Integration von Unternehmens-Apps unter Verwendung einer Platform zur Steuerung des App-Lebenszyklus

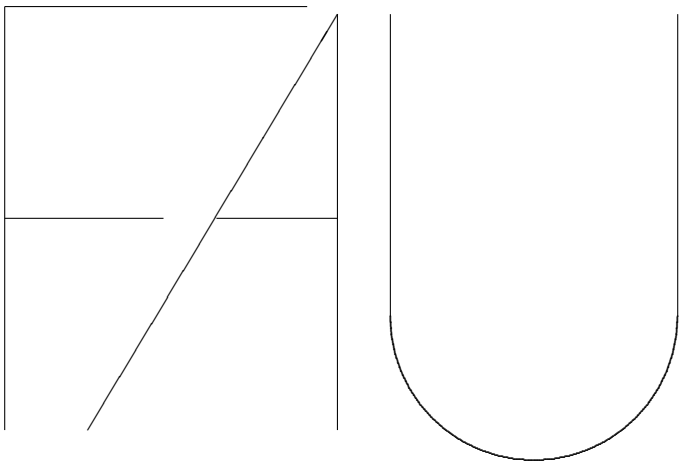
Masterarbeit

Alexander A. Schmidt

Open Source Research Group

Department Informatik
Technische Fakultät

Friedrich Alexander-
Universität
Erlangen-Nürnberg



Integration von Unternehmens-Apps unter Verwendung einer Platform zur Steuerung des App-Lebenszyklus

Masterarbeit im Fach Informatik

vorgelegt von

Alexander A. Schmidt

geb. 02.01.1987 in München

angefertigt am

Department Informatik
Professur für Opensource
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Univ.-Prof. Dr.-Ing. Dirk Riehle

Beginn der Arbeit: 15.09.2014
Abgabe der Arbeit: 15.04.2015

Erklärung zur Selbständigkeit

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass diese Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Der Universität Erlangen-Nürnberg, vertreten durch die Professur für Open Source, wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Masterarbeit einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt.

Erlangen, den 15.04.2015

(Alexander A. Schmidt)

Kurzfassung

Integration von Unternehmens-Apps unter Verwendung einer Platform zur Steuerung des App-Lebenszyklus

Aufgrund des vermehrten Einsatzes von mobilen Apps, insbesondere Productive Games, in Unternehmen werden zunehmend Dateninseln (engl. data silos) geschaffen. Im Rahmen dieser Masterarbeit wird eine strategische Lösung vorgestellt, Productive Games in einem Platform Server zu vereinen. Die Vereinheitlichung geschieht durch systematische Angleichung des Entwicklungsprozesses mit einem Versionskontrollsystems, der Veröffentlichung der Serverkomponenten durch einen Application Server und der Bereitstellung der Apps (Clientkomponenten) mittels eines für Unternehmen konzipierten App Store. Eine Middlewareschicht stellt eine Integrationsschnittstelle bereit, um gemeinsame Daten zwischen den Apps zu verwenden und Nachrichten auszutauschen. Darüberhinaus wird für den Spielifizierungsanteil (engl. Gamification) von Productive Games Schnittstellen für Monitoring und Gamification Daten implementiert.

Abstract

Integration of Enterprise Apps using a platform-driven app-lifecycle

The increase in mobile apps, especially productive games, within enterprise systems lead to the ermergence of data silos. In the course of this thesis, a strategic solution is proposed, to unite productive games wiothin a platform server. The unification is approached by aligning the development process with version control systems, publishing of servercomponents by an application server and the distribution of apps (client components) by an app store created for the requirements of enterise systems. A middleware layer is supporting integration interfaces, for sharing common data and exchanging messages between apps. For the gamification part of productive games, interfaces for monitoring and gamification data is being implemented.

Inhaltsverzeichnis

1 Einleitung

IT-Software kommt eine tragende Rolle zu, die Komplexität während der Bearbeitung von Geschäftsprozessen für Angestellte zu reduzieren. Es ist die Tendenz zu verzeichnen, dass effektives mobiles Arbeiten durch Verwendung von mobilen Apps vollzogen wird. Dieser Erfolg wurde im Endkonsumentenbereich erwiesen und setzt sich nun in der Geschäftswelt fort. [Chr12] Konsumenten wie auch Angestellte erwarten zunehmend Zugriff auf Dienstleistungen von Unternehmen mittels mobiler Endgeräte. [HPM13]

Mobile Apps ermöglichen den Angestellten, losgelöst von seinem Arbeitsplatz, mit Anwendungen zu interagieren und öffnet so die Türen zu neuen Möglichkeiten und Kategorien von Geschäftsanwendungen. Eine neue Kategorie sind **Productive Games**. [Tra15]

Productive Games sind "Apps / Agenten, innerhalb von Geschäftsprozessen, die nicht vorrangig oder ausschließlich zum Zwecke der Unterhaltung entworfen wurden. Das Hauptziel ist der Wissenstransfer und das Anlernen von Fertigkeiten, sowie die Verbesserung von Zusammenarbeit und Integration unter Verwendung von Spieledesigntechniken und Spielmechaniken, um in die Lage versetzt zu werden, effizienter in Geschäftsprozessen zu interagieren". My First Day ist ein Productive Games App mit dem Ziel "die Infrastruktur, die Prozesse und allgemeine Informationen eines Unternehmens einem neuen Angestellten zu vermitteln". [Tra]

Immer mehr Unternehmen verwenden Gamification [KVYM15], und somit ist es vorteilhaft **Productive Games** strategisch in die IT-Landschaft und in die Geschäftsabläufe von Unternehmen zu integrieren. **Mobile Productive Games** führen die Steigerung der Produktivität fort, indem diese ortsunabhängige Zugriffsmöglichkeiten, losgelöst vom Arbeitsplatz bieten, und somit die Antwortzeit minimieren, da Angestellte schnellere reagieren können.

Um jedes Jahr mehr und mehr Productive Games erfolgreich zu integrieren, müssen technische und organisatorische Prozesse optimiert und standardisiert werden. Der technische Aspekt beinhaltet den Prozess beginnend von der ursprünglichen Idee, Spezifikation, Entwicklung, Installation bis hin zur Integration in die Unternehmenssysteme. In dieser Arbeit wird eine technologische Lösung angestrebt zur Integration von Productive

Games innerhalb von Unternehmenssysteme mit Fokus auf Entwicklung, Installation und Integrationsaspekten.

1.1 Motivation

Für Unternehmen, die auf die Belange ihrer Angestellten eingehen und mehr mobile Productive Games anbieten, wird eine Unternehmensstrategie zur Etablierung einer grundlegenden Basis für alle Productive Games zielfördernd. Mit einer angemessenen strategischen Lösung, können Unternehmen ihren Entwicklungszyklus neuer Productive Games verbessern. Diese sieht die Erstellung einer einfach zu benutzenden Plattform vor, in der "Pattern " und "Best-Practices" zu einer integralen Gesamtlösung kombiniert werden.

1.2 Ziele

Eine Umfrage, in Auftrag gegen von Symantec und veröffentlicht unter dem Titel "state of mobility", erbrachte das Ergebnis, dass ein Drittel von 6275 befragten Unternehmen angeben, einen internen App Store für ihr Unternehmen zu errichten, während ein weiteres Drittel eine Umsetzung plant. [Bri12] In dieser Arbeit wird eine Plattform entworfen, mit dem Ziel, den Lebenszyklus einer App von der Entwicklung, Kompilation, Hosting und Veröffentlichung in einem eigenen App Store, zu unterstützen. Auf diese Weise wird die Phase von Idee zum Productive Games App Prototype verkürzt. Ein vorhandener Prototyp stellt eine bessere Kalkulationsgrundlage bereit, um die Implementierung einer neuen Idee als vorteilhaft zu bewerten und die Spezifikation, Installation und Benutzereinführungsphasen zu verkürzen.

1.3 Vorgehensweise

Das Ziel der Masterarbeit, kann daher folgend zusammengefasst werden:

Erstellung einer Plattform,
zur Unterstützung des Lebenszyklus von Productive Games.

Die Bearbeitung erfolgt in folgenden Schritten:

- **Untersuche die Productive Games mobile Geschäftsanwendungsapp My First Day**, entwickelt an der Professur für OpenSource von Prof. Dirk Riehle und bestimme die Erfordernisse für die Productive Games Platform.

- **Nachforschung** ist nötig um die erforderlichen Informationen zu erhalten um die angestrebte Plattformlösung mit tiefgreifenden Wissen über mobile Unternehmensapplikationen und ihren architektonischen Anforderungen für die Serverkomponenten einschließlich des Betreibens des Plattformkerns auszugestalten. Desweiteren werden die Voraussetzungen untersucht um Unternehmensintegration zwischen den Serverkomponenten und dem Unternehmenssystem (engl.enterprise systems) zu gewährleisten.
- **Spezifikation und Kurzdarstellung der Productive Games Platform** Die Voraussetzungen für den Entwurf der Architektur basierend auf der Literaturrecherche wird erarbeitet. Der **Entwurf** wird skizziert und eine Kurzdarstellung über die Platform folgt. Anschließend wird eine **prototypische Implementierung** mit Hilfe von best-practice Entwurfsmustern und Bausteinen mit dem Fokus auf modulare Bauweise.
- **Evaluation** Untersuchung, ob die Productive Games Platform verwendet werden kann, um den Prototyp My First Day mit der vorgeschlagenen Frameworklösung zu entwerfen. Abschließend folgt deine **Evaluation** der prototypischen Platform indem diese mit existierenden Produkten verglichen wird und der Vorteil im Bezug auf MyFirstDay erörtert wird.

2 Spielifizierung im Unternehmensbereich

Spielifizierung (engl. Gamification) ist der neueste Trend in Unternehmen. Es gibt zahlreiche wissenschaftliche Arbeiten und Pilotprojekte innerhalb Firmen, die das neue Konzept für Mitarbeitersoftware für sich nutzen möchten. [KVYM15] In diesem Kapitel soll ein kurzer Überblick gegeben werden, damit der Kontext, in dessen Umfeld sich diese Masterarbeit befindet, aufgezeigt wurde. Die Behandlung des Themas wird daher auf wenige Quellen beschränkt, da es nur als Startpunkt in das große Themenkomplexes dient.

2.1 Begriffserklärung

Die früheste Benutzung des Begriffs "gamification" im englischsprachigen Raum wird nach Deterding [DDKN11] auf das Jahr 2008 datiert, aber erst Ende des Jahres 2010 war der Begriff in Nutzung. Andere Begriffe sind unter anderem "productivity games", "surveillance entertainment", "applied gaming" und "playful design". Der Begriff wird in der Industrie laut Deterding mit zwei verschiedenen Nuancen benutzt. Einerseits, um die Allgegenwärtigkeit von Videospielen im alltäglichen Leben zu beschreiben. Die andere Bedeutung, die in dieser Arbeit durchgehend verwendet wird, ist folgende: Videospiele sind mit dem Fokus auf Unterhaltung konzipiert und haben nachhaltig bewiesen, dass sie Nutzer motivieren, sich arbeits- und zeitintensiv mit ihnen zu beschäftigen. Der Ausgangspunkt der Beschäftigung mit Spielifizierung ist die Annahme, dass Spielelemente auf Produkte ohne Spielcharakter angewendet ausreichen, um die Attraktivität dieser Produkte und andere Dienstleistungen zu erhöhen. Dies soll die Nutzer, angetrieben durch die Freude am Spielen sowie durch das Belohnungssystem des Spieles, motivieren, mehr und bessere Leistungen zu erbringen.

Nach Deterding [DDKN11] ist Gamification "die Verwendung von Spieldesignelementen in spielfremden Kontext" (englisches Original: " gamification is the use of game design elements in non-game contexts"). Dazu abzugrenzend ist, dass die Nutzung von Game

Controllern oder andere Eingabemethoden, die von Spielen bekannt sind, nicht direkt unter dem Begriff Gamification Anwendung finden. Gamification beschreibt ausschließlich die Entwurfsmethodik und -elemente. Die Entwurfselemente können in folgende Kategorien eingeteilt werden:

- **Vorlagen für Benutzeroberflächen** Bewährte Entwurfskomponenten, die zur Interaktion dienen, oder graphische Komponenten, die für einen bestimmten Kontext wieder verwendet werden können. Oftmals enthalten diese eine prototypische Implementierung, die noch angepasst werden kann. Bewährte Vorlagen gibt es meist für Badges, Leaderboards und Avatar-Levels.
- **Spielentwurfsmuster und -mechaniken** Wiederholende Muster, die den Spielverlauf beeinflussen, und wiederholt eingesetzt werden können. Dazu zählen zeitliche Vorgaben, limitierter Zugriff auf Ressourcen oder Spielrunden.
- **Entwurfsprinzipien, -heuristiken** Auf Evaluation gestützte Richtlinien um an ein Entwurfsproblem anzusetzen bzw. um eine Entwurfslösung analysieren zu können. Zu diesen Richtlinien zählt beständiges Spielen, klare Spielziele, Ausgewogenheit von Spielrichtungen.
- **Spielmodelle** Konzeptionelle Modelle, die den Spielen oder der Spielerfahrung zu Grunde liegen, wie beispielsweise MDA, Herausforderungen, Fantasie, Neugierde.
- **Spielentwurfmethoden** Entwurfsbezogene Prozesse und Vorgehen, wie beispielsweise Playtesting und spielbasierter Entwurf.

2.2 Anwendungsgebiete

Unternehmen sind vor allem an der Anwendung von Spielifizierung für reale Beweggründe interessiert. Diese werden unter dem Begriff 'serious games' [Abt87] kategorisiert.

2.2.1 GWAP

Eine weitere Bezeichnung für serious games ist die eingeführte Bezeichnung 'GWAP' (Games with a purpose) von Luis von Ahn [VAD08]. Der Startpunkt eines GWAP, ist ein Problem dass schwierig oder unmöglich von Computern, dafür aber leicht von Menschen gelöst werden kann. Es gilt Anreize für die menschlichen Spieler zu schaffen, um sie an das Spiel zu binden und korrekte verwertbare Ergebnisse zu liefern. Drei Spielevorlagen wurden näher untersucht, die von dem Spieler eine Ergebnisberechnung verlangen.

- **output-agreement games** Zu Beginn eines Spieles werden zwei Spieler zufällig aus einer Liste potentieller Spieler ausgewählt. Das Spiel ist rundenbasiert. In jeder Runde erhält ein Spieler eine neue Eingabe, und das Ziel des Spieles ist es, dieselbe Ausgabe (output) wie der andere Spieler zu produzieren. Die Spieler kennen sich aber nicht, und haben keine Kommunikationsmöglichkeit. Die Spieler müssen nicht zur selben Zeit, dieselbe Eingabe lösen, es reicht wenn dies in Zukunft miteinander abgeglichen werden kann. Wenn die Eingaben Bilder sind, und die Ausgaben Wörter sind, die die Bilder beschreiben, so wird das Spiel ESP genannt. In dieser Konstellation müssen die Spieler nicht zwangsweise, ein korrektes Ergebnis produzieren, sondern ähnlich wie der andere Spieler denken, und aufgrundlage derselben Eingabe, dieselbe Ausgabe produzieren. Ein motivierender Anreiz bei diesem Spiel ist es, für die Erfolge belohnt zu werden, das gleiche wie ein anderer Mensch zu denken.
- **inversion-problem-games** Zu Beginn eines Spieles werden zwei Spieler zufällig aus einer Liste potentieller Spieler ausgewählt. Das Spiel ist rundenbasiert. Ein Spieler bekommt die Aufgabe des Beschreibers, der andere die des Rätsellösers. In jeder Runde erhält der Beschreiber, eine Eingabe und muss dem Rätsellöser beschreibende Texte senden. Das Ziel des Spiels ist es, dass der Rätsellöser mit Hilfe der beschreibenden Texte, die ursprüngliche Eingabe, die der Beschreiber erhalten hat, errät. Die soziale Interaktion, zwischen den zwei Spielern kann erhöht werden, wenn die Lösungsversuche des Rätsellösers dem Beschreiber angezeigt werden, und dieser heiße und kalte Antworten markieren kann.
- **input-agreement-games** Zu Beginn eines Spieles werden zwei Spieler zufällig aus einer Liste potentieller Spieler ausgewählt. Das Spiel ist rundenbasiert. Beide Spieler erhalten in jeder Runde Eingaben von dem Spiel, sehen aber nicht die Eingabe, die der andere Spieler erhält. Beide müssen ihre Eingabe, dem anderen gegenüber beschreiben, und erraten ob beide dieselbe Eingabe erhalten haben oder nicht. Damit nicht unüberlegt die Lösung geraten wird, werden harte Strafen im Spiel für Falschantworten verhängt.

2.2.2 productive games

Jedoch ist Spielifizierung nicht nur auf GWAPs beschränkt, die ein Extrembeispiel darstellen, da sie ausschließlich zur Berechnung von Lösungen ausgelegt sind, für die der Mensch benötigt wird. Bei GWAPS ist Spielifizierung sehr förderlich um der Monotonie

der Anwendung entgegenzuwirken. Andere Anwendungen, die darauf ausgelegt sind, den Mitarbeiter zu motivieren, verfolgen ein nicht so schematisches Vorgehen wie bei GWAPs. Das Ziel von Productive Games ist es den Wissenstransfer und das Anlernen von Fertigkeiten mit spielerischen Elementen zu verbinden. Ein strenges Schema existiert nicht, sondern jeder Anwendungsfall bestimmt, welche Spielelemente 2.1 Anwendung finden.

2.3 Evaluation

Jennifer Thom, von dem IBM Thomas J. Watson Research Center führte eine Studie durch, in der das gängige Punktebelohnungssystem aus einer sozialen Kollaborationsplattform entfernt wurde. Ihre Untersuchung, die jedoch verschiedene starke Ausprägungen intrinsischer Beweggründen innerhalb der Teilnehmergruppe außer Betracht ließ, kam zu dem Ergebnis, dass die Beteiligung nach Entfernung eines Punktesystem fast um die Hälfte sank. [TMD12] Angesichts der Tatsache, dass 58 Prozent der App Store Downloads Spiele [Kim10] sind, und Menschen eine Affinität zu Spielen in die Wiege gelegt wurde [DDKN11], sollte weitere Forschung im Bereich Gamification betrieben werden. Um diese Forschung zu fördern, ist die Erweiterung der Plattform um Gamification Services, wie beispielsweise ein Punktesystem, gedacht.

