Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Rene Kretschmer

Spezifikation, Entwicklung und Integration eines modularen, Datenbank gestützten Fortbildungssystems in Docker mit Anbindung an ein Active-Directory

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Rene Kretschmer

Spezifikation, Entwicklung und Integration eines modularen, Datenbank gestützten Fortbildungssystems in Docker mit Anbindung an ein Active-Directory

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Ulrike Steffens
Zweitgutachter: Prof. Stefan Sarstedt

Eingereicht am: 1. November 2018

Rene Kretschmer

Thema der Arbeit

Spezifikation, Entwicklung und Integration eines modularen, Datenbank gestützten Fortbildungssystems in Docker mit Anbindung an ein Active-Directory

Stichworte

Microservice, Spring Boot, Angular, Docker, Kubernetes, Container, REST, Decomposing Monolith

Kurzzusammenfassung

Dieses Dokument befasst sich mit der Analyse einer bestehenden monolithischen Anwendung und dem Ablösen dieser durch eine Microservice Lösung. Basierend auf dieser Analyse werden die Fachlichkeiten der Anwendung klar spezifiziert und eine eingehende Risikoanalyse durchgeführt. Ein großer Teil befasst sich mit dem Entwurf der neuen Komponenten und deren Schnittstellen für die Microservice-Architektur. Zu dem Entwurf der neuen Software werden auch Abnahmetests definiert. Abschließend wird der Einsatz von Containern als Deployment Methode für Microservice-Architekturen beleuchtet.

Rene Kretschmer

Title of the paper

Specification, Development and Integration of a modular, database supported training management system in Docker with Active-Directory support

Keywords

Microservice, Spring Boot, Angular, Docker, Kubernetes, Container, REST, Decomposing Monolith

Abstract

This document describes the analysis and replacement of an existing monolithic application with a microservice based solution. Building on the analysis the domain model is specified and a risk analysis is developed. A big chunk of this work is dedicated to designing and specifying the new microservice componentes and interfaces. To complete the design of the new software, acceptance tests are defined. Lastly containers as a means of deploying microservice-based software are discussed.

Inhaltsverzeichnis

1 Motivation **1**
1.1 Krankenhaus Buchholz und Winsen GmbH . . . 1
1.2 Struktur der Bachelorarbeit . . . 1
1.3 Ziel der Arbeit . . . 2

2 Analyse **3**
2.1 Erfassung der bestehenden Umgebung . . . 3
2.2 User Rollen . . . 4
2.2.1 Berechtigungen altes System . . . 5
2.2.2 Berechtigungen neues System . . . 5
2.3 Use Cases . . . 9
2.3.1 Einloggen am System . . . 9
2.3.2 Abmelden von einer Veranstaltung . . . 10
2.3.3 Erstellen einer Veranstaltung . . . 10
2.3.4 Stammdaten-Pflege . . . 11
2.3.5 Teilnehmen an einer Veranstaltung . . . 12
2.4 Zielbestimmungen . . . 13

3 Risikoanalyse **16**
3.1 Definition . . . 16
3.2 Kritische Funktionen . . . 17
3.3 Erwartete Risiken . . . 17

4 Spezifikation der Fachlichkeit **20**
4.1 Benutzer und Veranstaltungen . . . 20
4.2 Fachliches Datenmodell . . . 21

5 Komponenten und Schnittstellen **23**
5.1 Komponenten . . . 23
5.1.1 Festlegen der einzelnen Komponenten . . . 23
5.2 Microservice Architektur . . . 25
5.3 REST-Endpunkte . . . 29
5.3.1 Main API . . . 30
5.3.2 Auth API . . . 31
5.3.3 Bookings, Events, Locations, Users . . . 31
5.3.4 Logging . . . 31

6 Docker **33**
6.1 Warum Container? 33
6.2 Docker und FBM 34

7 Abnahmetests **36**
7.1 Warteliste 36
7.2 Import der AD Benutzer 37
7.3 Buchen von Veranstaltungen 39

8 Fazit und Ausblick **41**

1 Motivation

1.1 Krankenhaus Buchholz und Winsen GmbH

Diese Bachelorarbeit entsteht aus dem Projekt ein Service des bestehenden Intranet abzulösen. In den Krankenhäusern Buchholz und Winsen GmbH wird eine Web-Plattform für das noch bestehende Intranet betrieben. Diese Web-Plattform entspricht sowohl optisch, als auch technologisch nicht mehr dem Stand der Technik und wird in großen Teilen in einen Sharepoint überführt. Für alle statischen Dokumente ist der Sharepoint eine geeignete Lösung. Das im Intranet enthaltene Planungsmodul für interne Fortbildungen lässt sich hingegen schwierig nativ in Sharepoint abbilden.
Dieses Fortbildungsmodul (fortan "FBM"genannt) ist die Grundlage dieser Bachelorarbeit.
Um diese zukünftige Lösung wartbar und agil zu halten, soll ein Microservice Ansatz verfolgt werden.

1.2 Struktur der Bachelorarbeit

Basierend auf dieser Arbeit soll das neue FBM implementiert und im Unternehmen bereitgestellt werden. Um möglichst umfassend darauf vorzubereiten, behandelt diese Arbeit einige wichtige Themen bei der Migration solch eines Systems. Der Schwerpunkt liegt dabei zwar auf Kapitel 5, dennoch geht es mehr darum möglichst viele Bereiche abzudecken, als ein Thema in voller Tiefe zu beleuchten. Kapitel 1 umreißt den Ursprung, grob die Anforderungen an dieses Projekt und beschreibt das Ziel der Arbeit.
Kapitel 2 befasst sich mit der umfangreichen Analyse des bestehenden Systems und der Anpassung der erfassten Elemente für die neue Lösung.
In Kapitel 3 wird eine Risikoanalyse begonnen, auf die im Verlauf des Projektes aufgebaut werden kann.
Kapitel 4 befasst sich mit der Spezifikation der Fachlichkeit der Anwendung als Vorbereitung für den Entwurf der neuen Architektur.
In Kapitel 5 geht es umfassend um den Entwurf der neuen Software, um die einzelnen Komponenten und deren Schnittstellen.

Kapitel 6 beleuchtet die Vorzüge von Containerisierung im Zusammenspiel mit einer Microservice-Architektur.
In Kapitel 7 werden Abnahmetests definiert, welche den Erfolg der Arbeit bzw. des Projektes messen sollen.
Kapitel 8 schließt die Arbeit mit einem Fazit ab.

1.3 Ziel der Arbeit

Ziel dieser Bachelorarbeit soll es sein im Rahmen des Software-Entwicklungsprozesses eine umfangreich spezifizierte Grundlage für die Entwicklung des FBM zu schaffen. Dabei sollen alle notwendigen Schnittstellen, Daten und Berechtigungen in Zusammenarbeit mit den Anwendern definiert und festgehalten werden. Dabei zu berücksichtigen sind die in der Firma vorhandenen technischen Anforderungen sowie die Anregungen und Wünsche der verantwortlichen Mitarbeiter der Firma für das interne Fortbildungswesen.

2 Analyse

2.1 Erfassung der bestehenden Umgebung

Da es sich bei dem Projekt um eine Reimplementierung bzw. eine Überarbeitung eines bestehenden Systems handelt ist der erste Schritt sich die bestehende Lösung anzuschauen. Anhand des User-Interfaces kann ein schematisches Komponentendiagramm (Abbildung 2.1) abgeleitet werden. Die bestehende Lösung bedient sich einer Active-Directory Schnittstelle, um zum einen die für die Veranstaltung in Frage kommenden Standorte auszulesen, als auch die Benutzerdaten zu füllen.
Insgesamt gibt es drei Hauptkomponenten die miteinander agieren: Veranstaltung, Benutzer und Anmeldung. Dabei ist das Zusammenspiel simpel, eine Veranstaltung hat mehrere Teilnehmer (*repräsentiert durch eine Anmeldung*), in einer Anmeldung wird auf einen Benutzer verwiesen. Da die Interna der Software nicht bekannt sind, kann natürlich nicht ausgeschlossen werden, dass die Interaktionen komplexer abgebildet sind. Dazu kommt, dass nicht alle möglichen Daten zu einer Veranstaltung eingegeben werden. Da es sich beim der bisher eingesetzten Software um ein monolithisches System handelt, gibt es praktisch keine Schnittstellen zu anderen Systemen. Der einzige Kontakt findet über eine Schnittstelle zum Active-Directory statt, wobei hier nur alle Benutzer-Daten in einem täglichen Import kopiert werden.

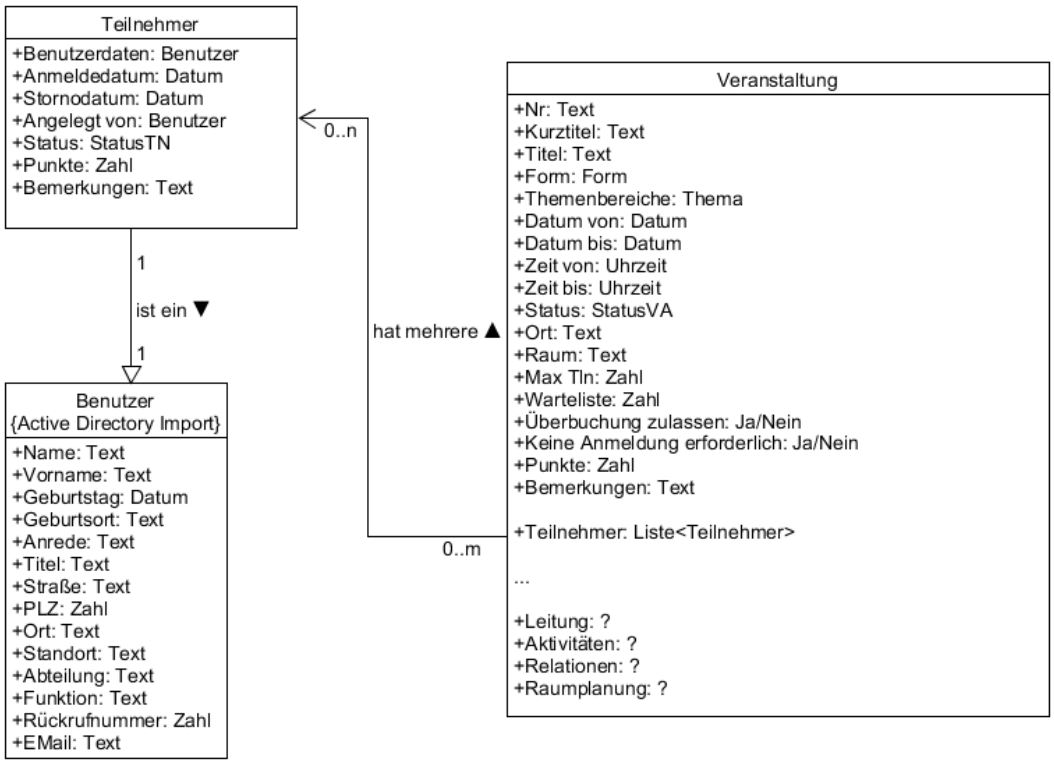

Abbildung 2.1: Komponentendiagramm nach der Analyse (*nicht vollständig*). Zur besseren Übersicht sind die möglichen Werte der Enumerationen **Thema**, **StatusVA** und **StatusTN** hier nicht aufgelistet.

