HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

DIPLOMARBEIT

Herr
Stefan Berger

Webbasierendes Interface für einen Software-Echtzeitencoder

2017

Fakultät **Ingenieurwissenschaften**

DIPLOMARBEIT

Webbasierendes Interface für einen Software-Echtzeitencoder

Autor:
Stefan Berger

Studiengang:
Elektro- und Informationstechnik

Seminargruppe:
EI11wI-D

Erstprüfer:
Herr Prof. Dr.-Ing. Frank Zimmer

Zweitprüfer:
Herr Dipl.-Ing. Birger Jesch

Mittweida, 07 2017

Bibliografische Angaben

Berger, Stefan: Webbasierendes Interface für einen Software-Echtzeitencoder, 81 Seiten, 14 Abbildungen, 2, Hochschule Mittweida, University of Applied Sciences, Fakultät Ingenieurwissenschaften

Diplomarbeit, 2017

Referat

Wie kann man einen grafische Oberfläche für eine Kommandozeilensoftware gestalten? Wie programmiert man sie, das sie plattformübergreifend ist und welche Programmiersprachen sind zu benutzen? Welche Komponenten benötigt man neben dem Videoencoder, um ein funktionierendes System aus der Hard- und Software aufzubauen? Diese und weitere Fragestellungen soll diese Diplomarbeit klären.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Vorwort

Die Bereitstellung von Online-Streams wird in der heutigen Zeit, in der das Medium Fernsehen an Bedeutung verloren hat, immer wichtiger. Das Angebot von Inhalten im Internet ist nicht nur auf gespeicherte Texte, Bilder, Musik und Videos beschränkt, sondern bietet auch die Möglichkeit der Verbreitung von Live-Aufzeichnungen, die der Zuschauer in Echtzeit verfolgen kann.
An der Fakultät Medien der Hochschule Mittweida wird der Software-Encoder FFmpeg für die Lehre und für öffentliche Veranstaltungen genutzt, um die Aufzeichnung und Konvertierung der Video- und Audiosignale zuverlässig zu gewährleisten. Die Steuerung dieses Encoders wird über die Kommandozeile vollzogen und ist somit, wegen seines Umfangs und den Möglichkeiten, sehr komplex in der Anwendung. Des Weiteren werden, weil dieser auf einem Linux-System verwendet wird, auch vertiefte Kenntnisse über Unix-Betriebssysteme benötigt und so wird eine Verwendung für ungeübte Benutzer erschwert.
So entstand die Idee, eine grafische Oberfläche in Form eines Webinterfaces und auf Basis eines Bash-Skriptes zur Steuerung für FFmpeg zu entwickeln, um die Handhabbarkeit zu erleichtern und die Einstellungen für den Benutzer übersichtlicher und greifbarer zu machen.

1 Einleitung

„Für einen unter dem Betriebssystem Linux laufenden Software-Videoencoder soll zur Steuerung und Parametrisierung desselben ein webbasierendes Interface entwickelt werden.“[1]
Es soll ein Prototyp für eine Schnittstelle zwischen einem Software-Encoder und dem Benutzer in einer Linux-Betriebssystemumgebung entworfen werden. Diese Schnittstelle soll als Webinterface ausgeführt und in den Programmier- und Skriptsprachen Python, HTML und PHP implementiert werden. Die Webseite soll klar gegliedert sein und dem Benutzer die wichtigsten Einstellungen bieten, die er für den Betrieb des Software-Encoders FFmpeg in Kombination mit der Capture-Karte Blackmagic DeckLink benötigt, um das Eingangssignal in einen Online-Stream umleiten zu können. Es sollen Klassen und Funktionen entwickelt werden, die einerseits die Software strukturieren und andererseits Erweiterungsmöglichkeiten bieten, um den Funktionsumfang weiter ausbauen zu können. Nach dem Vornehmen der Einstellungen soll die Webseite ein Bash-Skript generieren, das der Benutzer downloaden kann und welches ihm so zur weiteren Verfügung steht.

[1] vgl. B. Jesch, 2017 [BJeschTh]

2 Überblick über den Inhalt

Softwareentwicklung lässt sich auf Basis verschiedenster Methoden durchführen. Zum einen gibt es unter anderen das Bottom-Up- und sein Gegenstück, das Top-Down-Prinzip, aber auch das Wasserfallmodell. Beim Bottom-Up-Prinzip werden Teilprobleme abgegrenzt und gelöst und zu immer größeren Modulen zusammengesetzt, bis es zur Lösung des Gesamtproblemes kommt. [2] Im Kontrast dazu steht die Top-Down-Methode, bei der von einem Gesamtproblem mit hohem Abstraktionsgrad begonnen wird, die Aufgabe in immer kleinere Teilprobleme aufzuteilen, bis die Lösung ausreichend durchdrungen ist. [3] Der Entwicklung dieses Webinterfaces kommt das Top-Down-Prinzip am Nächsten, denn es wurde von einer großen Gesamtaufgabe immer weiter ins Detail entwickelt, um eine Lösung zu erzielen.

Nichtsdestotrotz soll zur Gliederung dieser Ausfertigung eine andere Methode genutzt werden, um die Softwareentwicklung darzustellen und eine Gliederung zu erhalten: das Wasserfallmodell. Bei diesem Prinzip wird die Software linear und organisiert nach konkreten Phasen durchgeführt und wie in einem Wasserfall vollzogen. [4] In den folgenden Kapiteln sollen so die Punkte 'Analyse', 'Anforderungen an die Software', 'Entwurf und theoretische Grundlagen', 'Implementierung' und 'Überprüfung' den Hauptteil der Arbeit bilden und die Software dem Leser darlegen.

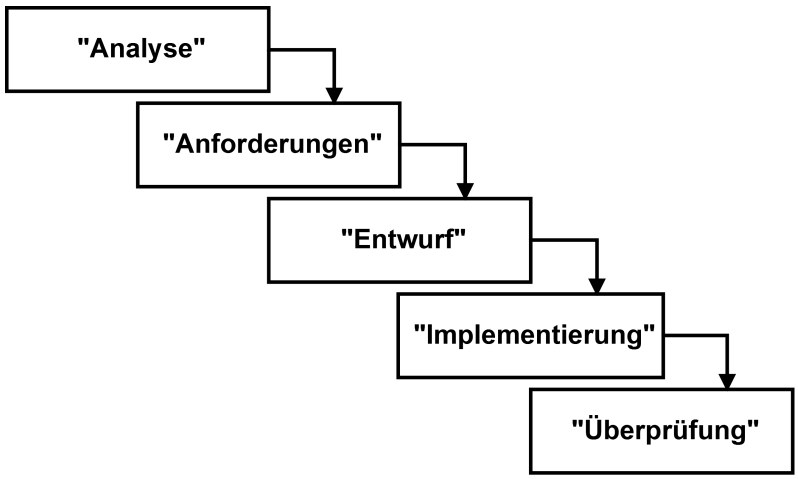

Abbildung 2.1: Schema Wasserfallmodell

[2] vgl. Springer Fachmedien Wiesbaden GmbH, 2017 [WirtLexBot]
[3] vgl. Springer Fachmedien Wiesbaden GmbH, 2017 [WirtLexTop]
[4] vgl. itemis AG, 2017 [ItAgWass]

Im Kapitel ’Analyse’ werden die Bedingungen dargestellt, die man zur Entwicklung des Webinterfaces benötigt. Dabei soll sowohl auf die Hardwarevoraussetzungen, also den PC und die Capture-Karte, als auch auf die Softwarevoraussetzungen, also unter anderem FFmpeg und das Bash-Skript, eingegangen werde.

Im Kapitel ’Anforderungen an die Software’ möchte der Bearbeiter, ausgehend von der Aufgabenstellung, auf die funktionalen und nichtfunktionalen Anforderungen eingehen, die die Software erfüllen soll.

Im Kapitel ’Entwurf und theoretische Grundlagen’ werden die Struktur der Software und die theoretischen Grundlagen für die Herangehensweise in der Entwicklung dargestellt.

Im Kapitel ’Implementierung der Klassen und Funktionen’ soll konkret auf den Quelltext eingegangen werden und die PHP-Klassen, die darin entworfenen Funktionen und die genutzten PHP-Funktionen und Sprachkonstrukte erläutert werden.

Im Kapitel ’Überprüfung’ sollen das fertige Webinterface und seine Ausgabe getestet werden und der Testaufbau erklärt werden. Des Weiteren soll überprüft werden ob alle Anforderungen erfüllt werden und eine abschließende Auswertung erfolgen.

Im Kapitel ’Fazit’ soll abschließend eine Zusammenfassung der Arbeit erfolgen. Außerdem möchte der Bearbeiter einen Ausblick geben, welche weiteren Funktionen in Zukunft integriert werden sollten, da es sich bei der Software nur um einen Prototyp handelt.

Im Anhang soll, neben dem kompletten Quelltext, eine Anleitung mitgegeben werden, mit der man ein Linux-System mit den Komponenten Ubuntu, Apache Webserver, PHP, Blackmagic DeckLink, FFmpeg und dem Wrapper aufbauen kann.

3 Analyse

3.1 Hardwarevoraussetzungen

3.1.1 Systemparameter und PCIe

Die Systemvoraussetzungen bzw. die Maschine, die zum Einsatz kommt, soll hier noch einmal kurz genannt werden. Beim Prozessor handelt es sich um eine Intel(R) Core(TM) i7-4790 CPU mit einer Taktfrequenz von 3,60 GHz. Der Arbeitsspeicher beträgt 8 GB, der PC verfügt über einen Gigabit Ethernet Controller und eine Festplatte mit einer Speichergröße von 256 GB. Genutzt wird ein Motherboard der Firma MSI mit der Bezeichnung MS-7817.
Gerade eine leistungsfähige CPU (mindestens Intel i5, besser Intel i7) und ein ausreichend großer Arbeitsspeicher (>= 8GB) sind notwendig, um FFmpeg und die Capture-Karte im Zusammenspiel zuverlässig und hochperformant betreiben zu können und ein gutes Ergebnis in der Aufzeichnung und Übertragung des Videosignales zu erzielen.

PCIe (PCI Express) ist die Abkürzung für „Peripheral Component Interconnect Express“. PCIe kann als Nachfolger von PCI, PCI-X und AGP angesehen werden und wurde im Jahr 2004 eingeführt. [5] In Computersystemen werden, wie auch in diesem Diplomprojekt, Erweiterungskarten in der Computertechnik verwendet, für die PCIe eine leistungsfähige interne Schnittstelle darstellt. [6]

3.1.2 Capture-Karte Blackmagic DeckLink

Im Zuge der Bearbeitung wird eine ’Blackmagic Decklink Duo 2 SDI’ Capture- Karte verwendet. Die australische Firma Blackmagic (oder vollständig: Blackmagicdesign) ist ein Hersteller von professioneller Video-Technologie[7] , dazu zählen unter anderem Profi- Kameras, Audio-/Video-Schnittplätze, Capture- und Playback- Karten und die dazugehörige Software.[8] An der Hochschule Mittweida werden fast ausschließlich Videoschnittkarten von Blackmagic verwendet, sodass die Verwendung im Projekt naheliegend war.

[5] vgl. Wikipedia, die freie Enzyklopädie, 2017 [WikPci]
[6] vgl. Elektronik-Kompendium.de, 2017 [ElKoPcie]
[7] vgl. Blackmagic Design Pty. Ltd., 2017 [BlDesAbo]
[8] vgl. Blackmagic Design Pty. Ltd., 2017 [BlDesPro]

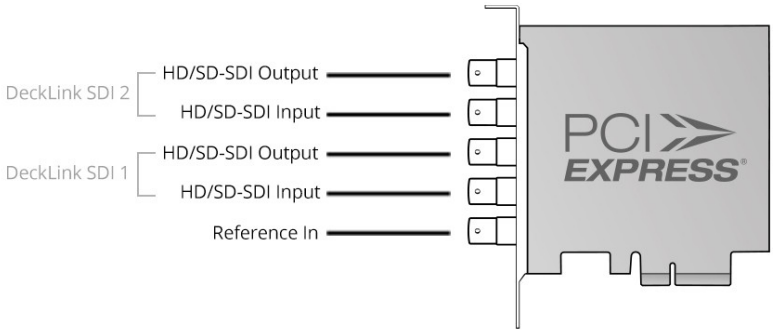

Abbildung 3.1: Blackmagic DeckLink Duo 2 SDI, vgl. picturetools.de, 2017 [PicBlaDec]

Die Blackmagic Decklink SDI Capture- Karte bietet 5 SDI- Anschlüsse. SDI steht dabei für Serial Digital Interface und bezeichnet eine serielle, digitale Schnittstelle. Dieser Anschlusstyp wird hauptsächlich in der professionellen Videotechnik verwendet, um Videodateien unverschlüsselt und unkomprimiert mittels Koaxialkabel oder Lichtwellenleiter übertragen zu können.[9] Die verwendete Erweiterungskarte besitzt 2 SDI- Ein- und Ausgänge für Audio-/Videosignale und einen SDI- Anschluss für ein externes Steuersignal.

3.2 Softwarevoraussetzungen

3.2.1 FFmpeg

Der verwendete Software-Encoder, für den das Webinterface entwickelt werden soll, ist FFmpeg. „FFmpeg ist das führende Multimedia Framework, das in der Lage ist zu dekodieren, zu kodieren, zu transkodieren, zu multiplexen, zu demultiplexen, zu streamen, zu filtern und nahezu alles abzuspielen, was Menschen und Maschinen erzeugt haben. Es unterstützt die außergewöhnlichsten alten Formate bis hin zu denen der neuesten Technologie. Egal ob sie von einem Standardisierungs-Komitee, einer Gemeinschaft oder einer Firma entwickelt wurden. Es ist außerdem höchst variabel: FFmpeg kompiliert, betreibt und übertragt unsere Test- Infrastruktur FATE zwischen Linux, Mac OS X, Microsoft Windows, BSDs, Solaris usw. innerhalb einer großen Anzahl von Entwicklungsumgebungen, Rechnerarchitekturen und Konfigurationen.“[10]

[9] vgl. Wikipedia, die freie Enzyklpädie, 2017 [WikSerInt]
[10] vgl. FFmpeg project, 2017 [FFAbo]

Abbildung 3.2: FFmpeg-Logo, vgl. FFmpeg project, 2017 [FFPic]

Dabei steht der Begriff FFmpeg als Synonym für verschiedene Werkzeuge und Bibliotheken für Entwickler. Diese Werkzeuge und einige Beispiele für die Bibliotheken sollen nachfolgend im Sinne der Vollständigkeit aufgelistet werden.

Werkzeuge [11]:

ffmpeg - das gleichnamige Kommandozeilenprogramm ffmpeg zur Konvertierung von Multimedia-Dateien zwischen Formaten

ffserver - ein Multimedia- Streaming-Server für die Echtzeitübertragung

ffplay - ein einfacher Media-Player basierend auf SDL

ffprobe - ein einfaches Multimedia Stream- Auswertungsprogramm

Bibliotheken [12]:

libavutil - ist eine Bibliothek, die Funktionen zur Erleichterung der Programmierungen beinhaltet, inklusive Zufallszahlengeneratoren, Datenstrukturen, mathematische Routinen, Kerndienstprogramme, und vieles mehr

libavcodec - ist eine Bibliothek die Decoder und Encoder für Audio/Video Codecs enthält

libavformat - ist eine Bibliothek, die Demultiplexer und Multiplexer für Multimedia Containerformate bereitstellt

[11] vgl. FFmpeg project, 2017 [FFTool]
[12] vgl. FFmpeg project, 2017 [FFLib]

3.2.2 Ubuntu und Apache

Das verwendete Betriebssystem ist Ubuntu: eine Linux-Distribution, die offiziell im Jahr 2004 veröffentlicht wurde. [13] Das Betriebssystem ist kostenlos und für die Entwicklung des Webinterfaces wird Version 16.04 LTS verwendet. Die Zusatzbezeichnung 'LTS' steht für 'Long Term Support' und bedeutet, das diese Version bis 5 Jahre nach ihrer Veröffentlichung mit Fehlerkorrekturen und Aktualisierungen gepflegt wird. [14] Um eine Website zu betreiben und über das Netzwerk auf sie zuzugreifen, benötigt man einen Webserver. Der verwendete Webserver ist ein Apache in der Version 2.4.18. Zusätzlich wird zum Aufsetzen des Quelltextes die integrierte Entwicklungsumgebung PhpStorm genutzt, die bei der Entwicklung der Klassen, Funktionen und des 'wrapper' sehr komfortabel ist.

3.2.3 Python, HTML, PHP und Bash - Die Findung der Sprache

In diesem Abschnitt soll nicht nur auf die verwendeten Skript- bzw. Programmiersprachen eingegangen werden, sondern auch erläutert werden, warum gerade diese Kombination der Sprachen gewählt wurde. Am Anfang war es klar, das die Software nicht nur mit der Verwendung einer Sprache funktionieren würde, aber welche Kombination der Sprachen in der Implementierung genutzt wird, wurde erst innerhalb der Entwicklung festgelegt.

Es wurde festgelegt, das die grafische Oberfläche mit der Skriptsprache HTML realisiert wird, weil diese die Struktur, Formulare und Formen bereitstellt, die für eine Webseite benötigt werden. Für die Kommunikation und den Datenaustausch zwischen Webinterface und dem 'Server' kommt die Skriptsprache PHP zum Einsatz. Sie stellt die Funktionen bereit, um die Ein- und Ausgaben und die Befehle des Webinterfaces und des Benutzers auszuführen. Diese wird vorerst als CGI-Version ('Common Gateway Interface') verwendet und nicht als Modul vom Apache Webserver, sodass ein spezielles Verzeichnis mit dem Namen '/cgi-bin' als Speicherort für die Software genutzt wird. Die PHP-Skripte werden dabei nicht als klassische '.php'-Dateien angelegt sondern indirekt über Bash-Skripte ausgegeben. In diesen Skripten werden die „PHP-Variablen“ außerdem in Bash-Variablen umgewandelt. Die Installation von PHP befindet sich auf demselben physischen Gerät wie FFmpeg und die Capture-Karte. Auf dieser 'Serverseite' sollen die Eingaben, die über die HTTP-Methode 'POST' übertragen werden, nun umgesetzt werden. Hier kommt die Programmiersprache Python zum Einsatz, in der ein Skript geschrieben wird, dass die Variablen aus dem PHP/Bash-Skript abfängt und verarbeitet. Die Werte, die als Kommandozeileneingabe verwendet werden, können nun über Python-Funktionen ausgeführt werden. Die Ausgabe wird wiederum vom Skript benutzt und zurück an das PHP/Bash-Skript übergeben. Die Werte, die für das finale

[13] vgl. Canonical Ltd., 2017 [UbStor]
[14] vgl. ubuntuusers.de, 2017 [UbUsLts]

Bash-Skript für den Benutzer übermittelt werden, können über Python-Funktionen zu einem String zusammengefasst und in besagtes Skript geschrieben werden, welches auch über Python erzeugt wird.

Diese Kombination aus HTML, PHP, Python und Bash wurde so nicht realisiert. Ein Nachteil dieser Realisierung stellt zum Beispiel die Tatsache dar, das das Python-Skript dauerhaft ausgeführt werden muss, um die Variablen von PHP auffangen zu können. Sollte es also aufgrund eines Fehlers nicht ausgeführt werden, kann man das Webinterface nicht benutzen. Weiterhin würde diese Variante eine Kombination von 4 verschiedenen Sprachen, das synchronisieren ihrer spezifischen Eigenheiten und das Variablenhandling zwischen allen bedeuten, was zwangsläufig eine gewisse Fehleranfälligkeit zur Folge hat.

Es wurde eine Kombination ausschließlich aus HTML und PHP gewählt, wobei die Interakion mit der Kommandozeile mit PHP-Funktionen realisiert wird. Des Weitern wird PHP als Modul von Apache verwendet und es wird das allgemein übliche Verzeichnis des Webservers verwendet: '/var/www/html'. Die Ein- und Ausgabe und die Befehle des Webinterfaces werden direkt in der Programmiersprache PHP und über Klassen, Funktionen und spezifische PHP-Funktionen realisiert. Die Fehleranfälligkeit wird somit verringert, einerseits durch die Reduzierung auf eine Haupt- Programmiersprache und weil andererseits zum Beispiel keine Variablen innerhalb von Programmiersprachen getauscht werden müssen.

