Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Studienarbeit

# Copy und Paste zwischen verschiedenen Smartphones


**Boris Goldshteyn**
Matr.Nr.: 633712
Bayer Pharma AG

**Clemens Wagner**
Matr.Nr.: 665250
Samuelson Kassensysteme GmbH

**Christian Wolter**
Matr.Nr.: 695136
Berliner Stadtreinigung


| | |
|---|---|
| **Gutachter:** | Herr Kalkbrenner |
| **Studienjahrgang:** | IT2011 |
| **Fachrichtung:** | Informatik |
| **Studienbereich:** | Technik |
| **Fachbereich:** | FB 2 - Duales Studium |
| **Wörter:** | ca. 8500 |

## Verfassung der Studienarbeit

Diese Arbeit wurde von allen drei Autoren zu gleichen Teilen verfasst. Auf eine genaue Kennzeichnung der einzelnen Abschnitte dieser Studienarbeit wird deshalb verzichtet.

# Kurzfassung

Zur gegenwärtigen Zeit sind Smartphones nicht mehr wegzudenken. Die Bezeichnung umfasst Handys mit besonders leistungsfähiger Hardware. Da diese Geräte mit kleinen Computern vergleichbar sind, haben sie auch ein komplexes Betriebssystem.

Häufig möchten Smartphonebenutzer Dateien, Filme, Musik und Programme austauschen. Dieser Vorgang gestaltet sich oftmals kompliziert und man wünscht sich eine intuitivere Bedienung. Ein simpler Kopiervorgang - Copy & Paste - wie beim Windows-Computer wäre ideal. Deshalb verfolgt dieses Projekt die Schaffung einer Applikation (App), die dieses Konzept umsetzt.

Nach der Analyse des Problems wird Grundlagenforschung betrieben. Es muss eines der vorhandenen gängigen Betriebssysteme als Basis ausgewählt werden. Das Bedienkonzept muss spezifiziert werden. Das beinhaltet sowohl die Bedienung auf Sender-, als auch auf Empfängerseite. Außerdem muss eine Übertragungstechnik bestimmt werden, die den vorher festgelegten Kriterien entspricht.

Anschließend wird in der Entwurfsphase die Architektur entwickelt und das Design der grafischen Oberfläche entworfen.

Die umfangreiche Implementierungsphase setzt die geplanten Funktionalitäten und Spezifikationen in einen lauffähigen Programmcode um. Hierbei werden auch einige Besonderheiten von Android, dem ausgewählten Betriebssystem, erläutert. Danach folgt eine Evaluation. Mithilfe einer Umfrage werden Schwachpunkte des Prototypen ermittelt, um diese für zukünftige Weiterentwicklungen zu dokumentieren.

# Inhaltsverzeichnis

# Abbildungsverzeichnis



# Tabellenverzeichnis

# Glossar

**Access Point**
: Zugangspunkt eines kabellosen Netzwerkes, Endgeräte kommunizieren über Funk mit dem Access Point, der die Daten weiter verteilt

**Android**
: Betriebssystem für Smartphones

**API**
: Programmierschnittstelle eines Systems für Entwickler

**APK**
: Installationspaket für das Android-Betriebssystem

**App**
: Kurzform für Applikation - häufig bei Anwendungen für Smartphones oder kleine leichtgewichtige Programme mit nur wenigen Funktionalitäten

**Apple Store**
: Appstore von Apple (siehe Appstore)

**Appstore**
: Onlineplattform zum kaufen von Anwendungen für Smartphones

**ARM**
: Architektur für Prozessoren entwickelt von dem gleichnamigen Unternehmen

**Batch-Datei**
: Eine Datei, welche Befehle enthält die sequenziell abgearbeitet werden

**Bootvorgang**
: Startvorgang eines System

**C++**
: Programmiersprache

**Chat**
: direkte (synchrone) Kommunikation über kurze Textnachrichten

**Cloud**
: onlinebasierter Speicher

**EAN-Nummer**

europaweite eindeutige Produktidentifikationsnummer

**Eclipse**

Entwicklungsumgebung für Java und andere Sprachen

**Emulator**

Ein System, welches das Verhalten des Zielsystems nachahmt

**Feature**

Funktionalität oder Eigenschaft eines Systems

**Framework**

Ein Rahmenwerk für Entwickler, welches viele Basisfunktionalitäten bereitstellt

**Google Playstore**

Appstore von Google (siehe Appstore)

**HowTo**

How To Do, Erklärung und Beschreibung wie etwas getan werden sollte

**Instant-Messaging**

siehe Chat

**iOS**

Betriebssystem für Smartphones von Apple

**iteratives Vorgehen**

Vorgehensweise bei dem die Entwicklung in kleinen Schritten passiert, wobei jede Iteration für sich geplant wird

**Java**

Programmiersprache

**Multitasking**

gleichzeitig bzw. pseudogleichzeitiges Abarbeiten mehrerer Prozesse

**Objective C**

Programmiersprache

**Plugin**

Erweiterung einer Software, um nicht vorhandene Funktionen zu ergänzen

**Prototyp**

Eine meist nicht ganz vollständige Umsetzung ohne jegliche Optimierung zu Demonstrationszwecken

**Push-Mail**

System, bei dem das Endgerät über neue Nachrichten informiert wird

**QT-Framework**

Ein C++ Framework für die systemübergreifende Programmierung inklusive einer grafischen Oberfläche

**Release**

meist veröffentlichter Stand einer Software welche einen definierten Funktionsumfang besitzt

**Symbian**

Betriebssystem für Smartphones ehemals verbreitet durch Nokia

**Terminal**

Zugangspunkt zu einem System, hier: Rechner der keine eigene Funktionalität anbietet, sondern lediglich als Schnittstelle zu einem Server dient

**Ticketsystem**

System zum Erfassen und Verwalten von Aufgaben, Problemen etc.

**Touchscreen**

berührungsempfindliches Display zur Bedienung (resistiv: durch druck, kapazitiv: aufgrund von elektrischer Ladung im Finger)

**Usability**

Bedienbarkeit und Benutzerfreundlichkeit eines Systems

**Workflow**

Arbeitsfluss, eingespielte Vorgehensweisen

**Zwischenablage**

Speicher eines Systems der zur kurzzeitigen Ablage von Daten dient und von jedem Programm zugänglich ist

# Abkürzungsverzeichnis

IT ..... Informationstechnik

SDK ..... Software Developement Kit

NFC ..... Near Field Communication

URL ..... Uniform Resource Locator

UML ..... Unified Modeling Language

XML ..... Extensible Markup Language

ADB ..... Android Debug Bridge

TFS ..... Team Foundation Server

GUI ..... Graphical User Interface

MIME ..... Multipurpose Internet Mail Extensions

WYSIWYG .... What you see is what you get

APK ..... Application package file

LE ..... Low Energy

EAN ..... European Article Number

# 1 Einleitung

Der Absatz und die Nutzung der mobilen Endgeräte mit einer hohen Computerfunktionalität, auch Smartphone genannt, ist in den letzten Jahren drastisch gestiegen. Ende 2012 gab es weltweit 740 Millionen Geräte - davon alleine 31 Millionen hier in Deutschland. Somit besitzt mehr als jeder Dritte deutsche Mitbürger ein Smartphone, das sind über 50% aller Handynutzer in Deutschland. [5]

Diese Zahlen führen zu einer hohen Zahl von Entwicklern, die Applikationen für diese Geräte schreiben. Die Anzahl solcher Applikationen, die in der Kurzform Apps genannt werden, ist im Google Playstore bis 2013 auf 850000 - im Apple Store auf sogar 900000 gestiegen. [4]

Im Rahmen des Informatikstudiums an der Hochschule für Wirtschaft und Recht in Berlin soll eine Applikation entwickelt werden, die das von einem herkömmlichen Computer bekannte Konzept des Copy&Paste auf das Smartphone überträgt. Die Idee ist, eine intuitive Bedienung nicht nur zwischen den Applikationen eines Gerätes zu erlauben, sondern die Erweiterung auf das Übertragen von Daten und Dateien zwischen mehreren verschiedenen Geräten.

# 2 Analyse

## 2.1 Problemstellung

Häufig möchten Smartphone-Nutzer einen URL-Link, Text oder kleine Dateien miteinander austauschen. Dieser Vorgang soll schnell ablaufen. Außerdem sollten auch keine komplizierten Bedienungsschritte notwendig sein. Nicht jeder Smartphone-Besitzer verfügt über weitreichende IT-Kenntnisse und ist in der Lage, ein aufwändiges Procedere zum versenden einer Datei durchzuführen.

Für das Versenden von Texten und Dateien gibt es bereits Applikationen. Nennenswert sind an dieser Stelle internetbasierte Chat-Anwendungen (z.B. WhatsApp), internetbasierte Cloud-Speicherdienste (z.B. Dropbox) und auch betriebssystemeigene Möglichkeiten zum Versenden von Dateien, zum Beispiel via Bluetooth oder NFC.

Jedoch erlauben diese keine konsequent einfache Bedienung. Bei einer Chat-Anwendung muss eine zu versendende Datei erst in der Applikation ausgewählt werden und ein Text muss kopiert und anschließend im Chat wieder eingefügt werden.

Bei einem Cloud-Speicherdienst muss das zu versendende Medium eine Datei sein, die in der Cloud gespeichert wird. Anschließend muss ein Download-Link dem empfangenden Smartphone übermittelt werden. An diesem Punkt scheitert ein Cloud-Speicherdienst bei der Usability. Die betriebssystemeigenen Möglichkeiten beschränken sich auf Dateien. Ein kopierter Text kann nicht, ohne als Text-Datei vorzuliegen, versandt werden.

Wünschenswert wäre für dieses Problem eine Copy&Paste-Funktion zwischen mehreren Smartphones. Auf einem Gerät wird ein Medium kopiert und auf dem empfangenden Gerät eingefügt.

## 2.2 Problemanalyse

Für das in 2.1 beschriebene Problem gibt es schon einige Lösungsansätze. Zum Einen existiert das von Google selbst angebotene neue Feature AndroidBeam, welches die Technik des NFC nutzt. Dieses bietet die Möglichkeit sehr einfach und schnell Webseiten, Texte und Dateien zwischen Smartphones mit der neuesten Android Version und einem NFC-Chip untereinander auszutauschen. Zum Anderen existieren schon einige Apps, die das Konzept des Copy&Paste aufgreifen. Hierbei wird auf die Applikation „ClipPick“ verwiesen. Das ist eine Anwendung für alle Arten von mobilen Geräten und auch für den PC. Hierbei wird genau dieses einfache Copy&Paste Konzept umgesetzt, indem es in erster Linie Texte, aber auch Screenshots und andere Dateitypen via Cloud auf ein anderes Gerät direkt in die Zwischenablage verschickt. Dabei müssen beide Geräte in der Cloud angemeldet sein und über eine aktive Internetverbindung verfügen. [19]

Unsere Lösung sollte keine schon vorhandene kopieren oder ersetzen, sondern alle verfügbaren Techniken und alle vorhandene Lösungen mit einem simplen und sicheren Bedienungskonzept vereinen und eine neue Applikation schaffen, die keine Einschränkungen, keine besondere Kenntnis und keine weiteren Anwendungen seitens des Benutzers braucht.

Das Augenmerk muss dabei auf der intuitiven Bedienbarkeit liegen. Unter diesen Gesichtspunkt muss die optimale Technologie ausgewählt und das optimale Bedienkonzept entwickelt werden. Die Umsetzung sollte das Konzept des Copy&Paste möglichst nahtlos umsetzen. Als Orientierung dient hier das Copy&Paste Prinzip, wie es unter Windows 7 implementiert ist. Ein Text oder eine Datei wird ausgewählt und durch einen Rechtsklick im Kontextmenü der Punkt Kopieren ausgewählt. Navigiert man nun zu der Stelle, an der das Dokument oder der Text eingefügt werden soll, gelingt dies über den Kontextmenüpunkt „Einfügen“. Auch die meisten Smartphone Betriebssysteme implementieren bereits eine Copy&Paste Funktionalität. So ist es z.B. bei Android und iOS möglich, durch langes Antippen eines Textes ein Kontextmenü aufzurufen, über welches sich der Text in die systemeigene Zwischenablage legen lässt. Dieser Text kann ebenfalls wieder über das Kontextmenü an anderer Stelle eingefügt werden. Eine Möglichkeit, diesen Text an ein anderes Smartphone zu schicken, bietet jedoch kein Betriebssystem. Somit muss das Copy&Paste Konzept dahingehen erweitert werden, dass Texte ebenfalls auf anderen mobilen Telefonen eingefügt werden können, ohne den Workflow des Benutzers zu unterbrechen.

