Projektdokumentation
zur
Winterprüfung 2007/08

Behandlungs-, Befund- und Diagnose-Verwaltungsprogramm mit Schnittstelle zur Software Vetera-Campus der Firma GP-Software

Prüfungsteilnehmer

René Mäurer (Azubi-Ident: 10703435449)
Waldsiedlung 22
15295 Brieskow-Finkenheerd

Ausbildungsbetrieb

Deutsche Rentenversicherung Bund
Ruhrstraße 2
10709 Berlin

Projektverantwortlicher

Dr. Manfred Sommerer

Ausführungszeitraum

22.10. bis 26.10.2007 und
05.11. bis 09.11.2007

Ausbildungsberuf

Fachinformatiker / Anwendungsentwicklung

Prüfungsausschuss

FIAN6

1. Einleitung

1.1 Ausbildungsbetrieb

Die Deutsche Rentenversicherung Bund ist ein Unternehmen des öffentlichen Rechts und mit über 57 Millionen Versicherten der größte Rentenversicherungsträger Europas. Sie beschäftigt über 70.000 Mitarbeiter.

Zu den Leistungen der Deutschen Rentenversicherung Bund zählen sowohl die Zahlung von unter anderem Alters-, Hinterbliebenen- und Erwerbsunfähigkeitsrenten als auch medizinische und berufliche Rehabilitation.

1.2 Auftraggeber und Projektumfeld

Das Projekt wurde in der IT-Abteilung des Fachbereichs Veterinärmedizin der Freien Universität Berlin in Auftrag gegeben und dort realisiert. Der Auftraggeber vor Ort war Herr Dr. Manfred Sommerer.

Die IT-Abteilung betreut circa 1.000 Mitarbeiter mit rund 650 Arbeitsplatzrechnern an drei Standorten (Düppel, Dahlem und Mitte) und 40 Servern in einem homogenen Windows-Netzwerk.

Sie ist verantwortlich für die Wartung und Instandhaltung der Rechner und Server, und steht den Mitarbeitern in Supportanfragen zur Verfügung.

2. Projektauftrag

2.1 Projektbeschreibung und Projektziel

Ziel des Projektes ist die Erstellung eines Behandlungs-, Befund- und Diagnose-Verwaltungsprogramm zur Erfassung und Bereitstellung von Daten für die Software Vetera-Campus der Firma GP-Software.

Es soll ein Programm entwickelt werden, das auf eine zentrale Datenbank zugreift und dort Behandlungen, Befunde, Diagnosen und Lokalisationen (Anatomie) einträgt beziehungsweise erfasst. Diese verschiedenen Klassifikationen können in bis zu 4 Ebenen gegliedert sein. Beispielhaft zeigt dies nachfolgendes Schema:

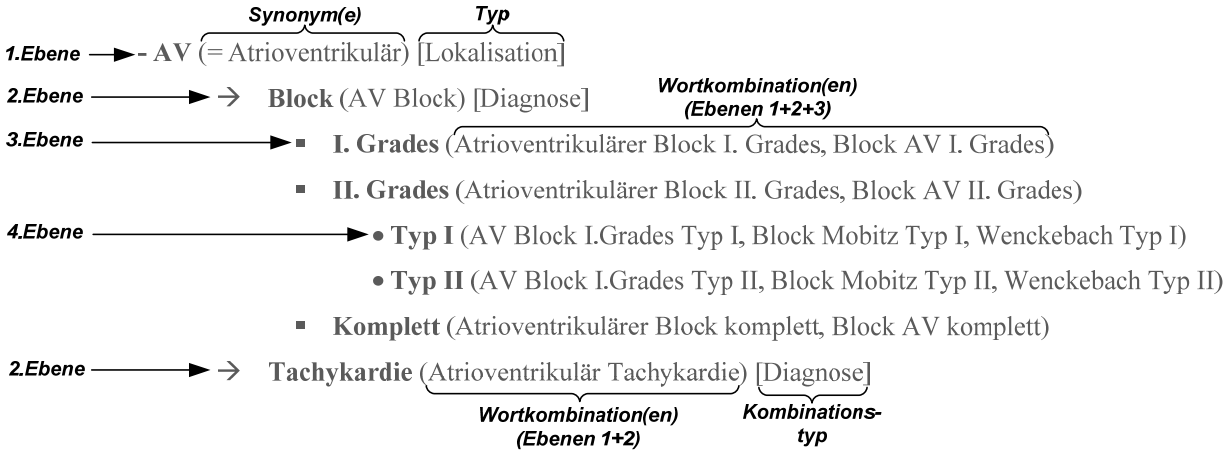


Abbildung 1: Beispiel Ebenenhierarchie

zur 1. Ebene:

- für die 1. Ebene müssen die Bezeichnung und eventuelle Synonyme und der Typ (Diagnose, Behandlung, Befund oder Lokalisation) erfasst werden

zur 2. Ebene:

- die 2. Ebene besteht nur aus Kombinationen von zwei Klassifikationen der 1. Ebene
- für sie muss der Typ festgelegt werden
- für Ebene 2 müssen (Wort-)Kombinationen definiert werden können

zur 3. und 4. Ebene:

- für diese beiden Ebenen müssen jeweils die (Wort-)Kombination und eventuelle Synonyme einzugeben sein
- Ebene 3 und 4 haben dabei immer den Typ der Ebene 2

Ferner muss in der Datenbank miterfasst werden, wer wann Neueinträge respektive Änderungen vorgenommen hat. Aus diesem Grund muss das Programm über eine eigenständige Benutzerverwaltung verfügen, die weder auf Vetera-eigene noch auf die Windows-typische Benutzerverwaltung („Active-Directory“) aufsetzt.

3. Ist-Analyse

3.1 Bisheriger Zustand

Die Tierärzte der Kleintierklinik hatten begonnen, die Begriffe der einzelnen Ebenen mit ihren jeweiligen Abhängigkeiten in einer Microsoft Access-Datenbank zu erfassen.

Die in dieser Datenbank vorgegebene Grundstruktur genügte den Ansprüchen des Auftraggebers nicht. Die Ebenenhierarchie reichte nur bis zur Ebene 3. Zudem waren viele Anforderungen in diesem Datenbankmodell noch nicht berücksichtigt beziehungsweise implementiert. Die hier mögliche Art der Erfassung bot zudem kaum Nutzerkomfort, da sie mit dem Access-Assistenten erstellt worden war. Ferner fehlte jegliche Benutzerverwaltung.

3.2 Arbeits- / Projektumgebung

Zur Durchführung und Realisierung des Projekts wurde ein Arbeitsplatzrechner mit

- Windows XP Professional SP2,
- einem breitbandigen Internetanschluss,
- Microsoft Office 2003,
- Java SDK/JRE 1.6.0_03[1],
- Eclipse V3.2.0[2] (mit Visual Editor V1.2.1[3] und Omondo V2.1.0[4]),
- Boomy Icon Set[5]
- SwingX[6]

bereitgestellt.

[1] http://java.sun.com/javase/downloads/index.jsp
[2] http://archive.eclipse.org/eclipse/downloads/drops/R-3.2-200606291905/index.php
[3] http://download.eclipse.org/tools/ve/downloads/drops/R-1.2.1-200609261748/index.html
[4] http://www.omondo.de/Registrierung_download_Free_3.2.asp
[5] http://www.iconarchive.com/category/application/boomy-icons-by-milosz-wlazlo.html
[6] http://swinglabs.org/projects.jsp

4. Soll-Konzept

4.1 Zielbestimmungen / Motivation

Aufgrund der Menge an zu digitalisierenden Daten (der Thesaurus umfasst weit über 300.000 Begriffe) ist eine komfortable Möglichkeit der Eingabe aus Gründen der Zeit- und Kostenersparnis unerlässlich.

Da die Eingabe ferner nur von teurem Fachpersonal (Veterinärmedizinern) vorgenommen werden kann, wiegen zusätzliche Kostengründe auf.

Weiterhin sollen zu einem späteren Zeitpunkt aufbauend auf dieses Programm Statistiken erstellt und Behandlungen ausgewertet werden. Daher muss gewährleistet sein, dass die Möglichkeiten zur Eingabe von Krankheitsbildern (Diagnosen, Behandlungen, Befunde und Lokalisationen) zentral vorgegeben sind und nur von dedizierten Personen angepasst werden können.

4.2 Anforderungen

Die Diagnosen, Behandlungen, Befunde bzw. Lokalisationen sollen in den einzelnen Ebenen über eine Eingabemaske eingegeben werden können. Des Weiteren soll es möglich sein vorhandene Ebenen auszulesen und zu bearbeiten.

Zu beachten ist auch, dass beim Neuerfassen einer Ebene 2 die Kombination der beiden Klassifikationen der Ebene 1 auch in umgekehrter Reihenfolge zwingend logisch zusammengehören. Somit kann durch ein automatisiertes Miterfassen dieser eine erhebliche Zeitersparnis erreicht werden; das Gleiche gilt in erweiterter Form ebenfalls für die Ebenen 3 und 4.
Werden zum Beispiel als Ebene 2 die beiden Ebene 1 Begriffe „Abdomen“ und „Abszess“ (Wortkombination „Abdominalabszess“) kombiniert, so werden in der Datenbank ebenfalls umgedreht „Abszess“ und „Abdomen“ mit der Wortkombination gespeichert.

Die eingegebenen Daten werden dann zusammen mit den Benutzerinformationen und des Eingabezeitpunkts in die Datenbank geschrieben, sodass sie im Vetera-System zur Verfügung stehen.

4.3 Einschränkungen

Vom Auftraggeber war vorgegeben, dass das Backend (die Datenbank) entweder ein Microsoft SQL-Server oder eine Microsoft Access-Datenbank sein soll.

4.4 Qualitätsanforderungen

Grundsätzlich gilt, dass alle Anforderungen vom Auftraggeber fachlich korrekt und vollständig erfüllt werden sollen.

Die Software muss darüber hinaus im Speziellen den Grundanforderungen an Benutzerfreundlichkeit und Robustheit erfüllen.

Die Benutzeroberfläche soll intuitiv und einfach aufgebaut und bedienbar sein. Fehleingaben sollen mit entsprechenden Hinweistexten, aus denen die Fehlerursache hervorgeht, angezeigt werden.

Außerdem soll durch eine ausführliche Dokumentation und Quellcodekommentierung die Wartbarkeit und Wiederverwendbarkeit des Programms erhöht werden. Hierzu zählt auch die Verwendung von Design-Patterns.

