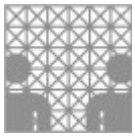


- Dokumentation -

magicBook

Programmiertechnisches Praktikum (PTP)

Veranstalter: Andreas Heymann
Fachbereich Informatik
Rechenzentrum
Sommersemester 2013
Abgabedatum: 15.08.2013

Autoren

Niels Petersen

1npeters@informatik.uni-hamburg.de
Studiengang: Wirtschaftsinformatik
Matr.--Nr. 6315732
Fachsemester 4

Jonas Schmid

1jschmid@informatik.uni-hamburg.de
Studiengang: Mensch-Computer-Interaktion
Matr.-Nr. 6338368
Fachsemester 4

Inhaltsverzeichnis

Beschreibung

Heutzutage gibt es viele Personen, die ihre Lieblings-Medien (Filme, Musik, Spiele, Websites, Programme oder Fotos) nur noch in digitaler Form vorliegen haben. Gerade für Sammler haben Musik – oder Film-Cover einen ideellen Wert, und daher besteht oftmals der Wunsch ihre „Schätze“ in der realen Welt zu konservieren, zu sortieren oder auszustellen.

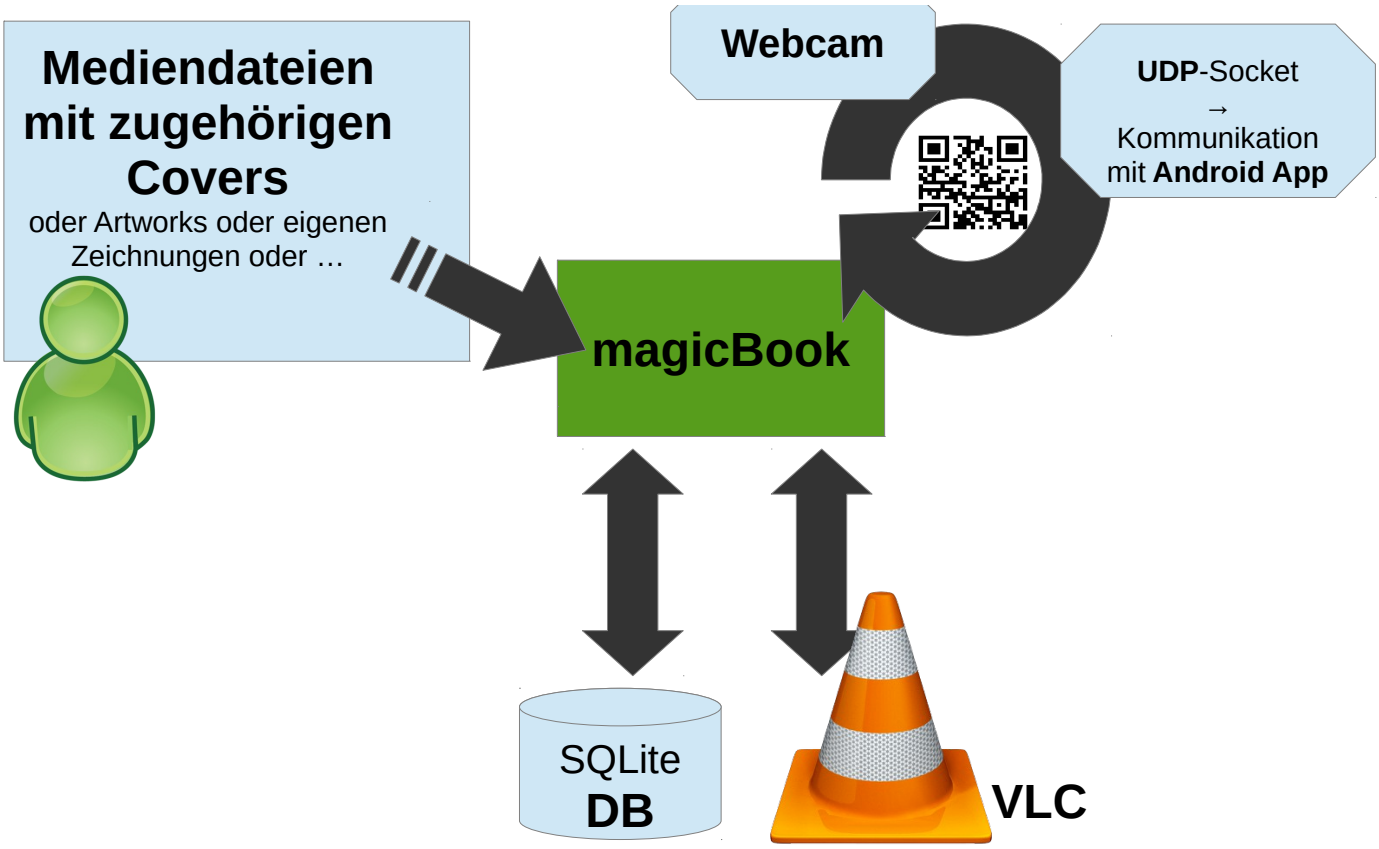
magicBook hilft dabei eine Brücke zwischen dem Alltag am Computer und der realen Welt zu schlagen, indem sog. **magicCards** gedruckt bzw. gebastelt werden können, welche anschließend durch intuitives (simples) Einscannen vor der Webcam bzw. einem Smartphone (Android) zu einer Aktion führt. **magicCard**s sind ausgedruckte Karten, die mit einem Cover/Bild und einem QR-Code versehen sind.

