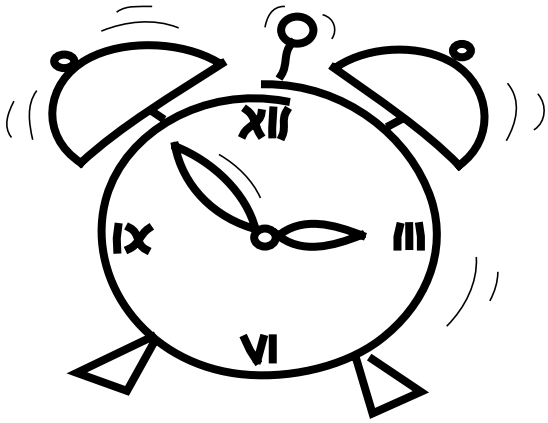


Projekt-Zeiterfassung

Referenznummer:	23.15	
Abgabedatum:	22. April 1999	
Autor:	Pascal Müller Buchenweg 2 2560 Nidau 032 / 331 2273	Sulzer Microelectronics AG Mattenstrasse 6a 2555 Brügg 032 / 374 3227
Klasse:	B23	
Betreuer:	Stefan Leuenberger Kirschbaumweg 13 2563 Ipsach 032 / 331 7312	Sulzer Microelectronics AG Mattenstrasse 6a 2555 Brügg 032 / 374 3234
Experte:	Stephan Fischli	Software-Schule Schweiz

Abstract: Das Erfassen von projektspezifischer Arbeitszeit ist Voraussetzung für eine genaue Nachkalkulation. Diese Software erlaubt dies auf Intranetbasis. Das Java-Eingabe-Frontend ist plattformunabhängig und die Daten werden in einer zentralen Datenbank abgelegt, so dass die Zeiten von jedermann jederzeit und überall erfasst werden können. Aus den so erfassten Daten können aussagekräftige Auswertungen erzeugt und auf dem Intranet zur Verfügung gestellt werden.

Keywords: Zeiterfassung, Intranet, Java, JDBC, Datenbank, RDBMS, SQL

Datum: 22. April 1999

Historie:

Datum	Version	Autor	Bemerkungen
16. 12. 1998	0.1	pm	Initialdokument, Formatdefinitionen
16. 2. 1999	0.2	pm	Datenbankteil fertiggestellt
4. 3. 1999	0.3	pm	Korrekturen nach Besprechung mit Herren Schmidhauser/Fischli (SWS) vom 3. 3. 1999, neuer Abstract
9. 4. 1999	0.5	pm	Design-Teil Entwurf
22. 4. 1999	1.0	pm	Ergänzt und fertiggestellt

Hinweis:

Referenzen auf andere Kapitel, Literaturverweise und URL's sind Hyperlinks. Durch anklicken wird automatisch an die entsprechende Stelle gesprungen.

1 • Allgemeines

1.1 Zweck

Dieses Dokument beschreibt die Realisierung einer Projekt-Zeiterfassungs-Anwendung und definiert *Wie* es gemacht wurde. Es dient der Beschreibung des Lösungsweges und ist Bestandteil der Diplomarbeit 23.15.

1.2 Definitionen und Abkürzungen

1.2.1 Änderungen

Änderungen im Dokument seit der letzten Version werden am linken Rand mit einem schwarzen Balken gekennzeichnet (hier zur Veranschaulichung vorhanden).

1.2.2 Abkürzungen

API	Application Programming Interface
AWT	Abstract Windowing Toolkit (Javas plattformunabhängiges GUI API)
ERM	Entity Relationship Model
JDBC	Java Database Connectivity
JDC	Java Developer Connection
JDK	Java Development Kit
JFC	Java Foundation Classes (Java GUI Komponenten, siehe [3])
JRE	Java Runtime Environment
JSDK	Java Servlet Development Kit
RDBMS	Relational Database Management System
RDM	Relational Data Model
SQL	Structured Query Language
SSI	Server Side Includes
URL	Uniform Resource Locator

1.2.3 Begriffe

Da diese Anwendung zwei verschiedenen Benutzergruppen genügen muss, werden in diesem Zusammenhang folgende Begriffe verwendet:

- Anwender sind alle, die in irgendeiner Form mit dem Zeiterfassungssystem in Berührung kommen (um z.B. Reports abfragen)
- Entwickler sind Anwender, welche das System aktiv nutzen, also ihre Zeiten erfassen. Dies sind die Leute aus der Entwicklungsabteilung (Engineering)

1.3 Referenzen

Literaturverweise werden im Text mit [<Nummer>] gekennzeichnet. Die Nummer (#) bezieht sich auf folgende Tabelle:

#	Titel	Autor	Bemerkungen / URL
1	Pflichtenheft 23.15	Pascal Müller	Projekt-Zeiterfassung
2	Java White Papers	Sun Microsystems	http://java.sun.com/docs/white/
3	JFC White Paper	Sun Microsystems	http://java.sun.com/marketing/collateral/foundation_classes.html
4	Java Development Kit	Sun Microsystems	http://www.sun.com/solaris/java/index.html
5	Java Plugin Software	Sun Microsystems	http://java.sun.com/products/plugin/
6	Java Workshop	Sun Microsystems	http://www.sun.com/software/Developer-products/java/
7	Year 2000 Product Compliance Status List	Sun Microsystems	http://www.sun.com/y2000/cpl.html
8	JDBC API Documentation	Sun Microsystems	http://java.sun.com/products/jdbc/
9	Java Code Conventions	Sun Microsystems	http://java.sun.com/docs/codeconv/html/CodeConventionsTOC.doc.html
10	Draft Java Coding Standard	Doug Lea	http://gee.cs.oswego.edu/dl/html/javaCodingStd.html
11	Netscapes's Software Coding Standarts Guide for Java	Christie Badeaux	http://developer.netscape.com/docs/technote/java/codestyle.html
12	Java documentation comments conventions	Doug Kramer	http://java.sun.com/products/jdk/javadoc/writingdoccomments.html
13	JAVA design: building better apps and applets (2nd edition)	Peter Coad Mark Mayfield	Yourdon Press, ISBN 0-13-911181-6 http://www.oi.com
14	100% Pure Java CookBook	Sun Microsystems	http://java.sun.com/100percent/cookbook.html
15	JDK Software Compatibility	Sun Microsystems	http://java.sun.com/products/jdk/1.2/compatibility.html
16	JDK 1.2 Deprecated API list	Sun Microsystems	http://java.sun.com/products/jdk/1.2/docs/api/deprecated-list.html
17	Java Servlet API	Sun Microsystems	http://java.sun.com/products/servlet/index.html
18	Java Servlet Fundametals Training	MageLang Institute	http://developer.javasoft.com/developer/onlineTraining/#Servlets (JDC) http://www.magelang.com/
19	The Apache JServ Project	The Apache Group	http://java.apache.org/
20	Apache JServ installation guide	Ari Halberstadt ari@shore.net	http://www.magiccookie.com/computers/apache-jserv/index.html
21	Apache Web site	The Apache Group	http://www.apache.org/
22	PostgreSQL Website	PostgreSQL Organization	http://www.postgresql.org/
23	PostgreSQL JDBC driver	Peter T. Mount	http://www.retep.org.uk/postgres/
24	Database-SQL-RDBMS HOW-TO	Alavoor Vasudevan	http://metalab.unc.edu/LDP/HOWTO/PostgreSQL-HOWTO.html
25	SQL Reference Page	Derrick Brashear	http://www.contrib.andrew.cmu.edu/~shadow/sql.html
26	Adobe Acrobat Reader	Adobe	http://www.adobe.com

2 • Evaluation

2.1 Entwicklungsumgebung

2.1.1 JDK

Das Java Development Kit [4] enthält alles, was zum Entwickeln und Ausführen von Java Software benötigt wird. Es wurde JDK 1.1.7_05 gewählt, da es das aktuellste 'production release' ist und sehr stabil läuft. Es ist auch die vollständige Dokumentation in elektronischer Form vorhanden.

2.1.2 JSDK

Das Sun Java Servlet Development Kit [17] wird benötigt, um Servlets zu erstellen. Das JSDK 2.0 *muss* für Servlets, welche mit Apache JServ (Siehe Kapitel 2.2.1 (Web Server) auf Seite 5) arbeiten sollen, verwendet werden.

2.1.3 Java Workshop

Die in [1] (Kapitel 2.4.2 (Entwicklungsumgebung) auf Seite 7) erwähnte Software Java Workshop 2.0 [6] wird aus folgenden Gründen nicht eingesetzt:

- Es basiert noch auf dem alten Event-Modell von Java 1.0. Damit wird die geforderte Portierbarkeit auf Java 1.2 beeinträchtigt.
- Es ist nach Sun nicht Jahr 2000 tauglich (siehe [7]). Es ist nicht klar, ob dies auch die Workshop Laufzeitbibliothek und den erzeugten Code betrifft.