5. Projektplanung

5.1 Zeit- und Ablaufplan

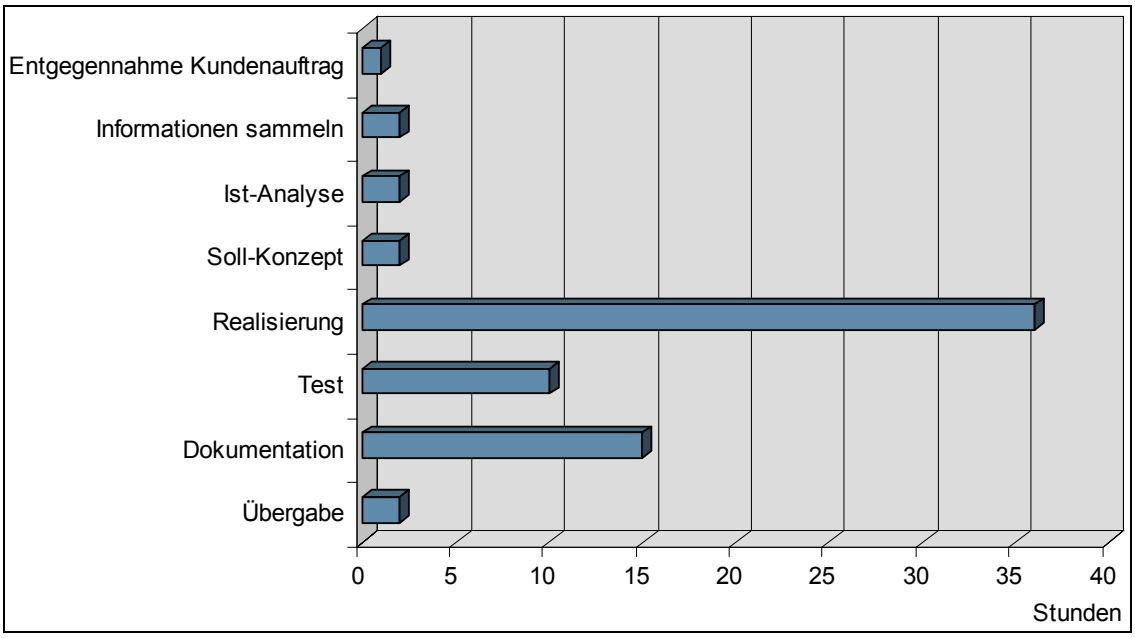


Abbildung 2: Zeitplanung

Entgegennahme des Auftrags vom Kunden:

1. Kundengespräch ..... 1 h

Informationen sammeln:

1. Kundengespräch ..... s.o.
2. Hintergrundinformationen sammeln/auswerten ..... 2 h

Ist-Analyse:

1. Analyse der derzeitigen Verfahrensweise ..... 2 h

Soll-Konzept:

1. Kundengespräch (Erarbeitung eines Pflichtenhefts) ..... 1,5 h
2. Planung und Entwurf eines Projektplans ..... 0,5 h

Realisierung:

1. Anpassung der bereits in Microsoft Access erfassten Daten / der Datenbank
   - Anpassung des DB-Modells an die Anforderungen ..... 2 h
   - Portierung der Daten auf das neue DB-Modell ..... 1 h
2. Entwicklung der Benutzeroberfläche (GUI-Frontend)
   - Entwicklung eines Konzepts (Aufbau/Struktur der GUI) ..... 1 h
   - Herstellung einer Datenbank-Verbindung ..... 2 h
   - Erstellen der Formulare ..... 4 h
   - Implementierung der Funktionen ..... 21 h
3. Implementierung der Benutzerverwaltung
   - Einbinden eines Logins ..... 3 h
   - Funktionen zum Hinzuzufügen / Entfernen von Nutzern ..... 2 h

Test:

1. Testen unter realen Bedingungen ..... 6 h
2. Gegebenenfalls Änderungen/Nachbesserungen vornehmen ..... 4 h

Dokumentation:

1. Kommentierung des Quelltextes / JavaDocs ..... 3 h
2. Erstellen der Projektdokumentation ..... 12 h

Übergabe des Projekts:

1. Präsentation des Projekts ..... 1 h
2. Einweisung des Kunden ..... 1 h

5.2 Kostenplanung

Die Kosten für das 70-stündige Projekt lassen sich aufgrund der internen Realisierung nur schwer einschätzen.

Aufgrund der Kürze des Projekts werden anfallende Gemeinkosten (Netzlast, Strom, Miete und ähnliches) nicht berücksichtigt.

Als Grundlage der fiktiven Kostenkalkulation wird daher der aktuell durchschnittliche Stundensatz für freiberufliche IT-Fachkräfte der Region Berlin herangezogen[7].

Stundenlohn:	65,00 €
Arbeitszeit:	70,00 h
Gesamtpreis (netto):	*4.550,00 €*
+ 19,00 % MwSt:	864,50 €
Gesamtpreis (brutto):	**5.414,50 €**

Tabelle 1: Kostenplanung

Das Angebot und die Rechnung zum Projekt sind im Anhang unter den Punkten 8.4.7 und 8.4.8 zu finden.

[7] http://www.gulp.de/kb/st/stdsaetze/sstext_LandRegion_f.html

6. Projektdurchführung

6.1 Durchführungszeitraum

Das Projekt wurde im Zeitraum vom 22.10.2007 bis 26.10.2007 und vom 05.11.2007 bis 09.11.2007 durchgeführt.
Die reine tägliche Arbeitszeit betrug durchschnittlich 7 Stunden, sodass sich eine Gesamtdauer von 70 Stunden ergibt.

Die Teilung des Projektzeitraums beruht auf der Tatsache, dass zwischen den Zeiten Berufsschulunterricht stattfand.

6.2 Entgegennahme des Kundenauftrags

Am 22.10.2007 wurde mit dem Kunden Herrn Dr. Sommerer in Zusammenarbeit des Veterinärmediziners Nico Wohllebe der Projektauftrag besprochen und entgegengenommen.

Die Gesprächsdetails können dem Anhang 8.4.1 entnommen werden.

6.3 Durchführung der Ist-Analyse

Zur Durchführung der Ist-Analyse wurde im Wesentlichen die Besprechung des Projektauftrages genutzt, welche durch diverse Mails und Gespräche mit dem Auftraggeber ergänzt wurde (siehe Anhang 8.4.1).

6.4 Erstellung des Soll-Konzepts

Auf der Grundlage der Ist-Analyse wurde das Soll-Konzept erstellt, in dessen Ergebnis ein schriftlicher Zeit- und Ablaufplan festgehalten wurde.

Zusammen mit dem Auftraggeber wurde darüber hinaus ein verbindliches Pflichtenheft erstellt (siehe Anhang 8.4.2).

6.5 Realisierung

6.5.1 Anpassung der bereits in MS Access erfassten Daten / der Datenbank

Anpassung des Datenbank-Modells an die Anforderungen:

Die bereits vorhandene Datenbank musste so angepasst werden, dass sie den Anforderungen an das Projekt genügt.

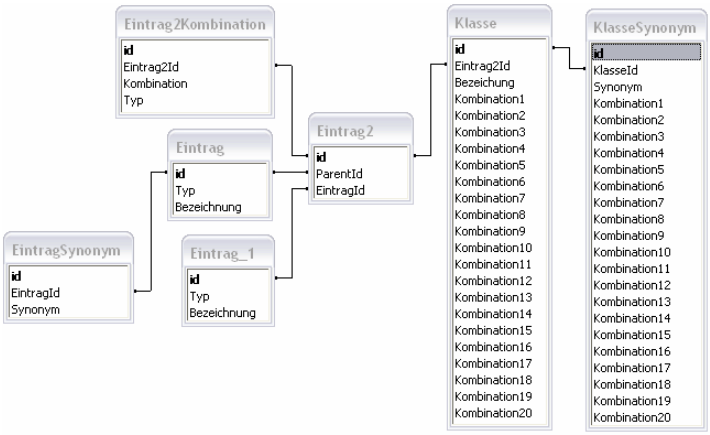


Abbildung 3: altes DB-Modell

In der vorhandenen Datenbank ist die Ebenehierarchie lediglich bis zur 3. Ebene realisiert, wobei noch keine Benutzerinformationen hinterlegt werden konnten.

Ferner lagen in diesem Datenmodell logische Wiederholungen vor, die es aufzulösen galt.

Das neu entwickelte Datenbankmodell entspricht nun allen Anforderungen an das Projekt.

Jede Ebene steht nun in richtiger Beziehung zur vorhergehenden und hat seine jeweilige Relation für Synonyme und Kombinationswörter.

Zum Speichern von Benutzerinformationen wurden je Relation die vier Attribute „ErstelltAm“, „ErstelltVon“, „GeaendertAm“ und „GeaendertVon“ in das Datenbankmodell aufgenommen. Ferner existiert nun die Tabelle „T_User“ zur Implementation einer Benutzerverwaltung.

Die Ebene 2 ist ein Zusammenschluss aus zwei Begriffen der ersten Ebene, daher liegt hier eine rekursive Beziehung vor.

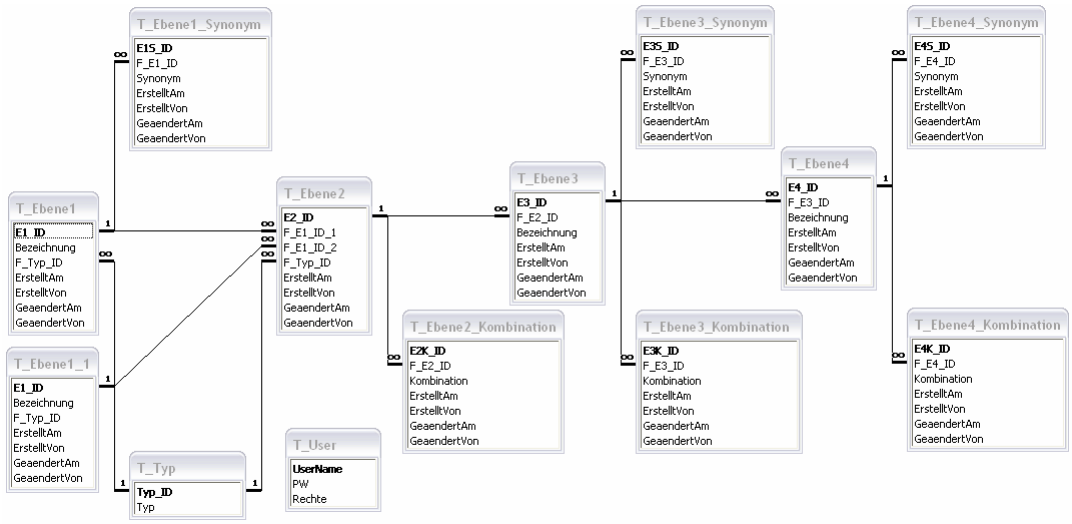


Abbildung 4: neues DB-Modell

Portierung der Daten auf das neue DB-Modell:

Da in der alten Datenbank bereits einige Daten erfasst wurden, mussten diese auf das neue Modell portiert werden.

Folgende Entitäten und Attribute wurden angepasst:

- das Attribut „Typ“ der Relation „Eintrag2“ musste durch die ID des Typs ersetzt werden
- das Attribut „Typ“ der Relation „Eintrag2Kombination“ musste durch die ID des Typs ersetzt werden
- das Attribut „Typ“ der Relation „Eintrag2Kombination“ musste auf die Relation „Eintrag2“ verschoben werden

Nachdem diese Schritte unternommen wurden, konnten die Daten eins zu eins in das neue Datenbank-Modell übertragen werden.

6.5.2 Entwicklung der Benutzeroberfläche (GUI-Frontend)

Der Aufbau der Klassen kann den JavaDocs im Anhang 8.4.10 entnommen werden.

Entwicklung eines Konzepts (Aufbau / Struktur der GUI):

Die GUI ist so konzeptioniert, dass sie einfach und intuitiv zu bedienen ist. In der linken Hälfte ist stets ein Menü mit ausklappbaren Einträgen für jede Ebene implementiert, um direkten Zugriff auf alle Funktionen des Programms anzubieten. Ferner stehen Nutzern mit administrativen Rechten zusätzlich Menüeinträge zur Nutzerverwaltung zur Verfügung.

Die Ebenen sind jeweils unterteilt in „Erfassen“ und „Bearbeiten“, wobei der Aufbau der einzelnen Schritte der Ebenen jeweils gleich gehalten ist und im rechten Teil dargestellt wird.

Fehler- und Statusmeldungen werden im unteren Teil des Fensters in einem entsprechenden Feld ausgegeben.

Die GUI-Übersicht kann dem Anhang 8.4.3 entnommen werden.

Herstellung einer Datenbank-Verbindung:

Aufgrund der unproblematischen Portierbarkeit auf einem Microsoft SQL-Server und den guten grafischen „What you see is what you get“-Möglichkeiten, wurde sich zu Gunsten der Microsoft Access-Datenbank entschieden.
Sollte es später notwendig sein, eine andere Datenbank zu benutzten, so muss lediglich die Klasse „*AccessDBConnection*“ entsprechend angepasst beziehungsweise ausgetauscht werden.