[DDKN11]

3 App Store

Software Ecosystems (SECOs) sind laut Slinger Jansen et al. [JBF09] definiert, als eine Gruppe von Akteuren, die zusammen als eine Einheit fungieren um mit einem gemeinsamen Software- und Servicemarkt zu interagieren. Dabei unterhalten die Akteure Beziehungen untereinander, die von einer gemeinsamen technologischen Plattform oder Markt untermauert wird. Sie nehmen auch am Austausch von Informationen, Ressourcen und Artefakten teil.

Ein Beispiel für ein SECO ist Apple's App Store, der eine Vorreiterposition auf dem mobilen Software Markt für Endkonsumenten hält. Die unkomplizierte Einkaufserfahrung und Verwaltung der Apps, sind das Markenzeichen des App Store. Das Konzept des App Stores hat maßgeblich den Ablauf beeinflusst, wie Software produziert, veröffentlicht, verkauft und verteilt wird. [GMBV12] [Kim10]

Andererseits wird der Software Markt für Unternehmen noch von eigenständigen (engl. stand-alone) Softwarelösungen dominiert. Zum einen von Customer Relationship Management (CRM), oder bereits integriert durch Enterprise Resource Planing (ERP). Das Vertriebskonzept ist ein meist langjähriger und aufwendiger Prozess, der oft zentralisiert gesteuert und von einer IT-Abteilung betrieben wird. Es vergehen oft mehrere Monate für Evaluation mit vergleichenden Softwarelösungen und Gesprächen mit den Anbietern, bis eine Softwareanschaffung abgeschlossen werden kann. [LLL07] [Med09]

Productive Games sollen jedem Anwender in einer einfachen Weise zur Verfügung gestellt werden. Dies soll von Privatanwendern, Besuchern auf Firmengeländen, sowie von Firmenangestellten innerhalb einer globalen Unternehmensstrategie, wie auch von einzelnen Firmenangestellten (einzelne Lizenzen) genutzt werden können. Aus diesem Grund soll die Basis jedes einzelnen Productive Games auf diesem Fundament aufbauen, der die erwähnten Möglichkeiten bietet.

Das App Store Konzept im Unternehmensbereich, kann Kosten senken, durch geringe Kosten pro App pro Mitarbeiter, die Entscheidung von zentralen Entscheidungspositionen in die der Abteilungen, oder sogar Endnutzer bringen. Die Auflistung von Apps verschiedener Anbieter, sowie die Bewertungsmöglichkeiten begünstigen das Auffinden geeigneter Apps. So können leichter Apps entdeckt werden, die Arbeitsprozesse des eigenen

Unternehmens produktiver gestalten können. Werden die Apps separat auf Internetseiten vertrieben, müssen diese Internetseiten erst durch Suchmaschinen gefunden werden. Eine Integration in einem bekannten App Store erleichtert diesen Prozess erheblich. Durch Nutzung eines App Stores wird die Verwaltung von Software eines einzelnen Anbieters, oder im besten Fall alle Apps, die das Unternehmen nutzt, innerhalb eines zentralen Katalogs (App Store) kontrollierbar. Dies betrifft zum einen die Verwaltung der Software auf Endgeräten, sowie die Verwaltung der erworbenen Lizenzen.

3.1 Faktoren zum Aufbau des App Stores

Die Auswertung, wie der Siegeszug von App Stores verlief, kommt zu dem Ergebnis, dass sich dies in folgenden vier Phasen vollzog. [Kim10].

- **Mobile Webseiten** wurden für mobile Endgeräte zugeschnitten. Unter der Vielzahl an Webseiten waren u.a. Google, Google Maps, Youtube, Facebook und Wikipedia. Diese führte zu einem Zuwachs an mobilen Endgeräten mit schnellem Internetzugang (3G, HSPA, Wifi).
- **Internet-Flatrates** der Telekommunikationsbetreiber zu immer niedrigeren Preisen, sorgten für günstigen Internetzugang.
- **Smartphones** wurden immer beliebter und erschwinglicher. Leistungsfähigere und multimediale Smartphones begünstigten weiterhin die Verwendung von mobilen Webseiten und internetbasierten Apps.
- Zugang für Drittanbieter auf mobilen Betriebssysteme zur Entwicklung und Bereitstellung von Apps wurde geschaffen. Entwicklern wurde Zugriff auf eine breit umfassende API einschließlich Zugang zu den Funktionalitäten der mobilen Betriebssysteme gewährt.

3.2 Ökosystem des App Stores

Ein App Store beinhaltet ein zentrales Softwareverzeichnis, Benutzerbewertungen und Feedback für jede Software und erleichtert den Erwerb und die Installation von Software und Updates. Apple baute den App Store umfassend und strategisch auf. Die angebotenen Schnittstellen sind das Fundament seines bestehenden Erfolges. Das Ökosystem des App Stores beinhaltet:[Kim10] [GMBV12]

- **Einfacher Zugriff auf den App Store** durch die App Store App auf dem iPhone bzw. durch die App Store Software für Windows oder MAC OS. Sie beinhaltet u.a. das Durchstöbern des App Store Katalog nach Apps und eine Suchfunktion um Apps zu finden. Der App Store übernimmt auch die Installation und falls erforderlich, den einfachen Bezahlungsprozess von nicht-kostenfreien Apps.
- **iOS Application Plattform** bietet eine breit umfassende API mit zahlreichen Funktionen für den Zugriff und Steuerung von Kamera, Ortung, Gyroskop, Nachrichtenservices und weiteren Sensoren und Services.
- **Integration von Drittanbieter-Apps** Enterprise App Stores werden zunehmend von Plattformentwicklern gebaut, so beispielsweise der Jive Apps Market. Jive 5 erlaubt die Integration von Drittanbieter-Apps. Die Vision ist, dass viele Unternehmen ihren internen App Store für Drittanbieter öffnen. Die Hauptakteure in der Business Intelligence, Oracle, MicroStrategy, IBM und SAP verfolgen diese Vision.
- **Einfaches und effizientes Geschäftsmodell** mit geringen Einstiegsbarrieren ermöglichen selbst Entwicklern selbstständig Apps im App Store zu verkaufen.
- **Ein attraktives Vergütungsmodell für Drittanbieterapps** behält für Apple 30 Prozent des Umsatzes von kostenpflichtigen Apps ein, und bezahlt den Entwicklern die verbleibenden 70 Prozent pro gekaufter App. Im Gegensatz dazu werden im mobilen Markt 40-50 Prozent vom Serviceleister und weitere 35-40 Prozent von dem Vertrieb einbehalten, sodass die Hersteller der Apps wenig von dem Umsatz erhalten. Das Vergütungsmodell ist aus diesem Grund für viele Anbieter sehr lukrativ.

3.3 Differenzierung von App Stores

Nach Andrea Giessmann et al. [GSSDV12] sind App Stores zur Softwareverteilung in Unternehmen einsetzbar. Sie können in drei Kategorien eingeteilt werden.

- **öffentlich App Stores** sind in Unternehmen nur einsetzbar, wenn Einschränkungen getroffen werden können, welche Apps von Nutzern verwendet werden dürfen. Unternehmen müssen strenge Richtlinien durchsetzen, damit Sicherheit und Kompatibilität gewährleistet ist.
- **herstellerspezifische App Stores** Anbieter von Geschäftsanwendungen stellen einen eigenen App Store zur Verfügung, der die Palette ihrer Geschäftsanwendungen

erweitert. Die darin enthaltenen Apps könenn vom Hersteller selbst oder von Drittanbietern stammen.

- **unternehmsinterne App Stores**, die von Unternehmen ins Leben gerufene App Stores, werden auch in-house oder corporate app stores genannt. Der Aufbau dieser App Stores ist laut Experten nur für große Unternehmen lohnenswert.

3.4 Funktionalität des App Stores

Nachdem auf das Ökosystem eingegangen wurde, soll nun genauer die Funktionalität des App Stores ausgearbeitet werden. Der App Store setzt sich laut Michael Goul et al. [GMBV12] aus dem Portal und der Plattform zusammen. Folgende Funktionalitäten ergeben das Gesamtkonzept eines App Store: [Kim10] [GMBV12]

App Store Portal

- **Softwareverteilung** Diese beginnt für Entwickler mit der Veröffentlichung der Apps im App Store, und unterstützt das Einbringen von Updates. Benutzer können Apps installieren, updaten oder entfernen.
- **App Katalog** Die veröffentlichen Apps müssen gefunden werden. Zu diesem Zwecke dient ein Katalog, der nach bestimmten Kriterien und Kategorien sortiert wurde. Jede App wird mit einem Namen und einem Logo und einem beschreibenden Text angezeigt. Als Zusatz werden oftmals Screenshots der App angeboten, um einen Eindruck zu gewinnen, welche Möglichkeiten die App bietet. Durch eine Suchfunktion, wird zudem das gezielte Auffinden von angebotenen Apps, erleichtert.
- **Vergütung** Ein Bezahlsystem steht zur Verfügung, das in konsistenter Weise den Käufer für jede angebotene App, durch den gleichen Bezahlprozess führt.
- **Feedback und Bewertungen** dienen dazu die Meinung von anderen Anwendern einzusehen, um den Kauf einer App abzuwägen, oder auch bei einer kostenfreien App, sich ohne Installation ein Urteil bilden zu können. Der leichte Zugang Feedbacks geben zu können, führt dazu, dass Anwender leichter Feedback geben, und durch evtl. Kommentarfunktionen, Entwickler in Kontakt mit den Anwendern treten können. Kundenwünschen werden schneller aufgedeckt, und können zur Festigung der Marktposition beitragen.

App Store Plattform

- Teilweises **Testen** von Apps innerhalb der Plattform sollte unterstützt werden.
- Die **Sicherheitsverwaltung** sollte konfigurierbar sein, um beispielsweise die Art der Authentifikation zu bestimmen, und in der Client-Server-Kommunikation die Verschlüsselungsart anzugeben.
- Die **Genehmigung** von Apps muss von der IT Abteilung erteilt, werden, sodass Apps von den Nutzern verwendet werden dürfen.
- **Richtlinien** müssen einstellbar sein, sodass eine zentrale Verwaltung dieser über die Plattform möglich ist.

4 Mobile Apps im Unternehmenskontext

Mobile Apps in Unternehmen erlauben effektiveres Arbeiten, Reaktionen auf Ereignisse können schneller erfolgen. Insgesamt unterstützen mobile Apps Angestellte innerhalb flexibler und agiler Prozesse besser als stationäre Softwaresysteme. Durch mobile Unterstützung der Angestellten, werden Geschäftsprozesse beschleunigt, die Produktivität erhöht und zugleich die Kosten der Geschäftsprozesse reduziert. Die Antriebsmotoren für mobile Apps im Unternehmenskontext sind Produktivitätssteigerung und Optimierung der Geschäftsprozesse. Allerdings werden mit mobilen Apps nicht nur Vorteile erkauft, sondern auch Nachteile stellen sich ein, die betrachtet werden müssen. Dazu zählt als dringlichste die Sicherheit. Nicht weniger wichtig sind die Verwaltung und Integration der mobilen Apps in die Unternehmensstruktur, da diese ohne Strategie hohe Kosten nach sich ziehen können. Nach allem Optimismus zu mobilen Apps, und in dieser Arbeit vor allem Productive Games, sind die damit einhergehenden Einschnitte in die Privatsphäre zu diskutieren.

4.1 Sicherheit

Der Sicherheitsaspekt in Unternehmen spielt eine große Rolle, ob Software zugelassen werden kann. [GMBV12] Anwendungen durchlaufen in Unternehmen meist teuren Prüfverfahren bis sie zur Installation und Verwendung zugelassen sind. Sicherheit ist ein Belang, der von beiden Kommunikationsseiten durchgesetzt werden muss. Schwachstellen oder falsche Konfiguration können fatal sein. [She12]

4.1.1 clientseitige Sicherheit

Die Sicherheit auf dem mobilen Endgerät kann durch folgende Maßnahmen erreicht werden.

4.1.1.1 Viren und Malware

Der typische Anhaltspunkt um Sicherheit auf der Clientseite zu erstellen, ist das Sicherstellen, dass keine Viren oder Malware auf dem Clientsystem installiert sind bzw. installiert werden können. [TJ07] In der Betrachtung wird ausschließlich auf mobile Betriebssystemsoftware eingegangen, da dies die Hauptanwendungsgruppe der zu erstellende Plattform ist.

4.1.1.2 Sicherheitsrichtlinien des mobilen Endgerätes

Am besten werden Sicherheitsrichtlinien verwendet, die es nur der App oder Apps desselben Entwicklers (durch Zertifikate ausgegeben von Apple oder im Fall von Android von einer öffentlichen Zertifizierungsstelle) gestatten, auf Daten einer App zuzugreifen. Bei der Verwendung von temporären Dateien muss sichergestellt werden, dass nur Berechtigte diese Dateien lesen können. Strenge Sicherheitsrichtlinien müssen auch bei der Anbindung und Verwendung von lokalen Services auf dem lokalen Endgerät verwendet werden. Dies beinhaltet die Kommunikation mit anderen Unternehmensapps, sowie Standardanwendungen auf dem Endgerät wie beispielsweise Google Maps, der Bilder-App bzw. Album-App, der Sms-App, der Telefon-App, Kontakte-App und weiteren Apps. [EOM09] [JMV+11]

4.1.1.3 Unternehmensdaten

Die größten Hürden sind die Vielfalt der mobilen Endgeräte und der unterschiedlichen mobilen Betriebssysteme um die Sicherheit der Unternehmensdaten zu gewährleisten. Die Herausforderung steigt mit der Zunahme von BYOD (Bring your own device) Richtlinien. BYOD heißt, dass Mitarbeiter ihr eigenen Smartphones ins Unternehmen mitbringen. Der Vorteil für Angestellte ist, dasss sie mit einem Ihnen sehr vertrauten mobilen Endgerät arbeiten können und kein Zweitgerät mitführen müssen. In diesem Fall wird jedoch die Verwendung des Gerätes meist durch lange Passworteingaben erschwert, da die Fingerabdrucktechniken, die aktuelle Geräte bieten, in Unternehmen nicht freigegeben wurden. Wichtig, um die Sicherheit der Unternehmensdaten zu gewährleisten, in dem die Bereitstellung der Unternehmensapps, deren Aktualisierung, sowie deren Entfernung ohne physischen Zugriff auf das Gerät extern kontrolliert werden kann. [GGR13]

4.1.2 serverseitige Sicherheit

Auf der Serverseite müssen die Sicherheitsaspekte des Unternehmens mit denen der Serverkomponenten der mobilen Apps integriert werden. Dies bedeutet, dass Authentifikation und Autorisierung entsprechend den Sicherheitsrichtlinien des Unternehmen erfolgen müssen. Nur Benutzer, die bestimmten Rollen angehören oder über bestimmten Rechte verfügen, soll die Installation und Verwendung von bestimmten Apps erlaubt sein. Des Weiteren muss festgelegt werden, mit welchen anderen unternehmensinternen Systemen die mobile App bzw. die zugehörige Serverkomponente kommunizieren darf. Dazu wird ein System benötigt, das Nutzerverwaltung sowie Integrationsverwaltung bereitstellt.

4.1.3 Kommunikationssicherheit

Kommunikation zwischen mobilen Endgeräten und dem Unternehmensnetzwerk findet bedingt durch die Limitation, sowie der Verwendungsweise der mobilen Endgeräte durch kabellose Verbindungen statt. Diese sind Wlan, Bluetooth oder Nfc, wobei Wlan die vornehmlich (am meistens) verwendete Technik darstellt. [FLM+10] Die CIA Sicherheitskriterien müssen hierzu erfüllt werden. CIA steht als Abkürzung für Confidentiality, Integrity und Availability. Confidentialty ist die Vertraulichkeit und bedeutet, dass nur beide Kommunikationspartner die Nachricht lesen können, und wird durch Verschlüsselung der Nachrichten erreicht. Integrität bedeutet, dass niemand den Inhalt der Nachricht zu dem anderen Kommunikationspartner verändern oder eine neue legitim erscheinende Nachricht senden kann, und wird durch ein sicheres Übertragungsprotokoll wie beispielsweise SSL garantiert. Der Legitimitätsnachweis erfolgt durch Authentifizierung. Availabilty bedeutet, dass zu jeder Zeit der Zugriff auf Dienstleistungen gegeben ist, und die Dienstleistungen aufgrund von Hackerangriffen beeinträchtigt sind, bis zu dem Ausmaß, dass diese eingestellt werden müssen. [SSVE04]

4.2 Verwaltung

Wie bereits angesprochen ist die Verwaltung von mobilen Geräten eine weitere Herausforderung. Unternehmen verwalten nur wenige unterschiedliche Desktop-PCs, auf denen meistens jedoch dasselbe Betriebssystem und dieselbe Sicherheitsschichten installiert worden sind. Das mobile Betriebssystem unterschiedlicher Smartphones, kann jedoch nicht so angeglichen werden, dass auf beiden dasselbe mobile Betriebssysteme läuft.

