Dateiverwaltung für eine Spielproduktion

Lukas Paul

BACHELORARBEIT

Nr. 1510238050-A

eingereicht am
Fachhochschul-Bachelorstudiengang

Medientechnik und -design

in Hagenberg

im Februar 2018

Diese Arbeit entstand im Rahmen des Gegenstands

Bachelorarbeit

im

Wintersemester 2017

Betreuung:

Wolfgang Hochleitner BSc MSc

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 8. Februar 2018

Lukas Paul

Inhaltsverzeichnis

Erklärung iii

Kurzfassung vi

Abstract vii

1 Einleitung 1
- 1.1 Problemstellung . . . 1
- 1.2 Aufbau der Arbeit . . . 1
- 1.3 Begriffsdefinitionen . . . 2

2 Anforderungen 4
- 2.1 Existierende Software . . . 4
- 2.2 Kosten . . . 4
- 2.3 Unterstütze Funktionen . . . 4
- 2.4 Hosting Art . . . 5
- 2.5 Dateiformate . . . 5
- 2.6 Anzahl/Größe Assets . . . 6
- 2.7 Wiederverwendungsgrad . . . 6

3 Nutzen von digitalem Asset Management 7
- 3.1 Vorteile . . . 7
  - 3.1.1 Generelle Vorteile . . . 7
  - 3.1.2 Individuelle Vorteile . . . 9
- 3.2 Stand der Technik . . . 10

4 Konzept 12
- 4.1 Konvertieren . . . 12
- 4.2 Metadaten . . . 12
- 4.3 Sammlungen . . . 13
- 4.4 Benennung von Dateien . . . 14
- 4.5 Ordner? . . . 14
- 4.6 Tags . . . 14
- 4.7 Hierarchien für Ordner/Tags . . . 14
- 4.8 Ratings . . . 15
- 4.9 Links zu Assets teilen . . . 15

4.10 Suche . . . 15
4.11 Plug-in System für Dateiformate . . . 15
4.12 Automatische Analyse . . . 16
4.12.1 Gesichtserkennung . . . 16
4.12.2 Texterkennung . . . 16
4.12.3 Automatisches hinzufügen von Tags . . . 17

5 Umsetzung **19**
5.1 Design . . . 19
5.2 Architektur . . . 19
5.2.1 Frontend . . . 20
5.2.2 Backend . . . 21
5.3 Tag Editor . . . 22
5.4 Statusbar . . . 22
5.5 Unterstütze Dateitypen . . . 22
5.6 Assets Verarbeitung . . . 22
5.6.1 Import von neuen Assets . . . 24
5.7 Assets Anzeige . . . 25
5.7.1 Transparente Bilder . . . 26

6 Schluss **28**

A Fragebogen **29**

Quellenverzeichnis **32**
Literatur . . . 32
Audiovisuelle Medien . . . 32
Online-Quellen . . . 32

Kurzfassung

Der Anteil an wiederverwendbaren Dateien in der Spielentwicklung ist vergleichsweise hoch zu anderen Branchen, die DAMs (Digitales Asset Management) verwenden können. Darüber hinaus nehmen Versionierung, Filterung und Sortierung einen großen Stellenwert in der Spielentwicklung ein. Diese Bachelorarbeit soll näher auf die Unterschiede und Möglichkeiten verschiedener DAMs eingehen und eine eigene Implementierung breit stellen.

Abstract

The percentage of reusable files in game development are relatively high compared to other industries which can use DAMs (Digital Asset Management). Furthermore versioning, filtering, and ordering play an essential role in game development. This bachelor thesis elaborates the differences and possibilities of different DAMs and introduces a custom implementation.

Kapitel 1

Einleitung

Die Grafik von modernen Spielen hat sich über die letzten Jahre an absoluten Realismus angenähert, selbst Spiele deren Grafik einfacher gehalten wird wie Fortnite[1] haben einen viel höheren Detailgrad als vor ein paar Jahren. Durch diesen steigenden Anspruch müssen stetig mehr Assets in der Spielentwicklung verwaltet werden [2, S. 35, S.135][1].

Diese Bachelorarbeit soll auf die unterschiedlichen Möglichkeiten der Asset Verwaltung eingehen, welche Funktionen dafür nötig sind und stellt darüber hinaus einen eigenen Lösungsansatz vor. Der Lösungsansatz basiert auf den Erkenntnissen gewonnen in dieser Bachelorarbeit und einer Umfrage, die auch für diese Bachelorarbeit mit österreichischen Spielefirmen durchgeführt wurde.

1.1 Problemstellung

Durch die Vielzahl an Assets, die Spielefirmen und Firmen mit komplexen Projektstrukturen erstellen, wird es umso wichtiger Assets so abzulegen, dass sie auch wieder gefunden werden können. Oft kann man sich an ein bestimmtes Asset erinnern, die Windowssuche liefert aber nur „Es wurden keine Suchergebnisse gefunden.“ zurück. Weiters kann es bei der Arbeit mit mehreren Kollegen dazu kommen, dass Arbeit doppelt gemacht wird, von Kollegen überschrieben wird oder ganz verloren geht. Diese Probleme treten insbesondere auf, wenn Mitarbeiter Assets nur mit Hilfe des Dateisystems verwalten. Die fehlende Benutzerverwaltung und Versionierung für Assets macht ihn fehleranfälliger als Systeme die diese Funktionen bieten, da Mitarbeiter nur Rechte für Assets besitzen die sie auch verändern sollen. Darüber hinaus können alte Versionen leicht wieder hergestellt werden [2, S. 9].

1.2 Aufbau der Arbeit

Im Kapitel 2 wird auf die generellen Bedürfnisse die mit einer Umfrage bei Spielfirmen ermittelt wurden eingegangen. Im Kapitel 3 wird auf die allgemeinen Vorteile die ein PAM für ein Projekt/Unternehmen hat und auf den aktuellen Stand der beliebtesten [13] DAM Systeme eingegangen. Das Kapitel 4 baut auf dem Kapitel 3 auf und stellt

[1] https://www.epicgames.com/fortnite

weiter Funktionen die ein PAM besitzen sollte vor. Im Kapitel 5 wird auf den eigens für diese Bachelorarbeit produzierten Prototypen eingegangen.

1.3 Begriffsdefinitionen

Im folgenden Abschnitt werden Begriffe, die in dieser Bachelorarbeit verwendet werden und eventuell eine Erklärung oder eine präzisiere Beschreibung in Hinsicht auf das Digitale Asset Management brauchen, beschrieben.

Asset

Das Wort stammt ursprünglich aus der Finanzwelt, in der es etwas von einem bestimmten Wert beschreibt, wie ein Grundstück, Aktien oder ein Kunstwerk. In unserem Fall beschreibt es eine Datei, für die wir als Unternehmen die Rechte zur Verwendung besitzen und die Metadaten, die diese Datei beschreiben. Die Metadaten können so rudimentär wie der Name oder das Erstellungsdatum sein aber auch komplexer wie Tags, die den Inhalt der Datei beschreiben [2, S. 2].

Tag

Ein Tag ist ein Attribut, das man einem Asset gibt. Beispiele sind,

- das Genre zu einem Lied,
- die Art der Lizenz zu einem Bild,
- der Inhalt eines Bildes (es ist eine Textur oder es kommen Autos darauf vor).

Digital Asset Manager (DAM)

Beschreibt im weiteren Sinn das Ordnen und Ablegen von Assets auf eine für den Betrieb sinnvolle Art und Weise. Die zugänglichste Methode ist eine gute Ordner- und Dateistruktur mit aussagekräftigen Namen. Durch den steigenden Verwaltungsaufwand bei gleichzeitiger Bearbeitung von vielen Assets werden schon mit wenigen Mitarbeitern mehr Funktionen benötigt, als jedes Dateisystem bieten kann. Für diese Anforderungen gibt es extra erstellte Programme, um Benutzer beim Sortieren zu unterstützen (Kapitel 3). Der Begriff DAM kann somit in vielen Bereichen verwendet werden, sei es bei der normalen Asset Verwaltung im Explorer, dem Speichern in einem extra Programm, dem erstellen von einem Katalog mit allen Assets für das Branding oder dem Ablegen in einem Content Management System (CMS) [2, S. 3].

Produktion Asset Manager (PAM)

Um den weit gefassten Bereich eines DAMs zu spezifizieren gibt es den Begriff des PAMs. Assets durchlaufen in einem Projekt mehrere Stadien, von Konzept bis zur konkreten Fertigstellung. Diese Stadien werden einem Task zugewiesen, der an verschiedene Mitarbeiter vergeben werden kann. Sind alle Task eines Projekts abgeschlossen ist auch das Projekt abgeschlossen. PAM beschäftigt sich somit nicht nur mit den Assets sondern mit den Assets im Rahmen ihres Erstellungsprozesses. In dieser Bachelorarbeit wird

hauptsächlich auf PAMs eingegangen aber da diese zur Familie der DAMS gehören, werden beide Begriffe verwendet [2, S. 25].

Sammlung

Eine Gruppe/Sammlung von Assets die entweder automatisch mit Filterkriterien oder händisch erstellt wurde.

Kontrolliertes Vokabular

Ein Feld kann nur die vorgegebenen Werte annehmen. Beispiel dafür ist, es kann bei dem Feld Fahrzeugtype nur Motorrad, Auto und LKW ausgewählt werden.

Convention over configuration

Ist ein Software Design Pattern, das versucht die Entscheidungen die ein Entwickler treffen muss zu reduzieren, ohne ihm die Möglichkeit zu nehmen später einzugreifen.

Entity

Ein Entity ist eine Klasse wie eine Person oder ein Asset. Der Begriff wird besonders in Zusammenhang mit Datenbanken und ihren Tabellen verwendet.

DbSet

Ein DbSet ist eine Sammlung von Entities die auf der Datenbank mit Hilfe des Entity Frameworks[2] gespeichert werden.

Repository

Unterstützt beim Datenbankzugriff. Stellt Funktionen für Lesen, Schreiben, Bearbeiten und Löschen von Entities über die dazugehörigen DbSets zur Verfügung.

Thumbnail

Vorschaubild für ein Asset, generiert um dieses nicht aufmachen zu müssen.

