Deutsche Rentenversicherung Bund

Projektdokumentation zur Winterprüfung 2007
- TroubleShooter -

Projektbezeichnung

Erstellen einer Applikation zur Verwaltung einer Best-Practices-Sammlung für den Kundensupport der IT-Abteilung des Fachbereichs Veterinärmedizin der Freien Universität Berlin mit Microsoft Access 2003 und Java 1.6

Prüfungsteilnehmer

Marcel Wieczorek (Azubi-Ident: 10703435414)
Prenzlauer Allee 122
10409 Berlin
Email: ***marcel_wieczorek@hotmail.com***

Ausbildungsbetrieb

Deutsche Rentenversicherung Bund
Ruhrstraße 2
10704 Berlin

Prüfungsbetrieb

Freie Universität Berlin
Robert-von-Ostertag-Strasse 15
14163 Berlin

Projektverantwortlicher

Herr Dr. Manfred Sommerer
Freie Universität Berlin
IT-Abteilung (Fachbereich Veterinärmedizin)
Email: ***sommerer.manfred@vetmed.fu-berlin.de***

Ausbildungsberuf

Fachinformatiker / Anwendungsentwicklung

Prüfungsausschuss

FIAN1

Ausführungszeitraum

Beginn: 19.11.2007
Ende: 30.11.2007

Inhaltsverzeichnis

1 Einführung

1.1 Ausbildungsbetrieb

Die Deutsche Rentenversicherung Bund ist Europas größter gesetzlicher Rentenversicherungsträger. Derzeit verzeichnet sie mehr als 57 Millionen Kunden und beschäftigt über 70.000 Mitarbeiter.
Der Leistungsumfang reicht von der Zahlung von Renten an Versicherte oder deren Angehörige über individuelle Beratung in allen Rentenfragen bis hin zur medizinischen und beruflichen Rehabilitation.

2 Projektauftrag

2.1 Auftraggeber

Der Auftraggeber für dieses Projekt ist die IT-Abteilung des Fachbereichs Veterinärmedizin der Freien Universität Berlin. Hier wurde das Projekt auch realisiert.
Hauptansprechpartner und Projektbetreuer ist Herr Dr. Manfred Sommerer (IT-Verantwortlicher, Fachtierarzt für Informationstechnologie).
Die IT-Abteilung betreut derzeit etwa 1.000 Mitarbeiter mit ca. 700 Arbeitsplatzrechnern an verschiedenen Standorten (Düppel, Dahlem, Mitte).
Immer wenn am Fachbereich software- oder hardwaretechnische Probleme auftreten, ist die IT-Abteilung die erste Anlaufstelle. Die Störfälle werden hier entweder telefonisch oder per Email gemeldet.

2.2 Projektbeschreibung

Für die IT-Abteilung des Fachbereichs Veterinärmedizin der Freien Universität Berlin soll zur Unterstützung des Supportteams eine Applikation entwickelt werden, mit der die Verwaltung einer Best-Practices-Sammlung realisiert werden kann.
Leider werden bisher beim Kundensupport Problemlösungsstrategien nur rudimentär und nahezu ungeordnet dokumentiert, was dazu führt, dass für gleiche oder ähnliche Probleme häufig aufs Neue überlegt und eine Lösungsstrategie erarbeitet werden muss.
Der entstehende Zeitverzug wirkt sich v.a. bei der Problembearbeitung durch neue Mitarbeitern dann oftmals überdeutlich aus.
Aus diesem Grund soll eine datenbankgestützte Applikation die erstellten Best-Practices künftig zentral speichern und die Möglichkeit bieten, diese bei Bedarf über eine komfortable Suchfunktion aufzurufen.

2.3 Projektziel

Ziel des Projektes ist es, alle auftretenden Probleme mit deren ausführlicher Beschreibung und einer detaillierten Lösung zentral festzuhalten, um für wiederkehrende bzw. ähnliche Probleme schnellstmöglich eine Lösung zu finden. Das Supportteam soll dadurch das Rad nicht stets neu erfinden müssen und im Arbeitsalltag effizienter arbeiten können.

3 Produktübersicht

3.1 Produktfunktionen

3.1.1 Problemaufnahme

Wird ein Problem gemeldet (per Telefon oder Email), wird dieses in die Datenbank aufgenommen. Dabei werden die Problembeschreibung, der Name der Person, die das Problem gemeldet hat, deren Telefonnummer und ein aussagekräftiger Titel für das Problem gespeichert.

Weiterhin können hier bereits problemspezifische Schlagwörter, die Kategorie des Problems (Hardware- oder Softwareproblem) festgelegt werden.
Dieses neue Problem taucht dann in einer Liste aller offenen Probleme auf und kann im Folgenden bearbeitet werden.

3.1.2 Problembearbeitung

Ist für ein offenes Problem eine Lösung gefunden, wird der betreffende Fall aufgerufen, die detaillierte Lösungsstrategie angegeben und eventuell noch erklärende Grafiken eingebunden. Diese sind dann später in einer PDF-Datei im Bilderverzeichnis zu finden.
Zu jeder Grafik können zusätzlich kurze Hinweise eingefügt werden, die dann in der PDF-Datei unter dem jeweiligen Bild stehen.
Zusätzlich gibt es bei der Problembearbeitung bereits die Möglichkeit, den aktuell bearbeiteten Fall als PDF-Datei zu exportieren oder ihn direkt auszudrucken. Ist die Problembearbeitung beendet, wird der Fall geschlossen und der Best-Practices-Sammlung hinzugefügt.

3.1.3 Best-Practices finden

Alle geschlossenen Fälle bilden zusammen die eigentliche Best-Practices-Sammlung. Um einen bestimmten Eintrag zu finden, kann man sich entweder Hard- und Softwareprobleme getrennt voneinander anzeigen lassen und anschließend sortieren oder man sucht direkt per Suchworteingabe.
In diesem Fall werden die Treffer sofort nach Relevanz sortiert angezeigt.

3.2 Ist-Analyse

3.2.1 Bisheriger Zustand

Derzeit muss sich jeder Mitarbeiter der IT-Abteilung, der sich im Kundensupport eines Falls annimmt, für wiederkehrende Probleme zumeist aufs Neue eine Lösungsstrategie überlegen, da nur für die wenigsten Fälle eine Dokumentation vorhanden ist.
Hinzu kommt, dass die Hälfte des Abteilungspersonals derzeit aus Praktikanten des Arbeitsbereichs Fachinformatik besteht. Daher gilt es häufig, neue Mitarbeiter einzuarbeiten. Gerade für neue Praktikanten ist es jedoch schwierig, zum Teil völlig unbekannte Probleme ohne Dokumentation zu lösen.
Zwar existiert bereits ein Trouble-Ticket-System, über das Kunden Probleme melden können, jedoch dient dieses eben nur der Meldung, nicht der Dokumentation. Somit bildet dies lediglich einen zweiten Kommunikationsweg zur IT-Abteilung, neben dem Telefon.

3.2.2 Arbeitsumgebung

Für die Umsetzung des Projekts wurde ein ideal ausgestatteter Computerarbeitsplatz zur Verfügung gestellt. Dieser verfügte sowohl über einen leistungsfähigen Rechner mit ausreichend viel Arbeitsspeicher als auch über einen Breitband-Internetanschluss.

4 Sollkonzept

4.1 Erfassen der Kundenwünsche

Bei einem Kundengespräch mit dem Auftraggeber wurde erläutert, welchen funktionalen Umfang das Projekt haben soll.

Es soll die Möglichkeit geben, zu einem Rechner, mit dem es ein Problem gibt, Informationen aus der vorhandenen Inventarisierungsdatenbank zu lesen, um bereits im Vorfeld bestimmte Fehler ausschließen zu können.
Eine Suchfunktion, über die man per Suchworteingabe passende Fälle findet, soll ebenso integriert werden wie Funktionen, Best-Practices als PDF-Datei zu exportieren und sie auszudrucken.

4.2 Anforderungen an das Produkt

Die Applikation darf nicht schreibend auf die bestehende Inventarisierungsdatenbank zugreifen. Weiterhin soll für die Anwendung eine eigenständige Access-Datenbank erstellt werden, die im Bedarfsfall auch auf einen Microsoft SQL-Server portierbar sein soll.
Idealerweise kann später via Webinterface auf die Best-Practices-Sammlung zugegriffen werden.
Für die hierfür notwendige Veröffentlichung des Projekts per Java Webstart müssen Bilddateien direkt dem Projekt als Ressource hinzugefügt werden.

4.2.1 Robustheit

4.2.1.1 Fehlende Dateien

Wenn eine (oder beide) Access-Datenbank nicht an dem standardmäßig eingestellten Ort liegt, so soll dies dem User mitgeteilt werden und er die Möglichkeit haben, einen alternativen Speicherort anzugeben.

4.2.1.2 Fehlende Benutzereingaben

Füllt der Benutzer die wichtigsten Felder (Titel, Problembeschreibung, Schlagwörter) nicht aus, sollen diese Felder markiert werden und dem User eine entsprechende Meldung ausgegeben werden.

4.2.1.3 SQL-Injection

Gibt der Benutzer in einem Feld z.B. ein Hochkomma (’) ein, so darf dies nicht zu einem SQL-Fehler führen und muss zwingend abgefangen werden.

4.2.2 Wartbarkeit und Wiederverwendbarkeit

Eine ausführliche Dokumentation sowie aussagekräftige Quelltextkommentare sollen jederzeit Aufschluss über den Aufbau und die Funktionen der Klassen bzw. ihrer Methoden geben.

5 Projektplanung

5.1 Zeit- und Ablaufplanung

Das Projekt wurde im Zeitraum vom 19.11.2007 bis 30.11.2007 mit einer Gesamtdauer von 70 Stunden durchgeführt.
In der nebenstehenden Übersicht sind die einzelnen Projektphase und ihre jeweilige Dauer abgebildet.

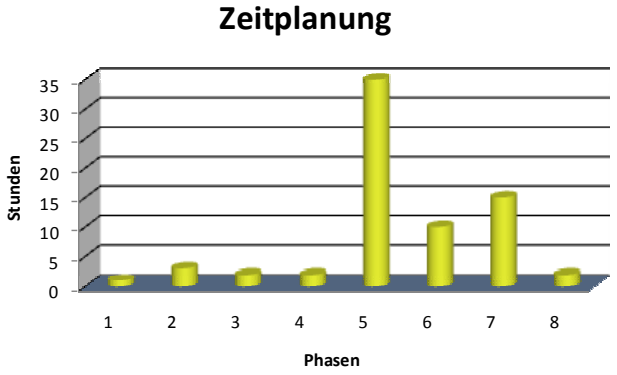


Abbildung 1: Zeitplanung

Erläuterung der Phasen

Phase	Bezeichnung
1	Entgegennahme des Kundenauftrags
2	Informationen einholen
3	Ist-Analyse
4	Soll-Konzept
5	Realisierung
6	Testen
7	Dokumentation
8	Projektübergabe

5.1.1 Projektphasen

Entgegennahme des Kundenauftrags (1 Stunde)

Gespräch mit dem Auftraggeber ..... 1 Stunde

Informationen einholen (3 Stunden)

Kundengespräch..... 1 Stunde
Hintergrundinformationen sammeln (Recherche) ..... 2 Stunden

Ist-Analyse (2 Stunden)

Analyse derzeitiger Verfahrensweise..... 1 Stunde
Analyse vorhandener Inventarisierungsdatenbank ..... 1 Stunde

Soll-Konzept (2 Stunden)

Kundengespräch..... 1,5 Stunden
Planung und Entwurf eines Projektplans..... 1 Stunde

Realisierung (35 Stunden)

Erstellen einer MS-Access Datenbank zur Datenhaltung .....1 Stunde
Entwickeln eines Datenbankkonzeptes ..... 1 Stunde
Entwickeln eines Datenbankfrontends in Java 1.6.....30 Stunden
Entwickeln eines Konzepts für das Datenbankfrontend..... 1,5 Stunden
Erstellen der benötigten Klassen und Methoden ..... 17 Stunden
Entwickeln der grafischen Benutzerschnittstelle..... 8,5 Stunden
Verbinden der Benutzerschnittstelle mit Verarbeitungsschicht des Programms ..... 3 Stunden
Dokumentieren des Quelltextes .....3,5 Stunden

Testen (10 Stunden)

Testen unter realen Bedingungen ..... 2 Stunden
Änderungen und Verbesserungen vornehmen ..... 8 Stunden

Dokumentation (15 Stunden)

Erstellen der Projektdokumentation ..... 15 Stunden

Projektübergabe (2 Stunden)

Präsentation des Projekts vor dem Kunden..... 1 Stunde
Einweisen des Kunden in das Produkt ..... 1 Stunde

5.2 Ressourcenplanung

Das Projekt wird an einem Arbeitsplatzrechner mit Microsoft Windows XP (Service Pack 2) realisiert. Die Projektplanung und –Dokumentation erfolgt mit Microsoft Office 2003 Professional und Microsoft Visio 2003.
Die Datenhaltung wird mit Microsoft Access 2003 realisiert.
Als Entwicklungsumgebung wird Eclipse 3.3 inklusive Omondo als UML-Designer verwendet.
Als Laufzeitumgebung dient die JRE 1.6.0_03.
Es werden zusätzlich das **SwingLabs Swing X** Projekt für die Entwicklung der grafischen Benutzeroberfläche sowie das **iText** Projekt in der Version 2.0.6 für die Entwicklung dynamisch generierter PDF-Dateien in das Gesamtprojekt eingebunden. Hierbei handelt es sich um zwei Open-Source-Projekte.
Weiterhin wird für die Darstellung der PDF-Dateien der Adobe Reader in der Version 8.0.0 verwendet.

5.3 Kostenplanung

Da es sich um ein internes Projekt der IT-Abteilung des Fachbereichs Veterinärmedizin der Freien Universität Berlin handelt und sowohl Hard- als auch Software kostenfrei zur Verfügung stehen, entstehen dem Auftraggeber hier keine realen Kosten.

Stundenlohn	70,00 €
Arbeitsstunden	70
Gesamtnetto	4.900,00 €
MwSt (19%)	931,00 €
Gesamtbrutto	**5.831,00 €**

Abbildung 2: Kostenplanung

Um dennoch zumindest eine kurze und fiktive Kostenplanung durchzuführen, wird hier lediglich die Arbeitskraft des Projektdurchführenden betrachtet und dabei für die Gesamtarbeitszeit von 70 Stunden ein repräsentativer Stundenlohn von 70,- € angesetzt.
Aus diesen Werten ergibt sich folgende Kostenplanung:

6 Projektdurchführung

6.1 Planung und Entwurf

6.1.1 Entgegennahme des Kundenauftrags

6.1.1.1 Kundengespräch

- Termin: 19.11.2007
- Zeitaufwand: 1 Stunde

Bei einem Treffen mit dem Auftraggeber wurde in einem Gespräch das Projektziel grob definiert und wichtige Eckpunkte bestimmt.

6.1.2 Informationen einholen

6.1.2.1 Kundengespräch

- Termin: 19.11.2007
- Zeitaufwand: 1 Stunde

Mit dem Auftraggeber wurden die Rahmenbedingungen für das Projekt definiert und erste detaillierte Anforderungen festgehalten. So sollte das Programm beispielsweise später nicht schreibend auf die vorhandene Inventarisierungsdatenbank zugreifen dürfen.

6.1.2.2 Hintergrundinformationen sammeln, auswerten und zusammenfassen

- Termin: 19.11.2007
- Zeitaufwand: 1,5 Stunden

Die Informationen aus den vorangegangenen Kundengesprächen wurden ausgewertet und es wurde recherchiert, wie sich die verschiedenen Vorgaben realisieren ließen.
Dabei wurde entschieden, zwei Open-Source-Projekte in das Projekt zu integrieren. Einerseits wurde **SwingLabs Swing X** für die Erstellung der grafischen Benutzeroberfläche gewählt und andererseits **iText** in der Version 2.0.6 für die dynamische Erstellung der PDF-Dateien.

6.2.3 Ist-Analyse

6.2.3.1 Analyse der derzeitigen Verfahrensweise

- Termin: 19.11.2007
- Zeitaufwand: 1 Stunde

Das vorhandene Trouble-Ticket-System wurde auf dessen Programmablauf hin untersucht.

6.2.3.2 Analyse vorhandener Inventarisierungsdatenbank

- Termin: 19.11.2007
- Zeitaufwand: 0,5 Stunden

Die Datenbankstruktur der vorhandenen Inventarisierungsdatenbank wurde analysiert und benötigte Tabellen bereits festgehalten.

Nebenstehende Grafik gibt einen groben Überblick über die vorgefundene Datenbankstruktur.

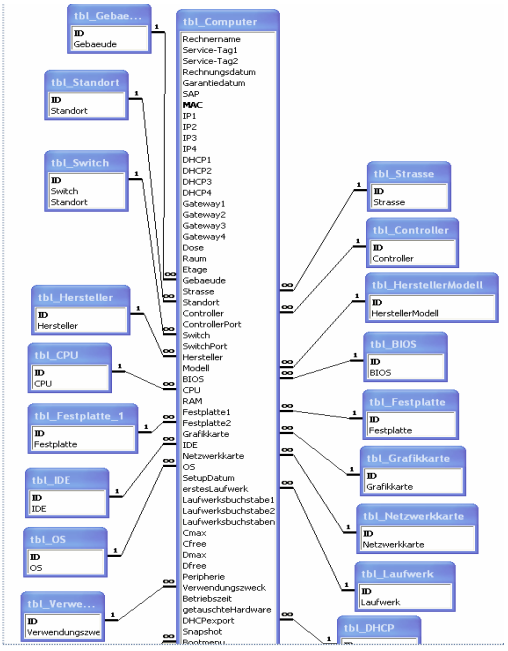


Abbildung 3: Inventarisierungsdatenbank

6.1.4 Soll-Konzept

6.1.4.1 Kundengespräch

- Termin: 19.11.2007
- Zeitaufwand: 1 Stunde

Die Projektziele wurden in einem Gespräch mit dem Auftraggeber besprochen.
Dabei wurde eine bindende und detailierte Beschreibung der zu erfüllenden Leistungen in einem Pflichtenheft festgehalten.

