|  |
| --- |
| HTL perg |
| Mobile SVG Editor |
| SVG |
|  |
| Markus Heilmann  Manuel Baumgartner |
| **2/20/2013** |

|  |
| --- |
|  |

Diplomarbeit

Fach Programmieren

**für Aberger Software GmbH**

Eingereicht an der

**HTL Perg**

Betreuung von

**DI Christian Aberger**

Eingereicht von

**Markus Heilmann**

**Manuel Baumgartner**

**14. Mai 2013**

Mobile SVG Editor

*Von Markus Heilmann, Manuel Baumgartner*

Diplomarbeit im Fachgebiet Programmieren,

Abteilung Elektronische Datenverarbeitung & Organisation,

HTBLA Perg,

14. Mai 2013

Zusammenfassung

Das Projekt „Mobile SVG Editor“ wurde für den Auftraggeber Aberger Software GmbH erstellt und dient dazu, Postkarten mit Hilfe eines Android Gerätes zu bearbeiten und zu bestellen.

Dabei werden in einem ersten Schritt Templates von einem Server auf das Gerät übertragen. In diese Vorlagen können Bilder, aus dem Android Gerätespeicher, integriert werden. Das ausgewählte Bild kann dann im Template gezoomt, gedreht und verschoben werden. Auf der Rückseite der Postkarte können noch die Kontaktdaten des Kunden eingetragen werden.

Das fertige Produkt wird dann wieder auf den Server geladen, damit der Auftrag von der Firma bearbeitet werden kann.

Die Gestik-Steurung von Bildern, wie das Zoomen, Drehen und Verschieben, ist dabei das Hauptziel der Diplomarbeit.

Mobile SVG Editor

*By Markus Heilmann, Manuel Baumgartner*

Diploma thesis in Programming,

College of electronic data processing and organisation specialising in software engineering Perg,

14. Mai 2013

Abstract

The Project “Mobile SVG Editor” was produced for the company Aberger Software GmbH. The benefit of the Application is that you can process and order postcards with an android device.

In a first step the templates will be downloaded to the Device. Pictures can be integrated into those templates from the Android Device Space. In the template the picture can be zoomed, rotated and moved. On the backside the costumer can enter his personal data.

The Product of the costumer will be uploaded to a server, so that the company can process the assignment.

The thesis main goal is the gestures control, such as zooming, rotating and moving.

Eidesstattliche Erklärung

Im Zuge dieser eidesstattlichen Erklärung versicher wir, Markus Heilmann & Manuel Baumgartner, dass diese Diplomarbeit mit dem Titel

*„Moblie SVG Editor“*

selbständig verfasst und ohne fremde Hilfe und der Benutzung anderer als der angegebenen Quellen angefertigt haben.

Jene Teilbereiche, die anderen Werken bzw. Internetpublikationen wortwörtlich oder dem Sinn gemäß entnommen wurden, haben wir in jedem einzelnen Fall durch eine Quellenangabe als solche kenntlich gemacht.

Perg, am 14. Mai 2013

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Markus Heilmann | Manuel Baumgartner |

Motivation

Zum Abschluss der Ausbildung ist eine Diplomarbeit natürlich ein probates Mittel, unsere erlenten Kenntnisse und Fähigkeiten in die Praxis umzusetzen.

Wir entschieden uns für die Gestaltung von Postkarten auf Mobilen Android Geräten, weil für uns der Mobile Sektor sehr wichtig ist. Des weiteren interessierten wir uns sehr für die Gestik-Steuerung an Touch-Eingabegeräten.

Ebenso wie die Erstellung und Umsetzung gehört die Päsentation und Vermarktung zum Alltag eines Absolventen in der Arbeitswelt und somit war es auch uns ein Anliegen, abseits der technischen Realisierung, die wirtschaftliche Seite miteinzubeziehen.

Somit kamen wir zu einem interessanten und abwechslungsreichen Diplomarbeitsthema.

Danksagung

An dieser Stellle möchten wir uns bei allen Personen bedanken, die diese Arbeit ermöglicht haben und uns dabei auch tatkräftig unterstützten.

Ein großer Dank gebührt unserem Betreuungslehrer DI Christian Aberger, der, trotz seiner Tätigkeit, als Geschäftsleiter einer Firma, immer wieder Zeit fand, uns eine entsprechende Kritik zu geben.

Auch bei fachlichen Problemstellungen gab er uns, durch sein fachliches Wissen, gute Ratschläge, damit die Diplomarbeit ein voller Erfolg wurde.

Weiters möchten wir uns bei der Firma Aberger Software GmbH bedanken, die uns die benötigten Ressourcen zur Verfügung stellten.

Die Mitarbeiter des Unternehmens gaben uns auch bei Fragen der Optik und des Designs gute Ratschläge zur Umsetzung.

Abschließend möchten wir uns bei der Höheren Technischen Lehranstalt Perg bedanken, die uns die Durchführung der Diplomarbeit ermöglichte.

Inhaltsverzeichnis

[Einleitung 10](#_Toc354682549)

[Diplomanden 10](#_Toc354682550)

[Markus Heilmann 10](#_Toc354682551)

[Manuel Baumgartner 11](#_Toc354682552)

[Auftraggeber 12](#_Toc354682553)

[Verwendete Technologien 13](#_Toc354682554)

[Android 13](#_Toc354682555)

[Versionen inkl. Erneuerungen 14](#_Toc354682556)

[Eclipse 17](#_Toc354682557)

[Feature 18](#_Toc354682558)

[Plug-Ins 18](#_Toc354682559)

[Subversion 19](#_Toc354682560)

[Funktionales Spezifikationsdokument 20](#_Toc354682561)

[Allgemein 20](#_Toc354682562)

[Zusammenfassung 20](#_Toc354682563)

[Voraussetzungen 21](#_Toc354682564)

[Abhängigkeiten 21](#_Toc354682565)

[Prozess Spezifikation 22](#_Toc354682566)

[Template-Auswahl bzw. Erweiterung 22](#_Toc354682567)

[SVG-Dateien bearbeiten 22](#_Toc354682568)

[Fertiges Bild hochladen 23](#_Toc354682569)

[Statistiken anzeigen 23](#_Toc354682570)

[Kostenplanung 24](#_Toc354682571)

[Arbeitsleistung 24](#_Toc354682572)

[Lizenzkosten 25](#_Toc354682573)

[Hardwarekosten 26](#_Toc354682574)

[Gesamtkosten 27](#_Toc354682575)

[Umsetzung 28](#_Toc354682576)

[Eclipse einrichten 28](#_Toc354682577)

[Verwendete Softwareschnittstellen 28](#_Toc354682578)

[SVG 28](#_Toc354682579)

[XML 29](#_Toc354682580)

[Aufbau 30](#_Toc354682581)

[ZIP 38](#_Toc354682582)

[Ziele und Anforderungen 40](#_Toc354682583)

[Gestik Steuerung 41](#_Toc354682584)

[Zoom-Gestik 41](#_Toc354682585)

[Rotations-Gestik 42](#_Toc354682586)

[Transformations Matrix 44](#_Toc354682587)

[SVG-Matrix 45](#_Toc354682588)

[Download von Templates 46](#_Toc354682589)

[Entpacken von ZIP-Dateien 47](#_Toc354682590)

# Einleitung

## Diplomanden

### Markus Heilmann

|  |  |  |
| --- | --- | --- |
| **Persönliche Daten** | | |
| Geburtsdaten | Amstetten, 4. Jänner 1994 | |
| Staatsangehörigkeit | Österreich | |
| Religionsbekenntnis | röm.-kath. | |
| Familienstand | ledig | |
| Eltern | Josef Heilmann (50), Landwirt  Elisabeth Heilmann (43), Landwirtin | |
| Geschwister | Lisa Heilmann (17), Berufstätige  Florian Heilmann (12), Schüler  Magdalena Heilmann (8), Schülerin | |
|  | |  |
| **Schulausbildung** | | |
| 2000-2004 | Volksschule in Waldhausen | |
| 2004-2008 | Hauptschule in Waldhausen | |
| seit Sep. 2008 | HTL für EDV und Organisation in Perg | |
|  | |  |
| **Arbeitserfahrung** | | |
|  | Firma **Gmeiner GmbH**, Waldhausen  Juli 2010 | |
|  |  | |
|  | Firma **BIOKOMPAKT Heiztechnik GmbH**, Waldhausen  Juli 2011 | |
|  | Firma **Aberger Software GmbH**,  Hagenberg im Mühlkreis  Juli 2012 | |
| **Kontakt** | | |
| Telefon | +43 664 922 77 44 | |
| E-Mail | Markusheilmann94@gmail.com | |