Die Klasse „*AccessDBConnection*“ ist als „Factory Method“-Pattern programmiert und öffnet bei Aufruf der entsprechenden Methode eine Verbindung zur Datenbank bzw. schließt die Verbindung wieder.

Erstellen der Formulare:

Die Formulare wurden mit dem Visual Editor-Plugin V1.2.1 für Eclipse erstellt. Die verschiedenen Icons für das Projekt wurden dem Boomy-Icon-Set[8] entnommen.

Folgende Steuerelemente entstammen dem SwingX-Projekt[9]:

- JXTitledPanel
- JXTaskPaneContainer
- JXTaskPane
- JXTable
- JXList

Alle weiteren Steuerelemente entstammen der proprietären Grafikbibliothek „Swing“.

MainFrame

Die Klasse „*MainFrame*“ ist das Hauptfenster des Programms. Auf der linken Seite werden in ausklappbaren „Task-Panes“ alle Menüpunkte angezeigt und die rechte Seite enthält den Aufgabenbereich. Dieser Aufgabenbereicht ist der so genannte „*workspace*“, auf das sämtliche Panels zum Bearbeiten und Erfassen der jeweiligen Ebenen bzw. zur Nutzerverwaltung gelegt werden.
Im unteren Teil des Fensters befindet sich das Statuslabel zur Ausgabe von Fehler- oder Statusmeldungen, die im Programmablauf auftreten.

[8] http://www.iconarchive.com/category/application/boomy-icons-by-milosz-wlazlo.html
[9] http://swinglabs.org/projects.jsp

MyTaskPane

Da die ausklappbaren Task-Panes der einzelnen Ebenen jeweils die gleichen Labels besitzen („Erfassen“ und „Bearbeiten“), wurde die Klasse „*MyTaskPane*“ erstellt.
Sie instanziiert ein „JXTaskPane“, legt darauf die beiden Labels und gibt die Referenz auf dieses Task-Pane zurück.
Es handelt sich hierbei um das Entwurfsmuster „Builder“.

UsermanageTaskPane

Die Klasse „*UsermanageTaskPane*“ ist äquivalent zur Klassen „*MyTaskPane*“ zu verstehen, mit dem Unterschied, dass sie das Task-Pane für die Nutzerverwaltung erstellt.

Implementierung der Funktionen:

MouseEventErfassen1

Die Klasse „*MouseEventErfassen1*“ ist der MouseListener für den Menüeintrag „Erfassen“ der 1. Ebene. Hier wird definiert, wie auf bestimmte Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll. Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Erfassen eines Begriffes der 1. Ebene dar. Es werden die Bezeichnung, Synonyme und der Typ erfasst.

MouseEventBearbeiten1

Die Klasse „*MouseEventBearbeiten1*“ ist der MouseListener für den Menüeintrag „Bearbeiten“ der 1. Ebene. Hier wird definiert, wie auf bestimmte Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll. Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Bearbeiten eines Begriffes der 1. Ebene dar. In einem 1. Schritt kann man den zu bearbeitenden Begriff auswählen und im 2. Schritt Änderungen vornehmen.

MouseEventErfassen2

Die Klasse „*MouseEventErfassen2*“ ist der MouseListener für den Menüeintrag „Erfassen“ der 2. Ebene. Hier wird definiert, wie auf bestimmte Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll. Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Erfassen einer 2. Ebene dar. Im 1. Schritt wählt man dazu die beiden Begriffe der 1. Ebene aus, die kombiniert werden sollen und im 2. Schritt werden Typ und Wortkombinationen definiert. Beim Speichern der Kombination werden die beiden Ebene 1-Begriffe nicht nur in der gewählten Reihenfolge, sondern auch in umgedrehter Abfolge gespeichert.

MouseEventBearbeiten2

Die Klasse „*MouseEventBearbeiten2*“ ist der MouseListener für den Menüeintrag „Bearbeiten“ der 2. Ebene. Hier wird definiert, wie auf bestimmte

Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll. Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Bearbeiten einer 2. Ebene dar. Dazu wird im 1. Schritt ausgewählt, welche Kombination man bearbeiten möchte, die man im 2. Schritt verändern kann. Die explizit dazugehörige Kombination (umgedrehte Reihenfolge der Ebene 1-Begriffe) wird ebenfalls mit verändert.

MouseEventErfassen3

Die Klasse „*MouseEventErfassen3*“ ist der MouseListener für den Menüeintrag „Erfassen“ der 3. Ebene. Hier wird definiert, wie auf bestimmte Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll. Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Erfassen einer 3. Ebene dar. Im 1. Schritt wählt man dazu die 2. Ebene aus, welche eine 3. Ebene erhalten soll. Im 2. Schritt werden Bezeichnung, Wortkombinationen und Synonyme definiert. Beim Speichern der Ebene bekommt die explizit dazugehörige Ebene 2 (umgedrehte Reihenfolge der Ebene 1-Begriffe) ebenfalls diese 3. Ebene zugeordnet.

MouseEventBearbeiten3

Die Klasse „*MouseEventBearbeiten3*“ ist der MouseListener für den Menüeintrag „Bearbeiten“ der 3. Ebene. Hier wird definiert, wie auf bestimmte Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll. Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Bearbeiten einer 3. Ebene dar. Dazu wird im 1. Schritt eine 3. Ebene ausgewählt, die man im 2. Schritt verändern kann. Die explizit dazugehörige Ebene 3 (umgedrehte Reihenfolge der Ebene 1-Begriffe) wird ebenfalls mit verändert.

MouseEventErfassen4

Die Klasse „*MouseEventErfassen4*“ ist der MouseListener für den Menüeintrag „Erfassen“ der 4. Ebene. Hier wird definiert, wie auf bestimmte Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll. Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Erfassen einer 4. Ebene dar. Im 1. Schritt wählt man dazu die 3. Ebene aus, welche eine 4. Ebene erhalten soll. Im 2. Schritt werden Bezeichnung, Wortkombinationen und Synonyme definiert. Beim Speichern der Ebene bekommt die explizit dazugehörige Ebene 3 (umgedrehte Reihenfolge der Ebene 1-Begriffe) ebenfalls diese 4. Ebene zugeordnet.

MouseEventBearbeiten4

Die Klasse „*MouseEventBearbeiten4*“ ist der MouseListener für den Menüeintrag „Bearbeiten“ der 4. Ebene. Hier wird definiert, wie auf bestimmte Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll.

Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Bearbeiten einer 4. Ebene dar. Dazu wird im 1. Schritt eine 4. Ebene ausgewählt, die man im 2. Schritt verändern kann. Die explizit dazugehörige Ebene 4 (umgedrehte Reihenfolge der Ebene 1-Begriffe) wird ebenfalls mit verändert.

6.5.3 Implementierung der Benutzerverwaltung

Einbinden eines Logins:

Der Login wurde mit der Klasse „*Login*“ realisiert. Diese Klasse stellt zum einen den Logindialog und die notwendige Loginlogik dar und ist zum anderen der Einstiegpunkt des Programms („main“).

In Erweiterung zum Projektantrag hat sich der Kunde für einen, vom Windowsanmeldenamen abhängigen Login gewünscht. Daher ist das Feld „Benutzername“ jeweils mit dem aktuell am System angemeldeten Benutzer ausgefüllt und deaktiviert.

Die Klasse realisiert darüber hinaus die Erstvergabe von Passwörtern. Sollte ein Benutzer neu angelegt worden sein und noch kein Passwort besitzen, so erweitert sich die Eingabe und er wird gezwungen ein Passwort zu vergeben.

Nach erfolgreichem Login öffnet sich das Hauptfenster (Klasse „*Mainframe*“).

Funktionen zum Hinzuzufügen / Entfernen von Nutzern:

Die folgenden Funktionen sind nur verfügbar, wenn der angemeldete Benutzer über administrative Rechte verfügt.

MouseEventUsernew

Die Klasse „*MouseEventUsernew*“ ist der MouseListener für den Menüeintrag „Nutzer Anlegen“. Hier wird definiert, wie auf bestimmte Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll. Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Erfassen von neuen Nutzern bereit.

MouseEventUserdel

Die Klasse „*MouseEventUserdel*“ ist der MouseListener für den Menüeintrag „Nutzer/PW löschen“. Hier wird festgelegt, wie auf bestimmte Mouse-Ereignisse (Klick, MouseOver, etc.) auf das Label reagiert werden soll. Ferner baut diese Klasse das Panel auf, welches bei Klick im „workspace“ geladen wird und realisiert den Effekt, dass der Eintrag beim Überfahren mit der Mouse unterstrichen dargestellt wird.

Diese Klasse stellt die Funktionalitäten zum Löschen von Nutzern und zum Zurücksetzten von Passwörtern bereit.

6.6 Test

Zum Testen der korrekten Funktionsweise wurde ein Black-Box-Test in Zusammenarbeit mit dem Kunden durchgeführt.

Im Testlauf wurde ein Aktualisierungsfehler beim Bearbeiten der Ebenen gefunden. Beim Ändern eines Begriffes der Ebene, wurde die Übersichtstabelle nicht korrekt aktualisiert. Die Behebung dieses Fehlers nahm circa 3 Stunden in Anspruch.

Die Testdokumentation befindet sich in der Anlage 8.4.5.

6.7 Übergabe

Das Projekt wurde am 09.11.2007 dem Kunden, Herrn Dr. Sommerer, übergeben. Bei der Übergabe war darüber hinaus Herr Nico Wohllebe, einer der künftigen Nutzer des Programms, anwesenden.

Das Übergabeprotokoll befindet sich in der Anlage 8.4.9.

7. Projektabschluss

7.1 Soll-Ist-Vergleich

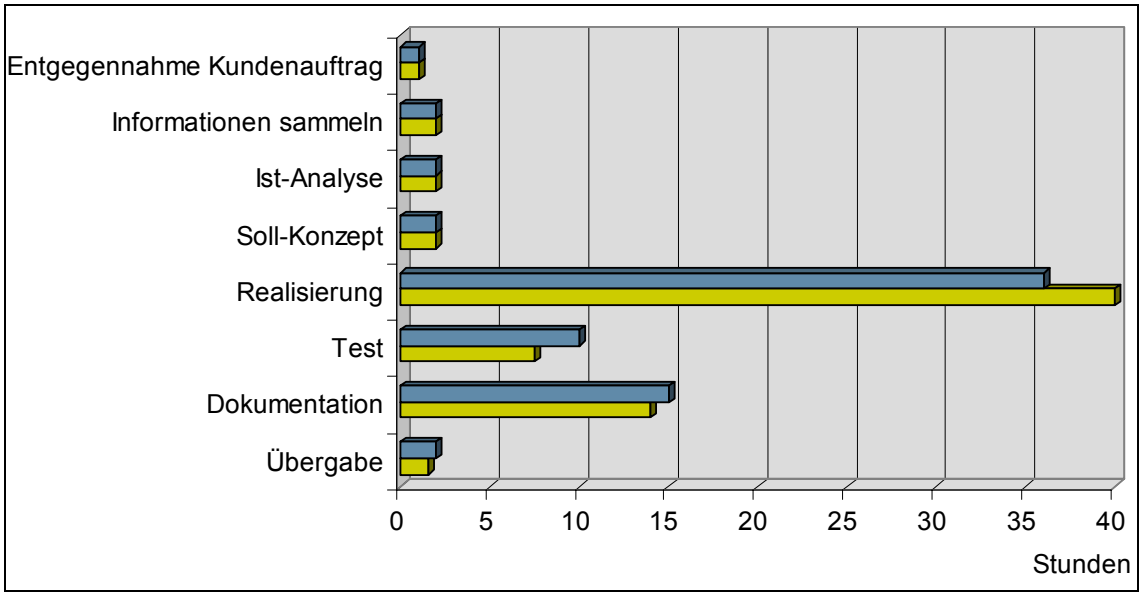


Abbildung 5: Soll-Ist-Vergleich

In folgenden Projektphasen traten gegenüber der Planung Zeitabweichungen auf:

Projektphase	Soll	Ist	Differenz
Realisierung	36 h	40 h	+ 4 h
Test	10 h	7,5 h	- 2,5 h
Dokumentation	15 h	14 h	- 1 h
Übergabe	2 h	1,5 h	- 0,5 h
			= +/- 0 h

Tabelle 2: Soll-Ist-Vergleich

Realisierung:

Aufgrund eines erst gegen Ende entdeckten Fehlers bei der Zuordnung der ID's in den sortierbaren Tabellen mussten alle Klassen zu den Ebenen eins bis vier (Bearbeiten und Erfassen) noch einmal überarbeitet und angepasst werden. Diese Verzögerung

nahm circa 4 Stunden zusätzlich in Anspruch. In dieser Zeit wurden ferner einige Leistungs- und Geschwindigkeitsanpassungen vorgenommen.

