Realisierungsbericht

Niklaus Hofer, Lukas Knöpfel, Kaleb Tschabold

May 30, 2011

Status	In Arbeit/In Prüfung/ **Abgeschlossen**
Projektname	Projektexplorer
Projektleiter	Lukas Knöpfel
Auftraggeber	M. Frieden, GIBB
Autoren	Kaleb Tschabold, Lukas Knöpfel, Niklaus Hofer
Verteiler	Lukas Knöpfel, Kaleb Tschabolt, Niklaus Hofer

Änderungskontrolle, Prüfung, Genehmigung

Version	Datum	Beschreibung, Bemerkung	Name oder Rolle
0.1	08.02.2011	Gesammelten Text einfügen	Kaleb Tschabold
0.9	08.02.2011	Abgabebereit	Kaleb Tschabold
0.99	May 30, 2011	Transfer nach LaTeX	Niklaus Hofer
1.0	May 30, 2011	Klassendiagram updaten, Sequenzdiagram einfügen, Rechtschreibefehler korrigieren, Benutzerhandbuch updaten	Niklaus Hofer

Definitionen und Abkürzungen

Begriff/ Abkürzung	Bedeutung
CLI	Command Line Interface
GUI	Graphical user interface
DB	Database

References

[1] Lukas Knoepfel Kaleb Tschabold Niklaus Hofer. Koneptbericht. Teil der Abgabedokumente, 2011.

Contents

1 Zweck des Dokuments

Wir hatte jetzt einige Wochen Zeit um an der Realisierungs zu arbeiten. Wir konnten jetzt unsere Programm Spezifikationen noch genauer ausarbeiten, weil wir während dem programmieren gesehen haben was noch verbessert oder ergänzt werden sollte. In diesem Dokumente sind jetzt die genauen Informationen zum Programm.

2 Technische Detailspezifikation

2.1 Innere Struktur

2.1.1 Lösungsvorschläge für die Struktur des Systemdesigns

Es gibt zwei wichtige Entscheidungen zum Systemdesign, die während der Realisierung getroffen wurden.
Die Erste betrifft, das GUI, die zweite die Art wie die Daten in der Datenbank abgelegt werden.

2.1.1.1 GUI Die Änderung am GUI betrifft die Art und Weise wie die Tags zu den Dateien zugeordnet werden.
Unser erster Einfall dazu war der, dass sich über das Kontextmenü der Dateien ein Popup öffnen liesse, in dem die Tags zugeordnet werde könnten.
Für ein Programm, dessen Hauptaufgabe gerade die Verwaltung der Tags darstellt, ist diese Art Tags Dateien zu zu ordnen aber recht aufwendig.
Die Verwaltung der Tags wird nun unabhängig von der Ansicht (Tag oder Hierarchisch) immer auf der rechten Seite des Programms angezeigt. Sobald in der linken Spalte eine Datei angewählt wird, werden deren Tags in der rechten aufgelistet.
Zudem können der Datein von dort aus weitere, bereits bestehende, Tags per Doppelklick zugeordnet oder ganz neue hinzugefügt werden, indem man deren Namen, Komma getrennt, der Liste der Tags anhängt.
Diese Lösung, für die wir uns entschieden haben ist weniger umständlich und macht die Aufgabe des Programms gleich beim Start deutlich.

2.1.1.2 Datenstruktur Die zweite wichtige Entscheidung betrifft die Art, wie die Pfade zu den Dateien in der Datenbank abgelegt werden.
Damit Dateien anhand ihrer URI in der Datenbank gefunden werden können muss die Art, wie die URI abgelegt wird immer gleich sein.
Aus Datenbank-technischer Sicht ist es von Vorteil, den Pfad zur Datei vom Dateinamen getrennt zu speichern. Abfragen nach ‘allen Dateien aus dem Verzeichnis X’ werden so deutlich einfacher aus zu führen.
Auch beim Darstellen der Dateien ist diese Art des Speicherns meist von Vorteil, da der Dateiname immer getrennt vom Pfad dargestellt wird (der Pfad oben, wie von Windows Explorer gewöhnt, und der Dateinamen unten im mittleren Panel).
Diese getrennt Speicherung hat aber zu der Frage geführt ob / (oder in Windows) am Ende der Pfadangabe mitgespeichert werden solle.
Wir entschieden uns dafür, damit Pfade ohne weiteren Aufwand vollständig zusammengesetzt werden können. Ist der ‘Dateinamen’ der Name eines neuen Verzeichnisses, so trägt auch er ein / (oder in Windows) am Ende.

2.1.2 Struktur des Systemdesigns

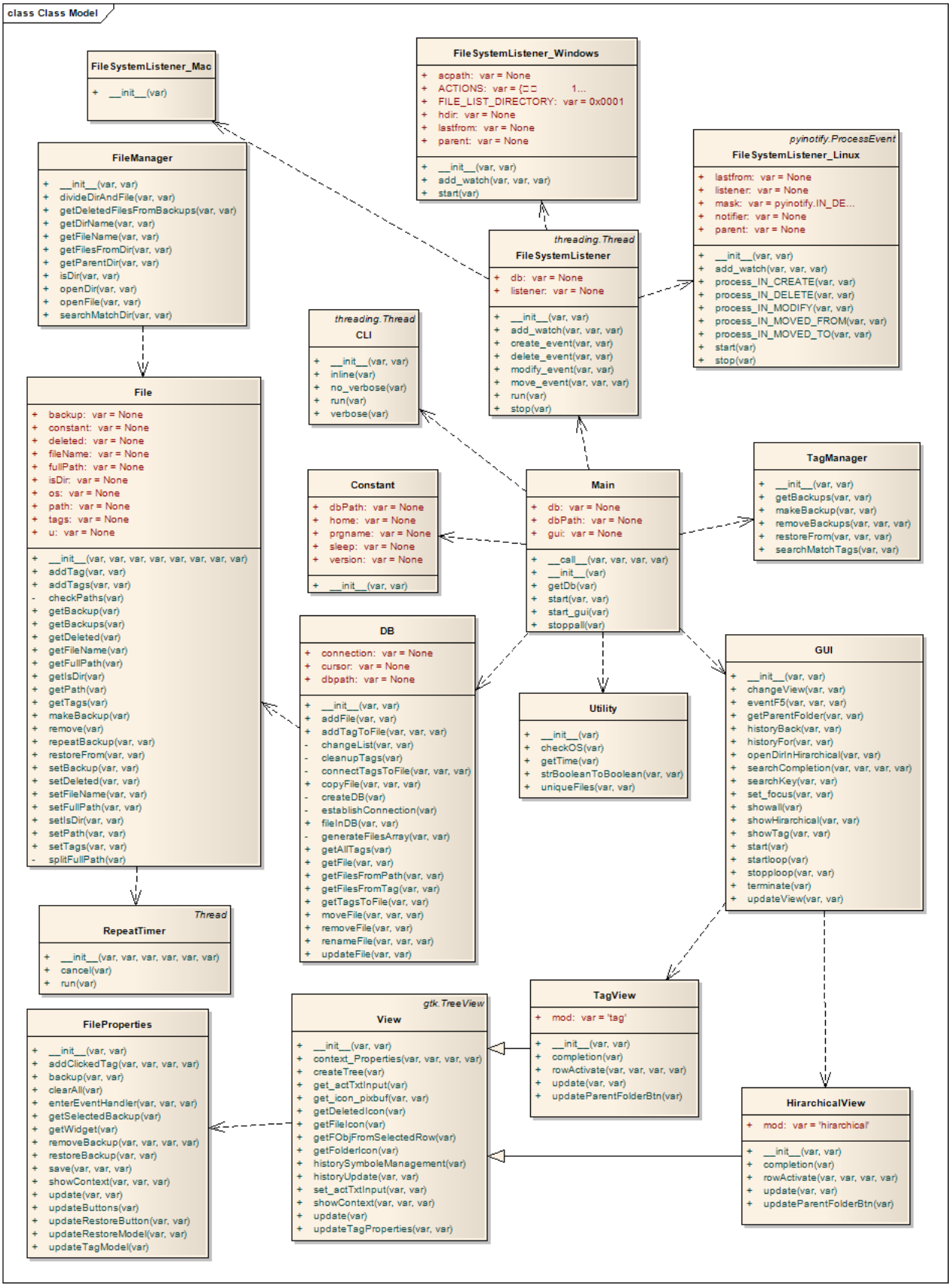

Figure 1: Klassendiagramm

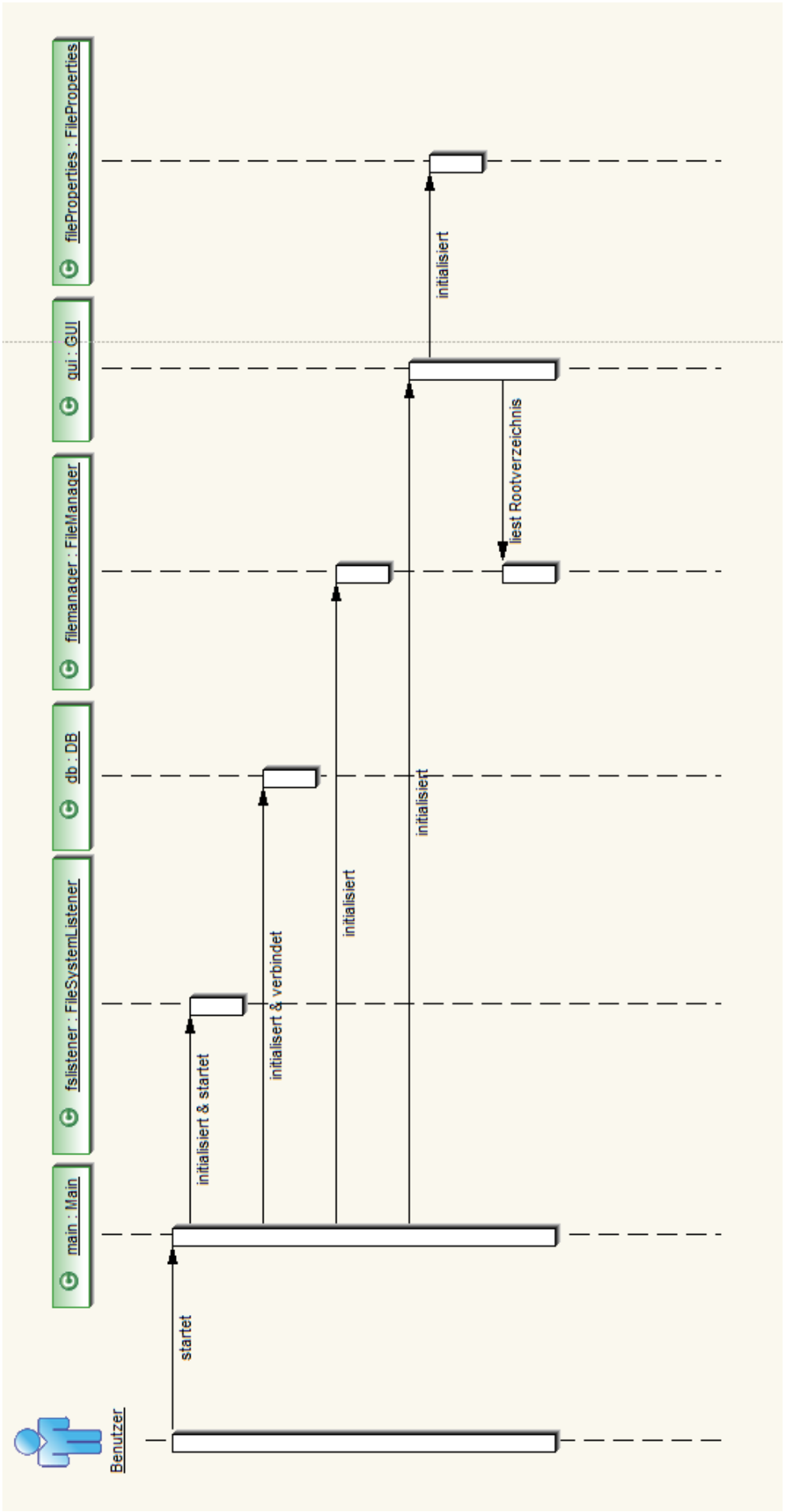

Figure 2: Sequenzdiagramm zum Verdeutlichen des Startprozesses.

2.1.3 Beschreibung der Elemente

newpage

Main Wird zum Starten des Programmes aufgerufen. Main.py instanziert alle weiteren Elemente die für das Funktionieren des Programms nötig sind und koordiniert die Kommunikation zwischen den einzelnen Elementen.

DB DB.py ist ein interface zur Datenbank. Es nimmt allen anderen Klassen die Aufgabe ab selbst SQL statements ab zu setzen und bietet stattdessen nach aussen hin verschiedene Funktionen an um Daten zu lesen oder zu schreiben.
Als Datenbank-backend spricht DB.py SQLite3 an.

Utility Enthält verschiedene nützliche Methoden die immer mal wieder von einzelnen Teilen des Programmes benötigt werden.

CLI Dient als Command-line-interface für das Program. Es nimm über die Kommandozeile beim Aufruf verschiedene Befehle entgegen die es dann asführt.

TagManager Der Tag Manger ist für kleine Tag Verwaltungsaufgaben zuständig.

FileManager Über den FileManager wird auf das Dateisystem zugegriffen. Hier wird aus jeder Datei aus dem File System ein File Objekt erstellt.

FileSystemListener Registriert beim Kernel Listener für zu überwachende Ordner. Wird in diesen Ordnern eine Operation ausgeführt (wie das Verschieben, Löschen, Umbenennen oder Erstellen einer Datei), so wird der FileSystemListener vom Kernel darüber in Kenntnis gesetzt, woraufhin er wiederum die nötigen Aktionen auslöst um die Datenbank auf dem aktuellen Stand zu halten.
Dies soll dazu beitragen, dass möglichst selten Dateien angezeigt werden, die auf Dateisystem-Ebene nicht existieren.

GUI Ist für die grafische Darstellung des Programms mittels GTK zuständig. Das GUI ist in der Lage je nach Bedarf eine andere ‘View’ darzustellen. Direkt nach dem Programmstart wird HierachicalView dargestellt. Im Betrieb kann jederzeit zwischen ‘TagView’ und ‘HirachicalView’ umgeschaltet werden.
Dazu kann GUI, per Polymorphismus, eine Klasse aufnehmen die von ‘View’ erbt.

TagView Enthält die Darstellung der Tag-Ansicht und wird von ‘GUI’ bei Bedarf geladen.

HierarchicalView Enthält die Darstellung der hierachischen Ansicht und wird von ‘GUI’ bei Bedarf geladen.

View Mutterklasse von GUI.

File Repräsentiert eine Datei und wird benutzt um Iformationen üeber Dateine zwischen den Elementen des Programms auszutauschen. Fuer mehr Informationen siehe 2.3.2.

2.2 Schnittstellendefinition

1. Interne Schnittstellen
   a) Intern ist die Kommunikation mit der Datenbank sehr wichtig.
      i. Die Datenbank-Schnittstelle ist in DB.py implementiert.
      ii. Die Schnittstelle nimmt in den meisten Fällen Objekte vom Typ File(.py). In anderen auch Strings.
      iii. Welche Funktionen in der Schnittstelle genau definiert sind, kann dem Klassendiagramm entnommen werden.
      Wie die einzelnen Methoden aufzurufen sind und was sie genau tun kann, ist jeweils im Methodenkommentar ersichtlich.
2. Externe Schnittstellen
   a) Für den normalen Gebrauch haben wie das GUI. Beim GUI wird Wert auf das einfache verwalten von Tags und Dateien gelegt.
      i. Das GUI bietet zwei Modi, einer der den herkömlichen Dateimanagern mit hierarchischer Ansicht entspricht und
      ii. Einen Tagmodus, in dem sich Dateien anhand deren Tags durchsuchen und ordnen lassen.

b) Für scripting oder für solche Systeme ohne grafische Ausgabe haben wir eine CLI Version. Es wird besonders Wert auf das einfache Aufrufen von anderen Programmen (Scripts) gelegt. (nicht implementiert)

   i. Die Kommandos sollen in der Bedienung weitgehend mit dem standard Unix-Tools kompatibel

2.3 Datenmodell

2.3.1 Datenbank

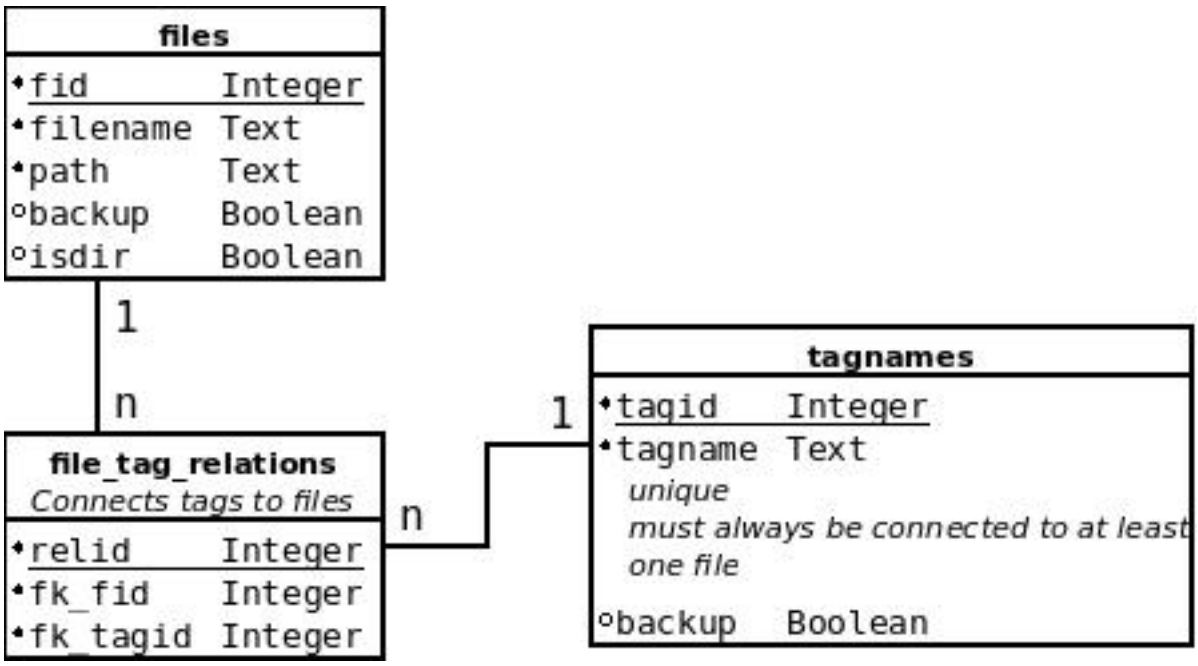

Figure 3: Datenbankschema

Die Datenbank lässt sich über DB.py ansprechen, siehe Schnittstellendefinition für mehr Informationen.
Welche Felder genau welche Information enthalten ist eim folgenden Abschnitt erläutert.

