

# **Realisierungsbericht**

Niklaus Hofer, Lukas Knöpfel, Kaleb Tschabold

May 30, 2011

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

### Änderungskontrolle, Prüfung, Genehmigung

Version	Datum	Beschreibung, Bemerkung	Name oder Rolle
0.1	08.02.2011	Gesammelten Text einfügen	Kaleb Tschabold
0.9	08.02.2011	Abgabebereit	Kaleb Tschabold
0.99	May 30, 2011	Transfer nach $\text{\LaTeX}$	Niklaus Hofer
1.0	May 30, 2011	Klassendiagramm updaten, Sequenzdiagramm einfügen, Rechtschreibfehler korrigieren, Benutzerhandbuch updaten	Niklaus Hofer

### Definitionen und Abkürzungen

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

### References

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

## Contents

<b>1</b>	<b>Zweck des Dokuments</b>	<b>5</b>
<b>2</b>	<b>Technische Detailspezifikation</b>	<b>5</b>
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	8
2.2	Schnittstellendefinition	8
2.3	Datenmodell	9
2.3.1	Datenbank	9
2.3.2	File-object	9
2.4	Sicherheit	10
2.5	Anforderungszuordnung	10
<b>3</b>	<b>Systemdokumentation</b>	<b>11</b>
3.1	Inline-Dokumentation	11
3.2	Benutzerhandbuch	11
3.2.1	Systemübersicht	11
3.2.1.1	Aufgabengebiet des Programms	11
3.2.1.2	Programmoberfläche	11
3.2.1.3	Anmerkungen zur korrekten Verwendung unter dem Aspekt der Sicherheit	11
3.2.2	Anwenderfunktionalität	12
3.2.2.1	Ansicht wechseln	13
3.2.2.2	Verzeichnis wechseln	13
3.2.2.3	Hisotry	13
3.2.2.4	Dateien Öffnen	13
3.2.2.5	Tags hinzufügen	13
3.2.2.6	Tags entfernen	13
3.2.2.7	Dateien anhand der Tags durchsuchen	13
3.2.2.8	Fehlermeldungen	13
3.2.2.9	Sichern	14
3.2.2.10	Wiederherstellen	14
3.3	Supporthandbuch	14
3.3.1	Massnahmen bei Benutzerproblemen	14
3.3.2	Massnahmen bei technischen Problemen	14
3.3.3	Anhang zum Supporthandbuch	14
<b>4</b>	<b>Systemtest</b>	<b>14</b>
4.1	Testspezifikation	14
4.1.1	Kritikalität der Funktionseinheit	14
4.1.2	Testanforderungen	15
4.1.3	Testverfahren	15
4.1.4	Testkriterien	15
4.1.5	Testfälle	15
4.2	Testprozedur	17
4.2.1	Vorbereitung	17
4.2.1.1	Voraussetzungen	17
4.2.1.2	Konfiguration	17
4.2.2	Durchführung	17
4.2.3	Nachbearbeitung	17
4.3	Testprotokoll	18
4.3.1	Testobjekt	18
4.3.2	Testresultate	18
4.3.3	Testauswertung	18
<b>5</b>	<b>Mittelbedarf</b>	<b>18</b>

<b>6 Planung und Organisation</b>	<b>18</b>
<b>7 Wirtschaftlichkeit</b>	<b>18</b>
<b>8 Konsequenzen</b>	<b>19</b>
<b>9 Antrag auf Freigabe der nächsten Projektphase</b>	<b>19</b>
<b>10 Sourcecode</b>	<b>19</b>
10.1 Main.py . . . . .	19
10.2 DB.py . . . . .	21
10.3 Utility.py . . . . .	28
10.4 CLI.py . . . . .	29
10.5 TagManager.py . . . . .	30
10.6 FileManager.py . . . . .	31
10.7 FileSystemListener.py . . . . .	33
10.8 FileSystemListener Linux.py . . . . .	34
10.9 FileSystemListener Windows.py . . . . .	35
10.10 FileSystemListener Mac.py . . . . .	36
10.11 GUI.py . . . . .	36
10.12 TagView.py . . . . .	40
10.13 HirarchicalView.py . . . . .	42
10.14 View.py . . . . .	43
10.15 File.py . . . . .	46
10.16 AddTag.py . . . . .	51

## 1 Zweck des Dokuments

Wir hatten jetzt einige Wochen Zeit um an der Realisierung 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 Dokument 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 werden könnten. Für ein Programm, dessen Hauptaufgabe gerade die Verwaltung der Tags darstellt, ist diese Art Tags Dateien 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 Dateien 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 auszuführen. 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 Dateiname unten im mittleren Panel). Diese getrennte Speicherung hat aber zu der Frage geführt ob / (oder in Windows) am Ende der Pfadangabe mitgespeichert werden sollte. Wir entschieden uns dafür, damit Pfade ohne weiteren Aufwand vollständig zusammengesetzt werden können. Ist der 'Dateiname' der Name eines neuen Verzeichnisses, so trägt auch er ein / (oder in Windows) am Ende.



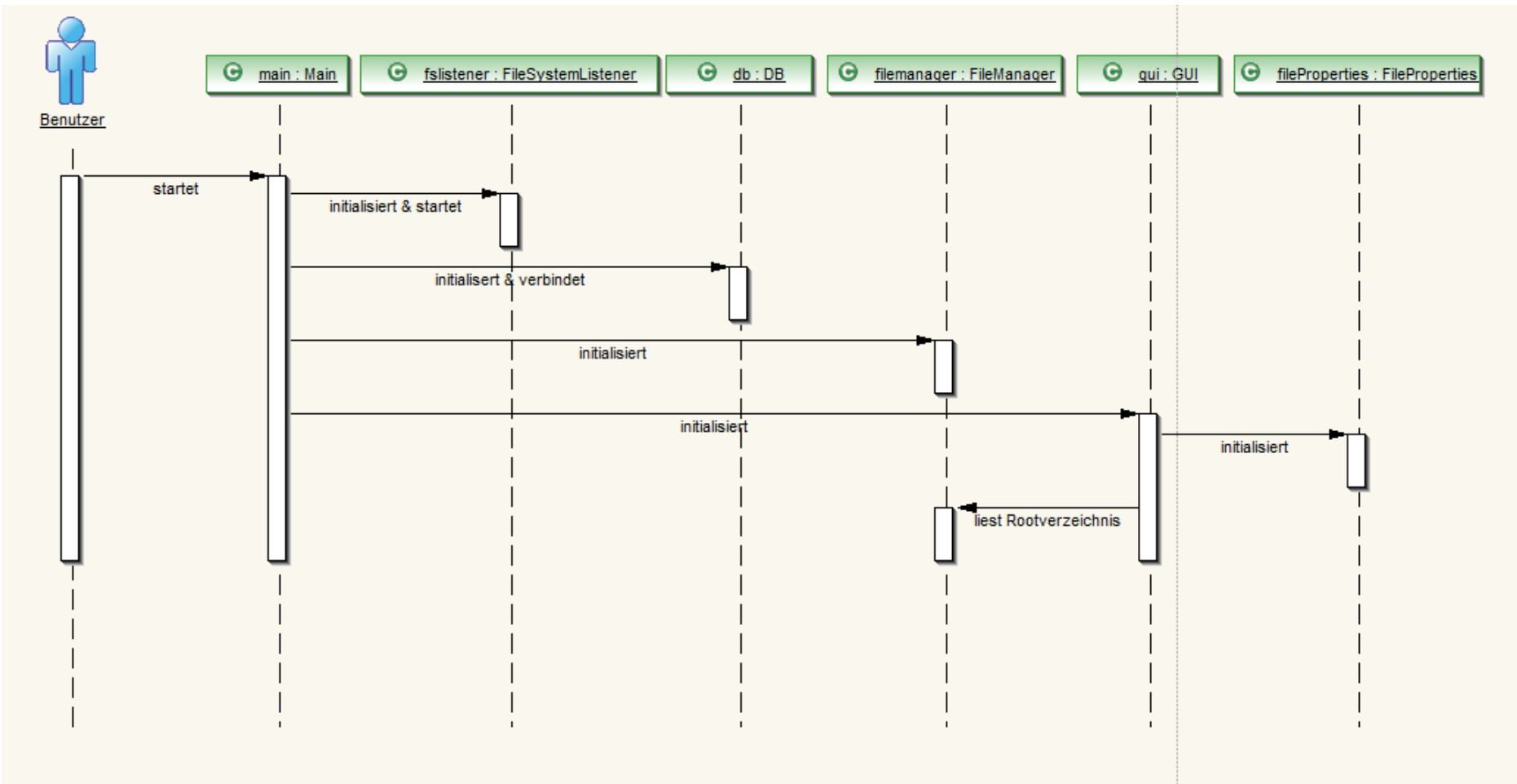


Figure 2: Sequenzdiagramm zum Verdeutlichen des Startprozesses.

### 2.1.3 Beschreibung der Elemente

newpage

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

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

Als Datenbank-backend spricht DB.py SQLite3 an.

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

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

**TagManager** Der Tag Manger ist für kleine Tag Verwaltungsaufgaben zuständig.

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

**FileSystemListener** Registriert beim Kernel Listener für zu überwachende Ordner. Wird in diesen Ordnern eine Operation ausgeführt (wie das Verschieben, Löschen, Umbenennen oder Erstellen einer Datei), so wird der FileSystemListener vom Kernel darüber in Kenntnis gesetzt, woraufhin er wiederum die nötigen Aktionen auslöst um die Datenbank auf dem aktuellen Stand zu halten.