2.2 Laufzeitumgebung

2.2.1 Web Server

Aus folgenden Gründen wird Apache 1.3.3 (siehe [21]) eingesetzt:

- Sehr verbreitet, läuft äusserst stabil und ist auf vielen Plattformen lauffähig.
- GNU Lizenzierung erfüllt die Forderung nach freier Software ([1], Kapitel 2.4 (Software) auf Seite 7).
- Es ein Servlet-Modul verfügbar. Verwendet wird Apache JServ 1.0b3 (siehe [19]). Dies ist eine Java Servlet Engine und ermöglicht das Ausführen von Servlets auf dem Apache Web Server.
- Es ist Software für SSI verfügbar. Verwendet wird Apache JServSSI 1.0 (19981216) (siehe [19]). Dies ermöglicht das Einbetten von Servlets (Server Side Includes) in HTML-Dokumente
- Die Version 1.3.3 ist mit den erwähnten Java Server-Erweiterungen ausgiebig ausgetestet worden.
- Wird in der Zielumgebung bereits eingesetzt

2.2.2 JRE

Da das Java Runtime Environment Bestandteil des JDK ist, wird natürlich dieses verwendet (siehe Kapitel 2.1.1 (JDK) auf Seite 5).

2.2.3 Datenbank Server (RDBMS)

Es wird, wie in [1], Kapitel 2.4.1 (Laufzeitumgebung) auf Seite 7 vorgeschlagen, PostgreSQL (in der aktuellen Version 6.4.2) verwendet (siehe auch [22]).

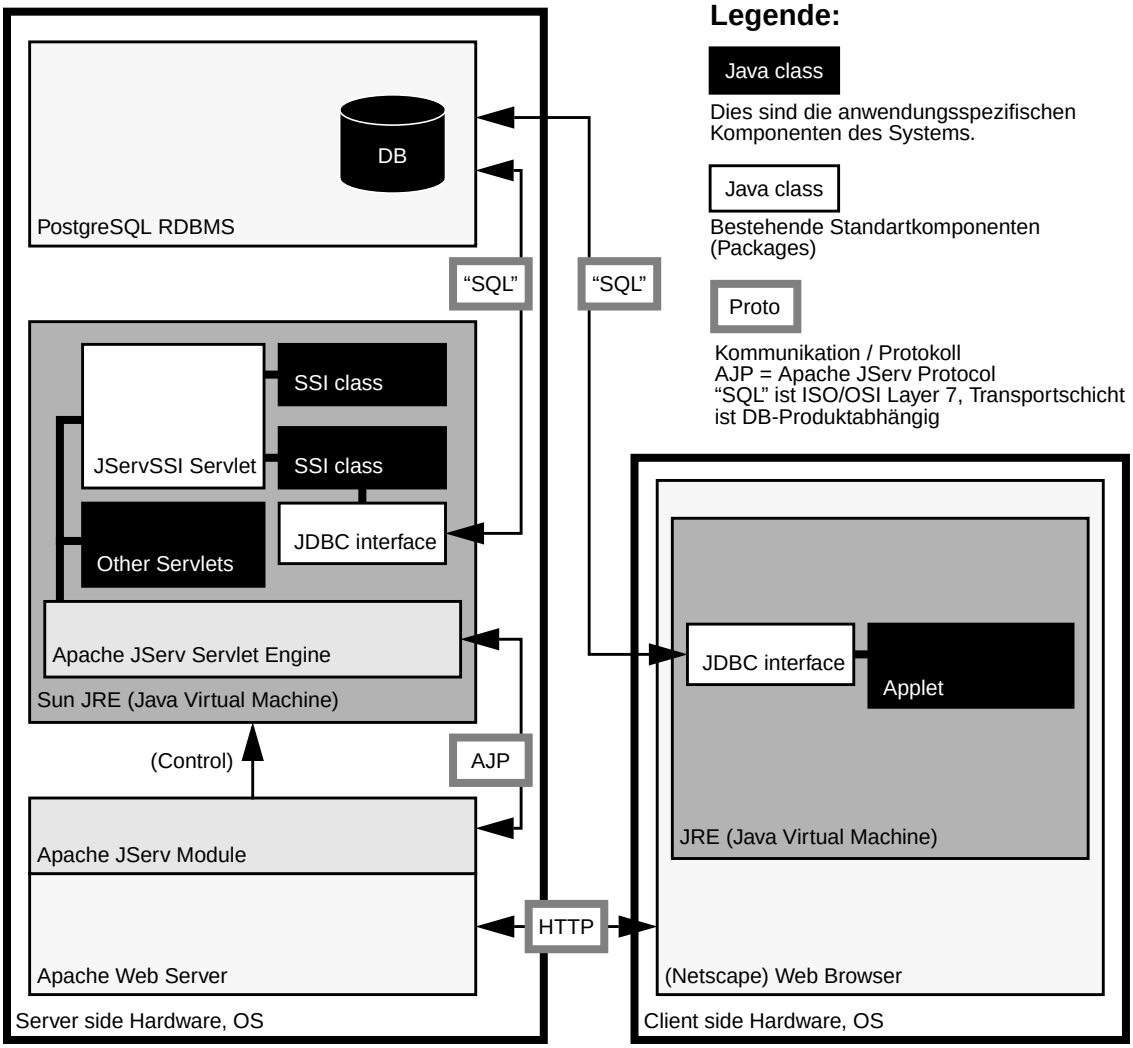
2.2.4 JDBC Treiber

Ist für die Datenbankanbindung von Java-Programmen über JDBC erforderlich. Es wird der mit PostgreSQL mitgelieferte Typ 4 Treiber (Typ 4 heisst, dass der Treiber in Pure Java geschrieben ist) von Peter T. Mount [23] in der Version 6.4 verwendet.

3 • System Design

3.1 Übersicht

Folgend die Übersicht über die verwendeten Software-Komponenten:



Bemerkungen:

Obiges Diagramm zeigt alle beteiligten serverseitigen Prozesse auf demselben Host. Der Web Server, die Java Virtual Machine mit der Servlet Engine und das RDBMS sind jedoch 3 unabhängige Prozesse, welche auf verschiedenen Maschinen laufen können und mittels IP kommunizieren.

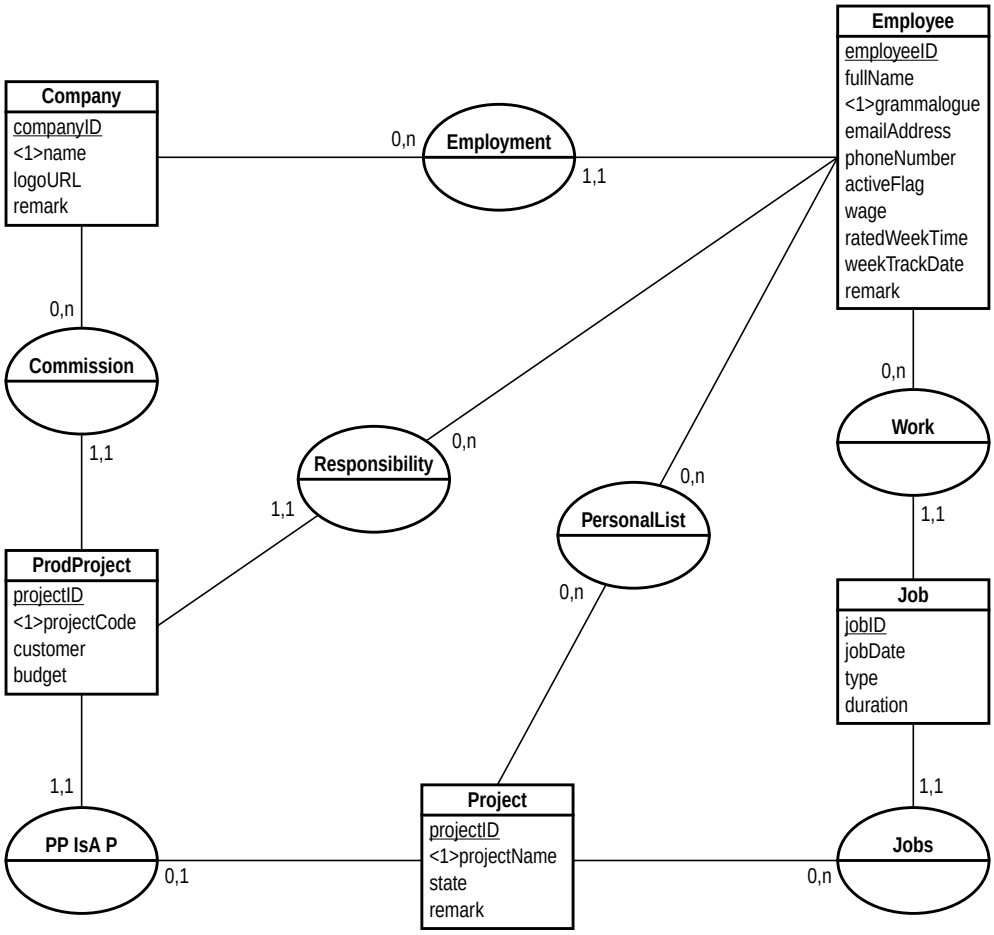
Werden diese Prozesse tatsächlich auf verschiedene Hosts (was möglich und zur Leistungssteigerung dienlich ist) verteilt, ist folgendes zu beachten:

