



Abschlussprüfung Januar 2017

Fachinformatikerin für Anwendungsentwicklung  
Dokumentation zur betrieblichen Projektarbeit

## **AO-Anmeldungen**

**Entwicklung einer Webanwendung mit Java EE 7**

Abgabetermin: Lingen, den 01. Dezember 2016

**Prüfungsbewerberin:**

Maria Elena Hollen  
Theodor-Heuss-Str. 96  
49377 Vechta



**Ausbildungsbetrieb:**

ALTE OLDENBURGER Krankenversicherung AG  
Theodor-Heuss-Str. 96  
49377 Vechta

## Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>Tabellenverzeichnis</b>	<b>III</b>
<b>Listings</b>	<b>III</b>
<b>Abkürzungsverzeichnis</b>	<b>IV</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Projektbeschreibung . . . . .	1
1.2 Projektziel . . . . .	1
1.3 Projektumfeld . . . . .	1
1.4 Projektbegründung . . . . .	2
1.5 Projektschnittstellen . . . . .	2
<b>2 Projektplanung</b>	<b>2</b>
2.1 Projektphasen . . . . .	2
2.2 Ressourcenplanung . . . . .	3
2.3 Entwicklungsprozess . . . . .	3
<b>3 Analysephase</b>	<b>4</b>
3.1 Ist-Analyse . . . . .	4
3.2 Wirtschaftlichkeitsanalyse . . . . .	4
3.2.1 „Make or Buy“-Entscheidung . . . . .	4
3.2.2 Projektkosten . . . . .	5
3.2.3 Amortisationsdauer . . . . .	5
3.3 Anwendungsfälle . . . . .	7
3.4 Lastenheft . . . . .	7
<b>4 Entwurfsphase</b>	<b>7</b>
4.1 Zielplattform . . . . .	8
4.2 Architekturdesign . . . . .	8
4.3 Entwurf der Benutzeroberfläche . . . . .	9
4.4 Datenmodell . . . . .	9
4.5 Geschäftslogik . . . . .	10
4.6 Deployment . . . . .	10
4.7 Pflichtenheft . . . . .	11
<b>5 Implementierungsphase</b>	<b>11</b>
5.1 Iterationsplanung . . . . .	11
5.2 Aufsetzen der benötigten Elemente . . . . .	11

5.3	Implementierung der gemeinsam genutzten Entitys . . . . .	12
5.4	Implementierung der Persistence . . . . .	12
5.5	Implementierung der Domain inkl. Geschäftslogik . . . . .	13
5.6	Test . . . . .	13
5.7	Implementierung der Views inkl. Benutzeroberfläche . . . . .	13
<b>6</b>	<b>Abnahme- und Einführungsphase</b>	<b>14</b>
6.1	Abnahme durch den Fachbereich . . . . .	14
6.2	Deployment und Einführung . . . . .	14
<b>7</b>	<b>Dokumentation</b>	<b>14</b>
<b>8</b>	<b>Fazit</b>	<b>15</b>
8.1	Soll-/Ist-Vergleich . . . . .	15
8.2	Ausblick . . . . .	15
	<b>Literaturverzeichnis</b>	<b>16</b>
	<b>Eidesstattliche Erklärung</b>	<b>17</b>
<b>A</b>	<b>Anhang</b>	<b>i</b>
A.1	Screenshot der bisherigen Anwendung im Intranet . . . . .	i
A.2	Detaillierte Zeitplanung . . . . .	ii
A.3	Ressourcenplanung . . . . .	iii
A.4	Break-Even-Analyse . . . . .	iv
A.5	Anwendungsfalldiagramm . . . . .	v
A.6	Lastenheft (Auszug) . . . . .	vi
A.7	Komponentendiagramm . . . . .	vii
A.8	Oberflächenentwürfe . . . . .	viii
A.9	Entity-Relationship-Model . . . . .	ix
A.10	Verteilungsdiagramm . . . . .	x
A.11	Gradle Build-Datei . . . . .	xi
A.12	JBoss-Konfigurationsdatei . . . . .	xii
A.13	Pflichtenheft (Auszug) . . . . .	xiii
A.14	Iterationsplan . . . . .	xiv
A.15	CheckStyle im Jenkins . . . . .	xv
A.16	Entity mit JPA-Anotationen . . . . .	xvi
A.17	Businesslogik im MessageService . . . . .	xvii
A.18	Test des MessageService . . . . .	xix
A.19	View mit JSF-Technologie . . . . .	xxi
A.20	Screenshots AO-Anmeldungen . . . . .	xxii
A.21	Benutzerdokumentation . . . . .	xxiii
A.22	Entwicklerdokumentation . . . . .	xxiv

A.23 Klassendiagramm . . . . .	xxv
--------------------------------	-----

## Abbildungsverzeichnis

1 Screenshot: Masseurauswahl in der bisherigen Anwendung . . . . .	i
2 Screenshot: Massageauswahl in der bisherigen Anwendung . . . . .	i
3 Break-Even-Analyse . . . . .	iv
4 Anwendungsfalldiagramm . . . . .	v
5 Komponentendiagramm . . . . .	vii
6 MockUp: Anlegen einer Veranstaltung . . . . .	viii
7 MockUp: Eintragen zum Telefondienst . . . . .	viii
8 Entity-Relationship-Model . . . . .	ix
9 Verteilungsdiagramm . . . . .	x
10 CheckStyle-Warnungen im Jenkins . . . . .	xv
11 Screenshot: Startseite AO-Anmeldungen . . . . .	xxii
12 Screenshot: Massage anlegen . . . . .	xxii
13 Entwicklerdokumentation . . . . .	xxiv
14 Klassendiagramm . . . . .	xxv

## Tabellenverzeichnis

1 Projektplanung . . . . .	3
2 Projektkosten . . . . .	5
3 Zeiteinsparung pro Monat . . . . .	6
4 Projektstrukturplan . . . . .	ii
5 Erläuterung der Eingabefelder . . . . .	xxiii

## Listings

1 Gradle Build-Datei . . . . .	xi
2 JBoss-Konfigurationsdatei . . . . .	xii
3 Entity mit JPA-Annotationen . . . . .	xvi
4 Businesslogik im MessageService . . . . .	xvii
5 Test des MessageService . . . . .	xix
6 View mit JSF-Technologie . . . . .	xxi

## Abkürzungsverzeichnis

<b>AO</b>	ALTE OLDENBURGER Krankenversicherung AG
<b>CMS</b>	Content-Management-System
<b>CSS</b>	Cascading Style Sheets
<b>EAP</b>	Enterprise Application Platform
<b>EDV</b>	Elektronische Datenverarbeitung
<b>ERM</b>	Entity-Relationship-Model
<b>HTML</b>	Hypertext Markup Language
<b>GUI</b>	Graphical User Interface
<b>Java EE 7</b>	Java Enterprise Edition 7
<b>JPA</b>	Java Persistence API
<b>JSF</b>	Java Server Faces
<b>MoSCoW</b>	Must, Should, Could, Won't have
<b>MVC</b>	Model View Controller
<b>ORM</b>	Objekt-Relationales-Mapping
<b>PHP</b>	Hypertext Preprocessor
<b>POJO</b>	Plain Old Java Objekt
<b>SCM</b>	Source Code Management
<b>TDD</b>	Test Driven Development
<b>VGH</b>	Versicherungsgruppe Hannover

## 1 Einleitung

In dieser Projektdokumentation wird der Ablauf des Abschlussprojektes, das die Autorin im Rahmen ihrer Abschlussprüfung zur Fachinformatikerin für Anwendungsentwicklung durchführt, erläutert. Das Projekt wird in der ALTE OLDENBURGER Krankenversicherung AG (AO) durchgeführt, welches der Ausbildungsbetrieb der Autorin ist. Die AO ist eine private Krankenversicherung mit Sitz in Vechta, die zur Versicherungsgruppe Hannover (VGH) gehört. Es sind bei der AO derzeit 236 Mitarbeiter beschäftigt.<sup>1</sup> Der Leistungsumfang umfasst private Krankheitskostenvollversicherungen, Pflegeversicherungen und Zusatzversicherungen zur gesetzlichen Krankenversicherung. Ziel dieser Dokumentation ist es, die durchzuführenden Schritte des Projektes von der Planung bis zum Deployment zu erläutern und dies mit geeigneten Diagrammen und Dokumenten zu unterstützen.

### 1.1 Projektbeschreibung

Für die AO soll im Rahmen der Umstellung des unternehmensinternen Intranets zu einem konzernweiten Mitarbeiterportal eine Webanwendung mit Java und dem Framework Java Enterprise Edition 7 (Java EE 7) entwickelt werden. Diese Webanwendung soll bestimmte interne Anforderungen umsetzen, die im Mitarbeiterportal nicht bereitgestellt werden, wie z.B. Veranstaltungsanmeldungen.

### 1.2 Projektziel

Das Ziel dieses Projektes ist die erfolgreiche Umsetzung einer neuen Webanwendung, die alle internen Anmeldungen von AO-Mitarbeitern verwaltet. Es sollen Veranstaltungen, Telefondienste und Massagen über diese Anwendung abzuwickeln sein. Nach der Einführung sollen alle Mitarbeiter die Anwendung bedienen können.

### 1.3 Projektumfeld

Es wird zur Einführung des Mitarbeiterportals das Projekt *Mitarbeiterportal* initiiert. Das Management von internen Projekten wird bei der AO von der Abteilung *Grundsatz* übernommen, die auch dieses Projekt leitet. Dieses Projekt ist ein Teilprojekt des Projektes *Mitarbeiterportal*, wodurch die Abteilung *Grundsatz* der Auftraggeber ist.

Die Betreuung der informationstechnischen Anforderungen in diesem Projekt wird von der Abteilung EDV (Elektronische Datenverarbeitung) übernommen.

Um die Anforderungen an dieses Projekt zu ermitteln, werden Projektmitglieder des Projektes *Mitarbeiterportal*, die in den Fachbereichen der AO tätig sind, konsultiert. Diese sind zugleich die künftigen Anwender, die die Webanwendung als Administrator und Benutzer nutzen werden.

---

<sup>1</sup>Vgl. ALTE OLDENBURGER KRANKENVERSICHERUNG AG [2016, S. 6].

## 1.4 Projektbegründung

Das bisherige Intranet der AO wird zu einem konzernweiten Mitarbeiterportal umgestellt. Die bisherige Anmeldung zu Veranstaltungen und Massage wurde über das Intranet gelöst. Screenshots dieser bisherigen Anwendung im Intranet sind im Anhang A.1, Seite i dargestellt. Dies ist eine Eigenentwicklung mit der Programmiersprache Hypertext Preprocessor (PHP), die auf das Content-Management-System (CMS) TYPO3 aufsetzt. Mit dem neuen Mitarbeiterportal wird das Intranet und somit auch TYPO3 abgeschaltet und nicht mehr verwendet. Zudem ist die TYPO3-Version veraltet und müsste bei weiterer Verwendung gegebenenfalls aktualisiert werden. Außerdem entspricht PHP nicht den Architekturrichtlinien der AO, welche die Verwendung von Java vorschreiben. Weiter ist die Wartung eines CMS wie TYPO3 für die geforderten Anwendungsfälle zu aufwendig, da dieses in erster Linie für die gesamte Organisation von z.B. Webseiten, Text- oder Multimedia-Dateien verwendet wird.

Die Telefondienstlisten werden in jeder Abteilung analog auf Papier gepflegt. Zu Beginn des Jahres wird eine Liste gefüllt, die bei Änderungen angepasst werden muss. Aufgrund der Intransparenz und des enormen Aufwands bei Änderungen dieser Liste, soll auch diese Funktionalität in der Webanwendung zu finden sein.

Des Weiteren bietet das Mitarbeiterportal keine Funktionalität oder Schnittstelle, um die Anforderungen der Anmeldung zu Veranstaltungen, Telefondiensten oder Massagen umzusetzen. Allerdings benötigen die Mitarbeiter der AO eine Möglichkeit, diese Veranstaltungen zu organisieren und sich für diese anzumelden.

## 1.5 Projektschnittstellen

Dieses Projekt ist, wie bereits in Kapitel 1.3 erwähnt, ein Teilprojekt des Projektes *Mitarbeiterportal*. Dieses Projekt ist allerdings weitgehend vom Projekt *Mitarbeiterportal* unabhängig, da eine Schnittstelle ausschließlich durch die Möglichkeit, im Mitarbeiterportal einen Link zur Webanwendung aufzurufen, besteht. Zeitlich ist das Projekt vom Projekt *Mitarbeiterportal* ebenfalls nicht abhängig: Die Möglichkeit der Anmeldung zu Veranstaltungen, Telefondienst und Massage steht im bisherigen Intranet zunächst weiterhin zur Verfügung.

# 2 Projektplanung

In diesem Kapitel werden die zur Verfügung stehende Zeit sowie die Ressourcen des Projektes geplant.

## 2.1 Projektphasen

Bevor mit der Umsetzung des Projektes begonnen wird, muss ein Projektplan erstellt werden (siehe Tabelle 1). Es werden die Hauptphasen des Projektes und die Dauer in Stunden aufgelistet. Eine genauere Zeitplanung inkl. Unteraufgaben ist im Anhang A.2, Seite ii dargestellt.

Tabelle 1: Projektplanung

Projektphase	Geplante Zeit (in h)
Analyse	10h
Entwurf	11h
Implementierung inkl. Tests	42h
Abnahme & Einführung	2h
Erstellung der Dokumentation	5h
<b>Gesamt</b>	<b>70h</b>

## 2.2 Ressourcenplanung

Im Folgenden werden alle für die Umsetzung des Projektes benötigten Ressourcen, d.h. Hardware- und Softwarekomponenten sowie benötigtes Personal betrachtet. Es wird nach Möglichkeit darauf geachtet, dass bei der Umsetzung des Projektes nur Software verwendet wird, die keine Anschaffung von Lizenzen nach sich zieht, wodurch die Projektkosten gering gehalten werden können. Die Ressourcenplanung ist im Anhang A.3, Seite iii dargestellt.