### Manuel Baumgartner

|  |  |  |  |
| --- | --- | --- | --- |
| **Persönliche Daten** | | | |
| Geburtsdaten | Linz, 22. Jänner 1994 | | |
| Staatsangehörigkeit | Österreich | | |
| Religionsbekenntnis | röm.-kath. | | |
| Familienstand | ledig | | |
| Eltern | Nikon Baumgartner (47), Lokführer  Birgit Baumgartner (39), Sozialkindergärtnerin | | |
|  | |  | |
| **Schulausbildung** | | | |
| 2000-2004 | Volksschule 1 in Enns | | |
| 2004-2008 | Hauptschule 1 in Enns | | |
| seit Sep. 2008 | HTL für EDV und Organisation in Perg | | |
| Sep. 2012 – Dez. 2012 | SO Zenit (Schüleraustausch mit Belgien) | |
| **Arbeitserfahrung** | | | |
|  | Firma **BBRZ Reha GmbH**, Linz  Juli 2010 | | |
|  |  | | |
|  | Firma **Kremsmüller Industrieanlagenbau KG**, Steinhaus bei Wels  Juli 2011 | | |
|  | Firma **Voestalpine Stahl GmbH**, Linz  Juli-August 2012 | | |
| **Kontakt** | | | |
| Telefon | +43 660 481 63 20 | | |
| E-Mail | baumi494@gmail.com | | |

## Auftraggeber

Der Auftraggeber der Diplomarbeit „Mobile SVG-Editor“ ist

# Verwendete Technologien

## Android

Android ist ein Betriebssystem für mobile touchscreen Geräte (Handys, Tablets, …) und basierend auf Linux. Die erste Version wurde 2007 von der Open Handset Alliance entwickelt. Diese Vereinigung widmete sich der Verbesserung der Standardisierung mobiler Endgeräte und brachte schließlich im Oktober 2008 das erste Android-Smartphone heraus.

Grundsätzlich ist das OS Open-Source, dies erlaubt es Geräteherstellern und Entwicklern das System weiterzuentwickeln und weiterzuverkaufen.

Darüber hinaus ist es mit der Programmiersprache Java und der IDE Eclipse möglich, eigene Apps zu schreiben und in den Google Play-Store hochzuladen.



Abbildung 1 - Android Logo

Im Vergleich zum iOS-System (iPhone, iPod) bietet Android wesentliche Vorteile, jedoch auch einige Nachteile.

Auf allen Android-Mobilgeräten (Hauptsächlich von den Herstellern: Samsung, HTC, Sony und LG) gibt es verschiedene Versionen, die sich teilweise vom Design und von den Fähigkeiten unterscheiden. Da die Entwickelte Version aufwärtskompatibel (z. B. Android 2.1 läuft auf allen Geräten, die die 2.1 Version haben oder eine höhere) ist, sollte man sich auf den „kleinsten gemeinsamen Nenner“ einigen, der meistens bei 2.2 (Froyo) und von fast jedem Gerät unterstützt wird.

### Versionen inkl. Erneuerungen

|  |  |  |  |
| --- | --- | --- | --- |
| Version | API | Veröffentlichung | Neues |
| 1.0 | 1 | 09.2008 | * Google-Apps * Youtube * Synchronisation |
| 1.1 | 2 | 02.2009 | * MMS-Anhänge speichern |
| 1.5 Cupcake | 3 | 04.2009 | * Automatische Bildschirmrotation * Bildschirm-Tastatur * Kamerafunktionen * Bluetooth |
| 1.6 Donut | 4 | 09.2009 | * VPN[[1]](#footnote-1) * Energieverbrauchsteuerung * Suchfunktion * TTS[[2]](#footnote-2) |
| 2.0 Eclair | 5 | 10.2009 | * Digitalzoom * Blitzlicht * Microsoft-Exchange * Bluetooth 2.1 |
| 2.0.1 | 6 | 12.2009 | * Bugfixes |
| 2.1 | 7 | 01.2010 | * animierte Wallpaper * Signalstärke-Informationen * Webkit[[3]](#footnote-3) * IPv6 |
| 2.2 Froyo | 8 | 05.2010 | * mehr RAM * OpenGL, Flash * Tethering[[4]](#footnote-4) * App-Daten auf der SD-Karte * Cloud-Anbindung * Bluetooth-Sprachwahl * (Bugfixes bei 2.2.1 und 2.2.2) |
| 2.3 Gingerbread | 9 | 12.2010 | * WebM[[5]](#footnote-5) * HTML5 * Google TV * NFC[[6]](#footnote-6) * ruckelfreie Animationen * soziale Netzwerke, Gyroskop[[7]](#footnote-7), Download-Manager, VoIP-Client, Ext4-Dateisystem |
| 2.3.1 | 9 | 12.2010 | * Bugfixes, Google-Maps update |
| 2.3.2 | 9 | 01.2011 | * Fix für SMS-Bug |
| 2.3.3 | 10 | 02.2011 | * Dual-Core Verbesserung * Single-Core Kompatibilität * NFC-Technik * Bluetooth Verbesserung |
| 2.3.4 | 10 | 04.2011 | * Google Talk, Verschlüsselte Termin- und Kontaktdatenübertragung |
| 2.3.5 | 10 | 07.2011 | * 4G-Unterstützung * Volle Unterstützung von NFC |
| 2.3.6 | 10 | 09.2011 | * Voice-Suche verbessert, Bugfixes |
| 2.3.7 | 10 | 09.2011 | * Bugfixes |
| 3.0 Honeycomb | 11 | 02.2011 | * benutzerfreundliche Oberfläche * besser für Tablet-Computer * Google Talk mit Videotelefonie * neue Browserfunktionen (Google Chrome, Inkognito-Modus) |
| 3.1 | 12 | 05.2011 | * USB-Host-Modus * Widgets * Erweiterungen am Browser * Proxy-Einstellungen für jedes LAN |
| 3.2 | 13 | 07.2011 | * Anpassung der Bildschirmdiagonale von 7 Zoll * Kompatibilitätsmodus zur Darstellung von Apps mit niedrigerer Auflösung * SD – USB Synchronisation * CPU-Hardware Erweiterung * bessere APIs für verschiedene Auflösungen |
| 3.2.1 | 13 | 09.2011 | Bugfixes |

|  |  |  |  |
| --- | --- | --- | --- |
| 4.0 | 14 | 09.2011 | * Zusammenführung von 2.x und 3.x und Google TV * Multitasking * NFC für Kontakte, Daten und Links mit Android Beam * Gesichtserkennung (Entsperren), verbesserte Google-Apps * Data-Tracking, Screenshot-App, Detailverbesserungen |
| 4.0.1 | 14 | 11.2011 | * Galaxy-Nexus version, Lautstärkebugs-Behebungen |
| 4.0.2 | 14 | 12.2011 | * neuer Linux-Kernel * VPN-Optimierung * 3G/4G mit schnellen Reaktionszeiten * falsche Benachrichtigung behoben (Roaming) * Verbesserte optische Effekte beim Camcorder mit der Frontkamera * E-Mail-Anhänge korrekte Anzeige * DivX-Unterstützung entfernt * verbesserte Lautstärke bei Turn-By-Turn-Nativation von Google |
| 4.0.3 | 15 | 12.2011 | * neue APIs, erste offizielle Version für Galaxy Nexus * Source für Dritthersteller |
| 4.0.4 | 15 | 02.2012 | * verbesserte Performance (Bildschirmdrehung, Kamera) * bessere Rufnummernerkennung * Lichtsensor-Verbesserung, * allgemeine Laustärke erhöht, * Multitouch-Bugs behoben * über 100 weitere Änderungen und Bugfixes |
| 4.1 Jelly Bean | 16 | 06.2012 | * Butter Project[[8]](#footnote-8) * Konstante Bildwiederholfrequenz * Verbesserte Reaktionsfähigkeit * Verbesserte Bildschirmtastatur bzw. Braille-Unterstützung * Erweiterte Notifications und Benachrichtigungen mit Aktionen * Google-Now * Diktieren ohne Internet * Einfacher anzupassender Startbildschirm * NFC mit Android Beam 2.0 * Mahrkanalaudio * USB-Audio (für externe DACs[[9]](#footnote-9)) * Mehrkanalaudio * Audio-Chaining (Unterbrechungsfreie Wiedergabe) * Unterstützung bidirektionaler Schriftsysteme, 18 neue Eingabesprachen, Emoji-Darstellung * WPS[[10]](#footnote-10)-Unterstützung |
| 4.1.1 | 16 | 07.2012 | * Unlock-Screen ohne Verzögerung * Google-Now flüssiger * Google-Wallet |
| 4.1.2 | 16 | 10.2012 | * Rotation des Lock/Home-Screens * Auf- und Zuklappen von Notifications mit nur einem Finger |
| 4.2 | 17 | 11.2012 | * Benutzer-Konten (Tablets) * Photo Sphere (Kugelpanoramen) * Tastatur mit Wortvorhersage * Systemweiter Streaming * Daydream (Bildschirmschoner) * Interaktive Navigationsleiste * Verbesserte Sicherheitseinstellungen |
| 4.2.1 | 17 | 11.2012 | * Bug in Kontakte behoben * Quick-Settings zum De/Aktivieren von WLAN und Bluetooth |

## Eclipse

Eclipse ist eine gemeinnützige Community, die Einzelpersonen und Organisationen dabei helfen soll, kommerzielle Open-Source-Software zu entwickeln. Die Projekte der Foundation konzentrieren sich auf den Aufbau einer offenen Entwicklungsplattform aus erweiterbaren Frameworks, Tools und Laufzeitumgebungen für den Aufbau, den Einsatz und die Verwaltung von Software, die den gesamten Lebenszyklus umfassen.