- Auf dem Web Server Host ist ein SQL-Gateway erforderlich, da Java Applets nur mit Hosts direkt kommunizieren können, von welchen sie geladen wurden (Java Sicherheitsmechanismus).
- Die Java Virtual Machine mit der Servlet Engine muss “von Hand” gestartet werden, da das Apache JServ-Modul dies nicht über Hostgrenzen hinweg automatisch vornehmen kann (Control).

3.2 Datenbank

3.2.1 ERM

Um die erfassten Daten abzulegen, wird folgende Datenstruktur verwendet:



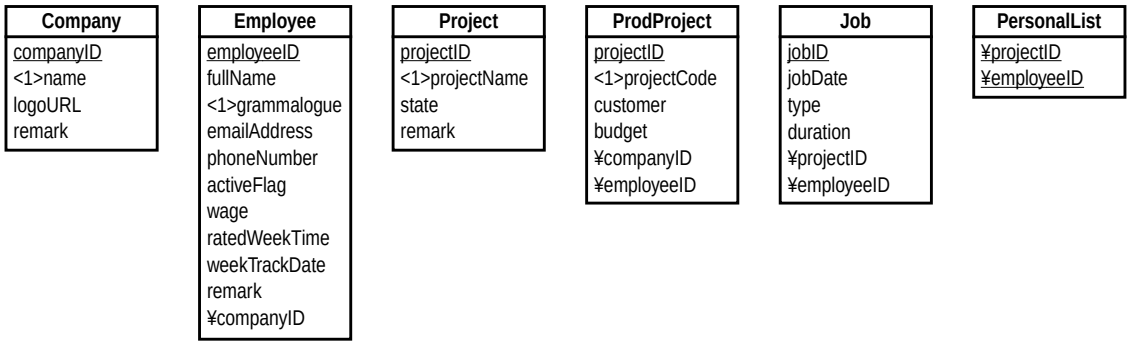
Bemerkungen:

- grammalogue ist das Kürzel. Dieses ist zwar eindeutig, aber nicht umbedingt konstant und kann deshalb nicht als Schlüssel verwendet werden. Dasselbe gilt für name in Company.
- In Project ist eine eindeutige und konstante projectID erforderlich (Schlüssel müssen konstant sein), da verlangt ist, dass alle projektrelevanten Angaben veränderbar sein müssen (Siehe [1], Kapitel 4.2.3 (Projekt) auf Seite 12).
- Da es grundsätzlich produktive und nichtproduktive Projekte gibt, welche sich in den verwendeten Attributen unterscheiden, wurde hier Project mittels einer IsA-Beziehung aufgespalten.
- Die Job-Datenstruktur wurde gegenüber dem Pflichtenheft ([1], Kapitel 4.2.4 (Job) auf Seite 13) abgeändert. Anstelle der typspezifischen Zeitfelder für Abklärung und Design wird ein universelles Zeitfeld duration eingesetzt, aber der Typ type mitgespeichert. Dies erlaubt eine einfache Erweiterung des Systems zur Unterstützung weiterer Typen, ohne die Datenstruktur verändern zu müssen. Mehrere Zeittypen am selben Tag ergeben so mehrere Einträge. Die Datenmenge wird dadurch nicht wesentlich grösser, da es höchst selten ist, dass der selbe Entwickler an einem Projekt am selben Tag Abklärungen und Design macht (üblicherweise vergehen zwischen diesen beiden Phasen einige Tage).

- Das Feld Project.state hält den momentanen Projekttyp fest und dient als Ausgangswert für Job.type.
- Die in [1], Kapitel 3.2.1 (Erfassen) auf Seite 10 beschriebene konfigurierbare, persönliche Liste wird mittels der PersonalList -Relation realisiert. Siehe auch Kapitel 3.2.4 (Definition der state-/type-Werte) auf Seite 10.
- Employee wurde gegenüber dem Pflichtenheft um weekTrackDate ergänzt, um die aktuelle Woche (zur Zeit bearbeitete Woche) festzuhalten. Dies erlaubt die Kontrolle über den Wochenabschluss (Siehe Pflichtenheft, Kapitel 3.2.1 (Erfassen) auf Seite 10). Ferner kann damit beim Erstellen eines neuen Benutzers damit das "Startdatum" (Arbeitsanfang) festgelegt werden.
- Es wurden die Felder Employee.remark und Company.remark hinzugefügt. Dies erlaubt z.B. eine Bemerkung zu einem inaktiven (activeFlag = FALSE) Entwickler anzubringen.

3.2.2 RDM

Aus dem ERM (Kapitel 3.2.1 (ERM) auf Seite 7) lässt sich nun folgendes RDM ableiten:



Bemerkungen:

- Da es sich bei der Project zu ProdProject -Beziehung um eine reine (freiwillige) 1:1-Beziehung handelt (ProdProject ist eine Erweiterung von Project, Vererbung), kann als Schlüssel in ProdProject der Fremdschlüssel projectID verwendet werden.
- Das hier gezeigte RDM wurde auf 4NF (vierte Normalform) normalisiert und ist deshalb für die Umsetzung auf einem RDBMS geeignet.

3.2.3 Definition der Datentypen

Hier zusammenfassend die Definition aller Attribute und deren Beschreibung (siehe auch [1], Kapitel 4.2 (Struktur) auf Seite 12):

Entität	Name	Typ	Optionen	Beschreibung / Bemerkung
Company	companyID	serial(1)		Eindeutige Firmenkennung (intern)
	name	varchar(32)	UNIQUE	Firmenname
	logoURL	varchar(128)	NULL	URL auf das Firmenlogo (Eine URL kann sehr lang sein)
	remark	varchar(128)	NULL	Bemerkungen zur Firma

Entität	Name	Typ	Optionen	Beschreibung / Bemerkung
Employee	employeeID	serial(1)		Eindeutige Entwicklerkennung (intern)
	fullName	varchar(32)		Voller Name des Entwicklers
	grammalogue	character(3)	UNIQUE	Das firmenintern verwendete Kürzel (2-3 Zeichen)
	emailAddress	varchar(64)	NULL	Email-Adresse
	phoneNumber	varchar(32)	NULL	Interne Telefonnummer (genügend Platz ermöglicht es, auch externe Mitarbeiter mit einer externen Nummer abzulegen)
	activeFlag	boolean	TRUE	Entwickler aktiv / passiv
	wage	real	0.0	Interner Stundenansatz [CHF]
	ratedWeek-Time	real	40.0	Wochenarbeitszeit [Std]
	weekTrackDate	date	NOW	Kennzeichnet die aktuelle (bearbeitete) Woche
	remark	varchar(128)	NULL	Bemerkungen zum Entwickler
	¥companyID	integer		Referenz auf die Arbeitgeberfirma (entspricht der Employment-Relation)
Project	projectID	serial(1)		Eindeutige Projektkennung (intern)
	projectName	varchar(32)	UNIQUE	Projektname (dies sind meistens Abkürzungen und prägnante, kurze Namen. Diese können bei nichtproduktiven Projekten auch länger sein, siehe Bild in [1], Kapitel 2.1.3 (Ist-Zustand) auf Seite 6, Spalte “Activity”)
	state(2)	integer	1000	Projektzustand (Typ) wird vom Projektverantwortlichen bestimmt
	remark	varchar(128)	NULL	Zusatzinformation / Bemerkung zu einem Projekt
ProdProject	projectID	integer		Referenz auf das Projekt
	projectCode	varchar(16)	UNIQUE	Projektcode oder Chipcode, z.B. L5A8260, TH4011.1C
	customer	varchar(32)		Name des Kunden (Auftraggebers)
	budget	real	0.0	Budgetierter Aufwand für die Entwicklung [CHF]
	¥companyID	integer		Referenz auf die Auftragnehmerfirma (entspricht der Commission-Relation)
	¥employeeID	integer		Referenz auf den Projektverantwortlichen (entspricht der Responsibility-Relation)
Job	jobID	serial(1)		Eindeutige Jobkennung (intern)
	jobDate	date		Datum
	type(1)	integer		Typ der gearbeiteten Zeit (wird von state im Project übernommen)
	duration	real		Gearbeitete Zeit in Stunden
	¥projectID	integer		Referenz auf das bearbeitete Projekt (entspricht der Jobs-Relation)
	¥employeeID	integer		Referenz auf den Entwickler (entspricht der Work-Relation)
PersonalList	¥projectID	integer		Verbindet einen beliebigen Employee mit einem beliebigen Project
	¥employeeID	integer		

