UNIVERSITÄT OSNABRÜCK

Institut für Informatik
AG Medieninformatik

Masterarbeit

Entwicklung einer Android Applikation zur Fernsteuerung und Nutzung eines PVR Systems

Sebastian Bruns

März 2014

Erstgutachter: Prof. Dr. Oliver Vornberger
Zweitgutachter: Reinhard Lüling

Abstract

Im Rahmen dieser Arbeit soll eine Anwendung für Android Geräte entwickelt werden, welche die Fernsteuerung und Nutzung eines PVR Systems ermöglichen soll. Die Anwendung soll in Kooperation mit der auf PVR Systeme spezialisierten Firma DiscVision GmbH entwickelt werden.

Ein PVR System bietet, angeschlossen an einen Fernseher, dem Nutzer nicht nur die Möglichkeit das aktuelle TV Programm wiederzugeben, sondern auch Aufzeichnungen von diesem zu erstellen. Dabei kann eine Aufzeichnung direkt gestartet, aber auch für einen gewissen Zeitraum eingeplant werden. Die Planung einer Aufnahme wird über eine elektronische Programmzeitschrift geregelt und verwaltet.

Mit Hilfe der Applikation soll dem Nutzer die Möglichkeit geboten werden diese Funktionen auch auf seinem Android Gerät überall abrufbereit zu haben. Somit soll die Applikation aktuelle Programm-Informationen abrufen, Einstellungen vornehmen, das PVR System von überall aus steuern können und sowohl abgelegte als auch Live-Videoinhalte wiedergeben können. Abgelegte Inhalte sollen dabei im lokalem Netzwerk über das UPnP Protokoll abrufbar sein und im entfernten Netzwerk über HTTP Live Streaming verteilt werden. Außerdem soll mittels des Sat>IP Protokolls die Übertragung des aktuell laufenden Programms auf das Android Gerät gestreamt werden können.

Zudem soll im Zuge dieser Arbeit versucht werden die Bildschirmansichten für die Nutzung von verschiedenen Android Geräten mit unterschiedlichen Bildschirmgrößen und -eigenschaften zu optimieren. Es soll ebenfalls versucht werden bei der Gestaltung und Implementation der Anwendung die Android Design Richtlinien einzuhalten.

Ziel dieser Arbeit ist neben der Implementation einer Android Applikation auch eine Evaluation über die Bedienung und Nutzerfreundlichkeit dieser. Dabei soll versucht werden auch die Design Richtlinien und verfügbare Werkzeuge genauer zu betrachten und zu bewerten. Zudem soll die Android API bezüglich der Optimierung der Bildschirmansichten in Bezug auf die verschiedenen Bildschirmgrößen untersucht werden und die verwendeten Protokolle auf ihre Funktionen und Verwendbarkeit evaluiert werden.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit beschäftigt sich mit der Implementation und Evaluation einer Android Applikation, welche die entfernte und lokale Nutzung eines PVR Systems ermöglichen soll. Das verwendete Personal-Video-Recorder (PVR) System ist eine Set-Top-Box der Firma SetOne und die sich darauf befindliche Software wird von der DiscVision GmbH entwickelt. Die Entwicklung der Android Applikation wird in enger Zusammenarbeit mit DiscVision geplant und erstellt.

Mit der Set-Top-Box ist es möglich das Fernsehprogramm aufzuzeichnen und zu einem späterem Zeitpunkt wiederzugeben. Die Aufzeichnung kann dabei sowohl vorher eingeplant werden aber auch direkt gestartet werden. Zur Planung wird ein Scheduling-System verwendet welches sich auf Daten eines elektronischen Programmführers stützt. Diese Möglichkeiten der Set-Top-Box sollen dem Nutzer durch diese Arbeit ebenfalls auf seinem Android Gerät bereitgestellt werden und sowohl eine vor Ort als auch eine entfernte Nutzung des Systems ermöglichen.

Der Nutzer soll mit seinem Smartphone oder Tablet nicht nur seine Aufzeichnungen organisieren können, sondern diese ebenfalls an seinem mobilem Endgerät abspielen können. Lokal soll dabei eine Verbindung via Universal Plug and Play (UPnP) zum Einsatz kommen, für die entfernte Nutzung soll ein HTTP Live Stream (HLS) zur Verfügung gestellt werden. Die Set-Top-Box stellt für diese Anforderungen sowohl einen UPnP Mediaserver als auch einen Webserver und Encoder für HLS bereit.
Außerdem soll der Nutzer die Möglichkeit haben lokal das aktuelle Live Bild eines gewünschten Fernsehprogramms auf seinem Android Gerät anschauen zu können. Für diese Anforderung soll das Sat-over-IP Protokoll (Sat>IP) verwendet werden, für welches die Set-Top-Box einen integrierten Sat>IP-Server bereitstellt.
Des Weiteren soll die Fernbedienung der Set-Top-Box durch eine in der Applikation integrierte virtuelle Version ersetzt werden können und mit dieser das Gerät sowohl lokal als auch entfernt bedient werden können.
Die Anbindung der Applikation wird dabei über ein von DiscVision entwickeltes Serversystem verwaltet. Durch dieses wird ebenfalls eine Nutzerverwaltung und die entsprechende Sicherheit gewährleistet.

Zu Beginn der Arbeit sollen zunächst grundlegende Informationen zum eingesetzten PVR System angeführt werden. Dabei wird die DiscVision GmbH vorgestellt, die Set-Top-Box und dessen Funktionen genauer betrachtet, sowie das für die Kommunikation verwendete Serversystem ConnectTV dargestellt. Direkt im Anschluss werden die Protokolle, von welchen Gebrauch gemacht wird, vorgestellt und genauer durchleuchtet. Dabei wird mit UPnP als Grundlage begonnen, welches dann in Sat>IP ebenfalls von Bedeutung ist und daraufhin wird das HLS betrachtet.

Im darauf folgenden Kapitel wird das Betriebssystem Android vorgestellt. Es wird auf Allgemeines zum hier verwendeten System, sowie auf die Geschichte und Verbreitung eingegangen. Im Anschluss wird die zugrundeliegende Architektur des Betriebssystems in Augenschein genommen. Zum Schluss und mit dem längstem Teil in den Android Grundlagen werden die in dieser Arbeit zum Einsatz kommenden Android-Komponenten vorgestellt.

Im vierten Kapitel werden dem Leser die Android Design Richtlinien vor Augen geführt. Dabei wird darauf eingegangen was diese besagen, welche Ideen dahinter stehen und wie diese bestmöglich innerhalb der Applikation umgesetzt werden können.

Anschließend werden kurz die verwendeten Werkzeuge und Hilfsmittel vorgestellt. Zu diesen gehören die browserbasierte Software FluidUI, welche für die Erstellung von Designentwürfen und Prototypen verwendet wurde. Außerdem wird die zum Einsatz gekommene Entwicklungsumgebung Eclipse und das für die Android Entwicklung entworfene Android Development Kit Plugin in einem kurzem Überblick dargestellt.

Im Hauptteil der Arbeit wird die eigentliche Implementation der Anwendung stehen. Dabei werden zu Beginn die umgesetzten Funktionen noch einmal genauer beschrieben und erklärt. Darauf folgt die Vorstellung des mit FluidUI erstellten Prototyps, sowie einer hinzugezogenen externen Bibliothek und einer als Media-Player verwendeten Applikation. In Folge dessen werden anhand der entwickelten Bildschirmansichten der Aufbau der Applikation vorgestellt und die Hintergründe und Anwendungen genauer erklärt.
Im Anschluss daran wird die allgemeine Datenhaltung, sowie die Verwendung des eingesetzten Models deutlich gemacht. Zudem wird die Manifest-Datei, welche die Grundlage einer jeden Android Applikation bildet, betrachtet und Schritt für Schritt erklärt.
Abschließend wird die Umsetzung der angewendeten Protokolle UPnP, HLS sowie Sat>IP beschrieben.

Im siebten Kapitel wird die Applikation mit Hilfe einer Evaluation hinterfragt. Hier wird zunächst der Ablauf erklärt, darauf folgt die Beschreibung der Anwendungsszenarien des Usability Tests sowie der Bewertungsmethodik. Zum Ende hin werden die Ergebnisse des Tests beschrieben und ausgewertet.

Zum Schluss der Arbeit folgt ein Fazit, in welchem die Durchführung der Arbeit bewertet wird. Dabei wird zum einen Bezug auf die Erreichung der gesteckten Ziele genommen. Zum anderen soll versucht werden die einzelnen Komponenten zu bewerten. Außerdem werden sowohl die Android Tools, API und Design Richtlinien kritisch betrachtet, als auch die verwendeten Protokolle des ConnectTV-Servers, UPnP, HLS sowie Sat>IP bewertet.

Kapitel 2

Grundlagen: PVR System

2.1 DiscVision

Die DiscVision GmbH ist eine auf die Forschung und Entwicklung von modernen und funktionellen PVR Systemen spezialisierte Firma. Sie wird geleitet von Herrn Reinhard Lüling, welcher auch einer der Ansprechpartner und Betreuer dieser Arbeit ist. Der Firmensitz befindet sich derzeit in Paderborn, Deutschland und es ist ein Team zwischen 5 bis 10 Entwicklern und Designern angestellt.

Bereits seit dem Jahr 2001 arbeitet DiscVision an einer eigenen Linux basierten Hardwareplattform, aber auch Hand in Hand mit Partnerunternehmen an integrierten Systemen. Neben alleinstehenden PVR Systemen wird auch an der Entwicklung von Smart-TVs und Set-Top-Boxen gearbeitet. [Disa] Eine dieser Set-Top-Boxen wird in dieser Arbeit als Hardware dienen.

Zudem stellt DiscVision eine Reihe von externen Tools und Services bereit, welche die Interaktion mit den entwickelten Systemen über einen lokalen PC oder über das Internet ermöglichen. Einer dieser Services ist das sogenannte „ConnectTV“, welches als Schnittstelle zur Kommunikation mit dem PVR System verwendet wird.

2.2 Set-Top-Box

Das hier verwendete PVR System ist in einem bereits weiterentwickeltem Modell einer Set-Top-Box integriert. Im Laufe dieser Arbeit wird daher oft PVR System und Set-Top-Box synonym verwendet. Die durch DiscVision bereitgestellte Set-Top-Box ist im Grunde genommen ein Satelliten-Receiver mit einigen Zusatzfunktionen. Es handelt sich um das Modell „Genius HD“ der Firma „Altech SetOne“ (siehe Abbildung 2.1). Die Box zeichnet sich durch eine von der DiscVision entwickelte, auf Linux basierte Benutzeroberfläche aus.

Über eine USB 2.0 Schnittstelle kann ein handelsüblicher USB-Speicher an das Gerät angeschlossen werden und als Festplatte verwendet werden. Diese Festplatte ermöglicht es Videodateien

Abbildung 2.1: Set-Top-Box des Modells Altech SetOne Genius HD [ALT]

abzuspeichern und für die spätere Betrachtung bereitzustellen. Außerdem ermöglicht sie das sogenannte Time-Shift, welches zeitversetztes Fernsehen ermöglicht. Da letztere Funktion in dieser Arbeit allerdings keine Rolle spielt, wird sie nicht weiter betrachtet. Das Bereitstellen der aufgezeichneten Videos ist allerdings von Bedeutung, da auch das Smartphone beziehungsweise das Tablet diese Videodateien wiedergeben können sollen.

Die Set-Top-Box ist netzwerkfähig und kann mittels eines RJ-45 Netzwerkanschlusses an das lokale Netzwerk und mit Hilfe eines Routers mit dem Internet verbunden werden. Über diese Schnittstelle kann die Box als Client die Kommunikation mit dem „ConnectTV“-Server aufnehmen und Daten austauschen. Wie dies vonstatten geht, wird in Abschnitt 2.3 genauer betrachtet.

Außerdem ist ein Sat>IP Server zum verteilen des aktuellen Programms eines Senders und ein UPnP Media Server zur Wiedergabe von abgelegten Mediendateien im lokalem Netzwerk implementiert. Zudem verfügt die Set-Top-Box über einen Transcoder, welcher die Videodateien in verschiedene Formate, Auflösungen und Größen transkodieren kann. Dadurch ist die Box ebenfalls in der Lage die erstellten Videodateien mit Hilfe von HTTP Live Streaming über das Internet in angepasster Auflösung an die verbundenen Clients zu verteilen.

2.3 ConnectTV

ConnectTV ist eine von der DiscVision GmbH entwickelte Server Plattform. Sie dient als Schnittstelle zur Kommunikation mit den Set-Top-Boxen und auch zur Bereitstellung von Daten von externen Dienstleistern. Außerdem hält sie die Daten aller Boxen persistent und ermöglicht einen entfernten Zugriff durch weitere Services. In Abbildung 2.2 wird der allgemeine Aufbau des Systems verdeutlicht. Die gesamte Technologie basiert auf Standard-Protokollen wie beispielsweise HTTP oder FTP. Zur Strukturierung der Kommunikationsdaten werden die kompakten Datenformate XML oder JSON eingesetzt.

Der Kern des Systems ist das Service Center(SC). Hier werden unter anderem die Programmdaten

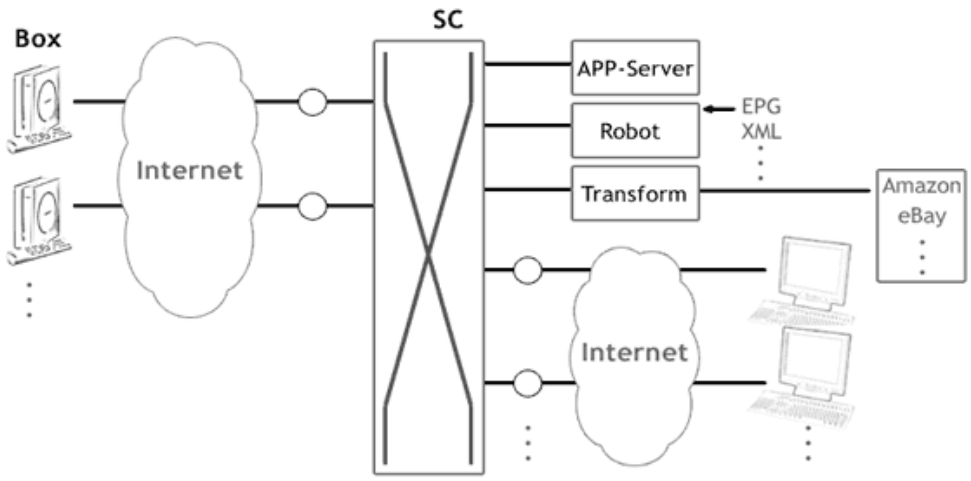


Abbildung 2.2: Schematischer Aufbau der ConnectTV Architektur [Disb]

(EPG), welche durch TV Digital, ein Service der Axel Springer AG zur Verfügung gestellt werden, verarbeitet und auf verschiedenste Weise zum Abruf bereitgestellt. Außerdem verwaltet dieses auch Applikations-Server und transformiert Inhalte von weiteren externen Diensten, wie Amazon oder eBay. Zudem kann mit jedem IP-basiertem Endgerät via HTTP über das Internet auf die Set-Top Boxen zugegriffen werden. Alle HTTP-Anfragen folgen dabei dem gleichem, im weiterem noch gezeigten, Schema. Dies ist auch das Hauptanwendungsgebiet der ConnectTV Plattform in dieser Arbeit.

Jede Set-Top-Box kann mit mehreren Benutzerkonten verknüpft werden. ConnectTV steuert über eine zentrale Datenbank die Benutzerverwaltung und somit den Zugriff auf die Set-Top-Boxen. Die Verknüpfung von Box und Nutzerkonto wird als Paarung bezeichnet. Es können über entsprechende Representational State Transfer-Services (REST) Benutzerkonten angelegt und registriert werden, Benutzer mit einer Box gepaart und wieder getrennt werden, sowie das Ein- und Ausloggen mit entsprechenden Sicherheitsabfragen geprüft werden. Jede Anfrage an den Server ist also immer mit einem Nutzerkonto verbunden und beinhaltet daher immer eine bei der Anmeldung generierte *Session-ID*.

```
1 GET /WWW/BOX_CMD/CONTENT/CHANNELLIST?protocol=1.0&sessionId=<sessionID> HTTP/1.1
2 HOST <serverAddr>:<serverPort>
3 ...
```

Listing 2.1: Beispiel einer Anfrage der Senderliste an den ConnectTV-Server

In Listing 2.1 wird der Aufbau einer Request-URL beispielhaft dargestellt. In diesem Beispiel wurde bereits über eine vorherige Anmeldung eine `sessionID` erstellt und an den Client zurückgeliefert. Diese ID ist nun innerhalb des ConnectTV Systems unmittelbar mit der entsprechenden Box verknüpft. Wird wie in diesem Beispiel nun eine Anfrage zur Senderliste der Set-Top-Box an den Server gestellt, so holt dieser sich die nötigen Informationen direkt von der Set-Top-Box. Die Daten werden dabei wie schon erwähnt im XML-Format zurückgeliefert und entspricht in

Inhalt und Reihenfolge genau der auf dem Gerät befindlichen Senderliste.

Listing 2.2 zeigt einen beispielhaften Auszug aus einer Antwort des ConnectTV-Servers. Innerhalb des Wurzelelementes `service_list` wird zunächst über das Element `numberChannels` die gesamte Anzahl der Sender bereitgestellt. Dann folgt über die einzelnen `service`-Elemente die strukturierte Auflistung der verschiedenen Sender. Die detaillierten Informationen zu den Sendern können dann über die eindeutige SID am ConnectTV-Server angefragt werden.

```
<?xml version="1.0" encoding="UTF-8" ?>
<service_list>
<numberChannels>108</numberChannels>
<service>
  <sid>286018768076800</sid>
  <servicename>ARD HD</servicename>
</service>
<service>
  <sid>286162830753792</sid>
  <servicename>RTL Television</servicename>
</service>
<service>
  <sid>286138758975373</sid>
  <servicename>Eurosport</servicename>
</service>
.
.
.
</service_list>
```

Listing 2.2: Auszug aus einer Antwort des ConnectTV-Servers auf eine Senderlisten Anfrage

2.4 Universal Plug and Play

Universal Plug and Play, kurz UPnP, ist ein Standard einer offenen Netzwerkarchitektur zur Steuerung von und zum Datentransfer zwischen IP-basierten Netzwerkgeräten. Es handelt sich um eine Peer-to-Peer Kommunikation bei der jeder Teilnehmer, anders als beim Client-Server-Modell, gleichgestellt ist und die Dienste der anderen gleichermaßen nutzen kann. UPnP wurde ursprünglich von der Firma Microsoft eingeführt, wird allerdings seit Oktober 1999 durch das UPnP Forum, welches über 1000 Mitglieder zählt, betreut. Die Version 1.1 der UPnP Architektur wird in dieser Arbeit verwendet und liegt seit Oktober 2008 vor. [UPn08]

Die wichtigsten Nutzen und Vorteile des weitverbreiteten Standards sollen kurz vorgestellt werden: [UPnb]

- **Medien- und Geräteunabhängigkeit:** Die UPnP-Technologie ist mit jeder Netzwerk-Technologie wie WLAN, Koax, Telefonleitung, Stromleitung, Ethernet oder FireWire lauffähig.
- **Plattformunabhängigkeit:** Jedes Betriebssystem und jede Programmiersprache kann zur Entwicklung von UPnP Produkten verwendet werden.

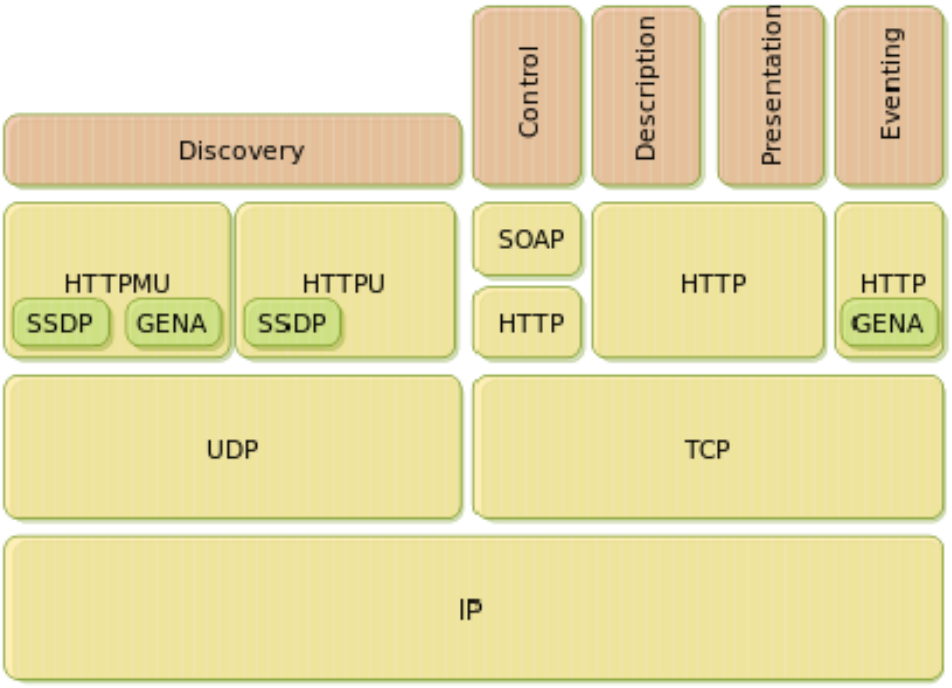


Abbildung 2.3: UPnP Architektur [UPn08]

- **Standardisierte Technologien:** Die UPnP-Technologie basiert auf IP, TCP, UDP, HTTP, XML und anderen Standards.
- **Steuerung durch Benutzerschnittstelle:** UPnP ermöglicht eine Steuerung direkt über die Gerätebenutzerschnittstelle oder auch über die Interaktion mit einem Browser.
- **Programmatische Steuerung:** Die UPnP-Architektur ermöglicht außerdem herkömmlichen Anwendungen programmatische Steuerung.
- **Basisprotokolle:** Nur Basisprotokolle vorgegeben.
- **Erweiterbarkeit:** Die Basisarchitektur kann durch individuelle Dienste überlagert werden

Im Folgenden soll genauer auf die Architektur und im Zuge dessen auch auf den allgemeinen Ablauf eines UPnP Medienzugriffs eingegangen werden. Wie Abbildung 2.3 gut verdeutlicht, besteht UPnP aus den Bereichen Discovery, Description, Control, Event Notfication und Presentation. Außerdem wird kurz auf das Addressing, die Grundlage für das UPnP Netzwerk, eingegangen. [UPn08]

Addressing: Damit eine Netzwerk Kommunikation via UPnP überhaupt möglich ist, muss das Gerät die Grundvoraussetzung erfüllen und eine gültige IP-Adresse besitzen. Ob diese Adresse mittels DHCP oder auf anderem Weg erlangt wird, ist dabei ohne belangen.

Discovery: Sobald eine erfolgreiche Adressierung vorgenommen wurde, wird die Lokalisierung

durchgeführt. Hier werden die wichtigsten Angaben über das Gerät, wie Name, Typ oder URL, mit den anderen Geräten des Netzwerks ausgetauscht. Dabei kann sich das Gerät als „einfaches“ Gerät oder aber als Controlpoint anmelden. Ein „einfaches“ Gerät versendet auf Basis des User Datagram Protocol (UDP) mittels Simple Service Discovery Protocol (SSDP) einen Multicast mit Informationen zu sich selbst, welchen alle Controlpoints registrieren. Ein Controlpoint versendet ebenfalls einen Multicast. Dieser besteht jedoch aus einer Anfrage an alle „normalen“ Geräte sich bei diesem zu registrieren. Listing 2.3 zeigt beispielhaft den Aufbau eines Discovery Multicasts zur Auffindung eines Sat>IP-Servers.

```
1 M-SEARCH * HTTP/1.1
2 HOST: 239.255.255.250:1900
3 ST: urn:ses-com:device:SatIPServer:1
4 MAN: "ssdp:discover"
5 MX: 2
```

Listing 2.3: Discovery Beipiel für einen Sat>IP Server

Description: Die Beschreibung ist via HTTP über die URL des Gerätes aus dem vorherigem Discovery in Form eines XML-Dokumentes abrufbar. In dieser Beschreibung sind nun genauere Informationen zum Hersteller, dem Gerät sowie dessen angebotene Dienste enthalten. Die Dienste werden ebenfalls in Form von URLs für das Controlling, das Eventing und das Presenting angeboten. Für jeden Dienst wird eine Liste von Kommandos und Aktionen, sowie deren Parameter und Antworten zurückgegeben.

Control: Mit Hilfe der aus der Beschreibung gewonnen Informationen über die Dienste eines Gerätes können diese nun über die Steuerung angesprochen werden. Steuerungsanfragen werden ebenfalls in Form von XML gestellt und werden über das Simple Object Access Protocol (SOAP) übertragen. Der Dienst des angesprochenen Geräts antwortet ähnlich eines Funktionsaufrufs auf die Anfragen und passt daraufhin wenn nötig interne Variablen des Dienstes an.

Event Notification: Ein Controlpoint kann als Beobachter auf das Eventsystem eines Dienstes horchen, indem er sich bei dessen Eventsystem anmeldet. Dadurch muss nicht immer wieder der Status eines Dienstes oder dessen Variablen abgefragt werden, sondern der Dienst informiert seine Beobachter falls Änderungen bestehen. Hierfür wird die General Event Notification Architecture (GENA), welche ebenfalls auf XML basiert, verwendet.

Presentation: Die Präsentation eines Gerätes wird über eine URL in der Description bekannt gegeben. Diese ist optional und bietet eine weitere Form der Darstellung des Geräts an. Hier wird eine URL angegeben, welche via HTTP über den Browser angezeigt eine grafische Oberfläche bietet. Mit dieser können ebenfalls die Informationen zu einem Gerät eingesehen werden und auch dessen Dienste angesprochen werden.

UPnP Geräte werden außerdem, wie bereits an den oben erwähnten Controlpoints und „normalen“ Geräten zu erkennen ist, in verschiedenen standardisierte Kategorien eingestuft. Dabei wird zwischen Audio/Video, Basic, Device Management, Home Automation, Networking, Printer, Remote Access, Remoting, Scanner, Sensor Management und Telephony unterschieden. Diese können dann weitergehend verfeinert werden. [UPna] Zudem werden noch weitere Add-on De-

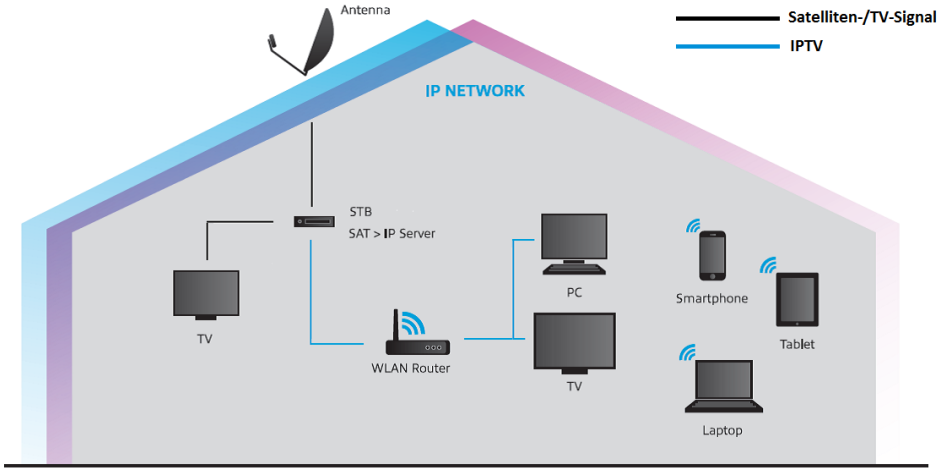


Abbildung 2.4: Übersicht zur Signalverteilung via Sat>IP (in Anlehnung an [SESa])

vice Services als Standards vorgegeben. In dieser Arbeit soll UPnP auf der Set-Top-Box als Audio/Video Mediaserver eingesetzt werden und innerhalb der entwickelte Android Applikation soll ein Controlpoint und Mediarenderer implementiert werden. Die detaillierte Verwendung wird im Laufe dieser Arbeit weiter vorgestellt.

2.5 Sat>IP

Sat>IP steht für Sat-over-IP und ist ein Protokoll, welches einen Weg bereitstellt um IP-basierten Geräten die Verteilung und den Empfang von Satellitenfernsehen zu ermöglichen. Dabei wird das vom Satelliten empfangene DVBS- beziehungsweise DVBS2-Signal in ein IPTV Datenformat umgewandelt und über das IP-Netzwerk verschickt. Das Protokoll wurde im April 2012 von der Europäische Satellitengesellschaft (SES) vorgestellt. [SES12] Es handelt sich bei diesem Verfahren um ein Client-Server-System. In Bezug auf diese Arbeit ist der Sat>IP Server in der Set-Top-Box integriert und verteilt das Signal an die verschiedenen Endgeräte. Die Applikation auf dem Android Gerät greift das bereitgestellte Signal dann als Client ab. [SESb] Abbildung 2.4 stellt eine Übersicht zur Verdeutlichung der Verteilung des Signals innerhalb eines Haushalts dar.

Da es sich um ein Client-Server-System handelt, werden diese nun zunächst separat betrachtet, bevor im darauf folgenden Abschnitt genauer auf die Architektur und den exakten Ablauf einer Kommunikation zwischen diesen eingegangen wird. Die Funktionen eines handelsüblichen Satelliten-Receivers werden bei der Verwendung von Sat>IP auf Server und Client verteilt. [SES13]

Das DVBS(2)-Signal wird in den **Server** eingespeist und von diesem in ein IP-basiertes Paketformat umgewandelt. Das Signal wird dabei nicht transkodiert, sondern lediglich über einen anderen Transportweg weitergegeben. Der Server ersetzt dadurch den Hochfrequenz-Tuner und

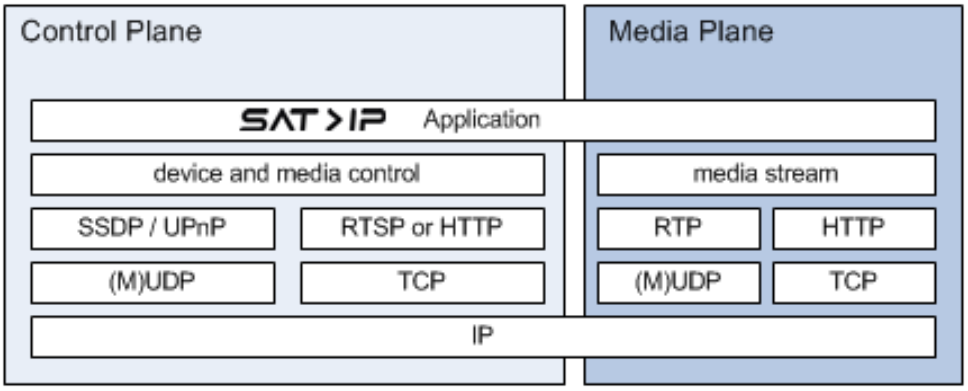


Abbildung 2.5: Sat>IP Architektur [SESb]

-Demodulator eines Receivers, welcher somit nicht mehr im Client integriert sein muss. Um mehrere Kanäle gleichzeitig abspielen zu können, können mehrere Tuner und Demodulatoren Parallel eingesetzt werden.