3.2.4 Shell-Umleitungen und Reguläre Ausdrücke

FFmpeg ist eine Kommandozeilenanwendung und lässt sich im Allgemeinen nur über die Konsole steuern. Für eine externe Steuerung müssen Eingabe und Ausgabe also auf speziellem Wege erfolgen. Die Eingabe wird über die PHP-Funktion 'exec' realisiert. Diese Funktion führt das Kommando direkt auf der Maschine aus. Die Ausgabe jedoch erfolgt standardmäßig direkt in der Konsole und wird somit nicht „mitgeschnitten“. Eine Lösung für dieses Problem stellt die sogenannte Shell-Umleitung dar. Hinter dem Befehl wird ein Zusatz hinzugefügt in Form von „2>&1“, der bedeutet, das sowohl der Standardausgabekanal ('1') als auch der Standardfehlerkanal ('2') der Konsole umgeleitet werden. [15] Bei der Funktion 'exec' wird die Ausgabe in eine Variable geschrieben. [16]

Ist die Ausgabe in einer Variablen gespeichert, ist ihr Inhalt oft noch nicht der, den man für die Weiterverwendung benötigt. Für diesen Zweck gibt es sogenannte „reguläre Ausdrücke“, mit denen man einen Text nach bestimmten Mustern durchsuchen kann.[17] Ein Beispiel dafür ist die Abfrage der Version von FFmpeg, die man durch das Komman-

[15] vgl. ubuntuusers, 2017 [UbUsUml]
[16] vgl. The PHP Group, 2017 [PhpExec]
[17] vgl. SELFHTML-Wiki, 2017 [SelHtRegAus]

do ’ffmpeg -version’ erhält. Die Versionsbezeichnung ist jedoch nur ein Segment der Ausgabe, die wie folgt aussieht:

„ffmpeg version N-83030-gcd09e3b Copyright (c) 2000-2017 the FFmpeg developers“.

Mit Hilfe der PHP-Funktion ’preg_match’[18] und dem regulären Ausdruck

’N-[0-9]5-[a-z|0-9]8’

lässt sich der Inhalt filtern. Dieser Ausdruck soll nun aufgeteilt und genauer erläutert werden.

Tabelle 3.1: Erläuterung regulärer Ausdruck

Befehl	Wirkung
N-	filtert nach einem Segment, das beginnt: mit einem großen N und darauf folgendem Bindestrich
[0-9]5-	Segment geht weiter: mit einem 5 Zeichen langen Abschnitt, bestehend aus Zahlen von 0 bis 9, und abermals mit darauffolgendem Bindestrich
[a-z|0-9]8	Segment geht weiter: mit einem 8 Zeichen langen Abschnitt, bestehend aus Kleinbuchstaben und Zahlen

Am Ende erhält man durch Umleitung und dem regulären Audsruck die neue Variable mit der genauen Versionsbezeichnung als Inhalt:

’N-83030-gcd09e3b’.

[18] vgl. The PHP Group, 2017 [PhpPreg]

4 Anforderungen an die Software

Die Software soll spezifische Eigenschaften besitzen, genauer gesagt bestimmte Anforderungen erfüllen. Bei der Softwareentwicklung unterscheidet man in 'funktionale Anforderungen' und 'nichtfunktionale Anforderungen'. Im Allgemeinen sind diese strukturiert und erkennbar an einem genauen Identifikator, einer Beschreibung, einer Problembeschreibung, einer Quelle und einem Abnahmekriterium.[19]
Für die hier vorliegenden Anforderungen soll diese Struktur als Vorlage dienen, indem die Punkte Identifikator (Nr.), Beschreibung und Problembeschreibung übernommen werden. Die wichtigsten funktionalen Anforderungen ('FA_x') und nichtfunktionalen Anforderungen ('NA_x') sind in den nachfolgenden Tabellen aufgelistet.

4.1 Funktionale Anforderungen

Tabelle 4.1: Funktionale Anforderungen

Nr.	Beschreibung	Problembeschreibung
FA_1	Es sollen Textboxen, Auswahllisten und Checkboxen für benutzerspezifische Eingaben für den Software-Encoder FFmpeg zur Verfügung stehen.	Es müssen HTML-Form-Elemente mit Formularen wie Textboxen, Auswahllisten, und Checkboxen implementiert werden.
FA_2	Die angebotenen Einstellungen sollen unter anderem Bildgröße, Bitrate, Quelle, Ziel, Anzahl der Streams beinhalten.	Es muss entschieden werden, welche Einstellung auf welcher Weise (HTML-Formular) angeboten wird und welche Werte vorgegeben werden.
FA_3	Das Webinterface soll die Einstellungen anbieten und auf einer weiteren Webseite wird das Ergebnis angezeigt.	Es müssen zwei voneindander abhängige Webseiten implementiert werden.
FA_4	Der Benutzer soll als Endresultat auf dieser zweiten Webseite ein Bash-Skript herunterladen können, mit dem er den Software-Encoder betreiben kann.	Es müssen Funktionen entwickelt werden, die ein Bash-Skript aus den Einstellungen erzeugen.

[19] vgl. Wikipedia, die freie Enzyklopädie, 2017 [WikAnSof]

4.2 Nichtfunktionale Anforderungen

Tabelle 4.2: Nichtfunktionale Anforderungen

Nr.	Beschreibung	Problembeschreibung
NA_1	Die Software soll in den Programmier- bzw. Skriptsprachen Python/PHP/HTML/Bash implementiert werden.	Es müssen Schnittstellen für die Programmier- und Skriptsprachen gefunden werden.
NA_2	Die Software soll in Klassen und Funktionen programmiert werden, um die Funktionen zu strukturieren und individuell nutzen zu können.	Für wiederkehrende Aufgaben müssen Funktionen implementiert werden, die in eine Struktur aus Klassen eingearbeitet werden.
NA_3	Das Webinterface muss mit der Kommandozeile des Linux-Systems kommunizieren und die Ausgabe auffangen.	Der Zugriff auf die Kommandozeile mit Hilfe der oben genannten Programmier- und Skriptsprachen und das Abfangen der Ausgabe müssen implementiert werden.
NA_4	Die Informationsübertragung des Benutzers mit dem Webinterface wird mit der HTTP-Methode 'POST' ausgeführt.	Das Webinterface muss so implementiert werden, dass es die Daten per 'POST' versendet, und die zweite Webseite muss die Daten per 'Post' empfangen.
NA_5	Das Webinterface soll klar gegliedert und die Einstellungen selbsterklärend sein.	Beide Webseiten sollen mit Hilfe von HTML- Überschriften und HTML- Tabellen implementiert und durch kurze Hinweise und passende Einheiten selbsterklärend werden.

5 Entwurf und theoretische Grundlagen

5.1 Die Rolle der Software als Wrapper

Die entwickelte Software übernimmt im Gesamtprojekt eine Position ein, die in der Fachsprache als 'wrapper' bezeichnet wird. Die allgemeinen Übersetzungen für das englische Wort 'wrapper' lauten unter anderem „Verpackung", „Hülle" oder „Umschlag".[20] Im weiteren Sinne gelten Sie auch für die Software bzw. das Interface, dass entwickelt wurde: es ist eine Verpackung für bestimmte, benutzerspezifische Funktionalitäten, die sich um die bereits existierende Software FFmpeg hüllen und ist dabei nur ein Umschlag, den man mit weiteren Funktionalitäten füllen kann. Im engeren Sinne stellt der erarbeitete 'Wrapper' auch eine Schnittstelle bzw. ein Adapter zwischen dem Benutzer und dem Multimediaframework dar.[21] Die Kommandozeilen- Ein- und Ausgabe, die eigentlich zur Steuerung von FFmpeg benötigt wird, wird mittels PHP und HTML auf eine grafische Oberfläche übertragen und für den Nutzer aufbereitet.

Abbildung 5.1: Schema 'Wrapper'

5.2 Die Gliederung der Software

Wie im Kapitel Anforderungen erwähnt, soll das Webinterface diverse Einstellungen für den Benutzer bereitstellen. Um die Arbeit zu strukturieren soll als erster Schritt die Gesamtsoftware gegliedert und in Unterkategorien geteilt werden. Diese Kategorien werden als Input, Filter und Output benannt. Die Kategorie Input steht hierbei im Allgemeinen für die Quelle des Video-/Audiosignales, dass der Benutzer zur Weiterverarbeitung nutzen möchte. Der Komplex Filter steht für alle Einstellungen, die der Benutzer vornehmen kann, um die Streams seinen Bedürfnissen anzupassen. Die Kategorie Output soll Einstellungen zum Zielort des Videosignales bieten. Im eigentlichen Webinterface werden alle Optionen gebündelt und dem Benutzer angezeigt. Der Punkt Downloadseite steht für die Zusammenfassung aller Einstellungen und ihrer Verdichtung zum Bash-Skript, dass am Ende zur Verfügung stehen soll.

[20] vgl. Ing. Paul Hemetsberger, 2017 [DictWrap]
[21] vgl. DATACOM Buchverlag GmbH, 2017 [ItWissWrap]

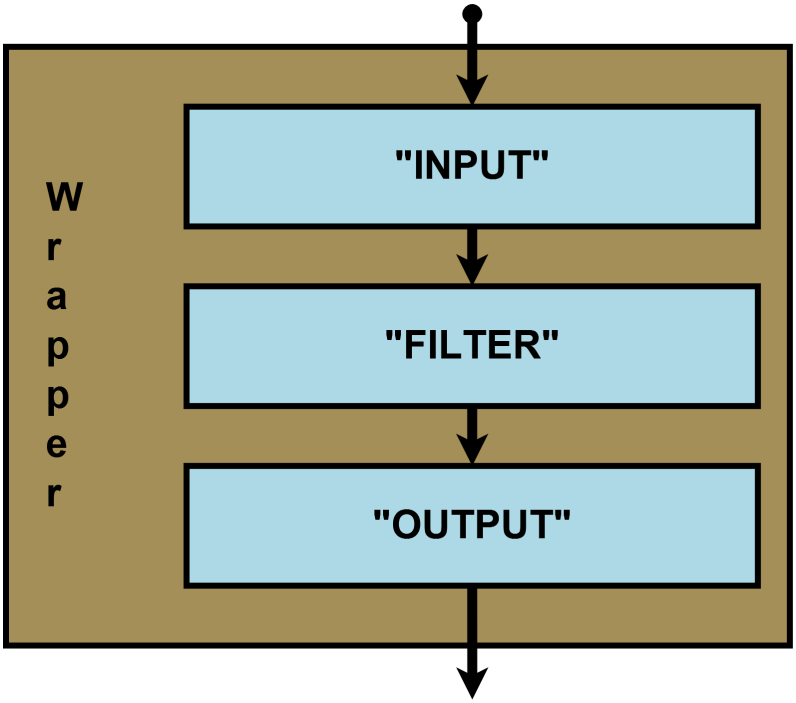

Abbildung 5.2: Gliederung des 'Wrapper'

5.3 Die Aufgaben von Input, Filter und Output

Auf die festgelegten Kategorien müssen nun alle Aufgaben aufgeteilt werden, die sich aus den Anforderungen ableiten lassen. Die Kategorie Input beinhaltet alle Aspekte, die mit der Signalquelle für den Software-Encoder im Zusammenhang stehen. Diese Signalquelle kann unterschiedlicher Art sein, zum Beispiel eine Audio- oder Video-Datei, eine Grafik oder wie im vorliegenden Fall, das Signal einer Blackmagic DeckLink Capture-Karte. Für den Prototypen muss also eine PHP-Klasse für Input entwickelt werden, die Funktionen enthält, die den Benutzer die Capture-Karte und ihre Optionen, vorzugsweise aus einer HTML-Auswahlliste, als Eingangsquelle wählen lässt. Diese Funktion muss einerseits die Aufnahme-Optionen der Karte auslesen und sie andererseits dem Benutzer in aufbereiteter Form zurückliefern können. Hier kommen die bereits erwähnten Themen 'Shell-Umleitungen und Reguläre Ausdrücke' zum Tragen. Es muss also zuerst der Befehl für das Auslesen der Aufnahme-Optionen ausgeführt werden, dessen Ausgabe dann in eine Variable umgeleitet wird. Diese Variable wird dann mit Hilfe eines speziellen regulären Ausdrucks gefiltert, um den Benutzer und der Software nur die Informationen auszugeben, die zur Weiterverwendung benötigt werden. An dieser Stelle sei auch erwähnt, dass allein schon für einen anderen Typ der Capture-Karte ein anderer regulärer Ausdruck entworfen werden muss. Für die Signalquellen Audio-/Video-Datei und Grafik müsste ein Funktion entwickelt werden, die den Benutzer auf das Dateisystem bzw. das Netzwerk zugreifen lässt, um eine Datei auswählen zu können deren Dateipfad dann zur Weiterverwendung in der Software genutzt wird.

Die Kategorie Filter steht für die eigentliche Konvertierung des Eingangs- zum Ausgangssignal, die in FFmpeg vorgenommen wird. Hier sollen die Funktionen untergebracht werden, die man zum 'Mixen' bzw. 'Splitten' der/des Eingangssignale(s) zu dem/-den Ausgangssignal(en) benötigt. Hier sollen vorerst Funktionen implementiert werden, die dem Benutzer Einstellungen für die Auflösung, für die Pufferate, für den Codec und für andere, speziell für Video-Signale relevante, Kenngrößen zur Verfügung stellen. Hier müssen vorgefertigte Listen bzw. Textboxen benutzt werden, damit die Eingaben gemacht werden können. Um die Einstellungen komfortabel im Quelltext anpassen zu können, müssen Konstanten definiert werden, aus denen die Auswahllisten ausgelesen werden. Bei den Textboxen, in denen der Benutzer händisch seine Werte eingibt, müssen Abfragen eingebaut werden, ob die Eingabe sinnvoll ist bzw. ob es sich überhaupt um eine Zahl handelt. Eine weitere Funktionen für diese Kategorie sollte die Einstellung sein, ob man einen Mixer oder einen Splitter in FFmpeg verwenden möchte, um Eingangssignale zusammenzuführen oder sie aufzuspalten. Diese Einstellung soll im Prototyp noch nicht verwendet werden, weil hier FFmpeg vorerst standardmäßig als Splitter verwendet wird. Für die Nutzung anderer Eingangssignale müssen natürlich auch weitere Einstellungen für den Filter vorgesehen werden. So wären Auswahlmöglichkeiten zum Audio-Codec, zur Grafik-Norm (.png, .jpeg, bmp. usw.) oder weitere Einstellungen für das Video-Signal denkbar.

Die Kategorie Output soll die Art des Ausgangssignales beinhalten. Im Prototyp ist es ein Stream, der bereits als statische Größe im endgültigen Bash-Skript vorgesehen ist. Hier sollen dem Benutzer Optionen zur Verfügung gestellt werden, wohin das Ausgangssignal geht. Neben der Auswahl, es in einen Stream umzuleiten, soll auch hier wieder die Möglichkeit gegeben werden, das Dateisystem bzw. Netzwerk zu durchsuchen, um Speicherort und Dateinamen festzulegen, die dann von der Software als Dateipfad weiterverwendet werden.

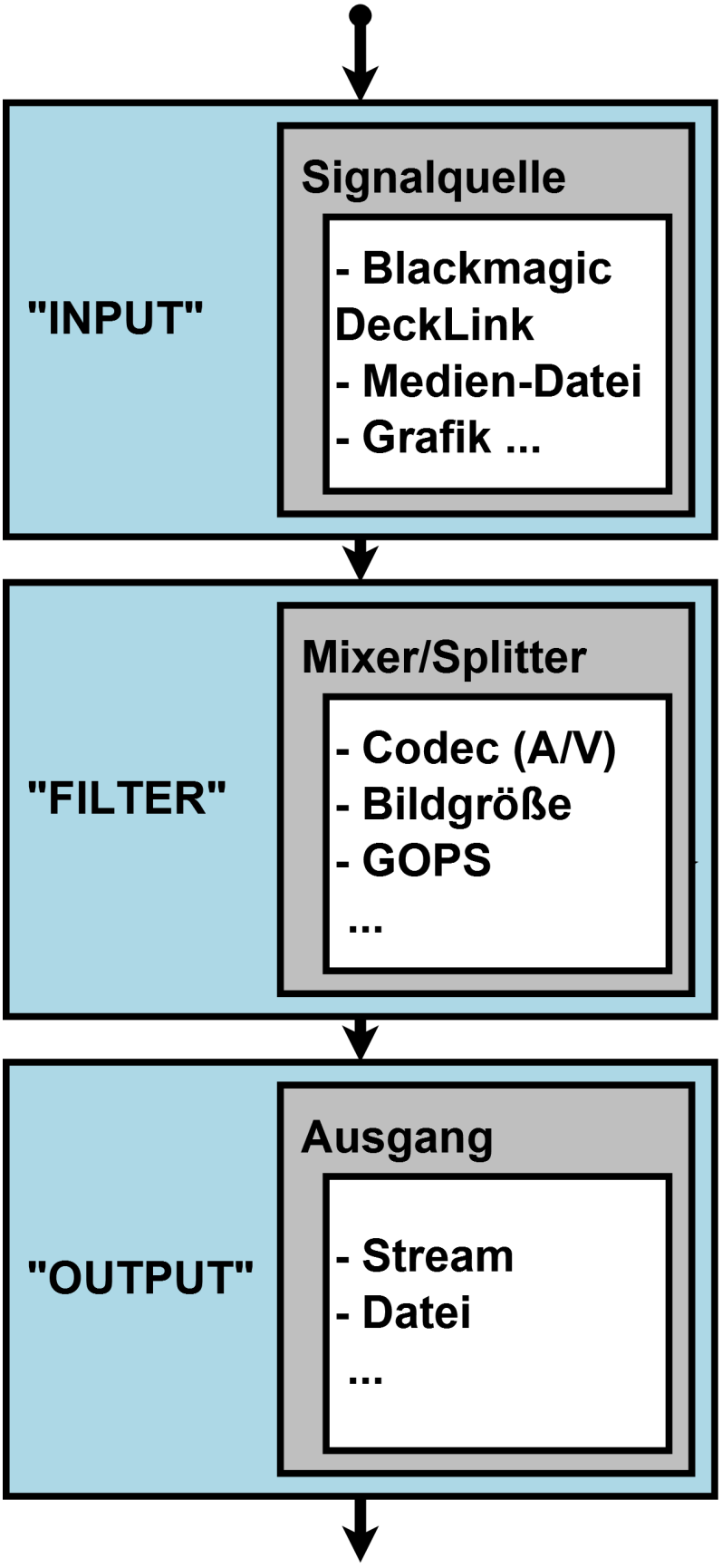

Abbildung 5.3: Die Aufgaben von Input, Filter und Output

5.4 Der Aufbau des Webinterfaces

Die vorher eingeführten Kategorien sollen auch auf der grafischen Oberfläche des Webinterfaces wiedererkennbar sein. Für den Prototypen sollen die Standardeinstellungen der HTML-Formen bezüglich Farbe und Formatierung beibehalten werden. Zur Strukturierung des Abschnittes Filter soll eine HTML-Tabelle genutzt werden, um die Seite besser zu strukturieren und gleiche Einstellungsmöglichkeiten für verschiedene Ausgänge besser unterscheiden zu können.