# 3 Konzeptioneller Entwurf

In der Entwurfsphase der Applikation war es für das Team notwendig einige Studien durchzuführen, um sich auf die Art und Weise der Funktion und Nutzung der Anwendung zu einigen. Daraus sollte eine strukturierte Planung hervorgehen, die wie bei einem iterativen Vorgehensmodell die einzelnen Schritte der Entwicklung festlegt.

## 3.1 Betriebssystemstudie

Einer der ersten Punkte war es, sich auf ein Betriebssystem zu einigen. Zur Auswahl standen alle uns bekannten, und bei Smartphones noch in Verwendung befindliche Betriebssysteme. Insbesondere Android und iOS sind hier zu nennen, da diese sich in den Marktanteilen deutlich von den anderen Systemen absetzten. Um eine möglichst gründliche Entscheidung zu treffen, stellten wir eine Entscheidungsmatrix auf. Dabei wurde die Wahl anhand von 3 Kriterien festgesetzt: Das Vorhandensein in den Geräten, die benötigte Entwicklungslandschaft und die Höhe der Entwicklungskosten.

### Symbian

Symbian ist ein bei den meisten heutigen Smartphonenutzern eher unbekanntes Betriebssystem. Dass es nur so wenige kennen, liegt eher an der Vermarktung als an der Verbreitung. Noch 2010 hatte Symbian einen Markteinteil von fast 38%. [20] Entwickelt wurde Symbian von einem Konsortium aus namhaften Handyherstellern wie Ericsson, Motorola und Nokia. Nokia lieferte nahezu ohne Ausnahme alle seine Geräte mit Symbian aus, weshalb Nokia sich zur stärksten treibenden Kraft bei der Entwicklung von Symbian entwickelte. Auch einige Samsung und Sony Ericsson-Geräte wurden mit Symbian ausgeliefert, jedoch stiegen diese Hersteller 2010 aus der Kooperation aus und setzten vermehrt auf das neue, immer stärker werdende Android. Mit Ausnahme des Umfangs bot Symbian jedoch alle Funktionalitäten, die man von einem heutigen Smartphonebetriebssystem erwartet.

Der Misserfolg kam mit dem Erfolg von Android und den Problemen von Nokia. Nokia hatte die Entwicklung Symbians so lange wie möglich aufrecht erhalten, doch es zeigte sich schnell, dass Nokia den Trend der Smartphones durch ihre Fixierung auf Symbian verpasst hatte. Durch weitere Schwierigkeiten verlor Nokia mehr an Boden und musste reagieren. Nokia entschloss sich somit auf das kommende Windows Phone 7 als Betriebssystem zu setzen[21], und stellte die Weiterentwicklung von Symbian vollständig ein.

Symbian ist in Anbetracht der Aufgabenstellung ein vollwertiges Betriebssystem, welches alle Funktionalitäten bietet, die zur Umsetzung nötig sind. Disqualifiziert wird es

jedoch durch die Tatsache, dass nicht mit einer Weiterentwicklung zu rechnen ist und die Marktanteile mittlerweile bereits im niedrigen einstelligen Bereich liegen.

Die Entwicklung von Anwendungen für Symbian ist aufgrund der kleinen Community nicht ganz so umfangreich dokumentiert. Dennoch stellt sich mit dem QT-Framework eine vollständige Entwicklungsumgebung für Symbiangeräte bereit, welche einem die Anwendungsprogrammierung in C++ erlaubt.

### iOS

Das von Apple für seine mobilen Geräte entwickelte Betriebssystem iOS dient als Grundlage einer ganzen Produktfamilie. So läuft das System nicht nur auf dem von Apple entwickelten iPhone, sondern ebenfalls auf deren Tablet-PC iPad und dem tragbaren Musikspieler iPod Touch.

iOS entstand unter dem Namen OS X als erste Version des von Apple 2007 herausgebrachten iPhone. Als Basis diente das damalige Mac OS X, welches an die Anforderungen eines Smartphones und eines ARM Prozessors angepasst wurde. Im tiefen Ursprung handelt es sich bei iOS somit um ein UNIX derivat. Nach einer weiteren Umbenennung in iPhone OS gelangte es aufgrund der Vielfalt der Geräte, auf denen es mittlerweile lief, im Jahr 2010 zu seinem heutigen Namen iOS.

Die Einführung des iPhone und damit auch die Einführung von iOS revolutionierten die Smartphonewelt. Smartphones gab es zu dieser Zeit bereits einige. Die Auswahl des Betriebssystems war allerdings sehr beschränkt. Blackberry und Symbian boten zwar je nach Rechenleistung des Handys einen großen Funktionsumfang, allerdings waren diese fast ausschließlich auf die Bedienung per Tastatur ausgelegt. Auch die alternative Windows CE bot zwar die Eingabe per Touchscreen, allerdings gelang dies nur mit einem Eingabestift oder einem angespitzten Fingernagel vernünftig. Grund waren die sehr teuren und kaum smartphoneoptimierten Touchscreens. Das Bedienkonzept, welches iOS nun mitbrachte, benötigte einen einfach mit dem Finger zu bedienenden kapazitiven Touchscreen. Somit trug das Betriebssystem iOS mit seinem Bedienkonzept entscheidend zum Erfolg des iPhones bei.

iOS wurde erst von Android vom Thron der Marktanteile verdrängt, hält sich aber weiterhin mit an der Spitze. Es ist somit nach wie vor sehr weit verbreitet und genießt eine ständige Weiterentwicklung. Durch die Produktarmut von Apple ist kaum eine individuelle Anpassung an die Hardware der mobilen Endgeräte notwendig, weshalb sich neue iOS Versionen sehr schnell auf den Geräten verbreiten. Der Umfang des Betriebsystem lässt sich durch ein umfangreiches App-Angebot schnell und einfach erweitern. Dennoch verfolgt Apple leider eine sehr strenge Produktpolitik. Grundlegend ist es auf den Geräten nur möglich, Apps aus Apples AppStore zu installieren. Eine eigenmächtige

Installation ist somit nicht möglich. Ein weiterer wunder Punkt ist, dass Apple das Multitasking ausschließlich für vorinstallierte Apps erlaubt. Dies sorgt zwar für eine meist gute Performance, erschwert aber an vielen Stellen die Entwicklung einer App. Schlussendlich erfordert die Entwicklung von Apps für iOS eine Registrierung bei Apple selbst. Dafür stellt Apple aber auch die vollständige Entwicklungsumgebung XCode zur Verfügung. Sie erlaubt es, in Objectiv C aber auch in C++ Apps zu entwickeln.

Bei der Wahl des Betriebsystems stellt sich iOS somit grundlegend positiv dar, allerdings sind die Bedingungen, an die Apple die Entwicklung einer App knüpft und die relativ seltene Programmiersprache Objective C sowie die Notwenigkeit eines Mac OS X, Hindernisse, welche iOS für die Entwicklung eines Prototypen disqualifizieren.

### Blackberry OS / Blackberry 10

Blackberry war einer der Marktführer auf dem Gebiet der Smartphones. Ursächlich hierfür ist die extrem starke Fixierung auf Businesskunden. Blackberry Geräte waren lange Zeit auf die Einhandbedienung ausgelegt. Die starke Businessoptimierung der Geräte führt dazu, dass diese ohne einen angebundenen BlackBerry-Enterprise-Server in ihrer Funktionalität sehr beschränkt sind. Schon seit jeher sind die mobilen Endgeräte von Blackberry mehr ein Terminal als ein vollständig eigenständiges Smartphone. Der jahrelange Erfolg der Geräte war jedoch genau durch diese starke Einschränkung gewährleistet. So waren Blackberrygeräte die ersten, die Push-Mail, Instant-Messaging, Synchronisation von Kalendern, E-Mail und Dokumenten schon seit Jahren unterstützen. Ermöglicht wird dies eben durch den Blackberry-Enterprise-Server und ein hochentwickeltes Datenübertragunsprotokoll. Durch die unglaubliche Optimierung schafft es Blackberry laut eigenen Angaben, dass das Empfangen von 500 E-Mails im Monat und etwa 100 Kalendereinträgen in der Woche kaum mehr als 1 MB im Monat verbraucht. Dies ermöglichte selbst in Zeiten von GPRS eine vollständige Funktionalität. Ein weiterer Punkt, für den der Name Blackberry stand, war die unabdingbare Datensicherheit und Datenschutz, sodass Firmen selbst kritische Daten an ihre mobilen Endgeräten zur Verfügung stellen konnten. Selbstverständlich bietet Blackberry OS auch Erweiterungen durch Apps. Durch den geringen Privatkundenanteil ist die Zahl der Anwendungen jedoch stark begrenzt. Durch diverse Schwierigkeiten mit der Erreichbarkeit der Blackberry-Server sowie Sicherheitslücken verlor auch Blackberry einiges an Marktanteilen. Mittlerweile bietet Blackberry auch ein Touchscreengerät an. Dafür wurde das alte Blackberry OS vollständig überarbeitet und erschien erst im Mai 2013 neu als Blackberry 10.

Durch die starke Businessorientierung und den mittlerweile bodenlos gesunkenen Marktanteil sowie die quasi zwingende Voraussetzung eines Blackberry-Enterprise-Servers ist es für die Entwicklung eines App-Prototypen absolut ungeeignet.

### WindowsPhone

WindowsPhone ist das von Microsoft entwickelte Smartphonebetriebssystem. Die erste Version WindowsPhone 7 setzt der Abstammung nach die Windows CE Linie fort. Genau wie iOS und Android ist WindowsPhone 7, anders als seine CE Vorgänger, ein vollständig auf Fingertouchbediengung ausgelegtes Betriebssystem. Auch Microsoft investierte viel Arbeit in die Entwicklung eines guten und innovativen Bedienkonzepts. So entwickelte Microsoft die Kacheloberfläche, welche ein sehr flexibles Design und Layout der Benutzeroberfläche zulässt. Zudem ist das WindowsPhone Bestandteil von Microsofts Plan ein System zu schaffen, welches über alle Geräte hinaus gleichermaßen zu bedienen ist. Mit Windows 8 zeigt sich, dass Microsoft die Kacheloberfläche auch auf Desktoprechnern eingeführt hat. Um diesen Schritt weiter zu gehen, entwickelte Microsoft sein WindowsPhone Betriebsystem in der Version 8 dahingehend weiter, dass des nun auf dem Windows NT Kernel basiert. Damit zieht es gleich mit Windows 8 und dem für ARM konzipierten Windows RT. Dennoch scheint Microsoft noch mit vielen Schwierigkeiten zwischen den einzelnen Systemen zu kämpfen, denn außer des Look-and-Feels funktioniert einiges noch nicht systemübergreifend. Dennoch lassen sich prinzipiell bereits alle im App-Store verfügbaren Apps nicht nur auf den mobilen Endgeräten mit WindowsPhone oder Windows RT installieren, sondern gelangen auch auf die Desktoprechner. Auch um dies zu gewährleisten, verfolgt Microsoft in seiner App-Politik eine ähnlich strenge Linie wie Apple in ihrem AppStore. Durch die zwanghafte Bindung an den Microsoft Store ist es nicht ohne weiteres möglich, Apps aus anderen Quellen zu installieren.

Diese Tatsache und auch die noch sehr geringe Verbreitung sprechen gegen die Verwendung von WindowsPhone zur Implementierung eines Prototypen. Dennoch stellt sich WindowsPhone als sehr positives System dar, die Entwicklung geschieht über für Microsoft typische Wege mit VisualStudio und dem .NET Framework und ist somit nicht weiter fremd.

### Android

Als weiteres System geht Android ins Rennen. Durch seine enorme Verbreitung setzt es sich bereits weit von allen anderen Betriebssystemen positiv ab. Android ist das zu großen Teilen quelloffene Betriebssystem der Open Handset Alliance, welche ins Leben gerufen wurde, um öffentliche Standards für Mobilgeräte zu entwickeln. Gründer und eine der großen treibenden Kräfte der Allianz ist Google. Die Entwicklung von Android begann bereits früher als Kamerabetriebsystem. In Zuge der Allianz gab Google jedoch 2007 bekannt, dass Android nun als Smartphonebetriebssystem entwickelt werden würde. Bereits 2008 erschien die erste Android Version auf dem Markt. Auch Android ist ein vollständig auf Fingertouchbedienung ausgelegtes mobiles Betriebssystem. Es basiert auf

| Kriterium | Android | iOS | Windows Phone | Symbian | BlackberryOS |
|---|---|---|---|---|---|
| Verbreitung | 74,30% | 16,60% | 6,40% | 2,00% | 0,40% |
| Entwicklungslandschaft | Java mit Eclipse, XML und GUI Designer | Objective C und Xcode | .NET Visual Studio | QT C++, GUI Designer | unbekannt |
| Entwicklungskosten | kostenfrei | Developer Account wird benötigt | kostenfrei | kostenfrei | unbekannt |

Tabelle 1: Studien zur Auswahl des Betriebssystems

einem Linuxkernel und wurde der mobilen Umgebung und der Touchbedienung angepasst. Google verfolgt, anders als Microsoft und Apple, eine sehr offene App-Politik. Prinzipiell ist es möglich Apps aus jeder beliebigen Quelle zu installieren. So existieren auch bereits mehrere Appstores neben dem Google eigenen Playstore, wie z.B. der AndroidPit Store.