[2] https://docs.microsoft.com/en-us/ef/

Kapitel 2

Anforderungen

Dieses Kapitel beschreibt die Anforderungen verschiedener Spielfirmen, die sich aus den Fragen im Fragenkatalog ergeben (Anhang A). Die Daten, die im Rahmen der Umfrage erhoben wurden, sind nicht Repräsentativ, da nur vier Unternehmen ihre Asset Verwaltungspraktiken bereit gestellt haben. Trotzdem können aus den Daten Rückschlüsse auf die Art der Verwaltung getroffen werden. Es haben drei kleinere Unternehmen mit 5–10 Angestellten teilgenommen und ein größeres Unternehmen mit 150 Angestellten.

2.1 Existierende Software

Jedes der Befragten Unternehmen verwendet eine Art von Versionskontrolle (Git, SVN, Perforce). Zusätzlich verwendet ein Unternehmen Netzwerklaufwerke und eines verwendet Google Drive.

2.2 Kosten

Die monatlichen Kosten sind für alle Methoden gratis außer für Google Drive.

2.3 Unterstütze Funktionen

Die nächsten zwei Absätze beschreiben welche Funktionen das aktuelle Tool des Unternehmens bietet und welche Funktionen von den Unternehmen gewünscht werden. Die Zahl in Klammer gibt an wie viele Unternehmen diese Funktion angegeben haben.

Aktuelles Tool

- suchen (2),
- Tags hinzufügen (3),
- automatisches hinzufügen von Tags zu Bildern (1),
- Benutzerverwaltung (2),
- Zuweisung zu Projekten (2),
- Versionierung (3),

- Thumbnails für Bilder/Videos (4),
- Thumbnails für 3D Assets (1).

Wunsch Funktionen

- suchen (2),
- Tags hinzufügen (2),
- automatisches hinzufügen von Tags zu Bildern (0),
- Benutzerverwaltung (2),
- Zuweisung zu Projekten (0),
- Versionierung (2),
- Thumbnails für Bilder/Videos (2),
- Thumbnails für 3D Assets (1),
- Backup Möglichkeit (inkl. Verschlüsselung) (1),
- Grafische Anmerkungen (1),
- Asset Unterschiede festellen (1).

Die Ergebnisse decken sich zu einem großen Teil mit den Funktionen des Explorers mit einem VCS.

2.4 Hosting Art

Auf welche Art die Software gehostet werden soll teilt sich genau 50/50 in selber hosten und Software as a Service auf. Software as a Service bedeutet das hosting von dem Hersteller des DAMs oder von einem dritt Anbieter gehostet und aktualisiert wird. Somit hat das Unternehmen keinen Zeitaufwand zum Herstellen der Funktionalität des DAMs.

2.5 Dateiformate

Gewünschte Dateiformate von den Unternehmen. Es wurden viele verschiedene Standardformate genannt aber auch nach Unterstützung für proprietäre Formate gebeten.

Bilder

jpg, png, tga, tif, gif, ico, psd, ai, dds

Audio/Video

mp3, mp4, avi, wmv, wav, aif, mov, mkv, ogg

3D

fbx, obj, ma (Maya)

Dokumente

txt, pdf, docx, xlsx, Open Office Formate

Sonstige

Komplette Adobe Suite, cmat (Material-Beschreibung der Szenen)

2.6 Anzahl/Größe Assets

Es müssen im Schnitt circa 20.000 Assets pro Spiel verwaltet werden, die ca. 15 Gigabyte an Speicher brauchen. Die größte gesamte Asset Bibliothek die angegeben wurde ist 2 Terabyte groß.

2.7 Wiederverwendungsgrad

Der Wiederverwendungsgrad von Assets ist bei den meisten evaluierten Firmen sehr niedrig von 1–10%, nur eine Firma gab an 50% wiederzuverwenden.

Kapitel 3

Nutzen von digitalem Asset Management

Dieses Kapitel beschreibt Szenarien in denen ein PAM von Nutzen ist und welche möglichen Anbieter es momentan auf dem Markt gibt.

3.1 Vorteile

Dieser Abschnitt beschreibt die verschieden Vorteile die ein PAM für ein Unternehmen bringen kann.

3.1.1 Generelle Vorteile

Dieser Abschnitt beschreibt die Vorteile für das ganze Team/Projekt und arbeitet hauptsächlich die messbaren Vorteile für das Management heraus.

Höhere Produktivität

Die Einführung eines PAMs erhöht die Produktivität der Asset Produktion aber auch die generelle Produktivität eines Projektes/Unternehmens. Es gibt zwei Möglichkeiten die Produktivität zu steigern:

1. durch Reduzierung der Kosten bei gleicher Qualität oder
2. durch Erhöhung der erstellten Assets bei gleichen Kosten.

Jede Firma kann für sich selber entscheiden, ob sie lieber mit niedrigeren Kosten produziert oder die Qualität ihrer Assets erhöht. In den meisten Fällen wird die Ersparnis aber in eine Verbesserung der Qualität investiert [2, S. 25]. Die Einführung eines PAMs macht es möglich, durch die Reduzierung der verschwendeten Zeit beim: Suchen, Teilen, Verlieren und Neuerstellen, Überschreiben und Wiederherstellen (falls möglich) von Assets [3, S. 4], die Produktivität zu erhöhen. Nach Gistics[1] verbrauchen kreative Personen 25–40 % ihres Tags mit diesen Tasks [2, S. 25]. Alleine eine Halbierung der Zeit verbracht mit diesen Aufgaben, kann eine immense Kostenersparnis bedeuten und darüber hinaus müssen Mitarbeiter weniger Zeit mit, oft als nervig aufgefassten, Task verbringen. Das bedeutet Firmen können entweder Geld sparen mit einem PAM oder bessere Qualität liefern und somit besser im Vergleich zu Konkurrenten dastehen.

[1] https://gistics.com/

Bessere Skalierbarkeit

Ein zusätzliches System bedeutet zwar eine Anlernphase für neue Nutzer, jedoch bietet es selbst direkt nach der Einführung eine Zeit und Kostenersparnis [2, S. 26]. Wenn Mitarbeiter Assets auf Anhieb finden, nicht mit falschen Versionen gearbeitet werden kann und mehr Assets wieder verwendet werden können, wächst der Vorteil mit der Größe des Teams/Projektanzahl und kann so Skalierungsproblemen entgegensteuern.

Verlässlichkeit / Planbarkeit

Oft werden größere Projekte unterschätzt, da durch die lange Umsetzungsdauer die Komplexität steigt und somit es stetig schwieriger wird den Überblick zu wahren. Mit einem PAM bleibt die Asset-Erstellungskomplexität über den Zeitraum des Projektes weitestgehend gleich einfach und zusätzlich vereinfacht es die Wiederverwendung in späteren Projekten. Assets können so ohne Relation zueinander die verschiedenen Stadien eines Projektes durchlaufen, wie zum Beispiel: „Platzhalter“, „Fertig und wartet auf Freigabe“ und „Bereit für den Einsatz im konkreten Produkt“. Der PAM kann somit Benachrichtigungen an die zuständigen Mitarbeiter schicken falls etwas zum Freigeben oder zum Überarbeiten ist. Somit kann leicht überprüft werden, welche Assets sich grade im welchem Status befinden und sie können so nicht in einem unfertigen Zustand im finalen Produkt überbleiben.

Bessere Qualität

Die Qualität kann auf zwei verschiedene Arten verbessert werden. Erstens kann die Anzahl an verwendeten falschen Assets (alte Versionen) verringert werden, da die Assets immer von einem zentralen Punkt mit der neuesten Version angeboten werden [2, S. 26]. Zweitens kann mehr Zeit für die kreative Arbeit hinter den Assets aufgebracht werden [2, S. 26]. Ein Beispiel ist, dass es Artisten leichter fällt verschiedene Variationen zu testen, da es einfacher ist mit mehreren Versionen zu arbeitet.

Mehr Flexibilität

Ändert sich zum Beispiel eine Mechanik des Spiels kann es sein, dass viele Assets angepasst werden müssen. Mit einem PAM kann leicht abbilden werden, welche Assets noch verändert werden müssen und welche schon einsetzbar sind. Ein anderes Problem, das bei größeren Teams schnell auftreten kann, ist, dass ein Mitarbeiter das Projekt verlässt [2, S. 26]. Mit einem PAM sieht man genau für welche Assets er zuständig war und in welchem Status die Assets verlassen wurden. Ein neuer Mitarbeiter kann sich im ersten Schritt die Zuständigkeit der Assets übertragen und im zweiten Schritt mit einer passenden Verschlagwortung der Assets und einer guten Beschreibung des dazugehörigen Tasks leicht in seine neue Arbeit einlesen.

Bessere Integration zwischen Abteilungen

Assets können leichter an das Marketing weiter geleitet werden, da sie nur freigeben werden müssen [2, S. 28]. Zusätzlich können Assets auch leichter von anderen Projekten

wieder verwertet werden, da alle Assets die fertig sind, also mit einem *fertig* Tag markiert wurden, angezeigt werden können.

Transparenter Projektfortschritt

Der Projektmanager weiß zu jeder Zeit welche Assets in welchem Status sind, ohne händisch nachzuschauen oder bei jedem Mitarbeiter nachfragen zu müssen [2, S. 28].

3.1.2 Individuelle Vorteile

Dieser Abschnitt beschreibt die Vorteile für ein einzelnes Projektmitglied und arbeitet Effekte heraus die Mitarbeiter konkret betreffen.

Weniger Mehraufwand

Jeder Mitarbeiter kann mehr Zeit mit dem konkreten Erstellen von Assets verbringen, da weniger Zeit mit Suchen und Verbreiten von Assets verbraucht wird [2, S. 27].

Besser parallel arbeiten

Durch die Zuteilung von Assets zu Tasks, kann leicht gemeinsam an einem Task gearbeitet werden, ohne aus den Augen zu verlieren wer was macht. Mit einem gutem PAM können Assets vom Server heruntergeladen, verändert und die neue Version, direkt von der Festplatte des Mitarbeiters, zurück auf den Server, mit einer neuen Versionsnummer, geladen werden. Für diese Funktion kommt man aber nicht um einen Desktop Client herum, der die Beziehung zwischen Asset am Server und Dateisystem speichert.

Zurücksetzen / Wiederherstellen

Der häufigste Anwendungsfall ist für diese Funktion vermutlich, falls beim aktuellen Bearbeitern etwas schief läuft, kann man die Vorgängerversion mit einem Klick vom Server wiederherstellen. Es können auch Versionen von viel früher ohne Nebeneffekte wieder hergestellt werden. Möchte man zum Beispiel eine Version von vor drei Wochen mit einem Version Control System (VCS) wiederherstellen, bleibt dem Benutzer nichts anderes übrig als die Commits lokal durch zugehen, bis die richtige Version gefunden wird.

