

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. May 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 these templates from the android device space. In the template the picture can be zoomed, rotated and moved. On the back 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 main goal thesis is the gestures control, such as zooming, rotating and moving.

Eidesstattliche Erklärung

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

*„Mobile 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 half er uns durch sein fachliches Wissen und gab 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 11](#_Toc355111823)

[Diplomanden 11](#_Toc355111824)

[Markus Heilmann 11](#_Toc355111825)

[Manuel Baumgartner 12](#_Toc355111826)

[Auftraggeber 13](#_Toc355111827)

[Projektumfeld 14](#_Toc355111828)

[Betreuungslehrer 14](#_Toc355111829)

[Projektstart 15](#_Toc355111830)

[Projektende 15](#_Toc355111831)

[Unterschriften 15](#_Toc355111832)

[Verwendete Technologien 16](#_Toc355111833)

[Java 16](#_Toc355111834)

[Geschichte 16](#_Toc355111835)

[Aufbau 17](#_Toc355111836)

[Eigenschaften von Java 18](#_Toc355111837)

[Weitere Merkmale 19](#_Toc355111838)

[Warum wir Java verwendeten 19](#_Toc355111839)

[Android 20](#_Toc355111840)

[Versionen für API Entwicklung 21](#_Toc355111841)

[Eclipse 24](#_Toc355111842)

[Feature 24](#_Toc355111843)

[Plug-Ins 25](#_Toc355111844)

[Subversion 26](#_Toc355111845)

[Funktionales Spezifikationsdokument 27](#_Toc355111846)

[Allgemein 27](#_Toc355111847)

[Zusammenfassung 27](#_Toc355111848)

[Voraussetzungen 28](#_Toc355111849)

[Abhängigkeiten 28](#_Toc355111850)

[Prozess Spezifikation 29](#_Toc355111851)

[Template-Auswahl bzw. Erweiterung 29](#_Toc355111852)

[SVG-Dateien bearbeiten 29](#_Toc355111853)

[Fertiges Bild hochladen 30](#_Toc355111854)

[Statistiken anzeigen 30](#_Toc355111855)

[Kostenplanung 31](#_Toc355111856)

[Arbeitsleistung 31](#_Toc355111857)

[Lizenzkosten 32](#_Toc355111858)

[Hardwarekosten 33](#_Toc355111859)

[Gesamtkosten 34](#_Toc355111860)

[Umsetzung 35](#_Toc355111861)

[Umgebung einrichten 35](#_Toc355111862)

[Betriebssystem 35](#_Toc355111863)

[Eclipse 37](#_Toc355111864)

[Android SDK Manager 39](#_Toc355111865)

[Android Grundkonzept 39](#_Toc355111866)

[Komponenten 39](#_Toc355111867)

[Architektur einer Applikation 40](#_Toc355111868)

[Verwendete Softwareschnittstellen 42](#_Toc355111869)

[SVG 42](#_Toc355111870)

[XML 43](#_Toc355111871)

[Aufbau 44](#_Toc355111872)

[ZIP 52](#_Toc355111873)

[Ziele und Anforderungen 54](#_Toc355111874)

[Gestik Steuerung 55](#_Toc355111875)

[Zoom-Gestik 55](#_Toc355111876)

[Rotations-Gestik 56](#_Toc355111877)

[Transformations Matrix 58](#_Toc355111878)

[SVG-Matrix 59](#_Toc355111879)

[Download von Templates 61](#_Toc355111880)

[Entpacken von ZIP-Dateien 62](#_Toc355111881)

# 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 die Firma Aberger Software GmbH.



Abbildung 1 - Aberger Software GmbH Logo

Aberger Software GmbH

|  |  |
| --- | --- |
|  | Softwarepark 37 |
| Hagenberg im Mühlkreis |
| Oberösterreich |
| 4232 Österreich |
|  |



Der Ansprechpartner des Unternehmens war:

Martin Brandner

Projektleiter

E-Mail:  [martin.brandner@aberger.at](mailto:martin.brandner@aberger.at)

Abbildung 2 - Martin Brandner

Tel. 0660 505 68 54

## Projektumfeld

### Betreuungslehrer

Betreuungslehrer der Diplomarbeit „Mobile SVG Editor“ war

DI Christian Aberger

E-Mail:  c.aberger@htl-perg.ac.at

Professor für:

* Angewandte Physik
* Mobile Computing
* Programmieren

Abbildung 3 - DI Christian Aberger

Unser Betreuungslehrer stand uns bei Fragen der Programmierung bei. Weiters stellte er uns die benötigten Templates zum Erstellen der Applikation zur Verfügung.

An dieser Stelle möchten wir uns noch einmal sehr herzlich für die Unterstützung bedanken.

### Projektstart

Das Projekt startete mit der Diplomarbeitsanmeldung am

12.10.2012

### Projektende

Das Projekt endete mit der Diplomarbeitsabgabe am

14.05.2013

### Unterschriften

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| DI Christian Aberger | Martin Brandner |

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

# Verwendete Technologien

## Java

### Geschichte

Java wurde von Patrick Naughton, Mike Sheridan, James Gosling und Bill Joy unter dem Namen „The Green Project“ entwickelt. Die Entwicklung dauerte 18 Monate bis zum Sommer 1992. Die Firma Sun stellte Java im März 1995 erstmals zum Download frei.

Sun wurde am 20. April 2009 von Oracle übernommen und stellt seit dem Java offiziell zur Verfügung.

Heute zählt Java zu den beliebtesten Programmiersprachen weltweit und wird, durch seine Technologie, fast überall eingesetzt.

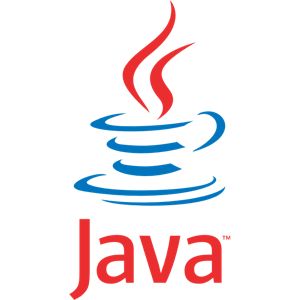


Abbildung 4 - Java Logo

### Aufbau

Java besteht im Grunde aus zwei Teilen, dem Java-Entwicklungswerkzeug JDK (Java Development Kit) und der Java-Laufzeitumgebung JRE (Java Runtime Environment).