Abbildung 2 - Eclipse Logo unserer Entwicklungumbebung Indigo

### Feature

Viele Komponenten in Eclipse sind so umfangreich, dass es unhandlich wäre, sie in einem einzelnen Plug-In zu "verpacken". Daher können beliebige Plug-Ins zu einem Feature zusammengefasst werden. Einige der Features in Eclipse sind z.B. das von uns verwendete "JDT" (Java Development Tools, also der Java Compiler, Java Debugger, Java Perspektive, ...).

### Plug-Ins

Ein Plug-In ist eine in sich abgeschlossene Komponente, die Verbindungen zu anderen Plug-Ins herstellen kann und so beliebig erweitert werden kann. Der eigentliche Kern von Eclipse ist ein einfaches Programm, das in der Lage ist, Plug-Ins zu laden und auszuführen. *Jede* andere Funktionalität in Eclipse wurde durch Plug-Ins hinzugefügt. Der Java Compiler, den Eclipse benutzt, ist ein Plug-In. Der Debugger in Eclipse ist ein Plug-In, usw. Diese Architektur ist das wesentliche Merkmal von Eclipse und erlaubt es uns Eclipse beliebig zu erweitern und zu ergänzen.

#### ADT Plugin

Das Android Development Tool ist ein Plugin für Eclipse, welches eine mächtige Entwicklungsumgebung bietet, um Android Applikationen zu erstellen.

Dieses Tool erweitert Eclipse und hilft uns, sehr schnell neue Android Projekte zu generieren, eine Benutzeroberfläche per Drag and Drop zu erstellen und automatisch ein „.apk“ File zu erstellen. Diese Datei wird durch Eclipse automatisch beim Ausführen des Programmes auf einem Android-Gerät installiert.

Darüber hinaus ist es durch das Plugin möglich, das Programm mittels Debugger nach Fehlern zu durchsuchen.

#### Subclipse

Dieses Plug-In ist frei verfügbar und ermöglicht es, in Eclipse mit Subversion (SVN) zu arbeiten. Es unterstützt unter anderem Team-Programmierung, indem es Files in denen gleichzeitig programmiert wird, zusammenführen kann. Ein anderes Feature ist, dass aus bestehenden Applikationen Branches erstellt werden können, auf diesen dann autonom programmiert und schließlich wieder in das eigentliche Projekt eingebunden werden kann. Wir benutzten Subversion für die Dokumentenverwaltung und um unseren Programmcode online zu sichern.

## Subversion

Apache Subversion (SVN) ist eine freie Software zur Versionsverwaltung von Dateien und Verzeichnissen.

Abbildung 3 - Subversion Logo

Die Versionierung erfolgt in einem zentralen Projektarchiv (engl. repository) in Form einer einfachen Revisionszählung.

Mittels dem Befehl „Checkout“ werden die Daten auf das lokale Dateisystem übertragen und könne dort, ohne Datenbankverbindung geändert werden. Damit die Änderungen auf dem Server wirksam werden, muss man diese Änderungen mittels Commit bestätigen. Vor einem Commit empfiehlt es sich, ein Update durchzuführen, dies bringt die Dateien auf den letzten akutellen Stand. Mithilfe dieser Technik ist es möglich, auf ältere Zustände des Projektes zurückzugreifen.

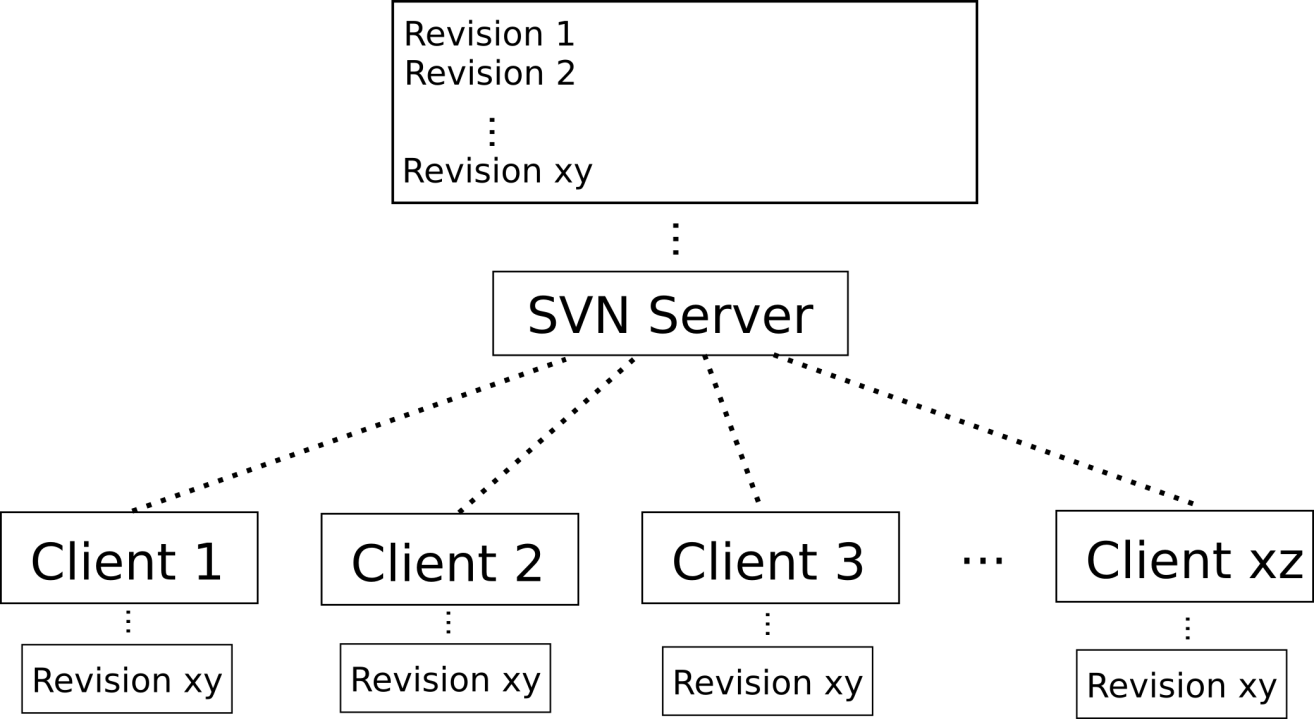


Abbildung 4 - Beschreibt das Zusammenspiel zwischen einem Subversion Server und mehreren Clients.

# 

# Funktionales Spezifikationsdokument

# Allgemein

## Zusammenfassung

Das Projekt für die Firma Aberger Software GmbH soll zum Erstellen von Postkarten dienen und diese anschließend hochzuladen. Der Bestellvorgang ist dabei nicht im Projektumfang.

Beim Start werden eine vorhandene Vorlage ausgewählt, die vorinstalliert sind bzw. weitere heruntergeladen werden können.  
Anschließend nimmt der Benutzer ein Bild von seinen eigenen Bildern und kann dieses dann in der Vorlage bearbeiten.

## Voraussetzungen

* Ein Webserver mit PHP-Unterstützung
* Vorhandene Bilder auf dem Gerät

## Abhängigkeiten

* Die Android App wird in den Android Market gestellt.

# Prozess Spezifikation

Beim ersten Start werden dem Benutzer bereits mitgelieferte Vorlagen angezeigt, von denen er sich eines davon auswählen kann. Es stehen ihm aber weitere Vorlagen bzw. Bearbeitungsmöglichkeiten zur Verfügung.