Bei allen Veranstaltungen wurden die Daten zur Leitung, Aktivitäten, Relationen oder der Raumplanung nicht gepflegt. In Rücksprache mit der verantwortlichen Mitarbeiterin im Unternehmen konnten noch andere Daten identifiziert werden, die entweder gar nicht oder mit doppelten Daten gefüllt werden. Die importierten Benutzerdaten sind unvollständig, da z.B. die Adressdaten oder das Geburtsdatum im Active-Directory gar nicht gepflegt werden.

2.2 User Rollen

Bei der Identifizierung von User Rollen geht es darum Berechtigungen zu identifizieren, die für eine Gruppe von Nutzern gelten. Eine Berechtigung ist die Kombination aus einer Ressource

und einer darauf ausführbaren Aktion (siehe TS17, S.3f. Aufbau einer Berechtigung). Diese Berechtigungen einzeln jedem Benutzer zu erteilen ist sehr aufwändig, sobald man mehrere Benutzer mit den gleichen Berechtigungen verwaltet. Wird eine Ressource mit einer Aktion hinzugefügt auf die alle Benutzer Zugriff bekommen sollen, muss allen Benutzern diese Berechtigung erteilt werden. Arbeitet man mit Rollen, haben alle Benutzer mit gleichen Berechtigungen eine gemeinsame Rolle und dieser kann die neue Ressource mit der entsprechenden Aktion hinzugefügt werden. So muss nur an einer Stelle eine Berechtigung hinzugefügt werden, um alle Nutzer dieser Rolle anzupassen (siehe TS17, S. 43).

2.2.1 Berechtigungen altes System

Grundsätzlich wurde bei der Analyse des alten Systems die Ablaufverfolgung bzw. Role Mining angewandt (TS17, S.53f.). Bei der Analyse der Berechtigungen im bestehenden System ist klar geworden, dass Aufgrund des minimalen Funktionsumfangs (Erstellen, Teilnehmen, nicht mehr Teilnehmen von Veranstaltungen) keine Einschränkungen für Benutzer bestand. Jeder Benutzer kann Veranstaltungen anlegen oder löschen, Veranstaltungen beitreten und seine Anmeldung entfernen. Darüber hinaus kann jeder Benutzer jeden anderen Benutzer für eine Veranstaltung anmelden. Dies war ein gewünschtes Feature, da nicht jeder Mitarbeiter einen Zugang zum System hat, aber dennoch an gewissen Veranstaltungen teilnehmen möchte. Die Anmeldung für diese Veranstaltung wird dann von der Abteilungsleitung für den entsprechenden Mitarbeiter durchgeführt.
Alle Stammdaten werden über ein separates Tool gepflegt, welches auf dem Server der Software läuft und somit nur die Administratoren Zugriff darauf haben. Stammdaten sind hier nur Listeneinträge für Drop-Down Menüs oder Auswahlen. Die Benutzerdaten basieren auf den Benutzer-Stammdaten des Active-Directory und entziehen sich somit komplett der Administration der Intranet-Software.
Durch die Trennung der Administration und der eigentlichen Anwendung, gibt es im Frontend effektiv nur die Ressource 'Veranstaltung' mit den Aktionen 'erstellen', 'löschen' und die Ressource 'Anmeldung' (im Sinne von: Ein Benutzer meldet sich für eine Veranstaltung an) mit den Aktionen 'erstellen' und 'löschen'. Natürlich müssen alle Benutzer das Recht besitzen alle statischen Ressourcen (z.B. Themenbereiche oder Orte) zu lesen.

2.2.2 Berechtigungen neues System

Da das neue System als One Page Application präsentiert wird und im Umfang dieser Applikation auch alle Verwaltungstools mitbringen soll, erfordert es diffizilere Berechtigungen als die

alte Software. Mit folgenden Ressourcen und Aktionen kann in der neuen Software interagiert werden:

Veranstaltung: Veranstaltungen können angezeigt (hierbei wird unterschieden zwischen der Anzeige von ***aktiven*** und ***inaktiven*** Veranstaltungen) , erstellt und bearbeitet (hierbei wird unterschieden, ob der Benutzer Referent der Veranstaltung ist oder nicht) werden.

Anmeldung: Anmeldungen können angezeigt, erstellt (hierbei wird unterschieden zwischen der eigenen Anmeldung eines Benutzer und der Anmeldung eines anderen Benutzers), bearbeitet und gelöscht werden.

Standort: Standorte können angezeigt, erstellt und gelöscht werden.[1]

Benutzerprofil: Benutzerprofile können angezeigt, erstellt und bearbeitet werden.

Raum: Räume können angezeigt, erstellt und gelöscht werden.[1]

Thema: Themen können angezeigt, erstellt und gelöscht werden.[1]

Externer Benutzer: Externe Benutzer können erstellt, bearbeitet und gelöscht werden.

Active-Directory: Schnittstellen zu Active-Directories können angezeigt, erstellt und gelöscht werden.[1]

Log: Logs können erstellt und angezeigt werden.

Diese Berechtigungen lassen sich in drei Gruppen unterteilen. Es existieren die Anwender, welche die Funktionen der Applikation nur konsumieren wollen. Es gibt die fachliche Administration der Anwendung, welche sich mit dem Erstellen und Bearbeiten der fachlichen Inhalte befasst. Die letzte Gruppe befasst sich mit der technischen Administration der Applikation. Durch diese Gruppe werden die Grundlagen für die Funktion der Applikation erzeugt und bearbeitet.
In der Gruppe der Anwender werden folgende Berechtigungen benötigt:
Die Anzeige von aktiven Veranstaltungen, Anmeldungen, Standorten, Räumen und Themen ist für alle Benutzer erlaubt, da es sich hierbei um öffentliche zugängliche Informationen handelt. Die Schnittstellen zu den Active-Directories müssen auch von allen Benutzern angezeigt werden können, da diese für die Anmeldung genutzt werden. Dies formt die Rolle 'Jeder', alle

[1] Das Bearbeiten wurde bewusst nicht aufgezählt, da die hier genannten Ressourcen nur als statische Informationen dienen. Bearbeiten kann durch löschen und neu erstellen emuliert werden ohne referentielle Integrität zu verletzen.

Personen, die in der Lage sind auf die Applikation zuzugreifen fallen in diese Gruppe.
Sobald eine Person sich am System authentifiziert hat, ist sie mindestens ein 'Benutzer'. Diese Rolle unterscheidet sich nur geringfügig von der Rolle 'Jeder'. Benutzern ist es zusätzlich erlaubt eine Anmeldung für sich selbst zu einer Veranstaltung zu erstellen, ihr eigenes Benutzerprofil anzuzeigen und zu bearbeiten.

Die fachliche Administration erfordert zusätzlich das Erstellen von Veranstaltungen, Standorten, Räumen, Themen und externer Benutzer. Das Bearbeiten von Veranstaltungen, das Erstellen von Anmeldungen für andere Benutzer sowie das anzeigen von inaktiven Veranstaltungen gehört auch zu den Berechtigungen der fachlichen Administration. Die fachliche Administration ist durch die Rolle 'Verwalter' abgebildet.
Ein Teil der fachlichen Administration liegt bei den Referenten einer Veranstaltung, da diese ihre eigenen Veranstaltungen bearbeiten dürfen. Die Rolle 'Referent' hat deswegen die einzelne zusätzliche Berechtigung, dass sie eigene Veranstaltungen bearbeiten darf.

Die Gruppe der technischen Administration beinhaltet die Rollen 'Administrator' und 'System'. Die Rolle 'Administrator' hat die Berechtigungen externe Benutzer zu erstellen, bearbeiten und zu löschen, sowie Schnittstellen zu Active-Directories zu erstellen und zu löschen.

Die Rolle 'System' hat die Berechtigungen Logs und Benutzerprofile zu erstellen, da dies automatisch im Backend geschieht und keine der menschlichen Rollen Zugriff auf diese Funktionen hat.

Dieses Rollenkonzept ist hierarchisch aufgebaut, sodass jede 'höhere' Rolle alle Berechtigungen der 'unteren' Rollen erhält. Die Hierarchie ist wie folgt aufgebaut:
Jeder < Benutzer < Referent < Verwalter < Administrator < System

Abbildung 2.2: Übersicht der einzelnen Ressourcen und deren Aktionen, farblich kodiert nach den Rollen. Standort, Raum und Thema sind zur besseren Übersicht zusammengefasst.

2.3 Use Cases

In diesem Abschnitt werden beispielhaft Anwendungsfälle des FBM beschrieben. Die hier getroffene Auswahl stellt die umfangreicheren Anwendungsfälle da. Im Anschluss sind die Zielbestimmungen des FBM aufgelistet. Alle aufgelisteten Use Cases und Zielbestimmungen wurden in Zusammenarbeit mit den Mitarbeitern der innerbetrieblichen Fortbildungen erarbeitet.

2.3.1 Einloggen am System

Beschreibung

- Ein Benutzer meldet sich mit seinen Zugangsdaten am System an

Auslöser

- Ein Anwender möchte sich am System einloggen, um Funktionen zu nutzen, die nur angemeldeten Benutzer vorenthalten sind.

Vorbedingungen

- Der Benutzer verfügt über Zugangsdaten zum System.

Nachbedingungen

- Der Benutzer ist im System angemeldet (Es erscheint die Meldung 'Sie haben sich erfolgreich angemeldet!').

Standardablauf

1. Der Benutzer öffnet die Maske zur Anmeldung.
2. Der Benutzer gibt seine Zugangsdaten (Benutzername und Passwort) ein.
3. Der Benutzer wählt seinen Standort aus und drückt/klickt auf den 'Anmelden'-Button.

Fehlerfälle

- Es erscheint die Fehlermeldung "Benutzername oder Kennwort ist falsch!".
  - Die Kombination aus Benutzer und Kennwort ist falsch.

  - Das Benutzerkonto ist nicht vorhanden.
  - Die Kombination aus Benutzer und Standort ist falsch.

2.3.2 Abmelden von einer Veranstaltung

Beschreibung

- Ein Benutzer meldet sich von einer Veranstaltung als Teilnehmer ab.

Auslöser

- Der Benutzer möchte sich von einer Veranstaltung abmelden.

Vorbedingungen

- Der Benutzer ist am System angemeldet (siehe Use Case 'Einloggen am System').
- Es existieren Veranstaltungen.
- Der Benutzer ist an dieser Veranstaltung angemeldet.
- Die Veranstaltung hat noch nicht begonnen.

