Softwarearchitektur im jZeis Thin-Client

Christoph Hohmann

6. September 2002

Inhaltsverzeichnis

1 Einleitung **3**
- 1.1 Motivation . . . 3
- 1.2 Zielsetzung . . . 3
- 1.3 Leserkreis . . . 4
- 1.4 Gegenstandsbereich . . . 4
- 1.5 Aufbau der Arbeit . . . 4

2 Systemanalyse **6**
- 2.1 Szenarien . . . 6
  - 2.1.1 Überblick über die Arbeitsweise eines Anwenders mit ZEIS . 6
  - 2.1.2 Überblick über die Arbeitsweise eines Administrators mit ZEIS 6
- 2.2 UseCases . . . 7
  - 2.2.1 Anmelden . . . 7
  - 2.2.2 Ist-Zeit erfassen . . . 7
  - 2.2.3 Report erstellen . . . 8
  - 2.2.4 Anwenderprofile bearbeiten . . . 8
  - 2.2.5 Projekte bearbeiten . . . 9

3 Vision **10**
- 3.1 Grundstruktur der Oberfläche . . . 10
- 3.2 Mitarbeiterverwaltung . . . 11
- 3.3 Projektbearbeitung . . . 11
- 3.4 Zeiterfassung . . . 13
- 3.5 Berichtanzeige . . . 14

4 Jakarta Struts **16**
- 4.1 Hintergrund von Struts . . . 16
- 4.2 Model-View-Controller Design Pattern . . . 16
- 4.3 Überblick über Struts . . . 17
- 4.4 Die Struts Konfigurationsdatei . . . 17
  - 4.4.1 Form-Beans . . . 17
  - 4.4.2 Global-Forwards . . . 18
  - 4.4.3 Action-Mappings . . . 18
- 4.5 Arbeitsweise des Controllers . . . 19
- 4.6 Struts Beispiel . . . 20

5 Die jZeis Services **25**
- 5.1 Anbindung der Services . . . 25
- 5.2 Überblick über die Materialien . . . 25
- 5.3 Überblick über die Services . . . 27

6 Realisierung **30**
- 6.1 Login . . . 30
- 6.2 Menübereich . . . 31
- 6.3 Zeiterfassung, Projektbearbeitung und Mitarbeiterverwaltung . . . 31
- 6.4 Berichtsanzeige . . . 32

7 Auswertung **34**
- 7.1 Zusammenfassung . . . 34
- 7.2 Ausblick . . . 34

Abbildungsverzeichnis

1 Use Case Diagramm 7
2 Grundstruktur der Oberfläche 10
3 Auswahlbereich der Mitarbeiterverwaltung 11
4 Arbeitsbereich der Mitarbeiterverwaltung 12
5 Auswahlbereich der Projektbearbeitung 12
6 Arbeitsbereich der Projektbearbeitung 13
7 Auswahlbereich der Zeiterfassung 13
8 Arbeitsbereich der Zeiterfassung 14
9 Auswahlbereich der Berichtsanzeige 14
10 Arbeitsbereich der Berichtsanzeige 15
11 Model-View-Controller 16
12 UML Sequenz Diagramm der Request Verarbeitung 19
13 UML Klassendiagramm zum Material FileCard 25
14 UML Klassendiagramm zum Material Activity 26
15 UML Klassendiagramm zum Material ProjectForm 26
16 UML Klassendiagramm zum Material TimeEntry 27
17 UML Sequenz Diagramm der LoginAction 30
18 UML Sequenz Diagramm der Action-Klassen des Auswahlbereiches von Zeiterfassung, Projektbearbeitung und Mitarbeiterverwaltung . 32
19 UML Sequenz Diagramm der Action-Klassen des Arbeitsbereiches von Zeiterfassung, Projektbearbeitung und Mitarbeiterverwaltung . 33

Tabellenverzeichnis

1 Gegenüberstellung der alten und neuen Berechtigungen 11
2 Interface des ZeisService 28
3 Interface des ProjectService 29

Literatur

[1] H. Züllighoven et al.: Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz, 1. Auflage (1998)

[2] M. Fayad et al.: Building Application Frameworks, 1. Auflage (1999)

[3] B. Kahlbrandt: Software-Engineering - objektorientierte Software-Entwicklung mit der Unified Modeling Language (1998)

[4] Jakarta Project: Struts Open Source Framework for building Web Applications, `http://jakarta.apache.org/struts/`

[5] Sun Microsystems: Java 2 Enterprise Edition Technologies Documentation, `http://java.sun.com/j2ee/docs.html`

1 Einleitung

1.1 Motivation

Durch die zunehmende Vernetzung werden immer mehr Dienstleistungen über das Internet zugänglich gemacht. Dabei zeigt sich derzeit ein Trend, diese Dienstleistungen immer mehr Menschen über die unterschiedlichsten Zugriffsmöglichkeiten zugänglich zu machen. Der Nutzer eines Dienstes soll in die Lage versetzt werden, den Dienst jederzeit von überall und mit jedem zur Verfügung stehenden Gerät nutzen zu köennen. Das beste Beispiel hierfür ist derzeit wahrscheinlich das Banking. Hier werden eine Vielzahl von verschiedenen Möglichkeiten angeboten. Es gibt eigenständige Programme wie Quicken und Star-Money, die eine Vielzahl von Funktionen bieten, um seine Bankgeschäfte zu tätigen. Es ist ebenso möglich, dies mit Hilfe eines Webbrowsers auf der Homepage der Bank zu tun, oder telefonisch über einen interaktiven Telefondienst. Wer lieber persönlich zur Bank geht, kann auch dort den Automaten benutzen. Alle diese verschiedenen Möglichkeiten nutzen dabei den gleichen zu Grunde liegenden Dienst, der die Bankgeschäfte durchführt. Schließlich will man die Aufträge, die man über sein Programm eingegeben hat, auch auf der Weboberfläche der Bank wiederfinden und möglicherweise ändern, wenn einem das Programm gerade nicht zur Verfügung steht. In Zukunft werden sicherlich noch weitere Möglichkeiten weitere Verbreitung finden, wie WAP-Handys oder SMS-Dienste. Alle diese Zugriffsmöglichkeiten auf den Dienst unterliegen jedoch gewissen Einschränkungen, die man bei der Realisierung beachten muss. So hat zum Beispiel ein WAP-Handy nur wenig Platz auf dem Display, um Texte anzuzeigen oder Benutzereingaben abzufragen. Web-basierte Oberflächen können Daten nicht direkt bei der Eingabe überprüfen, sondern immer erst, nachdem ein ausgefülltes Formular abgeschickt wurde. Die mächtigsten Vertreter unter den Dienstnutzungsmöglichkeiten sind daher sicher die eigenständigen Programme, denn sie unterliegen kaum Beschränkungen. In Zukunft wird es daher eine große Aufgabe werden, immer neue Zugangsmöglichkeiten zu Diensten anzubieten und diese besonders auf die vorhandenen Einschränkungen hin zu optimieren.

Ein weiterer Trend in der Informatik ist es, Rahmenwerke zu verwenden. Rahmenwerke helfen, große Projekte durch eine bereits vorhandene Struktur und Vorimplementierung möglichst vieler Funktionen, schnell und einfach zu entwerfen und zu realisieren. Da Rahmenwerke in verschiedensten Projekten eingesetzt werden, sind sie außerdem gut getestet. Durch den Einsatz eines Rahmenwerkes können also schon viele Fehler ausgeschlossen werden, die bei einer eigenen Implementierung von Projekten auftreten könnten. Außerdem setzen Rahmenwerke meistens auf bestimmten Paradigmen auf, die sich bereits im Einsatz bewährt haben, und führen somit auch zu einer ordentlichen Strukturierung des Projektes, was mit ihrer Hilfe realisiert werden soll. Dies macht es auch anderen Entwicklern später einfacher, die Programme anderer Leute zu verstehen und wenn dies nötig ist, zu verändern oder zu erweitern. Besonders Open-Source Projekte haben auf dem Gebiet der Rahmenwerke in letzter Zeit auf sich aufmerksam gemacht, und ihr Nutzen für die Entwicklung von Software-Projekten sollte nicht unterschätzt werden. Rahmenwerke haben natürlich auch einen wirtschaftlichen Vorteil, da sie Entwicklungszeit und damit auch Entwicklungskosten bei der Entwicklung neuer Software-Produkte reduzieren. Eingehendere Informationen zu Frameworks sind zu finden bei Fayad [2].

1.2 Zielsetzung

Ziel dieser Baccalaureatsarbeit ist es nun, genau so einen Dienstzugangspunkt, wie im letzten Abschnitt beschrieben, zu realisieren. Und zwar soll hier ein Web-Frontend für einen Arbeitszeiterfassungsdienst realisiert werden. Dieser steht bereits

in Form von Services zur Verfügung und muss durch die neue Oberfläche genutzt werden. Zur Realisierung der Oberfläche soll das Struts Framework (siehe dazu [4]) des Jakarta Projektes benutzt werden. Das Jakarta Projekt wird von der Apache Software Foundation unterstützt und entwickelt Open-Source Software Lösungen auf der Java Platform. Die Apache Software Foundation ist bekannt für die unterstützung hochwertiger Open-Source Projekte. Der für diese Arbeit verwendete Servlet Contailer "Tomcat" ist ebenfalls Teil des Jakarta Projektes.

Außerdem soll dem Leser ein ausreichender Einblick in Jakarta Struts gegeben werden, um ihm selber die Arbeit mit Struts zu ermöglichen, um Web-Applikationen zu entwickeln. Außerdem soll die Möglichkeit Struts mit bestehenden Services Architekturen zu nutzen ausreichend dagelegt werden, um die Möglichkeit der Entwicklung von Web-Frontends für andere Services zu erleichtern.

1.3 Leserkreis

Diese Studienarbeit richtet sich an Personen, die etwas über den Einsatz von Struts zur Realisierung von Webseiten und über die Möglichkeit der Anbindung von Services erfahren möchten. Das Verständnis von Java und objektorientierter Programmierung wird dabei vorausgesetzt. Ebenso wird vorausgesetzt, dass sich der Leser mit Java Servlets und JavaServer Pages (siehe dazu [5]) auskennt, da eine zustätzliche Einführung in diese Themen den Rahmen dieser Studienarbeit sprengen würde. Der in dieser Arbeit verwendete Service ist nach dem Werkzeug und Material Ansatz (siehe dazu [1]) konstruiert worden. In dieser Arbeit werden an verschiedenen Stellen Diagramme der Unified Modeling Language (UML) verwendet, eine Dokumentation hierzu ist unter anderem zu finden bei Kahlbrandt [3].

1.4 Gegenstandsbereich

Der Gegenstandsbereich dieser Studienarbeit ist eine kleine Firma mit etwa 30 Mitarbeitern. Die Mitarbeiter der Firma arbeiten in verschiedenen Projekten. Ein Mitarbeiter kann dabei auch in mehreren Projekten tätig sein. Die Mitarbeiter haben verschiedene Lohnklassen, nach denen sie bezahlt werden und die auch für die Kosten des Kunden entscheidend sind. Für die Abrechnung mit den Kunden und auch mit den Mitarbeitern müssen die Arbeitszeiten der Mitarbeiter erfasst werden. Für die Abrechnung mit den Kunden müssen außerdem die Arbeitszeiten auch den verschiedenen Projekten zugeordnet werden. Am Ende eines Monats soll es ersichtlich sein, wieviel jeder Mitarbeiter gearbeitet hat, um zu bestimmen, wieviel Zeit pro Mitarbeiter in den verschiedenen Projekten gearbeitet wurde, um die Kosten des Projektes zu bestimmen, da diese entweder mit dem Kunden abgerechnet werden müssen oder um zu sehen, wieviel vom Budget bei Festpreisaufträgen verbraucht wurde. Einige der Mitarbeiter sind Projektleiter, die die anderen Mitarbeiter den Projekten zuordnen.