6.1.4.2 Planung und Entwurf eines Projektplans

- Termin: 19.11.2007 / 20.11.2007
- Zeitaufwand: 1 Stunde

Zum besseren Überblick über den zeitlichen Ablauf des Projekts wurde ein Projektablaufplan entwickelt, in dem alle Projektphasen mit ihrer Dauer festgehalten wurden.

6.2 Realisierung

6.2.1 Entwickeln eines Datenbankkonzeptes

- Termin: 20.11.2007
- Zeitaufwand: 1 Stunde

Für die zentrale Datenhaltung der Applikation wurde folgendes Datenbankkonzept entwickelt.
Es bietet die Möglichkeit, aufgetretene Probleme nach ihrer Kategorie (Software oder Hardware) und ihrem Status (offen oder geschlossen) zu unterscheiden.
Weiterhin besteht die Möglichkeit, zu jedem Problem Bilder aufzunehmen, die eine Problemlösung näher beschreiben bzw. visualisieren sollen.

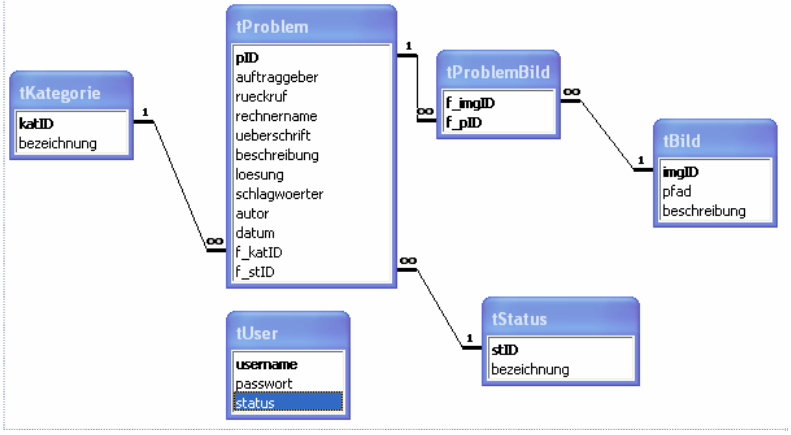


Abbildung 4: TroubleShooter-Datenbank

6.2.2 Entwickeln eines Konzepts für das Datenbankfrontend

- Termin: 20.11.2007
- Zeitaufwand: 1 Stunde

Es wurde ein Konzept für die benutzerfreundliche Gestaltung der grafischen Benutzeroberfläche entwickelt. Dabei standen besonders Techniken für die intuitive Gestaltung des Menüs im Vordergrund. Es wurde entschieden, die Menüs möglichst der gängigen Menügestaltung unter Microsoft Windows XP anzupassen. Dies wurde unter Verwendung des Open-Source-Projekts **SwingLabs Swing X** ermöglicht. Die dazugehörige Dokumentation gab Aufschluss über die zu verwendenden Klassen.

6.2.3 Erstellen der benötigten Klassen und Methoden

- Termin: 20.11.2007 bis 22.11.2007
- Zeitaufwand: 17 Stunden

Für das Projekt wurden insgesamt 14 Klassen entwickelt, die sich auf 4 Packages aufteilen.

6.2.3.1 Builder

Die Klasse Builder wird genutzt, um das Hauptfenster zu initialisieren und das Menü zu erstellen. Weiterhin liefert diese Klasse alle im Programm angezeigten Tabellen.

6.2.3.2 Content

Die Klasse Content repräsentiert alle Formulare, die ausschließlich eine Tabelle enthalten.

6.2.3.3 Corporate

Die Klasse Corporate beinhaltet eine Vielzahl klassenübergreifender Attribute, die für das optische Design der Anwendung notwendig sind.
Sie beinhaltet außerdem eine Methode, mit der Bilddateien byteweise aus dem projekteigenen Archiv gelesen werden.

6.2.3.4 Database

Über die Klasse Database werden alle Datenbankverbindungen aufgebaut und auch wieder geschlossen. Sie bietet außerdem Methoden zum Absetzen eines Updates und zum Holen eines Resultsets.

6.2.3.5 DBLoader

Die Methoden dieser Klasse werden in einem Thread gestartet und dienen dazu, alle im Programm verwendeten Tabellen beim Programmstart zu initialisieren, was die Wartezeit währen des Programmablaufs erheblich verkürzt.

6.2.3.6 Finder

Die Klasse Finder wird genutzt, um bestimmte Fälle anhand von Suchbegriffen zu finden und diese nach Relevanz sortiert in einer Tabelle auszugeben.

6.2.3.7 Handler

Die Klasse Handler behandelt die meisten der auftretenden Ereignisse während des Programmablaufs. Hierbei handelt sich unter anderem um Tastendrücke und Doppelklicks.

6.2.3.8 ImgPrev

Die Klasse ImgPrev stellt ein eigens Steuerelement dar, das aus verschiedenen Steuerelementen zusammengesetzt ist und einen eigenen Eventhandler implementiert hat.
Dieses Steuerelement dient der Bildbetrachtung und –Verwaltung bei der Problembearbeitung.

6.2.3.9 LoginDialog

Die Klasse LoginDialog ist der Haupteinstiegspunkt des Programms. Hier ist die Main-Methode implementiert.
Die Klasse realisiert die Benutzeranmeldung.
Hierbei werden sowohl Benutzername als auch Passwort zunächst über ein Objekt der Klasse MyMD5 verschlüsselt und dann über ein Objekt der Klasse Database mit den in der Datenbank eingetragenen Werten verglichen.

6.2.3.10 MainWindow

Die Klasse MainWindow stellt das Hauptfenster des Programms dar. Es wird zum Start des Programms über ein Objekt der Klasse Builder initialisiert.

6.2.3.11 MyMD5

Die Klasse MyMD5 bietet eine Methode an, die einen übergebenen String per MD5-Verschlüsselung verschlüsselt. Ein Objekt dieser Klasse wird von der Klasse LoginDialog verwendet.

6.2.3.12 PanelCloseIncident

Die Klasse PanelCloseIncident stellt das Formular zum Bearbeiten / Schließen eines Vorfalls dar.

6.2.3.13 PanelNewIncident

Die Klasse PanelNewIncident stellt das Formular zum Erstellen / Melden eines Vorfalls dar.

6.2.3.14 PDFCreator

Die Klasse PDFCreator wird verwendet, um aus den zu einem bestimmten Problem in der Datenbank hinterlegten Daten dynamisch eine PDF-Datei zu erstellen.
Dieses PDF-Dokument wird im Corporate Design der Freien Universität Berlin erstellt.
Es finden sich sowohl Problembeschreibung als auch Problemlösungsstrategie und evtl. dazugehörige Grafiken in diesem Dokument.

6.2.4 Entwickeln der grafischen Benutzerschnittstelle

- Termin: 22.11.2007 bis 23.11.2007
- Zeitaufwand: 8 Stunden

Bei der Entwicklung der grafischen Benutzeroberfläche wurde auf eine intuitive Menügestaltung geachtet. Das Hauptmenü wurde unter Verwendung des Open-Source-Projektes **SwingLabs Swing X** stark an die Menüs unter Microsoft Windows XP angelehnt.
Ausklappbare Panels mit aussagekräftigen Titeln sollen dem Benutzer die einfache Navigation durch das Programm ermöglichen.
Es wurden bewusst so wenige und so intuitive Steuerelemente wie möglich verwendet, um dem Benutzer das Arbeiten mit dem Programm so einfach wie möglich zu machen.
Für die Eingabe von Schlagwörtern bei der Aufnahme / Bearbeitung eines Vorfalls wurde ein eigener Algorithmus entwickelt, der es dem User ermöglicht, Schlagwörter per Mausklick einzufügen.

6.2.5 Verbinden der Benutzerschnittstelle mit Verarbeitungsschicht des Programms

- Termin: 23.11.2007 bis 26.11.2007
- Zeitaufwand: 3 Stunden

Bei dem Verbinden der Benutzerschnittstelle mit der Verarbeitungsschicht des Programms wurden den Steuerelementen Eventhandler hinzugefügt, die bei bestimmten Ereignissen bestimmte Methoden aufrufen.

6.2.6 Dokumentieren des Quelltextes

- Termin: 26.11.2007
- Zeitaufwand: 3,5 Stunden

Während der Realisierung der Klassen und Methoden wurden nur die wichtigsten Programmteile für den Programmierer dokumentiert. In diesen Schritt wurden Methoden näher beschrieben und Kommentare für die Javadocs eingefügt.

6.3 Testphase

6.3.1 Testen unter realen Bedingungen

- Termin: 26.11.2007
- Zeitaufwand: 3 Stunden

Das Programm wurde zu Testzwecken im Kundensupport verwendet. Es wurden verschiedene Problemberichte aufgenommen und bearbeitet. Dabei wurde das Programm auf unterschiedlichen Rechnern ausgeführt.
Es stellte sich heraus, dass es Probleme mit den PDF-Dokumenten gab, wenn ein erstelltes temporäres PDF-Dokument auf dem bearbeitenden Rechner noch nicht existierte. Wollte man dann nämlich einen Fall als PDF-Dokument exportieren, so wurde die Datei ohne die Endung .pdf erstellt.
Weiterhin gab es Probleme mit Bildern, die dem PDF-Dokument hinzugefügt werden sollten.
Während der Realisierungsphase war immer von Bildern der Größe 800 X 600 Pixel ausgegangen worden und die Bilder wurden dementsprechend skaliert. Nun erschienen manche Bilder stark verzerrt.

6.3.2 Änderungen und Verbesserungen vornehmen

- Termin: 26.11.2007 bis 27.11.2007
- Zeitaufwand: 9,5 Stunden

Die Fehler, die bei der Erstellung des PDF-Dokumentes auftraten, wurden behoben.
Weiterhin wünschte der Auftraggeber unbedingt eine benutzerabhängige Programmsteuerung. So mussten ein Login-Dialog implementiert und das Menü benutzerabhängig gestaltet werden. Die Rolle des Benutzers war an verschiednen Stellen des Programms relevant, weshalb an einigen Stellen zusätzlich kleine Änderungen vorgenommen werden mussten.

6.4 Dokumentation

6.4.1 Erstellen der Produktdokumentation

- Termin: 27.11.2007 bis 29.11.2007
- Zeitaufwand: 15,5 Stunden

Im Verlauf des Projektes wurden einzelne Teilschritte stichpunktartig dokumentiert. Aus diesen Stichpunkten und erstellten Grafiken entstand die Produktdokumentation.
Die einzelnen Klassen und Methoden wurden mittels Javadoc-Kommentaren dokumentiert, was es dem Entwickler schneller ermöglicht, Zusammenhänge und Funktionen schneller zu erfassen.

6.5 Übergabe

6.5.1 Präsentation des Projekts vor dem Kunden

- Termin: 29.11.2007
- Zeitaufwand: 1 Stunde

Dem Kunden wurde das Projekt mit allen Funktionen vorgeführt. Es wurden Beispielfälle aufgenommen, um alle Funktionen zu demonstrieren.
Anschließend wurde erläutert, welche Funktionen des Programms zur Verfügung stehen, wenn ein Gast sich anmeldet.

6.5.2 Einweisen des Kunden in das Produkt

- Termin: 29.11.2007
- Zeitaufwand: 0,5 Stunden

Unter Anweisung des Entwicklers sollte der Kunde nach der Präsentation selbst einen Beispielfall aufnehmen, um sich die Funktionalität des Programms besser einzuprägen.

7 Projektbewertung

7.1 Soll / Ist – Vergleich

Während der Projektdurchführung hat sich nicht alles exakt so wie in der Projektplanung skizziert entwickelt.
Alle zeitlichen Abweichungen sind im folgenden Soll / Ist – Vergleich dargestellt.

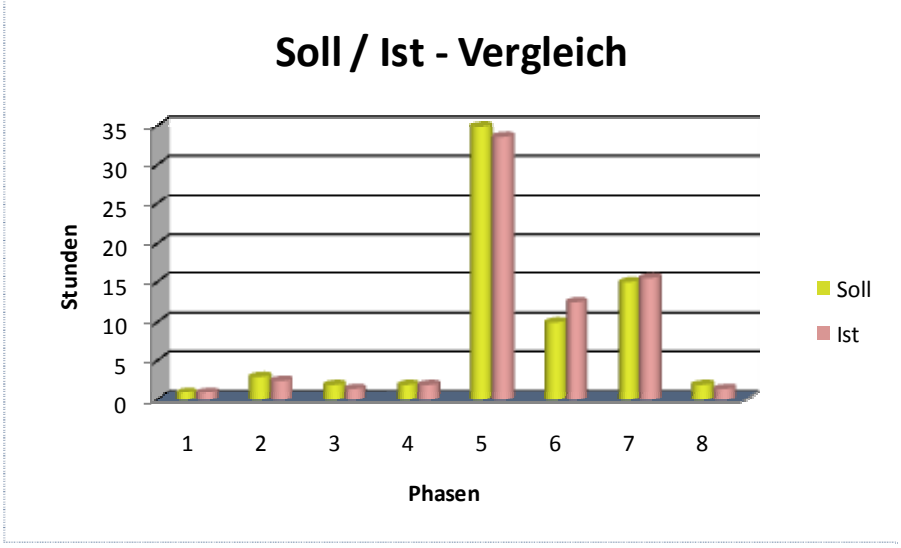


Abbildung 5: Soll / Ist – Vergleich

Phase	Soll (in Std.)	Ist (in Std.)	Differenz (in Std.)
1 Entgegennahme des Kundenauftrags	1	1	**+/- 0**
2 Informationen einholen	3	2,5	**- 0,5**
3 Ist-Analyse	2	1,5	**- 0,5**
4 Soll-Konzept	2	2	**+/- 0**
5 Realisierung	35	33,5	**- 1,5**
6 Testen	10	12,5	**+ 2,5**
7 Dokumentation	15	15,5	**+ 0,5**
8 Projektübergabe	2	1,5	**- 0,5**
Gesamt	**70**	**70**	**+/- 0**

7.1.1 Erläuterung der Zeitabweichungen

7.1.1.1 Informationen einholen

Da die Recherchearbeiten schnelle vorangingen als geplant, konnten hier 30 Minuten gegenüber der Planung eingespart werden.

7.1.1.2 Ist-Analyse

Aufgrund des geringeren Umfangs der bisher verwendeten Verfahrensweise konnten auch hier 30 Minuten eingespart werden.

7.1.1.3 Realisierung

Dank der ausführlichen Dokumentationen der verwendeten Open-Source-Projekte **iText** und **SwingLabs Swing X** konnten in den zeitintensivsten Bereichen (Erstellung der grafischen Benutzeroberfläche und Gestaltung dynamischer PDF-Dokumente) insgesamt 90 Minuten gegenüber der Planung eingespart werden.

7.1.1.4 Testen

Da der Auftraggeber die benutzergesteuerte Programmgestaltung doch als zwingend erforderlich empfand, mussten kleine bis mittelgroße Änderungen vorgenommen werden.
So musste ein Login-Dialog sowie eine benutzerabhängige Menügestaltung implementiert werden. Daher kam es in dieser Phase zu einem Zeitverzug von 2,5 Stunden.

7.1.1.5 Dokumentation

Im Zusammenhang mit den erforderlichen Programmänderungen entstand ebenfalls ein Mehraufwand bei der Dokumentation des Programms. Dieser betrug 30 Minuten.

7.1.1.6 Projektübergabe

Die Produktpräsentation sowie die Einweisung des Kunden in das Produkt liefen problemlos ab und somit konnten hier 30 Minuten gegenüber der Planung eingespart werden.

7.2 Wirtschaftlichkeitsbetrachtung

Grundsätzlich kann man sagen, dass das Projekt umso wirtschaftlicher arbeitet, je länger dessen Laufzeit ist. Je mehr Vorfälle in die Datenbank aufgenommen werden, desto weniger Zeit muss später für jeden einzelnen Fall aufgewendet werden.
In der nebenstehenden Darstellung wird davon ausgegangen, dass sich die aufzuwendende Arbeitszeit im Kundensupport halbiert.

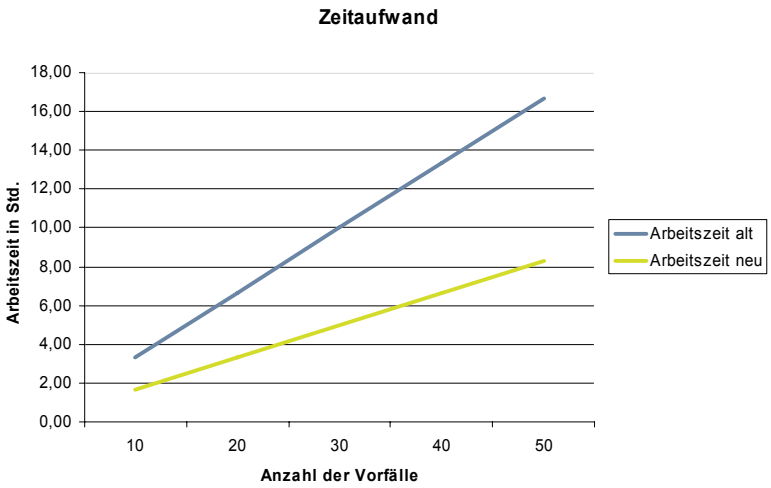


Abbildung 6: Wirtschaftlichkeitsbetrachtung

7.3 Fazit

Das Projekt konnte erfolgreich abgeschlossen werden, da alle Anforderungen an das Produkt erfüllt werden konnten. Der zeitliche Rahmen wurde eingehalten, auch wenn dies nur möglich war, weil Zeitüberschreitungen durch Zeiteinsparungen ausgeglichen werden konnten.
An diesem Projekt ist deutlich geworden, wie wichtig eine detaillierte Projektplanung für die Einhaltung des zeitlichen Rahmens ist.

7.4 Ausblick

Die Anwendung wird künftig per Java Webstart am Fachbereich eingesetzt. Somit sollen Kunden in Zukunft zunächst selbst nach Best-Practices schauen können, bevor sie den Kundensupport der IT-Abteilung anrufen.
Durch dieses Projekt wird die IT-Abteilung im Bereich des Kundensupports deutlich entlastet werden.