2.3.2 File-object

Informationen über einzelne Dateien werden innerhalb des Programms mit Hilfe des File objects festgehalten und ausgetauscht.
Das File Object hat für alle Variablen Getter und Setter, sie alle können aber auch im Konstruktor angegeben werden.
Hier eine Auflistung und erläuterung zu den einzelnen Variablen der File Klasse:

Name	Type	Erläuterungen	Entsprechung in der Datenbank
fileName	String	Hält den Namen der Datei. Ist der 'Name' der eines Verzeichnisses, so endet er mit / (oder \in Windows)	files.filename
path	String	Hält den Pfad zu dem Verzeichnis in dem die Datei liegt. Endet mit / (oder \in Windows)	files.path
isDir	Boolean	Besagt, ob das Objekt eine Datei oder ein Verzeichnis repräsentiert.	files.isdir
backup	Boolean	Besagt, ob Backups der Datei angelegt werden sollen.	files.backup
fullPath	String	Der ganze Pfad zur Datei inklusive deren Namen. Wird der Fullpath an ein File object übergeben, so wir dieser mittels Regex zerlegt und die Werte in fileName und path abgelegt.	files.path + files.filename
tags	list[String]	Entälte eine Liste aller Tags die der Datei zugeordnet sind als Strings.	Die Werte in Tagnames werden mit file_tag_relation mit denen in files verknüpft.

2.4 Sicherheit

Die Datenbank liegt auf dem lokaeln Dateisystem und stellt nach Aussen (über das Netzwerk) keine Schnittstelle zur Verfügung.
Fuer jeden Nutzer der Software wird in dessen Home-Verzeichnis (unter Unix also /home/username/) ein Ordner ./project-browser angelegt, in dem sich die Datenbank-Datei befindet.
Die Datenbank ist also durch die Zugriffsberechtigung des Dateisystems geschützt und fügt sich somit natlos in bestehende Sicherheits- und Datenschutzkonzepte ein.

2.5 Anforderungszuordnung

1. Main
2. DB
3. Utility
4. CLI
5. TagManager
6. FileManager
7. FileSystemListener
8. GUI
9. TagView
10. HierarchicalView
11. View
12. File

Nr.	Anforderungen	1	2	3	4	5	6	7	8	9	10	11	12
1	Das Programm starten												
2	GUI vorhanden												
3	CLI vorhanden												
4	Tag hinzufügen												
5	Datei auswählen												
6	Versionierung												
7	GUI: Versionierung für Tags												
8	Dateiänderungen werden erkannt												
9	Löschen wird erkannt												
10	Neue Dateien werden erkannt												
11	Hierarchische Anzeige												
12	Tag Anzeige												

3 Systemdokumentation

3.1 Inline-Dokumentation

Siehe Ende Dokument!

3.2 Benutzerhandbuch

3.2.1 Systemübersicht

3.2.1.1 Aufgabengebiet des Programms Das Programm Project-Explorer ist dazu da bei der Verwaltung von Dateien zu helfen.
In einer gewöhnlichen Arbeitsumgebung werden Dateien in einem hierarchischen System aus Ordnern abgelegt. Doch mit der Datenmenge steigt auch die Komplexität dieser hierarchischen Strukturen und besonders Dateien die selten verwendet werden können schwierig aufzufinden sein.
Programme zum Verwalten von Musik- und Bilddateien bieten deshalb schon seit vielen Jahren zusätzlich sogenannte Metadaten an, um weitere Informationen zu der Datei (z.B. Album, Artist, Aufnahmejahr, . . . bei Musikdateien) zu speichern mit deren Hilfe sich diese dann einfacher wiederfinden lassen.
Projekt-Explorer hat das Ziel ähnliches anzubieten - aber nicht auf eine Art von Dateien beschränkt, sonder für jede Datei, die auf dem lokalen Rechner liegt.
Der Nutzer kann dazu den einzelnen Dateien Tags zuordnen, anhand derer er die Dateien später wieder finden kann.
Die Tags können selbst festgelegt und vergeben werden.

3.2.1.2 Programmoberfläche Die Oberfläche des Programms lässt sich grob in vier Bereiche aufteilen.
Der Erste (4. auf dem Bild), ist die Multibar. Hier kann der Pfad zu einem Verzeichnis eingeben werden, oder, in der Tag-Ansicht, ein Tag nachdem man sucht.
Der Zweite Bereich (1.,2. und 3. auf dem Bild) ist der Navigationsbereich. Hier kann in der Ordnerstruktur navigiertt und zwischen den Ansichten umgeschaltet werden Im dritten Bereich (5. auf dem Bild), werden die Dateien angezeigt. Rechts der Dateien sind deren Tags zu sehen.
Der nächste Bereich (6. auf dem Bild) ist der Wichtigste. Hier werden die verfügbaren Tags angezeigt und hier können die Tags den Dateien zugeordnet werden. Im mit dem Button "Sichern" (7.) kann der selektierte Ordner oder selektierte Datei gesichert werden.

3.2.1.3 Anmerkungen zur korrekten Verwendung unter dem Aspekt der Sicherheit Den Autoren ist es wichtig an dieser Stelle einige Bemerkungen zur Datensicherheit zu machen. Grundsätzlich gilt für die Daten die im Projekt-Explorer erfasst werden dasselbe wie für alle anderen Daten auf dem Computer auch. Die Informationen werden in einem persönlichen Verzeichnis des Nutzers abgelegt.
Per Standard ist dieses vor dem Zugriff durch andere Benutzer geschützt. Es kann aber in einzelnen Fällen sein, dass diese Einstellung vom Systemadmnistrator verändert worden ist. Es ist zudem zu beachten, dass der Systemadministrator zu jeder Zeit Zugriff auf die Daten hat. Werden im Program sehr persönliche Informatione abgelegt oder handelt es sich beim verwendeten Gerät um einen portablen Computer (Ein Laptop oder gar ein Handset) so sollte die lokale Festplatte verschlüsselt werden, damit im Falle eines Verlustes des Gerätes kein unberechtigeter Zugriff auf die Dateien geschehen kann.

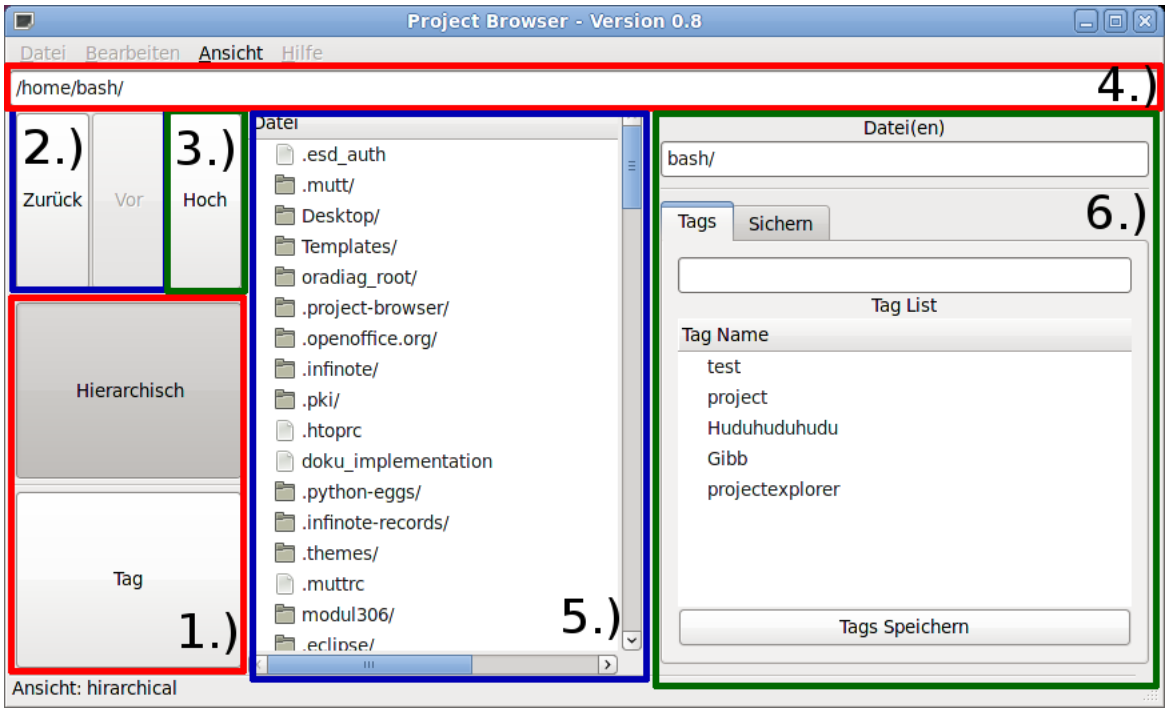

Figure 4: Programmoberfläche

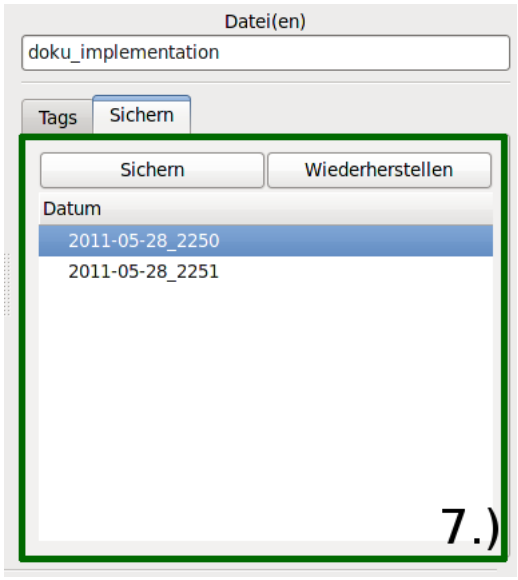

Figure 5: Programmoberfläche zum Verwalten der Backups.

3.2.2 Anwenderfunktionalität

In dieser Kategorie werden wir einige typische Anwendungen für Projekt-Explorer beschreiben. Die Erläuterungen werden sich zumeiste auf das Bild im Abschnitt 'Programmoberfläche' beziehen. Nummern aus dem Bild werden im Format #1.) angegeben. Wir werden folgende Szenarien beschreiben:

- Wechseln der Ansicht zwischen Tag- und Hierarchischer Ansicht.
- Wechseln in ein anderes Verzeichnis und Nutzen der Autovervollständigung.
- Navigieren in der History.
- Öffnen von Dateien.
- Tags zu Dateien zuordnen.
- Tags von Dateien entfernen.

- Dateien eines Tags anzeigen.
- Fehlermeldungen
- Sichern
- wiederherstellen

3.2.2.1 Ansicht wechseln Projekt-Explorer hat zwei verschiedene Ansichten, nämlich ‘Hierarchisch’ und ‘Tags’. Beim Wechseln zwischen den Ansichten ändert sich die Ansicht #5.).
Die Ansicht kann entweder üeber die grossen Knöpfe in #1.) gewechselt werden oder in der Menüleiste unter dem Punkt “Ansicht".

3.2.2.2 Verzeichnis wechseln Um in der hierarchischen Ansicht das Verzeichnis zu wechseln, gibt man das gewünschte Zielverzeichnis in #4.) ein. Während der Eingabe erschein ein Drop-Down mit Verzeichnisvorschlägen das während des Tippens laufen aktualisiert wird. In diesem Drop-Down kann ein Verzeichnis mittels der Pfeiltasten auf der Tastatur und Enter, oder mit der Maus ausgewählt werden.
Um in das Verzeichnis oberhalt zu wechseln, kann der Knopf “UP" in #3.) geklickt werden. Um in ein Unterverzeichnis zu öffnen, fürht man einen Doppelklick auf den entsprchenden Eintrag in #5.) aus.

3.2.2.3 Hisotry Projekt-Explorer merkt sich in welchem Verzeichnis man zuletzt war. Diese Werte werden zur Laufzeit in der History gespeichert. Über die Buttons “Back" und “Forward" kann in dieser History navigiert werden. Die History von Projekt-Explorer entspricht vom Konzept her ziemlich genau der entsprechenden Funktionalität von Webbrowsern.
Beim Schliessen der Applikation geht die History verloren.

3.2.2.4 Dateien Öffnen Dateien werden durch einen Doppelklick geöffnet. Sie werden mit dem Program geöffnet, das auf dem Betriebssystem als Standardapplikation fr̈ den entsprechenden Dateityp festgelegt ist. Ist für den Dateityp keine Standard-Applikation festgelegt, so kann die Datei auch nicht geöffnet werden.
Wichtig! Klickt man in der Tagview auf einen Ordner, so wechselt die Ansicht automatisch zur hierarchischen Ansicht zurück.

3.2.2.5 Tags hinzufügen Um einer Datei Tags hinzu zu fügen muss diese erst in #5.) angewählt werden. Das geschieht üeber einmaliges Klicken auf die Datei. Anschliessend werden in #6.) Informationen zu der Datei angezeigt.
Im oberen Feld der Name, im unteren Feld die Tags. Die einzelnen Tags sind jeweils durch ein Komma (“,") getrennt.
Im unteren Bereich von #6.) sind alle bereits verwendeten Tags ersichtlich. Um der Datei eines dieser Tags hinzu zu fügen, doppelklickt man den Eintrag, worfaufhin dieser in der Liste oben erscheint. Alternativ dazu, kann das Tag auch von Hand in der Liste geschrieben werden.
Um die Änderung zu übernehmen klickt man auf den Knopf ‘speichern’, der sich ganz unten in #6.) befindet.
Will man einen gänzlich neuen Tag anfügen, so muss dieser von Hand in die Liste der Tags geschrieben werden. Natürlich muss er durch ein Komma von den anderen Tags getrennt sein. Nach dem speichern, erscheint der neue Tag auch in der Liste unten.
Wichtig! Es können keine Tags bestehen, die keiner Datei zugeordnet sind!

3.2.2.6 Tags entfernen Um ein Tag von einer Datei zu entfernen, löscht man den Eitrag aus der Liste ‘Tags’ in #6.) und drückt speichern.
Ist der Tag mit keiner weiteren Datei verbunden, so verschwindet er aus der Äuswahlliste in #6.).

3.2.2.7 Dateien anhand der Tags durchsuchen Um Dateien eines Tags anzuzeigen wechselt man erst in die Tagview. Alle Tags werden dort übereinander angezeigt. Durch das Doppelklicken eines Eintrages werden alle Dateien dieses Tags sichtbar.
Alternativ kann der Tagname auch in #4.) eingegeben werden. Während der Eingabe werden Tags vorgeschlagen, die gleich geschrieben werden und im Drop-Down zur Auswahl gestellt.

3.2.2.8 Fehlermeldungen Im Normalbetrieb werden keine Fehlermeldungen ausgegeben. Um Trotzdem Informationen zu auftretenden Fehlern zu erhalten muss das Programm aus dem Terminal gestartet werden. Auftretende Fehler werden dann dort ausgegeben.

3.2.2.9 Sichern Zum Sichern von Dateien, Ordnern oder aller Dateien eines Tags, wählt man einen entsprechenden Eintrag an und wechselt dann in der rechten Spalte (#6.) auf das Tab sichern. In #7. klickt man nun den Button Sichern, um ein Backup zu erstellen. Wenn eine Tag-Gruppe selektiert ist werden alle Dateien und Ordner mit diesem Tag gesichert. Die Sicherung wird in den gleichen Ordner gemacht, in dem der zu sicherne Ordner oder zu sichernde Datei liegt. Es wird ein Unterordner mit dem Namen '.pb-backup' erstellt dort wird die Sicherungen nach Datum abgelegt.

3.2.2.10 Wiederherstellen Existiert ein Backup einer Datei, so kann dieses wiederhergestellt werden - selbst dann noch, wenn die Datei gelöscht worden ist. Dazu wählt man die Datei an (Wurde die Datei auf dem Filesystem gelöscht, so ist sie entsprechend markiert, kann aber dennnoch ausgewählt werden), und wechselt in #6. auf das Tab Sichern. Im unteren Bereich von #7. werden nun alle vorhandenen Backups angezeigt. Zum wiederherstellen wird einfach das gewünschte selektiert und der Button Wiederherstellen betätigt.

3.3 Supporthandbuch

3.3.1 Massnahmen bei Benutzerproblemen

- Die Aufgabenstellung von Projekt-Explorer ist lediglich die Verwaltung der Tags. Obwohl im Programm selbst sehr wohl Dateien angezeigt werden, können dateien weder mit copy 'nd paste noch per drag 'nd drop in den Ordnern bewegt werden. Das Programm selbst bietet auch keine Möglichkeit zum Umbenennen oder Löschen von Dateien.
- Es ist darauf zu achten, dass die Grösse der beiden rechten Pannels von Projekt-Explorer frei verstellbar ist, indem man die 'Trennlinie' mit der Maus weiter nach links oder rechts verschiebt. Schiebt man diese Linie zu weit in die eine Richtung kann es vorkommen, dass eine der Spalten komplett verschwindet. Das lässt sich ganz einfach beheben, indem man die Linie wieder verschiebt.
- Klickt man einen der Buttons zum Umschalten der Ansicht auf der linken Seite des Programmfensters mehrmals hintereinander, so ändert die Ansicht wiederholt.

3.3.2 Massnahmen bei technischen Problemen

- Lässt sich das Programm nicht starten, so ist sicher zu stellen, dass sowohl Ptyhon 2.7.X als auch das mitgelieferte PyGTK korrekt installiert sind.
- Werden Daten falsch angezeigt, so ist zu befürchten, dass in der Datenbank korrupte Daten liegen. Es gibt zwei Wege dies zu beheben:
  - Man kann die Datenbank manuell mit einem beliebigen SQLite Programm öffnen, die korrupten Einträge suchen und Löschen.
    Diese Methode ist vorzuziehen, da so alle Daten erhalten bleiben.
    Der Pfad zur Datenbank ist /.project-explorer/db.
  - Die zweite Lösung besteht darin, die Datenbank einfach zu Löschen. Beim nächsten Start erstellt das Programm dann automatisch eine neue, leere Datenbank.
    Bei dieser Methode gehen alle Daten verloren!
    Der Pfad zur Datenbank ist /.project-explorer/db

3.3.3 Anhang zum Supporthandbuch

4 Systemtest

4.1 Testspezifikation

4.1.1 Kritikalität der Funktionseinheit

Unsere Struktur des Programmes erlaubt Fehler in einigen Modulen. Je nachdem wo der Fehler auftritt stürzt nur ein Modul oder gerade das ganze Programm ab. Kritisch ist das grafik Modul. Ohne dieses kann der User das Programm nicht mehr benutzen. Nicht so wichtig ist zum Beispiel der Filesystem-listener. Wenn der abstürzt bekommt das Programm keine Filesystem-events mehr, was nicht sehr schlimm ist.

4.1.2 Testanforderungen

Die Tests sollen zuerst unter optimalen Bedingungen ausgeführt werden um die Funktionalität zu prüfen.
Dann soll jede Funktion noch einem "Schlechtwettertest" unterzogen werden. So wird die Qualität getestet.

