Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 248

Privatheit im Gesundheitsspiel Candy Castle

Corinna Giebler

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Bernhard Mitschang
Betreuer/in:	Dipl.-Inf. Christoph Stach
Beginn am:	13. Juli 2015
Beendet am:	12. Januar 2016
CR-Nummer:	J.3, K.4.1, K8

Kurzfassung

Die zunehmende Verwendung elektronischer Endgeräte stellt die heutige Gesellschaft vor das große Problem der Datenprivatheit. Die digitalen Helfer verwalten verschiedenste persönliche Daten, auf die wiederum eine Vielzahl von Anwendungen Zugriff hat. Selten ist es dem Benutzer möglich, direkten Einfluss oder auch nur Einblick in die Verwendung seiner Daten zu erlangen.

Besonders in der Kritik stehen hierbei mobile Plattformen, wie Smartphones oder Tablets. Die Applikationen, oder kurz Apps, für diese Plattformen verfügen nur selten über Datenschutzerklärungen oder Maßnahmen zum Datenschutz. Auch ist es auf bestimmten Betriebssystemen nicht möglich, die Berechtigungen einer App einzuschränken.

Dabei werden gerade diese mobilen Endgeräte mehr und mehr zur Verwaltung sensibler Daten verwendet. Hierzu gehören inzwischen auch medizinische Daten, die mithilfe diverser Gesundheits-Apps aufgezeichnet und verarbeitet werden können. Da es sich bei diesen medizinischen Werten um sehr private Daten handelt, sind diese besonders schützenswert. Zudem sollte der Nutzer einer solchen Gesundheits-App stets wissen, was mit seinen Daten geschieht und die Kontrolle über ihre Verwendung innehaben.

In dieser Arbeit soll darum erörtert werden, wie sich die persönlichen Daten mithilfe eines Berechtigungssystems schützen lassen. Zunächst wird hierfür ein Anforderungskatalog ermittelt, der neben Anforderungen an Funktionalität und Bedienbarkeit auch Forderungen an Privatheit und Nutzerbestimmung stellt. Mithilfe dieses Katalogs wird anschließend ein Gesundheitsspiel für Kinder mit Diabetes weiterentwickelt und um Privatheitsaspekte erweitert. So entsteht ein Prototyp für ein Spiel, dessen Privatheitsgrad vom Benutzer einstellbar ist. Zuletzt wird gezeigt werden, dass die zuvor gestellten Anforderungen erfüllt sind und dass verarbeitete Daten auch vor unerlaubtem Zugriff geschützt werden können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

1.1 Hintergrund

Seit es Smartphones gibt, ist die Nutzung von Taschenkalendern und Kameras mehr eine Frage der Vorliebe als eine der Notwendigkeit. Mit den Handys der Neuzeit ist es kein Problem mehr, schnell die nächsten Besprechungstermine zu überprüfen oder ein schönes Foto von sich und seinen Freunden zu schießen.

Neben herkömmlichen Funktionen, wie Telefonieren oder Nachrichten verschicken, kann das Smartphone noch zahllose andere Aufgaben erfüllen. Mit verschiedensten Anwendungen lässt sich das eigene Gerät nach Belieben personalisieren. Spiele, Bücher, soziale Netzwerke oder auch digitale Terminverwaltung liegen nur einen Tipp mit dem Finger entfernt. Auf mobilen Geräten heißen diese Anwendungen „Applikationen“ oder kurz „Apps“.

Durch die Verwendung dieser Apps können Smartphones und Tablets als mobile Computer benutzt werden, die jederzeit griffbereit sind. Daten können erfasst, gespeichert und verarbeitet werden und verschiedene Geräte lassen sich mit wenig Aufwand synchronisieren. Wie von M. Weiser bereits 1991 [Wei99] dargelegt, sind durch die Smartphones Computer Teil des modernen Alltags geworden, oft ohne als solche wahrgenommen zu werden.

Auch Gesundheits-Apps sind inzwischen auf dem Smartphone oder Tablet vertreten. Neben Fitnesstipps fürs Büro (z.B. *Office-Workout* [rem13]) oder Ernährungsratgebern (z.B. *Kalorienzähler* [MyF15]) kann das Smartphone beispielsweise auch die Umgebungslautstärke messen, um Gehörschäden vorzubeugen (z.B. *LärmApp* [Mon12]). Sogar Impfungen, Arzttermine und ganze Krankengeschichten lassen sich bequem per App verwalten und organisieren (z.B. *Gesundheitsbuch* [Pue15]).

Eine Studie des Universitätsklinikums Freiburg im Auftrag der Techniker Krankenkasse (siehe [Hüb15] und [LBBK15]) verweist auf die Potentiale, die Krankenkassen und Entwickler in der mobilen Gesundheitsbetrachtung sehen. Die sogenannten mobile Health-Apps (kurz *mHealth-Apps*) [Mat15], d.h. Apps, die Funktionen zur selbstständigen Gesundheitsfürsorge anbieten, könnten laut der Studie dabei helfen, den Patienten mehr Eigenverantwortung zu übertragen. Dadurch würden Ärzte entlastet und Patienten selbstständiger werden. Zudem können durch die Nutzung von mHealth die Kosten für Gesundheitsversorgung gesenkt werden [SRD+15].

Auch Kinder können von mHealth profitieren. Mithilfe diverser Apps können sie ebenfalls mehr über ihren Körper erfahren und sich aktiv mit dem Thema „Gesundheit“ auseinandersetzen. Dazu gehören auch Apps, die Kinder mit chronischen Krankheiten, z.B. Asthma oder Diabetes, im alltäglichen Leben unterstützen wollen (siehe z.B. [HJAA10]). Hierzu sind Benutzereingaben wie Blutzuckerwerte oder Luftvolumen unerlässlich. Auch Fotografien und Visualisierungen können bei der Therapie chronischer Krankheiten Einsatz finden, wie J. Frost und B. Smith in ihrer Arbeit diskutierten [FS03].

Doch diese Art der Gesundheitsfürsorge birgt Gefahren. Denn im Gegensatz zu einem analogen Taschenkalender ist das Smartphone mit verschiedenen Netzen verbunden, wie beispielsweise Onlinedienste und Telefonnetze. Über diese können Schadprogramme auf das Gerät gelangen, welche Daten verfälschen oder gar an Dritte weiterleiten (vgl. [Sop13]). Ebenso stellen physische Angriffe ein Problem dar (vgl. [MBK+13]). Hierbei handelt es sich um Angriffe durch Dritte, die durch physischen Besitz des mobilen Gerätes erfolgen. Ein Beispiel für einen solchen Angriff wäre ein Bekannter des Nutzers, der das Smartphone in seinen Besitz bringt und so Zugriff auf sensible Daten erhält.

Eine Gefahr der Verletzung der Privatsphäre geht jedoch nicht nur von Außenstehenden aus. Neben Angreifern, die sich Schwachstellen im System zu Nutze machen, kann auch die App selbst dem Nutzer Schaden zufügen. So kann es sein, dass eine App die verwendeten Daten unbemerkt an externe Stellen weiterleitet, ohne dass dies vom Benutzer gewünscht ist. Ein Szenario hierfür wäre eine App, mithilfe derer der Nutzer Gewicht und Aktivitätslevel überwachen kann. Diese Daten werden von der App gesammelt und ohne Wissen des Nutzers an beispielsweise seine Krankenkasse übermittelt.

Ein solches Szenario ist durchaus realistisch, wie die Arbeit von W. Enck et al. [EGC+10] zeigt. Demnach sind in 30 getesteten Apps 68 Fälle potentiellen Missbrauchs von Daten in 20 verschiedenen Apps festgestellt worden. Besonders problematisch ist dies, wenn die App aus einer vertrauenswürdigen Quelle stammt, z.B. von der Krankenkasse selbst veröffentlicht wurde, sodass der Nutzer keinen Verdacht hegt.

Doch auch, wenn der Entwickler der App nicht böswillig handelt, kann es trotzdem passieren, dass die Privatsphäre des Nutzers verletzt wird. Beispielsweise möchte ein Nutzer nicht, dass eine App Zugriff auf seine Standortdaten erhält. Die App wiederum nutzt diese Daten, um bestimmte Funktionen anbieten zu können, die vom Nutzer nicht gewünscht sind. Die Rechte, die eine App erhält, werden meist nicht direkt vom Nutzer, sondern vom Berechtigungssystems des Betriebssystems festgelegt [BO10]. Dieses erlaubt Zugriffe auf externe Funktionen dann, wenn sie zuvor beim Betriebssystem registriert wurden. Somit hat der Nutzer bei den heute vorherrschenden mobilen Betriebssystemen keine Möglichkeit, die Rechte der App einzuschränken, ohne ganz auf sie verzichten zu müssen. Eine weitere Auseinandersetzung mit Sicherheit in Smartphone-Betriebssystemen bietet der Artikel „Next Generation Mobile Application Security“ [PS11].

Hinzu kommt das Problem der Überprivilegierung. Dabei erhalten Apps Berechtigungen, die diese gar nicht oder in anderer Art und Weise benötigen. In ihrer Arbeit „Android permissions demystified“ [FCH+11] setzen sich A. Porter Felt et al. mit dieser Überprivilegierung auseinander.

Gerade Kindern fällt es oftmals schwer, den Begriff „Privatheit“ zu begreifen. Vor allem, wenn es um sensible Daten geht, möchten Eltern diese geschützt wissen. Es besteht also der Bedarf an einer Gesundheits-App, die für Kinder geeignet ist und den Nutzer bzw. den Erziehungsberechtigten über die Verwendung sensibler Daten entscheiden lässt.

1.2 Aufgabenbeschreibung

In dieser Arbeit soll eine mHealth-App entwickelt werden, deren Zugriffsrechte der Nutzer selbst festlegen kann. Bei dieser beispielhaften App handelt es sich um das Diabetesspiel *Candy Castle*, das erstmals von Martin Knöll [Knö10] vorgestellt wurde. Mithilfe dieser App sollen diabeteskranke Kinder lernen, sich auf den neuen Lebensstil einzustellen. Dabei sollen sie verschiedene Daten ihres Alltags sammeln, wie z.B. Blutzuckermesswerte. In Verbindung mit GPS-Werten beeinflussen diese Daten die Landschaft eines virtuellen Königreichs. Aufgabe dieser Bachelorarbeit ist es, das durch C. Stach und L. Schlindwein verfeinerte Konzept von Candy Castle [SS12] um Privatheitsaspekte zu erweitern.

Dazu muss zunächst festgelegt werden, welche Anforderungen die App zu erfüllen hat, sowohl funktionell als auch im Bereich Privatheit. Mithilfe dieser Anforderungen wird auf Basis der alten Entwürfe ein neues Konzept angefertigt, das die Funktionen Candy Castles anpasst und gegebenenfalls erweitert.

Um hierbei die Privatheit miteinbeziehen zu können, ist es notwendig, die anfallenden Daten und Datenquellen zu erfassen, um diese entsprechend absichern zu können. Hierfür werden die einzelnen unabhängigen Grundfunktionen der App auf sogenannte Service Features abgebildet. Diese Service Features können dann mithilfe der Privacy Management Platform (kurz *PMP*, [SM14, Sta13b, SM13, Sta13a, Sta15]) vom Nutzer beliebig aktiviert oder deaktiviert werden.

Ziel ist es, Candy Castle mit erweiterten Funktionen zu reimplementieren, sodass der Nutzer selbst über die Berechtigungen der App entscheiden kann.

1.3 Gliederung

Diese Arbeit ist wie folgt unterteilt:

Kapitel 2 – Verwandte Arbeiten: Dieses Kapitel dient dazu, einen Überblick über bereits vorhandene verwandte Arbeiten zu gewinnen. Hierzu gehören neben Arbeiten zu „Privatheit in Gesundheits-Apps“ auch eine Übersicht über existierende Apps, bei denen medizinische Daten im Vordergrund stehen. Diese Betrachtung soll zeigen, wie andere ähnliche Apps mit der Frage der Privatheit umgehen und welche Lösungsansätze bereits existieren. Zudem sollen diese Lösungen kritisch hinterfragt werden. Darüber hinaus werden unterschiedliche Anforderungen an mHealth-Apps betrachtet, um diese als Basis für den Anforderungskatalog in Kapitel 4 zu verwenden.

Kapitel 3 – Candy Castle: Bevor ein neues Konzept für Candy Castle entworfen werden kann, werden in diesem Kapitel zwei Vorgänger des Spiels vorgestellt. Bei diesen handelt es sich um die Grundidee von Martin Knöll, sowie um die Verfeinerung durch Christoph Stach und Luiz F. M. Schlindwein.

Kapitel 4 – Erstellung eines Anforderungskataloges: In diesem Kapitel werden Forderungen an das neue Konzept von Candy Castle gestellt. Die in Kapitel 2 betrachteten Anforderungskataloge werden als Basis verwendet und erweitert. Dabei geht es nicht nur um funktionale Anforderungen, sondern auch um Forderungen im Bereich von Datensicherheit und Privatheit.

Kapitel 5 – Technische Grundlagen für die Reimplementierung: Dieses Kapitel befasst sich mit der Technik, mit deren Hilfe die Reimplementierung Candy Castles durchgeführt wird. Neben dem Betriebssystem wird auch das System zur Verwaltung von Privatheitseinstellungen, die PMP (Privacy Management Platform), vorgestellt.

Kapitel 6 – Das neue Konzept für Candy Castle: Mithilfe der in Kapitel 4 gesammelten Anforderungen wird hier ein neues Konzept für Candy Castle entworfen. Anschließend werden in diesem Kapitel die einzelnen Funktionen auf die anfallenden Daten analysiert. Zudem werden externe Services vorgestellt, welche Candy Castle in seiner Funktion unterstützen.

Kapitel 7 – Realisierung: Nachdem Konzept und Technik vorgestellt wurden, handelt dieses Kapitel von der eigentlichen Implementierung. Hierzu wird erläutert, wie die verschiedenen Privatheitsaspekte auf die PMP übertragen werden. Ebenso wird die genaue Umsetzung der Privatheit und der App beschrieben. Zuletzt wird die Kommunikation mit den externen Services näher betrachtet.

Kapitel 8 – Evaluation: In diesem Kapitel wird das neue Konzept der App Candy Castle gegen die Anforderungen aus Kapitel 3 evaluiert. Zudem soll es mit bereits bestehenden Konzepten verglichen werden, die in Kapitel Kapitel 2 vorgestellt wurden.

Kapitel 9 – Zusammenfassung und Ausblick: Als Abschluss dieser Arbeit wird diese kurz zusammengefasst. Zudem werden die möglichen zukünftigen Entwicklungen im Bereich „Privatheit in Gesundheits-Apps“ thematisiert.

2 Verwandte Arbeiten

Die größte Neuerung, die Candy Castle durch die Reimplementierung erfährt, ist die Erweiterung um Privatheitsaspekte. Hierbei ist es wichtig, eine sinnvolle und effiziente Möglichkeit zu finden, diese umzusetzen.

Es existieren bereits verschiedene mHealth-Apps, die den Nutzer bei der Gesundheitsförderung unterstützen sollen. Nicht verwunderlich ist darum, dass es schon einige Arbeiten gibt, die sich mit den Themen Privatheit und Datensicherheit in mHealth-Apps beschäftigen. Diese sollen im Folgenden näher betrachtet und auf Lösungsvorschläge untersucht werden. Auch sollen anhand existierender Kriterien die Probleme dieser Lösungen herausgestellt werden.

2.1 Privatheit und Gesundheits-Apps

„Privatheit wird in der deutschen Bevölkerung als wichtiges und schützenswertes Gut angesehen“ [TM14]. So heißt es in einer Studie der Uni Hohenheim zum Privatheitsbewusstsein der deutschen Bevölkerung. Hierbei wurden über 3.000 Personen ab 16 Jahren nach ihren Ansichten zur Privatheit in der heutigen Zeit befragt. Signifikant ist hierbei, dass die Frage, wer Einsicht in private Daten erhält (informationale Privatheit) mit 4,22 von 5 Punkten für die Befragten die größte Rolle spielt. Selbst die physische Privatheit, die sich auf die räumliche Distanz zwischen zwei Personen bezieht, liegt mit 3,43 Punkten auf dem zweiten Platz, und wird somit geringer gewichtet. Nutzer möchten sich sicher sein können, dass ihre Daten ihnen allein gehören und nicht von außen eingesehen werden können. Dies zeigt deutlich, wie wichtig es ist, sensible Daten vor Weitergabe oder Einsicht durch Dritte zu schützen. Gerade wenn es um Apps geht, die mit dem Internet oder einem anderen Netz in Verbindung stehen, also online agieren, haben laut dieser Studie viele Nutzer Bedenken. So sind 44% der Befragten besorgt, beziehungsweise sehr besorgt, dass ihre GPS-Daten von Mobilanbietern gespeichert und eingesehen werden.

In vielen Fällen werden Apps jedoch trotz vieler Zugriffsforderungen heruntergeladen und installiert. So wurde beispielsweise die App *Runtastic Laufen & Fitness* [Run15] über zehn Millionen Mal via Google Play heruntergeladen. Mit 4,5 von 5 Sternen verfügt sie über eine gute Bewertung. Die Liste der geforderten Zugriffsrechte ist allerdings lang. Neben In-App-Käufen und Zugriff auf Standortdaten fordert die App auch Einsicht in den Geräte-

und App-Verlauf, sowie in Fotos und eigene Dateien. Auch Kamera- und Mikrofon-Nutzung müssen vor der Installation akzeptiert werden.

Doch warum wird diese App verwendet, obwohl Privatheit für die Nutzer eine so große Rolle spielt? Eine Erklärung ist sicherlich Unachtsamkeit. A. Felt et al. fanden in einer Studie [FHE+12] heraus, dass oftmals die Berechtigungsanfragen ignoriert oder nicht verstanden werden. Doch selbst, wenn die geforderten Rechte durch den Nutzer gewissenhaft analysiert werden, können die genehmigten Rechte zweckentfremdet werden (vgl. [FEW12]). Deutlich wird hierdurch, dass ein scheinbar harmloses Zugriffsrecht große Probleme verursachen kann. So kann beispielsweise eine App mit Zugriff auf die Kontaktliste diese nach Belieben verändern und sogar Kontakte hinzufügen.

Ein anderes Problem ist die Möglichkeit der Einflussnahme. So kann beispielsweise unter Android bis Version 6.0 lediglich die Gesamtheit der Anforderungen genehmigt werden. Tut man dies nicht, muss man gänzlich auf die App verzichten.

Dabei ist die Menge der zu genehmigenden Zugriffsrechte oftmals größer als benötigt (vgl. [FCH+11]). Sie sind damit über privilegiert und verfügen so über einen Spielraum, der einzudämmen wäre.

Dem „Alles oder nichts“-Ansatz versuchen Apps wie *AppGuard* [BGH+14] entgegenzuwirken, indem sie das Berechtigungsmanagement dem Nutzer zu übertragen versuchen. Im Falle von AppGuard geschieht dies dadurch, dass der Code von Apps überwacht und modifiziert wird, um sicherheitstechnisch bedenkliche Funktionsaufrufe zu verhindern. Problematisch hierbei ist jedoch, dass AppGuard Zugriff auf den Bytecode einer App erhält und diesen beliebig modifizieren kann. Gerade in Fällen wie mHealth, in denen viele sensible Daten verwaltet werden, ist dies kritisch zu betrachten, da auch AppGuard von Außen kompromittiert werden könnte.

Obwohl das Berechtigungssystems vor allem auf Android einige Kritikpunkte aufwirft, sollten die Vorteile von Apps nicht vernachlässigt werden. So untersucht z.B. J. Wiemeyer [Wie10] die Wirksamkeit unterschiedlicher ernsthafter Spiele, die sich mit gesundheitlichen Themen auseinandersetzen. Diesen sogenannten *Serious Games* schreibt er ein hohes Potential zu. Therapien verschiedener Erkrankungen, wie beispielsweise Krebs oder Asthma, würden von der Verwendung von Serious Games profitieren.

Auch die bereits erwähnte Studie im Auftrag der Techniker Krankenkasse sieht laut der Ärzte Zeitung große Vorteile in der Nutzung von mHealth-Apps [Hüb15]. Durch selbstständige Messung der Gesundheitswerte könne die Selbstständigkeit und Eigenverantwortung der Betroffenen erhöht werden. Dies würde vielen Ärzten ihre Arbeit erleichtern und Patienten während der Therapie unterstützen.

Weitere Vorteile werden von P. Klasnja und W. Pratt dargestellt [KP12]. Besonders das Smartphone wird hier als Plattform besprochen. Dieses eigne sich besonders gut als Plattform für mHealth-Apps, da es weit verbreitet und technisch ausgereift sei. Es soll dabei helfen

können, neue Verhaltensmuster zu erlernen und diese aktiv im Alltag umzusetzen. Dazu können beispielsweise die Sensoren des Smartphones verwendet werden, um ein digitales Tagebuch zu ermöglichen.

2.2 Betrachtung der Privatheit in einzelnen Apps

Obwohl ein Großteil der mHealth-Apps an Erwachsene gerichtet ist, gibt es auch für Kinder verschiedenste Helfer für den Alltag.

Eine Liste mit Kurzbeschreibungen dieser Apps findet man beispielsweise auf der Webseite des Mobile Health Competence Centre [Mob15]. Hierbei können zwei grundlegende Arten von mHealth-Spielen unterschieden werden, die im Folgenden näher betrachtet werden:

Informierende Spiele (Abschnitt 2.2.1): Als informierend sollen im Folgenden solche Spiele bezeichnet werden, deren einzige Aufgabe es ist, den Nutzer über ein bestimmtes Thema aufzuklären. Sie verlangen keine Eingaben von persönlichen Daten oder Messwerten.

Verarbeitende Spiele (Abschnitt 2.2.2): Diese Art der Spiele soll im Folgenden die Serious Games bezeichnen, die Nutzereingaben erfordern und diese verarbeiten. Um durch das Spiel einen Mehrwert zu erreichen, müssen persönliche medizinische Daten in die App eingetragen werden.

