HOCHSCHULE DER MEDIEN

Projektbericht

System zur Erfassung und Auswertung von Arbeitszeiten

Von

Jörg Ingelfinger

Matrikelnummer: 15695

Fakultät: Druck und Medien

Studiengang: Medieninformatik

Bearbeitungszeitraum: Sommersemester 2008

Betreuender Professor: Prof Dr. Toenniessen

Durchgeführt bei

CONSIPIO® SOFTWARE ENGINEERING

Consipio Software Engineering GmbH
Robert-Bosch-Straße 32
74081 Heilbronn

Betreuende Mitarbeiter u. Ansprechpartner:

Herr Alexander Dimitriadis (Dipl.-Ing.)

Tel.: +49 (0)7131 / 203 503 - 1

E-Mail: ad@consipio.de

Herr Gerhard Ganz (Dipl.-Ing.)

Tel.: +49 (0)7131 / 203 503 - 2

E-Mail: gg@consipio.de

Inhaltsverzeichnis

1 Einleitung

Die folgende Ausarbeitung beschreibt die von mir, im Sommersemester 2008 im Rahmen des Moduls „Bachelor Thesis", durchgeführte Projektarbeit (EDV-Nr.: 20556).

Ideengeber für dieses Projekt war die Firma Consipio Software Engineering GmbH[1] aus Heilbronn, welche die Aufgabenstellung in Form einer Ausschreibung publiziert hatte. Consipio zeichnete ebenfalls für die Betreuung während der Projektphase verantwortlich.

1.1 Abgrenzung zur Bachelor Thesis

Meine Thesis wird auf die während dieses Projektes erstellte Software aufbauen. Das Ziel meiner Thesis wird es sein, die Software mit weiteren Features zu versehen, um deren Funktionalität zu erhöhen. Die durch die Studien- und Prüfungsordnung vorgesehene und verpflichtende inhaltliche Abgrenzung der Thesis gegenüber der Projektarbeit ist in sofern gewährleistet, als dass evtl. herzustellende Implementierungsanteile der Thesis ausschließlich über eine PlugIn- oder Komponentenschnittstelle, mit der aus diesem Projekt hervorgegangener Software, kommunizieren werden. Die Entwicklung einer hierfür geeigneten Schnittstelle, wird dann Bestandteil meiner Thesis sein.

1.2 Das Unternehmen

Das seit 2006 bestehende Unternehmen Consipio Software Engineering GmbH (nachfolgend Consipio genannt) ist Dienstleister für professionelle Softwareentwicklung und hat zum jetzigen Zeitpunkt sechs

[1] http://www.consipio.de

Mitarbeiter. Zu den Kunden gehören beispielsweise Firmen wie SAP Deutschland und die DHL Express Vertriebs GmbH & Co. OGH. Das Unternehmen setzt auf Mischungen aus sinnvoller Standardsoftware mit bei Bedarf integrierter Individualsoftware.

2 Projektüberblick

Ziel des Projekts war es, eine Software zu entwickeln, die projektbezogene Zeiterfassung ermöglicht.

Abbildung 1: Prinzip der Anwendung

In Abbildung 1 ist die Basisfunktionalität der Software veranschaulicht. Den Consipio-Mitarbeitern ist es mittels einer während des Projekts implementierten Desktopapplikation ermöglicht worden, ohne größeren Aufwand, Zeiten auf zu bearbeitende Projekte zu buchen. Die Datenhaltung wurde dabei an zentraler Position mit Hilfe einer Datenbank realisiert.

Die gesammelten Daten sollen später durch eine Reportingfunktionalität ausgewertet werden können, die ich im Rahmen meiner Bachelor Thesis in die Software implementieren werde.

Das gesamte Projekt wurde innerhalb des Unternehmens „ESIT“[2] genannt und die Clientanwendung erhielt den Namen „Timerecording“. Diese beiden Begriffe finden in den nachfolgenden Kapiteln entsprechende Verwendung.

[2] ESIT – Effort Spent In Time

2.1 Anforderungen und Lösungsansätze

Die Anforderungen an die zu entwickelnde Software wurden zu Projektbeginn in Zusammenarbeit mit Consipio definiert und während der Planungs- und Implementierungsphase umgesetzt.

Die Anforderungen lauteten wie folgt:

- **Rollenkonzept**:

  Es wurde ein Rollenkonzept, mit den folgenden Merkmalen entworfen:

  - Der Administrator (Admin) hat die Möglichkeit Kunden, Projekte und User in speziell dafür vorgesehenen Administratormasken zu verwalten.