4.1.3 Testverfahren

Für die "Gutwettertests" wird vor dem Test das System in seinen Ursprungszustand zurückgesetzt und es werden nur valide Daten eingegeben. Bei den "Schlechtwettertests" werden dem Programm möglichst viele Steine in den Weg gelegt.

4.1.4 Testkriterien

Abdeckungsgrad: Die Tests umfassen alle Funktionen die das GUI anbietet.
Checklisten: siehe Tabelle unten
Endkriterien: Der Test ist erfolgreich verlaufen wenn das Prgramm nicht abstürzt und die erwarteten modifikation an GUI, DB oder Filesystem durchgeführt hat.

4.1.5 Testfälle

Nr.	Afo-Nr.	Anwendungsfall	Ausgangs- situation	Eingabedaten	erwartetes Ergebnis	Bemerkungen, Prüfergebnis
0	1	Normale Benutzung	Programm läuft nicht	Programm starten	Programm läuft	Wichtigster Test!
1	4	Anwender fügt einer Datei ein Tag hinzu.	Die Datei /.bashrc hat noch kein Tag.	Der Nutzer wechselt in das Verzeichnis , wählt die Datei .bashrc an und fügt ihr im rechten Panel das Tag configfile hinzu.	Nach einem Neustart des Programmes und dem erneuten Selektieren der Datei wird das Tag in der Tags-Liste angezeigt.	OK
2	2	Normale Benutzung	Programm wurde gestartet	keine	Fenster mit Menu, Buttons und Eingabefeldern erscheint.	
3	-	Erstellen von Tags mit Sonderzeichen.	Das Tag /' existiert noch nicht.	Eine beliebige Datei wird selektiert, der Name des Tags wird in der Tags-Liste eingeegeben und durch 'speichern' festgehalten. Danach wird das Tag einer weiteren Datei hinzugefügt.	Nach dem ersten Speichern, erscheint das Tag in der Liste aller Tags unten auf der rechten Seite des Programmes. Wird es einer zweiten Datei hnzugefügt, so wird es NICHT dupliziert.	OK

4	3	Benutzung auf nicht grafischen System	Programm wurde gestartet	keine	Command Prompt erscheint	
5	5	Normale Benutzung	Programm wurde erfolgreich gestartet	User wählt eine Datei an	Tags werden angezeigt	
6	-	Ein Verzeichnis das eine Datei mit Sonderzeichen im Namen enthält wird geöffnet und der Datei ein Tag hinzugefügt.	Die Datei /".txt hat keine Tags zugeordnet.	Die Datei /".txt wird angesteuert und ihr ein beliebiges Tag hinzugefügt.	Der Browser ist in der Lage das Verzeichnis mit der Datei zu öffnen. Das Tag wird der Datei erfolgrech zugewiesen und bleibt erhalten.	OK
7	6	ein Projekt soll Versioniert werden	Projektdaten liegen auf dem Filesystem	Der User aktiviert die Versionierung eines Tags	Die Dateien werden bei Veränderung und/oder nach einer Zeitlichen verzögerung kopiert.	
8	-	Wechseln der Ansicht.	Nach dem Programmstart wir die hierarchische Ansicht dargestellt.	Der Nutzer wechselt durch einen Klick auf den Button Tag (links unten im Programm), in die Tagansicht. Danach wechselt er über das Ansichts-Menü wieder zurück zur hierarchischen Ansicht.	Nach dem Klick auf den 'Tag'-Button wechselt das Programm zur Tag-Ansicht. Bei der über das Menü ausgelösten Aktion wieder zurück zur hierarchischen Ansicht.	OK
9	8	Nach veränderung einer Datei soll von dieser ein Backup angelegt werden	Die Datei liegt auf dem Filesystem und der FileSystemListener wurde gestartet.	Der User verändert die Datei	Der FileSystem-Listener sendet ein Änderungsevent	
10	9	Der User löscht eine Datei	Die Datei wurde in der Datenbank erfasst	Der User löscht eine Datei	Der FileSystem-Listener erkennt die Löschaktion und sendet ein Event an die Datenbank	
11	10	Der User legt eine Datei an	Der FileSystemListener überwacht das Verzeichnis	Der User legt eine Datei an	Der FileSystem-Listener erkennt die neue Datei und sendet ein Event an das GUI damit dieses die Datei anzeigt	

12	-	Der User möchte ein Backup anlegen	Dateien liegen auf dem Filesystem	Der User wählt die option Sichern an		OK

4.2 Testprozedur

Die Tests werden alle von Hand durchgeführt (keine Testprogramme). Alle Testfälle wurden in der Tabelle (oben) genügend genau definiert.

4.2.1 Vorbereitung

4.2.1.1 Voraussetzungen

- Computer
- OS (Windows/Linux)
- python 2.7
- pygtk

4.2.1.2 Konfiguration

1. Computer starten
2. Main.py starten
3. Test's durchführen

4.2.2 Durchführung

Die Tests, wie oben beschrieben, müssen manuell von Hand durchgeführt werden.
Zu diesem Zweck haben wir die Tests auf die verschiedenen Mitglieder des Teams aufgeteilt. Die Tests wurden jeweils unter Windows, Linux (Ubuntu 10.04 und Fedora 15 Beta) und Mac OS X durchgeführt.
Die Ordnerstruktur mit den speziell benannten Ordnern wurde zuerste erstellt und auf allen Systemen einheitlich gehalten. Sie ist aber für die Replikation der Tests nicht zwingend notwendig.

4.2.3 Nachbearbeitung

Die Resultate der Test's werden im Realisierungsbericht unter dem Punkt Testresultate festgehalten.

4.3 Testprotokoll

TO	Resultat	Auswertung
0	Programm läuft	Test bestanden
1		Test bestanden
2	GUI erscheint	Test bestanden
3		Test bestanden
4	Das Programm stürzt ab, weil es gtk nicht installiert hat.	Wir hatten zu wenig Zeit um eine CLI Version umzusetzen
5	Tags angezeigt	
6		Test bestanden
7	Der User sucht vergebens nach einer Versionierungsoption	Wir hatten zu wenig Zeit um eine Versionierung umzusetzen. Wir haben aber eine Backupfunktion, die ein Backup einer Datei/Ordner mit Zeitstempel erstellt.
8		Test bestanden
9	Die Datei wird nicht Versioniert	Da wir keine Versionierung haben ist auch dieser Test fehlgeschlagen. Die Dateiänderung konnte aber erkannt werden
10	Datei wurde aus der Datenbank gelöscht	Test bestanden
11	Die Datei erscheint	Test bestanden
12	Die Datei/Ordner wurden in das Unterverzeichnis .pb_backup und dort in einen Ordner mit dem aktuellen Zeitstempel verschoben.	Test bestanden

4.3.1 Testobjekt

Tester
Lukas Knöpfel, Kaleb Tschabold, Niklaus Hofer

Ort
GIBB, Bern

Datum und Zeit
2011.05.10, 16:00

4.3.2 Testresultate

Wenn in der Tabelle kein Kommentar ist, war der Test erfolgreich. Wenn der Test fehlgeschlage ist, dann ist das in der Zeile des betreffenden Test als Bemerkung angefügt.

4.3.3 Testauswerung

Unter Mac hat der FileSystemListener nicht funktioniert, weil er bis zum jetztigen Zeitpunkt nicht implementiert ist. Unter Windows sind auch einige Fehler aufgetreten, weil noch nicht alles vollständig implementiert ist.

5 Mittelbedarf

Siehe Konzeptbericht.[1] Neu dazu gekommen ist das Program 'Enterprise Architect' um ein UML Klassendiagramm zu erstellen.

6 Planung und Organisation

Siehe Konzeptbericht[1]

7 Wirtschaftlichkeit

Siehe Konzeptbericht[1]

8 Konsequenzen

Siehe Konzeptbericht[1]

9 Antrag auf Freigabe der nächsten Projektphase

Wir bitten sie uns die Freigabe der nächsten Projektphase freizugeben

10 Sourcecode

10.1 Main.py

```
1  #!/usr/bin/python
2
3  #File:                  Main.py
4  #Description:           Diese Datei ist die Start Datei fuer unser Projekt. Hier werden alle
       wichtigen Referenzen auf unsere Klassen erstellt.
5  #Author:                Kaleb Tschabold
6  #Creation Date:         29.3.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             29.3.2011       Grundfunktionalitaeten werden erstellt
10 #                       0.1             18.3.2011       Wichtigste Prozesse starten
11 #                       0.1             19.3.2011       GUI starten in eine seperate Function
       gepackt, damit das CLI auch das GUI starten kann.
12 #Link:
13 #PyInstaller:
14 #       http://www.marcogabriel.com/blog/archives/343-Python-Scripte-mit-PyInstaller-als-.exe-
       verteilen.html
15
16
17 #Unserer Klassen
18 from CLI import *
19 from DB import *
20 from FileManager import *
21 from TagManager import *
22 from FileSystemListener import *
23 from GUI import *
24 from Utility import *
25 from File import *
26 from Constant import *
27
28 #Andere Klassen
29 import sys
30 import os.path
31
32 class Main():
33         db       = None
34         dbPath   = None
35         gui      = None
36         def __init__(self):
37                 pass
38
39         def start(self,modus):
40                 self.mod = modus
41                 self.filemanager = FileManager(self)
42                 self.tagmanager = TagManager(self)
43                 self.u = Utility()
44                 self.c = Constant(self)
45                 print "path = "+ self.c.dbPath
46                 if not os.path.exists(self.c.dbPath):
47                         print "creating path"
48                         os.makedirs(self.c.dbPath)
```

```
49                  self.db = DB(self.c.dbPath+"db")
50
51                  #Array mit allen Thread. Wird gebraucht, dass diese beim beenden des
                        Programmes alle richtig beendet werden
52                  self.t = []
53
54                  #FileSystemListener in einem eigenen Thread
55                  self.fslistener = FileSystemListener(self)
56                  self.fslistener.daemon = True
57                  self.t.append(self.fslistener)
58                  self.fslistener.add_watch(os.path.abspath(self.c.home), True)
59                  self.fslistener.start()
60
61                  if modus == 'cli':
62                          #CLI in einem eigenen Thread
63                          self.cli = CLI(self)
64                          #dieser Thread ist dem Main unterordnet. So kann man mit einem
                                KeyInterrupt den Thread beenden
65                          #self.cli.daemon = True
66                          self.t.append(self.cli)
67                          self.cli.start()
68                  else:
69                          self.start_gui()
70
71          def start_gui(self):
72                  #GUI in einem eigenen Thread
73                  self.gui = GUI(self)
74                  #self.gui.daemon = True
75                  #self.t.append(self.gui)
76                  self.gui.start()
77
78          def stoppall(self):
79                  #Threads beenden
80                  #self.fslistener.stop()
81                  for t in self.t:
82                          print('close: '+str(t))
83                          try:
84                                  t.stop()
85                          except RuntimeError:
86                                  pass
87
88          def getDb(self):
89                  return self.db
90
91          #Wird gebraucht, dass diese Klasse wie ein Objekt aufgerufen werden kann
92          def __call__(self,a,b,c):
93                  pass
94
95  #PROGRAMM START
96  if __name__ == "__main__":
97          main = Main()
98          try:
99                  try:
100                         cmd = len(sys.argv[1])
101                         if cmd > 0:
102                                 main.start('cli')
103                         else:
104                                 main.start('gui')
105                 except:
106                                 main.start('gui')
107         except (KeyboardInterrupt,SystemExit):
108                 print('Programm geschlossen')
109                 main.stoppall()
```