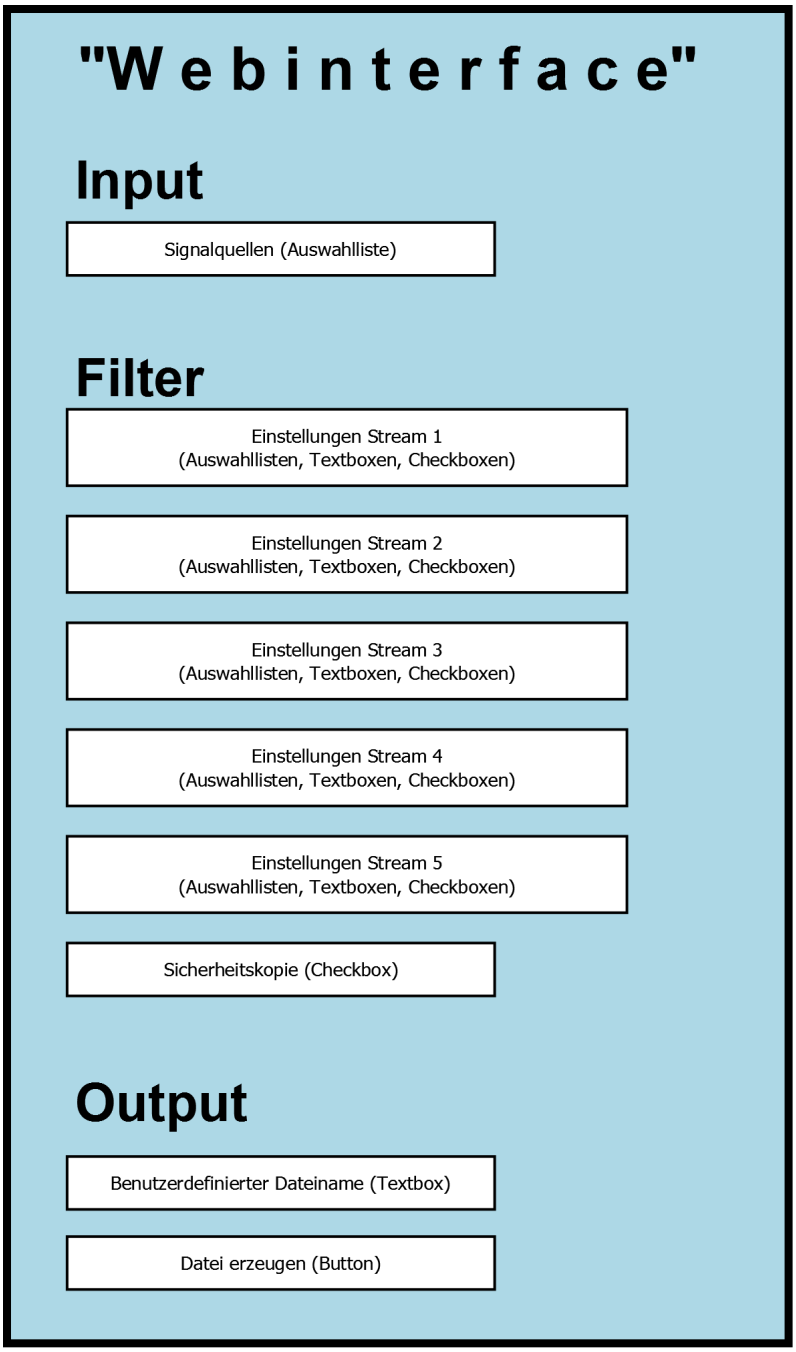

Abbildung 5.4: Aufbau des Webinterfaces

5.5 Das Zusammenspiel von Webinterface und Downloadseite

Wie bereits erwähnt werden alle Eingaben im Webinterface vorgenommen und alle Funktionen in einer HTML-Form integriert. Des Weiteren findet der Benutzer dort einen Button durch den das Bash-Skript erzeugt wird. Dies wird durch eine weitere PHP-Datei realisiert: die Downloadseite. Diese setzt die vordefinierten Variablen und die per HTTP-Methode 'POST' übermittelten Werte zu einem String zuammen. Die festen Bestandteile des Bash-Skriptes werden vom Bash-Skript 'live.sh' übernommen. Dieses wird im nächsten Kapitel näher erläutert. Die übermittelten Werte werden an den entsprechenden Stellen eingefügt und alle Variablen werden in der'creator.php' zu einem vollständigen String zusammengefügt und gleichzeitig in eine Datei geschrieben. Außerdem wird der Inhalt des Skriptes hier geordnet und strukturiert auf einer Seite angezeigt. Dies gibt dem Benutzer die Möglichkeit, seine Eingaben noch einmal zu überprüfen und bei Bedarf zum Webinterface zu wechseln und die Einstellungen zu ändern. Durch das Klicken auf den Download-Button wird vom Browser ein Dialogfenster geöffnet, das es dem Benutzer erlaubt die Datei zu speichern oder eventuell gleich in einem Editor zu öffnen.

5.6 Das Bash-Skript Live.sh

Ein Bash-Skript ist eine Abfolge von langen Kommandozeilenbefehlen für ein Linux-Betriebssystem zusammengefasst in einer Datei, sodass diese zusammengehörigen Befehle gleichzeitig ausgeführt werden können, nicht einzeln in die Konsole eingegeben werden müssen und besser konfigurierbar sind. Vorlage für das Skript, dass von der Downloadseite ausgegeben wird ist die Datei 'live.sh'.

Dieses Skript besteht aus einer Abfolge von FFmpeg- Befehlen, diese lassen sich in die Bereiche Input, Filter und Output gliedern. Genauer gesagt heißt das:
am Anfang steht eine Variable namens $OUTPUT, diese deklariert nur den Namen der Sicherheitskopie und wird am Ende noch einmal genauer erklärt. Danach folgt die erste FFmpeg-Zeile, in der die Signalquelle festgelegt wird, also der Input. Es ist eine 'Decklink SDI' bei der der Eingang '(2)' mit der Einstellung '@9', diese steht für eine spezielle Auflösung und Framerate, genutzt wird. Darauf folgen im Filter unter anderem 'split=4[v1][v2][v3][v4]', also eine Aufteilung des Videosignals auf 4 Ausgänge, und 'asplit=4[a1][a2][a3][a4]âĂŸ, die Aufteilung des Audiosignales auf 4 Ausgänge.

Danach beginnt der Abschnitt der Filter für die Outputs, bei denen die Signale für Audio sowie Video mit der Nummer 1 bis 3 wieder gleichnamig zusammengeführt werden. Die Outputs für v1/a1, v2/a2 und v3/a3 sind gleich aufgebaut und unterscheiden sich nur in den Werten. Exemplarisch soll hier der Output v1/a1 erklärt werden.

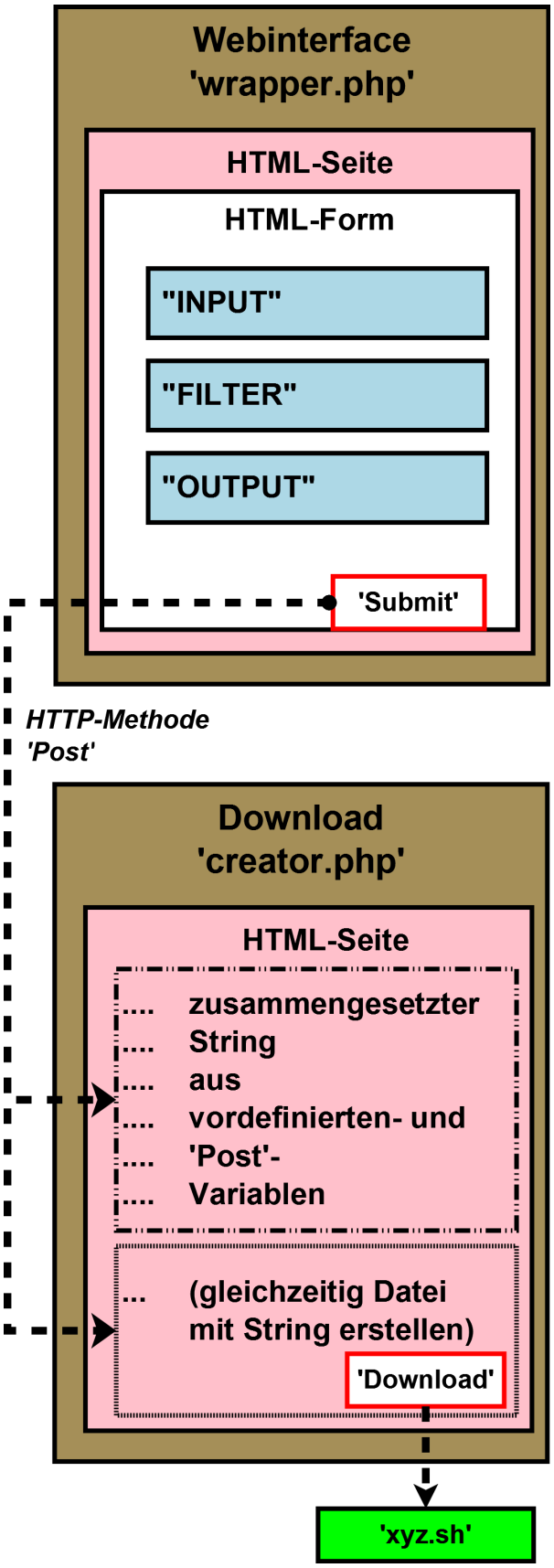

Abbildung 5.5: Wrapper und Downloadseite

```
#!/bin/sh
#
OUTPUT=/srv/encoder/$(date +%Y%m%d_%H.%M.%S).mkv
ffmpeg -nostats -loglevel 0 -f decklink -i 'DeckLink SDI (2)@9' -r 25 -threads 0 \
    -filter_complex "[0:1]yadif,split=4[v1][v2][v3][v4];[0:0]asplit=4[a1][a2][a3][a4]" \
    -map "[v1]" -map "[a1]" -s 1280x720 \
        -pix_fmt yuv420p -c:v libx264 -b:v 4500k -bufsize 9000k -g 100 -preset:v faster -c:a aac -ar 48000 -b:a 224k \
        -f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/liveevent1?adbe-live-event=liveevent" \
    -map "[v2]" -map "[a2]" -s 1024x576 \
        -pix_fmt yuv420p -c:v libx264 -b:v 1500k -bufsize 3000k -g 100 -preset:v faster -c:a aac -ar 48000 -b:a 192k \
        -f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/liveevent2?adbe-live-event=liveevent" \
    -map "[v3]" -map "[a3]" -s 640x360 \
        -pix_fmt yuv420p -c:v libx264 -b:v 450k -bufsize 900k -g 100 -preset:v faster -c:a aac -ar 48000 -b:a 128k \
        -f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/liveevent3?adbe-live-event=liveevent" \
    -map "[v4]" -map  "[a4]" -s 640x360 \
        -pix_fmt yuv420p -c:v libx264 -preset:v faster -c:a aac -ar 48000 -b:a 128k $OUTPUT
```

Abbildung 5.6: Das Bash-Skript ’live.sh’, vgl. [BJeschLive]

Zunächst wird mit’-s’ („solution“) die Auflösung festgelegt, danach folgen die Kurzformen für das Pixelformat, ’-pix_fmt’, und für den verwendeten Codec für Video, ’-c:v’. Nachfolgend schließen sich die ausgewählte Bitrate für das Videosignal, ’-b:v’, die Pufferrate, ’-bufsize’, die Zahl der „Group of Pictures“, ’-g100’ und die Video-Voreinstellung, ’-preset:v faster’, an. Auch für das Audiosignal werden Werte für den Codec, für die Abtastrate und die Bitrate festgelegt: ’-c:aaac’ für den Codec, ’-ar 48000’ für die Abtastfrequenz und ’b:a 224k’ für die Audio-Bitrate.

Die letzte Zeile legt das endgültige Ziel des Signales, im Falle des Projektes ist es ein Stream, fest. Die Kurzbezeichnung von FFmpeg ist hier ’-f flv’.

Das Signal v4/a4 geht nicht in einen Stream, sondern wird mit kleinerer Auflösung und weniger Einstellungen in eine Sicherheitskopie, also eine Videodatei, umgeleitet. Hier ist das Ziel durch die am Anfang erwähnte Variable ’$OUTPUT’ und den darin enthaltenen Pfad bzw. den darin enthaltenen Dateinamen festgelegt.

Diese Skript kann über einen sogenannten Daemon ausgeführt werden. Diese Programme laufen im Hintergrund und werden in Unix-Betriebssystemen genutzt, um bestimmte Dienste bereitzustellen. [22] Dazu gehört zum Beispiel das Starten und Beenden von Bash-Skripts, um sie benutzen und kontrollieren zu können.

[22] vgl. ubuntuusers.de, 2017 [UbUsDae]

6 Implementierung der Klassen und Funktionen

Die Funktionalitäten der Software lassen sich, wie im Abschnitt Entwurf bereits erwähnt, in die Kategorien Input, Filter und Output gliedern. Für jedes Element wurde eine PHP-Klasse erstellt, die die speziellen Funktionen und Variablen zur Erfüllung der Anforderungen enthält. Des Weiteren sollen an dieser Stelle das konkrete Webinterface und die Webseite für den Download genannt werden, die zusammen mit den Klassen in folgender Tabelle zusammengefasst sind.

Tabelle 6.1: Die Klassen und Funktionen der Software

Kategorie	Name	Funktionen
Input	'create_decklink_devices.php' (PHP-Klasse)	private function getlines, public function get_version public function get_decklink_devices
Filter	'create_ffmpeg_filter.php' (PHP-Klasse)	public function filter_res public function filter_bit_audio public function filter_gops public function filter_buf_video public function filter_bit_video public function filter_codec_video
Output	'create_ffmpeg_profil.php' (PHP-Klasse)	
Webinterface	'wrapper.php'	
Downloadseite	'creator.php'	

Die folgenden Unterpunkte beschreiben die Funktionen der einzelnen Klassen bzw. ihre Anwendung in den Kategorien Webinterface und Download. Zu jeder Klasse werden die vordefinierten Konstanten bzw. Variablen, die einzelnen Funktionen und ihre eventuelles Verhältnis zueinander erklärt.

6.1 Input - die PHP-Klasse 'create_decklink_devices'

Input steht dabei auch für die Quelle, bisher nur die Auswahl des Eingangs der DeckLink Blackmagic Karte, und bietet dem Benutzer die Auswahl des Ausgangspunktes für sein Audio-/Video- Signal.

6.1.1 Konstantendefinition

Listing 6.1: Konstantendefinition

```
const FFMPEGBIN = '/home/mtm-tvs/bin/ffmpeg';
# binary

const DEVICEPATTERN = '/\[decklink @ 0x[0-9|a-f]{7}\]/';
# Pattern from ffmpeg -f decklink -list_devices 1 -i dummy

const VERSIONPATTERN = '/N-[0-9]{5}-[a-z|0-9]{8}/';
# returns version string from ffmpeg -version

const DEVICEFORMATPATTERN =
'/\[decklink @ 0x[0-9|a-f]{7}\](.){2}[0-9]{1,2}/';
# returns capture formats
```

Am Anfang der PHP- Klasse werden 4 Konstanten festgelegt. Die erste dient hierbei als Platzhalter und beinhaltet den Dateipfad von FFmpeg auf dem Versuchsrechner, der für die Befehlseingabe in die Kommandozeile benötigt wird. Die drei folgenden Konstanten sind reguläre Ausdrücke. Diese werden in den Funktionen als Muster genutzt, für die Filterung der Ausgaben aus der Kommandozeile, die zurück an das Webinterface gelangen sollen. Sie werden jeweils bei ihrer Verwendung in den einzelnen Funktionen beschrieben.

6.1.2 Die Funktion 'getlines'

Tabelle 6.2: Ein- und Ausgabe von Variablen in 'private function getlines()'

Eingangsvariable	PHP-Funktion bzw. -Sprachkonstrukt	Ausgangsvariable
$pattern $lines	foreach preg_match	$out

Listing 6.2: private function getlines()

```
private function getlines($lines, $pattern) {
foreach ($lines as $line) {
if (preg_match($pattern, $line) === 1) {
$out[] = $line;
            }
        } return $out;
    }
```

Die erste Funktion dient dem Auslesen eines Arrays mit einem Muster. Über das Sprachkonstrukt 'foreach' wird das Array zeilenweise ausgelesen. In jeder Zeile wird nun die PHP-Funktion 'preg_match' verwendet, um mit dem Muster ('$pattern') bzw. dem reguären Ausdruck die Zeile zu filtern. Gibt es eine Übereinstimmung mit dem Muster wird das Ergebnis, zusammen mit allen anderen Treffern, am Ende wieder zu einem Array ('$out[]') zusammengesetzt und ausgegeben.

6.1.3 Die Funktion 'get_version'

Tabelle 6.3: Ein- und Ausgabe von Variablen in 'public function get_version'()

Eingangsvariable	PHP-Funktion bzw. -Sprachkonstrukt	Ausgangsvariable
$command (FFMPEGBIN)	exec()	$ret
- $pattern (VERSIONPATTERN) - $ret[0]	preg_match()	$match[0]

Listing 6.3: public function get_version()

```
public function get_version($pattern=
create_decklink_devices::VERSIONPATTERN) {
exec(create_decklink_devices::FFMPEGBIN.' -version',
$ret, $status);

if (preg_match($pattern, $ret[0], $match) === 1) {
return $match[0];
        }
return false;
    }
```

Diese Funktion dient dazu, die Version von FFmpeg auszugeben. Mittels der PHP-Funktion 'exec()' wird die Abfrage mit der Konstanten 'VERSIONPATTERN' (der binary von FFmpeg) und dem Zusatz „-version“ ausgeführt. Die Ausgabe wird in die Variable '$ret[]' umgeleitet. Auch hier wird mit der PHP-Funktion 'preg_match' und einem Muster (VERSIONPATTERN) die Ausgabe gefiltert und das Ergebnis in Form der Variable '$match' ausgegeben. Wenn keine Version gefunden wurde, gibt die Funktion ein 'false' aus.

6.1.4 Die Funktionen 'get_decklink_devices' und 'capture_formats'

Die Funktionen 'public function get_decklink_devices' & 'public function capture_formats' sollen in einem Abschnitt behandelt werden, da sie diekt voneinander abhängig sind. Die Ausgabe der Funktion durch '$this -> capture_formats' wird in der Funktion 'get_decklink_devices' unmittelbar genutzt, um eine sogenannte verschachtelte Auswahlliste auszugeben. Nachfolgend sind ein Schema der verschachtelten HTML-Auswahlliste und vorher ein Auschnitt der Ausgabe aus der Kommandozeile, die in die Liste integriert werden soll, dargestellt.

Listing 6.4: Ausgabe für Gerät DeckLink SDI (1) - (Ausschnitt)

```
[decklink @ 0x22d6560] Supported formats for 'DeckLink SDI (1)':
[decklink @ 0x22d6560] 1   720x486  at 30000/1001 fps (in..st)
[decklink @ 0x22d6560] 2   720x486  at 24000/1001 fps
[decklink @ 0x22d6560] 3   720x576  at 25000/1000 fps (in..st)
[decklink @ 0x22d6560] 4  1920x1080 at 24000/1001 fps
[decklink @ 0x22d6560] 5  1920x1080 at 24000/1000 fps
[decklink @ 0x22d6560] 6  1920x1080 at 25000/1000 fps
[decklink @ 0x22d6560] 7  1920x1080 at 30000/1001 fps
[decklink @ 0x22d6560] 8  1920x1080 at 30000/1000 fps
[decklink @ 0x22d6560] 9  1920x1080 at 25000/1000 fps (in..st)
[decklink @ 0x22d6560] 10 1920x1080 at 30000/1001 fps (in..st)
[decklink @ 0x22d6560] 11 1920x1080 at 30000/1000 fps (in..st)
[decklink @ 0x22d6560] 12 1280x720  at 50000/1000 fps
[decklink @ 0x22d6560] 13 1280x720  at 60000/1001 fps
[decklink @ 0x22d6560] 14 1280x720  at 60000/1000 fps
```

Listing 6.5: Schema HTML - 'verschachtelte Auswahlliste'

```
<select name='input_devices'>
<optgroup label="Geraet 1">
<option value='Option1-fuer-Geraet 1'>Option1</option>
<option label='Option2-fuer-Geraet 1'>Option2</option>
<option label='Option3-fuer-Geraet 1'>Option3</option>
</optgroup>
<optgroup label="Geraet 2">
<option label='Option1-fuer-Geraet 2'>Option1</option>
<option label='Option2-fuer-Geraet 2'>Option2</option>
<option label='Option3-fuer-Geraet 2'>Option3</option>
</optgroup>
</select>
```

Der „Rahmen“ dieser Auswahlliste wird von der ersten Funktion ausgegeben. An die Positionen des ’labels’ für die optgroup, also “Geraet 1“ und “Geraet 2“, werden durch die Funktion ’get_decklink_devices’ die richtigen Blackmagic- DeckLink- Geräte eingesetzt.