Verschiedene Variationen

Mit einem PAM kann man leichter verschiedene Variationen erzeugen, da man Assets auf verschiedene Arten gruppieren kann. Die antiquierte Anforderung, dass eine Datei nur in einem Ordner sein kann, wird mit einem PAM aufgehoben. Dadurch kann ein Asset zum Beispiel in ein Sammlung mit all seinen übersetzen Versionen und gleichzeitig in einer Sammlung *Deutsch* und der Sammlung *Küchenutensilien* sein.

Benutzerverwaltung

Die Benutzerverwaltung kann auf verschiedene Arten konfiguriert werden. Mitarbeiter könnten zum Beispiel nur Assets sehen die sie erstellt haben, die einen *fertig* Tag besitzen

oder die zu einem Bestimmten Task gehören. Neben den Ansichtsrechten können auch verschiedene Rechte für Erstellen, Bearbeiten und Löschen vergeben werden. Mit diesen Regeln kann das Risiko für ungewollten Schaden vermindert werden und Benutzer sehen nur Assets die für sie auch wirklich relevant sind [2, S. 28].

Zuversicht

Durch direktes anzeigen von Vorschaubildern von Photoshop Dateien / 3D Dateien können sich Mitarbeiter sicher sein bzw. sich leichter davon überzeugen, dass Assets in der richtigen Version am Server vorliegen [3, S. 6].

Kontrollierter Arbeitsablauf

Zusätzlich können Regeln je nach Dateityp eingeführt werden. Zum Beispiel kann eine Regel sein, dass alle Assets mit .fbx in PascalCase[2] benannt sein müssen und der Tag *in Arbeit* bei der Erstellung hinzugefügt wird. Sobald der Artist fertig ist mit dem Asset kann der Tag *in Arbeit* mit dem Tag *Entwurf* ausgetauscht werden und der Projektleiter oder Lead Artist wird mit einer Benachrichtigung informiert, dass er etwas zum Freigeben hat [2, S. 28]. Ist der Freigeber nicht zufrieden kann er Verbesserungsvorschläge in Textform zum Asset hinzufügen und den Tag wieder austauschen. Sobald das Asset den Ansprüchen des Freigebenden genügt, wird es auf *fertig* gesetzt und wird somit z. B. für das Level Design Team sichtbar [2, S. 28].

3.2 Stand der Technik

Dieses Kapitel beschreibt einige verschiedenen DAM Anbieter die im Rahmen dieser Arbeit evaluiert wurden. Es wurden die fünf meist verwendeten Systeme evaluiert [13]. Da alle Systeme ähnliche Funktionalität abdecken wurde aufgrund von Preis, Kunden und hosting Technologie verglichen. Eine weitere Gemeinsamkeit ist, dass alle evaluierten DAMs als *Software as a Service (SaaS)* in der Cloud angeboten werden und somit sind alle System Webserver basiert. Diese Eigenschaft unterstreicht das Unternehmen lieber Webseiten betreiben, als FAT Clients, da der Wartungsaufwand geringer ist.

hyperCMS

Preis: Der Startpreis von hyperCMS liegt bei 22€ pro Monat. Je nach gewählter Speichergröße und Zusatzpaketen steigt der Preis [9]. Zusätzlich wird eine gratis Open Source Version zum selber aufsetzen bereit gestellt.

Kunden: Die Kundenbasis von hyperCMS besteht hauptsächlich aus Finanzunternehmen, wie Banken und Versicherungen.

Technologie: Gehostet wird das CMS mit DAM Funktion entweder selber auf einem Apache Webserver oder von hyperCMS.

Canto

Canto ist das Unternehmen hinter dem Produkt Cumulus.

[2] https://en.wikipedia.org/wiki/PascalCase

Preis: Der Preis basiert auf der Anzahl der Nutzer und dem genutzten Speicher und startet mit 5000$ pro Jahr (~420$ pro Monat) [8].

Kunden: Breitgefächert von Einzelhandel bis zu Unternehmen und Forschungszentren wie Intel und NASA.

Technologie: Cloud gehostet von Canto, On Premise oder Hybrid (sensible Daten werden On Premise gespeichert und fertige Inhalte in der Cloud).

Celum

Preis: auf Anfrage.

Kunden: Verschiedene Produktionsunternehmen wie Voest, Handel und E-Kommerz. Generell liegt der Fokus auf Produkt und Waren Erzeugung/Vermarktung.

Technologie: Gehostet von Celum und On Premise.

Nuxeo

Preis: auf Anfrage.

Kunden: Breitgefächert von Raumfahrt bis Einzelhandel. Einer der wenigen DAM Anbieter mit einer Spielfirma (EA) in der Kundenliste. Leider nur für Spiele-Builds [11].

Technologie: Gehostet von Nuxeo und On Premise.

Shotfarm

Preis: Start Preis 40$ pro Monat [14] für Hersteller und gratis für Einzelhandel.

Kunden: Nicht öffentlich verfügbar. Circa 3500 Hersteller und 8000 Einzelhändler und Verteiler.

Technologie: Gehostet von Shotfarm, da sie eher einen Service zum Verbinden von Herstellern und Einzelhändler bieten als ein konventionelles DAM.

Kapitel 4

Konzept

Dieses Kapitel beschreibt welche Funktionen für ein PAM hilfreich sind. Ein Teil dieser Funktionen ergibt sich aus den Anforderungen aus dem Kapitel 2 und der andere Teil aus den Erkenntnissen gewonnen während der Bachelorrecherche.

Die Abschnitte *Konvertieren, Metadaten, Benennung von Dateien, Tags, Suche und Ratings* beschreiben Funktionen die in existierenden DAMs größtenteils Standard sind. Die Abschnitte *Automatische Analyse, Suche, Links zu Assets teilen, Hierarchien für Ordner/Tags, Sammlungen und Metadaten Sammlungen* beschreiben Funktionen die generell als Vorteil gesehen werden können und die Abschnitte *Tags und Plug-in System für Dateiformate* beschreiben Vorteile die sich aus der Umfrage herauskristallisiert haben. Die einzelnen Bereiche überschneiden sich häufig und viele Absätze können mehrfach zugeordnet werden.

4.1 Konvertieren

Es soll den Benutzern möglich sein Assets von einem Dateiformat in eine anderes zu konvertieren ohne das Original zu beschädigen. Es gibt zwei Möglichkeiten das zu erreichen:

1. konvertierte Versionen werden als neues Asset erstellt und automatisch mit den Metadaten, die nicht Format spezifisch sind (Beschreibung, Tags, usw.), des Originals befüllt oder
2. die konvertierte Version wird direkt heruntergeladen und hat somit keinen Einfluss auf den Zustand des PAMs.

4.2 Metadaten

Metadaten sind ein wichtiger Bestandteil eines PAMs. Sie bringen durch die zusätzlichen Informationen, die zu jedem Asset gespeichert werden, die Möglichkeit nach diesen Informationen zu filtern. Das ermöglicht Mitarbeitern Assets schneller zu finden und wenn nötig können dynamische Sammlungen erstellt werden. Ein Beispiel dafür ist, es sollen alle Assets von dem Projekt Spiel1 mit dem Tag *fertig* in die Sammlung *Level Design bereit* hinzugefügt werden. Ein weiterer Vorteil ist das Metadaten elementar zu jedem

Assets gehören und sie sich somit nur ändern wenn sich auch das Asset ändert. Diese Eigenschaft ist besonders wichtig um langfristige Konsistenz im PAM zu schaffen. Weiters Folgen drei Klassen von Metadaten, die von der billigsten zu der teuersten Erzeugung gereiht sind.

- Automatisch erzeugt Metadaten Komplexe automatische Metadaten, wie Breite, Höhe oder Farbart von Bilder, werden durch die Erstellungsprogramme von Assets, wie zum Beispiel Photoshop oder Gimp erzeugt und Basisinformationen werden vom Dateisystem bereit gestellt. Diese Daten sind am billigsten zum Wiederverwenden, da sich diese Metadaten schon auf den Assets befinden.
- Masseneintrag Masseneinträge ermöglichen das hinzufügen von Metadaten zu vielen Assets auf einmal. Einerseits sollte dieser Vorgang im PAM jederzeit auf eine ausgewählte Menge an Assets angewendet werden können, andererseits soll diese Möglichkeit auch beim Importieren bestehen, um Mitarbeiter anzuregen Assets richtig in das PAM einzupflegen. Beispiele für Metadaten die über einen Masseneintrag hinzugefügt werden können sind: die Lizenz, Name der Bibliothek aus der die Assets kommen oder Tags wie Seamless oder Texture. Mit je mehr Informationen Assets in das PAM eingepflegt werden, umso einfacher sind sie später wieder zu finden. Masseneinträge bitten dafür ein sehr gutes Aufwand/Leistung Verhältnis.
- Individuelle Metadaten Individuelle Metadaten sind im Vergleich zu den vorherigen zwei Methoden die aufwändigsten, stellen aber die spezifischsten Informationen über das Asset bereit. So können spezielle Beschreibungen zu Bilder hinzugefügt werden oder bestimmte Tags.

Metadaten Sammlungen

Da sich die benötigten Metadaten zwischen den verschieden Dateiformaten als auch Verwendungsarten der Assets wesentlich unterscheiden können, gibt es vordefinierte Metadaten Sammlungen. Eine Metadaten Sammlung enthält benutzerdefinierte Felder in die Text, Nummern oder Werte mit einem kontrollierten Vokabular (das Feld Lokalisierung bietet z. B. nur *Deutsch* und *Englisch* zur Auswahl) eingetragen werden können. Der Benutzer kann beliebige Metadaten Sammlungen zu einem Asset hinzufügen.

4.3 Sammlungen

Eine Sammlung von Assets die entweder automatisch mit Filterkriterien oder händisch erstellt wurde. Um ständiges Suchen nach den selben Assets zu vermeiden gibt es Sammlungen. Diese können entweder Persönlich sein, mit anderen geteilt werden oder an ein Projekt verlinkt werden. Beispiele für Sammlungen sind, Assets die den Tag *Deutsch* haben und zu einem bestimmten Projekt gehören oder verschiedene Versionen von einem Asset (sprachlich/optisch). Um das Navigieren zu erleichtern ist es hilfreich, wenn die dazugehörigen Sammlungen zu einem Asset angezeigt werden. Denn so kann vom jedem Asset leicht auf andere Sammlungen und somit zu andern Variationen von dem Asset navigiert werden.