In nur sehr seltenen Fällen, kann das mobile Betriebssystem durch ein anderes ersetzt werden. [Ltd15] Desweiteren sind diese mobilen Betriebssysteme in ihrer Anpassbarkeit limitiert. Im Vergleich dazu kann ein linuxbasierter Client in beliebigen Schichten des Betriebssystem angepasst werden. Die nicht vermeidbare Vielfältigkeit der mobilen Betriebssysteme erfordert deswegen komplexere Richtlinien und Prozessabläufe. Eine Lösung, die den gesamten App-Lebenszyklus unterstützt ist, vereinfacht die mobile Verwaltung. Der Lebenszyklus beginnt mit der Entwicklungsphase, die wechselseitig von der Testphase abgelöst wird, bis schließlich die Veröffentlichung der App ansteht. Nach der Veröffentlichung schließt sich die Wartungsphase an, bis zu einem Zeitpunkt, in dem die App abgeschafft und von allen Geräten entfernt wird. Verwaltung hat neben dieser Linearität verschiedene Aspekte, die zu unterschiedlichen Zeiten eintreten. Dazu zählen neben der Verwaltung der Entwicklungs- und Veröffentlichungsphase, die Verwaltung der laufenden App. Diese kann unterteilt werden, in Hosting, Benutzerverwaltung, Sicherheitsrichtlinien, Integrationsverwaltung. Hosting beinhaltet die Skalierbarkeit und Performanz. Es kann bestimmt werden, wie viele Ressourcen für die Serverkomponente einer App bereitgestellt werden müssen. Benutzerverwaltung beinhaltet u.a. das Anlegen und Vergeben von Rechten und Rollen. Dies kann global für eine Vielzahl von Apps geschehen oder explizit nur für bestimmte Apps. Die Sicherheitsrichtlinien beinhalten die Vorgabe der Kennwortstärke und der Sicherheitsprotokolle. Die Integrationsverwaltung gestattet das "Verdrahten" von Serverkomponenten auf Unternehmensseite, sowie das Verbinden von mobilen Apps auf dem mobilen Endgerät.

Zur Verwaltung von mobilen Apps in Unternehmen werden bisher zwei Konzepte angewendet. Einerseits der geräteorientierte Ansatz, der als "Mobile Device Managment" (MDM) bezeichnet wird. Diese tragen jedoch den Phasen des Apps-Lebenszyklus keine Rechnung. "Mobile Application Managment" (MAM) versprechen einen app-orientierten Ansatz zur Lösung der Verwaltungsaufgaben.

4.2.1 Geräteverwaltung

Mobile Device Management ist ein zentraler Punkt in dem Konfigurationen und Richtlinien hinterlegt werden, und auf allen Geräten im Unternehmensnetzwerk übertragen werden. Durch diese zentrale Kontrollinstanz werden Sicherheitsrisiken reduziert und Kosten für den Aufwand einer anderweitigen Verwaltung eingespart. "Der Nutzen von MDMs besteht in der Optimierung der Funktionalität und Sicherheit von mobilen Kommunikationsnetzwerken bei gleichzeitiger Minimierung von Kosten und Ausfallzeit." [Wik15b]

4.2.2 App-Lebenszyklus-Verwaltung

Mobile Application Management beschreibt Software, die zur Verteilung und Zugriffsverwaltung von mobilen Apps in Unternehmensumgebungen dienen. Zur Unterstützung des gesamten Lebenszyklus beginnend von der Idee, von Entwicklung zur Auslieferung und Wartung, bis schließlich zur Ablösung. [Wik15a]

4.3 Integration

Integration ist ein vielschichtiger Prozess, der auf unterschiedlichen Ebenen stattfindet und verschiedene Abläufe beinhaltet. Die Integration kann auf technologischer und semantischer Ebene stattfinden und erfordert deswegen die Einbeziehung unterschiedlicher Rollen, beispielsweise Administratoren und Experten.

4.3.1 durch Technologie

Auf der technologischen Ebene müssen Datenbestände integriert und gegebenenfalls optimiert (gesäubert, Redundanz beseitig, Abbildungen zu anderen Datensätzen hergestellt, mit Fremdschlüsselbeziehungen versehen) werden. Des Weiteren müssen Softwarekomponente durch Anbindung an Services oder vorhandene Schnittstellen angepasst werden, damit Kommunikation stattfinden kann. Schließlich müssen Sicherheitseinstellungen konform der Sicherheitsrichtlinien angepasst werden.

4.3.2 durch Semantik

Nachdem die technologischen Voraussetzungen gegeben sind, kann die semantische Integration stattfinden. Diese schlägt sich meist auch in technologischen Einstellungen oder Veränderungen nieder. Beispielsweise eine Redundanzbeseitigung bzw. Abbildung zwischen den Datensätzen zweier unterschiedlicher Anwendungen mit getrennt verwalteten Sichten auf dieselben realen Objekte.

4.4 Kontext-Sensitivität

Der momentane Aufenthaltsort, die Benutzeridentität und die aktuellen Daten bestimmten die von der App zu lieferenden kontextuellen Informationen.

4.5 Anwenderkreis

Zur groben Einteilung von Anwendern, wird oft die Klassifizierung B2C, B2B und B2E verwendet.

- **Business-to-Consumers (B2C)** Apps sind vorrangig dazu da, Kunden komfortablen Zugriff auf Dienstleistungen mit Wert auf überzeugenden Benutzererfahrung zu bieten, die das Image des Unternehmens als technologisch-fortschrittlich und kundenorientiert widerspiegeln soll.
- **Business-to-Business (B2B)** Apps liefern Dienstleistungen, die Geschäftspartnern effizienten und mobilen Zugriff auf Informationen und Prozesse liefern, um besser vernetzt zu arbeiten.
- **Business-to-Employees (B2E)** Apps sind ausschließlich für Mitarbeiter des Unternehmens gedacht. Hier kann das Unternehmen in voller technologischen Breite den Mitarbeitern zur Verfügung stehen.

4.5.1 Interaktionsbreite von Anwendungen

Mobile Unternehmensanwendungen weisen unterschiedliche Charakteristiken auf. (Unhelkkar & Murugesan, 2010) haben sich an einer Klassifizierung versucht und 5 Kategorien gebildet. Diese Kategorien bauen aufeinander auf und erweitern schrittweise den Umfang der Apps. Die ersten beiden Kategorien beschreiben einen einseitigen Informationskanal.

- **mobile broadcast** Apps, aus der ersten Kategorie dienen ausschließlich dazu dem Benutzer Informationen bereitzustellen.
- **mobile information** Die zweite Kategorie beinhaltet eine Vorauswahl des Benutzers, sodass dieser personalisierte Inhalte erhält.
- **mobile transaction** Neben einer nur lesenden Funktionalität wird eine schreibende hinzugefügt, sodass der Benutzer Aktionen durchführen kann.
- **mobile operation** In dieser Kategorie wird der Umfang der Aktionsbreite des Benutzer erhöht, indem der Benutzer mit mehreren Serverbackends interagiert.
- **mobile collaboration** Die letzte Kategorie unterstützt die agile und dynamische Zusammenarbeit mehrerer Benutzer.

4.6 App-Formate

Wie bereits angerissen, ist die Vielfalt der mobilen Endgeräte groß. Sie verfügen über verschiedene Displaygrößen, unterschiedliche Ausstattung und Qualität von Sensoren, und sprechen zudem eine ganz andere Sprache, gemeint ist die Ausstattung mit einem anderen Betriebssystem. Zur Erstellung nativer Apps stellt jedes mobiles Betriebssystem ihr eigenes Entwicklungswerkzeug zur Verfügung, und der Programmcode einer App für ein Betriebssystem, lässt sich ohne Mehraufwand nicht in Programmcode für ein anderes mobiles Betriebssystem konvertieren. Aus dieser Problematik ist eine Auseinandersetzung mit verschiedenen App-Formaten erforderlich, die unter gewissen Randbedingungen, den Mehraufwand signifikant reduzieren können. In der Literatur wird zwischen nativen Apps, Web-Apps und hybriden Apps unterschieden.

- **Native Apps** erfahren die vollumfassende Unterstützung des mobilen Betriebssystems. Diese Apps starten gewohnt zügig, und mit kurzer Verzögerungszeit verrichten sie die vom Benutzer gewünschten Aktionen. Sie sind vollständig in das Ökosystem des Handys integriert und erfüllen meist alle Anforderungen an Design und Layout, die der mobile Betriebssystemanbieter vorgibt. So findet sich der Benutzer in einer ihm vertrauten Umgebung, und intuitives und damit produktives Arbeiten wird gefordert. Manuelle Anmeldevorgänge entfallen in den meisten Situation vollständig. Desweiteren erfährt die App, neben der nahtlosen Integration, auch die volle Hardwareunterstützung. Sensoren können feingranular gesteuert werden. Beispielsweise kann der Gps-Sender so konfiguriert werden, dass eine höhere Genauigkeit der Ortsdaten geliefert wird. Der Nachteil ist, für jedes zu untersützende mobile Betriebssystem werden Entwickler mit entsprechenden Programmierkenntnissen benötigt. Der Vertrieb geschieht meist über einen App-Store, und Updates müssen entweder vom Benutzer oder von einer zentralen Stelle durchgesetzt werden.
- **Web-Apps** werden über einen auf dem mobilen Endgerät vorhanden Webbrowser aufgerufen. Die App ist damit nicht in das Ökosystem des mobilen Endgerätes integriert. Teilweise lassen sich Shortcuts auf dem Homescreen ablegen, die ein App-Symbol darstellen. Anstatt eine App zu öffnen, startet ein Webbrowser und navigiert zu der Internetadresse auf dem die Web-App veröffentlicht wurde. Die Anmeldung kann unterstützt durch Zertifikate automatisiert ablaufen, evtl. sind aber manuelle Anmeldungen unumgänglich. Das Design der Web-App könnte mit Mühe und Erkennung des mobilen Endgerätes angepasst werden. Allerdings sind nicht das individuelle eingestellte Design und Layout feststellbar, sodass selbst mit

erheblichem Aufwand nicht das individuelle und intuitive Benutzererlebnis wie bei nativen Apps vermittelbar ist. Allerdings muss diese App nur einmal erstellt und für keine anderen Plattformen konvertiert werden. Entwickler müssen nicht alle Eigenarten jedes einzelnen zu unterstützenden mobilen Betriebssystems kennen. Desweiteren ist die Veröffentlichung ohne App-Store möglich. Updates müssen nicht verteilt werden, sondern werden wirksam, in dem Moment, nachdem auf der Serverseite das Update aufgespielt wurde.

- **Hybrid Apps** versuchen die Brücke zwischen nativen und Web-Apps zu schlagen, und alle Vorteile in sich zu vereinen. Die Serverarchitektur muss so wie für native Apps gestaltet werden, denn die Web-App kann nicht als hybride App auf einem mobilen Endgerät installiert werden. Bei Verwendung von hybriden Apps, wird Web-App als Clientversion verwendet und unterscheidet sich damit signifikant von der Web-App als Serverversion, wie sie im Punkt zuvor besprochen wurde. In der Clientversion ist darunter zu verstehen, dass das Werkzeug für Web-Apps so verwendet werden kann um das Verhalten einer nativen App zu imitieren. Dies spart Entwicklungskosten, da jedes mobile Endgerät, dass neue mobile Apps installieren kann, auch über einen Webbrowser verfügt, der die Clientversion einer Web-App ausführen kann. Der mobile Webbrowser wird sozusagen als Zwischenschicht zwischen der App und dem mobilen Betriebssystem eingebaut. Seine funktionelle Rolle ist dabei die des Interpreters, der Javascript, Html und Css-Code als ausführbare Anwendung darstellt. Aus diesem Grund werden hybride Apps in einem nativen App-Projekt verpackt, dass aus einem einzigen nativen View besteht, dass als einziges Element ein WebView enthält. Dieses WebView stellt die Clientversion der Web-App zur Verfügung. Durch eine Javascript-Brücke kann durch mittels Javascript mit dem nativen App-Container kommuniziert werden, der der Clientversion der Web-App native Funktionen, wie beispielsweise Zugriff auf Sensoren bereitstellen kann.
- **Vergleich** Die Entwicklung von Web-Apps und hybride Apps sind ökonomisch zu präferieren, da sie weniger breites Fachwissen und weniger Einarbeitung in verschiedene Entwicklersprachen, -tools und Arbeitsabläufe erfordert, oder weniger Mitarbeiter mit unterschiedlichen Schwerpunkten zu beschäftigen sind. Desweiteren entfällt der Wartungsaufwand für mehrere unterschiedliche Versionen der gleichen App, sowie der Kommunikations- und Koordinierungsaufwand zwischen den verschiedenen Entwicklerteams der unterschiedlichen Versionen. Native Apps sind hingegen teurer, durch weitere Entwicklung der mobilen Betriebssystemhersteller,

die immer weitere Integration, in Wischbewegungen, Widgets, und Menüeinstellungen des Betriebssystems sich integrieren lassen, sowie die Integration mit den Sensoren, wie beispielsweise ein Livekamerabild, können der erhöhte Aufwand in ein mehr in das System integriertes Produkt gerechtfertigt sein.

- **Frameworks oder Middleware** Als weiterer Fall sind Frameworks oder Middleware denkbar, die in einer eigenen Sprache auf einer Metaebene, ermäglichen die Funktionalitäten einer App zu implementieren, und durch eine Abbildungsvorschrift, die Implementierung auf verschiedenen unterschiedlichen mobilen Endgerätbetriebssystem lauffähig machen. Für App-Spiele-Entwickler existieren einige Frameworks, die es ermöglichen, Spiellogiken, wie Spielphysik und andere Funktionalitäten aus den angebotenen Spielbibliotheken (engl. libraries) zu benutzen, und für unterschiedliche Betriebssysteme zu kompilieren. Auch Microsoft untersützt seit Visual Studio 2013 [Mic12] die Verwendung von in C++ geschriebenen Bibliotheken zwischen Windows Phone, Android und iOS.

5 Integration mobiler Apps in Unternehmen

Die mobile Softwarelandschaft ist von Heterogenität geprägt. Dies betrifft nicht nur die mobilen Betriebssysteme auf denen die Apps laufen, sondern auch die Serverarchitektur, die Kommunikationsprotokolle, sowie die Nachrichtenaustauschprotokolle. Selbst wenn die Entscheidung fiele beispielsweise nur mobile Apps für das iPhone 6 zu entwickeln, so wäre die Heterogenität zwar minimiert, aber bei weitem nicht aus der Welt geschafft. Die Kommunikation zwischen Client und Server ist für mobile Apps noch nicht standardisiert. Selbst verschiedene mobile Apps von einem Softwarehaus, werden größtenteils unterschiedliche Implementierungen der Kommunikationsschnittstelle aufweisen. Jemand der sich in die API der einen App eingearbeitet hat, kann nicht voraussetzen, dasselbe Schema in einer anderen App der Softwareschmiede wiederzuverwenden. Desweiteren befindet sich Unternehmenssoftware generell dauerhaft in Entwicklung, um auf die sich rasch ändernden Wettbewerbsbedingungen zu adaptieren. Aus diesen Gründen sind mobile Apps meist unabhängig entwickelte Softwaresysteme, deren schmaler Kontext, nur wenig Potential für Skalierbarkeit auf Unternehmensniveau zulässt, vor allem wenn diese mit anderen interagierenden Apps gemeinsam skaliert werden müssen. Enterprise Application Integration (EAI), ist eine Middlewaretechnologie mit dem Ansatz einzelne Anwendungen so zu integrieren, dass eine Kommunikation zwischen allen Anwendungen ermöglicht wird. Sie ermöglicht die Einbindung von Anwenduzngen von Kunden, Partnern und Zulieferern gleichermaßen. Darüberhinaus unterstützt EAI erleichtert das Erstellen von neuen Anwendungen und Services und die Adaption von existierenden. [JP01]

5.1 Herausforderungen von Mobile EAI

Augrund der Vielfältigkeit der mobilen Apps und ihren fortwährenden notwendigen Maßnahmen um wettbewerbsfähig zu bleiben, fordern kontinuierliche Integration. Integration kann auf Daten-, Anwendungs- und Prozessebene stattfinden.

- **Datenintegration** Unterschiedliche Datenbestände werden miteinander verbunden. Entitäten aus verschiedenen Datenbeständen, die dasselbe reale Objekt beschreiben, werden verknüpft.
- **Anwendungsintegration** Unterschiedlichen Anwendungen wird die Kommunikation untereinander erst ermöglicht bzw. durch einheitliches Vorgehen vereinfacht.
- **Prozessintegration** beschreibt auf einer gehobenen Metaebene die Verwaltung und Zusammenarbeit mehrerer Anwendungen. Somit können können Geschäftsprozesse bzw. -aufgaben dargestellt werden, die von unterschiedlichen Anwendungen gemeinsam verarbeitet werden.

Diese Ebenen bauen größtenteils stückweise aufeinander auf. Auf jeder Ebene müssen einzelne Anwendungen individuell integriert werden, sodass diese gemeinsame Schnittstellen verwenden können. Auf jeder Ebene ist mit Heterogenität zu rechnen und eine Lösungssuche um eine gemeinsame Schnittstelle verwenden zu können, verbunden.

5.1.1 Sicherheit

Wie bereits angesprochen ist Sicherheit ein bedeutender Aspekt, der ausreichender Beachtung bedarf. Sicherheitszertifizierungen sind mit erheblichen Zeitaufwand und Kosten verbunden. Aus diesem Grund wird eine Strategie, zur Etablierung von Sicherheitsmechanismen, vorzugsweise ausgehend von der Plattform präferiert. Mobile Apps benötigen unterschiedliche Sicherheitsanforderungen und gewichten ihre Verteilung der CIA Eigenschaften unterschiedlich. Eine Plattformunterstützung sollte dies mit unterschiedlichen Einstellungsmöglichkeiten Rechnung tragen. Das Speichern der Daten auf dem Gerät, sowie die Trennung der Unternehmensdaten und privaten Daten des Angestellten, kann durch MDM Software durchgesetzt werden.

Die Sicherheit kann aufgeteilt werden in Zugriff zu den Daten, den Kommunikationskanal sowie die Speicherung und Verwahrung der Daten auf dem mobilen Endgerät.

