueller Anreiz - makellose Grafiken, einheitliches und professionelles Design
5. Wie in Punkt 4 beschrieben, darüber hinaus: Überaus großer visueller Anreiz - einprägsam, sticht positiv heraus; Farbgestaltung wirkt sich positiv auf die Optionen und die Menüführung der App aus

C: Ästhetik Mittelwert=_____

SECTION D

Information - Beinhaltet Informationen mit hoher Güte, Qualität (z. B. Texte, Feedback, Messinstrumente, Messergebnisse, Hinweise auf Literatur zum Vertiefen) aus einer glaubwürdigen Quelle. Bitte wählen Sie N/A (nicht anwendbar) aus, wenn der Punkt unzutreffend ist.

XIII. Genauigkeit der App-Beschreibung aus dem App-Store: Beinhaltet die App alle angegebenen Funktionen?

1. Irreführend - die App beinhaltet nicht die beschriebenen Komponenten/Funktionen oder hat keine Beschreibung
2. Überwiegend nicht korrekt - die App beinhaltet nur wenige der beschriebenen Komponenten/Funktionen
3. OK - die App beinhaltet einige der beschriebenen Komponenten, Funktionen.
4. Überwiegend korrekt - die App beinhaltet die meisten der beschriebenen Komponenten/Funktionen
5. Völlig korrekte Beschreibung der App sowie deren Komponenten/Funktionen

XIV. Ziele: Weist die App spezifische, messbare und erreichbare Ziele auf (spezifiziert in der Beschreibung im App Store oder in der App selbst)

N/A Die Beschreibung enthält keine Ziele, oder die Ziele der App sind nicht relevant für Ziele der Forschung

1 Die App kann die beschriebenen Ziele nicht erreichen

2 In der Beschreibung sind einige Ziele aufgelistet, aber die App kann diese nur mit einer geringen Wahrscheinlichkeit erreichen

3 OK. Die App hat klare Ziele, die erreicht werden könnten.

4 Die App hat klar spezifizierte Ziele, die messbar und erreichbar sind

5 Die App hat klar spezifizierte und messbare Ziele, die mit einer hohen Wahrscheinlichkeit erreicht werden können

XV. Qualität der Information: Ist der Inhalt korrekt, gut verfasst, relevant für das Ziel und das Thema der App?

N/A Die App enthält keine Informationen

1. Irrelevant, ungeeignet, unzusammenhängend, nicht korrekt
2. Kaum relevant, kaum geeignet, kaum zusammenhängend, möglicherweise nicht korrekt
3. Mittelmäßig relevant, mittelmäßig geeignet, mittelmäßig zusammenhängend, wirkt korrekt
4. Relevant, geeignet, zusammenhängend, korrekt
5. Sehr Relevant, sehr geeignet, sehr zusammenhängend, sehr korrekt

XVI. Quantität der Informationen: Ist der Inhalt der App umfassend und dennoch prägnant?

N/A: App beinhaltet keine Information

1. Minimal oder überfordernd
2. Unzureichend oder möglicherweise überfordernd
3. Ok, aber weder umfangreich noch prägnant
4. Bietet eine große Menge an Informationen, hat Lücken oder unnötige Details oder bietet keine weiteren Hinweise auf Literatur zum Vertiefen
5. Umfangreich und prägnant; beinhaltet Hinweise auf Literatur zum Vertiefen

XVII. Visuelle Informationen: Sind die visuellen Erklärungen von Konzepten – anhand von Tabellen, Graphiken, Bildern, Videos etc. – klar, logisch, korrekt?

N/A: App beinhaltet keine visuellen Information (beinhaltet z.B. nur Audio-Dateien oder Text)

1. Vollkommen unklar/verwirrend/falsch oder visuelle Informationen sind eigentlich nötig, fehlen aber
2. Überwiegend unklar/verwirrend/falsch
3. Ok, aber oft unklar/verwirrend/falsch
4. Meistens klar, logisch und richtig, mit unerheblichen/vernachlässigbaren Problemen
5. Perfekt klar, logisch und richtig

XVIII. Glaubwürdigkeit: Kommt die App aus einer legitimen Quelle (ist dies aus der Appstore-Beschreibung oder in der App selbst zu entnehmen)?