Dies soll dazu beitragen, dass möglichst selten Dateien angezeigt werden, die auf Dateisystem-Ebene nicht existieren.

**GUI** Ist für die grafische Darstellung des Programms mittels GTK zuständig. Das GUI ist in der Lage je nach Bedarf eine andere 'View' darzustellen. Direkt nach dem Programmstart wird HierarchicalView dargestellt. Im Betrieb kann jederzeit zwischen 'TagView' und 'HierarchicalView' 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 hierarchischen Ansicht und wird von 'GUI' bei Bedarf geladen.

**View** Mutterklasse von GUI.

**File** Repräsentiert eine Datei und wird benutzt um Informationen über Dateien zwischen den Elementen des Programms auszutauschen. Für 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 herkömmlichen Dateimanagern mit hierarchischer Ansicht entspricht und
  - ii. Einen Tagmodus, in dem sich Dateien anhand deren Tags durchsuchen und ordnen lassen.



- b) Für scripting oder für solche Systeme ohne grafische Ausgabe haben wir eine CLI Version. Es wird besonders Wert auf das einfache Aufrufen von anderen Programmen (Scripts) gelegt. (nicht implementiert)
  - i. Die Kommandos sollen in der Bedienung weitgehend mit dem standard Unix-Tools kompatibel

## 2.3 Datenmodell

### 2.3.1 Datenbank

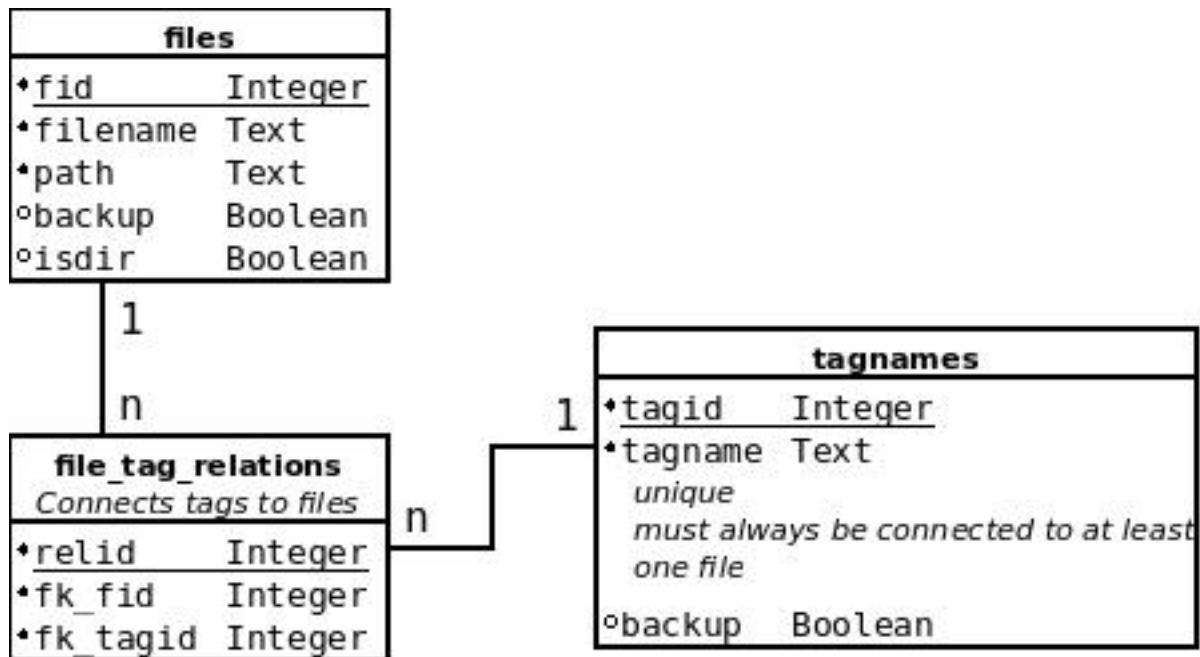


Figure 3: Datenbankschema

Die Datenbank lässt sich über DB.py ansprechen, siehe Schnittstellendefinition für mehr Informationen. Welche Felder genau welche Information enthalten ist im 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 wird dieser mittels Regex zerlegt und die Werte in fileName und path abgelegt.	files.path + files.filename
tags	list[String]	Enthält 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 lokalen Dateisystem und stellt nach Aussen (über das Netzwerk) keine Schnittstelle zur Verfügung.

Für 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 nahtlos 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, sondern 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 eingegeben 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 navigiert 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 Systemadministrator verändert worden ist. Es ist zudem zu beachten, dass der Systemadministrator zu jeder Zeit Zugriff auf die Daten hat. Werden im Program sehr persönliche Informationen 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 unberechtigter Zugriff auf die Dateien geschehen kann.

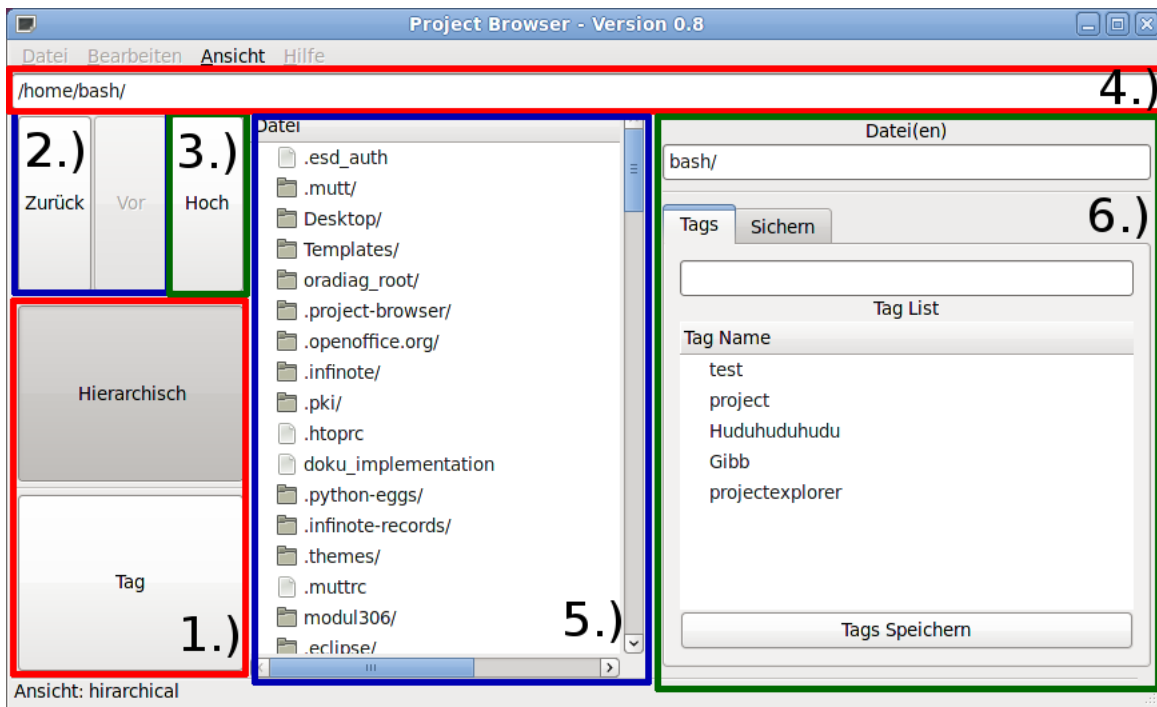


Figure 4: Programmoberfläche

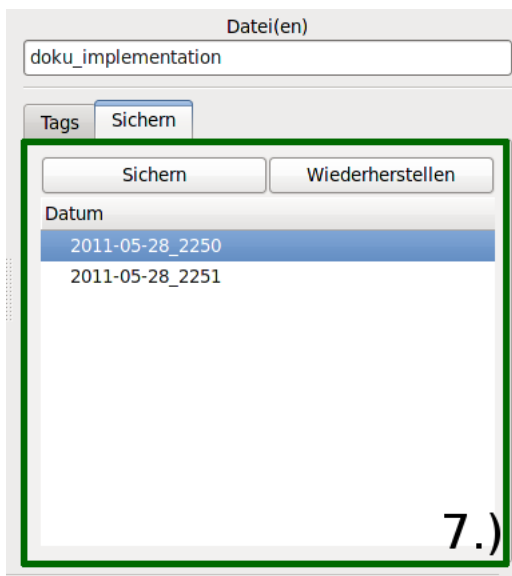


Figure 5: Programmoberfläche zum Verwalten der Backups.

### 3.2.2 Anwenderfunktionalität

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

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

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

**3.2.2.1 Ansicht wechseln** Projekt-Explorer hat zwei verschiedene Ansichten, nämlich 'Hierarchisch' und 'Tags'. Beim Wechseln zwischen den Ansichten ändert sich die Ansicht #5.). Die Ansicht kann entweder über 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 erscheint ein Drop-Down mit Verzeichnisschlägen das während des Tippens laufend 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 oberhalb zu wechseln, kann der Knopf "UP" in #3.) geklickt werden. Um in ein Unterverzeichnis zu öffnen, führt man einen Doppelklick auf den entsprechenden Eintrag in #5.) aus.