Der Hauptunterschied liegt hierbei in den angebotenen Funktionen. Während informierende Spiele sehr allgemein gehalten sind, können verarbeitende Spiele auf den Nutzer angepasst werden. Zwar erreichen Spiele der zweiten Gruppe so einen höheren Mehrwert, allerdings fallen durch das Personalisieren der App Daten an, die geschützt werden müssen.

2.2.1 Informierende Spiele

Repräsentativ für die informierenden Spiele sollen hier zwei Beispiele genannt werden.

Bei der ersten App handelt es sich um das Spiel *Kids Beating Asthma* von Media Net Software, S.A. (siehe Abb. 2.1) für iOS und Android [Med13]. Ziel des Spiels ist es, Kindern und Teenagern Wissen zum Thema Asthma zu vermitteln. Hierfür bietet die App acht verschiedene Minispiele an, die den Nutzern helfen sollen, ihre Krankheit zu verstehen. Wie in Abb. 2.1 dargestellt, kann innerhalb der App zwischen zwei Altersgruppen gewählt werden, wobei die Version für Teenager zum Zeitpunkt dieser Arbeit noch nicht vorhanden ist. Neben Puzzeln (ganz rechte Abbildung) bietet die App auch Quizfragen (mittlere Abbildung), wodurch der Nutzer lernt, seine körperliche Verfassung selbstständig einzuschätzen. Die App benötigt hierfür keine Eingaben der Nutzer, wodurch keine zu schützenden Daten anfallen. Allerdings

Abbildung 2.1: Screenshots Kids Beating Asthma [Med13]

fordert die App Zugriff auf Fotos, Medien und eigene Dateien. Über die Verwendung dieser Daten gibt die App keine Auskunft.

Die zweite App stammt vom Entwickler Psyop Games und ist für iOS, Android und Kindle Fire erhältlich. In *Nightmare: Malaria* [Psy13] (siehe Abb. 2.2) schlüpft der Spieler in die Rolle eines Mädchens, das an Malaria erkrankt ist. In mehreren Leveln flieht der Nutzer vor Moskitos durch Fieberträume und lernt dabei spielerisch, wie man sich vor den gefährlichen Insektenstichen schützen kann. Abb. 2.2 zeigt Ausschnitte aus dem Spiel, in dem der Spieler in Blut und Gehirn gegen die Krankheit ankämpft. Dabei wird er von Stechmücken attackiert, vor welchen er sich in Moskitonetzen verstecken kann. Nightmare: Malaria benötigt lediglich Zugriffsrechte auf Netzwerkinformationen sowie auf die verschiedenen Energiespar-Mechanismen, um diese auszuschalten [Ama13].

2.2.2 Verarbeitende Spiele

Wie Kids Beating Asthma ist auch *Wind Runners* [NPS+12] (siehe Abb. 2.3) für asthmakranke Kinder entwickelt worden. Ziel des Spiels ist es, einen Ball durch verschiedene Parcours zu leiten. Hindernisse, die dem Spieler den Weg versperren, können mithilfe eines Lufttanks überwunden werden. In Abb. 2.3 sind zwei dieser Parcours dargestellt. Dieser Tank wird mithilfe der Luftvolumenmessung am Anfang des Spiels gefüllt. Die Messung erfolgt dabei durch ein via Bluetooth verbundenes Spirometer. Obwohl durch die App persönliche medizinische Daten verwaltet werden, wird der Privatheitsaspekt im Konzept nicht berücksichtigt.

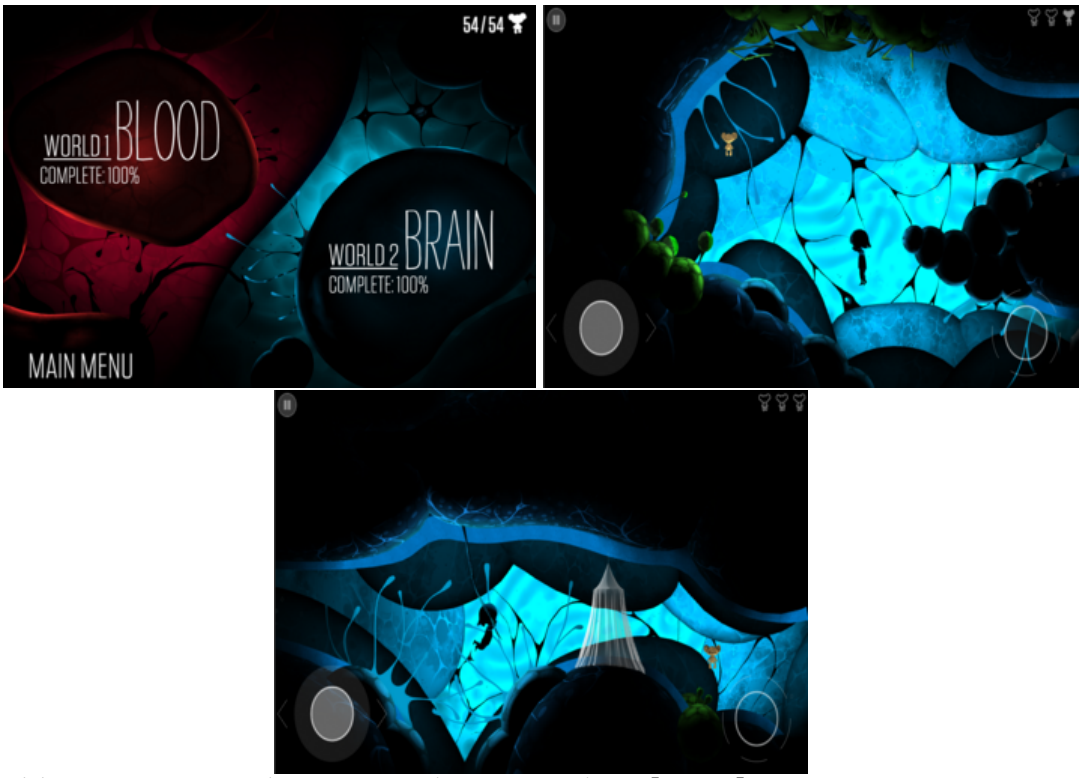

Abbildung 2.2: Screenshots von Nightmare: Malaria [Psy13]

Abbildung 2.3: Screenshots von WindRunners [NPS+12]

Bei *mySugr Junior* [myS14a] (siehe Abb. 2.4) handelt es sich um ein Diabetes-Tagebuch für Kinder. Damit stimmen die Aufgaben mySugr Junior und Candy Castle stark überein. Für jeden Eintrag, den der Nutzer in mySugr Junior tätigt, erhält er Punkte. Bei diesen Einträgen handelt es sich um Blutzuckerwerte, Insulindosen und Nahrungsmittel. Die verschiedenen Eingabemasken sind in Abb. 2.4 oben rechts und unten links dargestellt. Dabei werden die Kinder durch ein kleines Monster unterstützt, das sie motiviert. Jeder Eintrag kann per Mail oder SMS an eine Kontaktperson gesendet werden (in der Abbildung unten rechts). So kann der Nutzer jederzeit die Eltern oder andere Erwachsene um Hilfe bitten. Die Pro-Version [myS14b] bietet darüber hinaus weitere Möglichkeiten, wie die Berechnung der Insulindosis oder die Einbindung eines Bluetooth-Messgeräts. Um mySugr Junior nutzen zu können, ist eine Registrierung notwendig. Diese erfordert neben dem Namen auch eine Mailadresse und die Telefonnummer der Kontaktperson. Auch müssen in diesem Zuge die allgemeinen Nutzungsbedingungen akzeptiert werden. Hierin wird klargestellt, welche Daten die App verwendet. Die Daten werden vor allem auf dem Endgerät gespeichert. Dennoch gibt mySugr GmbH an, dass die Daten bei bestehender Internetverbindung an mySugr übermittelt werden. Allerdings kann die Zustimmung jederzeit widerrufen werden. Auch die von mySugr Junior erbetenen Zugriffsrechte (Identität, SMS, Telefon, Fotos und Kamera) werden durch die angebotenen Funktionen erläutert und sind nachvollziehbar. Zwar ist es nicht möglich, einzelne Funktionen zu unterbinden, dennoch bietet die App durch die Nutzungsbedingungen einen Überblick, welche Daten gesammelt und wie diese verwertet werden.

2.3 Lösungsansätze

Die Apps selbst bieten also nur selten eine sinnvolle Antwort auf die gegebene Privatheitsproblematik. Doch es gibt auch Arbeiten, die sich mit dem allgemeinen Schutz medizinischer Daten auf mobilen Geräten auseinandersetzen.

Zu diesen Arbeiten gehört beispielsweise „Device Data Protection in Mobile Healthcare Applications“ [WRR08]. Hier wird vor allem auf den Schutz der Daten eingegangen, der auf dem verwendeten Gerät selbst erfolgen sollte. Dafür wird die sogenannte *secure mobile capsule* entworfen und vorgestellt. Diese ist verantwortlich für die Kapselung der Daten. Zudem werden die Daten verschlüsselt gespeichert und können nur durch eine mitgelieferte Anzeige-App entschlüsselt werden. Das bedeutet, dass zu jedem Zeitpunkt die Daten für Außenstehende unlesbar sind.

Das Projekt *ECHO* [SWM+15, BKK+14] arbeitet dagegen mit einem Cloud-basierten Ansatz. Bei ECHO handelt es sich um eine Plattform, die den Arzt beim Patientenmanagement unterstützen soll. Dabei überwachen die Patienten ihren Gesundheitszustand selbst, beispielsweise mit dem Smartphone. Dazu wird die App ChronicOnline [Ope15b] (siehe Abb. 2.5) als Frontend verwendet. Die Überwachung funktioniert dabei mithilfe eines Fragekatalogs, den der Patient auf seinem Smartphone beantwortet. Die eingegebenen Daten werden an den Server

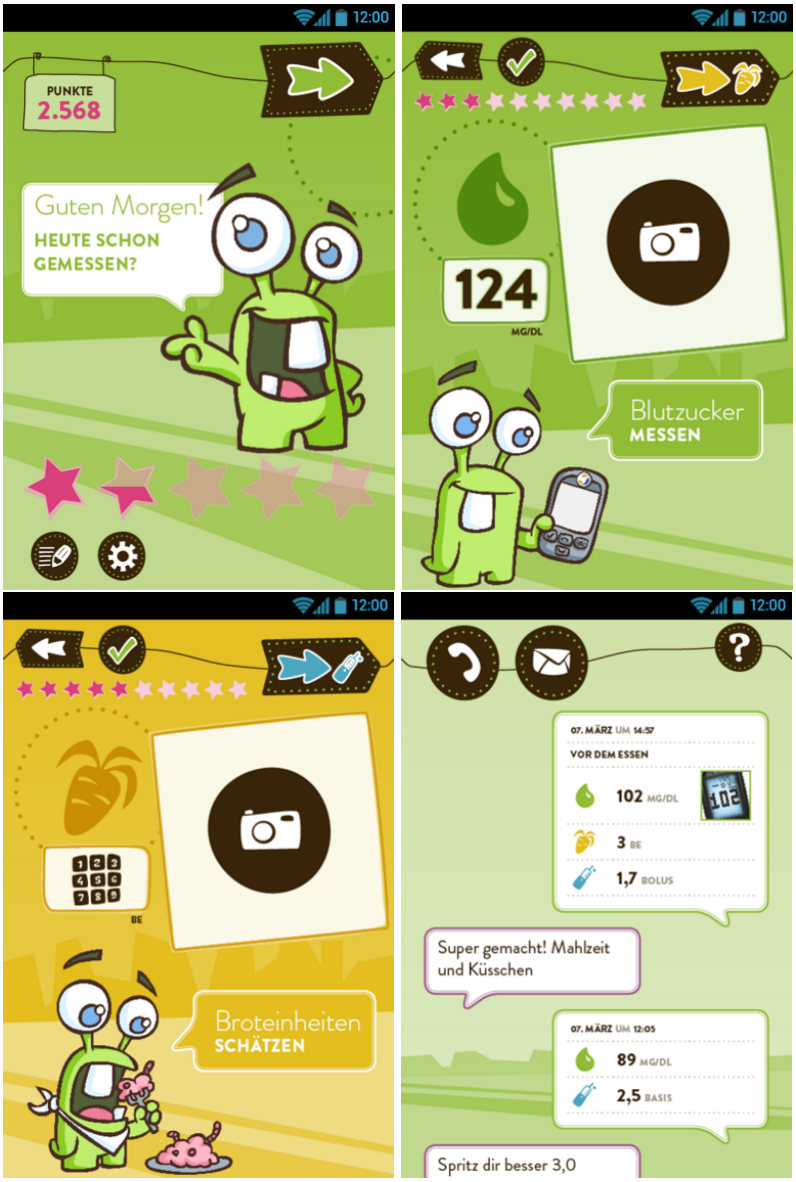

Abbildung 2.4: Screenshots von mySugr Junior [myS14a]

gesendet und dort in einer Datenbank gespeichert. Sicherheitsaspekte werden in ECHO gewährleistet, indem ein Zugriffsmanagementsystem der Datenbank in Kombination mit einem Rollensystem verwendet wird. Das bedeutet, dass jeder Nutzer nach Authentifizierung eigene Rechte und Sichten auf die Daten hat, die auf seiner Rolle basieren. So ist es möglich, per Arzt-IDs einem behandelnden Mediziner nur diejenigen Patienten anzuzeigen, die sich bei ihm in Behandlung befinden.

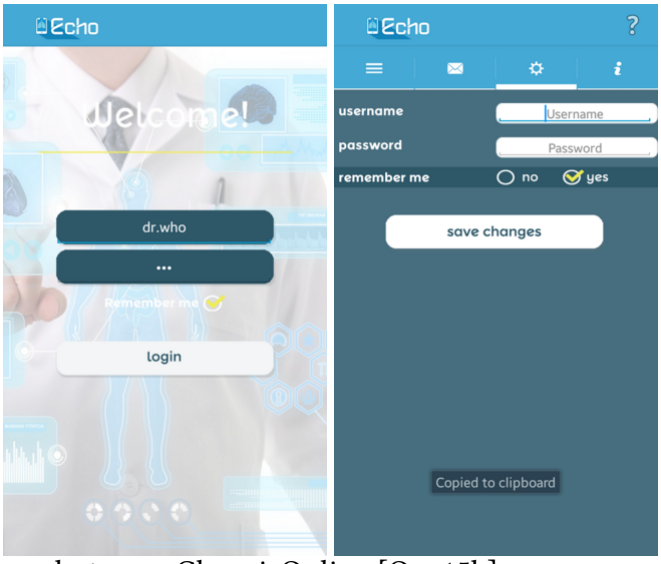

Abbildung 2.5: Screenshots von ChronicOnline [Ope15b]

2.4 Kriterien für Apps

Ein großes Problem bei der Verwendung von mHealth-Apps besteht darin, dass die Nutzer sich oftmals nicht im Klaren sind, welche Kriterien es zu beachten gilt. Verschiedene Krankenkassen und Magazine erstellen darum Checklisten, auf die die Nutzer zurückgreifen können.

Ein Beispiel hierfür ist die Checkliste der Techniker Krankenkasse (siehe [Tec15]). Sie fordert die Nutzer auf, sich zunächst darüber klar zu werden, welche Funktionen eine mHealth-App ihnen bieten soll. Anhand dieser Anforderungen fällt es ihm leichter, die passende App aus der Vielzahl der Angebote auszuwählen. Auch der Auftritt im App-Store wird behandelt. Dieser sollte ein Impressum des Anbieters sowie ausführliche Beschreibungen der Funktionalität beinhalten. Die Liste warnt allerdings auch vor Risiken die Privatheit der Daten. Sie weist daraufhin, dass Apps, die sensible Daten verarbeiten, mit Datenschutzerklärungen versehen sein müssen. Sollte diese Erklärung zu lang oder unverständlich sein, rät die Checkliste zum Verzicht.

Auch die DAK-Gesundheit [DAK15] bietet eine solche Übersicht an. Sie arbeitet hierfür mit einem Test, mithilfe dessen der Nutzer die App bewerten kann. Dieser fragt nach dem Impressum sowie den Autoren der App, um die fachliche Qualifikation einschätzen zu können. Auch hier wird auf die Wichtigkeit einer Datenschutzerklärung hingewiesen, allerdings erst, nachdem die zugehörige Frage mit „Nein“ beantwortet wurde. Über diese Erklärung hinaus sollte die App laut Test auch die Möglichkeit zur individuellen Anpassung der Datenschutzeinstellungen bieten. Weitere Kriterien sind beispielsweise das Stellen von Diagnosen, das als kritisch betrachtet wird, sowie eine ständige Internetverbindung. Diese Verbindung könnte Dritten zur unbefugten Einsicht in private Daten verhelfen.

Ein weiteres Kriterium, auf das C.Dirkes [VIT12] verweist, ist die Mitarbeit einer fachkundigen Person. Eine gute App zeichne sich darüber hinaus durch eine ansprechende Grafik, intuitive Bedienung und einen Mehrwert im Alltag aus. Lange Wartezeiten und Abstürze allerdings sprächen gegen die Qualität der App. Zudem solle man die App nicht als Arztersatz ansehen.

2.5 Probleme

Obwohl die Checklisten der Krankenkassen gute Anhaltspunkte für Nutzer und Entwickler bieten, finden sie nur selten Anwendung.

Die Ärzte Zeitung weist darauf hin, dass viele mHealth-Apps eine schlechte Qualität aufweisen, da selten Fachpersonal in deren Entwicklung eingebunden sei. Zudem sehen die Autoren die Datensicherheit kritisch, da „Datenschutzhinweise oder Autorenangaben häufig fehlten“ [Hüb15].

Genauer zeigt das eine Studie, die auf dem „Gesundheits-IT-Treff conhIT 2015“ vorgestellt wurde [Grä15]. Für diese Studie wurden 730 Apps auf ihre Datensicherheit getestet. Dabei schnitten mHealh-Apps nicht gut ab. Nur knapp 30% verfügten laut Studie über Datenschutzerklärungen.

Der Nutzer wird also nur selten über die Verwendung seiner privaten Daten informiert. Hinzu kommt, dass das Berechtigungssystem allein vom Betriebssystem des Smartphones verwaltet wird. Unerwünschte Funktionen oder Datensammlung können somit nicht deaktiviert werden. Die Privatheit des Nutzers wird hierdurch stark eingeschränkt.

2.6 Fazit

Die Arbeiten auf dem Gebiet der mHealth sind vielfältig. Viele Autoren sehen hier vor allem Potential und Möglichkeiten, um die moderne Gesundheitsfürsorge zu erweitern

(vgl. [Wie10] und [Hüb15]). Dennoch ist Vorsicht geboten. Obwohl den meisten Nutzern Privatheit sehr wichtig ist (vgl. [TM14]) sind die Möglichkeiten der Einflussnahme auf Apps begrenzt (vgl. [FEW12] und [FCH+11]). Oftmals fehlen grundlegende Angaben wie Datenschutzerklärungen (vgl. [Grä15]).

Gerade da es um medizinische Daten geht, ist Privatheit wichtig. Der Nutzer einer mHealth-App möchte seine Daten geschützt wissen und der App vertrauen können. Neben einigen positiven Beispielen (z.B. [myS14a]) finden sich im App-Store allerdings auch Apps, die große Mengen an Zugriffsrechten erfordern, ohne eine sichtbare Funktion zu liefern, die diese rechtfertigen würde (z.B. [Run15]). Umso wichtiger ist es darum, dem Nutzer aufzuzeigen, wofür welche Berechtigung benötigt wird und ihm die Möglichkeit zu geben, unerwünschte Zugriffe zu verbieten.

3 Candy Castle

Nachdem im vorangegangenen Kapitel bereits verschiedene Apps betrachtet wurden, wird im Folgenden die App Candy Castle vorgestellt. Sie bildet die Grundlage für diese Arbeit. Im weiteren Verlauf wird Candy Castle um Privatheitsaspekte erweitert werden, um dem Nutzer die Kontrolle über die Verwendung seiner Daten zu übertragen.

Um Candy Castle vorzustellen, werden zwei ältere Konzepte der App zur Betrachtung herangezogen. Diese liefern die Grundlage für die Überarbeitung des Spiels und erläutern die Idee hinter Candy Castle. Zudem können anhand der Vorgängerversionen erste Anforderungen für den Anforderungskatalog in Kapitel 4 erarbeitet werden.

3.1 Candy Castle nach Martin Knöll

Erstmals erwähnt wird das Konzept für Candy Castle 2010 in einer Veröffentlichung von M. Knöll [Knö10]. Der Fokus dieser Arbeit lag auf der Einbindung realer Ortsdaten in mHealth-Apps für Kinder mit Diabetes. Candy Castle wird hier als eine von drei Versionen für ortsgebundene mHealth-Apps vorgestellt.

Die Einbindung des Ortes geschieht dabei hauptsächlich aus zwei Gründen [KM12]. Zum einen können durch diese Verknüpfung weitere Daten gesammelt werden, die über den reinen Messwert hinausgehen. Mithilfe dieser Daten können neue Erkenntnisse zu Krankheitsverläufen und Umwelteinflüssen gesammelt werden. Zum anderen kann die Verknüpfung das Bewusstsein des Nutzers für das eigene Wohlbefinden gestärkt werden. Durch die Verknüpfung eines Messwerts mit GPS-Daten kann der Nutzer selbst Rückschlüsse auf den Einfluss ziehen, den eine bestimmte Umgebung auf sein Befinden hat.

Candy Castle ist hierbei eine Variation von Diabetes City [Knö08]. Diabetes City soll dabei jungen Diabetikern den Alltag erleichtern, indem es als mobiles Blutzuckertagebuch fungiert.

Die ursprüngliche Idee hinter Candy Castle war es, eine enge Zusammenarbeit zwischen Kind und Erziehungsberechtigten zu fördern. Hierbei sollten beide Nutzergruppen zusammen Tagebuch über ihren Alltag führen, indem sie beispielsweise Mahlzeiten oder Blutzuckermessungen dokumentieren. Mit diesen Einträgen verknüpft die App Ortsdaten der realen Welt. Diese Ortsdaten werden wiederum auf die im Spiel gezeigte Landschaft übertragen,

welche einen cartoonhaften Schlossgarten o.ä. darstellt. Die tägliche Dokumentation lässt an verschiedenen Orten dieser Landschaft Gebäude oder andere Elemente erscheinen, wobei diese Orte mit GPS-Daten der realen Welt assoziiert sind.