## 2.3 Entwicklungsprozess

Bevor mit der Implementierung begonnen wird, muss ein Entwicklungsprozess ausgewählt werden. Mit Hilfe des Entwicklungsprozesses wird das Vorgehen, nach dem die Umsetzung des Projektes erfolgen soll, definiert.

Es wird eine agile Vorgehensweise verwendet, in der es um eine schnelle Reaktion auf sich ändernde Anforderungen geht.<sup>2</sup> Nach kurzen Iterationen, in denen entwickelt wird, wird Rücksprache mit dem Fachbereich gehalten und Feedback eingeholt. Das schnelle Feedback ermöglicht eine unmittelbare Aufnahme von Anforderungsänderungen, die in den nächsten Iterationen umgesetzt werden können.

Außerdem wird durch die ständige Rücksprache ein Zeitersparnis bei der Abnahme des Projektes erzielt, da die Ansprechpartner des Fachbereichs während der Entwicklung in das Projekt integriert sind.

Des Weiteren soll das agile Vorgehen durch Tests erweitert werden. Es wird das Konzept des Test Driven Development (**TDD**) angewendet, bei dem vor der Implementierung des Programmcodes Tests geschrieben werden. Diese Tests stellen zum Einen sicher, dass die erwarteten Funktionalitäten auch von der Anwendung erfüllt werden. Zum Anderen dient das **TDD** dem Entwickler bei dem Aufbau des Systems und der Architektur.<sup>3</sup> Es wird ein Testfall geschrieben, der die korrekte Funktionsweise der nächsten zu entwickelnden Funktion sicherstellen kann. Dieser Test wird vom zu testenden Programmcode zunächst nicht erfüllt. Danach wird die zu testende Funktionalität implementiert, sodass der Test erfolgreich ist. Nun wird der Programmcode refaktoriert, d.h. es werden z.B. Wiederholungen entfernt, Programmcode ausgelagert oder abstrahiert. Bei einem erfolgreichen Test wird der nächste Testfall geschrieben.<sup>4</sup> Es wird in der Implementierungsphase

<sup>2</sup>Vgl. LANGR U. A. [2015, S. 153f.].

<sup>3</sup>Vgl. LANGR U. A. [2015, S. 154].

<sup>4</sup>Vgl. LANGR U. A. [2015, S. 153f.].



aufgrund des zusätzlichen Zeitaufwands durch die Tests, mehr Zeit eingeplant, als es ohne Tests der Fall wäre.

### 3 Analysephase

Nachdem die Planung des Projektes abgeschlossen ist, kann mit der Analysephase zur Ermittlung des Ist-Zustandes begonnen werden.

#### 3.1 Ist-Analyse

Das Intranet der AO stellt derzeit unter anderem folgende PHP-Anwendungen bereit: (1) Veranstaltungsanmeldung, (2) Massageplan und (3) Telefondienstliste.

(1) Die Veranstaltungsanmeldung dient der Terminabstimmung und der Anmeldung zu Veranstaltungen. Sie können unternehmens- oder abteilungsübergreifend erstellt werden. Jeder Mitarbeiter kann für seine Person eine Auswahl treffen. Des Weiteren können Administratoren festgelegt werden, die für andere Personen eine Auswahl treffen können.

(2) Der Massageplan dient der Eintragung der Mitarbeiter zur monatlichen Massage im Massageraum des Verwaltungsgebäudes der AO bei einem Masseur zu einem bestimmten Termin. Jeder Mitarbeiter kann seine Person einmal monatlich bei einem Masseur zu einem Termin eintragen. Des Weiteren können Administratoren festgelegt werden, die andere Personen ein- und austragen können.

(3) Die Telefondienstliste dient der Gewährleistung des Kundenservices der AO, dass zu einem bestimmten Anliegen Ansprechpartner bis 18:00 Uhr telefonisch zu erreichen sind. Durch die Anmeldung der Mitarbeiter zum Telefondienst verpflichten sie sich, zu dem ausgewählten Termin bis 18:00 Uhr den Kunden und den anderen Mitarbeitern zur Verfügung zu stehen. Jeder Mitarbeiter einer Abteilung kann seine Person eintragen. Des Weiteren können Abteilungsleiter Mitarbeiter ihrer Abteilung ein- und austragen.

Die Umsetzung des Projektes ist aufgrund der Umstellung des Intranets zu einem konzernweiten Mitarbeiterportal notwendig. Es werden im Mitarbeiterportal keine ähnlichen Anwendungen bereitgestellt oder Möglichkeiten gegeben, Anwendungen zu ergänzen.

#### 3.2 Wirtschaftlichkeitsanalyse

Durch eine Betrachtung der Projektkosten soll in diesem Kapitel untersucht werden, ob das Projekt auch wirtschaftlich gerechtfertigt ist.

##### 3.2.1 „Make or Buy“-Entscheidung

Zunächst wird in Zusammenarbeit mit der Abteilung *Grundsatz* ermittelt, ob es im zukünftigen Mitarbeiterportal eine Möglichkeit geben würde, die geforderten Anwendungsfälle abzubilden. Nachdem geklärt ist, dass dies nicht der Fall ist, wird in einer Internetrecherche nach Open-Source-Lösungen gesucht und bei einem Softwarelieferanten der AO ein Angebot für die Entwicklung der gewünschten Funktionalitäten eingeholt. Aufgrund der Individualität kommen keine Branchenlösungen in Frage. Auch eine externe Entwicklung kommt aufgrund eines hohen

Preises für die Umsetzung der gewünschten Funktionalitäten nicht in Frage. Daher wird die Entscheidung getroffen, die Anwendung intern zu entwickeln.

### 3.2.2 Projektkosten

Im Folgenden werden die bei der Projektdurchführung entstandenen Kosten kalkuliert. Die Kalkulation ist in Tabelle 2 zu sehen. Es werden Hard- und Softwarekomponenten sowie benötigtes Personal berücksichtigt. Da die genauen Personalkosten nicht veröffentlicht werden dürfen, wird die Kalkulation mit Stundensätzen durchgeführt, die von der Personalabteilung vorgegeben werden. Der Stundensatz eines Auszubildenden beträgt 10€, der eines Mitarbeiters 25€. Der Stundensatz für die Ressourcennutzung wird pauschal in Höhe von 40€ vorgegeben.

Außerdem werden jährlich 190€ zum Einen für die Wartung und den Betrieb des Webservers und zum Anderen für die manuelle Eintragung der Telefondienste in die Datenbank durch einen EDV-Mitarbeiter veranschlagt.

Tabelle 2: Projektkosten

Vorgang	Mitarbeiter	Zeit	Personalkosten	Ressourcenbeitrag	Gesamt
Entwicklung	1 x Auszubildende	70h	$10\text{€} \times 70\text{h} = 700\text{€}$	$15\text{€} \times 70\text{h} = 1.050\text{€}$	<b>= 1.750€</b>
Fachgespräch	2 x Mitarbeiter	6h	$2 \times 25\text{€} \times 6\text{h} = 300\text{€}$	$2 \times 15\text{€} \times 6\text{h} = 180\text{€}$	<b>= 480€</b>
Code-Review	1 x Mitarbeiter	1h	$25\text{€} \times 1\text{h} = 25\text{€}$	$15\text{€} \times 1\text{h} = 15\text{€}$	<b>= 40€</b>
Abnahme	2 x Mitarbeiter	0,5h	$2 \times 25\text{€} \times 0,5\text{h} = 25\text{€}$	$2 \times 15\text{€} \times 0,5\text{h} = 15\text{€}$	<b>= 40€</b>
					<b>= 2.310€</b>

### 3.2.3 Amortisationsdauer

Im Folgenden wird ermittelt, ab wann sich die Umsetzung des Projektes amortisiert hat. Das Projekt ist aus wirtschaftlicher Sicht nur dann sinnvoll, wenn sich die Projektkosten ab einer bestimmten Zeit ausgleichen und danach Kostenvorteile entstehen. Zur Ermittlung der Amortisationszeit wird die Zeiteinsparung durch die neue Lösung berechnet. Diese Zeiteinsparung ist in Tabelle 3 zu sehen.

(1) Bei der Veranstaltungsanmeldung kann ein Administrator durch eine verbesserte Übersicht über angemeldete oder abgemeldete Benutzer und die verkürzte An- und Abmeldung von anderen Benutzern Zeit einsparen. Es werden laut Fachbereich durchschnittlich 30 Benutzer pro Monat manuell ein- und ausgetragen.

(2) Beim Massageplan kann durch eine bessere Menüführung und der ausschließlichen Anzeige von nicht belegten Massageterminen Zeit eingespart werden. Es gibt laut Fachbereich 100 Massageterminen im Monat.

(3.1) Bei der Telefondienstliste kann durch das Hinzufügen der Telefondienste von einem EDV-Mitarbeiter, der die Datenbank mit Telefondiensten füllt, Zeit eingespart werden. Zuvor wurde

## 3 Analysephase

von jedem Abteilungsleiter eine Liste mit allen Terminen ausgedruckt, in die sich die Mitarbeiter eintragen können. Die Liste konnte nur von einem Mitarbeiter gleichzeitig gefüllt werden und musste koordiniert werden.

(3.2) Außerdem kann eine Auswertung über die Anzahl der geleisteten Telefondienste jedes Mitarbeiters mit der Statistik und nicht händisch erfolgen.

(3.3) Des Weiteren kann das Ein- und Austragen zum Telefondienst in der Anwendung geregelt werden und es kann somit das händische Füllen und Ändern einer ausgedruckten Liste vermieden werden. Es gibt laut Fachbereich 320 Telefondiensteinträge im Monat.

Tabelle 3: Zeiteinsparung pro Monat

Vorgang	Anzahl	Zeit (alt)	Zeit (neu)	Zeiteinsparung
(1) An- und Abmelden von anderen Benutzern	30	1 Min.	0,5 Min.	= 15 Min.
(2) Anzeige nicht belegter Massagetermine	100	2 Min.	1 Min.	= 100 Min.
(3.1) Telefondienste erstellen	1	5 Min.	2 Min.	= 3 Min.
(3.2) Auswertung Statistik der Telefondienste	1	5 Min.	0 Min.	= 5 Min.
(3.3) Ein- und Austragen zum Telefondienst	320	40 Sek.	20 Sek.	= 107 Min.
		= 453 Min.	= 223 Min.	= 230 Min.

**Berechnung der Kosteneinsparung**

Kosteneinsparungen =

$$12 \frac{\text{Monate}}{\text{Jahr}} \times \frac{\text{Zeiteinsparung}}{\text{Monat}} \times \text{Stundensatz}$$

$$\begin{aligned}
 &\Rightarrow 12 \frac{\text{Monate}}{\text{Jahr}} \times 230 \frac{\text{Min.}}{\text{Monat}} \times (25 + 15) \text{€} \\
 &= 2.760 \frac{\text{Min.}}{\text{Jahr}} \times 40 \text{€} \\
 &= 46 \frac{\text{Stunden}}{\text{Jahr}} \times 40 \text{€} \\
 &= 1.840 \frac{\text{€}}{\text{Jahr}}
 \end{aligned}$$

**Berechnung der Amortisationszeit**

Amortisationszeit

$$\Rightarrow \text{Projektkosten} = \frac{\text{Kosteneinsparungen}}{\text{Jahr}}$$

$\Rightarrow$  nach Jahre auflösen

$$\begin{aligned}
 2.310 \text{€} + 190 \frac{\text{€}}{\text{Jahr}} \times \text{Jahr} &= 1.840 \frac{\text{€}}{\text{Jahr}} \times \text{Jahr} \\
 &= 1,4 \text{ Jahre} \approx 17 \text{ Monate}
 \end{aligned}$$

Es ergibt sich eine Amortisationsdauer von 17 Monaten. Die Kosteneinsparung und Amortisationszeit ist im Anhang A.4, Seite iv grafisch dargestellt. Durch Personalkosteneinsparungen würden die Entwicklungskosten nach dieser Zeit ausgeglichen sein. Da von einem langfristigen

Einsatz der Software ausgegangen wird, kann das Projekt aus wirtschaftlicher Sicht als sinnvoll betrachtet werden.

### 3.3 Anwendungsfälle

Es wird im Zuge der Analyse des Projektes ein Anwendungsfalldiagramm erstellt. Dies stellt Interaktionen von Benutzern mit dem System dar und zeigt somit das erwartete Verhalten der Anwendung. Das Anwendungsfalldiagramm ist im Anhang A.5, Seite v dargestellt.

Es gibt die Akteure AO-Mitarbeiter, AO-Administrator, AO-Gruppenleiter und den Masseur. Der AO-Administrator und der AO-Gruppenleiter sind spezialisierte Akteure des AO-Mitarbeiters und somit an den gleichen Use-Case-Abläufen beteiligt wie der AO-Mitarbeiter. Der AO-Mitarbeiter wird sich für Veranstaltungen an- und abmelden, für Telefondienste ein- und austragen und für Massagen an- und abmelden können. Bevor er dies tun kann, muss er zunächst die Veranstaltungen und Telefondienste einsehen können. Der AO-Administrator kann zudem Massagen anlegen, Veranstaltungen verwalten und weitere Administratoren festlegen. Der AO-Administrator kann nicht nur sich von Veranstaltungen an-/abmelden, sondern auch andere Benutzer. Auch der AO-Gruppenleiter kann nicht nur sich zu Telefondiensten ein-/austragen, sondern auch andere Benutzer.

### 3.4 Lastenheft

Die Analysephase wird mit dem Lastenheft abgeschlossen. Es hält alle Anforderungen des Auftraggebers an das Projekt fest. Ein Auszug aus dem Lastenheft ist im Anhang A.6, Seite vi dargestellt.

Die einzelnen Anforderungen des Lastenheftes sind User-Stories (*Anwendererzählungen*), d.h. sie sind in alltäglicher Sprache formuliert und prägnant. User-Stories sind immer nach dem folgenden Schema aufgebaut: „Als *Rolle* möchte ich *Anforderung*, um *Begründung*...“

Außerdem sind sie mit der MoSCoW-Methode (Must, Should, Could, Won't have) formuliert worden. Mit Hilfe dieser Methode können die User-Stories priorisiert werden.