**3.2.2.3 History** 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 Programm geöffnet, das auf dem Betriebssystem als Standardapplikation für 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.) ausgewählt werden. Das geschieht über 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, woraufhin 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 Eintrag aus der Liste 'Tags' in #6.) und drückt speichern. Ist der Tag mit keiner weiteren Datei verbunden, so verschwindet er aus der Auswahlliste in #6.).

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

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

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

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

### 3.3 Supporthandbuch

#### 3.3.1 Massnahmen bei Benutzerproblemen

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

#### 3.3.2 Massnahmen bei technischen Problemen

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

#### 3.3.3 Anhang zum Supporthandbuch

## 4 Systemtest

### 4.1 Testspezifikation

#### 4.1.1 Kritikalität der Funktionseinheit

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

#### 4.1.2 Testanforderungen

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

#### 4.1.3 Testverfahren

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

#### 4.1.4 Testkriterien

Abdeckungsgrad: Die Tests umfassen alle Funktionen die das GUI anbietet.

Checklisten: siehe Tabelle unten

Endkriterien: Der Test ist erfolgreich verlaufen wenn das Programm 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 eingegeben 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 hinzugefügt, 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 erfolgreich zugewiesen und bleibt erhalten.	OK
7	6	ein Projekt soll Versioniert werden	Projektdateien 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 wird 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 Änderungsereignis	
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 zuerst 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 fehlgeschlagen ist, dann ist das in der Zeile des betreffenden Test als Bemerkung angefügt.

#### 4.3.3 Testauswertung

Unter Mac hat der FileSystemListener nicht funktioniert, weil er bis zum jetzigen 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
11 #                    0.3            23.04.2011      Will now autocreate tables if
12 #                    nonexistence
13
14 import sqlite3
15 import os.path
16 import File
17
18 class DB:
19     #TODO setBackup
20     connection = None
21     cursor = None
22     dbpath = None
23     def __init__(self, dbpath):
24         self.dbpath = dbpath
25         self.__establishConnection()
26         self.connection.text_factory = str
27
28     def __establishConnection(self):
29         #print self.dbpath
30         self.connection = sqlite3.connect(self.dbpath, check_same_thread = False)
31         self.cursor = self.connection.cursor()
32         print "connection established"
33         #Check wether the tables in the DB exist. If they don't, we'll create 'em
34         self.cursor.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='files'")
35         if len(self.cursor.fetchall()) != 1:
36             print "DB's empty. Creating tables"
37             self.__createDB()
38
39     def __createDB(self):
40         self.cursor.execute("CREATE TABLE files(fid INTEGER PRIMARY KEY, filename TEXT, path TEXT, backup BOOLEAN, isdir BOOLEAN)")
41         self.cursor.execute("CREATE TABLE tagnames(tagid INTEGER PRIMARY KEY, tagname TEXT UNIQUE, backup BOOLEAN)")
42         self.cursor.execute("CREATE TABLE file_tag_relations(relid INTEGER PRIMARY KEY, fk_fid INTEGER, fk_tagid INTEGER)")
43         self.connection.commit()
44
45     def __changeList(self, li):
46         ret_value = []
47         for row in li:
48             ret_value.append(row[0])
49         return ret_value
50
51     def __cleanupTags(self):
52         """ Delete old tags from database. This is tags that are not connected to any file """
53         delTagQuery = "DELETE FROM tagnames WHERE tagnames.tagid NOT IN (SELECT file_tag_relations.fk_tagid FROM file_tag_relations)"
54         self.cursor.execute(delTagQuery)
55         self.connection.commit()
56
57     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)
63                                 VALUES (?,?)", (row, False))
64             self.cursor.execute("SELECT tagid FROM tagnames WHERE tagname=?", (
65                                     row, ))
66             res = self.cursor.fetchall()
67             self.cursor.execute("INSERT INTO file_tag_relations (fk_fid, fk_tagid)
68                                 VALUES (?,?)", (fid, res[0][0]))
69
70         self.connection.commit()
71
72     def __generateFilesArray(self, li):
73         """Takes the result from SELECT * FROM files... statement.
74         Then gets for each of the files the corresponding tags, and puts it all
75         together
76         into a File.py object"""
77         ret_value = []
78         for row in li:
79             self.cursor.execute("SELECT tagnames.tagname FROM files LEFT JOIN
80                                 file_tag_relations, tagnames ON (files.fid=file_tag_relations.
81                                 fk_fid AND file_tag_relations.fk_tagid=tagnames.tagid) WHERE
82                                 files.fid=?", (row[0], ))
83             tagLi = self.cursor.fetchall()
84             tagList = self.__changeList(tagLi)
85
86             fi = File.File(fileName=row[1], path=row[2], backup=row[3], isDir
87                             =row[4], tags=tagList)
88             ret_value.append(fi)
89         return ret_value
90
91     def updateFile(self, fi):
92         #TODO update backup state!
93         """Updates The tags of a file. Updating the backup value is planned and coming
94         soon
95         @param fi, File, The file you want to update. The path and filename have to be
96         the same as on the DB
97         You can not use this to move a file, there is moveFile() for that.
98         If the file you pass does not exist on the DB yet, addFile will be called
99         instead
100         Make sure not to pass a File object with no tags, except if you want to wipe
101         out all tags of that file on DB level"""
102         new_tags = [] #Tags from file object (ATTENTION! They get filtered later!)
103         old_tags = [] #Tags from database
104         if self.fileInDB(fi):
105             old_tags = self.getTagsToFile(fi)
106             new_tags = fi.getTags()
107             for row in old_tags:
108                 try:
109                     #remove all occurrences of old tags from the new tag
110                     #array, so only new tags are left
111                     new_tags = filter(lambda a: a != row, new_tags)
112                 except:
113                     print "Error while removing element from old_tags"
114             self.cursor.execute("SELECT files.fid FROM files WHERE files.filename
115                                 =? AND files.path=?", (fi.getFileName(), fi.getPath(), ))
116             fid = self.cursor.fetchall()[0][0]
117             if len(new_tags) > 0:
118                 self.__connectTagsToFile(new_tags, fid)
119             else:
120                 pass

```

```

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
114                 #array of tags that are in the database
115                 #This leaves us with just the tags that are in the
116                 #database but that are NOT in the file object
117                 deprecatedTags = filter(lambda a: a != row,
118                                         deprecatedTags)
119             except:
120                 print ""
121                 if len(deprecatedTags) > 0:
122                     tagids = []
123                     for line in deprecatedTags:
124                         self.cursor.execute("SELECT tagid FROM tagnames WHERE
125                                             tagname=?", (line, ))
126                         tagids.extend(self.cursor.fetchall()[0])
127                     for line in tagids:
128                         self.cursor.execute("DELETE FROM file_tag_relations
129                                             WHERE fk_tagid=? AND fk_fid=?", (line, fid, ))
130                     self.connection.commit()
131                     self.__cleanupTags()
132             else:
133                 self.addFile(fi)
134
135 def fileInDB(self, fi):
136     """Checks wether a files is in the db or not
137     @param fi, File, The The file which's presence in the DB you want to check.
138     Only name and path are needed"""
139     self.cursor.execute("SELECT files.filename FROM files WHERE files.filename=?
140                         AND files.path=?", (fi.getFileName(), fi.getPath(), ))
141     value = self.cursor.fetchall()
142     if len(value) > 0:
143         return True
144     else:
145         return False
146
147 # def executeQuery(self, query):
148 #     """Deprecated! Won't be provided anymore for security reasons!"""
149 #     self.cursor.execute(query)
150 #     return self.cursor.fetchall()
151
152 def getTagsToFile(self, _file):
153     """Returns all tags that are connected to the passed file
154     @param _file, File, The file whichs tags you want (just fileName and path are
155     important)
156     @return List, List with the Tags of the file"""
157     li = None
158     self.cursor.execute("SELECT tagnames.tagname FROM files LEFT JOIN
159                         file_tag_relations, tagnames ON (files.fid=file_tag_relations.fk_fid AND
160                         file_tag_relations.fk_tagid=tagnames.tagid) WHERE path=? AND filename
161                         =?", (_file.getPath(), _file.getFileName()))
162     li = self.cursor.fetchall()
163     return self.__changeList(li)
164
165 def getFilesFromPath(self, path):
166     """Get all Files that are in a specific directory. Files will be returned
167     containing all tags 'n' stuff
168     @param path, String, the Path you want to get files from (make sure to include
169     / or \ at the end)
170     @return List, list of Files that are in the specified path"""
171     ret_value = []
172     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
226     #     at several points."""
227     #     query = "INSERT OR IGNORE INTO tagnames (tagname, backup) VALUES ('%s', 'False
228     #     ')" % (tag, )
229     #     self.cursor.execute(query)
230     #     self.connection.commit()
231
232     def moveFile(self, f1, f2):
233         """rename/move a file.
234         @param f1, File, a File object with the path and the name of the file as it is
235         BEFORE the movement
236         @param f2, File, a File object with the path and the name of the file as it is
237         AFTER the movement"""
238         if not self.fileInDB(f2):
239             self.cursor.execute("UPDATE files SET filename=?, path=? WHERE
                filename=? AND path=?", (f2.GetFileName(), f2.getPath(), f1.
                GetFileName(), f1.getPath(), ))
240             self.connection.commit()
241         else:
242             print "file_" + f2.getFullPath() + "_exists, can't move!"
243
244     def renameFile(self, f1, f2):
245         #Just calling moveFile
246         self.moveFile(f1, f2)
247
248     def getFile(self, fi):
249         """Gets all the details to a file (tags, backup state, ....
250         @param fi, File, A file object with the filename and the path of the file
251         whichs details you want to get"""
252         self.cursor.execute("SELECT * from files WHERE filename=? and path=?", (fi.
                GetFileName(), fi.getPath()))
253         f2 = self.cursor.fetchall()
254         f3 = self.__generateFilesArray(f2)[0]
255         return f3
256
257     def copyFile(self, f1, f2):
258         """Copy a file (including all tags, backup stae 'n' stuff).
259         @param f1, File, the file you want to copy
260         @param f2, File, a File object with the name and the path of the copy"""
261         if not self.fileInDB(f2):
262             f3 = self.getFile(f1)
263             f3.setFileName(f2.GetFileName())
264             f3.setPath(f2.getPath())
265             self.addFile(f3)
266
267 if __name__ == "__main__":
268     #Modul tests:
269     #=====
270     print 'Module tests start'
271     db = DB("testdb")
272     f1 = File.File(fileName="documentation.tex", path="/home/niklaus/Documents/", tags=[
        'LaTeX', 'documentation', 'filebrowser', 'project', 'school', 'test', ])
273     db.addFile(f1)
274     if len(db.getAllTags()) == 6:
275         print "Test_01: Succeed"
276     else:
277         print "Test_01: FAIL"

```