Auch der gemessene Blutzuckerwert soll einen Einfluss auf das Aussehen der Landschaft haben. Knöll definiert allerdings nicht näher, wie genau dies geschehen soll.

Candy Castle sowie die anderen Versionen von Diabetes City wurden gemeinsam mit Ärzten des Olgahospitals in Stuttgart erarbeitet. Die Evaluation fand durch zwölf Diabetespatienten zwischen acht und 13 Jahren statt.

3.2 Candy Castle nach Christoph Stach und Luiz F. M. Schlindwein

Zwei Jahre nach Knöll griffen C. Stach und L. F. M. Schlindwein das Grundkonzept für Candy Castle auf [SS12]. Candy Castle wurde hierbei als webbasierte Anwendung aufgebaut, die mit verschiedenen Endgeräten genutzt werden konnte. Auch hier steht die Verknüpfung von Ortsdaten mit Blutzuckermesswerten im Mittelpunkt. Anstatt allerdings die GPS-Daten auf eine spielinterne Landschaft zu übertragen, werden in dieser Version die realen Orte direkt mit den Messungen verknüpft. Das Schloss befindet sich anders als bei Knöll nicht in einer Cartoon-Welt, sondern wird auf einer normalen Straßenkarte angezeigt (siehe Abb. 3.1).

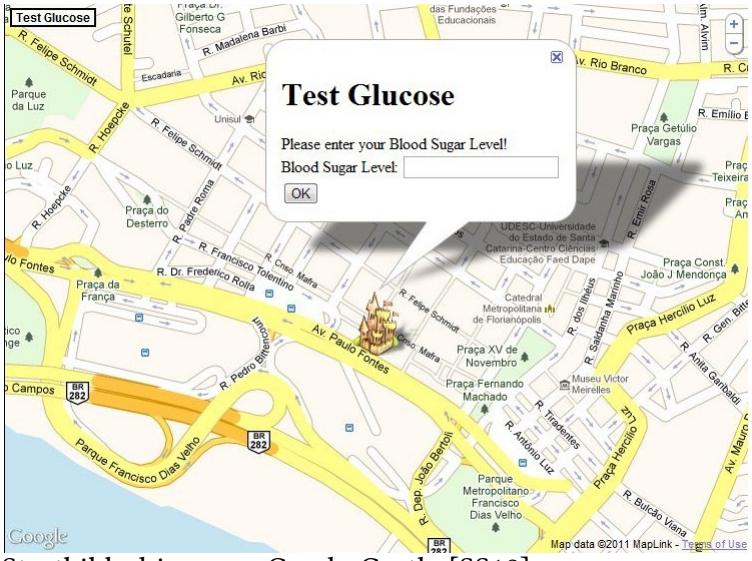

Abbildung 3.1: Startbildschirm von Candy Castle [SS12]

Der Spieler kann via Eingabefeld seinen Blutzuckerwert eintragen. Die Ortsdaten jeder Messung werden in dem Spiel als Turmstandort angezeigt. Zwischen je zwei Türmen wird eine

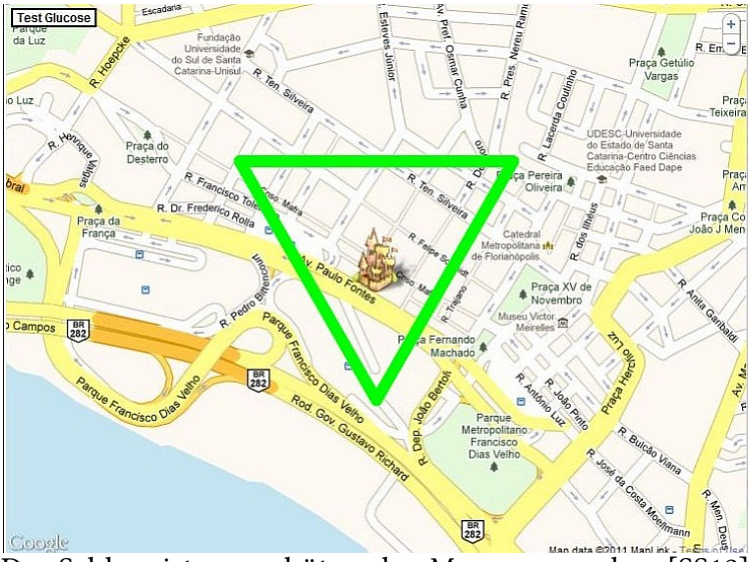

Abbildung 3.2: Das Schloss ist von schützenden Mauern umgeben [SS12]

Mauer gezogen, die das Schloss in der Mitte vor feindlichen Angriffen schützt (siehe Abb. 3.2). Wird die Messung an einem neuen Ort durchgeführt und so ein Turm hinzugefügt, erhält der Spieler mehr Punkte und stärkt damit sein Schloss. Das Ziel ist es, möglichst viele Punkte zu sammeln und so das Schloss vor Angriffen zu schützen.

Wird für längere Zeit kein Blutzuckermesswert eingetragen, wird das Schloss von den „dunklen Mächten“ angegriffen. Diese schwächen die Mauern. Sind die Mauern zerstört, ist das Spiel verloren und der Spieler muss erneut von vorn beginnen (siehe Abb. 3.3). Allerdings können die Mauern durch erneute Messungen repariert und durch zusätzliche Türme gestärkt werden.

Das Konzept sieht zusätzlich eine Schnittstelle für den behandelnden Arzt vor. Über diese können verschiedene Daten über die Gesundheit einzelner Patienten abgerufen werden.

3.3 Fazit

In beiden Konzepten handelt es sich bei Candy Castle also um ein Spiel, das als digitales Diabetestagebuch fungiert. Während es sich bei Knöll dabei um ein reines Konzept handelt, existiert bei Stach und Schlindwein bereits ein Prototyp. Dieser ist allerdings webbasiert, was sich einschränkend auswirken kann. So besitzt nicht jedes Kind ein Mobiltelefon mit Internetzugang. Auch Schwankungen in der Telefonnetzdichte können die Verfügbarkeit stark beeinträchtigen. Da Candy Castle aber mehrmals täglich genutzt werden und möglichst vielen Nutzern zugänglich sein soll, ist der webbasierte Ansatz kritisch zu bewerten. Zudem ist in diesem Konzept das Thema Privatheit nicht behandelt.

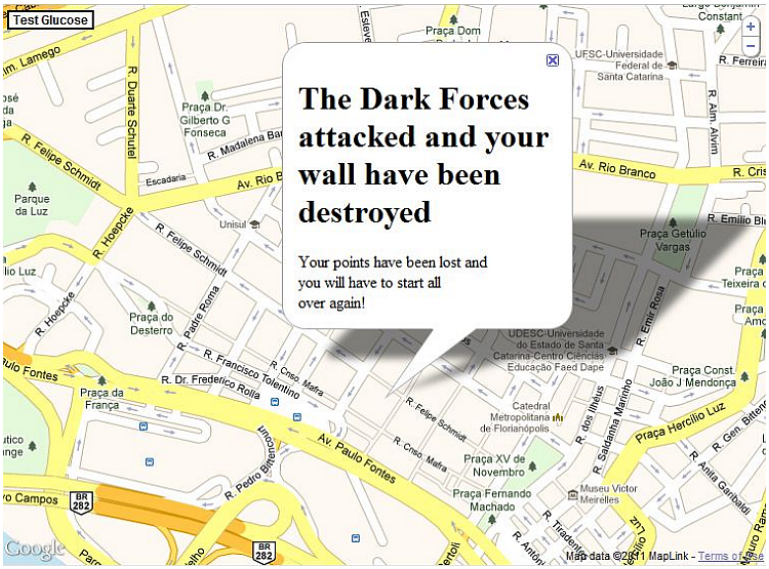

Abbildung 3.3: Ende des Spiels [SS12]

In der Evaluation durch Knöll wurden die jungen Patienten auch zu Privatheit befragt, zum Beispiel wer ihre Daten einsehen können sollte oder an welchen Orten sie ungern Messungen vornehmen würden. Hierbei fällt auf, dass die Antworten bereits sehr spezifisch sind. So antwortet ein achtjähriger Junge, dass er seinen Blutzucker nicht auf Partys messen wolle, da hierbei viele Menschen anwesend seien [Knö10]. Ein elfjähriges Mädchen sagte aus, dass sie in der Schule ungern Diabetes City spielen würde, unabhängig von Messungen. Auch die Personen, die eventuell Einblick in die Daten der App erhalten sollten, wurden sorgfältig ausgewählt. Vier von 12 Kindern wollten diese Informationen mit niemandem teilen, sieben wählten nur ihre Freunde als Mitwisser aus. Es ist also wichtig, auch jungen Nutzern Eigenbestimmung im Bereich Privatheit zu ermöglichen.

Diese Eigenbestimmung ist in der zweiten Version Candy Castles nicht gegeben. Die Verknüpfung zwischen Nutzer und Daten findet über den Google-Account statt. Hierdurch werden medizinische Daten an Dritte übertragen.

Zudem werden die Ortsdaten direkt auf der Karte dargestellt, wodurch Außenstehende durch Blick auf den Bildschirm Aufenthaltsorte des Nutzers bestimmen können.

Auch hat der Nutzer keine Möglichkeit, die Funktionen Candy Castles zu beschränken, falls er bestimmte Daten nicht preisgeben will. So ist beispielsweise eine Nutzung ohne die Eingabe von GPS-Daten nicht möglich.

Im Vergleich mit mySugr Junior (siehe Abschnitt 2.2.2) fällt auf, dass beide Konzepte von Candy Castle sich auf die Erfassung des Blutzuckerwertes konzentrieren, während mySugr Junior weitergehende Funktionen anbietet. So bietet mySugr Junior einen direkten Kontakt zu einer eingestellten Kontaktperson, um beispielsweise Fragen und Unklarheiten zu klären, und die Möglichkeit zur Eingabe von Broteinheiten, kurz *BE*. Diese BE geben an, wie viele

Kohlenhydrate in einem Lebensmittel vorhanden sind [Pet99] und werden zur Berechnung der Insulindosis benötigt. Sie stellen damit eine relevante Zusatzinformation dar, die für den Arzt von Interesse ist. Für den Nutzer und den behandelnden Arzt ergibt sich somit ein Mehrwert, den die Konzepte von Candy Castle nicht bieten. Allerdings fehlt in mySugr Junior die Verbindung zwischen Messwerten und Ort, welche ebenfalls für Nutzer und Arzt von Vorteil sein kann.

Diese Kritikpunkte sollen nun in Kapitel 4 als Anforderungen formuliert werden und so in die Planung des neuen Konzepts mit einfließen.

4 Erstellung eines Anforderungskataloges

Basierend auf den in Kapitel 2 und Kapitel 3 betrachteten Arbeiten soll im Folgenden ein Anforderungskatalog zusammengestellt werden. Es sollen hierbei sowohl positive als auch fehlende Aspekte anderer Arbeiten berücksichtigt werden. Der so entstandene Anforderungskatalog soll allgemeingültig sein und für die Entwicklung einer beliebigen mHealth-App herangezogen werden können.

Die verschiedenen Anforderungen werden hierfür in zwei Bereiche aufgeteilt:

1. **Funktionale Anforderungen:** Abschnitt 4.1 beschreibt die Anforderungen an die Funktionalität. Hierbei soll festgestellt werden, welche Aufgaben die App erfüllen können soll. Sie sind im Folgenden mit F und einer fortlaufenden Nummer gekennzeichnet.
2. **Nichtfunktionale Anforderungen:** Die in Abschnitt 4.2 genannten Anforderungen beschreiben die App über die Funktionalität hinaus. Hierzu gehören beispielsweise Anforderungen an die Bedienbarkeit. Doch auch Anforderungen an Datensicherheit und Privatheit werden hier diskutiert werden. Sie sind im Folgenden mit NF und einer fortlaufenden Nummer gekennzeichnet.

4.1 Funktionale Anforderungen

Im Folgenden sollen die funktionalen Anforderungen formuliert werden, die die verschiedenen Funktionen einer mHealth-App beschreiben.

4.1.1 Messung

mHealth-Apps werden oftmals für die Beobachtung eines bestimmten Messwertes eingesetzt. Bei diesem Messwert kann es sich beispielsweise um Gewicht, aber auch um Blutzuckermesswerte handeln. Um diese analysieren zu können, müssen die Werte über längere Zeiträume gespeichert werden und verfügbar sein.

F 1
Der Nutzer soll medizinische Messwerte eingeben können

F 2
Die App soll eingetragene Messwerte langfristig speichern können

4.1.2 Mehrwert

Im Gegensatz zu einem analogen Tagebuch hat eine mobile App deutlich mehr Möglichkeiten, den Nutzer zu unterstützen. Gerade im Bereich Gesundheit müssen Patienten oft viele Punkte berücksichtigen. Einige davon können ihnen durch die App abgenommen werden, wie beispielsweise die Erinnerung an die Medikamenteneinnahme oder die Möglichkeit, Kalorien zu berechnen.

F 3
Die App soll dem Nutzer wiederkehrende oder lästige Aufgaben abnehmen und ihn im Alltag unterstützen

4.1.3 Übersicht

Bei chronischen Krankheiten ist es wichtig, dass der behandelnde Arzt stets involviert ist. Hierzu sollten diesem die gemessenen Daten möglichst jederzeit und überschaubar zur Verfügung stehen. Auch sollte hier die Möglichkeit der Analyse geboten werden. Dies könnte über eine Webplattform wie beispielsweise *ECHO* [SWM+15] erreicht werden.

F 4
Die Messwerte sollen in eine externe Datenbank übertragen werden

F 5
Die externe Datenbank soll dem Arzt in seiner Arbeit unterstützen

4.1.4 Warnungen

Besonders in der Gesundheitsfürsorge von Kindern oder älteren Menschen sind meist weitere Personen eingebunden. Im Notfall sollten diese darum von der App benachrichtigt werden, um schnell handeln zu können.

F 6
Die App soll Warnungen versenden können

4.1.5 Personalisierbarkeit

Jeder Patient benötigt eine etwas andere Fürsorge. Es ist darum wichtig, dass eine mHealth-App personalisierbar ist. Die Eingabe von Blutdruckgrenzwerten oder Medikamentendosen sollte durch den Arzt vorgenommen werden können.

F 7
Die App soll personalisierbar sein

4.2 Nichtfunktionale Anforderungen

Dieser Abschnitt behandelt die nichtfunktionalen Anforderungen. Diese behandeln Aspekte, die über die reine Funktionalität hinausgehen, wie beispielsweise Bedienbarkeit oder Privatheit.

4.2.1 Motivation

Bei bestimmten Erkrankungen, wie beispielsweise Diabetes, ist es wichtig, mehrmals täglich Messungen durchzuführen. Dies kann anstrengend und demotivierend sein. Eine mHealth-App hat darum die Aufgabe, den Nutzer zu regelmäßigen Messungen zu ermutigen. Dies kann beispielsweise durch ein spielerisches Grundkonzept gelingen. Auch sollte die Darstellung der App klar und verständlich sein. Farben und Bilder können ebenso zur Motivation beitragen.

Wichtig ist, den Nutzer nicht zu demotivieren. Wird ein spielerisches Grundkonzept mit Punktevergabe gewählt, so sollte ein schlechter Messwert nicht durch Punkteabzug bestraft werden. Um regelmäßige Messungen zu erzielen, sollten Belohnungen darum unabhängig vom Messwert sein.

NF 1
Die App soll motivieren

NF 2
Die App soll eine ansprechende Darstellung besitzen

NF 3
Punktevergaben, oder ähnliches, sollen unabhängig von den gemessenen Werten sein

4.2.2 Nutzerfreundlichkeit

mHealth-Apps sollen schnell und verständlich nutzbar sein. Dauert die Bedienung der App zu lange, kann sich das negativ auf die Nutzungshäufigkeit auswirken. Ähnliches gilt für unverständliche und komplizierte Dialoge. Für eine intuitive und schnelle Bedienung ist es wichtig, dass Texte oder Symbole eindeutig und gut sichtbar sind.

NF 4
Die App soll leicht verständlich sein

NF 5
Die App soll intuitiv bedienbar sein

4.2.3 Diagnostik

Selbst, wenn eine App mithilfe medizinischer Fachkräfte entwickelt wurde, sollte sie nie als Arztersatz angesehen werden. Besonders kritisch ist die Stellung von Diagnosen durch die App, da diese den Nutzer dazu verleiten könnten, Arztbesuche durch die Nutzung der App zu ersetzen.

NF 6
Die App soll keine Diagnosen stellen

4.2.4 Verfügbarkeit

Für die Gesundheitsbetrachtung ist es wichtig, dass Daten regelmäßig gesammelt werden. Die Verfügbarkeit der Funktionalität spielt bei einer mHealth-App also eine wichtige Rolle. Die Grundfunktion der App soll auch bei fehlender Internetverbindung oder schlechtem Empfang gegeben sein. Hierfür ist es notwendig, dass Funktionen, die beispielsweise Onlinedienste benötigen, die Grundfunktion nicht beeinflussen.

NF 7
Die Grundfunktion soll jederzeit verfügbar sein

4.2.5 Privatheit

Da durch eine mHealth-App verschiedenste sensible Daten verwaltet werden, sollte der Nutzer allein das Bestimmungsrecht über diese Daten haben. Er sollte darum entscheiden können, auf welche externen Funktionen des Mobiltelefons die App Zugriff erhält. Auch soll der Nutzer stets die Kontrolle über die Verwendung seiner Daten haben. Alle Funktionen,

die die Daten verarbeiten, sollen durch den Nutzer aktiviert oder deaktiviert werden können. Zudem soll die App vor unbefugten Zugriffen von außen gesichert sein.

NF 8
Der Nutzer soll selbst bestimmen können, was der App erlaubt ist

NF 9
Der Nutzer soll selbst die über Verwendung seiner Daten bestimmen können

NF 10
Unbefugte sollen von Außen keinen Zugriff auf die App erhalten

4.2.6 Verfälschung

Da die Daten für die Analyse des Gesundheitszustandes verwendet werden, sollte es dem Nutzer nicht möglich sein, die Daten zu verfälschen. Dies ist für viele Apps nur schwer zu erreichen, kann aber zum Teil durch direkt verbundene Messgeräte erreicht werden.

NF 11
Die Messwerte sollen nicht durch den Nutzer verfälscht werden können

4.2.7 Medizinischer Mehrwert

Die Erfassung medizinischer Daten im Alltag der Patienten kann zu neuen Erkenntnissen oder verbesserten Behandlungsmethoden führen. Insofern sollte eine mHealth-App auch weitere Daten sammeln können, die solche weiterführenden Ergebnisse unterstützen. Beispielsweise könnte neben dem Lungenvolumen von Asthmapatienten auch der umgebende Luftdruck gemessen werden, um eventuelle Abhängigkeiten herauszufiltern. Dies kann beispielsweise durch die Vielzahl an Sensoren erreicht werden, über die ein Smartphone verfügt.

NF 12
Es soll ein medizinischer Mehrwert geschaffen werden

5 Technische Grundlagen für die Reimplementierung

Nachdem in Kapitel 4 die verschiedenen Anforderungen an eine mHealth-App erarbeitet wurden, wird in diesem Kapitel die Technik vorgestellt, mit deren Hilfe der Prototyp Candy Castles implementiert werden wird. Hierbei ist besonders zu beachten, dass die im vorangegangenen Kapitel geforderten Aspekte umgesetzt werden können. Darum wird neben dem Betriebssystem noch ein System für das Berechtigungsmanagement vorgestellt.

5.1 Das Betriebssystem

Anders als die erste Implementierung von Candy Castle (vgl. [SS12]) soll dieses Mal eine native App implementiert werden. Diese Entscheidung begründet sich auf Anforderung NF 7. Um eine möglichst hohe Verfügbarkeit zu erreichen, die auch ohne Internetverbindung gewährleistet werden kann, sollen die Daten lokal auf einem Smartphone oder Tablet verarbeitet werden können. Somit ist die Wahl eines Betriebssystems vonnöten, für das die App erhältlich sein soll. Die Wahl fiel auf Android.

Hierfür gibt es mehrere Gründe. Zum einen handelt es sich bei dem von Google angebotenen System um das am weitesten verbreitete Betriebssystem für mobile Geräte [hei15]. Ein weiterer Grund ist, dass Android die Nutzung systemeigener Funktionen durch die Apps Dritter unterstützt [Yuh15]. Diese Funktionen benötigt Candy Castle, um den Nutzer bestmöglich unterstützen zu können. Näheres hierzu findet sich in Kapitel 6.

Candy Castle soll sowohl mit älteren Versionen des Betriebssystems als auch mit Version 6.0 kompatibel sein, die 2015 auf den Markt kam. Während frühere Versionen von Android noch mit einem „Alles oder nichts“-Berechtigungssystem arbeiten, wurde dieses in Version 6.0, namentlich *Marshmallow*, geändert [Her15]. Anstatt einer App alle Zugriffsrechte gewähren zu müssen, können Nutzer von Version 6.0 jederzeit selbst festlegen, welche Berechtigungen eine App erhält. Allerdings bleibt unklar, für welche Funktionen die Berechtigungen genutzt werden. Auch bemängelt wird, dass die Apps nach Entzug der Berechtigungen oft abstürzten [Her15]. Dies ist darauf zurückzuführen, dass auch Zugriffe untersagt werden können, ohne die die App funktionsunfähig ist.

5.2 Das Berechtigungsmanagementsystem

Um Abstürze zu vermeiden, wird ein externes Berechtigungsmanagementsystem, oder Berechtigungssystem, für die Reimplementierung Candy Castles verwendet werden. Dieses bietet dem Nutzer mehr Freiheiten in der Wahl der akzeptierten Berechtigungen und ermöglicht der App auch ohne alle benötigten Rechte die fehlerfreie Funktion.

Hierfür gibt es verschiedene Lösungen, die C. Stach in seiner Arbeit „Wie funktioniert Datenschutz auf Mobilplattformen?“ [Sta13b] vorstellt und vergleicht. Hierfür formuliert der Autor fünf Anforderungen, anhand derer die unterschiedlichen Berechtigungssysteme bewertet werden:

Kontextsensitive Regeln: Ein Berechtigungssystem soll die Möglichkeit bieten, Regeln kontextsensitiv zu formulieren. Es soll also beispielsweise möglich sein, eine Regel abhängig von Tageszeit oder Ort anzugeben.