  - Die Benutzer (User) können Arbeitszeiten direkt, d.h. parallel zur eigentlichen Arbeit buchen. Hierfür wurde eine Stoppuhrfunktion entwickelt. Außerdem sollte es möglich sein Arbeitszeit nachträglich zu erfassen, was über ein entsprechendes Formular realisiert wurde.

- **Benutzerrechte auf Projektebene**

  Es sollten vier verschiedene Rechtelevels integriert werden, welche Auswirkungen darauf haben, welche Aktionen ein Benutzer innerhalb eines Projektes ausführen darf. Die Rechtelevels sollten dabei die folgenden Ausprägungen besitzen:

Recht	Eigene Datensätze		Fremde Datensätze	
	lesen	schreiben	lesen	schreiben
ALL	X	X	X	X
RAW	X	X	X	
WRITE	X	X		
READALL	X		X	

- **Einfache Bedienung**

  Eines der Hauptanliegen von Consipio war die möglichst simple Gestaltung des TimeRecording-Client. Hintergrund dieser Anforderung war eine im Vorfeld dieses Projekts von Consipio durchgeführte Evaluation diverser Zeiterfassungs-Tools, bei der kein Produkt gefunden werden konnte, dass sowohl eine einfache Handhabung, als auch alle benötigten Features vereinte.

- **Favoritensystem**

  Um den Vorgang der Zeitbuchung so weit wie möglich zu simplifizieren entschloss man sich, dem User die Möglichkeit zu geben, häufig benötigte Projekttasks in persönlichen Favoriten organisieren zu können um so den Zeitaufwand für Buchungsvorgänge so minimal wie möglich zu halten.

- **Schnellbuchungsfenster**

  Zur Anzeige der Favoriten eines Benutzers, wurde neben dem eigentlichen Hauptanwendungsfenster ein QuickRe-

cord Fenster realisiert, welches die zuvor schon genannte Favoritenliste mit der Stoppuhrfunktionalität verbindet.

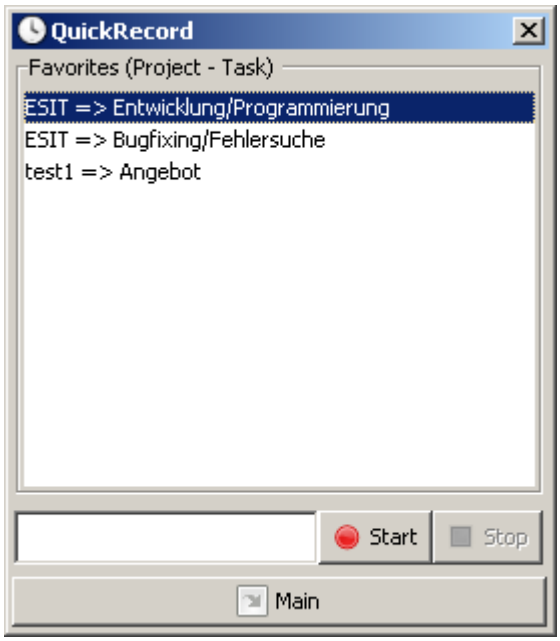


Abbildung 2: QuickRecord - Die Anzeige der Favoriten sowie die Bedienelemente der Stoppuhr.

- **Hauptanwendung**

  Das Hauptanwendungsfenster sollte die komplette Funktionalität in möglichst wenig Fenstern darstellen können.

  In der Hauptansicht werden daher alle nicht-administrativen Funktionen in einer zentralen Übersicht, der sogenannten „MainView", zusammengefasst.

  Die wichtigsten Bestandteile (vgl. Abbildung 3) der Hauptanwendung sind:

  1) Eine Übersicht über alle Kunden und die jeweils dazugehörigen Projekte.
  2) Anzeige aller Details zu einem in Punkt 1) gewählten Projekt.
  3) Anzeige aller Task des in Punkt 1) gewählten Projekts. Einschließlich einer Toolbar mit der hier ebenfalls notwendigen Stoppuhrbedienung, sowie der Möglichkeit,

Zeiten nachzutragen und einen Task in die persönliche Favoritenliste aufzunehmen.

4) Die zeitliche Ansicht zu einem Projekt, aufgeschlüsselt nach Jahr, Monat und gebuchten Stunden für den jeweiligen Monat.
5) Auflistung aller Datensätze des gewählten Monats in chronologisch absteigender Reihenfolge.
6) Anzeige des durch den Benutzer vergebenen Kommentars des in Punkt 5) gewählten Datensatzes.