1.5 Aufbau der Arbeit

Zu Beginn der Arbeit wird eine genauere Analyse des Gegenstandsbereichs mit Hilfe von Szenarien und UseCases durchgeführt, um die Arbeitsweise der beteiligten Personen eingehender zu beleuchten. Im folgenden wird dann eine Vision für das neue System entwickelt, die einen ersten Überblick über das gibt, was später als Web-Applikation dem Benutzer präsentiert werden soll. Dann wird genauer auf die technische Seite der Realisierung eingegangen. Dazu wird als erstes Struts genauer angesehen, um zu verstehen, wie Struts arbeitet und was man zu tun hat, um die geplante Web-Applikationen zu realisieren. Der zweite technische Aspekt ist der Service, der von der Web-Applikation genutzt werden soll. Der Service ist

nach dem Werkzeug- und Materialansatz (siehe [1]) konstruiert worden. Daher gibt es erst einen Überblick über die verwendeten Materialen und anschließend genauere Informationen zum Interface des Services. Danach folgt dann die tatsächliche Realisierung der neuen Web-Applikation mit Struts unter verwendung der beschriebenen Services. Abschließend folgt noch eine Zusammenfassung und ein Ausblick auf mögliche Erweiterungen oder Veränderungen.

2 Systemanalyse

Als Grundlage für die Entwicklung des Zeiterfassungssystems dient ein bereits vorhandenes System, das durch das neue System abgelöst werden soll. Dennoch soll der Umgang mit dem neuen System an das alte System angelehnt werden, um einen einfachen Wechsel für die Nutzer zu ermöglichen. Um dies zu ermöglichen, wurden Interviews mit den Nutzern des derzeitigen Systems durchgeführt, in denen sie den Umgang mit dem System auch vorgeführt haben, und daraus Szenarien und UseCases für den Umgang mit dem System erstellt. Das System unterscheidet die Nutzer nach normalen Anwendern und Administratoren. Die Rolle des Administrators schließt die Rolle der Anwender jedoch mit ein, so dass ein Szenario für die Anwender und ein Szenario für die zusätzlichen Möglichkeiten des Administrators erstellt wurde.

2.1 Szenarien

Nach Züllighoven [1] sind Szenarien eine Beschreiben der aktuelle Arbeitssituation. Dabei liegt das Augenmerk auf den Aufgaben im Anwendungsbereich und der Art und Weise, wie die jeweiligen Aufgaben unter Verwendung von Arbeitsmitteln und Arbeitsgegenständen erledigt werden. In diesem Fall also die Arbeitsweise der Anwender mit dem bisher verwendeten Zeiterfassungssystem.

2.1.1 Überblick über die Arbeitsweise eines Anwenders mit ZEIS

Viele der Mitarbeiter haben es sich angewöhnt, ihre Arbeitszeit erst auf einem Blatt Papier oder in einer Word-Datei aufzuschreiben, da sie erstens oft auch außerhalb arbeiten müssen und dort keinen Zugriff auf das Zeiterfassungssystem haben und zweitens der Umgang mit dem System als umständlich angesehen wird und sie den Einsatz des Systems auf ein Minimum reduzieren möchten. Soll die Arbeitszeit eingetragen werden, loggt sich der Benutzer zuerst im System mit seinem Namen und einem Kennwort ein. Daraufhin öffnet sich die Oberfläche des Programms, in dem die Arbeitszeiten eingegeben werden können. Dazu wird erst der entsprechende Tag ausgewählt, dann der Beginn der Aktivität und das Ende der Aktivität eingegeben und ein Projekt mit der dazu gehörigen Aktivität ausgewählt. Dann wird die Eingabe als neuer Zeiteintrag hinzugefügt. Dies wird mit allen Arbeitszeiten wierderholt.

Ist ein Monat vollständig ausgefüllt, dann erstellt der Anwender Berichte über seine Arbeitszeit. Von der Hauptoberfläche öffnet er dazu einen extra Dialog für Berichte. Er kann in diesem Dialog auswählen, für welchen Zeitraum er einen Bericht über seine Arbeitszeit erstellen möchte und kann die ausgegebenen Zeiteinträge auf ein bestimmts Projekt oder eine bestimmte Aktivität einschränken. Das System erstellt daraufhin einen druckfertigen Bericht und präsentiert diesen dem Anwender. Wenn er möchte, kann er den angezeigten Bericht daraufhin ausdrucken.

2.1.2 Überblick über die Arbeitsweise eines Administrators mit ZEIS

Ein Administrator hat zusätzlich zu den normalen Möglichkeiten eines Anwenders auch noch die Möglichkeit, neue Systemnutzer anzulegen oder die Profile der vorhandenen Systemnutzer zu verändern und Projekte anzulegen oder vorhandene Projekte zu verändern.

Vom der Hauptoberfläche aus öffnet der Administrator den Editor für die Mitarbeiter, Im Editor kann der Administrator für jeden Mitarbeiter die persönlichen Daten wie zum Beispiel den Namen eintragen. Außerdem kann er für jeden Monat die Soll-Arbeitszeit eingeben, die der Mitarbeiter erfüllen muss.

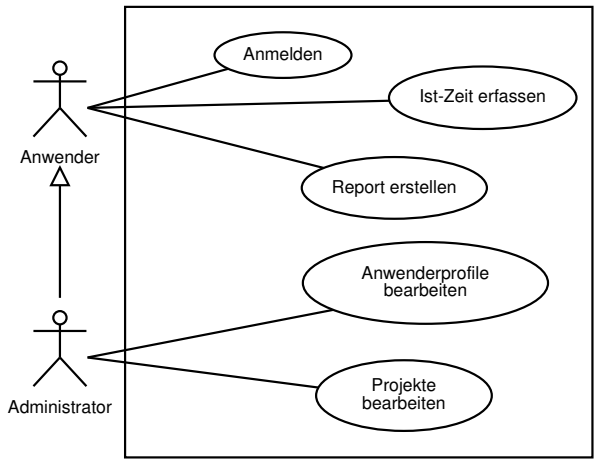
Im Editor für die Projekte kann der Administrator die genauen Daten für jedes Projekt eintragen und auch die möglichen Aktivitäten erstellen bzw. ändern, die

dem Projekt zugeordnet sind und in der Zeiterfassung von den Anwendern ausgewählt werden können.

2.2 UseCases

Da Szenarien doch eine recht unformelle Darstellung der Nutzungsweise des Systems sind, werden nun die einzelnen Handlungen noch einmal im Details als UseCases aufgeführt. In Abbildung 1 ist zunächst eine Übersicht der UseCases als UML-UseCase-Diagramm zu sehen.

Abbildung 1: Use Case Diagramm



2.2.1 Anmelden

Der UseCase "Anmelden" beschreibt die Anmeldung eines Benutzers am System. Das System muß den Benutzer bevor er etwas damit tun kann, erst authentifizieren, um sicherzustellen, dass auch nur die Funktionen genutzt werden können, für die der Benutzer eine Berechtigung hat.

Kurzbeschreibung	Anmeldung am ZEIS System
Beteiligte Akteure	Anwender
Vorbedingungen	Ein Anwenderprofil wurde für den Anwender erstellt
Nachbedingung	Der Anwender ist am System angemeldet
Ablaufbeschreibung	Der Akteur gibt Benutzername und Passwort ein und bestätigt seine Eingabe. Bei erfolgreicher Authentifikation öffnet sich die ZEIS-Oberfläche.

2.2.2 Ist-Zeit erfassen

Der UseCase "Ist-Zeit erfassen" beschreibt, wie die Mitarbeiter der Firma ihre Arbeitszeit im System eintragen oder bereits im System vorhandene Zeiteinträge ergänzen oder modifizieren.

Kurzbeschreibung	Der Akteur dokumentiert im ZEIS seine benötigte Zeit für eine geleistete Tätigkeit.
Beteiligte Akteure	Anwender
Vorbedingungen	Der Benutzer ist am System anmeldet
Nachbedingung	Keine
Ablaufbeschreibung	Der Akteur navigiert zu dem Arbeitstag, für den er einen Eintrag vornehmen möchte. Als Erstes trägt der Benutzer den Beginn und das Ende der Ist-Zeit ein. Danach wählt er das jeweilige Projekt sowie die Aktivität in dem Projekt aus. Zusätzlich kann noch eine Bemerkung eingegeben werden. Als Letztes wird der Eintrag gespeichert und erscheint in der Liste der Zeiteinträge.
Alternative Abläufe	Wenn ein Eintrag für die entsprechende Aktivität bereits vorhanden ist, kann der Akteur diesen aus der Liste der Zeiteinträge auswählen und zum Editieren aufrufen. Dann erscheinen die Daten des Eintrags in den Eingabefeldern, können daraufhin vom Akteur modifiziert werden und dann wie schon bei Neueinträgen gespeichert werden, woraufhin der Eintrag in der Liste der Zeiteinträge aktualisiert wird.

2.2.3 Report erstellen

Der UseCase "Report erstellen" beschreibt, wie die Mitarbeiter aus den Zeiteinträgen, die sie vorher gemacht haben, ihre monatlichen Reports über ihre Arbeitszeit erstellen und, wenn diese in Ordnung sind, ausdrucken.

Kurzbeschreibung	Der Akteur öffnet eine Berichtsansicht, um mit dem Report zu arbeiten.
Beteiligte Akteure	Anwender, Administrator
Vorbedingungen	Der Benutzer ist am System anmeldet
Nachbedingung	Keine
Ablaufbeschreibung	Der Akteur wählt eine ihm zur Verfügung stehende Reportart aus. Zum einen lässt er einen Report über die fakturierten Stunden eines Monats zu einem Projekt erstellen, druckt diesen aus und schickt sie dem Auftraggeber des Projektes zu Abrechnungszwecken. Zum anderen lässt er das Programm einen Report mit seinen fakturierten und unfakturierten Stunden zu jedem Projekt in diesem Monat erstellen, druckt diesen aus und gibt ihn an seinen Arbeitgeber. Diese Reports stehen jedem Akteur zur Verfügung. Zusätzlich stehen einem Administrator noch weitere Reports zur Verfügung, die ihm z.B. alle Arbeitszeiten aller Akteure in einem Projekt anzeigen.

2.2.4 Anwenderprofile bearbeiten

Im UseCase "Anwenderprofile bearbeiten" wird beschrieben, wie ein Mitarbeiter, der als Adiminstrator im System angemeldet ist, neue Mitarbeiter in das System einträgt oder auch die Daten von bereits vorhandenen Mitarbeitern modifiziert.

Dazu gehört auch, die monatliche Soll-Arbeitszeit des Mitarbeiters einzutragen, was in regelmäßigen Abständen für die weiteren Monate wiederholt werden muss.

Kurzbeschreibung	Der Administrator erstellt, bearbeitet und löscht Anwenderprofile.
Beteiligte Akteure	Administrator
Vorbedingungen	Der Benutzer ist am System anmeldet
Nachbedingung	Keine
Ablaufbeschreibung	Der Administrator öffnet den Dialog mit Anwenderprofilen, der eine Liste aller registrierten Anwender enthält. Er kann aus der Liste vorhandene Anwender auswählen, um diese zu bearbeiten oder zu löschen, oder einen neuen Benutzereintrag anlegen. Zusätzlich kann er für den Anwender einen Soll-Zeit Editor aufrufen und die monatlichen Soll-Arbeitszeiten des Mitarbeiters eintragen.

2.2.5 Projekte bearbeiten

Als letztes bleibt noch der UseCase "Projekte bearbeiten", der beschreibt, wie ein Administrator neue Projekte im System anlegt. Zur Bearbeitung des Projektes gehört ebenfalls, die Aktivitäten des Projektes einzutragen.