**Must** „Als Administrator muss ich...“ -> Anforderung muss umgesetzt werden.

**Should** „Als Administrator sollte ich...“ -> Anforderung sollte umgesetzt werden, wenn die *Must*-Anforderungen nicht beeinträchtigt werden.

**Could** „Als Administrator möchte ich...“ -> Anforderungen können umgesetzt werden, wenn die *Must*- und *Should*-Anforderungen nicht beeinträchtigt werden und Ressourcen frei sind.

**Won't** Anforderungen werden in diesem Projekt nicht umgesetzt oder werden nur für die Zukunft vorgemerkt.

## 4 Entwurfsphase

Nach der Analysephase folgt in diesem Kapitel der technische Entwurf des Projektes.

## 4.1 Zielplattform

Die Entwicklung dieses Projektes erfolgt in der deutschen Sprache, da dies die Architekturrichtlinien der AO vorgeben. Außerdem soll dieses Projekt als Webanwendung umgesetzt werden. Dies begründet sich dadurch, dass die Anmeldungen zu Veranstaltungen, der Massageplan und der Telefondienst über einen Link aus dem Mitarbeiterportal im Webbrowser aufgerufen werden sollen. Eine weitere Komponente ist das Datenbanksystem *MariaDB*. Dieses ist auf dem Webserver installiert und es laufen weitere Systeme der AO wie z.B. das Ticketsystem *Redmine* auf der *MariaDB*. Es wird daher auf die Installation eines weiteren Datenbanksystems verzichtet.

Des Weiteren wird in der Programmiersprache Java programmiert. Java ist die Standard-Programmiersprache bei der AO, die von genügend Entwicklern beherrscht wird, wodurch eine Wartung des Programmcodes gewährleistet ist. Für diese Webanwendung wird die Spezifikation *Java EE 7* verwendet. Diese Spezifikation wird bereits in anderen Web-Projekten der AO eingesetzt und ist etabliert.

Außerdem wird die Webanwendung auf dem *JBoss Enterprise Application Platform (EAP)* ausgeführt. Die AO besitzt Lizenzen für den Einsatz und den Support dieses Applicationsservers, der zudem auch in den bereits angesprochenen Web-Projekten der AO verwendet wird und sich als geeignet erwies. Ein Applicationserver ist ein Server innerhalb eines Computernetzwerks, der Anwendungsprogramme ausführt.<sup>5</sup>

## 4.2 Architekturdesign

Das Projekt basiert auf dem Architekturmuster Model View Controller (*MVC*) und Erweiterungen aus dem Java EE 7-Standard. Das Architekturmuster ist in einem Komponentendiagramm im Anhang A.7, Seite vii dargestellt.

Gemäß dieses Musters lässt sich Software in die drei Einheiten *Model* (Datenhaltung), *View* (Präsentation) und *Controller* (Anwendungssteuerung) unterteilen.<sup>6</sup> Die Erweiterungen des Java-Standards sind das *Repository*, der *Service* und das *ViewModel*. Im *Model* werden die domainspezifischen Entitys und Beziehungen sowie die Klassen der Businesslogik gespeichert und verwaltet. Eine Entity ist ein Konstrukt, in dem Informationen abgespeichert und bearbeitet werden können und das von einer Datenbank persistiert werden kann.

Das *Repository* ist die Persistenzschicht der Entitys und stellt die Verbindung zur Datenbank sicher. Der *Service* bildet die Use-Cases durch das Management mehrerer Entitys ab. Der *Controller* ist als Verbindung zwischen dem *Model* und der *View* für die Steuerung der Daten zuständig. Das *ViewModel* bereitet die Entitys des *Domainmodels* für den *View* auf. Im *View* werden Daten abgebildet und Benutzereingaben entgegengenommen. Das Komponentendiagramm veranschaulicht die Abhängigkeiten der Komponenten Persistence, Domain und Web.

Die *Domain* speichert die domainspezifischen Daten in der Komponente *Model*. Innerhalb der *Domain* greift der *Service* und das *Repository* auf das *Model* zu. Die *Persistence* enthält das *JpaRepository* (Java Persistence API (*JPA*)), das für den Zugriff auf die Datenbank zuständig

---

<sup>5</sup>Vgl. SALVANOS [2014, S. 521].

<sup>6</sup> Vgl. SALVANOS [2014, S. 43f.].

ist. **JPA** ist für das **ORM** (Objekt-Relationales-Mapping) zuständig, d.h. es wird ein Plain Old Java Objekt (**POJO**) in einer Tabelle abgebildet und umgekehrt.<sup>7</sup> Die **Persistence** nutzt die Schnittstelle der **Domain**. Das **Web** enthält den **Controller**, der von dem **Service** der **Domain** abhängig ist. Des Weiteren enthält es den **View**, der von dem **Controller** und dem **ViewModel** abhängig ist. Das **Web** nutzt ebenfalls die Schnittstelle der **Domain**. Die Trennung der einzelnen Schichten erfolgt aufgrund der Wiederverwendbarkeit und der losen Kopplung der einzelnen Schichten. Außerdem ist die **Domain** gekapselt und alle anderen Schichten hängen von dieser ab. Die **Domain** kann aufgrund der Unabhängigkeit in anderen Projekten wiederverwendet werden. Im Folgenden werden die Schichten nochmals kurz zur Übersicht erläutert.

**View** Anzeigen der ViewModels und Entgegennahme von Benutzereingaben

**ViewModel** Spezielle Aufbereitung des Domainmodels für den View

**Controller** Management der Kommunikation von ViewModel und Service

**Service** Abbildung der Use-Cases durch Management mehrerer Entitys

**Domainmodel** domainspezifische Entitys und Beziehungen sowie Businesslogik

**Repository** Persistenzschicht für Entitys

### 4.3 Entwurf der Benutzeroberfläche

Die Benutzeroberfläche der Webanwendung soll klar strukturiert und einfach sein. Hierzu werden zunächst MockUps mit *Balsamiq MockUps* erstellt, die den groben Aufbau und die Anordnung der Elemente der einzelnen Seiten zeigen. Zwei MockUps werden im Anhang A.8, Seite viii dargestellt. Diese werden zum Einen als Orientierung für Entwickler erstellt, zum Anderen werden sie intensiv in die Gespräche mit dem Fachbereich einbezogen. Die Mitarbeiter des Fachbereichs, die später als Administratoren die Webanwendung und die Benutzer betreuen, haben sich für ein Design, das in das Corporate Design der AO passt und schon in anderen Webanwendungen verwendet wird, entschieden. Zudem soll die Darstellung von mehreren Kalendertagen in Form eines Kalendermonats zu sehen sein und nicht in Form einer Liste von einzelnen Terminen. Weiter sollten von Mitarbeitern belegte Massage- und Telefondiensttermine in der Kalendermonatsansicht für andere Mitarbeiter kenntlich gemacht werden, damit diese eine sofortige Rückmeldung über belegte Massage- und Telefondiensttermine erhalten.

### 4.4 Datenmodell

Das Datenmodell wird in einem Entity-Relationship-Model (**ERM**) visualisiert. Das **ERM** ist im Anhang A.9, Seite ix dargestellt. Es lassen sich alle Daten in die Entitätstypen **Veranstaltung**, **Option**, **Auswahl**, **Gruppe** und **Benutzer** speichern.

Eine **Veranstaltung** speichert die Informationen zu einer Veranstaltung, einem Telefondienst oder einer Massage. Mit der Beziehung zum Entitätstyp **Art** kann festgelegt werden, um welche

---

<sup>7</sup>Vgl. SALVANOS [2014, S. 510f.].

**Art** von Termin (Veranstaltung, Telefondienst, Massage) es sich handelt. Eine **Art** kann mehreren **Veranstaltungen** zugeordnet werden, jedoch kann eine **Veranstaltung** nur eine **Art** haben. In dem Attribut **Datum** kann das Datum des jeweiligen Termins und in der **Deadline** das letzte Datum einer möglichen An- und Abmeldung gespeichert werden. Das Attribut **Info** kann den Titel einer **Veranstaltung** speichern und für mögliche weitere Informationen genutzt werden. Der Entitätstyp **Option** speichert die Auswahlmöglichkeiten der **Veranstaltung** in dem Attribut **Name**. Die Entitätstypen **Veranstaltung** und **Option** sind in einer m:n-Beziehung miteinander verbunden, da eine **Veranstaltung** mehrere Auswahlmöglichkeiten haben muss und eine **Option**, wie z.B. „Ich nehme teil.“, zu mehreren **Veranstaltungen** zugeordnet werden kann. In dem Entitätstyp **Benutzer** kann der Benutzername zu einem AO-Mitarbeiter in dem Attribut **Name** gespeichert werden. Ein **Benutzer** kann an mehreren Veranstaltungen teilnehmen und ist daher mit dem Entitätstyp **Veranstaltung** verbunden. Eine **Veranstaltung** kann mit mehreren **Benutzern** verbunden sein. Die **Benutzer** treffen für eine **Veranstaltung** eine Entscheidung über **Optionen**, die in dem Attribut **Auswahl** gespeichert werden. Es kann beispielsweise die **Option** „Ich nehme teil.“ mit der **Auswahl** „Ja“ oder „Nein“ der **Benutzer** gespeichert werden. Der Entitätstyp **Benutzer** ist ebenfalls mit dem Entitätstyp **Gruppe** verbunden, die den Namen einer Abteilung oder einer Arbeitsgruppe in dem Attribut **Name** speichert. Eine **Gruppe** ist mit einer m:n-Beziehung mit dem **Benutzer** verbunden, da eine **Gruppe** mehrere **Benutzer** haben kann und ein **Benutzer** in mehreren Abteilungen oder Arbeitsgruppen tätig sein kann. Am folgenden Beispiel soll das Datenmodell anhand des Anlegens einer Massage verdeutlicht werden:

Ein Administrator möchte einen Massagetermin beim Masseur am 01.12.2016 um 11:00 Uhr anlegen und den 30.11.2016 als letzten möglichen Tag der Ein- und Austragung festlegen. Es wird nun die **Veranstaltung** mit einer generierten ID, dem **Namen** *Masseur*, dem 01.12.2016 als **Datum** und dem 30.11.2016 als **Deadline** angelegt. Es wird ein Eintrag in den Entitätstypen **Art** angelegt. Außerdem wird ein Eintrag in die Beziehung **hat** mit der **Option** 11:00 Uhr angelegt. Weiter werden Einträge zu allen **Gruppen** in die Beziehung **lädt ein** angelegt, da sich alle **Benutzer** für diesen Termin eintragen können. Trägt sich nun ein **Benutzer** für diesen Termin ein, wird ein Eintrag in die Beziehung **entscheidet** angelegt, mit der **Auswahl** *Ja*.

## 4.5 Geschäftslogik

Die Geschäftslogik umfasst die Klassen, die die Logik der Webanwendung beinhalten. Aufgrund der testgetriebenen Programmierung entstehen die Klassen erst schrittweise während der Implementierungsphase. Die Komponenten der Anwendung stehen bereits zu Anfang des Projektes fest und werden im Komponentendiagramm (siehe Anhang A.7) visualisiert.

## 4.6 Deployment

Die Webanwendung ist auf verschiedene Systeme verteilt und wird in einem Verteilungsdiagramm visualisiert. Das Verteilungsdiagramm ist im Anhang A.10, Seite x dargestellt. In der



Entwicklungsumgebung *Eclipse* läuft [AO-Anmeldungen](#) und *Gradle*. *Gradle* ist ein Build-Management-Tool, das Projekte baut. Im Anhang [A.11](#), Seite [xi](#) ist die `build.gradle`-Datei dargestellt. Diese definiert den Build und die Abhängigkeiten des Projektes. Für den Build gibt es z.B. eine Abhängigkeit auf die `javax:javaee-api` in der Version 7.0, um Annotationen des [Java EE 7](#)-Standards auflösen zu können. Auf dem Buildserver läuft *Tomcat*, auf dem *Jenkins*, *SCM-Manager* (Source Code Management) und *Artifactory* laufen. *Tomcat* ist ein Webserver und Container für Java-Servlets, auf dem Java ausgeführt werden kann. *Jenkins* ist ein web-basiertes Tool zur Continuous Integration. *Artifactory* ist ein Programm, mit dem benötigte Bibliotheken für die Programmierung verwaltet werden.

[AO-Anmeldungen](#) wird ins [SCM](#) deployt. *Eclipse* enthält die Abhängigkeiten für Projekte von *Artifactory*. *Jenkins*, auf dem *Gradle* läuft, greift auf den [SCM-Manager](#) und *Artifactory* zu. Auf dem Webserver läuft die Anwendung [AO-Anmeldungen](#) auf dem *JBoss*. Der *JBoss* greift auf die *MariaDB*-Datenbank auf dem Datenbank-Server zu. Letztendlich ruft der Client mit dem Webbrowser *Firefox* die Anwendung [AO-Anmeldungen](#) auf.

Das Verbinden von *JBoss* mit der Datenbank wird automatisch durchgeführt. Hierfür wird eine *JBoss*-Konfigurationsdatei angelegt. Diese ist im Anhang [A.12](#), Seite [xii](#) dargestellt. Zunächst wird für den *JBoss* der Datenbank-Treiber und im Anschluss die Zugriffsdaten auf die Datenbank definiert.

## 4.7 Pflichtenheft

Die Entwurfsphase wird mit dem Pflichtenheft abgeschlossen. Das Pflichtenheft baut auf dem Lastenheft (siehe Anhang [A.6](#)) auf. Es enthält die technischen Anforderungen an das Projekt und die genauen Vorgaben des Auftragnehmers. Ein Auszug aus dem Pflichtenheft ist im Anhang [A.13](#), Seite [xiii](#) dargestellt.

# 5 Implementierungsphase

In diesem Kapitel wird die Implementierung der einzelnen Komponenten erläutert.