1. Es können weitere Vorlagen aus dem Internet heruntergeladen werden. Dies ist natürlich kostenlos möglich. Die auf dem Server vorgefertigt liegenden ZIP-Dateien werden in eine Repository-Datei gespeichert, da eine Suche auf dem Server zu aufwendig wäre bzw. diesen sehr auslasten würde. Erst aus diesem Repository, der den Pfad zu den Archiven enthält, kann der Download-Vorgang gestartet werden.
2. Der Benutzer kann, nachdem er sich für eine Vorlage entschieden hat, ein Bild aus seiner Galerie wählen. Anschließend kann er dieses Bild beliebig verschieben, drehen und/oder vergrößern bzw. verkleinern. Dabei werden immer die vorgefertigten Bilder als SVG-Datei spezifiziert.
3. Nachdem der Benutzer mit dem Bearbeiten fertig ist, kann er seine Datei auf den Server hochladen. Dabei muss er zuvor seine persönlichen Informationen angeben, um ihn zu identifizieren. Zahlungsinformationen sind dabei noch nicht nötig.

**Folgende TEXTE sind nur Beispiele von einem anderen Projekt, dies wird noch bearbeitet.**

### Template-Auswahl bzw. Erweiterung

Zuerst muss ein User oder eine Userin eingeloggt werden um überhaupt Zugriff auf Daten in der Datenbank zu haben. Das soll sowohl bei der Android App als auch auf der Website einmal passieren. Dann werden die Login-Daten gespeichert um jedes weiter Mal direkt starten zu können

Fehler: Es konnten keine passenden Logins gefunden werden, dann wird erneut gefragt

Danach werden drei Optionen angezeigt die ausgewählt werden können

### SVG-Dateien bearbeiten

Gesucht werden kann hier nach Artikelnummer oder Beschreibung, dann werden alle Artikel angezeigt die auf die Eingabedaten passen. Es kann auf einen der Artikel geklickt werden um die Detailansicht für diesen aufzurufen (Kapitel 2.1.5)

Fehler: Es konnten keine Daten gefunden werden, dann werden auch keine Daten angezeigt und es müssen neue Eingabedaten angegeben werden

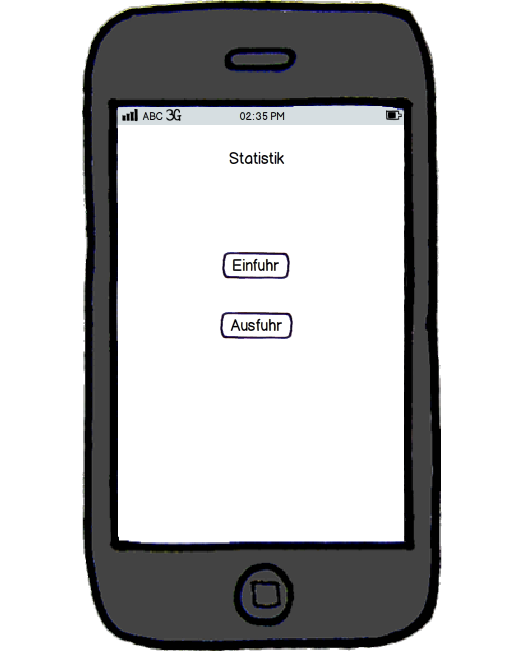
### Fertiges Bild hochladen

Die Bezugsnummer des Exportbegleitdokuments wird eingegeben und gesucht. Alle passenden Einträge in der Datenbank werden angezeigt und es kann auf einen dieser Einträge geklickt werden um den Barcode dieses Dokuments anzuzeigen

Fehler: Es wird kein Begleitdokument gefunden, dann muss neu gesucht werden

Fehler: Der Download und die Anzeige des Barcodes wurden nicht abgeschlossen, dann wird ein neuer Versuch gestartet oder ein Fehler angezeigt

### Statistiken anzeigen



Hier kann zwischen Einfuhr(Kapitel 2.1.7) und Ausfuhrstatistik(Kapitel 2.1.6) gewählt werden

## Kostenplanung

### Arbeitsleistung

In der folgenden Tabelle wurden alle Personen berücksichtigt, die an unserer Diplomarbeit Mitgewirkt und zu ihrem gelingen beigetragen haben.

|  |  |  |  |
| --- | --- | --- | --- |
| Kostenträger | Stundenanzahl | Stundensatz | Betrag |
| Markus Heilmann | 45 h | 20,00 € / h | 900,00 € |
| Manuel Baumgartner | 43 h | 20,00 € / h | 860,00 € |
| DI Christian Aberger | 4 h | 80,00 € / h | 320,00 € |
| Martin Brander | 3 h | 75,00 € / h | 225,00 € |
| Laptops | 48 h | 2,00 € / h | 96,00 € |
| Summe | | | **2.401,00 €** |

Tabelle 1 – Arbeitsleistung

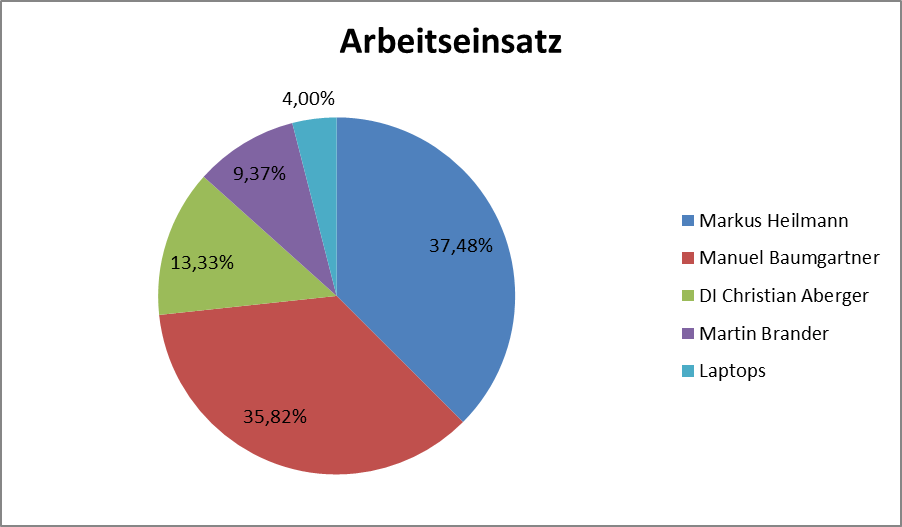


Abbildung 5 - Diagramm Arbeitsleistung

### Lizenzkosten

Die nachstehenden Beträge wurden in der Realität nicht beglichen, da es sich entweder um Lizenzen handelte. Die man bereits besaß oder zur Verfügung gestellt wurden. Diese Zahlen wurden aus Webshops entnommen.

|  |  |  |
| --- | --- | --- |
| Kostenträger | Anzahl | Betrag |
| Windows 7 Home Premium | 2 | 280,00 € |
| Microsoft Office Professional 2010 | 2 | 1.080,00 € |
| Adobe Photoshop CS 6 | 2 | 1.900,00 € |
| Summe | | **3.260,00 €** |

Tabelle 2 – Lizenzkosten

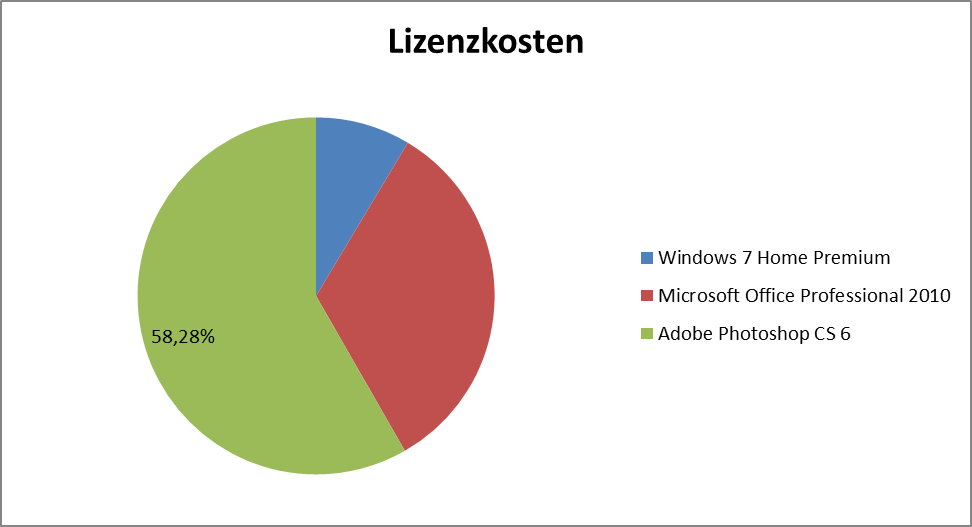


Abbildung 6 - Diagramm Lizenzkosten

### Hardwarekosten

Hierbei werden die anfallende Kosten der Hardware aufgezeigt.