```
273
274     f2 = File.File(fileName="Music/", path="/home/niklaus/", tags=['music', 'multimedia',
275                               'entertainment', 'test', ])
276     db.addFile(f2)
277     if len(db.getAllTags()) == 9:
278         print "Test_02: Succeed"
279     else:
280         print "Test_02: FAIL"
281
282     if len(db.getTagsToFile(f2)) == 4:
283         print "Test_03: Succeed"
284     else:
285         print "Test_03: FAIL"
286
287     f3 = File.File(fileName="Music/", path="/home/niklaus/", tags=['music', 'multimedia',
288                               'entertainment', ])
289     db.updateFile(f3)
290     if len(db.getAllTags()) == 9 and len(db.getTagsToFile(f3)) == 3:
291         print "Test_04: Succeed"
292     else:
293         print "Test_04: FAIL"
294
295     f4 = File.File(fileName="documentation.tex", path="/home/niklaus/Documents/", tags=['
296           LaTeX', 'documentation', 'filebrowser', 'project', 'school', ])
297     db.addFile(f4)
298     if len(db.getAllTags()) == 8 and len(db.getTagsToFile(f4)) == 5:
299         print "Test_05: Succeed"
300     else:
301         print "Test_05: FAIL"
302
303     f5 = File.File(fileName="documentation.tex", path="/home/niklaus/Documents", tags=['
304           LaTeX', 'documentation', 'filebrowser', 'project', 'school', 'dbtest', 'modultest',
305           ])
306     db.updateFile(f5)
307     if len(db.getAllTags()) == 10 and len(db.getTagsToFile(f5)) == 7:
308         print "Test_06: Succeed"
309     else:
310         print "Test_06: FAIL"
311
312     db.removeFile(f5)
313     if len(db.GetFilesFromPath("/home/niklaus/Documents/")) == 0 and len(db.getAllTags())
314         == 3:
315         print "Test_07: Succeed"
316     else:
317         print "Test_07: FAIL"
318
319     f6 = File.File(path="/home/niklaus/Videos/", fileName="Movies/", isDir=True, tags=['
320           entertainment', 'multimedia', 'movie'])
321     db.updateFile(f6)
322     fTest = db.GetFilesFromPath("/home/niklaus/Videos/")[0]
323     if fTest.GetFileName() == "Movies/" and fTest.getPath() == "/home/niklaus/Videos/" and
324         fTest.getIsDir() == True and fTest.getTags() == ['movie', 'multimedia', '
325           entertainment', ]:
326         print "Test_08: Succeed"
327     else:
328         print "Test_08: FAIL"
329
330     f7 = File.File(path="/home/niklaus/doku/", fileName="projektantrag.tex", tags=['LaTeX',
331           'projectexplorer', 'berufsschule', ], backup=True)
332     f8 = File.File(path="/home/niklaus/", fileName="doku/", isDir=True, tags=['
333           projectexplorer', 'berufsschule', ])
334     f9 = File.File(path="/home/niklaus/doku/", fileName="projektantrag.pdf", tags=['
335           projectexplorer', 'berufsschule'])
336     f10 = File.File(path="/home/niklaus/doku/", fileName="projektplan.tex", tags=['LaTeX',
337           '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',
382         'bash.vi'])
383     db.addFile(f17)
384     if len(db.getTagsToFile(f17)) == 3:
385         print "Test_17: Succeed"
386     else:
387         print "Test_17: FAIL"
388
389     f18 = File.File(path="/home/project/", fileName="heroes", tags=['shylux', 'tschabold',
390         'bash.vi', 'heroes', 'project'])
391     db.updateFile(f18)
392     if len(db.getTagsToFile(f17)) == 5:
393         print "Test_18: Succeed"
394     else:
395         print "Test_18: FAIL"
396
397     f19 = File.File(path="/home/project/", fileName="heroes", tags=['tschabold', ])
398     db.updateFile(f19)
399     if len(db.getTagsToFile(f17)) == 1:
400         print "Test_19: Succeed"
401     else:
402         print "Test_19: FAIL"
403
404     f20 = db.getFile(f18)
405     if len(f20.getTags()) == 1 and f20.getTags()[0] == 'tschabold':
406         print "Test_20: Succeed"
407     else:
408         print "Test_20: FAIL"
409
410     f21 = File.File(path="/home/niklaus/copy/", fileName="copy.text", tags=['copy', 'test',
411         'projectexplorer'])
412     f22 = File.File(path="/home/test/copy/", fileName="copy.tested")
413     db.addFile(f21)
414     db.copyFile(f21, f22)
415     f23 = db.getFile(f22)
416     if f23.getFileName() == "copy.tested" and f23.getPath() == "/home/test/copy/" and len(
417         f23.getTags()) == 3:
418         print "Test_21: Succeed"
419     else:
420         print "Test_21: FAIL"
421
422     fx1 = File.File(path="/test/27/", fileName="foo")
423     fx2 = File.File(path="/test/28/", fileName="bar")
424     db.moveFile(fx1, fx2)
425     print "Still running"
426     fy1 = File.File("/bin/fantasy/test.txt")
427     db.removeFile(fy1)
428     print "still running"
429     #TODO addTagToFile
430     print "="*20
431     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
5                  drin
6  #Author:         Kaleb Tschabold
7  #Creation Date:  14.4.2011
8  #

```

```

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
37     def getTime(self):
38         return strftime("%Y-%m-%d_%H:%M")
39
40 #Dynamisches Objekt, dass fuer normale Objekte gebraucht werden kann
41 class NO():
42     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.prnname)+"\n"+str(self.sys.c.version)
29         option_list = [
30             make_option("-m", "--modus", dest="modus",
31                         help="Start a inline 'commandprompt'"),
32             make_option("-v", "--verbose", dest="verbose",
33                         help="Zeigt alle Ausgaben", action="
34                             store_true"),
35             make_option("-q", "--quite", dest="verbose",
36                         help="Zeigt keine Ausgaben", action="
37                             store_false", default=True),
38         ]
39         parser = OptionParser(usage, version=version, option_list=option_list)
40         a = (options, args) = parser.parse_args()
41         if options.verbose:
42             self.verbose()
43         if not options.verbose:
44             self.no_verbose()
45         if options.modus:
46             if options.modus == 'gui':
47                 self.sys.start_gui()
48             if options.modus == 'inline':
49                 self.inline()
50
51     def verbose(self):
52         sys.settrace(self.sys)
53
54     def no_verbose(self):
55         #sys.stdout = os.devnull
56         sys.stderr = os.devnull
57         #sys.stderr = os.devnull
58         pass
59
60     def inline(self):
61         print('---\nINLINE\nCOMMAND\nPROMPT\n---')
62         print('0=>Ende')
63         cli = True
64         while cli:
65             time.sleep(self.sys.c.sleep)
66             cmd = raw_input('>>>')
67             #cmd = sys.stdin.readline()
68             #if cmd == '0\n':
69             if cmd == '0':
70                 #CLI wird beendet
71                 cli = False
72                 self.sys.stoppall()
73             if cmd == 'v' or cmd == 'verbose':
74                 self.verbose()
75             if cmd == 'q' or cmd == 'verbose_quit':
76                 self.no_verbose()
77             if cmd == 'm' or cmd == 'modus':
78                 print('gui=>Graphics\nInterface')
79                 tmp = raw_input('\n\nmodus>>>')
80                 if tmp == 'gui':
81                     self.sys.start_gui()
82                     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
11 #              laden
12
13 #Unsere Klassen
14 from File import *
15
16 #andere Klassen
17 import os
18
19 class FileManager:
20     def __init__(self, sys):
21         self.sys = sys
22         pass
23
24     def getFilesFromDir(self, path):
25         a = []
26         if not os.path.exists(path):
27             matched = self.searchMatchDir(path)
28             if len(matched) >= 1:
29                 array = matched
30             else:
31                 return a
32         for i in range(len(array)):
33             a.append(File(fullPath=array[i], isDir=self.isDir(array[i])))
34             a[i].setTags(self.sys.db.getTagsToFile(a[i]))
35         else:
36             if path == '':
37                 path = path + '/'
38             if path[-1:] != '/':
39                 path = path + '/'
40             array = os.listdir(path)
41             for i in range(len(array)):
42                 fullpath = path + array[i]
43                 if self.isDir(fullpath):
44                     fullpath = fullpath + '/'
45             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 ImportError as (errno, strerror):
23     print strerror
24     pass
25 import threading
26 from File import *
27
28 class FileSystemListener(threading.Thread):
29     listener = None
30     db = None
31     def __init__(self,sys):
32         threading.Thread.__init__(self)
33         self.sys = sys
34         self.db = sys.getDb()
35         stros = self.sys.u.checkOS()
36         if stros == "linux":
37             print("it's a linux!")
38             self.listener = FileSystemListener_Linux(self)
39             return
40         if stros == "windows":
41             print("it's a windows!")
42             self.listener = FileSystemListener_Windows(self)
43             return
44         print "Can't initialize FileSystemListener. Given:", stros
45
46     def add_watch(self, path, rec):
47         self.listener.add_watch(path, rec)
48
49     def run(self):
50         self.listener.start()
51
52     def stop(self):
53         self.listener.stop()
54
55     # Event kommt als String mit dem Dateipfad an.

```