10.2 DB.py

```
1  #!/usr/bin/python
2
3  #File:                  DB.py
4  #Description:           Zugriff auf die DB
5  #Author:                Kaleb Tschabold
6  #Creation Date:         29.3.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             29.03.2011      Grundfunktionalitaeten werden erstellt
10 #                       0.2             21.04.2011      Added a few methods to read some data
       from the DB
11 #                       0.3             23.04.2011      Will now autocreate tables if
       nonexistance
12
13 import sqlite3
14 import os.path
15 import File
16
17 class DB:
18         #TODO setBackup
19         connection      = None
20         cursor          = None
21         dbpath          = None
22         def __init__(self, dbpath):
23                 self.dbpath     = dbpath
24                 self.__establishConnection()
25                 self.connection.text_factory = str
26
27         def __establishConnection(self):
28                 #print self.dbpath
29                 self.connection = sqlite3.connect(self.dbpath,check_same_thread = False)
30                 self.cursor     = self.connection.cursor()
31                 print "connection established"
32                 #Check wether the tables in the DB exist. If they don't, we'll create 'em
33                 self.cursor.execute("SELECT name FROM sqlite_master WHERE type='table' AND
                       name='files'")
34                 if len(self.cursor.fetchall()) != 1:
35                         print "DB's empty. Creating tables"
36                         self.__createDB()
37
38         def __createDB(self):
39                 self.cursor.execute("CREATE TABLE files(fid INTEGER PRIMARY KEY, filename TEXT
                       , path TEXT, backup BOOLEAN, isdir BOOLEAN)")
40                 self.cursor.execute("CREATE TABLE tagnames(tagid INTEGER PRIMARY KEY, tagname
                       TEXT UNIQUE , backup BOOLEAN)")
41                 self.cursor.execute("CREATE TABLE file_tag_relations(relid INTEGER PRIMARY KEY
                       , fk_fid INTEGER, fk_tagid INTEGER)")
42                 self.connection.commit()
43
44         def __changeList(self, li):
45                 ret_value       = []
46                 for row in li:
47                         ret_value.append(row[0])
48                 return ret_value
49
50         def __cleanupTags(self):
51                 """Delete old tags from database. This is tags that are not connected to any
                       file"""
52                 delTagQuery = "DELETE FROM tagnames WHERE tagnames.tagid NOT IN (SELECT
                       file_tag_relations.fk_tagid FROM file_tag_relations)"
53                 self.cursor.execute(delTagQuery)
54                 self.connection.commit()
55
56         def __connectTagsToFile(self, tags, fid):
```

```
57                """Insert tags to db and connect them up with the file
58                @param tags, List, list of tags you want to connect to the file
59                @param fid, integer, the id of the file you want to connect the tags to"""
60                tags = list(set(tags))  #remove duplicates from list
61                for row in tags:
62                        self.cursor.execute("INSERT OR IGNORE INTO tagnames (tagname, backup)
                              VALUES (?, ?)", (row, False))
63                        self.cursor.execute("SELECT tagid FROM tagnames WHERE tagname = ?", (
                              row, ))
64                        res = self.cursor.fetchall()
65                        self.cursor.execute("INSERT INTO file_tag_relations(fk_fid, fk_tagid)
                              VALUES (?, ?)", (fid, res[0][0]))
66
67                self.connection.commit()
68
69
70        def __generateFilesArray(self, li):
71                """Takes the result from SELECT * FROM files... statement.
72                Then gets for each of the files the corresponding tags, and puts it all
                      together
73                into a File.py object"""
74                ret_value       = []
75                for row in li:
76                        self.cursor.execute("SELECT tagnames.tagname FROM files LEFT JOIN
                              file_tag_relations, tagnames ON (files.fid = file_tag_relations.
                              fk_fid AND file_tag_relations.fk_tagid = tagnames.tagid) WHERE
                              files.fid = ?", (row[0], ))
77                        tagLi           = self.cursor.fetchall()
78                        tagList         = self.__changeList(tagLi)
79
80                        fi      = File.File(fileName=row[1], path=row[2], backup=row[3], isDir
                              =row[4], tags=tagList)
81                        ret_value.append(fi)
82                return ret_value
83
84        def updateFile(self, fi):
85                #TODO update backup state!
86                """Updates The tags of a file. Updating the backup value is planned and coming
                       soon
87                @param fi, File, The file you want to update. The path and filename have to be
                       the same as on the DB
88                You can not use this to move a file, there is moveFile() for that.
89                If the file you pass does not exist on the DB yet, addFile will be called
                      instead
90                Make sure not to pass a File object with no tags, except if you want to wipe
                      out all tags of that file on DB level"""
91                new_tags = []   #Tags from file object (ATTENTION! They get filtered later!)
92                old_tags = []   #Tags from database
93                if self.fileInDB(fi):
94                        old_tags = self.getTagsToFile(fi)
95                        new_tags = fi.getTags()
96                        for row in old_tags:
97                                try:
98                                        #remove all occurences of old tags from the new tag
                                              array, so only new tags are left
99                                        new_tags = filter (lambda a: a != row, new_tags)
100                               except:
101                                       print "Error while removing element from old_tags"
102                       self.cursor.execute("SELECT files.fid FROM files WHERE files.filename
                              = ? AND files.path = ?", (fi.getFileName(), fi.getPath(), ))
103                       fid = self.cursor.fetchall()[0][0]
104                       if len(new_tags) > 0:
105                               self.__connectTagsToFile(new_tags, fid)
106                       else:
107                               pass
108
```

```
109                         #Remove old tags from database
110                         deprecatedTags = old_tags
111                         for row in fi.getTags():
112                                 try:
113                                         #Remove all tags tags of the file object from the
                                                array of tags that are in the database
114                                         #This leaves us with just the tags that are in the
                                                database but that are NOT in the file object
115                                         deprecatedTags = filter(lambda a: a != row,
                                                deprecatedTags)
116                                 except:
117                                         print ""
118                         if len(deprecatedTags) > 0:
119                                 tagids = []
120                                 for line in deprecatedTags:
121                                         self.cursor.execute("SELECT tagid FROM tagnames WHERE
                                                tagname = ?", (line, ))
122                                         tagids.extend(self.cursor.fetchall()[0])
123                                 for line in tagids:
124                                         self.cursor.execute("DELETE FROM file_tag_relations
                                                WHERE fk_tagid = ? AND fk_fid = ?", (line, fid, ))
125                                 self.connection.commit()
126                                 self.__cleanupTags()
127                 else:
128                         self.addFile(fi)
129
130
131         def fileInDB(self, fi):
132                 """Checks wether a files is in the db or not
133                 @param fi, File, The The file which's presence in the DB you want to check.
                        Only name and path are needed"""
134                 self.cursor.execute("SELECT files.filename FROM files WHERE files.filename = ?
                        AND files.path = ?", (fi.getFileName(), fi.getPath(), ))
135                 value = self.cursor.fetchall()
136
137                 if len(value) > 0:
138                         return True
139                 else:
140                         return False
141
142 #       def executeQuerry(self, query):
143 #               """Deprecated! Won't be provided anymore for security reasons!"""
144 #               self.cursor.execute(query)
145 #               return self.cursor.fetchall()
146
147         def getTagsToFile(self, _file):
148                 """Returns all tags that are connected to the passed file
149                 @param _file, File, The file whichs tags you want (just fileName and path are
                        important)
150                 @retur List, List with the Tags of the file"""
151                 li              = None
152                 self.cursor.execute("SELECT tagnames.tagname FROM files LEFT JOIN
                        file_tag_relations, tagnames ON (files.fid = file_tag_relations.fk_fid AND
                        file_tag_relations.fk_tagid = tagnames.tagid) WHERE path = ? AND filename
                        = ?", (_file.getPath(), _file.getFileName()))
153                 li      = self.cursor.fetchall()
154                 return self.__changeList(li)
155
156         def getFilesFromPath(self, path):
157                 """Get all Files that are in a specific directory. Files will be returned
                        containing all tags 'n' stuff
158                 @param path, String, the Path you want to get files from (make sure to include
                         / or \ at the end)
159                 @return List, list of Files that are in the specified path"""
160                 ret_value       = []
161                 li              = None
```

```
162
163             self.cursor.execute("SELECT␣*␣FROM␣files␣WHERE␣path␣=␣?", (path, ))
164             li      = self.cursor.fetchall()
165             return self.__generateFilesArray(li)
166
167     def getFilesFromTag(self, tag):
168             """Gets you all files that 'have' a specific tag
169             @param tag, String, The tag you want to get the corresponding files to"""
170             ret_value       = []
171             li              = None
172             ids             = []
173
174             self.cursor.execute("SELECT␣files.*␣FROM␣files␣LEFT␣JOIN␣file_tag_relations,␣
                    tagnames␣ON␣(files.fid␣=␣file_tag_relations.fk_fid␣AND␣file_tag_relations.
                    fk_tagid␣=␣tagnames.tagid)␣WHERe␣tagnames.tagname␣=␣?", (tag, ))
175             li      = self.cursor.fetchall()
176             return self.__generateFilesArray(li)
177
178     def getAllTags(self):
179             """Returns all tags in the form of strings
180             @return List, list of all tags"""
181             self.cursor.execute("SELECT␣tagname␣FROM␣tagnames")
182             li = self.cursor.fetchall()
183             return self.__changeList(li)
184
185     def addFile(self, fi):
186             """Adds a file to the database
187             @param, fi, File, Fileobject that you want to add"""
188             #IMPORTANT: For this to work, the field tagnames.tagname has to be marked as
                    UNIQUE!
189             #Otherwise, attempts to insert tags might cause trouble!
190             if not self.fileInDB(fi):
191                     self.cursor.execute("INSERT␣INTO␣files(filename,␣path,␣backup,␣isdir)␣
                            VALUES␣(?,␣?,␣?,␣?)", (fi.getFileName(), fi.getPath(), fi.
                            getBackup(), fi.getIsDir()))
192                     fid     = self.cursor.lastrowid
193                     self.connection.commit();
194                     self.__connectTagsToFile(fi.getTags(), fid)
195             else:
196                     self.updateFile(fi)
197
198     def removeFile(self, fi):
199             """Removes a file from the Database.
200             @param fi, File, Fileobject representing the file you want to remove (only
                    fileName and path are important)"""
201             if self.fileInDB(fi):
202                     self.cursor.execute("SELECT␣files.fid␣FROM␣files␣WhERE␣files.path␣=␣?␣
                            AND␣files.filename␣=␣?␣", (fi.getPath(), fi.getFileName()))
203                     fid = self.cursor.fetchall()[0][0]
204                     self.cursor.execute("DELETE␣FROM␣files␣WHERE␣files.fid␣=␣?", (fid, ))
205                     self.connection.commit()
206                     self.cursor.execute("DELETE␣FROM␣file_tag_relations␣WHERE␣
                            file_tag_relations.fk_fid␣=␣?␣", (fid, ))
207                     self.connection.commit()
208                     self.__cleanupTags()
209             else:
210                     print "The␣File␣"+fi.getFullPath()+"␣is␣not␣in␣the␣database,␣won't␣be␣
                            removed"
211
212     def addTagToFile(self, fi, tag):
213             """Adds a single tag to a file.
214             @param fi, File Fileobject that represents the file you want to add the tag to
                    (only fileName and path are important)
215             @tag, String, the tag you want to add to the file"""
216             if self.fileInDB(fi):
```

```
217                         idQuery = "SELECT files.fid FROM files WHERE files.filename = '%s' AND
                                files.path = '%s'" % (fi.getFileName(), fi.getPath())
218                         self.cursor.execute(idQuery)
219                         fid = self.cursor.fetchall()[0][0]
220                         self.__connectTagsToFile([tag, ], fid)
221
222 #       def addTag(self, tag):
223 #               """DEPRECATED!
224 #               Add a tag to the database without connecting it to a file.
225 #               Does not really make any sense, because we are deleting tags with no relations
                    at several points."""
226 #               query = "INSERT OR IGNORE INTO tagnames (tagname, backup) VALUES ('%s', 'False
                    ')" % (tag, )
227 #               self.cursor.execute(query)
228 #               self.connection.commit()
229
230         def moveFile(self, f1, f2):
231                 """rename/move a file.
232                 @param f1, File, a File object with the path and the name of the file as it is
                        BEFORE the movement
233                 @param f2, File, a File object with the path and the name of the file as it is
                        AFTER the movement"""
234                 if not self.fileInDB(f2):
235                         self.cursor.execute("UPDATE files SET filename = ?, path = ? WHERE
                                filename = ? AND path = ?", (f2.getFileName(), f2.getPath(), f1.
                                getFileName(), f1.getPath(), ))
236                         self.connection.commit()
237                 else:
238                         print "file '" + f2.getFullPath() +"' exists, can't move!"
239
240         def renameFile(self, f1, f2):
241                 #Just calling moveFile
242                 self.moveFile(f1, f2)
243
244         def getFile(self, fi):
245                 """Gets all the details to a file (tags, backup state, ....
246                 @param fi, File, A file object with the filename and the path of the file
                        whichs details you want to get"""
247                 self.cursor.execute("SELECT * from files WHERE filename = ? and path = ?", (fi
                        .getFileName(), fi.getPath()))
248                 f2 = self.cursor.fetchall()
249                 f3 = self.__generateFilesArray(f2)[0]
250                 return f3
251
252         def copyFile(self, f1, f2):
253                 """Copy a file (including all tags, backup stae 'n' stuff).
254                 @param f1, File, the file you want to copy
255                 @param f2, File, a File object with the name and the path of the copy"""
256                 if not self.fileInDB(f2):
257                         f3 = self.getFile(f1)
258                         f3.setFileName(f2.getFileName())
259                         f3.setPath(f2.getPath())
260                         self.addFile(f3)
261
262 if __name__ == "__main__":
263         #Modul tests:
264         #========================================
265         print 'Mdule tests start'
266         db = DB("testdb")
267         f1 = File.File(fileName="documentation.tex", path="/home/niklaus/Documents/", tags=['
                LaTeX', 'documentation', 'filebrowser', 'project', 'school', 'test', ])
268         db.addFile(f1)
269         if len(db.getAllTags()) == 6:
270                 print "Test 01: Succeed"
271         else:
272                 print "Test 01: FAIL"
```

```
273
274         f2 = File.File(fileName="Music/", path="/home/niklaus/", tags=['music', 'multimedia',
                'entertainment', 'test',])
275         db.addFile(f2)
276         if len(db.getAllTags()) == 9:
277                 print "Test 02: Succeed"
278         else:
279                 print "Test 02: FAIL"
280
281         if len(db.getTagsToFile(f2)) == 4:
282                 print "Test 03: Succeed"
283         else:
284                 print "Test 03: FAIL"
285
286         f3 = File.File(fileName="Music/", path="/home/niklaus/", tags=['music', 'multimedia',
                'entertainment',])
287         db.updateFile(f3)
288         if len(db.getAllTags()) == 9 and len(db.getTagsToFile(f3)) == 3:
289                 print "Test 04: Succeed"
290         else:
291                 print "Test 04: FAIL"
292
293         f4 = File.File(fileName="documentation.tex", path="/home/niklaus/Documents/", tags=['
                LaTeX', 'documentation', 'filebrowser', 'project', 'school', ])
294         db.addFile(f4)
295         if len(db.getAllTags()) == 8 and len (db.getTagsToFile(f4)) == 5:
296                 print "Test 05: Succeed"
297         else:
298                 print "Test 05: FAIL"
299
300         f5 = File.File(fileName="documentation.tex", path="/home/niklaus/Documents", tags=['
                LaTeX', 'documentation', 'filebrowser', 'project', 'school', 'dbtest', 'modultest'
                , ])
301         db.updateFile(f5)
302         if len(db.getAllTags()) == 10 and len(db.getTagsToFile(f5)) == 7:
303                 print "Test 06: Succeed"
304         else:
305                 print "Test 06: FAIL"
306
307         db.removeFile(f5)
308         if len(db.getFilesFromPath("/home/niklaus/Documents/")) == 0 and len(db.getAllTags())
                == 3:
309                 print "Test 07: Succeed"
310         else:
311                 print "Test 07:FAIL"
312
313         f6 = File.File(path="/home/niklaus/Videos/", fileName="Movies/", isDir=True, tags=['
                entertainment', 'multimedia', 'movie'])
314         db.updateFile(f6)
315         fTest = db.getFilesFromPath("/home/niklaus/Videos/")[0]
316         if fTest.getFileName() == "Movies/" and fTest.getPath() == "/home/niklaus/Videos/" and
                 fTest.getIsDir() == True and fTest.getTags() == ['movie', 'multimedia', '
                entertainment', ]:
317                 print "Test 08: Succeed"
318         else:
319                 print "Test 08: FAIL"
320
321         f7 = File.File(path="/home/niklaus/doku/", fileName="projektantrag.tex", tags=['LaTeX'
                , 'projectexplorer', 'berufsschule',], backup=True)
322         f8 = File.File(path="/home/niklaus/", fileName="doku/", isDir=True, tags=['
                projectexplorer', 'brufsschule', ])
323         f9 = File.File(path="/home/niklaus/doku/", fileName="projektantrag.pdf", tags=['
                projectexplorer', 'berufsschule'])
324         f10 = File.File(path="/home/niklaus/doku/", fileName="projektplan.tex", tags=['LaTeX',
                 'projectexplorer', 'berufsschule',], backup=True)
```

```
325         f11 = File.File(path="/home/niklaus/doku/", fileName="projektplan.pdf", tags=['
                projectexplorer', 'berufsschule'])
326         db.addFile(f7)
327         db.addFile(f8)
328         db.addFile(f9)
329         db.addFile(f10)
330         db.addFile(f11)
331         if len(db.getFilesFromPath("/home/niklaus/doku/")) == 4 and db.getFilesFromTag("LaTeX"
                )[0].getBackup() == True:
332             print "Test 09: Succeed"
333         else:
334             print "Test 09: FAIL"
335 
336         f12 = File.File(path="/home/niklaus/doku/projektplan/", fileName="projektplan.tex",
                tags=['LaTeX', 'projectexplorer', 'berufsschule',], backup=True)
337         f13 = File.File(path="/home/niklaus/doku/projektplan/", fileName="projektplan01.pdf",
                tags=['projectexplorer', 'berufsschule'])
338         db.moveFile(f10, f12)
339         db.renameFile(f11, f13)
340         if len(db.getFilesFromPath("/home/niklaus/doku/")) == 2 and len(db.getFilesFromPath("/
                home/niklaus/doku/projektplan/")) == 2:
341             print "Test 10: Succeed"
342         else:
343             print "Test 10: FAIL"
344 
345         if db.getFilesFromPath("/home/niklaus/doku/projektplan/")[1].getFileName()=="
                projektplan01.pdf" or db.getFilesFromPath("/home/niklaus/doku/projektplan/")[0].
                getFileName()=="projektplan01.pdf":
346             print "Test 11: Succeed"
347         else:
348             print "Test 11: FAIL"
349 
350         if db.getFilesFromPath("/home/niklaus/doku/projektplan/")[1].getFileName()=="
                projektplan.tex" or db.getFilesFromPath("/home/niklaus/doku/projektplan/")[0].
                getFileName()=="projektplan.tex":
351             print "Test 12: Succeed"
352         else:
353             print "Test 12: FAIL"
354 
355         if db.fileInDB(f13):
356             print "Test 13: Succeed"
357         else:
358             print "Test 13: FAIL"
359 
360         if not db.fileInDB(f10):
361             print "Test 14: Succeed"
362         else:
363             print "Test 14: FAIL"
364 
365         f14 = File.File(path="/home/niklaus/test/", fileName="test.txt", tags=['foo', 'bar',
                ])
366         f15 = File.File(path="/home/niklaus/test/", fileName="test.txt", tags=['foo', 'bar',
                ])
367         db.addFile(f14)
368         db.addFile(f15)
369         if len(db.getFilesFromPath("/home/niklaus/test/")) == 1:
370             print "Test 15: Succeed"
371         else:
372             print "Test 15: FAIL"
373 
374         f16 = File.File(path="/home/niklaus/test/", fileName="test.txt", tags=['foo', 'bar', '
                muh', ])
375         db.addFile(f16)
376         if len(db.getFilesFromPath("/home/niklaus/test/")) == 1 and db.getFilesFromPath("/home
                /niklaus/test/")[0].getTags() == ['foo', 'bar', 'muh', ]:
377             print "Test 16: Succeed"
```

```
378         else:
379                 print "Test 16: FAIL"
380 
381         f17 = File.File(path="/home/project/", fileName="heroes", tags=['shylux', 'tschabold',
                'bash.vi'])
382         db.addFile(f17)
383         if len(db.getTagsToFile(f17)) == 3:
384                 print "Test 17: Succeed"
385         else:
386                 print "Test 17: FAIL"
387 
388         f18 = File.File(path="/home/project/", fileName="heroes", tags=['shylux', 'tschabold',
                'bash.vi', 'heroes', 'project'])
389         db.updateFile(f18)
390         if len(db.getTagsToFile(f17)) == 5:
391                 print "Test 18: Succeed"
392         else:
393                 print "Test 18: FAIL"
394 
395         f19 = File.File(path="/home/project/", fileName="heroes", tags=['tschabold', ])
396         db.updateFile(f19)
397         if len(db.getTagsToFile(f17)) == 1:
398                 print "Test 19: Succeed"
399         else:
400                 print "Test 19: FAIL"
401 
402         f20 = db.getFile(f18)
403         if len(f20.getTags()) == 1 and f20.getTags()[0] == 'tschabold':
404                 print "Test 20: Succeed"
405         else:
406                 print "Test 20: FAIL"
407 
408         f21 = File.File(path="/home/niklaus/copy/", fileName="copy.text", tags=['copy', 'test'
                , 'projectexplorer'])
409         f22 = File.File(path="/home/test/copy/", fileName="copy.tested")
410         db.addFile(f21)
411         db.copyFile(f21, f22)
412         f23 = db.getFile(f22)
413         if f23.getFileName() == "copy.tested" and f23.getPath()=="/home/test/copy/" and len(
                f23.getTags()) == 3:
414                 print "Test 21: Succeed"
415         else:
416                 print "Test 21: FAIL"
417 
418         fx1 = File.File(path="/test/27/", fileName="foo")
419         fx2 = File.File(path="/test/28/", fileName="bar")
420         db.moveFile(fx1, fx2)
421         print "Still running"
422         fy1 = File.File("/bin/fantasy/test.txt")
423         db.removeFile(fy1)
424         print "still running"
425         #TODO addTagToFile
426         print "="*20
427         db.getFile(f18)
```

10.3 Utility.py

```
1  #!/usr/bin/python
2  
3  #File:                  Utility.py
4  #Description:           Hier sind alle kleinen Hilfsfunktionen die wir selber programmieren
       drin
5  #Author:                Kaleb Tschabold
6  #Creation Date:         14.4.2011
7  #
```

```
8  #History:              --Version--     --Date--        --Activities--
9  #                      0.1             14.4.2011       Grundfunktionalitaeten werden erstellt
10 #                      0.2             18.4.2011       Funktion um ein neues Objekt zu
       erstellen
11
12 import sys
13 from time import strftime
14
15 class Utility:
16
17         def __init__(self):
18                 self.NO = NO
19                 pass
20
21         def checkOS(self):
22                 platform = sys.platform
23                 if platform.startswith("linux"):
24                         return "linux"
25                 elif platform.startswith("win"):
26                         return "windows"
27                 elif platform.startswith("darwin"):
28                         return "mac"
29
30
31         def strBooleanToBoolean(self,s):
32                 if s == 'False':
33                         return False
34                 if s == 'True':
35                         return True
36         def getTime(self):
37                 return strftime("%Y-%m-%d_%H:%M")
38
39         def uniqueFiles(self,files):
40                 unique = []
41                 for f in files:
42                         isUnique = True
43                         for u in unique:
44                                 if f.getFileName() == u.getFileName():
45                                         isUnique = False
46                         if isUnique:
47                                 unique.append(f)
48                 return unique
49
50 #Dynamisches Objekt, dass fuer normale Objekte gebraucht werden kann
51 class NO():
52         pass
```

10.4 CLI.py

```
1  #!/usr/bin/python
2
3  #File:                 CLI.py
4  #Description:          Mit dieser Klasse kann man ueber die Kommandozeille mit dem Programm
       interagieren
5  #Author:               Kaleb Tschabold
6  #Creation Date:        29.3.2011
7  #
8  #History:              --Version--     --Date--        --Activities--
9  #                      0.1             29.3.2011       Grundfunktionalitaeten werden erstellt
10 #                      0.1             18.4.2011       Thread und Endlos Loop
11 #                      0.1             19.4.2011       Erste Option implementiert (modus,
       verbos)
12
13
14 #Modul um diese Klasse als Seperaten-Prozess zu starten
```

```
15 import threading
16 import time
17 import sys
18 import os
19 from optparse import *
20
21 class CLI(threading.Thread):
22         def __init__(self,sys):
23                 threading.Thread.__init__(self)
24                 self.sys = sys
25
26         def run(self):
27                         usage = "usage: %prog [options] arg"
28                         version = (self.sys.c.prgname)+" "+str(self.sys.c.version)
29                         option_list =   [
30                                                 make_option("-m", "--modus", dest="modus",
                                                    help="Start a inline 'commandprompt'"),
31                                                 make_option("-v", "--verbose", dest="verbose",
                                                     help="Zeigt alle Ausgaben", action="
                                                    store_true"),
32                                                 make_option("-q", "--quite", dest="verbose",
                                                    help="Zeigt keine Ausgaben", action="
                                                    store_false", default=True),
33                                         ]
34                         parser = OptionParser(usage,version=version,option_list=option_list)
35                         a = (options,args) = parser.parse_args()
36                         if options.verbose:
37                                 self.verbose()
38                         if not options.verbose:
39                                 self.no_verbose()
40                         if options.modus:
41                                 if options.modus == 'gui':
42                                         self.sys.start_gui()
43                                 if options.modus == 'inline':
44                                         self.inline()
45         def verbose(self):
46                 sys.settrace(self.sys)
47
48         def no_verbose(self):
49                 #sys.stdout = os.devnull
50                 sys.stderr = os.devnull
51                 #sys.stdwrn = os.devnull
52                 pass
53
54         def inline(self):
55                         print('--- INLINE COMMAND PROMPT ---')
56                         print('0 => Ende')
57                         cli = True
58                         while cli:
59                                 time.sleep(self.sys.c.sleep)
60                                 cmd = raw_input('>>> ')
61                                 #cmd = sys.stdin.readline()
62                                 #if cmd == '0\n':
63                                 if cmd == '0':
64                                         #CLI wird beendet
65                                         cli = False
66                                         self.sys.stoppall()
67                                 if cmd == 'v' or cmd == 'verbose':
68                                         self.verbose()
69                                 if cmd == 'q' or cmd == 'verbose quit':
70                                         self.no_verbose()
71                                 if cmd == 'm' or cmd == 'modus':
72                                         print('gui=> Graphics Interface')
73                                         tmp = raw_input('  modus >>> ')
74                                         if tmp == 'gui':
75                                                 self.sys.start_gui()
```

```
76                                                  break
```