Die wichtigsten Bestandteile von JDK sind:

* Java Compiler

Der Java-Compiler übersetzt, den Java-Quellcode in den Bytecode. Da der Compiler in Java geschrieben ist, ist er plattformunabhängig. Das bedeutet der Bytecode sieht auf jedem Betriebssystem gleich aus.

* Java Dokumentationswerkzeug

Das Software-Dokumentationswerkzeug *javadoc* erzeugt aus dem Java-Quellcode eine Programmdokumentation, die zu HTML Hypertext umgewandelt wird. Damit ist es möglich sich jegliche Dokumentation zu einem Programm im Browser anzusehen.

* Java Archiver

Mit Hilfe des Programmes *jar* ist es möglich Java Archive zu erstellen. Diese JAR Dateien sind vergleichbar mit ZIP Dateien. In diesen Archiven sind alle Daten enthalten, die die Java-Anwendung benötigt.

Das JRE beinhaltet die virtuelle Machine (JVM), eine Programmschnittstelle (API) und Programmbibliotheken.

* Die *JVM* ist für die Ausführung der Java Programme verantwortlich.
* Die *API* (Application and Programming Interface) stellt die Standard-Klassen von Java zur Verfügung
  + z.B. „java.lang.String“

### Eigenschaften von Java

Die wichtigsten Eigenschaften von Java sind:

#### Plattformunabhängigkeit

Ein wesentlicher Vorteil von Java ist die Plattformunabhängigkeit oder auch Portabilität genannt. Die Portabilität wird in Java durch genau spezifizierte Datentypen gewährleistet, die für alle Plattformen gleich sind.

Die Java-Umgebung selbst ist einfach auf andere Systeme portierbar, da der Java-Compiler bereits in Java geschrieben ist, so dass nur noch die Laufzeitumgebung portiert werden muss. So ist die Java-Umgebung für alle gängigen Betriebssysteme vorhanden.

#### Objektorientierung

Die Objektorientierung von Java ermöglicht es, Sachverhalte aus der realen Welt als Objekte darzustellen. Ein Objekt besteht aus Methoden und Eigenschaften, die das Objekt beschreiben. In Java ist es möglich diese Objekte und ihre Beziehungen zu modellerieren und so ein besseres Verständnis zu gewährleisten.

#### Erweiterbarkeit

Da Java ein Open-Source Projekt ist, gibt es auch eine eigene Community, OpenJDK genannt. Hier steht es jedem frei, sich den Code anzuschauen, und gegebenenfalls selbst neue Komponenten zu entwickeln.

#### Just-in-Time Kompilierung

Da Java-Programme durch eine virtuelle Maschine ausgeführt werden, ist die Ausführungsgeschwindigkeit im Vergleich zu nativ kompilierten Programmen wesentlich langsamer. Durch die Just-in-time-Kompilierung kann dieser Nachteil ein wenig ausgeglichen werden.

Der Just-in-time Compiler versucht häufig verwendete Programmteile zu identifizieren, zu optimieren und anschließend in Maschinencode zu übersetzen. Der erzeugte Code wird meist zwischengespeichert, um ihn zu einem späteren Zeitpunkt der Programmausführung wiederverwenden zu können.

### Weitere Merkmale

Java stellt eine automatische Speicherbereinigung mittels „garbage collector“ zur Verfügung. Dieser erkennt nicht mehr benötigte Variablen und gibt den Speicherplatz wieder frei.

Weiters besitzt die Technologie die Möglichkeit der Vererbung. Hierbei wird streng zwischen Klassen und Schnittstellen unterschieden. Klassen können von anderen Klassen erben. Mehrfachvererbung ist allerdings nicht möglich.

Schnittstellen können in einer Klasse beliebig viele implementiert werden.

Außerdem besitzt Java eine sehr gute Fehler- und Ausnahmebehandlung. Es ist nicht nur möglich Ausnahmen selbst zu definieren, sondern Java kümmert sich auch standardmäßig um Fehler zur Laufzeit.

### Warum wir Java verwendeten

Da unsere Postkarten App für Android Geräte beschränkt ist, ist es für uns notwendig Java zu verwenden. (siehe Android)

## Android

Android ist ein Betriebssystem für mobile touchscreen Geräte (Handys, Tablets, …) und basiert 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.

Des weiteren werden eine von Sun entwickelte virtuelle Machine und Java Klassenbibliothek für Android zur Verfügung gestellt. Diese Verbindung von Java und Android vereinfacht die Entwicklung von neuen Apps wesentlich.



Abbildung 5 - 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 für API Entwicklung

|  |  |  |  |
| --- | --- | --- | --- |
| Version | API | Veröffentlichung | Neues |
| 1.0 | 1 | 09.2008 | * Google-Apps * Youtube * Synchronisation |
| 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 |
| 2.1 | 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 |
| 2.3.3 | 10 | 02.2011 | * Dual-Core Verbesserung * Single-Core Kompatibilität * NFC-Technik * Bluetooth Verbesserung |
| 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 |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |
| 4.0.4 | 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 |
| 4.2 | 17 | 11.2012 | * Benutzer-Konten (Tablets) * Photo Sphere (Kugelpanoramen) * Tastatur mit Wortvorhersage * Systemweiter Streaming * Daydream (Bildschirmschoner) * Interaktive Navigationsleiste * Verbesserte Sicherheitseinstellungen |

Tabelle 1 - Android Versionen

## 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 6 - Eclipse Logo unserer Entwicklungsumgebung 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 7 - 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 können 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 aktuellen Stand. Mithilfe dieser Technik ist es möglich, auf ältere Zustände des Projektes zurückzugreifen.