Laufzeitänderungen: Dem Nutzer soll jederzeit die Möglichkeit gegeben sein, bereits eingestellte Berechtigungen zu ändern oder neue Regeln einzufügen.

Absturzsicher: Eine in ihren Berechtigungen eingeschränkte App soll nicht wegen diesen Einschränkungen abstürzen. Es ist hier die Aufgabe des Berechtigungssystems, der App trotz Beschränkungen die korrekte Funktionsweise zu ermöglichen.

Dummy Daten: In manchen Fällen benötigt die App Daten, die von einer aufgerufenen Systemfunktion zurückgegeben werden. In diesem Fall soll das Berechtigungssystem Dummy Daten zurückliefern. Dabei handelt es sich um Daten, die einen verfälschten Wert beinhalten. Ein Anwendungsfall wäre beispielsweise der Abruf von GPS-Daten, die allerdings nur auf 100 m genau sind.

Feedback: Der Nutzer soll über die Folgen seiner Einstellungen und Beschränkungen der App-Funktionen informiert werden.

Hierbei fiel auf, dass keines der vom Autor betrachteten Systeme alle Kriterien erfüllte. Darum soll auf die in der selben Arbeit vorgestellte Privacy Management Platform, kurz *PMP*, zurückgegriffen werden [SM14]. Diese bietet nicht nur eine absturzsichere Zugriffskontrolle, sondern informiert den Nutzer auch über die Folgen seiner Einstellungen.

Bei der PMP handelt es sich um ein System, welches zwischen der App und den Funktionen des Betriebssystems steht. Ihr Modell ist in Abb. 5.1 dargestellt. Wichtig ist hierbei, dass es sich bei der PMP um eine selbstständige Anwendung handelt.

Apps werden auf der linken Seite des Modells als potentielle Gefahrenquellen eingeordnet. Die Elemente der PMP dagegen gelten als vertrauenswürdig. Diese Klassifizierung dient dazu, beiden Komponenten unterschiedliche Rechte zuteilen zu können. Während die App

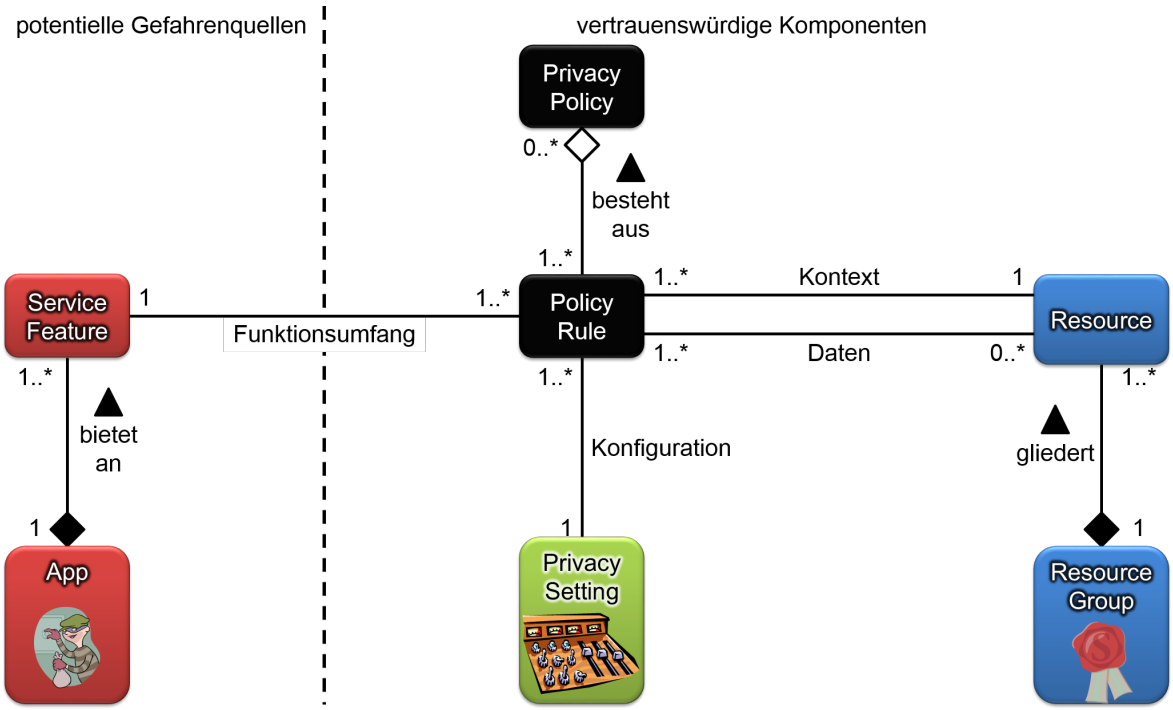

Abbildung 5.1: Das Modell der PMP [Sta13b]

keine Funktionen des Betriebssystems verwenden darf, verfügt die PMP über den vollen Zugriff auf diese.

Um die PMP nutzen zu können, müssen innerhalb der App die sogenannten *Service Features* definiert sein. Bei diesen handelt es sich um verschiedene Grundfunktionen der App, die klar voneinander getrennt werden.

Um die Interaktion zwischen der PMP und einer App genauer beschreiben zu können, werden im Folgenden einzelne Punkte an einem Beispiel verdeutlicht.

Beispiel
Nutzung der Kamera

Alle Service Features werden in einem xml-Dokument innerhalb der App benannt. Zudem beinhalten Apps, die auf die Nutzung der PMP ausgelegt sind, eine Methode zur Registrierung bei dieser. Verfügt eine App nicht über diese Daten, so wird ihre Ausführung durch die PMP blockiert. Um der PMP dennoch den Umgang mit diesen Apps zu ermöglichen, kann auf einen sogenannten *Gatekeeper* [Sta15] zurückgegriffen werden.

Enthält die App die Registrierungsmethode, wird diese bei der Installation automatisch ausgeführt. Hierbei wird zunächst überprüft, ob die PMP auf dem Gerät installiert ist. Ist

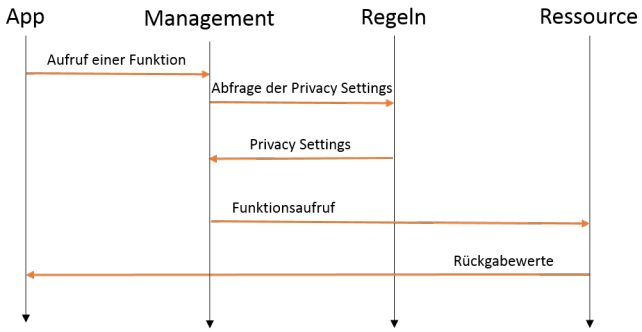

Abbildung 5.2: Ablauf eines Funktionsaufrufs

dies der Fall, registriert sich die App bei der PMP und die Berechtigungen für die einzelnen Service Features können gesetzt werden.

Diese Berechtigungen sind im Schaubild als *Privacy Settings* deklariert. Über sie kann der Nutzer festlegen, welche Berechtigungen die eben installierte App erhält. Zudem können sie jederzeit neu gesetzt werden. Die gesetzten Werte werden in sogenannten *Policy Rules* gespeichert. Diese ermöglichen auch eine kontextsensitive Einstellung.

Bei den Privacy Settings handelt es sich nicht nur um boolsche Werte, es können auch Zahlen oder Integrale eingestellt werden. Im Beispiel wären diese „Fotos schießen [Ja/Nein]“, „Videos aufnehmen [Ja/Nein], maximale Auflösung [Integer]“, „Blitz benutzen [Ja/Nein]“ und „Aufnahmen speichern [Ja/Nein]“. Welche Werte hierbei genau unterstützt werden, wird in den *Ressourcen* festgelegt.

Bei den Ressourcen handelt es sich um die eigentliche Schnittstelle zwischen App und Systemfunktion. Dabei verfügt jede Ressource über eine oder mehrere Methoden, auf die die App zugreifen kann. Sollte eine benötigte Ressource bei der Installation einer App fehlen, so lädt die PMP sich diese selbstständig herunter. Eine im Beispiel verwendete Kamera-Ressource würde unter anderem Methoden zur Fotografie oder zur Aufnahme von Videos bereitstellen. Über die korrekte Implementierung einer Funktion hinaus besitzt die Ressource allerdings Methoden, die verfälschte Werte zurückgeben und je nach gesetztem Privacy Setting statt der korrekten Methode aufgerufen werden.

Abb. 5.2 stellt den Ablauf eines Funktionsaufrufs dar. Das Management der PMP überprüft zunächst, ob bereits eine Regel für das genutzte Service Feature existiert. Danach wird, abhängig von den eingestellten Priacy Settings, die entsprechende Funktion ausgeführt. Falls noch keine Regel zu dem entsprechenden Service Feature existiert, wird der Nutzer darauf hingewiesen, diese festzulegen.

In Candy Castle soll die PMP dazu genutzt werden, die sensiblen Daten des Nutzers zu schützen. Hierfür sollen Ressourcen zur Datenkapselung von Messwerten verwendet werden. Auch soll der Nutzer selbst bestimmen können, welche Daten die App erhält (beispielsweise

GPS-Daten) und wie mit diesen umgegangen wird. Dennoch soll die App nutzbar bleiben und eine Grundfunktionalität erfüllen können. Auch ist es wichtig, die medizinischen Daten nicht zu verfälschen bzw. falls eine Verfälschung durch den Nutzer gewünscht ist, diese deutlich zu machen. Dies ist beispielsweise durch die Verwendung negativer Verfälschungswerte möglich.

Positiv an der PMP ist zusätzlich, dass entwickelte Ressourcen wiederverwendet werden können. So können die Ressourcen für Candy Castle abgeändert und beispielsweise in einem Spiel für Asthmapatienten genutzt werden.

Im folgenden Kapitel gilt es darum, ein Spielkonzept zu entwickeln, welches für die Nutzung der PMP geeignet ist.

6 Das neue Konzept für Candy Castle

Im Folgenden wird ein neues Konzept für Candy Castle ausgearbeitet , welches sich mithilfe der in Abschnitt 5.2 vorgestellten PMP realisieren lässt. Dazu wird zunächst das allgemeine Spielkonzept vorgestellt, das die Interaktion des Nutzers mit der App beschreibt. Zuletzt werde, die Funktionen des Konzepts in Service Features unterteilt und auf anfallende sensible Daten analysiert.

6.1 Das Spielprinzip

Die Entwicklung des Konzepts basiert auf fünf grundsätzlichen Anforderungen, die sich aus den vorangegangenen Versionen und dem Anforderungskatalog ergeben. Diese stellen die Grundlage für das Konzept. Alle anderen Anforderungen werden in der weiteren Ausarbeitung berücksichtigt.

Bedienbarkeit (Abschnitt 6.1.1): Unabhängig von der Zielgruppe und der Funktion der mHealth-App muss sie schnell und einfach zu bedienen sein. Hierbei sollen Leitfäden zur Usability (z.B. [ZFAIT15]) herangezogen werden.

Kinderfreundliches Design (Abschnitt 6.1.1): Da Candy Castle als Diabetestagebuch für Kinder fungieren soll, muss das Nutzungsprinzip sowie die Darstellung an diese Nutzergruppe angepasst werden.

Verknüpfung Messwert - Ort (Abschnitt 6.1.3): Wie bereits in beiden vorangegangenen Versionen soll auch das neue Candy Castle Messwerte mit Orten verknüpfen können. Hierdurch kann das Bewusstsein für den Einfluss verschiedener Umwelteinflüsse auf das eigene Wohlbefinden erhöht werden. Auch können die so gesammelten Daten zur medizinischen Forschung weiterverwendet werden.

Mehrwert (Abschnitt 6.1.4): Die App soll gegenüber einem analogen Diabetestagebuch einen echten Mehrwert bieten. Dieser kann durch Erinnerungen, zusätzliche Funktionen oder auch die automatische Zugänglichkeit der Daten für medizinisches Personal geschaffen werden.

Privatheit: Die Bestimmung über Funktionen und Zugriffe der App soll allein beim Nutzer liegen. Er bestimmt, ob und wie seine medizinischen Daten durch die App verwendet werden. Um dies zu realisieren, wird die App in Abschnitt 6.3 auf die Nutzung der PMP angepasst.

6.1.1 Bedienbarkeit

Um Candy Castle ansprechend zu gestalten, ist eine intuitive Bedienung notwendig. Hierfür wird bei der Erstellung des Konzepts darauf geachtet, Bezeichnungen eindeutig zu formulieren. Auch ist es wichtig, einzelne Ansichten innerhalb der App nicht zu überladen, sondern ein klares Design zu bieten. Die Schwierigkeit hierbei besteht darin, eine Mischung zwischen Übersichtlichkeit und Funktionalität zu finden, die gut bedienbar ist und trotzdem einen Mehrwert bietet. Näheres hierzu findet sich in Abschnitt 6.1.4.

6.1.2 Kinderfreundliches Design

Neben der Bedienung der App spielt auch das Design eine wichtige Rolle in Bezug auf Kinderfreundlichkeit. Hierfür gab es in vorangegangenen Versionen von Candy Castle zwei unterschiedliche Ansätze. Während M. Knöll von einer comichaften Landschaft spricht (siehe Abschnitt 3.1), verwenden C. Stach und L. Schlindwein eine schlichtere Darstellung (siehe Abschnitt 3.2), die das Schloss sowie eine Straßenkarte enthält. In der Reimplementierung soll jedoch eine gezeichnete Darstellung, siehe Abb. 6.1, genutzt werden. Dies geschieht aus drei Gründen:

Übersichtlichkeit: Straßenkarten beinhalten sehr viele Informationen, die für die Funktion von Candy Castle nicht relevant sind. Hierzu gehören beispielsweise Straßennamen, die Kennzeichnung wichtiger Orte und Straßenarten. Spielrelevante Informationen können hierbei untergehen.

Privatheit: Wird eine Straßenkarte als Grundlage verwendet, werden auf dieser konkrete GPS-Daten angezeigt. Es besteht für Dritte also die Möglichkeit, diese direkt einzusehen. Wird statt einer konkreten Umgebung eine abstrahierte Version genutzt, so kann eine Abbildung realer Orte auf Orte der Comic-Welt vollzogen werden, die von außen nicht invertiert werden kann.

Neben außenstehenden Dritten, die per Sicht auf das Handydisplay Daten einsehen können, kann auch die App selbst die konkreten GPS-Daten speichern und nach Außen weitergeben. Durch die Abbildung auf eine gezeichnete Welt ist es möglich, Orte darzustellen ohne der App Informationen über die realen Koordinaten mitzuteilen. Die App erhält lediglich Gitterzellen, in denen sie die Orte darstellen soll.

Abbildung 6.1: Die Welt Candy Castles im neuen Konzept

Kinderfreundlichkeit: Während Straßenkarten die direkte Umgebung des Nutzers widerspiegeln, bietet eine gezeichnete Comic-Welt mehr Platz für Ideen der Kinder. Das Ziel ist hierbei, Interesse an gezeichneten Elementen zu wecken und die Fantasie anzuregen.

Zuletzt soll eine Motivation für die regelmäßige Nutzung der App geschaffen werden. Hierfür wird Candy Castle als Spiel implementiert, um einen Anreiz zu bieten. Da die App vor allem dazu genutzt wird, Blutzuckermesswerte einzutragen, werden für diese Einträge Punkte vergeben. Hierbei ist der Wert, der eingetragen wurde, unwichtig. Punkte werden nur für die Nutzung, nicht für den Gesundheitszustand vergeben. Täglich findet innerhalb der App ein Angriff der „Dunklen Mächte“ statt, welcher den Spieler Punkte kostet. Dieser Punkteverlust soll den Spieler motivieren, mehrmals täglich Werte einzutragen. Eine genaue Aufstellung der Punkte folgt in Abschnitt 7.4. Um schnell und verständlich darzustellen, dass eine erneute Messung nötig ist, wirken sich die gesammelten Punkte direkt auf die Darstellung des Schlosses aus. In Abb. 6.2 sind die drei Stufen des Schlosses dargestellt, die je nach Punktestand angezeigt werden. Wird die dritte Stufe angezeigt, verbleibt noch maximal ein Drittel der Gesamtpunktzahl.

Abbildung 6.2: Die drei Stufen des Schlosses

Abbildung 6.3: Die Drachenhöhle markiert einen beliebigen Messort auf der Candy Castle Karte

6.1.3 Verknüpfung Messwert - Ort

Candy Castle war ursprünglich dafür gedacht, eine Verbindung zwischen medizinischen Messwerten und dem Ort herzustellen, an denen sie gemessen wurden. Diese Verbindung kann auch in der Reimplementierung hergestellt werden. Jeder ins Spiel eingetragene Messwert enthält darum neben dem Blutzuckerwert auch die Koordinaten des Messortes.

Um den Spieler zu motivieren, möglichst verschiedene Orte aufzusuchen, fließt die Erschließung dieser als Bonuspunkte in die Wertung ein. Allerdings werden hierbei nicht nur neue Orte berücksichtigt, also Koordinaten, die bislang nicht im System eingetragen waren, sondern auch Orte, an denen seit einer gewissen Zeit nicht gemessen wurde.

Im Spiel selbst werden die Koordinaten nicht direkt angezeigt, sondern auf die Welt Candy Castles abgebildet. Hierfür wird die Karte des Spiels als Gitter aufgefasst, wobei das Schloss den Nullpunkt markiert. Dessen reale Koordinaten werden vom Spieler festgelegt, beispielsweise an seinem Wohnort. Alle weiteren Koordinaten, die eingetragen werden, werden relativ zu diesem Punkt im Gitternetz angezeigt. Dies geschieht mit kleinen Elementen der Schlossumgebung, wie Gärten, Schätzen oder Drachenhöhlen (zum Beispiel Abb. 6.3).

Zudem wird jedes Mapping innerhalb der App mit einem Counter versehen, der bei jedem neuen Messwert dekrementiert wird. Nur Mappings, deren Counter größer als null ist, werden auf der Karte angezeigt, um eine Überladung zu verhindern. Initialisiert wird der Counter mit fünf.

Abbildung 6.4: Die Ampel-Darstellung der Launenskala

6.1.4 Mehrwert

Bei der Betrachtung der App mySugr Junior [myS14a] in Abschnitt 2.2.2 fiel auf, dass diese neben der reinen Speicherung des Blutzuckermesswertes weitere Funktionen beinhaltet, welche während der Behandlung vorteilhaft sind. Hierzu gehört die Erfassung der Broteinheiten, deren Wert einem Messwert zugeordnet werden kann. Da die BE eine wichtige Rolle im Alltag eines Diabetikers einnehmen, wird auch Candy Castle eine Möglichkeit bieten, diese Werte zu speichern. Zusätzlich können die BE für bestimmte Nahrungsmittel direkt über die App abgerufen werden. Dies ist für Kinder ein großer Vorteil, da die BE normalerweise aus Tabellen ausgelesen werden müssen. Für Kinder kann die Arbeit mit Tabellen allerdings schwierig und verwirrend sein. Das Abrufen der BE kann entweder durch manuelle Eingabe eines Nahrungsmittels oder durch das Scannen eines Barcodes realisiert werden.

Auch von Belang ist die Laune des Nutzers oder auch Informationen über Übelkeit und körperliche Aktivität. Diese drei Aspekte werden ebenfalls in einen Messwert einfließen. Für eine intuitive Eingabe dieser Werte wurde ein Ampelprinzip gewählt. Um diese kindgerecht zu gestalten, wurde die Ampel um Icons erweitert (siehe Abb. 6.4).

Trägt der Nutzer diese zusätzlichen Messdaten ein, so erhält er weitere Punkte im Spielverlauf.

Die Messwerte können dem behandelnden Arzt zugänglich gemacht werden, indem sie an eine Webdatenbank gesendet werden. Der behandelnde Arzt kann diese dort einsehen und verschiedene Analysen auf den Daten veranlassen, ähnlich zu ECHO [SWM+15].

Neben dem Arzt sollen auch die Eltern in die Therapie ihres Kindes miteinbezogen werden. Da bei Kindern mit Diabetes durch übermäßige körperliche Aktivität ein erhöhtes Risiko für Unterzucker besteht, sollen zu niedrige Blutzuckerwerte an die Eltern weitergeleitet werden. Dadurch sind diese informiert und haben die Möglichkeit, sich mit ihrem Kind in Verbindung zu setzen.

Um die externe Speicherung der Daten und die Analyse auf Unterzucker zu ermöglichen, ist es nötig, Candy Castle personalisieren zu können. Daten wie der Grenzwert zur Unterzuckeranalyse, eine zu informierende Kontaktperson und Zugangsdaten zur Messwertdatenbank müssen in der Datenbank hinterlegt sein und geändert werden können.

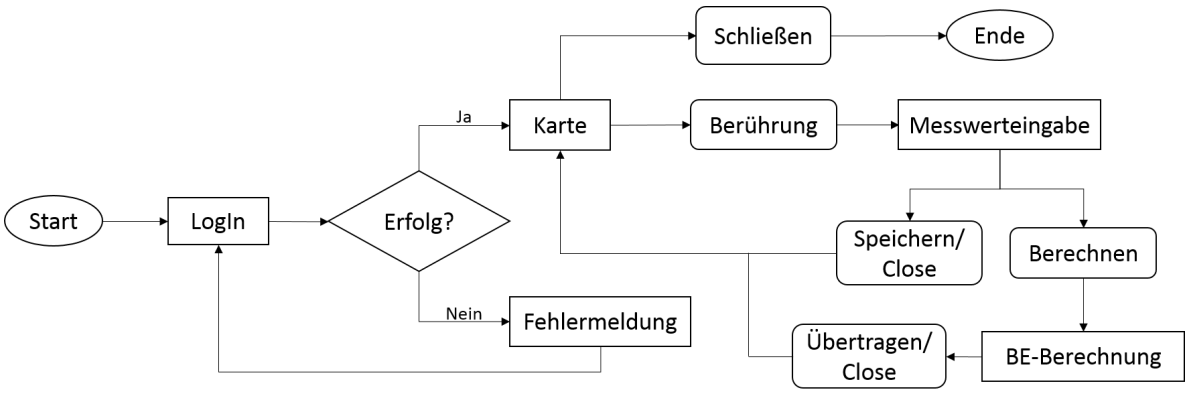

Abbildung 6.5: Das Ablaufdiagramm zu Candy Castle

Abbildung 6.6: Der LogIn-Bildschirm der Reimplementierung

6.2 Ein visueller Prototyp

Nun sollen die oben genannten Aspekte des Konzepts in ein Modell des Prototyps umgewandelt werden. Hierzu werden die verschiedenen Ansichten des Spiels konzeptionell dargestellt und kurz geschildert. In Abb. 6.5 ist der Ablauf innerhalb der App dargestellt. Rechteckige Felder stellen die einzelnen Ansichten der App dar, Rauten zeigen Überprüfungen an. Rechtecke mit abgerundeten Ecken stehen für Interaktionen des Nutzers mit der App.