Test:

Aufgrund des Zeitüberzuges in der Realisierungsphase, wurde in der Testphase in Absprache mit dem Auftraggeber der White-Box-Test ausgelassen. Zu diesem Vorgehen wurde sich entschieden, da der durchgeführte Black-Box-Test bereits die korrekte Funktionalität der Software sicherstellt.
Durch diese Maßnahme konnten zweieinhalb Stunden eingespart werden.

Dokumentation:

Die Dokumentierung hat, aufgrund der JavaDoc - Funktionalität, eine Stunde weniger als erwartet gedauert.

Übergabe:

Aufgrund des bereits vorhandenen technischen Verständnisses beim Kunden und der intuitiven Bedienbarkeit des Programms, tauchten während der Übergabe weniger Fragen als erwartet auf, sodass die Übergabe kürzer ausfiel als geplant.

7.2 Nutzwertanalyse

Als Grundlage der Nutzwertanalyse wurden die Aussagen des Tierarztes Nico Wohllebe, der bereits mit der alten Datenbank gearbeitet hatte, herangezogen.

Die Analyse ergab eine geschätzte Gesamtersparnis zur Erfassung aller Thesaurus-Begriffe von über 62.000,00 Euro gegenüber der alten Lösung.

Die detaillierte Analyse kann dem Anhang 8.4.6 entnommen werden.

7.3 Fazit & Ausblick

Das Projekt wurde erfolgreich und vollständig im geplanten Zeitrahmen erfüllt. Die einzelnen veterinärmedizinischen „Thesaurus“-Begriffe können nun in effizienter, einfacher und somit wirtschaftlicher Weise erfasst werden.
Hierbei kann mit Kosteneinsparungen gegenüber der früheren Lösung um mehr als die Hälfte gerechnet werden.

Da die Gestaltung der Benutzeroberfläche im Corporate Design der Freien Universität Berlin während des Projektablaufs aufgrund fehlender zeitlicher Ressourcen nicht mehr durchgeführt werden konnte (Kannkriterium im Pflichtenheft), wurden entsprechende Anpassungen erst im Anschluss durchgeführt.

Das Produkt befindet sich inzwischen im produktiven Einsatz an der Kleintierklinik des Fachbereichs Veterinärmedizin der Freien Universität Berlin.

Darüber hinaus wünscht der Auftraggeber nun die Erweiterung des Programms um die Möglichkeit der Erfassung so genannter „Schubladen“, um einzelnen Begriffen bestimmten Obergruppen (z.B. „neurologische Erkrankung“) zuordnen zu können (Kannkriterium im Pflichtenheft).

Als weiteren Ausblick des Projekts könnte sich der Auftraggeber die Möglichkeit der Generierung einer alphabetischen Liste vorstellen, die alle bisher erfassten Begriffe umfasst und zum Beispiel in Form einer pdf-Datei speichert.

8. Anhang

8.1 Glossar

Active Directory	Das Active Directory (kurz AD) ist der Verzeichnisdienst von Microsoft Windows Server. Es ist eine Art Datenbank, die den verschiedenen Netzwerkobjekten (Nutzer, Computer etc.) Eigenschaften zuordnet und verwaltet.
Black-Box-Test	Der Black-Box-Test ist eine Softwaretestmethode, bei der der Tester keine Kenntnis über die innere Funktionsweise des Systems hat. Er beschränkt sich lediglich auf das Außenverhalten des Programms, nicht die Implementierung.
Builder	Der Builder (dt. „Erbauer“) ist ein Design Pattern (dt. „Entwurfsmuster) in der Softwareentwicklung. Mit ihm wird die Erzeugung komplexer Objekte vereinfacht, indem der Konstruktionsprozess in eine spezielle Klasse ausgelagert wird.
Design Pattern	Design Patterns (dt. „Entwurfsmuster”) ist allgemein eine Lösungsstrategie/-vorlage für Entwurfsprobleme der Softwareentwicklung. Ziel von Design Patterns ist die Wiederverwendung für ähnliche Problemstellungen bei maximaler Flexibilität.
Eclipse	Eclipse ist eine quelloffene Entwicklungsumgebung (IDE) zur Entwicklung von Software in zum Beispiel der Programmiersprache „Java“.
Factory Method	Die Factory Method (dt. „Fabrik-Methode“) ist ein Design Pattern (dt. „Entwurfsmuster) in der Softwareentwicklung. Mit ihr wird eine Schnittstelle zur Erzeugung von Objekten definiert, die anhand Kriterien entscheiden kann, welches konkrete Objekt zurückgegeben werden soll.
GUI	GUI ist die Abkürzung für Graphical User Interface; zu deutsch „grafische Benutzerschnittstelle“ oder „grafische Benutzeroberfläche“
JavaDoc	JavaDoc ist ein Werkzeug zur Dokumentation von Java-Quelltext. Mittels JavaDoc ist es möglich, aus dem geschriebenen und kommentierten Quelltext HTML-Dokumentationsdateien zu erstellen.
JDK/JRE	Java Development Kit bzw. Java Runtime Environment sind Sammlungen von Java-APIs (Programmierschnittstellen), die benötigt werden, um Software in der Programmiersprache „Java“ zu erstellen, kompilieren und auszuführen.
Label	In der GUI-Programmierung bezeichnet ein Label ein Beschriftungsfeld zur Ausgabe von einzeiligen Texten.

MouseListener	Ein MouseListener ist ein „Event Handler“ (Ereignis-Behandler) der auf Aktionen der Mouse wartet und entsprechend darauf reagiert. Aktionen können z.B. sein ein Linksklick, ein Rechtsklick oder das Herüberfahren mit dem Cursor.
Omondo	Omondo ist eine freie Erweiterung (Plugin) für Eclipse zur Erstellung und Modellierung von UML-Diagrammen.
Panel	In der GUI-Programmierung bezeichnet ein Panel einen Behälter für andere Steuerelemente.
Rekursion	Eine Rekursion (vom lat. „zurücklaufen“) ist der Aufruf bzw. Bezug auf sich selbst.
SwingX	SwingLabs SwingX ist eine freie Erweiterung der proprietären Java-Grafikbibliothek „Swing“. Sie stellt z.B. ausklappbare Panels oder sortierbare Tabellen bereit.
Thesaurus	Ein Thesaurus ist eine Wörtersammlung, dessen Begriffe durch Relationen miteinander verbunden sind. Es besteht aus einer systematisch geordneten Sammlung von Begriffen, die einen thematischen Bezug zueinander haben.
UML	Die Unified Modeling Language (dt. „Vereinheitlichte Modellierungs-Sprache“) ist eine genormte Sprache für die (grafische) Modellierung von Software und anderen Systemen. Sie umfasst dazu verschiedene Diagrammarten (z.B. das Klassendiagramm).
Vetera	„Vetera-Campus“ ist ein zurzeit am Fachbereich Veterinärmedizin der Freien Universität Berlin laufendes Projekt, das es ermöglichen soll, tierische Patienten in Kliniken zu verwalten.
Visual Editor	Der Visual Editor ist eine freie Erweiterung (Plugin) für Eclipse zur grafischen Erstellung einer GUI nach WYSIWYG-Prinzip.
What you see is what you get	„What you see is what you get“ (kurz WYSIWYG) bedeutet, dass die Bildschirmanzeige während der Bearbeitung genau dem entspricht, was anschließend bei der Ausgabe herauskommt. Editoren in denen man sich z.B. eine Oberfläche mit der Mouse „zusammenklicken“ kann arbeiten nach diesem Prinzip.
White-Box-Test	Der White-Box-Test ist eine Softwaretestmethode, bei der der Tester Kenntnis über die innere Funktionsweise des Systems hat. Ein Blick in den Quelltext ist also, im Gegensatz zum Black-Box-Test, gestattet (es wird am Code geprüft).

8.2 Tabellen- & Abbildungsverzeichnis

8.3 Quellen

- http://de.wikipedia.org
- http://java.sun.com/javase/6/docs/api/
- http://archive.eclipse.org/eclipse/downloads/drops/R-3.2-200606291905/index.php
- http://download.eclipse.org/tools/ve/downloads/drops/R-1.2.1-200609261748/index.html
- http://java.sun.com/javase/downloads/index.jsp
- http://swinglabs.org/projects.jsp
- http://www.gulp.de/kb/st/stdsaetze/sstext_LandRegion_f.html
- http://www.iconarchive.com/category/application/boomy-icons-by-milosz-wlazlo.html
- http://www.omondo.de/Registrierung_download_Free_3.2.asp
- Karl Eilbrecht, Gernot Starke: Patterns kompakt – Entwurfsmuster für effektive Software-Entwicklung; Elsevier Spektrum Akademischer Verlag München; 2. Auflage 2007
- Hübscher, Petersen, Rathgeber, Richter, Scharf: IT-Handbuch IT-Systemelektroniker/-in Fachinformatiker/-in; westermann Braunschweig; 5. Auflage 2007

8.4 Anlagen

8.4.1 Gesprächsnotizen / E-Mail-Verkehr

Gesprächsnotiz	
Datum:	28.08.2007
Zeit:	16:30 Uhr - 17:20 Uhr
Ort:	Büro des Leiters der IT-Abteilung, Herr Dr. Sommerer
Teilnehmer:	- Herr Dr. Sommerer (Leiter der IT-Abteilung des Fachbereichs Veterinärmedizin der FU Berlin) - Herr Wohllebe (Veterinärmediziner und künftiger Nutzer des Programms) - Herr Mäurer (Projektleiter)
Betreff des Gespräches:	
Vorgespräch mit dem Kunden für den Projektantrag	
Inhalte des Gespräches:	
- kurzer Abriss/Einleitung in Vetera-Campus (was ist es / was kann es) - allgemeiner Aufbau des Thesaurus (Ebenen, Synonyme, Typen, Kombinationen) - Verwendung des Thesaurus in Vetera-Campus (Klärung der Bedeutung des Projekts) - Vorstellung der Microsoft Access-Datenbank (aufzeigen des Ist-Zustands) - Übergabe der Microsoft Access-Datenbank mit einigen Daten - Einigung über den Namen des Projekts („Behandlungs-, Befund- und Diagnoseschlüsselverwaltung“) - Übergabe eines kompletten Beispieles mit den Ebenen 1 bis 4	
Berlin, den 28.08.2007	

E-Mail	
Datum:	16.10.2007
Zeit:	12:00
Absender:	René Mäurer (Projektleiter)
Empfänger:	Dr. Manfred Sommerer (Leiter der IT-Abteilung des Fachbereichs Veterinärmedizin der FU Berlin)
Betreff der E-Mail:	
Terminanfrage zur Entgegennahme des Auftrages	