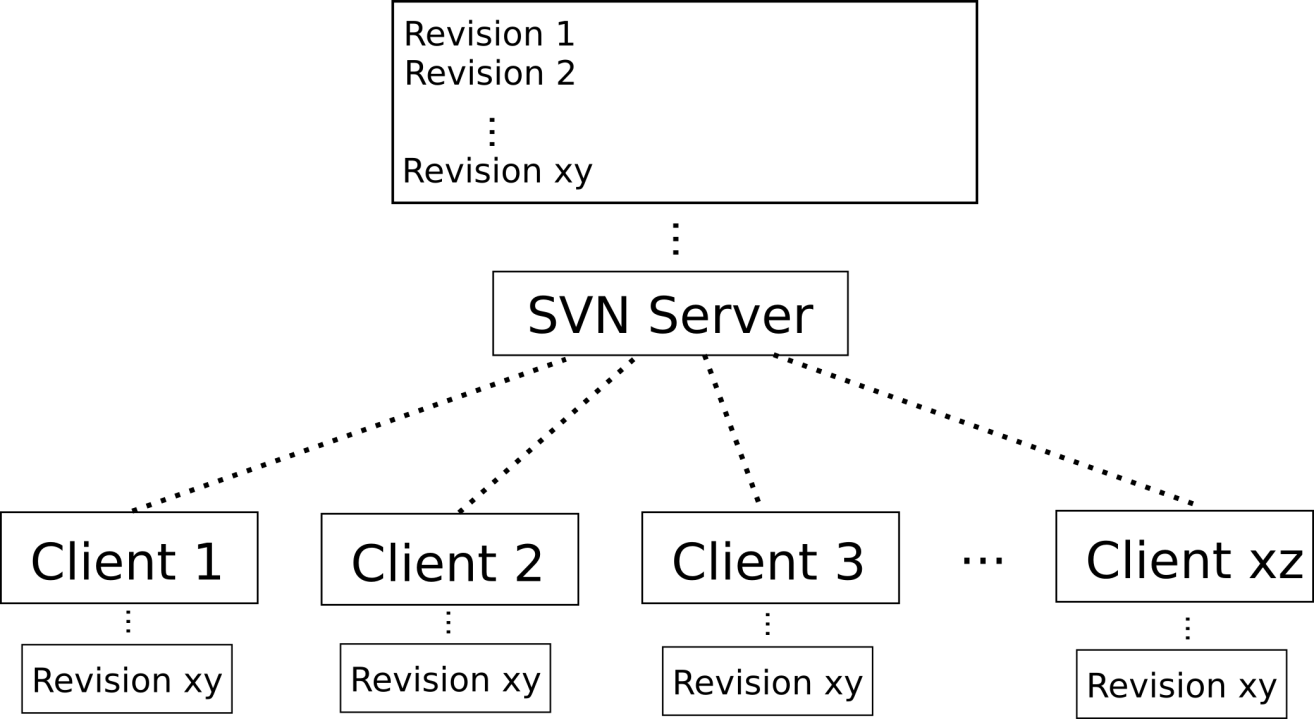


Abbildung 8 - Beschreibt das Zusammenspiel zwischen einem Subversion Server und mehreren Clients.

# 

# 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. (Formulierung Einzahl?)  
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 **Error! Reference source not found.**)

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 **Error! Reference source not found.**) und Ausfuhrstatistik(Kapitel **Error! Reference source not found.**) 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 Brandner | 3 h | 75,00 € / h | 225,00 € |
| Laptops | 48 h | 2,00 € / h | 96,00 € |
| Summe | | | **2.401,00 €** |