Nachbedingungen

- Der Benutzer taucht nicht mehr auf der Liste der Teilnehmer oder der Wartenden auf (Es erscheint die Meldung 'Erfolgreich abgemeldet').

Standardablauf

1. Der Benutzer klickt nach der Anmeldung auf den Haken auf einem Kärtchen einer Veranstaltung.
2. Der Benutzer bestätigt seine Abmeldung, da dadurch Wartende nachrutschen können.

2.3.3 Erstellen einer Veranstaltung

Beschreibung

- Ein Verwalter oder Administrator erstellt eine neue Veranstaltung.

Auslöser

- Es soll eine neue Veranstaltung erstellt werden.

Vorbedingungen

- Der Benutzer ist am System angemeldet (siehe Use Case 'Einloggen am System') und hat entweder die Rolle Verwalter oder die Rolle Administrator.

Nachbedingungen

- Eine neue Veranstaltung wurde angelegt (Es erscheint die Meldung 'Veranstaltung angelegt').

Standardablauf

1. Der Benutzer klickt nach der Anmeldung auf das Plus in der unteren rechten Ecke, um die Maske für eine neue Veranstaltung zu öffnen.
2. In der Maske müssen alle Pflichtfelder ausgefüllt werden. Pflichtfelder sind: Titel, Beschreibungstext, Referent, Teilnehmerzahl, Thema, Veranstaltungsort, mindestens ein Datum als Termin.
3. Der Benutzer klickt unten rechts auf 'Veranstaltung erstellen'.
4. Der Benutzer bestätigt in der nächsten Maske das Speichern der neuen Veranstaltung.

Fehlerfälle

- Es erscheint die Fehlermeldung "Referenten oder Teilnehmer Stammdaten unvollständig! Bitte wenden Sie sich an die IT-Abteilung.". Es wurde ein Benutzer aus dem Active-Directory importiert, der nicht alle notwendigen Pflichtfelder gefüllt hat (z.B. E-Mail Adresse oder Telefonnummer).

2.3.4 Stammdaten-Pflege

Beschreibung

- Ein Verwalter oder Administrator erstellt oder löscht ein Standort, Raum oder Thema. In diesem Beispiel: Erstellen eines Raumes.

Auslöser

- Es soll eine neuer Raum zu einem Standort erstellt werden.

Vorbedingungen

- Der Benutzer ist am System angemeldet (siehe Use Case 'Einloggen am System') und hat entweder die Rolle Verwalter oder die Rolle Administrator.
- Es existiert ein Standort, dem der Raum zugeordnet werden kann.

Nachbedingungen

- Ein neuer Raum wurde angelegt (Es erscheint die Meldung 'Raum erfolgreich angelegt').

Standardablauf

1. Der Benutzer klickt nach der Anmeldung auf den Button 'Admin-Panel' in der oberen rechten Ecke, um das Admin-Panel zu öffnen.
2. Im Admin-Panel wählt der Benutzer den Reiter Räume aus.
3. Auf der linken Seite kann ein Name für den neuen Raum eingetragen werden. Darunter wird der dazugehöroge Standort ausgewählt.
4. Der Benutzer klickt unten links auf 'Raum erstellen'.

2.3.5 Teilnehmen an einer Veranstaltung

Beschreibung

- Ein Benutzer meldet sich an einer Veranstaltung als Teilnehmer an.

Auslöser

- Der Benutzer möchte an einer Veranstaltung teilnehmen.

Vorbedingungen

- Der Benutzer ist am System angemeldet (siehe Use Case 'Einloggen am System').
- Es existieren Veranstaltungen.

- Der Benutzer ist noch nicht an dieser Veranstaltung angemeldet.
- Die Veranstaltung hat noch nicht begonnen und ist aktiv.

Nachbedingungen

- Der Benutzer wird in der Liste der Teilnehmer oder der Wartenden angezeigt (Es erscheint die Meldung 'Erfolgreich angemeldet').

Standardablauf

1. Der Benutzer klickt nach der Anmeldung auf das Plus Symbol auf einem Kärtchen einer Veranstaltung.

2.4 Zielbestimmungen

Aus der eingehenden Analyse und den Gesprächen mit der verantwortlichen Mitarbeiterin lassen sich die Zielbestimmungen für das Projekt ableiten.

Musskriterien	
Nummer	**Beschreibung**
MK10	Es gibt verschiedene Rollen: User, Referent, Verwalter, Administrator
MK20	Mitarbeiter können sich mit ihrem Domänen-Login anmelden
MK25	Beim Domänen-Login sollen vorgegebene Daten zu dem Benutzer aus dem AD übernommen werden
MK30	Mitarbeiter können sich abmelden
MK40	User können ihre Daten bearbeiten (persönliche Mail und Telefonnummer)
MK50	User können sich für Veranstaltungen anmelden
MK60	User können sich von Veranstaltungen abmelden
MK70	Referenten können ihre eigenen Veranstaltungen erstellen/bearbeiten
MK80	Verwalter können alle Veranstaltungen erstellen/bearbeiten
MK90	Verwalter können andere User für eine Veranstaltung anmelden
MK100	User können alle Veranstaltungen sehen, ohne sich anzumelden
MK110	Referenten oder Verwalter sollen Veranstaltungen kopieren können
MK120	Abmeldungen von Veranstaltungen sollen gespeichert werden
MK130	Externe Referenten müssen im System angelegt werden können
MK140	Das System muss in der Lage sein, eine Warteliste automatisch zu verwalten. Sobald die Anzahl der Teilnehmer, die der zur Verfügung stehenden Plätze übersteigt, werden die Anmeldung in einer Warteliste gehalten. Wenn wieder Plätze frei werden, sollen die Teilnehmer von der Warteliste automatisch in der richtigen Reihenfolge den Teilnehmern hinzugefügt werden.

Tabelle 2.1: Musskriterien der zu entwickelnden Software

Wunschkriterien	
Nummer	**Beschreibung**
WK10	Es sollen Emails verschickt werden bei: Anmeldung, Abmeldung, Nachrutschen, Erinnerung für baldiges Stattfinden der Veranstaltung
WK20	Es sollen Bilder für einzelne Veranstaltungen hochgeladen werden können
WK30	Es sollen Standard-Bilder für Veranstaltungen hinterlegt werden

Tabelle 2.2: Wunschkriterien der zu entwickelnden Software

Abgrenzungskriterien	
Nummer	**Beschreibung**
AK10	Die Verwaltung der Benutzer (bis auf externe Referenten) ist nicht Bestandteil der Software
AK20	Eine vollwertige Datensicherung ist nicht Bestandteil der Software

Tabelle 2.3: Abgrenzungskriterien der zu entwickelnden Software

3 Risikoanalyse

3.1 Definition

"Mit der Risikoanalyse werden Risiken transparent gemacht, geeignete Qualitätssicherungsmaßnahmen abgeleitet sowie Testfälle und Fehler priorisiert"(siehe Fra07, S.64). Da Risikomanagement keine Spezialität im Bereich der Software-Entwicklung ist, sondern universell einsetzbar ist gibt es nicht die Definition des Begriffes Risiko (siehe Bra13, S.22 3.1.1 Risiko). Christof Ebert beschreibt Risiko als "[...] die Auswirkung von Unsicherheit auf Ziele." (siehe Ebe13, S.7) Für diese Arbeit reicht diese Definition, welche Unsicherheiten bestehen bei der Erreichung der in Abschnitt 2.4 definierten Ziele.

Ein Vorgehen für die Durchführung des Risikomanagement beschreibt (SBS02, S.42) mit folgenden Schritten:

1. Erkennung und Beurteilung der Projektrisiken
2. Aufnahme und Beurteilung der Projektkontrollen
3. Analyse der verbleibenden Projektrisiken
4. Bewertung und Umsetzung von Maßnahmen
5. Laufende Verfolgung der Projektrisiken, -kontrollen und -maßnahmen

Ebert beschreibt dies als "Die fünf Elemente des Risikomanagement"(Ebe13, S.18). Ein sich wiederholender Kreislauf von Erkennen, Bewerten, Abschwächen und Kontrollieren der Risikien. Dieser Kreislauf hat im Zentrum die gewählte Strategie des Risikomanagements. Diese Arbeit befasst sich als Grundlage für ein Projekt nur mit den Punkten: Risiken erkennen, bewerten und ggf. abschwächen bzw. Maßnahmen zur Abschwächung vorschlagen. Während der Umsetzung des Projekts muss natürlich weiterhin Risikomanagement betrieben werden, denn Risikomanagement ist ein Prozess der das ganze Projekt begleitet.

3.2 Kritische Funktionen

In der Analyse haben sich folgende Funktionalitäten als äußerst wichtig für das neue System herausgestellt:

1. Import der Active-Directory Benutzerdaten, da ohne die Nutzung der bestehenden Benutzer-Infrastruktur die neue Software nicht praktikabel ist. Langfristiges Ziel ist hier die Etablierung von Single-Sign-On, sodass Benutzer sich automatisch mit ihren Active-Directory Benutzern anmelden.

2. Das Buchen von Veranstaltungen. Hierbei ist zu beachten, dass Anmeldungen nur möglich sein dürfen für Veranstaltungen die noch nicht begonnen haben und noch Plätze für Teilnehmer frei haben.

3. Eine automatisierte Warteliste. Da bisher eine Warteliste nicht existierte, bzw. es manuell durchgeführt wurde, soll eine automatisierte Warteliste zur Verfügung stellen, welche die Teilnehmer anhand des Datums ihrer Anmeldung selbstständig nachrutschen lässt.

Punkt zwei und drei erfordert eine korrekt implementierte Geschäftslogik. Der erste Punkt erfordert eine gut definierte Schnittstelle zum Active-Directory. In Kapitel 5 geht es um die Aufteilung der einzelnen Komponenten in Microservices. Dies hilft dabei wichtige Komponenten isoliert bei der Implementierung zu bearbeiten und entsprechend zu priorisieren, um die hier genannten kritischen Funktionen bevorzugt zu bearbeiten und somit das Risiko zu vermeiden, dass diese Funktionen nicht korrekt arbeiten.

3.3 Erwartete Risiken

Unabhängig von der Kritikalität der Anforderungen gibt es zusätzliche Risiken. Folgend sind einige davon beispielhaft aufgelistet.

Zusammenspiel der verschiedenen Technologien führt zu Schwierigkeiten in der Entwicklung oder dem Betrieb

Eintrittswahrscheinlichkeit: Sehr hoch aufgrund des umfangreichen Technologie-Stacks und der Schnellebigkeit der Technologie.
Auswirkungen: Können sehr schädigend für das Projekt sein, da sehr große Verzögerungen entstehen können, oder Teile der Software nicht mehr lauffähig sind.
Vorbeugende Maßnahmen: Sorgfältige Planung bei Einführung/Konfiguration einer neuen

Technologie. Schrittweises Einführen/Konfigurieren der neuen Technologien, damit bei Problemen der Auslöser schnell eingegrenzt werden kann und im Falle eines Rollbacks der Schaden minimiert werden kann. Das mindern dieses Risikos ist eine weitere Stärke von Microservices. Da die einzelnen Services sehr klein sind und ein spezialisierten Aufgabenbereich abarbeiten, können Änderungen (neue Technologien/Updates) sehr kontrolliert in das System eingebracht werden. Sollte dennoch ein Fehler auftreten, passiert dies im wahrscheinlichsten Fall isoliert und kann schnell beseitigt werden. James Lewis und Martin Fowler sprechen von einem "Design for failure", fällt ein Microservice aus, sollen alle Konsumenten dieses Services so gut wie möglich damit umgehen und im besten Fall mit eingeschränkter Funktion weiterarbeiten und nicht auch ausfallen (siehe LF14).