|  |  |  |
| --- | --- | --- |
| Kostenträger | Anzahl | Betrag |
| Acer Tablet A500 | 1 | 350,00 € |
| Samsung Galaxy S3 | 1 | 400,00 € |
| Laptop | 2 | 1.150,00 € |
| Summe | | **1.900,00 €** |

Tabelle 3 – Hardwarekosten

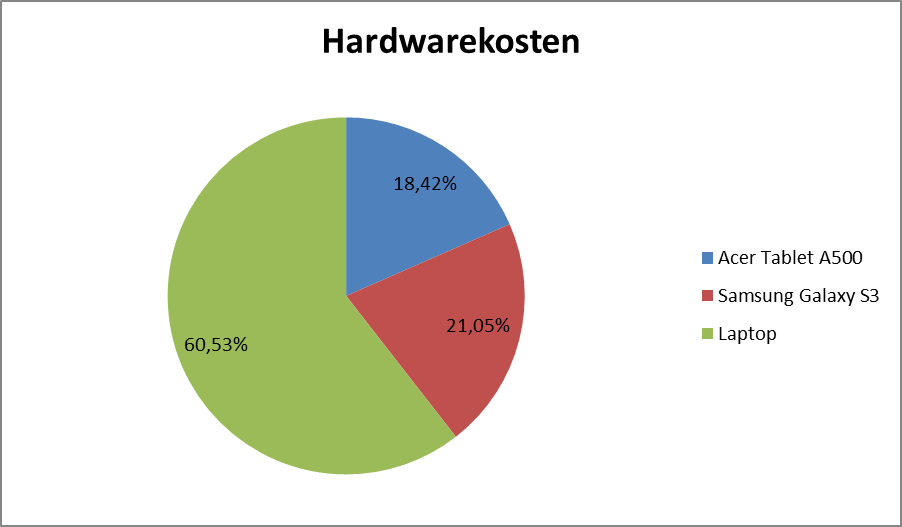


Abbildung 7 - Diagramm Hardwarekosten

### Gesamtkosten

Hier werden noch einmal die Kosten der Arbeitsleistung, Lizenz und Hardware zusammengefasst.

|  |  |
| --- | --- |
| Kostenträger | Betrag |
| Arbeitsleistung | 2.401,00 € |
| Lizenzkosten | 3.260,00 € |
| Hardwarekosten | 1.900,00 € |
| Summe | **7.561,00 €** |

Tabelle 4 – Gesamtkosten

Abbildung 8 - Diagramm Gesamtkosten

# Umsetzung

## **Eclipse einrichten**

Um Android Applikationen zu entwickeln mussten wir zuerst unsere Entwicklungsumgebung Eclipse einrichten.

# Verwendete Softwareschnittstellen

## **SVG**

SVG steht für die Abkürzung Scaleable Vector Graphic und ist ein bearbeitbares XML-File, welches mit einem Parser als Bild dargestellt wird. Dabei gibt es einige Vorteile, was die Qualität und den Speicherbedarf betrifft. Der Nachteil liegt aber darin, dass die Verarbeitung aufwendiger ist und daher mehr Zeit in Anspruch nimmt.

SVG-Images haben eine große Palette zur Anwendung. Beispielsweise werden damit einfache Darstellungen wie Spielkarten usw. realisiert, also bereits genormte Piktogramme.

Vorteile:

* leicht skalierbar
* Zoom ohne Qualitätsverlust
* wenig Speicherbedarf

Nachteile:

* höherer Rechenaufwand beim Parsen (im Vergleich zu PNG)
* höherer Zeitaufwand zur Darstellung

## XML

XML steht für Extensible Markup Language was so viel wie erweiterbare Anzeigesprache bedeutet.

Sie wird verwendet, um Daten strukturiert in einem Textfile darzustellen. Es wird zur digitalen Datenübertragung über das Internet verwendet und ist Plattform und Implementierungsunabhängig.

Die grundlegende Struktur wurde von der World Wide Web Consortium (W3C) als Norm festgelegt. Diese definiert strukturelle sowie auch inhaltliche Einschränkungen. Über die Anzeigesprache können auch über Metasprachen Limitationen (Schemen) festgelegt werden (Bsp. DTD, XML-Schema).

Anwendungsfälle für diese erweiterbare Anzeigesprache sind:

* HTML, XHTML
* SVG (Skalierbare Vektor-Grafik)
* RSS
* XAML (WPF-Spezifikationen in für C#)
* Layouts, Resourcen in Android (eigene Kreationen)

### Aufbau

#### Wohlgeformtheit (well-formed)

Der grundlegende Aufbau besteht aus ein paar wenigen Regeln.

* Ein Root-Element
* Jedes Element besitzt einen Anfangs und End-Anzeiger (evtl. in sich geschlossen).
* Alle Child-Elemente müssen geschlossen sein bevor das Parent-Element geschlossen wird (ebenentreu-paarig verschachtelt)
* Keine gleichnamigen Attribute im selben Element

|  |
| --- |
| **<?xml**version="1.0"encoding="UTF-8"standalone="yes"**?>**  <School name=”HTBLA Perg”>  <Class name=”5BHDVK” name=”5BHD”> END  </School>  <School name=”HTBLA Leonding”/> |

#### Physikalischer Aufbau

* Entitäten
* Deklaration (<?xml version=… encoding=… standaone=…?>)
* Dient zur Spezifikation bzgl. Version, Codierung und Schemas.
* Dokumententyp-Spezifikation von Entitäten und des logischen Aufbaus

#### Logischer Aufbau

* Elemente
* Attribute
* Verarbeitungsanweisungen: Daten an eine gewisse XML-Anwendung weiterleiten
  + <?Name Datas?>
* Kommentare
  + <!-- comment -->
* Text: Normal oder CDATA-Abschnitt
  + <![CDATA[Text]]>

#### Wichtige Begriffe

* Elemente: Name des XML-Feldes, kann frei gewählt werden (ausgenommen im Schema festgelegt).
  + **Child-Element:** Ein Element kann mehrere verschachtelte Elemente besitzen, diese werden als Kinder bezeichnet.
  + **Parent-Element:** Ein verschachteltes Element besitzt einen Elternteil, das eine Ebene über diesem Element steht.
  + **Silbling-Element:** Hat ein Parent-Element mehrere Kinder, so kann ein Child direkt auf das folgende Geschwister-Element zugreifen.
* Parser: Dies ist ein meist vordefiniertes Programm, das XML-Dateien ausliest, erzeugt oder bearbeitet. Weiter kann dieser auch die Datei auf ihre Gültigkeit prüfen.
* Validität: Die Gültigkeit einer XML-Datei kann mit dem Parser mithilfe des bereits vorhandenen Schemas überprüft werden. Die Validität gibt an, ob die Datei gültig ist oder nicht. Ist kein Schema angegeben, so wird der Aufbau überprüft, ob er „well-formed“ ist.

#### Klassifizierung

Nach ihrer Verwendung und Strukturierung lassen sich XML-Dateien in zwei Formen einteilen. Dabei gibt es eine Mischform.

* Dokumentzentriert: Die Datei ist hauptsächlich für das menschliche Verständnis formatiert und macht eine maschinelle Verarbeitung schwieriger.
* Datenzentriert: Starke Strukturierung, folgt einem Schema. Es ist schwieriger zum Verstehen, jedoch leichter zu verarbeiten. Elemente haben entweder Attribute oder Textinhalt.
* Semistrukturiert: Mischform, weniger dokumentzentriert und datenzentriert.