So können auch für Großeltern Programme einfach mit einem simplen Scan geöffnet, oder digitale Fotoalben oder Musik- bzw. Filmsammlungen abgespielt werden. Die erstellten **magicCards** (mit Albencover) können zudem als Dekorationsmaterial verwendet werden, ähnlich wie Schallplatten um seine Wohnung zu verschönern, oder die aktuell abspielenden Medien zu präsentieren.

magicBook stellt dem Anwender ein Werkzeug zur Verfügung, welches das Anlegen eines **physischen Medienverzeichnisses** unterstützt und parallel eine **intuitive Interaktion** mit dem Computer ermöglicht.

Die **Portabilität (Mobilität)** der Anwendung steht im Vordergrund, welche zusammen mit **Datenschutzbedenken** gegenüber z.B. Tracking und Profilerstellung bei vielen Onlinediensten bzw. nicht dezentralen Systemen und einer erforderlichen **schnellen Verfügbarkeit (d.h. persistent geringe Datenzugriffszeiten)**, dazu führt, dass **magicBook** alle Medien lokal,dezentral speichert.

Abbildung 1: magicBook Übersicht



Aufteilung, Modularisierung

magicBook ist grundsätzlich sehr modular gestaltet, um zukünftige Erweiterungen und Modifikationen zu erleichtern. Das MVC-Konzept wurde angewandt, um eine strikte Trennung zwischen GUI und der eigentlichen Programmlogik zu erreichen. Die Tools (und ihre entsprechenden UI-Klassen) übernehmen die Interaktion mit dem Benutzer. Die Tools selber sind nicht für fachliche Aufgaben zuständig, sondern delegieren diese an den fachlichen Klassen. Als fachliche Klassen dienen hier z.B. ***CopyBotTool*** oder ***BufferedImageTool***. Die persistente Speicherung von Daten erfolgt in einer SQLite-Datenbank bzw. in Form von XML-Dateien.

Der Entwurf der Programmlogik und der Schnittstellen zur Webcam und mittels UDP zum Smartphone wurde von Jonas übernommen. Des Weiteren wurde die Kommunikation mit VLC via Telnet-Interface von ihm durchgeführt.

Niels übernahm die Gestaltung der graphischen Benutzeroberfläche und die persistente Speicherung der Medien in der Datenbank sowie die Anpassung des Speichervorgangs(z.B. Auslagerung in Thread).

Der Kern der Anwendung ist die Klasse ***MagicBook***. Innerhalb dieser Klasse, wird die Anwendung gestartet, die graphische Benutzeroberfläche erzeugt und die Kommunikation mit Peripherie-Geräten durchgeführt.

Für das Hinzufügen von Medien sind im Wesentlichen das Tool ***AddMediumTool*** und später abhängig vom gewählten Medientyp, eine abgeleitete Klasse von ***MediumSpecificQuestionerTool*** zuständig.