4.4 Benennung von Dateien

Assetnamen sollen die wichtigsten Eigenschaften des Assets beinhalten. Diese Eigenschaften finden sich vermutlich wieder in anderen Metadaten, wie Tags und Metadaten Sammlungen, aber sind dennoch wichtig, da sie einen schnellen Überblick über die Assets nach einer Suche bieten. Für die Benennung von Assets kann es viele Regeln geben, die Unternehmen zu einem großen Teil nach persönlichen Vorliebe auswählen. Beispiele für Regeln sind,

- Groß und Kleinschreibung vorgeben,
- erlauben oder verbieten von '_', '-' und ' ',
- erlauben oder verbieten von Sonderzeichen,
- u. v. m.

Diese Regeln sollen variabel erstellbar sein und von dem PAM überprüft werden.

4.5 Ordner?

Ordner haben sich über lange Zeit einen Namen gemacht aber bringen sie wirklich einen Vorteil, der nicht mehr Nachteile mit sich bringt? Mit ausführlichen hinzufügen von Metadaten und Sammlungen sind Assets genauso gut, wenn nicht besser, als mit einer Ordnerstruktur zu finden [15]. Viele Benutzer sind zwar an Ordnerstrukturen gewöhnt aber sie bringt viel Ungewissheit mit sich. Diese Ungewissheit rührt daher, dass Assets in verschiedene Order passen aber nur einem Ordner zugewiesen werden können. Kommt eine 3D Asset von einem Messer in den Ordner Küchenutensilien oder Waffen? Um diese Frage zu Vermeiden gibt es Sammlungen und somit kann auf Ordner verzichtet werden.

4.6 Tags

Tags sind vermutlich eines der universell ersetzbarsten Metadaten, da für jede Situation neue Tags hinzugefügt werden können. Jedoch sollte falls es spezifischere Metadaten gibt auf diese zurück gegriffen werden, wie Metadaten Sammlungen. Gibt es eine Metadaten Sammlung für Texturen, die die Felder, Farbkanäle(R/RG/RGB/RGBA) und Tiling Texture(Ja/Nein) hat, soll auf diese zurück gegriffen werden, da anders Informationen, die für diese Art von Asset eingetragen werden, eventuell vergessen werden. Wie bei den Metadaten Sammlungen ist auch bei Tags ein kontrolliertes Vokabular von Vorteil, um Tippfehler und das einführen von ähnlichen Tags zu vermeiden. Tags sind eine unstrukturierte Form von Metadaten Sammlungen. Beide können die Rolle des anderen übernehmen ohne Informationsverlust, jedoch ist die Eingabe der Information unterschiedlich.

4.7 Hierarchien für Ordner/Tags

Für Ordner wurde mit Sammlungen zwar schon ein Ersatz gefunden 4.5, sie verhalten sich aber genauso wie Tags in Bezug auf Hierarchien und sind deshalb in diesem

Abschnitt inkludiert. Abbildung 5.3 zeigt wie eine Tag Hierarchie aussehen kann. Eine Hierarchie wirft selbst mit so wenigen Tags schon viele Fragen auf:

- Kann *Texture* als Wurzelelement bleiben oder sollte es unter *2D* stehen?
- Wenn die *CC BY Lizenz* gewählt wird, werden auch *CC* und *License* zu einem Asset hinzugefügt?
- Werden Mutter Tags erst angewandt, wenn alle Kind Tags ausgewählt wurden?
- Kann es einen Tag *Grass* in 2D und 3D geben?

Da Hierarchien viele Fragen auswerfen die immer subjektive Antworten liefern werden, sollten sie so gut wie möglich vermieden werden.

4.8 Ratings

Rating stellen eine Möglichkeit dar die Qualität von Assets zu Bewerten. Sie können verschiedene Variationen annehmen, 1–5, 1–10, −1–5 und viele mehr (Ein minus Wert kann z. B. für abgelehnt stehen). Auch wenn Ratings hilfreich sein können verschiedene Variationen zu Bewerten, bringen sie das Problem mit sich, dass sie nicht genau definiert sind. Ein Rating von 2 kann für Mitarbeiter1 etwas anderes bedeuten als Mitarbeiter2. Selbst mit genauen Unternehmenskonventionen ist der Spielraum oft zu groß um es schnell entscheiden zu können. Somit wäre eine optionale Rating Funktion sicher für manche Unternehmen interessant aber durch die ungenaue Verwendung sicher nicht in jedem Projekt erwünscht.

4.9 Links zu Assets teilen

Es sollen einfach *Google Drive* oder *One Drive* ähnliche direkt Links zu Asset erstellt werden können. Optionen die bereit stehen sollten,

- Benutzerschutz,
- öffentlich Zugänglich und ein
- Ablaufdatum für einen Link.

4.10 Suche

Die Suche ist, um das Finden und Wiederverwerten von Assets zu ermöglichen, eines der wichtigsten Werkzeuge in einem PAM. Ein *Fuzzy String* Vergleich wird hier vermutlich die besten Ergebnisse liefern, um eine Vielzahl von Suchverhalten abzudecken. Darüber hinaus können Assets über Sammlungen gefunden werden. Gibt der Benutzer irgend einen Begriff ein, wird in allen Metadaten danach gesucht und somit bei einer guten Verschlagwortung sicher gefunden.

4.11 Plug-in System für Dateiformate

Es sollen alle Standardformate in jeder Kategorie (Bild, Video, 3D, usw.) unterstützt werden. Da es jedoch unmöglich ist wirklich alle Dateiformate zu unterstützen, soll es

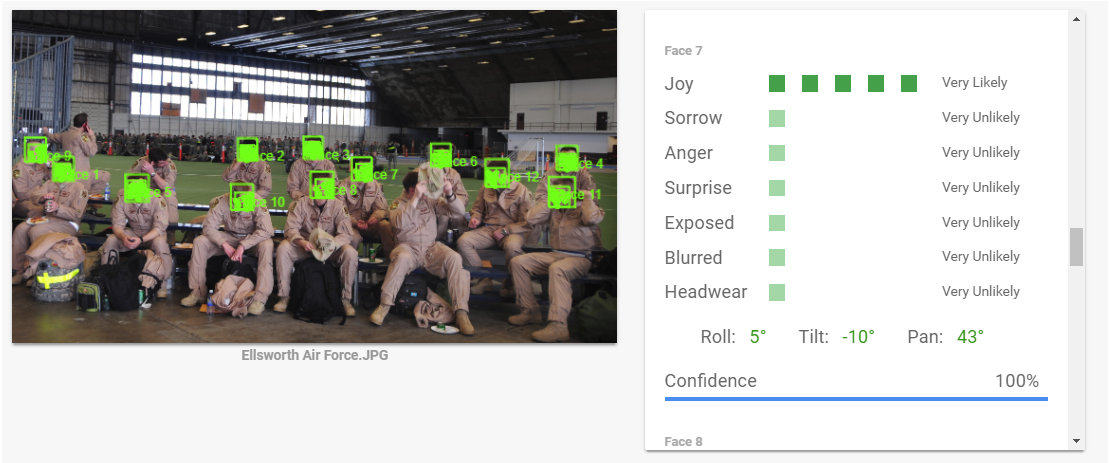


Abbildung 4.1: Google Cloud Vision Gesichts-Emotionserkennung.

den Benutzern möglich sein, eigene Erweiterungen für weniger verbreitete oder proprietäre Dateiformate zu erstellen.

4.12 Automatische Analyse

Dieser Abschnitt beschreibt die Vorteile die durch die Fortschritte in der automatischen Analyse von Bildern erzielt wurde. Es wird insbesondere auf die Ergebnisse von Google Cloud Vision[1] und Clarifai[2]eingegangen. Google Cloud Vision wurde mit der bereit gestellten online GUI getestet und Clarifai wurde ebenfalls mit der bereit gestellten online GUI und einer eigenen Node.js Installation getestet.

4.12.1 Gesichtserkennung

Gesichtserkennung bedeutet im einfachsten Sinne das Positionserkennen von Gesichtern, um das manuelle hinzufügen von Namen zu erleichtern. Schwerer ist das automatische benennen von Gesichtern, darum bieten die meisten Analyseanbieter auch nur ersteres an. Abbildung 4.1 zeigt US Soldaten [4], analysiert mit Google Cloud Vision. Es wird angezeigt welche Gesichter auf dem Foto erkannt wurden und welche Emotionen aus ihren Gesichter ausgelesen werden können. Gesichtserkennung mit automatischer Zuweisung eines Namens ist momentan noch nicht möglich (Stand 01.01.2018). Hier können andere Libraries wie tracking.js Abhilfe schaffen, die zumindest manuelles zuweisen von Namen erlauben. Clarifai findet auf dem selben Bild keine Gesichter.

4.12.2 Texterkennung

Ein PAM soll Text aus allen Bildformaten in Metadaten umwandeln können, um den Inhalt des Bildes durchsuchbar zu machen. Die automatische Texterkennung hat über

[1] https://cloud.google.com/vision/
[2] https://www.clarifai.com/

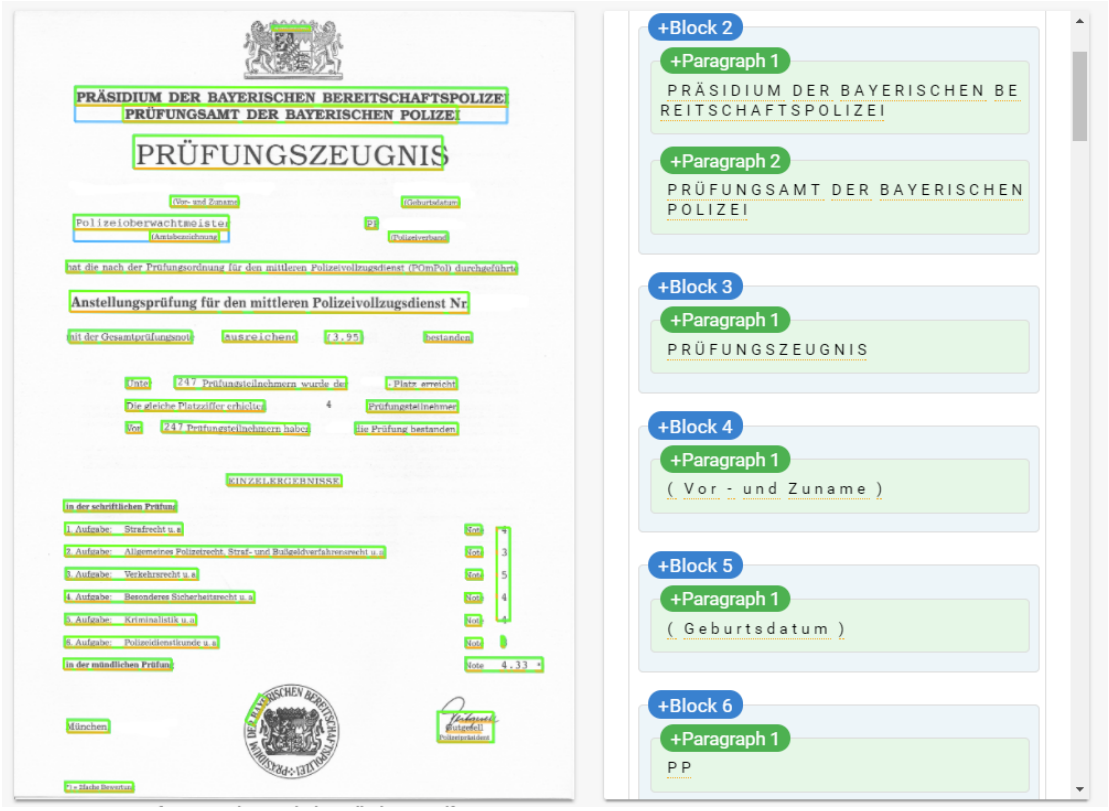


Abbildung 4.2: Google Cloud Vision automatisches erkennen von Text.