Die Benutzerdaten aus dem Active-Directory verursachen aufgrund schlechter Datenqualität beim Import Probleme

Eintrittswahrscheinlichkeit: Hohe Eintrittswahrscheinlichkeit, da es sich um ein älteres Active-Directory handelt. Nicht alle Benutzerdaten sind auf dem selben Stand was den Informationsumfang der einzelnen Datensätze betrifft. Zusätzlich zu den nötigen Mitarbeiter-Benutzerdaten existieren z.B. auch Systembenutzer oder RADIUS-Benutzer, welche nicht im FBM zur Anmeldung zur Verfügung stehen dürfen.
Auswirkungen: Die Auswirkungen können sehr schädigend sein, da nicht zu importierende Benutzer aufgrund fehlender Daten die Nutzung des FBM verhindern können (Ein Benutzer kann sich nicht anmelden, da in seinem AD-Datensatz ein Pflichtfeld nicht gepflegt wurde).
Vorbeugende Maßnahmen: Um diesem Risiko vorzubeugen, können alle gültigen Benutzer mit einer Gruppenmitgliedschaft markiert werden, damit beim Zugriff auf das AD nur Benutzer dieser Gruppe gesucht werden. Dadurch kann verhindert werden, dass Systembenutzer für die Anmeldung genutzt werden können. Um unvollständigen Datensätzen vorzubeugen, müssen entweder alle gültigen Benutzer geprüft und nachgebessert werden, oder es besteht die Möglichkeiten sobald ein unvollständiger Datensatz vom System erkannt wird, diesen Vorfall zu loggen, damit die Daten nachgearbeitet werden können. Wenn Benutzer sich nicht an einer Veranstaltung anmelden konnten, war es bisher auch schon möglich sich telefonisch über die Verwaltung der innerbetrieblichen Fortbildungen anzumelden. Dies würde diesen Fehlerfall auch entschärfen.

Die neue Lösung erfüllt nicht die Wünsche der Mitarbeiter/etabliert sich nicht im Unternehmen

Eintrittswahrscheinlichkeit: Gering bis mittel, da die neue Software von ca. 1000 Mitarbeiter verwendet wird.
Auswirkungen: Die Auswirkungen sind schwierig abzuschätzen, da es im Zweifel keine Alternativen gibt. Die Mitarbeiter haben nur diese Software, um sich mit den innerbetrieblichen Fortbildungen auseinander zu setzen. Theoretisch können schädigende Auswirkungen entstehen, wenn dadurch die Quote der Mitarbeiter sinkt, die sich für Fortbildungen anmeldet.
Vorbeugende Maßnahmen: Die Entwicklung der Software wird in einem agilen Format in Zusammenarbeit mit den MitarbeiterInnen der innerbetrieblichen Fortbildungen durchgeführt. Nach der Einführung der Software gibt es Replikations-Schulungen, welche den Mitarbeitern den Einstieg in die neue Software so leicht wie möglich machen sollen. Zusätzlich wird ein On-Screen Tutorial für Mitarbeiter entwickelt. Wenn Sie sich das erste Mal an der Software anmelden, werden ihnen die grundlegenden Funktionen des Systems erklärt.

4 Spezifikation der Fachlichkeit

Im FBM dreht sich alles um Veranstaltungen (Events), Benutzer (Application User) und Anmeldungen an Veranstaltungen (Event Application).

4.1 Benutzer und Veranstaltungen

Eine Veranstaltung hält relevante Informationen für Teilnehmer, wie z.B. eine Beschreibung, ein Veranstaltungsdatum oder einen Veranstaltungsort. Zu einer Veranstaltung gehören auch eine Reihe von Anmeldungen oder Anmeldungen auf der Warteliste.

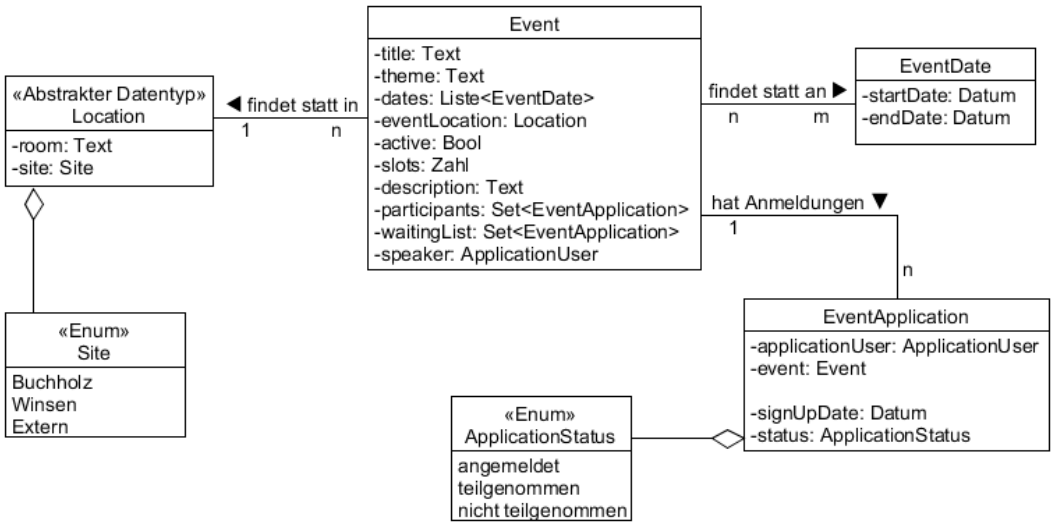

Abbildung 4.1: Datenmodell einer Veranstaltung

Benutzer (siehe Abbildung 4.2) sind wie folgt modelliert: Es existieren Informationen zum Benutzer selbst, wie z.B. Benutzername, Vorname oder E-Mail Adresse. Zusätzlich erhält jeder Benutzer eine explizite Rolle und eine Liste mit Veranstaltungen, die er für sich favorisiert hat.

Ein Benutzer hat die Möglichkeit sich für eine Veranstaltung anzumelden und somit eine Anmeldung zu generieren. Diese Anmeldung gehört zu einem bestimmten Benutzer und einer Veranstaltung, zusätzlich existiert ein Status der signalisiert, ob ein Benutzer angemeldet ist oder teilgenommen hat.

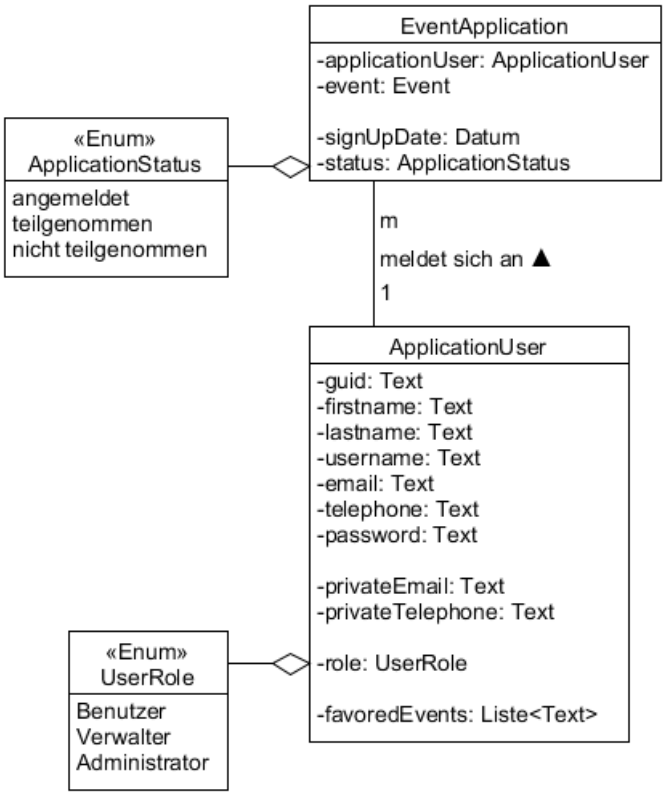

Abbildung 4.2: Datenmodell eines Benutzers

4.2 Fachliches Datenmodell

Zusammengefügt ergibt sich das Fachliche Datenmodell. Erweitert wurden die Veranstaltungen um ein Log, welches Aktionen (z.B. Anmeldungen oder Abmeldungen) auf Veranstaltungen protokolliert. Außerdem wurde das Thema der Veranstaltung (Event Theme) als Enum modelliert. Zusätzlich existiert eine Beziehung zwischen dem Referenten der Veranstaltung (Speaker) und dem Benutzer.

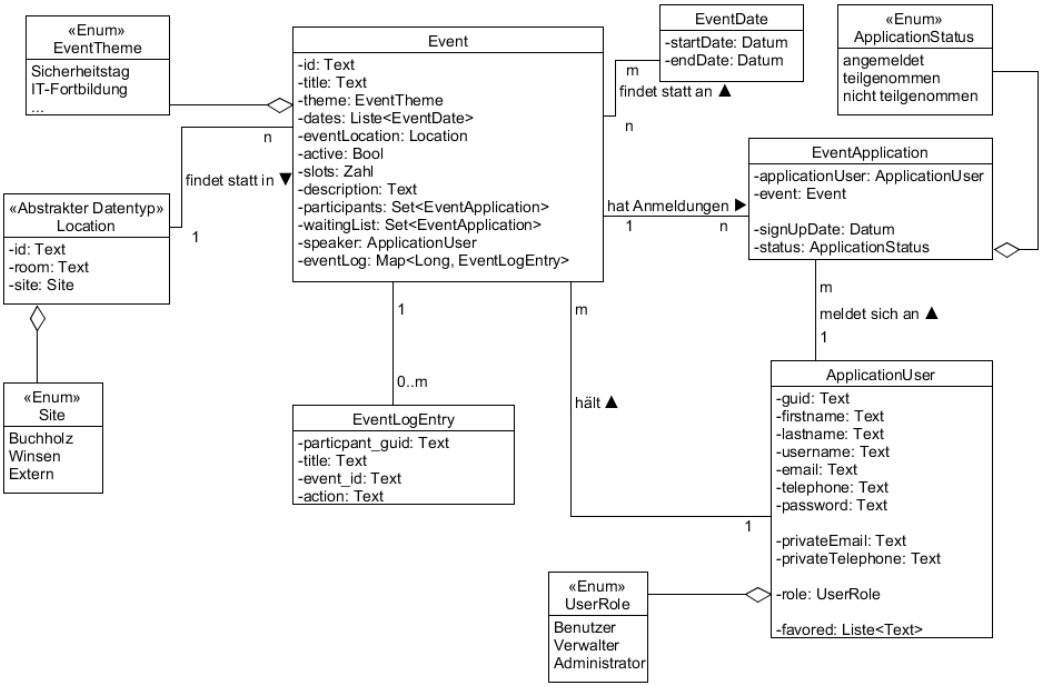

Abbildung 4.3: Fachliches Datenmodell. Enumeration EventTheme zur besseren Übersicht gekürzt

5 Komponenten und Schnittstellen

Dieses Kapitel befasst sich mit der Überführung des fachlichen Modells in einzelne Komponenten und Schnittstellen zwischen diesen.

