Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

*Diplomarbeit*

# Erweiterung des JavaServer Faces Frameworks für den Einsatz in mobilen Anwendungen

vorgelegt von
***Marco Bresch***
am 8. Oktober 2004

Studiengang Softwaretechnik

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. rer. nat. Christoph Klauck

**Fachbereich Elektrotechnik und Informatik**
**Department of Electrical Engineering and Computer Science**

Marco Bresch

### Thema der Diplomarbeit

Erweiterung des JavaServer Faces Framework für den Einsatz in mobilen Anwendungen

### Stichworte

JavaServer Faces, JSF, PDA, Web-Anwendungen, mobile Endgeräte

### Kurzzusammenfassung

Mit JavaServer Faces (JSF) wurde ein offizieller Standard zur Erstellung von dynamischen Webanwendungen bereitgestellt. Durch die Erweiterung des JavaServer Faces Frameworks soll in dieser Diplomarbeit eine für mobile Geräte optimierte Anzeige von Daten erreicht werden. Nach einem Überblick von JavaServer Faces werden dazu eigene Komponenten entwickelt. Anschließend findet eine Bewertung von JavaServer Faces statt.

Marco Bresch

### Title of paper

Extension of JavaServer Faces Framework for mobile applications

### Keywords

JavaServer Faces, JSF, PDA, web application, mobile devices

### Abstract

JavaServer Faces (JSF) presents an official standard to develop dynamic web applications. Extending the JavaServer Faces framework to achieve optimized mobile devices data presentation is the aim of this thesis. An overview of JavaServer Faces will be given first before particular components will be developed. The paper is completed by an evaluation of JavaServer Faces.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Listings

# 1 Einleitung

## 1.1 Motivation

Die Angebote im Internet werden immer vielfältiger. Zu nahezu jedem Thema finden sich im Internet Informationen. Neben den Informationen verbreitet sich auch das Angebot von Dienstleistungen. So kann ein Internetnutzer beispielsweise Bestellungen aufgeben oder Reisen buchen. Aber auch personalisierte Dienste werden angeboten, wie persönliche Adressbücher oder Terminkalender die von jedem Internetzugang erreichbar sind.

Des Weiteren haben mobile Endgeräte, wie Personal Digital Assistent (PDAs), eine große Verbreitung gefunden. Man kann sie überall hin mitnehmen und benutzen. Da die PDAs heutzutage zunehmend über Wireless LAN und Bluetooth verfügen wird auch der Bedarf an mobilen Anwendungen größer die über das Internet erreichbar sind. So erfreuen sich beispielsweise so genannte „mobile Ausgaben“ von Nachrichtenmagazinen immer größerer Beliebtheit. PDAs verfügen aufgrund ihrer kompakten Bauweise über eine sehr kleine Anzeige, so dass eine separate und speziell auf die Geräte angepasste Darstellung der Dienstleistung gewählt werden sollte.

Der Markt bietet für die Bereitstellung der Dienste verschiedenste Technologien an. Diese Arbeit beschäftigt sich mit einer Lösung aus dem Java Umfeld. Seit Anfang des Jahres 2004 gibt es aus diesem Bereich eine neue Spezifikation für Webanwendungen: JavaServer Faces. JavaServer Faces (JSF) ist ein User Interface Framework, mit dessen Hilfe sich Webanwendungen erstellen lassen. JSF ist Gegenstand dieser Diplomarbeit und soll anhand einer prototypischen Implementierung erprobt werden. Ziel hierbei ist die Erweiterung des JavaServer Faces Frameworks um eine kompakte Darstellung für mobile Endgeräte zu ermöglichen.

## 1.2 Gliederung der Diplomarbeit

In Kapitel 2 werden die zugrunde liegenden Beispielszenarios erläutert. Die Szenarios werden mit Hilfe von Anwendungsfällen konkretisiert. Abschließend werden in Kapitel 2 allgemeine Anforderungen an das System formuliert. Neben grundlegenden Entwurfsentscheidungen wird in Kapitel 3 auf Basis der Analyse das Fachkonzept modelliert. Nach der Betrachtung des Software-Entwurfs wird die Einbettung des Fachkonzepts unter zuhilfenahme von etablierten Entwurfsmustern in die Gesamt-

architektur diskutiert. Beleuchtet werden die Möglichkeiten die der Java-Standard im Umfeld von Webanwendungen bereitstellt. Nach der Entscheidung für JavaServer Faces wird in Kapitel 4 die Technologie vertieft und soweit beschrieben, wie sie zum Verständnis der Arbeit notwendig ist. Auf Grundlage der Beschreibung von JavaServer Faces wird im Kapitel 5 die prototypische Implementierung skizziert. Abschließend wird in Kapitel 6 die Arbeit bewertet und ein Ausblick auf mögliche Weiterentwicklung von JavaServer Faces gegeben.

# 2 Analyse

## 2.1 Ferienclub

Der hier behandelte fiktive Ferienclub verfügt über ein vielfältiges Freizeitangebot. Für jeden Gast soll es Möglichkeiten geben die für ihn passenden Aktivitäten zu finden. Ein breites Angebot für Paare und Familien sowie für Singles bzw. Alleinreisende aus den Bereichen Sport, Kultur oder Wellness steht dafür zur Verfügung.

Damit der Gast sich einen Überblick verschaffen kann, gibt es einen Veranstaltungskalender. In diesem sind sämtliche Angebote verzeichnet. Jeder Gast bekommt seinen persönlichen Kalender zur Verfügung gestellt, der zur Planung seiner eigenen Termine dienen soll.

## 2.2 Systemidee

In dem oben skizzierten Ferienclub soll eine Terminverwaltung durch ein Informationssystem unterstützt werden. Hierbei soll sowohl ein allgemeiner Veranstaltungskalender, als auch ein persönlicher Kalender für Gäste und Mitarbeiter zur Verfügung gestellt werden.

Die neu zu implementierende Terminverwaltung soll dabei sämtliche Funktionalitäten bereitstellen, die zur Pflege von öffentlichen und privaten Terminen notwendig sind, wie zum Beispiel das Anlegen oder Ändern von Terminen. Jeder Gast bekommt einen persönlichen Kalender, den er jederzeit einsehen und bearbeiten kann. Darüber hinaus können dem Gast gemäß seines eingestellten Profils, Terminvorschläge unterbreitet werden.

## 2.3 Beispielszenario

Anhand der vorhergehenden Abschnitte 2.1 und 2.2 soll in einem Beispielszenario der Ablauf skizziert werden, den ein Gast in dem hier behandelten fiktiven Ferienclub durchläuft. Dabei werden nicht alle erdenklichen Fälle behandelt, die im Zusammenhang mit einem Ferienclub stehen. Vielmehr werden die Bereiche beleuchtet, die durch die Systemidee (siehe 2.2 auf Seite 13 sowie [Lüpke 2004, Seite 10ff]) beschrieben wurden.

### 2.3.1 Ankunft eines Gastes

Nachdem der Gast seine Reise und somit den Aufenthalt in dem Ferienclub gebucht hat, werden schon erste Vorbereitungen getroffen. Dem Gast wird ein Zimmer zugeteilt sowie ein eigener Zugang zum Informationssystem eingerichtet, ein so genannter Account. Die dazu gehörigen Zugangsdaten werden dem Gast zugeschickt.

Wenn der Gast im Ferienclub ankommt, begibt er sich zunächst zur Rezeption, wo er einen Schlüssel zu seinem Appartement bekommt. Bei der Schlüsselübergabe wird der persönliche Account für Buchungen von Veranstaltungen durch die Rezeption aktiviert. Des Weiteren wird dem Gast ein mobiles Endgerät übergeben, ein so genannter Personal Digital Assistent (PDA). Der Gast erhält von der Rezeption eine kleine Einweisung und die Daten, mit denen er sich der Gast an dem Gerät anmeldet. Dabei bekommt der Gast einen Hinweis darauf, dass er die Anmeldedaten keinem Dritten bekannt geben darf. Ansonsten könnten kostenpflichtige Angebote in seinem Namen in Anspruch genommen werden.

### 2.3.2 Das eigene Profil

Der Gast hat vor der Anreise die Möglichkeit, sein Profil über das Internet mit Hilfe seiner Zugangsdaten zu spezifizieren. Hat er dies vor der Anreise noch nicht getan, so kann er es auch nach der Anreise im Ferienclub über seinen Personal Digital Assistent (PDA) tun. Dabei gibt er an welche Bereiche ihn interessieren, indem er die Kategorie mit einer Zahl von null bis neun – gar nicht bis sehr interessant – bewertet. Er kann zum Beispiel folgende Themengebiete bewerten:

- Sport
  - Tennis
  - Golf
  - Beachvolleyball
  - Reiten
  - Surfen
  - Tauchen
- Freizeit
  - Tanzen
  - Spielen
  - Motorrad-, Fahrrad- und Schiffstouren
- Kultur

  - Schlossbesichtigung
  - Museum
- Kunst
  - Malkurs
  - Töpfern
- Wellness
  - Ayurveda
  - Thalasso
  - Beauty
  - Massagen
- Kinder
  - Abenteuertag
  - Zirkusschule
  - Schwimmkurs

Nachdem der Gast seine Interessen eingetragen hat, entscheidet er sich für die Zusendung von Veranstaltungstipps.

### 2.3.3 Öffentlicher Kalender

Nach erfolgreicher Anmeldung und Initialisierung des Accounts sieht der Gast sich den öffentlichen Kalender an. Zunächst erhält er die Möglichkeit den Umfang des Kalenders zu bestimmen. So kann er sich zum Beispiel nur die Veranstaltungen ansehen, die seinem Profil entsprechen. Außerdem erhält er die Möglichkeit aus den oben genannten Kategorien auszuwählen. Der Gast entscheidet sich zum Beispiel für den Bereich Freizeit, da er Lust auf eine Radtour hat.

Der Gast erhält nun eine Liste mit verschiedenen Veranstaltungen des ausgewählten Themengebietes. Veranstaltungen, die sich mit seinen eigenen Terminen überschneiden sind markiert. Unter den Angeboten, die teilweise kostenpflichtig sind, findet er eine passende Tour für sich und ruft die Detailansicht auf.


**Mountainbike-Tour durch die Berge von Süd-Tirol**

Diese 30 km lange Mountainbike-Tour führt abseits der Straße durch die Berge. In einer Gruppe von maximal zehn Teilnehmern können Sie auf unseren hochwertigen Mountainbikes die Natur hautnah genießen. Unser Scout zeigt ihnen auf dem Fahrrad die schönsten Aussichtspunkte. Nach der Hälfte der Strecke wartet ein von uns vorbereitetes Picknick auf sie.

| | |
|---|---|
| Termin: | Montag, 11:00-17:00 |
| verfügbare Plätze: | 6 |
| max. Teilnehmerzahl: | 10 |
| min. Teilnehmerzahl: | 7 |
| Schwierigkeitsstufe: | Einsteiger |

Abbildung 2.1: Detailansicht zur Veranstaltung Radtour [Robinson 2004a]

### 2.3.4 Veranstaltung ansehen und buchen

In der Detailansicht findet der Gast ausführliche Informationen zu der ausgesuchten Radtour (siehe Abbildung 2.1 auf Seite 16).

Da es noch genug verfügbare Plätze gibt und der Gast keine parallele Veranstaltung gebucht hat, entscheidet er sich dafür den Kurs zu buchen. Der Termin wird daraufhin in seinen persönlichen Kalender eingetragen. Außerdem wird die Anzahl der verfügbaren Plätze um einen Platz verringert.

### 2.3.5 Veranstaltungstipps

Der Gast möchte sich die Veranstaltungstipps des Tages und der nächsten zwei Tage ansehen. Er meldet sich zunächst an das System an und wählt aus der Übersicht den Eintrag Veranstaltungstipps aus. Gemäß seines Profils werden die Veranstaltungstipps nach Relevanz sortiert angezeigt. Dabei werden die für den Gast interessantesten Angebote zuerst angezeigt, während die weniger attraktiven Veranstaltungen weiter unten in der Liste erscheinen. Des Weiteren kann er sich auch eine chronologisch sortierte Liste anzeigen lassen.

Obwohl die Radtour (siehe 2.3.4 auf Seite 17) seinen Interessen entspricht, wird sie bei den Veranstaltungstipps nicht angezeigt, da der Gast diesen Termin schon gebucht hat. Veranstaltungen, die parallel zu schon gebuchten Aktivitäten verlaufen, werden mit einem Symbol versehen.

Beim Stöbern in den Veranstaltungstipps hat der Gast festgestellt, dass es ein reizvolleres Angebot gibt, als die schon gebuchte Radtour. Er entscheidet sich für die GPS-Tour[1] und ruft die Detailansicht auf, um die Alternative zu buchen.

### 2.3.6 Konkurrierende Veranstaltung ansehen und buchen

In der Detailansicht (siehe Abbildung 2.2 auf Seite 18) erhält der Gast neben der Beschreibung einen Hinweis, dass der Termin sich mit einer gebuchten Veranstaltung überschneidet – in diesem Fall die Radtour. Trotz des Hinweises entscheidet er sich für die ausgewählte Veranstaltung und bucht diese.

Bevor die Buchung durchgeführt wird, erhält der Gast die Nachfrage, ob er die neue Veranstaltung buchen möchte. Er bejaht die Frage und die Radtour wird aus seinem persönlichen Kalender entfernt. Im Gegenzug wird die GPS-Tour[1] eingetragen. Des Weiteren wird die Anzahl der verfügbaren Plätze bei der Radtour korrigiert und um einen wieder hoch gesetzt.

[1] Global Positioning System. Ein Satelliten gestütztes System zur Ortung eines Standortes, welches über eine Genauigkeit von wenigen Metern verfügt.

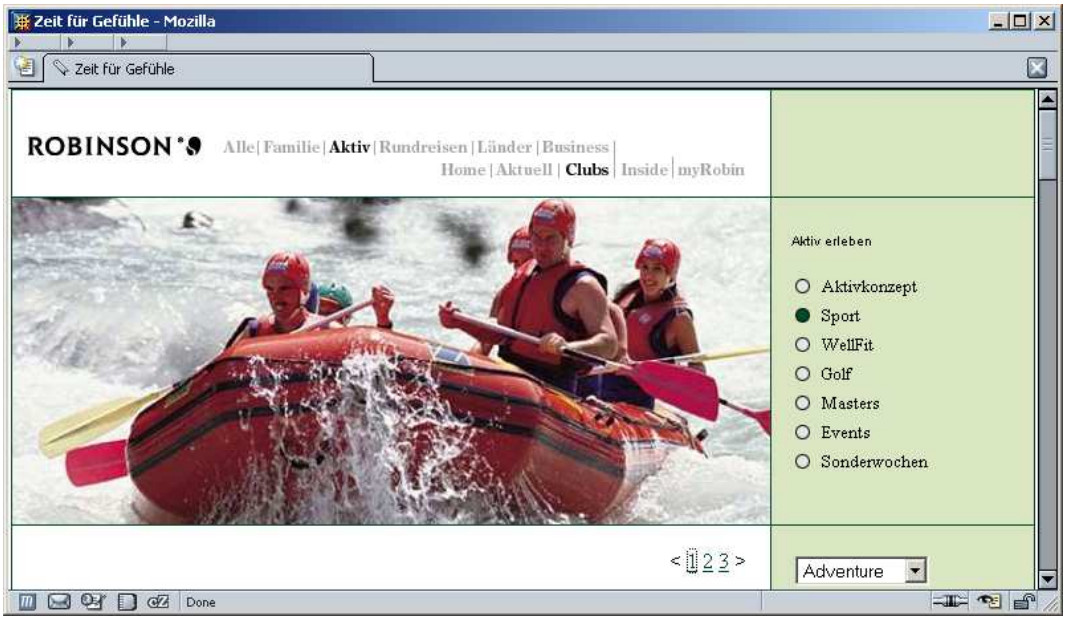

| | |
|---|---|
| Termin: | Montag, 10:00-18:00 |
| verfügbare Plätze: | 13 |
| max. Teilnehmerzahl: | 20 |
| min. Teilnehmerzahl: | ./. |

*Termin überschneidet sich mit der Radtour!*

Abbildung 2.2: Detailansicht zur Veranstaltung GPS-Tour [CenterParcs 2004]

### 2.3.7 Kostenpflichtige Veranstaltung buchen

Beim Stöbern im öffentlichen Kalender ist der Gast auf eine Rafting-Tour gestoßen (siehe Abbildung 2.3 auf Seite 19). Diese Tour ist kostenpflichtig. Der Gast entscheidet sich für die Tour und möchte sie buchen.




**Rafting-Tour**

Adrenalin pur: Im Rafting Team meistern Sie per Schlauchboot die Stromschnellen von Gebirgsflüssen. Neoprenanzug, Schwimmweste und Helm gehören zur vorgeschriebenen Sicherheitsausrüstung und werden vom Veranstalter gestellt.

| | |
|---|---|
| Termin: | Dienstag, 14:00-17:00 |
| verfügbare Plätze: | 5 |
| max. Teilnehmerzahl: | 12 |
| min. Teilnehmerzahl: | 4 |
| Schwierigkeitsstufe: | Fortgeschritten |
| *Kostenpflichtig*: | € 20,- |

Abbildung 2.3: Detailansicht zur Veranstaltung Rafting [Robinson 2004b]

Bevor die Buchung jedoch vollzogen wird, erhält der Gast einen Hinweis über mögliche Stornogebühren. Diese belaufen sich auf 25% der Teilnahmekosten, sofern der Gast die Tour mindestens einen Tag vorher absagt. Bei einer späteren Absage

trägt der Gast die vollen Kosten. Bevor der Gast die Rafting-Tour buchen kann, muss er die Bedingung annehmen, indem er den Hinweis als gelesen markiert. Lehnt er den Hinweis hingegen ab, so kann er die Tour auch nicht buchen.

Der Gast hat die Bedingungen gelesen und akzeptiert, womit er einen rechtsverbindlichen Vertrag eingegangen ist und die Tour gemäß Abschnitt 2.3.4 auf Seite 17 in seinen persönlichen Kalender aufgenommen wird.

### 2.3.8 Eigene Veranstaltung eingeben

Der Gast möchte gerne am Abend Skat spielen, ist jedoch Alleinreisender. Ihm fehlen also Mitspieler. Er meldet sich am System an und gibt eine Beschreibung zu seinem Vorhaben ein. Außerdem spezifiziert er Rahmenbedingungen wie die Anzahl der Teilnehmer, hier drei bis vier, und gibt noch einen Wunschort an.

Nachdem der Gast die Angaben vollständig eingegeben hat speichert er die eigene Veranstaltung. Der Clubmanager erhält eine Nachricht darüber, dass eine neue Veranstaltung vorliegt. Er prüft den Inhalt der Beschreibung und fügt gegebenenfalls weitere Informationen hinzu. Außerdem kategorisiert er die neue Veranstaltung. Abschließend prüft er den gewünschten Ort auf Verfügbarkeit. Sind alle Eingaben korrekt, so gibt der Clubmanager die neue Veranstaltung frei und sie erscheint im öffentlichen Kalender bzw. bei den Veranstaltungstipps der Gäste, die ein Interesse an Gesellschaftsspielen haben.

### 2.3.9 Veranstaltung wird abgesagt

Leider muss die Kanutour aufgrund mangelnden Interesses abgesagt werden. Die Veranstaltung wird aus dem öffentlichen Kalender entfernt. Außerdem wird allen bisher angemeldeten Gästen eine Nachricht zugestellt, die sie bei ihrer nächsten Anmeldung am System sofort erhalten.

Nachdem die betroffenen Gäste die Nachricht gelesen haben, bekommen sie speziell für diesen Termin Veranstaltungstipps präsentiert. Diese werden bei einem spezifizierten Profil nach Relevanz sortiert. Hat der Gast kein Profil angegeben, so werden die Veranstaltungstipps nach Kategorien sortiert dargestellt. Jeder betroffene Gast erhält dadurch die Möglichkeit, sich für den nun freien Zeitraum eine Alternative zu suchen.

### 2.3.10 Abreise eines Gastes

Bei der Abreise hat der Gast seinen Zimmerschlüssel und seinen PDA an der Rezeption abzugeben. Der Account wird von der Rezeption deaktiviert. Nach der Abreise wird der PDA vom technischen Administrator in seinen ursprünglichen Zustand versetzt und somit für den nächsten Gast vorbereitet.