Der **Client** stellt mittels des Sat>IP Protokolls eine Anfrage an den Server zur Bereitstellung eines Streams eines bestimmten Kanals. Dieser wird daraufhin über das Netzwerk direkt an den Client geliefert und von diesem selbst transkodiert und dargestellt.

Nun soll der Ablauf anhand der in Abbildung 2.5 dargestellten Architektur genauer betrachtet werden. In der Sat>IP Architektur wird zwischen der Control Plane und der Media Plane unterschieden. Über die Media Plane wird der Media Stream übertragen und über die Control Plane wird die weitere Kommunikation abgehandelt, wie beispielsweise die Beschreibungen der Geräte und dessen Funktionen oder die Auswahl eines entsprechenden Kanals.

Control Plane: Das Finden von Client und Server untereinander innerhalb eines Netzwerks, sowie ein Austausch über Informationen und Funktionen wird über das bereits in Kapitel 2.4 vorgestellte UPnP abgewickelt. Somit kommen auch hier nur Standard Technologien wie UDP oder SSDP zum Einsatz. Für die Steuerung der Geräte und der vom Server verwalteten Medien-Streams wird das Real-Time Streaming Protocol (RTSP) oder HTTP verwendet. Der Server unterstützt dabei beide Protokolle bei gleicher Funktionsanwendung.

Media Plane: Der Medien-Stream selbst, hier ein Stream eines einzelnen Kanals, wird über das Real-Time Transport Protocol/Real-Time Control Protocol (RTP/RTCP) oder ebenfalls über HTTP realisiert.

Zur Verdeutlichung der Steuerung eines Medien Streams durch einen Client wird anhand von Abbildung 2.6 ein kleines Beispiel vor Augen geführt.

Setup: Zur Erstellung einer Session wird eine RTSP-SETUP-Anfrage an den Server gesendet. Diese Anfrage beinhaltet alle Informationen darüber welcher Kanal wiedergegeben werden soll, beziehungsweise unter welcher Frequenz, Polarisation, Symbolrate und weiteres der Kanal zu finden ist. Der Server gibt in seiner Antwort eine Session-ID, eine Stream-ID und weitere Informationen zurück. Im Beispiel geschieht dies beim Setup von BBC1. Listing 2.4 zeigt ein Beispiel

einer SETUP-Anfrage.

```
SETUP rtsp://192.168.128.5/?src=1&fe=1&freq=12402&pol=v&msys=dvbs&sr=27500&fec=34
        &pids=0,16,50,104,166,1707 RTSP/1.0
CSeq: 1
Transport: RTP/AVP;unicast;client_port=1400-1401
Connection:close
```

Listing 2.4: Beispiel einer Sat>IP SETUP-Anfrage

Play: Mit den aus der Antwort der Setup Anfrage erhaltenen Daten wird vom Client eine RTSP-PLAY-Anfrage gesendet. Dieser beinhaltet die ID des gewünschten Streams. In der Antwort Nachricht des Servers befindet sich nun die URL unter welcher der Stream abgerufen werden kann. Im Beispiel passiert dies beim Start von BBC1. Da nun bereits ein Setup durchgeführt wurde und somit eine Stream-ID und Session-ID an den Client übertragen wurde, kann unter Angabe der Parameter eines anderen Kanals auch mittels PLAY-Anfrage zu diesem gewechselt werden. Im Beispiel passiert dies beim Wechsel zu Channel4, dann zu ARD und später zu RTL. Listing 2.5 zeigt ein Beispiel einer PLAY-Anfrage.

```
PLAY rtsp://192.168.178.57:554/stream=2 RTSP/1.0
CSeq:2
Session:2166e663b4be550
Connection: close
```

Listing 2.5: Beispiel einer Sat>IP PLAY-Anfrage

Teardown: Um die Session permanent zu beenden, muss der Client eine RTSP-TEARDOWN-Anfrage senden. Damit wird auch die Wiedergabe des Streams beendet. Im Beispiel passiert dies bei der Beendigung des RTL Streams. Listing 2.6 zeigt ein Beispiel einer TEARDOWN-Anfrage.

```
TEARDOWN rtsp://192.168.178.57:554/stream=2 RTSP/1.0
CSeq:5
Session:2166e663b4be550
Connection: close
```

Listing 2.6: Beispiel einer Sat>IP TEARDOWN-Anfrage

Options: Jeder Server hat eine Timeout Periode, welche normalerweise bei 60 Sekunden liegt. Wenn diese überschritten wird, ohne dass der Client eine Anfrage an den Server sendet, wird die Session und Wiedergabe des Streams ebenfalls beendet. Aus diesem Grund muss sich der Client in einem gegenüber des Timeouts geringerem Zeitabstand beim Server melden. Diese Meldung wird in regelmäßigen Abständen in Form einer RTSP-OPTIONS-Anfrage durchgeführt. Im Beispiel geschieht dies ab dem Zeitpunkt des Setups und solange ein Kanal wiedergegeben wird und die Session noch aktiv bleiben soll. Listing 2.7 zeigt ein Beispiel einer OPTIONS-Anfrage.

In der Abbildung ist außerdem zu sehen, dass beim Wechsel von ARD zu RTL zunächst die Kanalparameter falsch definiert wurden. Der Server informiert den Client auch darüber in der Antwort und die Session bleibt weiterhin erhalten, somit kann mit einer weiteren PLAY-Anfrage

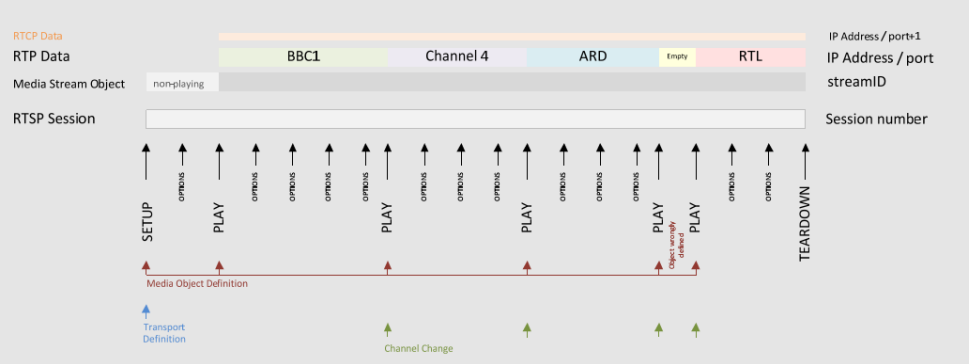


Abbildung 2.6: Sat>IP Medien-Stream Steuerung [SES13]

fortgefahren werden. Zudem wird bei jeder dieser Anfragen unter dem Parameter „CSeq“ ein fortlaufender Zähler mitgeschickt und jeweils vom Server mit der gleichen beantwortet. Dadurch wird ein kontrollierter Ablauf gewährleistet.

```
1 OPTIONS rtsp://192.168.178.57:554/stream=3 RTSP/1.0
2 CSeq:5
3 Session:2180f601c42957d
4 Connection:close
```

Listing 2.7: Beispiel einer Sat>IP OPTIONS-Anfrage

2.6 HTTP Live Streaming

HTTP Live Streaming oder auch HLS ist ein HTTP basiertes Protokoll, welches es ermöglicht über einen konventionellen Webserver Medien zu Streamen. Es wurde ursprünglich von Apple Inc. für eigene Zwecke entwickelt, kommt nun allerdings auch bereits auf anderen Systemen mit Android oder Windows zum Einsatz. Es handelt sich momentan noch um einen inoffiziellen Standard, es ist allerdings im Jahr 2013 ein erster Entwurf erstellt worden und ein Antrag an die Internet Engineering Task Force (IETF) gestellt worden. [iet13]

Da HLS auf HTTP basiert, wird die gesamte Kommunikation über Port 80 vollzogen. Somit stellen Firewalls im Normalfall keine Probleme dar, da der Datenverkehr über diesen Port immer freigegeben sein sollte. Anhand von Abbildung 2.7 soll im folgenden die Verarbeitung, Bereitstellung und Kommunikation innerhalb von HLS verdeutlicht werden.

Zunächst wird ein **Encoder** verwendet mittels dessen die Mediendatei in die Formate H.264 für Video und AAC für Audio transkodiert wird. In Zuge dessen wird die Datei in mehrere kleine Segmente geschnitten und zudem werden diese Segmente in verschiedenen Qualitäten erstellt. Außerdem wird eine m3u8-Index-Datei erstellt, welche Informationen zu den erstellten Qualitäten, META-Daten zur Mediendatei und eine Abspielliste der geschnittenen Teile bereithält. [App]

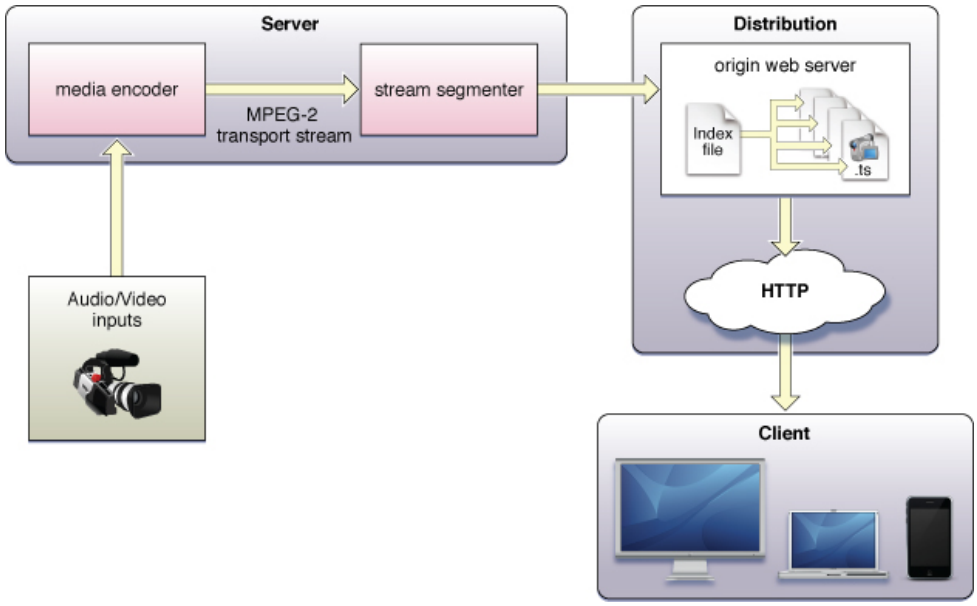


Abbildung 2.7: HTTP Live Streaming Konfiguration [App]

Damit ist das Transkodieren beendet und ein konventioneller **Webserver** wird zur Verwaltung und Bereitstellung dieser Dateien eingesetzt. Um diese Mediendatei nun über das Internet abspielen zu können, lädt der Client die Index-Datei vom Webserver. Mit Hilfe dieser Datei kann der Client nun nach und nach die benötigten Segmente aus der Indexdatei auslesen und zum Abspielen vom Webserver anfordern. Diese Segmente werden nun vom Client aneinander gereiht und abgespielt, sodass die Mediendatei ununterbrochen abgespielt werden kann. Es werden also während des Abspielens immer die nächsten benötigten Dateien vom Server angefordert und im Client an die vorherigen angebunden. Zusätzlich wird ein Puffer erstellt, um Verbindungsproblemen zu entgegnen. Da der Client über die Index-Datei ebenfalls einsehen kann in welchen Qualitäten das Video auf dem Server vorhanden ist, kann dieser anhand seiner Internetanbindung die entsprechend bestmögliche Quelle anfordern. So kann das Video auch bei schwankender Internetverbindung beziehungsweise -leistung weiterhin flüssig wiedergegeben werden. Die Auswahl der Medienqualität kann also bei jedem angefordertem Segment an die Qualität der Internetverbindung angepasst werden. [App]

Kapitel 3

Grundlagen: Android

3.1 Allgemeines

In diesem Kapitel soll das Betriebssystem Android vorgestellt und genauer durchleuchtet werden. Eine komplette Einführung mit allen Möglichkeiten von Android würde jedoch den Rahmen dieser Arbeit weit überschreiten. Aus diesem Grund sollen die Komponenten im Vordergrund stehen, welche auch in dieser Arbeit verwendet wurden. Zunächst wird auf die Entwicklung zurück geblickt bis hin zur Verbreitung. Danach wird das Betriebssystem mit seiner Architektur betrachtet und die einzelnen Komponenten kurz beschrieben um einen Einblick in die Funktionsweise zu bekommen und der späteren Implementation besser folgen zu können.

3.1.1 Geschichte

Im Jahre 2005 wurde die Firma Android Inc. von der Google Inc. akquiriert. Android ist ein quelloffenes System, welches auf Linux basiert und für den Einsatz auf mobilen Geräten konzipiert wurde. Die Entwicklung des Betriebssystems wurde zunächst von Google geleitet, bis 2007 die Open Handset Alliance [Opeb] gegründet wurde und daran beteiligt wurde. Die Open Handset Alliance (OHA) war zu Beginn ein Konsortium aus 34 Partnern aus Softwareunternehmen, Netzbetreibern, Mobiltelefonherstellern und ähnlichen Unternehmen. Heute liegt die Mitgliederanzahl bereits bei 87 Unternehmen. [Opec]
Im Jahr 2008 wurde Android in der Version 1.0 als Open Source freigegeben, das erste Software Development Kit (SDK) veröffentlicht und das erste Android Smartphone auf den Markt gebracht. [KM12] Heute ist Android bereits in der Version 4.4 erhältlich und wird noch immer unter der Leitung der OHA weiter entwickelt.

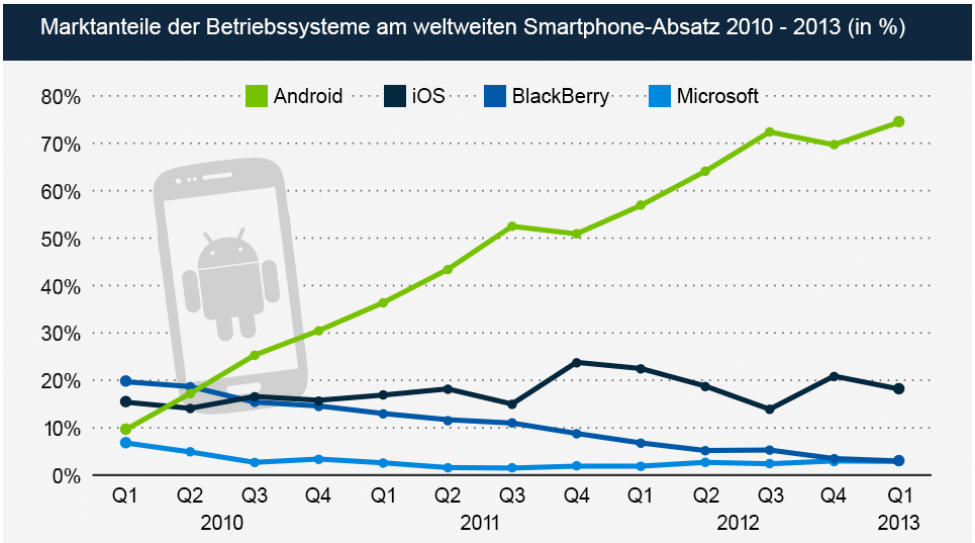


Abbildung 3.1: Marktanteile der Smartphone Betriebssysteme im August 2013 [Gar]

3.1.2 Verbreitung

Android ist ein Betriebssystem, welches hauptsächlich auf mobilen Geräten, wie Smartphones, Tablets oder Laptops zum Einsatz kommt. Es wird aber auch bereits als Grundlage für SmartTVs [Phi] und in naher Zukunft auch in Fahrzeugen [Opea] in Anspruch genommen.

Neben Android sind die größten Smartphone Betriebssystem Vertreter iOS von Apple Inc. und die von Blackberry oder Microsoft entwickelten Plattformen. Wie jedoch Abbildung 3.1 zu entnehmen ist, kann nur Android sich richtig am Markt etablieren und weiter verbreiten. Microsoft und Blackberry verlieren immer mehr Anteile, einzig iOS kann die Marktanteile von gut 20% halten. Smartphones mit Android haben jedoch den Markt regelrecht überflutet und sind in den letzten drei Jahren von 10% bis auf über 70% gestiegen. Dies unterstreicht den heutigen Stellenwert von Android und erklärt auch warum Hersteller vermehrt auf Android setzen und diese Arbeit ebenfalls für Android entwickelt werden soll.

Wie bereits erwähnt ist die Entwicklung von Android bereits bei der Version 4.4 angelangt. Da der Beginn dieser Arbeit jedoch mit Anfang September datiert ist, wird immer auf die Zahlen vom August 2013 als Grundlage von Entscheidungen verwiesen. Zu diesem Zeitpunkt war der Entwicklungsstand die Android Version 4.2 alias „Jelly-Bean“. Aus den Statistiken in Abbildung 3.2 ist zu entnehmen, dass bei weitem nicht alle Geräte mit der neusten Android Version ausgestattet sind. Dies ist auf mehrere Gründe zurückzuführen.
Die Hersteller arbeiten immer an neuen Smartphones und es ensteht Entwicklungsaufwand die alten Smartphones an die neuen Android Versionen anzupassen. Zum einem ist dies ein Hardware-Treiber Problem, welche immer weiterentwickelt werden müssten. Zum anderem haben viele Hersteller die Android Plattform noch mit eigenen Variationen von Software, Layouts und Anderem versehen, welche daraufhin ebenfalls wieder angepasst werden müssen. Außerdem erhoffen sich natürlich auch einige Hersteller auf Verkäufe ihrer neuen Modelle, welche direkt mit der neusten

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.2%
2.2	Froyo	8	2.5%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	33.0%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	22.5%
4.1.x	Jelly Bean	16	34.0%
4.2.x		17	6.5%

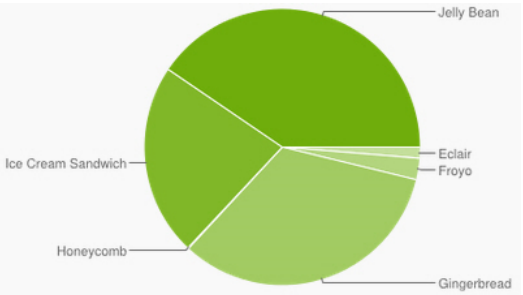


Abbildung 3.2: Verbreitung der Android Versionen im August 2013 [Wif]

Androidversion ausgestattet werden.

Das ist ein wichtiger Punkt für die Entwicklung dieser Arbeit, da natürlich möglichst allen Nutzern eine Verwendung der Applikation ermöglicht werden soll. Das Android Betriebssystem ist abwärtskompatibel, aber nicht aufwärtskompatibel. Das heißt, dass die Applikation in einer möglich frühen Version von Android entwickelt werden muss um vielen Nutzern die Verwendung zu ermöglichen. Unter Rücksprache mit der Discvision GmbH ist man zu dem Entschluss gekommen die Anwendung für die Version 2.3 zu entwickeln. So können zum einen über 95% aller Nutzer die Applikation auf ihrem Android Gerät installieren und verwenden. Zum anderen kann in der Entwicklung schon auf viele wichtige Funktionen des Betriebssystems zurückgegriffen werden. Die meisten Optimierungen für die Verwendung von Tablets und der neuen Design Richtlinien sind allerdings erst ab Version 3.0 verfügbar [Andh], dies hat zu einigen Problemen geführt, die allerdings weitestgehend in der Implementation gelöst werden konnten.

3.2 Architektur

In diesem Abschnitt soll die System Architektur von Android anhand des in Abbildung 3.3 dargestellten System-Stacks erläutert werden und kurz auf die Besonderheiten der einzelnen Ebenen eingegangen werden.

Linux Kernel: Als Grundlage des Android Systems dient ein Linux Kernel. Er liegt in der Version 2.6 vor, bis er ab Android 4.x von der Version 3.x ersetzt wird. Dieser verwaltet beispielsweise den Speicher und die Prozesse und ist für das Energie Management und die Sicherheit zuständig. Außerdem bildet er die Hardwareabstraktionsschicht und ist somit Schnittstelle zu den Gerätetreibern des Systems. [Mei12]

Libraries and **Android Runtime:** Die nächste Schicht wird von einigen Bibliotheken und der Android Laufzeitumgebung gebildet. Android beinhaltet eine Reihe von C/C++ Bibliothe-

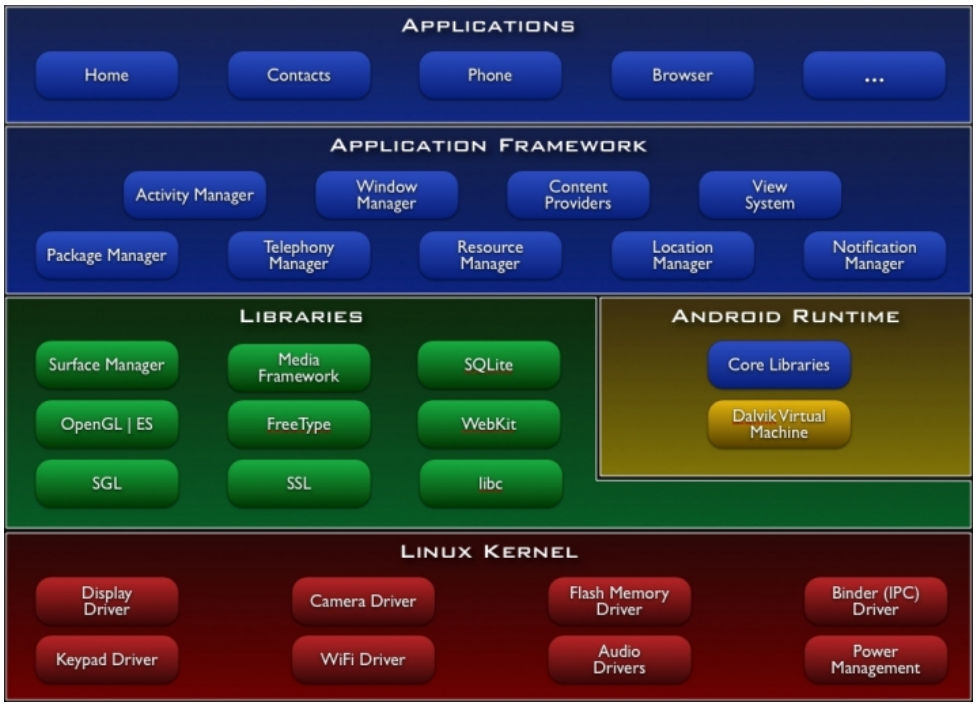


Abbildung 3.3: Android System Architektur [Ande]

ken, welche die Vorraussetzung schaffen Medien abzuspielen, SQLite Datenbanken zu verwalten, OpenGL Grafik Elemente darzustellen oder auch die Bildschirmsteuerung zu ermöglichen. Die Android Runtime ist der Java Runtime ähnlich, unterscheidet sich allerdings in wichtigen Details. Android Anwendungen werden ebenfalls nativ in Java geschrieben, werden aber von der Dalvik Virtual Machine verarbeitet. Diese ist, anders als die Java VM, eine Registermaschine, wurde speziell für Android entwickelt und ist unter anderem auf minimalen Speicherverbrauch und effiziente Ausführbarkeit von mehrere virtuellen Maschinen optimiert. [Kün12] Die Java-Dateien werden zunächst wie gewöhnlich in Bytecode, jedoch anschließend in ein eigens entwickeltes dex-Format umgewandelt. Außerdem bieten die Core Libraries viele Funktionen und Klasssen aus Java, aber auch einige explizit für Android optimierte Methoden.

Application Framework: Das Application Framework ist der Teil, den der Entwickler wirklich zu sehen bekommt und benutzt. Es stellt über eine gute API Funktionen und Klassen zur Verfügung um einfachen Zugriff auf die Hardware des Geräts zu haben. Außerdem können über das Framework die Komponenten und deren Funktionen erstellt und bearbeitet werden, welche im Abschnitt 3.3 genauer beschrieben werden. Ein wichtiges Konzept dabei ist die Bereitstellung von Funktionen einer Applikation für andere Applikationen. So kann zum Beispiel eine eigene Applikation ohne großen Aufwand den Browser, E-Mail-Client oder Media-Player einer anderen Applikation verwenden. [Kün12]

Applications: Die nächste und letzte Schicht sind die gerade schon erwähnten Applikationen selbst. Diese sind die Anwendungen, welche der Benutzer auf seinem mobilem Endgerät sieht

und mit denen er interagieren kann. Als Beispiele können hier der Browser, ein Musik-Player oder auch die Telefonfunktion angeführt werden. In dieser Arbeit wird eine solche Applikation für den Endnutzer entwickelt.

3.3 Komponenten

In dieser Sektion werden einige Komponenten des Application Frameworks vorgestellt. Es handelt sich dabei nicht um alle Komponenten, sondern nur um einige der Bereiche, welche auch in dieser Arbeit verwendet wurden. Wichtig für das Verständnis des Android Frameworks ist, dass es sich um eine Art Model-View-Controller Entwurfsmuster handelt. Das Muster strukturiert die Anwendung in Datenmodell, Präsentation und Programmsteuerung. Dies ermöglicht eine vereinfachte Austauschbarkeit und Wiederverwendbarkeit. Wie die Komponenten den einzelnen Bereichen zugeordnet werden, wird in den folgenden Untersektionen beschrieben.

3.3.1 Activity

Eine Android Activity ist die Anwendungskomponente, welche dem Nutzer ein Interface bereitstellt mit dem er interagieren kann. Eine Anwendung besteht im Normalfall aus mehreren leicht zusammenhängenden Activities. Es wird eine Haupt-Activity festgelegt, welche als Einstiegspunkt einer Applikation dient. Aus dieser Activity können durch Interaktion weitere Activities aufgerufen werden und zu einer Gesamtapplikation zusammengesetzt werden. [Andb]

Eine Komponente vom Typ Activity kann dabei als der Controller angesehen werden, welcher die Eingaben der View verarbeitet und mit dem Model kommuniziert. Das Aussehen einer Activity wird in den meisten Fällen durch eine XML Datei beschrieben und mittels der Methode `setContentView()` geladen. Diese XML Datei kann also als View angesehen werden und wird in Abschnitt 3.3.5 noch genauer betrachtet. Es ist allerdings auch möglich spezielle Activities zu verwenden, welche beispielsweise direkt die Form einer Liste besitzen und dementsprechende Interaktionselemente mit sich bringen. Diese müssen dann nur noch mit entsprechenden Datensätzen versorgt werden.

Die Activities werden in einem „Back Stack“ organisiert. Jede neu gestartete Activity bekommt den Fokus und wird damit oben auf diesen Stack gelegt, dadurch wird die vorherige Activity mit allen aktuellen Eingaben pausiert. Der Stack basiert dabei auf dem bekanntem „last in, first out“ Prinzip. Wenn der Nutzer die Interaktion innerhalb einer Activity abgeschlossen hat und diese über die Zurück-Taste verlässt, wird die Activity beendet und vom Stack gelöscht. Die zuvor aktive Activity wird fortgesetzt und in den Vordergrund geschoben. Ist der Stack leer, wird die Anwendung beendet und zum Betriebssystem zurückgekehrt. [Kün12]

Die Persistenz der einzelnen Activities kann hierbei durch ihren speziellen Lebenszyklus (siehe Abbildung 3.4) kontrolliert werden. Um eine gut organisierte und flexible Applikation entwickeln zu können, ist es essentiell die Callback-Methoden einer Activity zu verwenden. Eine Activity kann sich in drei verschiedenen Zuständen befinden: [Andb]

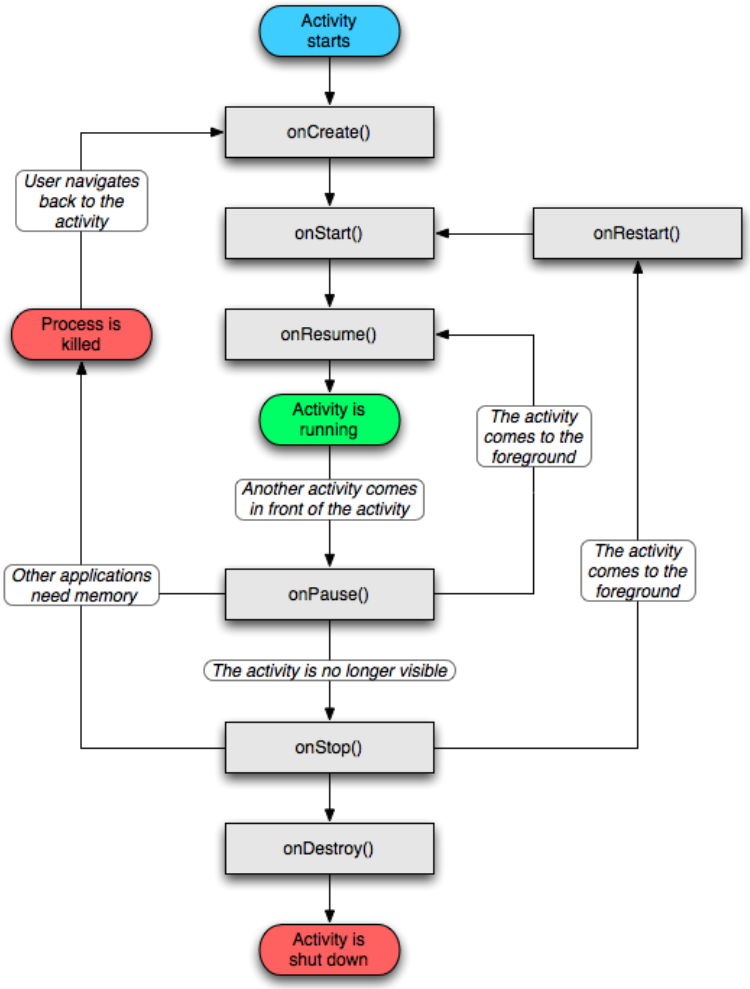


Abbildung 3.4: Activity Lebenszyklus [Andc]

- **Resumed/Running:** Die Activity befindet sich im Vordergrund und hat den Fokus des Nutzers.
- **Paused:** Eine andere Activity liegt im Vordergrund, aber verdeckt diese Activity nur bedingt, entweder teilweise oder sie ist transparent. Die Activity ist in diesem Status noch aktiv, dabei wird das Objekt im Speicher gehalten und wird nur bei sehr starker Speicherknappheit freigegeben.
- **Stopped:** Die Activity liegt vollkommen im Hintergrund. Auch diese Activity ist noch im Speicher vorhanden, wird aber bei Speicherknappheit ohne weiteres freigegeben.