die letzten Jahre große Fortschritte gemacht und ist mit Services wie Google Cloud Vision leicht zugänglich für jedermann, jedoch ist die Qualität der Resultate sehr abhängig vom bereitgestellten Bildmaterial [10]. Abbildung 4.2 zeigt welcher Text aus einem eher kompliziert strukturierten Dokument [5] erkannt wird. Clarifai hat noch keine Texterkennung.

4.12.3 Automatisches hinzufügen von Tags

Abbildung 4.3 zeigt welche Tags zu dem Herbstbild [6] hinzugefügt werden. Wie man sieht ist das Resultat der gefunden Tags ausgesprochen gut. Clarifai bietet ein ähnliches Ergebnis, mit Herbst, Blatt, Holz, Baum, Landschaft, Ahorn, Park, Natur und einige mehr, an.

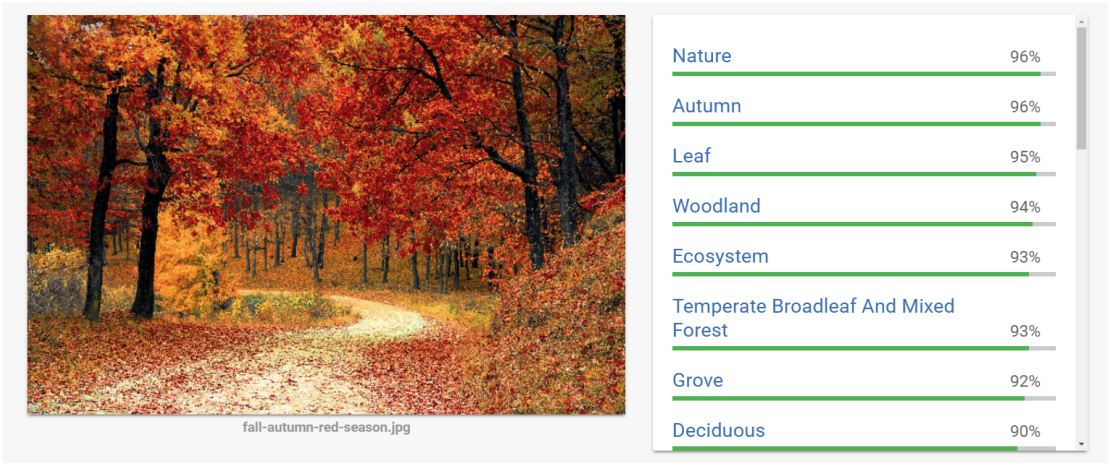


Abbildung 4.3: Google Cloud Vision automatisches Tag hinzufügen.

Kapitel 5

Umsetzung

Der erstellte Prototyp setzt nur einen Teil der Funktionen um, die in den letzten Kapiteln vorstellt wurden, da es wesentlich länger dauern würde, alle diese Funktionen zu implementieren. Trotzdem wurden während der Implementierung viele Erkenntnisse gewonnen, die auch in die anderen Kapitel mit eingeflossen sind. Der Prototyp hat den Arbeitstitel AssetManager (AM) und wird in den folgenden Abschnitten auch so genannt.

5.1 Design

In der Abbildung 5.1 ist der grundlegende Aufbau des Benutzerinterfaces abgebildet. Das Design ergibt sich aus der Bibliothek *Material Design In XAML*[1] und wurde gewählt um ein klares Interface ohne großen Mehraufwand zu erstellen. Neben einer online Dokumentation auf GitHub[2] gibt es einen Client zum Download in dem alle Elemente angesehen und getestet werden können. Um das Design zu integrieren ist nur ein Eintrag in der *App.xaml* nötig, die das *ResourceDictionary* des Paketes in das des Programms hinzufügt. *Material Design In XAML* basiert wie der Name vermuten lässt auf Googles Material Design[3].

Das Layout des Prototypen ist an den Windows Explorer und Adobe Bridge angelehnt. Beides Programme die sich schon über Jahre bewährt haben.

NuGet Package Manager

Alle Pakete wurden mit dem NuGet Package Manager hinzugefügt.

5.2 Architektur

Dieser Abschnitt beschreibt den technischen Aufbau des AssetManagers und welche Gedanken hinter den Entscheidungen stehen.

[1] http://materialdesigninxaml.net/
[2] https://github.com/ButchersBoy/MaterialDesignInXamlToolkit
[3] https://material.io/guidelines/

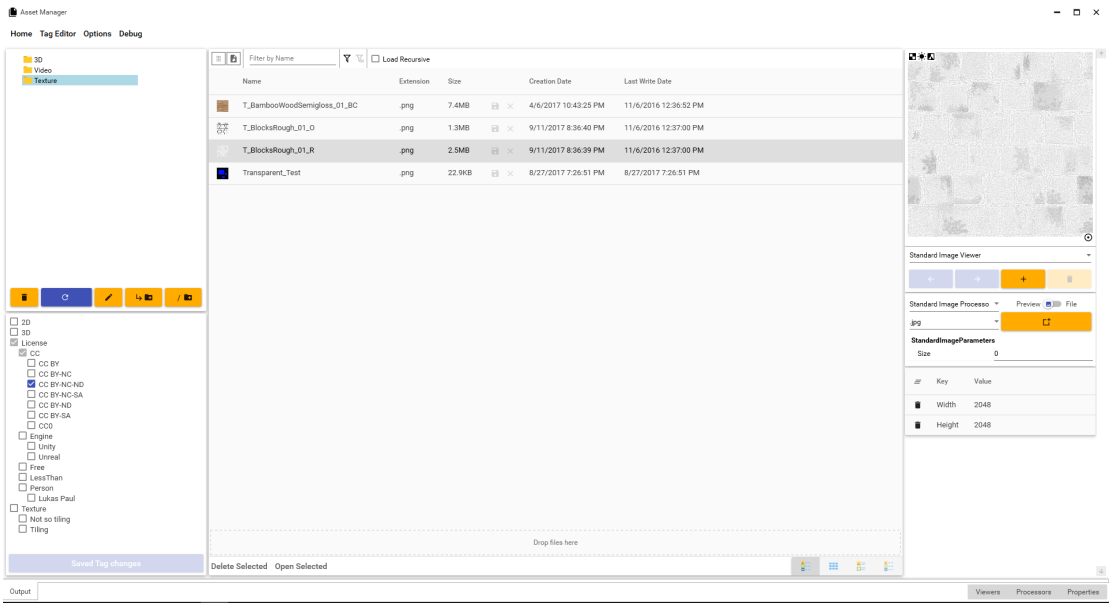


Abbildung 5.1: GUI

5.2.1 Frontend

Das Benutzerinterface wurde ganz normal mit XAML Dateien erstellt, wie für WPF üblich. Um eine saubere MVVM Trennung zu gewährleisten wurde das Framework Caliburn.Micro[4] verwendet. Der Hauptvorteil von dem Framework besteht darin, dass es zu 80% aus *convention over configuration* besteht und somit mit wenig Schreibaufwand ein Programm mit den Standardeinstellungen erstellt werden kann.

Das lästige binden von DelegateCommand oder RelayCommands wurde mit einem Textparser ersetzt. Ohne Caliburn:

```
<Button Command="{Binding ReloadCommand}" Content="Reload"></Button>
```

```
public DelegateCommand ReloadCommand { get; set; }
public ClassName()
{
  ReloadCommand = new DelegateCommand(Reload);
}
public void Reload()
{
  //...
}
```

Mit Caliburn:

```
<Button cal:Message.Attach="Reload()" Content="Reload"></Button>
```

```
public void Reload()
{
  //...
}
```

[4] https://caliburnmicro.com/

Programm 5.1: Beispiele für IoC Injection mit Caliburn.Micro

```
1 public class IoCTest
2 {
3     private Person _person;
4
5     //Constructor Injection
6     public IoCTest(Person person)
7     {
8       _person = person;
9     }
10
11    //Property Injection
12    //Call BuildUp on the ioCTestReference
13    //_container.BuildUp(ioCTest);
14    public Person Person { get; set; }
15
16    //Static IoC
17    public void GetPerson()
18    {
19      _person = IoC.Get<Person>();
20    }
21 }
```

Weiters biete es noch Entkopplung von den ViewModels, einen IoC Container, u. v. m. [5]

IoC Container

Ein IoC Container stellt konkrete Referenzen zu Objekttypen bereit, wenn sie gebraucht werden. Gibt es eine Klasse Person, kann diese mit dem Caliburn IoC auf drei Arten erstellt werden. Beispiel dafür ist Programm 5.1.