Inhalt der E-Mail:
Hallo Herr Dr. Sommerer, wäre Ihnen der 22.10.2007 gegen 10:00 Uhr als Termin recht, an dem wir uns genau mit Ihren Vorstellungen und Wünschen des Endprodukts auseinandersetzen und ein Pflichtenheft erarbeiten? Anbei erhalten Sie noch mein Angebot über die gewünschte Dienstleistung. Ich freue mich auf eine gute Zusammenarbeit. Freundliche Grüße René Mäurer

Gesprächsnotiz	
Datum:	22.10.2007
Zeit:	10:00 Uhr – 12:30 Uhr
Ort:	Büro des Leiters der IT-Abteilung, Herr Dr. Sommerer
Teilnehmer:	- Herr Dr. Sommerer (Leiter der IT-Abteilung des Fachbereichs Veterinärmedizin der FU Berlin) - Herr Wohllebe (Veterinärmediziner und künftiger Nutzer des Programms) - Herr Mäurer (Projektleiter)
Betreff des Gespräches:	
Entgegennahme des Projektauftrags; Pflichtenheft	
Inhalte des Gespräches:	
- Einteilung in bis zu 4 Ebenen (mit je Synonyme und Kombinationen) → Datenbank muss angepasst werden. - Ebene 2 besitzt keine Synonyme, sonder lediglich Kombinationen aus den Wörtern der Ebene 1 - beim Neuerfassen der 2. Ebene sollen Kombinationswörter aus den beiden Ebenen 1 automatisch angezeigt werden - Automatisierung der Ebene 1-2-Zuordnung und deren Kombinationen - Ebene 3 und 4 haben immer den Typ der Ebene 2 (müssen also nicht extra erfasst werden) - Ebene 3 und 4 haben beide eigene Synonyme und Kombinationen - das Backend soll entweder eine Microsoft Access oder Microsoft SQL-Server sein (Klärung mit Hr. Stiel, GP-Software) - es wurde ein kurzer Ansatz konzeptioniert, wie die GUI aussehen könnte - Einigung über das Pflichtenheft (Muss-, Soll-, Kannkriterien)	
Berlin, den 22.10.2007	

8.4.2 Pflichtenheft

1. Zielbestimmung

Auftrag ist es, ein benutzerfreundliches Programm zur Eingabe und Verwaltung von Behandlungs-, Befund- und Diagnosebegriffen zu erstellen.

1.1 Musskriterien

- Erfassung der vier Ebenen via grafischer Benutzeroberfläche
- Bearbeitung der vier Ebenen via grafischer Benutzeroberfläche
- Implementierung einer Benutzerverwaltung
- bereits erfasste Begriffe müssen sortierbar aufgelistet werden
- beim Erfassen/Ändern der Ebenen sollen die jeweils in umgedrehter Reihenfolge logisch dazugehörigen Ebenen automatisch mit erfasst/geändert werden (umgedrehte Reihenfolge der Ebene 1-Begriffe)

1.2 Sollkriterien

- ausreichende Kommentierung des Quellcodes
- gute Wiederverwendenbarkeit / Erweiterbarkeit der Klassen

1.3 Kannkriterien

- die Programmoberfläche im Corporate Design der Freien Universität Berlin gestallten
- Möglichkeit der Zuweisung der Ebenen-Begriffe zu bestimmten medizinischen „Oberbegriffen“ (z.B. „neurologische Erkrankung“)

2. Produkteinsatz

2.1 Anwendungsbereiche

Die Software wird in der Kleintierklinik des Fachbereichs Veterinärmedizin der Freien Universität Berlin eingesetzt.

2.2 Zielgruppen

Zielgruppe ist jeder praktisch arbeitende Tierarzt des Fachbereichs Veterinärmedizin der Freien Universität Berlin.

3. Produktübersicht

Das Produkt soll es den Veterinärmedizinern der Kleintierklinik erleichtern, einen „Thesaurus“ von miteinander in Relation stehenden, veterinärmedizinischen Begriffen zu erstellen.

Diese Begriffe können in bis zu vier Ebenen unterteilt sein und haben (je nach Ebene) einen Typ, Synonyme und/oder Kombinationsmöglichkeiten.

4. Produktfunktionen

Folgende Funktionen müssen zur Verfügung stehen:

- für die 1. Ebene müssen die Bezeichnung, und eventuelle Synonyme und der Typ (Diagnose, Behandlung, Befund oder Lokalisation) erfasst werden
- die 2. Ebene besteht nur aus Kombinationen von zwei Klassifikationen der 1. Ebene
- für die 2. Ebene muss der Typ erfasst werden
- für Ebene 2 müssen (Wort-)Kombinationen erfasst werden können

- für die Ebenen 3 und 4 müssen jeweils die (Wort-)Kombination und eventuelle Synonyme erfassbar sein
- Ebene 3 und 4 haben dabei immer den Typ der Ebene 2
- alle Ebenen müssen neu erfasst werden können
- bereits gespeicherte Begrifflichkeiten müssen bearbeitet werden können
- die Darstellung der Begriffe erfolgt jeweils in entsprechenden sortierbaren Tabellen
- ein Login sorgt für die Identifizierung und Authentifizierung der Nutzer

5. Produktdaten (Datenhaltung)

Die Datenhaltung erfolgt in einer relationalen Datenbank. Neben den eigentlichen Daten werden auch Informationen darüber abgelegt, wer und wann den Datensatz verändert beziehungsweise angelegt hat.

6. Qualitätsanforderungen

- Benutzerfreundlichkeit (intuitive Bedienbarkeit und selbsterklärende Aufbau der grafischen Oberfläche)
- Robustheit (Fehleingaben mit entsprechenden Hinweistexten, aus denen die Fehlerursache hervorgeht, anzeigen)
- Wartbarkeit und Wiederverwendbarkeit (der Quellcode sollte ausreichend kommentiert werden; Verwendung von Design Patterns)

7. Benutzungsoberfläche

- die Benutzeroberfläche soll intuitiv und benutzerfreundlich bedienbar sein
- die Oberfläche wird durch einen Login geschützt, sodass nur berechtigte Personen Zugriff erhalten

8. Nichtfunktionale Anforderungen

Durch den Einsatz der Java-Technologie ist eine Plattformunabhängigkeit von Grund auf gegeben. Einzige Vorraussetzung ist eine zum entsprechendem Betriebssystem passende Java-Laufzeitumgebung (Java Runtime Environment) ab Version 1.5.

9. Technische Produktumgebung

9.1 Software

- Windows XP Professional SP2
- Microsoft Office 2003
- Java SDK/JRE 1.6.0_03
- Eclipse V3.2.0 (mit Visual Editor V1.2.1 und Omondo V2.1.0)

9.2 Sonstige

- breitbandiger Internetanschluss
- Boomy Icon Set
- SwingX (Klassen zur Erweiterung der proprietären Grafikbibliothek „Swing“)

9.3 Produktschnittstellen

Extern fungiert das Programm als Schnittstelle zwischen dem Anwender und der Software Vetera-Campus der Firma GP-Software.

Intern wird eine Schnittstelle zwischen der Benutzeroberfläche und der Verarbeitungslogik geschaffen, als auch eine Schnittstelle für die Datenhaltung (Zugriff auf die Datenbank).

10. Unterschriften

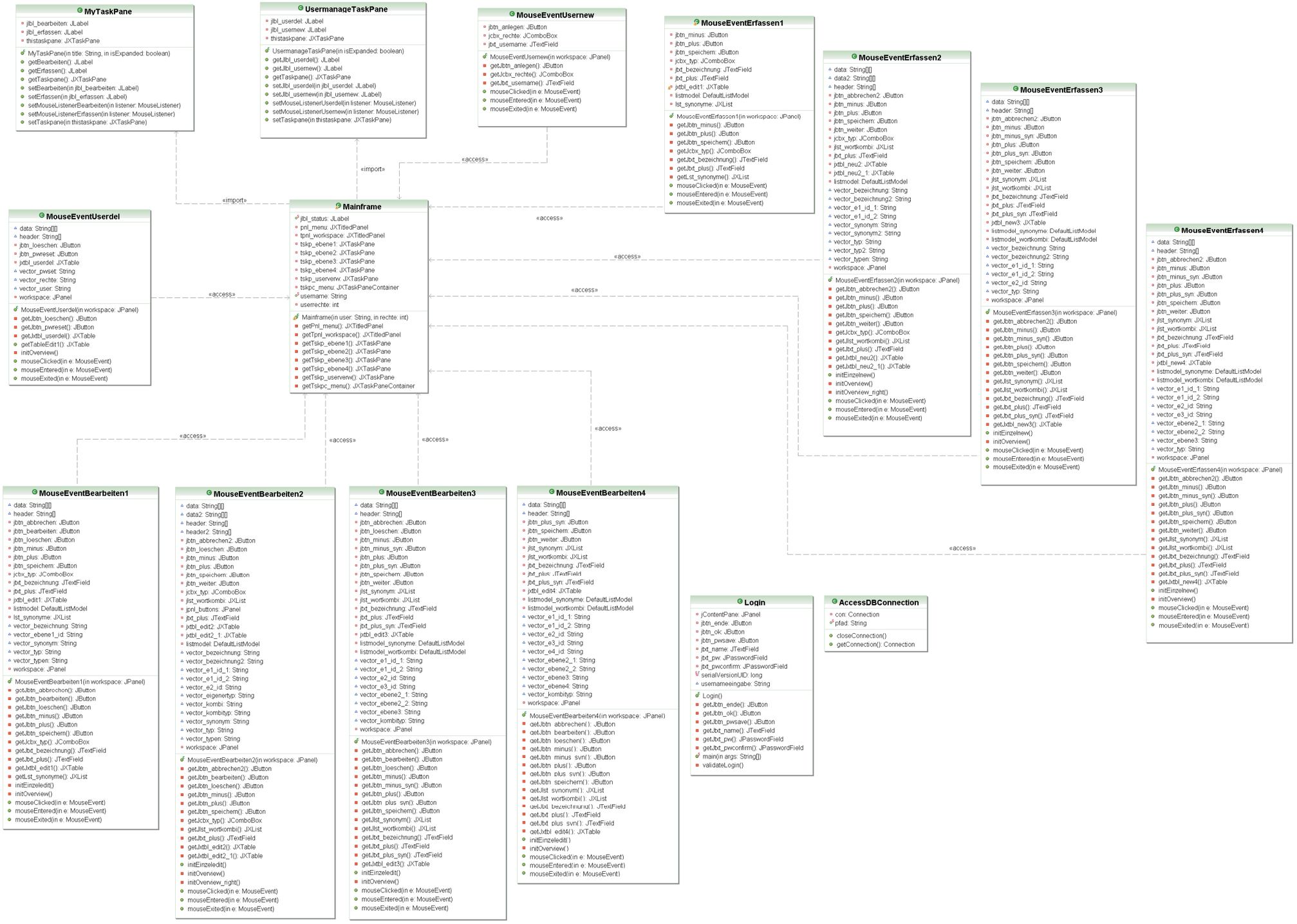
Berlin, den 22.10.2007

Dr. Manfred Sommerer	René Mäurer
Auftraggeber	*Auftragnehmer*

8.4.3 GUI-Übersicht

8.4.4 UML-Klassendiagramm

(die Referenzen auf Labels, Panels und Scrollpanes, sowie die Zugriffe zur Klasse „*AccessDBConnection*“ werden zur besseren Übersichtlichkeit nicht mit dargestellt)



8.4.5 Testdokumentation (Black-Box-Test)

Testdokumentation

– Black-Box-Test –

Folgende Funktionalitäten wurden anhand der Spezifikationen der GUI auf vollständige und korrekte Verarbeitung getestet:

Login-Formular

- Falsche Benutzernamen und Passwörter führen zu Fehlermeldungen. ☑
- Bei der (Erst-)Vergabe eines Passwortes für Benutzer ohne gesetztes Passwort wird eine Fehlermeldung ausgegeben, wenn das Passwort und die Bestätigung nicht übereinstimmen. ☑
- Das (erst-)vergebene Passwort wird korrekt gespeichert; ein Login muss nun möglich sein. ☑

Hauptfenster

- Die Nutzer-Verwaltung ist nur für administrative Nutzer sichtbar. ☑

1. Ebene – Erfassen

- Der Versuch zu speichern, ohne eine Bezeichnung eingegeben zu haben führt zu einer Fehlermeldung. ☑
- Synonyme lassen sich korrekt hinzufügen/entfernen. ☑
- Alle Typen werden angezeigt. ☑
- Ausgabe, dass die Daten in die Datenbank geschrieben wurden. ☑

1. Ebene – Bearbeiten

- Es werden alle bereits erfassten Begriffe aufgelistet. ☑
- Das Bearbeiten-Fenster (über Doppelklick oder den Button erreichbar) enthält die korrekten Werte. ☑
- Synonyme lassen sich korrekt hinzufügen/entfernen. ☑
- Alle Typen werden angezeigt. ☑
- Der Versuch zu speichern, ohne eine Bezeichnung eingegeben zu haben führt zu einer Fehlermeldung. ☑
- Ausgabe, dass die Daten in die Datenbank geschrieben wurden. ☑
- Die Anzeige nach dem Speichern wird korrekt aktualisiert. ☒
- Klick auf Löschen, ohne Auswahl eines Eintrages führt zu einer Fehlermeldung. ☑

- Korrektes Löschen eines Eintrags. ☑

2. Ebene – Erfassen

- Es werden die korrekten Daten in der Übersicht angezeigt (in der rechten Tabelle dürfen keine Einträge vorkommen, die bereits zugeordnet sind). ☑
- Wortkombinationen lassen sich korrekt hinzufügen/entfernen. ☑
- Alle Typen werden angezeigt. ☑
- Ausgabe, dass die Daten in die Datenbank geschrieben wurden. ☑
- Die Anzeige nach dem Speichern wird korrekt aktualisiert. ☑
- Das Erfassen wird für beide explizit zusammengehörigen Reihenfolgen übernommen. ☑

2. Ebene – Bearbeiten

- Es werden alle bereits erfassten Begriffe aufgelistet. ☑
- Das Bearbeiten-Fenster (über Doppelklick oder den Button erreichbar) enthält die korrekten Werte. ☑
- Wortkombinationen lassen sich korrekt hinzufügen/entfernen. ☑
- Alle Typen werden angezeigt. ☑
- Ausgabe, dass die Daten in die Datenbank geschrieben wurden. ☑
- Die Anzeige nach dem Speichern wird korrekt aktualisiert. ☑
- Klick auf Löschen, ohne Auswahl eines Eintrages führt zu einer Fehlermeldung ☑
- Korrektes Löschen eines Eintrags. ☑
- Löschen/Bearbeiten wird für beide explizit zusammengehörigen Reihenfolgen übernommen. ☑

3. Ebene – Erfassen

- Es werden die korrekten Daten in der Übersicht angezeigt. ☑
- Wortkombinationen lassen sich korrekt hinzufügen/entfernen. ☑
- Synonyme lassen sich korrekt hinzufügen/entfernen. ☑
- Der Versuch zu speichern, ohne eine Bezeichnung eingegeben zu haben führt zu einer Fehlermeldung. ☑
- Ausgabe, dass die Daten in die Datenbank geschrieben wurden. ☑
- Die Anzeige nach dem Speichern wird korrekt aktualisiert. ☑

➢ Das Erfassen wird für beide explizit zusammengehörigen Reihenfolgen übernommen.	☑

3. Ebene – Bearbeiten

➢ Es werden alle bereits erfassten Begriffe aufgelistet.	☑
➢ Das Bearbeiten-Fenster (über Doppelklick oder den Button erreichbar) enthält die korrekten Werte.	☑
➢ Wortkombinationen lassen sich korrekt hinzufügen/entfernen.	☑
➢ Synonyme lassen sich korrekt hinzufügen/entfernen.	☑
➢ Der Versuch zu speichern, ohne eine Bezeichnung eingegeben zu haben führt zu einer Fehlermeldung.	☑
➢ Ausgabe, dass die Daten in die Datenbank geschrieben wurden.	☑
➢ Die Anzeige nach dem Speichern wird korrekt aktualisiert.	☒
➢ Klick auf Löschen, ohne Auswahl eines Eintrages führt zu einer Fehlermeldung.	☑
➢ Korrektes Löschen eines Eintrags.	☑
➢ Das Löschen/Bearbeiten wird für beide explizit zusammengehörigen Reihenfolgen übernommen.	☑

4. Ebene – Erfassen

➢ Es werden die korrekten Daten in der Übersicht angezeigt.	☑
➢ Wortkombinationen lassen sich korrekt hinzufügen/entfernen.	☑
➢ Synonyme lassen sich korrekt hinzufügen/entfernen.	☑
➢ Der Versuch zu speichern, ohne eine Bezeichnung eingegeben zu haben führt zu einer Fehlermeldung.	☑
➢ Ausgabe, dass die Daten in die Datenbank geschrieben wurden.	☑
➢ Die Anzeige nach dem Speichern wird korrekt aktualisiert.	☒
➢ Das Erfassen wird für beide explizit zusammengehörigen Reihenfolgen übernommen.	☑

4. Ebene – Bearbeiten

➢ Es werden alle bereits erfassten Begriffe aufgelistet.	☑
➢ Das Bearbeiten-Fenster (über Doppelklick oder den Button erreichbar) enthält die korrekten Werte.	☑

➢ Wortkombinationen lassen sich korrekt hinzufügen/entfernen.	☑
➢ Synonyme lassen sich korrekt hinzufügen/entfernen.	☑
➢ Der Versuch zu speichern, ohne eine Bezeichnung eingegeben zu haben führt zu einer Fehlermeldung.	☑
➢ Ausgabe, dass die Daten in die Datenbank geschrieben wurden.	☑
➢ Die Anzeige nach dem Speichern wird korrekt aktualisiert.	☒
➢ Klick auf Löschen, ohne Auswahl eines Eintrages führt zu einer Fehlermeldung.	☑
➢ Korrektes Löschen eines Eintrags.	☑
➢ Das Löschen/Bearbeiten wird für beide explizit zusammengehörigen Reihenfolgen übernommen.	☑

Nutzer-Verwaltung – Nutzer Anlegen

➢ Das Anlegen ohne Angabe eines Nutzernamens führt zu einer Fehlermeldung.	☑
➢ Beide Nutzerrechte-Möglichkeiten werden angezeigt.	☑
➢ Das Speichern wird bestätigt.	☑

Nutzer-Verwaltung – Nutzer/PW löschen

➢ Alle Nutzer werden korrekt mit ihren Rechten angezeigt.	☑
➢ Das Zurücksetzten eines Passwortes, ohne eine Eintrag in der Tabelle ausgewählt zu haben führt zu einer Fehlermeldung.	☑
➢ Das Löschen eines Nutzers, ohne eine Eintrag in der Tabelle ausgewählt zu haben führt zu einer Fehlermeldung.	☑
➢ Nachdem ein Passwort oder Nutzer gelöscht wurde, wird die Übersichtstabelle aktualisiert.	☑

Berlin, den 07.11.2007

Der Test wurde durchgeführt / zur Kenntnis genommen von…

Dr. Manfred Sommerer
Auftraggeber

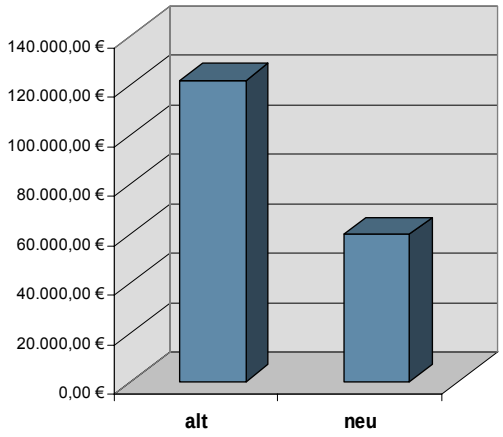
Nico Wohllebe
Künftiger Nutzer der Software

René Mäurer
Auftragnehmer

8.4.6 Nutzwertanalyse

Anzahl Begriffe[10]	325.000
durchschnittlicher Stundenlohn Tierarzt	30,00 €

	alt	neu
Erfassungszeit pro Begriff	45 s	20 s
Gesamtzeit (Stunden)	4.063 h	1.806 h
Lohnkosten Tierarzt	121.890,00 €	54.180,00 €
+ Kosten für das Programm	0,00 €	5.414,50 €
Gesamtkosten	**121.890,00 €**	**59.594,50 €**
Differenz	**62.295,50 €**	



Kostenvergleich Nutzwertanalyse

[10] ICD-10-GM (internationale statistische Klassifikation der Krankheiten […]; „Diagnosenthesaurus“) - Okt 05; die Anzahl der Begriffe sind ein an den Diagnosethesaurus-Katalog der Humanmedizin angelehnter Wert

8.4.7 Angebot

Deutsche Rentenversicherung Bund
René Mäurer
Soorstraße 80-82
14050 Berlin
E-Mail: maeurer.rene@vetmed.fu-berlin.de

Berlin, 16. Oktober 2007

Freie Universität Berlin
Fachbereich Veterinärmedizin
IT-Abteilung, Dr. Manfred Sommerer
Pathologiegebäude, Hs. 31
Robert-von-Ostertag-Strasse 15
14163 Berlin

Angebot zur Erstellung der Software „Behandlungs-, Befund- und Diagnoseschlüsselverwaltung“

Sehr geehrter Herr Dr. Sommerer,

ich freue mich über das von Ihnen entgegengebrachte Vertrauen und hoffe, dass Ihnen folgendes Angebot zusagt:

Stundenlohn:	65,00 €
Arbeitszeit:	70,00 h
Gesamtpreis (netto):	*4.550,00 €*
+ 19,00 % MwSt:	864,50 €
Gesamtpreis (brutto):	**5.414,50 €**

Verpackungs- und Lieferungskosten fallen nicht an.
Der Gesamtbetrag ist bis spätestens 14 Tagen nach erfolgter Leistung auf mein Konto zu überweisen.
Bei Zahlung innerhalb von 5 Werktagen gewähre ich ein Skonto von 3 %.

Für weitere Fragen stehe ich Ihnen unter der E-Mailadresse maeurer.rene@vetmed.fu-berlin.de zur Verfügung.

Ich hoffe Ihnen ein für Sie optimales Angebot unterbreitet zu haben und freue mich auf eine erfolgreiche Zusammenarbeit.

Mit Freundlichen Grüßen

René Mäurer

8.4.8 Rechnung

(Anmerkung: die folgende Rechnung ist als fiktiv zu betrachten)

Deutsche Rentenversicherung Bund
René Mäurer
Soorstraße 80-82
14050 Berlin
Email: maeurer.rene@vetmed.fu-berlin.de

Berlin, 09. November 2007

Freie Universität Berlin
Fachbereich Veterinärmedizin
IT-Abteilung, Dr. Manfred Sommerer
Pathologiegebäude, Hs. 31
Robert-von-Ostertag-Strasse 15
14163 Berlin

Rechnung über die Erstellung der Software „Behandlungs-, Befund- und Diagnoseschlüsselverwaltung“

Sehr geehrter Herr Dr. Sommerer,

über die erbrachte Leistung stelle ich Ihnen folgende Rechnung:

Stundenlohn:	65,00 €
Arbeitszeit:	70,00 h
Gesamtpreis (netto):	*4.550,00 €*
+ 19,00 % MwSt:	864,50 €
Gesamtpreis (brutto):	**5.414,50 €**

Bitte überweisen Sie den oben angegeben Gesamtbetrag innerhalb von 14 Tagen auf mein Geschäftskonto:

Kontoinhaber: René Mäurer
Kontonummer: 123 456 0815
Bankleitzahl: 100 100 0
Kreditinstitut: Berliner Bank

Bei Zahlung innerhalb von 5 Werktagen gewähre ich ein Skonto von 3 %.