Startbildschirm: Diese Ansicht (Abb. 6.6) wird angezeigt, sobald Candy Castle gestartet wurde. Bei der erstmaligen Ausführung kann der Nutzer einen Nutzernamen und ein beliebiges Passwort festlegen.

Kartenansicht: Sobald Benutzername und Password korrekt eingegeben wurden, kann auf die Karte zugegriffen werden (Abb. 6.7). Sie beinhaltet das Schloss, dessen Zustand die

Abbildung 6.7: Die Hauptanzeige Candy Castles

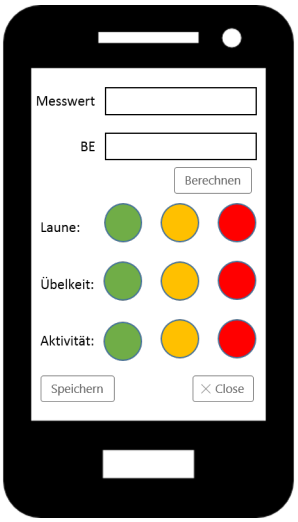

Abbildung 6.8: Die Messwerteingabe

momentane Punkteanzahl widerspiegelt, sowie die Elemente, die bereits besuchte Orte kennzeichnen.

Messwerteingabe: Tippt man auf den Kartenbildschirm, wird die Messwerteingabe(Abb. 6.8) aufgerufen. Hierin lassen sich die Blutzuckermesswerte sowie Broteinheiten und verschiedene Skalen eingeben.

BE-Berechnung: Drückt man bei der Messwerteingabe auf *Berechnen*, so öffnet sich die BE-Berechnung (Abb. 6.9). Die BE können entweder nur angezeigt oder mit *Übertragen* in die Messwerteingabe übernommen werden. Hier wurde die manuelle Eingabe gewählt, da diese für den Prototyp weniger umfangreich ist. Barcodescanner können mittels der ZXing-Bibliothek [Ope15a] unter Java realisiert werden.

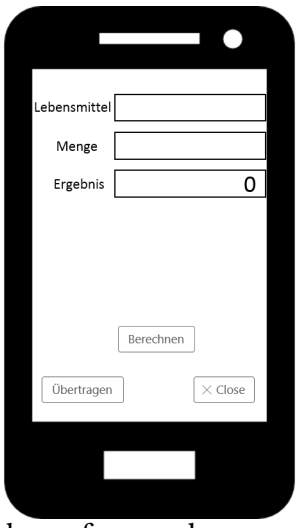

Abbildung 6.9: Hier können BE abgerufen werden

6.3 Anpassungen für die PMP

In seiner Funktion als Diabetestagebuch verarbeitet Candy Castle eine Vielzahl sensibler Daten. Darum soll der Nutzer die Möglichkeit erhalten, die Rechte der App selbst zu bestimmen. Candy Castle wird deshalb auf die Nutzung der PMP (Abschnitt 5.2) ausgelegt. Hierfür ist es nötig, die Funktionen der App in Service Features zu unterteilen und die Möglichkeiten der Beschränkung durch den Nutzer herauszuarbeiten. Das Konzept muss aufgrund dieser Beschränkungen so angepasst werden, dass der Nutzer trotz Deaktivierung bestimmter Funktionen das Spiel nutzen kann.

Um die Funktionen der App unterteilen und analysieren zu können, soll ein Use Case Diagramm als Grundlage verwendet werden. Da die App nur vom Patienten genutzt wird, sind nur dessen Use Cases dargestellt. Eltern sowie Arzt interagieren nicht mit der lokalen App, sondern über externe Dienste.

Abb. 6.10 stellt die Funktionen des Neukonzepts dar. Gestrichelte Linien stellen die *Extend-Beziehung* dar. In diesem Fall bedeutet das, das immer erst eine Anmeldung erfolgen muss, bevor auf die anderen Funktionen zugegriffen werden kann. Gepunktete Linien stehen für die *Include-Beziehung*. Wird ein neuer Eintrag angelegt, so werden, sofern vom Nutzer erlaubt, die damit verbundenen Funktionen gestartet. Aus deren Ergebnissen setzt sich dann der Messwert zusammen, der gespeichert wird.

Neben den in Abschnitt 6.2 aufgeführten Funktionen, kommen im Use Case Diagramm die Fälle *Messwerte verwalten*, *Personalisieren* sowie *Karte prüfen* hinzu.

Zusätzlich zu den im Diagramm angegebenen Funktionen müssen noch automatische Vorgänge berücksichtigt werden. Diese sind in Abb. 6.11 als gestrichelte Ellipsen angegeben.

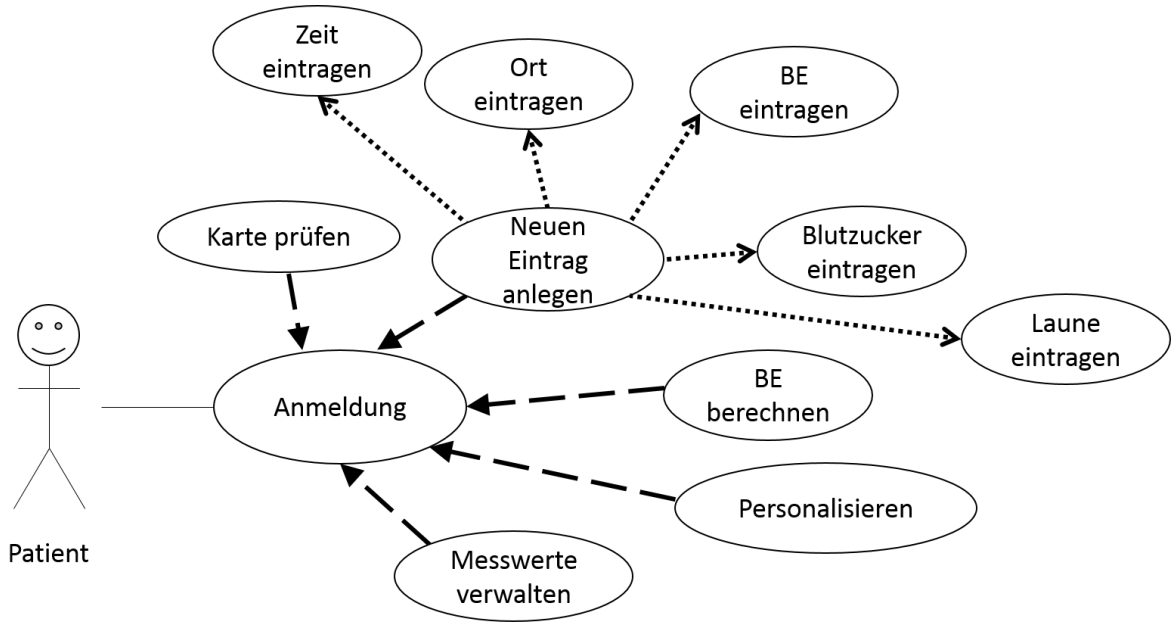

Abbildung 6.10: Das Use Case Diagramm für das Neukonzept Candy Castles

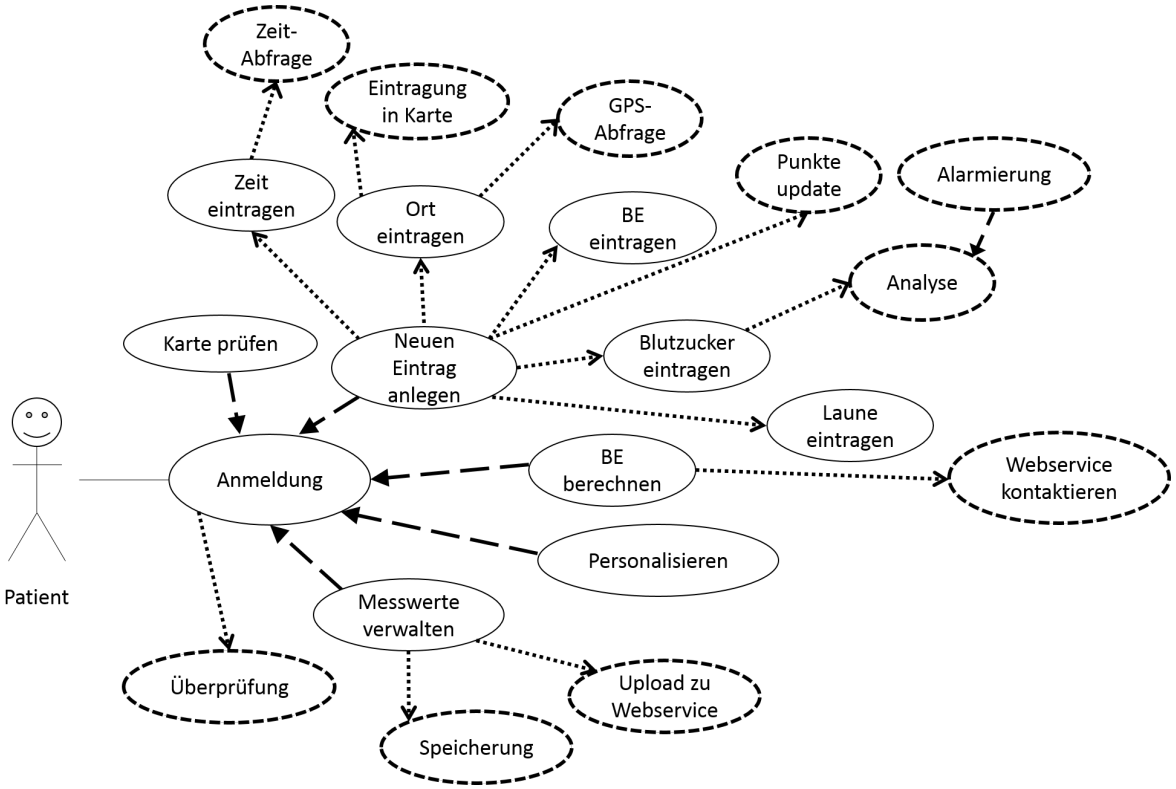

Abbildung 6.11: Das Diagramm wurde um die automatischen Vorgänge erweitert

Service Feature	Anfallende Daten
Überprüfung der Anmeldedaten	Benutzername, Passwort
Speicherung der Messwerte	**Messwerte** (Blutzuckerspiegel, Laune, Aufenthaltsort,...)
Upload der Messwerte	**Messwerte**, LogIn-Daten
Personalisieren	Benutzername, Passwort, **LogIn-Daten für Webservice**, **Grenzwert**, **Kontaktperson**
Abruf der BE von einem Webservice	**Verzehrte Nahrungsmittel**
Blutzucker eintragen	**Blutzuckermesswert**
Analyse des Messwerts	**Blutzuckermesswert**, **Grenzwert**
Alarmierung der Kontaktperson	**Blutzuckermesswert**, **Kontaktperson**
Laune eintragen	Befinden
Broteinheiten eintragen	**Verzehrte Nahrungsmittel**
GPS-Abfrage	**Aufenthaltsort**
Eintragung der Orte in die Karte	**Aufenthaltsort**, Abbildung
Zeit-Abfrage	**Uhrzeit der Messung**
Karte überprüfen	-

Tabelle 6.1: Die Service Features und die anfallenden Daten

Da nun alle Funktionen Candy Castles im Use Case Diagramm enthalten sind, kann dieses als Grundlage für die Herleitung der Service Features verwendet werden. Hierzu wird jede Ellipse als eigene Grundfunktion betrachtet. Ellipsen, die andere über eine Include-Beziehung beinhalten, werden nun ausgenommen und durch ihre Teilfunktionen ersetzt, da diese die Funktion ausreichend beschreiben. Die einzige Ausnahme hiervon ist *Blutzucker eintragen*, da die Teilfunktionen hier nicht die komplette Funktionalität abbilden. Die sich ergebenden Service Features sind in Tabelle 6.1 dargestellt. Nun gilt es, zu den einzelnen Service Features die anfallenden sensiblen Daten herauszuarbeiten. Diese sind ebenfalls in Tabelle 6.1 aufgelistet. Als *anfallend* sollen nicht nur die Daten betrachtet werden, die in dem jeweiligen Feature erhoben werden, sondern auch diejenigen, die von der Funktion verwendet werden. *Sensibel* sind solche Daten, die auf den Gesundheitszustand, den Aufenthaltsort oder andere persönliche Daten schließen lassen. Sie werden **besonders markiert**.

Mithilfe von Tabelle 6.1 kann nun festgestellt werden, welche Funktionen bei der Erstellung von Ressourcen für die PMP berücksichtigt werden müssen. Grundsätzlich soll es möglich sein, jedes Service Feature zu aktivieren oder zu deaktivieren. Um allerdings die Funktionalität Candy Castles garantieren zu können, müssen hierbei Einschränkungen vorgenommen werden:

Blutzucker eintragen: Die Aktivierung dieser Funktion ist die Voraussetzung für die Nutzung Candy Castles, da es sich hierbei um die Hauptfunktion handelt.

Speicherung der Messwerte: Auch diese Funktion ist notwendig, um die App zu nutzen. Werden die Messwerte nicht gespeichert, ist Candy Castle als Diabetestagebuch unbrauchbar.

Log in: Zwar ist diese Funktion nicht unbedingt für die volle Funktionalität Candy Castles notwendig, dennoch geht durch ihre Deaktivierung Privatheit verloren. Ohne Benutzername und Passwort können Dritte auf Candy Castle zugreifen und somit die Privatsphäre des Nutzers verletzen.

Auch wenn diese Funktionen nicht deaktivierbar sind, kann durch die Nutzung existierender PMP-Funktionen und -Erweiterungen dennoch ein Schutz der Daten erreicht werden. So können die Blutzuckermesswerte mithilfe von Verschlüsselungsmechanismen der PMP für die App unlesbar gemacht werden, ähnlich der bereits vorgestellten secure mobile capsule [WRR08]. Dies beeinflusst die Funktion der App nicht, da diese die benötigten Daten stets durch die PMP erhält. Die Speicherung der Messwerte kann im Secure Data Container [SM15] erfolgen, so dass auch hier Privatheit sichergestellt werden kann.

Da nun das Konzept vollständig ist und für die Implementierung als PMP-Anwendung vorbereitet wurde, folgt in Kapitel 7 die Realisierung Candy Castles.

7 Realisierung

Das in Kapitel 6 vorgestellte Konzept soll nun als Prototyp implementiert werden. Hierfür sind mehrere Schritte nötig. Zunächst soll in Abschnitt 7.1 die Datenhaltung des Prototyps beschrieben werden. Um diese Daten auch extern verfügbar zu machen, wird in Abschnitt 7.2 die dafür benötigte Struktur diskutiert. Auch werden hier weitere benötigte externe Services vorgestellt. Abschnitt 7.3 stellt anschließend die Ressourcen vor, die für die PMP implementiert werden. Diese müssen alle Service Features der App abdecken. Auch werden in diesem Abschnitt die benötigten Privacy Settings festgelegt. Zuletzt gilt es noch, die App selbst zu realisieren. In Abschnitt 7.4 werden darum letzte Implementierungsdetails geklärt, bevor die eigentliche Realisierung beschrieben wird.

7.1 Die Datenhaltung

Zunächst werden alle in Candy Castle eingegebenen Daten lokal auf dem mobilen Gerät gespeichert. Hierfür wird eine relationale Datenbank verwendet, die direkt auf dem Smartphone liegt. Für den Prototyp werden die Daten unverschlüsselt abgespeichert, allerdings kann die Implementierung auch mithilfe einer verschlüsselten Datenbank, wie z.B. der SDC [SM15], realisiert werden.

Zunächst müssen verschiedene Nutzerprofile in der Datenbank verwaltet werden. Auch, wenn in den meisten Fällen nur ein Benutzer die App verwenden wird, soll es dennoch die Möglichkeit zur mehrfachen Registrierung geben. So kann das Datenmodell auch für eine externe Speicherung verwendet werden, wo mehrere Nutzer verwaltet werden müssen. Die Datenhaltung für den Benutzer ist in Abb. 7.1 dargestellt.

Die User-ID (*UID*) wird fortlaufend vom System vergeben. Der *Name* dagegen kann bei der Registrierung vom Nutzer gewählt werden. Er wird zusammen mit dem *PIN/Passwort* zur Authentifizierung genutzt. Die *Patienten-ID*, die *Arzt-ID*, die *Webservice URL* sowie das *Webservice Pwd* werden für die Anmeldung in einer externen Plattform wie in Abschnitt 7.2 vorgestellt benötigt. Hierbei wird die URL einem Nutzer zugeordnet, damit es möglich ist, für jeden Nutzer einen eigenen Webservice anzulegen. Somit können mehrere Nutzer die gleiche Instanz von Candy Castle benutzen, auch wenn sie von verschiedenen Ärzten behandelt werden, die auf verschiedene externe Datenbanksysteme zugreifen. Bei der *Kontaktperson*

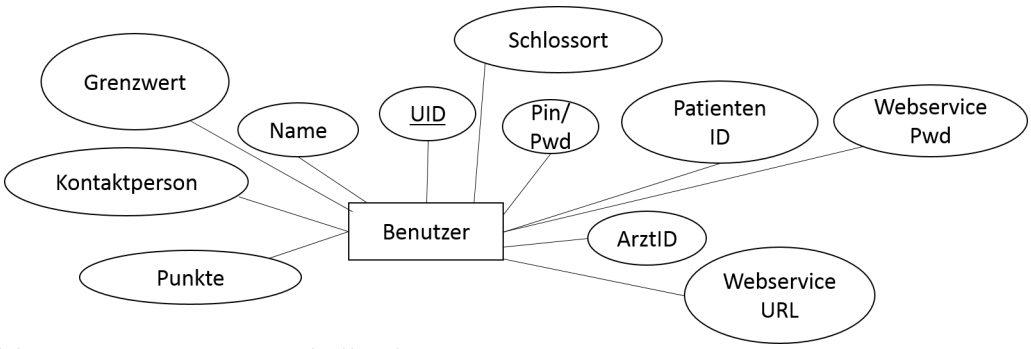

Abbildung 7.1: Die Nutzertabelle als ER-Diagramm

handelt es sich um eine Telefonnummer, die eine Textnachricht erhält, sollte der gemessene Blutzuckerwert den *Grenzwert* unterschreiten. Der Grenzwert ist als Gleitkommazahl hinterlegt, die UID als Ganzzahl. Alle anderen Werte werden als Strings gespeichert.

Lediglich der *Schlossort* sowie die *Punkte* beziehen sich auf tatsächliche Spielinhalte. Als Schlossort werden die Koordinaten gespeichert, die den Ausgangspunkt für das Gitternetz im Spiel darstellen. Diese werden als ein Paar von Gleitkommazahlen gespeichert. Die Spalte „Punkte“ enthält die aktuelle Punktezahl des Spielers, gespeichert als Ganzzahl.

Die UID dient als Fremdschlüssel in der *Messwertetabelle*. Ihr Aufbau ist in Abb. 7.2 aufgezeigt. Auch ist *Nahrungsmittel* in der Darstellung enthalten. Diese Tabelle beinhaltet *Name* und *Menge* eines Nahrungsmittels, das durch einen Fremdschlüssel *Messwert* mit der Messwert-ID verknüpft ist. Dies wurde so realisiert, um mehrere Nahrungsmittel mit verschiedenen Mengenangaben zu einem Messwert zuzuordnen. Name wird als String hinterlegt, Menge und ID sind Ganzzahlen.

Auch die Einträge der Messwertetabelle verfügen über eine *ID*. Sie wird, ebenso wie die User-ID, als Ganzzahl gespeichert. Der gemessene *Blutzuckerwert* ist als Gleitkommazahl hinterlegt. Der *Zeitpunkt* der Messung dagegen ist eine Zeichenkette, die als Date-Objekt im Format „yyyy-MM-dd HH:mm:ss“ interpretiert werden kann. Bei *gespeichert* handelt es sich um ein Flag, das gesetzt wird, sobald der zugehörige Messwert in eine externe Datenbank gespeichert wurde. Dadurch kann sichergestellt werden, dass nur solche Messwerte aus dem lokalen Speicher gelöscht werden, die bereits extern gesichert sind. Die Löschung ist notwendig, damit Candy Castle trotz häufiger Datenbankzugriffe auch über lange Zeit performant bleibt. Zudem können Mehrfachübertragungen desselben Messwertes an einen externen Speicher verhindert werden.

Aktivität, *Übelkeit* und *Laune* beschreiben das Befinden des Nutzers. Aktivität beschreibt das Maß an körperlicher Anstrengung, Übelkeit beschreibt eventuelles Unwohlsein, das beispielsweise durch niedrige Blutzuckerwerte auftreten kann. Laune beschreibt das Wohlbefinden im Allgemeinen. Diese drei Werte werden als Ganzzahlen zwischen 0 und 2 gespeichert, wobei 0 für einen niedrigen, 2 für einen hohen Wert steht.