5.2.2 Backend

Das Backend wurde mit Hilfe des Entity Frameworks[6] implementiert. Es müssen nur die Klassen mit ihren Felder erstellt werden und das dazugehörige Datenbankschema, zu sehen in Abbildung 5.2, wird daraus generiert. Neben dem üblichen Datenbank Kontext mit den DbSets, wurde eine generisches Repository mit Hilfe von [7] erstellt. Die Klassen *EntityFrameworkReadOnlyRepository* und *EntityFrameworkRepository* stellen einen generischen Zugang zur Datenbank zur Verfügung und werden in der Klasse *BaseDataAccess* und in den DataAccess Klassen der einzelnen Entities konsumiert. Das ermöglicht eine saubere Trennung über die DataAccess Klassen. Darüber hinaus stellt die Klasse *BaseDataAccess* zwei Funktionen bereit,

- *DeleteForeignEntities(TViewModel view, AssetRepository assetRepository)* um Aufräumarbeiten vor jedem Löschen ausführen zu können, wie das Löschen von Entities die auf diese Entity zeigen oder das Löschen der Assets, dazugehörigen Vor-

[5] https://caliburnmicro.com/documentation/
[6] https://docs.microsoft.com/en-us/ef/

schauen und Thumbnails und
- *InjectDependencies* damit ViewModels automatisch wieder zurück in Entities umgewandelt werden können, um sie weiter verarbeiten zu können.

Leider ist die Abtrennung in eine eigenes Projekt der Datenzugriffsschicht nicht möglich, ohne auch alle ViewModels in ein eigenes Projekt zu extrahieren, da sie sowohl eine Referenz auf die ViewModels als auch die Entities brauchen und es zu einer kreisförmigen Abhängigkeit kommen würde.

5.3 Tag Editor

Die Umsetzung der Basisfunktionen des Tag Editors war schnell erledigt, da nur hinzufügen (je nachdem welcher Tag ausgewählt ist), Veränderungen am Tag Namen speichern und den ausgewählten Tag löschen, implementiert werden musste.

Das Drag and Drop Feature bereitete mehr aufwand, wurde am Ende aber so gelöst, dass Tags genommen und auf andere Tags gezogen werden können, um sie unter dem letzteren Tag einzugliedern. Falls ein Tag auf das Wurzellevel soll, erscheint ein Bereich in den der Tag gezogen werden kann, siehe Abbildung 5.3.

5.4 Statusbar

Die Statusbar zeigt Informationen, die über das ILogger Interface geloggt wurden, an. Neben Programmevents können auch Verarbeitungs- und Anzeige-Pluginevents hier geloggt werden. Abbildung 5.4 zeigt Debugevents, die die Zeit die das Bilderanzeigezusatzmodul braucht loggt.

5.5 Unterstütze Dateitypen

Abbildung 5.5 zeigt welche Dateitypen bis jetzt mit einem eignen Viewer unterstützt werden.

5.6 Assets Verarbeitung

Die Asset Verarbeitung war eine der anspruchsvollsten Aufgaben, die im Rahmen dieses Prototypen implementiert wurden, da sie so generisch wie möglich alle Dateitypen abdecken muss. Sowohl die Asset Verarbeitung als auch die Asset Anzeige ist Plugin basiert, kann somit jederzeit, auch von dritt Anwendern, erweitert werden.

Abbildung 5.6 zeigt das Klassendiagramm des *StandardImageProcessors*. Jeder Asset Processor wird mit *AssetProcessorOptions* initialisiert. In diesen Optionen kann das Plugin angeben, ob es Assets beim Importieren verarbeiten will, welche Dateitypen gelesen werden können und welche Dateitypen produziert werden können. Neben den Optionen werden zwei mal *AssetProcessorParams* bereit gestellt. Mit diesen Parametern werden die Einstellungen beim Import von Assets und beim Verarbeiten im Programm festgelegt. Die Parameter werden darüber hinaus mit Reflection dem Benutzer zur Bearbeitung zur Verfügung gestellt.

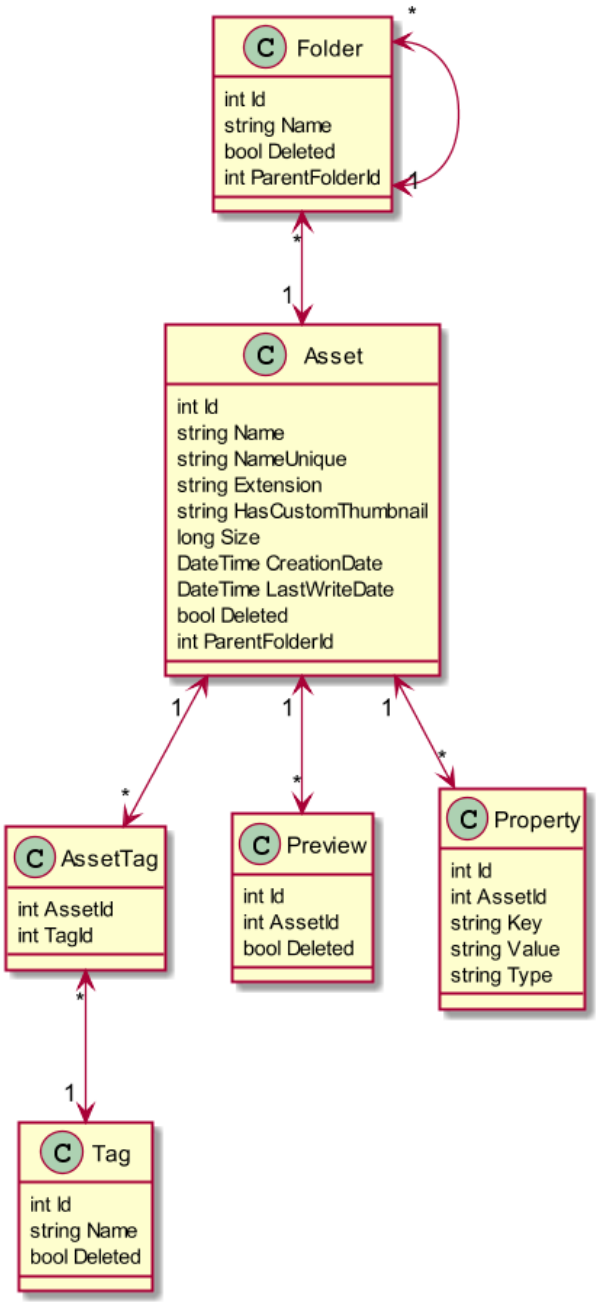


Abbildung 5.2: Prototyp Relationen

Das Resultat eines Asset Processor kann eine neue Datei sein, die als neues Asset importiert wird aber auch Schlüsselwertpaare die dem Asset hinzugefügt werden, wie die Breite und Höhe eines Bildes.

Abbildung 5.3: Tag Editor

Abbildung 5.4: Statusbar

5.6.1 Import von neuen Assets

Abbildung 5.7 zeigt welchen Ablauf eine Datei durchläuft, wenn sie zum AssetManager hinzugefügt wird und wenn das neu generierte Asset ausgewählt wird. Ablauf wie eine Datei hinzugefügt wird,

1. hineinziehen von Datei,
2. erstellen des Assets,
3. verschieben der Datei,
4. verarbeiten des Assets mit allen Asset Prozessoren, die sich als OnLoad registriert haben,
5. erstellen eines Thumbnails,
6. laden des Thumbnails.

Wird ein Asset im AssetManager ausgewählt läuft eine verkürzte Version des Importprozesses ab,

1. laden des Assets,
2. laden des Thumbnails.

Image	Supported	Convert From	Convert Into	Animated
bmp	■	■		
dds	■			
gif	■	■	■	
hdp	■			
ico	■			
jpg	■	■	■	
jpeg	■	■	■	
jxr	■			
png	■	■	■	
tif	■	■	■	
tiff	■	■	■	
wdp	■			
tga	■			

Video / Audio	Supported	VLC Supported
mp4	■	■
avi		■
wmv	■	■
mp3	■	■
wav	■	■
caf		■
aif	■	■
aiff	■	■
flv		■
ogv		■
webn		■
mov		■

Text	Supported
txt	■

	Supported	Syntax Highlighting
cs	■	■
js	■	■
htm	■	■
html	■	■
asp	■	■
aspx	■	■
asax	■	■
asmx	■	■
ascx	■	■
master	■	■
boo	■	■
atg	■	■
css	■	■
c	■	■
h	■	■
cc	■	■
cpp	■	■
hpp	■	■
java	■	■
patch	■	■
diff	■	■
ps1	■	■
psm1	■	■
psd1	■	■
php	■	■
tex	■	■
vb	■	■
xml	■	■

	Supported	Syntax Highlighting
xsl	■	■
xslt	■	■
xsd	■	■
manifest	■	■
config	■	■
addin	■	■
xshd	■	■
wxs	■	■
wxi	■	■
wxl	■	■
proj	■	■
csproj	■	■
vbproj	■	■
ilproj	■	■
booproj	■	■
build	■	■
xfrm	■	■
targets	■	■
xaml	■	■
xpt	■	■
xft	■	■
map	■	■
wsdl	■	■
disco	■	■
ps1xml	■	■
nuspec	■	■
md	■	■

General Files	Supported
pdf	■

Adobe	Supported
psd	■

Office	Supported
xlsx	■

Abbildung 5.5: Unterstützte Dateitypen

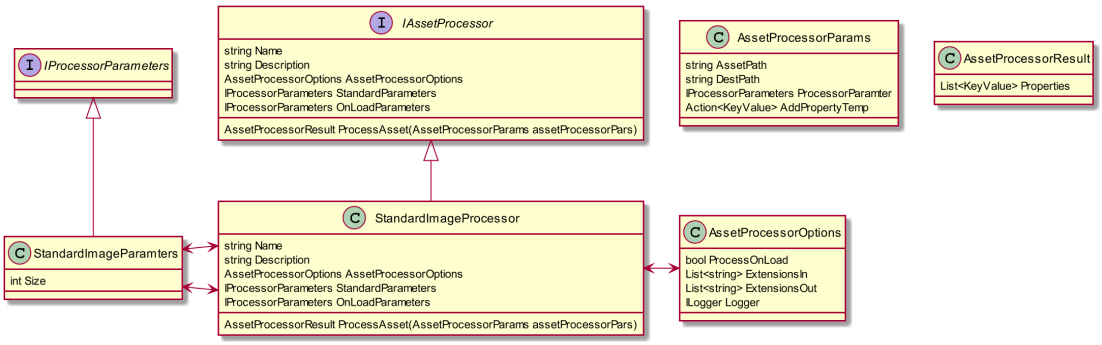


Abbildung 5.6: Prototyp Asset Prozessor

5.7 Assets Anzeige

Die Asset Anzeige verfolgt einen ähnlichen Ansatz wie die Asset Prozessoren. Es werden mit Hilfe der *AssetViewerOptions* festgelegt welche Dateitypen unterstützt werden und mit welcher Priorität der Viewer geladen wird. In der Abbildung 5.8 wird das Klassendiagramm des *GifViewerViewModel* dargestellt. Viewer bestehen nicht nur aus einer

