Realisierungsbericht

Niklaus Hofer, Lukas Knöpfel, Kaleb Tschabold

May 10, 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 10, 2011	Transfer nach LaTeX	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 5

2 Technische Detailspezifikation 5
2.1 Innere Struktur . . . 5
2.1.1 Lösungsvorschläge für die Struktur des Systemdesigns . . . 5
2.1.1.1 GUI . . . 5
2.1.1.2 Datenstruktur . . . 5
2.1.2 Struktur des Systemdesigns . . . 6
2.1.3 Beschreibung der Elemente . . . 7
2.2 Schnittstellendefinition . . . 7
2.3 Datenmodell . . . 8
2.3.1 Datenbank . . . 8
2.3.2 File-object . . . 8
2.4 Sicherheit . . . 9
2.5 Anforderungszuordnung . . . 9

3 Systemdokumentation 10
3.1 Inline-Dokumentation . . . 10
3.2 Benutzerhandbuch . . . 10
3.2.1 Systemübersicht . . . 10
3.2.1.1 Aufgabengebiet des Programms . . . 10
3.2.1.2 Programmoberfläche . . . 10
3.2.1.3 Anmerkungen zur korrekten Verwendung unter dem Aspekt der Sicherheit . . . 10
3.2.2 Anwenderfunktionalität . . . 11
3.2.2.1 Ansicht wechseln . . . 11
3.2.2.2 Verzeichnis wechseln . . . 11
3.2.2.3 Hisotry . . . 11
3.2.2.4 Dateien Öffnen . . . 12
3.2.2.5 Tags hinzufügen . . . 12
3.2.2.6 Tags entfernen . . . 12
3.2.2.7 Dateien anhand der Tags durchsuchen . . . 12
3.2.2.8 Fehlermeldungen . . . 12
3.2.2.9 Sichern . . . 12
3.3 Supporthandbuch . . . 12
3.3.1 Massnahmen bei Benutzerproblemen . . . 12
3.3.2 Massnahmen bei technischen Problemen . . . 12
3.3.3 Anhang zum Supporthandbuch . . . 13

4 Systemtest 13
4.1 Testspezifikation . . . 13
4.1.1 Kritikalität der Funktionseinheit . . . 13
4.1.2 Testanforderungen . . . 13
4.1.3 Testverfahren . . . 13
4.1.4 Tesenriterine . . . 13
4.1.5 Testfälle . . . 13
4.2 Testprozedur . . . 15
4.2.1 Vorbereitung . . . 15
4.2.1.1 Voraussetzungen . . . 15
4.2.1.2 Konfiguration . . . 15
4.2.2 Durchführung . . . 15
4.2.3 Nachbearbeitung . . . 15
4.3 Testprotokoll . . . 16
4.3.1 Testobjekt . . . 16
4.3.2 Testresultate . . . 16
4.3.3 Testauswerung . . . 16

5 Mittelbedarf 16

6 Planung und Organisation 16

7 Wirtschaftlichkeit **16**

8 Konsequenzen **17**

9 Antrag auf Freigabe der nächsten Projektphase **17**

10 Sourcecode **17**
10.1 Main.py . . . 17
10.2 DB.py . . . 19
10.3 Utility.py . . . 26
10.4 CLI.py . . . 27
10.5 TagManager.py . . . 28
10.6 FileManager.py . . . 29
10.7 FileSystemListener.py . . . 31
10.8 FileSystemListener Linux.py . . . 32
10.9 FileSystemListener Windows.py . . . 33
10.10 FileSystemListener Mac.py . . . 34
10.11 GUI.py . . . 34
10.12 TagView.py . . . 38
10.13 HirarchicalView.py . . . 40
10.14 View.py . . . 41
10.15 File.py . . . 44
10.16 AddTag.py . . . 49