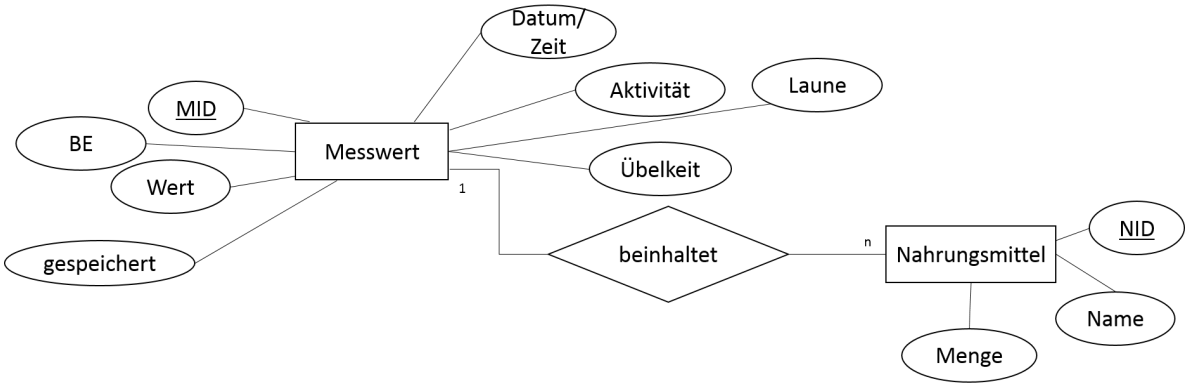

Abbildung 7.2: Die Messwerttabelle als ER-Diagramm

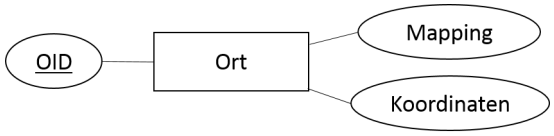

Abbildung 7.3: Die Ortstabelle als ER-Diagramm

Die Messwertetabelle enthält zudem auch die Zahl der konsumierten *BE* als Gleitkommazahl. Zusammen mit den Informationen aus der Nahrungsmittel-Tabelle können so die Essgewohnheiten des Nutzers analysiert werden.

Zuletzt fehlt die Speicherung des *Ortes*. Diese ist in Abb. 7.3 sichtbar. Neben der Orts-ID (*OID*) enthält diese Tabelle die GPS-Koordinaten als Paar von Gleitkommazahlen sowie die Nummer der Zelle, auf die der Ort im Spiel abgebildet wird. Somit kann bei einer Ortsabfrage durch die App die bereits berechnete Abbildung zurück gegeben werden.

Abb. 7.4 zeigt den Aufbau der Datenbank im Ganzen.

7.2 Externe Services und die Kommunikation mit diesen

Um eine volle Funktionalität der App zu erreichen, muss auf einige externe Services zurückgegriffen werden. Hierbei handelt es sich um Webservices, die verwendet werden, um Daten abzurufen sowie langfristig zu speichern.

Zunächst wird ein solcher Service benötigt, um die BE abzurufen. Dies kann auf mehrere Wege geschehen. Momentan ist im Konzept eine manuelle Eingabe des Produktnamens und der Menge vorgesehen, die dann als JSON-Objekte (JSON: JavaScript Object Notation [w3s15]) mittels REST-Calls (REST: Representational State Transfer [Elk15]) an eine externe Datenbank gesendet werden (Beispiel in Listing 7.1). Der Wert der *BU* (Bread Units, englisch für Broteinheit) ist hierbei noch auf null gesetzt, da er durch den Webservice angepasst wird. In der realen Implementierung würde hierfür eine sichere Verbindung verwendet werden

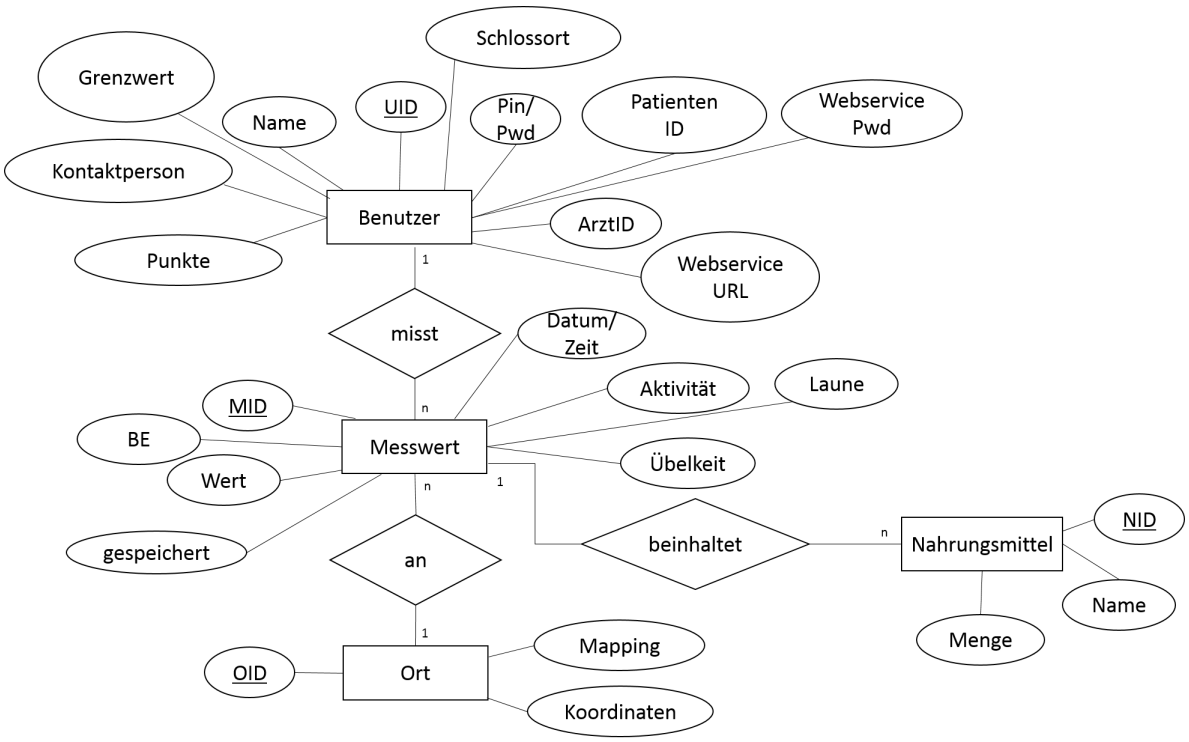

Abbildung 7.4: Der Gesamtaufbau der Datenbank als ER-Diagramm

Listing 7.1 Das Modell des JSON-Objektes zur BE-Abfrage für zwei Scheiben Roggenbrot

```
{"food":[
      {"name":"Roggenbrot",
      "amount":"2",
      "bu":"0"}
]}
```

(z.B. [AYA14]), für den Prototypen wird allerdings auf weitergehende Sicherungsmechanismen verzichtet. Der Service füllt dann das noch leere Feld *BE* aus, wobei der dort eingetragene Wert bereits mit der Menge multipliziert wurde. Dies ist möglich, da BE nicht pro Gramm oder ml in Tabellen abgelegt sind, sondern pro Portion (vgl. [LS12]). Das Nahrungsmittelobjekt wird dann wieder an die App zurückgesendet, wo die Daten weiterverwendet werden können.

Eine andere Möglichkeit, die für den Nutzer handlicher wäre, stellt das Fotografieren des Barcodes eines Lebensmittels dar. Statt den Namen an eine externe Datenbank zu senden, würde die Information des Codes verschickt werden. Dies würde eine genauere Einschätzung der konsumierten BE erlauben. Die Umsetzung kann mithilfe der ZXing-Bibliothek [Ope15a] für Java vorgenommen werden. Allerdings wäre hierfür eine entsprechende Datenbank notwendig, weshalb für den Prototypen auf die erste Variante zurückgegriffen werden soll.

Als zweiter Webservice wird eine Datenhaltung benötigt, die es dem Arzt erlaubt, auf die gespeicherten Daten zuzugreifen und sie zu analysieren. Hierfür eignet sich eine Plattform, die in Aufbau und Funktion ECHO [SWM+15] gleicht. Der Webservice muss eine Zuteilung zwischen Patient und Arzt erlauben. Zudem muss er die Möglichkeit bieten, die von Candy Castle erfassten Daten in ihrer Gesamtheit zu speichern und zu analysieren. Neben einem Langzeitzuckerwert sollten hierbei auch die Zusammenhänge zwischen Ort und Messwert betrachtet werden können. Der Arzt sollte zudem die Möglichkeit haben, weitere Patienten anzulegen und zu verwalten.

Um mit dem Service kommunizieren zu können, wird eine Art der Authentifizierung benötigt. Diese muss bei der Konfiguration Candy Castles eingestellt werden. Sie sollte aus einer Patienten-ID sowie einer Arzt-ID und einem Passwort bestehen. Zudem muss es die Möglichkeit geben, Datenobjekte in die Datenbank zu laden, ohne sich aktiv anzumelden. Dies kann über Berechtigungsnachweise (Authorization-Credentials) im HTTP-Request Header ermöglicht werden. Nach der Authentifizierung können dann die Messwertobjekte als JSON-Objekte an den Service gesendet werden. Auch hier sind weitere Sicherheitsmaßnahmen möglich (vgl. [MSS+14, AYA14]), im Prototypen jedoch nicht umgesetzt.

7.3 Die Ressourcen

Wie bereits in Kapitel 5 beschrieben, implementieren die Ressourcen die Funktionen, die ein Service Feature nutzen möchte. Sie beinhalten all die Funktionen der App, die sensible Daten verwalten oder mit ihnen arbeiten. Zunächst muss festgestellt werden, welche Ressourcen für Candy Castle benötigt werden. Dies geschieht in Abschnitt 7.3.1. In Abschnitt 7.3.2 wird anschließend deren Realisierung näher erläutert.

7.3.1 Erfassung der benötigten Ressourcen

Anhand Tabelle 6.1 sollen nun die Ressourcen herausgearbeitet werden. Hierzu werden all jene Service Features betrachtet, die mit sensiblen Daten arbeiten. Da sich die Ressourcen auf Systemfunktionen beziehen, sollen diese anschließend für jedes Feature festgestellt werden. Tabelle 7.1 zeigt diese Zuordnung.

Zusätzlich müssen einige Eingaben durch eine Ressource geschützt werden. Hierzu gehört die Eingabe des Messwertes. Zwar wird eine manuelle Eingabe durch die App verwaltet, jedoch wäre es möglich, wie in mySugr Pro [myS14b] ein Bluetooth-fähiges Messgerät zu verwenden und die Daten so zu erfassen. Die Sicherheit der Übertragung kann dabei durch verschiedene Maßnahmen erreicht werden (z.B. *Amulet* [SSP+12]). Dies ist für den Prototypen nicht geplant. Die Messwertressource wird somit die Messwertobjekte während der Eingabe verwalten und anschließend an die Datenbank senden.

Service Feature	Systemfunktion
Überprüfung der Anmeldedaten	Datenbankzugriff (lesend)
Speicherung der Messwerte	Datenbankzugriff (schreibend)
Upload der Messwerte	Datenbankzugriff (lesend), Internetnutzung
Personalisieren	Datenbankzugriff (schreibend)
Abruf der BE von einem Webservice	Internetnutzung
Blutzucker eintragen	Datenbankzugriff (schreibend)
Analyse des Messwerts	Datenbankzugriff (lesend)
Alarmierung der Kontaktperson	Datenbankzugriff (lesend), SMS-Versand
Laune eintragen	Datenbankzugriff (schreibend)
Broteinheiten eintragen	Datenbankzugriff (schreibend)
GPS-Abfrage	GPS
Eintragung der Orte in die Karte	Datenbankzugriff (lesend)
Zeit-Abfrage	Zeitabfrage

Tabelle 7.1: Die benötigten Systemfunktionen pro Service Feature

Auch das Personalisieren sollte durch eine Ressource geschützt sein, da hier mit dem Grenzwert und den Daten für den Webservicezugriff einige sensible Daten anfallen. Hier könnte man eine externe Eingabemaske nutzen, die die eingegebenen Daten an die Ressource schickt. Die Ressource leitet diese dann an die Datenbank weiter. Für den Prototypen soll allerdings zunächst nur ein Platzhalter implementiert werden. Aus der Tabelle und dem oben Genannten ergeben sich folgende Ressourcen:

Datenbankressource: Stellt alle Interaktionen mit der Datenbank zur Verfügung (anlegen, lesen, schreiben). Zudem sollen Messwerte hier vor der Speicherung analysiert werden. Erhöhte Sicherheit kann beispielsweise durch die Nutzung des sicheren Datencontainers (*SDC*) [SM15] erreicht werden.

Ressource für Service-Kommunikation: Ist zuständig für den Aufbau einer Verbindung zu einer bestimmten URL, versendet und empfängt JSON-Objekte. Hierbei werden die Verbindungsziele durch den Nutzer festgelegt.

SMS-Ressource: Ist zuständig für den Versand von SMS, die die eingestellte Kontaktperson bei zu niedrigem Blutzucker des Nutzers warnen.

GPS-Ressource: Ruft den GPS-Wert ab.

Zeitressource: Ruft den Zeitstempel ab.

Messwertressource: Erfassung und Verwaltung der Messwerte. Die in der App eingegebenen Werte werden hier zu einem Messwerteobjekt zusammengefügt.

Konfigurationsressource: Zuständig für das Personalisieren, zunächst Platzhalter.

Listing 7.2 Measurement-Objekte speichern die Daten zu Messwerten

```
public class Measurement implements Parcelable {

   private long id;
   private long userID;
   private double bloodSugarLevel;
   private boolean saved;
   private String timestamp;
   private ArrayList<String> food;
   private ArrayList<Integer> foodQuantity;
   private int activity;
   private int mood;
   private int nausea;
   private double be;
   private double longitude;
   private double latitude;

      ...}
```

Dabei sollen alle Funktionen der Ressourcen aktivierbar und deaktivierbar sein, außer die in Abschnitt 6.3 ausgenommenen. Die GPS-Ressource soll zusätzlich die Möglichkeit bieten, die Genauigkeit der Standorterfassung einzustellen.

7.3.2 Realisierung der Ressourcen

Da die Ressourcen nicht untereinander direkt kommunizieren können, sondern alle Parameter und Rückgabewerte über die App senden, sind Methoden zur Ver- und Entschlüsselung der so gesendeten Daten notwendig. Für den Prototyp werden diese eingebaut, jedoch nicht implementiert. Sie geben lediglich das erhaltene Objekt wieder zurück. Für die Verschlüsselung können Sealed Objects [Ora15] verwendet werden.

Zum Austausch zwischen den Ressourcen werden vier verschiedene Java-Objekte verwendet. Diese implementieren alle das Parcelable-Interface und zwei Methoden *seal()* und *unseal()* für die Erstellung von Sealed Objects bzw. die Entschlüsselung dieser. Sie sind in Listing 7.2, Listing 7.3, Listing 7.4 und Listing 7.5 dargestellt.

Zusätzlich enthalten die Ressourcen die verschiedenen Methoden, die sie benötigen, um ihre Arbeit auszuführen. Dabei sind nicht alle Methoden nach außen hin ausführbar. Die Methoden, die die App verwenden kann, werden in einer *AIDL*-Datei (AIDL: Android Interface Definition Langauge [And15]) aufgelistet. Diese beschreibt das Interface, über das die App die Ressource nutzen kann.

Listing 7.3 Config-Objekte repräsentieren die Konfigurationseinstellungen für einen Nutzer

```
public class Config implements Parcelable{

      private long id;
      private String name;
      private String pwd;
      private String contactNumber;
      private String patientID;
      private String doctorID;
      private String webservicePwd;
      private String webserviceUrl;
      private double castleLong;
      private double castleLat;
      private int points;
      private double limit;

      ...}
```

Listing 7.4 Food-Objekte stellen einzelne Nahrungsmittel dar

```
public class Food implements Parcelable {

   private String name;
   private long be;

      ...}
```

Die Ressourcen selbst bestehen aus einer richtigen Implementierung, sowie einer *Cloak*- und einer *Mock*-Implementierung, die genutzt werden, falls die App nicht die benötigten Rechte besitzt. In diesem Fall werden feste Werte zurückgegeben.

Im Rahmen dieses Prototyps mussten nicht alle Ressourcen neu implementiert werden. Die Ressource zur GPS-Abfrage sowie die Datenbankressource waren bereits aus früheren Projekten gegeben. Dabei musste die Datenbankressource auf das in Abschnitt 7.1 vorgestellte Datenbankmodell abgestimmt werden. Alle anderen Ressourcen wurden für den Prototypen neu implementiert.

Listing 7.5 SMS-Objekte werden an die SMS-Ressource zum Versand weitergeleitet

```
public class Sms implements Parcelable {

   private String number;
   private String user;
   private boolean send;

      ...}
```

In den folgenden Abschnitten sollen die Funktionen der Ressourcen genauer beschrieben werden. Auch sollen die Schnittstellen zur App klar definiert werden.

Datenbankressource

Die Datenbankressource ist für die verschiedenen Interaktionen zwischen App und Datenbank zuständig. Diese beinhalten das Anlegen einer Datenbank, sowie jede schreibende oder lesende Interaktion. Die Datenbankressourcewar zu Beginn der Arbeit bereits vorgegeben. Allerdings müssen für die Verwendung in Candy Castle noch Änderungen vorgenommen werden.

Bei den von der Ressource angelegten Datenbanken handelt es sich um SQlite-Datenbanken. Die Abfrage von Werten kann somit als SQL-Query erfolgen.

Für Candy Castle muss zunächst die Struktur der angelegten Datenbank an das in Abschnitt 7.1 vorgestellte Datenmodell angepasst werden. Zudem werden Methoden eingefügt, die übergebene Objekte ver- und entschlüsseln kann. Diese werden benötigt, um beispielsweise Messwertobjekte verschlüsselt über die App versenden zu können. Wichtig ist hierbei, dass die Datenbank nur dann Werte aktualisiert, wenn sie ein solches verschlüsseltes Objekt erhalten hat. In anderen Fällen wird die Anfrage verworfen. So kann die App nicht selbstständig Werte in der Datenbank verändern, wie beispielsweise Konfigurationsdaten. Die von der Ressource angebotenen Funktionen sind in Listing 7.6 aufgezeigt.

Deutlich wird hierbei, dass die App nur sehr spezifische Möglichkeiten hat, mit der Datenbankressource zu interagieren. So legt *createTables()* genau die in Abschnitt 7.1 vorgestellten Tabellen an. Nicht alle Methoden sind dabei dazu gedacht, der App direkt Werte zu liefern. Während beispielsweise *getPoints()* die Punktezahl direkt an die App zurückmeldet, sind die Methoden *insertMeasurement(Measurement ms)*, *analyze(Measurement ms)*, *setConfig(in Config config)*, *getForUpload()* und *uploaded(boolean worked)* für die Kommunikation mit anderen Ressourcen vorgesehen. Hier werden also nur Sealed Objects verschickt und akzeptiert. Während die meisten der gezeigten Methoden in die Datenbank schreiben bzw. aus ihr lesen, verfügen *analyze(Measurement ms)*, *uploaded(boolean worked)* und *getAttacked(String name)* über weitere Funktionalität. *uploaded(boolean worked* setzt das Saved-Flag aller Messwerte, die zuvor von *getForUpload()* ausgelesen wurden, auf den Wert von worked. *analyze(Measurement ms* vergleicht den Blutzuckerwert des Parameters mit dem eingestellten Limit des Nutzers und liefert ein SMS-Objekt zurück, welches durch einen booleschen Wert angibt, ob eine SMS an die Kontaktperson gesendet werden soll. *getAttacked(String name)* schließlich wird einmal täglich aufgerufen und senkt die Punktezahl des angegebenen Nutzers um 25 Punkte.

insertMeasurement(Measurement ms berechnet zudem für jedes GPS-Datum, das empfangen wird, eine Abbildung und speichert diese. Wird der Wert 200 empfangen, so ist die

Listing 7.6 Die AIDL-Schnittstelle der Datenbankressource

```
interface IDatabaseConnection {

   boolean createTables();

   int insertMeasurement(in Measurement ms);

   Sms analyze(in Measurement ms);

   List<Measurement> getForUpload();

   boolean uploaded(boolean worked);

   boolean setConfig(in Config config);

   Config getConfig();

   boolean isDBAllowed();

   long getUserID(String name);

   int getPoints(String name);

   void getAttacked(String name);

   boolean checkAuthorization(String name, String pwd);

   boolean deleteOldMeasurements();
}
```

GPS-Ressource deaktiviert und es werden zufällige Abbildungswerte berechnet. Diese Abbildungswerte werden an die App zurückgemeldet und dort gespeichert.

Der Nutzer kann drei Privacy Settings aktivieren oder deaktivieren. Diese drei Privacy Settings beziehen sich auf das Anlegen von Datenbanken, das Auslesen von Datenbanken, sowie das Modifizieren von Datenbanken. Um Candy Castle nutzen zu können, ist es nötig, alle drei Settings zu aktivieren. Ansonsten ist die App nicht in der Lage, Messwerte zu speichern, womit ihre Hauptfunktion nicht mehr erfüllbar wäre.

Zusätzlich wird ein weiteres Setting eingeführt, dass die Analyse von Messwerten erlaubt oder verbietet. Ist dieses Setting deaktiviert, wird der SMS-Ressource immer zurückgemeldet, dass keine SMS versendet werden soll. Diese Funktion kann aktiviert oder deaktiviert werden, ohne dass die Hauptfunktion der App beeinträchtigt wird.

Listing 7.7 Die AIDL-Schnittstelle der Webserviceressource

```
interface IWeb {

      Food getBU(in Food food);

      boolean uploadMeasurements(in List<Measurement> measurements, in Config config);

}
```

Service-Kommunikation

Im Gegensatz zur allgemeinen Internetnutzung soll diese Ressource lediglich den Zugriff auf zwei Webservices beinhalten. Bei diesen handelt es sich einmal um die externe Datenbank zur BE-Abfrage, zum anderen um die Plattform zur Messwertverwaltung und -analyse durch den behandelnden Arzt. Darum sind in der Ressource genau diese beiden Fälle implementiert. Somit kann gewährleistet werden, dass die App die Ressource nicht benutzen kann, um anderweitig Daten an fremde Internetseiten zu versenden. Die von ihr angebotenen Methoden sind in Listing 7.7 dargestellt.

getBU(in Food food) ruft einen fest eingestellten Dienst auf, an den das beim Aufruf übergebene Food-Objekt als JSON-Objekt versendet wird. Als Antwort erhält die Methode ein JSON-Objekt mit aktualisiertem BE-Wert zurück, das sie als Food-Objekt an die App sendet. Dort kann der BE-Wert angezeigt werden. Wird diese Funktion mithilfe von Barcodes realisiert, so können Nahrungsmittel und Broteinheiten vor der App geheimgehalten werden, da keine textuelle Benutzereingabe mehr vonnöten ist.