Android überzeugt in vielen Bereichen. Mit über 70% Marktanteil ist Android im Bereich der weltweiten Verbreitung ungeschlagen. [14] Auch die einfache und kostenlose Entwicklung unter Java mit einem vollständigen SDK bringt Android eine positive Bewertung ein und ist somit weit vor dem direkten Konkurrenten iOS, der für die Entwicklung eine eigene Sprache und einen Developer Account von Apple benötigt. Durch das quelloffene Design von Android und den großen Marktanteil ist die Entwicklercommunity riesig und es gibt viel Hilfestellung, Tutorial und HowTos. Somit zeigt sich Android als nahezu prädestiniertes Betriebssystem zur Entwicklung eines Prototypen.

Die Kriterien im Einzelnen zeigt ebenfalls die Tabelle 1. Es ist klar zu sehen, dass Android alle Kriterien am besten erfüllt. Daher wird zur Entwicklung des Prototypen Android als das Entwicklungssystem festgelegt.

## 3.2 Technologiestudie

Bestandteil der Entwurfsphase ist die Auswahl einer geeigneten Übertragungstechnologie zwischen den zwei Endgeräten. Zur Auswahl stehen verschiedene Techniken. Außer dem USB-Datenkabel sind die Techniken durchgehend drahtlos. Zu ihnen zählen Bluetooth, NFC, WiFi, Infrarot, QR-Code und das Internet in Verbindung mit einem Cloudspeicherdienst (hier Dropbox).

Zur objektiven Bewertung wurden Kriterien festgelegt. Abhängig von seiner Wichtigkeit besitzt jedes Kriterium eine Punktzahl. Die höchste Punktzahl wird an die Verfügbarkeit im Betrieb, Verbreitung in Geräten und Benutzerfreundlichkeit vergeben. Der Grund ist

eine größtmögliche Abdeckung der Kunden und eine einfache Nutzung der Technik für die Verringerung des Aufwands. Dazu hat jedes Kriterium einen dreistufigen Maßstab mit einer Farbskala von grün (gute Erfüllung des Kriteriums) über gelb (neutrale Wertung) bis rot (schlechte Wertung). Bei maximaler Erfüllung (grün) wird die entsprechende Punktzahl vergeben. Bei einem neutralen Erfüllungsgrad (gelb) werden keine Punkte vergeben. Bei der schlechtesten Wertung (rot) wird die Punktzahl für das entsprechende Kriterium abgezogen.

Zu den getesteten Techniken zählen absteigend nach der Punktzahl: Bluetooth, NFC, QR-Code, Wifi Direct, WLAN Infrastruktur, 3G Internet und Infrarot. Auch das USB-Kabel wurde mit in die Studie einbezogen und erreicht verhältnismäßig viele Punkte. Da das USB-Kabel allerdings große Einschränkungen in der Bewegungsfreiheit mit sich bringt und auch unterwegs nicht immer ein USB-Kabel und ggf. auch ein USB On The Go Adapter verfügbar sind, scheidet das USB als Übertragungstechnik aus.

Die komplette Technologiestudie im Detail ist im Anhang auf der Seite XII zu finden.

### Bluetooth und Bluetooth Low Energy

Den ersten Platz unserer Studie belegt mit 160 Punkten die Technik Bluetooth. Dieser Funkstandard wurde in den neunziger Jahren entwickelt. 2009 wurde Bluetooth um Bluetooth Low Energy erweitert, welches auf dem selben Protokoll beruht, aber eine weitaus energiesparendere Nutzung ermöglicht. Bluetooth Low Energy (LE) belegt mit 100 Punkten den 2. Platz unseres Rankings.

Bluetooth gilt in der aktuellsten Version 4.0 als die beste Technik. Durch die hohe Verbreitung in Geräten und die ständige Verfügbarkeit während der Nutzung senkt Bluetooth die Kosten und erhöht die Benutzerfreundlichkeit mit dem einfachen und bekanntem Umgang. Die Übertragungsdistanz von 20 Metern reicht für unsere Anforderungen vollkommen aus. Einzig die mittelmäßige Übertragungsrate und eine mögliche Datenunsicherheit sind Kritikpunkte an der Technik.

Die Low Energy Erweiterung wirkt sich im Gegensatz zu normalem Bluetooth in der Verbreitung in Geräten negativ aus. Außerdem ist die Übertragungsdistanz und -rate für einen geringeren Energieverbrauch eingeschränkt auf nur 10 Meter und 1Mbit/s. Deshalb büßt diese Erweiterung Punkte bei der Bewertung ein und landet deshalb auf dem 2. Platz.

### NFC

NFC ist bereits seit einigen Jahren im Einsatz. In die Smartphonewelt beginnt es jedoch erst jetzt seinen Einzug. NFC steht für Near Field Communication, was so viel heißt

wie Nahfeldkommunikation, was bereits darauf hinweist, dass es nur über eine stark begrenzte Reichweite verfügt. Dabei grenzt sich NFC durch einige Merkmale deutlich von Bluetooth und WLAN ab. Der wohl wichtigste Unterschied ist die Reichweite. NFC bietet eine Reichweite von maximal 5cm. Dies bedeutet, dass Geräte, die Daten austauschen wollen, sich direkt nebeneinander befinden müssen, damit überhaupt eine Verbindung möglich ist. Das Positive an NFC ist eine permanente Verfügbarkeit und durch diese geringe Übertragungsdistanz schlussfolgernde geringe Störanfälligkeit und Datensicherheit bei der Übertragung von Daten. Leider ist die Verbreitung von NFC Chips noch nicht weiter vorangeschritten, was hoffentlich in den nächsten Jahren anwachsen wird. Die niedrige Übertragungsdistanz und die schmale Übertragungsbandbreite von 424kbit/s sorgen allerdings bei größeren Datenmengen für eine geringe Benutzerfreundlichkeit. In solch einem Fall müssen die Geräte Rücken an Rücken gehalten werden, bis die Übertragung abgeschlossen ist. Im Gesamtwert kommt NFC als mögliche Technik auf 100 Punkte, welches ihm zusammen mit Bluetooth LE den 2. Platz einbringt.

### QR-Code

QR-Codes werden schon seit geraumer Zeit in der Industrie und Logistik verwendet. Toyota initiierte die Entwicklung des QR-Codes bereits 1994 und beauftragte das Tochterunternehmen des Zulieferers, Denso Wave, damit. [24] QR-Code steht für Quick Response und heißt damit so viel wie „schnelle Antwort“. Er sollte es ermöglichen, Informationen schnell aus einem Label oder Aufdruck auszulesen. Ziel war es mehr Informationen in dem Code speichern zu können als es ein einfacher Barcode vermag, der gerade einmal Platz für eine EAN-Nummer bietet. [22] Um dies zu erreichen, wurde aus dem eindimensionalen Barcode der zweidimensionale QR-Code. Er besteht aus kleinen Quadraten, die in einem quadratischen Raster angeordnet sind. In drei der Ecken des Rasters befinden sich zur Ausrichtung notwendige große Quadrate. Daneben besitzt ein QR-Code noch weitere Details zur Typisierung, Orientierung und Synchronisation. Die Daten selbst bilden den größten Teil der Pixel, wobei sich ebenfalls noch je nach Klasse des QR-Codes Prüfsummen und Redundanzen neben den Daten befinden. Diese stellen sicher, dass der Code selbst bei leichter Verschmutzung oder schlechter Lesbarkeit korrekt ausgelesen werden kann bzw. Fehler eindeutig erkannt werden.

Das Einlesen eines QR-Codes auf einem heutigen Smartphone dauert nicht länger als wenige Sekunden. Am längsten benötigt die Kamera, um den Code scharfzustellen und zu fokussieren. Allerdings ist für die Datenübertragung per QR-Code stets Sichtkontakt notwendig und die Mobiltelefone dürfen nicht viel mehr als etwa 20 cm voneinander entfernt sein. Ein weiteres Manko ist, dass ein QR-Code im Vergleich nur relativ geringe Datenmengen übermitteln kann. In einem maximal 177 x 177 Pixel großen QR-Code können

Abbildung 1: QRCode mit Zeichenkette: QR Code
Quelle: http://upload.wikimedia.org/wikipedia/commons/8/87/QRCode.png (26.07.13)

etwa 2.953 Byte enthalten sein, was die Übertragung von Dateien quasi ausschließt. Diese Einschränkungen schlagen sich deutlich auf die Bewertung des QR-Codes nieder.

### WLAN Infrastruktur

WLAN steht für Wireless Local Area Network. Es handelt sich dabei um eine kabellose Lösung für ein Lokales Netzwerk. Hier wird eine Infrastruktur aufgebaut, in der sich alle Clients mit einem WLAN Access Point verbinden. Jegliche Kommunikation läuft über ein lokales Netzwerk ab und zwingend über den WLAN Access Point.

Überzeugen kann diese Lösung durch eine gute Reichweite und eine sehr gute Übertragungsrate. Die Verfügbarkeit von Access Points ist jedoch leider viel zu gering und keineswegs allgegenwärtig. Da ein Access Point zwingend vorhanden sein muss und auch die meisten öffentlichen Access Point eine direkte Kommunikation zwischen den Clienten unterbinden, ist diese Technologie leider nicht praktikabel. Durch die erheblichen Nachteile bekommt es eine Bewertung von 45 Punkten.

### Wi-Fi Direct

Einen der großen Nachteile von WLAN löst Wi-Fi Direct. [16] Geräten, die diesen Standard unterstützen, ist es möglich sich, per WLAN direkt ohne Access Point zu verbinden. Dabei emuliert mindestens eines der beiden Geräte das Verhalten des Access Point per Software. Somit ist es sogar möglich, ganze Netzwerke über Wi-Fi Direct aufzubauen.

Es verfügt ebenfalls über eine gute Reichweite und eine sehr gute Übertragungsrate. Leider ist es derzeit noch auf nicht allzu vielen Smartphone vertreten und bietet sonst ebenfalls wenige positive Aspekte, weshalb es mit 50 Punkten nur eine geringfügig bessere Bewertung als WLAN Infrastruktur erhält.

### 3G Internet

Eine weitere Technologie ist die Verwendung einer mobilen Internetverbindung. Diese läuft im Allgemeinen für den Anwender transparent über verschiedene Standards ab. Die heute gängigen Standards sind GPRS, EDGE (2G), UMTS (3G), HSDPA/HSUPA (3G+), LTE (4G). Je nach Standard variiert hier auch die Geschwindigkeit extrem. Während z.B. bei schlechtem Netz und über GPRS gerade einmal 55 kbit/s möglich sind, ermöglicht das neuste LTE bei verfügbarem Netz eine sagenhafte Übertragungsrate von bis zu 300 Mbit/s. Dabei handelt es sich um das über 5000 fache der Geschwindigkeit bei GPRS Empfang. Auch wenn die Netzabdeckung zumindest von GPRS und EDGE mittlerweile quasi flächendeckend ist, kann es in Gebäuden oder an abgeschirmten Gebieten dazu kommen, dass kein Netz verfügbar ist. Ebenfalls fallen heutzutage noch immer nicht ganz unerhebliche Kosten für die Verwendung eines mobilen Internetanschlusses an, welches einen weiteren Nachteil darstellt. Da es sich bei 3G nicht um eine direkte Ende zu Ende Verbindung handelt, muss man sich hier weiterer Techniken bzw. Dienste bedienen, welche den Implementierungsaufwand und eventuell auch die Benutzerfreundlichkeit beeinträchtigen. Im Großen und Ganzen schneidet 3G nicht negativ ab, allerdings kann es auch nur in wenigen Punkten überzeugen. Dies führt zu einer moderaten Bewertung von nur 70 Punkten.

## 3.3 Bedienungsstudie

Die zu entwickelnde App Capood erfordert ein hohes Maß an Benutzerfreundlichkeit. Dafür ist ein intuitives Bedienungskonzept essentiell. Im Rahmen des konzeptionellen Entwurfes wurden je drei Szenarien für die Sender- und die Empfängerseite entwickelt und anschließend gegenübergestellt. Ähnlich zur Technologiestudie wurden Kriterien ausgesucht, die eine objektive Bewertung der Bedienkonzepte zulassen. Die Kriterien wurden mit Prozentzahlen priorisiert.