Verschiedene Aktionen innerhalb der App benötigen unterschiedliche Sicherheitsstufen. Der Grund dass nicht bei jeder Aktion die höchste Sicherheitsstufe durchgesetzt wird ist, dass dies mehr IT-Kosten verursacht und gravierender: die Benutzererfahrung einschränkt: Kein Benutzer möchte bei jeder Verwendung der App ein langes Passwort eingeben. Auch wenn ein Angesteller eine App benutzt und schon nach 3-5 Minuten ein anderer Angestellter auf seine Reaktion reagiert hat, so möchte er nach dieser Zeit nicht erneut das Passwort eingeben. Diese Zeitintervall wurde von den Sicherheitsrichtlinien des Apple AppStores zum Kauf einer weiteren App verwendet. Bei einem Kauf einer neuen

App muss das Apple-Kennwort eingegeben werden, von da an muss in den nächsten 5 Minuten das Kennwort bei einem weiteren Kauf nicht erneut eingegeben werden. Käufe sind sehr sensible und mit hoher Sicherheitsmechanismen verbundene Aktionen. Diese Sicherheitsstufe bei jeder Aktion durchzusetzen wäre der Benutzererfahrung abträglich. Aus diesem Grund muss für jede Aktion ein Sicherheitslevel, sofern sicherheitskritisch angegeben werden können. Eventuell ist eine Monitoring-Unterstützung sinnvoll, damit Usabilitytests durch Veränderung der Parameter angepasst werden können. Im Nachhinein kann geprüft werden, ob die Sicherheitsstufen, nach Optimierung der Benutzererfahrung, noch ausreichend sind.

Ein weiterer sicherheitsrelevanter Aspekt, ist die weltweite Verfügbarkeit (A von CIA) der Backendservices der mobilen App, der schärfere Sicherheitsmechanismen zur Durchsetzung erfordert. Dazu ist es erforderlich nur nutzungsrelevante Schnittstellen zu öffnen, und generische Schnittstellen zu vermeiden, da diese durch ihre Ausdrucksmächtigkeit mehr Möglichkeit für ungewollte böswillige Manipulation bieten. Die atomaren, zweckorientierten Schnittstellen sollten so konstruiert sein, dass sie die Parameter validieren, evtl. Überprüfen ob diese in dem richtigen Kontext aufgerufen werden, d.h. nicht per brute-force immer wieder, sondern von einer anderen Benutzerseite aufgerufen wurden. Logging, Auditing und eine Ausnahmebehandlungsstrategie, die keine Systeminformationen oder Implementierungsdetails, an den Anwender oder der darunterliegenden abhörbaren Schnittstelle, weitergeben. Techniken um Verfügbarkeit durchzusetzen, sind Redundanz, Offline-Modus mittels Cachingmethoden und asynchrone Kommunikation. [GSV+05] [JMV+11]

5.1.2 Integration

Ein verdrahtenes Netzwerk ist mehr als die Summe seiner Teile. Leider sind die Teile, in Bezug auf mobile Apps, in den seltensten Fällen aufeinander abgestimmt. Deswegen sollten Integrationskomponenten zur Verfügung stehen die sich auf verschiedenste Varianten mobiler Apps anwenden lassen. Dies betrifft interne oder extern entwickelte Applikationen. Auch die verschiedenen Anwendungsformate: native Apps, Web-basiert oder hybride Apps sollten unterstützt werden. Zur Kommunikation werden unterschiedliche Backendserver verwendet. Beispielsweise legacy systeme, interne oder kommerziell-erworbene Serverkomponenten, Middlewaresysteme. Das Hosting der Backends kann innerhalb des Unternehmens (on-premise), außerhalb (managed hosting, private cloud) oder in der public cloud geschehen. Aus diesem Grund müssen Interfaces unter den zu kommunizierenden mobilen Apps vereinbart werden, damit ein Austausch an Informationen ermöglicht

wird. Diese festglegten Interfaces sind evolutionsgemäß Veränderungen unterworfen, sodass Revisionsnummern bei der Kommunikation übermittelt und ausgehandelt werden müssen. Es ist auch denkbar dass für unterschiedliche Abteilungen innerhalb einer Firma einzelne Instanzen laufen, die unterschiedlichen Updatezyklen durch verschiedene Abhängigkeiten mit anderen System unterliegen, sodass es möglich sein sollte, unterschiedliche Versionen von Serverkomponenten und mobilen Clients ausführen und verwalten zu können. Je nach Interaktionsbreite 4.5.1 einzelner mobilen Apps, lassen sich Strategien zur Integration von Offline-Anwendungen mit wenig oder umfangreichen Aufwand umsetzen. Je höher die Interaktionsbreite desto komplexere Synchronisationsinteraktionen sind möglich.

5.1.3 Skalierbarkeit

Ein Aspekt erfolgreicher Apps ist die kurze Reaktionszeit. Skalierbarkeit ist ein Mittel damit, erfolgreiche mobile Apps die einen großen Anwenderzuwachs erfahren, weiterhin die Qualität liefern. Aus diesem Grund sollten schon zu Beginn Maßnahmen getroffen werden, um Anwendungen einfach skalieren zu können. Allerdings sollte die Anwendung schon so konzipiert sein, dass nur relevante Informationen mit dem Backendserver ausgetauscht werden, um die Bandbreite gering zu halten. Dies senkt Kosten in der Netzwerkinfrastruktur, der mobilen Datenverbindung und der Reaktionszeit der Anwendung auch in Gebieten mit schwacher Netzabdeckung durch 3G/EDGE/UMTS-Infrastruktur. Durch Anwendung von Caching kann Kommunikation zu geeigneten Zeitpunkten stattfinden um auf Daten zuzugreifen, die in kurzer Zeit benötigt werden. Auch die Bündelung mehrere Anfragen, zu unterschiedlichen Unternehmensystemen, kann durch Wegfall von Kommunikationspaketen, die Bandbreite reduzieren. Werden die Ergebnisse der unterschiedlichen Anfragen bereits im Backendserver zusammenführt, so können, nur für die Zusammenführung und Auswertung benötigte Informationen aus den einzelnen Serverantworten entfallen, und eine kompaktere Antwort vom Backendserver zur Client-App erfolgen. [PL00] [VRMB11]

5.2 Ansätze für Unternehmensanwendungsintegration

Nach Gregor Hohpe und Bobby Woolf [GH15] dient Integration zur Bereitstellung von einheitlicher Funktionalität durch Zusammenschluss von zuvor voneinander getrenn-

ten Anwendungen. Ein systematisches Vorgehen Integration zu betreiben, ist es die Ausgangslage durch Kriterien zu bestimmen und darauf sinnvolle Integrationsansätze aufzubauen.

5.2.1 Integrationskriterien

Die folgenden Kriterien sind dem Buch "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions" [HW03] entnommen. Um dem Leser weitere Recherche zu erleichtern werden im folgenden die Kriterien im englischen Original verwendet.

- **Application coupling** Kopplung zwischen Anwendungen sollte möglichst gering gehalten werden. Hohe Kopplung verlässt sich auf viele Annahmen über die interne Arbeitsweise von anderen Anwendungen, die durch unabhängige Entwicklung und Evolution der Anwendung schwer zu fortzuführen und zu verwalten sind. Aus diesem Grund sollte geringe Kopplung angestrebt werden und klar definierte Schnittstellen ausgearbeitet sein. Die Ausführung (Arbeitsweise) der Funktionalität sollte für andere Anwendungen irrelevant sein.
- **Integration simplicity** Die Integration einer Anwendung in ein Unternehmen sollte mit geringen Veränderungen an der Anwendung einhergehen. Zur Einfachheit sollte der zur Integration notwendige Programmcode gering gehalten werden. Jedoch kann es notwendig sein mehr Programmcode für die Integration zu schreiben als nötig wäre, wenn dadurch eine bessere Funktionsweise der Integration gewährleistet ist.
- **Integration technology** Je nach Wahl der Integrationstechnik verlangen diese, Software und Hardware, in größerem Umfang wie andere. Diese Hilfsmittel können eine hohe Einarbeitungszeit abverlangen, sowie hohe Kosten verursachen. Desweiteren ist die Gefahr, eines vendor-lockin, groß.
- **Data format extensibility** Anwendungen, die integriert wurden, müssen Informationen in einem ausgehandelten Datenformat austauschen. Eine weitere Möglichkeit wäre ein zwischengeschalteter Übersetzer, der die Nachrichten für andere Anwendungen übersetzt, sollte die Anwendungen nicht in der Lage sein, ein gemeinsames Austauschformat auszuhandeln. Eine verwandte Problematik ist die Evolution von Datenformaten, bei der sich Datentypen oder -strukturen ändern, sowie weitere Informationen gespeichert werden sollen.

- **Data timeliness** Die Aktualität der Informationen, die zwischen Anwendungen ausgetauscht werden, sollte sehr hoch sein. Werden Daten mit zu großer zeitlicher Verzögerung ausgetauscht, so ist das Risiko veraltete Daten zu übermitteln, groß und umso komplexer wird die Integration.
- **Data or functionality** Über den reinen Datenaustausch hinaus, wird gewünscht, dass eine Anwendung Funktionalität anderer Anwendungen aufrufen kann. Doch diese Mechanismen, die wie lokale Funktionsaufrufe abgebildet werden, sind mit Konsequenzen (beispielsweise Datenfluss bzw. Funktionsfluss wandert von lokalem zu remotem Rechner bzw. andersherum und das gegebenfalls mehrmals bis zum Ende der Funktionsausführung) verbunden, die darüber entscheiden ob dieser Integrationsweg erfolgreich beschritten werden kann.
- **Asynchronicity** Synchrone Funktionsaufrufe finden normalerweise lokal statt. Bei remoten Funktionsaufrufen wird vermehr auf Asynchronität gesetzt, da es unwirtschaftlich wäre, zu warten, im Falle einer Netzwerkunerreichbarkeit oder weil die Anwendung ausgelastet ist oder aus anderen Gründen den Befehl nicht innerhalb einer Latenz bearbeiten kann.

5.2.2 Integrationsansätze

Integration kann, laut dem Buch ”Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions” [HW03] durch folgende vier ”integration styles”, die wir als Integrationsansätze bezeichnen wollen, umgesetzt werden.

- **File Transfer** Informationsaustausch findet durch Dateien statt. Einerseits exportieren Anwendungen Daten, die sie mit anderen Anwendungen teilen wollen. Auf der anderen Seite konsumieren Anwendungen die Daten von anderen Anwendungen.
- **Shared Database** Anwendungen teilen Informationen durch Verwendung einer gemeinsamen Datenbank.
- **Remote Procedure Invocation** Anwendungen exportieren einen Teil ihrer Prozeduren, sodass andere Anwendungen diese remote aufrufen können. Dadurch kann Verhalten in einer anderen Anwendung ausgelöst bzw. Informationsfluss stattfinden.
- **Messaging** Jede Anwendung wird an ein Nachrichtensystem angebunden und durch den Austausch von Daten kann Verhalten in einer anderen Anwendung ausgelöst bzw. Informationsfluss stattfinden.

5.2.3 Problematiken

Bei der Integration können jedoch auch Fehler getätigt werden, es gilt diese zu kennen und versuchen diese zu vermeiden. [JP01]

- **Unstrukturierte und komplexe Modelle** Ausnahmebehandlungen führen zu einer Zunahme des Sourcecode. Des Weiteren fällt ein erhöhter Kommunikationsaufwand an, durch Vermittlung zwischen Middleware und verschiedenen integrierten Anwendungen.
- **komplexe Datenmodelle** Bei der Integration werden Datenmodelle nicht nur mit Nutzinhalten erweitert, sondern auch mit Daten, die dazu dienen dass die Daten von mehreren Anwendungen genutzt werden können.
- **Redundanz** Die zu integrierenden Anwendungen werden nicht von der Middleware kontrolliert. Sollte die Anwendung ihr Datenmodell ohne Anpassung der Integrationskonfiguration ändern, so ist es sinnvoll redundante Informationen zu speichern, um die die Datensätze zwischen Middleware und Anwendung dennoch zuordnen zu können.
-

5.2.4 service-orientierte Integration

Service-Oriented Architecture (SOA) ist eine Architektur, die Dienstleistungen anbietet und für die Ansprüche von Unternehmen entworfen wurde. Der Ansatz basiert auf das Bereitstellen von Schnittstellen, die Dienstleistungen ausführen. Jede Schnittstelle stellt einen Vertrag zwischen zwei Kommunikationspartnern dar, und fördert lose Kopplung, Wiederverwendbarkeit, Autonomie, Zusammensetzen von Services, Auffindbarkeit und Abstraktion. Wegen einer klar definierten Funktion (Vertragskonzept) und definierte Zugriffsmöglichkeit für jede Schnittstelle, besteht keine Notwendigkeit die Implementierungsdetails und internen Datenstrukturen zu kennen. [Erl08] [NL04]

5.2.5 resource-orientiert Integration

Resource-Oriented Architecture (ROA) ist eine Architektur, die auf das Verknüpfen von Ressourcen basiert. Eine Ressource ist eine Entität, die eindeutig durch eine oder mehre URIs (uniform resource identifier) gekennzeichnet ist. Jede URI identifiziert jedoch nur eine Ressource. ROA ist eine Rest-Architektur. Kommunikationspartner tauschen

Repräsentationen der Ressourcen aus. Repräsentationen können je nach Kommunikationspartner in Xml, Json oder anderen Dateiformaten verfasst sein. RESTful Web Services sind Services, die nach dem REST Architekturstil konstruiert, und mittels HTTP Protokoll ausgeliefert werden. [GTW10]

REST (representational state transfer) ist eine architektonische Stilrichtung [GTW10] und keine Technologievorgabe. Folgende Eigenschaften zeichen REST im Besonderen aus.

- **Sepration of Concerns (SoA)** wird durch Verwendung von REST gefördert, und somit ermöglicht, dass Client und Server unabhängig voneinander agieren können.
- **Zustandslose Kommunikation** erleichtert die Skalierung.
- **Cachbare Nachrichten** erlaubt die Angabe, dass die Nachricht gespeichert und nicht erneut angefordert werden muss.
- **Schichtenmodell** Dieser Ansatz unterteilt das Softwareprojekt in Schichten. Jede Schicht umfasst eine Abstraktionsebene. Kommunikation findet nur zwischen zwei nebeneinanderliegenden Schichten statt. REST ist meist auf einer Schicht implementiert.
- **Code-On-Demand** erlaubt es, v.a. Javascript-Code als JSONP nachzuladen, und auf Clientseite auszuführen.
- **uniforme Schnittstellen** basieren auf der Identifikation von Ressourcen mittels URI, der Manipulation der Ressourcen mittels Repräsentationen und sich-selbst beschreibenden Nachrichten.

6 Architekturentwurf

Die Referenzarchitektur beschreibt die Komponenten, sowie die Entwurfsmuster und die Verbindungen unter den Komponenten, die notwendig sind, um die mobilen Unternehmensapps und das Unternehmenssystem zu verbinden.

Sie soll als Leitfaden und Analysewerkzeug dienen, mobile Apps leichter einem Unternehmen zur Verfügung zu stellen. Die Referenzarchitektur ist nicht dazu gedacht, eine vollständige Lösung anzubieten. Sie ist nur ein Ansatzpunkt um Unternehmen den Zugang Und die Verwaltung von mobilen Apps zu vereinfachen. Sie soll dazu dienen die Lebenszyklen einzelner Apps zu begleiten, und die Möglichkeit bieten vereinfacht auf selbe Ressourcen zuzugreifen.

6.1 Annahmen

- **version-controlled** Software in Unternehmen sind stetigen Wandel unterworfen. Aus diesem Grund ist eine Versionshistorie wichtig, um die Bearbeitungsschritte nachvollziehen zu können, und evtl. auf frühere Versionen zurückgreifen zu können. Weit verbreitet sind svn und git.
- **cross-platform support** Unterstützung verschiedene mobiler Betriebssystem, vorrangig iOs, Windows Phone und Android sollte gegeben sein, da diese den Hauptanteil an dem mobilen Endgerätemarkt ausmachen.
- **cross-client-technology support** Unterstützung unterschiedlicher Implementierungstechnologien für Client-Apps soll gegeben sein, d.h. es sollen sowohl native Apps, Web-Apps und hybride Apps unterstützt werden.
- **cross-server-technology support** Unterstützung unterschiedlicher Implementierungstechnologien für Serverkomponenten soll gegeben sein, d.h. stand-alone Anwendungen, php-Skripte, jsnode Skripte, sowie CC++, Java, .NET Sprachen, JAVA EE Container, und beliebig weitere Implementierungssprachen.
- **platform-based communication** Kommunikation sollte mittels dem Platform Server aufgebaut werden. Es sollten keine eigenen Verbindungen aufgebaut werden,

da dann ansonst nicht die Mediator- und Middlewarefunktionalitäten des Platform Server zu tragen kommen.

- **cross-hosting support** Es sollen Serverkomponenten in unterschiedlichen Umgeben lauf können und von der Referenzarchitektur kontrolliert werden können. Sei es bei der Anbindung beispielsweise eines Tomcat Application Server oder eines Php-Servers.

6.2 Funktionen

Folgende Funktionen müssen unterstüzt werden.

- **Kommunikation Serverkomponente - Plattformserver** Beide müssen sich finden, und gegenüber ihre Identität bestätigen, damit die Art der Kommunikation geregelt ist. In diesem Fall, dass der Plattformserver weiß, dass er mit der Serverkomponente, und nicht mit der Clientkomponente oder einer Datenquelle kommuniziert.
- **Kommunikation Serverkomponente - Clientkomponente** Die Clientkomponente kommuniziert über den Plattformserver als Mediator mit der Serverkomponente. Aus diesem Grund muss die Clientkomponente identifiziert werden, sodass die Kommunikation entsprechend zur Serverkomponente aufgebaut werden kann.
- **Authentifizierung des Benutzers gegenüber dem Plattformserver** sodass die entsprechenden Datensätze, Einstellungen und Aktionsmöglichkeiten, die Benutzerbasiert geladen werden, und dies auch nur wenn der tatsächlich Zugangsberechtigte seine Identität bestätigt hat.
- **Kommunikation mit einer Unternehmensressource** Die Kommunikation wird durch den Plattformserver aufgebaut. Dieser bietet ein REST-Interface an, und führt die direkte Kommunikation mit der Unternehmensressource.
- **Kommunikation mit einer Unternehmensressource unter Berechtigungsvorlage**
- **Kommunikation mit einer anderen Serverkomponente** Die Kommunikation wird durch den Plattformserver aufgebaut. Dieser bietet ein REST-Interface an, und führt die direkte Kommunikation mit der anderen Serverkomponente. Zu beachten ist, dass die Kommunikation zwischen zwei Serverkomponenten asynchron und von beiden Seiten initiiert werden kann.