Kurzbeschreibung	Der Administrator erstellt, bearbeitet und löscht Projekte.
Beteiligte Akteure	Administrator
Vorbedingungen	Der Benutzer ist am System anmeldet
Nachbedingung	Keine
Ablaufbeschreibung	Der Administrator öffnet die Liste mit Projekten, wo er neue hinzufügen, vorhandene Einträge verändern oder löschen kann. Der Administrator kann für jedes Projekt die Aktivitäten bearbeiten.

3 Vision

Eine Vision ist ein Entwurf der Anwendung, die später realisiert werden soll. Dazu gehört ein Entwurf der Benutzeroberfläche und eine Beschreibung der Arbeitsweise mit dieser Oberfläche. Dies dient später als Orientierung für die tatsächliche Implementierung.

3.1 Grundstruktur der Oberfläche

Die Oberfläche ist in 3 Bereiche aufgeteilt (siehe Abbildung 2)

Abbildung 2: Grundstruktur der Oberfläche

Auf der linken Seite befindet sich das Menü, in dem die einzelnen Werkzeuge, mit denen im ZEIS gearbeitet werden kann, ausgewählt werden können. Zur Verfügung stehen die Werkzeuge "Zeiterfassung", "Berichtanzeige", "Projektbearbeitung" und "Mitarbeiterverwaltung". Unter der Auswahl für die Werkzeuge wird der derzeit angemeldete Benutzer angezeigt und darunter befindet sich ein Button, um sich abzumelden, wenn man mit seiner Arbeit am ZEIS-System fertig ist. Je nachdem, über welche Berechtigungen der angemeldete Benutzer verfügt sind nicht alle Verweise zu den Werkzeugen sichtbar. Nur die Verweise auf die Werkzeugen, die der Benutzer nutzen darf, sind dargestellt.

Auf der rechten Seite befinden sich der Auswahlbereich und der Arbeitsbereich. Der Inhalt dieser Bereiche ist vom jeweils gewählten Werkzeug abhängig. Alle Werkzeuge sind aber nach der vorgegebenen Struktur aufgebaut. Im Auswahlbereich hat der Benutzer die Möglichkeit, Materialen auszuwählen, die vorhanden sind, um diese dann im Arbeitsbereich anzeigen zu lassen und dort zu bearbeiten. Wenn möglich kann er im Auswahlbereich auch ein neues Material erstellen lassen, dass dann im Arbeitsbereich dargestellt wird, um es mit Information zu füllen.

Das neue ZEIS-System kennt 3 Rollen, die ein Benutzer haben kann. Diese Rollen können im Gegensatz zum alten ZEIS-System alle einzeln oder auch in beliebiger Kombination vergeben werden. Die erste ist die Rolle des normalen Benutzers. Ihm stehen die Werkzeuge "Zeiterfassung" und "Berichtanzeige" zur Verfügung. Die zweite Rolle ist die des Projektleiters. Ihm steht das Werkzeug "Projektbearbeitung" zur Verfügung. Die dritte Rolle ist die des Verwalters, der das Werkzeug

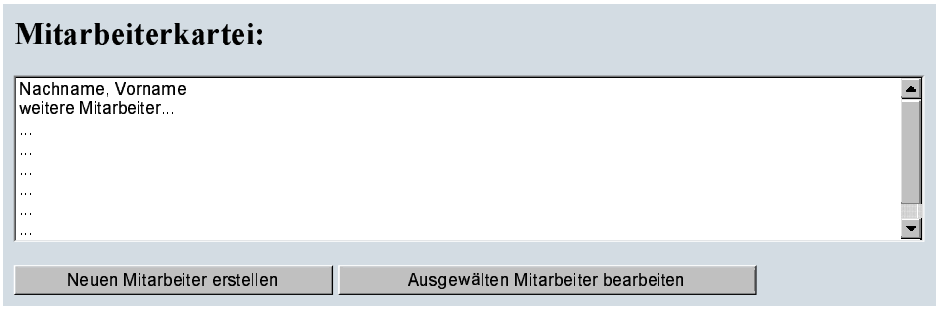
“Mitarbeiterverwaltung” nutzen kann. Die zusätzliche Rolle des Projektleiters wurde eingeführt, um auch einzelnen Mitarbeitern die Möglichkeit zu geben Projekte zu verwalten, also Mitarbeiter hinzuzufügen oder Aktivitäten zu erstellen, ohne dass ihnen auch gleichzeitig die Mitarbeiterverwaltung zur verfügung steht. In Tabelle 1 ist anhand der UseCases die alte und neue Rollenverteilung dargestellt.

Tabelle 1: Gegenüberstellung der alten und neuen Berechtigungen

UseCase	Alt	Neu
Anmelden	Anwender	Jeder Mitarbeiter
Ist-Zeit erfassen	Anwender	Benutzer
Report erstellen	Anwender	Benutzer
Anwenderprofile bearbeiten	Administrator	Verwalter
Projekte bearbeiten	Administrator	Projektleiter

3.2 Mitarbeiterverwaltung

Abbildung 3: Auswahlbereich der Mitarbeiterverwaltung



Im Auswahlbereich der Mitarbeiterverwaltung befindet sich eine Liste aller Mitarbeiter. In dieser Liste kann der Mitarbeiter ausgewählt werden, dessen Daten man verändern möchte. Wenn auf den Button “Ausgewählten Mitarbeiter bearbeiten” geklickt wird, werden die Daten des gewählten Mitarbeiters im Arbeitsbereich angezeigt. Über den Button “Neuen Mitarbeiter erstellen” wird der Arbeitsbereich geleert, um die neuen Daten einzugeben.

Im Arbeitsbereich können die Daten des ausgewählten Mitarbeiters bearbeitet werden oder die Daten für einen neuen Mitarbeiter eingegeben werden. Es sind Eingabefelder für Nachname, Vorname und Adresse vorhanden. Für die Beschreibung ist ein längeres Textfeld vorgesehen. Darunter wird das aktuelle Gehalt des Mitarbeiters angezeigt und ein Button rechts daneben ermöglicht es, den Editor für das Gehalt und die Soll-Arbeitszeit zu öffnen. In der letzten Zeile der Mitarbeiterdaten können die Berechtigungen des Mitarbeiters eingestellt werden. Zur Verfügung stehen “Benutzer”, “Verwalter” und “Projektleiter”. Unter dem Eingabebereich befindet sich der Button, mit dem die geänderten Daten gespeichert werden können.

3.3 Projektbearbeitung

Der Auswahlbereich der Projektbearbeitung entspricht im Wesentlichen dem der Mitarbeiterverwaltung. Im oberen Bereich befindet sich die Auswahlliste, in der

Abbildung 4: Arbeitsbereich der Mitarbeiterverwaltung

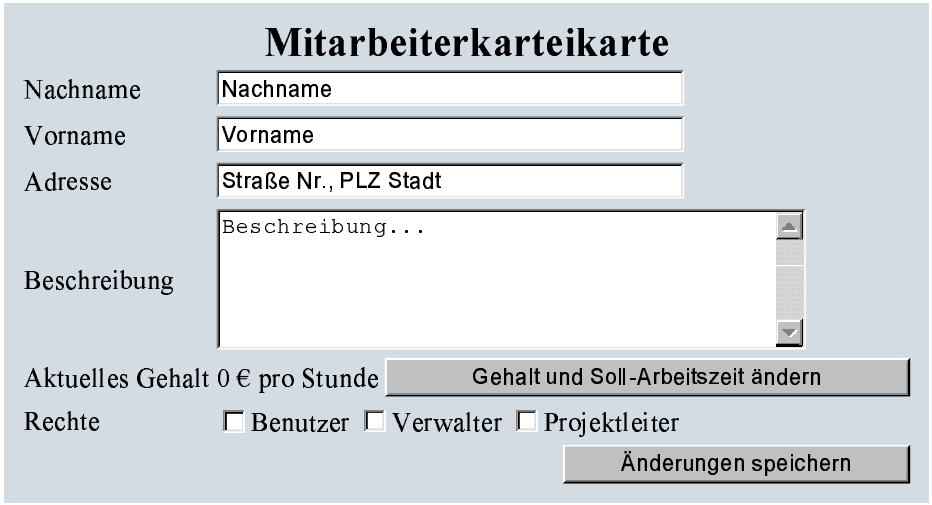


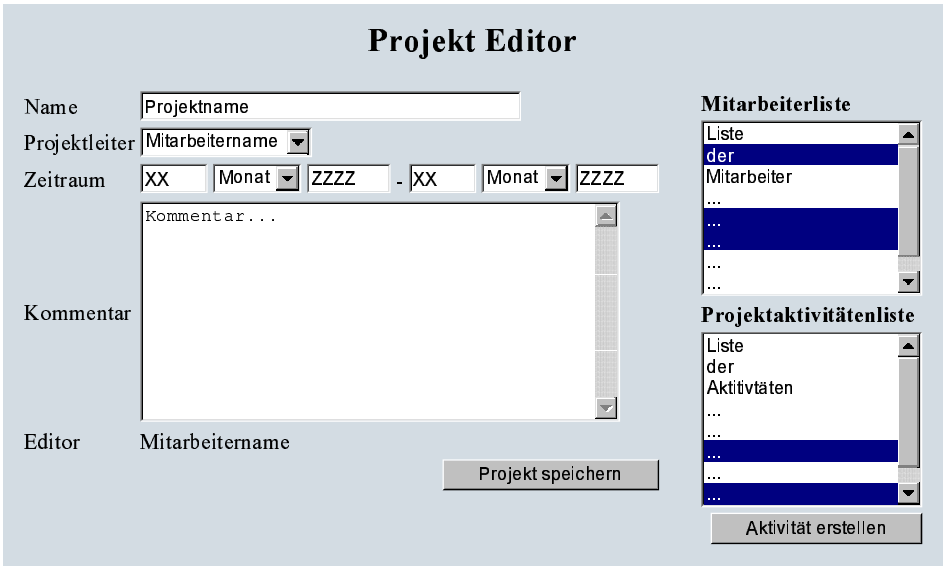
Abbildung 5: Auswahlbereich der Projektbearbeitung

alle exisitierenden Projekte aufgelistet sind. Darunter befinden sich die Buttons, um das gewählte Projekt im Arbeitsbereich der Projektbearbeitung zu öffnen oder ein neues Projekt zu erstellen.

Der Arbeitsbereich der Projektbearbeitung ist in 2 Spalten aufgeteilt. Auf der linken Seite können die Eingaben zu dem Projekt gemacht werden. Das oberste Feld ist für die Eingabe des Projektnamens. Darunter befindet sich eine Auswahlliste für den Projektleiter dieses Projektes. In der Zeile "Zeitraum" kann das Datum, an dem das Projekt beginnt, und das Datum, an dem es endet, eingegeben werden. Die Auswahl der Monate erfolgt dabei über Auswahllisten, da diese Auswahl immer gleich ist. Die Eingabe der Tage und der Jahre erfolgt über Eingabefelder. Unter dem Zeitraum ist ein Textfeld für die Eingabe des Projektkommentars. Dann folgt noch eine Zeile, die darüber Auskunft gibt, wer das Projekt zuletzt bearbeitet hat. Als letztes kommt dann noch der Button, mit dem die Eingaben gespeichert werden können.