Wie gerade erwähnt besitzen Activities sogenannte Callback-Methoden. Diese werden bei einem Wechsel der Status, wie in Abbildung 3.4 verdeutlicht, aufgerufen. Diese Methoden können innerhalb einer jeden Activity überschrieben werden, um geeignet auf die einzelnen Zustände der

Activities reagieren zu können. So kann auf jeden Zustand oder jede unerwartete Unterbrechung reagiert werden.

3.3.2 Intents

Intents werden in Android als internes Nachrichtensystem verwendet. Mit einem Intent können andere Activities gestartet und gleichzeitig Informationen übertragen werden. Dies wird mit der Methode `startActivity()`, welche den Intent übergeben bekommt, durchgeführt. Außerdem können mit Hilfe von Intents andere Applikationen gestartet werden, welche entsprechende Module freigegeben haben. Es können auch Broadcasts mit Deklarationen von generellen Anforderungen an das Androidsystem verschickt werden, wie beispielsweise das Abspielen einer Media-Datei. Alle auf dem System installierten Applikationen erhalten diesen Broadcast und können ihren Dienst, um diese Aufgabe zu bewältigen, anbieten. Der Nutzer wählt dann anhand einer Liste die gewünschte Applikation aus. [KM12]

Wenn eine Activity der eigenen Applikation oder eine spezielle Applikation gestartet werden soll, wird ein sogenannter **expliziter** Intent versendet. Das Pendant dazu ist der **implizite** Intent, welcher wie der oben beschriebene Broadcast keine spezielle Anwendung, sondern irgendeine Anwendung anfordert, welche die geforderte Aktion durchführen kann. [Andb]

Außerdem ist es möglich eine Activity zu startet und ein Resultat bei der Rückkehr zur ursprünglichen Activity anzufordern. Dies wird mit Hilfe der Methode `startActivityForResult()` erreicht. Um das Resultat der Activity zu erhalten wird die Callback-Methode `onActivityResult()` überschrieben und automatisch nach Rückkehr zur ursprünglichen Activity aufgerufen.

3.3.3 Tasks

Android ist eine Multithreading Plattform, das heißt jede Anwendung läuft in einem eigenem Thread. [Andp] Somit ist jede Anwendung für sich gekapselt und kann bei einem Absturz die anderen laufenden Anwendungen oder auch das Betriebssystem nicht beeinflussen. Dies wird auch als sogenannte Sandbox bezeichnet. Eine Anwendung wird in einem Haupt-Thread gestartet, genau wie auch alle weiteren Komponenten der Applikation im Normalfall diesen Thread verwenden. Er wird auch UI-Thread genannt, da in diesem auch die Benutzeroberfläche generiert wird.
Einige Arbeitsschritte benötigen etwas mehr Zeit, wie beispielsweise das Herunterladen eines Bildes oder das Laden von anderen Daten aus dem Speicher. Damit die Darstellung der Nutzeroberfläche nicht durch solche Prozesse beeinträchtigt beziehungsweise verlangsamt wird, hält sich Android an den sogenannten StrictMode [Andm]. Dieser soll verhindern, dass der UI-Thread durch solche Tasks verlangsamt wird und wirft eine Warnung, falls der UI-Thread nicht antwortet. Es ist eine Sicherheitsmaßnahme um eine flüssige Benutzerinteraktion zu gewährleisten.
Um den UI-Thread also nicht zu verlangsamen, bietet Android die Möglichkeit einen sogenannten AsyncTask auszuführen. [MDMN12] Dieser wird durch den UI-Thread gestartet und läuft daraufhin im Hintergrund. Sobald der AsyncTask durchlaufen ist, wird innerhalb des UI-Thread

eine Callback-Methode aufgerufen, welche die UI entsprechend anpassen kann. Listing 3.1 zeigt einen AsyncTask und dessen Aufruf.

```
1  public void onClick(View v) {
2      new DownloadImageTask().execute("http://example.com/image.png");
3  }
4
5  private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
6      /** The system calls this to perform work in a worker thread and
7        * delivers it the parameters given to AsyncTask.execute() */
8      protected Bitmap doInBackground(String... urls) {
9          return loadImageFromNetwork(urls[0]);
10     }
11
12     /** The system calls this to perform work in the UI thread and delivers
13       * the result from doInBackground() */
14     protected void onPostExecute(Bitmap result) {
15         mImageView.setImageBitmap(result);
16     }
17 }
```

Listing 3.1: Beispiel eines AsyncTasks [Andk]

In Zeile 2 wird der AsyncTask über die Methode `onClick()` erstellt und ausgeführt. Innerhalb des Tasks in Zeile 8-10 wird dann in der Methode `doInBackground()` die Aufgabe im Hintergrund gestartet, in diesem Beispiel wird ein Bild von einer übergebenen URL geladen. In der Methode `onPostExecute()` in den Zeilen 14-16 wird nach Abschluss des Ladevorgangs wieder im UI-Thread gearbeitet und das Bild entsprechend gesetzt. So wird die Interaktion der UI mit dem Nutzer nicht eingeschränkt und es führt zu keinen merklichen Verzögerungen.

3.3.4 Fragments

Fragments wurden mit Android 3.0 eingeführt und gehören zu den wichtigen Neuerungen für die Anpassung der Bildschirmansicht an größere Bildschirme wie bei Tablets. Durch die Einbindung der von Google bereitgestellten Android „Support Library“ [Andn], ist es jedoch auch möglich die Fragments bereits in vorherigen Android Versionen nutzen zu können. Somit ist dies auch in dieser Arbeit, in welcher die Version 2.3 verwendet wird, möglich.
Ein Fragment könnte als „kleiner Buder“ der Activity bezeichnet werden und sie werden verwendet um die Bildschirmansicht einer Activity modular gestalten zu können. Ein Fragment verfügt wie eine Activity über eine eigene Bildschirmansicht. Es können mehrere Fragments in einer Activity eingebettet sein. Dies hat den Vorteil, dass die Fragments unter Berücksichtigung der Bildschirmgröße und -orientierung entsprechend innerhalb der Activity angeordnet werden können, um die gesamte Größe des Bildschirms optimal nutzen zu können.
In Abbildung 3.5 ist ein Beispiel zu sehen, wie die Bildschirmansicht einer Applikation mit Hilfe von Fragments an einen größeren Bildschirm angepasst werden kann. Dabei werden Fragments, welche auf einem Smartphone mit relativ kleinem Bildschirm in verschiedenen Activities untergebracht wurden, auf einem Tablet nebeneinander angeordnet. Diese sind somit immer gleichzeitig

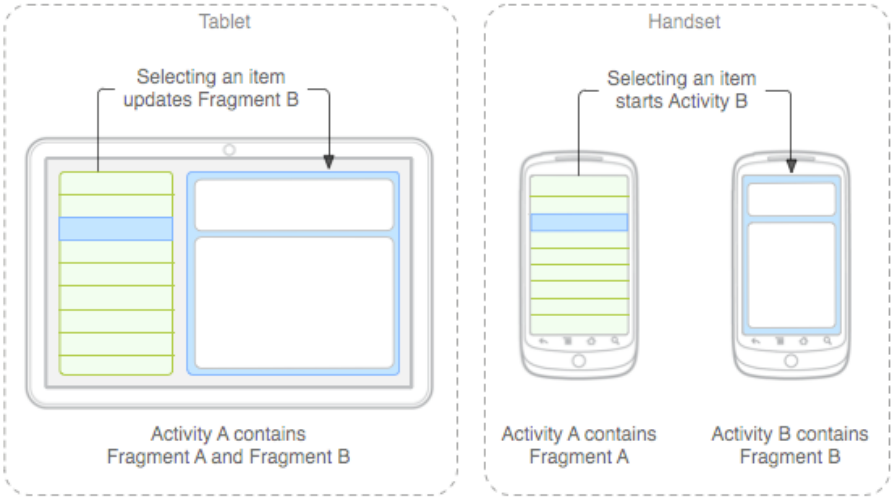


Abbildung 3.5: Design mit Fragments [Andg]

sichtbar und bedienbar. Diese dynamische Anpassung bringt den Mehrwert eines größeren Bildschirms erst wirklich zum Vorschein.

Fragments verfügen ähnlich der Activities über ihren eigenen Lebenszyklus, Status und Back Stack und werden über einen `FragmentManager` verwaltet. Außerdem können Fragments innerhalb des Programmcodes einer Activity durch Transaktionen dynamisch ausgetauscht und angepasst werden. Sie können überdies ineinander verschachtelt werden und so beliebig komplex angeordnet werden. [MDMN12]

Fragments haben zusätzlich noch einen weiteren Einsatzbereich. Sie können innerhalb eines sogenannten `ViewPagers` eingesetzt werden. [Andr] Ein Viewpager verwaltet die einzelnen Fragments nebeneinander und ermöglicht das sogenannte „sliden“ zwischen diesen. Dies wird durch horizontales Wischen bewirkt. So können innerhalb einer Activity mehrere Bildschirmansichten mit Hilfe von Fragments nebeneinander angeordnet werden und über automatisch erzeugte Animationen ausgewählt werden.

3.3.5 Ressources

Unter Ressourcen versteht man in Android austauschbare Daten wie beispielsweise Bilder, Strings, Layouts oder Farbdeklarationen. Diese Ressourcen sollten immer ausgelagert werden. Ein entscheidender Vorteil der Auslagerung liegt in der Transferierbarkeit und Wiederverwendbarkeit dieser Elemente. Auch wenn die Daten extern gelagert werden, ermöglicht die API immer einen einfachen und schnellen Zugriff innerhalb des Programmcodes. Nachfolgend sollen die in dieser Arbeit zum Einsatz kommenden, wichtigsten Ressourcentypen kurz vorgestellt werden.

Drawable: Unter Drawables versteht man alle Arten von Grafiken, welche auf dem Bildschirm dargestellt werden. In der Regel sind dies Bilder oder Icons wie auch das Starter-Icon der Ap-

plikation. Die Drawables unterliegen dabei einer weiteren Unterteilung nach ihrer Auflösung. So sollten für unterschiedliche Bildschirmgrößen und -auflösungen verschiedene Bildgrößen bereitgestellt werden. Denn während ein Smartphone mit einem vergleichsweise kleinem Bildschirm auch nur ein kleines Bild mit geringer Auflösung benötigt, sollte für ein Tablet das gleiche Bild mit höher Auflösung verwendet werden. Hierdurch wird zum einen überflüssiges Laden von zu großen Bildern verhindert und zum anderen eine immer gleichbleibend gute Bildqualität sichergestellt. Der entsprechende Unterordner wird dabei vom Betriebssystem selbst ausgewählt, sodass durch den Entwickler lediglich die richtigen Größen in entsprechend standardisierten Ordnern bereitgestellt werden müssen. [Andf]

Layout: Durch die Layouts werden die Bildschirmansichten der einzelnen Activities mittels der Markup Language XML beschrieben. Die Bildschirmansicht ist hierfür in einer Baumstruktur modelliert. Die einzelnen Elemente werden dabei ineinander verschachtelt und es wird durch ihre Art und entsprechende Parameter festgesetzt wie diese sich im Elternelement anordnen. Einige Beispiele hierfür werden später in Kapitel 6 genauer betrachtet. Es ist ebenfalls möglich im Programmcode auf die verschiedenen Elemente der Bildschirmansicht zuzugreifen und diese zur Laufzeit anzupassen.
Ein weiterer wichtiger Punkt ist, dass unterschiedliche Bildschirmansichten für verschiedene Bildschirmausrichtungen und -größen verwendet werden können. Wenn ein Smartphone im Hochformat gehalten wird, spricht man vom „Portraitmode“. Wird es allerdings um 90° ins Querformat gedreht, wird vom „Landscapemode“ gesprochen. Die Orientierung wird vom Betriebssystem automatisch erkannt und der Entwickler kann den Bildschirminhalt beziehungsweise die Anordnung der Elemente an die Orientierung anpassen. Hierfür werden wieder verschiedene Unterordner verwendet, welche auch hier vom System automatisch ausgewählt werden. Zudem können weitere Ordner für verschiedene Bildschirmgrößen angelegt werden, damit bei jeder Bildschirmgröße der Platz möglichst optimal genutzt werden kann. [Ando]

Menu: Jede Android Activity kann über ein eigenes (Options-)Menü verfügen. Dieses Menü kann zum einem im Code der Activity zur Laufzeit erzeugt werden, zum anderen kann es auch als Ressource bereitgestellt werden. Auch hier kommt wieder die Auszeichnungssprache XML zum Einsatz.

```
1  <menu xmlns:android="http://schemas.android.com/apk/res/android" >
2
3      <item
4          android:id="@+id/action_settings_date"
5          android:orderInCategory="100"
6          android:showAsAction="always"
7          android:icon="@drawable/action_date"
8          android:title="@string/action_settings_date"/>
9
10     <item
11         android:id="@+id/action_settings"
12         android:orderInCategory="100"
13         android:showAsAction="never"
14         android:title="@string/action_settings"/>
15
16 </menu>
```

Listing 3.2: Beispiel einer Menübeschreibung in XML

In Listing 3.2 wird ein kleines Menü beschrieben, welches über zwei Menü-Items verfügt. Wenn dieses in einer Activity geladen wird, werden dort über die Menü-Taste des Handys diese beiden Menüpunkte angezeigt. Die Titel werden über das Feld `title` angegeben. In diesem Fall werden die Titel aus der Ressource `string` geladen, auf diese wird im folgenden Abschnitt noch weiter eingegangen. Außerdem bekommt jedes Item eine ID um im Quellcode eindeutig ansprechbar zu sein. Als weitere optionale Elemente kann die Darstellungsreihenfolge der Menüpunkte mittels `orderInCategory` durch eine einfache Wertung angegeben werden. Wenn ein Menüpunkt als Icon dargestellt werden soll, kann dies durch das Feld `showAsAction` eingestellt werden. Wenn dies aktiviert ist, muss dazu noch ein Icon ausgewählt werden. Das Icon wird im ersten Item über eine `drawable` Ressource geladen. Ein Icon kann erst ab der Android Version 3.0 dargestellt werden, da erst ab dieser Version die sogenannte `ActionBar` eingeführt wurde. Durch die ActionBarSherlock Bibliothek kann allerdings auch diese in dieser Arbeit verwendet werden. Näheres dazu folgt in den Abschnitten 4.2 und 6.3.

Values: Im Bereich Values werden unter anderem die Ressourcen `string`, `colors` oder `styles` in XML Form abgelegt. Die String Ressource sollte für die Verwaltung aller in der Anwendung vorhandenen Strings verwendet werden. Die Verwendung der `string.xml` ermöglicht es die Applikation sehr simpel in verschiedenen Sprachen bereitzustellen. Es wird hierfür eine Default String-Datei in der Default Sprache angelegt und für jede weitere Sprache kann eine weitere String-Datei in einem entsprechend standardisiert benanntem Values Ordner angelegt werden. Anhand der eingestellten Systemsprache des Geräts wird dann automatisch die entsprechende Sprachdatei geladen. Ist diese Sprache nicht vorhanden, wird die Default-Sprache verwendet. Die verschiedenen Sprachordner werden dabei um den offiziellen ISO Sprachcode erweitert. Der Ordner für eine deutsche Sprachdatei würde beispielsweise wie folgt aussehen: „values-de“.
Über die Datei `colors.xml` können applikationsweit verwendete Farben gespeichert werden und im Programmcode einfach aufgerufen werden. Ein Wechsel einer applikationsweit verwendeten Farbe kann so durch die Änderung von nur einer Variabel erreicht werden.
Die Datei `styles.xml` stellt das entsprechende Pendant für applikationsweit verwendete Styles grafischer Elemente. Dies gibt dem Entwickler ebenfalls die einfache Möglichkeit des Austauschs und der Wiederverwendbarkeit der entsprechenden Elemente. [KM12]

3.3.6 Manifest

Das Manifest ist die Basis der gesamten Applikation und muss in jeder Anwendung im Hauptverzeichnis liegen. Es ist ebenfalls im XML-Format und gehört nicht zum ausführbaren Teil der Applikation, sondern beinhaltet eine Beschreibung sowie die Umgebungsparameter der Anwendung. In der Manifest-Datei werden die wichtigsten Informationen zur Applikation abgelegt, welche das Android System benötigt. Die folgende Liste soll die Inhaltselemente eines Manifests kurz beschreiben: [MDMN12]

- Die Version der Applikation, sowie der Name und das verwendete Icon im Betriebssystem.
- Das Java-package, welches als einzigartige Identifizierung dient.
- Alle Komponenten der Applikation, wie die Activities und dessen Namen, Services oder

Intentfilter, welche die zu verarbeitenden Broadcasts bestimmen.

- Die minimal benötigte Version der Android API zur Verwendung der Applikation.
- Die benötigten Rechte der Applikation, wie beispielsweise den Zugriff auf das Netzwerk, die Kontaktdaten, die Kamera, etc.
- Die erlaubten Rechte einer anderen Applikation, welche die Funktionen der freigegebenen Komponenten dieser Applikation verwenden möchte.
- Eine Liste der Bibliotheken die mit der Applikation verknüpft werden müssen.

3.3.7 Preferences

Um auf einem mobilem Endgerät Daten permanent speichern zu können, bietet Android mehrere Möglichkeiten. In dieser Arbeit wurden dafür ausschließlich die `SharedPreferences` verwendet, aus diesem Grund werden auch nur diese hier weiter erläutert.
Mit Hilfe der `SharedPreferences` können primitive Datenypen in Schlüssel-Wert-Paaren abgespeichert werden. [Mei12] Listing 3.3 zeigt ein Beispiel wie das Speichern und Laden dieser Daten erfolgt.

```
1  public class Calc extends Activity {
2      public static final String PREFS_NAME = "MyPrefsFile";
3
4      @Override
5      protected void onCreate(Bundle state){
6          super.onCreate(state);
7          . . .
8
9          // Restore preferences
10         SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
11         boolean silent = settings.getBoolean("silentMode", false);
12         setSilent(silent);
13     }
14
15     @Override
16     protected void onStop(){
17         super.onStop();
18
19       // We need an Editor object to make preference changes.
20       // All objects are from android.context.Context
21       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
22       SharedPreferences.Editor editor = settings.edit();
23       editor.putBoolean("silentMode", mSilentMode);
24
25       // Commit the edits!
26       editor.commit();
27     }
28 }
```

Listing 3.3: Beispielhaftes Laden und Speichern von `SharedPreferences` [Andl]

In der Methode `onStop()` wird das Speichern der Einstellungen gezeigt. Hier wird die Einstellungsdatei zunächst in die Variable `settings` geladen. Daraufhin wird in Zeile 22 der Editor der

Einstellungsdatei geladen, welcher das Bearbeiten der Einstellungen ermöglicht. In der darauf folgenden Zeile wird der Boolean-Wert `mSilentMode` unter dem Schlüssel „silentMode“ in den Einstellungen abgespeichert.
Das Laden der Einstellungen wird in der Methode `onCreate()` vollzogen. Hier wird wie beim Speichern zunächst die Einstellungsdatei geladen. In Zeile 11 wird unter Verwendung des Schlüssels „silentMode“ der Boolean-Wert geladen und in die lokale Variable `silent` geschrieben. Ist diese noch nie zuvor in den Einstellungen gesetzt worden, wird diese mittels des dahinter angegeben Standard-Wertes auf `false` gesetzt.
Durch dieses Verfahren kann nach der Beendigung der Anwendung eine vorher gemachte Einstellung wieder geladen werden. Diese Einstellungen werden über den Lebenszyklus der Applikation hinaus persistent gehalten.

Kapitel 4

Grundlagen: Android Design Richtlinien

Da in dieser Arbeit auch die Android Design Richtlinien untersucht werden und mittels Evaluation eventuelle Probleme, Verbesserungen oder auch positives Feedback ermittelt werden sollen, werden die wichtigsten Punkte in den nächsten Abschnitten beschrieben und erläutert. Alle hier aufgeführten Elemente und Funktionen stützen sich auf die Beschreibungen aus den Android Entwickler Design Richtlinien. [Andj] [Andq]

4.1 Struktur und Navigation

Die Hierarchie der Applikationsstruktur sollte immer möglichst flach bleiben um keine Verwirrung beziehungsweise Orientierungslosigkeit beim Nutzer hervorzurufen. Zudem bietet dies die Möglichkeit schnell und mit wenigen Klicks innerhalb der Anwendung zu navigieren.

Beim Start einer Applikation sollte eine „Top level view“ angezeigt werden. Hier sollte dem Nutzer bereits ein Überblick über die Hauptfunktion beziehungsweise ein schneller Zugriff auf die meist verwendeten Funktionen ermöglicht werden.
Die nächste Schicht sollte zunächst eine „Category view“ darstellen um einfaches übersichtliches Navigieren innerhalb der Daten zu gewährleisten. Diese ist bei weniger komplexen Gebieten nicht zwingend erforderlich.
Die letzte Schicht wird durch sogenannte „Detail/edit views“ gestellt. Hier soll der Nutzer wie der Name schon sagt mit den Details seiner ausgewählten Funktion oder Information konfrontiert werden.

Innerhalb der „Top level views“ sollte zudem eine schnelle Navigation, zum Beispiel durch verschiedene direkt auswählbare Tabs, Dropdown-Menüs oder das sogenannte „swipen“, bereitgestellt werden. Um die Navigation noch weiter zu vereinfachen und übersichtlicher zu gestalten, wird die Verwendung der in Android 3.0 eingeführten Action Bar nahe gelegt.

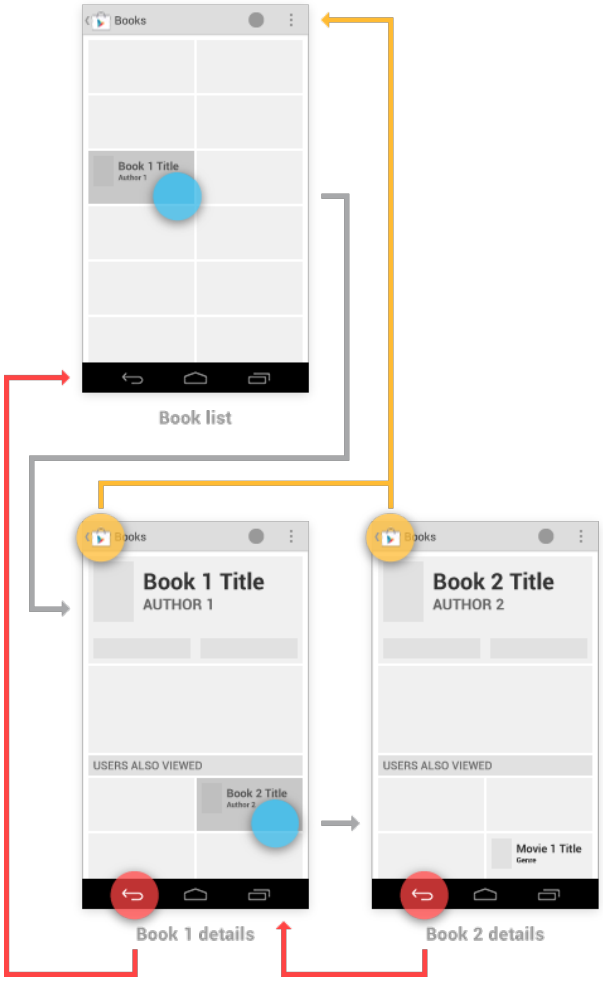


Abbildung 4.1: Navigation innerhalb einer Anwendung [Andi]

Eine konsistente Navigation ist essentiell für die Verwendung einer Applikation. Dabei kommen für das Zurückkehren zu vorherig angezeigten Daten die Zurück-Taste des Gerätes zum Einsatz und für eine hierarchische Navigation die Hoch-Taste aus der folgend beschriebenen Action Bar. Abbildung 4.1 soll dieses Schema noch einmal verdeutlichen.

4.2 Action Bar

Eine Action Bar ist ein Bildschirmelement, welches sich am oberem Bildschirmrand befindet. Mit Hilfe einer Action Bar kann die aktuelle Position innerhalb einer Applikation vermerkt werden, die Navigation erleichtert werden und es können häufig verwendete Aktionen in den Vordergrund gestellt und schnell erreichbar gemacht werden. Die Action Bar soll dem Benutzer ein Interface bieten, welches er aus anderen Applikationen bereits kennt und somit die Handhabung erleichtern

Abbildung 4.2: Design einer Action Bar [Anda]

soll.

Abbildung 4.2 zeigt das Design einer Action Bar und wie sie in einer Anwendung eingesetzt werden kann.

1. Im linkem Bereich wird im Normalfall das Icon der Applikation sowie die Überschrift der aktuellen Position innerhalb der Anwendung angezeigt. Durch Berührung der Überschrift sollte entweder über ein Dropdown-Menü eine schnelle Navigation zu einer anderen Position oder das Zurückkehren zur vorherigen Ansicht ermöglicht werden. Bei letzterem sollte dies zusätzlich durch einen kleinen Pfeil links des Icons, der sogenannten Hoch-Taste gekennzeichnet werden.

2. Im mittlerem Bereich können je nach Größe des Bildschirms einige häufig verwendete, wichtige oder typische Optionen mittels Icons bereitgestellt werden. Die Anzahl der Icons wird dabei automatisch über eine Prioritätenliste durch das Betriebssystem an die Bildschirmgröße angepasst. Platz bedingt nicht mehr passende Icons werden dabei in den dritten Bereich verschoben.

3. Am rechtem Rand werden weitere Optionen über ein Optionsmenü angezeigt. Hier können nicht so wichtige beziehungsweise nicht so häufig verwendete Aktionen untergebracht werden, sowie die verschriftlichte Version der aus Platzgründen nicht mehr passenden Icons des mittlerem Bereich.

4.3 Layout und Gesten

Das Layout einer Anwendung sollte sich immer an die Größe des verwendeten Bildschirms, sowie dessen Orientierung anpassen. Der erweiterte Platz eines größeren Gerätes sollte immer möglichst optimal genutzt werden und im Gegenzug dazu sollte ein kleinerer Bildschirm nicht mit zu vielen Informationen überflutet werden und so unübersichtlich erscheinen. Durch Android Fragments bietet sich dem Entwickler die Möglichkeit die Anwendung modular aufzubauen und an die

verschiedenen Geräte anzupassen. Dabei bieten sich für größere Geräte folgende Möglichkeiten:

- Mehrere Bildschirmansichten zu einer kombinieren
- Strecken und Stauchen von verwendeten Fragments
- Übereinander oder nebeneinander stapeln der Fragments
- Erweitern oder Reduzieren von Inhalten
- Zeigen oder Verstecken von einzelnen Elementen

Die verwendeten Gesten zur Navigation und zur Interaktion mit der Anwendung sollten, wie aus anderen Anwendungen bekannt, den Standards entsprechen. Sie sollten intuitiv sein und über die Applikation hinweg mit den Gesten des Betriebssystem persistent sein, sodass der Nutzer durch nachahmen von bekannten Gesten aus anderen Applikationen eine entsprechende Aktion erwarten kann.

4.4 Dialoge und Benachrichtigungen

Einige Interaktionen sind trotz Einhaltung der Standards dennoch nicht vom ersten Moment an für den Nutzer transparent. In einigen Situationen ist es daher sinnvoll den Nutzer darüber zu informieren was gerade passiert ist oder was als nächstes passieren wird, wenn eine bestimmte Aktion durchgeführt wird. Dies kann zum einen vor der Aktion durch einen Bestätigungsdialog erreicht werden, zum anderem nach der Aktion durch eine für kurze Zeit eingeblendete Benachrichtigung.

Wann eine solche Abfrage oder Bestätigung angezeigt werden sollte, ist durch die Android Design Richtlinien relativ eindeutig geregelt. Eine Bestätigung der Aktion sollte in folgenden Situationen eingefordert werden:

- Die Aktion hat signifikante Konsequenzen. (Beispiel: Unwiderrufliches Löschen von Daten, Belastung der Kreditkarte)
- Die Aktion kann leicht versehentlich ausgeführt werden und hat eventuell unangenehme Folgen (Beispiel: Die Aktion wird häufig hintereinander ausgeführt, die Interaktionszone ist sehr nah zu anderen gelegen oder nur sehr klein). Hier ist zu beachten, dass beispielsweise ein versehentlicher Seitenwechsel im Browser meist keine unangenehmen Folgen hat, das Löschen eines Datensatzes allerdings schon.

In folgenden Situationen sollte eine Benachrichtigung erfolgen:

- Die Aktion kann leicht versehentlich ausgeführt werden, aber es besteht die Möglichkeit sie einfach rückgängig zu machen. Hier sollte zu der Benachrichtigung direkt eine Funktion zur Widerrufung angeboten werden. (Beispiel: Das Löschen einer E-Mail im Standard Android E-Mail Programm)

- Die ausgeführte Aktion resultiert nicht in einer direkten visuellen Reaktion. (Beispiel: Das Anmelden über ein Benutzerkonto ist aus beliebigen Gründen gescheitert)

Außerdem ist bei Bestätigungsabfragen und Benachrichtigungen darauf zu achten, dass diese für den Nutzer unmissverständlich und eindeutig sind. Es sollte ebenfalls in den meisten Fällen auf technische Details verzichtet werden um den Nutzer nicht mit unnötigen Informationen zu verwirren. Wenn eine Benachrichtigung durch ein Problem hervorgerufen wird, sollte anstatt der Beschreibung des Problems eine Problemlösung vorgeschlagen werden.

4.5 Zugänglichkeit und Kompatibilität

Um eine Anwendung für den Nutzer leichter bedienbar, intuitiver und durchschaubarer gestalten zu können, sollte man auf einige weitere Standards des Betriebssystems zurückgreifen. In Android sind beispielsweise standardisierte Icons integriert, welche dem Entwickler zur Verfügung stehen. Diese Icons haben betriebssystemweit die gleichen Funktionen hinterlegt, wie das Teilen von Inhalten oder die Auswahl von Favoriten. Diese sollten auch in allen anderen Applikationen sinnvoll mit den gleichen Funktionen hinterlegt werden, sodass der Nutzer direkt weiß, welche Aktion er bei Betätigung der Schaltfläche zu erwarten hat.

Außerdem sollten interaktive Elemente unabhängig von der Bildschirmgröße eine Mindestgröße besitzen, sodass es dem Nutzer ermöglicht wird diese an jedem Gerät einfach und zielgenau zu betätigen und es nicht zu Frustration durch die Anhäufung von Fehleingaben kommt.

Kapitel 5

Grundlagen: Werkzeuge

In diesem Kapitel sollen kurz die Werkzeuge vorgestellt werden, welche zur Programmierung, Veranschaulichung und in Vorbereitung der Arbeit verwendeten wurden. FluidUI diente dabei zur Prototyp-Erstellung einiger Bildschirmansichten und deren Verknüpfung. Eclipse wird zur Programmierung der eigentlichen Applikation verwendet.