```

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

```

## 10.8 FileSystemListener Linux.py

```

1  #!/usr/bin/python
2
3  #File:           FileSystemListener_Linux.py
4  #Description:    Diese Klasse kann auf einem Linux das File System ueberwachen
5  #Author:         Lukas Knoepfel
6  #Creation Date:  14.4.2011
7  #
8  #History:        —Version—      —Date—          —Activities—
9  #               0.1             14.4.2011        Grundgeruest erstellt
10 #               0.2             18.4.2011        Events werden korrekt abgefangen. TODO
11               : add_watch implementieren und blockierendes .loop() in thread
12
13 import pyinotify
14
15 class FileSystemListener_Linux(pyinotify.ProcessEvent):
16     listener = None
17     notifier = None
18     parent = None
19     mask = pyinotify.IN_DELETE | pyinotify.IN_CREATE | pyinotify.IN_MODIFY | pyinotify.
20           IN_MOVED_FROM | pyinotify.IN_MOVED_TO # watched events
21     lastfrom = None
22     def __init__(self, tparent):
23         self.parent = tparent
24         self.listener = pyinotify.WatchManager()
25         self.notifier = pyinotify.Notifier(self.listener, self)
26         #self.listener.add_watch('/home/shylux/project-browser', self.mask, rec=True)
27
28     def add_watch(self, path, recur):
29         print recur
30         self.listener.add_watch(path, self.mask, rec=recur)
31
32     def start(self):
33         self.notifier.loop()
34
35     def stop(self):
36         self.notifier.stop()
37
38     def process_IN_CREATE(self, event):
39         self.parent.create_event(event.pathname)
40
41     def process_IN_DELETE(self, event):
42         self.parent.delete_event(event.pathname)
43
44     def process_IN_MODIFY(self, event):
45         self.parent.modify_event(event.pathname)
46
47     def process_IN_MOVED_FROM(self, event):
48         self.lastfrom = event.pathname
49
50     def process_IN_MOVED_TO(self, event):
51         if (self.lastfrom != None):
52             self.parent.move_event(self.lastfrom, event.pathname)
53         self.lastfrom = None

```

## 10.9 FileSystemListener Windows.py

```

1  #!/usr/bin/python
2
3  #File:           FileSystemListener_Windows.py
4  #Description:    Diese Klasse kann auf einem Windows das File System ueberwachen
5  #Author:         Lukas Knoepfel
6  #Creation Date:  14.4.2011
7  #
8  #History:        —Version—      —Date—           —Activities—
9  #               0.1             14.4.2011        Grundgeruest erstellt
10 #               0.2             19.4.2011        Grundfunktionalitaet erstellt
11
12 try:
13     import os
14     import win32file
15     import win32con
16 except ImportError:
17     #shit happens :P
18     pass
19
20 class FileSystemListener_Windows():
21     parent = None
22     acpath = None
23     hdir = None
24     ACTIONS = {
25         1 : "Created",
26         2 : "Deleted",
27         3 : "Modified",
28         4 : "Delete",#"Renamed from something"
29         5 : "Created"#"Renamed to something"
30     }
31     FILE_LIST_DIRECTORY = 0x0001
32
33     def __init__(self, tparent):
34         self.parent = tparent
35
36     def add_watch(self, path, recur):
37         self.acpath = path
38         self.hdir = win32file.CreateFile(
39             path,
40             self.FILE_LIST_DIRECTORY,
41             win32con.FILE_SHARE_READ | win32con.FILE_SHARE_WRITE,
42             None,
43             win32con.OPEN_EXISTING,
44             win32con.FILE_FLAG_BACKUP_SEMANTICS,
45             None
46         )
47
48     def start(self):
49         while 1:
50             results = win32file.ReadDirectoryChangesW(
51                 self.hdir,
52                 1024,
53                 True,
54                 win32con.FILE_NOTIFY_CHANGE_FILE_NAME |
55                 win32con.FILE_NOTIFY_CHANGE_DIR_NAME |
56                 win32con.FILE_NOTIFY_CHANGE_ATTRIBUTES |
57                 win32con.FILE_NOTIFY_CHANGE_SIZE |
58                 win32con.FILE_NOTIFY_CHANGE_LAST_WRITE |
59                 win32con.FILE_NOTIFY_CHANGE_SECURITY,
60                 None,
61                 None
62             )
63
64             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
12 #               verschiedenen Elemente
13 #Link:
14 #http://zignar.net/page/pygtk-mit-glade          mit xml(aus glade) Programm Grafische
15 #         Oberflaeche erstellen
16
17 from HirarchicalView import *
18 from TagView import *
19 from AddTag import *
20
21 import gobject
22 gobject.threads_init()
23
24 #Modul um diese Klasse als Seperaten-Prozess zu starten
25 import threading
26 import pygtk
27 pygtk.require('2.0')
28 import gtk
29 import gtk.glade
30 import sys
31
32 #class GUI(threading.Thread):
33 class GUI():
34     def __init__(self, sys):
35         #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.
50                               version))
51         #self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
52         self.window.connect("destroy", self.stoploop)
53         self.window.connect("key-release-event", self.eventF5)
54
55         #Connect Toogle-Buttons
56         self.btnHirarchical = self.xml.get_widget('btnHirarchical')
57         self.btnHirarchical.connect('toggled', self.showHirarchical)
58         self.btnTag = self.xml.get_widget('btnTag')
59         self.btnTag.connect('toggled', self.showTag)
60
61         #Connect TextInput Field
62         self.txtEntry = self.xml.get_widget('txtEntry')
63         #self.txtEntry.connect('key_release_event', self.searchKey)
64         self.txtEntry.connect('changed', self.searchKey)
65         self.com = gtk.EntryCompletion()
66         #com.set_inline_selection(True)
67         #com.connect('match-selected', self.searchCompletion)
68         self.txtEntry.set_completion(self.com)
69         self.listcompl = gtk.ListStore(gobject.TYPE_STRING)
70         self.com.set_model(self.listcompl)
71         self.com.set_text_column(0)
72         self.com.set_popup_set_width(True)
73
74         #self.txtEntry.connect('activate', self.searchKey)
75         #self.txtEntry.connect('backspace', self.searchKey)
76
77         #Add Menu-Item by 'Ansicht'
78         ansichtmenu = gtk.Menu()
79         ansicht = self.xml.get_widget('mnuAnsicht')
80         ansicht.set_submenu(ansichtmenu)
81         self.mnuHirarchical = gtk.RadioMenuItem(None, 'Hierarchisch')
82         self.mnuHirarchical.connect('toggled', self.showHirarchical)
83         ansichtmenu.append(self.mnuHirarchical)
84         self.mnuTag = gtk.RadioMenuItem(self.mnuHirarchical, 'Tag')
85         self.mnuTag.connect('toggled', self.showTag)
86         ansichtmenu.append(self.mnuTag)
87
88         #Modify other MenuItems
89         self.mnuBeenden = self.xml.get_widget('mnuBeenden')
90         self.mnuBeenden.connect('activate', self.stoploop)
91         self.mnuDatei = self.xml.get_widget('mnuDatei')
92         self.mnuBearbeiten = self.xml.get_widget('mnuBearbeiten')
93         self.mnuHilfe = self.xml.get_widget('mnuHilfe')
94         self.mnuDatei.set_sensitive(False)
95         self.mnuBearbeiten.set_sensitive(False)
96         self.mnuHilfe.set_sensitive(False)
97
98         #Init Status Bar
99         self.Status = self.xml.get_widget('stsStatusBar')
100        self.Status.push(1, 'init')
```