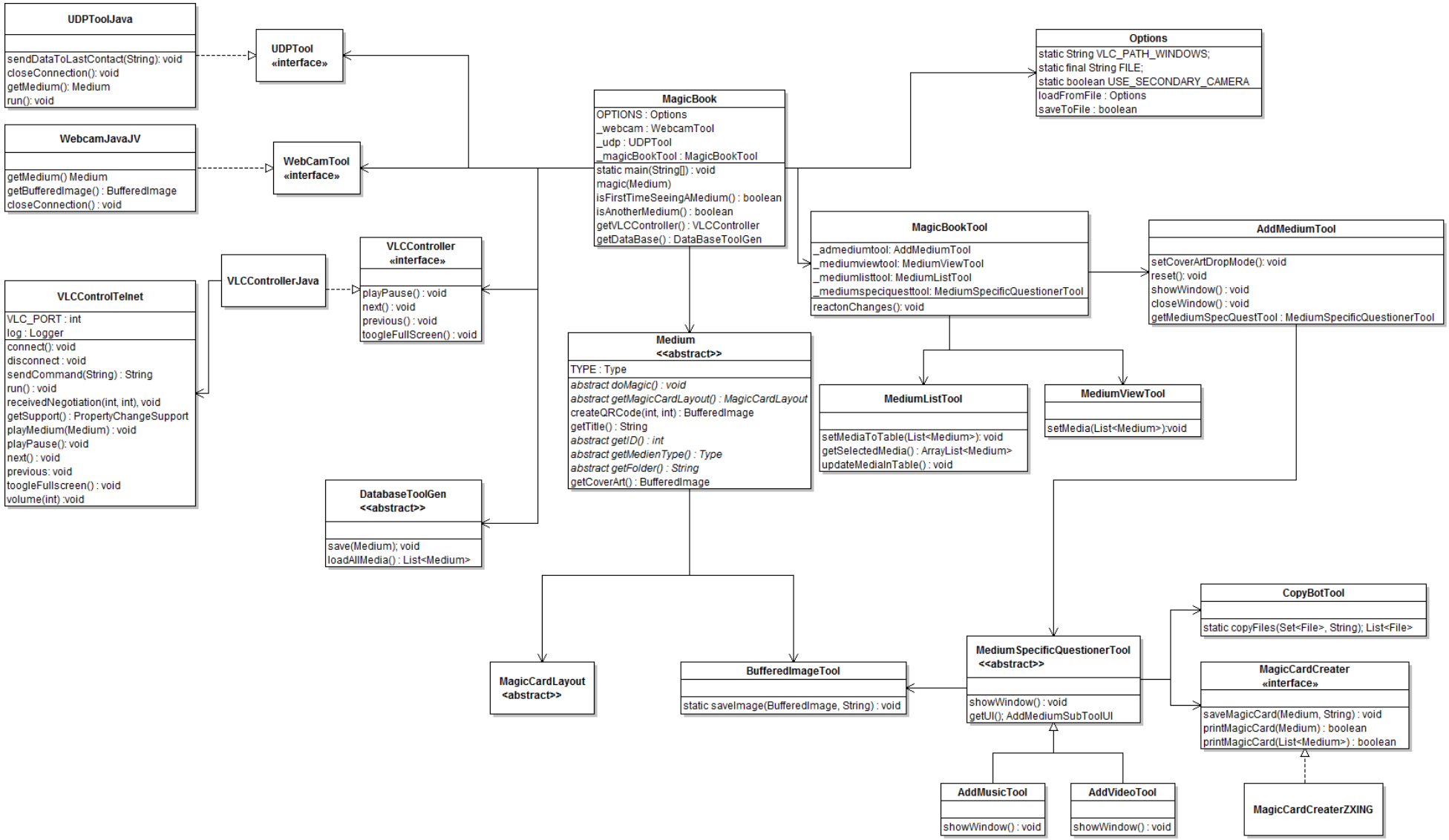
Da **magicBook** mit lokalen Spiegelungen der Daten arbeitet, werden die Medien in die Medien-Verzeichnisse der Anwendung kopiert und abschließend ein Datensatz in der Datenbank erzeugt. Für jede Gruppe von Medien wird dabei ein Hash-Wert erzeugt, der ebenfalls in der Datenbank mitgespeichert wird, um die Medien eindeutig identifizieren zu können. Anschließend kann der Nutzer eine **magicCard**,die mit einem QR-Code versehen ist und im Medienverzeichnis liegt, ausdrucken.