Auf der rechten Seite befinden sich 2 Auswahllisten, die auch Mehrfachauswahl erlauben. In der oberen Liste können die Mitarbeiter ausgewählt werden, die in

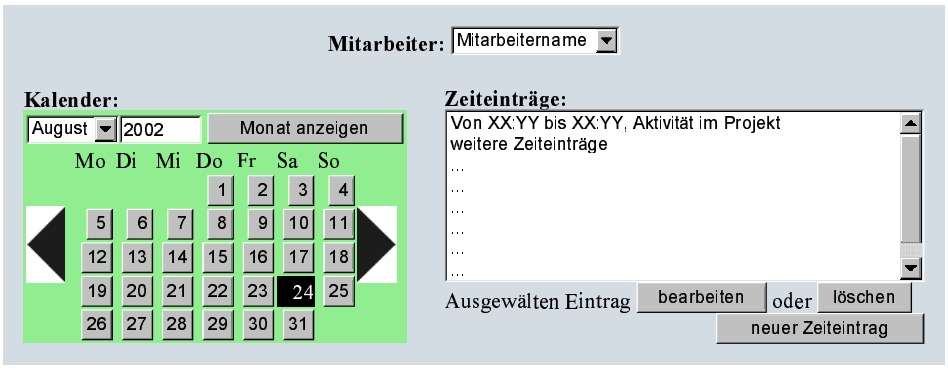
Abbildung 6: Arbeitsbereich der Projektbearbeitung



diesem Projekt arbeiten. Nur diese können in der Zeiterfassung auch Zeiteinträge für das Projekt erstellen, was gerade im Editor angezeigt wird. In der zweiten Liste sind die Aktivitäten aufgelistet, die das System kennt. Hier können die Aktivitäten ausgewählt werden, den in diesem Projekt nachgegangen werden kann. Nur die gewählten Aktivitäten können dann auch in der Zeiterfassung ausgewählt werden.

3.4 Zeiterfassung

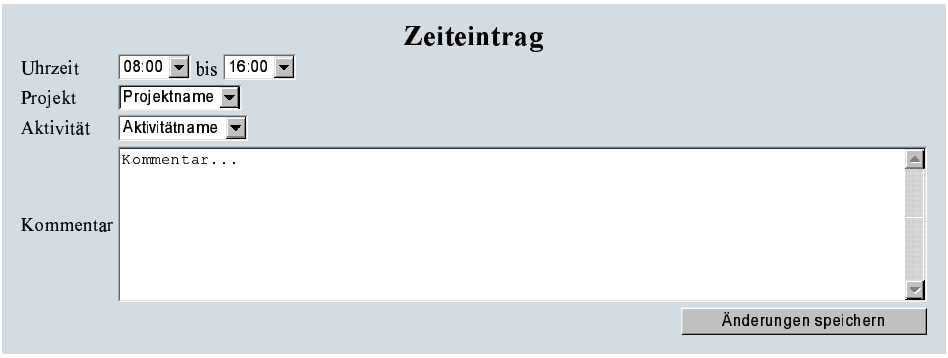
Abbildung 7: Auswahlbereich der Zeiterfassung



Oben im Auswahlbereich der Zeiterfassung kann der Mitarbeiter ausgewählt werden, für den Zeiteinträge erstellt wurden oder dessen Einträge bearbeitet werden sollen. Das neue ZEIS-System erlaubt im Gegensatz zum alten auch Zeiteinträge für andere Mitarbeiter zu erstellen, wenn diese vielleicht gerade keinen Zugang zum System haben und ihre Daten nur an einen anderen Mitarbeiter weitergeben können. Darunter befinden sich 2 Spalten. In der linken Spalte ist ein Kalender, in dem Mo-

nat, Jahr und Tag ausgewählt werden können, an dem Zeiteinträge erstellt werden sollen. Dies ist aber auch gleichzeitig der Tag, dessen vorhandene Einträge in der Liste in der rechten Spalte des Auswahlbereichs angezeigt werden. Darunter befinden sich 2 Buttons, mit denen die Aktionen "bearbeiten" und "löschen" auf den gewählten Eintrag angewendet werden können. "bearbeiten" öffnet den gewählten Eintrag im Arbeitsbereich und "löschen" entfernt den Eintrag aus der Liste. Unter diesen beiden Buttons befindet sich noch ein weiterer Button, um einen neuen Eintrag zu erstellen.

Abbildung 8: Arbeitsbereich der Zeiterfassung



Im Arbeitsbereich für einen Zeiteintrag kann in der ersten Zeile der Beginn und das Ende der Aktiviät ausgewählt werden. In der zweiten Zeile kann dann das Projekt ausgewählt werden, in dem gearbeitet wurde, und in der dritten Zeile die Aktivität die ausgeübt wurde. Darunter ist dann noch ein Textfeld in dem ein Kommentar zu der Arbeit eingegeben wurde. Unter dem ganzen Eingabebereich ist dann auf der rechten Seite ein Button, um den Zeiteintrag zu speichern.

3.5 Berichtanzeige

Abbildung 9: Auswahlbereich der Berichtsanzeige

Im Auswahlbereich kann auf der linken Seite das Mitarbeiter, das Projekt und die Aktivität ausgewählt werden, für die der Bericht erstellt werden soll. In den Feldern gibt es auch ein speziellen Eintrag, um alle Einträge zu verwenden. Daneben gibt es 2 Kalender in denen der erste und der letzte Tag ausgewählt werden können, von denen Zeiteinträge berücksichtigt werden sollen. Der Button unter den Auswahlfeldern auf der linken Seite öffnet die Berichtsinformationen im Arbeitsbereich.

Abbildung 10: Arbeitsbereich der Berichtsanzeige

Projekt: Projektname
Mitarbeiter: Mitarbeitername
Aktivität: Aktivitätsname
Tage: XX.YY.ZZZZ bis XX.YY.ZZZZ
Inhalt: XXX Zeiteinträge zu den ausgewählten Daten gefunden
erstellt von: Mitarbeitername
Berichtsansicht öffnen

Der Arbeitsbereich der Berichtsanzeige zeigt noch einmal die ausgewählten Daten an und wieviele Zeiteinträge für die Daten gefunden wurden. Von hier aus kann dann der eigentliche Bericht in einem neuen Fenster geöffnet werden.

4 Jakarta Struts

Dieses Kapitel wird sich mit dem Aufbau von Web-Applikationen mit Struts beschäftigen. Als erstes gibt es einige Informationen zum Hintergrund von Struts und dann einen überblick über die Funktionsweise von Struts. Danach wird der Aufbau der Konfigrationsdatei von Struts kurz erläutert und dann die Verwendeten Klassen von Struts beschrieben. Als letztes wird ein realtiv kleines Beispiel für eine Web-Applikation dokumentiert, um den Zusammenhang aller Klassen besser zu verdeutlichen.

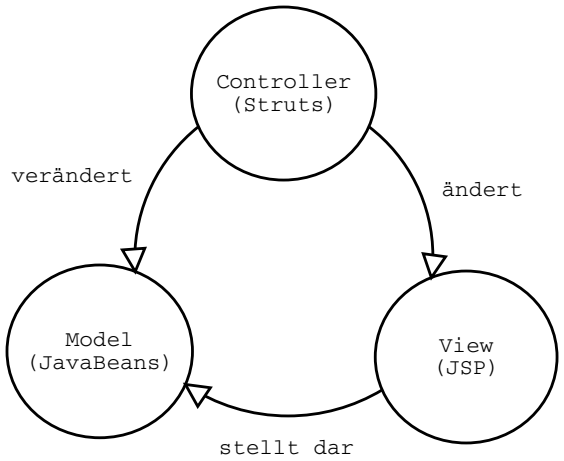
4.1 Hintergrund von Struts

Nach der Entwicklung der Java Servlets und der JavaServer Pages erkannte man schnell, dass man beide Dinge sehr gut verbinden kann, um umfangreiche Web-Applikationen zu bauen. Die Servlet Komponente übernimmt dabei die Steuerung des Kontrollflusses und die JavaServer Pages werden benutzt, um die HTML Ausgabe für den Browser zu erzeugen. Dies entspricht dem aus der Smalltalk Welt bekannten Model-View-Controller Design Pattern. Struts ist ein Framework, das genau dieses Pattern in einem Servlet umsetzt.

4.2 Model-View-Controller Design Pattern

Im Model-View-Controller Design Pattern wird der Kontrollfluss innerhalb einer Applikation von einem zentralen Controller gesteuert. Der Controller nimmt dabei die Anfragen, die an ihn gestellt werden (im Fall von Struts HTTP Requests) und übergibt sie dem dafür zuständigen Handler. Diese Handler sind an ein Modell gebunden und stellen die Verbindung zwischen dem Request und dem Modell her. Das Modell beschreibt die Anwendungs-Logik und den aktuellen Zustand des Gesamtsystems. Es wird dann der View bestimmt, der das Ergebnis des Requests für den Nutzer einer Applikation sichtbar macht (bei Struts JavaServer Pages). Die eigentliche Logik und die Darstellung für den Nutzer sind somit voneinander getrennt und ermöglichen eine einfachere Entwicklung und einfachere Wartbarkeit des Gesamtsystems.

Abbildung 11: Model-View-Controller



4.3 Überblick über Struts

Da Struts das Model-View-Controller Design Pattern implementiert, benötigt es natürlich 3 Komponenten, also einen Controller, ein Model und Views. Der Controller ist bereits Bestandteil von Struts. Er ist das eigentliche Struts-Servlet und wird wie jedes Servlet über die `/WEB-INF/web.xml` Konfigurationsdatei dem Servlet-Container bekannt gemacht. In dem struts-blank.war Servlet, das im Struts-Paket enthalten ist, ist bereits eine fertige web.xml enthalten, die alle nötigen Daten für Struts enthält.

Bei der Initialisierung lädt der Controller seine Konfigurationsdatei, die alle Informationen enthält die Struts für seine Arbeit benötigt. Die Datei heißt per default `/WEB-INF/struts-config.xml`. Der Name der Datei kann jedoch in der `web.xml`-Datei angegeben für das Controller-Servlet angeben werden und so im Prinzip jede Datei sein. Die in der Konfigurationsdatei angegebenen Daten definieren das Modell. Dies sind die Klassen, die den Kontrollfluss kontrollieren und die Zustandsdaten speichern. Diese werden später noch im einzelnen erklärt. Die View Komponente wird über JavaServer Pages realisiert. Über diese wird der aktuelle Zustand des Modells über HTML Dateien für den Anwender auf dem Browser sichtbar gemacht. Für das Erstellen der HTML-Ausgabe enthält Struts außerdem 4 TagLibs. Eine TagLib zum einfacheren Arbeiten mit Java Beans, eine TagLib zum Aufbau von HTML Formularen und einfachem Erstellen von Links und Verweisen auf Bilder. Die dritte TagLib bietet einiges an Programmierlogik und erlaubt zum Beispiel das Erstellen von Schleifen und Abfragen direkt als XML Tags in der JSP Seite, ohne richtigen Java Code in die JSP Seite einzubauen. Ob man diese TagLib benutzen will, ist jedoch mehr Geschmacksache, da diese Tags im großen und ganzen Java Code entsprechen, der anstelle der Tags auch benutzt werden könnte. Die vierte TagLib bietet Tags zum Erstellen von JSP Seiten mit Hilfe von Templates, um auf einfache Weise ein gemeinsames Layout für alle Seiten zu erreichen.

Der zweite wichtige Parameter naben der Struts-Konfigurationsdatei in der `web.xml`-Datei ist die Angabe der `ApplicationResources`. Diese Angabe wird wie eine Klassenbezeichnung angegeben und kann auch den Namen von Packages mit Punkt davor enthalten. Der Name der Datei die Struts anhand dieser Angabe sucht ist dann `ApplicationResources.properties` und muß sich an der angegebenen Stelle im Classpath befinden. Zur einfachen Unterstützung verschiedener Sprachen kann es von dieser Datei verschiedene Versionen geben, bei denen das Kürzel für die Sprache vor dem `.properties` angehängt wird, zum Beispiel `ApplicationResources_de.properties`. Struts testet beim öffnen der Datei automatisch zuerst die Sprachvariante und wenn diese nicht gefunden wird, wird die Datei ohne Sprachkürzel verwendet. In der Datei wird einfach in jeder Zeile einem Message-Key eine Meldung zugeordnet. Die Zeilen haben den Aufbau `key=message`.