```

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

```

```

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

```

```

228
229     def searchCompletion(self, completion, prefix, user_param1):
230         self.updateView()
231
232     def updateView(self, actor='fn'):
233         self.actview.set_actTxtInput(self.txtEntry.getText())
234         self.actview.update(actor)
235
236     def openDirInHirarchical(self, path):
237         self.showHirarchical('init')
238         self.actview.set_actTxtInput(path)
239         self.listcompl.clear()
240         self.txtEntry.setText(self.actview.get_actTxtInput())
241
242     def historyBack(self, event):
243         self.actview.triggeredByNavigation = True
244         self.actview.historyCursor = self.actview.historyCursor-1
245         self.actview.set_actTxtInput(self.actview.history[self.actview.historyCursor])
246         self.txtEntry.setText(self.actview.get_actTxtInput())
247         self.actview.triggeredByNavigation = False
248
249     def historyFor(self, event):
250         self.actview.triggeredByNavigation = True
251         self.actview.historyCursor = self.actview.historyCursor+1
252         self.actview.set_actTxtInput(self.actview.history[self.actview.historyCursor])
253         self.txtEntry.setText(self.actview.get_actTxtInput())
254         self.actview.triggeredByNavigation = False
255
256     def getParentFolder(self, event):
257         parent=self.sys.filemanager.getParentDir(self.actview.get_actTxtInput())
258         self.txtEntry.setText(parent)
259
260     def set_focus(self, widget):
261         self.window.set_focus(widget)
262
263     def eventF5(self, widget, event):
264         if event.keyval == gtk.gdk.keyval_from_name("F5"):
265             self.actview.update()
266
267     def showall(self):
268         self.window.show_all()
269
270     def terminate(self):
271         # must raise the SystemExit type, instead of a SystemExit() instance
272         # due to a bug in PyThreadState_SetAsyncExc
273         self.raise_exc(SystemExit)
274         pass
275
276     def startloop(self):
277         gtk.main()
278
279     def stopploop(self, event='') :
280         print('quite')
281         gtk.main_quit()
282         self.sys.stoppall()

```

## 10.12 TagView.py

```

1  #!/usr/bin/python
2
3  #File:           TagView.py
4  #Description:    Klasse um die Dateien zu einem Tag anzuzeigen
5  #Author:         Kaleb Tschabold
6  #Creation Date:  29.3.2011
7  #

```



```

8 #History:      —Version—      —Date—      —Activities—
9 #            0.1            29.3.2011      Grundfunktionalitaeten werden erstellt
10
11 from View import *
12 import gtk
13
14 class TagView(View):
15     mod = 'tag'
16     def __init__(self, sys):
17         View.__init__(self, sys)
18         self.set_actTxtInput(sys.c.initStrTag)
19         self.connect('row-activated', self.rowActivate)
20
21     def update(self, actor = 'fn'):
22         if self.sys.gui != None:
23             oldmodel = self.get_model()
24             self.model.clear()
25             #Dieser Durchgang ist, wenn noch nicht's im TextFeld steht
26             if self.get_actTxtInput() == '':
27                 t = self.sys.db.getAllTags()
28                 for i in range(len(t)):
29                     self.model.append(None, [self.getFolderIcon(), '*', [t[i]
30                                             ]], t[i]])
31                 self.set_model(self.model)
32                 self.historyUpdate(actor)
33                 self.historySymboleManagement()
34                 self.updateParentFolderBtn()
35                 self.completion()
36                 #Damit diese Funktion abgebrochen wird
37                 return 0
38             #Dieser Durchgang ist, wenn Zeichen im Text Feld stehen
39             itemarray = self.get_actTxtInput().split(',')
40             for j in range(len(itemarray)):
41                 oneitem = itemarray[j].strip()
42                 self.items = self.sys.db.GetFilesFromTag(oneitem)
43                 for i in range(len(self.items)):
44                     self.items[i].setIsDir(self.items[i].getIsDir())
45                     if self.items[i].getIsDir():
46                         self.model.append(None, [self.getFolderIcon(),
47                                                    self.items[i].getFileName(), self.items[i],
48                                                    ', '.join(self.items[i].getTags())])
49                     else:
50                         self.model.append(None, [self.getFileIcon(),
51                                                    self.items[i].getFileName(), self.items[i],
52                                                    ', '.join(self.items[i].getTags())])
53
54             self.set_model(self.model)
55             if len(self.items) > 0:
56                 self.historyUpdate(actor)
57                 self.historySymboleManagement()
58                 self.updateParentFolderBtn()
59                 self.completion()
60
61     def completion(self):
62         matched = self.sys.tagmanager.searchMatchTags(self.get_actTxtInput())
63         #try:
64         if self.sys.gui.listcompl != None:
65             #try:
66                 self.sys.gui.listcompl.clear()
67             #except:
68                 # pass
69         for i in range(len(matched)):
70             self.sys.gui.listcompl.append([matched[i]])
71         #except:
72         # pass

```

```

69     def updateParentFolderBtn(self):
70         self.sys.gui.btnUp.set_sensitive(False)
71
72     def rowActivate(self,treeview, path, user_data):
73         f = self.getFObjFromSelectedRow()
74         if isinstance(f,list):
75             print('f:'+str(f[0]))
76             self.sys.gui.txtEntry.set_text(f[0])
77             self.sys.gui.updateView()
78         else:
79             if not f.getIsDir():
80                 self.sys.filemanager.openFile(f.getFullPath())
81             else:
82                 self.sys.gui.openDirInHirarchical(f.getFullPath())

```

### 10.13 HirarchicalView.py

```

1  #!/usr/bin/python
2
3  #File:           HirarchicalView.py
4  #Description:    Klasse um die Orderstruktur 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
11
12     aufrufen
13
14 from View import *
15 import pygtk
16 pygtk.require('2.0')
17 import gtk
18
19 class HirarchicalView(View):
20     mod = 'hirarchical'
21     def __init__(self,sys):
22         View.__init__(self,sys)
23
24         self.history.append('/')
25         self.historyCursor = 0
26         self.sys = sys
27         self.set_actTxtInput(sys.c.initStrHirarchical)
28         self.connect('row-activated',self.rowActivate)
29
30     def update(self, actor = 'fn'):
31         if self.sys.gui != None:
32             if self.get_actTxtInput() == '':
33                 self.sys.gui.txtEntry.set_text('/')
34                 return 0
35             self.items = self.sys.filemanager.getFilesFromDir(self.acttxtinput)
36             if self.items != 'error':
37                 #Ordner konnte geoffnet werden
38                 self.model.clear()
39                 for i in range(len(self.items)):
40                     if self.items[i].getIsDir():
41                         self.model.append(None,[self.getFolderIcon(),
42                                                 self.items[i].getFileName(),self.items[i],
43                                                 ', '.join(self.items[i].getTags())])
44                     else:
45                         self.model.append(None,[self.getFileIcon(),
46                                                 self.items[i].getFileName(),self.items[i],
47                                                 ', '.join(self.items[i].getTags())])
48                 self.set_model(self.model)
49                 #Ruft History Verwaltung auf

```

```

44         if len(self.get_actTxtInput()) > 0:
45             if self.get_actTxtInput()[-1] == '/':
46                 self.historyUpdate(aktor)
47             self.updateParentFolderBtn()
48             self.historySymboleManagement()
49             #Ruft Auto-Completion Update Funktion auf
50             self.completion()
51
52     def completion(self):
53         matched = self.sys.filemanager.searchMatchDir(self.get_actTxtInput())
54         try:
55             if self.sys.gui.listcompl != None:
56                 try:
57                     self.sys.gui.listcompl.clear()
58                 except:
59                     pass
60             for i in range(len(matched)):
61                 self.sys.gui.listcompl.append([matched[i]])
62         except:
63             pass
64
65     def updateParentFolderBtn(self):
66         parent = self.sys.filemanager.getParentDir(self.get_actTxtInput())
67         if parent:
68             self.sys.gui.btnUp.set_sensitive(True)
69         else:
70             self.sys.gui.btnUp.set_sensitive(False)
71
72     def rowActivate(self,treeview, path, user_data):
73         f = self.getFObjFromSelectedRow()
74         print('fullpath_activated_in_rowActivated:_' + f.getFullPath())
75         if not f.getIsDir():
76             self.sys.filemanager.openFile(f.getFullPath())
77         else:
78             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
80     def getFolderIcon(self):
81         return self.get_icon_pixbuf('STOCK_DIRECTORY')
82
83     def getFileIcon(self):
84         return self.get_icon_pixbuf('STOCK_FILE')
85
86     def update(self):
87         self.show_all()
88
89     def set_actTxtInput(self, text):
90         self.acttxtinput = text
91
92     def get_actTxtInput(self):
93         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             f = self.getFObjFromSelectedRow()
122         except:
123             pass
124         if f != None:
125             self.sys.gui.addTagContent.update(f)
126         #except:
127         #    pass
128
129     def historyUpdate(self, actor='fn'):
130         #Wenn wieder vorwaertz gesprungen wird, werden alle Element nach der aktuellen
131         #Position geloescht
132         if self.historyCursor > -1:
133             if actor == 'user' and self.history[self.historyCursor] != self.
134                 get_actTxtInput():
135                 if self.historyCursor < len(self.history)-1:
136                     rem = (len(self.history)-1) - self.historyCursor
137                     l = len(self.history)
138                     i = 1
139                     #rem+1 weil die len von history eins mehr ist als der
140                     #cursor
141                     while i < rem+1:
142                         self.history.remove(self.history[l-i])
143                         i = i + 1
144                     self.historyCursor = self.historyCursor
145                     self.historyCursor = self.historyCursor + 1
146                     self.history.append(self.get_actTxtInput())
147                     self.historySymboleManagement()
148             elif len(self.history) == 0:
149                 self.historyCursor = self.historyCursor + 1
150                 self.history.append(self.get_actTxtInput())
151                 self.historySymboleManagement()
152
153     def historySymboleManagement(self):
154         h = len(self.history)-1
155         c = self.historyCursor
156         if h >= c and h != 0 and c > 0:
157             self.sys.gui.btnBack.set_sensitive(True)
158         if h >= c and h != 0:

```