- **Kommunikation mit einer Serverkomponente unter Berechtigungsvorlage**

6.3 Anforderungen

Die Anforderungen wurden durch Literaturrecherche in den Kapiteln Spielifizierung im Unternehmensbereich 2, App Store 3, Mobile Apps im Unternehmenskontext 4 und Integration mobiler Apps in Unternehmen 5 , erörtert.

6.3.1 Sicherheit

Sicherheit ist ein wichtiges Thema für Unternehmen, dieses wird jedoch individuell verschieden durchgesetzt. [Ser15] Aus diesem Grund lassen Unternehmen meist Sicherheitsentscheidungen nicht von externen Anbietern bestimmen, sondern setzten die von der IT-Abteilung vorgegebenen Sicherheitsstandards durch. Dies hängt auch von dem Umstand ab, welche Infrastruktur in dem Unternehmen gegeben ist.

6.3.1.1 LDAP

LDAP beispielsweise, wird häufig von Unternehmen genutzt, hält aber den CIA-Anforderungen nicht stand. [OF11]

6.3.1.2 Kerberos

Die deutlich erweiterte Version Kerberos, die in Unternehmen Einzug gefunden hat und noch immer den Status quo hält, ist jedoch auch mit Limitierungen behaftet. Beispielsweise kann durch die Wahl eines zu einfachen Passwortes, die mit Befugnissen ausgestattete Person imitiert werden. Kerberos basiert auf dem Needham and Schroeder Authentifizierungprotokoll, der um ein Ticketsystem erweitert wurde, damit die wiederholte Eingabe eines Passwortes, entfällt. Es wird auch eine cross-realm Authentifizierung unterstützt, d.h. die Authentifizierung eines Benutzers, der bei einem anderen Authentifizierungsserver registriert ist.

6.3.1.3 OAuth

OAuth ist eine Authorisierungframework, dass es Drittanbieteranwendungen ermöglicht definierten Zugriff auf ein Webservice zu erhalten. Der Webservice implementiert den

OAuth, sodass je nach Version und Implementierung die verwendet wird, Sicherheitslücken bei einzelnen Webserviceanbietern auftreten können. Benutzername und Passwort wird nur bei einem Identity Provider, der die Anmeldedaten und das Konto verwaltet eingegeben. Bei Anfrage eines Webservices, sich mit OAuth zu autorisieren, wird die Anfrage auf Clientseite zu dem Identity Provider geleitet. Eine Zustimmung zur Verwendung des Kontos von der Drittanbieterseite, führt zur Generierung und Weiterleitung eines Tokens zu der Drittanbieterseite. Mit Hilfe des Tokens wird der Benutzer autorisiert, da der Webservice von nun an, unter Verwendung des Tokens auf bestimmte Kontoinformationen zugreifen kann. [D15] [Wik15c] Jedoch gilt OAuth 1.0 noch 2.0 als sehr sicher, da beispielsweise OAuth 1.0 durch Abhören des Access Token, Session Swapping und Force-Login CSRF beeinträchtigt war. OAuth 2.0 ist laut Eran Hammer, zu komplex, weniger genau spezifiziert wie Version 1.0, und laut ihm unsicherer. [SB12]

6.3.1.4 Cross-Origin Resource Sharing (CORS)

Cross-Origin Resource Sharing (CORS). Normalerweise unterliegt die Ausführung von Webseiten der Origin Policy. Origin bedeutet hier ursprüngliche Adresse, meistens sind damit verschiedene Domainnamen gemeint. Die Origin Policy erlaubt im auszuführenden Javascript-Code auf der Clientseite, nur die Verwendung von Ressourcen, von der selben Domain, von der der Javascript-Code geladen wurde. Dies dient der Sicherheit der Web-Anwendungen, da bösartige Seiten, beispielsweise nicht mit den Berechtigungen des Benutzer vertrauliche Daten von anderen Webseiten erlangen können. Dies basiert darauf, dass der Benutzer bei einer anderen Webseite angemeldet hat, und die Anmeldesession noch andauert. Erfolgt die Authentifizierung durch Cookies, so werden die Cookies automatisch vom Internetbrowser mitgeschickt. Solange die Session nicht abgelaufen ist, und in dieser Zeit eine bösartige Anwendung versucht private Daten zu laden, wird dies mit der Origin Policy erfolgreich verhindert. Besteht nun aber ein Vertrauensverhältnis zwischen zwei Domains, so ermöglicht die Cross-Origin Policy genau die Verwendung von beispielsweise (asynchron) geladenen Ressourcen von anderen Origins. Hiermit wird die Kommunikation verschiedener Serverkomponenten, die unter verschiedenen Domains laufen, über den Client als Vermittler, ermöglicht.

6.3.1.5 Zusammenfassung

Es gibt eine Vielzahl von Mechanismen und Verfahren Sicherheit zu gewährleisten. Die Anforderungen und Ausgangslage sind jedoch individuell, und muss von jedem Unternehmen, für jeden Anwendungsbereich erneut evaluiert werden. Aus diesem Grund bietet die

Referenzimplementierung eine Schnittstelle an, die Wahl der Sicherheitsverfahren eigenständig festzulegen. Der Standardmechanismus, der von Web-Anwendungen verstanden wird, und cross-client-technology support unterstützt, ist CORS, weswegen dieses in der Referenzimplementierung mit Zuhilfenahme von SSL Anwendung findet.

6.3.2 Integration

Mobile Apps sind in immer mehr Bereichen anzutreffen und mobile Endgeräte stehen in hartem Webkampf gegeneinander. Aus diesen Gründen sollte die Referenzarchitektur unabhängig von den mobilen Betriebssystemen bleiben, sodass Marktänderungen und Evolutionen in den mobilen Betriebssystemen nicht zwangsläufig zu Inkompatibilitäten zur Referenzarchitektur führen. Desweiteren soll der komplette Anwendungsbereich B2C, B2E und B2B unterstützt werden. Die verschiedenen App-Formate, Native App, Web App oder hybride App sollen gemeinsam verwendet werden, und miteinander verknüpft werden können. Desweiteren soll die Integration zu Serviceschnittstellen, unabhängig davon ob sie lokal oder extern stattfindet, gelingen. Das Ziel ist es die Abhängigkeiten zwischen den einzelnen Apps aufzuheben und über die Referenzarchitektur zu vereinen.

6.3.3 Skalierung

Skalierung mit Zuhilfenahme von Cloudanbietern soll von der Referenzarchitektur unterstützt werden. Dies geschieht durch Zwischenspeicherung der Nachrichten auf der Clientseite, sowie durch transaktionale Schnittstellen und Speicherung der Zustandsinformationen auf dem Client. Dies hat zur Folge, dass es für den Client unerheblich ist, wenn er mit der nächsten Anfrage einen anderen Server anspricht, da er seinen Zustand bei jeder Anfrage mitsenden kann, und der Server kein Wissen über die Vorgeschichte des Clients kennen muss. Aufwendiges Verschieben und Übertragen von Sessions von einem zum anderen Server entfällt somit.

6.4 Komponenten und deren Zusammenspiel

Ein wichtiger Aspekt der Referenzarchitektur ist das aufschlüsseln der einzelnen Komponenten und deren Zusammenspiel um den App-Lebenszyklus einschließlich Integration von Unternehmensressourcen zu erleichtern. Durch Einführung von EAPS (Enterprise Application Platform Server), wird eine Lösung vorgeschlagen, um die Anforderungen, die im ersten Kapitel erläutert wurden, zu erfüllen. Das System erfüllt die Anforderungen

durch die Bereitstellung intuitiv verständlichen Schnittstellen, die auf die Belange von Fachexperten zugeschnitten sind. Diese Schnittstellen bauen auf einer überlegt gestalteten bottom-up Architektur API Schicht auf. Dies hat zur Folge, dass Programmierer die von Fachexperten begonnen Programmierprojekte schnell fortführen können, da keine Konvertierung in ein Programmierprojekt erforderlich ist. Der klassische Ansatz von Dependency-Injection wurde um die Bedeutung eines Mediators erweitert, der im EAPS, die späte Bindung und sofortige Rekonfiguration einer App erlaubt. Alle Services sind transaktional und deshalb ist es möglich mit Hilfe des Mediators live zwischen einer zentral verwalteten zu einer verteilten Architektur zu wechseln. Desweiteren wurde das System auf domain-unspezifische Ebenen gebaut, ungewahr über die ausgetauschten Inhalte, und damit in der Lage auch mit neuen und zukünftigen Apps zu interagieren. Der Ansatz unterstützt Evolution, durch Erweiterungsmöglichkeit von existierenden Objekten um weitere Datenfelder oder sogar ganzer Datenobjekte. Neue Datentypen und Interaktionen können auch definiert werden um neue Funktionalität in neue, sowie in bestehende PGAs einbringen zu können.

EAPS besteht aus voneinander unabhängigen Service-Infrastrukturen.

- **Enterprise App Store** bietet einen Katalog zum Durchstöbern und Auffinden von neuen PGAS, deren Installation und Entfernen von mobilen Endgeräten. Wird ein PGA installiert, so wird dieses basierend auf den angemeldeten Benutzer konfiguriert.
- **Enterprise Infrastructure Service** bietet Funktionalitäten zum Hinzufügen und Bearbeiten von PGAs. Es konfiguriert ein neues Git Repository und verbindet sich mit einer existierenden Datenbank und erlaubt weitreichende Konfigurationsmöglichkeiten der PGAs.
- **Enterprise Datawarehouse** bietet Zugriff zu allen gespeicherten, und aufgenommen Daten, die durch Monitoring und Evaluation zusammengetragen wurden.
- **Enterprise App Backend Server** bietet Hosting für die Serverkomponente der PGAs, wenn sie konform der Containerschnittstellen kompiliert wurden.
- **Enterprise Gamification Service** bietet spezielles Services an, um Gamification in verschiedenen Apps mit gleichen Schnittstellen zu verwirklichen. Durch Verwendung gleicher Schnittstellen, können alle Apps von Erweiterungen gleichzeitig profitieren. Darunter die Führung von Spielständen und die Speicherung von Achievements (d.h. einzelne Punkte für bestimmte Handlungen oder Aktionen).

Neue PGAs werden mit der Hilfe von EIS erstellt. Dieses legt ein Git Repository an und legt darin eine Projektmappe an. Entwickler können dieses Projekt auschecken und Änderungen committen. Fachexperten können über EIS Änderungen einpflegen, die durch Änderungen im git Repository umgesetzt werden. Standardmäßig wird ein eigenes Git Repository [Git15] für die Client- bzw. Serverkomponente verwendet. EIS erlaubt es ein oder mehr Zielserver für die Client- und Serverkomponente festzulegen. Mittels 1-click-install werden die korrespondierenden Client- und Serverversionen zu den entsprechenden Zielservern veröffentlicht. Beispielsweise kann die Clientkomponente zu einem einfachen http server oder einem app store server, der weitreichender Funktionalitäten anbietet, veröffentlicht werden. Die Serverkomponente kann auf einem selbstkonfigurierten Server oder auf dem EABS installiert werden.

6.4.1 Enterprise App Store

Es bestehen zwei Anwendersichten. Die Entwickler können neue Apps einstellen und die Produktseite für ihre App, wie beispielsweise auf Amazon promoten. Hierzu dient ein aussagekräfter Titel und eine Beschreibung über die Leistungen der App. Screenshots und Videos können zur besseren Darstellung des Leistungsumfangs hinzugefügt werden. Aus Benutzersicht können alle im App Store vorhanden Apps aufgelistet und durch Kategorien und Filter leichter durchsucht werden. Die Produktseiten der Apps sind einsehbar und die Möglichkeit des Kaufs, der Installation, der Aktualisierung und des Entfernens sind vorhanden. Darüber hinaus besteht die Möglichkeit Bewertungen in Form von 1 bis 5 Sternen abzugeben und ein Kommentar zu hinterlassen.

6.4.2 Enterprise Infrastructure Service

Dieser Dienst ist die Schlüsselkomponente, die den Beginn der Mehrwertschöpfung der Referenzarchitektur bildet. Sie bietet den Apps ein standardisiertes Interface, um auf Ressourcen und Services zuzugreifen. Dazu nimmt sie eine Mediator Position ein. Einige Ressourcen werden auf das Enterprise Datawarehouse abgebildet. Die Mediator Position ist der Mittelpunkt der Monitoring Fähigkeiten die im Enterprise Datawarehouse gespeichert werden.

6.4.2.1 Deployment

Eine wichtige Frage ist, ob der auf einem lokalen Server oder in Cloud betrieben werden soll. Der lokale Server ist etabliert und Fachkenntnisse bestehen in Unternehmen, wie

dieser verwaltet werden muss. Aus Sicherheitsbelangen sprechen oft Gründe für eine lokale Installation. Allerdings sollen mobile Apps, vor allem mobil sein, d.h. von jedem Ort aus angesprochen werden. Große Firmen unterhalten mehrere Standorte und müssten hier die Verwaltung und Organisation aufbringen, an jedem Standort Serverleistung in ausreichender Dimension für die App vorzuhalten. Ein kostengünstiger und ressourcenschonender Ansatz ist es, wenn die eigene Infrastruktur nicht ausreicht, spezialisierte Unternehmen, sprich Cloud-Anbieter, mit dem Deployment zu beauftragen. Dies bringt den Vorteil, dass die weltweiten Standorte zu einer kurzen Antwortzeit bei der Verwendung der mobilen Apps beitragen. Werden später Kooperationen mit anderen Softwareanbietern eingegangen und beide sind in derselben Public oder Private Cloud, kann die Antwortzeit weiterhin sehr gering gehalten, und die Kunden mit einer schnellen Benutzererfahrung von der fortwährenden Benutzung der mobilen App überzeugt werden.

6.4.2.2 Zugangspunkt

Zur Unterstützung von Aspekten, wie beispielsweise Monitoring und Logging, werden alle Zugriffe aus den Enterprise Infrastructure Service, durch einen einzigen Zugangspunkt getätigt. Als Transportprotokoll wird das von allen heutigen mobilen Endgeräten unterstützte Http-Protokoll verwendet. Die Adressierung von Services und Ressourcen geschieht durch URIs. Die Anfragen laufen durch den einzigen Zugangspunkt, dem FrontController [Sof15] und werden von dort zu den einzelnen Controllern (d.h. Servicekomponenten) anhand einer Routingtabelle und evtl. zusätzlicher Logik, geleitet.

6.4.2.3 Mediator

Die Mediatorfunktion wird durch verschiedene Extension-Points ermöglicht. Diese bieten Punkte, an denen anderer Code eingehängt werden kann. Somit kann die Mediatorfunktion auf feingranularer Ebene stattfinden, ohne ursprünglichen Code neuschreiben zu müssen. Ein weiterer Ansatz ist die Verwendung von Pipelines und Filtermethoden. Dadurch können die Aufrufe von Funktionen dynamisch neukonfiguriert werden, ohne bestehenden Code anzutasten, und evtl. ohne Neukompilieren zu müssen. Dadurch wird die Kapselung von Logik in Funktionen gefördert, Erweiterbarkeit und Veränderung der Logik durch Austausch von Funktionen oder Ausführung von zusätzlichen Funktionen bewerkstelligt.

6.4.2.4 Services

Services sind von extern ansprechbare Funktionen, die eine Eingabe entgegennehmen, eine Funktionalität umsetzen, und meistens eine Ausgabe produzieren. Die Services werden durch Controller implementiert.

6.4.2.5 Provider

Services von Drittanbietern, unternehmenseigenen Anwendungen oder Cloudanbietern werden durch Connectoren verbunden und mittels einer universellen Schnittstelle von dem Enterprise Infrastucture Service zur Verfügung gestellt. Ein Service Connector kann also Service Proxy oder Service Gateway implementiert sein. Service Proxies sind zur Anbindung von Nachrichten API, Service Gateways für Ressource API.

6.4.2.6 Clients

Jeder mobile Client unabhängig von dem verwendeten mobilen Betriebssystem, noch welches AppFormat (Nativ, WebApp, Hybrid) verwendet wird, sollen unterstützt werden. Die Anbindung zu dem Enterprise Infrastructure Service geschieht durch das Enterprise Infrastructure Mobile Gateway. Dieses übernimmt die Aufgabe der Authentifizierung, der Verschlüsselung, dem Versand von synchronen und synchronen Nachrichten, und die Synchronisierung der lokalen mit den Datenbeständen auf dem Server.

6.4.3 Enterprise Datawarehouse

Das Enterprise Datawarehouse ist der zentrale Datenspeicher der Referenzarchitektur. Allen Serverkomponenten und mobilen Client-Apps wird geraten, ihn als Datenspeicher zu verwenden. Damit hat er einen globalen Überblick über alle Datenquellen, verbunden mit den weiteren Ressourcen und den Trackingdaten von dem Monitoringdaten, ist er zentraler Ausgangspunkt um mittels Data Mining wertvolle Geschäftsinformationen auszulesen, und darauf aufbauend Strategien für mobile Apps oder in größen Unternehmensdimensionen Entscheidungen zu unterstützen.

6.4.3.1 Enterprise App Backend Server

Der Backendserver bietet eine oder mehre unterschiedliche Serverumgebungen an, die die Ausführung von Serverkomponenten unterstützt. Beispielweise einen Tomcat Application

Server oder einen php-Server. Es wird die Installation, Updates und das Entfernen der Serverkomponente mittels des Backend Server bewerkstelligt.

6.4.3.2 Unternehmenssysteme

Die Unternehmenssysteme, die es einzubinden gilt, sind unter anderem Legacy Systeme, d.h. Altsysteme, die an eine moderne Architektur angeglichen werden müssen, oder neue eingekaufte, wie auch selbst programmierte Systeme. In der heutigen Zeit kommen auch Private und Public Cloud Lösungen, sowie Schnittstellen zu B2B Services von anderen Geschäftspartnern hinzu. Wenn Integrationsansätze scheitern, so verbleibt die Möglichkeit der direkten Anbindung von Datenbanksystemen.