Falls Medien in der Datenbank vorhanden sind, so kann der Benutzer mithilfe von einer Webcam (integriert oder extern) oder mit dem Smartphone (Barcode-Scanner App vorausgesetzt) eine erzeugte **magicCard** einscannen und zugewiesene Mediendateien abspielen (über VLC). Wie in der obigen Grafik dargestellt, wird von **magicBook** regelmäßig überprüft, ob Medien zum Abspielen bereit stehen. Dies geschieht entweder durch Auswertung des gelieferten Bild von der Webcam oder durch Übermittlung der Daten über das Netzwerkprotokoll UDP vom Smartphone .

magicBook ermittelt anhand der übermittelten bzw. ausgewerteten (aus QR-Code) Hash-Codes die abzuspielenden Mediendateien und delegiert z.B. bei Musik- und Videodateien VLC zum Abspielen der Mediendateien.

Es wurde bewusst davon abgesehen, die Wiedergabe der Medien selber zu implementieren,da hier ständig Format-Neuerungen oder neue Codecs entstehen. Außerdem stellt VLC eine stabile,freie und zudem noch sehr flexible Medienabspielsoftware dar, weshalb es sich als Basis für **magicBook** sehr gut eignet.

Klassenübersicht in UML(grob, von Details abstrahiert)



User-Stories nach Priorität sortiert

1. kreatives Erstellen von magicCard

Beschreibung:

Für jedes Medium kann nach vordefinierten Layouts eine **magicCard** erstellt werden. Die sog. **magicCards** sind reale Repräsentation von digitalen Medien, die ausgedruckt werden können. Diese Karten enthalten neben den eigentlichen QR-Code, der eindeutig auf ein Medium verweist zusätzliche Information wie das Cover o.ä. .

Zeitaufwand

10 Stunden /...

2. Benutzung von Peripherie-Geräten : Webcam / Smartphone zum Einscannen von QR-Codes

Beschreibung:

Die Anwendung kommuniziert mit der Webcam des zugrundeliegenden Rechners bzw. eines im Netzwerk befindlichen Smartphones. Die Anwendung verarbeitet Daten, die von der Kamera/ Webcam übermittelt werden.

Zeitaufwand

15 Stunden (Einarbeitung in JavaCV ,ZXING,UDP und Entwicklung von Android-App)

3. Auswerten von QR-Codes

Beschreibung:

Der erkannte QR-Code sollte interpretiert werden und eine entsprechende Aktion auslösen(z.B. Mediendateien abspielen, Informationen zum Medium anzeigen oder Programme starten.

Zeitaufwand

8 Stunden (ZXING,SQL-Datenbank)

4. Auswahl der Medien

Beschreibung:

Es können Mediendateien aus dem Dateisystem und Programme ausgewählt und anschließend zu einer zu einer Liste hinzugefügt werden.

Zeitaufwand

geschätzt 3 Stunden / tatsächlich 2 Stunden (mit Drag&Drop)

5. Speicherung der Medien (SQL)

Beschreibung:

Für jedes Medium (bzw. Menge von Medien) wird ein Eintrag in der Datenbank erzeugt. Die Einträge aus der Datenbank werden in der UI angezeigt.

Zeitaufwand

5 Stunden (JDBC-Treiber) / tatsächlich 4 Stunden

6. Automatisches Laden von Metainformationen

Für jedes hinzugefügte Medium können Metainformationen nachträglich geladen werden. Diese Meta-Daten stammen aus externen Datenbanken.

Zeitaufwand

nicht implementiert

Beschreibung der (technischen) Umsetzung

Die Anwendung beinhaltet eine Vielzahl von Tools und Controller-Klassen, die eine Kommunikation mit Peripherie-Geräten oder Teilnehmern des Netzwerks bewerkstelligen.

Ausgehend von einer Hauptklasse (magicBook) werden die anderen benötigten Tools initialisiert. Im Kontext der Hauptklasse wird auch die Erzeugung der UI ausgelöst. Im Folgenden wird näher auf einzelne,wichtige Tools (Werkzeuge) eingegangen:

QRCodeCreatorZXING, ORCodeReaderZXING

Diese Klassen stellen Implementierungen von QRCodeCreator bzw. QRCodeReader dar, und ermöglichen das Erstellen und Lesen von QR-Codes. Diese Klassen nutzen ZXING[1], eine externe Library für den Umgang mit Barcodes (in unserem Fall QR-Codes).

wichtige Methoden:

getQRCodeInformation(BufferedImage) : String[]

Erkennt innerhalb eines BufferedImage den QR-Code und ermittelt Daten(Medientyp und Hashcode), die im QR-Code enthalten sind.