1. Automatisch generierte, eindeutige Referenz, siehe [22]
2. Siehe Tabelle Kapitel 3.2.4 (Definition der state-/type-Werte) auf Seite 10 für die möglichen Werte und deren Bedeutung

Bemerkungen:

- Die hier verwendeten Datentypen entsprechen bis auf den Typ “serial” (implizite Sequenz in PostgreSQL, [22]) den SQL92-Typen. Diese Tabelle dient als Ausgangslage für die Definition der Datenbanktabellen. Die effektiv verwendeten Typen können dann datenbankspazifisch optimal gewählt werden, sofern sie mindestens die Eigenschaften der hier vorgeschlagenen SQL92-Typen erfüllen.
- Bei der Namenswahl der Relationen und Felder ist darauf zu achten, dass keine SQL-Schlüsselworte (wie z.B. DATE oder DAY) verwendet werden, da dies bei einigen Datenbanken zu Problemen führen kann.

3.2.4 Definition der state-/type-Werte

Die Definition der state-/type-Werte und deren Bedeutung sind folgend aufgelistet:

state **Wert**	**Bedeutung**(1)	**Bemerkung**
0	Unproduktives Projekt	Kein ProdProject vorhanden
1000	Produktives Projekt, Abklärungsphase	Projektnamen und Bezeichnung sind normalerweise noch nicht definitiv
1001	Produktives Projekt, Designphase	Das Projekt wurde gebucht und ist in Arbeit
1002	Produktives Projekt, Nacharbeiten	Das Projekt wurde bereits abgeschlossen (und verrechnet)

1. (siehe auch [1], Kapitel 4.2.3 (Projekt) auf Seite 12)

Bemerkungen:

- Die Werte wurden so definiert, dass die logische Gruppierung bei einer allfälligen Erweiterung erhalten werden kann.

3.2.5 Referenzielle Integritätsbedingungen

Es müssen folgende referenziellen und semantische Integritätsbedingungen erfüllt werden:

LÖSCHEN C: Cascade N: Nullify R: Restricted	**Company**	**Employee**	**Project**	**ProdProject**	**Job**	**PersonalList**
Company	-	R	-	R	-	-
Employee	-	-	-	R	R	C
Project	-	-	-	C	C	C
ProdProject	-	-	-	-	-	-
Job	-	-	-	-	-	-
PersonalList	-	-	-	-	-	-

ERZEUGEN E: must Exist	**Company**	**Employee**	**Project**	**ProdProject**	**Job**	**PersonalList**
Company	-	-	-	-	-	-
Employee	E	-	-	-	-	-
Project	-	-	-	-	-	-
ProdProject	E	E	E*	-	-	-
Job	-	E	E	-	-	-
PersonalList	-	E	E	-	-	-

Bemerkungen:

- Eine Firma kann nur gelöscht werden, wenn sie keine Mitarbeiter und Projekte mehr hat (aber eine Firma kann jederzeit umbenannt werden und behält ihre Einträge).
- Ein Entwickler kann nur gelöscht werden, wenn er für kein Projekt mehr verantwortlich ist und er an keinem laufenden Projekt mitgearbeitet hat (da sonst der Bezug zu der zu belastenden Firma verloren geht). Dieselbe Problematik stellt sich bei einem Firmenwechsel eines Entwicklers wie bei obiger Regel (1). Siehe hierzu auch [1], Kapitel 3.2.3 (Administration) auf Seite 11, Zeile “Mitarbeiter ändern”.
- Beim Einfügen eines ProdProject muss (E*) das entsprechende Projekt existieren und zusätzlich muss dessen state einen entsprechenden Wert (siehe Kapitel 3.2.4 (Definition der state-/type-Werte) auf Seite 10) aufweisen.
- Die referenzielle Integrität soll mittels “Rules” direkt auf dem RDBMS implementiert werden.

3.2.6 Semantische Integritätsbedingungen

	Company	Employee	Project	ProdProject	Job	PersonalList
Company	-	-	-	-	-	-
Employee	-	-	-	-	(1)	-
Project	-	-	-	(2)	-	-
ProdProject	-	-	-	-	-	-
Job	-	-	-	-	-	-
PersonalList	-	-	-	-	-	-

1. Employee:¥companyID darf nicht geändert werden, solange noch ein Job mit diesem Employee existiert.
2. Wenn Project.state geändert wird, sollte der dazugehörende ProdProject-Eintrag gelöscht (0) oder erzeugt (>0) werden.

Bemerkungen:

- Die semantische Integrität soll durch die Applikation gewährleistet werden.

3.3 Applikationen

3.3.1 Allgemeines

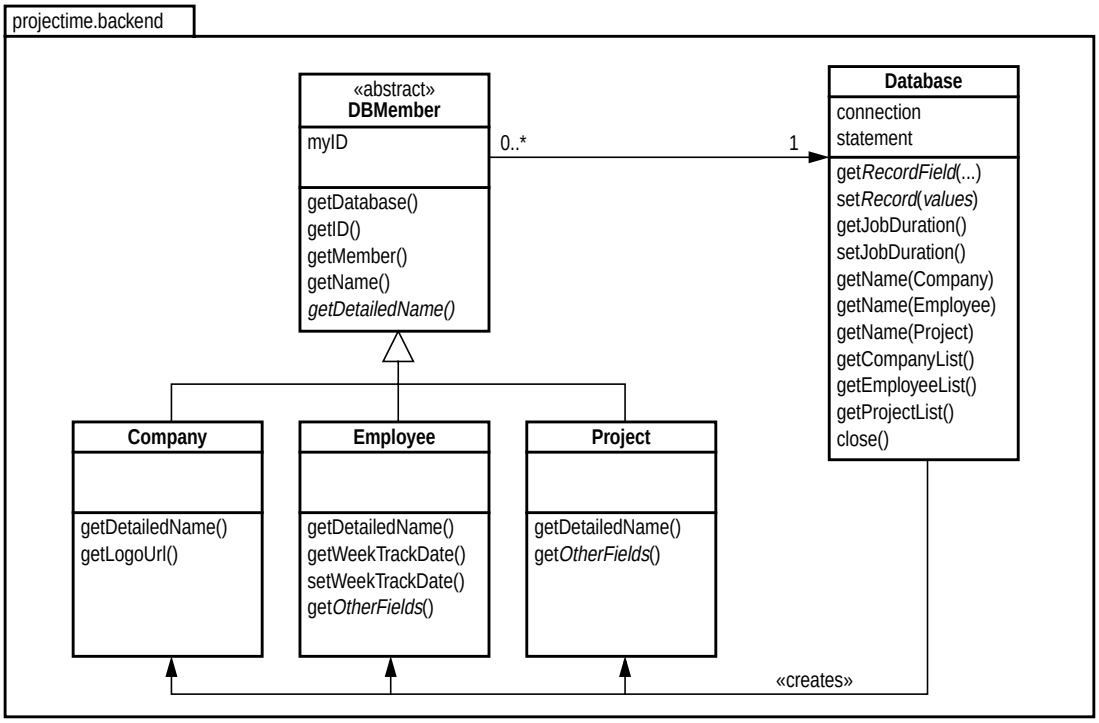
Das ganze Projekt umfasst folgende Teile auf Applikationsseite (Client-Seite):

- Zeiterfassungs-Applet (ist der direkt sichtbare Ersatz des Papier-Zeitblattes)
- Administrations-Applet (verwalten von Projekten und Entwicklern)
- Reportgeneratoren (Auswertung und Ausgabe)

All diese Teile arbeiten mit denselben Daten aus derselben Datenbank, so dass hier das Design der wichtigen, zentralen Klassen gesamthaft betrachtet wird.

3.3.2 Zentrale Klassen

Folgendes Klassendiagramm soll die Grundzüge verdeutlichen. Es enthält deshalb nicht alle Klassen, Methoden und Attribute.