6.4.4 Enterprise Gamification Services

Der Enterprise Gamification Service soll der Beginn sein, allen Apps eine gemeinsame Schnittstelle zu bieten um eine App mit spielerischen Elementen auszustatten ohne eine eigenständige Implementierung zu benötigen. Darüber hinaus sorgt eine gemeinsame Schnittstelle dass Erweiterungen und Auswertungen für eine App ohne großen Aufwand auf andere Apps mit spielerischen Elementen übertragen werden kann. Die Grundlage ist die Verwaltung von Spielerkonten, die von den Anmeldedaten in den einzelnen anderen Anwendungen entkoppelt ist, und so unabhängig von der Evolution einer anderen Anwendung weiter existieren. Es ist denkbar, dass eine andere App eingestellt, ihre spielerischen Elemente verliert, oder dass der Benutzer, zu einem anderen Unternehmen, seine Spieledaten mitnehmen möchte. Nur durch Entkopplung mit den Geschäftsanwendungen, kann ein Unternehmen bereit sein, diese spielerbezogene und unternehmensunkritische Daten an ein anderes Unternehmen weiterzugeben. Desweiteren ist so die Möglichkeit gegeben, mehrere Enterprise Gamification Services unterschiedlicher Anbieter zu verbinden, sodass ein Anwender seine Spieleergebnisse verknüpfen kann, da diese eigenständig von den Anmeldedaten im Unternehmen sind.

6.4.5 Mobile Apps

Es werden verschiedene App-Formate (native App, Web App, hybride App) und Anwenderszenarien (B2B. B2C, B2E) unterstützt. Zur Erfüllung ihrer Aufgaben müssen sie Informationen mit anderen Unternehmenssystem austauschen können. Dies geschieht durch den Enterprise Infrastructure Service. Er ist der Zugangspunkt für die Serverkomponenten zu den anderen System und Ressourcen worüber er den Zugriff verwaltet.

6.4.6 SDK und Tools von mobilen Betriebssystemen

Zur Automatisierung von Aufgaben, beispielsweise dem Kompilieren eines Projektes wird der Compiler aus dem SDK des entsprechenden mobilen Betriebssystem benötigt, unter dem die App laufen soll. Für weitere Services werden andere Komponenten, Tools und Schnittstellen der SDK oder weiteren Programmzusätzen verwendet.

6.5 Entwurfsmuster

In diesem Teil wird erläutert welchen Entwurfsprinzipien hinter dem Gesamtentwurf der Architektur standen.

- **standardisierte Schnittstelle** zum Zugriff auf Unternehmensressourcen mittels des Enterprise Infrastructure Service.
- **erweiterbare Architektur** um Evolutions der Unternehmensapps und den wachsenden Bedürfnissen begegnen zu können.
- **Verwendung von Standards** damit der Grundstein für Kompatibilität gelegt ist, und an die Referenzarchitektur anzuschließende Anwendungen, leicht mit standardisierten Schnittstellen erweitert werden können, um mit der Referenzarchitektur interagieren zu können.
- **Geringe Kopplung** um die Eigenständigkeit der einzelnen Komponenten zu gewährleisten, und Rekonfiguration und individuelle Anpassen der einzelnen Komponenten durchführen zu können.

6.6 Architekturstil

Das Ziel es es die verschiedenen mobile Apps auf gleicher Weise mit den Serverkomponenten bzw. der Referenzarchitektur kommunizieren zu lassen. Durch das geforderte Prinzip, dies durch Verwendung von Standards zu erreichen, ist die Wahl auf die service- und ressourcenorientierten Webstandards gefallen. Der serviceorientierte Architekturstil vereinbart sich gut mit den Unternehmenssystemen, die nach diesem Architekturstil entworfen wurden, und mit den Webstandards, die zu großem Teil serviceorientiert sind. Zum anderen ist der ressourcenorientierte Architekturstil sehr gut auf mobile Apps ausgerichtet, deren SDK nach diesem Stil konzipiert sind. Dieser basiert auf dem Http-Protokoll

mit SSL-Verschlüsselung und einer JSON-API. Die Nachrichten die ausgetauscht werden sind meist in JSON, können aber auch in anderen Dateiformaten verwendet werden.

6.7 Integration

Die Integration der einzelnen Komponente basiert durch Verwendung einer zentralen Einheit, dem Enterprise Infrastructure Service. Er enthält die Verknüpfungen zu Ressourcen und Services und bildet diese mittels einer REST-API ab. Der Zugriff untereinander basiert durch einen Servlet-Controller, der die Anfragen verarbeitet, und im Namen des anfragenden Kommunikationspartner, diese Anfrage durchführt, und die Antwort als Json konvertiert, zurück übermittelt. Zur Kommunikation wir das Http-Protokoll mit SSL-Verschlüssel und unter der Aktivierung von CORS verwendet.

7 Architektur Implementierung

Bevor mit einer Implementierungen begonnen wird, ist es wichtig die Werkzeuge und vorhandenen Technologien zu recherchieren. Sicherheits- und Verwaltungsaspekte spielen für Unternehmen eine große Rolle, sowie Einbindungsmöglichkeit in bestehende IT-Lösungen, ohne eine eigenständige Anwendung, für die alle Aspekte neukonfiguriert werden muss, zu erstellen. Schlussendlich führen teilweise die ausgesuchten Werkzeuge und vorhandene Technologien zur Festlegung auf eine Programmiersprache, damit die Werkzeuge und Technologien verwendet werden können.

Die Implementierung basiert auf der Referenzarchitektur, ist aber mit Einschränkungen und einer Gewichtung auf den Lebenszyklus von Apps, durchgeführt worden.

7.1 Implementierungsübersicht

Wie eingangs erwähnt, ist wegen der Dimension der Referenzarchitektur, eine Umsetzung von einer Person mangels Zeit, Zugriff auf technische Komponenten und Fokussierung auf einzelne hervorzuhebende Komponenten, eine vollständige Umsetzung nicht im Rahmen einer Masterarbeit zu bewältigen. Einige Themen brauchen tiefe Kenntnisse von dem Zusammenspiel und Arbeitsweise von Algorithmen. Nach Meisterung der Thematik ist meist jedoch noch nicht der Weg zur Umsetzbarkeit geebnet. In Unternehmensbereichen herrschen bei tiefgreifenden Systemänderungen komplizierte Installationsroutinen ein, die selbst in Büchern eigenständig behandelt werden. Dies ist ein illustratives Beispiel dass nahelegt warum nicht alle Komponenten umgesetzt, getestet oder lauffähig gemacht werden konnte. Am Ende ist es eine Abwägung der Ressource Zeit, sodass anderen Aufgaben die Zeit gewidmet wird, da ihre Erreichung mehr zum Gesamtziel beitragen.

7.1.1 Unternehmenssysteme

Die Implementierung enthält keine realen Unternehmenssysteme. Selbsterstellte Services dienen zur Emulation von Unternehmenssystemen.

7.1.2 Mobile Apps

Es stehen keine voll-funktionsfähigen Implementierungen von Apps zur Verfügung. Es werden jedoch Services aus Clientsicht aufgerufen um die Funktionsweise der Referenzarchitektur versichern zu können.

7.1.3 Enterprise App Store

beschränkt sich auf die Auflistung des Katalogs und der Installation von Apps.

7.1.4 Enterprise Infrastructure Service

ist das Kernstück der Referenzarchitektur und wird komplett implementiert.

7.1.5 Enterprise Datawarehouse

erlaubt die Speicherung von Gamification Datensätzen.

7.1.6 Enterprise App Backend Server

bietet Hosting für WAR kompilierte Serverkomponenten innerhalb eines Tomcat Application Servers.

7.1.7 Enterprise Gamification Service

verwaltet Benutzerprofile und den Punktestand für ein Unternehmen.

7.2 Werkzeuge und Technologien

Die Referenzarchitektur ist auf Unternehmen fokussiert, und soll da es schon mehrere verschiedene mobile Betriebssysteme unterstützt, auch auf allen gängigen Desktop oder Server Betriebsystemen lauffähig sein. Aus diesem Grund wurde JAVA, und insbesondere JAVA EE gewählt. Aufgrund der langjährigen Erfahrung mit Java Anwendungen und native Unterstützung von Webtechnologien wurde das Spring Framework gewählt. Zur Verwaltung der Projekte wurde das von Google eingeführte Gradle verwendet, das die Maven Projektverwaltung abgelöst hat. Die Datenbankanbindung ist durch Hibernate nahezu uneingeschränkt, in der Entwicklung wurde jedoch der auf allen Plattformen unterstütze MySql Datenbankserver verwendet.

7.3 Plattform Kern

Die grundlegenden Funktionen, die die Plattform bereitstellen muss, ist die Verwaltung des Lebenszyklus der Apps, Benutzerverwaltung und Zugriffsrechteverwaltung.

7.3.1 Lebenszyklus der mobilen Endgeräte

In Anlehnung der SAP AG, die eine mobile device management Lösung namens SAP Afaria [SAP11] entwickelt, kann der mobile Endgerätelebenszyklus (engl. mobile device lifecycle) in drei Phasen unterteilt werden: Bereitstellung (engl. Provision), Betriebsphase (engl. Production) und Außerbetriebnahme (engl. Decomission). Abgrenzend von SAP werden hier die Phasen abstrakter dargestellt.

- **Bereitstellung (engl. Provision)** Wenn ein Gerät, in Firmenbesitz oder gemäß des bring-your-own device (BYOD) Prinzips erstmalig in das Unternehmen gebracht wird, so ist eine Konfiguration notwendig um die Sicherheitsbelange zu erfüllen, indem Passwortschutz und Netzwerkzugriff, sowie grundlegende Unternehmensanwendungen und -daten bereitgestellt werden.
- **Betriebsphase (engl. Production)** Während der Nutzungszeit eines Gerätes innerhalb des Unternehmenskontextes, sind Konfigurations- and Softwarepatches, so wie das Entfern und Installieren von Anwendungen, und Überwachungsmöglichkeiten (engl. monitoring) erforderlich.
- **Außerbetriebnahme (engl. Decomission)** Am Ende der Lebenszeit eines Gerätes, aufgrund von Ersatz, Abhandenkommen, oder im Falle von BYOD durch Ausscheiden des Mitarbeiters aus dem Unternehmen, müssen alle unternehmensrelevanten Informationen entfernt werden.

7.3.2 App-Lebenszyklus

Der App-Lebenszyklus kann in drei Phasen eingeteilt werden.

- Die erste Phase ist die Entwicklungsphase, die mit dem Anlegen eines Applikation-Eintrages beginnt. Eine Applikation besteht entweder nur aus einer Serverkomponente, die ein Webinterface zur Verfügung stellt, oder sie besteht aus einer Server- und ein oder mehreren Clientkomponenten, die das Benutzerinterface zur Verfügung stellen. Die unterschiedlichen Clientkomponenten sind für unterschiedliche Betriebssysteme ausgelegt. Für jede Clientkomponente kann ein Versionsverwaltungspfad

angegeben werden um automatisch die aktuellste Version auf der Plattform bereitzustellen (engl. deployment). Ausgewählte Benutzer können auf die Entwicklungsversion zugreifen und testen. Desweiteren besteht eine Feedbackmöglichkeit um die Entwicklung zu unterstützen.

- Die zweite Phase ist die Installationsphase. In dieser Phase wurde die Applikation genehmigt und kann von der Zielgruppe verwendet werden. Die Applikation ist nun so konfiguriert, dass sie Zugriff auf Services und Daten hat, die im Betriebsmodus (engl. productive mode) laufen.
- Die dritte Phase ist die Wartungsphase (engl. maintenance phase). In dieser Phase werden kontinuierlich Fehler behoben, sowie Anwendungsänderungen vorgenommen. Die gesamte Nutzungsphase wird fast ausschließlich von der Wartungsphase eingenommen.
- Die vierte Phase ist die Ablösephase (engl. retirement phase). Diese Phase markiert das Ende einer Anwendung. Sie wird entweder nicht mehr benötigt, wurde durch eine neue grundlegend überarbeitete Anwendung ersetzt oder die Funktionalität wurde in eine Softwaresuite oder in andere Anwendungen integriert.

7.3.2.1 Entwicklungsphase

Der App-Lebenszyklus begint mit der Entwicklungsphase. Wenn ein Lastenheft für eine neue Anwendung geschrieben und zur Entwicklung freigegeben wurde, so wird für diese Applikation ein Eintrag in dem System angelegt. Der Eintrag erhält den Namen der neuen Anwendung und eine Beschreibung die den Anwendungszweck offenbart. Dieser Applikationseintrag ist der Grundstock für verschiedenste Aktionen und Konfigurationsoptionen innerhalb der Plattform. Dies schließt auch die Anzeige im App Store Katalog, die Verwaltung von Entwicklung und Auslieferung, sowie die Freigabe und Limitierung von Zugriffsrechten von Benutzern. Eine Applikation ist aus einer oder mehreren Komponenten zusammen gesetzt. Innerhalb der Plattform werden drei Komponententypen festgelegt.

- **Nur-Server-Komponente** beschreibt die Komponente für eine Applikation, die nur aus einer Serverkomponente aufgebaut ist, die neben der Logik auch die Benutzeroberfläche enthält.
- **Server-Komponente** beschreibt eine Komponente in einer Applikation, die aus einer Serverkomponente und mindestens einer Clientkomponente besteht. Die

Serverkomponente enthält die Applikationslogik und stellt Schnittstellen bereit, um mit Clientkomponenten kommunizieren zu können.

- **Client-Komponente** beschreibt eine graphische Benutzeroberfläche mit clientseitiger Logik. Eine Applikation kann für unterschiedliche Unternehmensbereiche in einem eigenen Kontext, mit unabhängigem Datenbestand laufen. In einem Kontext, kann jedoch nur eine Serverkomponente existieren, dafür aber mehrere Clientkomponenten. Für jedes unterstützte mobile Betriebssysteme wird eine Clientkomponente entwickelt.

Jede Komponente beschreibt ein Softwareartefakt, dass unabhängig verwaltet wird. Den Pfad, unter dem Sourcecode bzw. der Binärcode abgelegt sind kann für jede Komponente festgelegt werden. Im Falle einer Serverkomponente können unterschiedliche Kontexte und unterschiedliche Ausführungstypen definiert werden. Diese sind entweder **In-Entwicklung** oder **In-Betrieb**. Ein Kontext ist eine eigenständige Ansammlung von Daten, Zuständen und Konfigurationen. Mit der Auftrennung in Kontexten, wird aktiv SaaS (engl. Software as a Service) unterstützt, da für jede Abteilung ein Kontext verwaltet werden kann, falls die Plattform im Unternehmenrechenzentrum (engl. on-premise) betrieben wird. Falls die Plattform in der Public Cloud gehostet wird, so können dort auch Kontexte für verschiedene Unternehmen innerhalb einer Plattform verwaltet werden.

Es wird davon ausgegangen, dass jede Komponente mittels git, svn oder eines anderen Versionsverwaltungsystem verionskontrolliert wird. Aus diesem Grund können vorgefertigte Tags der Plattform genutzt werden um spezifische Versionen einer Komponente anzusprechen. Dies ermöglicht, die Angabe der Version der Server- sowie der Clientkomponente in der Beschreibung des Kontextes, und ermöglicht die Ausführung der stabilen Version im Betriebsmodus, während Entwickler an einer zukünftigen Version im selben Git Repository arbeiten. Desweiteren wird somit andere Abteilungen oder Unternehmen, die sich auf dem selben Plattform Server befinden, die Möglichkeit gegeben erst dann ein Upgrade durchzuführen, wenn sie dazu bereit sind. Die Produktivphase beginnt mit der Erstellung eines neuen geeigneten Kontextes.

Dieser Kontext beinhaltet auch die Information welche Benutzer oder Benutzergruppe die Anwendung benutzen dürfen. Im Kontext werden zusätzlich Datenbankverbindungen und Anschluss an Ressourcen eingerichtet, um die Anwendung mit der produktiven Infrastruktur des Unternehmens zu verbinden.

7.3.3 Benutzerverwaltung

Die Benutzerverwaltung der Plattform gestattet die zentrale Benutzerverwaltung über alle Applikationen hinweg. Jeder Benutzer kann einer oder mehreren Berechtigungsgruppen angehören, die von allen Anwendungen verwendet werden. Diese festgelegten Rechtegruppen sind Administrator, Benutzer und Entwickler. Ein Administrator hat auf alle Konfigurationsaspekte der gesamten Plattform Zugriff aber hat keinen Zugriff individuelle Apps zu benutzen. Entwickler haben Zugriff auf die Entwicklungskonfiguration der Plattform und haben nur Zugang zu den Apps, die sie entwickeln. Benutzer haben nur Zugriff auf Apps, auf denen Ihnen Zugriff gewährt wurde. Wie ersichtlich wurde, erlaubt eine Rechtegruppe nur grobgranularen Zugriff auf verschiedene Bereiche der Plattform.

7.3.4 Zentrale Anmeldedaten

Ein Anwender interagiert nicht nur mit Apps, sondern auch mit der Referenzarchitektur, wenn neue Apps erstellt werden, und dem App Store Server wenn er die Rechteverwaltung übernimmt, wer Zugriff auf bestimmte Apps hat. Neben den Anmeldedaten werden Berechtigungsgruppen zentral verwaltet, um die Rechte in Aufgabengebiete einteilen zu können, sowie die Zusammenarbeit von Benutzern in Apps, die unterschiedliche Rollen tragen, zu unterstützten.

7.3.5 Zugriffsverwaltung

Die Zugriffsverwaltung der Plattform gestattet die Konfiguration des allgemeinen Zugangs zu der Plattform, so wie auch zu den Anwendungen. Darüber hinaus werden auch applikationsspezifische Rechte für individuelle Benutzer unterstützt.

7.3.5.1 Berechtigungsgruppen

Berechtigungsgruppen können nach dem Kontext unterteilt werden. Rechtegruppen sind spezifisch für den Kontext, innerhalb dessen sie benutzt werden. Aus diesem Grund werden verschiedene Rollen für die Plattform, für alle Applikationen und individuell für einzelne Applikationen benötigt.