4.4 Die Struts Konfigurationsdatei

Die Konfigurationsdatei ist eine normale XML Datei. Die gesamte Konfiguration wird in dem `<struts-config>`-Tag geschrieben. Diese kann 3 weitere Tags enthalten. Diese sind `<form-beans>`, `<global-forwards>` und `<action-mappings>`.

4.4.1 Form-Beans

Der `<form-beans>`-Tag enthält `<form-bean>`-Tags, die Formularnamen Java Bean Klassen zuordnen. Die Tags haben die Form

```
<form-bean name="Formularname" type="Klassenname" />
```

Als Parameter "name" wird der Name eines Formulars angebenen, dem die Klasse, die als Parameter "type" angegeben wird, zugeordnet werden soll. Die Klasse muß natürlich im Classpath des Servlets zu finden sein, also sollte sie sich entweder im /WEB-INF/classes Verzeichnis gefinden oder in einem jar-Archiv in /WEB-INF/lib Verzeichnis. Diese Zuordnung wird später im `<action-mappings>`-Tag wieder benutzt. JavaBeans können im Prinzip jede Klasse sein, die für jeden Parameter, den sie speichern soll eine `getParameter()` und eine `setParameter(...)` Methode enthält. Benutzt man für die FormBeans jedoch eine Unterklasse der Struts Klasse `org.apache.struts.action.ActionForm`, dann setzt Struts bei der automatischen Erzeugung der Bean eine Referenz auf das Servlet, die später über `getServlet()` abgefragt und benutzt werden kann.

4.4.2 Global-Forwards

Der `<global-forwards>`-Tag enthält `<forward>`-Tags, die symbolische Namen für JSP Seiten definieren. Die Tags haben den Aufbau

```
<forward name="Name" path="/Ziel.jsp" />
```

Diese symbolischen Namen, die als "name" Parameter angegeben werden, können später in den Klassen des Modells verwendet werden, um einfach auf JSP Seiten, die als "path" Parameter angegeben werden, zu verweisen, ohne den genauen Pfad dieser Datei zu kennen. Wenn eine JSP Seite dann einmal umbenannt oder verschoben wird, muß diese Änderung nur in der Struts Konfiguration eingetragen werden und muß nicht in jeder Klasse des Modells geändert werden.

4.4.3 Action-Mappings

Der `<action-mappings>`-Tag enthält die Definition der ActionMappings, die die URLs der HTTP Anfragen auf `Action`-Klassen abbilden. Jedes ActionMapping wird durch einen `<action>`-Tag definiert. Dieser hat eine Menge Parameter, von denen einige optional sind. Die Tags haben den Aufbau

```
<action    path="/requestPath"
           type="ActionClass"
           name="formBeanName"
           scope="request"
           validate="true"
           input="/inputForm.jsp">
    <forward name="Name" path="/Ziel.jsp">
</action>
```

Der Parameter "path" gibt an, welcher URI dieses ActionMapping zugeordnet werden soll. Anfragen an diese URI werden dann mit den in den anderen Parametern angegebenen Daten bearbeitet. Im Parameter "type" wird eine Klasse angegeben. Diese Klasse muß eine Unterklasse der Klasse `org.apache.struts.action.Action` sein und wird vom Struts Controller mit den Daten des Requests aufgerufen. "name" gibt den symbolischen Namen einer FormBean an, die unter `<form-beans>` definiert wurde. Diese Java Bean wird dann automatisch mit den Daten, die aus einem Formular vom Browser beim Request übergeben wurden, gefüllt. Der Parameter "scope" gibt an, wie lange die Form Bean, die für die Formular-Daten erzeugt wurde, erhalten bleiben soll. Mögliche Werte sind zum Beispiel "request" oder "session" entsprechend der aus den JSP bekannten Verwendung von JavaBeans. Enthält die Klasse der JavaBean eine Methode `validate` dann kann als Parameter für das ActionMapping "validate" auf "true" gesetzt werden, was dem Controller sagt, dass

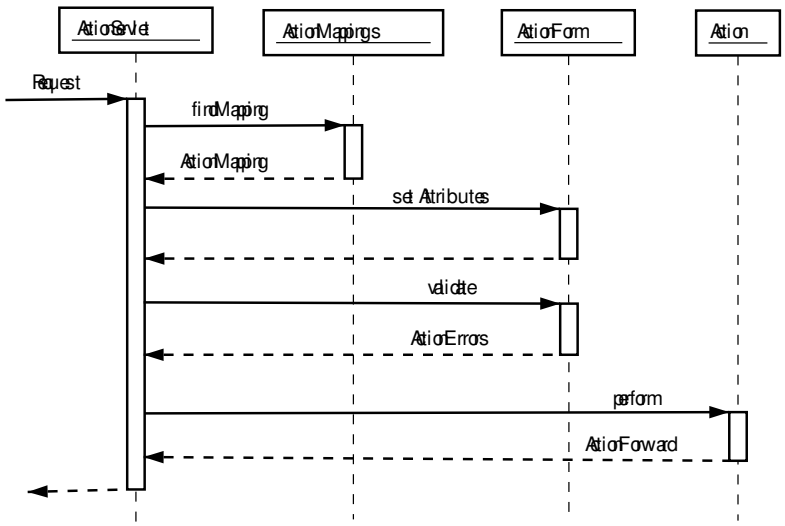
er den Inhalt der Bean mit Hilfe der `validate`-Methode erst prüfen soll, bevor er mit der eigentlichen Aktion weiter macht. Schlägt die Prüfung mit `validate` fehl, so wird automatisch wieder auf die Seite, die unter dem Parameter "input" angegeben ist, verwiesen.

Zusätzlich kann der `<action>`-Tag auch noch wieder `<forward>`-Tags enthalten, die genau wie bei `<global-forwards>` aufgebaut sind, aber nur für einen Request an die angegebene URI gültig sind.

4.5 Arbeitsweise des Controllers

Der Struts Controller läuft als Servlet. In der `web.xml` Datei werden bestimmte Requests auf das Servlet gemappt, so dass diese Requests vom Servlet verarbeitet werden. In der Standardkonfiguration sind dies alle Requests an Dateien mit der Endung `.po`. Der Controller sucht zuerst in den Action Mappings, die in der Konfigurationsdatei definiert sind, nach einem passenden Mapping, mit dem er den Request bearbeiten kann. Ist in dem Action Mapping eine `name` angegeben, werden die Parameter, die bei dem Request übergeben werden, automatisch an die set-Methoden der zugehörigen Form-Bean übergeben. Wenn die Form-Bean noch nicht existiert, wird sie automatisch erstellt. Beim Struts Controller ist es egal, ob die Parameter in der Form `?name=wert` an die Request URI angehängt werden oder als HTTP-POST Request übergeben werden. Sind alle Werte der Form-Bean gesetzt und ist der Parameter `validate="true"` in dem Action Mapping angegeben, ruft der Controller die `validate`-Methode der Form-Bean auf. Gibt diese Methode ein nicht leeres ActionErrors-Objekt zurück, dann sorgt der Controller automatisch dafür, dass die unter `input` angegebene JSP-Seite angezeigt wird. Dies sollte vernüftigerweise die Seite sein, in der die Daten für die Form-Bean eingegeben werden, damit der Anwender seine Eingabe korrigieren kann. Wird kein Fehler festgestellt oder keine Validierung verlangt, dann ruft der Controller die `perform`-Methode des `Action`-Objekts, des Typs, der unter `type` im Action Mapping angebenen ist, auf. Diese Methode kann dann anhand des aktuellen Zustands des Systems entscheiden, welcher View (also welche JSP-Seite) als nächstes angezeigt werden soll. Diese Information wird als Objekt der Klasse `ActionForward` an den Controller zurückgegeben.

Abbildung 12: UML Sequenz Diagramm der Request Verarbeitung



4.6 Struts Beispiel

Abschließend folgt nun das bereits angekündigte realtiv kurze Beispiel. Hierbei handelt es sich um die Eingabe der Spielernamen für ein kleines "Vier Gewinnt"-Spiel, bei dem 2 Spieler im Browser gegeneinander spielen können. Das Beispiel verwendet dafür 3 JSP-Seiten, 2 Form-Beans, 2 Action-Klassen und eine Klasse für die Representation des Spielfeldes, die aber für Struts selbst nicht weiter interessant ist. Das Spiel ist hier nicht vollständig dokumentiert, da die Vorgehensweise bei jeder Formular-Eingabe und -Verarbeitung in Struts immer gleich ist.

Auf der ersten JSP-Seite "index.jsp" werden die Spieler Namen abgefragt.

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html:html locale="true">
<head>
<title><bean:message key="index.title" /></title>
<html:base/>
</head>

<body bgcolor="white">

<bean:message key="index.info"/>:<br/>
<br/>
<html:form action="/playerNames">
<bean:message key="index.player" /> 1:
    <html:text property="player[0]"/>
    <html:errors property="player0"/><br/>
<bean:message key="index.player" /> 2:
    <html:text property="player[1]"/>
    <html:errors property="player1"/><br/>
<html:submit/>
</html:form>

</body>
</html:html>
```

Als erstes werden hier die verwendeten TagLibs angegeben und jeder TagLib ein Prefix zugeordnet. Mit diesem oder einem ähnlichen Kopf sollte daher jede JSP-Seite, die Struts verwendet, anfangen. Danach wird der Kopf der HTML-Datei erzeugt. Hierfür bietet die HTML-TagLib einige einfache Tags, die man verwenden sollte. Um den Titel der HTML-Seite anzugeben, wird der `<bean:message ... />`-Tag verwendet. Dieser gibt an seiner Stelle den Text aus, der als Meldung in der `ApplicationResources`-Datei unter dem angegeben Key zu finden ist. Als nächstes folgt dann der eigentliche Inhalt der Seite. Als erstes wird wieder mit dem `<bean:message ... />`-Tag die Information ausgegeben, was auf dieser Seite zu tun ist. Als nächstes kommt dann das Eingabeformular für die Namen der Spieler. Der `<html:form ... />`-Tag erzeugt automatisch ein Formular. Als "action" wird hier der "path" eines ActionMappings verwendet, das in der Konfigurations-Datei von Struts angegeben wird. Die nötige Dateiendnung für das Mapping auf das Servlet wird hierbei von Struts automatisch ergänzt. Außerdem bestimmt Struts über seine Konfiguration die Klasse der Form-Bean, die zu diesem Formular gehört und erzeugt eine Instanz davon, wenn dies nicht schon geschehen ist. Der

`<html:text ... />`-Tag erzeugt ein Eingabefeld. Zu dem als Parameter "property" angegebenen Namen muß es in der Form-Bean Klasse eine `set` und eine `get` Methode geben. In diesem Fall sind dies `setPlayer(...)` und `getPlayer(...)`. Der Tag `<html:errors ... />` gibt eine Fehlermeldung aus, wenn bei einem Abschicken des Formulars Fehler festgestellt werden. Mit "properties" kann hier auf eine bestimmte Kategorie von Fehlern verwiesen werden. Wird der "properties" Parameter nicht angegeben, dann werden alle Fehler ausgegeben. Hier sollen jedoch die Fehlermeldungen bei den Eingabefeldern in denen die aufgetreten sind, dargestellt werden. Danach wird nur noch das Formular abgeschlossen und die HTML-Seite beendet.

Die `ApplicationResources` Datei enthält für die Startseite die Einträge