Tabelle 2 – Arbeitsleistung

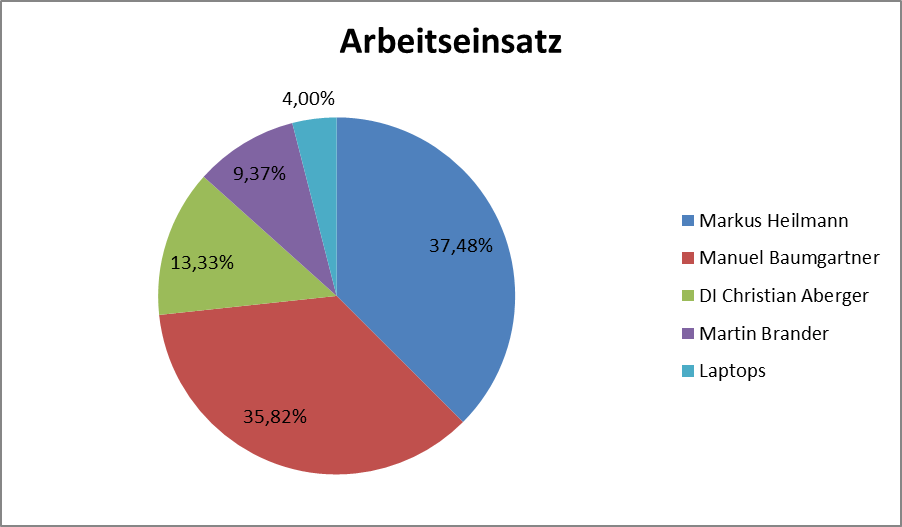


Abbildung 9 - Diagramm Arbeitsleistung

### 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 3 – Lizenzkosten

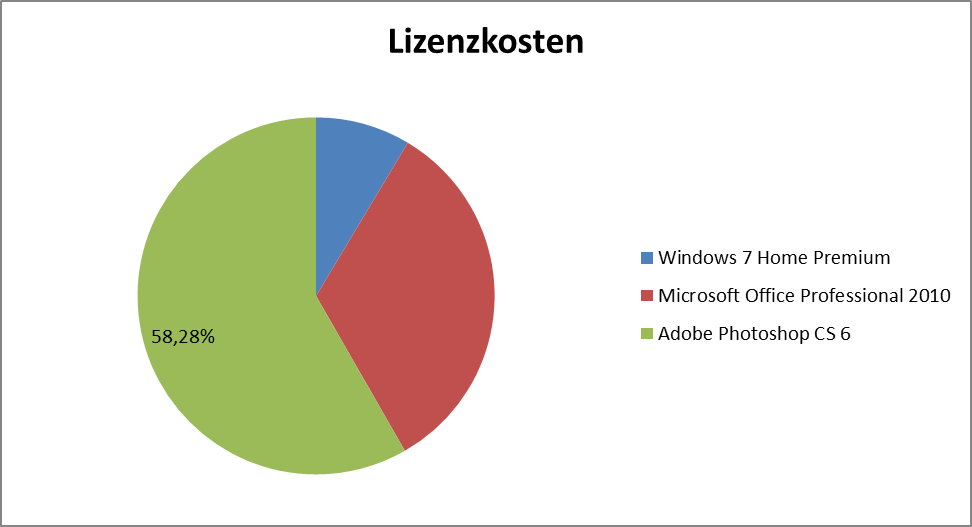


Abbildung 10 - Diagramm Lizenzkosten

### 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 4 – Hardwarekosten

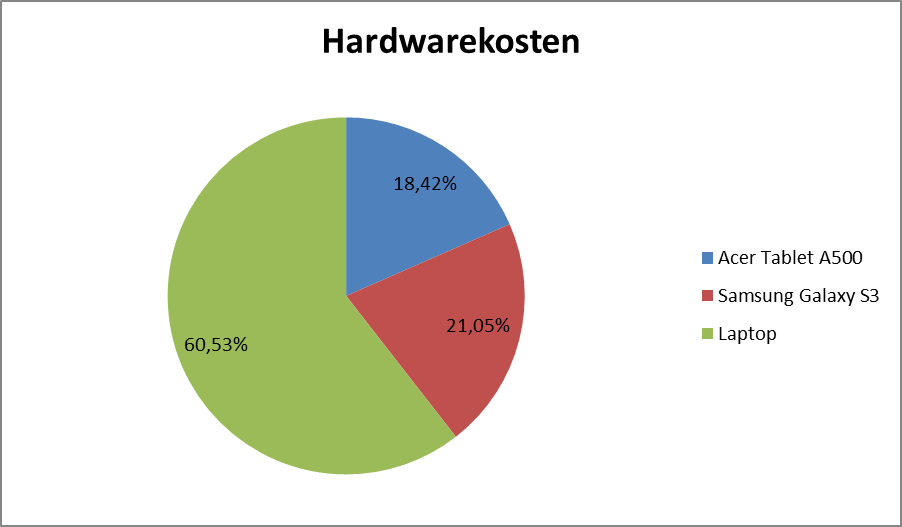


Abbildung 11 - Diagramm Hardwarekosten

### 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 5 – Gesamtkosten

Abbildung 12 - Diagramm Gesamtkosten

# Umsetzung

## Umgebung einrichten

### Betriebssystem

Damit wir optimale Voraussetzungen für unsere Programmiertätigkeit schaffen konnten, mussten wir in einem ersten Schritt das Betriebssystem einrichten.

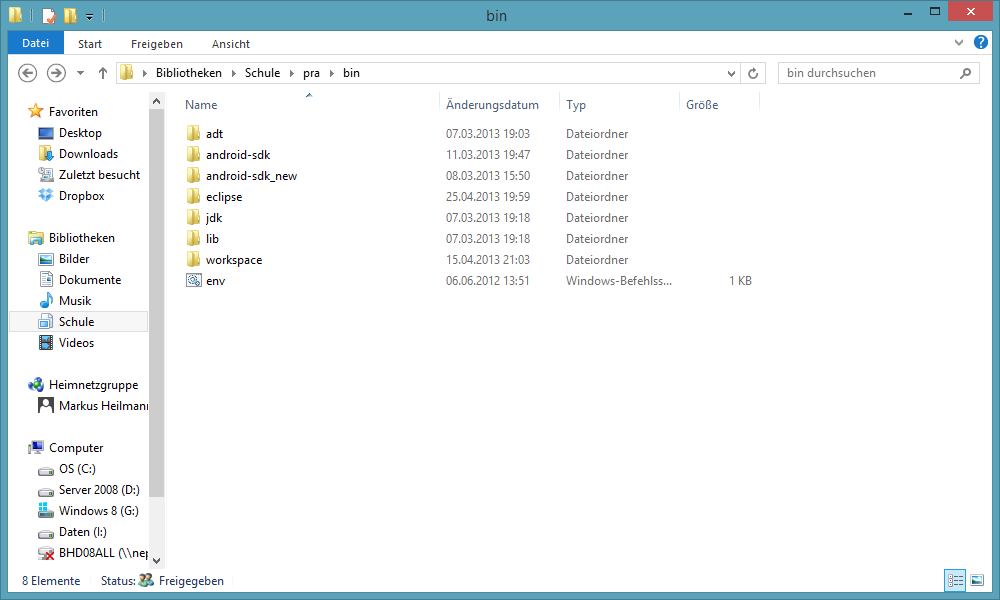


Abbildung 13 - Betriebssystem einrichten

#### adt

Die von für Eclipse benötigten *Android Development Tools* (ADT) befinden sich in diesem Verzeichnis.

#### android-sdk

Hier befindet sich das Android Software Develompent Kit (SDK). Es erlaubt Android Applikationen zu erstellen. Im SDK befinden sich Beispiel Projekte mit Source Code, Development Tools und ein Emulator zum Ausführen der Applikation.

#### eclipse

Auch die Entwicklungsumgebung Eclipse darf in unserer Struktur natürlich nicht fehlen. Für unsere Diplomarbeit verwendeten die Indigo-Version von Eclipse.

#### jdk

Hier steht die aktuelle Java-Version für die Programmierung zur Verfügung.

#### lib

Im lib Verzeichnis werden alle zusätzlich benötigten .jar Dateien abgelegt, die für die Android Programmierung benötigt werden. Durch diese zentrale Speicherung ist es in Eclipse sehr einfach möglich die .jar Dateien einzubinden.

#### workspace

Hier werden alle von Eclipse erstellten Android Projekte gespeichert. Weiters speichert Eclipse auch noch seine Metadaten in diesem Verzeichnis.

#### env.bat

Mit diesem ausführbaren Batch File werden die Umgebungsvariablen für die einzelnen Komponenten definiert. Durch den Befehl *%~dp0%* erhält man den Pfad des aktuellen Verzeichnisses.

Dies hat den Vorteil, dass man den Root Ordner ganz einfach verschieben kann, ohne irgendwelche Pfade oder Einstellungen zu ändern.