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

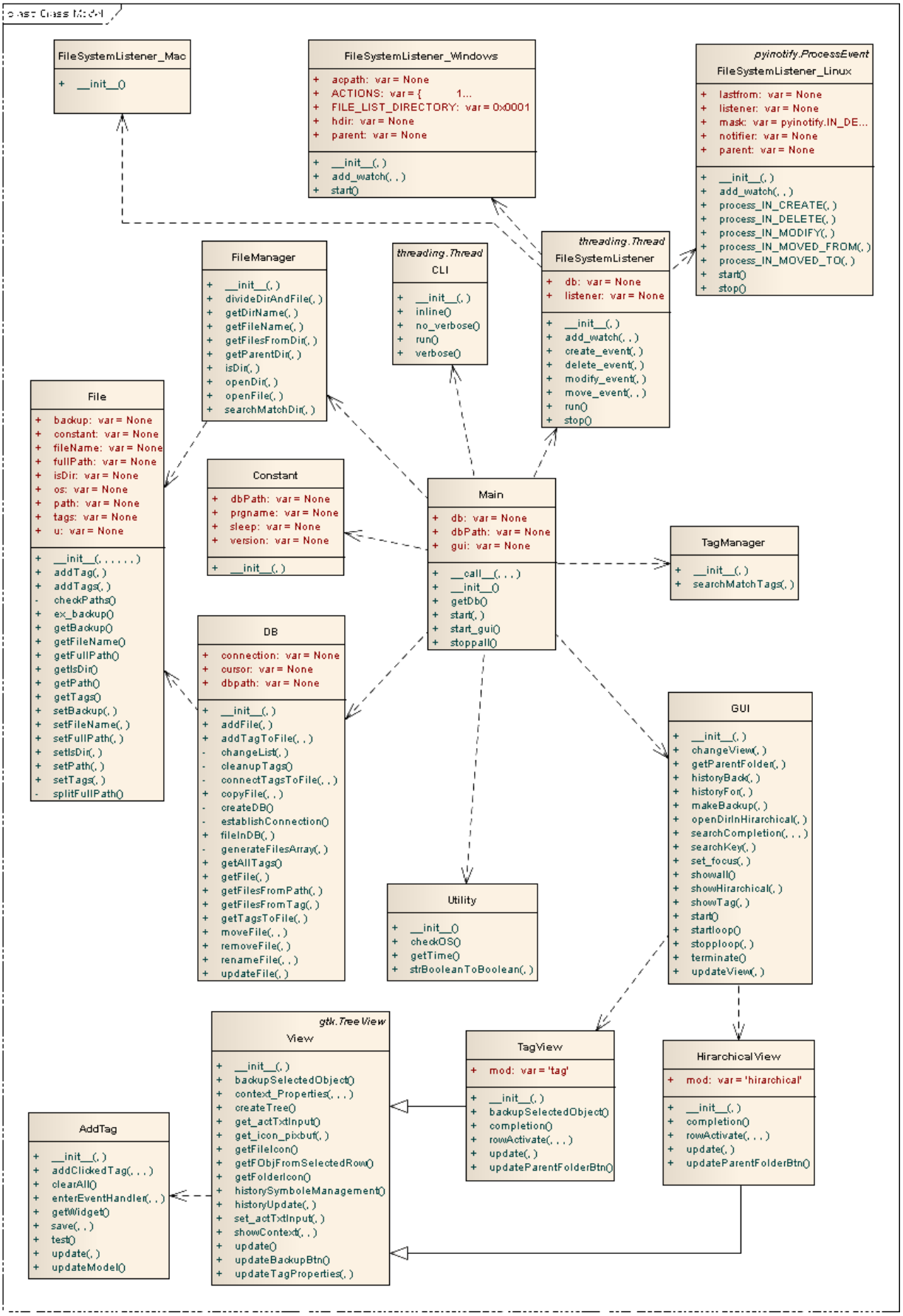

Figure 1: Klassendiagramm

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

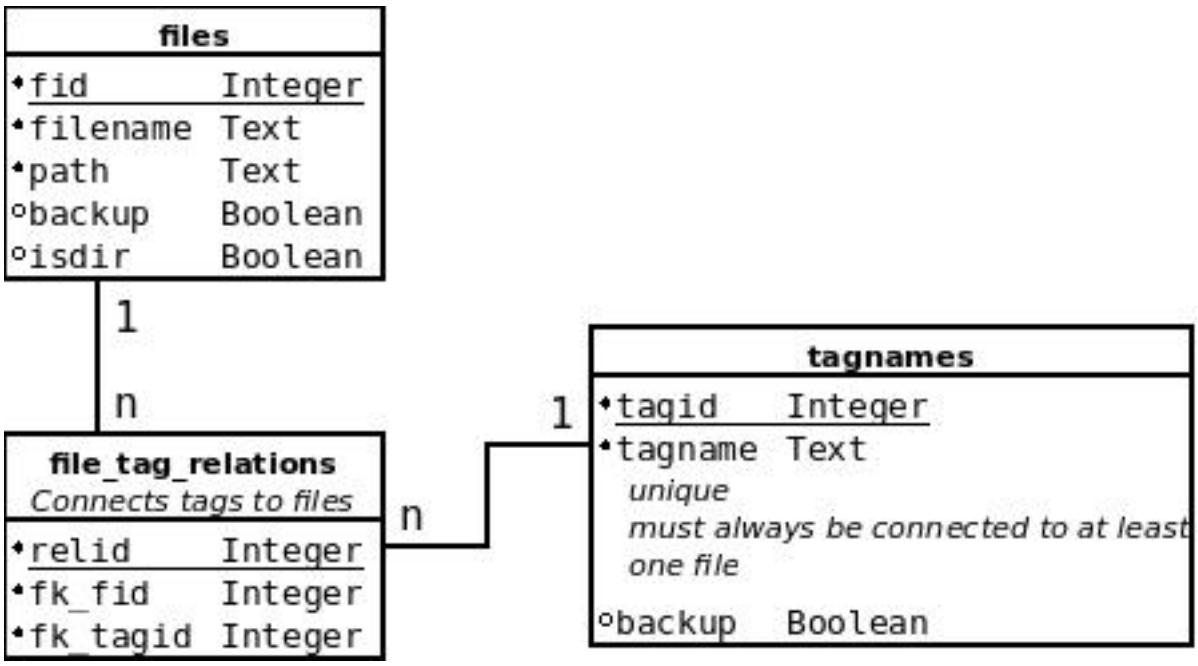

Figure 2: Datenbankschema

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.

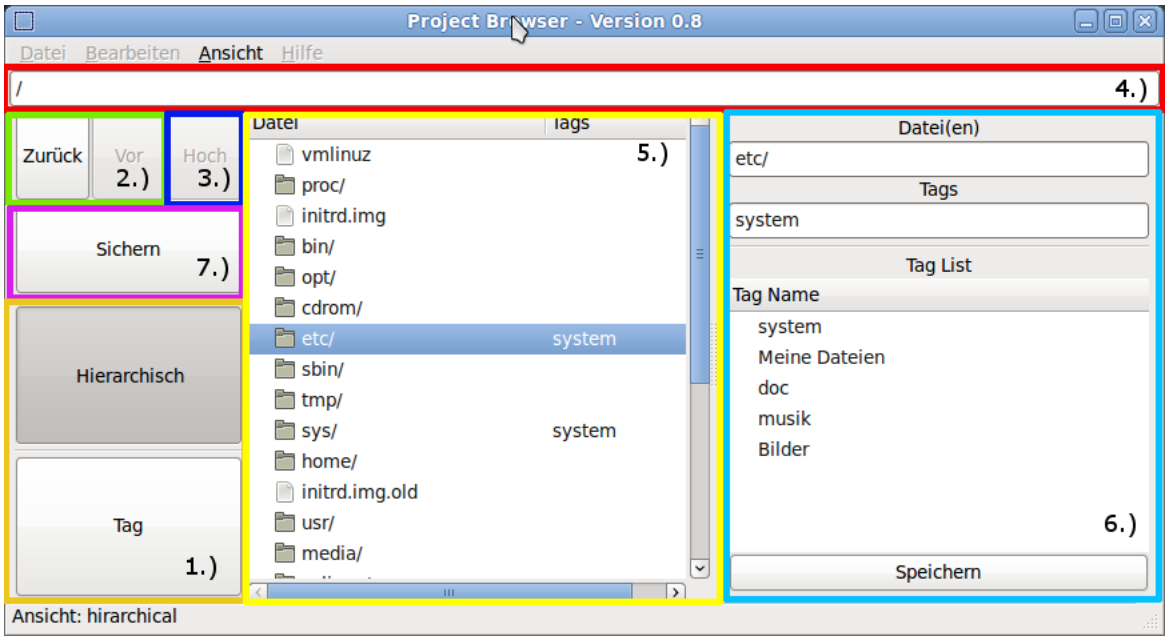

Figure 3: Programmoberfläche

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

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 fr 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 Um den selektierten Ordner oder das selektierte File zu sichern kann man das mit dem Button ’Sichern’ machen. 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.