Tabelle 6.4: Ein- und Ausgabe von Variablen in ’public function get_decklink_devices()’

Eingangsvariable	PHP-Funktion bzw. -Sprachkonstrukt	Ausgangsvariable
$command (FFMPEGBIN)	exec()	$ret
- $pattern (DEVICEPATTERN) - $ret	getlines()	$out
- $pattern - $out	foreach preg_match()	$devicelist

Listing 6.6: public function get_decklink_devices()

```
public function get_decklink_devices(
$pattern=create_decklink_devices::DEVICEPATTERN)
{

exec(create_decklink_devices::FFMPEGBIN.
' -f decklink -list_devices 1 -i dummy 2>&1', $ret, $status);

# $out gibt die Liste der Geraete aus
$out = $this->getlines($ret, $pattern);

# match for 'DeckLink SDI (x)' --> "\'(.+)'"
$devicelist = "<select name='input_devices'>".PHP_EOL;

foreach ($out as $line) {

if (preg_match("/\'(.+)'/", $line, $match) === 1) {
$devicelist .= "<optgroup label='".$match[1]."'>".PHP_EOL;
$devicelist .= $this->capture_formats($match[1]);
$devicelist .= "</optgroup>".PHP_EOL;
            }
        }
$devicelist .= "</select>".PHP_EOL;
return $devicelist;
    }
```

Mit der PHP-Funktion ’exec’ und der Konstanten FFMPEGBIN mit dem Zusatz ’ -f decklink -list_devices 1 -i dummy 2>&1’ wird ein Befehl ausgeführt, der die Geräte von FFmpeg ausgibt. Diese Ausgabe wird in die Variable ’$ret’ gespeichert. Dieses Array ’$ret’

wird mit der Funktion 'getlines()' zeilenweise ausgelesen und jede Zeile, in der es einen Treffer für die Konstante 'DEVICEPATTERN' ('$pattern') gibt, wird in der Variable $out gespeichert. Mit dem Sprachkonstrukt wird nun die Variable $out zeilenweise ausgelesen und mit der PHP-Funktion 'preg_match' und einem neuen regulären Ausdruck ("/ \' (.+) ' /") werden die endgültigen Geräte als '$match[1]' herausgefiltert: 'DeckLink SDI (1)' und 'DeckLink SDI (2)'. In die Variable '$deviceslist' wird nun jedes Gerät als '$match[1]' an die Stelle "<optgroup label="$match[1]">" geschrieben und an die folgende Funktion durch '$this->capture_formats($match[1])' übergeben.

Tabelle 6.5: Ein- und Ausgabe von Variablen in 'public function capture_formats()'

Eingangsvariable	PHP-Funktion bzw. -Sprachkonstrukt	Ausgangsvariable
- $command (FFMPEGBIN) - $match[1] -> $device	exec()	$ret
- $pattern (DEVICEFORMATPATTERN) - $ret	getlines()	$out
- $pattern - $out - $device	foreach preg_match()	$devicelist

Listing 6.7: public function capture_formats()

```
public function capture_formats(
$device,
$pattern=create_decklink_devices::DEVICEFORMATPATTERN) {

exec(create_decklink_devices::FFMPEGBIN.
" -f decklink -list_formats 1 -i '$device' 2>&1",$ret,$status);

$out = $this->getlines($ret, $pattern);
$i = 1;
$paramlist = "";
foreach ($out as $line) {
$paramlist .= "<option value='$device@$i'>".
trim(preg_replace($pattern, '', $line))."</option>".PHP_EOL;
$i++;
        }

return $paramlist;
    }
}
```

Diese Funktion gibt die verfügbaren Blackmagic DeckLink Formate aus und vollendet gleichzeitig die verschachtelte HTML-Auswahlliste. Mit der PHP-Funktion 'exec', der Konstanten FFMPEGBIN mit dem Zusatz ' -f decklink -list_devices 1 -i '$device' 2>&1' und der Variable 'match[1]' aus der vorherigen Funktion, die jetzt als Variable '$device' genutzt wird, wird der neue Befehl ausgeführt. Dieser gibt die Formate für jedes Gerät in die Variable '$ret' aus, wie sie am Anfang dieses Kapitels für 'DeckLink SDI (1)' (siehe Listing "Ausgabe für Gerät DeckLink SDI (1) - (Ausschnitt)") dargestellt sind. Diese Variable '$ret' wird nun zeilenweise ausgelesen und mit dem Muster 'DEVICEFORMATPATTERN' überprüft. Die Zeilen mit Treffern werden in die Variable $out gespeichert. Mit dem Sprachkonstrukt 'foreach' wird die Variable '$out' zeilenweise ausgelesen und in die Variable $paramlist geschrieben. In jeder Zeile wird die Option "<option value='$device@$i'>" hinzugefügt und durch die PHP-Funktion 'trim()' und das Muster 'DEVICEFORMATPATTERN' werden nun alle überflüssigen Zeichen entfernt und nur die Auflösungen als Bezeichnung für die Option geschrieben (im "Schema HTML - 'verschachtelte Auswahlliste'" wäre das die Position der 'Option1', 'Option2' usw.).

6.2 Filter - die PHP-Klasse 'create_ffmpeg_filter.php'

In der PHP-Klasse create_ffmpeg_filter sind die Funktionen zusammengefasst, die die Optionen für den Benutzer zum Anpassen der Filter für die späteren Streams in FFmpeg bereitstellen.

6.2.1 Variablendefinition

Listing 6.8: Variablendefinition

```
public $outputs =
array('1','2','3','4');

public $res_video_list =
array('640x360','1024x576','1280x720','1920x1080');

public $codec_video_list =
array('libx264');

public $bit_audio_list =
array('128','192','224','256');

public $bit_video_list = array();

public $gops_video =
array('25','50','75','100','125','150','175','200');
```

Für die Funktionen in der Kategorie Filter wurden Arrays vordefiniert. Diese enthalten Zahlenwerte, die gleichzeitig als Option und als Wert für die Auswahllisten dienen. Des Weiteren können sie später unkompliziert an andere Bedürfnisse angepasst werden können.

6.2.2 Die Funktionen 'filter_res', 'filter_bit_audio' und 'filter_gops'

Tabelle 6.6: Ein- und Ausgabe von Variablen in 'public function filter_res()'

Eingangsvariable	PHP-Funktion bzw. -Sprachkonstrukt	Ausgangsvariable
$res_video_list	foreach	$out

Listing 6.9: public function filter_res()

```
public function filter_res() {
$out = ">".PHP_EOL;
foreach ($this->
res_video_list as $item) {

$out .= "<option value=$item>$item
</option>".PHP_EOL;
            }
$out .= "</select>".PHP_EOL;
return $out;
    }
```

Die Funktonen 'filter_res' , 'filter_bit_audio' und 'filter_gops' sind identisch aufgebaut, deshalb soll hier beispielhaft auf die Funktion 'filter_res' eingegangen werden. Hier werden die Arrays verwendet, die am Anfang der Klasse definiert wurden. Mit dem Sprachkonstrukt 'foreach' wird das Array zeilenweise ausgelesen und die einzelnen Werte als Variable '$item' in die Variable '$out' integriert. Die Variable '$out' beinhaltet nun eine HTML-Auswahlliste mit den Werten aus dem Array als Wert und als Bezeichnung der Auswahl.

6.2.3 Die Funktionen 'filter_buf_video' und 'filter_bit_video'

Tabelle 6.7: Ein- und Ausgabe von Variablen in 'public function filter_bit_video()'

Eingangsvariable	PHP-Funktion bzw. -Sprachkonstrukt	Ausgangsvariable
$out	if()	$out

Listing 6.10: public function filter_bit_video()

```
public function filter_bit_video() {
$out = "<br>".PHP_EOL;
if (100<=$out and $out<=15000 and is_numeric($out)) {
return $out;
        }
return $out = "<input name='filter_bit_video' type='text' value
   ='Bitte Wert zw. 100 - 15000 eingeben.'>".PHP_EOL;
}
```

Auch die Funktionen 'filter_buf_video' und 'filter_bit_video' sind beide gleich aufgebaut. Sie geben eine Textbox aus, in die der Benutzer eigene Werte eingeben kann. Über eine if-Anweisung wird gepüft, ob es sich um Werte im Bereich von 100 bis 15000 handelt, um sicherzustellen das die Eingabe nur Zahlen enthält und das diese sinnvoll sind.

6.2.4 Die Funktion 'filter_outputs2'

Tabelle 6.8: Ein- und Ausgabe von Variablen in 'public function filter_outputs2()'

Eingangsvariable	PHP-Funktion bzw. -Sprachkonstrukt	Ausgangsvariable
$border	filter_res() filter_bit_audio() filter_gops() filter_buf_video() filter_bit_video() for	(echo "String....")

Listing 6.11: public function filter_outputs2()

```
public function filter_outputs2 () {
$border = 5;
for($i=1;$i<=$border;$i++){
echo "<h3>Output ".$i."</h3>";

echo "<table>";
echo "<tr><td><label>Output waehlen:</label></td><td><input type
   ='checkbox' id='mc' name='auswahl_output$i' value='auswahl
   '></td></td><td></tr>".PHP_EOL;

echo "<tr><td><label>Aufloesung:</label></td>".PHP_EOL;
echo "<td><select name='filter_resolution$i' ";
```

```
echo $this->filter_res()." </td><td>Pixel</td></tr>".PHP_EOL;

echo "<tr><td><label>Audio-Bitrate:</label></td>".PHP_EOL;
echo "<td><select name='filter_bit_audio$i' ";
echo $this->filter_bit_audio()." </td><td>kbit/s</td></tr>".
    PHP_EOL;

echo "<tr><td><label>Video-Bitrate:</label></td>".PHP_EOL;
echo "<td><input name='filter_bit_video$i' type='text' size
    ='10'>";
echo $this->filter_bit_video()." </td><td>kbit/s</td></tr>".
    PHP_EOL;

echo "<tr><td><label>Video-Puffergroesse:</label></td>".PHP_EOL;
echo "<td><input name='filter_buf_video$i' type='text' size
    ='10'>";
echo $this->filter_buf_video()." </td><td>kByte</td></tr>".
    PHP_EOL;

echo "<tr><td><label>Group of Pictures:</label></td>".PHP_EOL;
echo "<td><select name='filter_gops$i' ";
echo $this->filter_gops()."</td><td>(GoP)</td></tr>".PHP_EOL;

echo "</table>";
}
echo "<h3>Protokolldatei</h3>";

echo "<label>Aufzeichnung fuer Protokollierung erstellen?</label
    ><input type='checkbox' id='mc' name='auswahl_protokoll'
    value='protokoll'><br>".PHP_EOL;
}
```

Die Funktion 'public function filter_outputs2' nutzt die vorher beschriebenen Funktionen. Diese werden mit Hilfe des Sprachkonstruktes 'echo' in eine HTML-Tabelle eingeordnet. Die vordefinierte Variable '$border' dient dabei als Grenze für die For-Schleife, die 5 dieser Tabellen mit einer Checkbox für die Auswahl durch den Benutzer und mit den Funktionen ausgeben lässt. Nach dieser Tabelle wird weiterhin eine Checkbox für die Auswahl einer Protokolldatei ausgegeben.

Listing 6.12: Ausschnitt - public function filter_outputs2()

```
echo "<tr><td><label>Aufl\"osung:</label></td>".PHP_EOL;
echo "<td><select name='filter_resolution$i' ";
echo $this->filter_res()." </td><td>Pixel</td></tr>".PHP_EOL;
```

Die Ausgabe der Auflösung soll beispielhaft noch einmal näher betrachtet werden. Für die Erklärung der Einstellung wird ein HTML-label in der ersten Spalte der Tabelle ausgegeben. In der nächsten Spalte wird die Auswahllliste mit dem Attribut “name=’filter_resolution$i’“ definiert, wobei auch hier die Zählvariable ’$i’ genutzt wird, um die 5 verschieden Auflösungen für je einen Stream unterscheiden zu können. Anschließend wird die eigentliche Funktion ’filter_res’ aufgerufen und in der letzten Spalte mit der Einheit „Pixel“ versehen.

6.3 Output - die PHP-Klasse ’create_ffmpeg_profil’

6.3.1 Die Funktion ’public function create’, ein Fragment

Listing 6.13: public function create() - Fragment

```
public function create() {
# Dateiname

$name = htmlspecialchars($_POST["datei_name"]);

$myfile = fopen("'$name'.sh", "w") or die("Unable to open file!"
    );
## hier ueber funktion den inhalt von bashskript einfuegen
    lassen
$suff = '#!/bin/sh';
$suff .= '#';
$dir = 'OUTPUT=/srv/encoder/$(date +%Y%m%d_%H.%M.%S).mkv.\n'.
    PHP_EOL;

##Protyp furr Anzahl Streams bzw Einstellungen fuer einzelnen
    Stream aus live.sh
/*
-filter_complex "[0:1]yadif,split=4[v1][v2][v3][v4];[0:0]asplit
    =4[a1][a2][a3][a4]" \
-map "[v1]" -map "[a1]" -s 1280x720 \
-pix_fmt yuv420p -c:v libx264 -b:v 4500k -bufsize 9000k -g 100 -
    preset:v faster -c:a aac -ar 48000 -b:a 224k \
-f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/liveevent1?
    adbe-live-event=liveevent" \
*/
##
#### if-abfrage der radio buttons und dann einfach jeweils
    ausgabe der outs??? bzw der v1, v2, v....
####

## hier output variable einsetzen ...
```

```
## ffmpeg -nostats -loglevel 0 -f decklink -i 'DeckLink SDI (2)
   @9' -r 25 -threads 0 \

$in = 'ffmpeg -nostats -loglevel 0 -f decklink -i '.
   htmlspecialchars($_POST["input_devices"]) .' -r 25 -threads 0
    \ '.PHP_EOL;


## hier (ueber anzahl radio buttons???) anzahl der streams
   einfuegen lassen

$all = '-filter_complex "[0:1]yadif,split=4[v1][v2][v3][v4
   ];[0:0]asplit=4[a1][a2][a3][a4]" \ '.PHP_EOL;

## stream no1, variable fuer aufloesung

$out1 = '-map "[v1]" -map "[a1]" -s '. htmlspecialchars($_POST["
   filter_resolution1"]) . '\ '.PHP_EOL;

$out1 .= '-pix_fmt yuv420p -c:v libx264 -b:v '. htmlspecialchars
   ($_POST["filter_bit_video1"]) . 'k -bufsize '.
   htmlspecialchars($_POST["filter_buf_video1"]) . 'k -g '.
   htmlspecialchars($_POST["filter_gops1"]) . ' -preset:v faster
    -c:a aac -ar 48000 -b:a ' . htmlspecialchars($_POST["
   filter_bit_audio1"]) . 'k \ '.PHP_EOL;

$out1 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/
   liveevent1?adbe-live-event=liveevent" \ '.PHP_EOL;

## stream no2

$out2 = '-map "[v2]" -map "[a2]" -s '.htmlspecialchars($_POST["
   filter_resolution2"]) . ' \ '.PHP_EOL;

$out2 .= '-pix_fmt yuv420p -c:v libx264 -b:v '. htmlspecialchars
   ($_POST["filter_bit_video2"]) . 'k -bufsize '.
   htmlspecialchars($_POST["filter_buf_video2"]) . 'k -g '.
   htmlspecialchars($_POST["filter_gops2"]) . ' -preset:v faster
    -c:a aac -ar 48000 -b:a ' . htmlspecialchars($_POST["
   filter_bit_audio2"]) . 'k \ '.PHP_EOL;

$out2 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/
   liveevent2?adbe-live-event=liveevent" \ '.PHP_EOL;

## stream no3

$out3 = '-map "[v3]" -map "[a3]" -s '.htmlspecialchars($_POST["
   filter_resolution3"]) . ' \ '.PHP_EOL;

$out3 .= '-pix_fmt yuv420p -c:v libx264 -b:v '. htmlspecialchars
```

```
    ($_POST["filter_bit_video3"]) . 'k -bufsize '.
    htmlspecialchars($_POST["filter_buf_video3"]) . 'k -g '.
    htmlspecialchars($_POST["filter_gops3"]) . ' -preset:v faster
     -c:a aac -ar 48000 -b:a ' . htmlspecialchars($_POST["
    filter_bit_audio3"]) . 'k \ '.PHP_EOL;

$out3 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/
    liveevent3?adbe-live-event=liveevent" \ '.PHP_EOL;

## stream no4

$out4 = '-map "[v3]" -map "[a3]" -s ' . htmlspecialchars($_POST[
    "filter_resolution4"]) . ' \ '.PHP_EOL;

$out4 .= '-pix_fmt yuv420p -c:v libx264 -b:v '. htmlspecialchars
    ($_POST["filter_bit_video4"]) . 'k -bufsize '.
    htmlspecialchars($_POST["filter_buf_video4"]) . 'k -g '.
    htmlspecialchars($_POST["filter_gops4"]) . ' -preset:v faster
     -c:a aac -ar 48000 -b:a ' . htmlspecialchars($_POST["
    filter_bit_audio4"]) . 'k \ '.PHP_EOL;

$out4 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/
    liveevent3?adbe-live-event=liveevent" \ '.PHP_EOL;

## stream no5

$out5 = '-map "[v3]" -map "[a3]" -s ' . htmlspecialchars($_POST[
    "filter_resolution5"]) . ' \ '.PHP_EOL;

$out5 .= '-pix_fmt yuv420p -c:v libx264 -b:v '. htmlspecialchars
    ($_POST["filter_bit_video5"]) . 'k -bufsize '.
    htmlspecialchars($_POST["filter_buf_video5"]) . 'k -g '.
    htmlspecialchars($_POST["filter_gops5"]) . ' -preset:v faster
     -c:a aac -ar 48000 -b:a ' . htmlspecialchars($_POST["
    filter_bit_audio5"]) . 'k \ '.PHP_EOL;

$out5 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/
    liveevent3?adbe-live-event=liveevent" \ '.PHP_EOL;

## stream sicherheit

$safe = '-map "[v4]" -map  "[a4]" -s 640x360 \ '.PHP_EOL;
$safe .= '-pix_fmt yuv420p -c:v libx264 -preset:v faster -c:a
    aac -ar 48000 -b:a 128k $OUTPUT '.PHP_EOL;
$inhalt = $suff . $dir . $in . $all .$out1 . $out2 . $out3 .
    $out4 . $out5 . $safe;
fwrite($myfile, $inhalt);
#$txt = "Jane Doe\n";
#fwrite($myfile, $txt);
fclose($myfile);
```

```
return $myfile;

#<form method="get" action="<?php echo $_SERVER['PHP_SELF']; ">
echo "<a href='$myfile' download>Download</a>".PHP_EOL;
#<button type="submit">Download!</button>
#</form>
#<button type="submit" onclick="window.open('file.doc')">
    Download!</button>

}
```

Die Funktion public function create() ist nur als Fragment vorhanden und diente als Vorlage für den Inhalt der Downloadseite. Nichtsdestotrotz soll sie an dieser Stelle berücksichtigt werden. Eine nähere Beschreibung erfolgt im Kapitel Download.

6.4 Das Webinterface - 'wrapper.php'

Alle Einstellungen des Benutzers werden auf der PHP-Seite 'wrapper.php' vorgenommen, in der die vorher beschriebenen Funktionen integriert und angewendet werden.

6.4.1 Instanzierung der Klassen

Listing 6.14: Instanzierung

```
include 'create_decklink_devices.php';
include 'create_ffmpeg_filter.php';
include 'create_ffmpeg_profil.php';

$blackmagic = new create_decklink_devices();
$ffmpegfilter = new create_ffmpeg_filter();
$ffmpegprofil = new create_ffmpeg_profil();
```

Die Klassen 'create_decklink_devices.php', 'create_ffmpeg_filter.php' und 'create_ffmpeg_profil.php' werden inkludiert. Danach werden Instanzen für alle drei Klassen erzeugt, um die darin enthaltenenen Funktionen im PHP-Skript nutzbar zu machen. Über das Sprach-Konstrukt 'echo' werden der Kopf und der Haupteil, beginnend mit der Hauptüberschrift, ausgegeben.