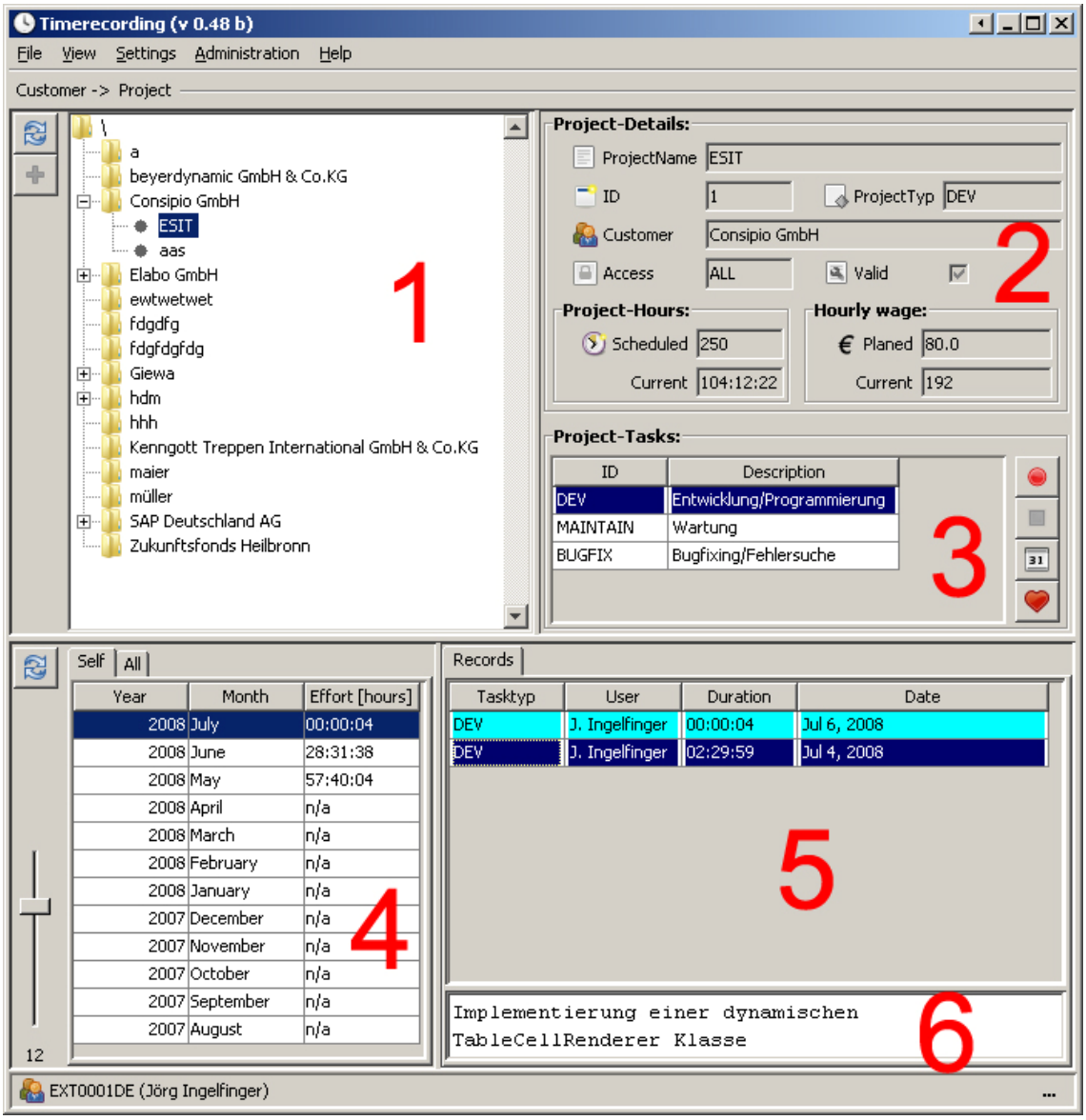


Abbildung 3: Das Hauptfenster der Applikation

Neben der beschriebenen Ansicht erreicht man über das Menü „Administration“ (die benötigte Benutzerrolle vorausgesetzt) den administrativen Bereich der Software, in dem Verwaltungsaufgaben und administratives im Allgemeinen untergebracht sind.

Über das Menü „Settings“ gelangt man zu der obligatorischen Einstellungsansicht für die Konfiguration von GUI-Parametern, Start-Optionen sowie Datenbank-Credentials.

Die Einstellungen werden in einer „.properties“ Datei lokal abgelegt.