#### Verarbeitung (Parsen)

Die grundlegende Verarbeitung ergibt sich aus drei Kriterien:

* Sequentielle Abfolge oder wahlfrei
* Push oder Pull Verarbeitung
  + Push: Ablauf übernimmt der Parser selbst.
  + Pull: Ablauf wird über den Programmcode gesteuert, der vom Parser aufgerufen wird.
* Baumstrukturmanagement: hierarchisch oder verschachtelt

#### Parser

Von Programmierumgebungen werden vordefinierte APIs zur Verfügung gestellt. Die drei Modelle für diese sind:

* DOM
* SAX
* Pull-API

#### DOM

|  |
| --- |
| DocumentBuilderFactorybuilderFactory =  DocumentBuilderFactory.newInstance();  DocumentBuilder builder = null;  try {  builder = builderFactory.newDocumentBuilder();  Document document =  builder.parse(newFileInputStream(“filename”);  } catch (Exception e)  {  System.err.println(e.getMessage());  }  Element rootElement = document.getDocumentElement();  NodeList nodes = element.getChildNodes();  for(int i=0; i<noder.getLength(); i++) {  Node node = nodes.item(i);  Element child = (Element) node;  Strintattrubite = child.getAttribute(“name”);  elem.getTextContent() //Text between the start tag and end tag  } |

Das XML-Dokument kann wahlfrei durchlaufen werden. Der Zugriff erfolgt mithilfe einer Baumstruktur. Es sind auch Änderungen in der Struktur bzw. der Datei möglich. Dabei kann mit dem Start bereits die Datei mittels eines Schemas auf seine Validität überprüft werden.

|  |
| --- |
| Schema schema = null;  Try {  String language = XmlConstants.W3C\_XML\_SCHEMA\_NS\_URI;  SchemaFactoryfactory = SchemaFactory.newInstance(language);  schema = factory.newSchema(new File(“filename”));  } catch (Exception e) {  System.err.println(e.getMessage());  }  Validator validation = schema.netValidator();  validation.validate(newDOMSource(“document-name”)); //returns true or false |

#### SAX

Das XML-Dokument wird hier sequentiell durchlaufen. Die API ruft dabei Rückruffunktionen (callback functions) auf, die vom Unterprogramm selbst definiert werden können und so das Dokument auswerten.

|  |
| --- |
| SAXParseFactory factory = SAXParserFactory.newInstance();  SAXParsersaxParser = factory.newSAXParser();  DefaultHandler handler = new DefaultHandler() {  publicvoid characters(char[] ch, int start, int length) throws SAXException {  Stringseminararbeit = newString(ch, start, length);  }  publicvoidstartElement(String name, AttributeListatts) throws SAXException {  if (name.equals("name")) //do something  publicvoidendDocument() throws SAXException {  System.out.println(“End of document“);  }  };  saxParser.parse(„file“, handler); |

#### Pull-API

Daten werden sequentiell, Verarbeitung wird ereignisbasiert und mit Iterator durchgeführt. Die Ablaufkontrolle liegt hierbei nicht beim Parser, sondern beim Programm.

#### XML-Schemas

Diese werden definiert, um gewisse Elementnamen bzw. den Aufbau von definierten XML-Dateien einzuschränken bzw. auf deren Validität zu überprüfen.

#### Dokumenttypdefinition (DTD)

Beschreibt die Struktur und Grammatik von dem Dokument und wurde mit XML standardisiert. Diese Methode ist jedoch veraltet, da es nur für die menschliche Lesbarkeit gedacht war und nicht für den Austausch mit anderen Computern. Als Text-Elemente können nur Zeichenketten gewählt werden, so können Zahlenwerte vorkommen, aber in Gemeinschaft mit anderen Zeichen.

#### XSD (XML-Schema)

Hierbei kann die komplette Struktur mit Elementnamen, Root-Element, Sequenzen, Datentypen, Attribute definiert werden. Es ist die meist verwendete Überprüfungsmethode. Mehrere XML-Schemas können mithilfe eines Importes hinzugefügt werden.

#### Beispiel mit Schema

XML-Datei:

|  |
| --- |
| <shiporderorderid="889923"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:noNamespaceSchemaLocation="shiporder.xsd">  <orderperson>John Smith</orderperson>  <shipto>  <name>Ola Nordmann</name>  <address>Langgt 23</address>  <city>4000 Stavanger</city>  <country>Norway</country>  <shipto>  <item>  <title>Empire Burlesque</title>  <note>Special Edition</note>  <quantity>1</quantity>  <price>10.90</price>  </item>  <item>  <title>Hide your heart</title>  <quantity>1</quantity>  <price>9.90</price>  </item>  </shiporder> |

Schema:

|  |
| --- |
| <?xml version="1.0" encoding="ISO-8859-1" ?>  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  <xs:element name="shiporder">  <xs:complexType>  <xs:sequence>  <xs:element name="orderperson" type="xs:string"/>  <xs:element name="shipto">  <xs:complexType>  <xs:sequence>  <xs:element name="name" type="xs:string"/>  <xs:element name="address" type="xs:string"/>  <xs:element name="city" type="xs:string"/>  <xs:element name="country" type="xs:string"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="item" maxOccurs="unbounded">  <xs:complexType>  <xs:sequence>  <xs:element name="title" type="xs:string"/>  <xs:element name="note" type="xs:string" minOccurs="0"/>  <xs:element name="quantity" type="xs:positiveInteger"/>  <xs:element name="price" type="xs:decimal"/>  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:sequence>  <xs:attribute name="orderid" type="xs:string" use="required"/>  </xs:complexType>  </xs:element>  </xs:schema> |

#### Vergleich mit CSV

Eine einfachere Methode wäre es, wenn man diese Daten als CSV-Datei darstellt. Es wäre weniger speicheraufwendig und einfacher auszulesen. Nur kann hier keine Verifikation durchgeführt werden und außerdem kann die Codierung bei einem anderen Rechner anders sein, so dass Äs, Ös oder andere Sonderzeichen als □ oder als andere Zeichen interpretiert werden. Außerdem sind keine Verschachtelungen und Hierarchien möglich.

## ZIP

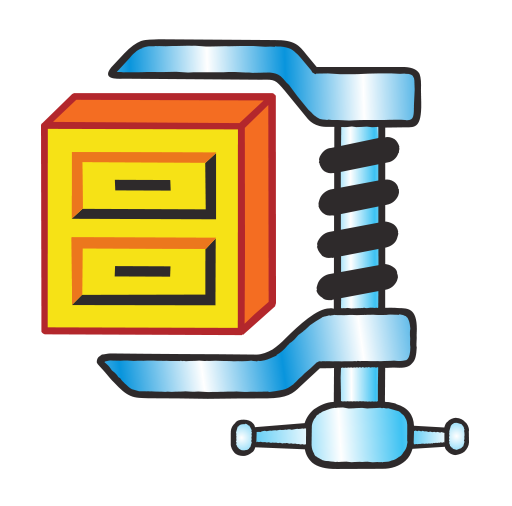
ZIP ist ein Format, um Dateien in ein File zusammenzufassen und diese gesamt zu komprimieren. Dabei können auch Ordner und mehrere Unterverzeichnisse erstellt werden.

Abbildung 9 - ZIP Logo

Einerseits ist es eine gute Gelegenheit, um viele Dateien auf einmal zu transportieren bzw. können mit einer hohen Komprimierungsrate auch größere Dateien versendet werden. Es wird aber auch dazu verwendet, um weniger Speicherplatz auf einem Datenträger zu verbrauchen.

Das Arbeiten mit ZIP-Archiven bringt jedoch auch ein paar wesentliche Nachteile. Der größte Nachteil ist, dass beim Öffnen der Datei diese zuerst diese File auf einem temporären Ort zwischengespeichert werden muss und anschließend erst geöffnet werden kann. Dies hat eine höhere Rechenleistung und einen höhere Zeitaufwand zur Folge.

Es ist aber möglich sämtliche oder selektive Dateien bereits vor dem Öffnen exportiert werden können und anschließend erst geöffnet werden. Hierbei kann man die Vorteile bei der Übertragung nutzen und die Datei lokal ohne Archivierung behandeln.

Bereits jeder Betriebssystem unterstützt das lesen und schreiben von ZIP-Files.

Dabei gibt es noch weitere Freie Programme, welche ZIP-Files in eine ausführbare Datei umwandeln. Dies ist z. B. ein Feature von Winrar. Die sogenannten SFX-Dateien müssen beim Entpacken nur mehr ausgeführt werden, wobei direkt ein Pfad angegeben werden kann.

Viele Installations-Anwendungen sind in ähnlicher Weise realisiert, jedoch werden meist eigene Programme oder andere Standards wie den Windows Installer.

# Ziele und Anforderungen

Die verwendeten Technologien von Android mit der Eclipse IDE werden zum Erstellen von SVG-Files verwendet, die an einen Server (mittels http) einfach übertragen werden. Dazu benötigt es viele Templates für die Vorderseite, sowie ein einfaches Template für die Rückseite.