5.1 FluidUI

FluidUI ist ein Browser basiertes Werkzeug zur Erstellung von Prototypen. Es wurde im Juli 2012 von Dave Kearney und Ian Hannigan veröffentlicht [Flua] und basiert auf HTML5. [Flub] Mit FluidUI ist es möglich eine Android-Nutzeroberfläche zu erstellen und diese mit Interaktiven Elementen zu versehen. Es können mittels Drag and Drop fertige Android-UI-Elemente ausgewählt werden und entsprechend im Editor in die einzelnen Bildschirmanzeigen eingefügt werden. Die einzeln erstellten Bildschirmanzeigen können miteinander gekoppelt werden und somit kann eine gesamte Android Applikation modelliert werden.
Mit diesem Werkzeug ist es zudem möglich den erstellten Prototypen direkt auf ein Android Gerät zu laden und dort zu testen. Außerdem kann auf die verschiedenen Bildschirmgrößen und Eigenschaften von Tablets und Smartphones eingegangen werden. Mit Hilfe dieses Werkzeugs wurden in dieser Arbeit verschiedene Mock-Ups für die zu entwickelnde Anwendung modelliert.

5.2 Eclipse ADT Plugin

Eclipse ist eine Softwareentwicklungsumgebung (IDE), welche ursprünglich für die Programmiersprache Java genutzt wurde. [Ecl] Mittlerweile kann die quelloffene Software durch viele Erweiterungen ergänzt werden.

Eine dieser Erweiterungen sind die Android Development Tools (ADT). [Andd] Mit diesem Plugin ist es möglich Android Projekte zu erstellen. Dabei wird bereits automatisch die Android

typische Projektstruktur mit dessen standardisierten Ordnern und Bibliotheken erstellt. Außerdem können dort Pakete, welche auf die Android Framework API aufbauen, direkt integriert werden und Applikationen zur Laufzeit getestet werden. Die Tests der Applikationen werden dem Entwickler dabei über ein virtuelles Gerät ermöglicht oder auch direkt auf einem echtem Android Gerät. Der eingebettete Debugger ermöglicht dem Entwickler die vereinfachte Diagnostizierung und Auffindung von Programmierfehlern.
Die über XML erstellten Bildschirmansichten können zusätzlich für verschiedene Bildschirmgrößen vorgerendert werden und bei Bedarf in diesem Modus auch per Drag and Drop bearbeitet werden. Zudem können signierte .apk-Dateien exportiert werden, welche daraufhin veröffentlicht werden können. Eine .apk-Datei ist die Installationsdatei einer einzelnen Applikation.

Kapitel 6

Implementation

In diesem Kapitel soll nun nach gründlicher Aufklärung über die zu Grunde liegenden Techniken auf die eigentliche Implementation der im Zuge dieser Arbeit entwickelten Applikation eingegangen werden. Es wird beschrieben welche Herangehensweise gewählt wurde, wie die Anwendung aufgebaut ist und warum sie auf diese Weise entwickelt wurde. Außerdem werden weitere Einblicke in die genaue Handhabung der in den Kapiteln 2 und 3 beschriebenen Grundlagen gewährt.

6.1 Anforderungen und Umsetzung

Zunächst soll noch einmal ein Blick auf die zu entwickelnde Applikation geworfen werden. Es wird beschrieben welche Funktionen implementiert werden sollen, wie das Zusammenspiel der einzelnen Elemente aussehen soll und auf welche Details besonderer Wert gelegt werden soll.

Zuerst wird betrachtet welche Funktionen in der Applikation implementiert werden sollen. Die folgende Liste führt die einzelnen Funktionen anschaulich auf. Dabei wird für den Zugriff der Applikation auf die Set-Top-Box aus dem gleichem Netzwerk der Begriff *Heimnetzwerk* und für den Zugriff außerhalb dieses Netzwerks der Begriff *entfernt* verwendet.

- Die Applikation soll *von überall* einen sicheren Zugang zur Set-Top-Box gewährleisten. Dieser Zugang wird über den ConnectTV-Server geregelt, welcher über eine Nutzerdatenbank die Verbindung zu den einzelnen Boxen verwaltet. Dabei muss der Nutzer zuvor das Nutzerkonto mit der Set-Top-Box verbinden, dies wird als Paarung bezeichnet.
- Die mitgelieferte Fernbedienung soll durch die Applikation ersetzt werden können. Es sollen über die Applikation alle Funktionen und Befehle durchführbar sein. Die Fernbedienung soll dabei auch *entfernt* einsetzbar sein. Die Kommunikation erfolgt auch hier über den ConnectTV-Server.
- Durch den ConnectTV-Server wird ebenfalls ein EPG (Electronic Program Guide) bereitgestellt. Das Programm der einzelnen Sender, sowie Informationen zu den Übertragungen sollen übersichtlich und dennoch detailliert *überall* abrufbar sein.

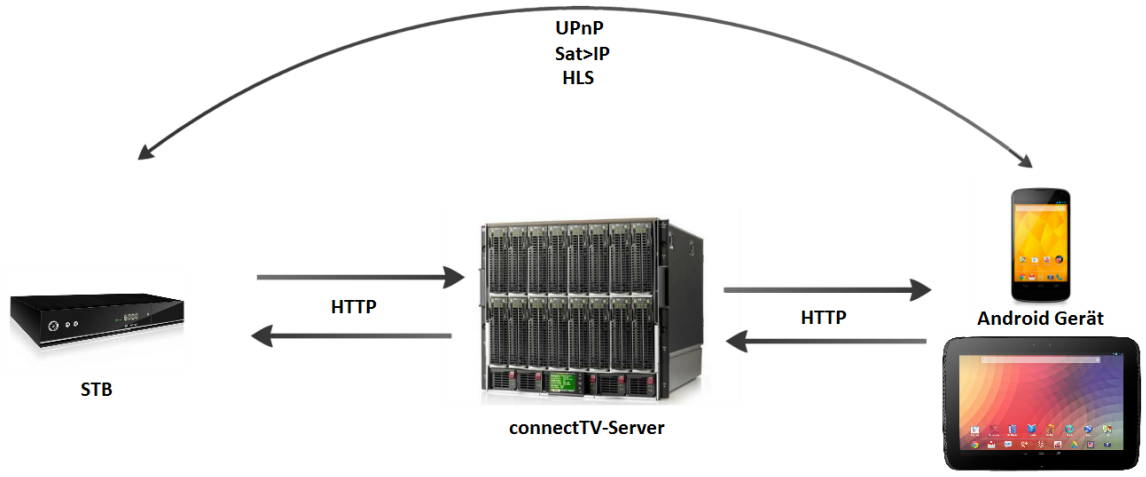


Abbildung 6.1: Übersicht über die Kommunikation der Geräte

- Mit Hilfe der Applikation soll dem Nutzer die Möglichkeit geboten werden von *überall* die Programmierungen der Set-Top-Box Aufnahmen zu organisieren. Es sollen Aufzeichnungen geplant, überprüft und gelöscht werden können. Diese Funktion wird ebenfalls über den ConnectTV-Server abgewickelt.
- Über die Applikation sollen die zuvor aufgezeichneten Sendungen im *Heimnetzwerk* betrachtet werden können. Hierfür verfügt die Set-Top-Box über einen UPnP-Medien-Server. In der Android Applikation soll daraufhin ein entsprechender Client implementiert werden. Diese Verbindung wird direkt zwischen den beiden Geräten hergestellt.
- Die Aufzeichnungen sollen ebenfalls *entfernt* abrufbar sein. Für diesen Fall verfügt die Set-Top-Box über einen Encoder, welcher die Aufzeichnung für das HTTP Live Streaming vorbereitet. Daraufhin wird es nach dem in Kapitel 2.6 beschriebenen Prinzip an die Android Applikation übertragen.
- Das aktuelle Live-Bild eines Programms soll ebenfalls im *Heimnetzwerk* auf dem verbundenem Android Gerät abrufbar sein. Hierfür soll das Sat>IP Protokoll verwendet werden. In der Set-Top-Box ist für dieses Ziel ein Sat>IP Server integriert.

Abbildung 6.1 zeigt eine Übersicht über die Kommunikation der beteiligten Geräte. Die Kommunikation der Set-Top-Box und der Android Geräte mit dem ConnectTV-Server wird dabei immer über HTTP abgewickelt. Der Hauptteil der Kommunikation läuft über die ConnectTV Schnittstelle.

Der in der Set-Top-Box integrierte UPnP Server und der Sat>IP Server sind, wie die Protokolle schon vorgeben, nur im gleichem Netzwerk ansprechbar und verwendbar. Hier wird immer die direkte Verbindung genutzt. Das HTTP Live Streaming wird zwar über den ConnectTV-Server angesprochen und aktiviert, die spätere Datenübermittlung verläuft allerdings nicht mehr über diesen, sondern auch direkt mit dem Android Endgerät.

Abbildung 6.2: Mock-Ups der EPG-Ansicht und einer Remotecontrol-Ansicht in FluidUI

Bei der Entwicklung soll ein besonderes Augenmerk auf die Android Design Richtlinien gelegt werden. Es soll versucht werden sie möglichst genau einzuhalten, um sie später über eine Evaluation angemessen bewerten zu können. Außerdem soll die Android API mit ihren Feinheiten für die Anpassung an verschieden Bildschirmgrößen und -auflösungen und ihrer teils automatischen Unterstützung in diesem Bereich genaustens betrachtet und möglichst effizient angewandt werden. Zum Schluss sollen auch die verwendeten Schnittstellen und Protokolle genauer untersucht werden, sowohl in Bezug auf ihre Funktionsweise als auch auf ihre Einfachheit in der Anwendung.

6.2 Prototyp mit FluidUI

Bevor mit der eigentlichen Implementation der Applikation begonnen wurde, wurden mit Hilfe von FluidUI verschiedene Mock-Ups der Bildschirmansichten angefertigt. Dies wurde stets unter Rücksprache mit der DiscVision GmbH getan, um die Wünsche des Auftraggebers möglichst detailliert erfüllen zu können. Wie bereits bei der Vorstellung von FluidUI angesprochen wurde, ist es mit diesem browserbasiertem Programm auf relativ einfache Art möglich die komplette Android Applikation als interaktiven Prototypen zu gestalten. Wie man in Abbildung 6.2 sehen kann, stimmt das Design eines solchen Entwurfs mit dem eines möglichen Endprodukts nahezu überein. Die in Android verwendbaren Elemente sind fast alle in FluidUI verfügbar und können somit zu einzelnen Ansichten zusammen gesetzt werden. Die einzelnen Elemente wurden mit Verlinkungen versehen, welche die Interaktion, wie im späteren Programm vorgesehen, imitieren. So konnten die einzelnen Bildschirmansichten miteinander verknüpft werden und zu einem gesamten Prototypen der Applikation zusammen gesetzt werden. Diese Prototypen konnten einfach

Abbildung 6.3: Teile eines Prototypen Aufbaus mittels FluidUI

per E-Mail an den Verantwortlichen bei DiscVision gesendet werden und von diesem wie eine Applikation auf dem Mobilgerät oder aber am Computer getestet werden. Wie die einzelnen Ansichten miteinander verbunden sind und wie einige dieser aussehen sollten, ist in Abbildung 6.3 dargestellt.

Die Erstellung von Bildschirmansichten speziell für Tablets ist mit FluidUI ebenfalls möglich. Dabei können die Eigenheiten eines Tablets in Form von Größe und Layout, sowie die speziellen Anpassungen des Designs ebenfalls in die Prototypisierung mit einbezogen werden.

Probleme bei der Erstellung eines gesamten Prototyps wurden dabei lediglich durch die Begrenzungen der kostenlosen Version hervorgerufen. Mit dieser Version war es innerhalb eines Projekts nur möglich zehn Bildschirmansichten miteinander zu verknüpfen. Aus diesem Grund mussten mehrere Projekte erstellt und mit den Verantwortlichen besprochen werden.

Wie bereits in den einzelnen Mock-Ups zu sehen ist, wurde schon bei der Prototypisierung versucht sich an die Android Design Richtlinien zu halten und die Ansichten diesen entsprechend zu gestalten. Die Action Bar ist in jeder Bildschirmansicht integriert und in den Prototypen wurde bereits versucht die Struktur und Navigation richtig zu realisieren. Die Layouts wurden speziell an die Bildschirmgrößen angepasst, sowie Dialoge eindeutig gestaltet und Icons sinnvoll und entsprechend dem Betriebssystem und anderen Applikationen gewählt. Gesten konnten im Prototyp noch nicht integriert werden, wurden aber mit den Verantwortlichen besprochen.

6.3 Externe Bibliotheken und Anwendungen

6.3.1 ActionBarSherlock

In Kapitel 4.1 wurde bereits darauf hingewiesen, dass die Android Action Bar erst ab der Android Version 3.0 in der API verfügbar ist. Da die Action Bar ein wichtiges Design- und Navigationselement darstellt, die zu entwickelnde Applikation allerdings auf Android 2.3 aufbauen sollte, musste eine andere Möglichkeit gesucht werden um diese integrieren zu können.
Diese Möglichkeit wurde in der externen Bibliothek *ActionBarSherlock* gefunden. [Jak] Diese wurde seit März 2011 von Jake Wharton entwickelt und liegt seit Juli 2013 in der Version 4.4.0 vor. Die aktuellste Version wird auch in dieser Arbeit verwendet. Durch Einbinden der ActionBarSherlock Bibliothek in das Applikationsverzeichnis ist es möglich die Android Action Bar auf Geräten bis zur Betriebssystem Version 2.x zu verwenden. Dabei wird automatisch in Fällen von Version 3.0 und höher die native Android Action Bar verwendet und bei früheren Versionen die Action Bar fast deckungsgleich durch eine Eigenimplementation ersetzt.

Zur Einbindung der Action Bar erben die Klassen von der `SherlockActivity` anstatt der normalerweise in Android verwendeten Klasse `Activity`. Die Fragments erben entsprechend von der Klasse `SherlockFragmentActivity`. Es wird also einfach ein Sherlock vor die normalen Klassennamen gesetzt. Eine Instanz der Action Bar erhält man dann nicht wie üblich über die Methode `getActionBar()`, sondern über die Methode `getSupportActionBar()`. Mit dieser Instanz kann daraufhin wie mit einer normalen Action Bar gearbeitet werden. Die Bibliothek bietet nahezu alle Funktionen und Eigenschaften der normalen Action Bar und wird auf die gleiche Weise verwendet.

6.3.2 MX Player

Über die zu entwickelnde Applikation sollten auch Videos abgespielt werden können. Hierfür bestand zum einen die Möglichkeit einen externen Player zu verwenden und zum anderem hätte ein eigener Player entwickelt werden können. Da letzteres ein zu großer Mehraufwand gewesen wäre und Android außerdem explizit dafür ausgelegt ist, die Fähigkeiten anderer Applikation zu nutzen, wurde die Verwendung eines externen Players vorgezogen. Dabei trat das Problem auf, dass nicht nur eine lokale Datei abgespielt werden musste. Ebenfalls mussten auch MPEG-Dateien via UPnP gestreamt werden können, ein HTTP Live Stream übergeben werden können

und ein für Sat>IP üblicher RTSP-Stream wiedergegeben werden können. An dieser Funktionsbreite scheitert der im Android Betriebssystem integrierte Videoplayer und fällt somit als Wiedergabeapplikation aus.

Nach einiger Recherche wurde der *MX Player* als die geeignetste Anwendung ausgewählt. [Goob] Der MX Player vereint alle nötigen Funktionen und unterstützt dabei alle hier verwendeten Protokolle. Weiterhin war die Dokumentation der API sehr präzise, umfangreich [MX] und mit einigen Beispielen hinterlegt. Zudem ist eine kostenlose Version des Players im Google Play Store verfügbar, sodass der Nutzer keine weiteren Kosten zu erwarten hat.

Wenn der Nutzer nun innerhalb der Applikation ein Video abspielen möchte wird zunächst geprüft ob der MX Player auf dem Gerät installiert ist. Listing 6.1 zeigt wie innerhalb der Methode `isAppInstalled()` mit Hilfe des im System integrierten `PackageManager` nach der Applikation gesucht wird. Dabei wird der `packageName`, in diesem Fall „com.mxtech.videoplayer.ad“ für die kostenlose beziehungsweise „com.mxtech.videoplayer.pro“ für die kommerzielle Version des Players, in Form eines Strings übergeben.

```
1 public boolean isAppInstalled(Activity a, String packageName) {
2     try {
3        a.getPackageManager().getPackageInfo(packageName, PackageManager.GET_ACTIVITIES);
4        return true;
5     } catch (PackageManager.NameNotFoundException e) {
6        return false;
7     }
8 }
```

Listing 6.1: Überprüfung ob der MX Player installiert ist

Ist der Player auf dem Gerät installiert, wird das Video automatisch mit diesem abgespielt. Ist er nicht installiert, kann der Nutzer sich einfach zum Google Play Store weiterleiten lassen und den Player von dort aus installieren. Dabei kann wieder direkt über den `packageName` die gewünschte Anwendung übergeben werden. Die Installation wird dabei über die installierte Applikation des Google Play Stores angefordert. Ist der Google Play Store auf dem Gerät nicht installiert, wird der systeminterne Browser auf die entsprechende Internetseite des Stores weitergeleitet(siehe Listing 6.2).

```
1 final String packageName = "com.mxtech.videoplayer.ad";
2 try {
3     f.startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("market://details?id=" +
          packageName)));
4 } catch (android.content.ActivityNotFoundException anfe) {
5     f.startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("http://play.google.com/
          store/apps/details?id=" + packageName)));
6 }
```

Listing 6.2: Starten des Google Play Stores zur Installation des MX Players

6.4 Bildschirmansichten

In den folgenden Sektionen wird ein Blick auf die einzelnen Bildschirmansichten der Applikation geworfen. Dabei soll auf die Umsetzung, Datenverwaltung, und auf das Design eingegangen werden. Jede Ansicht wird dabei in ihrem Portrait und ihrem Landscape Layout betrachtet, falls dort individuelle Anpassungen vorgenommen wurden. Diese Arbeit hat einen Schwerpunkt dahingehend liegend, die Layouts der Ansichten dynamisch an die Größe eines Tablets anzupassen. Im Laufe der Entwicklung der Applikation wurde jedoch festgestellt, dass es sinnvoller erscheint die Ansicht an die Orientierung des Bildschirms anzupassen, da viele Smartphones auch über einen genügend großen Bildschirm verfügen um auf diese Weise Inhalte gut darstellen zu können. Aus diesem Grund wird in dieser Arbeit nicht mehr zwischen Smartphone und Tablet Ansichten unterschieden, sondern zwischen den Orientierungen im Portrait und im Landscape Mode. Zudem sind einige der folgenden Bildschirmansichten in ihrem Aufbau sehr ähnlich, daher wird in diesen Fällen auf vorherige Erklärungen verwiesen und nur auf die Besonderheiten und Eigenheiten dieser verwiesen.

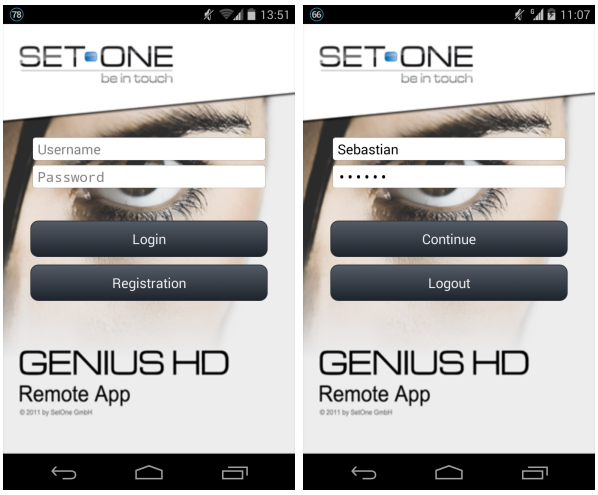
6.4.1 Anmeldung

Die Bildschirmansicht der Anmeldung entspricht vom Layout einem aus vielem Bereichen bekanntem Aufbau. Wie in Abbildung 6.4 zu sehen ist, werden dem Nutzer zwei editierbare Textfelder für den Nutzernamen und das Passwort bereitgestellt, sowie zwei Schaltflächen für die Anmeldung und die Registrierung eines neuen Nutzers im System. Wenn die Anmeldeansicht in den Vordergrund kommt, wird überprüft ob der Nutzer sich bereits angemeldet hat und bereits eine gültige Session besteht. Ist dies der Fall werden die Nutzerdaten automatisch in die jeweiligen Textfelder eingetragen und die Beschriftung der Schaltflächen sowie die dahinterliegende Logik wird dem entsprechend angepasst. Die Schaltflächen ermöglichen dem Nutzer in diesem Fall die vorhandene Session weiterzuführen oder sich abzumelden. Die Nutzerverwaltung wird dabei durch den ConnectTV-Server besorgt.

Der ConnectTV-Server ist gegen eine Vielzahl von Falscheingaben abgesichert. Bei der Registrierung eines Nutzers wird überprüft ob der Nutzername bereits vergeben ist und ob der Name oder das Passwort zu kurz ist. Ein Minimum von 5 Zeichen ist in beiden Fällen einzuhalten. Bei der Anmeldung wird zum einen der Name, zum anderem das zugehörige Passwort überprüft. Zudem wird in der Antwort des Servers angegeben ob die Set-Top-Box online ist und bereit ist Befehle entgegen zu nehmen. Dies ist für den Entwickler eine wichtige Information, um entsprechend darauf reagieren zu können. In der Applikation wird zu jeder Antwort des Servers eine entsprechende Benachrichtigung angezeigt, sodass der Nutzer ein mögliches Problem genau einsehen kann.

Die Nutzerdaten werden am Android Gerät durch die systeminternen `SharedPreferences` gespeichert und geladen. Ein Beispiel in den Grundlagen (siehe Abschnitt 3.3.7) beschreibt dies bereits und soll somit nicht noch ein weiteres mal aufgeführt werden.

Ist das Anmelden beziehungsweise das Abmelden, Registrieren oder Fortführen der aktuellen



(a) Unangemeldet **(b)** Angemeldet

Abbildung 6.4: Ansichten des Anmeldebildschirms

Session aus irgendwelchen Gründen nicht möglich, werden dem Nutzer über angezeigte Benachrichtigungen, in Android sogenannte `Toasts`, der mögliche Grund dafür mitgeteilt. Beispielsweise wird bei der Eingabe eines nicht zum Nutzer gehörigen Passworts dies entsprechend mit einer kurzen Meldung angezeigt. Oder falls es Verbindungsprobleme mit dem Server geben sollte, wird dies ebenfalls entsprechend angemerkt und der Nutzer gleichzeitig dazu aufgefordert seine Netzwerkverbindung zu überprüfen. Diese Benachrichtigungen entsprechen dabei immer den Android Design Richtlinien. Außerdem wird dementsprechend während einer Serveranfrage eine Benachrichtigung inklusive Ladeanzeige eingeblendet, sodass der Nutzer Feedback dazu erhält, dass im Hintergrund etwas passiert und die Applikation weiterarbeitet. Ist der Nutzer angemeldet wird er auf den Hauptbildschirm weitergeleitet.

6.4.2 Hauptbildschirm

Der Hauptbildschirm ist im Grunde genommen nur eine Hülle für die gleich folgenden EPG-, Planer- und Aufnahmeansichten. Denn diese Ansichten werden als Fragmente in den Hauptbildschirm eingebunden um dort variabel einsetzbar zu sein.

Am Beispiel der EPG Ansicht aus den Abbildungen 6.5 oder auch 6.6 sind die immer wieder auftauchenden Elemente des Hauptbildschirms erkennen. Darunter fallen die sich im oberen Bereich befindliche Action Bar und die sich am unterem Teil des Bildschirms angeordnete grüne Statusanzeige und die Hersteller- und Partnerbanner.

Die Statusanzeige überprüft die Verbindung zur Set-Top-Box in regelmäßigen Abständen. Die Abstände werden bei Verbindungsproblemen klein gehalten um möglichst schnell zu erfahren ob die Verbindung wieder hergestellt wurde. Bei erfolgreicher Verbindung wird der Abstand

größer gewählt um nicht zu häufig Anfragen zu stellen und somit das Netz und somit auch die Akkulaufzeit nicht unnötig zu belasten.

Die Action Bar wird immer an die im mittlerem Bereich angezeigten Fragment-Elemente angepasst. Dabei wird sowohl der angezeigte Titel, die Anzeige der Hoch-Taste als auch die Anzeige der Optionsmenüeinträge entsprechend angepasst. Die Anpassung der Action Bar kann von den einzelnen Fragmenten aus gesteuert werden und wird daher dort weiter beschrieben.

Die EPG-, Planer- und Aufnahmeansichten werden im Hauptbildschirm über Tabs verwaltet. Die Tabs können direkt ausgewählt oder durch „wischen“ gewechselt werden. Das Wechseln der Ansichten durch Wischen wird über einen `FragmentPagerAdapter` geregelt, welcher die Wisch-Geste erkennt und dementsprechend die Ansichten der von ihm verwalteten Fragmente aufruft. Wie man in den Abbildungen 6.6 oder 6.7 (c) sehen kann wird die Tableiste im Landscape Modus in die Action Bar integriert. Bei Platzmangel in der Action Bar werden die einzelnen Tabs zusätzlich in einem Drop-Down-Menü verwaltet. Diese Funktion ist in der Action Bar systemintern integriert und kann bei richtiger Handhabung der Fragmente einfach aktiviert werden. Somit wird die Orientierung und Platzvergabe innerhalb der Action Bar automatisch verwaltet.

6.4.3 EPG

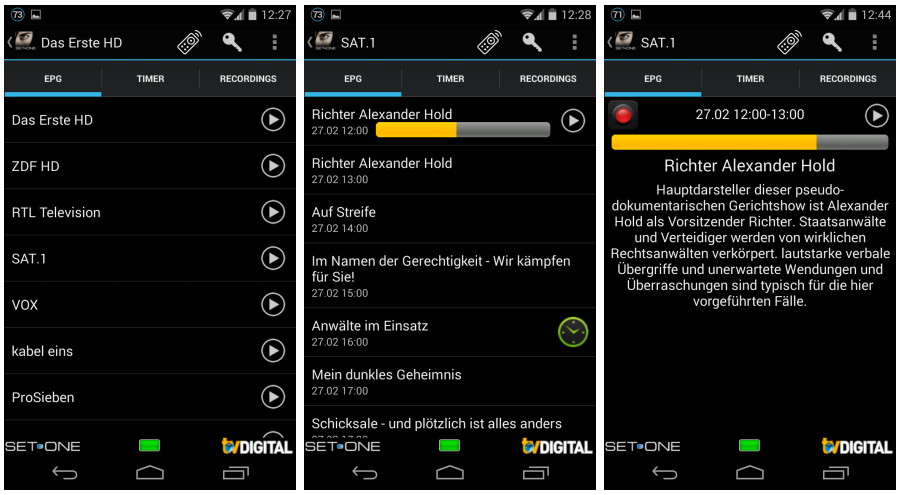
6.4.3.1 Allgemein

Beim Start des Hauptbildschirms erscheint als erstes die EPG Ansicht. Laut Android Design Richtlinien sollten zu Beginn der Applikation immer als erstes die am häufigsten genutzten und somit wichtigsten Informationen angezeigt werden. Diese Applikation soll hauptsächlich zum Programmieren von Aufnahmen, sowie zum Betrachten der TV Programme dienen, somit rücken diese Funktionen in den Vordergrund. Innerhalb der EPG Ansicht wird es dem Nutzer ermöglicht die aktuellen oder auch zu einem gewünschtem Zeitraum laufenden Sendungen einzusehen. Der Nutzer hat hier die Möglichkeit die Sender und dessen Programme zu durchsuchen, sowie sich detaillierte Informationen zu den einzelnen Sendungen einzuholen. Außerdem kann er in diesem Bereich Aufnahmen von Sendungen planen oder direkt ins Live Bild eines Senders springen.

Die Darstellung der EPG Informationen ist in drei Ebenen aufgeteilt. Abbildung 6.5 zeigt die drei Ebenen im Portrait Modus. In der ersten Ebene (a) werden alle auf der Set-Top-Box eingespeicherten Sender aufgelistet. Die Reihenfolge der Liste wird dabei automatisch durch die auf der Box gespeicherten Liste übernommen. In die zweite Ebene (b) gelangt der Nutzer durch das Auswählen eines Senders. In dieser Ebene wird ab dem aktuellen Zeitpunkt das auf diesem Sender laufende Programm aufgelistet. Wird dort eine Sendung ausgewählt, wird die dritte Ebene (c) aufgerufen, welche detailliertere Informationen zu dieser Sendung angezeigt.

6.4.3.2 Daten

Alle Daten werden über den ConnectTV-Server angefordert. Dabei werden nicht zum Start der Anwendung alle Daten komplett bezogen, sondern wenn benötigt, werden die entsprechenden



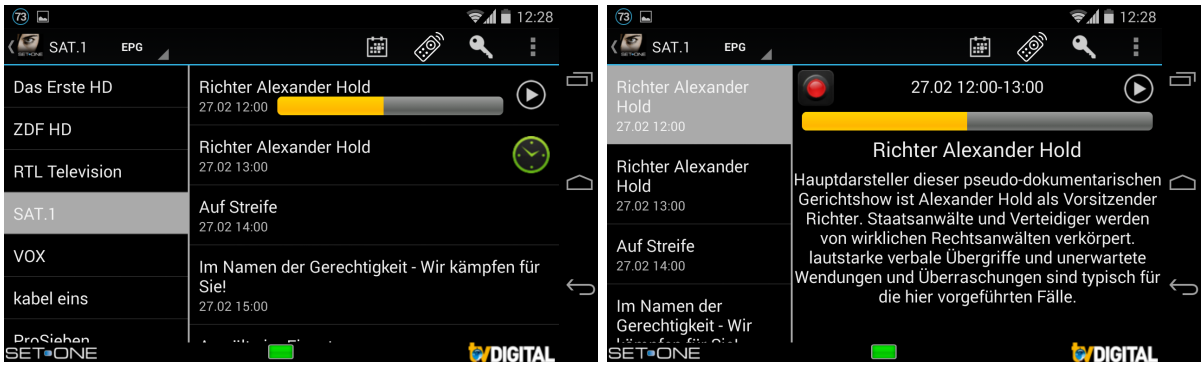
(a) Senderliste **(b)** Programmliste **(c)** Programminformation

Abbildung 6.5: Ansichten der EPG-Bildschirme (Portrait)

Daten nachgeladen. Den kompletten Datensatz vor dem Start zu laden, würde zu einer erheblichen Zeitverzögerung zu Beginn der Anwendung führen. Da davon ausgegangen wird, dass der Großteil der Nutzer meist gezielt etwas Einprogrammieren oder Abrufen will, würde dies zu einem erhöhten Mehraufwand führen. Durch das einzelne Nachladen der Informationen wird die Applikation zwar etwas langsamer zu bedienen sein, doch der Nutzen eines vorher komplett geladenen TV-Programms würde diesem nicht überwiegen.