## 5.1 Iterationsplanung

Bevor mit der Implementierung der Software begonnen wird, wird zunächst ein Iterationsplan erstellt. Der Iterationsplan ist im Anhang [A.14](#), Seite [xiv](#) dargestellt. Dieser definiert die einzelnen Iterationsschritte und deren Reihenfolge. In jeder Iteration wird jeweils ein Teil der Webanwendung (Veranstaltungsanmeldung, Messageplan, Telefondienst) vollständig implementiert, dessen Funktion im Anschluss mit dem Fachbereich abgestimmt werden kann.

## 5.2 Aufsetzen der benötigten Elemente

Es wird die Entwicklungsumgebung *Eclipse* Mars Java EE genutzt. Zunächst wird ein lokaler *JBoss* Applicationserver installiert, der mit *Eclipse* verbunden wird. So kann das Projekt während der Implementierung auf dem *JBoss* deployt und die Oberfläche gestaltet werden.



Außerdem wird ein *Jenkins*-Projekt eingerichtet, um eine kontinuierliche Integration der Anwendung zu gewährleisten. Dieses Projekt wird mit statischen Codeanalyse-Tools wie z.B. *CheckStyle* eingerichtet, die die Einhaltung von Java-Coderichtlinien überwachen. Im Anhang A.15, Seite xv ist ein Screenshot des Codeanalyse-Tools *CheckStyle* im *Jenkins* dargestellt. Auf Grundlage dieser priorisierten Meldungen kann der Programmcode verbessert werden.

Des Weiteren wird eine *MariaDB*-Datenbank in dem bereits installierten Datenbanksystem auf dem Webserver der AO erstellt. Es wird in *Eclipse* innerhalb eines Projektes programmiert. In dieses Projekt werden anschließend die Verbindungsdaten zum *JBoss* hinzugefügt.

### 5.3 Implementierung der gemeinsam genutzten Entitys

Wie im ERM (siehe Anhang A.9) zu sehen ist, nutzen die drei Komponenten Veranstaltungsanmeldung, Messageplan und Telefondienstliste gemeinsame Entitätstypen wie z.B. den Entitätstypen *Veranstaltung*. Dieser speichert zunächst die Art der *Veranstaltung* (*Veranstaltung*, *Message*, *Telefondienst*) und die dazugehörigen Informationen wie z.B. *Name* oder *Tag* der *Veranstaltung*. Weiter ist das ERM Grundlage für die Programmierung der Entitys. Aus den Entitätstypen werden Klassen, die Attribute der Entitätstypen werden ebenfalls Attribute der Klassen. Die n:m-Beziehungen werden so aufgelöst, dass separate Klassen erstellt werden wie z.B. die Klasse *Benutzeradministrator*, die die Beziehung zwischen der Klasse *Veranstaltung* und *Benutzer* abbildet. Diese Klasse hat nun die Attribute vom Typen *Benutzer* und *Veranstaltung*. Die Klassen *Veranstaltung* und *Benutzer* haben ein *Set*<sup>8</sup> der *Benutzeradministrator* als Attribut.

### 5.4 Implementierung der Persistence

Die Implementierung der Persistence umfasst die Repositorys der Entitys, mit denen mittels *JPA* auf die Datenbank zugegriffen wird. Es können so z.B. Listen aller *Benutzer* oder bestimmte *Benutzer* aus der Datenbank gelesen werden. Im Anhang A.16, Seite xvi ist ein Listing einer Entity mit *JPA* dargestellt. Es werden verschiedene *Java EE 7*-Annotationen wie z.B. `javax.persistence.Entity` genutzt. Die Annotation `@Entity` an der Klasse *Benutzer* kennzeichnet diese als Entity, die in die Datenbank geschrieben und aus dieser gelesen werden soll. Die Annotation `@Id` kennzeichnet ein Attribut als Primärschlüssel.<sup>9</sup> Die Annotation `@GeneratedValue` kennzeichnet, dass der Primärschlüssel automatisch generiert werden sollte.<sup>10</sup> Die Annotation `@OneToMany` kennzeichnet neben den weiteren Annotationen `@OneToOne` und `@ManyToMany` die Beziehung der Entity *Benutzer* zur Entity *BenutzerGruppe* als 1:n-Beziehung.<sup>11</sup> *JPA* ermöglicht somit eine Speicherung und Abbildung der Java-Objekte in der *MariaDB*-Datenbank.

---

<sup>8</sup>Ein *Set* in Java ist eine Menge von Elementen.

<sup>9</sup>Vgl. SALVANOS [2014, S. 556f.].

<sup>10</sup>Vgl. SALVANOS [2014, S. 573f.].

<sup>11</sup>Vgl. SALVANOS [2014, S. 583].

## 5.5 Implementierung der Domain inkl. Geschäftslogik

Die Implementierung der Domain umfasst die Implementierung der Services, die die Use-Cases abbilden. Es wird bei einem Anwendungsfall, der mehrere Entitys benutzt, ein Service der beteiligten Entitys erstellt und implementiert. Außerdem erhält der Service die Geschäftslogik, die sicherstellt, dass sich z.B. ein **Benutzer** nicht doppelt zu einer **Veranstaltung** anmelden kann und eine Nachricht an den View weitergeleitet wird, um den Benutzer zu informieren. Im Anhang A.17, Seite xvii ist ein Auszug der Klasse `MessageService` dargestellt. In dieser wird beispielsweise geprüft, ob ein Benutzer sich für eine Message eintragen darf oder ob er in dem angefragten Monat bereits für eine Message eingetragen ist. Im Kapitel 5.6 wird hierauf näher eingegangen.

Außerdem wird für die Domain-Entitys sichergestellt, dass diese in einem validen Zustand sind. Beispielsweise kann eine Veranstaltung nur angelegt werden, wenn der Termin hierfür in der Zukunft liegt.

## 5.6 Test

Während der Implementierung werden *JUnit*-Tests erstellt und durchgeführt. Es wird das Konzept des TDD angewandt, welches bereits in Kapitel 2.3 beschrieben wird. Ein Beispiel-Test ist in im Anhang A.18, xix dargestellt. Es wird hier der *MessageService* mit *JUnit* und *Mockito* getestet. Mockito ist ein Framework zum Erzeugen von Mock-Objekten, das dazu dient, Objekte unabhängig von ihrer Umgebung zu testen.

## 5.7 Implementierung der Views inkl. Benutzeroberfläche

Die Implementierung des Views umfasst die Implementierung der Elemente, die nur für die View benötigt werden, wie z.B. die `KalenderView`, welche für die Darstellung des Date-Pickers zuständig ist.

Die Benutzeroberfläche der Anwendung muss im Corporate Design der AO gestaltet sein. Es werden vorgegebene Cascading Style Sheets (CSS)-Dateien in das Projekt eingebunden und an das Projekt angepasst. Weiter wird zur Gestaltung der Webseiten Java Server Faces (JSF) verwendet, welches z.B. CSS-Dateien einbindet und die Navigation definiert. JSF ist ein Java-Standard für die Entwicklung von GUIs (Graphical User Interface).<sup>12</sup> Mit Hilfe von JSF kann auf die Views zugegriffen und diese Daten dargestellt werden. Ein Beispiel für eine JSF-Datei ist im Anhang A.19, Seite xxi dargestellt. Das Hypertext Markup Language (HTML)-Dokument wird mit dem `h:form`-Element gerendert.<sup>13</sup> Mit dem `c:forEach`-Element kann jedes Element einer Liste durchlaufen werden. Ein Hash-Zeichen (#) kennzeichnet den Aufruf einer Methode, in diesem Beispiel eine Getter-Methode des `VeranstaltungControllers`. Außerdem können JSF-Bestandteile importiert werden mit dem Element `ui:include`, wie hier z.B. der `footer.xhtml`.

---

<sup>12</sup>Vgl. SALVANOS [2014, S. 701f.].

<sup>13</sup>Vgl. SALVANOS [2014, S. 704].

## 6 Abnahme- und Einführungsphase

Der Implementierungsphase folgt die Abnahme- und Einführungsphase, in der die abschließenden Gespräche mit dem Auftraggeber und dem Fachbereich geführt und die Webanwendung deployt wird.

### 6.1 Abnahme durch den Fachbereich

Nachdem die Webanwendung fertiggestellt ist, kann diese dem Fachbereich und dem Auftraggeber vorgelegt werden. Aufgrund der agilen Entwicklung ist der Fachbereich bereits mit dem Aufbau und den Funktionen der Webanwendung vertraut. Es kann bereits während der Entwicklung auf Wünsche und Anregungen des Fachbereichs eingegangen werden. Diese werden dann direkt in die Webanwendung eingebunden. Somit ergeben sich bei der Endabnahme keine Schwierigkeiten und die Einführung der Webanwendung kann beginnen.

### 6.2 Deployment und Einführung

Die Webanwendung wird zunächst vom Buildserver auf den *JBoss* des Webserver der AO deployt. Die Webanwendung ist nun innerhalb des Netzwerkes unter der Adresse `ao-anmeldungen` zu erreichen. Screenshots der Anwendung sind im Anhang A.20, Seite xxii dargestellt. Es wird für jeden Benutzer eine Desktopverknüpfung der Webadresse angelegt, um die Webanwendung schnell im Standardbrowser zu öffnen.

## 7 Dokumentation

Die Dokumentation setzt sich aus dem Benutzerhandbuch und der Entwicklerdokumentation zusammen.

Das Benutzerhandbuch erläutert den Aufbau und die Funktionen der Anwendung und wie diese genutzt werden kann. Ein Ausschnitt des Benutzerhandbuchs ist im Anhang A.21, Seite xxiii dargestellt.

Die Entwicklerdokumentation erläutert die einzelnen Klassen, deren Funktionen, Attribute und Methoden sowie deren Beziehungen zu anderen Klassen. Mithilfe dieser Dokumentation sollen Entwickler einen Überblick über die Webanwendung erhalten. Die Dokumentation wird mit *JavaDoc* automatisch generiert. Ein Ausschnitt der Entwicklerdokumentation ist im Anhang A.22, Seite xxiv dargestellt.

Zusätzlich wird aus dem Quellcode ein Klassendiagramm erzeugt, das einen guten Überblick aller Klassen und deren Attribute und Methoden bietet. Ein Ausschnitt des Klassendiagramms ist im Anhang A.23, Seite xxv dargestellt. Der Ausschnitt zeigt die für die Messageauswahl benötigten Klassen. Es werden alle Attribute und Methoden aufgrund der Übersichtlichkeit entfernt. Es ist zu erkennen, dass die Klassen dem in Kapitel 4.2 definierten Architekturdesign folgen.

## 8 Fazit

In diesem Kapitel wird zum Ende des Projektes ein Fazit gezogen und ein Ausblick der Zukunft der Webanwendung gegeben.

### 8.1 Soll-/Ist-Vergleich

Im Rückblick betrachtet sind die Anforderungen des Projekt im vollen Umfang erfüllt worden. Die Webanwendung bietet den Mitarbeitern der AO die Möglichkeit, sich zu Veranstaltungen, Massagen oder Telefondiensten anzumelden. Außerdem können Administratoren die Webanwendung steuern. Der zu Beginn erstellte Projektplan (siehe Kapitel 2) konnte zeitlich eingehalten werden. Es wurden zwar in den Phasen *Entwurf* (-1h), *Implementierung inkl. Tests* (+2h) und *Erstellung der Dokumentation* (-1h) Abweichungen generiert, das Projekt aber dennoch im festgelegten Zeitraum von 70 Stunden umgesetzt. Die zeitlichen Abweichungen kamen aufgrund von Mehraufwand bei der Programmierung der Tests, gemindertem Aufwand bei der Erstellung des Pflichtenheftes aufgrund der gelungenen Planung der Geschäftslogik und ebenfalls gemindertem Aufwand bei der Erstellung der Benutzerdokumentation aufgrund von zuvor erstellten User-Stories zustande.

### 8.2 Ausblick

In den kommenden Monaten soll das Mitarbeiterportal der AO eingeführt werden. In Zuge dessen soll eine Verknüpfung im Mitarbeiterportal zur Webanwendung angelegt werden, um eine Möglichkeit zu schaffen, die Anmeldungen über das Mitarbeiterportal zu erreichen. Außerdem sollen die Mitarbeiter mittels einer E-Mail und dem Verweis auf das Benutzerhandbuch kurz in die Webanwendung eingeführt werden. Bei Fragen der Anwender steht ein Support telefonisch und mittels eines Ticketsystems zur Verfügung. Außerdem soll eine ausgesuchte Gruppe von künftigen Administratoren und Gruppenleitern (*Key-User*) eine kurze Einführung in die Webanwendung erhalten. Diese können dann wiederum andere Benutzer in die Anwendung einführen.

Des Weiteren ist die Webanwendung bewusst modular entwickelt worden, sodass Erweiterungen möglich sind. Die Abteilung Grundsatz hat bereits eine neue Anforderung an die Webanwendung gestellt. So sollen Ressourcen, die nicht im E-Mail-System *IBM Lotus Notes* aufgenommen werden können, ebenfalls über die Webanwendung zu buchen sein können. Außerdem benötigt ein Tochterunternehmen der VGH, die *Provinzial Krankenversicherung*, die Anwendung, um Anmeldungen der Mitarbeiter zu Veranstaltungen verwalten können.