1. Die Quelle ist bekannt aber die Legitimität/Glaubwürdigkeit der Quelle ist fragwürdig (z.B. Werbe/Handelsbranche mit eigennützigem Interesse)
2. Scheint von einer legitimen Quelle zu kommen, kann aber nicht nachgeprüft werden (z.B. keine Webseite)
3. Entwickelt von einer kleinen gemeinnützigen Organisation (NGO: z. B. Krankenhaus, Zentrum etc.) oder von einer spezialisierten wirtschaftlichen Firma oder von Fördermittelgeber
4. Von der Regierung, einer Universität oder einer ähnlichen Institution entwickelt
5. Entwickelt mit Hilfe von wettbewerbsorientierten Regierungs- oder Forschungsgeldern (z.B. DFG, BMBF)

XIX. Evidenzbasierung: Wurde die App geprüft/getestet (muss nachweisbar sein z. B. durch wissenschaftliche Publikation)?

N/A: App wurde nicht geprüft/getestet

1. Die Evidenz zeigt auf, dass die App nicht funktioniert
2. Die App wurde getestet (z.B. Akzeptanz, Benutzerfreundlichkeit, Zufriedenheitsratings) und zeigt teilweise positive Ergebnisse in nicht randomisiert-kontrollierten Studien oder es gibt wenig bzw. keine widersprüchliche Evidenz.

3. Die App wurde getestet (z.B. Akzeptanz, Benutzerfreundlichkeit, Zufriedenheitsratings) und zeigt positive Ergebnisse in nicht-randomisierten Studien, es besteht keine widersprüchliche Evidenz
4. App wurde getestet und die Resultate aus 1-2 randomisiert-kontrollierten Studien weisen auf positive Ergebnisse hin
5. App wurde getestet und die Resultate aus ≥ 3 hochqualitativen randomisiert-kontrollierten Studien weisen auf positive Ergebnisse hin

D. Information Mittelwert=_____*

* Fragen die mit „Nicht verfügbar / keine Antwort“ geratet wurden, nicht in die Mittelwert-Berechnung miteinbeziehen.

SECTION PT

Psychotherapie – Diese Sektion erfasst Gütekriterien von Apps in Hinblick auf Patientensicherheit, Sinnhaftigkeit der Darbietung als App sowie die Güte der therapeutischen Angebote.

XX. Gewinn für Patienten

1. Die App kann den Patienten bei der Behandlung seiner Symptome/Erkrankung sehr schlecht unterstützen.
2. Die App kann den Patienten bei der Behandlung seiner Symptome/Erkrankung schlecht unterstützen.
3. Die App kann den Patienten bei der Behandlung seiner Symptome/Erkrankung teilweise unterstützen.
4. Die App kann den Patienten bei der Behandlung seiner Symptome/Erkrankung gut unterstützen.
5. Die App kann den Patienten bei der Behandlung seiner Symptome/Erkrankung sehr gut unterstützen.

XXI. Gewinn für Therapeuten

* N/A: Nur bei blended-care Ansätzen anwendbar

1. Die App bietet nichts, was der Therapeut zur Optimierung der Therapie nutzen kann.
2. Die App bietet wenig, was der Therapeut zur Optimierung der Therapie nutzen kann.
3. Die App bietet einiges, was der Therapeut zur Optimierung der Therapie nutzen kann.
4. Die App bietet viel, was der Therapeut zur Optimierung der Therapie nutzen kann.
5. Die App bietet sehr viel, was der Therapeut zur Optimierung der Therapie nutzen kann.