Die Informationen werden mittels des im Abschnitt 2.3 beschriebenen Verfahrens via HTTP abgerufen. Da das Laden von Daten im gleichem Thread die reibungslose Nutzung des Nutzerinterfaces beeinträchtigen könnte, wird in Android zum Laden von externen Daten ein Hintergrund-Task gestartet. Im Falle dieser Arbeit wird dafür immer ein `AsyncTask` verwendet. So kann die Nutzeroberfläche weiterhin ohne Verzögerungen genutzt werden und die Daten nach Beendigung des Hintergrund-Tasks in die Oberfläche eingebunden werden.

Die Daten werden vom ConnectTV-Server in XML-Format zurückgeliefert, werden in der Anwendung im Hintergrund von einem XML-Parser geparst und im Model in einem passendem Format abgelegt. Dies ist im Falle des EPGs eine verschachtelte Liste. Diese Liste wird mit selbst erstellten Objekten gefüllt. Dabei wird zunächst eine Liste der Sender angelegt, jedes Listenelement eines Senders beinhaltet neben den Senderdaten eine Liste des Programms. Dieses Programm kann in einem weiterem Schritt mit zusätzlichen Informationen erweitert werden. Einmal geladene Daten werden im Model persistent gehalten und müssen somit bei erneutem Aufruf nicht ein weiteres mal geladen werden. Aus diesem Grund beinhaltet jedes Objekt eines Senders die Information über das letzte Update der Senderdaten. Nur wenn das Update zu weit in der Vergangenheit liegt, werden die Daten neu geladen.



(a) Sender- und Programmliste **(b)** Programmliste und -informationen

Abbildung 6.6: Ansichten der EPG-Bildschirme (Landscape)

6.4.3.3 Design

In den Listen sind zusätzlich Schaltflächen für den schnellen Start oder eine schnelle Übersicht einer Funktion eingebunden.
In jeder der drei Ebenen ist eine Abspiel-Taste integriert. Wenn der Nutzer sich das aktuelle Programm eines Senders live anschauen möchte, so kann er dieses direkt über die Abspiel-Taste starten. Diese Schaltfläche wird nur angezeigt wenn diese Funktion auch verfügbar ist. Da diese Funktion über das Sat>IP Protokoll zur Verfügung gestellt wird, ist eine Verwendung nur im Heimnetz möglich und wird somit auch nur dort angezeigt.
In der dritten Ebene wird dem Nutzer die Möglichkeit geboten die ausgewählte Sendung mit der Set-Top-Box aufzuzeichnen. Diese Funktion wird ebenfalls durch den ConnectTV-Server koordiniert. Dabei wird auf der Set-Top-Box darauf geachtet, ob eine Aufnahme der Sendung möglich ist und sich nicht mit anderen Aufnahmen überschneidet. Wenn eine nicht zulässige Überschneidung auftritt, wird eine entsprechende Antwort an die Applikation zurückgegeben. In der Android Applikation wird dann durch einen Dialog auf die Problematik hingewiesen und ein Lösungsvorschlag generiert. Dieser Dialog wurde wiederum versucht nach den Android Design Richtlinien zu gestalten.
In der zweiten und dritten Ebene wird zusätzlich ein Planer-Symbol eingeblendet, welches als Indikator für eine geplante Aufzeichnung dient und die folgenden Bedeutungen haben:

 Aufnahme eingeplant

 Aufnahme läuft

Innerhalb der Senderliste und der Programmdatenliste wird auch die aktuelle Scroll-Position der Liste gespeichert. Wenn ein Programm ausgewählt wird und später über die Zurück- oder Hoch-Taste zur Senderliste zurückgekehrt wird, wird die Liste wieder an der Stelle angezeigt wo

sie auch verlassen wurde. Dies bietet dem Nutzer ein besseres Navigationsverhalten und mehr Konsistenz in der Handhabung und entspricht somit ebenfalls den Android Design Richtlinien.

6.4.3.4 Layoutanpassung

Eine Anpassung des Layouts an den Landscape Modus ist eine weitere Eigenschaft, welches die Handhabung und Übersicht verbessern soll. Wie in den Abbildungen 6.6 (a) und (b) zu sehen ist, werden die einzelnen Ebenen dort strukturiert nebeneinander dargestellt. Außerdem werden die Abspiel-, Aufnahme- und Planer-Symbole nicht mehr in jeder Ansicht angezeigt, sondern immer nur maximal ein mal pro zusammengefasste Bildschirmansicht.
Diese Strukturierung der beiden Fragmente nebeneinander wird dadurch erreicht, indem man diesen Fragments ein „Super“-Fragment überordnet. Dieses Fragment wird anstatt eines der Fragments der drei Ebenen geladen und bietet zwei weiteren Fragments einen vordefinierten Platz.

```
1  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:tools="http://schemas.android.com/tools"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent"
5      android:baselineAligned="false"
6      android:orientation="horizontal"
7      tools:context=".MainActivity" >
8
9      <FrameLayout
10         android:id="@+id/fragment_container_epg1"
11         android:layout_width="fill_parent"
12         android:layout_height="match_parent"
13         android:layout_weight="2" />
14
15     <ImageView
16         android:id="@+id/divider"
17         android:layout_width="2dp"
18         android:layout_height="fill_parent"
19         android:background="@android:color/darker_gray" />
20
21     <FrameLayout
22         android:id="@+id/fragment_container_epg2"
23         android:layout_width="fill_parent"
24         android:layout_height="match_parent"
25         android:layout_weight="1" />
26
27 </LinearLayout>
```

Listing 6.3: Landscape Layout eines „Super“-Fragments

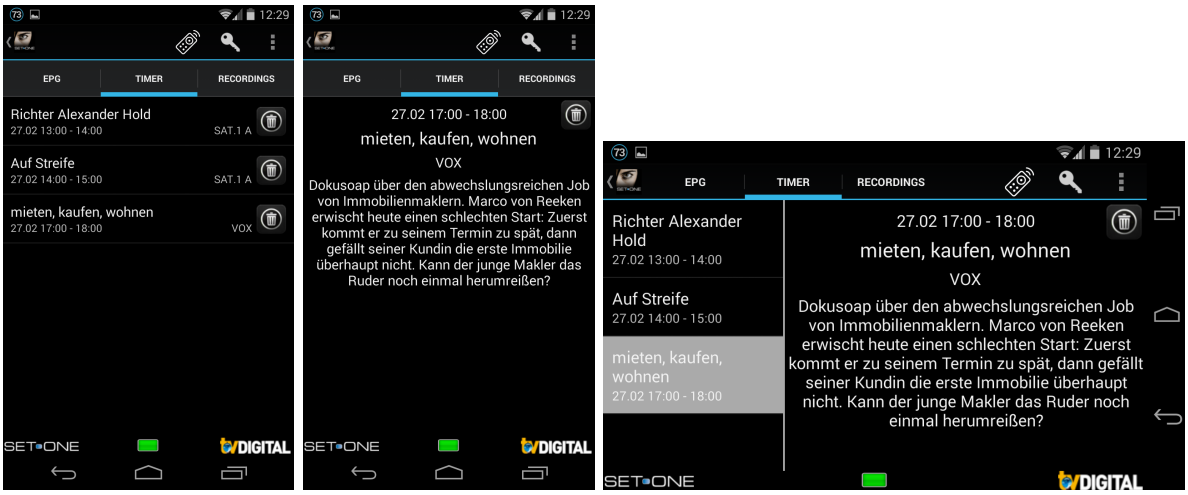
Listing 6.3 zeigt ein XML-Layout und Listing 6.4 zeigt die Aufrufe dieser, innerhalb eines „Super“-Fragments. Zunächst ist zu sagen, dass Android innerhalb der Ressourcen zwei Ordner besitzen kann, je einen für die Portrait und die Landscape Layouts. In diesen Ordnern wird für jede Orientierung ein „Super“-Fragment Layout des gleichen Namens abgelegt. Das Layout für den Landscape Modus ist in Listing 6.3 dargestellt. Es beinhaltet innerhalb eines `LinearLayouts` zwei horizontal nebeneinander angeordnete `FrameLayouts`. Diese bieten jeweils Platz für ein Fragment und werden durch eine `ImageView` getrennt, welches eine einfache dünne vertikale Linie beinhal-

tet. Durch den Parameter `layout_weight` in den Zeilen 13 und 25 wird die Gewichtung der einzelnen `FrameLayouts` angegeben. Da der Gewichtung des zweiten `FrameLayouts` eine doppelt so große Priorität gegeben ist wie der des ersten, wird diesem auch doppelt so viel Platz auf dem Bildschirm zugesprochen. Diese Zuweisung ist dabei immer abhängig von der Bildschirmgröße.

```
1  @Override
2  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
       savedInstanceState) {
3      ViewGroup root = (ViewGroup) inflater.inflate(R.layout.fragment_epg_super, null);
4
5      // clear backstack on rotate view
6      FragmentManager fm = getActivity().getSupportFragmentManager();
7      for (int i = 0; i < fm.getBackStackEntryCount(); ++i) {
8          fm.popBackStack();
9      }
10
11     EPGFragmentChannelList mEPGFragmentSenderliste = EPGFragmentChannelList.newInstance(
           getActivity());
12     FragmentTransaction transaction = getFragmentManager().beginTransaction();
13     transaction.replace(R.id.fragment_container_epg1, mEPGFragmentSenderliste).commit();
14
15     return root;
16 }
17
18 @Override
19 public void onActivityCreated(Bundle savedInstanceState) {
20     super.onActivityCreated(savedInstanceState);
21
22     // adapt view if in landscape mode
23     View epgDataFrame = getActivity().findViewById(R.id.fragment_container_epg2);
24     Model.getInstance().setDualPane(epgDataFrame != null && epgDataFrame.getVisibility()
           == View.VISIBLE);
25
26     if (Model.getInstance().isDualPane()) {
27         Model.getInstance().setEpgFirstScreen(true);
28         EPGFragmentEPGList mEPGFragmentSenderEPG = EPGFragmentEPGList.newInstance(0);
29         FragmentTransaction transaction = getFragmentManager().beginTransaction();
30         transaction.replace(R.id.fragment_container_epg2, mEPGFragmentSenderEPG).commit()
               ;
31     }
32 }
```

Listing 6.4: Layout Anpassung an die Orientierung

Während das Layout des Landscape Modes zwei `FrameLayouts` beinhaltet, beinhaltet das hier nicht aufgeführte, Portrait Layout nur eines und somit wird dort auch keine Abtrennung benötigt. Das Layout besitzt dabei die gleiche ID wie das erste `FrameLayout` im Landscape Ordner. Innerhalb des „Super“-Fragments (siehe Listing 6.4) wird in der Methode `onCreateView()` in Zeile 3 das Layout geladen. Das Android Gerät wählt automatisch anhand der aktuellen Orientierung das benötigte Layout aus dem richtigem Ordner. In Zeile 11 wird das Fragment für die Senderliste erstellt und in Zeile 13 in das erste `Framelayout` geladen. In der Methode `onActivityCreated()` wird daraufhin überprüft, welches Layout geladen wurde. Dafür wird in Zeile 23 nach dem zweitem `FrameLayout` gesucht, falls dies vorhanden ist, wird der `boolean`-Wert DualPane auf `true` gesetzt. Ist dies der Fall befindet sich das Gerät im Landscape Modus. Somit wird der benötigte Platz der Fragments entsprechend aufgeteilt und durch einen „Seperator“ getrennt. Das



(a) Planerliste **(b)** Planerinformation **(c)** Planerliste und -informationen

Abbildung 6.7: Ansichten der Planerbildschirme

zweite Fragment kann daraufhin geladen werden. Die Fragmente sind unabhängig voneinander verwendbar. Sobald im ersten Fragment ein Eintrag ausgewählt wird, wird jedoch im zweiten entsprechend der Inhalt angepasst.

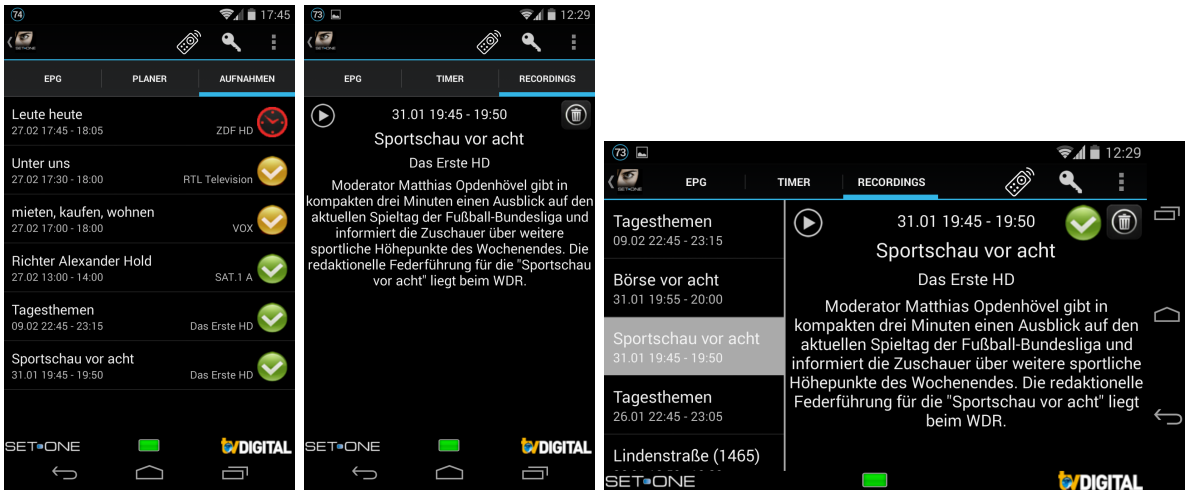
6.4.4 Planer

Im weiteren Verlauf der Bildschirmansichten wird keine weitere Sektionierung des Kapitels vorgenommen, sondern im Allgemeinen auf diese eingegangen. Es wird so vorgegangen, da die Ansichten sich in vielen Punkten ähneln und weitestgehend die gleichen Techniken zum Einsatz kommen.

Die Planer Ansicht zeigt eine Liste der geplanten Aufnahmen an. Diese ist nach der Uhrzeit der Sendestarts sortiert und bietet eine einfache Möglichkeit die geplanten Aufnahmen zu organisieren, sich noch einmal detaillierte Informationen der Sendungen einzuholen oder die Aufnahme abzubrechen.

Die Ansichten des Planers sind auf den Abbildungen 6.7 (a), (b) und (c) zu sehen. Hier gibt es zwei Ebenen, die Liste aller Planerelemente (a), sowie die detaillierten Informationen eines Elements (b). So ist es möglich in einer einzigen Landscape Ansicht (c) beide Ebenen zu vereinen und übersichtlich anzuzeigen.
In der Übersicht werden Datum, Uhrzeit und Sender aufgeführt. Die Daten werden auch hier vom ConnectTV-Server abgerufen, welcher eine separate Funktion für die Planerliste bereitstellt. Detailinformationen zu den Sendungen werden ebenfalls, wie auch in der EPG Ansicht, über eine zusätzliche Anfrage mit der entsprechenden Sendungs-ID vom Server abgerufen.

Im Grunde genommen ist die Datenverarbeitung und -haltung, das Grunddesign sowie die An-



(a) Aufnahmeliste **(b)** Aufnahmeinformation **(c)** Aufnahmeliste und -informationen

Abbildung 6.8: Ansichten der Aufnahmebildschirme

passung des Layouts an den Landscape Modus simultan zu den EPG Ansichten anzusehen und wird auf gleiche Art und Weise verarbeitet und gestaltet. So wird auch hier die Datenhaltung durch das Model persistent gehalten.

Dem Nutzer wird, wie aus der EPG Ansicht bekannt, über Schaltflächen innerhalb der Listenelemente die Möglichkeit geboten mit den einzelnen Listenelementen zu interagieren. Es wird über ein aussagekräftiges Symbol verdeutlicht, dass das Element über diese Schaltfläche gelöscht werden kann. Beim Drücken dieser Schaltfläche wird das Element nicht direkt gelöscht, respektive wird ein Dialog angezeigt, welcher den Nutzer um eine Bestätigung dieser Aktion bittet.
Mittels dieser zwei gerade aufgeführten Punkte wird ebenfalls versucht die Applikation den Android Design Richtlinien entsprechend zu gestalten. Zum einen werden aussagekräftige und aus anderen gebieten bekannte Symbole verwendet, um so die Zugänglichkeit und Kompatibilität zu verbessern. Zum anderem wird für Aktionen mit kritischen Folgen, hier das Löschen einer geplanten Aufnahme, über einen unmissverständlichen Dialog eine Bestätigung angefordert.
Außerdem wird auch hier eine erleichterte Navigation durch das Speichern der Scroll-Position innerhalb der Planerliste und eine kontrollierte Navigation mittels der Zurück- und Hoch-Taste gewährleistet.

6.4.5 Aufnahmen

Die Abbildungen 6.8 (a), (b) und (c) lassen bereits erahnen, dass der Aufbau der Aufnahmeansicht nahezu identisch zur Planer Ansicht ist. Dies ist auch weitestgehend der Fall. Es werden für das Grunddesign, die Datenverarbeitung und -haltung, sowie die Anpassung des Layouts an den Landscape Modus die gleichen Methoden wie schon in der EPG- und Planer Ansicht verwendet. Daher soll nun nur kurz auf die Funktionen dieser Bildschirmansicht, sowie auf einige Punkte

bezüglich der Android Design Richtlinien eingegangen werden.

Die Aufnahmeansicht verfügt auch über zwei Ebenen. Zum einen die Aufnahmeliste (a) und zum anderem die Elementinformationen (b). Diese können ebenfalls in einer Landscape Ansicht (c) zusammengefasst werden. Die Aufnahmeliste zeigt alle bereits aufgezeichneten Sendungen sortiert nach der Startzeit an. Hier hat der Nutzer außerdem eine Übersicht über den Status seiner Aufzeichnungen, zu welchem Zeitpunkt diese waren und auf welchem Sender diese ausgestrahlt wurden.
In der detaillieren Elementinformation einer Aufnahme kann der Nutzer diese wenn gewünscht löschen oder direkt anschauen. Die Wiedergabe der Aufzeichnungen erfolgt im Heimnetzwerk über das UPnP-Protokoll und entfernt über HTTP-Live-Streaming. Genaueres dazu folgt in den Abschnitten 6.7 und 6.8.

Um auch eine Persistenz innerhalb der Applikation zu erhalten, wird der Aufbau der Bildschirmansicht simultan zu der EPG und der Planer Ansicht gestaltet und es werden die gleichen grafischen Elemente für gleichartige Aktionen verwendet. Zum Löschen einer Aufzeichnung wird wieder das aussagekräftige und bekannte Symbol des Mülleimers verwendet und zur Wiedergabe wird wieder die bekannte Abspiel-Taste verwendet. Die kritische Aktion des Löschens muss dabei wieder durch einen Dialog bestätigt werden.

Neben den bekannten Symbolen werden hier noch Icons zur Anzeige der Status verwendet. Dabei werden sie jeweils mit der folgenden Bedeutung hinterlegt:

 Aufnahme läuft

 Aufnahme erfolgreich

 Aufnahme vom Nutzer abgebrochen

 Aufnahme aufgrund eines Fehlers fehlgeschlagen

 Aufnahme durch Timeshift Funktion und somit nicht abspielbar

6.4.6 Videoplayer

Zum Abspielen der Video Dateien wird ein externer Videoplayer verwendet. Es wird der in Abschnitt 6.3.2 vorgestellte MX Player verwendet. Der Start des Players erfolgt, wie in Listing 6.5 zu sehen, über einen Intent.

Zunächst wird ein neuer Intent erstellt und die URL, welche im `String` Format vorliegt entsprechend in eine URL geparst. Dem Intent werden in Zeile 3 die Daten und der Typ der Applikation zugeordnet. In diesem Fall wird der Typ mit „application/*“ gesetzt, was bedeutet, dass der Typ des Empfängers eine Applikation sein soll. Als Extras werden dem Intent zusätzlich der Titel der Datei hinzugefügt und es wird ein „Return Result“ erwartet. Das heißt, wenn der Player beendet wird, wird die `onActivityResult()` Methode der Activity aufgerufen, welche den Intent gestartet hat. So kann auf die Rückkehr zur Applikation entsprechend reagiert werden. Außerdem wird in den Zeilen 7-9 noch das Extra „Headers“ hinzugefügt, welches in diesem Fall genau angibt welche Applikation gestartet werden soll.
Zum Ende wird mittels der optionalen Methode `setPackage()` noch genauer spezifiziert in welchem Paket die auszuführende Applikation zu finden ist. Dies verbessert die Geschwindigkeit der Ausführung und erhöht die Sicherheit die richtige Anwendung zu starten. Zum Schluss wird die Activity anhand des gerade erstellten und modifizierten Intents gestartet.

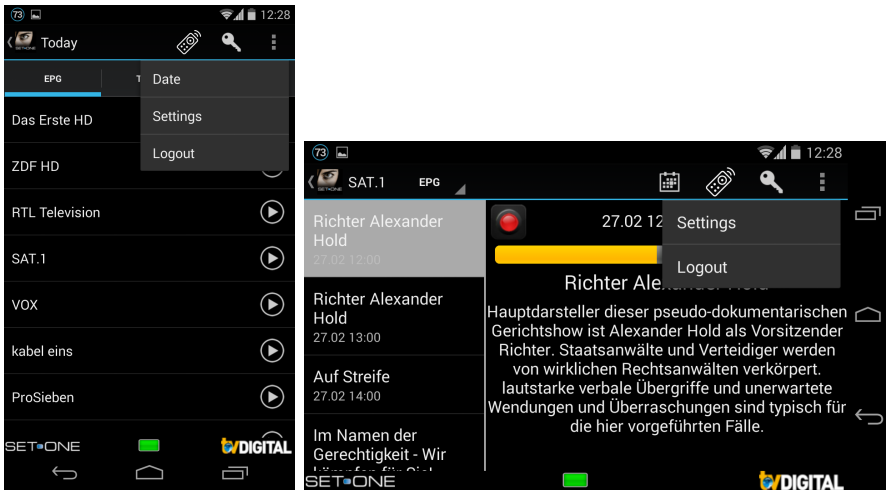
```
1  Intent intent = new Intent(Intent.ACTION_VIEW);
2  Uri videoUri = Uri.parse(url);
3  intent.setDataAndType( videoUri, "application/*" );
4  intent.putExtra( "return_result", true );
5  intent.putExtra( "title", Model.getInstance().getRecordinglist().get(
6      Model.getInstance().getChosenRecordingListPosition()).getName());
7  String[] headers = new String[] { "User-Agent", "MX Player Caller App/1.0",
8      "Extra-Header", "911" };
9  intent.putExtra( "headers", headers );
10 intent.setPackage( "com.mxtech.videoplayer.ad" );
11 startActivityForResult( intent, 0 );
```

Listing 6.5: Start des externen MX Players

6.4.7 Menü

Die Bildschirmansichten des EPGs, des Planers und der Aufnahmen verfügen alle über ein Optionsmenü. Seit der Android Version 3.0 können die Optionen in die Action Bar integriert werden. Wie in den Abbildungen 6.9 (a) und (b) zu sehen ist, können die Menüpunkte in Form von Icons und durch schriftliche Einträge dargestellt werden. Die einzelnen Menüpunkte können direkt im Source Code erstellt, oder über eine XML-Datei eingebunden werden. In dieser Arbeit wurde letztere Methode angewandt. Die Menüpunkte werden innerhalb des Ressourcen Ordners im Order *menu* abgelegt. In Abschnitt 3.3.5 ist bereits der Aufbau einer solchen Datei vorgestellt worden und soll daher hier nicht ein weiteres mal besprochen werden.

Da die Menüs der verschiedenen Fragmente sich nur minimal unterscheiden, wird das Menü in der Haupt-Activity geladen und wird daraufhin bei Anzeige der einzelnen Fragments entsprechend angepasst. Durch das Überschreiben der `onCreateOptionsMenu()` Methode kann die XML Datei über den `MenuInflater` geladen werden. Um das Menü nun an die einzelnen Fragments anpassen zu können, muss die aktuell angezeigte Seite des `FragmentPagers` immer bekannt sein. Mit Hilfe des `FragmentPagerAdapter` kann über einen Listener ermittelt werden welches Fragment gerade aktiv ist und in der Activity zwischengespeichert werden.



(a) Menü (Portrait) **(b)** Menü (Landscape)

Abbildung 6.9: Ansichten der Menüs

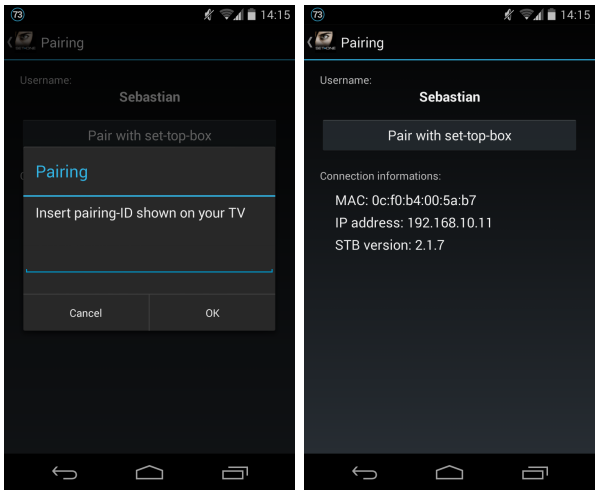
```
1 @Override
2 public boolean onPrepareOptionsMenu(Menu menu) {
3     if (currentPage == 0) {
4         menu.findItem(R.id.action_settings_date).setVisible(true);
5     } else {
6         menu.findItem(R.id.action_settings_date).setVisible(false);
7     }
8     return super.onPrepareOptionsMenu(menu);
9 }
```

Listing 6.6: Anpassung des Optionsmenüs

Mit diesem Wissen kann in der überschriebenen `onPrepareOptionsMenu()` Methode die Seite überprüft und das Menü entsprechend angepasst werden. Listing 6.6 zeigt wie ein Optionsmenüpunkt durch dieses Verfahren ein und ausgeblendet wird. Innerhalb der Variablen des Optionsmenüs kann mittels der in XML gesetzten ID der entsprechende Menüeintrag gesucht werden und sichtbar (Zeile 4) oder unsichtbar (Zeile 6) geschaltet werden. Danach muss noch wie in jeder überschriebenen Methode Androids die eigentliche Methode mittels `super` aufgerufen werden.

Die Optionsmenüitems müssen des weiteren noch mit Aktionen hinterlegt werden. Dafür wird die Methode `onOptionsItemSelected()` wie in Listing 6.7 überschrieben. Jedes der Menüitems besitzt eine ID und kann über eine `switch-case`-Abfrage eine Aktion zugeordnet bekommen. In diesem Fall wird beispielsweise bei Auswahl der `R.id.action_settings_remotecontrol` in den Zeilen 4-5 die Activity der Fernbedienung gestartet.
Außerdem muss in diesem Bereich auch auf die sich in der Action Bar befindlichen Hoch-Taste eingegangen werden. Diese hat innerhalb des Android Systems die ID `android.R.id.home`. In diesem Fall wird der Druck der Zurück-Taste imitiert.

Wie in den Abbildungen 6.9 (a) und (b) zu sehen ist, wird die Anzeige der Menüpunkte innerhalb der Action Bar an den sich dort befindlichen Platz angepasst. Im Portrait Modus sind dort



(a) Dialog bei Erststart **(b)** Übersicht

Abbildung 6.10: Ansichten des Paarungsbildschirms

nur zwei Icons und drei schriftlichen Einträge zu sehen. Wohin entgegen im Landscape Modus, aufgrund von mehr Spielraum drei Icons und zwei schriftliche Einträge zu sehen sind. Dies wird durch die Action Bar automatisch verwaltet und an die jeweilige Bildschirmgröße angepasst. Dies bezüglich werden, wie in den Grundlagen bereits beschrieben, die einzelnen Menüeinträge mit Icons und Prioritäten der Anzeigereihenfolge versehen. Der Entwickler hat also die Möglichkeit selbst zu entscheiden welche Optionen er für wichtig hält und als Icon angezeigt werden soll. Die Auswahl der Icons wurde dabei wieder an die dem Nutzer bekannten Symbole angelehnt um somit eine bessere Zugänglichkeit und Kompatibilität nach den Android Design Richtlinien zu erlangen.

```
1  @Override
2  public boolean onOptionsItemSelected(MenuItem item) {
3      switch (item.getItemId()) {
4      case R.id.action_settings_remotecontrol:
5          startActivity(new Intent(this, RemoteActivity.class));
6          break;
7      case android.R.id.home:
8          super.onBackPressed();
9          break;
10     default:
11         break;
12     }
13     return super.onOptionsItemSelected(item);
14 }
```

Listing 6.7: Optionsmenü mit Aktionen hinterlegen

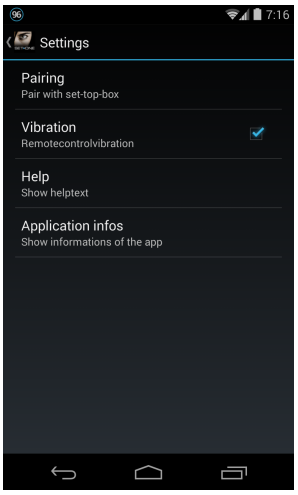


Abbildung 6.11: Ansicht des Einstellungsbildschirms

6.4.8 Paarung

Um die Set-Top-Box mit der Anwendung zu verknüpfen müssen diese gepaart werden. Da das Paaren in der Bedienungsanleitung der Set-Top-Box beschrieben wird, wird innerhalb der Applikation auf eine weitere Beschreibung verzichtet. Zur Verknüpfung der beiden Systeme wird zunächst über die Applikation ein Konto auf dem ConnectTV-Server erstellt. Dies geschieht über die Anmeldeansicht. Wenn der Anwender sich einen neuen Nutzeraccount erstellt, wird dieser bei der Anmeldung automatisch zum Paarungsbildschirm weitergeleitet. Dort wird er aufgefordert die an der Set-Top-Box im Optionsmenü erstellte Paarungs-ID einzugeben (siehe Abbildung 6.10 (a)). Ist diese eingetragen, wird der Nutzer dauerhaft mit dieser Set-Top-Box verknüpft und immer automatisch bei der Anmeldung mit dieser Box verbunden. Die Verknüpfung des Nutzers mit der Set-Top-Box kann nur im Optionsmenü dieser zurückgesetzt werden.

Ist der Nutzer mit der Set-Top-Box verbunden, kann er unter den Einstellungen die Paarungsübersicht aufrufen. In Abbildung 6.10 (b) ist diese Übersicht dargestellt. In dieser ist ersichtlich welcher Nutzer angemeldet ist und wie die Version der Set-Top-Box Software, sowie die MAC- und IP-Adresse der Box lauten. Diese Daten werden vom ConnectTV-Server abgerufen.

6.4.9 Einstellungen

Die Einstellungen können über das Optionsmenü aufgerufen werden. Wie Abbildung 6.11 zeigt sind diese einfach gehalten, aber dem Android Standard entsprechend. Die Activity ist dabei eine speziell für die Einstellungen einer Anwendung vergesehene `SettingsActivity`. Die Einstellungen können direkt im Source Code generiert werden, werden aber im Normalfall durch eine XML-Datei, welche innerhalb der Ressourcen im Ordner „xml“ abgelegt ist, generiert.