7.5 Persönliche Bewertung

Durch die Arbeit an diesem Projekt konnte ich sowohl meine Java-Kenntnisse erweitern und vertiefen als auch Kenntnisse im Umgang mit **SwingLabs Swing X** und **iText** erwerben.
Diese erworbenen Kenntnisse im Umgang mit diesen Bibliotheken lassen sich sehr gut in neue Projekte einbringen.

8 Quellen

8.1 Online-Dokumentationen

8.1.1 J2SE 1.6 API

http://java.sun.com/javase/6/docs/api/

8.1.2 SwingLabs Swing X API

http://download.java.net/javadesktop/swinglabs/releases/0.8/docs/api/index.html

8.1.3 iText API

http://itext.ugent.be/library/api/

9 Anhang

9.1 Pflichtenheft

9.1.1 Projektziel

Es sollen alle dem Support gemeldeten Probleme mit deren ausführlicher Beschreibung und einer detaillierten Lösung zentral festgehalten werden, um für wiederkehrende bzw. ähnliche Probleme schnellstmöglich eine Lösung zu finden.

9.1.1.1 Musskriterien

Das Programm wird in der Programmiersprache Java realisiert und hält die Daten in einer Microsoft Access Datenbank. Weiterhin muss das Programm folgende Funktionen erfüllen:

- Export einer Best-Practice als PDF-Dokument
- Einbinden von Grafiken zu einem Vorfall; Grafiken sollen ins PDF-Dokument integriert werden
- PDF-Dokument wird im Corporate Design der Freien Universität Berlin gehalten
- Komfortable Suchfunktion erleichtert das Finden von Best-Practices
- Intuitive Gestaltung der grafischen Benutzerschnittstelle

9.1.1.2 Sollkriterien

Das Programm schließt benutzerabhängig alle Bearbeitungsfunktionen sowie Informationsfunktionen zu Arbeitsplatzrechnern aus. Dem allgemeinen Benutzer stehen somit nur die Funktionen des Meldens eines Vorfalls sowie das Durchsuchen der Best-Practices-Sammlung zur Verfügung.
Alle weiteren Funktionen stehen nur einem Administrator zur Verfügung.

9.1.1.3 Kannkriterien

Das Programm wird per Java Webstart im Intranet des Fachbereichs Veterinärmedizin der Freien Universität Berlin veröffentlicht.

9.1.2 Produkteinsatz

9.1.2.1 Anwendungsbereiche

Die Software wird im Kundensupport der IT-Abteilung des Fachbereichs Veterinärmedizin der Freien Universität Berlin eingesetzt.

9.1.2.2 Zielgruppen

Zielgruppe sollen zunächst nur Mitarbeiter der IT-Abteilung sein. Wünschenswert ist jedoch auch, dass letzlich auch weitere PC-Nutzer am Fachbereich Veterinärmedizin zur Zielgruppe zählen.

9.1.3 Produktübersicht

Die IT-Abteilung des Fachbereichs Veterinärmedizin der Freien Universität Berlin hat unter anderem die Betreuung von Mitarbeitern in IT-Fragen zur Aufgabe. Zur Unterstützung des Supportteams soll eine Applikation entwickelt werden, mit der die Verwaltung einer Best-Practices-Sammlung realisiert werden kann.
Bisher werden beim Kundensupport Problemlösungsstrategien so gut wie nie dokumentiert, was dazu führt, dass für gleiche oder ähnliche Probleme stets aufs Neue überlegt werden muss, wie man ein bestimmtes Problem angeht.
Daher soll künftig eine datenbankgestützte Applikation die erstellten Best-Practices zentral speichern und die Möglichkeit bieten, diese bei Bedarf durch eine komfortable Suchfunktion aufzurufen.

9.1.4 Produktfunktionen

9.1.4.1 Problemaufnahme

Dem Support gemeldete Vorfälle werden in die Datenbank aufgenommen und gelten so lange als noch nicht abgeschlossen, wie niemand sie bearbeitet und geschlossen hat.

9.1.4.2 Problembearbeitung

Für alle offenen Fälle gilt es, eine geeignete Problemlösungsstrategie zu entwickeln und diese als Best-Practice in der Datenbank abzulegen, damit sie für künftige ähnliche Probleme herangezogen werden kann.

9.1.4.3 Best-Practices finden

Aus dem Pool der erstellten Best-Practices muss es möglich sein, über eine Suchfunktion den Fall zu ermitteln, der am ehesten dem aktuellen Fall entspricht.

9.1.4.4 Erzeugen von PDF-Dokumenten

Während der Problembearbeitung muss bereits die Funktion gegeben sein, einen Vorfall als PDF-Dokument zu exportieren, beziehungsweise dies zu drucken.
Es muss möglich sein, erläuternde Grafiken in dieses PDF-Dokument zu integrieren und diese Grafiken kurz zu beschriften.

9.1.5 Produktdaten

Für jeden Vorfall müssen folgende Daten festgehalten werden:

- DNS-Name des Rechners, für den ein Vorfall gemeldet wurde
- Name desjenigen, der einen Vorfall gemeldet hat
- Rufnummer desjenigen, der einen Vorfall gemeldet hat
- Titel / Überschrift des Vorfalls
- Beschreibung des Vorfalls
- Beschreibung der Lösung des Problems
- Erklärende Grafik(en) und dazugehörige Kurzbeschreibung(en)

9.1.6 Produktleistungen

Gerade im Bereich „Best-Practices finden“ gilt es, zuverlässig relevante Daten von nicht relevanten zu trennen und zu sortieren. Relevante Vorfälle müssen über geeignete Funktion schnell zu finden und zu öffnen sein.
Die Ladezeiten während der Programmausführung sind zu minimieren.

9.1.7 Qualitätsanforderungen

9.1.8 Benutzeroberfläche

Die grafische Benutzeroberfläche ist intuitiv und möglichst übersichtlich zu halten. Für alle Produktdaten sind dementsprechende Eingabefelder vorzusehen.
Der normale Benutzer hat, so die Benutzerabhängigkeiten implementiert werden, keinen Zugriff auf die „offenen Fälle“ und kann auch nicht die Informationen zu den einzelnen Arbeitsplatzrechnern einsehen. Diese Rechte bleiben allein den Administratoren vorbehalten.

9.1.9 Nichtfunktionale Anforderungen

9.1.9.1 Sicherheitsanforderungen

Das zu erstellende Programm hat keinerlei schreibenden Zugriff auf die Inventarisierungsdatenbank der IT-Abteilung.

9.1.9.2 Plattformabhängigkeiten

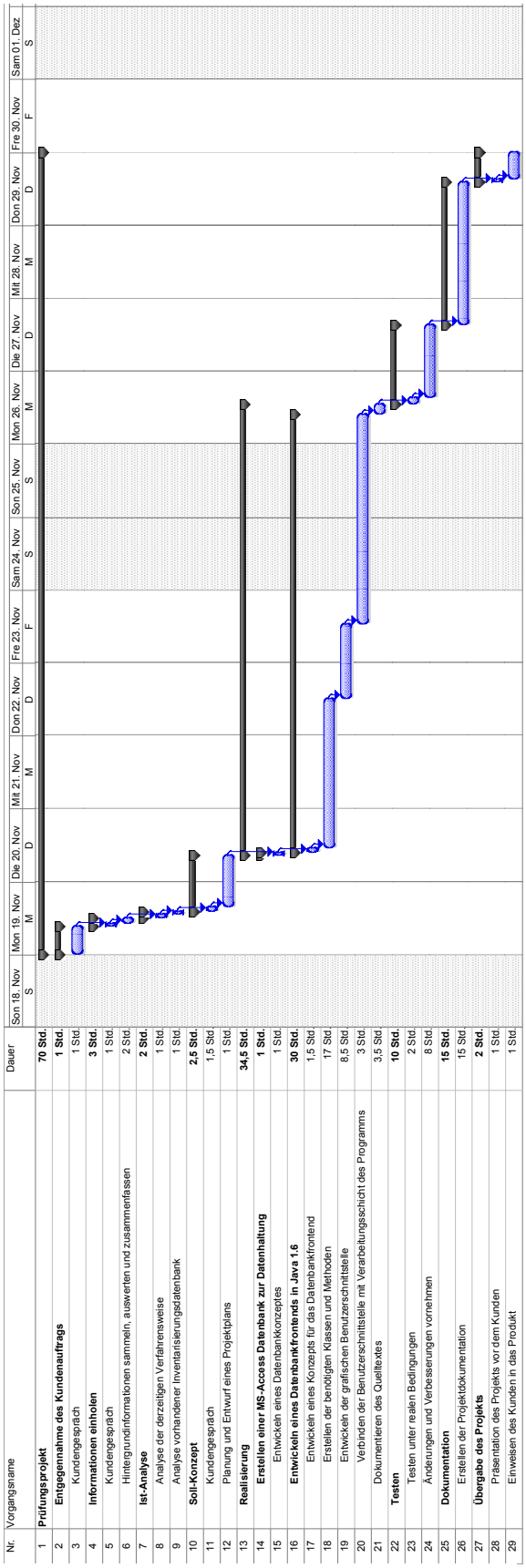
Unter der Voraussetzung einer geeigneten Java-Laufzeitumgebung ist das Projekt plattformunabhängig zu betreiben.

9.1.10 Technische Produktumgebung

9.1.10.1 Software

Betriebssystem	Microsoft Windows XP SP2
Entwicklungsumgebung	Eclipse 3.3
UML-Designer	Omondo
Java GUI-Bibliothek	SwingLabs Swing X
Java PDF-Bibliothek	iText 2.0.6
J2RE	(Java 2 Runtime Environment) Version 1.6.0_03

9.2 Projektablaufplan



Nr.	Vorgangsname	Dauer
1	**Prüfungsprojekt**	**70 Std.**
2	**Entgegennahme des Kundenauftrags**	**1 Std.**
3	Kundengespräch	1 Std.
4	**Informationen einholen**	**3 Std.**
5	Kundengespräch	1 Std.
6	Hintergrundinformationen sammeln, auswerten und zusammenfassen	2 Std.
7	**Ist-Analyse**	**2 Std.**
8	Analyse der derzeitigen Verfahrensweise	1 Std.
9	Analyse vorhandener Inventarisierungsdatenbank	1 Std.
10	**Soll-Konzept**	**2,5 Std.**
11	Kundengespräch	1,5 Std.
12	Planung und Entwurf eines Projektplans	1 Std.
13	**Realisierung**	**34,5 Std.**
14	**Erstellen einer MS-Access Datenbank zur Datenhaltung**	**1 Std.**
15	Entwickeln eines Datenbankkonzeptes	1 Std.
16	**Entwickeln eines Datenbankfrontends in Java 1.6**	**30 Std.**
17	Entwickeln eines Konzepts für das Datenbankfrontend	1,5 Std.
18	Erstellen der benötigten Klassen und Methoden	17 Std.
19	Entwickeln der grafischen Benutzerschnittstelle	8,5 Std.
20	Verbinden der Benutzerschnittstelle mit Verarbeitungsschicht des Programms	3 Std.
21	Dokumentieren des Quelltextes	3,5 Std.
22	**Testen**	**10 Std.**
23	Testen unter realen Bedingungen	2 Std.
24	Änderungen und Verbesserungen vornehmen	8 Std.
25	**Dokumentation**	**15 Std.**
26	Erstellen der Projektdokumentation	15 Std.
27	**Übergabe des Projekts**	**2 Std.**
28	Präsentation des Projekts vor dem Kunden	1 Std.
29	Einweisen des Kunden in das Produkt	1 Std.