- Usability 50%

  Eine gute Usability steht für eine intuitive Bedienung. Das Konzept erfüllt gängige Anforderungen der Dialoggestaltung.

- Sicherheit 30%

  Die mit der App zu verarbeitenden Inhalte können von privater Natur sein. Ein verantwortungsvoller Umgang ist deshalb wichtig und Daten dürfen nur auf expliziten Befehl des Anwenders verschickt werden.

- Ressourcenverbauch 10%

Zur gegenwärtigen Zeit werden Smartphones immer leistungsfähiger. Die damit verbundene erhöhte Leistungsaufnahme lässt die Akkulaufzeit sinken. Es ist daher wichtig, dass die Applikation keine unnötige Last erzeugt, die die Akkulaufzeit zusätzlich schmälert.

- Implementierungsaufwand 10%

  Der Implementierungsaufwand wurde vom Entwicklerteam subjektiv abgeschätzt. Eine niedrigere Bewertung ist zu bevorzugen.

### Konzepte der Senderseite

- Händisches Copy & Paste

  Der Anwender kopiert den Text, den er versenden möchte. Anschließend wird die App Capood gestartet. Der Text wird in eine Sendemaske eingefügt und anschließend wird das Empfängergerät ausgewählt. Im Anschluss startet der Sendevorgang. Dieses Konzept steht für volle Kontrolle durch den Nutzer. Die Usability ist vergleichsweise schlecht. Außerdem wird die App nur dann laufen, wenn sie auch explizit genutzt wird. Das Resultat ist ein geringer Ressourcenverbrauch.

- Copy - Benachrichtigung

  Der Anwender kopiert den Text, den er versenden möchte. Anschließend wird eine Benachrichtigung erstellt, die der Anwender über die Benachrichtigungsleiste (Android) aufrufen oder verwerfen kann. Sollte die Benachrichtigung aufgerufen werden, öffnet sich die App mit dem eingefügten Inhalt der Zwischenablage. Es muss nur ein Empfängergerät gewählt werden. Dann startet der Sendevorgang. Bei diesem Konzept hat der Nutzer die volle Kontrolle gepaart mit einer verbesserten Usability gegenüber dem ersten Konzept. Der Ressourcenverbrauch ist erhöht, da die Zwischenablage im Hintergrund überwacht werden muss.

- automatische Synchronisation

  Der Anwender kopiert den Text, den er versenden möchte. Anschließend wird der Inhalt der Zwischenablage automatisch mit allen empfangsbereiten und mit dem Sender gepaarten Geräten geteilt. Bei diesem Konzept ist die App Capood für den Nutzer praktisch unsichtbar. Die Usability ist sehr hoch, weil die App automatisch agiert und der Nutzer nur noch intuitiv den zu versenden Text kopieren muss. Allerdings fehlt jegliche Kontrolle, wer den kopierten Inhalt empfangen wird. Die Sicherheit ist damit stark geschwächt. Auch hier muss im Hintergrund die Zwischenablage überwacht werden.

| Kriterium | Gewichtung | | händisches Copy & Paste | Copy - Benachrichtung - Paste | automatische Synchronisation |
|---|---|---|---|---|---|
| Usability | 50% | | | | |
| Sicherheit | 30% | | vollständige Kontrolle durch Nutzer | | |
| Ressourcenverbrauch | 10% | | | | |
| Implementierungsaufwand | 10% | | | | |

Tabelle 2: Bedienungskonzepte Senderseite

Nach der Gegenüberstellung der drei Konzepte dominiert das Konzept „Copy - Benachrichtigung“ (siehe auch Tabelle 2). Der Versand von Dateien wird an dieser Stelle getrennt behandelt, weil die Zwischenablage des Android-Betriebssystems nicht mit Dateien funktioniert.

Der Versand von Dateien soll über die Teilen-Funktion realisiert werden. Sobald eine Datei aus einer anderen App heraus geteilt wird, kann Capood als App zum Versenden ausgewählt werden. Anschließend öffnet sich eine Ansicht, die wichtige Daten der Datei anzeigt und die Möglichkeit bietet ein Empfangsgerät zu wählen. Nach Auswahl des Empfängers wird eine Anfrage verschickt, ob dieser die Datei empfangen möchte.

### Konzepte der Empfängerseite

- Empfangen nur bei geöffneter App Capood

  Daten können nur empfangen werden, wenn der Anwender Capood geöffnet hat. Diese Variante überzeugt bei der Sicherheit, weil das Gerät keine unerwünschten Daten im Hintergrund empfangen kann. Der Implementierungsaufwand ist am geringsten und der Ressourcenverbrauch ist auch gering, weil die App nur dann geöffnet ist, wenn sie gebraucht wird. Allerdings ist die Usability schlecht, weil der Nutzer zum Empfang die App explizit öffnen muss.

- Benachrichtigung über verfügbare Nachricht

  Sobald eine Nachricht beim Empfänger eingeht, wird eine Benachrichtigung in der Benachrichtungleiste angezeigt. Sobald diese Benachrichtigung ausgewählt wird, öffnet sich Capood und präsentiert die Nachricht. Bei einer Textnachricht wird der

| Kriterium | Gewichtung | | Darstellung der empfangenen Daten NUR bei offener Applikation | Benachtigung über verfügbare neue Daten | In die Zwischen/Datei-ablage kopieren |
|---|---|---|---|---|---|
| Usability | 50% | | | | |
| Sicherheit | 30% | | | | |
| Ressourcenverbrauch | 10% | | | | |
| Implementierungsaufwand | 10% | | | | |

Tabelle 3: Bedienungskonzepte Empfängerseite

Inhalt dargestellt und über eine Schaltfläche kann dieser in die Zwischenablage kopiert werden. Sollte es sich um einen Dateitransfer handeln, wird der Anwender gefragt, ob er die Datei empfangen möchte. Nach positiver Bestätigung wird die Datei vom Sender geschickt und auf dem Empfängergerät gespeichert. Es wird die größtmögliche Kontrolle bei guter Usability gewährt. Im Hintergrund muss ständig überwacht werden, ob eine Nachricht an das Gerät versandt wurde. Das kostet wertvolle Systemressourcen und Akkulaufzeit.

- Automatisches Empfangen von Dateien

  Eine vom Sender verschickte Datei wird automatisch empfangen und in das Dateiverzeichnis abgespeichert. Der Nutzer hat keine Kontrolle, welche Datei er von wem und wann empfängt. Auch hier muss der Bluetoothadapter permanent überwacht werden, was zusätzliche Systemlast erzeugen kann.

Auf Empfängerseite überzeugt analog zur Senderseite das Konzept mit Benachrichtigungen „Benachrichtigung über verfügbare Nachricht“ (siehe Tabelle 3).

## 3.4 Releaseplanung

Bei der Softwareentwicklung existieren mehrere Vorgehensmodelle, die die Art der Entwicklung für das gesamte Team festlegen. Bei den Modellen unterscheidet man zwischen sequentiell („Wasserfall“), iterativ („spiralförmig“) und leichtgewichtig („agil“).

Für die Entwicklung der Applikation „Capood“ wurde eine agile Methode gewählt, die den Entwicklern größere Freiheiten einräumt. Das charakteristische an dieser Vorgehens-

| Version | Tasks |
|---|---|
| 0.1 | Pairing zwischen 2 Android-Geräten aus der Applikation heraus |
| 0.2 | Aus der Applikation heraus simples Verschicken und Empfangen von Texten |
| 0.3 | Hintergrunddienst zum Überwachen der Zwischenablage |
| 0.4 | Hintergrunddienst zum Überwachen von eingehenden Nachrichten |
| 0.5 | Benachrichtigung über Kopiervorgang zum Öffnen der Applikation |
| 0.6 | Benachrichtigung und Bestätigung über Empfang einer Nachricht |
| 0.7 | Unterstützung von Dateiversand und –empfang |
| 0.8 | Kontextmenü (Öffnen mit..) mit der Applikation verbinden |
| 0.9 | Dateiverwaltung am Empfangsgerät |
| 1.0 | Interpretation des Nachrichtentyps am Empfangsgerät (Link, Bild, Lied; etc..) |

Tabelle 4: Releaseplanung 1.0

weise ist eine kontinuierliche Anpassung der Änderungen (Continious Integration), welche zu jedem Zeitpunkt der Entwicklung eine funktionsfähige Applikation beinhaltet. [23]

Dennoch spielt bei der agilen Vorgehensweise die iterative Entwicklung eine Rolle. Dabei wird die gesamte Entwicklung in mehrere Teile (Iterationen) aufgespalten, deren Funktionalität nach und nach zu der Gesamtapplikation hinzugefügt wird. Nach jeder einzelnen Iteration gibt es eine neue Version der entwickelten Software, die diese um die in der letzten Iteration beschriebenen Funktionalitäten erweitert.

Am Ende ensteht ein fertiges Release, welches meistens die Markteinführung oder Veröffentlichung der Applikation mit sich zieht.

Für das fertige Release der Capood Applikation gab es 10 Iterationsschritte: von Version 0.1 bis 1.0 (siehe Tabelle 4).

Die Entwicklung der Applikation sollte sich in den Versionen 0.1 bis 0.6 nur auf das Versenden von Textnachrichten beziehen.

Zu Beginn der Entwicklung sollte zunächst nur eine erfolgreiche Bluetoothverbindung hergestellt werden. Sobald diese Verbindung erfolgreich hergestellt wurde, sollte das System in der Version 0.2 Textnachrichten versenden können, um die wichtigste Funktionalität am Anfang zu implementieren.

Die weiteren vier Versionen bezogen sich auf Hintergrunddienste und Benachrichtigungen für den Nutzer, um das intuitive Copy&Paste Konzept erfolgreich in die Applikation einzubinden. Erst ab Version 0.7 sollte auch der Dateiversand ermöglicht werden. Die weiteren Versionen befassten sich mit der Verwaltung der Dateien und dem automatischen Erkennen des empfangenen Dateityps.Dies wird von Android teilweise selbst durch das

| Version | Tasks |
|---|---|
| 1.1 | Weitere Übertragungstechniken (NFC, Ad-hoc WLAN, etc..) |
| 1.2 | Mehrsprachigkeit |

Tabelle 5: Releaseplanung 2.0

Betriebssystem bereitgestellt, indem es je nach Datentyp ein Kontextmenü mit einer Liste von Anwendungen öffnet, welche mit dem Typ umgehen können und somit auch als Standard für den Datentyp angegeben werden können. Damit sollten alle wichtigen Funktionen implementiert worden sein und die Applikation könnte in der Version 1.0 veröffentlicht werden.

Weitere Releaseschritte wurden bereits in Tabelle 5 definiert und im Abschnitt „Fazit & Ausblick“ auf Seite 34 weiter ausgeführt.

# 4 Technischer Entwurf & Spezifikation

## 4.1 Technologien

Nach den durchgeführten Studien zum Thema Betriebssystem und Übertragungstechnologie war es möglich, die vorhandenen Ressourcen des Entwicklerteams zu ermitteln. Zur Verfügung standen zwei Android-Smartphones und ein Android-Tablet. Außerdem ist es möglich ein Androidbetriebssystem virtualisiert auf einem herkömmlichen Windows-PC zu betreiben. Das Projekt „Android-x86“ widmet sich der Portierung von Android auf die x86-Architektur. [18]

Als Entwicklungsumgebung bietet Android das Android SDK an. Dieses arbeitet zusammen mit Eclipse. Ein komplettes Softwarepaket lässt sich herunterladen und beinhaltet ein Version von Eclipse mit integrierten Plugins für die Entwicklung von Android-Apps.

Der verfügbare Android-Emulator ist untauglich für die Entwicklung einer App, die Bluetooth verwenden soll, da der Emulator nicht in der Lage ist, eine Bluetoothverbindung bereitzustellen. Außerdem zeigt die Erfahrung, dass der Emulator sehr viel länger für den Bootvorgang des Android-Systems benötigt als ein virtualisiertes Android-x86 auf dem selben Computer. Sowohl Smartphones mit Bluetoothschnittstelle als auch Laptops mit virtualisiertem Android-x86 und Bluetoothschnittstelle sind geeignet für die Entwicklung von Capood.

Ein weiterer wichtiger Bestandteil des Android SDK ist die Android Debug Bridge (ADB). Sie verbindet Androidgeräte und die Entwicklungsumgebung Eclipse (bzw. dem ADT-Plugin). Zum Funktionsumfang gehören Datentransfer, Steuerung der Android-Geräte durch Befehle und auch Logausgaben für Eclipse - Logcat genannt. [1, S. 102] Reale Androidgeräte benötigen einen USB-Treiber, damit sie von Eclipse erkannt werden. Bei virtualisierten Androidgeräten sind bestimmte Netzwerkeinstellungen zu treffen und außerdem muss nach jedem Start von Eclipse erneut die Verknüpfung durch ein Kommandozeilenbefehl hergestellt werden. [3] Dieser Vorgang kann durch eine Batch-Datei vereinfacht werden.