Listing 6.8 zeigt den Aufbau einer leicht verkürzten XML-Definition der Einstellungen dieser Anwendung. Jedes Element verfügt dabei über einen Titel und eine kurze Zusammenfassung,

welche in der App direkt unter dem Titel angezeigt wird. In dieser Definition werden zwei Einstellungspunkte erstellt. Der Erste wird in der XML direkt mit Logik hinterlegt. Es handelt sich dabei um den Einstellungspunkt der Paarungs-Activity. Da hier keine variablen Informationen hinzugefügt werden müssen, kann einfach innerhalb der XML-Datei ein Intent hinterlegt werden. Dem Intent wird dabei eine Aktion, hier der Aufruf einer Activity, und das Ziel übergeben. Das Ziel ist über das Paket und die Klasse genau definiert. Die Definition dieser Klasse wird innerhalb des Manifests vorgenommen. So wird direkt beim Aufruf dieses Einstellungspunktes der Intent verschickt und die Paarung geladen.
Der zweite Eintrag in der XML-Datei verfügt nicht über eine dahinterliegende Logik, sondern wird mit einem Schlüssel versehen. Innerhalb der `SettingsActivity` kann der Methode `findPreference()` dieser Schlüssel übergeben und ein Click-Listener an den entsprechenden Eintrag gesetzt werden. Die hinterlegte Logik soll in diesem Fall nicht noch genauer betrachtet werden, da der Aufbau identisch der einer Schaltfläche oder eines Menüeintrags ist.

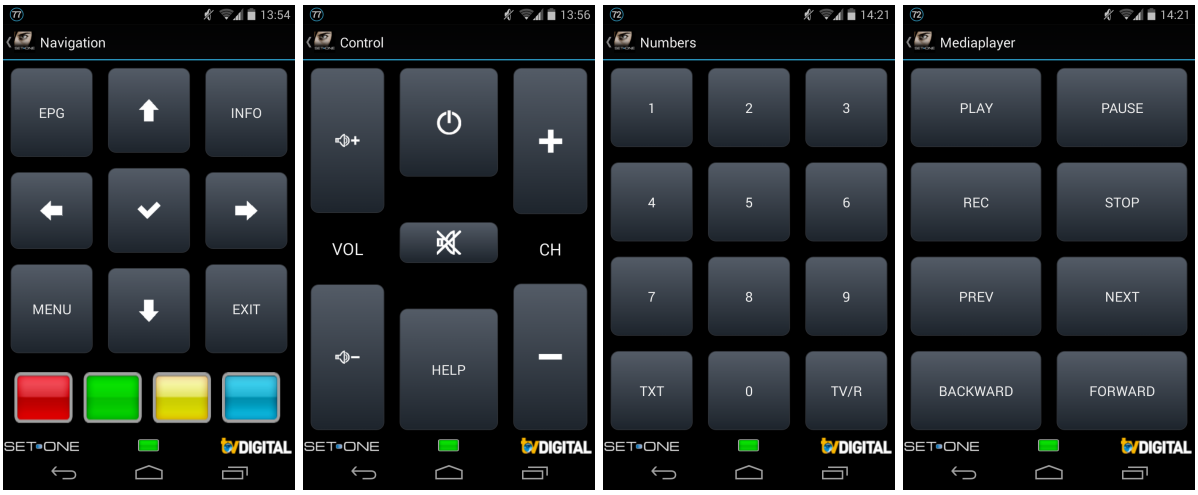
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
3      <PreferenceScreen
4          android:summary="@string/pairing_button_pair"
5          android:title="@string/pairing_button_pairing" >
6          <intent
7              android:action="android.intent.action.VIEW"
8              android:targetClass="de.discvision.setone.PairingActivity"
9              android:targetPackage="de.discvision.setone" />
10     </PreferenceScreen>
11     <PreferenceScreen
12         android:key="preference_info"
13         android:summary="@string/action_settings_info_summary"
14         android:title="@string/action_settings_info_title" >
15     </PreferenceScreen>
16 </PreferenceScreen>
```

Listing 6.8: Aufbau einer Einstellungen-Datei in XML

6.4.10 Fernbedienung

Die letzte große Bildschirmansicht ist die der Fernbedienung. Diese ist über einen Menüeintrag erreichbar und beinhaltet wie auch die Hauptbildschirmansicht einen `FragmentPager`, welcher es ermöglicht zwischen den einzelnen Fragmenten der vier Fernbedienungsbildschirme durch wischen zu wechseln. In diesen Fall wurde auf die Ansteuerung durch Tabs verzichtet, stattdessen wird der Name der aktuellen Ansicht innerhalb der Action Bar eingeblendet. Abbildung 6.12 zeigt eine Übersicht der verschiedenen Ansichten.

Die Fernbedienung ist nicht wie im Prototyp zu sehen über einen weiteren Tab im Hauptbildschirm zu erreichen, sondern aus mehreren Gründen über einen Menüeintrag. Zum einen wurde darauf verzichtet, weil eine Trennung der Organisation der Set-Top-Box und die einfache Bedienung durch die Fernbedienung durchaus sinnvoll ist. Für den Nutzer sind dies zwei verschiedene Anwendungsgebiete und durch die Trennung erlangt dieser eine bessere Übersicht in den Teilbereichen. Zumeist wird der Nutzer diese beiden Anwendungsbereiche nicht simultan nutzen, sodass



(a) Navigation **(b)** Grundbedienung **(c)** Nummernfeld **(d)** Mediensteuerung

Abbildung 6.12: Ansichten der Fernbedienung

eine Trennung Sinn macht. Außerdem ist eine Navigation innerhalb der Fernbedienungsansichten nun durch wischen möglich. Wäre die Fernbedienung bereits im Hauptbildschirm durch Wischen erreichbar, hätte die Navigation innerhalb der Fernbedienung anders organisiert werden müssen, da die Wisch-Geste bereits vergeben wäre.

Die Anwendung kann die Hardware Fernbedienung vollkommen ersetzen, da alle Tasten der mitgelieferten Fernbedienung auch in der Applikation verfügbar sind. Die Tasten sind in vier Bereiche aufgeteilt. Dabei wird nicht nur eine erhöhte Übersicht gewährleistet, sondern es wurde auch versucht die Fernbedienungselemente sinnvoll zu strukturieren. Es wurde eine Ansicht für die Bedienung des Menüs und die oft verwendeten Funktionen (a), eine für die Grundfunktionen (b) wie umschalten und regeln der Lautstärke, eine für die direkte Eingabe der Zahlen (c) und eine nur für die Bedienung des integrierten Media-Players (d) erstellt.

Die Fernbedienung besteht aus einer eigenen Activity. Diese besitzt wie auch schon der Hauptbildschirm einen `FragmentPager`, welcher die vier Ansichten der Fernbedienung nebeneinander verwaltet. Die Titel der einzelnen Ansichten werden in der Action Bar angezeigt. Zur Bestimmung der aktuellen Position wird auch hier ein Listener an den `FragmentPagerAdapter` gehangen und so auf die Seitenwechsel reagiert.
Innerhalb der verschiedenen Layouts wird jeder Schaltfläche eine ID zugewiesen. Wird eine Schaltfläche gedrückt, wird immer die Methode `buttonPress()` innerhalb der Activity aufgerufen. Dieser Methode wird auch die ID der Schaltfläche mitgegeben. Diese ID wird wiederum in einer `switch-case`-Abfrage auf den entsprechenden Tastenbefehl abgebildet und über den Server an die Box gesendet. Jeder Tastendruck wird also einzeln versendet. Somit erfährt der Nutzer sofort Feedback seines Tastendrucks auf seiner Set-Top-Box. Zudem bietet sich dem Nutzer die Möglichkeit ein weiteres Feedback über eine kurze Vibration bei Berührung der Schaltflächen zu erhalten. Diese Option kann in den Einstellungen der Anwendung aktiviert werden und wird

persistent gespeichert.

6.5 Model und Datenhaltung

In diesem Abschnitt soll die allgemeine Datenhaltung innerhalb der Applikation beleuchtet werden, wie das Model der Anwendung aufgebaut ist und welche Funktionen es übernimmt. Im Allgemeinen werden die Daten welche vom Server geladen werden innerhalb des Models gelagert. Das Model dieser Applikation ist global verfügbar und es wird mittels des Singleton Entwurfsmusters sichergestellt, dass immer nur eine Instanz dieser Klasse besteht. Die Variablen innerhalb des Models werden alle als `private` deklariert und können nur über die Model-Instanz via Getter- und Setter-Methoden abgerufen oder gesetzt werden.

Das Model beinhaltet zum einen feste Daten wie die Serveradresse des ConnectTV-Servers oder dessen Port. Zum Hauptteil stellt es allerdings Variablen, welche zur Laufzeit mit Daten gefüllt werden. Darunter fallen vor allem die Daten des EPGs, des Planers und der Aufnahmen, welche in verschachtelten Listen und selbsterstellten Klassen organisiert werden oder Informationen zum Status der Set-Top-Box. Es werden aber auch einfache Daten wie der Nutzername oder die aktuelle Session-ID zwischengelagert.
Die Daten werden in der Regel so lange im Speicher gehalten bis die Applikation vollständig beendet wird. Das heißt, dass sie durch den User manuell beendet wurde oder aufgrund von Arbeitsspeichermangel durch das System beendet wurde. Außerdem besitzt jeder Datensatz einen Zeitstempel des Ladezeitpunkts. Bleibt die Anwendung über einen gewissen Zeitraum aktiv, sodass zum Beispiel die Daten des EPGs veraltet sind, werden diese bei Fortsetzung der Anwendung neu geladen.

Es handelt sich in dieser Arbeit nicht um ein komplett reines Model, welches nur die Datenhaltung übernimmt, sondern es stellt auch einige Methoden zum Abruf von Daten, welche nach MVC-Richtlinien im Normalfall durch einen Controller gestellt werden würden. So werden unter anderem über das Model die HTTP-Anfragen an den ConnectTV-Server abgesetzt, die Antworten angenommen und über einen Parser weiterverarbeitet. Außerdem sind dort auch Methoden zu finden, welche in allen Bereichen verfügbar sein müssen. Darunter fallen Methoden zur Netzwerkerreichbarkeit oder Set-Top-Box-Konnektivität. Diese wurden im Model implementiert, da sie überall abrufbar sein sollen und somit nicht in jeder Klasse neu erstellt werden müssen, ganz nach dem Software Engineering Prinzip „Don't repeat yourself".

Einige Daten sollen in der Anwendung persistent gehalten werden und immer verfügbar sein, auch wenn die Anwendung einmal vollständig beendet wurde. Darunter fallen beispielsweise der Nutzername und das Passwort oder die Einstellungen. Um diese permanent zu speichern werden `SharedPreferences` verwendet. Diese werden durch das Betriebssystem auf dem Gerät abgelegt und können jederzeit in der Anwendung abgerufen werden. Android stellt hier außerdem die benötigte Sicherheit, sodass auf diese Daten außerhalb der Anwendung kein Zugriff möglich ist. Bei Deinstallation der Anwendung werden die gespeicherten Daten ebenfalls durch Android entfernt.

Das Android Betriebssystem stellt über die Ressourcen eine einfache Möglichkeit mehrere Sprachdateien einzubinden. Um dies nutzen zu können, müssen die in Abschnitt 3.3.5 vorgestellten String Dateien verwendet werden. Die Standardsprachdatei wird dabei im Ordner „values“ abgelegt. In dieser Arbeit wurde die Standardsprache mit Englisch gesetzt, da dies die meist verbreitetste Sprache ist. Um aber auch eine deutschsprachige Version anbieten zu können, wurde eine weitere String Datei des gleichen Namens mit den deutschen Übersetzungen erstellt. Diese Datei wurde im neu erstellten Ordner „values-de“ abgelegt. Ist die Betriebssystemsprache nun auf deutsch eingestellt, wird in der Applikation automatisch der Ordner mit der entsprechenden Kennung ausgewählt. Die Kennung der Ordner richtet sich dabei nach den ISO-Normen der Ländercodes. Ist die gewünschte Sprache nicht verfügbar, wird automatisch die Standardsprache ausgewählt.

6.6 Manifest

Das Manifest beinhaltet den Aufbau der Applikation, sowie dessen Umgebungsparameter. Listing 6.9 zeigt einen Auszug aus der in dieser Arbeit erstellten Manifest Datei.
Zunächst werden dort einige Randbedingungen gesetzt. Begonnen wird mit dem Java Package der Anwendung, welches auch als eindeutiger Identifikator der Anwendung dient. Dieses Paket dient zum Finden der Anwendung innerhalb des Betriebssystems. Dies wurde auch schon beim Start des Videoplayers in Abschnitt 6.4.6 als Indikator verwendet. Darauf folgt in den Zeilen 4-5 die Versionsnummer der Anwendung. Wenn die Applikation in den Google Play Store gestellt wird, muss diese bei einem Update immer erhöht werden. Darauf folgt die minimal benötigte Android API Version (Zeile 8) mit der diese Anwendung betrieben werden kann, sowie gegen welche Version die Applikation kompiliert wurde.

Darauf folgen die von der Applikation benötigten Berechtigungen in Form von XML-Elementen des Titels `uses-permission` in den Zeilen 11-15. Der `ACCESS_NETWORK_STATE` ist die erste Berechtigung, dieser erlaubt der Applikation Netzwerkverbindungen abzurufen. Darauf folgt die sehr ähnliche `ACCESS_WIFI_STATE`-Berechtigung, welche den Abruf von WLAN-Verbindungen ermöglicht. Diese werden für die Überprüfung einer aktiven Netzwerkverbindung des Geräts benötigt. Die dritte Berechtigung `CHANGE_WIFI_MULTICAST_STATE` erlaubt der Anwendung WLAN-Multicasts zu empfangen, welche für das Sat>IP und UPnP Protokoll benötigt werden. Über die `INTERNET`-Berechtigung darf die Anwendung auf das Internet zugreifen, welches für die Kommunikation mit dem ConnectTV-Server benötigt wird. Zu Letzt wird die Berechtigung `VIBRATE` angefordert, mit welcher die Vibrationsfunktion des Geräts aktiviert werden kann. Diese wird für ein Feedback bei Tastendruck der Fernbedienung verwendet.

Im nächstem Teil folgt der eigentliche Aufbau der Applikation. Zunächst werden der Anwendung in den Zeilen 19-20 ein Name und ein Icon zugeordnet, welche aus den entsprechenden Ressourcen Ordnern geladen werden. In Zeile 18 wird zusätzlich die Funktion `allowBackup` deaktiviert. Diese Funktion ermöglicht es ein Backup der Daten der Anwendung zu erstellen. Dies ist in diesem Fall jedoch ein Sicherheitsrisiko, da die gespeicherten Daten dann über ein USB-Kabel ausgelesen werden können. Da in diesem Fall auch Benutzername und Passwort permanent ge-

speichert werden, ist diese Funktion bewusst deaktiviert.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.discvision.setone"
    android:versionCode="9"
    android:versionName="0.9" >

    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE"/>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.VIBRATE" />

    <application
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name="de.discvision.setone.LoginActivity"
            android:label="@string/app_name"
            android:theme="@style/Theme.Sherlock.NoActionBar"
            android:windowSoftInputMode="stateHidden" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="de.discvision.setone.MainActivity"
            android:label="@string/app_name"
            android:theme="@style/Theme.Sherlock" >
        </activity>
        <activity
            android:name="de.discvision.setone.remote.RemoteActivity"
            android:label="@string/title_activity_remote"
            android:theme="@style/Theme.Sherlock" >
        </activity>
        .
        .
        .
    </application>
</manifest>
```

Listing 6.9: Auszug aus der Manifest Datei

Zum Schluss und den Großteil des Manifests nimmt die Deklarierung der Activities der Anwendung ein. Jede Activity wird dabei mit einem eindeutigem Namen versehen, über welchen sie innerhalb der Anwendung angesprochen werden kann. Da die Sherlock Action Bar verwendet werden soll, muss jede Activity über das Attribut `theme` mit dem entsprechendem Styling versehen werden. Die Anmelde-Activity erhält außerdem das Attribut `windowSoftInputMode` mit dem Status `stateHidden`. Da das oberste Element der Activity ein Textfeld ist, bekommt dieses zunächst den Fokus. Dies würde direkt das Tastenfeld aufrufen, das wird jedoch durch dieses

Attribut verhindert. Die Anmelde-Activity verfügt außerdem in den Zeilen 26-29 über einen `intent-filter`. Dieser wird benötigt, da diese Activity als Startpunkt der Anwendung dient und dementsprechend als `LAUNCHER` kategorisiert werden muss.

6.7 Universal Plug and Play

Ein reines Universal Plug and Play kommt in dieser Arbeit nur in einem gewissen Maße zum Einsatz. Es sollte zum Abspielen von abgespeicherten Aufzeichnungen im Heimnetzwerk zum Einsatz kommen, aber für dieses Szenario wurde ein einfacher Workaround gefunden, welcher im Folgenden erklärt wird. Nichtsdestotrotz kommt das UPnP Protokoll dennoch bei der Verwendung von Sat>IP zum Einsatz.
Befindet sich das Android Gerät im gleichem Netzwerk wie die Set-Top-Box wird das hier folgende Verfahren verwendet. Befindet sich die Box nicht im gleichem Netzwerk wird das HTTP Live Streaming verwendet, welches im Abschnitt 6.8 beschrieben wird.

Der Nutzer kann im Aufnahmebereich über eine Abspiel-Taste eine abgespeicherte Aufzeichnung auf dem Android Gerät abspielen. Dafür wird zunächst überprüft ob der MX Player installiert ist. Ist dies der Fall wird ein `AsyncTask` gestartet, welcher den Stream an den Player übergeben soll und dieser somit das Abspielen der Datei starten kann.

Das UPnP Protokoll sieht vor, dass der Client über einen Medienbrowser die gewünschte Datei auswählt. Diese wird dadurch vom Client angefordert und der Server stellt eine URL bereit unter welcher ein Stream der Datei anzufordern ist. In diesem Fall ist die gewünschte Datei allerdings schon bekannt, da das Video bereits vom Nutzer in der Aufnahmeoberfläche ausgewählt wurde. Somit muss dem Server lediglich übermittelt werden welches Video gestreamt werden soll. Der Client muss also den Namen der Datei kennen um den Stream anfordern zu können.

Durch analysieren der Namen der gespeicherten Aufzeichnungen ist schnell klar geworden, dass diese sich immer nach dem gleichem Schema zusammensetzen. Der Name ist ein Konstrukt aus der Startzeit der Sendung, der Programm-ID und der Sender-ID. Diese Informationen können alle aus den im Model gespeicherten Daten der Aufnahmeliste entnommen werden. Somit wird nur noch die IP-Adresse der Set-Top-Box, der Standard Übertragungsport sowie der Name des Datei Systems der Set-Top-Box benötigt. Diese Informationen sind ebenfalls über den ConnectTV-Server bereits im Model vorhanden.
Also kann die URL der anzufordernden Datei aus den vorhanden Daten zusammengestellt werden und an den MX Player übergeben werden. Die Aufrechterhaltung der Verbindung wird dabei durch den Player automatisch geregelt. Der Start des MX Players wurde bereits in Abschnitt 6.4.6 genauer beschrieben. Die Übertragung des Streams an das Android Gerät wird hier direkt über den in der Set-Top-Box integrierten UPnP Medienserver verwaltet. Durch das Drücken der Zurück-Taste innerhalb des MX Players kann wieder zur Applikation zurückgekehrt werden und am gleichem Punkt der Anwendung wieder eingestiegen werden, an dem diese verlassen wurde um das Video zu starten.

6.8 HTTP Live Streaming

Das HTTP Live Streaming wird für das entfernte Abspielen von gespeicherten Aufzeichnungen verwendet. Ist die Set-Top-Box also nicht im gleichem Netzwerk wird bei Betätigung der Abspiel-Taste im Aufnahmebereich diese Methode verwendet.
Auch hier wird zunächst überprüft ob der MX Player installiert ist. Ist dies der Fall wird analog zum UPnP ein `AsyncTask` gestartet, welcher das Streaming auf dem Player anstoßen soll.

Zunächst werden Informationen vom ConnectTV-Server eingeholt, ob ein Streamen der Aufzeichnung generell möglich ist. Dies ist nicht immer der Fall, da die Aufzeichnung zunächst mit Hilfe eines Encoders in andere Formate transkodiert werden muss. Möglicherweise ist dieser Vorgang zum Abspielzeitpunkt noch nicht abgeschlossen und daher eine Wiedergabe noch nicht ausführbar. Ist dieser abgeschlossen, werden eine ID, ein Zeitstempel und weitere Informationen zur zum Datenformat im XML-Format zurückgeliefert.

Mit Hilfe dieser ID und des Zeitstempels, sowie der MAC-Adresse der Set-Top-Box kann über den Server nun die Bereitstellung eines HTTP Live Streams angefordert werden. Die Box liefert daraufhin eine URL unter welcher der Stream abrufbar ist, sowie Informationen zur aufgezeichneten Sendung. Diese URL wird wiederum über einen Intent an den MX Player übergeben. Die Wiedergabe und die Aufrechterhaltung der Verbindung wird auch hier automatisch vom MX Player geregelt. Der Stream selbst wird über eine direkte Verbindung des Android Geräts mit der Set-Top-Box geladen. Das Verteilen der Stream-Segmente wird dabei über den in der Set-Top-Box integrierten Webserver übernommen.
Auch hier gilt, durch das Drücken der Zurück-Taste innerhalb des MX Players kann wieder zur Applikation zurückgekehrt werden und am gleichem Punkt der Anwendung wieder eingestiegen werden, an dem diese verlassen wurde.

Abschließend ist zum HLS jedoch zu sagen, dass auf Seiten des ConnectTV-Servers hier noch ein Problem vorliegt, welches bis zum Abschluss der Arbeit nicht von DiscVision gelöst werden konnte. Der Server zeigt an, dass die entsprechenden Aufzeichnungen vorhanden sind. Allerdings wird auf Anfrage, mittels der entsprechenden ID kein Stream bereitgestellt. Diese Fehlfunktion ist auch in einem Browser über die entsprechenden HTTP Anfragen reproduzierbar. Die Funktion selbst ist innerhalb der Applikation aktiviert und getestet. Sie sollte nach der Beseitigung des Problems seitens des ConnectTV-Servers ohne weitere Änderungen in der Anwendung lauffähig sein.

6.9 Sat>IP

Das Sat>IP Protokoll wird verwendet um im Heimnetzwerk das Abspielen des aktuellen Fernsehprogramms auf dem Android Gerät zu ermöglichen. Dem Nutzer wird für diesen Zweck innerhalb der EPG Ansichten eine Abspiel-Taste an den verschiedenen Sendern und am aktuellem Programm angezeigt. Über diese Schaltfläche kann unter Verwendung des MX Players die Wiedergabe des Programms gestartet werden.

Zunächst wird über das dem Sat>IP zugrundeliegende UPnP Protokoll im Netzwerk nach einem Sat>IP Server gesucht. Hierfür wird ein M-SEARCH Multicast im Netzwerk verschickt. Dabei wird als „Search Target“-Parameter `urn:ses-com:device:SatIPServer:1` beigefügt, sodass nur Sat>IP Server auf diesen Mulitcast antworten. Daraufhin wird ein `TimerTask` gestartet, welcher die entsprechenden Antworten empfängt, diese parst und die Geräte innerhalb einer `ArrayList` abspeichert. Die Geräte liefern jeweils eine URL zurück unter welcher genaue Infos zu diesen angefordert werden können. Diese werden daraufhin für jedes dieser Geräte bezogen und die Antworten mittels XML-Parser über eigens erstellte Sat>IP-Client-Device-Klassen organisiert. Der gewünschte Sat>IP-Server der Set-Top-Box wird dann anhand der zurückgegeben IP-Adresse und der vom ConnectTV-Server bereits bekannten IP-Adresse abgeglichen und im Model zwischengespeichert. Diese Prozedur dient der Überprüfung ob der in der Set-Top-Box integrierte Sat>IP Server aktiviert ist. Dieser kann in den Optionen der Set-Top-Box abgeschaltet werden und wäre dadurch nicht erreichbar.

Mit Hilfe der Bestätigung durch das eben durchgeführte Discovery kann nun die eigentlich Herstellung der Verbindung zum Sat>IP Server durchgeführt werden und der Stream angefordert werden. Hierfür wird zunächst mittels einer SETUP-Anfrage eine Session des gewünschten Programms erstellt. Die für diese Anfrage benötigten Daten, wie Frequenz, Polarisation oder Symbolrate, werden durch den ConnectTV-Server bereitgestellt. Bei jeder Anfrage an den Sat>IP-Server muss eine fortlaufende Sequenznummer unter dem Parameter `CSeq` übergeben werden. Der Server gibt im Erfolgsfall eine Session-ID und eine Stream-ID zurück.

Mit diesen IDs kann nun die PLAY-Anfrage abgeschickt werden. Wird dieser ebenfalls vom Server bestätigt, kann der MX Player nach bereits bekanntem Verfahren unter Angabe der URL „rtsp://"+ ip + ":"+ port + "/stream="+ streamID“ gestartet werden. In diesem Fall wird die Aufrechterhaltung der Verbindung nicht automatisch durch den Player geregelt. Aus diesem Grund wird ein Hintergrund Task gestartet, welcher alle fünf Sekunden eine OPTION-Anfrage an den Sat>IP-Server sendet um die Verbindung zu halten. Ansonsten würde je nach Einstellung am Server die Verbindung automatisch nach einem gewissen Zeitraum abgebrochen werden.

Der Hintergrund Task läuft solange innerhalb des Models die Boolean-Variable `satipAlive` den Wert `true` hat. Diese muss also bei Rückkehr vom Player zur Applikation wieder auf `false` gesetzt werden, damit dieser beendet wird. Um dies zu erreichen wird der MX Player mit einem `startActivityForResult()` Intent gestartet. Wird nun durch Beendigung des Players zur Applikation zurückgekehrt, kann innerhalb der Methode `onActivityResult()` die `stayAlive` Variable entsprechend gesetzt werden, sodass der Task eine TEARDOWN-Anfrage an den Server sendet, um die Übertragung zu beenden, und sich dann selbst beendet.

Die Wiedergabe eines jeden Programms muss also aus der Applikation gestartet werden. Da ein externes Abspielprogramm verwendet wird, kann das Programm nicht innerhalb des Players gewechselt werden, da hier keine Befehle mehr abgefangen werden können und entsprechend darauf reagiert werden kann.

Abschließend ist zur Sat>IP Funktion noch zu sagen, dass diese selbst funktionsfähig ist, allerdings noch ein Problem aufweist. Die Daten zur Erstellung eines Streams, wie die Frequenz oder die Polarisation des Senders, sollten durch den ConnectTV-Server von der Set-Top-Box geladen

werden. Die DiscVision GmbH hat es allerdings nicht geschafft bis zum Abschluss dieser Arbeit einen solchen Service bereitzustellen. Zudem wurde kein verlässlicher und kostenloser externer Service gefunden. Somit wird die Funktion momentan nur mit fixen Werten für einen bestimmten Sender gestartet. Sie ist also funktionsfähig, müsste allerdings noch bei Fertigstellung des ConnectTV-Services angepasst werden.

Kapitel 7

Evaluation

Neben der Entwicklung der Android Applikation soll ebenfalls eine Evaluation dieser durchgeführt werden um die Android Design Richtlinien, die Anpassung an verschiedene Bildschirmgrößen, sowie die Funktionen der Applikation selbst zu hinterfragen. Zu diesem Zweck wurde mit einer überschaubaren Gruppe von 13 Probanden ein Usability Test durchgeführt. Die sowohl männlichen als auch weiblichen Probanden sind im Alter von 22-31 Jahren und besitzen dabei teils sehr unterschiedliche Vorkenntnisse. Im Folgenden werden der Ablauf des Usability Tests und die Bewertungsmethodik vorgestellt sowie die Ergebnisse präsentiert.

7.1 Ablauf

Der Ablauf des Usability Tests wird immer gleich durchgeführt. Um die Integrität des Tests zu gewährleisten, wurde ein exakter Ablaufplan erstellt, welcher bei den Tests mit jedem Probanden Schritt für Schritt abgehandelt wurde. Der Ablaufplan im Anhang zu finden.

Vor Beginn des Tests wurden einige Vorbereitungen getroffen, sodass die Ausgangssituation für die Probanden immer die gleiche ist. Bei Eintreffen der Probanden wurden diese begrüßt und es wurde ihnen genauer erklärt worum es bei diesem Softwaretest geht. Das Masterarbeitsthema und die Geräte sowie ihre Funktionen wurden kurz vorgestellt. Diesem folgte eine kurze Erläuterung zum Ablauf des Tests und der Nachfrage nach weiteren Fragen ihrerseits. Dann wurden einige anonyme, personenbezogene Daten zu Alter, Geschlecht und Vorerfahrungen erfasst und der Test gestartet.

Der Test selbst ist dabei in zwei Teile gegliedert. Im ersten Teil des Tests werden dem Probanden sechs Szenarien vorgestellt, welche er schrittweise mittels eines Android Geräts abarbeiten muss. Als Testgerät wurde ein Nexus 4 der Marke LG mit der Android Version 4.4.2 verwendet. Nach jedem Szenario muss der Proband einen Fragebogen zur Schwierigkeit der Aufgabe ausfüllen. Als Grundlage für den Fragebogen diente dabei der NASA Task Load Index, welcher in Abschnitt 7.3 erläutert wird.
Im zweiten Teil des Tests werden dem Probanden weiterführende Fragen zur Applikation gestellt.

Dabei handelt es sich beispielsweise um Befragungen zum Layout oder zur Handhabung der Anwendung.

Zum Ende des Usability Tests wurde der Proband in einer offenen Frage um Kritik und Anregungen gebeten. Nach dem Test wurde der Fragebogen kurz geprüft, der Proband wurde durch ein kleines Präsent entlohnt und verabschiedet.

7.2 Personenbezogene Daten

Vor dem eigentlichem Softwaretest wurden einige personenbezogene Daten aufgenommen, da diese bei der Bewertung durchaus ins Gewicht fallen können oder auch Ausreißer außerhalb der erwarteten Messreihen erklären können. Diese Daten sind:

- Alter
- Geschlecht
- Vorerfahrung mit Smartphones
- Vorerfahrung mit dem Android Betriebssystem
- Vorerfahrung mit Set-Top-Boxen

7.3 NASA-TLX

Der NASA Task Load Index ist im ersten Teil des Usability Tests zum Einsatz gekommen und dient der Erfassung wahrgenommener Arbeitsbelastung bei der Bearbeitung von Aufgaben, die mit sechs verschiedenen Skalen erhoben wird. Entwickelt wurde der NASA-TLX von Hart & Staveland [HS88]. Grundlage für die Implementierung in dieser Untersuchung war eine deutsche Übersetzung von Unema et al. [URSW+88]. Es handelt sich bei dieser Bewertung um geschlossene Fragen, das heißt vorgegebenen Antwortmöglichkeiten.
Die sechs Skalen lauten:

- Geistige Anforderungen
- Körperliche Anforderungen
- Zeitliche Anforderungen
- Ausführung der Aufgaben
- Anstrengung
- Frustration

Die Skalen bestehen aus je einem Item, das im Original mit einer 21-stufigen Skala mit den Verankerungen „gering“ bis „hoch“ erhoben wird. Diese wurde - im Sinne der Skalier- bzw. Vergleichbarkeit - in diesem Softwaretest in eine 5-stufige Skala abgewandelt.