3.3 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 Tesenriterine

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 Änderung-sevent	
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(".", 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:%S")
38
39 #Dynamisches Objekt, dass fuer normale Objekte gebraucht werden kann
40 class NO():
41         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 class TagManager():
12         def __init__(self,sys):
13                 self.sys = sys
14
15         def searchMatchTags(self,name):
16                 all = self.sys.db.getAllTags()
17                 matched = []
18                 a = name.split(',')
19                 a[len(a)-1].strip()
20                 for i in range(len(all)):
21                         if all[i].find(name) == 0:
22                                 matched.append(all[i])
23                 return matched
```

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()
```

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:
23          pass
24  import threading
25  from File import *
26
27  class FileSystemListener(threading.Thread):
28          listener = None
29          db = None
30          def __init__(self,sys):
31                  threading.Thread.__init__(self)
32                  self.sys = sys
33                  self.db = sys.getDb()
34                  stros = self.sys.u.checkOS()
35                  if stros == "linux":
36                          print("it's a linux!")
37                          self.listener = FileSystemListener_Linux(self)
38                          return
39                  if stros == "win":
40                          print("it's a windows!")
41                          self.listener = FileSystemListener_Windows(self)
42                          return
43                  print "Can't initialize FileSystemListener."
44
45          def add_watch(self, path, rec):
46                  self.listener.add_watch(path, rec)
47
48          def run(self):
49                  self.listener.start()
50
51          def stop(self):
52                  self.listener.stop()
53
54          # Event kommt als String mit dem Dateipfad an.
55          def create_event(self, event):
```

```
56                 self.sys.gui.actview.update()
57                 print "create_event: ", event
58         def delete_event(self, event):
59                 self.db.removeFile(File(event))
60                 print "delete_event: ", event
61         def modify_event(self, event):
62                 print "modify_event: ", event
63         def move_event(self, fr, to):
64                 self.sys.gui.actview.update()
65                 self.db.moveFile(File(fr), File(to))
66                 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

```
#!/usr/bin/python

#File:                  FileSystemListener_Windows.py
#Description:           Diese Klasse kann auf einem Windows das File System ueberwachen
#Author:                Lukas Knoepfel
#Creation Date:         14.4.2011
#
#History:               --Version--     --Date--        --Activities--
#                       0.1             14.4.2011       Grundgeruest erstellt
#                       0.2             19.4.2011       Grundfunktionalitaet erstellt

try:
        import win32file
        import win32con
        import os
except ImportError:
        #shit happens :P
        pass

class FileSystemListener_Windows():
    parent = None
    acpath = None
    hdir = None
    ACTIONS = {
            1 : "Created",
            2 : "Deleted",
            3 : "Modified",
            4 : "Delete",#"Renamed from something",
            5 : "Created"#"Renamed to something"
            }
    FILE_LIST_DIRECTORY = 0x0001

    def __init__(self, tparent):
        self.parent = tparent

    def add_watch(self, path, recur):
        self.acpath = path
        self.hdir = win32file.CreateFile(
            path,
            self.FILE_LIST_DIRECTORY,
            win32con.FILE_SHARE_READ | win32con.FILE_SHARE_WRITE,
            None,
            win32con.OPEN_EXISTING,
            win32con.FILE_FLAG_BACKUP_SEMANTICS,
            None
            )

    def start(self):
        while 1:
            results = win32file.ReadDirectoryChangesW(
                self.hdir,
                1024,
                True,
                win32con.FILE_NOTIFY_CHANGE_FILE_NAME |
                win32con.FILE_NOTIFY_CHANGE_DIR_NAME |
                win32con.FILE_NOTIFY_CHANGE_ATTRIBUTES |
                win32con.FILE_NOTIFY_CHANGE_SIZE |
                win32con.FILE_NOTIFY_CHANGE_LAST_WRITE |
                win32con.FILE_NOTIFY_CHANGE_SECURITY,
                None,
                None
                )

            for action, file in results:
```

```
65                  full_filename = os.path.join(self.acpath, file)
66                  #print full_filename, self.ACTIONS.get(action, "Unknown")
67                  if action == 3:
68                      self.parent.modify_event(full_filename)
69                  elif action == 1 or action == 5:
70                      self.parent.create_event(full_filename)
71                  elif action == 2 or action == 4:
72                      self.parent.delete_event(full_filename)
```

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 AddTag import *
18
19 import gobject
20 gobject.threads_init()
21
22 #Modul um diese Klasse als Seperaten-Prozess zu starten
23 import threading
24 import pygtk
25 pygtk.require('2.0')
26 import gtk
27 import gtk.glade
28 import sys
29
30
31 #class GUI(threading.Thread):
32 class GUI():
33         def __init__(self,sys):
34                 #threading.Thread.__init__(self)
```

```
35                 gobject.threads_init()
36                 self.sys = sys
37                 self.mod = sys.c.startview
38                 self.hview = HirarchicalView(self.sys)
39                 self.tview = TagView(self.sys)
40                 self.actview = None
41 
42         #def run(self):
43         def start(self):
44                 print('run: GUI')
45 
46                 #Init-Window
47                 self.xml = gtk.glade.XML("gui.glade")
48                 self.window = self.xml.get_widget("winMain")
49                 self.window.set_title(self.sys.c.prgname + ' - Version ' + str(self.sys.c.
                       version))
50                 #self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
51                 self.window.connect("destroy", self.stopploop)
52 
53                 #Connect Toogle-Buttons
54                 self.btnHirarchical = self.xml.get_widget('btnHirarchical')
55                 self.btnHirarchical.connect('toggled', self.showHirarchical)
56                 self.btnTag = self.xml.get_widget('btnTag')
57                 self.btnTag.connect('toggled', self.showTag)
58 
59                 #Connect TextInput Field
60                 self.txtEntry = self.xml.get_widget('txtEntry')
61                 #self.txtEntry.connect('key_release_event',self.searchKey)
62                 self.txtEntry.connect('changed',self.searchKey)
63                 self.com = gtk.EntryCompletion()
64                 #com.set_inline_selection(True)
65                 #com.connect('match-selected',self.searchCompletion)
66                 self.txtEntry.set_completion(self.com)
67                 self.listcompl = gtk.ListStore(gobject.TYPE_STRING)
68                 self.com.set_model(self.listcompl)
69                 self.com.set_text_column(0)
70                 self.com.set_popup_set_width(True)
71 
72                 #self.txtEntry.connect('activate',self.searchKey)
73                 #self.txtEntry.connect('backspace',self.searchKey)
74 
75                 #Add Menu-Item by 'Ansicht'
76                 ansichtmenu = gtk.Menu()
77                 ansicht = self.xml.get_widget('mnuAnsicht')
78                 ansicht.set_submenu(ansichtmenu)
79                 self.mnuHirarchical = gtk.RadioMenuItem(None,'Hierarchisch')
80                 self.mnuHirarchical.connect('toggled',self.showHirarchical)
81                 ansichtmenu.append(self.mnuHirarchical)
82                 self.mnuTag = gtk.RadioMenuItem(self.mnuHirarchical,'Tag')
83                 self.mnuTag.connect('toggled',self.showTag)
84                 ansichtmenu.append(self.mnuTag)
85 
86                 #Modify other MenuItems
87                 self.mnuBeenden = self.xml.get_widget('mnuBeenden')
88                 self.mnuBeenden.connect('activate',self.stopploop)
89                 self.mnuDatei = self.xml.get_widget('mnuDatei')
90                 self.mnuBearbeiten = self.xml.get_widget('mnuBearbeiten')
91                 self.mnuHilfe = self.xml.get_widget('mnuHilfe')
92                 self.mnuDatei.set_sensitive(False)
93                 self.mnuBearbeiten.set_sensitive(False)
94                 self.mnuHilfe.set_sensitive(False)
95 
96                 #Init Status Bar
97                 self.Status = self.xml.get_widget('stsStatusBar')
98                 self.Status.push(1,'init')
99 
```