5.1 Komponenten

5.1.1 Festlegen der einzelnen Komponenten

Zu einer der grundlegendsten Aufgaben gehört das Identifizieren und Festlegen der Komponentengrenzen. Domain-Driven-Design bietet sehr gute Ansätze, Komponenten gegeneinander abzugrenzen. Der erste Schritt laut Evans (siehe Eva04, S. 26) ist die Definition bzw. das Erkennen einer 'ubiquitous language'. Diese setzt ein tiefes Verständnis der Geschäftsdomäne voraus. Daniel Takai zeigt den starken Zusammenhang der Allgemeinsprache mit einem Softwareprojekt: "Die Allgemeinsprache schafft Effizienz und Konzeptionelle Integrität"(siehe Tak17, S. 92f.). Nicht nur die geschriebene Dokumentation hängt von dieser Sprache ab, auch viel wichtigere Teile der Software, wie z.B. der eigentliche Code oder veröffentlichte Schnittstellen mit ihren Beschreibungen.

Die Aufteilung in Komponenten erfolgt anhand der sprachlichen Grenzen der Komponenten bzw. Konzepte innerhalb der Komponenten. Das Ergebnis der Unterteilung ist eine Context Map mit verschiedenen Domänen und Kontexten. Die Grenzen dieser Domänen sind nur definiert durch die sprachlichen Unterschiede einzelner Entitäten in verschiedenen Kontexten (siehe Eva04, S. 335). Als Beispiel existieren in mehreren Kontexten Benutzer-Entitäten. In allen Bereichen dieselbe Abbildung eines Benutzers einzusetzen wäre ungeschickt, da häufig nicht alle Eigenschaften der Schnittmenge aller Benutzer-Repräsentationen genutzt werden. Außerdem müssten bei Anpassungen der Attribute des Benutzer in allen Komponenten diese Entität angepasst werden und damit einhergehend ggf. auch Geschäftslogik. Alle Repräsentationen der Entität Benutzer sind zugeschnitten auf die einzelnen Kontexte, in denen diese Entität gültig ist. Im Kontext der Anmeldung sind für einen Benutzer hauptsächlich Benutzername und Passwort interessant. Im Kontext einer Veranstaltung der ggf. für alle Teilnehmer einsehbar

ist, sollten diese Informationen nicht angezeigt werden. Stattdessen ist hier nur der Name und vielleicht das Datum der Anmeldung relevant.

Wird das Prinzip des Domain Driven Design auf dieses Projekt angewandt, entsteht folgende Context Map für das FBM mit allen Domänen und Kontexten innerhalb der Domänen:

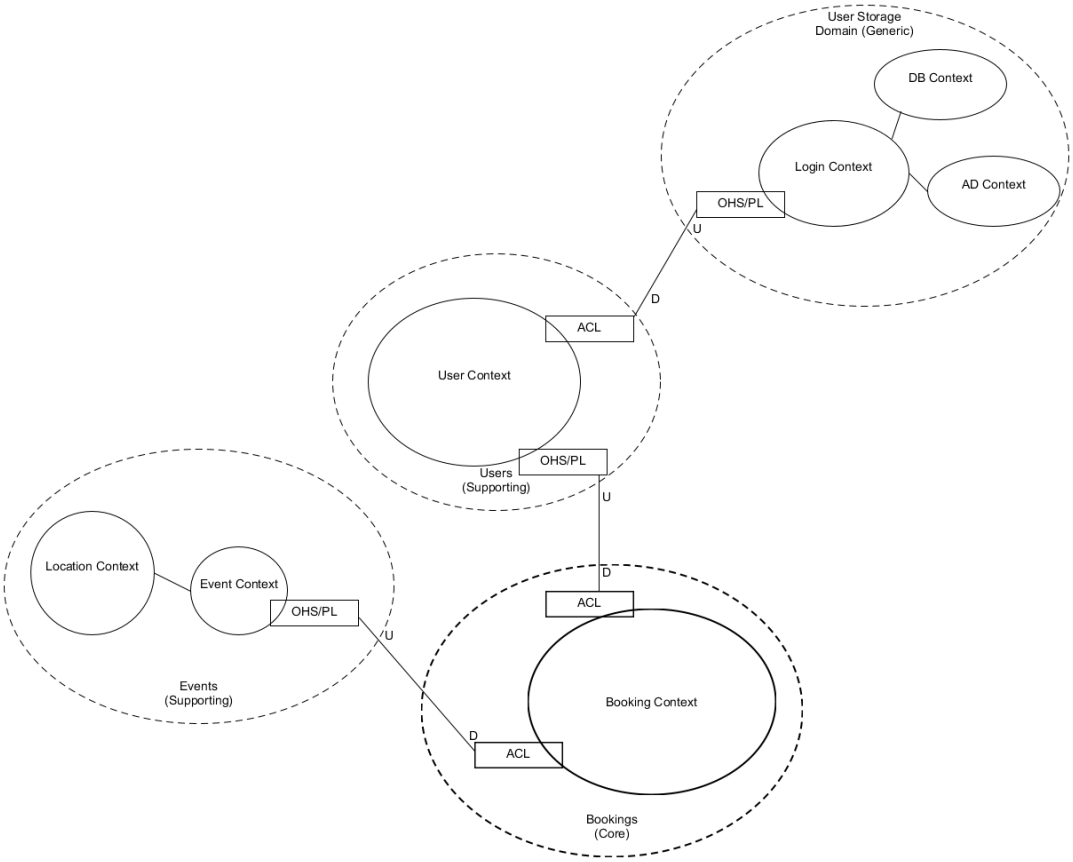

Abbildung 5.1: Context Map für das FBM. Beinhaltet zusätzlich Klassifizierungen der Adapter, Upstream / Downstream Beziehungen und Arten der verschiedenen Domänen.

Das Hauptaugenmerk des FBM liegt auf der Buchung von Veranstaltungen durch Benutzer. Deswegen ist die Domäne Bookings die Core-Domain (Eva04, S. 401f.), da hier das Kerngeschäft statfindet. Events und Users sind Supporting Sub-Domänen, da sie die Inhalte einer Buchung bereitstellen, die Entitäten Event und User liefern. Die Generic Subdomain (Eva04, S. 404) User Storage ist für die Kapselung der verschiedenen Datenquellen der Benutzer zuständig. In

ihr liegt der Login-Kontext, welcher in der geplanten Ausführung Datenbanken und Active-Directories abstrahieren kann.

Die Markierungen der Verbinder zwischen den Kontexten sind mit einem U oder D markiert. Dies deutet an, dass es sich um eine Upstream- oder Downstream-Verbindung handelt. Upstream-Dienste zeichnen sich dadurch aus, dass sie Downstream Dienste beeinflussen, weil Sie gewisse Datenstrukturen vorgeben bzw. den Ursprung der Daten darstellen. In 5.1 zu erkennen am Booking Context in der Bookings Domäne. Vom Booking Context aus gesehen sind alle anderen Dienste Upstream, da alles in der Bookings Domäne zusammenläuft.

Die Konnektoren der einzelnen Kontexte sind mit den Abkürzungen ACL, OHS und PL gekennzeichnet. Diese Abkürzungen stehen für Anti Corruption Layer, Open Host Service und Public Language (siehe Ver13, S.99f.).

ACL Ein **A**nti **C**orruption **L**ayer findet sich häufig an Downstream-Diensten, die eine Schnittstelle von Upstream Diensten konsumieren. Der Anticorruption Layer ist für die Übersetzung der empfangenen Daten zuständig. Dabei geht es darum die externe Repräsentation der Entität auf die interne zu mappen.

OHS Ein **O**pen **H**ost **S**ervice ist eine Methode, um an einem Dienst ähnlich eines Remote Procedure Calls Endpunkte zur Kommunikation mit dem Dienst anzubieten. Ein Beispielt wäre eine REST-Schnittstelle, welche von Client Services aufgerufen werden.

PL Eine **P**ublic **L**anguage wird meistens von einem Open Host Service angeboten und beschreibt öffentlich in einer festgelegten Sprache (z.B. XML oder JSON) die interne Repräsentation einer Domänen-Entität. Ein Beispiel wäre die JSON Repräsentation einer Entität an einer REST-Schnittstelle. Die Context-Map schafft den Einstieg in die technische Architektur des Systemes.

5.2 Microservice Architektur

Es entstehen mehrere Microservices, welche alle notwendigen Funktionen abbilden sollen. Einige werden zur Laufzeit erzeugt, somit ist die Anzahl dynamisch. Diese Microservices haben folgende Funktionen:

Main API Aggregiert zusammenhängende Daten, agiert als Api Gateway. Prüft bei Verwendnung von anderen APIs Zugangsberechtigungen.

AuthAPI Die AuthAPI stellt alle notwendigen Funktionen bereit, damit sich Benutzer am System authentifizieren können. Die AuthAPI bietet außerdem Schnittstellen an,

um die Autorisierung von Usern auf Ressourcen zu prüfen. Abstrahiert die Funktionalitäten der Services DatabaseLogin und LDAPLogin - somit ist spielt es keine Rolle, ob sich mit einem LDAP oder einem Datenbank-Login gearbeitet wird.

Users Verwaltet alle Benutzer der Software und speichert Daten wie z.B. favorisierte Veranstaltungen. Bietet Schnittstellen zum Abrufen, Erstellen und Speichern von Benutzern.

Events Verwaltet Veranstaltungen, bietet Schnittstellen für CRUD-Operationen.

Bookings Verwaltet Anmeldungen an Veranstaltungen. Ermöglicht An- / Abmelden an Veranstaltungen oder Bestätigen der Teilnahme.

Login Konsumiert Login-Anfragen und authentifiziert gegenüber der Datenbank oder dem LDAP. Dieser Microservice wird pro Benutzer-Datenquelle erzeugt. Sind also zwei Active-Directory und eine Datenbank Anbindung vorhanden, existieren drei dieser Microservices

Locations Beinhaltet alle statischen Daten, wie z.B. Räume oder Veranstaltungsorte. Bietet Schnittstellen für CRUD-Operationen an.