7.4 Usability Test

Im ersten Teil des Tests wurden dem Probanden sechs verschiedene Szenarien vorgestellt, welche er nach und nach durchführen und in ihrer Schwierigkeit bewerten sollte. Diese Szenarien befassen sich mit der Handhabung der Applikation und sollen Aufschluss darüber geben, ob diese übersichtlich und verständlich gestaltet ist und intuitiv zu bedienen ist. Die sechs Szenarien lauten:

1. **Anmelden:** Bei Erhalt des Smartphones ist die Applikation bereits gestartet. Bitte melden Sie sich mit folgenden Daten im System an. Name: Sebastian, Passwort: qqqqqq
2. **Planen einer Aufnahme:** Versuchen Sie die Aufzeichnung einer Sendung zu planen, welche übermorgen gegen 19 Uhr auf dem Sender „RTL2“ ausgestrahlt wird.
3. **Ansehen einer Aufnahme:** Sehen Sie sich eine bereits vorhandene Aufzeichnung einer Sendung an. Kehren sie nach einigen Sekunden wieder zur Applikation zurück.
4. **Löschen einer geplanten Aufnahme:** Löschen Sie die von Ihnen in Szenario 2 geplante Aufzeichnung wieder. Wählen sie danach wieder das heutige Datum aus.
5. **Fernbedienung:** Wählen Sie die Fernbedienung aus. Schalten Sie auf Kanal 7. Und schalten Sie dann den Ton ab.
6. **Abspielen des aktuellen Programms:** Starten Sie die Wiedergabe des Senders „ProSieben“ auf dem Smartphone. Kehren sie nach einigen Sekunden wieder zur Applikation zurück.

Im zweiten Teil des Usability Tests werden weiterführende Fragen zur Applikation gestellt. Diese Beziehen sich auf das Layout, die Handhabung, den von Nutzern zu erwartenden Einsatzbereichen sowie der Bewertung der Applikation im Allgemeinen. Die Antworten der Fragen konnten in einer Skala von 1-5 abgegeben werden. Einige Fragen konnten nur mit vorgegebenen Single-Choice Antworten bewertet werden. Dieser Teil besteht ebenfalls nur aus geschlossenen Fragen.

7.5 Auswertungsmethoden

Die Auswertung der erhobenen Daten orientiert sich an den in der Norm DIN EN ISO 9241-110 [Deu06] genannten Kriterien für Usability: Effektivität, Effizienz und Nutzerzufriedenheit. Zur Erfassung dieser drei Konstrukte wurden quantitative Methoden verwendet.

7.5.1 Effektivität

Der Grad der Zielerreichung wurde pro Teilnehmer und Nutzungsszenario einer der folgenden vier Abstufungen zugeordnet:

- **1 - nicht gelöst:** Die Aufgabe wurde nicht erfolgreich gelöst.

- **2 - teilweise gelöst:** Teilaspekte der Aufgabe konnten nicht gelöst werden.
- **3 - gelöst mit Hilfe:** Die Aufgabe konnte nur mit Hilfestellung des Versuchsleiters gelöst werden.
- **4 - vollständig gelöst:** Die Aufgabe wurde vollständig und ohne Hilfestellung gelöst.

Als zufriedenstellend wurden nur Ergebnisse betrachtet, die vollständig gelöst wurden. Bei Szenarios, die nicht oder nur mit Hilfe gelöst wurden, wird ein Problem vermutet und im Folgenden berichtet.

7.5.2 Effizienz

Das zweite verwendete Konstrukt ist der bis zur Erreichung der Zielstellung empfundene kognitive, körperliche sowie zeitliche Aufwand der Probanden. Diese wurde mit dem oben beschriebenen Fragebogen NASA-TLX erhoben. Die Skalen, die hierfür eingesetzt wurden sind „Geistige Anforderungen“, „Zeitliche Anforderungen“, „Ausführung der Aufgaben“ und „Anstrengung“. Für die Festlegung der Kriterien akzeptabler und nicht akzeptabler Effizienz lagen keine passenden Referenzwerte vor, was die Unterteilung der intervallskalierten Skala in Quartile nahelegt. Dabei wurden die ersten beiden Quartile als gut bzw. akzeptabel, die zweiten beiden Quartile als nicht akzeptabel eingestuft.

7.5.3 Nutzerzufriedenheit

Als Instrument zur Erhebung der Nutzerzufriedenheit diente die Skala „Frustration“ des NASA-TLX. Frustration lässt sich als Beeinträchtigung des Befindens verstehen. Zufriedenheit resultiert unter anderem aus der Abwesenheit von Beeinträchtigungen. Es wurden aus den bereits genannten Gründen wieder auf die Quartile als Maßstab für akzeptable und nicht akzeptable Nutzerzufriedenheit herangezogen.

7.6 Befunde und Empfehlungen

7.6.1 Einführung

Im Folgenden werden die aus den erhobenen Daten mittels der Auswertungsmethoden abgeleiteten Befunde vorgestellt und erklärt. Dabei werden zunächst die Szenarien des ersten Teils des Tests ausgewertet. Nach Ermittlung der Probleme soll versucht werden Empfehlungen zur Verbesserung der Applikation vorzustellen. Im Anschluss daran werden die weiterführenden Fragen das zweiten Teils betrachtet und beurteilt. Zudem werden einige durch die Evaluation gewonnene abschließende Eindrücke dargestellt. Zum Ende des Kapitels sollen nach Feststellung und Beurteilung der Befunde noch einige darauf aufbauende Anpassungen der Applikation vorgestellt werden.

Der Fragebogen sowie dessen Auswertung ist im Anhang hinterlegt. Außerdem sind dort die Anmerkungen zur Applikation seitens der Probanden aufgelistet. Ferner sind die vorgefallenen Probleme und die Verbesserungsvorschläge der Probanden, nach Häufigkeit des Auftretens aufgelistet, dort zu finden. Die von den Probanden bearbeiteten Fragebögen finden sich auf der beiliegenden CD wieder.

7.6.2 Befunde der Testszenarien

Szenarien 1: Anmelden, 3: Ansehen einer Aufnahme, 4: Löschen einer Aufnahme, 6: Abspielen des aktuellen Programms:
Diese Szenarien wurden von allen Probanden immer vollständig gelöst, weshalb die Bearbeitungs-Effektivität bei diesen sehr hoch ist. Außerdem liegt die gemessene Effizienz immer im Mittelwert unter 1,5, sodass die Anwendung in diesen Bereichen keine hohen Anforderungen an den Probanden zu stellen scheint. Dies wird auch in der Nutzerzufriedenheit widergespiegelt, welche im schlechtesten Fall mit einem Mittelwert von 1,31 bewertet wurde. Bei der Nutzerzufriedenheit gibt es nur im ersten Szenario drei Ausreißer. Die abschließenden Befragungen dieser Probanden hat ergeben, dass das Anmelden nur mittelmäßig bewertet wurde, weil bei Auswahl der Textfelder des Nutzernamens sowie des Passworts kein Cursor zu sehen ist. Somit ist nicht sofort erkenntlich ob das gewünschte Textfeld ausgewählt ist. Das Fehlen des Cursors ist allerdings nicht auf die Applikation zurückzuführen, denn dieser Fehler ist nicht auf jedem Android Gerät präsent. Das Testgerät ist mit der Android Version 4.4.2 ausgestattet. Seit Android 4.0.3 ist dies ein bekannter Fehler und sollte somit in einer künftigen Android Version behoben werden. [Gooa]
Die Szenarien 1, 3, 4 und 6 waren also von den Probanden schnell und einfach zu lösen und weisen somit keine gravierenden Probleme auf. Schwierigkeiten gab es bei den Szenarien 2 und 5, welche im Folgenden noch genauer betrachtet werden sollen.

Szenario 2: Planen einer Aufnahme:
Beim Planen einer Aufnahme gab es bei einigen Probanden Probleme. Auch wenn die Lösungs-Effektivität hier noch recht gut ausfällt, sind innerhalb der Effizienz und der Nutzerzufriedenheit relativ eindeutige Komplikationen aufgetreten.
Alle bis auf zwei Probanden haben das Szenario eigenständig vollständig gelöst. In zwei Fällen ist nach längerer Ratlosigkeit der Probanden der Versuchsleiter eingeschritten und hat den Probanden eine kleine Hilfestellung gegeben. In dieser Hilfestellung handelte es sich allerdings nur um Gedankenanstöße das Szenario auf einen anderen Weg anzugehen.
Die Effizienz spiegelt eindeutig wieder, dass es für einige Probanden merklich anstrengender war das gesteckte Ziel zu erreichen. Die Maximalwerte liegen dabei im Bereich von 3-5. Diese Werte sind nicht mehr als akzeptables Ergebnis anzusehen. Außerdem bestätigen die Mittelwerte in diesem Bereich, dass die meisten Probanden bei diesem Szenario vor Problemen standen.
Die Nutzerzufriedenheit liegt hier bei einem Mittel von 2,0, weist aber auch Ausreißer auf. Die Nutzerzufriedenheit ist wahrscheinlich nur aus dem Grund noch recht gut ausgefallen, da die meisten Probanden das Szenario dennoch erfolgreich abschließen konnten.

Durch Beobachtung und Befragung der Probanden wurde klar, dass über 50% der Probanden

zunächst versuchten die Aufnahme im Bereich des Planers zu planen. Dabei sind auch erfahrene Nutzer(Probanden 1 und 7) auf dieses Problem gestoßen. Zudem wurde von drei Probanden angemerkt, dass das Kürzel EPG für sie vollkommen unbekannt war. Erst durch ausprobieren der Funktionen in diesem Bereich wurde ihnen klar was darunter zu verstehen war.

Als Empfehlung wurde von zwei Probanden vorgeschlagen die Planung einer Aufnahme über eine Schaltfläche auch im Planer zu ermöglichen. Außerdem solle das Kürzel EPG in ein Wort wie Programm oder Programmübersicht geändert werden, sodass einem Nutzer die Funktion dieses Bereichs sofort klar ist und er sich so schneller zurechtfinden kann. Für eine bessere Anfangsorientierung könnte ebenfalls ein kleines Tutorial der Funktionen und Bereiche bei Erststart der Applikation aufgerufen werden.

Szenario 5: Fernbedienung:
Bei der Verwendung der Fernbedienung sind die größten Probleme aufgetreten. Dies zeigt gerade die Auswertung der Effektivität, aber auch der Effizienz.
Weniger als 50% der Probanden konnten das Szenario eigenständig vollständig lösen. Weitere knapp 50% haben bei der Bearbeitung Hilfe benötigt und darüber hinaus konnte ein Proband die Ziele des Szenarios nur teilweise lösen. Dies sind bereits eindeutige Anzeichen für ein Usability Problem.
Auch die Werte der Effizienz liegen im Maximum bei 4-5. Diese sind nicht als Ausreißer zu sehen, denn die Mittelwerte dieser Skalen liegen auch in diesem Bereich nicht mehr im akzeptablen Bereich. Obgleich die Bewertung einiger Probanden diesen noch recht hoch wirken lässt. Dies wird auch durch eine Standardabweichung von über 1,1 widergespiegelt.
Die Nutzerzufriedenheit lag hier im Mittel noch im akzeptablen Bereich, doch die Maximalwerte von 4 und die Tatsache, dass nur vier Probanden sehr zufrieden waren, lässt ebenfalls auf Probleme in diesem Bereich schließen.

In diesem Szenario konnte das Problem schnell ausfindig gemacht werden. Über 60% der Probanden waren sich nicht darüber im klarem, dass die Fernbedienung über mehr als eine Ansicht verfügt. Nur erfahrene Nutzer sind auf die Idee gekommen, dass noch mehr Ansichten vorhanden sein könnten und haben versucht diese mit den üblichen, aus anderen Applikationen bekannten Gesten, zu erreichen. Nur ein erfahrener Nutzer(Proband 7) hatte hier ebenfalls Probleme.

In Bezug auf Empfehlungen waren sich knapp 50% der Probanden einig, dass es einen Indikator oder wie im Hauptbildschirm die Verwendung von Tabs von Nöten ist um diese Funktion präsenter zu halten. Außerdem wurde mehrfach vorgeschlagen die Grundbedienungsansicht der Fernbedienung als erste Anzeige zu setzen, da diese wahrscheinlich am häufigsten zum Einsatz kommen würde. Ebenfalls wurde angemerkt, dass die Schriftgröße auf den Tasten als zu klein empfunden wurde und die Mediensteuerung durch entsprechend bebilderte Tasten ersetzt werden sollten.

7.6.3 Befunde durch weiterführende Fragen

Durch die im zweiten Teil der Evaluation angeführten Fragen konnten ebenfalls einige interessante Eindrücke gesammelt werden. Auch wenn nur knapp 50% der Probanden das Smartphone mit Absicht in den Landscape Modus gedreht haben, wurde die dort angepasste Ansicht von allen bis auf einen Ausreißer als sehr gut bewertet. Die einzige schlechte Bewertung wurde durch den Probanden dadurch begründet, dass er nie die Landscape Ansicht verwenden würde, da das Smartphone auf diese Weise schlecht in der Hand liegen würde.

In Bezug auf die Android Design Richtlinien wurde die Navigation und Struktur der Applikation durchweg als gut eingestuft. Die Verwendung von aussagekräftigen Icons wurde ebenfalls positiv bewertet. Die einzigen Makel waren, dass die Icons nicht von allen Probanden sofort als solche erkennbar waren. Sie hätten laut Probanden deutlicher gemacht werden sollen, beispielsweise durch einen dreidimensionalen Effekt. In Bezug auf die Darstellung von Icons können die Android Design Richtlinien also durchaus noch einmal überdacht werden.
Die durch die Design Richtlinien vorgeschlagenen Dialogformen und Benachrichtigungen wurden von allen Probanden als sehr gut verständlich und hilfreich eingestuft. Die in der Action Bar integrierte Hoch-Taste wurde nur von einem erfahrenem Probanden verwendet. Alle anderen Probanden haben ausschließlich die altbekannten Zurück-Taste verwendet. Dies deutet darauf hin, dass der Bekanntheitsgrad dieser Taste bei normalen Nutzern noch nicht sehr hoch ist oder dass diese Taste aus anderen Gründen gemieden wird. Ein Beispiel wäre die Bequemlichkeit, da die Zurück-Taste wesentlich leichter mit dem Daumen erreichbar ist als die Hoch-Taste.

Der Funktionsumfang und deren allgemeine Meinung zur Anwendung wurde von allen Probanden als sehr gut bewertet. Des weiteren gaben alle Probanden an, die Bedienung der Applikation bei weiterer Verwendung definitiv schneller und einfacher erledigen zu können und die Funktionsweise nach erstmaliger Anwendung klar war. Dies deutet auf eine steile Lernkurve innerhalb der Anwendung hin, was ebenfalls positiv zu bewerten ist.

Zum Schluss wird eine Verwendung der Applikation durch die Probanden genauer betrachtet. Hier herrscht durchweg eine gespaltene Meinung. Sowohl die Auswertung der spontanen Aufzeichnen einer Sendung, diese auf dem Android Gerät anzuschauen sowie das aktuelle Programm auf diesem wiederzugeben weisen eine hohe Standardabweichung auf. Die beiden zuerst genannten Funktionen wurden dabei nur von weiblichen beziehungsweise den jüngeren Probanden als nützlich bewertet. Bis auf einen Set-Top-Box erfahrenen Probanden(Proband 4) und ein im allgemeinen begeisterter Nutzer(Proband 8) bestand in diesem Bereich ansonsten keinerlei Interesse. Das aktuelle Bild auf einem Android Gerät anzusehen erschien der Mehrzahl der Probanden auch eher als uninteressant, da diese Funktion nur im Heimnetz zur Verfügung steht. Nur die Probanden 4 und 8, sowie ein Tablet Besitzer waren dieser Funktion gegenüber offen.
Auch bei der Frage nach Verwendung der virtuellen oder mitgelieferten Fernbedienung traten sehr unterschiedliche Meinungen zutage. Jeweils etwa 50% der Probanden würden das Android Gerät als Fernbedienung verwenden, da sie dieses immer zur Hand haben. Der andere Teil würde die Hardware Fernbedienung nutzen, da diese ertastbare Knöpfe besitzt und somit eine Verwendung ohne Betrachtung der Fernbedienung möglich sei.

7.7 Abschließende Eindrücke

Im Allgemeinen erscheint die Entwicklung der Applikation gut gelöst. Die Probanden waren sehr zufrieden mit dem Funktionsumfang und der Anwendung an sich. Nur in einigen Bereichen sind noch Probleme aufgetreten, welche eventuell eine Anpassung benötigen und im folgendem Abschnitt noch weiter betrachtet werden sollen.

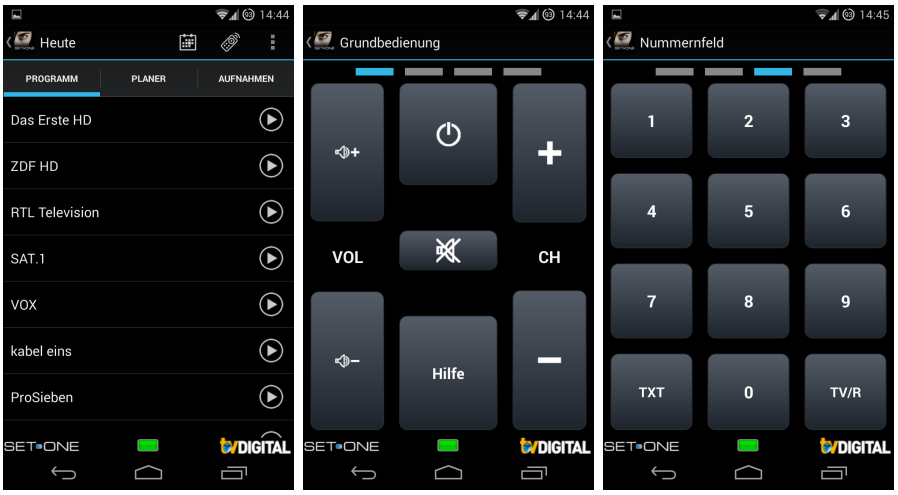
Die Anpassung der Ansicht an die Orientierung des Android Geräts hat große Zustimmung gefunden und die Navigation und Übersichtlichkeit durch das Design ist in jeder Hinsicht gegeben. Die Verwendung der in den Android Design Richtlinien als sehr wichtig dargestellten Action Bar ist nicht in allen Belangen für die Probanden zufriedenstellend gewesen. Die Verwendung der integrierten Hoch-Taste wurde nahezu komplett verweigert oder übersehen. Die durch die Action Bar geleistete Orientierung durch Überschriften, Tabs oder Icons wurde allerdings als sehr positiv eingestuft.

Die Verwendung der Applikation im Alltag wurde von den Probanden vollkommen unterschiedlich bewertet. Auf der einen Seite war Begeisterung und Enthusiasmus zu vernehmen, auf der anderen Seite etwa Gleichgültigkeit oder Abneigung. Dies muss nicht zwangsweise mit der Applikation an sich zusammenhängen, sondern kann auch auf den hier angestrebten Einsatzbereich zurückzuführen sein. Denn nicht jeder Proband ist gleichermaßen am Fernsehprogramm der heutigen Tage interessiert und möchte dieses immer verfügbar haben.

7.8 Anpassungen

In diesem Abschnitt sollen die aufgrund der Evaluation eingeführten Anpassungen innerhalb der Applikation vorgestellt werden. Es wird erläutert warum einige Vorschläge der Probanden Anklang und Umsetzung gefunden haben und warum andere Vorschläge vernachlässigt wurden.

Wie in Abbildung 7.1 (a) zu sehen ist wurde der Text „EPG“ durch „Programm“ ersetzt, da die gesamte Applikation auf deutsch gehalten ist und einige Probleme mit dem Kürzel aufgetreten sind, hat es durchaus Sinn ergeben dies ebenfalls anzugleichen.
Weitere Anpassungen wurden ausschließlich an der Fernbedienung vorgenommen. In den Abbildungen 7.1 (b) und (c) ist zu sehen, dass oberhalb der Fernbedienungsansichten ein Indikator für die verschiedenen Ansichten eingeführt wurde. Dieser soll dem Nutzer vor Augen führen, dass es noch weitere Ansichten der Fernbedienung gibt. Die Verwendung von Indikatoren sollte dem Nutzer bereits bekannt sein. Beispielsweise wird ein solcher Indikator auch auf dem Desktop, dem sogenanntem Homescreen, von Android verwendet.
Außerdem zeigen die Abbildungen, dass die Ansicht der Grundbedienung an die erste Stelle der Fernbedienungsansichten gesetzt wurde. Die Argumente der Probanden über die häufigere Verwendung und die hierfür vorteilhafte schnellere Auswahl dieser Ansicht haben hier überwiegt. Zudem wurde die Schrift der Fernbedienungstasten vergrößert und diese wird zudem nun „fett“ dargestellt. Auf einem Testgerät mit kleinerem Bildschirm war die Notwendigkeit dieser Anpassung noch besser zu sehen.



(a) Programm **(b)** Fernbedienung **(c)** Indikator

Abbildung 7.1: Ansichten nach den Anpassungen

Eine Empfehlung der Probanden, welche nicht umgesetzt wurde, ist das Planen im Bereich des Planers zu ermöglichen. Dies wurde aus mehreren Gründen nicht umgesetzt. Zum einen hat die Evaluation gezeigt, dass die Probanden auch ohne diese Funktion das Szenario eigenständig lösen konnten. Zum anderen hat sich gezeigt, dass die Probanden alle der Meinung waren die Applikation nach einmaliger Verwendung wesentlich schneller und besser bedienen zu können. Wenn die Anwendung der Funktion also erst einmal bekannt ist, ist die Umsetzung also durchaus sinnvoll. Außerdem wurde die Überlegung der Anpassung verstreut, da die Umsetzung einer solchen Funktion mit hoher Wahrscheinlichkeit der EPG-Ansicht sehr ähneln würde und somit nur ein anderer Weg für die gleiche Funktion bereitgestellt werden würde. Des Weiteren herrschte auch seitens der Probanden die Meinung, dass eine Sendung zu einem bestimmtem Zeitpunkt wohl nicht gezielt gesucht werden würde, so wie es im Szenario verlangt war. Es ist eher anzunehmen, dass ein Nutzer spontan eine Aufzeichnung einplanen würde, wenn dieser zufällig über die EPG-Ansicht auf eine interessante Sendung stößt.

Dies sind nach Meinung des Entwicklers sinnvolle Anpassungen der Applikation, welche die Verwendung dieser vereinfachen sollen. Alle Anpassungen sind dabei auf Probleme, Empfehlungen oder weitere Rücksprachen mit den Probanden zurückzuführen.

Kapitel 8

Fazit

Zusammenfassend lässt sich sagen, dass die Zielvorgaben der Arbeit durchaus machbar und auch erreicht worden sind. Die Entwicklung der Android Applikation zur lokalen und entfernten Nutzung eines PVR-Systems wurde im gewünschtem Umfang umgesetzt und der Usability Test hat zu einer weitestgehend positiven Rückmeldung geführt. Im Weiterem soll nun diskutiert werden, inwiefern die Android API die Unterstützung von unterschiedlichen Bildschirmgrößen ermöglicht oder auch erleichtert. Es sollen außerdem die verwendeten Werkzeuge zur Entwicklung der Applikation bewertet werden und die Android Design Richtlinien in Bezug auf ihren Nutzen und ihre Anwendung genauer betrachtet werden. Zudem sollen die zum Einsatz gekommenen Protokolle des ConnectTV-Servers, UPnP, HLS und Sat>IP in ihrer Verwendbarkeit beurteilt werden.

Das zur Entwicklung von Prototypen verwendete Werkzeug FluidUI ist sehr hilfreich gewesen. Gerade in Bezug auf die Gestaltung der Applikation und die damit zusammenhängende Absprache mit der kooperierenden Firma DiscVision. So konnte auf vereinfachte Weise auch eine Rücksprache via E-Mail gehalten werden, indem der Prototyp an den entsprechenden Ansprechpartner gesendet wurde. Die Erstellung eines Prototypen hat zwar zu mehr Arbeit im Vorfeld der Entwicklung geführt, konnte aber somit spätere Änderungen an der eigentlichen Applikation vermeiden und den Programmierprozess verkürzen.

Die Verwendung der Entwicklungsumgebung Eclipse mit eingebundenem Android Development Tool hat die Arbeit ebenfalls in vielerlei Hinsicht vereinfacht. Durch die Entwicklungsumgebung wurde die standardisierte Struktur des Android Projekts bereits bei der Erstellung automatisch umgesetzt. So waren alle benötigten Ordner in den Ressourcen voreingestellt, die Unterstützungsbibliothek für Android eingebunden und die Grundvoraussetzungen einer Applikation bereits im Vorfeld erfüllt. Außerdem ist die Android API bereits integriert und die Verwendung aller Android spezifischen Klassen und Methoden war ohne weiteres möglich. Diese wurde zudem durch die integrierte Autovervollständigung enorm erleichtert. Auch das Debugging Tool innerhalb der Android Development Tools ist gut strukturiert und zur Diagnostizierung von Fehlern sehr hilfreich. Der Entwickler kann mögliche Fehler leichter lokalisieren indem er den Programmcode Schritt für Schritt überprüfen kann. Dies wird zusätzlich durch den Zugriff auf den Programmheap und die

unterschiedlichen Threads unterstützt. Außerdem kann der Entwickler sich zu jeder Zeit, über das sogenannte Logging, Informationen zum aktuellen Programmverlauf oder Variablenwerte über die Konsole ausgeben.

Die Android API stellt insgesamt eine gute Unterstützung für verschiedene Bildschirmgrößen und -ausrichtungen bereit. Zum einen bietet sich die Möglichkeit innerhalb der Ressourcen verschiedene standardisierte Ordner für die entsprechenden Bildschirmgrößen oder -ausrichtungen zu erstellen. Die für das verwendete Endgerät passenden Ordner werden bei Verwendung der Applikation automatisch vom Android System ausgewählt. Somit ermöglicht bereits die Bereitstellung dieser Ordner durch den Entwickler eine Vielzahl von Anpassungen an die Gestaltung der einzelnen Ansichten. Zum anderen wird auch die automatische Anpassung von einzelnen Design-Elementen sehr gut unterstützt. So können Elemente mit Bildschirmgrößenabhängigen und -auflösungsabhängigen Variablen versehen werden, sodass sich die Größe dieser Elemente automatisch an die Bildschirmeigenschaften anpasst. Zudem ist die Verwendung von verschiedenen Sprachdateien sehr gut durchdacht und einfach umzusetzen. Es wird auf simple Art und Weise ermöglicht eine Applikation in vielen Sprachen anzubieten und diese werden ebenfalls bereits vom Betriebssystem automatisch ausgewählt.

Die Android Design Richtlinien machen in vieler Hinsicht Sinn und sind ebenfalls gut durchdacht. Die Evaluation der Applikation hat dabei gezeigt, welche Elemente dem Nutzer helfen und welche noch nicht den gewünschten Erfolg erzielen. Dabei hat sich ergeben, dass die Verwendung der Action Bar in Bezug auf die Navigation und Struktur sehr hilfreich ist. Jedoch findet die dort integrierte Hoch-Taste für eine erweiterte Navigation innerhalb der Anwendung kaum Verwendung. Die von den Design Richtlinien vorgeschlagene Verwendung von Benachrichtigungen und Dialogen bei entsprechenden Aktionen hat ebenfalls innerhalb der Evaluation Anklang gefunden. Außerdem wurden in den meisten Fällen die in der Action Bar eingebetteten Optionsmenüeinträge als positiv bewertet, da sie so schneller erreichbar sind. Die verwendeten Icons sollten allerdings noch besser als bedienbare Elemente deutlich gemacht werden.

Die Verwendung der ConnectTV Schnittstelle hat sich als simpel und gut durchdacht gezeigt. Die verfügbaren Funktionen wurden durch eine API gut beschrieben. Die Antworten des Servers konnten durch ihre XML Struktur gut ausgelesen und weiterverarbeitet werden. Das einzige Problem der ConnectTV Anbindung ist die Fehlfunktion innerhalb der Verwendung von HTTP Live Streaming. Dort werden die verfügbaren Daten angezeigt, doch leider wird auf Anfrage der Daten kein Stream zur Darstellung des Videoinhalts bereitgestellt. Hier muss die DiscVision GmbH den entsprechenden Fehler in der Implementation noch ausfindig machen und korrigieren. Ist dieser korrigiert, sollte die Verwendung dieser Funktion allerdings reibungslos in Betrieb genommen werden können, da sie mit anderen Streams bereits getestet wurde.

Das UPnP Protokoll wurde in dieser Arbeit nicht so verwendet wie dies zu Beginn vorgesehen war. Die Auffindung der Datei wurde dabei nicht über den UPnP-Medien-Browser durchgeführt, sondern über einen einfachen Workaround. Für das Abspielen dieser Datei wurde allerdings der UPnP-Medienserver verwendet und somit die direkte Verbindung zur Set-Top-Box hergestellt.
Das UPnP-Protokoll wurde dennoch als Grundlage für das Sat>IP Protokoll verwendet und dort erfolgreich umgesetzt. Das Protokoll ist gut zu verwenden und klar gegliedert. Die Funktionen der verwendeten Geräte waren dabei immer ohne Probleme anzusteuern und zu jeder Zeit ver-

fügbar. Auch in Bezug auf die Verwendung von Sat>IP muss die DiscVision GmbH hier jedoch noch nacharbeiten. Durch den ConnectTV-Server sollten die für die Erstellung eines Streams benötigten Daten, wie Frequenz oder Polarisation bereitgestellt werden. Doch dies ist bis zum Abschluss der Arbeit leider nicht umgesetzt worden. Eine geeignete, kostenlose Alternative wurde nicht gefunden. Die Sat>IP Funktion ist in der Applikation allerdings funktionsfähig. Bei Aufruf dieser Funktion werden feste Parameter übergeben, sodass immer der gleiche Sender abgespielt wird. Dies ist für die Marktreife der Applikation zwar nicht sinnvoll, zeigt aber das diese Funktion korrekt umgesetzt wurde. Für die Einführung der Applikation in den Markt müssten somit noch die zwei genannten Probleme gelöst werden.