7.3.5.2 Plattformberechtigungsgruppen

Zugriff zu einzelnen Bereichen der Plattform kann mittels der Rechtegruppe gewährt oder verweigert werden, die der angemeldete Benutzer inne hat.

- **Administrator** hat Zugriff auf die Anmeldedaten und kann Rechtegruppen für einzelne Anmeldedaten setzen.
- **Entwickler** hat Zugang zu dem Entwicklungsbereich.
- **Benutzer** hat nur Zugang zu dem App Store.

7.3.5.3 erweiterbare Plattformrechtegruppen für alle Apps

Innerhalb der Plattform können zusätzliche Rechtegruppen für das gesamte Unternehmen festgelegt werden, und sind für jede Applikations zugreifbar. Dies kann eine HR-Gruppe, eine bestimmte Angestelltengruppe oder eine beliebige andere Gruppe sein. Installation und Zugriff ist für einen Benutzer möglich, wenn dieser der erforderlichen Berechtigungsgruppe angehört. Eine Anwendung für einen HR Angestellten kann nur von einem Benutzer aus der HR-Gruppe installiert werden. Eine Benutzer ohne der HR Gruppe angehörig zu sein, ist nicht in der Lage diese App zu installieren.

7.3.5.4 app-spezifische Berechtigungsgruppen

Wenn eine Applikation sich auf unterschiedliche Berechtigungsgruppen stützt, wie in dem Fall von MyFirstDay, so werden app-spezifische Berechtigungsgruppen verwendet. In MyFirstDay sind dies die Gruppen Admin, Senior und Trainee.

- **Admin** Ein Pendant zu der Berechtigungsgruppe Admin existiert bereits. Allerdings bezieht sich Admin hier nur in der Verwaltung, welche Benutzer aus der Firma Zugriff haben. Eine Erweiterung ist, dass hier bestimmte Administratoren, nur die Anmeldedaten verändern dürfen, von Benutzerkonten, die nur mit der App MyFirstDay funktionieren. Dies ist sinnvoll wenn Besucher oder andere kurzzeitige Gastzugänge geschaffen werden sollen, ohne für die betreffende Person Anmeldedaten für die globale Unternehmensinfrastruktur bereit zu stellen.
- **Senior und Trainee** sind Benutzergruppen, die vermutlich dieselbe Jobposition inne haben. Der Senior mit langjähriger Erfahrung, der den neuen Trainee mit Hilfe von MyFirstDay unterweist. Beide könnten deshalb der HR-Gruppe angehören, unterscheiden sich daher in ihrer Gruppenagehörigkeit nicht um sie auch in der Anwendung unterscheiden zu können. Aus diesem Grund haben beide innerhalb der bestimmten App MyFirstDay unterschiedliche Zugehörigkeit zu den Gruppen Admin und Trainee. Diese beiden Gruppen sind daher app-spezifisch. Aus diesem Grund ist es möglich Benutzern, die die App MyFirstDay installiert haben, auf

der Benutzerverwaltungsseite eine app-spezifische Rolle zuzuweisen. Konkret heißt dies, dass die App die Gruppen vorgibt, die Plattform diese Gruppen erkennt und dadurch ermöglicht innerhalb der Plattformkonfiguration Benutzern diesen Gruppen zuordnen zu können.

- **Trainee**

7.3.5.5 Zugriffsrechte

Zugriffsrechte können für jede App festgelegt und pro Benutzer kann Zugriff gewährt oder verweigert werden.

7.4 Implementierung des Plattformkerns

Als Applikationsserver wird Tomcat 7 verwendet. Für den Plattformkern wird ein JAVA EE Container verwendet und wird als WAR-Datei ausgeliefert.

7.4.1 Datenstrukturen

7.4.1.1 Applikationseintrag

Ein Applikationseintrag definiert eine einzelne Applikation, die in unterschiedlichen Version und betriebsystemspezifischen Implementierungen existieren kann. Aus diesem Grund kann der Applikationseintrag nur auf eine ID und den Applikationsnamen reduziert werden.

7.4.1.2 Applikationskomponente

Eine Applikation, die für die Plattform entworfen wurde, besteht aus einer Serverkomponente und einer oder mehreren optionalen Clientkomponenten, die für verschiedene mobile Betriebssysteme ausgelegt sind. Es wird eine Id, der Fremdschlüssel für den Applikationseintrag, der Komponentenname sowie der Repositorypfad gespeichert.

7.4.1.3 Applikationskontext

Ein Applikationskontext definiert eine lauffähige Installation einer Applikation. Für eine lauffähige Installation wird eine Spezifikation für jede einzelne Applikationskomponente

zusammen mit der verwendeten Versionsnummer benötigt. It stores an id, the application-entry-foreign-id, installation-name, a user-access-table-foreign-id, a role-access-table-foreign-id, an application-component and version-number list, the deployment-path, the servlet-path.

7.4.1.4 Benutzer

Eine allgemeine Benutzertabelle speichert alle Benutzer, die Zugriff auf die Plattform, oder auf einzelne Apps innerhalb der Plattform haben. It stores id, firstname, lastname.

7.4.1.5 Anmeldedaten

Eine gesonderte Anmeldetabelle erlaubt verschiedene Authentifizierungsmechanismen für denselben Benutzer. Eine dem Plattformkern mitgeliefete Anmeldedatentabelle dient der Authentifikation mittels Benutzername und Passwort Kombination. It stores id, user-table-foreign-id ,loginname, password.

7.4.1.6 Applikation-Zugrifftabelle

Every application context has an access table that holds the user-foreign-ids of the user table to determine which user has access to an individual app.

7.4.1.7 Applikation-Berechtigungsgruppentabelle

Every application context can restrict the installation and usage to defined job-roles within the company and therefore provide sparse usage restrictions.

7.4.1.8 Applikation-Kontext-Komponenten-Tabelle

Every application context consists of one or more components and has to define which version for each component is used. It stores the application-context-foreign-id, application-component-foreign-id and the application-component-version-number.

7.4.2 grundlegende Service-Funktionalität des Plattformkerns

7.4.2.1 Auffindbarkeit (engl. discovery)

Bevor die Plattform und die Serverkomponente miteinander kommunizieren können, müssen diese sich zuerst finden. Zusätzlich muss die Clientkomponente die Serverkomponente finden.

7.4.2.2 Plattformserver - Serverkomponentenauffindbarkeit

Das Auffinden des anderen Kommunikationspartner kann von beiden Richtungen aus starten. Die Serverkomponente wird jedoch vom Plattformserver verwaltet, und der Pfad zu der Serverkomponente ist bereits bekannt. Somit ist die Auffindung (engl. discovery) bereits abgeschlossen, wenn der Plattformserver, der Serverkomponente seinen Pfad mitteilen kann, sodass beide Kommunikation iniitieren können.

Auf die Serverkomponente kann ohne zusätzlichen Aufwand zugegriffen werden. Aus diesem Grund werden nur die Optionen der Serverkomponente aufgelistet, den Plattformserver aufzufinden.

- Der **Application-Server-Konfigurations-Ansatz** benutzt ein Propertyname PlattformSERVERADR um auf den Pfad zum Plattformserver zu verweisen. Dies funktioniert für alle Applikationen, die auf demselben Server laufen. Die gehosteten Applikationen können einfach den Wert der globalen Property auslesen.
- **Injektion-durch-Konfigurationsdatei-Ansatz (engl. configuration injection approach)** verwendet eine Konfigurationsdatei der Serverkomponente und fügt dieser den Eintrag **PlattformSERVERADR** hinzu, der von der Serverkomponente gelesen werden kann. Abhängig von der Programmiersprache und dem Projekttyp, können sich der Dateipfad und das Dateiformat der standardmäßigen Konfigurationsdateien unterscheiden. Für eine der Programmiersprache und Projekttyp übergreifenden Kovention, kann eine ASCII txt-Datei mit dem Dateinamen Plattformserveradr.txt dienen, die einzig den Pfad enthält.
- **Injektion-durch-Aufruf-Ansatz (engl. calling injection approach)** verwendet ein festgeletes Interface, das von dem Plattformserver aufgerufen werden kann. Es kann so einfach sein, wie das senden einer Http GET Anfrage, mit der Url als ßervlet path + configuration?Plattformserveradr=http://company.intranet/Plattformserver". Der Zugriff auf diese URLs kann gesichert werden, indem nur Anfragen von localhost erlaubt werden. Die Verwendung eines anderen Ports würde auch die Sicherheit steigern. Der einzige Weg eine höhere Sicherheitsstufe zu erzielen ist der vorherige Aufbau einer Vertrauensbeziehung (engl. trusted relationship). Weil jedoch dann auch die Konfiguration zuvor festgelegt werden könnte, wird dieser Ansatz nicht weiter verfolgt.

7.4.2.3 Clientkomponente - Serverkomponentenauffindbarkeit

Damit die Clientkomponente mit der Serverkomponente kommunizieren kann, muss die Serverkomponente aufgefunden werden.

- **Injektion-durch-Konfigurationsdatei-Ansatz (engl. configuration injection approach)** Der Injektion-durch-Konfigurationsdatei-Ansatz wird wie auf Serverseite ausgeführt, indem eine Konfigurationsdatei verwendet wird. Allerdings werden im mobilen Umfeld alle Dateien einer App zusammen in eine Archivdatei gebündelt und diese Datei signiert. Für Änderungen an einer Datei innerhalb des Archivs müsste das gesamte Archiv neuerstellt und wieder signiert werden. Für eine robuste Lösung muss deshalb Zugriff auf den Sourcecode bestehen, die Konfigurationsdatei geändert werden, das Projekt kompiliert und zum App Store hochgeladen werden, von wo die App an die mobilen Endgeräte ausgeliefert wird.
- **Mobiler-Verteilungs-Ansatz (engl. mobile distribution approach)** verwendet einen von Apple geschaffene unternehmensinterne Lösung zur Verteilung von iOs Apps. Es besteht die Möglichkeit eine xml Konfigurationsdatei an die mobilen Endgeräte auszuliefern und zu erneuern, in der, der Pfad der Serverkomponente aufgeführt ist. Dieser Ansatz funktioniert nur mit mobilen Endgeräten, die unternehmensintern verwaltet werden. Für Besuchergeräte ist dieser Ansatz daher nicht anwendbar.
- **Konfigurations-App-Ansatz (engl. configuration app approach)** verwendet eine App, die bereits auf dem mobilen Endgerät installiert worden sein muss. Diese App hat eine festverdrahtete Adresse zu dem Plattformserver. Neue Apps bauen eine Interprozesskommunikation mit dieser App auf, die wiederum mit dem Plattformserver unter Verwendung der festverdrahteten Adresse kommuniziert, und die aktuelle Adresse für die Serverkomponente, der anfragenden App durch die Interprozesskommunikation mitteilen kann. Der Vorteil liegt darin, dass nur eine App geändert, d.h. durch Auslieferung oder manuell über den App-Store aktualisiert werden muss, sprich die **Konfigurations-App**. Falls einem Nutzer, die Möglichkeit gegeben werden würde, verschiedene Plattformserver eintragen zu können, würden verschiedene Apps, die auf diesem Mechanismus basieren, bei Verwendung eines anderen Plattformserver mit anderen Serverkomponenten kommunizieren, und damit andere Inhalte liefern. Hätten beispielsweise Mitarbeiter, die andere Unternehmensstandorte besuchen, eine andere Plattformserveradresse, so

würden sie nun kontextbezogen Informationen für den aktuellen Standort erhalten, obwohl die Apps nicht dafür ausgelegt waren, geschehe dies transparent.

7.4.2.4 Kommunikationskanäle

Nach dem Auffinden (engl. discovery) erfolgt beidseitige Kommunikation zwischen Serverkomponente und Plattformserver, sowie zwischen Serverkomponente und Clientkomponente.

7.4.2.5 Kommunikation Serverkomponente - Plattformserver

Zur Interprozesskommunikation zwischen Services werden TCP/IP Sockets, CORBA message passing oder Webservices verwendet. Der Plattformserver ist ausgelegt jeden Transportmechanismus und Protokoll zu unterstützen. Vorrangig werden jedoch Webserverkomponenten unterstützt, die trotz der Vielfalt der unterschiedlichen Programmiersprachen, das Webservice Schema unterstützen. Aus diesem Grund ist das Standardprotokoll REST-API mit Unterstützungsmöglichkeit für ODATA. Kommunikation wird nur zur Benutzerverwaltung benötigt. Die beteiligten Aktionen sind Authentifizierung von Benutzern, das Ändern von app-spezifischen Berechtigungsgruppen und das Gewähren oder Einschränken von feingranularen Berechtigungen von Apps.

7.4.2.6 Kommunikation Serverkomponent - Clientkomponente

Die Clientkomponente muss nur mit der Serverkomponente, aber nicht mit dem Plattformserver kommunizieren können. Das Transportprotokoll, sowie das Kommunikationsprotokoll kann frei zwischen Client- und Serverkomponente gewählt werden. Die höheren Ebenen, die auf dem Plattformkern beruhen, sind mittels REST-API und ODATA zugreifbar. Die Three roles can be identified to interact with the Plattform core. The developer group can create new application entries. An application entry describes an application with name. The application entry refers to different OS-implementations of the application with a differentiation of different versions and the distinction between development version for beta and alpha testers and a productive version.

8 Architektur Evaluation

In der Evaluation soll gezeigt werden, soll die Durchführbarkeit [Wik15d], der Lösung für das in der Einleitung beschriebenen Problem, gegeben ist. Dazu wird die Implementierung des Prototyps, der die Kernfunktionalität enthält, herangezogen.

8.1 MyFirstDay

MyFirstDay ist ein productive game, das prototypisch in AMOS, ein Praktikumsseminar unter der Leitung von Prof. Dr. Riehle, von eine Studentengruppe umgesetzt wurde.

8.1.1 Ausgangslage zu MyFirsDay

Jährlich stellen große Firmen mehr als 1000 neue Mitarbeiter (engl. trainees) ein. Für 6 bis 12 Monate werden diese von Paten (engl. senior) betreut. Ein Pate hat meist mehrere neue Mitarbeiter zu betreuen, und erstellt dazu eine Liste von Aufgaben, die von seinen Trainees abgearbeitet werden. Diese Aufgaben gehören der folgenden nicht vollständigen Liste von Kategorien an: Erster Arbeitstag, Netzwerk und Kontakte, externe Kontake, Verwaltung: Erster Tag, Erste Woche, Erster Monat, Tätigkeiten im Arbeitsalltag.

8.1.2 Konzept von der App MyFirstDay

Die Idee basiert auf der Grundlage einer Schnitzeljagd, verbunden mit dem Konzept des Geocaching, dass es neuen Mitarbeitern ermöglicht, selbstständig zu arbeiten un den Paten zu entlasten. Jede Aufgabe enthält dazu Gpsinformationen um den neuen Mitarbeiter zu festgelegten Wegpunkten zu führen. Zusätzlich ist die Karte des Unternehmensgelände in Gebiete aufgeteilt. Wird ein Gebiet betreten, so werden ortsgebundene Informationen angezeigt. Alle Aufträge und zusätzliche Informationen werden dauerhaft in einem Inventar gespeichert und sind uneingeschränt zugreifbar. Jede ausgeführte Tätigkeit innerhalb der My First Day App erhöht den Fortschritt des Benutzers, der sich mit anderen Mitarbeitern auf einer Skala messen kann. Der Pate kann zu jeder Zeit den

Fortschritt seiner zugeteilten neuen Mitarbeiter einsehen, ohne die Vertraulichkeit von sensiblen Daten zu verletzten. Darüber hinaus ist es ihm jeder Zeit möglich, neue Aufgaben und Informationen den neuen Mitarbeitern zu liefern, um das Training anzupassen und das Erreichen der Missionsziele zu unterstützen. Die Auswertung von Spielergebnissen für das productive game MyFirstDay geschieht durch Aktualisierung der Fortschrittsbalken, Analyse der anonymisierten Bewegungsprofile für jede Aufgabe, sowie durch Feedback zu den Aufgaben und ob diese erfolgreich absolviert werden konnten.

8.1.3 MyFirstDay Anbindung an die Referenzarchitektur

Die Unternehmenssysteme verfügen über Services, die von MyFirstDay nutzbar sind, damit keine redundanten Datenbestände geführt werden müssen, und Zeit zur Konfiguration von MyFirstDay eingespart werden kann, sollen diese genutzt werden.

- **Angestellteninformation** Dieser Service liefert eine Liste der Angestellten, sowie einzelne Informationen zu jedem Angestellten.
- **Gebäude- und Rauminformationen** Dieser Service liefert eine Liste der Räume, sowie einzelne Informationen zu jedem Raum.
- **Kontaktinformationen** Dieser Service liefert eine Liste der externen Kontakte, sowie einzelne Informationen zu jedem externen Kontakt.
- **Punktesystem** Dieser Service liefert Informationen zu einem Spieleprofil, und stellt Funktionen bereit neue Spielerfolge mit den erreichten Punkten einzupflegen.

Bevor eine mit hohen Kosten verbundene native App entwickelt wird, soll eine Web App genutzt werden. Diese soll den Mehrwert von MyFirstDay beweisen, und protokollieren zu welchen Anteilen, welches mobile Betriebssysteme genutzt wird. Desweiteren sollen verschiedene Services angebunden werden, die im Laufe der Entwicklung der App, und von zusätzlichen nativen Apps die Verwendung ermöglicht sein soll. Aus diesem Grund wird die Referenzarchitektur als ideale Ausgangsbasis betrachtet um den Proof Of Concept(PoC) darzulegen.

8.2 Werkzeuge und Technologien

Neben den Hilfsmitteln, die in der Architektur Implementierung 7.2 genannt wurden, werden für MyFirstDay JAVA EE mit Verwendung von Gradle zur Projektverwaltung, Git zur Versionskontrolle und Spring zur Programmiererleichterung verwendet.