[1] ZXING, erreichbar unter https://code.google.com/p/zxing/

UDPToolJava

Diese Klasse implementiert einen „UDP-Package-Listener“ d.h. öffnet einen Thread, in dem ein java.net.DatagramSocket auf dem Port 2562 mit einer Android-App kommunizieren kann. Die IP-Adresse des zugehörigen Smartphones wird aus dem ersten empfangenen Packet gelesen (Broadcasting).

Es wird an dieser Stelle mit dem Beobachtermodell gearbeitet: UDPToolJava implementiert Observable, und die Hauptklasse (magicBook) den dazugehörigen Observer, wodurch die Hauptklasse immer über neue Packets benachrichtigt wird, und auf die eingescannten **magicCards** bzw. auf Button-Events aus der Android App reagieren kann.

Da magicBook eine Heimanwendung ist, liegt ein Szenario vor, das nur **ein** Smartphone vorsieht, auf dem die Android App läuft.

Sollte in Zukunft ein „Tauschmodus“ eingebaut werden (wodurch mehrere User durch magicCards stöbern und mit ihrem Smartphone die magicCards anderer magicBooks einscannen können, die sie gerne auf ihr mitgebrachtes Speichermedium kopieren möchten) so muss dieses Tool entsprechend erweitert werden.

wichtige Methoden:

Ein UDP-Packet vom PORT empfangen:

```
DatagramSocket socket = new DatagramSocket(PORT);

socket.receive(new DatagramPacket(...));
```

Ein UDP-Packet schicken:

```
socket.send(new DatagramPacket());
```

VLCControlTelnet

VLCControlTelnet implementiert das VLCControl-Interface mittels der Funktionalität der Apache Commons – Library[2]. Die Kommunikation mit VLC findet mittels Telnet statt. Hierfür ist es wichtig, dass VLC mit dem entsprechenden Telnet-Interface gestartet wird. Dies kann über folgenden Aufruf passieren, und wird von **magicBook** auch so vollzogen:

vlc –-intf qt –-extraintf Telnet

Der Standard-Telnet-Port von VLC ist momentan 4212, deshalb wird dieser standardmäßig auch von magicBook benutzt. Nach dem magicBook sich über Telnet mittels des Standardbenutzerkontos (Standardpasswort: admin) bei VLC angemeldet hat, kann man fast jede Funktion von VLC fernsteuern.

[2] Apache Commons Library erreichbar unter : https://commons.apache.org/

wichtige Methoden:

sendCommand(String command);

Schickt Befehle an VLC.

playMedium(Medium m);

Spielt das Medium ab, indem es sendCommand(...) benutzt.

WebcamJavaCV

Diese Klasse implementiert das WebcamTool-Interface und stellt somit mit Hilfe der JavaCV-Library (Java Anbindung an OpenCV) eine Schnittstelle dar, um von der Webcam Bilder zu erhalten, auf welchen dann im Anschluss mit QRCodeReaderZXING QR-Codes gefunden und gelesen werden können.

CopyBotTool

Dieses Tool ist in der Lage eine gegebene Liste von Dateien mit dazugehörigem Zielpfad zu spiegeln bzw. zu kopieren

wichtige Methoden:

List<File> newFiles = copyFiles(List<File> oldfiles,String newPath);

BufferedImageTool

Dieses Tool ist für den Umgang mit BufferedImages geeignet. Es lässt laden,speichern und skalieren zu.

wichtige Methoden:

saveImage(BufferedImage new, String pfad);

BufferedImage scaled = scaleImage(BufferedImage old,int newWidth,int newHeight);

BufferedImage img = loadImage(File file /String pfad);

DataBaseSQLTool

Dieses von DataBaseToolGen abgeleitete Tool stellt die Schnittstelle zur SQLite[3]-Datenbank dar.

Als JDBC-Treiber wird SQLite JDBC[4] eingesetzt.

[3] SQLite , ausführliche Dokumentation erreichbar unter https://sqlite.org/docs.html

[4] Xerial SQLite JDBC von Taro L. Saito , erreichbar unter https://bitbucket.org/xerial/sqlite-jdbc/overview

wichtige Methoden:

save(Medium) ;

loadMediumForType(String, Stirng);

loadMediaByTerm(String);

Sucht Medien anhand eines Suchbegriffs.