```
100                 #Init Navigation Buttons
101                 self.btnBack = self.xml.get_widget('btnBackward')
102                 self.btnBack.connect('clicked',self.historyBack)
103                 self.btnBack.set_sensitive(False)
104                 self.btnFor = self.xml.get_widget('btnForward')
105                 self.btnFor.connect('clicked',self.historyFor)
106                 self.btnFor.set_sensitive(False)
107                 self.btnUp = self.xml.get_widget('btnUp')
108                 self.btnUp.connect('clicked',self.getParentFolder)
109                 self.btnUp.set_sensitive(False)
110                 self.btnBackup = self.xml.get_widget('btnBackup')
111                 self.btnBackup.connect('clicked',self.makeBackup)
112                 self.btnBackup.set_sensitive(False)
113
114                 #Init-View
115                 self.hspView = self.xml.get_widget('hspView')
116                 self.addTagContent = AddTag(self.sys)
117                 self.hspView.add(self.addTagContent.getWidget())
118                 self.view = self.xml.get_widget('View')
119                 if self.mod == 'hirarchical':
120                         self.actview = self.hview
121                         self.txtEntry.set_text(self.actview.get_actTxtInput())
122                         self.showHirarchical('init')
123                 elif self.mod == 'tag':
124                         self.actview = self.tview
125                         self.txtEntry.set_text(self.actview.get_actTxtInput())
126                         self.showTag('init')
127                 else:
128                         self.actview = self.hview
129                         self.txtEntry.set_text(self.actview.get_actTxtInput())
130                         self.showHirarchical('init')
131                 #self.actview.historyUpdate('user')
132
133                 #Zeigt alles an
134                 self.showall()
135
136                 #Loop damit das Fenster nicht wieder geschlossen wird
137                 self.startloop()
138
139
140
141
142
143         #GUI Functionen
144         def showHirarchical(self,event):
145                 #Toggle Function
146                 if isinstance(event,gtk.RadioMenuItem):
147                         if self.mnuHirarchical.get_active():
148                                 self.btnHirarchical.set_active(True)
149                         else:
150                                 self.btnHirarchical.set_active(False)
151                 else:
152                         if self.btnHirarchical.get_active():
153                                 self.mnuHirarchical.set_active(True)
154                         else:
155                                 #Wenn zweimal die gleiche Ansicht gewaehlt wird wird die
                                        andere Ansicht aktiviert
156                                 if isinstance(self.actview,HirarchicalView) and event != 'init
                                        ':
157                                         self.showTag('init')
158                                         return 0
159                                 self.mnuHirarchical.set_active(False)
160                 if event == 'init':
161                         self.btnHirarchical.set_active(True)
162                         self.mnuHirarchical.set_active(True)
163
```

```
164                 #Init-View
165                 self.changeView(self.hview)
166 
167         def showTag(self,event):
168                 #Toggle Function
169                 if isinstance(event,gtk.RadioMenuItem):
170                         if self.mnuTag.get_active():
171                                 self.btnTag.set_active(True)
172                         else:
173                                 self.btnTag.set_active(False)
174                 else:
175                         if self.btnTag.get_active():
176                                 self.mnuTag.set_active(True)
177                         else:
178                                 #Wenn zweimal die gleiche Ansicht gewaehlt wird wird die
                                        andere Ansicht aktiviert
179                                 if isinstance(self.actview,TagView) and event != 'init':
180                                         self.showHirarchical('init')
181                                         return 0
182                                 self.mnuTag.set_active(False)
183                 if event == 'init':
184                         self.btnTag.set_active(True)
185                         self.mnuTag.set_active(True)
186 
187                 #Init-View
188                 self.changeView(self.tview)
189 
190         def changeView(self,newview):
191                 #Speichert Text Input String in der zuschliessenden View
192                 self.actview.set_actTxtInput(self.txtEntry.get_text())
193                 self.addTagContent.clearAll()
194 
195                 #Schliesst die alte View
196                 #self.view.remove(self.actview)
197                 if self.actview.get_parent() != None:
198                         self.view.remove(self.actview)
199 
200                 #Fuegt die neue View an
201                 #self.view.add(newview)
202                 self.view.add(newview)
203 
204                 #Aender Aktuel View
205                 self.mod = newview.mod
206                 self.actview = newview
207 
208                 #Text Input aktualisieren
209                 #self.listcompl.clear()
210                 self.txtEntry.set_text(self.actview.get_actTxtInput())
211 
212                 #Fokus auf Text Input
213                 self.set_focus(self.txtEntry)
214 
215                 #Update Statusbar
216                 self.Status.pop(1)
217                 self.Status.push(1,'Ansicht:␣'+str(self.mod))
218                 self.showall()
219 
220                 #Update-View
221                 self.actview.update()
222 
223         #def searchKey(self,widget,event):
224         def searchKey(self,a):
225                 #if event.keyval != gtk.gdk.keyval_from_name("Down") and event.keyval != gtk.
                        gdk.keyval_from_name("Up"):
226                 if self.actview.triggeredByNavigation:
227                         self.updateView()
```