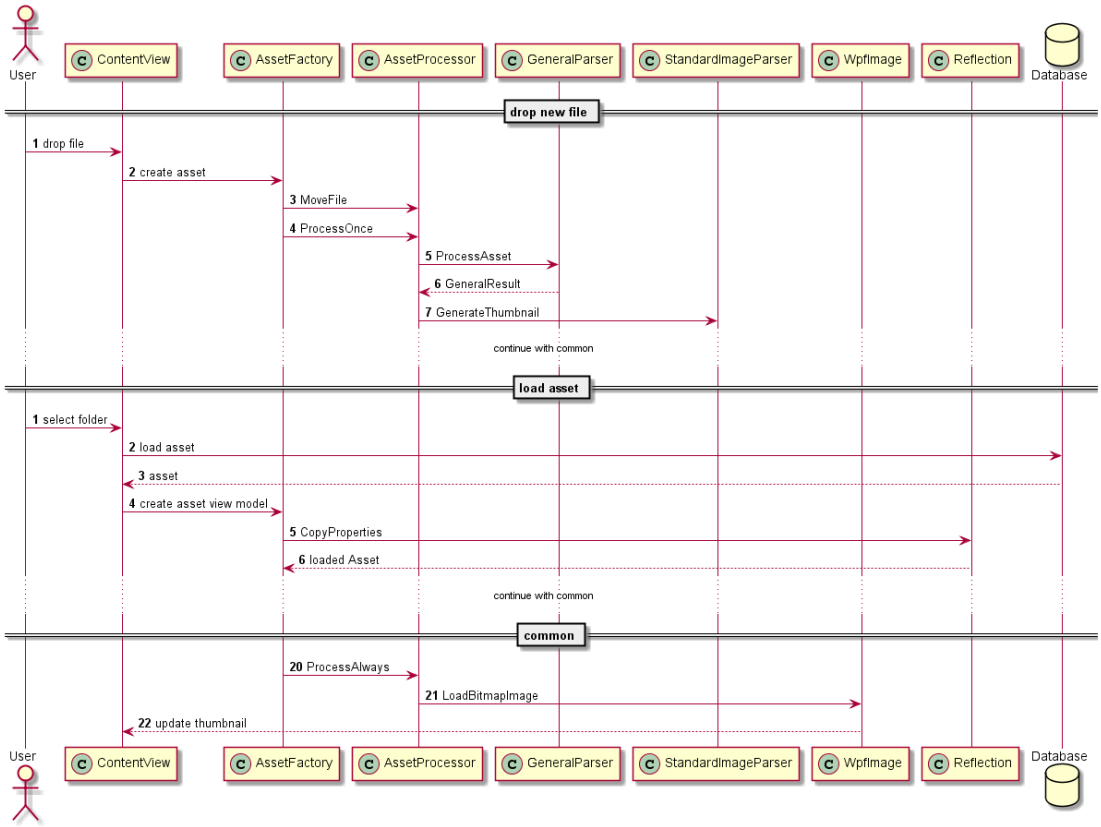


Abbildung 5.7: Prototyp Asset Verarbeitung

Klasse wie Asset Prozessoren, sonder besitzen auch eine dazugehörige View. Somit ist den Pluginerstellern die maximale Freiheit gegeben, wie sie ihren Viewer implementieren wollen. Um sicher zu gehen, dass der Viewer von *Screen* ableitet und somit von Caliburn geladen werden kann, leitet das Interface *IAssset Viewer* von *IScreen* ab.

5.7.1 Transparente Bilder

Abbildung 5.9 zeigt eine Funktion, die in den meisten Bildanzeigeprogrammen leider vergessen wird, das verändern der Hintergrundfarbe. Wäre der Hintergrund immer weiß oder schwarz, ohne die Möglichkeit zu wechseln, kann der Benutzer unmöglich sagen, ob das Bild transparent ist oder einfach nur die Farbe des Standardhintergrundes hat.

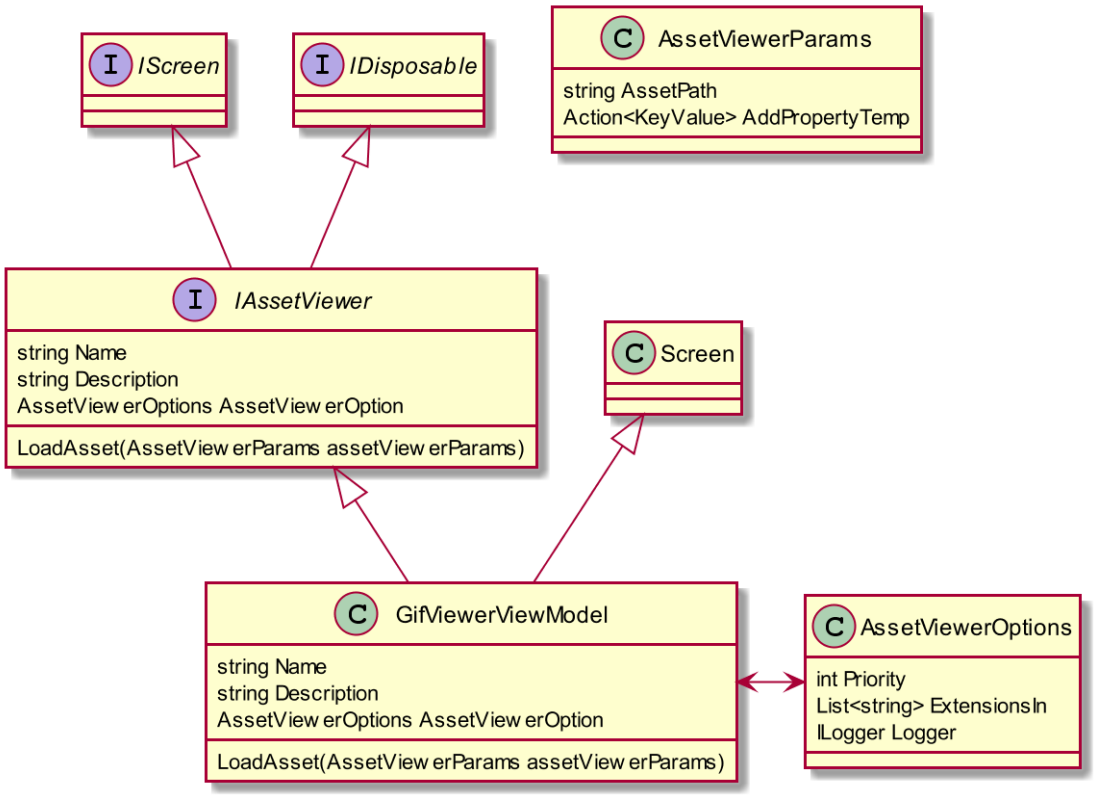


Abbildung 5.8: Prototyp Viewer

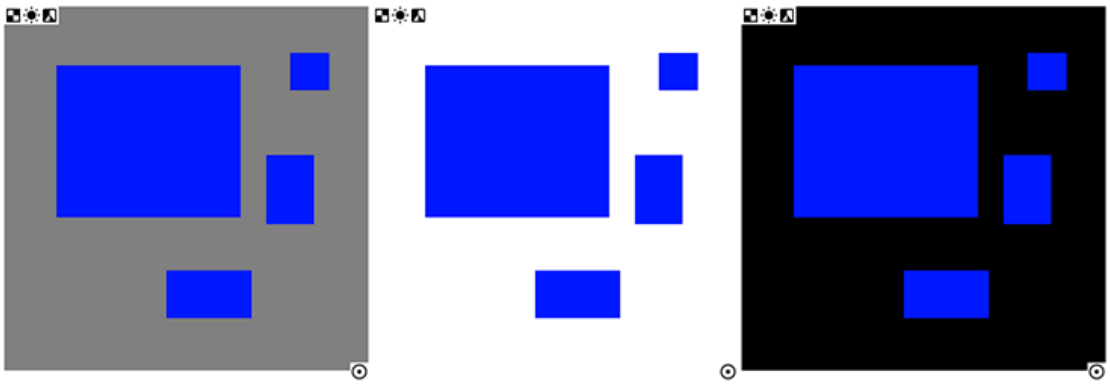


Abbildung 5.9: Transparenter Hintergrund

Kapitel 6

Schluss

Das Ziel dieser Bachelorarbeit war, wie in der Einleitung schon erwähnt, auf die verschiedenen Möglichkeiten der Asset Verwaltung einzugehen, die benötigten Funktionen eines PAMs für die Spielentwicklung zu evaluieren und darüber hinaus einen eigenen Lösungsansatz für ein PAM zu entwickeln. Im Kapitel 4 wurde auf die optimalen Funktionen, die ein PAM besitzen sollte, eingegangen, im Kapitel 3 wurde auf die generellen Vorteile eingegangen und im Kapitel 5 wurde der eigene Lösungsansatz vorgestellt.

Die Fragen, die in der Einleitung aufgeworfen wurden, sind somit beantwortet und der Lösungsansatz stellt einen Teil der evaluierten Funktionen zur Verfügung. Es war jedoch, wie von Anfang an erwartet, nicht möglich alle Funktionen zu implementieren, da selbst riesige Unternehmen nur einen Bruchteil unterstützen.

Generell bin ich sehr Zufrieden mit dem Prototypen und dem Ergebnis der Bachelorarbeit und ich freue mich drauf weiter daran zu arbeiten. Nächster Schritt wird sein eine Cloud Lösung für die Asset Verwaltung zu entwickeln. In der Cloud Lösung wird der Fokus mehr auf dem Asset, seinen Metadaten, der Suche und einem Ersatz für Ordnerstrukturen liegen. Funktionen die in dem Prototypen noch nicht inkludiert sind wie Sammlungen, Metadaten Sammlungen, Ratings, Links zu Assets teilen, Automatische Analyse, Versionierung und Benutzer/Rollen sollen noch hinzugefügt werden.

Anhang A

Fragebogen

Die Umfrage kann entweder Online [12] eingesehen werden oder mit den folgenden Abbildungen A.1, A.2 und A.3.