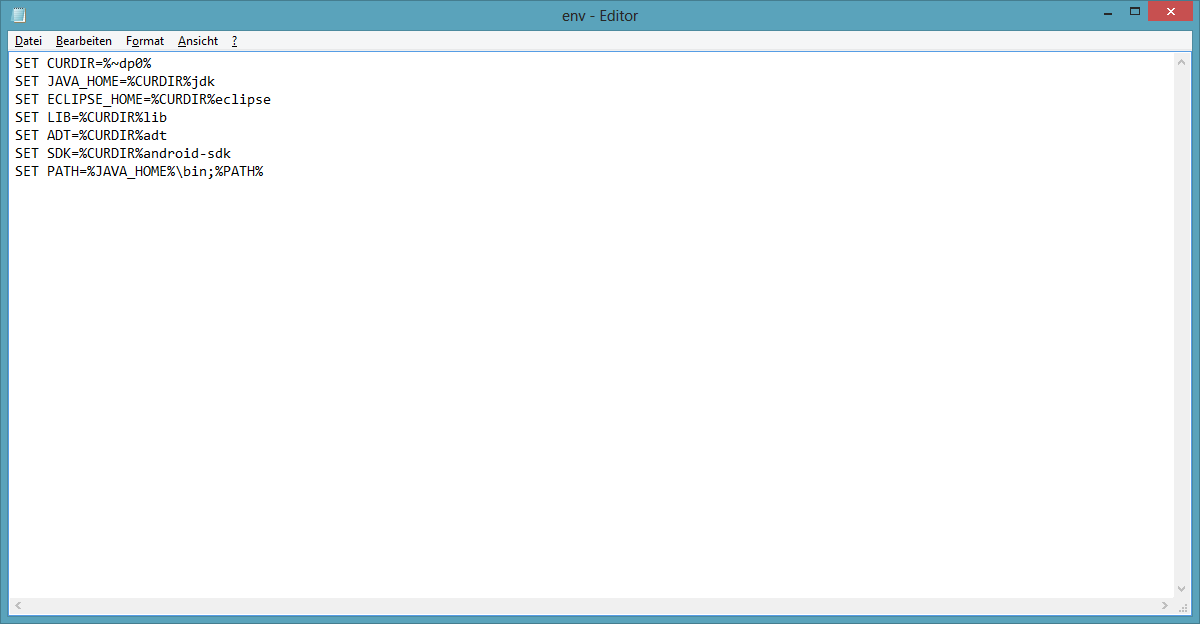


Abbildung 14 - Umgebungsvariablen setzen

### Eclipse

Um Android Applikationen auch wirklich in der Entwicklungsumgebung entwickeln zu können, mussten wir zuerst Eclipse einrichten.

#### Android Development Tools

Mit diesen Tools ist es in Eclipse sehr einfach möglich, neue Android Projekte zu erstellen, die App zu debuggen, oder die erstellte .apk App zu exportieren.

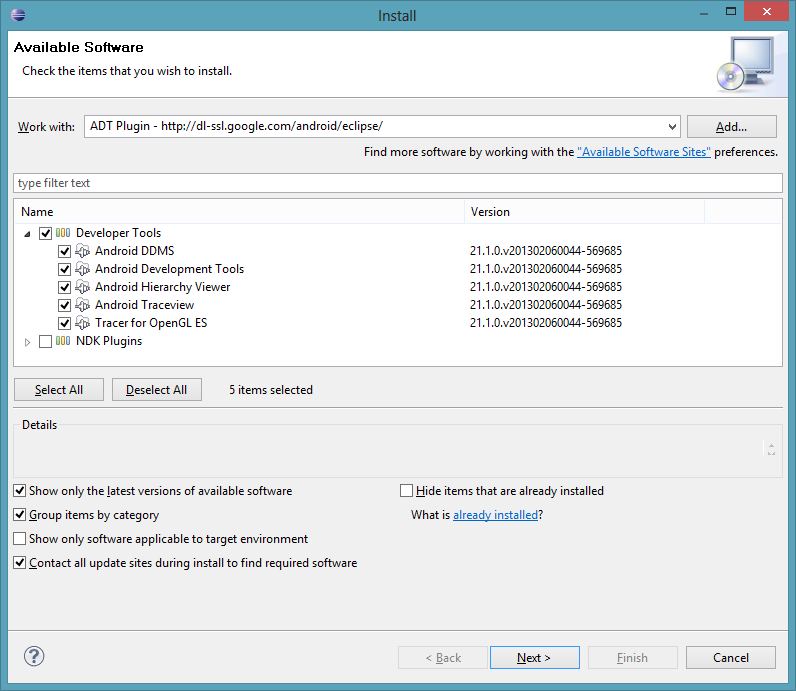


Abbildung 15 - ADT Plugin

#### Subclipse

Mit Subclipse ist es in Eclipse ganz einfach möglich Projekte von einem Repository auszuchecken, das Projekt zu verändern und diese Änderungen wieder im Repository zu speichern (mittels Befehl commit).