- **Kommentar zu jedem Datensatz**

  Wie zuvor bereits erwähnt, sollte dem Benutzer die Möglichkeit eingeräumt werden, einen Datensatz mit einem Kommentar zu versehen. Dies hat den Hintergrund eine genauere Beschreibung der gebuchten Arbeitszeit zu ermöglichen, um später besser nachvollziehen zu können was jeweils im Detail gemacht wurde.

2.2 Eingesetzte Technologien

Client: Die Clientseite wurde komplett in Java unter ausschließlicher Verwendung von Swing-Komponenten programmiert.

Server: Auf Serverseite kam eine Installation der freien PostgreSQL Datenbank in Version 8.3 zum Einsatz.

Die Entscheidung für diese beiden Technologien war bereits vor Projektbegin durch Consipio getroffen worden.

2.2.1 Programmierung

Die Programmierung erfolgte durchgängig in Eclipse. Das Designen der Datenbank sowie das Anlegen von Views und das Konstruieren von SQL-Statements wurden unter Zuhilfenahme von PGAdmin durchgeführt.

2.2.2 GUI-Design

Die komplette Oberflächengestaltung wurde im kostenpflichtigen Tool JFormDesigner der Firma FormDev Software vorgenommen.

JFormDesigner ist ein auf Swing-Komponenten spezialisiertes WYSIWYG[3] - Tool das ein komfortables Arrangieren von Komponenten ermöglicht und darüber hinaus einen XML-Export der Oberflächen sowie durchgängige Sprachneutralität hinsichtlich Beschriftungen bietet.

Die so gewonnenen JFD[4]-Dateien können in den entsprechenden GUI-Klassen dann mit wenig Aufwand geparsed und die darin enthaltenen Komponenten automatisiert initialisiert werden. Dieses Verfahren hat den großen Vorteil, dass man sich in seinen selbst geschriebenen GUI-Klassen hauptsächlich um das Event-Handling kümmern kann und mit dem GUI-Code als solchen nur wenig aufhalten muss. Einzelne Komponenten werden im Wesentlichen nur beim Zuweisen von Datenmodellen oder Abfragen von Usereingaben in die Hand genommen. Die GUI-Klassen bleiben dabei frei von GUI-Code der Kategorien Anordnung, Positionierung, Farbgebung, Fontauswahl, Layout und sonstigen optischen Belangen.

[3] WYSIWYG – What You See Is What You Get

[4] JFormDesigner Export Format – enthält reguläres XML

Im Folgenden wird exemplarisch für alle GUI-Klassen, das Initialisieren einer aus JFormDesigner exportierten Oberfläche und den Zugriff auf eine darin enthaltene Komponente dargestellt:

```
public class StatusBar {
   private static final String jfdURL =
   "de/consipio/timerecording/resources/jfd/StatusBar.jfd";

   private FormCreator creator = null;
   private JPanel panel = null;
   private JLabel labelUser = null;

   public StatusBar(){
      initComponents();
      initLabelUser();
      …
   }

   private void initComponents(){
      try{
         creator = JFDHandler.getFormCreator(jfdURL);     (1)
         creator.setTarget(this);                         (2)
         panel = (JPanel)creator.create();                (3)
      }
      catch (Exception e){
         e.printStackTrace();
         log.error(e);
      }
   }

   private void initLabelUser(){
      labelUser = creator.getLabel("labelUser");          (4)
      labelUser.setText(…);
   }
   …
}
```

Erläuterung der gekennzeichneten Codezeilen:

(1) Initialisiert ein creator Objekt für die entsprechende .jdf Datei.

(2) Setzt das Target, das die entsprechenden EventHandling Methoden beinhaltet.

(3) Erzeugt das Panel.

(4) Zugriff auf eine Komponente (in diesem Fall ein JLabel) welche zuvor im JFormDesigner innerhalb dieses Panels

mit dem Namen „labelUser" angelegt wurde.

2.2.3 JDIC – JDesktop Integration Components

Um den TimeRecording-Client mit einem Tray-Icon und einem Tray-Menü zu versehen, kam aus JDIC[5] das Tray Package zum Einsatz.

Dadurch wird ein Minimieren der Applikation in den Tray ermöglicht. Am Tray-Icon ist ebenfalls zu erkennen ob momentan eine Zeiterfassung läuft. Außerdem kann die Applikation mit einem einfachen Klick jederzeit wieder aus dem Tray reaktiviert werden.