Logging Zentraler Service, an den Logs gesendet werden. Alle Dienste außer der Main API und der AuthAPI können hier her Daten zum Logging übermitteln.

In der Gesamtstruktur der Service-Landschaft werden die einzelnen Microservices für das Frontend hinter einem API Gateway deployed (siehe Abbildung 5.4). Das API Gateway hat die Aufgabe für das Web-Frontend Schnittstellen anzubieten, die ganzheitlich Microservice übergreifend Daten anbietet. Fragt ein Client eine Schnittstelle zu einer Veranstaltung sind davon drei Microservices betroffen:

1. Events: Alle grundlegenden Daten zum z.B. Namen, Datum oder der Beschreibung der Veranstaltung

2. Locations: Räume und Standorte werden in Veranstaltungen nur über ihre ID referenziert und können am Location Microservice aufgelöst werden

3. Users: Ähnlich wie Räume und Standorte werden die Benutzer als Teilnehmer oder Referenten in den Veranstaltungen nur als ID referenziert.

Das API Gateway hat nun die Möglichkeit intern diese drei einzelnen Microservices anzusprechen, die Daten zu bündeln und dem Web-Client in einer Anfrage bereitzustellen. Dadurch

kann Traffic in Form von Verbindungs-Overhead am Client gespart werden, da dieser nur eine Anfrage an das System stellt anstatt mindestens drei (siehe Chr13).
Das API Gateway dient dabei als Trichter und sogar Filter, kann also Daten zusammenführen und ggf. kürzen. Hierbei ist aber darauf zu achten, dass das API Gateway nicht zu sehr an die dahinter liegenden Services gekoppelt wird. Es geht lediglich darum, die Daten stellvertretend abzurufen, zusammenzuführen und weiterzuleiten. Wenn ein Client direkt ein Event am Event-Service anfragen würde, erhält er ein JSON mit Referenzen auf andere Objekte (siehe Abbildung 5.2)

```
1  {
2      "id": "5b22773a8eea9f0001365ee0",
3      "title": "Ein Titel",
4      "dates": [
5        {
6          "startDate": 1528956000000,
7          "endDate": 1528981200000
8        }
9      ],
10     "eventLocation": "5b1e6f7a8eea9f000196fd80",
11     "speaker": "341d84a66f2ed541bdb2b449c6c11fc6",
12     ...
13 }
```

Abbildung 5.2: Auszug eines Events im JSON Format

Diese IDs, z.B. für die 'eventLocation', kann ein anderer Microservice in ein weiteres JSON Objekt auflösen (siehe Abbildung 5.3). Ohne Kenntnis über den Zusammenhang oder Aufbau der einzelnen Objekte, kann das API Gateway diese Daten zusammenbringen und so dem Client Anfragen ersparen.

```
{
    "id": "5b1e6f7a8eea9f000196fd80",
    "room": "Casino",
    "site": {
      "name": "Buchholz"
    }
}
```

Abbildung 5.3: Auszug eines Veranstaltungsortes im JSON Format

Sobald das API Gateway beim Zusammenbringen der Daten ein Verständnis über die Objekte oder gar die Geschäftslogik implementiert, um die eigentliche Änderung durchzuführen, entsteht eine starke Kopplung zwischen dem API Gateway und den dahinter liegenden Diensten. Die im obigen Beispiel vom API Gateway verrichtete Arbeit ist generisch und kann für alle Daten durchgeführt werden, wenn es nur darum geht IDs in Objekte aufzulösen.
Bringt man die Dienste zusammen und betrachtet ihre Interaktionen, erhält man die Struktur in Abbildung 5.4.

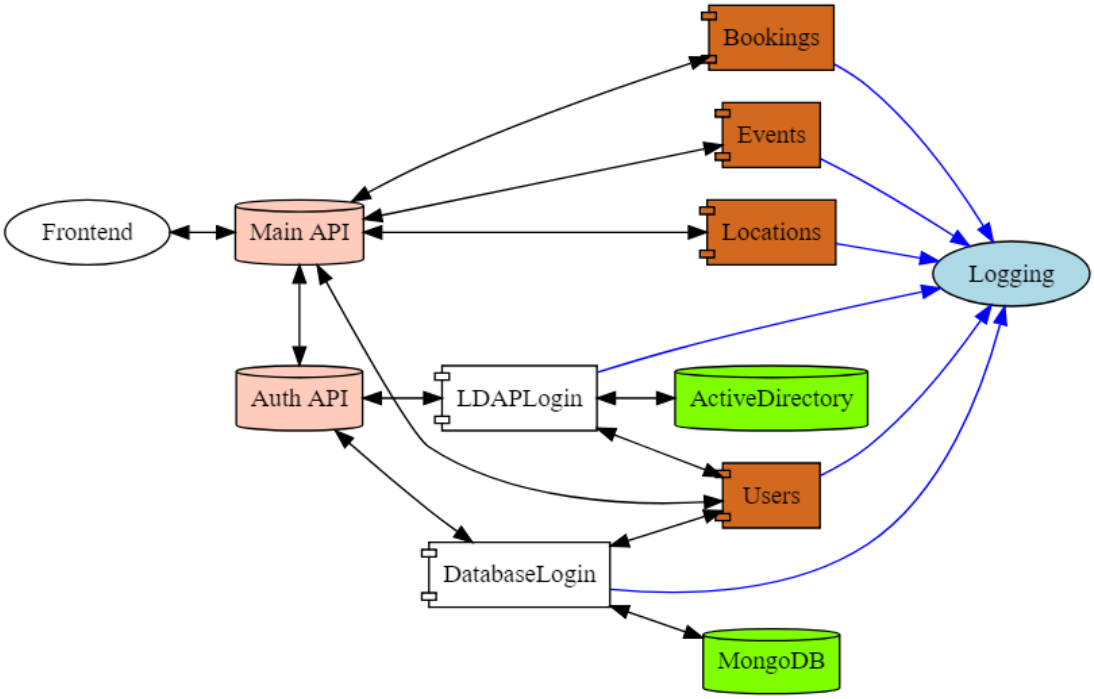

Abbildung 5.4: Entwurf der API-/Microservice-Struktur

Die Schnittstelle zwischen der Users-Komponente und LDAP Login/Database Login ermöglicht es den Login Komponenten neue Benutzer anzulegen/importieren.

5.3 REST-Endpunkte

In diesem Abschnitt werden die wichtigsten REST-Endpunkte für die API definiert.
Für alle REST-Anfragen sollen Fehlermeldungen die an das Frontend übergeben werden, durch eigene Fehlercodes ersetzt werden. Das bedeutet, dass Fehler die für den Anwender interessant sind über die REST-Schnittstelle als HTTP-Status Code 400 zurückgegeben werden - im Body der Antwort befindet sich dann eine umfangreiche Fehlerbeschreibung samt eigener Kodierung der Fehler. Zu allen Aufrufen zwischen Microservices muss davon ausgegangen werden, dass ein Aufruf fehlschlagen kann, wenn ein entfernter Microservice nicht erreichbar ist. In der folgenden Beschreibung werden diese Fehler nicht extra aufgeführt.

5.3.1 Main API

Die Main API fungiert als oberste Schicht im Kontakt mit dem FBM, bietet also die komplexesten bzw. umfangreichsten Endpunkte an.

Abrufen von Veranstaltungen

Das Abrufen von Veranstaltungen benötigt Informationen von fast allen Services. Beim Abrufen der Veranstaltungen werden Informationen von den Komponenten Bookings, Events, Locations und Users.
Hierbei ist wichtig, dass es eine Art von Pagination gibt, damit nicht unnötige Datensätze abgerufen werden und dadurch die Performance des Systems leidet. Die konkrete Ausprägung des Pagination ist dabei nicht relevant. Die zu Grunde liegende Schnittstelle bleibt davon unberührt, da die Parameter für das Pagination optional sind und als Query Parameter an den Aufruf angehangen werden können.
`..\api\v1\event`
Die Methode GET gibt eine Liste von Veranstaltungen zurück. Diese Liste kann leer sein.

Anmeldungen an Veranstaltungen

Dieser REST-Endpunkt ist ein zusammengesetzter Aufruf. Ein erfolgreicher Durchlauf berührt die APIs der Services Bookings, Events und Users.
`..\api\v1\event\{EVENTID}\booking\{USERID}`
Die Methode POST versucht eine Anmeldung für einen Benutzer (USERID) zu einer Veranstaltung (EVENTID) anzulegen. Diese Operation kann folgende Ergebnisse haben:

- Operation erfolgreich (HTTP-Status 200): Aktualisierte Veranstaltung als Rückgabewert
- ID nicht gefunden (HTTP-Status 404): Keinen Rückgabewert

DELETE Operationen löschen eine Anmeldung bzw. melden den Benutzer von dieser Veranstaltung ab. Ergebnisse dieser Operationen können sein:

- Operation erfolgreich (HTTP-Status 200, 202 o. 204): Keinen Rückgabewert
- Operation nicht erfolgreich (HTTP-Status 400): Eigene Fehlermeldung mit Fehlerbeschreibung als Rückgabewert

5.3.2 Auth API

Als zentraler Einstiegspunkt für alle Anfragen bietet die Main API auch Login-Endpunkte, diese werden aber an die Auth API weitergereicht. Dazu gehören Logins und Erneuerungen von Session Tokens.

`..\api\v1\login`
Der Login Endpunkt benötigt im Body Benutzername und Passwort und liefert als Rückgabewert einen gültigen Session Token zurück mit dem der Benutzer geschützte Bereiche aufrufen kann.

`..\api\v1\refreshToken`
Der Endpunkt Refresh Token erneuert bei Übergabe eines bestehenden Session Token die Gültigkeitsdauer und gibt einen neuen Token zurück.

5.3.3 Bookings, Events, Locations, Users

Diese Services bieten CRUD-Operationen für die entsprechenden Entitäten an. Diese werden von der Main API durchgereicht. Der User Microservice bietet zusätzlich die Möglichkeit, dass Benutzer Veranstaltungen favorisieren und diese somit in ihrem Benutzerprofil gespeichert werden. Da dies vermutlich eine häufiger vorkommende Aktion sein wird, soll dieser Zugriff extra modelliert werden, anstatt jedes Mal die ganze Benutzer Entität zu aktualisieren. Der REST-Endpunkt sieht wie folgt aus:
`..\api\v1\user\favoriteevent\{EVENTID}`
Ein POST Aufruf speichert die angegebene Veranstaltung (EVENTID) für diesen Benutzer als Favorit ab, ein DELETE entfernt diesen Eintrag für den Benutzer. Beide Aufrufe sind idempotent, es wird keine aktualisierte Entität zurückgegeben.

5.3.4 Logging

Beim Logging Dienst geht es hauptsächlich darum, entstandene Logs anderer Services zu aggregieren und gesammelt zur Verfügung zu stellen (siehe Ric17a). Dieser Dienst stellt anderen Microservices die Möglichkeit zur Verfügung ihre Logs zu persistieren. Darüberhinaus soll die Möglichkeit bestehen diese Logs einzusehen. Daraus ergibt sich der REST-Endpunkt:
`..\api\v1\logs\`
Über ein POST Aufruf wird ein neuer Eintrag erzeugt, ein GET ruft alle vorhandenen Logs ab.