Installation bzw. Was wird benötigt?

Folgende Libraries werden von der aktuellen magicBook-Version (2.1) benutzt :

JavaCV (→ OpenCV)	(Alternative Implementation vorhanden mit: JMyron)
SQLite JDBC-Driver	(Alternative Implementation vorhanden mit: XStream mit „Ordnerstruktur“)
ZXING (→ QR-Codes)	
FileDrop (Drag'n'Drop von Files)	
Apache Commons Telnet	JUnit

Um die Binary von magicBook zu starten sollten folgende Dinge auf dem System vorinstalliert sein:

OpenCV (2.4.5 oder höher) und eine damit kompatible **Webcam** (es wird fast jede Webcam unterstützt)

VLC (2.0.5 oder höher)

BeginnersGuide

Die Anwendung lässt sich über die ausführbare .jar-Datei starten. Nun öffnet sich die Benutzeroberfläche. Im oberen Teil des Fensters lassen sich Medien hinzufügen und anzeigen. Im unteren Teil wird eine Liste mit aktuell in der Datenbank vorhandenen Medien dargestellt, die durchsucht werden kann.

1.Medien hinzufügen

Wenn Sie Medien zu der Datenbank hinzufügen möchten, ziehen Sie einfach die gewünschten Dateien per Drag&Drop in das große Feld („Drop your media files here“). MagicBook erkennt automatisch, welcher Typ von Medien ausgewählt ist. Alternativ lassen sich Mediendateien auch durch Angabe einer URL hinzufügen.

Durch Klicken auf den ***Next***-Button gelangen Sie zum nächsten Schritt. Falls Sie ihre Auswahl rückgängig machen wollen, klicken Sie auf die Schaltflächen ***Reset*** oder ***Cancel***.

Falls Sie sich für ***Next*** entschieden haben, kann nun ein Cover oder allgemein Bilddatei ausgewählt werden. Dies kann wieder per Drag&Drop oder unter Angabe einer URL durchgeführt werden.

Anschließend kann durch Klicken auf den Button „***Add MUSIC“*** (falls sie Musik im vorigen Schritt ausgewählt haben und hinzufügen möchten), ein weiteres Fenster geöffnet werden. In diesem Fenster können Metainformationen wie Titel, Interpret bzw. Regisseur oder das Jahr eingegeben werden.

Durch Klicken auf ***Finish: Save!*** *w*erden die Medien von MagicBook in die Datenbank eingefügt. Für jede Gruppe von Medien, die zur Datenbank hinzugefügt werden, wird ein Verzeichnis unter medium/<Typ des Mediums> angelegt. In diesem Verzeichnis wird neben den eigentlichen Mediendateien noch eine MagicCard(Bild-Datei) gespeichert.

2.Medien abspielen

Um Medien abzuspielen, einfach die gewünschte (unter **1.** erstellte) MagicCard ausdrucken, und vor die eigene Webcam halten bzw. mit dem Smartphone über die mitgelieferte App einscannen. Der VLC-Mediaplayer wird gestartet und spielt die Medien ab.

Tests

Die fachlichen Klassen besitzen jeweils eine Testklasse und darüber hinaus viele weitere Tools(Werkzeug-Klassen). Die Interaktion wird im Rahmen der Test-Szenarien nicht getestet. Viele verwendete externe Libraries wurden bereits intensiv getestet, deshalb wurde beswusst davon abgesehen, diese erneut einem Test zu unterziehen.

aktueller Stand des Projektes

•Thread für die Speicherung eingeführt (SaveThread...)

•Tauschmodus noch nicht implementiert

•Bilder,Programme,URLs, RealLifeObjects als Medien vorgesehen, Implementationen teilweise angefangen

•Android App ist veraltet, sollte überarbeitet werden

•Portabilitätsanpassungen (Kommunikation mit Webcam bereitet teilweise Probleme auf verschiedenen Systemen, mit diversen Webcams)

bekannte Fehler (Bugs)

- Pufferüberlauf bei den von der Android App per UDP übermittelten Strings -> Überarbeitung der App bzw. Trennzeichen einführen
- Es ist möglich, dass nachdem eine magicCard mihilfe der Webcam eingescannt wurde, anschließend keine magicCards die per Smartphone eingescannt werden, ausgeführt werden. Ein Neustart der Anwendung + Schließen von VLC hilft.
- Beim Einscannen mit dem Smartphone ist es erforderlich den QR-Code zweimal einzuscannen. Beim ersten Einscannen wird zunächst die IP-Adresse des Geräts ermittelt.