## 2.4 Akteure

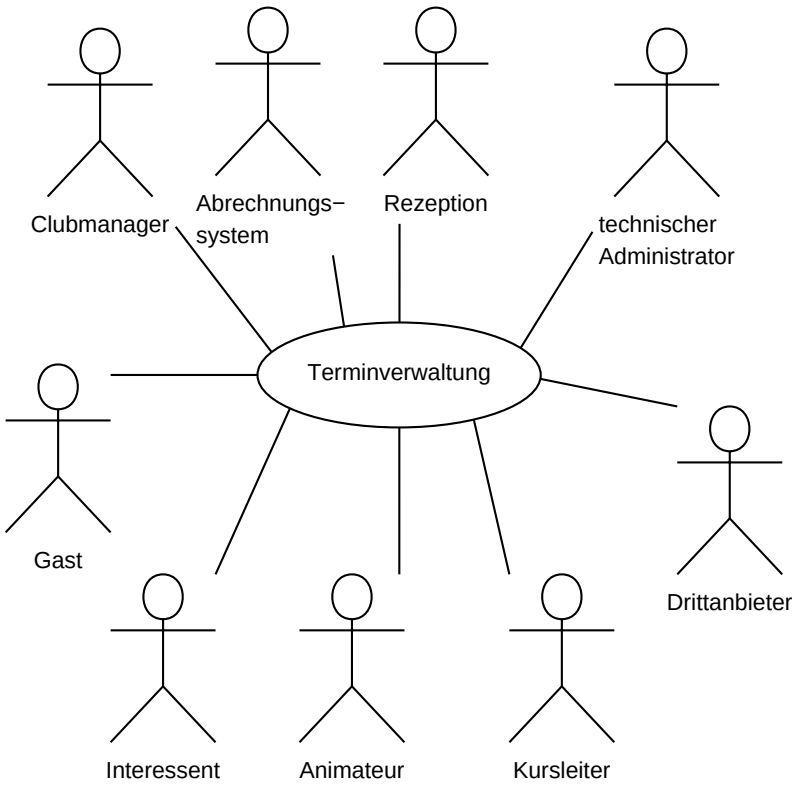



Abbildung 2.4: Akteure der Terminverwaltung

Aus dem oben beschriebenen Beispielszenarios aus Abschnitt 2.3 kristallisieren sich die in Abbildung 2.4 skizzierten Akteure heraus. Dieser Abschnitt fasst die Aufgaben und Zuständigkeiten der einzelnen Rollen zusammen.

### Gast

Der Gast hat während seines Aufenthaltes im Ferienclub Zugang zum Informationssystem. Er kann Termine des öffentlichen Kalenders einsehen und Veranstaltungen aus diesem buchen. Darüber hinaus verfügt der Gast über einen persönlichen Kalender auf den er vollen Zugriff genießt.

Des Weiteren gibt es für den Gast die Möglichkeit eigene Veranstaltungen anzulegen, die jedoch vom Clubmanager freigeschaltet werden müssen.

#### Rezeption und Clubmanager

Die Rezeption kann einen neuen Account für einen Gast einrichten und aktivieren. Sie teilt dem Gast ein Zimmer zu. Außerdem kann die Rezeption den öffentlichen Kalender einsehen, um Informationen über Veranstaltungen geben zu können.

Der Clubmanager verfügt über die gleichen Berechtigungen und kümmert sich darüber hinaus noch um die Gestaltung des öffentlichen Kalenders. Er entscheidet darüber, welche Veranstaltungen im öffentlichen Kalender aufgenommen werden und weist den Veranstaltungen eventuell benötigte Ressourcen zu, wie zum Beispiel Räume oder Tennis Courts.

#### Technischer Administrator

Der technische Administrator ist der Ansprechpartner bei technischen Problemen. Er sorgt dafür, dass das Informationssystem zur Verfügung steht. Des Weiteren kann der Administrator bei Bedarf sämtliche Daten des Informationssystems bearbeiten. Dazu gehört vor allem die Pflege von Stammdaten.

#### Animateur, Kursleiter und Drittanbieter

Die „Betreuer“ der Veranstaltungen verfügen wie der Gast über einen persönlichen Kalender. In diesem Kalender werden ihre Veranstaltungen gelistet. Dabei hält der Kalender auch die Anzahl der angemeldeten Gäste bereit, zur eventuellen Vorbereitung auf die Veranstaltung. Außerdem haben sie auch lesenden Zugang zum öffentlichen Kalender.

#### Abrechnungssystem

Das Abrechnungssystem trägt die entstehenden Kosten des Gastes zusammen, die sich aus Besuchen in der Bar oder im Restaurant ergeben, aber auch aus kostenpflichtigen Veranstaltungen.

## 2.5 Anwendungsfälle

Die Anwendungsfälle sind aus der Sicht des Ferienclubs beschrieben.

### 2.5.1 Gast Ankunft

Nachdem ein Gast eine Reise für den Ferienclub gebucht hat und bevor der Gast im Ferienclub anreist, erhält der Ferienclub die Daten des Gastes. Mit Hilfe dieser Daten wird entweder vom technischen Administrator, der Rezeption oder dem Clubmanager

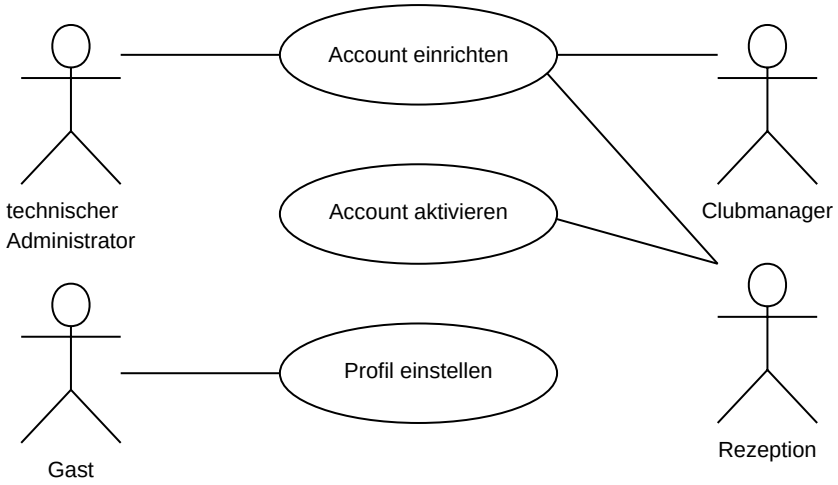



Abbildung 2.5: Anwendungsfalldiagramm Gast Ankunft

ein Account für den Gast eingerichtet. Darüber hinaus wird für den Gast ein Zimmer reserviert.

Bei Ankunft des Gastes werden zunächst seine Daten überprüft, womit der Gast zugleich identifiziert wird. Die Rezeption aktiviert den eingerichteten Account, so dass dieser sofort vom Gast nutzbar ist. Meldet sich der Gast zum ersten Mal am Informationssystem an, so kann er sich zunächst ein Passwort vergeben. Im Anschluss daran kann er sein Profil spezifizieren.

| Name: | **Account einrichten** |
|---|---|
| Kurzbeschreibung: | Für einen zukünftigen Gast wird ein Account eingerichtet. |
| Akteure: | Rezeption, Clubmanager, technischer Administrator |
| Auslöser: | Ein Gast hat eine Reise im Ferienclub gebucht. |
| Vorbedingungen: | ./. |
| Eingehende Informationen: | Daten des Gastes, Zeitraum der Buchung. |
| Nachbedingungen: | Deaktivierter Account für den Gast. |
| Essentieller Ablauf: | 1. Daten des Gastes suchen:<br>Zunächst wird anhand des Namens und der Adresse überprüft, ob der Gast dem System bekannt ist.<br>2. Account einrichten:<br>Ist der Gast bekannt werden die Daten überprüft, andernfalls neu erfasst. |

Tabelle 2.1: Anwendungsfall Account einrichten

| Name: | **Account aktivieren** |
|---|---|
| Kurzbeschreibung: | Bei Ankunft des Gastes wird sein Account aktiviert. |
| Akteure: | Rezeption, Clubmanager |
| Auslöser: | Ein Gast kommt im Ferienclub an. |
| Vorbedingungen: | Eingerichteter Account für den Gast. |
| Eingehende Informationen: | Daten des Gastes |
| Nachbedingungen: | Account ist aktiviert. |
| Essentieller Ablauf: | 1. Gast identifizieren:<br>Der gerade ankommende Gast nennt seinen Namen zur Identifikation. Reicht der Name für eine eindeutige Zuordnung nicht aus, so fragt die Rezeption weitere Daten ab, wie zum Beispiel die Adresse, bis die Zuordnung eindeutig ist.<br>2. Account aktivieren:<br>Die Rezeption oder der Clubmanager aktivieren den für den Gast eingerichteten Account. |

Tabelle 2.2: Anwendungsfall Account aktivieren

| Name: | **Profil einstellen** |
|---|---|
| Kurzbeschreibung: | Der Gast spezifiziert, seinen Interessen entsprechend, sein Profil. |
| Akteure: | Gast |
| Auslöser: | Erste Anmeldung oder der Gast möchte sein Profil ändern. |
| Vorbedingungen: | Gast ist am System angemeldet. |
| Eingehende Informationen: | Bewertungen der Kategorien. |
| Ergebnisse: | Profil ist eingestellt oder geändert. |
| Essentieller Ablauf: | 1. Kategorien bewerten:<br>Dem Gast werden der Reihe nach alle Kategorien zur Bewertung aufgezeigt. Dabei kann der Gast die Oberkategorie oder einzeln deren Unterkategorien bewerten.<br>2. Profil speichern:<br>Das Profil wird mit den vorgenommenen Einstellungen gespeichert. |

Tabelle 2.3: Anwendungsfall Profil einstellen

### 2.5.2 Kalender

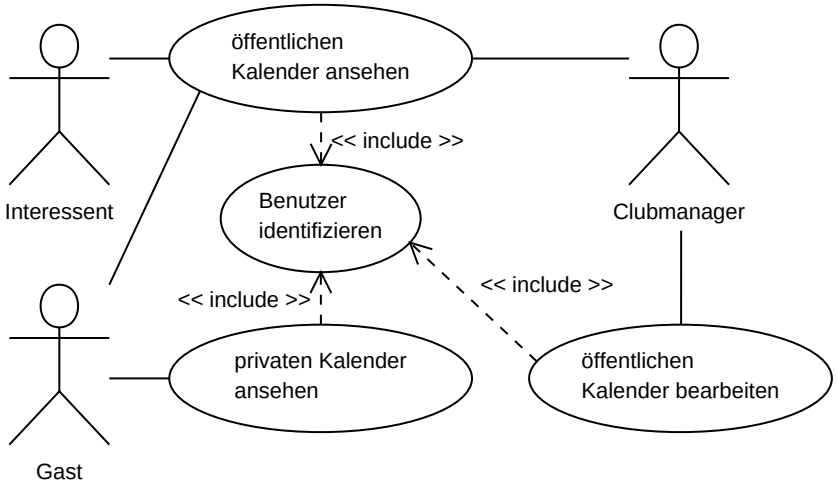



Abbildung 2.6: Anwendungsfalldiagramm Kalender

Bevor der öffentliche Kalender angesehen werden kann, muss der Benutzer sich identifizieren. Ein Interessent wählt den so genannten Gastzugang. Mit Hilfe des Gastzugangs kann der Interessent sich einen Überblick der angebotenen Veranstaltungen verschaffen. Dabei erhält er eine Auswahl von Veranstaltungen die für die Öffentlichkeit freigegeben sind. Nach der Identifikation erhält der Benutzer, seiner Rolle entsprechend, unterschiedliche Auswahlmöglichkeiten. Während der Gast und der Interessent eine Liste von vorhandenen Themengebieten erhält, hat der Clubmanager darüber hinaus die Möglichkeit, den Kalender zu bearbeiten.

Bei dem öffentlichen Kalender können unterschiedliche Filter für den Umfang der Anzeige ausgewählt werden. Der Benutzer kann hier zwischen unterschiedlichen Themengebieten auswählen, so dass die Anzeige der Veranstaltungen auf die entsprechende Kategorie beschränkt wird und somit die Übersichtlichkeit gewahrt wird. Neben der Wahl eines Themengebietes kann er sich Veranstaltungen gemäß seines Profils anzeigen lassen.

Die Veranstaltungen werden in einer Liste angezeigt. Aus der Darstellung gehen Anfangs- und Endzeit hervor. Neben dem Titel der Veranstaltung wird optional eine Kurzbeschreibung oder ein Bild angezeigt. Außerdem werden kostenpflichtige Veranstaltungen und sich mit einem privaten Termin überschneidende Veranstaltungen mit je einem Symbol versehen.

| Name: | **Öffentlichen Kalender ansehen** |
|---|---|
| Kurzbeschreibung: | Ein Benutzer erhält einen Überblick über angebotene Veranstaltungen. |
| Akteure: | Clubmanager, Gast, Interessent |
| Auslöser: | Ein Benutzer möchte sich einen Überblick über die angebotenen Veranstaltungen verschaffen. |
| Vorbedingungen: | Benutzer ist am System angemeldet |
| Eingehende Informationen: | Auswahl des Anzeigeumfangs |
| Ergebnisse: | Benutzer hat einen Überblick erhalten. |
| Essentieller Ablauf: | 1. Umfang bestimmen:<br>Der Benutzer wählt den gewünschten Umfang aus, den der Kalender haben soll. Dabei kann er verschiedene Kategorien oder Veranstaltungen auswählen, die seinem Profil entsprechen. Außerdem kann er den Zeitraum bestimmen.<br>2. Liste anzeigen:<br>Es wird eine Liste von Veranstaltungen angezeigt, die der Auswahl entsprechen.<br>3. Sortierung wählen:<br>Die Veranstaltungen können nach Relevanz, bezüglich des Profils, oder zeitlich sortiert werden. |

Tabelle 2.4: Anwendungsfall öffentlichen Kalender ansehen

### 2.5.3 Veranstaltungen

Der Clubmanager hat unter anderem die Aufgabe sich um den Inhalt des öffentlichen Kalenders zu kümmern. Dazu ist es notwendig neue Veranstaltungen zu erfassen, aber auch gegebenenfalls bestehende Veranstaltungen zu bearbeiten. Sollte eine Veranstaltung nicht über genügend Teilnehmer verfügen, so muss diese beispielsweise abgesagt werden.

Der Gast kann sich über Veranstaltungen informieren indem er sich die Details dazu ansieht. Er kann Veranstaltungen buchen, wobei hier zwischen einer kostenpflichtigen und einer normalen Veranstaltung unterschieden wird. Eine schon gebuchte Veranstaltung kann vom Gast storniert werden. Sollte der Gast eine konkurrierende[2] Veranstaltung buchen, so wird die bisherige Veranstaltung storniert.

Neben den *öffentlichen* Veranstaltungen erhält der Gast die Möglichkeit, auch so genannte *private* Veranstaltungen einzugeben. Als Beispiel wäre hier ein Skatspiel denkbar. So kann der Gast einen gewünschten Zeitraum angeben und die Anzahl der Teilnehmer bestimmen. Der Clubmanager erhält eine Mitteilung über eine neue private Veranstaltung und erweitert sie um weitere Angaben wie Ort und Kategorie.

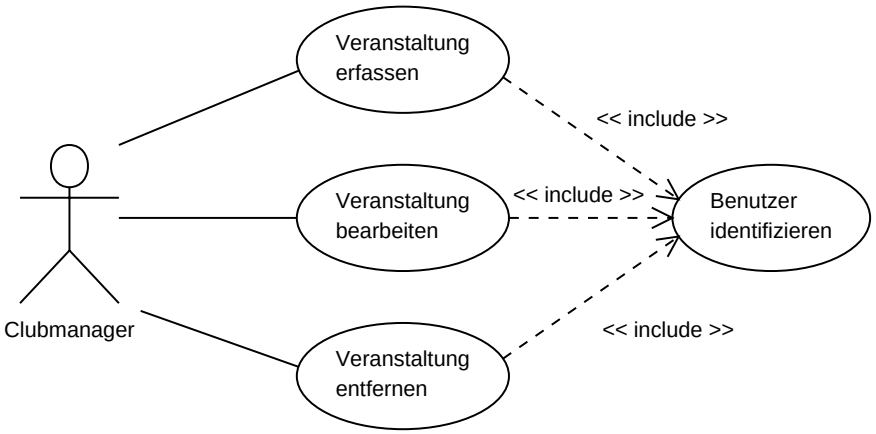



Abbildung 2.7: Anwendungsfalldiagramm Veranstaltung (Clubmanager)

[2]Konkurrierend bedeutet in diesem Falle zwei sich zeitlich überlappende Veranstaltungen.

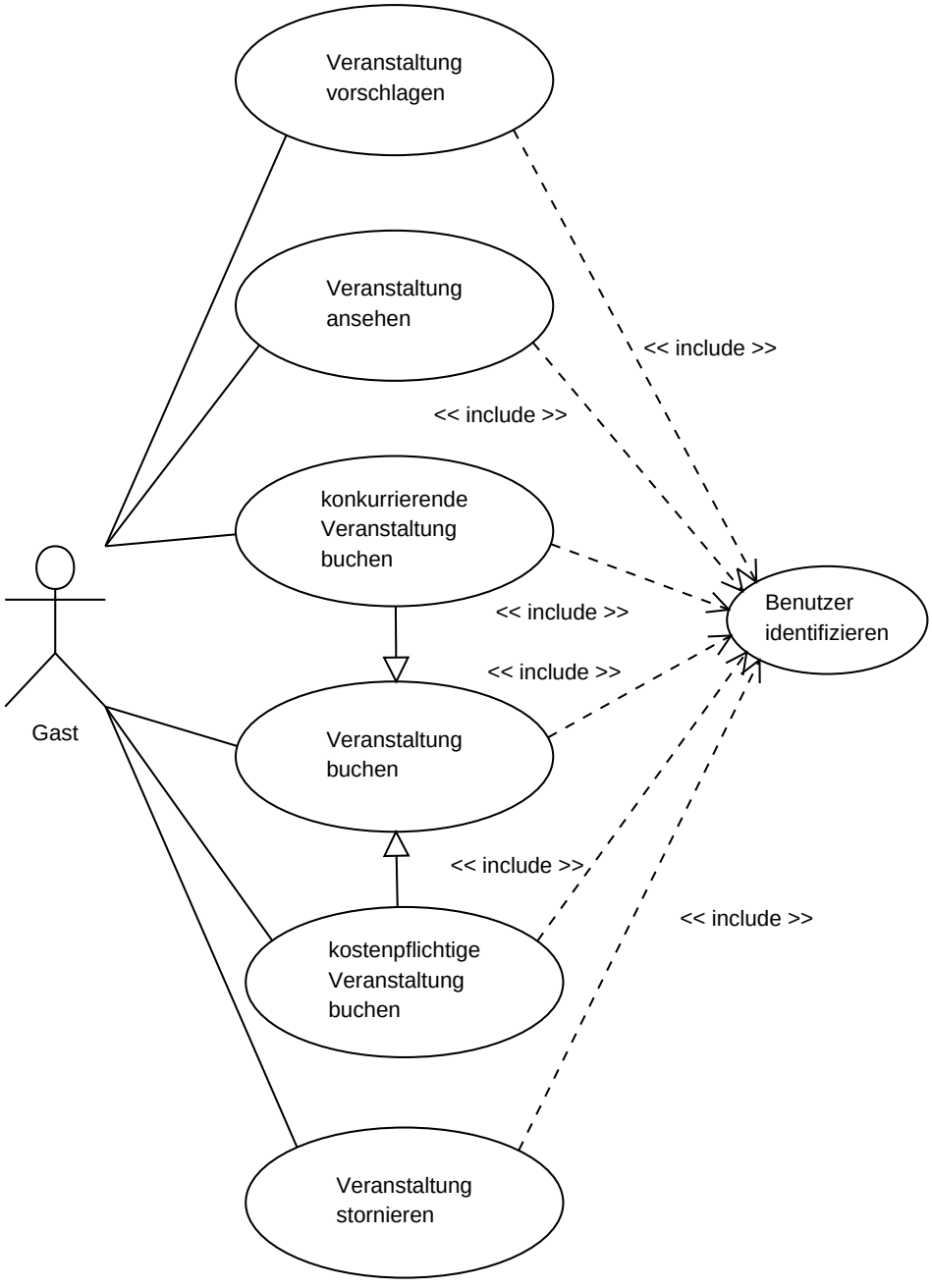



Abbildung 2.8: Anwendungsfalldiagramm Veranstaltung (Gast)

| Name: | **Veranstaltung ansehen** |
|---|---|
| Kurzbeschreibung: | Ein Benutzer erhält eine Detailansicht der Veranstaltung. |
| Akteure: | Gast, Clubmanager |
| Auslöser: | Ein Benutzer möchte Informationen über eine Veranstaltung erhalten. |
| Vorbedingungen: | Benutzer ist identifiziert. |
| Eingehende Informationen: | Daten der Veranstaltung (Titel, Datum, Zeit, Kategorie) |
| Essentieller Ablauf: | 1. Veranstaltung suchen:<br>Der Benutzer sucht die gewünschte Veranstaltung aus dem öffentlichen Kalender aus. Handelt es sich bei der gesuchten Veranstaltung um eine schon von ihm gebuchte Veranstaltung, so kann er sie auch aus seinem privaten Kalender auswählen.<br>2. Veranstaltung anzeigen:<br>Die Veranstaltung wird mit sämtlichen Daten angezeigt. |