```
228                 else:
229                         self.updateView('user')
230
231         def searchCompletion(self,completion, prefix, user_param1):
232                 self.updateView()
233
234         def updateView(self,actor='fn'):
235                 self.actview.set_actTxtInput(self.txtEntry.get_text())
236                 self.actview.update(actor)
237
238         def openDirInHirarchical(self,path):
239                 self.showHirarchical('init')
240                 self.actview.set_actTxtInput(path)
241                 self.listcompl.clear()
242                 self.txtEntry.set_text(self.actview.get_actTxtInput())
243
244         def historyBack(self,event):
245                 self.actview.triggeredByNavigation = True
246                 self.actview.historyCursor = self.actview.historyCursor-1
247                 self.actview.set_actTxtInput(self.actview.history[self.actview.historyCursor])
248                 self.txtEntry.set_text(self.actview.get_actTxtInput())
249                 self.actview.triggeredByNavigation = False
250
251         def historyFor(self,event):
252                 self.actview.triggeredByNavigation = True
253                 self.actview.historyCursor = self.actview.historyCursor+1
254                 self.actview.set_actTxtInput(self.actview.history[self.actview.historyCursor])
255                 self.txtEntry.set_text(self.actview.get_actTxtInput())
256                 self.actview.triggeredByNavigation = False
257
258         def getParentFolder(self,event):
259                 parent=self.sys.filemanager.getParentDir(self.actview.get_actTxtInput())
260                 self.txtEntry.set_text(parent)
261
262         def makeBackup(self,event):
263                 self.actview.backupSelectedObject()
264         ###
265
266
267
268
269
270         def set_focus(self,widget):
271                 self.window.set_focus(widget)
272
273         def showall(self):
274                 self.window.show_all()
275
276         def terminate(self):
277                 # must raise the SystemExit type, instead of a SystemExit() instance
278                 # due to a bug in PyThreadState_SetAsyncExc
279                 self.raise_exc(SystemExit)
280                 pass
281
282         def startloop(self):
283                 gtk.main()
284
285         def stopploop(self,event=''):
286                 print('quite')
287                 gtk.main_quit()
288                 self.sys.stoppall()
```

10.12 TagView.py

```
#!/usr/bin/python

#File:                  TagView.py
#Description:           Klasse um die Datein zu einem Tag anzuzeigen
#Author:                Kaleb Tschabold
#Creation Date:         29.3.2011
#
#History:               --Version--     --Date--        --Activities--
#                       0.1             29.3.2011       Grundfunktionalitaeten werden erstellt

from View import *
import gtk

class TagView(View):
        mod = 'tag'
        def __init__(self,sys):
                View.__init__(self,sys)
                self.set_actTxtInput(sys.c.initStrTag)
                self.connect('row-activated',self.rowActivate)

        def update(self, actor = 'fn'):
                if self.sys.gui != None:
                        oldmodel = self.get_model()
                        self.model.clear()
                        #Dieser Durchgang ist, wenn noch nicht's im TextFeld steht
                        if self.get_actTxtInput() == '':
                                t = self.sys.db.getAllTags()
                                for i in range(len(t)):
                                        self.model.append(None,[self.getFolderIcon(),'*',[t[i
                                            ]],t[i]])
                                self.set_model(self.model)
                                self.historyUpdate(actor)
                                self.historySymboleManagement()
                                self.updateParentFolderBtn()
                                self.completion()
                                #Damit diese Funktion abgebrochen wird
                                return 0
                        #Dieser Durchgang ist, wenn Zeichen im Text Feld stehen
                        itemarray = self.get_actTxtInput().split(',')
                        for j in range(len(itemarray)):
                                oneitem = itemarray[j].strip()
                                self.items = self.sys.db.getFilesFromTag(oneitem)
                                for i in range(len(self.items)):
                                        self.items[i].setIsDir(self.items[i].getIsDir())
                                        if self.items[i].getIsDir():
                                                self.model.append(None,[self.getFolderIcon(),
                                                    self.items[i].getFileName(),self.items[i],
                                                    ',␣'.join(self.items[i].getTags())])
                                        else:
                                                self.model.append(None,[self.getFileIcon(),
                                                    self.items[i].getFileName(),self.items[i],
                                                    ',␣'.join(self.items[i].getTags())])
                        self.set_model(self.model)
                        if len(self.items)>0:
                                self.historyUpdate(actor)
                        self.historySymboleManagement()
                        self.updateBackupBtn()
                        self.updateParentFolderBtn()
                        self.completion()