Im Allgemeinen ist zu sagen, dass die Implementierung von Android Applikationen auch für verschiedene Displaygrößen sehr ausgereift ist. Durch die gut durchdachte API wird dem Entwickler einiges an Arbeit erspart und bietet viele Möglichkeiten der Anpassung. Die Leistung des Testgeräts hat für alle implementierten Funktionen ausgereicht und eine flüssige Darstellung von Videoinhalten ist auch über das Netzwerk gut möglich. Die verwendeten Protokolle wirken alle gut durchdacht und ausgereift. Nur in Bezug auf eine Fehlfunktion und einen noch nicht implementierten Service seitens des ConnectTV-Servers müssen vom Gesamteindruck Abstriche gemacht werden.

Abbildungsverzeichnis

Listings

Literaturverzeichnis

[ALT] ALTECH SetOne: *Genius HD.* http://www.setone.eu/Support-Detail.95.0.html?&cHash=903c6a4a71&tx_ttnews%5BbackPid%5D=94&tx_ttnews%5Btt_news%5D=1175, Abruf: Januar. 2014

[Anda] Android Developers: *Action Bar.* http://developer.android.com/guide/topics/ui/actionbar.html, Abruf: Januar. 2014

[Andb] Android Developers: *Activities.* http://developer.android.com/guide/components/activities.html, Abruf: Januar. 2014

[Andc] Android Developers: *Activity.* http://developer.android.com/reference/android/app/Activity.html, Abruf: Januar. 2014

[Andd] Android Developers: *ADT Plugin.* http://developer.android.com/tools/sdk/eclipse-adt.html, Abruf: Januar. 2014

[Ande] Android Developers: *Android System Architektur.* http://developer.android.com/images/system-architecture.jpg, Abruf: Januar. 2014

[Andf] Android Developers: *App Resources.* http://developer.android.com/guide/topics/resources/index.html, Abruf: Januar. 2014

[Andg] Android Developers: *Fragments.* http://developer.android.com/guide/components/fragments.html, Abruf: Januar. 2014

[Andh] Android Developers: *Honeycomb.* http://developer.android.com/about/versions/android-3.0-highlights.html, Abruf: Januar. 2014

[Andi] Android Developers: *Navigation with Back and Up.* https://developer.android.com/design/patterns/navigation.html, Abruf: Januar. 2014

[Andj] Android Developers: *Patterns.* https://developer.android.com/design/patterns/index.html, Abruf: Januar. 2014

[Andk] Android Developers: *Processes and Threads.* http://developer.android.com/guide/components/processes-and-threads.html, Abruf: Januar. 2014

[Andl] Android Developers: *Storage Options.* http://developer.android.com/guide/topics/data/data-storage.html, Abruf: Januar. 2014

[Andm] Android Developers: *StrictMode.* http://developer.android.com/reference/android/os/StrictMode.html, Abruf: Januar. 2014

[Andn] Android Developers: *Support Library.* http://developer.android.com/tools/support-library/index.html?utm_content=buffer11f41&utm_source=buffer&utm_medium=twitter&utm_campaign=Buffer, Abruf: Januar. 2014

[Ando] Android Developers: *Supporting Different Screens.* http://developer.android.com/training/basics/supporting-devices/screens.html, Abruf: Januar. 2014

[Andp] Android Developers: *Tasks and Back Stack.* http://developer.android.com/guide/components/tasks-and-back-stack.html, Abruf: Januar. 2014

[Andq] Android Developers: *User Interface.* https://developer.android.com/guide/topics/ui/index.html, Abruf: Januar. 2014

[Andr] Android Developers: *Using ViewPager for Screen Slides.* http://developer.android.com/training/animation/screen-slide.html, Abruf: Januar. 2014

[App] Apple Inc.: *HTTP Streaming Architecture.* https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html#//apple_ref/doc/uid/TP40008332-CH101-SW2, Abruf: Januar. 2014

[Deu06] Deutsches Institut für Normung e.V.: *Ergonomie der Mensch-System-Interaktion - Teil 110: Grundsätze der Dialoggestaltung (DIN EN ISO 9241-110).* 2006

[Disa] DiscVision GmbH: *Company About DiscVision.* http://discvision.de/, Abruf: Januar. 2014

[Disb] DiscVision GmbH: *connectTV.* http://discvision.de/, Abruf: Januar. 2014

[Ecl] Eclipse: *About the Eclipse Foundation.* https://www.eclipse.org/org/, Abruf: Januar. 2014

[Flua] Fluid: *About us.* https://www.fluidui.com/aboutus, Abruf: Januar. 2014

[Flub] Fluid: *Documentation.* https://www.fluidui.com/documentation, Abruf: Januar. 2014

[Gar] Gartner: *Gartner Says Asia/Pacific Led Worldwide Mobile Phone Sales to Growth in First Quarter of 2013.* http://www.gartner.com/newsroom/id/2482816, Abruf: Januar. 2014

[Gooa] Google: *Android Open Source Project - Issue Tracker.* https://code.google.com/p/android/issues/detail?id=27609, Abruf: Januar. 2014

[Goob] Google Play Store: *MX Player*. https://play.google.com/store/apps/details?id=com.mxtech.videoplayer.ad&hl=de, Abruf: Januar. 2014

[HS88] Hart, Sandra G. ; Staveland, Lowell E.: *Development of NASA-TLX (Task Load Index): Results of empirical and theo-retical research*. Elsevier Science Publishing Company, 1988. – ISBN 978–0444703880

[iet13] ietf.org: *HTTP Live Streaming*. http://tools.ietf.org/pdf/draft-pantos-http-live-streaming-12.pdf. Version: Oktober 2013

[Jak] JakeWharton: *ActionBarSherlock*. http://actionbarsherlock.com/, Abruf: Januar. 2014

[KM12] Komatineni, Satya ; MacLean, Dave: *Pro Android 4*. Apress, 2012. – ISBN 978–1–4302–3930–7

[Kün12] Künneth, Thomas: *Android 4 - Apps entwickeln mit dem Android SDK*. Galileo Press, 2012. – ISBN 978–3–8362–1948–8

[MDMN12] Mednieks, Zigurd ; Dornin, Laird ; Meike, G. B. ; Nakamura, Masumi: *Programming Android*. O'Reilly, 2012. – ISBN 978–1–449–31664–8

[Mei12] Meier, Reto: *Professional Android 4 Application Development*. John Wiley & Sons, Inc., 2012. – ISBN 978–1–118–10227–5

[MX] MX Player: *API - Intent definition*. https://sites.google.com/site/mxvpen/api, Abruf: Januar. 2014

[Opea] Open Automotive Alliance: *Introducing the Open Automotive Alliance*. http://www.openautoalliance.net/#about, Abruf: Januar. 2014

[Opeb] Open Handset Alliance: *About*. http://www.openhandsetalliance.com/, Abruf: Januar. 2014

[Opec] Open Handset Alliance: *Members*. http://www.openhandsetalliance.com/oha_members.html, Abruf: Januar. 2014

[Phi] Philips: *TP Vision kündigt Philips Fernseher "powered byÄndroid an*. http://www.newscenter.philips.com/de_de/standard/news/consumerlifestyle/20140106_TPVision_kuendigt_Philips_Fernseher_poweredby_Android_an.wpd#.Ut5YZBCIXIU, Abruf: Januar. 2014

[SESa] SES S.A.: *SAT>IP Network*. http://www.satip.info/, Abruf: Januar. 2014

[SESb] SES S.A.: *SAT>IP Physical layer independent satellite distribution to IP devices*. http://www.satip.info/sites/satip/files/files/satip-white-paper.pdf, Abruf: Januar. 2014

[SES12] SES S.A.: *SES unveils IP-based in-home distribution of satellite TV signals*. http://www.ses.com/4233325/news/2012/11403011. Version: April 2012

[SES13] SES S.A.: *SAT>IP Protocol Specification Version 1.2.* http://www.satip.info/sites/satip/files/resource/satip_specification_version_1_2.pdf. Version: März 2013

[UPna] UPnP Forum: *Device Control Protocols.* http://upnp.org/index.php/sdcps-and-certification/standards/sdcps/, Abruf: Januar. 2014

[UPnb] UPnP Forum: *What is UPnP?* http://upnp.org/about/what-is-upnp/, Abruf: Januar. 2014

[UPn08] UPnP Forum: *UPnP Device Architecture 1.1.* http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf. Version: Oktober 2008

[URSW+88] Unema, P. ; Rötting, M. ; Sepher-Willenberg, M. ; Strümpfel, U. ; Kopp, U.: *Der NASA Task Load Index: Erste Ergebnisse mit der deutschen Fassung.* Jahresdokumentation 1988 der Gesellschaft für Arbeitswissenschaft e.V. - Bericht zum 34. Arbeitswissenschaftlichen Kongress an der RWTH Aachen. Gesellschaft für Arbeitswissenschaft e.V., 1988

[Wif] Wifimaku.com: *Mobiltelefon-Betriebssysteme.* http://wifimaku.com/online-marketing/mobile-marketing-und-mobile-applikationen/mobile-applikationen-und-ger%C3%A4te/mobiltelefon-betriebssysteme, Abruf: Januar. 2014

Anhang A

Unterlagen zur Evaluation

A.1 Ablaufplan

Ablaufplan

Vor Eintreffen des Probanden:

1. Fenster schließen
2. TV
   - Einschalten
   - Input HDMI-2 auswählen
3. Set-Top-Box
   - Einschalten
   - Ton ausschalten
   - Programmkanal 4 auswählen
4. Smartphone
   - Einschalten
   - WLAN aktivieren
   - Applikation öffnen
   - Vorherige Nutzerdaten entfernen
5. Szenarien, Fragebogen und Stift bereitlegen

Nach Eintreffen des Probanden:

1. Tür schließen
2. Begrüßung und Vorstellung
3. Mündliche Einweisung
   - Worum geht es?
     1. Masterarbeit
     2. Android Applikation geschrieben
     3. Bedienung der Set-Top-Box
     4. Set-Top-Box ist ein Receiver zum TV schauen
     5. Set-Top-Box kann Aufnahmen machen
     6. Hat Anschluss an das Internet --> Bedienung mit Smartphone
     7. Getestet wird die Anwendung und nicht wie gut der Proband ist
     8. Anonym
   - Was ist zu tun?
     1. Teil 1: 6 Szenarien/ Aufgaben
     2. Teil 1: Aufgaben so gut wie möglich erfüllen und nach jeder Aufgabe Fragebogenteil zur Schwierigkeit der Aufgabe ausfüllen
        (Skala 1-5, Bewertung ist gleich bleibend und steht immer dran)
     3. Teil 2: Zusätzliche Fragen zur Applikation

- Die Aufgaben immer versuchen eigenständig zu erfüllen, im Fall von Fragen bin ich aber im Raum
- Wenn Aufgabenerfüllung misslingt, dann ist das auch ok und es kann mit nächster weiter gemacht werden

4. Fragebogen noch einmal kurz zeigen und Reihenfolge erläutern
5. Gibt es noch Fragen?
6. Fragebogen übergeben
7. Ton einschalten
8. Starten

Nach Beendigung des Tests:

1. Fragebogen kurz überprüfen
2. Probanden fragen ob er noch etwas spezielles zur Applikation sagen möchte
3. Bedanken inklusive kleinem Präsent
4. Verabschieden
5. TV ausschalten
6. Set-Top-Box ausschalten
7. Smartphone ausschalten
8. Fragebogen abheften

A.2 Szenarien

Teil 1

Szenario 1: Anmelden

Bei Erhalt des Smartphones ist die Applikation bereits gestartet. Bitte melden Sie sich mit folgenden Daten im System an.

Name: Sebastian

Passwort: qqqqqq

Szenario 2: Planen einer Aufnahme

Versuchen Sie die Aufzeichnung einer Sendung zu planen, welche übermorgen gegen 19 Uhr auf dem Sender „RTL2“ ausgestrahlt wird.

Szenario 3: Ansehen einer Aufnahme

Sehen Sie sich eine bereits vorhandene Aufzeichnung einer Sendung an. Kehren sie nach einigen Sekunden wieder zur Applikation zurück.

Szenario 4: Löschen einer Aufnahme

Löschen Sie die von Ihnen in Szenario 2 geplante Aufzeichnung wieder. Wählen sie danach wieder das heutige Datum aus.

Szenario 5: Fernbedienung

Wählen Sie die Fernbedienung aus. Schalten Sie auf Kanal 7. Und schalten Sie dann den Ton ab.

Szenario 6: Abspielen des aktuellen Programms

Starten Sie die Wiedergabe des Senders „ProSieben“ auf dem Smartphone. Kehren sie nach einigen Sekunden wieder zur Applikation zurück.

A.3 Fragebogen

Test Nr: Datum:

Fragebogen

Alter: _____

Geschlecht: ☐ Männlich ☐ Weiblich

Bitte Stufen sie ihre Vorkenntnisse ein:

keine – Profi

	1	2	3	4	5
Erfahrung mit einem Smartphone					
Erfahrung mit dem Betriebssystem Android					
Erfahrung mit Set-Top-Boxen					

Test Nr: Datum:

Teil 1

Szenario 1: Anmelden

Geistige Anforderung: Wie hoch war die geistige Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Körperliche Anforderung: Wie hoch war die körperliche Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Zeitliche Anforderung: Wie empfandst du den Zeitaufwand beim Bearbeiten der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Ausführung der Aufgaben: Wie erfolgreich hast du deiner Meinung nach die vom Versuchsleiter gesetzten Ziele erreicht? Wie zufrieden warst du bei deiner Leistung bei der Verfolgung deiner Ziele?

gut		–		schlecht
1	**2**	**3**	**4**	**5**

Anstrengung: Wie hart musstest du arbeiten um deinen Grad an Aufgabenerfüllung zu erreichen?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Frustration: Wie unsicher, entmutigt, irritiert, gestresst und verärgert (versus sicher, bestätigt, zufrieden und entspannt) fühltest du dich während der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Test Nr: Datum:

Szenario 2: Planen einer Aufnahme

Geistige Anforderung: Wie hoch war die geistige Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Körperliche Anforderung: Wie hoch war die körperliche Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Zeitliche Anforderung: Wie empfandst du den Zeitaufwand beim Bearbeiten der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Ausführung der Aufgaben: Wie erfolgreich hast du deiner Meinung nach die vom Versuchsleiter gesetzten Ziele erreicht? Wie zufrieden warst du bei deiner Leistung bei der Verfolgung deiner Ziele?

gut		–		schlecht
1	**2**	**3**	**4**	**5**

Anstrengung: Wie hart musstest du arbeiten um deinen Grad an Aufgabenerfüllung zu erreichen?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Frustration: Wie unsicher, entmutigt, irritiert, gestresst und verärgert (versus sicher, bestätigt, zufrieden und entspannt) fühltest du dich während der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Test Nr: Datum:

Szenario 3: Ansehen einer Aufnahme

Geistige Anforderung: Wie hoch war die geistige Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Körperliche Anforderung: Wie hoch war die körperliche Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Zeitliche Anforderung: Wie empfandst du den Zeitaufwand beim Bearbeiten der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Ausführung der Aufgaben: Wie erfolgreich hast du deiner Meinung nach die vom Versuchsleiter gesetzten Ziele erreicht? Wie zufrieden warst du bei deiner Leistung bei der Verfolgung deiner Ziele?

gut		–		schlecht
1	**2**	**3**	**4**	**5**

Anstrengung: Wie hart musstest du arbeiten um deinen Grad an Aufgabenerfüllung zu erreichen?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Frustration: Wie unsicher, entmutigt, irritiert, gestresst und verärgert (versus sicher, bestätigt, zufrieden und entspannt) fühltest du dich während der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Test Nr: Datum:

Szenario 4: Löschen einer Aufnahme

Geistige Anforderung: Wie hoch war die geistige Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Körperliche Anforderung: Wie hoch war die körperliche Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Zeitliche Anforderung: Wie empfandst du den Zeitaufwand beim Bearbeiten der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Ausführung der Aufgaben: Wie erfolgreich hast du deiner Meinung nach die vom Versuchsleiter gesetzten Ziele erreicht? Wie zufrieden warst du bei deiner Leistung bei der Verfolgung deiner Ziele?

gut		–		schlecht
1	**2**	**3**	**4**	**5**

Anstrengung: Wie hart musstest du arbeiten um deinen Grad an Aufgabenerfüllung zu erreichen?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Frustration: Wie unsicher, entmutigt, irritiert, gestresst und verärgert (versus sicher, bestätigt, zufrieden und entspannt) fühltest du dich während der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Test Nr: Datum:

Szenario 5: Fernbedienung

Geistige Anforderung: Wie hoch war die geistige Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Körperliche Anforderung: Wie hoch war die körperliche Anstrengung bei der Bearbeitung der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Zeitliche Anforderung: Wie empfandst du den Zeitaufwand beim Bearbeiten der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Ausführung der Aufgaben: Wie erfolgreich hast du deiner Meinung nach die vom Versuchsleiter gesetzten Ziele erreicht? Wie zufrieden warst du bei deiner Leistung bei der Verfolgung deiner Ziele?

gut		–		schlecht
1	**2**	**3**	**4**	**5**

Anstrengung: Wie hart musstest du arbeiten um deinen Grad an Aufgabenerfüllung zu erreichen?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Frustration: Wie unsicher, entmutigt, irritiert, gestresst und verärgert (versus sicher, bestätigt, zufrieden und entspannt) fühltest du dich während der Aufgabe?

gering		–		hoch
1	**2**	**3**	**4**	**5**

Test Nr: Datum:

Szenario 6: Abspielen des aktuellen Programms

Geistige Anforderung: Wie hoch war die geistige Anstrengung bei der Bearbeitung der Aufgabe?

gering – hoch

1	2	3	4	5

Körperliche Anforderung: Wie hoch war die körperliche Anstrengung bei der Bearbeitung der Aufgabe?

gering – hoch

1	2	3	4	5

Zeitliche Anforderung: Wie empfandst du den Zeitaufwand beim Bearbeiten der Aufgabe?

gering – hoch

1	2	3	4	5

Ausführung der Aufgaben: Wie erfolgreich hast du deiner Meinung nach die vom Versuchsleiter gesetzten Ziele erreicht? Wie zufrieden warst du bei deiner Leistung bei der Verfolgung deiner Ziele?

gut – schlecht

1	2	3	4	5

Anstrengung: Wie hart musstest du arbeiten um deinen Grad an Aufgabenerfüllung zu erreichen?

gering – hoch

1	2	3	4	5

Frustration: Wie unsicher, entmutigt, irritiert, gestresst und verärgert (versus sicher, bestätigt, zufrieden und entspannt) fühltest du dich während der Aufgabe?

gering – hoch

1	2	3	4	5

Test Nr: Datum:

Teil 2

Haben Sie während des Tests das Smartphone gedreht um eine andere Ansicht der Daten zu bekommen?

☐ Ja ☐ Nein

Wenn Ja, war dies mit Absicht oder sind Sie versehentlich darauf gestoßen?

☐ Absicht ☐ Versehentlich

Wenn Nein, drehen Sie dies bitte jetzt und testen Sie kurz einige Funktionen.

Halten Sie die Darstellung der Applikation im Querformat für ein Smartphone geeignet?

geeignet – ungeeignet

1	2	3	4	5

Wie bewerten Sie die Struktur und Navigation der Anwendung?

gut – schlecht

1	2	3	4	5

Haben Sie zum Navigieren in der Applikation den Zurück-Button (unten links) oder den Up-Button (oben links) verwendet?

☐ Zurück- Button ☐ Up-Button ☐ Beide

Wie bewerten Sie den Funktionsumfang der Anwendung?

gut – schlecht

1	2	3	4	5

Test Nr: Datum:

Wie ist Ihre allgemeine Meinung zu der Anwendung?

gut		–		schlecht
1	**2**	**3**	**4**	**5**

Halten Sie die Verwendung von Icons anstatt Beschriftungen für sinnvoll?

sinnvoll		–		schlecht
1	**2**	**3**	**4**	**5**

Halten Sie die verwendeten Icons für passend und aussagekräftig?

passend		–		schlecht
1	**2**	**3**	**4**	**5**

Wie bewerten Sie die Verständlichkeit der Dialoge? (Wollen Sie wirklich löschen?, etc.)

gut		–		schlecht
1	**2**	**3**	**4**	**5**

Würden Sie gern Sendungen spontan mit Ihrem Smartphone oder Tablet aufzeichnen?

gerne		–		unwichtig
1	**2**	**3**	**4**	**5**

Test Nr: Datum:

Stellen Sie sich jetzt vor, Sie besitzen eine solche Set-Top-Box.

Würden Sie aufgezeichnete Sendungen auf einem Smartphone oder Tablet anschauen?

gerne		–		unwichtig
1	**2**	**3**	**4**	**5**

Würden Sie gerne auf Ihrem Smartphone oder Tablet fernsehen?

gerne		–		unwichtig
1	**2**	**3**	**4**	**5**

Würden Sie die Box von Ihrem Sofa aus eher mit dem Smartphone oder der mitgelieferten Fernbedienung bedienen?

☐ Smartphone ☐ Fernbedienung

Glauben Sie, Sie würden die Aufgaben aus Teil 1 in einem zweiten Durchlauf schneller und einfacher lösen können?

Ganz klar		–		nein
1	**2**	**3**	**4**	**5**

A.4 Anmerkungen

Anmerkungen der Probanden während der Usability Tests:

Test 1:

- Icons nicht klar erkennbar
- Login Textfeld Cursor nicht zu sehen

Test 2:

- Fernbedienung mit Tabs oder Indikator versehen
- Fernbedienung in Grundbedienung starten
- Planen im Planer gesucht/ermöglichen
- Medien Steuerung mit Bilderbuttons

Test 3:

- S2 mit Hilfe gelöst
- Fernbedienung nur die erste Ansicht gefunden
- S5 nicht vollständig gelöst

Test 4:

- Fernbedienung mit Tabs oder Indikator versehen
- Login Textfeld markieren
- Tutorial bei Erststart

Test 5:

- Login Textfeld markieren
- Planen im Planer gesucht/ermöglichen
- S2 mit Hilfe gelöst
- Fernbedienung nur die erste Ansicht gefunden
- S5 mit Hilfe gelöst

Test 6:

- Login Textfeld markieren
- Fernbedienung nur die erste Ansicht gefunden
- S5 mit Hilfe gelöst

Test 7:

- Planen im Planer gesucht
- Fernbedienung nur die erste Ansicht gefunden
- Aufgezeichnete Sendung nur entfernt schauen, sonst TV

Test 8:

- Fernbedienung nur die erste Ansicht gefunden
- S5 mit Hilfe gelöst
- Planen im Planer gesucht
- S3: Video auf TV anzeigen lassen können
- Name EPG unklar, Änderung in Programm
- Zahlen-/Schriftgröße größer machen
- Medien Steuerung mit Bilderbuttons

Test 9:

- Fernbedienung nur die erste Ansicht gefunden
- Kalender Icon nicht sofort ersichtlich
- Planen im Planer gesucht
- Name EPG unklar, Änderung in Programm
- S5 mit Hilfe gelöst

Test 10:

- Planen im Planer gesucht
- Fernbedienung nur die erste Ansicht gefunden
- Fernbedienung bei Erststart mit Tutorial
- S5 nur mit Hilfe gelöst
- Login Cursor nicht da

Test 11:

- Fernbedienung nur die erste Ansicht gefunden
- Fernbedienung mit Tabs versehen
- S5 nur mit Hilfe

Test 12:

- Fernbedienung nicht klar, dass mehrere Ansichten
- Login Cursor nicht da
- Fernbedienung in Grundbedienung starten

Test 13:

- Planen im Planer gesucht
- Fernbedienung nur die erste Ansicht gefunden
- Tabs in Fernbedienung hilfreich

A.5 Ergebnisse

Test-Nummer	1	2	3	4	5	6	7	8	9	10
Alter	27	27	26	27	26	26	28	31	23	22
Geschlecht	m	m	m	m	m	m	m	m	w	w
Vorerfahrung mit einem Smartphone	5	4	3	5	3	2	5	3	3	3
Vorerfahrung mit Android	5	4	2	3	3	4	5	3	3	3
Vorerfahrung mit einer STB	1	1	1	4	1	1	1	2	1	1
Teil 1										
1.1	1	1	1	1	1	1	1	1	1	1
1.2	1	1	1	1	1	1	1	1	1	1
1.3	2	1	1	1	1	1	1	1	1	1
1.4	2	1	1	1	1	1	1	1	1	1
1.5	1	1	1	2	1	1	1	1	1	1
1.6	3	1	1	2	1	1	1	1	1	1
Effektivität	4	4	4	4	4	4	4	4	4	4
2.1	3	2	1	2	4	3	3	2	3	4
2.2	1	1	1	2	2	1	1	1	1	1
2.3	3	2	1	2	5	1	3	1	2	3
2.4	2	1	1	1	1	2	1	1	2	3
2.5	4	1	1	2	3	2	2	1	2	4
2.6	2	1	1	2	3	2	4	1	1	4
Effektivität	4	4	3	4	3	4	4	4	4	4
3.1	1	1	1	1	1	1	1	1	1	1
3.2	1	1	1	1	1	1	1	1	1	1
3.3	1	1	1	1	2	1	1	1	1	1
3.4	1	1	1	1	1	1	1	1	1	1
3.5	1	1	1	1	1	1	1	1	1	1
3.6	1	1	2	2	1	1	1	2	1	1
Effektivität	4	4	4	4	4	4	4	4	4	4
4.1	2	1	1	3	1	1	1	2	1	1
4.2	1	1	1	1	1	1	1	1	1	1
4.3	2	1	1	2	2	1	1	1	1	1
4.4	4	1	1	1	1	1	1	2	1	1
4.5	2	1	1	2	2	1	1	1	1	1

4.6	2	1	1	2	2	1	1	2	1	1
Effektivität	4	4	4	4	4	4	4	4	4	4
5.1	1	2	2	2	4	2	3	3	5	3
5.2	1	1	1	1	2	1	1	1	1	1
5.3	1	2	2	2	5	3	3	3	4	2
5.4	1	2	3	1	1	1	1	2	3	2
5.5	1	1	1	2	5	1	3	3	4	3
5.6	1	1	2	2	4	1	4	2	2	2
Effektivität	4	4	2	4	3	3	4	3	3	3
6.1	1	1	1	2	1	1	1	1	2	1
6.2	1	1	1	1	1	1	1	1	1	1
6.3	1	1	1	2	1	1	1	1	2	1
6.4	1	1	1	1	1	1	1	1	2	1
6.5	1	1	1	1	1	2	1	1	2	1
6.6	1	1	1	2	1	1	1	1	1	1
Effektivität	4	4	4	4	4	4	4	4	4	4
Teil 2										
1	1	2	1	2	2	1	1	1	1	1
2	2	0	1	0	0	1	2	1	1	1
3	2	1	2	1	4	1	1	1	1	1
4	2	2	2	2	2	2	2	2	2	3
5	3	1	1	1	1	1	1	1	1	1
6	1	1	3	1	1	1	2	1	1	1
7	2	1	3	2	1	1	2	1	1	1
8	3	3	3	2	2	1	1	1	1	2
9	2	2	2	2	2	1	2	2	2	1
10	1	1	1	1	1	2	1	2	1	1
11	3	5	4	1	3	4	3	1	1	1
12	4	3	4	1	4	3	2	1	1	1
13	4	4	5	2	4	4	5	1	3	1
14	1	2	2	2	2	2	2	1	1	1
15	1	1	3	1	1	1	1	1	1	1

11	12	13	Minimum	Maximum	Mittelwert	Standardabweichung
23	28	23	22	31	25,92	2,56
w	m	w	-	-	-	-
3	5	3	2	5	3,62	1,04
3	5	3	2	5	3,54	0,97
1	4	1	1	4	1,54	1,13
1	1	1	1	1	1,00	0,00
1	1	1	1	1	1,00	0,00
1	1	1	1	2	1,08	0,28
1	1	1	1	2	1,08	0,28
1	1	1	1	2	1,08	0,28
1	1	1	1	3	1,23	0,60
4	4	4	4	4	4,00	0,00
3	2	4	1	4	2,77	0,93
1	1	2	1	2	1,23	0,44
2	2	3	1	5	2,31	1,11
3	2	3	1	3	1,77	0,83
2	1	4	1	4	2,23	1,17
1	2	2	1	4	2,00	1,08
4	4	4	3	4	3,85	0,38
1	1	1	1	1	1,00	0,00
1	1	1	1	1	1,00	0,00
1	1	1	1	2	1,08	0,28
1	1	1	1	1	1,00	0,00
1	1	1	1	1	1,00	0,00
1	1	1	1	2	1,23	0,44
4	4	4	4	4	4,00	0,00
2	2	1	1	3	1,46	0,66
1	1	1	1	1	1,00	0,00
1	1	1	1	2	1,23	0,44
1	1	1	1	4	1,31	0,85
1	1	1	1	2	1,23	0,44

1	1	1	1	2	1,31	0,48			
4	4	4	4	4	4,00	0,00			
4	2	4	1	5	2,85	1,14			
1	1	2	1	2	1,15	0,38			
3	2	3	1	5	2,69	1,03			
2	1	4	1	4	1,85	0,99			
2	1	4	1	5	2,38	1,39			
1	2	2	1	4	2,00	1,00			
3	4	4	2	4	3,38	0,65			
1	1	2	1	2	1,23	0,44			
1	1	1	1	1	1,00	0,00			
1	1	2	1	2	1,23	0,44			
1	1	2	1	2	1,15	0,38			
1	1	1	1	2	1,15	0,38			
1	1	1	1	2	1,08	0,28			
4	4	4	4	4	4,00	0,00			
1	1	1	-	-	-	-			
1	2	2	-	-	-	-			
1	1	1	1	4	1,38	0,87			
1	2	1	1	3	1,92	0,49			
1	1	1	-	-	-	-			
1	1	1	1	3	1,23	0,60			
1	2	1	1	3	1,46	0,66			
1	2	2	1	3	1,85	0,80			
1	2	1	1	2	1,69	0,48			
1	1	1	1	2	1,15	0,38			
1	3	1	1	5	2,38	1,45			
2	4	2	1	4	2,46	1,27			
2	2	3	1	5	3,08	1,38			
1	2	1	-	-	-	-			
1	1	2	1	3	1,23	0,60			

A.6 Probleme und Vorschläge

Probleme:

Problem	Häufigkeit
Fernbedienung nur die erste Ansicht gefunden	8
Planen zunächst im Planer gesucht	7
Login Textfeld nicht markiert	5
Name EPG ist unklar	3
Icons nicht klar erkennbar	2

Vorschläge:

Vorschlag	Häufigkeit
Fernbedienung mit Tabs oder Indikator versehen	6
Fernbedienung mit "Grundbedienung"-Ansicht starten	2
Planen im Planer ermöglichen	2
EPG in Programm umbenennen	2
Medien Steuerung mit Bilderbuttons	2
Tutorial bei Erststart	1
Zahlen-/Schriftgröße größer machen	1
Fernbedienung bei Erststart mit Tutorial versehen	1

Inhalt der CD

Android-Applikation

Masterarbeit

Dokumentation

Unterlagen zur Evaluation

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Osnabrück, März 2014