Geplante Erweiterungen und Verbesserungen

Die Portabilität ist eine wesentliche Eigenschaft von magicBook. Die Anwendung soll von einem portablen Speichermedium auf allen gängigen Betriebssystemen gestartet werden können. Die Unterstützung für MacOS ist für die nächste Zeit vorgesehen. Zur Zeit wird daran gearbeitet, die Interoperabilität zu verbessern, insbesondere auch die Anbindung von Webcams (verschiedenster Hersteller) und die Kommunikation mit VLC.

Neben Videos und Musik ist die Verarbeitung von Programmen, Bildern, URLs geplant. Durch konsequente Verwendung von abstrakten Klassen und Interfaces können die entsprechenden Implementation(nahezu fertig) schnell erstellt.

Darüber hinaus ist geplant, den Nutzer verstärkt über den aktuellen Status zu informieren, u. a. über Fehlermeldungen oder Wartedialoge. Aktuell fehlt noch die Möglichkeit Medien nachträglich zu löschen. Dies wird in einem der nächsten Updates eingeführt.

In nächster Zeit soll Funktionsumfang von **magicBook** ausgebaut werden. Geplant ist die Möglichkeit, dass der Nutzer die **magicCards** direkt aus der Anwendung heraus drucken kann.

Sobald die restlichen Medientypen (Programme, Bilder, URL) unterstützt werden, sollen kombinierte Aktionstypen eingeführt werden. D.h. der Anwender kann bestimmte in der Datenbank hinterlegte Medien kombinieren und per **magicCard** ausführen. Denkbar ist z.B. eine Kombination von Programmen und URL, sodass beim Einscannen entsprechender **magicCards**, der Mail Client und bestimmte URL im Browser aufgerufen werden.

Des Weiteren streben wir eine Verschlüsselung der Daten an.

Fazit

Der Workflow war sehr gut und insgesamt ist es ein Projekt, dass durch die Vielseitigkeit, sehr viel Spaß bereitet hat, weshalb auch die ein oder andere Nacht mit weniger Schlaf verbunden war, weil man wieder von einem Problem gefesselt war.

MagicBook entpuppt sich nach und nach als gute Basis, um darauf aufbauend neue Ideen, wie den „Tauschmodus“ oder kombinierte Medientypen, entstehen zu lassen.

Ein Test von **magicBook** auf dem „Keine Knete - Trotzdem Fete Festival 2013“ ergab sehr viel positives Feedback. Dafür wurde eine Webcam in einem Baum versteckt und eine 20m USB-Kabel-Verlängerung benutzt, um den Laptop, der letztlich die Musik abspielt in den Hintergrund rücken zu lassen. So war es den Festivalbesuchern in einem kleinen Naturareal möglich, mit über die Musikalben, die dort laufen abzustimmen bzw. diese zu beeinflussen, indem sie vorgefertigte **magicCards** nun vor diesem Baum einscannen konnten.

MagicBook lehnt die Kommerzialisierung ab, weshalb wir den Quelltext öffentlich zur Verfügung stellen werden. Eine Software kann nur überleben und dem Nutzer wirklich nutzen, wenn sie frei und dadurch flexibel bzw. anpassungsfähig ist.

Diese Freiheit wird durch eine Kommerzialisierung zerstört, aus dem einfachen Grund, dass die Software nicht mehr allen so transparent zugänglich wäre, wie sie es sein sollte und nur durch ein kostenloses OpenSource-Release mit einer angemessenen Dokumentation möglich ist.

Lessons Learned

Abschließend lässt sich feststellen, dass der Ansatz der Test-Driven Development es ermöglicht, dass durch umfangreiche Test-Szenarien Programmfehler deutlich früher entdeckt und behoben werden können.

Eine ausführliche Dokumentation ist nicht nur für spätere Anwender essentiell, sondern vermittelt dem Entwickler einen Eindruck über Software aus objektiver Perspektive.

Eine sorgfältige Planung(inklusive Aufwandsschätzung) kann die Projektdauer erheblich verkürzen. Insbesondere das Pareto-Prinzip macht deutlich das gefühlter Projektfortschritt oftmals von dem tatsächlichen Projektfortschritt abweicht.