Bemerkungen:

Drei wichtige Kenngrössen sind Firma, Entwickler und das Projekt. Projekt und Entwickler bestimmen zusammen mit der Dauer und dem Datum die 4 Dimensionen eines Zeiteintrages (Job) in der Datenbank. Die Firma, welche einem Entwickler zugeordnet ist, erlaubt den Projektaufwand dieser zuzuordnen. Für diese drei Kenngrössen steht in der Datenbank je eine Tabelle bereit (Kapitel 3.2.2 (RDM) auf Seite 8).

Die Klassen Company, Employee, und Project sind Repräsentanten dieser Tabellen, d.h. Objekte dieser Klassen sind Repräsentanten von Tabelleneinträgen. Diesen Einträgen ist gemeinsam, dass sie eine eindeutige Identifikation aufweisen (companyID, employeeID und projectID). Diese Identifikation wird hier benutzt, um ein Objekt einem Datenbankeintrag zuzuordnen. Diese gemeinsame Eigenschaft wurde in der Basisklasse DBMember zusammengefasst.

Mit dieser Identifikation (mylD) ist nun jede dieser Klassen Company, Employee, und Project in der Lage, direkt auf all ihre Datenbankfelder (Attribute) zuzugreifen. Das hier vorliegende Design sieht nur den lesenden Zugriff (get) auf einzelne Attribute vor, wenn diese im Normalfall (d.h. abgesehen von der Administration) nicht auch geschrieben werden müssen (wie z.B. weekTrackDate, welches beim Abschliessen einer Woche im Zeiterfassungs-Applet).

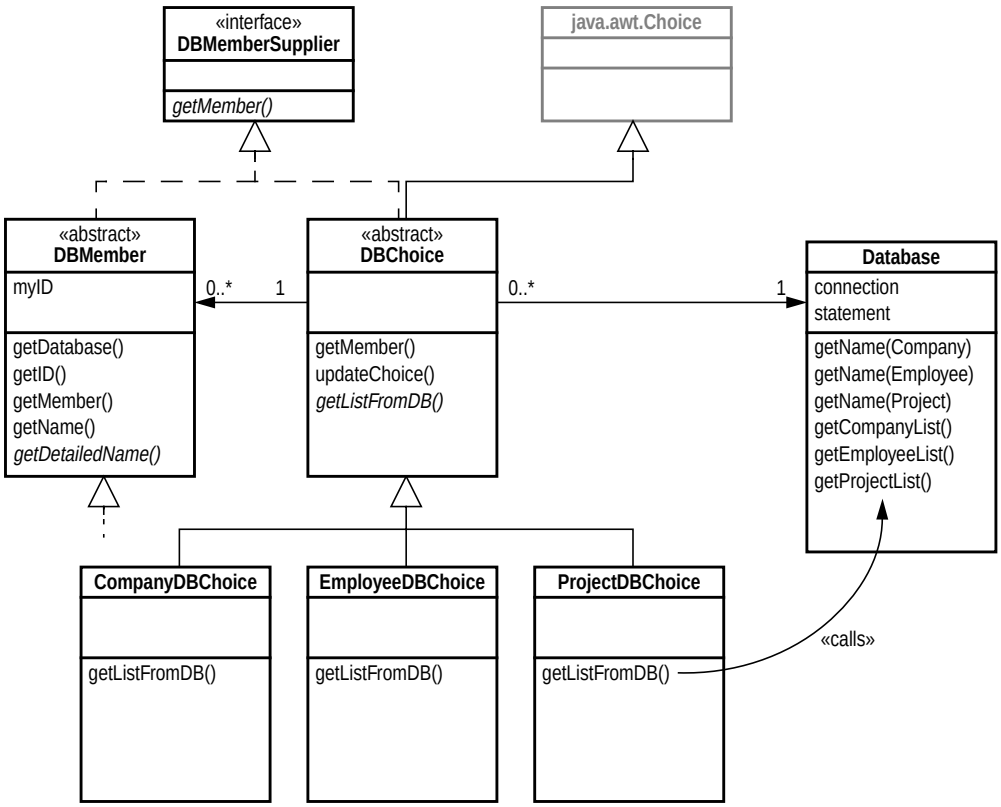
Die hierzu nötigen Datenbankabfragen (SQL über JDBC) sind in der Datenbank-Klasse Database gekapselt. Diese hält auch die Verbindung (connection) mit dem Datenbankserver während der ganzen Laufzeit. Dies minimiert den Overhead, den ein Verbindungsaufbau mit sich bringt.

Diese Datenbank-Klasse ist auch für das Erzeugen der Company, Employee, und Project - Objekte verantwortlich, da sie als einzige über die dazu nötigen Informationen (vom Datenbank-Server) verfügt. Denn jeder DBMember kennt seine Identifikation und “seine”

Datenbank-Klasse (sein Ursprung). Diese müssen zwingend beim Erzeugen eines DBMember Subklassen-Objektes mitgegeben werden.

3.3.3 Applet-spezifische Klassen

Wichtiger Bestandteil der Applets ist das GUI. Folgende Klassendiagramme sollen die Grundzüge verdeutlichen. Sie zeigen nur interessante Teilaspekte.
Hier soll aufgezeigt werden, wie ein(e) vom Benutzer selektierte(r/s) Firma / Entwickler / Projekt als DBMember -Objekt in die Applikation gelangt:



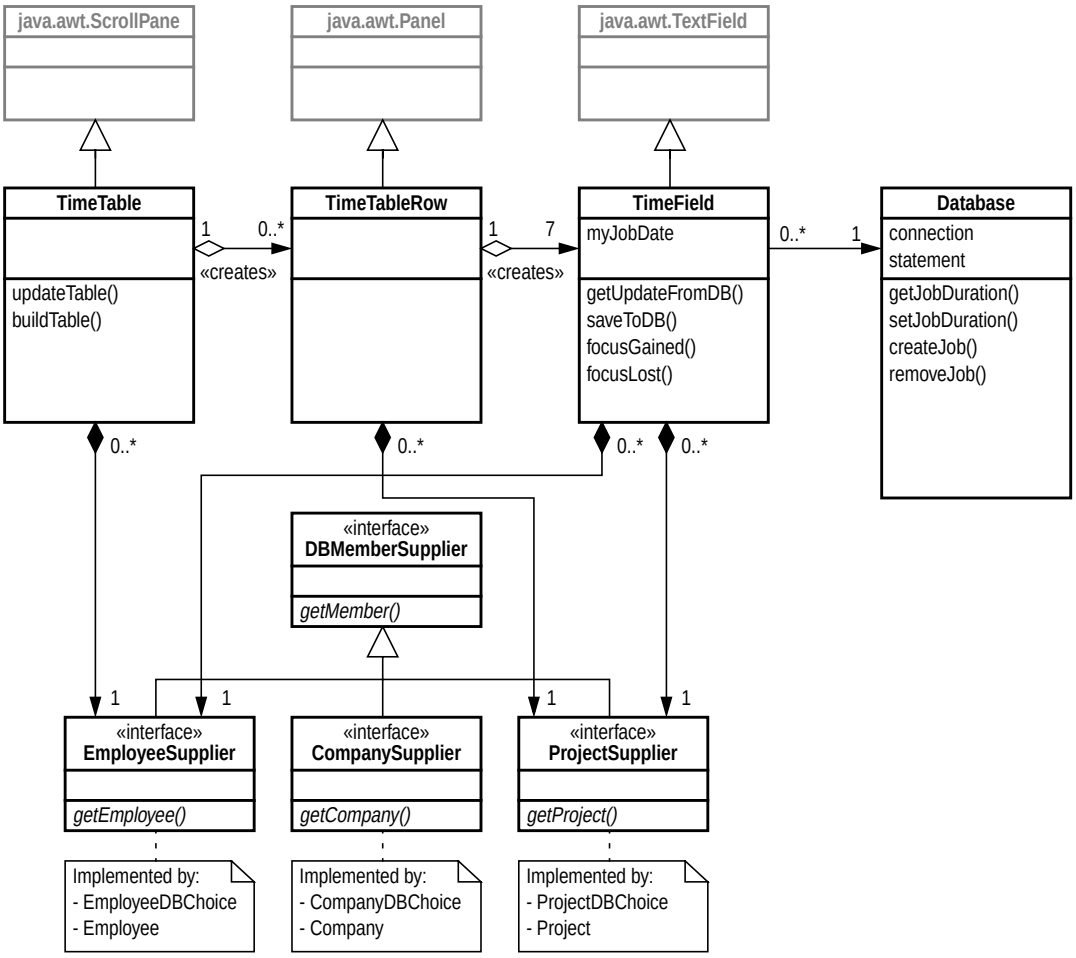
Bemerkungen:

In DBChoice wurde die Funktionalität von java.awt.Choice dahingehend erweitert, dass:

- Bei jedem Betätigen der Auswahl entsprechend mit getListFromDB() aus der Datenbank eine aktuelle Liste von DBMember Subklassen-Objekten geholt wird.
- Die in der Auswahl sichtbaren Einträge von jedem DBMember mittels getDetailedName() erfragt und dargestellt werden.
- Das der Wahl des Benutzers entsprechende DBMember Subklassen-Objekt mittels getMember() über das Interface DBMemberSupplier abgeholt werden kann.