An dieser Stelle ist es sicherlich sinnvoll in der Implementation die Möglichkeit zu bieten die Ergebnisse einer GET Anfrage zu filtern bzw. zu begrenzen.

6 Docker

Docker hat sich seit der Veröffentlichung im März 2013 (Mou16, S.9) zu einem Markennamen der Containerisierung entwickelt. Im Juni 2014 wurde Docker als Version 1.0 GA veröffentlicht und hatte im Dezember 2014 schon 100 Millionen Downloads, Anfang 2018 wurde das 50 Milliardenste Container Image aus dem öffentlichen Docker Repository heruntergeladen. (Doc18).

6.1 Warum Container?

Diese Frage beantwortet (Mou16, S. 3f.) prägnant im Vergleich von Containern mit virtuellen Maschinen.

- Container nutzen den vorhandenen Kernel des Betriebssystem auf dem sie laufen und teilen sich diesen. Dadurch sind Container wesentlich effizienter was das Starten und Stoppen angeht
- Da ein Container nur eine Anwednung oder einen Anwendungs-Stack kapselt, ist dieser in seiner Gesamtheit portierbar, überall identisch konfiguriert und verhält sich in allen Umgebungen identisch
- Container sind so kompakt, dass es mehr als natürlich ist eine gesamte Produktiv-Umgebung zu Testzwecken auf einem einzelnen PC laufen zu lassen

Container und virtuelle Maschinen verfolgen unterschiedliche Ziele. Gerade im Bereich der Microservices macht eine Container Lösung häufig Sinn, da der schlanke Aufbau von Containern zu der Idee von Microservices als minimale Dienste passt. Docker setzt nicht auf Hardware Virtualisierung, um ganze Server zu emulieren, sondern kapselt auf einem bestehenden Kernel nur die Applikation. Dadurch entstehen sehr schlanke und performante Dienste, da für den Start und die Installation von Docker Containern keine ganzen Betriebssysteme booten müssen. So kann jeder Microservice in einem einzelnen Docker Container ausgerollt werden und bietet nach außen eine Service API an (siehe 6.1).

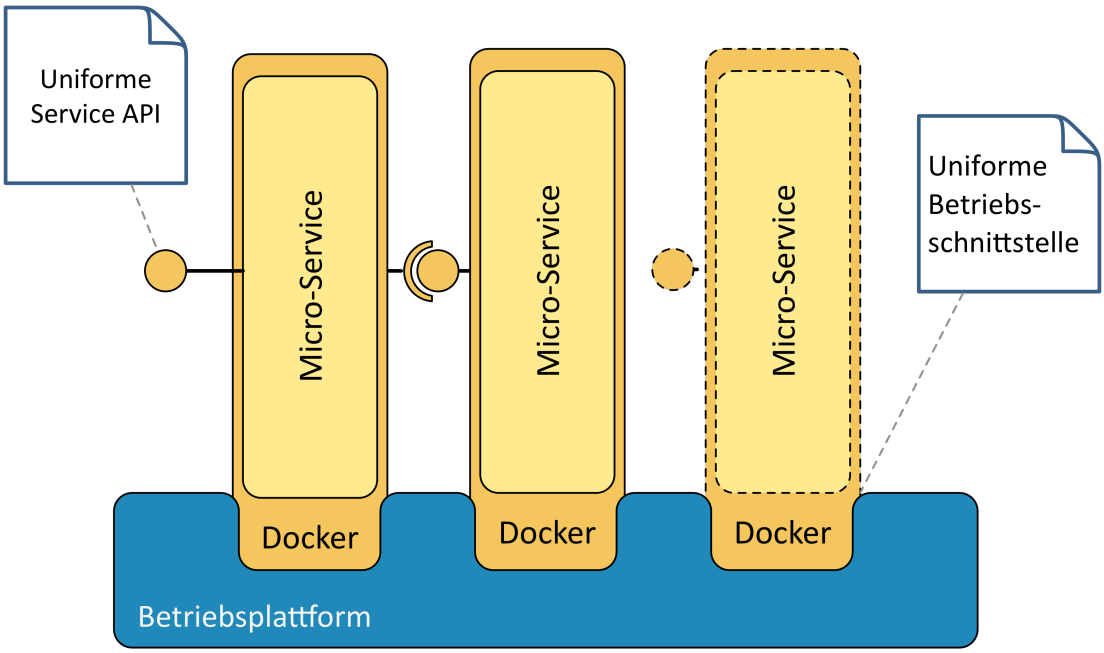

Abbildung 6.1: Microservices als Docker Container (Quelle: PT18, Abbildung 3)

6.2 Docker und FBM

Die einzelnen Microservices sollen als Docker Image bereitgestellt werden, damit diese über eine Orchestrierungslösung zur Verfügung gestellt werden können. Moderne Orchestrierungslösung bieten unzählige Funktionen, um Container auszurollen und zu verwalten. Ein bekannte Lösung ist Kubernetes (kub18), welche von Google entwickelt und als Open-Source Projekt zur Verfügung gestellt wurde. Google selbst nutzt Kubernetes, um eine Vielzahl von Containern zu managen (siehe Bai17, S.14f.). Für dieses Projekt interessante Features wären Load-Balancing, Self-healing, horizontal scaling und automated rollouts (kub18). Da Microservices aufgrund ihrer geringen Größe schnell und unabhängig entwickelt werden können (siehe Ric17b, Resulting context - Benefits), ist ein automatisierter Rollout erstrebenswert. Automatische Skalierung, Load-Balancing und Self-healing ist in einer Microservice-Archtitektur definitiv auch von Nöten, gerade in Fällen wie der hier geplanten Architektur (siehe Abbildung 5.4). Ein API Gateway ist unter Last das schwächste Glied, denn bei einer Überlastung/Ausfall und dem nicht erreichen der API ist das gesamte System von außen nicht für den Kunden zu erreichen.

Andere Dienste wie z.B. die Login Komponenten müssen zu Stoßzeiten deutlich mehr Anfragen bearbeiten, wo eine dynamische Skalierung sehr hilfreich ist.

7 Abnahmetests

Die Abnahmetests leiten sich direkt aus den vorhandenen Anforderungen ab. In Abschnitt 3.2 wurden die Kritischen Funktionen des Systems definiert. Zusammen mit den Zielbestimmungen in Abschnitt 2.4 ergeben sich einige gute Szenarien für Abnahmetests. Implizit ergeben sich allerdings durch die Wahl einer Microservice-Architektur eine ganze Reihe von Integrationstests, die vielleicht nicht direkt in die Sparte der Abnahmetests - im Sinne von „Abnahme durch den Kunden“- fallen, aber dennoch unerlässlich sind für ein verteiltes System.
Integrationstest setzen voraus, dass alle betroffenen Komponenten innerhalb von Modultests (Unit-Tests) erfolgreich getestet wurden (siehe Fra07, S.94f. 8.3 Integrationstest).

7.1 Warteliste

Die Warteliste ist im Verhältnis zu den anderen kritischen Funktionen am simpelsten zu testen. Die Funktionalität der Warteliste beschränkt sich nämlich nur auf eine Komponente, ist somit also kein Integrationstest sondern ein Unit-Test. Um ein akzeptablen Abnahmetest für diese Funktion zu entwerfen wir das Blackbox-Verfahren angewandt (siehe Fra07, S.31ff.).
MK140 in Abschnitt 2.4 spezifiziert: ”Sobald die Anzahl der Teilnehmer, die der zur Verfügung stehenden Plätze übersteigt werden die Anmeldung in einer Warteliste gehalten. Wenn wieder Plätze frei werden, sollen die Teilnehmer von der Warteliste automatisch in der richtigen Reihenfolge den Teilnehmern hinzugefügt werden.”
Das Überschreiten der verfügbaren Plätze durch die Anmeldung der Teilnehmer kann in zwei Fällen vorkommen:

1. Es meldet sich einer oder mehrere Teilnehmer an, wenn alle Plätze belegt sind.
2. Die Anzahl der verfügbaren Plätze wird verringert, wenn alle Plätze belegt sind.

Entsprechend gibt es für den Fall das Plätze wieder frei werden auch zwei Auslöser:

1. Es meldet sich einer oder mehrere Teilnehmer ab, wenn alle Plätze belegt sind.
2. Die Anzahl der verfügbaren Plätze wird erhöht, wenn alle Plätze belegt sind.

Bedingung	Äquivalenzklasse	Nr.
Teilnehmer >Plätze	Anmeldung bei vollen Plätzen	Z1
	Verringerung der verfügbaren Plätze	Z2
Teilnehmer <Plätze	Abmeldung bei vollen Plätzen	Z3
	Erhöhung der verfügbaren Plätze	Z4

Tabelle 7.1: Äquivalenzklassen Wartelisten-Tests

Daraus ergeben sich die in 7.1 aufgelisteten Äquivalenzklassen.

Testszenarien, welche diese Äquivalenzklassen abdecken wollen müssen die in 7.2 aufgelisteten Ergebnisse produzieren.

Nr.	Äquivalenzklasse	Erwartetes Ergebnis
1	Z1	Die zusätzliche(n) Anmeldung(en) wird/werden auf die Warteliste gebucht
2	Z2	Die zusätzliche(n) Anmeldung(en) wird/werden auf die Warteliste gebucht
3	Z3	Die älteste(n) Anmeldung(en) auf der Warteliste wird/werden auf die Teilnehmerliste gebucht
4	Z4	Die älteste(n) Anmeldung(en) auf der Warteliste wird/werden auf die Teilnehmerliste gebucht

Tabelle 7.2: Auflistung der Testfälle für Wartelisten-Tests

Da es sich bei den Bedingungen der Äquivalenzklassen um binäre Entscheidungen handelt und die beiden Bedingungen nicht gleichzeitig eintreten können, macht es keinen Sinn eine Grenzwertanalyse (siehe Fra07, S.36 4.1.2) oder eine Ursache-Wirkung-Analyse (siehe Fra07, S.38 4.1.3) durchzuführen.