6.4.2 Die Anwendung der Funktionen

Listing 6.15: Anwendung der Funktionen

```
echo "<label>FFmpeg-Version:  </label>";
echo $blackmagic->get_version();

#Input-Funktionen

echo "<h2>Input</h2>";

echo "<form name='form' action='creator.php' method='POST'
   target='_blank'><br>".PHP_EOL;
echo $blackmagic->get_decklink_devices();

#Filter-Funktionen

echo "<h2>Filter</h2>";

echo "<label>Bitte waehlen Sie die Anzahl der Outputs durch
   setzen der Haekchen!</label><br>";

echo $ffmpegfilter->filter_outputs2()."<br>".PHP_EOL;
```

Im Hauptteil werden die Funktionen für die Version, für den Input und für den Filter angewendet. Mit Hilfe der Instanz für die Klasse 'echo $blackmagic->get_version()' wird die Version von FFmpeg angezeigt.

Danach wird die HTML-Form eingeleitet, wie sie schon im Entwurf der 'wrapper.php' zu sehen war. Das Attribut 'action' wird mit der PHP-Datei 'creator.php' belegt, die beim Klicken auf den „Submit"-Button ausgeführt und deren Inhalt durch das Attribut „target='_blank'" in einem neuen Tabulator geöffnet wird.

In dieser Form wird für die Kategorie Input mit 'echo $blackmagic->get_decklink_devices()' die in der Funktion 'get_decklink_devices()' beschriebene verschachtelte HTML-Auswahlliste ausgegeben und es werden dem Benutzer somit die DeckLink-Geräte angezeigt.

Des Weiteren werden für die Kategorie Filter mit 'echo $ffmpegfilter->filter_outputs2()' die Einstellungen für die 5 Streams ausgegeben.

6.4.3 Weiterte Einstellungen in der HTML-Umgebung

Innerhalb der Form kann in einem Textfeld der Name für das Bash-Skript eingegeben werden und am Ende folgt eine Schaltfläche, also ein Button der Art “<input type=’submit’>“, mit der Bezeichnung ’Profil erzeugen’ durch den die eigentliche Aktion und somit die ’creator.php’ in der Form ausgeführt wird. Die HTML-Seite wird am Ende wieder mit dem Sprach-Konstrukt ’echo’ ordnungsgemäß beendet.

Listing 6.16: Einstellungen in der HTML-Umgebung

```
#### Profil-Name ####

echo "<h2>Profil</h2>";

echo "Hier bitte den Namen des Profils eingeben: <br>";
echo "<input type='text' name='datei_name'><br>";
echo "<br>";

#### Profil erstellen ####

echo "Wenn Sie auf erzeugen druecken, <br>";
echo "wird das Profil erzeugt und <br>";
echo "auf der naechsten Seite angezeigt! <br><br>";
echo "<input type='submit' value='Profil erzeugen'><br>".PHP_EOL
    ;
```

6.5 Download - ’creator.php’

6.5.1 Die Erzeugung des Skriptnamens und des Skriptes

Listing 6.17: Erzeugung Skriptname und Bash-Skript

```
#### Dateiname

$name = htmlspecialchars($_POST["datei_name"]);

$script = $name.'.sh';

$myfile = fopen($script, "w") or die("Unable to open file!");
```

```
#### Inhalt vom Skript

$suff = '#!/bin/sh'.PHP_EOL;
$suff .= '#'.PHP_EOL;
$dir = 'OUTPUT=/srv/encoder/$(date +%Y%m%d_%H.%M.%S).mkv'.
    PHP_EOL;

#### Input

$in = 'ffmpeg -nostats -loglevel 0 -f decklink -i '.
    htmlspecialchars($_POST["input_devices"]) .' -r 25 -threads 0
     \ '.PHP_EOL;
```

Die 'creator.php' hat ihren Namen erhalten, weil erst in ihr das Bash-Skript, sowohl aus variablen Werten als auch aus vordefinierten Werten, erzeugt wird. Die Variablen werden mit Hilfe der HTTP-Methode 'POST' von der 'wrapper.php' übergeben und der PHP-Funktion '$_POST' in der 'creator.php' abgefangen. Auch sie gibt im Wesentlichen eine HTML-Seite aus.

Am Anfang werden die Variablen, die später mit den Werten des Benutzers belegt werden, definiert und, um eine "saubere" Programmierung zu vollziehen, als leere Variablen vordefiniert.

Die Variable für den Namen des Skriptes wird als erstes mit einem Wert belegt. Zusammen mit dem Suffix '.sh' wird diese in einer neuen Variable als zusammengesetzter String benutzt um einen vollständigen Dateinamen zu erhalten. Dieser vollständige Dateiname wird benötigt, um danach mit der PHP-Funktion 'fopen' bereits die spätere Datei zum Schreiben zu öffnen bzw. zu erzeugen.

Nun wird der „Kopf" des Bash-Skriptes mit Variablen, die nur einen vordefinierten String enthalten, definiert. Darauf folgt die erste wichtige Variable, an die ein Wert aus der 'wrapper.php' übergeben wird. Der Input bzw. das Gerät, das der Benutzer aus der Auswahlliste gewählt hat und Einstellungen für FFmpeg werden zur Variable '$in' zusammengefügt.

6.5.2 Der Inhalt des Skriptes

Listing 6.18: Inhalt des Bash-Skriptes

```
## Stream Nr. 1

if (isset($_POST["auswahl_output1"])) {

$out1 = '-map "[v1]" -map "[a1]" -s ' . htmlspecialchars($_POST[
    "filter_resolution1"]) . ' \ ' . PHP_EOL;

$out1 .= '-pix_fmt yuv420p -c:v libx264 -b:v ' .
    htmlspecialchars($_POST["filter_bit_video1"]) . 'k -bufsize '
     . htmlspecialchars($_POST["filter_buf_video1"]) . 'k -g ' .
    htmlspecialchars($_POST["filter_gops1"]) . ' -preset:v faster
     -c:a aac -ar 48000 -b:a ' . htmlspecialchars($_POST["
    filter_bit_audio1"]) . 'k \ ' . PHP_EOL;

$out1 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/
    liveevent1?adbe-live-event=liveevent" \ ' . PHP_EOL;

$v1 = '[v1]';
$a1 = '[a1]';
$i =$i++;

    }
```

Es folgen die 5 möglichen Outputs mit ihren Filtern, die vom Benutzer auf der 'wrapper.php' gewählt werden konnten. Bei jedem der 5 Streams, denn das sind die hier verwendeten Outputs, wird über die PHP-Funktion 'isset' geprüft, ob der Benutzer die "checkbox" gewählt hat und nur dann wird dieser Output überhaupt im Skript verwendet.

Nun werden mit der PHP-Funktion '$_POST' die Werte für die Auflösung, für die Video-Puffer- und Bitrate, die 'Group of Pictures' und die Audio-Bitrate und die vordefinierten Werte zu den Variablen für die jeweiligen Streams zuammengeführt. Am Ende eines jeden Streams wird eine Zählvariable nach oben gezählt, weil die Anzahl der Outputs für das spätere Skript wichtig ist, da je nach ihrer Anzahl die Zeile für das "mapping" mit einer anderen Anzahl von Audio- und Videosignalen definiert werden muss.

6.5.3 Das Schreiben in das Skript

Listing 6.19: Schreiben in das Bash-Skript

```
## (Fortsetzung) Anzahl der Streams

$all = '-filter_complex "[0:1]yadif,split='. $i . $v1 . $v2 .
    $v3 . $v4 . $v5 .';[0:0]asplit='. $i .$a1 . $a2 . $a3 . $a4 .
     $a5 .'" \ '.PHP_EOL;

$inhalt = $suff . $dir . $in . $all .$out1 . $out2 . $out3 .
    $out4 . $out5 . $safe;

fwrite($myfile, $inhalt);

fclose($myfile);
```

Alle Variablen, der „Kopf“ des Skriptes, der Input, die Streams und alle anderen Bestandteile, werden jetzt zu einer Variablen mit dem Namen ’$Inhalt’ vereint. Diese wird nun in das am Anfang erzeugte Bash-Skript geschrieben, bevor es geschlossen wird.

6.5.4 Die Ausgabe auf der Seite

Listing 6.20: Ausgabe auf der ’creator.php’

```
echo "Dies ist ihr Profil:  ";
echo "$script<br>";
echo "<br>";

echo "<label>Stream1</label><br>";
echo "$out1<br>";

echo "<br>";

echo "<label>Stream2</label><br>";
echo "$out2<br>";

echo "<br>";

echo "<label>Stream3</label><br>";
echo "$out3<br>";

echo "<br>";
```

```
echo "<label>Stream4</label><br>";
echo "$out4<br>";

echo "<br>";

echo "<label>Stream5</label><br>";
echo "$out5<br>";

echo "<br>";

echo "<label>Sicherheitskopie</label><br>";
echo "$safe<br>";

echo "<br>";
echo "<br>";

echo "Hier koennen Sie das Profil als Bash-Skript herunterladen:
   ";

echo "<br>";
echo "<br>";

echo "<a style='color: red' href='$script' download>Download</a>
   ".PHP_EOL;

echo "</body>";
echo "</html>";
```

Die Variablen für den Skriptnamen, für den Input und für die Streams werden außerdem genutzt, um eine Übersicht auf der 'creator.php' darzustellen. Um den Download des Skriptes vollziehen zu können, wird kein klassischer „Download-Button“ implementiert, sondern das Wort „Download“ wird in einen HTML- '<a>'-tag integriert. Dieser 'a-tag' erhält die Attribute 'color, 'href' und 'download'. Das Attribut 'color' soll das einzelne Wort mit der Farbe rot hervorheben. Das Attribut 'href=$script' in Kombination mit dem Attribut 'download' sorgt dafür, das durch das Klicken auf das Wort die verlinkte Datei '$script', also das Bash-Skript, heruntergeladen wird.

7 Überprüfung

7.1 Testaufbau

Da die Capture- Karte keinen konventionellen Videoanschluss besitzt, wie zum Beispiel HDMI, muss ein Converter eingesetzt werden. Hierbei kommt ein yellobrik(c) CHD 1812 der Firma Lynx Technik AG zum Einsatz, der ein HDMI-Signal in ein SDI-Signal umwandeln kann. [23] Dieser wird direkt mittels eines SDI- Koaxialkabels mit einem Eingang der PCIe- Karte verbunden.
Als letztes Bindeglied, um ein Testsignal für FFmpeg zur Verfügung stehen zu haben, wird eine IPTV Set-Top-Box des Herstellers Amino mit der Bezeichnung A140 verwendet. [24] Dieser wird mit einem LAN- Kabel direkt an eine Netzwerksteckdose angeschlossen und empfängt mit einer bereits voreingestellten Konfiguration das IPTV der Hochschule.
Über all diese Komponenten kann nun ein Digitales- Fernsehsignal an den Testcomputer bzw. die Blackmagic DeckLink gelangen und die Funktionalität von FFmpeg und des Interfaces getestet werden.

In folgendem Schema sieht man den Testaufbau noch einmal in der Übersicht.

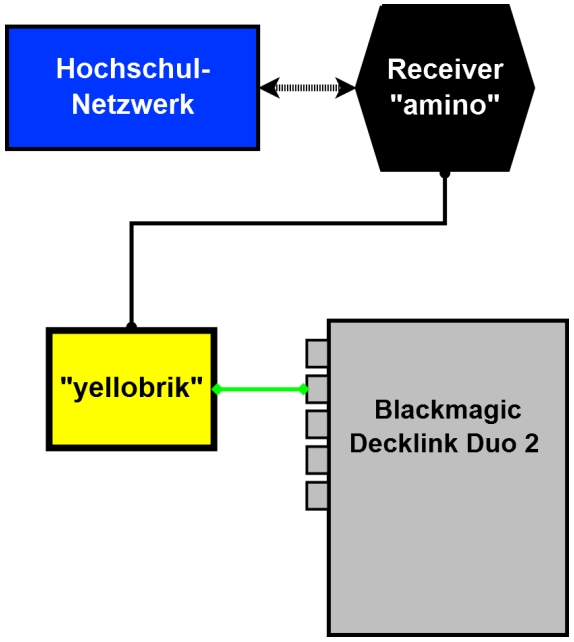

Abbildung 7.1: Testaufbau

[23] vgl. LYNXTechnikAG, 2017 [LyTeYell]
[24] vgl. Amino Communications, 2017 [AmA140]

7.2 Test der Software nach den Anforderungen

In diesem Kapitel werden die Anforderungen wieder aufgegriffen und überprüft, ob diese erfüllt wurden. Die Tabellen der funktionalen Anforderungen ('FA_x') und nichtfunktionalen Anforderungen ('NA_x') werden um die Spalte 'Feststellung' erweitert.

7.2.1 Test der Funktionalen Anforderungen

Tabelle 7.1: Prüfung der Funktionalen Anforderungen

Nr.	Beschreibung	Problembeschreibung	Feststellung
FA_1	Es sollen Textboxen, Auswahllisten, Checkboxen für benutzerspezifische Eingaben für den Software-Encoder FFmpeg zur Verfügung stehen.	Es müssen HTML-Form-Elemente mit Formularen wie Textboxen, Auswahllisten, und Checkboxen implementiert werden.	Im Webinterface wurde ein Form-Element implementiert, welches alle Formulare in sich vereint.
FA_2	Die angebotenen Einstellungen sollen unter anderem Bildgröße, Bitrate, Quelle, Ziel, Anzahl der Streams beinhalten.	Es muss entschieden werden, welche Einstellung auf welcher Weise (HTML-Formular) angeboten wird und welche Werte vorgegeben werden.	Für Video-Bitrate und -Puffergröße wurden Textboxen verwendet, für Auflösung, Audio-Bitrate und 'Group of Pictures' wurden Auswahllisten verwendet und für die Wahl des Outputs wurden Checkboxen verwendet.
FA_3	Das Webinterface soll die Einstellungen anbieten und auf einer weiteren Webseite wird das Ergebnis angezeigt.	Es müssen zwei voneindander abhängige Webseiten implementiert werden.	Es wurde das Webinterface mit dem Namen 'wrapper.php' und eine Downloadseite mit dem Namen 'creator.php' entwickelt.
FA_4	Der Benutzer soll als Endresultat auf dieser zweiten Webseite ein Bash-Skript herunterladen können, mit dem er den Software-Encoder betreiben kann.	Es müssen Funktionen entwickelt werden, die ein Bash-Skript aus den Einstellungen erzeugen.	Auf der Downloadseite kann man einen Download ausführen lassen, der dem Benutzer das Herunterladen des Bash-Skriptes ermöglicht

7.2.2 Test der Nichtfunktionalen Anforderungen

(siehe nächste Seite)

Tabelle 7.2: Prüfung der Nichtfunktionalen Anforderungen

Nr.	Beschreibung	Problembeschreibung	Feststellung
NA_1	Die Software soll in den Programmier- bzw. Skriptsprachen Python/PHP/HTML/Bash implementiert werden.	Es müssen Schnittstellen für die Programmier- und Skriptsprachen gefunden werden.	Die HTML-Seite wird durch das Sprachkonstrukt 'echo' vom PHP-Skript dargestellt, während das Skript mit Hilfe der PHP-Funktionen 'fopen', 'fwrite' und 'fclose' erzeugt wird.
NA_2	Die Software soll in Klassen und Funktionen programmiert werden, um die Funktionen zu strukturieren und individuell nutzen zu können.	Für wiederkehrende Aufgaben müssen Funktionen implementiert werden, die in eine Struktur aus Klassen eingearbeitet werden.	Es wurden drei Klassen zur Strukturierung der Software entworfen, die die Funktionen enthalten, und zwei PHP-Seiten entwickelt, die die Funktionen nutzen.
NA_3	Das Webinterface muss mit der Kommandozeile des Linux-Systems kommunizieren und die Ausgabe auffangen.	Der Zugriff auf die Kommandozeile mit Hilfe der oben genannten Programmier- und Skriptsprachen und das Abfangen der Ausgabe müssen implementiert werden.	Mit Hilfe der PHP-Funktion 'exec' werden die Befehle in der Kommandozeile ausgeführt, durch eine Shell-Umleitung wird die Ausgabe in eine Variable umgeleitet und durch reguläre Ausdrücke wird diese für den benötigten Zweck gefiltert.
NA_4	Die Informationsübertragung des Benutzers mit dem Webinterface wird mit der HTTP-Methode 'POST' ausgeführt.	Das Webinterface muss so implementiert werden, dass es die Daten per 'POST' versendet, und die zweite Webseite muss die Daten per 'Post' empfangen.	Attribut 'method' der HTML-Form des Webinterface wird mit der Defintion 'post' belegt, auf der Downloadseite werden die Werte mit der PHP-Funktion '$_POST' und den 'name'-Attributen abgefangen
NA_5	Das Webinterface soll klar gegliedert und die Einstellungen selbsterklärend sein.	Beide Webseiten sollen mit Hilfe von HTML-Überschriften und -Tabellen implementiert und durch kurze Hinweise und passende Einheiten selbsterklärend werden.	In der 'wrapper.php' wird über die implementierte Funktion eine HTML-Tabelle ausgegeben. (Inhalt: 5x die gleichen Einstellungen gekennzeichnet durch verschiedene Überschriften)

7.3 Funktionstests der ’wrapper.php’ und der ’creator.php’

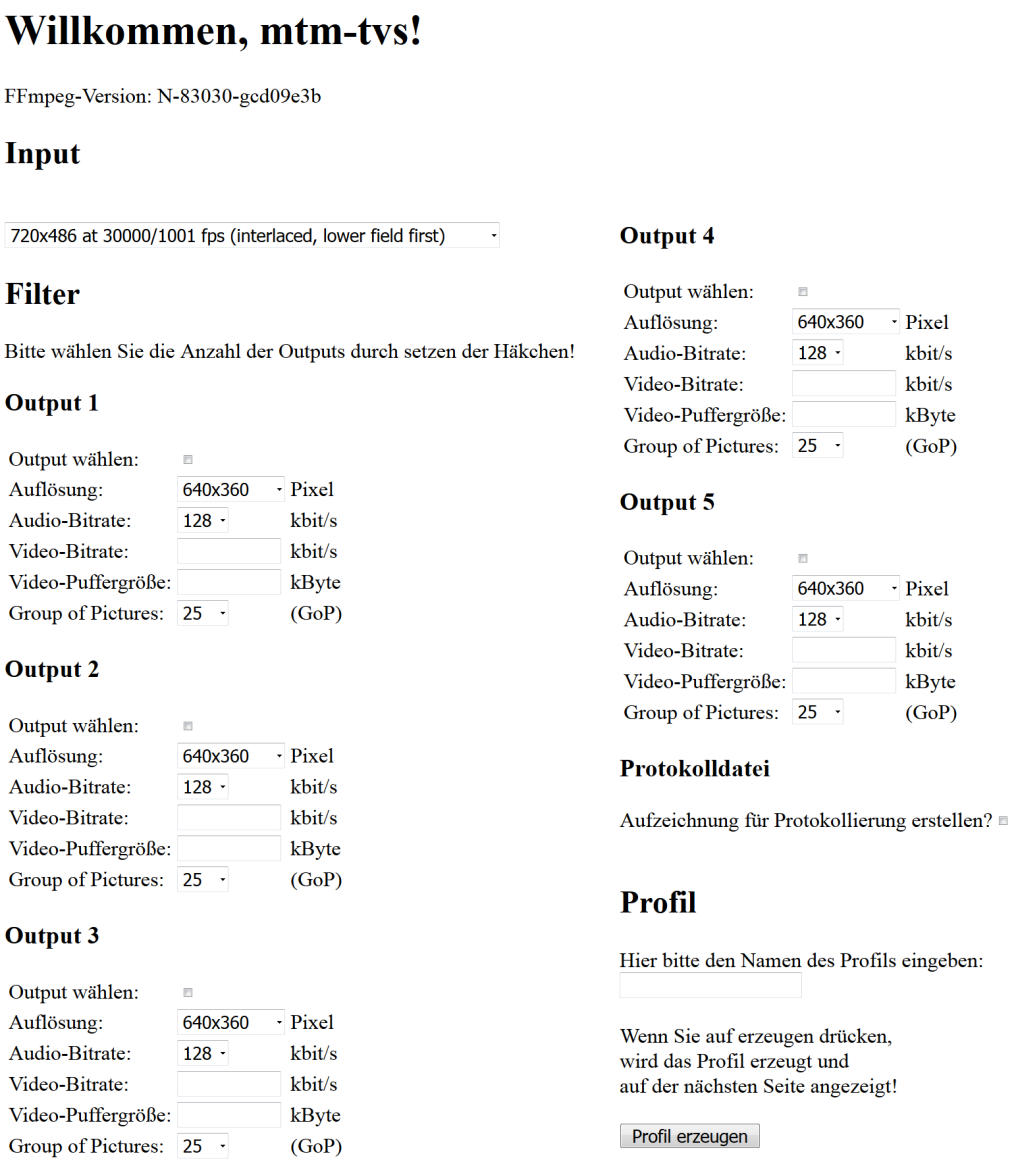

Abbildung 7.2: Der Prototyp der ’wrapper.php’, Screenshot

Die ’wrapper.php’ zeigt geordnet und wie im Entwurf vorgesehen die Kategorien Input, Filter, Output und die dazugehörigen Auswahllisten, Textboxen und Checkboxen an. (Es ist zu beachten, das der Screenshot bearbeitet wurde und auf der Originalseite alle Outputs untereinander angeordnet sind!)