10.5 TagManager.py

```
1  #!/usr/bin/python
2
3  #File:                  TagManager.py
4  #Description:           Tag Magement
5  #Author:                Kaleb Tschabold
6  #Creation Date:         29.3.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             29.3.2011       Grundfunktionalitaeten werden erstellt
10
11 from File import *
12
13 class TagManager():
14         def __init__(self,sys):
15                 self.sys = sys
16
17         def searchMatchTags(self,name):
18                 all = self.sys.db.getAllTags()
19                 matched = []
20                 a = name.split(',')
21                 a[len(a)-1].strip()
22                 for i in range(len(all)):
23                         if all[i].find(name) == 0:
24                                 matched.append(all[i])
25                 return matched
26
27         def getBackups(self,tag):
28                 files = self.sys.db.getFilesFromTag(tag)
29                 backupArray = []
30                 for i in range(len(files)):
31                         backupArray.extend(files[i].getBackups())
32                 backupArray = self.sys.u.uniqueFiles(backupArray)
33                 return backupArray
34
35         def makeBackup(self,tag):
36                 files = self.sys.db.getFilesFromTag(tag)
37                 for i in range(len(files)):
38                         files[i].makeBackup()
39
40         def restoreFrom(self,tag,backup):
41                 files = self.sys.db.getFilesFromTag(tag)
42                 for f in files:
43                         fb = f.getBackups()
44                         for eachBackup in fb:
45                                 if eachBackup.getFileName() == backup.getFileName():
46                                         f.restoreFrom(eachBackup)
47
48         def removeBackups(self,tag,backup):
49                 files = self.sys.db.getFilesFromTag(tag)
50                 for f in files:
51                         fb = f.getBackups()
52                         for eachBackup in fb:
53                                 if eachBackup.getFileName() == backup.getFileName():
54                                         o = File(eachBackup.getFullPath()+'/'+f.getFileName())
55                                         o.remove()
```

10.6 FileManager.py

```
1  #!/usr/bin/python
2
3  #File:                  FileManager.py
4  #Description:           Ist fuer den Dateizugriff zustaendig
5  #Author:                Kaleb Tschabold
6  #Creation Date:         14.4.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             14.4.2011       Grundfunktionalitaeten werden erstellt
10 #                       0.1             23.4.2011       Erste Funktionen um Dateilisten zu
       laden
11
12 #Unsere Klassen
13 from File import *
14
15 #andere Klassen
16 import os
17
18 class FileManager:
19         def __init__(self,sys):
20                 self.sys = sys
21                 pass
22
23         def getFilesFromDir(self,path):
24                 a = []
25                 if not os.path.exists(path):
26                         matched = self.searchMatchDir(path)
27                         if len(matched) >= 1:
28                                 array = matched
29                         else:
30                                 return a
31                         for i in range(len(array)):
32                                 a.append(File(fullPath=array[i],isDir=self.isDir(array[i])))
33                                 a[i].setTags(self.sys.db.getTagsToFile(a[i]))
34                 else:
35                         if path == '':
36                                 path = path + '/'
37                         if path[-1:] != '/':
38                                 path = path + '/'
39                         array =  os.listdir(path)
40                         for i in range(len(array)):
41                                 fullpath = path + array[i]
42                                 if self.isDir(fullpath):
43                                         fullpath = fullpath + '/'
44                                 a.append(File(fullPath=fullpath,isDir=self.isDir(fullpath)))
45                                 a[i].setTags(self.sys.db.getTagsToFile(a[i]))
46                 return a
47
48         def getFileName(self,path):
49                 return os.path.basename(path)
50
51         def getDirName(self,path):
52                 s = path.split('/')
53                 return s[len(s)-2]
54
55         def getParentDir(self,path):
56                 s = path.split('/')
57                 l = len(s)
58                 if l >= 3:
59                         s.remove(s[l-2])
60                         return '/'.join(s)
61                 else:
62                         return False
63
64         def isDir(self,path):
```

```
65                  return os.path.isdir(path)
66
67          def searchMatchDir(self,path):
68                  match = []
69                  try:
70                          ddf = self.divideDirAndFile(path)
71                          l = os.listdir(ddf[0])
72                          for i in range(len(l)):
73                                  if (l[i].find(ddf[1]) >= 0 and os.path.isdir(ddf[0] + '/' + l[
                                        i])) or (ddf[1] == '' and os.path.isdir(ddf[0] + '/' + l[i
                                        ])):
74                                          if ddf[0] == '/':
75                                                  match.append(ddf[0] + l[i] + '/')
76                                          else:
77                                                  match.append(ddf[0] + '/' + l[i] + '/')
78                  except:
79                          pass
80                  return match
81
82          def divideDirAndFile(self,dirandfile):
83                  dir = ''
84                  file = ''
85                  if dirandfile.find('/') >= 0:
86                          dir = dirandfile[0:dirandfile.rfind('/')]
87                          if dir.strip() == '':
88                                  dir = '/'
89                          file = dirandfile[dirandfile.rfind('/')+1:(len(dirandfile))]
90                  else:
91                          dir = '/'
92                          file = dirandfile[1:]
93                  divided = []
94                  divided.append(dir)
95                  divided.append(file)
96                  return divided
97
98
99          def openFile(self,path):
100                 if self.sys.c.os == 'linux':
101                         #Funktioniert nur bei Ubuntu
102                         os.system('/usr/bin/xdg-open␣'+path.replace(chr(32),'\␣'))
103                 elif self.sys.c.os == 'windows':
104                         os.system('"'+path+'"')
105                 elif self.sys.c.os == 'mac':
106                         os.system('open␣'+path.replace(chr(32),'\␣'))
107                 else:
108                         #Da muss noch eine Loesung sein, wenn die Datei nicht gestartet werden
                                kann
109                         pass
110
111         def openDir(self,path):
112                 self.sys.gui.txtEntry.set_text(path)
113                 self.sys.gui.updateView()
114
115
116         def getDeletedFilesFromBackups(self,path):
117                 items = self.getFilesFromDir(path)
118                 found = []
119                 originFolder = self.divideDirAndFile(path)[0] + '/'
120                 if originFolder == '//':
121                         originFolder = '/'
122                 backupFolderPath = originFolder+'.pb_backup'+'/'
123                 backupFolders = self.getFilesFromDir(backupFolderPath)
124                 for backup in backupFolders:
125                         backupPath = backup.getFullPath()
126                         backup = self.getFilesFromDir(backupPath)
127                         for backupFile in backup:
```

```
128                                 foundFile = False
129                                 if items != 'error':
130                                         for i in items:
131                                                 if backupFile.getFileName() == i.getFileName()
                                                        :
132                                                         foundFile = True
133                                 if foundFile == False:
134                                         found.append(File(originFolder+backupFile.getFileName
                                            (),deleted=True))
135                 return found
```

10.7 FileSystemListener.py

```
1  #!/usr/bin/python
2
3  #File:                  FileSystemListener.py
4  #Description:           Diese Klasse ist der Handler fuer Aktivitaeten im FileSystem
5  #Author:                Lukas Knoepfel
6  #Creation Date:         14.4.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             14.4.2011       Grundgeruest erstellt
10 #                       0.2             19.4.2011       Grundfunktionalitaet erstellt
11
12 try:
13         from FileSystemListener_Linux import *
14 except:
15         pass
16 try:
17         from FileSystemListener_Mac import *
18 except:
19         pass
20 try:
21         from FileSystemListener_Windows import *
22 except ImportError as (errno, strerror):
23         print strerror
24         pass
25 import threading
26 from File import *
27
28 class FileSystemListener(threading.Thread):
29         listener = None
30         db = None
31         def __init__(self,sys):
32                 threading.Thread.__init__(self)
33                 self.sys = sys
34                 self.db = sys.getDb()
35                 stros = self.sys.u.checkOS()
36                 if stros == "linux":
37                         print("it's a linux!")
38                         self.listener = FileSystemListener_Linux(self)
39                         return
40                 if stros == "windows":
41                         print("it's a windows!")
42                         self.listener = FileSystemListener_Windows(self)
43                         return
44                 print "Can't initialize FileSystemListener. Given: ", stros
45
46         def add_watch(self, path, rec):
47                 self.listener.add_watch(path, rec)
48
49         def run(self):
50                 self.listener.start()
51
52         def stop(self):
```

```
53                 self.listener.stop()
54
55         # Event kommt als String mit dem Dateipfad an.
56         def create_event(self, event):
57                 self.sys.gui.actview.update()
58                 print "create_event: ", event
59         def delete_event(self, event):
60                 self.db.removeFile(File(event))
61                 print "delete_event: ", event
62         def modify_event(self, event):
63                 print "modify_event: ", event
64         def move_event(self, fr, to):
65                 self.sys.gui.actview.update()
66                 self.db.moveFile(File(fr), File(to))
67                 print "move: from: ", fr, " to: ", to
```

10.8 FileSystemListener Linux.py

```
1  #!/usr/bin/python
2
3  #File:                  FileSystemListener_Linux.py
4  #Description:           Diese Klasse kann auf einem Linux das File System ueberwachen
5  #Author:                Lukas Knoepfel
6  #Creation Date:         14.4.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             14.4.2011       Grundgeruest erstellt
10 #                       0.2             18.4.2011       Events werden korrekt abgefangen. TODO
       : add_watch implementieren und blockierendes .loop() in thread
11
12 import pyinotify
13
14 class FileSystemListener_Linux(pyinotify.ProcessEvent):
15         listener = None
16         notifier = None
17         parent = None
18         mask = pyinotify.IN_DELETE | pyinotify.IN_CREATE | pyinotify.IN_MODIFY | pyinotify.
               IN_MOVED_FROM | pyinotify.IN_MOVED_TO # watched events
19         lastfrom = None
20         def __init__(self, tparent):
21                 self.parent = tparent
22                 self.listener = pyinotify.WatchManager()
23                 self.notifier = pyinotify.Notifier(self.listener, self)
24                 #self.listener.add_watch('/home/shylux/project-browser', self.mask, rec=True)
25
26         def add_watch(self, path, recur):
27                 print recur
28                 self.listener.add_watch(path, self.mask, rec=recur)
29
30         def start(self):
31                 self.notifier.loop()
32         def stop(self):
33                 self.notifier.stop()
34
35         def process_IN_CREATE(self, event):
36                 self.parent.create_event(event.pathname)
37         def process_IN_DELETE(self, event):
38                 self.parent.delete_event(event.pathname)
39         def process_IN_MODIFY(self, event):
40                 self.parent.modify_event(event.pathname)
41         def process_IN_MOVED_FROM(self, event):
42                 self.lastfrom = event.pathname
43         def process_IN_MOVED_TO(self, event):
44                 if (self.lastfrom != None):
45                         self.parent.move_event(self.lastfrom, event.pathname)
```

```
46                  self.lastfrom = None
```

10.9 FileSystemListener Windows.py

```
1  #!/usr/bin/python
2
3  #File:                  FileSystemListener_Windows.py
4  #Description:           Diese Klasse kann auf einem Windows das File System ueberwachen
5  #Author:                Lukas Knoepfel
6  #Creation Date:         14.4.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             14.4.2011       Grundgeruest erstellt
10 #                       0.2             19.4.2011       Grundfunktionalitaet erstellt
11
12 try:
13         import os
14         import win32file
15         import win32con
16 except ImportError:
17         #shit happens :P
18         pass
19
20 class FileSystemListener_Windows():
21     parent = None
22     acpath = None
23     hdir = None
24     lastfrom = None
25     ACTIONS = {
26             1 : "Created",
27             2 : "Deleted",
28             3 : "Modified",
29             4 : "RenameF",#"Renamed from something"
30             5 : "RenameT"#"Renamed to something"
31             }
32     FILE_LIST_DIRECTORY = 0x0001
33
34     def __init__(self, tparent):
35         self.parent = tparent
36
37     def add_watch(self, path, recur):
38         self.acpath = path
39         self.hdir = win32file.CreateFile(
40             path,
41             self.FILE_LIST_DIRECTORY,
42             win32con.FILE_SHARE_READ | win32con.FILE_SHARE_WRITE,
43             None,
44             win32con.OPEN_EXISTING,
45             win32con.FILE_FLAG_BACKUP_SEMANTICS,
46             None
47             )
48
49     def start(self):
50         while 1:
51             results = win32file.ReadDirectoryChangesW(
52                 self.hdir,
53                 1024,
54                 True,
55                 win32con.FILE_NOTIFY_CHANGE_FILE_NAME |
56                 win32con.FILE_NOTIFY_CHANGE_DIR_NAME |
57                 win32con.FILE_NOTIFY_CHANGE_ATTRIBUTES |
58                 win32con.FILE_NOTIFY_CHANGE_SIZE |
59                 win32con.FILE_NOTIFY_CHANGE_LAST_WRITE |
60                 win32con.FILE_NOTIFY_CHANGE_SECURITY,
61                 None,
```

```
62                  None
63                  )
64
65              for action, file in results:
66                  full_filename = os.path.join(self.acpath, file)
67                  #print full_filename, self.ACTIONS.get(action, "Unknown")
68                  if action == 3:
69                      self.parent.modify_event(full_filename)
70                  elif action == 1:
71                      self.parent.create_event(full_filename)
72                  elif action == 2:
73                      self.parent.delete_event(full_filename)
74                  elif action == 4:
75                      self.lastfrom = full_filename
76                  elif action == 5:
77                      if (self.lastfrom != None):
78                          self.parent.move_event(self.lastfrom, full_filename)
79                      self.lastfrom = None
```

10.10 FileSystemListener Mac.py

```
1  #!/usr/bin/python
2
3  #File:                  FileSystemListener_Mac.py
4  #Description:           Diese Klasse kann auf einem Mac das File System ueberwachen
5  #Author:                Kaleb Tschabold
6  #Creation Date:         14.4.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             14.4.2011       Grundfunktionalitaeten werden erstellt
10
11 class FileSystemListener_Mac:
12         def __init__(self):
13                 pass
```