Tabelle 2.5: Anwendungsfall Veranstaltung ansehen

| Name: | **Veranstaltung buchen** |
|---|---|
| Kurzbeschreibung: | Für einen Gast wird eine Veranstaltung gebucht. Diese Veranstaltung wird in seinen privaten Kalender eingetragen. |
| Akteure: | Gast |
| Auslöser: | Ein Gast möchte eine Veranstaltung buchen. |
| Vorbedingungen: | Benutzer ist identifiziert. Gewünschte Veranstaltung wurde ausgewählt. |
| Eingehende Informationen: | Daten der Veranstaltung, Buchungswunsch |
| Ergebnisse: | Buchung, Buchungsbestätigung |
| Nachbedingungen: | Die Veranstaltung ist im privaten Kalender des Gastes verzeichnet. Die Teilnehmeranzahl der Veranstaltung ist erhöht. |
| Essentieller Ablauf: | 1. Buchungswunsch aufnehmen:<br>Der Gast stellt eine Buchungsanfrage.<br>2. Verfügbarkeit prüfen:<br>Das System überprüft ob ausreichend Plätze vorhanden sind.<br>3. Veranstaltung buchen:<br>Für den Gast wird die Veranstaltung gebucht. Die Anzahl der Teilnehmer wird um eins erhöht. Gegebenenfalls wird die Anzahl der verfügbaren Plätze um eins erniedrigt.<br>4. Veranstaltung eintragen:<br>Die Veranstaltung wird in den privaten Kalender des Gastes eingetragen.<br>5. Buchung bestätigen:<br>Der Gast erhält eine Bestätigung über die Buchung der Veranstaltung. |

Tabelle 2.6: Anwendungsfall Veranstaltung buchen

| Name: | **Veranstaltung stornieren** |
|---|---|
| Kurzbeschreibung: | Ein Gast storniert eine Veranstaltung. Diese wird aus seinem privaten Kalender entfernt. |
| Akteure: | Gast |
| Auslöser: | Ein Gast möchte eine Veranstaltung stornieren. |
| Vorbedingungen: | Benutzer ist identifiziert und die entsprechende Veranstaltung ist ausgewählt. |
| Eingehende Informationen: | Daten der Veranstaltung, Stornierungswunsch |
| Ergebnisse: | Stornierung |
| Nachbedingungen: | Die Veranstaltung ist aus dem privaten Kalender des Gastes entfernt. Die Teilnehmerzahl der Veranstaltung ist verringert. |
| Essentieller Ablauf: | 1. Stornierungswunsch aufnehmen:<br>Der Gast stellt eine Stornierungsanfrage.<br>2. Prüfen, ob es sich hierbei um eine kostenpflichtige Veranstaltung handelt.<br>• kostenplichtige Veranstaltung:<br>Der Gast erhält einen Hinweis auf die entstehenden Kosten, die aufgrund seiner Stornierung anfallen werden.<br>• Bestätigung prüfen:<br>Das System überprüft, ob der Gast den Hinweis angenommen hat.<br>3. Veranstaltung stornieren:<br>Die Veranstaltung wird für den Gast storniert. Die Anzahl der Teilnehmer wird um eins erniedrigt. Gegebenenfalls wird die Anzahl der verfügbaren Plätze um eins erhöht.<br>4. Veranstaltung austragen:<br>Die Veranstaltung wird aus dem privaten Kalender des Gastes ausgetragen.<br>5. Stornierung bestätigen:<br>Der Gast erhält eine Bestätigung über die Stornierung der Veranstaltung. |

Tabelle 2.7: Anwendungsfall Veranstaltung stornieren

| Name: | **Kostenpflichtige Veranstaltung buchen** |
|---|---|
| Spezialisierung von: | Veranstaltung buchen |
| Kurzbeschreibung: | Für einen Gast wird eine kostenpflichtige Veranstaltung gebucht. Die Veranstaltung wird in seinem privaten Kalender eingetragen. |
| Akteure: | Gast |
| Auslöser: | Der Gast möchte eine kostenpflichtige Veranstaltung buchen. |
| Vorbedingungen: | Benutzer ist identifiziert. Gewünschte Veranstaltung wurde ausgewählt. |
| Eingehende Informationen: | Daten der Veranstaltung, Buchungswunsch |
| Ergebnisse: | Buchung, Buchungsbestätigung |
| Nachbedingungen: | Die Veranstaltung ist im privaten Kalender des Gastes eingetragen. Die Teilnehmeranzahl der Veranstaltung ist erhöht. |
| Essentieller Ablauf: | 1. Buchungswunsch aufnehmen:<br>Siehe Tabelle 2.6 auf Seite 32.<br>2. Rechtsverbindlicher Hinweis:<br>Der Gast erhält einen Hinweis auf Verbindlichkeit der Buchung. Dabei werden ihm auch die Kosten genannt, die bei einer Stornierung seinerseits entstehen können. Außerdem geht der Gast bei Bestätigung des Hinweises einen rechtsverbindlichen Vertrag ein.<br>3. Bestätigung prüfen:<br>Das System überprüft, ob der Gast die Bedingungen angenommen hat.<br>4. *Veranstaltung buchen* (Siehe Tabelle 2.6 auf Seite 32). |

Tabelle 2.8: Anwendungsfall kostenpflichtige Veranstaltung buchen

| Name: | **Konkurrierende Veranstaltung buchen** |
|---|---|
| Spezialisierung von: | Veranstaltung buchen |
| Kurzbeschreibung: | Für einen Gast wird eine konkurrierende Veranstaltung gebucht. Dabei wird die bisherige Veranstaltung aus seinem Kalender entfernt und die neue Veranstaltung eingetragen. |
| Akteure: | Gast |
| Auslöser: | Der Gast möchte eine konkurrierende Veranstaltung buchen |
| Vorbedingungen: | Benutzer ist identifiziert. Gewünschte Veranstaltung wurde ausgewählt. |
| Eingehende Informationen: | Daten der Veranstaltung, Buchungswunsch |
| Ergebnisse: | Buchung, Buchungsbestätigung, Stornobestätigung |
| Nachbedingungen: | Die neue Veranstaltung wurde im privaten Kalender des Gastes eingetragen. Im Gegenzug wurde die alte Veranstaltung aus dem privaten Kalender entfernt. Die Teilnehmeranzahl der neuen Veranstaltung wurde erhöht, während die Teilnehmeranzahl der alten Veranstaltung erniedrigt wurde. |
| Essentieller Ablauf: | 1. Buchungswunsch aufnehmen: Siehe Tabelle 2.6 auf Seite 32.<br>2. *Veranstaltung stornieren* (siehe Tabelle 2.7 auf Seite 33).<br>3. Prüfen, ob die konkurrierende Veranstaltung kostenpflichtig ist.<br>• kostenpflichtige Veranstaltung *kostenpflichtige Veranstaltung buchen* (siehe Tabelle 2.8 auf Seite 34)<br>• kostenlose Veranstaltung *Veranstaltung buchen* (siehe Tabelle 2.6 auf Seite 32) |

Tabelle 2.9: Anwendungsfall konkurrierende Veranstaltung buchen

| Name: | **Veranstaltung vorschlagen** |
|---|---|
| Kurzbeschreibung: | Ein Gast schlägt eine neue Veranstaltung vor. |
| Akteure: | Gast |
| Auslöser: | Ein Gast möchte eine private Veranstaltung erfassen. |
| Vorbedingungen: | Benutzer ist identifiziert. |
| Eingehende Informationen: | Daten der Veranstaltung. |
| Ergebnisse: | Clubmanager erhält eine Mitteilung über eine neue Veranstaltung. |
| Nachbedingungen: | Veranstaltung wird im öffentlichen Kalender aufgenommen. |
| Essentieller Ablauf: | 1. Daten der Veranstaltung aufnehmen:<br>Neben dem Zeitraum wird ein Titel und eine Beschreibung zur Veranstaltung benötigt.<br>2. Veranstaltung vorschlagen:<br>Die Veranstaltung wird im System erfasst. Der Clubmanager erhält eine Mitteilung darüber. |

Tabelle 2.10: Anwendungsfall Veranstaltung vorschlagen

| Name: | **Veranstaltung erfassen** |
|---|---|
| Kurzbeschreibung: | Ein Clugmanager erfasst eine neue Veranstaltung. |
| Akteure: | Clubmanager |
| Auslöser: | Ein Clubmanager möchte eine neue Veranstaltung erfassen oder ein Gast hat eine Veranstaltung vorgeschlagen |
| Vorbedingungen: | Benutzer ist identifiziert. |
| Eingehende Informationen: | Daten der Veranstaltung. |
| Ergebnisse: | Sofern ein Gast die Veranstaltung vorgeschlagen hat, erhält der Gast eine Mitteilung darüber, dass die Veranstaltung erfasst wurde. |
| Nachbedingungen: | Veranstaltung wurde im öffentlichen Kalender aufgenommen. |
| Essentieller Ablauf: | 1. Daten der Veranstaltung aufnehmen:<br>Neben dem Zeitraum wird ein Titel und eine Beschreibung zur Veranstaltung benötigt. Wurde die Veranstaltung vorgeschlagen, so können die eingegebenen Daten bearbeitet werden. Optional kann noch ein Bild hinzugefügt werden.<br>2. Veranstaltung kategorisieren:<br>Der Clubmanager weist der Veranstaltung eine Kategorie zu.<br>3. Veranstaltung eintragen:<br>Die Veranstaltung wird in den öffentlichen Kalender eingetragen und kann gebucht werden. Sollte ein Gast die Veranstaltung vorgeschlagen haben, so erhält der Gast eine Mitteilung darüber. |

Tabelle 2.11: Anwendungsfall Veranstaltung erfassen

| Name: | **Veranstaltung bearbeiten** |
|---|---|
| Kurzbeschreibung: | Der Clubmanager bearbeitet den Inhalt einer vorhandenen Veranstaltung. |
| Akteure: | Clubmanager |
| Auslöser: | Daten einer Veranstaltung haben sich geändert. |
| Vorbedingungen: | Benutzer ist identifiziert. Gewünschte Veranstaltung ist ausgewählt. |
| Eingehende Informationen: | zu ändernde Daten |
| Ergebnisse: | Änderungsmitteilung an Gäste, die diese Veranstaltung gebucht haben. |
| Nachbedingungen: | Veranstaltung wurde geändert. |
| Essentieller Ablauf: | 1. Daten der Veranstaltung bearbeiten<br>2. Auf Überschneidung prüfen:<br>Bei geändertem Datum oder geänderter Zeit überprüft das System ob es bei den privaten Kalendern zu Überschneidungen führt. Der Clubmanager erhält einen Hinweis darauf bei wie vielen Gästen eine Überschneidung entsteht und kann die Änderung korrigieren, abbrechen oder fortführen.<br>3. Veranstaltung speichern:<br>Die Veranstaltung wird unter Berücksichtigung der Änderung im öffentlichen und in den privaten Kalendern gespeichert. Sollte es in einem privaten Kalender zu Überschneidungen kommen, so wird sie trotzdem gespeichert.<br>4. Mitteilungen versenden:<br>Die Gäste, die diese Veranstaltung gebucht haben, erhalten eine Mitteilung. Sollte es zu einer Überschneidung kommen, so wird darauf hingewiesen. |

Tabelle 2.12: Anwendungsfall Veranstaltung bearbeiten

| Name: | **Veranstaltung entfernen** |
|---|---|
| Kurzbeschreibung: | Ein Veranstaltung wird aus dem öffentlichen Kalender entfernt. |
| Akteure: | Clubmanager |
| Auslöser: | Die Veranstaltung kann mangels Teilnehmern oder aus anderen Gründen nicht stattfinden. |
| Vorbedingungen: | Benutzer ist identifiziert. Gewünschte Veranstaltung wurde ausgewählt. |
| Eingehende Informationen: | Löschungsauftrag |
| Ergebnisse: | Änderungsmitteilung an Gäste, die diese Veranstaltung gebucht hatten. |
| Nachbedingungen: | Veranstaltung wurde gelöscht. |
| Essentieller Ablauf: | 1. Löschauftrag bestätigen:<br>Der Clubmanager muss die Löschung der Veranstaltung zusätzlich bestätigen.<br>2. Veranstaltung entfernen:<br>Die Veranstaltung wird sowohl aus dem öffentlichen, als auch aus den privaten Kalendern der Gäste, die diese Veranstaltung gebucht haben, entfernt.<br>3. Mitteilung versenden:<br>Allen Gästen eine Mitteilung zukommen lassen, dass die Veranstaltung abgesagt wurde und nicht stattfindet. Bei einer kostenpflichtigen Veranstaltung einen Hinweis hinzufügen, dass keine Stornierungsgebühren anfallen. |

Tabelle 2.13: Anwendungsfall Veranstaltung entfernen

### 2.5.4 Mitteilungen

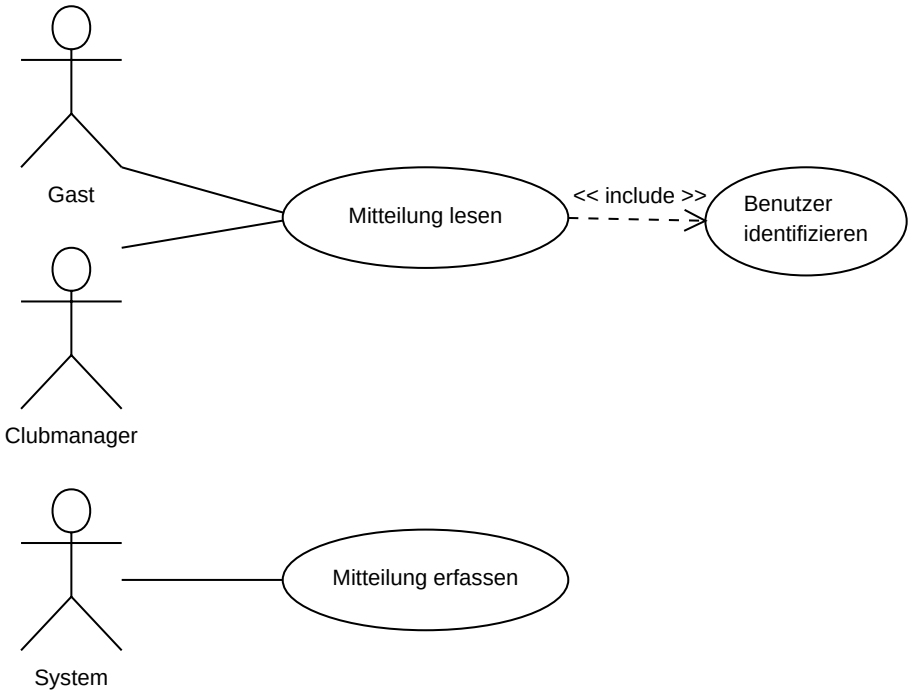



Abbildung 2.9: Anwendungsfalldiagramm Mitteilung

Bei den Mitteilungen handelt es sich um Benachrichtigungen die vom System erzeugt werden (siehe Tabelle 2.14 auf Seite 41). Wenn beispielsweise ein Gast eine Veranstaltung vorschlägt (siehe Tabelle 2.10 auf Seite 36), so erhält der Clubmanager eine Mitteilung darüber. Die Gäste erhalten eine Mitteilung zum Beispiel bei Änderung einer von ihnen gebuchten Veranstaltung.

Nachdem sich der Benutzer am System identifiziert hat erhält er neben verschiedenen Auswahlmöglichkeiten eine Liste von verfügbaren Mitteilungen. Er kann die gewünschte Mitteilung auswählen, um sie zu lesen (siehe Tabelle 2.15 auf Seite 42).

| Name: | **Mitteilung erfassen** |
|---|---|
| Kurzbeschreibung: | Eine Mitteilung wird vom System erfasst. |
| Akteure: | System |
| Auslöser: | Eine Veranstaltung wurde vorgeschlagen, bearbeitet, entfernt oder erfasst. |
| Eingehende Informationen: | Daten der Veranstaltung. |
| Ergebnisse: | Mitteilung wurde erfasst. |
| Essentieller Ablauf: | 1. Mitteilung erfassen:<br>Die Mitteilung wird mit den Daten der Veranstaltung versehen.<br>2. Empfänger ermitteln:<br>Das System prüft welche Benutzer die betroffene Veranstaltung gebucht haben und fügt sie der Empfängerliste hinzu. Wurde eine neue Veranstaltung erfasst, so werden anhand der Kategorie und des Profils der Gäste die entsprechenden Benutzer der Empfängerliste hinzugefügt.<br>3. Mitteilung bereitstellen:<br>Die Mitteilung wird den entsprechenden Benutzern zur Verfügung gestellt. |

Tabelle 2.14: Anwendungsfall Mitteilung erfassen

| Name: | **Mitteilung lesen** |
|---|---|
| Akteure: | Gast, Clubmanager |
| Auslöser: | Eine Mitteilung wurde vom System generiert. |
| Vorbedingungen: | Benutzer ist identifiziert. |
| Eingehende Informationen: | Auswahl der Mitteilung |
| Essentieller Ablauf: | 1. Auswahl der Mitteilung:<br>Der Benutzer erhält in der Übersicht eine Liste von verfügbaren Mitteilungen und wählt die gewünschte Mitteilung aus.<br>2. Mitteilung lesen:<br>Der Benutzer erhält die Mitteilung in voller Länge.<br>3. Mitteilung als gelesen markieren:<br>Der Benutzer entscheidet, ob die Mitteilung als gelesen markiert wird oder nicht. |

Tabelle 2.15: Anwendungsfall Mitteilung lesen

## 2.6 Anforderungen an das System

Die in diesem Kapitel skizzierte Anwendung beschreibt in den Szenarios den Einsatz eines Personal Digital Assistent (PDA). Dieser PDA verfügt standardmäßig über einfache Anwendungen, mit denen man Adressen, Termine und Aufgaben verwalten kann. Darüber hinaus bieten die Geräte heutzutage die Möglichkeit weitere Anwendungen zu installieren. So wäre auch die Installation einer Ferienclub-Terminverwaltung denkbar. Es gibt jedoch mehrere Hersteller solcher PDAs die durchaus unterschiedliche Betriebssysteme wie Microsoft Pocket PC[3], Microsoft Windows CE[4] oder Palm OS[5] einsetzen. Würde man einen Client für einen bestimmten PDA implementieren, so müsste man sich auf ein Betriebssystem festlegen. Bei einem Wechsel des PDAs könnte unter Umständen auch ein Wechsel des Betriebssystems stattfinden, wodurch der Client eventuell neu implementiert werden müsste.

Obwohl der Personal Digital Assistent (PDA) in der Regel mit einem Terminkalender ausgestattet ist, soll hier auf dessen Nutzung verzichtet werden. Zum Einen verfügt der PDA über eine unzureichende Benutzerverwaltung (siehe [Lüpke 2004, Seite 30ff]), zum Anderen ist der Einsatz verschiedener Clients wünschenswert, wodurch eine serverseitige Verwaltung der Termine erforderlich wird.