```

156         self.sys.gui.btnFor.set_sensitive(True)
157         if c <= 0 or h == 0:
158             self.sys.gui.btnBack.set_sensitive(False)
159         if c == h or h == 0:
160             self.sys.gui.btnFor.set_sensitive(False)
161
162
163 #Registriert diese Klasse als pygtk-widget
164 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 wird in einer solchen Klasse
5                        referenziert
6  #Author:              Kaleb Tschabold
7  #Creation Date:       14.4.2011
8  #
9  #History:             —Version—      —Date—          —Activities—
10 #                    0.1             14.04.2011      Grundfunktionalitaeten werden erstellt
11 #                    0.9             21.04.2011      Created methods and functionality.
12 #                    Added __doc__s. added test (bottom)
13
14 import re
15 import os
16 from Constant import *
17 from Utility import *
18 from shutil import copytree, ignore_patterns, copyfile, rmtree
19
20 class File:
21     """represents a file on the filesystem"""
22     fileName = None
23     path = None
24     isDir = None
25     tags = None
26     backup = None
27     fullPath = None
28     u = None
29     constant = None
30     os = None
31
32     def __init__(self, fullPath=None, fileName=None, path=None, tags=[], backup=False,
33                 isDir=False):
34         """Constructor
35         @param fileName, string, optional
36         @param path, string, optional
37         @param tags, list, optional
38         @param backup, boolean, optional
39         """
40         self.fileName = fileName
41         self.path = path
42         self.tags = tags
43         self.backup = backup
44         self.isDir = isDir
45         self.fullPath = fullPath
46
47         self.u = Utility()
48         self.constant = Constant(self)
49         self.os = self.constant.os
50
51         self.__splitFullPath()
52         self.__checkPaths()
53
54     def __splitFullPath(self):

```

```

51         if not self.fullPath == None:
52             if self.fullPath.endswith("/") or self.fullPath.endswith("\\"):
53                 p = re.compile("((?:.*\\|)(?:.*\\/))((?:[~/]*)(?:[^\|\\]*))"
54                     "([/|\\])")
55                 match = p.match(self.fullPath)
56                 if not match.group(1) == None and not match.group(2) == None:
57                     self.path = match.group(1)
58                     self.fileName = match.group(2)+match.group(3)
59             else:
60                 p = re.compile("((?:.*\\|)(?:.*\\/))((?:[~/]*)(?:[^\|\\]*))")
61                 match = p.match(self.fullPath)
62                 if not match.group(1) == None and not match.group(2) == None:
63                     self.path = match.group(1)
64                     self.fileName = match.group(2)
65
66     def __checkPaths(self):
67         # Check wether all paths have / (or \ on Windows) at the end
68         # This is highly experimental! Tell me if you have any trouble with it!
69         if self.os == 'linux' or self.os == 'mac':
70             if not self.path == None:
71                 if not self.path.endswith("/") and not self.path.endswith("\\"):
72                     self.path = self.path+"/"
73             if not self.fileName == None:
74                 if self.isDir == True and not self.fileName.endswith("/") and
75                     not self.fileName.endswith("\\"):
76                     self.fileName = self.fileName+"/"
77         elif self.os == 'win':
78             if not self.path == None:
79                 if not self.path.endswith("\\") and not self.path.endswith("/")
80                     ):
81                     self.path = self.path+"\"
82             if not self.fileName == None:
83                 if self.isDir == True and not self.fileName.endswith("\\") and
84                     not self.fileName.endswith("/"):
85                     self.fileName = self.fileName+"\"
86
87     def setFileName(self, fileName):
88         """@param filename, string"""
89         self.fileName = fileName
90         self.__checkPaths()
91
92     def setPath(self, path):
93         """@param path, string"""
94         self.path = path
95         self.__checkPaths()
96
97     def setIsDir(self, b):
98         """@param b, boolean"""
99         self.isDir = b
100         self.__checkPaths()
101
102     def setTags(self, tags):
103         """@param tags, list"""
104         self.tags = tags
105
106     def setBackup(self, backup):
107         """@param backup, boolean"""
108         self.backup = backup
109
110     def setFullPath(self, fullPath):
111         """@param fullPath, Path including filename"""
112         self.fullPath = fullPath
113         self.__splitFullPath()
114         self.__checkPaths()

```

```
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
135             the file's name"""
136         if self.fullPath == None:
137             if not self.path==None or not self.fileName==None:
138                 return self.path + self.fileName
139             else:
140                 print "either path or fileName (or both) are not specified.
141                     You can't use this function unless both are specified"
142         else:
143             return self.fullPath
144
145     def addTag(self, tag):
146         """Adds a single tag (passed as string) to the list of tags.
147         uses .append"""
148         self.tags.append(tag)
149
150     def addTags(self, tags):
151         """Adds a list of tags (passed as list) to the list of tags.
152         uses .extend"""
153         self.tags.extend(tags)
154
155     def ex_backup(self):
156         print "FullPath", self.getFullPath()
157         dir = self.getPath() + ".pb_backup"
158         if not os.path.exists(dir):
159             os.makedirs(dir)
160         bdir = dir + "/" + self.u.getTime()
161         if not os.path.exists(bdir):
162             os.makedirs(bdir)
163         if (os.path.isdir(self.getFullPath())):
164             copytree(self.getFullPath(), bdir + "/" + self.getFileName(), ignore=
165                 ignore_patterns('.*', '*.pyc'))
166         else:
167             copyfile(self.getFullPath(), bdir + "/" + self.getFileName())
168
169     def get_backups(self):
170         dir = self.getPath() + ".pb_backup"
171         founded = []
172         if os.path.exists(dir):
173             listOfBackups = os.listdir(dir)
174             for i in range(len(listOfBackups)):
175                 bdir = dir + '/' + listOfBackups[i]
176                 listOfDirs = os.listdir(bdir)
```



```

175         for j in range(len(listOfDirs)):
176             if listOfDirs[j] == self.getFileName()[:-1]:
177                 founded.append(File(bdir))
178             return founded
179         else:
180             return []
181
182
183     def restoreFrom(self, b):
184         rmtree(self.getFullPath())
185         if (os.path.isdir(b.getFullPath()+"/"+self.getFileName())):
186             copytree(b.getFullPath()+"/"+self.getFileName(), self.getPath()+self.
                getFileName(), ignore=ignore_patterns('.*', '*.pyc'))
187         else:
188             copyfile(b.getFullPath()+"/"+self.getFileName(), self.getPath()+self.
                getFileName())
189
190 if __name__ == "__main__":
191     print "Starting tests"
192     print "Note: Tests 9,10,14,15 will fail on Windows... and so will many others probably"
193
194     print "====="
195     x = File()
196     if x.getFileName() == None:
197         print "Test #01: Succeed"
198     else:
199         print "Test #01: FAIL"
200
201     x.setFileName("test")
202     if x.getFileName() == "test":
203         print "Test #02: Succeed"
204     else:
205         print "Test #02: FAIL"
206
207     x.setTags(["test", "tag"])
208     if x.getTags() == ["test", "tag"]:
209         print "Test #03: Succeed"
210     else:
211         print "Test #03: FAIL"
212
213     x.addTag("new_tag")
214     if x.getTags()[2] == "new_tag":
215         print "Test #04: Succeed"
216     else:
217         print "Test #04: FAIL"
218
219     x.addTags(["extend", "linux", "unix"])
220     if x.getTags()[3:] == ["extend", "linux", "unix"]:
221         print "Test #05: Succeed"
222     else:
223         print "Test #05: FAIL"
224
225     x.setIsDir(True)
226     if x.getIsDir():
227         print "Test #06: Succeed"
228     else:
229         print "Test #06: FAIL"
230
231     #Testing the path stuff...
232     f1 = File(fullPath="/home/niklaus/lol.txt")
233     if f1.getPath() == "/home/niklaus/" \
234         and f1.getFileName() == "lol.txt" and f1.getFullPath() == "/home/niklaus/lol.txt":
235         print "Test #07: Succeed"
236     else:
237         print "Test #07: FAIL"

```