2.2.4 launch4j

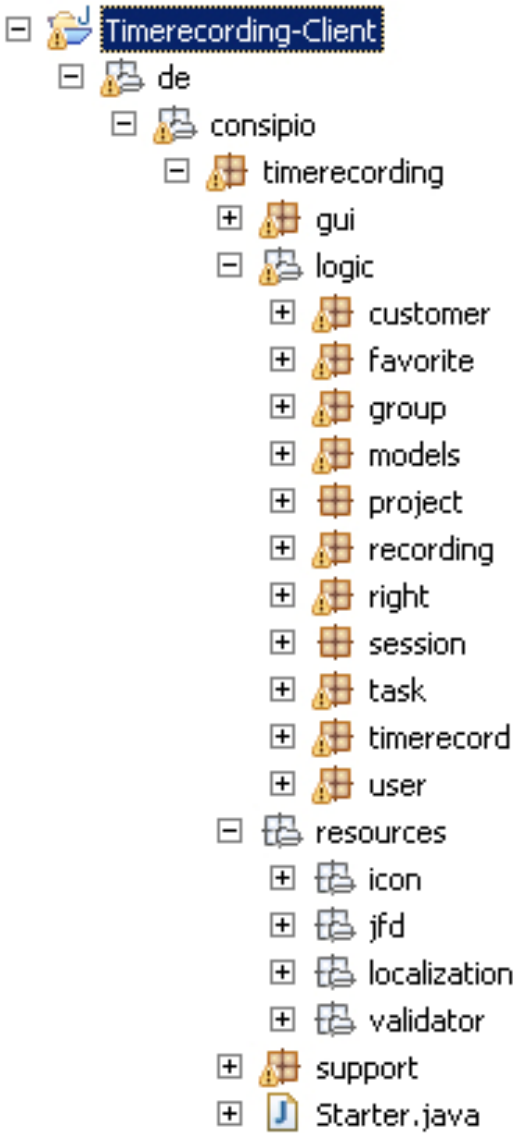
Mit Hilfe des Tools launch4j[6] lässt sich der Startvorgang der erzeugten .jar Datei in eine direkt aus dem jeweiligen Betriebssystem heraus zu startende Executable wrappen. Das hat unter anderem den Vorteile, dass die Applikation in der Prozessübersicht des Systems nicht als Java Virtual Machine erscheint, sondern wie eine native Anwendung unter eigenem Namen aufgeführt wird und sich dem Benutzer auch als Solche präsentiert. Außerdem können die allseits bekannten Java Icons durch eigene Anwendungsicons ersetzt werden.

[5] https://jdic.dev.java.net/

[6] http://launch4j.sourceforge.net/

3 Architektur

3.1 Allgemein



Das Projekt lässt sich grob in vier Bereiche aufteilen. Zum einen das *gui* Package das sämtliche GUI-Klassen für EventHandling, Rendering und Dialogsteuerung beinhaltet.

Zum anderen das Package *Logic,* welches alle einfachen Bean Klassen, die dazugehörigen Mapping-Klassen sowie diverse Datenmodelle für die Anzeige im View kapselt. Außerdem enthält es das *Recording* Package das sämtliche Zeiterfassungsvorgänge realisiert. Zusätzlich enthalt es Pakete zur Benutzersitzungsverwaltung und für das Rechtemanagement.

Auch das Package *resources* stellt einen eigenen Bereich dar und

Abbildung 4 : Die Projektstruktur des TimeRecording Clients

beinhaltet unter *icon* alle verwendeten Buttonsymbole.

Im Ordner *jfd* finden sich alle via JFormDesigner erstellen Oberflächenbeschreibungen und im Subpaket *localisation* alle Sprachfiles zur gesamten Oberfläche.

Das Package *validator* beinhaltet Informationen und Regeln im XML-Format für einen FormValidator der Firma Consipio, der für die Prüfung von Benutzereingaben verwendet wird.

Als letzten Hauptbereich kann das *support* Paket abgegrenzt werden, das diverse Hilfsklassen für Bereiche wie ConnectionHandling, Security, Konfigurationsmanagement, etc. beinhaltet.

Generell hatte ich den Anspruch die Software möglichst strukturiert und modular aufzubauen, um sie, auch bei steigender Komplexität übersichtlich, nachvollziehbar und erweiterbar zu behalten.

3.2 Datenmodel

Nachdem alle Anforderungen an die Software abgeklärt waren, wurde in einer frühen Phase des Projekts mit der Planung der Datenhaltung begonnen. Das Datenmodel der Applikation besteht aus mehreren relationalen Tabellen, die zueinander über Foreign-Key[7] Beziehungen in Verbindung stehen. Um Redundanzen zu vermeiden, wurden die Relationen an geeigneter Stelle normalisiert.