Abbildung 7: Projektablaufplan

9.3 UML-Klassenübersicht

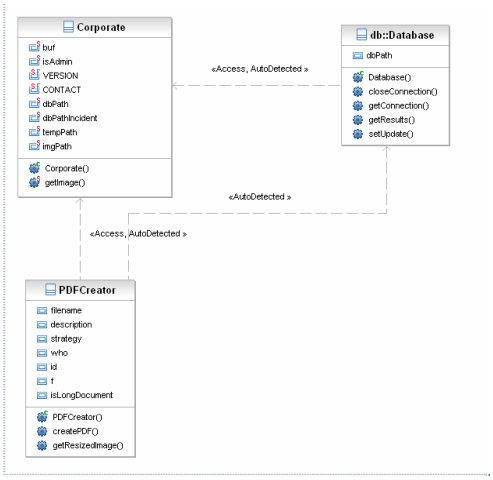


Abbildung 8: PDFCreator

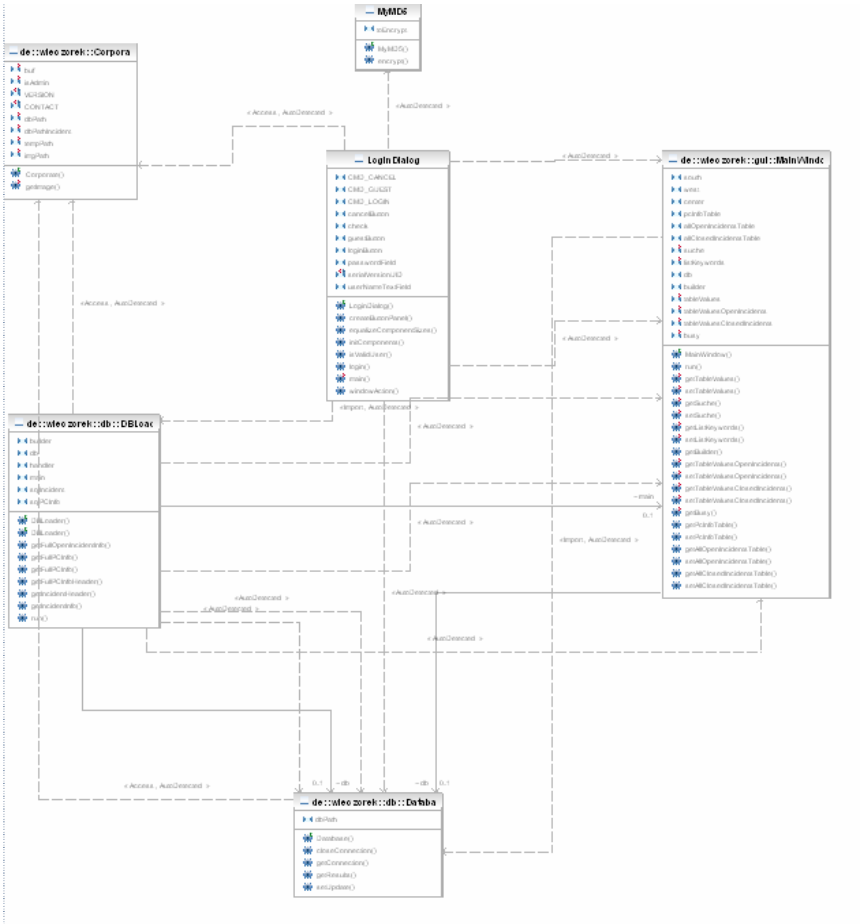


Abbildung 9: LoginDialog

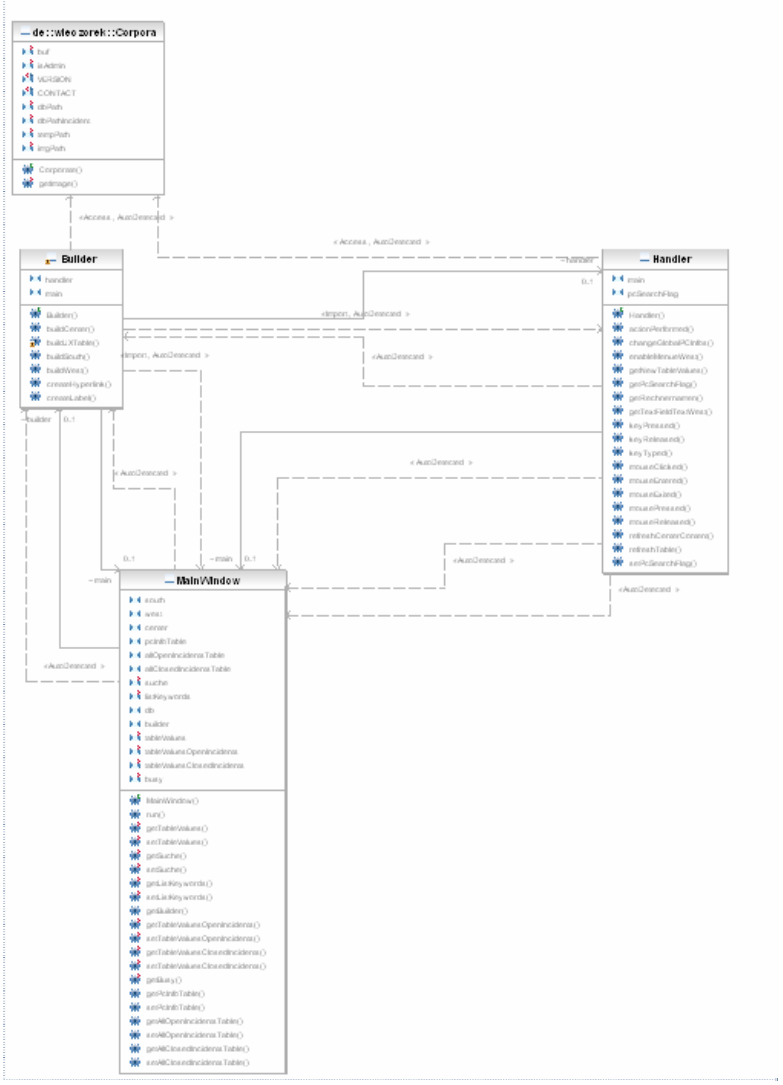


Abbildung 10: Builder

Abbildung 11: Gesamtübersicht

9.4 GUI-Übersicht

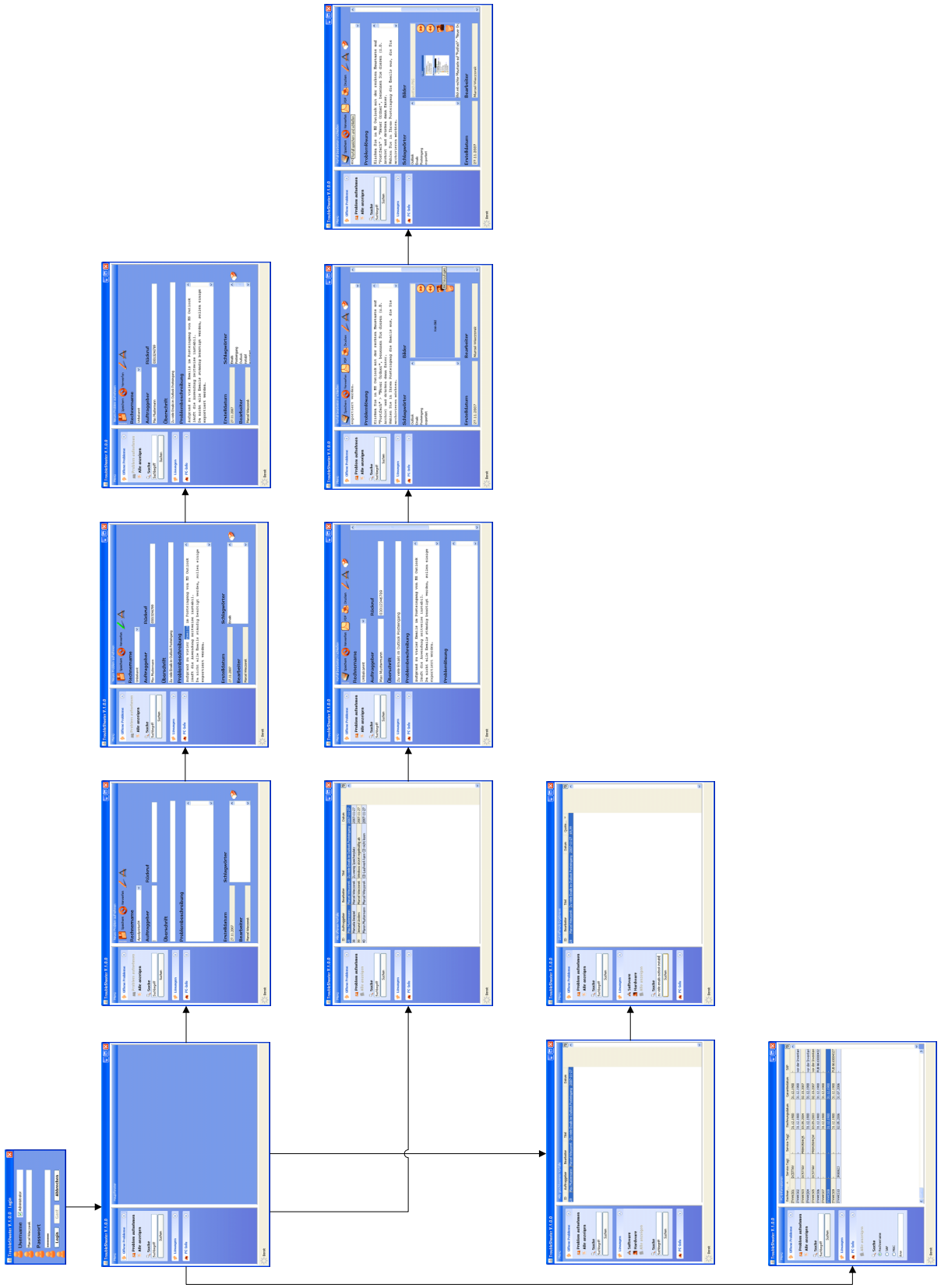


Abbildung 12: GUI-Übersicht, Administrator

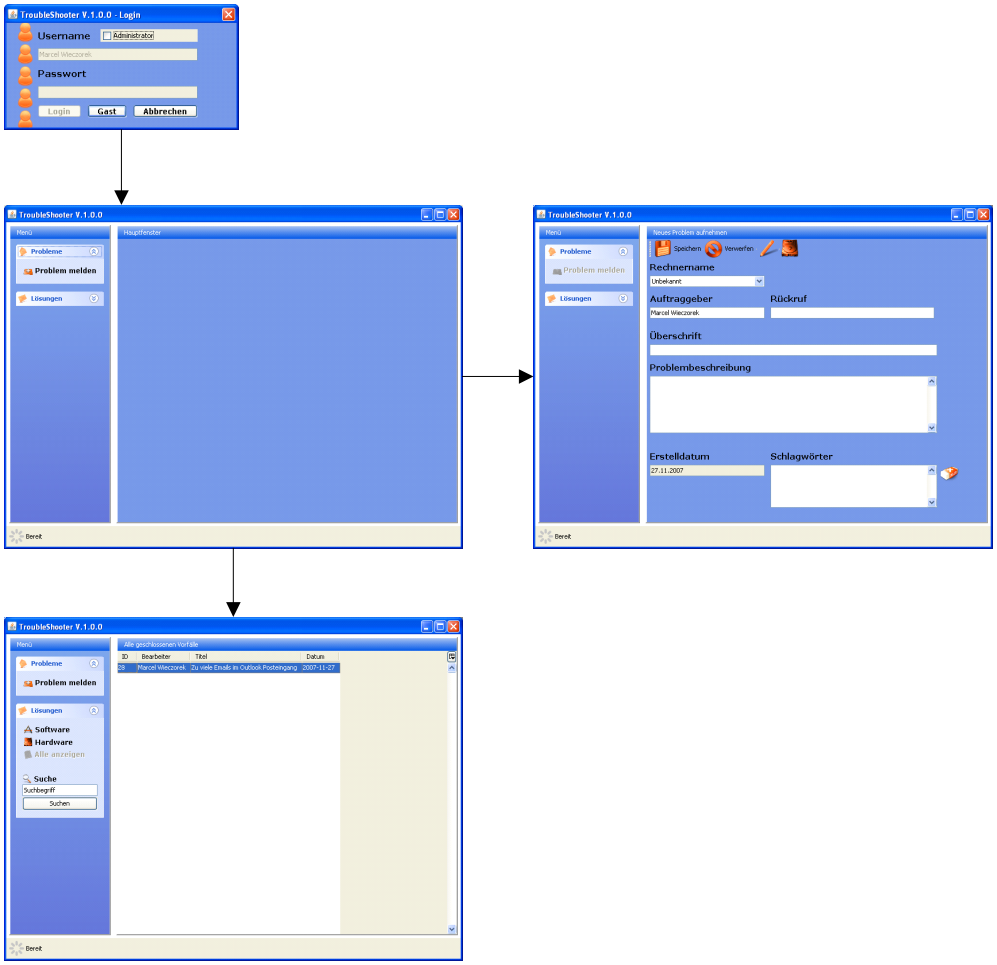


Abbildung 13: GUI-Übersicht, Gast

9.5 Javadocs

9.5.1 Package-Übersicht

TroubleShooter

Packages	
de.wieczorek	
de.wieczorek.db	
de.wieczorek.gui	
de.wieczorek.security	

9.5.2 Package: de.wieczorek

9.5.2.1 Corporate

de.wieczorek

Class Corporate

```
java.lang.Object
  └ de.wieczorek.Corporate
```

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

This class contains lots of static fields that are used from all the other classes.

Author:

Marcel Wieczorek

Field Summary	
static java.awt.Font	BOLD12
static java.awt.Font	BOLD15
static java.lang.String	CONTACT
static java.awt.Color	DARK_BLUE
static java.lang.String	dbPath
static java.lang.String	dbPathIncident
static java.lang.String	FU_LOGO

static java.awt.Color	**HEAVY_ERROR**
static javax.swing.ImageIcon	**IMG_ADD_PROBLEM**
static javax.swing.ImageIcon	**IMG_CLOSED_LETTER**
static javax.swing.ImageIcon	**IMG_COMPUTER**
static javax.swing.ImageIcon	**IMG_HARDWARE_PROBLEM**
static javax.swing.ImageIcon	**IMG_OPEN_LETTER**
static javax.swing.ImageIcon	**IMG_SAVE**
static javax.swing.ImageIcon	**IMG_SAVE_AS**
static javax.swing.ImageIcon	**IMG_SEARCH**
static javax.swing.ImageIcon	**IMG_SHOW_ALL**
static javax.swing.ImageIcon	**IMG_SOFTWARE_PROBLEM**
static javax.swing.ImageIcon	**IMG_USER**
static javax.swing.ImageIcon	**IMG_USER_ADD**
static javax.swing.ImageIcon	**IMG_USER_EDIT**
static javax.swing.ImageIcon	**IMG_USER_OK**
static javax.swing.ImageIcon	**IMG_USER_REMOVE**
static javax.swing.ImageIcon	**IMG32_ADD_IMAGE**
static javax.swing.ImageIcon	**IMG32_ADOBE**
static javax.swing.ImageIcon	**IMG32_ERROR**
static javax.swing.ImageIcon	**IMG32_FAVS**

static javax.swing.ImageIcon	**IMG32_HARDWARE_PROBLEM**
static javax.swing.ImageIcon	**IMG32_IMAGE_FORWARD**
static javax.swing.ImageIcon	**IMG32_IMAGE_REWIND**
static javax.swing.ImageIcon	**IMG32_INFO**
static javax.swing.ImageIcon	**IMG32_KEYWORD_NO**
static javax.swing.ImageIcon	**IMG32_KEYWORD_REMOVE**
static javax.swing.ImageIcon	**IMG32_KEYWORD_YES**
static javax.swing.ImageIcon	**IMG32_OK**
static javax.swing.ImageIcon	**IMG32_PRINTER**
static javax.swing.ImageIcon	**IMG32_REFRESH**
static javax.swing.ImageIcon	**IMG32_RESET_ALL**
static javax.swing.ImageIcon	**IMG32_SAVE**
static javax.swing.ImageIcon	**IMG32_SAVE_AS**
static javax.swing.ImageIcon	**IMG32_SOFTWARE_PROBLEM**
static javax.swing.ImageIcon	**IMG32_TRASH**
static javax.swing.ImageIcon	**IMG32_USER**
static javax.swing.ImageIcon	**IMG32_USER_ADD**
static javax.swing.ImageIcon	**IMG32_USER_EDIT**
static javax.swing.ImageIcon	**IMG32_USER_OK**

static javax.swing.ImageIcon	**IMG32_USER_REMOVE**
static java.lang.String	**imgPath**
static boolean	**isAdmin**
static java.awt.Color	**LIGHT_BLUE**
static java.awt.Color	**LIGHT_ERROR**
static java.awt.Color	**ORANGE**
static java.awt.Font	**PLAIN12**
static java.lang.String	**tempPath**
static java.lang.String	**VERSION**

Constructor Summary

Corporate()

Method Summary

static java.awt.Image	**getImage**(java.lang.Class<Corporate> relativeClass, java.lang.String relativeFilename)
static byte[]	**getImage**(java.lang.String relativeFilename)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

isAdmin

public static boolean **isAdmin**

PLAIN12
`public static final java.awt.Font PLAIN12`

BOLD12
`public static final java.awt.Font BOLD12`

BOLD15
`public static final java.awt.Font BOLD15`

DARK_BLUE
`public static final java.awt.Color DARK_BLUE`

LIGHT_BLUE
`public static final java.awt.Color LIGHT_BLUE`

ORANGE
`public static final java.awt.Color ORANGE`

HEAVY_ERROR
`public static final java.awt.Color HEAVY_ERROR`

LIGHT_ERROR
`public static final java.awt.Color LIGHT_ERROR`

VERSION
`public static final java.lang.String VERSION`
See Also:
Constant Field Values

CONTACT
`public static final java.lang.String CONTACT`
See Also:
Constant Field Values

dbPath
`public static java.lang.String dbPath`

dbPathIncident
`public static java.lang.String dbPathIncident`

tempPath
`public static java.lang.String tempPath`

imgPath
```
public static java.lang.String imgPath
```

IMG_ADD_PROBLEM
```
public static final javax.swing.ImageIcon IMG_ADD_PROBLEM
```

IMG_CLOSED_LETTER
```
public static final javax.swing.ImageIcon IMG_CLOSED_LETTER
```

IMG_COMPUTER
```
public static final javax.swing.ImageIcon IMG_COMPUTER
```

IMG_HARDWARE_PROBLEM
```
public static final javax.swing.ImageIcon IMG_HARDWARE_PROBLEM
```

IMG_OPEN_LETTER
```
public static final javax.swing.ImageIcon IMG_OPEN_LETTER
```

IMG_SAVE
```
public static final javax.swing.ImageIcon IMG_SAVE
```

IMG_SAVE_AS
```
public static final javax.swing.ImageIcon IMG_SAVE_AS
```

IMG_SEARCH
```
public static final javax.swing.ImageIcon IMG_SEARCH
```

IMG_SHOW_ALL
```
public static final javax.swing.ImageIcon IMG_SHOW_ALL
```

IMG_SOFTWARE_PROBLEM
```
public static final javax.swing.ImageIcon IMG_SOFTWARE_PROBLEM
```

IMG_USER
```
public static final javax.swing.ImageIcon IMG_USER
```

IMG_USER_ADD
```
public static final javax.swing.ImageIcon IMG_USER_ADD
```

IMG_USER_EDIT
```
public static final javax.swing.ImageIcon IMG_USER_EDIT
```

IMG_USER_OK
```
public static final javax.swing.ImageIcon IMG_USER_OK
```

IMG_USER_REMOVE
```
public static final javax.swing.ImageIcon IMG_USER_REMOVE
```

IMG32_ADD_IMAGE
```
public static final javax.swing.ImageIcon IMG32_ADD_IMAGE
```

IMG32_IMAGE_FORWARD
```
public static final javax.swing.ImageIcon IMG32_IMAGE_FORWARD
```

IMG32_IMAGE_REWIND
```
public static final javax.swing.ImageIcon IMG32_IMAGE_REWIND
```

IMG32_TRASH
```
public static final javax.swing.ImageIcon IMG32_TRASH
```

IMG32_ADOBE
```
public static final javax.swing.ImageIcon IMG32_ADOBE
```

IMG32_RESET_ALL
```
public static final javax.swing.ImageIcon IMG32_RESET_ALL
```

IMG32_ERROR
```
public static final javax.swing.ImageIcon IMG32_ERROR
```

IMG32_FAVS
```
public static final javax.swing.ImageIcon IMG32_FAVS
```

IMG32_HARDWARE_PROBLEM
```
public static final javax.swing.ImageIcon IMG32_HARDWARE_PROBLEM
```

IMG32_INFO
```
public static final javax.swing.ImageIcon IMG32_INFO
```

IMG32_KEYWORD_NO
```
public static final javax.swing.ImageIcon IMG32_KEYWORD_NO
```

IMG32_KEYWORD_REMOVE
```
public static final javax.swing.ImageIcon IMG32_KEYWORD_REMOVE
```

IMG32_KEYWORD_YES

```
public static final javax.swing.ImageIcon IMG32_KEYWORD_YES
```

IMG32_OK

```
public static final javax.swing.ImageIcon IMG32_OK
```

IMG32_PRINTER

```
public static final javax.swing.ImageIcon IMG32_PRINTER
```

IMG32_REFRESH

```
public static final javax.swing.ImageIcon IMG32_REFRESH
```

IMG32_SAVE

```
public static final javax.swing.ImageIcon IMG32_SAVE
```

IMG32_SAVE_AS

```
public static final javax.swing.ImageIcon IMG32_SAVE_AS
```

IMG32_SOFTWARE_PROBLEM

```
public static final javax.swing.ImageIcon IMG32_SOFTWARE_PROBLEM
```

IMG32_USER

```
public static final javax.swing.ImageIcon IMG32_USER
```

IMG32_USER_ADD

```
public static final javax.swing.ImageIcon IMG32_USER_ADD
```

IMG32_USER_EDIT

```
public static final javax.swing.ImageIcon IMG32_USER_EDIT
```

IMG32_USER_OK

```
public static final javax.swing.ImageIcon IMG32_USER_OK
```

IMG32_USER_REMOVE

```
public static final javax.swing.ImageIcon IMG32_USER_REMOVE
```

FU_LOGO

```
public static final java.lang.String FU_LOGO
```

See Also:

Constant Field Values

Constructor Detail

Corporate

```
public Corporate()
```

Method Detail

getImage

```
public static java.awt.Image
getImage(java.lang.Class<Corporate> relativeClass,
                                    java.lang.String relativeFilename)
```

---

getImage

```
public static byte[] getImage(java.lang.String relativeFilename)
```

9.5.2.2 PDFCreator

de.wieczorek

Class PDFCreator

```
java.lang.Object
  └ de.wieczorek.PDFCreator
```

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

This class is used to generate PDF files. It consists of the problem's description and its solution. Furthermore there is the possibility of adding images to the document.

Author:

Marcel Wieczorek

Constructor Summary

PDFCreator(java.lang.String filename, java.lang.String description, java.lang.String strategy, java.lang.String who, java.lang.String id)

Method Summary

boolean	**createPDF**(java.lang.String title) Generates a PDF file with the corporate design of the 'Freie Universität Berlin'.
com.lowagie.text.Image	**getResizedImage**(com.lowagie.text.Image image) Method resizes image to a maximum size of 300 X 225 pixels.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PDFCreator

```
public PDFCreator(java.lang.String filename,
                  java.lang.String description,
                  java.lang.String strategy,
                  java.lang.String who,
                  java.lang.String id)
```

Parameters:

`filename` - the name of the file or its absolute path

`description` - the problem's description

`strategy` - the problem's solution

`who` - the person / department who made the contents of the document

`id` - the problem's id in the data base

Method Detail

createPDF

```
public boolean createPDF(java.lang.String title)
                  throws java.io.IOException
```

Generates a PDF file with the corporate design of the 'Freie Universität Berlin'.
The document contains the description of the problem as well as the strategy to solve that problem and optionally an archive of graphics which can possibly be necessary to visualize the manual.

Parameters:

`title` - the document's headline

Throws:

`java.io.IOException`

---

getResizedImage

```
public com.lowagie.text.Image getResizedImage(com.lowagie.text.Image image)
```

Method resizes image to a maximum size of 300 X 225 pixels.