## Literaturverzeichnis

### Alte Oldenburger Krankenversicherung AG 2016

ALTE OLDENBURGER KRANKENVERSICHERUNG AG: *Geschäftsbericht über das Jahr 2015*. [https://www.alte-oldenburger.de/web/export/sites/aob/\\_resources/download\\_galerien/downloads\\_pdf/daten\\_und\\_fakten/Geschaeftsbericht\\_2015\\_ALTE\\_OLDENBURGER\\_Krankenversicherung\\_AG.pdf](https://www.alte-oldenburger.de/web/export/sites/aob/_resources/download_galerien/downloads_pdf/daten_und_fakten/Geschaeftsbericht_2015_ALTE_OLDENBURGER_Krankenversicherung_AG.pdf). Version: 2016. – Zugriff am: 27.11.2016

### Langr u. a. 2015

LANGR, Jeff ; THOMAS, Dave ; HUNT, Andy: *Pragmatic Unit Testing: in Java 8 with JUnit*. USA : The Pragmatic Programmers, 2015. – ISBN 9781941222591

### Salvanos 2014

SALVANOS, Alexander: *Professionell entwickeln mit Java EE 7: Das umfassende Handbuch*. Galileo Computing, 2014. – ISBN 9783836220040

## Eidesstattliche Erklärung

Ich, Maria Elena Hollen, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

*AO-Anmeldungen – Entwicklung einer Webanwendung mit Java EE 7*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Lingen, den 01. Dezember 2016



---

MARIA ELENA HOLLEN

## A Anhang

### A.1 Screenshot der bisherigen Anwendung im Intranet



Abbildung 1: Screenshot: Masseurauswahl in der bisherigen Anwendung

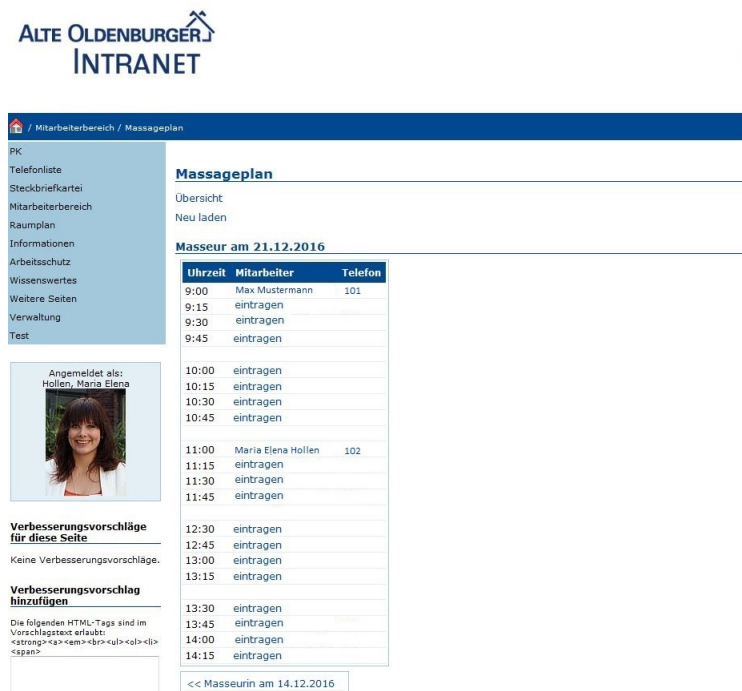


Abbildung 2: Screenshot: Massageauswahl in der bisherigen Anwendung

## A.2 Detaillierte Zeitplanung

Tabelle 4: Projektstrukturplan

Phase	Stunden
<b>Analyse</b>	<b>10h</b>
Durchführen der Ist-Analyse (Gespräche mit Ansprechpartnern des Fachbereichs)	3h
Durchführen der Wirtschaftlichkeitsanalyse	1h
Entwerfen des Use-Case-Diagramms	1h
Erstellen des Lastenhefts	5h
<b>Entwurf</b>	<b>11h</b>
Erstellen des Verteilungsdiagramms	0,5h
Erstellen des Komponentendiagramms	0,5h
Entwerfen des Domainmodels	3h
Planen der Geschäftslogik	2h
Entwerfen der Benutzeroberflächen	1h
Erstellen des Pflichtenhefts	4h
<b>Implementierung inkl. Tests</b>	<b>42h</b>
Aufsetzen des Application Servers	0,5h
Anlegen der Datenbank	0,5h
Einrichten der statischen Code-Analyse	0,5h
Einrichten des Continuous Integration	0,5h
Implementieren der gemeinsam genutzten Entities	8h
<b>Implementieren der Geschäftslogik</b>	<b>24h</b>
– Implementieren der Veranstaltungsanmeldungs-Logik	7h
– Implementieren der Messageplan-Logik	8h
– Implementieren der Telefondienstlisten-Logik	9h
Implementieren der Benutzeroberflächen	8h
<b>Abnahme und Einführung</b>	<b>2h</b>
Abnahme durch die Fachbereiche	0,5h
Deployment	0,5h
Code-Review mit dem Ausbilder	1h
<b>Erstellung der Dokumentation</b>	<b>5h</b>
Erstellen der Benutzerdokumentation	4h
Erstellen der Entwicklerdokumentation inkl. Klassendiagramm	1h
<b>Gesamt</b>	<b>70h</b>



## **A.3 Ressourcenplanung**

### **Hardware**

- Büroarbeitsplatz mit Thin-Client
- Webserver

### **Software**

- Windows 7 Enterprise mit Service Pack 1 – Betriebssystem
- Eclipse Mars Java EE IDE für Web-Entwickler – Entwicklungsumgebung
- JUnit – Framework zum Durchführen von Unit-Tests
- Red Hat JBoss Enterprise Application Platform – Application Server
- MariaDB – Datenbanksystem
- MySQL Workbench – Verwaltungswerkzeug für Datenbanksysteme
- Enterprise Architect – Programm zur Erstellung von Diagrammen und Modellen
- Git – verteilte Versionsverwaltung
- MiKTeX – Distribution des Textsatzsystems  $\text{\LaTeX}$
- Eclipse Luna mit TeXlipse – Entwicklungsumgebung  $\text{\LaTeX}$
- Balsamiq – Programm zur Erstellung von MockUps

### **Personal**

- Mitarbeiter des Fachbereichs – Festlegung der Anforderungen und Abnahme des Projektes
- Entwicklerin – Umsetzung des Projektes
- Anwendungsentwickler – Review des Codes

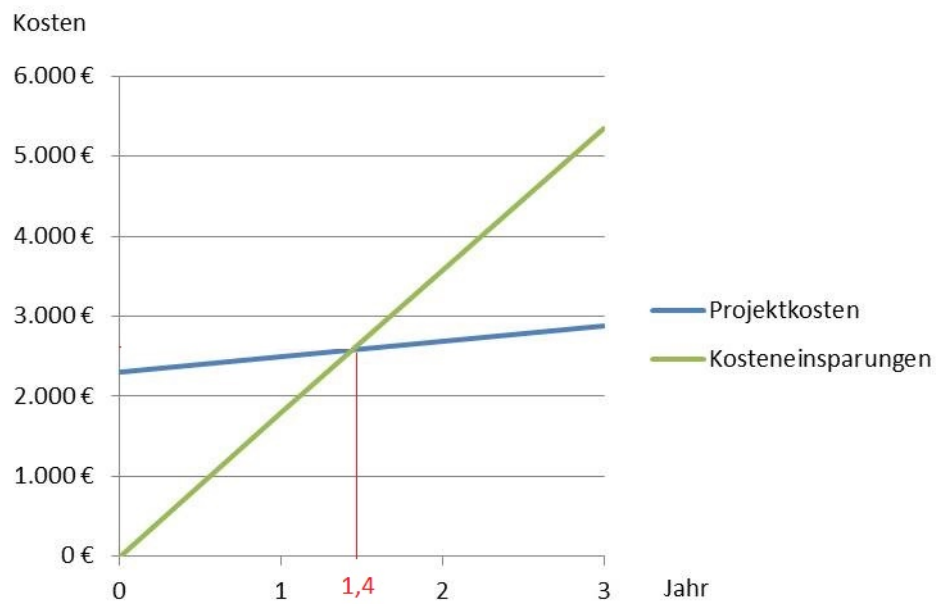
**A.4 Break-Even-Analyse**

Abbildung 3: Break-Even-Analyse

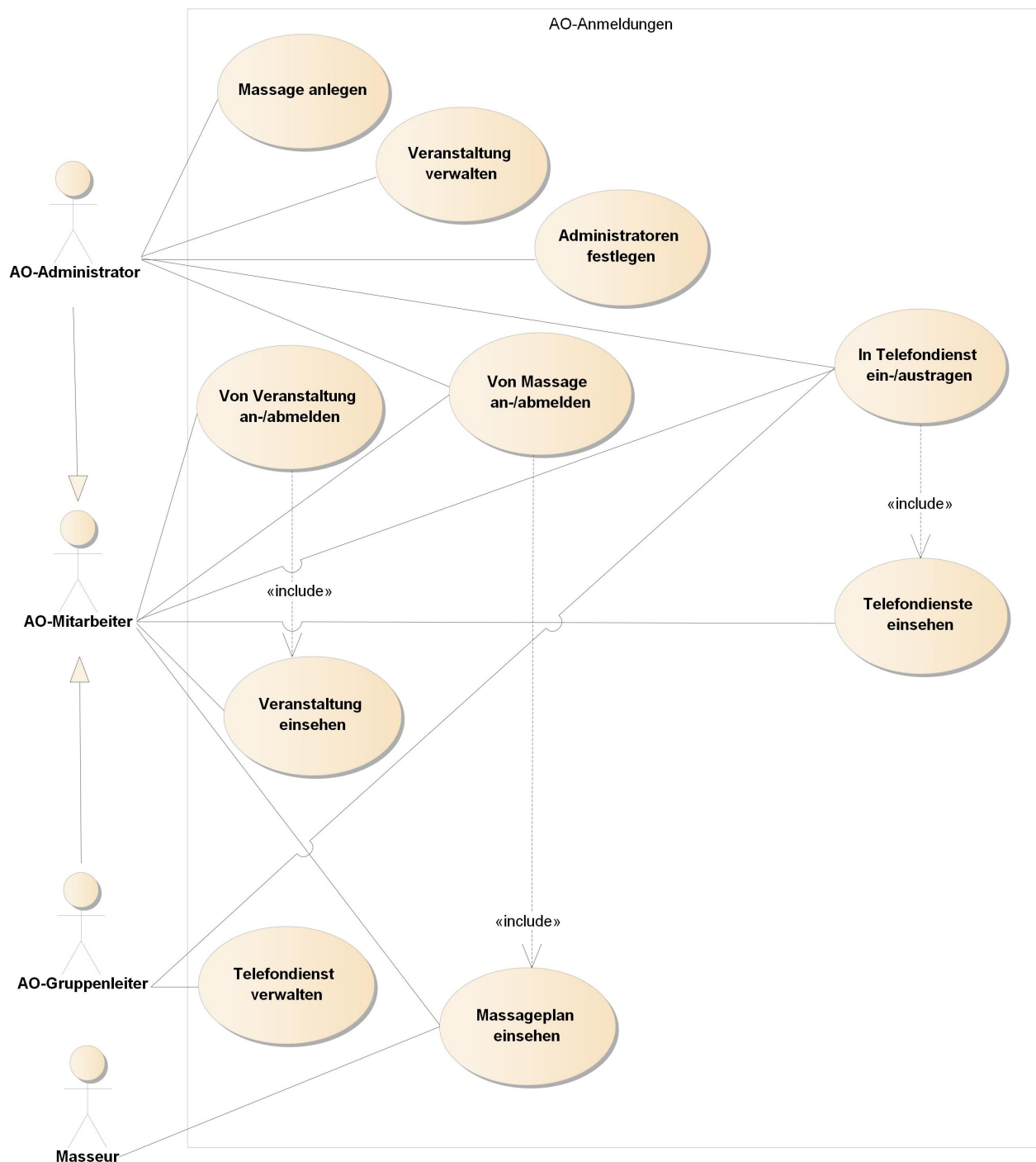
**A.5 Anwendungsfalldiagramm**

Abbildung 4: Anwendungsfalldiagramm

## A.6 Lastenheft (Auszug)

Im folgenden Auszug des Lastenheft werden Anforderungen an die Anwendung festgehalten.

### Veranstaltungsanmeldung

- Als Administrator muss ich eine Veranstaltung mit einem Titel, einer Beschreibung und einem Tag erstellen können, um den Benutzern die nötigen Informationen zu einer Veranstaltung geben zu können.
- Als Administrator muss ich zu einer Veranstaltung Benutzer sowie Gruppen einladen können, damit die Erstellung von Veranstaltungen und die Einladung von Benutzern einfach gestaltet ist.
- Als Administrator sollte ich alle unentschlossenen Benutzer anzeigen lassen können, um diesen eine Erinnerungsnachricht schicken zu können.
- Als Benutzer möchte ich die Möglichkeit haben, nach anderen Benutzern zu suchen, um deren Anmeldung einsehen zu können.
- ...

### Telefondienstliste

- Als Gruppenleiter muss ich eine Statistik über die abgeleisteten Telefondienste der Benutzer einsehen können, um Benutzer anzuweisen, sich für zukünftige Telefondienste einzutragen.
- Als Gruppenleiter muss ich Benutzer zu einem Telefondienst ein- und austragen können, um die Telefondienste organisieren zu können und das Ein- und Austragen für andere Benutzer übernehmen zu können.
- Als Benutzer muss ich mich selbst für einen Telefondienst eintragen können, um meine Telefondienste zeitlich einplanen zu können.
- ...

### Massageplan

- Als Administrator muss ich Benutzer zu einer Massage an- und abmelden können, um von Benutzern nicht wahrgenommene Termine an andere Benutzer vergeben zu können.
- Als Masseur muss ich den Massageplan inkl. eingetragener Benutzer und deren Telefonnummer einsehen können, um diese bei Nicht-Erscheinen zum Termin anrufen zu können.
- Als Benutzer möchte ich bei der Suche nach einem freien Massagetermin nur freie Termine angezeigt bekommen, um wenig Zeit bei der Eintragung zu verlieren.
- ...