Bei den meisten PDAs ist der Web-Browser Bestandteil der mitgelieferten Software. So liegt eine mögliche Lösung in der Verwendung einer Web-Anwendung. Dazu kommt das Interessenten des Ferienclubs die Möglichkeit erhalten sollen sich über Veranstaltungen zu informieren. Die Informationen sollten sinnvollerweise im Internet zur Verfügung stehen und über einen handelsüblichen Browser eines Personal Computers erreichbar sein. Des Weiteren ist die Spezifizierung des Profils eines Gastes über das Internet zu ermöglichen, damit er schon vor seiner Anreise Veranstaltungstipps erhalten kann. Das System soll also von unterschiedlichen Clients erreichbar sein.

Außerdem stellt die beschriebene Terminverwaltung einen kleinen Teil der im Ferienclub verwendeten Software dar. So werden neben der Terminverwaltung weitere Anwendungen, wie zum Beispiel eine Personalverwaltung oder ein Buchhaltungssystem, benötigt um den Ferienclub zu betreiben. Um diese Flexibilität zu gewährleisten muss die Anwendung bestimmte Kriterien erfüllen. So sollte die Anwendung modular aufgebaut sein damit Teile der Terminverwaltung von fremden Komponenten nutzbar sind. Außerdem sollte das System erweiterbar und wartbar sein damit zukünftige Funktionalitäten schnell abgebildet werden können.

Hierzu bedient sich die Informatik bestimmter Vorgehensweisen. Eine zentrale Eigenschaft ist die Modularität. Dabei werden so genannte Komponenten eingesetzt welche unabhängig voneinander benutzbar sind. Ein System setzt sich aus mehreren

[3] http://www.microsoft.com/windowsmobile/pocketpc/ppc/default.mspx
[4] http://msdn.microsoft.com/embedded/prevver/ce3/default.aspx
[5] http://www.palmsource.com/palmos/

unterschiedlichen Komponenten zusammen die einzeln ausgetauscht oder hinzugefügt werden können. Für die erfolgreiche Entwicklung wiederverwendbarer Komponenten ist die Definition von Schnittstellen zwingend notwendig. Dabei müssen die Schnittstellen für die Wiederverwendung allgemein gehalten werden, während die speziellen Anforderungen auch erfüllt werden müssen. Dieser Umstand beschreibt die Wartbarkeit und Erweiterbarkeit. Sollte eine Komponente fehlerhaft implementiert worden sein, so können die Fehler zunächst behoben werden und im Anschluss daran die fehlerhafte Komponente durch die nun fehlerbereinigte Komponente ersetzt, beziehungsweise ausgetauscht werden. Das System ist also wartbar. Genauso kann eine bestehende Komponente um weitere Funktionalitäten erweitert werden. Sobald die neuen Funktionen implementiert und getestet wurden, kann die neue Komponente die alte Komponente ersetzen und das System somit erweitert werden, es ist also erweiterbar.

Ein weiterer Vorteil der Modularität besteht darin, dass schon vorhandene Komponenten wiederverwendet werden können. So können Komponenten aus einem anderen Projekt verwendet werden, die die notwendige Funktionalität beinhalten. Bei der Buchung einer kostenpflichtigen Veranstaltung könnte hier zum Beispiel auf eine Komponente der Buchhaltung zugegriffen werden um die Buchung zu registrieren.

Des Weiteren soll die hier behandelte Anwendung portabel sein. Es soll möglich sein die Terminverwaltung auf unterschiedlichen Betriebssystemen, wie Linux oder Windows, betreiben zu können. Außerdem ist für die Portabilität und auch für den Einsatz fremder Komponenten die Nutzung von Standards notwendig.

Bei der Verwendung von Standards ergibt sich ein weiterer Vorteil. Neue Projektmitglieder, die sich an der Implementierung beteiligen möchten, haben eine kürzere Einarbeitungszeit. Wenn darüber hinaus etablierte Entwurfsmuster [Gamma u. a. 2000] in der Realisierung genutzt werden, so ist auch hier davon auszugehen, dass die Einarbeitung weiter verkürzt wird, da bekannte Lösungsansätze gewählt wurden.

# 3 Entwurf

## 3.1 Grundlegende Entwurfsentscheidungen

Für die Terminverwaltung soll eine objektorientierte Programmiersprache eingesetzt werden. Da die Anforderung besteht die Terminverwaltung unter Umständen auf unterschiedlichen Betriebssystemen, wie Windows oder Linux, zu betreiben, fällt die Wahl auf Java. Java wird in einer virtuellen Maschine ausgeführt die für mehrere Betriebssysteme zur Verfügung steht. Darüber hinaus hat Java sich auf dem Markt etabliert und verfügt über eine große Verbreitung.

Da die Terminverwaltung auf unterschiedlichen Clients ausgeführt werden soll (siehe Abbildung 3.1), wird sich hier für einen Thin-Client entschieden – einen Web-Browser. Der Web-Browser übernimmt die Darstellung der Benutzungsoberfläche. Dabei sollen im Client keinerlei Plausibilitätsprüfungen stattfinden.

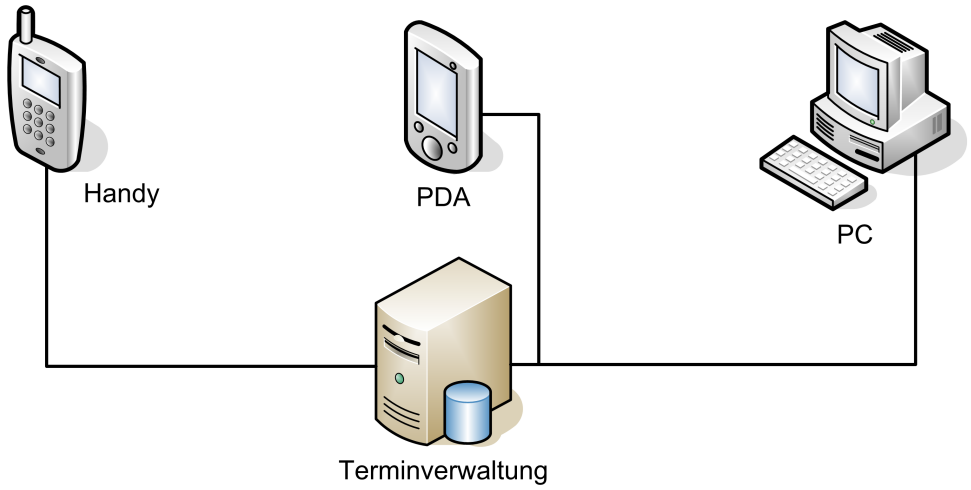



Abbildung 3.1: Einsatz von verschiedenen Clients

Neben der Präsentation beinhaltet die Terminverwaltung die Datenhaltung und die Logik. Es hat sich in der Vergangenheit etabliert diese drei Bereiche getrennt voneinander zu behandeln. Es hat sich gezeigt, dass sich im Fachkonzept konstant bleibt, während die Benutzerschnittstelle häufigeren Anpassungen unterliegt. Um diese Trennung zu realisieren wird die so genannte Drei-Schichten-Architektur ver-

wendet (siehe Abbildung 3.2). Bei diesem Verfahren ist zum Beispiel der Einsatz mehrerer Benutzerschnittstellen ohne redundante Implementierung des Fachkonzepts realisierbar.

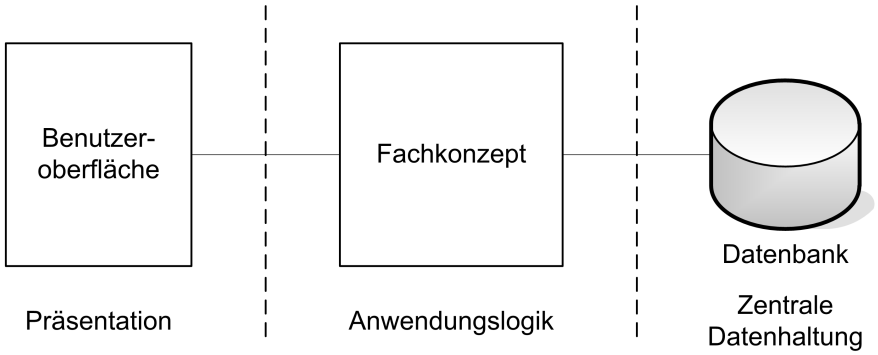



Abbildung 3.2: Drei-Schichten-Architektur

In Abbildung 3.2 ist eine *strenge* Drei-Schichten-Architektur dargestellt. Bei einer strengen Drei-Schichten-Architektur ist der Zugriff nur zwischen zwei benachbarten Schichten erlaubt, während die flexible Drei-Schichten-Architektur der Benutzeroberfläche einen Zugriff auf die Datenhaltung erlaubt. Hier soll die strenge Form der Drei-Schichten-Architektur verwendet werden. Der Vorteil liegt darin, dass die Präsentationsschicht unabhängig von der Datenhaltung ist [Balzert 1999].

Für die Datenhaltung wird im Prototypen zunächst das Dateisystem des Betriebssystems verwendet. Dabei sind die Schnittstellen zwischen dem Fachkonzept und der zentralen Datenhaltung so zu wählen, dass ein einfacher Austausch der Datenhaltung möglich ist.

## 3.2 Software-Entwurf

Dieser Abschnitt behandelt die Anwendungslogik. Anhand der bisher unternommenen Überlegungen und Annahmen werden notwendige Komponenten und deren Beziehungen untereinander bestimmt. Dazu ist es notwendig zunächst die Geschäftsklassen zu definieren die den entsprechenden Komponenten zugewiesen werden. Nach anschließender Verfeinerung des Klassenmodells wird anhand von Sequenzdiagrammen die Interaktion zwischen den Klassen aufgezeigt.

### 3.2.1 Geschäftsklassen

Im Folgenden werden die Geschäftsklassen definiert. Anhand der Szenarios und der Anwendungsfälle aus Kapitel 2 sollen die wichtigsten fachlichen Klassen und ihre Beziehungen untereinander identifiziert werden.

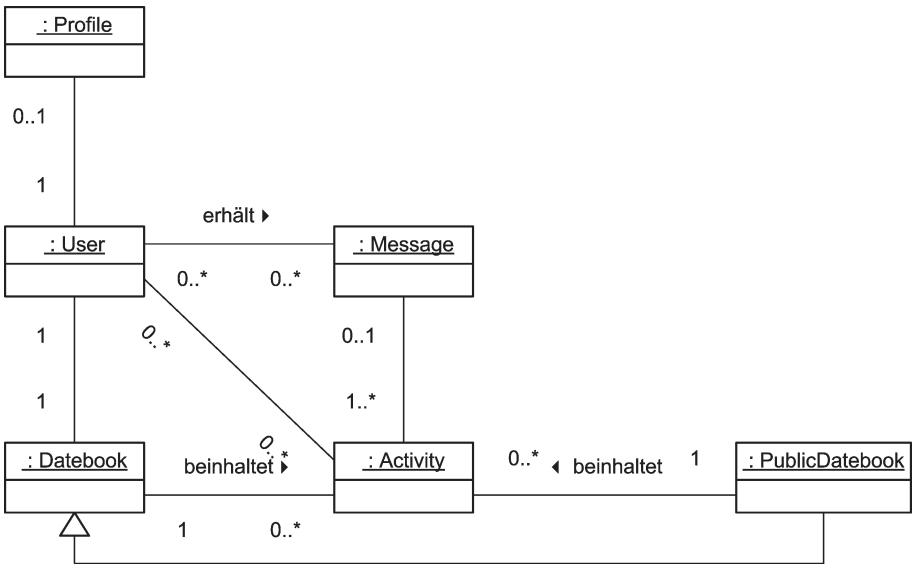



Abbildung 3.3: Die fachlichen Klassen in einem ersten Entwurf

**Profil** (`Profile`) beinhaltet die Angaben über die Interessen des Benutzers. Zu jedem Benutzer kann es ein Profil geben, wobei jedes Profil einem Benutzer zugeordnet sein muss. Aus den Informationen des Profils können später Veranstaltungstipps generiert werden.

**Benutzer** (`User`) repräsentiert den Gast, die Rezeption oder den Clubmanager des Ferienclubs. Der Benutzer ist eine Person, die Veranstaltungen seinen Rechten entsprechend buchen, ändern, löschen oder erfassen kann. Der Benutzer wird über eine eindeutige Kennung identifiziert.

**Kalender** (`Datebook`) ist nur vom Benutzer einsehbar und zu verwalten. Der Kalender enthält Veranstaltungen die vom Benutzer gebucht wurden. Über den Kalender ist es dem Benutzer möglich Veranstaltungen zu stornieren.

**Veranstaltung** (`Activity`) kann von einem Benutzer gebucht werden. Die Veranstaltung verfügt in der Regel über ein bestimmtes Kontingent an Plätzen, welches von ihr verwaltet wird. Sollten alle Plätze vergeben sein sind keine weiteren Buchungen möglich.

**Mitteilung** (`Message`) Zu jeder Mitteilung gibt es mindestens eine Veranstaltung. Mitteilungen werden vom System generiert. Wird beispielsweise eine Veranstaltung storniert, so erhält jeder Gast eine Mitteilung, die mit der entsprechenden Veranstaltung verknüpft ist. Wenn es sich bei der Mitteilung um einen Ver-

anstaltungstipp handelt, dann kann die Mitteilung mehrere Veranstaltungen enthalten.

**Öffentlicher Kalender** (`PublicDatebook`) ist eine Spezialisierung des Kalenders. Er enthält sämtliche Veranstaltungen und ist keinem Benutzer zugeordnet.

Bringt man die eben beschriebenen fachlichen Klassen in Beziehung zueinander, ergibt sich ein erstes Klassendiagramm, welches in Abbildung 3.3 dargestellt ist.

### 3.2.2 Fachliche Architektur

Bei der Betrachtung der Geschäftsklassen lassen sich drei Aufgabenfelder identifizieren. Die erste Aufgabe besteht in der Verwaltung der Benutzer. Zu diesem Bereich lässt sich auf der einen Seite die Klasse `User` zuordnen und zum Anderen die Klasse `Profile`. Der zweite Bereich befasst sich mit dem Thema Termine. Neben dem Kalender `Datebook` für den Benutzer, ist natürlich auch die Spezialisierung von `Datebook`, der öffentliche Kalender `PublicDatebook`, in der Komponente enthalten. Eine zentrale Rolle nehmen schließlich die Veranstaltungen (`Activity`) und die damit verbundenen Mitteilungen (`Message`) ein.

Um die Komplexität des Systems zu reduzieren empfiehlt es sich die Bereiche in einzelne Komponenten zu kapseln. Abbildung 3.4 veranschaulicht die obigen Überlegungen und skizziert mögliche Komponenten.

Im weiteren Verlauf des Entwurfes wird es eine Verfeinerung der Objekte geben, wodurch auch die Anzahl der Objekte gegenüber den fachlichen Klassen aus Abschnitt 3.2.1 weiter ansteigen wird. Die Komplexität wird also erhöht. Um genau diese Komplexität zu minimieren bedient man sich dem Fassadenmuster:

> „Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.“[1] [Gamma u. a. 2000, Seite 185]

Das Fassadenmuster reduziert die Anzahl von Objekten, die vom Klienten gehandhabt werden müssen. Des Weiteren ist dadurch eine lose Koppelung gegeben, was den Austausch der Komponenten erleichtert.

### 3.2.3 Klassendiagramme

Im Folgenden werden die Objekte und deren Zuständigkeiten verfeinert. Die ersten Entwürfe der Klassendiagramme werden getrennt nach den oben bestimmten Kompo-

[1] „Biete eine einheitliche Schnittstelle zu einer Menge von Schnittstellen eines Subsystems. Die Fassadenklasse definiert eine abstrakte Schnittstelle, welche die Benutzung des Subsystems vereinfacht.“

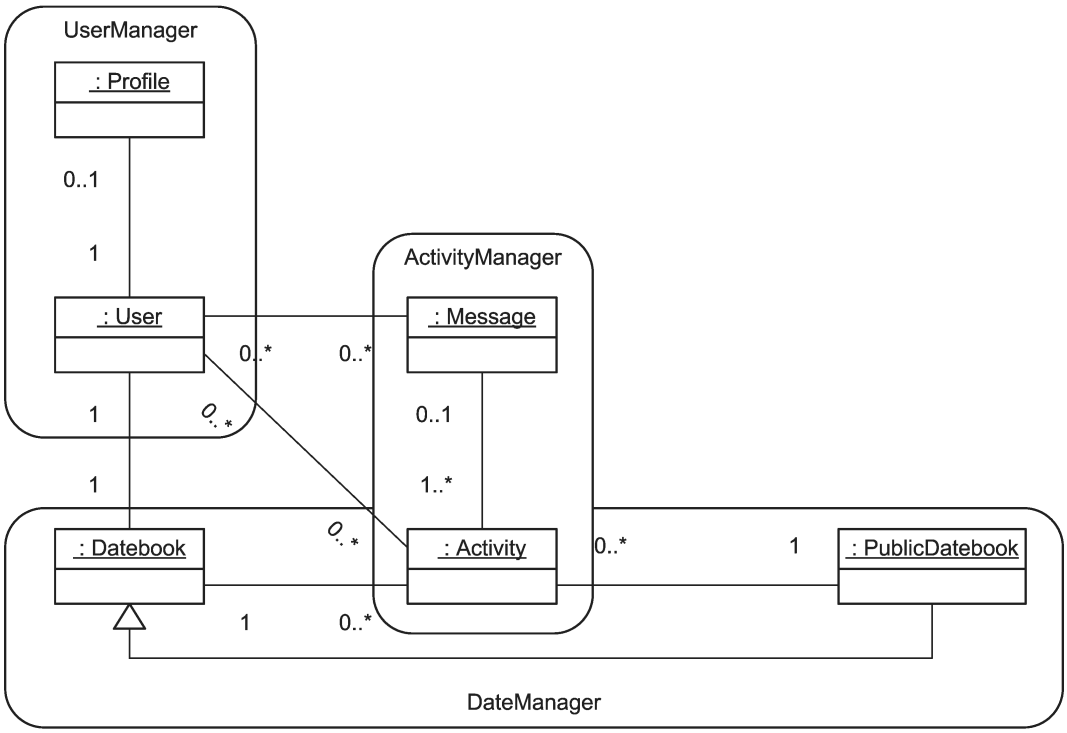



Abbildung 3.4: Aufteilung der fachlichen Klassen in die geplanten Komponenten

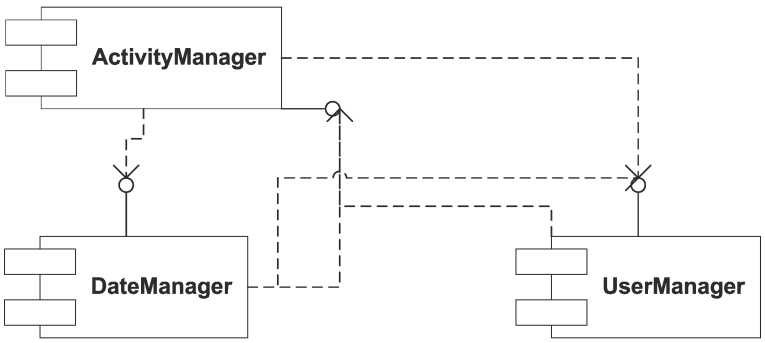



Abbildung 3.5: Komponentenmodell

nenten dargestellt. Die hier aufgezeigten Methoden decken folgende Anwendungsfälle ab:

- Öffentlichen Kalender ansehen (Tabelle 2.4, Seite 28),
- Veranstaltung buchen (Tabelle 2.6, Seite 32) und
- Mittelung erfassen (Tabelle 2.14, Seite 41).

### Komponente ActivityManager

`ActivityManager` behandelt alles rund um die Verwaltung der Veranstaltungen. Hierzu wird eine Klasse `ActivityRepository` benötigt die sämtliche Veranstaltungen verwaltet. Hier können Veranstaltungen hinzugefügt, gelöscht und geändert werden. Die Veranstaltungen selbst werden von der Klasse `Activity` repräsentiert. `Activity` verwaltet die Daten, wie beispielsweise den Zeitpunkt der Veranstaltung, verfügbare Plätze und angemeldete Teilnehmer. Die Klasse `ActivityBooking` stellt Funktionalitäten zum Buchen und Stornieren von Veranstaltungen zur Verfügung. Schließlich wird eine Klasse benötigt die für die Erstellung von Mitteilungen zuständig ist. Mit Hilfe der Klasse `Message` hat das System die Möglichkeit Mitteilungen zu erfassen. Dabei können Veranstaltungen einer Mitteilung hinzugefügt werden, um zum Beispiel Veranstaltungstipps zu generieren. Diese Veranstaltungstipps werden an eine Auswahl von Benutzern geschickt, die wiederum der Mitteilung bekannt gemacht werden. Abbildung 3.6 zeigt die Klassen aus der Komponente `ActivityManager`.