XXII. Mögliche Risiken, Nebenwirkungen und schädliche Effekte

[1] Von der Benutzung von Apps die hier einen Wert unter 3 erreichen wird dringend abgeraten.

1. Die App ist aus therapeutischer Sicht schädlich. Es besteht ein hohes Risiko für unerwünschte Wirkungen (z.B.: irreführende Ratschläge, schädigende Übungen, unterlassene Hilfe, etc.).
2. Die App ist aus therapeutischer Sicht fragwürdig. Es besteht ein gewisses Risiko für unerwünschte Wirkungen (z.B.: falsche Rückmeldungen zu Fragebogenwerten, unkorrekte Informationen und Empfehlungen, etc.).
3. Die App ist aus therapeutischer Sicht teilweise sicher. In Hinblick auf best practice Richtlinien zur Behandlung der jeweiligen Erkrankung bestehen Mängel (z.B.: keine Notfallfunktion, etc.).
4. Die App ist aus therapeutischer Sicht vorwiegend sicher. In Hinblick auf best practice Richtlinien zur Behandlung der jeweiligen Erkrankung bestehen nur minimale Mängel.

5. Die App ist aus therapeutischer Sicht sicher und entspricht den aktuellen best practice Richtlinien der jeweiligen Erkrankung.

XXIII. Übertragbarkeit in die Routineversorgung

1. Die Evidenz zeigt auf, dass die App nicht funktioniert bzw. es ist keine Information auffindbar.
2. die App wurde nicht an Patienten getestet (z. B. Probanden mit erhöhten Werten psychischer Symptomatik in Fragebögen wurden untersucht ohne dass psychische Störungen diagnostiziert wurden) sowie nur unter Bedingungen, die nicht repräsentativ für die psychotherapeutische Routine sind (z. B. von Studenten in universitären Settings)
3. die App wurde an Patienten getestet (z. B. Diagnose mittels SKID ermittelt oder Personen in psychotherapeutischer Behandlung), aber nur unter Bedingungen, die nicht repräsentativ für die psychotherapeutische Routine sind (z. B. nur in universitären Settings getestet)
4. die App wurde an Patienten getestet (z. B. Diagnose mittels SKID ermittelt oder Personen in psychotherapeutischer Behandlung) und die Bedingungen waren etwas repräsentativ für die psychotherapeutische Routine (z. B. nur in einer Klinik, nur in einer Psychotherapieambulanz)
5. die App wurde an Patienten getestet (z. B. Diagnose mittels SKID ermittelt oder Personen in psychotherapeutischer Behandlung) und die Bedingungen waren sehr repräsentativ für die psychotherapeutische Routine (z. B. in mehreren Kliniken, mehreren Psychotherapieambulanzen)

PT. Psychotherapie Mittelwert=_____*

*Fragen die mit „Nicht verfügbar / keine Antwort“ geratet wurden, nicht in die Mittelwert-Berechnung miteinbeziehen.

[1] Von der Benutzung von Apps die hier einen Wert unter 3 erreichen wird dringend abgeraten.

Ich hatte noch ein paar Gespräche und da kamen noch ein paar Sachen raus, die ich wenigstens in die Diskussion einwerfen möchte. Das heißt jetzt nicht, dass ich die alle berücksichtigt sehen muss, aber ich fand es waren vielleicht schon Überlegungen:

1.) Sollten wir eine Sektion mit „Red Flags“ machen: Also vielleicht Sachen, die zusätzlich zu der Mindestpunktzahl der Frage XXII. in die Frage eingehen, „Raten wir ab?“ Dazu könnte gehören:
    - die App (besonders bei alleinstehenden Apps) macht unseriöse Versprechungen (Heilung / mit der App werden sie ihre Symptome in xx Tagen los)
    - sie macht kostenpflichtige (über die App-Kosten hinaus) Angebote
    - sie beachtet Vorgaben der Berufsordnung nicht. Falls es Kontaktpersonen gibt: unseriöse Einflussnahme (Produkte, Weltanschauungen, Services, kostenpflichtige Angebote), unseriöser persönlicher Kontakt, Annahme von Geschenken o. Dienstleistungen
    - fragliche Qualifikation der Kontaktpersonen