Vorteil beim Verwenden des DBMemberSupplier Interfaces ist, dass auch direkt ein (konstantes) Objekt DBMember abgefragt werden kann (welches sich selbst liefert).

Hauptteil des Eingabe-Applets soll eine Tabelle sein, welche für die aktuelle Woche eines Entwicklers für jeden Wochentag und jedes seiner gewählten Projekte ein Zeiteingabefeld bereithält (siehe hierzu das bisher verwendete (papierene) Zeitblatt in [1], Kapitel 2.1.3 (Ist-Zustand) auf Seite 6).



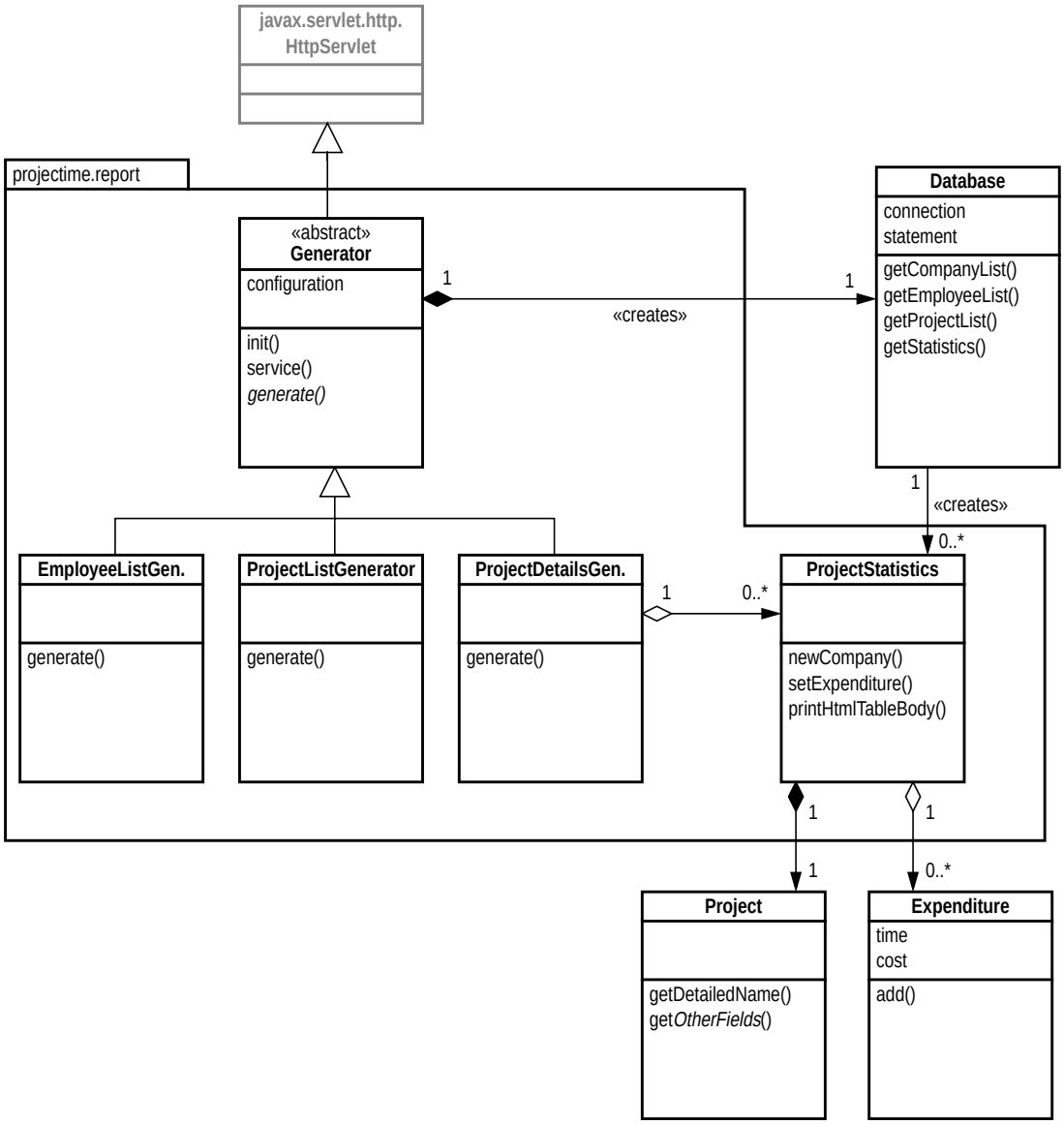
Bemerkungen:

Der Aufbau einer Tabelle läuft folgendermassen ab:

- TimeTable erhält über das EmployeeSupplier Interface seinen Employee und erhält von diesem dessen Liste von Projekten und das aktuelle Wochendatum (weekTrackDate). Nun wird für jedes Projekt ein TimeTableRow erzeugt und diesen das aktuelle Wochendatum mitgegeben.
- TimeTableRow seinerseits erzeugt für sein Projekt für 7 Tage ab erhaltenem Datum je ein TimeField und gibt diesen das entsprechende Datum mit.
- Das TimeField ist der Kern der ganzen Sache. Jedes dieser Felder gleicht seinen Inhalt selbständig mit seinem über Datum, Projekt und Entwickler spezifizierten Datenbankeintrag ab. Hierzu benutzt es die in Database zur Verfügung stehenden Methoden getJobDuration(), setJobDuration(), createJob() und removeJob().
- Diese Aktionen werden über die vom AWT automatisch aufgerufenen und von TimeField überschrieben Methoden focusGained() und focusLost() gesteuert, sobald der Benutzer das Feld selektiert oder wieder “verlässt” (etwas anderes selektiert).

3.3.4 Reportgeneratoren

Folgendes Klassendiagramm zeigt die Servlet-Klassen. Es enthält nicht alle Methoden und Attribute.



Bemerkungen:

Die Reportgeneratoren sollen als Inline Servlets (in eine HTML-Seite eingebetteter Aufruf, der im Gegensatz zum Applet nicht auf dem Web Browser, sondern auf dem Web Server ausgeführt wird, wird auch SSI genannt) verwendet werden. Dies erlaubt das freie Gestalten um den generierten Inhalt herum, ohne dabei des Servlet verändern zu müssen.

Obiges Diagramm zeigt 3 Generatoren. Deren Gemeinsamkeiten (Initialisieren der Datenbankanbindung, lesen der Konfiguration, Ausgabe von Kommentar) ist in der Basisklasse Generator zusammengefasst. Die Servlet Engine des Web Servers ruft die Methode service() auf, welche ihrerseits dann die spezialisierte generate()-Methode benutzt, um die gewünschte Ausgabe zu erzeugen.

ProjectStatistics wird von getStatistics() aus der Database-Klasse erzeugt und enthält ein etwas umfangreicheres Resultat. Das in ihr enthaltene Resultat soll diese Klasse gleich selbst mittels printHtmlTableBody() als Tabellenfragment in HTML ausgeben.

Expenditure ist eine reine Datenklasse, welche sich addieren kann.

4 • Implementation

4.1 Übersicht

Dieses Kapitel ist knapp gehalten, da die Implementation in den Source Files ausgiebig mittels Kommentaren dokumentiert wurde. Es sind hier nur einige Besonderheiten, Probleme und Begründungen aufgeführt, die zum vorliegenden Resultat geführt haben.

4.2 Datenbank

4.2.1 Datenstruktur erzeugen

Um diese Tabellen und Regeln auf der Datenbank zu definieren, wurde wie in [1], Kapitel 3.1.1 (Installation) auf Seite 9 gefordert ein SQL-Script erstellt, welches die Datenbank entsprechend konfiguriert.

••• `Siehe Datei RDBMS/create_db.sql`

Es wurde auch ein Script erstellt, welches ein paar Daten für erste Versuche in die Datenbank einfügt.