```
238 f2 = File(fullPath="C:\\windows\\system32\\chlous.txt.test")
239 if f2.getPath() == "C:\\windows\\system32\\" \
240     and f2.getFileName() == "chlous.txt.test" and f2.getFullPath() == "C:\\windows\\
    system32\\chlous.txt.test":
241     print "Test_#09: Succeed"
242 else:
243     print "Test_#09: FAIL"
244
245 f3 = File(fullPath="C:\\windows\\system32\\")
246 if f3.getPath() == "C:\\windows\\" \
247     and f3.getFileName() == "system32\\" and f3.getFullPath() == "C:\\windows\\
    system32\\":
248     print "Test_#10: Succeed"
249 else:
250     print "Test_#10: FAIL"
251
252 f4 = File("/home/niklaus/Music/")
253 if f4.getPath() == "/home/niklaus/" \
254     and f4.getFileName() == "Music/" and f4.getFullPath() == "/home/niklaus/Music/":
255     print "Test_#11: Succeed"
256 else:
257     print "Test_#11: FAIL"
258
259 f5 = File(fileName="document.odt", path="/home/niklaus/Documents/")
260 if f5.getFullPath() == "/home/niklaus/Documents/document.odt":
261     print "Test_#12: Succeed"
262 else:
263     print "Test_#12: FAIL"
264
265 f6 = File(fileName="Videos/", path="/home/niklaus/", isDir=True)
266 if f6.getFullPath() == "/home/niklaus/Videos/" and f6.getIsDir() == True:
267     print "Test_#13: Succeed"
268 else:
269     print "Test_#13: FAIL"
270
271 f7 = File(fileName="hosts", path="C:\\windows\\system32\\etc\\")
272 if f7.getFullPath() == "C:\\windows\\system32\\etc\\hosts":
273     print "Test_#14: Succeed"
274 else:
275     print "Test_#14: FAIL"
276
277 f8 = File(fileName="etc\\", path="C:\\windows\\system32\\", isDir=True)
278 if f8.getFullPath() == "C:\\windows\\system32\\etc\\" and f8.getIsDir() == True:
279     print "Test_#15: Succeed"
280 else:
281     print "Test_#15: FAIL"
282
283 #Testing the automatic path correction
284 f9 = File(fileName="ozzed.ogg", path="/home/niklaus/Music")
285 if f9.getPath() == "/home/niklaus/Music/" and f9.getFileName() == "ozzed.ogg":
286     print "Test_#16: Succeed"
287 else:
288     print "Test_#16: Fail"
289
290 f10 = File(fileName="Music", path="/home/niklaus", isDir=True)
291 if f10.getPath() == "/home/niklaus/" and f10.getFileName() == "Music/":
292     print "Test_#17: Succeed"
293 else:
294     print "Test_#17: FAIL"
295
296 f11 = File(fileName="Music", isDir=True)
297 if f11.getFileName() == "Music/" and f11.getPath() == None:
298     print "Test_#18: Succeed"
299 else:
300     print "Test_#18: FAIL"
301
```

```

302     f12 = File(path="/home/niklaus/somepath")
303     if f12.getPath() == "/home/niklaus/somepath/" and f12.getFileName() == None:
304         print "Test_#19: Succeed"
305     else:
306         print "Test_#19: FAIL"
307
308     f13 = File()
309     f13.setFullPath("/home/niklaus/File.php")
310     if f13.getPath() == "/home/niklaus/" and f13.getFileName() == "File.php":
311         print "Test_#20: Succeed"
312     else:
313         print "Test_#20: FAIL"
314
315     f14 = File(isDir=True)
316     f14.setFullPath("/home/niklaus/Music")
317     if f14.getPath() == "/home/niklaus/" and f14.getFileName() == "Music/":
318         print "Test_#21: Succeed"
319     else:
320         print "Test_#21: FAIL"
321
322     print File.__init__.__doc__
323
324     #Backup
325     #b = File("/home/laden")
326     #b.ex_backup()

```

## 10.16 AddTag.py

```

1  import gtk
2  import gobject
3  from File import *
4  class AddTag():
5      def __init__(self, sys):
6          self.sys = sys
7          self.fobj = None
8          self.xml = gtk.glade.XML("addTag.glade")
9          self.main = self.xml.get_widget("vbxMain")
10         self.txtObjNames = self.xml.get_widget("txtNames")
11         self.lblName = self.xml.get_widget("lblName")
12         self.txtTags = self.xml.get_widget("txtTags")
13         self.txtTags.connect('key-release-event', self.enterEventHandler)
14         #self.txtTags.connect('drag_data_received', self.test)
15         self.btnSave = self.xml.get_widget("btnSave")
16         self.btnSave.connect('button-release-event', self.save)
17         self.hbxTag = self.xml.get_widget("hbxTag")
18         self.tagCont = self.xml.get_widget("conTagList")
19         self.restoreCont = self.xml.get_widget("conRestoreCont")
20
21         #Backup/Restore
22         self.btnBackup = self.xml.get_widget("btnBackup")
23         self.btnBackup.connect('clicked', self.backup)
24         self.btnBackup.set_sensitive(False)
25         self.btnRestore = self.xml.get_widget("btnRestore")
26         self.btnRestore.connect('clicked', self.restore)
27         self.btnRestore.set_sensitive(False)
28
29
30
31         #Model Tag
32         self.tagModel = gtk.TreeStore(gobject.TYPE_STRING)
33         self.tagTree = gtk.TreeView(self.tagModel)
34         self.tagTree.connect('row-activated', self.addClickedTag)
35         self.tagCont.add(self.tagTree)
36         #Spalte 1
37         self.tagCl1 = gtk.TreeViewColumn('Tag_Name')

```

```

38         self.tagTree.append_column(self.tagCl1)
39         #Definition der Text der 1. Spalte
40         tagRender = gtk.CellRendererText()
41         self.tagCl1.pack_start(tagRender)
42         self.tagCl1.add_attribute(tagRender, 'text', 0)
43
44
45
46         #Model Tag
47         self.restoreModel = gtk.TreeStore(gobject.TYPE_STRING, gobject.TYPE_PYOBJECT)
48         self.restoreTree = gtk.TreeView(self.restoreModel)
49         self.restoreTree.connect('cursor-changed', self.updateRestoreButton)
50         self.restoreCont.add(self.restoreTree)
51         #Spalte 1
52         self.restoreCl1 = gtk.TreeViewColumn('Datum')
53         self.restoreTree.append_column(self.restoreCl1)
54         #Definition der Text der 1. Spalte
55         restoreRender = gtk.CellRendererText()
56         self.restoreCl1.pack_start(restoreRender)
57         self.restoreCl1.add_attribute(restoreRender, 'text', 0)
58         #self.restoreTree.connect('row-activated', self.showRestoreBtn)
59
60
61         self.update(None)
62         self.main.show_all()
63
64     def test(self):
65         print('test')
66
67     def update(self, fobj):
68         self.clearAll()
69         self.fobj = None
70         self.fobj = fobj
71         self.lblName.set_label('')
72         self.hbxTag.set_sensitive(False)
73         if isinstance(fobj, File):
74             self.lblName.set_label('Datei(en)')
75             self.txtTags.set_text(', '.join(self.fobj.getTags()))
76             self.txtObjNames.set_text(self.fobj.getFileName())
77             self.updateTagModel()
78             self.updateRestoreModel('file')
79             self.hbxTag.set_sensitive(True)
80         elif type(fobj) == list:
81             self.lblName.set_label('Tag(s)')
82             self.txtObjNames.set_text(fobj[0])
83             self.updateRestoreModel('tag')
84             self.hbxTag.set_sensitive(False)
85         self.updateButtons()
86
87     def updateTagModel(self):
88         self.tagModel.clear()
89         for tag in self.sys.db.getAllTags():
90             self.tagModel.append(None, [tag])
91
92     def updateRestoreModel(self, typ):
93         self.restoreModel.clear()
94         backups = self.fobj.get_backups()
95         for b in backups:
96             self.restoreModel.append(None, [b.getFileName(), b])
97
98     def clearAll(self):
99         self.fobj = None
100         self.tagModel.clear()
101         self.txtObjNames.set_text('')
102         self.txtTags.set_text('')
103         self.updateTagModel()

```

```
104
105     def getWidget(self):
106         return self.main
107
108     def addClickedTag(self,treeview, path, user_data):
109         selection = treeview.get_selection()
110         selection.set_mode(gtk.SELECTION_SINGLE)
111         tree_model, tree_iter = selection.get_selected()
112         tagname = tree_model.get_value(tree_iter,0)
113         if self.txtTags.get_text() == '':
114             self.txtTags.set_text(tagname)
115         else:
116             self.txtTags.set_text(self.txtTags.get_text()+',␣'+tagname)
117
118     def enterEventHandler(self,widget,event):
119         if event.keyval == gtk.gdk.keyval_from_name("Return"):
120             self.save(widget,event)
121
122     def save(self,widget,event):
123         tags = self.txtTags.get_text().split(',')
124         l = len(tags)
125         newtags = []
126         for i in range(len(tags)):
127             if tags[i].strip() != '':
128                 newtags.append(tags[i].strip())
129         tags = list(set(newtags))
130         self.fobj.setTags(newtags)
131         self.sys.db.updateFile(self.fobj)
132         self.update(self.fobj)
133         self.sys.gui.actview.update()
134
135     def updateButtons(self):
136         if isinstance(self.fobj,File):
137             # or type(self.fobj) == list
138             self.btnBackup.set_sensitive(True)
139         else:
140             self.btnBackup.set_sensitive(False)
141             self.btnRestore.set_sensitive(False)
142
143     def updateRestoreButton(self,event):
144         self.btnRestore.set_sensitive(True)
145
146     def backup(self,event):
147         if isinstance(self.fobj,File):
148             self.fobj.ex_backup()
149             self.update(self.fobj)
150             self.sys.gui.actview.update()
151         elif type(self.fobj) == list:
152             pass
153
154     def getSelectedBackup(self):
155         treeview = self.restoreTree
156         selection = treeview.get_selection()
157         selection.set_mode(gtk.SELECTION_SINGLE)
158         tree_model, tree_iter = selection.get_selected()
159         return tree_model.get_value(tree_iter,1)
160
161     def restore(self,event):
162         if isinstance(self.fobj,File):
163             self.fobj.restoreFrom(self.getSelectedBackup())
164         elif type(self.fobj) == list:
165             pass
```