8.2.1 Client Layout

Die Clientanwendung benutzt ein Model-View-Controller Pattern. Aus diesem Grund müssen Informationen für visuelle Seiten und Elemente gespeichert werden. Zusätzlich muss eine Abbildung zwischen den Daten von dem Server zu den visuellen Elementen, und von der Benutzereingabe zu den visuellen Elemente bestehen, bevor diese erneut zu der Serverkomponente geschickt werden können. Der Controller enthält die Anwendungslogik, die für die Darstellung der Benutzeroberfläche und Kommunikation zwischen Client und Server notwendig ist. Die Auswertungen laufen auf Serverseite, nur Logik zur Verarbeitung der Sensoren des mobilen Endgerätes wird auf dem Client ausgeführt. Die Kommunikation erfolgt nahezu ausschließlich in REST, um größtmögliche Kompatibilität zwischen Client und Serverkomponenten zu erreichen.

8.2.2 Server Layout

Die Serverkomponente verwendet auch das Model-View-Controller Pattern, erwartet jedoch rich clients, die nur mit Services kommunizieren und somit den Overhead der Serveranwendung reduziert, indem nur Datenmodelle und Services per REST-API zugänglich sein müssen. Eine Benutzeroberfläche muss daher von der Serverkomponente nicht zur Verfügung gestellt werden. Es wird angenommen, dass Desktop Clients auch MVC mit Hilfe eines Javascript Frameworks wie AngularJs verwenden.

8.3 Unternehmenssysteme

Es werden keine existierenden Unternehmenssysteme verwendet, sondern diese werden emuliert. Sie sind ebenfalls in Java EE und Spring entwickelt, und stellen unabhängige Serverkomponenten dar. Der persistente Datenspeicher wird durch eigenständige MySql Datenbanken, die mittels Hibernate innerhalb der Anwendung angeschlossen sind, realisiert.

8.3.1 Angestellteninformation

Die Angestellteninformationen können vielgestaltig sein und können dennoch von jeder App verwendet werden. Dazu wird ein Adapter geschrieben und in der Referenzarchitektur installiert. Nach diesem Schritt können alle Apps, die mit dem Enterprise Infrastructure Service sprechen, auf diese Daten zugreifen. Im wesentlichen wird eine Id zur Verfügung

gestellt, der Vor- und Nachname, sowie Kontaktinformationen wie E-Mail, Telefon- und Handyrufnummer sowie Sms.

8.3.2 Gebäude- und Rauminformationen

Unternehmen verwalten meist ein eigenes Informationssystem, das wesentliche Daten zu den Firmengebäuden enthält. Diese Information kann auch mittels eines Adapters integriert werden. Es existieren Strukturen für Gebäude, Räume, Zuordnung von Mitarbeiterarbeitsplätzen, sowie Points-of-Interest (POI).

8.3.3 Kontaktinformationen

Die Kontaktinformationen erhalten wesentliche Informationen zu externen Ansprechpartner, mit denen das Unternehmen Beziehungen unterhält. Nach der Verwendung eines Adapters, sind Vorname- und Nachname, Firma, Ansprechpartner, und Kontaktmöglichkeiten wie E-Mail, Telefon- und Handynummer, sowie Fax abfragbar.

8.3.4 Punktesystem

Der grundlegende Baustein von Spielifizierung (engl. Gamification), ist die Belohnung des Spielers mit einer Punktzahl für eine erfolgreich ausgeführte Leistung. Durch Anbindung an die Enterprise Gamification Komponente, kann für jede Abarbeitung einer Aufgabe in MyFirstDay ein Achievement eingetragen werden. Somit können auch von MyFirstDay mit der bekannten Schnittstelle zur Enterprise Gamification Komponente Spielergebnisse abgefragt werden.

9 Zusammenfassung und Ausblick

9.1 Zusammenfassung

Wie in der Einleitung erwähnt, ist das Ziel der Masterarbeit die Schritte einer App in ihrem Lebenszyklus, vom Anlegen der Projektmappe bis zur Ausführung der Anwendung und darüber hinaus zu unterstützen, indem Rechteverwaltung, Aufspielen von Updates und Änderungen in der Gamificationerfahrung, unterstützt werden. In den folgenden Abschnitten soll die Arbeit kapitelweise kurz zusammengefasst werden.

Gamification ist ein interessantes Thema. Durch Unterstützung der App durch eine leichte Anbindung an ein Gamification Service, ist es ohne hohe Kosten möglich, Gamification in vorgesehenen Apps zu testen. Sollte Gamification nicht den gewünschten Erfolg haben, so hat dies nur wenig Ressourcen gekostet, dafür enthält jedoch der Gamification Service neue Datensätze, die für bessere Auswertungen für andere Apps verwendet werden können.

Der Verteilung von Apps mittels eines App Store, ist ein von Kunden verstandenes und intuitives Konzept, neue Apps zu installieren, zu entfernen und sogar einzukaufen. Unternehmen sind nicht mehr gezwungen, lang anlaufende Softwarekäufe zu tätigen, sondern können bequem Volumenlizenzen oder sogar Einzellizenzen erwerben. So können beispielsweise Productive Games nur für einzelne Angestellte, und auch nur wenn dies nötig ist, sofort per App Store einkaufen. Durch die geringen einmaligen oder per In-App-Purchase wiederkehrenden Zahlungen, ist dies ohne bürokratischen Aufwand durchführbar.

Mobile Apps im Unternehmenskontext ist ein weites Feld. Der Sicherheitsaspekt wurde erörtert und ein Verfahren gewählt, dass die Sicherheitsstufe von Unternehmen berücksichtigt. Der Verwaltungsbereich der mobilen Endgeräte, sowie der App wurden dargestellt. Der Lebenszyklus einer App betrifft alle Aktionen die im Laufe des Lebens einer App anfallen. Dazu zählt auch die Integration und Kommunikation mit anderen Komponenten. Der Vorteil von mobilen Apps ist ihr Unterstützung von Kontext, sodass relevante Informationen ohne zusätzliche Benutzereingabe angezeigt werden können, sofern dies im Bereich der technischen Durchführbarkeit liegt, wie beispielsweise bei orts-

gebundenen Aktionen. Die Apps können für unterschiedliche Anwenderkreise bestimmt sein, intern oder extern, im Falle von Kunden oder Geschäftspartnern. Abschließend wurde ein Überblick über die unterschiedlichen Formate von Apps gegeben.

Im Kapitel zur Integration von mobilen Apps erfolgte eine technische Auseinandersetzung über relevante Kriterien bei der Integration von Apps ins Unternehmen. Dazu zählt die Gewährleistung der Unternehmenssicherheit, die Integrationsverfahren, und die Skalierbarkeit der Integration.

In dem Architekturentwurf wurde zur Erreichung der Ziele, die Referenzarchitektur in einen App Store, Infrastructure Service, Datawarehouse und Gamification Service eingeteilt. Diese stellen eigenständige Komponenten dar, die auf unterschiedlichen Server oder Rechenzentren betrieben werden können. Jede Komponente kann daher auch von einem anderen Anbieter betrieben werden. So kann beispielsweise der App Store von einem großen marktführenden Unternehmen geführt werden, der sich im Vertrieb von Softwareanwendungen einen Namen gemacht hat, bzw. Sicherheitseinstufen von Apps vornimmt und beispielsweise virenfreie und malwarefreie Apps garantieren kann. Der Gamification Service kann von einem Spielunternehmen angeboten werden, dass im Bereich von Spielifizierung ihren Schwerpunkt gesetzt hat. Das Datawarehouse kann von einem Statistik und Analyse Unternehmen betrieben werden, mit hohen Rechenkapazitäten um laufzeitintensive Berechnungen durchführen zu können. Der Backend App Server hostet, die Serverkomponente und kann in mehreren Rechenzentren gleichzeitig gehostet werden, um Skalierbarkeit und kurze Latenzzeiten zu gewährleisten. Der Infrastructure Service sollte von jedem Softwareentwicklungsunternehmen intern betrieben werden, da es sie bei dem Lebenszyklus jeder App unterstützt.

In der Implementierung wurde erläutert welche Technologien zum Einsatz kamen und wie technische Details gelöst wurden. Sie wurde auf wesentliche Kompetenzen der Referenzarchitektur beschränkt, da sie breit aufgestellt und durchdacht wurde, sodass der Leistungsumfang für einen einzelnen Bearbeiter nicht innerhalb des Zeitraumes lag, der für diese Masterarbeit angedacht war. Die Implementierung liefert jedoch die Grundkonzepte, des Anlegens eines neuen Projektes einschließlich Verwaltung mittels Git. Das Projekt kann automatisch gebaut und auf den App Backendserver übertragen werden. Eine Anbindung an den App Store ist auch gegeben, sodass ein Benutzer die App herunterladen und mit der Serverkomponente auf dem Backendserver kommunizieren kann.

In der Evaluation wurde eine exemplarische MyFirstDay Testanwendung erstellt, die Angestelltendaten verwendet, um die Benutzer als Anwender für die App einzutragen und

auf ihren Namen zugreifen zu können. Die Anbindung der externen Kontakte dient für den Anlauf von externen Kontakten innerhalb von MyFirstDay. Raum- und Gebäudeinformationen wurden mit den Aufgaben integriert um sicher den Weg zu ortsgebunden Zielen zu finden. Zur Spielifizierung von MyFirstDay wurde der Gamification Service genutzt der das Spielprofil des Anwender verwaltet und ihm ermöglicht Punkte zu sammeln.

9.2 Ausblick

Gamification ist ein neuer Trend der immer noch nicht ausreichend erforscht wurde. Die Erweiterung des Gamification Service um weitere Spielelemente und -mechaniken in Zusammenhang mit einer Usability Studie könnte diesen Bereich weiter erforschen.

App Stores sind ein sehr junges Verteilungskonzept, evtl. ist die Einteilung nach Benutzern oder Zugehörigkeiten zu Unternehmen oder bestimmten Geschäftsrollen von Vorteil, um verschiedene Features, beispielsweise die Vorkonfiguration der App selbstständig bei Auslieferung schon zu tätigen, und nicht erst nach der Installation auf dem mobilen Endgerät.

Zur Integration der Apps kann die Erforschung des Zusammenspiels von verschiedenen Productive Games und Geschäftsanwendungen dienen, um mehr standardisierte Schnittstellen zu schaffen, die es ermöglichen, dass sich unbekannte Productive Games dynamisch finden und gegenseitig Spielergebnisse oder Informationen zu erledigende Aufgaben austauschen können. Eventuell ist eine Aufgabe in einer App weniger Punkte wert, wenn eine Aufgabe aus einer anderen App in Kürze erledigt werden muss, oder es wird ein Hinweis geliefert, der den Angestellten in die andere App umleitet.

10 Wortschatz

Dieses Kapitel dient zum nachschlagen von häufig verwendeten Begriffen.

Clientkomponente
eine App oder eine andere Softwarekomponente, die dazu ausgelegt ist mit einer Serverkomponente, die in einem Rechenzentrum läuft, zu kommunizieren. Das wichtige Merkmal ist, dass die Clientkomponente und Serverkomponente paarweise und zum gemeinsamen Zweck entwickelt wurden.

Enterprise App Store
bietet einen Katalog zum Durchstöbern und Auffinden von neuen PGAS, deren Installation und Entfernen von mobilen Endgeräten. Wird ein PGA installiert, so wird dieses basierend auf den angemeldeten Benutzer konfiguriert.

Enterprise Infrastructure Service
bietet Funktionalitäten zum Hinzufügen und Bearbeiten von PGAs. Es konfiguriert ein neues Git Repository und verbindet sich mit einer existierenden Datenbank und erlaubt weitreichende Konfigurationsmöglichkeiten der PGAs.

Enterprise Datawarehouse
bietet Zugriff zu allen gespeicherten, und aufgenommen Daten, die durch Monitoring und Evaluation zusammengetragen wurden.

Enterprise App Backend Server
bietet Hosting für die Serverkomponente der PGAs, wenn sie konform der Containerschnittstellen kompiliert wurden.

Enterprise Gamification Service
bietet Services zur Spielifizierung an. Darunter die Führung von Spielständen und die

Speicherung von Achievements (d.h. einzelne Punkte für bestimmte Handlungen oder Aktionen).

PG
Productive Game

PGA
Productive Games App

Serverkomponente
Die Serverkomponente, ist eine Anwendung die Anfragen von Clientkomponenten bearbeitet. Das wichtige Merkmal ist, dass die Clientkomponente und Serverkomponente paarweise und zum gemeinsamen Zweck entwickelt wurden.

Spieler
Eine Benutzer, der eine App mit Spielifizierungslementen benutzt.

Appendices

Literaturverzeichnis

[Abt87] Clark C Abt. *Serious games*. University Press of America, 1987.

[Bri12] Patricia Titus, Jon Kuhn, Brian Duckering. State of mobility. 2012.

[Chr12] Stefan Christmann. Mobiles Internet im Unternehmenskontext. 2012.

[D15] Ed. D, Hardt. The oauth 2.0 authorization framework, 2015.

[DDKN11] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, pages 9–15. ACM, 2011.

[EOM09] William Enck, Machigar Ongtang, and Patrick McDaniel. Understanding android security. *IEEE security & privacy*, (1):50–57, 2009.

[Erl08] Thomas Erl. *Soa: principles of service design*, volume 1. Prentice Hall Upper Saddle River, 2008.

[FLM+10] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 281–287. ACM, 2010.

[GGR13] Arnab Ghosh, Prashant Kumar Gajar, and Shashikant Rai. Bring your own device (byod): Security risks and mitigating strategies. *Journal of Global Research in Computer Science*, 4(4):62–70, 2013.

[GH15] Bobby Woolf Gregor Hohpe. Enterprise integration patterns, 2015.

[Git15] Git. Git, 2015.

[GMBV12] Michael Goul, Olivera Marjanovic, Susan Baxley, and Karen Vizecky. Managing the enterprise business intelligence app store: Sentiment analysis

supported requirements engineering. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 4168–4177. IEEE, 2012.

[GSSDV12] Andrea Giessmann, Katarina Stanoevska-Slabeva, and Bastiaan De Visser. Mobile enterprise applications–current state and future directions. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 1363–1372. IEEE, 2012.

[GSV+05] Omar MA Gadir, Kartik Subbanna, Ananda R Vayyala, Hariprasad Shanmugam, Amod P Bodas, Tarun Kumar Tripathy, Ravi S Indurkar, and Kurma H Rao. High-availability cluster virtual server system, September 13 2005. US Patent 6,944,785.

[GTW10] Dominique Guinard, Vlad Trifa, and Erik Wilde. A resource oriented architecture for the web of things. In *Internet of Things (IOT), 2010*, pages 1–8. IEEE, 2010.

[HPM13] Open new doors - transform to a mobile computing environment. Technical report, Hewlett-Packard Development Company, 2013.

[HW03] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[JBF09] Slinger Jansen, Sjaak Brinkkemper, and Anthony Finkelstein. Business network management as a survival strategy: A tale of two software ecosystems. In *IWSECO@ ICSR*, 2009.

[JMV+11] Jinseong Jeon, Kristopher K Micinski, Jeffrey A Vaughan, Nikhilesh Reddy, Yixin Zhu, Jeffrey S Foster, and Todd Millstein. Dr. android and mr. hide: Fine-grained security policies on unmodified android. 2011.

[JP01] Paul Johannesson and Erik Perjons. Design principles for process modelling in enterprise application integration. *information systems*, 26(3):165–184, 2001.

[Kim10] Kristofer Kimbler. App store strategies for service providers. In *Intelligence in Next Generation Networks (ICIN), 2010 14th International Conference on*, pages 1–5. IEEE, 2010.

[KVYM15] AV Kamasheva, ER Valeev, R Kh Yagudin, and KR Maksimova. Usage of gamification theory for increase motivation of employees. *Mediterranean Journal of Social Sciences*, 6(1S3):77, 2015.

[LLL07] Xiuwu Liao, Yuan Li, and Bing Lu. A model for selecting an erp system based on linguistic information processing. *Information Systems*, 32(7):1005–1017, 2007.

[Ltd15] Canonical Ltd. Ubuntu phone, 2015.

[Med09] Konradin Mediengruppe. Einsatz von erp-lösungen in der industrie. *Konradin Mediengruppe, Leinfelden Echterdingen*, 2009.

[Mic12] Microsoft. Cross-platform development in visual studio, 2012.

[NL04] Eric Newcomer and Greg Lomow. *Understanding SOA with web services (independent technology guides)*. Addison-Wesley Professional, 2004.

[OF11] Charlie Obimbo and Benjamin Ferriman. Vulnerabilities of ldap as an authentication service. *Journal of Information Security*, 2(04):151, 2011.

[PL00] Wei Peng and Xi-Cheng Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 129–130. IEEE Press, 2000.

[SAP11] Enabling enterprise mobility for healthcare professionals - choose the technology platform and applications to support clinical care operations. Technical report, SAP AG, 2011.

[SB12] San-Tsai Sun and Konstantin Beznosov. The devil is in the (implementation) details: an empirical analysis of oauth sso systems. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 378–390. ACM, 2012.

[Ser15] Serverfault.com. Ssh key authentication using ldap, 2015.

[She12] Mike Shema. *Hacking web apps: detecting and preventing web application security problems*. Newnes, 2012.

[Sof15] Pivotal Software. 17. web mvc framework, 2015.

[SSVE04] Andrew Simmonds, Peter Sandilands, and Louis Van Ekert. An ontology for network security attacks. In *Applied Computing*, pages 317–323. Springer, 2004.

[TJ07] George Tuvell and Chunyu Jiang. System and method of reporting and visualizing malware on mobile networks, October 9 2007. US Patent App. 11/869,729.

[TMD12] Jennifer Thom, David Millen, and Joan DiMicco. Removing gamification from an enterprise sns. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1067–1070, New York, NY, USA, 2012. ACM.

[Tra] Mathias Tralau. Beschreibung myfirstday. AUDI.

[Tra15] Dipl.-Wi.-Ing. Mathias Tralau. Productive Games Projektseite, 2015.

[VAD08] Luis Von Ahn and Laura Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[VRMB11] Luis M Vaquero, Luis Rodero-Merino, and Rajkumar Buyya. Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review*, 41(1):45–52, 2011.

[Wik15a] Wikipedia. Mobile application managment, 2015.

[Wik15b] Wikipedia. Mobile device managment, 2015.

[Wik15c] Wikipedia. Oauth, 2015.

[Wik15d] Wikipedia. Proof of concept, 2015.