Zur elektronischen Kommunikation und Organisation des Projektteams wird auf Codeplex zurückgegriffen. Codeplex ist eine von Microsoft bereitgestellt Projektmanagement-Plattform. Pro Projekt wird ein Wiki zur Dokumentation, eine Versionsverwaltung, ein Forum, einen Downloadbereich und ein Ticketsystem für Probleme (Issues) zur Verfügung gestellt.

Für die Versionsverwaltung werden mehrere Varianten bereitgestellt. Für das Projekt Capood wird ein Teamfoundationserver (TFS) verwendet. Es kann gleichzeitig auch Subversion verwendet werden, um mit dem TFS zu kommunizieren.

## 4.2 Architektur

Zu Beginn des Technischen Entwurfes wurden UML-Diagramme erstellt, die die Architektur der Applikation darstellen sollten (siehe Abbildung 2 auf Seite 19).

UML bedeutet Unified Modeling Language und ist eine Sprache zur Visualisierung und Konstruktion von Modellen einer Software oder Anwendung vor ihrer Umsetzung. Diese Sprache bietet Entwicklern eine einheitliche Entwurfsbasis zur gemeinsamen Diskussion über das Aussehen der zu entwickelnden Software. Diese Sprache wurde Ende der 90er Jahre entwickelt und gilt heute mittlerweile mit der Version 2.4 als allgemeiner Standard.

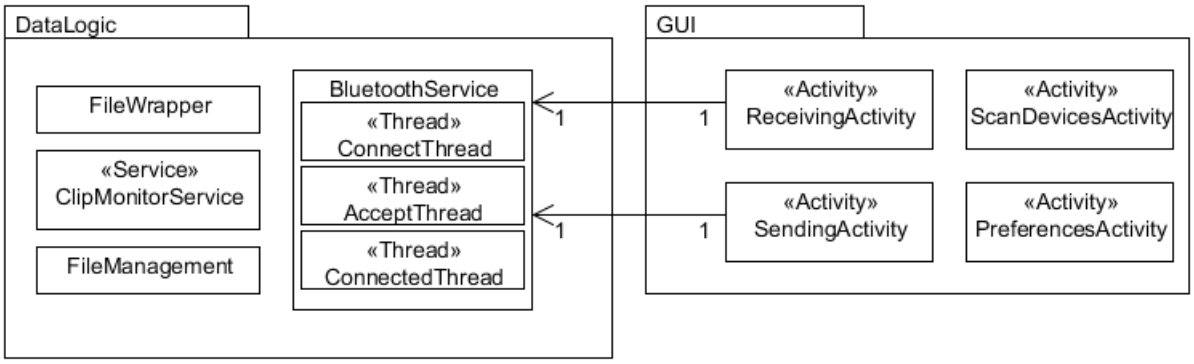


Abbildung 2: vereinfachtes Klassendiagramm (Entwurfsphase)

Quelle: eigene Darstellung (erstellt mit UMLet)

Bei der Modellierung kreiert man Diagramme, die verschiedene Sachverhalte der Software darstellen. Die bekanntesten sind Klassen-, Aktivitäts- und Anwendungsfalldiagramme. Das Klassendiagramm zeigt ein vollständiges Abbild des Systems: dessen Klassen, Methoden und Variablen. Das Aktivitätsdiagramm beschreibt das Verhalten der Software als eine Menge von elementaren Aktionen, zwischen welchen Datenflüsse existieren.

Anwendungsfalldiagramme beschreiben die Anforderungen an das System, welche Aufgaben dieses im Elementaren- und Gesamtkontext zu erfüllen hat. [17]

Für die Modellierung solcher Diagramme gibt es eine Vielzahl von Programmen, die den vollen Funktionsumfang von UML 2.4 abdecken.

Für unsere Modellierung haben wir das Open-Source- Tool UMLet in der Version 12.0 verwendet. [15] Es ist ein einfaches, Java und xml-basiertes Tool, welches auf simple Art dem Anwender eine persönlich angepasste Modellierung der gewünschten Diagramme ermöglicht und gleichzeitig eine Vielzahl von Vorlagen bietet. Ein Export als Bild ist ebenfalls ohne Probleme möglich. Ein Export als Programmcode wird nicht unterstützt, war für das Projekt aber auch nicht zwingend erforderlich.

Für unsere Modellierung der Architektur hatten wir uns nur auf das Erstellen eines Aktivitätsdiagramms für das Senden und Empfangen der Nachrichten und das Klassendiagramm beschränkt.

### Sendevorgang (Abbildung 3)

Nach dem Starten der Applikation muss das System den Status des BluetoothServices überprüfen. Dieser muss aktiv sein oder - bei Nichtaktivität - durch den Benutzer oder automatisch durch die Applikation selber - aktiviert werden. Der BluetoothService beinhaltet alle für die Bluetoothumgebung relevanten Funktionen.

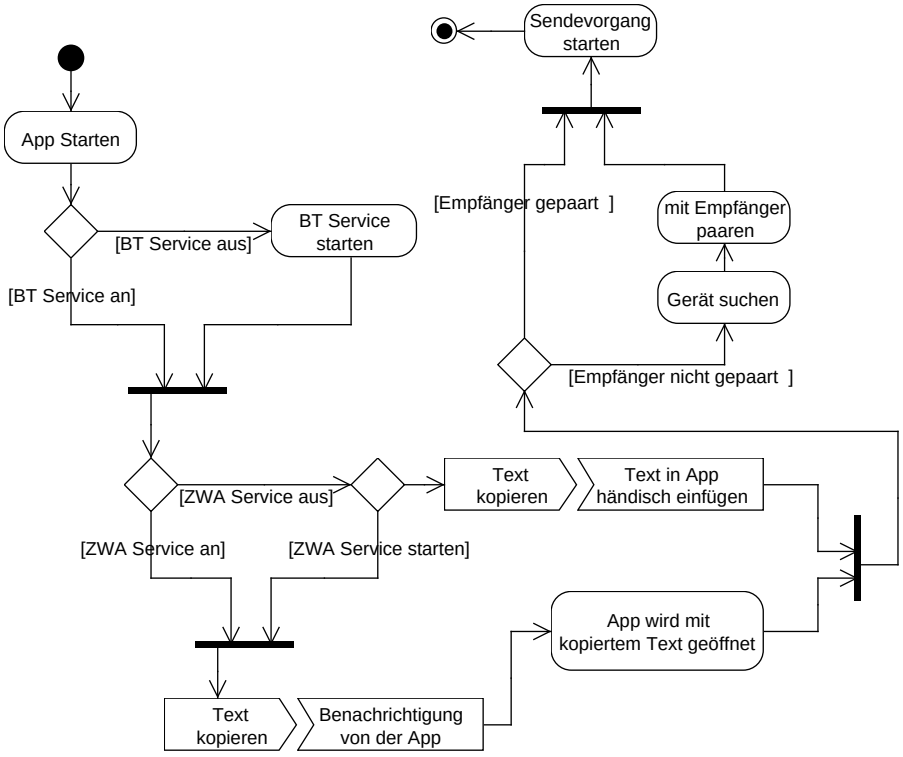


Abbildung 3: Aktivitätsdiagramm für den Sendevorgang

Quelle: eigene Darstellung

Nach der erfolgreichen Aktivierung des BluetoothService muss das System dem Nutzer die Möglichkeit geben, den Service der Zwischenablage zu aktivieren (ClipBoardService). Läuft der ClipBoardService nicht, so ist der Anwender gezwungen den Text außerhalb der Applikation zu kopieren und händisch in die Applikation wieder einzufügen. Ist der ClipBoardService aktiv bzw. wird durch den Anwender aktiviert, wird die Zwischenablage durch diesen überwacht, erkennt den Kopiervorgang und zeigt eine Benachrichtigung an,

durch welche die Applikation automatisch mit dem kopierten Text aus der Zwischenablage geöffnet wird.

Danach muss das System die Möglichkeit bieten einen Empfänger für die Nachricht auszuwählen: Man unterscheidet dabei zwischen im System gespeicherten, gepaarten und neuen Geräten. Existiert noch keine Paarung, muss die Anwendung dem Nutzer die Möglichkeit bereitstellen, neue Geräte zu suchen und sich mit diesen zu paaren. Nach erfolgreicher Auswahl des Empfängers muss das System den Sendevorgang starten und erfolgreich ausführen. Danach ist der Sendevorgang beendet.

### Empfangsvorgang (Abbildung 4)

Das System prüft zunächst, ob eine aktive Paarung der beiden Geräte existiert. Ist es nicht der Fall, bekommt der Nutzer zunächst eine Paarungsanfrage, die dieser in der Applikation akzeptieren muss. Ist eine erfolgreiche Verbindung hergestellt, muss das System den Nutzer über eine empfangene Nachricht benachrichtigen. Durch die Auswahl der Benachrichtigung wird die Applikation geöffnet und die empfangene Nachricht angezeigt.

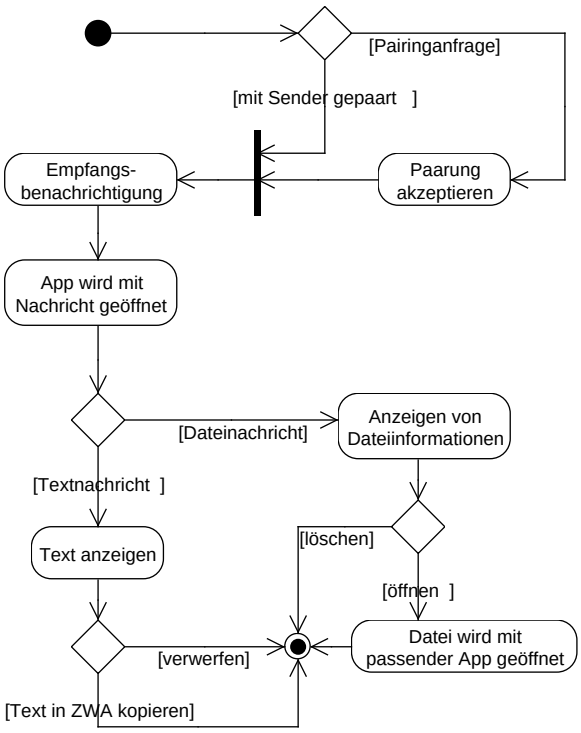


Abbildung 4: Aktivitätsdiagramm für das Empfangen von Nachrichten

Quelle: eigene Darstellung

Hier muss das System den Nachrichtentyp erkennen und eine Auswahl über die weiteren möglichen Aktionen treffen. Bei einer Textnachricht muss das System den empfangenen Text anzeigen und dem Nutzer die Möglichkeit geben, diesen zu verwerfen oder direkt in die eigene Zwischenablage zu kopieren. Bei einer Dateinachricht muss das System die Dateiinformationen anzeigen und dem Nutzer die Möglichkeit geben, die Nachricht mit dem Standardprogramm für den jeweiligen Dateitypen zu öffnen oder die Datei zu löschen. Nach beiden möglichen Aktionen ist der eigentliche Sendevorgang beendet.