### Komponente UserManger

Die Benutzerverwaltung wird in der Komponente `UserManager` behandelt. Dazu gehört die Verwaltung sämtlicher Benutzer, die von der Klasse `UserRepository` übernommen wird. Der Benutzer wird durch die Klasse `User` repräsentiert. Mit Hilfe von `User` kann zum Einen das Profil (`Profile`) des Benutzers eingestellt oder verändert werden, zum Anderen kann dem Benutzer eine Rolle (`Role`) zugewiesen werden.

### Komponente DateManager

Schließlich müssen noch die Termine verwaltet werden. Jedem Benutzer wird ein persönlicher Kalender zur Verfügung gestellt. Dieser persönliche Kalender wird von der Klasse `Datebook` dargestellt. Dem Kalender können neue Termine hinzugefügt und bestehende Termine entfernt oder geändert werden. Darüber hinaus erhält man vom Kalender eine Übersicht aller Termine eines ausgewählten Tages. Die Termine wiederum werden von der Klasse `Appointment` repräsentiert. Dabei wird dem Termin eine Veranstaltung zugeordnet. Eine besondere Rolle spielt die Klasse `PublicDatebook`, die den öffentlichen Veranstaltungskalender abbildet.

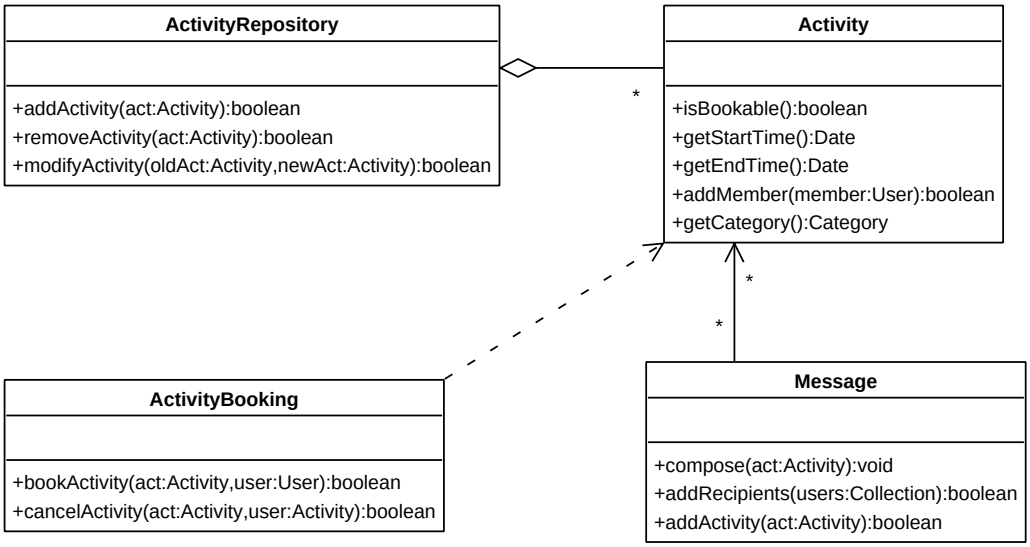



Abbildung 3.6: Klassendiagramm `ActivityManager`

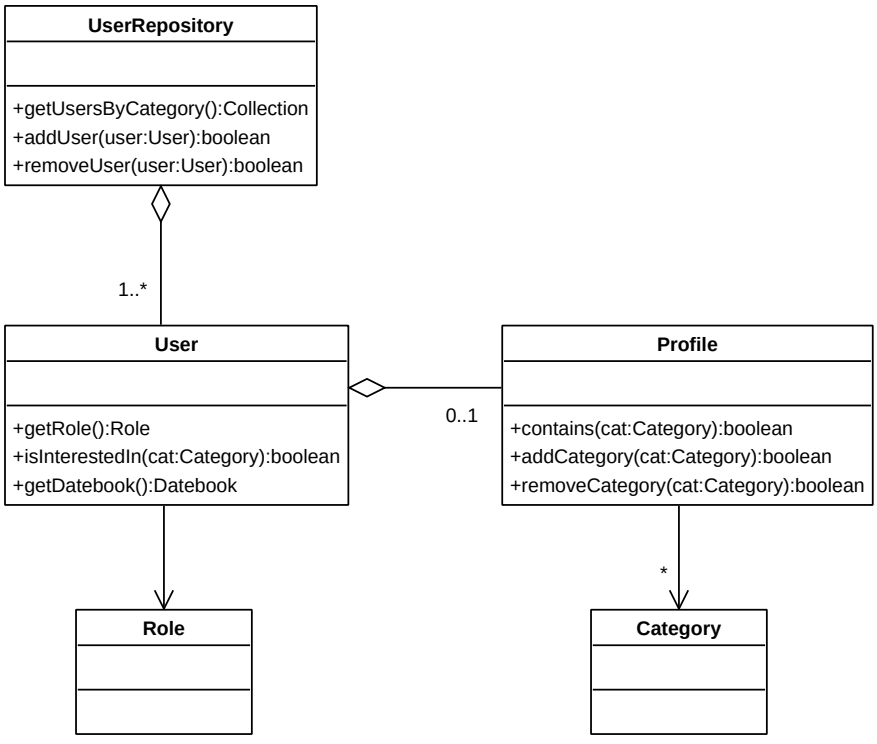



Abbildung 3.7: Klassendiagramm `UserManager`

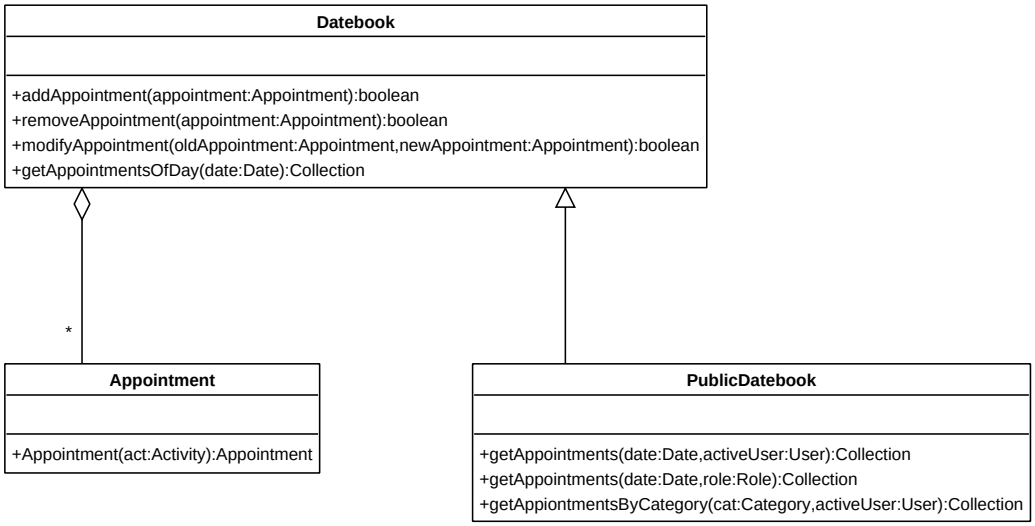



Abbildung 3.8: Klassendiagramm `DateManager`

### 3.2.4 Abbildung der Anwendungsfälle

Mit Hilfe von Sequenzdiagrammen soll nun eine Verfeinerung der im letzten Abschnitt[2] genannten Anwendungsfälle vorgenommen werden.

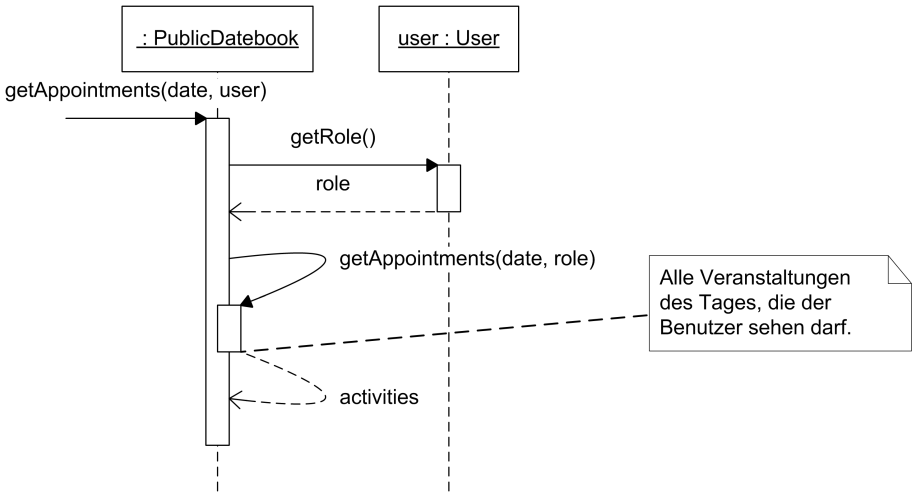



Abbildung 3.9: Sequenzdiagramm Öffentlichen Kalender ansehen

Abbildung 3.9 zeigt die Zusammstellung aller Veranstaltungen des Tages. Dabei

[2] Vergleiche Abschnitt 3.2.3 auf Seite 48.

wird dem `PublicDatebook` das aktuelle Datum und der angemeldete Benutzer übergeben. Zunächst wird die Rolle des Benutzers bestimmt anhand welcher der Umfang des Ergebnisses ermittelt wird. So wird ein Interessent der sich über den Gastzugang angemeldet hat weniger Veranstaltungen erhalten, als ein Benutzer der zurzeit im Ferienpark seinen Urlaub verbringt.

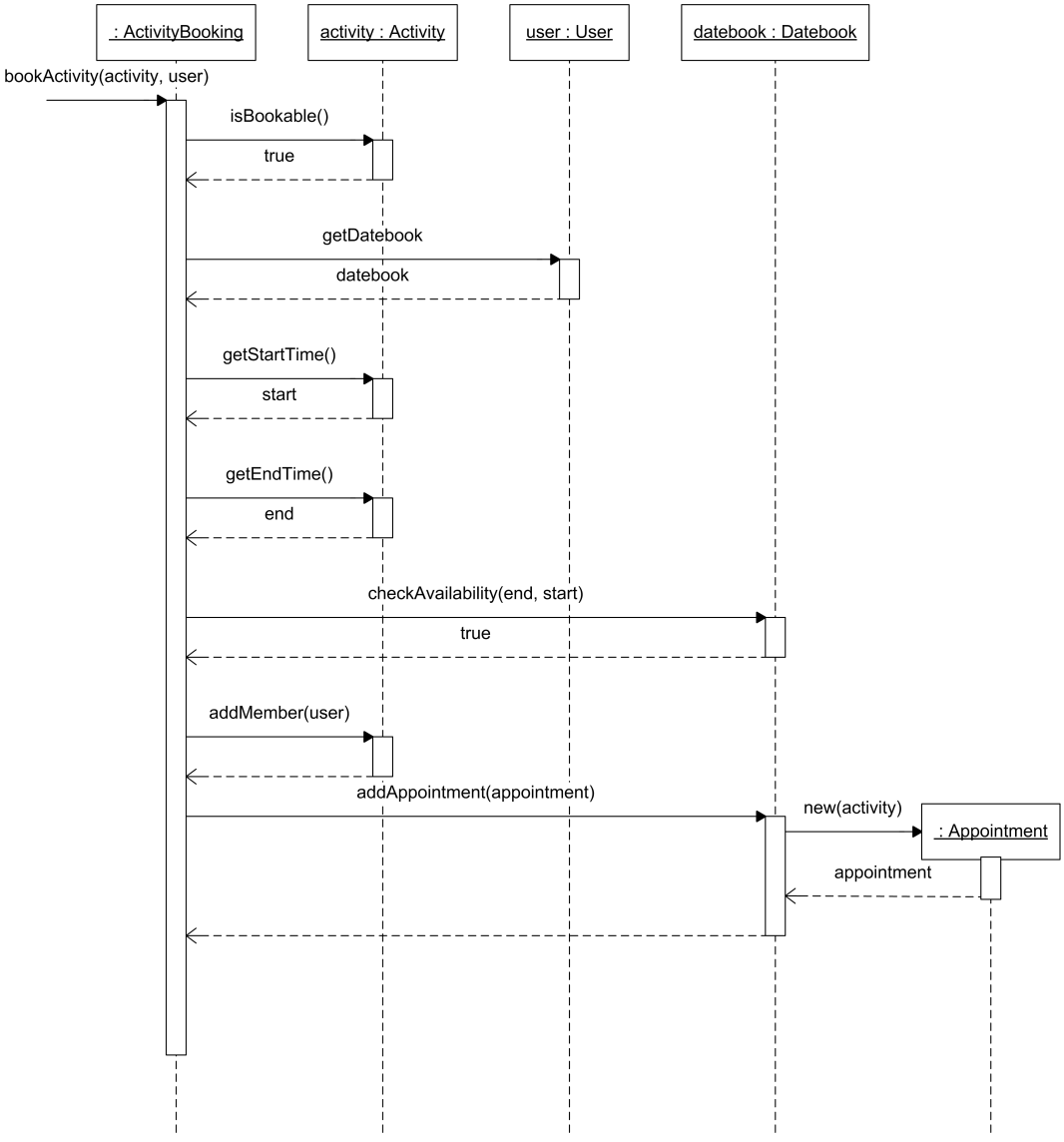



Abbildung 3.10: Sequenzdiagramm Veranstaltung buchen

Der Vorgang des Buchens einer Veranstaltung wird im Sequenzdiagramm in Abbil-

dung 3.10 veranschaulicht. Der Klasse `ActivityBooking` wird bei einer Buchungsanfrage die gewünschte Veranstaltung und der Benutzer übergeben. Vom System wird als Erstes überprüft, ob noch Plätze verfügbar sind, die Veranstaltung also noch über freie Plätze verfügt. Mit Hilfe des persönlichen Kalenders des Benutzers wird ermittelt, ob der Benutzer in der vorgesehenen Zeit frei hat. Ist dies der Fall, so wird der Veranstaltung der Benutzer als ein weiterer Teilnehmer hinzugefügt. Schließlich wird die Veranstaltung dem persönlichem Kalender des Benutzers hinzugefügt.

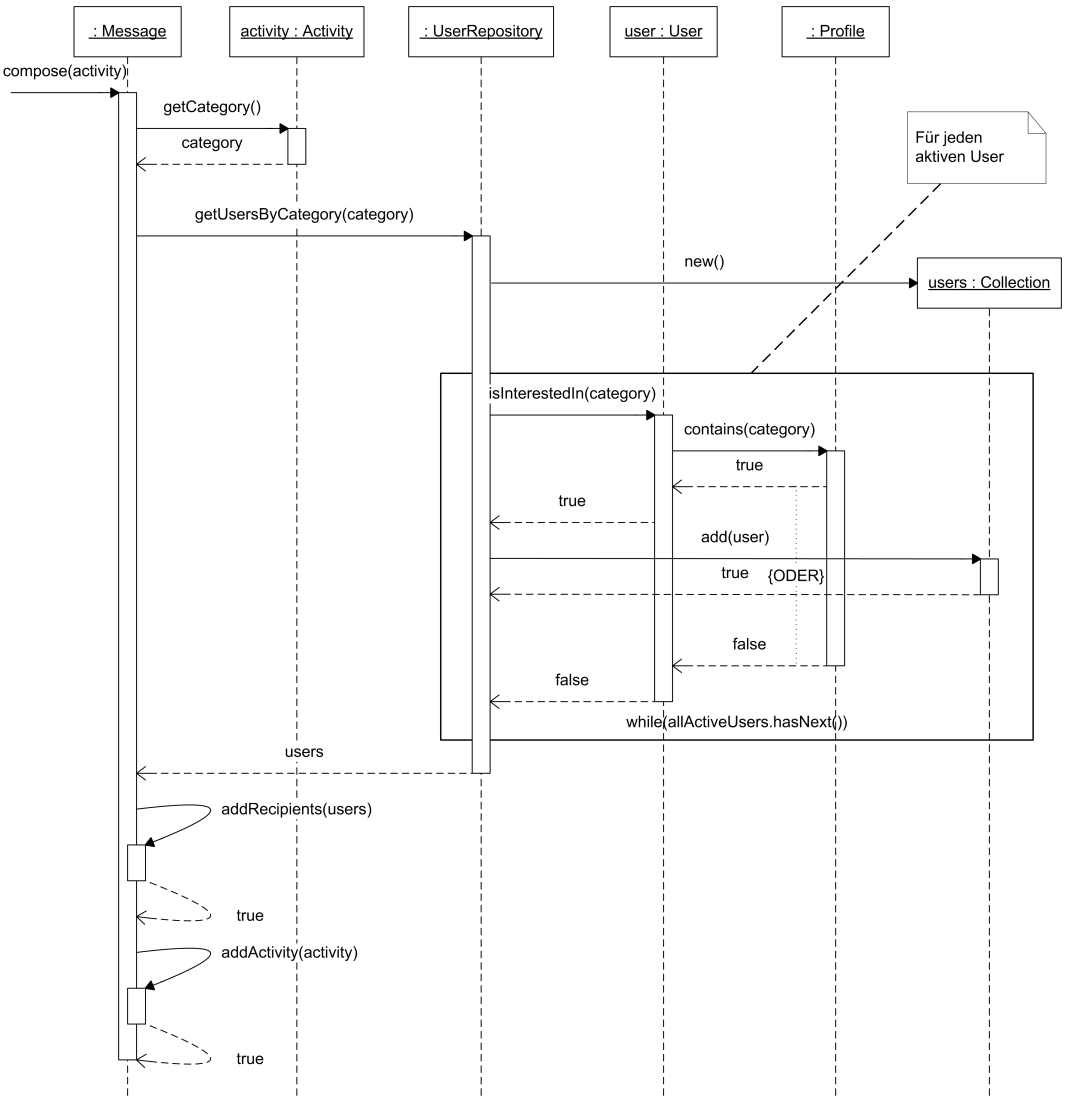



Abbildung 3.11: Sequenzdiagramm Mitteilung erfassen

Das Sequenzdiagramm aus Abbildung 3.11 skizziert das Erfassen einer Mitteilung, die im Zuge einer neu angelegten Veranstaltung generiert wird. Dazu wird zunächst die Kategorie der Veranstaltung bestimmt. Um allen Interessenten eine Mitteilung zukommen zu lassen, wird im `UserRepository` nach Benutzern gesucht, die sich für diese Kategorie interessieren. Dazu wird das Profil eines jeden aktiven Benutzers überprüft. Ist diese Kategorie im Profil vorhanden, wird der Benutzer als Empfänger hinzugefügt. Schließlich wird die Mitteilung noch um die Veranstaltung ergänzt.

### 3.2.5 Betrachtung des Sotfware-Entwurfs

In diesem Kapitel wurde anfänglich die Einteilung in Präsentation, Fachkonzept und zentraler Datenhaltung erläutert und deren Notwendigkeit dargelegt. Auf die Datenhaltung wurde aufgrund der prototypischen Implementierung nicht weiter eingegangen. Durch die Trennung mit Hilfe von klar definierten Schnittstellen kann später die einfache Datenhaltung auf Basis des Dateisystems durch eine Datenbank ersetzt werden.

Außerdem wurden die Zuständigkeiten der einzelnen Klassen identifiziert. Ihren Aufgaben entsprechend wurden die Klassen in Komponenten zusammengefasst. Daraus entstand ein Komponentenmodell, welches durch die Anwendung des Fassadenmusters über schlanke Schnittstellen verfügt. Ausführlich wurde auch die Interaktion zwischen den Klassen anhand von Sequenzdiagrammen dargestellt.

Der so erarbeitete Entwurf erfüllt aufgrund seines modularen Aufbaus und der Realisierung mittels Java die in der Analyse gestellten Forderungen nach einem wartbaren, erweiterbaren und portablen System.