7.2 Import der AD Benutzer

Der Import der AD Benutzer ist komplizierter als die Wartelisten-Funktion, da dieser Prozess sich über mehrere Dienste abspielt. Der Benutzer greift auf die Main API zu und meldet sich mit seinen Benutzerdaten aus dem Active-Directory an. Dieser Aufruf wird an die Auth API weitergeleitet und von der LDAP-Login Komponente verarbeitet. Das meiste dieser Komplexität spielt allerdings für den eigentlichen Import der AD-Benutzer keine große Rolle. Der Import der AD-Benutzer wird nur angestoßen, wenn die Zugangsdaten korrekt sind und es sich um

einen AD-Login handelt. Das eigentliche Importieren wird dann isoliert von der LDAP-Login Komponente durchgeführt.
Fachlich gesehen kann der Import korrekt nach den Vorgaben ablaufen oder der Import schlägt fehl. In Abschnitt 2.4 heißt es unter MK20 ”Mitarbeiter können sich mit ihrem Domänen-Login anmelden", unter MK25 "Beim Domänen-Login sollen vorgegebene Daten aus dem AD übernommen werden” und in Kombination mit dem fachlichen Modells des Benutzers aus Abschnitt 4.1 ergibt sich die Anforderung, dass für den Import aus dem Active-Directory folgende Daten erfolgreich abgerufen werden müssen:

- GUID zur eindeutigen Referenzierung der Benutzer
- Vorname und Nachname
- Email und Telefonnummer

Ein importierter Benutzer der nicht diese Felder aufweist gilt fachlich als falsch importiert. Zusätzlich kann der Benutzer bei der Anmeldung mit seinem Konto ein falsches Passwort oder einen falschen Benutzernamen eingeben, was zu einem nicht erfolgreichen Import führt. Technische Fehler sind in diesem Anwendungsfall deutlich mehr vorhanden. Vom Login bis zum korrekten Import der Benutzerdaten sind insgesamt sechs Services involviert. Sollte einer während des Vorgangs ein Dienst nicht erreichbar sein, kann der gesamte Prozess daran scheitern. Für jeden solcher technischen Fehler muss ein Testfall entwickelt werden (vgl. Fra07, S.98). Exemplarisch für den Ausfall der LDAPLogin Komponente:

Testfall:

Der Microservice LDAP Login ist nicht verfügbar

Beschreibung:

Der Benutzer übergibt im Frontend seine Login Daten für das Active-Directory. Diese werden über die Main. und Auth-API an die LDAP Login Komponente übergeben. Sollte dieser Service nicht erreichbar sein, muss der Login Versuch bzw. der Import der Benutzerdaten scheitern und der Benutzer angemessen darüber in Kenntnis gesetzt werden.

Erwartete Ergebnisse:

1. Der Dienst ist nicht erreichbar. Als HTTP-Response wird eine Meldung im Klartext an das Frontend zurückgegeben:”Einloggen ist zur Zeit nicht verfügbar, bitte versuchen Sie es später noch ein Mal!”

2. Alle bis hier hin vorgenommenen Aktionen müssen rückgängig gemacht werden, da die Transaktion nicht vollständig abgeschlossen wurde.

Alle anderen Microservices können ähnlich nicht erreichbar sein und müssen auf diesen Ausfall hin getestet werden.

7.3 Buchen von Veranstaltungen

Das Buchen von Veranstaltungen ist eine Kombination aus den beiden vorangegangenen Testfällen. Zum Einen betrifft das Buchen von Veranstaltungen mehrere Microservices, wodurch technische Testfälle entstehen wenn Services nicht zu erreichen sind. Zum Anderen existieren Anforderungen an die Möglichkeit eine Veranstaltung zu buchen. Diese umfassen:

- Veranstaltungen dürfen noch nicht begonnen haben, damit sich ein Benutzer anmelden kann (Kein Datum der Veranstaltung darf in der Vergangenheit liegen).
- Veranstaltungen müssen aktiv sein, damit sich ein Benutzer anmelden kann.
- Jeder Benutzer kann sich nur einmal für eine Veranstaltung anmelden).
- Wenn ein Benutzer als Referent für eine Veranstaltung eingetragen ist, kann er sich nicht noch einmal für die Veranstaltung anmelden.

Ähnlich wie bei den Wartelisten Tests lassen sich hieraus die Äquivalenzklassen (siehe 7.3) und die Testfälle (siehe 7.4) ableiten.

Bedingung	**zulässige Äquivalenzklasse**	**Nr.**	**unzulässige Äquivalenzklasse**	**Nr.**
Datum	Alle Daten sind in der Zukunft	Z1	Mindestens ein Datum in der Vergangenheit	U1
Aktiv?	ja	Z2	nein	U2
Schon angemeldet?	nein	Z3	ja	U3
Referent?	nein	Z4	ja	U4

Tabelle 7.3: Äquivalenzklassen Buchen von Veranstaltungen

Nr.	Äquivalenzklassen	Erwartetes Ergebnis
1	Z1,Z2,Z3,Z4	Der Benutzer kann sich für die Veranstaltung anmelden
2	U1,Z2,Z3,Z4	Hinweis: Die Veranstaltung hat bereits begonnen, Anmeldungen sind nicht mehr möglich
3	Z1,U2,Z3,Z4	Hinweis: Diese Veranstaltung ist bisher nur geplant, Anmeldungen sind noch nicht möglich
4	Z1,Z2,U3,Z4	Hinweis: Sie sind bereits für diese Veranstaltung angemeldet
5	Z1,Z2,Z3,U4	Hinweis: Als Referent einer Veranstaltung brauchen Sie sich nicht anmelden

Tabelle 7.4: Auflistung der Testfälle für das Buchen von Veranstaltungen

Die drei Kernprozesse, welche in diesem Kapitel beleuchtet wurden, decken die meist genutzten Features des FBM ab. Dennoch ist es wichtig auch alle anderen Teile der Software zu testen.

8 Fazit und Ausblick

Abschließend lässt sich sagen, dass sowohl der Ansatz des Domain Driven Design als auch die Wahl der Microservice-Architektur gut zu der Aufgabe passen. Die Anwendung einer 'ubiquitous language' zur Eingrenzung der Domänen und Kontexte ermöglicht einen schnellen Einstieg in eine Context Map, welche schnell einen Überblick über das System gibt.
Der Übergang von einer Context Map zu einer Microservice-Architektur schafft auch einen schnellen Einstieg in die Entwicklung der einzelnen Services. Unter diesem Aspekt ist es möglich gerade zu Beginn der Entwicklung schnell und agil Microservices bereitzustellen und zu testen.

Im Rahmen der Planung kam die Idee auf für Benachrichtigungen zu Veranstaltungen iCalendar Dateien zu versenden, damit Teilnehmer diese direkt im Outlook angezeigt bekommen. Als diese Idee weiter gedacht wurde, wären dann auch noch Aktualisierungen für diese Veranstaltungen mit iCalendar Dateien verschickt worden. An diesem Punkt wäre es wahrscheinlich auch genau so gut möglich, das gesamte FBM in Outlook abzubilden. Outlook bietet vielleicht nicht alle Funktionen maßgeschneidert für das FBM an, aber ein Großteil der Funktionen sollten damit abzubilden sein.

Nichtsdestotrotz soll das hier geplante FBM entwickelt werden und den Krankenhäusern Buchholz und Winsen zur Verfügung gestellt werden. Dabei soll das FBM noch um einige Funktionen erweitert werden, die hier in dieser Arbeit nicht behandelt wurden. Unter anderem soll ein SAML SSO für die Domänen-Authentifizierung genutzt werden und es wird ein automatisiertes Benachrichtungssystem entwickelt, welches Teilnehmern Informationen zu anstehenden, geänderten oder teilgenommenen Veranstaltungen zukommen lassen soll. Gerade das spätere Erweitern und Warten der Software soll durch die Microservice-Architektur deutlich einfacher und reibungsloser ablaufen.

Literaturverzeichnis

[Bai17] Baier, Jonathan: *Getting started with Kubernetes: harness the power of Kubernetes to manage Docker deployments with ease.* Packt Publishing, 2017. – ISBN 978–1–78728–336–7

[Bra13] Brandstäter, Jonathan: *Agile IT-Projekte erfolgreich gestalten.* Springer Fachmedien Wiesbaden, 2013. – ISBN 978–3–658–04430–5

[Chr13] Christensen, Ben: *Optimizing the Netflix API – Netflix TechBlog – Medium.* https://medium.com/netflix-techblog/optimizing-the-netflix-api-5c9ac715cf19. Version: Jan 2013. – [Online; Abgerufen 15.09.2018]

[Doc18] Docker: *Company.* https://www.docker.com/company. Version: Oct 2018. – [Online; Abgerufen 14.10.2018]

[Ebe13] Ebert, Christof: *Risikomanagement kompakt: Risiken und Unsicherheiten bewerten und beherrschen.* Springer Vieweg, 2013. – ISBN 978–3–642–41048–2

[Eva04] Evans, Eric: *Domain-Driven Design: Tackling Complexity in the Heart of Software.* Addison-Wesley, 2004. – ISBN 978–0–321–12521–7

[Fra07] Franz, Klaus: *Handbuch zum Testen von Web-Applikationen - Testverfahren, Werkzeuge, Praxistipps.* Berlin Heidelberg New York : Springer-Verlag, 2007. – ISBN 978–3–540–68185–4

[kub18] *Production-Grade Container Orchestration.* https://kubernetes.io/. Version: 2018. – [Online; Abgerufen 30.10.2018]

[LF14] Lewis, James ; Fowler, Martin: *Microservices.* https://martinfowler.com/articles/microservices.html. Version: Mar 2014. – [Online; Abgerufen 28.10.2018]

[Mou16] Mouat, Adrian: *Docker - Software entwickeln und deployen mit Containern.* Heidelberg : dpunkt.verlag, 2016. – ISBN 978–3–960–88037–0

[PT18] Preissler, Jerry ; Tigges, Oliver: *Docker - Perfekte Verpackung von Micro-services.* https://www.innoq.com/de/articles/2015/11/docker-perfekte-verpackung-fuer-micro-services/. Version: Oct 2018. – [Online; Abgerufen 20.10.2018]

[Ric17a] Richardson, Chris: *Microservices Pattern: Log aggregation.* https://microservices.io/patterns/observability/application-logging.html. Version: 2017. – [Online; Abgerufen 30.09.2018]

[Ric17b] Richardson, Chris: *Pattern: Microservice Architecture.* https://microservices.io/patterns/microservices.html. Version: 2017. – [Online; Abgerufen 30.09.2018]

[SBS02] Spitta, Thorsten ; Borchers, Jens ; Sneed, Harry M.: *Software Management 2002: Fortschritt durch Beständigkeit: Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschaftsinformatik (WI-MAW), 6. bis 8. November 2002, Hamburg.* Gesellschaft für Informatik, 2002

[Tak17] Takai, Daniel: *Architektur für Websysteme - Serviceorientierte Architektur, Microservices, Domänengetriebener Entwurf.* M : Carl Hanser Verlag GmbH Co KG, 2017. – ISBN 978–3–446–45248–0

[TS17] Tsolkas, Alexander ; Schmidt, Klaus: *Rollen und Berechtigungskonzepte - Identity- und Access-Management im Unternehmen.* Berlin Heidelberg New York : Springer-Verlag, 2017. – ISBN 978–3–658–17987–8

[Ver13] Vernon, Vaughn: *Implementing Domain-Driven Design.* 1st. Addison-Wesley Professional, 2013. – ISBN 978–0–321–83457–7

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 1. November 2018 Rene Kretschmer