An dieser Stelle soll nun eine kurze Einführung der wesentlichen Abhängigkeiten innerhalb des Datenmodels (vgl. Abbildung 5) erfolgen. Auf einzelne Spalten einer Tabelle soll dabei nur in besonderen Fällen eingegangen werden.

Allgemein beruhen Schlüsselwerte in diesem Schema auf unterschiedlichen Postgres-Sequenzen die zur ID-Generierung für Benutzer, Kunden, Projekte und der eigentlichen Zeitdatensätze herangezogen werden.

[7] Foreign Key - Fremdschlüssel

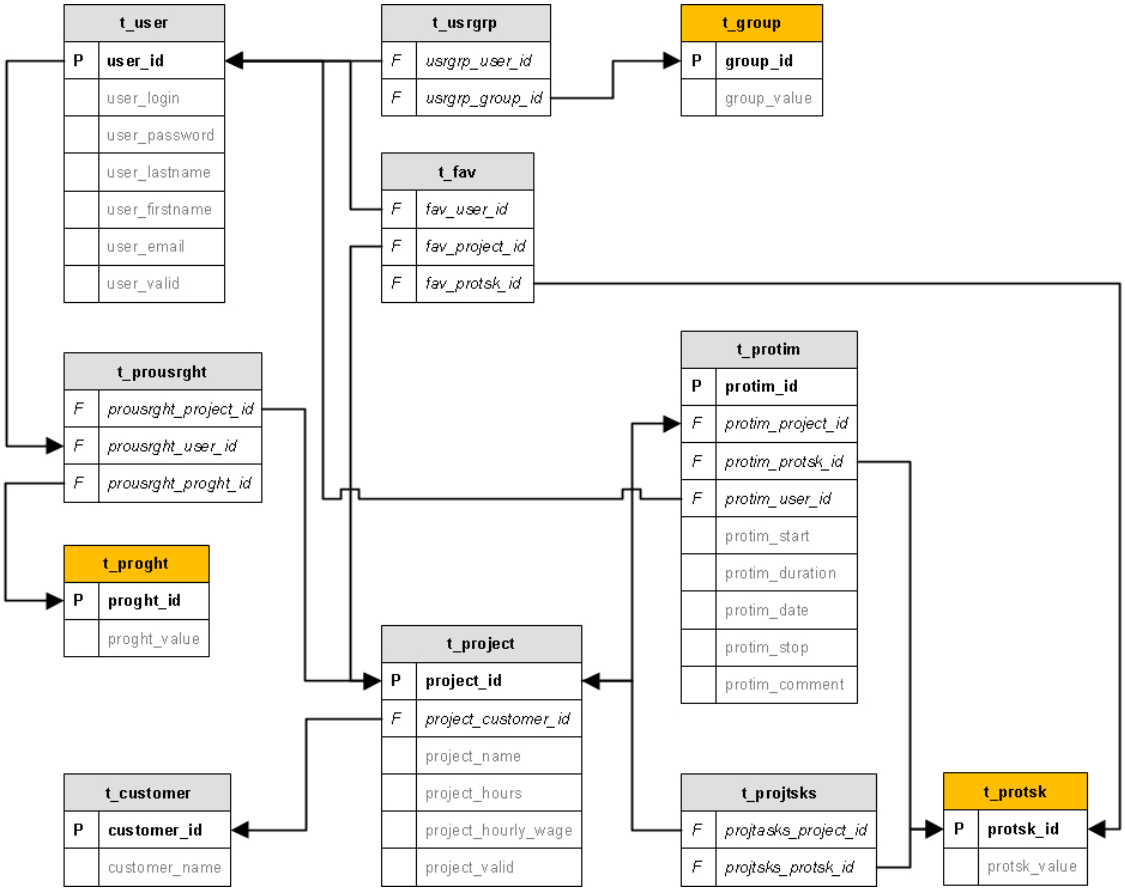


Abbildung 5 : Das (Kern)Datenbankschema für ESIT

Die Usertabelle besitzt als PK[8] eine *user_id*. Jeder User ist einer Gruppe zugeordnet. Diese Beziehung wird in *t_usrgrp* mit den jeweiligen Schlüsseln der Fremdtabellen abgebildet.

Analog, nur mit variierender Anzahl beteiligter Tabellen, verhält es sich bei weiteren Beziehungen, wie beispielsweise der Zuordnung von Benutzerrechten, Usern und Projekten oder Tasks mit Projekten.