Parameters:

`image` - Image that should be resized

Returns:

resized image (max 300 X 225 pixels)

9.3.3 Package: de.wieczorek.db

9.3.3.1 Database

de.wieczorek.db

Class Database

```
java.lang.Object
  └ de.wieczorek.db.Database
```

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

This class realizes to connections to the data bases and closes them. It returns ResultSets and sets updates.

Author:

Marcel Wieczorek

Constructor Summary

Database(java.lang.String dbPath)

Method Summary

boolean	**closeConnection**(java.sql.Connection con)
java.sql.Connection	**getConnection**()
java.util.Vector<java.lang.String>	**getResults**(java.lang.String sql, int[] cols)
boolean	**setUpdate**(java.lang.String sql)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Database

```
public Database(java.lang.String dbPath)
```

Parameters:

dbPath - the absolute path of the data base (either TroubleShooter.mdb or Backend.mdb)

Method Detail

getConnection

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

Returns:

Connection to the data base

closeConnection

```
public boolean closeConnection(java.sql.Connection con)
```

Parameters:

`con` - Connection that should be closed

Returns:

true if closing was successful otherwise false

---

setUpdate

```
public boolean setUpdate(java.lang.String sql)
```

Parameters:

`sql` - SQL statement as String

Returns:

true if update was successful otherwise false

---

getResults

```
public java.util.Vector<java.lang.String> getResults(java.lang.String sql,
                                                     int[] cols)
```

Parameters:

`sql` - SQL statement as String

`cols` - columns that should be read from the data base

Returns:

given columns

9.5.3.2 DBLoader

de.wieczorek.db

Class DBLoader

```
java.lang.Object
  └ de.wieczorek.db.DBLoader
```

All Implemented Interfaces:

java.lang.Runnable

```
public class DBLoader
extends java.lang.Object
implements java.lang.Runnable
```

This class is used to load all data from the database.

Author:

Marcel Wieczorek

Constructor Summary

`DBLoader()`
`DBLoader(MainWindow main)`

Method Summary

java.lang.String[]	**getFullOpenIncidentInfo**(java.lang.String constraint) Get information about respective incident
java.lang.String[][]	**getFullPCInfo**()
java.lang.String[]	**getFullPCInfo**(java.lang.String value)
java.lang.String[]	**getFullPCInfoHeader**()
java.lang.String[]	**getIncidentHeader**()
java.lang.String[][]	**getIncidentInfo**(java.lang.String constraint) Get incidents that fulfill a given constraint
void	**run**()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DBLoader

```
public DBLoader(MainWindow main)
```

Parameters:

`main` - MainWindow reference

DBLoader

```
public DBLoader()
```

Method Detail

run

```
public void run()
```

Specified by:

`run` in interface `java.lang.Runnable`

---

getFullPCInfoHeader

```
public java.lang.String[] getFullPCInfoHeader()
```

Returns:

single headlines for the table's columns

getFullPCInfo

`public java.lang.String[][] getFullPCInfo()`

Returns:

array with all information about each computer

---

getFullPCInfo

`public java.lang.String[] getFullPCInfo(java.lang.String value)`

Parameters:

`value` - name of the computer

Returns:

array that contains all information about respective computer

---

getIncidentHeader

`public java.lang.String[] getIncidentHeader()`

Returns:

single headlines for the table's columns

---

getIncidentInfo

`public java.lang.String[][] getIncidentInfo(java.lang.String constraint)`

Get incidents that fulfill a given constraint

Parameters:

`constraint` - final part of SQL statement

'WHERE tProblem.f_stID = 1' --> all open incidents

'WHERE tProblem.f_stID = 2' --> all closed incidents

'WHERE tProblem.f_stID = 2 AND f_katID = 2' --> all closed software incidents

'WHERE tProblem.f_stID = 2 AND f_katID = 1' --> all closed hardware incidents

Returns:

array that contains data depending on the constraint

---

getFullOpenIncidentInfo

`public java.lang.String[] getFullOpenIncidentInfo(java.lang.String constraint)`

Get information about respective incident

Parameters:

`constraint` - final part of a SQL statement that can only look like 'WHERE tProblem.pID = (int)'.

Returns:

array that contains information about one incident

9.5.3.3 Finder

de.wieczorek.db

Class Finder

```
java.lang.Object
  └ de.wieczorek.db.Finder
```

All Implemented Interfaces:

java.lang.Runnable

```
public class Finder
extends java.lang.Object
implements java.lang.Runnable

This class is used to find the respective incidents and shows them in a
table, ordered by relevance.
```

Author:

Marcel Wieczorek

Constructor Summary

Finder(MainWindow main, Builder builder, Handler handler, java.lang.String search)

Method Summary

java.lang.String[]	**getHitsHeader**()
void	**run**()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Finder

```
public Finder(MainWindow main,
              Builder builder,
              Handler handler,
              java.lang.String search)
```

Parameters:

`main` - MainWindow reference

`builder` - Builder reference

`handler` - Handler reference

`search` - string that contains keywords

Method Detail

run

```
public void run()
```

Specified by:

`run` in interface `java.lang.Runnable`

---

getHitsHeader

```
public java.lang.String[] getHitsHeader()
```

Returns:

headlines for the result table

9.5.4 Package: de.wieczorek.gui

9.5.4.1 Builder

de.wieczorek.gui

Class Builder

```
java.lang.Object
  └ de.wieczorek.gui.Builder
```

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

This class is used to initialize the main window and to build the respective tables.

Author:

Marcel Wieczorek

Constructor Summary

Builder(MainWindow main)

Method Summary

org.jdesktop.swingx.JXTitledPanel	**buildCenter**() builds the main content panel
org.jdesktop.swingx.JXTable	**buildJXTable**(java.lang.String[][] values, java.lang.String[] headers, java.lang.String name) Method builds a JXTable with non-editable cells.
javax.swing.JPanel	**buildSouth**(org.jdesktop.swingx.JXBusyLabel busy) builds the status bar
org.jdesktop.swingx.JXTitledPanel	**buildWest**() builds the menu on the left of the main window

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Builder

`public` **`Builder`**`(MainWindow main)`

Parameters:

`main` - reference of the MainWindow object

Method Detail

buildCenter

`public org.jdesktop.swingx.JXTitledPanel` **`buildCenter`**`()`

builds the main content panel

Returns:

JXTitledPanel that is added to the main window

---

buildSouth

`public javax.swing.JPanel` **`buildSouth`**`(org.jdesktop.swingx.JXBusyLabel busy)`

builds the status bar

Parameters:

`busy` - JXBusyLabel that shows if any action is performed

Returns:

JPanel that is added to the bottom of the main window

---

buildWest

`public org.jdesktop.swingx.JXTitledPanel` **`buildWest`**`()`

builds the menu on the left of the main window

Returns:

JXTitledPanel that is added to the left of the main window

---

buildJXTable

```
public org.jdesktop.swingx.JXTable
buildJXTable(java.lang.String[][] values,
                                    java.lang.String[] headers,
                                    java.lang.String name)
```

Method builds a JXTable with non-editable cells.

Parameters:

`values` - filled into grids

`headers` - headline of each column

`name` - the name of the JXTable

9.5.4.2 Content

de.wieczorek.gui

Class Content

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JPanel
                  └de.wieczorek.gui.Content
```

All Implemented Interfaces:

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

---

```
public class Content
extends javax.swing.JPanel
```

This class represents all panels that only contain a JXTable.

Author:

Marcel Wieczorek

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

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

Field Summary

Fields inherited from class javax.swing.JComponent
TOOL_TIP_TEXT_KEY, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

Content(Handler handler, org.jdesktop.swingx.JXTable table)

Method Summary

java.lang.String	**getPcSearchFlag**()
void	**setPcSearchFlag**(java.lang.String pcSearchFlag)

Methods inherited from class javax.swing.JPanel
getAccessibleContext, getUI, getUIClassID, setUI, updateUI

Methods inherited from class javax.swing.JComponent
...

Methods inherited from class java.awt.Container
...

Methods inherited from class java.awt.Component
...

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Content

```
public Content(Handler handler,
               org.jdesktop.swingx.JXTable table)
```

Parameters:

`handler` - Handler reference

`table` - the JXTable that should be added to the panel

Method Detail

getPcSearchFlag

`public java.lang.String` **`getPcSearchFlag`**`()`

Returns:

the value you are looking for searching a computer

---

setPcSearchFlag

`public void` **`setPcSearchFlag`**`(java.lang.String pcSearchFlag)`

Parameters:

`pcSearchFlag` - the value you are looking for searching a computer

9.5.4.3 Handler

de.wieczorek.gui

Class Handler

```
java.lang.Object
  └ de.wieczorek.gui.Handler
```

All Implemented Interfaces:

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

---

```
public class Handler
extends java.lang.Object
implements java.awt.event.ActionListener, java.awt.event.KeyListener,
java.awt.event.MouseListener
```

Class handles most important events, such as clicking-events in the menu or double-clicks on a table.

Author:

Marcel Wieczorek

Constructor Summary

Handler(MainWindow main)

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e)
java.lang.String[][]	**changeGlobalPCInfos**(java.lang.String suche) The content of MainWindow's pc-information-table is changed.
void	**enableMenueWest**(boolean b) This method activates the menu on the left.

java.lang.String[][]	**getNewTableValues**(java.lang.String suche, int dimX) This method creates an array with updated data for MainWindow's pc-information-table.
java.lang.String	**getPcSearchFlag**()
java.lang.String[]	**getRechnernamen**() Method summarizes the names of all the pc's which the MainWindow's array 'TableValues' contains.
java.lang.String	**getTextFieldTextWest**(java.lang.String name)
void	**keyPressed**(java.awt.event.KeyEvent e)
void	**keyReleased**(java.awt.event.KeyEvent e)
void	**keyTyped**(java.awt.event.KeyEvent e)
void	**mouseClicked**(java.awt.event.MouseEvent e)
void	**mouseEntered**(java.awt.event.MouseEvent e)
void	**mouseExited**(java.awt.event.MouseEvent e)
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)
void	**refreshCenterContent**(javax.swing.JPanel newContent, java.lang.String newTitle, boolean scrollbar) This method clears the center panel and adds a new one to it.
void	**refreshTable**() This method fills in the updated data into the table.
void	**setPcSearchFlag**(java.lang.String pcSearchFlag)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Handler

```
public Handler(MainWindow main)
```

Method Detail

enableMenueWest

```
public void enableMenueWest(boolean b)
```

This method activates the menu on the left. It is necessary because the menu must not be activated at the program's start to avoid that the user calls an element which is not referenced yet.

---

getTextFieldTextWest

```
public java.lang.String getTextFieldTextWest(java.lang.String name)
```

---

changeGlobalPCInfos

```
public java.lang.String[][] changeGlobalPCInfos(java.lang.String suche)
```

The content of MainWindow's pc-information-table is changed. Data is manipulated by entering text into the TextField (live searching)

Parameters:

`suche` - text entered in the TextField

---

getNewTableValues

```
public java.lang.String[][] getNewTableValues(java.lang.String suche,
                                              int dimX)
```

This method creates an array with updated data for MainWindow's pc-information-table.

Parameters:

`suche` - text entered in the TextField

`dimX` - column of the array (0 = 'Rechnername', 5 = 'SAP', 6 = 'MAC')

---

refreshTable

```
public void refreshTable()
```

This method fills in the updated data into the table.

---

refreshCenterContent

```
public void refreshCenterContent(javax.swing.JPanel newContent,
                                 java.lang.String newTitle,
                                 boolean scrollbar)
```

This method clears the center panel and adds a new one to it.

Parameters:

`newContent` - the JPanel that should be added to the center panel

`newTitle` - the text that should appear on top of the JXTitledPane

`scrollbar` - sets a vertical scroll bar or not

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

---

keyPressed

```
public void keyPressed(java.awt.event.KeyEvent e)
```

Specified by:

`keyPressed` in interface `java.awt.event.KeyListener`

---

keyReleased

```
public void keyReleased(java.awt.event.KeyEvent e)
```

Specified by:

`keyReleased` in interface `java.awt.event.KeyListener`

---

keyTyped

```
public void keyTyped(java.awt.event.KeyEvent e)
```

Specified by:

`keyTyped` in interface `java.awt.event.KeyListener`

---

mouseClicked

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

Specified by:

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

---

mouseEntered

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

Specified by:

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

mouseExited

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

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`

getPcSearchFlag

```
public java.lang.String getPcSearchFlag()
```

---

setPcSearchFlag

```
public void setPcSearchFlag(java.lang.String pcSearchFlag)
```

---

getRechnernamen

```
public java.lang.String[] getRechnernamen()
```

Method summarizes the names of all the pc's which the MainWindow's array 'TableValues' contains. The values are used in the ComboBoxes of PanelNewIncident and PanelCloseIncident.

Returns:

String[]

9.5.4.4 ImgPrev

de.wieczorek.gui

Class ImgPrev

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JPanel
                  └de.wieczorek.gui.ImgPrev
```

All Implemented Interfaces:

java.awt.event.KeyListener, java.awt.event.MouseListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.EventListener, javax.accessibility.Accessible

---

```
public class ImgPrev
extends javax.swing.JPanel
implements java.awt.event.KeyListener, java.awt.event.MouseListener
```

This class is used to handle the images that are connected to a incident. It is an image viewer. You can switch between certain images, add some more and delete them. The images that are added are also added to the PDF file.
The TextField on the top contains the name of the image file. The TextField on the bottom contains a description of the image. You can enter up to 255 characters here which are also added to the PDF file.

Author:

Marcel Wieczorek

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

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

Field Summary

Fields inherited from class javax.swing.Jcomponent
TOOL_TIP_TEXT_KEY, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

ImgPrev(java.awt.Color bgColor) creates an ImgPrev object with no single image
ImgPrev(java.util.Vector<java.lang.String> vPath, java.util.Vector<java.lang.String> vDescription, java.awt.Color bgColor)

Method Summary

void	**addImage**() Opens a FileChooser.
void	**deleteImage**(int index) Delete image on a given position.
java.util.Vector<java.lang.String>	**getDescriptionVector**()

java.util.Vector<java.lang.String>	**getPathVector**()
java.awt.Image	**getResizedImage**(java.awt.Image image) Method resizes image to a maximum size of 150 X 100 pixels.
void	**keyPressed**(java.awt.event.KeyEvent e)
void	**keyReleased**(java.awt.event.KeyEvent e)
void	**keyTyped**(java.awt.event.KeyEvent e)
void	**mouseClicked**(java.awt.event.MouseEvent e)
void	**mouseEntered**(java.awt.event.MouseEvent e)
void	**mouseExited**(java.awt.event.MouseEvent e)
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class javax.swing.JPanel
getAccessibleContext, getUI, getUIClassID, setUI, updateUI

Methods inherited from class javax.swing.JComponent
...

Methods inherited from class java.awt.Container
...

Methods inherited from class java.awt.Component
...

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ImgPrev

```
public ImgPrev(java.util.Vector<java.lang.String> vPath,
               java.util.Vector<java.lang.String> vDescription,
               java.awt.Color bgColor)
```

Parameters:

`vPath` - Vector that includes absolute path(s) of image file(s)

`vDescription` - Vector that includes description(s) for respective images

`bgColor` - the color that is set as background color

ImgPrev

```
public ImgPrev(java.awt.Color bgColor)
```

creates an ImgPrev object with no single image

Parameters:

`bgColor` - the color that is set as background color

Method Detail

getResizedImage

```
public java.awt.Image getResizedImage(java.awt.Image image)
```

Method resizes image to a maximum size of 150 X 100 pixels.

Parameters:

`image` - Image that should be resized

Returns:

resized image (max 150 X 100 pixels)

---

addImage

```
public void addImage()
```

Opens a FileChooser.

Selected file is added to the ImgPrev.

---

deleteImage

```
public void deleteImage(int index)
```

Delete image on a given position.

Parameters:

`index` - Position of the image that should be deleted.