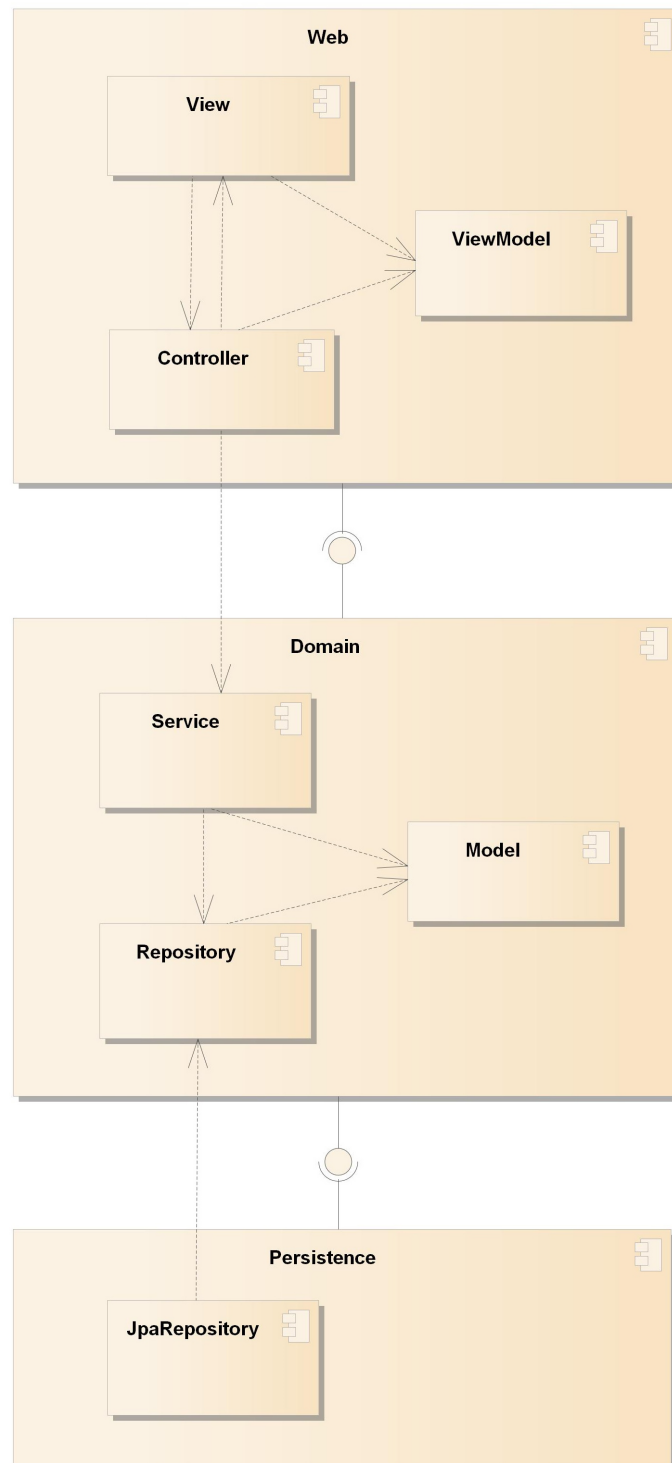
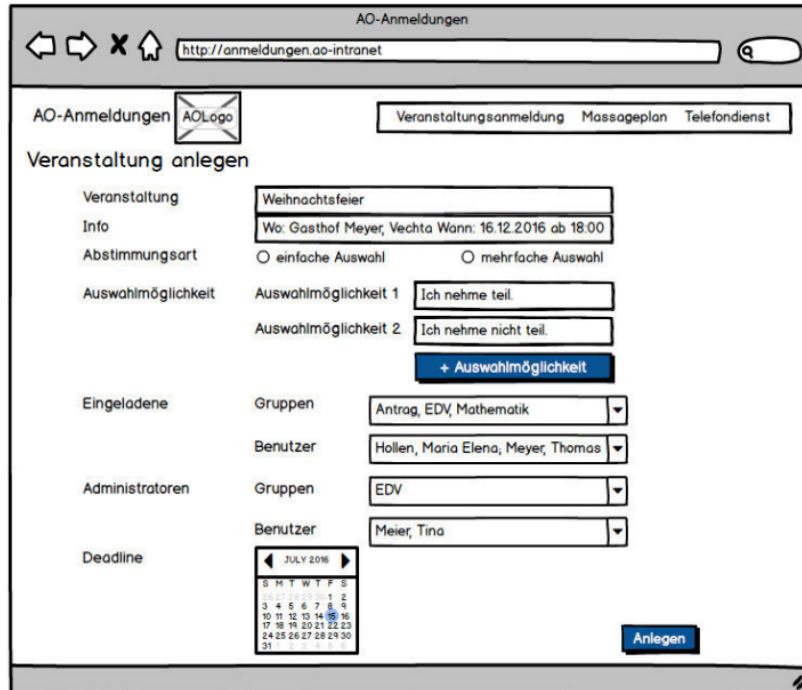

**A.7 Komponentendiagramm**

Abbildung 5: Komponentendiagramm

## A.8 Oberflächenentwürfe



AO-Anmeldungen  Veranstaltungsanmeldung Massageplan Telefondienst

### Veranstaltung anlegen

Veranstaltung:

Info:

Abstimmungsart: ☐ einfache Auswahl ☐ mehrfache Auswahl

Auswahlmöglichkeit

Auswahlmöglichkeit 1:

Auswahlmöglichkeit 2:

Eingeladene

Gruppen:

Benutzer:

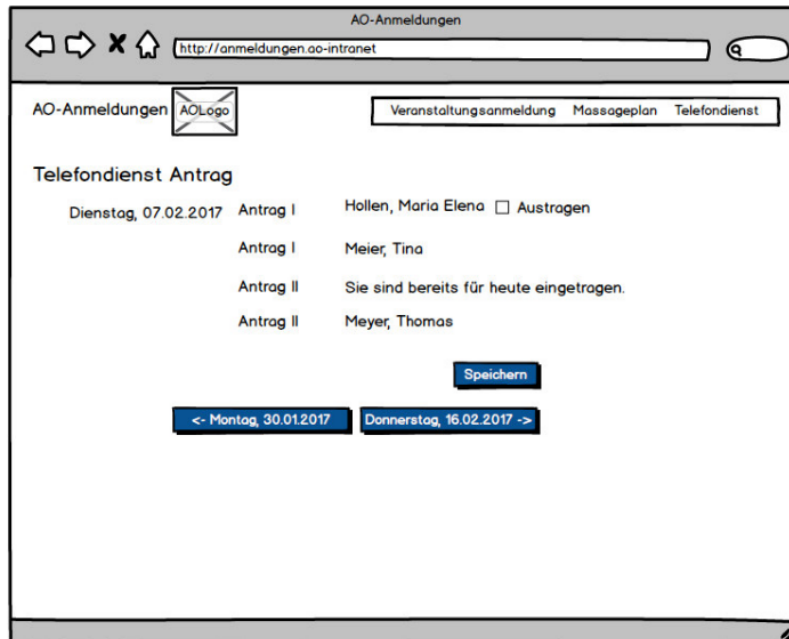
Administratoren


Gruppen:

Benutzer:

Deadline:

Abbildung 6: MockUp: Anlegen einer Veranstaltung



AO-Anmeldungen  Veranstaltungsanmeldung Massageplan Telefondienst

### Telefondienst Antrag

Dienstag, 07.02.2017

Antrag I:  ☐ Austragen

Antrag I:

Antrag II:

Antrag II:

Abbildung 7: MockUp: Eintragen zum Telefondienst

## A.9 Entity-Relationship-Model

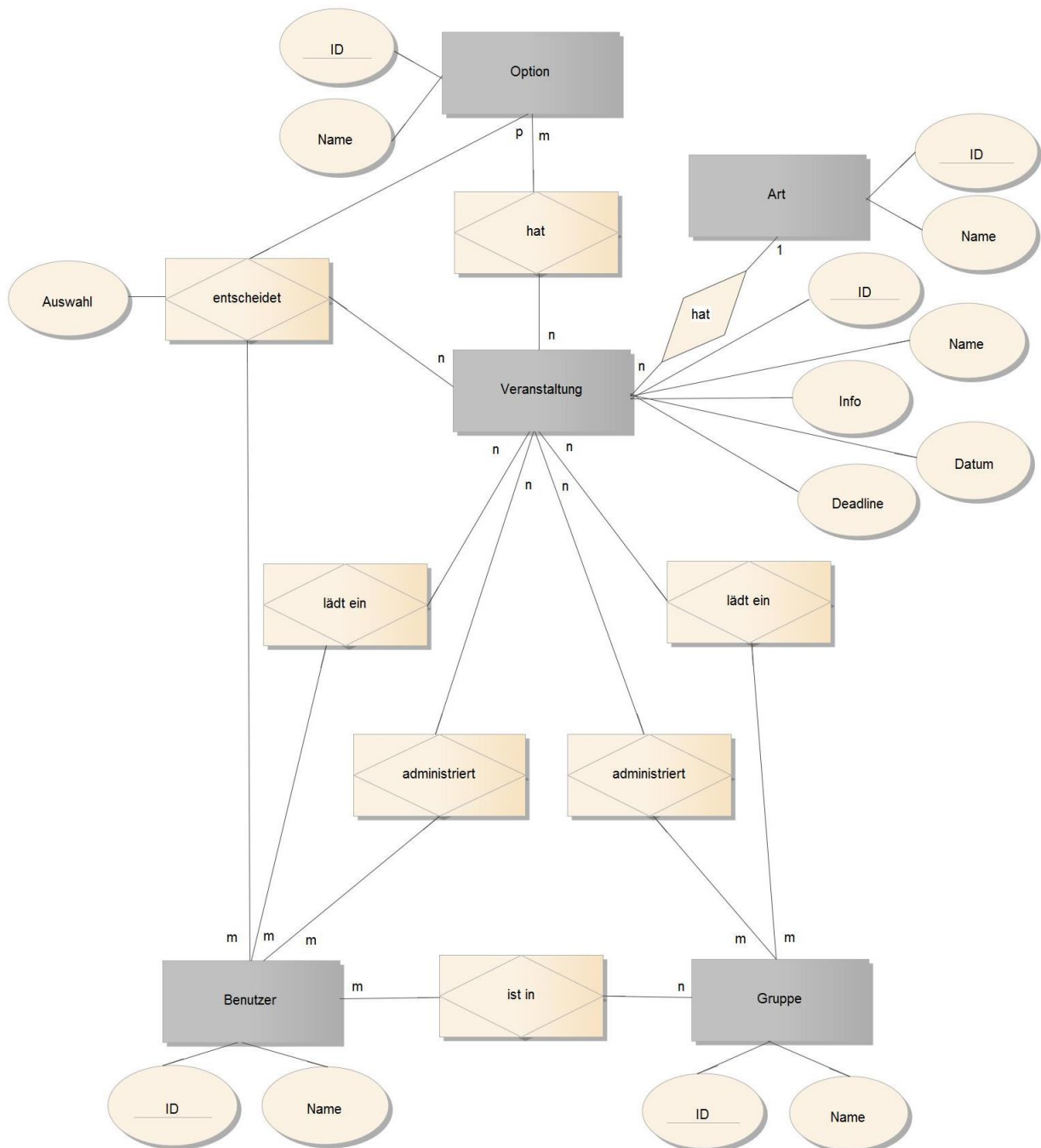


Abbildung 8: Entity-Relationship-Model

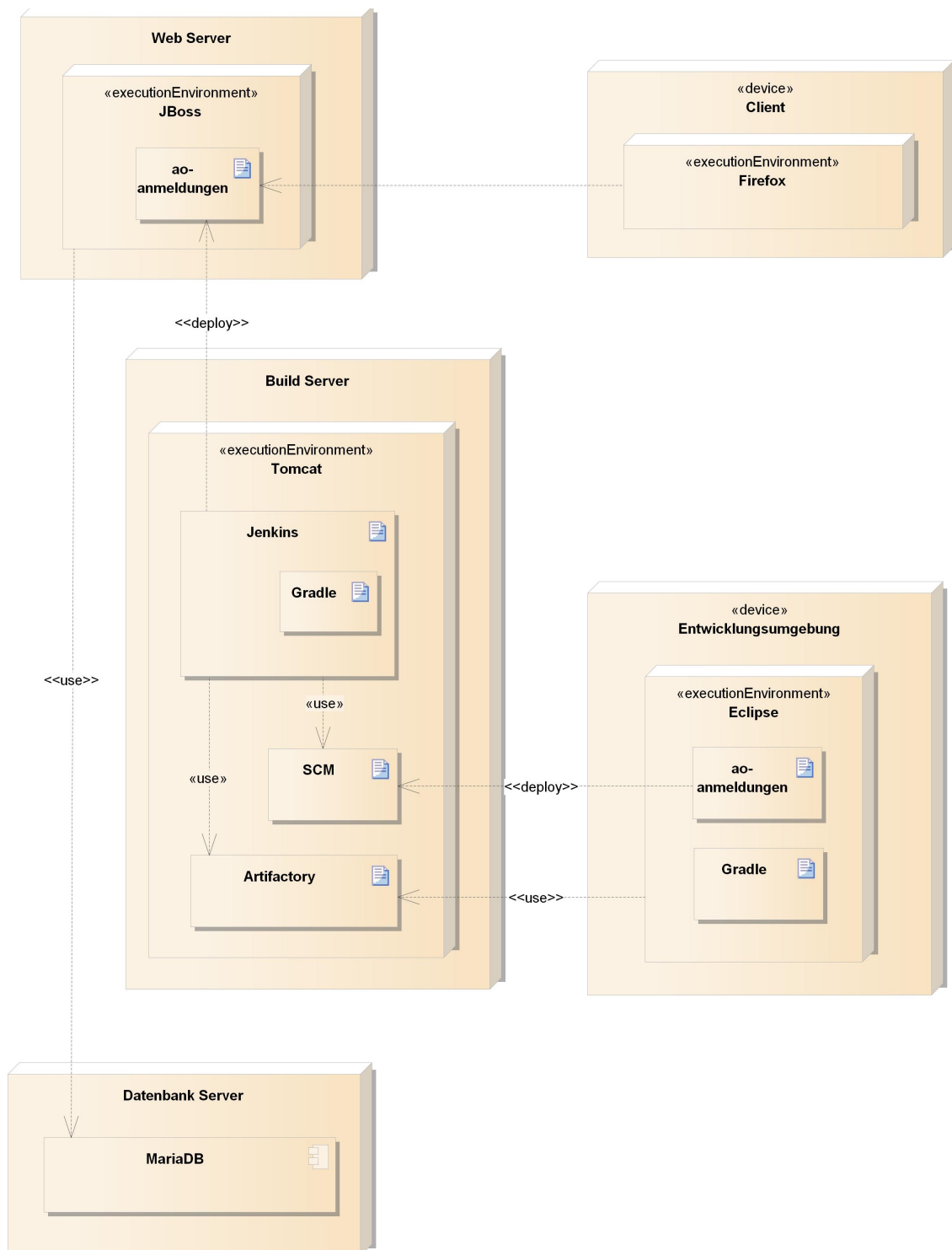
**A.10 Verteilungsdiagramm**

Abbildung 9: Verteilungsdiagramm



## A.11 Gradle Build-Datei

```

1 description = "AOAnmeldungen"
2
3 configurations.create('${project.name}')
4
5 buildscript {
6     repositories {
7         maven { url = "${project.artifactsURL}${project.artifactsRepoKey}".toURL() }
8         ivy { url = "${project.artifactsURL}${project.artifactsRepoKey}".toURL() }
9     }
10    apply from: "${project.gradleApplyFromPath}/net-aokv-classpath.gradle", to: buildscript
11 }
12 logger.debug('buildscript Einstellungen wurden aktiviert')
13
14 [
15     "maven",
16     "project-report",
17     "java ",
18     "eclipse ",
19     "integrationtests ",
20     "ee",
21     "junit4 ",
22     "mockito",
23     "checkstyle ",
24     "findbugs",
25     "pmd",
26     "jacoco",
27     "tasks ",
28     "application ",
29     "cargo ",
30     "jdepend"
31 ].each { apply from: "${project.gradleApplyFromPath}/net-aokv-${it}.gradle" }
32
33 repositories.each { logger.info("Repository ${it.name}") }
34
35 dependencies {
36     compile 'xml-apis:xml-apis:1.4.01',
37             'org.slf4j : slf4j-api:1.7.21',
38             'org.slf4j : slf4j-simple:1.7.21',
39             'org.slf4j : jul-to-slf4j:1.7.21',
40             'org.primefaces:primefaces:5.3',
41             'org.omnifaces:omnifaces:2.2',
42             'com.google.guava:guava:19.0',
43             'javax:javaee-api:7.0',
44             'org.eclipse.persistence:eclipselink:2.6.2'
45 }

```

Listing 1: Gradle Build-Datei

**A.12 JBoss-Konfigurationsdatei**

```
1 batch
2
3 # MariaDB-Treiber hinzufügen
4 module add
5     --name=org.mariadb.jdbc
6     --resources=S:\\\\Entwicklung\\\\JBoss\\\\mariadb-java-client-1.4.0.jar
7     --dependencies=javax.api/subsystem=datasources/jdbc-driver=mariadb:add
8     (driver-name=mariadb,
9      driver-module-name=org.mariadb.jdbc,
10     driver-class-name=org.mariadb.jdbc.Driver)
11
12 # DataSource anlegen
13 data-source add
14     --name=AnmeldungenDS
15     --driver-name=mariadb
16     --driver-class=org.mariadb.jdbc.Driver
17     --connection-url="jdbc:mariadb://ao-ws:3306/anmeldungen"
18     --jndi-name=java:jboss/jdbc/AnmeldungenDS
19     --user-name="username"
20     --password="password"
21
22 run-batch
```

Listing 2: JBoss-Konfigurationsdatei

### A.13 Pflichtenheft (Auszug)

Im folgenden Auszug aus dem Pflichtenheft werden die Anforderungen an die Anwendung festgehalten.

#### Plattform

- Die Webanwendung muss in Java programmiert werden.
- Die Webanwendung muss die Technologien des [Java EE 7](#)-Standards verwenden.
- Für das Objekt-Relationales-Mapping ([ORM](#)) muss [JPA](#) genutzt werden.
- Die Erzeugung der Datenbank-Tabellen soll durch [JPA](#) erfolgen.
- Für das Gestalten der Oberfläche muss [JSF](#) genutzt werden.
- Es muss der Applicationserver *JBoss* verwendet werden.
- Die Daten der Webanwendung müssen in einer *MariaDB*-Datenbank auf dem Webserver [AO-WS](#) der [AO](#) abgelegt sein.
- Es muss ein Repository im [SCM](#)-Manager erstellt werden, das den Namen *ao-anmeldungen* hat.
- Die Webanwendung muss innerhalb des [AO](#)-Netzwerkes unter *ao-anmeldungen* erreichbar sein.

#### Entwicklungsprozess

- Der Sourcecode muss mit *Git* versioniert werden.
- Der Programmcode der Webanwendung muss mit *JUnit*-Tests testbar sein.
- Der Sourcecode muss mit *Gradle* baubar sein.
- Es muss *Jenkins* als Continuous-Integration-Tool genutzt werden. Der Sourcecode muss hier baubar sein.

#### Oberfläche

- Die Webanwendung muss im Corporate Design der [AO](#) gestaltet sein.
- Die Webanwendung muss auf dem Webbrowser *Mozilla Firefox* in der Version 38.2 laufen.
- Die Oberfläche muss responsiv gestaltet werden.
- Die Oberfläche muss mit [HTML5](#) und [CSS3](#) gestaltet werden.

## **A.14 Iterationsplan**

- Aufsetzen der nötigen Systeme (Applicationserver usw.)
- Implementieren der gemeinsam genutzten Entities
- Implementieren der Veranstaltungsanmeldung inkl. Tests
- Implementieren des Messageplans inkl. Tests
- Implementieren des Telefondienstes inkl. Tests
- Abnahme
- Deployment
- Erstellung der Dokumentation

## A.15 CheckStyle im Jenkins

### CheckStyle Ergebnis

#### Vergleich mit letzter Analyse

Alle Warnungen	Neue Warnungen	Behobene Warnungen
37	<u>1</u>	0

#### Zusammenfassung

Gesamt	Hohe Priorität	Normale Priorität	Niedrige Priorität
37	0	<u>37</u>	0

#### Details





Packages	Dateien	Kategorien	Typen	Warnungen	Details	Neu
Package		Gesamt		Verteilung		
<a href="#">net.aokv.anmeldungen.controller</a>		6				
<a href="#">net.aokv.anmeldungen.domain</a>		16				
<a href="#">net.aokv.anmeldungen.repository</a>		3				
<a href="#">net.aokv.anmeldungen.service</a>		8				
<a href="#">net.aokv.anmeldungen.viewmodel</a>		2				
<a href="#">net.aokv.view</a>		2				
Gesamt		37				

Abbildung 10: CheckStyle-Warnungen im Jenkins

## A.16 Entity mit JPA-Anotationen

```
1 package net.aokv.anmeldungen.domain;
2
3 import javax.persistence.Entity;
4 import javax.persistence.Id;
5 import javax.persistence.GeneratedValue;
6 import javax.persistence.OneToMany;
7
8 @Entity
9 public class Benutzer
10 {
11     @Id
12     @GeneratedValue
13     private Long id;
14
15     private String name;
16
17     @OneToMany(mappedBy = "benutzer")
18     private Set<BenutzerGruppe> benutzerGruppen;
19
20     @OneToMany(mappedBy = "benutzer")
21     private Set<EingeladenerBenutzer> eingeladeneBenutzer;
22
23     @OneToMany(mappedBy = "benutzer")
24     private Set<Benutzeradministrator> benutzeradministratoren;
25
26     @OneToMany(mappedBy = "benutzer")
27     private Set<Teilnahme> teilnahmen;
28
29     public Benutzer(final String name)
30     {
31         this.name = name;
32     }
33
34     public Benutzer()
35     {}
36
37     public List<Teilnahme> getTeilnahmen()
38     {
39         return new ArrayList<Teilnahme>(this.teilnahmen);
40     }
41
42     public Long getId()
43     {
44         return id;
45     }
46 }
```

Listing 3: Entity mit JPA-Anotationen

## A.17 Businesslogik im MessageService

```

1 package net.aokv.anmeldungen.service;
2
3 import java.time.temporal.TemporalAdjusters;
4
5 import javax.inject.Inject;
6
7 /**
8  * Stellt die Use-Cases für den Messageplan bereit.
9  */
10 public class MessageService
11 {
12     @Inject
13     private TeilnahmeRepository teilnahmeRepository;
14
15     /**
16      * Konstruktor für Dependency Injection.
17      */
18     public MessageService()
19     {}
20
21     /**
22      * Konstruktor für Tests, da Dependency Injection außerhalb des Applicationsservers nicht
23      * greift.
24      *
25      * @param teilnahmeRepository Mock-Repository für Tests.
26      */
27     MessageService(TeilnahmeRepository teilnahmeRepository)
28     {
29         this.teilnahmeRepository = teilnahmeRepository;
30     }
31
32     /**
33      * Prüft, ob sich der Benutzer im Monat des angegebenen Datums zur Massage anmelden darf.
34      *
35      * @param benutzerID Die ID des Benutzers.
36      * @param massagetermin Der Termin der Massage, für die der Benutzer sich anmelden möchte.
37      * @return Ob sich der Benutzer anmelden darf.
38      */
39     public boolean benutzerKannSichZurMassageAnmelden(Long benutzerID, LocalDate massagetermin)
40     {
41         return teilnahmeRepository.getVeranstaltungenVonBenutzer(
42             benutzerID,
43             getErstenTagDesMonats(massagetermin),
44             getLetzenTagDesMonats(massagetermin))
45             .stream()
46             .filter(v -> v.getArt() == Veranstaltungsart.Message)
47             .count() == 0;
48     }
49

```

```
50 private LocalDate getErstenTagDesMonats(LocalDate tag)
51 {
52     return tag.with(TemporalAdjusters.firstDayOfMonth());
53 }
54
55 private LocalDate getLetzenTagDesMonats(LocalDate tag)
56 {
57     return tag.with(TemporalAdjusters.lastDayOfMonth());
58 }
59 }
```

Listing 4: Businesslogik im MessageService



## A.18 Test des MessageService

```

1 import static org.hamcrest.CoreMatchers.is;
2 import static org.junit.Assert.assertThat;
3 import static org.mockito.Mockito.mock;
4 import static org.mockito.Mockito.verify;
5 import static org.mockito.Mockito.when;
6
7 import org.junit.Before;
8 import org.junit.Test;
9
10 public class MessageServiceSollte
11 {
12     private static final long BENUTZER_ID = 1L;
13     private static final LocalDate ERSTER_TAG_DES_MONATS = LocalDate.of(2016, 12, 1);
14     private static final LocalDate TAG_IM_MONAT = LocalDate.of(2016, 12, 15);
15     private static final LocalDate LETZTER_TAG_DES_MONATS = LocalDate.of(2016, 12, 31);
16
17     private MessageService sut;
18     private List<Veranstaltung> veranstaltungen;
19     private TeilnahmeRepository teilnahmeRepository;
20
21     @Before
22     public void setUp()
23     {
24         veranstaltungen = new ArrayList<>();
25         teilnahmeRepository = mock(TeilnahmeRepository.class);
26         sut = new MessageService(teilnahmeRepository);
27     }
28
29     private void angenommenBenutzerIstFuerVeranstaltungenAngemeldet(final List<Veranstaltung>
        veranstaltungen)
30     {
31         // Simuliert den Rückgabewert.
32         when(teilnahmeRepository.getVeranstaltungenVonBenutzer(
33             BENUTZER_ID, ERSTER_TAG_DES_MONATS, LETZTER_TAG_DES_MONATS))
34             .thenReturn(veranstaltungen);
35     }
36
37     private void erwarteDassBenutzerSichAnmeldenDarf(boolean erwartet)
38     {
39         assertThat(sut.benutzerKannSichZurMessageAnmelden(
40             BENUTZER_ID, TAG_IM_MONAT),
41             is(erwartet));
42     }
43
44     private void stelleSicherDassKorrektZeitraumVerwendetWurde()
45     {
46         // Prüft, ob die angegebene Methode korrekt am Mock aufgerufen wurde.
47         verify(teilnahmeRepository).getVeranstaltungenVonBenutzer(
48             BENUTZER_ID, ERSTER_TAG_DES_MONATS, LETZTER_TAG_DES_MONATS);

```

```
49  }
50
51  @Test
52  public void benutzerAnmeldungErlaubenWennErNochNichtAngemeldetIst()
53  {
54      angenommenBenutzerIstFuerVeranstaltungenAngemeldet(Collections.emptyList());
55      erwarteDassBenutzerSichAnmeldenDarf(true);
56      stelleSicherDassKorrekterZeitraumVerwendetWurde();
57  }
58
59  @Test
60  public void benutzerAnmeldungNichtErlaubenWennErBereitsZuEinerMessageAngemeldetIst()
61  {
62      veranstaltungen.add(new Message());
63      angenommenBenutzerIstFuerVeranstaltungenAngemeldet(veranstaltungen);
64      erwarteDassBenutzerSichAnmeldenDarf(false);
65      stelleSicherDassKorrekterZeitraumVerwendetWurde();
66  }
67
68  @Test
69  public void benutzerAnmeldungErlaubenWennErNochNichtZuEinerMessageAngemeldetIst()
70  {
71      veranstaltungen.add(new Telefondienst());
72      angenommenBenutzerIstFuerVeranstaltungenAngemeldet(veranstaltungen);
73      erwarteDassBenutzerSichAnmeldenDarf(true);
74      stelleSicherDassKorrekterZeitraumVerwendetWurde();
75  }
76 }
```