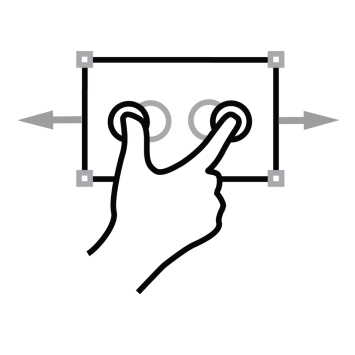
Das Template auf der Vorderseite ist die eigentliche Gestaltung durch den User. Er wählt aus verschiedenen mitgelieferten Vorlagen oder kann sich diese mithilfe eines Repository-Files vom Server herunterladen. Anschließend kann der User die Postkarte mit einem eigenen Bild frei gestalten. Er hat somit die Möglichkeit mit der 2-Finger-Zoom Geste die Größe bzw. die Position des eingefügten Bildes zu verändern. Darüber hinaus kann der Benutzer eine Rotation festlegen.

Abbildung 10 - Zwei-Finger-Zoom

Die Rückseite der Postkarte enthält zwar weniger, jedoch die wichtigsten Informationen, nämlich den Empfänger. Diese kann mithilfe des Adressbuches aus dem Telefon- oder SIM-Speicher direkt eingelesen und nachbearbeitet oder selbst eingegeben werden.

# Gestik Steuerung

In der Applikation gibt es einiges, was ab Bild verändert werden kann. Der User kann die Größe des eingefügten Bildes und seine Position im Rahmen bestimmen. Dies wird durch gewisse onTouchEvent(); Ereignisse (Events) realisiert. Einige dieser Gesten haben sich bereits am Smartphone-Markt durchgesetzt und werden auch von den meisten Benutzern als Standards angesehen. In unserer App werden zwei Zwei-Finger Gesten und eine Ein-Finger Geste verwendet.

## Zoom-Gestik

Eine der bekanntesten Zwei-Finger Gesten ist die Zoom-Funktion.  
Der Zoom-Level wird dabei mit der Veränderung des Abstandes zweier Berührungspunkte bestimmt. In Anwendungsfall sieht das folgendermaßen aus.

Point pa1 = rem[0];

Point pa2 = rem[1];

Point pb1 = **new**Point((**int**)e.getX(0), (**int**)e.getY(0));

Point pb2 = **new**Point((**int**)e.getX(1), (**int**)e.getY(1));

**//Kalkulation der Distanz zwischen den zwei Berührungspunkten beim letzten Event**

**float**distanceA =   
(**float**) Math.*sqrt*( Math.*pow*((pa2.x - pa1.x),2) + Math.*pow*((pa2.y - pa1.y),2 ));

**//Kalkulation der Distanz zwischen den zwei Berührungspunkten des aktuellen Events**

**float** distanceB =   
(**float**) Math.*sqrt*( Math.*pow*((pb2.x - pb1.x),2) + Math.*pow*((pb2.y - pb1.y),2 ));

**//Errechnen der Differenz der zwei Distanzen und Erhöhen oder Vermindern der Zoom-Variable**

**if**(distanceB >distanceA)

{

**//mit einer Division der Differenz kann die Zoom-Geschwindigkeit reguliert werden**

**float** diff = (**float**) ((distanceB - distanceA) / 400);

**if**(zoom + diff <= 6) {

zoom += diff;

}

} **else** {

**float** diff = (**float**) ((distanceA - distanceB)) / 400;

**if**(zoom - diff >= 0.5) {

zoom -= diff;

}

}**//Die aktuellen Punkte warden für den nächsten Event-Aufruf gespeichert**

rem[0] = **new** Point((**int**)e.getX(0), (**int**)e.getY(0));

rem[1] = **new** Point((**int**)e.getX(1), (**int**)e.getY(1));

Im Anschluss wird dem Bitmap mithilfe der Matrix-Komponente die Skalierung, also der Zoom-Faktor übergeben.

## Rotations-Gestik

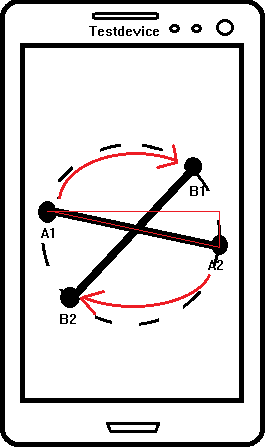
Eine größere Herausforderung ist die Bestimmung der Rotation. Die bekannte Geste hierfür ist, dass der Benutzer zwei Finger im Kreis bewegt und der Winkel von einem der absolute Winkel (relativ zur Bildschirmvertikale) bzw. der Unterschied in der Bewegung der Kreisbewegung die Rotationsänderung bestimmt.

Abbildung 11 – Zwei-Finger-Rotation

Es wird hierbei der Winkel von A1 zur Bildschirmvertikale besimmt, wobei auch A2 auch als herangezogen werden kann. Der Winkel selbst wird mit einem gedachten Dreieck zwischen den Punkten A1 und A2 berechnet.

Genau genommen wird mit der Formel  
 Wobei c der gesamte Abstand und b der Abstand nur in x-Richtung ist.

Da aber der Winkel für den gesamten Kreis errechnet werden muss, wird anschließend zwischen dem Ausgang des Winkels (links/rechts von x, oben/unten von y) unterschieden.

|  |  |  |  |
| --- | --- | --- | --- |
| A2(x) >A1(x) A2(y) < A1(y) | A2(x) > A1(x) A2(y) >= A1(y) | A2(x) <= A1(x) A2(y) >= A1(y) | A2(x) <= A1(x) A2(y) < A1(y) |
|  |  |  |  |

Zuletzt wird nur der Unterschied herangezogen, um eine neue Bildrotation über die Matrix dem Bitmap zu übergeben. Da die vorherigen Punkte bereits von der Zoom-Gestik gespeichert sind, werden diese wieder herangezogen.

Mithilfe der bereits erstellten Funktion, die den Winkel bestimmt sieht der Aufruf folgendermaßen aus:

**float** rot1 = rotation(rem[0].x, rem[0].y, rem[1].x, rem[1].y);

**float** rot2 = rotation(e.getX(0), e.getY(0), e.getX(1), e.getY(1));

**if**(rot1 > rot2) {

rotate -= (rot1 - rot2);

} **else** {

rotate += (rot2 - rot1);

}

Die eigentliche Funktion, die mit den Parametern der beiden Punkte den Winkel zur Bildschirmvertikale errechnet sieht im Ende so aus:

**private float** rotation(**float** x0, **float** y0, **float** x1, **float** y1) {

**float** rot = 0;

rot = (**float**) Math.acos(Math.abs(x1-x0) / distance(x0,y0,x1,y1));

**//Da der Winkel normalerweise im Bogenmaß (0 - 2π) angegeben wird, muss  
 //anschließend noch eine Umformung auf das Gradmaß (0° – 360°) erfolgen**

rot = (**float**) Math.toDegrees(rot);

**if**(x1 > x0 && y0 > y1) {

rot = 90 - rot;

} **elseif**(x1 > x0 && y0 <= y1) {

rot += 90;

} **elseif**(x1 <= x0 && y0 <= y1 ) {

rot = 260 - rot;

} **elseif**(x1 <= x0 && y0 > y1) {

rot += 260;

}

**return** rot;

}

## Transformations Matrix

Dem Bitmap-File wird beim Zeichnen eine Transformations-Matrix mitgegeben, welche die Skalierung, Verschiebung und Rotation auf das zu zeichnende Bild anwendet.

Die Angaben zu diesen Werten erfolgt in einer Float-Gleitkommazahl, da besonders Werte zwischen ganzen Zahlen von großer Bedeutung sind.

Die Matrix selbst besteht aus einer einfachen Matrix-Klasse, welche normal initialisiert wird und vom Typen android.graphics.Matrix ist.

**import** android.graphics.Matrix;

…

Matrix m = **new** Matrix();

Anschließend können die verschiedenen Operationen auf die Matrix durchgeführt werden. Diese sind sequentiell nacheinander durchführbar, aber auch von einem Bereit generierten Float-Array.

m.postScale(zoom, zoom, startX, startY);

m.postRotate(rotate, startX, startY);

m.postTranslate(x,y);

Die Prozedur **m.postTranslate(float dx, float dy)** kann nur die Parameter für die Verschiebung in X- und Y-Richtung enthalten, wobei die

Prozedur **postScale (float sx, float sy, float px, float py)** die Skalierung für X und Y in den Parametern sx, sy auch der Mittelpunkt für die neue Skalierung in den Parametern px und py enthalten kann.

Gleiches gilt für **public boolean postRotate (float degrees, float px, float py)**

hierbei bestimmen px und py den Mittelpunkt der Rotation und degrees den zu drehenden Winkel in Grad (0° – 360°) an.

Mit der Prozedur **setValues (float[] values)** ist es möglich einen schon vorhandenen Float-Array, der die Transformation-Elemente enthält, für die Matrix-Bildung heranzuziehen. Es ist aber zu beachten, dass der Array eindimensional ist und die Leserichtung zuerst von oben nach unten und dann erst von links nach rechts ist.