        def completion(self):
                        matched = self.sys.tagmanager.searchMatchTags(self.get_actTxtInput())
                #try:
                        if self.sys.gui.listcompl != None:
                                #try:
                                        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
70          def updateParentFolderBtn(self):
71                  self.sys.gui.btnUp.set_sensitive(False)
72
73          def rowActivate(self,treeview, path, user_data):
74                  f = self.getFObjFromSelectedRow()
75                  if isinstance(f,list):
76                          print('f: '+str(f[0]))
77                          self.sys.gui.txtEntry.set_text(f[0])
78                          self.sys.gui.updateView()
79                  else:
80                          if not f.getIsDir():
81                                  self.sys.filemanager.openFile(f.getFullPath())
82                          else:
83                                  self.sys.gui.openDirInHirarchical(f.getFullPath())
84
85          def backupSelectedObject(self):
86                  f = self.getFObjFromSelectedRow()
87                  if type(f) == list:
88                          tf = self.sys.db.getFilesFromTag(f[0])
89                          for i in range(len(tf)):
90                                  tf[i].ex_backup()
91                  else:
92                          f.ex_backup()
```

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                              self.set_model(self.model)
43                              #Ruft History Verwaltung auf
44                              if len(self.get_actTxtInput()) > 0:
45                                      if self.get_actTxtInput()[-1] == '/':
46                                              self.historyUpdate(actor)
47                      self.updateParentFolderBtn()
48                      self.updateBackupBtn()
49                      self.historySymboleManagement()
50                      #Ruft Auto-Completion Update Funktion auf
51                      self.completion()
52
53      def completion(self):
54              matched = self.sys.filemanager.searchMatchDir(self.get_actTxtInput())
55              try:
56                      if self.sys.gui.listcompl != None:
57                              try:
58                                      self.sys.gui.listcompl.clear()
59                              except:
60                                      pass
61                      for i in range(len(matched)):
62                              self.sys.gui.listcompl.append([matched[i]])
63              except:
64                      pass
65
66      def updateParentFolderBtn(self):
67              parent = self.sys.filemanager.getParentDir(self.get_actTxtInput())
68              if parent:
69                      self.sys.gui.btnUp.set_sensitive(True)
70              else:
71                      self.sys.gui.btnUp.set_sensitive(False)
72
73      def rowActivate(self,treeview, path, user_data):
74              f = self.getFObjFromSelectedRow()
75              print('fullpath␣activated␣in␣rowActivated:␣'+f.getFullPath())
76              if not f.getIsDir():
77                      self.sys.filemanager.openFile(f.getFullPath())
78              else:
79                      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 AddTag 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
45                 #1. Spalte
46                 self.cl1 = gtk.TreeViewColumn('Datei')
47                 self.append_column(self.cl1)
48
49                 #Definition des Icons der 1. Spalte
50                 render1 = gtk.CellRendererPixbuf()
51                 self.cl1.pack_start(render1, expand=False)
52                 self.cl1.add_attribute(render1, 'pixbuf', 0)
53
54                 #Definition der Text der 1. Spalte
55                 render2 = gtk.CellRendererText()
56                 self.cl1.pack_start(render2,True)
57                 self.cl1.add_attribute(render2,'text',1)
58
59                 #2. Spalte
60                 self.cl2 = gtk.TreeViewColumn("Tags")
61                 self.append_column(self.cl2)
62
63                 #Definition des Textes fuer die 2. Spalte
64                 render3 = gtk.CellRendererText()
65                 self.cl2.pack_start(render3,True)
66                 self.cl2.add_attribute(render3,'text',3)
67
68                 #Allgemeine Definitionen fuer den Baum
69                 #self.set_search_column(1)
70                 #self.cl1.set_sort_column_id(0)
71                 #self.cl2.set_sort_column_id(1)
72
73                 self.update()
74
```

```
75          def get_icon_pixbuf(self, stock):
76                  return self.render_icon(stock_id=getattr(gtk, stock),
77                                          size=gtk.ICON_SIZE_MENU,
78                                          detail=None)
79          def getFolderIcon(self):
80                  return self.get_icon_pixbuf('STOCK_DIRECTORY')
81
82          def getFileIcon(self):
83                  return self.get_icon_pixbuf('STOCK_FILE')
84
85          def update(self):
86                  self.show_all()
87
88          def set_actTxtInput(self,text):
89                  self.acttxtinput = text
90
91          def get_actTxtInput(self):
92                  return self.acttxtinput
93
94          def getFObjFromSelectedRow(self):
95                  treeview = self
96                  selection = treeview.get_selection()
97                  selection.set_mode(gtk.SELECTION_SINGLE)
98                  tree_model, tree_iter = selection.get_selected()
99                  return tree_model.get_value(tree_iter,2)
100
101         def showContext(self, treeview, event):
102                 if event.button == 3:
103                         f = self.getFObjFromSelectedRow()
104                         m = gtk.Menu()
105                         #m1 = gtk.MenuItem ('Add Tag')
106                         #m.append(m1)
107                         m2 = gtk.MenuItem('Properties')
108                         m.append(m2)
109                         #m1.connect('button_press_event',self.context_AddTag,f)
110                         m2.connect('button_press_event',self.context_Properties,f)
111                         m.show_all()
112                         m.popup( None, None, None, event.button, event.time)
113                         return True
114                 return False
115
116         def context_Properties(self,widget,event,fobj):
117                 print('properties')
118
119         def updateTagProperties(self,event):
120                 try:
121                         self.updateBackupBtn()
122                         self.sys.gui.addTagContent.update(self.getFObjFromSelectedRow())
123                 except:
124                         pass
125
126         def updateBackupBtn(self):
127                 try:
128                         if(isinstance(self.getFObjFromSelectedRow(),File) or type(self.
                                getFObjFromSelectedRow()) == list):
129                                 self.sys.gui.btnBackup.set_sensitive(True)
130                         else:
131                                 self.sys.gui.btnBackup.set_sensitive(False)
132                 except:
133                         self.sys.gui.btnBackup.set_sensitive(False)
134
135         def backupSelectedObject(self):
136                 f = self.getFObjFromSelectedRow()
137                 f.ex_backup()
138
139         def historyUpdate(self,actor='fn'):
```

```
140                 #Wenn wieder vorwaertz gesprungen wird, werden alle Element nach der aktuelen
                        Position geloescht
141                 if self.historyCursor > -1:
142                         if actor == 'user' and self.history[self.historyCursor] != self.
                            get_actTxtInput():
143                             if self.historyCursor < len(self.history)-1:
144                                     rem = (len(self.history)-1) - self.historyCursor
145                                     l = len(self.history)
146                                     i = 1
147                                     #rem+1 weil die len von history eins mehr ist als der
                                        cursor
148                                     while i<rem+1:
149                                             self.history.remove(self.history[l-i])
150                                             i = i + 1
151                                     self.historyCursor = self.historyCursor
152                             self.historyCursor = self.historyCursor + 1
153                             self.history.append(self.get_actTxtInput())
154                             self.historySymboleManagement()
155                 elif len(self.history) == 0:
156                         self.historyCursor = self.historyCursor + 1
157                         self.history.append(self.get_actTxtInput())
158                         self.historySymboleManagement()
159
160         def historySymboleManagement(self):
161                         h = len(self.history)-1
162                         c = self.historyCursor
163                         if h >= c and h != 0 and c > 0:
164                                 self.sys.gui.btnBack.set_sensitive(True)
165                         if h >= c and h != 0:
166                                 self.sys.gui.btnFor.set_sensitive(True)
167                         if c <= 0 or h == 0:
168                                 self.sys.gui.btnBack.set_sensitive(False)
169                         if c == h or h == 0:
170                                 self.sys.gui.btnFor.set_sensitive(False)
171
172
173 #Registriert diese Klasse als pygtk-widget
174 gobject.type_register(View)
```