Ich hoffe auf weiterhin erfolgreiche Zusammenarbeit.

Mit Freundlichen Grüßen

René Mäurer

8.4.9 Übergabeprotokoll

Übergabe- und Abnahmeprotokoll

Hiermit bestätigt der Auftraggeber

Herr Dr. Manfred Sommerer

dem Auftragnehmer

Herrn René Mäurer

die korrekte und vollständige Erfüllung und Übergabe der Dienstleistung zur Erstellung einer Software zur Behandlungs- Befund- und Diagnose-Verwaltung.

Berlin, den 09.11.2007

Dr. Manfred Sommerer
Auftraggeber

René Mäurer
Auftragnehmer

8.4.10 Java-Docs

de

Class AccessDBConnection

```
java.lang.Object
  └ de.AccessDBConnection
```

```
public class AccessDBConnection
extends java.lang.Object
```

Diese Klasse dient dem Auf- und Abbau einer Verbindung zur Access-DB.

Field Summary

static java.lang.String	**pfad**

Constructor Summary

AccessDBConnection()

Method Summary

void	**closeConnection**() Methode zum schließen der Verbindung
java.sql.Connection	**getConnection**() Methode, die ein Connection-Objekt zur DB zurückgibt

Field Detail

pfad

```
public static java.lang.String pfad
```

Constructor Detail

AccessDBConnection

```
public AccessDBConnection()
```

Method Detail

getConnection

```
public java.sql.Connection getConnection()
```

Methode, die ein Connection-Objekt zur DB zurückgibt

---

closeConnection

```
public void closeConnection()
```

Methode zum Schließen der Verbindung.

de

Class Login

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
                      └de.Login
```

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

```
public class Login extends javax.swing.JFrame
```

Diese Klasse ist das Login-Fenster und gleichzeitig Einstiegspunkt (main).

Nested Class Summary

Nested classes/interfaces inherited from class java.awt.Component
java.awt.Component.BaselineResizeBehavior

Field Summary

Fields inherited from class javax.swing.JFrame
...

Fields inherited from class java.awt.Frame
...

Fields inherited from class java.awt.Component
...

Fields inherited from interface javax.swing.WindowConstants
...

Fields inherited from interface java.awt.image.ImageObserver
...

Constructor Summary

Login() Konstruktor der Klasse "Login" (setzen der entspr. Eigenschaften des Frames)

Method Summary

static void	**main**(java.lang.String[] args) Einstiegspunkt des Programms (main).

Methods inherited from class javax.swing.JFrame

...

Methods inherited from class java.awt.Frame

...

Methods inherited from class java.awt.Window

...

Methods inherited from class java.awt.Container

...

Methods inherited from class java.awt.Component

...

Methods inherited from class java.lang.Object

...

Methods inherited from interface java.awt.MenuContainer

...

Constructor Detail

Login

```
public Login()
```

Konstruktor der Klasse "Login" (setzen der entspr. Eigenschaften des Frames).

Method Detail

main

```
public static void main(java.lang.String[] args)
```

Einstiegspunkt des Programms (main).

de

Class Mainframe

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
                      └de.Mainframe
```

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

```
public class Mainframe
extends javax.swing.JFrame
```

Diese Klasse ist das Hauptfenster des Programms.

Nested Class Summary

Nested classes/interfaces inherited from class java.awt.Component
java.awt.Component.BaselineResizeBehavior

Field Summary

static javax.swing.JLabel	**jlbl_status**
static java.lang.String	**username**

Fields inherited from class javax.swing.JFrame
...

Fields inherited from class java.awt.Frame
...

Fields inherited from class java.awt.Component
...

Fields inherited from interface javax.swing.WindowConstants
...

Fields inherited from interface java.awt.image.ImageObserver
...

Constructor Summary

Mainframe(java.lang.String user, int rechte)

parametrisierter Konstruktor der Klasse mit Übergabe von Benutzername und -rechte.

Method Summary

Methods inherited from class javax.swing.JFrame

...

Methods inherited from class java.awt.Frame

...

Methods inherited from class java.awt.Window

...

Methods inherited from class java.awt.Container

...

Methods inherited from class java.awt.Component

...

Methods inherited from class java.lang.Object

...

Methods inherited from interface java.awt.MenuContainer

...

Field Detail

username

```
public static java.lang.String username
```

---

jlbl_status

```
public static javax.swing.JLabel jlbl_status
```

Constructor Detail

Mainframe

```
public Mainframe(java.lang.String user, int rechte)
```

parametrisierter Konstruktor der Klasse mit Übergabe von Benutzername und -rechte.

Parameters:

`user` - der Username

`rechte` - die Rechte des Users

de

Class MouseEventErfassen1

```
java.lang.Object
  └ de.MouseEventErfassen1
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventErfassen1
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Erfassen"-Feld der Ebene 1 hier wird definiert, was bei best. Mouse-Ereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird

---

Constructor Summary

`MouseEventErfassen1(javax.swing.JPanel workspace)`	Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Summary

void	`mouseClicked(java.awt.event.MouseEvent e)` Auf Mausklick auf das "Erfassen"-Label
void	`mouseEntered(java.awt.event.MouseEvent e)` Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	`mouseExited(java.awt.event.MouseEvent e)` Wenn der Mouse-Zeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	`mousePressed(java.awt.event.MouseEvent e)`
void	`mouseReleased(java.awt.event.MouseEvent e)`

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventErfassen1

`public MouseEventErfassen1(javax.swing.JPanel workspace)`

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes) .

Method Detail

mouseClicked

```
public void mouseClicked(java.awt.event.MouseEvent e)
```

auf Mausklick auf das "Erfassen"-Label

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mouseEntered

```
public void mouseEntered(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited

```
public void mouseExited(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mousePressed

```
public void mousePressed(java.awt.event.MouseEvent e)
```

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MouseEventBearbeiten1

```
java.lang.Object
  └ de.MouseEventBearbeiten1
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventBearbeiten1
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Bearbeiten"-Feld der Ebene 1 hier wird definiert, was bei best. Mausereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird.

Constructor Summary

MouseEventBearbeiten1(javax.swing.JPanel workspace) Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).	

Method Summary

void	**mouseClicked**(java.awt.event.MouseEvent e) Auf Mausklick auf das "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.
void	**mouseEntered**(java.awt.event.MouseEvent e) Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	**mouseExited**(java.awt.event.MouseEvent e) Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventBearbeiten1

```
public MouseEventBearbeiten1(javax.swing.JPanel workspace)
```

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Detail

mouseClicked

```
public void mouseClicked(java.awt.event.MouseEvent e)
```

Auf Mausklick auf's "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mouseEntered

```
public void mouseEntered(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited

```
public void mouseExited(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mousePressed

```
public void mousePressed(java.awt.event.MouseEvent e)
```

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MouseEventErfassen2

```
java.lang.Object
  └ de.MouseEventErfassen2
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventErfassen2
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Erfassen"-Feld der Ebene 2 hier wird definiert, was bei best. Mausereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird.

Constructor Summary

MouseEventErfassen2(javax.swing.JPanel workspace)	Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Summary

void	**initEinzelnew**() Private Methode zum Initialisieren des Einzel-Erfassungs-Fensters.
void	**mouseClicked**(java.awt.event.MouseEvent e) auf Mausklick auf das "Erfassen"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.
void	**mouseEntered**(java.awt.event.MouseEvent e) Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	**mouseExited**(java.awt.event.MouseEvent e) Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventErfassen2

```
public MouseEventErfassen2(javax.swing.JPanel workspace)
```

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Detail

initEinzelnew

`public void` **`initEinzelnew`**`()`

Private Methode zum Initialisieren des Einzel-Erfassungs-Fensters.

---

mouseClicked

`public void` **`mouseClicked`**`(java.awt.event.MouseEvent e)`

Auf Mausklick auf das "Erfassen"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mouseEntered

`public void` **`mouseEntered`**`(java.awt.event.MouseEvent e)`

Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited

`public void` **`mouseExited`**`(java.awt.event.MouseEvent e)`

Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mousePressed

`public void` **`mousePressed`**`(java.awt.event.MouseEvent e)`

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

`public void` **`mouseReleased`**`(java.awt.event.MouseEvent e)`

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MouseEventBearbeiten2

```
java.lang.Object
  └ de.MouseEventBearbeiten2
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventBearbeiten2
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Bearbeiten"-Feld der Ebene 2 hier wird definiert, was bei best. Mauseereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird.

Constructor Summary

MouseEventBearbeiten2(javax.swing.JPanel workspace) Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).	

Method Summary

void	**mouseClicked**(java.awt.event.MouseEvent e) Auf Mausklick auf das "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.
void	**mouseEntered**(java.awt.event.MouseEvent e) Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	**mouseExited**(java.awt.event.MouseEvent e) Wenn der Mousezeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventBearbeiten2

```
public MouseEventBearbeiten2(javax.swing.JPanel workspace)
```

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Detail

mouseClicked

```
public void mouseClicked(java.awt.event.MouseEvent e)
```

Auf Mausklick auf das "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mouseEntered

```
public void mouseEntered(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited

```
public void mouseExited(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mousePressed

```
public void mousePressed(java.awt.event.MouseEvent e)
```

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MouseEventErfassen3

```
java.lang.Object
  └ de.MouseEventErfassen3
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventErfassen3
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Erfassen"-Feld der Ebene 3 hier wird definiert, was bei best. Mauseereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird.

Constructor Summary

MouseEventErfassen3(javax.swing.JPanel workspace)	Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Summary

void	**initEinzelnew**() private Methode zum Initialisieren des Einzel-Erfassungs-Fensters
void	**mouseClicked**(java.awt.event.MouseEvent e) Auf Mausklick auf das "Erfassen"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.
void	**mouseEntered**(java.awt.event.MouseEvent e) Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	**mouseExited**(java.awt.event.MouseEvent e) Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventErfassen3

```
public MouseEventErfassen3(javax.swing.JPanel workspace)
```

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Detail

initEinzelnew

`public void` **`initEinzelnew`**`()`

Private Methode zum Initialisieren des Einzel-Erfassungs-Fensters.

---

mouseClicked

`public void` **`mouseClicked`**`(java.awt.event.MouseEvent e)`

Auf Mausklick auf das "Erfassen"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mouseEntered

`public void` **`mouseEntered`**`(java.awt.event.MouseEvent e)`

Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited

`public void` **`mouseExited`**`(java.awt.event.MouseEvent e)`

Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mousePressed

`public void` **`mousePressed`**`(java.awt.event.MouseEvent e)`

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

`public void` **`mouseReleased`**`(java.awt.event.MouseEvent e)`

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MouseEventBearbeiten3

```
java.lang.Object
  └ de.MouseEventBearbeiten3
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventBearbeiten3
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Bearbeiten"-Feld der Ebene 3 hier wird definiert, was bei best. Mauseereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird.

Constructor Summary

MouseEventBearbeiten3(javax.swing.JPanel workspace) Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).	