Nach dem bisher das Fachkonzept betrachtet wurde, soll im Folgenden beleuchtet werden, wie das erarbeitete Fachkonzept in die Gesamtarchitektur einzubetten ist.

## 3.3 Model View Controller

Die bis hier gemachten Überlegungen beschäftigten sich in erster Line mit dem Fachkonzept. Benötigt wird ein Verfahren, um den Zugriff unterschiedlich gearteter Clients zu ermöglichen. So ist es vorstellbar, dass für den einen Client HTML generiert wird, während für den anderen Client WML erzeugt wird. Um der geforderten Wartbarkeit und Erweiterbarkeit nachzukommen, wurde am Anfang des Kapitels das Schichtenmodell eingeführt, welches eine Trennung zwischen Fachkonzept und Präsentation vorschreibt. Der Ursprung dieser Trennung findet sich im Model-View-Controller Entwurfsmuster wieder.

Um eine Unabhängigkeit der Clients zu realisieren, bedient man sich dem Model-View-Controller (MVC) Entwurfsmuster. Diese Architektur ist der Ursprung für die Entkoppelung von Benutzeroberfläche und Fachkonzept und wird seit Anfang der

achtziger Jahre erfolgreich bei GUI[3]-Anwendungen eingesetzt. Der Erfolg besteht darin, dass das MVC Entwurfsmuster Sourcecode-Redundanz minimiert, die Programmsteuerung zentralisiert und die Anwendung dadurch wartbarer macht. So können Entwickler ihren Fähigkeiten entsprechend eingesetzt werden, wie zum Beispiel Programmierung der Benutzeroberfläche auf der einen Seite und Geschäftslogik auf der anderen Seite. Später werden deren Module durch klar definierte Schnittstellen zusammengeführt. Außerdem können durch MVC sowohl neue Datenquellen wie auch neue Benutzeroberflächen leicht hinzugefügt werden [Singh u. a. 2002, S. 94f].

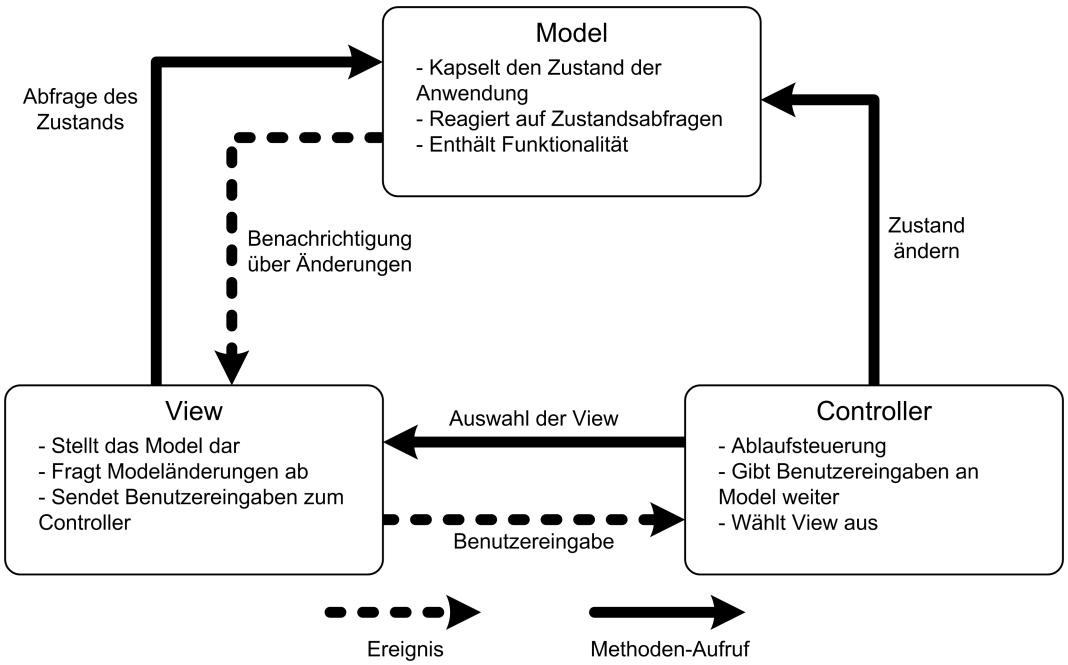



Abbildung 3.12: Model-View-Controller [Singh u. a. 2002, S. 349]

Das Model-View-Controller Entwurfsmuster setzt sich aus den drei Objekten *Model*, *View* und *Controller* zusammen (Abbildung 3.12). Das Model-Objekt stellt das Fachkonzept dar. Hier werden die Daten gespeichert und die Art und Weise festgelegt, wie mit den Daten umgegangen wird. Mit Hilfe des View-Objektes werden die Daten dargestellt. Dabei ist es von hoher Wichtigkeit, dass das View-Objekt ausschließlich zur Darstellung dient und keinerlei Logik enthält. Das View-Objekt greift dabei lesend auf die öffentliche Schnittstelle des Model-Objektes zu. Das Controller-Objekt steuert die Anwendung, es enthält also die Logik. Hier wird zum Beispiel auf Benutzereingaben reagiert. Wird in der Ansicht (View) etwas verändert, so prüft der Controller ob es zulässig ist und ändert gegebenenfalls die Daten im Model.

[3] Graphical User Interface: Grafische Benutzeroberfläche

Da es mehrere Möglichkeiten geben kann Daten zu repräsentieren, kann es auch mehrere View-Objekte geben, wobei jedes View-Objekt genau ein Controller-Objekt besitzt [Balzert 1999, S. 373f, 544] [Gamma u. a. 2000, S. 4f]. So können für mehrere Clients unterschiedliche View-Objekte zur Verfügung gestellt werden.

## 3.4 Java-Architektur und MVC

Im Folgenden wird gezeigt wie im Java Umfeld das Model-View-Controller Entwurfsmuster bei Webanwendungen umgesetzt wird.

### 3.4.1 Servlets und JavaServer Pages

Im Bereich der dynamischen Web-Anwendungen hat Sun Microsystems bisher Servlets und JavaServer Pages (JSP) angeboten. Dabei beinhaltet der Web-Server einen so genannten Servlet-Container, der neben den statischen HTML-Seiten Servlets ausführen kann und somit dynamischen Inhalt zur Verfügung stellt. Der Benutzer stellt eine Anfrage an ein Servlet, welches HTML-Code dynamisch generiert und an den Benutzer zurück schickt. Der Benutzer merkt keinen Unterschied zwischen der statischen HTML-Seite und dem Resultat des Servlets, da beides gleichermaßen von einem Web-Browser dargestellt wird.

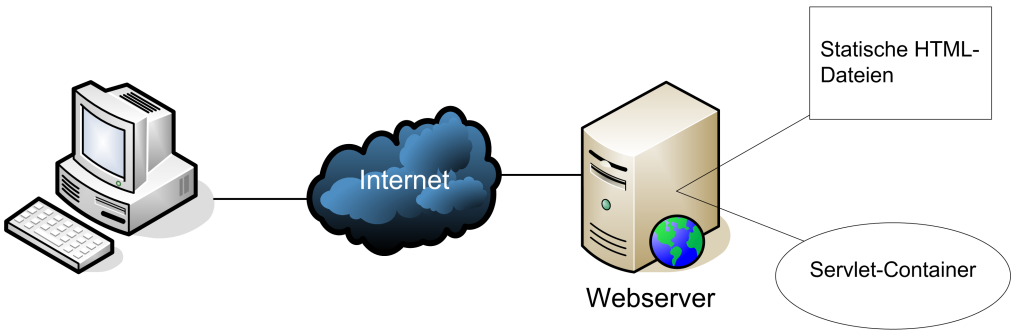



Abbildung 3.13: Webserver mit Servlet-Container

Bei einem Servlet handelt es sich um eine Java-Klasse die bestimmte Schnittstellen zur Verfügung stellt, um auf Anfragen aus dem Internet zu reagieren. Dabei wird neben der Logik, wie zum Beispiel die Auswertung eines Formulares, der HTML-Code in der Java-Klasse erzeugt. Hier sind View und Controller nicht voneinander getrennt, sondern in einer Klasse miteinander vermischt. Die Vorzüge vom Model-View-Controller Entwurfsmuster sind hier also nicht gegeben.

Um HTML-Code nicht ausschließlich im Servlet erzeugen zu müssen, hat Sun Microsystems kurze Zeit später die JavaServer Pages zur Verfügung gestellt. JavaServer

```
protected void doGet(
    HttpServletRequest req,
    HttpServletResponse res)
     throws ServletException,
         IOException
{
  res.setContentType("text/html");
  PrintWriter pw = res.getWriter();
  Date date = new Date();
  DateFormat df =
    DateFormat.getDateInstance();
  String strDate = df.format(date);
  String str =
    "Dynamischer Inhalt vom ";
  pw.println("<html><body><p>");
  pw.print(str);
  pw.print(strDate);
  pw.println("</p></body></html>");
}
```

Listing 3.1: Auszug aus einem Servlet


Abbildung 3.14: Ausgabe des Servlets

Pages sind mit Java-Code vermischte HTML-Seiten. Dabei wird der HTML-Code wie bei einer statischen HTML-Seite in die Datei geschrieben. Soll zum Beispiel eine Tabelle dargestellt werden die Daten aus einer Datenbank bezieht, so wird hier Java zu Hilfe genommen. Ein Vorteil gegenüber dem Servlet besteht darin, dass Änderungen am Design erleichtert werden. Allerdings sind auch hier nach wie vor View und Controller miteinander vermischt.

```
<html><body>
<p>
<%
  Date date = new Date();
  DateFormat df =
    DateFormat.getDateInstance();
  String strDate = df.format(date);
%>
 Dynamischer Inhalt vom <%= strDate %>
</p>
</body></html>
```

Listing 3.2: Auszug aus einer JavaServer Page

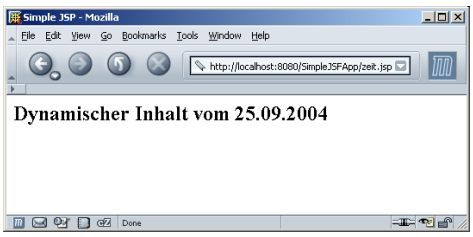



Abbildung 3.15: Ausgabe der JSP

Im Zusammenhang mit JavaServer Pages und Servlets ist häufig von *Model 1* und *Model 2* die Rede. Dabei wird die Model 1 Architektur für einfache und kleine Web-Anwendungen verwendet, während die Model 2 Architektur bei größeren Web-

Anwendungen eingesetzt wird [Seshadri 1999; Bosch 2004].

### JSP Model 1 Architektur

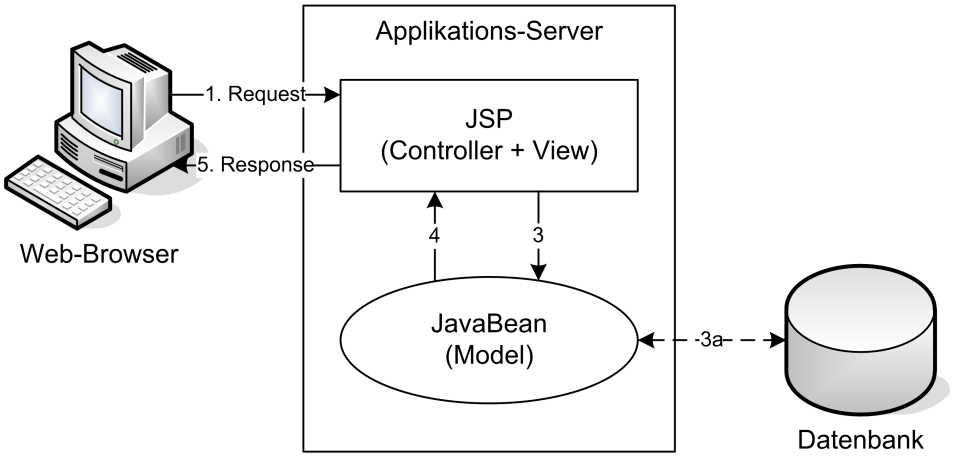



Abbildung 3.16: JSP Model 1 Architektur

In der Model 1 Architektur wird die JSP-Seite direkt aufgerufen, die wiederum auf eine JavaBean zugreift, welche zur Datenhaltung (Model) bestimmt ist. Die JSP hingegen ist sowohl für die Darstellung (View), als auch für die Ablaufsteuerung (Controller) zuständig. Da eine Web-Anwendung nach Model 1 aus mehreren JSP-Seiten besteht ist die Ablaufsteuerung dezentralisiert. Jede einzelne JSP ist für die eigene Navigation zuständig.

Der Vorteil dieser Architektur besteht in seiner Einfachheit. Die Web-Anwendung kann schnell implementiert werden, da keine zusätzlichen Verwaltungskomponenten entwickelt werden müssen. Allerdings verfügt die Model 1 Architektur über gravierende Nachteile. Da die Navigation über alle Seiten verteilt ist, muss bei einer Änderung des Ablaufes unter Umständen jede Seite korrigiert werden. Darüber hinaus ist der Java-Sourcecode einer JSP-Seite nicht wiederverwendbar und aufgrund der Vermischung von HTML- und Java-Code unübersichtlich [Bosch 2004].

### JSP Model 2 Architektur

Im Gegensatz zur Model 1 Architektur verfügt die Model 2 Architektur über ein Controller Servlet. Dadurch steht eine Architektur zur Verfügung, die dem Model-View-Controller Entwurfsmuster entspricht. Das Controller Servlet übernimmt dabei, seinem Namen entsprechend, die Rolle des Controllers. Die JavaBean ist für die Datenhaltung (Model) zuständig und die JSP bildet die Darstellung (View) ab. Hier

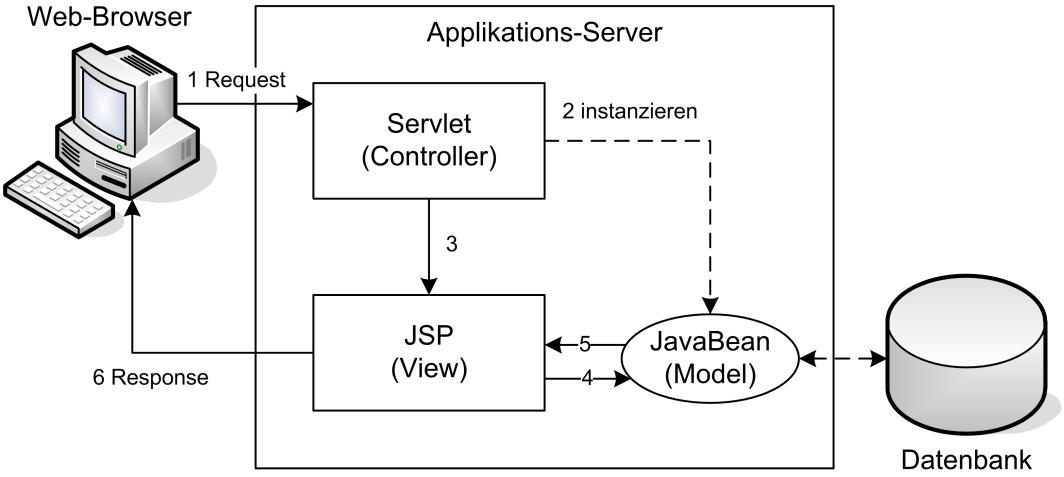



Abbildung 3.17: JSP Model 2 Architektur

wird also auf das Controller Servlet zugegriffen, welches sich um die Ablaufsteuerung und um die gegebenenfalls notwendige Initialisierung von JavaBeans kümmert. Das Servlet sucht außerdem die richtige View, sprich JSP-Seite, aus. Die JSP-Seiten referenzieren sich also nicht mehr untereinander, wie bei der Model 1 Architektur.

Zwar gewinnt die Model 2 Architektur an Komplexität, jedoch finden sich bei dieser Architektur die geforderten Vorzüge, wie Wiederverwendbarkeit und Erweiterbarkeit wieder [Singh u. a. 2002].

### 3.4.2 Struts

Da JavaServer Pages und Servlets der einzige offizielle Standard im Bereich der Web-Anwendungen war, etablierten sich mehrere Web-Frameworks auf dem Markt. Diese alle aufzuführen, würde den Rahmen dieser Arbeit sprengen. Jedoch ist eines davon hervorzuheben, da es zum De-Facto-Standard reifte: Struts[4] [Burns 2004][Wang 2004].

Struts verwendet die Model 2 Architektur. Dabei übernimmt das Action Servlet zusammen mit den Action-Klassen die Funktion des Controllers. Die Action-Klassen steuern, der Controller-Aufgabe entsprechend, die Anwendung und wählen die entsprechende JSP-Seite (View) aus. Dabei wird die Zuordnung der JSP-Seite aus der Konfigurationsdatei geladen. Bei Struts wird die Navigation also zentral von den Action-Klassen übernommen, wodurch ein Austausch oder die Erweiterung erleichtert wird.

Bei Struts handelt es sich in erster Linie um ein Framework für Web-Anwendungen.

[4] http://struts.apache.org

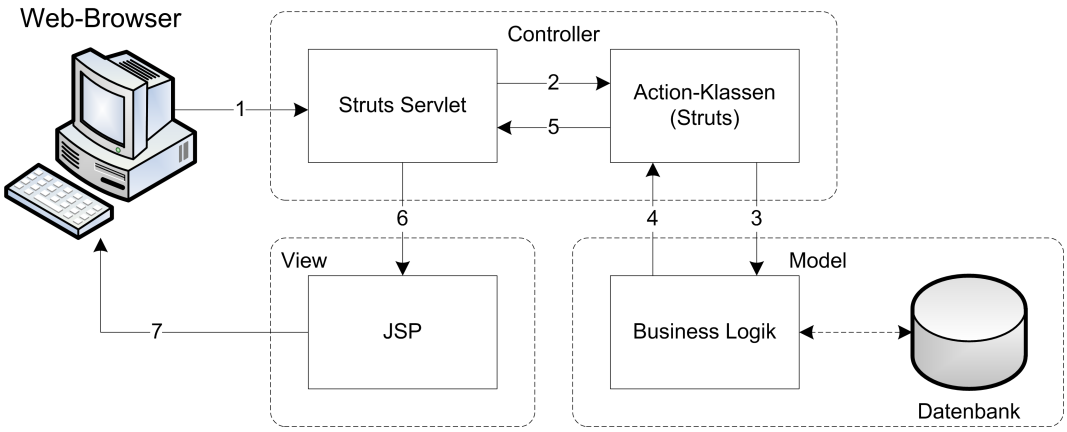



Abbildung 3.18: Struts und Model-View-Controller

Mit Hilfe von Struts wird dem Entwickler ein Hilfsmittel zur Verfügung gestellt, welches in erster Linie die Ablaufsteuerung unterstützt. Struts entscheidet anhand einer Benutzeranfrage und deren übermittelten Daten welche JSP-Seite aufzurufen ist. Dabei wird jedoch die Darstellung vom Framework nicht unterstützt. Für die Darstellung werden in der Regel JSP-Seiten genutzt. Die Art, wie beispielsweise eine Tabelle dargestellt wird, muss in der JSP-Seite definiert werden, da es nicht von Struts unterstützt wird [Bergsten 2004].

### 3.4.3 JavaServer Faces

Ende März 2004 hat Sun Microsystems das erste standardisierte Web-Framework spezifiziert – JavaServer Faces (JSF). JavaServer Faces basiert wie Struts auf der Model 2 Architektur. Jedoch handelt es sich hierbei nicht um ein Application Framework wie Struts, sondern in erster Linie um ein User Interface Framework. Dies bedeutet, dass sich die Funktionalitäten von JavaServer Faces und Struts im Bereich der Navigation durchaus überlappen. Wobei Struts und JavaServer Faces eine unterschiedliche Vorgehensweise wählen. Während Struts sich zunächst der Anwendungslogik bedient und am Ende der Bearbeitung das Ergebnis an die View weiter gibt, ruft JSF am Anfang der Bearbeitung die View und deren Komponenten auf, die ihrerseits Events auslösen. Die JSF Listener behandeln die anstehenden Events und steuern die Anwendung (siehe Abbildung 3.19).