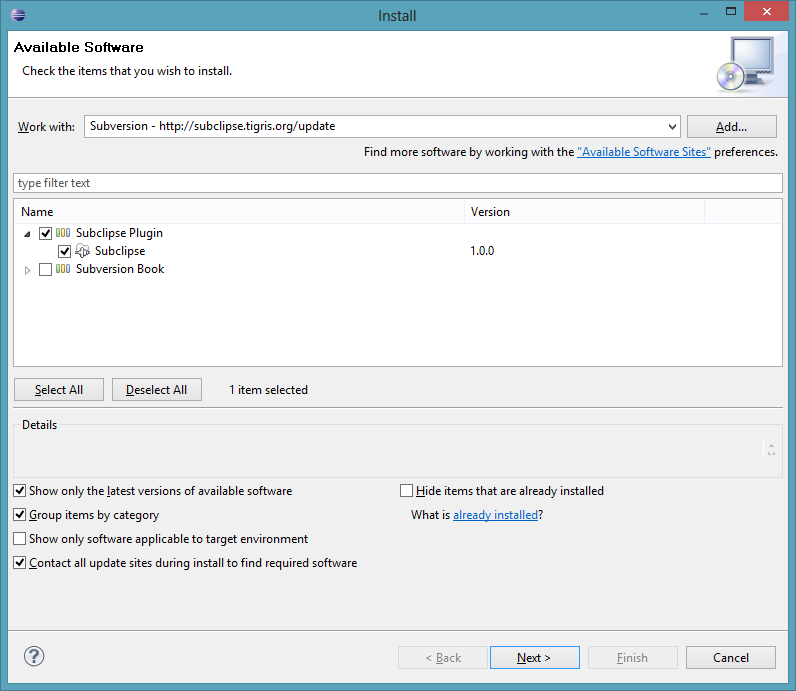
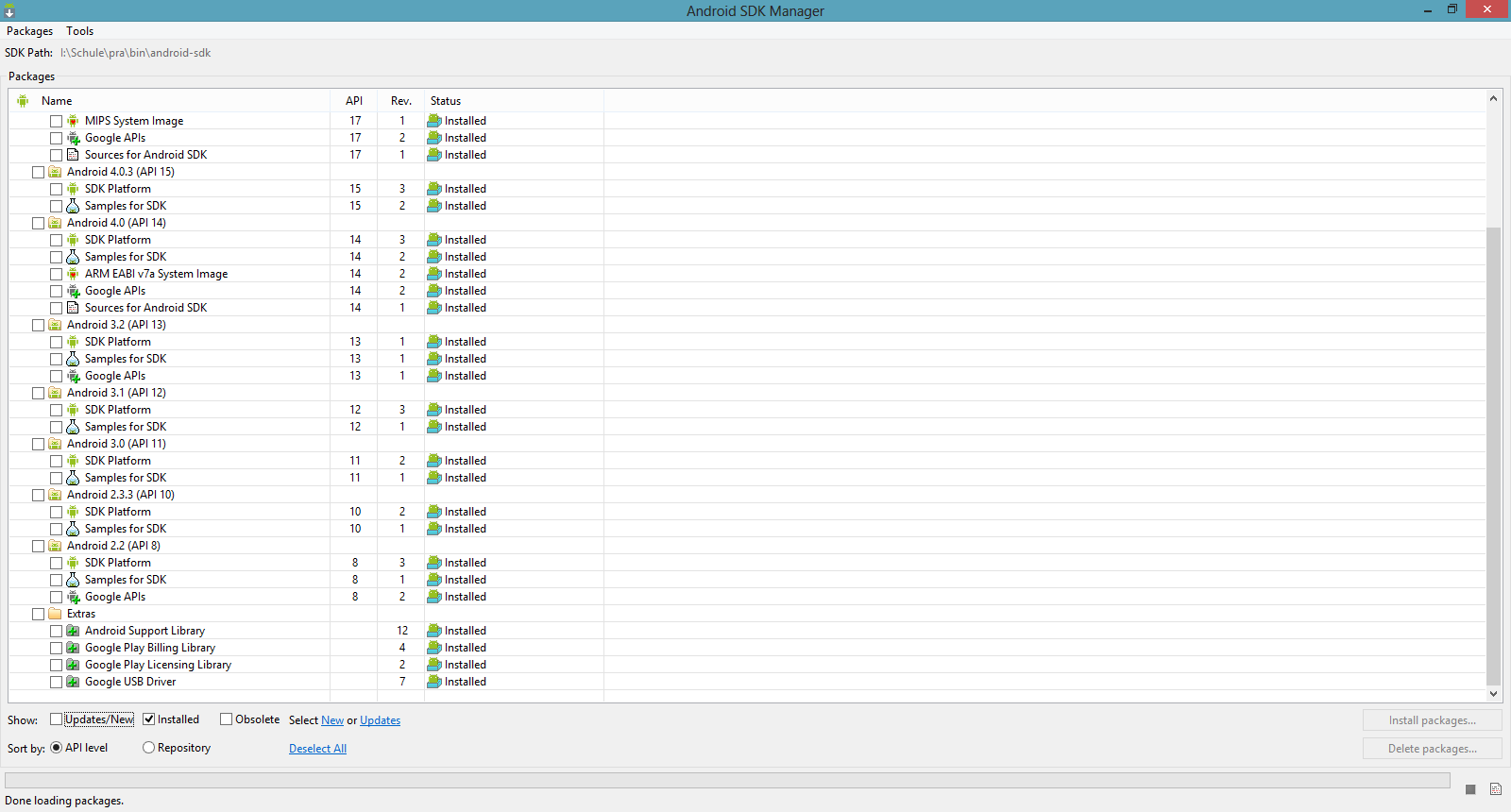


Abbildung 16 - Subclipse

### Android SDK Manager

Nachdem die Android Developler Tools und Subclipse installiert wurden, benötigten wir noch das entsprechende Software Development Kit (SDK).

Dazu wird der Android SDK Manager gestartet und es erfolgt eine Übersicht über die verfügbaren Android Versionen. Weiters wird beim Starten des Managers ein Update auf neue Versionen und Komponenten der API Versionen durchgeführt.

Bei der Installation der SDK ist noch zu beachten, sich auch die Beispiele und Dokumentationen mit zu installieren, da diese oft sehr hilfreich sein können.

Abbildung 17 - Installierte SDKs

Sind auch diese Punkte erledigt steht der Android Programmierung nichts mehr im Wege.

## Android Grundkonzept

### Komponenten

#### Activity

Eine Activity ist die Komponente einer Applikation, die die Präsenationsschicht für den Benutzer bereitstellt. Mit Activities ist es möglich interaktionen mit dem Benutzer zu realisieren. Ein Beispiel hierzu wäre, dass der Benutzer mit einer Activity eine Email versenden kann.

Üblicherweise gibt es eine „main“ Activity, die immer beim Starten der Applikation angezeigt wird. Jede Activity kann eine andere Activity aufrufen, um weitere Aktionen auszuführen.

#### View

Die View bildet den Bereich, in der alle User Interface Elemente enthalten sind. Eine View ist die Basisklasse von *Widgets*. Mit Widgets können alle UI Komponenten, wie z.B. Buttons, Textfelder,... erstellt werden.

#### Intent

Ein Intent (= Absicht) erlaubt es, dem Entwickler einer Applikation auf Ressourcen des Systems zuzugreifen, Dies kann entweder direkt oder indirekt passieren. Beim indirekten Aufruf ist noch nicht bekannt, welche Ressourcen verwendet werden, beim direkten Aufruf hingegen schon. So kann z.B. ein Intent definiert werden, der alle Programme zur Verfügung stellt, die zum Aufnehmen eines Fotos auf dem Handy installiert sind. Der User kann dann wählen welche App er zur Aufnahme eines Fotos verwenden möchte.