Nach der gesamten Matrix-Bildung wird dieses Bitmap mit der Transformation anschießend auf ein gegebenes Canvas gezeichnet.

draw.drawBitmap(bitmap, matrix, paint);

bzw. Mit den Parameter-Typen:

**drawBitmap (Bitmap bitmap, Matrix matrix, Paint paint);**

Die Variable „draw“ ist hierbei ein Objekt der Klasse **android.graphics.Canvas**

## SVG-Matrix

In der Definition von [www.w3.org](http://www.w3.org/) sind Transformationen einfach in einem eindimensionalen Array als Textform nacheinander gespeichert.  
Dies geschieht nach der Form **matrix(<a> <b> <c> <d> <e> <f>).**

Hiermit wäre es mit der Android-Definiton der Transformationmatrizen bereits möglich einen Float-Array (mit Länge 9) anzulegen, jedoch sollte hier die Leserichtung von bben nach unten beachtet werden.

Im konkreten Fall wäre der Float-Array folgendermaßen aufgebaut:

**Float ar = new Float[9];**

**ar[0]=a; ar[1]=b; ar[2]=0; ar[3]=c; ar[4]=d; ar[5]=0; ar[6]=e; ar[7]=f; ar[8]=1;**

Das SVG-Attribut wird davor nach Leerzeichen aufgeteilt und einzeln in Float-Attribute umgewandelt. Um Fehler zu vermeiden, sollte ein Exception-Handling eingebaut werden, da es bei Buchstaben als Transformation zu Problemen kommt.

## Download von Templates

Natürlich gibt es nicht nur die vier Standard-Templates zur Auswahl, sonder noch viele weitere, die man über das Internet herunterladen kann. Der Download soll hierbei schnell und ohne Probleme erfolgen. Der Pfad zum Repository-File ist in den String-Ressourcen unter dem Namen „repository“ gespeichert.

Dieses XML-Repository wird zuerst aufgerufen, um alle zur Verfügung stehenden ZIP-Files zu bekommen. Das Format dieses XML-Files ist wie folgt aufgebaut:

|  |  |
| --- | --- |
|  | <?xml version="1.0" encoding="utf-8"?> |
|  | <svg-repository> |
|  | <folder name="example\_frames" url="example\_frames.zip"/> |
|  | </svg-repository> |

Das Root-Element soll hierbei <svg-repository> sein, wobei der Parser den Namen des Root-Elementes nicht beachtet.  
Wichtiger sind eher die Folder-Tags, da diese sequentiell durchlaufen und deren Name und URL zwischengespeichert und anschließend die definierten ZIP-Files heruntergeladen werden.

Die Angabe der Zieldateien ist dabei immer relativ zum Repository, d. h. diese befinden sich im selben Verzeichnis. Eine absolute Angabe kann aber mit <http://domain.net/file.zip> angegeben werden. Damit soll es ermöglicht werden, dass die Dateien auf mehreren Servern gespeichert werden können.

### Entpacken von ZIP-Dateien

Da für das Herunterladen von Vorlagen (Templates) ZIP-Archive verwendet werden, müssen diese auch anschließend entpackt bzw. der Index zu allen Dateien in ein neues XML-File geschrieben werden. Das Index-File wäre hier eigentlich nicht nötig, beschleunigt aber die Ausführung der Applikation. Es ist nur einmal nötig alle Dateien vom ZIP-Archiv einzutragen.

In Android bzw. Java sind bereits vordefinierte Klassen für den Export vorhanden. Diese sind:

* **java.util.zip.ZipEntry**
* **java.util.zip.ZipFile**
* **java.util.zip.ZipInputStream**

Jede ZIP-Datei besteht aus mehreren Dateien, deshalb werden diese sequentiell, also nacheinander, durchlaufen. Die Klasse **ZipEntry** wird anschließend für den Zugriff auf eine Datei im Archiv verwendet.

1. Der **ZipInputStream** stellt die Verbindung zum Archiv bzw. stellt den Extrahierungs-Prozess zur Verfügung. Dabei stellt der Stream die Funktion **getNextEntry()** zur Verfügung, um sequentiell die Dateien zu durchlaufen. Nun muss unterschieden werden, ob der Eintrag ein Ordner oder eine Datei ist.
2. Bei einem Ordner erstellt man hierbei einen gleichnamigen Ordner auf der SD-Karte oder auf dem internen Speicher.
3. Bei einer Datei wird diese direkt mit dem **FileOutputStream** gespeichert, der die Daten wiederrum vom **ZipInputStream** enthält.
4. Es wurde mit der Methode **getNextEntry()** am Anfang ein Eintrag geöffnet, der jetzt wieder mit **closeEntry()** geschlossen werden muss, um einen neuen öffnen zu können.

Zwischen dieser Prozedur wird ein eigenes File zum Indizieren verwendet. Es müsste sonst jedes Mal die gesamte Ordnerstruktur des Projektes durchlaufen werden, das natürlich viel unnötige Zeit und Ressourcen benötigt. Es kann hierbei ein XML-File oder auch ein normales Text-File erstellt werden. Der Nachteil hierbei ist nur, dass ein erhöhter Speicherbedarf vorherrscht.

Anhang

Quellenverzeichnis

**Android**

http://de.wikipedia.org/wiki/Liste\_von\_Android-Versionen

Zugriff am 08.04.2013

**Eclipse**

http://www.eclipse.org/org/

http://wiki.fernuni-hagen.de/eclipse/index.php/Einige\_Begriffserklärungen

http://developer.android.com/tools/sdk/eclipse-adt.html

Zugriffe am 25.03.2013

**Subversion**

http://de.wikipedia.org/wiki/Apache\_Subversion

Zugriff am 08.04.2013

Abbildungsverzeichnis

[Abbildung 1 - Android Logo 7](#_Toc354505389)

[Abbildung 2 - Eclipse Logo unserer Entwicklungumbebung Indigo 11](#_Toc354505390)

[Abbildung 3 - Subversion Logo 13](file:///I:\Schule\HTBLA_Perg\Diplomarbeit\Dokumente\DiplomarbeitSvgEditor.docx#_Toc354505391)

[Abbildung 4 - Beschreibt das Zusammenspiel zwischen einem Subversion Server und mehreren Clients. 14](#_Toc354505392)

[Abbildung 5 - Diagramm Arbeitsleistung 15](#_Toc354505393)

[Abbildung 6 - Diagramm Lizenzkosten 16](#_Toc354505394)

[Abbildung 7 - Diagramm Hardwarekosten 17](#_Toc354505395)

[Abbildung 8 - Diagramm Gesamtkosten 18](#_Toc354505396)

[Abbildung 9 - ZIP Logo 29](file:///I:\Schule\HTBLA_Perg\Diplomarbeit\Dokumente\DiplomarbeitSvgEditor.docx#_Toc354505397)

[Abbildung 10 - Zwei-Finger-Zoom 31](file:///I:\Schule\HTBLA_Perg\Diplomarbeit\Dokumente\DiplomarbeitSvgEditor.docx#_Toc354505398)

[Abbildung 11 – Zwei-Finger-Rotation 33](file:///I:\Schule\HTBLA_Perg\Diplomarbeit\Dokumente\DiplomarbeitSvgEditor.docx#_Toc354505399)

Tabellenverzeichnis

[Tabelle 1 – Arbeitsleistung 15](#_Toc354504041)

[Tabelle 2 – Lizenzkosten 16](#_Toc354504042)

[Tabelle 3 – Hardwarekosten 17](#_Toc354504043)

[Tabelle 4 – Gesamtkosten 18](#_Toc354504044)

1. Virtual Private Network [↑](#footnote-ref-1)
2. TTS: Text zu Sprache [↑](#footnote-ref-2)
3. Browser mit HTML-5-, Javascript-, Geolocation-, Videounterstützung, … [↑](#footnote-ref-3)
4. Anbindung von anderen Geräten, um die Internetverbindung zu teilen [↑](#footnote-ref-4)
5. Video-Container Format [↑](#footnote-ref-5)
6. Near-Field-Communication: Direkte Datenübertragung (wie Infrarot) [↑](#footnote-ref-6)
7. Schwerkraftsensor [↑](#footnote-ref-7)
8. App für eine sanftere Bedienung [↑](#footnote-ref-8)
9. Docking-Stationen [↑](#footnote-ref-9)
10. Wi-Fi Protected Setup (Verbindungsherstellung mit Hotspot) [↑](#footnote-ref-10)