### 3.4.4 Fazit

Die Model 1 JSP Architektur ist zwar für einfache und kleine Anwendungen empfehlenswert, da ihre simple Struktur eine schnelle Realisierung erlaubt, jedoch ist

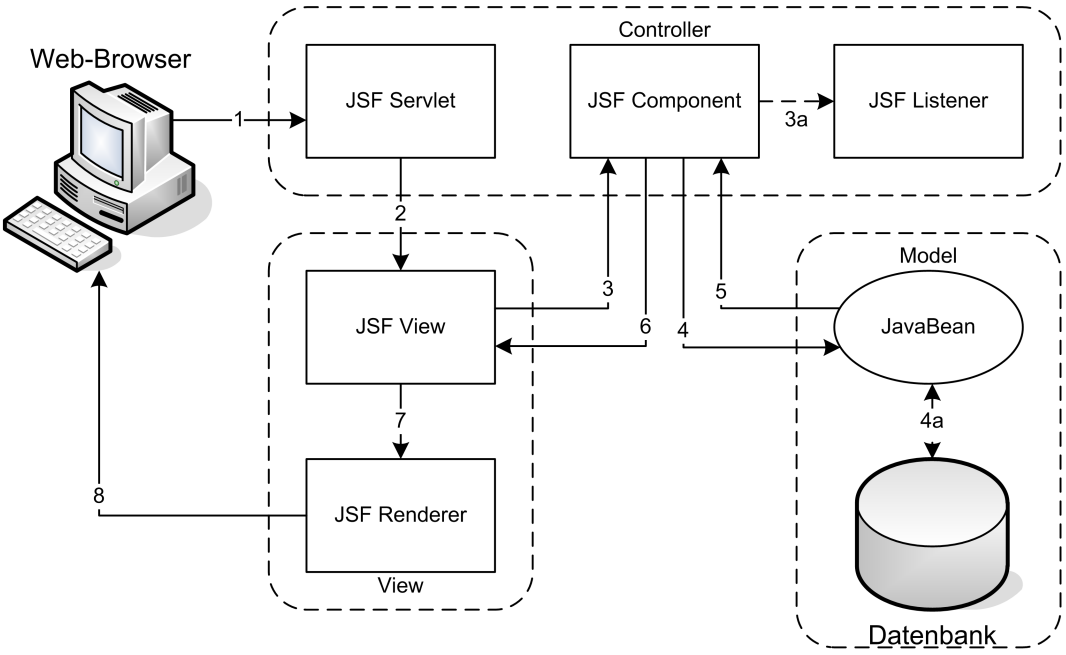



Abbildung 3.19: JavaServer Faces und Model-View-Controller

die Wiederverwendung des Java-Codes einer JSP-Seite nicht möglich. Beim Einsatz andersartiger Clients müsste für jeden Client eine redundante Implementierung der Ablaufsteuerung erfolgen, da die Navigation vollständig in der JSP-Seite enthalten ist. Bei einer Änderung des Seitenflusses müsste jede Clientart separat modifiziert werden. Die Anwendung ist somit nicht wartbar.

Für die hier diskutierten Anforderungen ist die Model 2 JSP Architektur unverzichtbar. Sowohl das Struts Framework, als auch das JavaServer Faces Framework haben die Model 2 Architektur als Grundlage. Beide Frameworks besitzen eine zentrale Navigation der Seiten. Möchte man jedoch eine Web-Anwendung für unterschiedlich geartete Clients realisieren, so ist das JavaServer Faces Framework dem Struts Framework vorzuziehen.

Einer der größten Vorteile von JavaServer Faces ist die Bedienung unterschiedlicher Clients. Da JavaServer Faces neben der Funktionalität rund um die Benutzerschnittstelle auch Mittel zur Ablaufsteuerung zur Verfügung stellt, ist dieses Framework für die Terminverwaltung die geeignetste Lösung. JavaServer Faces bietet die Möglichkeit, auf einfache Weise die Geschäftslogik unabhängig vom Client abzubilden. Außerdem stellt JavaServer Faces eine große Zahl an Standardvalidatoren zur Verfügung, die grundlegende Plausibilitätsprüfungen zulassen. So werden auch Mittel zur Fehlerbehandlung bereit gestellt, die eine einfache Ausgabe von Fehlermeldungen er-

möglichen. Ein weiterer Vorteil von JavaServer Faces, gerade im Zusammenhang mit einem Ferienclub, ist die Unterstützung von Mehrsprachigkeit. Auch hier stellt das JSF Framework einfache Mittel zur Verfügung, weitere Sprachen hinzuzufügen.

Somit wird im Folgenden die JavaServer Faces Technologie soweit vertieft und beschrieben, wie sie zum Verständnis der Arbeit notwendig ist.

# 4 JavaServer Faces

## 4.1 Überblick

Für dynamische Web-Seiten stellt Sun Microsystems JavaServer Pages (JSP) bzw. Servlets zur Verfügung. Eine JSP stellt eine Mischung aus statischem HTML-Code[1] und für den dynamischen Inhalt zuständigen Java-Code dar. Bei diesem Konzept modelliert der Web-Designer mittels HTML die Seiten, während der Entwickler im Anschluss daran die Funktionalität integriert. Diese Mischung von HTML- und Java-Code erfordert vom Entwickler HTML-Wissen, damit er an den entsprechenden Stellen die dynamische Funktionalität einbauen kann. Umgekehrt wird es für den Web-Designer unter Umständen schwierig, das Design im Nachhinein zu ändern, ohne den eingebetteten Java-Code zu zerstören. Des Weiteren wird innerhalb der JSP auch die Ablaufsteuerung vorgenommen, wie zum Beispiel die Fehlerbehandlung oder der Aufruf der nächsten Seite. Die JSP beinhalten also das Fachkonzept und die Benutzeroberfläche.

Der Java Community Process (JCP) hat sich im Java Specification Request (JSR) 127 dieser Problematik angenommen[2]. Angestrebt wurde ein Framework, welches die Benutzeroberfläche und das Fachkonzept trennt und sich nach dem Model-View-Controller-Konzept (MVC) verhält: das JavaServer Faces Framework.

Durch JavaServer Faces (JSF) wird es möglich die unterschiedlichen Aufgaben zu trennen. Sowohl die Darstellung (View), als auch die Navigation zwischen den einzelnen Seiten und die Fehlerbehandlung (Controller), werden getrennt voneinander behandelt. Dabei werden so genannte User-Interface Komponenten bereit gestellt, die unabhängig von ihrer Darstellung eine bestimmte Basisfunktionalität enthalten. Für die Ausgabe der Komponenten werden wiederum Render-Kits benötigt. JavaServer Faces stellt eine Auswahl von Standardkomponenten zur Verfügung, wie Ein- und Ausgabekomponenten, und ein Standard-Render-Kit für eine HTML-Ausgabe auf einem handelsüblichen Browser.

Darüber hinaus sieht das Framework Erweiterungen vor. So ist es möglich, eigene UI-Komponenten hinzuzufügen und auch eigene Render-Kits zu implementieren, die eine andere Ausgabe als HTML erzeugen, zum Beispiel WML[3] oder XML[4].

[1] HTML – Hypertext Markup Language
[2] Vgl. [jcp 2004a] und [jcp 2004b].
[3] WML – Wireless Markup Language
[4] XML – Extensible Markup Language

JavaServer Faces (JSF) ist ein User Interface Framework für Java Webapplikationen, die auf einem Java Application Server[5] laufen. Dabei werden die HTML-Seiten mittels JavaServer Pages (JSP) oder Servlets generiert und zum HTML-Client geschickt. JSF nimmt sich den allseits bekannten Problemen an, die bei Webapplikationen für HTML-Clients auftauchen. Damit sind zum Einen die browserspezifischen Probleme gemeint, wie zum Beispiel die unterschiedliche Darstellung des selben HTML-Codes, und zum anderen die Probleme, die aus dem zustandslosen HTTP-Protokoll entstehen. Denn typischerweise basieren diese Anwendungen auf sich über mehrere HTML-Seiten erstreckende Formulare, die dann auszuwerten sind. Das zustandslose HTTP-Protokoll bietet hier keine Möglichkeit die eingegebenen Daten der HTML-Seiten persistent zu halten. JSF unterstützt diese Webanwendungen durch folgende Eigenschaften [McClanahan u. a. 2004, Seite 1-7f]:

- Unterstützung der Verwaltung von Zuständen über mehrere Seiten bzw. Serveranfragen hinweg.
- Browserspezifische Generierung von HTML- oder WML-Code.
- Einfache Formularbearbeitung.
- Serverseitige Überprüfung der Formulardaten.
- Clientseitige Ereignisse können vom Server bearbeitet werden.
- Vereinfachte Migration der Anwendungsdaten zwischen Benutzeroberfläche und Server.
- Einheitliche Fehlerbehandlung mit für den Benutzer lesbaren Fehlermeldungen.
- Einheitliche Ereignisbehandlung.
- Vorhandene Standard-UI-Komponenten, die durch eigene UI-Komponenten erweitert werden können.

Somit bietet die JSF Technologie eine umfangreiche Architektur an, um den Zustand der UI-Komponenten zu verwalten, Daten der UI-Komponenten zu verarbeiten, Benutzereingaben zu validieren und Ereignisse zu behandeln [Sun 2003, S. 3].

Wie schon in Abschnitt 3.4.3 erwähnt, ist eines der größten Vorteile von JSF die Trennung von Präsentation und Fachkonzept – in „gewöhnlichen“ Java Webapplikationen enthält eine JSP sowohl die Präsentation, als auch die Logik. Durch diese Trennung können Entwickler mit unterschiedlichen Fähigkeiten gezielt eingesetzt werden. So kann beispielsweise ein Web-Designer ohne Java-Kenntnisse mit

[5] zum Beispiel JBoss (http://www.jboss.org) oder Tomcat (http://jakarta.apache.org/tomcat/index.html)

Hilfe der JSP custom tag library von JSF auf UI-Komponenten und Validatoren zugreifen. Die eingesetzten UI-Komponenten und Validatoren werden wiederum vom Anwendungsentwickler mit Java-Kenntnissen implementiert. Des Weiteren braucht der Web-Designer sich nicht um browserspezifische Eigenheiten kümmern, da durch JSF der für den Browser richtige HTML-Code zurückgegeben wird.

Im Grunde nutzt eine JSF-Applikation die gleiche Technik wie eine Java Webapplikation. Die JSF-Anwendung läuft in einem Servlet-Container der folgende Komponenten enthält [Amstrong u. a. 2004, S. 652]:

- JavaBeans für Anwendungsdaten und -funktionalität
- Event listeners
- JavaServer Pages
- Serverseitige Hilfsklassen für Datenbankzugriffe, etc.

Darüber hinaus enthält eine JSF-Anwendung [Amstrong u. a. 2004, S. 652]:

- JSP custom tag library für UI-Komponenten
- JSP custom tag library für Ereignis-, Fehlerbehandlung, Validatoren etc.
- UI-Komponenten zur Zustandsverwaltung auf dem Server
- Validatoren, Event Handler und Navigation Handler
- Eine Konfigurationsdatei zur Konfiguration der JSF-Anwendung

JSF ist derzeit[6] in der Version 1.1_01 verfügbar.

## 4.2 Aufgabenverteilung

Bei dem großen Bereich den JSF abdeckt sind unterschiedliche Entwicklertypen bzw. verschiedene Fähigkeiten von Entwicklern erforderlich. Das soll nicht heißen, dass zwingend mehrere Programmierer notwendig sind, um eine JSF-basierte Webapplikation zu implementieren. Vielmehr soll die Rollenverteilung die Möglichkeiten des JSF-Frameworks unterstreichen.

[6] Stand: September 2004, vgl. [Sun 2004]

### 4.2.1 Web-Designer

Der Web-Designer ist in erster Line für die Gestaltung der Web-Seiten zuständig. Dabei beherrscht er Auszeichnungssprachen wie HTML, aber auch Skriptsprachen wie JavaScript. Um die JSF-Technologie zu nutzen benutzt der Web-Designer die vorgefertigten UI-Komponenten aus der JSP custom tag library. Diese tag library stellt dem Web-Designer HTML-ähnliche Tags zur Verfügung. Dadurch kann er beispielsweise Validatoren einsetzen, ohne Java-Kenntnisse zu besitzen.

### 4.2.2 Komponentenentwickler

Komponentenentwickler implementieren Bibliotheken mit wiederverwendbaren UI-Komponenten. Er sollte sowohl Java-, als auch HTML- oder ggf. WML-Kenntnisse besitzen. Seine Aufgabe ist es, einen Konverter für Attribute und Eigenschaften der Komponente zu erstellen (encoding und decoding), aber auch entsprechende Validatoren zur Verfügung zu stellen. Darüber hinaus kümmert er sich um die richtige Darstellung der Komponenten auf den verschiedenen Clients (rendering).

Der Komponentenentwickler kann hierbei auch eine Integrationsrolle übernehmen. So ist es vorstellbar, dass ein Web-Designer die Client Representation der neuen Komponente implementiert, während ein Anwendungentwickler die serverseitige Funktionalität programmiert. Der Komponentenentwickler fügt diese dann zusammen.

### 4.2.3 Anwendungsentwickler

Der Anwendungsentwickler ist für die serverseitige Funktionalität zuständig. Dazu gehört die Geschäftslogik, wie auch die Datenhaltung – zum Beispiel mit Hilfe von Enterprise JavaBeans (EJB). Aber auch die Ablaufsteuerung der einzelnen Seiten wird vom Anwendungsentwickler modelliert. Hierfür muss der Anwendungsentwickler über Java-Kenntnisse verfügen.

## 4.3 Bearbeitung von Requests

Dieser Abschnitt soll zeigen, wie das JSF-Framework Serveranfragen bearbeitet. Dabei sind drei Szenarios möglich [McClanahan u. a. 2004, S. 2-1]:

- Non-Faces Request generiert Faces Response
- Faces Request generiert Faces Response
- Faces Request generiert Non-Faces Response

Die Begriffe haben folgende Bedeutung:

**Faces Response** ist eine Servlet-Antwort die vom JSF-Framework generiert wird.

**Non-Faces Response** beschreibt eine Servlet-Antwort die außerhalb des JSF-Frameworks entsteht. Das kann beispielsweise das Resultat einer JSP-Seite sein die keine JSF-Tags benutzt, oder eine Weiterleitung auf eine HTML-Seite.

**Faces Request** wird vom zuvor generierten Faces Response gesendet. Das ist der Fall, wenn von einer vorher generierten JSF-Seite ein Formular gesendet wird.

**Non-Faces Request** meint eine Anfrage die außerhalb der JSF-Anwendung entstanden ist. Das kann zum Beispiel der erste Aufruf der einer JSF-Seite sein.

Das häufigste Szenario wird das Zweite sein: Faces Request generiert Faces Response. Abbildung 4.1 zeigt die einzelnen Phasen, die in diesem Szenario durchlaufen werden.

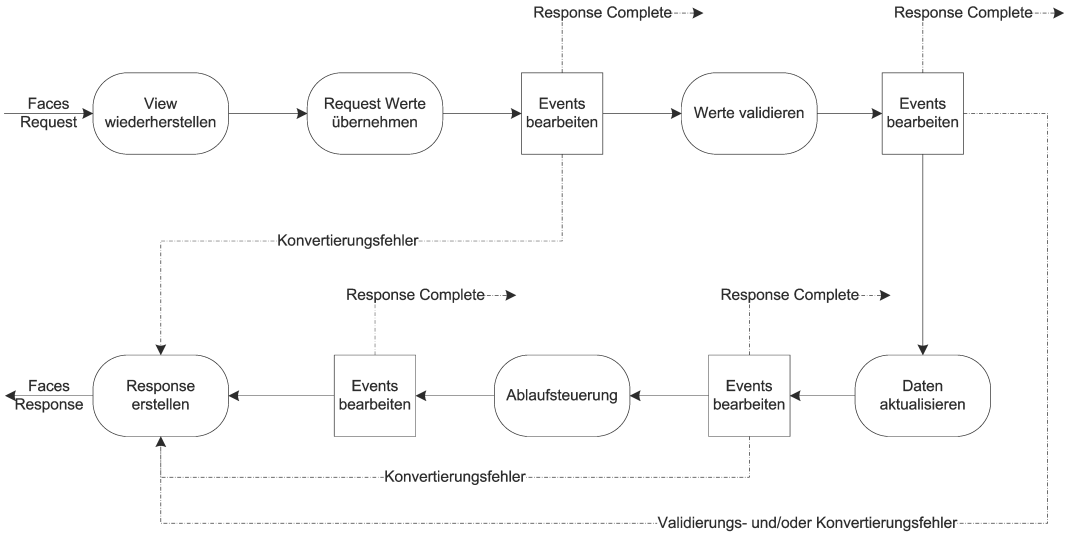



Abbildung 4.1: Bearbeitung von JavaServer Faces Requests [Amstrong u. a. 2004, S. 685]

Im Folgenden werden die einzelnen Phasen erläutert.

### View wiederherstellen

Zunächst wird der *Component Tree* rekonstruiert. Im *Component Tree* werden die UI-Komponenten einer JSF-Seite verwaltet. Dazu wird der Status vom Faces Request oder die gespeicherten Daten auf dem Server benötigt. Der *Component Tree* wird dynamisch aufgebaut und mit *Event Handler* und *Validatoren* verknüpft.

### Request Werte übernehmen

In dieser Phase wird jedem Knoten des *Component Tree* der neue Wert aus dem Request zugewiesen und einer syntaktischen Prüfung unterzogen. Der neue Wert ist dann lokal in der entsprechenden UI-Komponente gespeichert. Dazu ist gegebenenfalls eine Umformatierung oder Konvertierung des Wertes notwendig – zum Beispiel in einen anderen Datentyp. Schlägt die Konvertierung fehl, so wird eine entsprechende Nachricht erzeugt und in die Queue geschrieben. Bei unzulässigen Werten werden mindestens die Daten gespeichert, die eine Rekonstruktion der falschen Eingabe erlauben. Des Weiteren werden anstehende Ereignisse in die Queue geschrieben. Die Ereignisse werden gegebenenfalls vor der nächsten Phase bearbeitet.

### Werte validieren

Hier werden alle Validierungen aus dem *Component Tree* verarbeitet. Der lokal gespeicherte Wert wird auf Gültigkeit überprüft, zum Beispiel auf Minimal- und Maximalwert. Entspricht ein Wert nicht den Vorgaben, so wird eine entsprechende Nachricht in die Queue gestellt. Außerdem wird bei einer fehlerhaften Eingabe direkt zur Phase *Response erstellen* vorgerückt um die aktuelle Seite mit deren Fehlermeldungen anzeigen zu lassen.

### Daten aktualisieren

Nachdem die Daten konvertiert und validiert wurden, kann davon ausgegangen werden, dass die lokalen Daten in einer syntaktisch und semantisch korrekten Form vorliegen. Jetzt werden die Daten in die Datenhaltung, wie JavaBean oder Enterprise JavaBean, übernommen. Dabei werden ausschließlich die Daten von *Input-Komponenten* aktualisiert, die mit einer JavaBean, oder ähnlichem, verknüpft sind. Sollte hierbei ein Ereignis ausgelöst werden, weil die Daten zum Beispiel nicht gespeichert werden können, wird eine entsprechende Nachricht in die Queue gestellt und direkt zur Phase *Response erstellen* vorgerückt. Bei erfolgreicher Speicherung werden die lokalen Daten der UI-Komponenten gelöscht und mit der folgenden Phase fortgefahren.

### Ablaufsteuerung

Diese Phase behandelt alle Anwendungsereignisse, die zum Beispiel beim Senden eines Formulars oder beim Anklicken eines Links entstehen. Hier wird der Seitenfluss gesteuert und der entsprechende *Component Tree* mit seinen Werten aufgebaut.

#### Response erstellen

Der nun vorhandene *Component Tree* wird jetzt umgewandelt. Im Fehlerfall handelt es sich hierbei um den ursprünglichen *Component Tree*, der um die entsprechenden Fehlermeldungen erweitert wurde. Sonst ist es der von der Ablaufsteuerung neu erstellte *Component Tree*. Erzeugt wird ein für den Client verständlicher Code. Darüber hinaus werden die Daten des *Component Tree* in der HTTP-Session gespeichert, zum Beispiel in einem Cookie oder in versteckten Formularfeldern.