```
index.title=4 in a Row!
index.info=To Start a Game you have to enter the player names first
index.player=Player
error.player.name=Not a valid player name
errors.header=<font color="red">
errors.footer=</font>
```

Die ersten 4 Zeilen enthalten die Texte für die Keys, die bereits aus der index.jsp bekannt sind. Die Keys "errors.header" und "errors.footer" werden von Struts automatisch verwendet, um Fehlermeldungen einzuschließen. In diesem Beispiel wird einfach die Textfarbe auf rot gesetzt. Auch wenn man diese Funktion nicht verwenden möchte, sollte man die Keys in der Resource Datei angeben, da Struts sonst anstelle von header und footer den Text "null" in die HTML-Ausgabe schreibt.

Für die Verarbeitung der Formular-Daten bedarf es nun auch noch einiger Einträge in der `struts-config.xml`.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
  "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">

<struts-config>

  <!-- ========== Form Bean Definitions ================ -->
  <form-beans>
    <form-bean      name="playerNamesForm"
                    type="get4.PlayerNamesForm"/>
  </form-beans>

  <!-- ========== Global Forward Definitions =========== -->
  <global-forwards>
        <forward   name="start"  path="/index.jsp"/>
        <forward   name="nextmove"  path="/game.jsp"/>
        <forward   name="showwinner"  path="/winner.jsp"/>
  </global-forwards>

  <!-- ========== Action Mapping Definitions =========== -->
  <action-mappings>
    <action    path="/playerNames"
               type="get4.PlayerNamesAction"
               name="playerNamesForm"
```

```
            scope="session"
            validate="true"
            input="/index.jsp"/>
  </action-mappings>

</struts-config>
```

Im `<form-beans>`-Tag wird die Form-Bean für das Formular definiert. Dem Formularnamen "playerNamesForm" wird die Klasse `get4.PlayerNamesForm` zugeordnet, von der Struts dann bei Bedarf Instanzen erzeugt, wie zum Beispiel bei der Definition des Formulars in der HTML-Seite. Danach werden unter `<global-forwards>` einige Namen für JSP-Seiten definiert. Als letztes wird die eigentliche Action definiert. Hier wird die Request URI als "path" angegeben. Dies entspricht dem "action" Parameter des `<html:form>`-Tags in der HTML-Datei. Als Action Klasse wird `get4.PlayerNamesAction` angegeben. Als nächtes kommt die Angabe des Formulars, das zu dieser Action gehört. Als nächtes wird angegeben, dass die Form-Bean für die gesamte Session erhalten bleiben soll. Dann folgt noch die Angabe, dass der Controller die Formulareingaben prüfen soll und die Seite, an die er weiterleiten soll, wenn dabei ein Fehler festgestellt wird.

Um die Daten aus dem HTML-Formular zu speichern, muss nun die Form-Bean Klasse definiert werden. Die Form-Bean Klasse ist eine Unterklasse der Struts Klasse `org.apache.struts.action.ActionForm` und enthält für jedes Attribut eine `get` und eine `set`-Methode.

```
package get4;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.*;

public class PlayerNamesForm extends ActionForm
{
    private String player[] = new String[2];

    public void setPlayer(int index, String player) {
        this.player[index] = player;
    }

    public String getPlayer(int index) {
        return this.player[index];
    }

    public void reset(ActionMapping mapping,
                      HttpServletRequest request) {

        this.player[0] = null;
        this.player[1] = null;
    }

    public ActionErrors validate(ActionMapping mapping,
                                 HttpServletRequest request) {

        ActionErrors errors = new ActionErrors();

        for(int i = 0; i < player.length; i++) {
```

```
            if ((player[i] == null) || (player[i].length() < 3))
            {
                errors.add("player" + new Integer(i).toString(),
                           new ActionError("error.player.name"));
            }

        }

        return errors;
    }
}
```

Die Klasse enthält die bereits erwähnten Methoden zum Setzen und Auslesen der Player, `getPlayer(...)` und `setPlayer(...)`. In diesem Fall haben die Methoden noch eine Angabe für einen Index, der in der JSP-Seite in eckigen Klammern hinter dem Namen des Attributes angegeben wurde. Die `reset` Methode dient dazu, die Bean wieder in ihren initialen Zustand zu versetzen. Das interessanteste ist jedoch die `validate` Methode, die vom Controller aufgerufen wird, um die Attribute der Bean auf gültige Werte zu testen. Hier wird geprüft, ob die Namen der Spieler gesetzt sind und ob sie länger als 2 Zeichen sind. Ist dies nicht der Fall, wird dem vorher erzeugten Objekt der Klasse `org.apache.struts.action.ActionErrors` mit der `add` Methode ein Fehler hinzugefügt. Als ersten Parameter bekommt die Methode die Kategorie, zu der der Fehler gehört, angegeben. Diese wurde bereits in der JSP-Seite verwendet, um die Fehler bei der Ausgabe durch den `<html:errors>` Tag den Eingabefeldern zuzuordnen. Das zweite Attribut ist der Fehler selbst. Bei der Erzeugung des Objektes der Klasse `ActionError` wird als Parameter der Key übergeben, unter dem im `ApplicationResources` File die Fehlermeldung gefunden werden kann. Wie schon bei `<bean:message>` wird auch hier automatisch die Meldung in der aktuellen Sprache ausgewählt. Danach gibt die Methode das `errors` Objekt zurück. Wenn dies keine Fehler enthält oder der Rückgabewert `null` ist, fährt der Controller, wie bereits beschrieben, mit der Arbeit fort. Sonst verweist er auf die zugehörige Eingabeseite.

Wurde kein Fehler bei der Form-Bean festgestellt, dann führt der Controller die `perform`-Methode der `PlayerNamesAction` Klasse aus, die in der Konfiguration angegeben wurde, um die nächste Seite zu ermitteln, die dargestellt werden soll.

```
package get4;

import org.apache.struts.action.*;
import javax.servlet.http.*;

public final class PlayerNamesAction extends Action
{
    public ActionForward perform(ActionMapping mapping,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response)
                          throws java.io.IOException,
                                 javax.servlet.ServletException {

        HttpSession session = request.getSession();
        session.setAttribute("playfield", new PlayField());

        return mapping.findForward("nextmove");
```

```
    }
}
```

Die `perform` Methode tut in diesem Fall nichts weiter als eine neue Instanz des Spielfeldes (PlayField) zu erzeugen und diese der Session unter dem Attribut "playfield" zuzuweisen. Danach wird aus dem mit Hilfe der `findForward` Methode des ActionMapping, über das diese `Action` aufgerufen wurde, das `ActionForward` gesucht, dass an den Controller zurückgegeben werden soll und den Verweis auf die nächste JSP-Seite enthält. Hier wird der symbolsiche Name der im `<global-forwards>`-Tag der Struts Konfiguration angegeben wurde, um die Ziel-Seite zu finden.

5 Die jZeis Services

5.1 Anbindung der Services

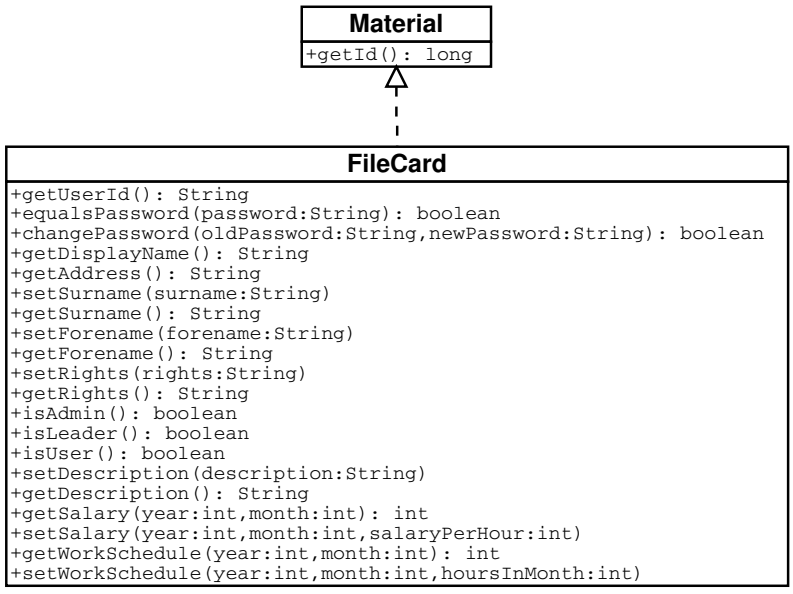
Wie bereits erwähnt, handelt es sich bei dieser Arbeit um die Entwicklung eines neuen Web-Frontends für eine bereits existierende Services Architektur. Um optimale Verteilung zu ermöglichen, erfolgt der Zugriff auf die Services über Remote Method Invocation (RMI). Um Zugriff auf die Services zu erhalten, muss als erstes ein Remote Objekt eines speziellen Login-Services angefordert werden. Dieser Login-Service stellt nach Aufruf einer Login-Methode mit Benutzername und Passwort und erfolgreicher Authentifikation ein Login-Token zur Verfügung, mit dem dann ohne erneute Authentifizierung die weiteren Services, die zur Materialbearbeitung benötigt werden, angefordert werden können.

5.2 Überblick über die Materialien

Die jZeis Services verwenden 5 Materialien für die Benutzer des Systems, die Projekte, die Aktivitäten, die Zeiteinträge und ein extra Material für die Authentifikation des Benutzers im System. Bei jedem Material ist ein UML Klassendiagramm, um den Zusammenhang mit den anderen Materialien darzustellen.

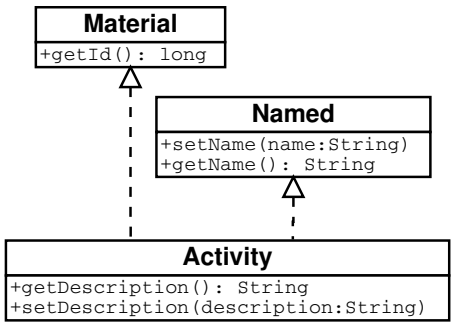
FileCard ist das Material für die Benutzer des Systems, die ihre Arbeitszeit mit dem System erfassen, Berichte erstellen, Projekte und Benutzer verwalten (je nachdem, über welche Berechtigung sie verfügen). Das Material enthält Informationen über den Vor- und Zunamen der Person, die Adresse, die Berechtigungen, eine zusätzliche Beschreibung, sowieso die monatlichen Soll-Arbeitszeiten und die Gehälter, die sich im Laufe der Zeit ändern können. Außerdem enthält das Material auch das Benutzer Passwort sowieso ein Login-Id, der systemweit eindeutig sein muss. Ein extra Login-Id wurde verwendet, weil der Nachname einer Person möglichweise nicht eindeutig sein könnte.

Abbildung 13: UML Klassendiagramm zum Material FileCard



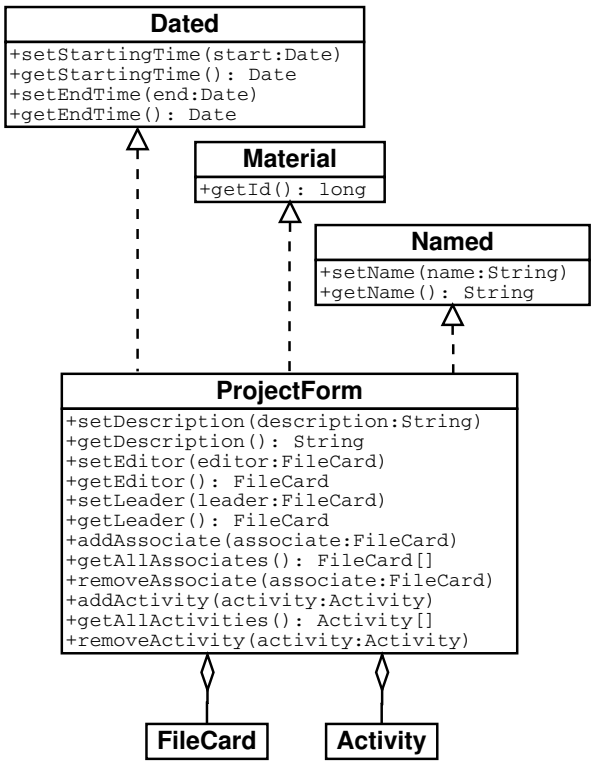
Activity ist das Material für die Aktivitäten, denen die Benutzer in ihren Projekten nachgehen können. Die Aktivität enthält nur einen Namen.