10.11 GUI.py

```
1  #!/usr/bin/python
2
3  #File:                  GUI.py
4  #Description:           Klasse die die Grafikanzeige verwaltet
5  #Author:                Kaleb Tschabold
6  #Creation Date:         29.3.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             29.3.2011       Grundfunktionalitaeten werden erstellt
10 #                       0.2             18.4.2011       Testen mit pygtk Elementen
11 #                       0.3             23.4.2011       Erstellte Event's auf die
       verschiedenen Elemente
12 #Link:
13 #http://zignar.net/page/pygtk-mit-glade                  mit xml(aus glade) Programm Grafische
       Oberflaeche erstellen
14
15 from HirarchicalView import *
16 from TagView import *
17 from FileProperties import *
18
19 import gobject
20 gobject.threads_init()
21 import threading
22 import pygtk
23 pygtk.require('2.0')
24 import gtk
```

```
25 import gtk.glade
26 import sys
27
28
29 #class GUI(threading.Thread):
30 class GUI():
31         def __init__(self,sys):
32                 #threading.Thread.__init__(self)
33                 gobject.threads_init()
34                 self.sys = sys
35                 self.mod = sys.c.startview
36                 self.hview = HirarchicalView(self.sys)
37                 self.tview = TagView(self.sys)
38                 self.actview = None
39
40         #def run(self):
41         def start(self):
42                 print('run: GUI')
43
44                 #Init-Window
45                 self.xml = gtk.glade.XML("gui.glade")
46                 self.window = self.xml.get_widget("winMain")
47                 self.window.set_title(self.sys.c.prgname + ' - Version ' + str(self.sys.c.
                        version))
48                 self.window.connect("destroy", self.stopploop)
49                 self.window.connect("key-release-event",self.eventF5)
50
51                 #Connect Toogle-Buttons
52                 self.btnHirarchical = self.xml.get_widget('btnHirarchical')
53                 self.btnHirarchical.connect('toggled', self.showHirarchical)
54                 self.btnTag = self.xml.get_widget('btnTag')
55                 self.btnTag.connect('toggled', self.showTag)
56
57                 #Connect TextInput Field
58                 self.txtEntry = self.xml.get_widget('txtEntry')
59                 self.txtEntry.connect('changed',self.searchKey)
60                 self.com = gtk.EntryCompletion()
61                 self.txtEntry.set_completion(self.com)
62                 self.listcompl = gtk.ListStore(gobject.TYPE_STRING)
63                 self.com.set_model(self.listcompl)
64                 self.com.set_text_column(0)
65                 self.com.set_popup_set_width(True)
66
67                 #Add Menu-Item by 'Ansicht'
68                 ansichtmenu = gtk.Menu()
69                 ansicht = self.xml.get_widget('mnuAnsicht')
70                 ansicht.set_submenu(ansichtmenu)
71                 self.mnuHirarchical = gtk.RadioMenuItem(None,'Hierarchisch')
72                 self.mnuHirarchical.connect('toggled',self.showHirarchical)
73                 ansichtmenu.append(self.mnuHirarchical)
74                 self.mnuTag = gtk.RadioMenuItem(self.mnuHirarchical,'Tag')
75                 self.mnuTag.connect('toggled',self.showTag)
76                 ansichtmenu.append(self.mnuTag)
77
78                 #Modify other MenuItems
79                 self.mnuBeenden = self.xml.get_widget('mnuBeenden')
80                 self.mnuBeenden.connect('activate',self.stopploop)
81                 self.mnuDatei = self.xml.get_widget('mnuDatei')
82                 self.mnuBearbeiten = self.xml.get_widget('mnuBearbeiten')
83                 self.mnuHilfe = self.xml.get_widget('mnuHilfe')
84                 self.mnuDatei.set_sensitive(False)
85                 self.mnuBearbeiten.set_sensitive(False)
86                 self.mnuHilfe.set_sensitive(False)
87
88                 #Init Status Bar
89                 self.Status = self.xml.get_widget('stsStatusBar')
```

```
90                  self.Status.push(1,'init')
91
92                  #Init Navigation Buttons
93                  self.btnBack = self.xml.get_widget('btnBackward')
94                  self.btnBack.connect('clicked',self.historyBack)
95                  self.btnBack.set_sensitive(False)
96                  self.btnFor = self.xml.get_widget('btnForward')
97                  self.btnFor.connect('clicked',self.historyFor)
98                  self.btnFor.set_sensitive(False)
99                  self.btnUp = self.xml.get_widget('btnUp')
100                 self.btnUp.connect('clicked',self.getParentFolder)
101                 self.btnUp.set_sensitive(False)
102
103                 #Init-View
104                 self.hspView = self.xml.get_widget('hspView')
105                 self.fileProperties = FileProperties(self.sys)
106                 self.hspView.add(self.fileProperties.getWidget())
107                 self.view = self.xml.get_widget('View')
108                 if self.mod == 'hirarchical':
109                         self.actview = self.hview
110                         self.txtEntry.set_text(self.actview.get_actTxtInput())
111                         self.showHirarchical('init')
112                 elif self.mod == 'tag':
113                         self.actview = self.tview
114                         self.txtEntry.set_text(self.actview.get_actTxtInput())
115                         self.showTag('init')
116                 else:
117                         self.actview = self.hview
118                         self.txtEntry.set_text(self.actview.get_actTxtInput())
119                         self.showHirarchical('init')
120                 #self.actview.historyUpdate('user')
121
122                 #Zeigt alles an
123                 self.showall()
124
125                 #Loop damit das Fenster nicht wieder geschlossen wird
126                 self.startloop()
127
128
129
130
131
132         #GUI Functionen
133         def showHirarchical(self,event):
134                 #Toggle Function
135                 if isinstance(event,gtk.RadioMenuItem):
136                         if self.mnuHirarchical.get_active():
137                                 self.btnHirarchical.set_active(True)
138                         else:
139                                 self.btnHirarchical.set_active(False)
140                 else:
141                         if self.btnHirarchical.get_active():
142                                 self.mnuHirarchical.set_active(True)
143                         else:
144                                 #Wenn zweimal die gleiche Ansicht gewaehlt wird wird die
                                        andere Ansicht aktiviert
145                                 if isinstance(self.actview,HirarchicalView) and event != 'init
                                        ':
146                                         self.showTag('init')
147                                         return 0
148                                 self.mnuHirarchical.set_active(False)
149                 if event == 'init':
150                         self.btnHirarchical.set_active(True)
151                         self.mnuHirarchical.set_active(True)
152
153                 #Init-View
```

```
154                 self.changeView(self.hview)
155
156         def showTag(self,event):
157                 #Toggle Function
158                 if isinstance(event,gtk.RadioMenuItem):
159                         if self.mnuTag.get_active():
160                                 self.btnTag.set_active(True)
161                         else:
162                                 self.btnTag.set_active(False)
163                 else:
164                         if self.btnTag.get_active():
165                                 self.mnuTag.set_active(True)
166                         else:
167                                 #Wenn zweimal die gleiche Ansicht gewaehlt wird wird die
                                        andere Ansicht aktiviert
168                                 if isinstance(self.actview,TagView) and event != 'init':
169                                         self.showHirarchical('init')
170                                         return 0
171                                 self.mnuTag.set_active(False)
172                 if event == 'init':
173                         self.btnTag.set_active(True)
174                         self.mnuTag.set_active(True)
175
176                 #Init-View
177                 self.changeView(self.tview)
178
179         def changeView(self,newview):
180                 #Speichert Text Input String in der zuschliessenden View
181                 self.actview.set_actTxtInput(self.txtEntry.get_text())
182                 self.fileProperties.update(None)
183
184                 #Schliesst die alte View
185                 if self.actview.get_parent() != None:
186                         self.view.remove(self.actview)
187
188                 #Fuegt die neue View an
189                 self.view.add(newview)
190
191                 #Aender Aktuel View
192                 self.mod = newview.mod
193                 self.actview = newview
194
195                 #Text Input aktualisieren
196                 self.txtEntry.set_text(self.actview.get_actTxtInput())
197
198                 #Fokus auf Text Input
199                 self.set_focus(self.txtEntry)
200
201                 #Update Statusbar
202                 self.Status.pop(1)
203                 self.Status.push(1,'Ansicht:␣'+str(self.mod))
204                 self.showall()
205
206                 #Update-View
207                 self.actview.update()
208
209         def searchKey(self,a):
210                 if self.actview.triggeredByNavigation:
211                         self.updateView()
212                 else:
213                         self.updateView('user')
214
215         def searchCompletion(self,completion, prefix, user_param1):
216                 self.updateView()
217
218         def updateView(self,actor='fn'):
```

```
219                 self.actview.set_actTxtInput(self.txtEntry.get_text())
220                 self.actview.update(actor)
221
222         def openDirInHirarchical(self,path):
223                 self.showHirarchical('init')
224                 self.actview.set_actTxtInput(path)
225                 self.listcompl.clear()
226                 self.txtEntry.set_text(self.actview.get_actTxtInput())
227
228         def historyBack(self,event):
229                 self.actview.triggeredByNavigation = True
230                 self.actview.historyCursor = self.actview.historyCursor-1
231                 self.actview.set_actTxtInput(self.actview.history[self.actview.historyCursor])
232                 self.txtEntry.set_text(self.actview.get_actTxtInput())
233                 self.actview.triggeredByNavigation = False
234
235         def historyFor(self,event):
236                 self.actview.triggeredByNavigation = True
237                 self.actview.historyCursor = self.actview.historyCursor+1
238                 self.actview.set_actTxtInput(self.actview.history[self.actview.historyCursor])
239                 self.txtEntry.set_text(self.actview.get_actTxtInput())
240                 self.actview.triggeredByNavigation = False
241
242         def getParentFolder(self,event):
243                 parent=self.sys.filemanager.getParentDir(self.actview.get_actTxtInput())
244                 self.txtEntry.set_text(parent)
245
246         def set_focus(self,widget):
247                 self.window.set_focus(widget)
248
249         def eventF5(self,widget,event):
250                 if event.keyval == gtk.gdk.keyval_from_name("F5"):
251                         self.actview.update()
252
253         def showall(self):
254                 self.window.show_all()
255
256         def terminate(self):
257                 self.raise_exc(SystemExit)
258
259         def startloop(self):
260                 gtk.main()
261
262         def stopploop(self,event=''):
263                 print('quite')
264                 gtk.main_quit()
265                 self.sys.stoppall()
```

10.12 TagView.py

```
1  #!/usr/bin/python
2
3  #File:                TagView.py
4  #Description:         Klasse um die Datein zu einem Tag anzuzeigen
5  #Author:              Kaleb Tschabold
6  #Creation Date:       29.3.2011
7  #
8  #History:             --Version--     --Date--        --Activities--
9  #                     0.1             29.3.2011       Grundfunktionalitaeten werden erstellt
10
11 from View import *
12 import gtk
13
14 class TagView(View):
15         mod = 'tag'
```

```
16          def __init__(self,sys):
17                  View.__init__(self,sys)
18                  self.set_actTxtInput(sys.c.initStrTag)
19                  self.connect('row-activated',self.rowActivate)
20
21          def update(self, actor = 'fn'):
22                  if self.sys.gui != None:
23                          oldmodel = self.get_model()
24                          self.model.clear()
25                          #Dieser Durchgang ist, wenn noch nicht's im TextFeld steht
26                          if self.get_actTxtInput() == '':
27                                  t = self.sys.db.getAllTags()
28                                  for i in range(len(t)):
29                                          self.model.append(None,[self.getFolderIcon(),'*',[t[i
                                                ]],t[i]])
30                                  self.set_model(self.model)
31                                  self.historyUpdate(actor)
32                                  self.historySymboleManagement()
33                                  self.updateParentFolderBtn()
34                                  self.completion()
35                                  #Damit diese Funktion abgebrochen wird
36                                  return 0
37                          #Dieser Durchgang ist, wenn Zeichen im Text Feld stehen
38                          itemarray = self.get_actTxtInput().split(',')
39                          for j in range(len(itemarray)):
40                                  oneitem = itemarray[j].strip()
41                                  self.items = self.sys.db.getFilesFromTag(oneitem)
42                                  for i in range(len(self.items)):
43                                          self.items[i].setIsDir(self.items[i].getIsDir())
44                                          if self.items[i].getIsDir():
45                                                  self.model.append(None,[self.getFolderIcon(),
                                                        self.items[i].getFileName(),self.items[i],
                                                        ',␣'.join(self.items[i].getTags())])
46                                          else:
47                                                  self.model.append(None,[self.getFileIcon(),
                                                        self.items[i].getFileName(),self.items[i],
                                                        ',␣'.join(self.items[i].getTags())])
48                          self.set_model(self.model)
49                          if len(self.items)>0:
50                                  self.historyUpdate(actor)
51                          self.historySymboleManagement()
52                          self.updateParentFolderBtn()
53                          self.completion()
54
55          def completion(self):
56                          matched = self.sys.tagmanager.searchMatchTags(self.get_actTxtInput())
57                  #try:
58                          if self.sys.gui.listcompl != None:
59                                  #try:
60                                          self.sys.gui.listcompl.clear()
61                                  #except:
62                                  #       pass
63                          for i in range(len(matched)):
64                                  self.sys.gui.listcompl.append([matched[i]])
65                  #except:
66                  #       pass
67
68
69          def updateParentFolderBtn(self):
70                  self.sys.gui.btnUp.set_sensitive(False)
71
72          def rowActivate(self,treeview, path, user_data):
73                  f = self.getFObjFromSelectedRow()
74                  if isinstance(f,list):
75                          self.sys.gui.txtEntry.set_text(f[0])
76                          self.sys.gui.updateView()
```

```
77                  else:
78                          if not f.getIsDir():
79                                  self.sys.filemanager.openFile(f.getFullPath())
80                          else:
81                                  self.sys.gui.openDirInHirarchical(f.getFullPath())
```

10.13 HirarchicalView.py

```
1  #!/usr/bin/python
2
3  #File:                  HirarchicalView.py
4  #Description:           Klasse um die Orderstrucktur anzuzeigen
5  #Author:                Kaleb Tschabold
6  #Creation Date:         29.3.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             29.3.2011       Grundfunktionalitaeten werden erstellt
10 #                       0.1             18.3.2011       Erben von View und init Funktion
       aufrufen
11
12 from View import *
13 import pygtk
14 pygtk.require('2.0')
15 import gtk
16
17 class HirarchicalView(View):
18         mod = 'hirarchical'
19         def __init__(self,sys):
20                 View.__init__(self,sys)
21
22                 self.history.append('/')
23                 self.historyCursor = 0
24                 self.sys = sys
25                 self.set_actTxtInput(sys.c.initStrHirarchical)
26                 self.connect('row-activated',self.rowActivate)
27
28         def update(self, actor = 'fn'):
29                 if self.sys.gui != None:
30                         if self.get_actTxtInput() == '':
31                                 self.sys.gui.txtEntry.set_text('/')
32                                 return 0
33                         self.items = self.sys.filemanager.getFilesFromDir(self.acttxtinput)
34                         if self.items != 'error':
35                                 #Ordner konnte geoffnet werden
36                                 self.model.clear()
37                                 for i in range(len(self.items)):
38                                         if self.items[i].getIsDir():
39                                                 self.model.append(None,[self.getFolderIcon(),
                                                        self.items[i].getFileName(),self.items[i],
                                                        ',␣'.join(self.items[i].getTags())])
40                                         else:
41                                                 self.model.append(None,[self.getFileIcon(),
                                                        self.items[i].getFileName(),self.items[i],
                                                        ',␣'.join(self.items[i].getTags())])
42                                 #Sucht nach geloeschten Dateien/Ordner in den Backups
43                                 deletedFiles = self.sys.filemanager.getDeletedFilesFromBackups
                                       (self.acttxtinput)
44                                 for df in deletedFiles:
45                                         self.model.append(None,[self.getDeletedIcon(),df.
                                               getFileName(),df,''])
46                                 self.set_model(self.model)
47                                 #Ruft History Verwaltung auf
48                                 if len(self.get_actTxtInput()) > 0:
49                                         if self.get_actTxtInput()[-1] == '/':
50                                                 self.historyUpdate(actor)
```

```
51                          self.updateParentFolderBtn()
52                          self.historySymboleManagement()
53                          #Ruft Auto-Completion Update Funktion auf
54                          self.completion()
55
56          def completion(self):
57                  matched = self.sys.filemanager.searchMatchDir(self.get_actTxtInput())
58                  try:
59                          if self.sys.gui.listcompl != None:
60                                  try:
61                                          self.sys.gui.listcompl.clear()
62                                  except:
63                                          pass
64                          for i in range(len(matched)):
65                                  self.sys.gui.listcompl.append([matched[i]])
66                  except:
67                          pass
68
69          def updateParentFolderBtn(self):
70                  parent = self.sys.filemanager.getParentDir(self.get_actTxtInput())
71                  if parent:
72                          self.sys.gui.btnUp.set_sensitive(True)
73                  else:
74                          self.sys.gui.btnUp.set_sensitive(False)
75
76          def rowActivate(self,treeview, path, user_data):
77                  f = self.getFObjFromSelectedRow()
78                  if not f.getIsDir():
79                          self.sys.filemanager.openFile(f.getFullPath())
80                  else:
81                          self.sys.filemanager.openDir(f.getFullPath())
```

10.14 View.py

```
1  #!/usr/bin/python
2
3  #File:                  View.py
4  #Description:           Diese Klasse kann die 2 Ansichten(Tag und Hirarchisch) verwalten
5  #Author:                Kaleb Tschabold
6  #Creation Date:         14.4.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             14.4.2011       Grundfunktionalitaeten werden erstellt
10 #                       0.2             18.4.2011       Erbt von gtk.Layout
11 #
12 #Link:
13 #http://www.pygtk.org/pygtk2tutorial/sec-TreeViewDragAndDrop.html       Bearbeitet selected
14
15 #Eigene Klassen
16 from FileProperties import *
17 from File import *
18
19 #Andere Klassen
20 import pygtk
21 pygtk.require('2.0')
22 import gtk
23 import gobject
24 import os
25
26 class View(gtk.TreeView):
27         def __init__(self,sys):
28                 gtk.TreeView.__init__(self,None)
29                 self.sys = sys
30                 self.acttxtinput = ''
31
```

```
32                  self.history = []
33                  self.historyCursor = -1
34
35                  self.triggeredByNavigation = False
36
37                  self.createTree()
38                  self.connect('button_release_event',self.showContext)
39                  self.connect('cursor-changed',self.updateTagProperties)
40
41          def createTree(self):
42                  #Objekt fuer den Baum
43                  self.model = gtk.TreeStore(gtk.gdk.Pixbuf, gobject.TYPE_STRING, gobject.
                        TYPE_PYOBJECT, gobject.TYPE_STRING)
44                  self.cl1 = gtk.TreeViewColumn('Datei')
45                  self.append_column(self.cl1)
46                  render1 = gtk.CellRendererPixbuf()
47                  self.cl1.pack_start(render1, expand=False)
48                  self.cl1.add_attribute(render1, 'pixbuf', 0)
49                  render2 = gtk.CellRendererText()
50                  self.cl1.pack_start(render2,True)
51                  self.cl1.add_attribute(render2,'text',1)
52                  self.cl2 = gtk.TreeViewColumn("Tags")
53                  self.append_column(self.cl2)
54                  render3 = gtk.CellRendererText()
55                  self.cl2.pack_start(render3,True)
56                  self.cl2.add_attribute(render3,'text',3)
57
58                  #Allgemeine Definitionen fuer den Baum
59                  #self.set_search_column(1)
60                  #self.cl1.set_sort_column_id(0)
61                  #self.cl2.set_sort_column_id(1)
62
63                  self.update()
64
65          def get_icon_pixbuf(self, stock):
66                  return self.render_icon(stock_id=getattr(gtk, stock),
67                                          size=gtk.ICON_SIZE_MENU,
68                                          detail=None)
69          def getFolderIcon(self):
70                  return self.get_icon_pixbuf('STOCK_DIRECTORY')
71
72          def getFileIcon(self):
73                  return self.get_icon_pixbuf('STOCK_FILE')
74
75          def getDeletedIcon(self):
76                  return self.get_icon_pixbuf('STOCK_CANCEL')
77
78          def update(self):
79                  self.show_all()
80
81          def set_actTxtInput(self,text):
82                  self.acttxtinput = text
83
84          def get_actTxtInput(self):
85                  return self.acttxtinput
86
87          def getFObjFromSelectedRow(self):
88                  treeview = self
89                  selection = treeview.get_selection()
90                  selection.set_mode(gtk.SELECTION_SINGLE)
91                  tree_model, tree_iter = selection.get_selected()
92                  return tree_model.get_value(tree_iter,2)
93
94          def showContext(self, treeview, event):
95                  if event.button == 3:
96                          f = self.getFObjFromSelectedRow()
```

```
97                          m = gtk.Menu()
98                          #m1 = gtk.MenuItem ('Add Tag')
99                          #m.append(m1)
100                         m2 = gtk.MenuItem('Properties')
101                         m.append(m2)
102                         #m1.connect('button_press_event',self.context_AddTag,f)
103                         m2.connect('button_press_event',self.context_Properties,f)
104                         m.show_all()
105                         m.popup( None, None, None, event.button, event.time)
106                         return True
107                 return False
108
109         def context_Properties(self,widget,event,fobj):
110                 print('properties')
111
112         def updateTagProperties(self,event):
113                 try:
114                         f = self.getFObjFromSelectedRow()
115                 except:
116                         pass
117                 if f != None:
118                         self.sys.gui.fileProperties.update(f)
119                 #except:
120                 #       pass
121
122         def historyUpdate(self,actor='fn'):
123                 #Wenn wieder vorwaertz gesprungen wird, werden alle Element nach der aktuelen
                        Position geloescht
124                 if self.historyCursor > -1:
125                         if actor == 'user' and self.history[self.historyCursor] != self.
                                get_actTxtInput():
126                                 if self.historyCursor < len(self.history)-1:
127                                         rem = (len(self.history)-1) - self.historyCursor
128                                         l = len(self.history)
129                                         i = 1
130                                         #rem+1 weil die len von history eins mehr ist als der
                                                cursor
131                                         while i<rem+1:
132                                                 self.history.remove(self.history[l-i])
133                                                 i = i + 1
134                                         self.historyCursor = self.historyCursor
135                                 self.historyCursor = self.historyCursor + 1
136                                 self.history.append(self.get_actTxtInput())
137                                 self.historySymboleManagement()
138                 elif len(self.history) == 0:
139                         self.historyCursor = self.historyCursor + 1
140                         self.history.append(self.get_actTxtInput())
141                         self.historySymboleManagement()
142
143         def historySymboleManagement(self):
144                         h = len(self.history)-1
145                         c = self.historyCursor
146                         if h >= c and h != 0 and c > 0:
147                                 self.sys.gui.btnBack.set_sensitive(True)
148                         if h >= c and h != 0:
149                                 self.sys.gui.btnFor.set_sensitive(True)
150                         if c <= 0 or h == 0:
151                                 self.sys.gui.btnBack.set_sensitive(False)
152                         if c == h or h == 0:
153                                 self.sys.gui.btnFor.set_sensitive(False)
154
155 #Registriert diese Klasse als pygtk-widget
156 gobject.type_register(View)
```