••• `Siehe Datei RDBMS/test_db.sql`

4.2.2 Probleme

Bei vorliegender Datenbank wurde folgendes Probleme gefunden:

- Es traten einige Probleme mit Rules in PostgreSQL auf, so dass einige Integritätsbedingungen gemäss Kapitel 3.2.5 (Referenzielle Integritätsbedingungen) auf Seite 10 nicht realisiert werden konnten.

••• `Siehe Datei RDBMS/create_db.sql`

4.3 Applikationen

4.3.1 Packages

Es wurden folgende Packages gebildet, um die Klassen etwas logisch zu gruppieren

- projectime: Enthält die Applet “Startklassen” und Projektallgemeine Klassen
- projectime.backend:Enthält Klassen der Datenbank und Datenrepräsentation
- projectime.gui: Enthält Klassen für die Bedienoberfläche der Applets
- projectime.report: Enthält Klassen der Servlets (Reportgeneratoren)

4.3.2 JAR-Files

Das mitgelieferte Makefile erzeugt aus den kompilierten Klassen ein JAR-File. Dessen Verwendung hat folgende Vorteile:

- Die Klassen werden darin komprimiert abgelegt. Dies hat eine wesentlich kürzere Ladezeit des Applets zur folge.
- Alle Klassen werden über eine Verbindung geladen, was den Web Server entlastet. Die traditionelle Variante (einzelne .class-Files) öffnet für jedes .class-File eine neue Verbindung.
- Die Handhabung einer einzelnen Datei ist wesentlich einfacher (Installation).

Für die beiden Applets werden separate JAR-Files erzeugt, obwohl sich dies in ein File packen liesse. Dies hat 2 Gründe:

- Da das Eingabe-Applet wesentlich häufiger gebraucht wird, wurde auf eine möglichst geringe Dateigrösse geachtet. Diese ist auch sehr klein: Nur 84 kByte!
- Ein getrenntes JAR-File für das Administrations-Applet lässt sich aus Sicherheitsüberlegungen in einem getrennten Directory ablegen und vom Web Server passwortschützen.

4.3.3 Datenbankanbindung

Wie in Kapitel 3.3.2 (Zentrale Klassen) auf Seite 12 beschrieben sind alle nötigen Datenbankabfragen (SQL über JDBC) in der Datenbank-Klasse *Database* gekapselt. Alle darin enthaltenen Abfragemethoden wurden in der vorliegenden Implementation synchronisiert (`synchronized`). Gründe:

- Beim benutzten JDBC-Treiber ist über eine Verbindung (connection) nur eine aktive Abfrage (statement) unterstützt.
- Arbeitet die Applikation nun mit mehreren Threads, könnten nun parallel mehrere Datenbankabfragen gleichzeitig erfolgen. Dies wird damit verhindert.

Bei der Programmerstellung wurde ein Fehler im JDBC-Treiber gefunden:

- Der JDBC-Treiber lieferte mit der `getUpdateCount()`-Methode nach einem ausgeführten `executeUpdate()` fälschlicherweise immer konstant 1 anstelle der Anzahl bearbeiteten Einträge zurück. Dieser Fehler wurde umgangen (siehe hierzu die Kommentare in Database.java). Mittlerweile ist aufgrund meiner Email an Peter T. Mount peter@retep.org.uk der Fehler behoben (die Funktion war noch nicht implementiert!).

••• `Siehe Datei Java/projectime/backend/Database.java`

4.3.4 Applets allgemein

Bei den Applets wurde darauf geachtet, dass sich die Datenbankparameter ohne Programmänderung anpassen lassen. Realisiert wurde dies über die `PARAM` -Attribute des `APPLET` -HTML Tags (siehe hierzu die HTML-Beispielseiten oder das Makefile).

••• `Siehe Datei WWW/entry.html`
••• `Siehe Datei WWW/admin.html`
••• `Siehe Datei Java/Makefile`

Das Eingabe- und Administrations-Applet lässt sich auch als eigenständige Applikation starten. Hierzu wurden die entsprechenden `main()` Methoden implementiert und eine eigene Frame-Klasse definiert, welche beim verlassen der Applikation die Datenbankverbindung sauber schliesst.

••• `Siehe Datei Java/projectime/gui/ExitableFrame.java`

4.3.5 Eingabe-Applet

Dieses Applet wurde auf verschiedene Plattformen getestet, namentlich:

- Netscape Communicator 4.51 auf Solaris 2.6
- Netscape Communicator 4.51 auf Solaris 2.5.1
- Sun HotJava 1.1.5 auf Solaris 2.6
- Netscape Communicator 4.51 auf Windows 95

Es war nur eine kleine Anpassung notwendig, um auf allen Plattformen das gewünschte Resultat zu erhalten.

••• `Siehe Datei Java/projectime/gui/TimeTableHeader.java`

Einigen Bedienelementen wurde “eigene Intelligenz” mitgegeben:

- So können die Knöpfe “Add Project” und “Remove Project” selbst überprüfen, ob sich ein selektiertes Projekt in die Liste einfügen oder aus der Liste entfernen lässt. Sie können diese Arbeit auch gleich auf Wunsch ausführen.

••• `Siehe Datei Java/projectime/gui/SmartButton.java`

- Der Knopf “Complete Week” zeigt beim Darüberfahren das aktuelle Wochentotal und überprüft, ob die Woche abgeschlossen werden kann oder nicht (und macht dies auch durch Farbe sichtbar). Er ist ebenfalls in der Lage, die Woche auf Wunsch abzuschliessen.

••• `Siehe Datei Java/projectime/gui/CompleteWeekButton.java`

4.3.6 Administrations-Applet

Diese Applet ist sehr einfach gehalten und deckt folgende Funktionen ab:

- Verwalten von Benutzern
- Verwalten von Projekten
- Korrektur von Datenbankeinträgen
- Eventuell Firmen erzeugen und löschen

(siehe hierzu Pflichtenheft, Kapitel 3.2.3 (Administration) auf Seite 11.)

Hier wurde ein Modul-orientierter Ansatz gewählt. Der Hauptfenster hat drei Bereiche: Links die Auswahl des Moduls, rechts die möglichen Aktionen und in der Mitte das der zu administrierenden Datenbanktabelle entsprechende Modul. Ein Modul umfasst die Benutzerschnittstelle und die gesamte Logik zum erstellen, ändern und löschen der entsprechenden Datenbankeinträgen. Es reagiert auf die Aktionen der links angeordneten Bedienelemente.

••• Siehe Datei Java/projectime/gui/DBAdministrator.java
••• Siehe Datei Java/projectime/gui/AdminPanel.java

4.3.7 Servlets allgemein

Wie bei den Applets wurde darauf geachtet, dass sich die Datenbankparameter ohne Programmänderung anpassen lassen. Realisiert wurde dies hier über Properties:

- Von der Servlet Engine werden die sich zugeordneten Properties angefordert und daraus der Dateiname der eigenen Properties-Datei erfragt.
- Die Datei wird geladen, alle darin definierten Parameter stehen zur Verfügung.

••• Siehe Datei Java/projectime/report/Generator.java
••• Siehe Datei Java/projectime.properties

4.3.8 Reportgeneratoren

Der in Kapitel 4.3.7 (Servlets allgemein) auf Seite 18 beschriebene Properties-Mechanismus wurde gleich benutzt, um die Ausgabe etwas zu parametrisieren (Farben und Darstellung der Tabellen).

••• Siehe Datei Java/projectime/report/*Generator.java
••• Siehe Datei Java/projectime.properties

5 • Dokumentation

5.1 Formate

Bei den in elektronischer Form vorliegenden Dokumentationsdateien wurde darauf geachtet, dass diese plattformunabhängig betrachtet werden können. Deshalb wurden Bericht und Pflichtenheft im PDF-Format (kann mit Adobes frei erhältlichen, auf vielen Plattformen verfügbaren Acrobat Reader [26] betrachtet werden), Sourcefile-Dokumentation im HTML-Format (kann mit jedem Web Browser betrachtet werden) bereitgestellt.

5.2 Bestandteile

Die vollständige Dokumentation dieses Projektes setzt sich aus folgenden Teilen zusammen:

5.2.1 Diplombericht

Dieser ist wie gefordert in Papierform geliefert worden und enthält auch das Pflichtenheft. Diese Dokumente sind auch auf der mitgelieferten CD-ROM in elektronischer Form vorhanden.

••• `Siehe Datei Documentation/Specification.pdf`
••• `Siehe Datei Documentation/Report.pdf`

5.2.2 Zusammenfassung

Wie gefordert liegt die Zusammenfassung (Kapitel 7 • (Zusammenfassung) auf Seite 22) auch im HTML-Format vor.