Das Androidbetriebssystem verwendet sog. Acitvities. Eine Activity (dt. Aktivität) soll genau eine Aktivität repräsentieren, die der Anwender dieser Software durchführen kann. In der Praxis heißt dies, dass eine Activity als GUI-Element ein Fenster ist, das dem Anwender eine Schnittstelle zur der entsprechenden Funktionalität der Acitvity bietet. [1, S. 281]

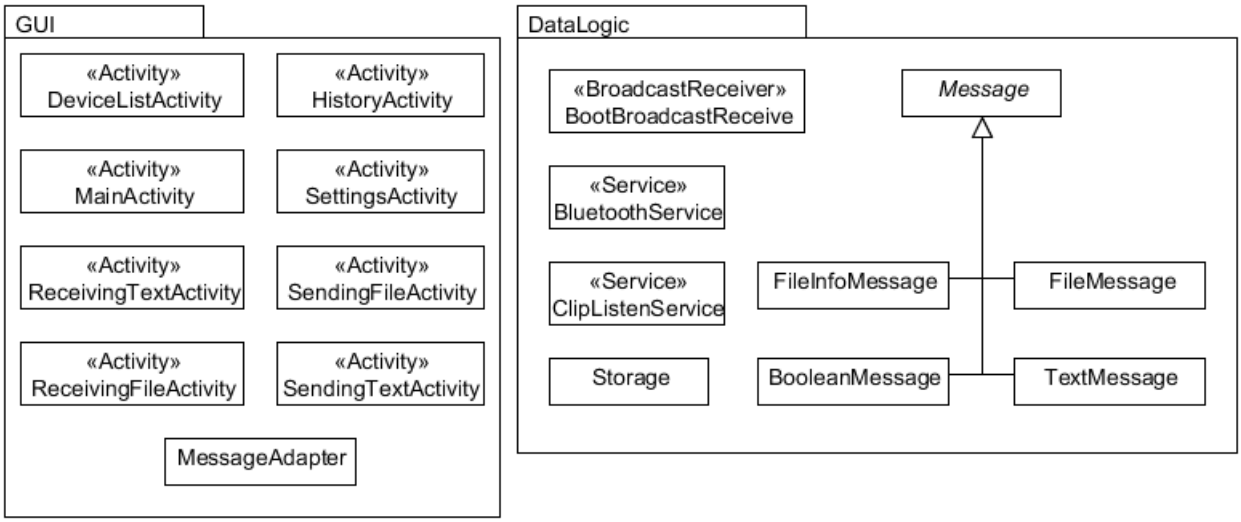


Abbildung 5: vereinfachtes Klassendiagramm (aktueller Stand)

Quelle: eigene Darstellung (erstellt mit UMLet)

Bei der App Capood sind die Klassen in zwei Pakete aufgeteilt. Ein Paket bildet die Datenlogik und das zweite die GUI.

Das GUI-Paket enthält alle Activities. Es gibt Activities für das Senden und Empfangen jeweils von Textinhalten und Dateien, zur Auflistung der empfangenen Nachrichten, zur Suche von Empfängergeräten, für persönliche Einstellungen (Optionsmenü) und das Hauptmenü. Das Datenlogik-Paket beinhaltet Klassen, deren Aufgaben das Verwalten der Daten und Dateien und der Bluetooth-Verbindung zwischen zwei Geräten sind.

Im Folgenden werden die Funktionen aller wichtigen Klassen einzeln erläutert. Diese sind auch im aktuellen Klassendiagramm (siehe Abbildung 5 auf Seite 22) einsehbar.

### Storage

Die Storage-Klasse verwaltet die empfangenen Nachrichten. Textnachrichten und Dateinachrichten werden persistent und serialisiert in einer Textdatei gespeichert. Diese liegt im internen Speicherbereich der App, sodass niemand anderes (Anwender oder App) berechtigt ist, auf diese Datei zuzugreifen.

### Message

Die abstrakte Message-Klasse bildet die Basis für alle versandfähigen Nachrichten. Eine Message hat immer ein Datum und einen Typ (realisiert als Enum).

### TextMessage

Die TextMessage-Klasse dient zum Versand von Textnachrichten.

### FileInfoMessage

Die FileInfoMessage-Klasse stellt Informationen zu einer Datei bereit. Sie wird vor dem eigentlichen Versand einer Datei verschickt. Der Empfänger kann die Informationen der Datei einsehen und diese dann nach Bedarf annehmen oder ablehnen.

### BooleanMessage

Die BooleanMessage-Klasse dient zum Annehmen oder Ablehnen einer Datei.

### FileMessage

Die FileMessage-Klasse ist ein Wrapper für eine Datei. Die eine Instanz dieser Klasse enthält ein byte-Array mit den Bytes der repräsentierten Datei.

### BluetoothService

Der BluetoothService repräsentiert die Buetoothverbindung. Er erbt von der Klasse Service und wird auf Wunsch den Nutzer im Hintergrund gestartet. Wenn der Bluetoothservice aktiv ist und der Bluetoothadapter des Gerätes eingeschaltet ist, dann wartet der BluetoothService permanent auf eine eingehende Bluetoothverbindung für die App Capood (AcceptThread). Gleichzeitig wird über ihn auch die aktive Verbindungsaufnahme zu einem Empfänger gestartet (ConnectThread). Nach erfolgreichem Verbindungsaufbau

haben sowohl Sender als auch Empfänger einen ConnectedThread, der die laufende Verbindung repräsentiert. Jedes Gerät hat Input- und Outputstreams des Verbindungspartners. Über die Klassen ObjectInputStream und ObjectOutputStream werden die Nachrichten als Message-Objekte serialisiert übertragen. Der BluetoothService basiert auf der Klasse BluetoothChatService aus dem Beispielprojekt BluetoothChat, welches von Android bereitgestellt wird.

### ClipListenService

Der ClipListenService ist ebenfall ein Service, der auf Wunsch des Users im Hintergrund läuft. Er überwacht die Zwischenablage (Interface OnPrimaryClipChangedListener). Sobald der Anwender einen neuer Text in die Zwischenablage kopiert, wird vom ClipListenService eine Benachrichtigung erstellt. Diese zeigt dem Anwender, dass ein Text in die Zwischenablage kopiert wurde. Nach dem Öffnen der Benachrichtung öffnet sich die SendingTextActivity. Der Inhalt der Zwischenablage wird automatisch in das Textnachrichtenfeld eingefügt.

### DeviceListActivity

Die DeviceListActivity wurde ohne Veränderungen aus dem Beispielprojekt Bluetoothchat übernommen. Sie dient zur Suche nach Bluetoothgeräten. Nach dem Aufruf wird sie als Popup über der aktiven Activity angezeigt und listet alle schon gepaarten Geräten auf. Der Nutzer hat die Möglichkeit weitere Geräte zu suchen.

### SendingTextActivity

Die SendingTextActivity ist die Nutzerschnittstelle zum Versenden von Textinhalten. Ein großes EditText-Feld bietet die Möglichkeit zur Texteingabe. Die maximale Zeichenanzahl ist auf 1000 Zeichen begrenzt und wird auch grafisch unterhalb des Eingabefeldes angezeigt. Sie ist durch die Konstante MAX_CHARS gegebenenfalls auch mit wenig Aufwand anpassbar.

### SendingFileActivity

Die SendingFileActivity repräsentiert das Versenden von Dateien. Sie kann nicht direkt vom Nutzer ausgerufen werden. Der Aufruf erfolgt über Intentfilter zu der systemeigenen Sendeaktion (android.intent.action.SEND) für die MIME-Typen (Multipurpose Internet Mail Extensions) für Bilder, Videos, Audio und Anwendungen. Der Intentfilter wird in der Androidmanifest-Datei definiert.

### ReceivingTextActivity

Die ReceivingTextActivity wird immer dann geöffnet, wenn ein Textmessage empfangen wird und die daraus resultierende Benachrichtigung angewählt wird. Der Textinhalt der Nachricht wird dem Nutzer präsentiert. Er kann außerdem über einen Button den Inhalt in die Zwischenablage kopieren.

### ReceivingFileActivity

Die ReceivingFileActivity wird immer dann geöffnet, wenn eine FileInfoMessage empfangen wird und die daraus resultierende Benachrichtigung angewählt wird. Dem Nutzer wird zuerst ein Dialog anzeigt mit der Frage, ob der Nutzer die durch die FileInfoMessage repräsentierte Datei empfangen möchte. Sollte der Anwender dies bestätigen, wird der Dateiversand durchgeführt. Mit einem Button kann die Datei, sofern sie existiert, geöffnet werden. Dies wird mit der Systemaktion zum Öffnen (android.content.Intent.ACTION_VIEW) realisiert. Anschließend wird die Datei mit dem Standardprogramm für den entsprechenden Dateitypen geöffnet.

### HistoryActivity

Die HistoryActivity zeigt alle empfangen Nachrichten sortiert nach Datum in einer ListView an. Dafür wurde eine eigene Adapterklasse (MessageAdapter) implementiert. Eine angewählte Nachricht kann über einen Button gelöscht werden. Durch längeren Druck am Touchscreen auf eine Nachricht wird diese geöffnet.

### SettingsActivity

Die SettingsActivity bietet über zwei Checkboxen die Möglichkeit, den ClipListenService und den BluetoothService an- und auszuschalten.

## 4.3 Design

Die Zielvereinbarung über das Projekt sieht einen Prototypen mit pragmatischem funktionalen Design vor. Das Design wurde zuerst auf Papier von Hand gezeichnet. Es wurden allen wichtigen Activities skizziert (siehe Abbildung 6 auf Seite 26).

Anschließend wurde das Design über die Entwicklungsumgebung Eclipse mit dem vorhandenen GUI-Builder für Android nachgebaut (siehe Abbildung 8 auf Seite 27). Der Editor arbeitet nach dem WYSIWYG-Prinzip. Da die grafischen Oberflächen auf XML-Dateien basieren, hat der Programmierer auch die Möglichkeit, direkt die XML-Datei zu

verändern und auf diese Weise eine umfangreichere Kontrolle zu gewinnen (siehe Abbildung 7 auf Seite 26).

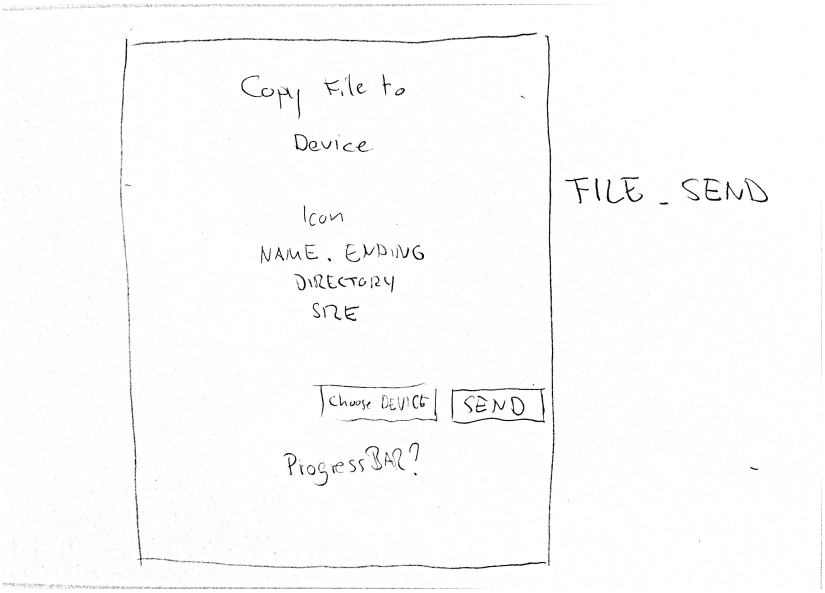


Abbildung 6: Handgezeichnete Skizze einer Activity

Quelle: eigene Darstellung

Wichtig war es, die richtige Layoutvariante zu verwenden, da das Gesamtlayout sonst abhängig von der Displayauflösung des Smartphones nicht mitwächst. Um eine flexible Oberfläche zu bauen, eignet sich das RelativeLayout. Es ordnet die einzelnen Oberflächenelemente in Beziehungen untereinander an. Das komplizierte Verschachteln von Layouts ist nicht nötig.

activity_sending_text.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".SendingTextActivity" >