Die zweite Methode der Webserviceressource erhält von der Datenbank eine Liste mit Messwerten, die es an die in der Konfigurationsressource festgelegte Plattform zu senden gilt. Hierzu werden die in der Konfiguration festgelegten Daten benötigt. Der Rückgabewert der Methode beschreibt den Erfolg des Versands und wird an die *uploaded(boolean worked)*-Methode der Datenbankressource als Parameter übergeben.

Die Kommunikationsressource verfügt über zwei boolesche Privacy Settings. Diese beziehen sich jeweils auf die BE-Abfrage bzw. den Messwerte-Upload. Der Nutzer kann beide Settings deaktivieren, ohne dass die Grundfunktion von Candy Castle beeinträchtigt wird. Ist die BE-Abfrage deaktiviert, wird auch die Eingabemaske in der App deaktiviert. Die BE können trotzdem vom Nutzer per Hand eingegeben werden.

SMS-Ressource

Bei jeder neuen Messung erhält die SMS-Ressource von der Datenbank ein SMS-Objekt, welches den Benutzernamen, die Nummer der Kontaktperson sowie eine Variable enthält, ob die SMS gesendet werden soll. Diese Variable dient dazu, den Gesundheitszustand des

Listing 7.8 Die AIDL-Schnittstelle der SMS-Ressource

```
interface ISMS{

      void sendSMS(in Sms sms);
}
```

Listing 7.9 Die AIDL-Schnittstelle der GPS-Ressource

```
interface IAbsoluteLocation {

      Measurement addPos(in Measurement ms);
}
```

Nutzers vor der App zu verschleiern, da *analyze(in Measurement ms)* immer ein SMS-Objekt zurückgibt, welches weitergeleitet werden muss. Der Versand dieser SMS an die gegebene Telefonnummer ist die einzige Funktion der Ressource (siehe Listing 7.8). Eine Warn-SMS enthält keine konkreten Daten, sondern lediglich den Hinweis, dass der Grenzwert für Blutzuckerwerte unterschritten wurde.

Die SMS-Ressource verfügt über ein boolesches Privacy Setting, das den Versand einer solchen SMS aktiviert oder deaktiviert. Dies kann nach Belieben eingestellt werden, ohne die Funktion der App zu beeinträchtigen.

GPS-Ressource

Auch diese Ressource stand zu Beginn der Arbeit bereit. Für die Nutzung von Candy Castle sind jedoch nicht alle gegebenen Funktionen vonnöten. Während die Ressource auch Geschwindigkeit u.a. angeben kann, benötigt Candy Castle lediglich die Möglichkeit, Koordinaten abzurufen. Alle anderen Funktionen der Ressource werden darum entfernt, um den Nutzer keine unnötigen Einstellungen vornehmen zu lassen (siehe Listing 7.9).

Nach diesem Schritt bietet die Ressource nur noch die Methode *addPos(in Measurement ms)*. Diese erhält ein verschlüsseltes Measurement-Objekt, welches entschlüsselt und um die momentane Position erweitert wird. Anschließend wird es wieder verschlüsselt und für die weitere Verwendung zurückgeschickt.

Die Ressource bietet für Candy Castle drei Privacy Settings. Mithilfe von Booleans kann festgelegt werden, ob GPS-Daten abgerufen werden dürfen und ob diese weiter verwendet werden dürfen. Zusätzlich kann der Nutzer einen Ungenauigkeitswert für die abgefragten Koordinaten eingeben. Je höher dieser Wert, desto unpräziser sind die zurückgegebenen Koordinaten.

Listing 7.10 Die AIDL-Schnittstelle der Zeitressource

```
interface ITimestamp {

      Measurement getTimestamp(in Measurement ms);

      String getCurrentTime();
}
```

Die GPS-Ressource kann nach Belieben aktiviert oder deaktiviert werden, ohne, dass die Grundfunktion des Spiels beeinflusst wird. Ist die Ressource deaktiviert, werden für Länge und Breitengrad jeweils 200 zurückgegeben.

Zeitressource

Ähnlich wie in der GPS-Ressource wird auch hier ein verschlüsseltes Measurement-Objekt an *getTimestamp(in Measurement ms)* übergeben und mit Zeitstempel wieder zurückgeschickt (siehe Listing 7.10). Zudem kann die App selbst mit *getCurrentTime()* auf die aktuelle Zeit zugreifen, beispielsweise um den Punkteabzug auszuführen.

Mit dem einzigen Privacy Setting kann der Nutzer die Zeitabfrage aktivieren oder deaktivieren. Die Funktion der App bleibt dadurch erhalten.

Messwertressource

In der Messwertressource wird ein neues Objekt der Klasse Messwert erzeugt und mit Werten belegt. Die Ressource bietet hierzu *createMeasurement(double bsl, double be, int activity, int mood, int nausea)* an. Der so erstellte Messwert wird dann an die GPS- oder Zeitressource weitergeleitet, um die restlichen Werte zu setzen. In zukünftigen Arbeiten soll diese manuelle Eingabe der Werte durch eine automatische Erfassung des Blutzuckerwertes ersetzt werden. Hierfür kann via Bluetooth ein Messgerät verknüpft und ausgelesen werden.

Wie in Listing 7.11 dargestellt, verfügt die Ressource über vier weitere Methoden. *isXXAllowed()* liefert einen booleschen Wert zurück, der beschreibt, ob der angefragte Wert in den Messwert eingetragen werden darf. Falls nicht, können die entsprechenden Eingabefelder deaktiviert werden.

addFood(String name, int amount, in Measurement ms) wird aufgerufen, wenn der Nutzer ein Nahrungsmittel zum Abruf der BE in die entsprechende Eingabemaske eingetragen hat. Hiermit wird ein neues Food-Objekt erstellt und in die Liste des Measurement-Objektes eingefügt.

Listing 7.11 Die AIDL-Schnittstelle der Messwertressource

```
interface IMeasurement {

      Measurement createMeasurement(double bsl, double be, int activity, int mood, int
          nausea);

      Measurement addFood(String name, int amount, in Measurement ms);

      boolean isBSLAllowed();

      boolean isBEAllowed();

      boolean isCondAllowed();
}
```

Listing 7.12 Die AIDL-Schnittstelle der Konfigurationsressource

```
interface IConfig {

      Config createConfig(String name, String pwd, String contact, String patientID,
          String doctorID,
                  String webPwd, String webURL, double castleLongitude, double
                      castleLatitude, double limit);

      boolean isConfigAllowed();
}
```

In dieser Ressource können die verschiedenen Attribute des Messwerts deaktiviert werden. Diese Attribute sind Laune, Aktivität, Übelkeit, BE und der Blutzuckermesswert. Ist die Speicherung eines Attributs deaktiviert, so füllt die Ressource einen negativen Wert an seiner Stelle ein. Ist allerdings die Speicherung des Blutzuckermesswerts deaktiviert, so kann Candy Castle seine Grundfunktion nicht mehr erfüllen. Die anderen Attribute sind beliebig aktivierbar und deaktivierbar.

Konfigurationsressource

In dieser Ressource werden die Konfigurationsdaten festgelegt und ein neues Config-Objekt erstellt. Dieses wird verschlüsselt an die App gesendet, welche dann die Eintragung in die Datenbank veranlasst. Zudem beinhaltet die Ressource die Methode *isConfigAllowed()* (siehe Listing 7.12), welche der App zurückgibt, ob sich die App konfigurieren lässt. Diese Methode wird benötigt, da durch die Deaktivierung der Ressource Privatheit verloren gehen kann (siehe Abschnitt 6.3).

Für den Prototypen wird hierbei eine festgelegte Konfiguration eingestellt. In späteren Erweiterungen kann die Konfiguration allerdings extern festgelegt werden, beispielsweise über einen Webservice.

Der Nutzer kann hier verschiedene Funktionen aktivieren oder deaktivieren. Diese sind die Registrierung eines Nutzers, das Einrichten eines Webservice sowie die Speicherung eines Grenzwertes. Bei Deaktivierung werden *null*-Objekte in die Datenbank eingetragen. Dies hat keinen Einfluss auf die Grundfunktion der App.

7.4 Die App

Nachdem nun die benötigten Ressourcen und ihre Schnittstellen bekannt sind, soll die Realisierung der App selbst beschrieben werden.

Da die Berechtigungen in der PMP jederzeit neu gesetzt werden können, müssen die Datenbankberechtigungen sowie die Berechtigung zur Blutzuckerwerteingabe bei jedem Start der App überprüft werden. Sind diese nicht gegeben, so gibt die App eine entsprechende Meldung aus.

Beim Start wird zunächst der Authentifizierungsbildschirm angezeigt. Ist diese Funktion in der Konfigurierung deaktiviert, so wird der Nutzer darauf hingewiesen, diese zu aktivieren. Dies dient dem Schutz vor physischen Angriffen.

Der folgende Bildschirm zeigt die Karte des Königreichs. Das Schloss befindet sich bereits dort, da der Ort des Schlosses in der Konfiguration angegeben wurde. Alle weiteren Orte werden in Relation zu diesem auf der Karte abgebildet. Die Karte ist dabei in 18 Gitterzellen unterteilt, wobei die des Schlosses sowie die obersten drei nicht nutzbar sind. Die Einteilung des Gitters ist in Abb. 7.5 dargestellt.

Die App fragt bei jedem Aufruf des Kartenbildschirms die Gitterzellen der aktiven Orte ab und fügt dort ein Bild ein. Eine Gitterzelle hat dasselbe Bild, bis der Ort als inaktiv gekennzeichnet wird. Hierfür werden Mapping-Objekte verwendet, die in Listing 7.13 dargestellt sind. Erst danach kann bei erneuter Aktivierung des Ortes ein neues Bild zugewiesen werden. *image* bezeichnet dabei die ID des anzuzeigenden Bildes. Die Mapping-Objekte werden in einem Array verwaltet, wo sie ersetzt werden, sobald ihr *counter* 0 erreicht.

Wie bereits erwähnt verändert sich die Darstellung des Schlosses je nach Punktezahl des Nutzers. Die Punkte berechnen sich wie folgt: Die maximale Punktezahl beträgt 90 Punkte. Für jeden Messwert erhält der Nutzer 10 Punkte, für das Eintragen von Laune und BE jeweils 5 Punkte. Das Eintragen eines Ortes gibt nur dann Punkte, wenn der Ort als inaktiv gekennzeichnet ist. In diesem Fall erhält der Spieler 5 Punkte.

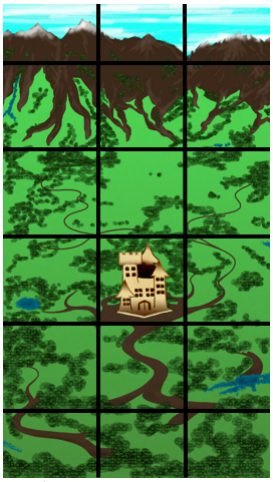

Abbildung 7.5: Die Gittereinteilung der Karte

Listing 7.13 Die Definition der Mapping-Objekte

```
private class Mapping {

                private int image;

                private int mapping;

                private int counter;

                ...}
```

Das Schloss erscheint als intakt, wenn der Spieler zwischen 60 und 90 Punkte besitzt. Die zweite Stufe wird bei 30 bis 59 Punkten angezeigt, die letzte bei 1 bis 29 Punkten. Bei 0 Punkten ist kein Schloss mehr zu sehen, sondern es erscheint eine Meldung, dass das Spiel verloren ist. Ein Angriff der Dunklen Mächte zieht dem Spieler 25 Punkte ab. So kommt man bei drei täglichen Messungen ohne Ortsangabe nicht unter 60 Punkte, wodurch das Schloss weiterhin als intakt erscheint. Dies soll gewährleisten, dass sich der Spieler trotz ausreichender Messhäufigkeit nicht demotiviert fühlt.

8 Evaluation

Nun, da das neue Konzept und die Umsetzung Candy Castles vorgestellt wurden, gilt es die App zu evaluieren. Hierfür werden Kapitel 2, Kapitel 3 und Kapitel 4 herangezogen. Durch die Evaluierung soll gezeigt werden, dass die in Kapitel 4 aufgestellten Anforderungen erfüllt wurden. Auch sollen die Vorteile des neuen Konzepts gegenüber anderen mHealth-Apps sowie der alten Versionen von Candy Castle aufgezeigt werden. Ein Überblick hierüber ist in Abschnitt 8.4 dargestellt.

8.1 Verwandte Arbeiten

Im Gegensatz zum allgemeinen Berechtigungssystem auf Android-Geräten bietet Candy Castle im Verbund mit der PMP einen hohen Grad an Nutzerbestimmung. Nahezu jede Funktion der App kann beliebig aktiviert oder deaktiviert werden, sodass es dem Nutzer selbst überlassen bleibt, welche Rechte er der App einräumt. Auch wird durch die Separierung der einzelnen Service Features klar, welche Funktionen der App die entsprechenden Rechte benötigen. Dies ist bei vielen anderen Apps unklar. Die Verbindung mit der PMP fördert somit nicht nur die Nutzerbestimmung, sondern auch die -information.

Bei Candy Castle handelt es sich um ein verarbeitendes Spiel (siehe Abschnitt 2.2). Es werden also nicht nur Informationen an den Spieler weitergegeben, sondern auch Daten über ihn gesammelt und ausgewertet. Zwar fehlt bislang die in mySugr Junior [myS14a] vorhandene Datenschutzerklärung, diese lässt sich jedoch jederzeit in das Konzept integrieren. Anders als in mySugr Junior und Wind Runners [NPS+12] verwaltet die App die Daten jedoch nicht selbst. Während Erstgenannte dadurch für kompromittierten Code anfällig sind, schützen bei Candy Castle die Ressourcen die Daten vor unerlaubtem Zugriff.

Die unter Abschnitt 2.3 aufgezeigten Lösungsansätze wurden bei der Erstellung des Konzepts mit einbezogen. Eine Kapselung der Daten (vgl. [WRR08]) kann durch die Kopplung der PMP mit dem Secure Data Container [SM15] erreicht werden. Das Prinzip von ECHO [SWM+15] wurde als Grundlage für den Aufbau eines externen Webservice verwendet.

Auch viele der in Abschnitt 2.4 gelisteten Kriterien der verschiedenen Checklisten wurden erfüllt. Zwar besitzt Candy Castle bislang weder eine Seite im App-Store, noch verfügt die App über eine Datenschutzerklärung. Diese Punkte sind allerdings leicht zu beheben.

Das Erstkonzept Candy Castles [Knö10] entstand in Zusammenarbeit mit Fachkräften des Olgahospitals in Stuttgart. Auch Kinder waren miteinbezogen. Candy Castle bietet zudem individuelle Einstellungsmöglichkeiten sowohl im Bereich App-Nutzung (Grenzwert, Kontaktperson, etc.) als auch bei den Berechtigungen. Zudem stellt die App weder Diagnosen, noch hat sie den Anspruch, einen Arztbesuch zu ersetzen. Im Gegenteil erleichtert sie die kontinuierliche Zusammenarbeit zwischen Arzt und Patient. Damit bietet sie auch den von Vital [VIT12] geforderten Mehrwert.

In Abschnitt 2.5 werden verschiedene Probleme von mHealth-Apps angesprochen, wie beispielsweise fehlende Datensicherheit bei Übertragungen via Internet. Auch hier sind im Konzept Candy Castles Sicherheitsvorkehrungen vorgesehen, wie beispielsweise Verschlüsselung oder Authentifizierung. Somit können durch eine Erweiterung der App auch diese Probleme gelöst werden.

Im Vergleich zu bestehenden mHealth-Apps erfüllt Candy Castle also bereits als Prototyp einige der Forderungen. Vor allem im Bereich Nutzerbestimmung hat die App deutliche Vorteile gegenüber vergleichbaren mobilen Anwendungen. Punkte, wie zum Beispiel die Sicherung von Datenübertragungen, sind im Prototyp zwar nicht umgesetzt, im Konzept aber durchaus vorgesehen. Andere Forderungen, wie das Vorhandensein einer Datenschutzerklärung oder eines Impressums, können ohne großen Aufwand in das Konzept eingefügt werden.

8.2 Frühere Versionen

Anders als im Grundkonzept von M. Knöll [Knö10] ist Candy Castle nicht mehr auf die gemeinschaftliche Nutzung von Eltern und Kind ausgelegt. Zwar werden die Eltern eingebunden, das Spiel selbst ist allerdings auf einen Spieler ausgelegt. Dies hat mehrere Gründe. Zum einen antworteten in der von M. Knöll geführten Umfrage nur drei von zwölf Kindern, dass sie gerne mit ihren Eltern zusammen spielen würden. Zum anderen sollen die Nutzer selbstständig Messungen vornehmen und Tagesabläufe erarbeiten. Die Eltern sollen darum eine unterstützende. aber keine partizipierende Rolle einnehmen.

Trotzdem verfügt Candy Castle über den motivierenden, spielerischen Aspekt, der beiden Vorgängerversionen zugrunde lag. Dabei arbeitet das neue Konzept mit einer bunten Fantasiewelt, die mit Drachen, Schätzen und anderen Elementen angereichert wird. Anders als auf einer Straßenkarte, wie in [SS12] vorgestellt, kann so der spielerische Aspekt weiter hervorgehoben werden.

Auch wurde Candy Castle als native App implementiert, wodurch die Notwendigkeit einer Internetverbindung umgangen wird.

8.3 Anforderungskatalog

Im Folgenden sollen die verschiedenen Anforderungen aus Kapitel 4 wiederholt, und ihre Umsetzung in Candy Castle erläutert werden.

F 1 Der Nutzer soll medizinische Messwerte eingeben können

F 2 Die App soll eingetragene Messwerte langfristig speichern können

Der Nutzer kann über Candy Castle seine Blutzuckerwerte eingeben und speichern. Zudem können Daten zum Befinden und zu den konsumierten Nahrungsmitteln eingetragen werden. Diese Daten, wie z.B. Übelkeit, können dem Arzt bei Diagnosen helfen. Dies ist mithilfe der Messwert– und der Datenbankressource möglich. Über die lokale Datenbank hinaus können zudem die Messwerte in einer externen Datenbank gesichert werden.

F 3 Die App soll dem Nutzer wiederkehrende oder lästige Aufgaben abnehmen und ihn im Alltag unterstützen

Diese Anforderung wird in Candy Castle durch die Möglichkeit zum Abrufen von Broteinheiten erfüllt. Somit wird dem Nutzer eine zeitaufwändige aber wichtige Tätigkeit abgenommen. Zudem kann bei Unterzucker ein Alarm versendet werden, was eine Unterstützung für die Erziehungsberechtigten darstellt.

F 4 Die Messwerte sollen in eine externe Datenbank übertragen werden

F 5 Die externe Datenbank soll dem Arzt in seiner Arbeit unterstützen

Die Übertragung der Messwerte kann an beliebige medizinische Datenbanken erfolgen, die über die nötige Schnittstelle verfügen. Die Datenbank muss in der Lage sein, einen Patienten anhand einer ID einem Arzt eindeutig zuzuordnen, sowie die Messwertobjekte Candy Castles zu verwalten. Da der Arzt seinen bevorzugten Webservice mit Candy Castle verknüpfen kann, erfolgt die Unterstützung automatisch.

F 6 Die App soll Warnungen versenden können

Ist die entsprechende Funktion aktiviert, kann Candy Castle bei Unterzucker Warnungen per SMS an eine zuvor eingestellte Kontaktperson versenden.

F 7 Die App soll personalisierbar sein

Mithilfe der Konfigurationsressource kann ein beliebiger Webservice zur Speicherung der Daten eingestellt werden. Auch kann so ein Grenzwert für den Versand von SMS festgelegt, sowie eine Kontaktperson eingetragen werden.

NF 1 Die App soll motivieren

Für Motivation sorgt in Candy Castle das spielerische Grundkonzept. Hierbei ist es das Ziel des Nutzers, möglichst viele Punkte zu erreichen. Als weitere Motivation könnte das Konzept um beispielsweise Errungenschaften erweitert werden.

NF 2 Die App soll eine ansprechende Darstellung besitzen

Durch die Darstellung als comichafte Fantasiewelt spricht Candy Castle vor allem Kinder an. Die Karte wird durch verschiedene, handgezeichnete Elemente erweitert, welche ebenfalls einen kinderfreundlichen Stil besitzen.

NF 3 Punktevergaben, oder ähnliches, sollen unabhängig von den gemessenen Werten sein

Diese Anforderung wird erfüllt, da Punkte lediglich für Messungen und weiterführende Einträge vergeben werden. Der Blutzuckerwert selbst fließt hierbei nicht mit ein.

NF 4 Die App soll leicht verständlich sein

Eingabefelder in Candy Castle sind durch kurze, prägnante Stichworte bezeichnet. Zudem werden bei der Eingabe von Laune, Übelkeit und Aktivität Bilder als Unterstützung angeboten. Die App verfügt über keine langen Informationstexte, sondern gibt Meldungen als einzelne Sätze aus.

NF 5 Die App soll intuitiv bedienbar sein

Durch die verschiedenen Stadien des Schlosses kann schnell ein Rückschluss auf die momentane Punktezahl gezogen werden. Auch verwendet die App bei der Launenangabe Bilder, um die verschiedenen Werte zu illustrieren.

NF 6 Die App soll keine Diagnosen stellen

Candy Castle selbst führt lediglich eine Analyse durch. Diese beruht auf einem voreingestellten Grenzwert, welcher durch den behandelnden Arzt eingetragen wurde. Somit ist dieser in die Analyse direkt mit eingebunden. Weitere Analysen werden durch den Verknüpften Webservice erstellt, welcher durch den Arzt kontrolliert wird.