••• `Siehe Datei Documentation/zsfsg.html`

5.2.3 Source Code

Für die Dokumentation des Sourcecode wurde das JDK-Tool `javadoc` verwendet. Deshalb kann diese im Sourcecode oder im HTML-Format eingesehen werden und ist auf der mitgelieferten CD-ROM zu finden. Als Dokumentationsrichtlinie wurde [12] befolgt.

••• `Siehe Datei Java/HTML_doc/index.html`

5.2.4 Makefile

Das Makefile zum automatischen Erstellen der benötigten Dateien enthält Hinweise zur Installation und Verwendung. Ebenfalls ist daraus ersichtlich, wie die .jar-Files aufgebaut werden.

••• `Siehe Datei Java/Makefile`

5.2.5 SQL Script

Die SQL Script zum Erstellen der Datenstruktur (siehe auch Kapitel 4.2.1 (Datenstruktur erzeugen) auf Seite 16) enthält zusätzliche Informationen.

••• `Siehe Datei RDBMS/create_db.sql`

5.2.6 Properties

Die Reportgeneratoren beziehen ihre Konfigurationsinformationen aus einer eigenen Datei. Beiliegende Beispieldatei enthält alle möglichen Optionen und ist mit Kommentaren versehen.

••• `Siehe Datei Java/projectime.properties`

5.2.7 Applet-Beispielseiten

Es wurden HTML-Seiten geschrieben welche die erstellten Applets einbinden. Diese Seiten enthalten weitere Informationen wie eine Schnellanleitung zum entsprechenden Applet oder bekannte Probleme und deren Lösung.

••• `Siehe Datei WWW/entry.html`
••• `Siehe Datei WWW/admin.html`
••• `Siehe Datei WWW/reports/*.jhtml`

6 • Schlussbetrachtung

6.1 Anforderungen

Die Anforderungen konnten grösstenteils erfüllt werden. Der Vergleich mit dem in [1] definierten Anforderungskatalog (Kapitel 3.2 (Detaillierte Funktionsbeschreibung) auf Seite 9) sieht wie folgt aus:

6.1.1 Erfassen

Funktion	P	Beschreibung
User selektieren	1	Erfüllt
Projekt selektieren	1	Erfüllt (Tabellenzeilen einem Projekt zugeordnet)
Tag selektieren	1	Erfüllt (Felder einer Zeile einem Tag zugeordnet)
Zeit erfassen	1	Erfüllt
Woche abschliessen	1	Erfüllt
Projekt in persönliche Liste aufnehmen	2	Erfüllt
Projekt aus persönlicher Liste entfernen	2	Erfüllt

6.1.2 Auswerten

Funktion	P	Beschreibung
Gesamtaufwand auf einem Projekt	1	Erfüllt
Arbeitszeit der Entwickler	1	Erfüllt
Status-Report	1	Erfüllt
Arbeitsprofil des Engineering	2	Nicht erfüllt
Budget überprüfen	2	Teilweise erfüllt (Nicht alle geforderten Details ersichtlich)
Wochenauslastung eines Entwicklers	3	Nicht erfüllt
Bezogene Ferienstunden	3	Nicht erfüllt

6.1.3 Administration

Funktion	P	Beschreibung
Authentifizierung	1	Erfüllt (über Web Server)
Passwort ändern	1	Erfüllt (muss auf dem Web Server vorgenommen werden)
Werte korrigieren	1	Erfüllt
Neues Projekt eröffnen	1	Erfüllt
Projekt löschen	1	Erfüllt
Projekt ändern	1	Erfüllt
Mitarbeiter erzeugen	1	Erfüllt
Mitarbeiter löschen	1	Erfüllt
Mitarbeiter ändern	2	Erfüllt
Mitarbeiter (de)aktivieren	2	Erfüllt
Firma erzeugen	3	Nicht erfüllt
Firma löschen	3	Nicht erfüllt

6.2 Verbesserungsvorschläge

In der Implementationsphase wurde bemerkt, dass die Verwendung der State-Variable nicht ideal gelöst wurde. Viel flexibler und durchgängiger wäre die Trennung von der produktiv/ nicht produktiv-Kennung und der Designphase. Würde so nun die Designphase in einer eigenen Tabelle abgebildet, könnte dies flexibel erweitert und die statistische Auswertung nach Wunsch verfeinert werden.

6.3 Inbetriebnahme

Das System wurde am 8. April 1999 provisorisch in Betrieb genommen. Der offizielle Wechsel von der papierbasierten Erfassung auf ProjecTime wird voraussichtlich auf den 1. Mai 1999 erfolgen.

6.4 Persönliches

Da ich weder in objektorientiertem Design, noch in Java, noch mit Datenbanken Praxiserfahrung mitbrachte, war dieses Projekt eine sehr grosse Herausforderung. Erschwerend war, dass bei meinem Arbeitgeber keine Software erstellt und somit auch keine Software-erfahrene Leute weiterhelfen und Ratschläge geben konnten. Zum Glück standen mir die Dozenten der Software Schule Schweiz zur Seite.
Als grösstes Handicap erwies sich die mangelnde Erfahrung in der Designphase. In dieser blieb ich stecken und so war der Ratschlag eines Dozenten, an die Implementation zu gehen, das einzig Richtige.

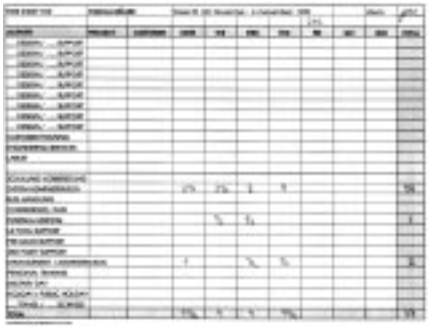
6.5 Fazit

Das Projekt ist in jeder Hinsicht ein Erfolg. Die Auftraggeberfirma ist nun im Besitze eines leistungsfähigen und flexiblen Projekt-Zeiterfassungssystems und ich konnte das erste Mal ein grösseres Softwareprojekt realisieren und mein in der SWS erworbenes Wissen anwenden und vertiefen.
Java scheint mir tatsächlich eine sehr ausgereifte Plattform zu sein und meine diesbezüglichen Erwartungen wurden nicht enttäuscht.

7 • Zusammenfassung

7.1 Einleitung

Wenn mehrere Leute mehrerer Firmen gleichzeitig an mehreren Projekten arbeiten, kann es sehr aufwendig werden, die Projektkosten entsprechend zuzuordnen. Dies war genau die Problemstellung des Auftraggebers, da unter einem Dach 3 verschiedene Firmen gemeinsam hochkomplexe Mikroelektronikkomponenten und Systeme entwickeln. Die auf Projekten gearbeitete Zeit wurde wochenweise auf einem Papierformular erfasst. Diese Zeiten wurden dann zur Auswertung von Hand in einer einfachen Excel-Tabelle eingetragen - eine fehleranfällige und sehr zeitaufwendige Arbeit, welche nur beschränkte Aussagen ermöglichte.



Auswertung von Hand

7.2 Zielsetzung

Es soll ein System zur Erfassung und Auswertung der projektbezogenen Arbeitszeit erstellt werden. Dieses soll die Zeiten durch ein einfaches Interface direkt am Arbeitsplatz erfassen und aus diesen erfassten Daten bei Bedarf automatisch eine gewünschte Auswertung erstellen. Eine solche Auswertung soll z.B der Buchhaltung eine saubere Nachkalkulation ermöglichen.

7.3 Realisierung

Es wurde ein intranet- und datenbankbasiertes System entwickelt, welches aus den folgenden Komponenten besteht:

- ProjecTime Entry: Datenerfassungs-Applet
- ProjecTime Report: Als inline-Servlet realisierte Reportgeneratorenfamilie
- ProjecTime Admin: Datenbankadministrations-Applet

Das ganze Projekt wurde mit frei erhältlicher Software realisiert (Sun JDK 1.1.7 Java Entwicklungsumgebung, PostgreSQL 6.4.2 RDBMS, Apache 1.3.3 Web Server).

ProjecTime

7.4 Besprechung

Die Anforderungen konnten erfüllt werden. Der seit dem 8. April 1999 laufende Versuchsbetrieb hat bisher keine Schwächen aufgezeigt.
Das Projekt ist in jeder Hinsicht ein Erfolg. Die Auftraggeberfirma ist nun im Besitze eines leistungsfähigen und flexiblen Projekt-Zeiterfassungssystems und ich konnte das erste Mal ein grösseres Softwareprojekt realisieren und mein in der SWS erworbenes Wissen anwenden und vertiefen.
Java scheint mir tatsächlich eine sehr ausgereifte Plattform zu sein und meine diesbezüglichen Erwartungen wurden nicht enttäuscht.