    <TextView
        android:id="@+id/textView_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <ProgressBar
        android:id="@+id/progressBar_sending"
        style="?android:attr/progressBarStyleSmall"
        android:layout_width="wrap_content"
```

Graphical Layout | activity_sending_text.xml

Abbildung 7: XML-Editor für Layouts

Quelle: Screenshot aus Eclipse

Das Design beschränkt sich hauptsächlich auf Standardbausteine wie EditText, TextView, ProgressBar, CheckBox und Button.

Eine Besonderheit ist die ListView in der HistoryActivity zur Anzeige der bereits empfangenen Nachrichten. Hier kommt ein selbst entwickeltes Layout (history_item.xml) für das Listenelement zum Einsatz. Es zeigt jeweils ein kleines Symbol auf der rechten Seite zur Visualisierung des Nachrichtentypes - entweder eine Datei- oder eine Textnachricht. Anschließend folgt eine TextView, die den Inhalt der toString()-Methode der jeweils repräsentierten Nachricht ausgibt. Die Verknüpfung von Datenlogik und grafischer Oberfläche erfolgt hier durch die selbst implementierte Adapterklasse MessageAdapter (erbend von der Klasse ArrayAdapter).

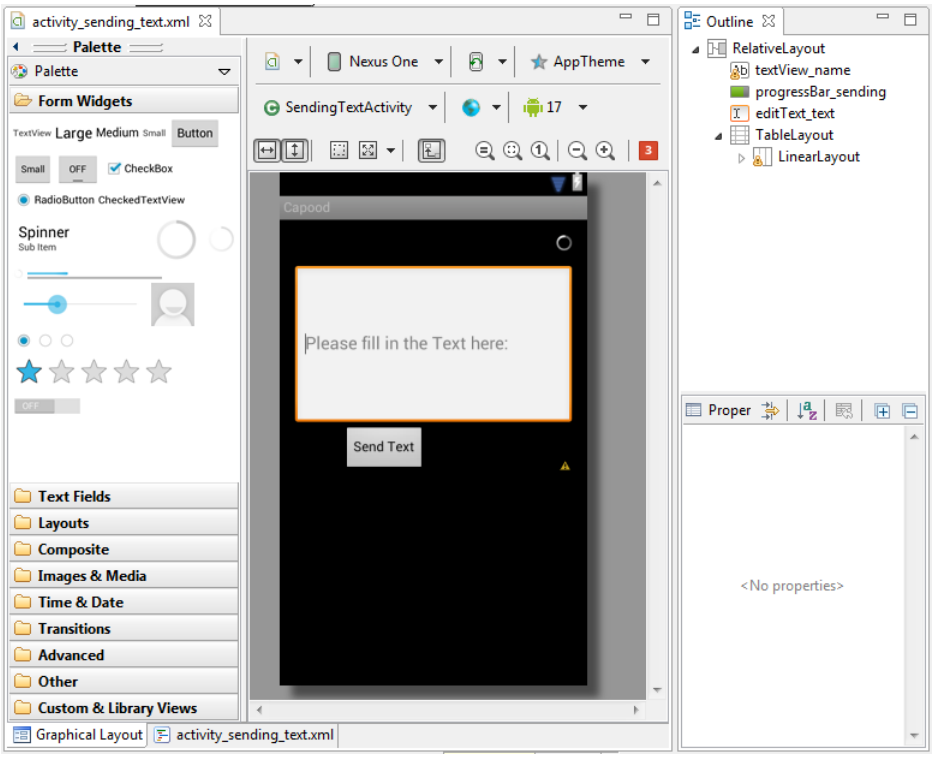


Abbildung 8: GUI-Builder des Android SDK

Quelle: Screenshot aus Eclipse

# 5 Implementierung

## 5.1 Besonderheiten

Android bietet eine eigene Klassenbibliothek, deren Klassen an das Betriebssystem und dessen Funktionalitäten angepasst sind.Bei der Implementierung der Applikation Capood wurde einige für Android spezifische Funktionen und Klassen genutzt. Im Folgenden werden diese Besonderheiten und deren Funktion innerhalb von Capood näher beschrieben:

### SharedPreferences

SharedPreferences bieten die Möglichkeit, Variablen persistent und klassen- oder applikationsübergreifend zu speichern. [8] Auf diese Wiese lassen sich zum Beispiel vom Nutzer vorgenommene Einstellungen nach dem Beenden der App und über einen Neustart des Systems hinweg sichern. Zum Einsatz kommt diese Technik zum Speichern des Zustandes der im Hintergrund laufenden Services. Die App weiß, ob der Nutzer die Services laufen lassen möchte oder nicht und kann diese gegebenenfalls nach einem Neustart des Systems wieder neu starten.

### Service

Die abstrakte Service-Klasse wird zum Realisieren von im Hintergrund agierenden Diensten genutzt. [7] Ein Dienst kann gestartet und beendet werden. Außerdem kann ein Dienst eigenständig neu starten, wenn er zum Beispiel wegen Speichermangel gestoppt wurde, ohne dass dies vom Anwender explizit gefordert wurde. [6] Der BluetoothService und der ClipListenService erben von Service.

### Storage

Der persistente Speicher von Android gliedert sich in den externen und den internen Speicherbereich. Im externen Speicher können Dateien, ohne spezielle Zugriffsrechte und für jeden einsehbar, abgelegt werden. Im internen Speicherbereich kann nur der Root-Benuter im gesamten und eine Applikation in ihrem eigenen Bereich zugreifen. Der normale Anwender kann den internen Speicherbereich nicht einsehen.

Außerdem muss der externe Speicher nicht fest am Gerät installiert sein. Es kann also auch einfach kein externen Speicher vorhanden sein. In so einem Fall muss eine empfangene Datei im internen Speicher abgelegt werden. Ob der externe Speicher verwendbar ist, lässt sich über die Klasse Environment mit der Methode getExternalStorageState() ermitteln. [12]

### Intents und Broadcastreceiver

Intents sind Nachrichten zur Kommunikation zwischen den Komponenten innerhalb einer Anwendung und auch zwischen verschiedenen Anwendungen. Es handelt sich hierbei um ein asynchrones Kommunikationsmittel. [2, S. 30f]

Ein Broadcastreceiver (Interface) kann eine Intent empfangen und anschließend als Reaktion den Code seine onReceive() Methode ausführen. Dafür muss sich der entsprechende Receiver für eine bestimmte Intent-Art anmeldet. [2, S. 34f] Eine Intent kann sich durch sogenannte Action definieren lassen.

Auf diesem Benachrichtigungskonzept basiert die Kommunikation zwischen dem BluetoothService und den mit ihm kommunizierenden Activities. Außerdem werden Intents zum Start von Activities und Service verwendet.

### Benachrichtigungen

Zur Kommunikation mit dem Anwender besitzt Android eine Benachrichtigungsleiste (meistens am oberen Bildschirmrand). Eine Benachrichtung ist dort sichtbar als kleines Symbol. Intuitiv kann man die Benachrichtigungsleiste durch Wischen auf dem Bildschirm maximieren und dann die Details zu den Benachrichtigungen einsehen. [9]

Benachrichtigungen werden zuerst mit einem NotificationCampat.Builder gebaut und dann über den SystemService NotficationManager versandt. Durch Integrieren einer Intent lässt sich zum Beispiel eine Activity starten, wenn die Benachrichtigung angeklickt (angetippt) wird. [10]

Notifications in Capood zeigen dem Anwender an, dass er einen Text in die Zwischenanlage kopiert hat oder dass er eine neue Nachricht empfangen hat.

## 5.2 Fortschritt

Der erste Release fand zur Evaluation statt. Bis zu diesem Zeitpunkt wurde die Releasestufe 0.8 erreicht. Es wurden allerdings auch schon inhaltlich Teile der Stufe 1.0 erfüllt. Android selbst bietet die View Action. Diese wird verwendet, um Dateien anzuzeigen. Alle für diese Datei geeigneten Programme melden sich, sobald eine View Action per Intent ausgesendet wird. Der Anwender kann dann ein Programm auswählen und ggf. auch als Standardprogramm festlegen.

Für den vollständigen Release 1.0 fehlt allerdings eine Dateiverwaltung (0.9) von empfangenen Dateien. Dafür sollten zumindest triviale Basisfunktionalitäten wie das Löschen, das Umbennnen und das Verschieben von empfangen Dateien implementiert werden.

Nach dem erfolgreichen Release 1.0 folgen die Implementierung weiterer Übertragungstechniken und die Mehrsprachigkeit. Bis jetzt ist die grafische Oberfläche in englischer

Sprache realisiert worden. Dafür bietet Android die Möglichkeit sogenannte Resources zu verwenden, die in XML-Dateien angelegt werden. Ein Teil der Resources sind die Strings. Über String lässt sich die Mehrsprachigkeit einer App realisieren, indem man mehrere String-Sätze für verschiedene Sprachen anlegt. [13]

Die nächste Übertragungstechnik, die integriert werden sollte, ist NFC. Sie könnte, sofern Sender und Empfänger über NFC verfügen, die Usability verbessern. Außerdem erweitert Android in den neueren Releaeses (ab API Level 16) NFC um eine Funktionalität, bei der bei Bedarf auf andere Übertragungstechniken automatisch umgeschaltet wird, um die Übertragungsbandbreite zu maximieren. [11]

# 6 Evaluation

Der fertige Prototyp (Release 0.8) sollte auf festgelegte Fragestellungen evaluiert werden. Dies sollte unabhängig und anonym außerhalb des Entwicklungsteams stattfinden. Um somit möglichst viele Tester in die Evaluation einbeziehen zu können, wurde auf eine Online-Umfrage gesetzt.

Als Anbieter für die Online-Umfrage dient die Website www.haekchen.at. Es ist eine kostenlose Seite zu Umfragen- und Evaluierungszwecken, wo man ohne umfangreiche Registrierung und Vorkenntnisse leicht einen Umfragebogen erstellen kann. Jede Frage kann mit verschiedenen Beantwortungsvarianten versehen werden, wie zum Beispiel ein Ranking (ein bis fünf Sterne), Checkboxen für geschlossene Fragestellungen und offene Fragen mit freier Texteingabe. Außerdem bietet die Seite an, die Umfrage zu bearbeiten, die Reihenfolge der Fragen und auch ihre Antworten leicht abrufen zu können. Die Veröffentlichung erfolgt über einen Link, einem html-Code oder einem Post in die sozialen Netzwerke.


Abbildung 9: Ausschnitt aus dem Fragebogen

Quelle: Screenshot (http://www.haekchen.at/haekchen/fragebogen.asp?uid=23031&id=1 (29.07.13))

Der Fragebogen für Capood bestand auf zwölf Fragen mit einem Sterne-Ranking und einer offenenen Frage mit Freitext (siehe Abbildung 9). Dabei ging es um zwei hauptsächliche Gesichtspunkte: der funktionale und der nicht-funktionale Teil der Applikation.

Der funktionale Teil sollte die Funktionalität unserer Applikation einhergehend mit der Technik Bluetooth und der Fehleranfälligkeit der App bewerten. Im nicht-funktionalen

| Frage | arithm. Mittel |
|---|---|
| Wie war ihr erster Eindruck? | 2,78 |
| Wie bewerten Sie den Funktionsumfang der App | 2,56 |
| Wie bewerten Sie die Nützlichkeit der App.? | 2,67 |
| Wie bewerten Sie die BluetoothTechnik als Übertragungsmöglichkeit? | 2,78 |
| Wie bewerten Sie das Aussehen der App? | 2,89 |
| Wie bewerten Sie die Übersicht der App? | 3,56 |
| Wie bewerten Sie den Umgang mit der App? | 3,11 |
| Wie bewerten Sie die Navigation in der App.? | 3,00 |
| Wie bewerten Sie die Fehleranfälligkeit der App? | 2,33 |
| Wie bewerten Sie die Umsetzung des Copy&Paste Konzeptes? | 3,22 |
| Wie bewerten Sie die Kommunikation der App mit dem Nutzer? | 2,22 |
| Wie ist Ihr Gesamteindruck? | 2,89 |

Tabelle 6: Bedienungskonzepte Senderseite

Teil wurden Design- und Konzeptfragen gestellt, die auf die Gestaltung und Bedienbarkeit der Applikation prüfen sollten.

Die Verbreitung des Releasetypen erfolgte über unser Projektmanagementtool Codeplex, welches einen Downloadbereich für fertige Releases bietet.

Dafür haben wir aus unserem vorhandenen Projekt in ein APK (Applikation Package File) exportiert. Das ist ein compiliertes Dateiformat von Android, welches auf einem Android Gerät ausführbar ist und Installationsanweisungen, besitzt. APK Dateien sind paketiert, ähnlich mit ausführbaren .jar-Dateien bei herkömmlichem Java.

Eine solche Datei konnten die Testnutzer per Downloadlink auf ihr Smartphone herunterladen und installieren. Somit konnten wir den Weg über den Google Playstore umgehen, was aber leider auf einigen Geräten zu Problemen mit ihrer Firewall führte, da unsere Applikation als Schadsoftware erkannt und gesperrt wurde. Dennoch gab es bisweilen über 70 Downloads (Stand: 30.07.2013).

Die Bewertung der Fragen liegt durchschnittlich um drei von fünf Sternen und wurde bislang von neun Teilnehmern durchgeführt. Positiv bewerten die Nutzer die Übersicht in der Applikation und das von uns umgesetzte Copy&Paste Konzept. Das deutet auf eine gute Zielerreichung hin. Auffallend negativ werden die Fehleranfälligkeit und die Kommu-

nikation der App mit dem Nutzer bewertet. Da die App allerdings fast gar nicht durch die gängigen Softwaretestmethoden getestet wurde, ist so ein Ergebnis nachvollziehbar.

Ein Befragter schrieb im Freitext, dass er sich den Versand von Dateien wünscht. Da noch keine Bedienungsanleitung existiert und keine direkten Hinweise auf die Möglichkeit, Dateien zu versenden, gegeben sind, schien dem Befragten diese Funktionalität verborgen zu bleiben. Der Dateiversand sollte in einem kommenden Release über die grafische Oberfläche besser ersichtlich werden und eine Bedienungsanleitung erstellt werden, welches auch im Ausblick einen Eintrag findet.