Abbildung 14: UML Klassendiagramm zum Material Activity



ProjectForm ist das Material für die Projekte. Ein Projekt hat jeweils einen Projektleiter, eine Projektbeschreibung und eine Information darüber, wer den Projekteintrag zuletzt bearbeitet hat. Außerdem enthält es 2 Listen. Eine Liste der Mitarbeiter, die diesem Projekt zugeordnet sind, und eine Liste der Aktivitäten, die in diesem Projekt durchgeführt werden können.

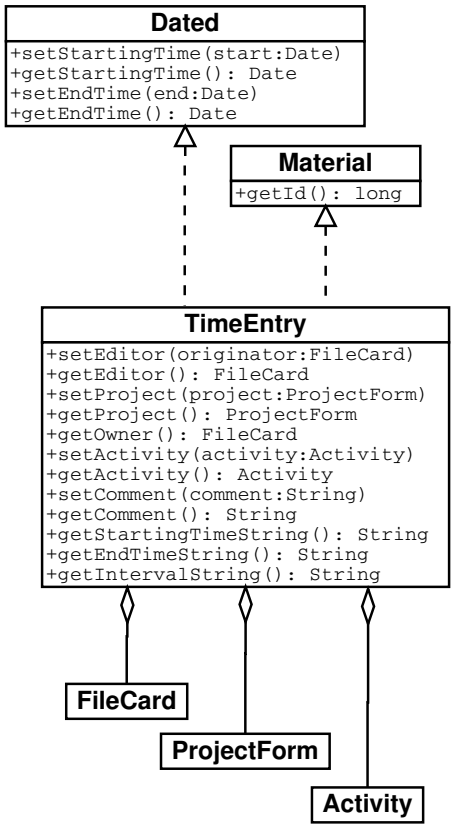
Abbildung 15: UML Klassendiagramm zum Material ProjectForm



TimeEntry ist das Material für die Arbeitszeit-Einträge. Arbeitszeit-Einträge enthalten, genau wie Projekte, eine Information, über die Person, die den Eintrag

zuletzt modifiziert hat. Außerdem die Information welchem Benutzer der Zeiteintrag zugeordnet ist, in welchem Projekt diese Person gearbeitet hat und welcher Aktivität sie dabei nachgegangen ist. Optional kann ein Kommentar zu jedem Zeiteintrag eingegeben werden. Und natürlich auch den Start-Zeitpunkt und den End-Zeitpunkt des Arbeitsintervalls.

Abbildung 16: UML Klassendiagramm zum Material TimeEntry



Token ist ein besonderes Material, das nur zur Authentifikation dient. Es wird jedesmal bei der Systemanmeldung erstellt. Es enthält keine für ein Frontend nutzbare Information.

5.3 Überblick über die Services

Im folgenden werden kurz die verfügbaren Services von jZeis beschrieben. Zu den Services **ZeisService** (Tabelle 2) und **ProjectService** (Tabelle 3) wird dabei auch das Interface mit angegeben. Das Interface der anderen Services entspricht im wesentlichen dem des **ProjectService**.

ZeisService ist der Hauptservice, der nur zur Authentifikation des Benutzers dient. Nach erfolgreicher Authentifikation erhält das aufrufende Programm einen Token, mit dem es Zugriff auf die anderen Services erhält.

TimeEntryService ist der Service zur Verwaltung der Zeiteinträge. Über diesen Service werden neue Zeiteinträge erstellt und alte gelöscht. Außerdem bietet er die Möglichkeit, Zeiteinträge nach ihrem Besitzer und nach Besitzer

Tabelle 2: Interface des ZeisService

Rückgabe	Methode	Beschreibung
Token	**login** (String userId, String password)	Prüft ob die Parameter "userId" und "password" zu einem Benutzer des Systems gehören und erzeugt bei erfolgreicher Authentifizierung ein Token für den Aufrufer
void	**logout** (Token token)	Meldet den Benutzer, dem der Token gehört, wieder von System ab
AssociatesService	**getAssociatesService** (Token token)	Gibt den AssociatesService an den Aufrufer zurück, wenn der Token gültig ist
ProjectService	**getProjectService** (Token token)	Gibt den ProjectService an den Aufrufer zurück, wenn der Token gültig ist
TimeEntryService	**getTimeEntryService** (Token token)	Gibt den TimeEntryService an den Aufrufer zurück, wenn der Token gültig ist

und Datum zu suchen. Er gibt dann eine Liste aller gefundenen Zeiteinträge zurück.

AssociatesService ist der Service, der die Mitarbeiter verwaltet. Über ihn können neue Mitarbeiter angelegt und alte wieder gelöscht werden. Außerdem kann über ihn eine Liste aller zur Verfügung stehenden Mitarbeiter angefordert werden. Die dritte Funktion, die der Service anbietet, sind 2 Suchoperationen. Die erste der beiden sucht Mitarbeiter nach ihrem Namen. Da Namen nicht eindeutig sein müssen, wird in diesem Fall eine Liste aller gefundenen Namen zurückgegeben. Die zweite Suchoperation sucht Mitarbeiter nach ihrem UserId. UserIds sind eindeutig und darum wird hier nur das gefundene Associates Material zurückgegeben.

ProjectService ist der Service, der zuständig ist für Projekte. Wie die beiden anderen Material-Services bietet er die Möglichkeit, Projekte anzulegen und zu löschen. Außerdem eine Suchoperation, die Projekte anhand ihres Namens findet. Eine zweite Suchoperation erlaubt das Suchen nach Projekten, in denen ein bestimmter Mitarbeiter mitwirkt. Eine Liste aller existierenden Projekte kann ebenfalls angefordert werden. Zusätzlich zu den Projekten verwaltet der **ProjectService** auch die Aktivitäten. Über ihn können neue Aktivitäten angelegt werden und eine Liste aller Aktivitäten angefordert werden.

Tabelle 3: Interface des ProjectService

Rückgabe	Methode	Beschreibung
ProjectForm	**addNewProject** ()	Erzeugt ein neues Projekt und gibt es als Rückgabewert an den Aufrufer zurück
void	**removeProject** (ProjectForm project)	Entfernt das Projekt, das als Parameter übergeben wird, aus dem System
ProjectForm[]	**getAllProjects** ()	Gibt eine Liste aller verfügbaren Projekte zurück
ProjectForm[]	**findProjectByName** (String name)	Sucht Projekte anhand des Projektnamens und gibt eine Liste aller gefundenen Projekte zurück
ProjectForm[]	**findProjectsByAssociate** (FileCard associate)	Sucht Projekte anhand des Mitarbeiters, der als Parameter übergeben wird, und gibt eine Liste aller Projekte in denen der Mitarbeiter arbeitet zurück
Activity	**addNewActivity** (String name)	Erzeugt eine neue Aktivität mit dem Namen, der als Parameter angegeben wird und gibt diese als Rückgabewert zurück
Activity[]	**getAllActivities** ()	Gibt eine Liste aller verfügbaren Aktivitäten zurück
boolean	**isProject** (Object project)	Prüft ob das Objekt, das als Parameter übergeben wird, ein Projekt ist und gibt das Ergebnis als Rückgabewert zurück

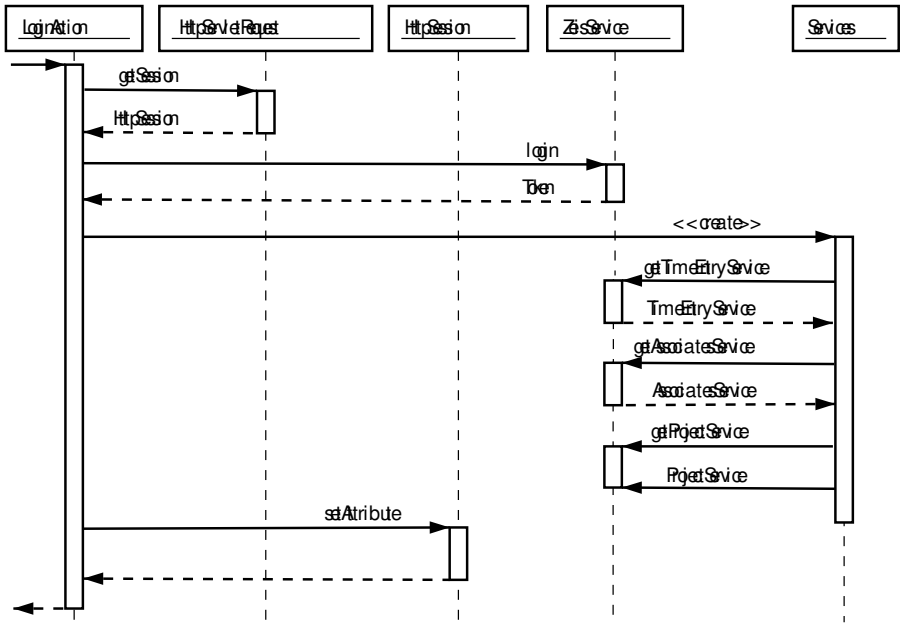
6 Realisierung

Für die Realisierung des Zeiterfassungssystems werden 5 JSP-Seiten verwendet. Eine ist dabei die Login-Seite. Die anderen sind Seiten für jedes Werkzeug. Zum Aufbau der Werkzeug-Seiten wurde die Struts Template-TagLib benutzt. Das Template enthält dabei den Menübereich, der ja für alle Seiten gleich ist, und die Tabelle, die die einzelnen Bereiche der Werkzeuge einschließt.

6.1 Login

Auf der Login-Seite gibt der Benutzer zunächst sein Benutzer-Id und sein Passwort in ein Formular ein. Die zugehörige `LoginForm`-Bean prüft, ob in allen Feldern Eingaben vorhanden sind. Ist dies der Fall wird holt sich die `LoginAction`-Klasse über RMI ein Remote-Objekt des `ZeisService` und ruft die `login`-Methode mit den Daten aus dem Menü auf und bekommt bei erfolgreicher Authentifikation ein Token als Ergebnis. Sollte dies nicht der Fall sein, wird ein Fehler erzeugt und wieder auf die Login-Seite verwiesen. Mit dem ZeisService und dem Token wird dann eine Instanz der `Services`-Klasse erzeugt. Die `Services`-Klasse holt sich vom `ZeisService` mit dem Token die anderen 3 Services und speichert die Referenzen. Die `Services`-Klasse enthält genau wie der `ZeisService` Methoden um alle 3 Services abzufragen. Wenn das Objekt der `Services`-Klasse zerstört wird, wird in der `finalize`-Methode die `logout`-Methode des `ZeisService` aufgerufen. Das Objekt der `Services`-Klasse wird dann in der `LoginAction`-perform-Methode einem Attribut der aktuellen Session zugewiesen, damit es später jederzeit zur Verfügung steht und der Zugriff auf die Services möglich ist. Danach wird auf die Werkzeugoberfläche verwiesen. Der Ablauf ist in Abbildung 17 dargestellt.

Abbildung 17: UML Sequenz Diagramm der LoginAction



6.2 Menübereich

Die Werkzeug-Oberfläche ist, wie bereits in der Vision in Abschitt 3 beschrieben, in 3 Bereiche geteilt, den Menübereich, den Auswahlbereich und den Arbeitsbereich. Der Menübereich enthält einfach eine Auflistung aller zur Verfügung stehenden Werkzeuge. Um zu ermitteln, welche Werkzeuge zur Verfügung stehen, wird aus der Session die Referenz auf das `Services`-Objekt geholt und darüber eine Referenz auf den `AssociatesService` erhalten, dessen `whoAmI`-Methode ein Objekt der Klasse `FileCard` liefert, mit den Daten des gerade angemeldeten Benutzers. Mit den Methoden `isUser`, `isLeader` und `isAdmin` kann dann ermittelt werden, welche Links auf die Werkzeuge in der Menüleiste angezeigt werden sollen. Unter dem Menü wird noch der Name des Benutzers angezeigt und unter dem Namen ist noch ein Link um sich abzumelden. Der Link verweist auf die `LogoutAction`-Klasse, deren `perform`-Methode, die aktuelle Session invalidiert und damit alle Referenzen auf die Services zerstört.