### Architektur einer Applikation

#### Android Manifest

Jede Android Applikation benötigt eine *AndroidManifest.xml* Datei. Mit Hilfe dieser Datei wird die Version festgelegt, auf der die *.apk* Datei laufen soll. Zusätzlich werden auch noch sämtliche Berechtigung, die die Applikation zum reibungslosen Ablauf benötigt, deklariert. Zu guterletzt müssen noch alle Klassen, welche Activities enthalten, müssen ebenfalls definiert werden.

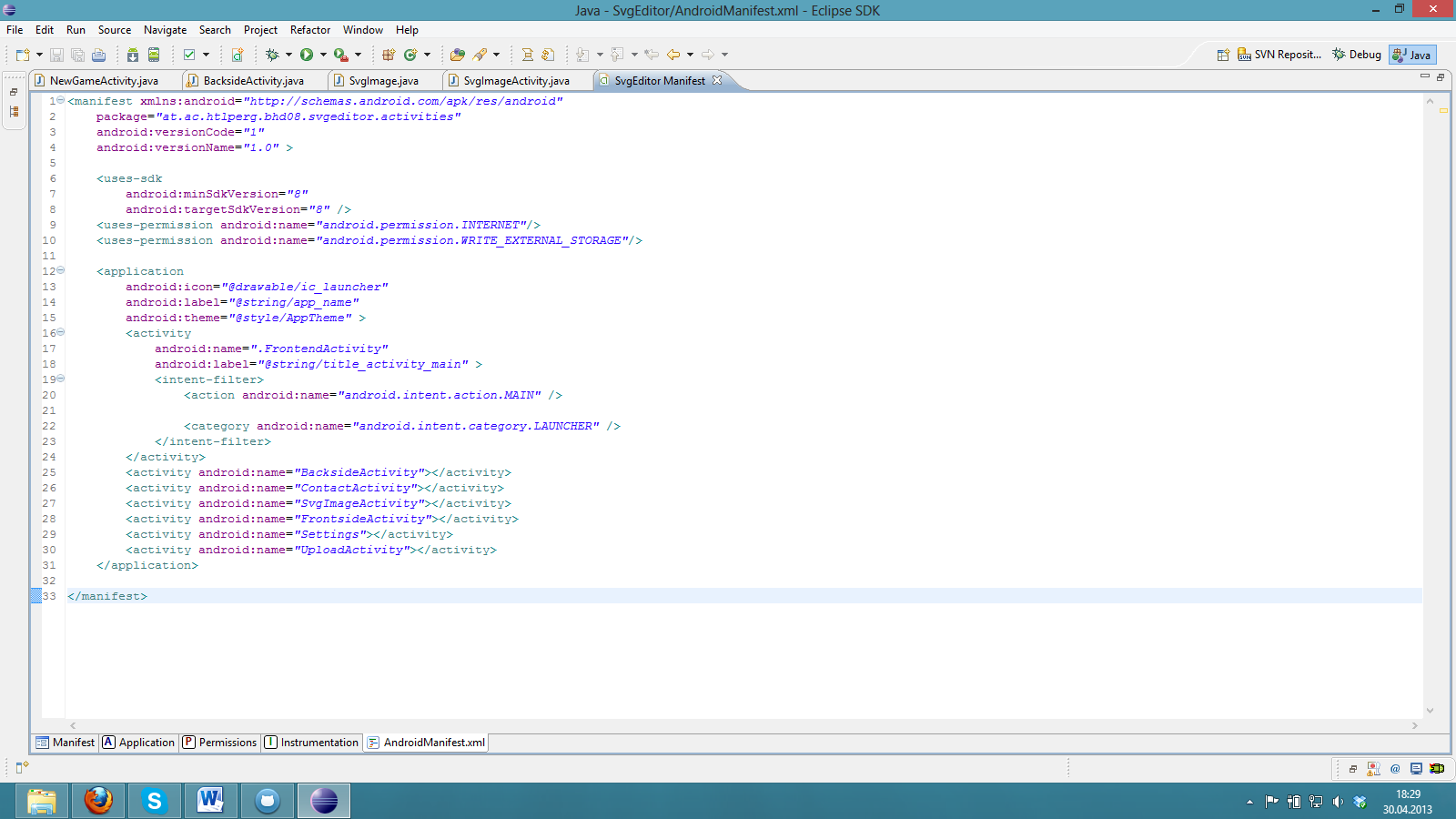


Abbildung 18 - AndroidManifest.xml unserer Diplomarbeit

#### R.Java und Ressourcen

Alle für die Applikation benötigten Ressourcen befinden sich im Ordner *res*. Für jeden Ressourcentyp gibt es eine eigen Unterklasse. Zum Beispiel für die Ressource „drawable“ wird eine Klasse R.drawable erzeugt.

Außerdem bekommen alle Elemente von Ressourcen eine eindeutige ID, die im *R.java* File gespeichert werden.

#### Assets

In diesem Ordner können alle Dateien abgelegt werden, auch solche, die von Android nicht erkannt werden. Mit dem AssetManager ist es möglich auf diese Dateien zuzugreifen.

# 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, also bereits genormte Piktogramme usw., realisiert.

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 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 „Ä“, „Ö“ 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.

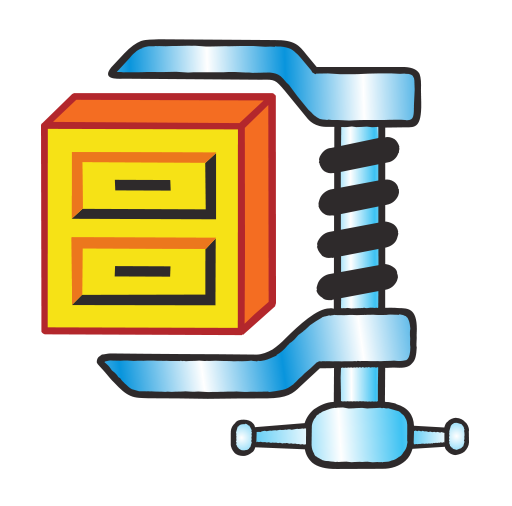


Abbildung 19 - ZIP Logo

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 jedes 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.