# 7 Fazit & Ausblick

Das Studienprojekt war erfolgreich. Die Zielvereinbarung zum Leistungsumfang des Prototypen wurde eingehalten. Vorgabe war es, einen Prototypen zu entwickeln, der zwei Geräte verbindet und den einfachen Versand von Texten zulässt. Dies wurde übertroffen mit der Funktionalität Dateien verschicken und speichern zu können.

Das Projektteam kann einen großen Wissenszuwachs verzeichnen. Vorhandene Kenntnisse in der Programmiersprache Java konnten ausgebaut werden. Ergänzend dazu kommt das Wissen über die Entwicklung von Android-Apps. Das umfasst die Architektur, die Klassenbibliothek und den Vorgang der Entwicklung selbst, zum Beispiel die Virtualisierung von Android-Geräten um Testgeräte bereitzustellen.

Außerdem konnten Kenntnisse mit dem Umgang von xml-Elementen und der Bluetooth Funktionsweise gewonnen werden. Es wurde auch eine Erweiterung des Wissens über das Android Betriebssystem verzeichnet.

Android stellt eine ausführliche Wissensdatenbank für alle Entwickler bereit. Sie enthält Tutorials zu allen wichtigen Themen wie Architektur, dem Design, der Klassenbibliothek bis hin zur Veröffentlichung im hauseigenen Google Playstore.

Es gibt noch nicht so viel Fachliteratur, da Android selbst noch verhältnismäßig jung ist. Außerdem sind die Fachbücher schnell veraltet, da Android teilweise signifikante Veränderungen durchlebt hat und wahrscheinlich auch zukünftig durchleben wird (zum Beispiel der Sprung von Android 2 nach Android 4). An dieser Stelle hilft das Internet als modernes, flexibles Informationsmedium. Selbst aktuellste Probleme und Neuerungen wurden hier schon besprochen. Vor allem spezielle Fachforen zur Programmierung, wie zum Beispiel `www.stackoverflow.com`, bieten einem Neuling eine große Hilfestellung durch vorhandene Themen und gegebenenfalls auch durch Erstellung von Anfragen und deren Beantwortung durch die gigantische Community.

Organisatorisch hat das Projekt viel Zeit in Anspruch genommen, da selbst Grundwissen über die App-Entwicklung erst durch Recherche angeeignet werden musste. Das Entwicklerteam stellt allerdings fest, dass diese Zeit eine wertvolle Investition für zukünftige Projekte ist.

Die Erfahrungssammlung beschränkt sich nicht nur auf Android-spezifische Inhalte, sondern auch auf Aspekte des Projektmanagements. So ist bei einem Projekt dieser Größe ein Projektmanagementtool (hier Codeplex) nicht wegzudenken. Es vereint Versionskontrolle, Dokumentationswerkzeug und Kommunikation. Damit einhergehend wurden auch unsere Kenntnisse mit dem Umgang von UML Werkzeugen und Inhalten gefestigt.

Die Entwicklung ist allerdings auch noch nicht abgeschlossen. Es fehlen noch einige Funktionalitäten, um den Release 1.0 zu veröffentlichen. Gegenwärtig ist Release 0.8 fertig und wurde evaluiert.

Durch die Evaluation und die Rückmeldungen der Nutzer wurden wir auf Probleme und Verbesserungsvorschläge hingewiesen, die es in der weiteren Entwicklung und dem nächsten Prototypen umzusetzen gilt.

Dabei geht es in erster Linie um ein ansprechenderes Design und auch eine kurze Anleitung zum Dateiversand. Die Funktionalität war für einen Nutzer wahrnehmbar, da sie nicht explizit aus der App heraus gestartet werden kann. Auch wurde die Kommunikation der Applikation mit dem Nutzer bemängelt, welche zweifellos für ein intuitives Konzept von großem Nutzen ist. Für die Verbesserung der genannten Punkte ist ein Refactoring des Codes und eine weitere Veränderung der Versionsplanung notwendig. Die Schritte sollte vor der weiteren Entwicklung in der Releaseplanung angepasst und ergänzt werden.

Nach Version 1.0 würden wichtige Erweiterungen folgen, die die Tauglichkeit zur Vermarktung fördern würden, wie zum Beispiel die Mehrsprachigkeit und die Implementierung weiterer Übertragungstechnologien.

Es wäre denkbar das Projekt in einem kommenden Studienprojekt weiter zu entwickeln und auch fertig stellen, dabei würde sich die Entwicklung aber eher auf eine ideenreiche grafische Oberfläche und die Erweiterung der möglichen Übertragungstechniken beschränken.

# Literatur

[1] Mike Bach. *Mobile Anwendungen mit Android.* Addison-Wesley Verlag, München, 2012.

[2] Marko Gargenta. *Einführung in die Android-Entwicklung.* O'Reilly Verlag, Köln, 2011.

# Internet

[3] `http://brunocosta.com/blog/2011/11/02/speeding-up-android-development-with-android-x86-and-virtualbox/` (18.07.2013).

[4] `http://de.statista.com/statistik/daten/studie/208599/umfrage/anzahl-der-apps-in-den-top-app-stores/` (03.07.2013).

[5] `http://de.statista.com/themen/581/smartphones` (03.07.2013).

[6] `http://developer.android.com/guide/components/services.html#ExtendingService` (23.07.2013).

[7] `http://developer.android.com/guide/topics/data/data-storage.html#pref` (23.07.2013).

[8] `http://developer.android.com/guide/topics/data/data-storage.html#pref` (q3.07.2013).

[9] `http://developer.android.com/guide/topics/ui/notifiers/notifications.html` (23.07.2013).

[10] `http://developer.android.com/guide/topics/ui/notifiers/notifications.html#CreateNotification` (23.07.2013).

[11] `http://developer.android.com/reference/android/nfc/NfcAdapter.html#setBeamPushUris(android.net.Uri[],android.app.Activity)` (24.07.2013).

[12] `http://developer.android.com/reference/android/os/Environment.html` (23.07.2013).

[13] `http://developer.android.com/training/basics/supporting-devices/languages.html(24.07.2013)`,.

[14] `http://mobile-studien.de/marktanteile-betriebssysteme/marktanteile-mobiler-betriebssysteme-q1-2013/` (05.07.2013).

[15] http://umlet.com/ (15.07.2013).

[16] http://wifi-direct.net/ (26.07.2013).

[17] http://www-ivs.cs.uni-magdeburg.de/ (15.07.2013).

[18] http://www.android-x86.org/ (29.07.2013).

[19] http://www.clippick.com/Landing/ (04.07.2013).

[20] http://www.gartner.com/newsroom/id/1543014 (05.07.2013).

[21] http://www.golem.de/1102/81342.html (05.07.2013).

[22] http://www.gs1-germany.de/gs1-standards/barcodesrfid/ean-barcode/ (26.07.2013).

[23] http://www.pst.ifi.lmu.de/lehre/WS0607/mse/material/22-MSEE1-Vorgehen.pdf (16.07.2013).

[24] http://www.qrcode.com/en/ (26.07.2013).

# Anhang

## Technologiestudie

| Kriterium | Wertung | Maßstab | | | | | Bluetooth | |
|---|---|---|---|---|---|---|---|---|
| Verbreitung (in Geräten) | 30 | etablierte Technik | wenige neue Geräte | | | Entwicklungsphase | | 30 |
| Verfügbarkeit (im Betrieb) | 40 | unabhängig von Umwelt verfügbar | | benötigt globale Infrastruktur | | benötigt lokale Infrastruktur | | 40 |
| Kosten (für Anwender) | 5 | kostenfrei | | geringe Kosten | | hohe Kosten | Anschaffungskosten | 5 |
| Übertragungsbandbreite | 20 | hohe Geschwindigkeit | | geringe bzw. variirende Geschwindigkeit | | nur geringe Datenübertragung möglich | 2,1 Mbit/s | 0 |
| Übertragungsdistanz | 10 | hohe Distanz > 10m | | mittlere Distanz 1-10 m | | geringe Distanz < 1m | max. 20 m | 10 |
| Datensicherheit | 20 | grundsätzliche Verschlüsselung | | optionale Verschlüsselung | | keine Verschlüsselung | | 0 |
| Implementierungsaufwand (subjektiver Eindruck) | 20 | machbar | | höherer Aufwand | | schwierig | Funktionen im SDK vorhanden | 20 |
| Benutzerfreundlichkeit | 30 | | | | | | | 30 |
| Störanfälligkeit der Übertragung | 25 | relativ stabil | | Störung möglich | | Störungen durchaus häufig | | 25 |
| Stromverbrauch (Gerät) | 10 | | | | | | | 0 |
| | | | | | | | | **160** |

Tabelle 7: Technologiestudie I (mit Maßstab)

| Kriterium | Wertung | Bluetooth low energy | | WLAN Ad-Hoc (WiFi Direct) | | WLAN Infrastruktur | | NFC | |
|---|---|---|---|---|---|---|---|---|---|
| Verbreitung (in Geräten) | 30 | | -30 | je nach Gerät | 0 | | 30 | | 0 |
| Verfügbarkeit (im Betrieb) | 40 | | 40 | oft nur zwischen gleichen Geräten | 0 | Hotspot selten zur Verfügung Problem Hotspot-Modus | -40 | | 40 |
| Kosten (für Anwender) | 5 | Anschaffungskosten | 5 | Anschaffungskosten | 5 | Infrastruktur Kosten | 0 | Anschaffungskosten | 5 |
| Übertragungsbandbreite | 20 | 1 Mbit/s | 0 | 54 Mbit/s evtl. mehr | 20 | 54 Mbit/s evtl. mehr | 20 | 424 kbit/s | 0 |
| Übertragungsdistanz | 10 | max. 10 m | 0 | ca. 100 m | 10 | ca. 100 m | 10 | ca. 5 cm | -10 |
| Datensicherheit | 20 | | 0 | | 0 | | 20 | | 0 |
| Implementierungsaufwand (subjektiver Eindruck) | 20 | wahrscheinlich analog Bluetooth | 20 | Funktionen vorhanden Paarung evtl aufwändig | 0 | | -20 | Funktionen vorhanden | 0 |
| Benutzerfreundlichkeit | 30 | | 30 | | 0 | | 0 | | 30 |
| Störanfälligkeit der Übertragung | 25 | | 25 | | 25 | | 25 | | 25 |
| Stromverbrauch (Gerät) | 10 | | 10 | | -10 | | 0 | kleine Sendeleistung | 10 |
| | | | **100** | | **50** | | **45** | | **100** |

Tabelle 8: Technologiestudie II

| Kriterium | Wertung | 3G Internet | | QR-Code | | Infrarot | | USB-Host | |
|---|---|---|---|---|---|---|---|---|---|
| Verbreitung (in Geräten) | 30 | | 30 | | 30 | | -30 | PCs & wenige Tablets | 0 |
| Verfügbarkeit (im Betrieb) | 40 | | 0 | evtl. nur mit Internetverbindung realisierbar | 40 | | 40 | | 0 |
| Kosten (für Anwender) | 5 | Infrastruktur Kosten | 0 | Anschaffungskosten | 5 | Anschaffungskosten | 5 | | 5 |
| Übertragungsbandbreite | 20 | 53,6 Kbit/s bis 7,2 Mbit/s und mehr | 0 | bis zu 23.648 bit/Code | -20 | | 0 | | 20 |
| Übertragungsdistanz | 10 | weltweit | 10 | ca. 10 cm Sichtkontakt | -10 | Sichtkontakt | -10 | abhängig von Kabellänge | |
| Datensicherheit | 20 | abhängig von Internetverbindung | 20 | ähnlich NFC durch Anwender | 20 | | 0 | | 20 |
| Implementierungsaufwand (subjektiver Eindruck) | 20 | API von Dropbox verfügbar Evtl. zweite Technik notwendig für Verbindung | 0 | Funktionen in Libraries enthalten, z.B. Zxing | 20 | | 0 | | 20 |
| Benutzerfreundlichkeit | 30 | | 0 | | 0 | | 0 | | 0 |
| Störanfälligkeit der Übertragung | 25 | | 0 | abhängig von Kamera + Display (evtl. Problem bei Kameras ohne Autofokus) | 0 | | -25 | | 25 |
| Stromverbrauch (Gerät) | 10 | 3G sowieso in Nutzung | 10 | ein Foto | 10 | | 10 | | 10 |
| | | | **70** | | **95** | | **-10** | | **100** |

Tabelle 9: Technologiestudie III

# Ehrenwörtliche Erklärung

Wir erklären ehrenwörtlich:

1. dass wir unsere Studienarbeit selbstständig verfasst haben,
2. dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet haben,
3. dass wir unsere Studienarbeit bei keiner anderen Prüfung vorgelegt haben.

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

| | |
|---|---|
| Ort, Datum | Boris Goldshteyn |
| Ort, Datum | Clemens Wagner |
| Ort, Datum | Christian Wolter |