10.15 File.py

```
#!/usr/bin/python

#File:                  File.py
#Description:           Jede Datei die im Programm bearbeitet wird wir in einer solchen Klasse referenziert
#Author:                Kaleb Tschabold
#Creation Date:         14.4.2011
#
#History:               --Version--     --Date--        --Activities--
#                       0.1             14.04.2011      Grundfunktionalitaeten werden erstellt
#                       0.9             21.04.2011      Created methods and functionality. Added __doc__s. added test (bottom)

import re
import os
from Constant import *
from Utility import *
from shutil import copytree, ignore_patterns, copyfile, rmtree
from RepeatTimer import *

class File:
        """represents a file on the filesysem"""
        fileName        = None
        path            = None
        isDir           = None
        tags            = None
        backup          = None
        fullPath        = None
        deleted         = None
        u               = None
        constant        = None
        os              = None
        def __init__(self, fullPath=None, fileName=None, path=None, tags=[], backup=False, isDir=False, deleted=False):
                """Constructor
                @param  fileName, string        , optional
                @param  path    , string        , optional
                @param  tags    , list          , optional
                @param  backup  , boolean       , optional
                """
                self.fileName   = fileName
                self.path       = path
                self.tags       = tags
                self.backup     = backup
                self.isDir      = isDir
                self.deleted    = deleted
                self.fullPath   = fullPath

                self.u = Utility()
                self.constant = Constant(self)
                self.os         = self.constant.os

                self.__splitFullPath()
                self.__checkPaths()

        def __splitFullPath(self):
                if not self.fullPath == None:
                        if self.fullPath.endswith("/") or self.fullPath.endswith("\\"):
                                p = re.compile("((?:.*\\\)|(?:.*/))((?:[^/]*)|(?:[^\\\]*))([/|\\\])")
                                match = p.match(self.fullPath)
                                if not match.group(1) == None and not match.group(2) == None:
                                        self.path = match.group(1)
                                        self.fileName = match.group(2)+match.group(3)
```

```
61                      else:
62                              p = re.compile("((?:.*\\\)|(?:.*/))((?:[^/]*)|(?:[^\\\]*))")
63                              match = p.match(self.fullPath)
64                              if not match.group(1) == None and not match.group(2) == None:
65                                      self.path = match.group(1)
66                                      self.fileName = match.group(2)
67
68      def __checkPaths(self):
69              # Check wether all paths have / (or \ on Windows) at the end
70              # This is highly experimental! Tell me if you have any trouble with it!
71              if self.os == 'linux' or self.os == 'mac':
72                      if not self.path == None:
73                              if not self.path.endswith("/") and not self.path.endswith("\\"
                                    ):
74                                      self.path = self.path+"/"
75                      if not self.fileName == None:
76                              if self.isDir == True and not self.fileName.endswith("/") and
                                    not self.fileName.endswith("\\"):
77                                      self.fileName = self.fileName+"/"
78              elif self.os == 'win':
79                      if not self.path == None:
80                              if not self.path.endswith("\\") and not self.path.endswith("/"
                                    ):
81                                      self.path = self.path+"\\"
82                      if not self.fileName == None:
83                              if self.isDir == True and not self.fileName.endswith("\\") and
                                     not self.fileName.endswith("/"):
84                                      self.fileName = self.fileName+"\\"
85
86
87      def setFileName(self, fileName):
88              """@param       filename        , string"""
89              self.fileName   = fileName
90              self.__checkPaths()
91
92      def setPath(self, path):
93              """@param       path    , string"""
94              self.path       = path
95              self.__checkPaths()
96
97      def setIsDir(self,b):
98              """@param       b       , boolean """
99              self.isDir = b
100             self.__checkPaths()
101
102     def setTags(self, tags):
103             """@param       tags    , list"""
104             self.tags       = tags
105
106     def setBackup(self, backup):
107             """@param backup        , boolean"""
108             self.backup     = backup
109
110     def setFullPath(self, fullPath):
111             """@param fullPath      , Path including filename"""
112             self.fullPath   = fullPath
113             self.__splitFullPath()
114             self.__checkPaths()
115
116     def getFileName(self):
117             """@return fileName, string"""
118             return self.fileName
119
120     def getPath(self):
121             """@return path, string"""
122             return self.path
```

```
123
124         def getIsDir(self):
125                 """@return isDir, boolean """
126                 return self.isDir
127
128         def getTags(self):
129                 """@return tags, list"""
130                 return self.tags
131
132         def getBackup(self):
133                 """@return backup, boolean"""
134                 return self.backup
135
136         def getFullPath(self):
137                 """@return fullPath, String representing the full path to the file, including
                        the file's name"""
138                 if self.fullPath == None:
139                         if not self.path==None or not self.fileName==None:
140                                 return self.path + self.fileName
141                         else:
142                                 print "either path or fileName (or both) are not specified.
                                    You can't use this function unless both are specified"
143                 else:
144                         return self.fullPath
145
146         def addTag(self, tag):
147                 """Adds a single tag (passed as string) to the list of tags.
148                 uses .append"""
149                 self.tags.append(tag)
150
151         def addTags(self, tags):
152                 """Adds a list of tags (passed as list) to the list of tags.
153                 uses .extend"""
154                 self.tags.extend(tags)
155
156         def remove(self):
157                 if os.path.exists(self.getFullPath()):
158                         if (os.path.isdir(self.getFullPath())):
159                                 rmtree(self.getFullPath())
160                         else:
161                                 os.remove(self.getFullPath())
162
163         def setDeleted(self,b):
164                 self.deleted = b
165
166         def getDeleted(self):
167                 return self.deleted
168
169         def makeBackup(self):
170                 dir = self.getPath() + ".pb_backup"
171                 if not os.path.exists(dir):
172                         os.makedirs(dir)
173                 bdir = dir + "/" + self.u.getTime().replace(':','')
174                 if not os.path.exists(bdir):
175                         os.makedirs(bdir)
176                 if (os.path.isdir(self.getFullPath())):
177                         copytree(self.getFullPath(), bdir + "/" + self.getFileName(), ignore=
                                ignore_patterns('.*', '*.pyc'))
178                 else:
179                         copyfile(self.getFullPath(), bdir + "/" + self.getFileName())
180
181
182         def getBackups(self):
183                 dir = self.getPath() + ".pb_backup"
184                 founded = []
185                 if os.path.exists(dir):
```

```
186                         listOfBackups = os.listdir(dir)
187                         for i in range(len(listOfBackups)):
188                                 bdir = dir + '/' + listOfBackups[i]
189                                 listOfDirs = os.listdir(bdir)
190                                 for j in range(len(listOfDirs)):
191                                         if listOfDirs[j] == self.getFileName()[:-1] or
                                                listOfDirs[j] == self.getFileName():
192                                                 founded.append(File(bdir))
193                         return founded
194                 else:
195                         return []
196
197
198         def restoreFrom(self,b):
199                 self.remove()
200                 if (os.path.isdir(b.getFullPath()+"/"+self.getFileName())):
201                         copytree(b.getFullPath()+"/"+self.getFileName(), self.getPath()+self.
                                getFileName(), ignore=ignore_patterns('.*', '*.pyc'))
202                 else:
203                         copyfile(b.getFullPath()+"/"+self.getFileName(), self.getPath()+self.
                                getFileName())
204
205         def repeatBackup(self, seconds):
206                 if seconds < 60:
207                         seconds = 60
208                 timer = RepeatTimer(seconds, self.makeBackup, 2)
209                 timer.start()
210
211 if __name__ == "__main__":
212         print "Starting tests"
213         print "Note: Tests 9,10,14,15 will fail on Windows... and so will many others probably
                "
214         print "=============="
215         x = File()
216         if x.getFileName() == None:
217                 print "Test #01: Succeed"
218         else:
219                 print "Test #01: FAIL"
220
221         x.setFileName("test")
222         if x.getFileName() == "test":
223                 print "Test #02: Succeed"
224         else:
225                 print "Test #02: FAIL"
226
227         x.setTags(["test", "tag"])
228         if x.getTags() == ["test","tag"]:
229                 print "Test #03: Succeed"
230         else:
231                 print "Test #03: FAIL"
232
233         x.addTag("new tag")
234         if x.getTags()[2] == "new tag":
235                 print "Test #04: Succeed"
236         else:
237                 print "Test #04: FAIL"
238
239         x.addTags(["extend","linux","unix"])
240         if x.getTags()[3:] == ["extend","linux","unix"]:
241                 print "Test #05: Succeed"
242         else:
243                 print "Test #05: FAIL"
244
245         x.setIsDir(True)
246         if x.getIsDir():
247                 print "Test #06: Succeed"
```

```
248         else:
249                 print "Test #06: FAIL"
250 
251         #Testing the path stuff...
252         f1 = File(fullPath="/home/niklaus/lol.txt")
253         if f1.getPath() == "/home/niklaus/" \
254             and f1.getFileName() == "lol.txt" and f1.getFullPath() == "/home/niklaus/lol.txt":
255                 print "Test #07: Succeed"
256         else:
257                 print "Test #07: FAIL"
258 
259         f2 = File(fullPath="C:\\windows\\system32\\chlous.txt.test")
260         if f2.getPath() == "C:\\windows\\system32\\" \
261             and f2.getFileName() == "chlous.txt.test" and f2.getFullPath() == "C:\\windows\\
                system32\\chlous.txt.test":
262                 print "Test #09: Succeed"
263         else:
264                 print "Test #09: FAIL"
265 
266         f3 = File(fullPath="C:\\windows\\system32\\")
267         if f3.getPath() == "C:\\windows\\" \
268             and f3.getFileName() == "system32\\" and f3.getFullPath() == "C:\\windows\\
                system32\\":
269                 print "Test #10: Succeed"
270         else:
271                 print "Test #10: FAIL"
272 
273         f4 = File("/home/niklaus/Music/")
274         if f4.getPath() == "/home/niklaus/" \
275             and f4.getFileName() == "Music/" and f4.getFullPath() == "/home/niklaus/Music/":
276                 print "Test #11: Succeed"
277         else:
278                 print "Test #11: FAIL"
279 
280         f5 = File(fileName="document.odt", path="/home/niklaus/Documents/")
281         if f5.getFullPath() == "/home/niklaus/Documents/document.odt":
282                 print "Test #12: Succeed"
283         else:
284                 print "Test #12: FAIL"
285 
286         f6 = File(fileName="Videos/", path="/home/niklaus/", isDir=True)
287         if f6.getFullPath() == "/home/niklaus/Videos/" and f6.getIsDir() == True:
288                 print "Test #13: Succeed"
289         else:
290                 print "Test #13: FAIL"
291 
292         f7 = File(fileName="hosts", path="C:\\windows\\system32\\etc\\")
293         if f7.getFullPath() == "C:\\windows\\system32\\etc\\hosts":
294                 print "Test #14: Succeed"
295         else:
296                 print "Test #14: FAIL"
297 
298         f8 = File(fileName="etc\\", path="C:\\windows\\system32\\", isDir=True)
299         if f8.getFullPath() == "C:\\windows\\system32\\etc\\" and f8.getIsDir() == True:
300                 print "Test #15: Succeed"
301         else:
302                 print "Test #15: FAIL"
303 
304         #Testing the automatic path correction
305         f9 = File(fileName="ozzed.ogg", path="/home/niklaus/Music")
306         if f9.getPath() == "/home/niklaus/Music/" and f9.getFileName() == "ozzed.ogg":
307                 print "Test #16: Succeed"
308         else:
309                 print "Test #16: Fail"
310 
311         f10 = File(fileName="Music", path="/home/niklaus", isDir=True)
```

```
312         if f10.getPath() == "/home/niklaus/" and f10.getFileName() == "Music/":
313                 print "Test #17: Succeed"
314         else:
315                 print "Test #17: FAIL"
316 
317         f11 = File(fileName="Music", isDir=True)
318         if f11.getFileName() == "Music/" and f11.getPath() == None:
319                 print "Test #18: Succeed"
320         else:
321                 print "Test #18: FAIL"
322 
323         f12 = File(path="/home/niklaus/somepath")
324         if f12.getPath() == "/home/niklaus/somepath/" and f12.getFileName() == None:
325                 print "Test #19: Succeed"
326         else:
327                 print "Test #19: FAIL"
328 
329         f13 = File()
330         f13.setFullPath("/home/niklaus/File.php")
331         if f13.getPath() == "/home/niklaus/" and f13.getFileName() == "File.php":
332                 print "Test #20: Succeed"
333         else:
334                 print "Test #20: FAIL"
335 
336         f14 = File(isDir=True)
337         f14.setFullPath("/home/niklaus/Music")
338         if f14.getPath() == "/home/niklaus/" and f14.getFileName() == "Music/":
339                 print "Test #21: Succeed"
340         else:
341                 print "Test #21: FAIL"
342 
343         print File.__init__.__doc__
344 
345         #Backup
346         b = File("/home/shylux/project-browser/experimental/small-test-programms")
347         b.repeatBackup(30.0)
```

10.16 FileProperties.py

```
1  import gtk
2  import gobject
3  from File import *
4  class FileProperties():
5          def __init__(self,sys):
6                  self.sys = sys
7                  self.fobj = None
8                  self.xml = gtk.glade.XML("fileproperties.glade")
9                  self.main = self.xml .get_widget("vbxMain")
10                 self.txtObjNames = self.xml.get_widget("txtNames")
11                 self.lblName = self.xml.get_widget("lblName")
12                 self.txtTags = self.xml.get_widget("txtTags")
13                 self.txtTags.connect('key-release-event',self.enterEventHandler)
14                 self.btnSave = self.xml.get_widget("btnSave")
15                 self.btnSave.connect('button_release_event',self.save)
16                 self.hbxTag = self.xml.get_widget("hbxTag")
17                 self.tagCont = self.xml.get_widget("conTagList")
18                 self.restoreCont = self.xml.get_widget("conRestoreCont")
19 
20 
21                 #Backup/Restore
22                 self.btnBackup = self.xml.get_widget("btnBackup")
23                 self.btnBackup.connect('clicked',self.backup)
24                 self.btnBackup.set_sensitive(False)
25                 self.btnRestore = self.xml.get_widget("btnRestore")
26                 self.btnRestore.connect('clicked',self.restoreBackup)
```

```
27                  self.btnRestore.set_sensitive(False)
28
29
30                  #Model Tag
31                  self.tagModel = gtk.TreeStore(gobject.TYPE_STRING)
32                  self.tagTree = gtk.TreeView(self.tagModel)
33                  self.tagTree.connect('row-activated',self.addClickedTag)
34                  self.tagCont.add(self.tagTree)
35                  self.tagCl1 = gtk.TreeViewColumn('Tag Name')
36                  self.tagTree.append_column(self.tagCl1)
37                  tagRender = gtk.CellRendererText()
38                  self.tagCl1.pack_start(tagRender)
39                  self.tagCl1.add_attribute(tagRender,'text',0)
40
41
42                  #Model Backups
43                  self.restoreModel = gtk.TreeStore(gobject.TYPE_STRING, gobject.TYPE_PYOBJECT)
44                  self.restoreTree = gtk.TreeView(self.restoreModel)
45                  self.restoreTree.connect('cursor-changed',self.updateRestoreButton)
46                  self.restoreTree.connect('button_release_event',self.showContext)
47                  self.restoreCont.add(self.restoreTree)
48                  self.restoreCl1 = gtk.TreeViewColumn('Datum')
49                  self.restoreTree.append_column(self.restoreCl1)
50                  restoreRender = gtk.CellRendererText()
51                  self.restoreCl1.pack_start(restoreRender)
52                  self.restoreCl1.add_attribute(restoreRender,'text',0)
53
54
55                  self.update(None)
56                  self.main.show_all()
57
58          def update(self,fobj):
59                  self.clearAll()
60                  self.fobj = None
61                  self.fobj = fobj
62                  self.lblName.set_label('')
63                  self.hbxTag.set_sensitive(False)
64                  if isinstance(fobj,File):
65                          self.lblName.set_label('Datei(en)')
66                          self.txtTags.set_text(', '.join(self.fobj.getTags()))
67                          self.txtObjNames.set_text(self.fobj.getFileName())
68                          self.updateTagModel()
69                          self.updateRestoreModel('file')
70                          self.hbxTag.set_sensitive(True)
71                          if fobj.getDeleted():
72                                  self.lblName.set_label('Sicherung aus Backup')
73                                  self.btnBackup.set_sensitive(False)
74                                  self.hbxTag.set_sensitive(False)
75                  elif type(fobj) == list:
76                          self.lblName.set_label('Tag(s)')
77                          self.txtObjNames.set_text(fobj[0])
78                          self.updateRestoreModel('tag')
79                  self.updateButtons()
80
81          def updateTagModel(self):
82                  self.tagModel.clear()
83                  for tag in self.sys.db.getAllTags():
84                          self.tagModel.append(None,[tag])
85
86          def updateRestoreModel(self,typ):
87                  self.restoreModel.clear()
88                  if isinstance(self.fobj,File):
89                          backups = self.fobj.getBackups()
90                          for b in backups:
91                                  self.restoreModel.append(None,[b.getFileName(),b])
92                  elif type(self.fobj) == list:
```

```
93                      backupArray = self.sys.tagmanager.getBackups(self.fobj[0])
94                      for b in backupArray:
95                              self.restoreModel.append(None,[b.getFileName(),b])
96
97
98      def clearAll(self):
99              self.fobj = None
100             self.tagModel.clear()
101             self.txtObjNames.set_text('')
102             self.txtTags.set_text('')
103             self.updateTagModel()
104
105     def getWidget(self):
106             return self.main
107
108     def addClickedTag(self,treeview, path, user_data):
109             selection = treeview.get_selection()
110             selection.set_mode(gtk.SELECTION_SINGLE)
111             tree_model, tree_iter = selection.get_selected()
112             tagname = tree_model.get_value(tree_iter,0)
113             if self.txtTags.get_text() == '':
114                     self.txtTags.set_text(tagname)
115             else:
116                     self.txtTags.set_text(self.txtTags.get_text()+', '+tagname)
117
118     def enterEventHandler(self,widget,event):
119             if event.keyval == gtk.gdk.keyval_from_name("Return"):
120                     self.save(widget,event)
121
122     def save(self,widget,event):
123                     tags = self.txtTags.get_text().split(',')
124                     l = len(tags)
125                     newtags = []
126                     for i in range(len(tags)):
127                             if tags[i].strip() != '':
128                                     newtags.append(tags[i].strip())
129                     tags = list(set(newtags))
130                     self.fobj.setTags(newtags)
131                     self.sys.db.updateFile(self.fobj)
132                     self.update(self.fobj)
133                     self.sys.gui.actview.update()
134
135     def updateButtons(self):
136                     if(isinstance(self.fobj,File) or type(self.fobj) == list):
137                             self.btnBackup.set_sensitive(True)
138                             if isinstance(self.fobj,File):
139                                     if self.fobj.getDeleted():
140                                             self.btnBackup.set_sensitive(False)
141                     else:
142                             self.btnBackup.set_sensitive(False)
143                     self.btnRestore.set_sensitive(False)
144
145     def updateRestoreButton(self,event):
146             self.btnRestore.set_sensitive(True)
147
148     def backup(self,event):
149             if isinstance(self.fobj,File):
150                     self.fobj.makeBackup()
151             elif type(self.fobj) == list:
152                     self.sys.tagmanager.makeBackup(self.fobj[0])
153             self.update(self.fobj)
154             self.sys.gui.actview.update()
155
156     def getSelectedBackup(self):
157             treeview = self.restoreTree
158             selection = treeview.get_selection()
```

```
159                 selection.set_mode(gtk.SELECTION_SINGLE)
160                 tree_model, tree_iter = selection.get_selected()
161                 return tree_model.get_value(tree_iter,1)
162 
163         def restoreBackup(self,event):
164                 if isinstance(self.fobj,File):
165                         self.fobj.restoreFrom(self.getSelectedBackup())
166                 elif type(self.fobj) == list:
167                         self.sys.tagmanager.restoreFrom(self.fobj[0],self.getSelectedBackup())
168                 self.sys.gui.actview.update()
169 
170         def removeBackup(self,widget,event,bf):
171                 if(isinstance(self.fobj,File)):
172                         f = File(bf.getFullPath()+'/'+self.fobj.getFileName())
173                         f.remove()
174                 elif type(self.fobj) == list:
175                         self.sys.tagmanager.removeBackups(self.fobj[0],bf)
176                 self.update(self.fobj)
177                 self.sys.gui.actview.update()
178 
179         def showContext(self, treeview, event):
180                 if event.button == 3:
181                         f = self.getSelectedBackup()
182                         m = gtk.Menu()
183                         m1 = gtk.MenuItem("Backup entfernen")
184                         m.append(m1)
185                         m1.connect('button_press_event',self.removeBackup,f)
186                         m.show_all()
187                         m.popup( None, None, None, event.button, event.time)
188                         return True
189                 return False
```