Neben einer Tabelle für die Favoritenverwaltung (*t_fav*) findet sich noch die Tabelle *t_protim*, welche als Kern des Schemas betrachtet werden kann. In ihr werden sämtliche erfassten Zeiten abgelegt und ist damit die Komponente, mit dem erheblichsten Informationsgehalt.

[8] PK – Primary Key - Primärschlüssel

Ein Datensatz enthält im Detail eine ID, eine zugehörige Projekt-ID, eine Task-ID sowie eine User-ID. Mit Hilfe dieser Informationen kann ein Datensatz vollständig seinem jeweiligen Kunden, Projekt, Task und Benutzer zugeordnet werden.

Zur Ablage der eigentlichen Zeitinformationen werden weitere vier Parameter benötigt:

1. protim_start:(SQL-TimeStamp) - Startzeitpunkt
2. protim_stop:(SQL-TimeStamp) - Stopzeitpunkt
3. protim_date:(SQL–Date) - Datum der Erzeugung des Eintrags
4. protim_duration(Postgres-Intervall-Typ) – Differenz aus Stop und Start Werten - abgelegt in einem *HH:MM:SS* - Format.

Zusätzlich enthält jeder Eintrag in dieser Tabelle noch ein optionales Kommentarfeld, um eine Freitextbemerkung durch den Benutzer zu ermöglichen.

Auf eine Erläuterung ebenfalls verwendeter Views sowie weniger wichtigen Tabellen, soll an dieser Stelle aus Gründen der Übersichtlichkeit verzichtet werden.

3.3 Mapping

Dieses Kapitel beschreibt die für dieses Projekt wahrgenommene Lösung des *Impedance Mismatch*, sprich dem Problem der Umsetzung zwischen relationaler und objektorientierter Datensicht.

Aufgrund der schon zu Projektbeginn absehbaren, eher überschaubaren Anzahl an zu persistierenden Klassen, entschloss ich mich das Mapping komplett manuell, mit Hilfe von JDBC[9] umzusetzen.

[9] Java Database Connectivity

Für jede persistente Klassen wurde ein entsprechender Mapper programmiert, der alle benötigten Methoden zur Datenspeicherung und Datenabfrage beinhaltet. Diese Mapping-Klassen sind somit die einzigen Klassen im Projekt die SQL beinhalten, was der Übersichtlichkeit zugute kommt und darüber hinaus Wartungen und spätere Erweiterungen vereinfachen.

Am Beispiel des *TimeRecordMappers* (vgl. Abbildung 6) soll nachfolgend das Mapping der Datensatzklasse *TimeRecord* beispielhaft für alle anderen Klassen erläutert werden.