---

getPathVector

```
public java.util.Vector<java.lang.String> getPathVector()
```

Returns:

Vector containing path(s) of image file(s)

---

getDescriptionVector

```
public java.util.Vector<java.lang.String> getDescriptionVector()
```

Returns:

Vector containing description(s) of image file(s)

---

keyReleased

```
public void keyReleased(java.awt.event.KeyEvent e)
```

Specified by:

`keyReleased` in interface `java.awt.event.KeyListener`

---

keyTyped

```
public void keyTyped(java.awt.event.KeyEvent e)
```

Specified by:

`keyTyped` in interface `java.awt.event.KeyListener`

---

mouseClicked

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

Specified by:

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

---

mouseEntered

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

Specified by:

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

---

mouseExited

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

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`

---

keyPressed

```
public void keyPressed(java.awt.event.KeyEvent e)
```

Specified by:

`keyPressed` in interface `java.awt.event.KeyListener`

9.5.4.5 MainWindow

de.wieczorek.gui

Class MainWindow

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
                      └org.jdesktop.swingx.JXFrame
                          └de.wieczorek.gui.MainWindow
```

All Implemented Interfaces:

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

---

```
public class MainWindow
extends org.jdesktop.swingx.JXFrame
implements java.lang.Runnable
```

Main window which contains the different panels
It is divided into three sections (west, center, south) which are built using respective methods in the class 'Builder' (e.g. buildWest()).

Author:

Marcel Wieczorek

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class org.jdesktop.swingx.JXFrame
`org.jdesktop.swingx.JXFrame.StartPosition`

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

Field Summary

javax.swing.JPanel	**center**
javax.swing.JPanel	**south**
javax.swing.JPanel	**west**

Fields inherited from class javax.swing.JFrame
EXIT_ON_CLOSE

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

MainWindow()
Initializes the main window.

Method Summary

org.jdesktop.swingx.JXTable	**getAllClosedIncidentsTable**()
org.jdesktop.swingx.JXTable	**getAllOpenIncidentsTable**()
Builder	**getBuilder**()
static org.jdesktop.swingx.JXBusyLabel	**getBusy**()
static java.lang.String[]	**getListKeywords**()
org.jdesktop.swingx.JXTable	**getPcInfoTable**()
static java.lang.String	**getSuche**()
static java.lang.String[]	**getTableHeaders**()
static java.lang.String[]	**getTableHeadersClosedIncidents**()
static java.lang.String[]	**getTableHeadersOpenIncidents**()
static java.lang.String[][]	**getTableValues**()
static java.lang.String[][]	**getTableValuesClosedIncidents**()
static java.lang.String[][]	**getTableValuesOpenIncidents**()
void	**run**()
void	**setAllClosedIncidentsTable**(org.jdesktop.swingx.JXTable allClosedIncidentsTable)
void	**setAllOpenIncidentsTable**(org.jdesktop.swingx.JXTable allOpenIncidentsTable)
static void	**setListKeywords**(java.lang.String[] listKeywords)

void	**setPcInfoTable**(org.jdesktop.swingx.JXTable pcInfoTable)
static void	**setSuche**(java.lang.String suche)
static void	**setTableHeaders**(java.lang.String[] tableHeaders)
static void	**setTableHeadersClosedIncidents**(java.lang.String[] tableHeadersClosedIncidents)
static void	**setTableHeadersOpenIncidents**(java.lang.String[] tableHeadersOpenIncidents)
static void	**setTableValues**(java.lang.String[][] tableValues)
static void	**setTableValuesClosedIncidents**(java.lang.String[][] tableValuesClosedIncidents)
static void	**setTableValuesOpenIncidents**(java.lang.String[][] tableValuesOpenIncidents)

Methods inherited from class org.jdesktop.swingx.JXFrame
. . .

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
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer
getFont, postEvent

Field Detail

south

```
public javax.swing.JPanel south
```

---

west

```
public javax.swing.JPanel west
```

---

center

```
public javax.swing.JPanel center
```

Constructor Detail

MainWindow

```
public MainWindow()
```

Initializes the main window.

Method Detail

run

```
public void run()
```

Specified by:

`run` in interface `java.lang.Runnable`

---

getTableValues

```
public static java.lang.String[][] getTableValues()
```

Returns:

values of computer information table

---

setTableValues

```
public static void setTableValues(java.lang.String[][] tableValues)
```

Parameters:

`tableValues` - values of computer information table

---

getTableHeaders

```
public static java.lang.String[] getTableHeaders()
```

Returns:

headlines of computer information table

---

setTableHeaders

```
public static void setTableHeaders(java.lang.String[] tableHeaders)
```

Parameters:

`tableHeaders` - headlines of computer information table

---

getSuche

```
public static java.lang.String getSuche()
```

Returns:

string that defines search for computer

---

setSuche

```
public static void setSuche(java.lang.String suche)
```

Parameters:

`suche` - string that defines search for computer

---

getListKeywords

```
public static java.lang.String[] getListKeywords()
```

Returns:

array of keywords that specify an incident

---

setListKeywords

```
public static void setListKeywords(java.lang.String[] listKeywords)
```

Parameters:

`listKeywords` - array of keywords that specify an incident

---

getBuilder

```
public Builder getBuilder()
```

Returns:

reference of the Builder object

---

getTableValuesOpenIncidents

```
public static java.lang.String[][] getTableValuesOpenIncidents()
```

Returns:

values of open incident information table

---

setTableValuesOpenIncidents

```
public static void
setTableValuesOpenIncidents(java.lang.String[][] tableValuesOpenIncidents)
```

Parameters:

`tableValuesOpenIncidents` - values of open incident information table

---

getTableHeadersOpenIncidents

```
public static java.lang.String[] getTableHeadersOpenIncidents()
```

Returns:

headlines of open incident information table

---

setTableHeadersOpenIncidents

```
public static void
setTableHeadersOpenIncidents(java.lang.String[] tableHeadersOpenIncidents)
```

Parameters:

`tableHeadersOpenIncidents` - headlines of open incident information table

---

getTableValuesClosedIncidents

```
public static java.lang.String[][] getTableValuesClosedIncidents()
```

Returns:

values of closed incident information table

---

setTableValuesClosedIncidents

```
public static void
setTableValuesClosedIncidents(java.lang.String[][] tableValuesClosedIncidents)
```

Parameters:

`tableValuesClosedIncidents` - values of closed incident information table

---

getTableHeadersClosedIncidents

```
public static java.lang.String[] getTableHeadersClosedIncidents()
```

Returns:

headlines of closed incident information table

---

setTableHeadersClosedIncidents

```
public static void
setTableHeadersClosedIncidents(java.lang.String[] tableHeadersClosedIncidents)
```

Parameters:

`tableHeadersClosedIncidents` - headlines of closed incident information table

getBusy

`public static org.jdesktop.swingx.JXBusyLabel` **`getBusy`**`()`

Returns:

JXBusyLabel from the status bar

getPcInfoTable

`public org.jdesktop.swingx.JXTable` **`getPcInfoTable`**`()`

Returns:

computer information table

---

setPcInfoTable

`public void` **`setPcInfoTable`**`(org.jdesktop.swingx.JXTable pcInfoTable)`

Parameters:

`pcInfoTable` - computer information table

---

getAllOpenIncidentsTable

`public org.jdesktop.swingx.JXTable` **`getAllOpenIncidentsTable`**`()`

Returns:

open incident information table

---

setAllOpenIncidentsTable

`public void`
`setAllOpenIncidentsTable``(org.jdesktop.swingx.JXTable allOpenIncidentsTable)`

Parameters:

`allOpenIncidentsTable` - open incident information table

---

getAllClosedIncidentsTable

`public org.jdesktop.swingx.JXTable` **`getAllClosedIncidentsTable`**`()`

Returns:

closed incident information table

---

setAllClosedIncidentsTable

`public void`
`setAllClosedIncidentsTable``(org.jdesktop.swingx.JXTable allClosedIncidentsTable)`

Parameters:

`allClosedIncidentsTable` - closed incident information table

9.5.4.6 PanelCloseIncident

de.wieczorek.gui

Class PanelCloseIncident

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JPanel
                  └de.wieczorek.gui.PanelCloseIncident
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.awt.event.KeyListener, java.awt.event.MouseListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.EventListener, javax.accessibility.Accessible

---

```
public class PanelCloseIncident
extends javax.swing.JPanel
implements java.awt.event.ActionListener, java.awt.event.KeyListener,
java.awt.event.MouseListener
```

The form to edit and close an open incident

Author:

Marcel Wieczorek

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

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

Field Summary

Fields inherited from class javax.swing.JComponent
TOOL_TIP_TEXT_KEY, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

PanelCloseIncident(Handler handler, MainWindow main, java.lang.String id)

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e)
void	**deleteFromList**() Deletes keyword from the list.
void	**generatePDF**(boolean print) Exports incident as PDF file.
void	**keyPressed**(java.awt.event.KeyEvent e)
void	**keyReleased**(java.awt.event.KeyEvent e)
void	**keyTyped**(java.awt.event.KeyEvent e)
void	**mouseClicked**(java.awt.event.MouseEvent e)
void	**mouseEntered**(java.awt.event.MouseEvent e)
void	**mouseExited**(java.awt.event.MouseEvent e)
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)
void	**refreshKeywordList**(java.lang.String toAdd) Updates keyword list.
void	**resetContent**() Clears the form.
void	**saveIncident**()
void	**selectKeyword**(java.awt.event.MouseEvent e, int currPos) Selects a word on the given position if a given MouseEvent is raised.

Methods inherited from class javax.swing.JPanel
...

Methods inherited from class javax.swing.JComponent
...

Methods inherited from class java.awt.Container
...

Methods inherited from class java.awt.Component
...

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

PanelCloseIncident

```
public PanelCloseIncident(Handler handler,
                          MainWindow main,
                          java.lang.String id)
```

Parameters:

`handler` - reference to a Handler object

`main` - MainWindow reference

`id` - incident's id

Method Detail

deleteFromList

```
public void deleteFromList()
```

Deletes keyword from the list.

---

generatePDF

```
public void generatePDF(boolean print)
```

Exports incident as PDF file.

Parameters:

`print` - true if PDF file should be printed otherwise false

---

saveIncident

```
public void saveIncident()
```

---

resetContent

```
public void resetContent()
```

Clears the form.

---

selectKeyword

```
public void selectKeyword(java.awt.event.MouseEvent e,
                          int currPos)
```

Selects a word on the given position if a given MouseEvent is raised.

The word is added to the keyword list.

Parameters:

`e` - MouseEvent (mouseClicked)

`currPos` - CaretPosition

---

refreshKeywordList

```
public void refreshKeywordList(java.lang.String toAdd)
```

Updates keyword list.

Parameters:

`toAdd` - keyword that should be added to the keyword list

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

---

keyPressed

```
public void keyPressed(java.awt.event.KeyEvent e)
```

Specified by:

`keyPressed` in interface `java.awt.event.KeyListener`

---

keyReleased

```
public void keyReleased(java.awt.event.KeyEvent e)
```

Specified by:

`keyReleased` in interface `java.awt.event.KeyListener`

---

keyTyped

```
public void keyTyped(java.awt.event.KeyEvent e)
```

Specified by:

`keyTyped` in interface `java.awt.event.KeyListener`

---

mouseClicked

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

Specified by:

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

---

mouseEntered

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

Specified by:

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

mouseExited

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

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`

9.5.4.7 PanelNewIncident

de.wieczorek.gui

Class PanelNewIncident

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JPanel
                  └de.wieczorek.gui.PanelNewIncident
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.awt.event.KeyListener, java.awt.event.MouseListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.util.EventListener, javax.accessibility.Accessible

---

```
public class PanelNewIncident
extends javax.swing.JPanel
implements java.awt.event.ActionListener, java.awt.event.KeyListener,
java.awt.event.MouseListener
```

The form to open an incident

Author:

Marcel Wieczorek

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent
Nested classes/interfaces inherited from class java.awt.Component
java.awt.Component.BaselineResizeBehavior

Field Summary

Fields inherited from class javax.swing.JComponent
TOOL_TIP_TEXT_KEY, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

PanelNewIncident(Handler handler, MainWindow main)

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e)
void	**deleteFromList**() Deletes keyword from the list.
void	**keyPressed**(java.awt.event.KeyEvent e)
void	**keyReleased**(java.awt.event.KeyEvent e)
void	**keyTyped**(java.awt.event.KeyEvent e)

void	**mouseClicked**(java.awt.event.MouseEvent e)
void	**mouseEntered**(java.awt.event.MouseEvent e)
void	**mouseExited**(java.awt.event.MouseEvent e)
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)
void	**refreshKeywordList**(java.lang.String toAdd) Updates keyword list.
void	**resetContentNewIncident**()
void	**saveNewIncident**() Saves the incident as an open one.
void	**selectKeyword**(java.awt.event.MouseEvent e, int currPos) Selects a word on the given position if a given MouseEvent is raised.

Methods inherited from class javax.swing.JPanel
...

Methods inherited from class javax.swing.JComponent
...

Methods inherited from class java.awt.Container
...

Methods inherited from class java.awt.Component
...

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

PanelNewIncident

```
public PanelNewIncident(Handler handler,
                        MainWindow main)
```

Parameters:

`handler` - reference to a Handler object

`main` - MainWindow reference

Method Detail

deleteFromList

```
public void deleteFromList()
```

Deletes keyword from the list.

---

saveNewIncident

```
public void saveNewIncident()
```

Saves the incident as an open one.

---

resetContentNewIncident

```
public void resetContentNewIncident()
```

---

selectKeyword

```
public void selectKeyword(java.awt.event.MouseEvent e,
                          int currPos)
```

Selects a word on the given position if a given MouseEvent is raised.

The word is added to the keyword list.

Parameters:

`e` - MouseEvent (mouseClicked)

`currPos` - CaretPosition

See Also:

`PanelCloseIncident`

---

refreshKeywordList

```
public void refreshKeywordList(java.lang.String toAdd)
```

Updates keyword list.

Parameters:

`toAdd` - keyword that should be added to the keyword list

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

keyPressed

```
public void keyPressed(java.awt.event.KeyEvent e)
```

Specified by:

`keyPressed` in interface `java.awt.event.KeyListener`

---

keyReleased

```
public void keyReleased(java.awt.event.KeyEvent e)
```

Specified by:

`keyReleased` in interface `java.awt.event.KeyListener`

keyTyped

```
public void keyTyped(java.awt.event.KeyEvent e)
```

Specified by:

`keyTyped` in interface `java.awt.event.KeyListener`

---

mouseClicked

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

Specified by:

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

---

mouseEntered

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

Specified by:

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

---

mouseExited

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

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`

9.5.5 Package: de.wieczorek.security

9.5.5.1 LoginDialog

de.wieczorek.security

Class LoginDialog

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Dialog
                  └javax.swing.JDialog
                      └de.wieczorek.security.LoginDialog
```

All Implemented Interfaces:

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

---

```
public class LoginDialog
extends javax.swing.JDialog
```

Main class; entry point of the program

Author:

Marcel Wieczorek

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class java.awt.Dialog
java.awt.Dialog.ModalExclusionType, java.awt.Dialog.ModalityType

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

Field Summary

Fields inherited from class java.awt.Dialog
DEFAULT_MODALITY_TYPE

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

LoginDialog(java.awt.Frame parent, boolean modal)

Method Summary

static void	**main**(java.lang.String[] args) program's entry point sets look and feel; opens login dialog

Methods inherited from class javax.swing.JDialog
...

Methods inherited from class java.awt.Dialog
...

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
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

LoginDialog

```
public LoginDialog(java.awt.Frame parent,
                   boolean modal)
```

Method Detail

main

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

program's entry point sets look and feel; opens login dialog

9.5.5.2 MyMD5

de.wieczorek.security

Class MyMD5

```
java.lang.Object
  └ de.wieczorek.security.MyMD5
```

```
public class MyMD5
extends java.lang.Object
```

This class encrypts a given string with MD5. It is used to encrypt the user name and the password.

Author:

Marcel Wieczorek

Constructor Summary

MyMD5(java.lang.String toEncrypt)

Method Summary

java.lang.String	encrypt() encrypts the string that is given.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MyMD5

```
public MyMD5(java.lang.String toEncrypt)
```

Parameters:

toEncrypt - the string that should be encrypted

Method Detail

encrypt

`public java.lang.String` **`encrypt`**`()`

encrypts the string that is given.

Returns:

encrypted string how it is written into the data base

9.6 Quelltextauszug

ImgPrev.java

```
package de.wieczorek.gui;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.ComponentOrientation;
import java.awt.Dimension;
import java.awt.Image;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.io.File;
import java.util.Vector;

import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JToolBar;
import javax.swing.filechooser.FileNameExtensionFilter;

import de.wieczorek.Corporate;
/**
 * This class is used to handle the images that are connected to a
 * incident.
 * It is an image viewer. You can switch between certain images, add some
 * more and delete them.
 * The images that are added are also added to the PDF file.
 * <br>
 * The TextField on the top contains the name of the image file. The
 * TextField on the bottom contains a
 * description of the image. You can enter up to 255 characters here which
 * are also added to the
 * PDF file.
 * @author Marcel Wieczorek
 */
public class ImgPrev          extends JPanel
                                        implements KeyListener,
MouseListener {
      JTextField txtName, txtDescription;
      JLabel lblImage, lblUp, lblDown, lblAdd, lblDelete;
      Vector<String> vPath, vDescription;
      String tooltip = "";

      private static final long serialVersionUID = 1L;
      private int imgValue = 0;
/**
 * @param vPath Vector that includes absolute path(s) of image file(s)
 * @param vDescription Vector that includes description(s) for respective
 * images
 * @param bgColor the color that is set as background color
 */
```

```
public ImgPrev(Vector<String> vPath, Vector<String> vDescription,
Color bgColor) {
	if(vPath == null || vDescription == null) {
		vPath = new Vector<String>();
		vDescription = new Vector<String>();
	}
	this.vPath = vPath;
	this.vDescription = vDescription;

	setLayout(new BorderLayout());
	setBackground(Corporate.DARK_BLUE);
	setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
	setBackground(bgColor);

setBorder(BorderFactory.createLineBorder(Corporate.LIGHT_BLUE));
	txtName = new JTextField();
		txtName.setPreferredSize(
		new Dimension(this.getWidth() - 5, 20));
		txtName.setEnabled(false);
	txtDescription = new JTextField();
		txtDescription.setName("TXTDESCRIPTION");
		txtDescription.addKeyListener(this);
		txtDescription.setPreferredSize(new
		Dimension(this.getWidth() - 5, 20));
		txtDescription.setEnabled(false);

	lblUp = new JLabel(Corporate.IMG32_IMAGE_FORWARD);
		lblUp.setName("BTNUP");
		lblUp.setToolTipText("Nächstes Bild");
	lblDown = new JLabel(Corporate.IMG32_IMAGE_REWIND);
		lblDown.setName("BTNDOWN");
		lblDown.setToolTipText("Vorheriges Bild");
	lblAdd = new JLabel(Corporate.IMG32_ADD_IMAGE);
		lblAdd.setName("BTNADD");
		lblAdd.setToolTipText("Bild hinzufügen");
	lblDelete = new JLabel(Corporate.IMG32_TRASH);
		lblDelete.setName("BTNDELETE");
		lblDelete.setToolTipText("Aktuelles Bild löschen");

	if(!this.vPath.isEmpty()) { // at least 1 element included
		showImage(imgValue);
	}
	else {
		lblImage = new JLabel("Kein Bild");
		lblImage.setName("lblImage");
		lblImage.setToolTipText("Kein Bild");
	}

	lblImage.setVerticalAlignment(JLabel.CENTER);
	lblImage.setHorizontalAlignment(JLabel.CENTER);
	lblImage.setBorder(
	BorderFactory.createLineBorder(Corporate.DARK_BLUE));
	lblImage.setPreferredSize(new Dimension(155, 105));

	txtDescription.setBorder(
	BorderFactory.createLineBorder(Corporate.DARK_BLUE));
	txtName.setBorder(
	BorderFactory.createLineBorder(Corporate.DARK_BLUE));

	lblUp.setBorder(
```

```
            BorderFactory.createLineBorder(Corporate.DARK_BLUE));
            lblDown.setBorder(
            BorderFactory.createLineBorder(Corporate.DARK_BLUE));
            lblAdd.setBorder(
            BorderFactory.createLineBorder(Corporate.DARK_BLUE));
            lblDelete.setBorder(
            BorderFactory.createLineBorder(Corporate.DARK_BLUE));

            JToolBar bar = new JToolBar(JToolBar.VERTICAL);

      bar.setBorder(BorderFactory.createLineBorder(Corporate.DARK_BLUE));
                  bar.setBackground(bgColor);
            bar.add(lblUp);
            bar.add(lblDown);
            bar.add(lblAdd);
            bar.add(lblDelete);

            for(Component c : bar.getComponents()) {
                  if(c instanceof JLabel)
                        ((JLabel) c).addMouseListener(this);
            }

            add(txtName, BorderLayout.NORTH);
            add(txtDescription, BorderLayout.SOUTH);
            add(lblImage, BorderLayout.CENTER);
            add(bar, BorderLayout.EAST);
      }
/**
 * creates an ImgPrev object with no single image
 * @param bgColor the color that is set as background color
 */
      public ImgPrev(Color bgColor) {
            this(new Vector<String>(), new Vector<String>(), bgColor);
      }
      /*
       * Method gets selected image and shows it.
       * If there is no single image 'Kein Bild' appears.
       */
      private void showImage(int index) {
            if(!this.vPath.isEmpty()) {
                  if(index < this.vPath.size()) {
                        if(index >= 0) {
                              File tmp = new File(this.vPath.get(imgValue));
                              txtName.setText(tmp.getName());

      txtDescription.setText(this.vDescription.get(imgValue));
                              Image image = new
      ImageIcon(tmp.getAbsolutePath()).getImage();
                              ImageIcon ico = new
      ImageIcon(getResizedImage(image), tmp.getName());
                              lblImage.setIcon(ico);
                              lblImage.setText("");
                                    lblImage.setName("lblImage");

      lblImage.setToolTipText(this.vDescription.get(imgValue));
                        }
                        else {
                              imgValue = this.vPath.size() - 1;
                              showImage(imgValue);
                        }
```

```
                }
                else {
                        imgValue = 0;
                        showImage(imgValue);
                }
            }
            else {
                lblImage.setIcon(null);
                lblImage.setText("Kein Bild");
                lblImage.setName("lblImage");
                lblImage.setToolTipText("Kein Bild");
                txtDescription.setText("");
                txtName.setText("");
            }
        }
/**
 * Method resizes image to a maximum size of 150 X 100 pixels.
 * @param image Image that should be resized
 * @return resized image (max 150 X 100 pixels)
 */
      public Image getResizedImage(Image image) {
            // get size of the image --> set new size
            int tmpWidth = image.getWidth(null);
            int tmpHeight = image.getHeight(null);

            if(tmpWidth > 150) {
                double      factor = 15000 / tmpWidth;
                            factor /= 100;
                tmpWidth = 150;
                tmpHeight = (int) (tmpHeight * factor);
            }
            if(tmpHeight > 100) {
                double      factor = 10000 / tmpHeight;
                            factor /= 100;
                tmpHeight = 100;
                tmpWidth = (int) (tmpWidth * factor);
            }
            image = image.getScaledInstance(tmpWidth, tmpHeight,
            Image.SCALE_SMOOTH);
            return image;
      }
/**
 * Opens a FileChooser.
 * <br>
 * Selected file is added to the ImgPrev.
 */
      public void addImage() {
            File f = null;
            JFileChooser chooser = new JFileChooser("C:");
          FileNameExtensionFilter filter = new FileNameExtensionFilter(
              "Image files", "jpg", "gif", "png", "JPG", "GIF", "PNG");
          chooser.setFileFilter(filter);
          chooser.setAcceptAllFileFilterUsed(false);
          chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
          int returnVal = chooser.showOpenDialog(null);
          if(returnVal == JFileChooser.APPROVE_OPTION) {
            f = chooser.getSelectedFile();
          }

          this.vPath.add(f.getAbsolutePath());
```

```
			this.vDescription.add("-");

			imgValue = this.vPath.size() - 1;
			showImage(imgValue);
		}// end of addImage()
/**
 * Delete image on a given position.
 * @param index Position of the image that should be deleted.
 */
		public void deleteImage(int index) {
			if(index >= 0 && !this.vPath.isEmpty()) {
				this.vPath.remove(index);
				this.vDescription.remove(index);
				imgValue--;
				if(imgValue >= 0 && !vPath.isEmpty())
					showImage(imgValue);
				else
					deleteImage(-1);
			}
			else {
		JOptionPane.showMessageDialog(null, "Keine Bilder mehr vorhanden.");
				showImage(-1);
				disableTextFields();
			}
		}// end of deleteImage()

		private void disableTextFields() {
			txtDescription.setEnabled(false);
			txtName.setEnabled(false);
		}
/**
 * @return Vector containing path(s) of image file(s)
 */
		public Vector<String> getPathVector() {
			return this.vPath;
		}
/**
 * @return Vector containing description(s) of image file(s)
 */
		public Vector<String> getDescriptionVector() {
			return this.vDescription;
		}

		public void keyReleased(KeyEvent e) {
			if(e.getSource() instanceof JTextField && ((JTextField)
e.getSource()).getName().equals("TXTDESCRIPTION")) { //Textfeld
Beschreibung
				tooltip = ((JTextField) e.getSource()).getText();
				lblImage.setToolTipText(tooltip);
				vDescription.set(imgValue, tooltip);
			}
		}

		public void keyTyped(KeyEvent e) {

		}

		public void mouseClicked(MouseEvent e) {
			if(e.getSource() instanceof JLabel) {
				if(e.getComponent().getName().equals("BTNUP")) {
```

```
                        imgValue++;
                        showImage(imgValue);
                  }
                  else if(e.getComponent().getName().equals("BTNDOWN")) {
                        imgValue--;
                        showImage(imgValue);
                  }
                  else if(e.getComponent().getName().equals("BTNADD")) {
                        txtDescription.setEnabled(true);
                        addImage();
                  }
                  else if(e.getComponent().getName().equals("BTNDELETE")) {
                        deleteImage(imgValue);
                  }
            }
      }

      public void mouseEntered(MouseEvent e) {
            if(e.getSource() instanceof JLabel) {
                  e.getComponent().setBackground(Corporate.DARK_BLUE);
            }
      }

      public void mouseExited(MouseEvent e) {
            if(e.getSource() instanceof JLabel) {
                  e.getComponent().setBackground(Corporate.DARK_BLUE);
            }
      }

      public void mousePressed(MouseEvent e) {
      }

      public void mouseReleased(MouseEvent e) {
      }

      public void keyPressed(KeyEvent e) {
      }
}
```

Finder.java

```
package de.wieczorek.db;

import java.text.DecimalFormat;
import javax.swing.JPanel;
import org.jdesktop.swingx.decorator.SortOrder;
import com.sun.org.apache.xalan.internal.xsltc.runtime.Hashtable;

import de.wieczorek.gui.Builder;
import de.wieczorek.gui.Content;
import de.wieczorek.gui.Handler;
import de.wieczorek.gui.MainWindow;
/**
 * @author Marcel Wieczorek
 * This class is used to find the respective incidents and shows them in a
 * table,
 * ordered by relevance.
 */
public class Finder implements Runnable {
MainWindow main;
Builder builder;
```

```
Handler handler;
	String[][] solutionsArray, hitsArray;
	String[] searchArray;
	String search;
/**
 * @param main MainWindow reference
 * @param builder Builder reference
 * @param handler Handler reference
 * @param search string that contains keywords
 */
	public Finder(MainWindow main, Builder builder, Handler handler,
	String search) {
		this.main = main;
		this.builder = builder;
		this.handler = handler;
		this.search = search;
	}

	public void run() {
		MainWindow.getBusy().setBusy(true);
		MainWindow.getBusy().setText("Warte auf Informationen...");

		this.solutionsArray = new DBLoader().getIncidentInfo("WHERE
		tProblem.f_stID = 2");// Status = 'geschlossen'
		// extract single words from the string
		this.searchArray = this.search.split(" ");
		for(int i = 0; i < this.searchArray.length; i++) {
		this.searchArray[i] = this.searchArray[i].replaceAll(" ", "");
		}
		/*
		 * check for each data set whether problem's description
		 * contains words from the string
		 * Hashtable includes ID as key and number of found words as
		 * value
		 */
		Hashtable hashHitsDesc = new Hashtable();
		for(int i = 0; i < this.solutionsArray.length; i++) {
			int count = 0;
			for(String s : this.searchArray) {

	if((this.solutionsArray[i][7].toLowerCase()).indexOf(s.toLowerCase())
	!= -1)// Wort gefunden?
		count++;
	hashHitsDesc.put(this.solutionsArray[i][0], count);// ID eintragen
			}
		}
		/*
		 * check for each data set whether list of keywords contains
		 * words from the string
		 * Hashtable includes ID as key and number of found words as
		 * value
		 */
		Hashtable hashHitsKey = new Hashtable();
		for(int i = 0; i < this.solutionsArray.length; i++) {
			int count = 0;
			for(String s : this.searchArray) {
				String tmpSearch = this.solutionsArray[i][9];
					tmpSearch = tmpSearch.replaceAll("%", "");
	if((tmpSearch.toLowerCase()).indexOf(s.toLowerCase()) !=
	-1)// Wort gefunden?
```

```
                  count++;
            hashHitsKey.put(this.solutionsArray[i][0], count);// ID eintragen
                  }
            }
            /*
             * check for each data set whether headline contains words from
             * the string
             * Hashtable includes ID as key and number of found words as
             * value
             */
            Hashtable hashHitsTitle = new Hashtable();
            for(int i = 0; i < this.solutionsArray.length; i++) {
                  int count = 0;
                  for(String s : this.searchArray) {

            if((this.solutionsArray[i][3].toLowerCase()).indexOf(s.toLowerCase())
            != -1)// Wort gefunden?
                  count++;
            hashHitsTitle.put(this.solutionsArray[i][0], count);// ID eintragen
                  }
            }
            /*
             * summarizing Hashtable that includes ID as key and the
             * incident's relevance as value
             * weights of each field:
             * - description --> 30 %
             * - keywords --> 40 %
             * - title --> 30 %
             */
            Hashtable hashQuote = new Hashtable();
            for(int i = 0; i < this.solutionsArray.length; i++) {
                  String id = this.solutionsArray[i][0];
                  int words = this.searchArray.length;
double hitsDesc = Double.valueOf(hashHitsDesc.get(id).toString());
double hitsKey = Double.valueOf(hashHitsKey.get(id).toString());
double hitsTitle = Double.valueOf(hashHitsTitle.get(id).toString());
double quote = (((100 * hitsKey) / words) * 0.4) + (((100 * hitsDesc) /
words) * 0.3) + (((100 * hitsTitle) / words) * 0.3);
                  if(quote > 0.00)
                        quote -= 0.01;
                  DecimalFormat df = new DecimalFormat("##0.00");
                        hashQuote.put(this.solutionsArray[i][0],
df.format(quote));
            }
            /*
             * create array that includes all necessary values
             */
            String[][] foundSolutions = new
String[this.solutionsArray.length][5];
            for(int i = 0; i < this.solutionsArray.length; i++) {
            foundSolutions[i][0] = this.solutionsArray[i][0];// ID (DB)
            foundSolutions[i][1] = this.solutionsArray[i][2];// Bearbeiter (DB)
            foundSolutions[i][2] = this.solutionsArray[i][3];// Titel (DB)
            foundSolutions[i][3] = this.solutionsArray[i][4];// Datum (DB)
                  foundSolutions[i][4] =
(String)hashQuote.get(this.solutionsArray[i][0]);// relevance
            }

            this.main.setAllClosedIncidentsTable(
```

```
builder.buildJXTable(foundSolutions, this.getHitsHeader(),
"FOUNDSOLUTIONS"));
            this.main.getAllClosedIncidentsTable().setSortOrder(4,
SortOrder.DESCENDING);
            handler.refreshCenterContent(new JPanel(), "Zutreffende
Ergebnisse", false);
            handler.refreshCenterContent(new Content(handler,
this.main.getAllClosedIncidentsTable()), "Zutreffende Ergebnisse", false);

            MainWindow.getBusy().setText("Bereit");
            MainWindow.getBusy().setBusy(false);
      }
      /**
       * @return headlines for the result table
       */
      public String[] getHitsHeader() {
            String[] header =
            {
                  "ID", "Bearbeiter", "Titel", "Datum", "Quote (%)"
            };
            return header;
      }//end of getFullPCInfoHeader
}
```

Database.java

```
package de.wieczorek.db;

import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Vector;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileNameExtensionFilter;

import de.wieczorek.Corporate;
/**
 * This class realizes to connections to the data bases and closes them.
 * It returns ResultSets and sets updates.
 * @author Marcel Wieczorek
 */
public class Database {
String dbPath;
/**
 * @param dbPath the absolute path of the data base (either
TroubleShooter.mdb or Backend.mdb)
 */
      public Database(String dbPath) {
            this.dbPath = dbPath;
            File f = new File(this.dbPath);
            if(!f.exists()) {
                  String errorMessage =   "Die Datei " + f.getName() +
            "\nkann nicht nicht am angegebenen Ort gefunden werden.\n" +
            "Bitte geben Sie den Pfad an, an dem sich die Datei befindet";
                  JOptionPane.showMessageDialog(null, errorMessage,
f.getName() + " fehlt!", JOptionPane.ERROR_MESSAGE, Corporate.IMG32_ERROR);
```

```
                JFileChooser chooser = new JFileChooser("\\\\[SERVER]");
              FileNameExtensionFilter filter = new
FileNameExtensionFilter(
                  "MS ACCESS DB", "mdb");
              chooser.setFileFilter(filter);
              chooser.setAcceptAllFileFilterUsed(false);
              chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
              int returnVal = chooser.showOpenDialog(null);
              if(returnVal == JFileChooser.APPROVE_OPTION) {
                this.dbPath =
chooser.getSelectedFile().getAbsolutePath();
                if(this.dbPath.endsWith("TroubleShooter.mdb"))
                      Corporate.dbPathIncident = this.dbPath;
                else
                      Corporate.dbPath = this.dbPath;
              }
            }
      }
/**
 * @return Connection to the data base
 */
      public Connection getConnection()
      {
            try {
            String connectionString = "jdbc:odbc:driver={Microsoft Access
      Driver (*.mdb)};DBQ=" + this.dbPath;
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con = DriverManager.getConnection(connectionString);
            return con;
            }
            catch(SQLException e) {
            String msg = "SQLException: Fehler beim Aufbauen\nder
      Datenbankverbindung.";
                  JOptionPane.showMessageDialog(null, msg);
            }
            catch (ClassNotFoundException e) {
                  String msg = "ClassNotFoundException\n";
                  JOptionPane.showMessageDialog(null, msg + e.toString());
            }
            return null;
      }//end of getConnection
/**
 * @param con Connection that should be closed
 * @return true if closing was successful otherwise false
 */
      public boolean closeConnection(Connection con)
      {
            try {
                  if(!con.isClosed())
                        con.close();
                  return true;
            }
            catch(SQLException e) {
      String msg = "SQLException: Fehler beim Schließen der Datenbank.";
                  JOptionPane.showMessageDialog(null, msg);
            }
            return false;
      }//end of setUpdate
/**
```

```
 * @param sql SQL statement as String
 * @return true if update was successful otherwise false
 */
	public boolean setUpdate(String sql)
	{
		try {
			Connection con = this.getConnection();
			Statement statement = con.createStatement();
			statement.executeUpdate(sql);
			con.close();
			return true;
		}
		catch(SQLException e) {
	String msg = "SQLException: Fehler beim Ausführen\neines Updates.";
			JOptionPane.showMessageDialog(null, msg);
			e.printStackTrace();
		}
		return false;
	}//end of setUpdate
	/**
	 * @param sql SQL statement as String
	 * @param cols columns that should be read from the data base
	 * @return given columns
	 */
	public Vector<String> getResults(String sql, int[] cols)
	{
		try {
			Vector<String> v = new Vector<String>();
			Connection con = this.getConnection();
			Statement statement = con.createStatement();
			ResultSet rs = statement.executeQuery(sql);
			if(rs != null) {
				while(rs.next()) {
					for(int value : cols)
						v.add(rs.getString(value));
				}
			}
			con.close();
			return v;
		}
		catch(SQLException e) {
	String msg = "SQLException: Fehler beim Ausführen\neiner Abfrage.";
			JOptionPane.showMessageDialog(null, msg);
			e.printStackTrace();
		}
		return null;
	}//end of getResults
}//end of class Database
```

LoginDialog.java

```
package de.wieczorek.security;

import java.awt.Component;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Frame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

```
import java.util.Vector;

import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.UIManager;

import org.jdesktop.swingx.plaf.LookAndFeelAddons;
import org.jdesktop.swingx.plaf.windows.WindowsLookAndFeelAddons;

import de.wieczorek.Corporate;
import de.wieczorek.db.Database;
import de.wieczorek.db.DBLoader;
import de.wieczorek.gui.MainWindow;
/**
 * Main class; entry point of the program
 * @author Marcel Wieczorek
 */
public class LoginDialog extends JDialog {
    private static final long serialVersionUID = 1L;

    private String CMD_CANCEL = "cmd.cancel";
    private String CMD_LOGIN = "cmd.login";
    private String CMD_GUEST = "cmd.guest";
    private JButton loginButton = null, guestButton = null, cancelButton =
    null;
    JTextField userNameTextField;
    JPasswordField passwordField;
    JCheckBox check = new JCheckBox("Administrator");

    public LoginDialog(Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }
    /*
     * add elements to the root-panel
     */
    private void initComponents() {
        Container contents = getContentPane();
            contents.setLayout(null);
            contents.setBackground(Corporate.DARK_BLUE);

        setTitle (Corporate.VERSION + " - Login");
        setSize(375, 200);
        setLocationRelativeTo(null);
        addWindowListener(new WindowAdapter () {
            public void windowClosing(WindowEvent event) {
                windowAction(CMD_CANCEL);
            }
        });
    // add repeating images to the left of the panel
    for(int i = -5; i < this.getHeight(); i += 35) {
          JLabel img = new JLabel(Corporate.IMG32_USER);
```