Listing 5: Test des MessageService

**A.19 View mit JSF-Technologie**

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml"
5   xmlns:h="http://java.sun.com/jsf/html"
6   xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
7   xmlns:c="http://java.sun.com/jsp/jstl/core">
8 <h:head>
9   <title>Veranstaltungsanmeldung</title>
10 </h:head>
11 <h:body>
12   <div id="wrapper">
13     <header>
14       <ui:include src="WEB-INF/includes/header.xhtml" />
15     </header>
16     <div id="breadcrumb">
17       </img>
18       <p>Auswahl Anmeldung</p>
19       </img>
20       <p>Auswahl Message</p>
21     </div>
22     <h:form id="form">
23       <c:forEach end="#{veranstaltungController.getMassagen().size() - 1}" var="i">
24         <div>
25           <h:commandButton
26             value="#{veranstaltungController.getMassagen().get(i).getDatumTag()}"
27             action="#{messageController.geheZuMessageanmeldung()}"
28             class="button">
29             <f:actionListener binding="#{auswahlService.setAusgewaehlteVeranstaltung
30               (veranstaltungController.getMassagen().get(i).getId())}" />
31           </h:commandButton>
32         </div>
33       </c:forEach>
34     </h:form>
35     <footer>
36       <ui:include src="WEB-INF/includes/footer.xhtml" />
37     </footer>

```

Listing 6: View mit JSF-Technologie

## A.20 Screenshots AO-Anmeldungen

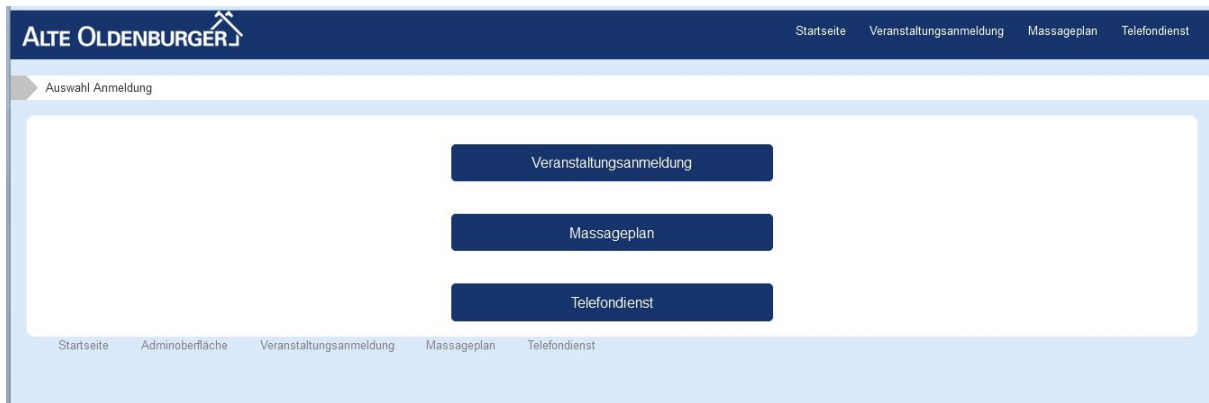


Abbildung 11: Screenshot: Startseite AO-Anmeldungen

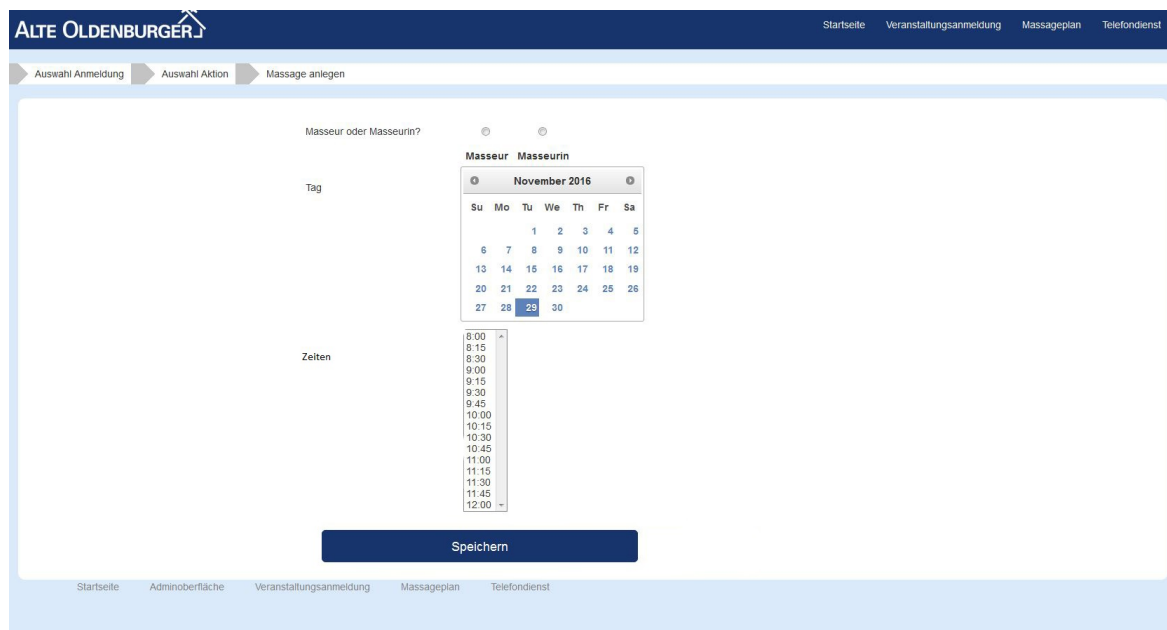


Abbildung 12: Screenshot: Massage anlegen

## A.21 Benutzerdokumentation

Im folgenden Auszug aus der Benutzerdokumentation wird das Vorgehen zur Anlegung einer Veranstaltung als Administrator erklärt.

### Aufrufen von AO-Anmeldungen als Administrator:

AO-Anmeldungen kann mit Eingabe *ao-anmeldungen* oder mit dem Aufruf des Links im Mitarbeiterportal unter **AO > Mitarbeiterbereich > Anmeldungen** aufgerufen werden.

Nun kann in der untersten Leiste die Administratoren-Sicht gewählt werden, sofern Sie ein Administrator sind. Sind Sie kein Administrator, steht Ihnen diese Auswahl nicht zur Verfügung. Im nächsten Schritt können Sie unter **Veranstaltungsanmeldung > Veranstaltung anlegen** mit der Veranstaltungs-Erstellung beginnen. Es werden im Folgenden die verschiedenen Eingabefelder erläutert.

### Erläuterung der Eingabefelder:

Tabelle 5: Erläuterung der Eingabefelder

Feld	Feldfunktion	Einschränkungen
Veranstaltung	Eingabe des Veranstaltungsnamens	Keine.
Info	Eingabe zusätzlicher Informationen wie z.B. der Ort der Veranstaltung	Keine.
Abstimmungsart	Entscheidung, ob der Benutzer sich nur für eine der Auswahlmöglichkeiten (einfache Auswahl) oder für mehrere der Auswahlmöglichkeiten (mehrfache Auswahl) entscheiden darf	Es kann nur entweder die einfache oder die mehrfache Auswahl ausgewählt werden.
Auswahlmöglichkeit	In die bereits vorhanden Eingabefelder <b>Auswahlmöglichkeit 1</b> und <b>Auswahlmöglichkeit 2</b> erfolgt die Eingabe der ersten zwei Auswahlmöglichkeiten. Mit dem Button <b>+ Auswahlmöglichkeit</b> können weitere Auswahlmöglichkeiten hinzugefügt werden.	Es müssen mindestens <b>Auswahlmöglichkeit 1</b> und <b>Auswahlmöglichkeit 2</b> eingegeben werden.
Eingeladene	Auswahl der einzuladenden Benutzern und Gruppen für die Veranstaltung	Keine.
Administratoren	Auswahl der Administratoren und Administratorengruppen für die Veranstaltung	Keine.
Datum	Auswahl des Tages, an dem die Veranstaltung statt findet.	Keine.
Deadline	Auswahl des Tages, bis zu dem die Benutzer abstimmen dürfen.	Keine.

Haben Sie die Eingabe der Daten erfolgreich abgeschlossen, können Sie mit dem Button **Speichern** die Veranstaltung speichern. Sie ist nun bis zur Deadline für die eingeladenen Benutzer und die Administratoren sichtbar

## A.22 Entwicklerdokumentation

All Classes

Packages

net.aokv.anmeldungen.controller

net.aokv.anmeldungen.domain

net.aokv.anmeldungen.repository

net.aokv.anmeldungen.service

net.aokv.anmeldungen.viewmodel

All Classes

AnlegungController

AnlegungEingabedaten

AuswahlService

Basisentitaet

Benutzer

Benutzeradministrator

BenutzeradministratorController

BenutzeradministratorPK

BenutzeradministratorRepository

BenutzerController

BenutzerGruppe

BenutzerGruppePK

BenutzerGruppeRepository

BenutzerRepository

CalendarView

EingeladeneGruppe

EingeladeneGruppenRepository

EingeladeneGruppePK

EingeladenerBenutzer

EingeladenerBenutzerController

EingeladenerBenutzerPK

EingeladenerBenutzerRepository

Gruppe

GruppeController

Gruppenadministrator

GruppenadministratorController

GruppenadministratorenRepository

GruppenadministratorPK

GruppeRepository

LocalDateKonvertierer

Message

MessageanlegungController

MessageanlegungEingabedaten

MessageController

OVERVIEW

PACKAGE

CLASS

TREE

DEPRECATED

INDEX

HELP

PREV CLASS

NEXT CLASS

FRAMES

NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

net.aokv.anmeldungen.domain

Class Veranstaltung

java.lang.Object

net.aokv.anmeldungen.domain.Veranstaltung

Direct Known Subclasses:

Message

Telefondienst

@Entity

public class Veranstaltung

extends java.lang.Object

Constructor Summary

Constructors

Constructor and Description

Veranstaltung()

Veranstaltung(Veranstaltungsart art, java.lang.String name, java.lang.String info, java.time.LocalDate tag, java.time.LocalDate deadline)

Konstruktor erstellt eine neue Veranstaltung mit den Parametern, die der Aufrufer reingeben darf.

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type

Method and Description

Veranstaltungsart

getArt()

Gibt Veranstaltungsart nach dem Enum (Veranstaltung, Message, Telefondienst) zurück.

java.util.List<Benutzer>

getBenutzerAdministratoren()

Gibt alle Benutzer zurück, die für diese

Abbildung 13: Entwicklerdokumentation

## A.23 Klassendiagramm

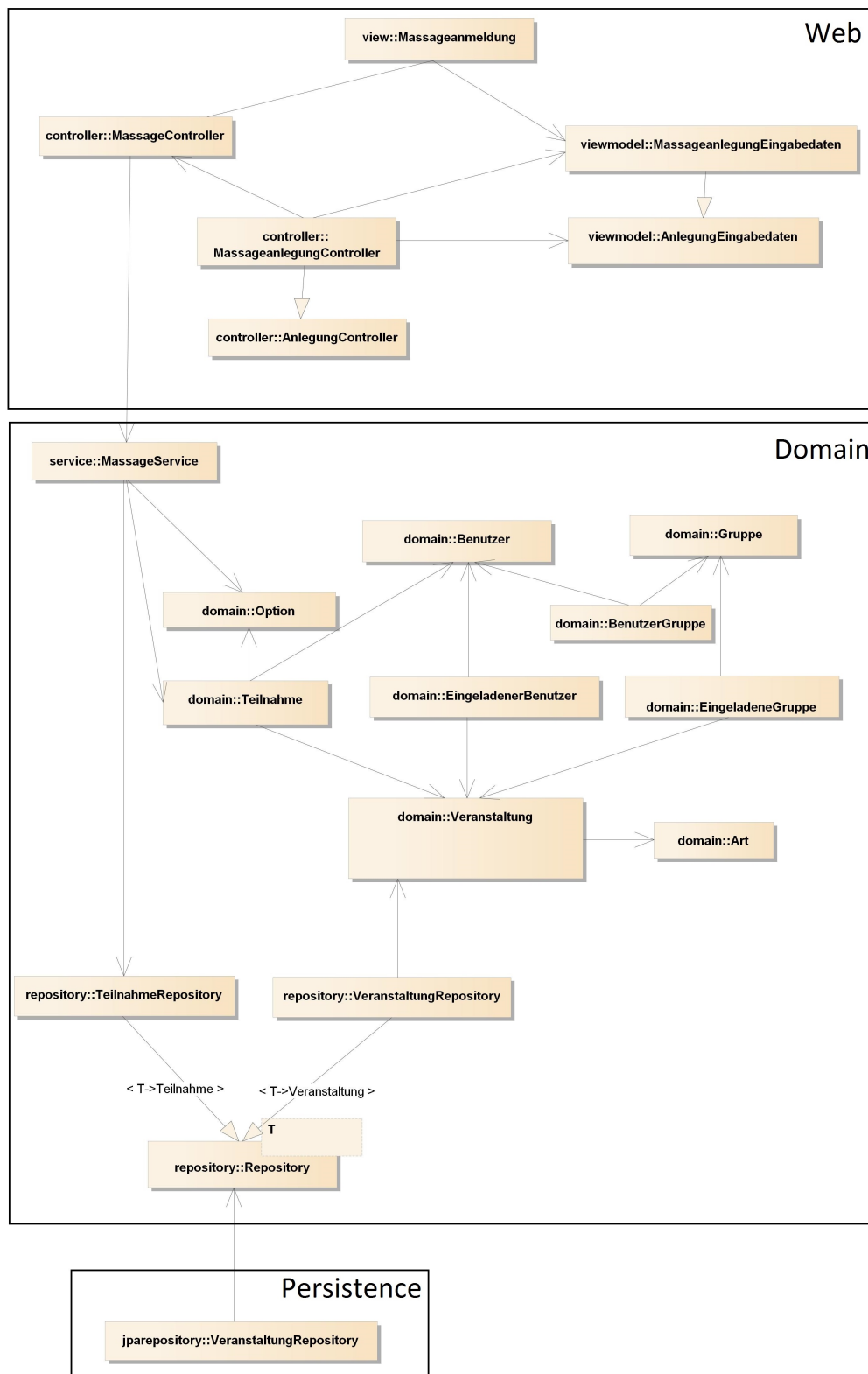


Abbildung 14: Klassendiagramm