10.15 File.py

```
1  #!/usr/bin/python
2
3  #File:                  File.py
4  #Description:           Jede Datei die im Programm bearbeitet wird wir in einer solchen Klasse
       referenziert
5  #Author:                Kaleb Tschabold
6  #Creation Date:         14.4.2011
7  #
8  #History:               --Version--     --Date--        --Activities--
9  #                       0.1             14.04.2011      Grundfunktionalitaeten werden erstellt
10 #                       0.9             21.04.2011      Created methods and functionality.
       Added __doc__s. added test (bottom)
11
12 import re
13 import os
14 from Constant import *
15 from Utility import *
16 from shutil import copytree, ignore_patterns, copyfile
17
18 class File:
19         """represents a file on the filesysem"""
20         fileName        = None
21         path            = None
22         isDir           = None
```

```
23          tags            = None
24          backup          = None
25          fullPath        = None
26          u               = None
27          constant        = None
28          os              = None
29          def __init__(self, fullPath=None, fileName=None, path=None, tags=[], backup=False,
                isDir=False):
30              """Constructor
31              @param  fileName, string        , optional
32              @param  path    , string        , optional
33              @param  tags    , list          , optional
34              @param  backup  , boolean       , optional
35              """
36              self.fileName   = fileName
37              self.path       = path
38              self.tags       = tags
39              self.backup     = backup
40              self.isDir      = isDir
41              self.fullPath   = fullPath
42
43              self.u = Utility()
44              self.constant = Constant(self)
45              self.os         = self.constant.os
46
47              self.__splitFullPath()
48              self.__checkPaths()
49
50          def __splitFullPath(self):
51              if not self.fullPath == None:
52                  if self.fullPath.endswith("/") or self.fullPath.endswith("\\"):
53                      p = re.compile("((?:.*\\\)|(?:.*/))((?:[^/]*)|(?:[^\\\]*))
                            ([/|\\\])")
54                      match = p.match(self.fullPath)
55                      if not match.group(1) == None and not match.group(2) == None:
56                          self.path = match.group(1)
57                          self.fileName = match.group(2)+match.group(3)
58                  else:
59                      p = re.compile("((?:.*\\\)|(?:.*/))((?:[^/]*)|(?:[^\\\]*))")
60                      match = p.match(self.fullPath)
61                      if not match.group(1) == None and not match.group(2) == None:
62                          self.path = match.group(1)
63                          self.fileName = match.group(2)
64
65          def __checkPaths(self):
66              # Check wether all paths have / (or \ on Windows) at the end
67              # This is highly experimental! Tell me if you have any trouble with it!
68              if self.os == 'linux' or self.os == 'mac':
69                  if not self.path == None:
70                      if not self.path.endswith("/") and not self.path.endswith("\\"
                            ):
71                          self.path = self.path+"/"
72                  if not self.fileName == None:
73                      if self.isDir == True and not self.fileName.endswith("/") and
                            not self.fileName.endswith("\\"):
74                          self.fileName = self.fileName+"/"
75              elif self.os == 'win':
76                  if not self.path == None:
77                      if not self.path.endswith("\\") and not self.path.endswith("/"
                            ):
78                          self.path = self.path+"\\"
79                  if not self.fileName == None:
80                      if self.isDir == True and not self.fileName.endswith("\\") and
                             not self.fileName.endswith("/"):
81                          self.fileName = self.fileName+"\\"
82
```

```
83
84          def setFileName(self, fileName):
85                  """@param       filename        , string"""
86                  self.fileName   = fileName
87                  self.__checkPaths()
88
89          def setPath(self, path):
90                  """@param       path    , string"""
91                  self.path       = path
92                  self.__checkPaths()
93
94          def setIsDir(self,b):
95                  """@param       b       , boolean """
96                  self.isDir = b
97                  self.__checkPaths()
98
99          def setTags(self, tags):
100                 """@param       tags    , list"""
101                 self.tags       = tags
102
103         def setBackup(self, backup):
104                 """@param backup        , boolean"""
105                 self.backup     = backup
106
107         def setFullPath(self, fullPath):
108                 """@param fullPath      , Path including filename"""
109                 self.fullPath   = fullPath
110                 self.__splitFullPath()
111                 self.__checkPaths()
112
113         def getFileName(self):
114                 """@return fileName, string"""
115                 return self.fileName
116
117         def getPath(self):
118                 """@return path, string"""
119                 return self.path
120
121         def getIsDir(self):
122                 """@return isDir, boolean """
123                 return self.isDir
124
125         def getTags(self):
126                 """@return tags, list"""
127                 return self.tags
128
129         def getBackup(self):
130                 """@return backup, boolean"""
131                 return self.backup
132
133         def getFullPath(self):
134                 """@return fullPath, String representing the full path to the file, including
                        the file's name"""
135                 if self.fullPath == None:
136                         if not self.path==None or not self.fileName==None:
137                                 return self.path + self.fileName
138                         else:
139                                 print "either path or fileName (or both) are not specified.
                                    You can't use this function unless both are specified"
140                 else:
141                         return self.fullPath
142
143         def addTag(self, tag):
144                 """Adds a single tag (passed as string) to the list of tags.
145                 uses .append"""
146                 self.tags.append(tag)
```

```
147
148         def addTags(self, tags):
149                 """Adds a list of tags (passed as list) to the list of tags.
150                 uses .extend"""
151                 self.tags.extend(tags)
152
153         def ex_backup(self):
154                 print "FullPath", self.getFullPath()
155                 dir = self.getPath() + ".pb_backup"
156                 if not os.path.exists(dir):
157                         os.makedirs(dir)
158                 bdir = dir + "/" + self.u.getTime()
159                 if not os.path.exists(bdir):
160                         os.makedirs(bdir)
161                 if (os.path.isdir(self.getFullPath())):
162                         copytree(self.getFullPath(), bdir + "/" + self.getFileName(), ignore=
                            ignore_patterns('.*', '*.pyc'))
163                 else:
164                         copyfile(self.getFullPath(), bdir + "/" + self.getFileName())
165
166
167 if __name__ == "__main__":
168         print "Starting tests"
169         print "Note: Tests 9,10,14,15 will fail on Windows... and so will many others probably
                "
170         print "=============="
171         x = File()
172         if x.getFileName() == None:
173                 print "Test #01: Succeed"
174         else:
175                 print "Test #01: FAIL"
176
177         x.setFileName("test")
178         if x.getFileName() == "test":
179                 print "Test #02: Succeed"
180         else:
181                 print "Test #02: FAIL"
182
183         x.setTags(["test", "tag"])
184         if x.getTags() == ["test","tag"]:
185                 print "Test #03: Succeed"
186         else:
187                 print "Test #03: FAIL"
188
189         x.addTag("new tag")
190         if x.getTags()[2] == "new tag":
191                 print "Test #04: Succeed"
192         else:
193                 print "Test #04: FAIL"
194
195         x.addTags(["extend","linux","unix"])
196         if x.getTags()[3:] == ["extend","linux","unix"]:
197                 print "Test #05: Succeed"
198         else:
199                 print "Test #05: FAIL"
200
201         x.setIsDir(True)
202         if x.getIsDir():
203                 print "Test #06: Succeed"
204         else:
205                 print "Test #06: FAIL"
206
207         #Testing the path stuff...
208         f1 = File(fullPath="/home/niklaus/lol.txt")
209         if f1.getPath() == "/home/niklaus/" \
210             and f1.getFileName() == "lol.txt" and f1.getFullPath() == "/home/niklaus/lol.txt":
```

```
211                 print "Test #07: Succeed"
212         else:
213                 print "Test #07: FAIL"
214 
215         f2 = File(fullPath="C:\\windows\\system32\\chlous.txt.test")
216         if f2.getPath() == "C:\\windows\\system32\\" \
217             and f2.getFileName() == "chlous.txt.test" and f2.getFullPath() == "C:\\windows\\
                system32\\chlous.txt.test":
218                 print "Test #09: Succeed"
219         else:
220                 print "Test #09: FAIL"
221 
222         f3 = File(fullPath="C:\\windows\\system32\\")
223         if f3.getPath() == "C:\\windows\\" \
224             and f3.getFileName() == "system32\\" and f3.getFullPath() == "C:\\windows\\
                system32\\":
225                 print "Test #10: Succeed"
226         else:
227                 print "Test #10: FAIL"
228 
229         f4 = File("/home/niklaus/Music/")
230         if f4.getPath() == "/home/niklaus/" \
231             and f4.getFileName() == "Music/" and f4.getFullPath() == "/home/niklaus/Music/":
232                 print "Test #11: Succeed"
233         else:
234                 print "Test #11: FAIL"
235 
236         f5 = File(fileName="document.odt", path="/home/niklaus/Documents/")
237         if f5.getFullPath() == "/home/niklaus/Documents/document.odt":
238                 print "Test #12: Succeed"
239         else:
240                 print "Test #12: FAIL"
241 
242         f6 = File(fileName="Videos/", path="/home/niklaus/", isDir=True)
243         if f6.getFullPath() == "/home/niklaus/Videos/" and f6.getIsDir() == True:
244                 print "Test #13: Succeed"
245         else:
246                 print "Test #13: FAIL"
247 
248         f7 = File(fileName="hosts", path="C:\\windows\\system32\\etc\\")
249         if f7.getFullPath() == "C:\\windows\\system32\\etc\\hosts":
250                 print "Test #14: Succeed"
251         else:
252                 print "Test #14: FAIL"
253 
254         f8 = File(fileName="etc\\", path="C:\\windows\\system32\\", isDir=True)
255         if f8.getFullPath() == "C:\\windows\\system32\\etc\\" and f8.getIsDir() == True:
256                 print "Test #15: Succeed"
257         else:
258                 print "Test #15: FAIL"
259 
260         #Testing the automatic path correction
261         f9 = File(fileName="ozzed.ogg", path="/home/niklaus/Music")
262         if f9.getPath() == "/home/niklaus/Music/" and f9.getFileName() == "ozzed.ogg":
263                 print "Test #16: Succeed"
264         else:
265                 print "Test #16: Fail"
266 
267         f10 = File(fileName="Music", path="/home/niklaus", isDir=True)
268         if f10.getPath() == "/home/niklaus/" and f10.getFileName() == "Music/":
269                 print "Test #17: Succeed"
270         else:
271                 print "Test #17: FAIL"
272 
273         f11 = File(fileName="Music", isDir=True)
274         if f11.getFileName() == "Music/" and f11.getPath() == None:
```

```
275                 print "Test #18: Succeed"
276         else:
277                 print "Test #18: FAIL"
278
279         f12 = File(path="/home/niklaus/somepath")
280         if f12.getPath() == "/home/niklaus/somepath/" and f12.getFileName() == None:
281                 print "Test #19: Succeed"
282         else:
283                 print "Test #19: FAIL"
284
285         f13 = File()
286         f13.setFullPath("/home/niklaus/File.php")
287         if f13.getPath() == "/home/niklaus/" and f13.getFileName() == "File.php":
288                 print "Test #20: Succeed"
289         else:
290                 print "Test #20: FAIL"
291
292         f14 = File(isDir=True)
293         f14.setFullPath("/home/niklaus/Music")
294         if f14.getPath() == "/home/niklaus/" and f14.getFileName() == "Music/":
295                 print "Test #21: Succeed"
296         else:
297                 print "Test #21: FAIL"
298
299         print File.__init__.__doc__
300
301         #Backup
302         #b = File("/home/laden")
303         #b.ex_backup()
```