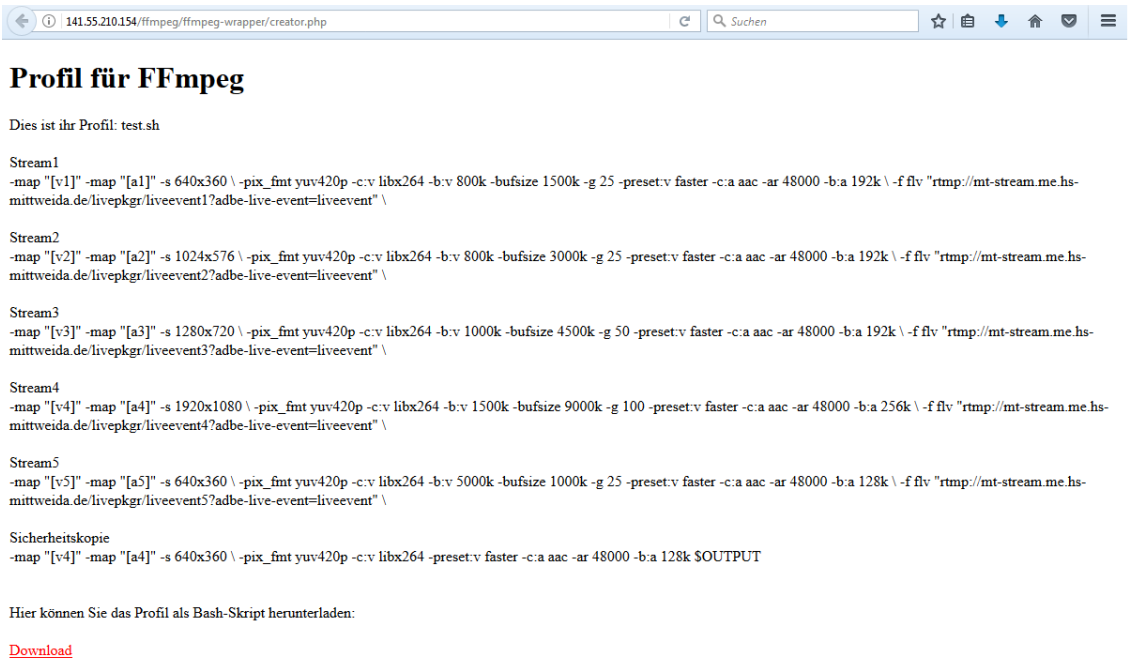

Abbildung 7.3: Der Prototyp der ’creator.php’, Screenshot

In der ’creator.php’ wird eine Zusammenfassung der Einstellungen angezeigt und es wird der Download des Skriptes bereitgestellt.

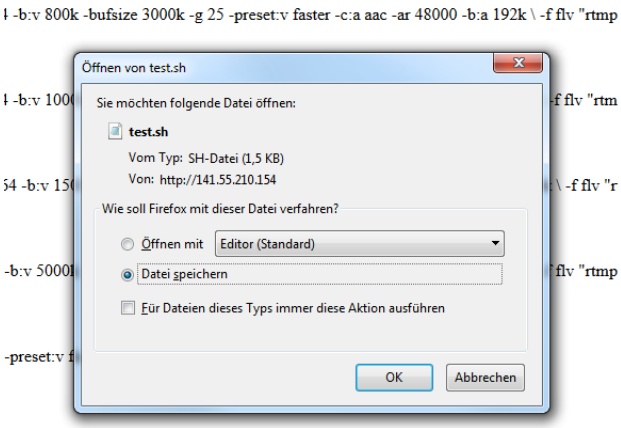

Abbildung 7.4: Dialogfenster zum Öffnen bzw. Speichern von ’test.sh’, Screenshot

Wie vorgesehen wird ein Dialogfenster geöffnet, wenn das Wort ’Download’ angeklickt wird. Der Benutzer kann sich jetzt, unabhängig vom Betriebssystem, das Skript herunterladen und speichern bzw. sofort in einem Editor öffnen.

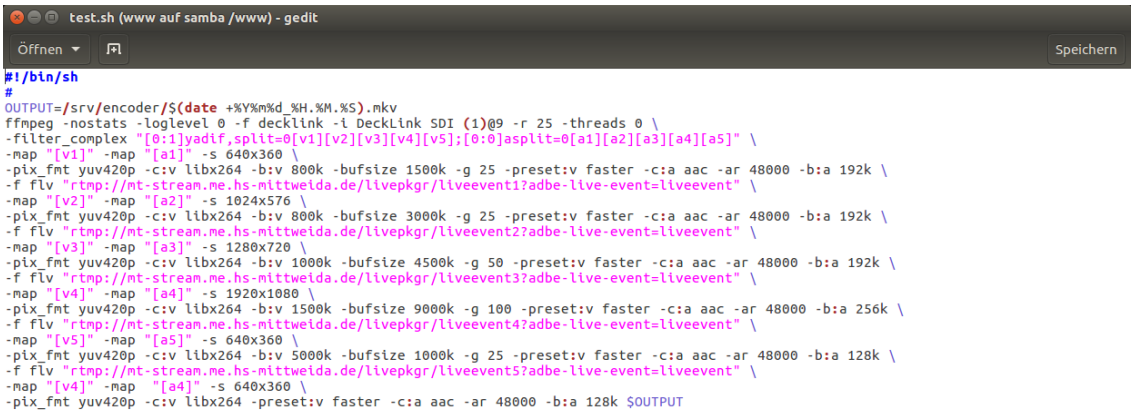

Abbildung 7.5: Das im Test ausgegebene Bash-Skript 'test.sh', Screenshot

Das Bash-Skript kann nach dem Download im Editor (hier auf einem Linux-System) angesehen werden, um gegebenfalls noch Anpassungen durchzuführen.

7.4 Auswertung

Der Entwurf wird übernommen und in die Webseite überführt. Die Kategorien 'Input', 'Filter' und 'Output' finden sich auf der grafischen Oberfläche wieder. Das Interface wird den funktionalen und nichtfunktionalen Anforderungen im Wesentlichen gerecht. Es ist jedoch einzuräumen, das es sich um einen Prototypen handelt und deshalb die Anzahl der gesetzten Anforderungen noch nicht so groß ist.

Die Überprüfung ergibt, das die Software die Aufgaben erfüllt, die angestrebt wurden. Das Webinterface bietet die wichtigsten Einstellungen für den Benutzer und kann im Rahmen seiner bisherigen Möglichkeiten genutzt werden.

8 Fazit

8.1 Zusammenfassung

Für das Webinterface liegt ein Prototyp vor, der die Stadien 'Analyse', 'Anforderungen', 'Entwurf', 'Implementierung' und 'Überprüfung' durchlaufen hat. Nichtsdestotrotz ist es nur ein Prototyp der die Grundlage für die Technologie bereitstellen soll, wie der Software-Encoder, die Capture-Karte und eine grafische Öberfläche zu einem funktionierenden System kombiniert werden können.

Gerade die Abschnitte 'Anforderungen' und 'Entwurf' stellen eine sinnvolle Grundlage für weitere Entwicklungen dar. Den größten Bedarf an Veränderung sieht der Bearbeiter jedoch nicht an den bereits vorhandenen Elementen der Software, sondern bei den Elementen, die bis dato noch nicht implementiert wurden. Auf mögliche Erweiterungen und die allgemeine Weiterentwicklung des Webinterfaces soll im Kapitel 'Ausblick' eingegangen werden.

Es wurde versucht, eine komfortable Benutzerobefläche für den Software-Encoder zu bieten, um ihn im Alltag besser nutzbar zu machen. Ob die Software diesen Anspruch erfüllt, kann und wird sich erst bei der regelmäßigen Nutzung herausstellen.

8.2 Ausblick

Als erste Erweiterung sollte eine Funktion für die Kategorie Output implementiert werden, sodass konsistent für jede Kategorie mindestens eine Klasse vorhanden ist, die die Basis für weitere Entwicklungen bieten kann. Diese Klasse sollte Funktionen bieten, um neben den Streams andere Ausgänge wählen zu können. Es wird vorgeschlagen, diese als ersten Schritt analog zu den Funktionen in der Kategorie Filter zu gestalten. Zunächst wird die Ausgabe einer einfachen Auswahlliste für jeden Filter auf dem Webinterface vorgeschlagen. Diese könnte mit vordefinierten Werte, wie zum Beispiel der Vorgabe von Stream, Dateipfad und/oder Dateityp realisiert werden.

Des weiteren sollte die „aktive Kommunikation“ zwischen dem Webinterface und der Linux-Umgebung weiter ausgebaut werden. Das heißt das Webinterface muss 'klüger' werden im Bezug auf die Möglichkeiten des Betriebssystem oder des Netzwerkes, in dem es benutzt wird. Bis jetzt werden nur die Capture-Karte bzw. ihre Optionen ausgelesen. Auf der gleichen technischen Basis, also mit regulären Ausdrücken und der Shell-Umleitung sollte auch die Mögliöchkeit gegeben werden, beispielsweise das Dateisys-

tem auslesen zu lassen, um gezielt Dateien aus bestimmten Verzeichnissen auswählen zu können. Mit Hilfe der PHP-Funktion 'exec' wird hier ein mächtiges Werkzeug geboten. Diese „aktive Kommunikation“ ermöglicht im Allgemeinen die gezieltere Bearbeitung des Bash-Skriptes und gibt noch mehr Möglichkeiten, es benutzerspezifischer zu machen.

Für die Kategorie Filter wird vorerst kein großes Potential gesehen, die Anzahl der Funktionen zu vergrößern. Weiterhin sollten aber im Hinblick auf mögliche Erweiterungen in der Kategorie Input weitere Einstellungen für andere Arten von Eingangssignalen vorgesehen werden.

Für die Darstellung des Webinterfaces wird empfohlen, die bisherige dezente Gestaltung beizubehalten. Der Einssatz von Farben wird nur für wichtige Hinweise bzw. bei der Eingabe von Fehlern vorgeschlagen. Die Darstellung ist bisher sehr vertikal ausgelegt und sollte aufgrund der horizontalen Ausrichtung des allgemein üblichen PC-Displays verstärkt in das Querfornmat überführt werden.

Betrachtet man den jetzigen Quellcode kritisch, besteht an diversen Stellen die Möglichkeit, einheitliche Prozesse in Funktionen zu integrieren. Es sollte bei der weiteren Entwicklung dementsprechend noch mehr objektorientiert implementiert werden, das heißt mehr Funktionen geschrieben werden, um proportional zum Umfang der Software den Quelltext zu vermindern.

Als langfristiges Ziel könnte auch die Möglichkeit bedacht werden, das Bash-Skript direkt über das Webinterface ausführen zu lassen. Das Webinterface könnte somit zu einer Art Fernbedienung für den Encoder ausgebaut werden. Ist dieser Schritt erst einmal getan, wäre auch die Entwicklung einer App denkbar, um die Steuerung unabhänig von einem PC durchführen zu können.

9 Danksagung

Mit der Abgabe dieser Diplomarbeit möchte ich die Gelegenheit nutzen und
die Menschen nennen, die mich in der Studienzeit und in meinem bisherigen Leben unterstützt haben und ohne die ich nicht an diesem Punkt angekommen wäre.

Meinen Eltern Elke Berger und Jürgen Berger,
meinem Bruder André Berger,
meinen Großeltern Helga Berger und Gottfried Berger,
und allen anderen Familienmitgliedern möchte ich Danke sagen!

Meiner Freundin Yulia Dolganova,
meinen Freunden Stefan Muhr und Patrick Weber,
meinen Kommolitonen Rico Domogalski, Anne-Katrin Dietz und Tommy Decker,
meinen Schulfreunden Sandra Dietz, Cindy Reißig, Mareen Hilbert,
Julia Mitev, Nicole Köhler, Sebastian Bär und Florian Fichtner,
und allen anderen aus meinem Freundeskreis möchte ich Danke sagen!

Ich möchte mich bedanken,
bei Frau Dipl.-Ing. Zimmer, für die gute Zusammenarbeit,
bei Herrn Dipl.-Ing. Jesch und Herrn Prof. Dr.-Ing. Zimmer,
für die geduldige Betreuung der Abschlussarbeit,
bei Herrn Dr.-Ing. Krupke, für die stete Hilfsbereitschaft,
bei Herrn Kilger, für die gute Handwerkskunst,
bei Herrn Prof. Dr.-Ing. Schmalwasser, für die stets ehrlichen und motivierenden Worte,
bei allen Mitgliedern der ehemaligen Fakultät Elektro- und Informationstechnik
und allen Mitgliedern der Hochschule, die uns im Laufe des Studiums geholfen haben!

“Man weiß erst, dass man ist, wenn man sich in anderen wiederfindet.“ [25]
- Johann Wolfgang von Goethe

[25] vgl. Goethe, Johann Wofgang von, 1775 [JoWoGo]

Anhang A: Quelltext

Listing A.1: class create_decklink_devices.php

```php
<?php
/**
 * Created by PhpStorm.
 * User: Stefan Berger
 * Date: 21.06.17
 * Time: 10:12
 */

class create_decklink_devices {

    ###############################################################
    # Konstanten
    ###############################################################

    const FFMPEGBIN = '/home/mtm-tvs/bin/ffmpeg';        # binary

    const DEVICEPATTERN = '/\[decklink @ 0x[0-9|a-f]{7}\]/';
    # Pattern from ffmpeg -f decklink -list_devices 1 -i dummy

    const VERSIONPATTERN = '/N-[0-9]{5}-[a-z|0-9]{8}/';
    # returns version string from ffmpeg -version

    const DEVICEFORMATPATTERN =
    '/\[decklink @ 0x[0-9|a-f]{7}\](.){2}[0-9]{1,2}/';
    # returns capture formats

    ###############################################################
    # Funktionen
    ###############################################################

        #### Funktion zum Auslesen von Arrays ####
    private function getlines($lines, $pattern) {
        foreach ($lines as $line) {
            if (preg_match($pattern, $line) === 1) {
                $out[] = $line;
            }
        }
        return $out;
    }

        #### Funktion zum Auslesen der FFmpeg-Version ####
    public function get_version($pattern=create_decklink_devices::
       VERSIONPATTERN) {
```

```
        exec(create_decklink_devices::FFMPEGBIN.' -version', $ret,
            $status);
        if (preg_match($pattern, $ret[0], $match) === 1) {
            return $match[0];
        }
        return false;
    }

        #### Funktion zum Auslesen der DeckLink-Geraete ####
    public function get_decklink_devices($pattern=
        create_decklink_devices::DEVICEPATTERN) {

        exec(create_decklink_devices::FFMPEGBIN.' -f decklink -
            list_devices 1 -i dummy 2>&1', $ret, $status);

        # $out gibt die Liste der Geraete aus
        $out = $this->getlines($ret, $pattern);

        # match for 'DeckLink SDI (x)' --> "\'(.+)'"
        $devicelist = "<select name='input_devices'>".PHP_EOL;
        foreach ($out as $line) {
            if (preg_match("/\'(.+)'/", $line, $match) === 1) {
                $devicelist .= "<optgroup label='".$match[1]."'>".
                    PHP_EOL;
                $devicelist .= $this->capture_formats($match[1]);
                $devicelist .= "</optgroup>".PHP_EOL;
            }
        }
        $devicelist .= "</select>".PHP_EOL;
        return $devicelist;
    }

    #### Funktion zum Auslesen der Aufnahmeformate ####
    public function capture_formats($device, $pattern=
        create_decklink_devices::DEVICEFORMATPATTERN) {

        exec(create_decklink_devices::FFMPEGBIN." -f decklink -
            list_formats 1 -i '$device' 2>&1", $ret, $status);
        $out = $this->getlines($ret, $pattern);
        $i = 1;
        # $paramlist = "<select name='capture_formats'>".PHP_EOL;
        $paramlist = "";
        foreach ($out as $line) {
            $paramlist .= "<option value='$device@$i'>".trim(
                preg_replace($pattern, '', $line))."</option>".
                PHP_EOL;
            $i++;
        }
        # $paramlist .= "</select>".PHP_EOL;
        return $paramlist;
    }
}
```

Listing A.2: class create_ffmpeg_filter

```php
<?php

/**
 * Created by PhpStorm.
 * User: Stefan Berger
 * Date: 21.06.17
 * Time: 09:41
 */
class create_ffmpeg_filter
{
    ################################################################
    # Vordefinierte Arrays
    ################################################################

    public $outputs = array('1', '2', '3', '4');
    public $res_video_list =
    array('640x360', '1024x576', '1280x720', '1920x1080');
    public $codec_video_list = array('libx264');
    public $bit_audio_list = array('128', '192', '224', '256');
    public $bit_video_list = array();
    public $gops_video =
    array('25', '50', '75', '100', '125', '150', '175', '200');

    ################################################################
    # Funktionen
    ################################################################

    #### Funktion zum Erzeugen der Kategorie Filter fuer alle
       Outputs ####

    public function filter_outputs2()
    {
        $border = 5;
        for ($i = 1; $i <= $border; $i++) {
            echo "<h3>Output " . $i . "</h3>";

            # Ausgabe als HTML-Tabelle #
            echo "<table>";
            echo "<tr><td><label>Output waehlen:</label></td>
            <td><input type='checkbox' id='mc'
            name='auswahl_output$i' value='auswahl'></td></td>
            <td></tr>" . PHP_EOL;

            echo "<tr><td><label>Aufloesung:</label></td>" .
               PHP_EOL;
            echo "<td><select name='filter_resolution$i' ";
            echo $this->filter_res() . " </td><td>Pixel</td></tr>"
               . PHP_EOL;

            echo "<tr><td><label>Audio-Bitrate:</label></td>" .
```

```
            PHP_EOL;
        echo "<td><select name='filter_bit_audio$i' ";
        echo $this->filter_bit_audio() . " </td><td>kbit/s</td
            ></tr>" . PHP_EOL;

        echo "<tr><td><label>Video-Bitrate:</label></td>" .
            PHP_EOL;
        echo "<td><input name='filter_bit_video$i' type='text'
            size='10'>";
        echo $this->filter_bit_video() . " </td><td>kbit/s</td
            ></tr>" . PHP_EOL;

        echo "<tr><td><label>Video-Puffergroesse:</label></td>"
             . PHP_EOL;
        echo "<td><input name='filter_buf_video$i' type='text'
            size='10'>";
        echo $this->filter_buf_video() . " </td><td>kByte</td
            ></tr>" . PHP_EOL;

        echo "<tr><td><label>Group of Pictures:</label></td>" .
             PHP_EOL;
        echo "<td><select name='filter_gops$i' ";
        echo $this->filter_gops() . "</td><td>(GoP)</td></tr>"
            . PHP_EOL;

        echo "</table>";

    }

    echo "<h3>Protokolldatei</h3>";

    echo "<label>Aufzeichnung fuer Protokollierung erstellen?
    </label><input type='checkbox' id='mc' name='
        auswahl_protokoll'
    value='protokoll'><br>" . PHP_EOL;

}

#### Funktion fuer Video-Aufloesung ####

public function filter_res()
{
    $out = ">" . PHP_EOL;
    foreach ($this->res_video_list as $item) {
        $out .= "<option value=$item>$item</option>" . PHP_EOL;
    }
    $out .= "</select>" . PHP_EOL;
    return $out;
}


#### Funktion fuer Bitrate Audio ####
```

```
public function filter_bit_audio()
{
    #$out = "<select name='filter_bit_audio'>".PHP_EOL;
    $out = ">" . PHP_EOL;
    foreach ($this->bit_audio_list as $item) {
        $out .= "<option value=$item>$item</option>" . PHP_EOL;
    }
    $out .= "</select>" . PHP_EOL;
    return $out;
}


#### Funktion fuer Bitrate Video ####

public function filter_bit_video()
{
    #echo "<label></label>";
    $out = "<br>" . PHP_EOL;
    if (100 <= $out and $out <= 15000 and is_numeric($out)) {
        return $out;
    }
    return $out = "<input name='filter_bit_video' type='text'
       value='Bitte Wert zw. 100 - 15000 eingeben.'>".PHP_EOL;
}


#### Funktion fuer Pufferrate Audio ####

public function filter_buf_video()
{
    $out = "<br>" . PHP_EOL;
    if (100 <= $out and $out <= 15000 and is_numeric($out)) {
        return $out;
    }
    return $out = "<input name='filter_buf_video' type='text'
       value='Bitte Wert zw. 100 - 15000 eingeben.'>".PHP_EOL;
}


#Funktion fuer Video-Codec

public function filter_codec_video()
{
    #$out = "<select name='filter_codec'>".PHP_EOL;
    foreach ($this->codec_video_list as $item) {
        $out = "<option value=$item>$item</option>" . PHP_EOL;
    }
    $out .= "</select>" . PHP_EOL;
    return $out;
}
```

```
    #Funktion fuer "Group of Pictures"

    public function filter_gops()
    {
        #$out = "<select name='filter_gops'>".PHP_EOL;
        $out = ">" . PHP_EOL;
        foreach ($this->gops_video as $item) {
            $out .= "<option value=$item>$item</option>" . PHP_EOL;
        }
        $out .= "</select>" . PHP_EOL;
        return $out;

    }

}
?>
```