Method Summary

void	**mouseClicked**(java.awt.event.MouseEvent e) Auf Mausklick auf das "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.
void	**mouseEntered**(java.awt.event.MouseEvent e) wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	**mouseExited**(java.awt.event.MouseEvent e) Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventBearbeiten3

```
public MouseEventBearbeiten3(javax.swing.JPanel workspace)
```

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Detail

mouseClicked

```
public void mouseClicked(java.awt.event.MouseEvent e)
```

Auf Mausklick auf das "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mouseEntered

```
public void mouseEntered(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited

```
public void mouseExited(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mousePressed

```
public void mousePressed(java.awt.event.MouseEvent e)
```

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MouseEventErfassen4

```
java.lang.Object
  └ de.MouseEventErfassen4
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventErfassen4
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Erfassen"-Feld der Ebene 4 hier wird definiert, was bei best. Mausereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird.

Constructor Summary

MouseEventErfassen4(javax.swing.JPanel workspace)	Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Summary

void	**initEinzelnew**() Private Methode zum Initialisieren des Einzel-Erfassungs-Fensters.
void	**mouseClicked**(java.awt.event.MouseEvent e) Auf Mausklick auf das "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.
void	**mouseEntered**(java.awt.event.MouseEvent e) Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	**mouseExited**(java.awt.event.MouseEvent e) Wenn der Mauszeiger das "Erfassen"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventErfassen4

public **MouseEventErfassen4**(javax.swing.JPanel workspace)

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Detail

initEinzelnew

```
public void initEinzelnew()
```

Private Methode zum Initialisieren des Einzel-Erfassungs-Fensters.

---

mouseClicked

```
public void mouseClicked(java.awt.event.MouseEvent e)
```

Auf Mausklick auf das "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mouseEntered

```
public void mouseEntered(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited

```
public void mouseExited(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger das "Erfassen"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mousePressed

```
public void mousePressed(java.awt.event.MouseEvent e)
```

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MouseEventBearbeiten4

```
java.lang.Object
  └ de.MouseEventBearbeiten4
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventBearbeiten4
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Bearbeiten"-Feld der Ebene 4 hier wird definiert, was bei best. Mausereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird.

Constructor Summary

MouseEventBearbeiten4(javax.swing.JPanel workspace)	Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Summary

void	**mouseClicked**(java.awt.event.MouseEvent e) auf Mausklick auf das "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.
void	**mouseEntered**(java.awt.event.MouseEvent e) Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	**mouseExited**(java.awt.event.MouseEvent e) Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventBearbeiten4

```
public MouseEventBearbeiten4(javax.swing.JPanel workspace)
```

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Detail

mouseClicked

```
public void mouseClicked(java.awt.event.MouseEvent e)
```

Auf Mausklick auf das "Bearbeiten"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mouseEntered

```
public void mouseEntered(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger über dem Bearbeiten-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited

```
public void mouseExited(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger das "Bearbeiten"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mousePressed

```
public void mousePressed(java.awt.event.MouseEvent e)
```

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MouseEventUserdel

```
java.lang.Object
  └ de.MouseEventUserdel
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventUserdel
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Nutzer/PW löschen"-Feld der Nutzerverwaltung hier wird definiert, was bei best. Mouseereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird.

Constructor Summary

MouseEventUserdel(javax.swing.JPanel workspace)	Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes)

Method Summary

void	**mouseClicked**(java.awt.event.MouseEvent e) Auf Mausklick auf das "Nutzer/PW löschen"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen
void	**mouseEntered**(java.awt.event.MouseEvent e) Wenn der Mauszeiger über dem Nutzer/PW löschen-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	**mouseExited**(java.awt.event.MouseEvent e) Wenn der Mauszeiger das "Nutzer/PW löschen"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventUserdel

```
public MouseEventUserdel(javax.swing.JPanel workspace)
```

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Detail

mouseExited

```
public void mouseExited(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger das "Nutzer/PW löschen"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mouseEntered

```
public void mouseEntered(java.awt.event.MouseEvent e)
```

Wenn der Mauszeiger über dem Nutzer/PW löschen-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseClicked

```
public void mouseClicked(java.awt.event.MouseEvent e)
```

Auf Mausklick auf das "Nutzer/PW löschen"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mousePressed

```
public void mousePressed(java.awt.event.MouseEvent e)
```

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MouseEventUsernew

```
java.lang.Object
  └ de.MouseEventUsernew
```

All Implemented Interfaces:

java.awt.event.MouseListener, java.util.EventListener

```
public class MouseEventUsernew
extends java.lang.Object
implements java.awt.event.MouseListener
```

Diese Klasse ist der MouseListener für das "Nutzer anlegen"-Feld der Nutzerverwaltung hier wird definiert, was bei best. Mouseereignissen (Klick, Hover) passieren soll ferner stellt diese Klasse das Panel dar, welches auf Klick in der rechten Hälfte des Mainframes geladen wird.

Constructor Summary

	MouseEventUsernew(javax.swing.JPanel workspace) Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Summary

void	**mouseClicked**(java.awt.event.MouseEvent e) Auf Mausklick auf das "Nutzer anlegen"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.
void	**mouseEntered**(java.awt.event.MouseEvent e) Wenn der Mauszeiger über dem Nutzer anlegen-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).
void	**mouseExited**(java.awt.event.MouseEvent e) Wenn der Mauszeiger das "Nutzer anlegen"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.lang.Object
...

Constructor Detail

MouseEventUsernew

```
public MouseEventUsernew(javax.swing.JPanel workspace)
```

Konstruktor der Klasse mit Übergabe einer Referenz auf den "workspace" (der rechte Teile des Mainframes).

Method Detail

mouseExited

`public void` **`mouseExited`**`(java.awt.event.MouseEvent e)`

Wenn der Mauszeiger das "Nutzer anlegen"-Label wieder verlässt wird ein nicht unterstrichener Schriftzug dargestellt (Hover-Effekt).

Specified by:

`mouseExited` in interface `java.awt.event.MouseListener`

---

mouseEntered

`public void` **`mouseEntered`**`(java.awt.event.MouseEvent e)`

Wenn der Mauszeiger über dem Nutzer anlegen-Label steht, wird der Schriftzug unterstrichen dargestellt (Hover-Effekt) (Anm.: JLabels können html-Code interpretieren).

Specified by:

`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseClicked

`public void` **`mouseClicked`**`(java.awt.event.MouseEvent e)`

Auf Mausklick auf das "Nutzer anlegen"-Label wird das Status-Label "zurückgesetzt" und die Methode zur Übersicht aufgerufen.

Specified by:

`mouseClicked` in interface `java.awt.event.MouseListener`

---

mousePressed

`public void` **`mousePressed`**`(java.awt.event.MouseEvent e)`

Specified by:

`mousePressed` in interface `java.awt.event.MouseListener`

---

mouseReleased

`public void` **`mouseReleased`**`(java.awt.event.MouseEvent e)`

Specified by:

`mouseReleased` in interface `java.awt.event.MouseListener`

---

de

Class MyTaskPane

```
java.lang.Object
  └ de.MyTaskPane
```

```
public class MyTaskPane
extends java.lang.Object
```

Diese Klasse beschreibt die Steuerelemente, die ein Task-Pane für die Ebenen jeweils hat.

Constructor Summary

MyTaskPane(java.lang.String title, boolean isExpanded) Konstruktor der Klasse (mit Übergabe des Titels des TaskPanes und dem gewünschtem Ausklapp-Status).	

Method Summary

javax.swing.JLabel	**getBearbeiten**()
javax.swing.JLabel	**getErfassen**()
org.jdesktop.swingx.JXTaskPane	**getTaskpane**()
void	**setBearbeiten**(javax.swing.JLabel jlbl_bearbeiten)
void	**setErfassen**(javax.swing.JLabel jlbl_erfassen)
void	**setMouseListenerBearbeiten**(java.awt.event.MouseListener listener) Setter zum hinzufügen eines MouseListener auf die Labels, sodass individuell reagiert werden kann, was auf Klick (o.ä.) passiert.
void	**setMouseListenerErfassen**(java.awt.event.MouseListener listener)
void	**setTaskpane**(org.jdesktop.swingx.JXTaskPane thistaskpane)

Methods inherited from class java.lang.Object
...

Constructor Detail

MyTaskPane

```
public MyTaskPane(java.lang.String title,
                  boolean isExpanded)
```

Konstruktor der Klasse (mit Übergabe des Titels des TaskPanes und dem gewünschtem Ausklapp-Status).

Method Detail

getBearbeiten
```
public javax.swing.JLabel getBearbeiten()
```

---

setBearbeiten
```
public void setBearbeiten(javax.swing.JLabel jlbl_bearbeiten)
```

---

getErfassen
```
public javax.swing.JLabel getErfassen()
```

---

setErfassen
```
public void setErfassen(javax.swing.JLabel jlbl_erfassen)
```

---

getTaskpane
```
public org.jdesktop.swingx.JXTaskPane getTaskpane()
```

---

setTaskpane
```
public void setTaskpane(org.jdesktop.swingx.JXTaskPane thistaskpane)
```

---

setMouseListenerBearbeiten
```
public void setMouseListenerBearbeiten(java.awt.event.MouseListener
                                       listener)
```
Setter zum hinzufügen eines MouseListener auf die Labels, sodass individuell reagiert werden kann, was auf Klick (o.ä.) passiert.

---

setMouseListenerErfassen
```
public void setMouseListenerErfassen(java.awt.event.MouseListener listener)
```

---

de

Class UsermanageTaskPane

```
java.lang.Object
  └ de.UsermanageTaskPane
```

```
public class UsermanageTaskPane
extends java.lang.Object
```

Diese Klasse beschreibt die Steuerelemente, die das Task-Pane für die Nutzerverwaltung haben soll.

Constructor Summary

UsermanageTaskPane(boolean isExpanded) Konstruktor der Klasse (mit Übergabe des gewünschtem Ausklapp-Status).	

Method Summary

javax.swing.JLabel	**getJlbl_userdel**()
javax.swing.JLabel	**getJlbl_usernew**()
org.jdesktop.swingx.JXTaskPane	**getTaskpane**()
void	**setJlbl_userdel**(javax.swing.JLabel jlbl_userdel)
void	**setJlbl_usernew**(javax.swing.JLabel jlbl_usernew)
void	**setMouseListenerUserdel**(java.awt.event.MouseListener listener)
void	**setMouseListenerUsernew**(java.awt.event.MouseListener listener) Setter zum hinzufügen eines MouseListener auf die Labels, sodass individuell reagiert werden kann, was auf Klick (o.ä.) passiert.
void	**setTaskpane**(org.jdesktop.swingx.JXTaskPane thistaskpane)

Methods inherited from class java.lang.Object
...

Constructor Detail

UsermanageTaskPane

```
public UsermanageTaskPane(boolean isExpanded)
```

Konstruktor der Klasse (mit Übergabe des gewünschten Ausklapp-Status).

Method Detail

getTaskpane

```
public org.jdesktop.swingx.JXTaskPane getTaskpane()
```

---

setTaskpane
```
public void setTaskpane(org.jdesktop.swingx.JXTaskPane thistaskpane)
```

---

getJlbl_userdel
```
public javax.swing.JLabel getJlbl_userdel()
```

---

setJlbl_userdel
```
public void setJlbl_userdel(javax.swing.JLabel jlbl_userdel)
```

---

getJlbl_usernew
```
public javax.swing.JLabel getJlbl_usernew()
```

---

setJlbl_usernew
```
public void setJlbl_usernew(javax.swing.JLabel jlbl_usernew)
```

---

setMouseListenerUsernew
```
public void setMouseListenerUsernew(java.awt.event.MouseListener listener)
```
Setter zum hinzufügen eines MouseListener auf die Labels, sodass individuell reagiert werden kann, was auf Klick (o.ä.) passiert.

---

setMouseListenerUserdel
```
public void setMouseListenerUserdel(java.awt.event.MouseListener listener)
```