10.16 AddTag.py

```
1  import gtk
2  import gobject
3  class AddTag():
4          def __init__(self,sys):
5                  self.sys = sys
6                  self.fobj = None
7                  self.xml = gtk.glade.XML("addTag.glade")
8                  self.main = self.xml .get_widget("vbxMain")
9                  self.txtFiles = self.xml.get_widget("txtFileNames")
10                 self.txtTags = self.xml.get_widget("txtTags")
11                 self.txtTags.connect('key-release-event',self.enterEventHandler)
12                 #self.txtTags.connect('drag_data_received', self.test)
13                 self.btnSave = self.xml.get_widget("btnSave")
14                 self.btnSave.connect('button_release_event',self.save)
15                 self.con = self.xml.get_widget("conTagList")
16
17                 #Model Definition
18                 self.model = gtk.TreeStore(gobject.TYPE_STRING)
19
20                 #Tree Definition
21                 self.tree = gtk.TreeView(self.model)
22                 #self.tree.drag_source_set(gtk.gdk.BUTTON1_MASK, self.model, gtk.gdk.
                       ACTION_DEFAULT)
23                 self.tree.connect('row-activated',self.addClickedTag)
24                 self.con.add(self.tree)
25
26                 #Spalte 1
27                 self.cl1 = gtk.TreeViewColumn('Tag Name')
28                 self.tree.append_column(self.cl1)
29
30                 #Definition der Text der 1. Spalte
31                 render = gtk.CellRendererText()
32                 self.cl1.pack_start(render)
```

```
33                  self.cl1.add_attribute(render,'text',0)
34
35                  #Allgemeine Tree Definitionen
36                  #self.tree.set_search_column(0)
37                  #self.cl1.set_sort_column_id(0)
38
39                  self.updateModel()
40                  self.main.show_all()
41
42          def test(self):
43                  print('test')
44
45          def update(self,fobj):
46                  self.clearAll()
47                  self.fobj = None
48                  self.fobj = fobj
49                  self.txtTags.set_text(', '.join(self.fobj.getTags()))
50                  self.txtFiles.set_text(self.fobj.getFileName())
51                  self.updateModel()
52
53          def updateModel(self):
54                  self.model.clear()
55                  for tag in self.sys.db.getAllTags():
56                          self.model.append(None,[tag])
57                  pass
58
59          def clearAll(self):
60                  self.fobj = None
61                  self.model.clear()
62                  self.txtFiles.set_text('')
63                  self.txtTags.set_text('')
64                  self.updateModel()
65
66          def getWidget(self):
67                  return self.main
68
69          def addClickedTag(self,treeview, path, user_data):
70                  selection = treeview.get_selection()
71                  selection.set_mode(gtk.SELECTION_SINGLE)
72                  tree_model, tree_iter = selection.get_selected()
73                  tagname = tree_model.get_value(tree_iter,0)
74                  if self.txtTags.get_text() == '':
75                          self.txtTags.set_text(tagname)
76                  else:
77                          self.txtTags.set_text(self.txtTags.get_text()+', '+tagname)
78
79          def enterEventHandler(self,widget,event):
80                  if event.keyval == gtk.gdk.keyval_from_name("Return"):
81                          self.save(widget,event)
82
83          def save(self,widget,event):
84                  print('save')
85                  tags = self.txtTags.get_text().split(',')
86                  for i in range(len(tags)):
87                          if tags[i].strip() != '':
88                                  tags[i] = tags[i].strip()
89                          else:
90                                  tags.remove(tags[i])
91                  tags = list(set(tags))
92                  if len(tags) != 0:
93                          self.fobj.setTags(tags)
94                          self.sys.db.updateFile(self.fobj)
95                  self.updateModel()
96                  self.sys.gui.actview.update()
```