Listing A.3: class create_ffmpeg_profil.php

```
<?php

/**
 * Created by PhpStorm.
 * User: mtm-tvs
 * Date: 21.06.17
 * Time: 14:31
 */
class create_ffmpeg_profil
{
        #
            ##################################################################

        # Konstanten
        #
            ##################################################################


    const FFMPEGBIN = '/home/mtm-tvs/bin/ffmpeg';      # binary
    const DEVICEPATTERN = '/\[decklink @ 0x[0-9|a-f]{7}\]/';
    # Pattern from ffmpeg -f decklink -list_devices 1 -i dummy
    const VERSIONPATTERN = '/N-[0-9]{5}-[a-z|0-9]{8}/';
    # returns version string from ffmpeg -version
    const DEVICEFORMATPATTERN = '/\[decklink @ 0x[0-9|a-f]{7}\](.)
       {2}[0-9]{1,2}/';
    # returns capture formats
    #const RESOLUTION = '720x486,720x576,1920x1080,1280x720';
    # (temporary) const for available formats
    #const RESOLPATTERN ='/[0-9]{3,4}x[0-9]{3,4}/';
    # returns resolutions ("SOMExSOME") out of string
    #const BITRATE = '128 | 192 | 224 | 256/';
    #const BITPATTERN = '/[0-9]{3}/';
```

```
#temporaerer regulaerer Ausdruck fuer filtern der Verzeichnisse
    aus Homeverzeichnis
const DIRECTORYPATTERN ='/(.)[A-W,a-w,_,Ä,Ü,Ö,ä,ü,ö]{3,12}\n/';

public $outputs = array('1','2','3','4');
public $res_video_list =
array('640x360','1024x576','1280x720','1920x1080');
public $codec_video_list = array('libx264');
public $bit_audio_list = array('128','192','224','256');
public $bit_video_list = array();
public $gops_video =
array('25','50','75','100','125','150','175','200');


#################################################################
# Funktionen
#################################################################

public function create() {
# textfeld fuer namen der datei vorsehen

# Dateiname

    $name = htmlspecialchars($_POST["datei_name"]);

    $myfile = fopen("'$name'.sh", "w") or die("Unable to open
       file!");
    ## hier ueber funktion den inhalt von bashskript einfuegen
       lassen
    $suff = '#!/bin/sh';
    $suff .= '#';
    $dir = 'OUTPUT=/srv/encoder/$(date +%Y%m%d_%H.%M.%S).mkv.\n
       '.PHP_EOL;

    ##Protyp fuer Anzahl Streams bzw Einstellungen fuer
    einzelnen Stream aus live.sh
    /*
    -filter_complex "[0:1]yadif,split=4[v1][v2][v3][v4];[0:0]
       asplit=4[a1][a2][a3][a4]" \
    -map "[v1]" -map "[a1]" -s 1280x720 \
    -pix_fmt yuv420p -c:v libx264 -b:v 4500k -bufsize 9000k
    -g 100 -preset:v faster -c:a aac -ar 48000 -b:a 224k \
    -f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/
    liveevent1?adbe-live-event=liveevent" \
    */
    ##

    #### if-abfrage der radio buttons und dann einfach jeweils
    ausgabe der outs??? bzw der v1, v2, v.... ####

    ## hier output variable einsetzen ...
```

```
## ffmpeg -nostats -loglevel 0 -f decklink -i
'DeckLink SDI (2)@9' -r 25 -threads 0 \

$in = 'ffmpeg -nostats -loglevel 0 -f decklink -i '.
htmlspecialchars($_POST["input_devices"]) .' -r 25
-threads 0 \ '.PHP_EOL;

## hier (ueber anzahl radio buttons???) anzahl der streams
einfuegen lassen

$all = '-filter_complex "[0:1]yadif,
split=4[v1][v2][v3][v4];
[0:0]asplit=4[a1][a2][a3][a4]" \ '.PHP_EOL;

## stream no1, variable fuer Aufloesung

$out1 = '-map "[v1]" -map "[a1]" -s '.
htmlspecialchars($_POST["filter_resolution1"]) .
 '\ '.PHP_EOL;

$out1 .= '-pix_fmt yuv420p -c:v libx264 -b:v '.
htmlspecialchars($_POST["filter_bit_video1"]).
'k -bufsize '.
htmlspecialchars($_POST["filter_buf_video1"]). 'k -g '.
htmlspecialchars($_POST["filter_gops1"]) .
' -preset:v faster -c:a aac -ar 48000 -b:a ' .
htmlspecialchars($_POST["filter_bit_audio1"]) .
 'k \ '.PHP_EOL;

$out1 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
livepkgr/liveevent1?adbe-live-event=liveevent" \ '.PHP_EOL;

## stream no2

$out2 = '-map "[v2]" -map "[a2]" -s '.
htmlspecialchars($_POST["filter_resolution2"]) .
 ' \ '.PHP_EOL;

$out2 .= '-pix_fmt yuv420p -c:v libx264 -b:v '.
htmlspecialchars($_POST["filter_bit_video2"]) .
'k -bufsize '.
htmlspecialchars($_POST["filter_buf_video2"]) . 'k -g '.
htmlspecialchars($_POST["filter_gops2"]) .
' -preset:v faster -c:a aac -ar 48000 -b:a ' .
htmlspecialchars($_POST["filter_bit_audio2"]) .
'k \ '.PHP_EOL;

$out2 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
livepkgr/liveevent2?adbe-live-event=liveevent" \ '.PHP_EOL;

## stream no3
```

```
$out3 = '-map "[v3]" -map "[a3]" -s '.
htmlspecialchars($_POST["filter_resolution3"]) .
' \ '.PHP_EOL;

$out3 .= '-pix_fmt yuv420p -c:v libx264 -b:v '.
htmlspecialchars($_POST["filter_bit_video3"]) .
'k -bufsize '.
htmlspecialchars($_POST["filter_buf_video3"]) . 'k -g '.
htmlspecialchars($_POST["filter_gops3"]) .
' -preset:v faster -c:a aac -ar 48000 -b:a ' .
htmlspecialchars($_POST["filter_bit_audio3"]) .
'k \ '.PHP_EOL;

$out3 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
livepkgr/liveevent3?adbe-live-event=liveevent" \ '.PHP_EOL;

## stream no4

$out4 = '-map "[v3]" -map "[a3]" -s ' .
htmlspecialchars($_POST["filter_resolution4"]) .
' \ '.PHP_EOL;

$out4 .= '-pix_fmt yuv420p -c:v libx264 -b:v '.
htmlspecialchars($_POST["filter_bit_video4"]) .
'k -bufsize '.
htmlspecialchars($_POST["filter_buf_video4"]) . 'k -g '.
htmlspecialchars($_POST["filter_gops4"]) .
' -preset:v faster -c:a aac -ar 48000 -b:a ' .
htmlspecialchars($_POST["filter_bit_audio4"]) .
'k \ '.PHP_EOL;

$out4 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
livepkgr/liveevent3?adbe-live-event=liveevent" \ '.PHP_EOL;

## stream no5

$out5 = '-map "[v3]" -map "[a3]" -s ' .
htmlspecialchars($_POST["filter_resolution5"]) .
' \ '.PHP_EOL;

$out5 .= '-pix_fmt yuv420p -c:v libx264 -b:v '.
htmlspecialchars($_POST["filter_bit_video5"]) .
'k -bufsize '.
htmlspecialchars($_POST["filter_buf_video5"]) . 'k -g '.
htmlspecialchars($_POST["filter_gops5"]) .
' -preset:v faster -c:a aac -ar 48000 -b:a ' .
htmlspecialchars($_POST["filter_bit_audio5"]) .
'k \ '.PHP_EOL;

$out5 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
livepkgr/liveevent3?adbe-live-event=liveevent" \ '.PHP_EOL;
```

```
        ## stream sicherheit

        $safe = '-map "[v4]" -map  "[a4]" -s 640x360 \ '.PHP_EOL;

        $safe .= '-pix_fmt yuv420p -c:v libx264 -preset:v faster
        -c:a aac -ar 48000 -b:a 128k $OUTPUT '.PHP_EOL;

        $inhalt = $suff . $dir . $in . $all .$out1 .
        $out2 . $out3 . $out4 . $out5 . $safe;
        fwrite($myfile, $inhalt);
        #$txt = "Jane Doe\n";
        #fwrite($myfile, $txt);
        fclose($myfile);

        return $myfile;

        #<form method="get" action="<?php echo
        $_SERVER['PHP_SELF']; ">

        echo "<a href='$myfile' download>Download</a>".PHP_EOL;

        #<button type="submit">Download!</button>
        #</form>

        #<button type="submit" onclick="window.open('file.doc')">
        Download!</button>

        }
        private function getlines($lines, $pattern) {
        foreach ($lines as $line) {
            if (preg_match($pattern, $line) === 1) {
                $out[] = $line;
            }
        }
        return $out;
        }

        private function profil_create_string() {
                $border = $_POST['filter_select_output'];

        #name='filter_resolution'
        #name='filter_bit_audio'
        #name='filter_bit_video'
        #name='filter_buf_video'
        #name='filter_codec
        #name='filter_gops'
        #name='auswahl_output'
        }
        # alte Funktionen

    # Funktion fuer Unterverzeichnisse im Home-Verzeichnis
    # oder diese gleich fuer Unter-Unterverzeichnisse +
```

```
        Unterverzeichnisse nehmen???
    #   Unterverzeichnis
    #       Unter-Unterverzeichnisse
    #       Unter-Unterverzeichnisse
    #       ...
    #   Unterverzeichnis
    #   ...

    # ueber type=file loesen
    #

    /*public function profil_directory_parent($pattern=
        create_ffmpeg_profil::DIRECTORYPATTERN) {
        #Verzeichnisse mit Unterverzeichnissen auslesen, ueber exec
            () und ls 2&<1 ????
        exec(' ls -l | grep 2>&1', $ret, $status);
        $out = $this->getlines($ret, $pattern);

        # match for 'DeckLink SDI (x)' --> "\'(.+)'"
        $devicelist = "<select name='input_devices'>".PHP_EOL;
        foreach ($out as $line) {
            if (preg_match("/\'(.+)'/", $line, $match) === 1) {
                $devicelist .= "<optgroup label='".$match[1]."'>".
                    PHP_EOL;
                $devicelist .= $this->capture_formats($match[1]);
                $devicelist .= "</optgroup>".PHP_EOL;
            }
        }
        $devicelist .= "</select>".PHP_EOL;
        return $devicelist;
    }
    */

    public function profil_create() {
    }
}
```

Listing A.4: wrapper.php

```
<?php
/**
 * Created by PhpStorm
 * User: Stefan Berger
 * Date: 21.06.17
 * Time: 13:57
 */

    #### Inkludierung der PHP-Klassen ####

include 'create_decklink_devices.php';
include 'create_ffmpeg_filter.php';
include 'create_ffmpeg_profil.php';

    #### Instanzierung ####

$blackmagic = new create_decklink_devices();
$ffmpegfilter = new create_ffmpeg_filter();
$ffmpegprofil = new create_ffmpeg_profil();

    #### Kopf der HTML-Seite ####

echo "<!DOCTYPE html>";
echo "<html>";
echo "<head>";
echo "<title>wrapper</title>";
echo "</head>";
echo "<body>";

echo "<h1>Willkommen, mtm-tvs!</h1>";

    #### Version von FFmpeg ####

echo "<label>FFmpeg-Version:  </label>";
echo $blackmagic->get_version();

    #### Input-Funktion ####

echo "<h2>Input</h2>";

    # Anfang HTML-Form #
echo "<form name='form' action='creator.php' method='POST'
target='_blank'><br>".PHP_EOL;
echo $blackmagic->get_decklink_devices();

    #### Filter-Funktion ####

echo "<h2>Filter</h2>";

echo "<label>Bitte wählen Sie die Anzahl der Outputs durch setzen
```

```
    der Häkchen!</label><br>";

echo $ffmpegfilter->filter_outputs2()."<br>".PHP_EOL;

    #### Profil-Name ####

echo "<h2>Profil</h2>";

echo "Hier bitte den Namen des Profils eingeben: <br>";
echo "<input type='text' name='datei_name'><br>";
echo "<br>";

    #### Profil erstellen ####

echo "Wenn Sie auf erzeugen drücken, <br>";
echo "wird das Profil erzeugt und <br>";
echo "auf der nächsten Seite angezeigt! <br><br>";
echo "<input type='submit' value='Profil erzeugen'><br>".PHP_EOL;

    # Ende HTML-Form #
echo "</form>";

echo "</body>";
echo "</html>";

?>
```

Listing A.5: creator.php

```php
<?php
/**
 * Created by PhpStorm
 * User: Stefan Berger
 * Date: 21.06.17
 * Time: 13:57
 */

    #### Kopf der HTML-Seite ####

echo "<!DOCTYPE html>";
echo "<html>";
echo "<head>";
echo "<title>creator</title>";
echo "</head>";
echo "<body>";

echo "<h1>Profil f\"ur FFmpeg</h1>";

    #### Variablenvordefinition ####

    $i = 0;          # Zaehlvariable
    $out1 = '';      # Stream 1
    $out2 = '';      # Stream 2
    $out3 = '';      # Stream 3
    $out4 = '';      # Stream 4
    $out5 = '';      # Stream 5
    $safe = '';      # Sicherheitskopie
    $dir = '';       # Verzeichnis bzw. Name Sicherheitskopie
    $v1 = '';        # Splitter Videosignal 1
    $v2 = '';        # Splitter Videosignal 2
    $v3 = '';        # Splitter Videosignal 3
    $v4 = '';        # Splitter Videosignal 4
    $v5 = '';        # Splitter Videosignal 5
    $a1 = '';        # Splitter Audiosignal 1
    $a2 = '';        # Splitter Audiosignal 2
    $a3 = '';        # Splitter Audiosignal 3
    $a4 = '';        # Splitter Audiosignal 4
    $a5 = '';        # Splitter Audiosignal 5

    #### Dateiname des Skriptes ####

    $name = htmlspecialchars($_POST["datei_name"]);
    # Bezeichnung fuer Skript aus wrapper.php per 'Post'
       uebertragen

    $script = $name.'.sh';
    # Dateiendung wird hinzugefuegt

    $myfile = fopen($script, "w") or die("Unable to open file!");
```

```
# Bash-Skript wird angelegt und geoeffnet

#### Kopf vom Skript ####

$suff = '#!/bin/sh'.PHP_EOL;
$suff .= '#'.PHP_EOL;
$dir = 'OUTPUT=/srv/encoder/$(date +%Y%m%d_%H.%M.%S).mkv'.
   PHP_EOL;

#### Protyp fuer Anzahl Streams bzw Einstellungen fuer
   einzelnen Stream aus live.sh ####
/**
-filter_complex "[0:1]yadif,split=4[v1][v2][v3][v4];
[0:0]asplit=4[a1][a2][a3][a4]" \
-map "[v1]" -map "[a1]" -s 1280x720 \
-pix_fmt yuv420p -c:v libx264 -b:v 4500k -bufsize 9000k -g 100
-preset:v faster -c:a aac -ar 48000 -b:a 224k \
-f flv "rtmp://mt-stream.me.hs-mittweida.de/livepkgr/
liveevent1?adbe-live-event=liveevent" \
*/

#### Input im Skript ####

$in = 'ffmpeg -nostats -loglevel 0 -f decklink -i '.
htmlspecialchars($_POST["input_devices"]) .'
-r 25 -threads 0 \ '.PHP_EOL;

#### Anzahl der Streams
#### Zeile kommt im Skript hier, durch Zaehlvariable aber
   Defintion von $all weiter unten
#$all = '-filter_complex "[0:1]yadif,split=4[v1][v2][v3][v4
   ];[0:0]asplit=4[a1][a2][a3][a4]" \ '.PHP_EOL;

#### Stream Nr. 1 im Skript ####

if (isset($_POST["auswahl_output1"])) {

    $out1 = '-map "[v1]" -map "[a1]" -s ' . htmlspecialchars(
       $_POST["filter_resolution1"]) . ' \ ' . PHP_EOL;

    $out1 .= '-pix_fmt yuv420p -c:v libx264 -b:v ' .
       htmlspecialchars($_POST["filter_bit_video1"]) . 'k -
       bufsize ' . htmlspecialchars($_POST["filter_buf_video1"
       ]) . 'k -g ' . htmlspecialchars($_POST["filter_gops1"])
       . ' -preset:v faster -c:a aac -ar 48000 -b:a ' .
       htmlspecialchars($_POST["filter_bit_audio1"]) . 'k \ ' .
        PHP_EOL;

    $out1 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
       livepkgr/liveevent1?adbe-live-event=liveevent" \ ' .
       PHP_EOL;
```

```
        $v1 = '[v1]';
        $a1 = '[a1]';
        $i =$i++;

    }

    #### Stream Nr. 2 im Skript ####

    if (isset($_POST["auswahl_output2"])) {

        $out2 = '-map "[v2]" -map "[a2]" -s ' . htmlspecialchars(
            $_POST["filter_resolution2"]) . ' \ ' . PHP_EOL;

        $out2 .= '-pix_fmt yuv420p -c:v libx264 -b:v ' .
            htmlspecialchars($_POST["filter_bit_video2"]) . 'k -
            bufsize ' . htmlspecialchars($_POST["filter_buf_video2"
            ]) . 'k -g ' . htmlspecialchars($_POST["filter_gops2"])
            . ' -preset:v faster -c:a aac -ar 48000 -b:a ' .
            htmlspecialchars($_POST["filter_bit_audio2"]) . 'k \ ' .
             PHP_EOL;

        $out2 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
            livepkgr/liveevent2?adbe-live-event=liveevent" \ ' .
            PHP_EOL;

        $v2 = '[v2]';
        $a2 = '[a2]';
        $i =$i++;

    }


    #### Stream Nr. 3 im Skript ####

    if (isset($_POST["auswahl_output3"])) {

        $out3 = '-map "[v3]" -map "[a3]" -s ' . htmlspecialchars(
            $_POST["filter_resolution3"]) . ' \ ' . PHP_EOL;

        $out3 .= '-pix_fmt yuv420p -c:v libx264 -b:v ' .
            htmlspecialchars($_POST["filter_bit_video3"]) . 'k -
            bufsize ' . htmlspecialchars($_POST["filter_buf_video3"
            ]) . 'k -g ' . htmlspecialchars($_POST["filter_gops3"])
            . ' -preset:v faster -c:a aac -ar 48000 -b:a ' .
            htmlspecialchars($_POST["filter_bit_audio3"]) . 'k \ ' .
             PHP_EOL;

        $out3 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
            livepkgr/liveevent3?adbe-live-event=liveevent" \ ' .
            PHP_EOL;

        $v3 = '[v3]';
```

```
        $a3 = '[a3]';
        $i = $i++;

    }


    #### Stream Nr. 4 im Skript ####

    if (isset($_POST["auswahl_output4"])) {

        $out4 = '-map "[v4]" -map "[a4]" -s ' . htmlspecialchars(
            $_POST["filter_resolution4"]) . ' \ ' . PHP_EOL;

        $out4 .= '-pix_fmt yuv420p -c:v libx264 -b:v ' .
            htmlspecialchars($_POST["filter_bit_video4"]) . 'k -
            bufsize ' . htmlspecialchars($_POST["filter_buf_video4"
            ]) . 'k -g ' . htmlspecialchars($_POST["filter_gops4"])
            . ' -preset:v faster -c:a aac -ar 48000 -b:a ' .
            htmlspecialchars($_POST["filter_bit_audio4"]) . 'k \ ' .
             PHP_EOL;

        $out4 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
            livepkgr/liveevent4?adbe-live-event=liveevent" \ ' .
            PHP_EOL;

        $v4 = '[v4]';
        $a4 = '[a4]';
        $i = $i++;
    }


    #### Stream Nr. 5 im Skript ####

    if (isset($_POST["auswahl_output5"])) {

        $out5 = '-map "[v5]" -map "[a5]" -s ' . htmlspecialchars(
            $_POST["filter_resolution5"]) . ' \ ' . PHP_EOL;

        $out5 .= '-pix_fmt yuv420p -c:v libx264 -b:v ' .
            htmlspecialchars($_POST["filter_bit_video5"]) . 'k -
            bufsize ' . htmlspecialchars($_POST["filter_buf_video5"
            ]) . 'k -g ' . htmlspecialchars($_POST["filter_gops5"])
            . ' -preset:v faster -c:a aac -ar 48000 -b:a ' .
            htmlspecialchars($_POST["filter_bit_audio5"]) . 'k \ ' .
             PHP_EOL;

        $out5 .= '-f flv "rtmp://mt-stream.me.hs-mittweida.de/
            livepkgr/liveevent5?adbe-live-event=liveevent" \ ' .
            PHP_EOL;

        $v5 = '[v5]';
        $a5 = '[a5]';
```

```
        $i = $i++;

    }

    #### Sicherheitskopie im Skript ####

    if (isset($_POST["auswahl_protokoll"])) {

        $safe = '-map "[v4]" -map  "[a4]" -s 640x360 \ ' . PHP_EOL;

        $safe .= '-pix_fmt yuv420p -c:v libx264 -preset:v faster -c
            :a aac -ar 48000 -b:a 128k $OUTPUT ' . PHP_EOL;

        $dir = 'OUTPUT=/srv/encoder/$(date +%Y%m%d_%H.%M.%S).mkv'.
            PHP_EOL;

    }

    #### Zusammensetzen des Splitter fuer die  Streams ####

    $all = '-filter_complex "[0:1]yadif,split='. $i . $v1 . $v2 .
        $v3 . $v4 . $v5 .';[0:0]asplit='. $i .$a1 . $a2 . $a3 . $a4
        . $a5 .'" \ '.PHP_EOL;

    #### Zusammensetzen des Inhalts fuer das Skript aus allen
        Abschnitten ####

    $inhalt = $suff . $dir . $in . $all .$out1 . $out2 . $out3 .
        $out4 . $out5 . $safe;

    #### Schreiben des Inhaltes in das Skript und schliessen des
        Skriptes ####
    fwrite($myfile, $inhalt);

    fclose($myfile);

    #### Anzeige der einzelnen Streams und des Downloads auf der
        Downloadseite ####

    echo "Dies ist ihr Profil:  ";
    echo "$script<br>";
    echo "<br>";

    echo "<label>Stream1</label><br>";
    echo "$out1<br>";

    echo "<br>";

    echo "<label>Stream2</label><br>";
    echo "$out2<br>";

    echo "<br>";
```

```
    echo "<label>Stream3</label><br>";
    echo "$out3<br>";

    echo "<br>";

    echo "<label>Stream4</label><br>";
    echo "$out4<br>";

    echo "<br>";

    echo "<label>Stream5</label><br>";
    echo "$out5<br>";

    echo "<br>";

    echo "<label>Sicherheitskopie</label><br>";
    echo "$safe<br>";

    echo "<br>";
    echo "<br>";

    echo "Hier können Sie das Profil als Bash-Skript herunterladen:
         ";

    echo "<br>";
    echo "<br>";

    echo "<a style='color: red' href='$script' download>Download</a
       >".PHP_EOL;

    echo "</body>";
    echo "</html>";

    ?>
```