## 4.4 Ereignisbehandlung

JSF unterstützt drei Arten von Ereignissen. Dabei wird zwischen *value-changed events*, *action events* und *data-model events* unterschieden [Amstrong u. a. 2004, S. 672]:

**Value-changed events** ereignen sich, wenn ein Benutzer den Wert einer UI-Komponente verändert. *Value-changed events* werden nur ausgelöst wenn keine Validierungsfehler vorliegen.

**Action events** werden beim Anklicken eines Links oder eines Buttons ausgelöst.

**Data-model Events** sind den *action events* ähnlich. Das Ereignis wird bei der Auswahl einer Zeile einer *UIData Komponente* ausgelöst.

# 5 Realisierung

## 5.1 Implementierung der Terminverwaltung

Nach der Vorstellung des JavaServer Faces Frameworks und dessen Arbeitsweise soll nun anhand der Terminverwaltung die praktische Umsetzung erfolgen. Die Realisierung wird an einem kleinen Ausschnitt skizziert. Ein Benutzer kann sich an der Terminverwaltung anmelden und erhält nach erfolgreicher Anmeldung eine Tagesübersicht des persönlichen Kalenders. Für dieses Beispiel werden zwei JSP-Seiten benötigt (siehe Abbildung 5.1).

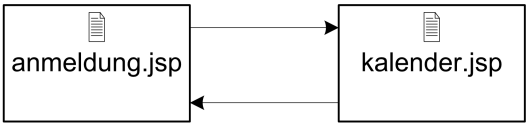



Abbildung 5.1: Webseiten des Prototypen

Um die JSF-Technologie innerhalb der JSP nutzen zu können, müssen zunächst die notwendigen JSP custom tag libraries geladen werden (Listing 5.1).

```
6  <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
7  <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
```

Listing 5.1: Einbinden der JSP custom tag libraries von JSF

Mit Hilfe der jetzt verfügbaren JSP-Tags können die UI-Komponenten von JSF genutzt werden. Ein Auszug aus `anmeldung.jsp` zeigt in Listing 5.2 die Verwendung von UI-Komponenten.

```
9     <f:view>
10      <h:graphicImage id="logo" url="logo3.gif" />
11        <h:form id="nameForm">
12          Benutzername:<br>
13          <h:inputText id="userName" value="#{UserBean.name}"
14              required="true" validator="#{UserBean.validateName}"/>
15          <br>
16          Passwort:<br>
17          <h:inputSecret id="userPass" value="#{UserBean.password}"
18              required="true" validator="#{UserBean.validatePassword}"
                    />
```

```
19            <br>
20            <h:commandButton id="submit" action="success" value="
                  Anmelden"/>
21            <p>
22          <h:messages style="color: red;" />
23        </h:form>
24      </f:view>
```

Listing 5.2: Auszug aus `anmeldung.jsp`

Beim Aufruf von `anmeldung.jsp` passiert folgendes: Der Browser stellt eine Anfrage an den Java Server. Dieser wandelt die JSP-Tags um und stellt dem Browser entsprechenden HTML-Code zur Verfügung (siehe Abbildung 5.2).

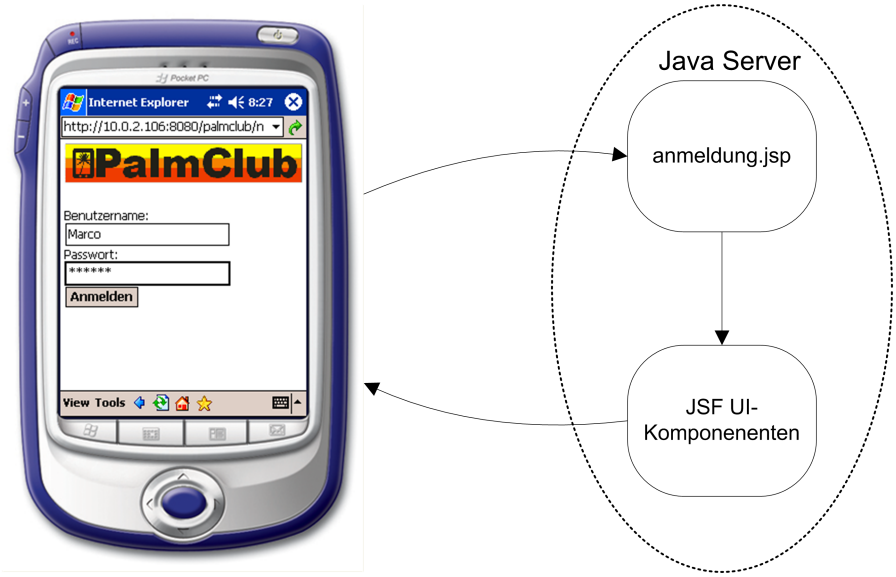



Abbildung 5.2: Anzeige von `anmeldung.jsp` auf dem PDA

Für die Datenhaltung muss eine JavaBean oder Enterprise JavaBean implementiert werden. Die JavaBean dient zur Datenhaltung über mehrere JSP-Seiten hinweg. Dabei werden die Daten in einer HTTP-Session gespeichert. Hier wurde eine JavaBean zur Verfügung gestellt, die in Zeile 13 in Listing 5.2 mit der UI-Komponente verknüpft wird. Darüber hinaus wird die JavaBean in der zentralen Konfigurationsdatei (`faces-config.xml`) der JSF-Anwendung angemeldet (siehe Listing 5.3).

```
5 <!DOCTYPE faces-config PUBLIC
6  "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
7  "http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
8
9 <faces-config>
```

```
10
11      <managed-bean>
12          <description>Holds informations about a user</description>
13          <managed-bean-name>UserBean</managed-bean-name>
14          <managed-bean-class>dipl.palmclub.UserManagment.User</managed-
                bean-class>
15          <managed-bean-scope>session</managed-bean-scope>
16      </managed-bean>

    ⋮

38  </faces-config>
```

Listing 5.3: Auszug aus `faces-config.xml`

Neben den JavaBeans wird in der `faces-config.xml` auch der Ablauf einer JSF-Anwendung definiert. Dabei wird festgelegt welche Seiten wie erreichbar sind. Beim Anklicken eines Buttons oder eines Links wird ein Ereignis ausgelöst, welches vom JSF Framework behandelt wird. In diesem Beispiel wird die Eingabe des Namens erwartet[1]. Wenn ein Name eingegeben und erfolgreich überprüft wurde, so wird die Seite `kalender.jsp` aufgerufen. Ist der Benutzername dagegen ungültig, dann wird wieder `anmeldung.jsp` aufgerufen und eine Fehlermeldung ausgegeben[2]. Listing 5.4 zeigt die Konfiguration der Ablaufsteuerung des Beispiels.

```
18      <navigation-rule>
19          <from-view-id>/anmeldung.jsp</from-view-id>
20          <navigation-case>
21              <from-outcome>success</from-outcome>
22              <to-view-id>/kalender.jsp</to-view-id>
23          </navigation-case>
24          <navigation-case>
25              <from-outcome>failure</from-outcome>
26              <to-view-id>/anmeldung.jsp</to-view-id>
27          </navigation-case>
28      </navigation-rule>
```

Listing 5.4: Auszug aus `faces-config.xml`

Um die JSF-Technologie in einer Webapplikation verwenden zu können muss sie mit Hilfe des *Web Application Deployment Descriptor* angemeldet werden. Dieser *Descriptor* ist ein XML-Dokument namens `web.xml`, in dem die Konfiguration der Ressourcen beschrieben werden (siehe Listing 5.5).

```
25    <!-- Faces Servlet -->
26    <servlet>
27      <servlet-name>Faces Servlet</servlet-name>
```

[1] vergleiche Listing 5.2 Zeile 14
[2] vergleiche Listing 5.2 Zeile 22

```
28      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
29      <load-on-startup> 1 </load-on-startup>
30    </servlet>
31
32    <!-- Faces Servlet Mapping -->
33    <!-- This mapping identifies a jsp page which includes JSF content.
          -->
34    <servlet-mapping>
35      <servlet-name>Faces Servlet</servlet-name>
36      <url-pattern>*.jsf</url-pattern>
37    </servlet-mapping>
```

Listing 5.5: Web Application Deployment Descriptor `web.xml`

Zunächst muss das *Faces Servlet* eingebunden werden, damit der Web-Container die JSF-Technologie verwenden kann. Das *Faces Servlet* nimmt die Serveranfragen entgegen und leitet sie an die geeignete JSF-Ablaufsteuerung weiter. Darüber hinaus muss dem Web-Container noch ein entsprechendes Servlet-Mapping mitgeteilt werden, damit das *Faces Servlet* angesprochen wird. Alle Anfragen, die hier mit `.jsf` enden, werden vom *Faces Servlet* bearbeitet.

## 5.2 Erweiterung des JavaServer Faces Framework

Im Zuge der Diplomarbeit wurde eine eigene UI-Komponente entwickelt. Diese Komponente ermöglicht die Tagesansicht des persönlichen Kalenders. Abbildung 5.3 zeigt die Darstellung der UI-Komponente auf dem PDA.

Die Komponente `UIDayView` ist von der Klasse `UIOutput` abgeleitet. Bei `UIOutput` handelt es sich um eine Ausgabekomponente von JavaServer Faces die lediglich der Darstellung von Werten dient. `UIDayView` ist somit für die Ausgabe des persönlichen Kalenders konzipiert. Eine Interaktion mit dem Benutzer findet nicht statt. Hierfür wäre eine weitere Komponente notwendig.

Die UI-Komponente wird aus der JSP-Seite mit Hilfe eines eigens dafür erstellten JSP-Tag angesprochen. Ein Blick in den Quellcode der JSP-Seite `kalender.jsp`, unterstreicht den einfachen Einsatz eigener UI-Komponenten (siehe Listing 5.6, Zeile 13).

```
9   <f:view>
10    <h:form id="datebookForm">
11      Hallo <h:outputText id="name" value="#{UserBean.name}"/>
12      <p>
13      <pda:dayView id="datebook" value="#{UserBean.datebook}"/>
14      <h:commandButton id="submit" action="success" value="Zurueck"/>
15    </h:form>
16  </f:view>
```

Listing 5.6: Einbinden der eigenen UI-Komponente


Abbildung 5.3: Eigene UI-Komponente `UIDayView`

Die Nutzung der Tagesansicht geschieht analog zur Eingabe des Benutzernamens und des Passwortes. Die UI-Komponenten `UIDayView` wird mit der JavaBean `User` verknüpft. Dadurch kann `UIDayView` auf den persönlichen Kalender des angemeldeten Benuzters zugreifen und dessen Inhalt auslesen.

Um dies zu realisieren musste zunächst eine eigene Tag Library erstellt werden, in welcher das Tag `dayView` mit seinen Attributen definiert, sowie dessen Tag Handler benannt wird. Letzterer legt unter anderem den Renderer Typ fest.

Zur Darstellung von `UIDayView` wird noch ein `DayViewRenderer` benötigt. Der Renderer ist für die Visualisierung zuständig und erzeugt einen für den PDA angepassten HTML-Code.

Mittels einer Konfigurationsdatei (`pda-config.xml`) wird sowohl `UIDayView`, als auch `DayViewRenderer` am JavaServer Faces Framework angemeldet (siehe Linsting 5.7).

```
8  <faces-config>
9      <component>
10         <component-type>DayView</component-type>
11         <component-class>dipl.pamlclub.jsf.UIDayView</component-class>
12         <component-extension>
```

```
13              <component-family>DayView</component-family>
14              <renderer-type>DayViewRenderer</renderer-type>
15          </component-extension>
16      </component>
17
18      <render-kit>
19          <renderer>
20              <component-family>DayView</component-family>
21              <renderer-type>DayViewRenderer</renderer-type>
22              <renderer-class>
23                  dipl.palmclub.jsf.DayViewRenderer
24              </renderer-class>
25          </renderer>
26      </render-kit>
27  </faces-config>
```

Listing 5.7: Konfiguration der eigenen UI-Komponente und dessen Renderer

## 5.3 Zusammenfassung

Es wurde ein Prototyp entwickelt, der das JavaServer Faces Framework nutzt. Dabei wurden neben den Standardkomponenten von JSF auch dessen Validatoren für einfache Überprüfungen verwendet. Für die Darstellung des persönlichen Kalenders wurde das JavaServer Faces Framework um eigene Komponenten und Renderer erweitert. Die Erweiterung nutzt hierfür die Mittel, die vom Framework zur Verfügung gestellt werden.

# 6 Fazit und Ausblick

In dieser Diplomarbeit wurde erfolgreich das JavaServer Faces Framework für den Einsatz in mobilen Anwendungen erprobt. Es ist möglich das Framework so zu erweitern, dass auf einfache Art und Weise mobile Endgeräte bedient werden können. Dabei sind keine Anpassungen am Fachkonzept notwendig. Das Framework ist so ausgelegt, dass auf Änderungen in der Präsentationsschicht flexibel reagiert werden kann.

Bei der Realisierung stellte sich heraus, dass es sich bei JavaServer Faces um ein umfangreiches Framework handelt, welches letztendlich mit einfachen Mitteln erweiterbar ist. Jedoch handelt es sich aufgrund der Komplexität um keine einfache Aufgabe, die Erweiterungen durchzuführen. Wie aus Kapitel 4.2 ersichtlich wird, gibt es bei der Realisierung einer JSF-Anwendung mehrere Aufgaben zu bewerkstelligen. Neben der Implementierung des Fachkonzepts ist auch die Gestaltung der Benutzeroberfläche gefragt. Dazu kommt die Konfiguration der JSF-Anwendung. Das erfordert ein breites Wissen über unterschiedliche Technologien die im JavaServer Faces Framework genutzt werden. Die Bandbreite erstreckt sich von Bereichen wie HTML und JSP für die Erstellung der Benutzeroberfläche, über XML zur Konfiguration, bis hin zu Java für die Entwicklung der JSF-Komponenten.

Wird man jedoch seinen Fähigkeiten entsprechend in einem JSF-Projekt eingesetzt, so sollte der Aufwand der Einarbeitung gering sein. Ein Web-Designer, der weiß wie man JSP-Tags benutzt, wird sich schnell in seinem Aufgabenbereich einarbeiten können. Eine der Stärken von JavaServer Faces ist die klare Trennung der Aufgaben. Einzig die Implementierung eines Renderers erfordert technologieübergreifendes Wissen.

Weiter fiel auf, dass die noch junge Technologie über eine hohe konzeptionelle Qualität verfügt. Hierbei konnte von den umfangreichen Erfahrungen des Struts-Projekt profitiert werden, welches durch das Mitwirken von Craig McClanahan, dem Struts Initiator, gefördert wurde. Aufgrund des durchdachten Konzepts und der daraus resultierenden Erweiterbarkeit, werden in naher Zukunft weitere JSF-Komponenten zur Verfügung stehen. Diese Entwicklung deutet sich bereits mit dem JSR 252 an [jcp 2004c].

Die implementierte Terminverwaltung kann in einer weiterführenden Betrachtung so fortgesetzt werden, dass sie auch von einem Handy aus erreichbar ist. Es könnte ein auf dem Handy vorhandener Wap-Browser für die Darstellung gewählt werden. Hierfür ist ein weiteres JavaServer Faces Render-Kit erforderlich, welches WML statt

HTML generiert. Für eine vollständige Funktionalität müsste dafür ein komplett neues Render-Kit implementiert werden, welches auch die Standardkomponenten von JSF in WML abbildet.

Weiterhin zu Untersuchen wäre das Zusammenspiel von JavaSever Faces und Struts[1]. Da es sehr viele Webanwendungen gibt, die mit Struts realisiert werden, könnte an dieser Stelle geprüft werden, inwiefern der Einsatz von JavaServer Faces zusammen mit Struts möglich ist und welche Vor- oder Nachteile sich daraus ergeben.

[1] Vergleiche http://cvs.apache.org/builds/jakarta-struts/nightly/struts-faces/ und http://struts.apache.org/faqs/kickstart.html#jsf

# Literaturverzeichnis


**Amstrong u. a. 2004** Amstrong, Eric ; Ball, Jeniffer ; Bodoff, Stephanie ; Carson, Debie ; Evans, Ian ; Green, Dale ; Haase, Kim ; Jendrock, Eric ; Sun Microsystems (Hrsg.): *The J2EE 1.4 Tutorial*. URL: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/J2EETutorial.pdf. März 2004. – Dateigröße: 15.028.092 Bytes 67, 69, 71

**Balzert 1999** Balzert, Heide: *Lehrbuch der Objektmodellierung*. Heidelberg, Berlin : Spektrum, Akademischer Verlag, 1999 (Lehrbücher der Informatik). – ISBN 3-8274-0285-9 46, 57

**Bergsten 2004** Bergsten, Hans: *JavaServer Faces*. Sebastopol : O'Reilly, April 2004 (First Edition). – ISBN 0-596-00539-3 61

**Bosch 2004** Bosch, Andy: *Java Server Faces - Das Standard-Framework zum Aufbau webbasierter Anwendungen*. München : Addison-Wesley, 2004 (Programmer's Choice). – ISBN 3-8273-2127-1 59

**Burns 2004** Burns, Ed ; Sun Microsystems (Hrsg.): *About Faces: The JavaServer Faces API and how it relates to Struts*. URL: https://javaserverfaces.dev.java.net/presentations/20040729-OJUG.pdf. Juli 2004 60

**CenterParcs 2004** CenterParcs N.V. (Hrsg.): *CenterParcs*. URL: http://www.centerparcs.com. April 2004 18

**Gamma u. a. 2000** Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Design Patterns – Elements of Reusable Object-Oriented Software*. 20. Auflage. Addison-Wesley, 2000. – ISBN 0-201-63361-2 44, 48, 57

**jcp 2004a** Java Community Process (Hrsg.): *The Java Community Process(SM) Program*. URL: http://www.jcp.org. Juli 2004 65

**jcp 2004b** Java Community Process (Hrsg.): *The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 127:*. URL: http://www.jcp.org/en/jsr/detail?id=127. Juli 2004 65

**jcp 2004c** JAVA COMMUNITY PROCESS (Hrsg.): *The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 127:*. URL: http://www.jcp.org/en/jsr/detail?id=252. September 2004 79

**Lüpke 2004** LÜPKE, Andre: *Entwurf einer Sicherheitsarchitektur für den Einsatz mobiler Endgeräte*. Berliner Tor 7, 20099 Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, April 2004. – URL: http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/luepke.pdf 13, 43

**McClanahan u. a. 2004** MCCLANAHAN, Craig ; BURNS, Ed ; KITAIN, Roger ; SUN MICROSYSTEMS (Hrsg.): *JavaServer Faces Specification, Version 1.1*. URL: http://java.sun.com/j2ee/javaserverfaces/download.html. February 2004 66, 68

**Robinson 2004a** ROBINSON CLUB GMBH (Hrsg.): *Zeit für Gefühle*. URL: http://www.robinson.de im Bereich Aktiv, Clubs, Sport, Biken. April 2004 16

**Robinson 2004b** ROBINSON CLUB GMBH (Hrsg.): *Zeit für Gefühle*. URL: http://www.robinson.de im Bereich Aktiv, Clubs, Sport, Adventure. April 2004 19

**Seshadri 1999** SESHADRI, Govind ; JAVAWORLD.COM (Hrsg.): *Understanding JavaServer Pages Model 2 architecture – Exploring the MVC design pattern*. URL: http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc_p.html. Dezember 1999 59

**Singh u. a. 2002** SINGH, Inderjeet ; STEARNS, Beth ; JOHNSON, Mark ; THE ENTERPRISE TEAM: *Designing Enterprise Applications with the J2EE Platform, Second Edition*. Addison-Wesley, 2002. – ISBN 0-201-78790-3 56, 60

**Sun 2003** SUN MICROSYSTEMS (Hrsg.): *JavaServer Faces Technology Tutorial*. URL: http://java.sun.com/j2ee/javaserverfaces/docs/JSF.pdf. März 2003. – Dateigröße: 710.683 Bytes 66

**Sun 2004** SUN MICROSYSTEMS (Hrsg.): *JavaServer Faces*. URL: http://java.sun.com/j2ee/javaserverfaces/. September 2004. – Dateigröße: 22.593 Bytes 67

**Wang 2004** WANG, Dapeng: Bunte Rahmen – Überblick über Web-Frameworks. In: *Java Magazin* 9.2004 (2004), August, S. 36–44 60

## Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 8. Oktober 2004

Ort, Datum Unterschrift