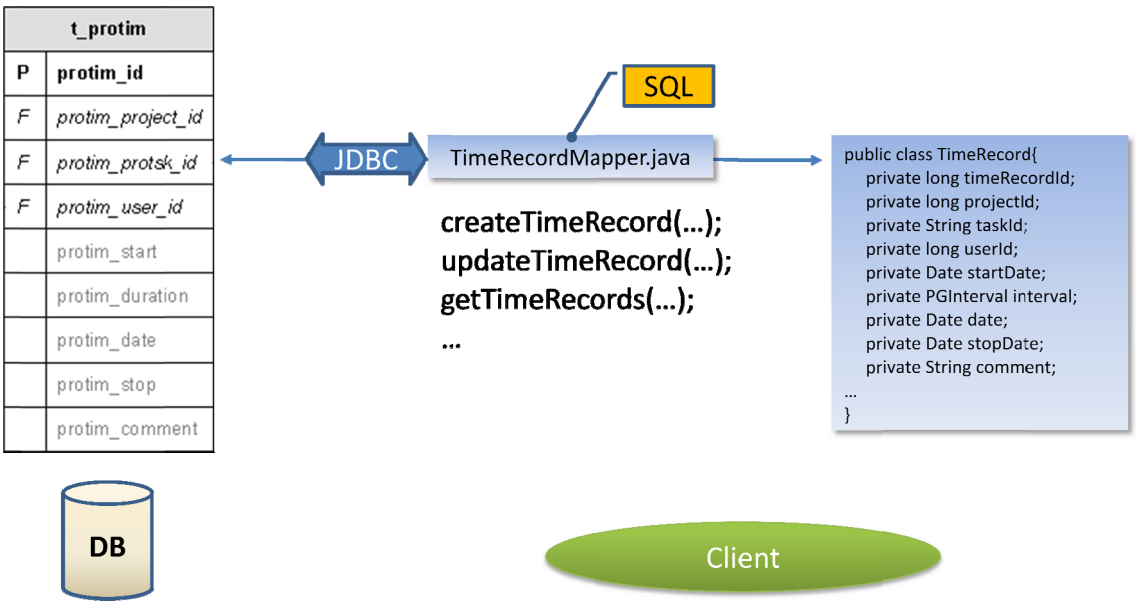


Abbildung 6 : Das Objekt-Relationale-Mapping des Systems

Die Klasse *TimeRecordMapper.java* bietet der Applikation verschiedene Methoden für Persistierungs- und Ladevorgänge von *TimeRecord*-Objekten. Beispielweise erwartet die Methode *createTimeRecord(…)* als Parameter ein *TimeRecord*-Objekt, dessen Attribute dann innerhalb der Methode in ein PreparedStatement eingefügt werden.

Das abgesetzte Statement legt dann datenbankseitig einen entsprechenden Eintrag in der zur Klasse gehörenden Tabelle *t_protim* an.

Der Ladevorgang, also die „Wiederbereitstellung" von Objekten funktioniert nach demselben Prinzip. Die Methode *getTimeRecord(…)* erwartet als Parameter beispielsweise einen Primärschlüssel (=Objekt-ID) für einen bestimmten Datensatz. Diese Abfrage liefert dann ein entsprechendes ResultSet der Datenbank zurück, das dann ausgewertet wird um ein Objekt mit den entsprechenden Attributen zu erzeugen.

Da es auf dem beschriebenen Weg, in „RoundTrips" gemessen, sehr aufwendig wäre z.B. alle Datensätze eines Projektes oder Mitarbeiters abzufragen, bieten die Mapping Klassen auch mengenorientierte Abfragemethoden an. Diese liefern dann Objekt-Container wie ArrayListen oder ähnliches zurück, welche dann wiederum als Grundlage für Datenmodelle bestimmter Komponenten wie JTable oder JTree verwendet werden können.

Das beschriebene Mapping-Verfahren wurde nach identischem Prinzip für alle Klassen implementiert.

4 Probleme

Probleme gab es während dieses Projektes im Großen und Ganzen keine, bzw. nichts worauf an dieser Stelle explizit eingegangen werden müsste. Die meist kleineren Probleme konnten dank guter Dokumentationen der eingesetzten Technologien oder entsprechend hartnäckigen Web-Recherchen relativ kurzfristig gelöst werden.

Allgemein war jedoch viel Einarbeitungszeit in die verschiedenen Swing-Komponenten von Nöten um vor allem deren jeweils zugrundeliegenden Datenmodelle verstehen und selbst umsetzen zu können.

Bedauerlicherweise wurde die ursprünglich angedachte ActiveDirectory[10] Anbindung zu Projektbeginn von Consipio wieder verworfen. Einige, im Vorfeld, diesbezüglich durchgeführte Recherchen und Tests meinerseits, konnten somit als „Fleißarbeit" betrachtet werden.

[10] Verzeichnisdienst der Firma Microsoft (kurz: AD)

5 Fazit und Ausblick

Das beschriebene Projekt war mein erstes größeres GUI Projekt in Form einer richtigen Desktopanwendung. Dementsprechend viel konnte ich über die Programmierung von grafischen Oberflächen und Java im Allgemeinen dazulernen. Auch die Datenbankthematik und der rege Umgang mit SQL, was ich seither eher vernachlässigt behandelt hatte, brachten mich fachlich voran.

Durch das Projekt konnte ich mein im Studium angeeignetes Wissen praxisbezogen ausbauen. Es war sehr interessant, die Abläufe und Vorgehensweisen innerhalb eines Software-Unternehmens kennenzulernen.

Die aus diesem Projekt resultierende Software wird nun seit 6 Wochen bei der Firma Consipio erfolgreich eingesetzt. Die geforderten Features wurden alle umgesetzt und sowohl die Betreuer, als auch die anderen Mitarbeiter des Unternehmens sind mit dem Ergebnis hochzufrieden.

Das ESIT-Projekt wird von mir bei Consipio weiterhin gepflegt, ausgebaut und im Rahmen meiner anstehenden Thesis um ein PlugIn für eine Reportingfunktionalität erweitert.

6 Abbildungsverzeichnis

7 Erklärung

Hiermit erkläre ich, dass ich die vorliegende Ausarbeitung selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt.

Heilbronn, den 10.07.2008
Jörg Ingelfinger