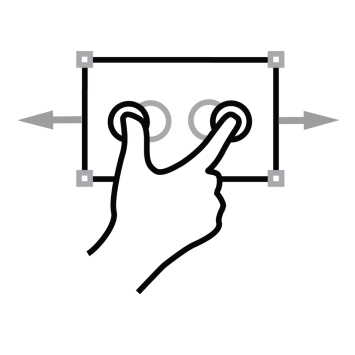
Das Template auf der Vorderseite ist die eigentliche Gestaltung durch den User. Er wählt aus verschiedenen mitgelieferten Vorlagen oder kann sich diese mithilfe eines Repository-Files vom Server herunterladen. Anschließend kann der User die Postkarte mit einem eigenen Bild frei gestalten. Er hat somit 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 20 - 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;

}

}**//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

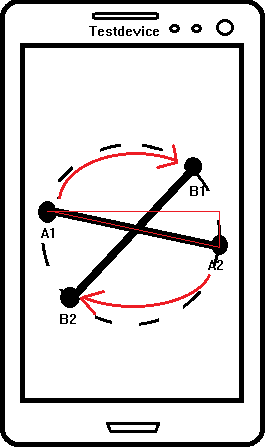
Eine größere Herausforderung ist die Bestimmung der Rotation. Die bekannte Geste hierfür ist, dass der Benutzer zwei Finger im Kreis bewegt und der Winkel von einem der absolute Winkel (relativ zur Bildschirmvertikale) bzw. der Unterschied in der Bewegung der Kreisbewegung die Rotationsänderung bestimmt.

Abbildung 21 – Zwei-Finger-Rotation

Es wird hierbei der Winkel von A1 zur Bildschirmvertikale besimmt, wobei auch A2 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 oben 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, sondern 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

**Projektumfeld**

http://www.aberger.at/index.php/unternehmen/team.html

http://www.htl-perg.ac.at/organisation/lehrer

Zugriff am 25.04.2013

**Java**

http://de.wikipedia.org/wiki/Java-Technologie

http://de.wikipedia.org/wiki/Sun\_Microsystems

http://de.wikipedia.org/wiki/Java\_(Programmiersprache)

http://www.teialehrbuch.de/Kostenlose-Kurse/JAVA/6534-Plattformunabhaengigkeit.html

http://www.bbs-1.de/bbs1/umat/java\_kurs/SelfJAVA/java0102.htm

http://de.wikipedia.org/wiki/OpenJDK

http://de.wikipedia.org/wiki/Java\_Development\_Kit

http://de.wikipedia.org/wiki/Java-Laufzeitumgebung

http://de.wikipedia.org/wiki/Just-in-time-Kompilierung

**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

**Betriebssystem einrichten**

http://developer.android.com/sdk/installing/installing-adt.html

http://stackoverflow.com/questions/4419868/what-is-the-current-directory-in-a-batch-file

**Eclipse einrichten**

<http://de.wikipedia.org/wiki/Umgebungsvariable>

http://www.admin-wissen.de/tutorials/eclipse\_workshop/subversion\_in\_eclipse.html

**Android Programmaufgabe**

http://developer.android.com/guide/components/fundamentals.html

http://developer.android.com/guide/components/activities.html

http://developer.android.com/guide/topics/resources/accessing-resources.html

http://developer.android.com/reference/android/view/View.html

Abbildungsverzeichnis

[Abbildung 1 - Aberger Software GmbH Logo 13](#_Toc355111887)

[Abbildung 2 - Martin Brandner 13](file:///I:\Schule\HTBLA_Perg\Repository\Dokumente\DiplomarbeitSvgEditor.docx#_Toc355111888)

[Abbildung 3 - DI Christian Aberger 14](file:///I:\Schule\HTBLA_Perg\Repository\Dokumente\DiplomarbeitSvgEditor.docx#_Toc355111889)

[Abbildung 4 - Java Logo 16](#_Toc355111890)

[Abbildung 5 - Android Logo 20](#_Toc355111891)

[Abbildung 6 - Eclipse Logo unserer Entwicklungsumgebung Indigo 24](#_Toc355111892)

[Abbildung 7 - Subversion Logo 26](file:///I:\Schule\HTBLA_Perg\Repository\Dokumente\DiplomarbeitSvgEditor.docx#_Toc355111893)

[Abbildung 8 - Beschreibt das Zusammenspiel zwischen einem Subversion Server und mehreren Clients. 27](#_Toc355111894)

[Abbildung 9 - Diagramm Arbeitsleistung 31](#_Toc355111895)

[Abbildung 10 - Diagramm Lizenzkosten 32](#_Toc355111896)

[Abbildung 11 - Diagramm Hardwarekosten 33](#_Toc355111897)

[Abbildung 12 - Diagramm Gesamtkosten 34](#_Toc355111898)

[Abbildung 13 - Betriebssystem einrichten 35](#_Toc355111899)

[Abbildung 14 - Umgebungsvariablen setzen 37](#_Toc355111900)

[Abbildung 15 - ADT Plugin 38](#_Toc355111901)

[Abbildung 16 - Subclipse 38](#_Toc355111902)

[Abbildung 17 - Installierte SDKs 39](file:///I:\Schule\HTBLA_Perg\Repository\Dokumente\DiplomarbeitSvgEditor.docx#_Toc355111903)

[Abbildung 18 - AndroidManifest.xml unserer Diplomarbeit 41](#_Toc355111904)

[Abbildung 19 - ZIP Logo 52](file:///I:\Schule\HTBLA_Perg\Repository\Dokumente\DiplomarbeitSvgEditor.docx#_Toc355111905)

[Abbildung 20 - Zwei-Finger-Zoom 54](file:///I:\Schule\HTBLA_Perg\Repository\Dokumente\DiplomarbeitSvgEditor.docx#_Toc355111906)

[Abbildung 21 – Zwei-Finger-Rotation 56](file:///I:\Schule\HTBLA_Perg\Repository\Dokumente\DiplomarbeitSvgEditor.docx#_Toc355111907)

Tabellenverzeichnis

[Tabelle 1 - Android Versionen 23](#_Toc355111882)

[Tabelle 2 – Arbeitsleistung 31](#_Toc355111883)

[Tabelle 3 – Lizenzkosten 32](#_Toc355111884)

[Tabelle 4 – Hardwarekosten 33](#_Toc355111885)

[Tabelle 5 – Gesamtkosten 34](#_Toc355111886)

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)