```
            img.setBounds(10, i, 40, 40);
          contents.add(img);
    }
    // username
      userNameTextField = new JTextField(System.getProperty("user.name"));
        JLabel userNameLabel = new JLabel("Username");
            userNameLabel.setBounds(50, 10, 100, 20);
            userNameLabel.setFont(Corporate.BOLD15);
        contents.add(userNameLabel);

        userNameTextField.setToolTipText("Benutzername eingeben");
        userNameTextField.setBounds(50, 40, 255, 20);
        userNameTextField.setEnabled(false);
        contents.add(userNameTextField);
    // user rights
        check.setBounds(150, 10, 155, 20);
        check.setActionCommand("ADMIN");
        check.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent event) {
                    windowAction(event);
                }
            });
        contents.add(check);
    // password
    passwordField = new JPasswordField();
        JLabel passwordLabel = new JLabel("Passwort");
            passwordLabel.setBounds(50, 70, 255, 20);
            passwordLabel.setFont(Corporate.BOLD15);
        contents.add(passwordLabel);

        passwordField.setToolTipText("Passwort eingeben");
        passwordField.setBounds(50, 100, 255, 20);
        passwordField.setEchoChar('\u2022');
        passwordField.setEnabled(false);
        contents.add(passwordField);
    // buttons
    JPanel buttonPanel = createButtonPanel();
      buttonPanel.setBounds(50, 130, 255, 20);
        contents.add(buttonPanel);

      getRootPane().setDefaultButton(guestButton);
      setVisible(true);
    }
    /*
     * create panel with two buttons on it (ok, cancel)
     */
    private JPanel createButtonPanel() {
      JPanel panel = new JPanel();
            panel.setBackground(Corporate.DARK_BLUE);
        panel.setLayout(new BoxLayout(panel, 0));
        {
          //login button
            loginButton = new JButton();
              loginButton.setText("Login");
            loginButton.setToolTipText("Login");
            loginButton.setActionCommand(CMD_LOGIN);
            loginButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent event) {
                    windowAction(event);
                }
```

```
            });
            loginButton.setOpaque(true);
            loginButton.setBackground(Corporate.DARK_BLUE);
            loginButton.setFont(Corporate.BOLD12);
            loginButton.setEnabled(false);
              panel.add(loginButton);
        }
       // space
          panel.add(Box.createRigidArea(new Dimension(10, 0)));
          {
              // guest button
              guestButton = new JButton();
                guestButton.setText("Gast");
              guestButton.setActionCommand(CMD_GUEST);
              guestButton.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent event) {
                      windowAction(event);
                  }
              });
              guestButton.setBackground(Corporate.DARK_BLUE);
              guestButton.setFont(Corporate.BOLD12);
                panel.add(guestButton);
          }
       // space
          panel.add(Box.createRigidArea(new Dimension(10, 0)));
          {
              // cancel button
              cancelButton = new JButton();
                cancelButton.setText("Abbrechen");
              cancelButton.setActionCommand(CMD_CANCEL);
              cancelButton.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent event) {
                      windowAction(event);
                  }
              });
              cancelButton.setBackground(Corporate.DARK_BLUE);
              cancelButton.setFont(Corporate.BOLD12);
                panel.add(cancelButton);
          }
      // add buttons to a vector of Component
        Vector<Component> buttons = new Vector<Component>(3);
              buttons.add(cancelButton);
              buttons.add(loginButton);
              buttons.add(guestButton);
        equalizeComponentSizes(buttons);
        // simplify gc
        buttons.removeAllElements();
        return panel;
      }
      /*
       * make all elements having the same size
       */
      private void equalizeComponentSizes(Vector<Component> components) {
        Dimension maxPreferred = new Dimension(0, 0);
        Dimension thisPreferred = null;
        for (Component c : components) {
            thisPreferred = c.getPreferredSize();
            maxPreferred.width = Math.max(maxPreferred.width,
(int)thisPreferred.getWidth());
```

```
            maxPreferred.height = Math.max(maxPreferred.height,
(int)thisPreferred.getHeight());
        }
// reset preferred and maximum size since BoxLayout takes both into account
        for (Component c : components) {
              c.setPreferredSize((Dimension)maxPreferred.clone());
              c.setMaximumSize((Dimension)maxPreferred.clone());
          }
      } // end of equalizeComponentSizes()
      /*
       * a window event appeared
       */
      private void windowAction(Object actionCommand) {
        String cmd = null;
          if (actionCommand != null) {
                if (actionCommand instanceof ActionEvent)
                    cmd = ((ActionEvent)actionCommand).getActionCommand();
                else
                    cmd = actionCommand.toString();
        }
      if (cmd.equals("ADMIN")) { // administrator check box was clicked
        if (check.isSelected()) { // administrator
              userNameTextField.setEnabled(true);
              passwordField.setEnabled(true);
              loginButton.setEnabled(true);
              getRootPane().setDefaultButton(loginButton);
              guestButton.setEnabled(false);
              check.setBackground(userNameTextField.getBackground());
        }
        else { // non administrator
              userNameTextField.setEnabled(false);
              passwordField.setEnabled(false);
              guestButton.setEnabled(true);
              getRootPane().setDefaultButton(guestButton);
              loginButton.setEnabled(false);
              check.setBackground(userNameTextField.getBackground());
        }
      }
      else if (cmd.equals(CMD_CANCEL)) { // cancel button was clicked
              System.exit(0);
        }
        else if (cmd.equals(CMD_GUEST)) { // guest button was clicked
              login("nobody", "hallo");
        }
        else if (cmd.equals(CMD_LOGIN)) { // login button was clicked
              login(userNameTextField.getText(),
              String.valueOf(passwordField.getPassword()));
            }
        else {
JOptionPane.showMessageDialog(null, "Der Login ist fehlgeschlagen.\n" +
              "Bitte versuchen Sie es noch einmal.", "Login fehlgeschlagen!",
JOptionPane.ERROR_MESSAGE, Corporate.IMG32_ERROR);
              passwordField.setText("");
            }
      } // end of windowAction()
      /*
       * login user with user name and password
       */
        private void login(String user, String password) {
              String userMd5 = new MyMD5(user).encrypt();
```

```
            String passMd5 = new MyMD5(password).encrypt();
          if(isValidUser(userMd5, passMd5)) {
                  MainWindow main = new MainWindow();
                  DBLoader dbloader = new DBLoader(main);
                  Thread tMain = new Thread(main);
                  Thread tDBLoader = new Thread(dbloader);
                  // open the main window
                        tMain.start();
                        synchronized(tMain) {
                          try {
                                tMain.wait();
                                setVisible(false);
                          }
                          catch (InterruptedException e) {
                                e.printStackTrace();
                          }
                        }
                        // load data from database (start thread)
                        tDBLoader.start();
          }
      } // end of login()
      /*
       * check if user and password are valid or not
       */
    private boolean isValidUser(String user, String pass) {
            String sql = "SELECT username, passwort, status FROM tUser " +
                         "WHERE username='" + user + "' AND passwort='" +
            pass + "'";
            int cols[] = {1, 2, 3};
            Vector<String> results = new
Database(Corporate.dbPathIncident).getResults(sql, cols);
            if(results.isEmpty())
                  return false;
            else {
                  if(results.lastElement().equals("Administrator"))
                        Corporate.isAdmin = true;
                  return true;
            }
      } // end of isValidUser()
    /**
     * program's entry point
     * sets look and feel; opens login dialog
     */
    public static void main(String[] args) {
      try
            {

UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFee
l");
              LookAndFeelAddons.setAddon(new WindowsLookAndFeelAddons());
            }
            catch (Exception e)
            {
              e.printStackTrace();
            }
      new LoginDialog(null, true);
    }
} // end of class LoginDialog
```

PDFCreator.java

```
package de.wieczorek;

import java.awt.Color;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Vector;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileNameExtensionFilter;

import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.Font;
import com.lowagie.text.Image;
import com.lowagie.text.Paragraph;
import com.lowagie.text.pdf.PdfWriter;

import de.wieczorek.db.Database;
/**
 * This class is used to generate PDF files.
 * It consists of the problem's description and its solution.
 * Furthermore the is the possibility of adding images to the document.
 * @author Marcel Wieczorek
 */
public class PDFCreator {
	private String filename, description, strategy, who, id;
	File f;
	// description and strategy longer than 2000 characters
	boolean isLongDocument = false;
	/**
	 * @param filename the name of the file or its absolute path
	 * @param description the problem's description
	 * @param strategy the problem's solution
	 * @param who the person / department who made the contents of the
	 * document
	 * @param id the problem's id in the data base
	 */
	public PDFCreator(String filename, String description, String
	strategy, String who, String id) {
		this.description = description;
		this.strategy = strategy;
		this.who = who;
		this.id = id;
		f = new File(filename);
		// long document
		if(description.length() + strategy.length() > 2000)
			this.isLongDocument = true;

		if(f.exists()) {
			if(!filename.equals(Corporate.tempPath + "\\tmp.pdf")) {
				if(
				JOptionPane.showConfirmDialog(null, "Die Datei " +
				existiert bereits.\nSoll die vorhandene Datei" +
				" ersetzt werden?", "Datei überschreiben",
			JOptionPane.YES_NO_OPTION, JOptionPane.ERROR_MESSAGE)
					== JOptionPane.YES_OPTION) {
					this.filename = filename;
```

```
                        }
                }
                else
                        this.filename = filename;
        }
        else {
                if(!filename.equals(Corporate.tempPath + "\\tmp.pdf")) {
                        JFileChooser chooser = new JFileChooser(new
                        File(filename).getParent());
                        FileNameExtensionFilter filter = new
                        FileNameExtensionFilter("PDF", "pdf");
                            chooser.setFileFilter(filter);
                        if(chooser.showSaveDialog(null) ==
                        JFileChooser.APPROVE_OPTION)
                                this.filename =
                        chooser.getSelectedFile().getAbsolutePath();
                }
        }
        if(this.filename != null && !this.filename.endsWith(".pdf"))
                this.filename += ".pdf";
        else
                this.filename = Corporate.tempPath + "\\tmp.pdf";
}
/**
 * Generates a PDF file with the corporate design of the 'Freie
 * Universität Berlin'.<br>
 * The document contains the description of the problem as well as
 * the strategy to solve that
 * problem and optionally an archive of graphics which can possibly
 * be necessary to visualize
 * the manual.
 * @param title the document's headline
 */
public boolean createPDF(String title) throws IOException {
        // create a document with multiple pages and bookmarks
        Document document = new Document();
        try {
                PdfWriter writer = PdfWriter.getInstance(document, new
                FileOutputStream(this.filename));
                document.open();
                Color black = new Color(0, 0, 0);
                Color blue = new Color(0, 51, 102);;

                Font  plain11 = new Font();
                        plain11.setFamily("\"NexusSans\", \"Verdana\",
                        \"Helvetica\"");
                        plain11.setSize(11.0F);
                        plain11.setStyle(Font.NORMAL);
                        plain11.setColor(black);
                Font  bold14 = new Font();
                        bold14.setFamily("\"NexusSans\", \"Verdana\",
                        \"Helvetica\"");
                        bold14.setSize(14.0F);
                        bold14.setStyle(Font.BOLD);
                        bold14.setColor(blue);
                Font  bolditalic18 = new Font();
                        bolditalic18.setFamily("\"NexusSans\",
                        \"Verdana\", \"Helvetica\"");
                        bolditalic18.setSize(18.0F);
                        bolditalic18.setStyle(Font.BOLDITALIC);
```

```
                        bolditalic18.setColor(blue);

                Paragraph p11 = new Paragraph("Problembeschreibung",
                bold14);
                Paragraph p22 = new Paragraph("Lösungsstrategie",
                bold14);
                Paragraph p33 = new Paragraph("Kontakt", bold14);
                Paragraph p44 = new Paragraph("Verfasst am, von",
                bold14);

                String headline = "TroubleShooter - Best Practices\n" +
                title;
                      Paragraph p0 = new Paragraph(headline,
                      bolditalic18);
                      Paragraph p1 = new Paragraph(this.description,
                      plain11);
                      Paragraph p2 = new Paragraph(this.strategy,
                      plain11);
                      Paragraph p3 = new Paragraph(Corporate.CONTACT,
                      plain11);
                      Paragraph p4 = new Paragraph(this.who.substring(0,
                      10) + this.who.substring(21, this.who.length()),
                      plain11);
                      Image img = Image.getInstance(Corporate.FU_LOGO);
                            img.setAlignment(Image.ALIGN_RIGHT);
                            img.scaleAbsolute(255.0F, 67.5F);
                document.add(img);
                document.add(p0);
                document.add(p11);
                document.add(p1);

                document.add(p22);
                document.add(p2);

                if(isLongDocument)
                      document.newPage();

                document.add(p33);
                document.add(p3);

                document.add(p44);
                document.add(p4);

                String sql =
"SELECT b.pfad, b.beschreibung FROM  tProblemBild pb, tBild b WHERE" +
"pb.f_pID = " + this.id + " AND pb.f_imgID = b.imgID";
                int[] cols = {1, 2};
                Vector<String> images = new
                Database(Corporate.dbPathIncident).getResults(sql, cols);
                if(images != null && images.size() != 0) {
                      if(!isLongDocument)
                            document.newPage();
                      document.add(new Paragraph("Bilderverzeichnis",
                      bolditalic18));
                      int added = 0;
                      int item = 0;
                      if(isLongDocument)
                            added--;
                      for(String s : images) {
                            item++;
```

```
                        if(item % 2 == 0) { // description of the
                                            // image
                              document.add(new Paragraph(new
                              File(s).getName(), plain11));
                        }
                        else { // file's path
                              Image image =
                              getResizedImage(Image.getInstance(s));
                                    image.setAlignment(Image.LEFT);
                                    if(added % 2 == 0)
                                          document.newPage();
                              document.add(new Paragraph(new
                              File(s).getName(), bold14));
                              document.add(image);
                              added++;
                        }// end else
                  }// end for
            }

            writer.setFullCompression();
            document.close();
            return true;
      }
      catch (DocumentException de) {
            System.err.println(de.getMessage());
      }
      catch (IOException ioe) {
            System.err.println(ioe.getMessage());
            throw new IOException(ioe.getMessage());
      }
      return false;
  }// end of createPDF()
/**
 * Method resizes image to a maximum size of 300 X 225 pixels.
 * @param image Image that should be resized
 * @return resized image (max 300 X 225 pixels)
 */
  public Image getResizedImage(Image image) {
      // get size of the image --> set new size
      float tmpWidth = image.getWidth();
      float tmpHeight = image.getHeight();

      if(tmpWidth > 300) {
            float       factor = 300 / tmpWidth;
            tmpWidth = 300;
            tmpHeight = tmpHeight * factor;
      }
      if(tmpHeight > 225) {
            float       factor = 225 / tmpHeight;
            tmpHeight = 225;
            tmpWidth = tmpWidth * factor;
      }
      image.scaleAbsolute(tmpWidth, tmpHeight);
      return image;
  }// end of getResizedImage()
}
```

MyMD5.java

```
package de.wieczorek.security;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
/**
 * This class encrypts a given string with MD5.
 * It is used to encrypt the user name and the password.
 * @author Marcel Wieczorek
 */
public class MyMD5 {
	private String toEncrypt = "";
/**
 * @param toEncrypt the string that should be encrypted
 */
	public MyMD5(String toEncrypt) {
		this.toEncrypt = toEncrypt;
	}
/**
 * encrypts the string that is given.
 * @return encrypted string how it is written into the data base
 */
	public String encrypt() {
		String md5val = "";
		MessageDigest md = null;
		try {
			md = MessageDigest.getInstance("MD5");
		}
		catch (NoSuchAlgorithmException e) {
			System.err.println("Cannot find digest algorithm");
		}

		byte defBytes[] = toEncrypt.getBytes();
		md.reset();
		md.update(defBytes);
		byte encrypted[] = md.digest();
		StringBuffer hexString = new StringBuffer();

		for(int i = 0; i < encrypted.length; i++) {
			String hex = Integer.toHexString(0xFF & encrypted[i]);
			if (hex.length() == 1)
			hexString.append('0');
		hexString.append(hex);
		}
		md5val = hexString.toString();
	  return md5val;
	}
}
```

9.7 Abbildungsverzeichnis

9.8 Glossar

Begriff	Erklärung
Adobe Reader	Programm zum Betrachten von PDF-Dokumenten der Firma Adobe; in diesem Projekt in der Version 8.0 verwendet
Best-Practice	"bestes Verfahren" → bewährtes Konzept, Standardverfahren
Eclipse	Open-Source-Entwicklungsumgebung für eine Vielzahl von Programmiersprachen; in diesem Projekt für die Programmiersprache Java verwendet
GUI	(engl. Graphical User Interface) "grafische Benutzerschnittstelle") → Eingabemasken / Formulare
iText	Freie Java PDF-Bibliothek von Bruno Lowagie und Paulo Soares; dient dynamischer Erstellung von PDF-Dokumenten; in diesem Projekt in der Version 2.0.6 verwendet
Java SE 6	Sammlung von Java-APIs (engl. Application Programming Interface → "Schnittstelle zur Anwendungsprogrammierung") u.a. für GUI-Erstellung, Datenbankverbindung
Klasse	Dient in der objektorientierten Programmierung zur Abstrahierung von Objekten; Klasse kann Methoden und Attribute besitzen
Microsoft Access	DBMS (Datenbankmanagementsystem) der Firma Microsoft; Verwaltung von Daten
Omondo	UML-Plugin für die Entwicklungsumgebung Eclipse; dient der Erstellung von UML-Diagrammen
Package	Dient Organisation von Klassen; fasst mehrere Klassen in sich zusammen
PDF	(engl. Portable Document Format) → "übertragbares Dokumentformat"; plattformunabhängiges Dateiformat der Firma Adobe
SwinLabs Swing X	Freie Java GUI-Bibliothek; enthält Erweiterungen zu den vorhandenen Swing-Steuerelementen
UML	(engl. Unified Modelling Language) → "vereinheitlichte Modellierungssprache"; Sprache für die Modellierung von Software