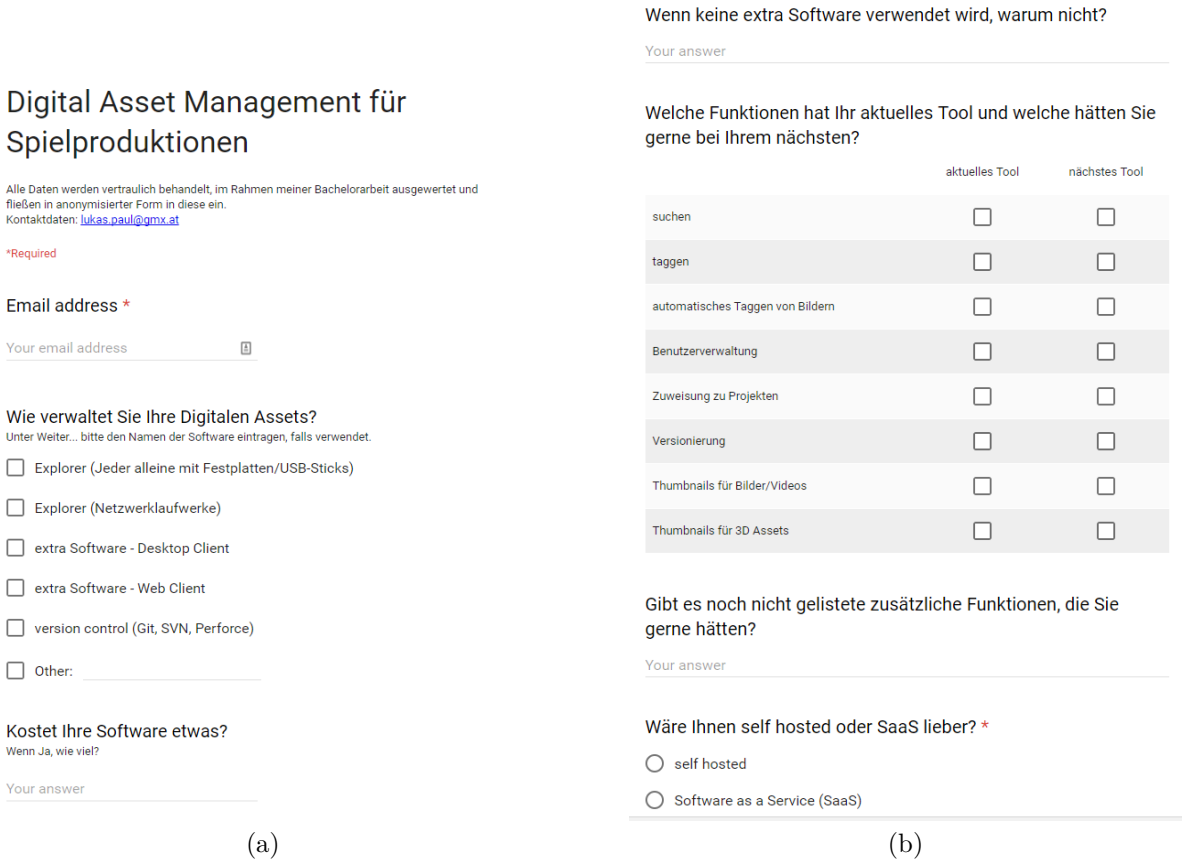
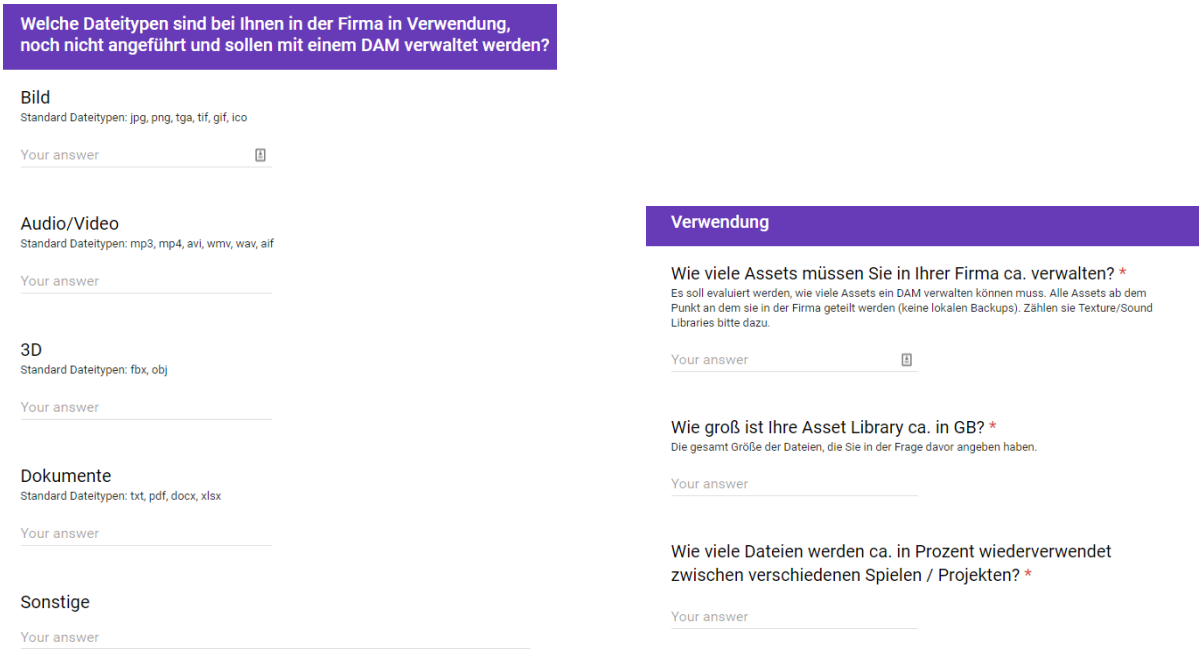


Abbildung A.1: Fragebogenteil 1



(c) (d)

Abbildung A.2: Fragebogenteil 2

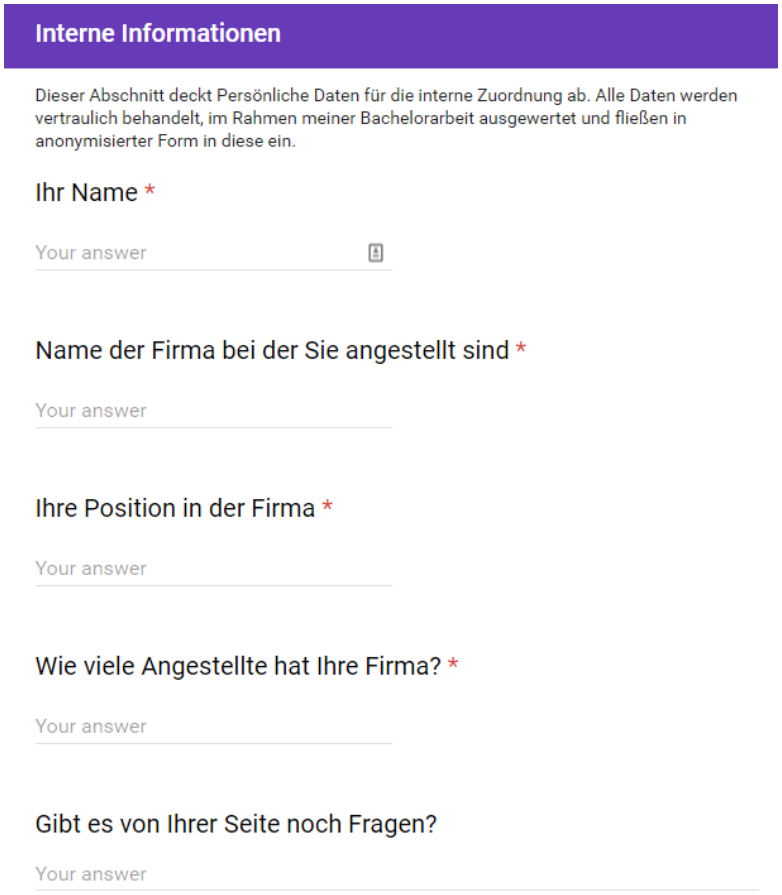


Abbildung A.3: Fragebogenteil 3

Quellenverzeichnis

Literatur

[1] John Gantz und David Reinsel. *The Digital Universe Decade – Are You Ready?* Techn. Ber. USA: IDC, Mai 2010. URL: https://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf (siehe S. 1).

[2] Jens Jacobsen, Tilman Schlenker und Lisa Edwards. *Implementing a Digital Asset Management System. For animation, computer games, and web development.* 1. Aufl. USA: Focal Press, 2005 (siehe S. 1–3, 7–10).

[3] Peter Krogh. *The DAM Book. Digital Asset Management for Photographers.* 2. Aufl. Canada: O'Reilly Media, 2009 (siehe S. 7, 10).

Audiovisuelle Medien

[4] URL: https://goo.gl/1TEpCq (siehe S. 16).

[5] URL: https://de.wikipedia.org/wiki/Polizeiausbildung_in_Bayern (siehe S. 17).

[6] URL: https://pixabay.com/en/fall-autumn-red-season-woods-1072821/ (siehe S. 17).

Online-Quellen

[7] *A Truly Generic Repository*. 2016. URL: https://cpratt.co/truly-generic-repository/ (besucht am 06. 01. 2018) (siehe S. 21).

[8] *Canto Preisliste*. 2018. URL: https://www.capterra.com/p/32191/Canto-Digital-Asset-Management/ (besucht am 09. 01. 2018) (siehe S. 11).

[9] *Hyper Preisliste*. 2018. URL: https://www.hypercms.com/cloud/home/02_Pricing/calculator.xhtml?option=calculator (besucht am 09. 01. 2018) (siehe S. 10).

[10] Peter Krogh. *Using Google Cloud Vision for OCR*. 2017. URL: http://thedambook.com/category/the-dam-book-3-sneak-peek/ (besucht am 01. 01. 2018) (siehe S. 17).

[11] *Nuxeo EA*. 2018. URL: https://www.nuxeo.com/customers/electronic-arts/ (besucht am 09. 01. 2018) (siehe S. 11).

[12] Lukas Paul. *Digital Asset Management Fragebogen*. 2018. URL: https://goo.gl/YjMKMo (besucht am 03. 01. 2018) (siehe S. 29).

[13] Chris Pratt. *THE TOP 20 MOST POPULAR DAM Software*. 2017. URL: https://www.capterra.com/digital-asset-management-software/?utf8=%E2%9C%93&v=4#infographic (besucht am 06. 01. 2018) (siehe S. 1, 10).

[14] *Shotfarm Preisliste*. 2018. URL: https://www.capterra.com/p/131745/Product-Content-Network/ (besucht am 09. 01. 2018) (siehe S. 11).

[15] *The Best Way to Name Your Files*. 2012. URL: https://getmethod.com/blog/2012/6/30/the-best-way-to-name-your-files.html (besucht am 07. 01. 2018) (siehe S. 14).