Eine grundlegende Frage, die noch aufkam war, ob wir die Indikationsstellung bewerten sollten: Unternimmt die App (bei standalone Apps) Schritte, um sicherzustellen, dass es sich um die richtige App für die Störung handelt? Ist die App überhaupt das Richtige für den Patienten? Wird darauf hingewiesen, wann der Behandlungsbedarf über eine Appnutzung hinausgeht und die Person weitergehende Hilfe benötigt?

SECTION E (Subjektive Qualität)

XXIV. Würden Sie die App Personen empfehlen die davon profitieren könnten?

1. Überhaupt nicht: Ich würde die App niemanden empfehlen
2. Ich würde die App nur wenigen Personen empfehlen
3. Vielleicht: Ich würde die App schon einigen Personen empfehlen
4. Ich würde die App vielen Personen empfehlen
5. Auf jeden Fall: Ich würde die App jedem empfehlen

XXV. Wie viele Male würden Sie die App in den nächsten zwölf Monaten nutzen, falls diese relevant für Sie wäre?

1. Nie
2. 1-2
3. 3-10
4. 10-50
5. >50

XXVI. Würden Sie diese App erwerben, wenn sie kostenpflichtig wäre?

1. Nein
2. Vielleicht
3. Ja

XXVII. Wie viele Sterne würden Sie für die App insgesamt vergeben?

1. ★ Eine der schlechtesten Apps, die ich je genutzt habe
2. ★★
3. ★★★ Durchschnittlich
4. ★★★★
5. ★★★★★ Eine der besten Apps, die ich je genutzt habe

Bewertung

Bewertung der App-Qualität für Sektion

A. Engagement Mittelwert = _____

B. Funktionalität Mittelwert = _____

C. Ästhetik Mittelwert = _____

D. Information Mittelwert = _____

PT: Psychotherapie Mittelwert= _____

Mittelwert der App-Qualität = _____

Subjektive App- Qualität = _____

SECTION F

Diese zusätzlichen Angaben erfassen die wahrgenommen Auswirkungen der App-Nutzung auf das Wissen, die Einstellung und Veränderungsintention sowie die Wahrscheinlichkeit einer tatsächlichen Veränderung des Verhaltens hin zum gewünschten Gesundheitsverhalten.

1. Bewusstsein: Die App steigert das Bewusstsein für die Wichtigkeit, sich mit dem Gesundheitsverhalten zu beschäftigen

Stimme überhaupt nicht zu Stimme sehr zu

1 2 3 4 5

2. Wissen: Die App erhöht das Verständnis und Wissen über das gewünschte Gesundheitsverhalten

Stimme überhaupt nicht zu Stimme sehr zu

1 2 3 4 5

3. Einstellung: Die App ändert die Einstellung, das Gesundheitsverhalten zu verbessern

Stimme überhaupt nicht zu Stimme sehr zu

1 2 3 4 5

4. Absicht zur Verhaltensänderung: Die App fördert die Absicht und Motivation, das Gesundheitsverhalten zu verbessern.

Stimme überhaupt nicht zu Stimme sehr zu

1 2 3 4 5

5. Suche nach Hilfe: Die App ermutigt dazu, sich weitere Hilfen bezüglich des Gesundheitsverhaltens (wenn erforderlich) zu suchen

Stimme überhaupt nicht zu Stimme sehr zu

1 2 3 4 5

6. Verhaltensänderung: Die App ermöglicht die gewünschte Änderung des Gesundheitsverhaltens

Stimme überhaupt nicht zu Stimme sehr zu

1 2 3 4 5

Name: Philipp Dörzenbach Matrikelnummer: 849794

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Philipp Dörzenbach