10.17 gui.glade

```
1  <?xml version="1.0"?>
2  <glade-interface>
3    <!-- interface-requires gtk+ 2.16 -->
4    <!-- interface-naming-policy toplevel-contextual -->
5    <widget class="GtkWindow" id="winMain">
6      <property name="title" translatable="yes">Aendere Titel</property>
7      <property name="window_position">center</property>
8      <property name="default_width">700</property>
9      <property name="default_height">350</property>
10     <child>
11       <widget class="GtkVBox" id="vbxMain">
12         <property name="visible">True</property>
13         <child>
14           <widget class="GtkMenuBar" id="mnuBar">
15             <property name="visible">True</property>
16             <child>
17               <widget class="GtkMenuItem" id="mnuDatei">
18                 <property name="visible">True</property>
19                 <property name="label" translatable="yes">_Datei</property>
20                 <property name="use_underline">True</property>
21                 <child>
22                   <widget class="GtkMenu" id="menu1">
23                     <property name="visible">True</property>
24                     <child>
25                       <widget class="GtkImageMenuItem" id="mnuNeu">
26                         <property name="label">gtk-new</property>
27                         <property name="visible">True</property>
28                         <property name="use_underline">True</property>
29                         <property name="use_stock">True</property>
30                       </widget>
31                     </child>
```

```
32                    <child>
33                      <widget class="GtkImageMenuItem" id="mnuOeffnen">
34                        <property name="label">gtk-open</property>
35                        <property name="visible">True</property>
36                        <property name="use_underline">True</property>
37                        <property name="use_stock">True</property>
38                      </widget>
39                    </child>
40                    <child>
41                      <widget class="GtkImageMenuItem" id="mnuSpeichern">
42                        <property name="label">gtk-save</property>
43                        <property name="visible">True</property>
44                        <property name="use_underline">True</property>
45                        <property name="use_stock">True</property>
46                      </widget>
47                    </child>
48                    <child>
49                      <widget class="GtkImageMenuItem" id="mnuSpeichernUnter">
50                        <property name="label">gtk-save-as</property>
51                        <property name="visible">True</property>
52                        <property name="use_underline">True</property>
53                        <property name="use_stock">True</property>
54                      </widget>
55                    </child>
56                    <child>
57                      <widget class="GtkSeparatorMenuItem" id="separatormenuitem1">
58                        <property name="visible">True</property>
59                      </widget>
60                    </child>
61                    <child>
62                      <widget class="GtkImageMenuItem" id="mnuBeenden">
63                        <property name="label">gtk-quit</property>
64                        <property name="visible">True</property>
65                        <property name="use_underline">True</property>
66                        <property name="use_stock">True</property>
67                      </widget>
68                    </child>
69                  </widget>
70                </child>
71              </widget>
72            </child>
73            <child>
74              <widget class="GtkMenuItem" id="mnuBearbeiten">
75                <property name="visible">True</property>
76                <property name="label" translatable="yes">_Bearbeiten</property>
77                <property name="use_underline">True</property>
78                <child>
79                  <widget class="GtkMenu" id="menu2">
80                    <property name="visible">True</property>
81                    <child>
82                      <widget class="GtkImageMenuItem" id="mnuAusschneiden">
83                        <property name="label">gtk-cut</property>
84                        <property name="visible">True</property>
85                        <property name="use_underline">True</property>
86                        <property name="use_stock">True</property>
87                      </widget>
88                    </child>
89                    <child>
90                      <widget class="GtkImageMenuItem" id="mnuKopieren">
91                        <property name="label">gtk-copy</property>
92                        <property name="visible">True</property>
93                        <property name="use_underline">True</property>
94                        <property name="use_stock">True</property>
95                      </widget>
96                    </child>
97                    <child>
```

```
98                    <widget class="GtkImageMenuItem" id="mnuEinfuegen">
99                      <property name="label">gtk-paste</property>
100                     <property name="visible">True</property>
101                     <property name="use_underline">True</property>
102                     <property name="use_stock">True</property>
103                   </widget>
104                 </child>
105                 <child>
106                   <widget class="GtkImageMenuItem" id="mnuLoeschen">
107                     <property name="label">gtk-delete</property>
108                     <property name="visible">True</property>
109                     <property name="use_underline">True</property>
110                     <property name="use_stock">True</property>
111                   </widget>
112                 </child>
113               </widget>
114             </child>
115           </widget>
116         </child>
117         <child>
118           <widget class="GtkMenuItem" id="mnuAnsicht">
119             <property name="visible">True</property>
120             <property name="label" translatable="yes">_Ansicht</property>
121             <property name="use_underline">True</property>
122           </widget>
123         </child>
124         <child>
125           <widget class="GtkMenuItem" id="mnuHilfe">
126             <property name="visible">True</property>
127             <property name="label" translatable="yes">_Hilfe</property>
128             <property name="use_underline">True</property>
129             <child>
130               <widget class="GtkMenu" id="menu4">
131                 <property name="visible">True</property>
132                 <child>
133                   <widget class="GtkImageMenuItem" id="mnuInfo">
134                     <property name="label">gtk-about</property>
135                     <property name="visible">True</property>
136                     <property name="use_underline">True</property>
137                     <property name="use_stock">True</property>
138                   </widget>
139                 </child>
140               </widget>
141             </child>
142           </widget>
143         </child>
144       </widget>
145       <packing>
146         <property name="expand">False</property>
147         <property name="position">0</property>
148       </packing>
149     </child>
150     <child>
151       <widget class="GtkEntry" id="txtEntry">
152         <property name="visible">True</property>
153         <property name="can_focus">True</property>
154         <property name="invisible_char">&#x25CF;</property>
155       </widget>
156       <packing>
157         <property name="expand">False</property>
158         <property name="position">1</property>
159       </packing>
160     </child>
161     <child>
162       <widget class="GtkHBox" id="hbxBody">
163         <property name="visible">True</property>
```

```
164             <child>
165               <widget class="GtkVBox" id="vbxButtons">
166                 <property name="visible">True</property>
167                 <property name="border_width">4</property>
168                 <child>
169                   <widget class="GtkVBox" id="vbox1">
170                     <property name="visible">True</property>
171                     <property name="spacing">3</property>
172                     <child>
173                       <widget class="GtkHBox" id="hbox2">
174                         <property name="visible">True</property>
175                         <property name="homogeneous">True</property>
176                         <child>
177                           <widget class="GtkButton" id="btnBackward">
178                             <property name="label">Zur&#xFC;ck</property>
179                             <property name="visible">True</property>
180                             <property name="can_focus">True</property>
181                             <property name="receives_default">True</property>
182                             <property name="image_position">bottom</property>
183                           </widget>
184                           <packing>
185                             <property name="position">0</property>
186                           </packing>
187                         </child>
188                         <child>
189                           <widget class="GtkButton" id="btnForward">
190                             <property name="label">Vor</property>
191                             <property name="visible">True</property>
192                             <property name="can_focus">True</property>
193                             <property name="receives_default">True</property>
194                           </widget>
195                           <packing>
196                             <property name="position">1</property>
197                           </packing>
198                         </child>
199                         <child>
200                           <widget class="GtkButton" id="btnUp">
201                             <property name="label">Hoch</property>
202                             <property name="visible">True</property>
203                             <property name="can_focus">True</property>
204                             <property name="receives_default">True</property>
205                           </widget>
206                           <packing>
207                             <property name="position">2</property>
208                           </packing>
209                         </child>
210                       </widget>
211                       <packing>
212                         <property name="position">0</property>
213                       </packing>
214                     </child>
215                   </widget>
216                   <packing>
217                     <property name="position">0</property>
218                   </packing>
219                 </child>
220                 <child>
221                   <widget class="GtkHSeparator" id="hseparator1">
222                     <property name="visible">True</property>
223                   </widget>
224                   <packing>
225                     <property name="expand">False</property>
226                     <property name="padding">3</property>
227                     <property name="position">1</property>
228                   </packing>
229                 </child>
```

```
230                 <child>
231                   <widget class="GtkToggleButton" id="btnHirarchical">
232                     <property name="label" translatable="yes">Hierarchisch</property>
233                     <property name="width_request">150</property>
234                     <property name="visible">True</property>
235                     <property name="can_focus">True</property>
236                     <property name="receives_default">True</property>
237                   </widget>
238                   <packing>
239                     <property name="position">2</property>
240                   </packing>
241                 </child>
242                 <child>
243                   <widget class="GtkHSeparator" id="hseparator">
244                     <property name="width_request">0</property>
245                     <property name="visible">True</property>
246                   </widget>
247                   <packing>
248                     <property name="expand">False</property>
249                     <property name="padding">3</property>
250                     <property name="position">3</property>
251                   </packing>
252                 </child>
253                 <child>
254                   <widget class="GtkToggleButton" id="btnTag">
255                     <property name="label" translatable="yes">Tag</property>
256                     <property name="width_request">150</property>
257                     <property name="visible">True</property>
258                     <property name="can_focus">True</property>
259                     <property name="receives_default">True</property>
260                   </widget>
261                   <packing>
262                     <property name="position">4</property>
263                   </packing>
264                 </child>
265               </widget>
266               <packing>
267                 <property name="expand">False</property>
268                 <property name="position">0</property>
269               </packing>
270             </child>
271             <child>
272               <widget class="GtkHPaned" id="hspView">
273                 <property name="visible">True</property>
274                 <property name="can_focus">True</property>
275                 <property name="position">300</property>
276                 <child>
277                   <widget class="GtkScrolledWindow" id="View">
278                     <property name="visible">True</property>
279                     <property name="can_focus">True</property>
280                     <property name="hscrollbar_policy">automatic</property>
281                     <property name="vscrollbar_policy">automatic</property>
282                     <child>
283                       <placeholder/>
284                     </child>
285                   </widget>
286                   <packing>
287                     <property name="resize">True</property>
288                     <property name="shrink">True</property>
289                   </packing>
290                 </child>
291                 <child>
292                   <placeholder/>
293                 </child>
294               </widget>
295               <packing>
```

```
296                 <property name="position">1</property>
297               </packing>
298             </child>
299           </widget>
300           <packing>
301             <property name="position">2</property>
302           </packing>
303         </child>
304         <child>
305           <widget class="GtkStatusbar" id="stsStatusBar">
306             <property name="visible">True</property>
307             <property name="spacing">2</property>
308           </widget>
309           <packing>
310             <property name="expand">False</property>
311             <property name="position">3</property>
312           </packing>
313         </child>
314       </widget>
315     </child>
316   </widget>
317 </glade-interface>
```

10.18 fileproperties.glade

```
1  <?xml version="1.0"?>
2  <glade-interface>
3    <!-- interface-requires gtk+ 2.16 -->
4    <!-- interface-naming-policy project-wide -->
5    <widget class="GtkVBox" id="vbxMain">
6      <property name="visible">True</property>
7      <property name="border_width">8</property>
8      <child>
9        <widget class="GtkVBox" id="vbox1">
10         <property name="visible">True</property>
11         <property name="spacing">1</property>
12         <child>
13           <widget class="GtkLabel" id="lblName">
14             <property name="visible">True</property>
15             <property name="label" translatable="yes">[platzhalter]</property>
16           </widget>
17           <packing>
18             <property name="expand">False</property>
19             <property name="position">0</property>
20           </packing>
21         </child>
22         <child>
23           <widget class="GtkEntry" id="txtNames">
24             <property name="visible">True</property>
25             <property name="can_focus">True</property>
26             <property name="invisible_char">&#x25CF;</property>
27           </widget>
28           <packing>
29             <property name="expand">False</property>
30             <property name="position">1</property>
31           </packing>
32         </child>
33         <child>
34           <widget class="GtkHSeparator" id="hseparator1">
35             <property name="visible">True</property>
36           </widget>
37           <packing>
38             <property name="expand">False</property>
39             <property name="padding">7</property>
40             <property name="position">2</property>
```

```
41            </packing>
42          </child>
43          <child>
44            <widget class="GtkNotebook" id="notebook1">
45              <property name="visible">True</property>
46              <property name="can_focus">True</property>
47              <property name="scrollable">True</property>
48              <property name="tab_hborder">8</property>
49              <child>
50                <widget class="GtkHBox" id="hbxTag">
51                  <property name="visible">True</property>
52                  <child>
53                    <widget class="GtkVBox" id="vbox2">
54                      <property name="visible">True</property>
55                      <property name="border_width">10</property>
56                      <property name="spacing">1</property>
57                      <child>
58                        <widget class="GtkEntry" id="txtTags">
59                          <property name="visible">True</property>
60                          <property name="can_focus">True</property>
61                          <property name="invisible_char">&#x25CF;</property>
62                        </widget>
63                        <packing>
64                          <property name="expand">False</property>
65                          <property name="position">0</property>
66                        </packing>
67                      </child>
68                      <child>
69                        <widget class="GtkLabel" id="label7">
70                          <property name="visible">True</property>
71                          <property name="label" translatable="yes">Tag List</property>
72                        </widget>
73                        <packing>
74                          <property name="expand">False</property>
75                          <property name="position">1</property>
76                        </packing>
77                      </child>
78                      <child>
79                        <widget class="GtkScrolledWindow" id="conTagList">
80                          <property name="visible">True</property>
81                          <property name="can_focus">True</property>
82                          <property name="hscrollbar_policy">automatic</property>
83                          <property name="vscrollbar_policy">automatic</property>
84                          <child>
85                            <placeholder/>
86                          </child>
87                        </widget>
88                        <packing>
89                          <property name="position">2</property>
90                        </packing>
91                      </child>
92                      <child>
93                        <widget class="GtkButton" id="btnSave">
94                          <property name="label" translatable="yes">Tags Speichern</property>
95                          <property name="visible">True</property>
96                          <property name="can_focus">True</property>
97                          <property name="receives_default">True</property>
98                        </widget>
99                        <packing>
100                         <property name="expand">False</property>
101                         <property name="position">3</property>
102                       </packing>
103                     </child>
104                   </widget>
105                   <packing>
106                     <property name="position">0</property>
```

```
                    </packing>
                  </child>
                </widget>
              </child>
              <child>
                <widget class="GtkLabel" id="label2">
                  <property name="visible">True</property>
                  <property name="label" translatable="yes">Tags</property>
                </widget>
                <packing>
                  <property name="tab_fill">False</property>
                  <property name="type">tab</property>
                </packing>
              </child>
              <child>
                <widget class="GtkVBox" id="vbox3">
                  <property name="visible">True</property>
                  <property name="border_width">10</property>
                  <property name="spacing">2</property>
                  <child>
                    <widget class="GtkHBox" id="hbox3">
                      <property name="visible">True</property>
                      <property name="homogeneous">True</property>
                      <child>
                        <widget class="GtkButton" id="btnBackup">
                          <property name="label" translatable="yes">Sichern</property>
                          <property name="visible">True</property>
                          <property name="can_focus">True</property>
                          <property name="receives_default">True</property>
                        </widget>
                        <packing>
                          <property name="position">0</property>
                        </packing>
                      </child>
                      <child>
                        <widget class="GtkButton" id="btnRestore">
                          <property name="label" translatable="yes">Wiederherstellen</property>
                          <property name="visible">True</property>
                          <property name="can_focus">True</property>
                          <property name="receives_default">True</property>
                        </widget>
                        <packing>
                          <property name="position">1</property>
                        </packing>
                      </child>
                    </widget>
                    <packing>
                      <property name="expand">False</property>
                      <property name="position">0</property>
                    </packing>
                  </child>
                  <child>
                    <widget class="GtkScrolledWindow" id="conRestoreCont">
                      <property name="visible">True</property>
                      <property name="can_focus">True</property>
                      <property name="hscrollbar_policy">automatic</property>
                      <property name="vscrollbar_policy">automatic</property>
                      <child>
                        <placeholder/>
                      </child>
                    </widget>
                    <packing>
                      <property name="position">1</property>
                    </packing>
                  </child>
                </widget>
```

```
173                 <packing>
174                   <property name="position">1</property>
175                 </packing>
176               </child>
177               <child>
178                 <widget class="GtkLabel" id="label3">
179                   <property name="visible">True</property>
180                   <property name="label" translatable="yes">Sichern</property>
181                 </widget>
182                 <packing>
183                   <property name="position">1</property>
184                   <property name="tab_fill">False</property>
185                   <property name="type">tab</property>
186                 </packing>
187               </child>
188               <child>
189                 <placeholder/>
190               </child>
191               <child>
192                 <placeholder/>
193                 <packing>
194                   <property name="type">tab</property>
195                 </packing>
196               </child>
197             </widget>
198             <packing>
199               <property name="position">3</property>
200             </packing>
201           </child>
202         </widget>
203         <packing>
204           <property name="position">0</property>
205         </packing>
206       </child>
207     </widget>
208 </glade-interface>
```