NF 7 Die Grundfunktionalität soll jederzeit verfügbar sein

Die Grundfunktionalität Candy Castles besteht darin, Messwerte zu speichern. Da diese Funktion ohne die Verwendung von Internet oder GPS erfüllt werden kann, ist sie überall nutzbar. Wird die Berechtigung zur Erfüllung dieser Funktion entzogen, so wird der Nutzer darauf hingewiesen.

NF 8 Der Nutzer soll selbst bestimmen können, was der App erlaubt ist

NF 9 Der Nutzer soll selbst die über Verwendung seiner Daten bestimmen können

Diese beiden Punkte werden mithilfe der PMP abgedeckt. Der Nutzer kann selbst beliebige Berechtigungen aktivieren oder deaktivieren. Zudem werden alle Daten über Ressourcen verwaltet. Als zusätzliche Sicherheit können die Daten mithilfe des Secure Data Containers [SM15] gespeichert werden.

NF 10 Unbefugte sollen von Außen keinen Zugriff auf die App erhalten

Hierfür muss die Authentifizierung vor Start der App aktiviert sein. Durch die Wahl eines Passworts können Unbefugte die Daten nicht abrufen.

NF 11 Die Messwerte sollen nicht durch den Nutzer verfälscht werden können

Dies kann durch eine Bluetooth-Verbindung direkt zum Messgerät erreicht werden.

NF 12 Es soll ein medizinischer Mehrwert geschaffen werden

Dieser Mehrwert kann durch die Verknüpfung von GPS-Daten und Blutzuckerwerten erreicht werden. Wie bereits von M. Knöll [KM12] erwähnt, liegt hier medizinische Relevanz.

8.4 Überblick

In diesem Abschnitt sollen die beiden Apps „Wind Runners“ [NPS+12], „mySugr Junior“ [myS14a] sowie die drei Versionen von Candy Castle hinsichtlich des Anforderungskatalogs untersucht werden. Dargestellt ist diese Übersicht in Tabelle 8.1. *CC 1* bezeichnet hierbei die Candy Castle-Version von M. Knöll [Knö10], *CC 2* beschreibt die Version von C. Stach und L. F. M. Schlindwein [SS12] und *CC 3* bezeichnet die in dieser Arbeit vorgestellte Version.

Ein Fragezeichen in der Übersicht zeigt an, dass diese Anforderung weder klar erfüllt, noch eindeutig nicht erfüllt ist. Haken in Klammern weisen darauf hin, dass eine Anforderung teilweise erfüllt ist oder, im Fall von CC 2 oder 3, noch erfüllt werden wird.

8.5 Fazit

Das neue Konzept Candy Castles hebt sich vor allem im Bereich Nutzerbestimmung positiv von anderen Apps ab. Der Nutzer hat die Kontrolle über seine Daten. Fehlende Elemente, die in Checklisten der Techniker Krankenkasse oder der DAK aufgeführt sind, können entweder nachträglich eingefügt werden oder sind bereits im Prototyp vorgesehen.

Anforderung	**App**				
	Wind Runners	mySugr Junior	CC 1	CC 2	CC 3
F 1	✓	✓	✓	✓	✓
F 2	✓	✓	✓	✓	✓
F 3	X	✓	X	X	✓
F 4	?	✓	?	✓	✓
F 5	?	✓	?	✓	✓
F 6	X	(✓)	?	✓	✓
F 7	X	✓	X	?	✓
NF 1	✓	✓	✓	✓	✓
NF 2	✓	✓	✓	✓	✓
NF 3	X	✓	X	✓	✓
NF 4	✓	✓	?	✓	✓
NF 5	✓	✓	?	✓	✓
NF 6	✓	✓	✓	✓	✓
NF 7	✓	✓	✓	X	✓
NF 8	X	X	X	(✓)	✓
NF 9	X	?	?	(✓)	✓
NF 10	X	?	?	✓	✓
NF 11	✓	(✓)	X	(✓)	(✓)
NF 12	X	X	✓	✓	✓

Tabelle 8.1: Eine Übersicht über die Erfüllung der verschiedenen Anforderungen in verschiedenen Apps

Die Reimplementierung beinhaltet weiterhin die zentralen Aspekte früherer Versionen Candy Castles, erweitert diese allerdings um weitere Funktionen, die Mehrwert für den Nutzer schaffen.

Die meisten Anforderungen aus Kapitel 4 wurden im Prototyp erfüllt. Für die unerfüllten Anforderungen finden sich im Konzept verschiedene Umsetzungsstrategien.

9 Zusammenfassung und Ausblick

In dieser Arbeit werden bestehende Konzepte für eine mHealth-App erweitert und überarbeitet. Dabei werden besonders Privatheitsaspekte berücksichtigt. Mithilfe verschiedener verwandter Arbeiten wird ein Anforderungskatalog erstellt, anhand dessen ein neues Konzept für das mobile Spiel Candy Castle erarbeitet wird.

Candy Castle fungiert als spielerisches Diabetestagebuch für Kinder. Hierbei liegt besonderes Augenmerk auf der Verknüpfung zwischen Messwert und Ort der Messung. Das Konzept motiviert den Spieler darum dazu, möglichst viele verschiedene Orte für seine Messung aufzusuchen. Die Verknüpfung zwischen Messwert und Ort kann dazu genutzt werden, neue Erkenntnisse über den Einfluss der Umwelt auf Blutzuckerwerte zu erlangen. Auch kann der Nutzer selbst Vorteile aus den gemachten Beobachtungen ziehen. So wird seine Wahrnehmung für Zusammenhänge zwischen seiner Umgebung und seinem Gesundheitszustand gestärkt. Die so gesammelten Daten können an einen in der App einstellbaren Webservice übertragen werden.

Durch die Verbindung der App mit der Privacy Management Platform, kurz PMP, kann der Nutzer beliebige Funktionen der App aktivieren oder deaktivieren. Das Konzept wird darum so ausgelegt, dass möglichst wenige Berechtigungen zur grundlegenden Funktion der App benötigt werden. Der Nutzer hat dadurch jederzeit Kontrolle darüber, welche Funktionen von der App ausgeführt werden können.

Zuletzt wird das Konzept anhand verschiedener Anforderungen evaluiert. Hierbei kann gezeigt werden, dass alle geforderten Punkte im Konzept vorhanden sind oder mit wenig Mehraufwand hinzugefügt werden können.

Durch diese Arbeit wird gezeigt, dass es möglich ist, eine mHealth-App mithilfe der PMP zu implementieren, ohne ihre Grundfunktion zu beeinträchtigen. In Zukunft können weitere Apps mithilfe der PMP implementiert werden, um dem Nutzer ein größeres Bestimmungsrecht über seine Daten zu erteilen. Hierfür können die entwickelten Ressourcen verwendet und angepasst werden.

Auch kann die konzeptionelle Erstellung einer mHealth-App am erarbeiteten Anforderungskatalog orientiert werden.

Mithilfe der durch Candy Castle gesammelten Daten kann weitere Forschung zur Verbindung von Ort und Blutzuckermesswerten durchgeführt werden.

Das hier erstellte Konzept kann auf verschiedene Weisen erweitert werden. So kann die Abfrage von Broteinheiten durch das Scannen eines Barcodes erfolgen. Auch können weitere Sensoren und Messgeräte per Bluetooth eingebunden werden, um die Messungen zu vereinfachen. Zu diesen können beispielsweise Blutzuckermesswerte oder tragbare Computersysteme (*Wearables*) gehören. Um die Daten noch besser zu schützen, können Verschlüsselungsmaßnahmen angewandt werden.

Literaturverzeichnis

[Ama13] Amazon. Nightmare: Malaria. Webseite, 2013. http://www.amazon.com/Psyop-Nightmare-Malaria/dp/B00H5982F4//. (Zitiert auf Seite 16)

[And15] Android Developers. Android Interface Definition Language (AIDL), 2015. http://developer.android.com/intl/zh-cn/guide/components/aidl.html. (Zitiert auf Seite 59)

[AYA14] I. D. Addo, J.-J. Yang, S. I. Ahamed. SPTP: A Trust Management Protocol for Online and Ubiquitous Systems. In *Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference*, COMPSAC '14. IEEE Computer Society, 2014. (Zitiert auf den Seiten 56 und 57)

[BGH+14] M. Backes, S. Gerling, C. Hammer, M. Maffei, P. von Styp-Rekowsky. AppGuard – Fine-Grained Policy Enforcement for Untrusted Android Applications. In J. Garcia-Alfaro, G. Lioudakis, N. Cuppens-Boulahia, S. Foley, W. M. Fitzgerald, Herausgeber, *Data Privacy Management and Autonomous Spontaneous Security*, S. 213–231. Springer, 2014. (Zitiert auf Seite 14)

[BKK+14] M. Bitsaki, C. Koutras, G. Koutras, F. Leymann, B. Mitschang, C. Nikolaou, N. Siafakas, S. Strauch, N. Tzanakis, M. Wieland. An Integrated mHealth Solution for Enhancing Patients' Health Online. In *Proceedings of the 6 th European Conference of the International Federation for Medical and Biological Engineering, MBEC 2014, 07-11 September 2014, Dubrovnic, Croatia*, S. 0–4. IFMBE, 2014. (Zitiert auf Seite 18)

[BO10] D. Barrera, P. van Oorschot. Secure Software Installation on Smartphones. *IEEE Security and Privacy*, 9(3), 2010. (Zitiert auf Seite 10)

[DAK15] DAK-Gesundheit. Checkliste für Fitness- und Gesundheits-Apps: Wie sicher ist meine App? Webseite, 2015. https://www.dak.de/dakonline/live/dak/formulare/leistungen/Checkliste_Fitness-_und_Gesundheits-Apps-1664782.html?/1664780/0. (Zitiert auf Seite 21)

[EGC+10] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, A. N. Sheth. TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, S. 1–6. USENIX Association, 2010. (Zitiert auf Seite 10)

[Elk15] M. Elkstein. Learn REST: A Tutorial. Webseite, 2015. http://rest.elkstein.org/. (Zitiert auf Seite 55)

[FCH+11] A. P. Felt, E. Chin, S. Hanna, D. Song, D. Wagner. Android Permissions Demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11. ACM, 2011. (Zitiert auf den Seiten 11, 14 und 22)

[FEW12] A. P. Felt, S. Egelman, D. Wagner. I've Got 99 Problems, But Vibration Ain't One: A Survey of Smartphone Users' Concerns. In *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '12. ACM, 2012. (Zitiert auf den Seiten 14 und 22)

[FHE+12] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, D. Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12. ACM, 2012. (Zitiert auf Seite 14)

[FS03] J. Frost, B. K. Smith. Visualizing Health: Imagery in Diabetes Education. In *Proceedings of the 2003 Conference on Designing for User Experiences*, DUX '03, S. 1–14. ACM, 2003. (Zitiert auf Seite 10)

[Grä15] P. Grätzel. Gesundheits-Apps: Sicherheit oft lausig. Webseite, 2015. http://www.kardiologie.org/gesundheits-apps-sicherheit-oft-lausig/57010. (Zitiert auf den Seiten 21 und 22)

[hei15] heise online. Android. Webseite, 2015. http://www.heise.de/thema/Android. (Zitiert auf Seite 35)

[Her15] E. Herrmann. Android Marshmallow: Features und Patches. Webseite, 2015. https://www.androidpit.de/android-marshmallow-features-release-news. (Zitiert auf Seite 35)

[HJAA10] H. Hong, H. Y. Jeong, R. I. Arriaga, G. D. Abowd. TriggerHunter: Designing an Educational Game for Families with Asthmatic Children. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, S. 3577–3582. ACM, 2010. (Zitiert auf Seite 10)

[Hüb15] M. Hübner. Gesundheits-Apps werden für Chroniker wichtig. Webseite, 2015. http://www.aerztezeitung.de/praxis_wirtschaft/w_specials/gesundheitsapps2011/article/888408/tk-studie-gesundheits-apps-chroniker-wichtig.html. (Zitiert auf den Seiten 9, 14, 21 und 22)

[KM12] M. Knöll, M. Moar. The Space of Digital Health Games. *International Journal of Computer Science in Sport*, 2012. (Zitiert auf den Seiten 23 und 73)

[Knö08] M. Knöll. Diabetes City: How Urban Game Design Strategies Can Help Diabetics. In D. Weerasinghe, Herausgeber, *Electronic Healthcare*, S. 82 – 89. Springer, 2008. (Zitiert auf Seite 23)

[Knö10] M. Knöll. On the top of high towers - Discussing Locations in a Mobile Health Game for Diabetics. *IADIS'10*, 2010. (Zitiert auf den Seiten 11, 23, 26, 70 und 73)

[KP12] P. Klasnja, W. Pratt. Healthcare in the pocket: Mapping the space of mobile-phone health interventions. *Journal of Biomedical Informatics*, 45(1), 2012. (Zitiert auf Seite 14)

[LBBK15] M. Lucht, R. Bredenkamp, M. Boecker, U. Kramer. Gesundheits- und Versorgungs-Apps - Hintergründe zu deren Entwicklung und Einsatz, 2015. Studie des Universitätsklinikums Freiburg. (Zitiert auf Seite 9)

[LS12] K. Larisch, T. Saller. BE-Tabelle. Webseite, 2012. http://www.netdoktor.de/Krankheiten/Diabetes/Tipps/BE-Tabelle-9342.html. (Zitiert auf Seite 56)

[Mat15] D. Matusiewicz. Gabler Wirtschaftslexikon - Das Wissen der Experten. Webseite, 2015. http://wirtschaftslexikon.gabler.de/Definition/mobile-health.html. (Zitiert auf Seite 9)

[MBK+13] I. Muslukhov, Y. Boshmaf, C. Kuo, J. Lester, K. Beznosov. Know Your Enemy: The Risk of Unauthorized Access in Smartphones by Insiders. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '13, S. 271–280. ACM, 2013. (Zitiert auf Seite 10)

[Med13] Media Net Software, S.A. Kids Beating Asthma. App, 2013. https://play.google.com/store/apps/details?id=es.medianet.hcsc01&hl=de. (Zitiert auf den Seiten 15 und 16)

[Mob15] Mobile Health Competence Centre. Mobile apps for sick children: Learning by playing. Webseite, 2015. http://www.mobilehealthglobal.com/in-the-news/news/188/mobile-apps-for-sick-children-learning-by-playing. (Zitiert auf Seite 15)

[Mon12] Monks Vertriebs GmbH. LärmApp. App, 2012. https://play.google.com/store/apps/details?id=de.monks.laermApp&hl=de. (Zitiert auf Seite 9)

[MSS+14] S. Mare, J. Sorber, M. Shin, C. Cornelius, D. Kotz. Hide-n-Sense: Preserving Privacy Efficiently in Wireless mHealth. In I. Chlamtac, Herausgeber, *Mobile Networks and Applications - The Journal of SPECIAL ISSUES on Mobility of Systems, Users, Data and Computing*. Springer, 2014. (Zitiert auf Seite 57)

[MyF15] MyFitnessPal, Inc. Kalorienzähler - MyFitnessPal. App, 2015. https://play.google.com/store/apps/details?id=com.myfitnesspal.android&hl=de. (Zitiert auf Seite 9)

[myS14a] mySugr GmbH. mySugr Junior. App, 2014. https://play.google.com/store/apps/details?id=com.mysugr.android.junior&hl=en. (Zitiert auf den Seiten 18, 19, 22, 45, 69 und 73)

[myS14b] mySugr GmbH. mySugr Pro Upgrade. Webseite, 2014. https://mysugr.com/de/mysugr-pro/. (Zitiert auf den Seiten 18 und 57)

[NPS+12] S. Nikkila, G. Patel, H. Sundaram, A. Kelliher, A. Sabharwal. Wind Runners: Designing a Game to Encourage Medical Adherence for Children with Asthma. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12. ACM, 2012. (Zitiert auf den Seiten 16, 17, 69 und 73)

[Ope15a] Open Source. ZXing. Webseite, 2015. https://github.com/zxing. (Zitiert auf den Seiten 47 und 56)

[Ope15b] OpenIT. ChronicOnline. App, 2015. https://play.google.com/store/apps/details?id=gr.openit.echo. (Zitiert auf den Seiten 18 und 20)

[Ora15] Oracle. Class SealedObject, 2015. https://docs.oracle.com/javase/7/docs/api/javax/crypto/SealedObject.html. (Zitiert auf Seite 59)

[Pet99] R. Petzoldt. *Sprechstunde Diabetes*. Gräfe und Unzer Verlag GmbH, 1999. (Zitiert auf Seite 27)

[PS11] J. Posegga, D. Schreckling. Next Generation Mobile Application Security. In U. Bub, K.-D. Wolfenstetter, Herausgeber, *IT-Sicherheit zwischen Regulierung und Innovation - Tagungsband zur zweiten EICT-Konferenz IT-Sicherheit*, S. 181–199. Springer, 2011. (Zitiert auf Seite 10)

[Psy13] Psyop Games. Nightmare: Malaria. App, 2013. http://nightmare.againstmalaria.com/press/sheet.php?p=nightmare#description. (Zitiert auf den Seiten 16 und 17)

[Pue15] Pueblo2708. Gesundheitsbuch. App, 2015. https://play.google.com/store/apps/details?id=com.akoudri.healthrecord.app&hl=de. (Zitiert auf Seite 9)

[rem13] remind4u2. Office-Workout. App, 2013. https://play.google.com/store/apps/details?id=com.remind4u2.office.workout.exercises&hl=de. (Zitiert auf Seite 9)

[Run15] Runtastic. Runtastic Laufen & Fitness. App, 2015. https://play.google.com/store/apps/details?id=com.runtastic.android&hl=de. (Zitiert auf den Seiten 13 und 22)

[SM13] C. Stach, B. Mitschang. Privacy Management for Mobile Platforms — A Review of Concepts and Approaches. *MDM'13*, 2013. (Zitiert auf Seite 11)

[SM14] C. Stach, B. Mitschang. Design and Implementation of the Privacy Management Platform. *MDM'14*, 2014. (Zitiert auf den Seiten 11 und 36)

[SM15] C. Stach, B. Mitschang. Der Secure Data Container (SDC) - Sicheres Datenmanagement für mobile Anwendungen. *DBSP'15*, 2015. (Zitiert auf den Seiten 51, 53, 58, 69 und 73)

[Sop13] Sophos Ltd. Security Threat Report 2014 - Smarter, Shadier, Stealthier Malware. 2013. (Zitiert auf Seite 10)

[SRD+15] B. M. Silva, J. Rodrigues, I. De la Torre Diez, M. Lopez-Coronado, K. Saleem. Mobile-health: A review of current state in 2015. *Journal of Biomedical Informatics*, 56(6), 2015. (Zitiert auf Seite 9)

[SS12] C. Stach, L. F. M. Schlindwein. Candy Castle - A Prototype for Pervasive Health Games. *PerCom'12*, 2012. (Zitiert auf den Seiten 11, 24, 25, 26, 35, 70 und 73)

[SSP+12] J. Sorber, M. Shin, R. Peterson, C. Cornelius, S. Mare, A. Prasad, Z. Marois, E. Smithayer, D. Kotz. An Amulet for Trustworthy Wearable mHealth. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, S. 7:1–7:6. ACM, 2012. (Zitiert auf Seite 57)

[Sta13a] C. Stach. How to Assure Privacy on Android Phones and Devices? *MDM'13*, 2013. (Zitiert auf Seite 11)

[Sta13b] C. Stach. Wie funktioniert Datenschutz auf Mobilplattformen? *Informatik'13*, 2013. (Zitiert auf den Seiten 11, 36 und 37)

[Sta15] C. Stach. How to Deal with Third Party Apps in a Privacy System —The PMP Gatekeeper—. *MDM'15*, 2015. (Zitiert auf den Seiten 11 und 37)

[SWM+15] F. Steimle, M. Wieland, B. Mitschang, S. Wagner, F. Leymann. Design and Implementation Issues of a Secure Cloud-Based Health Data Management System. *INPROC'15*, 2015. (Zitiert auf den Seiten 18, 30, 45, 57 und 69)

[Tec15] Techniker Krankenkasse. Gesundheits-Apps bewusst auswählen. Webseite, 2015. http://www.tk.de/tk/beratungsangebote/kompetent-als-patient/bewusst-auswaehlen/723596. (Zitiert auf Seite 20)

[TM14] S. Trepte, P. K. Masur. Privatheit im Wandel - Eine repräsentative Umfrage zur Wahrnehmung und Beurteilung von Privatheit. Forum Privatheit, 2014. Umfrage der Universität Hohenheim. (Zitiert auf den Seiten 13 und 22)

[VIT12] VITAL.de. Fitness- und Gesundheits-Apps. Webseite, 2012. http://www.vital.de/gesundheit/ratgeber/artikel/fitness-und-gesundheits-apps, Seiten 1 - 3. (Zitiert auf den Seiten 21 und 70)

[w3s15] w3schools. JSON Tutorial. Webseite, 2015. http://www.w3schools.com/json/. (Zitiert auf Seite 55)

[Wei99] M. Weiser. The Computer for the 21st Century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, 1999. (Zitiert auf Seite 9)

[Wie10] J. Wiemeyer. Gesundheit auf dem Spiel? – Serious Games in Prävention und Rehabilitation. *Deutsche Zeitschrift für Sportmedizin*, 61(11), 2010. (Zitiert auf den Seiten 14 und 22)

[WRR08] D. Weerasinghe, M. Rajarajan, V. Rakocevic. Device Data Protection in Mobile Healthcare Applications. In D. Weerasinghe, Herausgeber, *Electronic Healthcare*, S. 82 – 89. Springer, 2008. (Zitiert auf den Seiten 18, 51 und 69)

[Yuh15] Yuhiro. Android Marshmallow: Features und Patches. Webseite, 2015. http://www.yuhiro.de/welches-betriebssystem-fuer-app-waehlen/. (Zitiert auf Seite 35)

[ZFAIT15] B. C. Zapata, J. L. Fernández-Alemán, A. Idri, A. Toval. Empirical Studies on Usability of mHealth Apps: A Systematic Literature Review. In J. Ehrenfeld, Herausgeber, *Journal of Medical Systems*, Band 39. Springer, 2015. Issue 2. (Zitiert auf Seite 41)

Alle URLs wurden zuletzt am 08. 01. 2016 geprüft.

Erklärung

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

Ort, Datum, Unterschrift