Die Auswahlbereiche für alle 4 Werkzeuge sind im Prinzip gleich. Jeder Bereich enthält eine Form-Bean, die die aktuelle Auswahl des Benutzers speichert. Da sich Auswahlbereiche der Zeiterfassung, der Projektbearbeitung und der Mitarbeiterverwaltung nur wenig unterscheiden werden diese zunächst näher betrachtet. Die Berichtanzeige folgt später, da diese sich von den anderen unterscheidet.

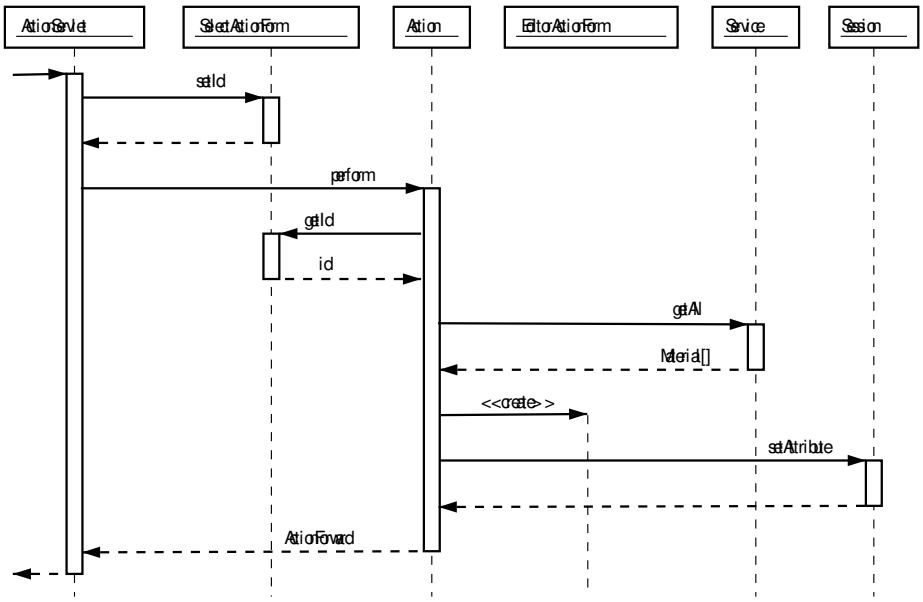
6.3 Zeiterfassung, Projektbearbeitung und Mitarbeiterverwaltung

In den JSP-Seiten der Zeiterfassung, der Projektbearbeitung und der Mitarbeiterverwaltung wird über das `Services`-Objekt auf die einzelnen Services zugegriffen und Listen aller oder einer Auswahl der Materialen angezeigt. Die Projektbearbeitung und die Mitarbeiterverwaltung zeigen immer alle möglichen Materialien an und die Zeiterfassung nur die des gerade ausgewählten Tages. Die Form-Beans speichern dabei die Id des ausgewählten Materials, die über die `getId`-Methode der Materialien abgefragt werden kann. Die Form-Bean der Zeiterfassung speichert zusätzlich auch noch den Tag im Kalender, der ausgewählt ist. In jedem der Auswahlbereiche steht ein Button zur Bearbeitung des gewählten Materials zur Verfügung und ein zweiter Button, um ein neues Material zu erzeugen.

Die `Action`-Klassen zu den Auswahlbereichen verwenden dann die Daten der Form-Beans, wie zum Beispiel die Id des Materials, um das eigentliche Material zu erhalten. Es wird eine Liste der Materialien angefordert und dann aus der Liste durch ein Vergleich der Ids das ausgewählte Material ermittelt. Dies ist natürlich nur nötig, wenn der "Bearbeiten"-Knopf des Auswahlbereiches benutzt wurde, um das Formular abzusenden. Die `perform`-Methode der Action Klasse erstellt daraufhin eine Form-Bean für den Arbeitsbereich. Diese Form-Beans bekommen in dem Fall, dass ein Material bearbeitet werden soll, dieses als Parameter im Konstruktur übergeben. Wenn ein neues Material erstellt werden soll, wird einfach eine neue Form-Bean erzeugt. Die Session-Beans, die Struts für seine Formulare verwendet, werden unter dem Namen, der in der Konfigurationsdatei angegeben ist, an die Session angefügt. Es ist also auch möglich diese selbst über `session.setAttribute` zu erzeugen und der Session zuzuweisen.

Die Form-Bean speichert die Referenz auf das Material, wenn eines beim Aufruf des Konstruktors übergeben wurde, und fragt beim Aufruf einer `get`-Methode den Original-Wert vom Material ab, wenn in der Form-Bean noch kein neuer Wert gespeichert wurde. Wenn das Formular des Arbeitsbereiches abgeschickt wird, dann trägt Struts die neuen Werte in die Form-Bean ein. Diese speichert die Werte aber erst in lokalen Attributen und schreibt sie noch nicht in das Material. Wenn dann bei der Überprüfung der Eingabe ein Fehler festgestellt wird, bleibt der Inhalt der Form-

Abbildung 18: UML Sequenz Diagramm der Action-Klassen des Auswahlbereiches von Zeiterfassung, Projektbearbeitung und Mitarbeiterverwaltung



Bean erhalten, und die neu eingegebenen Werte können wieder in das Formular eingesetzt werden, ohne das Material zu verändern. Jeder der Form-Bean enthält eine `save`-Methode, die die neuen Werte in das Material schreibt. Wenn der Form-Bean bei der Erstellung kein Material übergeben wurde, dann erzeugt die `save`-Methode erst ein neues Material und setzt dann Attribute des neuen Materials.

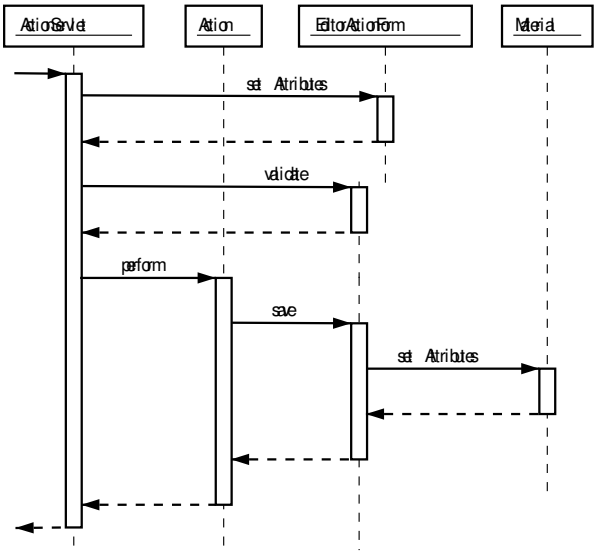
Die `Action`-Klasse des Arbeitsbereiches kann noch einmal die Daten der Form-Bean prüfen und ruft dann die `save`-Methode auf, um die Änderungen endgültig zu speichern.

Mit diesem Aufbau kann auch schon die ganze Funktionalität der 3 Werkzeuge erreicht werden. In jedem wird praktisch nur ein Material ausgewählt, um dieses dann zu bearbeiten, oder es wird ein neues Material erstellt.

6.4 Berichtsanzeige

Der Auswahlbereich der Berichtanzeige unterscheidet sich von diesem Schema und bietet eine Auswahl für den Mitarbeiter, das Projekt und die Aktivität, über die der Bericht erstellt werden soll. Um den Bericht zu erstellen, steht auch nur ein Button zur Verfügung. Außerdem muss die zugehörtige Form-Bean das Anfangs- und Enddatum für den Bericht speichern. Die `Action`-Klasse für die Berichtsauswahl erzeugt dann nurnoch eine Kopie dieser Form-Bean und weist sie der Session unter einem neuen Attributsnamen zu, so dass sie vom Arbeitsbereich verwendet werden kann. Dadurch hat man im Auswahlbereich die Möglichkeit seine Auswahl schon wieder zu verändern ohne die Daten des im Arbeitsbereich angezeigten Berichtes zu verändern. Im Arbeitsbereich werden zunächst nur die Daten der Form-Bean angezeigt und ermittelt wieviele Zeiteinträge überhaupt zu der Auswahl gehören. Über ein weiteren Link kann dann der vollständige Bericht in einem neuen Fenster geöffnet werden.

Abbildung 19: UML Sequenz Diagramm der Action-Klassen des Arbeitsbereiches von Zeiterfassung, Projektbearbeitung und Mitarbeiterverwaltung



7 Auswertung

7.1 Zusammenfassung

Insgesamt war die Realisierung des Web-Frontends mit Struts recht einfach. Zu jedem Formular muß nur eine Form-Bean und eine `Action`-Klasse erstellt werden. Der Controller nimmt einem viel der restlichen Arbeit schon ab. Die Eingabedaten des Formulars werden automatisch in die Form-Bean eingetragen und in der `Action`-Klasse können diese Eingaben problemlos verwendet werden, wenn dies nötig ist, da die `perform`-Methode die Form-Bean als Parameter bekommt. Durch die automatische Prüfung der Eingaben durch den Controller mit der `validate`-Methode braucht für die Weiterleitung an die Eingabeseite überhaupt nichts selber programmiert werden, da der Controller dies automatisch erledigt, wenn von der `validate`-Methode Fehler zurückgegeben werden. In der `Action`-Klasse braucht nurnoch entschieden werden, welche Seite als nächtes angezeigt werden soll. Hierfür helfen die Forward-Definitionen in der Struts-Konfigurationsdatei. Objekte die über mehrere Aufrufe erhalten bleiben sollen, wie die Referenz auf die Services, können einfach der aktuellen `HttpSession` hinzugefügt werden und später an jeder Stelle wieder verwendet werden. Die Struts TagLibs sind auch sehr nützlich. Sie erlauben es durch einfache XML-Tags Formulare zu erstellen, deren Felder automatisch mit dem Inhalt der Form-Bean gefüllt wird, oder die Attribute von JavaBeans auszugeben.

Die Einschränkung von Web-Seiten, dass man die eingegebene Information immer nur dann verarbeiten kann, wenn diese vom Browser übermittelt wird, und nicht jederzeit interaktiv die Eingabe überwachen kann, wie zum Beispiel bei einer Swing Oberfläche, hat für das Bearbeiten der Materialien keine Probleme bereitet. Zu jedem Material wird einfach eine Form-Bean erstellt, die die Vermittlung zwischen den Materialien und den Formularen darstellt. Neue Eingaben können zuerst in der Bean gespeichert werden, bevor das Material tatsächlich geändert wird.

7.2 Ausblick

Der jZeis Service hat derzeit kein wirkliches Kooperationsmodell. Dies muß sicher in der Zukunft noch eingebaut werden und dann muß es auch in den Frontends berücksichtigt werden. Wenn ein Material modifiziert werden soll wird es erst in der `Action`-Klasse vom Service angefordert. Es dürfte keine Schierigkeit sein, hier zu Prüfen, ob ein Material modifiziert werden kann und einen entsprechenden Fehler zu liefern, wenn dies nicht möglich ist.

Struts hat sich für die Entwicklung von Applikationen als sehr nützlich erwiesen. Auch der Einsatz in größeren Applikationen dürfte problemlos möglich sein. Die ständige Weiterentwicklung vo Struts durch die OpenSource-Entwickler wird Struts sicher in Zukunft um noch weitere nützliche Funktionen erweitern, die die Realisierung von Web-Applikationen weiter unterstützen werden. Außerdem steht es natürlich jedem frei, Struts selbst zu erweiteren, wenn es nötig ist.