Anhang B: Anleitung

In dieser Anleitung soll beschrieben werden, wie man ein funktionsfähiges System für den Betrieb der Software in Kombination mit einer Blackmagic DeckLink Capture-Karte aufbaut. Diese wurde anhand der gewonnen Erfahrungen und aussagekräftiger Quellen erstellt. [26] [27] Nachfolgend soll zunächst die wichtigste Hard- und Software aufgelistet werden, die für den Aufbau benötigt wird.

Software:

Betriebssystem Ubuntu
Apache Webserver
PHP, als Apache-Modul
FFmpeg, mit Blackmagic DeckLink SDK
Software 'FFmpeg-Wrapper' ('wrapper.php','creator.php' und die 3 PHP-Klassen)

Hardware:

Intel Core i5- bzw. i7-Prozessor
Netzwerkanschluss
Blackmagic-DeckLink 2 Duo Capture-Karte

Des Weiteren setzt diese Anleitung folgende Anforderungen vorraus:
Die Capture-Karte wurde bereits im PC eingebaut und auf der Festplatte befinden sich weder wichtige Daten noch ein bereits installiertes Betriebssystem. Für den Download benötigt man außerdem einen funktionsfähigen PC mit der Software „SD-Formatter“ und „Win32 Disk Imager“ und einen USB-Stick als Speichermedium, um das Image des Betriebssystem darauf zu schreiben.

0. Der USB-Stick muss mit Hilfe einer speziellen Software formatiert werden. Dafür kann die kostenlose Software „SD-Formatter“ genutzt werden. Diese lässt sich z.B. von der Internetseite 'https://www.heise.de/download/product/sd-formatter-74314' herunterladen. Nach erfolgreicher Installation wählt man den USB-Stick in dieser Software aus der Liste der Laufwerke aus und formatiert ihn.

1. Das Betriebssystem Ubuntu kann von der Seite „https://www.ubuntu.com/desktop“ heruntergeladen werden. Man benötigt eine spezielle Software, um das Image von Ubuntu auf den formatierten USB-Stick zu kopieren und um aus ihm einen bootfähigen

[26] vgl. Herr Dipl.-Ing. Birger Jesch, 2017 [BJeschAnl]
[27] vgl. DigitalOcean(TM) Inc., 2016 [DigOcLam]

Datenträger zu machen. Dafür wird die Software „Win32 Disk Imager“ genutzt, welche von der Internetseite ’https://www.heise.de/download/product/win32-disk-imager-92033/download’ heruntergeladen werden kann. Bei ihr wählt man den USB-Stick als Medium und beschreibt ihn mit dem Image.

2. Der USB-Stick wird an das neue System angeschlossen. Nach dem Anschalten sollte der PC direkt vom USB-Stick booten und es wird automatisch der Installationsassistent gestartet. (Sollte der PC nicht automatisch vom USB-Stick booten, muss im BIOS die Bootreihenfolge dementsprechend angepasst werden.)

3. Im Installationsassistenten werden alle benutzerspezifischen Einstellungen vorgenommen. Sollte sich ein weiteres Betriebssystem auf dem PC befinden, kann man wählen, ob dieses gelöscht oder Ubuntu parallel dazu installiert wird.

4. Wenn Ubuntu erfolgreich installiert ist und man den Desktop vor sich hat, sollte man als ersten Schritt die Konsole öffnen. Hier gibt man den Befehl „sudo apt-get update“ ein, um seine Ubuntu-Version auf den neuesten Stand bezüglich der Betriebssystemupdates zu bingen.

5. Die Installation des Apache Webservers erfolgt ebenfalls über die Konsole. Mit dem Befehl „sudo apt-get install apache2“ wird der Webserver installiert.

6. Nun muss in der Konfigurationsdatei von Apache eine Zeile mit dem Servernamen, also der IP-Adresse des Servers, hinzugefügt werden. Dafür muss man die Datei mit dem Befehl „sudo nano /etc/apache2/apache2.conf“ im Editor öffnen und am Ende die Zeile „ServerName [IP-Adresse des PCs]“ einfügen. Diese Datei wird nun gespeichert und wieder geschlossen.

7. Die Konfiguration des Webservers wird mit der Eingabe „sudo apache2ctl configtest“ getestet: es muss die Ausgabe „Syntax OK“ erscheinen.

8. Der Webserver wird jetzt über die Konsole mit dem Befehl „sudo systemctl restart apache2“ neu gestartet, um die Änderungen wirksam zu machen. Man kann die erfolgreiche Installation testen, in dem man in einem Browser die Testseite von Apache mit der Adresse „[IP-Adresse des PCs]/index.html“ aufruft.

9. PHP wird nun als Modul von Apache installiert und ebenfalls über die Konsole realisiert. Der Befehl lautet „ sudo apt-get install php libapache2-mod-php php-mcrypt“. Nun wird sowohl PHP als auch das passende Modul für Apache installert.

10. Auch bei PHP kann man testen, ob die Installation erfolgreich war. Hierzu erstellt man mit dem Befehl „sudo nano /var/www/html/info.php“ in der „web root“ eine neue php- Datei, in die folgender Inhalt geschrieben wird:

```
<?php
phpinfo();
?>
```

Auch diese Datei muss nun gespeichert und wieder geschlossen werden. Bei erfolgreicher Installation wird nach Eingabe der Adresse „[IP-Adresse des PCs]/info.php“ die PHP-Testseite angezeigt.
Hinweis: ab Ubuntu 16.04 wird die PHP-Version 7 benötigt.

11. Mit der erneuten Eingabe von „sudo apt-get update“ werden aktuelle Updates heruntergeladen und mit dem Befehl „sudo apt-get install unzip dkms software-properties-common“ wird das Dynamic Kernel Modul Support installiert.

12. Nun wird das Blackmagic DeckLink SDK (’Software Development Kit’) heruntergeladen und entzippt. Dafür gibt man folgende Befehle in die Konsole ein:

```
unzip BMDL_SDK 10.6.6.ZIP -d .
rm BMDL_SDK 10.6.6.ZIP
```

14. Für Blackmagic werden weitere Treiber benötigt. Auch diese werden geladen und installiert.

```
tar -xzf Blackmagic_Desktop_Video_Linux_10.6.6.tar.gz
cd Blackmagic_Desktop_Video_Linux_10.6.6/deb/amd64
sudo dpkg -i desktopvideo_10.6.6a10_amd64.deb
sudo apt-get install -f
```

15. Für die Installation von FFmpeg sind folgende Eingaben nötig.

```
sudo apt-get update
sudo apt-get -y install autoconf automake build-essential libass-dev libfreetype6-dev
libtheora-dev libtool libvorbis-dev libxcb1-dev pkg-config texinfo zlib1g-dev

cd
mkdir ffmpeg_sources

sudo apt-get install yasm libx264-dev libmp3lame-dev libopus-dev

cd  /ffmpeg_sources
wget -O fdk-aac.tar.gz https://github.com/mstorsjo/fdk-aac/tarball/master
tar xzvf fdk-aac.tar.gz
cd mstorsjo-fdk-aac*
autoreconf -fiv
./configure –prefix=“$HOME/ffmpeg_build“ –disable-shared
```

```
make
make install
make distclean


cd  /ffmpeg_sources
wget http://ffmpeg.org/releases/ffmpeg-snapshot.tar.bz2
tar xjvf ffmpeg-snapshot.tar.bz2
cd ffmpeg
PATH="$HOME/bin:$PATH" PKG_CONFIG_PATH="$HOME/ffmpeg_build/lib/pkgconfig"
./configure
–prefix="$HOME/ffmpeg_build"
–pkg-config-flags="–static"
–extra-cflags="-I$HOME/ffmpeg_build/include"
–extra-ldflags="-L$HOME/ffmpeg_build/lib"
–bindir="$HOME/bin"
–enable-gpl
–enable-libass
–enable-libfdk-aac
–enable-libfreetype
–enable-libmp3lame
–enable-libopus
–enable-libtheora
–enable-libvorbis
–enable-libx264
–enable-nonfree
–enable-decklink
–extra-cflags=-I/home/mtm-tvs/BMDL_SDK_10.6.6/Linux/include
–extra-ldflags=-L/home/mtm-tvs/BMDL_SDK_10.6.6/Linux/include


PATH= „$HOME/bin:$PATH" make
make install
make distclean
hash -r


source  /.profile
```

16. Nach der Kompilierung sollte getestet werden, ob alle Elemente erfolgreich installiert wurden. (siehe auch https://www.ffmpeg.org/ffmpeg-devices.html#toc-decklink)

```
ffmpeg -f decklink -list_devices 1 -i dummy
ffmpeg -f decklink -list_formats 1 -i 'DeckLink SDI (1)'
```

17. Um die Blackmagic DeckLink Capture Karte zu testen, kann man mit folgendem Befehl ein 10 sekündiges Testvideo erstellen.

ffmpeg -f decklink -i 'DeckLink SDI (2)@9' -t 10 -s 1280:720 -vcodec h264
-crf 20 -r 25 -acodec aac -b:a 192k output.mkv

18. Des Weiteren kann man mit folgendem Befehl das RTMP-Streaming (Real Time Messaging Protocol) testen.

ffmpeg -f decklink -i 'DeckLink SDI (2)@9' -s 1280:720 -c:v libx264 -preset fast -pix_fmt yuv420p -c:a aac -b:a 192k -ar 48000 -threads 0 -f flv „rtmp://mt-stream.me.
hs-mittweida.de/livepkgr/liveevent1?adbe-live-event=liveevent “

19. Als letzten Schritt muss man die Dateien des FFmpeg-Wrappers, also den gesamten Ordner 'ffmpeg-wrapper' in das Verzeichnis des Webservers '/var/www/html/' kopieren, um darauf zugreifen zu können. Nutzt man nun dieses neue System oder einen PC im selben Netzwerk und gibt dort im Browser die Adresse „[IP-Adresse des wrappers]/ffmpeg-wrapper/wrapper.php“ ein, kann man die Software nutzen.

Literaturverzeichnis

[AmA140] Amino Communications: „ViewCDA140“ (2017),
URL: https://www.aminocom.com/products/amino-view/client-devices/a140
(Stand: 13.06.17)

[BlDesAbo] Blackmagic Design Pty. Ltd.: „Informationen über uns“ (2017),
URL: https://www.blackmagicdesign.com/de/company/aboutus (Stand: 13.06.17)

[BlDesPro] Blackmagic Design Pty. Ltd.: „Produkte“ (o.J.),
URL: https://www.blackmagicdesign.com/de/products (Stand: 13.06.17)

[BJeschAnl] Herr Dipl.-Ing. Jesch, Birger: „Linux_ffmpeg_kompilieren“,
Mittweida, HS Mittweida, Fakultät Medien, 2017

[BJeschLive] Dipl.-Ing. Jesch, Birger: „live.sh“,
Mittweida, HS Mittweida, Fakultät Medien, 2017

[BJeschTh] Dipl.-Ing. Jesch, Birger: „Thema Abschlussarbeit“,
Mittweida, HS Mittweida, Fakultät Medien, 2017

[DictWrap] Ing. Hemetsberger, Paul: „wrapper“ (2017),
URL: http://www.dict.cc/englisch-deutsch/wrapper.html (Stand: 13.06.17)

[DigOcLam] DigitalOcean(TM) Inc.: „How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu 16.04“ (21.04.2016),
URL: https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-16-04 (Stand: 13.06.17)

[ElKoPcie] Elektronik-Kompendium.de: „PCIe - PCI Express(1.1 / 2.0 / 3.0 / 4.0)“ (o.J.),
URL: https://www.elektronik-kompendium.de/sites/com/0904051.htm
(Stand: 13.06.17)

[FFAbo] FFmpeg project: „ About FFmpeg “ (o.J.),
URL: https://ffmpeg.org/about.html (Stand: 13.06.17)

[FFLib] FFmpeg project: „ FFmpeg Libraries for developers “ (o.J.),
URL: https://ffmpeg.org/about.html (Stand: 13.06.17)

[FFPic] FFmpeg project: „FFmpeg Logo“ (o.J.),
URL:https://trac.ffmpeg.org/ (Stand: 13.06.17)

[FFTool] FFmpeg project: „ FFmpeg Tools “ (o.J.),
URL: https://ffmpeg.org/about.html (Stand: 13.06.17)

[ItWissWrap] DATACOM Buchverlag GmbH: „Wrapper“ (01.11.2013),
URL: http://www.itwissen.info/Wrapper-wrapper.html (Stand: 13.06.17)

[ItAgWass] itemis AG: „Wasserfallmodell“ (o.J.),
URL:https://blogs.itemis.com/de/scrum-kompakt-wasserfallmodell (Stand: 13.06.17)

[JoWoGo] Goethe, Johann Wolfgang von: „Briefe. An Auguste Gräfin zu Stolberg“,
13. Februar 1775

[LyTeYell] LYNXTechnikAG: „CHD 1812“ (o.J.),
URL: http://www.lynx-technik.com/products/yellobrik/video-conversion/chd-1812-hdmi-to-sdi-converter-with-frame-synchronizer/ (Stand: 13.06.17)

[PhpExec] The PHP Group: „exec“ (2017),
URL: http://php.net/manual/de/function.exec.php (Stand: 13.06.17)

[PhpPreg] The PHP Group: „ preg_match “ (2017),
URL: http://php.net/manual/de/function.preg-match.php (Stand: 13.06.17)

[PicBlaDec] picturetools.de: „Blackmagic DeckLink Duo 2“ (2017),
URL:https://www.picturetools.de/Hersteller/Blackmagic_Design/Blackmagic_DeckLink_Karten (Stand: 13.06.17)

[SelHtRegAus] SELFHTML-Wiki: „Perl/Reguläre Ausdrücke“ (25.05.2017),
URL:https://wiki.selfhtml.org/wiki/Perl/Reguläre_Ausdrücke (Stand: 13.06.17)

[UbStor] Canonical Ltd.: „The Ubuntu story“ (2017),
URL:https://www.ubuntu.com/about/about-ubuntu (Stand: 13.06.17)

[UbUsDae] ubuntuusers.de: „Dienste“ (2017),
URL: https://wiki.ubuntuusers.de/Dienste/ (Stand: 13.06.17)

[UbUsLts] ubuntuusers.de: „Long Term Support“ (2017),
URL: https://wiki.ubuntuusers.de/Long_Term_Support/ (Stand: 13.06.17)

[UbUsUml] ubuntuusers.de: „Umleitungen“ (2017),
URL:https://wiki.ubuntuusers.de/Shell/Umleitungen/ (Stand: 13.06.17)

[WikAnSof] Wikipedia, die freie Enzyklopädie: „Anforderung (Informatik)“ (22.06.2017), URL: https://en.wikipedia.org/wiki/Anforderung_(Informatik) (Stand: 30.05.17)

[WikBlDes] Wikipedia, die freie Enzyklopädie: „ Blackmagic Design “ (22.06.2017), URL:https://en.wikipedia.org/wiki/Blackmagic_Design (Stand: 22.06.17)

[WikPci] Wikipedia.org: „PCI Express“ (o.J.), URL:https://de.wikipedia.org/wiki/PCI_Express (Stand: 13.06.17)

[WikSerInt] Wikipedia, die freie Enzyklopädie: „ Serial Digital Interface “ (07.07.2017), URL:https://de.wikipedia.org/wiki/Serial_Digital_Interface (Stand: 07.07.17)

[WirtLexBot] Springer Fachmedien Wiesbaden GmbH: „ Bottom-up-Prinzip “ (o.J.), URL: http://wirtschaftslexikon.gabler.de/Definition/bottom-up-prinzip.html (Stand: 13.06.17)

[WirtLexTop] Springer Fachmedien Wiesbaden GmbH: „Top-Down-Prinzip“ (o.J.), URL:http://wirtschaftslexikon.gabler.de/Definition/top-down-prinzip.html (Stand: 13.06.17)

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 31.07.2017